Inductive biases for pixel representation learning

Shi, Z.

Publication date
2022
Document Version
Final published version

As the important part of human intelligence, inductive biases or heuristics allow us to solve problems and make decisions quickly and efficiently. This thesis is dedicated to exploring the inductive biases that humans may exploit to achieve better machine intelligence with deep learning.

Inductive Biases for Pixel Representation Learning

Zenglin Shi

Inductive Biases for Pixel Representation Learning

Zenglin Shi

Inductive Biases for Pixel Representation Learning

Zenglin Shi

This book was typeset by the author using LaTeX 2_ε.

Inductive Biases for Pixel Representation Learning

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. ir. K.I.J. Maex
ten overstaan van een door het College voor Promoties ingestelde commissie,
in het openbaar te verdedigen in de Aula der Universiteit
op vrijdag 29 april 2022, te 14.00 uur

door

Zenglin Shi

geboren te Henan, China

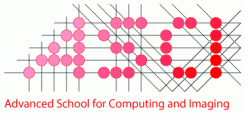
Promotiecommissie

Promotor:	prof. dr. C. G. M. Snoek	Universiteit van Amsterdam
Co-promotor:	dr. P. S. M. Mettes	Universiteit van Amsterdam
Overige leden:	prof. dr. ir. A. W. M. Smeulders	Universiteit van Amsterdam
	prof. dr. ing. Z. J. M. H. Geradts	Universiteit van Amsterdam
	prof. dr. ir. P. H. N. de With	Technische Universiteit Eindhoven
	dr. S. Maji	University of Massachusetts Amherst
	dr. X. Zhen	Universiteit van Amsterdam

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

UNIVERSITEIT VAN AMSTERDAM

This work was carried out in the ASCI graduate school with dissertation number 433, and at the Video & Image Sense Lab, University of Amsterdam.



CONTENTS

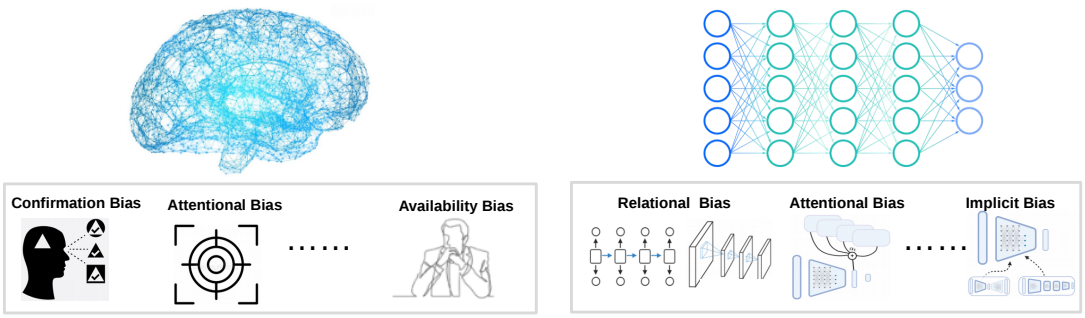
1

INTRODUCTION

In daily life, a wide range of biases are continually influencing us by affecting our thinking, behavior and decisions. For example, when we meet two new friends in the Netherlands, one tall and one short, we may think the taller one is more likely to be Dutch. Generally, a bias is a tendency to lean in favor of or against a person, group, idea, or thing, and it may rely on our experiences and prior knowledge. Biases help us solve problems and speed up our decision-making process by simplifying information processing. There are many examples of bias, see Figure 1(a). For example, the salience bias [161] makes us focus on items that are more prominent or emotionally striking and helps us ignore those that are less remarkable. Attentional bias [162] makes us pay attention to positive stimuli. At the same time, biases can introduce errors. For example, biases may lead to inaccurate judgments about how commonly things occur and how representative certain things may be. Machine-based decision-making also relies on biases.

A machine-learning algorithm with an ability to generalize beyond the data it has seen during training must have an inductive bias. Typically, given a particular dataset and objective function to optimize, there are many possible solutions to the learning problem that exhibit equally good performance on the training data. An inductive bias allows a learning algorithm to prioritize one solution over another, independent of the observed data [137]. Ideally, inductive biases improve the search for solutions without substantially diminishing performance, and also help find solutions which generalize in a desirable way. However, coming up with practical inductive biases, which align with the structure of the problem at hand, is challenging. Mismatched inductive biases can also lead to suboptimal performance by introducing too strong constraints. Several inductive biases have been successfully used in classical machine learning methods. For example, linear regression [201] assumes that the target has a linear relationship with each of the input features. Nearest Neighbors assumes that cases that are near each other tend to belong to the same class. Decision tree [159] assumes that shorter trees are preferred over larger trees.

Inductive biases have also been explored for deep learning, and this is what in part contributed to its success. Some examples of these inductive biases are shown in Figure 1(b), where the relational inductive biases are explicitly encoded into the network architecture. For example, convolutional layers have spatial translation invariance and induce a relational inductive bias of locality, whereas recurrent layers have a temporal invariance that induces the inductive bias of sequentiality [14]. Such relational inductive biases are extremely powerful when well-matched to the data on which they are applied. Besides explicit biases, there are also many implicit biases, which come from implicit regularization [55, 135, 181]. Implicit biases play a crucial role in learning deep neural



(a) **Biases in human learning** (b) **Biases in deep learning**

Figure 1: **Biases in human and deep learning.**

networks. For example, the implicit bias introduced by optimization algorithms enables over-parameterized networks to exhibit good generalization. Different from explicit inductive biases, however, it is hard to explicitly identify and derive a formulation of implicit bias. Also, there is no direct way to control the strength of an implicit bias. As a result, it is difficult to apply an implicit bias in practice. Moreover, we still do not have a good understanding of these implicit biases in different contexts. The current research mostly focuses on identifying implicit biases in the context of image representation learning for image classification [55, 135, 151]. The main focus of this thesis is to uncover and exploit implicit inductive biases in the context of *pixel* representation learning.

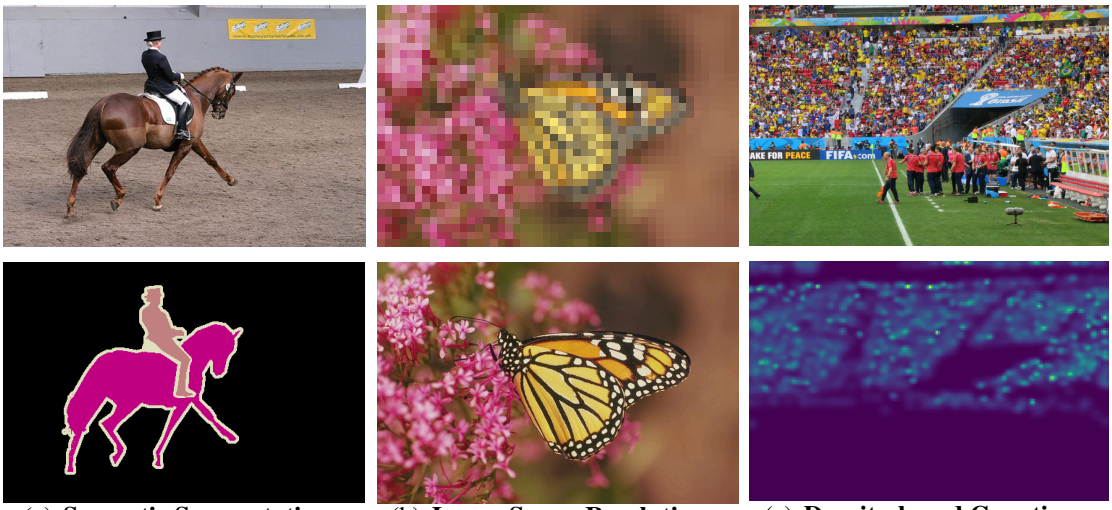
A number of computer vision problems can be formulated as a pixel representation learning problem. Basically, these tasks require a function from a multi-channel input with a spatial dimension to a structured output map with the same spatial dimensions as the input. These include tasks such as semantic segmentation, image restoration, and density-based counting, with typical examples shown in Figure 2. Compared to image-level tasks, pixel-level tasks are more challenging because the pixel values are unstructured and reside in a high-dimensional space. Identifying and understanding the implicit inductive biases for pixel-level tasks is less explored. In addition, it requires more research on encoding new inductive biases into the pixel representation learning by exploiting prior knowledge. We therefore pose the central research question:

How to uncover and exploit inductive biases for pixel representation learning?

As a starting point, we consider the spectral bias. Spectral bias is an inductive bias in neural networks that manifests itself not just in the process of learning, but also in the parameterization of the model itself. This bias leads over-parameterized networks to prioritize learning simple patterns that generalize across data samples. Rahaman *et al.* [151] and Xu *et al.* [208] provide empirical evidence of spectral biases in image classification, *i.e.*, lower frequencies are learned first. We provide an in-depth investigation on spectral bias for pixel representation learning. We pose the research question:

What is the importance of spectral bias for pixel representation learning?

In Chapter 2, we study the spectral bias for generative models with a single image,



(a) **Semantic Segmentation** (b) **Image Super-Resolution** (c) **Density-based Counting**

Figure 2: **Examples of pixel representation learning problems.** The first row shows the input images, and the second row show the output images. The output is either a category classification or a regressed value for each pixel location.

which is known as the deep image prior [186]. The deep image prior showed that a randomly initialized network with a suitable architecture can be trained to solve inverse imaging problems by simply optimizing it's parameters to reconstruct a single degraded image. However, it suffers from two practical limitations. First, it remains unclear how to control the prior beyond the choice of the network architecture. Second, training requires an oracle stopping criterion as during the optimization the performance degrades after reaching an optimum value. To address these challenges, we introduce a frequency-band correspondence measure to characterize the spectral bias of the deep image prior, where low-frequency image signals are learned faster and better than high-frequency counterparts. Based on our observations, we propose techniques to prevent the eventual performance degradation and accelerate convergence. We introduce a Lipschitz-controlled convolution layer and a Gaussian-controlled upsampling layer as plug-in replacements for layers used in the deep architectures. The experiments show that with these changes the performance does not degrade during optimization, relieving us from the need for an oracle stopping criterion.

Prior research has found that the salience bias is encoded into the network architecture and affects model predictions. By computing a saliency map, Simonyan *et al.* [175] revealed that convolutional networks make predictions based mainly on the salient contents in the input image, rather than the entire image. Since then, saliency maps have become a popular visualization tool for gaining insight into how a representation learning model arrives at an individual decision. One may wonder how to explore the salience bias beyond visualization. We therefore pose the research question:

What is the importance of salience bias for pixel representation learning?

We study the salience bias in the context of guided filtering in Chapter 3. The key idea of guided filtering is to leverage an additional guidance image as a structure prior

and transfer the structure of the guidance image to a target image. By doing so, it strives to preserve salient features, such as edges and corners, while suppressing noise. Due to the salience bias, the network can implicitly transfer the structure of the guidance to the target image simply by means of feature fusion from the guidance and target images. Yet, this implicit way of structure-transfer may fail to transfer the desired edges and may suffer from transferring undesired salient content to the target image. To address these problems, we propose a new and simplified formulation of the guided filter. Our formulation enjoys a filtering prior from a low-pass filter and enables explicit structure transfer by estimating a single coefficient. Based on our proposed formulation, we introduce a successive guided filtering network, which provides multiple filtering results from a single network, allowing for a trade-off between accuracy and efficiency. Extensive ablations, comparisons and analysis show the effectiveness and efficiency of our formulation and network, resulting in state-of-the-art results across filtering tasks like upsampling, denoising, and cross-modality filtering.

The attentional bias has been encoded into the network architecture by adding an attention module [24, 49, 214]. The attention module outputs an attention map to emphasize the desirable features. Usually, the attention map is implicitly learned with the task-specific objective. In this way, the learned attention map can not explicitly guide the network to focus on task relevant features. One may wonder about the importance of explicitly encoding the attentional bias into the learning process for a given pixel-level task. We therefore pose the question:

What is the importance of attentional bias for pixel representation learning?

In Chapter 4, we study the attentional bias for the problem of object counting. The leading counting approaches start from point annotations per object from which they construct density maps. Then, their training objective transforms input images to density maps through deep convolutional networks. We posit that the point annotations serve more supervision purposes than just constructing density maps. We introduce ways to repurpose the points for free. First, we propose supervised focus from segmentation, where points are converted into binary maps. The binary maps are combined with a network branch and accompanying loss function to focus on areas of interest. Second, we propose supervised focus from global density, where the ratio of point annotations to image pixels is used in another branch to regularize the overall density estimation. Experiments on six datasets show that our supervised focuses allow the counting network to explicitly emphasize meaningful features and suppress undesired ones, and thus better reduce the counting error, compared to the standard attentional bias [24, 83, 105, 206].

In the first three chapters, we reveal the importance of three different inductive biases for pixel-level tasks. In the last chapter, we seek to develop new inductive biases by exploiting prior knowledge. Prior knowledge is auxiliary information about the learning task that originates from some discovery processes or domain experts [217]. Prior knowledge can be used to guide the learning process for better generalization and faster convergence. However, exploiting the right prior knowledge for a particular type of task is not always straightforward, since prior knowledge is not obvious and its representation varies. We therefore pose the following research question:

How to exploit prior knowledge as inductive bias for pixel representation learning?

In Chapter 5, we explore prior knowledge for object counting. We take a closer look at density-based counting and identify three things that limit every counting approach, and for each we propose a simple network plug-in module as a mitigation. Specifically, (i) we find that predicting densities on the background induces over half the error rate in counting, and we outline a cascade to limit the effect of counting on background pixels; (ii) occlusions are persistent in counting, yet do not occur often enough in training images to learn to handle them properly. We propose an augmentation on both input and density images to learn to be more robust to occlusions; (iii) constructing density maps from point annotations with Gaussian convolutions is suboptimal for counting. We propose an alternative that learns to distill density maps from an auxiliary density prediction network. Such distilled maps are smoother and more robust to noise than their Gaussian counterparts. All three proposals are simple, can be plugged into any density-based counting network and when combined achieves state-of-the-art results.

List of Publications

- **Chapter 2** is based on "On Measuring and Controlling the Spectral Bias of the Deep Image Prior", *International Journal of Computer Vision*, `https://doi.org/10.1007/s11263-021-01572-7`, 2022 [169], by Zenglin Shi, Pascal Mettes, Subhransu Maji, and Cees G. M. Snoek.

  Contribution of authors

  Zenglin Shi: all aspects,
  Pascal Mettes: guidance and technical advice,
  Subhransu Maji: guidance and technical advice,
  Cees G. M. Snoek: supervision and insight.

- **Chapter 3** is based on "Unsharp Mask Guided Filtering", published in *IEEE Transactions on Image Processing*, vol. 30, pp. 7472-7485, 2021 [168], by Zenglin Shi, Yunlu Chen, Efstratios Gavves, Pascal Mettes, and Cees G. M. Snoek.

  Contribution of authors

  Zenglin Shi: all aspects,
  Yunlu Chen: help with experiments,
  Efstratios Gavves: guidance and technical advice,
  Pascal Mettes: guidance and technical advice,
  Cees G. M. Snoek: supervision and insight.

- **Chapter 4** is based on "Counting with Focus for Free", published in *IEEE/CVF International Conference on Computer Vision*, 2019 [170], by Zenglin Shi, Pascal Mettes, Cees G. M. Snoek.

  Contribution of authors

  Zenglin Shi: all aspects,
  Pascal Mettes: guidance and technical advice,
  Cees G. M. Snoek: supervision and insight.

- **Chapter 5** is based on "Three Things Everyone Should Know to Improve Density-Based Counting", in submission to *European Conference on Computer Vision*, 2022, by Zenglin Shi, Pascal Mettes, and Cees G. M. Snoek.

  Contribution of authors

  Zenglin Shi: all aspects,
  Pascal Mettes: guidance and technical advice,
  Cees G. M. Snoek: supervision and insight.

More works by the author are provided in the Complete List of Publications.

2

SPECTRAL BIAS OF THE DEEP IMAGE PRIOR

2.1 INTRODUCTION

This chapter considers the problem of inverse imaging, where the task is to recover the original image from the one that is degraded due to noise, blur, down-sampling and other hardships [16]. This problem is ill-posed, as a degraded image may correspond to several original images. Hence, reconstructing a unique solution that fits the degraded image is difficult, or impossible even, without some prior knowledge about the image or the degradation [46].

The classical computer vision approaches to inverse imaging minimize a regularized cost function to incorporate some prior knowledge into the solution, *e.g.*, [3, 41, 58, 104]. Despite their excellent results, it remains difficult to handcraft an appropriate regularizer and choose a suitable regularisation parameter for a given application because expert knowledge is often required [82, 155]. Rather than providing the priors as input, deep neural networks offer the ability to learn image priors from numerous image samples, *e.g.*, [5, 119, 134]. By doing so, the image priors are gradually encoded into network parameters during training and reused in the inference phase. Despite its promise, the dependence on image pairs seen during training may result in poor generalization of the learned priors [223, 224].

Contrary to the belief that learning on numerous image samples is necessary to obtain useful image priors, Ulyanov *et al.* [186, 187] show that the architecture of a generator network itself contains an inductive bias independent of learning, where a *deep image prior* can be implicitly captured by a particular network architecture like an encoder-decoder. To leverage the deep image prior for solving inverse imaging problems, a suitably designed network is optimized, starting from a random initialization and a random input, on just a single degraded image through gradient descent. The network is able to output a well-restored image, when its optimization is stopped at the right time, with an early-stopping oracle. The literature studying the deep image prior mostly focuses on designing network architectures [28, 30, 63, 67]. However, it remains unclear how to control the deep image prior beyond the choice of the network architecture and prevent performance degradation when an oracle to stop the optimization at peak performance is unavailable. in this chapter, we study the deep image prior from a complementary perspective to address these problems.

As our first contribution, we study the deep image prior through measuring its spectral bias (Section 2.3). We find that both the networks of the original deep image prior [186, 187] and its variants [30, 63] exhibit a spectral bias during optimization, where the low frequency components of the target images are learned better and faster than

the high-frequency components. We believe that the spectral bias leads the networks to capture deep image priors during optimization, beyond the choice of the architecture, since natural images are well approximated by low-frequency components according to the power spectrum [174]. We measure the spectral bias with a new Frequency-Band Correspondence metric and pinpoint why the performance of the deep image prior gradually degrades after reaching a peak during the optimization.

We observe that deep image prior performance degrades when high-frequency noise is learned beyond a certain level, which could affect the high-frequency image details. As our second contribution, we therefore propose to prevent performance degradation by restricting the ability of the network to fit high-frequency noise (Section 2.4). We bound the layers of our network with Lipschitz regularization and introduce a Lipschitz-variant of batch normalization to accelerate and stabilize the optimization. We also observe that widely used upsampling methods, like bilinear upsampling, over-smooth, which introduces a bias towards lower frequencies. This slows down the learning of the desired higher frequencies, delaying optimization convergence. Therefore, we propose an upsampling method which allows controlling the amount of smoothing and is capable of balancing performance and convergence. Besides these two methods for controlling spectral bias, we further introduce a simple automatic stopping criterion to avoid superfluous computation.

Lastly, we demonstrate the effectiveness of our method on four inverse imaging applications and one image enhancement application: image denoising, JPEG image deblocking, image inpainting, image super-resolution and image detail enhancement (Section 2.5). The experiments show that our method no longer suffers from eventual performance degradation during optimization, relieving us from the need for an oracle criterion to stop early. The automatic stopping criterion avoids superfluous computation. Our method also obtains favorable restoration and enhancement results compared to current approaches, across all tasks.

2.2 RELATED WORK

2.2.1 *Inverse problems in imaging*

An inverse problem in imaging is the task of recovering an unknown image $x^* \in \boldsymbol{X}$ from its noisy measurements $y \in \boldsymbol{Y}$, where $y = \mathcal{A}(x^*) + e$. Here $e \in \boldsymbol{Y}$ denotes some noise in the measurements. The mapping $\mathcal{A} : \boldsymbol{X} \to \boldsymbol{Y}$ denotes the forward operator, which could represent various inverse problems, such as an identity operator for image denoising, convolution operators for image deblurring, filtered subsampling operators for super-resolution, *etc.* Since the operator $\mathcal{A}$ has a non-trivial null space, these inverse problems are often ill-posed. Meaning that the solution is unstable with respect to the measurements, or there are several possible solutions that are consistent with the measurements [16]. To solve these ill-posed inverse problems, we review the classical knowledge-driven approaches and the recent data-driven approaches with deep neural networks.

The classical knowledge-driven approaches assume some prior knowledge about the image x^*, such as smoothness [85, 185] or sparsity [34, 45]. These approaches typically

aim to find a solution that fits well with the measurements y and is consistent with the assumed prior knowledge. To do so, an optimization criterion is used, such as the minimization of the l_2 error norm $\|y - \mathcal{A}(x^*)\|^2$. Then, prior knowledge is incorporated into the solution process through regularization. Specifically, Rudin *et al.* [158] leveraged the fact that in natural images nearby pixels tend to have similar values, and proposed a denoising model with the total variation regularization, which promotes smoothness while preserving edges in images. Based on the finding that natural images can be generally coded by structural primitives such as edges and line segments [142], sparse representation-based regularization models, *e.g.*, [34, 45, 149], have been successfully used in image deconvolution tasks. A natural image often has many repetitive local patterns, and thus a local image patch always has many similar patches across the image [43]. This non-local self-similarity prior was later employed in many inverse imaging problems such as image denoising [32], image deblurring [88] and super-resolution [150]. Later, Mairal *et al.* [123] proposed non-local sparse regularization models which combine the local sparsity and the non-local self-similarity into a unified framework, where the similar image patches are simultaneously coded to improve the robustness of the inverse reconstruction. Despite their excellent results, a downside of these approaches is that their handcrafted regularization only captures a fraction of the prior knowledge about the image, limiting the inverse imaging ability of their models [82, 155].

Data-driven approaches leverage large collections of training data to directly compute regularized reconstructions with deep neural networks. The central idea is to create a paired dataset of ground truth images x and corresponding measurements y, which can be done by simulating (or physically implementing) the forward operator $\mathcal{A}$ on clean data. Subsequently, one can train a network to learn a direct mapping from measurements y to the ground truth images x. Most approaches have focused on designing a proper network architecture to learn a high-performing mapping. For example, Dong *et al.* [40] learned a convolutional neural network for image super-resolution, and Jain *et al.* [78] learned a convolutional neural network for image denoising. Mao *et al.* [124] demonstrated convolution neural networks with encoder-decoder architectures perform better for restoring degraded images. Zhang *et al.* [223] proposed to use the convolution neural networks with residual blocks and skip connections to further improve image super-resolution and denoising performance. Ledig *et al.* [92] proposed a generative adversarial network for image super-resolution to recover the finer texture details. Li *et al.* [95] proposed a computationally efficient frequency domain deep network for image super-resolution. Despite their excellent results, these approaches are sensitive to changes or uncertainty to the forward operator $\mathcal{A}$. For image denoising, for example, a specific network needs to be trained for each considered noise level. To remedy this issue, Lefkimmiatis *et al.* [93] proposed a universal denoising network with non-local filtering layers, which is able to handle a wide range of noise levels using a single set of learned parameters. Recently, Chen *et al.* [23] proposed a plugin module, which can be inserted into any backbone networks. This plugin allows the once trained network to be used for multiple forward operators in various image processing tasks, including image smoothing, image denoising, image deblocking, image enhancement and neural style transfer. Wan *et al.* [193] proposed a triplet domain translation network for restoring old photos, in which multiple degradations exist and are mixed. Such supervised approaches typically perform

very well but rely on a paired dataset of ground truth images and their measurements, which may not be available. In this chapter, we consider the unsupervised inverse imaging approach with a deep image prior.

2.2.2 *Deep image prior*

The deep image prior, introduced by [186, 187], revealed the remarkable ability of untrained convolution neural networks to solve challenging inverse problems by optimizing on just a single degraded image. Let $f_\theta : \boldsymbol{Z} \rightarrow \boldsymbol{Y}$ denote a convolutional neural network parameterized by $\theta \in \Theta$, which transforms a tensor/vector $z \in \boldsymbol{Z}$ to a degraded image $y \in \boldsymbol{Y}$. Without training, the network f_θ has no knowledge about high-level semantic concepts such as the categories of objects in the images. However, the deep image prior found that the network does contain knowledge about the low-level structure of natural images. This prior knowledge is sufficient to model the conditional image distribution $p(x^*|y_0)$. Here, the unknown image x^* has to be determined given a measurement y_0, which allows solving inverse problems in imaging. Specifically, we consider energy minimization problems of the type, $\theta^* = \arg\min_\theta E(f_\theta(z); y_0)$ where $E(f_\theta(z); y_0)$ is a task-dependent data term. For inverse imaging problems, y_0 is a noisy, low-resolution, compressed, or occluded image. The minimizer θ^* is obtained using an optimizer such as gradient descent, starting from a random initialization of the parameters. Given a minimizer θ^* obtained by N steps of gradient descent, we obtain a restoration result by $y^* = f_{\theta^*}(z)$. Competitive performance is even feasible when stopping the network optimization with an early-stopping oracle.

The deep image prior has inspired many to investigate how to expand its applications [33, 50, 86, 154, 188], how to improve its performance [6, 28, 106, 133, 231], how to understand its workings [30, 64, 186, 187], and how to avoid its early-stopping oracle [30, 63].

Liu *et al.* [106] and Mataev *et al.* [133] employ extra regularization to boost performance of the deep image prior. Chen *et al.* and Ho *et al.* [28, 67] leverage neural architecture search to obtain a better deep image prior network for improved performance. Asim *et al.* [6] employ deep image prior on image patches, which improves its reconstruction ability. Zukerman *et al.* [231] improve the deep image prior by using a backprojection loss function. These approaches improve results, but still require an oracle to determine when to stop the optimization. In this chapter, we boost the performance of the deep image prior by controlling its spectral bias, and achieve an automatic stopping with a new criterion.

An intuition provided by [186, 187] for the workings of the deep image prior is that their network follows an encoder-decoder architecture, which imposes strong priors about natural images. Heckel and Hand [64] further attribute the effects of the deep image prior to the special architecture with convolutions using fixed interpolating filters. Alternatively, Cheng *et al.* [30] explain the deep image prior from a Bayesian perspective by showing that the model behaves like a stationary Gaussian process at initialization. These works have focused on studying the workings of deep image prior, mostly from the view of the network architecture design. In this chapter, we provide a complementary perspective. We show that the spectral bias leads the networks to capture deep image

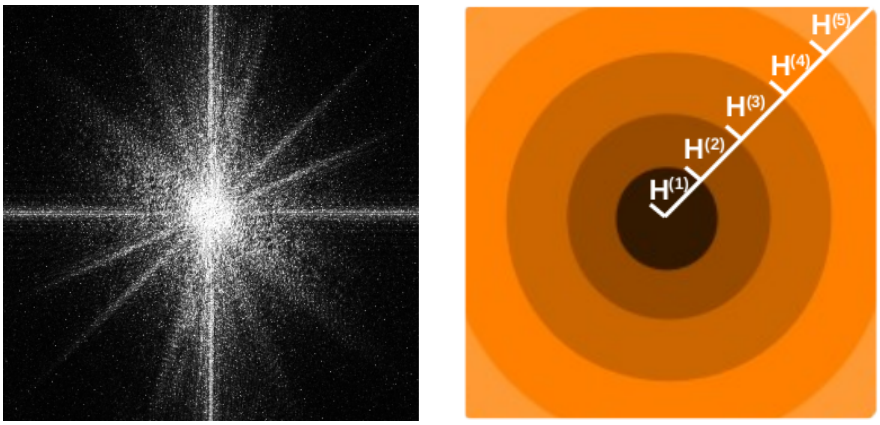


Figure 3: **Frequency-band correspondence metric.** The left image shows an example of correspondence map H, which is computed according to Eq. (2.1). We divide the correspondence map into N subgroups corresponding to N non-overlapping frequency bands. Since the correspondence map is symmetrical around the center, we group it according to the distance between its elements and its center uniformly, as illustrated by the right image when $N = 5$. Different colors represent different subgroups. We compute the mean correspondence for each band to transform the 2D map to the 1D one.

priors during optimization, beyond the choice of the architecture. We do so by introducing a metric, the Frequency Band Correspondence, which offers a spectral measurement of the deep image prior, revealing the low-frequency natural image signals are learned faster and better than high-frequency noise signals.

A downside of the original deep image prior [186, 187] is the requirement of an oracle to determine when to stop the optimization as its performance degrades after reaching a peak over the iterations of optimization. Heckel *et al.* [63] tackle this problem with an underparameterized network, at the expense of reduced performance. Cheng *et al.* [30] avoid the need for early stopping with a Bayesian approach, at the expense of slower convergence. In this chapter, we prevent the performance degradation over iterations with Lipschitz-controlled spectral bias and enable stopping the optimization automatically at an appropriate moment with a new criterion.

A few recent works [21, 151, 208] have paid attention to the spectral bias as well. Rahaman *et al.* [151] and Xu *et al.* [208] analyze the spectral bias for classification problems with supervised learning, not for generative models with a single image. Chakrabarty *et al.* [21] exposed the deep image prior has a spectral bias by adding noise at different frequencies to the image and analyzing the optimization trajectories from different noisy versions of the input. However, they do not measure and control the bias. In this chapter, we propose a frequency band correspondence to measure the spectral bias of the deep image prior. We further control the bias to address the performance degradation problem and the performance-convergence trade-off problem.

2.3 MEASURING SPECTRAL BIAS

The literature attributes the ability of an untrained network to obtain restored results from degraded target images to a particular architecture, like an encoder-decoder, which

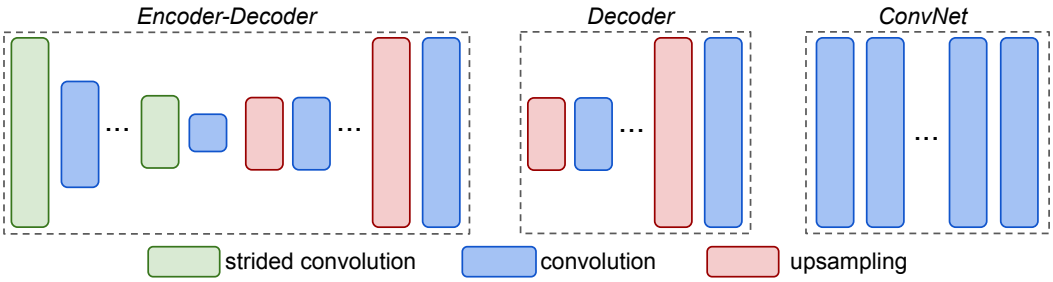


Figure 4: Network architectures used in the experiments of Section 2.3. The *Encoder-Decoder* is the same as the one used in [187]. Specifically, the encoder contains five convolution blocks. Each block contains two convolution layers with the kernel size of 3×3 and the channel number of 128. The stride of the first convolution layer is set to 2 to achieve the downsampling. The decoder contains five bilinear upsampling layers, where each upsampling layer is followed by a convolution layer with the kernel size of 3×3 and the channel number of 128. Each convolution layer is followed by a batch normalization layer and a leaky ReLU layer with a negative slope of 0.01. The *Decoder* is obtained by removing the encoder from the *Encoder-Decoder*. Removing the upsampling layers from the *Decoder* finally leads to the *ConvNet*.

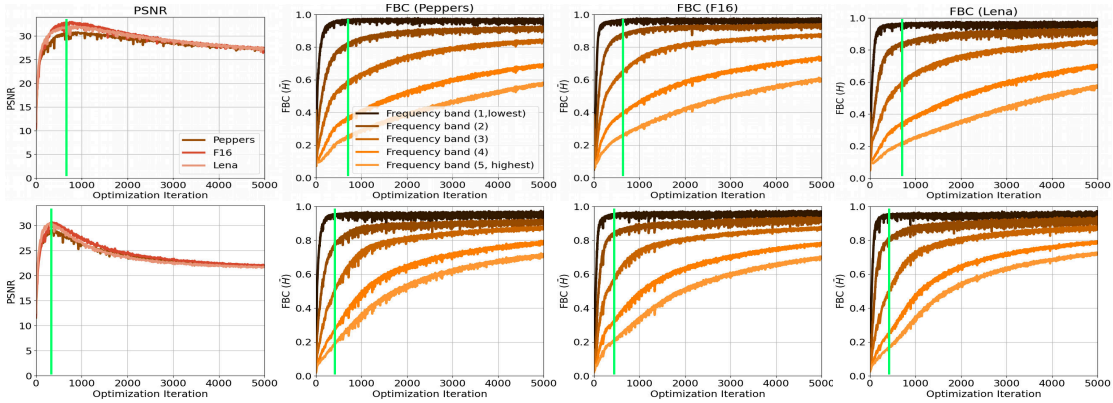
imposes strong priors about natural images. In this chapter, we show that the spectral bias leads the networks to capture deep image priors during optimization, beyond the choice of the architecture. We do so by introducing a metric, the Frequency Band Correspondence, which offers a spectral measurement of the deep image prior, revealing the low-frequency natural image signals are learned faster and better than high-frequency noise signals, and pinpoint why inverse images can be restored, when the network optimization is stopped at the right time.

