

Combining Classic Methods with Deep Learning Priors for Lightweight Visual Denoising



Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Clément Bled

Department of Electronic and Electrical Engineering
Trinity College Dublin

This dissertation is submitted for the degree of
Doctor of Philosophy

July 2025

Declaration

I declare that this thesis has not been submitted as an exercise for a degree at this or any other university and it is entirely my own work.

I agree to deposit this thesis in the University's open access institutional repository or allow the Library to do so on my behalf, subject to Irish Copyright Legislation and Trinity College Library conditions of use and acknowledgement.

I consent to the examiner retaining a copy of the thesis beyond the examining period, should they so wish (EU GDPR May 2018).

Clément Bled
July 2025

Acknowledgements

Firstly, I would like to extend my gratitude to my supervisor, Dr. François Pitié, for affording me the invaluable opportunity to pursue this research under his guidance. His support and profound expertise in image and signal processing have been a constant source of inspiration, motivating my work over the past four years. I am appreciative of the trust he has placed in my abilities and the encouragement he has provided throughout this journey.

To my Mum and Dad, Caroline and Didier, without whom this PhD would not have been possible. I would not be who I am today without them. They have always encouraged me to pursue whatever career I chose, and have always been supportive and understanding when I find myself most preoccupied. I will always appreciate the perspective on life they have given me.

To my brothers, Julien and Corentin: thank you for always looking out for me, no matter what.

To my friend and office-mate Vibhoothi: I have thoroughly enjoyed the years we spent together in our coveted, luxuriously quiet (and occasionally air-conditioned) office. Our daily intermissions of chats and rants provided essential distractions from work and were a highlight of the experience.

To the rest of my friends in the extended Sigmedia universe, past and present: Sophia Rosney, Ciaran Donegan, DJ Ringis, Darren Ramsook, Mark Anderson, Ed Storey, Meegan Gower, Ayushi Pandey, Zhaofeng Lin, Danilo Li, Varoun Hanooman, Jialiang Zhu, Udit Aurangabadkar, Conall Daly, Julien Zouein, Aris Papadopoulos, Shashwat Khandelwal, Illaria Fachi, Dee O'Regan and Victoria Ivanova. You have all made the lab a fun and vibrant place to work, and there has never been a dull moment with you all around.

To the wonderful academic and technical department staff—John Squires, Mark Linane, Eugene O'Rourke, Shane Hunt, Michael O'Riordan, Caroline Murphy, Harun Siljak, Arman Farhang, and Naomi Harte: thank you for always lending a hand when needed, whether it

was helping me repair my bike, unlock my office when I lose my keys, acquire new hard drives, or engage in spirited political debates.

To Gary Baugh, without whom I would not have had the opportunity to intern within Intel's NPU team. During my time there, you went above and beyond to expose me to various aspects of your team's work. Your kindness and enthusiasm for my work were greatly encouraging.

Abstract

Image and video denoising is a fundamental problem in signal processing, with applications spanning from video compression and smartphone image pipelines to cinema post-production and medical imaging. While traditional denoising techniques rely on carefully crafted priors to model natural image properties, modern approaches increasingly employ deep-learning-based models that leverage large-scale data to achieve state-of-the-art results. However, the trend toward large and computationally expensive architectures, such as transformers, raises significant challenges in terms of practicality and efficiency.

This thesis explores innovative approaches to denoising by combining classical signal processing techniques with modern neural networks to create lightweight, high-performance denoisers. We first develop a set of guidelines for designing compact, real-noise image denoisers, focusing on training strategies, architectural considerations, and the effective use of synthetic-real noise datasets. Building on these principles, we present a hybrid Wiener-CNN image denoising framework that combines the classic Wiener filter with small auxiliary neural networks, achieving comparable performance to popular denoisers like DnCNN while maintaining an order-of-magnitude reduction in parameters.

Extending this hybrid approach to video denoising, we design a 4D Wiener-CNN framework capable of leveraging temporal and spatial frequency information. This model outperforms state-of-the-art transformers in high-noise tasks and offers significantly faster denoising times, demonstrating the potential of lightweight architectures in video restoration.

Finally, we explore the use of implicit natural image priors modelled by denoisers to solve linear inverse imaging problems, building on the framework proposed by Kadkhodaie and Simoncelli. By fine-tuning the denoiser within the iterative process, we improve its performance across multiple tasks, including inpainting, super-resolution, and compressed sensing. These findings highlight the broader potential of integrating classical and modern methods to address longstanding challenges in denoising, offering a path toward efficient and adaptable solutions.

Table of contents

1	Introduction	1
1.1	Image and Video Denoising Applications Today	2
1.2	From Classic Signal Processing Methods to Deep Learning Denoising . . .	4
1.3	Recent Trends in Denoiser Design	5
1.3.1	Beyond Additive White Gaussian Noise	6
1.3.2	Increasingly Large Networks	7
1.3.3	Unlocking the Implicit Natural Image Prior Modelled by Denoisers	7
1.4	Key Observations and Research Questions	8
1.5	Thesis Outline	10
1.6	Contributions	13
1.7	Publications	14
2	Literature Review	15
2.1	Sources of Noise in Photography	17
2.1.1	Film Grain Noise	17
2.1.2	Noise in Digital Photography	20
2.2	Classic Image Denoisers	24
2.2.1	Early Spatial Denoising Filters	25
2.2.2	The Wiener Filter	29
2.2.3	Wavelet Denoising	32
2.2.4	Non-Local Priors	35
2.2.5	Variational Denoisers	36
2.3	Deep Learning Priors	40
2.3.1	Image Denoising CNN Architectures	40
2.3.2	Transformer Image Denoisers	45
2.4	Video Denoising	47
2.4.1	Deep Learning Video Denoisers	48

2.5	Training for Real Noise	51
2.5.1	Model-Based Noise Generation	51
2.5.2	Machine-Learning Based Noise Generation	52
2.6	Using Denoisers as a Universal Prior	53
2.7	Conclusions	56
3	Optimising Image Denoising Networks for Real Noise	59
3.1	Introduction	60
3.2	Proposed Testing Platform	62
3.2.1	Testing Architecture	62
3.2.2	Training Datasets	63
3.2.3	Evaluation Benchmarks	64
3.2.4	Training Regime	64
3.3	Ablation Studies	65
3.3.1	Balance of Noise Types in Training Data	65
3.3.2	Impact of Network Size	67
3.3.3	Use of Noise Maps in Image Denoising Networks	68
3.4	Training of Larger Networks and Final Results	72
3.4.1	Final Results	73
3.5	Discussion	75
3.6	Conclusions	79
4	The Wiener Image Denoising Network	81
4.1	A Review of the Baseline Wiener Filter	82
4.2	Optimisation of the Baseline Wiener Filter	84
4.2.1	Implementation Framework	84
4.2.2	Noise Gain Factor λ	87
4.2.3	Windowing and Padding	87
4.2.4	DC-Offset Removal	89
4.3	Automated Per-Block Noise Estimation	90
4.3.1	A Local AWGN Assumption	90
4.3.2	Estimating the Noise Variance	91
4.4	The Coring Refinement Network	93
4.5	Multi-Scale Wiener Filtering	95
4.6	Ablation Studies	95
4.6.1	Optimal Padding Method	95
4.6.2	Optimisation of Noise Gain λ	97

4.6.3	2D vs. 3D Wiener Filtering	98
4.6.4	Windowing Ablation Studies Results	98
4.6.5	Per-Block Noise Estimation	102
4.6.6	DC Offset Removal	103
4.6.7	Multi-Scale Wiener Filtering	104
4.6.8	Coring Refinement Network	105
4.7	Comparison with the State-of-the-Art	105
4.7.1	Visual Assessment	107
4.8	Discussion	109
4.9	Conclusions	113
5	Wiener Video Denoising Network	115
5.1	Image Denoising vs. Video Denoising	116
5.2	Wiener for Video Denoising	117
5.3	Optimisation and Enhancing of the Baseline	118
5.3.1	Windowing Optimisations and DC Offset Removal	119
5.3.2	Wiener Coring Refinement Network for Video	121
5.3.3	Blind denoising	122
5.3.4	Multi-Scale Wiener Filtering	123
5.3.5	Motion Compensation as a Preprocessing Step	123
5.4	Experimental Results	124
5.4.1	Training and Test Sets	125
5.4.2	From 2D to 4D: The Effects of Multi-Dimensional FFT on Video Denoising Performance	126
5.4.3	Number of Input Frames	126
5.4.4	Windowing and DC-Offset Optimisation	128
5.4.5	Multi-Scale Filtering for Video Denoising	132
5.4.6	Final Results	133
5.4.7	Additional Analysis: Motion Compensation	134
5.5	Discussion	135
5.6	Conclusions	139
6	Fine-Tuning the CNN Image Prior for Linear Task Solving	145
6.1	Solving Inverse Problems Using the Denoisers Implicit Prior	146
6.1.1	Plug and Play Priors	146
6.1.2	The Tweedie-Miyasawa Denoiser-Residual Relationship	148
6.1.3	Kadkhodaie and Simoncelli's Linear Inverse Solver	149

6.2	Method: Fine-Tuning Denoisers For Solving Inverse Problems	152
6.2.1	Differentiable Training Platform for the Denoiser	152
6.2.2	Fixing the Number of Iterations	153
6.2.3	Generalising the Denoiser via Multi-Task Training	153
6.3	Experimental Setup	155
6.3.1	Inverse Problem Tasks	155
6.3.2	Training/Testing Set	157
6.3.3	Choice of Baseline Denoiser	158
6.3.4	Hyperparameter Tuning for Fixed Number of Iterations	158
6.3.5	Training Details	158
6.4	Experiments	159
6.4.1	Exploratory Single-Task Experiments	159
6.4.2	Multi-Task Training Scenarios	160
6.4.3	Loss Weighting Scenarios	160
6.4.4	Length of Iterations	162
6.4.5	Beyond the Baseline	162
6.5	Results	162
6.5.1	Single Task Training	163
6.5.2	Multi Task Training	164
6.5.3	U-Net Performance.	167
6.5.4	Visual Assessment	167
6.6	Discussion	171
6.6.1	The Perception-Distortion Trade-off	171
6.6.2	Training Challenges	172
6.7	Conclusions	173
7	Conclusions	177
7.1	Summary of Thesis Contributions	177
7.2	Open Challenges in Denoiser Design	179
7.2.1	The Difficulty in Training for a Broad Range of Noise Intensities . .	179
7.2.2	The Difficulty in Training for Real Noise	180
7.2.3	The Perception-Distortion Conflict in Training Denoisers	180
7.2.4	The Difficulty in Quantifying Denoising Speed and Efficiency in High-Level Implementations	182
7.3	Final Remarks	182
	References	185

Chapter 1

Introduction

Image denoising is a classic, ill-posed problem which seeks to recover the original signal x from an observed, noisy signal y . Although real noise is typically signal-dependent, the problem is often simplified to take the form:

$$y = x + n, \tag{1.1}$$

where n is often modelled as additive white Gaussian noise (AWGN). Since Einstein's work on Brownian motion [1, 2] in the early 20th century, noisy processes have been recognised as an inherent characteristic of all recorded analogue signals. Consequently, the field of signal processing has long been concerned with the removal of noise from such signals. Early developments, such as the filters designed by Campbell [3] (1910) and Zobel [4] (1915), provided some of the first practical solutions through low-pass and band-pass filter implementations in telephone systems. These advancements laid the groundwork for noise suppression technologies that proliferated in audio and radio telecommunications, often relying on RC and LC circuits. The post-war declassified works of Wiener [5] and Shannon [6] introduced more sophisticated frameworks for noise suppression, including the Wiener filter and foundational principles of information theory. Since then, image and video denoising has grown into a significant field of research with numerous applications, supported by the development of a wide variety of priors over the years. More recently, the advent of machine learning techniques has revolutionised denoising by enabling data-driven approaches that outperform traditional methods, at a high computational cost.

1.1 Image and Video Denoising Applications Today

Image denoising remains a critical problem across numerous industries and research domains, with applications that are increasingly relevant in the era of digital media.

Denoising in the Video Compression Pipeline. One of the most significant applications of denoisers today is their integration within video compression pipelines, which play a key role in the widespread adoption of over-the-top (OTT) streaming platforms. As users migrate from traditional television services to OTT platforms, the demand for efficient video encoding has grown exponentially.

The popularity of streaming services such as Netflix and Disney+ has surged in recent years, amplifying the importance of optimising video delivery systems. As of 2023, Netflix [7] reported approximately 238 million subscribers worldwide, while user-generated content platforms like YouTube [8] boast over 2.5 billion monthly active users. Other prominent platforms include Instagram [9] with 2 billion active users, TikTok [10] exceeding 1 billion users, and Facebook [9] with nearly 3 billion users. The sheer scale of video transmission demands efficient encoding strategies to reduce bandwidth consumption and ensure seamless content delivery.

Denoising plays a vital role in this context by preprocessing image sequences to reduce frame entropy, enabling higher compression ratios at a given bitrate. Research has shown that among various artefacts, including flicker and shake, noise removal has the greatest impact on improving bitrate efficiency for video codecs [11]. For the transmission of analogue motion pictures, it has been demonstrated that removing film grain from scanned sequences can yield bitrate savings of up to 50% for sequences affected by heavy film grain [12]. Modern codecs such as AV1 [13] and H.264 [14] incorporate denoisers within their in-loop filters. AV1 employs a Wiener filter for this purpose, while H.264 utilises a deblocking filter that also performs denoising, highlighting the critical role of denoisers in video compression pipelines.

Image Denoising in the Smartphone ISP pipeline. Another industry where efficient image denoisers have gained increasing importance is the smartphone Image Signal Processor (ISP) pipeline. With each new generation of smartphones, consumers expect enhanced camera performance. However, packaging constraints limit the size of the lens and the imaging sensor, restricting the amount of light that can reach the sensor. To compensate, manufacturers rely on amplification circuits within the sensor silicon, which can inadvertently introduce

higher levels of noise. Consequently, the ISP pipeline plays a crucial role in processing the raw sensor data, reducing noise, and enhancing the overall image quality.

In recent years, smartphone manufacturers have integrated deep-learning accelerators into their devices, enabling advanced deep-learning models to work alongside traditional ISP pipelines to enhance image quality. These accelerators include Samsung's 4 nm Exynos 2400 [15], featuring a 17K MAC neural processing unit (also used by Google under the G4 [16] branding), and Apple's A17 chip [17], a 3 nm CPU manufactured by TSMC with 16 neural cores. Google has leveraged this hardware for features like 'Super Res Zoom' and 'Real Tone', while its 'Night Sight' mode [18] is specifically designed for denoising. Night Sight works by consolidating multiple raw frames captured in low-light conditions and applying a deep-learning model to reduce noise, significantly improving the clarity and detail of nighttime photos [19]. Similarly, Samsung's Night Sky Scene Optimiser [20] and Apple's Photonic Engine (formerly Deep Fusion) use AI to refine texture, enhance detail, and suppress noise. A key design consideration for implementing denoisers in smartphone hardware is the need for smaller, low-power models. This requirement underscores the importance of efficiency, rendering large, slow state-of-the-art models unsuitable for these devices.

Video Denoising for Cinema Post-Production. In the post-production industry, high-resolution raw video is intentionally processed without the typical ISP post-processing pipelines, allowing filmmakers to retain full creative control over the final image. This approach ensures the raw footage remains unaltered, enabling precise adjustments to contrast, colour grading, and other artistic elements during editing. As a result, high-quality denoisers are essential to remove noise from the raw data while preserving the original signal's contrast and avoiding artefacts that could disrupt the creative process. The processed output must meet exacting standards and withstand expert scrutiny at every stage of the editing pipeline.

While recent denoising networks demonstrate impressive performance on traditional distortion metrics such as PSNR, perceptual challenges often disqualify them for post-production workflows. Factors such as temporal consistency and the faithful preservation of dark tones are critical, however, deep learning models frequently struggle to reproduce these qualities. For example, the vanishing gradient problem during training can hinder the accurate reproduction of zero-valued pixels, which are essential for maintaining rich blacks in dark sequences. This limitation often leads to outputs with a slight shift toward grey, compromising the desired cinematic quality.

Several industry-standard tools incorporate denoisers tailored for post-production workflows. These include The Foundry’s compositing tool, Nuke [21]; Topaz DeNoise AI [22]; and Neat Video’s plug-ins [23], which are supported by major software suites such as DaVinci Resolve [24] and Adobe Premiere Pro [25]. While the specifics of these algorithms are often protected as intellectual property, Neat Video, a market leader, has disclosed the use of wavelet filtering techniques. These likely incorporate non-local block-matching methods similar to BM3D [26], reflecting the industry’s preference for robust, controllable denoising models that balance noise reduction with artistic precision.

Other Denoising Applications. As we have mentioned, the application of denoising in signal processing extends to virtually any domain that records analogue signals. Within the realm of imaging, denoisers find additional utility in scenarios involving small sensor sizes or challenging lighting conditions, both of which tend to generate higher levels of noise compared to standard cameras. Examples include action cameras used in extreme sports, smart doorbell cameras that must function reliably under variable lighting conditions, night-time infrared CCTV systems for security, and the emerging class of wearable cameras embedded in smart glasses. In each of these cases, effective denoising enhances image clarity, ensuring that captured content remains usable even under suboptimal conditions.

Denoising techniques are also widely utilised in medical imaging, where the accuracy and reliability of image data are critical. Although the dynamic range and number of channels in medical imaging data differ from those of traditional photographic images, the underlying noise models often exhibit similarities. Applications include MRI scans, CT scans, ultrasounds, and nuclear imaging, all of which benefit from noise reduction to improve diagnostic precision. By removing noise from such images, denoisers enhance the visibility of critical features, facilitating more accurate diagnoses, and reducing the potential for misinterpretation.

1.2 From Classic Signal Processing Methods to Deep Learning Denoising

Today, the state-of-the-art in image denoising is occupied by large, data-driven, deep-learning models. This marks a recent shift from the use of carefully designed priors that model the statistical properties of natural images, such as sparsity, smoothness, or self-similarity. These classic approaches relied on explicit assumptions about the underlying image structure, formalised through mathematical models. From a Bayesian perspective, these methods aim

to estimate the clean image x from the noisy observation y , which is expressed using Bayes' theorem as:

$$P(x|y) \propto P(y|x)P(x), \quad (1.2)$$

where $P(y|x)$ represents the noise model (likelihood) and $P(x)$ encodes assumptions about the clean image (prior). Popular methods based on these principles include the Wiener filter [27], which assumes Gaussian noise and signal distributions to minimise the Mean Squared Error, and wavelet thresholding [28], which leverages sparsity priors in the wavelet domain. Regularisation-based techniques, such as Total Variation [29] (TV), incorporate smoothness priors into optimisation frameworks, balancing noise suppression and edge preservation. BM3D [26], combines non-local self-similarity and sparsity through block-matching and collaborative filtering, representing the state-of-the-art before the advent of deep learning models.

The machine learning revolution of the mid-2010s was quickly adopted to image and video denoising, and by 2017, deep neural networks (DNN) had surpassed the performance of the best non-learning-based filters. This was demonstrated by image-denoising networks such as DNCNN [30] outperforming BM3D. Similarly, inspired by DNCNN, DVDNet [31], a video denoising network which outperformed BM4D as well as Bayesian video denoisers such as VNLB [32]. In recent years, the focus in research has shifted towards the network architecture design of these networks, which have become increasingly large in terms of parameters and memory usage. These new designs follow trends in convolutional neural network design, such as residual networks, U-Nets, multi-stage networks, attention modules and most recently, transformer networks. The most powerful image-denoising networks today are based around large U-Net architectures, for instance, the MWCNN [33] (Multi-level Wavelet) convolutional network, the SWIN (Shifted-WINDOW) transformer [34], and the Restormer transformer [35]. For video denoising, the landscape is presently similar, with transformer networks such as VRT [36] (Vision Restoration Transformer) commonly ranking highest in recent benchmarks.

1.3 Recent Trends in Denoiser Design

In this section, we discuss recently emerging trends in denoising literature, which will form the basis for our key observations and research questions.

1.3.1 Beyond Additive White Gaussian Noise

Although most recent research on image denoising focuses on network architecture, the choice and quality of training data are foundational to a network’s ability to generalise when deployed on unseen data. Until recently, machine-learning-based denoisers were primarily trained and tested on synthetic additive white Gaussian noise (AWGN). However, the introduction of real noise benchmarks [37, 38] has highlighted that denoisers trained on Gaussian noise perform poorly on non-Gaussian, real camera noise. Unlike Gaussian noise, real camera noise is signal-dependent and influenced by factors such as scene luminance, sensor temperature, ISO levels, and the proprietary in-camera processing pipeline.

While classical denoising methods offer flexibility through explicit priors (e.g., the noise power spectral density in Wiener filtering), machine-learning-based techniques rely on implicit priors learned from the provided training data. With real noise benchmarks being a relatively recent development, there is limited research on how to specifically optimise deep neural networks for real noise denoising. Moreover, the availability of real noise datasets [37, 39] is constrained by the labour-intensive process of acquiring high-quality noisy-ground-truth pairs. This typically involves photographing the same scene under varying ISO levels and lighting conditions, with the ground truth derived from averaging low-ISO images captured in well-lit conditions. Producing a sufficiently diverse dataset of thousands of scenes through this method is impractical. Consequently, Synthetic noise models have been developed to address this limitation, simulating real noise by reverting sRGB images to pseudo-raw states, applying signal-dependent noise, and reconstructing the noisy image through a simulated inverse camera ISP [40–42].

Recent advancements have also explored incorporating noise-specific features into network architectures. Examples include models that allow user-provided noise-profiles [31, 43], and networks that dynamically generate noise intensity maps through ancillary modules to guide the main denoising process [40]. However, evaluating the effectiveness of such features is complicated by the persistent trend toward increasing network size. As denoisers incorporate more features and parameters, it becomes challenging to disentangle the contributions of specific innovations from the overall gains due to larger model capacities. This underscores the need for a systematic approach to real-noise denoiser design—one that balances training data noise composition, novel noise-specific features, model size, and practical deployment constraints to deliver optimal performance for real-world applications.

1.3.2 Increasingly Large Networks

Recent developments in state-of-the-art denoiser design reveal a clear trend toward increasingly large models dominating benchmarks. For instance, the original DnCNN network [30] (2017) contains fewer than 1 million parameters, while recent transformer-based models such as SwinIR [44] (2021) can be configured to exceed 100 million parameters. Although these large networks achieve impressive performance, they are often impractical for real-world applications, requiring high-memory machines to process even moderately sized images. Moreover, subsequent models often deliver only marginal gains in output quality at the cost of millions of additional parameters.

This class of denoiser has undoubtedly outperformed classical methods in output quality; however, its design has remained focused on architectural innovation, a computationally expensive approach. This data-driven, black-box design philosophy frequently overlooks the integration of domain knowledge into network architectures. In convolutional neural networks, the network’s understanding of images is limited to inherent inductive biases, including assumptions of locality (features extracted from small patches defined by the kernel), translation invariance, and the stationarity of image properties. In contrast, classical methods are grounded in explicit, well-defined priors that encode strong domain knowledge. This disparity underscores a gap in research: the opportunity to combine the robustness and efficiency of classical methods with the flexibility and power of modern deep-learning approaches, rather than abandoning computationally efficient, well-established techniques.

1.3.3 Unlocking the Implicit Natural Image Prior Modelled by Denoisers

Recently, significant efforts have been made towards leveraging the complex natural image priors implicitly modelled by denoising networks for generative tasks. These 7 trace their origins to pre-deep-learning ‘plug-and-play’ methods, which access the prior indirectly by using a denoiser as a regularisation tool to guide the solution of linear inverse problems such as medical imaging reconstruction, super-resolution, deblurring and deblocking. More recently, a direct link between the denoiser and its image prior has been made through the work of Kadkhodaie and Simoncelli in the paper *Solving Linear Inverse Problems Using the Prior Implicit in a Denoiser* [45]. The authors outline a rediscovered proof connecting the denoising residual, $y - \hat{x}$, (where $\hat{x}$ is the estimated clean image), to the gradient of the log-likelihood of the image. By iteratively applying denoising and renoising operations to an observation degraded by a known linear operator (e.g., downsampling, inpainting, or

blurring), the authors demonstrate that it is possible to reconstruct the original image x via a gradient ascent procedure.

The proof outlined by the authors also serves as the linchpin for diffusion models, which adopt a similar gradient-based approach but with a physics-inspired non-equilibrium thermodynamics framework. Unlike Kadkhodaie’s method, which directly utilises a pre-trained denoiser optimised for AWGN under an MSE loss, diffusion models require a network to be trained specifically for the diffusion process to learn the backwards denoising stage.

In Kadkhodaie’s linear inverse solver, the framework is designed to allow solutions to converge through an unconstrained number of iterations, relying on smoothness metrics as a threshold for stopping. While not as computationally intensive as diffusion models, this approach raises questions about the optimality of the natural image prior modelled by a given denoiser for various linear inverse tasks. An off-the-shelf denoiser trained for a generic denoising task may not inherently provide the most suitable prior for specific applications within a linear inverse solver, suggesting the need for further exploration of task-specific adaptations.

1.4 Key Observations and Research Questions

As we study the current landscape in image denoising, a number of observations emerge that motivate the work in this thesis. The field has recently shifted towards real-image denoising, driven by the availability of online benchmarking systems [37, 38]. This shift, coupled with the advent of powerful, high-parameter CNN and transformer network architectures [34, 40, 41, 46–48], has led to a proliferation of computationally demanding networks seeking the best-in-class performance on these benchmarks. Much of the focus in denoiser design has been directed towards maximising benchmark performance through increasingly large models, with less attention given to other critical factors, such as the composition of noise in the training datasets and the integration of denoising-specific features such as noise prediction modules.

In parallel, the rapid and frequent release of state-of-the-art models in related restoration tasks such as image super-resolution [49], and broader fields of computer vision such as classification [50], has encouraged research to focus on adapting these models for image denoising. This deep-learning-centric focus has led to a relative neglect of classic, well-established priors, which contain valuable domain knowledge. The absence of these classic methods in modern denoiser design represents a missed opportunity to explore collaborative, lightweight solutions that integrate both paradigms.

In the context of video denoising, the application of neural networks demands additional care due to the increased volume of data handled, and the impracticality of applying large CNN or transformer networks, often containing tens of millions of parameters, to each frame. Many state-of-the-art transformer networks are unable to perform inference on high-resolution input sequences without compromise, such as processing frames in smaller sub-blocks within a loop on high-memory GPUs. This limitation again highlights the need for more efficient lightweight networks that can leverage temporal information, potentially through techniques such as localised frequency analysis, to process multiple frames more efficiently.

Finally, Kadkhodaie’s [45] implicit prior framework highlights the opportunity to fine-tune denoisers for specific tasks, leveraging the implicit prior they model to solve linear inverse problems more effectively. This perspective opens up avenues for optimising denoisers to better suit particular applications, as well as for accelerating the iterative process within such frameworks.

Research Questions

We summarise the research questions that motivate this thesis as follows:

- RQ1.** What components of recent neural network architectures, such as CNNs and transformers, are most effective for handling real noise in terms of both performance and computational efficiency?
- RQ2.** What are the potential benefits and challenges of designing a hybrid denoiser that combines classical priors with deep neural networks, and how might such a design improve performance or efficiency in real noise applications?
- RQ3.** Can a hybrid video denoising algorithm, which leverages both temporal information and spatial frequency analysis, be developed to compete with or surpass the current state-of-the-art in video restoration tasks?
- RQ4.** How can Kadkhodaie and Simoncelli’s implicit prior method for solving linear inverse problems be optimised through targeted fine-tuning of the denoiser, and what improvements can be achieved across a range of linear tasks such as inpainting, deblurring, and super-resolution?

1.5 Thesis Outline

In this thesis, we develop a number of different denoising algorithms, with the aim of minimising computational cost while maximising output quality. In our methods, we consider both classical signal processing techniques and the current deep-learning-led landscape to design the most cost-effective denoiser which takes advantage of both methods. As well as image denoising methods, our work extends to video denoising, and the use of image priors as universal solvers in linear imaging problems.

In our first method, we establish a series of best practices which should be considered when designing compact networks for real image denoising, from architectural design choices to the make-up of the training data. In Chapter 3 we present this work and compare a simple denoising network to the state of the art for real image denoising.

Following the first method, in our second approach, we put forward a lightweight, hybrid denoising system which combines a classical denoising technique (the Wiener filter) with small ancillary networks to produce a cost-effective denoising technique which performs comparably to popular image denoisers in real image denoising. This design is put forward in Chapter 4 and extended in Chapter 5 to video denoising.

Lastly, in Chapter 6, we present our final method, which builds on recent works presented by Kadkhodaie and Simoncelli [45], which outlines how the implicit image prior modelled by deep-learning denoiser may be used to solve linear imaging problems. In this method, we fine-tune denoisers within an iterative framework for enhanced global image restoration.

In the following sections, we summarise each chapter.

Chapter 2: Literature Review

This chapter provides a comprehensive review of the denoising problem, briefly tracing its evolution from early theoretical foundations to contemporary applications. It highlights the transition from initial studies on noise and filtering to modern applications in image and video processing. A critical examination of state-of-the-art image and video denoising techniques reveals the increasing size of modern networks and their limited integration of domain knowledge. Deep-learning-based denoisers, often characterised as "black boxes," rely heavily on optimising large numbers of trainable parameters, learning the problem largely from scratch.

The chapter also explores the noise generation process within camera sensors, providing a foundation for understanding the creation of synthetic real-noise datasets. Current methodolo-

gies for generating such datasets are reviewed, along with recent advancements in uncovering the previously ill-defined natural image priors captured by deep denoisers. These priors are shown to offer potential solutions to general linear imaging constraints.

In conclusion, several shortcomings in denoising network design are identified. These include the ever-increasing size of state-of-the-art networks, challenges in achieving the correct synthetic-real noise balance in denoising datasets, the absence of user-controllable parameters in large networks, and the lack of high-performing blind video denoisers in recent benchmarks.

Chapter 3: Optimising Image Denoising Networks for Real Noise

In this chapter, we examine recent developments in the field of real-image denoising and evaluate the impact of these features (impact of noise type used in training, network size, use of noise maps in training) through a series of ablation studies. Specifically, we focus on the increasing number of parameters in state-of-the-art denoising networks and investigate whether their performance can be attributed to architectural design, network size, or a combination of both. To this end, we introduce a simple baseline network to systematically assess the effects of key factors, including the type and balance of noise in the training data, the influence of network size on denoising performance, and the utility of incorporating noise maps within denoising models.

Additionally, we apply our training scheme to more complex networks to evaluate the impact of increased model complexity. Based on these analyses, we present a series of recommendations for implementing real-noise denoisers. A key observation from this chapter is that as network sizes continue to grow, they become increasingly difficult to train and require significant memory, which limits their applicability to certain devices.

In response to these challenges, we advocate for a shift in design philosophy, revisiting classic denoising priors underpinned by established domain knowledge. We propose leveraging small networks as complementary tools to enhance the denoising process, rather than relying on them as the primary backbone.

Chapter 4: The Wiener Image Denoising Network

In this chapter, we develop a new denoising network from the ground up for real-image denoising and evaluate its performance against popular and state-of-the-art image denoising methods. Building upon the classic Wiener filter, we implement a differentiable filter within a deep-learning framework, ensuring that every aspect of the filter is optimised for both output

quality and denoising speed. This work is motivated by a lack of lightweight denoisers in recent literature.

During the design process, we incorporate small auxiliary convolutional neural networks to enhance the denoising process. Additional modules, including automatic noise estimation and filter refinement components, are integrated to further improve performance. The chapter concludes with the presentation of a competitive denoising solution that achieves comparable results to popular networks, such as DnCNN, while requiring significantly fewer training parameters.

Chapter 5: The Wiener Video Denoising Network

This chapter serves as a natural extension of the Wiener image denoising network, incorporating a complete redesign to address the additional dimensions inherent in video data while maintaining a lightweight denoising solution. Both the Wiener filter and the ancillary networks are re-optimised, and additional ablation studies are conducted to evaluate key factors, including the optimal number of input frames, the impact of motion compensation on denoising performance, the effect of increasing FFT dimensionality, and the introduction of trainable analysis and synthesis windowing.

The final implementation demonstrates superior performance in SSIM across all tested denoisers and is only marginally surpassed in PSNR (on low-noise tasks) by significantly larger and slower networks, such as the VRT transformer. Unlike the tested state-of-the-art denoisers, we also evaluate a blind version of our model, which eliminates the need for users to estimate noise variance or select a model trained for a specific noise level. This blind implementation outperforms all tested models in SSIM while maintaining efficient denoising times.

Chapter 6: Extending Image Denoisers for Linear Inverse Problems

In our final technical chapter, we build upon the recent work of Kadkhodaie and Simoncelli [45], which extends the applicability of AWGN image denoisers by leveraging their implicit natural image priors to solve linear inverse imaging problems. Their approach builds on a rediscovered result, attributed to rediscovered proofs by Miyasawa [51] and Tweedie [52], which establishes that the denoising residual of a blind denoiser is proportional to the gradient of the log-likelihood of the image. By iteratively applying the residual in a gradient ascent framework, starting from an image of pure noise, a plausible image can

be synthesised. Moreover, by introducing constraints via a linear measurement matrix and an inverse projection matrix, the algorithm can recover a degraded image by progressive denoising and using the gradient of the log-likelihood of the image prior.

In this chapter, we contribute to this framework by developing a method to fine-tune the denoiser within the inverse problem-solving process, enhancing its ability to generalise to other tasks. We evaluate the fine-tuned network across a range of tasks, employing various loss functions, and compare its performance on both trained and untrained tasks, including denoising. Our results demonstrate success in improving the global mean performance across tasks for the fine-tuned models.

However, a visual assessment of the results leads to a secondary conclusion. We observe that the process of converging on a high probability solution for x may not align well with classical distortion metrics such as MSE and PSNR. During training, traditional loss functions penalise the network for synthesising image content, a phenomenon associated with the perception-distortion trade-off [53]. This observation highlights an intriguing opportunity for future research into training strategies that balance perceptual quality and distortion metrics more effectively.

1.6 Contributions

In this thesis, we make several key contributions, summarised as follows:

1. We establish a set of guidelines outlining best practices for designing real-noise image denoisers.
2. We propose an enhanced DnCNN-based denoising network optimised for blind denoising of real-noise images.
3. We develop a differentiable implementation of the 3D Wiener filter suitable for GPU-accelerated frameworks. The design of the filter is such that all filtering occurs in parallel.
4. We introduce a lightweight, hybrid, blind Wiener-CNN denoising platform that uses the Wiener filter as a backbone and incorporates small ancillary networks to assist the denoising process. This includes a per-block noise estimation network and a coring refinement network.
5. A second, 4D hybrid Wiener-CNN network is implemented for video denoising, producing a denoiser that outperforms state-of-the-art transformers in high-noise tasks

while demonstrating rapid denoising times. We implement both blind and non-blind versions of the denoiser, introducing novel trainable analysis and synthesis windows, as well as new coring refinement network architectures.

6. We optimise the iterative linear inverse solving framework proposed by Kadkhodaie and Simoncelli [45] to converge in far fewer iterations while maintaining output performance.
7. We demonstrate that the denoiser used within the linear inverse solver proposed by Kadkhodaie and Simoncelli can be fine-tuned within the framework to improve output quality across multiple linear imaging tasks. This process involves simultaneously fine-tuning the same denoiser model for several tasks and balancing each loss function using a custom weight equalisation scheme.

1.7 Publications

Sections of the work in this thesis appear in the following publications:

1. Clement Bled and Francois Pitie. “Assessing advances in real noise image denoisers”. In: *Proceedings of the 19th ACM SIGGRAPH European Conference on Visual Media Production*. 2022, pp. 1–9
2. Clément Bled and François Pitié. “Pushing the Limits of the Wiener Filter in Image Denoising”. In: *2023 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2023, pp. 2590–2594
3. Clément Bled and François Pitié. “Lightweight Video Denoising Using a Classic Bayesian Backbone”. In: *2024 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE. 2024, pp. 1–6

Chapter 2

Literature Review

The removal of noise from a recorded signal is a historically rich, ill-posed problem that has remained a focus of signal processing research since it first gained attention during the domestic adoption of telephones and radios in the early 20th century. In 1905, Albert Einstein published [1] his explanation for Brownian motion, proposing that the movement of pollen suspended in a fluid can be explained by smaller, unobservable particles moving through the fluid in all directions. His model related this movement to the fluid's properties and the fluid's temperature, showing that the pollen's mean-square distance travelled by the pollen was proportional to time. This publication strongly endorsed atomic theory, supporting the existence of the atom before it could be observed.

By 1928, Einstein's model was adapted by engineers Harry Nyquist and John B. Johnson of American Telephone and Telegraph (AT&T) Research Labs (consolidated into Bell Labs in 1934) to model thermally generated noise in conductors [57, 58]. In Nyquist's foundational paper, *Thermal Agitation of Electric Charge in Conductors*, the model for thermal noise in charge carriers was defined as:

$$\overline{v_n^2} = 4k_B T R, \quad (2.1)$$

where $\overline{v_n^2}$ is the voltage variance, or noise power spectral density, k_B is the Boltzmann constant, T is the absolute temperature of the resistor, and R is the resistance value. This model was subsequently used to model noise in the information theory works of Claude Shannon (also of Bell Labs), *A Mathematical Theory of Communication* [59] and *Communication in the Presence of Noise* [6], published post-war in 1948 and 1949, respectively. These works introduced noise as a random variable, described the maximum channel capacity for noisy signals for a given error rate, and proposed encoding and decoding schemes to mitigate errors caused by noise.

Wiener, a teacher of Shannon's, published notable works on information theory [60], and stochastic signal processing [5]. The latter introduced the declassified Wiener filter, which reduces the noise of a signal by comparing it to the estimated noiseless signal. Other efforts in reducing noise include the long-awaited successful implementation of frequency-modulated (FM) radio [61] in 1936. In the 1960s, high-quality denoisers for analogue audio recordings came into focus. Dolby became the industry leader in hiss-free tape recordings [62], and Wiener-inspired, power spectrum filtering of speech recordings was developed by Berouti [63].

In the early 1990s, the digitisation of film for motion picture production was brought about by scanners produced by Kodak [64, 65], Cintel [66], Imagica [67] and Oxberry [68, 69]. For the first time, degradations such as noise, dust, and tears could be addressed digitally, eliminating the labour-intensive process of physically treating the film. In 1993, Kodak completed the first full digitisation and restoration of a feature film by scanning all 119,500 frames of Disney's 1937 *Snow White* [70].

With the advent of digital image processing, the removal of noise from images became a widely studied problem, and a range of methodologies and philosophies have been employed to best approximate the original unseen image signal, x , from an observed noisy source, y . To arrange the literature review of this body of work, it is convenient to frame the problem in the Bayesian setting, as it allows us to separate denoising methods by their various assumptions. Baye's rule is used to express the posterior distribution of the clean image, $p(y|x)$, as a function of the likelihood $p(y|x)$, the image prior $p(x)$ and the marginal probability $p(y)$:

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)}, \quad (2.2)$$

The likelihood encapsulates our knowledge of the noise model, which in turn depends on the imaging method. We present an overview of proposed models of film grain noise in Section 2.1.1 and modern digital sensors in Section 2.1.2.

In Section 2.2, we review popular denoising methods by their priors. Firstly, early spatial smoothness priors such as the box filter, Gaussian filter and Bilateral filter are covered in Section 2.2.1. Next, we consider the additive Gaussian noise prior of the Wiener filter in Section 2.2.2. Priors which exploit sparse representations, such as the Wavelet filters and Singular-Value Decomposition (k-SVD), are considered in section 2.2.5 and priors which exploit repeated patterns in natural images, such as non-local means denoising, are discussed in Section 2.2.4. The state-of-the-art priors of neural networks, which implicitly learn natural

image priors and dominate denoising literature today in terms of output quality, are reviewed in Section 2.3.1, and this is extended to video denoising in Section 2.4.

Recent works [45] have explored innovative ways to leverage the highly effective natural image priors inherent in deep learning models, taking the relationship between noisy images and their denoised counterparts in a new direction. A foundational result by Miyasawa [51] demonstrated that the residual of a denoised image, $n = y - \hat{x}$, represents the gradient of the log image prior as $n = -\sigma^2 \nabla_y \log p(y)$, where σ^2 is the noise variance. This insight establishes a powerful framework for using denoisers as priors to iteratively generate plausible real images from random noise. The process involves successive denoising and re-noising steps, each advancing toward the direction that maximises the image prior.

This concept has been indirectly realised in the highly successful stable diffusion models [71]. Additionally, denoisers have been directly applied as general-purpose solvers for image enhancement tasks, exemplified by the universal inverse solver of Kadkhodaie and Simoncelli [45] and the IRCNN framework of Zhang [72]. These models, which we discuss further in Section 2.6, provide a foundation for maximising the utility of our denoiser.

2.1 Sources of Noise in Photography

Camera noise is unique from sensor to sensor, and it has been well established that raw photographs may be used to identify a camera alone [73–75]. Many denoisers, approximate camera noise to be additive white Gaussian noise (AWGN), such that $y = x + n$, where $n \sim \mathcal{N}(0, \sigma^2)$, which simplifies the likelihood function, $p(y|x)$ to a Gaussian distribution. This noise model is not strictly true for real camera sensors, however - failing to account for the characteristics of real noise when modelling image noise can lead to sub-optimal denoising performance. This is the case for the Wiener filter (Section 2.2.2) and additional measures, such as patch processing, should be taken to account for this. In the following section, we discuss the origins of noise in analogue and digital cameras, to better model their distributions in our work.

2.1.1 Film Grain Noise

Before digital camera sensors became widely available, the removal of film grain from analogue photography was a primary area of research [76–78] in image processing literature. This grainy texture cannot be avoided in the capture and development process as the light-sensitive grains themselves are fundamental to the imaging method.

The Chemistry of Film Grain Noise. Analogue photography relies on the reaction of chemicals in the photographic emulsion to light. The emulsion typically consists of silver halide crystals [76, 79] (such as silver bromide, silver chloride or silver iodide) suspended in a gelatin binder which is coated onto a plastic base to create the film. When exposed to light, the silver halide decomposes into elemental silver to create a latent image.

Development begins with the film being submerged in a developer solution, which contains a reducing agent, such as hydroquinone. This reduces the exposed crystals to metallic silver, at which point the negative image becomes visible. To prevent further reduction, the film is then placed in an acid stop bath to neutralise the alkaline developer. Next, the images are treated with a sodium thiosulfate solution to "fix" the image and dissolve the remaining unexposed silver halide crystals, leaving only the metallic silver image. As the name suggests, light regions in the original scene appear dark on the negative, while dark regions appear light. The negative film may then be scanned and converted to a positive image digitally. Before digital scanners, final prints were produced by projecting the negative image onto photographic paper designed to produce a positive image, which is subsequently redeveloped.

The initial distribution of silver halide in the photographic emulsion is random, meaning that the distances between grains are not constant. Both the exposure and development processes are stochastic, with not every silver grain reacting identically to the same levels of exposure or chemical treatment. During development, some grains of silver halide clump together in the negative. This uneven distribution of developed crystals is known as film grain noise. Figure 2.1 illustrates examples of film photography with varying levels of film grain, affected by factors such as ISO and development conditions.

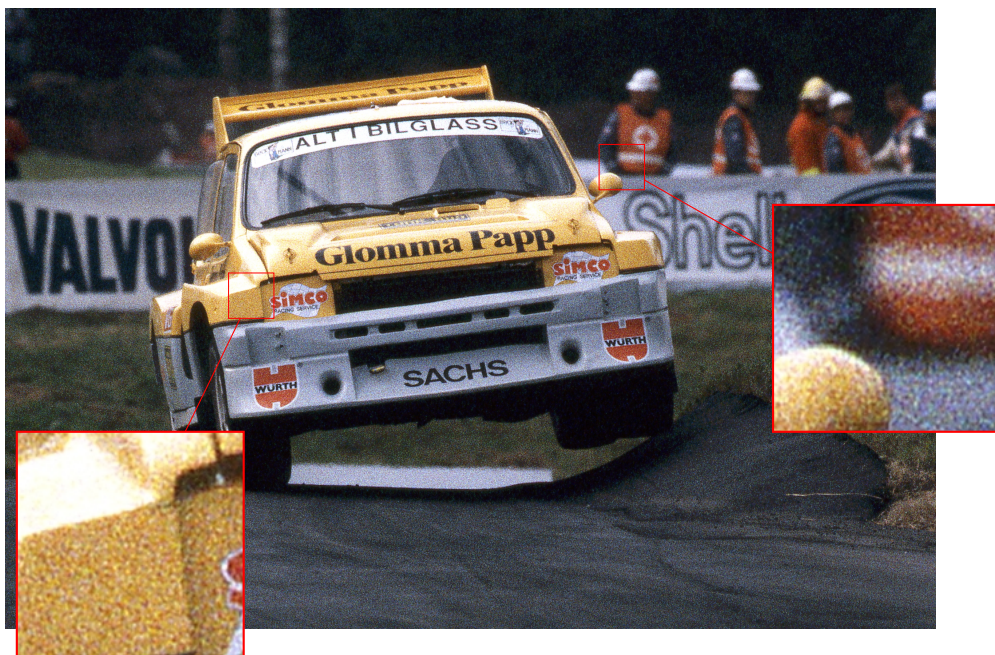
The characteristics of this noise have been shown to depend on grain size [80], grain density [81], and grain geometry [82]. While grain noise can be reduced by using finer grains, this comes at the cost of reduced light sensitivity in the film. Other factors influencing the pattern of film grain include film speed (light sensitivity/ISO), colour, and development conditions (e.g., temperature, chemicals, and agitation). For instance, high-ISO film uses larger silver halide crystals to increase light sensitivity, making the grain more prominent.

Film Grain Noise Models are typically signal-dependent and non-stationary [83]. They are commonly based on an auto-regressive (AR) model [84–86]:

$$N_f(i, j) = \sum_{m=0}^L \sum_{n=0}^L a_{m,n} N_f(i-m, j-n) + N_s(i, j), \quad (2.3)$$



(a) Low ISO Film used for still landscape scene: Killarney National Park shot on Olympus 35RC with Kodak Ultra Max 400 ISO 35mm Film. Photo by Corentin Bled.



(b) High ISO Analogue Film used for dynamic scenes: Shot on AGFACHROME 1000 ISO film. Photo by Eddi Laumanns, 1991. Licensed under GNU Free Documentation License and CC BY-SA 3.0.

Fig. 2.1 Examples of different film grains produced by low and high ISO film. We note the larger grains/crystals in the high ISO image.

where $N_f(i, j)$ is the film grain noise, L is the filter order, $a_{m,n}$ are the AR coefficients and $N_s(i, j)$ is the noise excitation, whose variance is signal-dependent. This AR causal model depends only on passed grain values in the raster scan, which is convenient for rapid noise modelling in the case of noise synthesis [83, 86]. As the spatial dependence of the noise reduces quickly to zero with increased spatial distance, the order of these models is often small, such as 3×3 neighbourhood. The noise variance and signal intensity are assumed to be the same in this small neighbourhood, and as such the AR coefficients are derived by Yule-Walker [84] equations, with approximated spatial correlation coefficients. The noise is temporally and colour-independent.

Disregarding the signal dependence has been shown to negatively impact denoising performance [76]. Some techniques have been developed to denoise signal-dependent film grain noise, such as Bayesian optimal estimators [87, 88] (MMSE, MAP, ML) and homomorphic transformations [89], coupled with denoisers such as the Wiener filter [90, 91] (See Section 2.2.2).

Film Grain Noise Synthesis is an integral part of the post-production pipeline for film-based projects, as computer-generated (CG) effects, which introduced new elements or frames (interlacing), did not match the original film texture. Film grain is reintroduced to avoid spatial and temporal inconsistencies in these cases [92]. Film grain synthesis has also recently become part of video transcoding pipelines [12, 93, 94], such as the AV1 codec [95], especially in the decoding of classic motion pictures which have been denoised to improve the encoding efficiency. Especially in such cases where the original material was shot in analogue, film grain is considered to be an important part of the viewing experience.

2.1.2 Noise in Digital Photography

The first digital imaging sensor was developed by Bell Labs in the late 1960s utilising CCD (Charge-Coupled Device) arrays [96, 97]. No longer constrained by film emulsion, digital image sensor quality began to surpass its analogue counterpart as pixel densities increased. In addition to the improved image quality, digital cameras benefited from instant image previews and convenient, interchangeable storage. While cheaper CMOS (Complementary Metal-Oxide-Semiconductor) sensors also existed, it was not until the early 2000s that their advancements saw digital cameras become affordable enough to overtake the market share once occupied by analogue film cameras.

A digital camera sensor is comprised of a silicon photosensor array, with individual wells, or pixels, which accumulate photoelectric charge [98]. After exposure, the captured charge in

each well is amplified according to ISO gain, converted to a voltage by an ADC (Analogue to Digital Converter) for measuring, quantised and taken off the sensor silicon or ‘read-out’, to the ISP (Image Signal Processor) where the raw image data undergoes post-processing. While CMOS sensors take advantage of their transistor composition to include a preamplifier (and sometimes the ADC) at each pixel, CCD sensors amplify a single row in the array at a time and are thus slower.

The ISP Pipeline. As indicated in Figure 2.2, the raw image is typically received by the ISP as a single-channel image, in a Bayer mosaic, with quarter resolution for the red and blue channels, and half resolution for the green channel. An optical black adjustment is made, where closed-shutter current (dark current) is corrected to ensure that black levels are accurate and hot pixels are corrected using a look-up table (LUT). The image then undergoes demosaicing (or debayering) to generate the full-resolution image for all three colour channels. Following this, white balancing is applied for colour constancy. This step is necessary as the human visual system (HVS) perceives whites similarly, despite different sources of radiance (incandescent, fluorescent, daylight) producing different RAW sensor readings suggesting the colours to be different. White balancing techniques include the gray world assumption [99], white patch method [100] as well as more recent deep learning approaches [101–103].

The image is then converted from the native sensor colour space to an unrendered canonical colour space, such as CIE XYZ. This conversion places the image in a colourimetric space that aligns with the human visual system (HVS), ensuring consistent colour representation across devices. In this space, gamma correction is applied to compensate for the non-linear response of display hardware. Additional adjustments, such as colour balance, contrast, and saturation, may be applied before converting the image to a rendered colour space like sRGB (for images) or Rec.709 (for video). During this stage, the dynamic range is typically compressed from the original 12–16 bits per channel to 8 bits. The image may also undergo tone mapping or artefact reduction to optimise visual quality. The final output can be saved in a still-image format, either uncompressed (e.g., TIFF) or compressed (e.g., PNG, JPEG), or encoded into a video format. For video, frames are stored sequentially in a container such as MP4, MOV, or AVI, often using lossy compression codecs like H.264 or H.265, which exploit temporal and spatial redundancy to reduce file size while preserving perceptual quality.

Due to the proprietary nature of imaging sensors and ISP pipelines, every digital camera possesses a unique noise profile. Generalised models exist which describe the main sources of noise [104–107] produced by imaging sensors. Noise in photographs is not a product of

one single process but rather several processes occurring at different stages in the capture pipeline relating to thermal noise, quantum uncertainty and manufacturing non-uniformities across the sensor array. This includes photon shot noise, dark noise, read noise and fixed pattern noise.

Photon Shot Noise is the first source of noise to be introduced in the capture pipeline, caused by variations (quantum uncertainty) in the arrival rate of incident photons on each photodiode [108–110]. It can also be described as the variation in flux a pixel experiences from its neighbourhood mean during exposure. Photon noise is modelled by Poissonian distribution as the process counts uncertain discrete events (the random arrival times of photons) in a fixed period of time. It is the only source of digital camera noise originating from outside the sensor silicon and is exposure (signal) dependent. The signal S , in terms of electron charge, is described by the equation:

$$S = QE \cdot \Phi \cdot t, \quad (2.4)$$

where QE is the sensor quantum efficiency, Φ is the average photon flux in photons/sec and t is the exposure time in seconds. According to Poissonian statistics, the noise standard deviation is equal to the square root of an average number of observations:

$$\sigma_S = \sqrt{S} = \sqrt{QE \cdot \Phi \cdot t}, \quad (2.5)$$

where σ_S is the photon noise standard deviation in terms of input charge. As we see in Equation 2.5, photon noise increases with the signal; however, the relative magnitude of the noise, compared to the signal, decreases as the signal increases, which is evident by the signal-to-noise measurement: $SNR = S/\sqrt{S}$. This SNR measurement defines the fundamental limit in the quality of an ideal, noiseless camera. Photon shot noise is spatially independent but exposure (signal) dependent. It is the dominant source of noise at higher exposure lengths.

Dark Current Noise or dark shot noise [106] is a form of thermal noise characterised by the current present in the image sensor, even when the sensor is in complete darkness. While the mean bias component of dark current is commonly subtracted from the image (dark frame subtraction), the charge variance of the dark current causes noise. Like photon shot noise, the formation of dark shot noise is a discrete measurable event which follows a Poissonian distribution, specifically the random release of thermally generated electrons into

the sensor wells. Its standard deviation is therefore the square root of the dark current, σ_D :

$$\sigma_D = \sqrt{I_D t}, \quad (2.6)$$

Where I_D is the dark current which has accumulated during exposure. This dark current increases linearly with exposure time and exponentially with temperature. Dark current is usually only perceived in long exposures (10+ seconds). For this reason, long-exposure photography (for instance, astrophotography) often takes the mean of multiple short-exposure photographs to reduce thermal noise, and active thermoelectric cooling is used [111].

Read Noise or readout noise is a broad term for noise introduced in the signal creation process, after the sensor array measurement has taken place, and before the final raw image readout [112]. Ideally, the number of photons recorded and the raw digital readout would be directly proportional, as the amplifier gain is set inversely proportional to ISO, but this does not happen in practice. While the sub-processes responsible for read noise can be separately modelled, the magnitude of their noise contributions is unique to each camera and depends on the sensor type. Read noise is simply modelled as an additive constant for a given ISO and is dominant at low exposures. The majority of read noise originates from amplification circuits and hence it is sometimes also referred to as amplifier noise. In the case of CMOS sensors, each pixel has its own pre-amplifier which produces some noise due to amplifier resistance. Similarly, reset noise occurs due to an uncertainty in the reference voltage in the sensor wells between exposures, when the array is cleared of charge. Both are independent and are modelled by the Johnson-Nyquist noise equation, as defined in Equation 2.1:

$$\sigma_{read} = \sqrt{N_{nyquist}} = \sqrt{4kTB(R_{reset} + R_{MOSFET})}, \quad (2.7)$$

Where $N_{Nyquist}$ [58] is the total noise created by the integration circuit, R_{reset} is the effective channel resistance in ohms of the reset capacitor and R_{MOSFET} is the resistance of the preamplifier MOSFET for each pixel.

Fixed Pattern Noise (FPN) is most commonly associated with CMOS sensors [113, 114] and appears when the sensor is exposed to an evenly illuminated surface with no features. This artefact is caused by the pixel-to-pixel optical non-uniformity of the lenses used to direct light to each pixel, as well as the non-uniformity of the individual transistors used in each pixel of a CMOS sensor. FPN is signal-dependent, and it must be accounted for through dark frame subtraction at different exposure levels. As well as spatial FPN, CMOS sensors also suffer from column FPN, which is a result of the column amplifier non-uniformity. CCD

cameras are less susceptible to this kind of noise as they do not use per-column amplifiers. FPN is given as:

$$\sigma_{FPN} = \sqrt{N_{FPN}(S)} = \sqrt{\sigma_{Col}^2 + (\sigma_{spatial} \cdot S)^2}, \quad (2.8)$$

Where σ_{Col} is the standard deviation between the gain of the column amplifiers and $\sigma_{spatial}$ is the standard deviation of the pixels within columns. σ_{Col} is effectively an offset term and $\sigma_{spatial}$ is a gain variance term which is dependent on the raw input signal S . The column FPN is dominant in low exposure conditions (low S), while in well-lit conditions, the spatial FPN is more prevalent.

In summary, photon noise, dark-shot noise, read noise, and FPN noise can be approximated as Poissonian-Gaussian in their distribution, with two signal-dependent components (multiplicative) and a signal-independent component (additive). The total effective noise standard deviation, σ_{eff} per pixel is:

$$\sigma_{eff} = \sqrt{\sigma_D^2 + \sigma_R^2 + \sigma_S^2 + \sigma_{FPN}^2} \quad (2.9)$$

This can be generalised and expressed as a 2nd-order polynomial function with respect to the input signal, S :

$$\sigma_{eff} = \sqrt{c_0 + c_1 S^2 + S_{ph}} \quad (2.10)$$

Where c_0 and c_1 are positive non-negative constant values. This equation separates the noise sources by their independence of noise. c_0 represents the independent sources (read and dark noise), c_1 represents the square relationship of the signal to the Fixed Pattern Noise, and S_{ph} represents the photon shot noise.

Spatially Correlated Noise develops as the image passes through the ISP pipeline [115–118]. For instance, the demosaicing process introduces a spatial dependency on the noise within each channel [105]. Additionally, the noise variance is affected by the Gamma correction step, where the noise standard deviation decreases in dark regions and increases in bright regions. Other enhancement algorithms, such as edge enhancement and white balancing, which consider neighbourhoods of pixels when generating output pixels, also contribute to the spatial dependency in the noise distribution.

2.2 Classic Image Denoisers

In this section, we discuss several popular denoising techniques that were widely used before the advent of deep learning methods. Each technique is based on carefully constructed priors

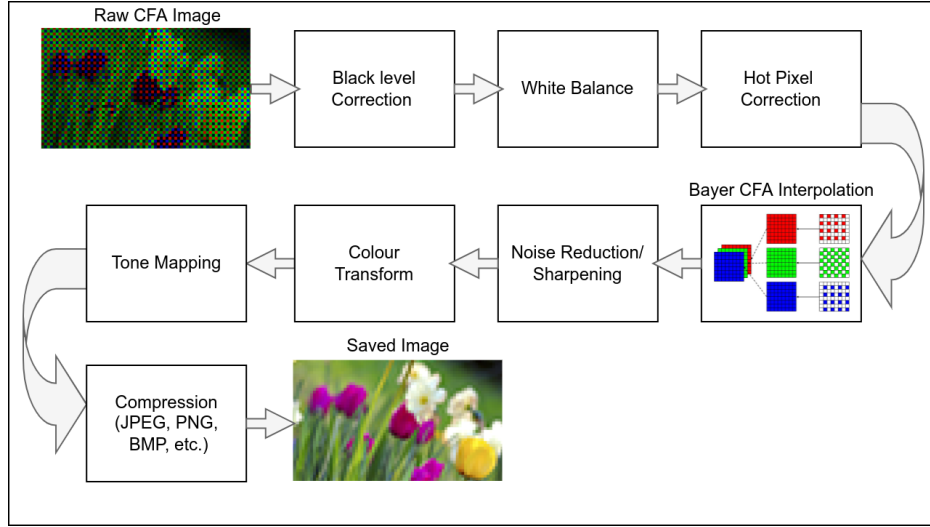


Fig. 2.2 An example of an Image Signal Processor (ISP) pipeline, illustrating the key stages involved in processing raw sensor data into a rendered image.

designed to model the characteristics of real images. To support this review, we evaluate each denoising method on 24 images taken from the Kodak dataset [119] with AWGN noise of standard deviation, $\sigma=20$. These results are arranged in Table 2.1, with quality measured in PSNR and SSIM.

A collection of denoising outputs is shown in Figure 2.3. Image denoising remains one of the most active fields in image processing literature, with hundreds of submissions made to online denoising benchmarks each year. Therefore, the following reviews are not intended to be exhaustive but rather to capture the most popular methodologies.

