Image Reconstruction with a GAN Gaussian Denoiser

Master's Thesis

submitted by

Benjamin Wilhelm

at the

Universität Konstanz

Faculty of Sciences

Department of Computer and Information Science

1. **Reviewer:** Prof. Dr. Bastian Goldlücke
2. **Reviewer:** Prof. Dr. Oliver Deussen

Konstanz, 2022

Benjamin Wilhelm
Image Reconstruction with a GAN Gaussian Denoiser
Master's Thesis, University of Konstanz, 2022.

Abstract

Many fundamental problems in image analysis and image processing are inverse problems. Examples are image reconstruction tasks like denoising, deblurring, super-resolution, inpainting, and demosaicing. Simple learning-based methods learn an end-to-end mapping between degraded and clean images to solve these tasks. Recently, deep neural networks have become the default choice. They are very effective but have the critical disadvantage of retraining if the task changes. Classical model-based methods rely on a prior; it is easy to adapt them to a new task. However, the choice of the prior is crucial. Plug-and-play methods for image reconstruction have been introduced using a Gaussian denoiser as a prior. In this work, we propose a new powerful denoiser called DRUGAN. Trained using the GAN framework, DRUGAN achieves compelling denoising results. It outperforms the perceptual quality of all other models by reconstructing edges and texture accurately. For more complex image reconstruction, we plug DRUGAN into two algorithms; the deep mean-shift prior (DMSP) [Big+17] algorithm and an iterative algorithm based on half quadratic splitting (HQS) [Zha+17b]. We evaluate the algorithms on non-blind deblurring, single image super-resolution, multi-frame super-resolution, and inpainting. We report all results and find that DRUGAN achieves outstanding results for non-blind deblurring with the HQS algorithm. Fine details and sharp edges are reconstructed much better than with other models. For other tasks, we observe that the algorithms can be adapted more easily to slightly changing objectives which is not the case for end-to-end learned methods.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Solving ill-posed inverse problems is a fundamental part of image analysis and image processing. Essential image reconstruction tasks like denoising, deconvolution, super-resolution, demosaicing, and inpainting are inverse problems. In general, inverse problems are the inverse of a forward problem. If the **forward problem** is applying a mathematical model M given its parameters m to obtain an observation o

$$o = M(m),$$

the **inverse problem** is to obtain the parameters m of a mathematical model M given the observation o.

$$m = M^{-1}(o)$$

An inverse problem is **ill-posed** if there is not exactly one unique solution or if the solution does not continually depend on the data [Gol17].

Classical approaches that solve inverse problems rely on optimization using known properties of the inverse problem and a regularizer or prior, which drives the solution to a reasonable image. We use the classical image degradation model [Big+17]

$$y = H\chi + n, \quad n \sim \mathcal{N}(0, \sigma_n^2), \tag{1.1}$$

where H is an operation on the image (blur for deconvolution, downscaling for single image super-resolution, masking regions for inpainting), χ is the unknown clear image, and n is Gaussian noise. A maximum a posterior (MAP) estimator can solve the inverse problem. To get the estimate $\hat{x}$ for the clear image χ, we

want to maximize the probability $p(x|y)$.

$$\begin{aligned}
\hat{x} &= \underset{x}{\operatorname{argmax}}[p(x|y)] \\
&= \underset{x}{\operatorname{argmax}}[p(y|x)\ p(x)] && \text{|Bayes theorem} \\
&= \underset{x}{\operatorname{argmax}}[\underbrace{\log(p(y|x))}_{\text{data}(x)} + \underbrace{\log(p(x))}_{\text{prior}(x)}] && \text{|Using log-likelihood}
\end{aligned}$$

Therefore, we end up maximizing a data term $\text{data}(x)$ and a prior term $\text{prior}(x)$. The data term ensures our estimate fits our observation y while the prior term drives our estimate to be a reasonable image.

$$\hat{x} = \underset{x}{\operatorname{argmax}}\left[\text{data}(x) + \text{prior}(x)\right] \tag{1.2}$$

Analogously, we can solve the inverse problem by minimizing an energy function

$$\begin{aligned}
\hat{x} &= \underset{x}{\operatorname{argmin}}[E(x)] \\
&= \underset{x}{\operatorname{argmin}}\left[\frac{1}{2\sigma_n^2}\left\|y - Hx\right\|^2 + \lambda\Phi(x)\right],
\end{aligned} \tag{1.3}$$

where $1/2\sigma_n^2 \left\|y - Hx\right\|^2$ is the data fidelity term, and $\lambda\Phi(x)$ is the regularizer or prior term [Zha+21a].

After defining a suitable prior that will force the solution to have the properties of an authentic image, an optimization algorithm can find the estimate. Methods like this are called **model-based** methods.

However, modern methods go another way. With the upswing of deep-learning-based image analysis methods, they were also used for inverse problems [MSY16; Zha+17a; Zha+21b; Don+16; Led+17; Wan+19b; Liu+18]. The general idea of **end-to-end learned** methods is to train the parameters of a machine learning model to map directly from the degraded image or observation to the clean image. Training data can be generated easily by applying the degradation model to a dataset of clean images. While these methods proved to be very effective in quality and speed, they have a significant disadvantage. Unlike model-based methods, these machine learning models must be retrained for each task. Depending on the specific model, retraining often requires a vast amount of computational resources (multiple days on a high-performance GPU).

To combine the versatility of model-based methods with the performance of deep learning, we will use a deep neural network to optimize the prior of the energy function. In particular, we will look at two proposed methods that utilize a denoising network to optimize the prior.

(a) Degradation model for denoising **(b)** Degradation model for deblurring

Figure 1.1

The first method was introduced by Bigdeli et al. [Big+17] and will be called DMSP. They defined a prior, representing a smoothed version of the natural image distribution. For optimizing this prior, they used the fact that the output of an optimal denoising network corresponds to the mean-shift on the data distribution.

The second method uses the Half Quadratic Splitting (HQS) algorithm. Zhang et al. [Zha+17b; Zha+21a] used a denoiser to solve the second subproblem of the HQS algorithm, which separates the prior from the data term.

Chapter 3 describes both methods in detail.

Degradation Models DMSP and HQS can be adapted to work on a wide range of image reconstruction problems using one single denoiser.

The denoiser will be learned to solve the simple degradation of additive white Gaussian noise with a certain standard deviation σ_n added to each pixel of the clean image. In the general degradation model in Equation (1.1), H is the identity. See Figure 1.1a for an illustration of the image degradation model that is assumed.

The DMSP and HQS algorithms can be used for deblurring, super-resolution, and inpainting. The imaging model assumed for deblurring is more complex than for denoising. Before adding noise, the image is blurred by a convolution with a blur kernel. Optimally, the blur kernel is just the point spread function of the imaging system. However, with a hand-held camera, motion blur is a relevant factor. Figure 1.1b shows the degradation model for deblurring with an authentic blur kernel by Levin et al. [Lev+09]. Note that the blur kernel can change across the image in practice [Lev+09], but this is not considered in this degradation model. If the blur kernel and the standard deviation of the noise are known, the task is called non-blind deblurring. If they are unknown, the problem gets much more complicated and is called blind deblurring.

The model used for super-resolution additionally assumes a downsampling operation after blurring the image. The downsampling operation selects each sth pixel for a downsampling factor of s. If only one low-resolution image is available, the task is called single image super-resolution. Figure 1.2 illustrates the degradation model for this task. If multiple images are observed, the task is called multi-frame super-resolution. In this case, the model also assumes that the in-

Figure 1.2: Degradation model for single image super-resolution

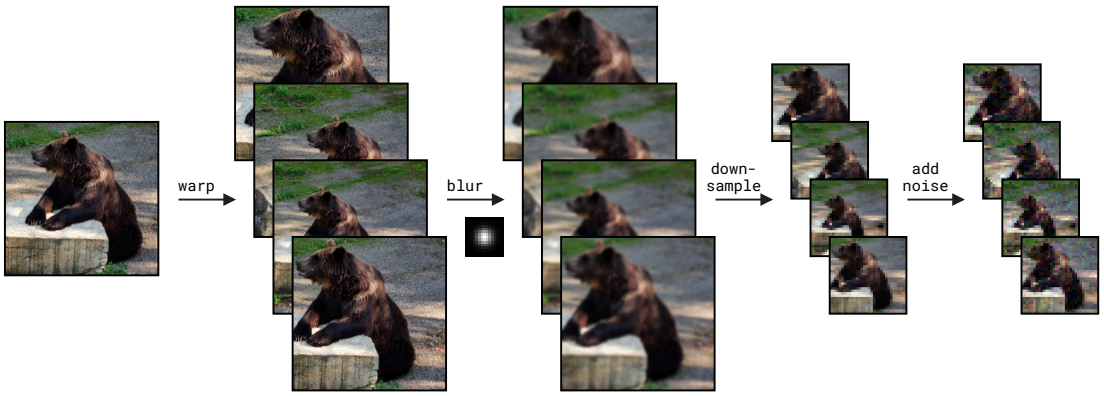


Figure 1.3: Degradation model for multi-frame super-resolution

dividual images are somehow warped compared to the clean ground truth image (offsetting the camera or motion in the frame). The frames must be warped differently and with sub-pixel differences because otherwise, there would not be any additional information compared to single image super-resolution. See Figure 1.3 for a visualization of the model using four frames.

The last reconstruction task that we will inspect is called image inpainting. The image degradation model assumes that a part of the image got missing. Figure 1.4 shows how a mask is applied to the image removing some information. Of course, the region could be deleted manually to remove unwanted objects from the image.

Section 3.3 will describe how the reconstruction task for the mentioned degradation problems can be solved using DMSP and HQS.

Figure 1.4: Degradation model for inpainting

Denoising GAN The main contribution of this work is to improve the visual quality of the reconstructed images by using the GAN framework to train a superior denoiser. In the GAN framework [Goo+14], a generator and a discriminator compete against each other. The generator is trained to generate artificial samples indistinguishable from samples of the data distribution. This is done by optimizing the parameters to fool the discriminator, trained to distinguish between artificial and real samples. Hence, the GAN framework forces the models to learn distinct properties of the data distribution. We will add a discriminator loss to a denoising neural network to use this. Our denoiser is trained to produce an image that is the denoised version of the input and to fool a discriminator. This forces the denoiser to focus more on details that improve the perceived quality of the image for the discriminator. In the results, we will see that this also improves the visual quality as perceived by a human. Using the GAN framework for image reconstruction follows the idea of SRGAN [Led+17] and ESRGAN [Wan+19b], where the authors trained GAN models for single image super-resolution. Section 3.4 describes in detail how the model is constructed and trained.

Contributions

- This work introduces a new Gaussian denoiser based on the GAN framework called DRUGAN. The model uses the powerful architecture of DRUNet but improves the training procedure. DRUGAN is compared to the state-of-the-art using a perceptual metric based on deep features. The denoised images have better visual quality than every other tested model. The perceptual quality improvement is evident in visual examples that are given in Chapter 4 and Appendix A.

- This work provides an implementation of two optimization algorithms that utilize a Gaussian denoiser to optimize a prior that relates to the distribution of natural images. The algorithms are applied to a range of image reconstruction tasks. For DMSP and HQS, solutions for non-blind image deblurring, single image super-resolution (bicubic and blur downscaling), and inpainting are provided. Only using HQS, the more complicated task of multi-frame super-resolution is solved and applied to videos and light fields.

- This work evaluates the optimization algorithms using the newly trained DRUGAN denoiser. The results for all tasks are compared to state-of-the-art methods showing that the DRUGAN denoiser can help to improve the perceptual quality of the result. Quantitative and qualitative results are given in Chapter 4 and Appendix A.

The source code is available online.

https://github.com/HedgehogCode/denoising-gan
: Training code for the DRUGAN denoiser and other denoiser models using TensorFlow 2 [Ten21].

https://github.com/HedgehogCode/deep-plug-and-play-prior
: Implementation of the DMSP and HQS algorithm using TensorFlow 2.

https://github.com/HedgehogCode/masters-thesis-evaluation
: Evaluation code.

Chapter 2

Related Work

This chapter will look at noteworthy publications related to this work. First, we will summarize work on image priors. Next, we will look at other image reconstruction methods for denoising, deblurring, single image super-resolution, multi-frame super-resolution, and inpainting. Finally, we will sum up work on Generative Adversarial Networks (GANs).

2.1 Image Priors

Classical choices of regularization include the L_2 and L_1 norm [Mit+09; Cha04]. For a reference of classical image priors, we refer to Shaham and Michaeli [SM16]. They explored a range of different image priors and demonstrated a method to visualize them.

There is quite some work on using a denoiser as an image prior for image restoration tasks. A BM3D denoiser [Dab+07] was used for image deblurring in [DKE10; DKE12]. In [EK15], the authors used a BM3D denoiser for single image super-resolution. A framework for image reconstruction using an arbitrary denoising model has been introduced by Venkatakrishnan et al. [VBW13]. They used the ADMM technique [Boy10] for the optimization. Brifman et al. [BRE16] applied this framework to single image super-resolution. Chan et al. [CWE17] explored the convergence of the framework and applied it to single image super-resolution and single-photon imaging. FlexISP [Hei+14] is a framework for camera image processing that uses the primal-dual algorithm [CP10]. The authors evaluated the framework for multiple tasks using different denoisers. Half Quadratic Splitting (HQS) was used by Zoran et al. [ZW11] for denoising, deblurring, and inpainting.

The natural next step is using deep-learning-based denoisers with the introduced frameworks. Zhang et al. [Zha+17b] used the HQS method for deblurring

and single image super-resolution. They made use of a simple DnCNN [Zha+17a] using dilated convolutions [YK16], batch normalization [IS15], and rectified linear units [NH10]. Since the HQS algorithm requires denoisers for different noise levels, they trained multiple networks. Later, Zhang et al. [Zha+21a] improved their work using another denoiser. They introduced DRUNet, a powerful denoiser based on the U-Net architecture [RFB15] with residual blocks [He+16]. DRUNet is trained on a range of noise levels by adding a noise-level-map input. Therefore, only one model is required when applying the HQS method. Bigdeli et al. [Big+17] used the Bayes' risk formulation from [JRF17] and a denoiser to compute the gradient for the prior. They used a simple network architecture for their denoiser.

Lempitsky et al. [LVU18] followed another approach. They used a randomly initialized network as a prior. The network architecture already forces the image to look realistic.

2.2 Image Reconstruction

This section will list relevant work related to the image reconstruction tasks handled in this work. The image degradation models are explained in the introduction (Chapter 1).

2.2.1 Denoising

Fan et al. [Fan+19] gave a brief overview of image denoising techniques. Classical methods work on the spatial domain and include filtering with different kernels and variational methods with different regularizations. Transform techniques first transform the image into another domain (like the wavelet transform [Mal89]) to apply the denoising technique. One powerful and prominent example is BM3D [Dab+07]. Recently, CNN-based methods have achieved outstanding performance. Zhang et al. [Zha+17a] have introduced DnCNN, a simple convolutional neural network with batch-normalization and a global residual. Later, he improved with FFDNet [ZZZ18], which has a noise-level map input and works on downsampled sub-images. He recently developed an architecture update to FFDNet, called DRUNet [Zha+21a]. DRUNet uses an architecture based on U-Net [RFB15] but with residual blocks. To our knowledge, DRUNet is the best denoiser currently available for additive Gaussian noise in terms of PSNR. In this work, we will make use of the DRUNet architecture.

Other network architectures have been introduced by Mao et al. [MSY16] and Zhang et al. [Zha+21b]. Both used their architectures on multiple image reconstruction tasks by retraining the network for the specific problem. In [MSY16], skip connections were introduced between an encoding and decoding

part. In [Zha+21b], the authors introduced residual dense blocks to boost the network performance of RDN.

Brooks et al. [Bro+19] introduced a technique called "unprocessing" to work with real camera image noise. Their pipeline allows them to generate training data corresponding to raw sensor data. Chen et al. [Che+18] used a GAN to generate noise patches that fit the noise distribution present in raw image data.

2.2.2 Deblurring

Mao et al. [MSY16] and Zhang et al. [Zha+21b] used their already mentioned network architectures for deblurring by training the networks for specific blur kernels. Both achieve good results, but their method is not flexible since the network needs to be retrained for each kernel and noise level.

Xu et al. [Xu+14] defined DCNN and ODCNN for non-blind deblurring. Their network architecture is based on the singular value decomposition of the pseudo inverse blur kernel. ODCNN adds a simple CNN with two hidden layers to reject outliers on top of DCNN. Again, their method has to be retrained for each blur kernel. However, they use the SVD of the pseudo inverse blur kernel to initialize the network parameters.

Kruse et al. [KRS17] used convolutional neural networks to replace parts of FFT-based deconvolution. Their method does not have to be retrained for new blur kernels or noise levels.

Most recent approaches try to solve the much more challenging task of blind deblurring. Nah et al. [NKL17] used a deep convolutional neural network that operates on multiple scales. Kupyn et al. [Kup+18] used the GAN framework to train DeblurGAN. In [Kup+19], they improved DeblurGAN and called it DeblurGAN-v2.

2.2.3 Single Image Super-Resolution

Nasrollahi and Moeslund created a survey of image super-resolution methods [NM14]. They classified methods based on the domain they operate in (frequency/spatial) and the number of low-resolution images used (single image/-multiple images). Most methods have been developed in the spatial domain. We will look at a few recent deep-learning-based methods for single image super-resolution.

Dong et al. [Don+16] introduced SRCNN, a simple convolutions network with two hidden layers for single image super-resolution. While Dong et al. did not see improvements when increasing the number of layers, Kim et al. [KLL16] achieved better results with a 20 layer CNN called VDSR. They added a global residual,

gradient clipping, and high learning rates to ensure convergence with this many layers. Ledig et al. [Led+17] improved the network architecture using ResNet [He+16] for SRResNet. Lim et al. [Lim+17] simplified the architecture of SRResNet and improved the PSNR values of the super-resolved images with EDSR. They also introduced a multiscale super-resolution network called MDSR that reuses most weights for different scales. While they only focused on PSNR performance, Ledig et al. [Led+17] used their SRResNet in the GAN framework to improve the perceptual quality of the super-resolved images. They call SRResNet trained with an adversarial loss SRGAN. It cannot compete in PSNR but achieves visually more pleasing results. Wang et al. [Wan+19b] built upon SRGAN for ESRGAN. They improved the network architecture of the generator and used a relativistic discriminator [Jol19].

2.2.4 Multi-Frame Super-Resolution

Again, we refer to the survey by Nasrollahi and Moeslund [NM14]. They categorized multi-frame super-resolution methods. Methods that use the imaging model to simulate low-resolution observations, compare them with the actual observations, and project the error back to the high-resolution grid to refine the high-resolution estimate, are called iterative back projection (IBP) methods. Direct methods warp the low-resolution frames to a high-resolution grid and combine them directly, which gives them a speed advantage over IBP methods. Probability-based methods try to find an ML solution or a MAP estimate of the high-resolution image.

Video Super-Resolution There are many deep-learning-based methods for video super-resolution.

Kappeler et al. [Kap+16] introduced the first CNN for video super-resolution called VSRnet. They used optical flow estimation and backward warping to compensate for motion. The motion compensation was improved by Tao et al. [Tao+17], who introduced a "sub-pixel motion compensation" layer. Jo et al. [Jo+18] went in another direction by avoiding explicit motion compensation. Wang et al. [Wan+19a] introduced EDVR and achieved very competitive results. They aligned frames using deformable convolutions [Dai+17] in a first module and used attention to find important features in the spatial and temporal axis in a second module. Recently, Chan et al. [Cha+21] tried to find a strong baseline for VSR. They developed BasicVSR and IconVSR, which are slightly better than EDVR but significantly less complicated and faster.

Light Field Super-Resolution For light field super-resolution, how the individual images relate to each other is well-defined. Therefore, only a depth map is required to register the images. Wanner and Goldlücke [WG12; WG14] did spatial and angular super-resolution using a variational framework. Their framework uses the total variation prior, is very flexible, and can synthesize super-resolved novel views. Yoon et al. [Yoo+15] first used CNNs for spatial and angular light field super-resolution. They significantly outperform classical methods.

Multiple improvements of CNN-based light field super-resolution have been made. Wang et al. [Wan+18] introduced LFNet for spatial light field super-resolution. They upscaled the horizontal and vertical center slice (the vertical/horizontal angle is fixed) separately and combined them later. To upscale the 3D slices, they used a bidirectional recurrent neural network with a novel "Implicit Multi-scale Fusion" (IMsF) layer. They demonstrated that their complex architecture achieves excellent results on multiple light fields. Zhang et al. [ZLS19] used the horizontal and vertical angular slices and additionally the diagonal slices. Their network resLF uses residual blocks to extract features from the four different slices. The features are then combined and upscaled to a high-resolution light field. The presented results look very promising, beating LFNet and EDSR [Lim+17] by a large margin. Jin et al. [Jin+20] finally used the whole light field for each view. Their method consists of two modules. The first module extracts features from each view and fuses them to upscale a reference view. The second module regularizes the result using a CNN consisting of alternating spatial and angular convolutions. Their method outperforms resLF.

2.2.5 Inpainting

Traditional approaches for digital image inpainting often try to continue the edge of the missing region into the region preserving structure [Ber+00; Oli+01; DH18]. Texture can be reconstructed by replicating texture from another location in the image [Ber+03].

There are multiple deep-learning-based methods. Pathak et al. [Pat+16] introduced Context Encoders that learn how to reproduce an image region using the surrounding context. Using an adversarial loss, they achieved sharper results for large missing regions. They showed that Context Encoders learn valuable features for classification, detection, and segmentation. Yang et al. [Yan+17] introduced a multiscale framework to improve the image resolution when inpainting. They reconstructed high-resolution texture by finding patches in the image using deep features. Iizuka et al. [ISI17] used two discriminators to train a model for inpainting. One of the discriminators checks if the inpainted image is globally consistent, while the other checks that local patches are reasonable. The network was trained to fool both, generating images with a realistic global structure and local texture.

Yu et al. [Yu+18] used a similar approach but introduced several improvements. Their main contribution is to use a contextual attention layer that learns how to borrow features from known parts of the image. Liu et al. [Liu+18] introduced partial convolutions that take the mask into account. They showed that this helps for inpainting if the holes are irregular.

2.3 Generative Adversarial Networks

The GAN framework was introduced by Goodfellow et al. [Goo+14]. They introduced a learning scheme where a discriminator model and a generator model are trained against each other. The goal of the discriminator is to differentiate between training examples of a data distribution from examples generated by the generator. The generator's goal is to fool the discriminator with the generated examples. This way, the generator learns to reproduce the data distribution. Using powerful models for the generator and discriminator and a large enough set of training samples, one can learn to generate adversarial samples of complex distributions. Goodfellow et al. demonstrated GANs on small images of numbers, faces, and objects.

Many papers showed the capability of the GAN framework for generating images. Radford et al. [RMC16] defined deep convolutional network architectures for the generator and discriminator. They demonstrated the architectures on slightly larger natural images of bedrooms, faces, and general objects. Additionally, they explored interpolation and vector arithmetic in the latent space and the features learned by the models. Arjovsky et al. [ACB17] improved GAN training with an algorithm named Wasserstein GAN (WGAN), stabilizing the training procedure. They used a critic instead of a discriminator that does not discriminate between real and fake samples but outputs a score on how real the sample is. Therefore, the critic cannot saturate and gives clean gradients for the generator. WGAN-GP [Gul+17] further improves the training stability by adapting the clipping of the critic weights. Jolicoeur-Martineau [Jol19] introduced another loss for the GAN framework using a relativistic discriminator. She called the models Relativistic GANs (RGANs) and Relativistic average GANs (RaGANs) and showed that they generate even higher quality images than WGAN-GP. RaGANs will be used in this work. The relativistic losses will be described in section 3.4.2.

Using progressive growing of GANs, Karras et al. [Kar+18] managed to generate very realistic high-resolution images. Later, they introduced a style-based generator architecture [KLA19] that generates even better images and controls the style.

For single image super-resolution, Ledig et al. [Led+17] introduced SRGAN. They used a residual convolutional network architecture for upscaling as a gener-

ator. This generator was trained to upscale low-resolution input images to high-resolution images using matched training data. They combined a content loss (MSE loss or VGG loss) with the adversarial loss. The adversarial loss improves the visual quality of the super-resolved images.

There is little work on denoising using GANs. Wang et al. [Wan+20] described a denoising GAN. However, they did not explain the essential details of the "smoothing loss" they use (the naming suggests something which goes against the idea of reproducing sharp details) and do not provide source code. They neither compared their method with state-of-the-art nor provided quantitative results that could be compared. Yan and Wang [YW17] used a DCGAN to do super-resolution, denoising, and deblurring. Their results do not look promising, and they do not provide source code.