2.3.1 *Frequency-band correspondence metric*

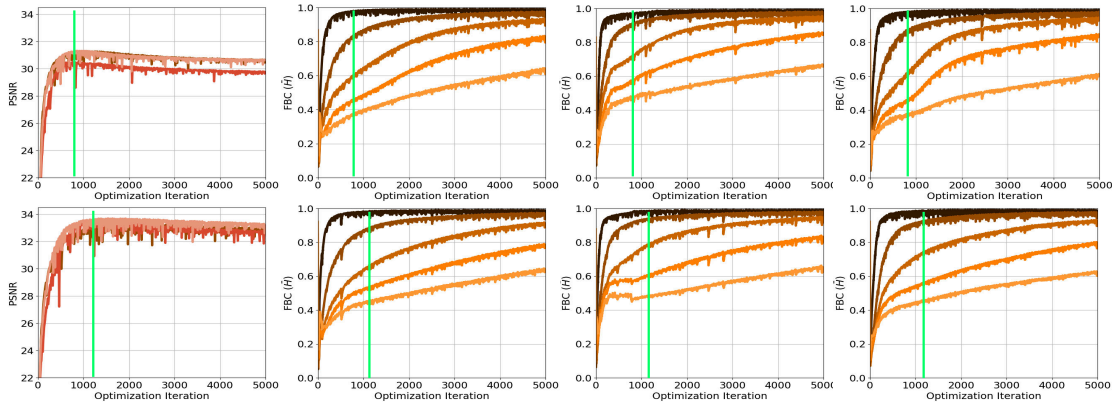
The proposed Frequency-Band Correspondence metric examines the input-output correspondence in the frequency domain across several frequency bands. For this metric, let $\{\theta^{(1)}, \ldots, \theta^{(T)}\}$ denote the trajectory of T steps of gradient descent in the parameter space and let $\{f_{\theta^{(1)}}, \ldots, f_{\theta^{(T)}}\}$ denote the corresponding trajectory in the output space. We propose to analyze the Fourier spectrum of the output images $f_{\theta^{(t)}, t=1,\ldots,T}$ to show the convergence dynamics of different frequency components of the target image. The Fourier spectrum of the output image $f_{\theta^{(t)}}$ is obtained by the Fourier transform $\mathcal{F}$, denoted as $\mathcal{F}\{f_{\theta^{(t)}}\}$ for step t. We similarly compute the Fourier transform for the target image y_0, denoted as $\mathcal{F}\{y_0\}$. We then compute an element-wise correspondence between both transforms as:

$$H_{\theta^{(t)}} = \frac{\mathcal{F}\{f_{\theta^{(t)}}\}}{\mathcal{F}\{y_0\}}. \tag{2.1}$$

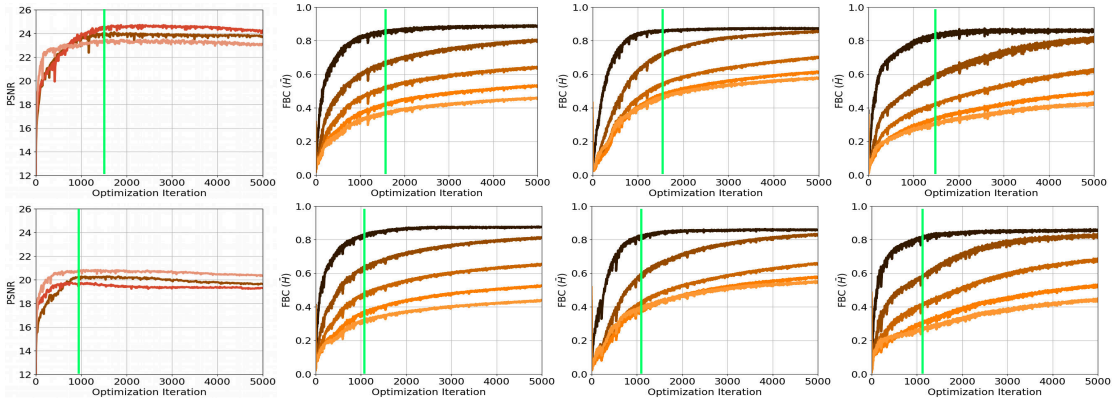
Intuitively, $H_{\theta^{(t)}}$ denotes to what extent any deep image prior at step t corresponds with image y_0 in the frequency domain; the closer the values are to 1, the higher the correspondence. As we are interested in the spectral bias of the deep image prior, we divide the correspondence map into N subgroups corresponding to N non-overlapping frequency bands. Since the correspondence map is symmetrical around the center, we



(a) **Image denoising (top:** σ=15**, bottom:** σ=25)



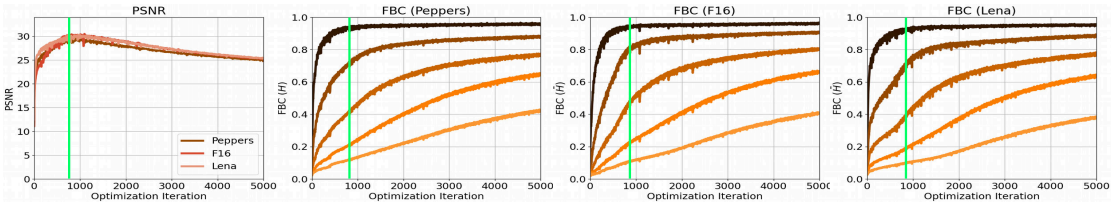
(b) **JPEG image deblocking (top:** *quality*=10**, bottom:** *quality*=20)



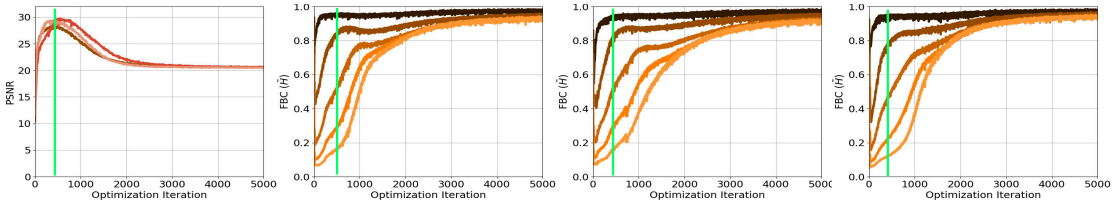
(c) **Image inpainting (top:** *ratio*=0.1**, bottom:** *ratio*=0.25)

Figure 5: **Spectral measurement of the deep image prior** on image denoising, JPEG image deblocking and image inpainting. The network of the deep image prior [187] exhibits a spectral bias during optimization across inverse inaging problems, degradation levels and degraded images, where lower frequencies are learned faster and better than high-frequencies. The degraded images can be restored well when optimizations are stopped at the right time, as marked by the green vertical lines.

group it according to the distance between its elements and its center uniformly, as illustrated in Fig. 3. To transform the 2D map to the 1D one, we compute the mean



(a) ***Decoder*** **architecture**



(b) ***ConvNet*** **architecture**

Figure 6: **Spectral measurement of the deep image prior** with different architectures on image denoising. The spectral bias is not specific to the *Encoder-Decoder* architecture of [187]. Alternative architectures, such as a *Decoder* and a *ConvNet*, also exhibit a bias towards specific image frequencies during optimization. Also, the *ConvNet* learns higher frequencies faster than the *Decoder* by removing the upsampling layers, but at the expense of reduced peak performance.

correspondence for each band, denoted as $\bar{H}^{(n)}_{\theta^{(t)}}$, with $n{=}1,\ldots,N$. The value of $\bar{H}^{(n)}_{\theta^{(t)}}$ indicates the convergence dynamics of different frequency components of a target image.

2.3.2 *Spectral measurement of deep image prior*

We use this metric, denoted as FBC (Frequency-Band Correspondence), to measure how well the network output of the deep image prior corresponds to the target image as a function of N frequency bands. Since the FBC metric is computed with the Fourier transform, our spectral measurement in this section denotes the frequency domain analysis. The Fourier transform $\mathcal{F}$ in Eq. (2.1) is implemented by means of the 2D Fast Fourier Transform, where only the magnitude is used to compute the Fourier spectrum of the images. We use $N{=}5$ where frequency bands are divided into the lowest frequency, low frequency, medium frequency, high frequency and the highest frequency. We perform empirical studies on three inverse imaging problems, including image denoising, JPEG image deblocking, and image inpainting with the 'peppers', 'F16' and 'Lena', images from [32]. For image denoising, the image is degraded by adding Gaussian noise with two noise levels, including $\sigma{=}15$ and $\sigma{=}25$, following [223]. Following [39], we evaluate JPEG image deblocking on the gray-scale images, which are compressed with the PIL encoder into two quality levels, including *quality*$=$10 and *quality*$=$20. For image inpainting, the image is degraded by using a central region mask, and we consider two hole-to-image area ratios, including *ratio*$=$0.1 and *ratio*$=$0.25, following [145]. Following [186, 187], the network input is given as uniform noise between 0 and 0.1 with a depth of 32 by default.

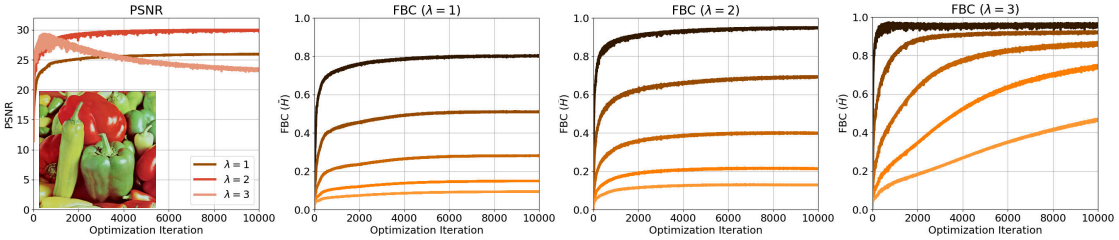
First, we investigate whether the network of the original deep image prior exhibits any form of spectral bias in its optimization. We take the *Encoder-Decoder* architecture of [186, 187] and show its Frequency Band Correspondences for five frequency bands in Fig. 5. The plot highlights, across inverse imaging problems, degradation levels and degraded images, low frequencies are learned quickly and with high correspondence to the target image, while high frequencies are learned slower and with lower correspondence. We conclude that the network of the deep image prior during optimization has a spectral bias towards low frequencies, and this bias helps to obtain a meaningful performance. The peak PSNR (Peak Signal-to-Noise Ratio) performance of the deep image prior occurs when the lowest frequencies are matched nearly perfect, while the highest frequencies are less used, as marked by the green vertical lines. However, once the higher frequencies obtain a higher correspondence, the performance starts to drop.

Next, we show that such a spectral bias is not specific to the *Encoder-Decoder* architecture. We take two other architectures as examples, as shown in Fig. 4. We remove the Encoder from the *Encoder-Decoder* architecture of [186, 187] to obtain the *Decoder*. We additionally remove the upsampling layers from the *Decoder* to get the *ConvNet*. Fig. 6(a) and 6(b) show that both *Decoder* and *ConvNet* learn low-frequency components of the target image faster than learning the high-frequency components, reaffirming the spectral bias. We also observe that *ConvNet* learns high-frequency components faster than *Decoder* by removing the upsampling layers, but at the expense of reduced peak performance. Having established the architecture is not critical for the deep image prior, we use from now on the *Decoder* as the default network architecture to benefit from a good trade-off between performance and run-time.

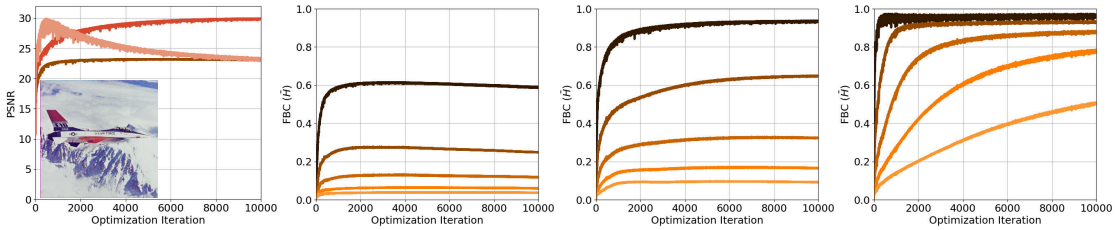
Our study provides a clear implication: untrained solutions for inverse imaging problems work by a latent ability to learn low frequencies faster than learning high frequencies. As natural images are well approximated by low-frequency components, degraded images can be restored well when optimizations are stopped at the right time. The network is optimized to fit the degraded image, in which higher frequencies consist of both structured high-frequency image details and random high-frequency noise. The structured high-frequency image details, that have self-similarity across the image, are fitted better and faster. However, once the random high-frequency noise is fitted over a certain level, which could affect the structured high-frequency image details, the output quality degrades. This behavior explains why the performance in the deep image prior degrades when training longer. Hence, a key enabler for improving the deep image prior is to control the spectral bias by restricting the fitting of random high-frequency noise in the output. Our study also finds that the upsampling layer is beneficial for obtaining good peak performance, but may introduce too much spectral bias towards the low frequencies, slowing down the learning of desired high frequencies. Hence, it's a feasible way to balance peak performance and convergence by controlling the spectral bias in upsampling.

2.4 CONTROLLING SPECTRAL BIAS

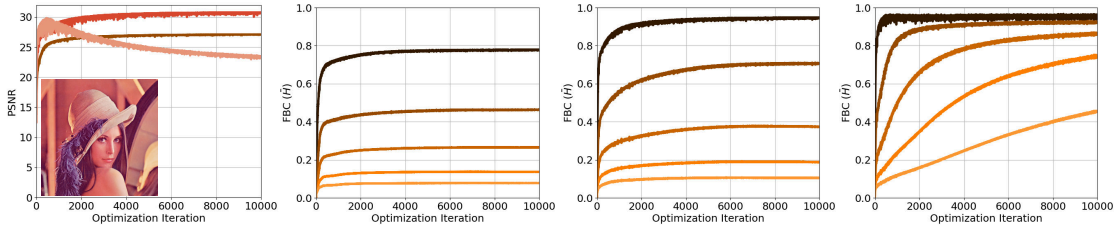
We exploit the measured spectral bias to avoid the degradation of performance over iterations and to balance peak performance and convergence. We do so by controlling spectral



(a) **Peppers**



(b) **F16**



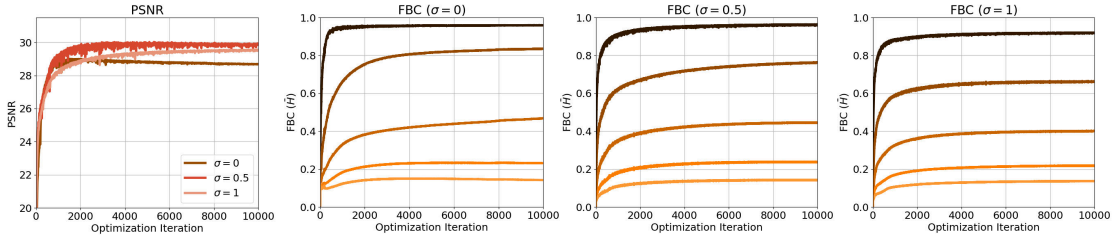
(c) **Lena**

Figure 7: **Lipschitz-controlled spectral bias** for image denoising. Setting the right Lipschitz constant (λ=2) avoids performance decay while maintaining a high PSNR. Different constants result in different levels of spectral bias. A high constant (λ=3) still incorporates a lot of high-frequency noise signals, while a low constant (λ=1) fails to incorporate the important low frequency image signals. With the right balance (λ=2), we maintain the low frequencies while avoiding the high-frequency noise signals.

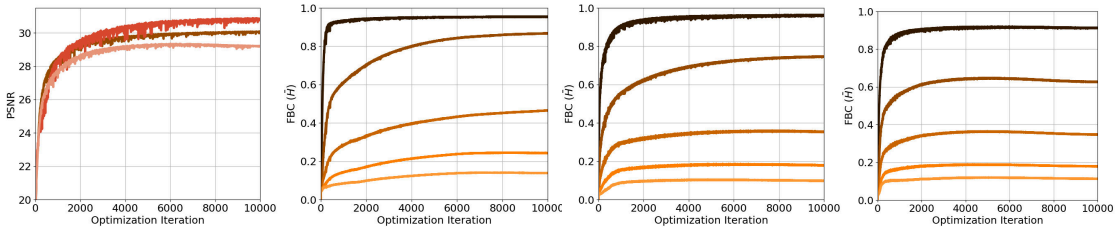
biases in the two core layer types of inverse imaging networks: the convolution layer and the upsampling layer. We present a Lipschitz-controlled approach for the convolution and a Gaussian-controlled approach for the upsampling layer. The approaches are general in their setup, making them applicable to any network form and scale. Besides these two methods for controlling spectral bias, we further introduce a simple stopping criterion to avoid superfluous computation.

2.4.1 *Lipschitz-controlled spectral bias*

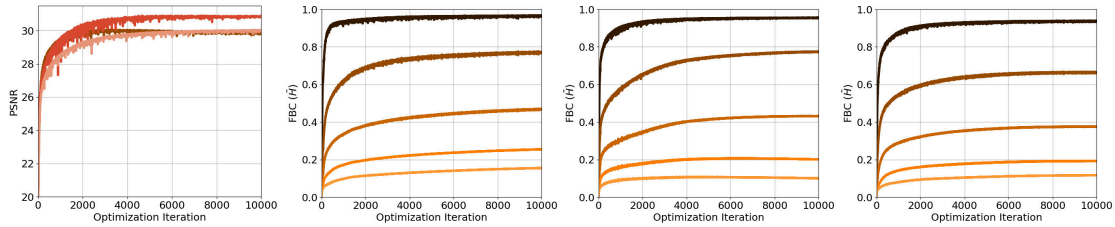
From the point of view of the frequency domain, the Fourier spectrum of the network indicates its ability to learn higher frequencies. Lower frequencies are learned first, while higher frequencies are learned later in the optimization process. This implies that the ability of the network to learn higher frequencies is gradually enhanced by optimizing the learnable layers. Improving the Fourier spectrum of the network is only achievable through adjusting the spectrum of the learnable layers. Based on this observation, we aim to upper bound the Fourier coefficients of the convolutional layers, for the sake



(a) **Peppers**



(b) **F16**



(c) **Lena**

Figure 8: **Gaussian-controlled spectral bias** for image denoising. Varying the Gaussian kernel by σ controls convergence and performance. Too small values (σ=0) results in worse performance, while too big values (σ=1) introduce too much smoothing, slowing down the convergence. With a suitable value (σ=0.5), our upsampling introduces an appropriate spectral bias, leading to fast convergence and good denoising performance.

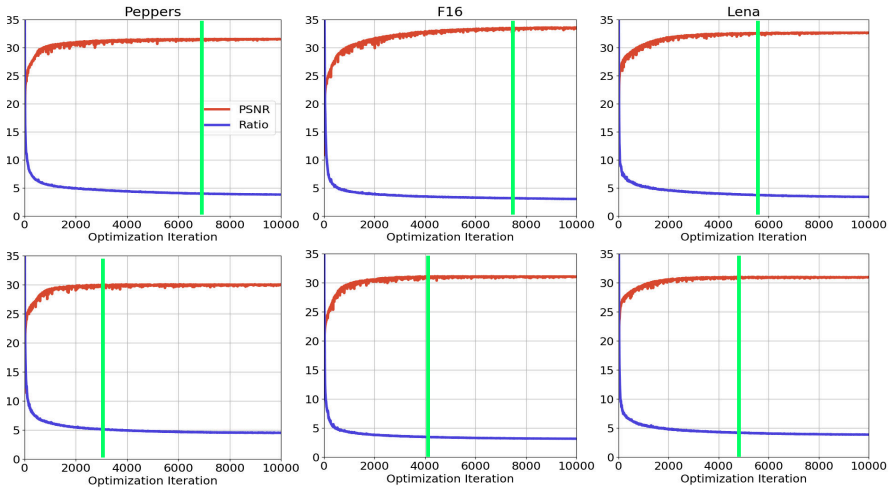
of constraining the Fourier spectrum of the network. We are able to impose an upper bound on the Fourier coefficients of a convolution layer by enforcing Lipschitz continuity, according to [87]. Specifically, if a convolution layer f is Lipschitz continuous, there exists a constant L for any inputs x, y satisfying $\|f(x) - f(y)\| \leq L\|x - y\|$. The minimum over all such values satisfying this condition is called the Lipschitz constant of f, denoted by C. Then the Fourier coefficients of f, *i.e.*, $|\hat{f}(\boldsymbol{k})|$, is bounded by,

$$|\hat{f}(\boldsymbol{k})| \leq \frac{C}{|\boldsymbol{k}|^2}. \tag{2.2}$$

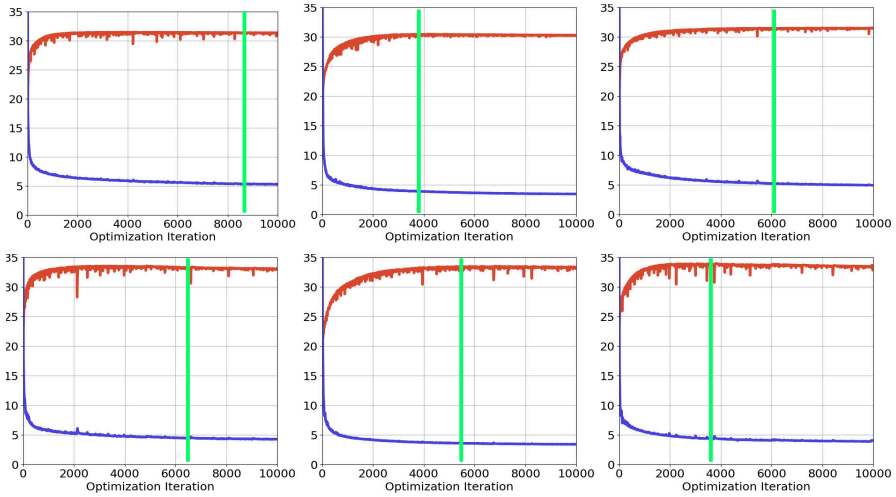
Further, the Lipschitz constant of a convolution layer is bounded by the spectral norm of its parameters. Then we obtain,

$$|\hat{f}(\boldsymbol{k})| \leq \frac{C}{|\boldsymbol{k}|^2} \leq \frac{\|\boldsymbol{w}\|_{sn}}{|\boldsymbol{k}|^2}, \tag{2.3}$$

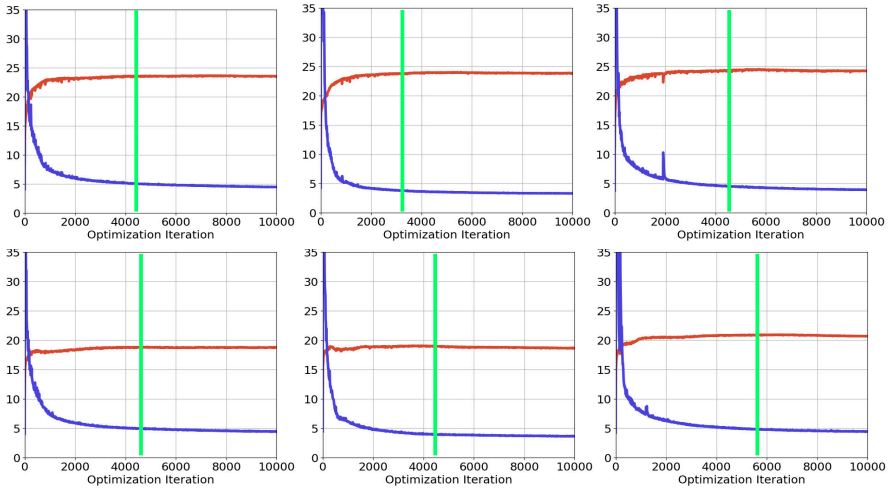
where $\boldsymbol{w}$ is the weight of a convolution layer f, and $\|\cdot\|_{sn}$ denotes the spectral norm, which can be approximated relatively quickly using a few iterations of the power method



(a) **Image denoising (top:** σ=15**, bottom:** σ=25**)**



(b) **JPEG image deblocking (top:** *quality*=10**, bottom:** *quality*=20**)**



(c) **Image inpainting (top:** *ratio*=0.1**, bottom:** *ratio*=0.25**)**

Figure 9: **Automatic stopping criterion** evaluated on image denoising, JPEG image deblocking and image inpainting. The vertical green line shows the selected iteration by the proposed stopping criterion. Across inverse imaging problems, degradation levels and degraded images, we observe the optimization can be stopped earlier, with a minimal performance loss compared to a fixed stop at 10,000 iterations.

[138]. The power law $|\boldsymbol{k}|^{-2}$ indicates that the spectral decay is stronger towards higher frequencies, which means that learning higher frequencies requires a higher spectral norm. Thus, we are able to manipulate the ability of a convolution layer in learning higher frequencies by upper bounding its spectral norm to a specific value λ with $\frac{\boldsymbol{w}}{\max(1,\|\boldsymbol{w}\|_{sn}/\lambda)}$. Where we leave the weight matrix $\boldsymbol{w}$ untouched if its spectral norm is lower than λ. Otherwise, we normalize $\boldsymbol{w}$ by $\|\boldsymbol{w}\|_{sn}/\lambda$.

To accelerate and stabilize the optimization, batch normalization [74] is often used after convolution layers. However, we find it is not compatible with our Lipschitz constraining as its output is invariant to the channel weight vector norm $\|\boldsymbol{w}\|_p$, *i.e.*,

$$BN(\boldsymbol{w}\boldsymbol{x}/\|\boldsymbol{w}\|_p) = BN(\boldsymbol{w}\boldsymbol{x}), \tag{2.4}$$

where $\boldsymbol{x}$ denotes the channel input. We therefore propose a Lipschitz normalization by exploring the idea of combining Lipschitz constraining with a special version of batch normalization: mean-only batch normalization. We only subtract out the mini-batch means, without dividing by the minibatch standard deviations. The Lipschitz normalization is defined as:

$$\mathrm{LN}(\boldsymbol{w},\boldsymbol{x}) = \frac{\boldsymbol{w}\boldsymbol{x}}{\max(1,\|\boldsymbol{w}\|_{sn}/\lambda)} - \mu + b, \tag{2.5}$$

where μ denotes the channel mean of the pre-activation $\boldsymbol{w}\boldsymbol{x}$ and b is a scalar bias term. The Lipschitz normalization layer is inserted between a convolutional layer and a ReLU activation. With this normalization, the Lipschitz constant of a convolution layer is bounded by the hyperparameter λ. As a result, we can manipulate the ability of the network in learning high frequencies by tuning λ, leading to a controlled spectral bias of the deep image prior.

2.4.2 *Gaussian-controlled spectral bias*

Upsampling is an important operation in network architectures for inverse imaging problems, as it produces high-resolution outputs from low-resolution inputs. Well-known approaches such as the bilinear and nearest neighbor upsampling have a constant smoothing effect [21, 64]. Different tasks, however, might operate best under different levels of smoothing. Too strong a smoothing introduces too much spectral bias towards lower frequencies. This slows down the learning of the desired higher frequencies, delaying convergence of optimization (as shown in Fig. 6). Therefore, we propose an upsampling method which allows controlling the amount of smoothing and is capable of balancing performance and convergence.

We first decompose the upsampler into an expansion and a filtering step. Let x_i be the i-th channel of input x. For expansion, x_i is padded with a "bed of nails" scheme, *i.e.*, inserting $s-1$ zeros between the pixels of x_i along its rows and columns. Such a "bed of nails" expansion creates a high-frequency replica of the original signal. To smooth out the noisy high-frequencies, we perform filtering by convolving the upsampled signal

with an interpolating filter. We use a Gaussian filter sampled by $\mathcal{N}(0, \sigma^2)$. Hence, we define our Gaussian upsampling by:

$$\mathrm{Up}(x_i) = \uparrow_s (x_i) * G_\sigma, \tag{2.6}$$

where $\uparrow_s (x_i)$ denotes expanding x_i with factor s, $*$ is the convolution operation, G_σ denotes the Gaussian filter. In the frequency domain, we obtain the Fourier spectrum of our upsampling by,

$$\mathcal{F}(\mathrm{Up}(x_i)) = \mathcal{F}(\uparrow_s (x_i)) \odot \mathcal{F}(G_\sigma), \tag{2.7}$$

where $\mathcal{F}$ is the Fourier transform, $\odot$ is the Hadamard product and $\mathcal{F}(G_\sigma)[k] = 1/e^{2\pi^2\sigma^2k^2}$. We manipulate the Fourier spectrum of our upsampling by choosing different σ, allowing us to control the spectral bias in the upsampling.

2.4.3 *Automatic stopping criterion*

With the ability to control the spectral bias, we can fix the number of iterations for network optimization without fear of performance degradation. As different tasks have different levels of convergence, however, using a fixed number of iterations still leads to redundant optimization. To improve efficiency, we introduce a simple criterion to automatically perform early stopping.

It is well known that an image looks blurry when there is a high amount of low frequencies in its Fourier spectrum. We exploit this property by computing the blurriness and sharpness for an output image and use their ratio as the metric to stop the optimization. In case of a spectral bias, low frequencies will be learned first, while high-frequencies will be learned later. Our Lipschitz normalization limits the ability of the network in learning high frequencies to an upper bound. Hence, when this upper bound is reached, the ratio of blurriness to sharpness of the output image will converge as well. To that end, we design the following measure:

$$\begin{aligned} r(f_\theta) &= \mathcal{B}(f_\theta)/\mathcal{S}(f_\theta), \\ \Delta r(f_\theta{}^t) &= \left| \frac{1}{n}\sum_{i=1}^{n} r\left(f_\theta{}^{(t-i)}\right) - \frac{1}{n}\sum_{i=1}^{n} r\left(f_\theta{}^{(t-n-i)}\right) \right|, \end{aligned} \tag{2.8}$$

where f_θ denotes the output image and $f_\theta{}^{(t)}$ denotes an instance in iteration t. $\mathcal{B}(f_\theta)$ denotes the blurriness of the output image y computed using [31]. $\mathcal{S}(f_\theta)$ denotes the sharpness of the output image y computed using [11]. $r(f_\theta)$ denotes the ratio of blurriness to sharpness of the output image f_θ. Then, $\frac{1}{n}\sum_{i=1}^{n} r\left(f_\theta{}^{(t-i)}\right)$ computes the mean ratio of output images from iteration t to $t-n$, and $\frac{1}{n}\sum_{i=1}^{n} r\left(f_\theta{}^{(t-n-i)}\right)$ computes the mean ratio of output images from iteration $t-n$ to $t-2n$. If their absolute difference is smaller than a constant value ϵ, the optimization is stopped.

Compared to the ratio r itself, the ratio difference Δr between optimization iterations is independent of the images. Since the deep image prior no longer suffers from performance degradation with the controlled spectral bias, the ratio r barely changes when the performance is stable. Thus, we can set the ratio difference threshold ϵ to a small value, like 0.01. As the main benefit of the auto-stopping is to avoid redundant computation, it

does not directly affect the inverse imaging performance. Note that the stopping criterion fails for the original deep image prior [186, 187] because the high-frequency components of its output image keeps increasing until the degraded target image is fully fitted.

2.4.4 *Performance analysis*

We empirically analyze the deep image prior with the Lipschitz-controlled spectral bias, the Gaussian-controlled spectral bias and the automatic stopping criterion.

Lipschitz-controlled spectral bias. Following the work of [186, 187], we use bilinear upsampling in this experiment. In Eq. (2.5), λ is the only parameter which controls the ability of the network in learning high frequencies. Finding the best λ for each image is still an open question. Here we just empirically study three settings, *i.e.*, λ=1,λ=2, and λ=3. The spectral norm $\|\boldsymbol{w}\|_{sn}$ is estimated with the power iteration method [138]. The results are shown in Fig. 7. Setting a suitable constraint (*e.g.*, λ=2) results in a PSNR curve without performance decay. The FBC graphs show this is because setting a low Lipschitz constant amplifies the spectral bias. High frequencies are hardly incorporated at all, while low frequencies still obtain a high correspondence to the target image. Using a too high constraint (*e.g.*, λ=3) results in a similar performance peak and decay as the original deep image prior. When using a too low constraint (*e.g.*, λ=1), we not only suppress high frequencies, but also the low frequencies, which generally corresponds to the structure of the image, hampering the performance. We conclude, utilizing Lipschitz normalization with a suitable value of λ addresses the problem of performance degradation.

Gaussian-controlled spectral bias. Next, we study the effect of the Gaussian-controlled spectral bias to balance performance and convergence. We replace the bilinear upsampling with our Gaussian upsampling and use λ=2 to maintain the effect of the Lipschitz-controlled spectral bias on avoiding performance degradation. We consider Gaussian upsampling with three settings in Eq. (2.6), σ=0, σ=0.5, σ=1 where the kernel size is fixed to 5×5. We show the effect of different settings on the denoising performance and amount of spectral bias in Fig. 8. The smaller the value for σ, the faster the convergence is reached. However, a too small value *e.g.*, σ=0 results in worse performance, because the upsampling reduces to the "bed of nails" expansion. A value of σ=1 introduces too much smoothing, slowing down the convergence. With a suitable value, *e.g.*, σ=0.5, our upsampling introduces an appropriate spectral bias, leading to fast convergence and good denoising performance. Furthermore, compared to the widely used upsampling, like bilinear upsampling (refer to its performance in Fig. 7), our upsampling achieves a better trade-off between performance and convergence. We conclude our upsampling allows to control the spectral bias, enabling us to improve the performance of deep image prior for inverse imaging problems like image denoising.