2.2.1 Early Spatial Denoising Filters

One of the most common priors in denoising is the assumption of spatial smoothness in natural images, meaning that pixel values do not change much in their local neighbourhoods. A basic form of these filters is linear filters, which are commonly applied by convolution:

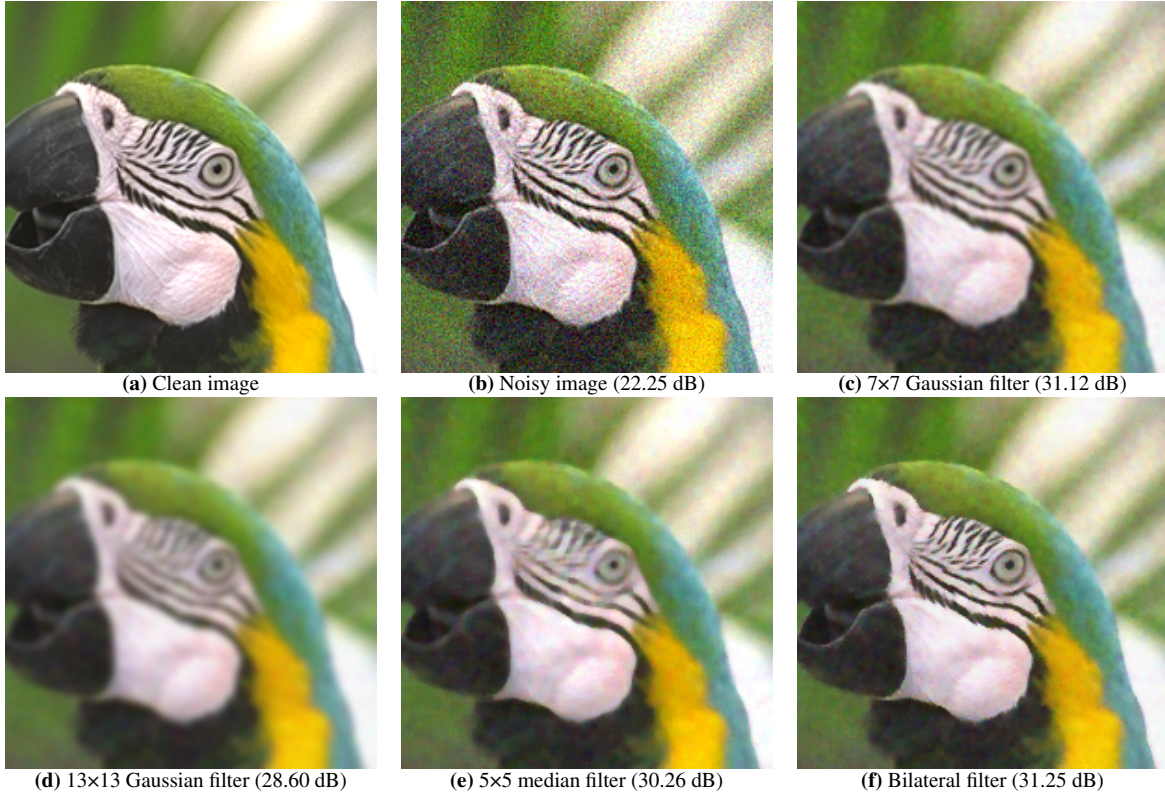
$$\hat{x}[i, j] = y[i, j] \otimes h[i, j], \quad (2.11)$$

where $\hat{x}$ is the prediction for the original image x and h is the convolution kernel. Equivalently, by the convolution theorem, convolution in the spatial domain is equal to multiplication in the frequency domain:

$$\hat{X}[\omega_1, \omega_2] = Y[\omega_1, \omega_2]H[\omega_1, \omega_2] \quad (2.12)$$

Method	PSNR (dB)	SSIM [0-1]	Time (s)	Blind/Non-Blind	GPU/CPU	First Proposed
Noisy Image	22.31	0.6027	—	—	—	—
Box filter	26.74	0.8231	0.04	Blind	CPU	~1960s
Gaussian filter	27.56	0.8346	0.19	Blind	CPU	~1970s
Median filter [120]	26.54	0.7952	0.57	Blind	CPU	1971
Bilateral filter [121]	28.44	0.8599	1.77	Blind	CPU	1998
Wiener filter [122]	30.65	0.8429	0.28	Non-Blind	GPU	1993
Wavelet Undecimated [123]	28.40	0.8701	16.21	Non-Blind	CPU	1995
TV Regularisation [124]	29.08	0.7889	17.40	Blind	CPU	1992
NLM [125]	29.95	0.8062	13.01	Non-Blind	CPU	2005
BM3D [126]	30.75	0.8367	279.71	Non-Blind	CPU	2007
DnCNN [30]	32.88	0.8971	0.13	Non-Blind	GPU	2017
IRCNN [72]	32.89	0.8960	0.14	Non-Blind	GPU	2017
FFDNet [43]	33.35	0.9012	0.13	Non-Blind	GPU	2018
UNet/RDUNet [127]	33.86	0.9117	169.27	Blind	GPU	2021
SWIN transformer [44]	35.46	0.9304	12.45	Non-Blind	GPU	2022

Table 2.1 Summary of popular denoiser performance on the Kodak dataset [119] with AWGN of standard deviation $\sigma = 20$. The PSNR and SSIM values represent the mean results across all denoised images. "Time" denotes the total denoising time, excluding data loading. For the Wiener filter, we benchmark our GPU-accelerated implementation, while the Wavelet filter uses a stationary transform with sym4 wavelets and a depth of 4.



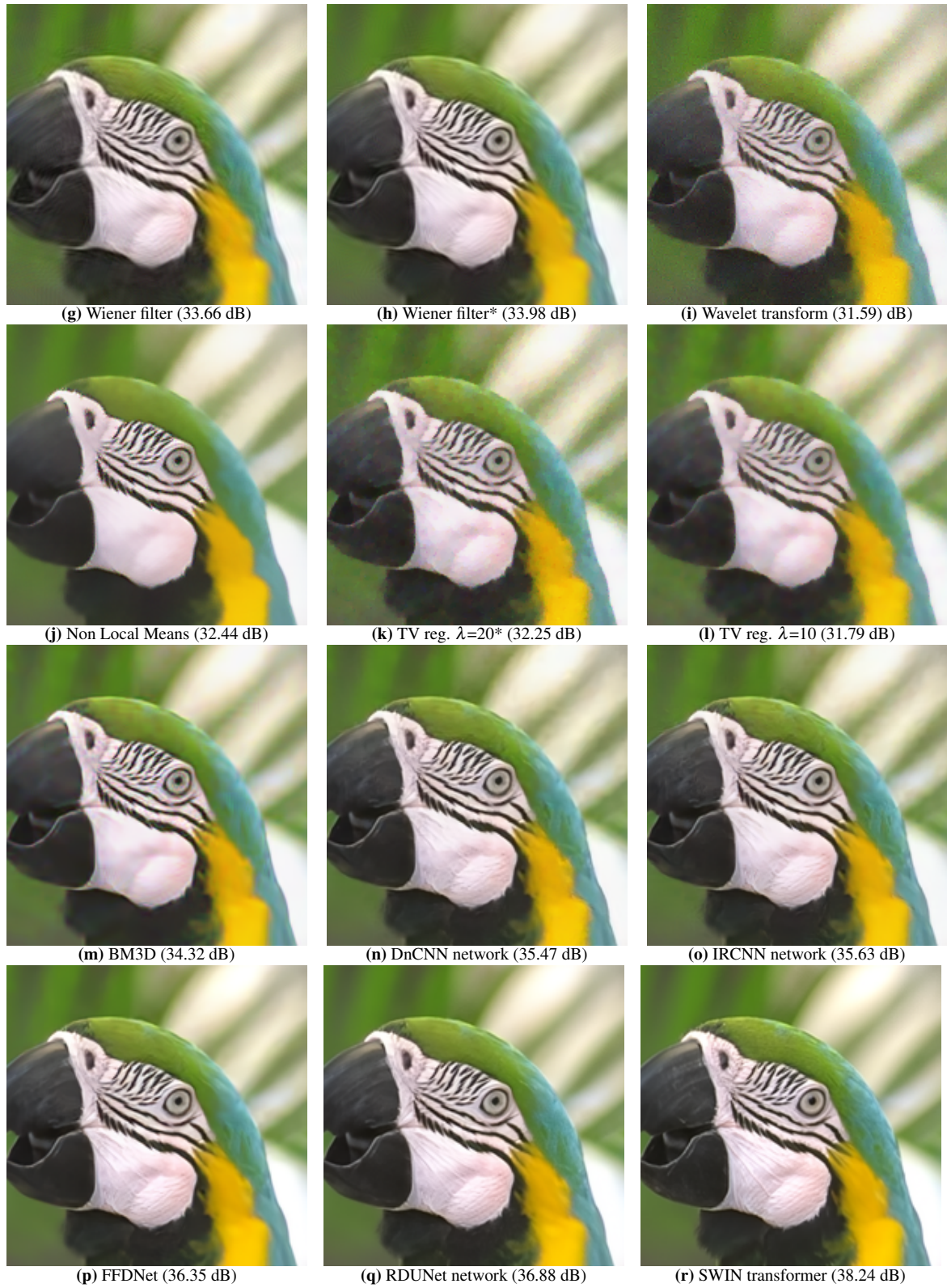


Fig. 2.3 Denoising output samples of algorithms listed in Table 2.1. The Wiener filter is configured to use half-window and quarter-window strides (Chapter 4). The wavelet filter employs an undecimated transform with sym4 wavelets at depth 4.

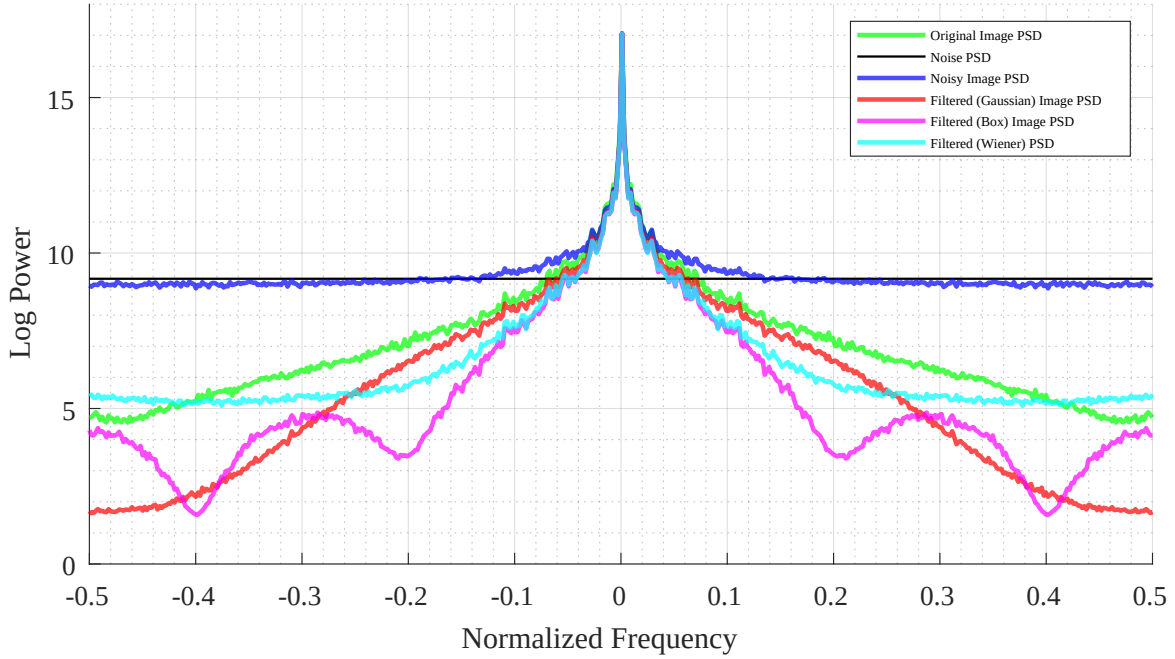


Fig. 2.4 Visualisation of power spectral density (PSD) for the noisy image shown in Figure 2.3, compared to the original image PSD, the stationary noise PSD, and the PSD of the filtered images. The side lobes of the box filter's frequency response (pink) are evident here, which causes ringing in high-frequency parts of the denoised image. The smoother roll-off is shown for the Gaussian filter. The Wiener filter most closely follows the original image noise power at higher frequencies.

The goal here is to choose a kernel whose frequency response forms a low-pass filter, which attenuates the noisy high frequencies and preserves the smooth low frequencies. For example, the frequency response of a box filter [128] is a Sinc function, which functions as a low-pass filter. Gaussian filters [128] achieve a similar effect but with a smoother cut-off frequency roll-off, as well as minimal side-band artefacts which minimise the appearance of ringing artefacts. This is illustrated in Figure 2.4, where we show the power spectral density of an image filtered using various methods. In Figure 2.3c we apply the Gaussian filter to our benchmark image; however, increasing the filter size, as in Figure 2.3d, results in a loss of high-frequency information.

Early on, simple non-linear filters [129, 130] were proposed as alternatives to linear filters for denoising tasks, such as the median [120] filter, which seeks to remove outliers from otherwise smooth sections. Gaussian noise and real noise, do not, however, add statistically significant outliers for the median filter, which is more suited to impulse noise. As in the case of the box filter, the median filter over smooths images and results in a loss of detail, as shown in Figure 2.3e. Edge-aware non-linear filters such as the bilateral filter [121] denoise the image using a weighted average of pixel values based on both the Euclidean distance

from the target pixel and the difference in pixel intensity between the target and neighbouring pixels. This technique applies less denoising around edges and more smoothing in flat areas, thus preserving more image detail than a box filter as illustrated in Figure 2.3f, where we see sharper edges than the Gaussian filter. Similarly, edge-aware [131], weighted [132], versions of the median filter have been subsequently proposed in attempts to more effectively filter Gaussian noise.

Despite the edge-aware strategies of these filters, they do not separate high-frequency components of the image from the noise well enough and perform poorly in low-frequency smoothing. In the following section, we will see how an adaptive filter like the Wiener filter [122, 133] changes its frequency response based on the strength of the signal to more effectively restore the original spatial frequencies of the image.

2.2.2 The Wiener Filter

The Wiener filter [5] was a major advancement in image denoising [134], operating on the prior that both the original image signal x and the noise n are stationary signals, of fixed mean and standard deviation. In deriving the Wiener filter, it is assumed that the noise is additive: $y[i, j] = x[i, j] + n[i, j]$ and the noise is modelled as AWGN. As our work makes extensive use of the Wiener filter, we derive it in more detail here. The Wiener filter can be derived from first principles in the IIR case from the posterior defined in Equation 2.2. The MAP estimate for x is as follows:

$$\hat{x} = \arg \max_x p(x|y) = \arg \max_x p(y|x)p(x),$$

By assuming a Gaussian distribution for $p(y|x)$, and a no-informative uniform prior $p(x)$, the mode of the posterior distribution is identical to its mean. Thus, the MAP in this case is equivalent to the Minimum Mean Square Error (MMSE):

$$\mathbb{E} \|\hat{x} - x\|^2.$$

Remembering that our signal uses the AWGN noise model: $y = x + n$, the Wiener filter aims to find an optimal filter for the estimate x by the action:

$$\hat{x} = h \otimes y,$$

where h is the filter impulse response which must be estimated.

Derivations in 1D. Here, we proceed with a 1D signal, indexed with t , but the derivation can easily be extended for the 2D case. The MMSE estimate for h may now be written as follows.

$$\hat{h} = \arg \min_h L(h) = \sum_t \mathbb{E} (x[t] - (h \otimes y)[t])^2. \quad (2.13)$$

where $L(h)$ is the expected squared error. Assuming h to be an IIR filter, the frequency domain model takes the form:

$$Y(\omega) = X(\omega) + N(\omega) \quad (2.14)$$

$$\hat{X}(\omega) = H(\omega)Y(\omega) \quad (2.15)$$

Since the observation model is linear and additive, the MAP estimate reduces to a linear MMSE problem. By Parseval's theorem, the objective can equivalently be expressed in the frequency domain as:

$$\hat{H} = \arg \min_H L(H) = \arg \min_H \mathbb{E} \|X(\omega) - H(\omega)Y(\omega)\|^2 \quad (2.16)$$

Expanding the quadratic:

$$\hat{H}(\omega) = \arg \min_{H(\omega)} \mathbb{E} \|X(\omega) - H(\omega)Y(\omega)\|^2 \quad (2.17)$$

$$\begin{aligned} &= \arg \min_{H(\omega)} \mathbb{E} [(X(\omega) - H(\omega)Y(\omega))(X(\omega) - H(\omega)Y(\omega))^*] \\ &= \arg \min_{H(\omega)} \mathbb{E} [|X(\omega)|^2 - X(\omega)H^*(\omega)Y^*(\omega) - H(\omega)Y(\omega)X^*(\omega) + |H(\omega)|^2|Y(\omega)|^2] \end{aligned} \quad (2.18)$$

To minimise the complex-valued objective function with respect to $H(\omega)$, we apply Wirtinger calculus [135], which is classically used in this context to handle optimisation involving complex variables. This approach treats $H(\omega)$ and $H^*(\omega)$ as independent when the objective function is real-valued. For such functions $L(H, H^*)$, the Wirtinger derivatives satisfy $\frac{\partial L}{\partial H} = \left(\frac{\partial L}{\partial H^*} \right)^*$, so setting one derivative to zero implies the other also vanishes. Thus, the necessary and sufficient condition for a stationary point is $\frac{\partial L}{\partial H^*(\omega)} = 0$, which we use to solve for $H(\omega)$.

The first and third terms of equation 2.18 are treated as constants, and the fourth term, $|H(\omega)|^2|Y(\omega)|^2$, expands to $H(\omega)H^*(\omega)|Y(\omega)|^2$. Thus, the derivative of equation 2.18 can

be found as:

$$\frac{\partial L}{\partial H^*(\omega)} = -\mathbb{E}[X(\omega)Y^*(\omega)] + H(\omega)\mathbb{E}[|Y(\omega)|^2], \quad (2.19)$$

Setting this equal to zero and solving gives the optimal Wiener filter:

$$\hat{H}(\omega) = \frac{\mathbb{E}[X(\omega)Y^*(\omega)]}{\mathbb{E}[|Y(\omega)|^2]} = \frac{P_{xy}(\omega)}{P_{yy}(\omega)}, \quad (2.20)$$

where $P_{xy}(\omega)$ is the cross power spectral density describing the frequency-domain correlation between the clean signal $x(t)$ and the observed noisy signal $y(t)$. Also $P_{yy}(\omega)$ is the power spectral density of the observed signal.

In practice, the original signal $X(\omega)$ is unknown, which prevents direct computation of the cross-spectrum $P_{xy}(\omega)$ and the signal power $P_{xx}(\omega)$. To proceed, we assume that the noise $N(\omega)$ is zero-mean, white, and statistically independent of the signal. Under this assumption:

$$\begin{aligned} P_{xy}(\omega) &= \mathbb{E}[X(\omega)(X(\omega) + N(\omega))^*] \\ &= \mathbb{E}[|X(\omega)|^2] + \mathbb{E}[X(\omega)N^*(\omega)] \\ &= P_{xx}(\omega) + 0 = P_{xx}(\omega), \end{aligned} \quad (2.21)$$

where the cross-term $\mathbb{E}[X(\omega)N^*(\omega)]$ vanishes by independence. Substituting Eqn. 2.21 into 2.20, the Wiener filter becomes:

$$\hat{H}(\omega) = \frac{P_{xx}(\omega)}{P_{yy}(\omega)}. \quad (2.22)$$

As P_{xx} is also unknown, it is instead estimated by $P_{xx} \approx P_{yy} - P_{nn}$, which follows from equation 2.14. Thus, we have the MSE optimal estimated Wiener filter:

$$\hat{H}(\omega) = \frac{P_{yy}(\omega) - P_{nn}(\omega)}{P_{yy}(\omega)}, \quad (2.23)$$

where $P_{yy}(\omega)$ is computed from the observed noisy signal and $P_{nn}(\omega)$ is estimated from known noise statistics. The assumption that the signal and noise are perfectly uncorrelated does not always hold, particularly for real-world images. As a result, the estimated numerator $P_{yy}(\omega) - P_{nn}(\omega)$ may become negative at some frequencies. To preserve non-negative gain, we use instead:

$$\hat{H}(\omega) = \frac{\max(P_{yy}(\omega) - P_{nn}(\omega), 0)}{P_{yy}(\omega)} \quad (2.24)$$

or in the 2D case:

$$\hat{H}(\omega_1, \omega_2) = \frac{\max(P_{yy}(\omega_1, \omega_2) - P_{nn}(\omega_1, \omega_2), 0)}{P_{yy}(\omega_1, \omega_2)} \quad (2.25)$$

Once the Wiener filter is found, $\hat{H}(\omega_1, \omega_2)$ can be multiplied by $Y(\omega_1, \omega_2)$ (as in Eqn. 2.15) and the DFT (Discrete Fourier Transform) is reversed to obtain the estimate for the clean image, $\hat{x}(i, j)$, in the spatial domain.

Due to the assumption of stationarity made by the Wiener filter, it performs poorly when applied directly to images which are not split into smaller analysis windows (e.g., 16×16 pixels). This adds complexity as the Wiener filter must be derived for each block.

In summary, the Wiener filter can be characterised as a frequency-selective coring mechanism whose weighting is based on the power of the observed image signal, P_{yy} , and an estimation of the noise power, P_{nn} . A coring function suppresses frequency components with low energy—typically associated with noise—while retaining stronger components that are more likely to represent signal content. This allows for high levels of smoothness in low-frequency areas corrupted by noise while selectively preserving details in high-frequency regions where the signal-to-noise ratio is higher.

The Wiener filter has continued to be successfully iterated upon for use in image [133, 136, 137], and video restoration [138, 139]. Kokaram et al. [138] proposed the use of motion estimation in image sequences to effectively use the Wiener Filter in 3 dimensions. In our work, we recreate the Wiener filter as presented by Kokaram using the Pytorch framework for GPU-accelerated parallelism. In Figure 2.3g, we demonstrate the improvement in denoising quality brought by the Wiener filter, smoothing the flat background area while preserving high-frequency detail in the bird’s face. Although a significant improvement on classical linear filters, the Wiener filter suffers from ringing artefacts around edges and textures. This is a well-observed effect of Fourier series compression known as Gibbs phenomenon [140], where the Fourier sum overshoots the original signal at discontinuities in the spatial domain. This effect can be reduced by increasing the amount of overlap between each window used in the filter, as in Figure 2.3h.

2.2.3 Wavelet Denoising

Developed by Simoncelli as a natural extension to Wiener denoising in the paper *Noise Removal Via Bayesian Wavelet Coring* [141], wavelet denoising proposes to instead carry out coring or ‘wavelet thresholding’ in a wavelet domain which is more well-suited to capturing

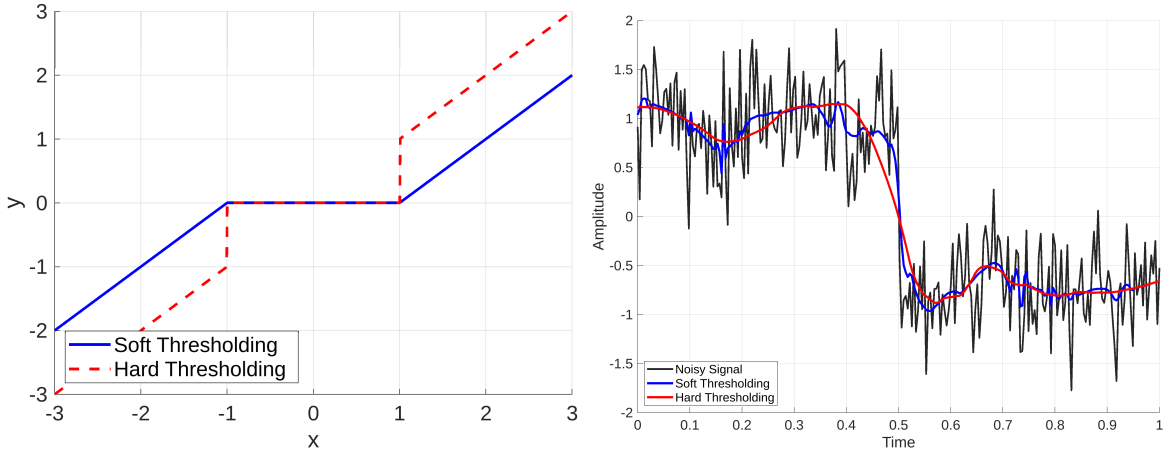


Fig. 2.5 Left: Example of Wavelet soft thresholding and hard thresholding functions. Right: Wavelet smoothing of 1D signal via hard and soft thresholding.

the non-Gaussian characteristics of natural images. Using the same posterior probability framework used to derive the Wiener filter, the probability that a wavelet coefficient belongs to the signal or the noise is computed and used to selectively reduce or eliminate high-frequency components. A 4-level steerable pyramid wavelet [123] is used in this case, applying the filter successively to the low-pass sub-bands of the image. Unlike other wavelet schemes which downsamples at each level, Simoncelli's version is undecimated and shift-invariant which resulting in an output which reflects input textures, at greater computational cost.

As illustrated in Figure 2.6, undecimated wavelet filtering operates by decomposing the image into high-frequency and low-frequency coefficients at each level. The decomposition process is iterative, with the low-pass coefficients from the current level serving as the input for the next level. At each stage, the coefficients are thresholded to suppress noise before the inverse filters are applied. Finally, the image is reconstructed by combining the high-frequency and low-frequency coefficients across all levels.

Many wavelets have been proposed for the task [142–144], as well as different methods of coring or ‘thresholding’ the signal [145], either classed as hard-thresholding, which sets all coefficients below a certain value to zero, or soft-thresholding [28], which also shrinks coefficients below the threshold, as well as reducing those above the threshold to zero. This is similar to Wiener denoising, which applies coring weights to each frequency band which depend on the signal-to-noise strength. For an illustrated example of hard and soft thresholding, see Figure 2.5.

Further methods to optimise thresholding include SURE (Stein’s unbiased risk estimate) thresholding [146, 147], which uses a mean square error optimal soft-thresholding technique;

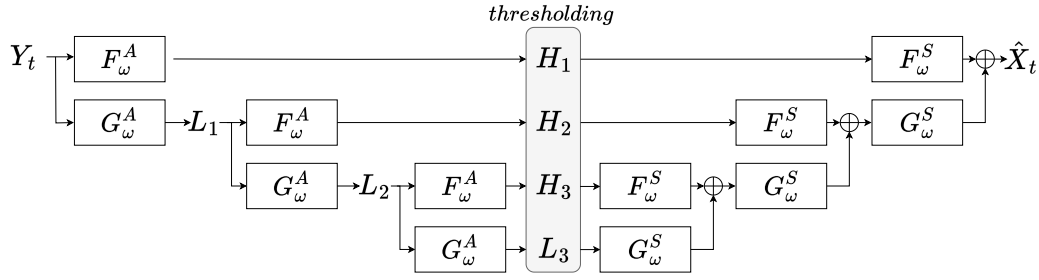


Fig. 2.6 Example of 1D Undecimated Wavelet filtering, where F_w is the high pass filter and G_w is the low pass filter. L and H denote the output low-pass and high-pass coefficients, respectively. As the analysis and reconstruction filters are not guaranteed to be the same, they have been labelled F^A, G^A and F^S, G^S respectively.

Bayes-Shrink [148], which uses unique thresholding values for each sub-band based on estimates of the sub-band noise variance; and wavelet domain Gaussian-Scale Mixture (GSM) model [149]. The latter models the wavelet coefficients as Gaussian scale mixtures to better capture the heavy-tailed nature of the coefficient distribution. Each wavelet is modelled as a random variable of its local neighbourhood, which is defined by its spatial neighbours in the same sub-band and the coarser sub-band, as well as different orientations of the same spatial location. A Bayesian estimation for the new coefficients is then made based on the local Gaussian model.

Wavelet denoising is a technique commonly employed in high-end video post-production tools such as Foundry's Nuke [150], a platform widely used in the film industry, where denoising is an important step in the video production pipeline. In Figure 2.6, we demonstrate stationary (or undecimated) wavelet denoising using a Symlet (Sym4) wavelet with a decomposition level of 4. Soft thresholding is applied using the BayesShrink algorithm[148]. Unlike the Fourier transform, used by the Wiener filter, which is fully localised in frequency, wavelets provide a dual localisation in both space and frequency. This characteristic allows wavelets to effectively minimise boundary artefacts, such as ringing, which are introduced when high-frequency coefficients are sharply attenuated and poor spatial localisation occurs. Wavelets achieve this by compromising on frequency-localisation accuracy, thus balancing spatial detail preservation and frequency resolution. However, artefacts still occur in the form of repetitive patterns in homogeneous areas, which arise due to the interaction of wavelet basis functions and the thresholding process.

2.2.4 Non-Local Priors

Unlike spatial filters, which are limited to information provided by local pixels in a small neighbourhood, non-local priors exploit the fact that natural images contain recurring patterns to effectively denoise images. Baudes et al. are credited as the first to introduce non-local denoising in their 2005 paper, *A non-local algorithm for image denoising* [125]. In their work, they propose the Non-Local Means (NLM) method, which defines a neighbourhood around a target pixel and performs a raster scan of all possible neighbourhoods, assessing their similarity to the target pixel neighbourhood by Gaussian-weighted Euclidean distance. Once the scan is complete, the highest-ranked neighbourhoods average their centre pixel values and the result is returned to the target pixel. The algorithm is costly but has been further optimised [151, 152] by specifying search region sizes and performing weighted averaging based on a window's similarity score. A sample of NLM denoising is included in Figure 2.3j, which can be characterised by the smoothness of low-frequency areas and the preservation of lines and edges. As the algorithm relies on repeated patterns, unique details in the bird's facial marking are not as accurately reproduced as the rest of the image, leaving the noise behind.

Inspired by non-local denoising and transform domain filtering, Block-Matching 3D [126] (BM3D), set the benchmark among denoisers from the late 2000s to the mid-2010s, being the benchmark to beat in early deep learning denoisers. The paper by Dabov et al., *Image Denoising by Sparse 3-D Transform-Domain Collaborative Filtering* proposes a two-step denoiser (as shown in Fig. 2.7) which denoises images in grouped blocks (rather than single pixels) in two passes using collaborative filtering.

In the first pass, similar image patches are placed into groups by matching using a distance function. Each group is then transformed by FFT and hard-thresholded. The inverse FFT is applied and the grouped results are aggregated for every patch by weighted mean. For the second pass, instead of hard thresholding, a Wiener filter is devised using the previous estimate as a pilot clean signal, as well as the original noisy image. BM3D has been continuously developed by Tampere University, with implementations proposing different cost/distance functions and search algorithms [26], replacing the Wiener filter with a wavelet filter [153, 154] and BM4D [155], the video denoising counterpart. BM3D is officially part of the GNU library and stands as the main benchmark for image denoising in pre-deep learning publications. We include a sample of the output produced by BM3D in Figure 2.7. The algorithm benefits from the smoothness of NLM while preserving more high-frequency detail using its Wiener filter module. However, some colour inconsistencies are visible in the high-frequency face markings which can also be found in the NLM example.

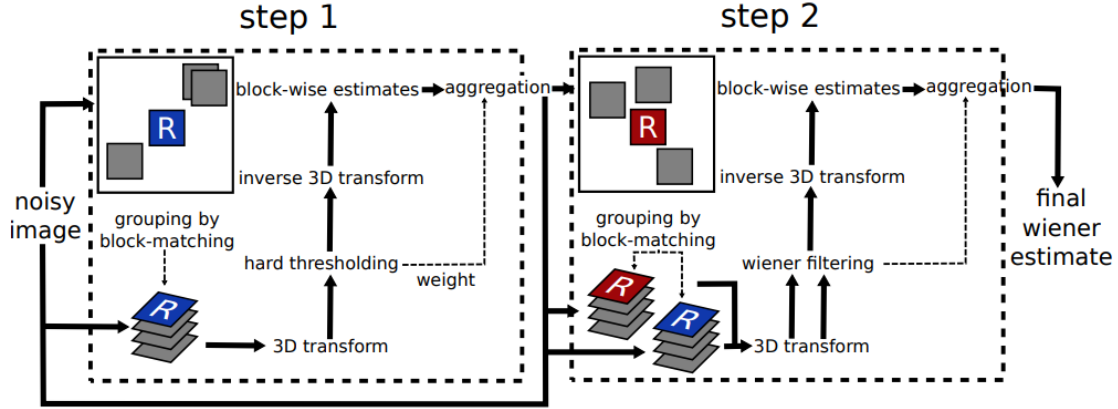


Fig. 2.7 The two-stage BM3D Denoiser [126] as presented by Dabov et al. In step 1, a basic estimate is formed. Similar patches are grouped and undergo a 3D transformation. The transform coefficients are thresholded and returned to the spatial domain where the final patches are estimated via aggregation. In step 2, the same groups are used in the transform domain, and Wiener filtering is applied to produce a refined estimate. The basic estimate from step 1 is used to approximate the clean signal needed by the Wiener filter.

2.2.5 Variational Denoisers

Variational denoising proposes a framework whereby both the data fidelity of a prediction is considered (how close the prediction, $\hat{x}$, is to the observed input image y), as well as the spatial smoothness of the prediction. The sum of these terms is considered as part of an energy function which is minimised to find the optimal prediction $\hat{x}$. As shown in the Wiener derivation in Section 2.2.2, a prediction can be made for x using the MAP of the posterior, which is equivalent to minimising the negative log-posterior:

$$\begin{aligned}
 \hat{x} &= \arg \max_x p(x|y) = \arg \max_x p(y|x)p(x) \\
 &= \arg \min_x \left\{ \underbrace{-\log p(y|x)}_{\text{Data Fidelity Term}} + \underbrace{\log p(x)}_{\text{Smoothness Term}} \right\}
 \end{aligned} \tag{2.26}$$

Here, $-\log p(y|x)$ corresponds to the log-likelihood, and is often referred to as the data fidelity term. $\log p(x)$ represents our prior expectation about the image gradient and is known as the smoothness term.

Tikhonov Regularisation Tikhonov regularisation, also known as ridge regression [156, 157], serves as a foundational method in variational denoising. It can be derived directly from the posterior probability formulation in Equation 2.2. Assuming the observed image

y is corrupted by Gaussian noise with zero mean and variance σ^2 , the likelihood function $p(y|x)$ is given by:

$$\begin{aligned} p(y|x) &\propto \exp\left(-\frac{1}{2\sigma^2}\|y-x\|_2^2\right), \\ -\log p(y|x) &= \frac{1}{2\sigma^2}\|y-x\|_2^2 + c, \end{aligned} \quad (2.27)$$

where c is a constant. The prior $p(x)$ depends on the regularisation model. For Tikhonov regularisation [158], the prior penalises the L_2 norm of the image gradient, representing a Gaussian prior:

$$\begin{aligned} p(x) &\propto \exp\left(-\frac{\alpha}{2}\|\nabla x\|_2^2\right), \\ -\log p(x) &= \frac{\alpha}{2}\|\nabla x\|_2^2 + c, \end{aligned} \quad (2.28)$$

where ∇x is the gradient of the image, α is the regularisation weight, and c is a constant. Substituting Equations 2.27 and 2.28 into the posterior formulation yields:

$$\begin{aligned} \hat{x} &= \arg \min_x \left\{ \frac{1}{2\sigma^2}\|y-x\|_2^2 + \frac{\alpha}{2}\|\nabla x\|_2^2 \right\} \\ &\approx \arg \min_x \left\{ \|y-x\|_2^2 + \lambda \|\nabla x\|_2^2 \right\}, \end{aligned} \quad (2.29)$$

where λ controls the trade-off between noise reduction and detail preservation. When $\lambda = 0$, no smoothing occurs, and as $\lambda \rightarrow \infty$, the output becomes overly smooth, losing resemblance to the original image.

Minimisation of Equation 2.29 is typically achieved by solving the corresponding normal equations through iterative methods such as Conjugate Gradient [159, 160] or Generalised Minimum Residual (GMRES) [161, 162].

Total Variation (TV) Regularisation operates on the assumption that natural images are smooth in small areas, whereas noisy signals exhibit high total variation. In other words, the absolute image gradient is higher in noisy images. By reducing this gradient

while constraining the prediction to remain similar to the input image y , TV regularisation effectively removes noise while preserving edges. This method more closely models real images because it enforces sparsity in the gradient, thus preserving edge sharpness. However, TV regularisation is more challenging to solve than Tikhonov regularisation - while it is also a convex function, the TV regulariser is non-smooth at zero. This necessitates more complex algorithms to find optimal solutions.

TV regularisation can be expressed as the minimisation of the objective function:

$$\hat{x} = \arg \min_x \|y - x\|_2^2 + \lambda \|\nabla x\|_1 \quad (2.30)$$

Here again, the parameter λ is the regularisation parameter that controls the level of smoothness. Unlike L_2 -norm-based regularisation (like Tikhonov), TV regularisation preserves edges as the TV penalty is lower for piecewise constant images, where intensity changes abruptly at edges but remain constant within regions.

In the original paper, the authors solved the minimisation problem by deriving the Euler-Lagrange equations from the objective function and using gradient descent until the system converges to a steady state. Later developments introduced more efficient solvers for variational denoising problems such as primal-dual methods [124], ADMM [163] and the Split Bregman method [164]. TV regularisation can produce staircase artefacts in flat image regions and can also reduce contrast in areas [124, 165, 166]. The algorithm has been optimised for speed [167] and performance [168, 169]. Regularisation was among the most popular approaches to denoising before the arrival of deep learning models.

In Figure 2.3k and 2.3l, examples of TV regularisation are shown using the primal-dual solver method. In this implementation, λ weights the data fidelity term, rather than the image gradient term. As a result, a lower λ produces smoother images at the cost of reduced data fidelity, as demonstrated in the examples. Staircase artefacting is common in TV regularisation and is visible in Figure 2.3k. This occurs as the cost function is calculated image-wise and piecewise-constant regions (i.e. flat regions) are favoured, while gradual intensity changes are penalised. As a result, abrupt changes in intensities or “steps” appear between flat regions.

Non-Local Means Regularisation has also been used in the variational framework [170] given the success of NLM in general denoising tasks, as described in Section 2.2.4. A non-local self-similarity regulariser R_{NSS} is implemented, replacing the traditional smoothness term from the previous methods. This regulariser weighs the total variation as a function of

the similarity between image patches. The similarity is measured by the Euclidean distance between patches, and these weights guide the denoising process to preserve important features such as edges while reducing noise. In this approach, pixels with high non-local self-similarity (i.e. pixels that match well with others) are penalised less than pixels with low self-similarity scores. The regularisation prior to this method is given by:

$$-\log p(x) = R_{NSS}(x) = \lambda \sum_i \sum_j w[i, j] (\hat{x}_i - \hat{x}_j)^2,$$

where $w[i, j]$ is the self-similarity score between pixels at locations i and j , and λ is the regularisation parameter which controls the strength of the denoising. The weights are found by the equation:

$$w[i, j] = \exp \left\{ -\frac{\|P_i - P_j\|_2^2}{h^2} \right\},$$

where P_i and P_j are vectors stacking all the pixel values inside patches centred at pixels i and j respectively, and h is a filtering parameter that controls the decay of the weights with respect to the patch dissimilarity. As the performance of non-local algorithms overtook local denoising, much work was carried out in optimising in terms of efficiency [152, 171–175] and performance [176–178].

Sparse Representation Regularisation Sparse representation smoothing uses the idea that an image patch can be reconstructed using a linear combination of other patches. As such, an over-complete dictionary of patches is used to build the image from a matrix of sparse vectors representing the image. When optimising the sparse vector, the noise should be treated as redundant information and left behind. This variational minimisation equation is as follows:

$$\hat{\beta} = \arg \min_{\beta} \|y - D\beta\|_2^2 + \lambda \|\beta\|_1, \quad (2.31)$$

where β is the sparse representation vector patch and D is the dictionary.

The regularisation term here, $\lambda \|\beta\|_1$, is the L1-norm sparsity regularisation. The sparse vector β is minimised while preserving similarity to the original image. K-Singular Value Decomposition (K-SVD) [179, 180] is a popular method of dictionary learning by Aharon, Elad et al., which divides the image into overlapping patches, treating them as vectors which should be represented as a sparse linear combination of the ‘dictionary atoms’. The dictionary is learned from the set of all patches and the K-SVD algorithm alternates between updating the sparse codes and updating the dictionary atoms to minimise the reconstruction error. The prior has also been combined with the NSS prior (non-local centralised sparse representation [181])

and shown very good results. These methods are, however, computationally demanding, limiting their applicability.

Similarly to sparse representation, Low-rank minimisation [182] seeks to represent the target image in a more compact form to reduce the dimensionality of the data. Instead of using a dictionary of column vector patches, low-rank minimisation groups similar patches into matrices, where the low-rank property of the matrices is exploited to remove noise. In this approach, the goal is to find a low-rank approximation of matrices that preserves image detail while removing noise.

This can be achieved using methods such as low-rank matrix factorisation [183, 184] and nuclear norm minimisation [185], which approximate or enforce the low-rank structure of the patch matrices. A popular technique for low-rank minimisation includes Weighted Norm Minimisation [186, 187], which has been developed to improve performance and denoising quality. However, like sparse representation methods, these approaches are computationally intensive, which is a limiting factor in their practical applicability.

2.3 Deep Learning Priors

Following the success of convolutional neural networks in image classification tasks with early networks such as AlexNet [188], VGGNet [189], GoogleNet [190] and Resnet [191], similar architectures were quickly developed for image enhancement tasks such as super resolution [192, 193], demosaicing [72, 194], and of course, image denoising. In this section, we briefly discuss the development of image denoising neural networks, which define the current state-of-the-art in output quality.

2.3.1 Image Denoising CNN Architectures

The first image denoising convolution neural networks (CNNs), processed images in an end-to-end manner, preserving the input resolution through the network and directly predicting the denoised image. The concept was proven to be competitive with the Multi-Layer Perceptron (MLP) network proposed in: *Image denoising: Can plain Neural Networks compete with BM3D?* [195] (2012), which demonstrated competitive denoising performance against popular non-local denoisers. With larger datasets being made accessible for classification challenges [196], and the availability of more powerful consumer graphic cards, fixed-resolution networks surpassed denoisers such as BM3D [26] in terms of output quality.

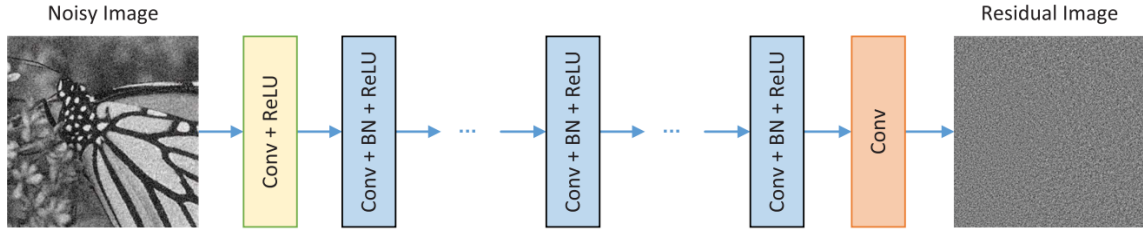


Fig. 2.8 The DnCNN [30] (Denoising CNN) architecture as proposed by Zhang et al. The noise is predicted (image residual) instead of directly predicting the output for improved inference quality and faster training.

One of the first networks to outperform BM3D was DnCNN, proposed by Zhang et al., in their 2017 paper: *Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising* [30]. The proposed network consists of a simple end-to-end fixed-resolution architecture, with 17 CNN layers. Each layer is made up of a 3×3 convolution, a batch-normalisation layer, and a ReLU [197] non-linear activation function. The first layer of the network uses a dilated convolution to increase the receptive field size. Inspired by Resnet [191], the authors choose to indirectly denoise the image by predicting the image residual, and subtracting it from the image afterwards for faster training times and increased output quality.

Despite its simplicity, DnCNN remains popular today as a lightweight, general-purpose image enhancement backbone [72, 198]. We include DnCNN in our denoising benchmark and show a sample output in Figure 2.3n. Unlike the priors of classic denoisers, the training process of CNNs enables them to build much more powerful image priors, allowing them to reconstruct detail and texture which is obscured and not repeated. For example, in our case, the network reconstructs textures in the bird's cheek which are lost in the noisy image. However, as we will see, the output may still be improved. The shallow receptive field of the network produces uneven transitions in the band of lights seen in the background.

Following DnCNN, the same authors created FFDNet - *Fast and Flexible denoising network* [199], a more shallow network which achieves a greater effective receptive field size by firstly downsampling the noisy input image into four, quarter-resolution images before passing them through the 15-layer network. As well as subsampled images, FFDNet introduces the concept of adding a noise-map channel to the network input. In this work, where the noise is Gaussian and stationary, the noise map is simply a single number which fills an image-shaped matrix. Unlike DnCNN, FFDNet does not predict the residual image. A high-level representation of the network is shown in Figure 2.9.

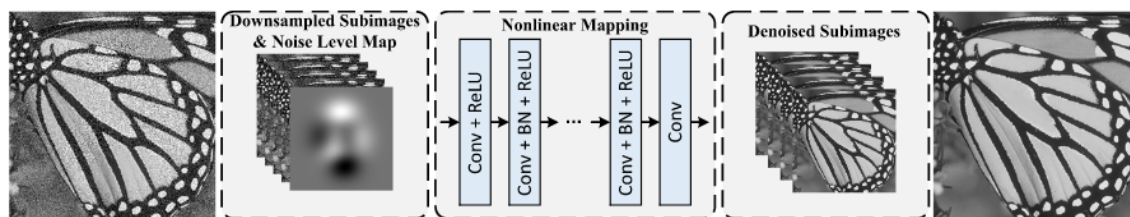


Fig. 2.9 The FFDNet (Fast and Flexible Denoising Architecture) as presented by Zhang et al. [199]. The network introduces the use of a noise map as an extra input to a network to guide the denoising. The input image is downsampled into four images, along with the noise map, before they are input to the network as a five-channel array. The output consists of 4 denoised subsampled images which are combined to produce the final full resolution image.

The U-Net architecture, initially developed for biomedical image segmentation [200], has become a cornerstone in CNN architectures for denoising tasks. Its encoder-decoder structure (inspired by autoencoders), resembling a U-shape, was later adapted in the ResUNet architecture by Heinrich et al. [201] for image denoising. This design involves multiple downsampling stages that distil noisy inputs into a latent space where details are preserved while noise is left behind. This strategy not only enhances feature richness per layer but also maintains a compact network footprint.

The original ResUNet image denoiser downsamples the input image four times in total, via a 2×2 max-pooling layer between every second layer of the 10-layer network. For the decoder, bilinear interpolation is used to upscale the feature maps before reconstruction convolution layers are applied. As the name suggests, the network incorporates residual connections, or skip connections, between corresponding contracting and expanding encoder-decoder layers, originally implemented as channel concatenation, as shown in Figure 2.10. This gives the decoder layers access to information lost in compression and facilitates the back-propagation stage of training by reducing vanishing gradients. This method of compressing and decoding the images in a latent space has proven to be efficient in training, with faster convergence than end-to-end networks with the same parameter count. In Figure 2.3q, we include an output sample from a U-Net-based image denoising architecture known as RDUNet [127] (Residual-Dense U-Net). In this example, even more details are recreated in the face, as well as beak textures.

Today, U-Net stands as a widely favoured backbone for denoising models [34, 40, 48, 202–209], underlining its effectiveness and versatility in image restoration applications. These networks are also larger in terms of their model sizes and the number of trainable parameters used in comparison to networks such as DnCNN. This becomes a limiting constraint when denoising images on low-power devices which do not have sufficient memory to hold the

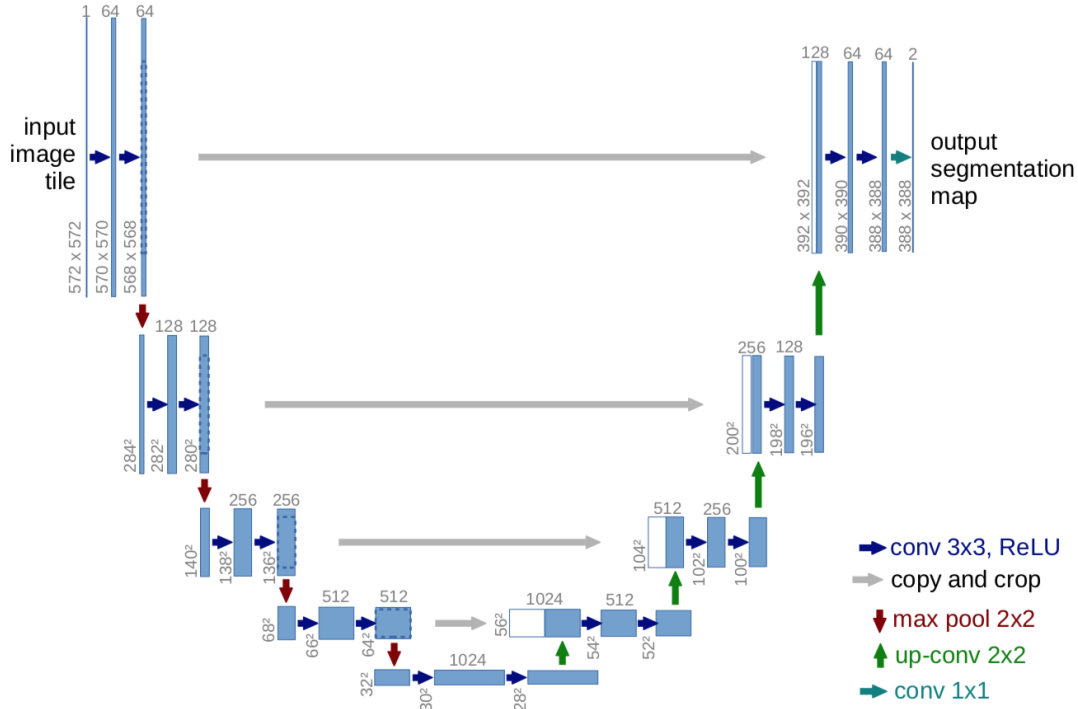


Fig. 2.10 The original U-Net architecture for biomedical image segmentation by Ronneberger, Fischer and Box [200]. U-Nets are composed of an encoder, which repeatedly downsamples the image, with convolutions in between, and a decoder which subsequently upsamples the image. Skip connections are used to create links between the encoder and decoder to preserve information and speed up training. The network's effectiveness has made it one of the most popular architectures for deep learning tasks today.

expanded feature maps in memory during inference. For example, the original U-Net is comprised of 31 million parameters and networks, which cascade U-Nets such as U2-Net [209] use 44M parameters.

A U-Net of particular interest, presented in the paper *Toward Convolutional Blind Denoising of Real Photographs* (2019) by Guo et al., is CBDNet [40], which puts forward a blind denoiser designed for real-noise removal. The first key contribution made by the authors is the use of a more realistic real noise model, used to synthesise the training dataset. This model features the inclusion of signal-dependent (photon noise) and signal-independent (read noise, thermal noise) noise components, as well as the recreation of some ISP pipeline steps such as demosaicing, colour correction and compression. As producing identically exposed noisy-ground-truth image pairs is labour-intensive, time-consuming and limited to changing the sensor's ISO settings, and with the knowledge that each camera produces a unique noisy profile, such a scheme as used by Guo et al. provides an alternative to the

limited real-noise datasets currently available [38, 39, 210]. This synthetic noise model is applied to images from the BSD500 [211], Waterloo [212], and MIT-Adobe FiveK [213] datasets, and mixed with real noise images from the Renoir dataset [39] to make up the training dataset for CBDNet.

The second key contribution made by CBDNet is the inclusion of the noise estimation prediction subnetwork. Unlike previous blind Gaussian noise networks such as FFDNet [30], in which the user supplies the noise which is subsequently passed to the network, CBDNet trains a 5-layer module to make a per-pixel prediction of the noise before passing it as a fourth channel to the input of the denoising U-Net. The network is trained by saving noise maps in the generation of the dataset, which is used as ground truth in the noise-estimation loss component. In training, the network switches between the real noise and synthetic noise batches, where only the latter includes the noise estimation loss. In their results, the authors show that they surpass other non-blind denoisers, trained on Gaussian noise only, for real image denoising benchmarks [38].

Another notable network which incorporates the U-Net backbone is MWCNN [202] (*Multi-level Wavelet-CNN*) which combines the efficient feature extraction of CNNs with the sparse representation space offered by orthogonality wavelets. Inspired by multi-level wavelet transforms, as described in Section 2.2.3, the U-Net is modified to replace the max-pooling downsample operations with 2D discrete Haar wavelet transforms, which reduces the image resolution via the decimation following the first-row wavelet transform. In place of a quantisation or thresholding step, the wavelet coefficients are passed to CNN layers to carry out the noise filtering. Other modifications to the U-Net include the replacement of residual concatenations with a summation for computational efficiency. The network is non-blind and trained for three noise levels.

Inspired by their success in natural language processing [214, 215] (NLP), many CNNs include attention mechanisms in their architectures to improve their performance, especially popular before the advent of vision transformers [50]. When applied to an image, the attention mechanism signals to the proceeding part of the network which part of the image to emphasise (spatial attention), or in the case of channel attention, which channel to emphasise. This is achieved by applying global average pooling across the channel dimension for spatial attention, followed by a convolution and sigmoid activation to scale the output. The ‘attention-map’ is then multiplied, element-wise, with the original feature map to complete the attention operation. In the case of channel attention, the operations are the same, but the global pooling is applied on each channel, across the spatial dimensions. The first image restoration network to use attention maps in this manner was the *Attention U-Net* network [216] (2018), and

other popular denoisers which use the method include MPRNet [208] (2021), MIRNet [217] (2020) and RIDNet [218] (2019).

2.3.2 Transformer Image Denoisers

Transformer networks currently occupy the state-of-the-art for denoising output quality today. Their development has been led by industry research groups with access to large, private datasets and substantial computational resources. In contrast to small CNNs, which consider the nature of the denoising problem, implementing task-specific features such as noise maps, transformers are trained with no knowledge of the problem and learn to model the space of real images from scratch. Proposed vision transformer architectures are usually designed and trained for a broad number of restoration tasks.

The first successful implementation of vision transformers was the classification network, ViT (Vision Transformer), published by Google Research in the 2021 paper, *An Image is Worth 16×16 Words: Transformers for Image Recognition at Scale* [50]. Before its publication, it was unclear if transformers could be successfully trained for image processing as they do not possess the inductive bias of CNNs, whose use of convolutions captures spatial hierarchies and translational invariances. As well as this, transformers have a computational complexity which is quadratic with respect to the number of input tokens. While NLP transformers may expect hundreds of words as input, a single 4K image contains more than 8 million pixels per channel. Lastly, in order to overcome the lack of an inductive bias, transformers require larger datasets to train.

To overcome the input size limitations of transformers, the Vision Transformer (ViT) network uses patch embeddings, where each 16×16 image patch is treated as an input token. Since the position of each embedding is unknown to the transformer, a learnable 1D positional embedding is added to each token to represent its position in raster scan order. The vision transformer network employs the same encoder architecture implemented by Vaswani et al. [214], where the queries, keys, and values are split into multiple heads using trainable linear transformations, as shown in Figure 2.11. For each head, scaled dot-product attention is computed between the key and query vectors, followed by applying the Softmax function to obtain the attention weights. These weights are then used to compute a weighted sum of the value vectors. The outputs from all heads are concatenated and passed through a final linear layer to combine the information. Multiple such multi-headed attention blocks are stacked, and the resulting representation is passed through a final linear layer to produce the classification result. The network was trained by Google research using an unpublished classification dataset of 300 million images with over a billion labels overall and proprietary

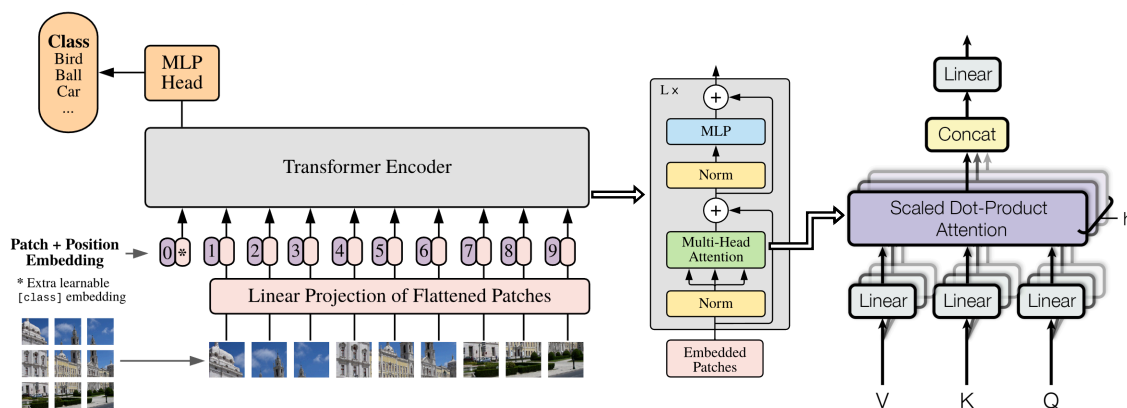


Fig. 2.11 Overview of the Vision Transformer (ViT) [50] architecture: On the left, the process of converting an image into token patches with positional embeddings is shown. In the middle, the transformer block includes residual connections, a multi-head attention stage, and a Multi-Layer Perceptron (MLP) processing the concatenated outputs. The rightmost image illustrates the internal workings of the transformer, where multiple heads operate in parallel, each with its linear transformations for the query, key, and value vectors.

V3 TPUs (Tensor Processing Units). The most powerful network proposed consists of 32 layers, 16 heads per layer and 632 million parameters in total.

Following the demonstrated feasibility of the Vision Transformer, the Shifted Window (SWIN) Transformer [34] by Microsoft was published, with a more efficient network, capable of attending to each pixel, and the ability to output segmentation maps. Instead of applying self-attention across all tokens in the image, the SWIN architecture applies windowed multi-headed self-attention (W-MSA) to reduce the complexity from being quadratic to linear with respect to the number of patches in a window, which is set to 7×7 by default. As well as this, the token size is reduced to 4×4 pixels ($C_{input} = 48$, excluding positional embeddings) to increase the spatial precision of the network. For connections to be made between patches located in non-overlapping windows, a shifted window partitioning scheme is implemented (SW-MSA), which alternates between two partitioning configurations in consecutive Swin blocks. In addition, for each consecutive layer, a feature pyramid network (FPN) is used, where 2×2 neighbouring feature maps are merged at the output of each transformer. This results in a hierarchical architecture where the first stage of the network has high spatial resolution and low feature resolution, and the final stage of the network has low spatial resolution and high feature resolution, allowing the network to capture detailed local information as well as broader contextual features. In the case of segmentation, the output at each stage is up-sampled to the original image resolution via bilinear interpolation or trainable transpose convolutions and combined to form the output mask.

Following the Swin classification network, image restoration transformers [44, 47, 48, 219] have been implemented, which now dominate the field of image denoising in quality comparison benchmarks, most notably, SwinIR [44] (2021) by Liang, Timofte et al. In their work, the authors take direct inspiration from the Swin classification network by reusing the same transformer blocks and arranging them inside cascaded Residual Swin Transformer Blocks (RSTB), which each contain six Swin transformer layers (STLs) and are terminated by a 2D convolution, which helps to enhance the network’s translational equivariance and enables channel mixing. The network is summarised by the RSTB number, STL number, window size number, channel number and attention head number as being set to 6, 6, 180 and 6, respectively. A single convolution is applied to the input image as a preprocessing shallow feature extraction step, which is combined with the output of the transformer layers for reconstruction of the final image using another, final convolution. Unlike the original Swin transformer, the pyramidal pooling is not used to preserve the image detail. SWINIR is the final benchmark and output sample included in this section as shown in Figure 2.3r. The amount of detail generated is again increased in this example, with more cheek textures, more markings on the beak, as well as feathers visible protruding on the neck below the beak.

Restormer [47], introduced by Zamir et al. in 2022, is one of the most advanced transformer denoisers among the current state-of-the-art. Unlike W-MSA blocks, Restormer uses channel-wise self-attention blocks (MDTA) to avoid the loss of global information caused by windowing, eliminating the need for tokens of grouped pixels. Depth-wise 2D convolutions are applied to the key, query, and value vectors before calculating the channel attention maps, enabling the network to learn spatial information from neighbouring pixels. The network uses a U-Net backbone, where feature maps are downsampled and upsampled in the encoder and decoder stages using pixel unshuffle and pixel shuffle, respectively. This approach allows for larger channel-wise self-attention maps and richer features in the bottleneck stage of the U-Net. A custom post-MDTA feed-forward network (GDFN) is used similarly to the Swin architecture to enhance the features extracted by the self-attention mechanism.