Chapter 3

Method

The following two sections will look at two approaches to solve inverse reconstruction problems. Both methods use a denoising model that acts as a prior. Section 3.3 lists the details for four reconstruction problems using these approaches. A better denoising model based on the GAN framework to improve the algorithms is described in Section 3.4.

3.1 Deep Mean-Shift Priors (DMSP)

Bigdeli et al. [Big+17] did not directly optimize the MAP estimate but optimized an extension. They formulated a prior representing a smoothed version of the natural image distribution. The optimization problem can be solved using gradient descent, and the gradient of the prior can be computed using a learned denoiser. We will now follow the basics of the mathematical foundation of their method as described in their work.

As already mentioned, we do not solve the inverse problem via a maximum a posterior estimate. Instead, we will use a utility function G and maximize its posterior expectation. This gives us the Bayes estimator x.

$$E_{\tilde{x}}[G(\tilde{x}, x)] = \int G(\tilde{x}, x) p(y|\tilde{x}) p(\tilde{x})\, d\tilde{x} \tag{3.1}$$

The utility function (defined later) favors the arguments to be similar. An x that maximizes $E_{\tilde{x}}[G(\tilde{x}, x)]$ is a good estimator because many $\tilde{x}$ close to x have a high probability of being the clean image for the observation y.

We define the prior on a smoothed data distribution.

$$p'(x) := E_{\eta}[p(x + \eta)] = \int g_{\sigma}(\eta) p(x + \eta)\, d\eta, \tag{3.2}$$

where $\eta \sim \mathcal{N}(0, \sigma^2)$ and g_σ is a Gaussian kernel with the same distribution. Later, we will see that the gradient of the logarithm of this data distribution can be easily computed using a learned denoiser.

The same Gaussian kernel will now be used to define the utility function G.

$$G(\tilde{x}, x) := g_\sigma(\tilde{x} - x)\frac{p'(x)}{p(\tilde{x})} \tag{3.3}$$

Using the utility function G and smooth data distribution p', we can rewrite the posterior expectation of G.

$$\begin{aligned}
E_{\tilde{x}}[G(\tilde{x}, x)] &= \int G(\tilde{x}, x)p(y|\tilde{x})p(\tilde{x})\, d\tilde{x} \\
&= \int g_\sigma(\tilde{x} - x)\frac{p'(x)}{p(\tilde{x})}p(y|\tilde{x})p(\tilde{x})\, d\tilde{x} && \text{|eq. (3.3)} \\
&= \int g_\sigma(\tilde{x} - x)p'(x)p(y|\tilde{x})\, d\tilde{x} \\
&= \int g_\sigma(\tilde{x} - x)p(y|\tilde{x}) \int g_\sigma(\eta)p(x + \eta)\, d\eta d\tilde{x} && \text{|eq. (3.2)} \\
&= \int g_\sigma(\epsilon)p(y|x + \epsilon) \int g_\sigma(\eta)p(x + \eta)\, d\eta d\epsilon && \text{|Subst. } \epsilon = x - \tilde{x}
\end{aligned}$$

Note that we only have the prior on the smoothed data distribution left, which we can optimize using a learned denoiser.

The objective is given by maximizing the logarithm of posterior expectation. Jensen's inequality is used to split the term into a data term and a prior term.

$$\begin{aligned}
\log E_{\tilde{x}}[G(\tilde{x}, x)] &= \log\left(\int g_\sigma(\epsilon)p(y|x + \epsilon) \int g_\sigma(\eta)p(x + \eta)\, d\eta d\epsilon\right) \\
&\geq \int g_\sigma(\epsilon) \log\left(p(y|x + \epsilon) \int g_\sigma(\eta)p(x + \eta)\, d\eta\right) d\epsilon && \text{|Jensen's ineq.} \\
&= \underbrace{\int g_\sigma(\epsilon) \log p(y|x + \epsilon)\, d\epsilon}_{\text{data}(x)} + \underbrace{\log \int g_\sigma(\eta)p(x + \eta)\, d\eta}_{\text{prior}(x)}
\end{aligned}$$

We will maximize the logarithm of the posterior expectation using gradient descent with momentum. The estimate $\hat{x}$ is given by

$$\begin{aligned}
\hat{x} &= \operatorname*{argmax}_x \left[\log E_{\tilde{x}}[G(\tilde{x}, x)]\right] \\
&= \operatorname*{argmax}_x \left[\text{data}(x) + \text{prior}(x)\right].
\end{aligned}$$

Therefore, we get the update steps as shown in Algorithm 1. The data gradient $\nabla \text{data}(x)$ depends on the reconstruction task and will be defined in Section 3.3. The stochastic prior gradient $\nabla \text{prior}^s_L(x)$ will be defined in the following section.

Algorithm 1: Gradient descent for DMSP [Big+17].

Data: Observation y, Learning rate α, Momentum μ, Iterations T
Result: Estimate $\hat{x}$
initialize x_0
initialize velocity v with 0
for $t \leftarrow 1$ **to** T **do**
  $u_t \leftarrow -\nabla \operatorname{data}(x_{t-1}) - \nabla \operatorname{prior}_L^s(x_{t-1})$ `// Compute the gradient`
  $v \leftarrow \mu v - \alpha u_t$ `// Update the velocity`
  $x_t \leftarrow x_t + v$ `// Update the estimate`
$\hat{x} \leftarrow x_T$

3.1.1 Prior Gradient

A denoising autoencoder (DAE) $\mathcal{G}_\sigma^{\mathrm{DAE}}$ is a model with parameters w that maps from images to images of the same size. Usually, the parameters w are optimized to minimize "mean-squared error" (MSE) between the output of the DAE given an input that is corrupted by additive white Gaussian noise and the uncorrupted input.

$$\mathcal{L}_{\mathrm{MSE}} := \mathbb{E}_\eta \left[\left(\tilde{x} - \mathcal{G}_\sigma^{\mathrm{DAE}}(\tilde{x} + \eta) \right)^2 \right], \tag{3.4}$$

with $\eta \sim \mathcal{N}(0, \sigma^2)$.

Alain and Bengio [AB14] showed (in Theorem 1) that a perfect DAE has the output

$$\begin{aligned} \mathcal{G}_\sigma^{\mathrm{DAE}}(x) &= \frac{\mathbb{E}_\eta[p(x-\eta)(x-\eta)]}{\mathbb{E}_\eta[p(x-\eta)]} \\ &= \frac{\mathbb{E}_\eta[p(x-\eta)x] - \mathbb{E}_\eta[p(x-\eta)\eta]}{\mathbb{E}_\eta[p(x-\eta)]} \\ &= x - \frac{\mathbb{E}_\eta[p(x-\eta)\eta]}{\mathbb{E}_\eta[p(x-\eta)]}. \end{aligned} \tag{3.5}$$

They showed that this is valid for every kind of DAE, assuming that it has enough capacity and that the parameters are optimized to minimize the MSE over samples $\tilde{x}$ of the natural image manifold. Their proof considers additive white Gaussian noise, but other noise could also be used. On the other hand, we will need η to be Gaussian noise for the next steps.

Using the continuous formulation of Equation (3.5), we can reformulate the output of a perfect DAE to include the gradient of our prior.

$$\begin{aligned}
\mathcal{G}_\sigma^{\text{DAE}}(x) &= x - \frac{\int g_\sigma(\eta)p(x-\eta)\eta\, d\eta}{\int g_\sigma(\eta)p(x-\eta)\, d\eta} \\
&= x + \frac{\sigma^2 \int \nabla g_\sigma(\eta)p(x-\eta)\, d\eta}{\int g_\sigma(\eta)p(x-\eta)\, d\eta} && \text{|Gaussian derivative} \\
&= x + \frac{\sigma^2 \nabla \int g_\sigma(\eta)p(x-\eta)\, d\eta}{\int g_\sigma(\eta)p(x-\eta)\, d\eta} && \text{|Leibniz rule} \\
&= x + \sigma^2 \nabla \underbrace{\log \int g_\sigma(\eta)p(x-\eta)\, d\eta}_{\text{prior}(x)}
\end{aligned}$$

Therefore, we can compute the gradient of the prior using a DAE.

$$\nabla \text{prior}(x) = \frac{1}{\sigma^2}(\mathcal{G}_\sigma^{\text{DAE}}(x) - x)$$

We now have a simple way of computing the gradient for a valuable image prior.

However, in practice, a DAE is not very effective for images close to the manifold of noise-free images. It overfits to images with noise.

Therefore, we rewrite our prior and define a lower bound.

$$\begin{aligned}
\text{prior}(x) &= \log \int g_\sigma(\eta)p(x+\eta)d\eta \\
&= \log \int g_{\sigma_2}(\eta_2) \int g_{\sigma_1}(\eta_1)p(x+\eta_1+\eta_2)d\eta_1 d\eta_2 && |\sigma_1^2 + \sigma_2^2 = \sigma^2 \\
&\geq \int g_{\sigma_2}(\eta_2) \log \left[\int g_{\sigma_1}(\eta_1)p(x+\eta_1+\eta_2)d\eta_1 \right] d\eta_2 && \text{|Jensen's ineq.} \\
&= \int g_{\sigma_2}(\eta_2) \left[\text{prior}_{\sigma_1}(x+\eta_2) \right] d\eta_2 \\
&=: \text{prior}_L(x)
\end{aligned}$$

Using the new lower bound on the prior and $\sigma_1 = \sigma_2 = \sigma/\sqrt{2}$, we get the gradient

$$\begin{aligned}
\nabla \text{prior}_L(x) &= \int g_{\sigma/\sqrt{2}}(\eta_2) \left[\nabla \text{prior}_{\sigma/\sqrt{2}}(x+\eta_2) \right] d\eta_2 \\
&= \frac{2}{\sigma^2} \int g_{\sigma/\sqrt{2}}(\eta_2) \left[\mathcal{G}_{\sigma/\sqrt{2}}^{\text{DAE}}(x+\eta_2) - (x+\eta_2) \right] d\eta_2.
\end{aligned}$$

We cannot compute the integral over η_2. Instead, we use one noise sample $\eta_2 \sim \mathcal{N}(0, \sigma^2/2)$ during runtime, resulting in a stochastic gradient evaluation.

$$\boxed{\nabla \operatorname{prior}_L^s(x) = \frac{2}{\sigma^2}\left(\mathcal{G}^{\mathrm{DAE}}_{\sigma/\sqrt{2}}(x+\eta_2) - x\right)}$$

Note that we apply the DAE on a noisy image with the noise distribution it was trained on for this approximation of the prior gradient. Therefore, the DAE is as effective as it can be.

The theory for this gradient is only valid for a DAE trained with the MSE loss because this is the assumption in Equation (3.5). However, later we will introduce a denoiser trained with other losses. While this does not fit the theory, we will show empirically that this denoiser can be used. It produces better results than the denoiser trained with the MSE loss.

3.2 Half Quadratic Splitting (HQS)

The HQS algorithm [GY95] is a method to maximize the log-likelihood of a MAP estimate. Zhang et al. [Zha+17b; Zha+21a] used the HQS algorithm for image reconstruction using a denoiser as a prior. In the following section, we will replicate their method.

Remember the energy function (Equation (1.3)) from the introduction.

$$\hat{x} = \operatorname*{argmin}_x \left[\frac{1}{2\sigma_n^2}\|y - Hx\|^2 + \lambda\Phi(x)\right]$$

To split the data term from the regularization term, we use an auxiliary variable z which must be equal to x. We get the constrained problem

$$\hat{x} = \operatorname*{argmin}_x \left[\frac{1}{2\sigma_n^2}\|y - Hx\|^2 + \lambda\Phi(z)\right] \qquad s.t. \quad z = x.$$

Using the HQS method, we ensure the constraint is satisfied by adding the term $\mu/2\,\|x - z\|^2$ with a large penalty parameter μ.

$$\hat{x} = \operatorname*{argmin}_x \left[\frac{1}{2\sigma_n^2}\|y - Hx\|^2 + \lambda\Phi(z) + \frac{\mu}{2}\|x - z\|^2\right] \tag{3.6}$$

Since the data and prior terms are decoupled now, they can be solved separately in an iterative scheme.

$$x_t = \operatorname*{argmin}_x \left[\frac{1}{2\sigma_n^2}\|y - Hx\|^2 + \frac{\mu_t}{2}\|x - z_{t-1}\|^2\right] \tag{3.7}$$

$$z_t = \operatorname*{argmin}_z \left[\lambda\Phi(z) + \frac{\mu_t}{2}\|x_t - z\|^2\right] \tag{3.8}$$

The penalty parameter μ_t is increased with each iteration.

Equation (3.7) only contains the data term and depends on the specific reconstruction problem. We will look at the solution for this part in Section 3.3.

Equation (3.8) only contains the prior term and can be solved using a learned Gaussian denoiser $\mathcal{G}_\sigma$ [Zha+21a].

$$\begin{aligned} z_t &= \operatorname*{argmin}_z \left[\lambda \Phi(z) + \frac{\mu_t}{2} \left\| x_t - z \right\|^2 \right] \\ &= \operatorname*{argmin}_z \left[\frac{1}{2(\sqrt{\lambda/\mu_t})^2} \left\| x_t - z \right\|^2 + \Phi(z) \right] \\ &= \mathcal{G}_{\sqrt{\lambda/\mu_t}}(x_t) \end{aligned}$$

Note that because μ is a parameter that changes in every iteration, our denoiser needs to handle a range of noise levels (or we would need multiple denoisers).

Following [Zha+21a], we set μ_t implicitly by specifying the standard deviation of the denoiser $\sigma_t = \sqrt{\lambda/\mu_t}$ in each iteration and fixing $\lambda = 0.23$. This gives us $\mu_t = 0.23/\sigma_t^2$. For μ to increase, σ_t needs to decrease. We start with $\sigma_1 = 0.2$ ($0.2\times$ the range of the pixel values) and decrease the value logarithmically to the noise value in the last iteration $\sigma_T = \sigma_n$.

Algorithm 2: HQS algorithm for image restoration [Zha+21a].

Data: Observation y, Iterations T, Denoiser noise levels σ_t
Result: Estimate $\hat{x}$
initialize z_0
for $t \leftarrow 1$ **to** T **do**
  $x_t \leftarrow \operatorname{argmin}_x \left[\frac{1}{2\sigma_n^2} \left\| y - Hx \right\|^2 + \frac{\mu_t}{2} \left\| x - z_{t-1} \right\|^2 \right]$ `// Data solution`
  $y_t \leftarrow \mathcal{G}_{\sigma_t}(x_t)$ `// Prior solution`
$\hat{x} \leftarrow x_T$

3.3 Reconstruction Problems

We will now look at how to solve the data part of the optimization for four different reconstruction problems. Remember the classical degradation model from the introduction.

$$y = H\chi + n, \quad n \sim \mathcal{N}(0, \sigma_n^2)$$

For DMSP, we define data gradient $\nabla \text{data}(x)$, and for HQS, we solve Equation (3.7). Section 3.3.1 describes non-blind deblurring/deconvolution. Single image super-resolution and multi-frame super-resolution are described in Section 3.3.2 and Section 3.3.3, respectively. Finally, Section 3.3.4 describes the data solution for inpainting.

3.3.1 Deblurring

For deblurring/deconvolution, H is a matrix representing the convolution with the blur kernel k. For simplicity, we will focus on non-blind deconvolution, which means that the blur kernel k and the standard deviation of the additive Gaussian noise σ_n are known. Bigdeli et al. [Big+17] showed that the DMSP method is also applicable to noise- and kernel-blind deconvolution.

DMSP We get the data gradient

$$\boxed{\nabla \text{data}(x) = -\frac{1}{\sigma_n^2}\ \bar{k} * (k * x - y),}$$

where $\bar{k}$ is the flipped kernel.

HQS If circular boundary conditions are used for the convolution, there is a closed-form solution to Equation (3.7) [Zha+21a].

$$\boxed{x_t = \hat{f}^{-1}\left(\frac{\overline{\hat{f}(k)}\hat{f}(y) + \rho_k \hat{f}(z_{t-1})}{\overline{\hat{f}(k)}\hat{f}(k) + \rho_k}\right),}$$

where $\hat{f}(\cdot)$ is the Fast Fourier Transform, $\bar{\cdot}$ is the complex conjugate, and $\rho_k := \mu_k \sigma_n^2$.

3.3.2 Single Image Super-Resolution

For single image super-resolution H is a downscaling operation. Usually, it consists of two steps. First, the image is blurred with a blur kernel k. Next, the blurred image is downsampled by choosing every sth pixel for a scaling factor s (noted by $\downarrow^s$). We will not add any noise $\sigma_n = 0$. Therefore, we have the degradation model $y = (k * \chi) \downarrow^s$.

Another option for the downscaling operation is the well-known bicubic degradation (as done by the Matlab `imresize`). Bicubic downscaling can only be approximated using blur kernel and downsampling [ZGT20]. Therefore, we will use the slightly adapted degradation model $y = \chi \downarrow^s_{\text{bic}}$, where $\downarrow^s_{\text{bic}}$ denotes the bicubic downscaling operation.

DMSP The data gradient is

$$\boxed{\nabla \operatorname{data}(x) = -\bar{k} * ((k * x) \downarrow^s - y) \uparrow^s .}$$

For the case with bicubic downscaling, we get the gradient

$$\boxed{\nabla \operatorname{data}(x) = -(x \downarrow^s_{\text{bic}} - y) \uparrow^s_{\text{bic}} .}$$

Note that $\uparrow^s$ denotes upsampling without interpolation, and $\uparrow^s_{\text{bic}}$ denotes bicubic upscaling.

Since no noise is added to the degraded image, the data term has no natural weighting. We will use the value $w_t = {}^{w}/_{\sqrt{t}}$ to weight the prior gradient and decrease its influence in later iterations. The value of w_t starts with $w = 10^{-3}$ and decreases with every iteration.

HQS We solve Equation (3.7) using a closed-form solution similar to the non-blind deblurring task [Zha+21a]. We get

$$\boxed{x_t = \hat{f}^{-1}\left(\frac{1}{\rho_k}\left(d - \overline{\hat{f}(k)} \odot^s \frac{(\hat{f}(k)d) \Downarrow^s}{(\overline{\hat{f}(k)}\hat{f}(k)) \Downarrow^s + \rho_k}\right)\right),}$$

with

$$d = \overline{\hat{f}(k)}\hat{f}(y \uparrow^s) + \rho_k \hat{f}(z_{t-1}).$$

$\Downarrow^s$ averages the $s \times s$ regions of the image. $\odot^s$ is a kind of element-wise multiplication. Each value on the left-hand side is multiplied with the value on the right-hand side corresponding to the $s \times s$ region.

The above approach could be used with the approximated blur kernel for bicubic downscaling. However, the bicubic case can also be solved using an iterative approach by repeating the following computation five times [Zha+17b].

$$\boxed{x_t = z_{t-1} - \alpha(y - z_{t-1} \downarrow^s_{\text{bic}}) \uparrow^s_{\text{bic}}}$$

The step size α is set to 1.5.

3.3.3 Multi-Frame Super-Resolution

For multi-frame super-resolution (MFSR), we do not have only one degraded observation y However, we have multiple observations $y_1, \dots, y_N$. The degradation operations consist of a warping operation W_i followed by a blur operation with the kernel k and a downsampling operation $\downarrow^s$ [Mit+09]. Therefore, we get the degradation model

$$y_i = (k * W_i \chi) \downarrow^s + n_i, \quad n_i \sim \mathcal{N}(0, \sigma_n^2), i = 1 \dots N.$$

Remember the illustration of the multi-frame super-resolution model in Figure 1.3 in the introduction.

We get the following data sub-problem for the HQS algorithm.

$$x_t = \underset{x}{\operatorname{argmin}} \left[\frac{1}{2\sigma_n^2} \sum_{i=1}^{N} \|(k * W_i x) \downarrow^s - y_i\|^2 + \frac{\mu_t}{2} \|x - z_{t-1}\|^2 \right]$$

This can easily be solved by an iterative back-projection (IBP) scheme [NM14; IP91; Mit+09]. The idea is to simulate the image degradation process (see Figure 1.3) on the estimated high-resolution image. Suppose the high-resolution estimation is coherent with the low-resolution observations. In that case, the simulation result should be equal to the observations. The error between the simulated low-resolution image and the observation is computed and back-projected to refine the estimate. This is done by up-sampling the error, undoing the blur, and warping the error back to the reference. Formally, this results in the iterative scheme

$$\boxed{x_t = z_{t-1} + \alpha \left[\sum_{i=1}^{N} W_i^T \left(\bar{k} * ((y_i - (k * W_i z_{t-1}) \downarrow^s) \uparrow^s) \right) \right].}$$

W_i^T is the warping operation from the domain of the ith image to the domain of the reference image. Note that for bicubic downscaling, the blur operations vanish.

Five iterations with $\alpha = 0.2$ are executed for blur downscaling, and for bicubic downscaling, only one iteration with $\alpha = 1$ is executed.

Registration

A critical element of this algorithm is warping the images to a reference frame. The details of the warping operation are usually unknown. Therefore, we have to estimate a good W_i for each frame. This can be done by computing the pixel displacement $u_{i,k}$ between the images y_i and y_k. Without occlusion, noise, or illumination changes, the pixel displacement would fulfill the following condition.

$$y_i(\varphi) = y_k(\varphi + u_{i,k}(\varphi))$$

We can see that the pixel displacement defines a warping which we can use.

Videos For videos, $u_{i,k}$ is the optical flow. There are many methods to estimate the optical flow. Classically they are divided into local methods like Lukas-Kanade [LK81] and global methods like Horn-Schunk [HS81]. For larger pixel displacements, a coarse-to-fine scheme can be used. The finer image is warped such that the magnitude of the flow on this resolution level is not too high using this coarse optical flow [Bro+04]. In this work, we will use `pyflow` [Pat21]. `Pyflow` is a wrapper for Liu's C++ optical flow library [Liu09]. Liu's algorithm is based on [Bro+04; BWS05]. We use `pyflow` to compute the displacements between the reference frame and nine other frames by inputting them directly into the library. Larger displacements between nonconsecutive frames are handled sufficiently well by the coarse-to-fine scheme.

Light fields For light fields, only a disparity map has to be estimated. In this work, we will use LFattNet [Tsa+20], which is a CNN-based method. They use attention to select important views from the entire light field and achieve top-of-the-line results in the 4D Light Field Benchmark [Hon+17; Joh+17]. It is easy to integrate LFattNet because the authors provide source code and model weights.

Note that the super-resolution scheme from above could be improved by considering occlusion as done in [WG12]. We do not implement this improvement and leave it for future work.

3.3.4 Inpainting

For inpainting, H applies a mask to the image. We do not add noise to the image ($\sigma_n = 0$).

DMSP The data gradient is

$$\boxed{\nabla \operatorname{data}(x) = -(Hx - y).}$$

We use the weight w_t as in the single image super-resolution case with $w = 0.2$ to weight the prior term. Also, since we are certain about the non-masked values of the image, we set them to the values of x after the last iteration. With the decreasing weight of the prior, they converge to these values, but they might not be there when the last iteration ends.

HQS Solving Equation (3.7), we get

$$\boxed{x_t = Hy + H^{-1} z_{t-1},}$$

where H^{-1} is the inverse mask, masking all and only the values that H does not mask.

3.4 Denoising GANs

The previous sections show that denoising models can act as a prior for various inverse image reconstruction problems. To improve the reconstruction performance, we try to train a better denoiser. This section describes the main contribution of this work—the DRUGAN denoiser based on the GAN framework [Goo+14].

In our case, the generator $\mathcal{G}$ of the GAN does not generate samples based on a random input noise but denoises noisy images z. The Discriminator $\mathcal{D}$ predicts if the sample is a clean image drawn from the data distribution or a denoised image $\mathcal{G}(z)$.

The following section describes the architecture of the generator, which is just a denoising network. After that, Section 3.4.2 describes how this denoising network is incorporated in the GAN framework to improve the visual quality of the denoised images.

