



UCLouvain

Institute for Information
and Communication Technologies,
Electronics and Applied Mathematics

Joint learned compression and denoising schemes for improved photographic image quality and reduced processing complexity

Benoit Brummer

Thesis submitted in partial fulfillment
of the requirements for the degree of
Ph.D. in Engineering Sciences

Dissertation committee:

Prof. Christophe De Vleeschouwer (UCLouvain, advisor)

Prof. Benoît Macq (UCLouvain)

Prof. Antonin Descampe (UCLouvain)

Dr. Tim Bruylants (intoPIX)

Prof. Michael S. Brown (York University, Toronto)

Prof. David Bol (UCLouvain, chair)

Version of March 2, 2025.

Thanks

I would like to thank everyone involved in the completion of this thesis, and in particular Agathe Demeure, Christophe De Vleeschouwer, Gauthier Crucifix, the many great tits that adorn my office window in exchange for sunflower seeds, Hatim Boustique, the intoPIX research team, Iulia ‘Tralala’ Grosman, Johannes Hanika, my jury members, Kursti DeLello, Larry the cat 🐈, Leslie E. Sanderson, Pascal Pellegrin, Paula Martins, Wikimedia Commons QIC reviewers, and Yvonne Lens ♥.

Research statement

THIS thesis addresses two fundamental challenges in image processing—image compression and image denoising—using deep learning techniques to improve both visual quality and computational efficiency.

The first challenge is image compression, which aims to minimize the storage and transmission cost of images while preserving visual quality. Recent methods based on convolutional autoencoders have shown superior results, but their reliance on complex entropy models for predicting the probability distribution of each feature leads to higher computational costs. This work proposes a simplified compression scheme that uses a single convolutional autoencoder with a set of multiple learned prior distributions stored in static tables of cumulative distribution functions (CDFs), as an alternative to computationally expensive single-feature parametric priors. During inference, these static priors allow the entropy coder to efficiently compress spatial features across all channels. The proposed method achieves comparable rate-distortion performance to other state-of-the-art models, while significantly reducing entropy coding and decoding complexity.

The second challenge, image denoising, involves the removal of unwanted noise from images captured under suboptimal conditions. Noise not only degrades image quality but also impairs the performance of both standard and learned compression algorithms, as it is inherently non-compressible. This thesis proposes a unified model that performs joint denoising and compression (JDC). By training the model on noisy-clean image pairs across a wide range of noise levels, it learns to denoise images effectively as part of the compression process, all while maintaining the computational cost of compression alone. This joint approach improves rate-distortion performance compared to compressing noisy images or using separate denoising and compression models. Additionally, the model is capable of producing decompressed images with visual quality superior to that of the noisy uncompressed inputs.

The final part of this thesis focuses on raw input images, specifically Bayer pattern images produced by most digital cameras. It is demonstrated that processing raw or minimally processed images (e.g., debayered and converted to the standard linear Rec. 2020 color profile) offers substantial gains in both compression efficiency and denoising quality compared to working with fully processed sRGB images. Treating Bayer images as 4-channel inputs reduces the computational complexity of denoising models and compression encoders by a factor of four, while also improving compression performance at lower bitrates. Moreover, denoising raw or linear RGB images early in the processing pipeline enables greater generalization. Unlike models trained on processed sRGB images, which perform well only on data processed by specific camera image signal processors (ISPs) or software pipelines, early-stage denoising of raw or linear RGB data ensures compatibility across diverse imaging systems, development software, and processing styles.

To facilitate further research and development, a novel dataset of raw clean-noisy image pairs is introduced, with each scene containing at least one ground-truth clean image paired with multiple noisy versions captured at varying ISO levels and exposure times. This dataset supports academic research and the development of denoising models integrated into the pixel pipeline of image processing software, providing a valuable resource for advancing denoising and compression techniques in real-world applications.

Contents

Thanks	iii
Research statement	v
Table of contents	vii
List of figures	xi
List of tables	xiii
List of publications	xv
List of symbols and abbreviations	xvii
1 Introduction	1
1.1 Motivation	1
1.1.1 Learned image compression complexity	1
1.1.2 Impact of noise on image quality and compression efficiency	2
1.1.3 Challenges with fully developed images	3
1.2 Thesis contributions	3
1.3 Thesis structure	5
2 Background	7
2.1 Image development	7
2.2 Convolutional neural networks	11
2.3 Image denoising	15
2.3.1 Image noise	15
2.3.2 Traditional denoising algorithms	17
2.3.3 Learned denoising methods	18
2.4 Image compression	22

vii

2.4.1	Standard image compression codecs	23
2.4.2	Learned compression	28
3	End-to-end optimized image compression with competition of prior distributions	33
3.1	Related works	34
3.2	Competition of prior distributions	35
3.3	Experiments	39
3.3.1	Method	39
3.3.2	Results	41
3.4	Conclusion	45
4	On the importance of denoising when learning to compress images	47
4.1	Background	49
4.2	Jointly learned denoising and compression	51
4.2.1	Our proposed NIN supervision strategies	52
4.3	Experiments	54
4.3.1	Practical implementation details	54
4.3.2	Results	55
4.4	Conclusion	63
5	Learning joint denoising, debayering and compression from the Raw Natural Image Noise Dataset	65
5.1	Related work	67
5.1.1	Image denoising	67
5.1.2	Raw image development pipelines	68
5.1.3	Learned image compression	68
5.1.4	Generalization across sensors	70
5.2	Proposed approach	70
5.2.1	RawNIND Dataset	71
5.2.2	Proposed denoising methods	75
5.2.3	Joint denoising and compression (with implicit demosaicing for Bayer input)	75
5.2.4	Advantages of the proposed approach	76
5.3	Validation methodology	77
5.3.1	Dataset preparation	77
5.3.2	Test sets and evaluation conditions	79
5.3.3	Baseline models for denoising	80
5.3.4	Joint denoising and compression experiments	80
5.4	Results and discussion	81
5.4.1	Denoising performance	81

5.4.2	Compression performance	85
5.4.3	Insights and implications	88
5.5	Conclusion	89
6	Conclusion	91
6.1	Results summary	92
6.2	Perspectives	93
6.2.1	Denoising outlook	93
6.2.2	Compression outlook	94
6.2.3	Limitations and future directions	95
	Acknowledgements	97
	Bibliography	99

List of figures

1.1	Raw denoising prior to development	3
2.1	Close-up photograph of a grid of pixels.	8
2.2	Image development pipelines	10
2.3	Convolution layer operating on an RGB image.	12
2.4	Image noise caused by increasing shutter speed.	16
2.5	Visual denoising results from NIND [9]	21
2.6	Rate-distortion and visual results with JPEG compression	24
2.7	JPEG XS compression adapted for different tasks.	27
2.8	Rate-IoU gains from adapting JPEG XS for segmentation	28
2.9	Compression autoencoder [32] [6] architecture	30
2.10	CDFs from a trained compression autoencoder [32]. . . .	31
3.1	Compression AE with competition of prior distributions.	36
3.2	Diversity among competing CDF tables	37
3.3	Test photographs segmented by CDF tables utilization. .	38
3.4	Plot of CDF tables utilization	40
3.5	Rate-distortion plots with 1-to-128 priors and HP	42
3.6	RD and visual results with learned compression.	43
4.1	Impact of noise on visual quality and JPEG compression	48
4.2	Compression and JDC of a clean/noisy image.	50
4.3	Joint denoising and compression architecture.	52
4.4	RD curve of standard codecs, prior denoising, and JDC. .	56
4.5	RD curve with of all codecs with(out) denoising and JDC.	57
4.6	Figures 4.5a and 4.5b combined in one plot	58
4.7	Different compression (and JDC) methods on a clean image.	60
4.8	RD curves on clean images	61
4.9	Comparison of different methods on a clean/noisy image	62
4.10	Plot of JDC methods and train/test noise thresholds . . .	63

5.1	Noisy image development pipeline	69
5.2	Raw Natural Image Noise Dataset preview	72
5.3	RawNIND composition by noisiness.	73
5.4	Bayer CFA conversion	74
5.5	Denoising a test image with different formats and methods	83
5.6	Denoising an unknown sensor test image	84
5.7	Compression, JDC, and JDDC of a test image	86
5.8	Rate-distortion of different methods on the RawNIND test set with different input formats and noise levels. . . .	87

List of tables

3.1	Maximum uncompressed bitrate required to store prior indices based on the number of distributions ($\log_2(N_{\text{CDF}}) \div 16^2$ bpp)	37
3.2	Hyperprior vs “Manypriors” complexity	43
3.3	Hyperprior vs “Manypriors” runtime	44
4.1	JDC data pairs abbreviations	53
5.1	RawNIND composition by scenes and camera models . .	73
5.2	Camera sensors specifications	74
5.3	Model complexity for denoising and compression	81
5.4	Denoising results for different input types	82
5.5	Denoising results, ablation study.	82

List of publications

- Benoit Brummer and Christophe De Vleeschouwer.
Adapting JPEG XS Gains and Priorities to Tasks and Contents.
In 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pages 629–633, IEEE, June 2020.
- Benoit Brummer and Christophe De Vleeschouwer.
End-to-End Optimized Image Compression with Competition of Prior Distributions.
In 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pages 1890–1894. IEEE, June 2021.
- Benoit Brummer and Christophe De Vleeschouwer.
On the Importance of Denoising when Learning to Compress Images.
In 2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), pages 2439–2447. IEEE, January 2023.
- Benoit Brummer and Christophe De Vleeschouwer.
Learning Joint Denoising, Demosaicing, and Compression from the Raw Natural Image Noise Dataset.
In review.

List of symbols and abbreviations

- λ rate-distortion training parameter increasing the image quality (thus bitrate) of compression models. 36, 37, 39, 41, 43, 54, 55, 75
- AE** autoencoder [32]. v, xi, 2, 4, 6, 14, 28, 29, 33–36, 39, 41, 44, 45, 47, 49–51, 54–57, 62, 70, 75, 80, 92, 94
- BM3D** block-matching and 3D filtering [18]. 18, 21, 22, 67, 80, 82
- bpp** bits per pixel. 24, 27, 28, 41, 43, 60, 62, 75, 81, 85–87
- CamRGB** camera-specific color space. 9, 10, 68, 71, 72, 74–76
- CDF** cumulative distribution function. v, xi, 4, 29, 31, 33–41, 43–45, 47
- CFA** color filter array. xii, 8, 66, 67, 74, 76, 95
- CNN** convolutional neural network [33]. 5, 7, 11–15, 18, 19, 57, 67
- COM:FP** Wikimedia Commons Featured Pictures [1]. 40, 52–54
- CPM** cumulative probability model. 29, 30, 33, 35, 39, 45
- CPU** central processing unit. 41, 44, 59
- FP** floating-point. 33
- GMAC** billion multiply-accumulate operations. 40, 41, 43, 59, 64, 81
- HP** hyperprior [7]. xi, xiii, 2, 4, 29, 31, 32, 34, 35, 39–45, 70, 92, 94
- IoU** intersection over union. xi, 25, 26, 28

- ISO** ISO *camera sensitivity* standard. vi, 15, 16, 20–22, 47, 52, 54, 69, 71, 78
- ISO** International Organization for Standardization. 23, 25, 26
- ISP** image signal processor. vi, 3, 6, 9, 21, 65, 68, 93, 94
- JDC** joint denoising and compression. v, xi–xiii, 4, 6, 22, 32, 49–55, 57, 59, 62–64, 70, 75, 76, 80, 81, 85, 89, 92–95
- JDDC** joint denoising, demosaicing and compression. xii, 5, 76, 81, 85, 93, 95
- LUT** look-up table. 34, 41, 44, 45
- MP** megapixel. 44, 59, 81
- MS-SSIM** multi-scale structural similarity index measure [64]. 13, 14, 16, 24–27, 39, 41, 42, 53, 55–57, 59, 61–64, 73, 75, 77, 79, 81–85, 87
- MSE** mean squared error. 13, 14, 37, 39, 41, 47, 54, 75
- NIND** Natural Image Noise Dataset [9]. xi, 5, 19–22, 48, 50, 51, 54, 56, 57, 61–64, 67, 70, 92
- NLM** non-local means [13]. 17, 18, 60, 67
- NN** neural network. 1, 11, 12, 18, 41, 43, 44, 55, 93, 94
- PSNR** peak signal-to-noise ratio. 14, 25, 41–43
- RawNIND** Raw Natural Image Noise Dataset. xii, xiii, 4–6, 21, 65–67, 70–73, 79, 81–85, 87, 89–91, 93–95, 97
- RD** rate-distortion. v, xi, 2–7, 23, 24, 28, 29, 31, 32, 34, 35, 39–42, 45, 47, 49, 51, 53, 55–57, 59, 61–64, 66, 67, 70, 75, 76, 81, 85–87, 89–95
- Rec. 2020** ITU-R Recommendation BT.2020 [26] color space. vi, 9–11, 68, 71, 72, 74–77
- ReLU** Rectified Linear Unit [49]. 12, 80
- RGB** red green blue. vi, xi, 4, 5, 8, 9, 11, 12, 66–71, 75–82, 84, 85, 88–90, 93, 95
- s** second. 16, 21, 44, 59

s.s. shutter speed. 16, 20, 21, 47, 48

SID See-in-the-Dark [14] dataset. 19, 53, 67

SIDD Smartphone Image Noise Dataset [2]. 19–21, 51, 64, 67, 70

sRGB standard RGB color space [59] (IEC 61966-2-1:1999). vi, 4, 5, 10, 20, 21, 80, 81, 85, 88

XMP Extensible Metadata Platform (ISO 16684). 5, 66, 67, 79, 83, 84, 89

XYZ CIE 1931 XYZ color space. 9, 10, 72, 74

1

Introduction

DIGITAL photography, with its ubiquitous presence and ever-increasing megapixel-count, has made image processing a cornerstone of modern computing and communication. Notably, two essential tasks within this domain are *image compression* and *image denoising*. Image compression aims to minimize storage and transmission costs of digital images while preserving visual quality, whereas image denoising removes noisy artifacts captured as part of the acquisition process while attempting to minimize the loss of useful details. Learning-based techniques have shown great promise in tackling these problems but their adoption has been limited, in part due to the computational complexity of most neural network-based approaches and the poor applicability of learned denoising techniques in actual workflows. This thesis aims to bridge the gap between state-of-the-art compression and denoising algorithms, and real-world imaging conditions. The focus is on improving image quality in real-world denoising, the integration of denoising within learned compression frameworks, and reducing the computational complexity of it all.

1.1 Motivation

1.1.1 Learned image compression complexity

Image compression has long been guided by hand-engineered codecs such as JPEG [63], JPEG 2000 [57], JPEG XS [28], and JPEG XL [29]. While these methods effectively exploit statistical redundancies to re-

duce image size, their block-based transforms or wavelet decompositions have inherent limitations—namely, susceptibility to blocking artifacts (as shown in Figure 2.6) and an inability to adapt to different types of content. Recent approaches based on *convolutional autoencoders* (*autoencoders* [32] (*AEs*)) have demonstrated superior rate-distortion (RD) performance [7, 48, 47, 11], learning data-driven transforms that more effectively capture the underlying image structure.

However, deep-learning-based approaches have yet to gain meaningful adoption, partly because of their increased computational complexity, which increases runtime and power consumption and often limits users to those with specialized hardware. Learned compression models often rely on complex *entropy models* to predict the probability distributions of latent features. For instance, hyperprior [7] (HP) methods or autoregressive models assign a unique probability distribution to each feature at every spatial location—a strategy that can be computationally expensive. Balancing high compression performance with manageable complexity remains a crucial challenge.

1.1.2 Impact of noise on image quality and compression efficiency

Noise is an ever-present issue in photography, arising from image sensor limitations and challenging scenarios such as low-light or fast moving scenes. Denoising algorithms tend to oversmooth images and they must be carefully tuned to strike the right balance between noise removal and loss of details. This thesis addresses effective blind (parameter-free) denoising as a stand-alone problem as well as a joint issue to improve learned compression. While standard lossy compression codecs may inadvertently smooth some noise, they typically allocate more bits to reconstruct noise which contains high entropy, significantly degrading RD performance. This inefficiency persists even in learned codecs, as models trained on “clean” data alone do not learn to discard noise effectively.

A naive solution is to denoise before compression; yet, this approach significantly increases the computational burden (as one network is needed for denoising and another one for compression) and does not necessarily yield optimal results. A more efficient strategy is to teach a single model to learn *both* denoising and compression end-to-end, thereby discarding noise early and allocating bits primarily to semantically relevant details.

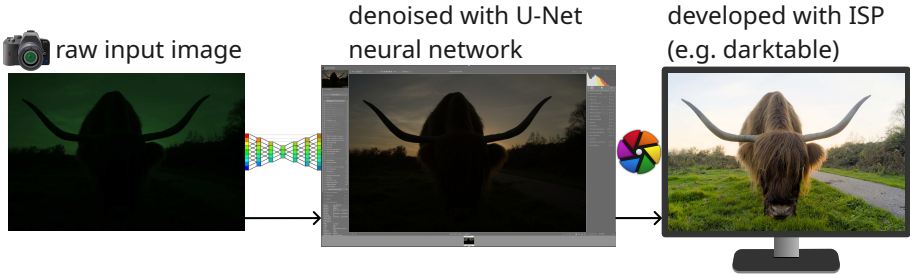


Figure 1.1: Raw input image captured by a camera sensor (left) and a developed image ready for display (right). An intermediate step involves denoising with a neural network (center) and converting to a standard wide-gammut color space. Denoising raw images before they undergo non-linear operations in an ISP enhances denoising performance and generalization. A complete development pipeline is shown in Figures 2.2 and 5.1.

1.1.3 Challenges with fully developed images

In practice, most images are processed by camera ISPs or raw-development software (e.g., Darktable, Adobe Lightroom). These processing pipelines include nonlinear operations such as sharpening and tone mapping (as shown in Figures 2.2 and 5.1), which alter noise characteristics in a complex, often irreversible manner. Models trained on such developed images risk becoming tightly coupled to the software pipeline or proprietary ISP used, undermining generalization and potentially limiting the scope of denoising.

By contrast, working on *raw* or *minimally processed* images, as shown in Figure 1.1, preserves linear relationships and circumvents unpredictable ISP-based transformations. This approach has profound benefits, including improved denoising performance and generalization by tackling noise directly with a physical model of the sensor data, enhanced compression RD efficiency by avoiding redundancies introduced by color interpolation (demaicing), and allowing for computationally simpler models with reduced dimensions.

1.2 Thesis contributions

This thesis introduces multiple technical contributions, organized around three key themes:

A simplified compression scheme with multiple learned prior distributions Recent learned compression models rely on hyperpriors [7], producing a unique parameterized probability distribution per feature. In Chapter 3, we propose *competition of prior distributions*, where a limited set of static CDF tables is learned. Each spatial location in the latent code (across all channels) uses the prior which yields the lowest bit cost. This approach:

- Matches the RD performance of existing hyperprior-based methods.
- Significantly reduces entropy coding and (especially) decoding overhead by eliminating the need for computationally expensive per-variable CDFs.
- Simplifies the learned compression pipeline without sacrificing performance.

A unified model for joint denoising and compression (JDC) In Chapter 4, we demonstrate that training a *single* AE to both denoise and compress *noisy* inputs yields substantial rate-distortion gains. Notably, the JDC model:

- Produces images with higher visual quality and a lower bitrate than compressing noisy data or performing denoising as a separate pre-processing step.
- Improves general compression performance even on clean images, provided the model sees mixed (paired) clean-noisy and (unpaired) clean data during training.
- Performs the dual tasks of denoising and compression with almost an order-of-magnitude fewer operations compared to separate networks.

Extending denoising and compression to raw sensor data and introducing a new dataset The final part of this thesis, Chapter 5, shows that operating on *raw Bayer* or minimally processed (linear red green blue (RGB)) images significantly outperforms methods that rely on fully developed sRGB images. We make three notable contributions:

- **Raw Natural Image Noise Dataset (RawNIND):** A novel dataset of paired noisy and clean raw images from diverse camera sensors. It enables training and evaluation of models that generalize across sensors, capturing realistic noise in varied scenes.

- **Joint denoising, demosaicing and compression (JDDC):** We demonstrate that compressing raw Bayer images with integrated denoising not only improves RD performance but also *reduces* computational cost by working at quarter-resolution data before the final demosaicing step.
- **Efficient denoising:** The U-Net model can denoise linear RGB and Bayer images using fewer hidden channels while preserving the same quality. This reduces computational complexity to between one-quarter and $1/16^{\text{th}}$ of what was previously required to denoise sRGB images with the Natural Image Noise Dataset [9] (NIND).
- **Practical denoising:** Denoising models trained with RawNIND to process raw or linear data can integrate or precede the processing pipeline of any image development software, allowing practical denoising applications compatible with any workflow.
- **Better Generalization.** Early-stage noise removal and compression yield consistent performance across any image development pipeline. Raw or linear denoising enables practical denoising applications compatible with any image development workflow, and joint denoising, demosaicing and compression enables effective compression which may embed development instructions (XMP) and non-destructive editing.

1.3 Thesis structure

To address these research objectives in a structured manner, the thesis is organized into the following chapters:

Chapter 2: Background Provides an overview of the image development pipeline, the fundamentals of convolutional neural networks [33] (CNNs), image denoising, and image compression. It surveys classical and learned approaches to both compression and denoising, highlighting key challenges and design trade-offs.

Chapter 3: End-to-end optimized image compression with competition of prior distributions Introduces our simplified compression scheme that uses multiple static learned prior distributions to reduce computational overhead during entropy coding while maintaining competitive RD performance.

Chapter 4: On the importance of denoising when learning to compress images Explains our joint denoising and compression (JDC) framework. We show how training a single autoencoder [32] (AE) to remove noise and compress images significantly improves RD outcomes and reduces computational complexity.

Chapter 5: Learning joint denoising, demosaicing, and compression from the Raw Natural Image Noise Dataset Extends our approach to raw sensor data, introducing the Raw Natural Image Noise Dataset and integrating denoising and compression at the earliest stage of the camera pipeline. This chapter demonstrates how operating on raw or linear color data improves both denoising and compression performance while enabling cross-sensor generalization and reducing computational complexity.

Chapter 6: Conclusion Summarizes the key findings, underscores the implications of performing noise removal *before* compression, and highlights promising avenues for future research—such as improved architectures for raw data, standardization challenges, and real-world deployment in camera ISPs and raw-development software.

In summary, this thesis puts forth a novel perspective: *noise is neither desirable nor compressible*, and the architecture of learned compressors should be adapted accordingly. By showcasing the advantages of static learned priors for simpler entropy coding, and by demonstrating the benefits of combining denoising with compression—particularly when working with raw or linear images—we aim to establish a more efficient and robust imaging pipeline. The subsequent chapters detail the theoretical underpinnings, experimental evaluations, and broader implications of these contributions.

2

Background

THIS chapter lays the groundwork for the contributions presented in this thesis, which center on improving image quality and compression efficiency by integrating denoising directly into the learned compression process. This work draws from advances in deep learning, image processing, and digital photography, addressing the dual challenges of preserving visual fidelity while reducing storage and bandwidth requirements.

We begin with a brief overview of the *image development* pipeline in Section 2.1, outlining the steps needed to output of a digital image from its acquisition by the sensor through various processing stages. We then introduce *convolutional neural networks* [33] (CNNs) in Section 2.2, with an emphasis on their application to both compression and denoising tasks. The chapter then introduces the problem of *image denoising* in Section 2.3, reviewing classical algorithms alongside modern deep-learning approaches. Special attention is given to real-world noise, the challenges noise poses to compression, and image noise datasets that enable effective model training. Finally, Section 2.4 discusses *image compression*, outlining the evolution from standard codecs to end-to-end learned methods, and highlighting the trade-offs between processing complexity and RD performance.

2.1 Image development

Digital images are a cornerstone of modern communication. At their most fundamental level, digital images are structured grids of discrete



Figure 2.1: Close-up photograph of an LCD display showing a grid of pixels with their individual RGB components (sub-pixels).

picture elements, or *pixels*. Each pixel represents a single point of color and brightness within the scene. As shown in Figure 2.1, each pixel typically comprises multiple channels, the red, green, and blue (RGB) components, which define the color at that specific location. These numerical values, often ranging from 0 to 255 for each channel in standard 8-bit representations, collectively define the visual content of the image. Understanding this discrete, multi-channel nature of image representation is essential for comprehending the subsequent processing steps involved in creating and manipulating digital images.

Digital image creation commences with *acquisition* by an image sensor. These sensors, found in devices ranging from consumer cameras to sophisticated scientific instruments, convert incoming light into electrical signals. When light photons strike the sensor's photosites, they generate an electrical charge proportional to the light's intensity. To capture color information, most digital cameras employ a *Bayer color filter array (CFA)* overlaid on the sensor. This array is a mosaic pattern of tiny color filters, typically arranged with alternating rows of red and green filters, and blue and green filters. Consequently, each photosite captures only the intensity of either red, green, or blue light. This arrangement enables the capture of color information, which is later interpolated to form a full-color image. The raw data captured by the sensor represents the scene's light information before any significant processing.

Following acquisition, the raw sensor data undergoes a series of transformations within an *image signal processor (ISP)* to produce a viewable image, an example of which is depicted in Figures 2.2 and 5.1. The initial step involves adjusting black and white points to establish the image’s dynamic range. Subsequently, a *demosaicing* algorithm interpolates the missing color values at each pixel location, leveraging the known Bayer pattern arrangement to create an initial three-channel RGB representation. To ensure consistency across different camera sensors, the camera-specific color information is then transformed into a standardized *color space*. A common approach, as implemented in software like Darktable, involves an initial conversion from the camera-specific color space (CamRGB) to a device-independent color space like the CIE 1931 XYZ color space (XYZ), and then to a working color space such as the linear ITU-R Recommendation BT.2020 [26] color space (Rec. 2020) with a daylight white balance. This early white balance is often applied because camera manufacturers consistently provide a single XYZ→CamRGB conversion matrix that is calibrated for the standard D65 illuminant (daylight white balance). Further refinement of the color appearance can be achieved through color calibration later in the pipeline. Numerous other processing modules can be applied depending on the desired aesthetic, the specific shooting conditions, and the intended purpose of the image. These include lens correction to mitigate distortion, sharpening to enhance detail, and various artistic filters. Tone mapping is then applied to map the high dynamic range of the sensor data into the limited dynamic range of display devices. Two essential processes within this pipeline are *image denoising* and *image compression*. As discussed in detail in Section 2.3, noise is an inherent part of the image acquisition process, and its reduction is crucial for improving visual quality. Similarly, *image compression*, elaborated upon in Section 2.4, is vital for reducing the storage and transmission requirements of the large image files generated by modern sensors.