2.4 Video Denoising

Many image denoisers have been extended to video, such as the Wiener filter [138], Wavelet filtering [220, 221], and V-BM4D [155]. With the benefit of neighbouring frames, these methods have access to more information than image denoisers; however, precautions should be taken to account for the motion between frames to best align image blocks and ensure temporal consistency during playback. Examples of careful temporal preprocessing of frames include Kokaram’s advanced Wiener filter, which applies motion compensation [138, 222] by

determining the displacement vectors for two successive image frames and only qualifying a motion vector as real displacement if the MAE associated with the vector is not greater than a fraction of the mean absolute error between that block and the corresponding block in the other frame.

One of the best performing non-CNN video denoisers is Video Non-Local Bayes [32] (VNLB) proposed by Arias et al., in their paper *Video Denoising via Empirical Bayesian Estimation of Space-Time Patches (2018)*. In their work, the authors put forward a method for denoising by grouping sufficiently similar spatio-temporal blocks in a sequence and considering each as a random independent variable drawn from the same prior distribution. Assuming a Gaussian prior, Bayesian inference is used to estimate the clean patches p from the noisy patches q using the following formula:

$$\hat{p}_i = \mu + \frac{C}{C(C + \sigma^2 I_d)}(q_i - \mu), \quad (2.32)$$

where q_i and p_i are the i th noisy patch and predicted patches respectively, μ and C are the mean and covariance of the distribution estimated from the grouped samples, and σ^2 is the known noise variance. A key innovation in VNLB is the assumption of low intrinsic dimensionality of the patches, which leads to a low-rank covariance model. This assumption simplifies the estimation of the covariance matrix. This approach is essentially a form of the Wiener filter, where the patches are weighted based on their covariance, demonstrating the efficiency of Wiener filtering in separating signal from noise when a strong prior is available.

2.4.1 Deep Learning Video Denoisers

As with image denoising, in recent years, deep learning methods have overtaken the output quality of non-learning-based methods in video denoising tasks. In 2019, Tassano et al. introduced DVDNet [31], which outperformed VNLB [32] in high-noise scenarios. The proposed network architecture utilises a two-stage denoising network. The first stage denoises the $2T + 1$ input frames individually. Subsequently, each frame undergoes motion compensation with respect to the central target frame. The resultant frames are concatenated and passed through a second temporal denoiser to produce the output frame. DVDNet adopts trends from image denoising, such as an extra noise-map channel in both networks and residual connections akin to those used in ResNet and DnCNN. Additionally, four quarter-resolution inputs are employed in both networks, as seen in FFDNet. A faster denoiser, FastDVDNet [223], was also published by the authors. This version removes the costly motion-compensation step and introduces a new two-tiered network. Both networks accept five frames but reduce

them to three through three calls of the stage 1 denoiser. The second denoising stage then reduces these three frames to a single denoised target frame. FastDVDNet's larger networks increase the parameter count from 1.3M to 2.5M in total.

VideNN [224] followed DVDNet with another two-step network, using the backbone of DnCNN in its modules with residual connections, resulting in a 3.5 million parameter network. In the same year, VNLnet [225] was released by the authors of VNLB, which incorporates a non-trainable, non-local pixel matching step before the DnCNN denoising backbone. This step creates feature maps of pixels with the highest similarity in the spatio-temporal neighbourhood to the target pixel. Each pixel in the target frame is matched to the closest N patches in the spatio-temporal neighbourhood before passing through the network for denoising. PacNet [226] by Google Research (2021) is another two-stage network that exploits self-similarity in video by recreating the input frames with N "patch-craft" frames. These frames search within a spatio-temporal volume for the N most similar patches to sew together N new frames, which are subsequently passed through spatial and temporal denoisers.

Most recently, transformer architectures have been applied to video restoration tasks such as super-resolution [227–229], deblurring and denoising [230], video frame interpolation [231], spatio-temporal video super-resolution [232] and compressed video super-resolution [233]. *VRT: A Video Restoration Transformer* [36] (2024) by Liang et al. (of SwinIR), has emerged as one of the most popular video denoising transformer networks characterised by parallel frame warping and state-of-the-art benchmark scores, outperforming DVDNet [31] on average, by +2 dB across all datasets and noise levels.

Unlike SwinIR [44], VRT makes use of a U-Net architecture, which allows the network to handle object and camera motion at different scales with its custom TRSA (Temporal Reciprocal Self-Attention) and PW (Parallel Wrapping) modules. After each encoder stage, the spatial resolution is reduced by squeezing 2×2 neighbourhoods to the channel dimension, followed by a linear layer to restore the original channel width. As with many U-Net architectures, shallow features are extracted via a single convolution layer before the encoder stage, and deep features are extracted through the multi-scale layers of encoder and decoder layers.

Each stage of the U-Net is made up of TRSA layers which terminate with a parallel wrapping stage. Each TRSA layer accepts reference frame features, X^R and supporting frame features X^S , where $\{X^R, X^S\} \in \mathbb{R}^{H \times W \times N \times C}$. Instead of using self-attention within a single frame, the query, Q^r , is obtained as a linear projection of the reference frame, and the key K^S and value matrices V^S are found using linear projections of the supporting frame. The mutual attention

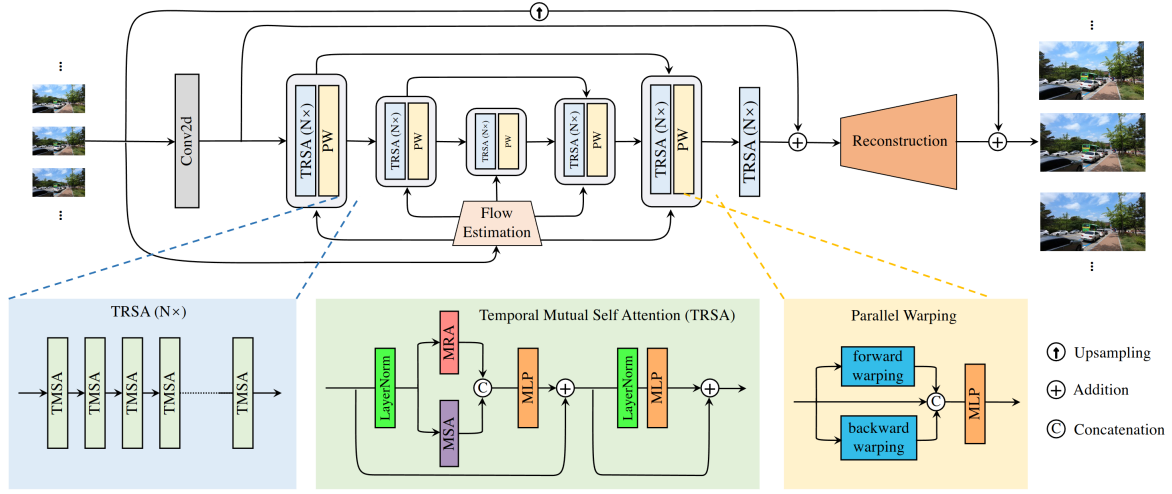


Fig. 2.12 The VRT network [44] by Liang et al. combines a U-Net backbone with transformer architecture for video restoration. The computation is made possible by dividing the attention into overlapping context windows and tokenising frames into 16×16 or 32×32 patches. Temporal Relation Self-Attention (TRSA) blocks align features across frames, capturing motion and temporal dependencies.

(MA) is found by:

$$MA(Q^R, K^S, V^S) = \text{SoftMax} \left\{ \frac{Q^R (K^S)^T}{\sqrt{D}} \right\} V^S,$$

Where $\text{Softmax}\{\cdot\}$ is a row Softmax operation which outputs an $N \times N$ attention map between the reference and supporting frame features. When the dot product of this attention map is applied to V^S , the output is effectively a map of weighted sums of features in the supporting frame which most closely resemble features in the same position in the reference frame. In other words, the features in X^S are brought closer to those in X^R . Similarly to multi-head self-attention, the attention is carried out h times in parallel, and the process is named Multi-head Reciprocal Attention (MRA). In addition to reciprocal attention, self-attention (MSA) is also applied separately to the two feature maps and concatenated with the output of the MRA block. The stacked features are reduced and passed through MLP layers for better feature extraction. For TRSA, the MRA blocks are stacked and the input frames are shifted by one after every MRA block. This means that at layer i , each frame can utilise information from up to $2(i-1)$ frames.

VRT also makes use of the shifted window mechanism [34, 44] to reduce computational complexity, which prevents the reciprocal attention mechanism from capturing larger motions. To improve on this, parallel warping is introduced, where the output of the final TRSA block

is reduced in terms of its channels, and a CNN-based model predicts the optical flow [234] and a deformable convolution [235] is used for alignment. Both forward and backward frames are warped to the target frame position before they are concatenated and output through an MLP for consolidation of the extra channels.

The denoising model proposed in the paper uses a 7-stage, 4-level U-Net including the bottleneck. Each stage consists of 8 TRSA modules with a channel dimension of 96, a patch embedding size of $1 \times 8 \times 8$ and a window size, M^2 , of 8×8 embeddings. The window size is increased for the final two TRSA modules of each stage to $8 \times 8 \times 8$. A final 8th stage is included before the reconstruction with 120 channels and 16 TRSA blocks. The network is trained as a non-blind denoiser and takes 10 days to train using $\times 8$ A100 GPUs.

2.5 Training for Real Noise

While early denoising networks such as DnCNN and FFDNet have been trained to remove Gaussian, additive noise, the recent introduction of real-noise benchmarks has shown that failing to account for the characteristics of real noise in training will result in suboptimal denoising for real-world images. These online benchmarks include the Smartphone Image Denoising Dataset [210] (SIDD) hosted by York University Toronto as well as the Darmstadt Noise Dataset [38]. One difficulty, however, with training a ‘real world’ denoiser is the lack of noisy image datasets. Such datasets are labour-intensive to create and few exist [39, 210].

2.5.1 Model-Based Noise Generation

As mentioned in section 2.3.1, the authors of CBDNet [40] proposed a model for creating realistic noisy samples from sRGB images by adding signal-dependent noise to the pseudo-raw image. As the ground truth raw file is not supplied, the authors create an image processing pipeline and pass sRGB images through it in reverse, applying custom CRF functions, colour balancing and colour space conversions, before creating a quarter-resolution Bayer mosaic from the image. The synthetic noisy images as modelled by CBDNet are summarised by:

$$y = \text{JPEG}(f(\text{DM}(L + n(L)))), \quad (2.33)$$

where f is the CRF, colour space conversions and gamma correction, DM is demosaicing, L is the raw pixel irradiance and $n(L)$ is the signal-dependent noise which is modelled by:

$$n(L) = n_s(L) + n_c, \quad (2.34)$$

where n_s is signal-dependent noise function, of variance σ_s^2 , and n_c is the stationary noise component, of variance σ_c^2 .

For each simulated noisy image, a camera profile is randomly chosen. These camera profiles are made up of a collection of look-up tables corresponding to different camera response functions [236]. Training with this acceptably realistic real noise surpasses models trained with Gaussian noise only. As such, this noise model has become popular in recent networks doing real-world denoising.

2.5.2 Machine-Learning Based Noise Generation

Instead of modelling the in-camera signal processing (ISP) pipeline by hand, it has been more recently proposed to train a network to generate specific noise profiles or noise patterns. Given a noisy image and an acceptably denoised prior, the Pseudo ISP model [42, 237] trains three networks to reproduce noise profiles:

1. A ‘sRGB to RAW’ network, which takes the input noisy image and converts it to a pseudo-raw space. This is followed by a mosaicing filter.
2. A noise estimation network which estimates a noise profile for the clean raw image. This sub-network uses the raw noisy image as ground truth in its loss function.
3. A ‘RAW to sRGB’ network which learns to convert the image back to sRGB from the previous raw space. The network receives four Bayer images as input and outputs the full resolution sRGB image

Refer to Figure 2.13 for an illustrated view of the network. The network trains to recreate both the noisy image and the clean image at the same time, and both results are used in the loss functions for the ‘sRGB to RAW’ and ‘RAW to sRGB’ networks.

Once trained, the model is used to create a new noisy dataset whose noise profile closely corresponds to the image or images on which they were trained. PseudoISP is geared towards fine-tuning an existing denoiser by capturing the noise of the target dataset. It requires an estimated ‘pseudo-clean’ image prior and is computationally expensive to compute. Pseudo ISP has recently topped denoising benchmarks by fine-tuning networks such as MWCNN and MPRNet.

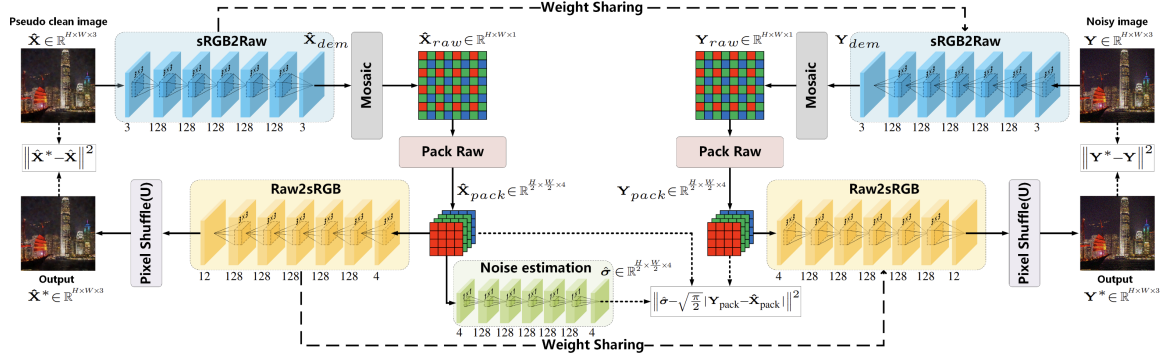


Fig. 2.13 The Pseudo ISP Architecture, as proposed by Cao [237], is designed to replicate a camera’s in-camera signal processing (ISP) and noise characteristics to generate new noise samples, expanding the corpus of real noise images for training purposes. The architecture involves three Convolutional Neural Networks (CNNs): one for sRGB to RAW conversion, another for RAW to sRGB conversion, and a third for estimating the noise map in the RAW space. By learning the camera’s ISP and noise model, this system can synthesise realistic noisy images from clean ones. Pseudo-ground-truth images, produced by an auxiliary denoiser, enable the network to form noisy-clean training pairs, ultimately allowing it to refine the pseudo-clean image back into a more accurate representation of the real noisy sample.

2.6 Using Denoisers as a Universal Prior

Originally inspired by non-equilibrium physics [238], diffusion probabilistic models learn the reversal of a diffusion process to generate a tractable solution for modelling complex data, such as real images. While the original model learned a score function to guide the reverse Markov chain model, it has been shown in *Denoising Diffusion Probabilistic Models* [239] that directly predicting the noise added at each step of the diffusion process simplifies the learning objective and improves the quality of generated samples. This approach leverages a discrete-time Markov chain for both the forward process, which gradually adds noise to the data, and the reverse process, which denoises it, resulting in a powerful generative model capable of producing high-fidelity images from a completely noisy image. Generative diffusion models [240, 241] have outpaced Generative adversarial networks (GANs) in image fidelity scores such as FID [242] and the introduction of text-guided Stable-Diffusion by Rombach et al. [71] has brought about new landmark tools for image generation such as OpenAI’s Dall-E [243–245] and Google’s ImageGen [246, 247].

Simoncelli and Kadkhodaie arrived at a similar tractable solution for image synthesis [45] in their work *Solving Linear Inverse Problems Using the Prior Implicit in a Denoiser* (2021) by extracting the prior from a denoising network and creating a framework to synthesise images from noise. Their extraction of the image prior from a neural network relies on

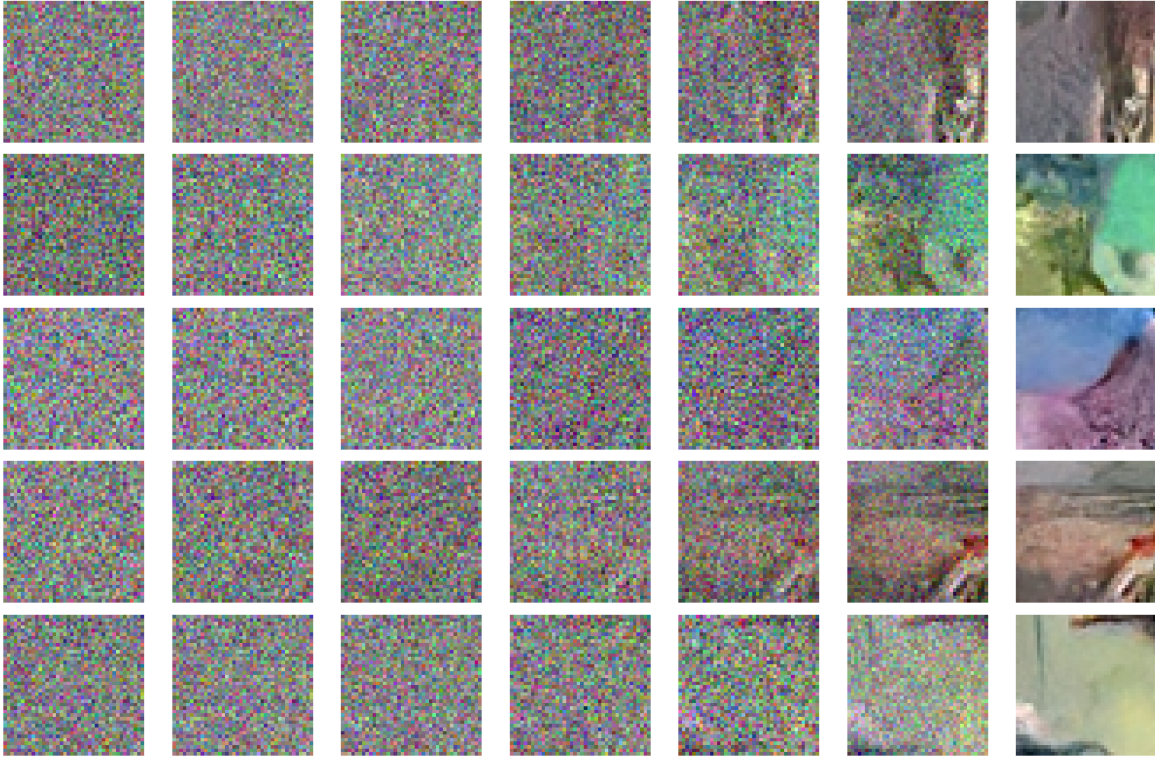


Fig. 2.14 Natural Image Synthesis as described by Kadkhodaie and Simoncelli. Each result starts from a completely noisy image and is progressively denoised and re-noised until a sufficiently smooth output is reached. In this example, we save every tenth image, as well as the final image.

a statistical result found by Miyasawa, which observes that the MMSE estimate $\hat{x}$ from a Gaussian denoiser (again, given $y = x + n$), showed the following property:

$$\hat{x} = y + \sigma^2 \nabla_y \log p(y). \quad (2.35)$$

This equation is of significance as it allows us to unlock the denoisers prior by capturing it within the image residual, $\hat{x} - y$, showing it to be proportional to the gradient of $\log p(y)$, where $p(y)$ is the prior distribution of the image. The authors demonstrate that by starting from an initial image of pure noise x_0 , and iteratively denoising and adding noise, we can take steps in the direction of increasing the image log prior, and eventually arrive at a plausible image, as shown in Figure 2.14. It is noted that $p(y)$ is not the prior of the clean image; however, as the noise in the image is reduced to zero ($\sigma \rightarrow 0$), the density $p(y)$ converges to $p(x)$. The original authors describe how convergence is controlled by starting at a high noise level ($\sigma_0 = 255$) and reducing the noise to a minimum value ($\sigma_L = 2$), stopping the process once a desired smoothness has been met.

Kadkhodaie and Simoncelli also create a framework for solving linear inverse problems using this prior. Applications considered include image super-resolution, inpainting, and compressed sensing, where 90% of the data is removed and the goal is to restore colour images from sparse measurements. Given some linear constraints or measurements:

$$x^c = M^T x, \quad (2.36)$$

the aim is to sample images which maximise the constrained prior $p(x|x^c = M^T x)$. Equation 2.35 then becomes:

$$\sigma^2 \nabla_y \log p(y|x^c) = (I - MM^T)f(y) + M(x^c - M^T y), \quad (2.37)$$

where $(I - MM^T)$ is the projection of y onto the orthogonal complement of the measurement subspace. Classically, the matrix M^T is assumed to have orthogonal columns and thus M is its pseudo-inverse. The log gradient is computed at each step, and the predicted image y_t is updated with a mix of the log gradient and a newly generated Gaussian noise map $n \sim \mathcal{N}(0, \sigma I)$:

$$y_t = y_{t-1} + h_t d_t + \gamma_t n_t, \quad (2.38)$$

where d_t is the log gradient for iteration t , h is step size or rate of ascent and γ decreases from σ_0 to σ_L , controlling the noise added in each iteration. The algorithm requires a blind denoiser trained on AWGN to minimise the MSE from a large amplitude range of noise, typically at least $\sigma \leq 100$. In Kadkhodaie and Simoncelli's case, they rely on their previously proposed bias-free denoiser [198] (Mohan et al.), which takes the DnCNN architecture and removes the bias term from the convolution as well as the batch-normalisation. This work demonstrated that denoisers showed an improved capacity to generalise to a larger range of noise in this configuration.

We note that the use of a denoiser as a prior in linear inverse problems was proposed before the popularisation of deep learning by Venkatakrishnan and Bouman. In their paper *Plug-and-Play Priors for Model-Based Reconstruction* [248], they introduced a two-step iterative method that reformulates the regularisation problem into an algorithm that alternates between solving the data fidelity term and applying a denoiser, where the degradation model is known. This work was adapted for image super-resolution [249] by Brifman et al. in 2016 and used with CNN priors for the first time in 2017 by the same authors as DNCNN (Zhang et al), in the paper *Learning Deep CNN Denoiser Prior for Image Restoration* [72]. Here, a new DNCNN-like network with strided convolution is trained for denoising, and results show

high-quality results in deblurring and super-resolution tasks. A new version of the network also exists with U-Nets and additional inverse tasks [250].

2.7 Conclusions

CNNs are now able to denoise images to a greater standard than traditional spatial and Bayesian methods, which rely on the pixels presented to them and the assumptions made in their design. Training datasets have become as important as the models themselves, and the previous Gaussian testing criteria have been found to be obsolete. All of these factors have brought about challenges in the field.

Comparing Networks of Different Sizes. As with many fields in deep learning, there is a tendency for new networks to showcase good performance by rolling the latest trends into new backbones. It is becoming increasingly difficult to ascertain whether the improvements reported are due to such novel features or the associated size increase of the network. Improvements in denoising performance are often backbone-centric, and the contributions of noise-specific models are unclear. In Chapter 3, we break down the performance gains that may be attributed to a larger network, and those which are attributed to noise-specific features.

Noise Modelling. It is difficult to optimise synthetic datasets for a target noisy image. Despite this, many neural networks are trained only on Gaussian noise and achieve acceptable results without considering the effectiveness of noise model optimisation. Perhaps modelling the noise after real image noise as in CBDNet would be a better focus for denoising images and would allow smaller networks to generalise comparably well to larger networks. This aspect of network training is evaluated in Chapter 3.

Lack of User Controllable Parameters. It is often not possible to modify the output of a pre-trained denoiser. The recent success of the multi-wavelet denoiser [202] has shown that classic denoising methods can still be relevant in denoising today, and it is worth exploring the possibility of further leveraging classic, well-optimised denoisers with modern CNNs to give the user more control. In Chapter 4 we present a framework for Wiener image denoising which incorporates deep learning techniques to enhance the filter performance while maintaining a small network size and rapid inference timings. In Chapter 5, we present a Wiener denoising network for video denoising.

Blind Video Denoising Current state-of-the-art video denoisers are not blind and rely on the user to input a measure of the noise for best results. Transformer video denoisers are also demanding in terms of computational resources and require tiling of the input video at standard 1080p resolution to avoid exceeding the GPU memory capacity. In Chapter 5 we also focus on creating a fast and memory-efficient blind denoiser which also surpasses the performance of current non-blind denoisers.

Unexplored Implicit Prior of the Image Denoiser. As discussed in Section 2.6, the hidden implicit image prior of deep learning denoisers can be extracted [51] from the image residual and applied to any linear inverse image enhancement problem. However, the literature on this subject has not been thoroughly explored. In Chapter 6, we explore the benefit of training a single denoiser model to solve multiple tasks.

Chapter 3

Optimising Denoising Networks for Real Noise

In Chapter 3, we described the rapid progress in denoising performance in recent years, as demonstrated by online real-noise benchmarks, such as SIDD and DND. Some of these improvements have made use of noise-specific techniques, including the creation of more realistic synthetic noise datasets, and the implementation of noise maps to enrich the input data. This development cycle has been accelerated by the use of powerful backbones from fields of computer vision, which are quickly adopted in the denoising literature. For instance, U-Nets [200], originally implemented for image segmentation, have been utilised to create significantly larger denoising networks in terms of trainable parameters. More recently, transformer networks - taking inspiration from Dosovitskiy’s classification model [50] - have achieved the highest benchmark scores. However, these networks are limited in their usefulness due to the amount of memory they require during inference, restricting their use to systems with high-capacity graphic cards. Moreover, the extent to which noise-specific features in these larger architectures contribute to overall denoising remains unclear.

In this chapter, we conduct a detailed assessment of these denoising features to determine their importance in the denoising outcome. Our study aims to inform future denoiser design, focusing on maximising performance - especially in low-power applications which may restrict network size. A version of this work was published as *Assessing Advances in Real Noise Image Denoisers* [54], and was presented at the ACM SIGGRAPH European Conference on Visual Media Production (CVMP) in December 2022. Here, we have updated the data presented in tables and figures to reflect the latest research in the field.

Network	Year	Params. (M)
DnCNN [30]	2017	0.56
FFDNet [43]	2018	0.48
RIDNet [218]	2019	1.50
CBDNet [40]	2019	4.34
DANet [254]	2020	9.15
AINDNet [255]	2020	13.76
MWCNN [203]	2020	16.14
MIRNet [253]	2020	31.78
NBNet [256]	2021	13.30
ResUNeXt [127]	2021	53.58
U-former [48]	2022	89.46
NAFNet [257]	2022	116.00
KBNet [258]	2023	141.97
CGNet [259]	2024	119.22

Table 3.1 Publication year and parameter count of denoising networks presented in Figure 3.1.

3.1 Introduction

In recent years, the field of image denoising has turned its attention towards the evaluation of denoisers on real-noise data, which has been facilitated by online benchmarking systems such as the DND and SIDD benchmarks outlined in Section 2.1. This has motivated many to develop increasingly complex networks to achieve best-in-class denoising performance. Many of these networks do not account for the specific properties of digital camera noise, opting instead to retrain state-of-the-art architectures, such as large U-Nets [127, 208, 251–253] and more recently, transformer networks [35, 44, 48] to predict the clean image $\hat{x}$ from the observed noisy image y . To illustrate this trend, Figure 3.1 presents a graph comparing the number of trainable parameters in real-noise denoising networks with their denoising quality in terms of PSNR (dB). Additionally, Table 3.1 shows the publication year of these networks, along with their parameter count in millions of parameters. It is clear from this data that in recent years, denoisers have become larger in terms of their parameter count and, as a result, more computationally demanding.

In the case of denoisers which implement noise-specific architectural features, it is difficult to ascertain the extent to which their increased denoising performance may be attributed to those features directly, or indirectly by the increased number of trainable parameters added by the feature. One popular example of such a feature is the noise-map, which acts as an extra input channel to the main network, containing information about the quantity of noise in the image. One such implementation was used by FFDNet [43], in which the user provides a single standard deviation value to the network. CBDNet [40], later proposed a blind denoiser

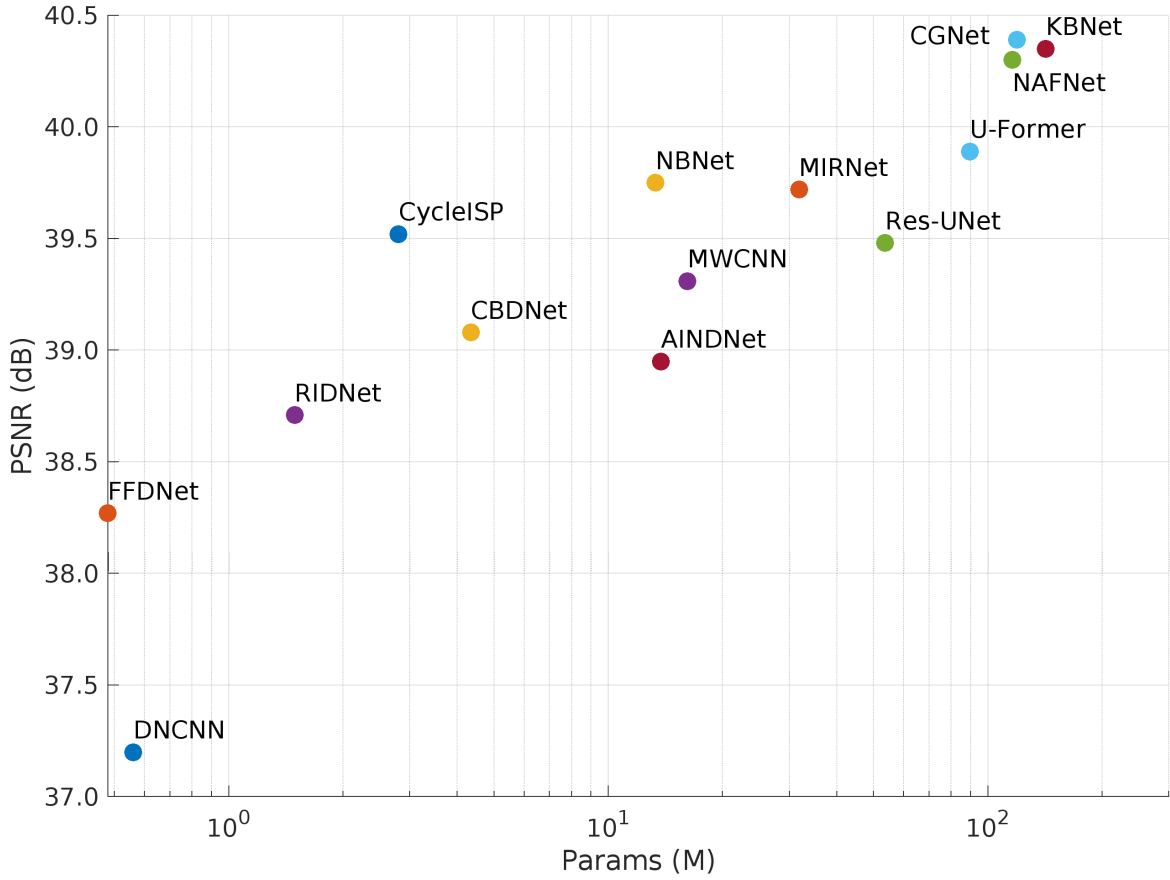


Fig. 3.1 Sample of denoising results taken from the SIDD real noise benchmark, measured in PSNR (dB), along with network size, measured in the number of trainable parameters.

which uses an ancillary network for noise map generation, as discussed in Section 2.3.1. Unlike FFDNet, which repeats the same number over the entire noise map channel, the noise map produced by CBDNet is a per-pixel estimate of the input image noise. At the time of its publication, CBDNet demonstrated excellent denoising results on the DND benchmark; however, it is unclear whether this can be attributed directly to the noise-map feature or the increased size of the overall network brought by the ancillary network.

Another challenge specific to training real-noise denoising networks lies in the type of data used for training. Until recently, denoisers were primarily trained and evaluated using Additive White Gaussian Noise (AWGN), which does not accurately represent real noise. In practice, as discussed in Section 2.5, real noise behaves quite differently; it is non-stationary and comprises both signal-dependent and independent components, such as photon noise, dark noise, read noise and Fixed Pattern Noise (FPN). Denoisers that do not account for these differences and continue to train under the assumption of AWGN often yield lower quality results when applied to real-noise data. However, high-quality, diverse, real-noise datasets are

scarce. To address this gap, these datasets have been supplemented by synthesised real-noise models such as that proposed by the authors of CBDNet.

In the following chapter, we aim to establish guidelines for the optimisation of real-noise denoising networks with a series of ablation studies. We firstly introduce a baseline backbone network in Section 3.2, which will serve as the foundation for our experiments. In Section 3.3.1, we investigate how the impact of different noise types (real, Gaussian, synthetic) in the training data on denoising performance for real benchmarks, and assess whether combining multiple noise types during training can further enhance performance. Next, in Section 3.3.2 we evaluate how the size of the network affects denoising performance, focusing on whether the largest network, in terms of parameters, always delivers the best results, or if smaller networks can achieve comparable quality. The reduction of denoiser size is especially important in terms of energy preservation, as well as the portability of a denoiser to low power devices. Following this, in Section 3.3.3 we evaluate whether incorporating an ancillary noise mapping module, similar to that used by CBDNet, improves denoising quality, or if an equivalently sized network without any noise mapping module can achieve similar results. Lastly, in Section 3.3.3, we train U-Net style networks with the same training parameters as our optimised network to compare their performance against one another.

3.2 Proposed Testing Platform

In this section, we outline the testing methodology applied to each of our experiments. This includes the baseline architecture used, the training data available to us, the benchmarks used to evaluate denoising performance, and the parameters used in training the network.

3.2.1 Testing Architecture

To conduct controlled experiments, we use a common baseline network for evaluation. We propose here to work with the DnCNN [30] backbone, a well-established and lightweight network that has demonstrated success in Gaussian denoising, being among the first networks to outperform BM3D. In its original configuration, the network consists of only 0.64 million trainable parameters, making it both efficient to train (in under a day) and rapid in inference. Unlike larger architectures such as U-Nets or transformers, DnCNN has low memory requirements during inference, making the architecture suitable for local denoising of high-resolution images on low-power devices. While the original DnCNN has shown high potential, its simple architecture may now benefit from recent advancements in real image denoising, as discussed in this Section.

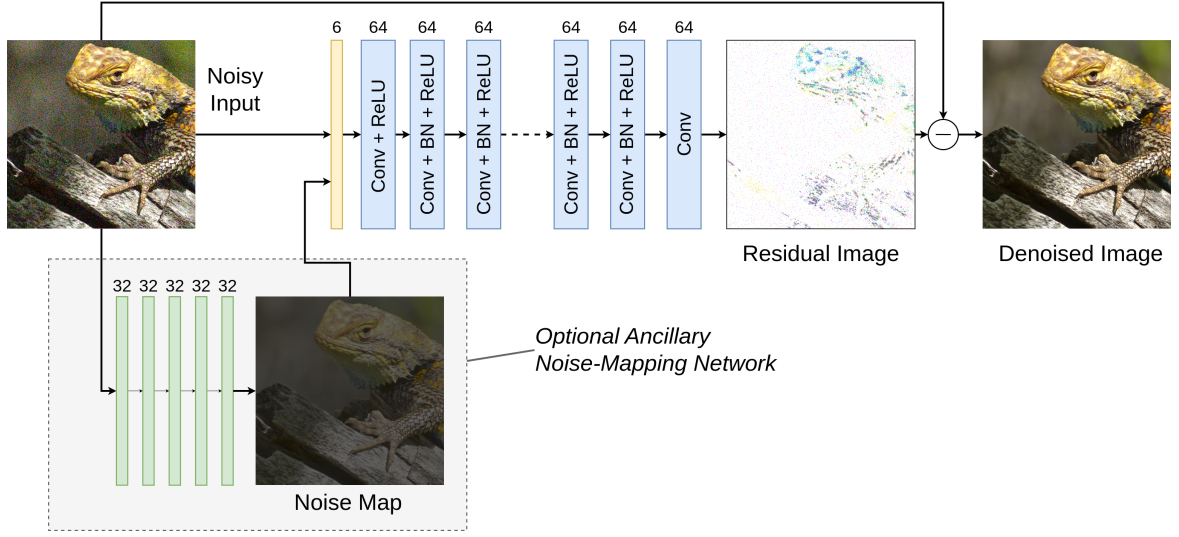


Fig. 3.2 Our DnCNN implementation. The green network is the ancillary noise map prediction network, whose output is concatenated with the input noisy image to form a 6-channel input tensor to the twenty-layer DnCNN network (in blue). The output of DnCNN is the residual noise image which, added to the input image, produces the output denoised image.

We configure our DnCNN network (as shown in Figure 3.2) to be 20 blocks deep, with each block being made up of a 2D 3×3 convolution, a 2D batch normalisation and a ReLU activation function. The input layer is the only layer not to use a batch normalisation operation, and the output layer does not include an activation function. A skip connection links the first and final layer of the network via a pixel-wise summation operation. As a result, the network outputs a residual noise image which is added to the original image to produce a clean output. Figure 3.2 also includes an optional ancillary noise mapping network, which we have added in our implementation for future experiments. This will be discussed in further detail in Section 3.3.3.

3.2.2 Training Datasets

In our experiments, we have three types of noisy data: real noise images, synthetic noise images and Gaussian (AWGN) noise images.

Synthetic Camera Noise. Using images from the DIV2K (800 images) and the MIT-Adobe FiveK [213] (5000 images) datasets, we have generated a training dataset of 3000 images with synthetic noise by adopting the noise generation scheme of CBDNet [40]. The images are converted from sRGB to a pseudo-RAW format using randomly selected inverse camera

response functions and Bayer filters. Signal-dependent noise is applied to the raw images, and they are then converted back to sRGB by reversing the initial process.

Real-Noise. There are few real-noise datasets available. These datasets are typically obtained by capturing the same scene at different ISO levels, where low ISO levels create the ground truth data, and high ISO images create the noisy images. In our experiments, we included images from the SIDD dataset [37]. The dataset consists of 320 image pairs taken using smartphones. Ten scenes are recorded under different brightness conditions and illumination temperatures.

Gaussian Noise. AWGN was also considered to assess the possible benefits of including a subset of Gaussian noise to a larger training set and also establish a baseline for a training purely based on Gaussian noise. For this dataset, 1000 unused images from the MITfiveK dataset was used, with Gaussian noise of standard deviation of $\sigma \in [0, 30]$ randomly chosen for each image during the data-loading process.

3.2.3 Evaluation Benchmarks

We evaluate denoising quality for the following experiments using the SIDD and DND real noise online benchmarks, both of which only provide the user with the noisy images. Both benchmarks provide a measurement of output quality in terms of PSNR (dB) and SSIM ([0-1]).

The Smartphone Image Denoising Dataset [37] (SIDD) consists of 1280 noisy image blocks, taken from 40 images, with 32 non-overlapping crops taken from each image. This benchmark is hosted by York University, Ontario, Canada.

The Darmstadt Noise Dataset [38] (DND) consists of 1000 image blocks, taken from 50 images, with 20 non-overlapping crops taken from each image. This benchmark is hosted by the Technical University of Darmstadt, Germany.

3.2.4 Training Regime

Each network is trained for 5000 epochs from sets of real, user-generated, and Gaussian datasets. Every epoch, a random 96×96 crop is taken from each image in the training set. An Adam optimiser was used with an initial learning rate of 1×10^{-3} which decayed to 1×10^{-6} using a cosine annealing learning rate function.

3.3 Ablation Studies

This section explores the influence of dataset balance, network size, and noise maps on the performance of real-noise denoising.

3.3.1 Balance of Noise Types in Training Data

As previously mentioned, few real-noise datasets exist, and it is only in the last few years that deep learning denoisers have turned their attention from Gaussian noise experiments to real noise problems. Real noise datasets are time-consuming and labour-intensive to create and are limited to a few controlled scenes of still objects. This is because the camera must capture the same scene multiple times, once at a low ISO level to capture the ground-truth image, and subsequently at higher ISO levels and shorter exposure times, forcing the amplification circuits to compensate for the lower amount of light received, which in turn generates noise. The camera should maintain the same focus between shots, and the data should be spatially aligned afterwards. Real-noise datasets include Renoir [39], the Smartphone Image Denoising Dataset [210] and ‘See-in-the-Dark’ [260], which collectively contain 534 unique scenes in total.

Due to the low number of real noisy-clean image pairs, recent works have proposed to synthesise real noise using camera ISP models. Such a method has been used by CBDNet (as discussed in Section 2.5.1) where a clean sRGB image is transformed to a pseudo-raw format by applying ISP functions in reverse, before adding noise to the image. Both a constant noise component is added to the raw Bayer image, as well as a multiplicative, signal-dependent component, which is scaled by the constant component. This constant noise component is AWGN (zero mean) and is measured by its standard deviation, often denoted by the letter sigma, σ . Once the noise has been introduced to the raw image, the ISP functions are applied in the forward direction to arrive at the original image with added noise. The ISP functions simulated by the synthetic noise generator include Bayer demosaicing, white balancing, colour space conversion and randomly selected camera response functions. Sample images from this model-based noisy image generation are shown in Figure 3.3.

While the authors of CBDNet discuss how training with different types of synthetic noise affects denoising performance, there is no consensus on how many real-noise images should be included in training sets versus synthesised noisy images. Current networks that use different combinations of real-synthetic training data include GRDN [261], RIDNet [218], DANet [254], AINDNet [255] and PT-MWRN [46]. In our first ablation study, we evaluate

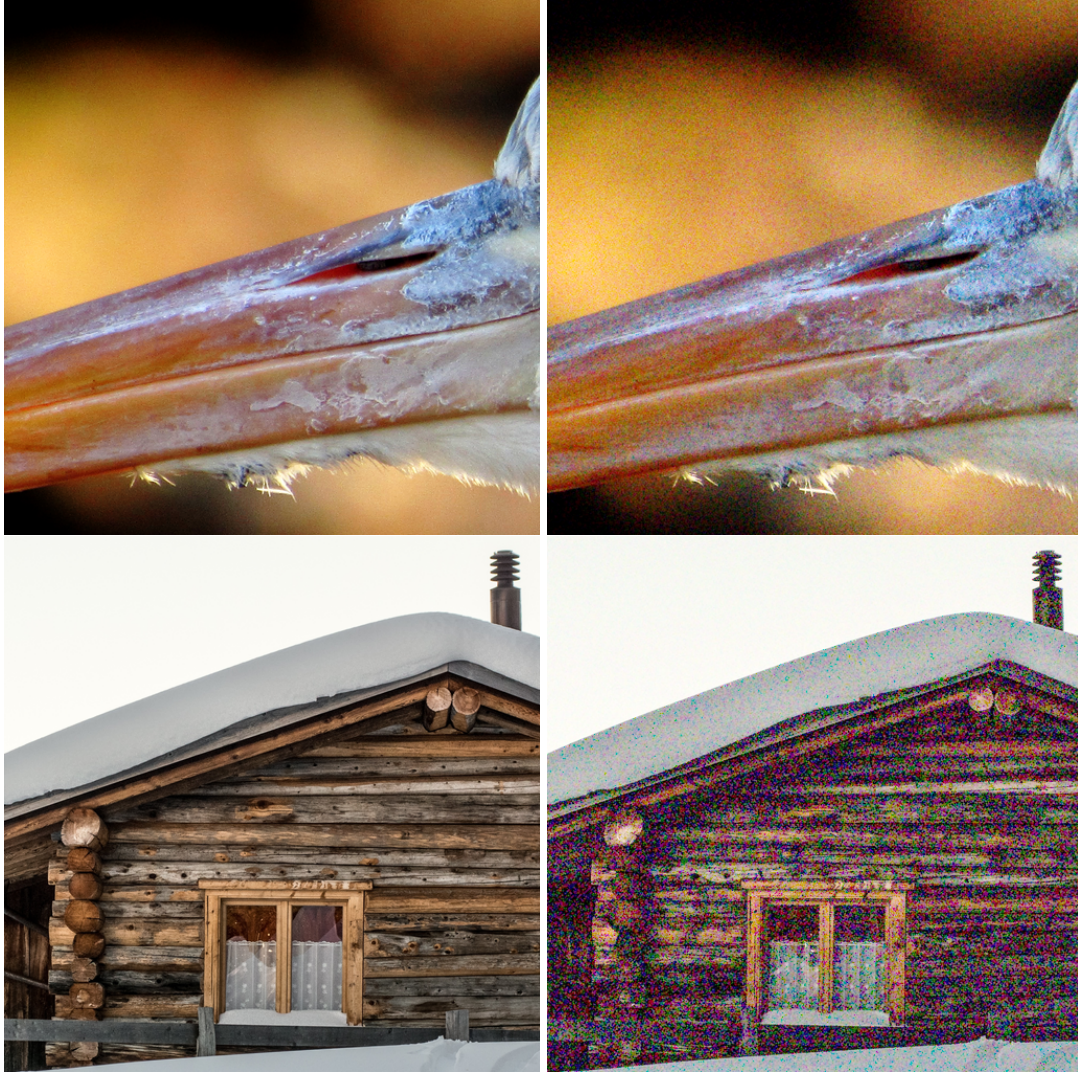


Fig. 3.3 Noise samples using CBDNet [40] noise generator. Left: original, Right: synthetic camera noise.

the performance impact of training our denoiser on different types of noise (Gaussian, real, pseudo-real), as well as using a combination of noise types.

In this ablation study, we evaluate how different noise types in the training set affect image quality and examine the impact of incorporating a mixture of noise types in the overall dataset. Specifically, we compare the performance of the DnCNN network trained from scratch under the following scenarios:

1. Gaussian noise only.
2. Real noise only.

	Gaussian/Synthetic/Real Mix				
	100/0/0	0/0/100	0/100/0	0/91/9	10/83/7
PSNR	34.31	36.97	38.67	39.09	39.02
SSIM	0.8649	0.9248	0.9503	0.9514	0.9519

Table 3.2 Denoising results on the DND benchmark using the DnCNN backbone network. Each experiment represents the same network trained with a different mix of Gaussian, synthetic and real noise. The network is configured to have 64 feature-maps per layer.

3. Synthetic noise only.
4. A mix of Synthetic and real noise (91/9 split).
5. A mix of Gaussian, Synthetic and real noise (10/83/7 split).

Results

We present our results for this section in Table 3.2. For each network, an average PSNR and SSIM score is returned. Each trained model is evaluated on the same blind benchmark (DND). We see that the AWGN trained network performs the worst, with a PSNR of 34.41 dB, while a combination of synthetic and real noise training perform the best in terms of PSNR, at 39.09 dB. Interestingly, the denoiser trained with synthetic noise only, significantly outperforms the real-noise trained denoiser, with a difference of 1.7 dB between them. This may be explained by the smaller size of the real noise dataset, as well as the variety of subject matter in the synthetic noise dataset. Another observation we make is that, although the AWGN dataset contains more than twice the number of images in the real noise dataset, it’s performance was 2.66 dB worse. This illustrates the ineffectiveness of AWGN training for real photography. Lastly, the incorporation of AWGN into the synthetic-real dataset dropped the final PSNR by 0.07 dB, but increased the SSIM result by 0.0005. This suggests that adding Gaussian noise to a dataset may rank the denoiser’s performance higher in subjective metrics, but note that the margin is too small to definitively make this conclusion.

It is clear that training on purely synthetic noise yields some over-fitting, and that mixing this noise with Gaussian or, better, real noise, improves the overall performance.

3.3.2 Impact of Network Size

As new architectures vary wildly in size and number of parameters, in our second ablation study, we propose to assess the impact of the network size on the performance of our DnCNN. In the following experiment, we increase the size of our DnCNN denoiser by increasing

the number of feature maps per convolutional layer. By doubling the number of channels in each layer in this manner, the number of total trainable parameters increases by a factor of four. Six DnCNN networks are created with 32, 64, 96, 128, 256, and 512 channels per convolution block, with sizes ranging from 0.17 million parameters (DnCNN-32) to 42.51 million (DnCNN-512).

Results

In Table 3.3 we present the results of our denoising experiments, which assess the impact of increasing the number of features per layer in our backbone network. The experiments are evaluated using the SIDD test set. Following the findings of Ablation study 1, each network was trained on the same mixture of synthetic and real noise datasets with a 91:9 split, for 5000 epochs.

Overall, the highest performing network, in terms of PSNR, was our 256 channel network, with a PSNR of 38.44 dB. The 512 channel network performed worse, despite having over 32 million additional parameters. This is most likely due to over-fitting in training. The 512 channel network equalled the result of the 128 channel network at 38.35 dB for the second best performing network. As expected, the 32-channel network, with only 0.17 million parameters, gave the lowest PSNR score of 37.41. Despite the smaller size of the network, this score outperforms BM3D, a method once considered to be state-of-the-art.

In terms of SSIM, the 128 channel network scored the highest with a score of 0.946. The 512, 256 and 96 channel networks gave the same, second highest SSIM score of 0.945. The SIDD benchmarking service reports SSIM to three significant figures, making it impossible to separate these scores. Again here, the 32 channel network scores the lowest with a SSIM of 0.936.

From this experiment, we observe that increasing the network size generally improves performance, although gains diminish due to overfitting beyond a certain point. For example, increasing the width from 64 to 256 channels improves PSNR from 38.02 dB to 38.44 dB, but increases the parameter count from 0.67 to 10.36 million. This suggests that while network size plays a role, other architectural factors may contribute more significantly to performance.

3.3.3 Use of Noise Maps in Image Denoising Networks

Noise maps (see Figure 3.2), as recently popularised by CBDNet [40], are implemented with the goal of estimating the amount of noise in the image and sharing that information with the main network, as an extra channel input, to guide the denoising process. In the case

Network Width	PSNR	SSIM	Parameters (Millions)
32 Channels	37.41	0.936	0.17
64 Channels	38.02	0.943	0.67
96 Channels	38.25	0.945	1.53
128 Channels	38.35	0.946	2.67
256 Channels	38.44	0.945	10.36
512 Channels	38.35	0.945	42.51

Table 3.3 Denoising results from the network size ablation study conducted on the SIDD benchmark. The number of feature maps in each layer of the backbone increases across experiments, with each network retrained from scratch. The table includes the PSNR, SSIM, and the number of trainable parameters for each network configuration.

of CBDNet, an ancillary network is used to generate the noise map from the input noisy image during inference. Unlike FFDNet, this noise map provides a per-pixel estimate of the noise variance. This aligns more closely with the characteristics of real noise, whose noise distribution is not stationary. This feature brings with it the additional advantage of creating a blind denoising network which does not require the user to input an estimate of the noise.

While noise maps have been adopted in multiple works [40, 206, 207, 262], clear ablation studies are not presented to demonstrate their benefits in real image denoising. FFDNet reports marginal improvements over DnCNN in Gaussian colour denoising tasks at low to medium levels of noise, with the largest improvements showing for noise of standard deviations of $\sigma = 75$. This level of degradation is much greater than that typically encountered in real photography. CBDNet is evaluated only using noise maps, making it difficult to assess whether the network’s performance is due to the addition of synthetic noise data in training, the noise prediction sub-network, or both. The ancillary network also carries additional trainable weights to the network, which may directly contribute to the denoising process. As such, we wish to evaluate the impact of noise maps on denoising performance.

In our third ablation study, we implement our own ancillary noise estimation network and add it to our denoising backbone to evaluate whether an improvement in denoising quality is observed. The network is comprised of only five layers, with each layer being made up of a 3×3 2D convolution, followed by a ReLU activation function. Each layer uses 32 feature maps, which results in a final parameter count of only 29,507. Like CBDNet, the output of this ancillary network is passed as an additional input to our backbone network, which is modified to accept a 6-channel input, instead of 3 - 3 channels from the noisy input image and 3 channels from the noise map received from the ancillary network.

To train the ancillary network, ground truth standard deviation maps are collected during the creation of the synthetic noise dataset. The synthetic noise dataset is used to train the network, and the corresponding noise maps are used to generate a loss. As described by the authors of CBDNet, we train our ancillary network denoiser using a split loss function [40] which sums the denoised image loss with the predicted noise map loss:

$$\mathcal{L}_{Total} = \mathcal{L}_{Denoised} + \lambda \mathcal{L}_{Noisemap}, \quad (3.1)$$

where λ is a weight applied to the noise map loss, which we set to 0.5. The denoised image loss is simply the L2 loss between the predicted image, $\hat{x}$ and the ground truth image, x :

$$\mathcal{L}_{Denoised} = \|\hat{x} - x\|_2^2. \quad (3.2)$$

The noise map loss function is asymmetric, further penalising the loss when the predicted noise, $\hat{\sigma}$, is less than the ground truth noise σ :

$$\mathcal{L}_{Noisemap} = \sum_i |\alpha - \mathbb{I}_{(\hat{\sigma}(y_i) - \sigma(y_i)) < 0}| \cdot (\hat{\sigma}(y_i) - \sigma(y_i))^2, \quad (3.3)$$

where $\mathbb{I}$ returns 1 at every pixel where the predicted noise standard deviation $\hat{\sigma}$ is less than the ground truth noise standard deviation. α is a scaling factor which weights the importance of the error. α is a penalising factor for the asymmetric loss.

We evaluate the performance of the ancillary network by assessing its performance with and without the noise-map loss, to establish if specifically learning the noise maps changes the denoising outcome. We also create an extra control experiment, where the size of the backbone is increased to match the size of the network configured with the ancillary network. A second supporting ablation study is also carried out.

Following our previous ablation studies, we configure our backbone to have 96 channels per layer and train using the synthetic-real noise mix. The impact of the noise mapping module is assessed by arranging three different training scenarios. Firstly, our network is trained with the ancillary module, using the split loss described in Equation 3.3. Secondly, the same network is trained with a single loss L2 loss function which simply compares the predicted image to the ground-truth clean image as in Equation 3.2. Next, a control network is trained, where the size of the backbone is increased to match the size of the network configured with the ancillary network.

A second supporting ablation study is also carried out to assess whether the use of a noise map may become redundant once the denoising network becomes sufficiently large. We

Network	PSNR	SSIM
96 Channel, Trained Noise Map	38.67	0.949
96 Channel, Untrained Noise Map	38.27	0.946
97 Channel, No Noise Map	38.28	0.946
Original Network	38.25	0.945

Table 3.4 Denoising results from the noise map ablation study. The network is evaluated across three configurations and a control: ‘Trained Noise Map’—the network learns to produce a noise map via a split loss function; ‘Untrained Noise Map’—the network is trained using only the denoised image loss, without supervision for the ancillary network; ‘97 Channels, No Noise Map’—the size of the original DnCNN is increased from 96 to 97 channels to match the ancillary module, with the ancillary network removed; and ‘Original Network’—the performance of the original network without the ancillary network.

complete this study by training and evaluating the network repeatedly at increasingly large network sizes. At each stage, the network is compared to its non-noise-map counterpart to evaluate whether a significant difference in performance is observable at each level.

Results

We present our results in Table 3.4 for the SIDD benchmark. We evaluate the noise map performance with a targeted noise map loss (*96 Channel, Trained Noise Map*), without the noise map loss (*96 Channel, Untrained Noise Map*), with an equivalently sized denoising network which does not use a noise mapping network (*97 Channel, No Noise Map*) and the original network for comparison (*96 Channel, No Noise Map*).

It is clear to us from the presented results that training the network with the ancillary noise map estimator to predict the noise maps using the full loss function provides a significant performance boost in terms of PSNR, with a +0.39 dB against similarly sized networks that do not exploit the ground-truth noise maps, i.e. ‘Untrained Noise Map’ and ‘97 Channel, No Noise Map’. Thus, the benefits of using a noise map do not stem from the additional parameters brought by the ancillary network but indeed by exploiting the ground-truth noise profiles. The differences in terms of SSIM are more difficult to discern due to the low precision provided by the online benchmarking service.

We continue the study by evaluating networks with 32, 64, 128 and 256 channels with and without noise maps. The networks with noise maps are trained using the dual loss function method. Results are presented in Table 3.5. The performance of the 32-channel, 64-channel, 96-channel and 128-channel networks are improved by the ancillary network configuration. The most significant improvement in performance is noted with the 32-channel network (+

	32 channel	64 channel	96 channel	128 channel	256 channel
W/o Noise-map	37.41	38.02	38.25	38.35	38.44
W/ Noise-map	38.09	38.13	38.67	38.38	38.36

Table 3.5 Denoising results for the denoising backbone, with and without the noise mapping module, evaluated on the SIDD benchmark. In each experiment, the size of both networks is increased by expanding the number of feature maps per convolution.

0.68 dB). This suggests that the noise map can be useful in further improving the performance of smaller denoisers, at low cost. The gains obtained for very large networks are not so clear, with only + 0.03 dB at 128 channels and even - 0.08 dB at 256 channels. This result suggests that once a network becomes sufficiently large, the performance benefits of noise maps become redundant. This is further motivated by the closeness of both the noise map and non-noise map 128 channel networks (0.03 dB). The lesser performance of the 256-channel noise map network may be accounted for by its loss function being weighted towards the now redundant noise map.

3.4 Training of Larger Networks and Final Results

To complement our ablation studies and establish an upper bound on performance with more sophisticated architectures, we introduce two novel networks into the benchmark that have not previously been applied to image denoising. These networks, implemented and trained as part of this study, serve as additional controlled benchmarks for larger models trained using the same methodology and optimised dataset as our lightweight DnCNN network.

The first network, ResUNeXt, is a custom design inspired by recent U-Net models that incorporate residual connections within CNN blocks alongside traditional skip connections between encoder-decoder sections. In this implementation, the ResNet encoder found in larger U-Nets is replaced with a ResNeXt [263]-style encoder. ResNeXt extends the ResNet architecture by splitting convolutions into smaller parallel blocks, distributing features evenly among them. This approach, defined by the "cardinality" of the network, retains comparable complexity to the original ResNet while delivering superior performance, as demonstrated in ImageNet classification tasks. For our ResUNeXt, the encoder is modelled after the ResNeXt-50 variant, balancing complexity with computational feasibility. However, U-Net-based networks, including ResUNeXt, are significantly larger than lightweight alternatives like DnCNN.

The second network, U2-Net, is adapted from an architecture originally designed for object detection [209]. In U2-Net, each stage of the encoder and decoder functions as a smaller U-Net, as illustrated in Figure 3.4. At the deepest levels of the architecture, where image resolution becomes too small for traditional downsampling, dilated convolutions are employed instead. Unlike conventional U-Nets, U2-Net produces intermediate outputs at each decoder stage. These outputs are upsampled, concatenated with outputs from other stages, and passed through a final convolutional layer to produce the final image.

For our implementation, we closely follow the original U2-Net architecture, modifying only the decoder layers to output a 3-channel image at each stage. Additionally, during training, we leverage the upsampled outputs of all decoder stages, incorporating them into the overall loss function to guide the denoising process effectively.

3.4.1 Final Results

In our final results, we compare the performance of our optimised denoiser against several state-of-the-art networks, including BM3D, FFDNet, U-Net, CBDNet, RIDNet, MWCNN, and U-Former. Each network is evaluated using PSNR (dB) and SSIM on two widely-used test sets: DND and SIDD. This comparison highlights the relative performance of our approach in the context of existing high-performing methods.

The results are summarised in Table 3.6 and Table 3.7, which report the PSNR and SSIM values for our U2-Net and ResUNeXt models alongside several previously evaluated DnCNN variants. For context, we include performance metrics for some of the best-performing networks on these test sets, such as MWCNN and U-Former. Additionally, network sizes are reported in terms of millions of parameters, providing insight into the trade-offs between performance and computational cost.