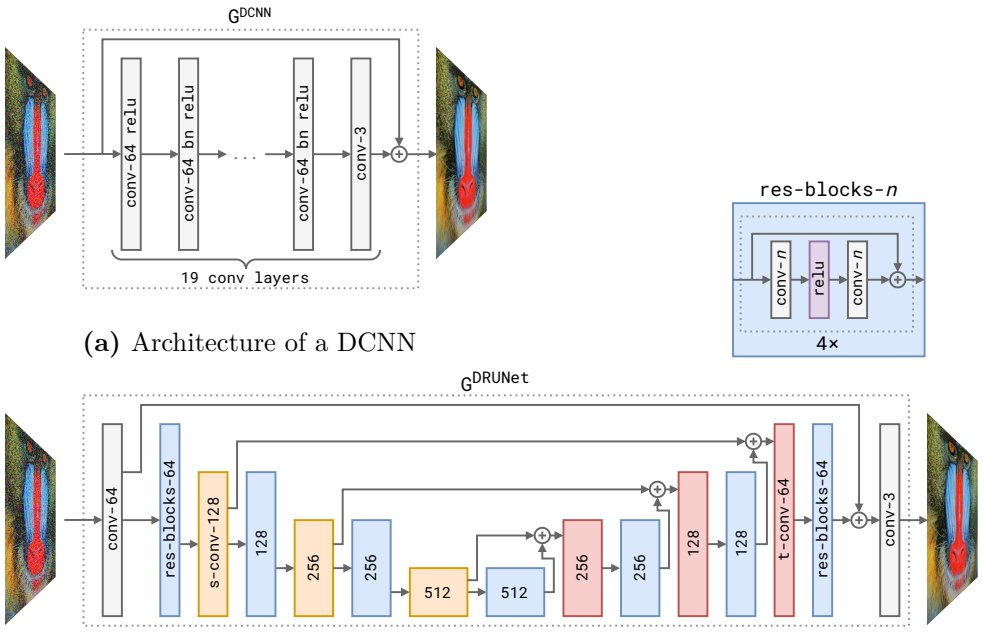
3.4.1 Denoising Autoencoder

Since our generator network is just a denoising network, we will explore the architecture of the generator by training different approaches using only the MSE loss (See Equation (3.4)).

DCNN: Denoising CNN As a base model, we use the architecture of DnCNN by Zhang et al. [Zha+17a] with 19 convolutional layers as displayed in Figure 3.1a. This model has around 630K trainable parameters.

DRCNN: Denoising Residual CNN The first modification is to use internal residuals. The input of each block is the sum of the input and the output of the previous block. This modification does not increase the number of trainable parameters.

DUNet: Denoising U-Net U-Net is defined by Ronneberger et al. [RFB15] and uses an encoder-decoder architecture. In the encoder part, the image resolution is decreased by using max-pooling. In the decoder part, the resolution is increased using transpose convolutions. A skip connection between the encoder part of a resolution and the decoder part of the equal resolution concatenates the higher resolution features to the upscaled decoded features. We use a U-Net with



(a) Architecture of a DCNN

(b) Architecture of a DRUNet

Figure 3.1: Network architectures of the denoiser

four resolution levels and 64 filters for the first resolution level, resulting in a model with around 8.6M trainable parameters.

DRUNet: Denoising Residual U-Net Zhang et al. [Zha+21a] defined the DRUNet architecture, which is a U-Net with residual blocks (note that using residual block in a U-Net is not a new idea [ZLW18; Ven+18]). Figure 3.1b shows the precise architecture.

They train the DRUNet on a range of noise levels. The noise level of the input is provided by concatenating the image with the noise map. This is useful for the HQS algorithm, which relies on denoisers for different noise levels. Models trained on a range of noise levels will be denoted by a subscript showing the range (e.g., $\text{DRUNet}_{\sigma_1 \ldots \sigma_2}$).

Training Details All models were trained for 4M steps using the Adam optimizer [KB17] with a learning rate of 10^{-4}. Each batch consisted of 16 random crops of size 96×96. The simple models were trained on the DIV2K training dataset [AT17]. Later models were trained on the DIV2K dataset, the Imagenette dataset [How21], the Waterloo exploration dataset [Ma+17], and the Flickr2K dataset [Lim+17] combined. Models trained on multiple datasets will be labeled with a "+" in the superscript of the model name. If the validation PSNR on the

DIV2K validation dataset did not increase for 100 steps, the learning rate was decreased by a factor of 0.2.

3.4.2 Generative Adversarial Networks

Embedding the denoising network into a GAN framework can improve the denoising performance further. A discriminator will not help increase the PSNR of the denoised images but will improve the visual quality.

The denoiser from the previous subsection was trained using the mean-squared error loss. This loss corresponds directly to the PSNR metric. However, it does not correspond well to the visual quality of an image. Other losses have been used to overcome this issue. The mean-absolute error is more robust against outliers and sometimes considered a better candidate [Zha+21a; Wan+19b].

$$\mathcal{L}_{\text{MAE}} := \mathbb{E}\left[|x - \mathcal{G}_\sigma(x+\eta)|\right] \tag{3.9}$$

Others have used a VGG19 model to create a **perceptual loss** that correlates closer to the human visual system [JAF16; Led+17; Wan+19b]. The idea is that the VGG19 model, trained to classify natural images, extracts features similar to the features extracted by a human brain. Comparing these features of two different images is a better measure of how similar the images appear to a human.

$$\mathcal{L}_{\text{VGG}kl} := \mathbb{E}\left[\left(\text{VGG}_{k,l}(x) - \text{VGG}_{k,l}(\mathcal{G}_\sigma(x+\eta))\right)^2\right], \tag{3.10}$$

where $\text{VGG}_{k,l}(x)$ is the feature map of a VGG-19 [SZ15] model after the lth convolutional layer before the kth max-pooling layer. In line with [Wan+19b], we will use the features before activation.

While we will use the VGG54 loss as an additional loss, we will focus on improving the model using the GAN framework [Goo+14]. The discriminator $\mathcal{D}$ is trained to discriminate between images from the real non-noisy data distribution x and the denoised images $\mathcal{G}(x+\eta)$. The loss is defined a follows.

$$\mathcal{L}_{\mathcal{D}} := -\mathbb{E}\left[\log(\mathcal{D}(x))\right] - \mathbb{E}\left[\log(1 - \mathcal{D}(\mathcal{G}_\sigma(x+\eta)))\right] \tag{3.11}$$

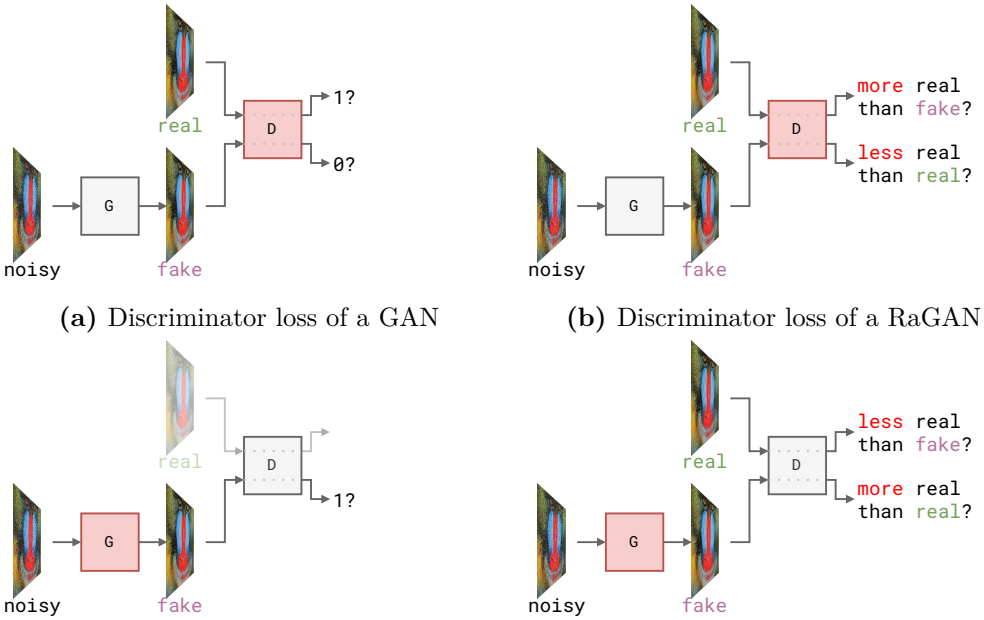
The generator is trained to fool the discriminator. Therefore, its **adversarial loss** is defined to drive the discriminator to predict 1 for denoised images.

$$\mathcal{L}_{\text{ADV}-\mathcal{G}} := -\mathbb{E}\left[\log(\mathcal{D}(\mathcal{G}_\sigma(x+\eta)))\right]. \tag{3.12}$$

See Figures 3.2a and 3.2c for a visualization of the loss.

To force the generator to create images that are the denoised version of the input image and not just some image that fools the discriminator, we combine the $\text{GAN} - \mathcal{G}$ loss with the MAE and VGG54.

$$\mathcal{L}_{\mathcal{G}} := \lambda\mathcal{L}_{\text{ADV}-\mathcal{G}} + \tau\mathcal{L}_{\text{MAE}} + \mathcal{L}_{\text{VGG54}} \tag{3.13}$$



(a) Discriminator loss of a GAN

(b) Discriminator loss of a RaGAN

(c) Generator loss of a GAN

(d) Generator loss of a RaGAN

Figure 3.2: Visualization of the losses of a GAN and a RaGAN

Relativistic Discriminator Jolicoeur-Martineau [Jol19] defined a relativistic average GAN. Wang et al. [Wan+19b] used this and found that it helped improve the visual quality of super-resolved images compared to a classical discriminator. The relativistic discriminator is trained to predict higher values for clean data than for denoised data and the other way around. We define the discriminator

$$\mathcal{D}_{Ra}(a, b) := \sigma(C(a) - \mathbb{E}_b\left[C(b)\right]), \tag{3.14}$$

where $C(\cdot)$ is the CNN without activation function, σ is the sigmoid activation function, and $\mathbb{E}_b[C(b)]$ is the mean over the whole mini-batch of samples. The output is close to 1 if the CNN output for a is much higher than the average CNN output for b. If it is much lower, the output is close to 0.

The loss of the relativistic discriminator is

$$\mathcal{L}_{\mathrm{Ra}\mathcal{D}} := -\mathbb{E}\left[\log(\mathcal{D}_{Ra}(x, \mathcal{G}_\sigma(x+\eta)))\right] - \mathbb{E}\left[\log(1 - \mathcal{D}_{Ra}(\mathcal{G}_\sigma(x+\eta), x))\right]. \tag{3.15}$$

The **adversarial loss** of the generator using a relativistic discriminator is

$$\mathcal{L}_{\mathrm{ADV-Ra}\mathcal{G}} := -\mathbb{E}\left[\log(1 - \mathcal{D}_{Ra}(x, \mathcal{G}_\sigma(x+\eta)))\right] - \mathbb{E}\left[\log(\mathcal{D}_{Ra}(\mathcal{G}_\sigma(x+\eta), x))\right]. \tag{3.16}$$

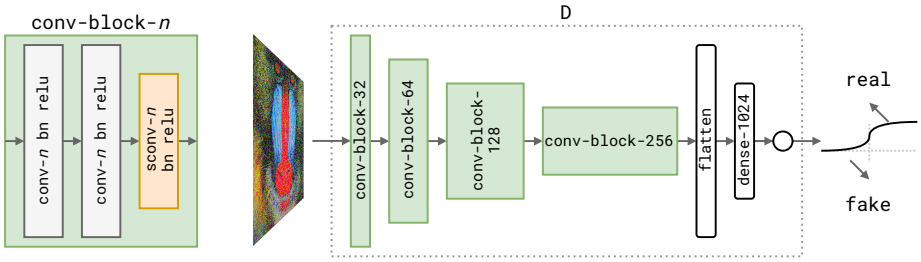


Figure 3.3: Architecture of the discriminator model

The losses using a realistic average GAN are visualized in Figures 3.2b and 3.2d. The discriminator is trained to predict that real images are more real than fake ones in a batch. Fake images must be more fake than the real images in a batch. The generator needs to fool the discriminator. Therefore, it has the opposite objective.

Like the standard GAN, we combine the ADV $-$ Ra$\mathcal{G}$ loss with the MAE and VGG54.

$$\mathcal{L}_{\text{Ra}\mathcal{G}} := \lambda \mathcal{L}_{\text{ADV}-\text{Ra}\mathcal{G}} + \tau \mathcal{L}_{\text{MAE}} + \mathcal{L}_{\text{VGG54}} \tag{3.17}$$

Training Details The generator was initialized with the weights of DRUNet$^{+}_{0...0.2}$ after 700K steps, while the discriminator was initialized with random weights. Analogously to the denoiser training, the discriminator and the generator were optimized using the Adam optimizer with a learning rate of 10^{-4}. Batches of 16 random crops of size 96×96 of the DIV2K, Imagenette, Waterloo exploration, and Flickr2K dataset were used. For $\mathcal{L}_{\text{Ra}\mathcal{G}}$, we set $\lambda = 0.5$ and $\tau = 15$. Training with the standard GAN was too unstable. Therefore, we will not report results for the standard GAN. A simple CNN architecture was used for the discriminator model (see Figure 3.3).

Chapter 4

Results

In this chapter, we will evaluate the DRUGAN denoiser on the tasks of denoising, non-blind deblurring, single image super-resolution, and inpainting. First, we will look at the denoising performance of the DRUGAN itself and compare it with state-of-the-art Gaussian denoisers. Next, we will use the DRUGAN denoiser for more complex reconstruction tasks using the DMSP and HQS algorithms. Throughout the evaluation, we will show results for intermediate models that do not use the GAN framework and use various architectures described in the previous chapter. The source code for all experiments executed can be found online at `https://github.com/HedgehogCode/masters-thesis-evaluation`.

Note that we will use a value range from 0 to 1 for image points throughout the work, and all experiments were conducted on RGB color images. Appendix A lists more qualitative results and an exhaustive list of quantitative results.

Metrics

PSNR The Peak signal-to-noise ratio is a well-known measure for the quality of a reconstructed image. The PSNR can be defined using the mean-squared error (MSE) [Wik21].

$$PSNR(x, y) = 10 \cdot \log_{10} \left(\frac{\text{max\_val}^2}{MSE(x, y)} \right)$$

However, the MSE and, therefore, the PSNR is not a good measure for the perceptual quality of an image [Zha+11; Zha+18]. Therefore, other metrics have been developed.

SSIM Another prominent metric is the Structural Similarity Index [Wan+04]. By assessing structural information of the image, SSIM tries to be a better perceptual metric.

Additive noise σ_n	0.05		0.10		0.20	
Model	PSNR	LPIPS	PSNR	LPIPS	PSNR	LPIPS
CBM3D [MAF20]	34.45	0.0639	30.67	0.1543	27.36	0.2897
CDnCNN-B [Zha+17a]	34.77	0.0538	31.12	0.1132	27.82	0.2170
DnCNN (DMSP) [Big+17]	33.57	0.0477	—	—	—	—
DRUNet (DPIR) [Zha+21a]	**35.17**	0.0472	**31.56**	0.0990	**28.38**	0.1841
$\text{DCNN}_{0.05}$	34.64	0.0512	—	—	—	—
$\text{DRCNN}_{0.05}$	34.88	0.0498	—	—	—	—
$\text{DUNet}_{0.05}$	34.69	0.0508	—	—	—	—
$\text{DUNet}^{+}_{0.05}$	34.84	0.0503	—	—	—	—
$\text{DRUNet}^{+}_{0.05}$	33.62	0.0528	—	—	—	—
$\text{DUNet}^{+}_{0\ldots0.2}$	34.82	0.0492	31.22	0.1033	28.01	0.1946
$\text{DRUNet}^{+}_{0\ldots0.2}$	<u>35.11</u>	0.0466	<u>31.50</u>	0.0972	<u>28.31</u>	0.1807
$\text{DRUGAN}^{+}_{0\ldots0.2}$ ($\lambda = 0$)	34.96	<u>0.0280</u>	31.31	<u>0.0630</u>	28.11	<u>0.1265</u>
$\text{DRUGAN}^{+}_{0\ldots0.2}$	34.54	**0.0260**	30.80	**0.0575**	27.48	**0.1196**

Table 4.1: Denoising results for the CBSD68 [Mar+] dataset. The **best** and the <u>second-best</u> results are marked.

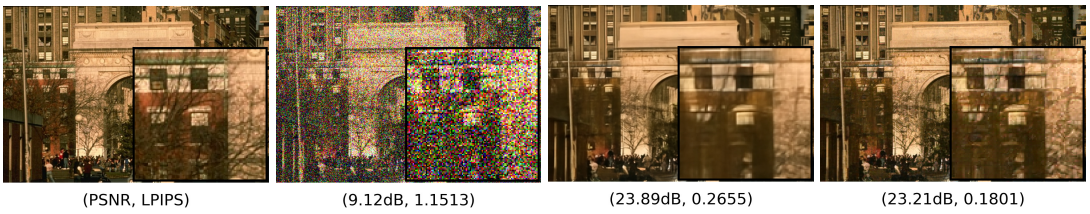
FSIM The feature similarity index for image quality assessment has been defined by Zhang et al. [Zha+11]. It is based on features from the phase congruency and the gradient magnitude. They showed that the FSIM is very consistent with subjective image quality and, therefore, better suited as a perceptual metric than PSNR or SSIM.

LPIPS More recently, Zhang et al. [Zha+18] explored features from deep neural network architectures for perceptual similarity. They found that deep features across architectures and training datasets significantly outperform other metrics (including PSNR, SSIM, and FSIM). We will use their LPIPS metric with AlexNet [KSH12] and the default settings in version 0.1 (linear layers on top of the trunk network, trained with human perceptual judgments). Lower values of the LPIPS metric are better than higher values.

This chapter will quantify the results using the PSNR and LPIPS metrics. The quantitative evaluation in Appendix A lists all four metrics.

4.1 Denoising

To evaluate the Denoising GAN's performance and different denoiser architectures, we ran the models for Gaussian denoising with different noise levels. The experiment was conducted on the CBSD68 [Mar+] dataset, which is commonly used for denoising evaluation. Appendix A lists additional results on other datasets.



(a) GT **(b)** Noisy **(c)** $\text{DRUNet}^{+}_{0...0.2}$ **(d)** $\text{DRUGAN}^{+}_{0...0.2}$

Figure 4.1: Denoising results on one image of the CBSD68 [Mar+] dataset with a noise standard deviation of $\sigma_n = 0.35$

The results can be found in Table 4.1. We can see that all models are very competitive with state-of-the-art Gaussian denoisers (DRUNet [Zha+21a]). The architecture changes did not provide a substantial performance improvement for models only trained for $\sigma_n = 0.05$. The PSNR values even dropped when switching to a U-Net, which is most likely due to overfitting the training dataset. Using a training dataset consisting of multiple large datasets (as described in the training details in Section 3.4.1) improved the performance again.

When training the models for a range of noise levels, we can see that using the more powerful architecture DRUNet improved the performance compared to DUNet. $\text{DRUNet}^{+}_{0...0.2}$ can be compared to the DRUNet trained by Zhang et al. [Zha+21a] (called "DRUNet (DPIR)" in the table). We can see that their implementation achieved the best PSNR values out of all methods. However, our DRUNet implementation achieved slightly lower LPIPS values. This indicates that they managed to optimize their model very well on the PSNR metric at the cost of visual quality.

The best LPIPS values were achieved by the model introduced by this work—the DRUGAN. It significantly improved the LPIPS metric and, therefore, the visual quality for all noise levels. Both added losses, the perceptual loss (Equation (3.10)) and the adversarial loss from a relativistic discriminator (Equation (3.16)), contributed to this improvement. Setting $\lambda = 0$ (deactivating the adversarial loss) already improved the LPIPS values. However, setting $\lambda = 0.5$ resulted in the best model trained. This shows that the relativistic discriminator helps train a model with excellent visual quality.

A visual comparison between the DRUNet and DRUGAN can be found in Figure 4.1. For visualization purposes, a noise level of $\sigma_n = 0.35$ was used—which is outside the noise level range for both models. Luckily, the denoiser architecture DRUNet generalizes very well for unseen noise levels, as stated in [Zha+21a]. We can see that DRUGAN produced less smooth results than DRUNet. The result for DRUGAN is perceptually closer to the ground truth, and the LPIPS value is better compared to DRUNet. On the other hand, the PSNR value is better for

Method	Additive noise σ_n	0.01		0.02		0.03		0.04	
		PSNR	LPIPS	PSNR	LPIPS	PSNR	LPIPS	PSNR	LPIPS
FDN [KRS17]		23.32	0.3265	23.81	0.4388	23.08	0.5058	22.64	0.5485
DMSP [Big+17]		26.63	0.2239	24.93	0.3920	23.85	0.4908	23.12	0.5540
DPIR [Zha+21a]		**29.09**	0.1700	<u>26.64</u>	0.3176	**25.43**	0.4057	**24.67**	0.4621
DMSP	$\text{DCNN}_{0.05}$	27.85	0.1821	25.71	0.3525	24.48	0.4528	23.64	0.5209
	$\text{DRCNN}_{0.05}$	27.90	0.1810	25.75	0.3552	24.51	0.4568	23.69	0.5249
	$\text{DUNet}_{0.05}$	27.67	0.1895	25.39	0.3658	24.08	0.4700	23.24	0.5410
	$\text{DRUNet}^{+}_{0.05}$	26.01	0.2073	23.46	0.3899	22.34	0.4893	21.54	0.5544
	$\text{DRUNet}^{+}_{0\ldots0.2}$	25.57	0.1589	24.92	0.2804	24.40	0.3688	23.90	0.4321
	$\text{DRUGAN}^{+}_{0\ldots0.2}$	23.65	0.1800	22.26	<u>0.2639</u>	22.38	<u>0.3335</u>	22.71	<u>0.3895</u>
HQS	$\text{DRUNet}^{+}_{0\ldots0.2}$	<u>29.08</u>	<u>0.1559</u>	**26.66**	0.3007	<u>25.42</u>	0.3946	<u>24.61</u>	0.4584
	$\text{DRUGAN}^{+}_{0\ldots0.2}$	27.62	**0.0725**	25.38	**0.1653**	24.45	**0.2496**	23.88	**0.3188**

Table 4.2: Non-blind deblurring results for a subset of the BSDS500 [Arb+11] validation dataset

DRUNet, which produces an over-smoothed image. This is an excellent example that PSNR does not relate to visual quality but prohibits texture reconstruction if accurate information is missing.

4.2 Deblurring

Table 4.2 shows the results for non-blind deblurring. Our implementations of DMSP and HQS are compared to the results from FDN [KRS17], the original DMSP paper [Big+17], and the HQS implementation by Zhang et al. DPIR [Zha+21a].

We can see that the HQS algorithm worked much better for lower noise levels and our results for the HQS algorithm are very comparable with the original implementation. DPIR archived the best PSNR values for most noise levels, just as for denoising. However, the LPIPS values were slightly worse. Using the DRUGAN for HQS gave better LPIPS values for all noise levels and overall best results. This shows that our powerful denoiser, which focuses more on visual quality than on PSNR, can be used effectively for the HQS algorithm to improve the visual quality of the reconstructed image.

Using the DMSP algorithm, the results are more complicated to interpret. For models trained on a range of different noise levels, noise with a standard deviation of 0.1 was added to the image for the stochastic evaluation of the prior gradient ($\sigma/\sqrt{2} = 0.1$). This works better for tasks with a higher noise level. See Figure 4.2 for a visualization of PSNR and LPIPS values when using different noise levels for the prior gradient. We can see that lower values around 0.05 work well to get good PSNR values. For higher noise in the image, the values should be slightly

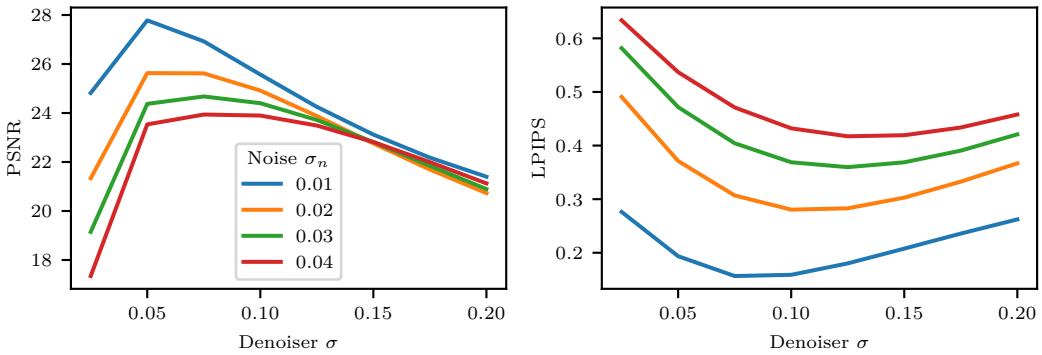


Figure 4.2: PSNR and LPIPS values for the DMSP non-blind deblurring algorithm using different noise levels for the stochastic evaluation of the prior gradient. The $\text{DRUNet}^{+}_{0...0.2}$ model was used to get these values.