Stopping criterion. Finally, we analyze the effect of the proposed stopping criterion on image denoising, JPEG image deblocking and image inpainting. For each problem, we evaluate on different degradation levels, as specified before in Section 2.3.2. We use n=100 and ϵ=0.01 throughout the experiment. We set the fixed stopping iteration to 10,000. We show the dynamics of the Peak Signal-to-Noise score and ratio values in Fig. 9. We observe the stopping criterion is effective, it reduces the number of required

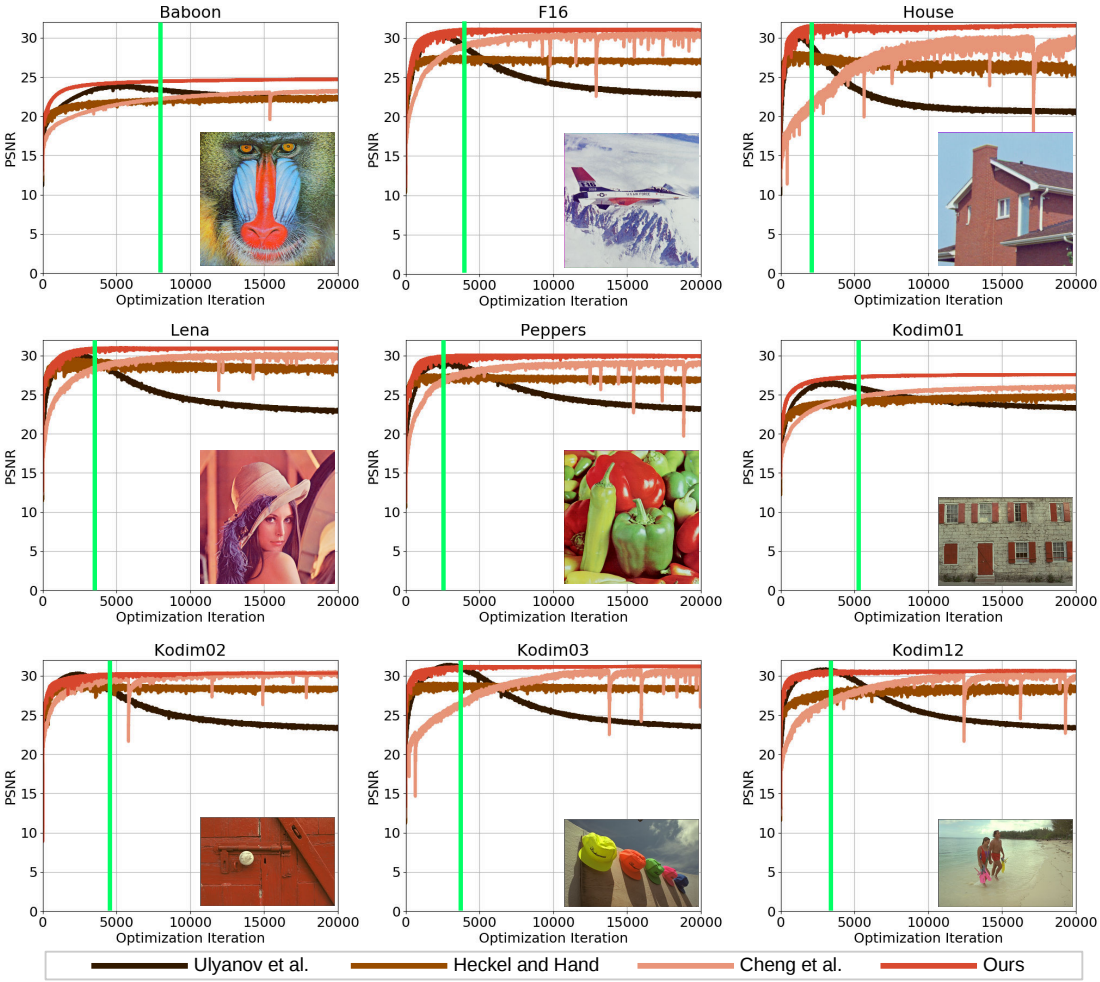


Figure 10: **Image denoising.** PSNR scores of various methods over multiple iterations for removing additive Gaussian white noise with σ=25. Compared to [187], our method doesn't suffer from performance degradation, and we can stop the optimization automatically at an appropriate moment for each image (marked by the green vertical lines), leading to good PSNR scores. Compared to [63] and [30], we either achieve a faster convergence or obtain a higher PSNR score.

iterations considerably with only a minimal loss in performance, across inverse imaging problems, degradation levels, and degraded images. For the worst performing "F16" image for denoising with σ=25, the PSNR drops from 31.04 to 30.98 when reducing the iterations from 10,000 to 3,896. We also found that the performance in terms of PSNR changes less than 0.1 when the ratio difference threshold ϵ ranges from 0.001 to 0.1. A bigger threshold means the optimization stopped earlier.

2.5 APPLICATIONS

With the gained ability to control the spectral bias in the deep image prior, we consider four inverse imaging applications and one image enhancement application for comparative evaluation: image denoising, JPEG image deblocking, image inpainting, image super-resolution and image detail enhancement. On all tasks, we compare to the deep im-

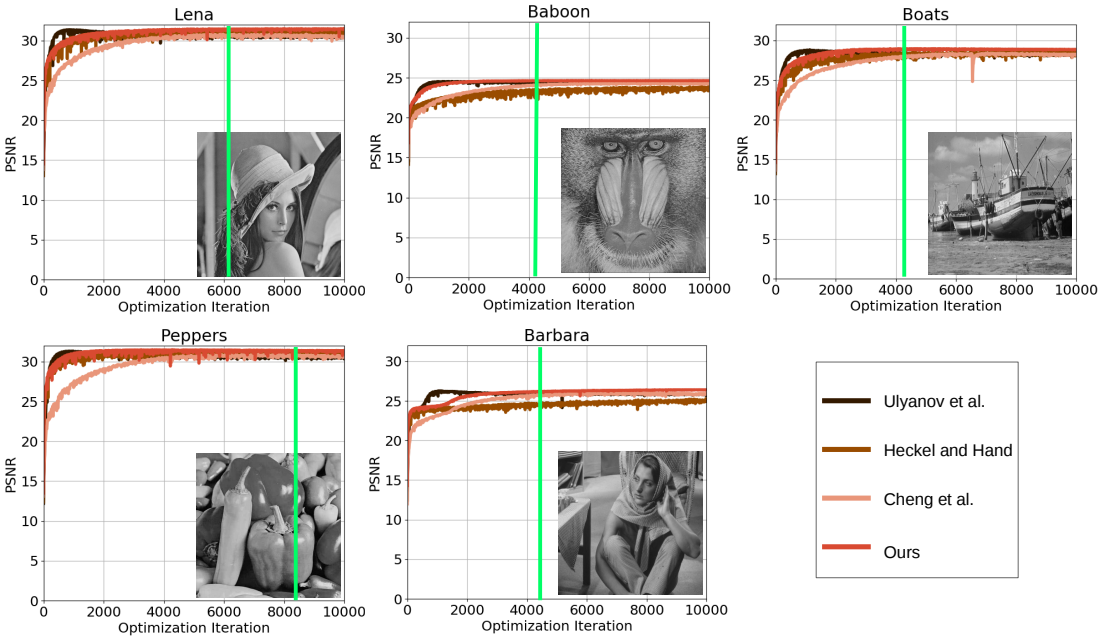


Figure 11: **JPEG image deblocking.** PSNR scores of various methods when reducing artifacts of a compressed JPEG image with *quality*=10. We again observe that the performance of the deep image prior [187] degrades. [30] and [63] do not suffer from degradation, at the expense of either reduced performance or slow convergence. Our method achieves a good trade-off between PSNR score and convergence (marked by the green vertical lines).

age priors of [186, 187], [63] and [30]. For reference, we also report the results obtained by classical methods like [32], and supervised-learning based methods like [223].

We report our results with the *Decoder*, introduced in Section 2.3.2, as our network architecture. Lipschitz normalization with λ=2 and Gaussian upsampling with σ=0.5 are combined into the *Decoder* to achieve a controllable deep image prior. Network parameters are initialized with He initialization [61]. Our approach works with popular optimizers such as standard gradient descent and Adam [89]. Following [186, 187], we use Adam with a mini-batch of 1 to optimize our networks. We set β_1 to 0.9, β_2 to 0.999 and the initial learning rate to 0.001. The network input is a uniform noise between 0 and 0.1 with a depth of 32 by default. Our code will be released.

2.5.1 *Image denoising*

For the denoising comparison we use two datasets, *i.e.*, the standard dataset by [32] consisting of 9 RGB images, and CBSD68 by [156] consisting of 68 RGB images. Each noisy image is generated by adding an additive Gaussian white noise with three noise levels, including σ=15, σ=25 and σ=50. The goal is to distill the original image without Gaussian noise. Results on the dataset of [32] are shown in Fig. 10, where PSNR scores of various methods are shown over multiple iterations. The performance of the deep image prior [186, 187] gradually degrades after reaching a peak. For each image, the peak is reached at a different number of iterations, so simply using a fixed number of iterations will be suboptimal for most images.

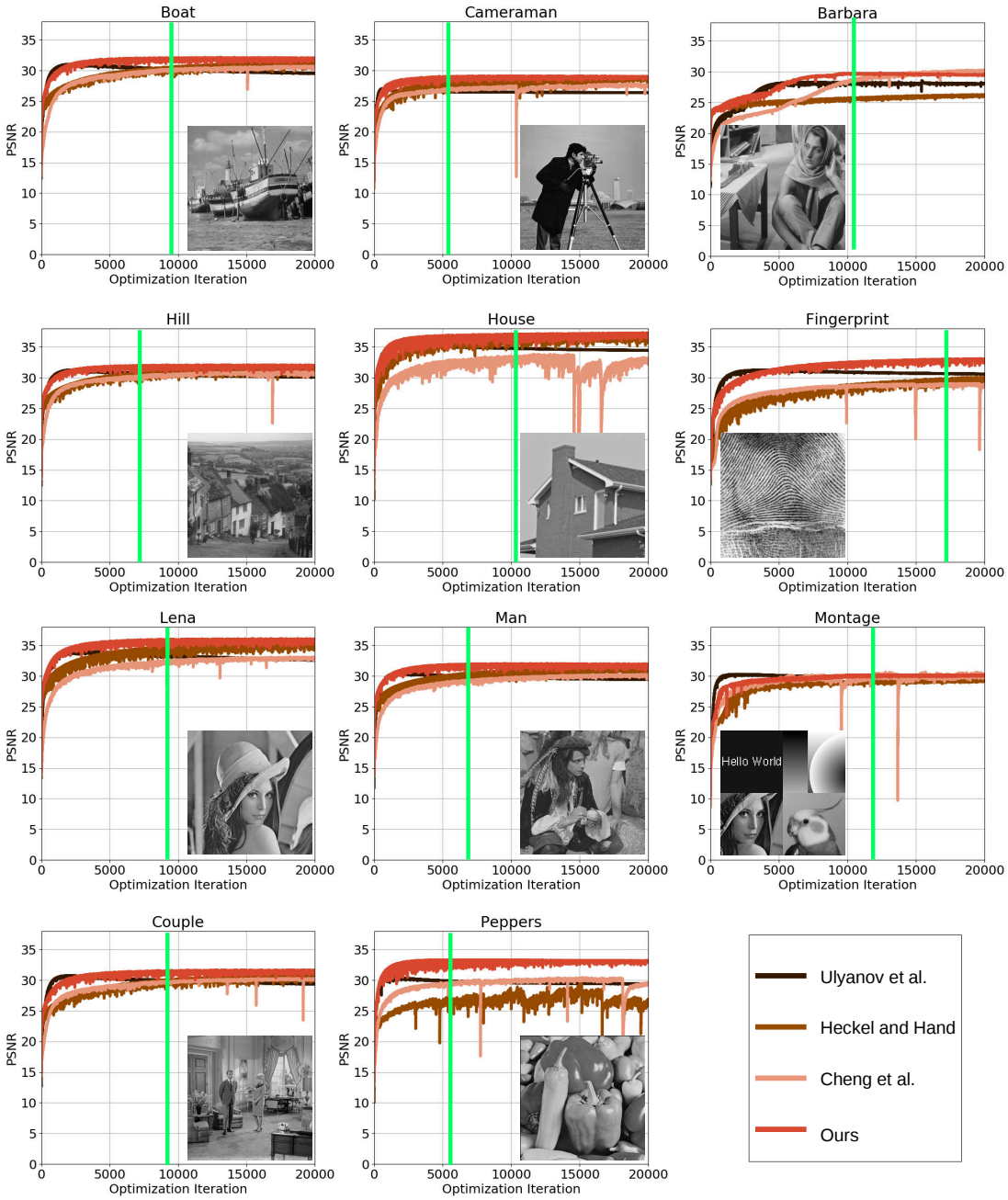
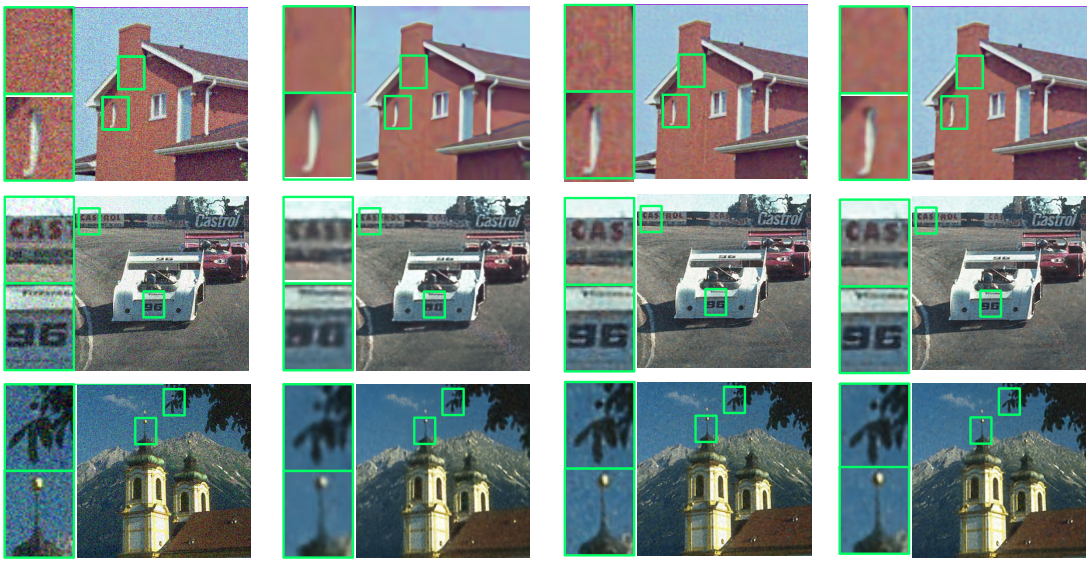


Figure 12: **Image inpainting.** PSNR scores of various methods for pixel inpainting. We again observe the degradation of performance over iterations for the deep image prior [187]. [30] and [63] do not suffer from the degradation problem, at the expense of either reduced performance or slow convergence. Our method achieves a good trade-off between PSNR score and convergence (marked by the green vertical lines).

Our method provides two advantages: 1) The performance does not decay over iterations with controlled spectral bias; 2) The optimization can be automatically stopped at an appropriate moment using the proposed stopping criterion, leading to good PSNR scores for all images (marked by the green vertical lines). [63] achieve fast convergence without performance degradation, but at the expense of reduced performance. [30] obtain comparable PSNR scores, but they require 2 to 4 times as many iterations to converge.



(a) **Noisy image** (b) **Heckel and Hand** (c) **Cheng *et al.*** (d) **Ours**

Figure 13: **Image denoising**. The goal is to remove the additive Gaussian white noise with σ=25. From the top regions masked by the green rectangles, we observe the method of [30] still overfits some high-frequency noise, while our method does not. From the bottom regions masked by the green rectangles, we observe the method of [63] has difficulty preserving high-frequency edges, while our method performs better.

Table 1: **Image denoising** on CBSD68 for varying σ. Supervised approaches and CBM3D prevail, but our unsupervised method obtains better PSNR than the deep image prior and its variants across three noise levels.

	15	25	50
Ulyanov *et al.* [187]†*	30.58	27.84	24.59
Heckel and Hand [63]†	28.66	26.60	24.06
Cheng *et al.* [30]†	30.77	28.08	24.71
Ours	**30.80**	**28.15**	**24.83**
CBM3D [32]††	33.52	30.71	27.38
CDnCNN [223]††	33.89	31.23	27.92
FFDNet [224]	33.87	31.21	27.96

†Results based on author-provided code.
*Results obtained with oracle stopping.
††Results provided by [224].

So far, we have shown the performance of various methods per image over a varying number of optimization iterations. Next, we compare their overall PSNR performance on the 68 images in CBSD68, as shown in Table 1. While our unsupervised method is outperformed by supervised-learning alternatives [223, 224] and CBM3D [32], it does better than the deep image prior [186, 187], and its variants [30, 63] across three noise levels. We also provide qualitative results for denoising in Fig. 13, where we observe our method preserves the high-frequency edges without overfitting to high-frequency noise.

(a) **Corrupted image** (b) **Cheng et al.** (c) **Ours** (d) **Ground truth**

Figure 14: **JPEG image deblocking**. The goal is to reduce the artifacts of the compressed JPEG image with *quality*=20. From the regions masked by the green rectangles, we observe our method performs well, especially when reducing the artifacts and recovering high-frequency image details.

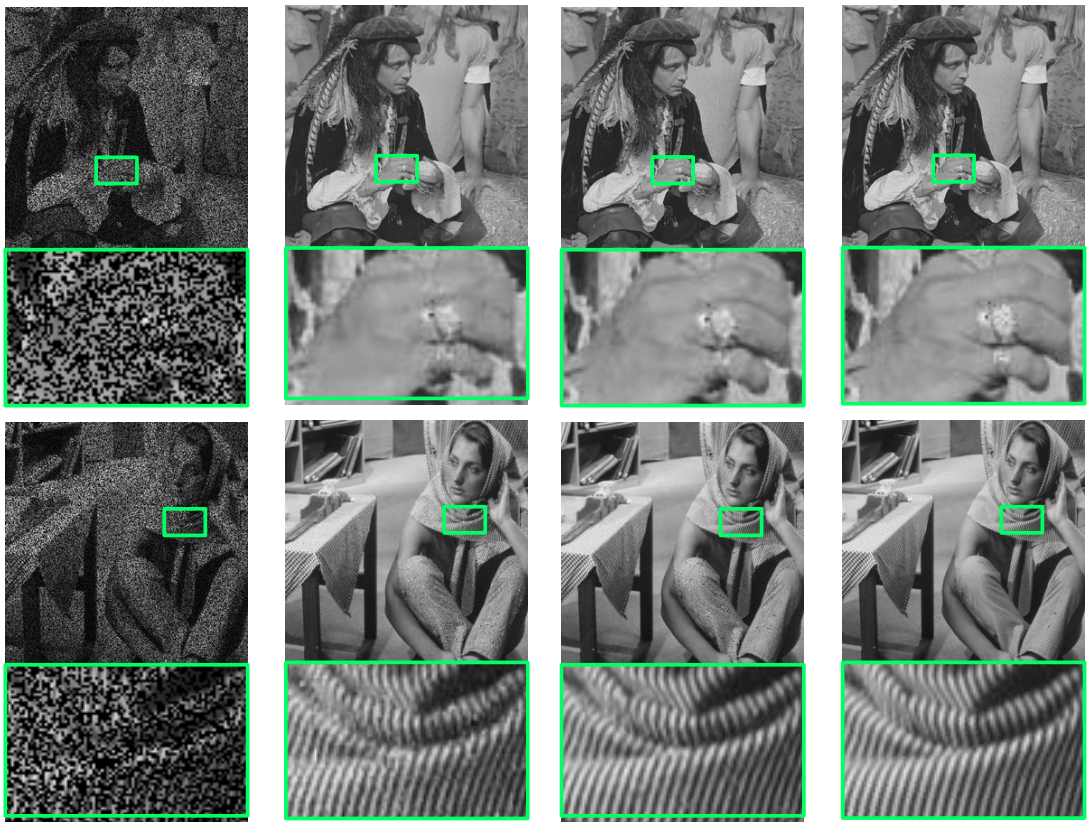
Table 2: **JPEG image deblocking** on LIVE1 for varying quality levels. Supervised approached prevail, but compared to the unsupervised deep image prior and its variants, our method obtains better performance in terms of PSNR.

	10	20	30
Ulyanov *et al.* [187]†*	27.52	29.75	31.08
Heckel and Hand [63]†	26.63	27.65	28.99
Cheng *et al.* [30]†	27.59	29.81	31.12
Ours	**27.70**	**29.86**	**31.14**
AR-CNN [39]††	28.96	31.29	32.67
TNRD [27]††	29.15	31.46	32.84
DnCNN [223]	29.19	31.59	32.98

†Results based on author-provided code.
*Results obtained with oracle stopping.
††Results provided by [223].

2.5.2 *JPEG image deblocking*

JPEG image deblocking is the process of reducing the compression artifacts in JPEG images. We evaluate on the Classic5 dataset by [48] and the LIVE1 dataset by [163]. Classic5 consists of 5 gray-scale images, and LIVE1 consists of 29 color images. Following [39], the color images are transformed to gray-scale using the YCbCr color model by



(a) **Corrupted image** (b) **Cheng et al.** (c) **Ours** (d) **Ground truth**

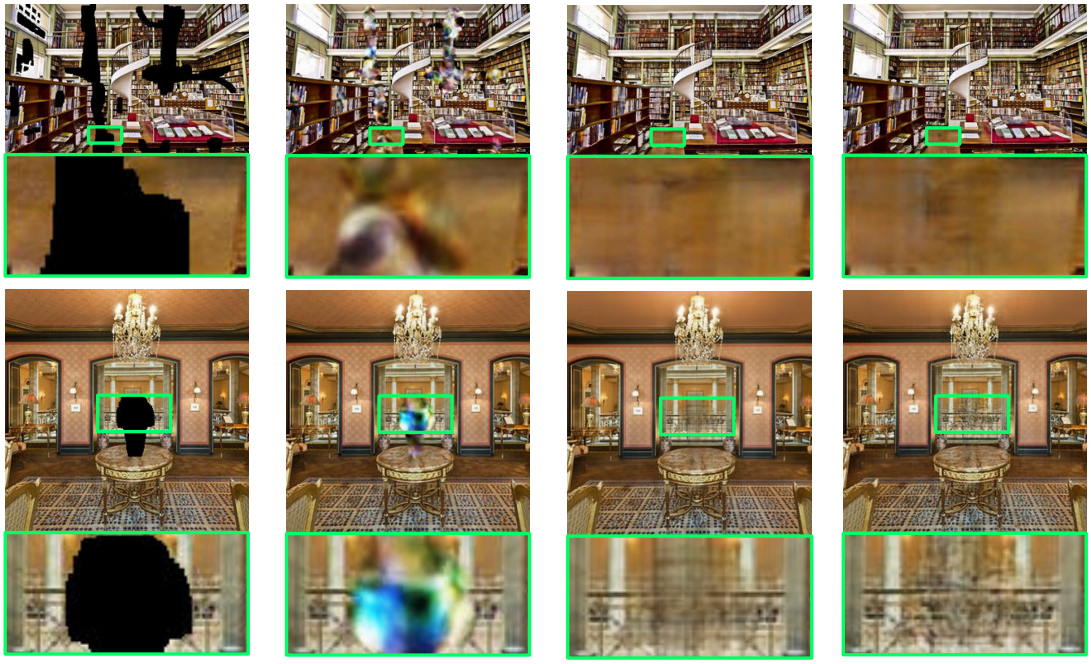
Figure 15: **Image inpainting**. The goal is to reconstruct the 50% missing pixels resulting from a binary Bernoulli mask. From the regions masked by the green rectangles, we observe our method performs well, especially when recovering high-frequency details.

Table 3: **Image inpainting** on CBSD68 for varying ratio. In terms of PSNR, our method outperforms the deep image prior and its variants on region-based inpainting, across three hole-to-image area ratios.

	0.1	0.25	0.5
Ulyanov *et al.* [187] †*	22.78	19.42	17.26
Heckel and Hand [63]†	21.52	18.67	16.81
Cheng *et al.* [30]†	22.83	19.49	17.28
Ours	**22.87**	**19.58**	**17.36**

†Results based on author-provided code.
*Results obtained with oracle stopping.

keeping the Y component only. Then, the gray-scale images are compressed with the PIL encoder into three qualities, 10, 20, and 30. Fig. 11 provides a quantitative comparison on Classic5 for *quality*=10. Akin to the denoising comparison, we again observe the degradation of performance over iterations for the deep image prior [186, 187]. [30] and [63] do not suffer from the degradation problem, at the expense of either reduced performance or slow convergence. With the controlled spectral bias and automatic stopping



(a) **Corrupted image** (b) **Heckel and Hand** (c) **Cheng et al.** (d) **Ours**

Figure 16: **Image inpainting**. The goal is to reconstruct the missing pixels resulting from a binary region mask. From the regions masked by the green rectangles, we observe our method performs better than [63] and as good as [30].

criterion, we achieve a good trade-off between PSNR score and convergence (marked by the green vertical lines).

We also provide quantitative results for LIVE1 in Table 2. Naturally, the learning-based methods [27, 39, 223] perform best. Across three quality levels, our unsupervised method performs better than the deep image prior [186, 187] and its two variants [30, 63]. We also provide qualitative examples in Fig. 14, which shows that our method better reduces the artifacts and recovers high-frequency image details.

2.5.3 *Image inpainting*

In image inpainting, we are given an image with missing pixels resulting from a binary mask. The goal is to reconstruct the missing data. We evaluate on the standard dataset by [65], consisting of 11 grayscale images, and the CBSD68 dataset by [156] consisting of 68 RGB images. Following [30, 186, 187], we consider inpainting with masks that are randomly sampled according to a binary Bernoulli distribution on the standard dataset. Each mask is sampled to drop 50% of the pixels at random. For CBSD68, we consider inpainting with central region masks and we evaluate on three hole-to-image area ratios, *ratio*=0.1, *ratio*=0.25 and *ratio*=0.5, following [145]. Fig. 12 provides a quantitative comparison on the standard dataset. We also provide quantitative results for CBSD68 in Table 3. Our observations are the same as for the denoising and deblocking comparison. We provide qualitative examples for pixel inpainting in Fig. 15 and region inpainting in Fig. 16, which shows our ability to recover high-frequency details.

(a) **Ulyanov et al.** (b) **Cheng et al.** (c) **Ours** (d) **Ground truth**

Figure 17: **Super-resolution**. Results on the 'baby' image and the 'flowers' image for 4× super-resolution, and on the 'butterfly' image for 8× super-resolution. From the regions masked by the green rectangles, we observe our method is able to better recover details with fewer artifacts (best viewed digitally).

2.5.4 *Super-resolution*

In image super-resolution, a low-resolution image is given; the goal is to recover its scaled-up version. Following [186, 187], the network generates a high-resolution image from the random noise input. The high-resolution image is then downsampled using a differentiable Lanczos filter to compute the loss with the provided low resolution image for optimizing the network. We report on the standard Set14 dataset by [219] and Set5 by [17]. We evaluate the performance for an up-scaling of 4 and 8. For the super-resolution task, the deep image prior [186, 187] does not suffer from the performance degradation over iterations because the optimization objective strives to find the low-resolution image without high-frequency noise. Following [186, 187], we report the PSNR score at a stopping iteration of 2,000 for the scaling of 4, and 4,000 for the scaling of 8. Results on Set 14 are provided in Table 4 and results on Set 5 are summarized in

(a) **Original image** (b) λ=1 (c) λ=2 (d) λ=3

Figure 18: **Image enhancement**. The goal is to enhance the image details. We obtain the smoothed images (second row) using the controlled deep image priors with different λ, as defined in Eq. (2.5). We then subtract the smoothed version from the original image to get fine details and enhance them (first row). The smaller the λ, the higher smoothness of the output images and the more enhancement to the image details.

Table 5. On most images our method achieves better performance, not only for [186, 187] but also compared to [63] and [30]. We provide a qualitative comparison in Fig. 17. We observe that our method produces fewer high-frequency artifacts than [186, 187] and [30]. We postulate that our Lipschitz normalization contributes to the benefit. Interestingly, our method also recovers fine details. A likely explanation is that our Gaussian upsampling is better at learning the desired higher frequencies. Note that fine details like textures are high-frequency compared to flat regions, but still relatively low-frequency compared to most artifacts.

2.5.5 *Image enhancement*

Following [186, 187], we also evaluate our method on image enhancement. The deep image prior performs sharpness enhancement by means of unsharp masking [140], which can be described by $x_e = (x_0 - x_s) + x_0$, where an enhanced image is represented by x_e, an original image by x_0, an unsharp mask by $(x_0 - x_s)$ where x_s denotes the smoothed version of the original image. The smoothness of x_s controls the size of the region around the edge pixels that is affected by sharpening. The higher the smoothness, the wider the regions around the edges that got sharpened. The deep image prior obtains the smoothed images by stopping the optimization at different iterations. However, the smoothness of

Table 4: **Super-resolution** on Set14. The PSNR scores are reported for a stopping iteration of 2,000 for the scaling of 4, and 4,000 for the scaling of 8, following [186, 187]. On most images we achieve better performance than existing methods, and we obtain the highest PSNR on average for 4× and 8× super-resolution.

	Baboon	Barbara	Bridge	Coastguard	Comic	Face	Flowers	Foreman	Lenna	Man	Monarch	Pepper	Ppt3	Zebra	*Average*
4× resolution															
Ulyanov *et al.* [187]*	22.29	25.53	24.38	25.81	22.18	31.02	26.14	31.66	30.83	26.09	29.98	32.08	24.38	25.71	27.00
Heckel and Hand [63]†	20.54	21.51	20.97	23.52	18.86	28.15	20.88	24.44	24.07	21.18	21.21	23.89	17.28	18.59	21.79
Cheng *et al.* [30]†	21.51	24.84	23.74	25.02	21.94	30.11	25.41	30.57	28.62	25.37	28.41	30.25	23.69	24.48	26.07
Ours	**22.81**	**25.74**	**25.02**	**25.86**	**22.31**	**32.09**	**26.74**	**32.77**	**31.29**	**26.42**	**30.77**	**32.62**	**24.73**	**25.87**	**27.50**
Bicubic††	22.44	25.15	24.47	25.53	21.59	31.34	25.33	29.45	29.84	25.70	27.45	30.63	21.78	24.01	26.05
TV prior††	22.34	24.78	24.46	25.78	21.95	31.34	25.91	30.63	29.76	25.94	28.46	31.32	22.75	24.52	26.42
[184]††	22.83	25.69	25.36	26.21	22.90	32.62	27.54	33.59	31.98	27.27	31.62	33.88	25.36	26.98	28.13
8× resolution															
Ulyanov *et al.* [187]*	21.38	23.94	22.20	**24.21**	19.86	29.52	22.86	27.87	**27.93**	23.57	**24.86**	**29.18**	20.12	**20.62**	24.15
Heckel and Hand [63]†	20.07	19.86	19.67	22.30	18.01	26.53	19.57	22.76	22.06	19.54	19.71	21.44	15.64	17.20	20.31
Cheng *et al.* [30]†	19.81	23.69	22.19	19.22	19.72	28.88	22.81	27.34	19.69	23.36	24.43	28.72	26.14	19.89	20.67
Ours	**21.57**	**24.48**	**22.64**	24.18	**19.71**	**29.94**	**22.92**	**27.95**	27.67	**23.86**	24.46	28.91	**23.28**	19.93	**24.17**
Bicubic††	21.28	23.44	22.24	23.65	19.25	28.79	22.06	25.37	26.27	23.06	23.18	26.55	18.62	19.59	23.09
TV prior††	21.30	23.72	22.30	23.82	19.50	28.84	22.50	26.07	26.74	23.53	23.71	27.56	19.34	19.89	23.48
[184]††	21.51	24.21	22.77	24.10	20.06	29.85	23.31	28.13	28.22	24.20	24.97	29.22	20.13	20.28	24.35

†Results based on author-provided code. *Results obtained with oracle stopping. ††Results provided by [187].

Table 5: **Super-resolution** on set5. The PSNR scores are reported for a stopping iteration of 2,000 for the scaling of 4, and 4,000 for the scaling of 8, following [187]. On most images we achieve better performance than existing methods, and we perform best on average for both 4× and 8× super-resolution.

	Baby	Bird	Butterfly	Head	Woman	*Average*
4× **resolution**						
Ulyanov *et al.* [187] *	31.49	31.80	26.23	31.04	**28.93**	29.89
Heckel and Hand [63]†	24.57	24.66	18.46	27.64	22.44	23.55
Cheng *et al.* [30]†	27.35	28.37	24.21	27.45	25.48	26.57
Ours	**32.76**	**32.71**	**26.47**	**31.79**	28.54	**30.45**
Bicubic ††	31.78	30.20	22.13	31.34	26.75	28.44
TV prior ††	31.21	30.43	24.38	31.34	26.93	28.85
LapSRN [184] ††	33.55	33.76	27.28	32.62	30.72	31.58
8× **resolution**						
Ulyanov *et al.* [187] *	28.28	**27.09**	20.02	29.55	24.50	25.88
Heckel and Hand [63]†	21.95	22.97	16.18	26.56	20.46	21.62
Cheng *et al.* [30]†	27.18	26.64	19.64	24.76	23.81	24.41
Ours	**28.46**	26.79	**20.32**	**30.07**	**24.76**	**26.08**
Bicubic ††	27.28	25.28	17.74	28.82	22.74	24.37
TV prior ††	27.93	25.82	18.40	28.87	23.36	24.87
LapSRN [184] ††	28.88	27.10	19.97	29.76	24.79	26.10

†Results based on author-provided code.
*Results obtained with oracle stopping.
††Results provided by [187].

the output image is quite sensitive to the number of optimization iterations, which is hard to control. By contrast, our method is able to manipulate the smoothness of the output image by tuning λ in Eq. (2.5). Thus, we obtain the smoothed images with different λ, by optimizing the network in a fixed iteration of $5,000$. The smaller the λ, the higher the smoothness of the output images and the more enhancement to the image details, as shown in Fig. 23.

2.5.6 *Success and failure cases*

We return to the denoising task to analyze a success and failure case or our approach in Fig. 19. The goal is to remove additive Gaussian noise from a natural image. Our method performs well when the noise level is modest, as shown in Fig. 19(b). However, with higher noise levels, the proposed method fails to remove the noise, as shown in Fig. 19(d). We attribute this to the fact that in the frequency domain, additive Gaussian noise has equal intensity at different frequencies. By contrast, the power spectrum of a natural image decays rapidly from low frequencies to high frequencies [157]. Consequently, when the noise level is low, noise is usually dominant at high frequencies and the natural signal is more dominant at lower frequencies. However, the noise can also be more dominant at lower frequencies with higher level. In this case, separating low-frequencies from high-frequencies through spectral bias fails to remove the noise.

(a) **Clean image** (b) $\sigma = 25$ (c) $\sigma = 50$ (d) $\sigma = 100$

Figure 19: **Success and failure case** of our method for image denoising on image 'F16'. Our method performs well when the noise level is modest (σ=25), while it fails to remove noise when the noise level is too high (σ=100).

2.6 CONCLUSION

In this chapter, we show the spectral bias leads inverse imaging networks to capture the deep image prior during optimization, independent of their architectures. We do so by introducing a metric, the Frequency Band Correspondence, which offers a spectral measurement of the deep image prior, revealing the low frequency natural image signals are learned faster and better than high-frequency noise signals. We also introduce Lipschitz normalization and Gaussian upsampling that allow to manipulate and adjust the spectral bias for inverse imaging problems. Besides these methods for controlling spectral bias, we further introduce a simple automatic stopping criterion to avoid superfluous computation. The experiments show that our method does not suffer from the performance degradation over iterations with controlled spectral bias and enables stopping the optimization automatically at an appropriate moment using the proposed stopping criterion. Our method also obtains favorable performance compared to current approaches for denoising, deblocking, inpainting, super-resolution and detail enhancement.

3

UNSHARP MASK GUIDED FILTERING

3.1 INTRODUCTION

Image filtering has been widely used to suppress unwanted signals (*e.g.* noise) while preserving the desired ones (*e.g.* edges) in image processing tasks like image restoration [7, 13, 44], boundary detection [77, 84, 120], texture segmentation [42, 152, 202], and image detail enhancement [8, 56, 136]. Standard filters, such as Gaussian filters and box mean filters, swiftly process input imagery but suffer from content-blindness, *i.e.*, they treat noise, texture, and structure identically. To mitigate content-blindness, guided filters [52, 60, 90, 112, 147, 204], have received a great amount of attention from the community. The key idea of guided filtering is to leverage an additional guidance image as a structure prior and transfer the structure of the guidance image to a target image. By doing so, it strives to preserve salient features, such as edges and corners, while suppressing noise. The goal of this chapter is guided image filtering.