For the DND benchmark, ResUNeXt outperforms DnCNN-96 by 0.2 dB and 0.0025 SSIM, while U2-Net outperforms the same network by 0.14 dB and 0.0037 SSIM. Although ResUNeXt is comprised of almost twice the parameters of U2-Net (88.6 vs. 44 M), the performance gap between these networks is minimal, with only a 0.06 dB difference. As well as this, U2-Net outperforms ResUNeXt in the SSIM measurement for the same dataset, despite being a smaller network. This may be explained by the larger size of ResUNeXt, which may require more data and far longer training times to reach its maximum denoising potential. This indicates network size is not always the most crucial factor in achieving higher performance, though it can be beneficial, as demonstrated in our network size ablation study. The significance of architecture design and optimisation is further highlighted by the fact

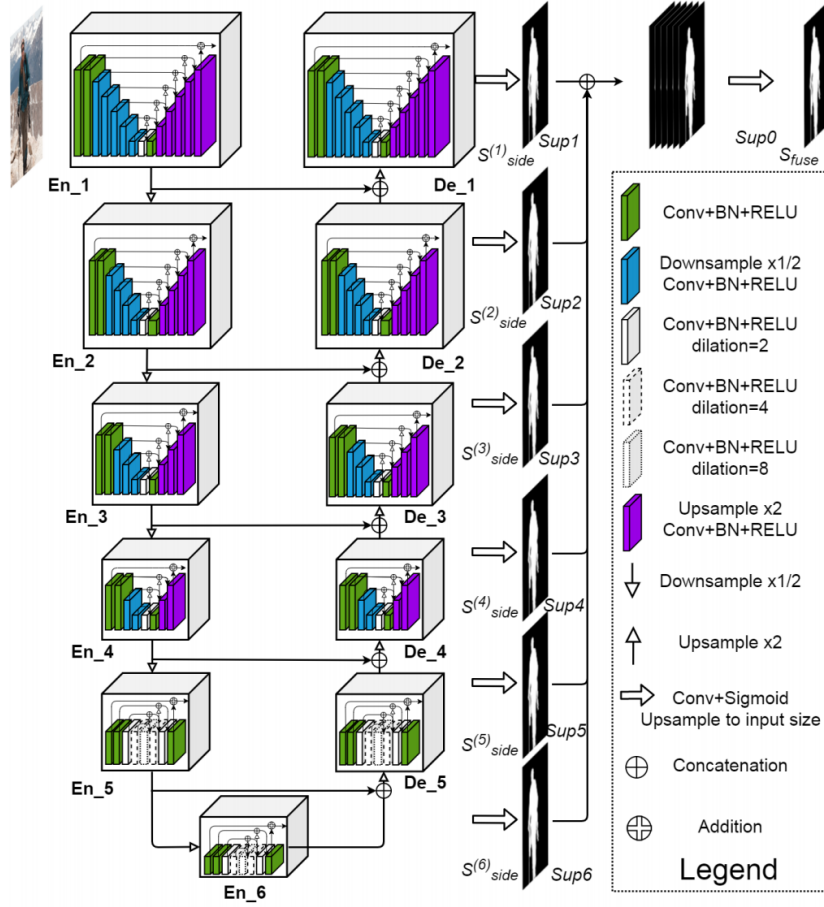


Fig. 3.4 The U2-Net architecture, originally proposed by Qin [209], retrained in this study for denoising. Each stage of the encoder and decoder is a U-Net in itself. The output of each decoder stage is further reduced to a 3-channel output, contributing to the final image reconstruction and the loss function.

that our best-performing DnCNN network is 59 times smaller than ResUNeXt and 29 times smaller than U2-Net, yet trails in performance by only 0.2 dB. For both the SIDD and DND benchmark, our DnCNN implementation outperforms all other submissions with the same backbone (not listed in the tables below). Notably, for the DND benchmark, our DnCNN-64 network trained on mixed noise using the noise map module outperforms CBDNet, U-Net (original) and RIDNet - networks that are significantly larger and were recently regarded as state-of-the-art.

U-former’s denoising performance (39.98 dB/0.9554 SSIM) shows greater performance than we have been able to achieve thus far. This may be attributed to its powerful transformer backbone network. Like U2-Net and ResUNeXt, U-former is a large network (50.88 M parameters), indicating that there may still be gains to be made from our larger networks.

Method	PSNR	SSIM	Size (M)
Noisy Image	29.84	0.7018	n/a
BM3D	34.51	0.9308	n/a
FFD-NET+	37.61	0.9415	0.85
<i>DNCNN-64 on Gaussian</i>	34.31	0.8649	0.64
<i>DNCNN-64 on mixed</i>	36.97	0.9248	0.64
U-Net	38.58	0.9467	30.0
CBDNet	38.06	0.9421	4.36
RIDNET	39.25	0.9528	1.50
<i>DNCNN-64 on mixed + noise map</i>	39.26	0.9530	0.64
<i>DnCNN-96 on mixed + noise map</i>	39.35	0.9530	1.51
<i>ResUNeXt</i>	39.55	0.9555	88.6
<i>U2-NET</i>	39.49	0.9567	44.0
MWCNN	39.51	0.9526	16.14
U-former	39.98	0.9554	50.88

Table 3.6 Final table of results on the DND benchmark. Networks shown in italics were trained in this study, while those not trained here are sourced from the official DND online benchmark. The network size is reported in millions of parameters.

Lastly, the networks we trained tend to rank lower on the SIDD benchmark. We believe this is primarily due to the similarity between the provided SIDD training set and the SIDD test set. As a result, our training scheme is at a disadvantage compared to networks exclusively trained on SIDD data. This highlights a potential limitation of the SIDD benchmark. In contrast, the DND benchmark appears to provide a more balanced evaluation, as it incorporates a greater diversity of data.

3.5 Discussion

In reviewing our results, we make several observations that will guide our future denoiser design. First, when training a denoiser for real-world photographic noise removal, it is essential to maximise the amount of realistic noise in the training dataset. As shown in our results (Section 3.3.1), training exclusively with Gaussian noise led to a performance drop of over 4 dB, despite using the same network architecture. In addition to this, while the number of images available in real-noise training sets is low, they can be supplemented with the use of realistic noise generation models to further improve results.

Through our evaluation of the noise mapping module, it has been shown that improving the performance of deep learning denoisers is not solely dependent on building larger and more

Method	PSNR	SSIM	Size (M)
Noisy Image	23.70	0.480	n/a
BM3D	34.34	0.911	n/a
FFD-NET+	38.27	0.948	0.85
<i>DnCNN-64 on Gaussian</i>	35.32	0.877	0.64
<i>DnCNN-64 on mixed</i>	37.59	0.938	0.64
U-Net	38.88	0.952	30.0
CBDNet	34.00	0.868	4.36
RIDNET	38.71	0.952	1.50
<i>DnCNN-64 on mixed + noise map</i>	38.13	0.944	0.64
<i>DnCNN-96 on mixed + noise map</i>	38.67	0.949	1.51
<i>ResUNeXt</i>	38.06	0.942	88.6
<i>U2-NET</i>	38.68	0.951	44.0
MWCNN	39.31	0.956	16.14
U-former	39.89	0.958	50.88

Table 3.7 Final table of results on the SIDD benchmark. Networks shown in italics were trained in this study, while those not trained here are sourced from the online benchmark leaderboards. The network size is reported in millions of parameters.

complex architectures or maximising the training dataset size. By addressing the nature of the problem more closely, and incorporating features that guide the denoiser, rather than relying on it to independently identify the optimal solution, we can achieve enhanced denoising performance. Furthermore, we have shown that as a network becomes sufficiently large, the impact of the noise map on performance is diminished or completely negated. However, at this point, the network can no longer be considered lightweight.

In our second ablation, we highlight the importance of evaluating the full performance potential of the backbone to assess its capabilities and balance computational cost against performance. By increasing the network size, we explore how far performance can be improved before diminishing returns set in, while still maintaining a reasonable model size. In our results, we achieve within 0.1 dB of the largest network we could evaluate, using a model with 26 times fewer parameters.

Our final ablation study shows that there is some performance to be gained by the larger networks on the DND benchmark (+ 0.2 dB for ResUNeXt and + 0.14 dB for U2-Net); however, the DnCNN network actually outperforms ResUNeXt on the SIDD benchmark, as shown in Figure 3.5. As well as this, the highest performing implementation of DnCNN (DnCNN-96 w/noise-map), comes within 0.02 dB of U2-Net on the same benchmark. In

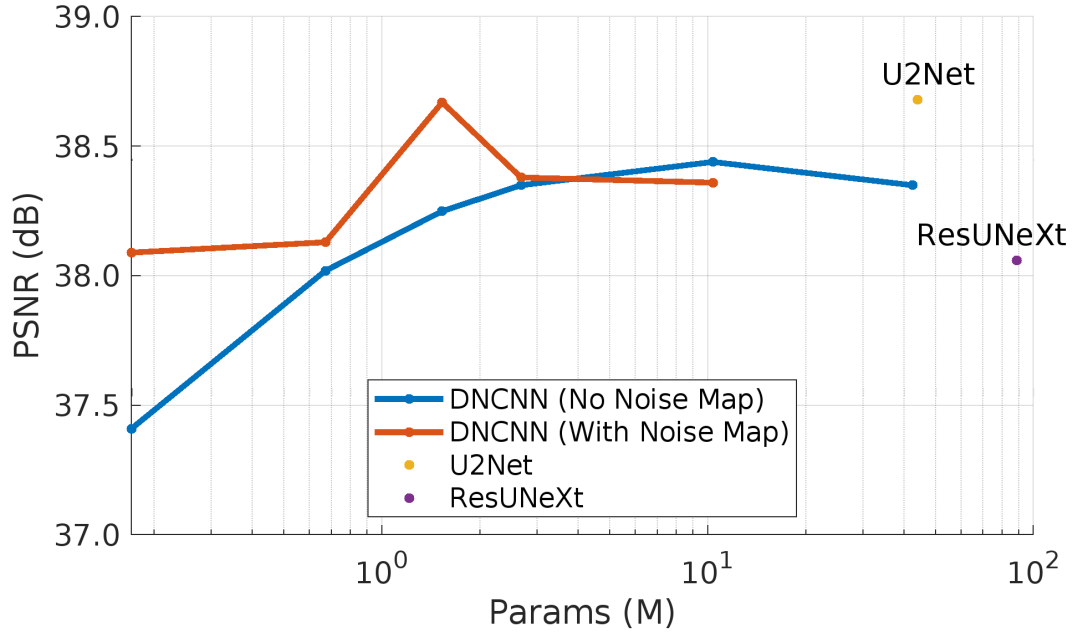


Fig. 3.5 Comparison of the DnCNN backbone performance with high-performing SIDD benchmark denoisers, U-Former and MWCNN. The number of parameters (in millions, log-scaled) is plotted against PSNR (in dB) to illustrate the balance between cost and performance.

Figure 3.6 we show example outputs of DND denoising results on the same networks (U2-Net, ResUNeXt and DnCNN-96 w/noise-map).

Should we continue this study, we would further investigate the noise generation models. While the CBDNet model is sophisticated, some of the camera profiles and noise patterns it produces are based on film photography data, which is not the target of our benchmarks. Furthermore, new methods such as PseudoISP [237] learn to convert compressed images to their pseudo-raw Bayer format and return a raw image to its compressed form following the addition of noise. This pipeline allows for the learning of a specific camera’s noise profile with only noisy images. Once the noise profile is learned, the network may be used to synthesise a new real noise dataset from clean images. Another observation from this work is that some benchmarks may favour over-trained networks on their own closely related training sets, making it harder to differentiate between denoisers. This tends to favour larger networks that model specific datasets rather than general-purpose denoisers.



Fig. 3.6 Denoising results from the original DND benchmark, showing 128×128 crops of denoising outputs. From left to right: Original Noisy, U2-Net, ResUNeXt, DnCNN-96.

3.6 Conclusions

In this chapter, we have investigated several aspects related to targeting real noise in neural networks.

Firstly, we have highlighted the importance of training from a combination of real and synthetic noise.

Secondly, we have shown that a simple DnCNN backbone, with only 0.64 million parameters, when properly trained and optimised, can outperform the denoising performance networks of more complicated recent networks. This includes the original U-Net denoiser (30 M parameters) and CBDNet (4.36 M parameters). We have also shown that DnCNN-96 is only within 0.63 dB of the state-of-the-art network U-former on the DND benchmark and within 1.2 dB on the SIDD benchmark.

Thirdly, we studied the benefits of estimating noise maps. We found that adding an ancillary noise map estimation network did improve performance (by up to +0.6 dB). We also established that this was not due to the increased network size. We also observed that the benefits of using noise maps become less noticeable as the network gets larger.

Fourthly, we trained two novel larger networks, ResUNeXt and U2-Net, of similar size to the highest performing denoising networks available today and found further increases in denoising performance, with U2-Net SSIM results are currently within the top 20 DND scores. A comparison with the top-scoring denoisers also shows that there is still a small margin of performance to be gained, which cannot be simply explained by the network size alone. The backbone architecture has therefore still an impact (of about +1 dB).

While our DnCNN-based backbone is not as sophisticated as more recent, larger networks, such as our ResUNeXt and U2-Net implementations, we conclude this chapter having shown that we have been able to come quite close to matching their performance. We have also demonstrated that the burden of learning the denoising process placed on the network may actually be reduced by leveraging denoising-specific features. By doing so, the network remains lightweight but more closely approaches state-of-the-art performance.

In the following chapters, we further explore this strategy to design small yet powerful denoising networks. Rather than relying on complex architectures with millions of parameters, we incorporate a classic denoising technique—the Wiener filter—into the denoising backbone. This approach leverages the strengths of an established denoising pipeline, providing a robust foundation that enhances the network’s performance even before training begins.

Chapter 4

The Wiener Image Denoising Network

Today, the state-of-the-art in image denoising is dominated by large deep-learning architectures capable of learning sophisticated models for clean image estimation. These models continue to grow in size and now outperform classic statistical solutions, including the Wiener filter, the wavelet filter and more complex non-local adaptations such as BM3D. Unlike these classical signal processing methods, neural networks begin training as a black-box with little understanding of the problem other than the inductive biases of a 2D-CNN and learn to denoise through the supervised learning process. By creating architectures that scale well with more trainable parameters, such as U-Nets and transformers, denoising results have improved.

In this chapter, we revisit the Wiener filter and propose to carefully re-implement it as the backbone of a denoising network. By starting with an established denoiser, we aim to enhance its performance using neural networks rather than developing a solution from scratch, with the goal of keeping the denoiser lightweight. First, the Wiener filter is re-implemented using the PyTorch deep-learning framework, before the associated preprocessing pipeline is optimised to maximise denoising quality. With this foundation in place, we introduce two compact CNNs into the Wiener filter framework. The first CNN transforms the Wiener filter into a blind denoiser suitable for real image denoising, and the second refines the estimated Wiener frequency transform function $\hat{H}$. Through this iterative design approach, we present an improved, parallelised Wiener filter that can outperform larger denoising networks such as DnCNN.

A version of this work was presented under the title *Pushing the Limits of the Wiener Filter in Image Denoising* at the IEEE International Conference on Image Processing in October 2023.

4.1 A Review of the Baseline Wiener Filter

The Wiener filter provides an optimal solution for image denoising by minimising the mean square error (MSE) between the predicted image and the true image, assuming additive noise and second-order stationarity for both the signal and noise. As derived in Section 2.2.2, when the original image, x , is unknown, the Wiener transfer function for the IIR case can be expressed as:

$$\hat{H}[\omega_i, \omega_j] = \frac{\max(P_{YY}[\omega_i, \omega_j] - \lambda \cdot P_{NN}[\omega_i, \omega_j], 0)}{P_{YY}[\omega_i, \omega_j]}, \quad (4.1)$$

where $P_{YY}[\omega_i, \omega_j]$ represents the Power Spectral Density (PSD) of the observed image y , and $P_{NN}[\omega_i, \omega_j]$ is the PSD of the noise n , derived from the user input noise STD. The scaling factor, λ (typically ranging from $1 \leq \lambda \leq 1.8$) is applied to P_{NN} to maximise the PSNR.

The filtered image is obtained by multiplying the Wiener transfer function coefficients element-wise by the noisy image's coefficients:

$$\hat{X}[\omega_1, \omega_2] = \hat{H}[\omega_1, \omega_2] \cdot Y[\omega_1, \omega_2], \quad (4.2)$$

before converting the image back to the spatial domain to yield the image estimate $\hat{x}[i, j]$. In regions of the frequency spectrum where the signal-to-noise ratio (SNR) is high, the Wiener filter allows the signal to pass with minimal attenuation. Conversely, in regions where the noise dominates (low SNR), the filter attenuates those frequencies to suppress the noise.

Although the Wiener filter assumes stationarity in both the image and the noise, this is rarely true for real-world images, as the mean and variance can vary significantly across different regions of the image. Additionally, the noise may not always be purely additive or stationary. Therefore, the Wiener filter is typically applied to smaller image blocks (e.g., 16×16) where the stationarity assumption holds more closely [27, 128, 264]. This windowing process, along with preparing the image for the DFT, forms a preprocessing pipeline that must be carefully managed to achieve the highest quality denoising results.

In Algorithm 1, we present an example baseline Wiener filter pipeline. In this example, a sliding window processes one block at a time, in a raster scan pattern across the image, with a half block's width overlap between each block. For every block in the image, the mean is estimated and subtracted from the block. This removes the DC component, which improves the frequency domain representation by eliminating the large low-frequency bias. The zero-mean block then has a windowing function applied to it to address the non-periodicity of real images, removing the edge discontinuity from the block. This prevents the leakage of

Algorithm 1 Baseline Wiener for a Raised Cosine Window**Require:** Noisy image, y , noise STD σ , Block Size

```

1:  $w(h, k) \leftarrow \text{RaisedCosine}(h, k)$  ▷ windowing definition
2: for all blocks  $y$  in image  $y$ , for stride=BlockSize/2 do
3:    $\bar{y} \leftarrow \text{mean}(y)$  ▷ predicts block mean
4:    $y_w \leftarrow (y - \bar{y}) \odot w$  ▷ windowing
5:    $Y \leftarrow \text{FFTn}(y_w)$ 
6:    $P_{yy} \leftarrow Y \odot Y^*$ 
7:    $P_{nn} \leftarrow \hat{\sigma}^2 \|w\|^2$ 
8:    $P_{xx} \leftarrow \max(P_{yy} - P_{nn}, 0)$  ▷ coring
9:    $\hat{x}_w \leftarrow \text{iFFTn}(Y \odot P_{xx} \oslash P_{yy}) + \bar{y}w$ 
10:   $\hat{x} \leftarrow \text{overlap\_add}(w \odot \hat{x}_w)$  ▷ combine blocks

```

erroneous high-amplitude, high-frequency components into the spectrum we wish to analyse. This window is known as the analysis window.

Next, a 2D Fast Fourier Transform (FFT) is applied to the windowed data y_w , obtaining its frequency spectrum Y . The power spectral density (PSD) of the noisy signal P_{YY} is calculated as the product of the frequency representation Y and its complex conjugate Y^* . The power of the input noise P_{NN} is then derived using the square of the input noise standard deviation σ and scaled by the squared norm of the raised cosine window, ensuring that the noise power is appropriately weighted by the window function.

The numerator of the Wiener filter is calculated as the difference between the signal and noise power, $P_{XX} = \max(P_{YY} - P_{NN}, 0)$, ensuring that negative values are set to zero to prevent negative power estimates ("coring").

An inverse FFT is then applied to the filtered frequency-domain data, weighted by the ratio of the estimated clean signal power to the noisy signal power $P_{XX} \oslash P_{YY}$ (where $\oslash$ is the element-wise division operator). The result is the estimated denoised block $\hat{x}_w$, which is converted back to the spatial domain, and the previously subtracted mean is added back to restore the original signal amplitude.

In line 10, the synthesis or reconstruction window $w(h, k)$ is applied to the inverse-transformed block to ensure that overlapping regions are smoothly blended during reconstruction. This windowing step is crucial for mitigating boundary artefacts and achieving seamless transitions between adjacent blocks. The final image is assembled using the overlap-add method: each windowed block is added to its corresponding location in the output image, and the contributions are aggregated. The result, $\hat{x}$, is a denoised reconstruction of the original noisy input. A 1D example illustrating this process for a noisy square wave is shown in Figure 4.1.

In summary, the frequency-selective filtering of the Wiener filter results in smooth outputs in low frequency areas, while edges and areas of high image activity, such as textures, have much less attenuation applied to them, preserving image detail and edges. As a bonus, the human visual system is less sensitive to noise in regions of high activity, so the noise in these regions is less easily observed. Additionally, the Wiener filter has smooth transitions in its frequency response, resulting in a reduction in the Gibbs phenomenon [140] and minimising ringing artefacts.

However, at high noise levels ($\sigma > 20$)¹, the performance of the Wiener filter is significantly surpassed by advanced denoising networks like DnCNN, which excel at restoring details that are otherwise lost. Under such high noise conditions, the Wiener filter tends to over-smooth textures and introduce noticeable ringing artefacts, as shown in Figure 4.2. The following section will outline a series of optimisations and strategies developed to enhance the denoising capabilities of the Wiener filter, including the integration of neural networks to improve the coring process.

4.2 Optimisation of the Baseline Wiener Filter

In this section, we describe the steps we have taken to improve the performance of the Wiener filter, starting with several basic optimisations to the preprocessing pipeline, to the transformation of the filter to a hybrid convolutional neural network which refines the filtered image coefficients and automatically selects the noise standard deviation for the user.

4.2.1 Implementation Framework

As of the time of our research, no GPU-accelerated implementation of the image Wiener filter was freely available. Existing implementations [27] compute the solution for each block in raster scan order, processing a single image at a time. This is time-consuming and does not take advantage of the parallelisation offered by modern Graphics Processing Units. As well as this, to enable the training of neural networks using the Wiener filter as a backbone, the Wiener filter must be implemented using differentiable functions which support back-propagation. For these reasons, we have rewritten the Wiener filter using the PyTorch deep learning framework for Python.

Our implementation eliminates loops and processes all blocks, colour channels, and batches of images in parallel. Before splitting the image into patches, the image is padded, adding

¹Here, σ is the noise standard deviation, and we assume a standard pixel intensity range of [0-255].

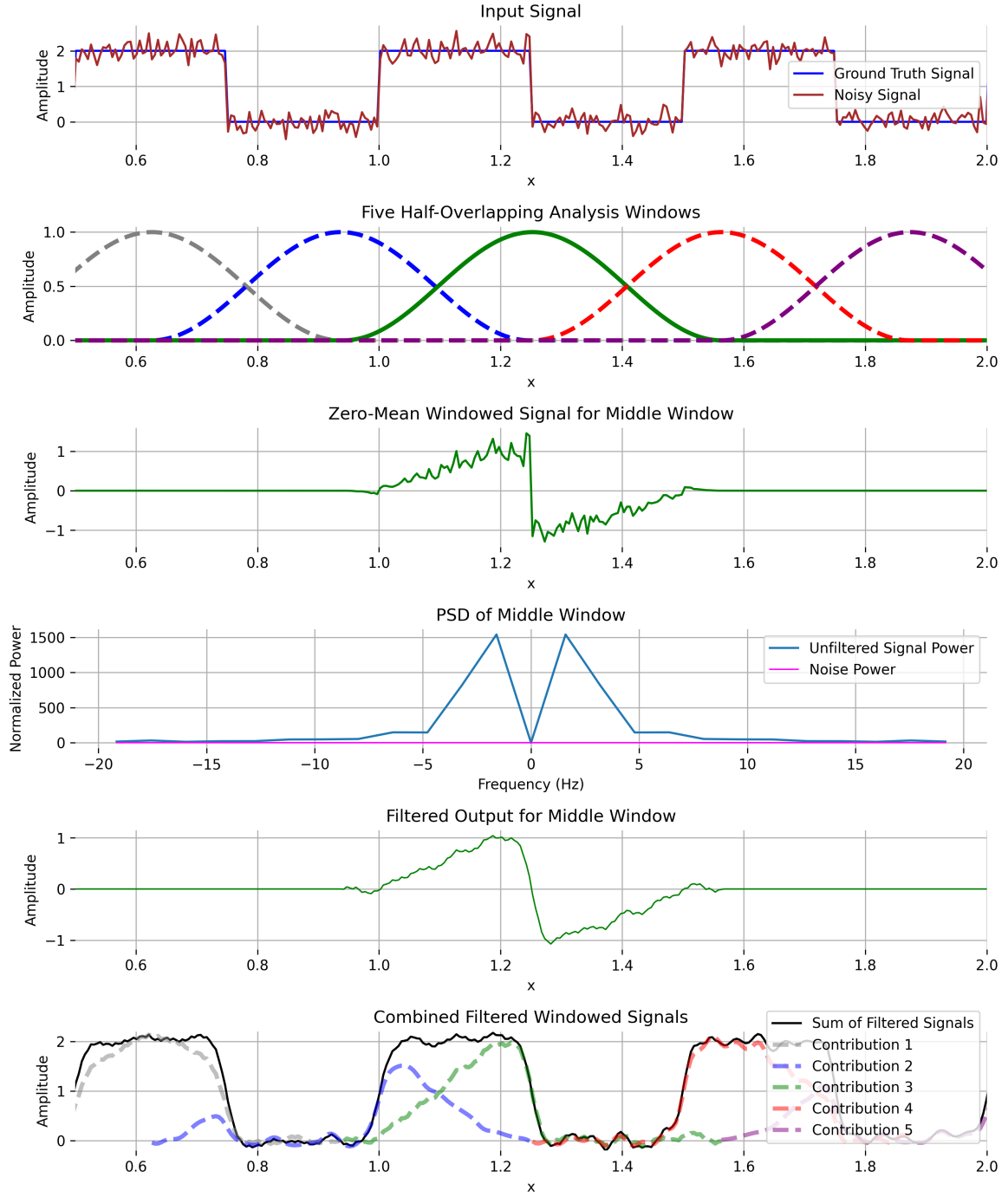


Fig. 4.1 1D Wiener filter demonstration using five overlapping windows. The first plot contains the ground truth square wave signal with a positive DC offset, along with the same signal with Additive White Gaussian Noise (AWGN) added to it. The second plot shows the raised-cosine (Hann) windowing functions, each overlapping by half their length. The third plot illustrates the product of the noisy signal and the central window. The fourth plot presents the power spectral density (PSD) of the windowed signal along with the constant noise PSD, used to compute the Wiener filter transfer function. The fifth plot shows the filtered central windowed signal ($\hat{x} = H \otimes y$) after convolution with the Wiener filter. Finally, the sixth plot displays the reconstructed signal, obtained by summing the five filtered windowed signals.

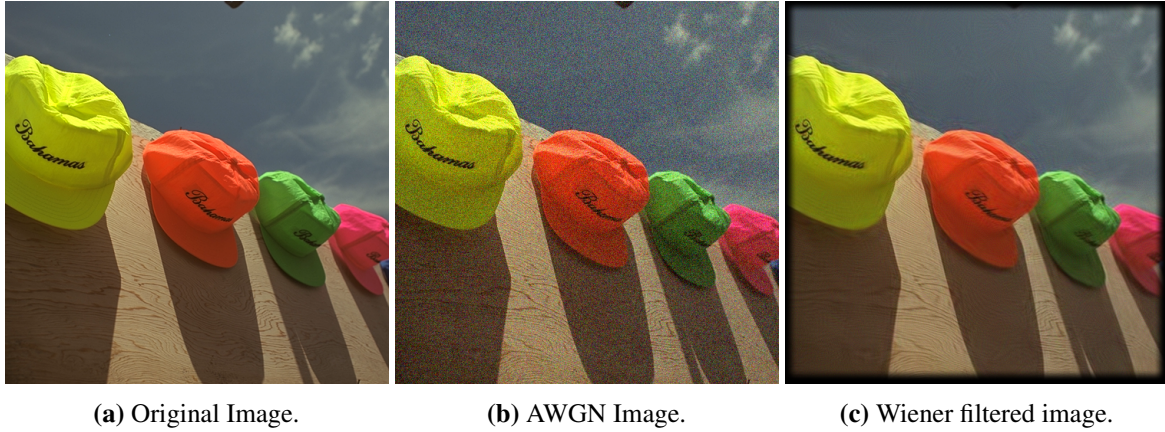


Fig. 4.2 Output example of a baseline Wiener filter implementation applied to a noisy image with a noise standard deviation of $\sigma = 20$ and a window size of 32×32 . In this example, the filtered image is neither padded during filtering nor cropped post-filtering. Ringing artefacts are visible around the edges of the hats due to the strong coring of high-frequency coefficients.

half a window's worth of pixels to each edge, using a mirrored reflection of the edge pixels for the fill values. This padding eliminates artefacts which would otherwise appear due to the trailing window edge on the outer image blocks. The image is then divided into overlapping blocks using an unfold function and organised into a six-dimensional array: $[\text{Batch}, \text{channels}, \text{Block}_x, \text{Block}_y, i, j]$, where Block_x and Block_y index the block locations along the spatial dimensions, and i, j index the spatial coordinates within each block. The batch dimension represents multiple independent input images processed in parallel; their data remain separate throughout the computation, but their losses are averaged during training to provide a more stable update to the model.

The image is processed using the same arithmetic as a traditional Wiener filter pipeline, but each operation is applied in parallel across the channel, block, and batch dimensions. We modify the 2D FFT to instead use a 3D FFT to better exploit correlations not only within spatial dimensions (height and width) but also across the colour channels. By transforming the data in three dimensions, the filter can more effectively suppress noise while preserving details that are shared across channels. This approach can capture joint frequency components and inter-channel dependencies, which are not accessible when using a 2D FFT that processes each channel independently. Thus, the 3D FFT enables a more sophisticated analysis of the frequency content across all dimensions, potentially leading to improved denoising performance.

The FFT is executed jointly across both pixel indices ($\text{Pel}_i, \text{Pel}_j$) and the channel dimension, rather than applying the Wiener filter to each channel independently. Once the Wiener filter

is found and applied to the image coefficients, the result is transformed back to the spatial domain and reassembled using overlap-add. To ensure the filter remains differentiable, we replace the max operator (see Equation 4.1) with a ReLU activation, and add a small epsilon (e.g., 1×10^{-6}) to the denominator of the Wiener transfer function, $P_{YY}[\omega_i, \omega_j] + \varepsilon$, to prevent division by zero.

4.2.2 Noise Gain Factor λ

The introduction of λ in Equation 4.1 addresses a fundamental issue arising from the variability in the noise estimation. In the frequency domain, the observed signal $Y[\omega_i, \omega_j] = X[\omega_i, \omega_j] + N[\omega_i, \omega_j]$ comprises the true signal X and noise N . The power of the noise component $|N[\omega_i, \omega_j]|^2$ is influenced by the chi-squared distribution, since it is derived from the sum of squares of independent Gaussian variables. Specifically, the real and imaginary parts of N are independent Gaussian variables with zero mean and variance $\frac{\sigma^2}{2}$, resulting in a chi-squared distribution [265] with two degrees of freedom for $|N[\omega_i, \omega_j]|^2$.

While the expected value of this distribution is σ^2 , the chi-squared distribution introduces variability in the noise power across different frequency coefficients. This variability has a variance of $2\sigma^4$, meaning that for many frequency coefficients $[\omega_i, \omega_j]$, the noise power can be higher or lower than expected. This results in an underestimation of the true noise power, allowing residual noise to pass through [266]. The factor λ compensates for this underestimation, ensuring more effective noise suppression across different frequency components, thereby improving the overall PSNR. As part of our Wiener optimisation, in our initial optimisation in Section 4.6.2 we carry out an analysis of the optimal gain value for λ .

4.2.3 Windowing and Padding

Before proceeding with the training of neural networks to enhance the performance of the Wiener filter, we optimise the parameters of the preprocessing pipeline, which are tunable - such as the padding of the image prior to windowing and the properties of the windows themselves, including their size, overlap and shape. As shown in Figure 4.2, the outermost analysis windows leave trailing edges without neighbouring windows to ensure the image scales correctly during the overlap-add process. To mitigate the impact of edge artefacts on denoising performance, our implementation (described in Section 4.2.1) pads the image by half a block's width on each side. Several padding methods exist, including constant padding, replicative padding and reflective padding. In Section 4.6.1, we evaluate the denoising performance of the Wiener filter using these different padding techniques.

Analysis Windows. We turn our attention to the properties of the analysis windows themselves and begin by assessing the impact of window size on denoising performance. Smaller windows offer better spatial localisation, allowing the Wiener filter to adapt more precisely to local image variations, reducing the appearance of ringing artefacts near edges. However, their limited frequency resolution may not capture enough frequency information to effectively suppress noise, and blocking artefacts may arise between neighbouring windows. Larger windows, on the other hand, benefit from enhanced frequency resolution for finer coring but suffer from poorer spatial localisation, increasing the risk of ringing artefacts. As well as this, larger windows are more computationally expensive, as the cost of performing an FFT for each window increases with window size according to an $O(N \log N)$ relationship, where N is the number of pixels in the window. Therefore, selecting the optimal window size involves a trade-off between frequency and spatial resolution, denoising effectiveness, and computational efficiency. In Section 4.6.4 we assess the impact of window size for two datasets of different resolution - Kodak and Div2K - in terms of output quality (PSNR/SSIM) and denoising time.

Windowing Overlap. We examine the effect of increasing the overlap between analysis windows by reducing the stride between blocks. By increasing the overlap, more pixels are processed at maximum window weighting, and multiple blocks contribute to each pixel. As a result of this more granular processing, the filter’s ability to suppress noise is enhanced. During reconstruction, the combined contributions from the overlapping blocks smooth transitions between blocks and reduce blocking artefacts.

Before we can carry out this study, we must adjust our reconstruction window scheme, because the half cosine window sums to one only with the overlap factor of two. To accommodate different values, we implement a new synthesis window that is applied to the blocks before the overlap-add. The synthesis windows are duplicated and arranged into an overlapping grid that matches the spatial dimensions of the padded image. The overlap-add procedure is then applied independently to both the filtered image blocks and the corresponding synthesis windows. The latter produces a weight accumulation map, which is used to normalise the reconstructed image by performing an element-wise division of each pixel by the sum of window contributions at that location. This normalisation ensures that the final output is correctly scaled, preserving the original image intensity where block overlaps occur.

Again, we must consider the computational cost of reducing the stride as it increases the number of analysis windows that cover the image. As the stride is reduced, the number of

windows along each dimension increases linearly with the overlap factor O :

$$N_{1D} = \frac{L \cdot O}{M}, \quad (4.3)$$

Where L is the image size along one axis and M is the width of a block. For a 2D image, the total number of windows increases quadratically with O increase: $N_{2D} = N_{1D}^2$. While the arithmetic is processed in parallel on the GPU, increasing O also increases the memory used quadratically, limiting the number of batches or the resolution of the images. As well as this, the folding and unfolding functions in Pytorch, which segment and reconstruct the image, use loops in the underlying C code, which cause the denoising time to increase with O . This cost to performance analysis for the overlap factor is also carried in our windowing optimisation results of Section 4.6.4.

Window Shape. In Section 4.6.4, we also consider the shape of the analysis and synthesis windows. As we no longer benefit from the half-overlap property of the raised-cosine window, we evaluate the performance of a Gaussian window and compare it to the Cosine window. Unlike the raised-Cosine window, the Gaussian window is isotropic in 2D space, meaning that its value depends only on its distance from the centre and not on direction. As the Fourier transform of a Gaussian function is also a Gaussian, the isotropy is preserved in the frequency domain. This may help reduce directional biases in the filtering and produce a smoother frequency response with more gradual transitions.

4.2.4 DC-Offset Removal

As mentioned in Section 4.1, the mean of each block is subtracted from the image before finding the FFT to omit the DC component from the frequency analysis and prevent a loss of low-frequency detail. In the case of heavy noise, the Wiener coring will suppress most of the signal, only leaving this DC offset. Estimating a DC value as close to the ground-truth DC value is thus critical for the overall performance. However, real noise is not necessarily zero-mean over every block. Additionally, near the extremities of the image intensity range (i.e., 0 and 255 for an 8-bit sRGB image), the noise distribution becomes skewed due to clipping or truncation, leading to a bias in the mean intensity of the block. For example, in a dark region that should ideally have zero intensity, the presence of noise can raise the mean intensity above zero. This elevated mean, representing the DC component, is then subtracted from the block before computing the FFT, resulting in an erroneously high DC coefficient being removed from the image. This can have a significant impact on denoising results as blacks become more gray following the filtering.

We propose here to study two solutions. Firstly, we replace the block average with the block median. The median will be more robust in dark areas, as outlier values will not bias the DC estimate.

The second tested approach is to implement a dedicated ancillary CNN to predict the DC value of each block. Predicting the mean of a noisy image block requires a large receptive field. Because of this, we implement a U-Net with the number of parameters minimised to reduce the memory footprint of the network. In the network’s bottleneck, only 48 feature maps are used, and only three downsampling layers are used. Each encoder layer is comprised of two 2D convolutions with batch normalisation and a ReLU activation function. The connections from encoder to decoder are changed from concatenations to element-wise summation for speed.

4.3 Automated Per-Block Noise Estimation

Up to this point, we have discussed the Wiener filter as a non-blind denoiser, where the noise power spectral density (PSD) P_{NN} , used to form the Wiener transfer function (Equation 4.1), is either known or estimated, and must be input by the user. In this section, we outline the automated noise estimation method we implement in the denoiser.

4.3.1 A Local AWGN Assumption

Estimating the variance of the noise σ^2 is a classic problem in image denoising. Typically, two main approaches are taken:

1. *Global filtering-based methods*, where the noisy image is filtered using a low-pass filter, and the variance of the residual image is computed [267]. Extensions of this method include techniques like separable difference filters [268] and Immerkaer’s Laplacian operator [269], which aim to remove image structures and isolate noise.
2. *Block-based methods*, where a percentage of image blocks with the lowest variance are identified and averaged to estimate the global image variance [270]. This approach can be improved by pre-filtering the image [271] to further reduce the influence of noise when choosing candidate blocks.

More sophisticated industry tools, as proposed by Kokaram et al. [272] combine the filter and block-based approach. In their method, candidate patches for noise selection are selected by applying a light Gaussian filter to each patch and calculating the gradient covariance matrices. These matrices are then assessed for corner-ness, total gradient energy and eigenvalue ratios,

to ensure the blocks contain noise and lack structure. Once these conditions are met, the noise PSD is calculated for the remaining blocks and the median value for each frequency bin is used as the noise PSD estimate.

While this approach provides an estimate of the noise PSD, it assumes that the noise is spatially symmetric and not signal-dependent. Most of these methods operate under the assumption that the noise is AWGN and stationary across the frame. However, as discussed in Section 2.1, real noise generated by CMOS sensors is signal-dependent and spatially correlated. As a result, choosing a single standard-deviation value (σ) for the Wiener filter is sub-optimal, as the noise characteristics vary across the image.

To address this, we propose a solution that assumes each image block is small enough to be treated as locally Gaussian with a single standard deviation. Instead of choosing a global value for the noise P_{NN} , we propose to estimate noise on a per-block basis, enabling a more accurate adaptation of the Wiener filter.

Manually specifying noise values would be impractical - e.g. a 512×512 image would create 3,969 blocks using a 16×16 block size with half-overlap. Thus, we propose to create a standard deviation prediction network to automate the estimation of the per-block P_{NN} estimate.

4.3.2 Estimating the Noise Variance

To better denoise real noise images and to match the practicality of modern DNN denoisers, we propose to implement a blind version of the Wiener filter, which does not rely on human input. We train a lightweight ancillary neural network to predict the noise standard deviation for the user, and experiment with different network architectures, as well as different network output formats. These output formats consist of predicting the noise per image, per channel, and per individual block. We devise a straightforward backbone for the ancillary network consisting of convolution, batch-norm, ReLU layers, modifying the number of layers and the number of features per layer between experiments.

To train the network using a signal-dependent model, the CBDNet model [40] is used to synthesise a training dataset for the ancillary network, as in Chapter 3. Although the authors of CBDNet introduce their own noise mapping system, the ground truth noise measurements do not correspond directly to the noise standard deviation. For this reason, to accurately estimate the noise standard deviation (STD), we generate multiple noisy copies of each clean image, all sharing the same additive Gaussian noise component but differing in their

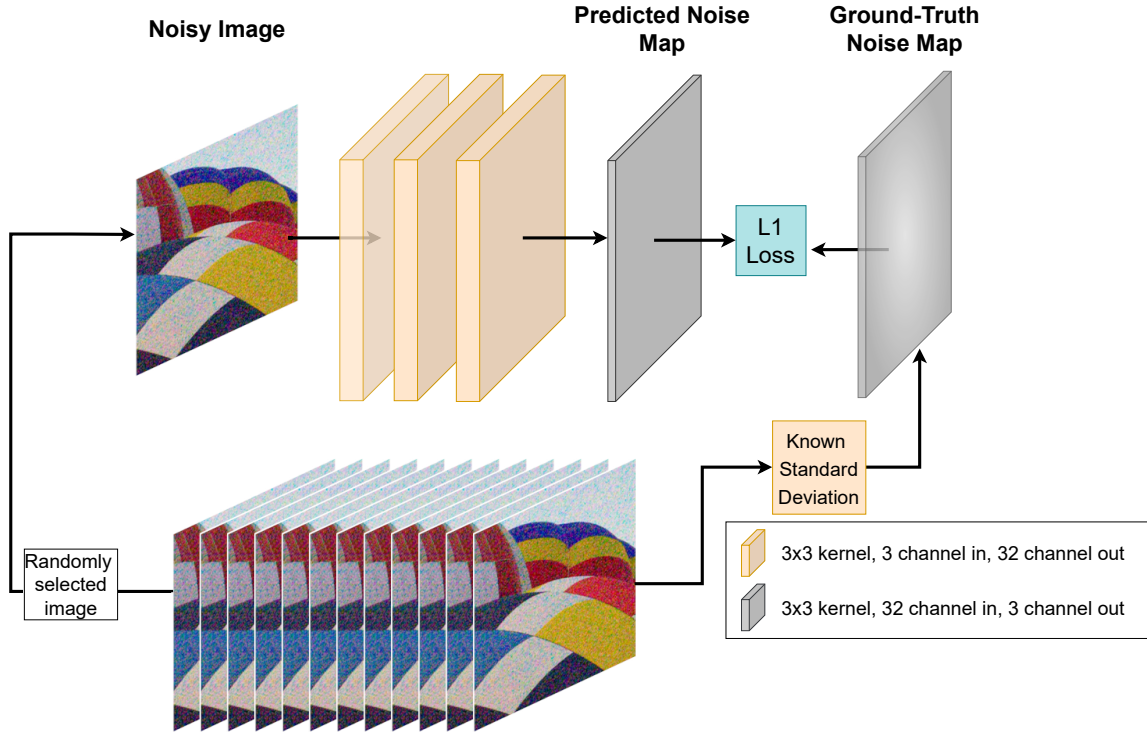


Fig. 4.3 Illustration of the training pipeline for Stage 1 of the standard deviation prediction network. In Stage 2, the network is integrated and trained jointly with the Wiener filter.

signal-dependent (multiplicative) noise components. This setup allows us to create diverse training samples while maintaining a consistent underlying noise level across the set.

During training, all noisy copies of an image are loaded, and the residual noise is computed for each sample using the clean ground truth image: $n_i = y_i - x$. The variance at each pixel is estimated by averaging the squared residuals over all samples, and the square root of this variance yields a ground-truth STD map for supervision.

The noise prediction network is trained in two stages. In **Stage 1** (see Figure 4.3), the network is trained independently of the Wiener filter. A randomly selected noisy image is passed through the network to produce a predicted STD map, which is compared to the ground-truth map using an ℓ_1 loss. In **Stage 2**, the network is fine-tuned jointly with the Wiener filter: the predicted STD values are passed to the filter, and the network is optimised to minimise the ℓ_2 loss between the denoised output and the clean image. This two-stage procedure enables the network to benefit from both synthetic supervision and end-to-end tuning using real data. . In Section 4.6.5 we assess the performance of the different noise standard deviation prediction networks.

4.4 The Coring Refinement Network

Finally, we turn our attention to the heart of the denoiser itself: the Wiener coring function. Although we have discussed the tuning and optimisation of several parameters, the predicted transfer function itself $H[\omega_1, \omega_2]$ is responsible for noise reduction while preserving image fidelity. As the noise increases, maintaining this balance becomes increasingly difficult, leading to the loss of high-frequency information, such as textures. Despite our previous optimisations, ringing artefacts also persist. To address these issues and restore details destroyed by noise, we develop another ancillary network to refine the coefficients of the Wiener transform function $H[\omega_1, \omega_2]$, before it is applied to the image frequency coefficients.

The proposed architecture for the coring refinement network is illustrated in Figure 4.4. The input to the network is the initial Wiener transfer function H , and the output is a refined, data-driven transfer function H_{opt} . At this point in the pipeline, the frequency-domain representation of the filter has shape $[B, C, M_x, M_y, N_i, N_j]$, where B is the batch size, C is the number of channels, M_x and M_y are the number of spatial blocks along the horizontal and vertical axes, and $N_i \times N_j$ are the spatial dimensions of each block.

The filter is refined in two stages. First, to process the frequency coefficients within each block independently, the block indices are folded into the batch dimension, resulting in a tensor of shape $[G, C, N_i, N_j]$, where $G = B \times M_x \times M_y$. This allows a 2D convolutional neural network to operate over each block's frequency coefficients as a standalone input, learning to refine the filter locally within each block. In the second stage, the within-block frequency indices are folded into the batch dimension instead, producing a tensor of shape $[Q, C, M_x, M_y]$, where $Q = B \times N_i \times N_j$. A second 2D convolutional network is applied across this spatial block grid, enabling the network to model inter-block correlations at each frequency coordinate. Together, these two stages provide both local (intra-block) and global (inter-block) refinement of the frequency-domain transfer function. Skip connections are used between layers to improve training stability.

The network architecture aims to minimise the number of parameters used. A simple 2D convolution-leaky ReLU pairing is used per layer, with 20 channels per layer, with 5 layers in stage 1, and 4 layers in stage 2. The final number of trainable parameters in the network is 22,506. As with the STD estimation network, the network is trained through the differentiable Wiener filter, with the estimated denoised image and ground truth image used to calculate the L1 loss.

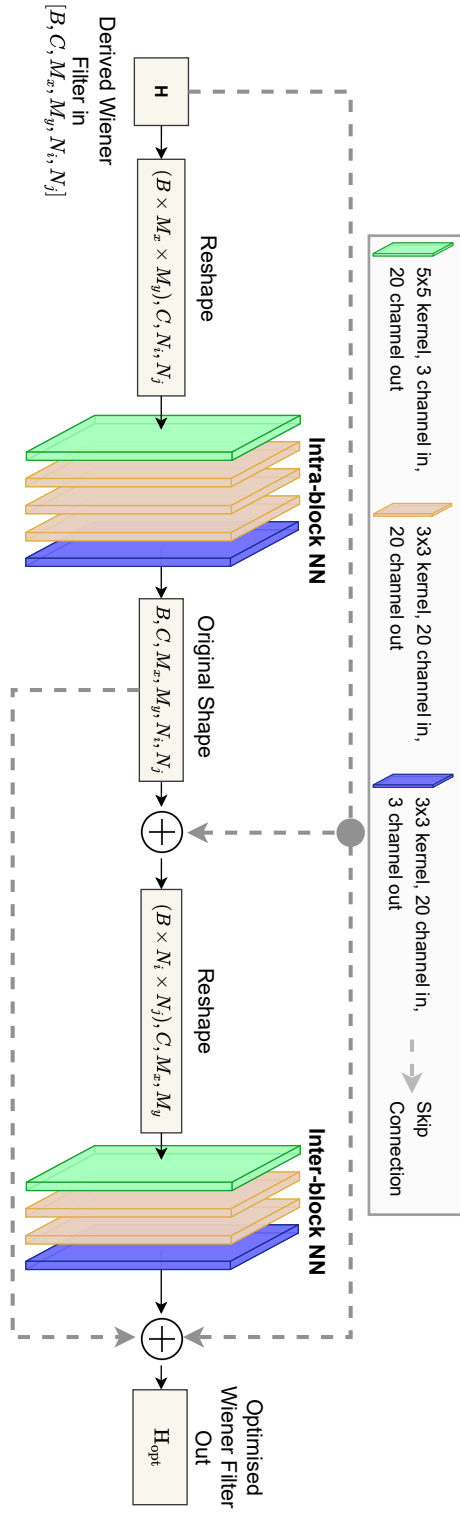


Fig. 4.4 Illustration of the coring refinement ancillary network integrated within the Wiener filter backbone architecture to optimise the initial prediction of the coring transfer function $H(\omega_1, \omega_2)$.

4.5 Multi-Scale Wiener Filtering

The final consideration we take in improving the Wiener filter is to implement a collaborative filtering technique whereby the image is filtered at multiple scales (window sizes) and the resulting outputs are consolidated into a single image. The nature of the Fourier transform is such that, depending on the content of an image, a larger or smaller block size may be more desirable. Low-frequency, flat areas benefit from larger windows, where spatial localisation is less critical, and greater frequency resolution can be achieved. Conversely, high-frequency, textured areas yield better denoising results with smaller windows, where edges need to be carefully preserved. By integrating the outputs from different scales, we aim to reduce noise effectively while maintaining image fidelity across all regions. It may also be beneficial to capture image details at multiple scales for a more complete frequency analysis.

In Algorithm 2, we summarise the proposed optimisations made in our final implementation. All image blocks are filtered in parallel; however, we use for-loops for illustrative purposes. In line 1, the automatic noise estimation ancillary network is denoted CNN_{STD} . The new isotropic exponential window scheme is shown, followed by the use of multiple block sizes for the multi-scale filtering case. The block size in this implementation is changed to a quarter of a block's width, and the mean DC offset removal is changed to median subtraction. The Wiener coring refinement network is used in line 12, and the image is reconstructed using the new synthesis window scheme.

4.6 Ablation Studies

In this section, we outline the impact the modifications we have discussed have on the Wiener filter. By considering these results and their computational cost, we optimise the Wiener filter step by step. The SIDD benchmark is used to assess denoising performance using PSNR and SSIM as measures of quality. We also use the Kodak and DIV2K datasets with AWGN in our initial ablation studies where the use of real noise is not critical in our optimisations. Each experiment is evaluated on the Pytorch parallelised Wiener filter.

4.6.1 Optimal Padding Method

We begin by evaluating the optimal padding strategy to apply before filtering. We utilise the baseline Wiener filter implementation, as described in Section 4.1, with the 3D FFT added. In our approach, a half block's width is added to each image border. We experiment with constant padding (using fill values of 0, 0.5 and 1), replication padding, which repeats the

Algorithm 2 Proposed Wiener Filter Implementation**Require:** Noisy image, y

- 1: $\hat{\sigma} \leftarrow \text{CNN}_{\sigma}(y)$ ▷ noise STD prediction
- 2: $w(h, k) \leftarrow \exp(-\alpha(h^2 + k^2))$ ▷ windowing definition
- 3: **for all** block sizes $\in [8, 16, 32, 64, 96]$ **do**
- 4: **for all** blocks y in image y , for stride=BlockSize/4 **do**
- 5: $\bar{y} \leftarrow \text{median}(y)$ ▷ predicts block mean
- 6: $y_w \leftarrow (y - \bar{y}) \odot w$ ▷ windowing
- 7: $Y \leftarrow \text{FFT}n(y_w)$
- 8: $P_{yy} \leftarrow Y \odot Y^*$
- 9: $P_{nn} \leftarrow \hat{\sigma}^2 \|w\|^2$
- 10: $P_{xx} \leftarrow \max(P_{yy} - P_{nn}, 0)$ ▷ coring
- 11: $H \leftarrow P_{xx} \oslash P_{yy}$ ▷ Wiener transfer function
- 12: $H_{opt} \leftarrow \text{CNN}_{\text{WienerNet}}(H)$ ▷ Wiener Refinement Network
- 13: $\hat{x}_w \leftarrow \text{iFFT}n(Y \odot H_{opt}) + \bar{y}$
- 14: $x_{all} \leftarrow \text{overlap_add}(w \odot \hat{x}_w)$ ▷ combine all image blocks
- 15: $w_{all} \leftarrow \text{overlap_add}(w \odot w)$ ▷ combine all synthesis windows
- 16: $\hat{x} \leftarrow \frac{x_{all}}{w_{all}}$

Method	None	Constant 0	Constant 1	Constant 0.5	Replicate	Reflect	Circular
PSNR (dB)	18.61	30.28	29.38	30.13	30.52	30.55	30.48

Table 4.1 Ablation study results for different edge padding modes on Wiener denoising performance. The block size is fixed at 32×32, with a half-overlap applied between blocks.

outside pixel value, reflect padding, which copies the image value equidistant along the axis of reflection, and circular padding, which wraps the image values around to the opposite edge. The ablation study is conducted using AWGN, as the use of a real noise model is not critical for this exercise. The mean PSNR for each method is reported in Table 4.1. In this case, we observe that reflection padding offers +0.27 dB greater PSNR than padding with constant zeros, and marginally greater performance than replication and circular padding. Reflective padding performs best because it preserves local image continuity at the borders without introducing artificial edges or discontinuities, which can otherwise introduce high-frequency artefacts during Fourier-based filtering. Going forward, we proceed with reflective padding due to the improved PSNR performance on our benchmark.

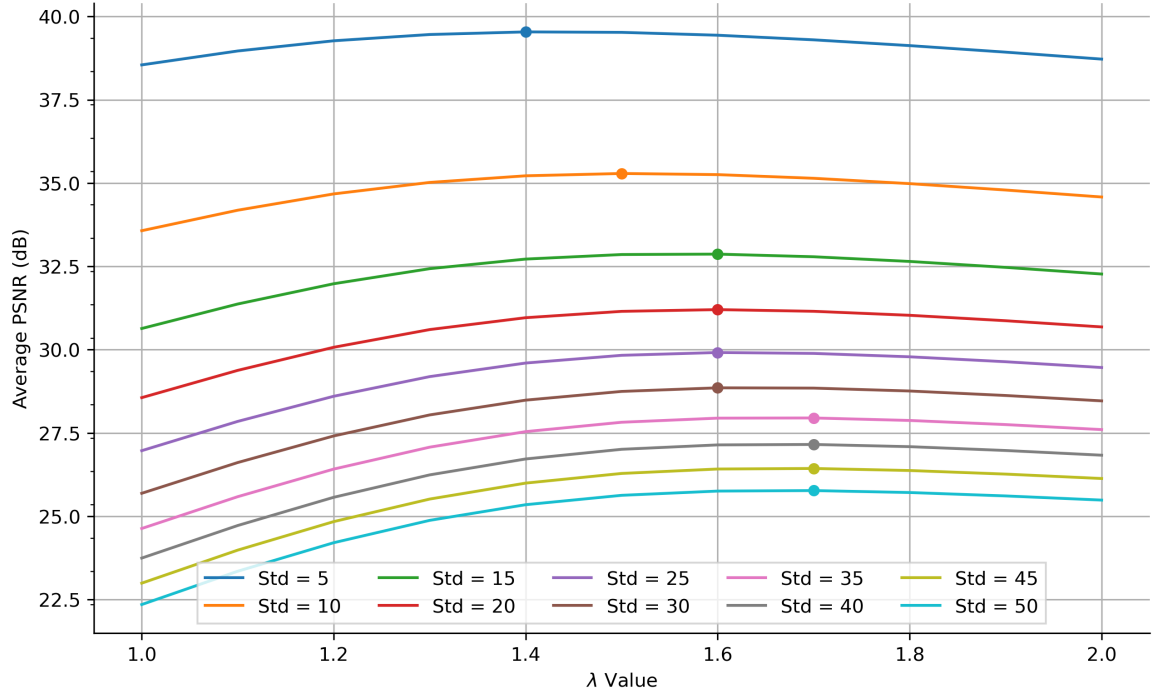


Fig. 4.5 Ablation study results for the impact of the λ noise STD gain parameter on the denoising performance of the Wiener filter. The noise PSD is calculated as $P_{NN} = \|\sigma\|^2 \cdot \lambda$. Each curve represents the average PSNR performance on the Kodak dataset as λ varies, with each curve corresponding to a different noise standard deviation (std) used in the experiments. The marker on each plot signifies the maximum measured PSNR achieved for the noise std.

4.6.2 Optimisation of Noise Gain λ

To determine the optimal value of the noise gain factor λ , we evaluate the baseline Wiener filter output PSNR for a noise gain in the range $\lambda \in [1.0, 2.0]$. As before, the study is evaluated using the Kodak dataset and AWGN. The experiment is repeated ten times, at noise standard deviation of 5 to 50 in increments of 5. That is $\sigma = \{5, 10, 15, \dots, 50\}$. This allows us to record the ideal noise gain λ as the noise increases. In Figure 4.5 we plot the λ to PSNR graph for each noise standard deviation experiment, with the greatest PSNR for each experiment marked. These results show that the optimal value for λ ranges between 1.4 and 1.7, increasing with the amount of noise added to the image noise standard deviation, as expected given the chi-squared distribution of the noise power. Based on these results, we fix the gain in our implementation $\lambda = 1.6$, which provides an increase in performance across all noise variances.

	5	10	15	20	25	30	35	40	45	50
2D FFT (dB)	37.56	33.38	31.09	29.54	28.35	27.40	26.59	25.88	25.23	24.65
3D FFT (dB)	39.41	35.17	32.76	31.11	29.83	28.79	27.89	27.11	26.40	25.74

Table 4.2 PSNR (dB) Ablation study results comparing the performance of baseline 2D FFT and 3D FFT Wiener filtering on the Kodak dataset. The improved 3D results verify the enhancement over the 2D FFT approach.

4.6.3 2D vs. 3D Wiener Filtering

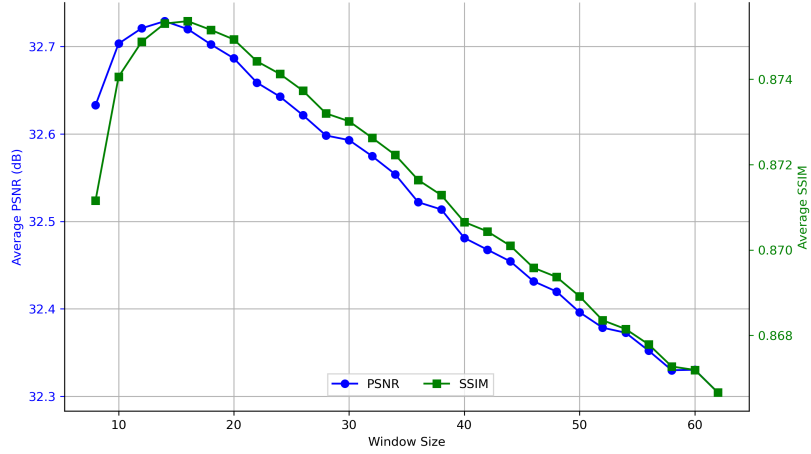
In this ablation study, we validate the improvements introduced by the 3D Wiener filter over the 2D case. As shown in Table 4.2, the 3D Wiener filter, which incorporates channel information in the FFT computation, achieves superior denoising performance compared to the 2D FFT approach that operates only over the spatial dimensions. For the 2D case, the λ parameter is optimised for each noise level to maximise performance, while all other parameters are consistent across both methods.

The results demonstrate that the 3D FFT Wiener filter achieves an average improvement of 1.45 dB in PSNR over the 2D FFT approach. This substantial gain highlights the effectiveness of the 3D approach in leveraging additional information to enhance noise suppression, making it a more robust solution for denoising tasks.

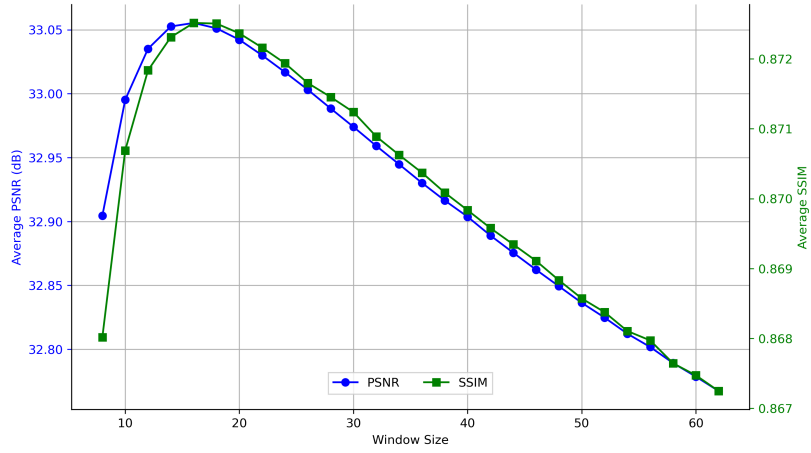
4.6.4 Windowing Ablation Studies Results

Window Size. We evaluate the impact of window size on denoising performance in terms of PSNR and SSIM, using the Kodak dataset (768×512) and the DIV2K validation set (2K DCI), both corrupted with additive white Gaussian noise of $\sigma = 15$. Figure 4.6 shows PSNR and SSIM curves as functions of window size for each dataset. In both cases, the average PSNR and SSIM are highest for smaller window sizes, underscoring the importance of preserving spatial localisation in the denoising process. On the Kodak dataset, the maximum PSNR of 32.73 dB is achieved at a window size of 14×14, while the highest SSIM of 0.8754 occurs at 16×16. Similarly, on DIV2K, both the optimal PSNR of 33.06 dB and SSIM of 0.8725 are observed at a window size of 16×16. These results suggest that similar block sizes are optimal across datasets of differing resolution.

