UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

BERNARDO HENZ

Improving Digital Photography: Revisiting Core Aspects through a Deep-Learning Lens

Thesis presented in partial fulfillment
of the requirements for the degree of
Doctor of Computer Science

Advisor: Prof. Dr. Manuel Menezes de Oliveira Neto

Porto Alegre
December 2020

CIP — CATALOGING-IN-PUBLICATION

Henz, Bernardo

Improving Digital Photography:
Revisiting Core Aspects through a Deep-Learning Lens / Bernardo Henz. – Porto Alegre: PPGC da UFRGS,

.

104 f.: il.

Thesis (Ph.D.) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS,

. Advisor: Manuel Menezes de Oliveira Neto.

1. Digital photography. 2. Color filter array. 3. Demosaicing. 4. Natural noise. 5. Denoising. I. Menezes de Oliveira Neto, Manuel. II. Título.

“It is not knowledge, but the act of learning,
not possession but the act of getting there,
which grants the greatest enjoyment.”

— CARL FRIEDRICH GAUSS

ABSTRACT

This dissertation revisits fundamental aspects of computational photography using state-of-the-art deep-learning strategies to improve them. We propose two novel techniques that enhance the digital-photography pipeline. More specifically, we show how to improve image acquisition and noise handling. For image acquisition, we present a novel approach based on convolutional neural networks for jointly optimizing the design of color filter arrays and demosaicing. Our trained models provide high-quality color reconstruction and achieve higher peak signal-to-noise ratio (PSNR) values on standard datasets, surpassing previous techniques for both noise-free and noisy data. For noise handling, we propose an architecture based on generative adversarial networks (GANs) that is capable of adjusting the noise level of a given image. Through validation tests, we show that the noise generated by our models are much closer to what is found in photographs when compared to existing methods (Gaussian/Poisson noise). We evaluate the use of our generative models when training a denoiser, showing that training with images synthesized by our models indeed have superior performance on natural-images denoise benchmarks. By improving core steps of the digital-imaging pipeline, our methods have the potential to improve the overall quality of digital photographs.

Keywords: Digital photography. Color filter array. Demosaicing. Natural noise. Denoising.

Aperfeiçoando a Fotografia Digital:
Revisitando Aspectos Fundamentais sob uma Visão de Deep Learning

RESUMO

Esta tese revisita aspectos fundamentais da fotografia computacional, enquanto os aperfeiçoa utilizando estratégias de deep-learning do estado da arte. Esta tese propõe duas técnicas que aprimoram o pipeline de fotografia digital. Especificamento, nós mostramos como melhorar o processo de aquisição de imagens e o tratamento de ruído natural. Para aquisição de imagens, nós apresentamos uma técnica inovadora baseada em Convolutional Neural Networks que otimiza conjuntamente o design de *color filter arrays* e o método de *demosaicing*. Os modelos treinados proporcionam reconstruções de alta qualidade, alcançando valores de PSNR (Peak Signal-to-Noise Ratio) maiores em diversos datasets, superando técnicas anteriores tanto para o caso de imagens sem ruído quanto para imagens ruidosas (*demosaicing+denoising*). Para o tratamento de ruído, nós propomos uma arquitetura baseada em GANs (*Generative Adversarial Networks*) que é capaz de ajustar o nível de ruído presente em uma imagem. Através de testes de validação, nós mostramos que o ruído sintetizado por nossos modelos é muito mais próximos que o encontrado em fotografias reais quando comparado com os métodos existentes (Gaussiano/Poisson). Nósavaliamos o uso de nossos modelos generativos no treinamento de *denoisers*, mostrando que os *denoisers* treinados utilizando imagens sintetizadas pela nossa ténica conseguem uma performance superior em *benchmarks* de remoção de ruído de images naturais. Visto que nossas técnicas melhoram aspectos importante do *pipeline* de imagens digitais, elas têm o potencial de melhorar a qualidade geral de fotografias digitais.

Palavras-chave: fotografia digital, color filter array, demosaicing, denoising, composição de imagens.

LIST OF ABBREVIATIONS AND ACRONYMS

AI	Artificial Intelligence
MLP	Multi-Layer Perceptrons
ReLU	Rectified Linear Unit
GAN	Generative Adversarial Network
CFA	Color Filter Array
CNN	Convolutional Neural Network
PSNR	Peak Signal-to-Noise Ratio
CPSNR	Color Peak Signal-to-Noise Ratio
MSE	Mean-Squared Error
VAE	Variational Autoencoder
IR	Infrared
NIR	NearInfrared
HDR	High Dinamic Range
AWGN	Additive White Gaussian Noise
KL div	Kullback-Leibler divergence
KS test	Kolmogorov-Smirnov test
SIDD	Smartphone Image Denoising Dataset

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

The advent of digital photography brought several advantages over traditional analog photography, such as reduced camera sizes and instant image previews. But most important, digital photography has also led to several technical improvements, such as higher-resolution images[1], faster capture times, less noise, higher dynamic range, image stabilization, and post-production possibilities (MILBURN; ROCKWELL; CHAMBERS, 2002; SIMON, 2004; WHITE; DOWNS, 2005; GUSTAVSON; HOUSE, 2012; ARCHAMBAULT, 2015b). Nonetheless, digital photography still faces several challenges, ranging from image acquisition to post-processing.

Research on digital photography has grown substantially in recent years, and many research subareas have emerged and broadened. Image acquisition, image manipulation, and computational photography are a few examples. For instance, recent works on image acquisition propose distinct color filter arrays (CFAs) (HIRAKAWA; WOLFE, 2008; CONDAT, 2009; CONDAT, 2010; CONDAT, 2011; HAO et al., 2011; CHAKRABARTI; FREEMAN; ZICKLER, 2014; BAI et al., 2016; CHAKRABARTI, 2016), new demosaicing methods for the current Bayer CFA (WANG, 2014; KAUR; BANGA, 2015; GHARBI et al., 2016; RATNASINGAM, 2019), and new ways to handle noise (DABOV et al., 2007; BURGER; SCHULER; HARMELING, 2012; ZHANG et al., 2016; CHEN et al., 2018; BROOKS et al., 2019). Many more works focus on image manipulation: only in the past five years there have been dozens of new techniques addressing image inpainting (LIU et al., 2016; THONAT et al., 2016; YI et al., 2020), style transferring (YAN et al., 2015; CHANG et al., 2015; LEE et al., 2016; HEO; LEE; JUNG, 2016; KIM et al., 2020), edge-aware filtering (XU et al., 2015; CHAMPANDARD, 2016; DENG, 2016), automatic colorization of grayscale images (LARSSON; MAIRE; SHAKHNAROVICH, 2016; ZHANG; ISOLA; EFROS, 2016; IIZUKA; SIMO-SERRA; ISHIKAWA, 2016; SHAMSABADI; SANCHEZ-MATILLA; CAVALLARO, 2020), image composition (ZHU et al., 2015; ZHAO et al., 2015; KEMELMACHER-SHLIZERMAN, 2016; TSAI et al., 2016; SHOCHER et al., 2020), saliency-map estimation (WANG et al., 2015; LIU et al., 2015; BYLINSKII et al., 2016; PAN et al., 2016; TSIAMI; KOUTRAS; MARAGOS, 2020), and many others (SHIH et al., 2015; SCHULER et al., 2016; FRIED et al., 2016; Xu et al., 2017; ZHANG et al., 2020).

Despite the number of works that improved the digital-photography pipeline in

[1] compared to 35mm films

some way, there is still need for improvement in several areas. For instance, currently, most digital cameras rely on color filter arrays to sparsely sample colors that are interpolated by a demosaicing algorithm. In addition, noise is still a challenge when capturing photographs. Such noise may be introduced by high temperature, statistical quantum fluctuations (*e.g.*, shot noise), electromechanical interference, and many other factors related to the electronic circuitry responsible for capturing and processing the image. In this dissertation, we focus on these two core problems: the capture of color images through CFAs, and noise handling. We make use of deep learning algorithms for designing CFAs and demosaicing methods that achieve higher quality reconstructions, as well for synthesizing noise with greater resemblance to natural noise. We demonstrate that the noise synthesized by our method improves the performance of a trainable denoiser. Since practically all digital cameras today make use of CFAs for acquiring color pictures, and all digital photographs suffer (to a certain degree) from noise, the proposed methods have direct impact on the quality of digital photography.

1.1 Thesis Statement

This dissertation introduces novel methods for improving the quality of digital photographs. Its central idea can be summarized as:

> *It is possible to use deep learning techniques to improve the acquisition of color during the digital imaging pipeline, as well as to synthesize camera noise, which can then be used for image denoising. These techniques have the potential to improve the overall quality of digital photographs.*

I will demonstrate this statement by proposing techniques to improve two fundamental aspects of digital photography: first, a technique for simultaneous design of CFAs and demosaicing strategies that improves the acquisition of color images; and second, a technique for synthesizing camera noise. Such more realistic noise can be used for training denoisers that more effectively reduce/remove noise from real photographs.

1.2 Contributions

This dissertation addresses two important aspects of digital photography: the color acquisition, and noise handling. Its main **contributions** include:

- *A method for the joint design of CFA pattern and demosaicing that minimizes color-reconstruction errors* (Section 3.3). Our model is the first to optimize CFA colors over the entire RGB color space, while jointly optimizing demosaicing. The results produced by our system outperform previous solutions in terms of PSNR for both noise-free and noisy data (Section 3.4);
- *An autoencoder architecture that models the color-image capture process on single monochromatic sensors*. Our architecture achieves fast training convergence on image patches, and works with CFAs of different sizes, including existing ones (Section 3.3.1);
- *A joint design of CFA and demosaicing for capturing NIR along with visible light (Section 3.4.3). Our method is the first one to consider the use of color filters that capture NIR and visible information jointly;*
- *A method for adjusting the noise level of an input photograph to match a target ISO level (Section 4.4). Its results produce significantly better approximations to natural noise than previous synthetic noise generators;*
- *A new loss formulation for use with CycleGANs for allowing the adjustment of noise level* (Section 4.4.4). Such new loss function results in more realistic noise, whose statistics approximate the ones of real photographs with the same ISO value.

1.3 Publications

This work resulted in two publications: *Deep Joint Design of Color Filter Arrays and Demosaicing* (HENZ; GASTAL; OLIVEIRA, 2018), published at Eurographics 2018 (Chapter 3); and *Synthesizing Camera Noise using Generative Adversarial Networks* (HENZ EDUARDO S. L. GASTAL, 2020), published at IEEE Transactions on Visualization and Computer Graphics 2020 (Chapter 4). We provide the implementations of these techniques, as well as their trained models and data, in the following webpages:

- https://bernardohenz.github.io/projects/joint_cfa_demosaicing
- https://bernardohenz.github.io/projects/synthesizing_noise

1.4 Thesis Structure

This dissertation is organized as follows. Chapter 2 presents an introduction to Convolutional Neural Networks (CNNs) and Generative Adversarial Networks (GANs), which are central to this research.

Chapter 3 presents a novel approach for jointly optimizing the design of CFA and demosaicing, encountering pairs (CFA+demosaicing strategy) that yield better image reconstructions, with better color fidelity. We also show how to train a model suited for images corrupted with noise, where our method also surpasses existing ones. It is worth to mention that, as CFAs and demosaicing are the standard way of capturing color information on image sensors, any improvements on them should increase the quality of digitally-captured images.

Chapter 4 covers an old, yet recurrent problem: noise. While there are many works focusing on estimating/attenuating noise, they are still far from being fully reliable. The main reason for that is the lack of a good model for synthesizing realistic noise. While many methods for denoising train and validate their results on images corrupted by synthetic noise, they tend to not perform well in the presence of natural noise[2]. This happens because natural noise is much more complex, and cannot be modeled by a simple Gaussian model. We present a model that generates noise closer to what is found in digital photographs, and demonstrate that training a denoiser with such a model results in superior performance on natural-noise benchmarks compared to previous methods.

Chapter 5 summarizes this dissertation and discusses potential ideas for future exploration.

[2]We refer to natural noise as the one associated with the image acquisition process.

2 BACKGROUND AND RELATED WORKS

This Chapter presents a brief review of Convolution Neural Networks (CNNs) and Generative Adversarial Networks (GANs), as they provide the foundation for the techniques described throughout this dissertation. We start by giving an intuitive and gentle introduction, followed by a list of recent works that use them to improve the Digital Photography area. For a more detailed explanation, we refer to the readers the following references (LI; JOHNSON; YEUNG, 2017; GOODFELLOW YOSHUA BENGIO, 2016; CHOLLET, 2020).

2.1 Preliminaries

Convolutional neural networks were first introduced in 1998 (LECUN et al., 1998), but they had only become popular in computer vision, image-processing, and computer graphics very recently (KRIZHEVSKY; SUTSKEVER; HINTON, 2012). This became possible due to the power and large memory sizes of modern GPUs, and to the availability of tons of image datasets. CNNs that once were not powerful enough to compete with human-engineered algorithms now generate state-of-the-art results, surpassing humans in several tasks. Image classification was one of the first problems that CNNs were used for (SIMONYAN; ZISSERMAN, 2014; SZEGEDY; IOFFE; VANHOUCKE, 2016), and now it achieves better accuracy than humans (HE et al., 2016). This was highly motivated by dataset challenges like ImageNet (RUSSAKOVSKY et al., 2015), Microsoft COCO: Common Objects in Context (LIN et al., 2014), and others. Besides image classification, such challenges evaluate other tasks; needless to say that CNNs had the top scores on object detection (WANG et al., 2016; LI et al., 2016; DAI et al., 2017; CAO et al., 2020), people-keypoints detection (FATHI et al., 2016; NEWELL; YANG; DENG, 2016), and image captioning (VINYALS et al., 2014; FANG et al., 2014; CHEN et al., 2015; TRAN; MATHEWS; XIE, 2020; SAMMANI; MELAS-KYRIAZI, 2020).

Other works had also addressed many known problems: human-pose estimation (TOSHEV; SZEGEDY, 2014; WANG; TIGHE; MODOLO, 2020), superresolution (KIM; LEE; LEE, 2015; DONG et al., 2016; JOHNSON; ALAHI; LI, 2016; BAHAT; MICHAELI, 2020; HUSSEIN; TIRER; GIRYES, 2020), intrinsic decomposition (NARIHIRA; MAIRE; YU, 2015; ZHOU; KRÄHENBÜHL; EFROS, 2015; LETTRY; VANHOEY; GOOL, 2018; LIU et al., 2019), multiple-illuminant estimation (BIANCO; CU-

SANO; SCHETTINI, 2015; XU et al., 2020). Even subjective problems such as judging aesthetics (WANG; LIN; MECH, 2015; MAI; JIN; LIU, 2016; WANG et al., 2019), saliency-map estimation (MNIH et al., 2014; PAN et al., 2016; LIU et al., 2015; PAN et al., 2016; TSIAMI; KOUTRAS; MARAGOS, 2020), and semantic- or artistic-style transfer (GATYS; ECKER; BETHGE, 2016; ENGSTROM, 2016; CHAMPANDARD, 2016; HEO; LEE; JUNG, 2016; KIM et al., 2020) are addressed using CNNs. In fact, in the last 7 years, CNNs are being used in any problem from face recognition (TAIGMAN et al., 2014; LIU et al., 2019) to image generation (DENTON et al., 2015; MANSIMOV et al., 2015; KARRAS; LAINE; AILA, 2019), image deconvolution (XU et al., 2014; ALJADAANY; PAL; SAVVIDES, 2019), deblurring (SCHULER et al., 2016; ZHANG et al., 2019), dehazing (CAI et al., 2016; LIU et al., 2019), and even automatic colorization of grayscale images (LARSSON; MAIRE; SHAKHNAROVICH, 2016; ZHANG; ISOLA; EFROS, 2016; IIZUKA; SIMO-SERRA; ISHIKAWA, 2016; SHAMSABADI; SANCHEZ-MATILLA; CAVALLARO, 2020).

CNNs are powerful tools that have been widely used in many areas. Due to their characteristics, they are the most suited architecture to be used with images, being our choice for addressing our problems. This chapter briefly introduces CNNs, providing basic concepts and a practical overview that are required for a better understanding of our work. After CNNs, we discuss Generative Adversarial Networks, which will be used in the work of Chapter 4.

2.2 Machine Learning Basics

The basic pipeline of *supervised* training algorithms can be resumed as: (1) take some data; (2) train a model to fit the training data; (3) use the trained model to make predictions on new data. Training a model is nothing more than exposing the model to known data, making it to iteratively learn patterns on the (training) data. At each iteration, the model processes the input data, generating an output that is compared to some ground truth. This feedback is used to update the model, improving its ability to make better predictions.

This entire process is math based: the model is composed by a set of mathematical operations that, when applied to some input data (*forward-pass*), generates an output. Such output is compared to the ground-truth by a *loss function*, which indicates how good the prediction is (a lower loss value indicates a lower distance between the predicted

and ground-truth values, *i.e.*, a good prediction). The error in the loss function is back-propagated to all internal weights of the model's operations (*backward-pass*). An *optimizer*, often based on gradient-descent, updates the trainable weights aiming to minimize the *loss function*. The training phase consists of several of these *forward* and *backward passes*, while the *optimizer* updates the weights hoping that, after some iterations, the model has learned to output values similar to what is desired (ground-truth) for a given input.

2.2.1 Neural Networks

A neural network is a model composed by several operations, very often matrix multiplications, intercalated by non-linear functions (also called *activation functions*). Such a combination of linear and non-linear transformations provides higher expressive power for the model when learning to identify distinct patterns in data. Neural networks are often organized in consecutive layers, each one followed by an activation function. The number of layers, as well as their size (number of parameters inside each one) are hyper-parameters. Although there are many options for activation functions, the most common is the ReLU (Rectified Linear Unity) or some of its derivations, due to its simplicity, fast computation, and non-generation of vanishing gradients (like $tanh$, for example).

2.2.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs), as the name suggests, are neural networks that use convolutional layers. Such operation (convolution) is very good for identifying patterns on images. *Conv* layers are capable of learning spatial correlations among pixels, requiring fewer trainable parameters (the kernels of the layer) than dense layers (*i.e.*, fully-connected layers). Because of this nature, CNNs are less susceptible to overfitting and are easier to generalize to unseen data. Also, such layers are great for transfer learning, *i.e.*, taking a model trained in some problem A and using it as initialization for training to a problem B.

The lower number of trainable parameters allows one to stack several convolutional layers, creating the so called *deep networks*, which makes CNNs extremely powerful. Such a strategy makes CNNs great feature extractors, where first layers learn to extract low-level features (such as horizontal or vertical edges, and low-frequency information), and each

consecutive layer is capable of learning to identify features with increasing complexity (from shapes to eyes, to faces, for instance).

Although CNNs had taken of only a few years ago (KRIZHEVSKY; SUTSKEVER; HINTON, 2012), there is a great amount of work proposing improvements and alternatives. Nowadays, there are several types of layers that can be combined in the network's architecture, such as pooling layers (KRIZHEVSKY; SUTSKEVER; HINTON, 2012), batch-normalization layers (IOFFE; SZEGEDY, 2015), dropout layers (SRIVASTAVA et al., 2014), dilated convolutional layers (YU; KOLTUN, 2015), locally-connected layers (TAIGMAN et al., 2014), and deconvolution layers (NOH; HONG; HAN, 2015). Although this dissertation only mentioned the ReLU activation function (NAIR; HINTON, 2010), several others can be used as well: sigmoid, tanh, PRelu (HE et al., 2015), LeakyReLU (MAAS; HANNUN; NG,), swish (EGER; YOUSSEF; GUREVYCH, 2018) and others. Similarly, several optimizers had been proposed: RMSprop (TIELEMAN; HINTON, 2012), Adagrad (DUCHI; HAZAN; SINGER, 2011), Adadelta (ZEILER, 2012), and Adam (KINGMA; BA, 2014). In addition, several architecture strategies had also become famous: use of Inception Modules (SZEGEDY; IOFFE; VANHOUCKE, 2016), Residual Networks (HE et al., 2016), 1x1 convolutions (LIN; CHEN; YAN, 2013), and Variational Autoencoders (KINGMA; WELLING, 2013). There is an ever-increasing list of alternatives for each building block (optimizer, activation functions, regularizers, etc.) of the deep-learning strategy, each one trying to address a problem in particular. Unfortunately, there is no best alternative, and the choice of what to use is more-than-often based on experimentation.

2.3 Generative Adversarial Networks

Generative Adversarial Networks (GANs) were first introduced by Goodfellow et al. (GOODFELLOW et al., 2014), but many variations were proposed in the following years. A GAN consists of two neural-networks competing against each other, making them appropriate for unsupervised training. Fig. 2.1 illustrates a simple GAN: it consists of a *generator*, focused on mapping information from a latent space (some noise in Fig. 2.1) to a specific domain (for instance, image of digits); and a *discriminator*, which focuses on discriminating real data from the ones synthesized by the *generator*. Fig. 2.1 shows an important particularity of GANs, while the *discriminator* aims at *maximizing the probability of assigning the correct label* for real and synthesized data, the *generator* aims

at *minimizing* it, learning to generate samples that have a low probability of being classified as fake. The traditional adversarial loss for the discriminator is given by:

$$\mathbb{E}_x[\log(D(x))] + \mathbb{E}_z[\log(1 - D(G(z)))], \tag{2.1}$$

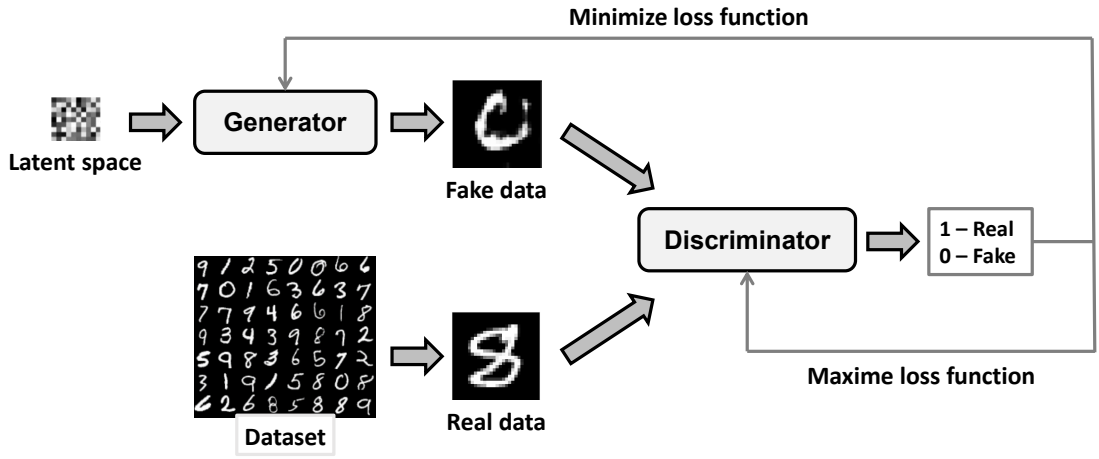
while for the generator is:

$$\mathbb{E}_z[\log(1 - D(G(z)))], \tag{2.2}$$

where $\mathbb{E}_x$ is the expected value over real-data samples x, $D(\cdot)$ is the discriminator's probability of the given data being real, $\mathbb{E}_z$ is the expected value over all random inputs z to the generator, and $G(z)$ is the generator's synthesized output given a noise z. These formulas derive from the cross-entropy between real and synthesized distributions, and while the discriminator tries to minimize Eq. (2.1), the generator tries to maximize Eq. (2.2). Some alternatives to the adversarial loss were proposed (ZHAO; MATHIEU; LECUN, 2016; BERTHELOT; SCHUMM; METZ, 2017; QI, 2020), the most popular being the Least-square loss (LSGAN) (MAO et al., 2017) and Wasserstein loss (WGAN) (ARJOVSKY; CHINTALA; BOTTOU, 2017).

The GAN architecture brings training benefits for both networks: the *discriminator* will aid the *generator* to synthesize more natural images (via backpropagation[1]), and the *generator* will output new (and unseen) images to feed the training of the *discriminator*.

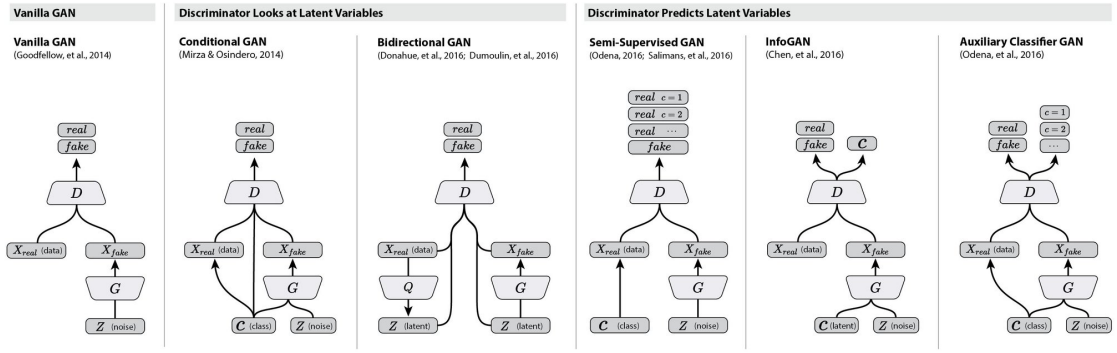
Figure 2.1: GAN concept. The *generator* tries to map information from a latent space (noise for instance) to a desired data distribution, while the *discriminator* is trained to differentiate between synthesized and real data. This competitive fashion brings several advantages during training, making both networks to improve together.



Although the main concept dates back to 2014, many variants were proposed

[1]Both networks are trained in an altogether architecture.

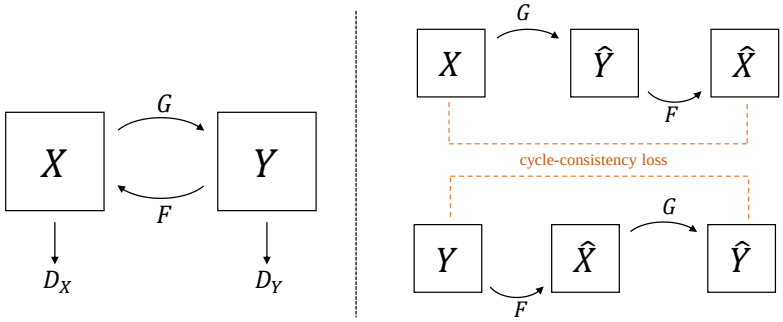
Figure 2.2: GAN variants. There are several works that changes the original architecture to improve GAN, normally by including known information (like class of images) along the latent space.



since then (MIRZA; OSINDERO, 2014; Odena, 2016; CHEN et al., 2016). Fig. 2.2 presents several of these variants, image credits to Chris Olah, who has also written a work about improving the training of GANs for image synthesis through the use of label conditioning (ODENA; OLAH; SHLENS, 2016). Due to their advantages, GANs have become a trend topic on deep learning, being used for image synthesis (DENTON et al., 2015), image super-resolution (LEDIG et al., 2016; ZHANG et al., 2019), image inpainting (YEH et al., 2016), semantic segmentation (LUC et al., 2016), and photo editing (HEINRICH, 2017).

Among the recent works, we want to highlight and discuss the work by Zhu et al. called *Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks* (ZHU et al., 2017). This work presents a novel architecture to perform unsupervised image-translation between two domains, introducing design choices that bring important benefits, making it possible for addressing the problem of noise in Chapter 4. In addition, we discuss the main reasons why GANs are hard to train.

Figure 2.3: Unpaired training and cycle-consistency loss. The architecture proposed by (ZHU et al., 2017) (left) let us to train the networks without the need of paired data, since the *discriminators* D_X and D_Y will tell us when a translated image belongs (or not) to the specific domain. (right) The cycle-consistency loss, enforcing G and F to be inverses of each other.



2.3.1 GANs for Image-to-Image Translation

Image-to-image translation seeks two mapping functions: $G : X \mapsto Y$, and $F : Y \mapsto X$. Consider a practical example of two image domains: X representing natural real images, and Y representing Monet paintings[2]. Normally, for a neural network to learn such functions, it would require paired training data (in order to compute the error between the mapped and real image), consisting of the same sample on both domains. Unfortunately, for many tasks[3], paired datasets are very limited, even unavailable. Zhu et al. propose using discriminators to enable the training of networks on unpaired data (ZHU et al., 2017). Such strategy was named as CycleGANs.

First, to allow the training using non-paired data, their paper makes use of two *discriminators*: D_X, which aims at telling us if a given image belongs to the X domain, and a D_Y, discriminating images from the Y domain. Fig. 2.3(left) illustrates how such process works: D_Y encourages G to translate from X to Y, and D_X enforces F to take an image from Y and outputs the corresponding image inside the image-distribution of domain X. Note that, in this case, G and F are the *generators*, but instead of mapping from a latent space, they are mapping between domains with different image distributions. This particular 'trick' lets us use images from domain X to train D_X, and same for Y and D_Y, avoiding the need of paired data.

The other contribution of Zhu et al.'s work is the additional loss term, called by them as the *cycle-consistency loss*. This term enforces that G and F should be inverses (conceptually), making both mappings to be bijections. Practically, by taking an image x from the X domain, translating using G, and translating it back using F, the final output should be similar to the initial image: $F(G(x)) \approx x$, and similarly $G(F(y)) \approx y$. This is a powerful regularizer, and we further discuss how to adapt it for our problem in Chapter 4.

2.3.2 Challenges in GAN Training

> *"It's also pretty similar to what Andrew [Andrew Ng] said about how supervised deep learning went from the lab to real world. The state of GANs today kind of reminds me of the state of supervised deep learning maybe like cerca 2012 that it used to really take a wizard to train a deep learning*

[2] Using the same example described in the former paper.
[3] Such as our noise-handling problem

> *system, and to some extent today GANs are still like that. Now deep learning is considered relatively reliable and it's because we found all these nice recipes like always using ReLUs, always using momentum, maybe having a few technologies that didn't radically change the paradigm but made it so much more reliable, like Adam and ResNets. I'm hoping that we get those kinds of reliability technologies that help us to apply GANs in lots of applications without needing a GAN wizard."*
>
> — Ian Goodfellow, 2020

As pointed by the creator of GANs, these networks are hard to train in practice, requiring several trial-and-error experiments before generating convincing results. In this Section, we discuss the main challenges associated with the training of GANs.

Training of GANs is known to be unstable. This is due to the fact that both generator and discriminator are trained simultaneously, meaning that an improvement in one of the networks often reduces the performance of the another. In a perfect scenario, both networks would converge to a Nash equilibrium, but this is quite difficult in GANs, "where the cost functions are non-convex, the parameters are continuous, and the parameter space is extremely high-dimensional" (SALIMANS et al., 2016). In addition, as both networks tackle different problems (one for generating samples of a given distribution and other for discriminating them), one network may start overperforming another during training, which often leads to non-convergence. Even when the training seems to converge, the losses suffer great oscillations, which are propagated when updating the parameters of the models.

Another well-known problem when training GANS is the *mode collapse*. This happens when the generator learns to map different inputs to the same or to a limited set of output(s). This problem happens quite often on vanilla GANs, and it is quite concerning because, besides limiting the outputs, once the generator collapses to a single/few point(s) in training, it cannot get out, being necessary to restart the training.