The specific order and implementation of these steps can vary significantly depending on whether the processing is performed by the camera’s internal ISP or by raw development software such as Darktable or Adobe Lightroom. In-camera ISPs prioritize speed and general-purpose usability, employing proprietary algorithms. Raw development software, on the other hand, offers users extensive control and customization options. This variability in processing pipelines and the unique characteristics of different camera sensors pose challenges for denoising and compression models, which are often trained on data processed through a specific pipeline. This thesis explores the hypothesis that models trained on raw or minimally processed images, such as linear RGB

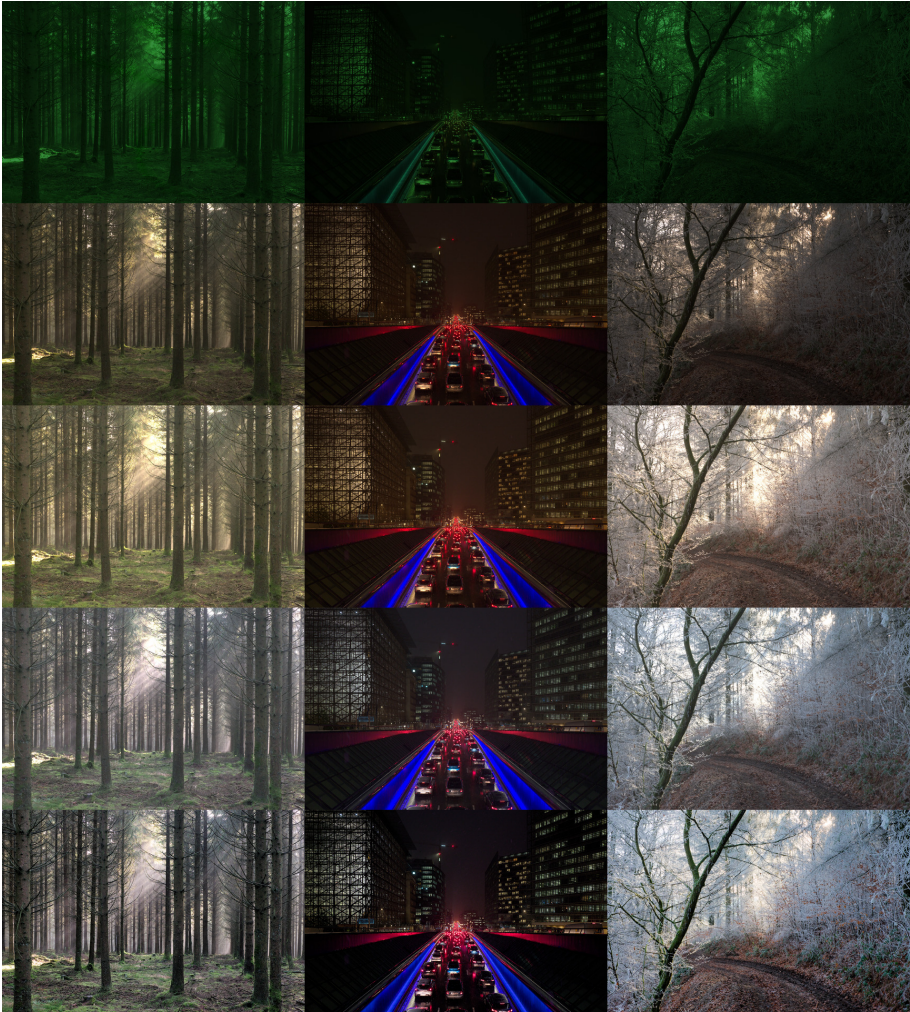


Figure 2.2: An image development pipeline illustrating the development of three photographs using Darktable 5.0. The main stages shown are as follows, from top to bottom: 1: Raw image with Bayer or X-Trans mosaic, 2: highlight reconstruction, demosaicing and color conversion from CamRGB to XYZ to Rec. 2020 (working profile) at D65 (daylight white balance), 3: denoising, lens and perspective correction, crop, exposure adjustment, 4: color calibration, 5: local contrast enhancement, lens deblurring, and sharpening, color balancing, filmic tone mapping, and tone curve. The developed images are then exported, typically with an sRGB color profile and JPEG compression.

data in a standardized color space like Rec. 2020, can achieve higher performance by avoiding the non-linearities and potential information loss introduced by more advanced processing steps. Working with raw or linear RGB data also has the potential to improve the generalization of models across different camera sensors and processing styles.

The complexity and sophistication of image processing have been significantly enhanced by the advent of *neural networks* (NNs). These powerful computational models, particularly convolutional neural networks [33] introduced in Section 2.2, have demonstrated remarkable capabilities across a wide range of image processing tasks. Their ability to learn intricate patterns and relationships directly from image data has revolutionized fields like object recognition, image segmentation, and, crucially for this thesis, both image denoising and compression. Neural networks can be trained to effectively remove noise by learning the underlying structure of clean images and distinguishing it from unwanted artifacts. Furthermore, learned compression techniques leverage neural networks to develop more efficient representations of image data, achieving higher compression ratios while maintaining acceptable visual quality. The following sections will delve deeper into the specific applications of CNNs in addressing the challenges of image denoising and compression, setting the stage for the contributions presented in this thesis.

2.2 Convolutional neural networks

Convolutional neural networks [33] are a class of deep learning models that have revolutionized the field of computer vision, demonstrating remarkable success in a wide array of tasks including image classification, object detection, and image generation [33, 34]. Their architecture is specifically designed to process grid-like data, such as images, by exploiting spatial hierarchies and translational invariance.

The core building block of a CNN is the **convolution**, which is illustrated in Figure 2.3. A convolutional layer applies a set of learnable filters, also known as kernels, across the input image. These filters convolve with local regions of the input, extracting features such as edges, corners, and textures. By sharing weights across the spatial dimensions, convolutional layers drastically reduce the number of parameters compared to fully connected networks, making them more efficient to train and less prone to overfitting. Multiple convolutional layers are typically stacked, with each subsequent layer learning increasingly complex and abstract features from the outputs of the previous layers. The weights

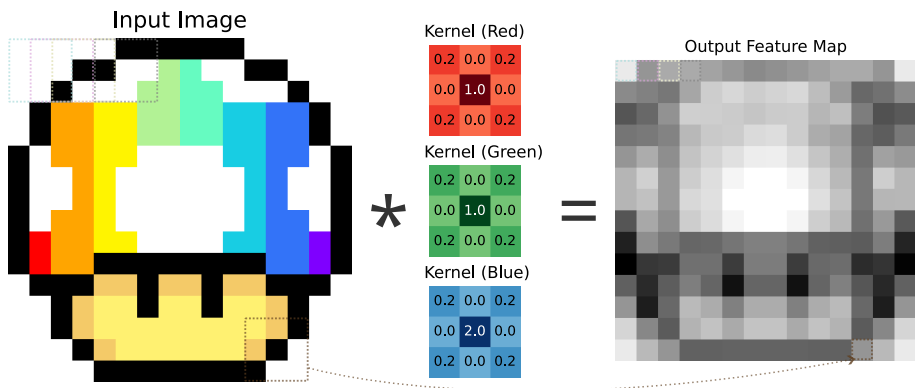


Figure 2.3: Convolution layer operating on an RGB (3-channels) image. The kernels are learned by the neural network to extract useful features. The same $3 \times 3 \times \text{channels}$ kernels are applied to every part of the image using a sliding window. Each kernel produces a feature map, which is used as one of the (channel) input to the next layer. A neural network typically consists of many layers and its inner layers contain many channels. As these feature maps propagate through the network, they progressively capture more complex patterns, forming the foundation for tasks like classification, segmentation, and denoising.

of these kernels are learned during the training process using back-propagation, an algorithm that computes the gradient of a loss function with respect to the network’s parameters and updates these parameters iteratively to minimize the loss [55].

Following convolutional layers, non-linear activation functions, such as Rectified Linear Unit [49] (ReLU), are applied element-wise to introduce non-linearity into the model, enabling it to learn intricate patterns. Pooling layers, such as max pooling, are often interspersed between convolutional layers to reduce the spatial dimensionality of the feature maps, decreasing computational cost and increasing the network’s robustness to small translations in the input.

The ability of CNNs to automatically learn hierarchical spatial features makes them particularly well-suited for image processing tasks. Lower layers typically detect basic local patterns, while deeper layers combine these features to recognize more global and semantic information. This hierarchical representation learning is a key reason for their success in understanding visual data. While CNNs have been the dominant architecture for many image processing tasks, alternative architectures like Transformers [62], initially developed for natural lan-

guage processing, have recently shown promising results in various computer vision applications, including image compression [39, 3] and denoising. Transformers leverage self-attention mechanisms to capture long-range dependencies in images, potentially offering advantages in capturing global context. Although this thesis focuses on CNN-based approaches, the exploration of transformer-based methods presents a compelling direction for future research in joint learned compression and denoising.

For a CNN to effectively learn the desired mapping from input to output, it requires a **loss function** (or objective function). The loss function quantifies the discrepancy between the network’s predictions and the ground truth, providing a signal for the backpropagation algorithm to adjust the network’s weights. Loss functions used in CNNs can be broadly categorized based on the nature of the task. For tasks like image classification or object detection, the loss function typically aggregates the errors across the entire output to produce a single scalar value that represents the overall performance. Conversely, for tasks like image segmentation, denoising, or compression, the network’s output has the same spatial dimensions as the input, and the loss function is often computed pixel-wise (or patch-wise) and then averaged.

In the context of image denoising and compression, the loss function mainly serves as a distortion metric, guiding the network to produce outputs that are both visually similar to the target and efficient to encode (in the context of compression). Common distortion metrics used for image reconstruction include the L1 loss, the mean squared error (MSE), and the multi-scale structural similarity index measure [64] (MS-SSIM).

The L1 loss (or Mean Absolute Error) calculates the average of the absolute differences between the predicted pixel values ($\hat{y}_i$) and the ground truth pixel values (y_i):

$$L_1(\hat{x}, x) = \frac{1}{N} \sum_{i=1}^N |\hat{x}_i - x_i| \quad (2.1)$$

where N is the total number of pixels. This loss function is less sensitive to outliers compared to MSE, as it penalizes all errors linearly.

The mean squared error (MSE) is another commonly used loss function, calculating the average of the squared differences between the predicted and ground truth pixel values:

$$\text{MSE}(\hat{x}, x) = \frac{1}{N} \sum_{i=1}^N (\hat{x}_i - x_i)^2 \quad (2.2)$$

MSE penalizes larger errors more heavily than smaller errors due to the squaring operation. While MSE is mathematically convenient, it does not correlate very well with perceived visual quality.

A related metric often used for evaluating image quality, particularly in the context of compression, is the peak signal-to-noise ratio (PSNR). PSNR is inversely proportional to MSE and is typically expressed in decibels (dB):

$$\text{PSNR}(\hat{x}, x) = 10 \cdot \log_{10} \left(\frac{\text{MAX}_I^2}{\text{MSE}(\hat{x}, x)} \right) \quad (2.3)$$

where MAX_I is the maximum possible pixel value of the image (e.g., 255 for an 8-bit image). Higher PSNR values generally indicate better image quality. Maximizing PSNR is equivalent to minimizing MSE.

The multi-scale structural similarity index measure [64] (MS-SSIM) is designed to better reflect human visual perception by considering image degradation as perceived changes in structural information. Unlike pixel-wise metrics like L1 and MSE, MS-SSIM assesses image similarity across multiple scales and incorporates factors like luminance, contrast, and structure. The MS-SSIM between two images x and y is calculated as:

$$\text{MS-SSIM}(\hat{x}, x) = l_M(x, y)^\alpha \prod_{j=1}^M c_j(\hat{x}, x)^\beta s_j(\hat{x}, x)^\gamma \quad (2.4)$$

where l_M is the luminance comparison at the M -th scale, c_j is the contrast comparison at the j -th scale, and s_j is the structure comparison at the j -th scale. α , β , and γ are parameters used to adjust the relative importance of each component. The MS-SSIM value ranges from 0 to 1, with higher values indicating greater similarity. This makes MS-SSIM a more perceptually relevant metric than L1 or MSE for assessing the quality of denoised or compressed images. MS-SSIM can be used as a loss function by complementing it such that $\text{MS-SSIM}_{\text{loss}} = 1 - \text{MS-SSIM}$.

In the context of this thesis, CNNs serve as the fundamental building blocks for both our proposed image compression and image denoising methods. Specifically, autoencoder [32] (AE) architectures, which consist of an encoder network that compresses the input into a lower-dimensional latent representation and a decoder network that reconstructs the input from this representation [32], form the basis of our learned compression framework. These autoencoders learn to extract the most salient features of the image for efficient encoding. For image denoising, we leverage the U-Net architecture [54], which is an encoder-decoder network with skip connections that facilitate the preservation

of fine-grained details during the denoising process. The skip connections allow the decoder to access high-resolution feature maps from the encoder, enabling more accurate reconstruction of the clean image.

The subsequent sections will delve deeper into how CNNs are applied and adapted for the specific challenges of image denoising and compression.

2.3 Image denoising

2.3.1 Image noise

Image noise is an undesirable artifact that corrupts images during the acquisition process, obscuring fine details and degrading visual quality [21, 9, 51]. This degradation manifests as random variations in pixel values, originating from fundamental limitations inherent in the image capture process. These limitations include the discrete nature of light and its interaction with the sensor, thermal and electronic noise within the sensor and camera circuitry, and the analog-to-digital conversion process. A primary factor influencing the level of noise is the amount of light reaching the sensor. As illustrated in Figure 2.4, insufficient exposure time, often necessitated by fast-moving subjects or logistical constraints preventing the use of a tripod, forces the camera to operate at higher ISO settings. This amplifies the signal and, consequently, the noise, to produce a properly exposed image.

While simplified mathematical models like Gaussian noise and Poisson noise can be useful for analytical purposes, they are often idealized representations and fail to fully capture the complex, signal-dependent nature of real-world photographic noise encountered in practical scenarios [9, 2]. Real-world noise is influenced by intricate interactions within the camera's imaging pipeline and varies significantly across different camera models and capture settings. Consequently, denoising methods developed and evaluated solely on synthetically generated noise, based on these simplified models, often exhibit limited effectiveness when applied to actual photographic images.

In photography and other imaging domains, image noise presents a persistent challenge. Reducing noise is crucial for achieving visually clean images, but any denoising process inevitably involves a trade-off with the preservation of fine details and textures. Aggressive noise reduction can lead to undesirable over-smoothing, effectively erasing subtle details and the unique characteristics of the photographic capture. Photographers must carefully balance noise reduction with detail retention, employing various techniques in-camera and during post-

2.3. Image denoising

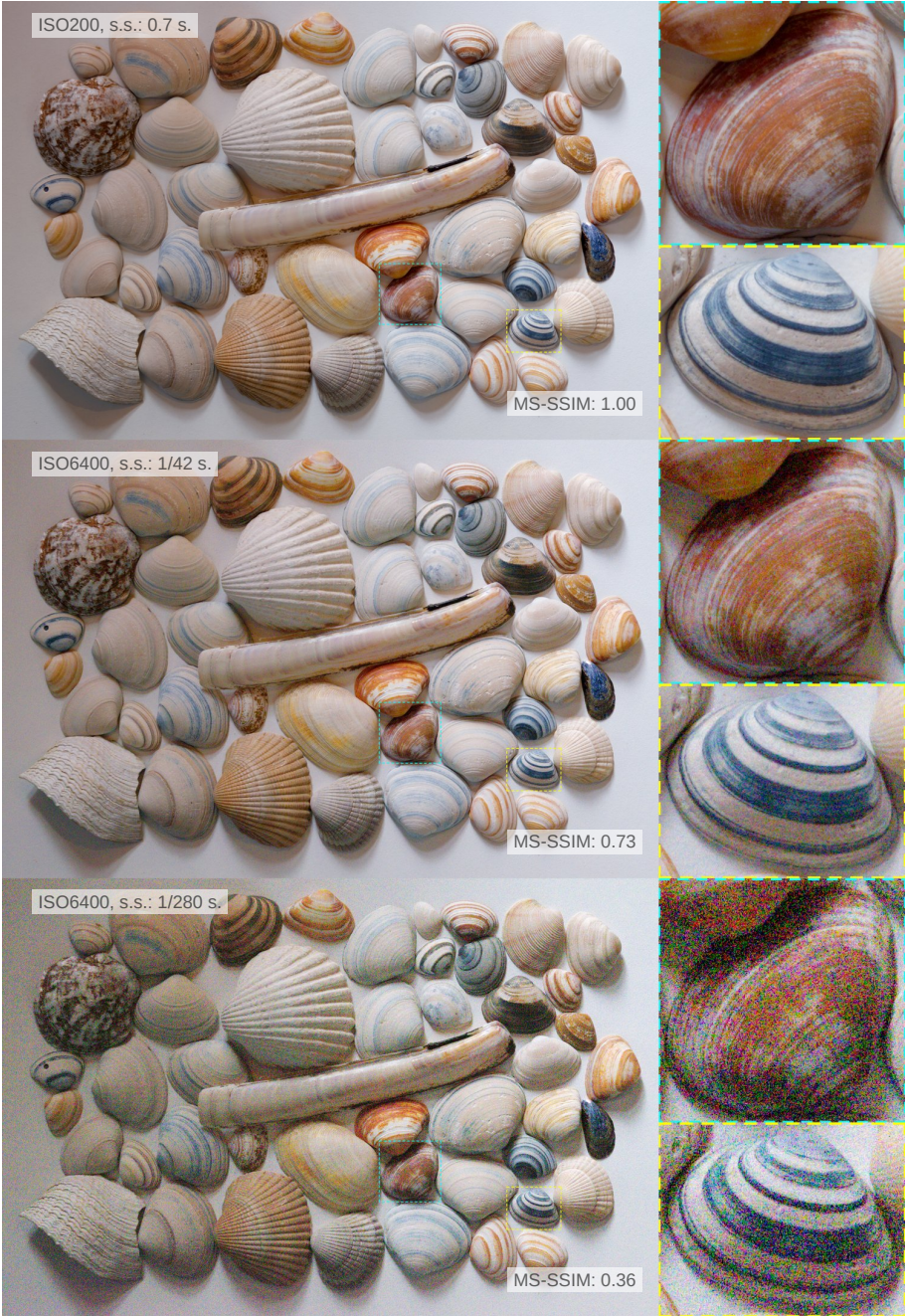


Figure 2.4: Image noise increases as the shutter speed (and ISO) increase. The MS-SSIM assesses image quality with respect to the ground-truth (top) image which was captured with an optimal shutter speed of 0.7 seconds (s).

processing to achieve an acceptable result. This balancing act highlights the need for robust denoising methods that minimize the loss of image information. Beyond its visual impact in photography and other imaging domains, noise also presents a challenge for subsequent image processing tasks. For example, in image compression, noise increases entropy and reduces coding efficiency, compelling compression algorithms to allocate more bits to encode information that is not representative of the desired image content and is, therefore, fundamentally superfluous. Similarly, many computer vision algorithms perform optimally on clean data, with noise acting as a confounding factor. Addressing image noise remains a critical aspect of both enhancing visual quality and enabling efficient and reliable image processing pipelines, motivating the core investigations of this thesis.

2.3.2 Traditional denoising algorithms

Image denoising has been a central focus in image processing, and before the advent of deep learning, a variety of effective algorithmic approaches were developed based on mathematical modeling and signal processing principles. These traditional methods remain relevant, particularly in established image processing workflows, due to their computational efficiency and well-understood performance characteristics. For instance, popular open-source raw image processing applications like Darktable implement several traditional denoising techniques, including non-local means [13] (NLM) and wavelet-based methods, highlighting their continued practical importance.

Simple spatial filtering techniques, such as mean filtering and median filtering, offer computationally efficient ways to reduce noise. Mean filtering averages pixel values within a local neighborhood, effectively smoothing noise but potentially blurring fine details. Median filtering, which replaces a pixel with the median of its neighbors, is more effective at removing impulsive noise while better preserving edges.

Transform domain filtering techniques leverage the principle that image signal and noise often exhibit different characteristics when transformed into another domain, such as the frequency domain. Wavelet-based denoising [19] exemplifies this by decomposing the image into different frequency bands. This decomposition allows for the selective attenuation of coefficients that are likely to correspond to noise, effectively separating noise from important image features. Due to their efficiency and effectiveness against certain types of noise, wavelet methods remain valuable tools and are implemented in contemporary image processing software.

Non-local means [13] represents an important class of traditional denoising algorithms that exploits the inherent self-similarity present in natural images. NLM operates by averaging pixel values based on the similarity of their surrounding neighborhoods, effectively leveraging information from spatially distant but similar regions of the image to achieve impressive denoising results while better preserving fine details than simpler filtering methods. While more computationally intensive than basic spatial filters, NLM remains a valuable technique and is utilized in software like Darktable.

Building upon the concept of non-local similarity, the block-matching and 3D filtering [18] (BM3D) algorithm extends NLM by grouping similar image patches into three-dimensional blocks and applying collaborative filtering in a transform domain. This approach has proven highly effective and has consistently served as a high-performing benchmark in image denoising research, demonstrating superior results under controlled conditions, albeit with a higher computational cost compared to simpler methods.

In summary, these traditional denoising algorithms offer valuable and often computationally efficient solutions for reducing image noise, and they remain integral to many image processing pipelines. However, they also have limitations. Their performance can be constrained when dealing with complex, signal-dependent real-world noise, and they often require careful, manual parameter tuning for optimal results. The emergence of learned denoising methods, which leverage the ability of deep neural networks to learn complex mappings directly from data, provides a powerful alternative that addresses some of these limitations, as will be discussed in the following section.

2.3.3 Learned denoising methods

The emergence of deep learning has led to a paradigm shift in image denoising, with convolutional neural networks [33] achieving state-of-the-art performance, often surpassing traditional algorithmic approaches in both effectiveness and the ability to handle complex, real-world noise. Unlike traditional methods that rely on hand-crafted features and explicit noise models, learned denoising methods leverage the capacity of deep neural networks to learn complex mappings directly from noisy input images to their clean counterparts. This data-driven approach allows the models to implicitly learn the intricate characteristics of noise distributions present in the training data.

Early successes in learned denoising involved relatively simple CNN architectures trained primarily on synthetically generated Gaus-

sian noise [30]. While these methods demonstrated the potential of CNNs for denoising, they often struggled to generalize effectively to real-world scenarios where noise is significantly more complex and signal-dependent. The development of deeper and more sophisticated architectures, such as the Denoising Convolutional Neural Network (DnCNN) [65], marked a significant advancement in network architecture. DnCNN introduced techniques like residual learning and batch normalization, enabling the training of deeper networks and improving performance on a wider range of (often still synthetic) noise levels. However, a significant limitation of DnCNN and many contemporary methods was their continued reliance on synthetic noise for training, which often led to suboptimal performance when applied to real-world noisy images.

A key factor driving the success and practical applicability of learned denoising methods on real-world data has been the increasing availability of large datasets of paired clean-noisy images captured under realistic conditions. Recognizing the limitations of synthetic noise, researchers introduced valuable datasets that capture the complex characteristics of real-world noise. For instance, the See-in-the-Dark [14] dataset (SID) focused on extreme low-light scenarios, followed by the Smartphone Image Noise Dataset [2] (SIDD) targeting noise in smartphone photography, and the Natural Image Noise Dataset [9] (NIND), which covers a broader range of noise levels and targets DSLR and mirrorless-like cameras. These datasets provided valuable resources for training and evaluating denoising algorithms on real-world noise. The characteristics and contributions of the Natural Image Noise Dataset are further detailed in the following subsection.

Natural Image Noise Dataset

A critical factor in the advancement of learned denoising methods for real-world applications is the availability of high-quality datasets containing paired noisy and clean images captured under realistic conditions. Early learned denoising methods often relied on synthetically generated noise, which, while useful for initial exploration, fails to capture the complex, signal-dependent nature of noise encountered in real photographs [9, 2]. This discrepancy often leads to models that perform well on synthetic benchmarks but struggle to generalize to real-world noisy images.

To address this limitation, several datasets of real noisy-clean image pairs have been introduced, each with its own focus and characteristics. The See-in-the-Dark [14] dataset provides data specifically for denois-

ing extremely low-light images, while the Smartphone Image Noise Dataset [2] focuses on the noise characteristics prevalent in smartphone photography, where smaller sensors and apertures can lead to noticeable noise even at base ISO. Building on this trend, the Natural Image Noise Dataset [9] was created to provide a more general-purpose resource for training denoising models on realistic noise from DSLR and mirrorless-like cameras.

The NIND comprises a collection of image sets, each capturing a static scene under varying levels of real noise induced by adjusting the camera’s ISO sensitivity. The core principle behind its creation was to capture multiple images of the same scene with identical framing and content, differing only in their noise levels, as illustrated in Figure 2.4. Each scene includes at least one ground-truth clean image captured at the camera’s base ISO setting, along with several corresponding noisy images taken at incrementally higher ISO values and consequently faster shutter speeds (s.s.s) to maintain consistent exposure. This controlled acquisition process ensures a direct correspondence between the noisy and clean versions, facilitating supervised training of denoising models. The dataset includes images captured with different DSLR and mirrorless cameras (primarily a Fujifilm X-T1 and a Canon EOS 500D), allowing for the study of cross-sensor generalization. The scenes encompass a diverse range of subjects, including indoor and outdoor scenes, natural objects, and even text, aiming to provide the variety necessary for training robust, general-purpose denoising models. Figure 2.5 shows a visual comparison of different denoising methods applied to a sample image from the NIND.

A significant strength of the NIND lies in its open and collaborative nature. Published on Wikimedia Commons, the dataset is publicly accessible and allows for community contributions, curation, and discussion. This open platform enables continuous expansion, refinement, and validation of the dataset’s content. The inclusion of multiple ground-truth images for some scenes was a deliberate design choice to mitigate overfitting to specific noise instances. By randomly sampling from different clean versions during training, the model is encouraged to learn the underlying image structure rather than the specific noise present in a single ground-truth. Furthermore, careful attention was paid to avoid overexposed regions in the ground-truth images, as overexposure can mask detail and potentially give an advantage to higher ISO images where the sensor might not be saturated.