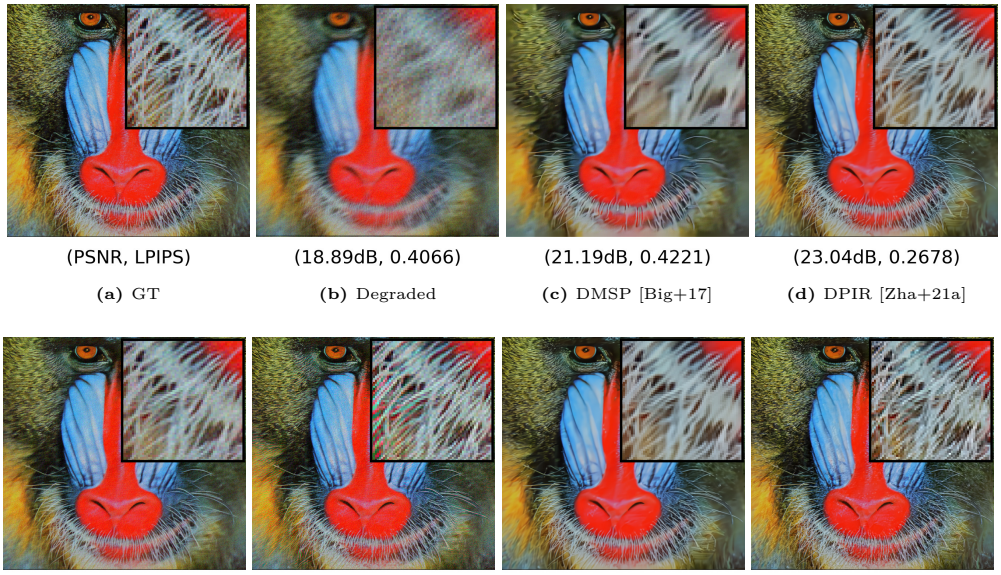


Figure 4.3: Non-blind deblurring results on the image "Baboon" of Set14. The image was degraded using the first blur kernel of [Lev+09] and with a noise standard deviation of $\sigma_n = 0.04$.

higher. However, good LPIPS values are achieved with higher noise levels used for the prior. Again, for higher noise in the image, the noise levels used for the prior should be higher. Judging from these graphs, 0.1 seems to be a good compromise for different levels of noise focused on the LPIPS metric.

Therefore, DRUNet and DRUGAN got worse PSNR values for the DMSP algorithm when $\sigma_n < 0.03$. The LPIPS values, on the other hand, were competitive for all noise levels. DMSP with DRUNet even beat DPIR using the LPIPS metric. Like for HQS, we got the best LPIPS values for DMSP and the second-best results of all methods when using DRUGAN. Only for very little noise of $\sigma_n = 0.01$ DRUNet achieved a better LPIPS value.

Figure 4.3 provides a visual comparison between the different methods using DRUNet and DRUGAN. We can see that DRUGAN reconstructed the texture of fine details much better. The hair is less smoothed out but fine and sharp. This is also reflected in the LPIPS value. DRUGAN achieved a much better visual quality with both algorithms than the existing methods in Figures 4.3c and 4.3d.

In Section 3.1, the theory for the prior gradient of the DMSP algorithm required the denoiser to be trained with the MSE loss. However, the results show that the DMSP algorithm works well with the DRUGAN model. The better properties of the DRUGAN model are reflected in the deblurring results. While the theoretical explanation is still missing, the natural assumption that the DMSP algorithm profits from better denoisers is valid.

4.3 Single Image Super-Resolution

Table 4.3 shows the results for single image super-resolution using bicubic downscaling. We can see that our implementation achieved PSNR values comparable to the values reported in the DMSP paper [Big+17]. There is no significant difference between the different models for DMSP, but DRCNN seems to do surprisingly well. The HQS algorithm achieved similar results to the DMSP algorithm.

The end-to-end learned method EDSR [Lim+17] was evaluated for scaling factors 2, 3, and 4. ESRGAN [Wan+19b] was included for a scaling factor of 4. These methods were evaluated using the BasicSR toolbox [Wan+21]. They were trained specifically for the individual downscaling factors and bicubic downscaling by the authors of the respective method. For these cases, they achieved much better results than the model-based methods. Notably, the LPIPS result for ESRGAN is better than all other methods by a large margin. ESRGAN was (similar to DRUGAN) trained with a perceptual loss and an adversarial loss which explains the remarkable result in the perceptual metric.

Figure 4.4 visualizes super-resolved images using bicubic downscaling. There is no notable difference between the DMSP and HQS algorithm and the differ-

(PSNR, LPIPS)
(a) GT

(20.25dB, 0.5014)
(b) Bicubic

(22.37dB, 0.2172)
(c) EDSR [Lim+17]

(19.53dB, 0.1372)
(d) ESRGAN [Wan+19b]

(21.49dB, 0.3679)
(e) DMSP-DRUNet

(21.47dB, 0.3402)
(f) DMSP-DRUGAN

(21.35dB, 0.3612)
(g) HQS-DRUNet

(21.49dB, 0.3289)
(h) HQS-DRUGAN

Figure 4.4: Single image super-resolution results on the image "Comic" of Set14 with bicubic downscaling

Method	Scaling factor s	2 PSNR	2 LPIPS	3 PSNR	3 LPIPS	4 PSNR	4 LPIPS	5 PSNR	5 LPIPS
Bicubic		31.81	0.1282	28.62	0.2543	26.70	0.3423	25.29	0.4052
EDSR [Lim+17]		**36.03**	**0.0544**	**32.59**	**0.1231**	**30.47**	0.1714	—	—
ESRGAN [Wan+19b]		—	—	—	—	28.47	**0.0752**	—	—
DMSP [Big+17]		35.16	—	31.38	—	29.16	—	27.38	—
IRCNN [Zha+17b]		35.07	—	31.26	—	29.01	—	27.13	—
DMSP	$\text{DCNN}_{0.05}$	35.17	0.0627	31.56	0.1405	29.31	0.2029	27.32	**0.2583**
	$\text{DRCNN}_{0.05}$	35.30	0.0642	31.66	0.1434	29.46	0.2071	**27.53**	0.2626
	$\text{DUNet}_{0.05}$	34.79	0.0663	31.03	0.1480	29.03	0.2158	26.75	0.2788
	$\text{DRUNet}^{+}_{0.05}$	32.86	0.0718	28.86	0.1622	27.11	0.2387	25.24	0.3130
	$\text{DRUNet}^{+}_{0...0.2}$	34.97	0.0692	31.32	0.1586	29.16	0.2362	27.37	0.2978
	$\text{DRUGAN}^{+}_{0...0.2}$	35.07	0.0608	31.52	0.1411	28.44	0.2250	26.59	0.2889
HQS	$\text{DRUNet}^{+}_{0...0.2}$	34.97	0.0751	31.24	0.1598	29.12	0.2267	27.30	0.2838
	$\text{DRUGAN}^{+}_{0...0.2}$	34.35	0.0573	31.27	0.1370	28.83	0.2109	26.98	0.2748

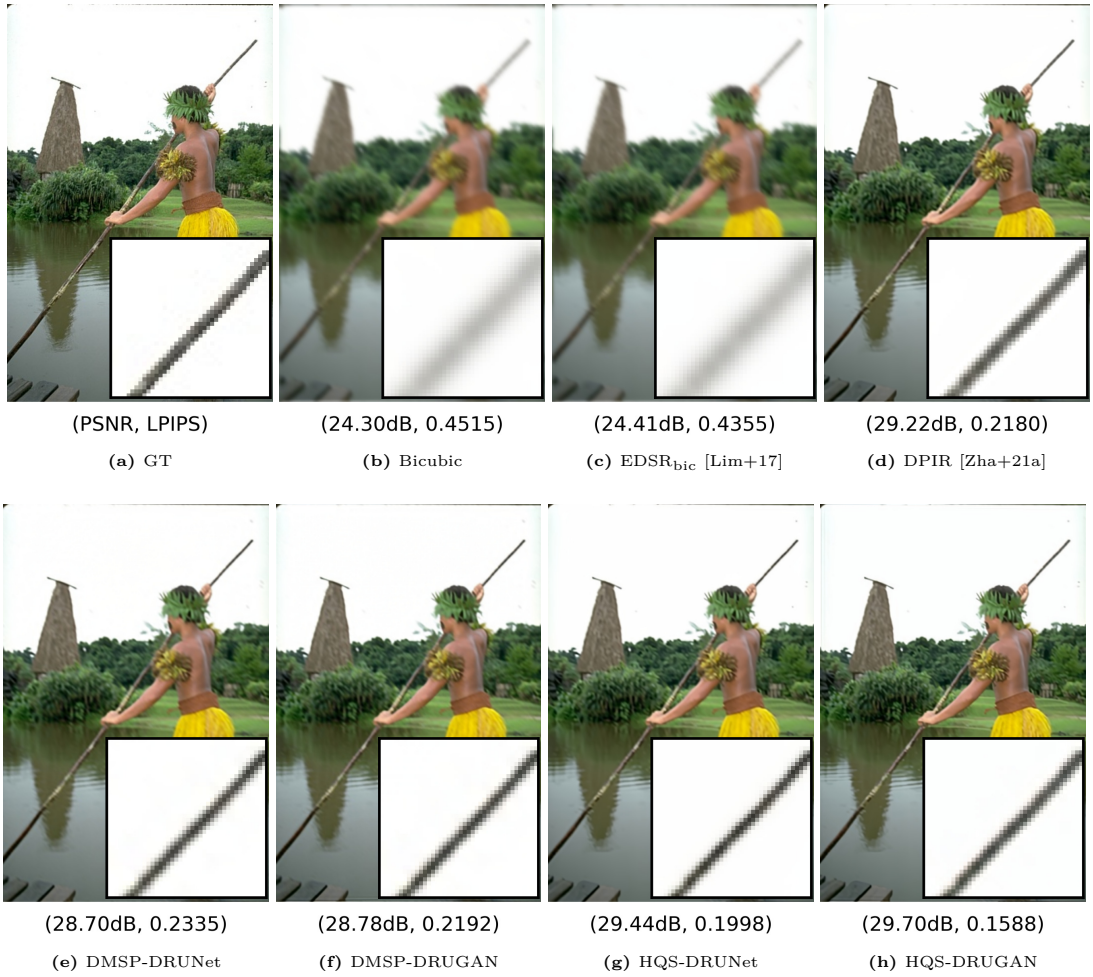
Table 4.3: Single image super-resolution results for Set5 with bicubic downscaling. The results for IRCNN and DMSP were copied from the DMSP paper.

ent denoising models. While the reconstructed images look sharper than bicubic interpolation, they are still not very sharp. The end-to-end learned frameworks, however, reconstructed a much sharper image. They introduced some artifacts, but edges were reconstructed considerably better. Especially when not zoomed in extremely close, the results of ESRGAN are more visually appealing than the results of model-based methods.

We can see that our model-based methods can not compete with end-to-end learned methods for bicubic downscaling. However, model-based methods have an advantage over end-to-end frameworks. They can easily be adapted and applied to different tasks. The end-to-end methods cannot handle different scales and downscaling operations. On the other hand, model-based methods can be adapted easily to handle these new tasks.

Table 4.4 shows results for super-resolution using blur downscaling with four isotropic Gaussian kernels and four anisotropic Gaussian kernels. We can see that EDSR failed horribly. This is not surprising because the model was not trained for this downscaling operation. EDSR would need to be retrained (possibly for each blur kernel) to get good results, which requires a vast amount of data, time, and energy. Our methods using DMSP and HQS can handle blur downscaling and achieve better results than DPIR. Here we can see again that DRUGAN generally gets the best LPIPS values.

Figure 4.5 shows an example of super-resolution with blur downscaling. An anisotropic Gaussian blur kernel was used, which blurred the stick extremely. Since EDSR is unaware of the blur kernel, it failed to reconstruct the stick. The model-based methods reconstructed the image well.



(a) GT **(b)** Bicubic **(c)** EDSR_{bic} [Lim+17] **(d)** DPIR [Zha+21a]

(e) DMSP-DRUNet **(f)** DMSP-DRUGAN **(g)** HQS-DRUNet **(h)** HQS-DRUGAN

Figure 4.5: Single image super-resolution results on an image of the CBSD68 [Mar+] dataset with blur downscaling. An anisotropic Gaussian blur kernel was used.

Method	Blur kernels	isotropic				anisotropic			
	Scaling factor s	2		3		2		3	
		PSNR	LPIPS	PSNR	LPIPS	PSNR	LPIPS	PSNR	LPIPS
Bicubic		26.56	0.2649	24.35	0.3089	24.44	0.3843	23.47	0.3959
EDSR_{bic} [Lim+17]		26.76	0.2251	23.97	0.2216	24.50	0.3730	23.59	0.3499
DPIR [Zha+21a]		34.13	0.1085	30.83	0.1735	32.21	0.1625	30.71	0.1931
DMSP	$\text{DCNN}_{0.05}$	33.84	0.1064	31.18	0.1424	31.10	0.1738	30.68	0.1798
	$\text{DRCNN}_{0.05}$	33.93	0.1088	**31.31**	0.1455	31.26	0.1764	30.85	0.1824
	$\text{DUNet}_{0.05}$	33.51	0.1125	30.49	0.1506	30.70	0.1846	30.07	0.1915
	$\text{DRUNet}^{+}_{0.05}$	31.44	0.1213	28.37	0.1691	28.59	0.1988	27.81	0.2093
	$\text{DRUNet}^{+}_{0...0.2}$	33.84	0.1071	30.95	0.1522	31.33	0.1820	30.62	0.1934
	$\text{DRUGAN}^{+}_{0...0.2}$	34.09	0.0926	31.04	**0.1363**	31.59	0.1603	30.82	0.1723
HQS	$\text{DRUNet}^{+}_{0...0.2}$	**34.31**	0.0982	31.03	0.1617	**32.51**	0.1468	30.95	0.1760
	$\text{DRUGAN}^{+}_{0...0.2}$	33.62	**0.0735**	31.04	0.1375	32.28	**0.1182**	**31.07**	**0.1523**

Table 4.4: Single image super-resolution results for Set5 with blur downscaling. Four isotropic and four anisotropic Gaussian kernels from [Zha+21a] were used. Models marked with the subscript "bic" denote end-to-end frameworks trained for bicubic downscaling.

4.4 Multi-Frame Super-Resolution

Evaluation of multi-frame super-resolution was done on the four videos of the Vid4 [LS11] dataset and the light fields of the HCI [Hon+17] test set.

Table 4.5 shows the results of Vid4. The values reported were computed on the center frame of the video. EDSR and ESRGAN are single image super-resolution models. Therefore, they got only the center image of the video as input. EDVR got seven frames as input because it was trained for this amount of frames. For the HQS algorithm, nine frames were considered.

We can see that for bicubic downscaling end-to-end frameworks achieve the best results. Although EDSR and ESRGAN had less information than the HQS algorithm, they performed better. EDVR got the best PSNR value for $\times 4$ upscaling with bicubic downscaling. However, ESRGAN got a lower LPIPS value, indicating that the results are perceptually better.

The end-to-end learned methods failed again for blur downscaling because they were not adapted to the task, which would be costly. The HQS algorithm was easy to adapt and worked well. When comparing DRUNet and DRUGAN, we see that DRUNet has higher PSNR values while DRUGAN wins the perceptual LPIPS metric. This result is as expected.

Figure 4.6 shows the results of upscaling the center frame of the small "Bookcase 1" sequence. The original video has a resolution of 91×121 and was upscaled with a scaling factor of 4. This is a blind problem because the blur kernel is unknown. We can see that end-to-end learned methods produce undesirable visual artifacts

Method	Downscaling	bicubic				isotropic blur		anisotropic blur	
	Scaling factor s	2		4		4		4	
		PSNR	LPIPS	PSNR	LPIPS	PSNR	LPIPS	PSNR	LPIPS
Bicubic		26.8374	0.2175	22.2613	0.5066	20.0150	0.5324	20.0248	0.6008
EDSR_{bic} [Lim+17]		**30.5931**	**0.1036**	23.9269	0.3253	19.0251	0.3780	20.0205	0.4945
$\text{ESRGAN}_{\text{bic}}$ [Wan+19b]		—	—	21.6173	**0.1903**	17.2765	0.3681	19.5474	0.4312
EDVR_{bic} [Wan+19a]		—	—	**25.5342**	0.2477	17.7089	0.3620	19.5031	0.4777
HQS	$\text{DRUNet}^{+}_{0...0.2}$	27.6802	0.1847	22.9253	0.4030	**22.0766**	0.3563	**21.8919**	0.4187
	$\text{DRUGAN}^{+}_{0...0.2}$	27.0509	0.1369	22.8392	0.3838	21.6548	**0.3338**	21.8897	**0.3949**

Table 4.5: Video super-resolution results for the Vid4 [LS11] dataset. For blur downscaling four isotropic and four anisotropic Gaussian kernels from [Zha+21a] were used. Models marked with the subscript "bic" denote end-to-end frameworks trained for bicubic downscaling.

(a) Bicubic **(b)** EDSR_{bic} [Lim+17] **(c)** $\text{ESRGAN}_{\text{bic}}$ [Wan+19b]

(d) EDVR_{bic} [Wan+19a] **(e)** HQS-DRUNet **(f)** HQS-DRUGAN

Figure 4.6: Video super-resolution results on the "Bookcase 1" video [Mil21]. The high-resolution image and the downscaling kernel are unknown. Each method assumed bicubic downscaling.

Method	Downscaling	bicubic				isotropic blur		anisotropic blur	
	Scaling factor s	2		4		4		4	
		PSNR	LPIPS	PSNR	LPIPS	PSNR	LPIPS	PSNR	LPIPS
Bicubic		30.9892	0.1942	26.8588	0.4491	24.2972	0.4736	24.2566	0.5369
EDSR_{bic} [Lim+17]		34.8349	0.0884	29.4590	0.2617	23.0932	0.3247	24.1839	0.4295
$\text{ESRGAN}_{\text{bic}}$ [Wan+19b]		—	—	26.6828	**0.1289**	21.0068	0.3442	23.7343	0.3699
EDVR_{bic} [Wan+19a]		—	—	**30.7001**	0.2025	22.0967	0.3266	23.7275	0.4276
$\text{LFSSR}_{\text{bic}}$ [Jin+20]		**35.3224**	**0.0360**	29.5067	0.2417	22.8422	0.3095	24.0424	0.4037
HQS	$\text{DRUNet}^{+}_{0..,0.2}$	30.8832	0.1731	28.3765	0.3485	**27.6333**	0.2813	**26.4208**	0.3877
	$\text{DRUGAN}^{+}_{0...0.2}$	29.4760	0.1386	28.2575	0.3265	27.3270	**0.2545**	26.3081	**0.3679**

Table 4.6: Light field super-resolution results for the HCI [Hon+17] test set. For blur downscaling four isotropic and four anisotropic Gaussian kernels from [Zha+21a] were used.

because they cannot handle this form of downscaling. The results of our method look much better. While the text was not reconstructed correctly, the images look much smoother overall while preserving sharp edges. For the HQS algorithm, bicubic downscaling was assumed. Given that the end-to-end frameworks failed dramatically, this does not seem to be correct. However, the HQS algorithm still produced much better results. The reason for this is the regularizer which forces the output to be a plausible image even if the data term does not fit perfectly.

Table 4.6 shows the results for light field super-resolution. Only the center image of the light field was upscaled. Again, EDSR and ESRGAN only had the center image as an input. The input for EDVR consisted of every 11th image of the light field resulting in 7 frames. LFSSR was trained on 7×7 light field frames. Therefore, the outermost frames of the given 9×9 light field were cropped off. The HQS algorithm had the whole 9×9 light field as an input.

As for video super-resolution, HQS did not perform as well as end-to-end learned methods for bicubic downscaling. Surprisingly, LFSSR, which was trained explicitly for light field super-resolution and had almost the entire light field as input, performed worse than EDVR and ESRGAN. ESRGAN achieved by far the best LPIPS values. For downscaling with a blur kernel, the results are analog to the results for video super-resolution. The end-to-end frameworks for bicubic downscaling failed, and HQS performed much better. DRUGAN reached better LPIPS values, while DRUNet was better in PSNR.

The quality of the depth estimation influences how well the HQS super-resolution algorithm works. We ran super-resolution with ground truth disparities and compared them to the results with estimated disparities to evaluate the influence. With ground-truth disparities of the "additional" light field of the HCI dataset with bicubic downscaling and a scaling factor of 4, we got a mean PSNR of 31.05 and an LPIPS of 0.3207. In contrast with estimated disparities, we got a

(PSNR, LPIPS) (26.93dB, 0.4802) (25.20dB, 0.3151) (22.56dB, 0.3523)

(a) GT **(b)** Bicubic **(c)** EDSR_{bic} [Lim+17] **(d)** $\text{ESRGAN}_{\text{bic}}$ [Wan+19b]

(23.41dB, 0.3732) (24.30dB, 0.3801) (30.69dB, 0.2947) (30.21dB, 0.2734)

(e) EDVR_{bic} [Wan+19a] **(f)** $\text{LFSSR}_{\text{bic}}$ [Jin+20] **(g)** HQS-DRUNet **(h)** HQS-DRUGAN

Figure 4.7: Light field super-resolution results on the "Bedroom" image of the HCI [Hon+17] test dataset. An isotropic Gaussian kernel and a scaling factor of 4 was used.

mean PSNR of 30.40 and LPIPS of 0.3277. This shows that we could improve the results by improving the disparity estimation. However, the differences are not very drastic. The disparity estimation does not seem to be a handicap.

Figure 4.7 shows the results of upscaling the "Bedroom" image of the HCI dataset. Blur downscaling was used. Some end-to-end learned methods show visual artifacts—especially with the wall's texture. This is expected because they were not trained for the downscaling used. The HQS results do look slightly blurry but introduce no visual artifacts.

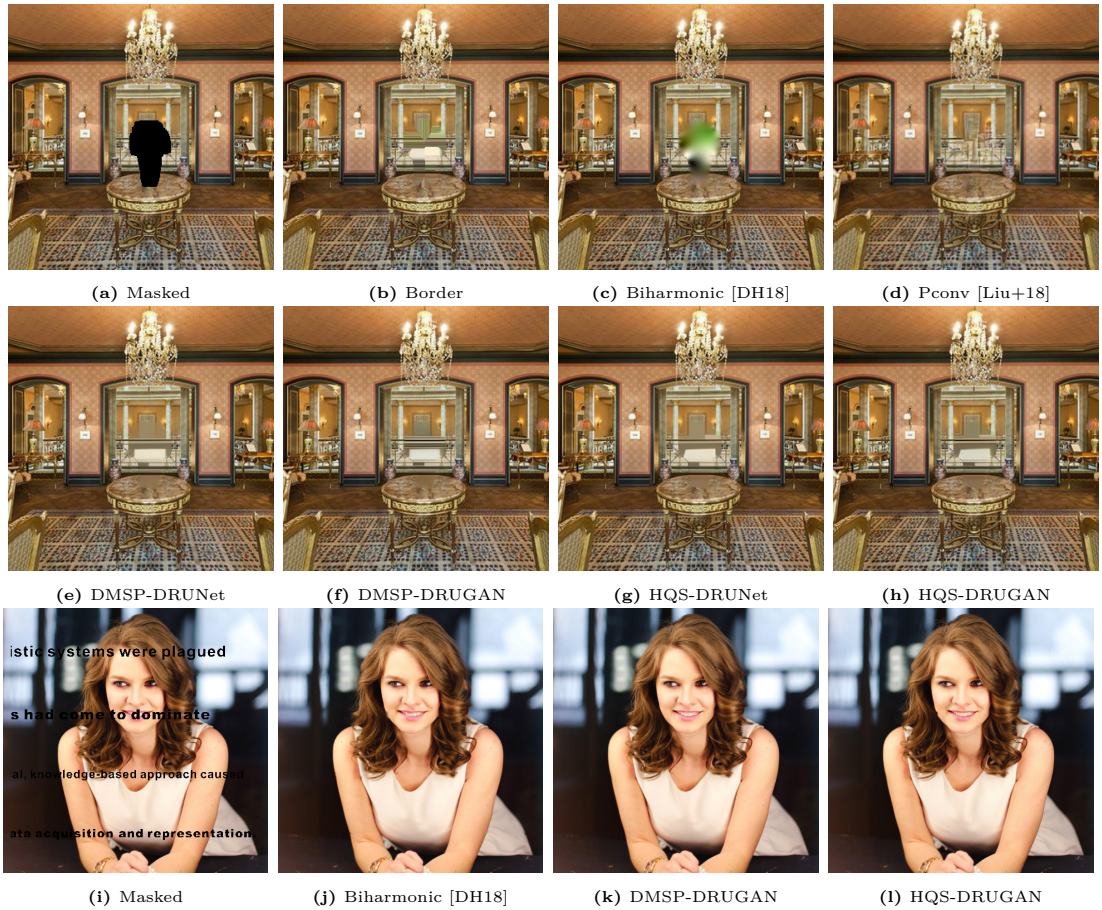
4.5 Inpainting

See Figure 4.8 for inpainting results using the DMSP and HQS algorithm. The "Border" method is a simple inpainting algorithm. The mean of all defined neighbors is used for each undefined pixel next to at least one defined pixel. This procedure is repeated until all pixels are defined.