GAN's training also suffers from weak/diminished gradients. In this case, when the discriminator gets too successful (which often happens when the real and fake distributions are pretty far), the gradients passed to the generator are pretty small, and the generator learns practically nothing.

All these issues, combined with high sensibility to hyperparameters selection, make GANs training pretty hard and time-consuming. Many recent papers try to address these problems, either by proposing new loss functions (JOLICOEUR-MARTINEAU, 2018),

weights-update strategies (SALIMANS et al., 2016), proposing regularization schemes on gradient (KODALI et al., 2017), and others (SALIMANS et al., 2016; WIATRAK; ALBRECHT, 2019). Additionally, there are several practical tips that can be found online [4,5,6,7], ranging from experiments on which optimizer and activation functions work best, to normalization and regularization techniques, and multiscale training. Unfortunately, these are still open problems that hamper GAN training.

2.4 Summary

This Chapter provided a brief overview of the tools that will be used in Chapters 3 and 4. The basics of CNNs and GANs were presented. Further information on how we use them to address each problem can be found on the following Chapters. For instance, Chapter 3 shows how to model the acquisition process by using convolution and masks, which enabled the training of the CFA colors when optimizing the parameters of the full autoencoder. Chapter 4 presents loss terms that make CycleGAN suited for mapping between domains with different noise levels.

[4]https://machinelearningmastery.com/how-to-train-stable-generative-adversarial-networks/

[5]https://github.com/soumith/ganhack

[6]https://medium.com/intel-student-ambassadors/tips-on-training-your-gans-faster-and-achieve-better-results-9200354acaa5

[7]https://towardsdatascience.com/10-lessons-i-learned-training-generative-adversarial-networks-gans-for-a-year-c9071159628

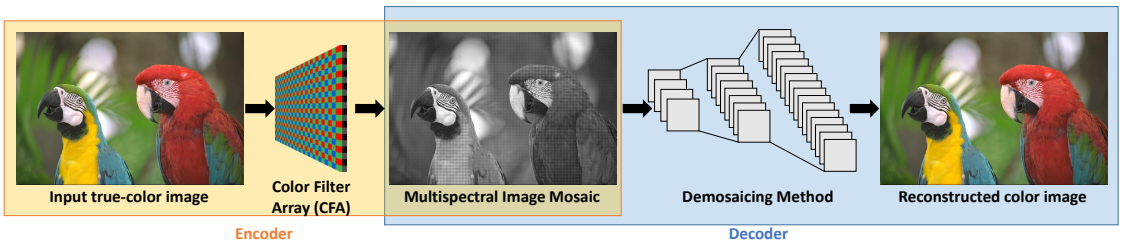
3 JOINT DESIGN OF CFA AND DEMOSAICING

"As soon as we realized that it was possible to capture light with our cameras, we wanted to harness all the colors associated with it." (ARCHAMBAULT, 2015a). Capturing color is one of the most fascinating things about photography. In Digital Photography, this is achieved through the use of Color filter arrays to filter the incoming light before it is captured by a monochromatic sensor. A demosaicing method is used to interpolate the missing color samples, generating a full RGB image. In this Chapter we present a Convolutional Neural Network architecture for performing joint design of color filter array (CFA) patterns and demosaicing. The patterns and algorithms produced by our method provide high-quality color reconstruction, surpassing existing techniques on standard demosaicing datasets, both for noise-free and noisy data. We show that our approach can also be used for simultaneous capture and reconstruction of color and infrared information. As image capture is the first step of the photography pipeline, any improvement at this stage should benefit the entire pipeline.

3.1 Introduction

Color filter arrays (CFAs) are a key component of digital imaging devices, allowing the capture of color pictures using a single monochromatic sensor. Superimposed on the sensor, a CFA selectively allows photons with certain wavelengths to reach the sensor, creating a *multispectral mosaic* (Fig. 3.1 (center)) defined by the properties of the filters in the CFA pattern. These filtered samples are then interpolated through a process known as *demosaicing* that reconstructs the resulting color image (Fig. 3.1 (right)). The Bayer

Figure 3.1: Color image capture and reconstruction on single-CCD (charge-coupled device) cameras. A color filter array superimposed on a monochromatic sensor selectively allows photons with certain wavelengths to reach the sensor, creating a *mosaic* defined by the filters in the CFA pattern. A color image is then reconstructed from the filtered samples. We model this process as an autoencoder: the CFA projection encodes color information onto the monochromatic sensor, which is later decoded by the color-reconstruction method.



pattern (BAYER, 1976) is the most popular CFA pattern for digital cameras and many demosaicing algorithms have been proposed to improve the quality of the reconstructed images sampled with it (MAIRAL et al., 2009; GETREUER, 2011b; HEINZE; LÖWIS; POLZE, 2012; WANG, 2014). Nonetheless, several techniques have focused on designing new CFA patterns, either by looking for patterns whose representation in the frequency domain have little overlap between luma and chroma information (HAO et al., 2011; CONDAT, 2011; BAI et al., 2016) (designed specifically for use with frequency-selection demosaicing algorithms (ALLEYSSON; SUSSTRUNK; HERAULT, 2005; DUBOIS, 2005)), or by designing CFAs for sparse-representation-based demosaicings (LI et al., 2017; MAIRAL et al., 2009). Essentially, previous solutions have been restricted to either design demosaicing algorithms for a given CFA pattern (*e.g.*, the Bayer pattern), or to design CFA patterns that work with a specific demosaicing algorithm (*e.g.*, (ALLEYSSON; SUSSTRUNK; HERAULT, 2005; DUBOIS, 2005; MAIRAL et al., 2009)). Since sampling and reconstruction are tightly-coupled processes, by predefining the CFA pattern or the reconstruction algorithm, one severely constrains the search space and, therefore, the ability to obtain an optimal solution.

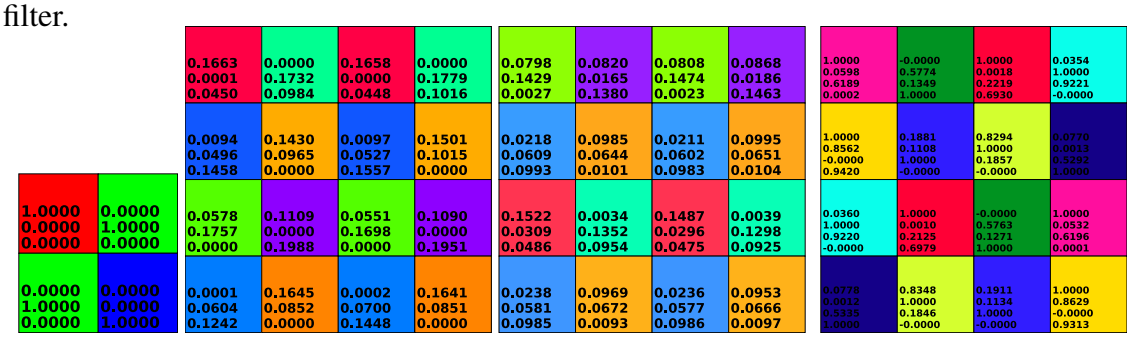
We simultaneously address the problem of CFA pattern design and demosaicing. For this, we use an end-to-end autoencoder that mimics the process of image acquisition (Fig. 3.1). An autoencoder is a learning model that tries to reconstruct the original information after projecting it into a lower-dimensional space (KINGMA; WELLING, 2013). *Our autoencoder jointly obtains a CFA pattern and a demosaicing algorithm.* The encoding step projects the light filtered by the CFA onto a monochromatic sensor, generating a multispectral image mosaic. The decoding step recovers a color image from the mosaic (Fig. 3.1). By training encoder and decoder simultaneously, for any given CFA dimensions our approach automatically finds the CFA pattern and corresponding demosaicing algorithm that minimizes color-reconstruction error. Our technique produces high-quality color reconstructions, outperforming the state-of-the-art techniques in all standard demosaicing datasets both for noise-free and noisy data.

Fig. 3.2 shows examples of color filters obtained with our method for different cases, such as noise-free and noisy image-reconstruction, and for acquiring nearinfra-red (NIR) along with RGB information. Such patterns are tiled side-by-side to produce a CFA with the desired number of pixels.

The **contributions** of our work include:

- *A method for the joint design of CFA pattern and demosaicing that minimizes color-*

Figure 3.2: CFA patterns. From left to right: Bayer pattern, our 4×4 CFA for noise-free data, our 4×4 CFA for noisy data, and our 4×4 CFA for RGB-NIR data. The numbers inside each cell are (from top to bottom) the R, G, and B coefficients defining the color filter.

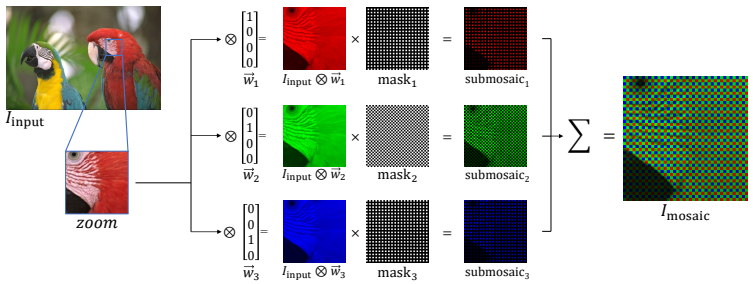


Bayer Our 4x4 noise-free Our 4x4 noise Our 4x4 NIR

reconstruction errors (Section 3.3). Our model is the first to optimize CFA colors over the entire RGB color space, while jointly optimizing demosaicing. The results produced by our system outperform existing solutions in terms of PSNR for both noise-free and noisy data (Section 3.4);

- *An autoencoder architecture that models the color-image capture process on single monochromatic sensors*. Our architecture achieves fast training convergence on image patches, and works with CFAs of different sizes, including existing ones (Section 3.3.1);
- *A joint design of CFA and demosaicing for capturing NIR along with visible light (Section 3.4.3). Our method is the first one to consider the use of color filters that capture NIR and visible information jointly.*

Figure 3.3: Example of an encoder mimicking the Bayer pattern. Each color filter is represented by weights $\vec{w}_i = [w_{ir}, w_{ig}, w_{ib}, w_{it}]$ and a mask (mask$_i$), generating a submosaic. For the Bayer pattern, the bias term $w_{it} = 0$ for all three color filters w_i. The $I_{\text{input}} \otimes \vec{w}_i$, submosaic$_i$, and I_{mosaic} images are colored just for illustration, as they are single-channel images.



3.2 Related work

3.2.1 CFA Design

Following the work of Bayer (BAYER, 1976), several color filter arrays have been proposed over the years. Lukac and Plataniotis (LUKAC; PLATANIOTIS, 2005) analysed the performance of ten RGB CFAs. More recently, researchers have proposed a variety of new CFA design strategies. Hirakawa and Wolfe (HIRAKAWA; WOLFE, 2008) introduced the idea of designing CFAs directly in the Fourier domain by optimizing the carrier waves. Lu and Vetterli (LU; VETTERLI, 2009) presented a CFA pattern that minimizes the reconstruction error of a linear-minimum-mean-square-error demosaicing method. Condat presented three different designs: RGB CFAs arranged in a non-periodic pattern (CONDAT, 2009), randomly generated patterns with specific blue-noise characteristics (CONDAT, 2010), and a 2×3 CFA pattern which has enhanced sensitivity properties and robustness to noise (CONDAT, 2011). Hao et al. (HAO et al., 2011) introduced CFAs based on the frequency structure, which are manually initialized according to some guidelines, and then optimized using a geometric method. Bai et al. (BAI et al., 2016) proposed a method for CFA design in the frequency domain. Given the CFA pattern size, their method suggests frequency-structure candidates and then optimizes the parameters by maximizing numeric stability of color transformations. Chakrabarti et al. (CHAKRABARTI; FREEMAN; ZICKLER, 2014) proposed a predominantly panchromatic CFA that samples color at sparse sets of locations, which are then propagated throughout the image guided by the luminance channel. In a subsequent work, Chakrabarti (CHAKRABARTI, 2016) used a convolutional neural network (CNN) architecture to design a CFA from a set of predefined colors, while training the demosaicing method concurrently. Li et al. (LI et al., 2017) proposed a CFA design optimized for sparse-representation-based demosaicing (MAIRAL et al., 2009) and showed how to minimize mutual coherence of CFAs with constraints for physical realizability.

The work closest related to ours is the one proposed by Chakrabarti (CHAKRABARTI, 2016), which is the only previous technique to jointly optimize CFA design and demosaicing. However, our method differs from Chakrabarti's in a crucial aspect: while Chakrabarti treats CFA design as a decision problem, we model it as a regression problem, thus optimizing CFA colors over the entire RGB color space. As such, our approach explores a much larger parameter space, resulting in better color-reconstructed images, with PSNR gains

ranging from 5 up to 30 dB, for images in the standard demosaicing datasets (Section 3.4).

3.2.2 Demosaicing

Demosaicing is a well-studied problem, with many surveys on existing methods (LI; GUNTURK; ZHANG, 2008; MENON; CALVAGNO, 2011; KAUR; BANGA, 2015). Demosaicing algorithms have been proposed for the frequency domain (ALLEYSSON; SUSSTRUNK; HERAULT, 2005; DUBOIS, 2005; LIAN et al., 2007), and based on hard-coded heuristics for interpolation (JAISWAL et al., 2014; LI, 2005; ZHANG; WU, 2005), self-similarities (ZHANG et al., 2011; BUADES et al., 2009), optimization schemes (CONDAT; MOSADDEGH, 2012; HEIDE et al., 2014; KLATZER et al., 2016), and compressive sensing (MAIRAL et al., 2009; MOGHADAM et al., 2013; DAVE; VADATHYA; MITRA, 2016). Next, we discuss demosaicing strategies based on neural networks.

Kappa and Hel-Or (KAPAH; HEL-OR, 2000) and Go et al. (GO; SOHN; LEE, 2000) were the first to use neural networks for demosaicing. Long and Huang (LONG; HUANG, 2006) later proposed an adaptive scheme to improve Go et al.'s method. Heinze et al. (HEINZE; LÖWIS; POLZE, 2012) proposed multi-frame demosaicing using a neural network for estimating the pixel color based on its surroundings. Wang (WANG, 2014) used 4×4 patches to train a multilayer neural network while minimizing a suitable objective function. Gharbi et al. constructed a dataset with hard cases, which were used to train a CNN for joint demosaicing and denoising (GHARBI et al., 2016). All these methods were specifically designed to reconstruct Bayer filtered images.

3.2.3 Autoencoders and Residual CNNs

An autoencoder is a variation of a neural network that tries to learn a representation of its input in a lower-dimensional space and then reproduce the original information from such a sparse representation. Introduced in the CNN literature as a data-driven compression method (KINGMA; WELLING, 2013), the autoencoder concept has already been used for image denoising (VINCENT et al., 2010), data visualization (MAATEN; HINTON, 2008b), superresolution (ZENG et al., 2015), and to learn priors used for image reconstruction (CHOI et al., 2017).

Our architecture is based on Residual Nets (HE et al., 2016; GROSS; WILBER,

2016; ZAGORUYKO; KOMODAKIS, 2016), and exploits many improvements such as Batch Normalization (IOFFE; SZEGEDY, 2015), Adam optimizer (KINGMA; BA, 2014), and small-size filters (3x3 and 1x1 kernels) (LIN; CHEN; YAN, 2013; SZEGEDY et al., 2015a). Residual architectures (with skip connections) have been used in several image-processing tasks such as denoising (ZHANG et al., 2017), superresolution (TIMOFTE et al., 2017), and others (JIN et al., 2017). Our architecture similarly makes use of skip connections, but instead of merging branches by summing feature maps (as in the original ResNet (HE et al., 2016)), we do so by concatenating the skipped and original feature maps. Such a choice has already been used in recent works (SZEGEDY et al., 2015a; SZEGEDY et al., 2015b).

3.3 Joint Design of CFAs and Demosaicing

Our joint design of CFA pattern and demosaicing is expressed as the training of an autoencoder. Given a set of training images, the encoding procedure consists of projecting the corresponding input RGB information on the (trainable) color filter array pattern (Section 3.3.1). Such a projection generates a single-channel multispectral image mosaic (Fig. 3.1 (center)), which serves as input for the decoding (color reconstruction) step.

The training process minimizes a loss function defined as the mean squared error (MSE) between the provided ground truth and the reconstructed color images. The trainable parameters are the colors of the CFA pattern (encoder) and the CNN weights for demosaicing (decoder). Fig. 3.1 illustrates the concept. When designing the encoder, we are restricted by physical limitations imposed by the construction of actual CFAs, which precludes the use of non-linear activation functions and stacked layers. The decoder, however, may use as many layers as desired, as it is computed after the sampling process. The architecture of our network is detailed in Section 3.3.1.

Using only convolutional, ReLU and batch-normalization layers (IOFFE; SZEGEDY, 2015), our CNN architecture supports images of different sizes. By avoiding the use of fully-connected layers, the network can be trained using small image patches (with 128×128 pixels) and still reconstruct images of variable resolution (without the need of breaking the image into patches). This provides our CNN great flexibility and significantly reduces training time.

3.3.1 Our Autoencoder Architecture

Encoding: Our architecture optimizes colors over the entire RGB color-space. Each component (*i.e.*, color filter) of the CFA pattern is represented by a four-dimensional vector $\vec{w} = [w_r, w_g, w_b, w_t]$. The weights $[w_r, w_g, w_b] \in \mathbb{R}^3_{\geq 0}$ are RGB coefficients that represent the actual color filter (Fig. 3.2), and $w_t \in \mathbb{R}$ is a bias term. Appendix A shows the full vectors $\vec{w}_i$ associated with our CFAs. Given a pixel from an input image with RGB coefficients $\vec{p} = [p_r, p_g, p_b] \in \mathbb{R}^3_{\geq 0}$, we model the encoding of $\vec{p}$ by the color filter $\vec{w}$ using an affine functional $\otimes$ defined as:

$$\vec{p} \otimes \vec{w} := p_r w_r + p_g w_g + p_b w_b + w_t. \tag{3.1}$$

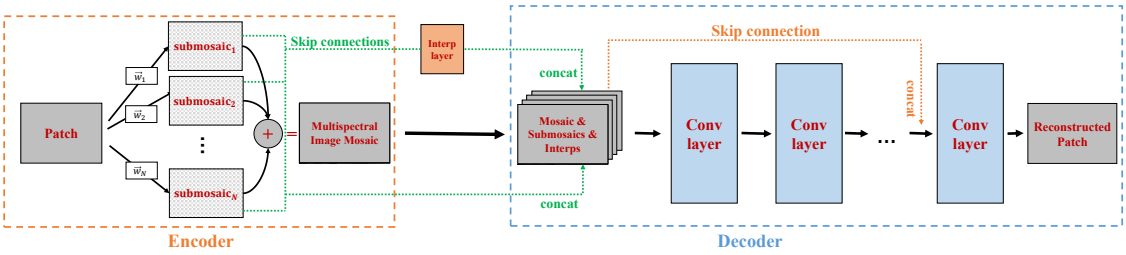
Eq. (3.1) is modeled as a convolution of the input image with a $1 \times 1 \times 3$ kernel plus a bias term. Note that since all coefficients of $\vec{w}$ are trainable, we must enforce non-negative weights (*i.e.*, $w_r, w_g, w_b \geq 0$) to guarantee that the color filter is physically realizable. For this, after each update, all negative weights are clamped to zero. We do not constrain the value of the bias parameter w_t since it is added after image capture. Similarly, we do not constrain the maximum value of the weights w_r, w_g, and w_b to allow for a wider range of admissible parameters during training (any constant rescaling may be performed after image capture as well). While such restrictions could be included on the architecture, having them could hamper the training convergence (by reducing the optimizer's search-space). In practice, however, the weights produced by our method seem to always fall in the $[0, 1]$ interval (see Fig. 3.2).

Each color filter w_i has an associated binary mask (mask_i) that specifies the pixels projected through it. Thus, simulating a CFA containing N distinct color filters requires the use of N distinct functionals and N disjoint binary masks. The projected-submosaic generated by the i-th color filter is then defined as

$$\mathrm{submosaic}_i = (I_{\mathrm{input}} \otimes \vec{w}_i) \times \mathrm{mask}_i, \tag{3.2}$$

where I_{input} is an RGB input image, $\vec{w}_i = [w_{ir}, w_{ig}, w_{ib}, w_{it}]$, mask_i is the binary mask corresponding to the i-th CFA color, and both the functional $\otimes$ and product $\times$ are evaluated pixelwise. Thus, the multispectral mosaic generated by a CFA with N color filters is define

Figure 3.4: Our autoencoder architecture. The encoding step projects the colored image patches through the trainable CFA, generating a multispectral image mosaic. Skip connections (green arrows) contribute submosaics consisting of the separate color channels as well as per-channel interpolated images, which are stacked with the multispectral image mosaic forming a deeper representation (total of $2N + 1$ channels for N color filters).The decoding component is based on residual blocks. This autoencoder can produce CFA patterns of different sizes and works with networks of distinct depths. The beginning-to-end decoder skip connection (orange arrow) improves color reconstruction and training convergence. Parameters details are given in Section 3.3.1.



as

$$I_{\text{mosaic}} = \sum_{i=1}^{N} \text{submosaic}_i. \tag{3.3}$$

Note that one can define CFAs of arbitrary sizes while enforcing how the pattern should repeat by using the disjoint masks. The CFA is a periodic structure, with the pattern corresponding to one period. Given an $M \times N$ CFA pattern, this will result in MN (M times N) distinct $M \times N$ binary masks for the CFA pattern (*i.e.*, one binary mask for each CFA element). Each such binary mask has a single non-zero element (with value 1), at the position corresponding to the given CFA element. Thus, in a $M \times N$ CFA, the CFA element at position (i, j), $1 \leq i \leq M$, $1 \leq j \leq N$, has a corresponding binary mask containing zeros everywhere, except at mask position (i, j), which contains the value 1. Similar to a complete CFA, the actual binary masks cover the entire image, being obtained by tiling the corresponding CFA-element binary masks.

During training, the weights $\vec{w}_i$ are optimized to minimize color-reconstruction error, while all masks remain fixed. Fig. 3.3 shows an example of an encoder mimicking the Bayer pattern, for which the bias $w_{it} = 0$ for all i. Note that the mask corresponding to the green component is twice as dense as the others.

Decoding: The decoding step receives as input the multispectral image mosaic I_{mosaic} produced by the encoder (Eq. (3.3)) and tries to reconstruct the original RGB image I_{input}. The decoder architecture consists of stacked convolutional layers, each one followed by a batch-normalization layer (IOFFE; SZEGEDY, 2015) and by the ReLU activation function. All convolutional layers use 3×3 kernels, using padding to ensure the same xy-dimensions

for all receptive fields.

In addition to the monochromatic image mosaic, we provide the following additional inputs to the decoder: the submosaics of each color filter w_i, and linearly-interpolated versions of each submosaic. Although the submosaics themselves do not add new information to the decoding sub-network, they save the effort of learning how to separate individual channels, thus reducing training time. The linearly-interpolated versions of the submosaic, in turn, result in better results and faster convergence, since such an initial guess for the color-reconstructed image is much closer to the target image.

In our CNN architecture, linear interpolation is achieved by convolving the submosaics with a tent kernel. Since this kernel is separable, the 2D interpolation kernel $k_{n,m}$ is defined as the outer product $k_n k_m^T$, where

$$k_c = \left[{}^1\!/_c \quad {}^2\!/_c \quad \ldots \quad {}^{(c-1)}\!/_c \quad 1 \quad {}^{(c-1)}\!/_c \quad \ldots \quad {}^2\!/_c \quad {}^1\!/_c \right]^T .$$

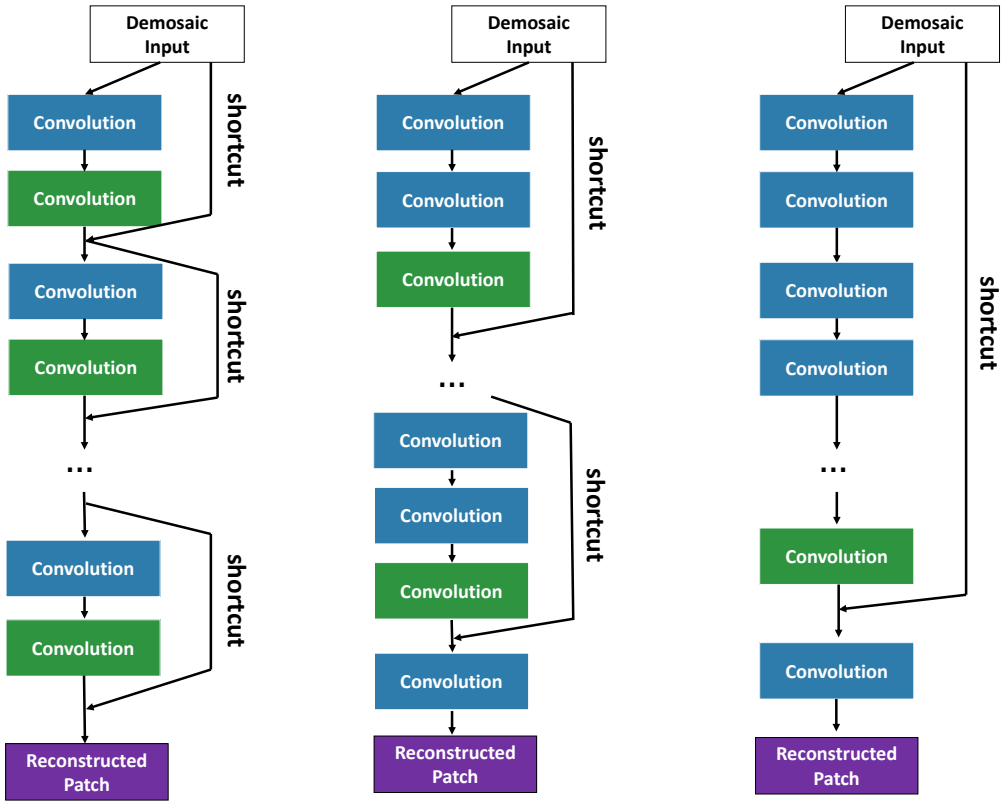
The vector $k_c \in \mathbb{R}^{2c-1}$ defines a 1D tent kernel that interpolates a mosaic generated by a 1D CFA pattern containing c colors. For example, $k_2 = [{}^1\!/_2 \; 1 \; {}^1\!/_2]$ and, for a 4×4 CFA pattern with 16 distinct color filters, the required tent kernel is

$$k_{4,4} = \frac{1}{16} \begin{bmatrix} 1 & 2 & 3 & 4 & 3 & 2 & 1 \\ 2 & 4 & 6 & 8 & 6 & 4 & 2 \\ 3 & 6 & 9 & 12 & 9 & 6 & 3 \\ 4 & 8 & 12 & 16 & 12 & 8 & 4 \\ 3 & 6 & 9 & 12 & 9 & 6 & 3 \\ 2 & 4 & 6 & 8 & 6 & 4 & 2 \\ 1 & 2 & 3 & 4 & 3 & 2 & 1 \end{bmatrix} .$$

This interpolation is performed by a standard convolutional layer with fixed kernel weights. Although these weights could also be learned, we have found that fixing them (*i.e.*, not updating them during training) produces better results.

In addition, we have experimented three different disposition of skip-connections (illustrated in Fig. 3.5): one that uses a skip-connections every two convolutional layers (Architecture A), one that uses a skip-connection every three convolutional layers (Architecture B), and another that uses a single skip-connection along the entire decoder (Architecture C). During our experiments, Architecture C - using a single skip-connection - has achieved better performance, both for converging faster as for performing better

Figure 3.5: Three architectures evaluated for demosaicing. All convolutional layers are followed by a BatchNormalization and ReLU, except the ones preceding a merge with a skip-connection (shown in green).



reconstructions. All the results presented in Section 3.4 used this architecture.

The complete architecture of our autoencoder is depicted in Fig. 3.4, while Section 3.4 provides implementation details. Besides the skip connection from the beginning of the decoder to its end (for improving color reconstruction and training convergence) indicated by the dotted orange arrow, we also use skip connections from each submosaic and corresponding linearly-interpolated versions to the decoder's input. Such connections are indicated by the dotted green arrows and proved to speed up the training, providing a path for gradient backpropagation.

Training: Given the number of color filters for the CFA and the topology of the decoding network (number of convolutional layers with their inner parameters – Fig. 3.4), the autoencoder is trained by iteratively feeding patches to the network. Such patches are used to update the colors of the CFA and weights of the demosaicing method, trying to minimize the MSE between the input color patches and their reconstructions.

3.4 Demosaicing Results and Evaluation

Our particular instantiation of the autoencoder architecture described in Section 3.3.1 and illustrated in Fig. 3.4 includes a decoder consisting of a stack of 12 convolutional layers. The number of (3×3) kernels used in each of these 12 layers are [64, 64, 64, 64, 64, 64, 128, 128, 128, 128, 128, 128], respectively. This setup presents a good trade-off between network expressiveness and training time. We show that the quality of our reconstructions surpasses the ones generated by existing methods (CHAKRABARTI, 2016; MAIRAL et al., 2009; GHARBI et al., 2016).

We implemented our network using Keras (CHOLLET, 2015), running on top of Theano (TEAM, 2016), using MSE as the loss function, and the Adam optimizer (KINGMA; BA, 2014) ($lr = \alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$), with batch size of 32. Training was performed using two datasets provided by Gharbi et al. (GHARBI et al., 2016): *vdp* and *moiré*. We used the same images as Gharbi et al. for training, consisting of 2,590,186 128×128-pixel patches. In addition, we used horizontal and vertical flips, as well as random 90° rotations, for data augmentation. Our model for reconstructing noise-free images (as well as our demosaicing method for the Bayer pattern) was trained for 3 epochs, which corresponds to approximately 5 days of training time on a GeForce GTX TITAN X GPU. Our model for reconstructing noisy data was trained for 6 epochs.