However, the NIND also presents fundamental limitations tied to its specific processing pipeline. Most significantly, the dataset delivers clean and noisy image pairs exclusively in the sRGB color space, an



Figure 2.5: Denoising results on the “stefantiek” test image from NIND. From left to right: 1: ISO200 ground-truth (s.s.: 1/20 s), 2: high ISO (s.s.: 1/1900 s), 3: 2 denoised using a U-Net model trained with NIND, 4: 2 denoised using a U-Net model trained with SIDD (320 image pairs), 5: 2 denoised using BM3D ($\sigma = 90$).

inherent consequence of the raw-to-sRGB conversion performed using a particular version of Darktable. This conversion is not merely a format change; it’s a cascade of irreversible operations – demosaicing, white balancing, color correction, and tone mapping – that discard valuable information present in the original raw sensor data. This constraint is compounded by the fact that the majority of the NIND was captured using cameras with Fujifilm’s unique X-Trans sensor. Consequently, even if the dataset were to be re-released in its raw form (as is the case with the subsequent Raw Natural Image Noise Dataset), its heavy bias towards the less common X-Trans pattern would limit its broader utility for training models intended for the more prevalent Bayer sensor architecture. Furthermore, the very notion of applying denoising algorithms to sRGB images in the middle of a processing pipeline – after these initial transformations but before final sharpening – is largely impractical for real-world integration. Denoising sRGB images at this stage is also inherently suboptimal, as the specific processing steps, particularly non-linear operations like sharpening, introduce artifacts and distort the noise characteristics, leading to poorer denoising outcomes. Indeed, while a significant body of research focuses on denoising sRGB images, this scenario doesn’t accurately reflect real-world image processing workflows, where denoising is typically performed early in the pipeline, within the camera’s ISP or raw development software, directly on the raw data. Finally, the reliance on a specific and now obsolete version of Darktable introduces another layer of constraint.

The default processing styles within Darktable have evolved since the NIND's creation, meaning that even when using the same software, a model trained on the NIND might not generalize effectively to images processed with current versions and their default settings. This underlines the inherent limitations of datasets tied to specific and potentially outdated processing pipelines.

Despite these limitations, the NIND has served as a valuable resource for training and evaluating learned denoising methods on real-world ISO noise. Its publicly available nature and the paired noisy-clean image structure have facilitated advancements in the field. The experiments conducted with the NIND, as detailed in our published paper, demonstrate the superior performance of models trained on real noise compared to those trained on synthetic noise and highlight the potential for learned methods to outperform even well-established traditional algorithms like BM3D when trained appropriately. The insights gained from the NIND directly inform the research directions explored in this thesis, especially the investigation into joint denoising and compression methods operating on raw image data.

2.4 Image compression

Image compression is the process of encoding digital images to reduce their file size, reducing the amount of storage space required to store images and the bandwidth required to transmit them over networks. Image compression is essential for a wide range of applications, such as storing photograph collections, web browsing, and medical imaging. Compression algorithms aim to represent image data using fewer bits than the original representation by exploiting redundancies present in the image data. These redundancies can be statistical (e.g., frequently occurring pixel values), spatial (e.g., smooth regions with similar colors), or perceptual (e.g., information the human eye is less sensitive to). Image compression techniques can be broadly categorized into two main types: lossless and lossy. Lossless compression methods aim to reduce file size without any loss of information, ensuring that the original image can be perfectly reconstructed from the compressed data. Examples of lossless compression codecs include PNG [27], as well as the lossless modes of JPEG 2000 [57] and JPEG XL [29]. Lossy compression, on the other hand, achieves significantly higher compression ratios by discarding some information that is deemed less perceptually important. While the reconstructed image is not identical to the original, the goal is to minimize the visual difference and maintain acceptable image quality.

for the intended application, i.e. optimize the RD tradeoff.

2.4.1 Standard image compression codecs

Standard image compression codecs have been instrumental in enabling the widespread use and sharing of digital images. These codecs typically rely on a combination of transform coding, quantization, and entropy coding techniques to achieve compression. Since its introduction in 1992, the JPEG standard [63] has become the most ubiquitous lossy image compression method. JPEG operates by dividing the image into 8x8 pixel blocks and applying a Discrete Cosine Transform (DCT) to convert each block from the spatial domain to the frequency domain. The DCT coefficients are then quantized, where high-frequency coefficients, which often correspond to less visually significant details, are represented with fewer bits or discarded altogether. Finally, the quantized coefficients are entropy encoded using techniques like Huffman coding and arithmetic coding [53]. While widely supported and efficient, JPEG can introduce noticeable block artifacts, especially at lower bitrates, as shown in Figure 2.6.

Successors like JPEG 2000 [57] offer improved compression efficiency and features compared to the original JPEG standard. However, its increased computational complexity has hindered its widespread adoption compared to JPEG. More recent standards include High Efficiency Video Coding (HEVC) [60]’s intra-frame coding (BPG), which can be used for still image compression and provides significant compression gains over earlier standards but with increased encoding and decoding complexity, and JPEG XL (ISO/IEC 18181) [29] which is designed to supersede JPEG with superior compression efficiency and support for both lossy and lossless compression, along with features like support for wide color gamuts thus high dynamic range.

For applications requiring lightweight image compression with low latency, the JPEG XS standard (ISO/IEC 21122) [28] has emerged as a relevant solution. JPEG XS employs a line-based wavelet transform and operates with low complexity, making it a compelling alternative to uncompressed video and more complex compression standards in resource-constrained environments such as in-camera pipelines, embedded systems, and real-time video streaming.

Adapting JPEG XS gains and priorities to tasks and contents

While standard compression codecs offer generic, broadly compatible solutions, their performance can often be enhanced by tuning codec



Figure 2.6: Rate-distortion (bits per pixel (bpp) - MS-SSIM) and visual results for compressing a photograph of Larry the cat with the JPEG codec.

parameters for specific tasks and image contents. The JPEG XS standard accommodates this need by allowing encoder-side configuration of gains and priorities for its wavelet sub-bands [28]. These gains (G_b) and priorities (P_b) are analogous to the quantization tables in traditional JPEG: each defines how aggressively different spatial-frequency bands and color components should be compressed. This design enables encoders to balance compression efficiency and fidelity in specific sub-bands, potentially improving perceptual quality, analysis accuracy, or other metrics of interest. The ISO 21122 standard provides a default set of weights aimed at maximizing PSNR [28, Tab. H.3], but it also recognizes that task-specific or content-specific adjustments may yield better overall performance.

Our motivation for experimenting with JPEG XS weights was twofold. First, JPEG XS is inherently low-complexity, aligning with the overarching theme of simplicity explored throughout this thesis. Unlike more advanced standards (e.g., HEVC Intra or AV1) or end-to-end learned codecs, JPEG XS maintains a small computational footprint, making it attractive for real-time or resource-constrained environments such as in-camera pipelines, embedded systems, and real-time video streaming. Second, as research in compression increasingly moves toward task-aware paradigms—especially in AI-driven applications—we identified an opportunity to align classical, standard-based compression with task-specific criteria. By fine-tuning sub-band gains and priorities, we can reduce the overall bitrate for particular target metrics while preserving features critical for either visual quality or subsequent computer vision tasks.

To demonstrate the benefits of this approach, we developed a procedure for automatically tuning the JPEG XS parameters using the covariance matrix adaptation evolution strategy [23] (CMA-ES). As detailed in our published work [10], this framework treats JPEG XS weight selection as an optimization problem:

1. Define a goal metric: The metric could be a standard perceptual quality measure (e.g., MS-SSIM) or a downstream artificial intelligence (AI) task performance metric (e.g., intersection over union (IoU) for semantic segmentation).
2. Encode images with candidate weights: CMA-ES proposes candidate sets of gains (G_b) and priorities (P_b), and the encoder compresses a training set of images using those parameters.
3. Evaluate fitness: The metric (CMA-ES or IoU) is measured on the decompressed images.

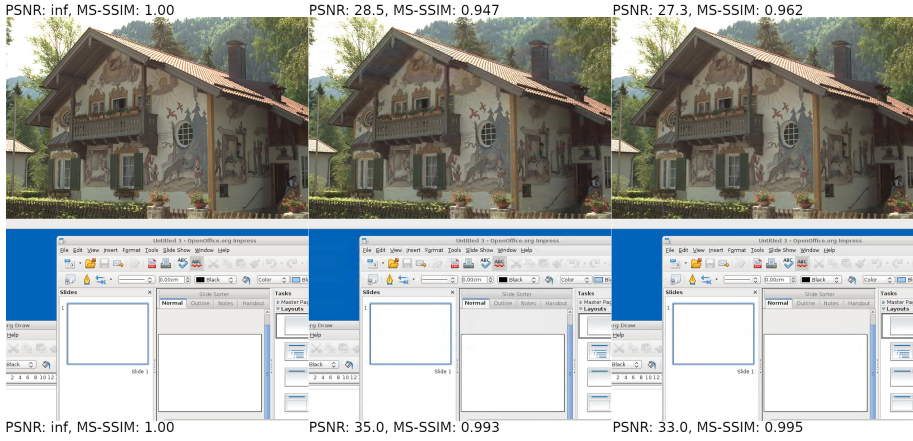
4. Iterate: Based on the outcomes, CMA-ES updates the population of candidate solutions to improve performance in subsequent iterations.

Since gains and priorities are embedded in the encoded image header, any ISO/IEC 21122-compliant decoder can properly decode these adapted streams without requiring modifications, thereby preserving compatibility.

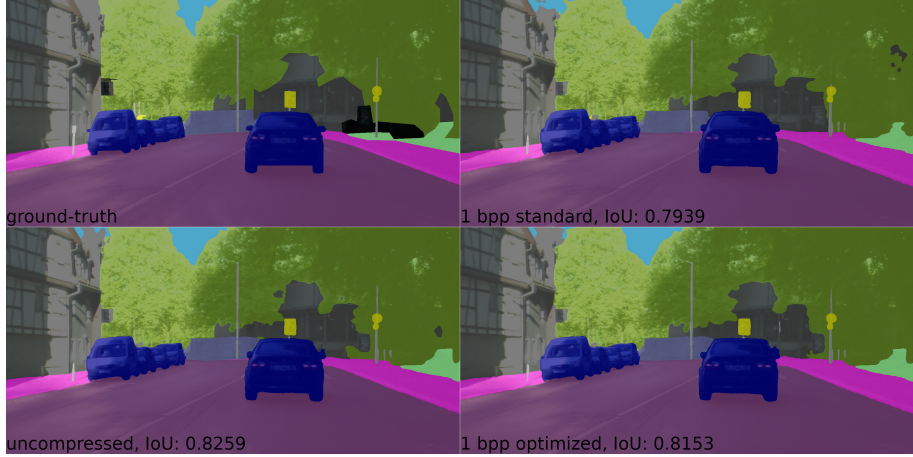
1. Visual quality improvements: Optimizing weights to maximize the MS-SSIM on photographic content resulted in bitrate savings of up to 14% relative to the default weights, while maintaining similar or better perceptual fidelity (Figure 2.7a). For synthetic images, such as screenshots, the gains were even more pronounced, demonstrating that content-aware optimization yields measurable improvements.
2. Task-aware optimization: We also optimized weights for a semantic segmentation task using the Cityscapes dataset [17]. By tuning JPEG XS weights to optimize the IoU metric, we achieved up to 59% bitrate savings at the same segmentation accuracy compared to the default weights (Figures 2.7b and 2.8). These "AI-optimized" weights ensure that critical features used by the segmentation network are retained, while less relevant details are more aggressively quantized.
3. Compatibility and simplicity: A significant advantage of this approach is that any JPEG XS-compliant decoder can interpret and decode bitstreams containing the adapted gains and priorities. The underlying standard remains intact, ensuring broad compatibility. This aligns with the thesis's focus on finding simpler, more efficient solutions—leveraging established compression frameworks and tailoring them to modern, AI-driven imaging scenarios.

Although these experiments focused on customizing a traditional (yet modern and lightweight) codec, they highlight an important takeaway: even *standard-based* techniques can be tailored to meet various goals such as visual quality and AI analysis. More significantly, they demonstrate that *content-aware* and *task-aware* compression yields clear advantages—motivating a similar mindset in fully learned compression, where there is more complete control over the encoding pipeline.

In other words, while JPEG XS presents an excellent low-complexity solution that can be fine-tuned for various tasks and contents, it remains



(a) Visual comparison of JPEG XS compression with different optimized weights at 1.00 bpp. Top: Kodak image 24. Bottom: OpenOffice screenshot. Left: uncompressed, Middle: standard weights, Right: MS-SSIM optimized weights.



(b) Semantic segmentation results on a Cityscapes image (Lindau) at 1 bpp. Top-Left: ground-truth, Top-Right: standard weights, Bottom-Left: uncompressed, Bottom-Right: optimized weights.

Figure 2.7: Examples of JPEG XS compression with adapted gains and priorities for visual quality and semantic segmentation tasks.

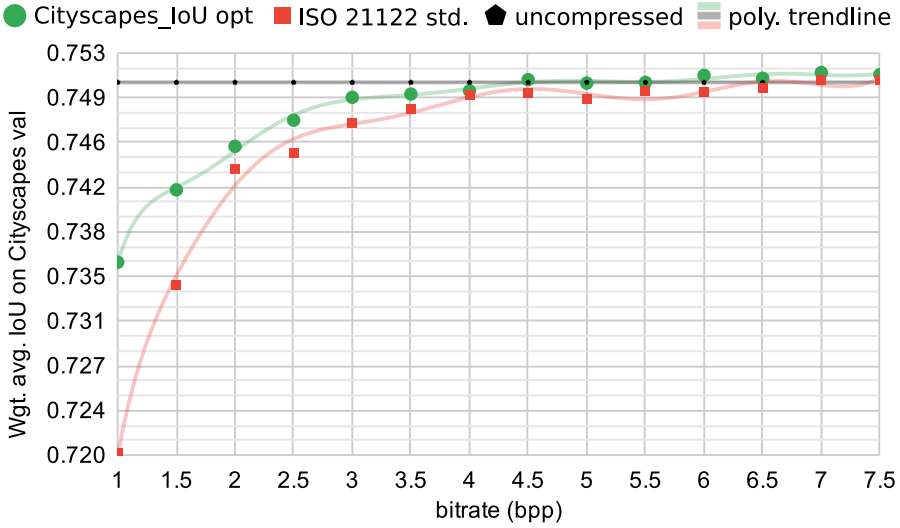


Figure 2.8: Rate-distortion performance for semantic segmentation on the Cityscapes validation set using JPEG XS. Comparison between standard JPEG XS weights and weights optimized for the IoU metric. Optimization is performed at 1.00, 3.00, and 5.00 bpp, with other bitrates interpolated.

limited by its fixed transformation approach (wavelet-based) and quantization structure. End-to-end learned image compression, which is explored in the subsequent sections, aims to take this “task-aware” philosophy further. By training entire autoencoder-based pipelines (rather than just tuning sub-band priorities), learned compression can *jointly* optimize any given task (such as denoising or feature extraction), quantization, and entropy modeling steps in a manner that is often more flexible and potentially more powerful than any fixed standard.

2.4.2 Learned compression

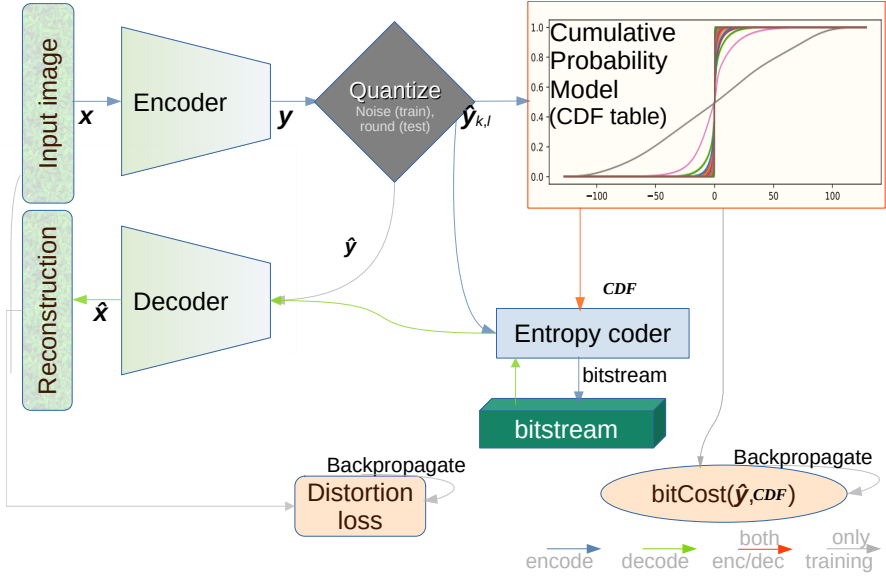
Learned image compression has emerged as a powerful alternative to traditional codecs, achieving state-of-the-art rate-distortion performance by leveraging the capabilities of deep learning. Unlike hand-engineered codecs with fixed transforms and quantization strategies, learned compression frameworks optimize the entire compression pipeline end-to-end, often resulting in more efficient representations of image data. The seminal work of Ballé et al. [6] laid the foundation for this paradigm, demonstrating that a convolutional autoencoder, coupled with a learned statistical model of the latent representation, could be jointly trained to

minimize the rate-distortion trade-off. Figure 2.9 provides an overview of this model and its core components.

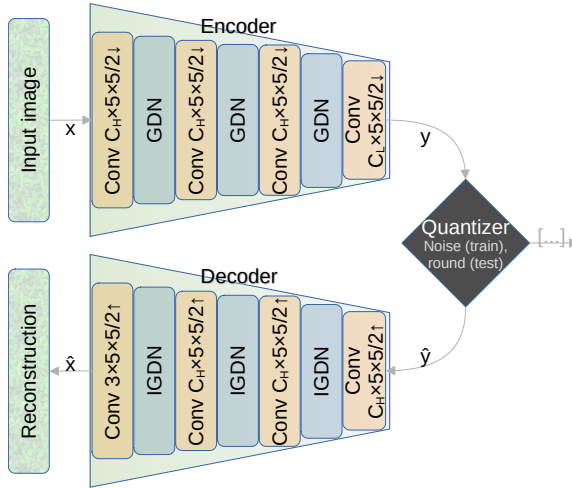
At the heart of learned compression lies the convolutional autoencoder [32], an architecture comprising an encoder network that transforms the input image into a lower-dimensional latent representation and a decoder network that reconstructs the image from this latent code. The encoder learns to extract the most salient features of the image, discarding less important information, while the decoder aims to generate a visually faithful reconstruction. Figure 2.9b shows the detailed architecture of the encoder and decoder used in Ballé et al.’s 2017 work, highlighting the convolutional layers and Generalized Divisive Normalization (GDN) [4] activation functions. The latent representation, after quantization and entropy coding, forms the compressed representation of the image.

A crucial component of learned compression is the entropy model, which estimates the probability distribution of the discrete latent variables produced by the encoder after quantization. This probabilistic model is essential for efficient entropy coding, where shorter codewords are assigned to more probable symbols, thereby minimizing the number of bits required to represent the latent code. In their foundational work, Ballé et al. [6] proposed a cumulative probability model (CPM) where each channel of the latent representation had its own cumulative distribution function (CDF). These CDFs were learned during training and, once optimized, could be evaluated at every possible value to create static lookup tables. Figure 2.10 shows an example of these learned CDFs for a trained model. These static CDF tables were then used during inference to perform efficient range encoding [42] of the quantized latent variables. This approach, where the same CDF table is reused for every spatial location within a channel, offered a significant speedup in encoding and decoding compared to using the differentiable CPM directly during inference.

Building upon this initial framework, subsequent research explored more sophisticated entropy models to further improve rate-distortion performance. Ballé et al. introduced the (hyperprior [7] (HP)) model, which uses an additional hyperencoder and hyperdecoder network to generate parameters of a conditional probability distribution for each latent variable. Specifically, the hyperprior network predicts a scale parameter for a Gaussian distribution for each variable in the latent representation. This allowed the entropy model to adapt to the local statistical properties of the latent code, leading to more accurate probability estimates and improved compression efficiency. However, this increased flexibility came at the cost of significantly increased computational com-



(a) Overview of the model. The input image is passed through an encoder, resulting in a latent representation which is then quantized. The quantized latent representation is compressed using an entropy coder guided by the cumulative probability model (CPM). The decoder reconstructs the image from the received bitstream.



(b) Detailed encoder and decoder architectures. The encoder and decoder consist of multiple convolutional layers with GDN activation functions, facilitating the extraction and reconstruction of image features.

Figure 2.9: Architecture of Ballé et al.'s 2017 learned compression model. (a) provides a high-level overview of the system, while (b) details the specific layers and activation functions within the encoder and decoder networks.

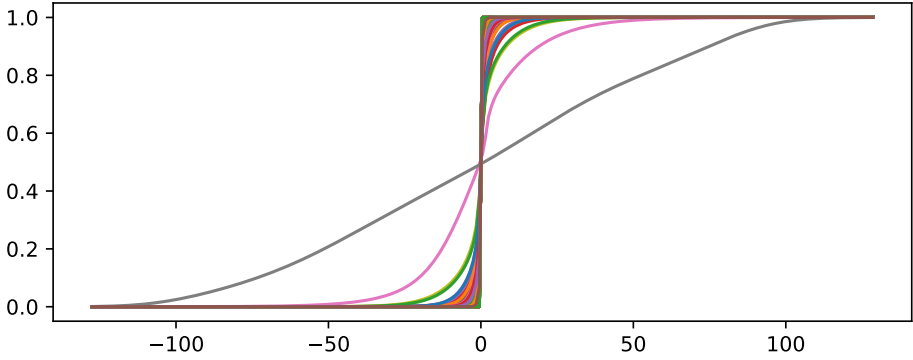


Figure 2.10: Learned cumulative distribution function (CDF) for the 256 channels of a fully trained Ballé et al. (2017) compression model. Each line represents the CDF for a single channel, illustrating the learned probability distributions of the latent variables.

plexity during entropy coding and decoding, as different CDFs (based on the predicted scale parameter) needed to be considered for each individual variable across channels and spatial locations. As detailed in Chapter 3, pre-computing and looking up CDFs for a discrete set of scale values mitigates some of this complexity, but the fundamental need for per-variable parameterization remains a computational bottleneck.

Other advancements in learned compression have focused on further refining the entropy model. Minnen et al. [46] proposed a method for locally adapting the entropy model by selecting from a set of predefined probability distributions. While this approach offered some improvements, it did not achieve the same level of performance as the hyperprior model. Later, Minnen et al. [48] introduced an autoregressive context model that sequentially predicts the probability of each latent variable based on previously decoded variables. This method achieved state-of-the-art rate-distortion performance but introduced a substantial increase in runtime complexity due to the sequential nature of the decoding process. Subsequent works, such as [47], continued to explore increasingly complex entropy models, often involving autoregressive components or Gaussian Mixture Models [15], further pushing the boundaries of compression efficiency at the expense of computational cost.

In contrast to these trends towards more complex entropy models, our work builds upon the foundations laid by Ballé et al. [6] and explores the potential of simplified entropy coding approaches. As detailed in Chapter 3, we demonstrate that by utilizing a set of multiple learned prior distributions stored in static tables of cumulative distribution func-

tions, it is possible to achieve comparable rate-distortion performance to more complex models like the hyperprior [7] (HP) architecture while significantly reducing the computational burden of entropy coding and decoding. This approach directly addresses the complexity concerns associated with per-variable parametric priors, offering a more computationally efficient alternative for applications where encoding and decoding speed are critical. The choice of Ballé et al.’s (2017) architecture as a starting point is motivated by its conceptual simplicity and its role as a common reference point in the learned compression literature [7, 48, 37, 47].

The emergence of transformer-based architectures in computer vision has also influenced the field of learned compression. Models like those proposed by Lu et al. [39] and Arezki et al. [3] leverage self-attention mechanisms to capture long-range dependencies in images, potentially leading to more efficient latent representations. While promising, these transformer-based approaches often introduce their own set of computational complexities.

The development of learned compression has also spurred interest in joint denoising and compression (JDC), recognizing that noise inherently hinders compression efficiency. By integrating denoising capabilities directly into the compression framework, models can learn to suppress noise during the encoding process, leading to more compact representations and improved overall rate-distortion performance. This concept is central to this thesis and will be explored in detail in Chapters 4 and 5.

In summary, learned image compression has evolved rapidly, with advancements in entropy modeling and network architectures leading to significant improvements in compression efficiency. However, this progress has often come at the cost of increased computational complexity, particularly in the entropy coding and decoding stages. This thesis explores approaches that aim to strike a better balance between rate-distortion performance and computational cost, specifically by investigating simplified entropy coding techniques and the integration of denoising within the compression framework.

3

End-to-end optimized image compression with competition of prior distributions

IMAGE compression typically consists of a transformation step (including quantization) and an entropy coding step that attempts to capture the probability distribution of a transformed context to generate a smaller compressed bitstream. Entropy coding ranges in complexity from simple non-adaptive encoders [63, 52] to complex arithmetic coders [53] with adaptive context models [41, 29]. The entropy coding strategy has been revised to address the specificities of learned compression. More specifically, for recent works that make use of a convolutional autoencoder [5] (AE) as the all-inclusive transformation and quantization step, the entropy coder relies on a cumulative probability model (CPM) trained alongside the autoencoder [6]. This model estimates the cumulative distribution function (CDF) of each channel coming out of the AE and passes these learned CDFs to an entropy coder such as range encoding [42].

Such a simple method outperforms traditional codecs like JPEG2000 but work is still needed to surpass complex codecs like BPG. Johannes Ballé et al. (2018) [7] proposed analyzing the output of the convolutional encoder with another AE to generate a floating-point (FP) scale parameter that differs for every variable that needs to be encoded by the entropy coder, thus for every location in every channel. This method has been widely used in subsequent works but introduces substantial complexity

in the entropy coding step because a different CDF is needed to encode every variable in the latent representation of the image, whereas the single AE method by Ballé et al. (2017) [6] reused the same CDF table for every latent spatial location.