For the DMSP and HQS algorithm, the image was initialized with the "Border" inpainted image. This initialization makes it easy to reconstruct straight lines because they are already present. If we look closely at the top two rows, we can see that the DMSP and HQS algorithm failed to reconstruct details like the texture on the table or the railing. The Pconv method, on the other hand, did a good job reconstructing the texture of the table.

For the uncomplicated example in the bottom row, DMSP and HQS did a better job. There is only one obvious mistake by the HQS algorithm at the woman's cheek. Otherwise, the text is not visible anymore in the image, and the areas are filled reasonably well.

We can see that DMSP and HQS can not compete with state-of-the-art inpainting methods. There could be several reasons for this. Many variables are involved in the inpainting algorithms. The user has to choose the initialization and the weighting of the prior in each step. We might not have discovered the best settings. It is even more likely that the denoiser cannot reconstruct a texture in one part of an image without any information but just based on the valid parts of the image.



(a) Masked (b) Border (c) Biharmonic [DH18] (d) Pconv [Liu+18]

(e) DMSP-DRUNet (f) DMSP-DRUGAN (g) HQS-DRUNet (h) HQS-DRUGAN

(i) Masked (j) Biharmonic [DH18] (k) DMSP-DRUGAN (l) HQS-DRUGAN

Figure 4.8: Inpainting results on two images. The Biharmonic result was obtained using the scikit-image [Wal+14] implementation. The Pconv result was obtained using the NVIDIA web app at `https://www.nvidia.com/research/inpainting/index.html`.

Chapter 5

Conclusion

In this work, we have introduced a new Gaussian denoiser that was trained using the GAN framework. Our denoiser, called DRUGAN, is based upon DRUNet by Zhang et al. [Zha+21a] but trained with an additional perceptual loss and an adversarial loss. It is a capable model that can handle a range of different levels of Gaussian noise by using a noise level map input (like FFDNet [ZZZ18]). DRUGAN achieves better than state-of-the-art denoising performance. We used the LPIPS [Zha+18] metric to evaluate the visually perceived similarity between the ground truth and the denoised image. No other denoiser known to us can compete with DRUGAN on this metric. We provided visual examples showing that DRUGAN better reconstructs texture and sharp edges and does not give over-smooth results as other models do.

DRUGAN was used for two model-based methods for image reconstruction on four distinct reconstruction tasks. We reproduced the deep mean-shift priors (DMSP) [Big+17] and implemented non-blind deblurring, single image super-resolution, and inpainting. We implemented the same tasks for the half quadratic splitting (HQS) algorithm [Zha+17b; Zha+21a]. Additionally, we implemented the more complex task of multi-frame super-resolution for light fields and videos. Both methods were evaluated using our DRUGAN denoiser. For non-blind deblurring, we found that the HQS algorithm works very well using our denoiser, and we beat every other method on the LPIPS metric. Also, the DMSP algorithm benefits from the powerful denoiser and achieves the second-best results for most settings. Again, we showed a visual example that confirms that for non-blind deblurring, the DRUGAN denoiser has the same advantages as for denoising. Texture and edges are reconstructed better than with other denoisers. These results show that the DMSP algorithm profits from a more powerful denoiser even if it is not trained with the MSE loss as the theory would require.

For single image super-resolution, our results are two-fold. When upscaling images that were downscaled using the classical bicubic resizing, we can observe

that our methods cannot compete with state-of-the-art end-to-end frameworks. Additionally, the performance of the denoiser seems to play a minor role, and DRUGAN cannot improve the results compared to other denoisers. However, the primary advantage of model-based solutions is that they can be adapted easily to slightly different tasks. We demonstrated this by evaluating single image super-resolution using downscaling by a set of Gaussian blur kernels. Our methods tackled the new task reasonably well while the end-to-end methods, trained to handle bicubic downscaling, failed. Adapting the end-to-end frameworks would require significantly more work because they would need to be retrained for every slight change in the task (probably for every blur kernel). For single image super-resolution with blur downscaling, the DRUGAN denoiser helps achieve results with better perceptual quality than other denoisers.

Next, the more complex task of multi-frame super-resolution was implemented. We used a simple iterative scheme to optimize the data term of the HQS algorithm that relies on accurate registration between the individual images. The HQS algorithm was used to upscale the center images of videos, where the registration was done using an existing optical flow library. Our experiments show that our method cannot compete with the state-of-the-art for bicubic downscaling. Even methods that do not use multiple images outperform our model-based approach. On the other hand, we demonstrated the same advantage as in the single image super-resolution task. Our method can be adapted easily to work with other kinds of downscaling and outperforms end-to-end learned methods that were not trained to handle this case. Additionally, the center images of light fields were upscaled. For light fields, the registration was done using a state-of-the-art method for disparity estimation—LFattNet [Tsa+20]. The results do not differ much from the results for videos. For all multi-frame super-resolution results, we showed that the DRUGAN denoiser helps to improve the perceptual quality.

The last reconstruction task explored was image inpainting. We showed two examples and can see that we cannot compete with state-of-the-art methods for larger missing areas. This is not surprising because inpainting is a special problem where realistic values must be generated without any valid data in a region. Most likely, our DRUGAN denoiser cannot do this, but it would need a more thorough exploration to be sure of the reason. For smaller areas, however, the prior using DRUGAN does a pretty good job reconstructing edges and inpainting the region realistically.

5.1 Future Work

Occlusions between frames could be considered to improve the method for multi-frame super-resolution. Wanner and Goldlücke [WG12] did this for light field

super-resolution, but it could also be adapted for video super-resolution. This would require the registration algorithm to output the occlusions in the individual frames. Not taking occlusion into account causes the method to use wrong information that hinders the algorithm from reaching full performance.

Currently, the method for deblurring assumes that the convolution was performed using circular boundary conditions. This is easy to implement for both algorithms but not very realistic in a practical use case. There are approaches to prevent boundary artifacts for deconvolution [LJ08; KRS17]. Future work could implement such an approach to apply the method to a wide range of blurred images.

The method is easy to adapt to more image reconstruction tasks. We have already applied the method to non-blind deblurring and super-resolution. The degradation model (blur kernel and noise level) must be known for both tasks. Bigdeli et al. [Big+17] already used the deep mean-shift prior for noise- and kernel-blind deblurring. This approach could be extended to upscale the image. Blind single image super-resolution would be valuable for reconstructing noisy low-resolution images from cameras.

Demosaicing is another classical and practically relevant image reconstruction task that can be handled with our approach. The DURGAN denoiser could be a valuable prior for sharp edges and a realistic texture.

Other than DMSP and HQS, the ADMM algorithm was used with a denoiser to solve inverse reconstruction tasks [BRE16]. We could try our DRUGAN denoiser with this algorithm and explore if it can achieve even better results than DMSP or HQS.

Bibliography

[AB14] Guillaume Alain and Yoshua Bengio. "What regularized auto-encoders learn from the data-generating distribution". In: *Journal of machine learning research* 15.110 (2014), pp. 3743–3773.

[ACB17] Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein generative adversarial networks". In: *Proceedings of the 34th international conference on machine learning*. Vol. 70. PMLR, 2017, pp. 214–223.

[Arb+11] Pablo Arbeláez et al. "Contour detection and hierarchical image segmentation". In: *IEEE transactions on pattern analysis and machine intelligence* 33.5 (May 2011), pp. 898–916. DOI: 10.1109/tpami.2010.161.

[AT17] Eirikur Agustsson and Radu Timofte. "NTIRE 2017 challenge on single image super-resolution: dataset and study". In: *2017 IEEE conference on computer vision and pattern recognition workshops (CVPRW)*. IEEE, July 2017. DOI: 10.1109/cvprw.2017.150.

[Ber+00] Marcelo Bertalmio et al. "Image inpainting". In: *Proceedings of the 27th annual conference on computer graphics and interactive techniques - SIGGRAPH '00*. ACM Press, 2000. DOI: 10.1145/344779.344972.

[Ber+03] M. Bertalmio et al. "Simultaneous structure and texture image inpainting". In: *IEEE transactions on image processing* 12.8 (Aug. 2003), pp. 882–889. DOI: 10.1109/tip.2003.815261.

[Big+17] Siavash Arjomand Bigdeli et al. "Deep mean-shift priors for image restoration". In: *Advances in neural information processing systems*. Vol. 30. Curran Associates, Inc., 2017.

[Boy10] Stephen Boyd. "Distributed optimization and statistical learning via the alternating direction method of multipliers". In: *Foundations and trends® in machine learning* 3.1 (2010), pp. 1–122. DOI: 10.1561/2200000016.

[BRE16] Alon Brifman, Yaniv Romano, and Michael Elad. "Turning a denoiser into a super-resolver using plug and play priors". In: *2016 IEEE international conference on image processing (ICIP)*. IEEE, Sept. 2016. DOI: 10.1109/icip.2016.7532589.

[Bro+04] Thomas Brox et al. "High accuracy optical flow estimation based on a theory for warping". In: *Lecture notes in computer science*. Springer Berlin Heidelberg, 2004, pp. 25–36. DOI: 10.1007/978-3-540-24673-2_3.

[Bro+19] Tim Brooks et al. "Unprocessing images for learned raw denoising". In: *2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*. IEEE, June 2019. DOI: 10.1109/cvpr.2019.01129.

[BWS05] Andrés Bruhn, Joachim Weickert, and Christoph Schnörr. "Lucas/kanade meets horn/schunck: combining local and global optic flow methods". In: *International journal of computer vision* 61.3 (Feb. 2005), pp. 1–21. DOI: 10.1023/b:visi.0000045324.43199.43.

[Cha+21] Kelvin C.K. Chan et al. "BasicVSR: the search for essential components in video super-resolution and beyond". In: *2021 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*. IEEE, June 2021. DOI: 10.1109/cvpr46437.2021.00491.

[Cha04] Antonin Chambolle. "An algorithm for total variation minimization and applications". In: *Journal of mathematical imaging and vision* 20 (Jan. 2004), pp. 89–97. DOI: 10.1023/b:jmiv.0000011325.36760.1e.

[Che+18] Jingwen Chen et al. "Image blind denoising with generative adversarial network based noise modeling". In: *2018 IEEE/CVF conference on computer vision and pattern recognition*. IEEE, June 2018. DOI: 10.1109/cvpr.2018.00333.

[CP10] Antonin Chambolle and Thomas Pock. "A first-order primal-dual algorithm for convex problems with applications to imaging". In: *Journal of mathematical imaging and vision* 40.1 (Dec. 2010), pp. 120–145. DOI: 10.1007/s10851-010-0251-1.

[CWE17] Stanley H. Chan, Xiran Wang, and Omar A. Elgendy. "Plug-and-play ADMM for image restoration: fixed-point convergence and applications". In: *IEEE transactions on computational imaging* 3.1 (Mar. 2017), pp. 84–98. DOI: 10.1109/tci.2016.2629286.

[Dab+07] Kostadin Dabov et al. “Image denoising by sparse 3-d transform-domain collaborative filtering”. In: *IEEE transactions on image processing* 16.8 (Aug. 2007), pp. 2080–2095. DOI: 10.1109/tip.2007.901238.

[Dai+17] Jifeng Dai et al. “Deformable convolutional networks”. In: *2017 IEEE international conference on computer vision (ICCV)*. IEEE, Oct. 2017. DOI: 10.1109/iccv.2017.89.

[DH18] S. B. Damelin and N. S. Hoang. “On surface completion and image inpainting by biharmonic functions: numerical aspects”. In: *International journal of mathematics and mathematical sciences* 2018 (2018), pp. 1–8. DOI: 10.1155/2018/3950312.

[DKE10] Aram Danielyan, Vladimir Katkovnik, and Karen Egiazarian. “Image deblurring by augmented lagrangian with BM3D frame prior”. In: *Workshop on information theoretic methods in science and engineering*. Vol. 1. 2010.

[DKE12] Aram Danielyan, Vladimir Katkovnik, and Karen Egiazarian. “BM3D frames and variational image deblurring”. In: *IEEE transactions on image processing* 21.4 (Apr. 2012), pp. 1715–1728. DOI: 10.1109/tip.2011.2176954.

[Don+16] Chao Dong et al. “Image super-resolution using deep convolutional networks”. In: *IEEE transactions on pattern analysis and machine intelligence* 38.2 (Feb. 2016), pp. 295–307. DOI: 10.1109/tpami.2015.2439281.

[EK15] Karen Egiazarian and Vladimir Katkovnik. “Single image super-resolution via BM3D sparse coding”. In: *2015 23rd european signal processing conference (EUSIPCO)*. IEEE, Aug. 2015. DOI: 10.1109/eusipco.2015.7362905.

[Fan+19] Linwei Fan et al. “Brief review of image denoising techniques”. In: *Visual computing for industry, biomedicine, and art* 2.1 (July 2019). DOI: 10.1186/s42492-019-0016-7.

[Fra13] Richard Franzen. *Kodak lossless true color image suite*. 2013. URL: http://r0k.us/graphics/kodak/ (visited on 10/16/2021).

[Gol17] Bastian Goldlücke. *Lecture notes in Image Analysis and Computer Vision II*. Apr. 2017.

[Goo+14] Ian Goodfellow et al. “Generative adversarial nets”. In: *Advances in neural information processing systems*. Vol. 27. Curran Associates, Inc., 2014.

[Gul+17] Ishaan Gulrajani et al. “Improved training of wasserstein gans”. In: *Advances in neural information processing systems*. Vol. 30. Curran Associates, Inc., 2017.

[GY95] Donald Geman and Chengda Yang. “Nonlinear image recovery with half-quadratic regularization”. In: *IEEE transactions on image processing* 4.7 (July 1995), pp. 932–946. DOI: 10.1109/83.392335.

[He+16] Kaiming He et al. “Deep residual learning for image recognition”. In: *2016 IEEE conference on computer vision and pattern recognition (CVPR)*. IEEE, June 2016. DOI: 10.1109/cvpr.2016.90.

[Hei+14] Felix Heide et al. “FlexISP: a flexible camera image processing framework”. In: *ACM transactions on graphics* 33.6 (Nov. 2014), pp. 1–13. DOI: 10.1145/2661229.2661260.

[Hon+17] Katrin Honauer et al. “A dataset and evaluation methodology for depth estimation on 4d light fields”. In: *Computer vision – ACCV 2016*. Springer International Publishing, 2017, pp. 19–34. DOI: 10.1007/978-3-319-54187-7_2.

[How21] Jeremy Howard. *Imagenette*. 2021. URL: https://github.com/fastai/imagenette/ (visited on 05/27/2021).

[HS81] Berthold K.P. Horn and Brian G. Schunck. “Determining optical flow”. In: *Artificial intelligence* 17.1-3 (Aug. 1981), pp. 185–203. DOI: 10.1016/0004-3702(81)90024-2.

[IP91] Michal Irani and Shmuel Peleg. “Improving resolution by image registration”. In: *CVGIP: graphical models and image processing* 53.3 (May 1991), pp. 231–239. DOI: 10.1016/1049-9652(91)90045-l.

[IS15] Sergey Ioffe and Christian Szegedy. “Batch normalization: accelerating deep network training by reducing internal covariate shift”. In: *Proceedings of the 32nd international conference on machine learning*. Vol. 37. PMLR, 2015, pp. 448–456.

[ISI17] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. “Globally and locally consistent image completion”. In: *ACM transactions on graphics* 36.4 (July 2017), pp. 1–14. DOI: 10.1145/3072959.3073659.

[JAF16] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. “Perceptual losses for real-time style transfer and super-resolution”. In: *Computer vision – ECCV 2016*. Springer International Publishing, 2016, pp. 694–711. DOI: 10.1007/978-3-319-46475-6_43.

[Jin+20] Jing Jin et al. “Light field spatial super-resolution via deep combinatorial geometry embedding and structural consistency regularization”. In: *2020 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*. IEEE, June 2020. DOI: 10.1109/cvpr42600.2020.00233.

[Jo+18] Younghyun Jo et al. “Deep video super-resolution network using dynamic upsampling filters without explicit motion compensation”. In: *2018 IEEE/CVF conference on computer vision and pattern recognition*. IEEE, June 2018. DOI: 10.1109/cvpr.2018.00340.

[Joh+17] Ole Johannsen et al. “A taxonomy and evaluation of dense light field depth estimation algorithms”. In: *2017 IEEE conference on computer vision and pattern recognition workshops (CVPRW)*. IEEE, July 2017. DOI: 10.1109/cvprw.2017.226.

[Jol19] Alexia Jolicoeur-Martineau. “The relativistic discriminator: a key element missing from standard GAN”. In: *International conference on learning representations*. 2019.

[JRF17] Meiguang Jin, Stefan Roth, and Paolo Favaro. “Noise-blind image deblurring”. In: *2017 IEEE conference on computer vision and pattern recognition (CVPR)*. IEEE, July 2017. DOI: 10.1109/cvpr.2017.408.

[Kap+16] Armin Kappeler et al. “Video super-resolution with convolutional neural networks”. In: *IEEE transactions on computational imaging* 2.2 (June 2016), pp. 109–122. DOI: 10.1109/tci.2016.2532323.

[Kar+18] Tero Karras et al. “Progressive growing of GANs for improved quality, stability, and variation”. In: *International conference on learning representations*. 2018.

[KB17] Diederik P. Kingma and Jimmy Ba. *Adam: a method for stochastic optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[KLA19] Tero Karras, Samuli Laine, and Timo Aila. “A style-based generator architecture for generative adversarial networks”. In: *2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*. IEEE, June 2019. DOI: 10.1109/cvpr.2019.00453.

[KLL16] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. “Accurate image super-resolution using very deep convolutional networks”. In: *2016 IEEE conference on computer vision and pattern recognition (CVPR)*. IEEE, June 2016. DOI: 10.1109/cvpr.2016.182.

[KRS17] Jakob Kruse, Carsten Rother, and Uwe Schmidt. "Learning to push the limits of efficient FFT-based image deconvolution". In: *2017 IEEE international conference on computer vision (ICCV)*. IEEE, Oct. 2017. DOI: 10.1109/iccv.2017.491.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. Vol. 25. Curran Associates, Inc., 2012.

[Kup+18] Orest Kupyn et al. "DeblurGAN: blind motion deblurring using conditional adversarial networks". In: *2018 IEEE/CVF conference on computer vision and pattern recognition*. IEEE, June 2018. DOI: 10.1109/cvpr.2018.00854.

[Kup+19] Orest Kupyn et al. "DeblurGAN-v2: deblurring (orders-of-magnitude) faster and better". In: *2019 IEEE/CVF international conference on computer vision (ICCV)*. IEEE, Oct. 2019. DOI: 10.1109/iccv.2019.00897.

[Led+17] Christian Ledig et al. "Photo-realistic single image super-resolution using a generative adversarial network". In: *2017 IEEE conference on computer vision and pattern recognition (CVPR)*. IEEE, July 2017. DOI: 10.1109/cvpr.2017.19.

[Lev+09] Anat Levin et al. "Understanding and evaluating blind deconvolution algorithms". In: *2009 IEEE conference on computer vision and pattern recognition*. IEEE, June 2009. DOI: 10.1109/cvpr.2009.5206815.

[Lim+17] Bee Lim et al. "Enhanced deep residual networks for single image super-resolution". In: *2017 IEEE conference on computer vision and pattern recognition workshops (CVPRW)*. IEEE, July 2017. DOI: 10.1109/cvprw.2017.151.

[Liu+18] Guilin Liu et al. "Image inpainting for irregular holes using partial convolutions". In: *Computer vision – ECCV 2018*. Springer International Publishing, 2018, pp. 89–105. DOI: 10.1007/978-3-030-01252-6_6.

[Liu09] Ce Liu. "Beyond pixels: exploring new representations and applications for motion analysis". PhD thesis. Massachusetts Institute of Technology, May 2009.

[LJ08] Renting Liu and Jiaya Jia. "Reducing boundary artifacts in image deconvolution". In: *2008 15th IEEE international conference on image processing*. IEEE, 2008. DOI: 10.1109/icip.2008.4711802.

[LK81] Bruce D. Lucas and Takeo Kanade. "An iterative image registration technique with an application to stereo vision". In: *Proceedings of DARPA imaging understanding workshop*. 1981, pp. 121–130.

[LS11] Ce Liu and Deqing Sun. "A bayesian approach to adaptive video super resolution". In: *CVPR 2011*. IEEE, June 2011. DOI: 10.1109/cvpr.2011.5995614.

[LVU18] Victor Lempitsky, Andrea Vedaldi, and Dmitry Ulyanov. "Deep image prior". In: *2018 IEEE/CVF conference on computer vision and pattern recognition*. IEEE, June 2018. DOI: 10.1109/cvpr.2018.00984.

[Ma+17] Kede Ma et al. "Waterloo exploration database: new challenges for image quality assessment models". In: *IEEE transactions on image processing* 26.2 (Feb. 2017), pp. 1004–1016. DOI: 10.1109/tip.2016.2631888.

[MAF20] Ymir Makinen, Lucio Azzari, and Alessandro Foi. "Collaborative filtering of correlated noise: exact transform-domain variance for improved shrinkage and patch matching". In: *IEEE transactions on image processing* 29 (2020), pp. 8339–8354. DOI: 10.1109/tip.2020.3014721.

[Mal89] S.G. Mallat. "A theory for multiresolution signal decomposition: the wavelet representation". In: *IEEE transactions on pattern analysis and machine intelligence* 11.7 (July 1989), pp. 674–693. DOI: 10.1109/34.192463.

[Mar+] D. Martin et al. "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics". In: *Proceedings eighth IEEE international conference on computer vision. ICCV 2001*. IEEE Comput. Soc. DOI: 10.1109/iccv.2001.937655.

[Mil21] Peyman Milanfar. *MDSP super-resolution and demosaicing datasets*. 2021. URL: https://users.soe.ucsc.edu/~milanfar/software/sr-datasets.html (visited on 12/09/2021).

[Mit+09] Dennis Mitzel et al. "Video super resolution using duality based TV-l 1 optical flow". In: *Lecture notes in computer science*. Springer Berlin Heidelberg, 2009, pp. 432–441. DOI: 10.1007/978-3-642-03798-6_44.

[MSY16] Xiao-Jiao Mao, Chunhua Shen, and Yu-Bin Yang. *Image restoration using convolutional auto-encoders with symmetric skip connections*. 2016. arXiv: 1606.08921 [cs.CV].

[NH10] Vinod Nair and Geoffrey E. Hinton. "Rectified linear units improve restricted boltzmann machines". In: *Proceedings of the 27th international conference on international conference on machine learning*. Omnipress, 2010, pp. 807–814.

[NKL17] Seungjun Nah, Tae Hyun Kim, and Kyoung Mu Lee. "Deep multi-scale convolutional neural network for dynamic scene deblurring". In: *2017 IEEE conference on computer vision and pattern recognition (CVPR)*. IEEE, July 2017. DOI: 10.1109/cvpr.2017.35.

[NM14] Kamal Nasrollahi and Thomas B. Moeslund. "Super-resolution: a comprehensive survey". In: *Machine vision and applications* 25.6 (June 2014), pp. 1423–1468. DOI: 10.1007/s00138-014-0623-4.

[Oli+01] Manuel M. Oliveira et al. "Fast digital image inpainting". In: *Proceedings of the international conference on visualization, imaging and image processing (VIIP 2001)*. 2001, pp. 106–107.

[Pat+16] Deepak Pathak et al. "Context encoders: feature learning by inpainting". In: *2016 IEEE conference on computer vision and pattern recognition (CVPR)*. IEEE, June 2016. DOI: 10.1109/cvpr.2016.278.

[Pat21] Deepak Pathak. *Python dense optical flow*. 2021. URL: https://github.com/pathak22/pyflow (visited on 12/13/2021).

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: convolutional networks for biomedical image segmentation". In: *Lecture notes in computer science*. Springer International Publishing, 2015, pp. 234–241. DOI: 10.1007/978-3-319-24574-4_28.