3.4.1 Comparisons to Other Approaches

Table 3.1 compares the PSNR of the results obtained with our 4×4 noise-free CFA (Fig. 3.2) with the ones produced by the most successful demosaicing techniques (MAIRAL et al., 2009; GETREUER, 2011b; BAI et al., 2016; HAO et al., 2011; CONDAT, 2011; WANG, 2014; CHAKRABARTI, 2016; GHARBI et al., 2016). For all comparisons, we have used either source or executable code provided by the authors. All images were saved to disk to guarantee similar color quantization and later compared based on PSNR (error averaged over pixels and color channels before computing the logarithm). In addition, all measurements were performed on full-resolution images (borders included). We did not use any of the test images in our training phase. Table 3.1 shows the average PSNR for the traditional Kodak (FRANZEN, 1999) and McMaster (ZHANG XIAOLIN WU, 2011) datasets, as well as for the two datasets of Gharbi et al. (*vdp* and *moiré*) (GHARBI et al., 2016). The techniques on the top portion of Table 3.1 perform demosaicing for the Bayer

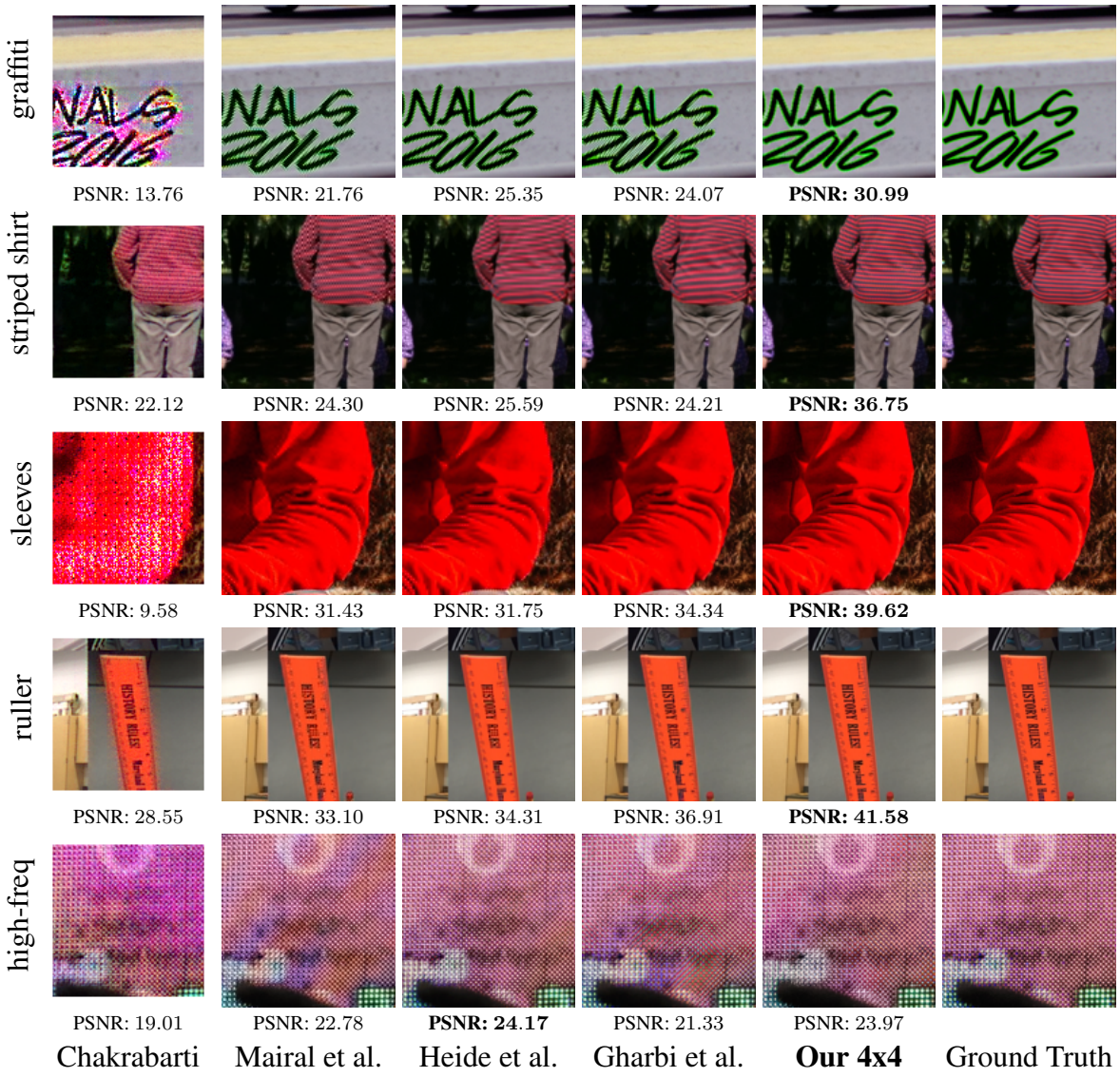
Table 3.1: Comparison of our 4×4 noise-free CFA and demosaicing technique against existing methods. The numbers show the average PSNR values of reconstructions for four datasets. All results were generated using code provided by the authors, except the ones marked with †, whose numbers were taken from the corresponding publications. Our 4×4 noise-free CFA and demosaicing outperform all other techniques in all four datasets (best results in bold). Our demosaicing network for the Bayer pattern also outperforms all previous techniques for the Kodak, McMaster, and vdp datasets, and got very close to Gharbi et al.'s in the moiré dataset.

Demosaic (Bayer CFA)	**Kodak**	**McMaster**	**vdp**	**moiré**
Bilinear	29.51	32.32	24.97	27.39
(ZHANG et al., 2011)	35.66	29.87	24.85	28.04
(GETREUER, 2011a)]	35.98	35.87	29.88	31.70
(BUADES et al., 2009)	36.62	35.24	29.34	31.30
(LU; KARZAND; VETTERLI, 2010)	37.17	32.22	27.67	28.70
(CONDAT; MOSADDEGH, 2012)	38.51	33.29	29.03	30.96
(JAISWAL et al., 2014)	38.71	36.84	30.27	31.75
(HEIDE et al., 2014)	38.83	38.30	30.93	34.61
(KIKU et al., 2016)	38.84	36.86	30.52	31.90
(JEON; DUBOIS, 2013)	40.03	33.78	29.34	31.33
(GETREUER, 2011b)	40.13	34.17	30.06	32.25
(MAIRAL et al., 2009)	41.23	36.13	30.94	33.16
(GHARBI et al., 2016)	41.79	39.14	33.96	36.64
Our 2x2 Bayer	41.86	39.51	34.28	36.33
Demosaic (non-Bayer CFA)	**Kodak**	**McMaster**	**vdp**	**moiré**
(CHAKRABARTI, 2016)	31.52	28.05	23.97	23.97
(CHAKRABARTI; FREEMAN; ZICKLER, 2014)	33.51	30.94	25.91	28.77
(CONDAT, 2011)	38.10	32.90	28.65	30.45
(HAO et al., 2011)	39.42†	—	—	—
(BAI et al., 2016)	40.24†	—	—	—
(HIRAKAWA; WOLFE, 2008)	40.36†	—	—	—
(LI et al., 2017)	41.59†	—	—	—
Our 4x4 noise-free	**43.13**	**40.18**	**35.17**	**37.70**

pattern, while the ones on the bottom portion perform demosaicing for non-Bayer CFAs. Our 4×4 noise-free CFA and demosaicing solution (last row of Table 3.1) surpasses all existing methods in all four datasets (even for Kodak and McMaster, from which no images were used for training). We also trained our architecture using the Bayer pattern, *i.e.*, only optimizing the decoder (Fig. 3.3). Our demosaicing network for the Bayer pattern also outperforms all previous techniques for the Kodak, McMaster, and *vdp* datasets, and got very close to Gharbi et al.'s (GHARBI et al., 2016) in the *moiré* dataset. These results clearly demonstrate the effectiveness of our autoencoder architecture and the advantage of jointly optimizing CFA design and demosaicing.

Fig. 3.6 compares the reconstruction quality of our 4×4 noise-free model with the state-of-the-art demosaicing techniques (CHAKRABARTI, 2016; MAIRAL et al.,

Figure 3.6: Comparison of reconstruction quality of our 4×4 noise-free method and the state-of-the-art techniques (best PSNR values shown in bold). Patches from Gharbi et al.'s datasets (GHARBI et al., 2016). Better visualized in the digital version.



2009; HEIDE et al., 2014; GHARBI et al., 2016). For such comparison, we have used code provided by the authors for their noise-free trained models. Chakrabarti's method (CHAKRABARTI, 2016) does not reconstruct the full patch, so we measure the PSNR only for the reconstructed area. The examples in Fig. 3.6 are from Gharbi et al.'s datasets. Note how our method better handles high-frequency information, being less susceptible to aliasing artifacts than other techniques (see the stripped shirt example in Fig. 3.6). Additional examples can be found in the supplementary materials.

Chakrabarti (CHAKRABARTI, 2016) trained and tested his model using the dataset of Shi and Funt (SHI; FUNT, 2010). We also tested our 4×4 noise-free CFA on the same test images. Chakrabarti (CHAKRABARTI, 2016) achieves an average PSNR of 41.50,

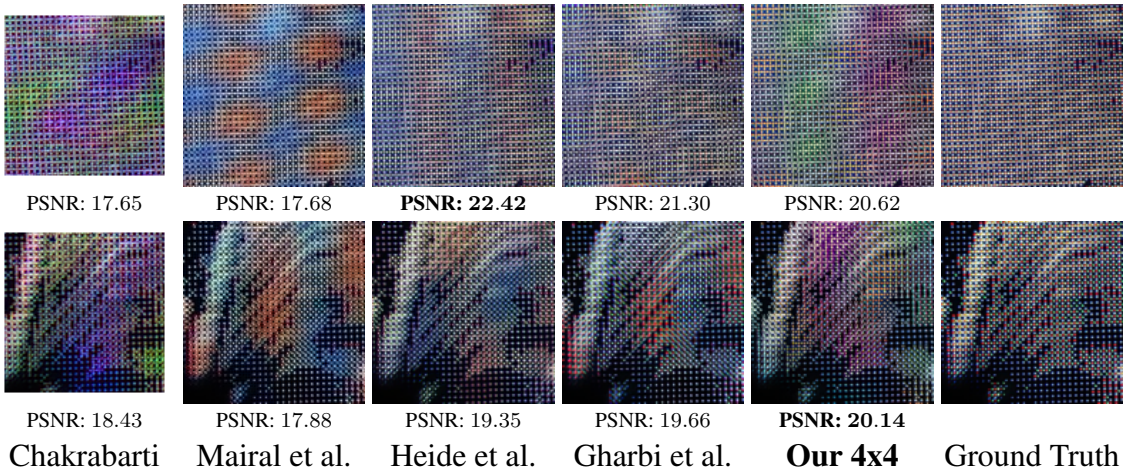
Table 3.2: Average running times of state-of-the-art techniques for reconstructing images from the Kodak dataset.

Running Times (seconds)	
Chakrabarti (GPU) (CHAKRABARTI, 2016)	0.09
Gharbi et al. (GPU) (GHARBI et al., 2016)	0.16
Our 4x4 (GPU)	0.34
Mairal et al. (CPU) (MAIRAL et al., 2009)	565.23
Heide et al. (CPU) (HEIDE et al., 2014)	652.63

while our model achieves 48.96, a significant improvement in reconstruction quality, even though no images from Shi and Funt's dataset were used for training our CFA.

Table 3.2 shows the average running times of the state-of-the-art techniques for reconstructing images from the Kodak dataset (resolution of 768×512). All measurements were made on an Intel Core i7-2660 CPU and GeForce GTX TITAN X GPU. Our technique is slightly slower than other GPU-based implementations, but still much faster than methods based on compressive sensing (MAIRAL et al., 2009) or optimization schemes (HEIDE et al., 2014).

Figure 3.7: Challenging cases: images with extreme high-frequencies are challenging for all demosaicing methods (CHAKRABARTI, 2016; MAIRAL et al., 2009; HEIDE et al., 2014; GHARBI et al., 2016), whose results exhibit aliasing artifacts. These patches are from the moiré dataset (GHARBI et al., 2016).



3.4.2 Reconstruction in the presence of noise

The idea of jointly performing demosaicing and denoising has been explored by many works (CONDAT, 2011; CONDAT; MOSADDEGH, 2012; HEINZE; LÖWIS; POLZE, 2012; KLATZER et al., 2016; GHARBI et al., 2016; CHAKRABARTI, 2016).

Enhancing our autoencoder with denoising capabilities only requires feeding the network with patches corrupted by (artificial) noise during the training phase, while comparing the network's output to the noise-free images. To demonstrate the flexibility of our architecture, we have trained the same network structure from scratch, using the same datasets used for training our 4×4 noise-free CFA. This time, however, each input patch was corrupted by additive Gaussian noise. Although in linear space camera noise should be modeled as a combination of Poissonian and Gaussian noise (FOI et al., 2008), according to Jeon and Dubois (JEON; DUBOIS, 2013), for white-balanced, gamma-corrected images (such as the case of the vdp and moiré datasets (GHARBI et al., 2016) used for training) one can model noise as signal-independent white Gaussian noise. By corrupting the images with Gaussian noise with a varying standard deviation randomly picked from the set $\{0, 4, 8, 12, 16, 20\}$, we avoid the need of specialized networks for each noise level (such as in Chakrabarti (CHAKRABARTI, 2016)), training a single model that handles a large range of noise variance.

Table 3.3 compares the PSNR for our 4×4 CFA for noisy data (Fig. 3.2 (right)) against existing techniques. The quality of our reconstructions surpasseses previous approaches in all datasets, for all noise levels. Moreover, unlike Gharbi et al.'s approach (GHARBI et al., 2016), ours does not require an estimate of the noise level, and thus the quality of our denoising results are not dependent on the accuracy of any noise estimation step.

Fig. 3.8 compares our results to the the state-of-the-art techniques. Note that our method performs an optimization that jointly improves CFA design, demosaicing, and denoising. As a result, it can reduce noise without oversmoothing the images, generating higher-quality reconstructions. Our CFA pattern for noisy datasets can be seen in Fig. 3.2 (right). Additional examples can be found in the supplementary materials.

3.4.3 Capturing RGB and NIR

Near-infrared (NIR) images have recently been exploited for many applications, such as dark flash photography (KRISHNAN; FERGUS, 2009) and denoising low-light photographs (GASTAL; OLIVEIRA, 2012; Wang et al., 2019), among others (FENG et al., 2013; HONDA; TIMOFTE; GOOL, 2015; KUMAR; NONGMEIKAPAM; SINGH, 2019). To take advantage of both visible and NIR information, we must capture simultaneously NIR and color images for each scene. Recently, authors have proposed different CFAs to

Figure 3.8: Comparison of the reconstructions obtained by our method and state-of-the-art techniques (CHAKRABARTI, 2016; CONDAT; MOSADDEGH, 2012; GHARBI et al., 2016) that jointly perform denoising and demosaicing (best PSNR in bold). The input images were corrupted with Gaussian noise (left column), whose level is indicated by the standard deviation σ (in RGB $[0, 255]$ units). Images from the Kodak dataset. Better visualized in the digital version.

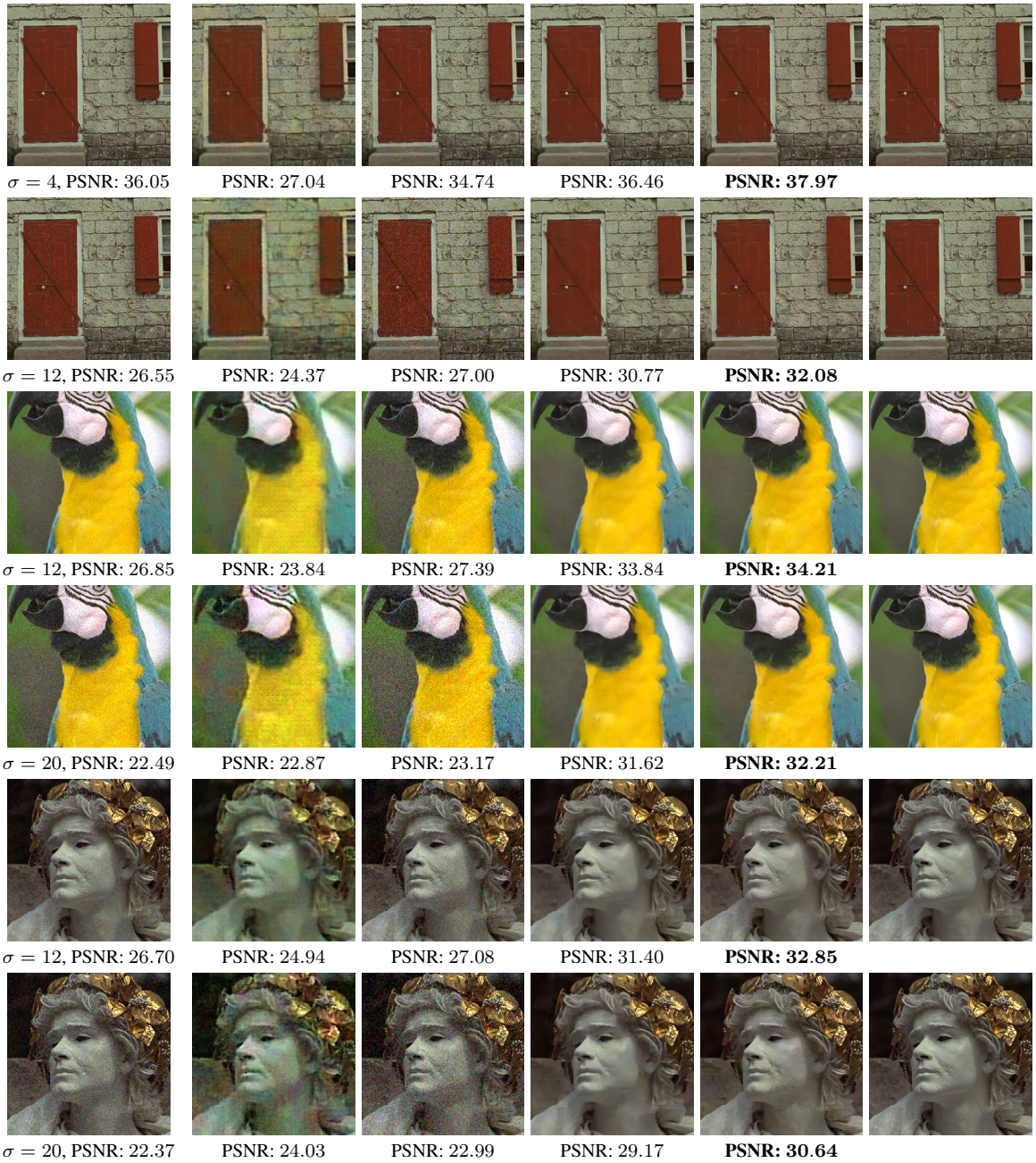


Table 3.3: Comparison of our model with existing methods for joint denoise and demosaic. The numbers show the average PSNR values of reconstructions for four datasets corrupted by noise of different intensities. Our model outperforms all other techniques in all four datasets and for all noise intensities.

Noise $\sigma = 4$	**Kodak**	**McMaster**	**vdp**	**moiré**
(CHAKRABARTI, 2016)	28.59	26.32	21.96	21.72
(CHAKRABARTI; FREEMAN; ZICKLER, 2014)	30.70	28.34	25.34	27.71
(CONDAT, 2011)	34.15	31.19	27.86	29.30
(CONDAT; MOSADDEGH, 2012)	34.43	31.53	28.19	29.69
(GHARBI et al., 2016)	36.90	36.02	31.61	33.31
Our 4x4 noise	**38.01**	**36.59**	**32.83**	**34.54**
Noise $\sigma = 8$	**Kodak**	**McMaster**	**vdp**	**moiré**
(CHAKRABARTI, 2016)	26.63	25.16	20.79	20.52
(CHAKRABARTI; FREEMAN; ZICKLER, 2014)	27.26	25.56	23.75	25.39
(CONDAT, 2011)	29.83	28.50	26.26	27.20
(CONDAT; MOSADDEGH, 2012)	30.14	28.84	26.56	27.52
(GHARBI et al., 2016)	34.19	33.97	29.87	31.34
Our 4x4 noise	**35.08**	**34.39**	**31.16**	**32.50**
Noise $\sigma = 12$	**Kodak**	**McMaster**	**vdp**	**moiré**
(CHAKRABARTI, 2016)	25.59	24.32	20.22	20.04
(CHAKRABARTI; FREEMAN; ZICKLER, 2014)	24.62	23.41	22.20	23.34
(CONDAT, 2011)	26.74	26.16	24.59	25.19
(CONDAT; MOSADDEGH, 2012)	27.07	26.50	24.87	25.49
(GHARBI et al., 2016)	32.40	32.41	28.39	29.87
Our 4x4 noise	**33.31**	**32.90**	**29.73**	**31.02**
Noise $\sigma = 16$	**Kodak**	**McMaster**	**vdp**	**moiré**
(CHAKRABARTI, 2016)	25.28	23.96	20.16	20.26
(CHAKRABARTI; FREEMAN; ZICKLER, 2014)	22.60	21.68	20.81	21.65
(CONDAT, 2011)	24.44	24.23	23.06	23.45
(CONDAT; MOSADDEGH, 2012)	24.76	24.56	23.32	23.74
(GHARBI et al., 2016)	31.07	31.19	27.18	28.73
Our 4x4 noise	**32.17**	**31.81**	**28.56**	**29.88**
Noise $\sigma = 20$	**Kodak**	**McMaster**	**vdp**	**moiré**
(CHAKRABARTI, 2016)	24.60	23.40	19.98	20.31
(CHAKRABARTI; FREEMAN; ZICKLER, 2014)	20.93	20.24	19.60	20.23
(CONDAT, 2011)	22.61	22.62	21.70	21.96
(CONDAT; MOSADDEGH, 2012)	22.93	22.94	21.95	22.23
(GHARBI et al., 2016)	30.00	30.15	26.17	27.80
Our 4x4 noise	**31.20**	**30.87**	**27.57**	**28.93**

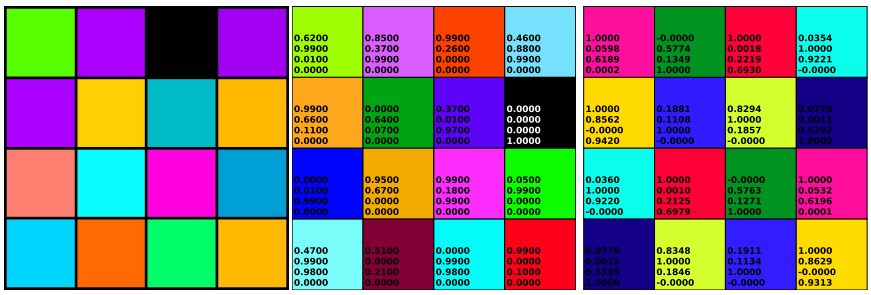
joint capture those information (LU et al., 2009; SADEGHIPOOR; LU; SüSSTRUNK, 2011; TANG et al., 2015; TERANAKA et al., 2016). Our generic architecture can be slightly modified to allow training of CFA and demosaicing for reconstructing RGB-NIR images. Fig. 3.9 compares the designs of the two existing works against ours. We didn't find the values of the colors of the CFA from Lu et al. (2009), the image was taken from their paper.

The reconstruction of RGB-NIR images is similar to the case of RGB ones. In

this case, however, the input consists of a 2-D 4-channel multispectral image, which is then projected into a 1-D mosaic, from which a full 4-channel image is reconstructed. For training, we have used the *RGB-NIR Scene Dataset* (LAQUERRE NICOLAS ETIENNE, 2011), taking 128×128 random patches from 430 images. As these images have already been demosaiced, we used an approach similar to Wang's ((WANG, 2014)) to minimize the effects of previous demosaicing methods: we downsampled (by a factor of 2) the image with a Gaussian-windowed Sinc filter ($\sigma = 1.2$ pixels), which has been proven to be an effective approach to remove bias of demosaiced images (MOREL; YU, 2011; WANG, 2014). On the test images of this dataset, our model has achieved a CPSNR of 38.92 for reconstructing a 4-channel image (39.33 for reconstructing visible light and 38.11 for reconstructing NIR).

We have found no code of existing techniques, but supplementary images (and reconstructions) were found for two techniques (LU et al., 2009; SADEGHIPOOR; LU; SüSSTRUNK, 2011), using a different test set. Table 3.4 compares the PSNRs for reconstructing the RGB (visible), NIR, and the full 4-channel image. Note how our model

Figure 3.9: CFA patterns for RGB-NIR. The CFA designs of the works we have compared previously (LU et al., 2009; SADEGHIPOOR; LU; SüSSTRUNK, 2011). Their models only include a single NIR color-filter (black color), while our allows the capture of NIR jointly with RGB on each color filter. Image from the CFA from Lu et al. (left) was taken from their paper, no additional data were found.



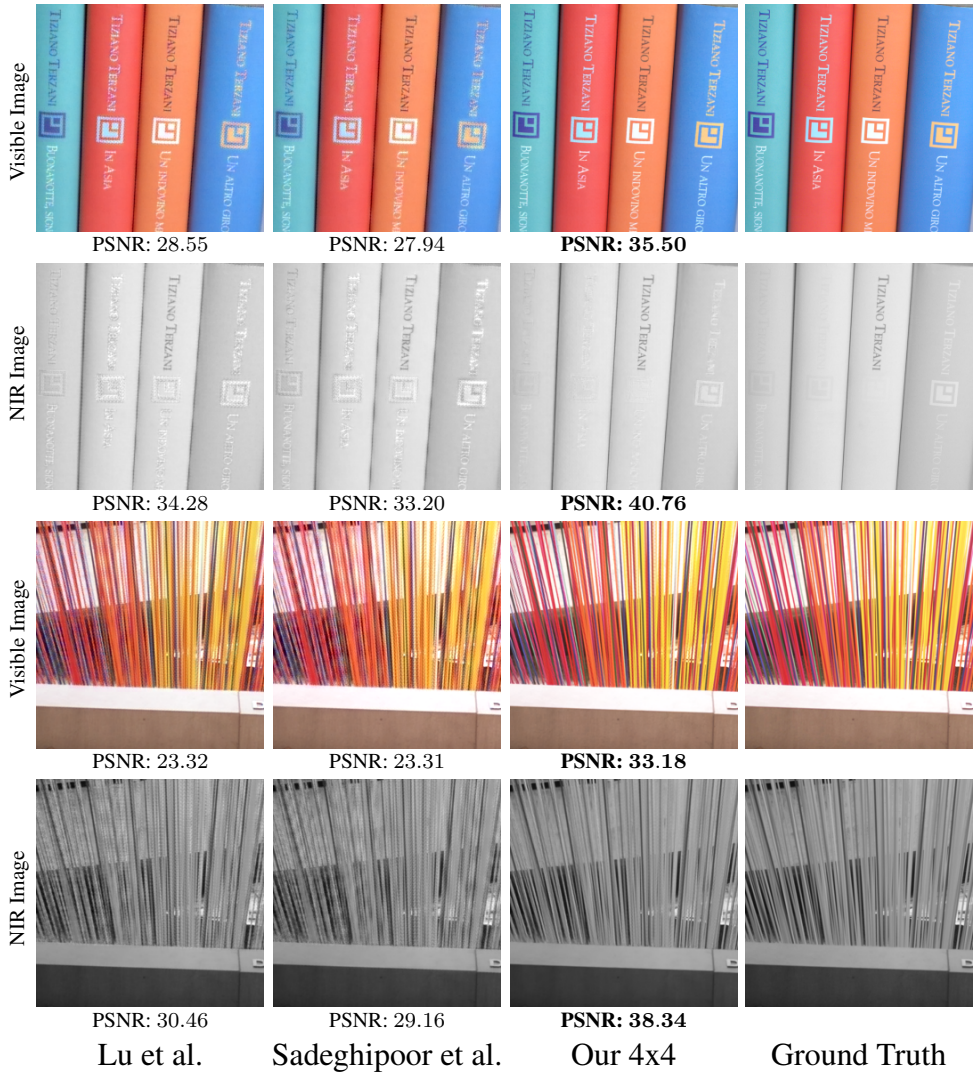
Lu et al. Sadeghipoor et al. Our 4x4 RGB-NIR

Table 3.4: Comparison of our pattern against state-of-the-art CFA for capturing RGBNIR images. The numbers show the average CPSNR values for reconstructing the visible, NIR and 4-channel image, respectively. The test and reconstructed images were taken from the supplementary material of (SADEGHIPOOR; LU; SüSSTRUNK, 2011). Our trained pattern outperforms other techniques (results in bold).

	Sadeghipoor Testset		
	Visible	NIR	All channels
Lu et al. (2009)	33.3	35.18	33.61
Sadeghipoor et al. (2011)	32.66	34.01	32.90
Our 4x4 pattern	**37.11**	**37.09**	**36.95**

achieves better PSNRs, even by training in a different dataset.

Figure 3.10: Reconstructions comparison of visible and NIR channels, respectively. Comparison of state-of-the-art techniques for capturing visible and NIR information (LU et al., 2009; SADEGHIPOOR; LU; SüSSTRUNK, 2011) against our trained model (best PSNR in bold). Images from the Sadeghipoor test set. Better visualized in the digital version.



3.5 Discussion

We evaluated several network architectures beyond the ones presented, and some observations worthy mentioning:

Deepness versus wideness: We tested shallower and deeper, as well as thinner and wider networks. Contrary to many works that claim that deepness is the key for good performance, we have found similar results to Zagoruyko and Komodakis (ZAGORUYKO; KOMODAKIS, 2016), suggesting that wideness is as important as deepness.

Skip connections: We have used skip connections to stack submosaic images for the decoder, which improved the performance and convergence of the network. We have also tested different architectures using distinct dispositions, observing no correlation between the number of connections and higher PSNR.

Additional input: By providing additional information to the demosaicing (decoder) scheme, our network is capable of learning faster and achieving better reconstructions. In addition to the mosaic image, we have also provided the CFA submosaics, and their linearly interpolated versions. This simplifies the training, as the network does not need to learn to separate each color submosaic, nor the interpolation kernels from scratch.

Masks: We use disjoint binary masks to enforce pattern repetition, and to be able to handle images of arbitrary sizes. But the masks can be used to impose additional constraints. For instance, one can use them to find 2×2 patterns with only three colors (as in the Bayer pattern), or to enforce designs that follow specific patterns, such as blue-noise characteristics (CONDAT, 2010), or diagonal designs (LUKAC; PLATANIOTIS, 2005; BAI et al., 2016) (check Fig. 3.11 for examples).

CFA pattern size: Our architecture can train CFAs with an arbitrary number of color filters, and we have opted to train designs larger than the traditional 2×2 patterns. Training bigger-sized CFA patterns has several advantages. First, they contain a larger number of distinct colors and thus provide more coverage during sampling of the color space, allowing the CNN to make better use of correlations among colors. Second, smaller CFAs are more susceptible to aliasing due to pattern repetition, while bigger CFAs allow the learning of more stochastic patterns. Third, from an optimization perspective, the 2×2 search-space is a sub-space of the 4×4 search-space, meaning that a 4×4 CFA can learn a 2×2 pattern if it is advantageous. For instance, a careful inspection of Fig. 3.2 reveals that each of our 4×4 CFAs actually consists of two side-by-side copies of a 4×2 pattern. The small differences among the corresponding RGB coefficients in the 4×2 patterns in each CFA are fairly small, and are likely to be reduced with longer training. This suggests that 4×2 patterns are the most efficient tileable representations for CFAs achievable with a 4×4 pattern.

Manufacturing our CFAs: Our encoder optimizes the CFA colors over the full RGB space. Although our 4×4 CFAs do not use the standard Bayer color filters, the colors used in our patterns are a linear combination of these standard filters, which should simplify the manufacturing process. Alternatively, Chiulli (CHIULLI, 1989) has patented a technique for creating color filters from any combinations of red, green, blue, cyan, yellow

and magenta dies. More recently, SILIOS Technologies has developed a manufacturing technique called COLOR SHADES® for producing band-pass filters (SILIOS Technologies, 2017). This technology combines thin film deposition and micro/nano-etching processes onto a silica substrate (LAPRAY et al., 2014). COLOR SHADES® provides band-pass filters in the visible range from 400 nm to 700 nm (as well as in the IR range). Lapray et al. (LAPRAY et al., 2014) describe the construction of a multispectral CFA using eight optical filter bands produced with COLOR SHADES®, and compare the simulated and measured responses of the individual filters. This technology could be used to produce our CFAs.