Our work uses the principle of competition of experts [50, 40] to get the best out of both worlds. Multiple prior distributions compete for the lowest bit cost on every spatial location in the quantized latent representation. During training, only the best prior distribution is updated in each spatial location, further improving the prior distributions specialization. CDF tables are embedded in the model as look-up tables (LUTs) at the end of training. Hence, at testing, the CDF table resulting in the lowest bitcost is assigned to each spatial location of the latent representation. The RD performance obtained is comparable to that obtained with a parametrized distribution [7], yet the entropy coding process is greatly simplified since it does not require a per-variable CDF and can build on LUTs rather than the computation of analytical distributions.

3.1 Related works

Learned image compression has emerged as a transformative approach, offering state-of-the-art performance by optimizing the entire compression pipeline through deep learning. This shift was largely pioneered by Ballé et al. [6], whose convolutional autoencoder framework introduced the first end-to-end optimized model. Their approach leveraged a learned entropy model to estimate the bit cost of latent features, enabling joint optimization of rate and distortion. A more detailed overview of learned compression methods can be found in Section 2.4.2.

Following this foundational work, subsequent research sought to improve RD performance by refining the entropy model. A key advancement came with the introduction of hyperpriors by Ballé et al. [7], which improved RD performance by conditioning the entropy model on a set of side-information generated by a hyper-encoder. This method allowed for adaptive probability distributions at each spatial location and channel in the latent space, at the expense of increased complexity in the entropy coding process.

Subsequent models sought to further refine this idea by capturing local spatial dependencies. Minnen et al. [48] incorporated an autoregressive context model, achieving state-of-the-art performance but introducing significant runtime overhead due to the sequential nature of autoregressive decoding. This complexity has prompted ongoing research into reducing the computational cost of learned compression

without compromising RD performance.

Alongside these developments, other works have explored the use of multiple fixed priors to model spatial variations in the latent space. Minnen et al. [46] proposed a model that selects from a set of predefined distributions after encoding. While this approach offered an alternative to hyperprior methods, the separation of prior selection from the encoder’s optimization limited its effectiveness.

This thesis introduces an approach that leverages the competition of expert priors, jointly optimizing multiple learned priors during training. By selecting the most suitable prior at each spatial location across all channels, this method enables spatial adaptation without the need for additional hyper-encoders or autoregressive components, providing a simpler and computationally efficient alternative.

3.2 Competition of prior distributions

Our proposed method introduces competitions of expert [50, 40] prior distributions: a single AE transforms the image and a set of prior distributions are trained to model the *CDF* of the latent representation in each spatial location. For each latent spatial dimension the CDF table which minimizes bit cost is selected; that prior is either further optimized on the features it won in the training mode, or its index is stored for decoding in the inference mode. This scheme is illustrated in Figure 3.1, a set of 16 optimized CDF tables is shown in Figure 3.2, and three sample images are segmented by “winning” CDF table in Figure 3.3.

All prior distributions are estimated in parallel by considering N_{CDF} CDF tables, and selecting, as a function of the encoded latent spatial location, the one that minimizes the entropy coder bitcount. The CDF table index is determined for each spatial location by evaluating each CDF table in inference, per Algorithm 1. This can be done in a vectorized operation given sufficient memory. During training the CPM is evaluated instead of CDF tables such that the probabilities are up to date and the model is differentiable, and the bit cost is returned as it contributes to the loss function.

The cost of CDF table indices has been shown to be neglectable in Table 3.1 (and lower in practice using lossless compression) due to the reasonably small number of priors, which in turns results from the fact that little gain in latent code entropy has been obtained by increasing the number of priors.

In all our experiments, the AE architecture follows the one in Ballé et al. (2018) [7], without the HP, since we found that the AE from [7] offers

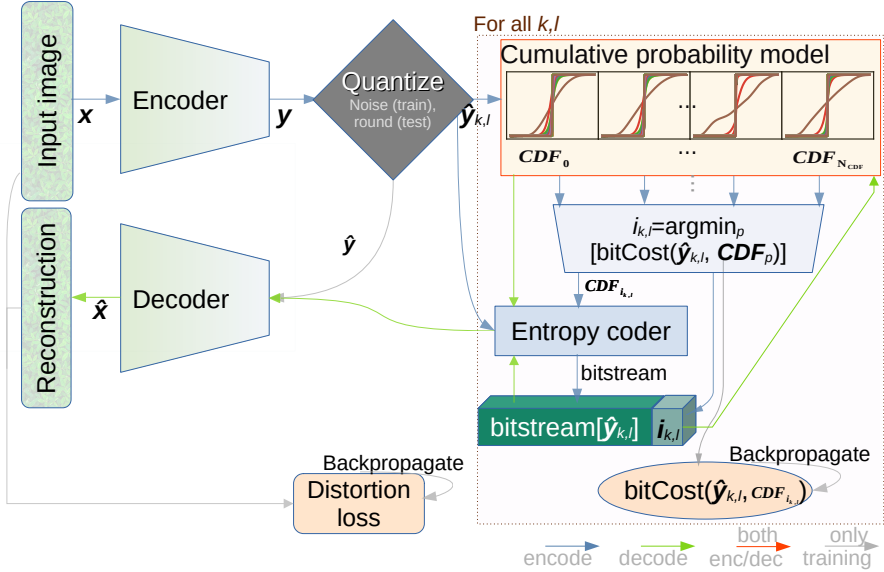


Figure 3.1: AE compression scheme with competition of prior distributions. The AE architecture is detailed in [7, Fig. 4]. The indices i denote the indices of CDF tables that minimizes the bitcount for each latent spatial dimension. Loss = Distortion + λ bitCost.

Algorithm 1 CDF tables selection at inference

```

procedure GET_CDF_INDICES( $\hat{y}, \mathcal{D}$ )
  for  $0 \leq k < H_L$  and  $0 \leq l < W_L$  do
     $i[k, l] \leftarrow \arg \min_{i < N_{\text{CDF}}} | -\log_2 (CDF_i(\hat{y}[k, l] + 0.5) - CDF_i(\hat{y}[k, l] - 0.5)) |$ 
  end for
  return  $i$ 
end procedure
  
```

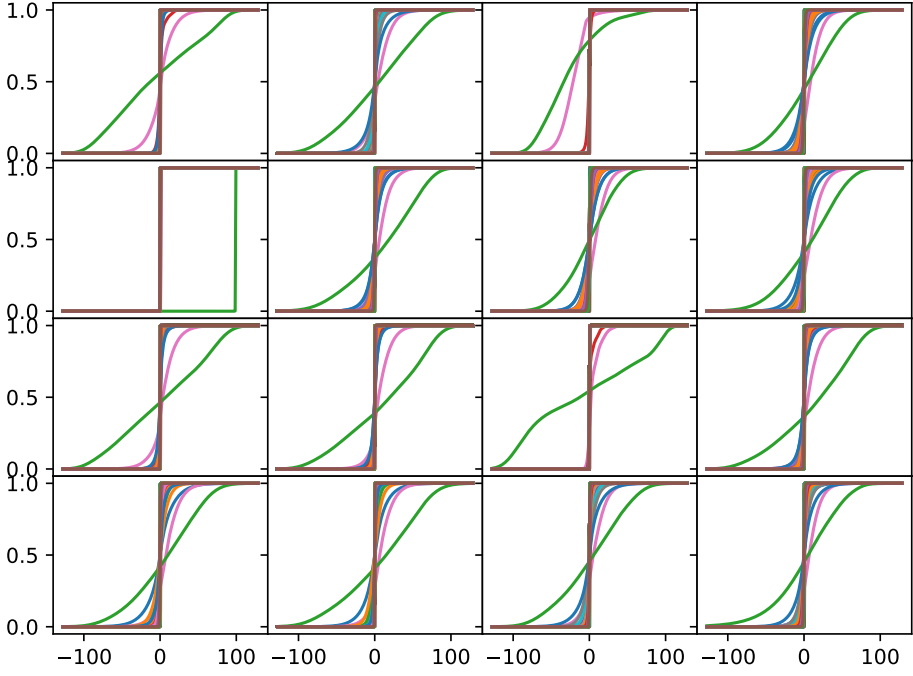


Figure 3.2: We observe some diversity among the 16 CDF tables learned by a network trained with MSE loss and $\lambda = 4096$ (high bitrate). Each box presents a CDF table and each colored line corresponds to the *cdf* of one of 256 latent channels. The best fitting CDF table is selected for each latent spatial location.

Table 3.1: Maximum uncompressed bitrate required to store prior indices based on the number of distributions ($\log_2(N_{\text{CDF}}) \div 16^2$ bpp)

number of distributions	bitrate (bpp)
256	0.03125
128	0.02734
64	0.02344
32	0.01953
16	0.01563
8	0.01172
4	0.00781
2	0.00391
1	0

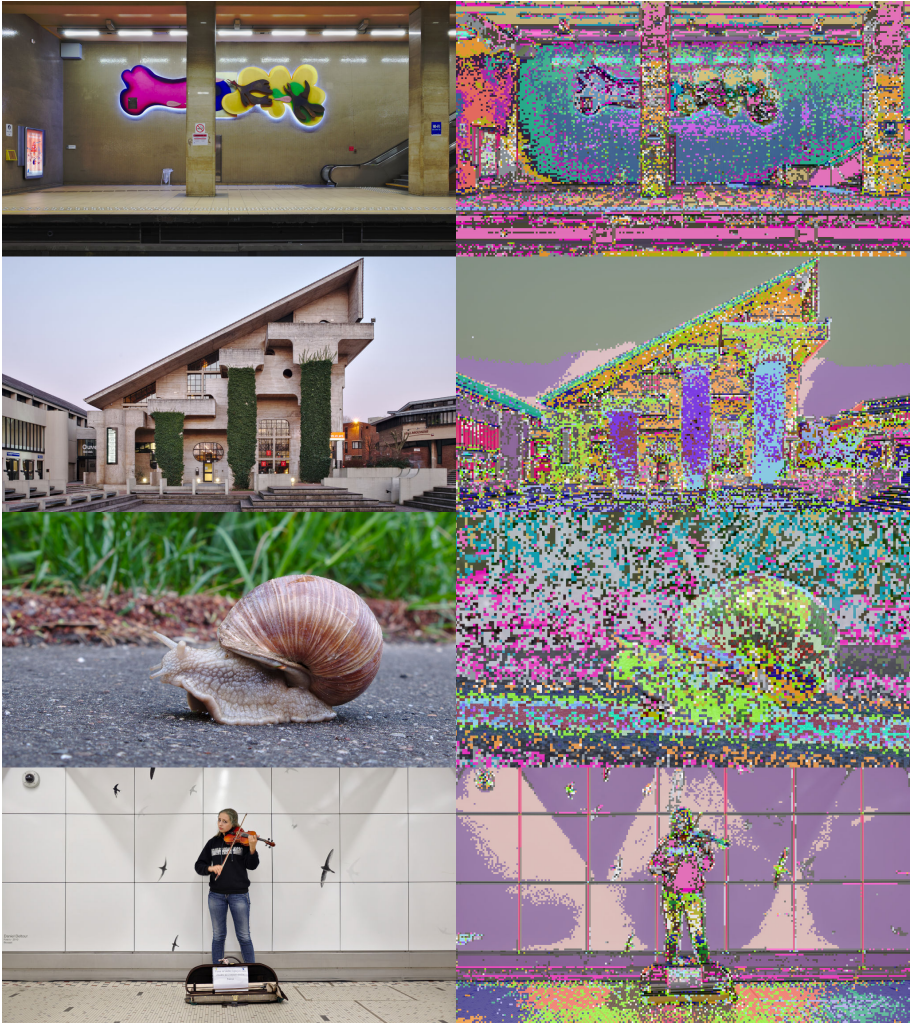


Figure 3.3: Segmentation of four test images: each distinct color represents one of 64 CDF tables used to encode a latent spatial location (16×16 pixels patch).

better RD than the one described in Ballé et al. (2017) [6], even with a single CDF table. A functional training loop is described in Algorithm 2.

Algorithm 2 Training loop

```

 $y \leftarrow \text{model.Encoder}(x)$ 
 $\hat{y} \leftarrow \text{quantize}(y)$ 
 $\hat{x} \leftarrow \text{clip}(\text{model.Decoder}(\hat{y}), 0, 1)$ 
 $\text{distortion} \leftarrow \text{visualLossFunction}(\hat{x}, x)$ 
for  $0 \leq k < H_L$  and  $0 \leq l < W_L$  do
     $\text{bitCost}[k, l] \leftarrow \min_{i < N_{\text{CDF}}} | -\log_2 (\text{CPM}_i(\hat{y}[k, l] + 0.5) - \text{CPM}_i(\hat{y}[k, l] - 0.5)) |$ 
end for  $\triangleright$  CPM is the differentiable version of CDF
 $\text{Loss} \leftarrow \text{distortion} \times \lambda + |\text{bitCost}|$ 
 $\text{Loss.backward}()$ 

```

3.3 Experiments

3.3.1 Method

These experiments are based on the PyTorch implementation of Ballé et al. (2018) [7] published by Liu Jiaheng [36, 37]. To implement our proposed method, the HP is omitted in favor of competition of expert prior distributions. The CPM is that defined in [36] with an additional N_{CDF} dimension to compute all CDF tables in parallel. Theoretical results are verified using the torchac range coder [44, 43, 42]. A functional training loop is described in Algorithm 2, and source code is provided on <https://github.com/trougnouf/Manypriors>. To ensure that all priors get an opportunity to train, the prior distributions that have not been used for at least fifty steps are randomly assigned to spatial locations with largest bitcounts, to be forced to train. The resulting priors utilization is illustrated in Figure 3.4

The Adam optimizer [31] is used with a starting learning rate (LR) of 0.0001 for the AE and 0.001 for the CPM. Performance is tested every 2500 steps in inference mode on the validation set, and the LR is decayed by a factor of 0.99 if the performance have not improved for two tests. Reported performance is the one of the model that minimizes $(\text{visualLoss} \times \lambda + \text{bitCost})$ on the validation set at the end of training. Base models are trained for six million steps at $\lambda = 4096$ with the MSE loss. Smaller λ values and MS-SSIM models are trained for four million steps

3.3. Experiments

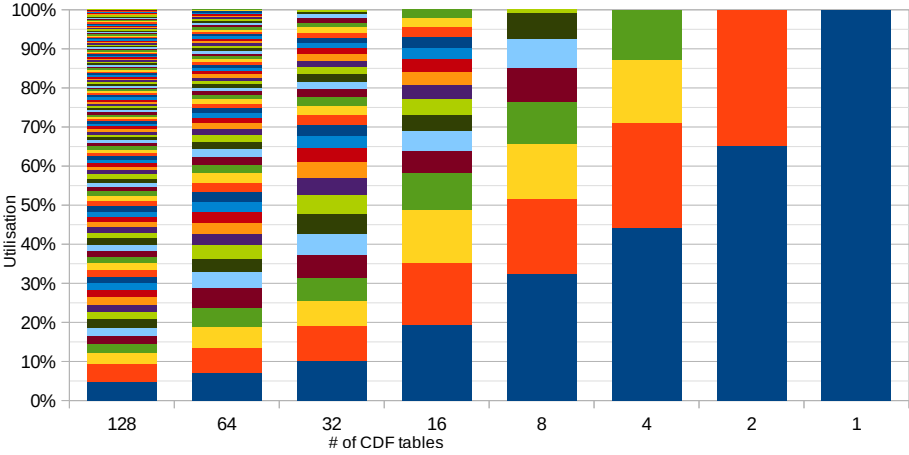


Figure 3.4: CDF tables utilization on the CLIC pro test set. Each color represents a CDF table.

starting from the base model with their LR and optimizer reset. All models use $C_H = 192$ (hidden layers channels) and $C_L = 256$ (output channels) such that a single base model is needed for each prior configuration. The training and validation dataset is made of free-license images from Wikimedia Commons; mainly Wikimedia Commons Featured Pictures [1] (COM:FP) which consists of 13928 images of the highest quality. The images are cropped into 1024^2 pixels patches on disk to speed up further resizing, then they are resized on-the-fly by a random factor down to 256^2 pixels during training. A batch size of 4 patches is used. The kodak set ¹ is used as a validation set and the CLIC professional test dataset ² is used for testing.

The RD curve of our “multiprior” model is compared with that of the HP model, which is trained from scratch using Liu Jiaheng’s PyTorch implementation [36, 37]. Liu Jiaheng’s code differs slightly from the paper’s definition [7] in that a Laplace distribution is used in place of the normal distribution to stabilize training. Complexity is measured as the number of billion multiply-accumulate operations (GMAC) using the ptflops counter [58] and the number of memory lookup operations is calculated manually.

¹<https://r0k.us/graphics/kodak/>

²<https://clic.compression.cc/2021/tasks/>

3.3.2 Results

The PSNR RD curve measured on the CLIC professional test set² is shown on top of Figure 3.5. The performance of a 64-priors model is in line with that of the HP model : they both perform slightly better than BPG at high bpp, and achieve significantly better RD than the single-prior model. In the middle plot, the RD value at $\lambda = 4096$, the highest bitrate, is shown for 1, 2, 4, 8, 16, 32, 64, and 128 prior distributions. 128-priors offer marginal gains and costs an increased training time ($\times 1.5$) and encoding time. MS-SSIM performance of fine-tuned models is shown in the bottom of Figure 3.5; the 64-priors model still performs similarly to [7], and both learned compression models benefit from this more perceptual metric compared with traditional codecs. A visual comparison of images compressed with the MSE loss ($\lambda = 512$) and the equivalent bitrate settings in conventional codecs is shown in Figure 3.6.

Computational complexity of our Manypriors has been compared to the one of the HP model [7]). This complexity is expressed in GMAC for the neural network parts and number of memory lookup operations. It is summarized in Table 3.2. The lack of a HP AE saves 3 % to 6 % GMAC, depending on whether only the HP decoder (image decoding) or the whole HP codec (image encoding) is used.

Decoding with the Manypriors scheme is greatly simplified compared to [7] because the CDF tables generation process takes the optimal indices stored as side-information and looks up one static CDF table per latent spatial dimension, that is C_L (typically 256) fewer lookups than with a HP. During encoding, the Manypriors scheme must lookup every latent variable with every CDF table in order to determine the most cost effective CDF tables. This results in N_{CDF} (typically 64) times more lookup operations than the HP scheme overall, although these lookup operations are relatively inexpensive because only two values are retrieved (variable ± 0.5). Moreover, it is challenging to make an accurate CDF LUT for the HP scheme, because quantizing the distribution scale parameter reduces the accuracy of the resulting CDFs, negatively impacting the bitrate. This challenge is exacerbated when the distribution has multiple parameters [48] or a mixture of distributions [15] is used. In Figure 3.5, LUTs are replaced by accurate but complex Laplace distribution computation for the HP scheme in order to maximize the reported RD performance.

Time complexity is measured for every step on CPU, where it can be reliably profiled due to synchronous execution. It is summarized in Table 3.3 with the following distinct sub-categories: NN is the time spent in the AE, CDF generation is the time spent building the CDF

3.3. Experiments

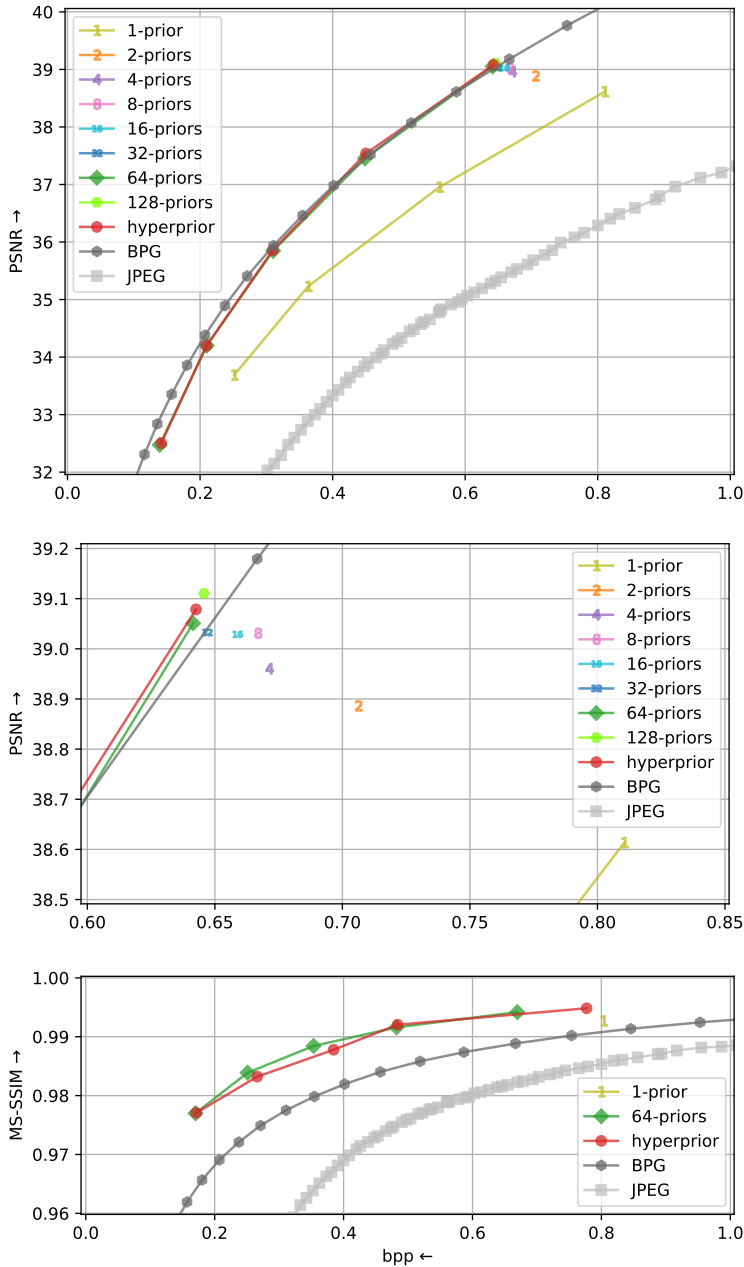


Figure 3.5: Top: PSNR RD curve of a 64-priors model on the CLIC pro. test set, compared with the HP model [7], and the BPG and JPEG codecs. Middle : Zoom in on models with 1, 2, 4, 8, 16, 32, and 64 priors. Bottom: MS-SSIM RD curve.

3: End-to-end optimized image compression with competition of prior distributions



Figure 3.6: Visual comparison of Larry the cat compressed with learned ($\lambda = 512$) and conventional methods. Top-left: uncompressed, top-middle: JPEG (PSNR: 29.3, 0.224 bpp), top-right: BPG (PSNR: 32.9, 0.217 bpp), bottom-left: 1-prior (PSNR: 32.4, 0.252 bpp), bottom-middle: hyperprior (PSNR: 32.8, 0.217 bpp), bottom-right: 64-priors/ours (PSNR: 32.9, 0.218 bpp)

Table 3.2: Complexity of the HP model [7]) compared to Many priors (ours), expressed in GMAC for the neural network parts and number of memory lookup operations (* or parametrized Laplace CDF generations in full-precision) for the CDF tables generation, to process a 4K image.

(\downarrow)			Hyperprior	Many priors	ratio MP÷HP
Encoding	<u>GMAC</u>	main encoder	770	770	
		hyper encoder	24.0		
		hyper decoder	24.0		
		total	817	770	0.942
	<u>Lookups</u>	indices		531 M	
		CDF	8.29 M *	32.4 K	
		total	8.29 M *	531 M	$N_{\text{CDF}} = 64$
Decoding	<u>GMAC</u>	hyper decoder	23.2		
		main decoder	770	770	
		total	793	770	0.971
	<u>Lookups</u>	CDF (total)	8.29 M *	32.4 K	$\frac{1}{C_L} = \frac{1}{256} = 0.004$

3.3. Experiments

Table 3.3: Breaking down the image encoding and decoding time, in second. Image: 4.5 megapixel (MP) snail. central processing unit (CPU): AMD Ryzen 7 2700X. Time avg. of 50 runs.

(↓)		Hyperprior (Ballé2018)	64-priors (ours)	ratio (ours÷HP)
Encoding	NN encode: main + hyperprior	3.81 + 0.41	3.79 + 0.00	0.90
	entropy encode, main + hyperprior	0.15 + 0.02	0.15 + 0.00	
	CDF: select indices + gather tables	0.00 + FP: 15.95 LUT: 5.66	1.90 + 0.81	FP: 0.17 LUT: 0.48
	encode (total)	FP: 20.33 LUT: 10.04	6.65	FP: 0.32 LUT: 0.66
	NN decode : main + hyperprior	10.66 + 0.34	10.50	0.95
Decoding	CDF : gather tables	FP: 15.95 LUT: 5.66	0.81	FP: 0.05 LUT: 0.14
	entropy decode : main + hyperprior	0.24 + 0.02	0.24	0.92
	decode (total)	FP: 27.21 LUT: 16.92	11.54	FP: 0.42 LUT: 0.68

tables for a specific image, and entropy is the bitstream generation. All operations are done using the PyTorch framework in python, except for entropy encoding which makes use of the torchac range coding library [44, 43], written in C++, and the prior indices are compressed using the LZMA library [16]. The total encoding time of the 64-priors model is 0.32 time that of the HP model and the decoding time is 0.42 times that of the HP model. The timing is more significant when it is broken down by sub-category because each component has a different response time depending on the hardware (and software) architecture in place. The AE (“NN”) encoding time is 0.90 that of the HP scheme and decoding time is 0.95 time as much as the HP. Both the hyper-encoder and hyper-decoder are called during encoding, thus it appears that each part of the HP sub-network costs 5 % of the AE time. The time taken to build the CDF tables for the HP model was measured both by estimating the per-variable Laplace distributions (“full-precision”) and with a quantized scale parameter LUT. In any case, finding the best indices of a 64-priors model appears to be relatively inexpensive and the total CDF tables generation time is only 0.17 to 0.48 that of the HP model (depending on whether the HP model uses full-precision or LUT) for encoding. During decoding, the 64-priors model spends 0.05 to 0.14 as much time building the CDF tables as the HP model, because the optimal CDF table indices have already been determined during encoding and they are included in the bitstream.