[RMC16] Alec Radford, Luke Metz, and Soumith Chintala. *Unsupervised representation learning with deep convolutional generative adversarial networks*. 2016. arXiv: 1511.06434 [cs.LG].

[Sch+15] Kevin Schelten et al. "Interleaved regression tree field cascades for blind image deconvolution". In: *2015 IEEE winter conference on applications of computer vision*. IEEE, Jan. 2015. DOI: 10.1109/wacv.2015.72.

[SM16] Tamar Rott Shaham and Tomer Michaeli. "Visualizing image priors". In: *Computer vision – ECCV 2016*. Springer International Publishing, 2016, pp. 136–153. DOI: 10.1007/978-3-319-46466-4_9.

[SZ15] Karen Simonyan and Andrew Zisserman. *Very deep convolutional networks for large-scale image recognition*. 2015. arXiv: 1409.1556 [cs.CV].

[Tao+17] Xin Tao et al. “Detail-revealing deep video super-resolution”. In: *2017 IEEE international conference on computer vision (ICCV)*. IEEE, Oct. 2017. DOI: 10.1109/iccv.2017.479.

[Ten21] TensorFlow Developers. *Tensorflow*. 2021. DOI: 10.5281/ZENODO.5189249.

[Tsa+20] Yu-Ju Tsai et al. “Attention-based view selection networks for light-field disparity estimation”. In: *Proceedings of the AAAI conference on artificial intelligence* 34.07 (Apr. 2020), pp. 12095–12103. DOI: 10.1609/aaai.v34i07.6888.

[VBW13] Singanallur V. Venkatakrishnan, Charles A. Bouman, and Brendt Wohlberg. “Plug-and-play priors for model based reconstruction”. In: *2013 IEEE global conference on signal and information processing*. IEEE, Dec. 2013. DOI: 10.1109/globalsip.2013.6737048.

[Ven+18] G. M. Venkatesh et al. “A deep residual architecture for skin lesion segmentation”. In: *Lecture notes in computer science*. Springer International Publishing, 2018, pp. 277–284. DOI: 10.1007/978-3-030-01201-4_30.

[Wal+14] Stéfan van der Walt et al. “Scikit-image: image processing in python”. In: *PeerJ* 2 (June 2014), e453. DOI: 10.7717/peerj.453.

[Wan+04] Z. Wang et al. “Image quality assessment: from error visibility to structural similarity”. In: *IEEE transactions on image processing* 13.4 (Apr. 2004), pp. 600–612. DOI: 10.1109/tip.2003.819861.

[Wan+18] Yunlong Wang et al. “LFNet: a novel bidirectional recurrent convolutional neural network for light-field image super-resolution”. In: *IEEE transactions on image processing* 27.9 (Sept. 2018), pp. 4274–4286. DOI: 10.1109/tip.2018.2834819.

[Wan+19a] Xintao Wang et al. “EDVR: video restoration with enhanced deformable convolutional networks”. In: *2019 IEEE/CVF conference on computer vision and pattern recognition workshops (CVPRW)*. IEEE, June 2019. DOI: 10.1109/cvprw.2019.00247.

[Wan+19b] Xintao Wang et al. “ESRGAN: enhanced super-resolution generative adversarial networks”. In: *Lecture notes in computer science*. Springer International Publishing, 2019, pp. 63–79. DOI: 10.1007/978-3-030-11021-5_5.

[Wan+20] Ziyuan Wang et al. “An image denoising method based on deep residual GAN”. In: *Journal of physics: conference series* 1550.3 (May 2020), p. 032127. DOI: 10.1088/1742-6596/1550/3/032127.

[Wan+21] Xintao Wang et al. *BasicSR: open source image and video restoration toolbox*. 2021. URL: https://github.com/xinntao/BasicSR (visited on 12/08/2021).

[WG12] Sven Wanner and Bastian Goldluecke. "Spatial and angular variational super-resolution of 4d light fields". In: *Computer vision – ECCV 2012*. Springer Berlin Heidelberg, 2012, pp. 608–621. DOI: 10.1007/978-3-642-33715-4_44.

[WG14] Sven Wanner and Bastian Goldluecke. "Variational light field analysis for disparity estimation and super-resolution". In: *IEEE transactions on pattern analysis and machine intelligence* 36.3 (Mar. 2014), pp. 606–619. DOI: 10.1109/tpami.2013.147.

[Wik21] Wikipedia contributors. *Peak signal-to-noise ratio — Wikipedia, the free encyclopedia*. 2021. URL: https://en.wikipedia.org/wiki/Peak_signal-to-noise_ratio (visited on 04/30/2021).

[Wu11] Xiaolin Wu. "Color demosaicking by local directional interpolation and nonlocal adaptive thresholding". In: *Journal of electronic imaging* 20.2 (Apr. 2011), p. 023016. DOI: 10.1117/1.3600632.

[Xu+14] Li Xu et al. "Deep convolutional neural network for image deconvolution". In: *Advances in neural information processing systems*. Vol. 27. Curran Associates, Inc., 2014.

[Yan+17] Chao Yang et al. "High-resolution image inpainting using multi-scale neural patch synthesis". In: *2017 IEEE conference on computer vision and pattern recognition (CVPR)*. IEEE, July 2017. DOI: 10.1109/cvpr.2017.434.

[YK16] Fisher Yu and Vladlen Koltun. *Multi-scale context aggregation by dilated convolutions*. 2016. arXiv: 1511.07122 [cs.CV].

[Yoo+15] Youngjin Yoon et al. "Learning a deep convolutional network for light-field image super-resolution". In: *2015 IEEE international conference on computer vision workshop (ICCVW)*. IEEE, Dec. 2015. DOI: 10.1109/iccvw.2015.17.

[Yu+18] Jiahui Yu et al. "Generative image inpainting with contextual attention". In: *2018 IEEE/CVF conference on computer vision and pattern recognition*. IEEE, June 2018. DOI: 10.1109/cvpr.2018.00577.

[YW17] Qiaojing Yan and Wei Wang. *DCGANs for image super-resolution , denoising and debluring*. 2017. URL: http://stanford.edu/class/ee367/Winter2017/yan_wang_ee367_win17_report.pdf.

[ZGT20] Kai Zhang, Luc Van Gool, and Radu Timofte. "Deep unfolding network for image super-resolution". In: *2020 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*. IEEE, June 2020. DOI: 10.1109/cvpr42600.2020.00328.

[Zha+11] Lin Zhang et al. "FSIM: a feature similarity index for image quality assessment". In: *IEEE transactions on image processing* 20.8 (Aug. 2011), pp. 2378–2386. DOI: 10.1109/tip.2011.2109730.

[Zha+17a] Kai Zhang et al. "Beyond a gaussian denoiser: residual learning of deep CNN for image denoising". In: *IEEE transactions on image processing* 26.7 (July 2017), pp. 3142–3155. DOI: 10.1109/tip.2017.2662206.

[Zha+17b] Kai Zhang et al. "Learning deep CNN denoiser prior for image restoration". In: *2017 IEEE conference on computer vision and pattern recognition (CVPR)*. IEEE, July 2017. DOI: 10.1109/cvpr.2017.300.

[Zha+18] Richard Zhang et al. "The unreasonable effectiveness of deep features as a perceptual metric". In: *2018 IEEE/CVF conference on computer vision and pattern recognition*. IEEE, June 2018. DOI: 10.1109/cvpr.2018.00068.

[Zha+21a] Kai Zhang et al. "Plug-and-play image restoration with deep denoiser prior". In: *IEEE transactions on pattern analysis and machine intelligence* (2021), pp. 1–1. DOI: 10.1109/tpami.2021.3088914.

[Zha+21b] Yulun Zhang et al. "Residual dense network for image restoration". In: *IEEE transactions on pattern analysis and machine intelligence* 43.7 (July 2021), pp. 2480–2495. DOI: 10.1109/tpami.2020.2968521.

[ZLS19] Shuo Zhang, Youfang Lin, and Hao Sheng. "Residual networks for light field image super-resolution". In: *2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*. IEEE, June 2019. DOI: 10.1109/cvpr.2019.01130.

[ZLW18] Zhengxin Zhang, Qingjie Liu, and Yunhong Wang. "Road extraction by deep residual u-net". In: *IEEE geoscience and remote sensing letters* 15.5 (May 2018), pp. 749–753. DOI: 10.1109/lgrs.2018.2802944.

[ZW11] Daniel Zoran and Yair Weiss. "From learning models of natural image patches to whole image restoration". In: *2011 international conference on computer vision*. IEEE, Nov. 2011. DOI: 10.1109/iccv.2011.6126278.

[ZZZ18] Kai Zhang, Wangmeng Zuo, and Lei Zhang. "FFDNet: toward a fast and flexible solution for CNN-based image denoising". In: *IEEE transactions on image processing* 27.9 (Sept. 2018), pp. 4608–4622. DOI: 10.1109/tip.2018.2839891.

Appendix A

Additional Results

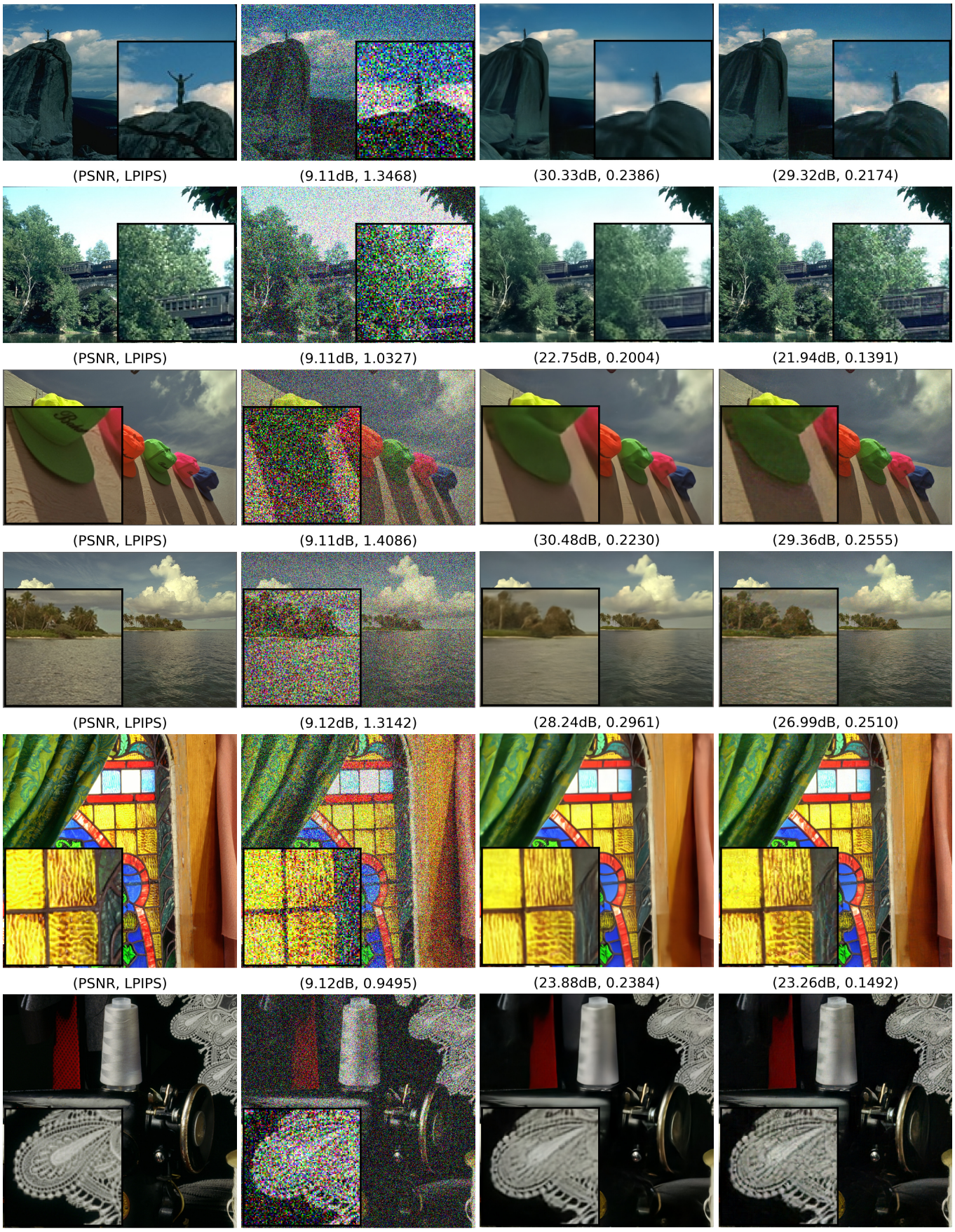
This Appendix shows an almost complete listing of the results of all conducted experiments. All results and the evaluation code are available at

`https://github.com/HedgehogCode/masters-thesis-evaluation`.

In the following tables, the names of the metrics will be abbreviated by the starting letter (PSNR: P, SSIM: S, FSIM: F, LPIPS: L). All metrics were computed over all channels of the RGB color image.

			CBM3D [MAF20]	CDnCNN-B [Zha+17a]	DnCNN (DMSP) [Big+17]	DRUNet (DPIR) [Zha+21a]	$\text{DCNN}_{0.05}$	$\text{DRCNN}_{0.05}$	$\text{DUNet}_{0.05}$	$\text{DUNet}^{+}_{0.05}$	$\text{DRUNet}^{+}_{0.05}$	$\text{DUNet}^{+}_{0\ldots0.2}$	$\text{DRUNet}^{+}_{0\ldots0.2}$	$\text{DRUGAN}^{+}_{0\ldots0.2}$ ($\lambda = 0$)	$\text{DRUGAN}^{+}_{0\ldots0.2}$
CBSD68 [Mar+]	σ_n=0.01	P	44.60	43.90	—	44.88	—	—	—	—	—	44.08	44.86	44.81	44.16
		S	0.9912	0.9902	—	0.9918	—	—	—	—	—	0.9908	0.9918	0.9916	0.9909
		F	0.9987	0.9987	—	0.9988	—	—	—	—	—	0.9986	0.9988	0.9990	0.9989
		L	0.0061	0.0043	—	0.0048	—	—	—	—	—	0.0052	0.0050	0.0029	0.0029
	σ_n=0.05	P	34.45	34.77	33.57	35.17	34.64	34.88	34.69	34.84	33.62	34.82	35.11	34.96	34.54
		S	0.9363	0.9402	0.9055	0.9448	0.9396	0.9414	0.9406	0.9414	0.9211	0.9421	0.9444	0.9424	0.9381
		F	0.9858	0.9870	0.9891	0.9884	0.9869	0.9875	0.9872	0.9874	0.9895	0.9880	0.9886	0.9911	0.9916
		L	0.0639	0.0538	0.0477	0.0472	0.0512	0.0498	0.0508	0.0503	0.0528	0.0492	0.0466	0.0280	0.0260
	σ_n=0.1	P	30.67	31.12	—	31.56	—	—	—	—	—	31.22	31.50	31.31	30.80
		S	0.8674	0.8807	—	0.8907	—	—	—	—	—	0.8852	0.8897	0.8860	0.8764
		F	0.9617	0.9686	—	0.9715	—	—	—	—	—	0.9704	0.9723	0.9784	0.9801
		L	0.1543	0.1132	—	0.0990	—	—	—	—	—	0.1033	0.0972	0.0630	0.0575
	σ_n=0.2	P	27.36	27.82	—	28.38	—	—	—	—	—	28.01	28.31	28.11	27.48
		S	0.7636	0.7854	—	0.8073	—	—	—	—	—	0.7962	0.8054	0.7999	0.7799
		F	0.9140	0.9318	—	0.9380	—	—	—	—	—	0.9360	0.9397	0.9522	0.9557
		L	0.2897	0.2170	—	0.1841	—	—	—	—	—	0.1946	0.1807	0.1265	0.1196
Kodak24 [Fra13]	σ_n=0.01	P	44.26	42.99	—	44.70	—	—	—	—	—	44.16	44.64	44.57	43.98
		S	0.9860	0.9847	—	0.9875	—	—	—	—	—	0.9863	0.9874	0.9872	0.9860
		F	0.9981	0.9979	—	0.9983	—	—	—	—	—	0.9984	0.9983	0.9986	0.9986
		L	0.0110	0.0077	—	0.0066	—	—	—	—	—	0.0068	0.0064	0.0037	0.0040
	σ_n=0.05	P	35.28	35.39	34.11	36.12	35.60	35.73	35.63	35.69	35.96	35.74	36.05	35.87	35.38
		S	0.9292	0.9318	0.8893	0.9389	0.9326	0.9345	0.9337	0.9341	0.9372	0.9354	0.9384	0.9357	0.9301
		F	0.9770	0.9791	0.9854	0.9817	0.9800	0.9805	0.9804	0.9803	0.9810	0.9809	0.9819	0.9880	0.9887
		L	0.0889	0.0759	0.0512	0.0607	0.0695	0.0683	0.0683	0.0683	0.0651	0.0665	0.0606	0.0333	0.0321
	σ_n=0.1	P	31.77	32.04	—	32.81	—	—	—	—	—	32.39	32.73	32.52	31.95
		S	0.8669	0.8758	—	0.8904	—	—	—	—	—	0.8835	0.8893	0.8854	0.8738
		F	0.9469	0.9545	—	0.9591	—	—	—	—	—	0.9587	0.9602	0.9718	0.9743
		L	0.1737	0.1367	—	0.1145	—	—	—	—	—	0.1192	0.1123	0.0756	0.0655
	σ_n=0.2	P	28.56	28.85	—	29.78	—	—	—	—	—	29.26	29.68	29.46	28.75
		S	0.7758	0.7886	—	0.8183	—	—	—	—	—	0.8043	0.8160	0.8099	0.7842
		F	0.8975	0.9150	—	0.9224	—	—	—	—	—	0.9233	0.9260	0.9428	0.9476
		L	0.2932	0.2421	—	0.1925	—	—	—	—	—	0.2042	0.1891	0.1376	0.1261
McMaster [Wu11]	σ_n=0.01	P	42.77	37.90	—	43.66	—	—	—	—	—	42.39	43.38	43.29	42.67
		S	0.9793	0.9570	—	0.9840	—	—	—	—	—	0.9770	0.9830	0.9825	0.9799
		F	0.9978	0.9930	—	0.9985	—	—	—	—	—	0.9982	0.9985	0.9986	0.9986
		L	0.0061	0.0154	—	0.0041	—	—	—	—	—	0.0045	0.0041	0.0025	0.0027
	σ_n=0.05	P	34.76	34.00	34.06	36.13	35.24	35.58	35.26	35.41	34.56	35.36	35.95	35.77	35.31
		S	0.9193	0.9116	0.8910	0.9405	0.9299	0.9324	0.9292	0.9298	0.9150	0.9283	0.9385	0.9353	0.9287
		F	0.9832	0.9815	0.9894	0.9859	0.9853	0.9854	0.9849	0.9854	0.9882	0.9851	0.9860	0.9899	0.9895
		L	0.0574	0.0621	0.0303	0.0441	0.0487	0.0481	0.0488	0.0507	0.0461	0.0490	0.0433	0.0214	0.0218
	σ_n=0.1	P	31.55	31.43	—	33.04	—	—	—	—	—	32.34	32.87	32.65	32.18
		S	0.8637	0.8671	—	0.9021	—	—	—	—	—	0.8863	0.8990	0.8933	0.8835
		F	0.9655	0.9655	—	0.9707	—	—	—	—	—	0.9708	0.9717	0.9779	0.9786
		L	0.1120	0.1028	—	0.0826	—	—	—	—	—	0.0877	0.0815	0.0490	0.0438
	σ_n=0.2	P	28.40	28.52	—	29.98	—	—	—	—	—	29.22	29.79	29.55	29.06
		S	0.7794	0.7945	—	0.8446	—	—	—	—	—	0.8191	0.8386	0.8280	0.8102
		F	0.9323	0.9369	—	0.9461	—	—	—	—	—	0.9438	0.9472	0.9558	0.9568
		L	0.2001	0.1756	—	0.1427	—	—	—	—	—	0.1559	0.1423	0.0949	0.0874

Table A.1: Denoising results for three datasets with four different noise levels



(a) GT **(b)** Noisy **(c)** $\text{DRUNet}^{+}_{0...0.2}$ **(d)** $\text{DRUGAN}^{+}_{0...0.2}$

Figure A.1: Denoising results on two images of each, the CBSD68 [Mar+] dataset, the Kodak24 [Fra13] dataset, and the McMaster [Wu11] dataset. A noise standard deviation of $\sigma_n = 0.35$ was used.

			FDN [KRS17]	DMSP [Big+17]	DPIR [Zha+21a]	DMSP						HQS	
						$\text{DCNN}_{0.05}$	$\text{DRCNN}_{0.05}$	$\text{DUNet}_{0.05}$	$\text{DRUNet}^{+}_{0.05}$	$\text{DRUNet}^{+}_{0\ldots0.2}$	$\text{DRUGAN}^{+}_{0\ldots0.2}$	$\text{DRUNet}^{+}_{0\ldots0.2}$	$\text{DRUGAN}^{+}_{0\ldots0.2}$
BSDS500 [Arb+11]	σ_n=0.01	P	23.32	26.63	29.09	27.85	27.90	27.67	26.01	25.57	23.65	29.08	27.62
		S	0.7232	0.7706	0.8420	0.7814	0.7843	0.7806	0.7632	0.6651	0.6051	0.8444	0.8259
		F	0.9168	0.9651	0.9851	0.9777	0.9792	0.9786	0.9800	0.9756	0.9622	0.9856	0.9866
		L	0.3265	0.2239	0.1700	0.1821	0.1810	0.1895	0.2073	0.1589	0.1800	0.1559	0.0725
	σ_n=0.02	P	23.81	24.93	26.64	25.71	25.75	25.39	23.46	24.92	22.26	26.66	25.38
		S	0.6254	0.6700	0.7489	0.6849	0.6891	0.6812	0.6757	0.6236	0.5501	0.7529	0.7440
		F	0.8618	0.9055	0.9536	0.9229	0.9256	0.9235	0.9337	0.9637	0.9397	0.9605	0.9719
		L	0.4388	0.3920	0.3176	0.3525	0.3552	0.3658	0.3899	0.2804	0.2639	0.3007	0.1653
	σ_n=0.03	P	23.08	23.85	25.43	24.48	24.51	24.08	22.34	24.40	22.38	25.42	24.45
		S	0.5734	0.5986	0.6890	0.6147	0.6193	0.6078	0.6106	0.5944	0.5482	0.6927	0.6937
		F	0.8323	0.8549	0.9149	0.8731	0.8746	0.8722	0.8812	0.9463	0.9326	0.9290	0.9568
		L	0.5058	0.4908	0.4057	0.4528	0.4568	0.4700	0.4893	0.3688	0.3335	0.3946	0.2496
	σ_n=0.04	P	22.64	23.12	24.67	23.64	23.69	23.24	21.54	23.90	22.71	24.61	23.88
		S	0.5381	0.5476	0.6471	0.5627	0.5681	0.5542	0.5601	0.5674	0.5501	0.6494	0.6562
		F	0.8142	0.8254	0.8837	0.8418	0.8417	0.8413	0.8407	0.9240	0.9257	0.9026	0.9401
		L	0.5485	0.5540	0.4621	0.5209	0.5249	0.5410	0.5544	0.4321	0.3895	0.4584	0.3188
Set5 + 2nd kernel from [Lev+09]	σ_n=0.01	P	31.70	31.03	34.69	31.38	31.45	31.34	31.23	27.02	26.34	34.58	33.17
		S	0.8708	0.8272	0.9197	0.8090	0.8109	0.8087	0.8052	0.6303	0.6053	0.9196	0.9001
		F	0.9775	0.9857	0.9872	0.9866	0.9867	0.9862	0.9870	0.9620	0.9544	0.9879	0.9902
		L	0.1049	0.0574	0.0695	0.0628	0.0602	0.0629	0.0621	0.1671	0.1939	0.0604	0.0283
	σ_n=0.02	P	29.47	30.37	32.40	30.56	30.72	30.50	30.05	26.60	24.02	32.32	30.86
		S	0.8140	0.8277	0.8867	0.8090	0.8133	0.8099	0.8034	0.6034	0.5253	0.8868	0.8645
		F	0.9607	0.9762	0.9695	0.9786	0.9794	0.9784	0.9784	0.9496	0.9208	0.9719	0.9810
		L	0.1693	0.1122	0.1297	0.1017	0.0985	0.1047	0.1057	0.2025	0.2744	0.1187	0.0537
	σ_n=0.03	P	28.40	29.14	31.11	29.53	29.75	29.35	28.64	27.04	22.02	31.01	29.56
		S	0.7904	0.8111	0.8636	0.8008	0.8068	0.8001	0.7985	0.6249	0.5061	0.8630	0.8392
		F	0.9502	0.9627	0.9561	0.9671	0.9684	0.9661	0.9658	0.9498	0.9039	0.9586	0.9707
		L	0.2002	0.1671	0.1560	0.1481	0.1460	0.1577	0.1568	0.1954	0.2821	0.1477	0.0867
	σ_n=0.04	P	27.64	28.00	30.15	28.48	28.82	28.23	27.38	27.31	23.65	30.04	28.74
		S	0.7724	0.7874	0.8435	0.7859	0.7940	0.7835	0.7839	0.6503	0.5687	0.8421	0.8201
		F	0.9411	0.9495	0.9456	0.9552	0.9575	0.9534	0.9526	0.9525	0.9221	0.9471	0.9576
		L	0.2177	0.2137	0.1749	0.1888	0.1855	0.2016	0.2077	0.1894	0.2362	0.1713	0.1144
Set5 + 4th kernel from [Lev+09]	σ_n=0.01	P	26.98	28.56	34.48	31.21	31.28	31.16	31.06	26.84	26.13	34.33	32.84
		S	0.8352	0.8049	0.9168	0.8034	0.8054	0.8028	0.7994	0.6212	0.5956	0.9163	0.8937
		F	0.9597	0.9757	0.9857	0.9857	0.9859	0.9853	0.9861	0.9595	0.9508	0.9864	0.9891
		L	0.1188	0.0648	0.0711	0.0656	0.0630	0.0655	0.0653	0.1731	0.2004	0.0629	0.0311
	σ_n=0.02	P	25.77	28.42	32.17	30.31	30.47	30.27	29.78	26.45	23.70	32.06	30.49
		S	0.7830	0.8081	0.8821	0.8013	0.8056	0.8022	0.7956	0.5955	0.5188	0.8818	0.8551
		F	0.9433	0.9673	0.9665	0.9767	0.9774	0.9763	0.9758	0.9482	0.9199	0.9687	0.9797
		L	0.1820	0.1202	0.1351	0.1107	0.1066	0.1137	0.1177	0.2083	0.2769	0.1258	0.0613
	σ_n=0.03	P	25.71	27.65	30.85	29.30	29.56	29.17	28.29	26.90	22.09	30.73	29.33
		S	0.7608	0.7921	0.8585	0.7938	0.8007	0.7940	0.7904	0.6179	0.5054	0.8569	0.8342
		F	0.9340	0.9530	0.9524	0.9644	0.9658	0.9635	0.9617	0.9503	0.9103	0.9549	0.9682
		L	0.2134	0.1774	0.1644	0.1551	0.1515	0.1630	0.1672	0.2004	0.2781	0.1598	0.0918
	σ_n=0.04	P	25.56	26.81	29.96	28.22	28.52	27.95	26.85	27.14	23.79	29.84	28.61
		S	0.7470	0.7705	0.8388	0.7775	0.7870	0.7759	0.7747	0.6420	0.5598	0.8372	0.8169
		F	0.9281	0.9405	0.9418	0.9515	0.9539	0.9501	0.9470	0.9520	0.9233	0.9426	0.9536
		L	0.2300	0.2243	0.1848	0.1994	0.1944	0.2121	0.2209	0.2016	0.2465	0.1826	0.1248