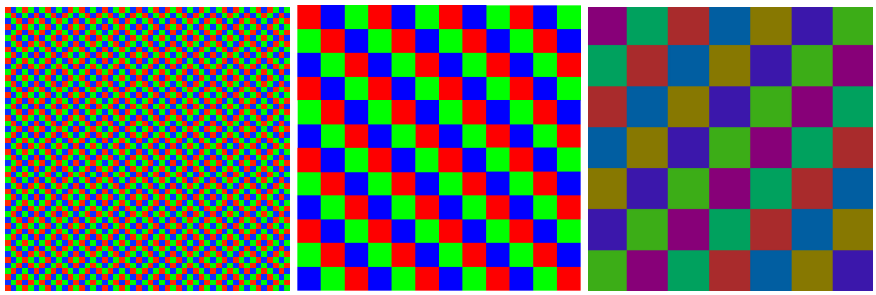
Demosaicing of extremely high-frequency content is challenging to all demosaicing methods. Fig. 3.7 shows examples of two image patches the for which all techniques, including ours, are unable to obtain high-quality image reconstructions. Such problems are due to aliasing, when color high-frequency details cannot be appropriately sampled by the CFA (GHARBI et al., 2016). Note that the artifacts in the reconstructions by the techniques of Mairal et al. and Gharbi et al. are similar. Both use the same Bayer CFA, indicating that such moiré artifacts are due to CFA color subsampling, rather than to the demosaicing method itself.

Before arriving at the described architecture, we have systematically tried many alternatives. Such exploration included the use of L1 and L2 regularizers, different optimizers (Adadelta and Adam), dropouts (SRIVASTAVA et al., 2014), various configurations of skip-connections (Fig. 3.5), different sizes of CFA patterns (including 6×6 and 8×8), and trainable/fixed interpolation layers. While testing all combinations of these elements is unfeasible, we have made extensive experimentation. The results of these tests indicated that the architecture for the 4×4 patterns (both for noise-free and noisy patterns) achieved the overall best PSNR results. Note that the CFA designs learned for the noise-free and for the noisy cases are similar, one being a shifted version of the other. This indicates that those colors were not found by chance, and they indeed provide lower reconstruction errors.

3.6 Summary

This Chapter presented a convolutional neural network architecture for performing joint design of color filter arrays, demosaicing, and denoising. By expressing the CFA projection and linear interpolation as convolutional layers, our network finds the filter

Figure 3.11: Examples of patterns following specific designs. From left to right: patterns proposed by Condat (2010), Lukac et al. (2005), and Bai et al. (2016), respectively. Our architecture can implement such design strategies by training with appropriate masks.



pattern and corresponding demosaicing method that jointly minimize image reconstruction error. The patterns and algorithms produced by our method provide high-quality color reconstructions, surpassing the state-of-the-art techniques on all standard demosaicing datasets.

Our approach can also reconstruct high-quality images from noisy data, outperforming existing techniques for all noise levels, without requiring any information about the noise level in the input data. In addition, it can be used to obtain effective demosaicing strategies for existing CFA patterns.

As color capture (and reconstruction) is the first step of image acquisition, any advance made in this area would have an important impact on all the Digital Photography field. Camera manufacturers and, in turn, photographers and the general public can benefit from the superior color reconstruction provided by our method. In the same way, the design of CFA patterns and reconstruction algorithms could be further tuned for acquiring HDR content in single captures.

4 SYNTHESIZING CAMERA NOISE USING GENERATIVE ADVERSARIAL NETWORKS

Noise is an old and well studied problem. Nonetheless, there is still no method or technique capable of synthesizing noise as it is found when capturing natural photographs. This Chapter presents a technique for synthesizing realistic noise for digital photographs. It can adjust the noise level of an input photograph, either increasing or decreasing it, to match a target ISO level. Our solution learns the mappings among different ISO levels from unpaired data using generative adversarial networks. We demonstrate its effectiveness both quantitatively, using Kullback-Leibler divergence and Kolmogorov-Smirnov test, and qualitatively through a large number of examples. We also demonstrate its practical applicability by using its results to significantly improve the performance of a state-of-the-art trainable denoising method. Our technique should benefit several computer-vision applications that seek robustness to noisy scenarios.

4.1 Introduction

Noise is a fundamental problem in graphics, image processing, and computer vision, and many image-denoising techniques have been proposed in recent years (DABOV et al., 2007; LIU et al., 2008; BARBU, 2009; BURGER; SCHULER; HARMELING, 2012; CHEN et al., 2013). While some of these approaches are highly successful in removing artificial additive white Gaussian noise (AWGN), recent works (ANAYA; BARBU, 2014; PLOTZ; ROTH, 2017; ABDELHAMED; LIN; BROWN, 2018) have shown that the performance of such techniques is severely reduced when applied to real photographs. This is particularly true for the case of recent deep learning strategies (ZHANG et al., 2017; CHEN; POCK, 2017). The main difficulty faced by these techniques results from the fact that noise found in digital photographs, which we refer to as *natural noise* (in opposition to *synthetic noise*), has multiple sources (*e.g.*, thermal, quantization, etc.), being much more complex than just white Gaussian noise.

While the importance of noise reduction is well understood, increasing noise level is also very useful. It can provide data for training techniques that need to handle noisy scenarios. These include, for instance, improving the performance of denoisers (as we will demonstrate), classifiers for low-light and challenging conditions (DODGE;

KARAM, 2016; DIAMOND et al., 2017; ROY et al., 2018); performing superresolution in the presence of different noise levels (MANDAL; BHAVSAR; SAO, 2017); performing noise reduction during demosaicing (HENZ; GASTAL; OLIVEIRA, 2018); and detecting forgeries based on noise statistics (KAUR; WALIA, 2016). These and other applications would benefit from the synthesis of realistic noise.

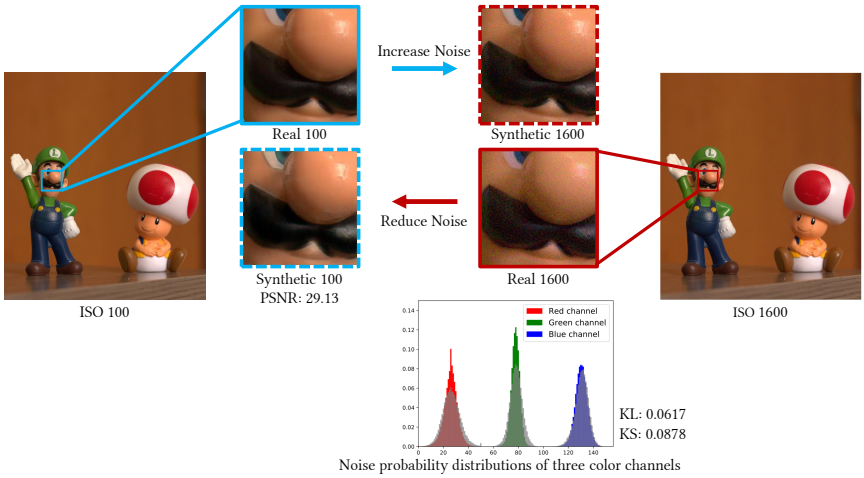
Although many works have studied the nature of noise theoretically (ALTER; MATSUSHITA; TANG, 2006; HASINOFF; DURAND; FREEMAN, 2010; HASINOFF, 2014; EMVA, 2016), no denoising technique seems to be able to directly use such information. We propose a *practical* data-driven solution for synthesizing noise that can, for instance, be used for training/fine-tuning denoising algorithms intended for real-world applications. Our technique can adjust the noise level of an input photograph to match a target ISO level. Fig. 4.1 illustrates the process: a photograph captured with ISO 100 (left) has its noise level adjusted to ISO 1600 (blue arrow). Likewise, another photograph of the same scene taken with ISO 1600 (right) has its noise level adjusted to ISO 100 (red arrow).

The noise distributions of the highlighted patches for the ISO 1600 photograph (Real 1600) and the one generated by our technique (Synthetic 1600) have a Kullback-Leibler (KL) divergence of 0.0617 and the result of their Kolmogorov-Smirnov (KS) test is 0.0878 (with a p-value of 7.81×10^{-220}), indicating that such distributions are very similar. Since we use the ISO 100 photograph as baseline for estimating the noise distributions, we cannot apply the same tests to ISO 100 patches. For this reason, Fig. 4.1 shows the PSNR value (29.13) computed for the image generated by our technique (Synthetic 100). Such value indicates good agreement with the patch of the actual ISO 100 photograph (Real 100).

Although it would be preferable to directly use noise variance instead of ISO levels for parameterizing a noise-adjustment process, existing variance-estimation techniques (Makovoz, 2006; LIU; TANAKA; OKUTOMI, 2013; Chen; Zhu; Heng, 2015; PETROVIC; PETROVIC; NIKOLIC, 2016) do not provide reliable estimates, as they consider additive white Gaussian noise (AWGN). ISO level, in turn, is a readily available and reliable information, which justifies our choice. As robust noise variance-estimation techniques become available, our approach can be adapted to use them.

To perform noise-level adjustment, we use a convolutional neural network (CNN). Unfortunately, the availability of datasets containing paired photographs captured under different ISO settings is limited, and creating a large one is a non-trivial task. Thus, we designed our technique to use unpaired datasets. We modify the cycle-consistency

Figure 4.1: Using our technique to adjust the noise level of photographs to different ISO values. A photograph captured with ISO 100 (left) has its noise level adjusted to ISO 1600 (blue arrow). Likewise, another photograph of the same scene taken with ISO 1600 (right) has its noise level adjusted to ISO 100 (red arrow). The noise distributions for patches Real 1600 and Synthetic 1600 have a Kullback-Leibler (KL) divergence of 0.0617 and the result of their Kolmogorov-Smirnov (KS) test is 0.0878 (with a p-value of 7.81×10^{-220}), indicating that the two distributions are very similar. The red, green, and blue histograms underneath patch Real 1600 show the noise distributions corresponding to the R, G, and B channels of patch Real 1600, respectively. The superimposed gray histograms are the corresponding noise distributions for the patch Synthetic 1600. The PSNR value computed for patch Synthetic 100 is 29.13, also indicating a good agreement with patch Real 100.

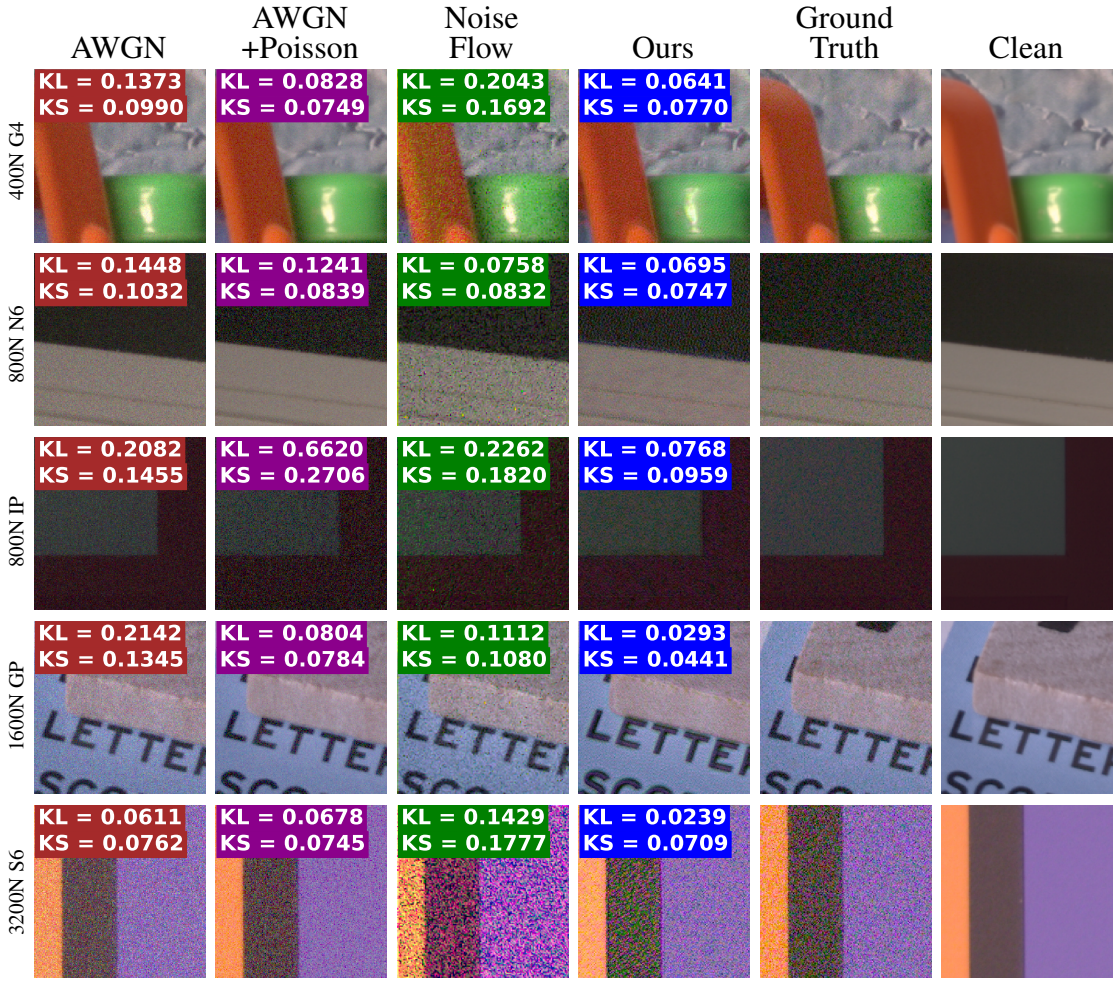


loss, and introduce a *low-frequency-consistency loss* term to preserve the contrast of the input image in the synthesized one. An ablation study shows how these changes and our modified generator architecture lead to high-frequency content that approximates natural noise (Section 4.5.1).

Fig. 4.2 compares the results produced by our technique with the ones generated with AWGN, Gaussian-Poissonian (AWGN+Poisson) and with Noise Flow (ABDELHAMED; BRUBAKER; BROWN, 2019) for various ISO levels and different smartphone camera models. For the three techniques, each noisy result was obtained by corrupting the corresponding *clean patch* provided as part of the SIDD (Smartphone Image Denoising Dataset) dataset. Each *clean image* in SIDD was obtained after processing 150 pictures taken from the same scene (ABDELHAMED; LIN; BROWN, 2018). AWGN and our technique were applied in sRGB space, while Noise Flow was applied in raw space. The numbers inside synthesized patches are values of the KL divergence and the KS test computed with respect to the corresponding ground truth, which consists of an actual photograph captured at the target ISO level. For all examples shown in Fig. 4.2, the images synthesized by our method obtained KL and KS results significantly smaller (better), and

textures similar to the corresponding ground truths.

Figure 4.2: Comparison of synthesized noise obtained by corrupting a *clean patch* for different ISO values and camera models. AWGN and AWGN+Poisson use the average noise variances computed from the SIDD paired dataset, in sRGB and linear space, respectively. Noise Flow are applied in raw space. Our results achieved the best (smaller) KL divergence and KS values, and exhibit textures similar to the corresponding ground truths (actual photographs taken at the target ISO levels). The clean images are provided as part of the SIDD dataset.



We validate our technique both quantitatively and qualitatively. For this, we use KL divergence, KS test, discriminative evaluation, and t-SNE visualizations. Together, these evaluations show that the proposed model generates noise much closer to natural than existing techniques. Finally, we demonstrate a practical application of our technique: a significant improvement in the performance of a state-of-the-art denoiser (Section 4.7).

The **contributions** of this work include:

- *A method for adjusting the noise level of an input photograph to match a target ISO level (Section 4.4). Its results produce significantly better approximations to natural noise than previous synthetic noise generators;*

- *A new loss formulation for use with CycleGANs for allowing the adjustment of noise level* (Section 4.4.4). Such new loss function results in more realistic noise, whose statistics approximate the ones of real photographs with the same ISO value;
- *A large (unpaired) dataset containing over 2.1 million* 256×256 *patches from photographs captured under different ISO values with a Canon Rebel T3i* (Section 4.5). The distribution of patches per ISO level is balanced, and the patches did not undergo any denoising, making this a suitable dataset for denoising and noise-synthesis applications.

4.2 Related Work

Our technique focus on adjusting the noise level of an image. Next, we discuss works on denoising, noise synthesis, and paired noise datasets.

4.2.1 Denoising Methods

Denoising is a well-studied problem and several techniques have been proposed to handle it. Among them, many methods model image priors based on non-local similarities (BUADES; COLL; MOREL, 2005; DABOV et al., 2007; BUADES; COLL; MOREL, 2008), sparse representations (ELAD; AHARON, 2006; MAIRAL et al., 2009; DONG et al., 2013; GIRYES; ELAD, 2014), total-variation optimization (RUDIN; OSHER; FATEMI, 1992; OSHER et al., 2005), and Markov-Random-Field (MRF) models (LAN et al., 2006; LI, 2009; ROTH; BLACK, 2009). Such methods have high-computational costs, heavily relying on the selection of parameter values.

Discriminative learning methods focus on learning inference functions, either based on random-field architectures (SCHMIDT; ROTH, 2014), reaction-diffusion models (CHEN; YU; POCK, 2015; CHEN; POCK, 2017), or conditional random fields (SCHMIDT et al., 2013; SCHMIDT et al., 2016). Recently, deep learning has become a trend on denoising methods. Jain and Seung proposed one of the first methods to use CNNs for denoising (JAIN; SEUNG, 2009). Burguer et al. (BURGER; SCHULER; HARMELING, 2012) showed how a plain multi-layer perceptron (MLP) trained on large datasets can compete with BM3D (DABOV et al., 2007). Xie et al. combine sparse coding and denoising autoencoders to address low-level-vision problems such as denoising and inpainting (XIE;

XU; CHEN, 2012). Mao et al. proposed an autoencoder architecture with symmetric skip connections, training a single model to handle different noise levels (MAO; SHEN; YANG, 2016). Zhang et al. used a single residual CNN, combined with batch-normalization layers, for blind Gaussian denoising (ZHANG et al., 2017). Later, Guo et al. proposed a convolutional blind denoising network (CBDNet) (GUO et al., 2019) trained with a noise model more complex than AWGN, surpassing existing methods on benchmarks with real photographs. Lehtinen et al. introduced the idea of learning to denoise using pairs of corrupted images (LEHTINEN et al., 2018). While it removes the necessity of laboriously collecting noisy-clean pairs for the denoiser training, it still requires at least two realizations of each scene.

The majority of these methods rely on pairs of clean and corrupted images, either to find optimal parameters (for approaches based on image priors), or to fully train discriminative learning techniques. We emphasize that *our technique is not intended to replace denoising methods*. On the contrary, it benefits them by providing more realistic training data, as we demonstrate in the paper.

A recent work by Brooks et al. (BROOKS et al., 2019) proposes a technique to invert the transformations performed during the imaging-processing pipeline (gain, color correction, etc.) before performing denoising. Our work, on the other hand, learns how the entire pipeline affects noise. The two techniques could be combined, with our method applied to the "untransformed" images produced by their approach.

4.2.2 Noise Synthesis

Besides AWGN, a few additional synthetic noise models have been proposed. Foi et al. (FOI et al., 2008) described one of the first models to try to improve AWGN by combining Poissonian and Gaussian noise. Hwang et al. use a Skellam distribution for modeling Poisson photon noise (HWANG; KIM; KWEON, 2012). More recent approaches propose in-camera imaging models, being able to account for cross-channel noise modeling (KIM et al., 2012; NAM et al., 2016). Similar to the works described in (KIM et al., 2012; NAM et al., 2016), our data-driven approach takes into account the in-camera pipeline, both in terms of how it can modify the captured noise (*e.g.*, through gamut mapping or demosaicing), as well as accounting for other sources of noise (*e.g.*, quantization). Unlike (KIM et al., 2012; NAM et al., 2016), our method provides a practical solution that learns the marginal distribution of patches with a given ISO setting, allowing it

to map an input image from one ISO setting to another. Newson et al. (NEWSON; DELON; GALERNE, 2017) and Eckel et al. (Eckel et al., 2020) presented techniques that try to faithfully model film noise. The work most similar to ours is Noise Flow (ABDELHAMED; BRUBAKER; BROWN, 2019): a recent machine-learning approach that seeks to minimize the negative log-likelihood (NLL) between the generated and ground-truth noise. We show that our method, besides not needing paired data, is capable of training a denoiser with superior performance compared to Noise Flow.

4.2.3 Paired Noise Datasets

Recently, researchers have built paired datasets with images captured with different ISO settings. Anaya and Barbu (ANAYA; BARBU, 2014) constructed a dataset obtained under low-light conditions, taken with a point-and-shoot (Canon PowerShot S90), a DSLR (Canon EOS Rebel T3i), and a mobile (Xiaomi Mi3) camera. They captured pairs of images (shot at ISO 100 and at a higher ISO value), and showed how to align the pixel intensity values from the pairs of images taken with the same camera. Plotz and Roth (PLOTZ; ROTH, 2017) propose a similar dataset, capturing images using four different consumer cameras (Sony A7R, Olympus OMD E-M10, Sony RX100 IV, and Nexus 6P), with different sensor sizes. They also propose a post-processing procedure based on the heteroscedastic Tobit regression model to align pixel intensities. Abdelhamed et al. (ABDELHAMED; LIN; BROWN, 2018) presented the SIDD dataset consisting of 10 scenes, each captured with five smartphone cameras and using different lighting conditions. All these works show that many recent techniques trained for denoising AWGN (BURGER; SCHULER; HARMELING, 2012; DONG et al., 2013; SCHMIDT; ROTH, 2014; CHEN; YU; POCK, 2015) are outperformed by BM3D (DABOV et al., 2007) when applied to natural noise. This suggests that *the use of AWGN (either for training or evaluation) does not generalize well for the case of natural noise.*

For training our generative model, we have built a large *unpaired dataset* containing over 2.1 million 256×256 image patches taken from photographs captured under different ISO settings (from 100 to 3200) using a Canon Rebel T3i camera. We refer to it as *our_T3i dataset* (to avoid confusion with Renoir T3i (ANAYA; BARBU, 2014)). *our_T3i* has a balanced distribution of ISO levels, and consists of sRGB images with no denoising applied. It is a valuable resource for training denoising and noise-synthesis applications. We intend to make this dataset publicly available.

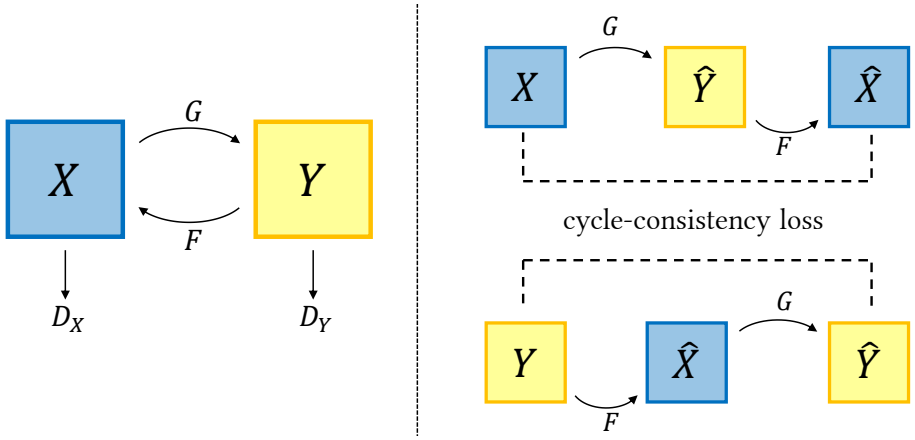
4.3 Generative Adversarial Networks

This section provides a brief review of Generative Adversarial Networks (GANs), touching on concepts that will be relevant for following sections. GANs were introduced by Goodfellow et al. (GOODFELLOW et al., 2014), with many variations appearing in the following years (MIRZA; OSINDERO, 2014; Odena, 2016; CHEN et al., 2016). A GAN consists of two neural networks – a *generator* and a *discriminator* – that are trained together, often aiming to learn to generate images from a specific domain. While the *generator* focuses on learning how to map data from a latent space to the target domain, the *discriminator* learns to distinguish actual elements of the target domain from ones synthesized by the generator. This strategy offers training benefits for both networks: the discriminator helps the generator to synthesize images that better fit the target domain (via backpropagation), while the generator produces unseen samples to train the discriminator.

Image-to-image translation seeks to obtain two mapping functions: $G : X \mapsto Y$, and $F : Y \mapsto X$, where X and Y are distinct image domains. Normally, for a neural network to learn such functions, it would require paired training data, consisting of the same sample on both domains. Zhu et al. show how the use of discriminators can enable the training on unpaired data (ZHU et al., 2017). Their technique, called CycleGAN, makes use of two *discriminators*: D_X, which aims at telling whether a given image belongs to the X domain, and D_Y, doing the same for the Y domain. Fig. 4.3 (left) illustrates how such process works: D_Y encourages G to translate from X to Y, and D_X enforces F to take an image from Y and output the corresponding image inside the image-distribution of domain X. Note that, in this case, G and F are the *generators*, but instead of mapping from a latent space, they map between domains with different image distributions. To constrain such an ill-posed problem, the authors use an additional loss function term called *cycle-consistency loss*. This term enforces that G and F should be inverses (conceptually), making both mappings bijective. In practice, it enforces that given $x \in X$, then $F(G(x)) \approx x$. Likewise, given $y \in Y$, then $G(F(y)) \approx y$ (Fig. 4.3 (right)). A similar idea was independently proposed by Yi et al. (YI et al., 2017a).

Existing alternatives for performing unpaired training of image-to-image translation include CoGAN (LIU; TUZEL, 2016), DualGAN (YI et al., 2017b), and MUNIT (HUANG et al., 2018). We selected CycleGANs due to its superior results and flexibility for working on many tasks. However, the original CycleGAN formulation is not appropriate for our problem. Thus, we have designed a new loss formulation suited for adjusting the noise

Figure 4.3: Unpaired training and cycle-consistency loss. *Discriminators* D_X and D_Y tell if a translated image belongs (or not) to a given domain (left). The cycle-consistency loss enforces G and F to be inverse-like (right).



level of a given image to match the noise level of a target ISO value.

4.4 Adjusting Image Noise Levels

Our method uses a GAN architecture inspired by the work of Zhu et al. (ZHU et al., 2017). Its goal is to learn mapping functions $G : X \mapsto Y$ and $F : Y \mapsto X$ between the domains X and Y. X represents the domain of images captured under a lower ISO setting, and Y represents images captured with a higher ISO. Thus, the mapping function G should learn to increase the noise level, while F should learn to reduce it. During training, two discriminators D_X and D_Y learn to discriminate images belonging to domains X and Y, respectively. Our loss function consists of three terms: an adversarial loss (GOODFELLOW et al., 2014), a modified cycle-consistency loss (ZHU et al., 2017), and a novel *low-frequency-consistency loss*.

4.4.1 Adversarial Loss

Adversarial losses are used to enforce that the output of generators G and F match the image distributions learned by the discriminators D_Y and D_X, respectively. Instead of using the original adversarial loss presented in (GOODFELLOW et al., 2014), we use the Least-Squares adversarial loss (MAO et al., 2016) due to its increased stability during

training. For G and D_Y, the adversarial loss is defined as:

$$\begin{aligned}\mathcal{L}_{\text{LSGAN}}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{data}(y)}[D_Y(y)^2] \\ + \mathbb{E}_{x \sim p_{data}(x)}[(1 - D_Y(G(x)))^2],\end{aligned} \tag{4.1}$$

where D_Y outputs 1 when it accepts its input as part of distribution of domain Y, and 0 otherwise. During training, G aims to minimize this loss, while D_Y aims at maximizing it. This way, D_Y is trained to accept images from domain Y and reject the ones generated by $G(x)$. In turn, G is trained to make its output similar to the ones in domain Y (*i.e.*, make its output to be accepted by D_Y). A similar adversarial loss is used for F and D_X, *i.e.*, $\min_F \max_{D_X} \mathcal{L}_{\text{LSGAN}}(F, D_X, Y, X)$.

4.4.2 Cycle-consistency Loss

Zhu et al. argue that both generators should be cycle-consistent, *i.e.*, for each image $x \in X$, $F(G(x)) \approx x$ (ZHU et al., 2017). Likewise, for each image $y \in Y$, $G(F(y)) \approx y$. However, *such a constraint is not entirely suited to our problem due to the stochastic nature of noise*. In particular, two noisy photographs of the same scene have significantly different pixel values, despite being corrupted by noise realizations coming from *the same* ISO-level noise distribution. As such, in combination with an l_1 or l_2 norm, the optimization would smooth images x and y to maximize their similarity, being unable to increase noise level.

To overcome this limitation, we propose a new cycle-consistency loss, based on the observation that the noise-to-signal ratio is higher in the high-frequency Fourier components of natural images. Thus, our loss isolates these components, and considers only their total energy (in contrast to exact pixel values). More precisely, it is defined as the l_2-norm $\|\cdot\|_2$ of the channel-wise (R, G, and B) variance (var) of the high-frequencies of the patches x against $F(G(x))$, and likewise for the patches y against $G(F(y))$. Thus, *our cycle-consistency loss* is defined as:

$$\begin{aligned}\mathcal{L}_{cyclic}(G, F) = \mathbb{E}_{x \sim p_{data}(x)}[\|\text{var}(\mathbb{H}(F(G(x)))) - \text{var}(\mathbb{H}(x))\|_2] \\ + \mathbb{E}_{y \sim p_{data}(y)}[\|\text{var}(\mathbb{H}(G(F(y)))) - \text{var}(\mathbb{H}(y))\|_2],\end{aligned} \tag{4.2}$$

where $\mathbb{H}(x) = x - \mathbb{G}_\sigma(x)$ is the high-frequency content of image patch x, and $\mathbb{G}_\sigma(\,\cdot\,)$ denotes a blur using a Gaussian kernel with standard deviation σ. Appendix A.1 discusses how to choose σ values for different ISO values. The low-frequency component

of x, $\mathbb{G}_\sigma(x)$, is a key part of our low-frequency-consistency loss term (Section 4.4.3).

During training, we have noticed no major changes when computing variance over the entire patch or computing moving variance over small windows across the patch. Thus, to speed up the training, the variance is computed once over the entire patch.