Figure 4.7 illustrates the relationship between window size and total denoising time, measured over all 100 images in the DIV2K validation set. We observe that denoising time increases as the window size decreases. This is due to the increased overhead associated with folding and unfolding a larger number of blocks, which becomes relatively more expensive than the



(a) Kodak Dataset Result.



(b) DIV2K Dataset Result.

Fig. 4.6 Ablation study results evaluating the impact of block size on PSNR and SSIM for the Kodak and DIV2K datasets, affected by AWGN with $\sigma = 15$.

FFT operations when using small windows. Notably, denoising time exhibits an oscillating pattern: it is minimised at window sizes divisible by 4 and increases sharply at sizes that are not. This behaviour is likely due to optimisations in PyTorch’s FFT backend (cuFFT), which uses radix-4 algorithms that operate more efficiently when the window dimensions are multiples of 4.

Based on these observations, we select a window size of 32×32 for subsequent experiments, as it offers efficient runtime while preserving high denoising performance.

Window Overlap. Next, we evaluate the effect of reducing the stride between windows, or equivalently, increasing the overlap between them, on denoising performance. For this study, we use the real noise SIDD dataset. Since the SIDD online benchmark was unavailable

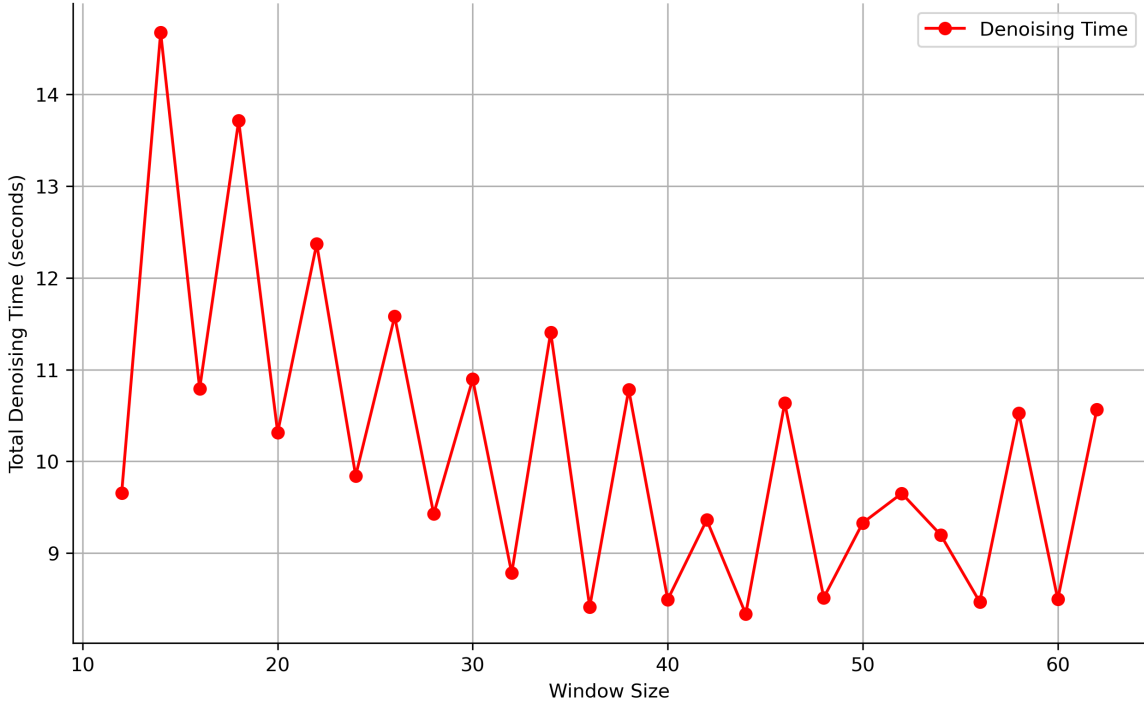


Fig. 4.7 Ablation study results showing the impact of window size ($N \times N$ pixels) on denoising time (seconds) for the Wiener filter with a half-block-stride between windows. Denoising time is measured as the total processing time, excluding data-loading, for denoising 100 2K validation images from the DIV2K test set.

due to maintenance during this period, a subset of the SIDD training set was utilised for our evaluation. Specifically, 50 images were selected from the 4K dataset and cropped to 128×128 pixels. These images are excluded from the training dataset of subsequent experiments. As before, we employ a raised cosine window with a block size of 32×32 pixels. The block stride is reduced from the original half block overlap to a one-seventh block stride. In cases where the stride, expressed as a fraction of the block width, does not result in an integer number of pixels, the stride distance is rounded down to the nearest integer. To accommodate changes in overlap, and as discussed in Section 4.2.3, a new synthesis window scheme is used to normalise overlapping window magnitudes, since the original synthesis window only sums to one under half block overlap.

The results for this experiment are shown in Table 4.3, reporting the mean PSNR and total denoising time for the dataset. As the stride decreases relative to block width, denoising performance improves. Compared to the original half block overlap filter, we observe a maximum improvement of **+0.42 dB** in PSNR at a one-seventh block stride. However,

Block Stride	1/2	1/3	1/4	1/5	1/6	1/7
PSNR (dB)	35.06	35.33	35.42	35.46	35.47	35.48
Denoising time (s)	0.94	1.99	2.85	4.89	7.41	13.26

Table 4.3 Ablation study results evaluating the impact of window stride (measured as a fraction of the block size) on denoising performance (PSNR) and denoising time. The experiment uses a fixed block size of 32×32 with a Hamming window. A 1/4 block overlap achieves the best balance between performance improvement (+0.36 dB) and computational complexity.

Analysis Window	Synthesis Window	PSNR (dB)
Hamming	Hamming	35.42
Gaussian ($\sigma = 0.5$)	Gaussian ($\sigma = 0.5$)	35.49
Gaussian ($\sigma = 0.2$)	Gaussian ($\sigma = 0.7$)	35.52

Table 4.4 Ablation study results comparing the effect of analysis and synthesis window shapes on denoising performance (PSNR) in the SIDD validation benchmark. The baseline Hanning (raised cosine) window is compared with isotropic Gaussian windows. In the final experiment, the standard deviations of the analysis and synthesis windows are individually optimised for PSNR. The window stride is set to one-quarter of the window width.

we choose to use a quarter block overlap in our implementation, as finer overlaps yield diminishing returns that do not justify the additional computational cost.

Window Shape. Next, we assess whether changing the windowing functions can enhance denoising performance. We evaluate three different scenarios: the original raised cosine window, the use of a Gaussian window, and finally, the application of two unique Gaussian windows with optimal distributions - one for the analysis window and another for the normalising synthesis window (double Gaussian window). The Gaussian window distribution is given by a standard deviation parameter. Here we choose standard deviation which maximises the PSNR of the SIDD test set. The overlap is set to one-quarter of a window and the block size to 32×32 . Our results in Table 4.4 indicate that the double Gaussian window yields the best result on the test set, improving the output PSNR to 35.52 dB - a **+0.1 dB** improvement over the raised-cosine window, with STD values of 0.3 and 0.8 for the analysis and synthesis windows respectively.

As a final comparison, Figure 4.8 illustrates the impact of denoising a real noisy image crop with the raised cosine versus the double Gaussian window scheme. While blocking artefacts remain visible with the raised cosine window, the double Gaussian windows effectively eliminate these artefacts, resulting in a smoother image.

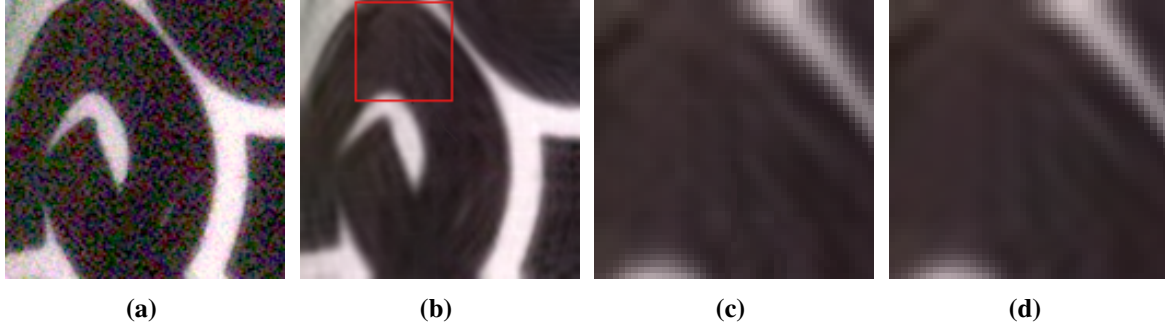


Fig. 4.8 Comparison of denoising results using different windowing schemes to reduce blocking artefacts. (a) Original noisy image ($PSNR$: 25.80 dB, $SSIM$: 3.89, MS - $SSIM$: 8.08). (b) Image denoised using a single window ($PSNR$: 32.79 dB, $SSIM$: 14.34, MS - $SSIM$: 16.46). (c) Crop of (b) highlighting blocking artefacts caused by window distortion. (d) Blocking artefacts resolved using an additional interpolation window ($PSNR$: 33.18 dB, $SSIM$: 15.04, MS - $SSIM$: 17.24).

4.6.5 Per-Block Noise Estimation

In this experiment, as described in Section 4.3, we train and evaluate the noise estimation ancillary network. Each network is trained in two stages: a coarse training stage targeting standard deviation estimation, and a fine-tuning stage targeting overall image quality, where the ancillary network is attached to the Wiener filter. We create nine configurations for the prediction network, using 2, 4, or 6 layers, with 16, 32, or 64 channels per layer. For the first stage of training, ground-truth STD maps are generated from 800 images of the DIV2K dataset [273], using the noise model of CBDNET [40]. An additional 320 real-noise images from the SIDD training set [37] are included for the fine-tuning stage. Each CNN is trained for 4,000 epochs with a cosine-annealing learning rate that decays from 1×10^{-3} to 1×10^{-5} every 300 epochs. We use a mini-batch size of 24 and randomly select 128×128 crops from each image every epoch. To create a better comparison, we also evaluate the performance of the noise prediction network when it provides a single prediction for the entire image (per image), and a prediction per channel.

In Table 4.5, we present the result for each network architecture and prediction scope. Additionally, we include the denoising result for the hand-picked standard deviation value of $\sigma = 10$, which provided the best outcome across the dataset. We demonstrate that the denoising capabilities of the Wiener filter improve as our ancillary network makes predictions at a finer scope. We observe a **+1.2 dB** (36.72 dB) improvement over hand-optimising the denoiser for the dataset (Global), with the smallest network (2 layer, 16-channel). Our largest network achieves a maximum denoising performance of 37.20 dB (**+1.68 dB**). We choose to use our 4-layer network in future experiments due to its more favourable performance-to-parameter-count ratio of only 20.4 thousand parameters.

Noise STD Prediction	Scope	PSNR(dB)	#Params
Fixed to $\sigma = 10$	Global	35.61	-
CNN, 4 layers, 16 channels	Per Image	36.72	5.6k
CNN, 4 layers, 16 channels	Per Channel	36.91	5.6k
CNN, 2 layers, 16 channels	Per Block	36.72	0.9k
CNN, 2 layers, 32 channels	Per Block	36.69	1.8k
CNN, 2 layers, 64 channels	Per Block	36.70	3.6k
CNN, 4 layers, 16 channels	Per Block	37.07	5.6k
CNN, 4 layers, 32 channels	<u>Per Block</u>	<u>37.11</u>	<u>20.4k</u>
CNN, 4 layers, 64 channels	Per Block	37.11	77.0k
CNN, 6 layers, 16 channels	Per Block	37.12	10.0k
CNN, 6 layers, 32 channels	Per Block	37.12	39.0k
CNN, 6 layers, 64 channels	Per Block	37.20	152.0k

Table 4.5 Ablation study results evaluating the effect of different noise standard deviation (STD) prediction networks on Wiener denoising performance. Each row compares models with varying numbers of layers, feature maps, and prediction scopes, reporting their PSNR and the number of parameters. Quarter-overlap Gaussian windows on 32×32 blocks are used. The underlined method represents the best trade-off between performance and model complexity.

4.6.6 DC Offset Removal

In Table 4.6, we report the results for the different DC offset estimation methods proposed in section 4.2.4. Results are obtained from our current best Wiener configuration (ie, Gaussian window, $\frac{1}{4}$ block overlap, 32×32 and 4×16 standard deviation prediction network). The ground-truth mean is also included in our results to demonstrate the upper bound of possible denoising improvement for optimal DC offset estimation. Three custom U-Net architectures are implemented and trained to predict the mean value of each block: UNet-L, U-Net-M and U-Net-S with 2M, 0.3M and 0.12M parameters, respectively. Layer concatenations are replaced with summations to make relationships between skip-connections easier to learn. The network is trained using the L1 loss between the network output and the mean of the network output. The same training scheme is followed as the fine-tuning step of the noise prediction network.

We obtain the results 37.24 dB, 37.48 dB and 37.41 dB for the small, medium and large networks, respectively. While the results are improved over the mean removal method, these networks do not outperform the simple median filter, which yields a denoising PSNR of 37.55 dB (+ **0.48 dB** over the original mean method). The subtraction of the ground-truth mean gives a result of 38.55 dB, outlining the performance possible with further optimisation in this space. As the median estimation method contributes no additional parameters to the

Method	Mean	Median	U-Net-S	U-Net-M	U-Net-L	GT
PSNR	37.07	<u>37.55</u>	37.24	37.43	37.42	38.55

Table 4.6 Ablation study results comparing different methods for block mean estimation as a preprocessing step in the Wiener pipeline. The table reports PSNR values for various estimation methods, including simple mean, median, and U-Net models of different sizes (S, M, L), with ground truth (GT) provided for reference.

Window Sizes					PSNR (dB)
8	16	32	64	96	
✓	✓				36.72
✓	✓	✓			37.49
✓	✓	✓	✓		37.81
✓	✓	✓	✓	✓	37.89
✓		✓			37.05
✓			✓		37.50
	✓	✓	✓		37.80
	✓	✓	✓	✓	37.74

Table 4.7 Ablation study results evaluating the impact of multi-scale filtering on denoising performance for the SIDD dataset. Each row shows PSNR results for different combinations of Hann window sizes, with checkmarks indicating the window sizes used in each experiment.

Wiener denoiser and is faster and less computationally expensive to run, we adopt it as the DC offset removal method over the CNNs.

4.6.7 Multi-Scale Wiener Filtering

With the preprocessing pipeline fully optimised and the automatic noise estimation module implemented, we evaluate the impact of multi-scale Wiener filtering. Again here, our offline SIDD benchmark is used, with a quarter-block stride analysis window and per-block automatic noise selection. The noisy image is passed through the denoiser using multiple different analysis window sizes, and the individual results are averaged at the output to give a final estimate of $\hat{x}$. We present these results in Table 4.7. Interestingly, over our test set, not all combinations of windows yield an improvement over the previous best effort of 37.55 dB. The best result is obtained for a combination of windows of sizes [8,16,32,64,96] at 37.89 dB, marking a **+0.34 dB** improvement over a single window size of 32×32.

4.6.8 Coring Refinement Network

Our final contribution in enhancing the Wiener filter is the implementation of the Wiener-CNN coring refinement network, as illustrated in Figure 4.4. The network is trained using the same real-synthetic dataset mix used to train the STD prediction network (refinement stage). As well as this, the same optimiser and learning-rate settings are used. The 4×32, per-block noise estimation network is used with its weights frozen during the training of the coring refinement network.

Our final trained coring network results in a benchmark score of 38.13 dB, making a significant **+0.58 dB** gain over the non-optimised (single resolution) coring result, using only an additional 23k trainable parameters. In the next section, we will discuss the final results in more detail.

4.7 Comparison with the State-of-the-Art

To assess the performance of the Wiener filter at different steps of the optimisation process, we create a final benchmark, where the improvements are added in stages and each stage is assessed. We label each stage as Wiener 0 to Wiener 4 (W0-W4):

Stage W0 refers to the baseline Wiener filter, where the value for σ is set by choosing the best value for the dataset. The overlap between windows here is set to half a block.

Stage W1 refers to the same filter as **W0**, with the standard deviation estimation module added. Here, the module is limited to choosing a global noise STD per image channel.

Stage W2 follows from **W1**, with the noise-estimation working per-block. The windowing is also enhanced with the double Gaussian (analysis and synthesis) windows. Lastly, the DC-offset removal is switched from mean subtraction to median subtraction.

Stage W3 next, the finer overlaps are implemented such that the windowing stride is a quarter of the block width. As well as this, the multi-scale filtering is implemented, at window sizes of [16, 32, 64].

Stage W4 Lastly, the coring refinement module is added.

In Table 4.8, we summarise the performance of these incrementally improved versions of the Wiener filter, alongside several popular denoising networks: DnCNN [274], CBDNet [40], and the SUNet denoising transformer [34]. We use the bias-free variant of DnCNN [198], re-trained on the SIDD training dataset, as the original model was designed for Gaussian noise. For CBDNet, we assess the off-the-shelf model, which was trained using the same

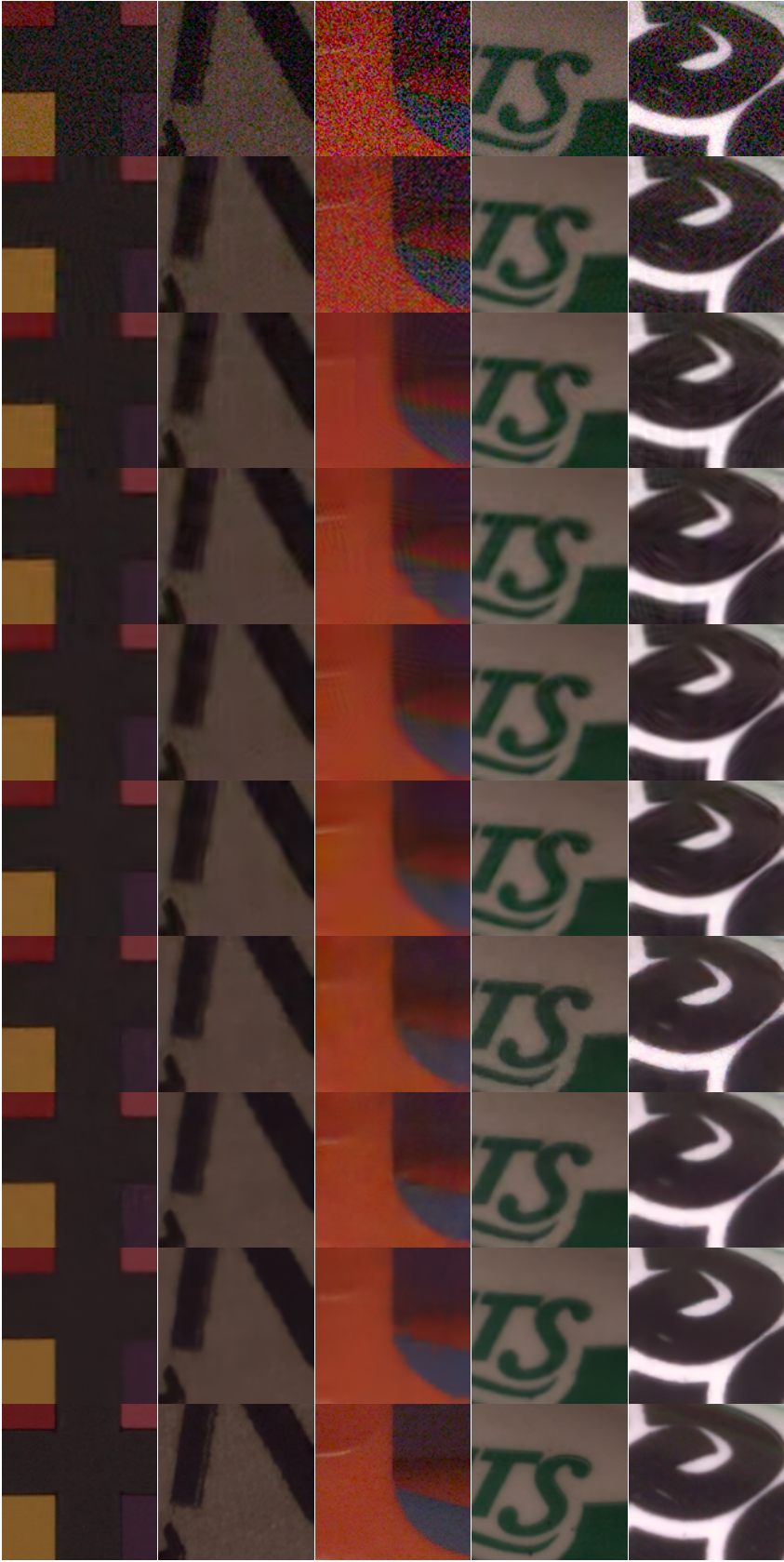


Fig. 4.9 Cropped results for methods presented in Table 4.8. W3 and W4 correspond to our optimised Wiener without and with NN coring.

Method	PSNR (dB)	#Params
W0: Original Wiener	35.06	n/a
W1: (W0) + <i>per-channel</i> σ	36.58	n/a
W2: (W0) + <i>median</i> + <i>Gauss.</i> + <i>per-block</i> σ	36.99	20k
W3: (W2) + $\frac{1}{4}$ <i>stride</i> & <i>multi-scale overlaps</i>	37.87	20k
DnCNN*	37.94	640k
W4: (W3) + <i>Coring NN</i>	38.17	43k
CBDNet	39.32	4.36 M
Swin/SUNet	39.60	99.00 M

Table 4.8 Final results comparing different optimisation levels of the Wiener filter on our real-noise benchmark. All models were trained using the same synthetic noise model, ensuring a fair comparison. Each row reports PSNR (dB) and parameter count. W3 and W4 achieve competitive performance relative to the real-noise-trained, bias-free DnCNN (DnCNN*), while using significantly fewer parameters. DnCNN* was re-trained on the SIDD dataset using the bias-free variant from [198]. CBDNet and SUNet are evaluated using off-the-shelf models trained with the same noise model. SUNet used the DIV2K dataset, while CBDNet used the same dataset as our models.

synthetic real-noise model and dataset as our denoisers. For SUNet, we also evaluate the off-the-shelf model, trained using the same synthetic noise model applied to the DIV2K dataset [196]. By ensuring that all methods are trained using the same synthetic noise model, we establish a fair basis for comparison. As before, the benchmark is run on our SIDD real-noise test set.

From these results, we observe that Wiener stage 3 (without coring refinement NN) performs comparably (within 0.07 dB) to the improved bias-free DnCNN denoiser. W4 outperforms the DnCNN implementation with 43,000 trainable parameters, almost 15 times fewer trainable parameters than DnCNN. The denoiser comes within 1.1 dB of CBDNet, a UNet with 4.36 million parameters. We also include denoising results for the SWIN transformer, a SOTA vision transformer of 99M parameters.

4.7.1 Visual Assessment

In this Section, we discuss the visual impact of our improvements by comparing denoised samples at different levels of optimisation. In these comparisons, popular (DnCNN) and state-of-the-art denoising samples (SUNet) are provided for reference. Both real noise images from the SIDD benchmark and Gaussian noise images from the NTIRE 2023 Challenge on Image Denoising are assessed. The latter images are included to provide higher resolution, non-cropped, well-lit examples of interesting subject matter to study. In addition to this, the low-light images of the SIDD benchmark are noisy themselves.

Real Noise Comparison

Figure 4.9 presents sample crops from the SIDD benchmark for evaluated methods: W0 through W4 (as described in the previous section), DnCNN, CBDNet and SUNet. Comparing W0 to W1 - where per-image noise estimation has been added - we observe that each block in W1 is significantly smoother. However, blocking artefacts becomes more prevalent in W1 due to the variation in image statistics between blocks. The increased predicted noise power in W1 may cause more frequency bins to reduce to zero, causing more blocking between images. This effect may also be worsened by the mean being incorrectly predicted in preprocessing, and the half-overlapping windows being insufficient to create a smooth result.

In the W2 samples, blocking is reduced as the noise estimates are made on the block level, and DC subtraction is switched from mean to median. Additionally, the double Gaussian windowing scheme is introduced here, which further mitigates blocking. Despite these improvements, some blocking remains visible, and ringing artefacts are also noticeable (as in the third sample of column (e)), caused by sharp discontinuities introduced into the high-frequency coefficients by the Wiener filter.

With W3, windowing is further refined by adopting a quarter-block overlap and utilising the multi-scale filtering strategy. Any residual blocking from W2 is eliminated (as shown in row 2), and ringing artefacts are reduced due to the smoothing effect of the multi-scale window averaging.

By introducing the Wiener coring refinement network (W4), discontinuities in frequency coefficients are minimised, effectively reducing the appearance of ringing artefacts. The images produced by this final version are smoother and less grainy than results produced by DnCNN (column (g)) and CBDNet (column (h)). Furthermore, when comparing the result for the third image (row 3) produced by the transformer SUNet, we notice a distinct staircase artefact along diagonal edges, which is not present in our final Wiener implementation. Notably, dark pixels remain dark in our results, achieving good contrast (as in row 1) due to the properties of the Wiener filter. Unlike other popular denoisers that rely on deep neural networks - which can cause blacks to appear gray in an effort to avoid vanishing gradients near zero - our method preserves black levels effectively. The improvement in DC offset removal, from mean to median subtraction, further maintains blacks in our implementation. This particular effect is clear in the first row of images, where the other DNN solutions produce grayer blacks with weaker contrast.

Gaussian Noise Samples

In Figures 4.10 and 4.11, we present higher-resolution denoising results on AWGN data generated from the DIV2K validation dataset. For each noisy image, we present results for the baseline Wiener denoiser, the DnCNN denoiser and our final WienerNet implementation. When comparing the baseline Wiener denoiser to our final implementation, the improvements are evident: no residual noise remains, and smooth areas exhibit minimal ringing artefacts.

Comparing our final implementation to DnCNN, we notice improved contrast, as demonstrated in the first image of Figure 4.10 (fly). The background appears more gray in the DnCNN filtered image, whereas the original black background is preserved in our implementation. Greater contrast is also achieved in the wings using the WienerNet denoiser, as shown in the zoomed areas where dark ridges are much clearer in the wing structure. In the second image of Figure 4.11 (sloth), the result produced by DnCNN shows that the unfocused background is uneven and spurious artefacts are introduced. In our implementation, the unfocused background remains visually pleasing and smooth.

Lastly, we note that in some cases, for high noise levels, WienerNet over-smooths regions, such as in the image of the orange (Figure 4.11). In this case, DnCNN benefits from a greater spatial receptive field, allowing it to retain more detailed textures and prevent over-smoothing. Similarly, in high-noise examples, ringing artefacts may still present themselves along high-contrast edges.

Overall, WienerNet demonstrates superior performance in preserving image contrast and reducing artefacts compared to both the baseline Wiener denoiser and DnCNN, though addressing over-smoothing in high-noise scenarios remains an area for future improvement.

4.8 Discussion

In this chapter, we have demonstrated that incorporating a simple CNN can significantly enhance the performance of traditional denoisers like the Wiener filter. This improvement was particularly evident in the noise estimation module, where even the shallowest network proposed led to a marked increase in denoising performance. Similarly, the introduction of a Wiener refinement network rapidly boosted output quality without the need for extensive tuning of the architecture or its hyperparameters. The filtered images produced by this hybrid approach are largely free from the typical artefacts associated with Wiener filtering, such as ringing, while also preserving high-frequency details and enhancing contrast, especially in areas with strong blacks.

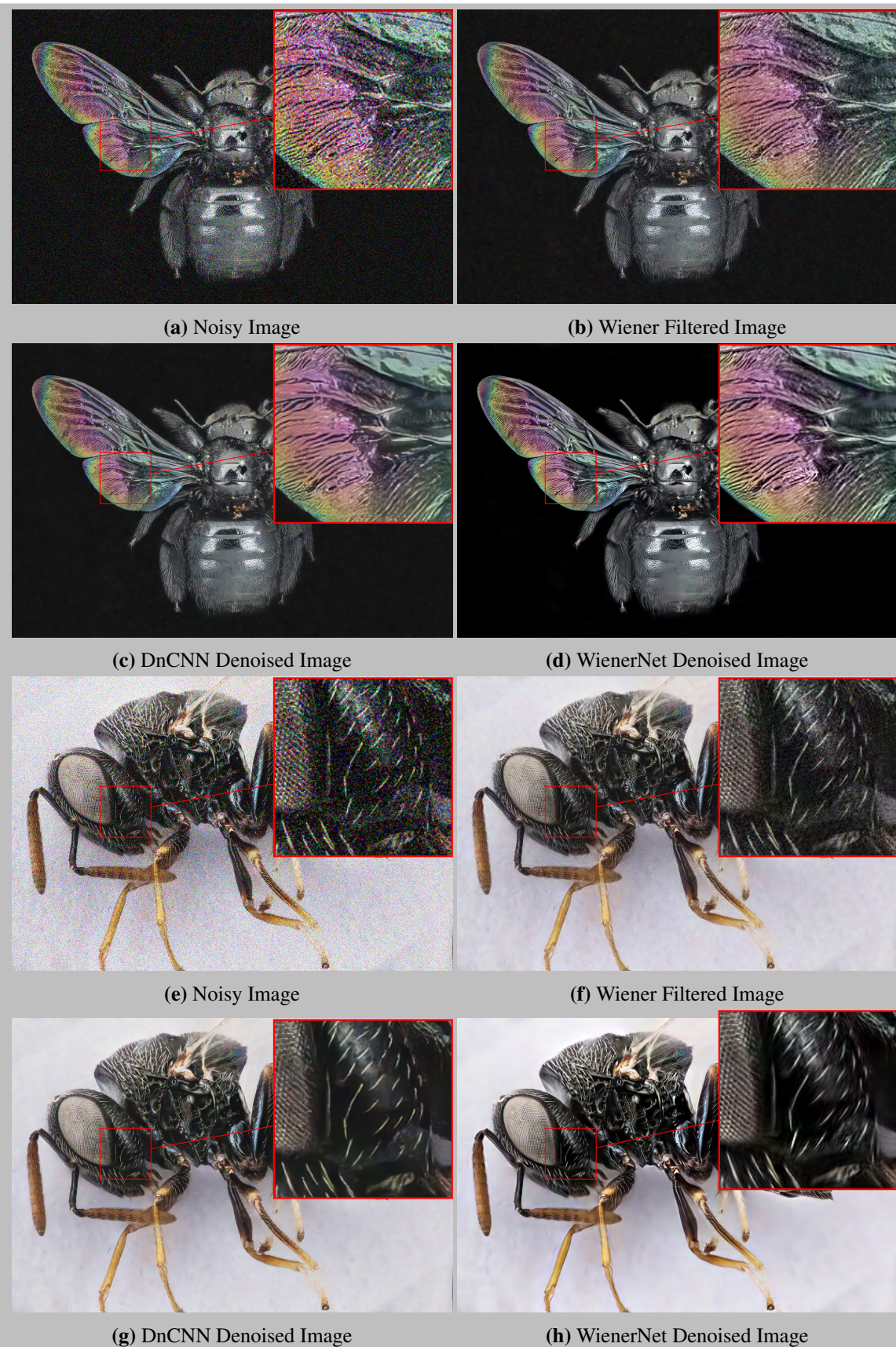


Fig. 4.10 Comparison of DnCNN and our Wiener denoisers on AWGN samples.



Further Improvements in DC-Offset Removal. In the case of DC-offset prediction, however, the introduction of a CNN did not yield best results in terms of PSNR or computational cost. In the results of the DC-offset study (Table 4.6), we show the maximum achievable performance on our dataset (38.55 dB) when the DC offset is estimated using the ground truth value. After several design iterations, we settled on a U-Net architecture in three sizes (S, M, L). Nevertheless, even the largest U-Net only reached 37.42 dB, falling short of the ideal DC component estimation by 1.3 dB. While this represents an improvement over traditional mean subtraction, the increased computational cost, due to the added network parameters, is non-trivial. As the DC component reflects the expected image in extreme filtering cases, its estimation becomes nearly synonymous with estimating the denoised image itself, which likely contributes to the challenge of training a network to predict this value. Ultimately, we found that a median pixel selection method outperformed the network, providing improved results at a fraction of the computational cost. However, there remains a 1 dB gap between this method and the ideal ground truth, indicating room for future work.

A Potentially Fully CNN Approach to Coring. During the optimisation process, we sought to directly predict the optimal Wiener transfer function H by providing the noise power P_{NN} and the signal power P_{SS} to a convolutional neural network. The input signals were concatenated together and the network was trained through the Wiener pipeline, using the L1 loss between the predicted and ground truth images to train the network. Despite many design iterations which saw numerous implementations of architectures, loss functions and training strategies, this method did not yield the expected results. With the success of the coring refinement network, we choose not to spend more resources on the approach. However, we believe the method has high potential for denoising improvement once the difficulty in training the network is resolved. This may be achieved through a coefficient normalisation scheme to handle the difference in magnitude between the input noisy and dominant clean coefficients, as well as the large magnitude differences in the output transfer function. Additionally, more success may be achieved by targeting a loss function between a ground-truth transfer function H_{opt} , generated from the clean image x , and the predicted transfer function H .

Other Areas for Improvement. Several further improvements are also possible with the proposed denoising platform. For instance, the current coring function, as defined in 4.1, introduces sharp discontinuities between frequencies by clamping transfer function coefficients to zero. Introducing a scaling factor in these cases, instead of reducing the coefficients to zero, could potentially mitigate ringing artefacts, particularly in scenarios

with high noise levels. Additionally, the multi-scale filtering presents another area for improvement. The current implementation simply averages the results without exploring whether an optimal weighting scheme might exist. Furthermore, the present setup requires a uniquely trained coring refinement network for each window size. A more efficient approach may involve designing a single model capable of serving all window sizes, potentially by upsampling the frequency coefficient representation.

A Note on Denoising Benchmarks. Finally, through our experiments, it has become apparent to us that there is a potential issue of over-fitting when using large networks on the SIDD benchmark, as some scenes in the training set are similar. If these experiments were to be repeated, it would be essential to validate our findings on another real-image benchmark to mitigate this effect. Moreover, we observed that some low-light ground truth images in the SIDD dataset contain inherent noise, as shown in dark ground truth samples in Figure 4.9. In these high-noise examples, the highest-scoring network in terms of PSNR may not necessarily present visually smooth results, but rather emulate the residual noise left in the noisy ground truth samples.

4.9 Conclusions

In this chapter, we have not focused on surpassing the SOTA, but on maximising the performance of the Wiener filter while minimising computational cost. We demonstrate that with careful consideration, the Wiener filter can perform on par, or even outperform popular CNNs such as DnCNN. We have presented a novel method of automating STD selection using a small CNN, creating a blind Wiener denoiser. Artefacts such as ringing and blocking have been improved upon, and the optimisation of window functions and finer overlapping allows for increased results. While we take advantage of neural networks to improve filtering performance, the final Wiener filter uses only 43k combined trainable parameters, while outperforming popular CNN-backbone denoisers which require millions of parameters. We note also that there is still much to gain from creating hybrid signal processing-CNN denoisers, adopting a best-of-both-worlds approach.

Chapter 5

Wiener Video Denoising Network

As a natural extension to our development of the Wiener image denoising network, this chapter focuses on the creation of a lightweight, hybrid denoiser for video denoising. Throughout the design process, we prioritise denoising speed, compact model size, and low memory requirements during inference. This focus is motivated by the increasingly high computational demands of state-of-the-art video denoisers in the literature today.

As before, we approach the optimisation of the filter by applying systematic improvements across the pipeline. This includes fundamental steps such as the tuning of analysis window characteristics and the selection of the number of frames used, to the implementation of ancillary CNNs to boost the filtering performance. Re-evaluating these stages of optimisation is necessary in the context of video denoising because increasing the dimensionality of the FFT introduces a greater computational load, resulting in slower denoising times. Conversely, the additional information brought by adjacent frames may reduce the need for densely overlapping windows. This prompts a reassessment of these parameters concerning output quality and denoising effectiveness.

In our final benchmarks, our implementation outperforms several popular denoisers and remains within 0.2 dB, on average, of the popular VRT transformer network. Moreover, our approach achieves these results over ten times faster than the transformer method and requires significantly fewer parameters.

The work presented in this chapter has been published under the title *Lightweight Video Denoising Using a Classic Bayesian Backbone* following an oral presentation at the IEEE Conference on Multimedia and Expo (ICME) in June of 2024.

5.1 Image Denoising vs. Video Denoising

Video denoising has remained an important area of research in the field of signal processing, driven by evolving trends in modern media consumption. For instance, in recent years, consumers have migrated from terrestrial television services to over-the-top streaming services. This has in turn placed greater importance on efficient video encoders, where denoisers are used as a means of reducing frame entropy, allowing for higher compression ratios at a given bitrate. The same is also true for social media platforms which distribute their content to billions of monthly users. These industry motivations extend to cinema post-production, where studios preprocess large volumes of noisy raw footage, as well as to flagship smartphone cameras, which rely on ‘AI’ post-processing to compensate for smaller sensor sizes.

However, implementing a new video denoiser is not as straightforward as reimplementing a state-of-the-art image denoiser. The nature of the signal has changed significantly with the addition of a temporal dimension, which increases the volume of data and adds complexity to the processing. Key considerations include deciding how many neighbouring frames should be used to denoise each target frame, as well as how to account for motion to ensure proper alignment of content across the frames. Additionally, the output must maintain temporal consistency, preserving a smooth appearance throughout the video sequence.

Practical limitations which are less important when developing an image denoiser, such as memory usage and denoising time per frame, become major design considerations in video denoising design. Image denoisers that leverage large transformer models with global receptive fields and hundreds of millions of parameters are challenging to scale to video. For example, the SwinIR transformer requires 3 gigabytes of memory to denoise a 512×512 16-bit image, but this demand rises to 10.7 gigabytes for a 1024×1024 image, reflecting a quadratic growth in memory usage due to the nature of self-attention and the increasing number of tokens in each window. With the addition of temporal data and batch processing, training and deploying these networks for video become exceptionally challenging tasks.

Unlike larger image denoising models, the WienerNet backbone presented in the previous chapter is sufficiently lightweight that it may be modified to accommodate the expanded input signal without requiring excessive amounts of memory during training. As well as this, the per-frame denoising speed of the Wiener filter should remain competitive due to the parallelisation of the filter. Lastly, as we will further discuss, the nature of frequency filtering is such that dedicated motion compensation modules are not critical to the performance of the Wiener filter, unlike popular denoisers like FastDVDNet.

We carry out this design process with current video denoisers in mind. While published in 2020, FastDVDNet remains the most popular video denoiser to date, due to unmatched denoising speeds. In our optimisation, we seek to approach the denoising speeds of this network while producing benchmark scores that match those of more recent (and far slower) transformer architectures such as VRT (Video Restoration Transformer). Similarly to image denoisers, video denoisers have also grown in size, through the years, as shown in Table 5.1. In this work, effort is also placed into minimising network size as before.

Lastly, it is worth noting that the open-source research landscape for video denoising is not as extensively developed as it is for image denoising, due to several key factors with important implications. First, video processing is inherently more labour-intensive and demands significantly higher computational resources, which poses a barrier to entry for smaller research groups. Second, the use of real-noise datasets for training and evaluation is not yet prevalent in the video denoising literature, resulting in the absence of standardised blind benchmarks and requiring researchers to establish their own. Although some models trained on synthetic noise exist [275, 276], most video denoising architectures are non-blind, with separate models trained for different noise levels. Given these considerations, we have chosen to focus our investigation on Gaussian noise, narrowing the scope to the optimisation of the 4D Wiener denoiser. Nevertheless, this work does include the implementation of a blind, noise-predictive model, ensuring that our approach can generalise beyond the specific case of Gaussian noise.

In Section 5.2 and Section 5.3, the implementation of a classic Wiener filter for video denoising is discussed, along with the modifications and optimisations that will need to be carried out to convert our image denoiser to a video denoiser. In Section 5.4, a series of ablation studies are carried out, followed by final results of these studies in Section 5.4.6 in which our proposed networks are compared to the state of the art, along with example outputs from the network.

5.2 Wiener for Video Denoising

To apply the Wiener filter to image sequences, the coring function is modified to accommodate the temporal dimension. the IIR solution is otherwise identical to the 2D case given in Equation 4.1:

$$\hat{H}[\omega_i, \omega_j, \omega_t] = \frac{\max \{P_{YY}[\omega_i, \omega_j, \omega_t] - \lambda \cdot P_{NN}[\omega_i, \omega_j, \omega_t], 0\}}{P_{YY}[\omega_i, \omega_j, \omega_t]}, \quad (5.1)$$

Network	PSNR	# Params (M)	Year
DVDNet	35.77	1.25	2019
FastDVDNet	35.77	2.5	2020
PacNet	36.82	2.9	2021
RVRT	38.05	10.8	2022
VRT	38.15	35.6	2022
ReVID	38.50	15.0	2023

Table 5.1 Comparison of popular video denoising networks, highlighting their PSNR performance, model sizes (in millions of parameters), and release years. The table illustrates the trend of increasing network sizes over time to achieve higher denoising performance.

with the only difference being the addition of the temporal index coefficient, ω_t . As before, the signal mean is removed, the signal is windowed, and a 3D FFT replaces the 2D FFT used in the baseline image implementation. Next, the Wiener transfer function, H , is found using Equation 5.1 and it is applied to the noisy signal coefficients Y .

In our implementation, the frequency coefficients are represented by seven-dimensional PyTorch tensors with the shape $[\text{batch_size}, \text{num_frames}, 3, M_x, M_y, H_i, W_j]$, where batch_size denotes the number of samples in a batch, num_frames represents the number of frames, the third dimension corresponds to the three colour channels, and $M_x \times M_y$ specifies the spatial block positions, while i and j index the frequency coefficients. With the filtering complete, the signal is returned to the spatial domain, the mean is reapplied, and the signal is reconstructed using the overlap-add method. This baseline implementation is summarised in Algorithm 3.

5.3 Optimisation and Enhancing of the Baseline

We begin the optimisation process from a baseline 3D Wiener video denoising filter implementation. This version makes use of half-overlapping raised-cosine windows whose magnitudes sum to 1 for the reconstruction of the signal post filtering. The spatial window size is set to 32×32 for initial experiments as it performed well in image denoising tests. In contrast to Kokaram’s Wiener filter, our initial approach does not include motion compensation; we defer its integration until the remaining optimisations are complete. The filter operates in a causal manner, processing $2N + 1$ frames for each denoised frame: the target frame along with N neighbouring frames on either side.

Algorithm 3 Kokaram's Video 3D Wiener Filter[138]**Require:** Noisy image seq, noise STD σ , Block Size

```

1:  $w(t, h, k) \leftarrow \text{RaisedCosine}(t, h, k)$  ▷ windowing definition
2: for all frames in seq do
3:    $y \leftarrow$  3D framebuffer made of current grayscale frame and 4 nearby motion compen-
     sated neighbouring frames
4:   for all blocks  $y$  in  $y$ , for stride=BlockSize/2 do
5:      $\bar{y} \leftarrow \text{mean}(y)$  ▷ predicts block mean
6:      $y_w \leftarrow (y - \bar{y}) \odot w$  ▷ windowing
7:      $Y \leftarrow \text{FFT3D}(y_w)$ 
8:      $P_{yy} \leftarrow Y \odot Y^*$ 
9:      $P_{nn} \leftarrow \hat{\sigma}^2 \|w\|^2$ 
10:     $P_{xx} \leftarrow \max(P_{yy} - P_{nn}, 0)$  ▷ coring
11:     $\hat{x}_w \leftarrow \text{iFFT3D}(Y \odot P_{xx} \oslash P_{yy}) + \bar{y}w$ 
12:     $\hat{x} \leftarrow \text{overlap\_add}(w \odot \hat{x}_w)$  ▷ combine blocks

```

The first step we take in improving the output quality of the filter is expanding the FFT computation to include not only the spatial and temporal dimensions but also the colour channels. As demonstrated in our image denoiser, incorporating the colour channels in the FFT enhances denoising performance by capturing the correlations across the RGB channels, allowing the filter to more effectively distinguish between noise and true signal. In the video context, this approach extends the benefits to spatio-temporal correlations, enabling the filter to better preserve colour consistency across frames and improve the overall quality of the denoised video by addressing noise in a more coherent multi-dimensional frequency space.

5.3.1 Windowing Optimisations and DC Offset Removal

As we have mentioned in our introduction, due to the change in the nature of the signal, as well as the expanded FFT, we must reconsider the windowing optimisations to optimise for denoising quality and execution time. The first and most obvious assessment to make is to find the optimal number of neighbouring input frames, N , used to calculate the output frame. In Section 5.3.1 we assess the impact of temporal window size on denoising quality to assess the optimal number of neighbouring frames.

Next, we carry out an ablation study to determine the optimal spatial stride, measured as a fraction of the block's width - from half a block's width to an eighth of a block's width - between windowed blocks. As well as denoising quality - measured in PSNR (dB) and SSIM - the denoising time is assessed for this experiment. As we have seen in Chapter 3, by moving the position of the raised cosine window such that they no longer overlap by half the window

width, the overlapping sections no longer sum to one. To ensure smooth transitions between windows, we modify the synthesis window scheme. The new windows are firstly applied to the image blocks post filtering, before the overlap-add is applied. The sum of the overlapping windows is also computed, and this weight map is used to normalise the overlapped image patches to create a smooth output.

The DC component subtraction is evaluated again in this study to measure the impact of median subtraction on the expanded signal. Gaussian noise of five different noise levels [10, 20, 30, 40, 50] is assessed on the test dataset for mean subtraction, median subtraction, and the ground-truth mean subtraction.

Following this, the spatial window size is assessed for denoising time and output quality. As before, the average quality is found across the testing dataset and the window is tested from 14×14 to 126×126 .

In our final Windowing optimisation, the impact of the window shape is assessed. As before, we compare the performance of the Cosine window against Gaussian windows, using optimised standard deviation values for both the analysis and synthesis window. Additionally, we explore the potential for further improvement by training the window shape itself. To achieve this, we start by initialising two Gaussian windows (analysis and synthesis), duplicating them for the total number of frames (N), and converting these into trainable weights. For a five-frame Wiener filter ($N = 5$), this results in a total of 10 trainable windows.

These weights are used solely for modulating their respective image blocks, without being part of a larger network. The window shape does not depend on an input, and there are no additional layers, activation functions, convolutions or biases involved. During inference, the trained weights functions are loaded as constants, adding no extra computation overhead compared to traditional windows.

A second approach for the trainable window is also tested, which enforces a positive, isotropic 2D shape. Instead of learning the optimal value for each individual weight directly, this method learns a 1D scaling function for the window weights based on the distance from the window's centre. Once the 1D scaling factors are determined, they are extrapolated across the entire window to generate the final 2D shape during inference. In this approach, we will assess whether applying more consistent weighting independent of orientation produces higher quality results.

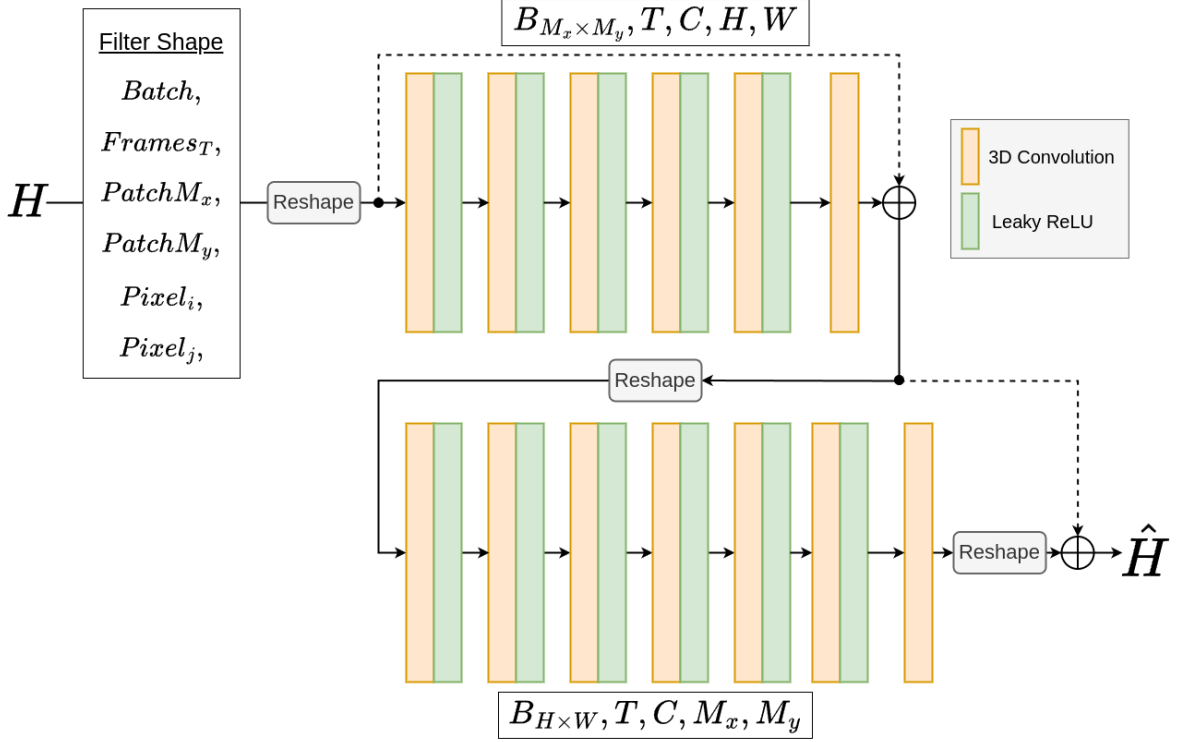


Fig. 5.1 Architecture of the two-stage coring refinement network, designed to optimise the initial prediction of the coring function H .

5.3.2 Wiener Coring Refinement Network for Video

In a similar manner to Chapter 3, for the expanded video signal representation, we propose to refine the predicted Wiener transfer function $\hat{H}[\omega_t, \omega_c, \omega_i, \omega_j]$ (Equation 4.1) using a new lightweight CNN architecture. This network is designed with the goal of reducing ringing artefacts and spurious textures generated by the poor localisation of the Wiener filter.

A simplified 2D block diagram of our proposed network is presented in Figure 5.1. To accommodate the larger 4D signal, the network is redesigned from scratch. 3D convolutional filters are implemented instead of 2D convolutions, with $3 \times 3 \times 3$ kernels applied to 3D temporal image sequence blocks. As usual, the expanded feature maps are arranged across the colour channel. The two-stage layout of the network is reused, allowing for sharing of pixel values between the spatio-temporal windows.

The network accepts the initial prediction for the transfer function, H , of shape $[B \times T \times C \times M_x \times M_y \times H_i \times W_j]$, where B represents the batch dimension, T represents the temporal dimension size, M_x and M_y represent the number of overlapping windows horizontally and vertically respectively, and H_i and H_j represent the height and width of each window. Before

applying the 3D convolutions, the data must be rearranged to a five-dimensional tensor. To do this, the block dimension of the analysis windows are contiguously consolidated to the batch dimension B , forming a tensor of shape: $[B_{x,y} \times T \times C \times H \times W]$, where $B_{x,y}$ contains the ‘stacked’ patches of all batches.

In this form, the tensor is passed through the first stage of the coring refinement network. In the second stage of refinement, the tensor is rearranged to have shape $[B_{H,W} \times T \times C \times M_x \times M_y \times]$ such that we may refine the network via inter-block pixel relationships. We name two parts of the network the *intra-block* and the *inter-block* stages respectively.

The final network is composed of 11, 3D-Convolution layers, each paired with a LeakyReLU activation, with the exception of the last layer in each block, from which they are omitted. 40 filters/channels are used throughout, and each convolution is bias-free to improve generalisation on unseen data [198]. The low number of features per layer is chosen to reduce memory consumption during inference. The network is trained using a weighted loss scheme which emphasises the output quality of the centre frame. Specifically, we have two L1 losses: that of the target centre frame, and that of the remaining 4-frame sequence:

$$\mathcal{L}(\theta) = \frac{1}{(N-1) + 10} \left(\sum_{i=0, i \neq 2}^{N-1} \|\hat{x}_i - x_i\| + 10\|\hat{x}_2 - x_2\| \right). \quad (5.2)$$

As shown above, we weigh the L1 loss for the output centre frame x_2 to have ten times the weight of the other individual frames.

The final network consists of 289,755 trainable parameters. We initially implement a blind denoising approach when training this network, as is common in the current literature. This means that a network is trained for multiple Gaussian noise levels [10, 20, 30, 40, 50] and the user must select the desired model before inference.

5.3.3 Blind denoising

Although real noise is not evaluated in this chapter, we still seek to implement a blind version of the Wiener video denoiser. This is motivated by the current lack of blind denoisers in the state of the art for video denoising. In order to achieve this, we implement a small 2D ancillary network which accepts the centre frame of the current input sequence and produces a per-block estimate of the noise standard deviation. This module is first trained in isolation, learning to predict the standard deviation of each image in the batch via an L1 loss between the ground truth noise standard deviation value and the predicted value. The STD value applied to each clean image is sampled from a uniform distribution between 1 and 50 during

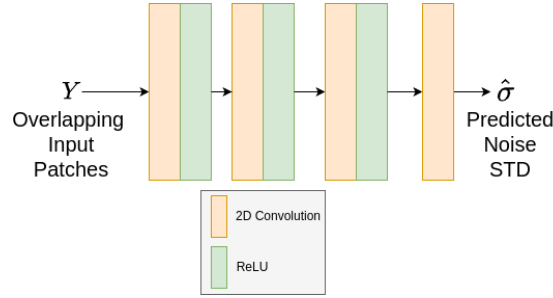


Fig. 5.2 The lightweight 2D noise estimation CNN used to predict the noise for each block of the centre frame in the input sequence. The final implementation employs 20 feature maps per layer.

the loading of the batch. In the second stage of training, the coring refinement network is trained while the weights of the noise prediction network are also left unfrozen for further fine tuning. During this stage, the weighted loss described in the previous section is used, targeting only output image quality.

To maintain a small network size, the network predicts the noise standard deviations for the centre frame only, which is then duplicated for the neighbouring frames. Every channel and window in the target frame, however, still receives unique STD values. The network we implement for this purpose consists of only four 3×3 2D convolution layers, with 20 features per layer and three ReLU activations, as shown in Figure 5.2. In this configuration, the network contributes only 4,763 additional parameters to the overall network.

5.3.4 Multi-Scale Wiener Filtering

With the windowing optimised and the coring refinement and noise prediction networks implemented, we return to the implementation of the multi-scale Wiener filter, where the image is denoised at multiple block sizes, and the outputs are averaged. Capturing multi-scale frequency information in this manner increases the amount of information available to the denoiser and creates a smoother final image. Smaller window sizes perform better for high frequency areas where localisation is more important, while large window sizes perform well for flat regions, which produces satisfyingly smooth results.

5.3.5 Motion Compensation as a Preprocessing Step

The estimation of motion between frames is a classic, and often essential step in the pipeline for classic and SOTA video denoising algorithms. By using motion vectors to align frames relative to the target frame, adjacent frames can provide alternate representations of the target frame's content, with different noise realisations. This alignment allows for more

effective noise reduction, as information from multiple frames can be combined to enhance the true signal while attenuating noise. Neglecting motion in video denoising often leads to temporally inconsistent results with shimmering artefacts in low-frequency areas and ghosting around edges becoming more noticeable.

Classic video denoisers which utilise motion estimation include Kokaram’s Wiener filter [27], which uses multi-resolution, block matching with thresholding for motion estimation, Video Non-Local Bayes [32] (VNLB) which uses TV-L1 regularisation [277] for motion estimation, and VBM4D [278], which employs block matching for motion estimation. Deep learning-based denoisers also integrate motion estimation; for instance, DVDNet [31] leverages the DeepFlow [279] optical flow algorithm to align input frames, while some of the high-performing video denoisers such as ReVID [280] incorporate custom optical flow prediction networks within their architectures to align latent features. ReVID additionally employs loss functions that account for clean-noisy flow estimation errors in their loss functions to improve temporal consistency. On the other hand, networks that do not explicitly perform motion compensations, such as FastDVDNet [223], instead cascade the input frames through a sequence of U-Nets, processing five frames in overlapping sets of three. Each U-Net network produces one frame, and the results are then fused by a final U-Net to generate a denoised central frame. This approach minimises motion-related artefacts without directly applying motion compensation.

However, for larger and more complex networks, the impact of motion estimation on denoising performance is not always straightforward. For example, with DVDNet, some video sequences show degraded results when motion estimation is applied. To thoroughly evaluate the influence of motion compensation in trained networks, we compare the denoising performance of our optimised, Wiener filter with and without motion compensation, for both our trained Wiener filter and our untrained filter. We choose to assess motion compensation using the widely-used DeepFlow algorithm and the transformer-based RAFT model [281] for their established accuracy in optical flow estimation.

To summarise our proposed implementation for the 4D Wiener denoiser, we present a pseudocode implementation in Algorithm 4.

5.4 Experimental Results

In this section, we iterate through proposed optimisations made to the Wiener video denoising filter, measuring quality improvements at each step. The number of frames and FFT dimensions are analysed, followed by the preprocessing hyperparameters. With the Wiener

Algorithm 4 Our Video 4D Wiener Filter**Require:** Noisy image seq., Block Size

```

1:  $w_a \leftarrow \text{get\_trained\_window\_weights}(\text{analysis\_window})$   $\triangleright$  load analysis window
2:  $w_s \leftarrow \text{get\_trained\_window\_weights}(\text{synthesis\_window})$   $\triangleright$  load synthesis window
3: for all frames in seq do
4:    $y \leftarrow$  4D frame-buffer made of current RGB frame and 4 motion compensated neighbouring RGB frames
5:    $\hat{\sigma} \leftarrow \text{CNN}_{\sigma}(y_2)$ 
6:   for all blocks  $y$  in  $y$ , for stride=BlockSize/3 do
7:      $\bar{y} \leftarrow \text{median}(y)$   $\triangleright$  predicts block' DC offset
8:      $y_w \leftarrow (y - \bar{y}) \odot w_a$   $\triangleright$  analysis window
9:      $Y \leftarrow \text{FFTN}(y_w)$ 
10:     $P_{yy} \leftarrow Y \odot Y^*$ 
11:     $P_{nn} \leftarrow \hat{\sigma}^2 \|w\|^2$ 
12:     $P_{xx} \leftarrow \max(P_{yy} - P_{nn}, 0)$   $\triangleright$  coring
13:     $\hat{x}_w \leftarrow \text{iFFTN}(Y \odot P_{xx} \oslash P_{yy}) + \bar{y}$ 
14:     $x_{\text{all}} \leftarrow \text{overlap\_add}(w_s \odot \hat{x}_w)$   $\triangleright$  combine blocks
15:     $w_{\text{all}} \leftarrow \text{overlap\_add}(w_s \odot w_a)$   $\triangleright$  combine windows
16:     $\hat{x} \leftarrow \frac{x_{\text{all}}}{w_{\text{all}}}$   $\triangleright$  denoised frame

```

filter optimised, we introduce our ancillary networks before implementing a final multi-scale version. For our initial baseline, use half overlapping 38×38 cosine windows with five input frames and mean DC offset removal.