Classical guided filtering, *e.g.* [60, 91, 100, 183], performs structure-transferring by relying on hand-designed functions. Nonetheless, it is known to suffer from halo artifacts and structure inconsistency problems (see Fig. 20), and it may require a considerable computational cost. In recent years, guided image filtering has advanced by deep convolutional neural networks. Both Li *et al.* [96] and Hui *et al.* [71] demonstrate the benefits of learning-based guided filtering over classical guided filtering. These works and their follow-ups, *e.g.* [1, 97, 182], directly predict the filtered output by means of feature fusion from the guidance and target images. Yet, this implicit way of structure-transferring may fail to transfer the desired edges and may suffer from transferring undesired content to the target image [97, 144].

Pan *et al.* [144] propose an alternative way to perform deep guided filtering. Rather than directly predicting the filtered image, they leverage a shared deep convolutional neural network to estimate the two coefficients of the original guided filtering formulation [60]. While their approach obtains impressive filtering results in a variety of applications, we observe their network has difficulty disentangling the representations of the two coefficients, resulting in halo artifacts and structure inconsistencies, see Fig. 20. Building on the work of Pan *et al.* [144], we propose a new guided filtering formulation, which depends on a *single* coefficient and is therefore more suitable to be solved by a single deep convolutional neural network.

We take inspiration from another classical structure-transferring filter: unsharp masking [35, 140, 148, 212]. From the original guided filter by He *et al.* [60] we first derive a simplified guided filtering formulation by eliminating one of its two coefficients. So there is only one coefficient left to be estimated for deciding how to perform edge

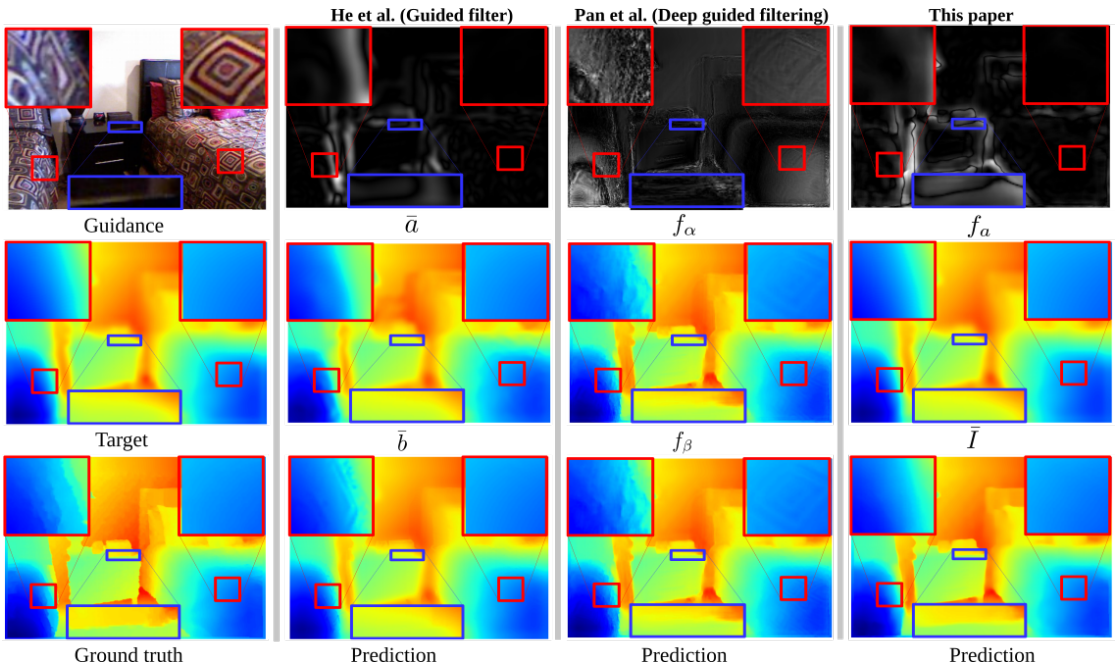


Figure 20: **Motivation of this chapter.** We show an example of depth upsampling (16×) using an RGB image as guidance. Both the conventional guided filter by He *et al.* [60] and the state-of-the-art deep guided filter by Pan *et al.* [144] explicitly estimate two coefficients, respectively $(\bar{a}, \bar{b})$ and (f_α, f_β). In their current formulation, however, both methods are likely to over-smooth edges (compare edges in blue boxes) and transfer unwanted textures (compare highlighted details in red boxes). Our proposed guided filter, taking inspiration from unsharp masking, only requires learning a single coefficient f_a (notation details provided in Sections 2 and 3). As a result, we obtain a more desirable upsampling result, free of undesirable structures and textures from the guidance image.

enhancement, akin to unsharp masking. To arrive at our formulation, we rely on the filtering prior used in unsharp masking and perform guided filtering on the unsharp masks rather than the raw target and guidance images themselves. The proposed formulation enables us to intuitively understand how the guided filter performs edge-preservation and structure-transferring, as there is only one coefficient in the formulation, rather than two in [144]. The coefficient explicitly controls how structures need to be transferred from guidance image to target image and we learn it adaptively by a single network. To that end, we introduce a successive guided filtering network. The network obtains multiple filtering results by training a single network. It allows a trade-off between accuracy and efficiency by choosing different filtering results during inference. This leads to fast convergence and improved performance. Experimental evaluation on seven datasets shows the effectiveness of our proposed filtering formulation and network on multiple applications, such as upsampling, denoising, and cross-modality filtering.

3.2 BACKGROUND AND RELATED WORK

In guided filtering, we are given an image pair (I, G), where the image pair has been properly registered by default. Image I needs to be filtered, *e.g.* due to the presence of noise or due to low resolution because it is captured by a cheap sensor. Guidance

image G contains less noise and is of high resolution with sharp edges, *e.g.* because it is captured by accurate sensors under good light conditions. The low-quality image I can be enhanced by filtering under the guidance of high-quality image G. Such a guided filtering process is defined as $\hat{I} = \mathcal{F}(I, G)$, where $\mathcal{F}(\cdot)$ denotes the filter function and $\hat{I}$ denotes the filtered output image. Below, we first review the benefits and weaknesses of classical guided filter functions and existing deep guided filter functions. We then present how our proposal can be an improved guided filtering solution by establishing a link to unsharp masking.

3.2.1 *Classical guided filtering*

The guided image filter [60] assumes that the filtered output image $\hat{I}$ is a linear transform of the guidance image G at a window w_k centered at pixel k:

$$\hat{I}_i = a_k G_i + b_k, \quad \forall i \in w_k, \tag{3.1}$$

where a_k and b_k are two constants in window w_k. Their values can be obtained by solving:

$$E(a_k, b_k) = \sum_{i \in w_k} ((a_k G_i + b_k - I_i)^2 + \epsilon a_k^2). \tag{3.2}$$

Here, ϵ is a regularization parameter penalizing large values for a_k. The optimal values of a_k and b_k are computed as:

$$a_k = \frac{\frac{1}{|w|}\sum_{i \in w_k} I_i G_i - \bar{I}_k \bar{G}_k}{\sigma_k^2 + \epsilon}, \tag{3.3}$$

$$b_k = \bar{I}_k - a_k \bar{G}_k. \tag{3.4}$$

Here, $\bar{G}_k$ and σ_k^2 are the mean and variance of G in w_k, $\bar{I}_k$ is the mean of I in w_k, and $|w|$ is the number of pixels in w_k. For ease of analysis, the guidance image G and filtering target image I are assumed to be the same [60], although the general case remains valid. As a result, we can obtain:

$$a_k = \sigma_k^2 / (\sigma_k^2 + \epsilon), \quad b_k = (1 - a_k)\bar{G}_k. \tag{3.5}$$

Based on this, the regions and edges with variance (σ_k^2) much larger than ϵ are preserved, whereas those with variance (σ_k^2) much smaller than ϵ are smoothed. Hence, ϵ takes control of the filtering. However, the value of ϵ in the guided image filter [60] is fixed. As such, halos are unavoidable when the filter is forced to smooth some edges [60, 91, 100, 183]. An edge-aware weighted guided image filter is proposed in [100] to overcome this problem, where ϵ varies spatially rather than being fixed:

$$\epsilon = \frac{\lambda}{\Gamma_G}, \quad \Gamma_G = \frac{\sigma_k^2 + \varepsilon}{\frac{1}{N}\sum_{k=1}^{N} \sigma_k^2 + \varepsilon}, \tag{3.6}$$

where λ and ε are two small constants and σ_k^2 is the variance of G in a 3×3 window centered at the pixel k. Γ_G measures the importance of a pixel k with respect to the whole guidance image. Kou *et al.* [91] propose a multi-scale edge-aware weighted guided image filter, in which Γ_G is computed by multiplying the variances of multiple windows. By taking the edge direction into consideration, Sun *et al.* [183] further improve the filter's performance. However, predefined parameters still remain in all these three methods.

Another limitation of classical guided image filters is their assumption that the target image and the guidance image have the same structure. In practice, there are also situations conceivable where an edge appears in one image, but not in the other. To address this issue, recent works [57, 59, 79, 164, 213] enhance guided image filters by considering the mutual structure information in both the target and the guidance images. These methods typically build on iterative algorithms that minimize global objective functions. The guidance signals are updated at each iteration to enforce the outputs to have similar structure as the target images. These global optimization methods generally use hand-crafted objectives that usually involve fidelity and regularization terms. The fidelity term captures the consistency between the filtering output and the target image. The regularization term, typically modeled using a weighted L2 norm [47], encourages the filtering output to have a similar structure as the guidance image. However, such hand-crafted regularizers may not transfer the desired structures from the guidance image to the filtering output [97, 144]. For this reason, we prefer a guided filtering solution based on end-to-end deep representation learning.

3.2.2 *Deep guided filtering*

Li *et al.* [96, 97] introduce the concept of deep guided image filtering based on an end-to-end trainable network. Two independent convolutional networks f_I and f_G first process the target image and guidance image separately. Then their outputs from the last layers are concatenated and forwarded to another convolutional network f_{IG} to generate the final output. We define the method as:

$$\hat{I} = f_{IG}\Big(f_I(I) \oplus f_G(G)\Big), \tag{3.7}$$

where $\oplus$ denotes the channel concatenation operator. Several works follow the same definition but vary in their feature fusion strategies [1, 71, 182]. AlBahar *et al.* [1] introduce a bi-directional feature transformer to replace the concatenation operation. Su *et al.* [182] propose a pixel-adaptive convolution to fuse the outputs of networks f_I and f_G adaptively on the pixel level. Hui *et al.* [71] propose to perform multi-scale guided depth upsampling based on spectral decomposition, where low-frequency components of the target images are directly upsampled by bicubic interpolation and the high-frequency components are upsampled by learning two convolutional networks. The high-frequency domain learning leads to an improved performance. Wu *et al.* [203] alternatively combine convolutional networks and traditional guided image filters. Two independent convolutional networks f_I and f_G first amend the target image and the guidance image, and then feed their outputs into the traditional guided image filter $\mathcal{F}_{IG}$ [60]:

$$\hat{I} = \mathcal{F}_{IG}\Big(f_I(I), f_G(G)\Big). \tag{3.8}$$

Rather than directly predicting the filtered image, Pan *et al.* [144] leverage deep neural networks to estimate the two coefficients of the original guided filtering formulation [60] based on a spatially-variant linear representation model, leading to impressive filtering results. It is defined as:

$$\hat{I} = f_\alpha(I,G) \odot G + f_\beta(I,G), \tag{3.9}$$

where f_α and f_β are two convolutional networks, and $\odot$ denotes element-wise multiplication. To reduce the complexity of learning, in the implementation Pan *et al.* [144] learn a single network and predict an output with two channels, one channel for f_α and another channel for f_β. However, we observe that the shared network has difficulty disentangling the representations of the two coefficients, resulting in halo artifacts and structure inconsistencies. Differently, we propose a new guided filtering formulation, which depends on a *single* coefficient only and is therefore more suitable to be solved by a single deep convolutional neural network.

The aforementioned existing deep guided filtering works implicitly perform structure-transferring by learning a joint filtering network, usually resulting in undesired filtering performance [37,129,144]. Recent works [36,37,126–130] take inspiration from coupled dictionary learning [179], and incorporate sparse priors into their deep networks for explicit structure-transferring. Marivani *et al.* [126–129] propose a learned multimodal convolutional sparse coding network with a deep unfolding method for explicitly fusing information from the target and guidance image modalities. Deng *et al.* propose a deep coupled ISTA network with a multimodal dictionary learning algorithm [36], and a common and unique information splitting network with multi-modal convolutional sparse coding [37], for the sake of explicitly modeling the knowledge from the guidance image modality. Most of these works focus on guided image super-resolution. In this chapter, we propose an explicit structure-transferring method for general guided filtering problems. In particular, we propose a guided filtering formulation with a single coefficient, and we learn to estimate the coefficient to explicitly decide how to transfer the desirable structures from guidance image to target image. As we will demonstrate, this leads to more desirable filtering results.

3.2.3 *Unsharp masking*

Our formulation is inspired by the classical sharpness enhancement technique of unsharp masking [35, 140, 148, 212], which can be described by the equation:

$$\hat{I} = \lambda(I - \mathcal{F}_L(I)) + I, \tag{3.10}$$

where an enhanced image is represented by $\hat{I}$, an original image by I, an unsharp mask by $(I - \mathcal{F}_L(I))$ where $\mathcal{F}_L$ denotes a low-pass filter like Gaussian filters or box mean filters, and an amount coefficient by λ which controls the volume of enhancement achieved at the output. Essentially, guided filtering shares the same function of edges enhancement as unsharp masking by means of the structure-transferring from an additional guidance image. Based on this viewpoint, we derive a simplified guided filtering formulation from the original guided filter [60], with only one coefficient to be estimated, akin to the formulation of unsharp masking in Eq. (3.10).

3.3 FILTERING FORMULATION

Here, we outline our new guided filtering formulation, in which only one coefficient needs to be estimated. Compared to estimating two coefficients (a, b) as in the original guided filtering formulation and subsequent deep learning variants, our formulation is more suitable to be solved with one single deep network. We start the derivation of our guided filtering formulation from the classical guided filter [60], summarized in Eq. (3.1), Eq. (3.3) and Eq. (3.4). In Eq. (3.1), $\hat{I}$ is a linear transform of G in a window w_k centered at the pixel k. When we apply the linear model to all local windows in the entire image, a pixel i is involved in all the overlapping windows w_k that covers i. In this case, the value of $\hat{I}_i$ in Eq. (3.1) is not identical when it is computed in different windows. So after computing (a_k, b_k) for all windows w_k in the image, we compute the filtered output image $\hat{I}_i$ by averaging all the possible values of $\hat{I}_i$ with:

$$\hat{I}_i = \frac{1}{|w|} \sum_{k \in w_i} (a_k G_i + b_k). \tag{3.11}$$

Similar in spirit to unsharp masking, summarized in Eq. (3.10), we want to maintain only the coefficient a to control the volume of structure to be transferred from guidance G to the filtered output image $\hat{I}$. Thus, we put Eq. (3.4) into Eq. (3.11) to eliminate b, and obtain:

$$\hat{I}_i = \frac{1}{|w|} \sum_{k \in w_i} a_k G_i + \frac{1}{|w|} \sum_{k \in w_i} (\bar{I}_k - a_k \bar{G}_k). \tag{3.12}$$

Next, we rewrite the formulation as:

$$\hat{I}_i = \frac{1}{|w|} \sum_{k \in w_i} a_k (G_i - \bar{G}_k) + \widetilde{I}_i, \tag{3.13}$$

where $\widetilde{I}_i = \frac{1}{|w|} \sum_{k \in w_i} \bar{I}_k$. Since $\bar{G}_k$ is the output of a mean filter, it's assumed that $\bar{G}_k$ is close to its mean in the window w_i. Next we rewrite Eq. (3.13) as follows

$$\hat{I}_i = \bar{a}_i (G_i - \widetilde{G}_i) + \widetilde{I}_i, \tag{3.14}$$

where $\bar{a}_i = \frac{1}{|w|} \sum_{k \in w_i} a_k$, and $\widetilde{G}_i = \frac{1}{|w|} \sum_{k \in w_i} \bar{G}_k$. For convenience, we will omit subscript i in the following.

The formulation in Eq. (3.14) enables us to intuitively understand how the guided filter performs edge-preservation and structure-transferring. Specifically, the target image I is first smoothed to remove unwanted components like noise/textures, and the smoothing result is denoted by $\widetilde{I}$. However, the smoothing process usually suffers from the loss of sharp edges, leading to a blurred output. To enhance the edges, an unsharp mask $(G - \widetilde{G})$ with fine edges generated from the guidance image G is added to $\widetilde{I}$ under the control of the coefficient a, leading to the structure being transferred from the guidance image to the filtered output image $\hat{I}$. Finally, we rewrite Eq. (3.14) to obtain a more general formulation for deep guided filtering:

$$\hat{I} = f_a(I_m, G_m) \odot G_m + \mathcal{F}_L(I), \tag{3.15}$$

where $I_m = I - \mathcal{F}_L(I)$ and $G_m = G - \mathcal{F}_L(G)$ denote the unsharp masks of the target image and the guidance image, which contain the structures of the guidance and the target images. $\mathcal{F}_L$ denotes a linear shift-invariant low-pass filter like the Gaussian filter or the box mean filter. f_a denotes the amount function, which controls the volume of structure to be transferred from the guidance image to the filtered output image. Next, we will elaborate on this function.

Amount function f_a. The output of f_a is the volume of the structure to be transferred from the guidance image to the filtered output image. Thus, the input of f_a should involve the structure of both the target and the guidance image, which together determine the output, *i.e.*, $f_a(I_m, G_m)$. Ideally, f_a should determine the structure-transferring in a pixel-adaptive fashion. It can be a manually designed function, as the function a of the guided filter in Eq. (3.3). It also can be estimated by learning a deep neural network. Compared to hand-crafted functions, learnable functions are more flexible and allow for a better generalization to various image conditions.

Successive filtering. Successive operations of the same filter generally result in a more desirable output, thus we develop a successive guided filtering based on our formulation in Eq. (3.15). Instead of directly iterating the filtering output $\hat{I}$, we iterate the outputs of f_a as the function decides the effect of filtering. Let $f_a^\star$ be a composition of a set of basic functions $\{f_a^{(l)}\}_{l=1}^L$:

$$f_a^\star = f_a^{(L)} \circ f_a^{(L-1)} \circ \cdots \circ f_a^{(1)} \tag{3.16}$$

in which $\circ$ denotes the function composition operation, such as $\big(f \circ u\big)(\cdot) = f(u(\cdot))$. With $f_a^\star$ we obtain a successive filtering formulation from Eq. (3.15),

$$\hat{I} = f_a^\star(I_m, G_m) \odot G_m + \mathcal{F}_L(I). \tag{3.17}$$

In the next section we will detail how to implement our filters with deep convolutional neural networks.

3.4 FILTERING NETWORK

There is a function f_a in our formulation, which governs a guided filtering coefficient. We propose to solve this function with a single convolutional neural network.

Network for amount function f_a. The function f_a decides how to transfer the structure of the guidance image to the filtered output image. There are two inputs G_m and I_m for this function. Two options are available for the network architecture. Like [71,97], we can separately process these two inputs with two different sub-networks at first, and then fuse their outputs with another sub-network. Alternatively, we can concatenate these two inputs together and forward the concatenation into a single network, similar to the framework of [144]. Empirically, we find that the second option is easily implemented and achieves a desirable filtering accuracy and efficiency. Thus, we design the network of f_a with the second option in this chapter.

In our approach the unsharp masks, rather than the raw images themselves, are used as the inputs of the network. The unsharp mask is more sparse than the raw image itself, since most regions in the unsharp mask are close to zero. The learning of spatially-sparse data usually requires a large receptive field. The dilated convolution is a popular

technique to enlarge the receptive fields in convolutional networks [25, 215]. We design our network by cascading four dilated convolutional layers with increasing dilation factors, where all the dilated convolutional layers have the same channel number of 24 and the same kernel size of 3×3. Their dilation factors are set to $\{1, 2, 4, 8\}$. Leaky ReLU with a negative slope of 0.1 is used as an activation function after each dilated convolutional layer. Finally, a 1×1 convolution layer generates the target output for f_a.

Network for successive filtering. To develop a network for the successive filtering formulation in Eq. (3.17), we consider the network of the basic functions $\{f_a^{(l)}\}_{l=1}^{L}$ as a convolutional block, as shown in Fig. 21 (a). Then stacking this block multiple times results in a deep network for $f_a^{\star}$. There are two outputs in the block. By concatenating its input and its feature maps from the last layer results in the first output. The concatenation output allows feeding the previous multi-level outputs to the following blocks, leading to improved performance. We develop the second output by using a convolutional layer on top of the last layer of the block, for the sake of balancing accuracy and efficiency.

Stacking more blocks results in higher accuracy at the expense of an increased computational complexity. To allow users to choose between accuracy and computational complexity, we generate filtering outputs from each block. If users want to obtain filtering results fast, the filtering results from the first blocks can be used. When the accuracy is more important, the filtering results from the later blocks can be used. In short, we obtain multiple filtering results while training one single network. The overall network architecture is visualized in Fig. 21 (b).

Optimization. During training, we are given N samples $\{(I_n, G_n, Z_n)\}_{n=1}^{N}$, with $I_n \in \mathcal{I}$ the target image, $G_n \in \mathcal{G}$ the guidance image and $Z_n \in \mathcal{Z}$ the task-dependent ground-truth output image. Our goal is to learn the parameters θ of the network for f_a. Two types of loss, L_1 loss and L_2 loss, have been widely used in deep guided filtering works. Early works, like [71, 96], have adopted a L_2 loss, while recent works, like [1, 144], prefer a L_1 loss because it is less sensitive to outliers and leads to less blurry results compared to a L_2 loss [15, 75]. Following these recent works, we minimize the difference between filtered output image $\hat{I}$ and its ground-truth Z using the L_1 loss, which is defined as:

$$\mathcal{L}(I, G, Z; \theta) = \frac{1}{N} \sum_{n=1}^{N} \| \hat{I}_n(I_n, G_n; \theta) - Z_n \|_1 \ . \tag{3.18}$$

3.5 EXPERIMENTS

In this section, we provide extensive experimental evaluations. Section 3.5.1 introduces the experimental setup. Sections 3.5.2, 3.5.3, 3.5.4, and 3.5.5 emphasize ablations, comparisons and analysis. Sections 3.5.6, 3.5.7, and 3.5.8 show further qualitative and quantitative results, and state-of-the-art comparisons on various applications, including upsampling, denoising, and cross-modality filtering.

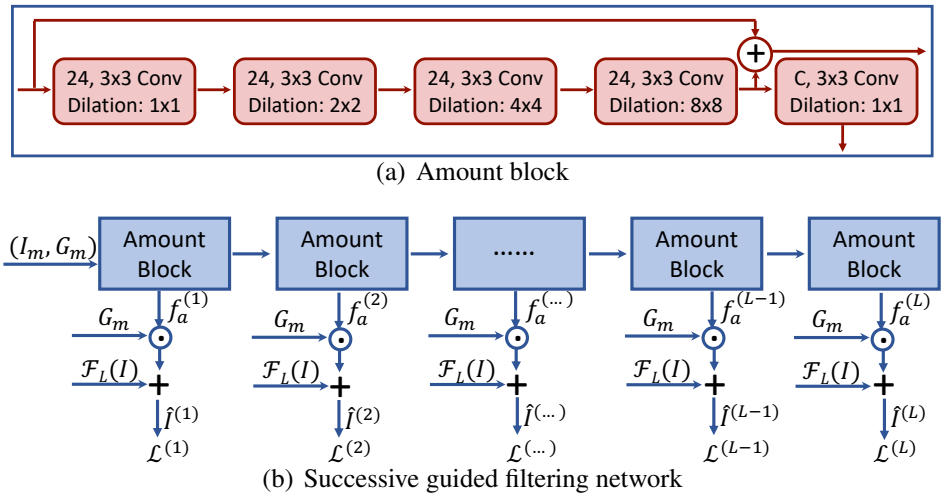


Figure 21: **Network architecture for unsharp-mask guided filtering**. a) Dilated convolutional block for amount function f_a; b) Network architecture for successive unsharp-mask guided filtering. Here, $\odot$ denotes the element-wise product, $\oplus$ denotes the concatenation operation, $+$ denotes the element-wise sum. Leaky ReLU is used as activation function after each convolutional layer. There are $(L-4)$ amount blocks in the box, indicated with dots. $\mathcal{L}$ denotes the loss function in Eq. (3.18). Here, G_m and $\mathcal{F}_L(I)$ are shared by $\{f_a^{(l)}\}_{l=1}^{L}$.

3.5.1 *Experimental setup*

Image upsampling datasets. We perform upsampling experiments on NYU Depth V2 [173], and Sintel optical flow [19]. For NYU Depth V2 we follow [97]. We use the first 1000 RGB/depth pairs for training and the remaining 449 pairs for testing, where each low-resolution depth image is generated by applying a nearest-neighbor interpolation. For Sintel, following [182], 908 samples, and 133 samples from clean pass are used for training, and testing, where each low-resolution flow image is generated by applying a bilinear interpolation.

Image denoising datasets. We perform denoising experiments also on NYU Depth V2 [173], as well as on BSDS500 [132]. For NYU Depth V2, we use the same split as for upsampling. BSDS500 contains 500 natural images. We train on the training set (200 images) and the validation set (100 images). We evaluate on the provided test set (200 images). Following [223], to train a blind Gaussian denoiser, random Gaussian noise is added to the clean training depth images in NYU Depth V2 and the clean RGB images in BSDS500, with a noise level $\sigma \in [0, 55]$. For testing, we consider three noise levels, $\sigma = \{15, 25, 50\}$. Thus, three separate noisy test images are generated for each original test image.

Pre-processing. For all datasets, we normalize the input images by scaling their pixel values to the range $[0, 1]$. According to Eq. (3.15), the guidance image G and target image I should have the same number of channels. In the depth/RGB dataset, the target is the 1-channel depth image. Thus, rgb2gray operation is used to transform a 3-channel RGB image into a 1-channel grayscale image as guidance. During training, we augment

the images by randomly cropping 256×256 patches. No cropping is performed during testing.

Network and optimization. The proposed network is optimized in an end-to-end manner. We implement the network with TensorFlow on a machine with a single GTX 1080 Ti GPU. The optimizer is Adam with a mini-batch of 1. We set β_1 to 0.9, β_2 to 0.999, and the initial learning rate to 0.0001. Optimization is terminated after 1000 epochs.

Evaluation metrics. To evaluate the quality of the predicted images we report four standard metrics: RMSE (Root Mean Square Error) for depth upsampling, EPE (End-Point-Error) for flow upsampling, PSNR (Peak Signal-to-Noise Ratio) for denoising, and SSIM (Structural Similarity Index Measure) for all applications.

3.5.2 *Unsharp-mask guided filtering without learning.*

The goal of the first experiment is to validate our formulation as a valid guided filter. Here we do not rely on any deep learning for estimating f_a. Instead, we use the function a of the guided filter [60] and the weighted guided filter (WGF) [100] as f_a. $\mathcal{F}_L(G)$ and $\mathcal{F}_L(I)$ are generated by cascaded box mean filters. Using our formulation as a conventional guided filter, we provide qualitative and quantitative results to demonstrate that our filter performs as good as, or even better than the guided filter [60] and the weighted guided filter [100].

Qualitative results. The first example performs edge-preserving smoothing on a gray-scale image. The second example is about detail enhancement. For both, the target image and guided image are identical. Fig. 22 and Fig. 23 show the results of filtering, where we can see that our filter performs as good as the guided filter [60] in preserving structures. In the third example, we denoise a no-flash image under the guidance of its flash version to verify the effect of structure-transferring. The denoising results of our filter and the guided filter [60] in Fig. 24 are consistent and don't have gradient reversal artifacts.

Quantitative results. Next, we compare our filter with the guided filter (GF) [60] and the weighted guided filter (WGF) [100] for natural image denoising on BSDS500 and depth upsampling on NYU Depth V2. There are two hyperparameters, r and ϵ in GF and WGF. Grid-search is used to find the optimal hyperparameters. The results in Table 6 show that our filter performs at least as good as the guided filter [60] and the weighted guided filter [100], indicating that our formulation makes sense as a guided filter.

3.5.3 *Unsharp-mask guided filtering with learning*

Next, we assess the benefit of our formulation when f_a is learned by a neural network. We compare to four baselines: (*i*) DMSG [71], (*ii*) DGF [203] (*iii*) DJF [97], and (*iv*) SVLRM [144]. The experiments are performed on NYU Depth V2 [173] for depth upsampling (16×) and depth denoising ($\sigma = 50$). We compare these baselines separately. For each comparison, the network for f_a is the same as the network used in the compared method. We use a box mean filter with radius $r = 8$ to obtain $\mathcal{F}_L(I)$ and $\mathcal{F}_L(G)$. For depth upsampling (16×), we first upsample the low-resolution depth image by bicubic

(a) **Input** $I(G)$ (b) G_m (c) $f_a(I_m, G_m)$

(d) $\mathcal{F}_L(I)$ (e) **GF [60]** (f) ***Ours***

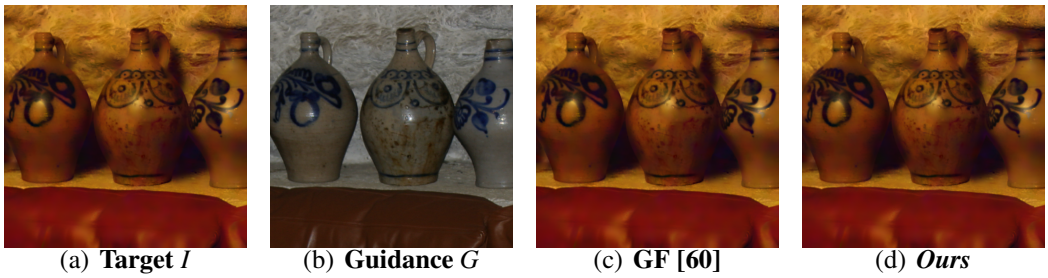
Figure 22: **Edge-preserving filtering.** a) Target and guidance image $I = G$; b) G_m containing the important structures. c) $f_a(I_m, G_m)$ estimated by Eq. (3.3) with $\epsilon = 0.05^2$; d) $\mathcal{F}_L(I)$ obtained by a cascade of two box filters with radius $r = 8$; e) The smoothing result obtained by the guided filter [60]; f) Our smoothing result. Both our filter and guided filter [60] can preserve good edges while removing noise.

interpolation to obtain the target resolution for $\mathcal{F}_L(I)$. We also use the upsampled depth image as the input of the network, following [97, 144, 203]. One exception is the comparison with DMSG [71] which uses the original low-resolution depth image as the input of the network.

Comparison with DMSG [71]. Fig. 25 (a) demonstrates our approach achieves better upsampling results than DMSG [71] in terms of RMSE. DMSG performs depth upsampling based on spectral decomposition. Specifically, a low-resolution depth image is first decomposed into low-frequency components and high-frequency components. The low-frequency components are directly upsampled by bicubic interpolation. The high-frequency components are upsampled by learning two convolutional networks. The first network is used to obtain multi-scale guidance. The other one performs multi-scale joint upsampling. The difference between our method and DMSG mainly lies in two

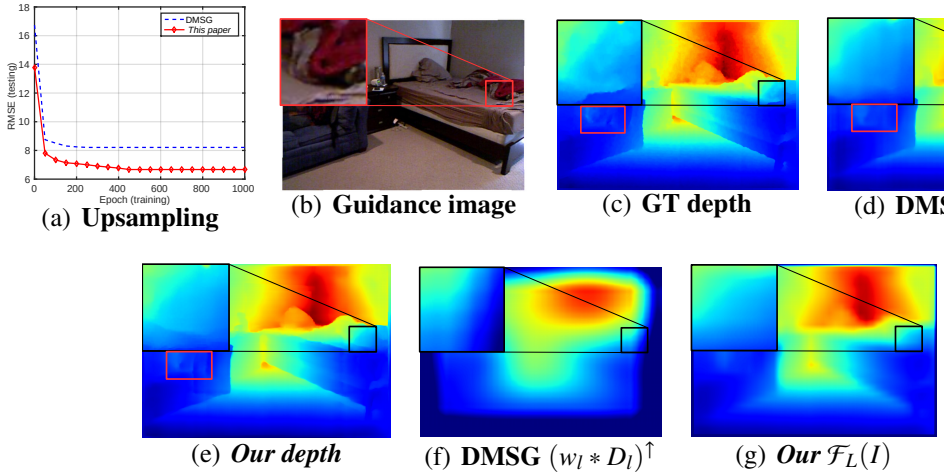
(a) **Target** I (b) **GF [60]** (c) ***Ours***

Figure 23: **Detail enhancement.** The parameters are $r = 16$, $\epsilon = 0.1^2$. Our filter without learning, as defined in Eq. (3.14), performs as good as the guided filter [60].



(a) **Target** I (b) **Guidance** G (c) **GF [60]** (d) ***Ours***

Figure 24: **Flash/no-flash denoising.** The parameters are $r = 8$, $\epsilon = 0.2^2$. Our filter without learning, as defined in Eq. (3.14), performs as good as the guided filter [60].