4.4.3 Low-frequency-consistency Loss

The adversarial loss terms (Eq. 4.1) make the outputs of generators to be accepted by the corresponding discriminators, and the cycle-consistency loss term (Eq. 4.2) provides additional constraints, enforcing a given input to maintain high frequencies after passing through both G and F. Despite these terms, the problem of image-to-image translation is still ill-posed. For instance, we have found that only using adversarial and cycle-consistency loss terms, the resulting mapping functions tend to not preserve the original colors and global contrast of the input image. We thus introduce another novel loss term that addresses this problem. It is conceptually a dual of the loss from Eq. 4.2, based on the observation that *given a scene photographed using two different ISO levels, the low-frequency content of both images should be the same, regardless of the used ISO levels*. This happens because, as mentioned earlier, low-frequency components are not significantly corrupted by the relative noise amplitude. Thus, for a given image $x \in X$, the low-frequency contents of both x and $G(x)$ should be similar. The same goes for y and $F(y)$. *Our novel low-frequency-consistency loss* is defined as:

$$\begin{aligned}\mathcal{L}_{low\text{-}freq}(G, F) = \mathbb{E}_{x\sim p_{data}(x)}[\|\mathbb{G}_\sigma(G(x)) - \mathbb{G}_\sigma(x)\|_2] \\ + \mathbb{E}_{y\sim p_{data}(y)}[\|\mathbb{G}_\sigma(F(y)) - \mathbb{G}_\sigma(y)\|_2].\end{aligned} \tag{4.3}$$

4.4.4 The Complete Loss Function

The complete loss is then given by:

$$\begin{aligned}\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{\text{LSGAN}}(G, D_Y, X, Y) \\ + \mathcal{L}_{\text{LSGAN}}(F, D_X, Y, X) \\ + \lambda_1 \mathcal{L}_{cyclic}(G, F) \\ + \lambda_2 \mathcal{L}_{low\text{-}freq}(G, F),\end{aligned} \tag{4.4}$$

where λ_1 and λ_2 control the importance of each term. After some experimentation, we empirically found that $\lambda_1 = \lambda_2 = 10$ works best. For all results described in this Chapter we used this setup. The optimization process is guided by

$$G, F = \arg\min_{G,F} \max_{D_X, D_Y} \mathcal{L}(G, F, D_X, D_Y). \tag{4.5}$$

4.4.5 Network Architectures

Fig. 4.4 illustrates the architectures for both generators (top) and discriminators (bottom). For discriminators D_X and D_Y, we use the architecture described by Zhu et al. (ZHU et al., 2017), which consist of a convnet classifier based on PatchGANs (ISOLA et al., 2017; LI; WAND, 2016), with patch size of 70×70 pixels. For the generators G and F, however, we have opted for different architectures.

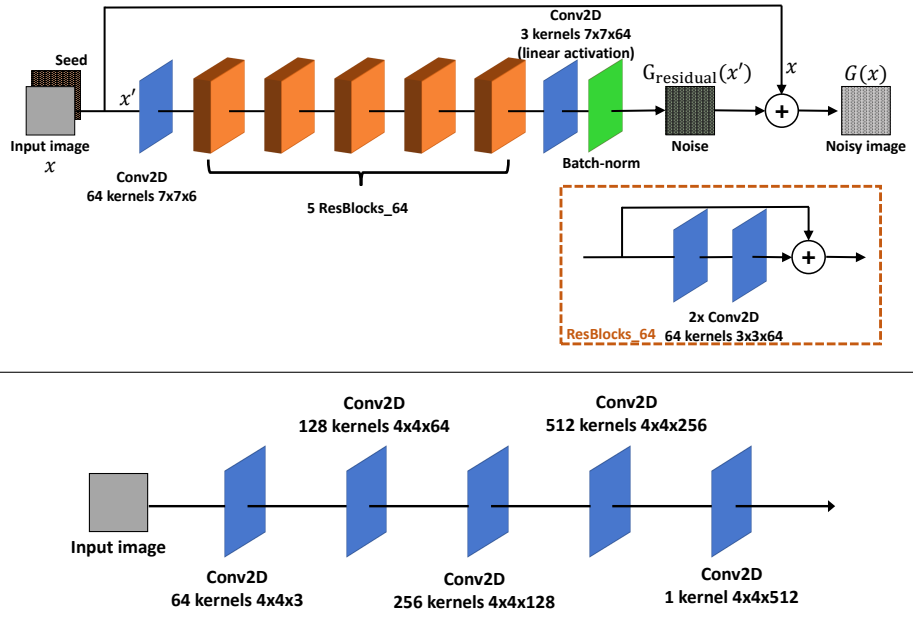
Different from architectures like U-Net (RONNEBERGER; P.FISCHER; BROX, 2015), we do not use any downscaling/upsampling layers (nor any convolutional layer with stride different from 1). Our generator G adds the input image x to a residual produced by a modified version of the architecture described by Zhu et al. (ZHU et al., 2017) obtained by removing its `tanh` layer. The output of G for a given input image x is given by

$$G(x) = x + \mathrm{G}_{\mathrm{residual}}(x'), \tag{4.6}$$

where $\mathrm{G}_{\mathrm{residual}}(x')$ is the residual to be added to the input image x, and x' is the concatenation of x with a random noise image of same size (*i.e.*, for a 3-channel $W \times H$ image x, x' is a 6-channel $W \times H$ image). The noise in x' works like a seed for the generator, guaranteeing that different versions of the output will be produced even if the same image is provided as input multiple times. A similar use of seed for modeling a distribution of possible outputs appears in (ZHU et al., 2017).

The generator F follows the same idea, with the exception that $x' = x$, as F does not require a stochastic behavior to reduce noise. Section 4.5.1 shows that our architecture is more suited to our problem, being capable of increasing (or attenuating) high-frequency noise.

Figure 4.4: Architecture of our GANs: (top) our generators consist of five residual blocks, two Conv2D and one Batch-Normalization layers. Each residual block consists of two Conv2D layers. All Conv2D layers (including the ones in the residual blocks) are preceded by a reflection padding and followed by an Instance-Normalization and ReLU layers; (bottom) the discriminators consist of five Conv2D layers, all followed by an Instance-Normalization and leaky-ReLU layer.



4.4.5.1 Implementation Details

Our GAN implementation was inspired by Zhu et al.'s work (ZHU et al., 2017). This includes the use of residual blocks (HE et al., 2016), instance normalization (ULYANOV; VEDALDI; LEMPITSKY, 2016), and PatchGANs in the discriminators (ISOLA et al., 2017). However, we made some important modifications, especially on the generators architecture, for improving the handling of high frequencies. A textual description of the final architectures can be found in the next section. For training, we used the Adam optimizer (KINGMA; BA, 2014) with default parameters and learning rate of 0.0002, linearly decaying to zero in the course of 2 epochs (totaling more than 800k iteration updates). We used batch size of 1. Specifically for training the discriminator, we followed the idea of Shrivastava et al. (SHRIVASTAVA et al., 2017) of updating the discriminator using a history of generated images.

4.4.5.2 Network Descriptions

Generators' architecture. The architecture of our F and G generators were designed to better handle high-frequency content. Let `Ck-f` denote a $k \times k$ 2D convolution layer with f filters, and let `Res-k` represent a residual block consisting of a Reflection padding, Conv

layer, Reflection padding and another Conv layer, in this order, where both Conv layers use 3×3 kernels and k filters. `Ref` represents a Reflection padding layer, and `BN` is a batch-normalization layer. All Conv layers are followed by an Instance-Normalization layer and a ReLU. The architecture of the generators is as follows: `Ref, C7-64, Res-64, Res-64, Res-64, Res-64, Res-64, Ref, C7-64, BN` (Fig. 4.4, top).

Discriminators' architecture. We have used the same architecture as Zhu et al. (ZHU et al., 2017) for the discriminators D_X and D_Y. Following the same notation of names defined above, and considering that for the discriminator all Conv layers are followed by Instance-Normalization and leaky-ReLU (with decay of 0.2), the discriminator architecture consists of: `C4-64, C4-128, C4-256, C4-512` (Fig. 4.4, bottom).

4.5 Experiments and Results

We present a series of experiments demonstrating the effectiveness of our technique. Our CNN was trained using our unpaired dataset. The training and test datasets consist of sRGB photographs captured under different ISO levels (ISO 100, ISO 200, ISO 400, ISO 800, ISO 1600, and ISO 3200). All images (total of 13,224) were acquired in raw mode using a Canon EOS Rebel T3i, including photos from many places and object types. The images were then post-processed (demoisaicing, white balancing, gamma correction) using the *rawpy* library with default parameters for outputting images in the sRGB color space in PNG format (lossless compression). This intentionally makes our CNN account for cross-channel effects that might come from demosaicing and gamut mapping, as well as for quantization noise. To speed-up the reads from disk during training, each 5184×3456-pixel image was split into 256×256 patches. We removed patches with mean intensity above 0.5 to avoid over-exposed areas with little noise, which would not contribute to training. The remaining patches were split into training and test sets, consisting of 1,883,501 and 235,318 patches, respectively. During training, we randomly crop a 128×128-pixel region from each 256×256 patch. Data augmentation, in the form of horizontal/vertical flips, and $\pm 90^{\circ}$ rotations, is used for each cropped region.

Figure 4.5: Ablation study, mapping from ISO 100 to ISO 1600. (a) input image patch (captured w/ ISO 100). (b) Result obtained with the model trained with Zhu et al.'s (ZHU et al., 2017) architecture and loss formulation. Note the significant color shift and change in global contrast. (c) output of model trained with Zhu et al.'s architecture and **our** loss formulation. Colors have been better preserved, but contrast has not. (d) output of model trained with the proposed architecture (Section 4.4.5) and our loss formulation; (e) a patch taken from an image captured with ISO 1600 (ground truth). *We encourage the reader to zoom in for better visualization of the noise.*

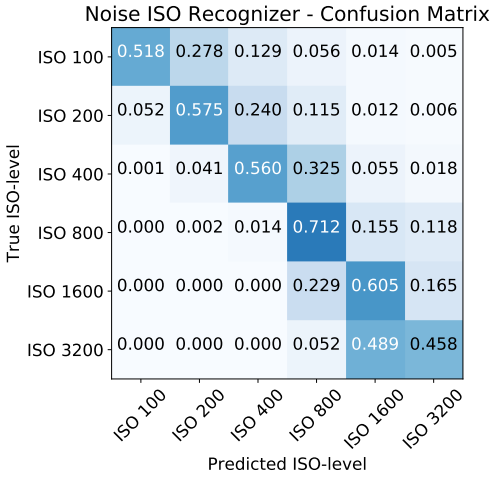
(a) (b) (c) (d) (e)

4.5.1 Ablation Study

We present an ablation study that shows the impact of each of our decisions (regarding the loss function and architecture designs). For this study, we have trained our entire CNN (G, F, D_X, and D_Y), X being the domain of photographs captured with ISO 100, and Y the domain of images captured with ISO 1600. The number of training patches was 297,211 for ISO 100, and 303,368 for ISO 1600.

Fig. 4.5 compares three training experiments: Fig. 4.5(b) shows results obtained using Zhu et al.'s (ZHU et al., 2017) loss functions and architecture. Fig. 4.5(c) shows results obtained using our modified cycle-consistency loss (Eq. (4.2)) and our low-frequency-consistency loss (Eq. (4.3)), but still using Zhu et al.'s (ZHU et al., 2017) generator architecture. Fig. 4.5(d) shows results obtained using our loss formulation (Eq. (4.5)) and architectures for generators G and F (see Section 4.4.5). Fig. 4.5(e) shows the corresponding patch taken from an image captured with ISO 1600 (ground truth). All the models were trained using the same training set and with the same number of iterations (4 epochs

Figure 4.6: Difficulty in distinguishing adjacent ISO levels: normalized confusion-matrix obtained for an ISO-level classifier. Note the higher confusion values around the main diagonal.



each). Notice how the additional constrains imposed by the low-frequency-consistency loss enforce color and global scene contrast preservation. Our convolutional architecture, combined with residual strategy (Eq. (4.6)) enables the network to better learn how to synthesize (in the case of G) and remove (in the case of F) high-frequency noise.

4.5.2 Multi-ISO Mapping

One possible strategy for transforming between arbitrary pairs of ISO values is to define a chain of mappings between pairs of adjacent ISO levels. However, mappings between adjacent ISO values tend to undershoot the noise, behaving like an identity function. This happens because the domains of neighbor ISO-levels are very similar, making it very hard for discriminators to learn how to separate such domains. This is illustrated in Fig. 4.6, where the confusion matrix for a classifier that detects the ISO of a given patch got an average accuracy of only 59.03%. This shows how hard it is to distinguish between adjacent ISO values. Details about the classifier's architecture can be found in Appendix D.

An inspection of the confusion matrix in Fig. 4.6 shows that it is virtually impossible to distinguish between ISO 1600 and ISO 3200 photographs, as well as between ISO 100 and ISO 200 pictures. The inspection also reveals that one can confidently map between ISO 100 and all other ISO values, except ISO 200. Thus, we designed a multi-ISO mapping scheme that takes advantage of this observation. The mapping between ISO 100 and ISO 200 is done indirectly through ISO 1600. By transitivity, it allows mappings among all

Figure 4.7: Our CNN can adjust the noise level of input images to match that of a target ISO value. In all these examples, our CNN takes an input patch captured with ISO 100 and maps it to various target ISO values: 200, 400, 800, 1600, and 3200. We encourage the reader to zoom in these images to inspect the noise levels.

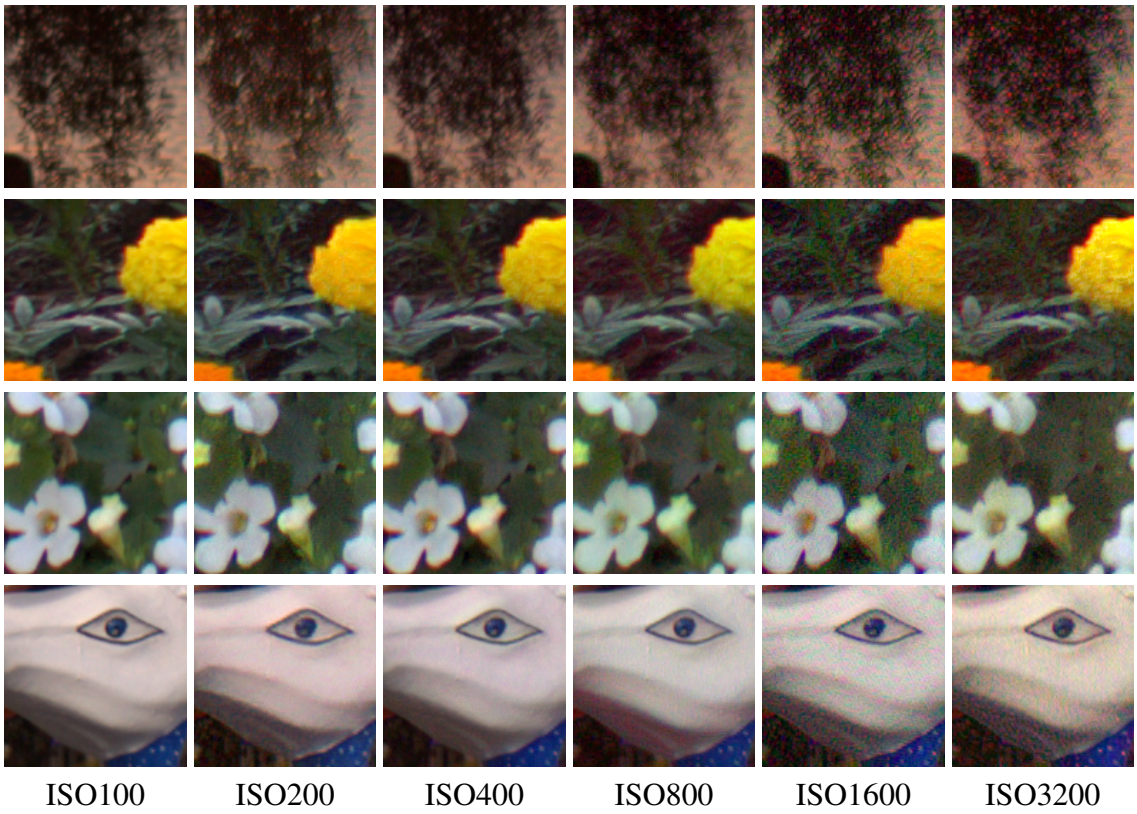


Figure 4.8: Photographs mapped from ISO 1600 to ISO 100 using our technique. Zoom in to see the reduced noise in the synthetic versions.

pairs of ISO values.

Fig. 4.7 illustrates the use of our technique to adjust the noise levels of several photographs taken at ISO 100 to match different ISO values, from 200 to 3200. Fig. 4.8 shows the use of our method to reduce the noise level of five photographs originally taken with ISO 1600 to ISO 100. We encourage the reader to zoom in for better visualization of the increased and reduced noise.

4.6 Evaluation

To evaluate our method we use the small version of SIDD (ABDELHAMED; LIN; BROWN, 2018), which consists of paired noisy-clean images captured under different ISO values with five different smartphone cameras: LG G4 (G4), Google Pixel (GP), iPhone 7 (IP), Motorola Nexus 6 (N6), and Samsung Galaxy S6 Edge (S6). Having a paired dataset allows us to compare our results both quantitatively and qualitatively. It also enables the computation of the standard deviation of the noise for each ISO value, which is used to obtain a fair implementation of AWGN. Moreover, since Noise Flow (ABDELHAMED; BRUBAKER; BROWN, 2019) was trained in such dataset, the use of SIDD also allows for a fair comparison with this method.

Section 4.6.1 describes the five noise models compared to ours. In Section 4.6.2 we use a discriminative model trained to classify noise between natural and synthetic (produced by previous models). Section 4.6.3 presents another classifier trained to distinguish among natural noise and the several noise models described in Section 4.6.1, including ours. Finally, Section 4.6.4 compares the performance of our technique against the most competitive ones using KL divergence and KS test as objective metrics. Such experiment measures the similarities between each model's noise distributions and the noise distributions of actual photographs taken at the target ISO levels (ground truth). We also qualitatively compare results obtained with these three techniques for different ISO values and camera models.

4.6.1 Noise Models

We call an algorithm used to corrupt a clean patch a *noise model*. We consider five popular/recent noise models for comparisons against ours:

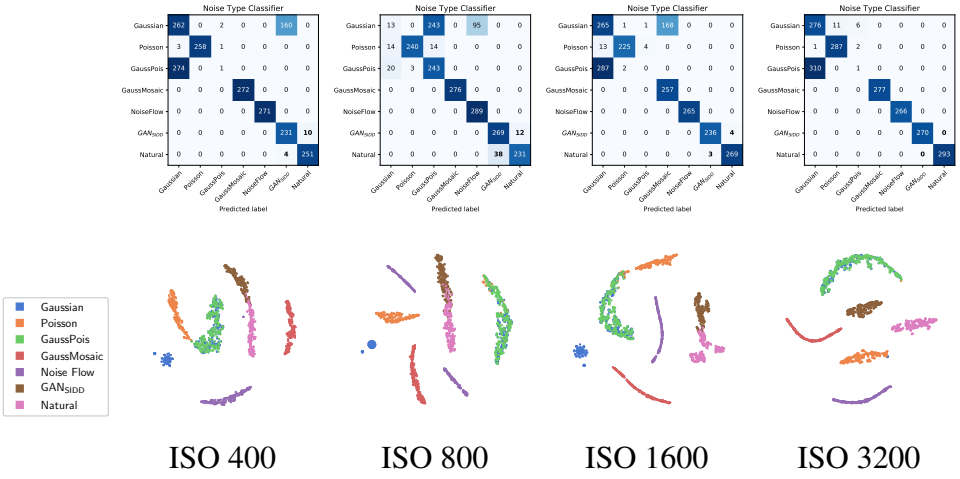
AWGN: the most traditional noise model, consisting of zero-mean Gaussian noise[1]. By having paired noisy-clean patches from the SIDD, we compute the average noise-variance for each ISO value. Such value is used when synthesizing noise using AWGN for the corresponding ISO level;

Poisson: another traditional noise model, synthesizes noise following a Poissonian distribution[1];

GaussianPoissonian: a composition of the two previous noise models (AWGN+Poisson), inspired by (FOI et al., 2008). First, we apply AWGN to the input, then apply Poisson

[1]Generated with the *random_noise* function from the python package *skimage*.

Figure 4.9: Classification of images corrupted by several noise models, including our GAN_{SIDD} trained on the small SIDD dataset. We trained one classifier for each ISO level. (top) Confusion matrices obtained with a 2,000-patch testset for each ISO level, including multiple camera models per ISO. These matrices exhibit noticeable confusion between natural noise and our GAN_{SIDD} noise, specially for the ISO levels 800. (bottom) Projections using t-SNE for the same test sets. The projections of the noise generated by our GAN_{SIDD} overlap with natural for most ISO values.



noise to the intermediate result.

AWGN followed by Bayer sampling and demosaicing: consisting of the application of AWGN followed by sampling (simulating a Bayer color filter array – CFA) and demosaicing (using bilinear interpolation). A similar noise model is used by CBDNet (GUO et al., 2019);

Noise Flow (ABDELHAMED; BRUBAKER; BROWN, 2019): a deep learning based model for synthesizing noise for different types of cameras and ISO values. As the provided models were trained in raw space, we synthesize (and add) noise in raw before converting the images to sRGB.

We refer to our models trained on the small SIDD dataset as $\textbf{GAN}_{\textbf{SIDD}}$. Although SIDD provides noisy-clean pairs, the training of our generative models was unsupervised (*i.e.*, did not use such paired information). The small version of the SIDD dataset contains 160 pictures, and only their most representative ISO values were used for training our models: ISO 400 (17 pictures), ISO 800 (36 pictures), ISO 1600 (22 pictures), and ISO 3200 (13 pictures). This resulted in 88 images and a total of 18,951 256×256 training patches distributed over these four ISO levels and five camera models. We trained one GAN_{SIDD} for each combination of ISO level and camera model. Training one model per ISO value regardless of camera model results in *mode collapsing* (GOODFELLOW et al., 2014), where the generator learns to mimic only one camera model. Each model was

trained for 40 epochs using 128×128 random crops from the training patches (to expedite training).

Except for Noise Flow, the remaining algorithms operate directly in the sRGB space. For each scene, SIDD provides a so called *clean image*, obtained after processing 150 pictures taken from the same scene (ABDELHAMED; LIN; BROWN, 2018). For the comparisons described in the following sections, all noise models take as input one 256×256 *clean patch* at a time. Their outputs are used as input to discriminative models, and the images and noise distributions are compared to patches from actual photographs (ground truth) taken at the target ISO level.

4.6.2 Classifying Natural and Artificial Noise

An evaluation network was trained for classifying a given input into one of the two classes: (i) corrupted with natural noise, or (ii) corrupted by traditional noise models (*i.e.*, AWGN, Poissonian, GaussianPoissonian, and AWGN followed by Bayer sampling and demosaicing). During training, each patch had a 20% probability of containing natural noise (*i.e.*, taken directly from an actual noisy photography in the SIDD dataset). There was also a 20% probability that a training patch is obtained by corrupting a clean patch with each one of the four noise models (adding up to the remaining 80%). Details of the architecture of this binary-classification network can be found in Appendix A.2. The obtained classifier achieved an accuracy above 99.75% on the test set on all ISO levels. This simple experiment shows how easily separable the sets of images corrupted by natural and by artificially-generated noise are. It also shows how *current synthetic noise models fail to mimic natural noise found in digital photographs*.

Table 4.1: KL divergences and KS values for different noise models, ISO values, and lighting conditions ([L]ow and [N]ormal light brightness levels). Best (lower) values are highlighted in bold. These values were measured over the whole population of image patches (see the text for details). Notice how our method achieves better values for these metrics across all ISO values.

		ISO 400		ISO 800		ISO 1600		ISO 3200	
		L	N	L	N	L	N	L	N
KL divergence	AWGN	0.0910	0.1063	0.0938	0.1108	0.1780	0.1478	0.0765	0.0796
	AWGN+Poisson (linear)	0.1358	0.0781	0.2053	0.1010	0.0732	0.1786	0.0784	0.0383
	NoiseFlow	0.1052	0.1557	0.0593	0.1140	0.0517	0.0634	0.0801	0.0520
	GAN_{SIDD}	**0.0091**	**0.0323**	**0.0104**	**0.0249**	**0.0228**	**0.0129**	**0.0339**	**0.0188**
KS value	AWGN	0.0693	0.0787	0.0842	0.0909	0.1096	0.1237	0.1100	0.0896
	AWGN+Poisson (linear)	0.1000	0.0668	0.1520	0.0726	0.0937	0.1277	0.0784	**0.0628**
	NoiseFlow	0.1138	0.1422	0.0900	0.1057	0.0929	0.0821	0.1270	0.0925
	GAN_{SIDD}	**0.0333**	**0.0564**	**0.0404**	**0.0550**	**0.0600**	**0.0643**	**0.0761**	0.0677

4.6.3 Discriminating among Noise Models

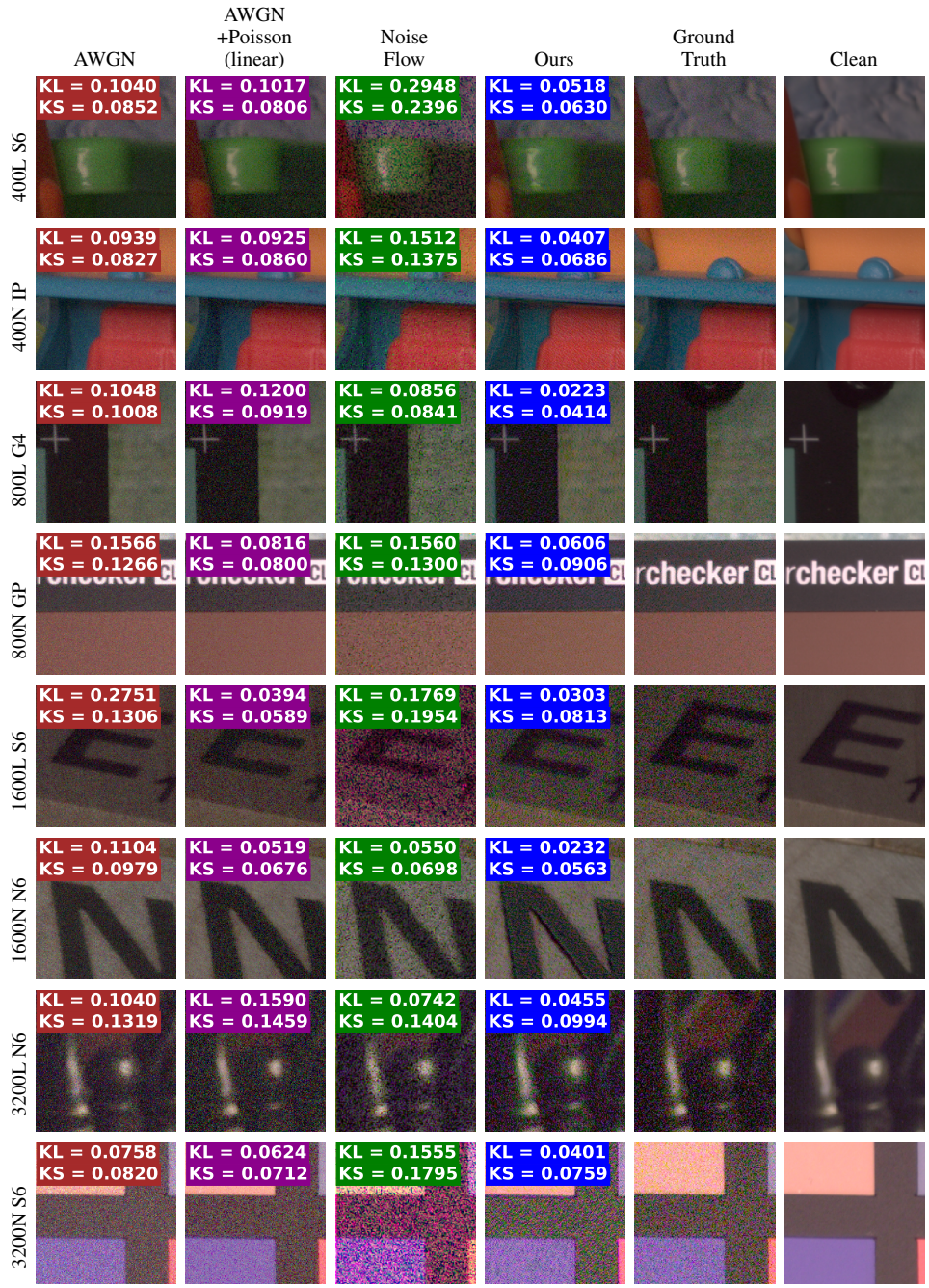
We also trained classifiers for discriminating images containing natural noise as well as ones corrupted by the six noise models described in Section 4.6.1, which include ours. We trained a single classifier for each ISO level (400, 800, 1600, and 3200), regardless of the used cameras models. Although the trained classifiers are capable of discriminating between natural and our GAN-generated noise, we show that the noise produced by our generative models is much closer to natural than the ones created by previous noise models. Fig. 4.9 shows the results of these classifiers. The confusion matrices on the top were created with 2,000 test patches for each ISO level. Notice that the confusion between natural and our GAN generated noise is higher than with the previous noise generators. The t-SNE visualizations at the bottom of Fig. 4.9 show the output of the last layer of each classifier. t-SNE is a visualization technique that uses a non-linear transformation to project the data onto a 2D plane, trying to keep the pairwise distance between samples (MAATEN; HINTON, 2008a). Note that the projections of other noise models are further away than ours (brown) from natural noise (pink). The projections of the noise generated by our GAN_{SIDD} overlap with natural for most ISO values. The labels of each sample (AWGN, Poisson, natural, etc.) were not used for learning the t-SNE projections, but only for coloring the visualizations *after* t-SNE has been applied.

The use of classifiers is a known strategy for measuring the quality of generative methods (LOPEZ-PAZ; OQUAB, 2017). According to Lopez-Paz and Oquab, the lower the classifier's accuracy when discriminating between natural and synthesized samples, the higher the quality of the generative model. It is important to mention that when evaluating trained generative models (LOPEZ-PAZ; OQUAB, 2017), all experiments got near-perfect results (real and synthetic samples being easily distinguishable). In a similar experiment presented in this section (Fig. 4.9), the classifier failed to distinguish our results from natural noise in several occasions.

4.6.4 Quantitative Metrics

For quantitative evaluation, we compared patches containing natural noise with synthesized ones produced using AWGN, AWGN+Poisson, Noise Flow, and our GAN models. We use the KL divergence and KS test as objective metrics for assessing the similarity of the noise distributions generated by each technique with the noise distributions

Figure 4.10: Additional comparisons of synthesized noise obtained by corrupting *clean patches* for different ISO values and camera models. Our results achieved the best (smaller) KL divergence and KS values. Ground truth is an actual photograph taken at the target ISO level. The clean images are provided as part of the SIDD dataset.



obtained from natural noise. Lower values for KL and KS represent better results (more similar to real noise).

Fig. 4.10 shows additional quantitative and qualitative comparisons (besides the ones in Fig. 4.2) of patches from different ISO values and lighting conditions (L for low and N for normal light-brightness levels) from the small SIDD datataset. The variances used for AWGN and AWGN+Poisson-linear are mean variances measured in sRGB (post-gamma)

and in linear space (obtained by applying inverse gamma to sRGB), respectively. For each ISO level, the variances were computed from all noisy-clean image pairs for the given ISO value. Noise Flow were applied in raw space, with metadata (camera and ISO value) taken from the patch[2], using a trained model provided by the authors. AWGN+Poisson-linear was applied in linear space. Both AWGN and our GAN_{SIDD} models are applied directly in the sRGB space. Both metrics (KL divergence and KS value) are computed in the sRGB space.

Note in Fig. 4.10 how our model achieves smaller values of KL divergence, despite of not using paired data at all. Table 4.1 compares these metrics, for the most competitive methods, for the whole population of patches from SIDD. The metrics were obtained by computing the Red, Green and Blue histograms of all patches as a group, resulting in three numbers per population (KL_R, KL_G, KL_B), which are averaged together. The same procedure is performed for KS values. For more details please refer to Appendix A.3. Appendix A.4 provides a similar Table comparing additional methods. As seen in Table 4.1, our approach obtains the best metrics across practically all scenarios, attesting to its effectiveness in synthesizing noise that is statistically similar to natural noise. Appendix A.5 provides more comparison cases for visual inspection.