5.4.1 Training and Test Sets

We evaluate our denoiser using ten, 64-frame sequences, taken from a combination of Derf's Collection [282] (HD, gaming) and the BVI (SynTex [283], DVC[284]) datasets. Each clip is centre-cropped to 500×500 for evaluation. Additive Gaussian noise is applied to the datasets at standard deviations of $\sigma = [10, 20, 30, 40, 50]$. Each sequence is saved at every noise level to ensure identical test cases. A frame from each test sequence is shown in Figure 5.3.

For training, we choose 173 unrelated, uncropped, full-length videos from the BVI and Derf corpus, omitting the test sequences. When training the broad noise, blind denoiser, the noise added to each image is sampled from a uniform distribution of $\sigma \sim U(0, 50)$. During training, file paths are randomly chosen and five frames are randomly selected for each sequence in the batch and randomly cropped to a size of 5×128×128.



Fig. 5.3 Sample clean frames from the test dataset used in our experiments, featuring images sourced from the BVI [283, 284] and Derf’s collections [282]. These examples illustrate the diversity of content and image quality in the benchmark.

5.4.2 From 2D to 4D: The Effects of Multi-Dimensional FFT on Video Denoising Performance

The baseline Wiener filter is evaluated at multiple levels of FFT operations: 2D, 3D, and 4D FFTs. These results are presented in Table 5.2. Each data point represents the average PSNR across ten denoised sequences, providing a robust comparison for various noise standard deviations (STDs), ranging from 10 to 50.

The 2D FFT baseline applied across the spatial provides the lowest scoring PSNR results and results in temporally inconsistent results. For the 3D FFT configurations, we evaluated two setups: one which applies the FFT across the colour channels following the 2D FFT (3D Colour FFT), and another that applies the FFT across the temporal channels following the 2D FFT (3D Temporal FFT). The 3D Temporal FFT outperforms the 3D Colour FFT, highlighting the greater benefit of utilising temporal information for video denoising. This suggests that integrating temporal features helps preserve motion continuity and reduces noise more effectively than colour-based frequency filtering alone.

The 4D FFT approach, which combines spatial, colour, and temporal dimensions, achieved the highest PSNR values across all noise levels, demonstrating the most effective denoising performance. This configuration significantly enhanced noise reduction, benefiting from the comprehensive use of all available dimensions in the frequency domain, leading to superior restoration quality and temporally consistent results.

5.4.3 Number of Input Frames

Next, the impact the number of input frames has on the denoising result is evaluated. We carry out this study on our denoising dataset at a noise STD of $\sigma = 20$. The performance is evaluated between $N = 3$ and $N = 15$, adding one frame on either side of the target

AWGN STD	2D Spatial FFT	3D Colour FFT	3D Temporal FFT	4D Temporal + Colour FFT
10	33.12	34.42	34.87	36.02
20	29.48	30.60	31.20	32.23
30	27.48	28.44	29.05	29.97
40	26.05	26.87	27.45	28.25
50	24.89	25.58	26.09	26.77

Table 5.2 Ablation study results comparing Wiener video denoising performance using 2D, 3D, and 4D FFT operations. The 2D FFT is applied across the spatial dimensions, while the 3D FFT is evaluated with combinations of spatial and colour dimensions or spatial and temporal dimensions. The 4D FFT incorporates spatial, colour, and temporal dimensions. Denoising performance is measured in terms of PSNR (dB).

Number of Frames	3	5	7	9	11	13	15
PSNR (dB)	31.89	32.30	32.49	32.57	32.59	32.59	32.57
SSIM [0 – 1]	0.8377	0.8439	0.8459	0.8459	0.8451	0.8439	0.8424
Memory (GB)	1.3	1.6	2.0	2.6	3.0	3.4	3.8
Time (s)	12.66	17.15	21.65	26.01	30.11	33.90	37.07

Table 5.3 Ablation study results evaluating the effect of varying the number of input frames on Wiener denoising performance metrics (PSNR and SSIM), processing time (seconds), and maximum video memory usage during inference (gigabytes).

frame between experiments. We present these results in Table 5.3, which also includes the maximum video memory consumed during inference and the sum of the denoising time for the 10, 64-frame sequences (excluding data loading).

As N increased from 3 to 15, we observed notable changes in both denoising effectiveness and computational requirements. Specifically, the PSNR improved from 31.89 dB at $N = 3$ to a maximum of 32.59 dB at $N = 11$. Similarly, the SSIM increased from 0.8377 to a peak of 0.8459 at $N = 7$ and $N = 9$. These enhancements indicate that incorporating more frames generally leads to better denoising performance due to the additional temporal information available. However, beyond certain values of N , the improvements in PSNR diminish. As more frames are added, motion between frames introduces content not relevant to the target centre to the FFT analysis. The inclusion of such motion-induced discrepancies can cause the algorithm to consider inconsistent information, thus lowering the PSNR.

On the computational side, both processing time and memory usage increased approximately linearly with the number of input frames. Processing time rose from 12.66 s at $N = 3$ to 37.07 s at $N = 15$, while memory consumption grew from 1.3 GB to 3.8 GB. The linear growth indicates that each additional frame adds a consistent computational burden. This escalation in resource requirements can be a critical factor, especially in environments with limited computational capabilities or where real-time processing is needed.

Balancing the gains in denoising performance against the increased computational costs, we find that using $N = 5$ input frames offers a favourable trade-off. This marks a +0.41 dB improvement over the $N = 3$ case with a relatively modest increase in processing time and memory usage. Beyond $N = 5$, the performance gains become marginal, whereas the computational demands continue to grow substantially. Therefore, proceeding with an input size of $N = 5$ frames is a practical choice.

5.4.4 Windowing and DC-Offset Optimisation

In the following experiments, we evaluate the hyperparameters of the FFT preprocessing pipeline used in the Wiener filter, namely the windowing parameters and DC offset removal approach.

Window Stride. Firstly, we examine the window stride — or the overlap between blocks — as a fraction of the window size, ranging from a stride of half a block width to one-eighth of a block width. As discussed in Section 5.3.1, a new synthesis window is introduced to account for the overlapping blocks only summing to one for half overlaps.

In Table 5.4, we present the results of this experiment. As before, we record PSNR and SSIM, as well as the denoising time for the entire dataset. Denoising times are faster here than in our previous experiment because the evaluation was carried out on a more powerful GPU. A noise standard deviation of $\sigma = 20$ is used again.

We observe a PSNR gain of +0.19 dB by reducing the stride from half to one-third of a block width. Further reducing the stride to one-fifth of a block width yields a maximum gain of +0.23 dB but increases processing time by over 2.5 times. As observed in the image denoising implementation, the significant increase in denoising time with reduced stride is due to the quadratic cost associated with adding more windows to the filter. Unfolding the original image into patches — the most costly operation in the pipeline—depends on the number of windows. As a result, we proceed with a stride equal to one third of the block width.

Stride	PSNR (dB)	SSIM ([0-1])	Time (s)
1/2	32.32	0.84756	5.39
1/3	32.51	0.85287	13.26
1/4	32.53	0.85344	19.94
1/5	32.54	0.85364	35.19
1/6	32.54	0.85365	16.83
1/7	32.55	0.85370	49.80
1/8	32.55	0.85370	78.46

Table 5.4 Ablation study results examining the effect of Wiener window stride, expressed as a fraction of block size, on output quality (PSNR and SSIM) and denoising time. A block size of 32×32 is used. Quality metrics are averaged across the 10-sequence dataset, while time measurements represent the total denoising time for all sequences.

Block Size. We continue the windowing optimisation by considering the impact of the block size, using the same noise STD as before ($\sigma = 20$). In Figure 5.4 we plot denoising quality w.r.t block size for both PSNR and SSIM. We observe that PSNR peaks at a window size of 20 (32.63 dB), and decreases as the analysis block size increases to 126 (31.78 dB). SSIM peaks at a window of size 32 (0.8523), and the lowest quality (0.8442 SSIM) is recorded at the largest window size, 126. Increasing window size also results in slightly lower denoising times. While larger windows slow down the FFT transforms, fewer windows are required to cover the images, and as the cost of unfolding the image into patches is greater than the FFTs themselves, the total denoising time falls. As a result, we proceed with the window size of 32×32 as perceptual quality (SSIM) is highest while PSNR remains within .1 dB of the highest result for the dataset.

DC Offset Optimisation. As mentioned in Section 5.3.1, it is necessary to zero-mean the temporal block before applying the FFT. For this purpose, we compare classic mean subtraction to median subtraction. To assess the performance loss incurred by using the noisy mean, we also evaluate the denoiser when provided with the unseen ground truth mean. For the mean method, the best results were obtained by subtracting the mean of each individual frame from itself. In contrast, the median method achieved optimal performance by subtracting the median computed across all five frames (for each channel). In Table 5.5, we show that, for all noise levels, using the median of the block improves denoising performance compared to subtracting the mean. This effect is most noticeable at higher noise standard deviations, where pixels near brightness boundaries deviate more significantly from their ground truth values. At a noise level of $\sigma = 50$, we observe a **+0.22 dB** increase in PSNR over using the noisy mean. Interestingly, the SSIM metric reports higher quality for the

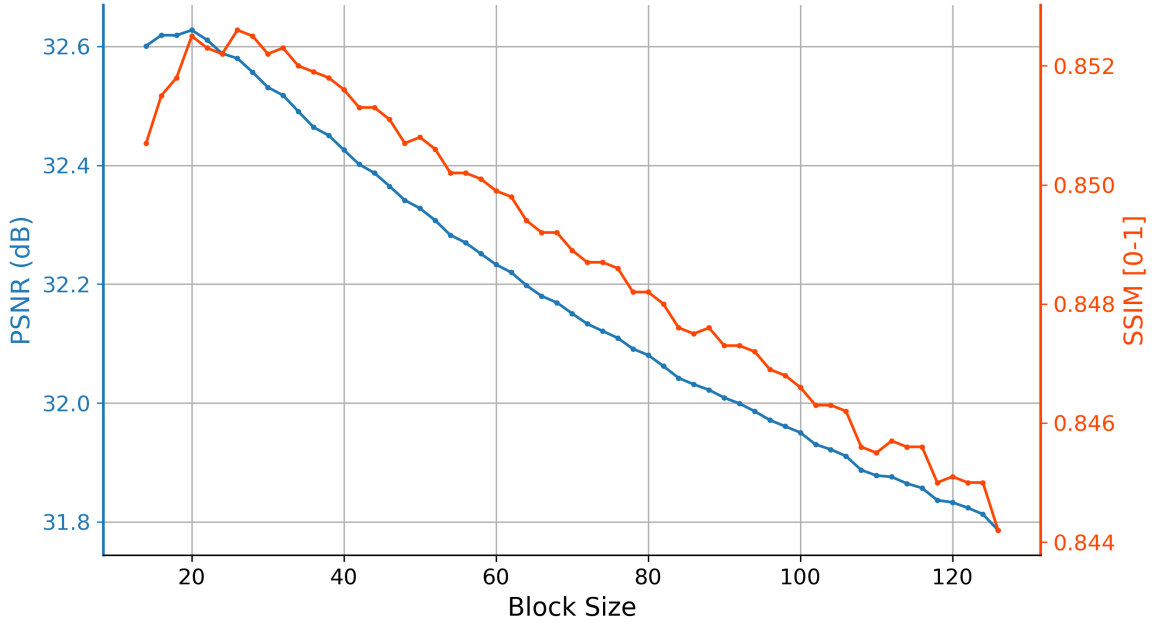


Fig. 5.4 Graph evaluating the impact of Wiener window block size on output quality, measured in terms of PSNR (dB) and SSIM (0–1). Quality metrics are averaged across the 10-sequence test set, with a $1/3$ overlap stride applied.

median case than when using the ground truth mean, which may be attributed to the emphasis SSIM places on luminosity.

Trainable Windows Next, the impact of window shape on denoising performance is evaluated. In addition to the raised-cosine and Gaussian windows, we introduce two new analysis-interpolation window pairs: trainable Gaussian (non-isotropic) windows and trainable isotropic windows.

The trainable Gaussian windows are initialised as normal 2D Gaussian windows, with their weights made fully trainable.

The isotropic window is defined by a learnable 1D radial profile $w(r)$, applied to each pixel based on its normalised Euclidean distance from the window centre (i_0, j_0) :

$$r_{i,j} = \sqrt{\frac{(i - i_0)^2 + (j - j_0)^2}{2B^2}}$$

Each window value is computed by linearly interpolating the discrete profile w as:

$$W_{\text{iso}}(i, j) = (1 - \alpha)w_{\lfloor r_{i,j} \rfloor} + \alpha w_{\lfloor r_{i,j} \rfloor + 1}, \quad \alpha = r_{i,j} - \lfloor r_{i,j} \rfloor$$

Noise STD	Mean PSNR / SSIM	Median PSNR / SSIM	GT PSNR / SSIM
10	36.25 / 0.91379	36.26 / 0.91425	36.27 / 0.91423
20	32.52 / 0.85226	32.56 / 0.85317	32.57 / 0.85282
30	30.24 / 0.80271	30.33 / 0.80459	30.33 / 0.80379
40	28.43 / 0.76035	28.59 / 0.76347	28.59 / 0.76214
50	26.84 / 0.72358	27.06 / 0.72809	27.07 / 0.72621

Table 5.5 Ablation study results comparing DC offset removal strategies as preprocessing steps for the Wiener filter, using a 32×32 window. The ‘GT’ column represents the ground truth mean, calculated from the unknown clean image, and serves as a reference benchmark. Results are reported as the average PSNR/SSIM across 10 video sequences.

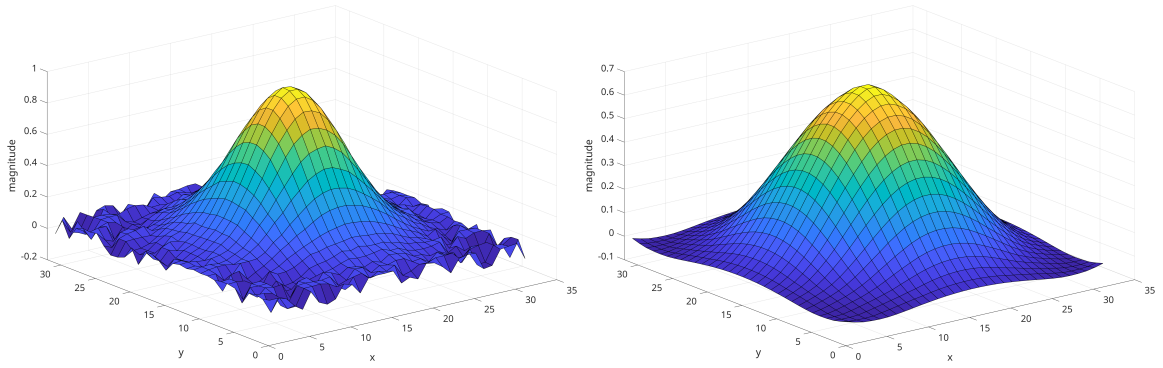


Fig. 5.5 3D surface plots visualising the trained 3D video denoising windows. Left: an unconstrained trained window, initialised as a Gaussian window. Right: an interpolated trained isotropic window. For a five-frame analysis, ten unique windows are generated: five analysis windows and five synthesis windows. Both plots show the red channel analysis window for the centre frame.

Here, $\alpha \in [0, 1)$ represents the fractional distance between the two nearest samples of the profile. It controls the relative weighting between the values $w[k]$ and $w[k + 1]$, where $k = \lfloor r_{i,j} \rfloor$. When $\alpha = 0$, the interpolation returns $w[k]$; when $\alpha = 1$, it returns $w[k + 1]$; intermediate values blend the two linearly. This construction ensures that the window function can be evaluated smoothly at any pixel location, while maintaining radial symmetry and a compact set of learnable parameters.

The window weights are trained using the same loss function described in Section 5.3.2, with the AdamW optimiser [285] and a cosine annealing learning rate scheduler that reduces the learning rate from $1e^{-3}$ to $1e^{-5}$ every 300 epochs, over a total of 1200 epochs. In Figure 5.5, 3D surface plots are shown for the resultant Gaussian and isotropic windows. While ten windows are generated for the five-frame denoiser (five analysis and five synthesis windows), only the centre red channel analysis window is shown in this case.

Scene	Cosine PSNR / SSIM	Gaussian PSNR / SSIM	Trainable PSNR / SSIM	Isotropic PSNR / SSIM
India	34.15 / 0.9554	34.29 / 0.9565	34.42 / 0.9561	34.31 / 0.9568
Market	32.34 / 0.8514	32.19 / 0.8441	32.30 / 0.8482	32.31 / 0.8494
DOTA2	35.18 / 0.9020	35.20 / 0.8995	35.33 / 0.9015	35.29 / 0.9035
Hamster	30.67 / 0.5349	30.67 / 0.5337	30.71 / 0.5340	30.68 / 0.5342
Shopping	34.37 / 0.9554	34.54 / 0.9572	34.69 / 0.9550	34.55 / 0.9573
Football	29.37 / 0.8884	29.28 / 0.8854	29.37 / 0.8881	29.38 / 0.8887
Tree	32.81 / 0.8524	32.86 / 0.8500	32.93 / 0.8518	32.92 / 0.8540
Minecraft	28.35 / 0.7420	28.26 / 0.7341	28.27 / 0.7370	28.32 / 0.7396
Bridge	34.94 / 0.9565	35.03 / 0.9562	35.19 / 0.9563	35.07 / 0.9573
Christmas	33.44 / 0.8933	33.44 / 0.8904	33.55 / 0.8927	33.53 / 0.8941
Mean	32.56 / 0.8530	32.58 / 0.8507	32.68 / 0.8521	32.64 / 0.8535

Table 5.6 Ablation study results evaluating the effect of different windowing functions on denoised quality, measured in terms of PSNR and SSIM. ‘Trainable+’ denotes a trained window constrained to positive values only. All experiments are conducted on the test set with a noise standard deviation of $\sigma = 20$, using a window size of 32×32 and a quarter overlap.

Denoising results for the different window shapes are presented in Table 5.6. In this experiment, results are reported for each video sequence at an AWGN noise level of $\sigma = 20$. The raised-cosine window is outperformed by the trainable Gaussian window by **+0.10 dB** in PSNR, with no additional inference cost. Meanwhile, the isotropic window achieves the best SSIM scores. We proceed with the trainable Gaussian window in subsequent experiments, as our focus remains on maximising PSNR.

5.4.5 Multi-Scale Filtering for Video Denoising

With the preprocessing pipeline optimised, we extend our investigation by evaluating the performance of a multi-scale approach to video denoising. The multi-scale Wiener denoiser leverages multiple window resolutions to better capture spatial and frequency information across different scales.

In this approach, we retrain the coring refinement network for window sizes of 16×16 , 32×32 , and 64×64 , creating three distinct networks tailored to different spatial resolutions. During inference, the Fourier domain coefficients are processed according to their respective window sizes, and the outputs from the three networks are combined by taking the per-pixel mean result. This method enables us to assess whether integrating information from multiple scales improves denoising quality, particularly for textures with varying frequency

characteristics. We evaluate the performance of the multi-scale denoiser in the following section.

5.4.6 Final Results: Coring Refinement Network

With the preprocessing pipeline optimised, we now evaluate the performance of the coring refinement network, as described in Section 5.3.2, and compare our final results to the state of the art in video denoising. The network is trained with optimised hyperparameters: 5 frames, 4D FFTs, 32×32 trained Gaussian windows, median DC offset removal, and a $1/3$ block stride.

For our non-blind denoiser, we train five networks separately, each targeting a specific Gaussian noise profile: $\sigma = [10, 20, 30, 40, 50]$. These non-blind models require the user to provide the noise standard deviation as input.

For the blind denoiser, we employ a two-stage training process. In the first stage, a coarse training phase is conducted where the noise prediction module is trained using an L_1 loss to improve noise estimation accuracy. In the second stage, the module is fine-tuned within the Wiener filter framework using the same 5-frame loss function as before. Finally, the coring refinement network is introduced, and its weights are trained while freezing the noise prediction module. This blind denoiser generalises to noise STDs in the range $[0-50]$, requiring no user input.

Performance Summary and Comparisons

In Table 5.7, we present our final benchmark results, comparing the performance of our models (WNet, WNetBlind, and WNetBlind_{MS}) to popular denoisers such as DVDNet, FastDVDNet, VNLB, and the state-of-the-art transformer-based VRT model. The results for VRT, FastDVDNet, and DVDNet are obtained using off-the-shelf models released by the original authors. These models were trained on the DAVIS dataset, with DVDNet’s spatial denoising modules additionally trained on the Waterloo dataset. All were trained using Gaussian noise with standard deviations sampled from the range $[0, 50]$. Unlike our proposed methods, which include both blind and non-blind variants, all of the baseline deep denoisers are non-blind and require the noise level as input at inference time. In all cases, test data is unseen.

The coring refinement network significantly enhances performance, with WienerNet (**WNet**) achieving an average improvement of **+1.8 dB** in PSNR (+0.08 SSIM) over the optimised baseline Wiener filter (*Wiener Opt.*). On average, WNet achieves results within 0.2 dB of the

Method	10 PSNR / SSIM	20 PSNR / SSIM	30 PSNR / SSIM	40 PSNR / SSIM	50 PSNR / SSIM	Time (s)	Params (M)
Noisy	28.37 / 0.6862	22.52 / 0.4431	19.22 / 0.3119	16.97 / 0.2325	15.27 / 0.1804	-	-
DVDNet [31]	35.41 / 0.9174	32.75 / 0.8724	30.58 / 0.8292	28.66 / 0.7818	26.87 / 0.7345	4k	1.3
FastDVDNet [223]	35.48 / 0.9180	32.72 / 0.8700	30.67 / 0.8277	28.84 / 0.7863	27.14 / 0.7461	70	2.5
VRT [286]	37.59 / 0.9307	34.55 / 0.8957	32.28 / 0.8661	30.22 / 0.8354	28.29 / 0.8029	2k	35.6
VNLB [32]	<u>37.58</u> / 0.9257	<u>33.82</u> / 0.8720	31.13 / 0.8178	28.90 / 0.7665	26.96 / 0.7195	27k	-
Wiener Opt.	36.26 / 0.8941	32.68 / 0.8129	30.33 / 0.7480	28.59 / 0.6950	27.06 / 0.6515	23	-
WNet	37.14 / 0.9538	33.78 / 0.9148	<u>31.96</u> / <u>0.8817</u>	30.71 / 0.8539	30.35 / 0.8528	149	0.3
WNetBlind	36.38 / <u>0.9480</u>	33.14 / 0.9071	31.92 / 0.8823	<u>30.69</u> / <u>0.8534</u>	<u>29.71</u> / <u>0.8261</u>	150	0.3
WNetBlind _{MS}	36.41 / 0.9469	33.54 / <u>0.9112</u>	31.80 / 0.8794	30.56 / 0.8503	29.59 / 0.8232	2k	0.9

Table 5.7 Final quality benchmark comparing popular denoisers with WienerNet, evaluated in terms of PSNR (dB), SSIM ([0–1]), and denoising time (seconds) across different noise levels (standard deviations). *Time* represents the total denoising time for the 10 test sequences, while ‘Params’ indicates the number of trainable parameters in each deep denoiser. *Wiener Opt.* refers to our non-blind denoiser without the coring refinement network, *WienerNet* represents our non-blind denoiser with the coring refinement network, *WienerNetBlind* is our blind denoiser, and *WienerNetBlind+MS* denotes our multiscale blind denoiser. VRT, FastDVDNet, and DVDNet are trained using the DAVIS [287] dataset, with DVDNet’s spatial denoising modules additionally trained on the Waterloo dataset. All models are trained on Gaussian noise with standard deviations sampled from the range [0, 50]. Unlike our models, all of these baseline denoisers are non-blind and require noise level information at inference time. In all cases, test data is unseen.

VRT transformer, outperforming it at higher noise levels ($\sigma = 40$ and $\sigma = 50$), while being over ten times faster on equivalent hardware.

The blind denoising implementation (WNetBlind) improves performance over the optimised baseline by **+1.1 dB** (+0.07 SSIM) on average, achieving results within 0.4 dB of the non-blind denoiser. Despite being blind, WNetBlind outperforms the VRT transformer at higher noise levels.

Finally, the multi-scale blind denoiser (WNetBlind_{MS}), discussed in Section 5.4.5, demonstrates enhanced performance in lower noise bands ($\sigma = 10$ and $\sigma = 20$). This suggests that multi-scale information can improve denoising performance, particularly for textures requiring fine spatial localisation. Figures 5.6c and 5.7d provide visual comparisons of denoised results for WienerNet and other denoisers, with additional examples shown in Figures 5.8, 5.9, and 5.10.

5.4.7 Additional Analysis: Motion Compensation

Lastly, we evaluate motion compensation as a preprocessing step for denoising using both trained and untrained (WNetBlind) filters, with motion estimation performed via the Deep-

	None	Deepflow	Raft	
Sigma	PSNR / SSIM	PSNR / SSIM	PSNR / SSIM	
10	35.67 / 0.9106	34.87 / 0.9071	34.94 / 0.9074	Wiener Opt.
20	31.84 / 0.8441	31.51 / 0.8480	31.68 / 0.8502	
30	29.74 / 0.7925	29.49 / 0.8016	29.53 / 0.8022	
40	27.97 / 0.7466	27.81 / 0.7591	27.75 / 0.7567	
50	26.42 / 0.7063	26.30 / 0.7196	26.17 / 0.7137	
10	36.38 / 0.9480	36.38 / 0.9337	36.35 / 0.9338	WNetBlind
20	33.14 / 0.9071	33.74 / 0.8914	33.62 / 0.8903	
30	31.92 / 0.8823	31.99 / 0.8593	31.73 / 0.8556	
40	30.69 / 0.8534	30.65 / 0.8310	30.24 / 0.8232	
50	29.71 / 0.8261	29.51 / 0.8044	28.96 / 0.7915	

Table 5.8 Ablation study results evaluating the impact of motion compensation as a preprocessing step on video denoising performance, before and after training the Wiener Refinement network. Experiments are conducted across all datasets with noise levels $\sigma = [10-50]$. *WNetBlind* refers to our blind denoiser.

Flow [279] and RAFT [281] optical flow algorithms. The assessment is conducted for the five-frame case, where motion vectors are calculated between the outer frames and the centre frame. We present these results in Table 5.8.

In the untrained scenario, neither DeepFlow nor RAFT improves denoising performance in terms of PSNR, though SSIM is enhanced at all noise levels except for $\sigma = 10$. This outcome may be due to artefacts introduced by occlusions in the motion-compensated frames.

For the trained network, DeepFlow matches the non-motion-compensated performance at $\sigma = 10$ and exceeds it at $\sigma = 20$ and $\sigma = 30$. However, no gains in SSIM are observed, and PSNR improvements are absent at higher noise levels ($\sigma = 40$ and $\sigma = 50$). These results suggest that the trained denoisers may have adapted to handle occlusions caused by motion compensation algorithms. Further performance gains could be achieved by discarding or ignoring frames with occlusion levels that exceed a defined threshold.

5.5 Discussion

Our results demonstrate that a hybrid denoiser strikes a strong balance between model size, denoising speed and output quality, with our blind model outperforming state-of-the-art non-blind transformer models at high noise. The larger data volume in video denoising favours non-deep learning approaches in terms of memory efficiency. By implementing a



Fig. 5.6 Sample output frames at input noise strength of $\sigma = 20$, taken from the India benchmark scene.



Fig. 5.7 Sample output frames at input noise strength of $\sigma = 30$, taken from the Footbridge benchmark scene.

Bayesian filter, we begin the process with domain knowledge via the Wiener prior, and its performance can be enhanced in a straightforward process by carefully implementing small ancillary networks.

As temporal frames introduce redundancies through duplicate realisations of blocks along the temporal axis, the tuning of spatial hyperparameters becomes less critical. For instance, a third overlap suffices to achieve near-maximal denoising performance in video denoising, while a quarter overlap is required in the image denoising case. These redundancies present opportunities for further optimisation. For example, similar results might be achieved by processing every other frame and using all output frames to interpolate the skipped frame.

Although the current implementation is faster than all tested methods except FastDVDNet, significant room remains for optimising denoising speed. While our denoiser was designed to have a small footprint, denoising speed was not a priority during most of the development. In the five-frame output scenario, the filter could be accelerated by calculating the inverse FFT only for the centre frame. Additionally, the current implementation rebuilds windowing tensors at every pass when these maps can be precomputed on initialisation of the filter. Removing boolean condition checks introduced during the proof-of-concept phase, such as determining which window type to use, would further streamline the process.

The Wiener coring network and noise prediction network presented here are just one set of results among many architectures we evaluated. The WienerNet started as a single-stage network, but a two-stage approach was found to provide a better balance between parameter count and output quality. We initially experimented with larger numbers of feature maps before settling on 40 channels per layer to minimise denoising time. We also tried using 3D convolutions across the temporal axis (with a colour channel kernel) rather than across the colour axis (with a temporal axis kernel), but this yielded inferior results.

We also explored early-stage development of a single-model, multi-scale Wiener denoising approach, which involved creating the filter for the largest window size and upsampling smaller input windows accordingly. Due to publishing timelines, we could not fully invest in this solution. Although the multiscale implementation in our results is more straightforward, it suggests that greater denoising quality may be achieved by refining this process, possibly through adaptive block size selection based on content and noise levels or by employing a custom weighting scheme for multi-pass denoising.

Our denoiser is capable of processing full 1080p sequences, unlike transformer models, which consumed over 20 GB of memory on the testing machine. We also developed an extra high memory model to handle higher-resolution images. During inference, peak memory

usage occurs when the stacked image patches pass through the two-stage 3D CNN. To mitigate memory use, we created memory-contiguous views by dividing the patches into smaller groups along the batch dimension. Each batch is then processed individually without moving data off the GPU. This method was originally implemented for larger versions of WienerNet but could still prove useful for untested, higher-resolution sequences.

As our Wiener filter is frequency-based, it is more resilient to misalignment caused by inter-frame motion. Our results show that motion compensation only improves output quality in certain cases, while the computational cost of optical flow makes it impractical for inclusion in the final denoiser. Nevertheless, specific test sequences do show substantial quality gains from motion compensation. This aligns with observations in the DVDNet paper, where some sequences degrade with motion compensation while others significantly improve. Kokaram [27] suggested measuring the number of occluded pixels in the motion-compensated frames and excluding frames that exceed a certain threshold. Adopting a similar approach may be crucial for further enhancing the Wiener filter.

Lastly, while our results focus largely on PSNR, a classic distortion metric, it should be kept in mind that perceptual metrics are becoming increasingly important in evaluating denoising performance. Although we have also included SSIM in our analysis, which provides some insight into structural similarity, more sophisticated perceptual metrics such as LPIPS [288] and VMAF [289] could offer a deeper understanding of the perceptual quality of the outputs. Revisiting our results with these perceptual metrics could provide additional insights and help further refine the balance between denoising quality and computational efficiency in future work.

5.6 Conclusions

In this chapter, we have presented a blind and non-blind hybrid 4D Wiener denoising filter, which leverages neural networks to achieve maximum output quality while preserving a compact model size and rapid denoising speeds. Our denoiser is far smaller in terms of parameters than all tested networks, and faster than the state-of-the-art transformer models, while remaining within 0.2 dB of the overall average quality and surpassing all tested models in SSIM.

Beyond denoising, the sophisticated priors learned by neural networks offer significant potential for solving a variety of other tasks, particularly in the context of linear inverse problems. In the following chapter, we will explore how denoisers, as implicit models of

natural image priors, can be extended to applications such as inpainting, deblurring, and super-resolution, demonstrating their versatility and broader utility across different domains.



Fig. 5.8 Larger denoising samples for FastDVDNet and WienerNet. **Top:** Noisy input image at $\sigma = 40$ (17.1 dB). **Middle:** FastDVDNet Filtered image (28.96 dB). **Bottom:** WienerNet Filtered image (31.54 dB).

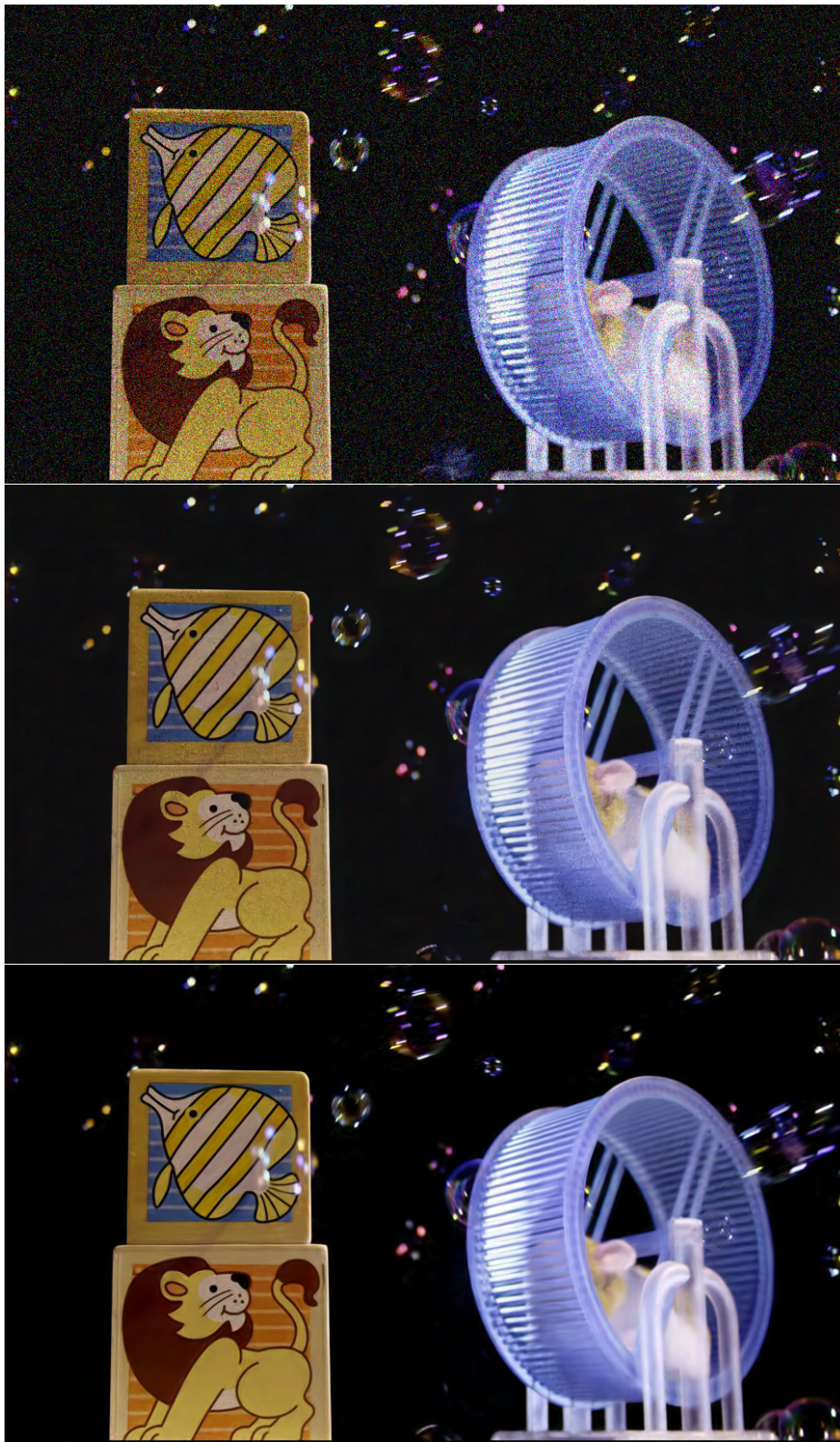


Fig. 5.9 Larger denoising samples for FastDVDNet and WienerNet. **Top:** Noisy input image at $\sigma = 40$ (17.73 dB). **Middle:** FastDVDNet Filtered image (26.21 dB). **Bottom:** WienerNet Filtered image (32.75 dB).



Fig. 5.10 Larger denoising samples for FastDVDNet and WienerNet. **Top:** Noisy input image at $\sigma = 40$ (16.64 dB). **Middle:** FastDVDNet Filtered image (26.60 dB). **Bottom:** WienerNet Filtered image 28.39 dB).

Chapter 6

Fine-Tuning the CNN Image Prior for Linear Task Solving

Since their introduction, deep neural networks (DNNs) have demonstrated a remarkable ability to surpass the denoising performance of classical Bayesian and other optimal estimators, achieving this by implicitly modelling complex image priors $p(x)$ that were previously unattainable. An intuitive example of this capability is the networks' ability to reconstruct lost image details that are not replicated elsewhere in the image, reflecting their capacity to capture sophisticated structural and textural information. By explicitly harnessing this natural image prior $p(x)$, the potential applications of these denoisers can extend well beyond noise removal to a range of other tasks. This prior was first indirectly exposed through the "Plug-and-Play" priors (PnP) approach (see section below), and more recently, the connection between the denoiser and the prior $p(x)$ has been formalised through the works of Kadkhodaie and Simoncelli in *Solving Linear Inverse Problems Using the Prior Implicit in a Denoiser* [45]. In this work, the authors explicitly link the residual image of a blind AWGN denoiser, $\hat{x}(y) - y$, with the gradient of the log-likelihood of the image:

$$\hat{x}(y) - y = \frac{1}{\sigma^2} \nabla_y \log p(y), \quad (6.1)$$

a result, made possible through the rediscovery of earlier proofs by Tweedie [290] and Miyasawa [290]. Although this connection was not explicitly made at the time, it also underpins the functionality of contemporary diffusion models [238]. While both PnP methods and Simoncelli's solvers present iterative approaches for solving linear inverse problems, the former relies on using denoisers as a regulariser within an optimisation framework, whereas the latter leverages Equation 6.1, offering a more streamlined approach. Given

these considerations, we proceed with the lightweight method presented by Kadkhodaie and Simoncelli.

In this chapter, we extend the linear inverse solver framework of Kadkhodaie and Simoncelli by fine-tuning the denoiser at the core of the iterative solver across multiple tasks simultaneously, including super-resolution, inpainting, and demosaicing. This joint training encourages the model to generalise more effectively and function as a versatile image restoration module.

We also assess how iterative training affects the denoiser’s performance in its original role. Unlike single-pass denoising, early artefacts in iterative pipelines may accumulate and amplify, leading to greater penalties during training. By evaluating the fine-tuned model, we investigate whether this process yields smoother, artefact-free outputs. In doing so, we examine both the denoiser’s task generalisation and its improved fidelity in image restoration.

6.1 Solving Inverse Problems Using the Denoisers Implicit Prior

In this section, we explore the progression of approaches that initially leveraged denoisers indirectly and later directly utilised the implicit natural image prior they model for solving inverse problems.

6.1.1 Plug and Play Priors

The use of denoisers as priors for solving inverse problems originates with the *Plug-and-Play Prior* (PnP) framework, introduced by Venkatakrishnan et al. [248] in 2013. Traditionally, inverse problems are formulated as the minimisation of an objective function comprising a data fidelity term and a regularisation term that encodes prior knowledge, such as smoothness. In the original PnP formulation, the MAP estimate of the solution x is written as:

$$\hat{x} = \arg \min_x \{l(y;x) + \beta s(x)\},$$

where $l(y;x)$ is the data fidelity term, $s(x)$ is the prior, and β balances their relative contributions. To make this optimisation tractable, PnP adopts a variable splitting strategy that decouples the two terms by introducing an auxiliary variable v , leading to the augmented Lagrangian [159]:

$$L_\lambda(x, v, u) = l(y;x) + \beta s(v) + \frac{\lambda}{2} \|x - v + u\|_2^2,$$

where u is the scaled dual variable [291] and λ is the penalty parameter. This reformulation makes the optimisation more tractable by decoupling the data fidelity and prior terms into separate subproblems. Instead of solving a single, complex objective involving both terms simultaneously, the augmented Lagrangian enables efficient alternating updates: the data term is handled in a quadratic minimisation, while the prior term is replaced by a denoiser, avoiding the need to explicitly model or differentiate the prior. This objective is then minimised using the Alternating Direction Method of Multipliers (ADMM), by iteratively updating x , v , and u .

Critically, the proximal step for v is replaced with a denoising operation:

$$\begin{aligned} \text{Update } x : \quad & x^{k+1} = \arg \min_x \left\{ l(y; x) + \frac{\lambda}{2} \|x - v^k + u^k\|_2^2 \right\}, \\ \text{Update } v : \quad & v^{k+1} = \text{Denoiser}(x^{k+1} + u^k, \sigma), \\ \text{Update } u : \quad & u^{k+1} = u^k + (x^{k+1} - v^{k+1}). \end{aligned}$$

This formulation enables any state-of-the-art denoiser to be “plugged in” as a prior, effectively replacing the explicit regulariser $s(x)$ with an implicit prior defined by the denoising algorithm. At the time, these denoisers were non-learned (e.g., BM3D), but the framework laid the foundation for incorporating deep learning-based denoisers in subsequent work.

While the original paper demonstrates the concept by applying the optimisation framework to X-ray and MRI reconstruction, it was soon extended to other applications. For instance, PnP was extended to image super-resolution by Brifman and Milano in 2016 [249]. In 2017, Romano and Milanfar proposed *Regularisation by Denoising (RED)* [292], a PnP modification that simplifies the regularisation to use the denoising prior without ADMM. Later, PnP was adapted for deep neural networks, as demonstrated by Zhang et al. in *Learning Deep CNN Denoiser Prior for Image Restoration* [72] (2017) and further by Zhang, Timofte et al. in *Plug-and-Play Image Restoration With Deep Denoiser Prior* [250] (2021), showcasing the effectiveness of a U-Net denoiser (DRUNet) in tasks like JPEG deblocking, deblurring, super-resolution, and demosaicing.

The PnP approach, though versatile, relies on the indirect use of priors, incorporating them through the update steps of optimisation framework. In 2021, Kadkhodaie and Simoncelli demonstrated that the denoiser prior may be used much more directly, offering a framework where the prior is tightly integrated into the solution process itself. This direct utilisation of the prior enhances both the interpretability and effectiveness of the reconstruction for inverse problems. In the next section, we derive the Tweedie-Miyasawa result which makes their interpretation possible.

6.1.2 The Tweedie-Miyasawa Denoiser-Residual Relationship

We note that while the identity is credited to Miyasawa by Kadkhodaie and Simoncelli, the general formulation can be traced back to Tweedie’s Formula (1956), which was first published by Robbins [290] after he received the formula by letter from Maurice Tweedie. The formula was later rediscovered and republished by Efron [52] in 2011, contributing to its recent popularisation.

To harness denoisers for image generation and inverse problem solving, we aim to establish a connection between the denoiser residual and the natural image prior $p(x)$. This connection is made explicit through the *Tweedie-Miyasawa identity*, which we derive below. The identity links the denoiser to the gradient of the *noisy image prior* $p(y)$, where $y = x + n$. Under the assumption of additive Gaussian noise, this noisy prior $p(y)$ approximates the true image prior $p(x)$ as the noise variance tends to zero, thereby enabling the use of denoisers as learned priors in probabilistic image restoration.

We begin with the observation model for AWGN:

$$y = x + n,$$

where $x \in \mathbb{R}^N$ is the original image with N pixels, y is the observed noisy image, and n is additive Gaussian noise with $n \sim \mathcal{N}(0, \sigma^2 I_N)$, where I_N is the identity matrix indicating that the noise is independent and identically distributed across all N pixels. By marginalising over x , the prior predictive density of y is expressed as:

$$p(y) = \int p(y|x) p(x) dx \quad (6.2)$$

where the likelihood $p(y|x)$ is equivalent to the multivariate Gaussian:

$$p(y|x) = \frac{1}{(2\pi\sigma^2)^{N/2}} \exp\left(-\frac{\|y-x\|^2}{2\sigma^2}\right). \quad (6.3)$$

To understand how the observed image y relates to the underlying image distribution, we take the gradient of the marginal likelihood $p(y)$ with respect to y , which for a Gaussian function, gives us:

$$\nabla_y p(y|x) = \frac{x-y}{\sigma^2} p(y|x). \quad (6.4)$$

Applying this to Equation 6.2, we obtain:

$$\nabla_y p(y) = \frac{1}{\sigma^2} \int (x - y) p(x) p(y|x) dx. \quad (6.5)$$

Now, dividing both sides by $p(y)$:

$$\nabla_y \log p(y) = \frac{1}{\sigma^2} \int (x - y) \frac{p(x) p(y|x)}{p(y)} dx. \quad (6.6)$$

Using Bayes' theorem, we substitute to obtain:

$$\nabla_y \log p(y) = \frac{1}{\sigma^2} \int (x - y) p(x|y) dx. \quad (6.7)$$

This is the expected residual between the posterior estimate and the observed noisy image, taken over the posterior distribution $p(x|y)$. Recognising that the posterior mean $\mathbb{E}[x | y]$ is the MMSE-optimal estimate of x , we express this as:

$$\nabla_y \log p(y) = \frac{1}{\sigma^2} (\mathbb{E}[x | y] - y). \quad (6.8)$$

Substituting the MMSE denoiser function $\hat{x} := \mathbb{E}[x | y]$, we obtain the **Tweedie-Miyasawa identity**, which shows that the denoising residual equals the gradient of the log probability:

$$\sigma^2 \nabla_y \log p(y) = \hat{x} - y. \quad (6.9)$$

This result is fundamental to generative denoising models, as it provides a way to access the neural network image prior $p(x)$ via the denoising residual $\hat{x} - y$. Equation 6.9 demonstrates that denoising can be expressed as gradient ascent, progressively removing noise from y by following the probability gradient.

6.1.3 Simoncelli and Kadkhodaie's Linear Inverse Solver

Building on Miyasawa's result, Simoncelli and Kadkhodaie introduced two algorithms that demonstrate the application of denoiser priors. First, they proposed a framework to generate realistic images from pure noise using a denoiser. Second, they extended this method to address linear inverse problems, such as super-resolution and inpainting.

Generation of Real Images. With the Tweedie-Miyasawa result established, the authors introduce a helpful analogy: the image prior model can be thought of as occupying a low-dimensional subspace or "manifold" within the high-dimensional pixel space $\mathbb{R}^N$. Outside this

manifold, the prior probability $p(x)$ is low, resulting in noisy or distorted images. Conversely, within the manifold, $p(x)$ is high and varies slowly, corresponding to the smooth, structured characteristics of natural images. By iteratively denoising a noisy image and advancing along the gradient that maximises $p(x)$ (the log probability gradient), the algorithm moves closer to the manifold of natural images. Thus, by beginning with a noisy image and taking small steps guided by this gradient, the method synthesises realistic images.

Starting from a random noise image y_0 , the authors demonstrate that a "high-probability" image can be approached by following the gradient of $\log p(y)$, as shown in Equation 6.9. This gradient is approximated using the residual produced by their choice of denoiser. In each iteration, a small step is taken in the specified direction, and the noise amplitude is progressively reduced. Between denoising steps, the step size is also adjusted as a function of the current noise strength to ensure convergence to a point on the real-image manifold. To avoid getting stuck in local maxima, controlled noise is reintroduced at each step, proportional to the current noise level. The generated noisy image y_t at iteration t is updated as follows:

$$y_t \leftarrow y_{t-1} + h_t \sigma^2 \nabla_y \log p(y) + \gamma_t z_t, \quad (6.10)$$

where h_t controls the step size along the gradient of $p(x)$, and the amount of noise reintroduced at each iteration $z_t \sim \mathcal{N}(0, \sigma^2 = 1)$ is controlled by γ_t . The scheduling for h_t and γ_t is controlled by the hyper-parameters σ_0 , σ_L , h_0 , and β (see original paper [45]).

The authors use the BF-CNN [198] denoising network, a bias-free variant of DnCNN. As the name suggests, this network removes the bias terms from all convolutions as well as the batch normalisation layers. By removing the bias terms, it has been shown that networks achieve improved generalisation across a broader range of noise levels, which is essential given that the framework starts with a heavily noisy image and gradually reduces the noise. The model used by the authors is trained using AWGN, and the standard deviation of the noise in each image is uniformly sampled as $\sigma \sim U(0, 100)$.

Universal Inverse Solver. The method can be extended to solve inverse linear problems by placing linear constraints on a noisy image through a low-rank measurement matrix $x^c = M^T x$, where x^c is the vector of measurements. For example, in super-resolution, M^T represents a downsampling operation. M^T can also be viewed as the degradation model, while M is the observation model. Without loss of generality, Kadkhodaie and Simoncelli assume that M and M^T are semi-orthogonal, which implies that M is the pseudo-inverse of M^T and that MM^T can be used to project an image onto the measurement subspace. With

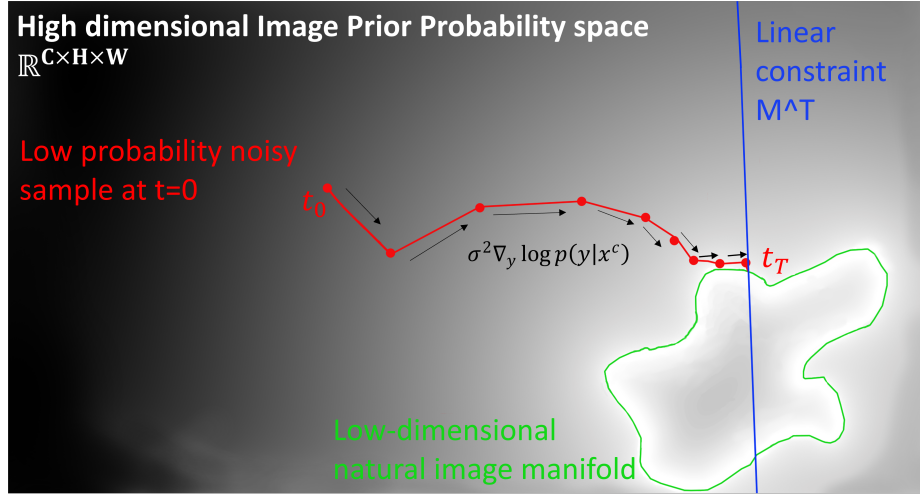


Fig. 6.1 Illustration of solving a linear inverse problem in image probability space $\mathbb{R}^{C \times H \times W}$. Starting from a low-probability sample y_0 (red point, left), the solution is iteratively refined via gradient ascent on the log posterior $\nabla_y \log p(y | x^c)$, guided by the denoiser’s residual. The red trajectory represents the evolution of the sample toward higher-probability regions, constrained to remain within the linear subspace defined by the data fidelity constraint $M^T y = x^c$ (blue line). The solution converges near the natural image manifold (outlined in green), which captures high-probability images under the learned prior.

these constraints, the gradient of the conditional density can be derived as:

$$\sigma^2 \nabla_y \log p(y|x^c) = (I - MM^T)(\hat{x} - y) - M(x^c - M^T y), \quad (6.11)$$

where I is the identity matrix. This conditional gradient can then be used within a gradient-ascent iterative scheme shown in equation 6.10, where the constrained gradient $\sigma^2 \nabla_y \log p(y|x^c)$ replaces the unconstrained gradient found using the residual alone. For convenience, the overall algorithm is reported in Algorithm 5, and an illustration of solving the linear inverse problem in the probability space is shown in Figure 6.1. Here, we show how a low probability image sample converges towards the low-dimensional subspace of real images through constrained sampling of the image prior.

In Figure 6.2 we provide a demonstration of the solver output after each iteration. In this case, we restore an image degraded by a partially sampled random basis function (see measurement matrices section). The first image is the degraded image y_0 projected onto the image space.

Algorithm 5 Universal Inverse Solver as defined in [45] for sampling from $p(x|M^T x = x^c)$, based on the denoiser residual $\hat{x} - y$. Note: e is an image of ones, σ_0 and σ_L are the initial and final noise standard deviations, h_0 is the initial step size, β controls the noise scheduling, M is the observation model and x^c is the degraded image.

```

1: Parameters:  $\sigma_0, \sigma_L, h_0, \beta, M, x^c$ 
2: Initialization:  $t \leftarrow 1$ ;
3: Draw  $y_0 \sim \mathcal{N}(0.5(I - MM^T)e + Mx^c, \sigma_0^2 I)$ , with  $e = [1, \dots, 1]^T$ 
4: while  $\sigma_{t-1} \leq \sigma_L$  do
5:    $h_t \leftarrow \frac{h_0 t}{1 + h_0(t-1)}$ 
6:    $d_t \leftarrow (I - MM^T)(\hat{x}_{t-1} - y_{t-1}) + M(x^c - M^T y_{t-1})$ 
7:    $\gamma_t^2 \leftarrow ((1 - \beta h_t)^2 - (1 - h_t)^2) \frac{\|d_t\|^2}{N}$ 
8:    $y_t \leftarrow y_{t-1} + h_t d_t + \gamma_t z_t$  with  $z_t \sim \mathcal{N}(0, I)$ 
9:    $t \leftarrow t + 1$ 

```

6.2 Method: Fine-Tuning Denoisers For Solving Inverse Problems

Kadkhodaie and Simoncelli’s method hinges on using a pre-trained blind denoiser to approximate the log prior’s gradient within their iterative solver. A key strength of this method is its adaptability, enabling the use of any compatible denoiser across various inverse problems. Recognising that the denoiser’s characteristics will significantly affect the iterative method’s outcome, a crucial question arises: can the denoiser be specifically optimised to enhance performance when applied in this manner to solve inverse problems?

6.2.1 Differentiable Training Platform for the Denoiser

To this end, we introduce a multi-task training platform designed to fine-tune a denoiser by integrating it directly into the iterative solver and training it on multiple linear inverse problem tasks (see Figure 6.3). A key aspect of our approach is treating the complete reconstruction as a differentiable computation graph. In training, a degraded input is fed through multiple iterations of the solver, with each step utilising the denoiser, until the final reconstruction is achieved. An image quality loss (L_1) is calculated on this final output and propagated backwards through all intermediate stages to update the denoiser’s parameters. Consequently, the network learns from a diverse distribution of intermediate reconstructions, enhancing its ability to generalise to the data encountered during iterative inverse problem solving.

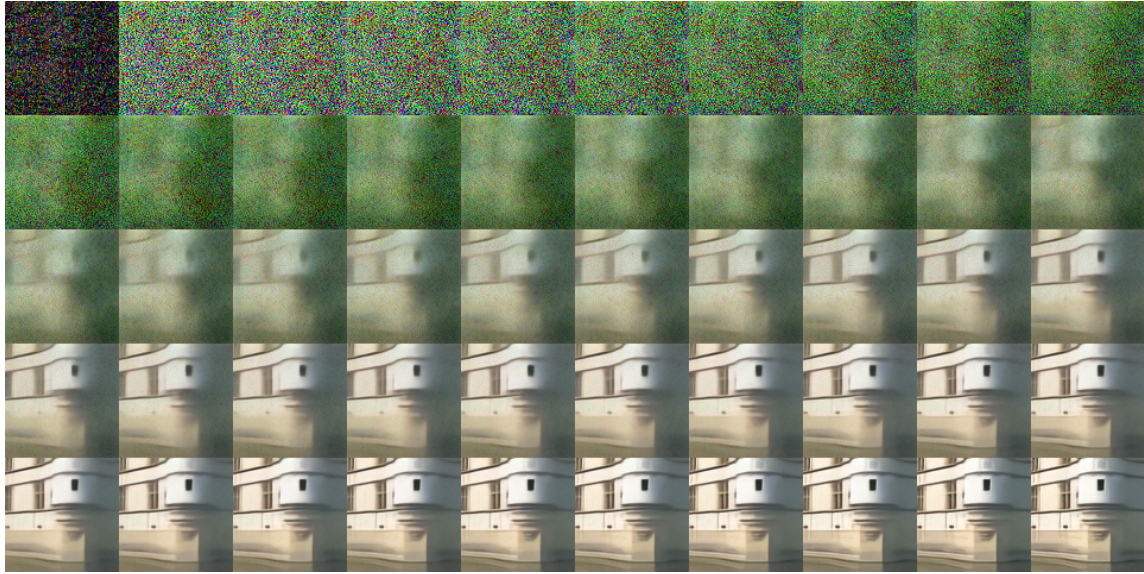


Fig. 6.2 Demonstration of the inverse solver reconstructing an image from an undersampled random basis projection, retaining only 10% of the original basis. The solver is parametrised to converge in 50 iterations. The first image (top left) in the sequence shows the initial degraded image y_0 , obtained by projecting the compressed measurement $y = M^\top x$ back into the image space: $y_0 = M(M^\top x)$. For details on the random basis formulation, see Section 6.3, *Random Basis Projection*.

6.2.2 Fixing the Number of Iterations

To enable training, we need to modify the original framework. In the original formulation, iterations proceed until the solution is deemed sufficiently smooth, specifically, when the standard deviation of the image residual falls below a predefined threshold. While appropriate for inference, this convergence-based stopping criterion is impractical for training, as it requires hundreds or thousands of iterations and results in inconsistent iteration counts across different images in the batch. To address this, we remove the smoothness-based termination condition and instead fix the number of iterations per training step. This not only standardises the training loop, but also enables backpropagation through a consistent unrolled graph. The selection of an appropriate iteration count and associated hyperparameters is addressed in Section 6.3.4.

6.2.3 Generalising the Denoiser via Multi-Task Training

To promote generalisation, we aim to fine-tune the denoiser not for a single inverse problem, but for a broader class of linear degradation processes. Rather than optimising it on a fixed task or constraint, we expose the denoiser to a variety of forward models during training. By applying the same network across different iterative reconstruction pipelines, the denoiser

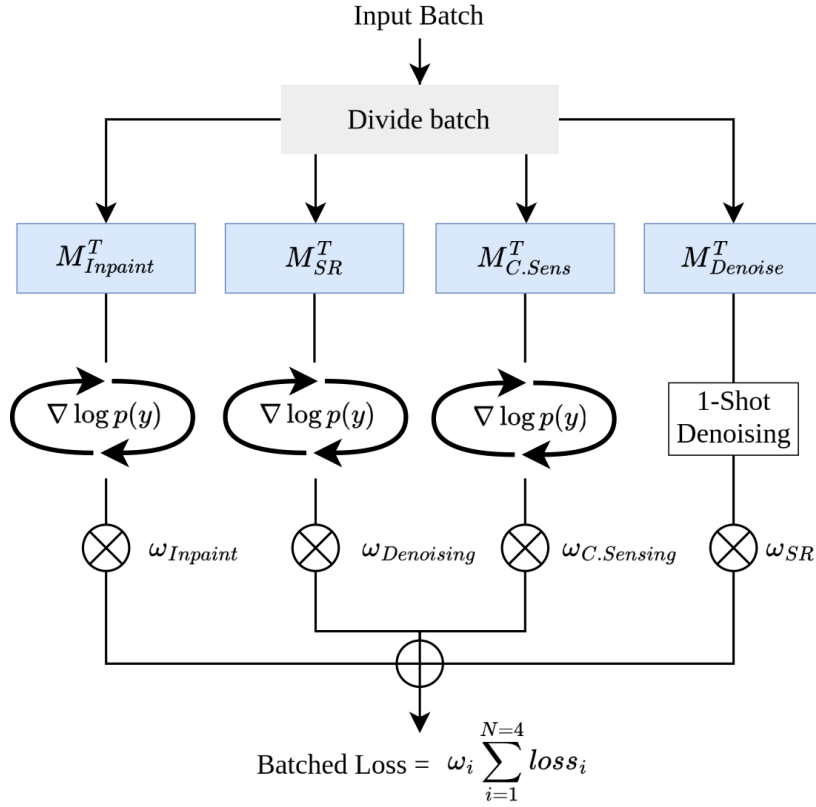


Fig. 6.3 Illustration of our multi-task training framework. An input batch is divided into subsets for each inverse problem (inpainting, super-resolution, compressed sensing, and denoising). Each subset is passed through a task-specific forward operator M^T to produce the measurement vectors x_i^c , then, output images are iteratively reconstructed using a shared denoiser model that produces gradient estimates $\nabla \log p(y)$. The circular arrows represent the iterative refinement process. Each task output is used to compute a reconstruction loss, weighted by task-specific coefficients ω_i , and summed to form the final training loss.

learns to produce gradient estimates that remain effective across a range of input conditions. This approach encourages the model to internalise a more flexible and broadly applicable image prior, rather than one tailored to a single inverse mapping. The specific set of tasks and their configuration are detailed in Section 6.4.2.