(a) **Upsampling** (b) **Guidance image** (c) **GT depth** (d) **DMSG depth**

(e) ***Our depth*** (f) **DMSG** $(w_l * D_l)^{\uparrow}$ (g) ***Our*** $\mathcal{F}_L(I)$

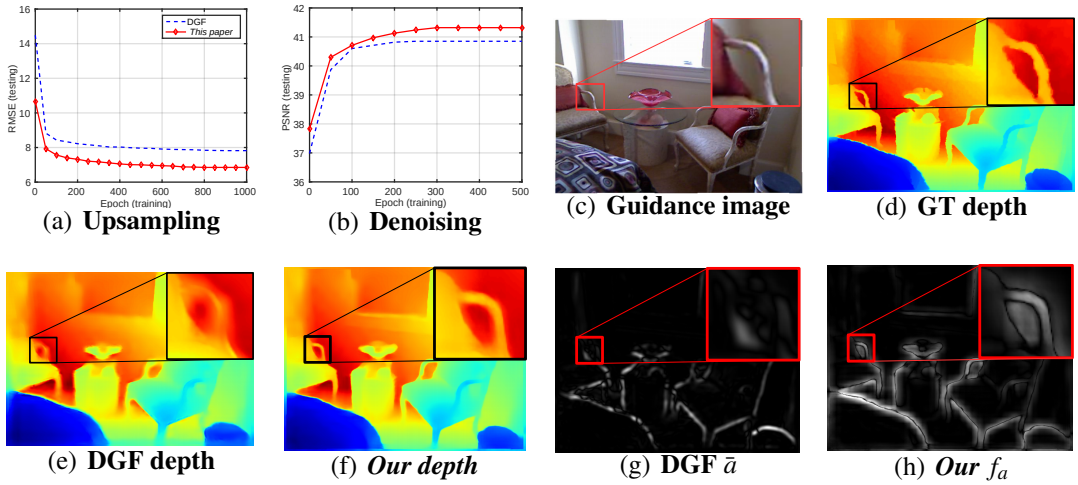
Figure 25: **Comparison with DMSG [71].** Compared to DGF, our approach better recovers finer edges as shown in the regions marked by the red boxes, and avoids producing artifacts as shown in the regions marked by the black boxes.

aspects. First, we don't use the first network and just use the second network for amount function f_a to explicitly perform structure transfer instead of directly predicting the

Table 6: **Quantitative results** for image denoising on BSDS500 and depth upsampling on NYU Depth V2. When the functions a of the guided filter [60] and the weighted guided filter [100] are used for f_a, our filter is denoted by "Ours + GF" and "Ours + WGF", respectively. Our filters perform at least as good as the baselines.

	Denoising (PSNR) ↑			**Upsampling (RMSE)** ↓		
	$\sigma = 15$	$\sigma = 25$	$\sigma = 50$	4×	8×	16×
Bicubic/Input	24.61	20.17	14.15	8.21	14.03	22.48
GF [60]	29.16	26.47	23.82	7.25	12.38	19.86
Ours + GF	29.24	26.59	23.84	7.18	**12.28**	**19.75**
WGF [100]	**29.40**	26.92	23.98	**7.17**	12.33	19.79
Ours + WGF	29.35	**26.96**	**23.99**	7.18	12.30	19.76

filtered output image. We find that our approach avoids halo effects more successfully, as shown in Fig. 25 (d) and (e). Second, Hui *et al.* use a Gaussian filter to smooth the low-resolution target depth image when generating its low-frequency components. After that, they upsample the low-frequency components by bicubic interpolation. However, this step is likely to produce artifacts, as shown in Fig. 25 (f). Since the network learning focuses on upsampling high-frequency components, the artifacts still remain in the final upsampling output, as shown in Fig. 25 (d). By contrast, we first upsample the low-resolution target depth image before smoothing. By doing so, the artifacts generated by bicubic interpolation can be removed by smoothing, as shown in Fig. 25 (g).

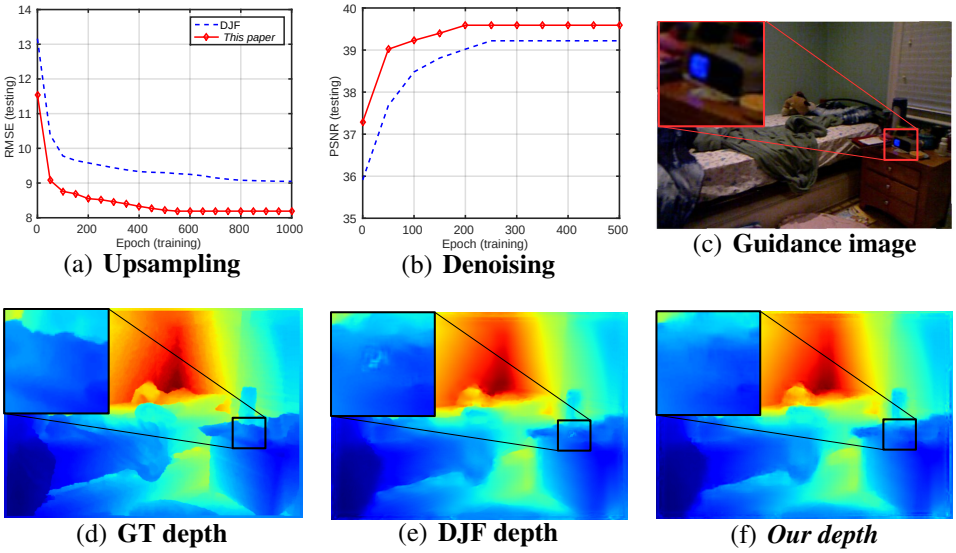


(a) **Upsampling** (b) **Denoising** (c) **Guidance image** (d) **GT depth**

(e) **DGF depth** (f) ***Our depth*** (g) **DGF** $\bar{a}$ (h) ***Our*** f_a

Figure 26: **Comparison with DGF [203].** The parameters of the guided filter [60] used in DGF are $r = 4$, $\epsilon = 0.1^2$. Our learned amount function performs better on structure-transferring than the manually designed one as shown in the region marked by red boxes of (g) and (e). Thus, our filter reduces the over-smoothing of important edges as shown in the region marked by black boxes of (e) and (f).

Comparison with DGF [203]. Wu *et al.* [203] learn two networks to amend the guidance and target images before feeding them to the guided filter [60]. The learned guidance and target images fit the guided filter [60] better than the original ones. However, DGF still suffers from the halo problem since its final filtering output is generated by the guided filter [60]. Our approach performs better than DGF [203] for both upsampling and



(a) **Upsampling** (b) **Denoising** (c) **Guidance image**

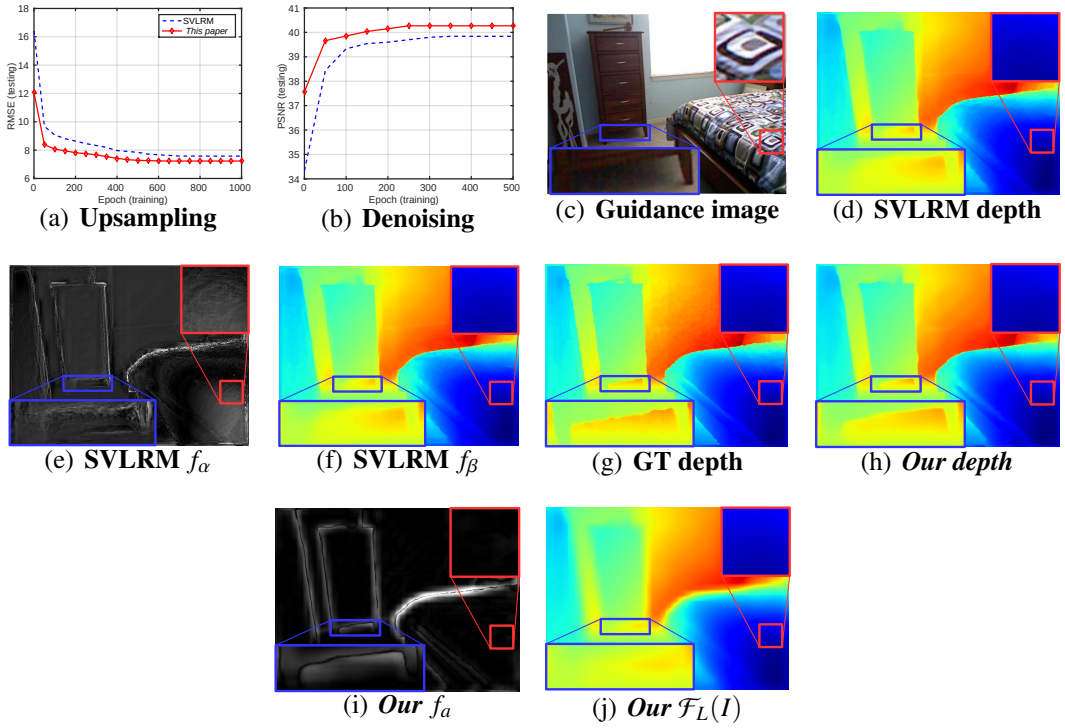
(d) **GT depth** (e) **DJF depth** (f) ***Our depth***

Figure 27: **Comparison with DJF [97].** Our approach avoids unwanted structures transferred from the guidance image to the target image as shown in the regions marked by the black boxes, leading to more desirable filtering results than DJF.

denoising tasks, as demonstrated in Fig. 26 (a) and (b). As shown in Fig. 26 (e) and (g), the important edges are unavoidable to be smoothed because the structure-transferring is performed in an undesirable fashion. By contrast, our approach performs better on preserving and transferring important structures, as shown in 26 (f) and (h), as the amount function f_a is learned in a pixel-adaptive way through a deep neural network instead of designed manually.

Comparison with DJF [97]. The network in our method directly uses the unsharp masks of the target image and the guided image as inputs, and learns to estimate the amount function f_a for explicitly deciding how to transfer the desired structure from the guidance image to the target image. By contrast, the network in DJF [97] uses the original guidance image and target image as inputs, and directly predicts filtered output relying on feature fusion. The implicit structure transfer is likely to cause slow convergence and the unwanted contents to be transferred from guidance image to target image, as shown in Fig. 27. From Fig. 27 (a) and (b), we can see our approach convergences faster and achieves better filtering performance than DJF [97] on both upsampling and denoising tasks.

Comparison with SVLRM [144]. Lastly, we compare to the state-of-the-art in deep guided filtering, namely SVLRM [144]. Here, we analyze two drawbacks of SVLRM [144], as illustrated in Fig. 28 (c-e). First, $f_\alpha(I,G)$ and $f_\beta(I,G)$ is likely to learn similar structure information. This is because they share the same training dependencies; such as input, network architecture and objective function. As a result, $f_\alpha(I,G)$ can't transfer the desired structure from guidance G to the output image of $f_\beta(I,G)$. Second, SVLRM behaves like DJF [97] when the filtering performance is determined by $f_\beta(I,G)$. The implicit joint filtering causes slow convergence and the unwanted structures are transferred. By contrast, our approach focuses on estimating the amount function f_a for explicit structure transfer, leading to more desirable filtering results, as illustrated in



(a) **Upsampling** (b) **Denoising** (c) **Guidance image** (d) **SVLRM depth**

(e) **SVLRM** f_α (f) **SVLRM** f_β (g) **GT depth** (h) ***Our depth***

(i) ***Our*** f_a (j) ***Our*** $\mathcal{F}_L(I)$

Figure 28: **Comparison with SVLRM [144].** In SVLRM, f_α and f_β are likely to learn the same structure information. In this case, f_α can't transfer the structure desired by f_β, resulting from over-smoothing of edges, as shown in the regions marked by the blue boxes of (d) and (e). When the filtering output is determined by f_β, SVLRM behaves like DJF [97], causing the transfer of unwanted contents from guidance image to target image as shown in the regions marked by the red boxes of (c-e). By contrast, our approach resolve the problems of SVLRM by explicitly learning structure-transferring, leading to more desirable filtering results as shown in regions marked by the blue and red boxes of (h-j).

Fig. 28 (h-j). Fig. 28 (a) and (b) demonstrate the better performance of our approach compared to SVLRM [144], for both upsampling and denoising.

Amount function f_a. When we estimate the amount function f_a through a convolutional neural network, the network architecture plays an important role in filtering performance. We have compared our filtering formulation with several baselines when f_a is estimated by different networks used in these baselines. Generally, we found deep networks perform better than shallow networks, *e.g.* the network of SVLRM with a depth of 12 achieves an RMSE of 7.23 for upsampling (16×) and a PSNR of 40.27 for denoising ($\sigma = 50$), better than the RMSE of 8.19 and the PSNR of 39.59 obtained by the network of DJF with a depth of 6. One explanation for this is the fact that the deep network has more ability to express complex functions than shallow ones.

In Eq.(3.15), we use the unsharp masks of the target image and guidance image as the input of the amount function network f_a. The raw target image and guidance image can also be the input. Next, we perform an experiment to study which input performs better. The network of DJF [97] is used for f_a. On NYU Depth V2, using the unsharp mask as input achieves an RMSE of 8.19 for upsampling (16×) and a PSNR of 39.59 for denoising ($\sigma = 50$), better than the RMSE of 8.64 and the PSNR of 39.31 obtained by using the

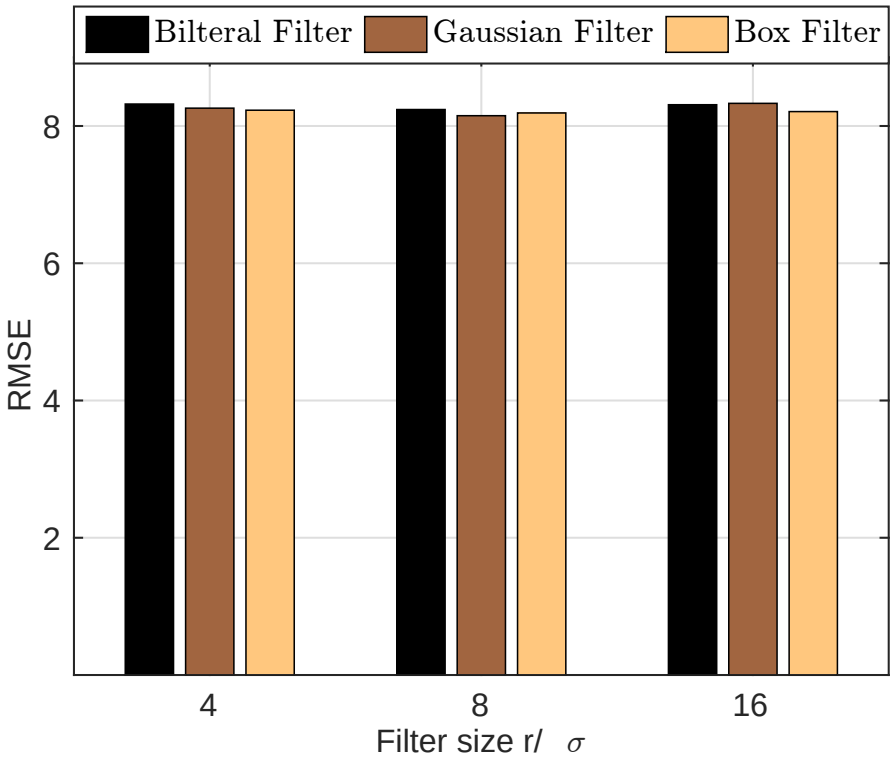


Figure 29: **The effect of smoothing filter** $\mathcal{F}_L$ on NYU Depth V2 for depth upsampling (16×). Our method is robust across smoothing filter type and size.

raw image as input. We find that using the unsharp mask as input not only achieves better filtering performance, but also convergences faster because network learning can focus on extracting the desired structure without the interference of redundant signals from the smooth basis of the image.

Smoothing filter $\mathcal{F}_L$. To obtain the unsharp masks G_m and I_m, we need a smoothing filter for $\mathcal{F}_L(I)$ and $\mathcal{F}_L(G)$. Next, we explore how the smoothing process affects the final filtering performance, we compare three different smoothing filters with different hyper-parameters on NYU Depth V2 for 16× depth upsampling. The hyper-parameter is the filtering size r for the box filter, or the Gaussian variance σ for the Gaussian and bilateral filters. We use three different values: $(4, 8, 16)$. The network of DJF [96] is used for f_a. As shown in Fig. 29, our method is robust across filter type and size. We opt for the box filter with a filtering size of 8 throughout our experiments because it is simple, efficient and effective.

3.5.4 *Successive filtering network*

Next, we investigate the effect of the network we designed for our successive filtering formulation. There are multiple amount blocks used in the successive filtering network. We explore how the number of amount blocks L affects the filtering performance on NYU Depth V2 for depth upsampling (16×) and depth denoising ($\sigma = 50$). $\mathcal{F}_L$ is a box mean filter with radius $r = 8$. For depth upsampling (16×), we first upsample the low-resolution depth image by bicubic interpolation to obtain the target resolution for $\mathcal{F}_L(I)$ and network learning. We make two observations from the results shown in Table

Table 7: **Ablation studies for our network** on NYU Depth V2 for depth upsampling (16×, RMSE) and denoising ($\sigma = 50$, PSNR). our model's filtering performance is consistently improved when increasing L from 1 to 5.

	L=1		**L=2**		**L=3**		**L=4**		**L=5**	
	Upsampling ↓	Denoising ↑	Upsampling ↓	Denoising ↑	Upsampling ↓	Denoising ↑	Upsampling ↓	Denoising ↑	Upsampling ↓	Denoising ↑
$\hat{I}^{(1)}$	7.97	40.08	8.03	39.95	8.05	39.92	8.09	39.87	8.16	39.81
$\hat{I}^{(2)}$	n.a.	n.a.	6.76	40.93	6.75	40.87	6.77	40.85	6.87	40.79
$\hat{I}^{(3)}$	n.a.	n.a.	n.a.	n.a.	6.33	41.27	6.28	41.25	6.32	41.21
$\hat{I}^{(4)}$	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	6.07	41.53	6.09	41.49
$\hat{I}^{(5)}$	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	6.02	41.61

7. First, our model's filtering performance is consistently improved when increasing L from 1 to 5. Second, we can obtain multiple (L) filtering results by training a single network. Each filtering result is as good as the result obtained by an independently trained network. The model's filtering performance doesn't improve a lot when L is increased from 4 to 5. Thus, we opt for $L = 4$ in the following experiments.

3.5.5 *Performance analysis.*

Next, we analyze the performance of different deep guided filtering methods from three aspects: run-time performance, model parameters and filtering accuracy. For our methods, we use the successive filtering network with $L = 4$. Thus, we can obtain four filtering models by training a single network, indicated by Ours($\hat{I}^{(1)}$), Ours($\hat{I}^{(2)}$), Ours($\hat{I}^{(3)}$) and Ours($\hat{I}^{(4)}$). We perform upsampling (16×) with all methods on the testing datasets (499 RGB/depth pairs) of NYU Depth V2. We perform all the testings on the same machine with an Intel Xeon E5-2640 2.20GHz CPU and an Nvidia GTX 1080 Ti GPU. The average run-time performance on 499 images with the size of 640×480 is reported in GPU mode with TensorFlow. From Table 8, we can see that Ours($\hat{I}^{(1)}$) has the fewest parameters (17 k), and Ours($\hat{I}^{(4)}$) achieves the best filtering accuracy (6.07 RMSE). Ours($\hat{I}^{(1)}$) achieves a competitive average run-time performance (31 ms) compared to the best one achieved by DJF [96] (29 ms).

Table 8: **Performance analysis.** on NYU Depth V2 for depth upsampling (16×). Our filtering models achieve competitive performance in terms of run-time, model parameters and filtering accuracy.

	Run-time (ms) ↓	Parameters (k)↓	Accuracy (RMSE)↓
DMSG†	36	534	8.21
DJF†	**29**	40	9.05
DGF†	34	32	7.82
SVLRM†	47	371	7.58
Ours($\hat{I}^{(1)}$)	31	**17**	8.09
Ours($\hat{I}^{(2)}$)	45	38	6.77
Ours($\hat{I}^{(3)}$)	58	59	6.28
Ours($\hat{I}^{(4)}$)	66	85	**6.07**

†Results from our reimplementation.

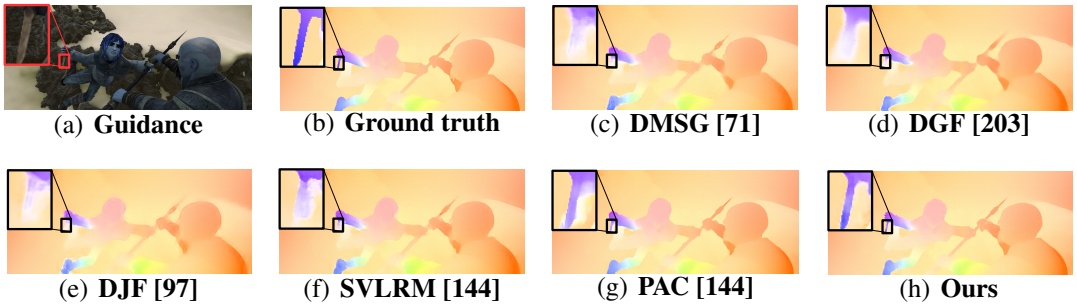
3.5.6 *Depth and flow upsampling*

Tables 9 and 10 show results for upsampling a depth image or optical flow image, under the guidance of its RGB image. We have noted that the existing works use different training settings and evaluation protocols. For fair comparison, we reimplement the main baseline methods under our experimental settings. Our filter performs well, especially on Sintel and the larger upsampling scales on NYU Depth. Different from the related works [1, 71, 97, 144, 182, 203], our model learns an amount function f_a to explicitly decide how to transfer the desired structure from guidance image to target image. Thus,

Table 9: **Depth upsampling** for 2×, 4×, 8× and 16× on NYU Depth V2. The depth values are measured in centimeter, and a boundary with 6 pixels is excluded for evaluation. We outperform alternative filters for almost all settings.

	2×		4×		8×		16×	
	RMSE ↓	SSIM ↑	RMSE ↓	SSIM ↑	RMSE ↓	SSIM ↑	RMSE ↓	SSIM ↑
DMSG [71]	-	-	3.78	-	6.37	-	11.16	-
DJF [97]	-	-	3.38	-	5.86	-	10.11	-
bFT [1]	-	-	3.35	-	5.73	-	9.01	-
PAC [182]	-	-	2.39	-	4.59	-	8.09	-
FWM [207]	-	-	2.16	-	4.32	-	7.66	-
SVLRM [144]	-	-	**1.74**	-	5.59	-	7.23	-
DMSG†	2.12	0.9957	3.43	0.9864	4.19	0.9814	8.21	0.9607
DGF†	2.29	0.9940	3.18	0.9897	4.78	0.9776	7.82	0.9568
DJF†	1.37	0.9972	2.85	0.9934	4.48	0.9801	9.05	0.9548
SVLRM†	1.28	0.9975	2.62	0.9946	3.96	0.9835	7.58	0.9616
Ours($\hat{I}^{(1)}$)	2.02	0.9963	2.90	0.9925	4.23	0.9839	8.09	0.9563
Ours($\hat{I}^{(2)}$)	1.65	0.9971	2.61	0.9938	3.82	0.9851	6.77	0.9657
Ours($\hat{I}^{(3)}$)	1.34	0.9974	2.40	0.9940	3.65	0.9857	6.28	0.9690
Ours($\hat{I}^{(4)}$)	**1.21**	**0.9976**	2.33	**0.9949**	**3.58**	**0.9863**	**6.07**	**0.9706**

†Results from our reimplementation.



(a) **Guidance** (b) **Ground truth** (c) **DMSG [71]** (d) **DGF [203]**

(e) **DJF [97]** (f) **SVLRM [144]** (g) **PAC [144]** (h) **Ours**

Figure 30: **Optical flow upsampling** (16×) on Sintel. We are able to maintain sharp and thin edges.

our model can be more effective and efficient to learn the desired output. Fig. 30 show our ability to better recover finer edges.

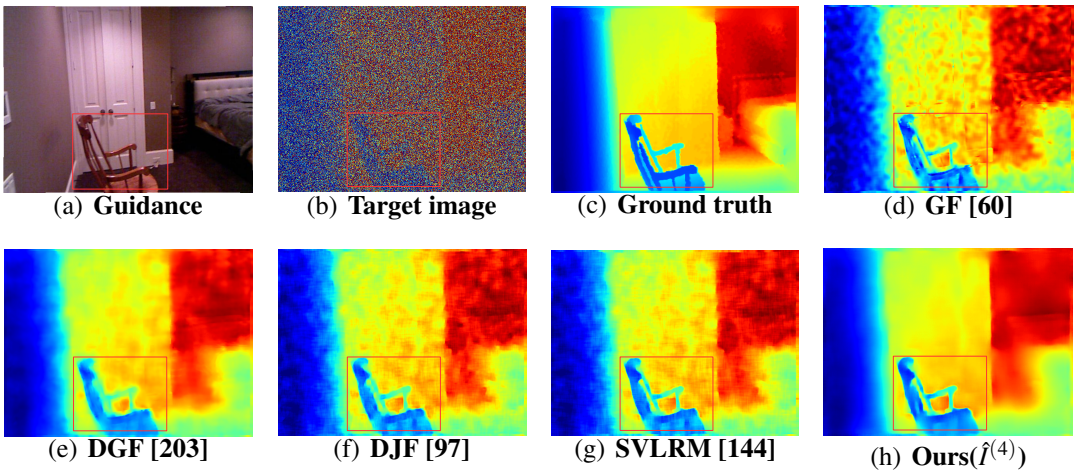
3.5.7 *Depth and natural image denoising*

Our formulation also allows for standard filtering without a guidance image by simply making G identical to I in Eq. (3.17). Intuitively, such a setup defines a structure-preservation filter, while the guided variant defines a structure-transfer filter. Next, we evaluate the ability to remove Gaussian noise from depth and natural images. For depth image denoising, its RGB image is used as guidance. For natural image denoising, the target and guidance image are the same RGB image. The quantitative results are shown in Tables 11 and 12. We obtain the best PSNR and SSIM scores on both datasets for all three noise levels. Fig. 31 and Fig. 32 highlight our ability to better remove noise and preserve finer edges compared to other filters.

Table 10: **Flow upsampling** for 2×, 4×, 8× and 16× on Sintel. We outperform alternative filters for almost all settings.

	2×		4×		8×		16×	
	EPE ↓	SSIM ↑	EPE ↓	SSIM ↑	EPE ↓	SSIM ↑	EPE ↓	SSIM ↑
DJF [97]	-	-	0.18	-	0.44	-	1.04	-
PAC [182]	-	-	0.11	-	0.26	-	0.59	-
FWM [207]	-	-	0.09	-	0.23	-	0.55	-
DMSG†	0.14	0.9928	0.24	0.9895	0.41	0.9811	0.96	0.9560
DGF†	0.11	0.9942	0.13	0.9934	0.31	0.9842	0.78	0.9692
DJF†	0.10	0.9951	0.17	0.9927	0.43	0.9837	1.04	0.9547
SVLRM†	0.09	0.9957	0.16	0.9921	0.36	0.9845	0.98	0.9567
Ours($\hat{I}^{(1)}$)	0.06	0.9988	0.11	0.9936	0.36	0.9851	0.86	0.9684
Ours($\hat{I}^{(2)}$)	0.05	0.9990	0.07	0.9942	0.29	0.9859	0.68	0.9734
Ours($\hat{I}^{(3)}$)	0.03	0.9991	0.05	0.9943	0.18	0.9864	0.52	0.9754
Ours($\hat{I}^{(4)}$)	**0.03**	**0.9991**	**0.04**	**0.9947**	**0.16**	**0.9867**	**0.45**	**0.9773**

†Results from our reimplementation.



(a) **Guidance** (b) **Target image** (c) **Ground truth** (d) **GF [60]**

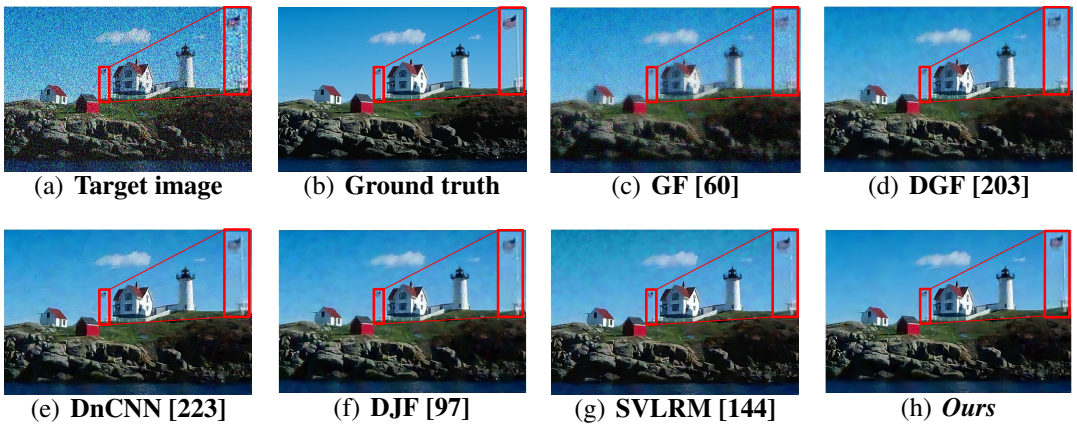
(e) **DGF [203]** (f) **DJF [97]** (g) **SVLRM [144]** (h) **Ours($\hat{I}^{(4)}$)**

Figure 31: **Depth denoising result** ($\sigma = 50$) on NYU depth v2. Our filter better preserves edges and removes noise.

3.5.8 *Cross-modality filtering*

Finally, we demonstrate that our models trained on one modality can be directly applied to other modalities. Here, we use the models trained with RGB/depth image pairs for the joint upsampling of bw/color and RGB/saliency image pairs, and the joint denoising of RGB/NIR and flash/no-flash image pairs. For our method, we use the model of $Ours(\hat{I}^{(4)})$. Following [97], for the multi-channel target image, *i.e.*, no-flash image, we apply the trained models independently for each channel. For the single-channel guidance image, *i.e.*, NIR image, we replicate it three times to obtain a 3-channel guidance image.

Joint upsampling. To speed up the translation from one image to another image, one strategy is to perform translation at a coarse resolution and then upsample the low-resolution solution back to the original one with a joint image upsampling filter. Here,



(a) **Target image** (b) **Ground truth** (c) **GF [60]** (d) **DGF [203]**

(e) **DnCNN [223]** (f) **DJF [97]** (g) **SVLRM [144]** (h) ***Ours***

Figure 32: **Denoising result** ($\sigma = 50$) on BSDS500. Our filter better preserves finer edges with fewer noise artifacts.

Table 11: **Depth image denoising** on NYU Depth V2. Our filter achieves the best results for all settings.

	$\sigma = 15$		$\sigma = 25$		$\sigma = 50$	
	PSNR ↑	SSIM ↑	PSNR ↑	SSIM ↑	PSNR ↑	SSIM ↑
DGF†	45.52	0.9650	43.96	0.9579	40.15	0.9422
DJF†	46.06	0.9633	43.58	0.9462	39.24	0.8826
SVLRM†	47.35	0.9722	44.38	0.9524	39.84	0.8891
Ours($\hat{I}^{(1)}$)	46.02	0.9627	43.62	0.9474	39.87	0.9112
Ours($\hat{I}^{(2)}$)	46.92	0.9704	44.53	0.9586	40.85	0.9310
Ours($\hat{I}^{(3)}$)	47.30	0.9732	44.93	0.9635	41.25	0.9422
Ours($\hat{I}^{(4)}$)	**47.45**	**0.9750**	**45.09**	**0.9664**	**41.53**	**0.9488**

†Results from our reimplementation.

we demonstrate that our models act as joint upsampling filters well on bw/color and RGB/saliency image pairs translation tasks. For bw/color translation, we use 68 bw images from BSD68 dataset [156], and the colorization model proposed by Lizuka *et al.* [73] is used as translation model. For RGB/saliency translation, we use 1000 RGB image from ECSSD dataset [166], and the saliency region detection model proposed by Hou *et al.* [69] as translation model. The input images, *i.e.*, bw images and RGB images, are first downsampled by a factor of 4× using nearest-neighbor interpolation, and then are feed into the translation models to generate the output images. After that, we recover the output images to the original resolution under the guidance of the original input images by various joint upsamling methods. Table 13 shows the quantitative results, and we can see that our model achieves the best performance for both two tasks. The joint upsampling pipeline performs more than two times faster than direct translation on the GPU mode. We also provide the qualitative results in Fig. 33 and 34 to show that the proposed model better recovers finer details.

Joint denoising. We introduce two datasets for the joint denoising experiments. **Flash/no-flash** [62] consists of 120 image pairs, where the no-flash image is used for denoising under the guidance of its flash version [60, 97, 146, 209]. **Nirscene1** [18]

Table 12: **Natural image denoising** on BSDS500. Our filter achieves the best results for all settings.

	$\sigma = 15$		$\sigma = 25$		$\sigma = 50$	
	PSNR ↑	SSIM ↑	PSNR ↑	SSIM ↑	PSNR ↑	SSIM ↑
DnCNN†	33.12	0.9166	30.48	0.8618	27.10	0.7495
DGF†	31.89	0.9010	29.56	0.8409	26.31	0.7229
DJF†	33.10	0.9168	30.49	0.8646	27.09	0.7496
SVLRM†	33.01	0.9202	30.60	0.8712	27.37	0.7686
Ours($\hat{I}^{(1)}$)	33.41	0.9272	30.80	0.8758	27.43	0.7716
Ours($\hat{I}^{(2)}$)	33.65	0.9341	31.05	0.8868	27.73	0.7829
Ours($\hat{I}^{(3)}$)	33.76	0.9354	31.16	0.8890	27.87	0.7895
Ours($\hat{I}^{(4)}$)	**33.79**	**0.9361**	**31.19**	**0.8906**	**27.91**	**0.7938**

†Results from our reimplementation.

Table 13: **Cross-modality filtering** for joint upsampling (4×) on bw/color and RGB/saliency pairs. Our filter achieves the best results for all settings.

	bw/color		**RGB/saliency**	
	RMSE ↓	SSIM ↑	F-measure ↑	SSIM ↑
GF†	11.51	0.6054	0.685	0.5365
DGF†	11.17	0.6041	0.701	0.5431
DJF†	10.96	0.6046	0.697	0.5378
SVLRM†	10.78	0.6074	0.699	0.4588
Ours	**10.39**	**0.6095**	**0.705**	**0.6087**

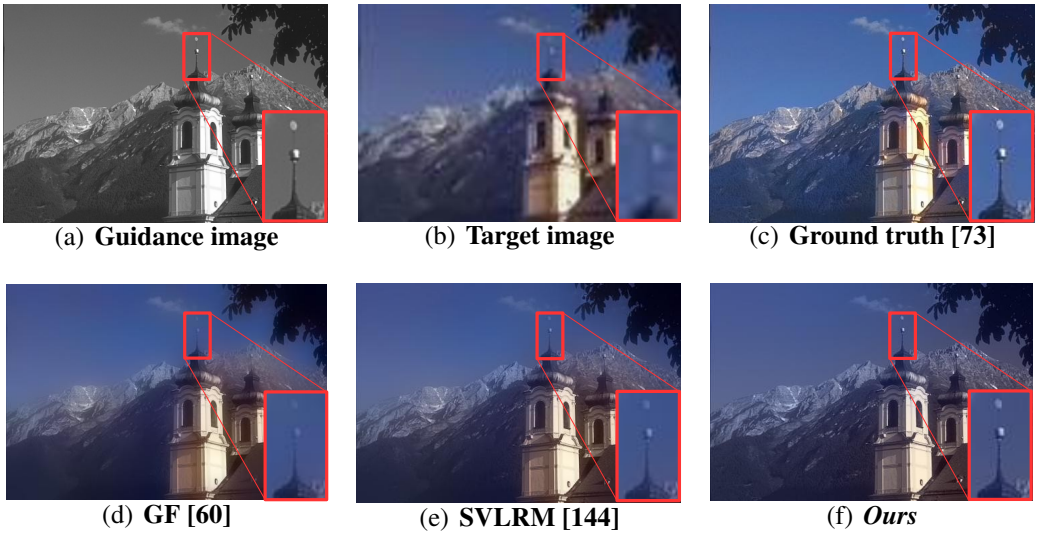
†Results from our reimplementation.