3.4 Conclusion

Convolutional autoencoders trained for compression are optimized for both rate and distortion. Rate is estimated with a cumulative probability model, which in turns generates a CDF for every latent variable to be encoded. A single CDF per latent channel is not sufficient to capture the statistics at the output of the encoder, nor to allow the encoder to express a wide variety of features. To support multiple statistics, the HP parametrizes a standard distribution, but this introduces a great deal of complexity in the entropy coding stage because the CDF differs for every latent variable to be encoded. The proposed method uses multiple prior distributions working as a competition of experts to capture the relevant features which they specialize on. This approach is advantageous because the learned CDF tables are stored in a static LUT once training is finished, and a model trained with 64 prior distributions performs with a similar RD as one trained with a HP sub-network. Moreover, a learned CDF table includes the CDF for all channels in the latent code. Hence, accessing the CDF table for a spatial location provides the CDF for each of its channels and the number of lookups is reduced to the number of latent spatial locations. In our experiments, CDF tables generation in the encoding step takes 0.17 to 0.48 as much time with a 64-priors model as it does with the HP model (depending on the precision of the HP model). This ratio is lowered to 0.05 to 0.14 during decoding because the prior indices have already been determined during the encoding.

4

On the importance of denoising when learning to compress images

IMAGE sensors capture noise along with useful image information. This noise increases with the camera's ISO sensitivity setting, but noise is virtually always present to some extent and it is both incompressible and undesirable. Lossy image compressors inherently perform some image denoising because removing random noise is often the most effective way to reduce entropy in a signal, but without proper training (or algorithm design) the resulting image size is still inflated and the results look sub-par, as shown in Figures 4.1 and 4.2 and by our experiments in Figure 4.6. This increase in bitrate is readily observable in both conventional and learned codecs. A learned lossy image compression scheme is trained by forwarding the image through an autoencoder [32] (AE) and backpropagating from the loss, whose components are the bitrate and the distortion [6]. The bitrate is computed from an optimized, i.e. trained, cumulative distribution function, and the distortion quantifies the difference between the output of the autoencoder and the input image, typically by computing the MSE.

Any image compression scheme can attain better rate-distortion by having the noise removed first.

An image denoiser can typically be trained to reconstruct clean images from noisy inputs, using a dataset of paired images where static scenes are captured using a progressively faster shutter speed [9].

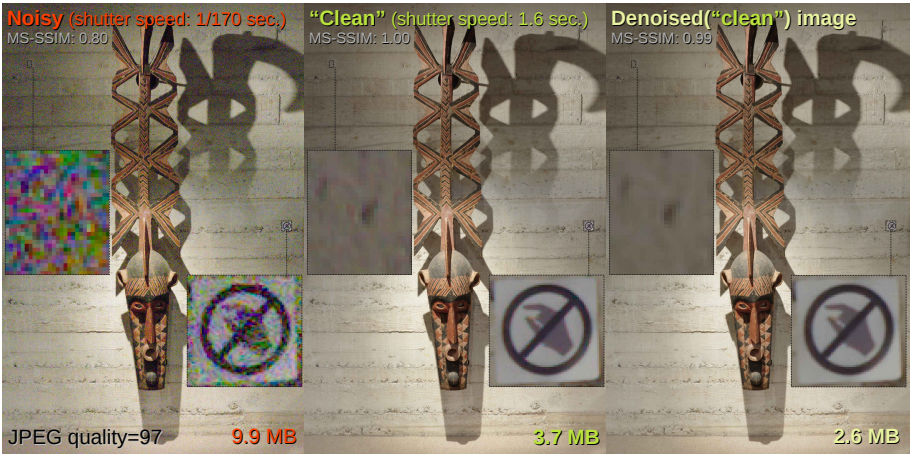


Figure 4.1: Noise is both incompressible and undesirable, as shown in this clean–noisy image pair from the Natural Image Noise Dataset [9] (NIND), where the same scene has been captured with increasingly faster shutter speed. The JPEG compressed (Q=97) image size increases from 3.7 MB for the ground-truth to 9.9 MB when encoding the noisy image. Even the ground-truth image (center) contains some background noise and artifacts such as chromatic aberrations, and JPEG compresses it down to 2.6 MB when it is denoised with a trained denoiser from [9] prior to compression (left).

In this work, we consider joint compression and denoising. Adding a denoising functionality essentially comes down to feeding the network with a potentially noisy image and comparing its output with a clean image that may have a better quality than what was initially input to the network. The goal is to generate an image that is of higher quality than the one used as input, while decreasing the necessary bitrate close to that of a clean image. Meanwhile, there is no added complexity because the inference process and the network architecture remain unchanged. The network is trained with both noisy images and some clean images as input such that it removes noise while retaining the ability to compress clean input images efficiently.

Our experiments analyze the impact of image noise on the rate-distortion of different standard and learned compression methods, the benefit of performing denoising prior to compression, and denoising while compressing. Our original supervision strategy, introduced to promote the reconstruction of clean images when the learned codec is fed with noisy ones, appears to be effective. The resulting joint denoising and compression models perform properly on clean images as well as noisy ones, effectively replacing a standard compression models for general-purpose image compression and substantially improving the rate-distortion on noisy images. As illustrated in the second line of Figure 4.2 (comparison between second and third columns), it is shown to significantly improve rate-distortion performance (using non-noisy images as ground-truth) compared to relying on the implicit noise removal induced by the conventional adoption of a perception-based loss function during training. It also reaches slightly better rate-distortion than a computationally heavy two-step procedure, involving one AE for denoising, followed by another AE-based network for compression.

This chapter is organized as follows: Section 4.1 summarizes the work on which this chapter builds. The main concepts behind our joint denoising and compression supervision are introduced in Section 4.2. The implementation details are given in Section 4.3.1 followed by the results, and Section 4.4 summarizes the impact of the present work.

4.1 Background

Learned image compression has emerged as a powerful alternative to traditional codecs, achieving state-of-the-art rate-distortion performance by leveraging deep learning techniques. As detailed in Section 2.4, methods based on convolutional autoencoders learn efficient image representations and their probability distributions.



Figure 4.2: Visualization of a clean (top) and noisy (bottom) test image from NIND [9]. From left to right: (i) ground-truth/noisy input, (ii) compression autoencoder trained with standard supervision [11], relying on the adoption of a perception-based loss function to mitigate the impact of noise, (iii) our proposed joint denoising and compression model trained with Natural Image Noise Removal supervision using both clean and low noise images (“JDC-Cn.8”).

However, the presence of noise in images poses a significant challenge to compression algorithms. Noise is inherently incompressible and increases the entropy of the image, leading to suboptimal rate-distortion performance. While lossy compression inherently discards some information, including noise, handling it explicitly within the compression framework offers potential for further improvements.

One approach is to perform denoising as a pre-processing step before compression. As discussed in Section 2.3, effective denoising algorithms exist, and applying them before standard compression techniques can improve the final rate-distortion. However, this sequential approach might not be optimal as the denoising is performed without considering the subsequent compression stage.

More recently, efforts to jointly learn denoising and compression have been explored by training a single model to perform both tasks. Testolina et al. [61] explored this by extending the decoder of a learned compression model to handle noisy inputs. However, their proposed approach using synthetic noise and focusing solely on the decoder yield no improvement in rate-distortion.

This chapter builds upon the foundations of learned image compression and addresses the critical interplay between denoising and compression. We propose a joint learning strategy that leverages the characteristics of real-world noise to achieve improved rate-distortion performance compared to compressing noisy images directly or using separate denoising and compression pipelines. Our approach focuses on training the encoder to produce a latent representation that inherently suppresses noise, leading to more efficient compression.

4.2 Jointly learned denoising and compression

An effective denoising network can be trained to reconstruct clean images given noisy images as input, using a dataset of paired noisy-clean images such as NIND [9] and SIDD [2]. We propose to adopt a similar principle to train an autoencoder originally designed for image compression.

A joint denoising and compression autoencoder [11] is trained to generate a clean image from either a matching noisy image in a paired dataset or from the same clean image. The aim is to obtain a decoded image whose quality is potentially higher than that of the input image, while saving the space that would otherwise be wasted in encoding noise. Different methods are proposed to train such a joint denoising and compression model using Natural Image Noise Removal (NINR)

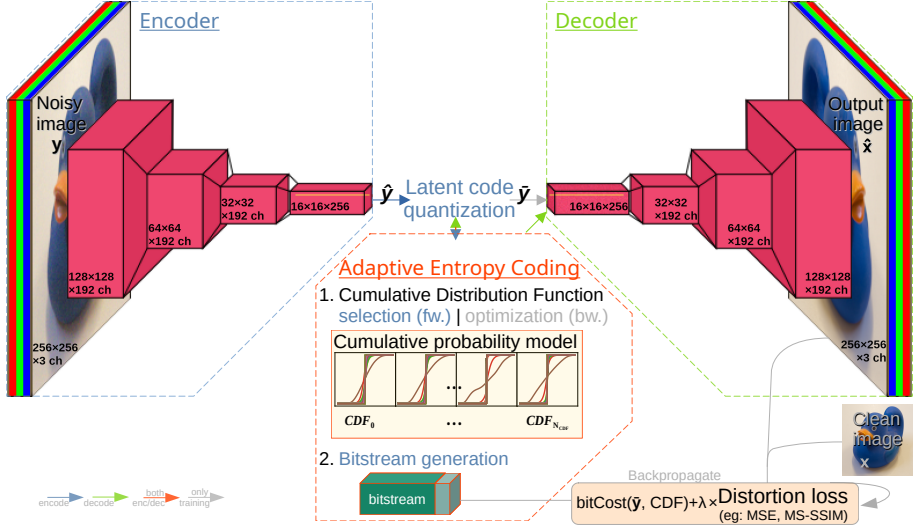


Figure 4.3: Denoising and compression joint training: the distortion loss is computed between the reconstructed image $\hat{x}$ and a clean image x . The input image y may be noisy. The network [11] is made of four (transposed) convolutions with stride of 2 and kernel size of 5, each followed by a GDN activation [4] except for the last (transposed) convolution. (Best viewed in color.)

supervision. They are described in Section 4.2.1 and Figure 4.3 illustrates the general training process.

4.2.1 Our proposed NIN supervision strategies

Four different strategies are envisioned and compared to implement our novel Natural Image Noise Removal (NINR) supervision paradigm. They are listed in Table 4.1 and described below.

Noisy Pairs (JDC-N) The simplest joint denoising and compression implementation consists of training with all noisy–clean image pairs available in the dataset(s).

Clean and Noisy Pairs (JDC-CN) This method considers some clean–clean image pairs, in addition to the noisy–clean image pairs, to ensure that the network’s performance does not degrade when the input images contain no unwanted noise. The dataset of clean images can be selected as a set of images which has been assessed and promoted by human reviewers, such as the COM:FP, then further refined by eliminating images whose metadata indicates a high ISO value in order to ensure

Table 4.1: Data pairs considered in this chapter to train a joint denoising and compression model. JDC-Cn is also referred to with its training noise threshold (e.g. JDC-Cn.8 is trained with $\text{MS-SSIM} \geq 0.8$, see the text for details).

Method	Training Input	Expected Output
JDC-N	Noisy image from clean-noisy paired dataset [9]	Clean ground-truth
JDC-CN	Noisy image from clean-noisy paired dataset, clean image from high-quality dataset (COM:FP)	Clean ground-truth, clean input
JDC-Cn	Weakly noisy image from clean-noisy paired dataset, clean image from a high-quality dataset	Clean ground-truth, clean input
JDC-UD	Arbitrary input image (COM:FP, [9])	Provided by a universal denoiser
Testolina [61]	Clean image from high-quality dataset + artificial noise	Clean input

the absence of noise.

Clean and Low-noise Pairs (JDC-Cn) To specialize the model to the most frequent input image noise levels, we have also considered placing a threshold on the training data noise level. Our experiments reveal that it is beneficial to filter out the most noisy input training images because the overall rate-distortion degrades when the network is trained to perform more extreme denoising. Such extreme denoising would require added complexity on the encoder and, although possible, extreme denoising is outside the scope of a combined denoiser whose aim is to improve rate-distortion by removing noise that is inherently present in most photographs rather than learning to see in the dark, as proposed in [14]. The paired image noise dataset is analyzed prior to training such that the MS-SSIM score between each noisy crop and its clean ground-truth is stored in a file, and the training dataset can be initialized such that all training crops exceed a set quality threshold. The effect of different noise thresholds is analyzed in the ablation study.

Building Pairs from a Universal Denoiser (JDC-UD) A fourth train-

ing method consists of running a pre-trained blind denoising model [54, 9] on all the training data to generate the ground-truth images, and computing the training loss between the input images and the denoised images. This method effectively performs knowledge distillation [24] from a powerful universal denoising network to the joint denoising and compression network. All input images are considered noisy and the training dataset is virtually limitless because the ground-truth images are generated (in advance), thus entire image datasets are used without filtering.

4.3 Experiments

4.3.1 Practical implementation details

These experiments are based on the PyTorch implementation of the autoencoder base codec introduced in [11]. Source code is provided as Supplementary Material and available on <https://github.com/trougnouf/compression>. The training loss of the compression autoencoder is computed as $\text{Loss} = \text{bitrate}(\hat{x}) + \lambda \times \text{MSE}(\hat{x}, x)$, where $\hat{x}$ is the decoded image and x is the clean ground-truth which, as explained in Section 4.2, may differ from the input image, and λ balances the rate/distortion trade-off of the model. The combined denoising and compression autoencoder [11] is trained with batches of four noisy images from NIND [9] and one clean image from the COM:FP dataset whose ISO value does not exceed 200, with a crop size of 256 as is typically done in the learned compression literature [11, 6, 7]. The pre-trained “**universal denoiser**” used to train the JDC-UD model is the U-Net-like blind denoising model published with NIND, which was updated such that its training includes clean-clean image pairs. The CLIC¹ professional test set is used to assess the models on other clean images.

The JDC model defined in Testolina et al. (2021) [61] is trained entirely as described with Poissonian-Gaussian artificial noise [45] (with noise parameters $a = 0.2^2$, $b = 0.04^2$), the encoder from Balle et al. (2018) [7], and their proposed decoder which has twice as many layers as the one recommended by Ballé et al. An additional JDC-Cn model is trained with the larger decoder proposed by Testolina et al. in order to assess their proposed network architecture separately from the training method.

Most models are trained for six million steps with $\lambda = 4096$ to yield the highest bitrate, then the λ value is halved and training continues

¹<https://clic.compression.cc/2021/tasks/>

for three million steps for each λ value all the way down to $\lambda = 256$, like in [11]. Both the JDC-Cn.8-Tdec method that is matched with the decoder defined by Testolina et al. and the JDC-N model trained with only noisy-clean image pairs have had an additional model trained with $\lambda = 8192$ in order to reach the other methods' highest bitrate. Likewise, the whole method defined by Testolina et al. [61] was trained with up to $\lambda = 16384$.

The standard codecs comparisons are made by encoding images using GraphicsMagick 1.3.37 (JPEG), the JPEG XL encoder v0.6.1, and the BPG Image Encoder version 0.9.8. The “**standard autoencoder**” is that defined [11].

4.3.2 Results

On the importance of denoising.

The first experiment, shown on Figure 4.4, measures the impact of denoising prior to compression with different compression methods. Figure 4.5a plots the rate-distortion curves obtained without specific denoising of the input images, for a variety of codecs. We observe that for all codecs compression is an effective denoising method at the lowest bitrate. However, at reasonable bitrates, all conventional codecs (learned or not) tend to reproduce the noise. This is in contrast with our proposed joint denoising and compression paradigm, which continuously increases quality when the bitrate increases. Denoising before compression might be considered to solve the quality issue when using conventional codecs. As shown in Figures 4.5b and 4.6, this bridges (most of) the quality gap compared to our proposed JDC method, but at the cost of a significantly increased complexity (see Section 4.3.2).

Our joint denoising and compression scheme

Figure 4.5 also introduces the proposed joint denoising and compression models, JDC-CN (no noise threshold) and JDC-Cn.8 (training noise limited to $MS\text{-}SSIM \geq 0.8$). These models are trained like the dedicated denoising model in that the input training batch is made of four noisy images and one clean image, and the model is tasked with reconstructing the clean ground-truths. This single JDC model generally achieves better rate-distortion than a duo of denoising then compression neural networks (except at high bitrate), while using significantly less computational overhead.

The best results shown are obtained with the “JDC-Cn.8” model which is trained with paired images whose input noise is limited to

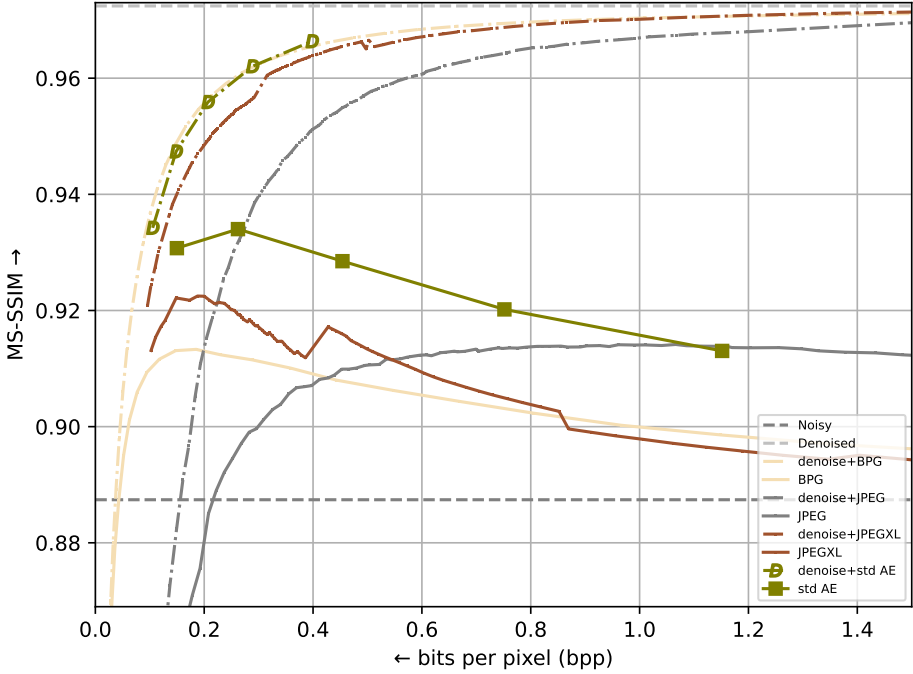
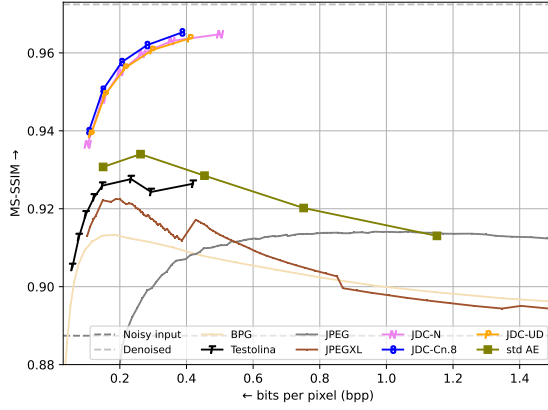
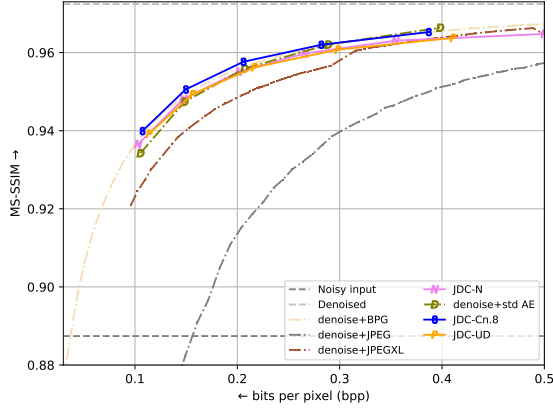


Figure 4.4: Lossy compression of noisy ($\text{MS-SSIM} \in [0.7, 1.0)$) test images from NIND [9] with standard methods (JPEG, JPEG XL, BPG, and a standard compression autoencoder [11] abbreviated as “std AE”): rate-distortion with respect to the clean ground-truth images when encoding noisy images (■), same compression schemes applied after the test images were denoised with a trained “universal denoiser” prior to compression (D). Compression alone performs some implicit denoising and the image quality is higher than that of the noisy input given sufficient bitrate, but image quality degrades as the bitrate increases and the noisy signal is eventually reconstructed. Applying a universal denoiser before the compression (dashed lines) greatly improves rate-distortion at the cost of added complexity.



(a) Compression when taking the original (and thus noisy) image as input.



(b) Applying CNN-based denoising before compression (except for our JDC which inputs the originally captured image).

Figure 4.5: Lossy compression of noisy ($\text{MS-SSIM} \in [0.7, 1.0]$) test images from NIND [9] with respect to their matching clean ground-truth. **(a)** Original (noisy) images are provided as input. Standard methods (JPEG, JPEG XL, BPG, and a standard compression autoencoder [11]) perform some implicit denoising at low bitrates, but image quality degrades as the bitrate increases since the noisy signal is reconstructed. **(b)** A universal denoiser is applied before presenting the image to the standard method. This greatly improves rate-distortion but adds of an order of magnitude of complexity. Our joint denoising and compression (JDC) autoencoder with Natural Image Noise Removal supervision allows noise-free reconstruction without prior denoising.

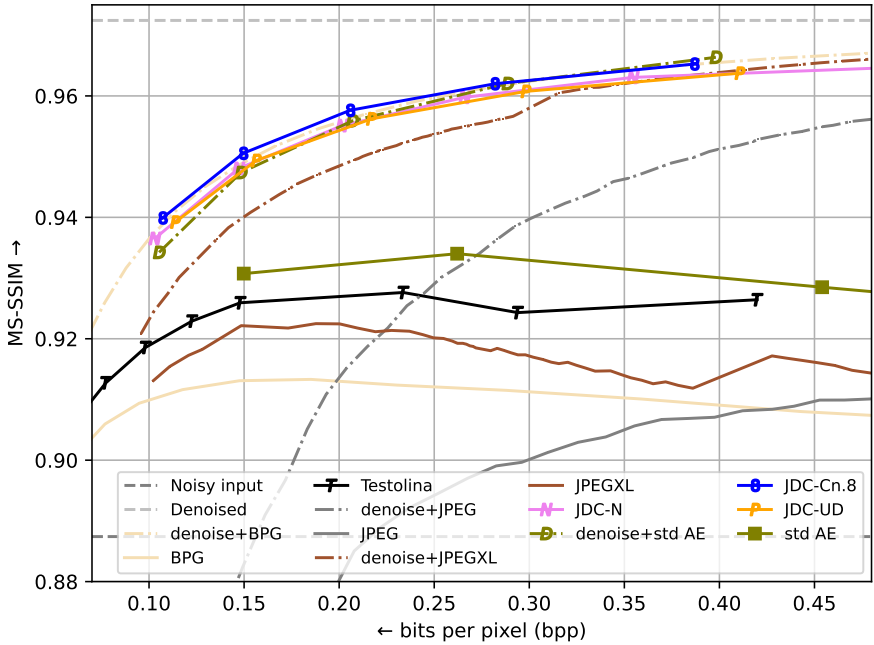


Figure 4.6: Comparison combining Figure 4.5a and Figure 4.5b for lossy compression of noisy test images ($\text{MS-SSIM} \in [0.7, 1.0)$). Note that the clean ground-truth is different from the noisy input so the non-denoising methods inherently have a lower score.

$MS\text{-}SSIM \geq 0.8$ as well as with unpaired clean images to promote generalization. T is the method trained with artificial noise described by Testolina et al. [61], which performs worse than a duo of models as is also shown in their results.

Computational complexity

Computational cost is measured in terms of GMAC and runtime on an AMD Threadripper 3960X CPU. The dedicated denoising U-Net performs 812 GMAC per MP in 65.8 s/MP, whereas the JDC model’s compression encoder performs 92.8 GMAC/MP [11] in 2.9 s/MP. The dual model approach (denoising then compression) thus performs a total of 904.8 GMAC/MP, whereas a single joint denoising and compression model operates with 10.3% of that computational complexity.

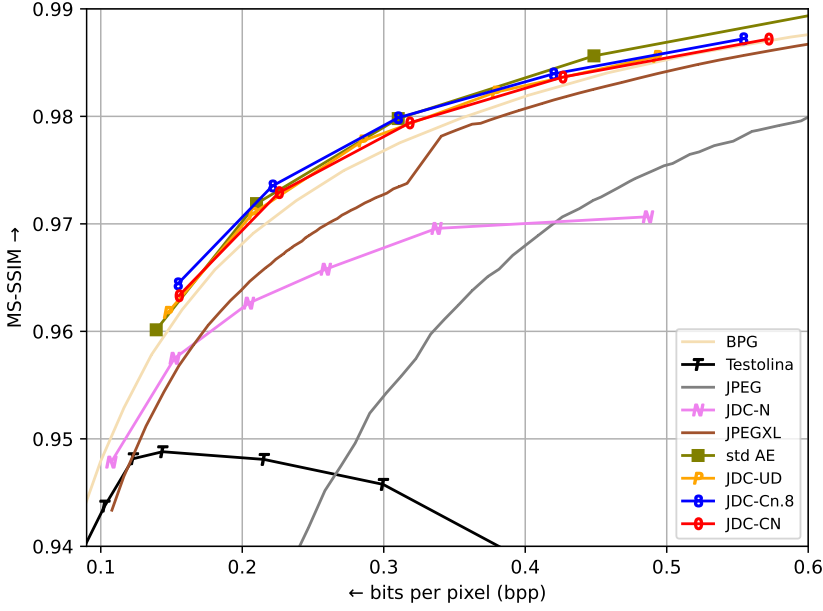
Handling clean images

JDC-C models are trained with both noisy-clean and clean-clean paired images in order to better generalize and maintain good rate-distortion on clean images. Figure 4.8a and Figure 4.7 show the behavior of different JDC training strategies when compressing clean images. JDC models trained with some clean input images or with the JDC-UD knowledge distillation technique yield a rate-distortion similar to the model trained for compression only, even when no minimum training noise threshold is set, thus incorporating clean images in the training data (JDC-CN) reinstates a good rate-distortion. Limiting the input noise to $MS\text{-}SSIM \geq 0.8$ (JDC-Cn.8) further improves rate-distortion such that it is slightly better with a JDC model than a standard model at low bitrates, and slightly worse at high bitrates due to the perception-distortion tradeoff [8] where only reconstruction fidelity matters. The JDC-N model trained with only noisy-clean image pairs performs significantly worse on clean images, and the model trained with artificial noise (“ T ”) performs worst.