Figure 6.3 presents a generalised illustration of this training process. For each training step, a batch of B images is divided across N tasks. Each mini-batch is assigned a task and processed using a corresponding task-specific measurement operator M_{task}^T . The same denoiser is then applied iteratively within each task's reconstruction pipeline to solve the constrained gradient ascent problem. This produces N task-specific losses, which are scaled by corresponding weighting factors (see Section 6.4.3) and summed to produce the overall training loss used to update the denoiser parameters.

6.3 Experimental Setup

With the training framework outlined, we now establish the training and test dataset, the deep learning architecture used for fine-tuning during experimentation, as well as the hyperparameters that were tuned according to the number of iterations used.

6.3.1 Inverse Problem Tasks

In our experiments, we include the tasks from the original paper. We additionally introduce demosaicing as an additional task.

The measurement matrices M^T project the clean image x onto a lower-dimensional subspace to become the constrained or degraded image $x^c \in \mathbb{R}^K$:

$$x^c = M^T x, \quad (6.12)$$

In all of our tasks, M is set to be semi-orthogonal; thus the back-projection to image space can be performed using M :

$$\hat{x} = Mx^c, \quad (6.13)$$

where $\hat{x} \in \mathbb{R}^{HW}$ is the reconstructed image vector.

Super-Resolution. Here, M^T is a downsampling operator, represented as a convolution kernel K , whose size and stride depend on the desired reduction. For example, in 2×2 super-resolution, the ground truth image x (of size $H \times W \times C$) is downsampled with a 2×2 convolution kernel and a stride of 2, effectively averaging pixels over non-overlapping 2×2 regions. The kernel is given by:

$$K = \frac{1}{4} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

and the downsampled image x^c is given by $x^c = M^T x = x * K$, where $*$ denotes the convolution operation with a stride of 2. We note that this method is suboptimal in its frequency response (characterised as a Sinc function), causing ringing due to side lobe amplification of high frequencies and aliasing due to insufficient filtering of high frequencies. As such, we will also explore the use of frequency-optimal downsampling filters such as bi-orthogonal methods (see section 6.4.5).

Inpainting. For inpainting, the measurement matrix M^T is a binary diagonal mask matrix where the elements on the diagonal are set to zero or one, depending on whether that pixel falls within the missing region or not. During training, this region is randomly selected. Each diagonal entry in M^T is defined as:

$$M_{ii}^T = \begin{cases} 1 & \text{if pixel } i \text{ is retained,} \\ 0 & \text{if pixel } i \text{ is within the } W \times W \text{ masked region.} \end{cases}$$

This matrix M^T selectively removes a contiguous block of pixels by element-wise multiplication with the original image, providing only the observed pixels outside the $W \times W$ region and requiring the model to reconstruct the missing block. For an image of size $H \times W \times C$, the matrix M^T is of size $(HWC) \times (HWC)$, though in practice it is applied as a binary mask tensor of shape $H \times W \times C$ via element-wise multiplication.

Compressed Sensing. This task aims to recover signals from a small number of random linear measurements. In this scenario, we will consider that 10% of the pixels are randomly preserved. M^T is also a diagonal binary mask matrix, similar to inpainting, but with randomly selected pixels rather than a contiguous block. Each diagonal entry in M^T is set as follows:

$$M_{ii}^T = \begin{cases} 1 & \text{if pixel } i \text{ is selected by the random sampling pattern,} \\ 0 & \text{otherwise.} \end{cases}$$

This random sampling matrix M performs an element-wise multiplication with the original image, leaving only a sparse set of observed pixels. The model uses these sparse measurements to reconstruct the missing data.

Demosaicing. For demosaicing, M^T follows the Bayer Colour Filter Array (CFA) pattern, where each 2×2 block contains one red, two green, and one blue pixel. The diagonal entries of M^T are, again, binary values indicating the availability of the colour channels.

For each pixel index i and each colour channel $k \in \{1, 2, 3\}$, the diagonal entry in M is defined as:

$$M_{3i+k, 3i+k}^T = \begin{cases} 1 & \text{if primary } k \text{ is available for pixel } i \text{ according to the Bayer pattern,} \\ 0 & \text{otherwise.} \end{cases}$$

This matrix M selectively retains only the available colour channels for each pixel, following the Bayer pattern, resulting in a sparse observed vector. The colour value for primary k at pixel i is then given by $x_i^c = x_{3i+k}$.

Frequency Super-Resolution In this process, the 2D FFT of x is computed, and a mask W is applied to zero out a fraction of the high-frequency coefficients (e.g., 20%). Following the masking step, an inverse FFT is performed to obtain a low-resolution spatial-domain approximation: $x^c = \text{IFFT}(W \cdot \text{FFT}(x))$, where $\text{FFT}(x)$ and IFFT represent the Fourier transform and its inverse, respectively.

Random Basis Projection. In the random basis projection task, the image is compressed by projecting it onto a lower-dimensional, randomly generated subspace. This is accomplished using a random, semi-orthogonal measurement matrix $M^\top$, whose columns form an orthonormal basis spanning the target subspace.

To perform the projection, the input image of size $H \times W$ is first reshaped into a vector $x \in \mathbb{R}^{HW}$. The measurement matrix $M \in \mathbb{R}^{HW \times K}$ is constructed such that its columns form a semi-orthogonal basis—i.e., they are orthonormal but do not span the full space of $\mathbb{R}^{HW}$. The dimensionality reduction is controlled by a compression ratio $p = \frac{K}{HW}$, which is fixed at 20% in our experiments. The forward measurement matrix $M^\top \in \mathbb{R}^{K \times HW}$ is expressed as:

$$M^\top = \begin{bmatrix} \uparrow & \uparrow & & \uparrow \\ m_1 & m_2 & \cdots & m_K \\ \downarrow & \downarrow & & \downarrow \end{bmatrix},$$

where each row $m_i^\top$ corresponds to one of the K orthonormal directions. For more information on this task, see Kadkhodaie’s implementation [45].

6.3.2 Training/Testing Set

The training dataset consists of 3,400 uncropped images selected from DIV2K [196] and LSDIR [293].

Our test set for all of our experiments will be the 100 images from the BSD100 [294] dataset.

Configuration	Iterations	β	h_0	σ_0
Short	25	0.4	0.10	0.6
Medium	50	0.2	0.09	0.8
Long	100	0.06	0.5	0.08

Table 6.1 Hyperparameters for Short and Long task configurations. The number of iterations per task is fixed to enable batched training, with other parameters (β , h_0 , and σ_0) adjusted to ensure convergence.

6.3.3 Choice of Baseline Denoiser

Our choice of baseline denoiser follows, as in Kadkhodaie and Simoncelli [45], a BF-CNN architecture. This is essentially a modification of DnCNN [198] that removes biases from convolutional and batch normalisation layers, enabling improved generalisation across a wide range of input noise levels.

6.3.4 Hyperparameter Tuning for Fixed Number of Iterations

To reduce the number of iterations, the smoothness criterion is replaced with a fixed iteration count based on task difficulty. By constraining the algorithm to a set number of iterations, hyperparameters must be adjusted to ensure convergence within this limit. The key hyperparameters are β , which controls reintroduced noise; h_0 , the initial step size; and σ_0 , the starting noise level. For most tasks, starting with a heavily noised version of the constrained image (e.g., $\sigma_0 = 0.8$) is sufficient to achieve smooth results, avoiding the need for the higher noise levels ($\sigma_0 = 1$) typically used in the original framework.

We observed that different tasks converge at varying rates, which motivated the creation of separate iteration configurations: *short*, *medium*, and *long*. Super-resolution and frequency super-resolution are assigned the *short* configuration, while inpainting, compressed sensing, demosaicing, and random basis reconstruction are assigned the *medium* configuration, as summarised in Table 6.1. Additionally, a *long* configuration is introduced to evaluate the potential benefits of allowing the solver to run for an extended number of iterations for all tasks.

6.3.5 Training Details

During training, input images are cropped to 96×96 and augmented with random flips and rotations. Each task is supervised using the mean squared error (MSE) loss, and optimisation is performed with the AdamW optimiser over 500 epochs. A cosine annealing learning

rate scheduler with warm restarts is employed: the learning rate is initially decreased from 10^{-3} to zero over the first 100 epochs, then reset and decayed to zero every subsequent 50 epochs. To accommodate the high memory demands of unrolled backpropagation through the iterative solver, the batch size is limited to 16.

6.4 Experiments

In this section, we describe the experiments conducted to evaluate the feasibility and effectiveness of fine-tuning a denoiser within an iterative framework for solving linear inverse problems. We begin by assessing whether fine-tuning improves performance on a single task compared to a fixed, pre-trained model. We then investigate the denoiser’s ability to generalise across multiple inverse problems when trained jointly on several tasks. To address the challenge of balancing task-specific objectives during multi-task training, we explore different loss weighting strategies. Finally, we examine how fixing the number of iterations—rather than relying on convergence—affects both the quality and efficiency of the resulting reconstructions.

6.4.1 Exploratory Single-Task Experiments

In these experiments, we evaluate the performance of the iterative solver in a controlled setting where the denoiser is fine-tuned for a single linear inverse task: super-resolution ($\times 4$). This setup allows us to investigate how modifications to the original algorithm’s hyperparameters affect convergence speed, and to assess the benefits of task-specific fine-tuning.

Default Configuration. We first examine the default settings of the framework proposed by Kadkhodaie and Simoncelli [45]. In this configuration, the solver continues until the image residual standard deviation falls below $\sigma = 2.5$. The gradient step size is initialised with $h_0 = 0.01$, and the re-noising parameter is set to $\beta = 0.01$. These default values provided by the original authors create a point of reference for subsequent modifications.

Shortened Convergence. To reduce computational cost, we configure the solver to complete within a fixed number of 25 iterations. This is achieved by increasing the gradient step size to $h_0 = 0.2$, lowering the initial noise standard deviation to $\sigma_0 = 0.8$, and setting the re-noising parameter to $\beta = 0.4$. This shortened configuration aims to preserve output quality while accelerating inference.

Fine-Tuning for the SR ($\times 4$) Task. With the shortened convergence configuration in place, we fine-tune the denoiser specifically for the super-resolution task. The iterative solver runs for 25 fixed steps per training example. The denoiser is fine-tuned as described in Section 6.2.3 using a single task instead of four. The training hyperparameters are the same as described in Section 6.3.5.

6.4.2 Multi-Task Training Scenarios

We conduct a series of fine-tuning experiments to assess how well a denoiser can generalise across multiple inverse problems when trained with multiple objectives. Three tasks, frequency super-resolution, random basis reconstruction, and demosaicing, are excluded from training and instead used as benchmarks to evaluate generalisation across the unseen problems.

All trained networks share the same DnCNN backbone and differ only in the set of tasks used during fine-tuning. We define the configurations as follows:

- BF-CNN** - The original, publicly available, bias-free DnCNN model (BF-CNN) was trained solely for denoising. The network is not fine-tuned further and serves as a baseline.
- 3-Task** - In this configuration, we fine-tune the BF-CNN network using three iterative tasks: 16×16 inpainting, SR- $\times 2$ and 10% compressed sensing. The denoising task is excluded.
- 4-Task** - This configuration extends the 3 Task setup by including an additional denoising objective, where the noise standard deviation is sampled uniformly as $\sigma \sim U(0, 100)$. Unlike the other tasks, denoising is performed in a single pass (one-shot) rather than iteratively. This reduces training complexity and reflects the structure of the reverse process in diffusion models.

6.4.3 Loss Weighting Scenarios

With the different task configurations established, we also seek to evaluate the impact of several weighting schemes. As previously discussed, some tasks produce loss values that converge to different scales due to varying task difficulties, causing tasks with smaller loss values to stop contributing to training early. The loss schemes evaluated are as follows:

Some tasks produce loss values that converge to different scales due to varying task difficulties, resulting in tasks with larger losses having a far larger impact on training than ‘easier’ tasks with smaller loss values. These smaller loss tasks subsequently underperform, stop

	Inpaint	SR-×2	Sensing	AWGN Denoising					
				$\sigma = 5$	$\sigma = 10$	$\sigma = 20$	$\sigma = 30$	$\sigma = 40$	$\sigma = 50$
1/w	0.014	0.035	0.048	0.011	0.017	0.026	0.033	0.040	0.046

Table 6.2 Table of weights used for the Normalised loss scheme in training. Here, the weights $\mathbf{w}$ are saved from previous losses to balance the contribution of each task to the overall loss, ensuring certain tasks do not dominate the gradient updates.

contributing to training early and underperform in testing relative to their single-task training results. Essentially, the loss in the Multi-Task configuration amounts to a weighted average over the tasks $\mathcal{L}_{\text{total}} = \sum_{i=0}^N w_i \mathcal{L}_{\text{task}_i}$. We consider here to adjust the relative weights of each task according to different strategies:

Uniform - In this configuration, the loss is weighted equally across all tasks. It is given by:

$$\mathcal{L}_{\text{total}} = \sum_{i=0}^N \mathcal{L}_{\text{task}_i},$$

where N is the number of tasks.

Fixed - In this configuration, the loss function is weighted to be biased towards the denoising task. This is implemented by applying a $\times 5$ weight to the denoising loss, while all other weights remain at 1. It is given by:

$$\mathcal{L}_{\text{total}} = 5 \cdot \mathcal{L}_{\text{denoise}} + \sum_{i=0}^{N-1} \mathcal{L}_{\text{task}_i},$$

where $\mathcal{L}_{\text{denoise}}$ is the loss for the denoising task, and the remaining terms correspond to the other $N - 1$ tasks.

Normalised - In this configuration, the loss function is optimised to equalise the impact of each task on the gradient computation, preventing tasks with larger loss values from disproportionately influencing the optimisation process. The weight for each task, w_i , is computed from a previous training run and saved. It is defined as:

$$\mathcal{L}_{\text{total}} = \sum_{i=0}^N w_i \cdot \mathcal{L}_{\text{task}_i},$$

where $w_i = 1 / \mathcal{L}_{\text{task}_i}^{\text{prev}}$ normalises the contribution of each task, ensuring balanced gradients during training. For the denoising task, the weights are interpolated based on the sampled noise standard deviation. These saved weights are shown in Table 6.2.

6.4.4 Length of Iterations

As discussed in Section 6.3.4, we configure the framework to converge at a different number of iterations. In our experiments, we assess both the original BF-CNN model (with optimised scheduling parameters) and our fine-tuned model in the 50 iteration configuration and the 100 iteration configuration.

6.4.5 Beyond the Baseline

In order to fairly assess the fine-tuning of the denoiser, we have not modified the denoising architecture between experiments. In our final experiments, we assess the impact of replacing the DnCNN backbone with our own U-Net.

Custom U-Net Architecture

For our final ablation study, we implement a custom U-Net, with three downsampling operations in the encoder stage of the network, implemented through strided convolutions. In the decoder, we use pixel shuffle [295] operations to upsample the image, paired with skip connections from the corresponding layer in the encoder. These skip-connections are implemented as element-wise additions, rather than concatenations, to simplify training.

To improve generalisation over a wide input variance, all convolutions are bias-free, and 3×3 kernel sizes are used throughout. Between each convolution, we apply GELU activation [296] functions, which allow small negative values to pass through, preserving valuable information in high-noise, low-brightness images. This contrasts with ReLU, which zeroes out all negative values, potentially discarding useful information.

The final network consists of 11,656,643 trainable parameters. Inference time remains comparable to BF-CNN due to the use of fewer layers (15 vs. 20). Additionally, by concentrating the largest number of feature maps at the lowest resolution stages of the U-Net, we effectively manage memory usage during inference. In our experiments, we firstly train the U-Net as a pure denoiser before it is fine-tuned using the multi-task training scheme to recreate the denoiser natural image prior as before.

6.5 Results

This section presents the results of our experiments, beginning with the initial single-task fine-tuning studies, followed by the multi-task training evaluations and the U-Net model

AWGN Noise STD	Noisy Image (dB)	Original Denoiser (dB)	SR- $\times 4$ Fine-Tuned Denoiser (dB)
$\sigma=5$	34.16	37.17	34.25
$\sigma=10$	28.14	34.39	32.48
$\sigma=20$	22.12	31.57	29.95
$\sigma=30$	18.60	29.93	28.31
$\sigma=40$	16.10	28.77	27.14
$\sigma=50$	14.16	27.85	26.28

Table 6.3 Denoising Results with the original denoiser, and with the fine-tuned denoiser, trained on super-resolution ($\times 4$). We also include the degraded PSNR for the downsampled input image, projected onto the measurement space using M^T .

performance. We also provide a qualitative analysis of visual quality through selected image comparisons.

6.5.1 Single Task Training

In the bottom left and bottom right plots of Figure 6.4, we present the results for the shortened convergence framework (left), and the fine-tuned denoiser trained specifically for super-resolution (right). For clarity, only the first four images from the Set14 dataset are plotted.

In the shortened convergence experiment, comparable performance to the original framework is achieved using only 25 iterations, with a +0.38 dB improvement in average PSNR and a processing time reduction from 7.8 seconds per image to under one second. This demonstrates that carefully chosen hyperparameters can significantly accelerate inference without sacrificing quality.

Further improvements are observed with the fine-tuned denoiser, which delivers an average +0.95 dB gain over the original model. These results validate the effectiveness of task-specific fine-tuning within the proposed iterative framework.

To assess how this task-specific training affects denoising performance, we evaluate the fine-tuned model on a standard denoising task using AWGN noise levels of [5, 10, 20, 30, 40, 50]. As shown in Table 6.3, the denoising performance degrades substantially compared to the original denoiser, with an average PSNR loss of 3.5 dB. This confirms that fine-tuning improves performance for the target task but compromises general-purpose denoising capabilities.

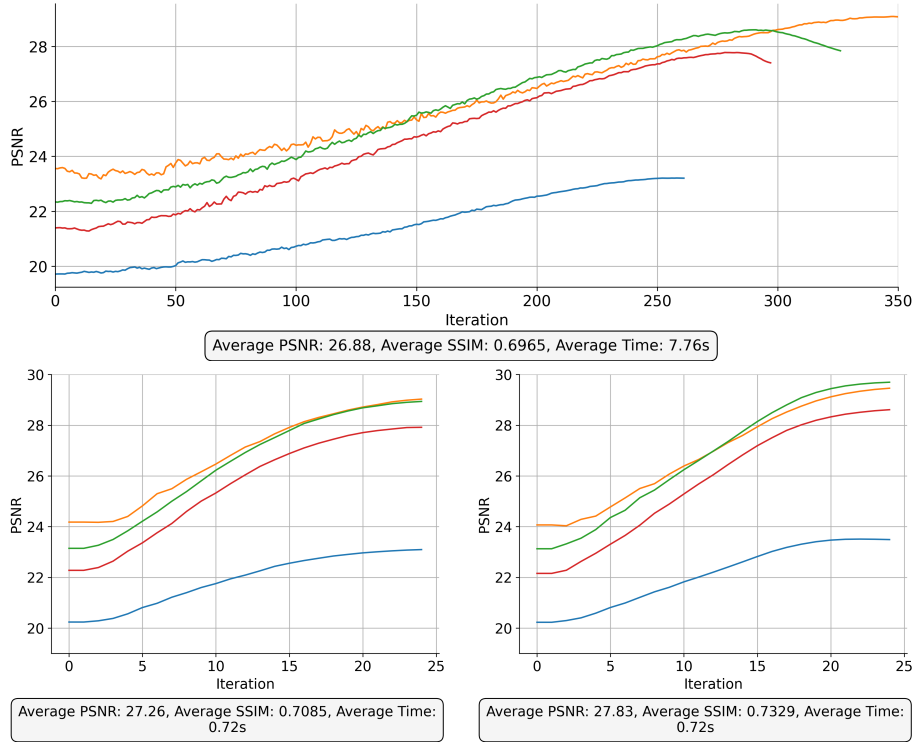


Fig. 6.4 Initial results of single-task denoiser fine-tuning on the SR-x4 task using the iterative inverse solver. Each subfigure displays PSNR trajectories for four test images as they converge through gradient ascent, where the plot colour corresponds to the same image. **Top:** Original configuration from Kadkhodaie et al. **Bottom left:** Convergence fixed to 25 iterations using hand-tuned hyperparameters ($\sigma_0 = 0.8$, $\beta = 0.2$, $h_0 = 0.03$). **Bottom right:** Model fine-tuned specifically for the SR-x4 task. Average PSNR, SSIM, and mean denoising time are reported below each plot.

Lastly, Figure 6.5 shows a qualitative comparison between the original solver output and the result of the trained model. The latter shows enhanced sharpness and texture continuity, particularly around the zebra’s head and neck, indicating improved perceptual quality.

6.5.2 Multi Task Training

In Table 6.4, we report PSNR results (in dB) for each network across both the trained tasks (inpainting, SR-x2, compressed sensing) and the untrained tasks (frequency SR, random basis reconstruction, demosaicing). Each value represents the mean result computed over 100 test images. For fine-tuned models, the difference in performance relative to the equivalent BF-CNN baseline is shown in parentheses. Across 12 experimental runs using different random seeds, the 95% confidence intervals for PSNR remain within 0.09 dB for all evaluated tasks, with the exception of demosaicing, which exhibits higher variability (± 0.13 dB). This indicates a high degree of consistency in the results across most tasks.

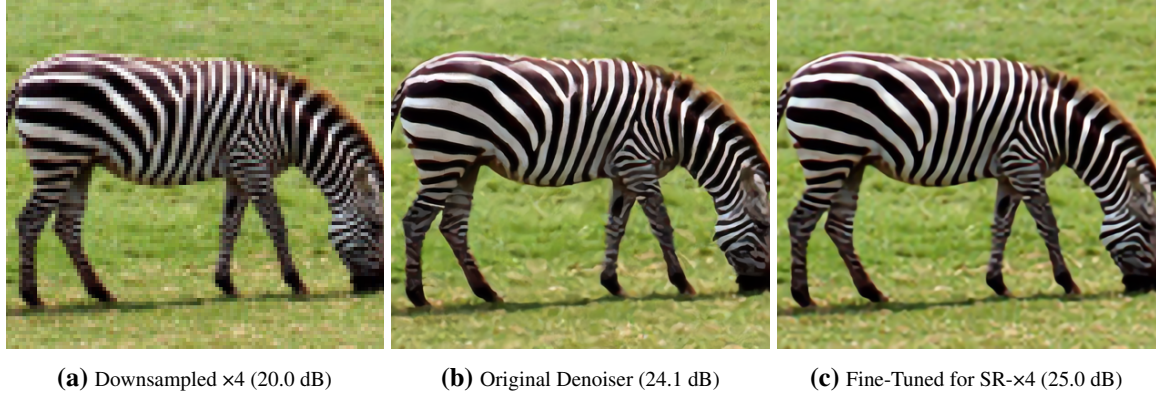


Fig. 6.5 Super-resolution ($\times 4$) output samples for the original denoiser prior, with unmodified linear inverse framework (b), and the fine-tuned denoiser with the fixed iteration framework (c). Image taken from Set14 test set.

The 3 Task^U model, trained with a uniform loss weighting scheme for 50 iterations, yields substantial improvements on the tasks seen during training: inpainting improves by +4.08 dB, super-resolution by +1.66 dB, and compressed sensing by +2.09 dB. However, this model exhibits poor generalisation to the untrained tasks, most notably with a -6.90 dB drop on demosaicing compared to the baseline. As a result, it achieves the lowest overall mean improvement across all tasks.

When the denoising task is included during training (4 Task^U), generalisation improves considerably, with a mean gain of +1.12 dB over the baseline. This inclusion also helps close the gap in demosaicing performance relative to the original denoiser.

Both the original and fine-tuned networks show improved performance when evaluated at 100 iterations. The best overall results are obtained by the 4 Task^N model, which achieves a mean PSNR of 32.53 dB—representing a +1.34 dB improvement over the baseline BF-CNN evaluated at the same iteration count.

Denoising Performance

In Table 6.5, we report one-shot denoising results for all iteratively trained models across a range of noise levels. As expected, the original BF-CNN model, trained exclusively for denoising, achieves the highest average PSNR (32.47 dB). In contrast, the 3 Task^U model, which excludes denoising during training, performs the worst, with an average PSNR drop of 4.65 dB relative to the baseline.

Introducing the denoising task into training substantially improves results. The 4 Task^U model recovers much of the lost performance, achieving an average within 0.93 dB of the

	Degraded	BF-CNN	3 Task ^U	4 Task ^U	4 Task ^F	4 Task ^N	BF-CNN	4 Task ^N	
Iterations	-	50	50	50	50	50	100	100	
trained	Inpainting	23.27	33.67	37.75 (+4.08)	36.89 (+3.22)	36.61 (+2.94)	37.36 (+3.69)	36.16	37.41 (+1.25)
	SR- $\times 2$	27.32	28.06	29.72 (+1.66)	29.22 (+1.16)	28.61 (+0.55)	27.36 (-0.70)	28.78	29.78 (+1.00)
	Sensing	7.33	24.79	26.88 (+2.09)	26.28 (+1.49)	25.54 (+0.75)	26.27 (+1.48)	25.22	27.09 (+1.87)
untrained	Freq. SR	27.65	29.92	30.95 (+1.03)	30.59 (+0.67)	30.24 (+0.32)	29.38 (-0.54)	30.30	31.23 (+0.93)
	Rand. Basis	7.53	29.26	27.66 (-1.60)	30.46 (+1.20)	29.36 (+0.10)	30.50 (+1.24)	29.10	31.45 (+2.35)
	Demosaic	8.68	38.74	31.84 (-6.90)	37.70 (-1.04)	36.36 (-2.38)	37.47 (-1.27)	37.59	38.22 (+0.63)
	Mean	16.96	30.74	30.80 (+0.06)	31.86 (+1.12)	31.12 (+0.38)	31.39 (+0.65)	31.19	32.53 (+1.34)

Table 6.4 PSNR Results (dB) for solving linear inverse problems using various denoiser training strategies. **BF-CNN** denotes the model weights used by Kadkhodaie, **3 Task** is trained on three iterative tasks: inpainting, $\times 2$ super-resolution, and compressed sensing. **4 Task** adds a fourth, non-iterative denoising task ($\sigma \sim U(0, 100)$). Superscripts indicate the loss weighting scheme: ^U = uniform, ^F = fixed, ^N = normalised. Differences (in parentheses) are relative to BF-CNN at the same iteration setting. With the exception of demosaicing, all values lie within 0.09 dB 95% confidence intervals across repeated runs.

Model	Degraded	BF-CNN	3-Task ^U	4-Task ^U	4-Task ^F	4-Task ^N	4-Task ^N	
Iterations	50	50	50	50	50	50	100	
<i>Noise STD.</i>	$\sigma = 5$	34.17	39.94	37.50 (-2.44)	39.19 (-0.75)	39.20 (-0.74)	39.60 (-0.34)	39.25 (-0.35)
	$\sigma = 10$	28.15	35.91	32.95 (-2.96)	35.56 (-0.35)	35.71 (-0.20)	35.79 (-0.12)	35.35 (-0.44)
	$\sigma = 20$	22.13	32.17	27.92 (-4.25)	31.59 (-0.58)	32.09 (-0.08)	32.01 (-0.16)	31.92 (-0.25)
	$\sigma = 30$	18.61	30.16	24.84 (-5.32)	29.29 (-0.87)	30.07 (-0.09)	29.97 (-0.19)	29.99 (-0.31)
	$\sigma = 40$	16.11	28.81	22.67 (-6.14)	27.56 (-1.25)	28.70 (-0.11)	28.58 (-0.23)	28.67 (-0.13)
	$\sigma = 50$	14.17	27.82	21.01 (-6.81)	26.04 (-1.78)	27.68 (-0.14)	27.52 (-0.30)	27.68 (-0.14)
Mean	22.22	32.47	27.82 (-4.65)	31.54 (-0.93)	32.24 (-0.23)	32.25 (-0.22)	32.15 (-0.32)	

Table 6.5 PSNR (dB) results for denoising using the same models shown in Table 6.4. All denoising results are computed in a one-shot manner and are not iterative. The same test dataset is denoised at six separate noise levels (σ) in each experiments.

original denoiser. Applying fixed loss weighting (4 Task^F), which emphasises denoising, further narrows the gap to just 0.23 dB. Notably, the 4 Task^N model, trained with normalised loss weighting, performs nearly identically—just 0.22 dB below the baseline—while maintaining the strongest overall performance on the inverse problems. This indicates that normalised weighting provides a favourable trade-off between task-specific accuracy and broader generalisation.

Finally, the 100-iteration version of the same model (4 Task^N) achieves slightly lower denoising performance (32.15 dB), suggesting a small degradation which may result from prolonged exposure to high noise levels during iterative training, or the distribution shift introduced by additional solver iterations.

		BF-CNN (50)	4 Task ^N (50)	U-Net ^N (50)
<i>trained</i>	Inpainting	33.67	37.36 (+3.69)	38.27 (+4.60)
	SR-x2	28.06	27.36 (-0.70)	29.79 (+1.73)
	Sensing	24.79	26.27 (+1.48)	26.91 (+2.12)
<i>untrained</i>	Freq. SR	29.92	29.38 (-0.54)	30.77 (+0.85)
	Rand. Basis	29.26	30.50 (+1.24)	31.19 (+1.93)
	Demosaic	38.74	37.47 (-1.27)	38.83 (+0.09)
	Mean	30.74	31.39 (+0.65)	32.63 (+1.89)

Table 6.6 PSNR (dB) Inverse solving results comparing baseline architecture best-effort model for the 50 iteration model (4-Task^N) and the U-Net model.

		BF-CNN (50)	4-Task ^N (50)	U-Net (50)
<i>Noise STD.</i>	$\sigma = 5$	39.94	39.60 (-0.34)	40.05 (+0.11)
	$\sigma = 10$	35.91	35.79 (-0.12)	36.02 (+0.11)
	$\sigma = 20$	32.17	32.01 (-0.16)	32.31 (+0.14)
	$\sigma = 30$	30.16	29.97 (-0.19)	30.33 (+0.17)
	$\sigma = 40$	28.81	28.58 (-0.23)	29.02 (+0.21)
	$\sigma = 50$	27.82	27.52 (-0.30)	28.04 (+0.22)
	Mean	32.47	32.25 (-0.22)	32.63 (+0.16)

Table 6.7 PSNR (dB) denoising results comparing baseline architecture best-effort model for the 50 iteration model (4-Task^N) and the U-Net model.

6.5.3 U-Net Performance.

We conclude with results for the iteratively trained U-Net model. Due to the increased model size and GPU memory constraints, training is limited to 50 iterations. Inverse problem results are summarised in Table 6.6, while denoising performance is shown in Table 6.7. In both cases, we compare the U-Net architecture against the best-performing DnCNN variant, 4 Task^N . For inverse solving, the U-Net consistently outperforms the DnCNN across all tasks, increasing the average PSNR improvement over the baseline from +0.65 dB to **+1.89 dB**. In the denoising evaluation, the U-Net also surpasses the original denoiser, achieving a mean gain of **+0.16 dB**. These findings highlight the potential of larger networks to generalise effectively across a broad range of linear inverse problems, including denoising.

6.5.4 Visual Assessment

Figures 6.6 and 6.7 present reconstructed samples from the test dataset for two tasks: compressive sensing (10%) and frequency super-resolution, respectively. For each task, we show the original image, its degraded counterpart, the 50-iteration reconstruction using the original model ($BF\text{-CNN}$), and the 100-iteration output using the fine-tuned model (4 Task^N).

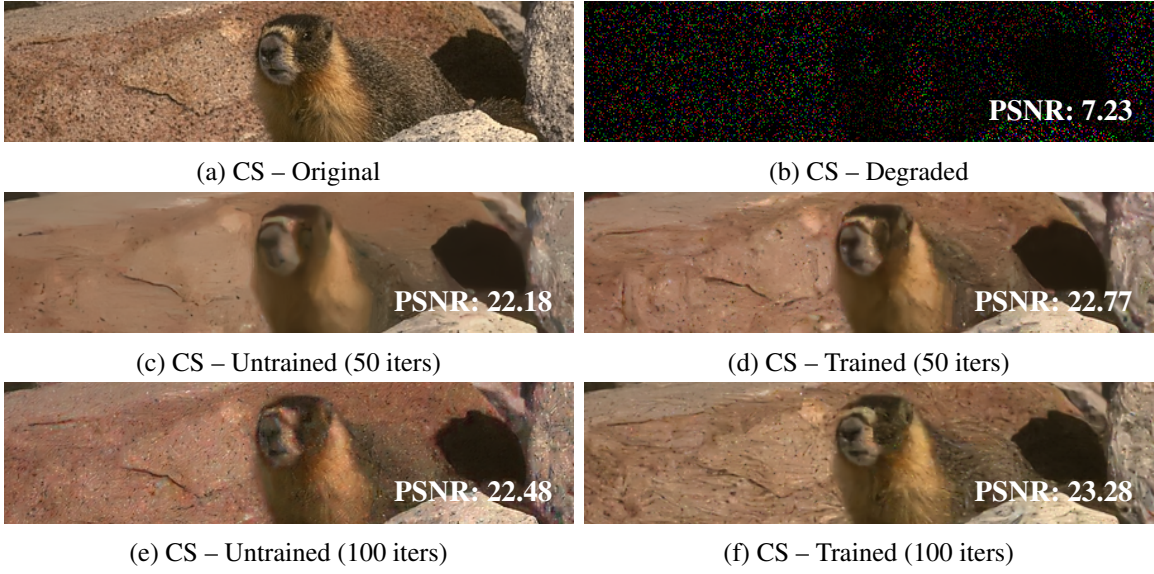


Fig. 6.6 Compressed-sensing reconstruction examples. PSNR values are printed at the lower-right of each image.

In the compressive sensing examples, notable differences are observed in texture synthesis. Across all samples, the iterative solver generates distinct textures in the rocky background. The trained models consistently recover sharper high-frequency details—especially around the rodent’s face and the surrounding terrain. For instance, the 100-iteration trained model can reconstruct fine hair on the rodent’s neck that is entirely absent from the BF-CNN output.

Similar improvements are observed in the frequency super-resolution samples. The wooden pier in the original image contains strong diagonal structures which are better preserved by the trained models. These structures appear blurred or softened in the BF-CNN output but are much more clearly defined after fine-tuning. Additionally, while both trained models improve edge fidelity, the 100-iteration variant produces a smoother result and reduces the staircasing artefacts that remain visible in the 50-iteration version.

Figure 6.8 provides comparative output samples for the DnCNN backbone networks trained at 50 iterations and 100 iterations. The tasks examined are inpainting, super-resolution ($\times 2$), $\sigma = 40$ denoising, frequency super-resolution and random basis reconstruction. The first column shows the constrained images, which have been projected back onto the original image space. Cropped portions of each image are displayed, along with the PSNR values for the full images.

Inpainting. In the inpainting task, the 3-Task model achieves the highest PSNR but reconstructs less structure within the masked region compared to the 4-Task models. It produces a

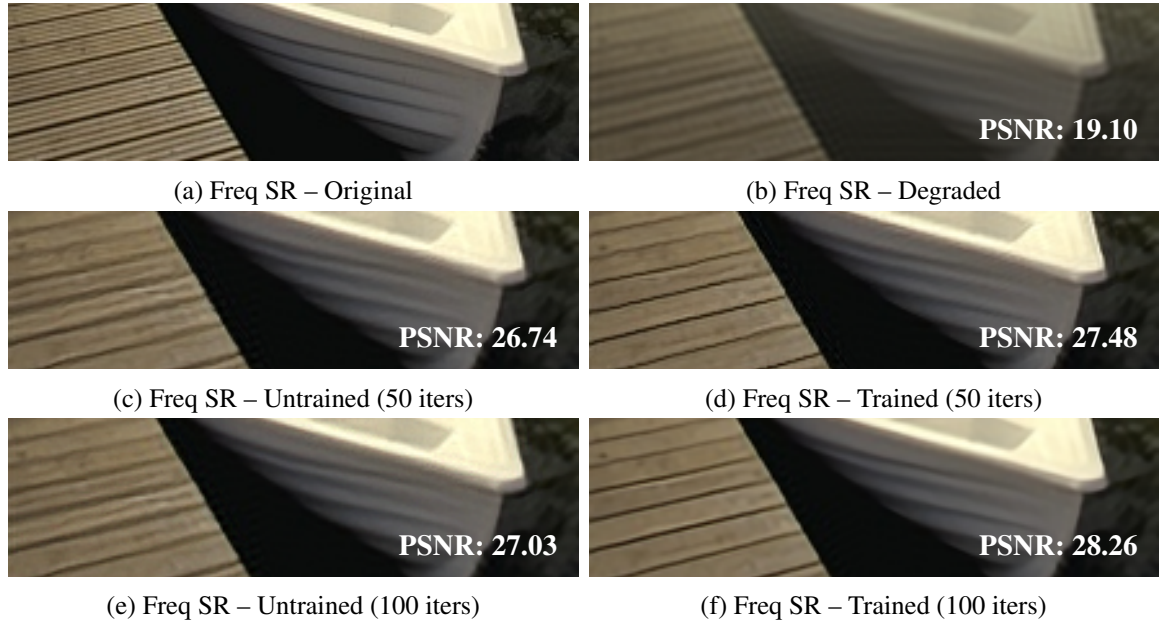


Fig. 6.7 Frequency-super-resolution results. PSNR values are printed at the lower-right of each image.

blurry texture with weak definition, likely reflecting high fidelity only outside the masked area, as we measure the entire image PSNR. In contrast, the 4-Task models introduce sharper, more plausible structures, albeit at the cost of lower pixel-level similarity to the ground truth.

Compressive Sensing. The 3-Task (50) model, despite lacking denoising training, recovers more texture on the penguin’s chest than the 4-Task (50) model. This detail is further improved by the 4-Task (100) model, which benefits from additional iterations.

Super-Resolution. In the super-resolution task, the 3-Task model achieves a higher PSNR with smooth textures, especially on the statue. The 4-Task (50) model shows weaker results, but the 100-iteration version closes the gap and matches the detail recovery.

Denoising. The 3-Task model performs poorly in denoising, leaving noticeable noise. While the original denoiser yields the highest PSNR, it introduces over-sharpening and piecewise smooth artefacts. The 4-Task models reduce these issues, and the 100-iteration version slightly over-smooths the boundary between the wolf and background.

Frequency Super-Resolution. Ringing artefacts above the tiger’s head are reduced by the fine-tuned models and are best suppressed by the 4-Task (100) model, in line with the improved PSNR.

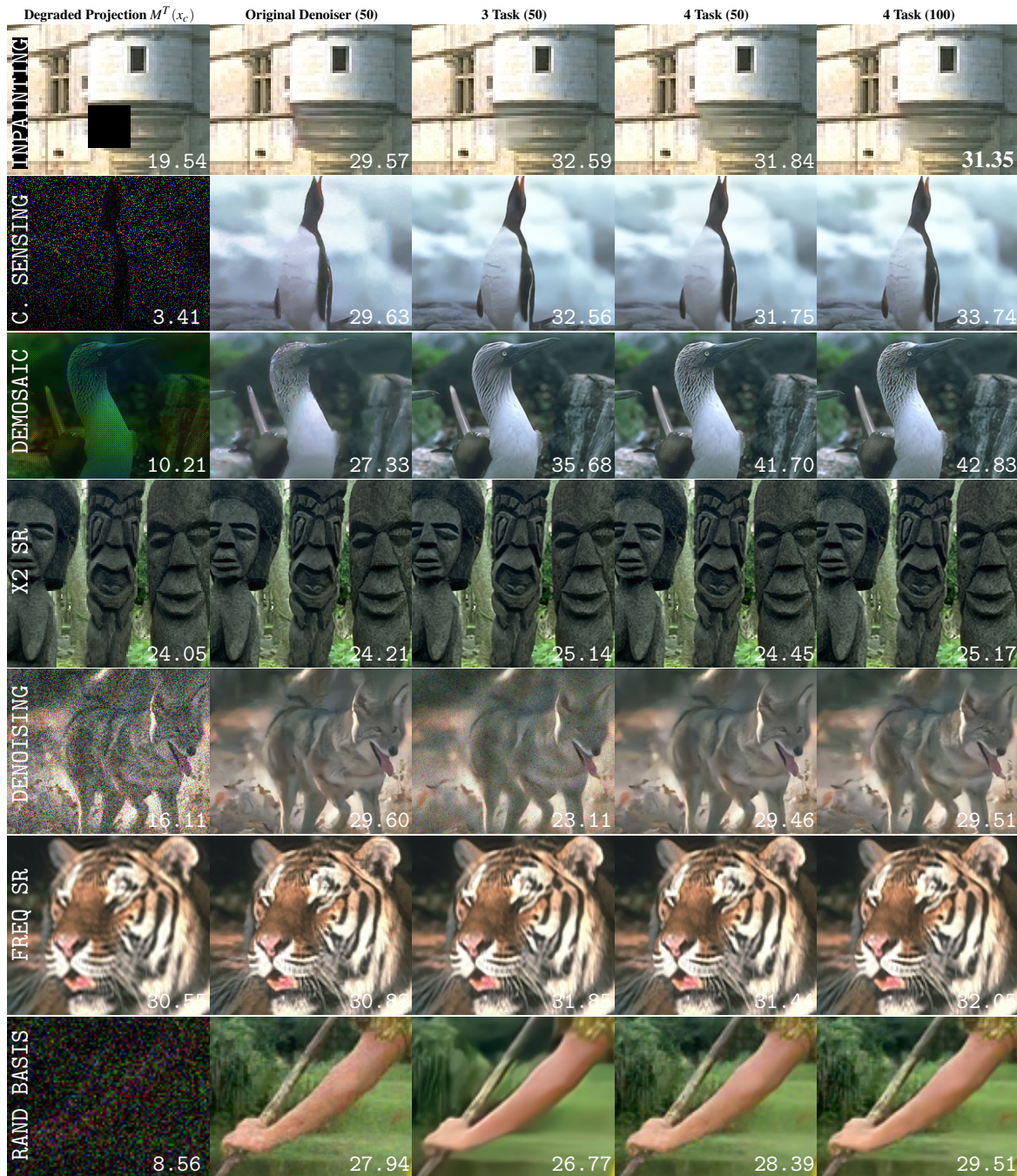


Fig. 6.8 Sample Results of trained denoisers for inverse solving. Each column represents a different denoiser and each row represents a different task. From left to right we show the projection of the constrained measurement matrix x_c , the result of using the original BF-CNN denoiser prior, the 3 task fine-tuned prior and the 4-task, normalised loss fine-tuned denoiser. From top to bottom, we the following tasks: inpainting, x2 super-resolution, $\sigma = 40$ AWGN denoising, frequency super resolution, 10% random basis reconstruction.

Random Basis. Despite the difficulty of the task, all models produce a recognisable reconstruction. The original denoiser retains noise-like texture, while the 4-Task model recovers sharp textures in the background and smooth tones in the arm. The 3-Task model, though lower in PSNR, generates clean edges and structured texture. The 4-Task (100) model balances smoothness, edge definition, and texture detail most effectively, consistent with its superior PSNR.

Two additional pages of output samples, which compare our final U-Net model against the original denoiser and the task normalised loss model, are provided in Figures 6.9 and 6.10. The latter two models are outperformed by the U-Net in every task, with superior texture reconstruction apparent in a number of examples. For example, in the compressed sensing samples, the U-Net model recreates more texture in the penguin’s feathers, as well as on the rocks beneath it. The quality improvement is also clear in the final denoising example, where the sea anemone tentacles in the background are smoother in the U-Net example with clearer edges and no erroneous texture generation, unlike the original model.

6.6 Discussion

This section provides an in-depth discussion of key insights from our results, including the perception-distortion trade-off and challenges associated with training denoisers for multiple tasks.

6.6.1 The Perception-Distortion Trade-off

Our results demonstrate that fine-tuning the natural image prior inherent in denoisers enhances performance across various linear imaging tasks. However, perceptual assessments reveal that optimising solely for PSNR may not suffice. Recalling the Miyasawa-Tweedie identity (Equation 6.9), the gradient ascent process converges when it reaches a point in the prior probability space of real images that maximises the probability $p(x)$. Within this subspace lies a manifold of equally likely versions of the predicted image $\hat{x}$, each differing in synthesised details, textures, and objects. While these images are considered equally natural by the modelled prior, only one may maximise PSNR.

As illustrated in our sample images—particularly the inpainting example—the image that maximises PSNR may actually be further from the perceptually preferred sample. This highlights the challenge of training the algorithm using traditional distance functions like MSE, which penalise the introduction of new details. This issue is increasingly significant in

image and video restoration, where large networks capable of generating images that align well with perceptual scores are penalised due to conventional loss functions. This problem is thoroughly examined in *The Perception-Distortion Tradeoff* [53], which mathematically proves that distortion measurements (e.g., PSNR/SSIM) and perceptual quality are at odds. In essence, minimising distortion often requires deviating from the statistics of natural scenes.

Given this observation, a logical extension of our work would be to adopt a training scheme that prioritises perceptual quality, aligning more closely with the philosophy underpinning Miyasawa’s proof upon which our framework is built.

6.6.2 Training Challenges

In this section, we discuss some of the challenges encountered in optimising the models for inverse solving. One of the main focuses throughout the optimisation process was to ensure denoising performance remained close to the original denoiser while simultaneously improving performance on all other tasks. In the 3 task model, maximising the performance for the trained tasks is straightforward, but denoising quality suffers by 4.6 dB on average. As the goal is to produce an enhanced, generic denoiser suitable for all tasks, this result is not acceptable. The experiments that followed were motivated by these results.

Including denoising in the fine-tuning process (4-Task) was a logical step in improving overall performance. With this implementation, the overall mean improves by + 2.4 dB. Across all tasks except denoising and demosaicing, we outperform the original BF-CNN denoiser. To further approach the original models’ denoising performance, the weighted loss function is introduced. Denoising performance is further improved with this model (4-task weighted); however, the other tasks suffer and the overall mean is reduced. Next, by equalising (or normalising) the loss of each task, the network trains more efficiently and denoising is also improved (4-task equal).

In our work, we found that training denoisers (prior to the fine-tuning stage) for such a large range of input noise levels is not a straightforward task. During training, the network is exposed to noise with a maximum standard deviation of 100. However, in the initial stages of framework inference, the noise standard deviation can approach 255. We observed that larger models often fail to generalise well to noise levels significantly greater than those encountered during training. Furthermore, while the BF-CNN utilises bias-free batch normalisation, its inclusion in our models adversely affected performance. This occurs because normalising across a batch of images with varying noise variances leads to a loss of critical noise magnitude information.

The implementation of a U-Net specifically introduced additional challenges in this context. For instance, the use of transposed convolutions in the decoder generated checkerboarding artefacts, which were exacerbated by the iterative nature of the algorithm. Transposed convolutions work by inserting zeros between input pixels and then applying a standard convolutional kernel, effectively spreading single pixel values over a larger area. If the kernel and stride are not well-matched, this uneven overlap can lead to visible artefacts such as checkerboarding. Additionally, due to the downsampling operations in the U-Net, padding is often required during inference to preserve dimensionality. Zero-padding in particular created an edge case not observed during training. Incorrect single pixels were sometimes introduced in the initial iterations, typically at the corners of padded images, and these artefacts propagated throughout the image, ultimately destroying it. To address this issue, we employed reflection padding and augmented the training set with padded images to ensure the model learned to handle such cases effectively.

We also carried out several additional experiments which are not included in the results section due to their inconclusive or marginal significance. These include experiments involving noise sampling distributions, training task-specific denoisers for different inference stages, and learning the scheduler parameters (β , h_0 , σ_0) of the iterative solver. While these showed some promising trends, they are omitted for clarity and to maintain focus on the primary findings of this chapter.

6.7 Conclusions

This chapter concludes the technical portion of this thesis by showcasing how denoising networks, originally trained for noise removal, can extend their utility by leveraging the powerful implicit priors they model to solve inverse problems. We contribute to this area of research by introducing a novel training process that directly fine-tunes the denoiser for integration within the iterative framework. This approach fundamentally redefines the denoising problem, adapting the denoiser to accommodate various linear constraints rather than treating noise removal as an isolated task. We also implement our own more powerful architecture to further enhance results and once again validate the use of the implicit prior in denoising networks.

Our final implementation demonstrates the effectiveness of this method, presenting an enhanced natural image prior through our fine-tuned denoiser. The refined model outperforms the original BF-CNN denoiser across our benchmark, which consists of trained and untrained tasks. These findings highlight the versatility of denoising networks and their potential to

tackle broader challenges in computational imaging. However, our results also reveal the perception-distortion trade-off, indicating that optimising for traditional distortion metrics like PSNR may not align with perceptual quality. Future work should consider training schemes that prioritise perceptual quality to more closely align with human visual preferences and further enhance the applicability of denoising networks in real-world scenarios.



Fig. 6.9 Sample Results of trained denoisers for inverse solving. Each column represents a different denoiser and each row represents a different task. From left to right, we show the projection of the constrained measurement matrix x^c , the result of using the original BF-CNN denoiser prior, the 3-task fine-tuned prior and the 4-task, normalised loss fine-tuned denoiser. From top to bottom, we have the following tasks: inpainting, $\times 2$ super-resolution, $\sigma = 40$ AWGN denoising, frequency super resolution, 10% random basis reconstruction.

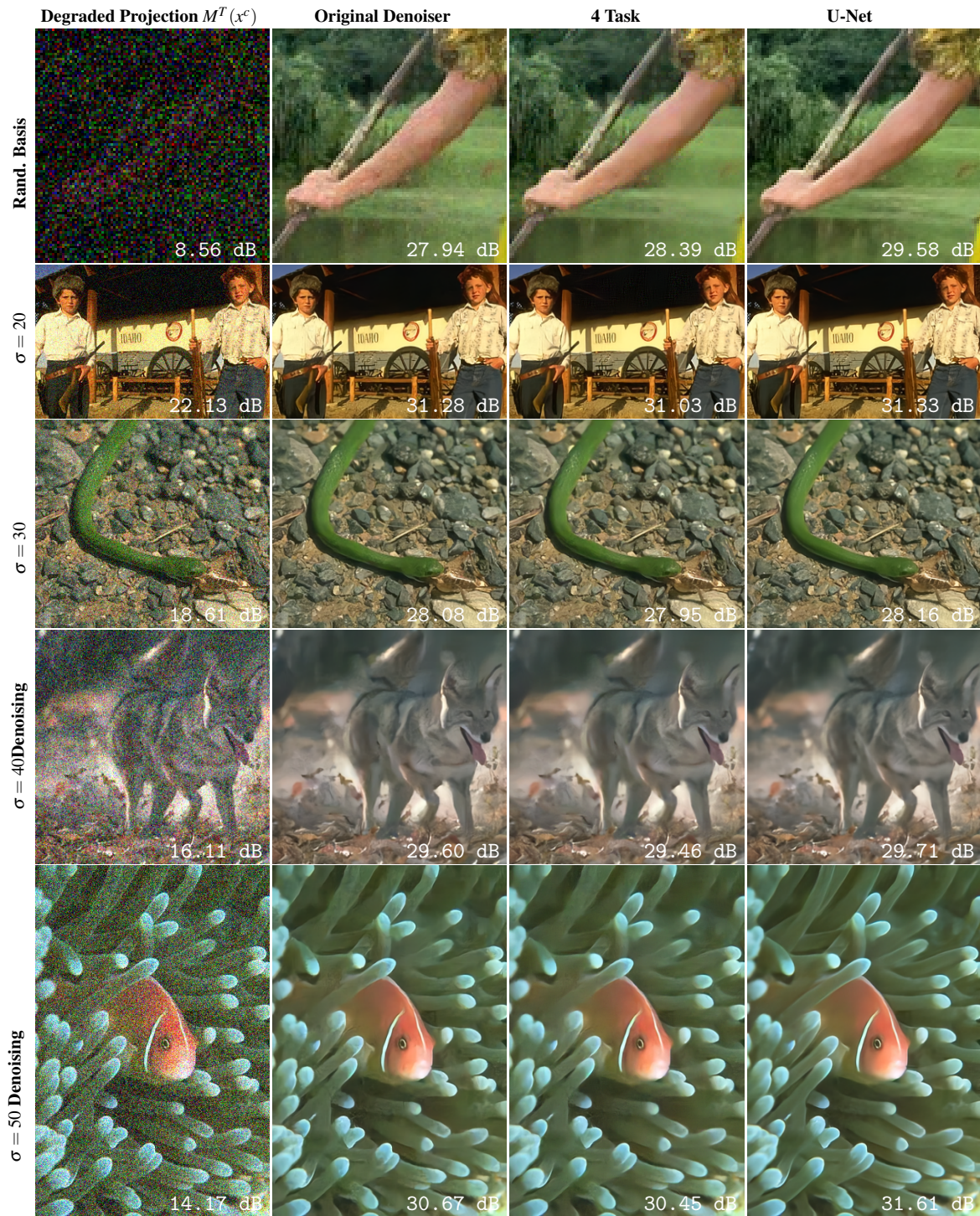


Fig. 6.10 Sample Results of trained denoisers for inverse solving. Each column represents a different denoiser and each row represents a different task. From left to right, we show the projection of the constrained measurement matrix x^c , the result of using the original BF-CNN denoiser prior, the 3-task fine-tuned prior and the 4-task, normalised loss fine-tuned denoiser. From top to bottom, we have the following tasks: inpainting, $\times 2$ super-resolution, $\sigma = 40$ AWGN denoising, frequency super resolution, 10% random basis reconstruction.

Chapter 7

Conclusions

This thesis has presented a comprehensive investigation into visual denoising techniques, focusing on the development of lightweight models in response to the growing trend of computationally demanding denoisers in contemporary image and video denoising literature. Central to this work are the four research questions posed in Chapter 1 which have been systematically addressed through the course of this research.

Firstly, a set of guidelines for designing real-noise denoisers is proposed, detailing optimal training strategies and architectural considerations (RQ1). Secondly, a hybrid image denoiser is implemented, combining classical denoising methods with deep neural networks to enhance performance while minimising computational costs (RQ2). Thirdly, a novel hybrid video denoising algorithm is developed, leveraging temporal and spatial frequency data to outperform state-of-the-art methods across various denoising tasks (RQ3). Finally, we fine-tune a denoiser to improve its performance in solving a series of linearly constrained imaging tasks, when used within an iterative, implicit prior framework (RQ4).

7.1 Summary of Thesis Contributions

In this section, we summarise the key contributions made in response to the research questions previously outlined.

Through our assessment of real-noise-optimal design (Chapter 3), several recommendations are presented to maximise denoising performance in this context. These include the use of synthetic-real noise combinations in training sets, the optimisation of network size in terms of parameters per-layer, and the implementation of real-noise specific, noise mapping

ancillary networks. The original DnCNN architecture is redesigned with these considerations in mind, and final results achieve a + 5.04 dB improvement on the DND benchmark and + 3.35 dB improvement on the SIDD benchmark.

The hybrid Wiener-CNN image denoiser demonstrates the feasibility of integrating classical and deep learning methods to achieve competitive real-noise denoising performance on a lightweight platform. On final benchmarks, it improves the baseline Wiener filter's performance by +3.11 dB, surpassing DnCNN while using an order of magnitude fewer parameters. This success is achieved through several design innovations: the filter is made fully differentiable, parallelised, and GPU-accelerated, enabling end-to-end training of ancillary CNNs while significantly reducing denoising times. The preprocessing pipeline is optimised to refine windowing characteristics and address DC offset removal. To enable blind denoising, an ancillary network performs localised noise estimation, while a secondary network refines each transfer function in the frequency domain to minimise output artefacts. Additionally, a multi-scale filtering process captures frequency details across different spatial resolutions, resulting in output images with strong contrast and smooth textures, effectively suppressing artefacts commonly associated with Wiener filters.

A hybrid Wiener-CNN video denoiser is also developed, competing with state-of-the-art transformer models and outperforming networks such as VRT [36] in SSIM across all noise levels in both blind and non-blind configurations. This performance is achieved with only a fraction of the trainable parameters required by transformers and with denoising times 90% faster on the same hardware. To handle the larger temporal signal, the filter is redesigned to incorporate 4D FFT analysis, custom-trained windowing functions, and 3D convolutional ancillary networks for noise prediction and Wiener refinement. Comparative analysis with other video denoisers highlights enhanced contrast in dark regions and strong temporal consistency across a wide range of noise levels.

Lastly, we demonstrate that the performance of the implicit prior method proposed by Kadkhodaie and Simoncelli [45] for inverse problem solving can be enhanced by fine-tuning the denoiser within the framework itself. This is accomplished by modifying the framework to converge more quickly, replacing the stopping criteria threshold with a fixed number of iterations, and optimising hyperparameters to take larger steps in the direction that maximises $p(x)$. Several training modes are tested, incorporating multiple tasks with different loss weighting schemes. Best-effort training results show a + 0.22 dB overall improvement over the original model across benchmarked tasks - including inpainting, super-resolution ($\times 2$), compressed sensing, denoising, frequency super-resolution, random basis reconstruction and demosaicing - and a + 0.99 dB improvement with a custom U-Net implementation.

7.2 Open Challenges in Denoiser Design

Through the course of our thesis research, we have encountered several fundamental challenges as we design and implement denoising algorithms. These challenges are not unique to our work and represent significant hurdles in field which must be overcome to further advance the state-of-the-art in denoising.

7.2.1 The Difficulty in Training for a Broad Range of Noise Intensities

Unlike the majority of imaging tasks (e.g., classification, super-resolution, segmentation, deblurring), where input pixel values are typically normalised to the range $[0, 1]$, a denoiser trained on a broad range of noise strengths (e.g., $\sigma = [0, 50]$) is likely to encounter pixel intensities that extend far beyond this range. Both positive and negative pixel values can exceed $[0, 1]$ by a factor of two or three due to the additive nature of noise. This variability presents significant challenges during training, as high inter-batch and intra-batch variance exacerbate internal covariate shift, destabilising training, making it difficult for the optimiser to converge effectively. These issues can be compounded by the adoption of network architectures intended for other tasks, where unsuitable activation functions may discard critical negative noise information. Additionally, the use of batch normalisation, which assumes consistent data distributions across batches, proves ineffective in this context as noise severely alters the distribution of individual images, leading to poor normalisation and degraded performance.

This issue has been identified through both our review of current denoising techniques and the training of our own models. For example, authors of denoisers commonly provide multiple models trained on the same architecture, each tailored to a narrow noise range, leaving users to select the most appropriate model for their needs. In contrast, the Wiener-CNN models were less affected by this phenomenon due to the preprocessing applied to the data. However, this issue became a critical consideration in fine-tuning the denoiser prior in the final chapter, where a small CNN model is required to handle a wide range of input noise intensities effectively, despite optimisations such as the use of bias-free networks and the loss normalisation scheme.

This difficulty in training denoisers for a large input noise range presents an interesting opportunity for further optimisation in deep denoisers. Such studies may include an assessment of the activation function used in early layers, as well as ablation studies on the impact of alternative normalisation functions in early layers such as instance-norm and layer-norm.

7.2.2 The Difficulty in Training for Real Noise

One of the fundamental challenges in training for real noise arises from the difficulty of defining the space of possible noise profiles. While the image prior $p(x)$ is relatively intuitive to conceptualise, the noise likelihood $p(\varepsilon|x)$ —the probability distribution of noise given an image—is far less well-defined. Noise can stem from a wide range of sources and manifest in highly variable ways, making it necessary to make assumptions about its characteristics in order to formalise a model. These assumptions inevitably limit the ability to accurately represent all potential noise profiles.

Training a CNN denoiser in a data-driven manner further complicates this problem. Capturing every conceivable noise profile in a dataset is practically impossible. To synthesise noise for training, it is necessary to create models that encode specific assumptions about the nature of the noise. This formalisation is inherently constrained, as it cannot fully encompass the diversity of noise profiles encountered in real-world scenarios. As we have already discussed, few real noise datasets exist and few cameras are used to capture them.

As a result, achieving a truly blind denoiser — capable of generalising to any noise type — is an unrealistic objective. For example, when a new camera is introduced, its noise profile, shaped by unique hardware characteristics and processing pipelines, will likely differ from those captured in existing training datasets. Consequently, any denoiser trained on prior data will struggle to generalise to this unseen noise.