Table 14: **Cross-modality filtering** for joint denoising ($\sigma = 25$) on Flash/no-flash and RGB/NIR pairs. Our filter achieves the best results for all settings.

	Flash/no-flash		**RGB/NIR**	
	PSNR ↑	SSIM ↑	PSNR ↑	SSIM ↑
GF†	29.43	0.7675	27.22	0.6971
DGF†	28.11	0.7246	26.40	0.6436
DJF†	29.53	0.7407	27.56	0.6826
SVLRM†	30.27	0.7584	28.33	0.7084
Ours	**30.76**	**0.7699**	**28.95**	**0.7226**

†Results from our reimplementation.

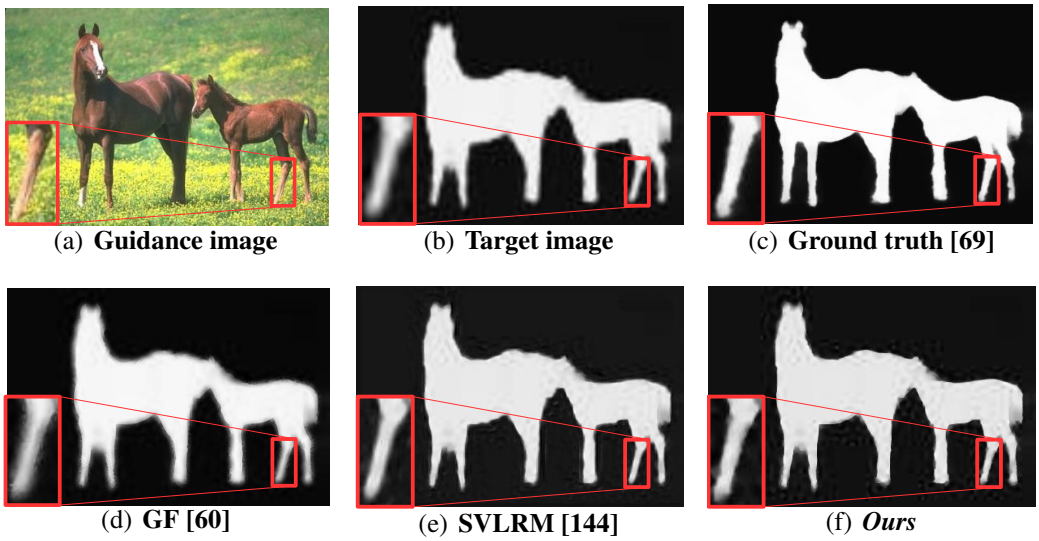
consists of 477 RGB/NIR image pairs in 9 categories, where the RGB image is used for denoising under the guidance of its NIR version [97, 199, 209]. Table 14 shows that our filter has a better denoising ability than alternatives. Fig. 35 and Fig. 36 provide qualitative results, which shows that our model better preserves important structures while removing noises.



(a) **Guidance image** (b) **Target image** (c) **Ground truth [73]**

(d) **GF [60]** (e) **SVLRM [144]** (f) ***Ours***

Figure 33: **Joint upsampling result** (4×) on bw/color. Our filter recovers more desirable details.



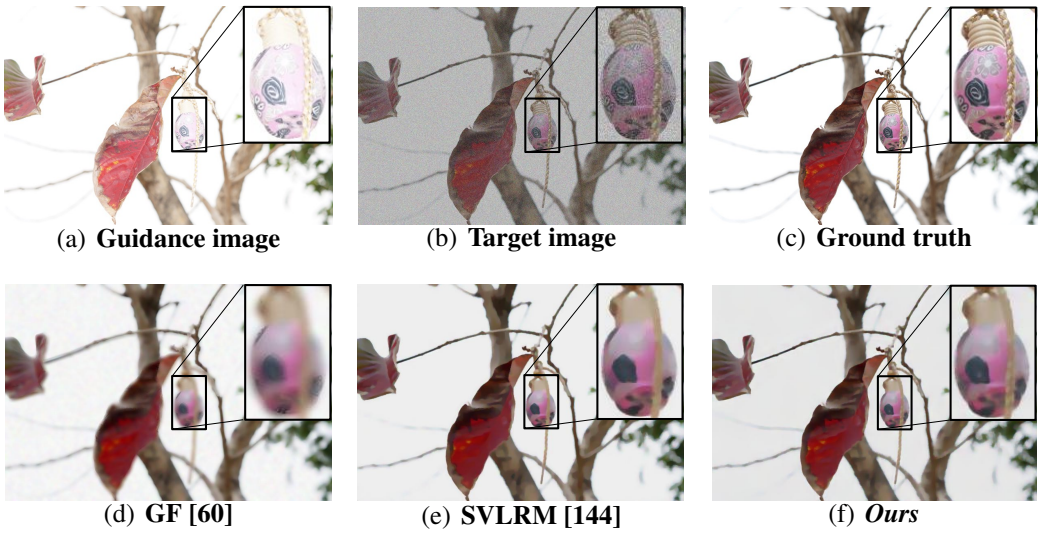
(a) **Guidance image** (b) **Target image** (c) **Ground truth [69]**

(d) **GF [60]** (e) **SVLRM [144]** (f) ***Ours***

Figure 34: **Joint upsampling result** (4×) on RGB/saliency. Our filter recovers sharper edges.

3.6 CONCLUSION

In this chapter, we have introduced a new and simplified guided filter. With inspiration from unsharp masking, our proposed formulation only requires estimating a single coefficient, in contrast to the two entangled coefficients required in current approaches. Based on the proposed formulation, we introduce a successive guided filtering network. Our network allows for a trade-off between accuracy and efficiency by choosing different filtering results during inference. Experimentally, we find that the proposed filtering method better preserves sharp and thin edges, while avoiding unwanted structures transferred from guidance. We furthermore show that our approach is effective for various applications such as upsampling, denoising, and cross-modal filtering.



(a) **Guidance image** (b) **Target image** (c) **Ground truth**

(d) **GF [60]** (e) **SVLRM [144]** (f) ***Ours***

Figure 35: **Joint denoising result** ($\sigma = 25$) on Flash/no-flash. Our filter better removes noises with less blurring.

(a) **Guidance image** (b) **Target image** (c) **Ground truth**

(d) **GF [60]** (e) **SVLRM [144]** (f) ***Ours***

Figure 36: **Joint denoising result** ($\sigma = 25$) on RGB/NIR. Our filter better preserves finer details with fewer noises.

4

COUNTING WITH FOCUS FOR FREE

4.1 INTRODUCTION

This chapter strives to count objects in images, whether they are people in crowds [211, 228], cars in traffic jams [54] or cells in petri dishes [131]. The leading approaches for this challenging problem count by summing the pixels in a density map [94] as estimated with a convolutional neural network, *e.g.* [20, 76, 98, 131]. While this line of work has shown to be effective, the rich source of supervision from the point annotations is only used to construct the density maps for training. The premise of this work is that point annotations can be repurposed to further supervise counting optimization in deep networks, for free.

The main contribution of this chapter is summarized in Figure 37. Besides creating density maps, we show that points can be exploited as free supervision signal in two other ways. The first is focus from segmentation. From point annotations, we construct binary segmentation maps and use them in a separate network branch with an accompanying segmentation loss to focus on areas of interest only. The second is focus from global density. The relative amount of point annotations in images is used to focus on the global image density through another branch and loss function. Both forms of focus are integrated with the density estimation in a single network trained end-to-end with a multi-level loss. Different from standard attention [24, 83, 105, 206], where a form of focus needs to be learned for the task at hand and the learned weighing map implicitly guides the network to focus on task-relevant features, our proposed focus learns weighing maps with a specific supervision derived for free from point annotations. Focus for free allows the counting network to explicitly emphasize meaningful features and suppress undesired ones.

Overall, we make three contributions in this chapter: (*i*) We propose supervised focus from segmentation, a network branch which guides the counting network to focus on areas of interest. The supervision is obtained from the already provided point annotations. (*ii*) We propose supervised focus from global density, a branch which regularizes the counting network to learn a matching global density. Again the supervision is obtained for free from the point annotations. (*iii*) We introduce a new kernel density estimator for point annotations with non-uniform point distributions. For the deep network, we design an improved encoder-decoder network to deal with varying object scales in images. Experimental evaluation on four counting datasets shows the benefits of our focus for free, kernel estimation, and end-to-end network architecture, resulting in state-of-the-art counting accuracy. To further demonstrate the potential of our approach for counting

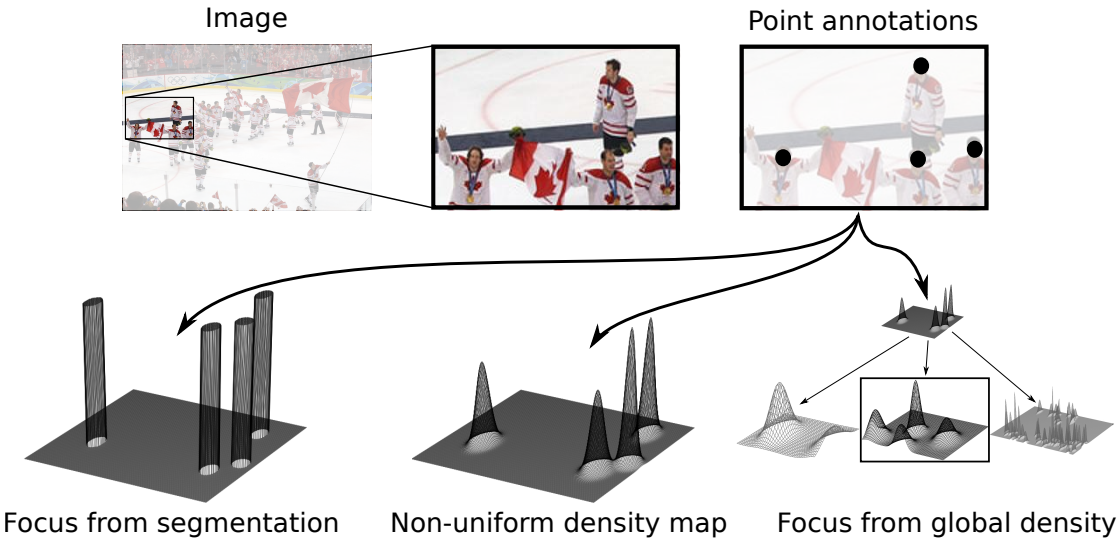


Figure 37: **Focus for free** in counting. From point supervision, we learn to obtain a focus from segmentation, a focus from global density, and an improved density maps. Combined, they result in better counting estimation irrespective of the base network.

under varying object scales and crowding levels, we provide the first counting results on WIDER FACE, normally used for large-scale face detection [211].

4.2 RELATED WORK

Density-based counting. Deep convolutional networks are widely adopted for counting by estimating density maps from images. Early works, *e.g.* [143, 176, 222, 228], advocate a multi-column convolutional neural network to encourage different columns to respond to objects at different scales. Despite their success, these types of networks are hard to train due to structure redundancy [98] and conflicts resulting from optimization among different columns [9, 165].

Due to their architectural simplicity and training efficiency, single column deep networks have received increasing interest *e.g.* [20, 98, 111, 117, 171]. Cao *et al.* [20] , for example, propose an encoder-decoder network to predict high-resolution and high-quality density maps using a scale aggregation module. Li *et al.* [98] combine a VGG network with dilated convolution layers to aggregate multi-scale contextual information. Liu *et al.* [117] rely on a single network by leveraging abundantly available unlabeled crowd imagery in a learning-to-rank framework. Shi *et al.* [171] train a single VGG network with a deep negative correlation learning strategy to reduce the risk of over-fitting. We also employ single column networks, but rather than focusing solely on density map estimation, we repurpose the point annotations in multiple ways to improve counting.

Recently, multi-task networks have shown to reduce the counting error [9, 105, 153, 160, 165, 172]. Sam *et al.* [160], for example, train a classifier to select the optimal regressor from multiple independent regressors for particular input patches. Ranjan *et al.* [153] rely on one network to predict a high resolution density map and a helper-network to predict a density map at a low resolution. In this chapter, we also investigate counting from a multi-task perspective, but from a different point of view. We posit that the point annotations serve more purposes than just constructing density maps, and we

propose network branches with supervised focus from segmentation and global density to repurpose the point annotations for free. Our focus for free benefits counting regardless of the base network, and is complementary to other state-of-the-art solutions.

Counting with attention. Attention mechanisms [206] have enabled progress in a wide variety of computer vision challenges [24, 53, 99, 227, 230]. Soft attention is the most widely used since it is differentiable and thus can be directly incorporated in an end-to-end trainable network. The common way to incorporate soft attention is to add a network branch with one or more hidden layers to learn an attention map which assigns different weights to different regions of an image. Spatial and channel attention are two well explored types of soft attention [24]. Spatial attention learns a weighting map over the spatial coordinates of the feature map, while channel attention does so for the feature channels of the map.

A few works have investigated density-based counting with spatial attention [68, 83, 105]. Liu *et al.* [105], for example, estimate the density of a crowd by generating separate detection- and regression-based density maps. They fuse these two density maps guided by an attention map, which is implicitly learned together with the density map regression loss. While we share the notion of assisting the density-based counting with a focus, we show in this chapter that such an attention does not need to be learned from scratch and instead can be derived from the existing point annotations. More specifically, we construct a segmentation map and a global density derived from the ground-truth annotated points as two additional, yet free, supervision signals for better counting.

4.3 FOCUS FOR FREE

We formulate the counting task as a density map estimation problem, see *e.g.* [94, 171, 228]. Given N training images $\{(X_i, \mathcal{P}_i)\}_{i=1}^{N}$, with $X_i \subset \mathcal{X}$ the input image and $\mathcal{P}_i$ a set of point annotations, one for each object, we use the point annotations to create a ground-truth density map by convolving the points with a Gaussian kernel,

$$D_i(p) = \sum_{P \in \mathcal{P}_i} \mathcal{N}(p|\mu = P, \sigma_P^2), \tag{4.1}$$

where p denotes a pixel location, P denotes a single point annotation and $\mathcal{N}(p|\mu = P, \sigma_P^2)$ is a normalized Gaussian kernel with mean P and an isotropic covariance σ_P^2. The global object count T_i of the image X_i can be obtained by summing all pixel values within the density map D_i, *i.e.*, $T_i = \sum_{p \in X_i} D_i(p)$. Learning a transformation from input images to density maps is done through deep convolutional networks. Let $\Psi(X) : \mathbb{R}^{3 \times W \times H} \mapsto \mathbb{R}^{W \times H}$ denote such a mapping given an arbitrary deep network Ψ for image X, with W and H the width and height of the image. In this chapter, we investigate two ways that repurpose the point annotations to help supervising the network Ψ from input images to density maps. An overview of our approach, in which multiple branches are combined on top of a base network, is shown in Figure 38.

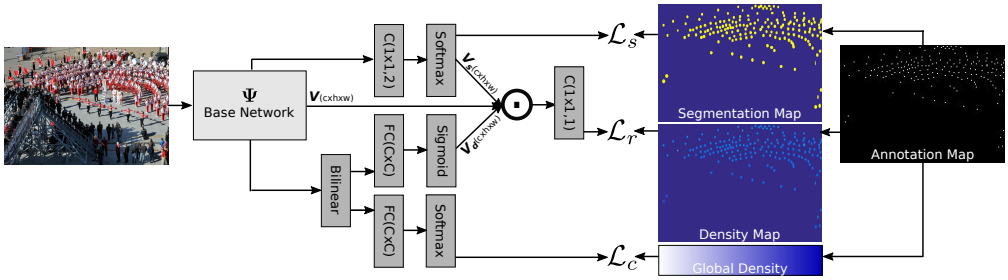


Figure 38: **Overview of our approach**. Top branch: focus from segmentation learns a focus map V_s with the aid of a segmentation map (Section 4.3.1). Bottom branch: focus from global density learns a focus map V_d with the aid of a global density (Section 4.3.2). Both supervision signals are obtained from the same point-annotations, for which we introduce an improved kernel estimator (Section 4.3.3). Both branches with focus for free are integrated with the output of a base network by element-wise multiplication and end-to-end optimized through a multi-level loss (Section 4.3.4).

4.3.1 *Focus from segmentation*

The first way to repurpose the point annotations is to provide a spatial focus. Intuitively, pixels that are within a specific range of any point annotation should be of high focus, while pixels in undesired regions should be mostly disregarded. In the standard setup where the optimization is solely dependent on the density map, each pixel counts equally to the network loss. Given that only a fraction of the pixels are near point annotations, the loss will be dominated by the majority of irrelevant pixels. To overcome this limitation, we reuse the point annotations to create a binary segmentation map and exploit this map to provide the focused supervision through a stand-alone loss function.

Segmentation map. The binary segmentation map is obtained as a function of the point annotations and their estimated variance. The binary value for each pixel p in training image i is determined as:

$$S_i(p) = \begin{cases} 1 & \text{if } \exists_{P \in \mathcal{P}_i}\left(\|p - P\|^2 \leq \sigma_P^2\right), \\ 0 & \text{otherwise.} \end{cases} \tag{4.2}$$

Equation 4.2 states that a pixel p obtains a value of one if at least one point P is within its variance range σ_P as specified by a kernel estimator.

Segmentation focus. Let $V \in \mathbb{R}^{C \times W \times H}$ denote the output of the base network. We add a new branch on top of the network denoted as $\mathcal{F}_s$ with network parameters θ_s. Furthermore, let θ_n denote the parameters of the base network. We propose a per-pixel weighted focal loss [102] to obtain a supervised focus from segmentation for input image X:

$$\begin{aligned} \mathcal{L}_s(X; \theta_n, \theta_s) = \sum_{l \in \{0,1\}} -\alpha^l S^l \\ (1 - \mathcal{F}_s(X; \theta_n, \theta_s))^{\gamma_s} log(\mathcal{F}_s(X; \theta_n, \theta_s)), \end{aligned} \tag{4.3}$$

where $\alpha^l = 1 - \frac{|S^l|}{|S|}$. The focal parameter γ_s is set to 2 throughout this network, as recommended by [102]. The segmentation branch is visualized at the top of Figure 38.

Network details. After the output of the base network, we perform a 1×1 convolution layer with parameters $\theta_s \in \mathbb{R}^{C\times 2\times 1\times 1}$, followed by a softmax function δ to generate a per-pixel probability map $\mathbb{P}_i = \delta(\theta_s V) \in \mathbb{R}^{2\times W\times H}$. From this probability map, the second value along the first dimension represents the probability of each pixel being part of the segmentation foreground. We furthermore tile this slice C times to construct a separate output tensor $V_s \in \mathbb{R}^{C\times W\times H}$, which will be used in the density estimation branch itself.

4.3.2 *Focus from global density*

Next to a spatial focus, point annotations can also be repurposed by examining their context. It is well known that low density crowds exhibit coarse texture patterns while high density crowds exhibit very fine texture patterns [125]. Here, we exploit this knowledge for the task of counting. Given a network output $V \in \mathbb{R}^{W\times H\times C}$, we employ a bilinear pooling layer [51, 103] to capture the feature statistics in a global context, which is known to be particularly suitable for texture and fine-grained recognition [51, 103]. In this chapter, we match global contextual patterns to the distribution of points in training images to obtain a supervised focus from global density.

Global density. For patch j in training image i, its global density is given as:

$$G_{j,i} = \frac{|\mathcal{P}_{j,i}|}{L}, \tag{4.4}$$

where $|\mathcal{P}_{j,i}|$ denotes the number of point annotations in patch j and L denotes the global density step size, which is computed for a dataset as:

$$L = \left\lfloor \max_{i=1,..,N} \left(\frac{|\mathcal{P}_i|}{Z_i} \cdot Z_{j,i} \right) / M \right\rfloor + 1, \tag{4.5}$$

with Z_i and $Z_{j,i}$ the number of pixels in image i and patch j respectively. Intuitively, the step size computes the maximum global density over image patches and M states how many global density levels are used overall.

Global density focus. With $V \in \mathbb{R}^{C\times W\times H}$ again the output of the base network, we add a second new branch $\mathcal{F}_c$ with network parameters θ_c. We propose the following global density loss function:

$$\begin{aligned} \mathcal{L}_c(X;\theta_n,\theta_c) = \sum_{l\in\{0,1,..,M\}} -G^l \\ (1-\mathcal{F}_c(X;\theta_n,\theta_c))^{\gamma_c} log(\mathcal{F}_c(X;\theta_n,\theta_c)), \end{aligned} \tag{4.6}$$

where γ_c is set to 2 as well. The above loss function aims to match the global density of the estimated density map with the global density of the ground truth density map. The corresponding global density branch is visualized at the bottom of Figure 38.

Network details. For network output V, we first perform an outer product $B = VV^T \in \mathbb{R}^{C\times C}$, followed by a mean pooling along the second dimension to aggregate the bilinear features over the image, *i.e.* $\hat{B} = \frac{1}{C}\sum_{i=1}^{C} B[:,i] \in \mathbb{R}^{C\times 1}$. The bilinear vector $\hat{B}$ is ℓ_2-normalized, followed by signed square root normalization, which has shown

to be effective in bilinear pooling [103]. Then we use a fully connected layer with parameters $\theta_c \in \mathbb{R}^{C \times M}$ followed by a softmax function δ_c to make individual prediction $\mathbb{C} = \delta_c(\theta_c \hat{B}) \in \mathbb{R}^{M \times 1}$ for the global density. Furthermore, another fully-connected layer with parameters $\theta_d \in \mathbb{R}^{C \times C}$ followed by sigmoid function δ_d also on top of the bilinear pooling layer is added to generate global density focus output $\mathbb{D} = \delta_d(\theta_d \hat{B}) \in \mathbb{R}^{C \times 1}$. We note that this results in a focus over the channel dimensions, complementary to the focus over the spatial dimensions from segmentation. Akin to the focus from segmentation, we tile the output vector into $V_d \in \mathbb{R}^{C \times W \times H}$, also to be used in the density estimation branch.

4.3.3 *Non-uniform kernel estimation*

Both the density estimation itself and the focus from segmentation require a variance estimation for each point annotation, where the variance corresponds to the size of the object. Determining the variance σ_P for each point P is difficult because of object-size variations caused by perspective distortions. A common solution is to estimate the size (*i.e.* the variance) of an object as a function of the K nearest neighbour annotations, *e.g.* the Geometry-Adaptive Kernel of Zhang *et al.* [228]. However, this kernel is effective only under the assumption that objects in images are uniformly distributed, which typically does not happen in counting practice. As such, we introduce a simple kernel that estimates the variance of a point annotation P by splitting an image into local regions:

$$\sigma_P = \frac{1}{|R_{(w,h)}|} \sum_{a \in R_{(w,h)}} \beta \bar{d}_a, \quad \bar{d}_a = \frac{1}{K} \sum_{k=1}^{K} d_{k,a} \tag{4.7}$$

where w and h are the hyper-parameters which determine the range of point annotation P-centered local region R, and we set their value to one-eighth of image size in our experiments. a denotes an arbitrary point annotation located in R. $|R_{(w,h)}|$ means the number of p. $\bar{d}_p$ indicates the average distance between annotated point p and its k nearest neighbors, and β is a user-defined hyper-parameter. By estimating the variance of point annotations locally, we no longer have to assume that points are uniformly distributed over the whole image.

4.3.4 *Architecture and optimization*

Network. To maximize the ability to focus and use the most accurate kernel estimation, we want the network output to be of the same width and height as the input image. Recently, encoder-decoder networks have been transferred from other visual recognition tasks [101, 216] to counting [20, 153, 165, 226]. We found that to make the encoder-decoder architectures better suited for counting, the wide variation in object-scale under perspective distortions needs to be addressed. As such, in our encoder-decoder architecture a distiller module is added between the step from encoder to decoder. The purpose of this module is to aggregate multi-level information from the encoder by distilling the most vital information for counting.

For the encoder, we make the original dilated residual network [216] suitable for our task by changing the channel of the feature maps after level 4 from 256/512 to 96 to reduce the model's parameters for the sake of avoiding over-fitting, given the low amount of training examples in counting. After the encoder, the distiller module fuses the features from level 4, 5, 7 and 8 in the encoder module by using skip connections and a concatenation operation. Then four convolution layers are used to further process the fused features to obtain a more compact representation. The reason why we do not fuse the features from level 6 is that level 6 comprises convolution layers with large dilation rates, which is prone to cause gridding artifacts [197, 216]. Compared to other works which fuse multiple networks with different kernels to deal with object-scale variations [143, 176, 228], the proposed network aggregates the features from different layers which have different receptive fields, and is much more efficient and easy to train. The decoder module uses 3 deconvolution layers with a kernel size of 4×4 and a stride size of 2×2 to progressively recover the spatial resolution. To avoid the checkerboard artifact problem caused by regular deconvolutional operation [141, 197], we add two convolution layers after each deconvolution layer. We provide a detailed ablation on the encoder-distiller-decoder network in Appendix 4.6.

Multi-level loss. The final counting network with a focus for free contains three branches, $\mathcal{F}_r$ for the pixel-wise density estimation, $\mathcal{F}_s$ for the binary segmentation, and $\mathcal{F}_c$ for the global density prediction. Let $(\theta_n, \theta_r, \theta_s, \theta_c, \theta_d)$ denote the network parameters for the base network and the branches. For the density estimation, we first combine the outputs of the base network V with the tiled outputs V_s and V_d from the focus for free. We fuse the three sources of information by element-wise multiplication and feed the fusion to a 1×1 convolution layer with parameters $\theta_r \in \mathbb{R}^{C \times 1 \times 1 \times 1}$, resulting in an output density map.

For the density estimation, the $L2$ loss is a common choice, but it is also known to be sensitive to outliers, which hampers generalization [15]. We prefer to learn the density estimation branch by jointly optimizing the $L2$ and $L1$ loss, which adds robustness to outliers:

$$\begin{aligned} \mathcal{L}_r(X; \theta_n, \theta_r, \theta_d) = \frac{1}{2} \parallel \mathcal{F}_r(X; \theta_n, \theta_r, \theta_d) - Y \parallel_2^2 + \\ \parallel \mathcal{F}_r(X; \theta_n, \theta_r, \theta_d) - Y \parallel_1, \end{aligned} \tag{4.8}$$

where Y denotes the ground truth density map. The loss functions of the three branches are summed to obtain the final objective function:

$$\begin{aligned} \mathcal{L}(X; \theta_n, \theta_r, \theta_s, \theta_c, \theta_d) = \lambda_r \mathcal{L}_r(X; \theta_n, \theta_r, \theta_d) + \\ \lambda_s \mathcal{L}_s(X; \theta_n, \theta_s) + \lambda_c \mathcal{L}_c(X; \theta_n, \theta_c), \end{aligned} \tag{4.9}$$

where $(\lambda_r, \lambda_s, \lambda_c)$ denote the weighting parameters of the different loss functions. Throughout this work these parameters are set to $(1, 10, 1)$, since the loss values of the segmentation branch are typically an order of magnitude lower than the others.

4.4 EXPERIMENTS AND RESULTS

4.4.1 *Experimental setup*

DATASETS. **ShanghaiTech** [228] consists of 1198 images with 330,165 people. This dataset is divided into two parts: **Part_A** with 482 images in which crowds are mostly dense (33 to 3139 people), and **Part_B** with 716 images, where crowds are sparser (9 to 578 people). Each part is divided into a training and testing subset as specified in [228]. **TRANCOS** [54] contains 1,244 images from different roads to count vehicles, varying from 9 to 105. We train on the given training data (403 images) and validation data (420 images) without any other datasets, and we evaluate on the test data (421 images). **Dublin Cell Counting (DCC)** [131] is a cell microscopy dataset, consisting of 177 images, with a cell count from 0 to 100. For training 100 images are used, the remaining 77 form the test set. **WIDER FACE** [211] is a recent large-scale face detection benchmark. In this chapter, we reuse this data for counting as a complementary crowd dataset. Compared to the existing crowd dataset like ShanghaiTech [228], WIDER FACE is more challenging due to large variations in scale, occlusion, pose, and background clutter. Moreover, it contains more images, in total 32,203, divided in 40% training, 10% validation and 50% testing. The ground truth of the test set is unavailable, so we use the validation set for testing. Each face is annotated by a bounding box, instead of a point annotation, which enables us to evaluate our kernel estimator and allows for ablation under varying object scales and crowding levels.

PRE-PROCESSING. For all datasets, we normalize the input RGB images by dividing all values by 255. During training, we augment the images by randomly cropping 128×128 patches. No cropping is performed during testing.

NETWORK. We implement the proposed method with TensorFlow on a machine with a single GTX 1080 Ti GPU. The network is trained using Adam with a mini-batch of 16. We set the β_1 to 0.9, β_2 to 0.999 and the initial learning rate to 0.0001. Training is terminated after a maximum of 1000 epochs. Code and networks will be released.

KERNEL COMPUTATION. For datasets with dense objects, *i.e.* ShanghaiTech Part_A and TRANCOS, we use our proposed kernel with $\beta = 0.3$ and $k = 5$. For ShanghaiTech Part_B and DCC, we set the Gaussian kernel variance to $\sigma = 5$ and $\sigma = 10$ respectively, following [98, 171]. For WIDER FACE, we obtain the Gaussian kernel variance by leveraging the box annotations. For the focus from global density, we use $M = 8$ density levels for ShanghaiTech Part_A and 4 for the other datasets.

EVALUATION METRICS. Following We report the standardized Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) metrics given count estimates and ground truth counts [171,222,228]. Since these global metrics ignore where objects have been counted, we also report results using the Grid Average Mean absolute Error (GAME) metric. [54]. GAME aggregates count estimates over local regions as: $GAME(L) = \frac{1}{N} \cdot \sum_{n=1}^{N} (\sum_{l=1}^{4^L} |(y_n^l - \tilde{y}_n^l)|)$, with N the number of images and y_n^l and $\tilde{y}_n^l$ the ground truth and the estimated counts in a region l of the n^{th} image. 4^L denotes the number

Table 15: **Effect of focus from segmentation** in terms of MAE on ShanghaiTech Part_A and WIDER FACE. Across both datasets and across multiple object scales (small, medium, large), our approach outperforms the base network, even when adding spatial attention.

	Part_A	**WIDER FACE**			
	overall	small	medium	large	overall
Base network	74.8	9.2	2.7	2.2	4.7
w/ Spatial attention [24]	84.5	8.7	2.6	3.1	4.8
w/ Segmentation focus	**72.3**	**8.6**	**2.3**	**2.0**	**4.3**

of grids, non-overlapping regions which cover the full image. When L is set to 0 the GAME is equivalent to the MAE. Finally, we report two standard metrics, PSNR (Peak Signal-to-Noise Ratio) and SSIM (Structural Similarity in Image [200]), to evaluate the quality of the predicted density maps. We only report these two metrics on ShanghaiTech Part_A because they are not commonly reported on the other datasets.

4.4.2 *Focus from segmentation*

We first analyze the effect of the proposed focus from segmentation. This experiment is performed on both ShanghaiTech Part_A and WIDER FACE. We compare to two baselines. The first performs counting using the base network, where the loss is only optimized with respect to the density map estimation. Unless stated otherwise, the encoder-distiller-decoder network is used as base network by default in the following experiments. The second baseline adds a spatial attention on top of this base network, as proposed in [24]. The results are shown in Table 15.

For ShanghaiTech Part_A, the base network obtains an MAE of 74.8 and the addition of spatial-attention actually increases the count error to 84.5 MAE, as it fails to emphasize relevant features. In contrast, our proposed focus from segmentation can explicitly guide the network to focus on task-relevant regions and it reduces the count error from 74.8 to 72.3 MAE.

For WIDER FACE, the box annotations allow us to perform an ablation study on the accuracy as a function of the object scale. We define the scale levels of each image as $I_{scale} = \frac{F_s}{F_n}$, where F_s and F_n denote face size and face number. We sort the test images in ascending order according to their scale level. Finally, the test images are divided uniformly into three sets: small, medium and large. In Table 15, we provide the results across multiple object scales. We observe that across all object scales, our approach is preferred, reducing the MAE from 4.7 (base network) and 4.8 (with spatial attention) to 4.3. The ablation study also reveals why spatial attention is not very effective overall; while improvements are obtained when objects are small, spatial attention performs worse when objects are large. Our segmentation focus from reused point annotations avoids such issues.

Table 16: **Effect of focus from global density** in terms of MAE on ShanghaiTech Part_A and WIDER FACE. Our approach is preferred for both datasets. The ablation study on WIDER FACE shows our focus from global density is most effective when scenes are sparse in number of objects.

	Part_A	**WIDER FACE**			
	overall	sparse	medium	dense	overall
Base network	74.8	2.1	2.5	9.5	4.7
w/ Channel attention [24]	73.4	1.6	2.3	**7.8**	3.9
w/ Global-density focus	**71.7**	**0.9**	**1.6**	8.0	**3.5**

4.4.3 *Focus from global density*

Next, we demonstrate the effect of our proposed focus from global density. For this experiment, we again compare to two baselines. Apart from the base network, we compare to the channel attention of [24]. For fair comparison, we replace the mean pooling used in the channel attention of [24] with bilinear pooling as used in our method for the sake of better encoding global context cues. The counting results are shown in Table 16. Channel-attention can reduce the error (from 74.8 to 73.4 MAE) in ShanghaiTech Part_A compared to using the base network only, since the attention map is learned on top of a bilinear pooling layer which encodes global context cues. Our focus from global density reduces the count error further to 71.7 MAE due to more specific focus from free supervision.