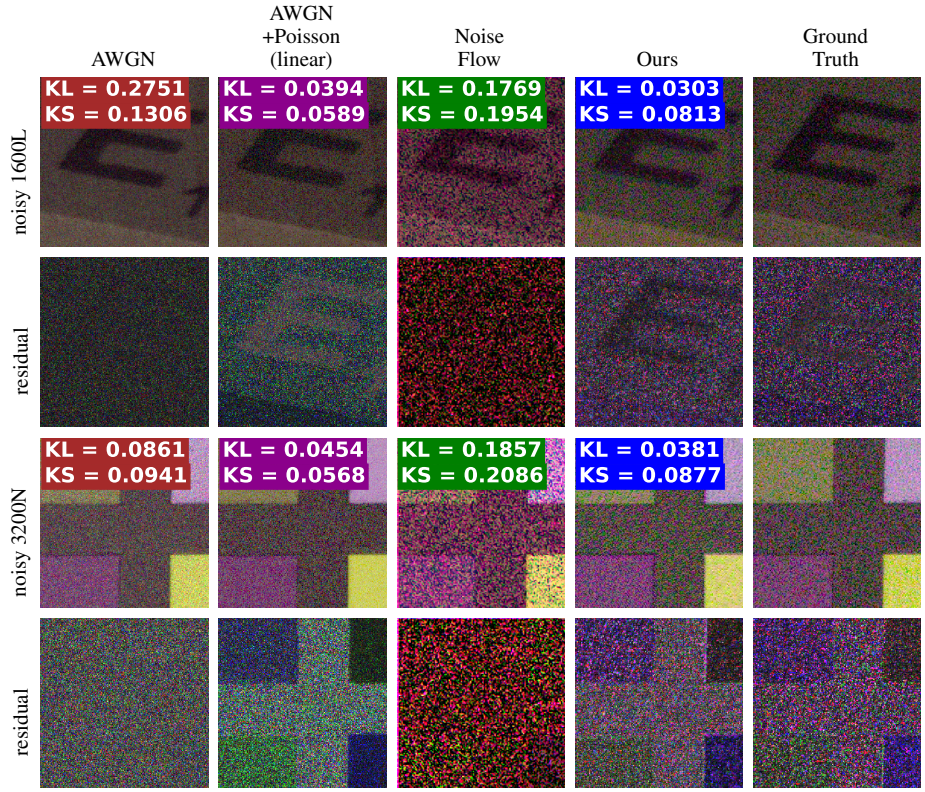
Fig. 4.11 compares the synthesized noise (residual), *i.e.*, the difference between the noisy and clean image versions, generated by the noise models in Fig. 4.10. Notice how our method better mimics camera noise, specially regarding the size of noise grain and color distribution. Appendix A.6 provides additional examples of residual images.

4.7 Applications

To demonstrate potential applications of our method, we have used our trained models for synthesizing noise for training a denoising algorithm, and a CFA-design along with demosaicing method robust to noise (similar to what was trained in Chapter 3). We show that the noise synthesized by our method improves the performance of both applications in benchmarks of natural images.

[2]Our statistical metrics are computed in sRGB space, and we noticed that the raw-to-sRGB postprocessing implemented by Noise Flow differs from the one implemented by SIDD. To be fair in our comparisons, the metrics for Noise Flow were computed against the clean sRGB images provided in the Noise Flow package.

Figure 4.11: Comparison of synthesized noisy images and corresponding noise (residual) produced by the various methods. For better visualization, contrast of the residual images has been enhanced by factor of $3\times$. Our results better mimic the noise found in digital photographs.



4.7.1 Denoiser

The first application is the use of our method for training denoising models. In this experiment, we trained several versions of the DnCNN method for blind denoising (ZHANG et al., 2017). We have used the authors' implementation, using the DnCNN-S architecture, which consists of 17 Conv2D layers with 64 filters each. Each version of the DnCNN denoiser is trained on an extensive noisy-clean paired dataset created with a different realization of our GAN. More precisely, we select a set of clean (noise-free) images from the dataset of Gharbi et al. (GHARBI et al., 2016), and compute the corresponding noisy images using our GAN_D noise generator (for several ISO values). The subscript D indicates on which (unpaired) dataset the GAN_D noise model was trained: $\text{D} \in \{\text{our\_T3i}, \text{SIDD}\}$. Although we describe a denoising application, in principle, our method could be used to generate input to any trainable computer-vision task that seeks robustness to noisy scenarios.

To compare our noise generator against existing alternatives, we also trained several DnCNN denoisers using paired datasets created by the AWGN and Poissonian-Gaussian

Table 4.2: PSNR results of evaluating a denoiser (DnCNN) trained using noise generated by existing techniques, and by using the generators trained with our Canon dataset, and with SIDD dataset. The version of DnCNN trained using images corrupted by both GANs (trained with Canon and SIDD datasets) achieved higher PSNR (in bold) in all natural-image benchmarks.

		Renoir			Darmstadt	SIDD	
id	Noise Model	T3i	Mi3	S90		Low lighting	Normal lighting
1	CBDNet (GUO et al., 2019)	27.35	25.08	26.62	32.58	24.48	28.99
2	AWGN	29.12	25.07	28.91	30.63	27.07	29.31
3	AWGN-linear	29.64	25.48	29.48	32.13	29.17	31.20
4	PoisGauss	29.61	25.18	29.31	31.15	27.30	29.71
5	PoisGauss-linear	30.02	25.94	29.70	32.11	29.93	31.67
6	Noise Flow	30.62	26.15	29.93	32.84	29.53	31.68
7	GAN_{our_T3i}	33.41	28.03	31.55	32.24	30.32	31.48
8	GAN_{SIDD}	32.16	27.38	31.04	32.79	31.85	33.02
9	GAN_{our_T3i} + GAN_{SIDD}	**33.49**	**28.14**	**31.69**	**35.32**	**33.27**	**34.67**

models (both applied in linear and post-gamma spaces), CBDNet (GUO et al., 2019) and Noise Flow (ABDELHAMED; BRUBAKER; BROWN, 2019) (for these last two, we used implementations provided by their authors). For all experiments, we trained the denoisers past convergence: taking the checkpoint with highest PSNR on the validation set. We used the same data-augmentation in the training of all models (see Section 4.7.3 for details).

Table 4.2 summarizes the results of the DnCNN denoisers trained with our GAN noise models versus the alternatives. We evaluate each denoiser using the Renoir (ANAYA; BARBU, 2014), Darmstadt (PLOTZ; ROTH, 2017), and SIDD (ABDELHAMED; LIN; BROWN, 2018) denoising benchmarks. In these, we discriminate each camera model in the Renoir dataset, as well as between scenes captured under low (L) and normal (N) light brightness levels in the SIDD dataset. As can be seen by the higher PSNR values in Table 4.2, using our GAN noise models lead to improved performance of the DnCNN denoiser across all benchmarks.

Compared to previous noise generators (rows 1 to 6 in Table 4.2), the denoiser trained only with noisy images generated by GAN_{our_T3i} (row 7 in Table 4.2) got a significant improvement in PSNR performance, especially in the Renoir dataset. Similarly, when trained only with GAN_{SIDD} (row 8 in Table 4.2), the denoiser also performed significantly better than previous approaches, in particular on the SIDD dataset. Intriguingly, a more generic denoiser trained using randomly selected patches generated by GAN_{our_T3i} and GAN_{SIDD} got the overall best results (row 9 in Table 4.2). We conjecture that the combined use of the two models might help to fight overfitting, thus explaining the improved performance. Indeed, while the denoisers separately trained with GAN_{our_T3i} and GAN_{SIDD}

Table 4.3: PSNR results of evaluating the CFA+demosaicing architecture (Chapter 3) trained using noise generated by existing techniques, and by using the generators trained with our Canon dataset, and with SIDD dataset. The versions trained using images corrupted by both GANs (trained with Canon and SIDD datasets) achieved higher PSNR (in bold) in all natural-image benchmarks (with exception to the Renoir S90).

		Renoir			Darmstadt	SIDD	
id	Noise Model	T3i	Mi3	S90		Low lighting	Normal lighting
1	No noise-model	30.51	26.22	30.30	31.31	27.76	30.01
2	CBDNet (GUO et al., 2019)	31.20	27.11	30.48	34.82	27.18	31.07
3	AWGN	34.21	28.59	33.31	33.87	30.35	32.63
4	AWGN-linear	32.49	28.13	32.22	35.44	32.70	34.48
5	PoisGauss	34.15	28.48	**33.37**	33.82	30.34	32.64
6	Noise Flow	32.58	28.50	31.63	34.74	31.49	33.72
7	GAN_{our_T3i}	34.66	29.08	32.66	34.08	31.27	33.04
8	GAN_{SIDD}	**34.79**	**30.02**	32.89	35.65	33.46	34.56
9	GAN_{our_T3i} + GAN_{SIDD}	34.50	29.17	32.47	**35.66**	**33.71**	**35.23**

both converged between epochs 3 and 4, the denoiser trained with both noise models (GAN_{our_T3i} + GAN_{SIDD}) converged after epoch 5.

4.7.2 CFA Design and Demosaicing

In Chapter 3, we compared our CFA patterns and demosaicing models against existing methods in the presence of Gaussian noise. By using the generative model proposed in this Chapter, we can train CFA+demosaicing models that are robust to natural noise, evaluating the performance in the benchmark datasets used in the denoiser experiment (Section 4.7.1).

This experiment is similar to the comparison of trained denoisers in Section 4.7.1, but instead of training a DnCNN, we trained several versions of the CFA+demosaicing architecture (presented in Chapter 3), using a fixed 4x4 CFA pattern. Similarly, we also used the dataset of Gharbi et al. (GHARBI et al., 2016) as clean images, and compute the corresponding noisy images using the same methods described in Section 4.7.1.

Table 4.3 summarizes the results of the CFA+demosaicing architectures trained with our GAN noise models versus the alternatives. We evaluate each trained version by using the same natural-image datasets of the first experiment: Renoir (ANAYA; BARBU, 2014), Darmstadt (PLOTZ; ROTH, 2017), and SIDD (ABDELHAMED; LIN; BROWN, 2018). The models trained with our generative methods achieved higher PSNR in all datasets, except for the Renoir-S90, a camera that our generative method did not see during training.

Figure 4.12: Learned CFA patterns when using different noise models. In top row, we can see the learned patterns when using traditional noise models, as well as using no noise model. In bottom, the learned patterns when using most recent noise models and ours.

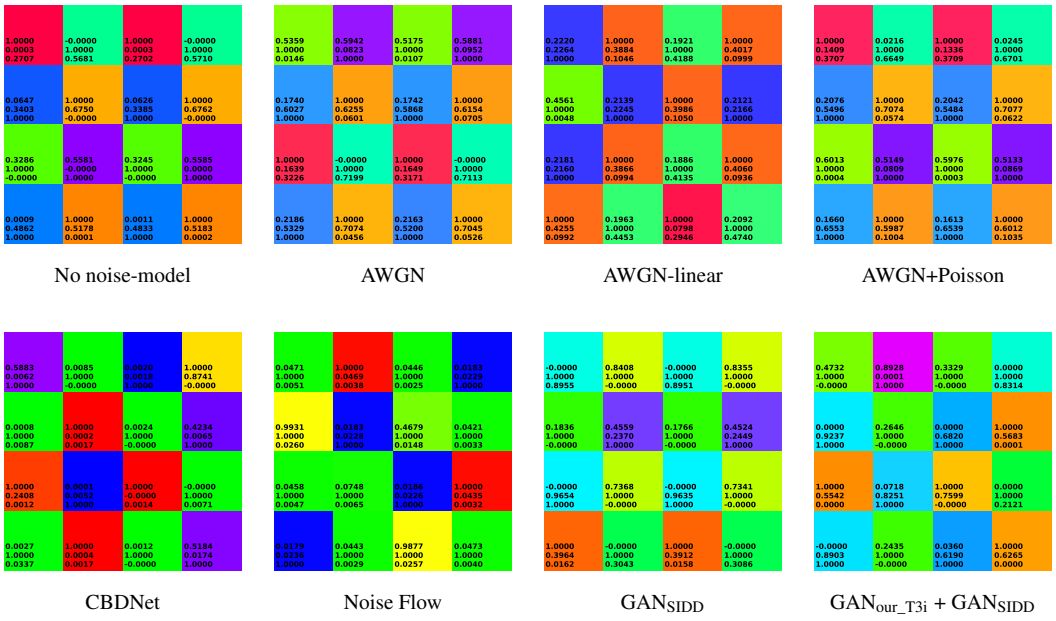


Fig. 4.12 shows the CFA patterns encountered in some of the trainings. It is easy to notice that all CFA patterns trained with AWGN-based noise models (top row), as well as the 'no noise-model', share similar color patterns (with exception to the AWGN in linear space). It is not by chance, as AWGN corrupts all color channels with the same variance. But it is already known that this is not how natural noise behaves (PLOTZ; ROTH, 2017). Also, when training using noise-models based on the SIDD (bottom row, except for CBDNet), the learned patterns exhibit many greenish color-filters, which indicates that such dataset might consist of many green-predominant images. Nonetheless, the pattern encountered after training using our GAN$_{our_T3i}$ + GAN$_{SIDD}$ noise model has quite different colors, which are presumably more robust to (natural) noise.

4.7.3 Data Augmentation on Application Trainings

For all experiments described in Section 4.7, besides using random 90^o rotations, we also augmented the training dataset using patches defined as

$$\alpha\, x_{noisy} + (1 - \alpha)\, y_{clean},$$

where x_{noisy} is a noisy patch synthesized by the noise model, y_{clean} is the corresponding clean image (noise free), and α is an interpolation parameter whose value is randomly

selected from a uniform distribution in $[0.7, 1]$ (values chosen empirically). Despite its simplicity, such strategy helped to improve the trained-models' PSNR around 2 dB by delaying overfitting (our optimizer was able to train an additional epoch when using such augmentation).

4.8 Discussion

The classifiers for identifying the ISO level of a given patch (Section 4.5.2), for distinguishing natural from artificial noise (Section 4.6.2), and for noise-model classification (Section 4.6.3) all share similar architectures. The architectures of the GAN generators (Section 4.4.5) and of these classifiers were obtained after experimenting with different designs (*e.g.*, residual network (HE et al., 2016), U-Net (RONNEBERGER; P.FISCHER; BROX, 2015)) and different deep-learning layers (*e.g.*, batch-normalization (IOFFE; SZEGEDY, 2015), half-strided convolutions, and instance normalization (ULYANOV; VEDALDI; LEMPITSKY, 2016)). We have also tried different metrics for the cycle-consistency loss (Section 4.4.2), such as l_1 and l_2 norm, and SSIM.

Using a residual design (Eq. (4.6)) for our generators greatly improved the convergence of the training and the quality of the results. In this same residual design, we include a batch-normalization as the last layer of the generator, initializing Γ (the scaling parameter) as 0.1, making Eq. (4.6) to be close (at initialization) to an identity function. This actually speeds up the training convergence, while reducing generated artifacts. As mentioned in Section 4.4.1, we replaced the traditional adversarial loss by the least-squares adversarial loss, as this has been reported to increase training stability (MAO et al., 2016). While the *tanh* activation function is often used as the last layer of generators, for our application it tends to saturate highlights and darken low-lit regions. Removing *tanh* improved our results.

Adding noise to raw image values more closely simulates most noise sources, besides being simpler to implement compared to simulating these processes in demosaiced, white-balanced, quantized, and compressed images. However, doing so would preclude our method from being used with images post-processed by such transformations (such as JPEG, PNG, etc.), which are used by most applications. Therefore, we perform all training in the sRGB color space. This was a design decision where we favored wider applicability over slightly better precision. Nonetheless, by retraining our models for input/output raw pixel values, our method can be used with RAW images.

A key factor for AWGN-based noise-models is the choice of the variance. In our KL/KS comparison, as well as in the denoiser experiment, the variance for AWGN was computed for each ISO level using the paired noisy-clean images from SIDD[3]. When such paired dataset is unavailable, the variance must be estimated or guessed, resulting in a poorer representation of the natural noise. Our method, on the other hand, does not rely on paired datasets, thus, the gap between results obtained with our and AWGN- based methods should increase in practical applications.

Training GANs is hard not only due to problems during training (*e.g.*, mode collapsing, non-convergence, diminished gradients, and sensibility to hyperparameters), but specifically to potential artifacts introduced by image-generative models, like ringings and blurring. The fact that our GAN architecture is capable of generating high-frequency noise illustrates its potential. Nonetheless, color shifts and halos may happen in some situations. The cause of these artifacts are not quite clear, and its understanding and correction is a subject for future investigation. Despite such artifacts, our model provides a better alternative to existing noise generators, as shown in Table 4.2.

Noise encountered in photographs of different camera models might exhibit different noise statistics (ABDELHAMED; LIN; BROWN, 2018). We show how our GAN model can be trained for generating noise mimicking multiple camera models (*e.g.*, Canon T3i and several smartphone cameras from SIDD). We show that the performance of a denoiser trained using such networks is greatly improved on several benchmarks, which demonstrates the potential and generalization of our method. For instance, the denoiser trained with $\text{GAN}_{\text{our\_T3i}}$ + GAN_{SIDD} got improved PSNR on cameras Mi3 and S90 (row 9 in Table 4.2), even when our generative models were not trained with images from such cameras.

We have demonstrated that our generative models can be trained in an unsupervised way using small, unpaired datasets. Thus, it can be easily applied to other camera models. Its main limitation is the mode collapsing observed when training a single generator for several camera models. While we have dealt with this issue by training one generator for each camera model, ideally this problem should be addressed directly in the GAN architecture. This is the subject of future investigation.

[3]We computed the variance for the post-gamma and linear spaces.

4.9 Summary

This Chapter presented a practical data-driven technique for adjusting the noise level of input photographs to match target ISO levels. Our solution learns the mappings among different ISO levels from unpaired data using GANs, for which we defined a new loss formulation and network architectures tailored to the problem. An ablation study justifies our decisions, confirming the superior results achieved when using our method. By not requiring a paired noisy-clean dataset, our technique can be easily trained for specific camera models by just collecting photographs for the various ISO levels.

We demonstrate the effectiveness of our approach both quantitatively and qualitatively, by demonstrating its superior performance over previous methods. As a practical application, we have shown that images generated by our technique greatly improve the performance of a state-of-the-art trainable denoiser.

Several applications can benefit from our technique. These include not only trainable denoising methods, but also demosaicing (GHARBI et al., 2016; HENZ; GASTAL; OLIVEIRA, 2018) and image-reconstruction (MANDAL; BHAVSAR; SAO, 2017), forgery detection (KAUR; WALIA, 2016; SCHETINGER et al., 2017), as well as computer-vision tasks seeking noise robustness (DODGE; KARAM, 2016; DIAMOND et al., 2017; ROY et al., 2018).

5 CONCLUSION AND FUTURE WORK

This dissertation presents two novel techniques that improve core aspects of the digital-photography pipeline. Chapter 3 describes a technique to improve acquisition of color images by concurrently designing CFA and demosaicing methods that yield better color reconstruction. We show how to train architectures that can be used to capture and demosaic both noiseless and/or noisy images, as well as to capture NIR information along with RGB. Our trained models surpass state-of-the-art techniques for all those cases, in all datasets (and all noise intensities).

Chapter 4 addresses the problem of noise by introducing a model capable of generating noise more similar to whats is found in digital photographs (natural noise). Our technique learns the mapping among image domains (each one representing an ISO level), being capable of adjusting the noise level accordingly. By using GANs, our technique can be trained with unpaired datasets, easing the process of creating new training datasets for other cameras. Through discriminative-evaluation techniques and quantitative analysis (using KL divergence and the KS test), we show that the images synthesized by our method are closer to real photographs when compared to the ones synthesized by previous methods. We also demonstrate direct applications of our method, as the training of a denoiser; whose performance improved in all benchmark datasets when trained with noisy images synthesized using our models.

The presented techniques have the potential to improve the overall quality of digital photographs. The embedding of such solutions in actual cameras (either DSLR or smartphone ones) is an area for future work. The proposed CFAs could be implemented using band-pass filters produced by companies as SILIOS Technologies, with COLOR SHADES® (SILIOS Technologies, 2017). For the demosaicing step, recent light-weight CNN architectures, like MobileNet (HOWARD et al., 2017), could be be used to run our models. The same analogy holds for the noise work: by training a MobileNet denoiser using images corrupted by our synthetic noise, it is straightforward to port such denoiser to smartphones. Regarding regular digital cameras (*e.g.*, DSLR), it is reasonable to speculate that companies may soon incorporate deep-learning acceleration hardware, as it is already found in current generation of smartphones.

5.1 Future Work

Some ideas for further exploration include the design of CFAs for capturing higher dynamic range (Section 5.1.1); the improvement of our noise generative model by using paired noisy-clean samples (Section 5.1.2); and the training of noise models to generate noise given a noise-intensity input (Section 5.1.3).

5.1.1 Neutral Density Filter Arrays

One of the limitations of actual cameras, even professional ones, is the narrow dynamic range they can capture in a single shot. In order to capture and visualize a larger dynamic range, one can combine bracketed captures (photos captured with different exposure) or, as proposed by Hasinoff et al. (HASINOFF et al., 2016), fuse a burst of photos to create an HDR picture. We believe that our CFA-design autoencoder could be adapted for obtaining an additional filter array capable of acquiring higher-dynamic range with a single capture. Such idea was already proposed by Yasuma et al. (Yasuma et al., 2010), but while their work use pre-defined transmittance levels, we believe that a learning algorithm could be used for finding those levels. Thus, in addition to the CFA, one could use an array of neutral density filters (dubbed NDFA) that would learn different transmittance levels to sparsely sample the scene radiance, from which a reconstructing algorithm would recover a high-dynamic range image. The entire pipeline (NDFA+CFA+demosaicing) would be trained in an end-to-end fashion.

5.1.2 Paired Samples during Noise-Generative Training

Given that there are few paired noisy-clean datasets, all of them being small (a few hundreds of images), we had opt for a GAN scheme that allows training with unpaired dataset. Nonetheless, such paired samples could be used to further guide the training process. We made some experiments trying to use such paired images, with no noticeable improvement. But as soon as more paired datasets become available, we believe that such a semi-supervised learning will greatly improve the results.

5.1.3 Noise Intensity as Input Feature to Generative Model

We opted to use ISO levels for defining the domains our model learned to map from/to, but the use of noise variance (*i.e.*noise intensity) to parametrize the noise generation is preferable. Unfortunately, we can only precisely assess this information using paired noisy-clean datasets, which, unfortunately, are scarce. Similarly to the idea discussed in Section 5.1.2, as paired datasets become available, this is a front that should be explored.

REFERENCES

ABDELHAMED, A.; BRUBAKER, M. A.; BROWN, M. S. Noise flow: Noise modeling with conditional normalizing flows. In: **IEEE ICCV**. [S.l.: s.n.], 2019.

ABDELHAMED, A.; LIN, S.; BROWN, M. S. A high-quality denoising dataset for smartphone cameras. In: **IEEE CVPR**. [S.l.: s.n.], 2018.

ALJADAANY, R.; PAL, D. K.; SAVVIDES, M. Douglas-rachford networks: Learning both the image prior and data fidelity terms for blind image deconvolution. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2019. p. 10235–10244.

ALLEYSSON, D.; SUSSTRUNK, S.; HERAULT, J. Linear demosaicing inspired by the human visual system. **TIP**, v. 14, n. 4, p. 439–449, 2005. ISSN 1057-7149.

ALTER, F.; MATSUSHITA, Y.; TANG, X. An intensity similarity measure in low-light conditions. In: ____. **ECCV**. [S.l.: s.n.], 2006. p. 267–280. ISBN 978-3-540-33839-0.

ANAYA, J.; BARBU, A. Renoir - a dataset for real low-light noise image reduction. **arXiv preprint arXiv:1409.8230**, 2014.

ARCHAMBAULT, M. **A Brief History of Color Photography, From Dream to Reality**. 2015. Available from Internet: <https://petapixel.com/2015/10/11/a-brief-history-of-color-photography-from-dream-to-reality/>.

ARCHAMBAULT, M. **Film vs. Digital: A Comparison of the Advantages and Disadvantages**. 2015. Available from Internet: <https://petapixel.com/2015/05/26/film-vs-digital-a-comparison-of-the-advantages-and-disadvantages/>.

ARJOVSKY, M.; CHINTALA, S.; BOTTOU, L. Wasserstein generative adversarial networks. In: . PMLR, 2017. (Proceedings of Machine Learning Research, v. 70), p. 214–223. Available from Internet: <http://proceedings.mlr.press/v70/arjovsky17a.html>.

BAHAT, Y.; MICHAELI, T. Explorable super resolution. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2020.

BAI, C. et al. Automatic design of color filter arrays in the frequency domain. **TIP**, v. 25, n. 4, p. 1793–1807, 2016. ISSN 1057-7149.

BARBU, A. Training an active random field for real-time image denoising. **IEEE TIP**, v. 18, n. 11, p. 2451–2462, 2009. ISSN 1057-7149.

BAYER, B. **Color imaging array**. 1976. US Patent 3,971,065. Available from Internet: <https://www.google.com/patents/US3971065>.

BERTHELOT, D.; SCHUMM, T.; METZ, L. Began: Boundary equilibrium generative adversarial networks. **arXiv preprint arXiv:1703.10717**, 2017.

BIANCO, S.; CUSANO, C.; SCHETTINI, R. Single and multiple illuminant estimation using convolutional neural networks. **CoRR**, abs/1508.00998, 2015. Available from Internet: <http://arxiv.org/abs/1508.00998>.

BROOKS, T. et al. Unprocessing images for learned raw denoising. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2019. p. 11036–11045.

BUADES, A.; COLL, B.; MOREL, J. M. A non-local algorithm for image denoising. In: **IEEE CVPR**. [S.l.: s.n.], 2005. v. 2, p. 60–65 vol. 2. ISSN 1063-6919.

BUADES, A.; COLL, B.; MOREL, J.-M. Nonlocal image and movie denoising. **IJCV**, v. 76, n. 2, p. 123–139, 2008. ISSN 1573-1405.

BUADES, A. et al. Self-similarity driven color demosaicking. **IEEE Transactions on Image Processing**, v. 18, n. 6, p. 1192–1202, June 2009. ISSN 1057-7149.

BURGER, H. C.; SCHULER, C. J.; HARMELING, S. Image denoising: Can plain neural networks compete with bm3d? In: **IEEE CVPR**. [S.l.: s.n.], 2012. p. 2392–2399. ISSN 1063-6919.

BYLINSKII, Z. et al. Where should saliency models look next? In: LEIBE, B. et al. (Ed.). **Computer Vision – ECCV 2016**. Cham: Springer International Publishing, 2016. p. 809–824. ISBN 978-3-319-46454-1.

CAI, B. et al. Dehazenet: An end-to-end system for single image haze removal. **CoRR**, abs/1601.07661, 2016. Available from Internet: <http://arxiv.org/abs/1601.07661>.

CAO, J. et al. D2det: Towards high quality object detection and instance segmentation. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2020.

CHAKRABARTI, A. Learning sensor multiplexing design through back-propagation. **CoRR**, abs/1605.07078, 2016.

CHAKRABARTI, A.; FREEMAN, W. T.; ZICKLER, T. Rethinking color cameras. In: **ICCP**. [S.l.: s.n.], 2014. p. 1–8.

CHAMPANDARD, A. J. Semantic style transfer and turning two-bit doodles into fine artworks. **CoRR**, abs/1603.01768, 2016. Available from Internet: <http://arxiv.org/abs/1603.01768>.

CHANG, H. et al. Palette-based photo recoloring. **ACM Transactions on Graphics (Proc. SIGGRAPH)**, v. 34, n. 4, jul. 2015.

CHEN, C. et al. Learning to see in the dark. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2018. p. 3291–3300.

Chen, G.; Zhu, F.; Heng, P. A. An efficient statistical method for image noise level estimation. In: **IEEE ICCV**. [S.l.: s.n.], 2015. p. 477–485. ISSN 2380-7504.

CHEN, X. et al. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. **CoRR**, abs/1606.03657, 2016. Available from Internet: <http://arxiv.org/abs/1606.03657>.

CHEN, X. et al. Microsoft COCO captions: Data collection and evaluation server. **CoRR**, abs/1504.00325, 2015. Available from Internet: <http://arxiv.org/abs/1504.00325>.

CHEN, Y.; POCK, T. Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration. **IEEE TPAMI**, v. 39, n. 6, p. 1256–1272, 2017. ISSN 0162-8828.

CHEN, Y. et al. Revisiting loss-specific training of filter-based mrfs for image restoration. In: WEICKERT, J.; HEIN, M.; SCHIELE, B. (Ed.). **Pattern Recognition**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. p. 271–281. ISBN 978-3-642-40602-7.

CHEN, Y.; YU, W.; POCK, T. On learning optimized reaction diffusion processes for effective image restoration. In: **IEEE CVPR**. [S.l.: s.n.], 2015.

CHIULLI, C. **Method for manufacturing an optical filter**. [S.l.]: Google Patents, 1989. US Patent 4,808,501.

CHOI, I. et al. High-quality hyperspectral reconstruction using a spectral prior. **ACM TOG**, v. 36, n. 6, p. 218:1–218:13, nov. 2017. ISSN 0730-0301.

CHOLLET, F. **Keras: Deep Learning library for Theano and TensorFlow**. 2015. <https://keras.io/>.

CHOLLET, F. **Deep learning with Python, Second Edition**. [S.l.]: Manning Publications Company, 2020.

CONDAT, L. A new random color filter array with good spectral properties. In: **ICIP**. [S.l.: s.n.], 2009. p. 1613–1616. ISSN 1522-4880.

CONDAT, L. Color filter array design using random patterns with blue noise chromatic spectra. **Image and Vision Computing**, v. 28, n. 8, p. 1196 – 1202, 2010. ISSN 0262-8856.

CONDAT, L. A new color filter array with optimal properties for noiseless and noisy color image acquisition. **TIP**, v. 20, n. 8, p. 2200–2210, 2011. ISSN 1057-7149.

CONDAT, L.; MOSADDEGH, S. Joint demosaicking and denoising by total variation minimization. In: **2012 19th IEEE International Conference on Image Processing**. [S.l.: s.n.], 2012. p. 2781–2784. ISSN 1522-4880.

DABOV, K. et al. Image denoising by sparse 3-d transform-domain collaborative filtering. **IEEE TIP**, v. 16, n. 8, p. 2080–2095, 2007. ISSN 1057-7149.

DAI, J. et al. Deformable convolutional networks. **CoRR**, abs/1703.06211, 2017. Available from Internet: <https://arxiv.org/abs/1703.06211>.

DAVE, A.; VADATHYA, A. K.; MITRA, K. Compressive image recovery using recurrent generative model. **CoRR**, abs/1612.04229, 2016. Available from Internet: <http://arxiv.org/abs/1612.04229>.

DENG, G. Edge-aware bma filters. **IEEE Transactions on Image Processing**, v. 25, n. 1, p. 439–454, Jan 2016. ISSN 1057-7149.

DENTON, E. L. et al. Deep generative image models using a laplacian pyramid of adversarial networks. **CoRR**, abs/1506.05751, 2015. Available from Internet: <http://arxiv.org/abs/1506.05751>.