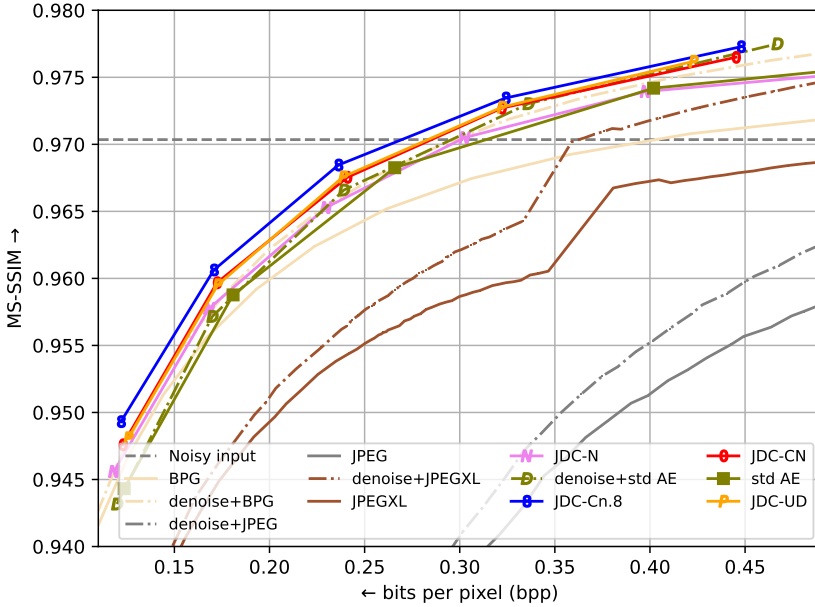
Figure 4.8b shows a common use-case where the amount of noise is low ($MS\text{-}SSIM \in [0.95, 1)$). Prior denoising still improves rate-distortion, and joint denoising and compression methods yield the most significant rate-distortion benefits. All compression methods benefit from prior or joint denoising even when the level of noise is minor. Traditional compression schemes benefit the most from prior denoising, and joint denoising outperforms prior denoising in learned methods.



Figure 4.7: Visualization of a clean test image compressed using different methods with a target bitrate of 0.23 bpp. Standard methods (JPEG, JPEG XL, BPG) tend to produce block artifacts. The method defined by Testolina et al. [61] produces oversmoothed results on clean images. Training with only noisy images can produce the same level of quality at the cost of increased bitrate. Other methods—training with only clean images, training with both clean and noisy images, and knowledge distillation from a powerful denoiser—perform well on clean images. The input image has been processed in the darktable software with (among other methods) non-local means [13] (NLM) profiled denoising.



(a) High quality (unpaired) test images



(b) Test images with $(\text{MS-SSIM} \in [0.95, 1.0], \text{MS-SSIM}_\mu = 0.97)$

Figure 4.8: MS-SSIM rate-distortion curve of different compression methods on (a) high quality images from the CLIC pro test set and (b) nearly noiseless test images from NIND [9].



Figure 4.9: Visualization of a clean (top) and noisy (bottom) test image from NIND [9] encoded with a target bitrate of 0.13 bpp using different trained compression autoencoders. From left to right: ground-truth/noisy input, standard autoencoder [11], autoencoder from Testolina et al. (2021) [61] (trained on artificial noise; increasing bitrate did not yield quality improvements on the ground-truth), joint model trained with knowledge distillation from a universal denoiser (JDC-UD), and joint model trained with both clean and low noise images (JDC-Cn.8). Images are best visualized after zooming in on a screen.

Ablation study

The effect of different training noise thresholds is analyzed when compressing noisy images. In Figure Figure 4.10a the test noise is limited to $\text{MS-SSIM} \in [0.7, 1)$ which is qualitatively fairly noisy, as shown in Figure 4.2. None of the methods perform significantly better or worse than the denoise and compress duo of models. It is worth noting that the three worst JDC training schemes are the knowledge distillation JDC-UD model, the model trained with a quality threshold of $\text{MS-SSIM} \geq 0.9$, and the decoder defined by Testolina et al. which contains twice as many layers. The JDC-Cn models trained with an MS-SSIM threshold of 0.6 and 0.8 yield the best rate-distortion. A visualization of the different denoising and compression methods at low bitrates is shown as Figure 4.9.

In Figure 4.10b, the testing noise is increased to $\text{MS-SSIM} \in [0.5, 1)$, showing how the models behave under extreme input noise. The results

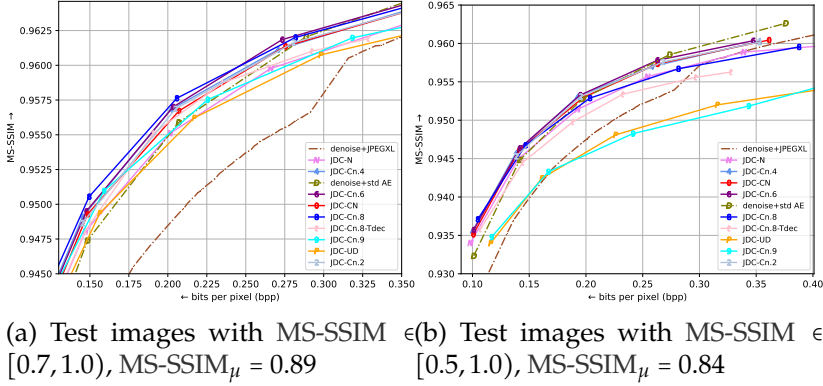


Figure 4.10: Lossy compression of noisy test images from NIND [9] with different joint denoising and compression methods including JDC-Cn trained with different MS-SSIM thresholds, JDC-CN trained with no such threshold, JDC-N trained with no clean images, JDC-UD trained with knowledge distillation from a universal denoiser, and JDC-Cn.8 trained with the larger decoder defined by Testolina et al. [61]. **(a)** The JDC-Cn methods tend to perform well, especially when the quality threshold is set between 0.6 and 0.8, but the rate-distortion worsens when the training quality threshold increases to 0.9 (i.e. little noise is seen by the model during training). The JDC-UD model yields similarly lower performance, and so does the larger decoder despite using the same training scheme as JDC-Cn.8. **(b)** The noise level is more extreme than what is likely to occur in photographs. This further shows that compressing with more noise than is ever seen during training (e.g. JDC-Cn.9) yields worse rate-distortion. The model trained without clean image pairs (JDC-N) does not perform as well even when the test images are noisier.

are largely the same; the model trained with $\text{MS-SSIM} \geq 0.9$ struggles even more due to the increased noise and its performance is close to that of the JDC-UD method, the model trained with $\text{MS-SSIM} \geq 0.8$ does not perform as well whereas the model trained with $\text{MS-SSIM} \geq 0.6$ is still competitive, and it remains beneficial to train with clean image pairs as well.

4.4 Conclusion

Denoising images improves rate-distortion whenever there is noise present, regardless of the compression method. Denoising can be per-

formed prior to compression using a dedicated denoiser (as is typically done in professional image development workflow) with no adaptation to the compression scheme. A joint model that is trained to perform denoising and compression simultaneously yields further improvements in rate-distortion. As a result, a joint denoising and compression model performs 8.9 times fewer GMAC operations than a U-Net denoiser followed by a compression encoder. Since the JDC model only differs from standard learned compression models by the adopted supervision strategy, it can be implemented using any of the compression architectures available in the literature (such as [6, 7, 11]). Our proposed Natural Image Noise Removal supervision strategy thus provides a fundamental and generic contribution that is expected to become popular in future works related to learned compression.

In practice, joint denoising and compression models may be trained using a dataset of noisy-clean image pairs with natural noise, such as NIND [9] and SIDD [2]. Performance is improved by setting a quality threshold on the training images, such as $\text{MS-SSIM} \geq 0.8$ or $\text{MS-SSIM} \geq 0.6$ depending on the maximum expected noise. The rate-distortion curve is preserved on clean images and improved in any case by incorporating clean-clean image pairs in the training data.

An alternative method consists of performing knowledge distillation [24] by using the output of a dedicated denoiser as ground-truth images during training. This has the benefit of allowing a virtually limitless training dataset because a paired dataset is no longer required, but it requires pre-processing of the training images and results in a slightly worse rate-distortion.

5

Learning joint denoising, debayering and compression from the Raw Natural Image Noise Dataset

IMAGE denoising and compression are fundamental tasks in digital photography and image development, but current methods applied to developed images suffer from limitations in performance and generalization. These limitations arise because traditional compression methods and most of the literature in image denoising focus on developed images, which have undergone complex and often proprietary transformations applied by a camera ISP or an image development software. This chapter presents a methodology that leverages raw sensor data and the Raw Natural Image Noise Dataset (RawNIND), a contribution of this work, to achieve significant improvements in computational efficiency (four to sixteen times fewer operations), produce high-quality images from noisy raw data, and effectively generalize across diverse camera sensors and development workflows.

In contrast, raw sensor data offers a more consistent and flexible representation of the captured scene. Unlike developed images, raw data preserves the linear sensor output and avoids artifacts introduced during image development, making it more suitable for denoising and compression tasks. However, raw image manipulation introduces its own challenges, as each sensor has unique characteristics, such as noise

patterns and native color spaces. These variations have historically led to the assumption that models trained on one camera’s raw data would not generalize to others.

To address these challenges, we introduce the *Raw Natural Image Noise Dataset (RawNIND)*, a collection of paired raw images from a variety of camera sensors. RawNIND enables the development of models that generalize across sensors, workflows, and noise conditions. Using this dataset, we explore two complementary denoising methods:

- **Raw Bayer Image Denoising:** This approach offers significant computational efficiency (reducing processing by a factor of four) by denoising raw Bayer data directly, operating on one-quarter of the spatial dimensions and deferring demosaicing until the final step.
- **Linear RGB Denoising:** This method operates on debayered images in a standardized linear RGB color space, maintaining high image quality and facilitating better generalization across different color filter arrays (CFAs) while maintaining the advantages of raw data.

Both of the proposed denoising methods outperform approaches that take developed images as input, and they integrate seamlessly into any image development workflow, providing adaptability for any real-world application.

Building on these denoising advancements, we then integrate denoising and compression into a unified framework that operates directly on raw sensor data. This joint approach significantly enhances both computational efficiency at the encoder stage and rate-distortion performance by addressing noise at the earliest stage of the image pipeline. Unlike conventional workflows which compress fully developed images, our method compresses raw data, preserving flexibility for subsequent editing. It also enables new compression paradigms where raw images are stored alongside lightweight XMP sidecar files, which contain all the necessary development instructions to produce the final image.

This chapter makes the following key contributions:

- **RawNIND Dataset:** A novel dataset of paired raw images enabling generalized denoising and compression across diverse sensors and noise conditions.
- **Denoising Methods:** Two high-performing approaches for raw image denoising—considering Bayer denoising for computational

efficiency and linear RGB representations for different CFAs—that generalize across all image development workflows.

- **Integrated Denoising and Compression:** A joint model that significantly improves rate-distortion performance and computational efficiency compared to traditional sequential pipelines. **Compression Paradigm:** We propose a paradigm for storing compressed raw images with metadata in the form of XMP sidecar files. These files store the development instructions, enabling non-destructive editing and efficient storage.

The rest of the chapter is organized as follows: Section 5.1 reviews related work on image denoising and compression. Section 5.2 describes our proposed methods and the creation of RawNIND. Section 5.3 details the experimental setup and evaluation metrics, while Section 5.4 presents the results and discusses key findings. Finally, Section 5.5 concludes the chapter and outlines directions for future research.

5.1 Related work

5.1.1 Image denoising

Image denoising aims to suppress noise while preserving important image details. Traditional methods, such as non-local means [13] (NLM), wavelet-based denoising [19], and BM3D [18], rely on mathematical models to exploit redundancies in image structure. Among these, BM3D remains a widely used benchmark due to its high-quality results under controlled conditions, despite its computational intensity, while NLM and wavelet-based methods are implemented in raw image development software such as Darktable, due to their simplicity and efficiency.

The emergence of deep learning techniques has shifted the focus to data-driven approaches. Convolutional neural networks [33] (CNNs), including architectures like U-Net [54] and methods such as DnCNN [65], and CBDNet [22], have shown significant improvements over traditional methods. However, these models often rely on synthetic noise for training and are sensitive to mismatches between training and testing noise distributions. While datasets like SID [14], SIDD [2], and the Natural Image Noise Dataset [9] (NIND) provide real noisy-clean image pairs for training, their focus on specific sensors and development workflows limits their applicability to diverse real-world scenarios.

Critically, most existing work evaluates denoising on developed images, which does not reflect real-world use cases. In practice, denoising

is almost always performed by the camera’s ISP or raw development software, either directly on raw data or before applying non-linear transformations like sharpening. Post-development denoising is rare and generally suboptimal, as the introduction of non-linearities limits both denoising performance and its integration into the imaging pipeline. These limitations underscore the importance of addressing noise earlier in the processing workflow, such as directly on raw data.

Methods combining denoising and demosaicing [25, 20, 38] have demonstrated improved noise suppression by leveraging raw data’s linear characteristics with respect to illumination. However, these studies often assume a single camera model, leaving cross-sensor generalization largely unexplored. This work addresses this gap by explicitly evaluating generalization across multiple sensors and workflows using raw image data.

5.1.2 Raw image development pipelines

Raw image development converts sensor data into visually interpretable images through steps such as demosaicing, color space transformations, and tone mapping. The variability in raw development pipelines arises from the unique characteristics of camera sensors and user-defined adjustments during development. In-camera ISPs apply proprietary pipelines optimized for speed and versatility, while raw development software like Darktable (illustrated in Figures 2.2 and 5.1) or Adobe Lightroom offer extensive customization options. These variations pose challenges for denoising models which are often trained on images from a specific raw sensor or development pipeline.

To maintain consistency across sensors, color space transformations are applied to convert camera-specific color spaces (“CamRGB”) to a common working color space such as linear Rec. 2020 [26]. In this study, we hypothesize that models trained on Bayer or linear RGB data may achieve higher performance by avoiding the non-linearities introduced during image development, and we evaluate this assumption experimentally. Previous studies have not explicitly addressed cross-sensor variability, which we also aim to investigate and address through our proposed methods and dataset.

5.1.3 Learned image compression

Image compression reduces storage and transmission costs while maintaining perceptual quality. Traditional methods like JPEG [63], JPEG2000 [57], and JPEG XL [29] rely on transform and entropy coding to exploit



Figure 5.1: An image development pipeline illustrated using Darktable 5.0 and a moderately noisy image (taken at ISO16000). **Throughout this work, we refer to stage 1 as the Bayer input, stage 2 as the Linear RGB input, and stage 8 as the developed image** (although NLM denoising is shown in step 3 for illustrative purposes, prior denoising is not applied in the development pipeline for the developed images used as input to the models in this work, as shown in Figure 5.5.) Darktable was chosen as our primary tool for raw image development due to its wide use within the photography community and its open-source nature, ensuring transparency and reproducible experiments.

redundancies in image data. In these methods, noise increases entropy and reduces compression efficiency.

Learned image compression, pioneered by Ballé et al. [6], builds upon the autoencoder architecture [32], using convolutional autoencoders and learned entropy models to optimize compression end-to-end. Autoencoders, initially proposed for dimensionality reduction, form the backbone of learned compression by encoding high-dimensional data into compact latent representations. Subsequent works have improved performance with hyperprior [7] (HP) models, autoregressive priors [48], and transformers [39, 3]. However, these advancements often come with increased computational complexity.

Joint denoising and compression (JDC) integrates both tasks into a single framework, as proposed in [12]. By addressing noise during compression, JDC allocates bits more efficiently to meaningful image content, and improves rate-distortion performance and computational efficiency compared to sequential pipelines. This work extends these principles to raw data, demonstrating the benefits of integrating denoising, demosaicing, and compression into a unified pipeline.

5.1.4 Generalization across sensors

Generalizing denoising models across sensors remains a significant challenge due to variations in sensor characteristics, noise patterns, and raw image development pipelines. While datasets like SIDD and NIND have shown that models can generalize across certain cameras when trained on developed images, this is often contingent on consistent development workflows. Dealing with raw data avoids this constraint but introduces inherent variabilities such as sensor-specific noise and color profiles, which require careful handling to ensure robust generalization.

This work explicitly addresses these challenges by introducing a dataset that spans multiple sensors and by developing models capable of generalizing across raw development workflows. We demonstrate the effectiveness of these models in handling diverse sensor characteristics, enabling robust performance across a variety of cameras and workflows. Furthermore, our approach preserves the flexibility inherent to raw data, ensuring compatibility with a wide range of post-processing options.

5.2 Proposed approach

This section describes our approach to image denoising and compression, focusing on methods that operate directly on raw Bayer and linear RGB data. We introduce the *Raw Natural Image Noise Dataset (RawNIND)*,

designed to facilitate research and development on generalized denoising and compression across multiple sensors and workflows. We then detail the proposed methodologies for denoising and compression of raw Bayer data and linear RGB images, highlighting their advantages over traditional approaches reliant on developed images.

5.2.1 RawNIND Dataset

Dataset overview

RawNIND comprises paired noisy and clean images captured under diverse conditions using multiple camera sensors. The noisy images were captured at higher ISO settings and with shorter exposure times compared to their clean counterparts, which were acquired at base ISO and optimal exposure settings to minimize noise. Each noisy image is paired with one or more corresponding clean image(s) of the same scene, ensuring that both images capture the same content and lighting conditions are consistent. This pairing is achieved by capturing the sequences of noisy and clean images on a sturdy tripod and in rapid succession, minimizing changes in the scene. A preview of the dataset's ground-truth images is shown in Figure 5.2. By retaining the raw sensor data, the dataset preserves linear characteristics and avoids artifacts introduced during image development. To promote generalization across sensors, RawNIND includes data from a variety of Bayer pattern cameras, ensuring compatibility with consumer and professional imaging devices. Tables 5.1 and 5.2 summarizes the dataset composition, while Figure 5.3 illustrates the distribution of noise levels. The dataset is published under <https://dataverse.uclouvain.be/dataset.xhtml?persistentId=doi:10.14428/DVN/DEQCIM>

Cross-Sensor Model Consistency

Raw images are captured in sensor-specific color spaces ("CamRGB"), which vary across cameras. For models working with linear RGB images as input, we pre-process the raw data by demosaicing and converting CamRGB to the standardized linear Rec. 2020 color space. Models working with raw Bayer data receive 4-channel Bayer images (CamRGB) as input, preserving the native sensor format. For these models, Bayer images are first cropped to a consistent RGGGB pattern using the method illustrated in Figure 5.4, ensuring compatibility across cameras with varying Bayer arrangements. The CamRGB-to-Rec. 2020 transformation is applied solely to the model's 3-channel CamRGB output (upsampled using PixelShuffle) during training before computing the loss function.

Table 5.1: RawNIND composition by camera model. The “unknown sensor” set is made of the Canon EOS M100 scenes.

Camera	# Scenes	# Clean	# Noisy	# Images
Sony ILCE-7C	148	361	1423	1784
Canon EOS 500D	13	13	71	84
Canon EOS 6D	8	8	80	88
Canon EOS 7D	7	7	49	56
Canon EOS D60	3	3	12	15
Nikon Z 6	3	3	41	44
Nikon D40	2	8	8	16
Lumix DMC-GH1	1	1	5	6
Canon EOS M100	11	20	50	70
Fujifilm X-T1 (X-Trans)	104	128	486	614
Fujifilm X-T2 (X-Trans)	10	10	54	54
Total	310	562	2279	2831

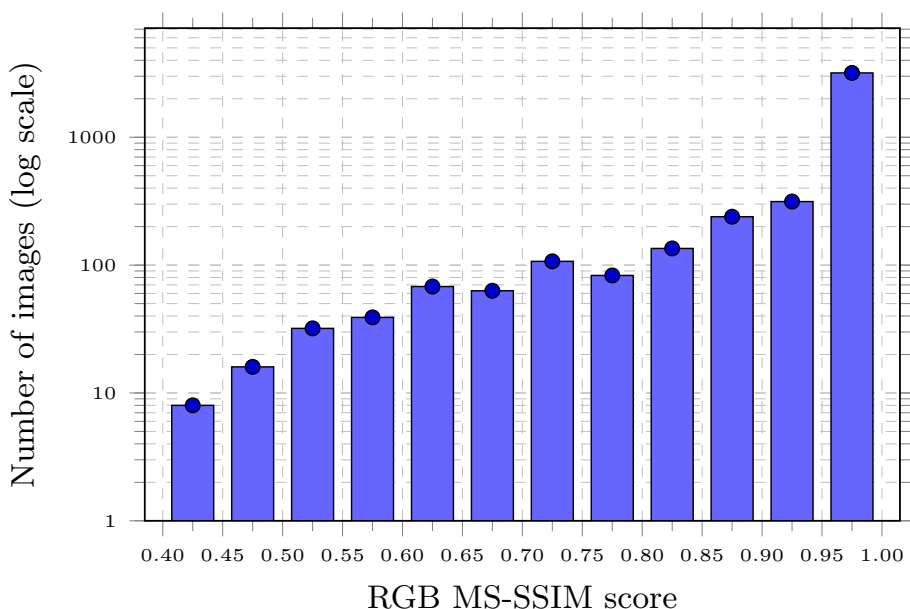


Figure 5.3: RawNIND composition by noisiness: histogram of MS-SSIM between ground-truth and noisy images.

Table 5.2: Camera sensors specifications

Camera	Sensor Type	Sensor Size (mm)	Resolution	CFA
Sony ILCE-7C	CMOS (Exmor R)	35.6 x 23.8	6000 x 4000	Bayer
Canon EOS 500D	CMOS	22.3 x 14.9	4752 x 3168	Bayer
Canon EOS 6D	CMOS	35.8 x 23.9	5472 x 3648	Bayer
Canon EOS 7D	CMOS	22.3 x 14.9	5184 x 3456	Bayer
Canon EOS D60	CMOS	22.7 x 15.1	3072 x 2048	Bayer
Nikon Z 6	BSI-CMOS	35.9 x 23.9	6048 x 4024	Bayer
Nikon D40	CCD	23.7 x 15.6	3008 x 2000	Bayer
Lumix DMC-GH1	Live MOS	17.3 x 13.0	4000 x 3000	Bayer
Canon EOS M100	CMOS	22.3 x 14.9	6000 x 4000	Bayer
Fujifilm X-T1	X-Trans CMOS II	23.6 x 15.6	4896 x 3264	X-Trans
Fujifilm X-T2	X-Trans CMOS III	23.6 x 15.6	6000 x 4000	X-Trans

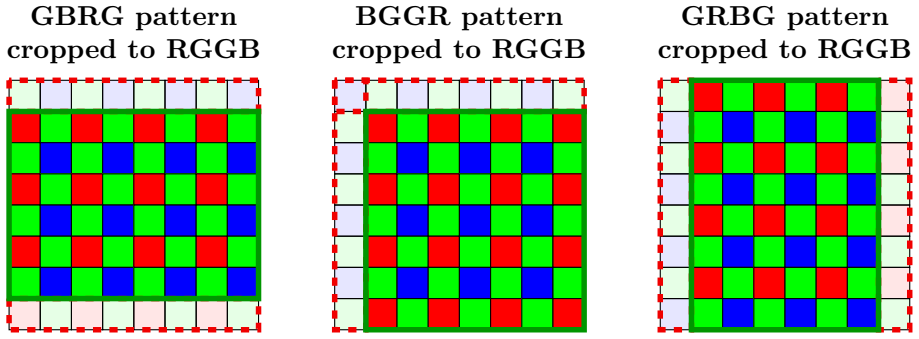


Figure 5.4: Method for converting various Bayer color filter array patterns (GBRG, BGGR, GRBG) to a consistent RGB pattern by cropping up to two rows or columns (shown with red dashes). This standardization is crucial for training models that generalize across different camera sensors.

$$\mathbf{M}_{\text{XYZ} \rightarrow \text{Rec2020}} = \begin{bmatrix} 1.7167 & -0.3557 & -0.2534 \\ -0.6667 & 1.6165 & 0.0158 \\ 0.0176 & -0.0428 & 0.9422 \end{bmatrix} \quad (5.1a)$$

$$\mathbf{M}_{\text{Rec2020} \rightarrow \text{CamRGB}} = \mathbf{M}_{\text{XYZ} \rightarrow \text{Rec2020}} \cdot \mathbf{M}_{\text{XYZ} \rightarrow \text{CamRGB}}^{-1} \quad (5.1b)$$

$$\text{Image}_{\text{Rec2020}} = \text{Image}_{\text{CamRGB}} \cdot \mathbf{M}_{\text{Rec2020} \rightarrow \text{CamRGB}} \quad (5.1c)$$

5.2.2 Proposed denoising methods

We propose two denoising methods tailored to specific representations: raw Bayer images and linear RGB images.

Raw Bayer image denoising

This method denoises raw Bayer data directly, deferring demosaicing to the final stage. Denoising is performed on the 4-channel Bayer representation, preserving the native sensor format and significantly reducing computational overhead by operating on one-quarter the spatial dimensions of fully debayered images. The model outputs 3-channel CamRGB data, upsampled to full resolution using PixelShuffle. During training, this output is transformed to linear Rec. 2020 for computing the loss, ensuring compatibility with standard profiles.

Linear RGB denoising

Linear RGB denoising begins with demosaicing raw images using an edge-aware interpolation method [35], or the Markesteijn algorithm for X-Trans images, and converting them to the linear Rec. 2020 color space. This approach avoids non-linearities introduced by different image development modules while maintaining the consistency of a standardized color space, facilitating cross-sensor generalization.

5.2.3 Joint denoising and compression (with implicit demosaicing for Bayer input)

To extend the benefits of raw data processing, we integrate denoising and compression into a single framework. This joint approach is investigated with both raw Bayer and linear RGB image inputs, mirroring the methodology applied in the denoising methods. For raw Bayer input, this inherently involves demosaicing as part of the process, while for linear RGB input, demosaicing is assumed to have already occurred. Our framework utilizes a convolutional autoencoder with a learned entropy model [11, 12] and is trained by minimizing a rate-distortion loss, typically expressed as $\mathcal{L} = D(x, \hat{x}) + \lambda \text{bpp}(\hat{x})$, where D is a distortion measure between the clean ground-truth image x and the reconstructed image $\hat{x}$ (after joint denoising and compression of the input image y), $\text{bpp}(\hat{x})$ is the bitrate after entropy-coding, and λ is a trade-off parameter. The distortion measure D is typically MSE or MS-SSIM, with our work focusing on MS-SSIM.