This underscores the persistent advantage of training on a specific noise type or tailoring a denoiser to a particular use case. While our Wiener filter demonstrates robustness to variations in noise partly due to its explicit measurement of noise power within the filtering process, purely CNN-based approaches must carefully consider the diversity and limitations of their training data. Developing more adaptive methods or incorporating explicit noise modelling into CNN denoisers could help address this challenge.

7.2.3 The Perception-Distortion Conflict in Training Denoisers

In this thesis, we have trained our denoisers using typical objective reference metrics, such as the L1 (MAE) and L2 (MSE) loss functions. For evaluation, we have also relied on reference metrics like PSNR, which measure how closely the predicted pixels match the ground truth image.

While these metrics are effective for quantifying pixel-wise similarity, a fundamental limitation arises when the input noise reaches a level where the original image can no longer be

reliably inferred from the context provided by the network's receptive field. The prioritisation of pixel-level accuracy over perceived naturalness becomes problematic in such cases, as it often leads to over-smoothed, textureless, and blurry results. This issue is particularly evident when denoisers are trained with traditional loss functions such as L2, which penalise deviations from the ground truth even when synthesising plausible image content would result in more visually natural outputs.

Although perceptually motivated metrics, such as SSIM [297], MS-SSIM [298], and LPIPS [288], may be deployed as part of a loss function to address this limitation, they remain constrained by their dependence on the unrecoverable ground truth. For high-noise samples, where the ground truth is severely degraded or absent, the distance to the reference image becomes too large to effectively guide the network's optimisation. Consequently, any reference metric, whether perceptually based or not, still penalises the network's ability to synthesise plausible content, often resulting in outputs that lack detail and texture. This phenomenon is encapsulated by the perception-distortion trade-off [53], which formalises the paradox between achieving low distortion (e.g., high PSNR) and maintaining perceptual quality.

This insight highlights a critical design choice for future denoisers, depending on the intended application. Should the model prioritise fidelity to the ground truth by minimising distortion metrics like MSE, resulting in outputs that align more closely with traditional benchmarks but appear blurry in high-noise cases? Or should the model instead prioritise perceived naturalness by maximising the likelihood of producing a plausible natural image, $p(x)$, even at the cost of scoring poorly on conventional metrics like PSNR? In the latter case, the outputs would produce richer textures and details, scoring well in perceptual evaluations but diverging from the ground truth in pixel-wise comparisons.

This trade-off presents a compelling avenue for further research into denoiser design. Exploring adversarial loss functions, such as those used in GANs, could help balance noise reduction, image fidelity, and the generation of plausible textures, particularly for high-noise scenarios. By optimising along the perception-distortion curve, future denoisers may be better equipped to address the nuanced demands of both subjective and objective quality assessments.

7.2.4 The Difficulty in Quantifying Denoising Speed and Efficiency in High-Level Implementations

In this work, the denoising speed of our models is benchmarked by measuring the total inference time across all images in the dataset, excluding data loading. These measurements are performed within PyTorch, a Python-based deep-learning framework that facilitates the implementation of CUDA-accelerated models and CNN training. While this approach provides a consistent baseline for comparing inference times, it introduces overheads inherent to general-purpose APIs, which must accommodate a wide range of use cases. The flexibility of the Wiener-CNN model, with its unique combination of Fourier transforms, image patch unfolding/reassembly, and lightweight CNN modules, adds further overheads in software implementations.

To fully exploit the potential of the Wiener-CNN pipeline, performance could be maximised through a custom dataflow design on specialised hardware platforms such as FPGAs or ASICs. These platforms are particularly well-suited to accelerating specific features of the Wiener-CNN, such as parallel patch processing and hardware-optimised FFT operations, which could significantly reduce latency. Custom hardware would remove the general-purpose constraints of software frameworks, allowing for optimised parallelisation and resource allocation tailored to the model. Recent works on FPGA-accelerated neural networks [299] demonstrate the feasibility of such optimisations, combining quantisation and hardware-specific designs to deliver substantial improvements in speed and efficiency.

While the Wiener-CNN pipeline benefits from its relatively small parameter count compared to large CNN-based denoisers, parameter count alone does not fully capture the computational efficiency or dataflow advantages achievable through hardware implementations. A hardware-focused evaluation would provide a clearer measure of its efficiency and enable direct comparison to conventional CNN denoisers under similar conditions, accounting for both algorithmic and hardware-level optimisations.

7.3 Final Remarks

The rapidly evolving field of image and video denoising has recently seen significant advancements, with benchmarks increasingly dominated by computationally expensive and slow transformer models. In this work, we have shown that reviving the domain knowledge embedded in classic signal processing denoisers and combining it with lightweight neural network modules can approach—and in some cases, match—the performance of transform-

ers. This highlights a clear opportunity to explore the integration of other classic filters as foundational components in neural network architectures, paving the way for more efficient and practical denoising solutions.

Additionally, the recent connection between deep denoisers and the complex natural image priors they model presents a promising avenue for extending the utility of denoisers. As demonstrated in this thesis, fine-tuning denoisers for tasks such as inverse problem solving underscores their potential for broader applications and inspires future research into harnessing these implicit priors for new challenges.

References

- [1] Albert Einstein. “Über die von der molekularkinetischen Theorie der Wärme geforderte Bewegung von in ruhenden Flüssigkeiten suspendierten Teilchen”. In: *Annalen der physik* 4 (1905).
- [2] Albert Einstein. *Investigations on the Theory of the Brownian Movement*. Courier Corporation, 1956.
- [3] George A. Campbell. “Physical Theory of the Electric Wave-Filter”. In: *Bell System Technical Journal* 1.2 (1922), pp. 1–32. URL: <https://archive.org/details/bstj1-2-1>.
- [4] Otto J. Zobel. “Theory and Design of Uniform and Composite Electric Wave-Filters”. In: *Bell System Technical Journal* 2.1 (1923), pp. 1–46. URL: <https://archive.org/details/bstj2-1-1>.
- [5] Norbert Wiener et al. *Extrapolation, interpolation, and smoothing of stationary time series: with engineering applications*. Vol. 113. 21. MIT press Cambridge, MA, 1949.
- [6] Claude Elwood Shannon. “Communication in the presence of noise”. In: *Proceedings of the IRE* 37.1 (1949), pp. 10–21.
- [7] Inc. Netflix. *Netflix Q2 2023 Earnings Report*. Accessed: 2024-10-09. 2023. URL: https://s22.q4cdn.com/959853165/files/doc_financials/2023/q2/FINAL-Q2-23-Shareholder-Letter.pdf (visited on 10/09/2024).
- [8] Statista. *Most Popular Social Networks Worldwide as of January 2023*. Accessed: 2024-10-09. 2023. URL: <https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/> (visited on 10/09/2024).
- [9] Inc. Meta Platforms. *Meta Platforms Q2 2023 Earnings Release*. Accessed: 2024-10-09. 2023. URL: <https://investor.fb.com/financials/default.aspx> (visited on 10/09/2024).
- [10] TikTok. *Celebrating our Global Community of 1 Billion Users*. Accessed: 2024-10-09. 2021. URL: <https://newsroom.tiktok.com/en-us/1-billion-people-on-tiktok> (visited on 10/09/2024).
- [11] Varoun Hanooman et al. “The effect of degradation on compressibility of video”. In: *Applications of Digital Image Processing XLIV*. Vol. 11842. SPIE. 2021, pp. 277–289.
- [12] Andrey Norkin and Neil Birkbeck. “Film grain synthesis for AV1 video codec”. In: *2018 Data Compression Conference*. IEEE. 2018, pp. 3–12.
- [13] Alliance for Open Media. *AOMedia Video 1 (AV1) Specification*. Tech. rep. Accessed: 2024-04-27. Alliance for Open Media, 2020. URL: <https://aomediacodec.github.io/av1-spec/av1-spec.pdf>.

- [14] ITU-T. *Advanced Video Coding for Generic Audiovisual Services: H.264 Specification*. Tech. rep. Accessed: 2024-04-27. International Telecommunication Union, 2003. URL: <https://www.itu.int/rec/T-REC-H.264>.
- [15] Samsung Electronics. *Exynos 2400 Mobile Processor*. <https://semiconductor.samsung.com/processor/mobile-processor/exynos-2400/>. Accessed: 2024-11-26. 2024.
- [16] Google. *Google Tensor G4: The Chip Powering Pixel Smartphones*. https://store.google.com/intl/en_in/ideas/articles/google-tensor-pixel-smartphone/. Accessed: 2024-11-26. 2024.
- [17] Apple. *Apple Unveils iPhone 15 Pro and iPhone 15 Pro Max*. <https://web.archive.org/web/20230912221839/https://www.apple.com/newsroom/2023/09/apple-unveils-iphone-15-pro-and-iphone-15-pro-max/>. Accessed: 2024-11-26, Archived by the Internet Archive. 2023.
- [18] Google. *Astrophotography: How to Take Photos of the Moon and Stars*. <https://store.google.com/intl/en/ideas/articles/astrophotography-how-to-take-photos-of-the-moon-and-stars/>. Accessed: 2024-11-26. 2024.
- [19] Alexander Schiffhauer. *Night Sight gets faster and better on Pixel phones*. Oct. 2019. URL: <https://blog.google/products/pixel/night-sight-ai-faster/>.
- [20] Samsung Electronics. *Samsung Night Mode Camera: Capture Amazing Low-Light Photos*. <https://www.samsung.com/uk/mobile-phone-buying-guide/samsung-night-mode-camera/>. Accessed: 2024-11-26. 2024.
- [21] The Foundry Visionmongers Ltd. *Nuke*. Version 13. Accessed: 2024-10-09. 2023. URL: <https://www.foundry.com/products/nuke>.
- [22] Topaz Labs. *Topaz DeNoise AI*. <https://www.topazlabs.com/denoise-ai>. Accessed: 2024-11-26.
- [23] Neat Video GmbH. *Neat Video*. Version 5.23. Accessed: 2024-10-09. 2024. URL: <https://www.neatvideo.com/>.
- [24] Blackmagic Design. *DaVinci Resolve 19*. Version 19. Accessed: 2024-10-09. 2023. URL: <https://www.blackmagicdesign.com/products/davinciresolve/>.
- [25] Adobe Inc. *Adobe Premiere Pro*. Version 2024. Accessed: 2024-10-09. 2024. URL: <https://www.adobe.com/products/premiere.html>.
- [26] Kostadin Dabov et al. “BM3D image denoising with shape-adaptive principal component analysis”. In: *SPARS’09-Signal Processing with Adaptive Sparse Structured Representations*. 2009.
- [27] Anil C Kokaram. *Wiener filter for filtering noise from a video signal*. US Patent 5,500,685, filed March 19, 1996. 1996.
- [28] David L Donoho. “De-noising by soft-thresholding”. In: *IEEE transactions on information theory* 41.3 (1995), pp. 613–627.
- [29] Leonid I Rudin, Stanley Osher, and Emad Fatemi. “Nonlinear total variation based noise removal algorithms”. In: *Physica D: nonlinear phenomena* 60.1-4 (1992), pp. 259–268.
- [30] Kai Zhang et al. “Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising”. In: *IEEE Transactions on Image Processing* 26.7 (July 2017). arXiv: 1608.03981 version: 1, pp. 3142–3155. ISSN: 1057-7149, 1941-0042. DOI: 10.1109/TIP.2017.2662206. URL: <http://arxiv.org/abs/1608.03981> (visited on 11/16/2019).

- [31] Matias Tassano, Julie Delon, and Thomas Veit. “Dvdnet: A fast network for deep video denoising”. In: *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2019, pp. 1805–1809.
- [32] 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.
- [33] Pengju Liu et al. “Multi-level Wavelet-CNN for Image Restoration”. In: *arXiv:1805.07071 [cs]* (May 2018). arXiv: 1805.07071 version: 2. URL: <http://arxiv.org/abs/1805.07071> (visited on 11/19/2019).
- [34] Ze Liu et al. “Swin transformer: Hierarchical vision transformer using shifted windows”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 10012–10022.
- [35] Syed Waqas Zamir et al. “Restormer: Efficient Transformer for High-Resolution Image Restoration”. In: *arXiv preprint arXiv:2111.09881* (2021).
- [36] Jingyun Liang et al. “Vrt: A video restoration transformer”. In: *IEEE Transactions on Image Processing* (2024).
- [37] Abdelrahman Abdelhamed, Stephen Lin, and Michael S. Brown. “A High-Quality Denoising Dataset for Smartphone Cameras”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.
- [38] Tobias Plotz and Stefan Roth. “Benchmarking denoising algorithms with real photographs”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1586–1595.
- [39] Josue Anaya and Adrian Barbu. “Renoir—a dataset for real low-light image noise reduction”. In: *Journal of Visual Communication and Image Representation* 51 (2018), pp. 144–154.
- [40] 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.
- [41] Syed Waqas Zamir et al. “Cycleisp: Real image restoration via improved data synthesis”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 2696–2705.
- [42] Yue Cao et al. “Pseudo-ISP: learning pseudo in-camera signal processing pipeline from a color image denoiser”. In: *arXiv preprint arXiv:2103.10234* (2021).
- [43] Kai Zhang, Wangmeng Zuo, and Lei Zhang. “FFDNet: Toward a Fast and Flexible Solution for CNN based Image Denoising”. In: *IEEE Transactions on Image Processing* (2018).
- [44] Jingyun Liang et al. “Swinir: Image restoration using swin transformer”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 1833–1844.
- [45] Zahra Kadkhodaie and Eero P. Simoncelli. *Solving Linear Inverse Problems Using the Prior Implicit in a Denoiser*. arXiv:2007.13640 [cs, eess, stat]. May 2021. DOI: 10.48550/arXiv.2007.13640. URL: <http://arxiv.org/abs/2007.13640> (visited on 06/21/2024).
- [46] Yali Peng et al. “Progressive training of multi-level wavelet residual networks for image denoising”. In: *arXiv preprint arXiv:2010.12422* (2020).
- [47] Syed Waqas Zamir et al. “Restormer: Efficient transformer for high-resolution image restoration”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 5728–5739.

- [48] Zhendong Wang et al. “Uformer: A general u-shaped transformer for image restoration”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 17683–17693.
- [49] Jiezhong Cao et al. “Video super-resolution transformer”. In: *arXiv preprint arXiv:2106.06847* (2021).
- [50] Alexey Dosovitskiy et al. “An image is worth 16x16 words: Transformers for image recognition at scale”. In: *arXiv preprint arXiv:2010.11929* (2020).
- [51] Koichi Miyasawa et al. “An empirical Bayes estimator of the mean of a normal population”. In: *Bull. Inst. Internat. Statist* 38.181-188 (1961), pp. 1–2.
- [52] Bradley Efron. “Tweedie’s formula and selection bias”. In: *Journal of the American Statistical Association* 106.496 (2011), pp. 1602–1614.
- [53] Yochai Blau and Tomer Michaeli. “The perception-distortion tradeoff”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 6228–6237.
- [54] Clement Bled and Francois Pitie. “Assessing advances in real noise image denoisers”. In: *Proceedings of the 19th ACM SIGGRAPH European Conference on Visual Media Production*. 2022, pp. 1–9.
- [55] Clément Bled and François Pitié. “Pushing the Limits of the Wiener Filter in Image Denoising”. In: *2023 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2023, pp. 2590–2594.
- [56] Clément Bled and François Pitié. “Lightweight Video Denoising Using a Classic Bayesian Backbone”. In: *2024 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE. 2024, pp. 1–6.
- [57] John Bertrand Johnson. “Thermal agitation of electricity in conductors”. In: *Physical review* 32.1 (1928), p. 97.
- [58] Harry Nyquist. “Thermal agitation of electric charge in conductors”. In: *Physical review* 32.1 (1928), p. 110.
- [59] Claude Elwood Shannon. “A mathematical theory of communication”. In: *The Bell system technical journal* 27.3 (1948), pp. 379–423.
- [60] Norbert Wiener and John von Neumann. “Cybernetics or Control and Communication in the Animal and the Machine”. In: *Physics Today* 2.5 (1949), pp. 33–34.
- [61] Edwin H Armstrong. “A method of reducing disturbances in radio signaling by a system of frequency modulation”. In: *Proceedings of the Institute of Radio Engineers* 24.5 (1936), pp. 689–740.
- [62] Ray Dolby. “An audio noise reduction system”. In: *Journal of the audio engineering society* 15.4 (1967), pp. 383–388.
- [63] M. Berouti, R. Schwartz, and J. Makhoul. “Enhancement of speech corrupted by acoustic noise”. In: *ICASSP '79. IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 4. 1979, pp. 208–211. DOI: 10.1109/ICASSP.1979.1170788.
- [64] Glenn Kennel. “Digital film scanning and recording: The technology and practice”. In: *SMPTE journal* 103.3 (1994), pp. 174–181.
- [65] Glenn Kennel. *Color and mastering for digital cinema*. Routledge, 2012.
- [66] Peter Richard Swinson and Graham Malcolm Alexander Barber. *High resolution film scanner*. US Patent 5,644,356. 1997.
- [67] Kanta Shibata. “Japanese Film History and the Challenges of IMAGICA WEST Corp.” In: *Routledge Handbook of Japanese Cinema*. Routledge, 2020, pp. 301–313.

- [68] Oxberry. *Oxberry Official Website*. Accessed: 2024-06-19, Archived by the Internet Archive. 2014. URL: <https://web.archive.org/web/20140223072346/http://www.oxberry.com/home.html>.
- [69] Deepesh Salgia. "Mughal-e-Azam: restoration-cum-colorization for 35mm release". In: *The Moving Image* 5.1 (2005), pp. 128–135.
- [70] Los Angeles Times. "Computers Bust Dust From Disney's 'Snow White' : Film: A new Eastman Kodak computer process removes dirt from the negative. It used to be done with razor blades." In: *Los Angeles Times* (1993). Accessed: 2024-06-19, Archived by the Los Angeles Times. URL: <https://www.latimes.com/archives/la-xpm-1993-07-31-ca-18865-story.html>.
- [71] Robin Rombach et al. "High-Resolution Image Synthesis With Latent Diffusion Models". en. In: 2022, pp. 10684–10695. URL: https://openaccess.thecvf.com/content/CVPR2022/html/Rombach_High-Resolution_Image_Synthesis_With_Latent_Diffusion_Models_CVPR_2022_paper.html (visited on 06/21/2024).
- [72] Kai Zhang et al. "Learning Deep CNN Denoiser Prior for Image Restoration". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 3929–3938. URL: https://openaccess.thecvf.com/content_cvpr_2017/html/Zhang_Learning_Deep_CNN_CVPR_2017_paper.html (visited on 06/21/2024).
- [73] Jan Lukas, Jessica Fridrich, and Miroslav Goljan. "Digital camera identification from sensor pattern noise". In: *IEEE Transactions on Information Forensics and Security* 1.2 (2006), pp. 205–214.
- [74] Chang-Tsun Li. "Source camera identification using enhanced sensor pattern noise". In: *IEEE Transactions on Information Forensics and Security* 5.2 (2010), pp. 280–287.
- [75] Tomás Filler, Jessica Fridrich, and Miroslav Goljan. "Using sensor pattern noise for camera model identification". In: *2008 15th IEEE international conference on image processing*. IEEE. 2008, pp. 1296–1299.
- [76] Charles Edward Kenneth Mees. "The Theory of the Photographic Process." In: *Image* 3.8 (1954).
- [77] Jacky Chun Kit Yan. *Statistical methods for film grain noise removal and generation*. University of Toronto, 1999.
- [78] Harry C Andrews and Bobby Ray Hunt. *Digital image restoration*. Prentice Hall Professional Technical Reference, 1977.
- [79] Beaumont Newhall. "History of Photography". In: *The Museum of Modern Art* (1982).
- [80] A. E. Saunders. "Mathematical Quirks of Photographic Science". EN. In: *The Journal of Photographic Science* (Sept. 1994). Publisher: Taylor & Francis. ISSN: 0022-3638. URL: <https://www.tandfonline.com/doi/abs/10.1080/00223638.1994.11738592> (visited on 06/07/2024).
- [81] Eugene A. Trabka. "Crowded Emulsions: Granularity Theory for Monolayers*". EN. In: *JOSA* 61.6 (June 1971). Publisher: Optica Publishing Group, pp. 800–810. DOI: 10.1364/JOSA.61.000800. URL: <https://opg.optica.org/josa/abstract.cfm?uri=josa-61-6-800> (visited on 06/07/2024).
- [82] Stephen A. Benton and Richard E. Kronauer. "Properties of Granularity Wiener Spectra*". EN. In: *JOSA* 61.4 (Apr. 1971). Publisher: Optica Publishing Group, pp. 524–529. DOI: 10.1364/JOSA.61.000524. URL: <https://opg.optica.org/josa/abstract.cfm?uri=josa-61-4-524> (visited on 06/07/2024).

- [83] Leslie D Stroebe. *Basic photographic materials and processes*. 2000.
- [84] Steven M Kay. *Modern spectral estimation*. Pearson Education India, 1988.
- [85] Byung Tae Oh, Shaw-min Lei, and C-C Jay Kuo. “Advanced film grain noise extraction and synthesis for high-definition video coding”. In: *IEEE transactions on circuits and systems for video technology* 19.12 (2009), pp. 1717–1729.
- [86] Jingjing Dai et al. “Film grain noise removal and synthesis in video coding”. In: *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*. 2010, pp. 890–893. DOI: 10.1109/ICASSP.2010.5495254.
- [87] Gary K. Froehlich, John F. Walkup, and Robert B. Asher. “Optimal estimation in signal-dependent noise”. EN. In: *JOSA* 68.12 (Dec. 1978). Publisher: Optica Publishing Group, pp. 1665–1672. DOI: 10.1364/JOSA.68.001665. URL: <https://opg.optica.org/josa/abstract.cfm?uri=josa-68-12-1665> (visited on 06/07/2024).
- [88] Gary K Froehlich, John F Walkup, and Thomas F Krile. “Estimation in signal-dependent film-grain noise”. In: *Applied optics* 20.20 (1981), pp. 3619–3626.
- [89] Ioannis Pitas. “I and AN Venetsanopoulos, Nonlinear Digital Filters”. In: *Principles and Applications*. Kluwer Academic Publishers Dordrecht, 1990.
- [90] Henri H. Arsenault and M. Denis. “Image processing in signal-dependent noise”. In: *Canadian Journal of Physics* 61.2 (Feb. 1983). Publisher: NRC Research Press, pp. 309–317. ISSN: 0008-4204. DOI: 10.1139/p83-042. URL: <https://cdnsiencepub.com/doi/abs/10.1139/p83-042> (visited on 06/07/2024).
- [91] Henri H. Arsenault and Martin Levesque. “Combined homomorphic and local-statistics processing for restoration of images degraded by signal-dependent noise”. EN. In: *Applied Optics* 23.6 (Mar. 1984). Publisher: Optica Publishing Group, pp. 845–850. ISSN: 2155-3165. DOI: 10.1364/AO.23.000845. URL: <https://opg.optica.org/ao/abstract.cfm?uri=ao-23-6-845> (visited on 06/07/2024).
- [92] Antonio De Stefano, Bill Collis, and Paul White. “Synthesising and reducing film grain”. In: *Journal of Visual Communication and Image Representation* 17.1 (2006), pp. 163–182.
- [93] Jingjing Dai et al. “Film grain noise removal and synthesis in video coding”. In: *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE. 2010, pp. 890–893.
- [94] Anil C Kokaram. “On missing data treatment for degraded video and film archives: a survey and a new Bayesian approach”. In: *IEEE Transactions on Image Processing* 13.3 (2004), pp. 397–415.
- [95] Alliance for Open Media. *AV1 Codec source code repository*. <https://aomedia.googlesource.com/aom>. Accessed: 2024-05-21.
- [96] Gilbert F Amelio, Michael F Tompsett, and George E Smith. “Experimental verification of the charge coupled device concept”. In: *Bell System Technical Journal* 49.4 (1970), pp. 593–600.
- [97] Willard S Boyle and George E Smith. “Charge coupled semiconductor devices”. In: *Bell System Technical Journal* 49.4 (1970), pp. 587–593.
- [98] M. Bigas et al. “Review of CMOS image sensors”. en. In: *Microelectronics Journal* 37.5 (May 2006), pp. 433–451. ISSN: 00262692. DOI: 10.1016/j.mejo.2005.07.002. URL: <https://linkinghub.elsevier.com/retrieve/pii/S0026269205002764> (visited on 06/11/2024).
- [99] Gershon Buchsbaum. “A spatial processor model for object colour perception”. In: *Journal of the Franklin institute* 310.1 (1980), pp. 1–26.

- [100] Edwin H Land and John J McCann. “Lightness and retinex theory”. In: *Josa* 61.1 (1971), pp. 1–11.
- [101] Yuanming Hu, Baoyuan Wang, and Stephen Lin. “Fc4: Fully convolutional color constancy with confidence-weighted pooling”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 4085–4094.
- [102] Zhongyu Lou et al. “Color Constancy by Deep Learning.” In: *BMVC*. 2015, pp. 76–1.
- [103] Jonathan T Barron. “Convolutional color constancy”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 379–387.
- [104] Hui Tian, Boyd A. Fowler, and Abbas El Gamal. “Analysis of temporal noise in CMOS APS”. In: *Sensors, Cameras, and Systems for Scientific/Industrial Applications*. Vol. 3649. SPIE, Apr. 1999, pp. 177–185. DOI: 10.1117/12.347073. URL: <https://www.spiedigitallibrary.org/conference-proceedings-of-spie/3649/0000/Analysis-of-temporal-noise-in-CMOS-APS/10.1117/12.347073.full> (visited on 06/08/2024).
- [105] SukHwan Lim. “Characterization of noise in digital photographs for image processing”. In: *Digital Photography II*. Vol. 6069. SPIE. 2006, pp. 219–228.
- [106] David X. D. Yang and Abbas El Gamal. “Comparative analysis of SNR for image sensors with enhanced dynamic range”. In: *Sensors, Cameras, and Systems for Scientific/Industrial Applications*. Vol. 3649. SPIE, Apr. 1999, pp. 197–211. DOI: 10.1117/12.347075. URL: <https://www.spiedigitallibrary.org/conference-proceedings-of-spie/3649/0000/Comparative-analysis-of-SNR-for-image-sensors-with-enhanced-dynamic/10.1117/12.347075.full> (visited on 06/08/2024).
- [107] Samuel W Hasinoff, Frédo Durand, and William T Freeman. “Noise-optimal capture for high dynamic range photography”. In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE. 2010, pp. 553–560.
- [108] Carlo Beenakker and Christian Schönenberger. “Quantum shot noise”. In: *Physics Today* 56.5 (2003), pp. 37–42.
- [109] Walter Schottky. “Über spontane Stromschwankungen in verschiedenen Elektrizitätsleitern”. In: *Annalen der physik* 362.23 (1918), pp. 541–567.
- [110] Walter Schottky. “On spontaneous current fluctuations in various electrical conductors”. In: *Journal of Micro/Nanolithography, MEMS, and MOEMS* 17.4 (2018), pp. 041001–041001.
- [111] Mitsuhiro Mase et al. “A wide dynamic range CMOS image sensor with multiple exposure-time signal outputs and 12-bit column-parallel cyclic A/D converters”. In: *IEEE Journal of Solid-State Circuits* 40.12 (2005), pp. 2787–2795.
- [112] Xinqiao Liu and Abbas El Gamal. “Photocurrent estimation from multiple nondestructive samples in CMOS image sensor”. In: *Sensors and Camera Systems for Scientific, Industrial, and Digital Photography Applications II*. Vol. 4306. SPIE, May 2001, pp. 450–458. DOI: 10.1117/12.426983. URL: <https://www.spiedigitallibrary.org/conference-proceedings-of-spie/4306/0000/Photocurrent-estimation-from-multiple-nondestructive-samples-in-CMOS-image-sensor/10.1117/12.426983.full> (visited on 06/08/2024).
- [113] Arnaud Barral. “Fixed Pattern Noise Reduction: Temporal High Pass Filter”. In: (2021).
- [114] Dileepan Joseph and Steve Collins. “Modelling, calibration and correction of nonlinear illumination dependent fixed pattern noise in logarithmic CMOS image sensors”. In: *IMTC 2001. Proceedings of the 18th IEEE Instrumentation and Measurement Technology Conference. Rediscovering Measurement in the Age of Informatics (Cat. No. 01CH 37188)*. Vol. 2. IEEE. 2001, pp. 1296–1301.

- [115] Lionel J D'Luna and Kenneth A Parulski. "A systems approach to custom VLSI for a digital color imaging system". In: *IEEE journal of solid-state circuits* 26.5 (1991), pp. 727–737.
- [116] Shin-Shu Wang and Ye-Quang Chen. "New color image processor for the video camera". In: *Visual Communications and Image Processing '95*. Vol. 2501. SPIE, Apr. 1995, pp. 931–936. DOI: 10.1117/12.206799. URL: <https://www.spiedigitallibrary.org/conference-proceedings-of-spie/2501/0000/New-color-image-processor-for-the-video-camera/10.1117/12.206799.full> (visited on 06/08/2024).
- [117] Wen-Hsin Chan and Ching-Twu Youe. "Video CCD based portable digital still camera". In: *IEEE transactions on consumer electronics* 41.3 (1995), pp. 455–459.
- [118] Wen-Hsin Chan and Chi-Fu Chou. "Megapixel resolution PC digital still camera". In: *Solid State Sensor Arrays and CCD Cameras*. Vol. 2654. SPIE, Mar. 1996, pp. 164–171. DOI: 10.1117/12.236095. URL: <https://www.spiedigitallibrary.org/conference-proceedings-of-spie/2654/0000/Megapixel-resolution-PC-digital-still-camera/10.1117/12.236095.full> (visited on 06/08/2024).
- [119] Eastman Kodak Company. *Kodak Lossless True Color Image Suite*. <http://r0k.us/graphics/kodak/>. Available online: <http://r0k.us/graphics/kodak/> (accessed on [your access date]). 1993.
- [120] JW Tukey. "Nonlinear (nonsuperposable) methods for smoothing data". In: *Proc. Cong. Rec. EASCOM'74* (1974), pp. 673–681.
- [121] Carlo Tomasi and Roberto Manduchi. "Bilateral filtering for gray and color images". In: *Sixth international conference on computer vision (IEEE Cat. No. 98CH36271)*. IEEE. 1998, pp. 839–846.
- [122] William K Pratt. "Generalized Wiener filtering computation techniques". In: *IEEE Transactions on Computers* 100.7 (1972), pp. 636–641.
- [123] Eero P Simoncelli and William T Freeman. "The steerable pyramid: A flexible architecture for multi-scale derivative computation". In: *Proceedings., International Conference on Image Processing*. Vol. 3. IEEE. 1995, pp. 444–447.
- [124] Antonin Chambolle. "An algorithm for total variation minimization and applications". In: *Journal of Mathematical imaging and vision* 20 (2004), pp. 89–97.
- [125] 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.
- [126] Kostadin Dabov et al. "Image denoising by sparse 3-D transform-domain collaborative filtering". In: *IEEE Transactions on image processing* 16.8 (2007), pp. 2080–2095.
- [127] Javier Gurrola-Ramos, Oscar Dalmau, and Teresa E Alarcón. "A residual dense u-net neural network for image denoising". In: *IEEE Access* 9 (2021), pp. 31742–31754.
- [128] Rafael C Gonzalez. *Digital image processing*. Pearson education india, 2009.
- [129] Ioannis Pitas and Anastasios N. Venetsanopoulos. *Nonlinear Digital Filters: Principles and Applications*. en. Google-Books-ID: zWfmBwAAQBAJ. Springer Science & Business Media, Mar. 2013. ISBN: 978-1-4757-6017-0.
- [130] Edward R Dougherty and Jaakko Astola. *An introduction to nonlinear image processing*. Vol. 16. SPIE press, 1994.
- [131] Humor Hwang and Richard A Haddad. "Adaptive median filters: new algorithms and results". In: *IEEE Transactions on image processing* 4.4 (1995), pp. 499–502.
- [132] David RK Brownrigg. "The weighted median filter". In: *Communications of the ACM* 27.8 (1984), pp. 807–818.

- [133] Michael A King et al. "A Wiener filter for nuclear medicine images". In: *Medical physics* 10.6 (1983), pp. 876–880.
- [134] Saleem A Kassam and Tong Leong Lim. "Robust wiener filters". In: *Journal of the Franklin Institute* 304.4-5 (1977), pp. 171–185.
- [135] Are Hjørungnes. *Complex-valued matrix derivatives: with applications in signal processing and communications*. Cambridge University Press, 2011.
- [136] Anil K Jain. *Fundamentals of digital image processing*. Prentice-Hall, Inc., 1989.
- [137] Jacob Benesty, Jingdong Chen, and Yiteng Huang. "Study of the widely linear Wiener filter for noise reduction". In: *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE. 2010, pp. 205–208.
- [138] AC Kokaram. "3D Wiener filtering for noise suppression in motion picture sequences using overlapped processing". In: *Signal Processing VII, Vol3* (1994), pp. 1780–1783.
- [139] Fabien Dekeyser, Patrick Bouthemy, and Patrick Pérez. "Spatio-temporal Wiener filtering of image sequences using a parametric motion model". In: *Proceedings 2000 International Conference on Image Processing (Cat. No. 00CH37101)*. Vol. 1. IEEE. 2000, pp. 208–211.
- [140] Edwin Hewitt and Robert E Hewitt. "The Gibbs-Wilbraham phenomenon: an episode in Fourier analysis". In: *Archive for history of Exact Sciences* (1979), pp. 129–160.
- [141] Eero P Simoncelli and Edward H Adelson. "Noise removal via Bayesian wavelet coring". In: *Proceedings of 3rd IEEE International Conference on Image Processing*. Vol. 1. IEEE. 1996, pp. 379–382.
- [142] Jean-Marc Lina. "Image processing with complex Daubechies wavelets". In: *Journal of Mathematical Imaging and Vision* 7.3 (1997), pp. 211–223.
- [143] Abhinav Dixit and Swatilekha Majumdar. "Comparative analysis of coiflet and daubechies wavelets using global threshold for image denoising". In: *International Journal of Advances in Engineering & Technology* 6.5 (2013), p. 2247.
- [144] Ashwani Kumar Yadav et al. "De-noising of ultrasound image using discrete wavelet transform by symlet wavelet and filters". In: *2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE. 2015, pp. 1204–1208.
- [145] Arun Dixit and Poonam Sharma. "A comparative study of wavelet thresholding for image denoising". In: *IJ Image, Graphics and Signal Processing* 12 (2014), pp. 39–46.
- [146] Charles M Stein. "Estimation of the mean of a multivariate normal distribution". In: *The annals of Statistics* (1981), pp. 1135–1151.
- [147] Florian Luisier, Thierry Blu, and Michael Unser. "A new SURE approach to image denoising: Interscale orthonormal wavelet thresholding". In: *IEEE Transactions on image processing* 16.3 (2007), pp. 593–606.
- [148] Felix Abramovich, Theofanis Sapatinas, and Bernard W Silverman. "Wavelet thresholding via a Bayesian approach". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 60.4 (1998), pp. 725–749.
- [149] Javier Portilla et al. "Image denoising using scale mixtures of Gaussians in the wavelet domain". In: *IEEE Transactions on Image processing* 12.11 (2003), pp. 1338–1351.
- [150] Nuke | VFX and Film Editing Software. en. URL: <https://www.foundry.com/products/nuke-family/nuke> (visited on 05/10/2022).
- [151] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. "Non-local means denoising". In: *Image Processing On Line* 1 (2011), pp. 208–212.

- [152] Jin Wang et al. “Fast non-local algorithm for image denoising”. In: *2006 International Conference on Image Processing*. IEEE. 2006, pp. 1429–1432.
- [153] Bo Chong and Yong-Kai Zhu. “Speckle reduction in optical coherence tomography images of human finger skin by wavelet modified BM3D filter”. In: *Optics Communications* 291 (2013), pp. 461–469.
- [154] Marc Lebrun. “An analysis and implementation of the BM3D image denoising method”. In: *Image Processing On Line* 2012 (2012), pp. 175–213.
- [155] Matteo Maggioni et al. “Nonlocal transform-domain filter for volumetric data denoising and reconstruction”. In: *IEEE transactions on image processing* 22.1 (2012), pp. 119–133.
- [156] Andrei Nikolaevich Tikhonov. “On the solution of ill-posed problems and the method of regularization”. In: *Doklady akademii nauk*. Vol. 151. 3. Russian Academy of Sciences. 1963, pp. 501–504.
- [157] Per Christian Hansen. *Rank-deficient and discrete ill-posed problems: numerical aspects of linear inversion*. SIAM, 1998.
- [158] Ralph A Willoughby. “Solutions of ill-posed problems (AN Tikhonov and VY Arsenin)”. In: *SIAM Review* 21.2 (1979), p. 266.
- [159] Magnus Rudolph Hestenes, Eduard Stiefel, et al. *Methods of conjugate gradients for solving linear systems*. Vol. 49. 1. NBS Washington, DC, 1952.
- [160] Jonathan Richard Shewchuk et al. “An introduction to the conjugate gradient method without the agonizing pain”. In: (1994).
- [161] Youcef Saad and Martin H Schultz. “GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems”. In: *SIAM Journal on scientific and statistical computing* 7.3 (1986), pp. 856–869.
- [162] Yousef Saad. *Iterative methods for sparse linear systems*. SIAM, 2003.
- [163] Stephen Boyd et al. “Distributed optimization and statistical learning via the alternating direction method of multipliers”. In: *Foundations and Trends® in Machine learning* 3.1 (2011), pp. 1–122.
- [164] Tom Goldstein and Stanley Osher. “The split Bregman method for L1-regularized problems”. In: *SIAM journal on imaging sciences* 2.2 (2009), pp. 323–343.
- [165] Leonid I Rudin and Stanley Osher. “Total variation based image restoration with free local constraints”. In: *Proceedings of 1st international conference on image processing*. Vol. 1. IEEE. 1994, pp. 31–35.
- [166] Curtis R Vogel and Mary E Oman. “Iterative methods for total variation denoising”. In: *SIAM Journal on Scientific Computing* 17.1 (1996), pp. 227–238.
- [167] Amir Beck and Marc Teboulle. “Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems”. In: *IEEE transactions on image processing* 18.11 (2009), pp. 2419–2434.
- [168] Yue Hu and Mathews Jacob. “Higher degree total variation (HDTV) regularization for image recovery”. In: *IEEE Transactions on Image Processing* 21.5 (2012), pp. 2559–2571.
- [169] Yifei Lou et al. “A weighted difference of anisotropic and isotropic total variation model for image processing”. In: *SIAM Journal on Imaging Sciences* 8.3 (2015), pp. 1798–1823.
- [170] Guy Gilboa and Stanley Osher. “Nonlocal operators with applications to image processing”. In: *Multiscale Modeling & Simulation* 7.3 (2009), pp. 1005–1028.

- [171] Mona Mahmoudi and Guillermo Sapiro. “Fast image and video denoising via nonlocal means of similar neighborhoods”. In: *IEEE signal processing letters* 12.12 (2005), pp. 839–842.
- [172] Pierrick Coupé et al. “An optimized blockwise nonlocal means denoising filter for 3-D magnetic resonance images”. In: *IEEE transactions on medical imaging* 27.4 (2008), pp. 425–441.
- [173] Tanaphol Thaipanich et al. “Improved image denoising with adaptive nonlocal means (ANL-means) algorithm”. In: *IEEE Transactions on Consumer Electronics* 56.4 (2010), pp. 2623–2630.
- [174] Bart Goossens et al. “An improved non-local denoising algorithm”. In: *2008 International Workshop on Local and Non-Local Approximation in Image Processing (LNLA 2008)*. 2008, pp. 143–156.
- [175] Chao Pang et al. “A fast NL-means method in image denoising based on the similarity of spatially sampled pixels”. In: *2009 IEEE International Workshop on Multimedia Signal Processing*. IEEE. 2009, pp. 1–4.
- [176] Linwei Fan et al. “Nonlocal image denoising using edge-based similarity metric and adaptive parameter selection.” In: *Sci. China Inf. Sci.* 61.4 (2018), pp. 049101–1.
- [177] Sven Grewenig, Sebastian Zimmer, and Joachim Weickert. “Rotationally invariant similarity measures for nonlocal image denoising”. In: *Journal of Visual Communication and Image Representation* 22.2 (2011), pp. 117–130.
- [178] David Tschumperlé and Luc Brun. “Non-local image smoothing by applying anisotropic diffusion PDE’s in the space of patches”. In: *2009 16th IEEE International Conference on Image Processing (ICIP)*. IEEE. 2009, pp. 2957–2960.
- [179] Michal Aharon, Michael Elad, and Alfred Bruckstein. “K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation”. In: *IEEE Transactions on signal processing* 54.11 (2006), pp. 4311–4322.
- [180] 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.
- [181] Weisheng Dong, Guangming Shi, and Xin Li. “Nonlocal image restoration with bilateral variance estimation: a low-rank approach”. In: *IEEE transactions on image processing* 22.2 (2012), pp. 700–711.
- [182] Ivan Markovsky. *Low rank approximation: algorithms, implementation, applications*. Vol. 906. Springer, 2012.
- [183] Hui Ji et al. “Robust video denoising using low rank matrix completion”. In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE. 2010, pp. 1791–1798.
- [184] Hui Ji et al. “Robust video restoration by joint sparse and low rank matrix approximation”. In: *SIAM Journal on Imaging Sciences* 4.4 (2011), pp. 1122–1142.
- [185] Jian-Feng Cai, Emmanuel J Candès, and Zuowei Shen. “A singular value thresholding algorithm for matrix completion”. In: *SIAM Journal on optimization* 20.4 (2010), pp. 1956–1982.
- [186] Shuhang Gu et al. “Weighted nuclear norm minimization with application to image denoising”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 2862–2869.
- [187] Shuhang Gu et al. “Weighted nuclear norm minimization and its applications to low level vision”. In: *International journal of computer vision* 121.2 (2017), pp. 183–208.

- [188] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems* 25 (2012).
- [189] Karen Simonyan and Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *arXiv:1409.1556 [cs]* (Apr. 2015). arXiv: 1409.1556. URL: <http://arxiv.org/abs/1409.1556> (visited on 11/12/2019).
- [190] Christian Szegedy et al. “Inception-v4, inception-resnet and the impact of residual connections on learning”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 31. 1. 2017.
- [191] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: 2016, pp. 770–778. URL: http://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html (visited on 11/12/2019).
- [192] Chao Dong et al. “Learning a deep convolutional network for image super-resolution”. In: *Computer Vision—ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part IV 13*. Springer. 2014, pp. 184–199.
- [193] Chao Dong et al. “Image super-resolution using deep convolutional networks”. In: *IEEE transactions on pattern analysis and machine intelligence* 38.2 (2015), pp. 295–307.
- [194] Michaël Gharbi et al. “Deep joint demosaicking and denoising”. en. In: *ACM Transactions on Graphics* 35.6 (Nov. 2016), pp. 1–12. ISSN: 07300301. DOI: 10.1145/2980179.2982399. URL: <http://dl.acm.org/citation.cfm?doid=2980179.2982399> (visited on 11/20/2019).
- [195] Harold C Burger, Christian J Schuler, and Stefan Harmeling. “Image denoising: Can plain neural networks compete with BM3D?” In: *2012 IEEE conference on computer vision and pattern recognition*. IEEE. 2012, pp. 2392–2399.
- [196] Eirikur Agustsson and Radu Timofte. “Ntire 2017 challenge on single image super-resolution: Dataset and study”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2017, pp. 126–135.
- [197] Vinod Nair and Geoffrey E Hinton. “Rectified linear units improve restricted boltzmann machines”. In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010, pp. 807–814.
- [198] Sreyas Mohan et al. “Robust and interpretable blind image denoising via bias-free convolutional neural networks”. In: *arXiv preprint arXiv:1906.05478* (2019).
- [199] Yulun Zhang et al. “Residual Dense Network for Image Restoration”. en. In: *arXiv:1812.10477 [cs]* (Dec. 2018). arXiv: 1812.10477. URL: <http://arxiv.org/abs/1812.10477> (visited on 11/19/2019).
- [200] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. en. In: *arXiv:1505.04597 [cs]* (May 2015). arXiv: 1505.04597. URL: <http://arxiv.org/abs/1505.04597> (visited on 11/12/2019).
- [201] Mattias P. Heinrich, Maik Stille, and Thorsten M. Buzug. “Residual U-Net Convolutional Neural Network Architecture for Low-Dose CT Denoising”. en. In: *Current Directions in Biomedical Engineering* 4.1 (Sept. 2018), pp. 297–300. ISSN: 2364-5504. DOI: 10.1515/cdbme-2018-0072. URL: <http://www.degruyter.com/view/j/cdbme.2018.4.issue-1/cdbme-2018-0072/cdbme-2018-0072.xml> (visited on 11/12/2019).
- [202] Pengju Liu et al. “Multi-Level Wavelet Convolutional Neural Networks”. In: *IEEE Access* 7 (2019), pp. 74973–74985.

- [203] Shuo-Fei Wang, Wen-Kai Yu, and Ya-Xin Li. “Multi-wavelet residual dense convolutional neural network for image denoising”. In: *IEEE Access* 8 (2020), pp. 214413–214424.
- [204] Chi-Mao Fan, Tsung-Jung Liu, and Kuan-Hsien Liu. “SUNet: Swin transformer UNet for image denoising”. In: *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE. 2022, pp. 2333–2337.
- [205] Yiyun Zhao et al. “Pyramid real image denoising network”. In: *2019 IEEE Visual Communications and Image Processing (VCIP)*. IEEE. 2019, pp. 1–4.
- [206] Zongsheng Yue et al. “Variational denoising network: Toward blind noise modeling and removal”. In: *Advances in neural information processing systems* 32 (2019).
- [207] Chunwei Tian et al. “Attention-guided CNN for image denoising”. In: *Neural Networks* 124 (2020), pp. 117–129.
- [208] Syed Waqas Zamir et al. “Multi-stage progressive image restoration”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 14821–14831.
- [209] Xuebin Qin et al. “U2-Net: Going deeper with nested U-structure for salient object detection”. In: *Pattern recognition* 106 (2020), p. 107404.
- [210] Abdelrahman Abdelhamed, Stephen Lin, and Michael S Brown. “A high-quality denoising dataset for smartphone cameras”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 1692–1700.
- [211] 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. 2001, pp. 416–423.
- [212] Kede Ma et al. “Waterloo Exploration Database: New Challenges for Image Quality Assessment Models”. In: *IEEE Transactions on Image Processing* 26.2 (2017), pp. 1004–1016.
- [213] Vladimir Bychkovsky et al. “Learning Photographic Global Tonal Adjustment with a Database of Input / Output Image Pairs”. In: *The Twenty-Fourth IEEE Conference on Computer Vision and Pattern Recognition*. 2011.
- [214] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).
- [215] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. “Neural machine translation by jointly learning to align and translate”. In: *arXiv preprint arXiv:1409.0473* (2014).
- [216] Ozan Oktay et al. “Attention u-net: Learning where to look for the pancreas”. In: *arXiv preprint arXiv:1804.03999* (2018).
- [217] Syed Waqas Zamir et al. “Learning enriched features for fast image restoration and enhancement”. In: *IEEE transactions on pattern analysis and machine intelligence* 45.2 (2022), pp. 1934–1948.
- [218] Saeed Anwar and Nick Barnes. “Real image denoising with feature attention”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 3155–3164.
- [219] Hanting Chen et al. “Pre-trained image processing transformer”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 12299–12310.
- [220] Fu Jin, Paul Fieguth, and Lowell Winger. “Wavelet video denoising with regularized multiresolution motion estimation”. In: *EURASIP Journal on Advances in Signal Processing* 2006 (2006), pp. 1–11.

- [221] Vladimir Zlokolica, Aleksandra Pizurica, and Wilfried Philips. “Wavelet-domain video denoising based on reliability measures”. In: *IEEE Transactions on circuits and systems for video technology* 16.8 (2006), pp. 993–1007.
- [222] Anil C Kokaram. *Motion picture restoration: digital algorithms for artefact suppression in degraded motion picture film and video*. Springer Science & Business Media, 2013.
- [223] 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. 1354–1363.
- [224] Michele Claus. “ViDeNN: Deep Blind Video Denoising”. en. In: (), p. 10.
- [225] Axel Davy et al. “Non-local video denoising by CNN”. In: *arXiv preprint arXiv:1811.12758* (2018).
- [226] Gregory Vaksman, Michael Elad, and Peyman Milanfar. “Patch craft: Video denoising by deep modeling and patch matching”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 2157–2166.
- [227] Chengxu Liu et al. “Learning trajectory-aware transformer for video super-resolution”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 5687–5696.
- [228] Mingyang Song, Yang Zhang, and Tunç O Aydın. “Tempformer: Temporally consistent transformer for video denoising”. In: *European Conference on Computer Vision*. Springer. 2022, pp. 481–496.
- [229] Meiqin Liu et al. “Temporal consistency learning of inter-frames for video super-resolution”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 33.4 (2022), pp. 1507–1520.
- [230] Dasong Li et al. “A Simple Baseline for Video Restoration with Grouped Spatial-Temporal Shift”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2023, pp. 9822–9832.
- [231] Kun Zhou et al. “Exploring motion ambiguity and alignment for high-quality video frame interpolation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 22169–22179.
- [232] Wenhui Zhang et al. “Cross-frame transformer-based spatio-temporal video super-resolution”. In: *IEEE Transactions on Broadcasting* 68.2 (2022), pp. 359–369.
- [233] Zhongwei Qiu et al. “Learning spatiotemporal frequency-transformer for compressed video super-resolution”. In: *European Conference on Computer Vision*. Springer. 2022, pp. 257–273.
- [234] Kelvin CK Chan et al. “Basicvsr++: Improving video super-resolution with enhanced propagation and alignment”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 5972–5981.
- [235] Jifeng Dai et al. “Deformable convolutional networks”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 764–773.
- [236] M.D. Grossberg and S.K. Nayar. “Modeling the space of camera response functions”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26.10 (2004), pp. 1272–1282. DOI: 10.1109/TPAMI.2004.88.
- [237] Yue Cao et al. “Pseudo-ISP: learning pseudo in-camera signal processing pipeline from a color image denoiser”. In: *Neurocomputing* (2024), p. 128316.
- [238] Jascha Sohl-Dickstein et al. “Deep unsupervised learning using nonequilibrium thermodynamics”. In: *International conference on machine learning*. PMLR. 2015, pp. 2256–2265.

- [239] Jonathan Ho, Ajay Jain, and Pieter Abbeel. “Denoising diffusion probabilistic models”. In: *Advances in neural information processing systems* 33 (2020), pp. 6840–6851.
- [240] Prafulla Dhariwal and Alexander Nichol. “Diffusion models beat gans on image synthesis”. In: *Advances in neural information processing systems* 34 (2021), pp. 8780–8794.
- [241] Alexander Quinn Nichol and Prafulla Dhariwal. “Improved denoising diffusion probabilistic models”. In: *International conference on machine learning*. PMLR. 2021, pp. 8162–8171.
- [242] Martin Heusel et al. “Gans trained by a two time-scale update rule converge to a local nash equilibrium”. In: *Advances in neural information processing systems* 30 (2017).
- [243] Aditya Ramesh et al. “Zero-Shot Text-to-Image Generation”. In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, July 2021, pp. 8821–8831. URL: <https://proceedings.mlr.press/v139/ramesh21a.html>.
- [244] Alex Nichol et al. “Glide: Towards photorealistic image generation and editing with text-guided diffusion models”. In: *arXiv preprint arXiv:2112.10741* (2021).
- [245] James Betker et al. “Improving image generation with better captions”. In: *Computer Science*. <https://cdn.openai.com/papers/dall-e-3.pdf> 2.3 (2023), p. 8.
- [246] Su Wang et al. “Imagen editor and editbench: Advancing and evaluating text-guided image inpainting”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2023, pp. 18359–18369.
- [247] Chitwan Saharia et al. “Photorealistic text-to-image diffusion models with deep language understanding”. In: *Advances in neural information processing systems* 35 (2022), pp. 36479–36494.
- [248] Singanallur V Venkatakrishnan, Charles A Bouman, and Brendt Wohlberg. “Plug-and-play priors for model based reconstruction”. In: *2013 IEEE global conference on signal and information processing*. IEEE. 2013, pp. 945–948.
- [249] Alon Brifman, Yaniv Romano, and Michael Elad. “Turning a denoiser into a super-resolver using plug and play priors”. In: *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2016, pp. 1404–1408.
- [250] Kai Zhang et al. “Plug-and-play image restoration with deep denoiser prior”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.10 (2021), pp. 6360–6376.
- [251] Chi-Mao Fan et al. “Selective residual m-net for real image denoising”. In: *2022 30th European Signal Processing Conference (EUSIPCO)*. IEEE. 2022, pp. 469–473.
- [252] Yuda Song, Yunfang Zhu, and Xin Du. “Grouped multi-scale network for real-world image denoising”. In: *IEEE Signal Processing Letters* 27 (2020), pp. 2124–2128.
- [253] Syed Waqas Zamir et al. “Learning enriched features for real image restoration and enhancement”. In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXV 16*. Springer. 2020, pp. 492–511.
- [254] Zongsheng Yue et al. “Dual adversarial network: Toward real-world noise removal and noise generation”. In: *European Conference on Computer Vision*. Springer. 2020, pp. 41–58.
- [255] Yoonsik Kim et al. “Transfer learning from synthetic to real-noise denoising with adaptive instance normalization”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 3482–3492.

- [256] Shen Cheng et al. “Nbnet: Noise basis learning for image denoising with subspace projection”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 4896–4906.
- [257] Liangyu Chen et al. “Simple baselines for image restoration”. In: *European conference on computer vision*. Springer. 2022, pp. 17–33.
- [258] Yi Zhang et al. “Kbnet: Kernel basis network for image restoration”. In: *arXiv preprint arXiv:2303.02881* (2023).
- [259] Amirhosein Ghasemabadi et al. “CascadedGaze: Efficiency in Global Context Extraction for Image Restoration”. In: *arXiv preprint arXiv:2401.15235* (2024).
- [260] Chen Chen et al. “Learning to see in the dark”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 3291–3300.
- [261] Dong-Wook Kim, Jae Ryun Chung, and Seung-Won Jung. “Grdn: Grouped residual dense network for real image denoising and gan-based real-world noise modeling”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*. 2019.
- [262] Yu Jin et al. “Channel estimation for mmWave massive MIMO with convolutional blind denoising network”. In: *IEEE Communications Letters* 24.1 (2019), pp. 95–98.
- [263] Saining Xie et al. “Aggregated residual transformations for deep neural networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1492–1500.
- [264] Jae S Lim. “Two-dimensional signal and image processing”. In: *Englewood Cliffs* (1990).
- [265] MN Sanders. “Characteristic function of the central chi-squared distribution”. In: *Planet mathematics*. Retrieved March 6 (2009).
- [266] Anil Christopher Kokaram. “Motion picture restoration”. PhD thesis. University of Cambridge, 1993.
- [267] Soeren I. Olsen. “Estimation of noise in images: An evaluation”. In: *CVGIP: Graphical Models and Image Processing* 55.4 (1993), pp. 319–323.
- [268] Klaus Rank, Markus Lendl, and Rolf Unbehauen. “Estimation of image noise variance”. In: *IEE Proceedings-Vision, Image and Signal Processing* 146.2 (1999), pp. 80–84.
- [269] John Immerkaer. “Fast noise variance estimation”. In: *Computer vision and image understanding* 64.2 (1996), pp. 300–302.
- [270] JS Lee and K Hoppel. “Noise modeling and estimation of remotely-sensed images”. In: *12th Canadian Symposium on Remote Sensing Geoscience and Remote Sensing Symposium*, vol. 2. IEEE. 1989, pp. 1005–1008.
- [271] Dong-Hyuk Shin et al. “Block-based noise estimation using adaptive Gaussian filtering”. In: *IEEE Transactions on Consumer Electronics* 51.1 (2005), pp. 218–226.
- [272] Anil Kokaram et al. “Measuring noise correlation for improved video denoising”. In: *2012 19th IEEE International Conference on Image Processing*. IEEE. 2012, pp. 1201–1204.
- [273] 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*. 2017.
- [274] Kai Zhang et al. “Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising”. In: *IEEE transactions on image processing* 26.7 (2017), pp. 3142–3155.

- [275] Huanjing Yue et al. “Supervised raw video denoising with a benchmark dataset on dynamic scenes”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 2301–2310.
- [276] Yi Zhang et al. “Rethinking noise synthesis and modeling in raw denoising”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 4593–4601.
- [277] Javier Sánchez Pérez, Enric Meinhardt-Llopis, and Gabriele Facciolo. “TV-L1 Optical Flow Estimation”. In: *Image Processing On Line* 3 (2013). <https://doi.org/10.5201/ipol.2013.26>, pp. 137–150.
- [278] 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.
- [279] 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.
- [280] Jiezhong Cao et al. “Learning Task-Oriented Flows to Mutually Guide Feature Alignment in Synthesized and Real Video Denoising”. In: *arXiv preprint arXiv:2208.11803* (2022).
- [281] Zachary Teed and Jia Deng. “Raft: Recurrent all-pairs field transforms for optical flow”. In: *Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II* 16. Springer. 2020, pp. 402–419.
- [282] Christopher Montgomery and H Lars. “Xiph. org video test media (derf’s collection)”. In: *Online*, <https://media.xiph.org/video/derf> 6 (1994).
- [283] Di Ma, Fan Zhang, and David R Bull. “BVI-DVC: A training database for deep video compression”. In: *IEEE Transactions on Multimedia* 24 (2021), pp. 3847–3858.
- [284] Angeliki V Katsenou et al. “BVI-SynTex: A synthetic video texture dataset for video compression and quality assessment”. In: *IEEE Transactions on Multimedia* 23 (2020), pp. 26–38.
- [285] Ilya Loshchilov, Frank Hutter, et al. “Fixing weight decay regularization in adam”. In: *arXiv preprint arXiv:1711.05101* 5 (2017).
- [286] Jingyun Liang et al. “Vrt: A video restoration transformer”. In: *arXiv preprint arXiv:2201.12288* (2022).
- [287] Federico Perazzi et al. “A benchmark dataset and evaluation methodology for video object segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 724–732.
- [288] Richard Zhang et al. “The unreasonable effectiveness of deep features as a perceptual metric”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 586–595.
- [289] Netflix. *VMAF: Video Multi-Method Assessment Fusion*. Accessed: 2024-11-26. 2023. URL: <https://github.com/Netflix/vmaf>.
- [290] Herbert E Robbins. “An empirical Bayes approach to statistics”. In: *Breakthroughs in Statistics: Foundations and basic theory*. Springer, 1992, pp. 388–394.
- [291] Michael JD Powell. “A method for nonlinear constraints in minimization problems”. In: *Optimization* (1969), pp. 283–298.
- [292] Yaniv Romano, Michael Elad, and Peyman Milanfar. “The little engine that could: Regularization by denoising (RED)”. In: *SIAM Journal on Imaging Sciences* 10.4 (2017), pp. 1804–1844.

- [293] Yawei Li et al. “Lsdir: A large scale dataset for image restoration”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 1775–1787.
- [294] David Martin et al. “A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics”. In: *Proceedings eighth IEEE international conference on computer vision. ICCV 2001*. Vol. 2. IEEE. 2001, pp. 416–423.
- [295] Wenzhe Shi et al. “Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 1874–1883.
- [296] Dan Hendrycks and Kevin Gimpel. “Gaussian error linear units (gelus)”. In: *arXiv preprint arXiv:1606.08415* (2016).
- [297] Zhou Wang et al. “Image quality assessment: from error visibility to structural similarity”. In: *IEEE transactions on image processing* 13.4 (2004), pp. 600–612.
- [298] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. “Multiscale structural similarity for image quality assessment”. In: *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*. Vol. 2. Ieee. 2003, pp. 1398–1402.
- [299] Emmet Murphy, Shashwat Khandelwal, and Shreejith Shanker. “Custom precision accelerators for energy-efficient image-to-image transformations in motion picture workflows”. In: *Applications of Digital Image Processing XLVI*. Vol. 12674. SPIE. 2023, pp. 191–202.