DIAMOND, S. et al. Dirty pixels: Optimizing image classification architectures for raw sensor data. **CoRR**, abs/1701.06487, 2017.

DODGE, S.; KARAM, L. Understanding how image quality affects deep neural networks. In: **QoMEX**. [S.l.: s.n.], 2016. p. 1–6.

DONG, C. et al. Image super-resolution using deep convolutional networks. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 38, n. 2, p. 295–307, Feb 2016. ISSN 0162-8828.

DONG, W. et al. Nonlocally centralized sparse representation for image restoration. **IEEE TIP**, v. 22, n. 4, p. 1620–1630, 2013. ISSN 1057-7149.

DUBOIS, E. Frequency-domain methods for demosaicking of bayersampled color images. **IEEE Signal Processing Lett**, p. 847–850, 2005.

DUCHI, J.; HAZAN, E.; SINGER, Y. Adaptive subgradient methods for online learning and stochastic optimization. **J. Mach. Learn. Res.**, JMLR.org, v. 12, p. 2121–2159, jul. 2011. ISSN 1532-4435. Available from Internet: <http://dl.acm.org/citation.cfm?id=1953048.2021068>.

Eckel, S. et al. Realistic film noise generation based on experimental noise spectra. **IEEE TIP**, v. 29, p. 2987–2998, 2020.

EGER, S.; YOUSSEF, P.; GUREVYCH, I. Is it time to swish? comparing deep learning activation functions across NLP tasks. In: **Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing**. Brussels, Belgium: Association for Computational Linguistics, 2018. p. 4415–4424. Available from Internet: <https://www.aclweb.org/anthology/D18-1472>.

ELAD, M.; AHARON, M. Image denoising via sparse and redundant representations over learned dictionaries. **IEEE TIP**, v. 15, n. 12, p. 3736–3745, 2006. ISSN 1057-7149.

EMVA. **EMVA Standard 1288 Release 3.1**. 2016.

ENGSTROM, L. **Fast Style Transfer**. 2016. <https://github.com/lengstrom/fast-style-transfer/>.

FANG, H. et al. From captions to visual concepts and back. **CoRR**, abs/1411.4952, 2014. Available from Internet: <http://arxiv.org/abs/1411.4952>.

FATHI, A. et al. **G-RMI Object Detection**. 2016.

FENG, C. et al. Near-infrared guided color image dehazing. In: **ICIP**. [S.l.: s.n.], 2013. p. 2363–2367.

FOI, A. et al. Practical poissonian-gaussian noise modeling and fitting for single-image raw-data. **TIP**, v. 17, n. 10, p. 1737–1754, 2008. ISSN 1057-7149.

FRANZEN, R. **Kodak Lossless True Color Image Suite**. 1999. <http://r0k.us/graphics/kodak/>.

FRIED, O. et al. Perspective-aware manipulation of portrait photos. **ACM Transactions on Graphics (Proc. SIGGRAPH)**, jul. 2016.

GASTAL, E. S. L.; OLIVEIRA, M. M. Adaptive manifolds for real-time high-dimensional filtering. **ACM TOG**, v. 31, n. 4, p. 33:1–33:13, 2012.

GATYS, L. A.; ECKER, A. S.; BETHGE, M. Image style transfer using convolutional neural networks. In: **2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2016. p. 2414–2423.

GETREUER, P. Color demosaicing with contour stencils. In: **2011 17th International Conference on Digital Signal Processing (DSP)**. [S.l.: s.n.], 2011. p. 1–6. ISSN 1546-1874.

GETREUER, P. Zhang-Wu Directional LMMSE Image Demosaicking. **Image Processing On Line**, v. 1, 2011.

GHARBI, M. et al. Deep joint demosaicking and denoising. **ACM ToG**, ACM, v. 35, n. 6, p. 191:1–191:12, nov. 2016. ISSN 0730-0301.

GIRYES, R.; ELAD, M. Sparsity-based poisson denoising with dictionary learning. **IEEE TIP**, v. 23, n. 12, p. 5057–5069, 2014. ISSN 1057-7149.

GO, J.; SOHN, K.; LEE, C. Interpolation using neural networks for digital still cameras. **IEEE Transactions on Consumer Electronics**, v. 46, n. 3, p. 610–616, 2000. ISSN 0098-3063.

GOODFELLOW, I. et al. Generative adversarial nets. In: GHAHRAMANI, Z. et al. (Ed.). **Advances in Neural Information Processing Systems 27**. Curran Associates, Inc., 2014. p. 2672–2680. Available from Internet: <http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf>.

GOODFELLOW YOSHUA BENGIO, A. C. I. **Deep Learning (Adaptive Computation and Machine Learning series)**. 1. ed. [S.l.]: The MIT Press (November 18, 2016), 2016. ISBN 978-0262035613.

GROSS, S.; WILBER, M. **Training and investigating Residual Nets**. 2016. <http://torch.ch/blog/2016/02/04/resnets.html>.

GUO, S. et al. Toward convolutional blind denoising of real photographs. **IEEE CVPR**, 2019.

GUSTAVSON, T.; HOUSE, G. E. **Camera: A History of Photography from Daguerreotype to Digital**. [S.l.]: Sterling Signature, 2012. ISBN 978-1454900023.

HAO, P. et al. A geometric method for optimal design of color filter arrays. **TIP**, v. 20, n. 3, p. 709–722, 2011. ISSN 1057-7149.

HASINOFF, S. W. Photon, poisson noise. In: **Computer Vision**. [S.l.]: Springer US, 2014. p. 608–610. ISBN 978-0-387-31439-6.

HASINOFF, S. W.; DURAND, F.; FREEMAN, W. T. Noise-optimal capture for high dynamic range photography. In: **IEEE CVPR**. [S.l.: s.n.], 2010. p. 553–560. ISSN 1063-6919.

HASINOFF, S. W. et al. Burst photography for high dynamic range and low-light imaging on mobile cameras. **ACM Trans. Graph.**, v. 35, n. 6, p. 192:1–192:12, 2016. ISSN 0730-0301.

HE, K. et al. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. **CoRR**, abs/1502.01852, 2015. Available from Internet: <http://arxiv.org/abs/1502.01852>.

HE, K. et al. Deep residual learning for image recognition. In: **Proceedings of the CVPR**. [S.l.: s.n.], 2016.

HEIDE, F. et al. Flexisp: A flexible camera image processing framework. **ACM Transactions on Graphics (Proceedings SIGGRAPH Asia 2014)**, ACM, v. 33, n. 6, December 2014.

HEINRICH, G. **Photo Editing with Generative Adversarial Networks**. 2017. Available from Internet: <https://devblogs.nvidia.com/parallelforall/photo-editing-generative-adversarial-networks-1/>.

HEINZE, T.; LÖWIS, M. von; POLZE, A. Joint multi-frame demosaicing and super-resolution with artificial neural networks. In: **IWSSIP**. [S.l.: s.n.], 2012. p. 540–543. ISSN 2157-8672.

HENZ, B.; GASTAL, E. S. L.; OLIVEIRA, M. M. Deep joint design of color filter arrays and demosaicing. **CGF**, v. 37, n. 2, p. 389–399, 2018. ISSN 1467-8659.

HENZ EDUARDO S. L. GASTAL, M. M. O. B. Synthesizing camera noise using generative adversarial networks. **IEEE Transactions on Visualization and Computer Graphics**, 2020.

HEO, Y. S.; LEE, S.; JUNG, H. Y. Consistent color and detail transfer from multiple source images for video and images. **Vis. Comput.**, Springer-Verlag New York, Inc., Secaucus, NJ, USA, v. 32, n. 10, p. 1273–1289, oct. 2016. ISSN 0178-2789.

HIRAKAWA, K.; WOLFE, P. J. Spatio-spectral color filter array design for optimal image recovery. **TIP**, v. 17, n. 10, p. 1876–1890, 2008. ISSN 1057-7149.

HONDA, H.; TIMOFTE, R.; GOOL, L. V. Make my day - high-fidelity color denoising with near-infrared. In: **CVPRW**. [S.l.: s.n.], 2015. p. 82–90.

HOWARD, A. G. et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications. **CoRR**, abs/1704.04861, 2017. Available from Internet: <http://arxiv.org/abs/1704.04861>.

HUANG, X. et al. Multimodal unsupervised image-to-image translation. In: **ECCV**. [S.l.: s.n.], 2018.

HUSSEIN, S. A.; TIRER, T.; GIRYES, R. Correction filter for single image super-resolution: Robustifying off-the-shelf deep super-resolvers. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2020.

HWANG, Y.; KIM, J.; KWEON, I. S. Difference-based image noise modeling using skellam distribution. **IEEE TPAMI**, v. 34, n. 7, p. 1329–1341, 2012. ISSN 0162-8828.

IIZUKA, S.; SIMO-SERRA, E.; ISHIKAWA, H. Let there be Color!: Joint End-to-end Learning of Global and Local Image Priors for Automatic Image Colorization with Simultaneous Classification. **ACM Transactions on Graphics (Proc. of SIGGRAPH 2016)**, v. 35, n. 4, p. 110:1–110:11, 2016.

IOFFE, S.; SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: **Proceedings of the ICML**. [S.l.: s.n.], 2015. p. 448–456.

ISOLA, P. et al. Image-to-image translation with conditional adversarial networks. **CVPR**, 2017.

JAIN, V.; SEUNG, S. Natural image denoising with convolutional networks. In: **NIPS**. [S.l.: s.n.], 2009. p. 769–776.

JAISWAL, S. P. et al. Exploitation of inter-color correlation for color image demosaicking. In: **2014 IEEE International Conference on Image Processing (ICIP)**. [S.l.: s.n.], 2014. p. 1812–1816. ISSN 1522-4880.

JEON, G.; DUBOIS, E. Demosaicking of noisy bayer-sampled color images with least-squares luma-chroma demultiplexing and noise level estimation. **IEEE Trans. Image Processing**, v. 22, n. 1, p. 146–156, 2013. Available from Internet: <https://doi.org/10.1109/TIP.2012.2214041>.

JIN, K. H. et al. Deep convolutional neural network for inverse problems in imaging. **IEEE TIP**, v. 26, n. 9, p. 4509–4522, 2017. ISSN 1057-7149.

JOHNSON, J.; ALAHI, A.; LI, F. Perceptual losses for real-time style transfer and super-resolution. **CoRR**, abs/1603.08155, 2016. Available from Internet: <http://arxiv.org/abs/1603.08155>.

JOLICOEUR-MARTINEAU, A. **GANs beyond divergence minimization**. 2018.

KAPAH, O.; HEL-OR, H. Z. Demosaicking using artificial neural networks. In: **Proc. SPIE**. [S.l.: s.n.], 2000. v. 3962, p. 112–120.

KARRAS, T.; LAINE, S.; AILA, T. A style-based generator architecture for generative adversarial networks. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2019. p. 4401–4410.

KAUR, M.; WALIA, S. Forgery detection using noise estimation and hog feature extraction. **Int. J. Multimed. Ubiquitous Eng**, v. 11, n. 4, p. 37–48, 2016.

KAUR, S.; BANGA, V. K. A survey of demosaicing: Issues and challenges. **International Journal of Science, Engineering and Technologies**, v. 2, n. 1, 2015.

KEMELMACHER-SHLIZERMAN, I. Transfiguring portraits. **ACM Trans. Graph.**, ACM, New York, NY, USA, v. 35, n. 4, p. 94:1–94:8, jul. 2016. ISSN 0730-0301. Available from Internet: <http://doi.acm.org/10.1145/2897824.2925871>.

KIKU, D. et al. Beyond color difference: Residual interpolation for color image demosaicking. **IEEE Transactions on Image Processing**, v. 25, n. 3, p. 1288–1300, March 2016. ISSN 1057-7149.

KIM, B. et al. Lagrangian neural style transfer for fluids. **ACM Trans. Graph.**, Association for Computing Machinery, New York, NY, USA, v. 39, n. 4, jul. 2020. ISSN 0730-0301. Available from Internet: <https://doi.org/10.1145/3386569.3392473>.

KIM, J.; LEE, J. K.; LEE, K. M. Accurate image super-resolution using very deep convolutional networks. **CoRR**, abs/1511.04587, 2015. Available from Internet: <http://arxiv.org/abs/1511.04587>.

KIM, S. J. et al. A new in-camera imaging model for color computer vision and its application. **IEEE TPAMI**, v. 34, n. 12, p. 2289–2302, 2012. ISSN 0162-8828.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. **CoRR**, abs/1412.6980, 2014. Available from Internet: <http://arxiv.org/abs/1412.6980>.

KINGMA, D. P.; WELLING, M. Auto-encoding variational bayes. **CoRR**, abs/1312.6114, 2013.

KLATZER, T. et al. Learning joint demosaicing and denoising based on sequential energy minimization. In: **2016 IEEE International Conference on Computational Photography (ICCP)**. [S.l.: s.n.], 2016. p. 1–11.

KODALI, N. et al. **On Convergence and Stability of GANs**. 2017.

KRISHNAN, D.; FERGUS, R. Dark flash photography. **TOG**, v. 28, n. 3, p. 96:1–96:11, 2009. ISSN 0730-0301.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: **NIPS 25**. [S.l.: s.n.], 2012. p. 1097–1105.

KUMAR, W. K.; NONGMEIKAPAM, K.; SINGH, A. D. Enhancing scene perception using a multispectral fusion of visible–near-infrared image pair. **IET Image Processing**, IET, v. 13, n. 13, p. 2467–2479, 2019.

LAN, X. et al. Efficient belief propagation with learned higher-order markov random fields. In: **ECCV**. [S.l.: s.n.], 2006. p. 269–282. ISBN 978-3-540-33835-2.

LAPRAY, P.-J. et al. Multispectral filter arrays: Recent advances and practical implementation. **Sensors**, v. 14, n. 11, p. 21626–21659, 2014. ISSN 1424-8220.

LAQUERRE NICOLAS ETIENNE, N. V. C. D. A. L. S. S. P.-F. **RGB-NIR Scene Dataset**. 2011. <http://ivrl.epfl.ch/supplementary_material/cvpr11/>.

LARSSON, G.; MAIRE, M.; SHAKHNAROVICH, G. Learning representations for automatic colorization. In: **European Conference on Computer Vision (ECCV)**. [S.l.: s.n.], 2016.

LECUN, Y. et al. Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, v. 86, n. 11, p. 2278–2324, 1998. ISSN 0018-9219.

LEDIG, C. et al. Photo-realistic single image super-resolution using a generative adversarial network. **CoRR**, abs/1609.04802, 2016. Available from Internet: <http://arxiv.org/abs/1609.04802>.

LEE, J.-Y. et al. Automatic content-aware color and tone stylization. In: **The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2016.

LEHTINEN, J. et al. Noise2Noise: Learning image restoration without clean data. In: **ICML**. [S.l.: s.n.], 2018. v. 80, p. 2965–2974.

LETTRY, L.; VANHOEY, K.; GOOL, L. V. Darn: a deep adversarial residual network for intrinsic image decomposition. In: IEEE. **2018 IEEE Winter Conference on Applications of Computer Vision (WACV)**. [S.l.], 2018. p. 1359–1367.

LI, C.; WAND, M. Precomputed real-time texture synthesis with markovian generative adversarial networks. In: **ECCV**. [S.l.: s.n.], 2016. v. 9907, p. 702–716. ISBN 978-3-319-46486-2.

LI, F.-F.; JOHNSON, J.; YEUNG, S. **CS231n: Convolutional Neural Networks for Visual Recognition**. 2017. Available from Internet: <http://cs231n.stanford.edu/>.

LI, J. et al. Optimized color filter arrays for sparse representation-based demosaicking. **IEE TIP**, v. 26, n. 5, p. 2381–2393, 2017. ISSN 1057-7149.

LI, S. Z. **Markov Random Field Modeling in Image Analysis**. [S.l.]: Springer-Verlag London, 2009. ISBN 978-1-84800-279-1.

LI, X. Demosaicing by successive approximation. **IEEE Transactions on Image Processing**, v. 14, n. 3, p. 370–379, March 2005. ISSN 1057-7149.

LI, X.; GUNTURK, B.; ZHANG, L. Image demosaicing: a systematic survey. In: **Proc. SPIE**. [S.l.: s.n.], 2008. v. 6822, p. 1–15.

LI, Y. et al. Fully convolutional instance-aware semantic segmentation. **CoRR**, abs/1611.07709, 2016. Available from Internet: <http://arxiv.org/abs/1611.07709>.

LIAN, N. X. et al. Adaptive filtering for color filter array demosaicking. **IEEE Transactions on Image Processing**, v. 16, n. 10, p. 2515–2525, Oct 2007. ISSN 1057-7149.

LIN, M.; CHEN, Q.; YAN, S. Network in network. **CoRR**, abs/1312.4400, 2013.

LIN, T.-Y. et al. Microsoft coco: Common objects in context. In: **European Conference on Computer Vision (ECCV)**. Zürich: [s.n.], 2014. Oral. Available from Internet: </se3/wp-content/uploads/2014/09/coco_eccv.pdf,http://mscoco.org>.

LIU, C. et al. Automatic estimation and removal of noise from a single image. **IEEE TPAMI**, v. 30, n. 2, p. 299–314, 2008. ISSN 0162-8828.

LIU, H. et al. Image inpainting based on hidden markov random field. In: **2016 IEEE 13th International Conference on Signal Processing (ICSP)**. [S.l.: s.n.], 2016. p. 697–701. ISSN 2164-5221.

LIU, H. et al. Adaptiveface: Adaptive margin and sampling for face recognition. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2019. p. 11947–11956.

LIU, M.-Y.; TUZEL, O. Coupled generative adversarial networks. In: LEE, D. D. et al. (Ed.). **NIPS 29**. [S.l.: s.n.], 2016. p. 469–477.

LIU, N. et al. Predicting eye fixations using convolutional neural networks. In: **2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2015. p. 362–370. ISSN 1063-6919.

LIU, R. et al. Enhanced residual dense intrinsic network for intrinsic image decomposition. In: IEEE. **2019 IEEE International Conference on Multimedia and Expo (ICME)**. [S.l.], 2019. p. 1462–1467.

LIU, X.; TANAKA, M.; OKUTOMI, M. Single-image noise level estimation for blind denoising. **IEEE TIP**, v. 22, n. 12, p. 5226–5237, 2013. ISSN 1057-7149.

LIU, Y. et al. Learning deep priors for image dehazing. In: **Proceedings of the IEEE International Conference on Computer Vision**. [S.l.: s.n.], 2019. p. 2492–2500.

LONG, Y.; HUANG, Y. Adaptive demosaicking using multiple neural networks. In: **IEEE Signal Processing Society Workshop on MLSP**. [S.l.: s.n.], 2006. p. 353–357. ISSN 1551-2541.

LOPEZ-PAZ, D.; OQUAB, M. Revisiting classifier two-sample tests. In: **ICLR**. [S.l.: s.n.], 2017.

LU, Y.; KARZAND, M.; VETTERLI, M. Demosaicking by alternating projections: Theory and fast one-step implementation. **IEEE Transactions on Image Processing**, v. 19, n. 8, p. 2085–2098, 2010. Available from Internet: <http://scholar.harvard.edu/yuelu/software/demosaicking-matlab-code-implementing-fast-demosaicking-algorithm-described-following>.

LU, Y. M. et al. Designing color filter arrays for the joint capture of visible and near-infrared images. In: **2009 16th IEEE International Conference on Image Processing (ICIP)**. [S.l.: s.n.], 2009. p. 3797–3800. ISSN 1522-4880.

LU, Y. M.; VETTERLI, M. Optimal color filter array design: quantitative conditions and an efficient search procedure. In: **Digital Photography**. [S.l.]: SPIE, 2009. v. 7250, p. 1–9. ISBN 978-0-8194-7500-8.

LUC, P. et al. Semantic segmentation using adversarial networks. **CoRR**, abs/1611.08408, 2016. Available from Internet: <http://arxiv.org/abs/1611.08408>.

LUKAC, R.; PLATANIOTIS, K. N. Color filter arrays: design and performance analysis. **IEEE Transactions on Consumer Electronics**, v. 51, n. 4, p. 1260–1267, 2005. ISSN 0098-3063.

MAAS, A. L.; HANNUN, A. Y.; NG, A. Y. Rectifier Nonlinearities Improve Neural Network Acoustic Models. Available from Internet: <http://web.stanford.edu/~{}awni/papers/relu_hybrid_icml2013_final.p>.

MAATEN, L. van der; HINTON, G. Visualizing high-dimensional data using t-sne. **JMLR**, v. 9, p. 2579—2605, 2008.

MAATEN, L. van der; HINTON, G. E. Visualizing high-dimensional data using t-sne. **JMLR**, v. 9, p. 2579–2605, 2008.

MAI, L.; JIN, H.; LIU, F. Composition-preserving deep photo aesthetics assessment. In: **2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2016. p. 497–506.

MAIRAL, J. et al. Non-local sparse models for image restoration. In: **ICCV**. [S.l.: s.n.], 2009. p. 2272–2279.

Makovoz, D. Noise variance estimation in signal processing. In: **IEEE ISSPIT**. [S.l.: s.n.], 2006. p. 364–369. ISSN 2162-7843.

MANDAL, S.; BHAVSAR, A.; SAO, A. K. Noise adaptive super-resolution from single image via non-local mean and sparse representation. **Signal Proc.**, v. 132, p. 134 – 149, 2017. ISSN 0165-1684.

MANSIMOV, E. et al. Generating images from captions with attention. **CoRR**, abs/1511.02793, 2015. Available from Internet: <http://arxiv.org/abs/1511.02793>.

MAO, X. et al. Multi-class generative adversarial networks with the L2 loss function. **CoRR**, abs/1611.04076, 2016.

MAO, X. et al. Least squares generative adversarial networks. In: **Proceedings of the IEEE International Conference on Computer Vision (ICCV)**. [S.l.: s.n.], 2017.

MAO, X.; SHEN, C.; YANG, Y. Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. In: **NIPS**. [S.l.: s.n.], 2016. p. 2802–2810.

MENON, D.; CALVAGNO, G. Color image demosaicking: An overview. **Image Commun.**, v. 26, n. 8-9, p. 518–533, 2011.

MILBURN, K.; ROCKWELL, R.; CHAMBERS, M. I. **Digital Photography Bible: Desktop Edition**. 2. ed. [S.l.]: John Wiley & Sons, 2002. ISBN 978-0764549519.

MIRZA, M.; OSINDERO, S. Conditional generative adversarial nets. **CoRR**, abs/1411.1784, 2014. Available from Internet: <http://arxiv.org/abs/1411.1784>.

MNIH, V. et al. Recurrent models of visual attention. In: GHAHRAMANI, Z. et al. (Ed.). **Advances in Neural Information Processing Systems 27**. Curran Associates, Inc., 2014. p. 2204–2212. Available from Internet: <http://papers.nips.cc/paper/5542-recurrent-models-of-visual-attention.pdf>.

MOGHADAM, A. A. et al. Compressive framework for demosaicing of natural images. **IEEE Transactions on Image Processing**, v. 22, n. 6, p. 2356–2371, June 2013. ISSN 1057-7149.

MOREL, J. michel; YU, G. Is sift scale invariant. In: **Inverse Problems and Imaging**. [S.l.: s.n.], 2011.

NAIR, V.; HINTON, G. E. Rectified linear units improve restricted boltzmann machines. In: FüRNKRANZ, J.; JOACHIMS, T. (Ed.). **Proceedings of the 27th International Conference on Machine Learning (ICML-10)**. Omnipress, 2010. p. 807–814. Available from Internet: <http://www.icml2010.org/papers/432.pdf>.

NAM, S. et al. A holistic approach to cross-channel image noise modeling and its application to image denoising. In: **IEEE CVPR**. [S.l.: s.n.], 2016. p. 1683–1691. ISSN 1063-6919.

NARIHIRA, T.; MAIRE, M.; YU, S. X. Direct intrinsics: Learning albedo-shading decomposition by convolutional regression. **CoRR**, abs/1512.02311, 2015. Available from Internet: <http://arxiv.org/abs/1512.02311>.

NEWELL, A.; YANG, K.; DENG, J. Stacked hourglass networks for human pose estimation. **CoRR**, abs/1603.06937, 2016. Available from Internet: <http://arxiv.org/abs/1603.06937>.

NEWSON, A.; DELON, J.; GALERNE, B. A stochastic film grain model for resolution-independent rendering. **CGF**, v. 36, n. 8, p. 684–699, 2017.

NOH, H.; HONG, S.; HAN, B. Learning deconvolution network for semantic segmentation. **CoRR**, abs/1505.04366, 2015. Available from Internet: <http://arxiv.org/abs/1505.04366>.

Odena, A. Semi-Supervised Learning with Generative Adversarial Networks. **ArXiv e-prints**, jun. 2016.

ODENA, A.; OLAH, C.; SHLENS, J. Conditional image synthesis with auxiliary classifier gans. **arXiv**, 2016. Available from Internet: <https://arxiv.org/abs/1610.09585>.

OSHER, S. et al. An iterative regularization method for total variation-based image restoration. **Multiscale Modeling & Simulation**, v. 4, n. 2, p. 460–489, 2005.

PAN, J. et al. Shallow and deep convolutional networks for saliency prediction. In: **The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2016.

PETROVIC, M.; PETROVIC, D.; NIKOLIC, G. An approach to noise variance estimation in very low signal-to-noise ratio stochastic signals. **International Journal of Computer, Electrical, Automation, Control and Information Engineering**, v. 10, p. 407–410, 2016.

PLOTZ, T.; ROTH, S. Benchmarking denoising algorithms with real photographs. In: **IEEE CVPR**. [S.l.: s.n.], 2017. v. 00, p. 2750–2759. ISSN 1063-6919.

QI, G.-J. Loss-sensitive generative adversarial networks on lipschitz densities. **International Journal of Computer Vision**, v. 128, n. 5, p. 1118–1140, 2020.

RATNASINGAM, S. Deep camera: A fully convolutional neural network for image signal processing. In: **Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops**. [S.l.: s.n.], 2019.

RONNEBERGER, O.; P.FISCHER; BROX, T. U-net: Convolutional networks for biomedical image segmentation. In: **MICCAI**. [S.l.: s.n.], 2015. (LNCS, v. 9351), p. 234–241.

ROTH, S.; BLACK, M. J. Fields of experts. **International Journal of Computer Vision**, v. 82, n. 2, p. 205, Jan 2009. ISSN 1573-1405.

ROY, S. S. et al. A robust system for noisy image classification combining denoising autoencoder and convolutional neural network. **IJACSA**, v. 9, n. 1, 2018.

RUDIN, L. I.; OSHER, S.; FATEMI, E. Nonlinear total variation based noise removal algorithms. **Physica D: Nonlinear Phenomena**, v. 60, n. 1, p. 259 – 268, 1992. ISSN 0167-2789.

RUSSAKOVSKY, O. et al. ImageNet Large Scale Visual Recognition Challenge. **International Journal of Computer Vision (IJCV)**, v. 115, n. 3, p. 211–252, 2015.

SADEGHIPOOR, Z.; LU, Y. M.; SüSSTRUNK, S. Correlation-based joint acquisition and demosaicing of visible and near-infrared images. In: **ICIP**. [S.l.: s.n.], 2011. p. 3165–3168. ISSN 1522-4880.

SALIMANS, T. et al. Improved techniques for training gans. arxiv 2016. **arXiv preprint arXiv:1606.03498**, 2016.

SAMMANI, F.; MELAS-KYRIAZI, L. Show, edit and tell: A framework for editing image captions. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2020.

SCHETINGER, V. et al. Image forgery detection confronts image composition. **Computers and Graphics**, v. 68, p. 152 – 163, 2017. ISSN 0097-8493. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S0097849317301498>.

SCHMIDT, U. et al. Cascades of regression tree fields for image restoration. **IEEE TPAMI**, v. 38, n. 4, p. 677–689, 2016. ISSN 0162-8828.

SCHMIDT, U.; ROTH, S. Shrinkage fields for effective image restoration. In: **IEEE CVPR**. [S.l.: s.n.], 2014. p. 2774–2781. ISSN 1063-6919.

SCHMIDT, U. et al. Discriminative non-blind deblurring. In: **IEEE CVPR**. [S.l.: s.n.], 2013. p. 604–611. ISSN 1063-6919.

SCHULER, C. J. et al. Learning to deblur. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 38, n. 7, p. 1439–1451, July 2016. ISSN 0162-8828.

SHAMSABADI, A. S.; SANCHEZ-MATILLA, R.; CAVALLARO, A. Colorfool: Semantic adversarial colorization. In: **IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2020.

SHI, L.; FUNT, B. **Re-processed Version of the Gehler Color Constancy Dataset of 568 Images**. 2010. <http://www.cs.sfu.ca/~colour/data/shi_gehler/>.

SHIH, Y. et al. Reflection removal using ghosting cues. In: **2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2015. p. 3193–3201. ISSN 1063-6919.

SHOCHER, A. et al. Semantic pyramid for image generation. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2020. p. 7457–7466.

SHRIVASTAVA, A. et al. Learning from simulated and unsupervised images through adversarial training. In: **IEEE CVPR**. [S.l.: s.n.], 2017.

SILIOS Technologies. **COLOR SHADES**. 2017. Available from Internet: <https://www.silios.com/color-shades>.

SIMON, D. **Digital Photography Bible: Desktop Edition**. 1. ed. [S.l.]: Wiley Publishing, Inc, 2004. ISBN 978-0764568756.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. **CoRR**, abs/1409.1556, 2014.

SRIVASTAVA, N. et al. Dropout: A simple way to prevent neural networks from overfitting. **JMLR**, v. 15, n. 1, p. 1929–1958, 2014. ISSN 1532-4435.

SZEGEDY, C.; IOFFE, S.; VANHOUCKE, V. Inception-v4, inception-resnet and the impact of residual connections on learning. **CoRR**, abs/1602.07261, 2016. Available from Internet: <http://arxiv.org/abs/1602.07261>.

SZEGEDY, C. et al. Going deeper with convolutions. In: **CVPR**. [S.l.: s.n.], 2015. p. 1–9. ISSN 1063-6919.

SZEGEDY, C. et al. Rethinking the inception architecture for computer vision. **CoRR**, abs/1512.00567, 2015.

TAIGMAN, Y. et al. Deepface: Closing the gap to human-level performance in face verification. In: **Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition**. Washington, DC, USA: IEEE Computer Society, 2014. (CVPR '14), p. 1701–1708. ISBN 978-1-4799-5118-5. Available from Internet: <http://dx.doi.org/10.1109/CVPR.2014.220>.

TANG, H. et al. High resolution photography with an rgb-infrared camera. In: **ICCP**. [S.l.: s.n.], 2015. p. 1–10.

TEAM, T. D. Theano: A Python framework for fast computation of mathematical expressions. **arXiv e-prints**, abs/1605.02688, may 2016.

TERANAKA, H. et al. Single-sensor RGB and NIR image acquisition: Toward optimal performance by taking account of CFA pattern, demosaicking, and color correction. In: **Digital Photography and Mobile Imaging XII**. [S.l.: s.n.], 2016. p. 1–6.

THONAT, T. et al. Multi-view inpainting for image-based scene editing and rendering. In: **International Conference on 3D Vision (3DV)**. [s.n.], 2016. Available from Internet: <http://www-sop.inria.fr/reves/Basilic/2016/TSPD16>.