For raw Bayer input, the framework integrates joint denoising, demosaicing and compression (JDDC). By addressing noise at the earliest stage, this joint approach reduces computational overhead and improves rate-distortion performance. Compressing raw Bayer data directly avoids encoding redundant interpolated information, further enhancing compression efficiency. PixelShuffle is used at the end of the decoder stage to reconstruct full-resolution RGB images from Bayer inputs.

For linear RGB input, the framework performs joint denoising and compression (JDC) on a demosaiced linear Rec. 2020 input. This approach still benefits from addressing noise early in the pipeline, improving rate-distortion performance compared to compressing noisy linear RGB images or JDC of developed images. Furthermore, by operating on linear RGB data after demosaicing, the framework is agnostic to the specific CFA pattern of the input sensor, such as Bayer or X-Trans, enhancing its versatility.

5.2.4 Advantages of the proposed approach

The proposed methods offer the following advantages:

- **Computational Efficiency:** Raw Bayer denoising and compression reduces computational requirements by operating on one-quarter the spatial dimensions of debayered images.
- **Generalization Across Sensors:** Both Bayer and linear RGB methods are designed to handle diverse sensor characteristics and raw workflows, ensuring robust performance across devices.
- **Avoidance of Non-Linearities:** By operating on raw or linear RGB data, our methods avoid artifacts introduced by development modules such as tone mapping, gamma correction, or sharpening. This preserves the intrinsic characteristics of the captured scene and prevents noise amplification, leading to more robust denoising performance.
- **Development Flexibility:** Outputs from Bayer models (3-channel CamRGB) and linear RGB models (Rec. 2020) integrate seamlessly into raw development software like Darktable, enabling users to work on denoised images as if they were raw.
- **Improved Compression Efficiency:** The joint framework minimizes redundancy, achieving superior rate-distortion performance compared to traditional methods.

5.3 Validation methodology

This section presents the experimental setup, including dataset preparation, baseline models, and evaluation methodologies, to validate the proposed denoising and compression approaches.

5.3.1 Dataset preparation

Dataset preprocessing

The dataset, which is presented in Section 5.2.1 and downloadable on <https://dataverse.uclouvain.be/dataset.xhtml?persistentId=doi:10.14428/DVN/DEQCIM>, is stored in two formats: raw Bayer images, processed only to remove empty borders, subtract the black level, and normalize pixel values to the sensor’s white level, and debayered images converted to the linear Rec. 2020 color space. Both formats are divided into overlapping patches of 1024×1024 pixels (512×512 for Bayer) with a stride of 256 pixels (128×128 for Bayer). These patches facilitate efficient loading and flexible filtering during training. Metadata is included with each image pair, including alignment parameters, gain normalization values, alignment loss, and MS-SSIM scores, to allow flexible dataset filtering based on quality thresholds.

Alignment of noisy and clean images is critical for supervised training. We find the optimal alignment by searching for the translation vector (horizontal and vertical shift) that minimizes the $L1$ loss between gain-normalized images in the RGB domain (demosaiced images). Aligning in the Bayer domain is impractical due to the single color channel per pixel. The search for the best shift is performed iteratively. Starting from an initial alignment, we explore a small, 3×3 pixel neighborhood of potential shifts around the current best alignment. If a shift within this neighborhood reduces the $L1$ loss, it becomes the new best, and the search continues from that point. This local search is constrained by a maximum search range of ± 128 pixels in each direction. The iterative process stops when no better alignment is found within the 3×3 neighborhood or the maximum shift is reached. The resulting optimal shift and its corresponding loss are recorded. Image pairs with an alignment loss exceeding 0.035 are discarded (2.4% of the dataset).

Despite careful dataset preparation, issues such as lighting variations, foreign objects or transient elements like insects can occasionally disrupt the image pairs. To further address inconsistencies, binary loss masks are generated to exclude problematic regions from loss computation during training. Masks are applied to pixels with excessive $L1$ loss (> 0.4

or exceeding the 99.99th percentile of the loss distribution). Overexposed pixels (≥ 0.99 in the ground-truth image) are similarly masked. The mask is refined using a binary opening operation to remove small, isolated regions, resulting in smoother and more meaningful masking. Crops with more than 50% of their area masked are excluded.

Clean data for generalization

To enhance sensor diversity and improve model generalization, we augmented the dataset with clean (unpaired) raw images from various sources. This augmentation included 561 Bayer images from numerous cameras available on the `raw.pixls.us` platform, with a maximum ISO limit of 200 to ensure high image quality. Additionally, we incorporated 11,815 Bayer images captured with several cameras from our personal collection, also restricted to ISO 200 or lower. These unpaired clean images aim to expose the model to a broader range of sensor characteristics, facilitating improved generalization to unseen sensors. By including these diverse clean images, we evaluate their contribution to training models capable of handling raw data from previously unseen camera sensors.

Loading train data

During training, pairs of aligned image patches are loaded. For RGB-Bayer image pairs, the Bayer shift is halved using integer division. If the shift is odd, an additional row or column is trimmed from both the RGB and Bayer images to ensure proper alignment. To account for differences in brightness or intensity between the clean and noisy images, their average pixel intensities (gains) are matched. Random 256×256 (128×128 for Bayer) crops are then extracted from the aligned patches.

For models working with developed images, a custom proxy development pipeline is applied to simulate typical image development steps. This pipeline includes logarithmic tone mapping on the luminance channel, edge enhancement using a Laplacian filter, gamma correction, contrast enhancement with a sigmoid function, and image sharpening via Gaussian blur and weighted blending. Parameters and activations for these operations are randomized within plausible ranges, ensuring variability and robustness across a wide range of image characteristics. While this simulated pipeline does not replicate all variations of real-world image development, it provides consistent training data for comparison across methods. The use of this pipeline enables direct

evaluation of models trained on raw Bayer, linear RGB, and developed images representations.

5.3.2 Test sets and evaluation conditions

RawNIND test set

The test set comprises 10 scenes from RawNIND, captured using six different cameras to ensure diverse evaluation conditions. These scenes include 7D-2 (Canon EOS 7D), Vaxt-i-trad (Canon EOS 6D), Pen-pile (Panasonic DMC-GH1), MuseeL-vases-A7C, TitusToys, boardgames_top, Laura_Lemons_platformer, and MuseeL-bluebirds-A7C (all captured with the Sony A7C), as well as D60-1 (Canon EOS D60) and MuseeL-Saint-Pierre-C500D (Canon EOS 500D). This diverse selection of scenes and camera models ensures representation of varying sensor characteristics and noise profiles.

To evaluate generalization across entirely unseen sensors, an additional test set includes six scenes captured with the Canon EOS M100 that has not been seen at training, and referred to as the “unknown sensor” test set. These scenes are denoted LucieB_bw_drawing1, LucieB_bw_drawing2, LucieB_board, LucieB_painted_wallpaper, LucieB_painted_plants, and LucieB_groceries.

Evaluation pipeline

Raw model outputs are not evaluated directly due to their lack of immediate visual usability. Instead, outputs are developed using Darktable with manually curated settings. The same development instructions are applied to raw ground-truth images and model outputs, enabling consistent comparisons using the MS-SSIM metric. For models trained to take developed images as input, the same development steps are applied directly to their input images. This evaluation method may slightly favor models processing developed images, as their outputs are assessed directly. When compressing clean images, these models only need to reconstruct the input, whereas the outputs of models handling raw or linear RGB data undergo development where imperfections can be amplified differently. Despite this, applying the same development process to all outputs reflects real-world workflows and allows for the fairest possible comparison.

The sidecar XMP files containing the manually curated settings for test images are published alongside the dataset, allowing reproducibility of results and enabling further analysis or experimentation by the community.

5.3.3 Baseline models for denoising

We evaluate denoising models trained on raw Bayer images, linear RGB images, and developed images (“sRGB”). The base model for developed and linear RGB images is a variant of the U-Net architecture, with the number of channels halved, reducing computational complexity by a factor of four. ReLU activations are replaced with Leaky ReLU (with a negative slope of 0.2).

For Bayer images, which operate on one-quarter of the pixels, a PixelShuffle upscaling layer [56] is added at the end of the network, resulting in an additional fourfold reduction in complexity. Unless otherwise specified, all U-Net models in this chapter refer to this simplified variant. Additional variants in the ablation study include models with:

- **Extra Pairs:** Incorporating additional noisy-clean pairs from the unknown sensor.
- **No Clean Data:** Excluding unpaired clean images during training (i.e. denoising-only).
- **Pre-Upsampled:** Bilinear upsampling of Bayer data before input.
- **Gamma Correction:** Applying the gamma correction output $\leftarrow \text{output}^{1/2.2}$ for output > 0 before loss computation.
- **More channels:** Expanding channels by $1.5\times$ over the base model.

BM3D is used as a traditional baseline for linear RGB and fully developed inputs.

5.3.4 Joint denoising and compression experiments

We evaluate JDC models trained on Bayer, linear RGB, and developed images. The base model for fully developed and linear RGB images is the compression autoencoder architecture defined in [11] and used in for standard JDC in [12], while the Bayer models get one additional convolution and a PixelShuffle layer. Variants include pre-upsampling Bayer inputs and excluding clean data. Compression-only models and sequential pipelines (denoise first using the base U-Net from Section 5.3.3, then compress) are also evaluated. The computational complexity of compression encoders as well as that of denoising models is described in Table 5.3. The complexity of the compression decoder is not significantly reduced because, despite working with reduced Bayer dimensions, the last layer involves a final convolution which takes place in

Table 5.3: Model complexity for denoising and compression

Model	Input Image Type	GMAC/MP
Denoising		
U-Net ($\frac{1}{2}$ # of channels) (1)	Bayer	55
U-Net ($\frac{1}{2}$ # of channels) (2)	RGB (linear or developed)	220
U-Net (all channels) (3) [54]	RGB (linear or developed [9])	875
Compression encoder		
JDDC	Bayer	22
JDDC (pre-upsampled)	Bayer	87
JDC	RGB (linear or developed [12])	86
Denoise (1) then compress	Bayer	141
Denoise (2) then compress	RGB (linear or developed)	305
Denoise (3) then compress	RGB (linear or developed)	1095

high-resolution; further experiments are needed to reduce the complexity of the compression decoder. For comparison, JPEG XL is tested on linear RGB and developed images, with rate-distortion curves used to assess performance across noise levels.

Results are evaluated using developed outputs to ensure fair comparisons across all formats. The MS-SSIM and bpp metrics capture image quality and compression efficiency. RD curves are generated for the RawNIND test set, highlighting the impact of noise, (non-linear) image development, and the advantages of integrating denoising into the compression pipeline.

5.4 Results and discussion

5.4.1 Denoising performance

The proposed denoising methods were evaluated using raw Bayer, linear RGB, and developed images representations. Tables 5.4 and 5.5 present the MS-SSIM indices across the RawNIND test set and an additional “unknown sensor” test set. Figures 5.5 and 5.6 show quantitative and qualitative results on two test images.

Advantages of raw and linear RGB data: Models denoising raw Bayer and linear RGB images consistently outperformed those trained on developed (“sRGB”) images in MS-SSIM. This underscores the advantage of working with raw or linear RGB data, as it retains more linear characteristics and avoids the artifacts introduced in image development.

Bayer vs. linear RGB: The performance of Bayer and linear RGB

Table 5.4: Average MS-SSIM scores of denoised images from the RawNIND main and unknown sensor test sets, after manual development with Darktable. Columns correspond to subsets with varying noise levels (average MS-SSIM of developed input images), with the highlighted column representing the entire main test set. Bold values are within 99.5% of the best score in each column. All models use the U-Net architecture, except BM3D.

Denoising method (input image)	RawNIND test set						Unk. sensor
	0.468	0.598	0.680	0.763	0.846	0.933	
U-Net (Bayer, ours)	0.930	0.941	0.949	0.956	0.962	0.972	0.873
U-Net (linear RGB, ours)	0.930	0.942	0.949	0.957	0.963	0.973	0.874
U-Net (developed images, [12])	0.862	0.891	0.910	0.928	0.945	0.967	0.836
BM3D [18] (linear RGB)	0.838	0.873	0.893	0.906	0.921	0.940	-
BM3D [18] (developed images)	0.826	0.870	0.895	0.919	0.944	0.968	0.823

Table 5.5: Ablation study exploring different configurations of the Bayer and Linear RGB models described in Table 5.4. The models configurations used in this table are described in Section 5.3.3.

Input image (train. method)	RawNIND test set						Unk. sensor
	0.468	0.598	0.680	0.763	0.846	0.933	
Bayer (extra pairs)	0.925	0.938	0.946	0.954	0.961	0.972	0.876
Bayer (more channels)	0.929	0.940	0.947	0.954	0.960	0.969	-
Bayer (no clean data)	0.926	0.938	0.946	0.954	0.961	0.971	0.871
Bayer (pre-upsampled)	0.931	0.943	0.950	0.958	0.964	0.973	0.877
Bayer (with gamma)	0.910	0.928	0.939	0.950	0.961	0.973	0.871
Linear RGB (extra pairs)	0.929	0.940	0.947	0.955	0.960	0.970	0.873
Linear RGB (no clean data)	0.929	0.940	0.947	0.955	0.960	0.970	0.870
Linear RGB (with gamma)	0.931	0.942	0.950	0.958	0.963	0.973	0.875

models was comparable, with Bayer models achieving a significant edge in computational efficiency by processing four-channel Bayer data at reduced spatial dimensions. This approach reduces the computational complexity by a factor of 4, as Bayer models require only a quarter of the computations needed for fully debayered linear RGB images. The use of PixelShuffle to upscale Bayer outputs ensures high-quality results without adding significant computational overhead.

Generalization to unknown sensors: The models demonstrated strong cross-sensor performance. Bayer models trained with “extra pairs” from the unknown sensor achieved an average MS-SSIM of 0.876 on the unknown sensor test set, while just adding clean images from different sensors resulted in an MS-SSIM of 0.873 and training without clean data yield a score of 0.871. This marginal difference highlights that incorporating clean data aids the Bayer model’s generalization, but

5: Learning joint denoising, debayering and compression from the Raw Natural Image Noise Dataset

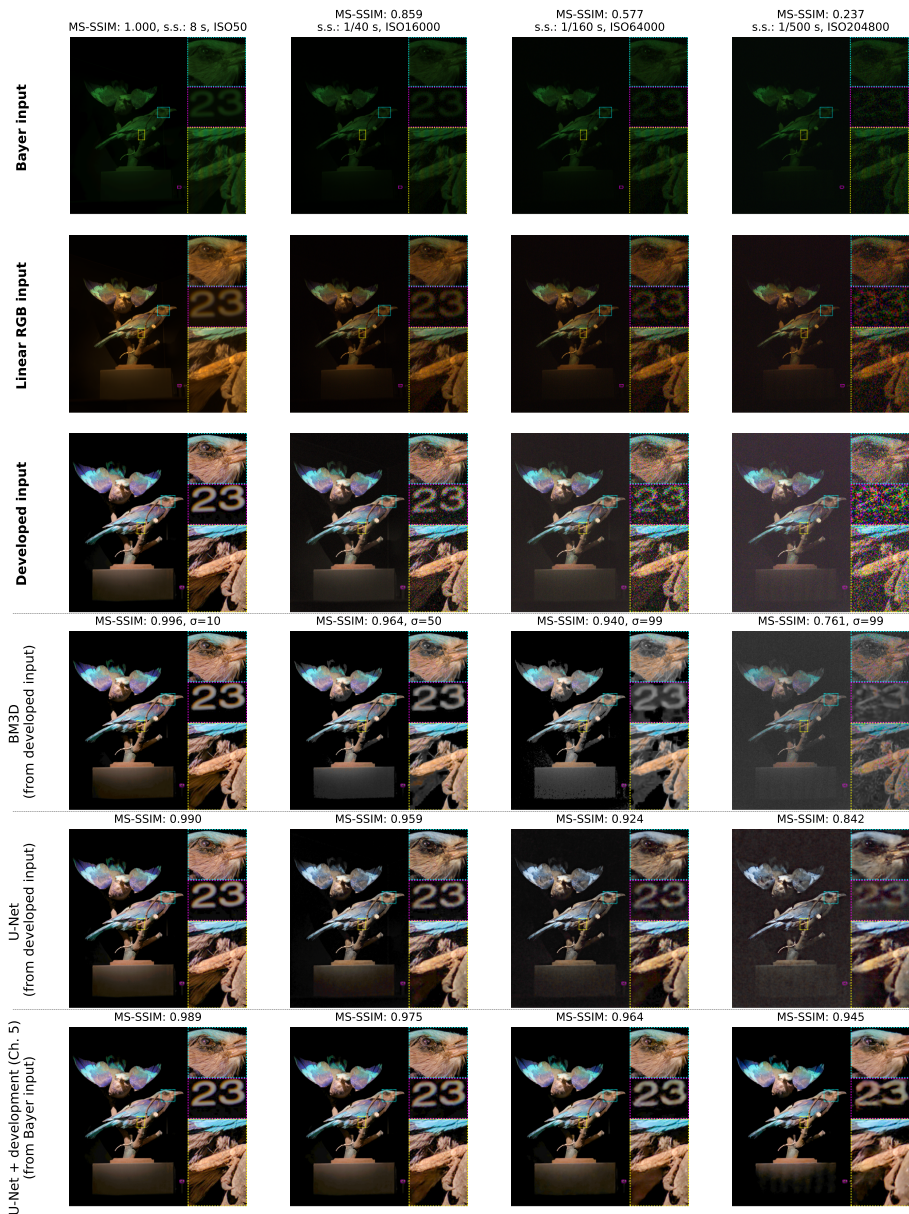


Figure 5.5: Visual and MS-SSIM results for denoising a test image from the RawNIND main test set (MuseeL-bluebirds-A7C) under varying input noise levels. Results are shown for different denoising methods and input formats. Development is performed using darktable-cli and the same set of development instructions (XMP file) whose steps are illustrated in Figure 5.1.

5.4. Results and discussion

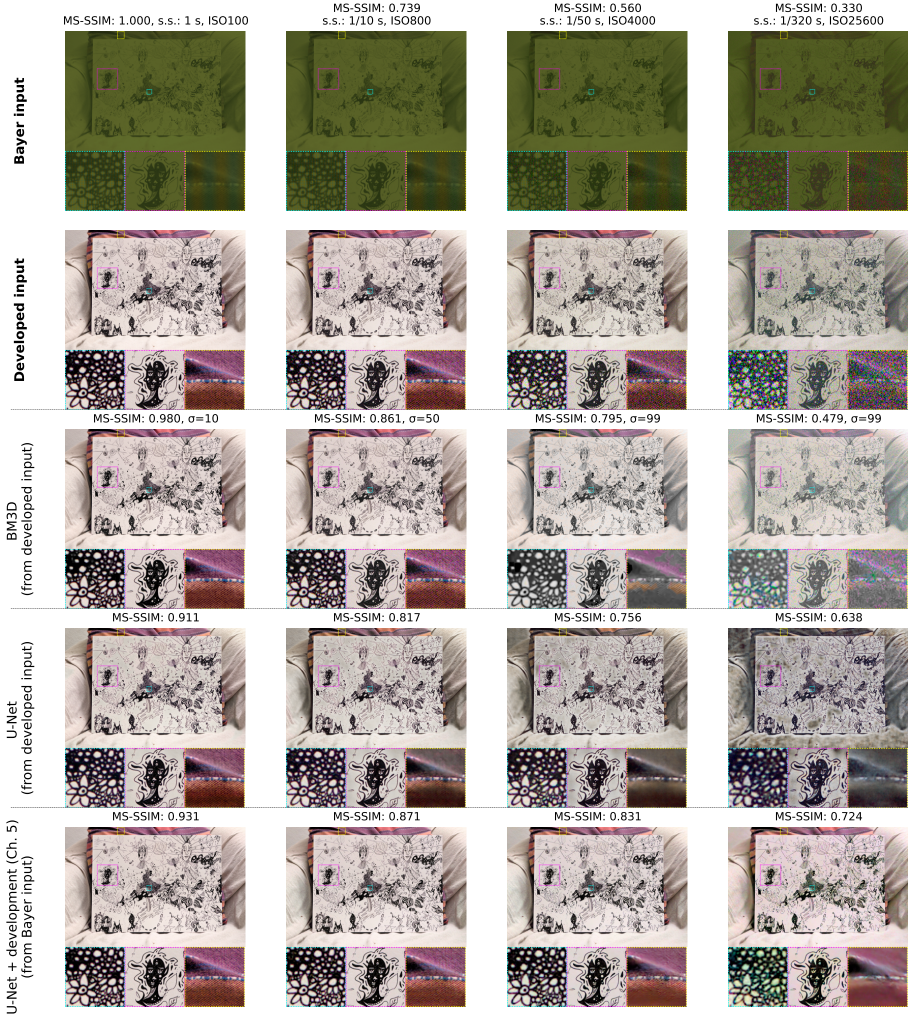


Figure 5.6: Visual and MS-SSIM results for denoising an image from the RawNIND unknown sensor test set (LucieB_bw_drawing1) under varying input noise levels. Development is performed using darktable-cli and the same set of development instructions (XMP file).

the improvement is minimal. Notably, Bayer models generalize well when trained exclusively with RawNIND. For Linear RGB models, the inclusion of clean training data provided no benefit.

Gamma Correction: Applying gamma correction before loss computation did not yield significant benefits and, in some cases, harmed training stability. Bayer models experienced a 0.6% MS-SSIM drop, likely due to the mismatch between the corrected outputs and the intrinsic

characteristics of raw data.

5.4.2 Compression performance

We assessed the compression performance by evaluating the rate-distortion curves of our models. Figure 5.8 shows the bpp versus MS-SSIM performance for different models on the RawNIND test set under various noise conditions.

Our results show that models trained to compress Bayer data consistently achieve superior rate-distortion performance compared to those compressing linear RGB or developed (“sRGB”) images, regardless of noise level. Notably, joint denoising and compression of developed images perform significantly worse than models using linear RGB or Bayer data, underscoring the limitations of compressing developed images in the presence of noise.

As shown in Figure 5.8(a), the joint denoising, demosaicing and compression model generally outperforms sequential models that apply denoising followed by compression. However, consistent with the findings of [12], the sequential approach achieves better rate-distortion performance at the highest bitrates in the presence of strong noise, suggesting that U-Net is a more effective denoising architecture. Merely increasing the number of channels did not yield any improvements in the JDDC model.

Compression-only models, including those compressing linear RGB images or using standard codecs like JPEG XL, struggle significantly in the presence of noise. Image quality even degrades as bitrate increases, as these methods reconstruct more of the input noise. Interestingly, these compression-only models perform some implicit denoising; although this denoising is suboptimal, the compressed noisy images often have slightly higher image quality than the original noisy inputs. However, this implicit denoising is insufficient for practical purposes and does not replace the need for explicit denoising. In our experiments, we were unable to reliably train a Bayer compression-only model, indicating that integrating denoising helps stabilize training and improve results in the raw domain. This underscores the necessity of denoising to achieve effective compression, particularly for raw Bayer data.

Additionally, models compressing upscaled Bayer images (using bi-linear interpolation) achieve rate-distortion performance that is slightly better than models receiving debayered linear RGB inputs, which use more complex edge-aware interpolation methods. The performance of these upscaled Bayer models falls between that of the raw Bayer models and the linear RGB models.

5.4. Results and discussion

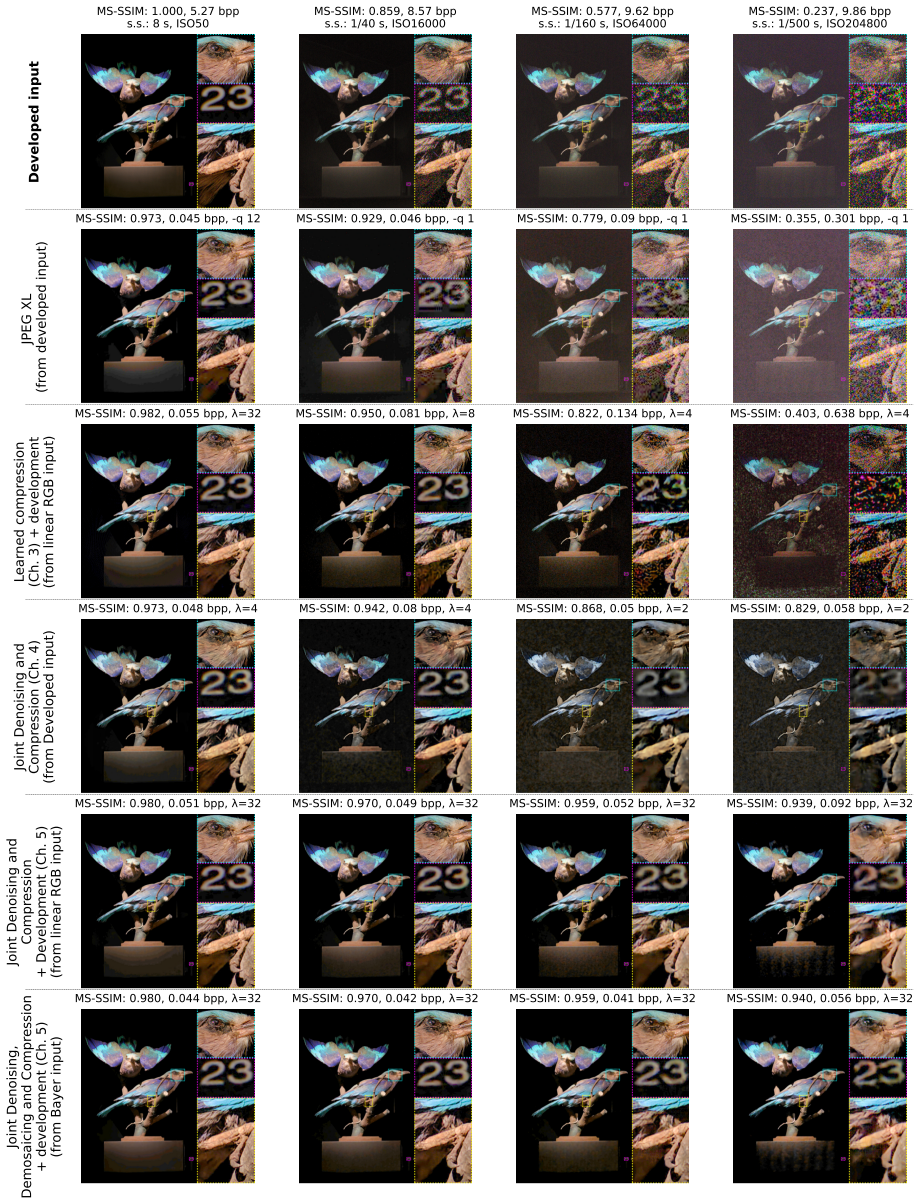


Figure 5.7: Rate-distortion and visual results for compressing a test image (MuseeL-bluebirds-A7C) under varying input noise levels. Results are presented for different compression methods and input formats. The input bpp is measured using lossless xz compression on the raw image. See Figure 5.5 for all input formats.