To demonstrate that our focus has a lower error on different crowding levels, we perform a further ablation study on WIDER FACE. We define the crowding levels of each image as $I_{crowding} = \frac{F_s}{I_s} * \frac{F_n}{I_s}$, where F_s, I_s, and F_n denote face size, image size, and face number respectively. Then we sort the test images in ascending order according to their global density level. Finally, the test images are divided uniformly into three sets, sparse, medium and dense. As shown in Table 16, our method achieves the lowest error especially when scenes are sparse. This result highlights the potential complementary nature of the two forms of focus.

4.4.4 *Combined focus for free*

In the aforementioned experiments, we have shown that each focus matters for counting. In this experiment, we combine these two focuses for more accurate counting, in view that these two focuses aid density map estimation respectively from a local and global perspective, complementing each other. The results are shown in Table 17. The combination achieves a reduced count error of 67.9 MAE on ShanghaiTech Part_A, and obtains a reduced MAE of 3.2 on WIDER FACE. We compare our combined approach to an alternative combined attention baseline from Chen *et al.* [24]. While the combination of attentions achieves a better result than using the base network alone, our approach is preferred across datasets, object scales, and crowding levels.

The focus for free is agnostic to the base network and to demonstrate this capability, we have applied our approach to four different base networks. Apart from our base

Table 17: **Effect of combined focus** in terms of MAE on ShanghaiTech Part_A and WIDER FACE. Across dataset, object scale, and crowding level our approach outperforms the base network and a combined spatial and channel attention variant.

	Part_A	**WIDER FACE**						
	overall	small	medium	large	sparse	medium	dense	overall
Base network	74.8	9.2	2.7	2.2	2.1	2.5	9.5	4.7
w/ Spatial- & channel-attention [24]	71.6	8.3	2.0	2.3	1.8	2.6	8.2	4.2
w/ Our combined focus	**67.9**	**7.7**	**1.3**	**0.6**	**0.9**	**1.4**	**7.3**	**3.2**

Table 18: **Focus for free across base networks** on ShanghaiTech Part_A and WIDER FACE. Base network results based on our reimplementations. Regardless of the base network, our combined focus from segmentation and global density lowers the count error.

	Part_A		**WIDER FACE**	
Network from	base	**w/** our focus	base	**w/** our focus
Zhang *et al.* [228]	114.5	**110.1**	7.1	**6.1**
Cao *et al.* [20]	75.2	**72.7**	8.5	**8.2**
Li *et al.* [98]	74.0	**72.4**	4.3	**3.9**
Ours	74.8	**67.9**	4.7	**3.2**

network, we consider the multi-column network of Zhang *et al.* [228], the deep single column network of Li *et al.* [98] and the encoder-decoder network of Cao *et al.* [20]. We have re-implemented these networks and use the same experimental settings as in our base network. The results in Table 18 show that our focus for free lowers the count error for all these networks on ShanghaiTech Part_A and WIDER FACE.

4.4.5 *Non-uniform kernel estimation*

Next, we study the benefit of our proposed kernel for generating more reliable ground-truth density maps. For this experiment, we compare to the Geometry-Adaptive Kernel (GAK) of Zhang *et al.* [228]. For WIDER FACE, the spatial extent of objects is provided by the box annotations and we use this additional information to measure the variance quality of our kernel compared to the baseline. The counting and variance results are shown in Table 22. The proposed kernel has a lower count error than the commonly used GAK on both ShanghaiTech Part_A and WIDER FACE. To show that this improvement is due to the better estimation of the object size of interest, we compare the estimated variances σ obtained by different methods with the ground truth variance obtained by leveraging the box annotations of WIDER FACE. Our kernel reduces the MAE of σ from 2.6 to 2.2 compared to GAK.

4.4.6 *Comparison to the state-of-the-art*

Global count comparison. Table 20 shows the proposed approach outperforms all other models in terms of MAE on all five datasets. The proposed method achieves a new state

Table 19: **Benefit of our kernel** on ShanghaiTech Part_A and WIDER FACE. Our network with our kernel obtains lower count error than with GAK [228], shown in the columns MAE (n). To show that this improvement is due to better object size estimation, we compare our kernel to the ground-truth on WIDER FACE, shown in the column MAE(σ), which indicates a lower size error than with GAK.

	Part_A	**WIDER FACE**	
Kernel from	MAE (n)	MAE (n)	MAE (σ)
GAK [228]	67.9	4.2	2.6
Ours	**65.2**	3.6	**2.2**
Ground-truth	n.a.	**3.2**	n.a.

Table 20: **Comparison to the state-of-the-art for global count error** on ShanghaiTech Part_A, Part_B, TRANCOS, DCC, and WIDER FACE. Results on WIDER FACE based on our reimplementations. Our results set a new state-of-the-art on all five datasets for almost all metrics.

	Part_A				**Part_B**		**TRANCOS**	**DCC**	**WIDER FACE**	
	MAE	RMSE	PSNR	SSIM	MAE	RMSE	MAE	MAE	MAE	NMAE
Zhang *et al.* [228]	110.2	173.2	21.4	0.52	26.4	41.3	-	-	7.1	1.10
Marsden *et al.* [131]	85.7	131.1	-	-	17.7	28.6	9.7	8.4	-	-
Shen *et al.* [165]	75.7	**102.7**	-	-	17.2	27.4	-	-	-	-
Sindagi & Patel [176]	73.6	106.4	21.7	0.72	20.1	30.1	-	-	-	-
Ranjan *et al.* [153]	68.5	116.2	-	-	10.7	16.0	-	-	-	-
Issam *et al.* [76]	-	-	-	-	13.1	-	3.6	-	-	-
Li *et al.* [98]	68.2	115.0	23.8	0.76	10.6	16.0	3.6	-	4.3	0.53
Cao *et al.* [20]	67.0	104.5	-	-	8.4	13.6	-	-	8.5	1.10
Ours	**65.2**	109.4	**25.4**	**0.78**	**7.2**	**12.2**	**2.0**	**3.2**	**3.2**	**0.40**

of the art on ShanghaiTech Part_B, and a competitive result on ShanghaiTech Part_A in terms of RMSE. Shen *et al.* [165] achieve the lowest RMSE on ShanghaiTech Part_A, but their approach is not competitive on Part_B. Moreover, they rely on four networks with a total of 4.8 million parameters, while our proposal just needs a single network with 2.6 million parameters. On TRANCOS our method reduces the count error from 3.6 (by Issam *et al.* [76] and Li *et al.* [98]) to 2.0. A considerable reduction. For the challenging DCC dataset proposed by Marsden *et al.* [131], we predict a more accurate global count without any post-processing, reducing the error rate from 8.4 to 3.2. For WIDER FACE, we evaluate using MAE and a normalized variant (NMAE). For NMAE, we normalize the MAE of each test image by the ground-truth face count. This additional metric is because counts in WIDER FACE vary from 1 to 1965. Again, our method achieves best results on both MAE and NMAE compared to the existing methods.

Local count comparison. Figure 39 shows the results obtained by various methods in terms of the commonly used GAME metric on TRANCOS. The higher the GAME value, the more counting methods are penalized for local count errors. For all GAME settings, our method sets a new state-of-the-art. Furthermore, the difference to other methods increases as the GAME value increases, indicating our method localizes and counts extremely overlapping vehicles more accurately compared to alternatives.

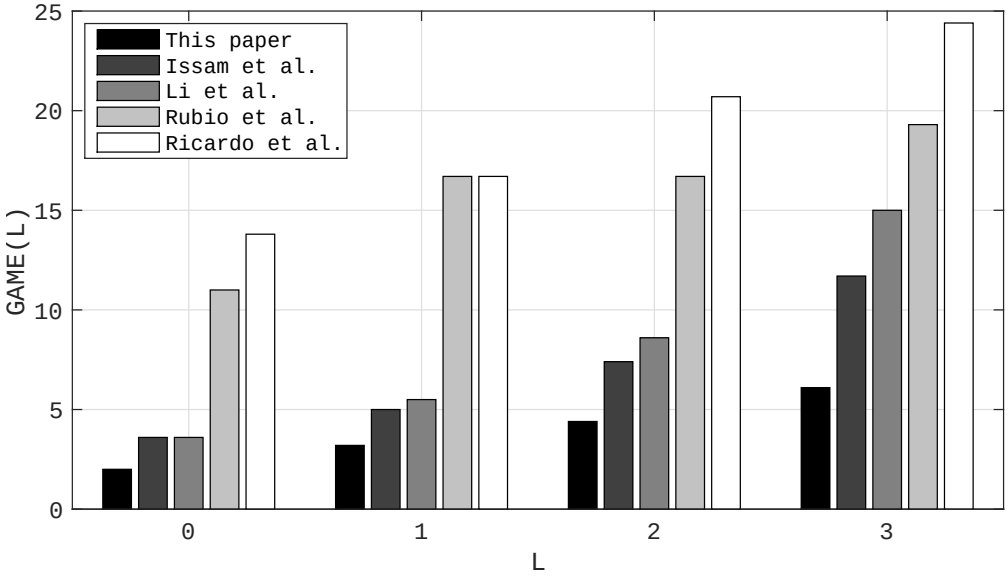


Figure 39: **Comparison to the state-of-the-art for local count error** on TRANCOS. Note that the difference to other methods increases as the GAME value increases, indicating that our method localizes and counts extremely overlapping vehicles more accurately.

Density map quality. To demonstrate that our method also generates better quality density maps, we provide comparative results on ShanghaiTech Part_A in terms of the PSNR and SSIM metrics. In agreement with the results in MAE and RMSE, our proposed method also achieves a better performance along this dimension. Compared to counting methods such as [98], which produces a density map with a reduced resolution and recovers the resolution by bilinear interpolation, our method directly learns the full resolution density maps with higher quality.

Success and failure cases. Finally, we show qualitatively some success and failure results in Figure 40. Even in challenging scenes with relatively sparse small objects or relatively dense large objects, our method is able to achieve an accurate count, as shown in the first two rows of Figure 40. Our approach fails when dealing with extremely dense scenes where individual objects are hard to distinguish, or where objects blend with the context, as shown in the last two rows of Figure 40. Such scenarios remain open challenges in counting, where further focus is required.

4.5 CONCLUSION

This chapter introduces two ways to repurpose the point annotations used as supervision for density-based counting. Focus from segmentation guides the counting network to focus on areas of interest, and focus from global density regularizes the counting network to learn a matching global density. Our focus for free aids density estimation from a local and global perspective, complementing each other. This chapter also introduces a non-uniform kernel estimator. Experiments show the benefits of our proposal across object scales, crowding levels and base networks, resulting in state-of-the-art counting results on five benchmark datasets. The gap towards perfect counting and our qualitative

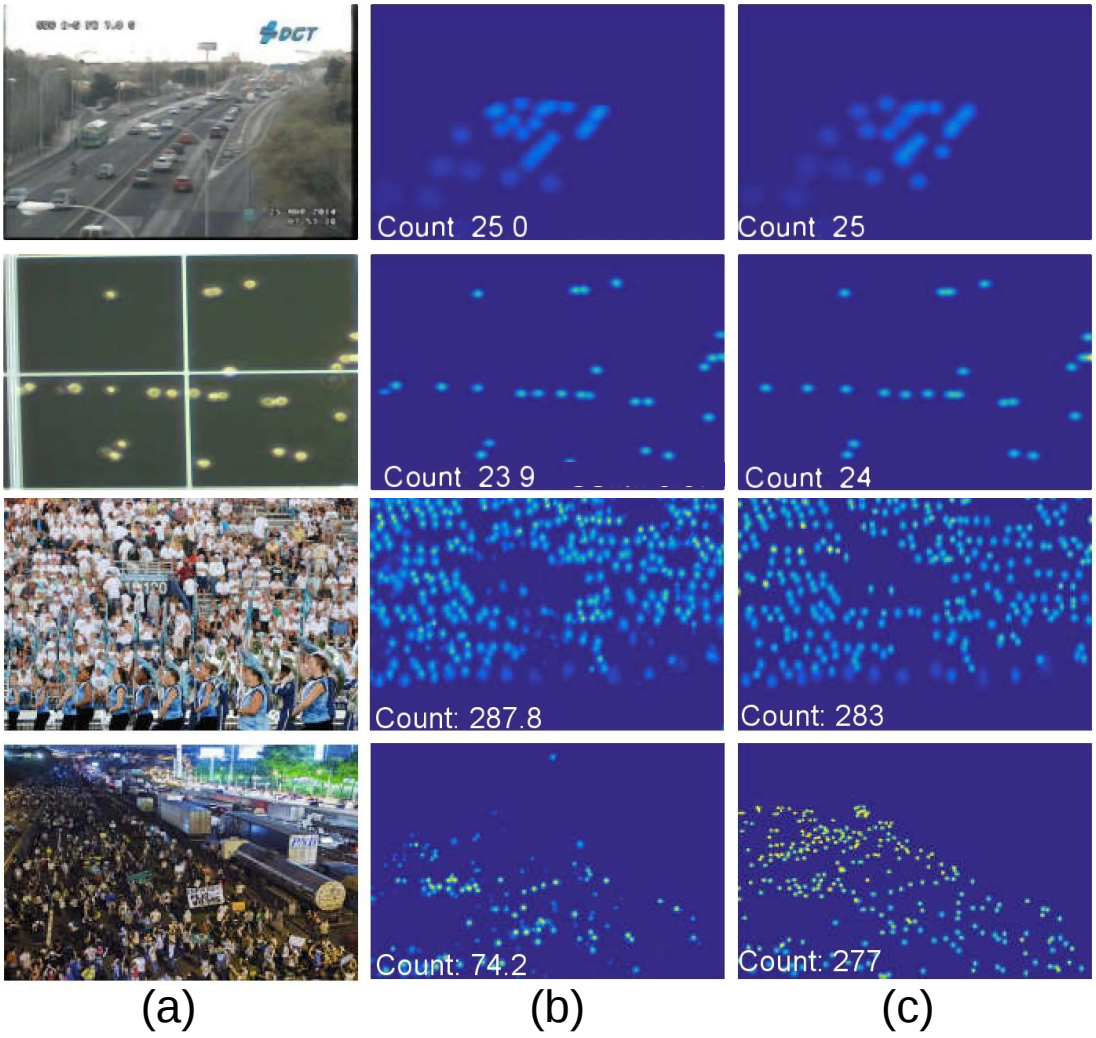


Figure 40: **Density map quality.** (a) Sample images from WIDER FACE, (b) predicted density map, and (c) the ground truth. When objects are individually visible, we can count them accurately. Further improvements are required for dense settings where objects are hard to distinguish from each other and their context.

analysis shows that counting in extremely dense scenes remains an open problem. Further boosts are possible when counting is able to deal with this extreme dense scenario.

4.6 APPENDIX

In this section, we provide the architecture and ablation study of encoder-distiller-decoder network, the benefit of non-uniform kernel estimation across counting networks, and additional qualitative examples of (i) our encoder-distiller-decoder network, (ii) the effect of focus from segmentation, focus from global density and our combined focus.

Encoder-distiller-decoder network. The proposed encoder-distiller-decoder network (Section 3.4) is visualized in Fig. 41, and an ablation study on it is elaborated next.

We perform an ablation study on ShanghaiTech Part_A to analyze the encoder-distiller-decoder network configuration. We vary the architecture by including and excluding the distiller and decoder. When relying on the encoder and distiller only, the predicted

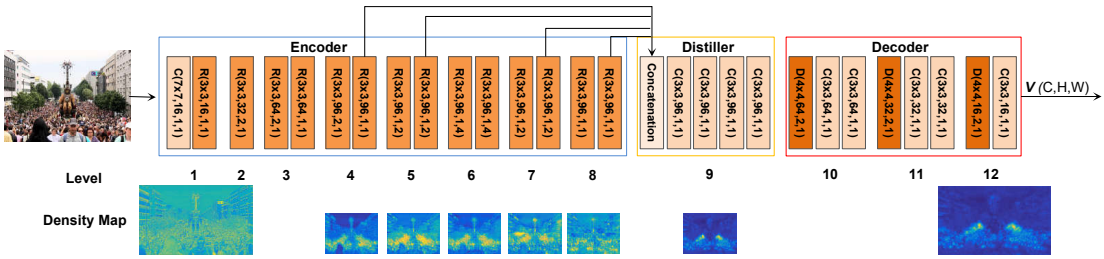


Figure 41: **Encoder-distiller-decoder network**. The network consists of convolution layers (C), residual blocks (R) and deconvolution layers (D) with parameters $(k \times k, c, s, d)$, where $k \times k$ is the kernel size, c is the number of channels, s is the stride size and d is the dilation size. Each convolution layer is followed by a ReLU activation layer and a batch normalization layer. The network is divided into several levels, such that all layers within a level have the same dilation and spatial resolution. The bottom row visualizes the mean feature map from different levels. The distiller module integrates the features from several encoder levels by attending to different parts of the image content for a better overall representation.

Table 21: **Ablation study of encoder-distiller-decoder network** on ShanghaiTech Part_A. Incorporating the proposed distiller module improves the performance of both an encoder-only network and an encoder-decoder network.

Encoder-distiller-decoder			**Metrics**	
Encoder	Distiller	Decoder	MAE	RMSE
✓			114.8	178.2
✓	✓		82.5	140.6
✓		✓	78.8	137.4
✓	✓	✓	**74.8**	**131.0**

density maps are upsampled to full resolution using bilinear interpolation. Results are in Table 21. Adding a distiller module on top of the encoder reduces the MAE from 114.8 to 82.5. The distiller module fuses different features from multiple convolution layers with varying dilation rates, which is beneficial when counting multiple objects which appear in multiple scales in the image. A traditional encoder-decoder network gives a better count than just encoder and an encoder-distiller network. An encoder-only network would compress the target objects to smaller size resulting in loss of fine details. Moreover, it produces density maps with a reduced resolution due to the downsample strides in the convolution operations. The distiller can compete with the decoder to some extent, but it cannot recover the spatial resolution and important details as well as the decoder. Incorporating the distiller in between an encoder and decoder into a single network gives the best counting results on all metrics due to the merits of both scale invariance and detail-preserving density maps. In Fig. 42 we show qualitatively that the network obtains a lower count error and generates higher quality density maps with less noise.

Benefit of non-uniform kernel across counting networks. Next, we study the benefit of our non-uniform kernel estimation for existing counting methods. Apart from our own network, we also evaluate the benefit on two other counting networks, *i.e.* [228]

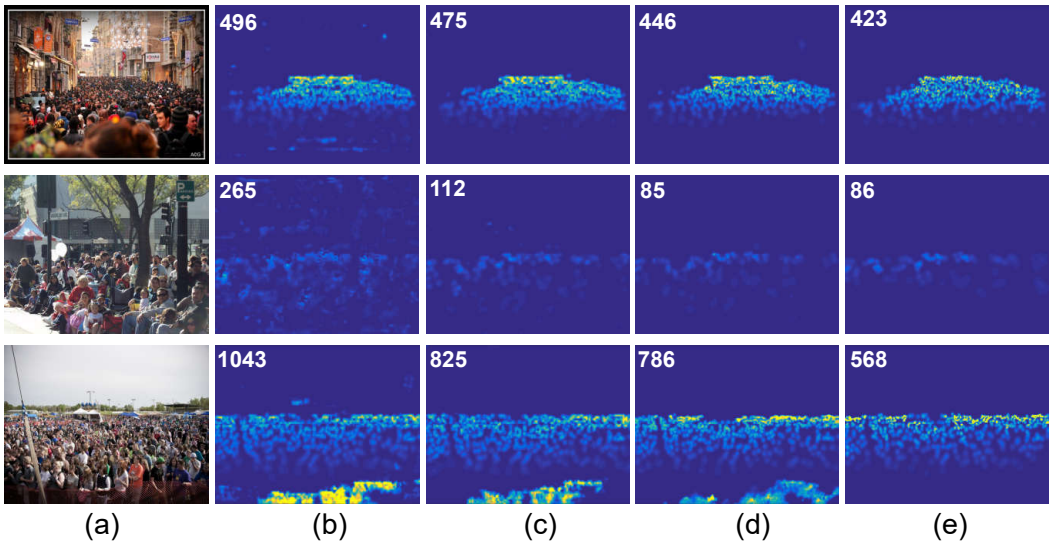


Figure 42: **Ablation study of encoder-distiller-decoder network.** (a) Sample images from ShanghaiTech Part_A and (b) predicted density map by encoder. Effect of (c) encoder-distiller and (d) encoder-distiller-decoder. For comparison, we show the ground truth for each sample in (e).

Table 22: **Benefit of non-uniform kernel estimation** on ShanghaiTech Part_A. Relying on a ground truth density map generated by the proposed kernel, rather than GAK [228], lowers the counting error for our method as well as alternatives.

	Zhang *et al.* [228]		Shi *et al.* [171]		*Ours*	
	MAE	RMSE	MAE	RMSE	MAE	RMSE
GAK [228]	110.2	173.2	73.5	112.3	67.9	115.6
Ours	107.0	156.5	71.7	109.5	**65.2**	**109.4**

and [171], for which code is available. Results in Table 22 demonstrate the proposed kernel has a better MAE and RMSE performance than the commonly used geometry-adaptive kernel [228] for all three networks. It demonstrates our non-uniform kernel is independent of the counting model.

Qualitative results for segment-, density- & combined-focus. To illustrate the beneficial effect of the proposed focuses for reducing the counting error and suppressing background noise, we refer to Fig. 43. As shown in Fig. 43 (c) and Fig. 43 (d) compared to Fig. 43 (b), both segmentation focus and global-density focus show the ability to suppress noise and reduce the counting error. The combination of these two focuses leads to the lowest counting error and higher quality density maps with less noise as shown in Fig. 43 (e).

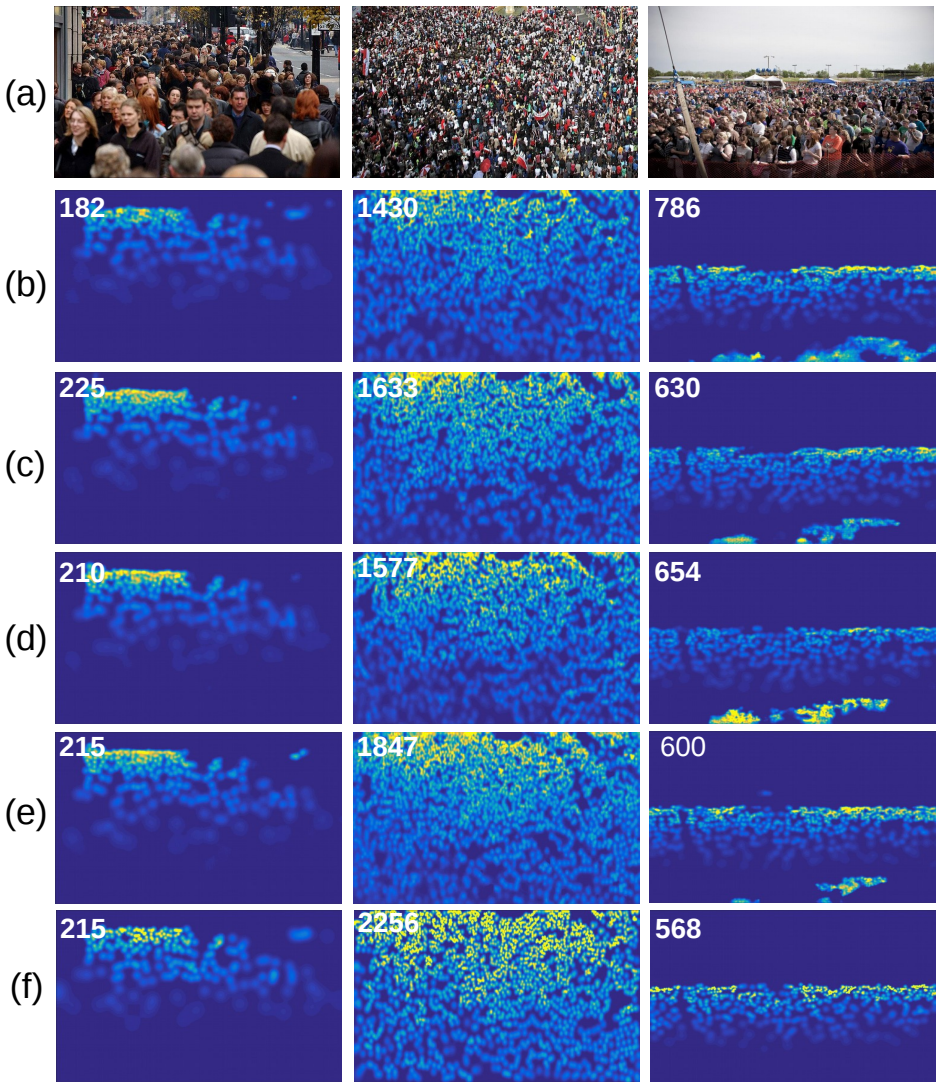


Figure 43: **Effect of segment-, density- & combined-focus** (a) Sample images from ShanghaiTech Part_A and (b) predicted density map without focus. Effect of (c) focus from segmentation, (d) focus from global density, and (e) our combined focus. For comparison, we show the ground truth for each sample in (f).

5

THREE THINGS FOR IMPROVING DENSITY-BASED COUNTING

5.1 INTRODUCTION

This paper considers the problem of counting arbitrary entities in images by learning from density maps, as posed by Lempitsky and Zisserman [94]. For each training image, the entities to count are annotated with points, which are convolved with Gaussian kernels to obtain density maps. By doing so, counting becomes an image-to-image translation problem; from an input image to a continuous density map. Given a trained image-to-image model, the final count for a test image is obtained by summing over all pixel values in the predicted density map. Since its inception, density-based counting has witnessed immense progress, with success achieved through, amongst others, improvements in network architectures [12, 70, 80, 98, 110, 113, 114, 116, 177, 220, 221], improved ways of leveraging auxiliary tasks [10, 115, 167, 170, 210, 229] or data [115, 118, 198], and advances in loss functions [20, 108, 122, 171, 190, 225]. As a direct quantifiable result, the average counting error on the leading ShanghaiTech benchmark [228] has been reduced from 110.2 [228] to 54.6 [195] in just five years.

Despite amazing progress, several works have identified inherent limitations with Gaussian density maps for counting. Examples include ambiguity regarding object size [121, 189, 194] and noise caused by occlusions [12, 205]. Due to the persistent nature of these limitations, recent approaches have even investigated bypassing Gaussian density maps in favor of point-based counting [180, 191, 194]. Motivated by these uncovered limitations and inspired by similar investigations for other computer vision challenges [2], we take a closer look at density-based counting and identify three things that limit every counting approach, and for each we propose a simple network plug-in as a mitigation:

I. Do not count on the background. Through an oracle experiment we find in Section 5.8 that confusion with the background makes up over half the error rate(!) across approaches and datasets. A key towards better counting lies in not counting on background pixels in the first place. To limit the effect of counting on the background, we cascade a segmentation and density network to reduce the hefty effect of background pixels on the counting performance.

II. Create occlusion to handle occlusion. Common in counting is dealing with congested scenes with high levels of occlusion. Occlusions, however, appear not often enough in training imagery to properly learn to handle occlusions. We show in Section 5.8 that occlusions can be simulated through augmentation on both the input image and density maps.

III. Gaussians are not ground-truth. Gaussian convolutions on point annotations were introduced to decrease sparsity and increase spatial robustness. However, as indicated numerous times, Gaussian densities are inherently ambiguous regarding object size and sensitive to occlusions. In Section 5.8 we propose to distill learned density maps from an initial counting network trained on images with blacked-out backgrounds. The discovered maps from distillation are smoother and more noise-robust than their Gaussian counterparts, and can be used in any counting network.

For each finding, the proposed solution is simple, yet provides considerable reduction in counting error. The findings are generic, can be plugged into any existing approach, and are complimentary. Upon combining our proposals with the canonical counting framework HRNet [22, 196] in Section 5.6, we obtain state-of-the-art results on ShanghaiTech Part_A and Part_B, JHU-Crowd++, and TRANCOS, with the first MAE scores below 50 on ShanghaiTech Part A.

Before elaborating on the three things that everyone working on density-based counting should know, we first discuss in Section 5.2 the evaluation, datasets, and networks used during our studies. We will release all code and plug-in modules to the counting community.

5.2 EVALUATION, DATASETS, AND NETWORKS

Evaluation. The counting performance of density-based approaches is evaluated on standard and publicly available image datasets. We report the standardized Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) metrics given count estimates and their ground-truth.

Datasets. We consider five counting datasets in this paper, as also commonly used in the recent literature, *e.g.*, [12, 116, 121, 180, 191, 195]. ***ShanghaiTech*** [228] consists of 1,198 images with 330,165 pedestrians. This dataset is divided into two parts: *Part_A* with 482 images in which crowds are mostly dense, and *Part_B* with 716 images, where crowds are sparser. Each part is divided into a training and testing subset as specified in [228]. ***UCF-QNRF*** [72] consists of 1,535 images, with the count ranging from 49 to 12,865. For training 1,201 images are available, the remaining 334 form the test set. ***JHU-CROWD++*** [178] consists of 4,372 images with a total of 1.51 million point annotations. The dataset is split into a training set of 2,272 images, a validation set of 500 images and a testing set of 1,600 images. ***TRANCOS*** [54] contains 1,244 images from different roads to count vehicles, varying from 9 to 105. The dataset is split into a training set of 403 images, a validation set of 420 images and a testing set of 421 images. For all datasets, we augment the images by randomly performing horizontal flipping, and randomly cropping 256×256 patches for ShanghaiTech, JHU-CROWD++ and TRANCOS and 416×416 patches for UCF-QNRF. For ground-truth density map generation, we follow the previously suggested dataset settings from [72, 170, 178, 228].

Networks. We follow the standard in density-based counting by deep convolutional networks. As our two baseline networks, we use CSRNet [98] and HRNet [196]. CSRNet is the most widely-used architecture in the literature, *e.g.*, [121, 122, 190, 191, 194, 229]. HRNet has very recently shown excellent counting ability [22]. Following [12], our implementation of CSRNet has batch normalization layers, which perform better than

	ShanghaiTech Part_A		ShanghaiTech Part_B		UCF_QNRF		JHU-CROWD+	
	CSRNet	*HRNet*	*CSRNet*	*HRNet*	*CSRNet*	*HRNet*	*CSRNet*	*HRNet*
Standard	61.9	59.7	12.3	10.7	93.6	90.4	72.4	70.9
w/o background	29.8	28.3	3.1	2.8	57.0	55.4	26.7	25.9

Table 23: **Impact of background interference.** Removing the background pixels with an oracle before density prediction reduces the count error (MAE) considerably across datasets and networks.

the original CSRNet in [98]. Our implementation of HRNet is based on HRNetv2-W48. Both networks are initialized by their ImageNet pre-trained models and trained using Adam with batches of 10. By default, we use an ℓ_1 loss function. The learning rate is fixed to 1e-4.

5.3 DO NOT COUNT ON THE BACKGROUND

Density-based approaches for counting learn a density prediction network by optimizing a loss function $\mathcal{L}{=}\ell_p(f(I)-D)$, where f the network, I the input image, D the ground-truth density map and ℓ_p the Lp norm loss function. Not every pixel in this loss function, however, should be treated equally. We divide the density map into two parts, one corresponding to the background of the input image and the other one corresponding to the foreground of the input image with the actual objects to count. The loss function then becomes: $\mathcal{L}{=}\ell_p(f(I)_{bg}) + \ell_p(f(I)_{fg} - D_{fg})$. Naturally, D_{bg} is omitted as every single element in D_{bg} is zero. Based on this reformulation, we analyze two problems of current approaches. First, they are likely to mistakenly predict the background as objects (densities) when $\ell_p(f(I)_{bg})$ is not optimized perfectly. Second, they cannot exploit all their capacity for learning to transform objects to densities because they predict densities ($f(I)_{fg}$) not only on the objects, but also on the background. Our hypothesis is therefore straightforward: density-based counting should only optimize for non-zero density regions, the background should be ignored from the start.

Since the ground-truth density map is generated based on annotated points on objects, it is invariant to the background of the input image. From this point of view, the background of the input image is not useful for density prediction. If we could remove all background pixels from the input image, $f(I)_{bg}$ would naturally be zero. Then the loss function becomes: $\mathcal{L}{=}\ell_p(f(\tilde{I})_{fg} - D_{fg})$ with $\tilde{I}$ the image with zeros for background pixels. With this loss the network fits densities only on the objects. Here, we seek to investigate two questions: *(i)* how much can we gain in counting with perfect awareness of foreground and background in images and *(ii)* how can we learn to reduce the background for density-based counting networks?

Impact of background interference. We first quantify how much the background hinders counting performance. We do so by generating oracle segmentation maps based on the ground-truth density map; all non-zero density pixels are set to 1, the rest to 0. We then mask the input image based on the oracle segmentation map. On this masked image, we predict the density map with a baseline counting network. We emphasize that

	PartA			UCF_QNRF		
	BG	*FG*	*Overall*	*BG*	*FG*	*Overall*
Standard	3.4	61.5	61.9	8.7	91.3	93.6
w/ background reduction	3.0	55.3	56.1	3.4	83.6	84.4

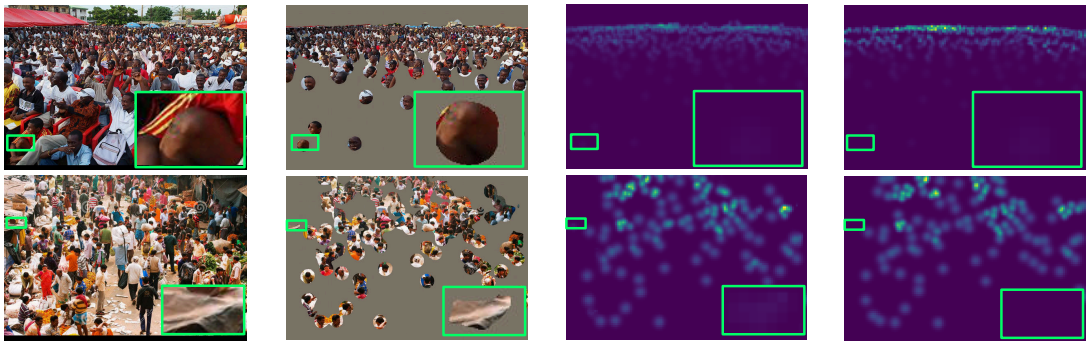
Table 24: **Learning to reduce the background.** Compared to a standard CSRNet base network, which predicts density maps on the background, our simple cascade model with background reduction reduces the overall count error (MAE), for both background (BG) and foreground (FG). Same conclusion for an HRNet, see supplementary materials.

the original input image and its background-free equivalent share the same ground-truth density map.