TIELEMAN, T.; HINTON, G. **Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude**. [S.l.], 2012.

TIMOFTE, R. et al. Ntire 2017 challenge on single image super-resolution: Methods and results. In: **IEEE CVPRW**. [S.l.: s.n.], 2017. p. 1110–1121.

TOSHEV, A.; SZEGEDY, C. Deeppose: Human pose estimation via deep neural networks. In: **2014 IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2014. p. 1653–1660. ISSN 1063-6919.

TRAN, A.; MATHEWS, A.; XIE, L. Transform and tell: Entity-aware news image captioning. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2020.

TSAI, Y.-H. et al. Sky is not the limit: Semantic-aware sky replacement. **ACM Trans. Graph.**, ACM, New York, NY, USA, v. 35, n. 4, p. 149:1–149:11, jul. 2016. ISSN 0730-0301. Available from Internet: <http://doi.acm.org/10.1145/2897824.2925942>.

TSIAMI, A.; KOUTRAS, P.; MARAGOS, P. Stavis: Spatio-temporal audiovisual saliency network. In: **IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2020.

ULYANOV, D.; VEDALDI, A.; LEMPITSKY, V. S. Instance normalization: The missing ingredient for fast stylization. **CoRR**, abs/1607.08022, 2016.

VINCENT, P. et al. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. **JMLR**, v. 11, p. 3371–3408, 2010. ISSN 1532-4435.

VINYALS, O. et al. Show and tell: A neural image caption generator. **CoRR**, abs/1411.4555, 2014. Available from Internet: <http://arxiv.org/abs/1411.4555>.

WANG, L. et al. Deep networks for saliency detection via local estimation and global search. In: **2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2015. p. 3183–3192. ISSN 1063-6919.

WANG, L. et al. Aspect-ratio-preserving multi-patch image aesthetics score prediction. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops**. [S.l.: s.n.], 2019. p. 0–0.

WANG, L. et al. Temporal segment networks: Towards good practices for deep action recognition. In: **ECCV**. [S.l.: s.n.], 2016.

WANG, M.; TIGHE, J.; MODOLO, D. Combining detection and tracking for human pose estimation in videos. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2020.

WANG, P.; LIN, Z.; MECH, R. Learning an aesthetic photo cropping cascade. In: **2015 IEEE Winter Conference on Applications of Computer Vision**. [S.l.: s.n.], 2015. p. 448–455. ISSN 1550-5790.

Wang, X. et al. Near-infrared image guided neural networks for color image denoising. In: **IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)**. [S.l.: s.n.], 2019. p. 3807–3811.

WANG, Y. Q. A multilayer neural network for image demosaicking. In: **ICIP**. [S.l.: s.n.], 2014. p. 1852–1856. ISSN 1522-4880.

WHITE, R.; DOWNS, T. E. **How Digital Photography Works (How It Works)**. [S.l.]: Que, 2005. ISBN 978-0789733092.

WIATRAK, M.; ALBRECHT, S. V. **Stabilizing Generative Adversarial Network Training: A Survey**. 2019.

XIE, J.; XU, L.; CHEN, E. Image denoising and inpainting with deep neural networks. In: **NIPS**. [S.l.: s.n.], 2012. p. 341–349.

XU, B. et al. End-to-end illuminant estimation based on deep metric learning. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2020. p. 3616–3625.

XU, L. et al. Deep convolutional neural network for image deconvolution. In: GHAHRAMANI, Z. et al. (Ed.). **Advances in Neural Information Processing Systems 27**. Curran Associates, Inc., 2014. p. 1790–1798. Available from Internet: <http://papers.nips.cc/paper/5485-deep-convolutional-neural-network-for-image-deconvolution.pdf>.

XU, L. et al. Deep edge-aware filters. In: BACH, F. R.; BLEI, D. M. (Ed.). **ICML**. [S.l.]: JMLR.org, 2015. (JMLR Workshop and Conference Proceedings, v. 37), p. 1669–1678.

Xu, N. et al. Deep Image Matting. **ArXiv e-prints**, mar. 2017.

YAN, Z. et al. Automatic photo adjustment using deep neural networks. **ACM Transactions on Graphics**, 2015.

Yasuma, F. et al. Generalized assorted pixel camera: Postcapture control of resolution, dynamic range, and spectrum. **IEEE Transactions on Image Processing**, v. 19, n. 9, p. 2241–2253, 2010.

YEH, R. et al. Semantic image inpainting with perceptual and contextual losses. **CoRR**, abs/1607.07539, 2016. Available from Internet: <http://arxiv.org/abs/1607.07539>.

YI, Z. et al. Contextual residual aggregation for ultra high-resolution image inpainting. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2020. p. 7508–7517.

YI, Z. et al. Dualgan: Unsupervised dual learning for image-to-image translation. In: **IEEE ICCV**. [S.l.: s.n.], 2017. v. 00, p. 2868–2876. ISSN 2380-7504.

YI, Z. et al. Dualgan: Unsupervised dual learning for image-to-image translation. In: **IEEE ICCV**. [S.l.: s.n.], 2017. p. 2868–2876. ISSN 2380-7504.

YU, F.; KOLTUN, V. Multi-scale context aggregation by dilated convolutions. **CoRR**, abs/1511.07122, 2015. Available from Internet: <http://arxiv.org/abs/1511.07122>.

ZAGORUYKO, S.; KOMODAKIS, N. Wide residual networks. **CoRR**, abs/1605.07146, 2016.

ZEILER, M. D. ADADELTA: an adaptive learning rate method. **CoRR**, abs/1212.5701, 2012.

ZENG, K. et al. Coupled deep autoencoder for single image super-resolution. **IEEE Transactions on Cybernetics**, PP, n. 99, p. 1–11, 2015. ISSN 2168-2267.

ZHANG, H. et al. Deep stacked hierarchical multi-patch network for image deblurring. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2019. p. 5978–5986.

ZHANG, K. et al. Beyond a gaussian denoiser: Residual learning of deep CNN for image denoising. **CoRR**, abs/1608.03981, 2016. Available from Internet: <http://arxiv.org/abs/1608.03981>.

ZHANG, K. et al. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. **IEEE TIP**, v. 26, n. 7, p. 3142–3155, 2017. ISSN 1057-7149.

ZHANG, L.; WU, X. Color demosaicking via directional linear minimum mean square-error estimation. **IEEE Transactions on Image Processing**, v. 14, n. 12, p. 2167–2178, Dec 2005. ISSN 1057-7149.

ZHANG, L. et al. Color demosaicking by local directional interpolation and nonlocal adaptive thresholding. **J. Electronic Imaging**, v. 20, n. 2, p. 023016, 2011.

ZHANG, R.; ISOLA, P.; EFROS, A. A. Colorful image colorization. **ECCV**, 2016.

ZHANG, W. et al. Ranksrgan: Generative adversarial networks with ranker for image super-resolution. In: **Proceedings of the IEEE International Conference on Computer Vision**. [S.l.: s.n.], 2019. p. 3096–3105.

ZHANG, X. et al. Portrait shadow manipulation. In: . [S.l.: s.n.], 2020. v. 39, n. 4.

ZHANG XIAOLIN WU, A. B. X. L. L. **McMaster dataset**. 2011. <http://www4.comp.polyu.edu.hk/~cslzhang/CDM_Dataset.htm>.

ZHAO, J.; MATHIEU, M.; LECUN, Y. Energy-based generative adversarial network. **ArXiv**, abs/1609.03126, 2016.

ZHAO, Y. et al. Parallel style-aware image cloning for artworks. **IEEE Transactions on Visualization and Computer Graphics**, v. 21, n. 2, p. 229–240, Feb 2015. ISSN 1077-2626.

ZHOU, T.; KRÄHENBÜHL, P.; EFROS, A. A. Learning data-driven reflectance priors for intrinsic image decomposition. **CoRR**, abs/1510.02413, 2015. Available from Internet: <http://arxiv.org/abs/1510.02413>.

ZHU, J. et al. Toward multimodal image-to-image translation. In: **NIPS**. [S.l.: s.n.], 2017. p. 465–476.

ZHU, J.-Y. et al. Learning a discriminative model for the perception of realism in composite images. In: **Computer Vision (ICCV), 2015 IEEE International Conference on**. [S.l.: s.n.], 2015.

ZHU, J.-Y. et al. Unpaired image-to-image translation using cycle-consistent adversarial networks. **arXiv preprint arXiv:1703.10593**, 2017.

APPENDIX A — SYNTHESIZING CAMERA NOISE USING GENERATIVE ADVERSARIAL NETWORKS

A.1 Choosing σ for Low-Frequency Loss

For extracting the low-frequency content used for computing the low-frequency-consistency loss term, we perform a Gaussian Blur with a pre-defined σ on each patch. The noise found in higher ISO levels (normally encoded in the higher frequencies) has greater intensity compared to lower ISO levels. Thus, we use different σ values for different mappings (e.g. ISO 100→1600 vs. ISO 100→3200). But rather than choosing the values empirically, we pick σ based on the dataset containing scenes captured on each ISO level (Section 5.2). Thus, for each pair of domains (e.g., 100→1600, 200→1600), we linearly search for the σ value that satisfies the following inequality:

$$\mathrm{MSE}(\mathbb{G}_\sigma(x), \mathbb{G}_\sigma(y)) < 0.005, \tag{A.1}$$

where MSE is the mean-squared error between the blurred versions of x (image belonging to domain X) and y (belonging to domain Y), and $\mathbb{G}$ is a Gaussian blur implemented as a convolution by a 21×21 kernel with the appropriate σ (measured in pixels). We choose a low-threshold in Eq. (A.1) (0.005) to enforce the blurred versions to be at least similar. The σ values chosen for each mapping of the experiments can be found in Table A.1.

A.2 Evaluation Network Architectures

All classifiers, for natural versus artificial noise (Section 6.2), for identifying the ISO level of a given patch (Section 5.2), and for noise-model classification (Section 6.3) share similar architectures. All networks take as input a 128×128 RGB patch, and the patch's Fourier amplitude coefficients for each channel in log-space and in their shifted version (*i.e.*, with the DC term translated to the center of the matrix representation). Thus, each classifier takes a $128 \times 128 \times 6$ input, and outputs one (for the case of natural noise classification), six (for the case of ISO level identification), or seven probabilities (for discriminating among noise models).

Let `Ck-f` denote a $k \times k$ Conv layer with `f` filters and $\mathrm{stride} = k/2 + 1$, followed by a ReLU; and `M2` a 2×2 max-pooling layer. Also, let `Fl` denote a flattening layer,

Table A.1: Chosen σ for different pair of domains (ISO levels).

Mapping	**Chosen** σ
$100, 400$	1.5
$100, 800$	2.5
$100, 1600$	3.5
$100, 3200$	4.5
$200, 1600$	2.5

`FC-n` a fully-connected layer with n hidden units, also followed by a ReLU. Let `Dp-d` be a dropout layer with d representing the fraction (between $[0, 1]$) of units that will be randomly set to 0, `Sig` the sigmoid activation function, and `Soft` the Softmax activation function. The architecture for the natural-noise classifier is as follows:
`C5-256, C5-256, C5-256, M2, C3-256, C3-256, C3-256, M2, C3-256, C3-256, M2, C3-256, M2, Fl, FC-128, Dp-0.2, FC-64, Dp-0.1, FC-1, Sig`.

The architecture of the ISO-level classifier is similar, replacing the last two layers with: `FC-6, Soft`.

The architecture of the noise-model classifier replaces the last two layers with: `FC-7, Soft`.

The natural-noise classifier uses the *binary cross-entropy* as loss function, while the ISO-level identifier and noise-model classifier use the *categorical cross-entropy*. All network training used the Adam optimizer with default parameters and learning rate of 0.001, batch size of 12, and trained for 2000 iterations. We also employed data augmentation: horizontal and vertical flips, random crops, and random 90° rotations.

A.3 Computation of KL divergence and KS-value

We use the following Python methods for computing the KL divergence and KS-value for our experiments, where p and q are the normalized histograms of noise being compared:

```
import numpy as np

def kl_div(p, q):
    idx = (p > 0) & (q > 0)
    p = p[idx]
```

```
    q = q[idx]
    return np.sum(p * np.log(p/q))

def ks_value(p, q):
    cum_p = np.cumsum(p)
    cum_q = np.cumsum(q)
    return np.max(np.abs(cum_p - cum_q))
```

Each noise histogram is computed from pixels obtained by the subtraction of the clean image from the noisy one, which extracts only the (per channel) additive noise component. Input images are stored with 8-bits per channel and as such the noise components are in the range $[-255, 255]$, but mostly concentrated around zero. Histograms are computed with the numpy.histogram function with 50 bins evenly distributed in $[-50, 50]$, and two additional bins for the extremes: "bins = np.concatenate(([-256], np.arange(-50, 50, 2), [256]), axis=0) - 0.1". The bins' intervals are shifted by -0.1 to avoid quantization artifacts. Finally, as mentioned in Section 6.4, we compute these metrics for each color channel, which are then averaged together.

A.4 Computation of KL divergence and KS-value

Table A.2 compares the KS divergence and KL-values over all population patches (see Section 6.4) for the following noise models: AWGN and AWGN+Poisson (both in

Table A.2: KL divergences and KS values for different noise models, ISO values, and lighting conditions ([L]ow and [N]ormal light brightness levels). Best (lower) values are highlighted in bold. These values were measured over the whole population of image patches. Notice how our method achieves better values for these metrics across all but one case.

		ISO 400		ISO 800		ISO 1600		ISO 3200	
		L	N	L	N	L	N	L	N
KL divergence	AWGN	0.0910	0.1063	0.0938	0.1108	0.1780	0.1478	0.0765	0.0796
	AWGN (linear)	0.3204	0.1358	0.2053	0.1009	0.0731	0.1785	0.0580	0.0374
	AWGN+Poisson	0.0559	0.0912	0.0938	0.1110	0.1782	0.1478	0.0764	0.0807
	AWGN+Poisson (linear)	0.1358	0.0781	0.2053	0.1010	0.0732	0.1786	0.0784	0.0383
	NoiseFlow	0.1052	0.1557	0.0593	0.1140	0.0517	0.0634	0.0801	0.0520
	GAN_{SIDD}	**0.0091**	**0.0323**	**0.0104**	**0.0249**	**0.0228**	**0.0129**	**0.0339**	**0.0188**
KS value	AWGN	0.0693	0.0787	0.0842	0.0909	0.1096	0.1237	0.1100	0.0896
	AWGN (linear)	0.1700	0.1000	0.1521	0.0726	0.0937	0.1277	0.0784	0.0629
	AWGN+Poisson	0.1806	0.0693	0.0842	0.0910	0.1096	0.1236	0.1100	0.0895
	AWGN+Poisson (linear)	0.1000	0.0668	0.1520	0.0726	0.0937	0.1277	0.0784	**0.0628**
	NoiseFlow	0.1138	0.1422	0.0900	0.1057	0.0929	0.0821	0.1270	0.0925
	GAN_{SIDD}	**0.0333**	**0.0564**	**0.0404**	**0.0550**	**0.0600**	**0.0643**	**0.0761**	0.0677

sRGB and linear space), Noise Flow, and GAN_{SIDD}.

A.5 Additional Comparisons

Here we show more comparisons among different noise models. Figs. A.1 to A.4 compare closeups of photographs captured by several cameras and lighting conditions for ISO 400, ISO 800, ISO 1600, and ISO 3200, respectively. Note how the results produced by our models consistently achieve better (lower) KL and KS scores, and look more similar to the corresponding ground truth images. For better visualization, we opt to show only the most competitive noise models: AWGN, AWGN+Poisson (in linear space), Noise Flow, and ours.

A.6 Residual Comparison

Fig. A.5 exhibit only the synthesized noise (residual), i.e., the difference between the noisy and clean versions. For better visualization, we take the absolute value and multiply it by 3. Notice how our method better mimics camera noise, specially regarding the size of noise grain and color distribution (check residual images).

Figure A.1: Comparison among noise models for synthesizing noise for ISO 400. (a) Patch corrupted by AWGN (using average std computed from the paired dataset, in this case $\sigma_{400} = 0.0420$); (b) patch corrupted by AWGN+Poisson in linear space; (c) patch corrupted by Noise Flow (applied in raw space); (d) patch corrupted by our GAN_{SIDD}; (e) the noisy patch; and (f) the clean patch.

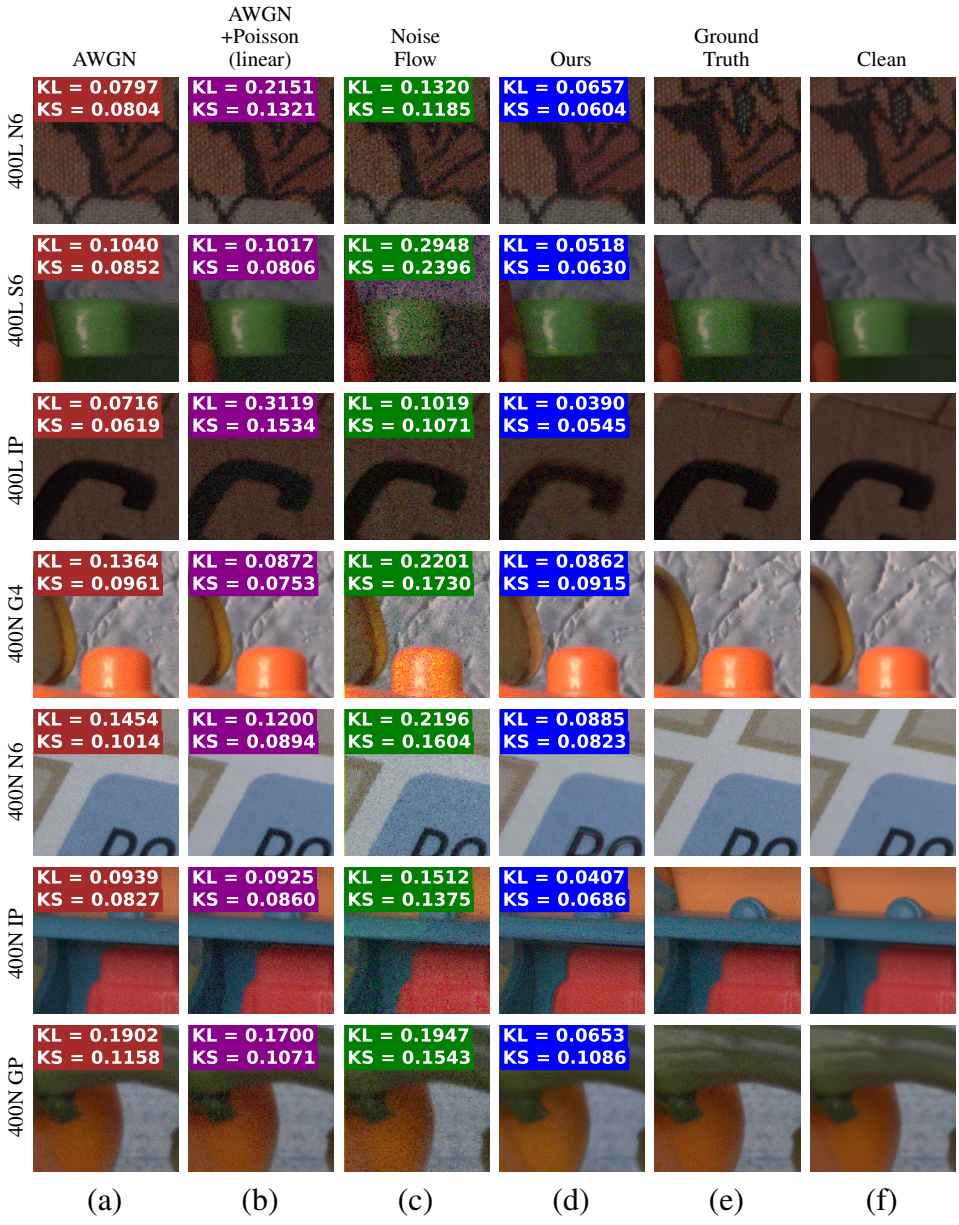


Figure A.2: Comparison among noise models for synthesizing noise for ISO 800. (a) Patch corrupted by AWGN (using average std computed from the paired dataset, in this case $\sigma_{800} = 0.0536$); (b) patch corrupted by AWGN+Poisson in linear space; (c) patch corrupted by Noise Flow (applied in raw space); (d) patch corrupted by our GAN_{SIDD}; (e) the noisy patch; and (f) the clean patch.

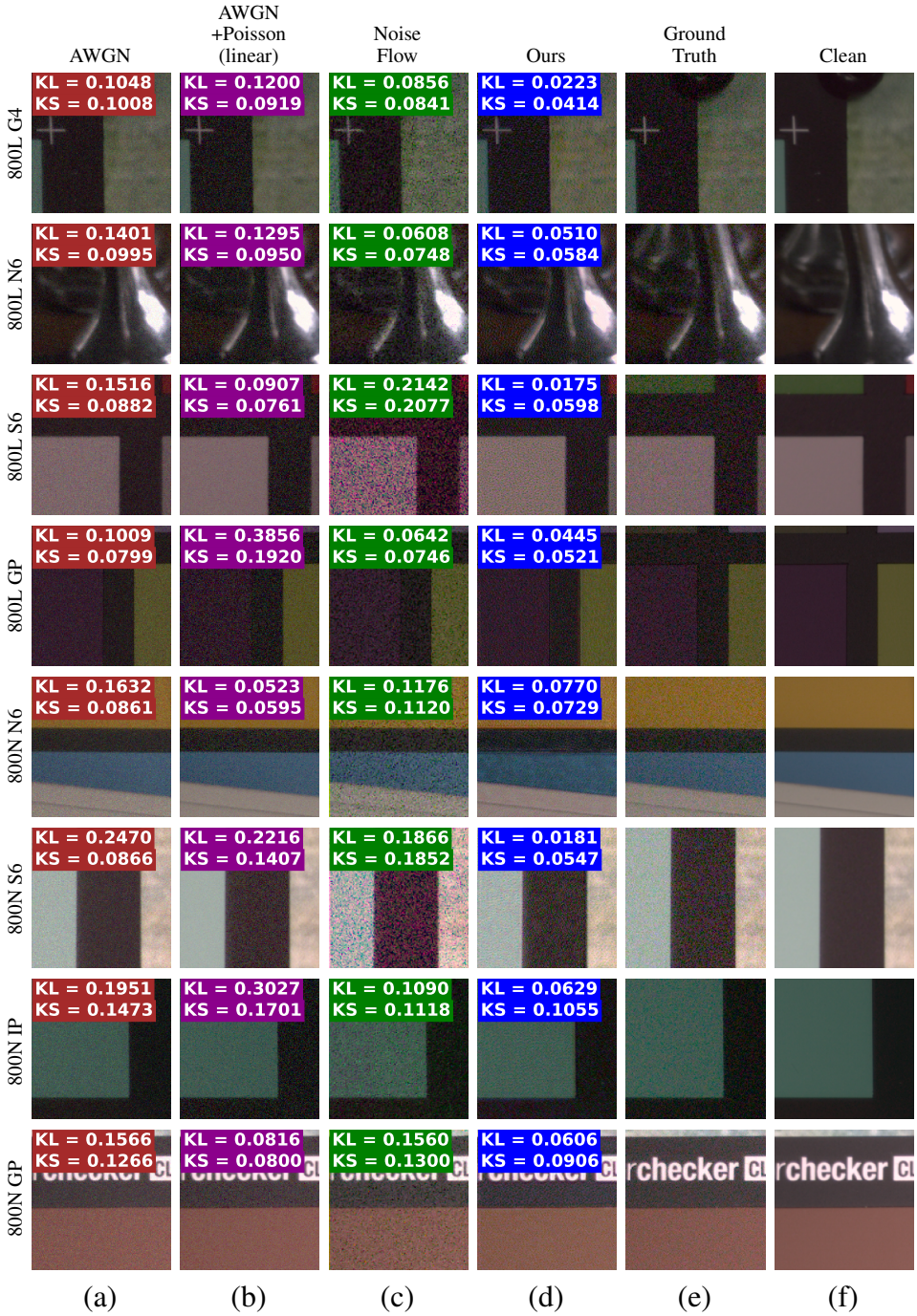


Figure A.3: Comparison among noise models for synthesizing noise for ISO 1600. (a) Patch corrupted by AWGN (using average std computed from the paired dataset, in this case $\sigma_{1600} = 0.0714$); (b) patch corrupted by AWGN+Poisson in linear space; (c) patch corrupted by Noise Flow (applied in raw space); (d) patch corrupted by our GAN_{SIDD}; (e) the noisy patch; and (f) the clean patch.

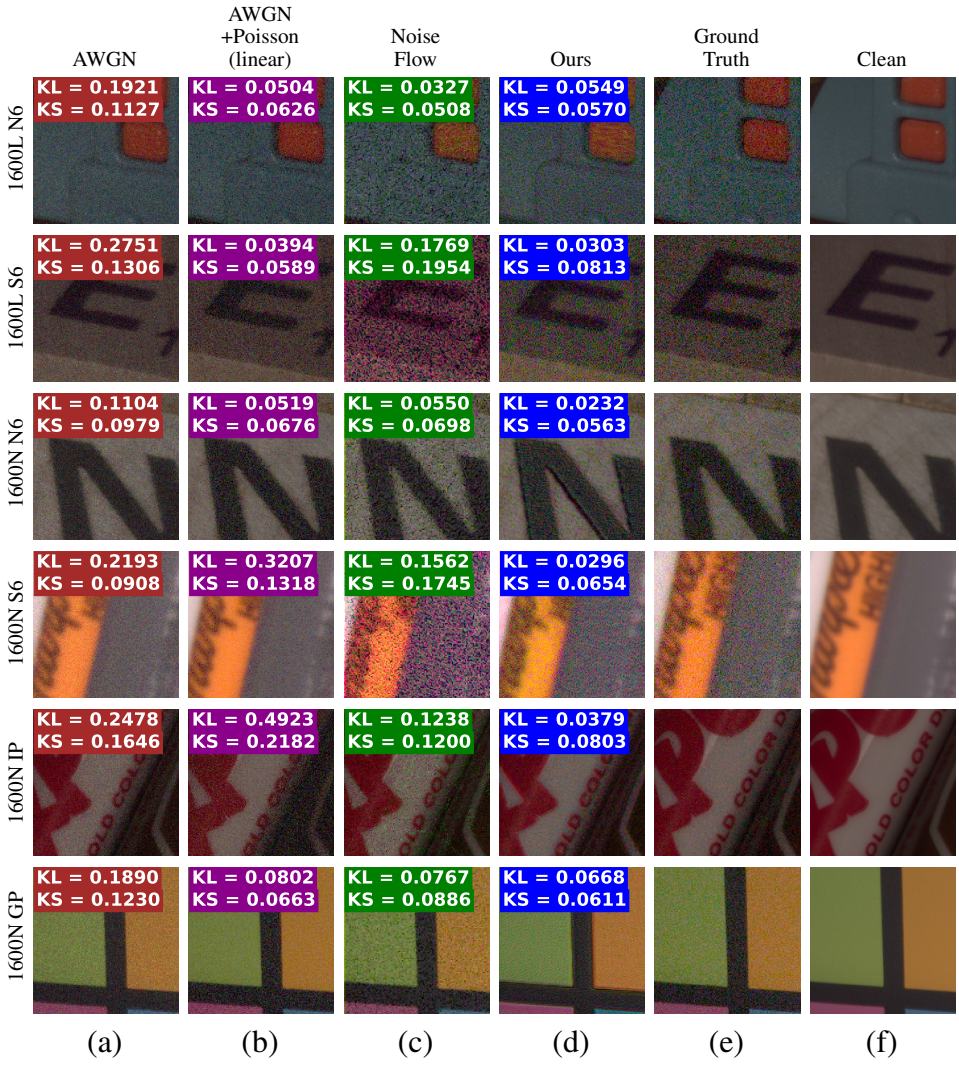


Figure A.4: Comparison among noise models for synthesizing noise for ISO 3200. (a) Patch corrupted by AWGN (using average std computed from the paired dataset, in this case $\sigma_{3200} = 0.1352$); (b) patch corrupted by AWGN+Poisson in linear space; (c) patch corrupted by Noise Flow (applied in raw space); (d) patch corrupted by our GAN_{SIDD}; (e) the noisy patch; and (f) the clean patch.

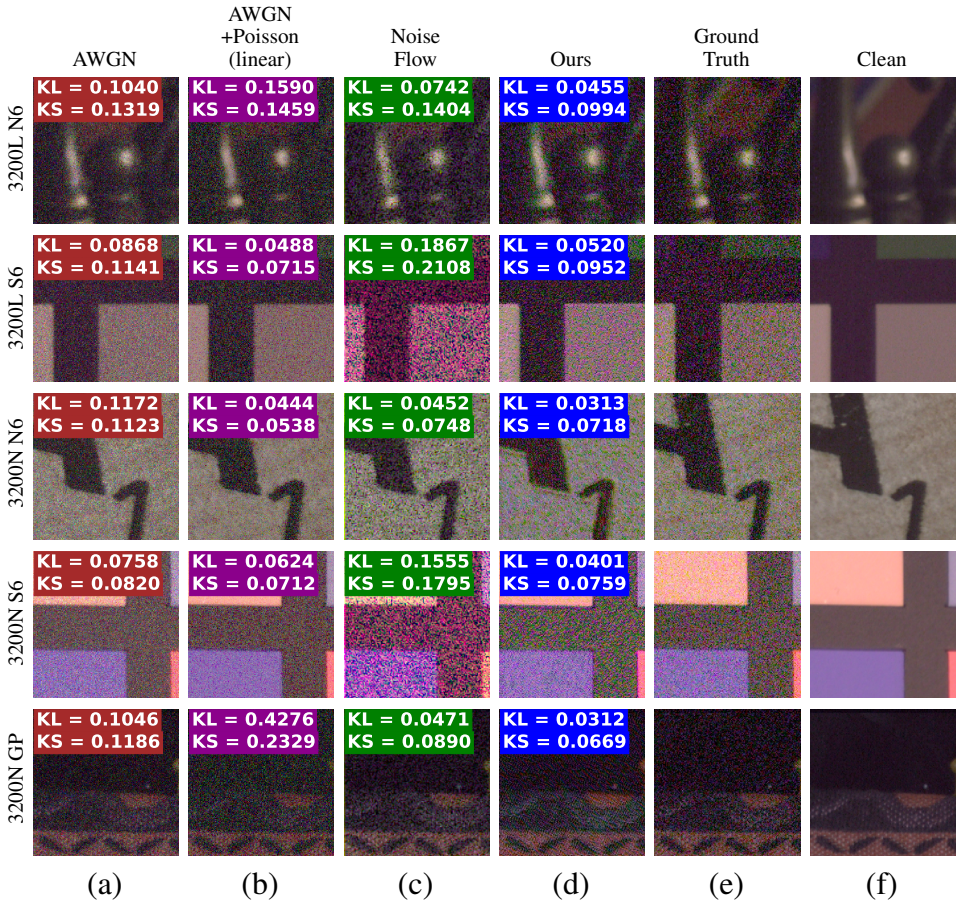


Figure A.5: Comparison of synthesized noisy images and corresponding noise (residual) produced by the various methods. For better visualization, contrast of the residual images has been enhanced by factor of $3\times$. Our results better mimic the noise found in digital photographs (note noise grain and color distribution).