5: Learning joint denoising, debayering and compression from the Raw Natural Image Noise Dataset

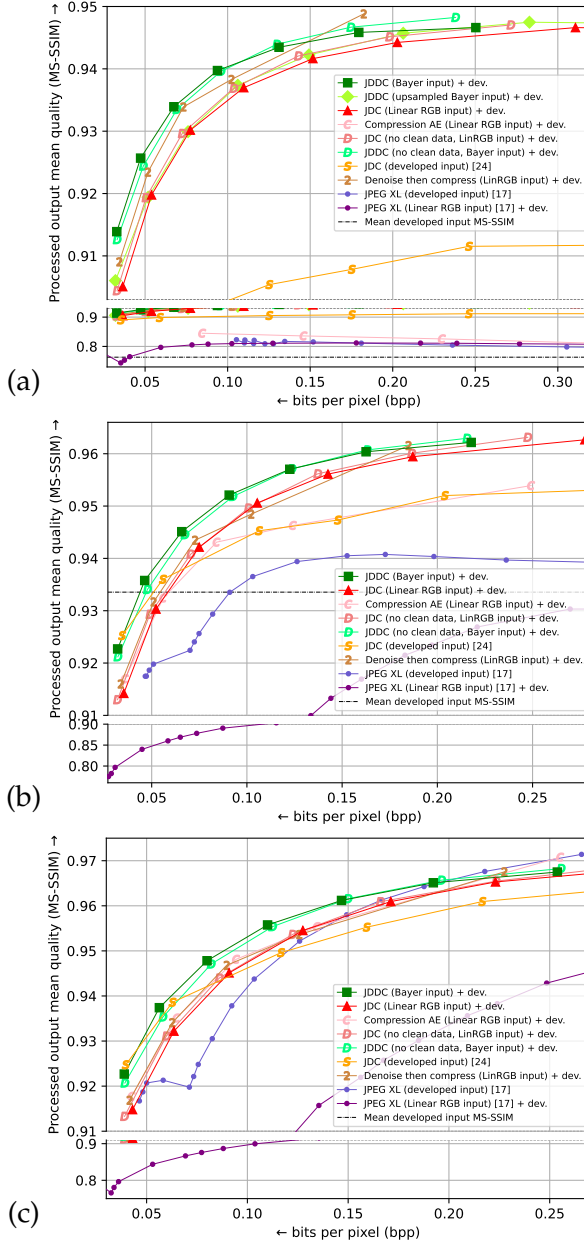


Figure 5.8: Rate-distortion (bpp vs. MS-SSIM) performance of different models on the RawNIND test set after manual development with Dark-table. (a) Entire RawNIND test set (163 image pairs, $\overline{\text{MS-SSIM}} = 0.763$). (b) Low-noise images (80 image pairs, $\overline{\text{MS-SSIM}} = 0.933$). (c) Ground-truth (clean) images only (15 images, $\overline{\text{MS-SSIM}} = 1.00$).

Compressing linear RGB images with JPEG XL and then developing them yields worse results than compressing developed images directly. This aligns with expectations, as JPEG XL is optimized for developed images, and applying it to linear RGB data without adjustments results in suboptimal performance. These findings emphasize the importance of aligning the compression method with the characteristics of the image data.

For clean images (Figure 5.8(c)), the performance gap between the models narrows. Both Bayer and linear RGB joint models perform well, even when trained exclusively on noisy datasets without clean images. Notably, Bayer models still outperform linear RGB models, even those trained solely for image compression. “sRGB” models exhibit a slight advantage over linear RGB models at the lowest bitrates.

Compression-only methods perform adequately on clean images but suffer when even small amounts of noise are present, as shown in Figure 5.8(b). This reinforces the importance of incorporating denoising in compression pipelines for robust performance across varying noise levels.

We also tested training compression models with gamma correction applied before the loss function, but this approach consistently resulted in poor performance, with models ceasing to improve early in training. This suggests that applying gamma correction is not only ineffective but also detrimental to the training process.

5.4.3 Insights and implications

Our experiments demonstrate the advantages of processing raw images over developed images for both denoising and compression tasks. Models operating on raw Bayer data or linear RGB images consistently outperform those trained on developed images, particularly in the presence of noise.

The comparable performance between Bayer and linear RGB models indicates that both methods are effective for denoising. However, Bayer models offer substantial computational savings in both denoising and compression by operating on data with reduced spatial dimensions, processing only one-quarter of the pixels compared to debayered RGB images.

Incorporating clean images from diverse sensors slightly enhances generalization to unknown sensors, especially for Bayer models. The minimal performance drop when working with images from unseen sensors suggests that the models inherently generalize well across different camera types, reducing the need for extensive data acquisition

and retraining across devices.

In compression tasks, integrating denoising is essential for effective performance, as compression-only models struggle with noisy data. We've also observed increasing stability in training models with joint denoising and compression versus those trained for compression only. In addition to being less computationally expensive, Bayer compression models offer superior rate-distortion performance as they operate directly on meaningful image data without the overhead introduced when encoding interpolated RGB values. This hypothesis is supported by the performance of the "pre-upsampled" bilinear Bayer model, which falls between that of Bayer and linear RGB models; simpler upsampling appears to make it easier for the model to discard interpolated data.

For practical applications, these findings suggest that revisiting traditional image development pipelines to incorporate denoising and compression at the raw data level can lead to significant improvements in image quality and data efficiency. One potential workflow involves compressing raw images and transmitting sidecar files containing development instructions, allowing for non-destructive editing while leveraging the benefits of raw data compression. Furthermore, integrating denoising models directly into the raw image development pipeline ensures compatibility with any image development workflow.

5.5 Conclusion

This work addresses the limitations of denoising and compression on developed images by introducing the Raw Natural Image Noise Dataset (RawNIND) and demonstrating the clear advantages of operating directly on raw or linear data. We consistently showed that models using raw Bayer and linear RGB data outperform those using developed images, highlighting the benefits of preserving linearity and avoiding development artifacts. RawNIND enabled the development of denoising models that effectively generalize across diverse camera sensors and image development workflows. Furthermore, our exploration of joint denoising and compression at the raw data level showcases substantial improvements in rate-distortion performance and computational efficiency. This raw-centric approach not only enhances image quality and compression efficiency but also supports flexible, non-destructive editing workflows through the storage of compressed raw data with accompanying development metadata. The comparable performance between Bayer and linear RGB models indicates that both methods are effective for denoising. However, Bayer models offer significant computational

savings in both denoising and compression by operating on data with reduced spatial dimensions, processing only one-quarter of the pixels compared to debayered RGB images. Incorporating clean images from diverse sensors slightly enhances generalization to unknown sensors, especially for Bayer models. In compression tasks, integrating denoising is essential for effective performance, as compression-only models struggle with noisy data, and our joint models show increased training stability. Bayer compression models offer superior rate-distortion performance by operating directly on meaningful image data without interpolative overhead. Future research should explore further architectural optimizations for raw data processing and the integration of these techniques into practical image processing systems. To foster further innovation and collaboration, RawNIND and its associated tools are publicly available.

6

Conclusion

THIS thesis has tackled two interconnected challenges in image processing: achieving high-quality image compression and effective denoising. Traditional approaches often address these tasks separately, neglecting the fact that noise is not only visually disturbing but also inherently non-compressible. Consequently, noise inflates bitrates and degrades the performance of compression algorithms. Motivated by these issues, we explored a variety of techniques—both conceptual and practical—to jointly improve image denoising and compression in a way that reduces computational complexity while achieving superior rate-distortion performance.

Central to this thesis is the assertion that noise removal is best performed early in the image development pipeline and, ideally, should be integrated into the compression process itself. This perspective exposes several drawbacks of conventional workflows, where noise is either transmitted or suppressed after compression, leading to suboptimal rate-distortion trade-offs. By aligning the denoising and compression processes in a single learned model, we have demonstrated significant gains in image quality, especially at lower bitrates, and reduced the computational cost.

One major contribution of this thesis is the introduction of the *Raw Natural Image Noise Dataset (RawNIND)*, a resource designed to facilitate the development of denoising and compression methods that work directly on raw sensor data and that generalize across camera sensors and development pipelines. RawNIND provides paired noisy-clean raw images from multiple sensors to enable robust training of image

denoising models. We integrated this dataset into our research on *joint denoising and compression* to show how working with raw or linear sensor data leads to better noise suppression, improved compression performance, better generalization and increased flexibility for subsequent image development.

Finally, the thesis has advanced the state of the art in learned compression through a novel approach called *competition of prior distributions*, which reduces entropy coding complexity without sacrificing rate-distortion performance. When combined with denoising at the raw data level, we arrive at a pipeline that is efficient both in terms of computational cost and image quality, especially when operating on noise-prone (e.g. low-light or high-speed) scenarios. These conclusions are further supported by comprehensive experimental evaluations indicating that our proposed methods achieve better visual quality, improved compression efficiency, and greater generalization capabilities.

6.1 Results summary

The results of this thesis are presented in three main chapters:

Chapter 3 focused on improving learned image compression by simplifying the entropy coding stage. Our proposed *competition of priors* scheme utilizes multiple static, learned distributions (priors) to model different portions of the latent representation. This approach rivals hyperprior-based codecs in terms of rate-distortion performance yet offers substantially lower complexity in entropy coding and decoding. The key insight is that each latent spatial location (across all channels) can be encoded using the single best adapted prior among a learned set, removing the need for a per-feature parametric distribution. As a result, the overhead associated with generating hyperprior-associated distributions at inference time is substantially reduced. The proposed method delivers competitive rate-distortion performance to state-of-the-art models while substantially reducing the overhead associated with encoding and (more so) decoding.

In Chapter 4, we introduced the concept of *joint denoising and compression*, building on the observation that noise is inherently non-compressible. We trained a single compression autoencoder to directly produce a denoised—and hence higher-quality—image as part of the compression pipeline. By leveraging noisy-clean image pairs from the Natural Image Noise Dataset [9] (NIND), the model learns to discard noise early

in the encoding stage, effectively suppressing bitrate allocation to random noise. This yields a superior rate-distortion trade-off compared to either compressing noisy images directly or applying denoising as a separate, sequential step. Importantly, the joint model substantially reduces computational requirements because the same encoder-decoder architecture accomplishes both tasks. Experiments confirm that for noisy inputs, joint denoising and compression can outperform even a two-step pipeline of high-quality denoising followed by compression while using an order of magnitude fewer operations.

Chapter 5, joint denoising, demosaicing and compression (JDDC), extends our integrated approach to noise removal and image compression in the raw domain, and shows the benefits thereof. Supported by RawNIND which is introduced as part of this work, we demonstrated that operating on *raw Bayer* or *linear RGB* data enables markedly improved denoising and rate-distortion performance (thus image quality) in the presence of noise. This can be attributed to two main factors: first, raw data is free from complex non-linearities introduced by conventional camera ISPs, improving the efficacy of learned denoising and compression; second, *Bayer data*, which stores only one color channel per pixel, can be stacked to form a 4-channel image which naturally contains one-quarter of the spatial dimensions compared to fully developed RGB images, thereby reducing both the computational complexity of neural network operations and the overhead in entropy coding. Indeed, our experiments show that by working on Bayer inputs, it is possible to achieve an approximately *fourfold reduction* in complexity for both the denoising network and the compression encoder while producing outputs that largely surpass the quality of methods operating on fully developed RGB images and the rate-distortion obtained from linear RGB images. Furthermore, the JDDC approach showed strong generalization across sensors, indicating that learned models trained on raw data are less susceptible to variations in camera hardware or processing pipelines, and it allows generalization across and integration into any image development pipeline due to its generic nature.

6.2 Perspectives

6.2.1 Denoising outlook

One of the most impactful contributions of this thesis is the demonstration of highly effective denoising methods integrated into real-world im-

age workflows, underpinned by the Raw Natural Image Noise Dataset. The practical value of early-stage denoising is evident, as our raw denoising models can be readily integrated into camera ISPs or raw development software such as Darktable. This would enable direct application of denoising to raw images, thus mitigating noise amplification in subsequent non-linear operations like deblurring and sharpening, and making it impossible for a model to overfit on a specific development pipeline.

The Raw Natural Image Noise Dataset itself opens numerous avenues for further study. Future work could explore simplifying existing architectures and designing specialized neural networks that operate more efficiently on raw data, allowing effective denoising models to run on most computers and image acquisition devices. By pushing denoising further up in the processing stack, manufacturers and software developers could reduce downstream complexity, preserve more meaningful data for editing, and ensure that images remain as close to the physically captured signal as possible.

6.2.2 Compression outlook

On the compression side, this thesis has made a case for *joint denoising and compression*, demonstrating that explicitly modeling noise and discarding it early yields a more efficient and simpler overall system. The competition-of-priors method introduced in Chapter 3 offers a promising direction for *simplified learned compression* compared to hyperpriors and autoregressive-based models. Notably, operating on Bayer data enables further rate-distortion improvements, and for the encoder to leverage reduced spatial dimensions and thus reduced processing overheads. Future directions here include:

1. *Architectural optimizations tailored to raw data.* Although the compression encoder we used benefits from performing a quarter as many operations when compressing Bayer images, the decoder performs an additional convolution near the final upsampling stage, which introduces a computational bottleneck that offsets most of the efficiency gains achieved during earlier compression stages. This last layer could be optimized to benefit from the lower complexity afforded to the encoder.
2. *Architectural optimizations targeting stronger denoising.* While we investigated the use of an off-the-shelf compression autoencoder to perform joint denoising and compression and found that it outperforms sequential approaches in most cases, strong denoisers like the U-Net still provide superior denoising (and rate-distortion)

at the highest bitrates in the presence of strong noise. This benefit may be inherent to the U-Net’s skip connections (which are inherently incompatible with compression), but different compression architectures should be investigated nonetheless.

3. *Standardization and industry adoption* While current image development workflows would immediately benefit from the raw denoising methodology we propose, the same cannot always be said about new compression methods because their success depends on compatibility which is driven by wide adoption and standardization. An effort could be made to promote our proposed *competition of prior distributions, joint denoising and compression, and joint denoising, demosaicing and compression* methods to a standardization body such as the JPEG AI committee.

6.2.3 Limitations and future directions

While our results are promising, several limitations merit discussion. First, working on effective research and development with raw data requires more work and certain knowledge of its characteristics such as specific color profiles and the underlying processing pipeline, because few off-the-shelf tools are available due to the diverse nature of raw images. Moreover, certain sensors with incompatible color filter arrays would need to work with our proposed linear RGB approach and forego some of the computational and rate-distortion benefits offered by Bayer models. Moreover, additional work would be useful to further optimize the denoising model and the compression decoder’s architecture in order to minimize computational complexity and democratize the use of such models on most computing device with a reasonable runtime and memory requirements. This will directly enable the integration of powerful (results-wise) yet light (resources-wise) denoising models directly into the pixel pipeline of image development software such as Darktable.

In summary, this thesis has established that noise removal can and should be an integral component of learned compression pipelines, especially when working with raw or linear data. By addressing noise early and leveraging the reduced spatial dimensionality in Bayer data, it is possible to produce higher-quality images at lower bitrates and with decreased computational complexity. The Raw Natural Image Noise Dataset has enabled the demonstration of these benefits while ensuring that these solutions remain robust across diverse cameras and workflows. We hope that these ideas will spur further research into

practical, efficient, and universally applicable algorithms for real-world image processing.

Acknowledgements

This research has been funded by the Walloon Region and intoPIX. Christophe De Vleeschouwer is funded by the 'Fond de la Recherche Scientifique de Belgique (F.R.S.-FNRS)'. Computational resources have been provided by the supercomputing facilities of the Université catholique de Louvain (CISM/UCL) and the Consortium des Équipements de Calcul Intensif en Fédération Wallonie Bruxelles (CÉCI) funded by the Fond de la Recherche Scientifique de Belgique (F.R.S.-FNRS) under convention 2.5020.11 and by the Walloon Region. 31 of the 310 RawNIND scenes were captured by the following contributors: Iain Fergusson, Matti Blume, Peter Wemmert, and "sillyxone".

Bibliography

- [1] Featured pictures on Wikimedia Commons. https://commons.wikimedia.org/wiki/Category:Featured_pictures_on_Wikimedia_Commons. Accessed: 2020-04-03.
- [2] Abdelrahman Abdelhamed, Stephen Lin, and Michael S. Brown. A High-Quality Denoising Dataset for Smartphone Cameras . In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* , pages 1692–1700, Los Alamitos, CA, USA, June 2018. IEEE Computer Society.
- [3] Bouzid Arezki, Fangchen Feng, and Anissa Mokraoui. Convolutional Transformer-Based Image Compression. In *2023 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA)*, pages 154–159. IEEE, September 2023.
- [4] Johannes Ballé. Efficient Nonlinear Transforms for Lossy Image Compression. In *2018 Picture Coding Symposium (PCS)*, 2018.
- [5] Johannes Ballé, Sung Jin Hwang, Nick Johnston, and David Minnen. compression. <https://github.com/tensorflow/compression>, 2020.
- [6] Johannes Ballé, Valero Laparra, and Eero Simoncelli. End-to-end optimized image compression. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, apr 2017*.
- [7] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. In *International Conference on Learning Representations*, 2018.
- [8] Yochai Blau and Tomer Michaeli. The Perception-Distortion Trade-off. June 2018.

- [9] Benoit Brummer and Christophe De Vleeschouwer. Natural Image Noise Dataset. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1777–1784. IEEE, June 2019.
- [10] Benoit Brummer and Christophe De Vleeschouwer. Adapting JPEG XS gains and priorities to tasks and contents . In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 629–633, Los Alamitos, CA, USA, June 2020. IEEE Computer Society.
- [11] Benoit Brummer and Christophe De Vleeschouwer. End-to-end optimized image compression with competition of prior distributions. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1890–1894. IEEE, June 2021.
- [12] Benoit Brummer and Christophe De Vleeschouwer. On the Importance of Denoising when Learning to Compress Images. In *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2439–2447. IEEE, January 2023.
- [13] A. Buades, B. 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)*, volume 2, pages 60–65 vol. 2, 2005.
- [14] Chen Chen, Qifeng Chen, Jia Xu, and Vladlen Koltun. Learning to See in the Dark. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3291–3300, jun 2018.
- [15] Zhengxue Cheng, Heming Sun, Masaru Takeuchi, and Jiro Katto. Learned image compression with discretized gaussian mixture likelihoods and attention modules. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.
- [16] Lasse Collin and Igor Pavlov. Xz utils. <https://tukaani.org/xz/>, Jul 2009.
- [17] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes Dataset for Semantic Urban Scene Understanding. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

- [18] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Image denoising with block-matching and 3D filtering. In Nasser M. Nasrabadi, Syed A. Rizvi, Edward R. Dougherty, Jaakko T. Astola, and Karen O. Egiazarian, editors, *Image Processing: Algorithms and Systems, Neural Networks, and Machine Learning*, volume 6064, page 606414. International Society for Optics and Photonics, SPIE, 2006.
- [19] D.L. Donoho. De-noising by soft-thresholding. *IEEE Transactions on Information Theory*, 41(3):613–627, 1995.
- [20] Michaël Gharbi, Gaurav Chaurasia, Sylvain Paris, and Frédo Durand. Deep joint demosaicking and denoising. *ACM Trans. Graph.*, 35(6), December 2016.
- [21] Rafael Gonzalez and Zahraa Faisal. *Digital Image Processing*. 2017.
- [22] Shi Guo, Zifei Yan, Kai Zhang, Wangmeng Zuo, and Lei Zhang. Toward Convolutional Blind Denoising of Real Photographs. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1712–1722, 2019.
- [23] Nikolaus Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.
- [24] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the Knowledge in a Neural Network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015.
- [25] K. Hirakawa and T.W. Parks. Joint demosaicing and denoising. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, volume 3, pages III–309, 2005.
- [26] International Telecommunication Union. Recommendation ITU-R BT.2020: Parameter values for ultra-high definition television systems for production and international programme exchange, 2012.
- [27] ISO. *Information technology — Computer graphics and image processing — Portable Network Graphics (PNG)*. Geneva, CH, Mar 2004.
- [28] ISO. *Information technology — JPEG XS low-latency lightweight image coding system — Part 1: Core coding system*, volume 2019. ISO, Geneva, CH, May 2019.

- [29] ISO/IEC JTC 1/SC 29/WG 1. Information technology – JPEG XL image coding system – Part 1: Core coding system. Standard ISO/IEC 18181-1:2024, International Organization for Standardization, Geneva, CH, July 2024.
- [30] Viren Jain and H. Sebastian Seung. Natural image denoising with convolutional networks. In *Proceedings of the 21st International Conference on Neural Information Processing Systems, NIPS’08*, pages 769–776, Red Hook, NY, USA, 2008. Curran Associates Inc.
- [31] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.
- [32] Mark A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37(2):233–243, 1991.
- [33] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [34] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015.
- [35] Xin Li. Demosaicing by successive approximation. *IEEE Transactions on Image Processing*, 14(3):370–379, 2005.
- [36] Jiaheng Liu. compression. <https://github.com/liujiaheng/compression>, 2020.
- [37] Jiaheng Liu, Guo Lu, Zhihao Hu, and Dong Xu. A unified end-to-end framework for efficient deep image compression. *arXiv preprint arXiv:2002.03370*, 2020.
- [38] Lin Liu, Xu Jia, Jianzhuang Liu, and Qi Tian. Joint Demosaicing and Denoising With Self Guidance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, jun 2020.
- [39] Ming Lu, Peiyao Guo, Huiqing Shi, Chuntong Cao, and Zhan Ma. Transformer-based Image Compression. In *2022 Data Compression Conference (DCC)*, pages 469–469, 2022.

- [40] Shunta Maeda. Fast and flexible image blind denoising via competition of experts. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2239–2247, 2020.
- [41] Detlev Marpe, Heiko Schwarz, and Thomas Wiegand. Context-based adaptive binary arithmetic coding in the h.264/avc video compression standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):620–636, 2003.
- [42] Gloria Martín. Range encoding: an algorithm for removing redundancy from a digitised message. In *Video and Data Recording Conference, Southampton, 1979*, pages 24–27, 1979.
- [43] Fabian Mentzer. torchac. <https://github.com/fab-jul/L3C-PyTorch/tree/master/src/torchac>, 2020.
- [44] Fabian Mentzer, Eirikur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool. Practical full resolution learned lossless image compression. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10621–10630, 2019.
- [45] Joseph Meola, Michael T. Eismann, Randolph L. Moses, and Joshua N. Ash. Modeling and estimation of signal-dependent noise in hyperspectral imagery. *Appl. Opt.*, 50(21):3829–3846, Jul 2011.
- [46] D. Minnen, G. Toderici, S. Singh, S. J. Hwang, and M. Covell. Image-dependent local entropy models for learned image compression. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 430–434, 2018.
- [47] David Minnen and S. Singh. Channel-wise autoregressive entropy models for learned image compression. pages 3339–3343, 2020.
- [48] David C. Minnen, Johannes Ballé, and George Toderici. Joint Autoregressive and Hierarchical Priors for Learned Image Compression. In *Neural Information Processing Systems*, 2018.
- [49] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning, ICML’10*, pages 807–814, Madison, WI, USA, 2010. Omnipress.
- [50] G. Parascandolo, N. Kilbertus, M. Rojas-Carulla, and B. Schölkopf. Learning independent causal mechanisms. In *Proceedings of the 35th*

- International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pages 4033–4041. PMLR, July 2018.
- [51] Tobias Plötz and Stefan Roth. Benchmarking Denoising Algorithms with Real Photographs. In *2017 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPR)*, pages 2750–2759, 2017.
 - [52] Thomas Richter, Joachim Keinert, Antonin Descampe, and Gael Rouvroy. Entropy coding and entropy coding improvements of jpeg xs. In *2018 Data Compression Conference*, pages 87–96, 2018.
 - [53] Jorma J. Rissanen. Generalized kraft inequality and arithmetic coding. *IBM Journal of Research and Development*, 20(3):198–203, 1976.
 - [54] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing.
 - [55] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.
 - [56] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1874–1883, 2016.
 - [57] A. Skodras, C. Christopoulos, and T. Ebrahimi. The JPEG 2000 still image compression standard. *IEEE Signal Processing Magazine*, 18(5):36–58, 2001.
 - [58] Vladislav Sovrasov. Flops counter for convolutional networks in pytorch framework. <https://github.com/sovrasov/flops-counter.pytorch>, 2020.
 - [59] Michael Stokes, Matthew Anderson, Srinivasan Chandrasekar, and Ricardo Motta. A standard default color space for the internet — srgb. <http://www.color.org/contrib/sRGB.html>, November 1996.

- [60] Gary J. Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand. Overview of the high efficiency video coding (hevc) standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1649–1668, 2012.
- [61] Michela Testolina, Evgeniy Upenik, and Touradj Ebrahimi. Towards image denoising in the latent space of learning-based compression. In Andrew G. Tescher and Touradj Ebrahimi, editors, *Applications of Digital Image Processing XLIV*, volume 11842, pages 412–422. SPIE, 2021.
- [62] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
- [63] G.K. Wallace. The JPEG still picture compression standard. *IEEE Transactions on Consumer Electronics*, 38(1):xviii–xxxiv, 1992.
- [64] 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*, volume 2, pages 1398–1402 Vol.2, Nov 2003.
- [65] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising. *Trans. Img. Proc.*, 26(7):3142–3155, July 2017.



♥🐱 In loving memory ☺