Table A.2: Non-blind deblurring results for a subset of the BSDS500 [Arb+11] validation dataset with kernels from [Sch+15] (same as in [Big+17]) and Set5 with two kernels from [Lev+09] (same as in [Zha+21a])

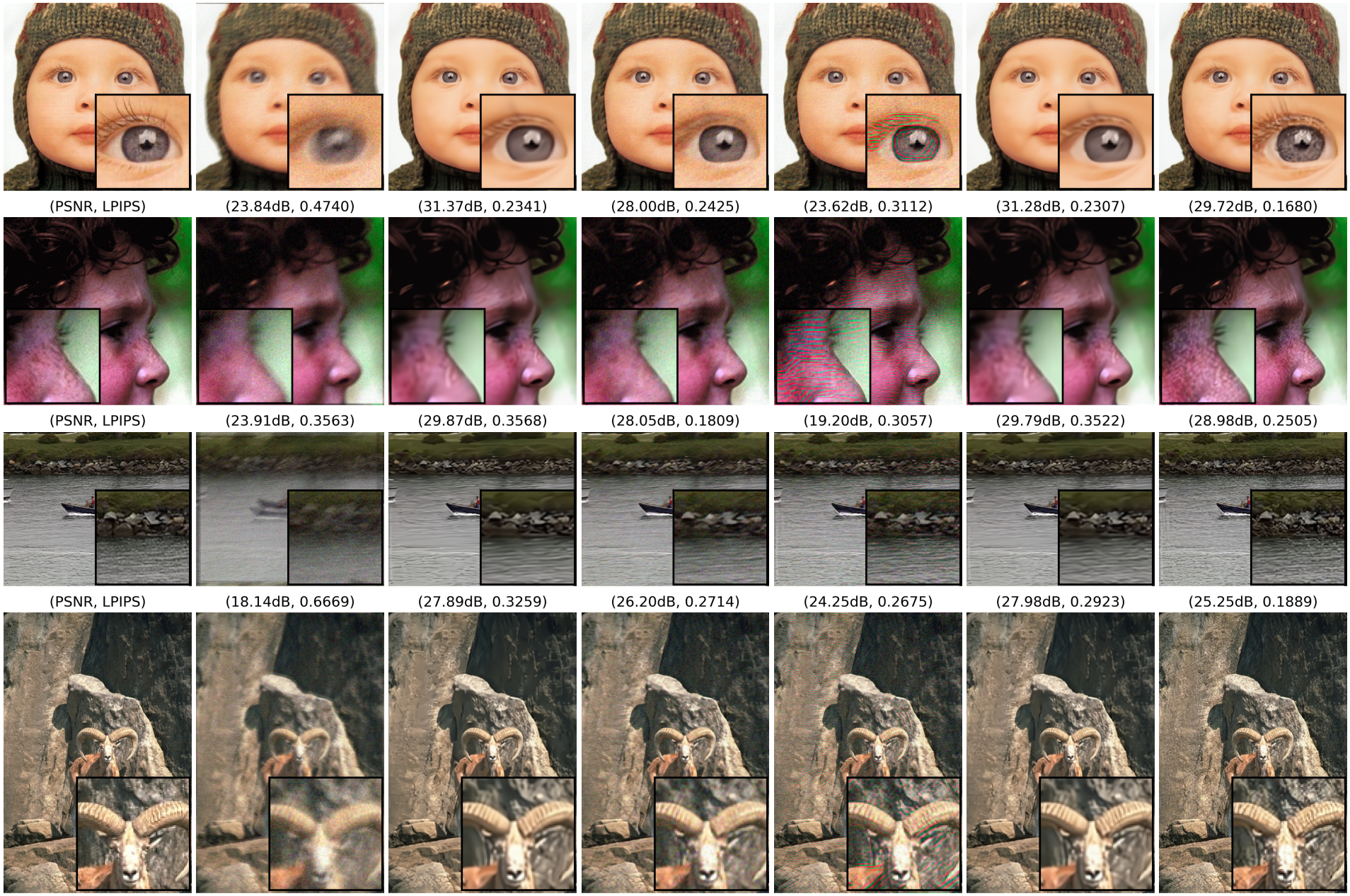


Figure A.2: Denoising results on four images of Set5, Set14, and CBSD68 [Mar+]. The second, third, fourth, and fifth blur kernel from Levin et al. [Lev+09] and a noise standard deviation of $\sigma_n = 0.04$ were used.

			Bicubic	EDSR [Lim+17]	ESRGAN [Wan+19b]	DMSP						HQS	
						$\text{DCNN}_{0.05}$	$\text{DRCNN}_{0.05}$	$\text{DUNet}_{0.05}$	$\text{DRUNet}^{+}_{0.05}$	$\text{DRUNet}^{+}_{0\ldots0.2}$	$\text{DRUGAN}^{+}_{0\ldots0.2}$	$\text{DRUNet}^{+}_{0\ldots0.2}$	$\text{DRUGAN}^{+}_{0\ldots0.2}$
Set5	s=2	P	31.81	36.03	—	35.17	35.30	34.79	32.86	34.97	35.07	34.97	34.35
		S	0.9097	0.9454	—	0.9405	0.9411	0.9402	0.9365	0.9393	0.9399	0.9388	0.9385
		F	0.9864	0.9973	—	0.9967	0.9968	0.9964	0.9939	0.9966	0.9967	0.9962	0.9959
		L	0.1282	0.0544	—	0.0627	0.0642	0.0663	0.0718	0.0692	0.0608	0.0751	0.0573
	s=3	P	28.62	32.59	—	31.56	31.66	31.03	28.86	31.32	31.52	31.24	31.27
		S	0.8388	0.9061	—	0.8935	0.8953	0.8922	0.8862	0.8908	0.8954	0.8917	0.8969
		F	0.9600	0.9868	—	0.9846	0.9848	0.9835	0.9778	0.9845	0.9850	0.9821	0.9829
		L	0.2543	0.1231	—	0.1405	0.1434	0.1480	0.1622	0.1586	0.1411	0.1598	0.1370
	s=4	P	26.70	30.47	28.47	29.31	29.46	29.03	27.11	29.16	28.44	29.12	28.83
		S	0.7736	0.8695	0.8155	0.8480	0.8514	0.8454	0.8366	0.8432	0.8398	0.8442	0.8461
		F	0.9297	0.9680	0.9702	0.9634	0.9640	0.9622	0.9571	0.9631	0.9617	0.9581	0.9583
		L	0.3423	0.1714	0.0752	0.2029	0.2071	0.2158	0.2387	0.2362	0.2250	0.2267	0.2109
	s=5	P	25.29	—	—	27.32	27.53	26.75	25.24	27.37	26.59	27.30	26.98
		S	0.7173	—	—	0.7976	0.8040	0.7927	0.7813	0.7966	0.7927	0.7981	0.7960
		F	0.9034	—	—	0.9366	0.9386	0.9332	0.9287	0.9377	0.9354	0.9321	0.9307
		L	0.4052	—	—	0.2583	0.2626	0.2788	0.3130	0.2978	0.2889	0.2838	0.2748
Set14	s=2	P	28.30	31.76	—	30.79	30.93	30.54	29.38	30.86	30.92	30.66	29.97
		S	0.8433	0.8975	—	0.8874	0.8882	0.8871	0.8846	0.8875	0.8888	0.8846	0.8838
		F	0.9757	0.9952	—	0.9940	0.9941	0.9936	0.9925	0.9941	0.9944	0.9931	0.9930
		L	0.1871	0.0887	—	0.1055	0.1100	0.1139	0.1218	0.1226	0.1065	0.1344	0.1055
	s=3	P	25.73	28.36	—	27.69	27.75	27.37	26.41	27.69	27.67	27.52	27.22
		S	0.7407	0.8155	—	0.7977	0.7990	0.7965	0.7926	0.7982	0.7997	0.7945	0.7977
		F	0.9312	0.9734	—	0.9698	0.9703	0.9691	0.9681	0.9708	0.9715	0.9678	0.9699
		L	0.3479	0.1986	—	0.2318	0.2394	0.2446	0.2606	0.2598	0.2400	0.2685	0.2381
	s=4	P	24.24	26.67	24.34	26.00	25.93	25.72	24.91	25.90	25.65	25.80	25.54
		S	0.6643	0.7516	0.6566	0.7296	0.7307	0.7267	0.7225	0.7282	0.7273	0.7258	0.7272
		F	0.8871	0.9368	0.9526	0.9297	0.9300	0.9298	0.9286	0.9322	0.9318	0.9266	0.9287
		L	0.4437	0.2754	0.1328	0.3145	0.3215	0.3304	0.3444	0.3468	0.3317	0.3442	0.3277
	s=5	P	23.27	—	—	24.75	24.74	24.32	23.71	24.70	24.40	24.59	24.43
		S	0.6091	—	—	0.6730	0.6744	0.6691	0.6662	0.6720	0.6698	0.6709	0.6716
		F	0.8561	—	—	0.8885	0.8887	0.8870	0.8860	0.8915	0.8900	0.8855	0.8853
		L	0.5157	—	—	0.3809	0.3898	0.4014	0.4169	0.4213	0.4077	0.4130	0.4001
CBSD68 [Mar+]	s=2	P	28.43	31.31	—	30.61	30.67	30.53	30.51	30.52	30.64	30.41	30.43
		S	0.8414	0.9018	—	0.8915	0.8922	0.8902	0.8905	0.8904	0.8924	0.8881	0.8908
		F	0.9684	0.9924	—	0.9910	0.9912	0.9909	0.9910	0.9911	0.9914	0.9900	0.9911
		L	0.2431	0.1312	—	0.1513	0.1575	0.1651	0.1731	0.1766	0.1530	0.1896	0.1473
	s=3	P	26.04	28.21	—	27.53	27.58	27.12	24.17	27.44	27.52	27.24	27.42
		S	0.7328	0.8074	—	0.7888	0.7894	0.7861	0.7756	0.7872	0.7900	0.7821	0.7907
		F	0.9157	0.9620	—	0.9576	0.9581	0.9568	0.9518	0.9585	0.9595	0.9557	0.9587
		L	0.4168	0.2650	—	0.3042	0.3152	0.3235	0.3619	0.3446	0.3215	0.3520	0.3131
	s=4	P	24.75	26.62	24.34	25.98	26.01	25.92	25.87	25.92	25.87	25.82	25.82
		S	0.6582	0.7367	0.6550	0.7133	0.7139	0.7102	0.7098	0.7119	0.7124	0.7085	0.7114
		F	0.8671	0.9156	0.9346	0.9078	0.9084	0.9092	0.9093	0.9117	0.9118	0.9068	0.9073
		L	0.5128	0.3448	0.1555	0.3968	0.4063	0.4169	0.4297	0.4336	0.4168	0.4280	0.4082
	s=5	P	23.91	—	—	24.97	25.01	24.92	24.89	24.91	24.82	24.86	24.80
		S	0.6080	—	—	0.6585	0.6591	0.6557	0.6551	0.6568	0.6560	0.6551	0.6555
		F	0.8380	—	—	0.8616	0.8620	0.8625	0.8622	0.8657	0.8654	0.8582	0.8571
		L	0.5838	—	—	0.4633	0.4742	0.4860	0.5006	0.5057	0.4918	0.4933	0.4793

Table A.3: Single image super-resolution results for Set5, Set14, and CBSD68 [Mar+] with bicubic downscaling

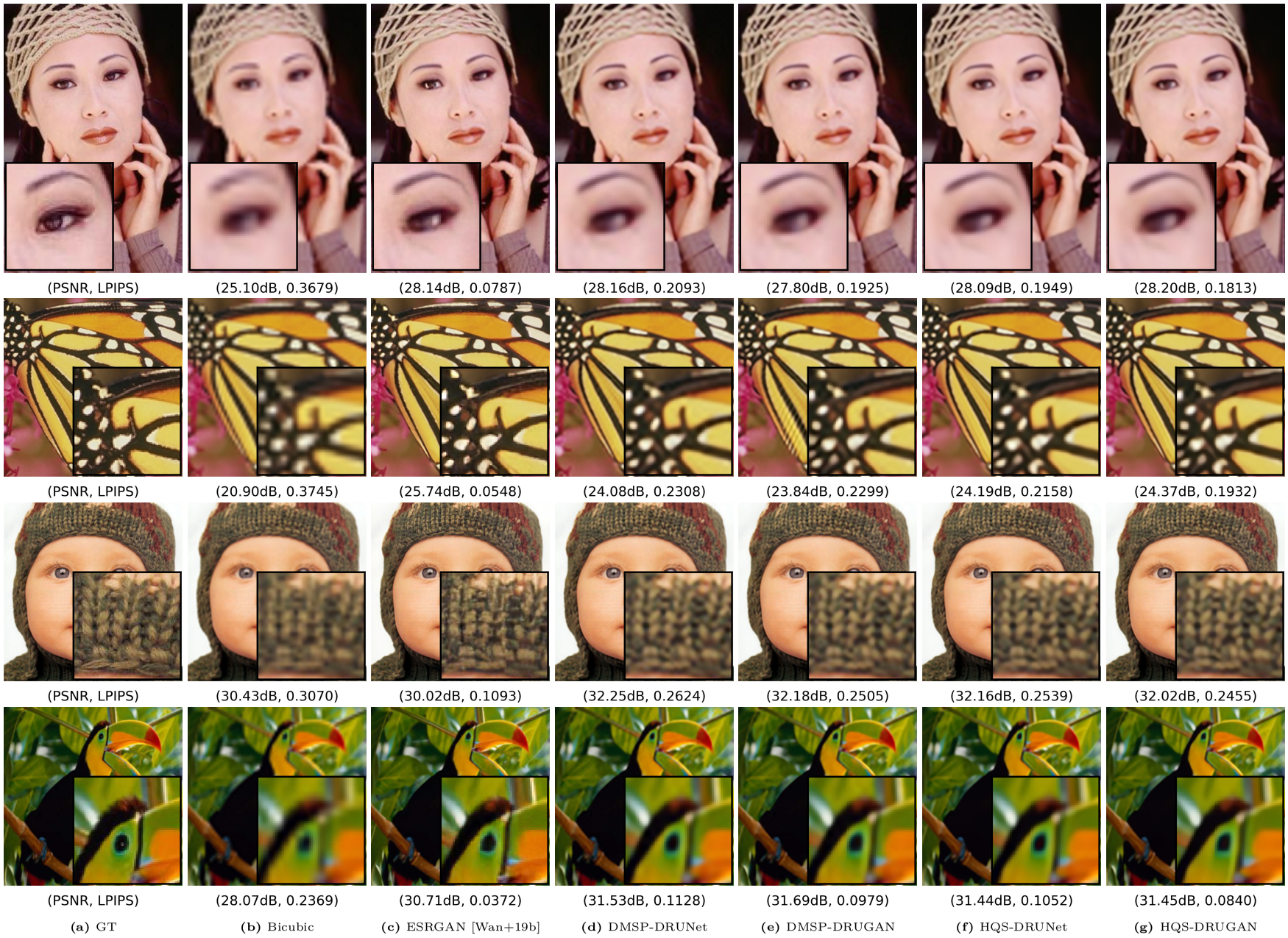


Figure A.3: Single image super-resolution results on four images of Set5 with bicubic downscaling. All examples used a scaling factor of 4.

			Bicubic	EDSR_{bic} [Lim+17]	$\text{ESRGAN}_{\text{bic}}$ [Wan+19b]	DPIR [Zha+21a]	DMSP						HQS	
							$\text{DCNN}_{0.05}$	$\text{DRCNN}_{0.05}$	$\text{DUNet}_{0.05}$	$\text{DRUNet}^{+}_{0.05}$	$\text{DRUNet}^{+}_{0\ldots0.2}$	$\text{DRUGAN}^{+}_{0\ldots0.2}$	$\text{DRUNet}^{+}_{0\ldots0.2}$	$\text{DRUGAN}^{+}_{0\ldots0.2}$
(a)	s=2	P	28.31	28.44	—	34.53	34.65	34.77	34.28	32.99	34.43	34.18	34.60	33.28
		L	0.1388	0.0580	—	0.0800	0.0534	0.0550	0.0562	0.0613	0.0503	0.0478	0.0731	0.0582
	s=3	P	24.77	23.13	—	30.08	30.28	30.49	29.74	27.84	30.19	29.72	30.43	30.05
		L	0.2350	0.1423	—	0.1642	0.1164	0.1197	0.1209	0.1444	0.1102	0.1054	0.1575	0.1335
	s=4	P	22.67	20.30	16.94	26.61	27.22	27.39	27.05	25.32	27.48	25.83	27.74	27.05
		L	0.3386	0.2430	0.3956	0.2277	0.1744	0.1774	0.1805	0.2210	0.1702	0.1646	0.2193	0.2017
(b)	s=2	P	27.05	27.47	—	34.82	35.06	35.15	34.79	32.76	34.96	35.19	35.00	34.16
		L	0.2385	0.1900	—	0.0873	0.0813	0.0830	0.0856	0.0910	0.0800	0.0684	0.0783	0.0580
	s=3	P	24.60	24.37	—	31.02	31.43	31.50	30.64	28.64	31.15	31.29	31.16	31.18
		L	0.2834	0.1511	—	0.1703	0.1360	0.1386	0.1441	0.1612	0.1498	0.1341	0.1582	0.1356
	s=4	P	22.74	21.50	19.86	27.98	28.51	28.54	27.86	26.30	28.52	28.03	28.53	28.14
		L	0.3490	0.1904	0.1814	0.2435	0.1921	0.1981	0.2065	0.2403	0.2138	0.2028	0.2254	0.2107
(c)	s=2	P	25.94	26.13	—	34.20	33.67	33.72	33.36	30.81	33.88	34.39	34.44	34.02
		L	0.3130	0.2919	—	0.1153	0.1271	0.1299	0.1341	0.1437	0.1252	0.1063	0.1033	0.0706
	s=3	P	24.25	24.41	—	31.15	31.64	31.76	30.90	28.74	31.33	31.66	31.29	31.46
		L	0.3351	0.2512	—	0.1748	0.1484	0.1511	0.1574	0.1720	0.1640	0.1444	0.1611	0.1368
	s=4	P	22.68	22.29	21.77	28.41	29.03	29.01	28.31	26.59	28.89	28.57	28.84	28.56
		L	0.3722	0.2125	0.1202	0.2471	0.2004	0.2062	0.2169	0.2456	0.2321	0.2196	0.2270	0.2121
(d)	s=2	P	24.95	25.02	—	32.97	31.99	32.07	31.61	29.20	32.10	32.62	33.19	33.05
		L	0.3693	0.3604	—	0.1513	0.1639	0.1674	0.1743	0.1894	0.1728	0.1481	0.1381	0.1070
	s=3	P	23.79	23.95	—	31.08	31.36	31.48	30.68	28.26	31.11	31.48	31.25	31.48
		L	0.3819	0.3416	—	0.1847	0.1690	0.1724	0.1802	0.1987	0.1849	0.1613	0.1699	0.1441
	s=4	P	22.53	22.55	22.32	28.60	29.26	29.30	28.71	26.94	29.10	28.93	29.02	28.80
		L	0.4018	0.3020	0.2546	0.2494	0.2050	0.2103	0.2222	0.2461	0.2389	0.2228	0.2279	0.2129
(e)	s=2	P	24.70	24.76	—	32.36	31.41	31.53	31.02	28.97	31.56	31.88	32.59	32.32
		L	0.3845	0.3754	—	0.1639	0.1724	0.1760	0.1839	0.1964	0.1844	0.1603	0.1489	0.1176
	s=3	P	23.62	23.71	—	30.94	30.80	30.93	30.15	27.97	30.67	30.92	31.09	31.25
		L	0.3982	0.3573	—	0.1924	0.1831	0.1868	0.1959	0.2130	0.2005	0.1757	0.1777	0.1513
	s=4	P	22.45	22.39	22.15	28.56	29.08	29.14	28.29	26.78	28.95	28.73	29.01	28.86
		L	0.4174	0.3320	0.2927	0.2492	0.2071	0.2119	0.2242	0.2512	0.2384	0.2212	0.2285	0.2126
(f)	s=2	P	24.28	24.33	—	32.19	31.00	31.28	30.68	28.37	31.30	31.57	32.57	32.35
		L	0.3838	0.3708	—	0.1543	0.1697	0.1688	0.1798	0.1923	0.1737	0.1517	0.1354	0.1051
	s=3	P	23.22	23.19	—	30.45	30.57	30.86	30.01	27.69	30.56	30.74	30.81	30.83
		L	0.3947	0.3442	—	0.1962	0.1753	0.1742	0.1860	0.2030	0.1869	0.1675	0.1738	0.1494
	s=4	P	22.09	21.68	21.22	28.18	28.72	28.90	28.31	26.64	28.69	28.52	28.76	28.46
		L	0.4184	0.3220	0.2555	0.2705	0.2140	0.2166	0.2321	0.2531	0.2466	0.2324	0.2369	0.2245
(g)	s=2	P	24.75	24.86	—	32.74	31.52	31.66	31.19	29.15	31.95	32.18	33.03	32.76
		L	0.3531	0.3343	—	0.1467	0.1576	0.1608	0.1652	0.1758	0.1555	0.1392	0.1320	0.1064
	s=3	P	23.77	24.07	—	30.76	30.94	31.10	30.39	28.17	30.95	31.08	31.01	31.05
		L	0.3670	0.2951	—	0.1824	0.1639	0.1675	0.1724	0.1866	0.1703	0.1545	0.1661	0.1467
	s=4	P	22.63	22.73	22.25	28.26	28.88	28.90	28.38	26.78	28.74	28.46	28.83	28.57
		L	0.3959	0.2697	0.2011	0.2488	0.2060	0.2113	0.2194	0.2411	0.2289	0.2155	0.2240	0.2125
(h)	s=2	P	24.02	24.05	—	31.55	30.48	30.57	29.91	27.87	30.50	30.73	31.83	31.68
		L	0.4159	0.4115	—	0.1853	0.1955	0.2000	0.2097	0.2308	0.2146	0.1900	0.1708	0.1436
	s=3	P	23.26	23.38	—	30.70	30.41	30.52	29.74	27.42	30.29	30.55	30.89	31.15
		L	0.4236	0.4030	—	0.2014	0.1969	0.2011	0.2115	0.2345	0.2159	0.1915	0.1862	0.1617
	s=4	P	22.31	22.47	22.31	28.62	29.19	29.24	28.69	26.84	29.09	28.99	29.07	28.87
		L	0.4344	0.3781	0.3644	0.2514	0.2141	0.2190	0.2307	0.2559	0.2450	0.2267	0.2293	0.2145

Table A.4: Single image super-resolution results for Set5. Downscaling was done with the Gaussian kernels (a)–(h) from Zhang et al. [Zha+21a]. Models marked with the subscript "bic" denote end-to-end frameworks trained for bicubic downscaling.

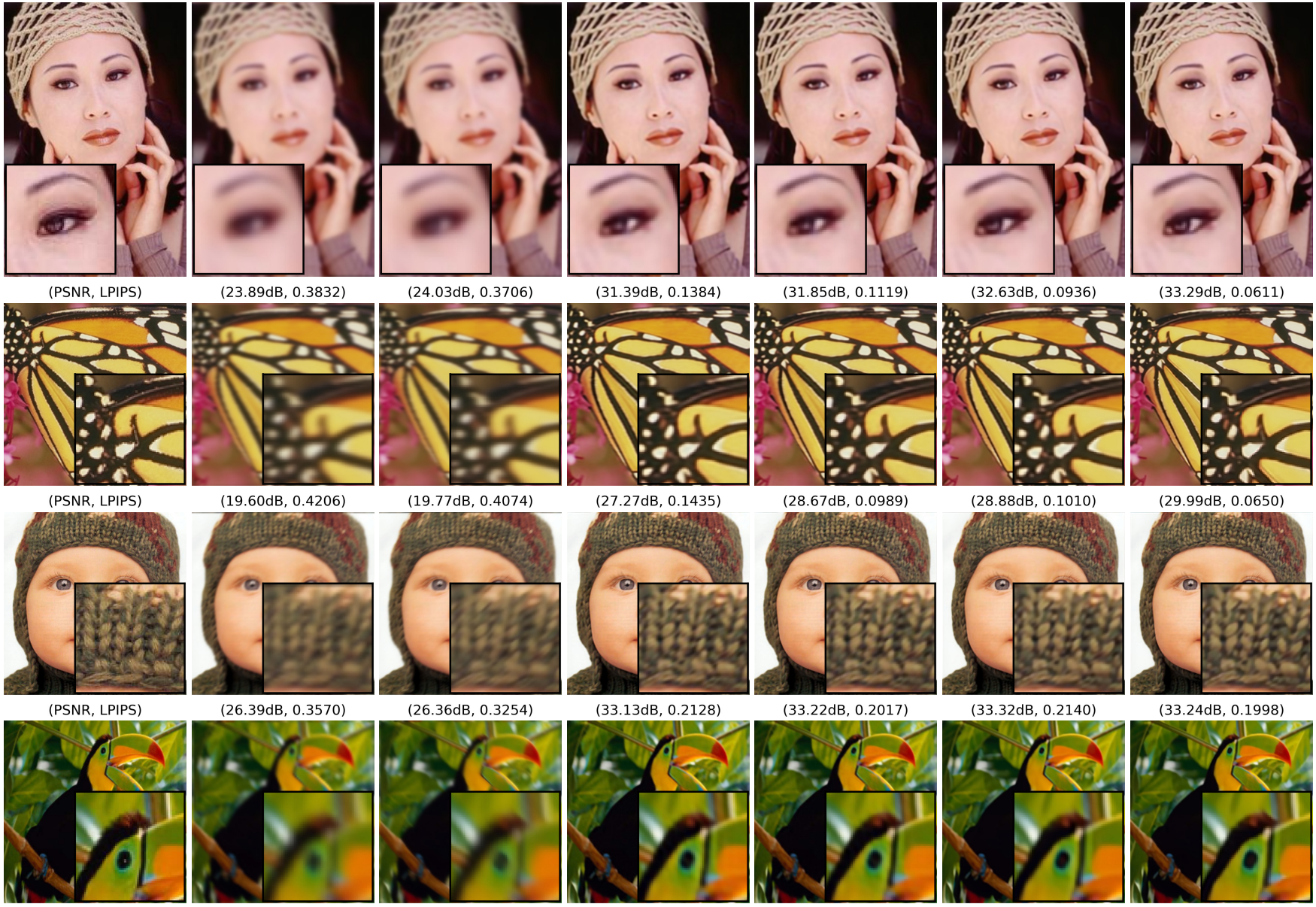
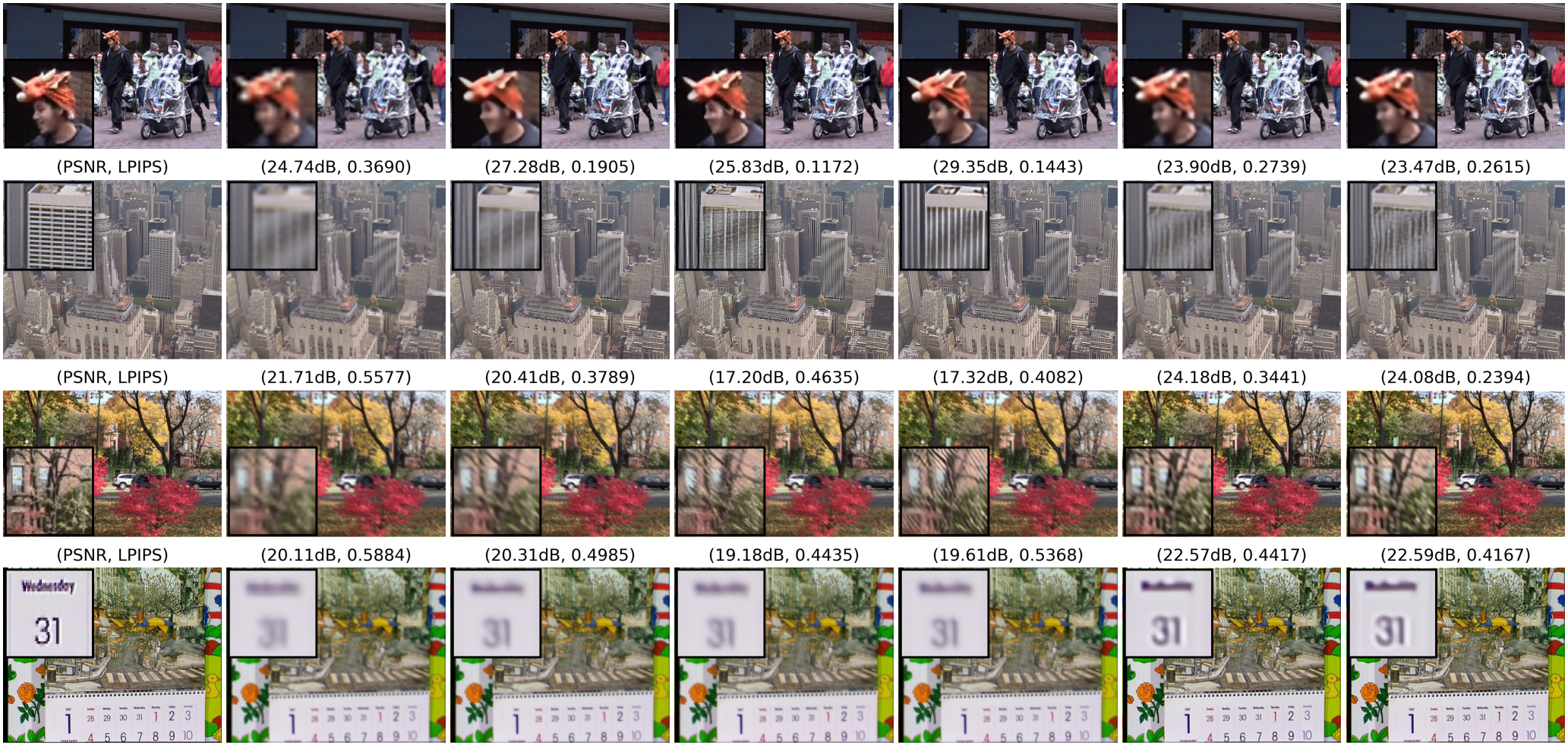


Figure A.4: Single image super-resolution results on four images of Set5 with blur downscaling. The first two examples use a scaling factor of 2, and the last two examples use a scaling factor of 3. All examples use Gaussian blur kernels from [Zha+21a]. An isotropic blur kernel was used for the first example, and anisotropic blur kernels were used for the other examples.

			Bicubic	EDSR_{bic} [Lim+17]	$\text{ESRGAN}_{\text{bic}}$ [Wan+19b]	EDVR_{bic} [Wan+19a]	HQS	
							$\text{DRUNet}^{+}_{0\ldots0.2}$	$\text{DRUGAN}^{+}_{0\ldots0.2}$
bicubic	s=2	P	26.84	30.59	—	—	27.68	27.05
		S	0.8535	0.9263	—	—	0.9042	0.9046
		F	0.9648	0.9932	—	—	0.9863	0.9856
		L	0.2175	0.1036	—	—	0.1847	0.1369
	s=3	P	23.74	25.79	—	—	24.40	24.26
		S	0.7084	0.8111	—	—	0.7951	0.8077
		F	0.8997	0.9612	—	—	0.9556	0.9616
		L	0.3995	0.2326	—	—	0.3417	0.3005
	s=4	P	22.26	23.93	21.62	25.53	22.93	22.84
		S	0.6028	0.7097	0.6006	0.7952	0.6942	0.7018
		F	0.8364	0.9014	0.9256	0.9544	0.8993	0.9047
		L	0.5066	0.3253	0.1903	0.2477	0.4030	0.3838
	s=5	P	21.31	—	—	—	21.92	21.83
		S	0.5294	—	—	—	0.6151	0.6180
		F	0.7983	—	—	—	0.8400	0.8414
		L	0.5961	—	—	—	0.4705	0.4538
(a)	s=2	P	24.44	24.60	—	—	24.06	23.07
		L	0.2217	0.1009	—	—	0.1824	0.1905
	s=3	P	21.31	19.39	—	—	21.78	20.48
		L	0.3919	0.2352	—	—	0.2539	0.3062
	s=4	P	19.65	17.23	13.96	15.94	20.38	19.11
		L	0.5010	0.3462	0.5088	0.3908	0.2910	0.3211
(b)	s=2	P	23.40	24.16	—	—	24.25	23.93
		L	0.3414	0.2942	—	—	0.3015	0.2515
	s=3	P	21.49	21.24	—	—	24.48	23.96
		L	0.4220	0.2719	—	—	0.2726	0.2062
	s=4	P	20.11	18.86	16.31	16.58	22.93	22.49
		L	0.5095	0.3339	0.3558	0.3717	0.3460	0.2786
(c)	s=2	P	22.48	22.82	—	—	22.43	22.29
		L	0.4409	0.4155	—	—	0.3925	0.3669
	s=3	P	21.28	21.44	—	—	23.55	23.49
		L	0.4806	0.3768	—	—	0.3562	0.3161
	s=4	P	20.18	19.82	18.83	18.39	22.83	22.83
		L	0.5387	0.3701	0.2307	0.3173	0.3805	0.3480
(d)	s=2	P	21.72	21.87	—	—	21.16	21.07
		L	0.5310	0.5146	—	—	0.4562	0.4362
	s=3	P	20.96	21.21	—	—	22.41	22.39
		L	0.5486	0.4856	—	—	0.4012	0.3798
	s=4	P	20.12	20.19	20.00	19.92	22.17	22.19
		L	0.5804	0.4617	0.3771	0.3684	0.4076	0.3873
(e)	s=2	P	21.66	21.86	—	—	21.57	21.49
		L	0.5544	0.5386	—	—	0.4582	0.4347
	s=3	P	20.88	21.04	—	—	22.51	22.43
		L	0.5796	0.5256	—	—	0.4176	0.3852
	s=4	P	20.05	20.00	19.77	19.44	22.20	22.16
		L	0.6085	0.5128	0.4450	0.5004	0.4267	0.3982
(f)	s=2	P	21.32	21.43	—	—	20.79	20.67
		L	0.5544	0.5338	—	—	0.4606	0.4419
	s=3	P	20.60	20.60	—	—	22.02	21.97
		L	0.5684	0.5007	—	—	0.3993	0.3716
	s=4	P	19.84	19.51	18.81	18.65	21.84	21.83
		L	0.6001	0.4815	0.4077	0.4660	0.4095	0.3835
(g)	s=2	P	21.62	21.79	—	—	20.91	20.79
		L	0.5112	0.4906	—	—	0.4428	0.4233
	s=3	P	20.98	21.34	—	—	22.19	22.16
		L	0.5283	0.4544	—	—	0.3833	0.3574
	s=4	P	20.21	20.37	19.49	19.71	21.98	22.00
		L	0.5694	0.4324	0.3637	0.4383	0.3917	0.3672
(h)	s=2	P	21.08	21.14	—	—	20.25	20.19
		L	0.6002	0.5917	—	—	0.5109	0.4931
	s=3	P	20.61	20.78	—	—	21.50	21.48
		L	0.6091	0.5748	—	—	0.4485	0.4306
	s=4	P	20.00	20.20	20.11	20.22	21.54	21.56
		L	0.6251	0.5515	0.5084	0.5059	0.4469	0.4309

Table A.5: Video super-resolution results for Vid4 [LS11] with bicubic downscaling and downscaling using the Gaussian kernels from Zhang et al. [Zha+21a]. The metric was computed on the middle frame of the video.



(a) GT **(b)** Bicubic **(c)** EDSR_{bic} [Lim+17] **(d)** $\text{ESRGAN}_{\text{bic}}$ [Wan+19b] **(e)** EDVR_{bic} [Wan+19a] **(f)** HQS-DRUNet **(g)** HQS-DRUGAN

Figure A.5: Video super-resolution results on the videos of Vid4. All examples use a scaling factor of 4. The first example uses bicubic downscaling while the others use blur downscaling with Gaussian kernels from [Zha+21a]. An isotropic blur kernel was used for the second example, and anisotropic blur kernels were used for the third and fourth.

			Bicubic	EDSR_{bic} [Lim+17]	$\text{ESRGAN}_{\text{bic}}$ [Wan+19b]	EDVR_{bic} [Wan+19a]	$\text{LFSSR}_{\text{bic}}$ [Jin+20]	HQS	
								$\text{DRUNet}^{+}_{0\ldots0.2}$	$\text{DRUGAN}^{+}_{0\ldots0.2}$
bicubic	s=2	P	30.99	34.83	—	—	35.32	30.88	29.48
		S	0.8752	0.9259	—	—	0.9468	0.9006	0.8993
		F	0.9741	0.9956	—	—	0.9961	0.9845	0.9817
		L	0.1942	0.0884	—	—	0.0360	0.1731	0.1386
	s=3	P	28.32	31.41	—	—	—	29.53	29.16
		S	0.7889	0.8575	—	—	—	0.8390	0.8437
		F	0.9265	0.9721	—	—	—	0.9570	0.9635
		L	0.3452	0.1886	—	—	—	0.2697	0.2461
	s=4	P	26.86	29.46	26.68	30.70	29.51	28.38	28.26
		S	0.7266	0.8017	0.7101	0.8519	0.8275	0.7844	0.7888
		F	0.8813	0.9310	0.9455	0.9707	0.9595	0.9075	0.9162
		L	0.4491	0.2617	0.1289	0.2025	0.2417	0.3485	0.3265
	s=5	P	25.84	—	—	—	—	27.40	27.30
		S	0.6814	—	—	—	—	0.7419	0.7434
		F	0.8513	—	—	—	—	0.8675	0.8711
		L	0.5310	—	—	—	—	0.4148	0.3977
(a)	s=2	P	28.77	28.81	—	—	28.34	26.73	25.60
		L	0.2001	0.0895	—	—	0.0823	0.2438	0.2272
	s=3	P	25.74	23.68	—	—	—	27.95	26.89
		L	0.3347	0.2059	—	—	—	0.1646	0.1615
	s=4	P	24.01	21.35	16.83	20.54	21.29	28.66	27.92
		L	0.4383	0.3105	0.5499	0.3468	0.3205	0.1440	0.1181
(b)	s=2	P	27.87	28.48	—	—	28.48	26.50	25.01
		L	0.3099	0.2603	—	—	0.2321	0.3343	0.3169
	s=3	P	25.87	25.53	—	—	—	28.17	27.63
		L	0.3749	0.2219	—	—	—	0.2752	0.2452
	s=4	P	24.38	22.77	19.92	21.07	22.31	27.68	27.41
		L	0.4513	0.2777	0.3139	0.3422	0.3093	0.2761	0.2389
(c)	s=2	P	26.99	27.28	—	—	27.27	25.89	24.75
		L	0.4023	0.3786	—	—	0.3709	0.3833	0.3713
	s=3	P	25.66	25.92	—	—	—	27.59	27.19
		L	0.4354	0.3454	—	—	—	0.3373	0.3111
	s=4	P	24.44	23.84	23.03	22.66	23.53	27.37	27.24
		L	0.4827	0.3027	0.1689	0.2928	0.2775	0.3306	0.3069
(d)	s=2	P	26.23	26.35	—	—	26.35	25.18	24.64
		L	0.4763	0.4660	—	—	0.4624	0.4338	0.4246
	s=3	P	25.33	25.58	—	—	—	26.65	26.42
		L	0.4950	0.4470	—	—	—	0.3853	0.3646
	s=4	P	24.37	24.41	24.26	24.11	24.24	26.82	26.73
		L	0.5223	0.4078	0.3443	0.3247	0.3307	0.3743	0.3542
(e)	s=2	P	26.10	26.25	—	—	26.24	24.97	24.05
		L	0.4800	0.4665	—	—	0.4623	0.4342	0.4212
	s=3	P	25.21	25.38	—	—	—	26.48	26.15
		L	0.5033	0.4485	—	—	—	0.3870	0.3630
	s=4	P	24.28	24.06	23.87	23.27	23.84	26.54	26.44
		L	0.5312	0.4254	0.3756	0.4756	0.4023	0.3797	0.3579
(f)	s=2	P	25.81	25.91	—	—	25.90	24.65	24.05
		L	0.4841	0.4687	—	—	0.4648	0.4401	0.4291
	s=3	P	24.94	24.96	—	—	—	26.20	25.90
		L	0.5026	0.4430	—	—	—	0.3902	0.3702
	s=4	P	24.07	23.65	23.06	22.83	23.44	26.42	26.30
		L	0.5349	0.4125	0.3231	0.4230	0.3972	0.3839	0.3644
(g)	s=2	P	26.09	26.24	—	—	26.24	24.79	24.23
		L	0.4624	0.4442	—	—	0.4388	0.4306	0.4246
	s=3	P	25.32	25.69	—	—	—	26.38	26.09
		L	0.4831	0.4141	—	—	—	0.3770	0.3566
	s=4	P	24.45	24.58	23.68	24.37	24.50	26.63	26.51
		L	0.5165	0.3812	0.2991	0.3392	0.3573	0.3661	0.3459
(h)	s=2	P	25.53	25.59	—	—	25.59	24.39	24.00
		L	0.5372	0.5326	—	—	0.5304	0.4800	0.4742
	s=3	P	24.94	25.12	—	—	—	25.76	25.52
		L	0.5497	0.5244	—	—	—	0.4312	0.4143
	s=4	P	24.23	24.44	24.33	24.44	24.40	26.09	25.99
		L	0.5651	0.4988	0.4818	0.4726	0.4582	0.4211	0.4031

Table A.6: Light field super-resolution results for the HCI [Hon+17] test set with bicubic downscaling and downscaling using the Gaussian kernels from Zhang et al. [Zha+21a]. Only the center frame of the light field was upscaled.

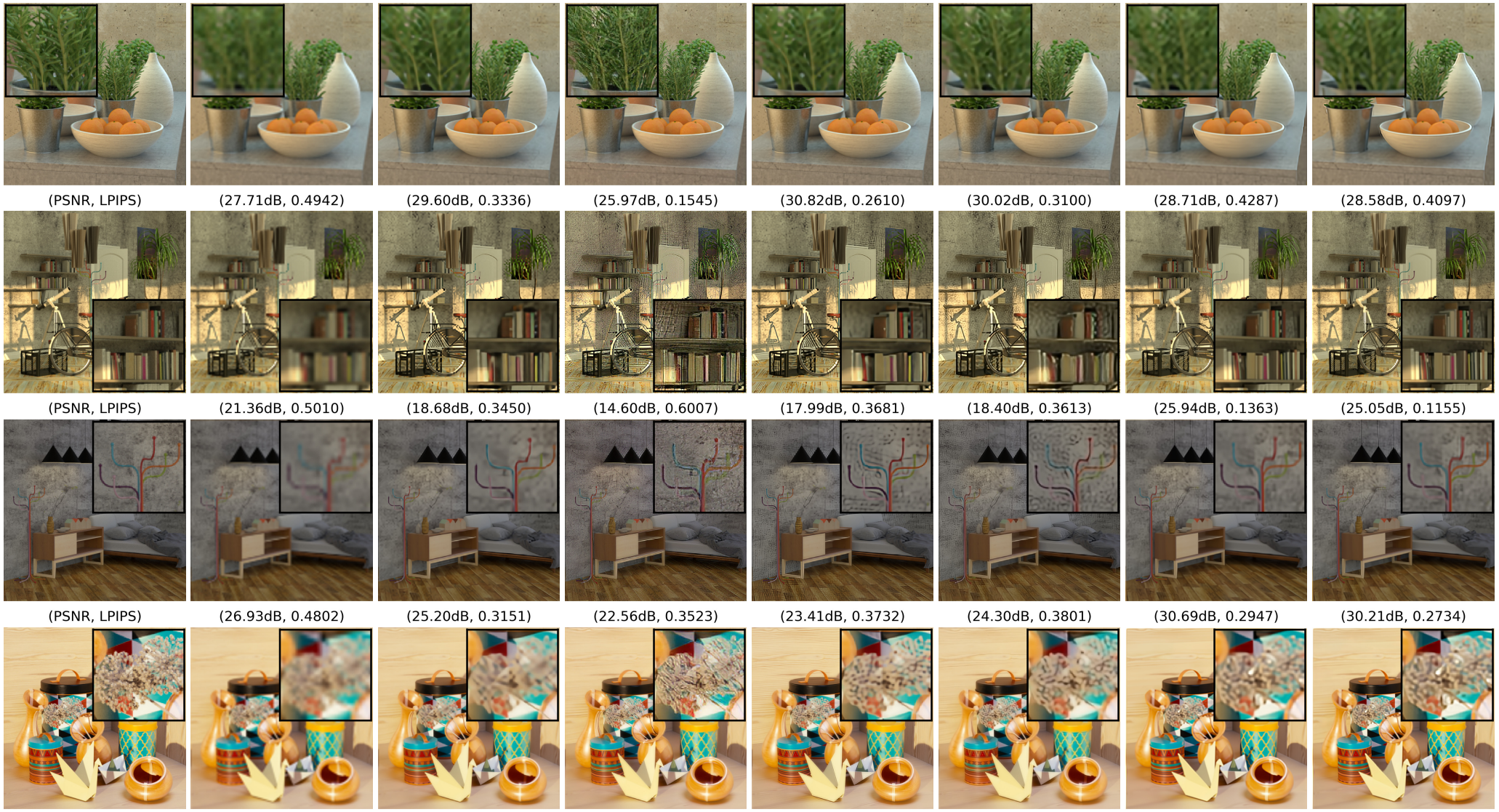


Figure A.6: Light field super-resolution results on the light fields of the HCI [Hon+17] test set. All examples use a scaling factor of 4. The first example uses bicubic downscaling, while the other examples use blur downscaling with isotropic Gaussian kernels from [Zha+21a].