In Table 23, we provide results based on two counting networks and four datasets. The results across all experiments are consistent: counting without background reduces the MAE considerably. On ShanghaiTech PartA, CSRNet reduces the MAE from 61.9 to 29.8 simply by predicting the density map not on the background; a number far lower than the current state-of-the-art on this dataset. Depending on the dataset and network, background interference is responsible for most of the counting error. From this oracle experiment, we conclude that density-based counting can be significantly improved by avoiding predicting densities on the background.

Learning to reduce the background. Motivated by the oracle experiment, we propose a simple approach toward counting by learning to reduce the impact of background pixels from the start. Specifically, we predict a segmentation map with a model such as DeepLab v3+ [26] or HRNet [196]. Our counting model then cascades a segmentation network and a density prediction network. Let I be the input image, $\hat{D}$ be the predicted density map, $\hat{S}$ be the predicted segmentation map, f_s be the segmentation network, f_d be the density network. Then $\hat{S}=f_s(I)$, and $\hat{D}=f_d(I\odot\hat{S})$ where $\odot$ denotes the element-wise product. We first train the segmentation network with a cross-entropy loss and then fix it to train the density network with a ℓ_1 loss. Throughout this paper, we use HRNetv2-W48 [196] as the segmentation network. We show the effect of learning to reduce the background through cascading in Table 29. On ShanghaiTech PartA our approach provides a reduction of 5.8 in the overall MAE. On UCF QNRF, the overall MAE even goes down by 9.2. Even though our proposed cascade is simple, the direct effect for counting is evident, emphasizing the importance of removing the background in the input image for counting. We find that a learned background reduction improves both background and foreground errors, with a special focus on the foreground error. We also provide qualitative results in Figure 48. We observe that our predicted segmentation masks remove many background pixels and the density prediction network is able to count well on the remaining pixels. Overall, we recommend learning to remove the background from the input image for density-based counting. The gap towards oracle performance does highlight that big gains should be feasible with more advanced image segmentation tactics.

Related work. Several works have investigated count errors that come from the background, *e.g.*, [139, 170, 229]. Zhao *et al.* [229] handle the cluttered background with an auxiliary segmentation loss. Shi *et al.* [170] and Modolo *et al.* [139] reduce the



(a) **Input image** (b) **Cascaded image** (c) **Predicted density** (d) **Ground-truth**

Figure 44: **Learning to reduce the background.** Segmentation and density prediction results by our cascade model. While by no means perfect, the predicted segmentation masks already reduce the background pixels considerably. The density prediction network is then able to count correctly on the remaining amount of background pixels, as masked in the green regions.

background error by learning a segmentation attention map for the output density map. However, these approaches still predict the density map on the background. Differently, our approach learns a segmentation mask to reduce the background from the input image and then predicts the density map on a background-reduced input image. Cheng *et al.* [29] also propose a two-stage counting approach. They first regress a probability map to identify possible object regions, and then predict a density map on the probability map to count the objects. The probability map does not contain information about the occluded objects in the input image, however, resulting in underestimation. Differently, we predict the density map based on the background-reduced input image, which hardly misses information about the occluded objects. To emphasize that the key is to remove the background from the input image altogether, we have performed additional experiments where we employ the recent approaches of Modolo *et al.* [139] and Cheng *et al.* [29] on top of the same counting network from Table 29. While both improve over the base network, neither work as well as our simple cascade (PartA: 59.3 [139], 57.5 [29], and 56.1 for ours - UCF QNRF: 89.4 [139], 85.6 [29], and 84.4 for ours). The full comparison is provided in the supplementary material.

5.4 CREATE OCCLUSION TO HANDLE OCCLUSION

Another key challenge for counting is occlusion, as it often happens in congested scenes. For counting approaches, one possible solution to handle various occlusion levels is to learn a network with sufficient number of training samples for each occlusion level. However, the occlusion level presents a long-tailed distribution in all benchmarks [54, 72, 178, 228]. In other words, there are simply not enough images in current datasets to properly learn from natural occlusions. As a remedy, we propose to simulate various occlusion levels through augmentation.

Occlusion simulation. Given a training sample $(I, \{P_i\}_{i=1}^N)$, with I the input image and $\{P_i\}_{i=1}^N$ a set of 2D point annotations, one for each object, we create the ground-truth density map D by convolving each annotated point with a Gaussian kernel. Then an

object O_i in the image I can be represented by $(x_i, y_i, 2\sigma_i)$ where x_i, y_i is the coordinate of the corresponding annotated point P_i, and σ_i is the standard deviation of the corresponding Gaussian kernel. Here, we assume that the object O_i is circular, and its radius is approximately equal to $2\sigma_i$ following [94]. We randomly select an object which will be occluded, denoted by O_{occ}, and then we randomly select one of its neighboring objects. This object denoted by O_{copy} will be copied and pasted to a specific position to occlude the object O_{occ}. The pasted position of O_{copy} decides how the object O_{occ} will be occluded, defined as:

$$\begin{aligned} x_{paste} &= \lfloor r \cdot cos(\theta) \rfloor + x_{occ}, \\ y_{paste} &= \lfloor r \cdot sin(\theta) \rfloor + y_{occ}, \end{aligned} \tag{5.1}$$

where $r{=}2\sigma_{copy} + 2\sigma_{occ}\epsilon_r$ and $\theta{=}2\pi\epsilon_\theta$. ϵ_r and ϵ_θ are randomly sampled from $\mathcal{U}(0,1)$. $\lfloor \cdot \rfloor$ is the floor operator. r and θ decide how much and where O_{occ} will be occluded.

Blending. Directly pasting objects on an image creates boundary artifacts, which may affect the network's learning ability. To handle boundary artifacts, we perform blending to smooth out the boundary artifacts. Specifically, we first copy and paste the object O_{copy} to the position (x_{paste}, y_{paste}) of the image I, generating a new image $\hat{I}$. Then, we compute the binary mask α of the pasted object O_{paste}, which can be represented by $(x_{paste}, y_{paste}, 2\sigma_{paste})$ where $\sigma_{paste}{=}\sigma_{copy}$. To smooth out the edges of the pasted object, we apply a Gaussian filter to the binary mask α and obtain a smoothed mask $\tilde{\alpha}$. Then we use $\tilde{\alpha}$ to construct the occluded image I_{occ} and its corresponding ground-truth density map:

$$\begin{aligned} I_{occ} &= (1 - \tilde{\alpha}) \odot I + \tilde{\alpha} \odot \hat{I}, \\ D_{occ} &= D + G(O_{paste}), \end{aligned} \tag{5.2}$$

where D denotes the ground-truth density map of the image I, $G(O_{paste})$ denotes the density map of the pasted object O_{paste}, and G denotes the Gaussian kernel. Equation 5.2 states that the pixels to be occluded in the original image are replaced by the pixels of the pasted object to obtain the occluded image. The corresponding occluded density map is obtained by adding the density map of the pasted object to the original density map.

Occluding adaptively. For a training image, we need to decide how many of its objects will be occluded with our approach. A simple way is to set a fixed ratio. However, the training image may already have naturally occluded objects. Thus, we should create new occlusion for the training image according to its current occlusion level. Intuitively, the lower the occlusion level, the higher the amount of occlusions that should be simulated. To compute the occlusion level, we create an occlusion map M for the training image by,

$$\begin{aligned} M(x,y) &= \sum_{i=1}^{N} S(x,y;O_i), \text{where} \\ S(x,y;O_i) &= \mathbb{1}(\|x - x_i\|^2 + \|y - y_i\|^2 \leq (2\sigma_i)^2). \end{aligned} \tag{5.3}$$

Here (x,y) denotes a pixel location, $O_i{=}(x_i, y_i, 2\sigma_i)$ denotes an object, and $\mathbb{1}(\cdot)$ is a binary indicator function, which states that a pixel obtains a value of one if it is within an object region. Hence, each pixel value in the occlusion map M indicates the amount of object regions it belongs to. The occlusion level $\bar{M}$ is computed as the average value

	PartA			UCF_QNRF		
	Low	*High*	*Overall*	*Low*	*High*	*Overall*
Standard	39.2	95.9	61.9	18.7	106.5	93.6
w/ occlusion creation	37.8	91.2	59.1	18.3	99.3	87.4

Table 25: **Create occlusion to handle occlusion.** Compared to a standard CSRNet base network, our occlusion augmentation reduces the count error (MAE) for the entire test set, the *low* occlusion set, and especially for the *high* occlusion set. Same conclusion for an HRNet, see supplementary materials.

of non-zero pixels in the occlusion map M. Then the percentage of the objects to be occluded can be adaptively obtained by $\beta/\bar{M}$ where β is the upper boundary. We have empirically found that β=0.3 performs well.

Analysis. To demonstrate the potential of our simple occlusion handling augmentation, we apply a similar setup as the previous Section. To highlight effectiveness in highly occluded images, we divide the test images into two sets according to their occlusion level, as computed by Eq. 5.3. The test images are grouped into the *low* occlusion set if their occlusion level is lower than 1.5, the remaining images are grouped into the *high* occlusion set. For ShanghaiTech PartA, we obtain 109 low and 73 high occlusion images and for UCF_QNSF, we obtain 49 low and 285 high occlusion images. The results are shown in Table 30. Compared to the base counting network, our simple occlusion augmentation reduces the MAE on both datasets and the occlusion subsets. Especially when occlusion levels are high, the occlusion augmentation is effective. This experiment solidifies our point: occlusions weight heavily on the counting error, and there are not enough natural occlusions to learn from. Instead, by simulating occlusions, we can easily reduce the MAE by 2.8 (PartA) and 6.2 (UCF QNRF).

The simulated occlusions take the form of an augmentation approach, which can be employed by any counting approach. To highlight that the key here is occlusions, not just the augmentation, we have additionally investigated the effect of adding well-known image augmentation approaches such as Cutout [38] and CutMix [218]. On both datasets, such augmentations also reduce the counting error, but cannot compete with our occlusion simulation (PartA: 61.2 [38], 60.6 [218], and 59.1 for ours - UCF_QNRF: 91.8 [38], 90.6 [218], and 87.4 for ours). The full comparison is in the supplementary materials. We conclude that to handle occlusions in density-based counting, it pays off to create the occlusions by augmentation.

5.5 GAUSSIANS ARE NOT GROUND-TRUTH

At the core of density-based counting approaches are the Gaussian density maps. These are obtained by spatially convolving point annotations with Gaussian kernels and then using the obtained density map as the learning target. The Gaussian density targets make the optimization less sparse and increase spatial robustness. However, these maps are imperfect and tend to be sensitive to occlusion and perspective distortions. Occlusions between objects result in overlapping Gaussian blobs. By design, the overlapping regions of the Gaussian densities are accumulated, even though this accumulation is no

(a) **Original image** (b) **Our image** (c) **Original density** (d) **Our density**

Figure 45: **Occlusion creation**. Illustrative examples for creating new occlusions on images and corresponding density maps. Our occlusion-augmented image and density look quite natural, as masked in the green regions.

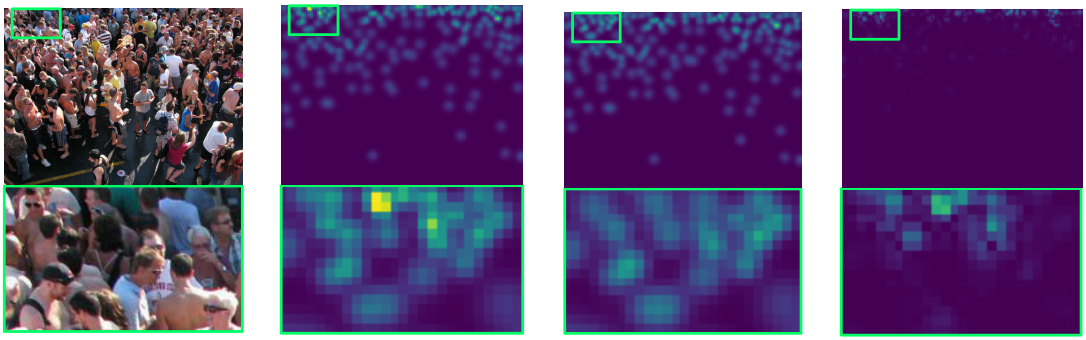
longer visible in the image itself, resulting in noisy local maxima [12, 205]. Moreover, perspective distortions result in large variations in object size. Since we are only given points and no ground-truth information about local object size, there is no direct way to correctly match Gaussian densities to all objects [170, 189, 228]. We therefore seek to go beyond Gaussian density maps as the final ground-truth for counting.

Density maps from distillation. We have a simple proposal to improve the Gaussian density maps that we optimize on: we learn new density maps from the original Gaussian density maps through distillation. According to [4], deep networks are capable of memorizing noisy data, but learn simple patterns first. Inspired by this observation, we propose to learn an auxiliary network to capture the mapping patterns between objects and densities, and avoid memorizing the noisy densities. Since the auxiliary network focuses on capturing the mapping patterns between objects and densities, we train the auxiliary network on images with background areas blacked-out, as recommended in Section 5.8. The training loss is $\mathcal{L}_a{=}\ell_1(f_a(\tilde{I}) - D)$, where f_a the auxiliary network, $\tilde{I}$ the image, and D the original Gaussian density map. The optimization of the auxiliary network will be stopped when it achieves maximum accuracy on the validation set. Then we distill the knowledge of the trained auxiliary network to the density prediction network with the loss $\mathcal{L}_d{=}\ell_1(f_d(I) - f_a(\tilde{I}))$. Different from the standard distillation [66] that aims for compression, our distillation seeks to improve the ground-truth density map. Thus, our distillation loss is not combined with the ground-truth loss. In fact, we found combining with the ground-truth loss degrades the counting performance.

In Table 31, we show the effect of counting on distilled density maps over Gaussian density maps. We use the same base network for auxiliary distillation and density prediction. On ShanghaiTech PartA our approach provides a reduction of 5.2 in MAE. On UCF QNRF, the MAE goes down by 7.1. This simple distillation is an easy schema that can operate on any network and provides a direct reduction in counting error. In Figure 50, we provide some examples of Gaussian and distilled density maps. We observe that distilled densities better match the observed objects, especially the occluded objects in the image, compared to the Gaussian densities. Thus, distillation provides a more natural relation between images and density maps.

	PartA	UCF_QNRF
Gaussian density map	61.9	93.6
Distilled density map	56.7	86.5

Table 26: **Effect of our distillation** in terms of MAE on ShanghaiTech Part_A and UCF_QNRF. Compared to Gaussian density, our distilled density reduces the count error (MAE) considerably. All results based on CSRNet base network. Same conclusion for an HRNet, see supplementary materials.



(a) **Training image** (b) **Gaussian density** (c) **Distilled density** (d) **Density difference**

Figure 46: **Density generation with distillation**. Illustrative examples of Gaussian and distilled density maps. Compared to Gaussian densities, distilled densities reduce the local noisy maxima (yellow color region) and better match the objects in the image, especially the occluded objects. Also, the density difference shows that distilled densities changes Gaussian densities mostly from the occluded region.

Related work. Several works have noted limitations of Gaussian densities for counting, *e.g.* [72, 122, 170, 189, 205, 228]. Rather than using fixed and uniform Gaussian densities [94], numerous attempts have been made to improve the Gaussian densities. Zhang *et al.* [228] and Shi *et al.* [170] compute locally adaptive Gaussians as a function of the K nearest neighbor annotations. Xu *et al.* [205] address the Gaussian scale problem by rescaling the Gaussian densities into similar scale levels. Wan *et al.* [189] learn an attention-based network to adaptively fuse multi-scale Gaussian densities. Bai *et al.* [12] introduce a self-correction approach to improve Gaussian density. Rather than adapting the Gaussian kernels to construct density maps, we propose to move beyond Gaussian densities through distillation. To highlight the benefits of improving a density map with distillation, we have performed additional experiments where we employ the recent approach of Wan *et al.* [189] on top of the same counting network from Table 31. While this approach also improves over the base network, it does not work as well as our simple distillation (PartA: 59.6 [189], and 56.7 for ours - UCF QNRF: 91.1 [189], and 86.5 for ours). The full comparison is provided in the supplementary materials. Lastly, we note that our approach is inherently different from the counting distillation approach of Liu *et al.* [107]. Their distillation seeks to compress the network, while we seek to obtain a more desirable density maps.

5.6 COMPARATIVE EVALUATION

For the final experiments, we first show the effect of combining the three solutions we propose for dealing with the three error sources together. Then we compare to the state-of-the-art in counting.

Combining the three solutions. So far, we have shown that each solution matters for counting by itself. In this experiment, we evaluate whether they are also complementary. The results are shown in Table 27 using both CSRNet and HRNet as backbones. Compared to standard counting with just the base networks, we reduce the MAE considerably by reducing background inference with cascading (Section 5.8). By combining background reduction with ground-truth distillation (5.8) or occlusion creation (Section 5.8)), the MAE is further reduced. Combining all three solutions achieves the lowest count error across datasets and base networks.

Comparison to the state-of-the-art. As the ultimate test, we draw a comparison to the state-of-the-art in counting on five datasets. We report our results using both CSRNet [98] and HRNet [196] as backbones. Table 28 shows the comparative evaluation over all datasets and metrics. The results show that despite using canonical counting backbones, which on their own can not compete with the state-of-the-art, we obtain the best density-based counting results on 8 of the 10 metrics. This highlights the potential of our simple solutions. We also compare to recent point-based counting alternatives and find that we maintain the best results for 6 of the 10 metrics. On ShanghaiTech Part_A, we score below 50 in terms of MAE for the first time. The results on TRANCOS (1.4 MAE versus 2.6 for Bai *et al.* [12] and 2.1 RMSE versus 3.9) also indicate that our approach is not limited to person-centric counting and generalizes to vehicle counting. We conclude that with our proposed solutions, we can make density-based optimization the most effective way to count again. We show some success and failure results in Figure 47. Even in challenging scenes with relatively sparse small objects or relatively

Table 27: **Effect of combining our solutions for the three error sources** in terms of MAE on ShanghaiTech Part_A and UCF_QNRF. With each new addition, the counting error decreases, showing their complementary nature.

	Three solutions for error sources				
	Background	Ground-truth	Occlusion	**Part_A**	**UCF_QNRF**
CSRNet					
				61.9	93.6
	✓			56.1	84.4
	✓	✓		54.7	82.2
	✓		✓	54.3	81.6
	✓	✓	✓	**52.7**	**80.1**
HRNet					
				59.7	90.4
	✓			55.4	83.6
	✓	✓		53.4	81.5
	✓		✓	52.1	80.7
	✓	✓	✓	**49.3**	**78.4**

Table 28: **Comparison to the state-of-the-art** on ShanghaiTech Part_A, Part_B, UCF_QNRF, JHU-CROWD++ and TRANCOS. We combine our three simple solutions with CSRNet [98] and HRNet [196] to obtain counting results which are either competitive or better than the current state-of-the-art, highlighting their potential for density-based counting. **Bold** indicates the best density-based counting results, and underline indicates the best overall counting approach.

	Part_A		**Part_B**		**UCF_QNRF**		**JHU-CROWD++**		**TRANCOS**	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
Density-based counting										
Wan *et al.* [192]	63.8	99.2	7.8	12.7	99.5	173.0	69.7	268.3	-	-
Wan *et al.* [190]	61.9	99.6	7.4	11.3	85.8	150.6	67.7	258.5	-	-
Jiang *et al.* [81]	57.8	90.1	-	-	91.6	159.7	-	-	-	-
Hu *et al.* [70]	56.7	93.4	6.7	**10.2**	101.8	163.2	-	-	-	-
Liu *et al.* [109]	55.9	97.1	7.3	11.3	88.1	143.7	-	-	-	-
Cheng *et al.* [29]	57.2	93.0	6.3	10.7	81.7	137.9	-	-	-	-
Bai *et al.* [12]	55.4	97.7	6.4	11.3	**71.3**	132.5	-	-	2.6	3.9
Wang *et al.* [195]	54.6	91.2	6.4	10.9	81.1	131.7	-	-	-	-
Ma *et al.* [121]	52.9	87.3	-	-	79.5	140.7	57.4	**251.6**	-	-
CSRNet [98] + ***three solutions***	52.7	86.4	6.9	11.5	80.1	129.6	59.4	253.7	2.1	3.4
HRNet [196] + ***three solutions***	**49.3**	**83.7**	**6.2**	10.6	78.4	**127.3**	**57.1**	254.3	**1.4**	**2.1**
Point-based counting										
Wang *et al.* [194]	59.7	95.7	7.4	11.8	85.6	148.3	68.4	283.3	-	-
Wan *et al.* [191]	61.3	95.4	7.3	11.7	84.3	147.5	59.9	259.5	-	-
Liu *et al.* [116]	55.4	91.3	6.9	10.3	76.2	121.5	59.9	259.5	-	-
Song *et al.* [180]	52.7	85.1	6.3	9.9	85.3	154.5	-	-	-	-

dense large objects, our method is able to achieve an accurate count (first three rows). Our approach fails when dealing with extremely dense scenes where individual objects are hard to distinguish, or where objects blend with the context (last two rows). Such scenarios remain open counting challenges.

Complexity analysis. The distilled density maps and occlusion simulations do not change inference time. Only the segmentation cascade adds complexity and increases the inference time, *e.g.*, from 0.27s to 0.53s per image for the UCF_QNRF dataset.

5.7 CONCLUSION

This chapter dives into density-based counting, the most widely-used approach for counting entities in images. Motivated by recent work that identifies limitations in counting with Gaussian density maps, this chapter seeks to uncover which persistent issues are common and shared amongst all current approaches. We take a closer look at three error sources, including *background*, *ground-truth*, and *occlusion*, that should be aware of in density-based counting. For each error source, we provide a simple proposal to tackle them. We evaluate our three solutions over a number of datasets (ShanghaiTech Part_A and Part_B, UCF_QNRF, JHU-Crowd++, and TRANCOS) and demonstrate substantial improvements in counting performance. Upon integrating them in canonical counting networks, we obtain state-of-the-art counting performance, highlighting their potential as general tools to use in density-based counting.

(a) **Input image** (b) **Cascaded image** (c) **predicted density** (d) **Ground-truth**

Figure 47: **Success and failure cases.** Illustrative examples of success and failure cases. When objects are individually visible, we can count them accurately (first two rows). Further improvements are required for extremely dense scenes where individual objects are hard to distinguish, or where objects blend with the context (last two rows).

5.8 APPENDIX

This supplementary file provides additional quantitative and qualitative results for: 1) *do not count on the background*, 2) *create occlusion to handle occlusion*, and 3) *Gaussians are not ground-truth*.

Do not count on the background We provide additional results on multiple datasets with baselines to show the effect of learning to reduce the background through cascading in Table 29, which is a supplement to the Table 2 in the main paper. We compare our approach with the segmentation attention approach [139] and the two-stage approach [29]. For fair comparison, we implement our approach and the compared approaches on top of CSRNet and HRNet. Compared to the base networks, all three approaches further reduce the background, foreground, and overall count error in terms of MAE. Our approach performs better than the segmentation attention approach and the two-stage approach. We also provide more qualitative results in Figure 48.

Create occlusion to handle occlusion Table 30 is a supplement to the Table 3 in the main paper, which shows the effect of creating occlusion to handle occlusion. We compare our approach with Cutout [38] and CutMix [218]. Compared to these approaches, our occlusion augmentation reduces the overall count error (MAE) the most, the *low* occlusion set, and especially for the *high* occlusion set. Cutout randomly zero-outs a region in the training images and their corresponding ground-truth labels. Instead of simply removing pixels, CutMix replaces the removed regions with a patch

	CSRNet						HRNet					
	PartA			UCF_QNRF			PartA			UCF_QNRF		
	BG	*FG*	*Overall*	*BG*	*FG*	*Overall*	*BG*	*FG*	*Overall*	*BG*	*FG*	*Overall*
Base network	3.4	61.5	61.9	8.7	91.3	93.6	3.2	59.1	59.7	8.2	89.9	90.4
w/ segmentation attention [139]†	**2.9**	58.9	59.3	4.5	88.3	89.4	**2.8**	57.9	58.3	4.3	88.1	88.6
w/ two-stage [29]†	3.1	57.1	57.5	3.5	84.8	85.6	2.9	55.8	56.7	**4.1**	83.6	84.2
w/ background reduction (*ours*)	3.0	**55.3**	**56.1**	**3.4**	**83.6**	**84.4**	2.9	**54.2**	**55.4**	4.8	**82.7**	**83.6**

†Results based on our reimplementation.

Table 29: **Learning to reduce the background** on ShanghaiTech Part_A and UCF_QNSF. Compared to the baselines, which predicts density maps on the background, our simple cascade model with background reduction reduces the overall count error (MAE), for both background (BG) and foreground (FG).

	CSRNet						HRNet					
	PartA			UCF_QNRF			PartA			UCF_QNRF		
	Low	*High*	*Overall*	*Low*	*High*	*Overall*	*Low*	*High*	*Overall*	*Low*	*High*	*Overall*
Base network	39.2	95.9	61.9	18.7	106.5	93.6	36.5	94.3	59.7	17.2	103.0	90.4
w/ Cutout [38]†	38.6	95.0	61.2	**16.5**	104.7	91.8	35.6	93.5	58.8	**15.4**	101.5	88.5
w/ CutMix [218]†	38.0	94.3	60.6	17.8	103.1	90.6	35.2	92.8	58.3	16.3	99.8	87.6
w/ CutOcc (*ours*)	**37.8**	**91.2**	**59.1**	18.3	**99.3**	**87.4**	**34.7**	**90.5**	**57.1**	16.8	**98.1**	**86.2**

†Results based on our reimplementation.

Table 30: **Create occlusion to handle occlusion** on ShanghaiTech Part_A and UCF_QNSF. Compared to the baselines, our occlusion augmentation better reduces the count error (MAE) for the entire test set, the *low* occlusion set, and especially for the *high* occlusion set.

	CSRNet		HRNet	
	Part_A	*UCF_QNRF*	*Part_A*	*UCF_QNRF*
Gaussian density map	61.9	93.6	59.7	90.4
Multiscale Gaussian density map [189]†	59.6	91.1	58.1	88.2
Distillated density map (*ours*)	**56.7**	**86.5**	**55.9**	**85.6**

†Results based on our reimplementation.

Table 31: **Effect of our distillation** in terms of MAE on ShanghaiTech Part_A and UCF_QNSF. Compared to Gaussian density and multiscale Gaussian density, our distilled density reduces the count error (MAE) considerably.

from another image. Cutout and CutMix create new training samples, but they do not create new occluded samples. Differently, our approach creates a variety of occluded samples by explicitly simulating the occlusion in real world. Therefore, our approach better handles occlusion. We also provide more examples of our created occluded images and corresponding density maps in Figure 49.

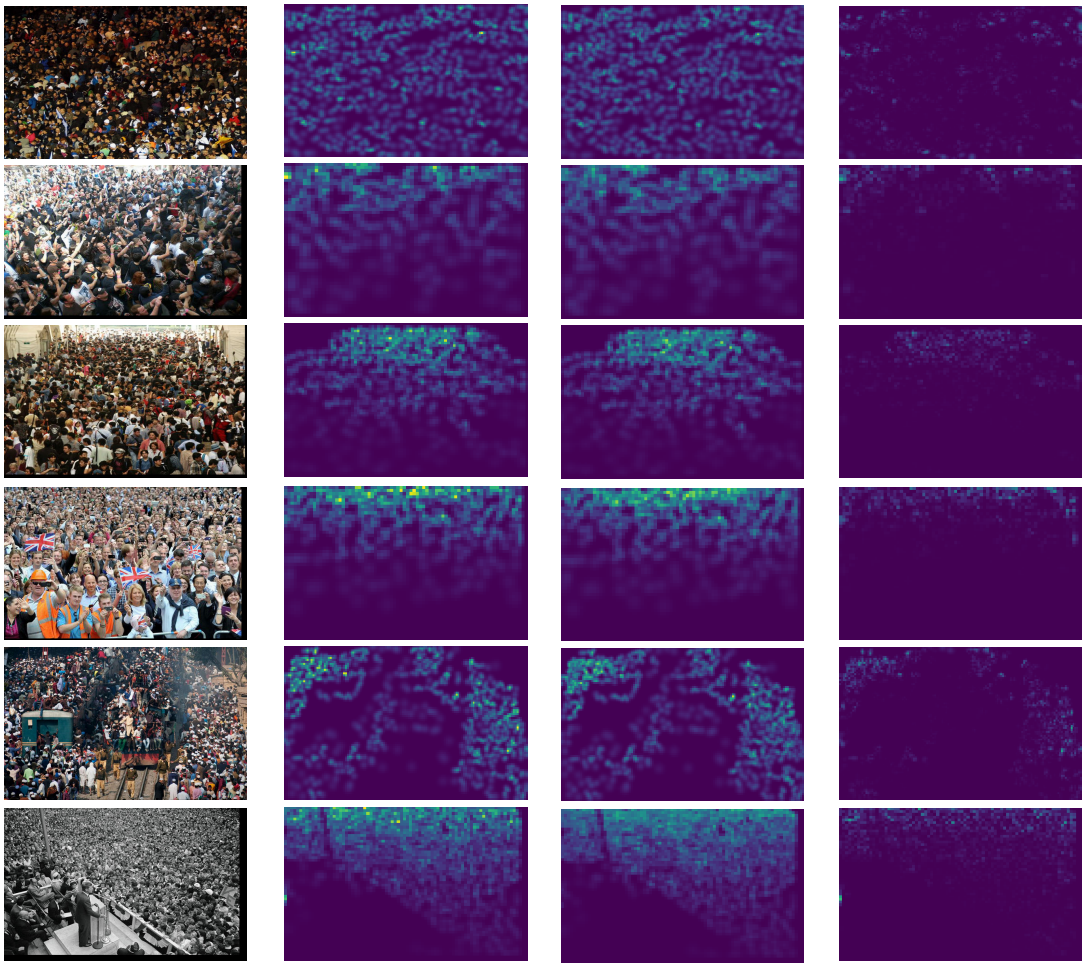
Gaussians are not ground-truth In Table 31, we show the effect of counting on distilled density maps over Gaussian density maps and multiscale Gaussian density maps [189]. This table is a supplement to Table 4 in the main paper. We use the same base network for auxiliary distillation and density prediction. Compared to using Gaussian densities and multiscale Gaussian densities, our distilled densities reduce the count error (MAE) considerably across networks and datasets. In Figure 50, we provide more examples of Gaussian and distilled density maps.

(a) **Input image** (b) **Cascaded image** (c) **Predicted density** (d) **Ground-truth**

Figure 48: **Learning to reduce the background.** Segmentation and density prediction results by our cascade model. While by no means perfect, the predicted segmentation masks already reduce the background pixels considerably. The density prediction network is then able to count correctly on the remaining amount of background pixels.

(a) **Original image** (b) **Our image** (c) **Original density** (d) **Our density**

Figure 49: **Occlusion creation**. Illustrative examples for creating new occlusions on images and corresponding density maps. Our occlusion-augmented image and density look quite natural.



(a) **Training image** (b) **Gaussian density** (c) **Distilled density** (d) **Density difference**

Figure 50: **Density generation with distillation**. Illustrative examples of Gaussian and distilled density maps. Compared to Gaussian densities, distilled densities reduce the local noisy maxima (yellow color region) and better match the objects in the image, especially the occluded objects. Also, the density difference shows that distilled densities changes Gaussian densities mostly from the occluded region.

6

SUMMARY AND CONCLUSIONS

6.1 SUMMARY

This thesis is dedicated to exploring inductive biases for pixel representation learning. Specifically, this thesis focuses on the research question: *How to uncover and exploit inductive biases for pixel representation learning?* We first uncovered three inductive biases and revealed their importance for different pixel-level tasks, including spectral bias for the deep image prior, salience bias for guided filtering, and attentional bias for object counting. Beyond uncovering different inductive biases, we then seek to develop new inductive biases by exploiting prior knowledge. We developed three inductive biases for best-in-class object counting by discovering new knowledge. A brief summary of each chapter is provided as follows:

Chapter 2: Spectral Bias of the Deep Image Prior. The deep image prior showed that a randomly initialized network with a suitable architecture can be trained to solve inverse imaging problems by simply optimizing it's parameters to reconstruct a single degraded image. However, it suffers from two practical limitations. First, it remains unclear how to control the prior beyond the choice of the network architecture. Second, training requires an oracle stopping criterion as during the optimization the performance degrades after reaching an optimum value. To address these challenges we introduce a frequency-band correspondence measure to characterize the spectral bias of the deep image prior, where low-frequency image signals are learned faster and better than high-frequency counterparts. Based on our observations, we propose techniques to prevent the eventual performance degradation and accelerate convergence. We introduce a Lipschitz-controlled convolution layer and a Gaussian-controlled upsampling layer as plug-in replacements for layers used in the deep architectures. The experiments show that with these changes the performance does not degrade during optimization, relieving us from the need for an oracle stopping criterion. We further outline a stopping criterion to avoid superfluous computation. Finally, we show that our approach obtains favorable results compared to current approaches across various denoising, deblocking, inpainting, super-resolution and detail enhancement tasks.

Chapter 3: Unsharp Mask Guided Filtering. The goal of this chapter is guided image filtering, which emphasizes the importance of structure transfer during filtering by means of an additional guidance image. Where classical guided filters transfer structures using hand-designed functions, recent guided filters have been considerably advanced through parametric learning of deep networks. The state-of-the-art leverages deep networks to estimate the two core coefficients of the guided filter. In this work, we posit that simultaneously estimating both coefficients is suboptimal, resulting in halo

artifacts and structure inconsistencies. Inspired by unsharp masking, a classical technique for edge enhancement that requires only a single coefficient, we propose a new and simplified formulation of the guided filter. Our formulation enjoys a filtering prior from a low-pass filter and enables explicit structure transfer by estimating a single coefficient. Based on our proposed formulation, we introduce a successive guided filtering network, which provides multiple filtering results from a single network, allowing for a trade-off between accuracy and efficiency. Extensive ablations, comparisons and analysis show the effectiveness and efficiency of our formulation and network, resulting in state-of-the-art results across filtering tasks like upsampling, denoising, and cross-modality filtering.

Chapter 4: Counting with Focus for Free. This chapter aims to count arbitrary objects in images. The leading counting approaches start from point annotations per object from which they construct density maps. Then, their training objective transforms input images to density maps through deep convolutional networks. We posit that the point annotations serve more supervision purposes than just constructing density maps. We introduce ways to repurpose the points for free. First, we propose supervised focus from segmentation, where points are converted into binary maps. The binary maps are combined with a network branch and accompanying loss function to focus on areas of interest. Second, we propose supervised focus from global density, where the ratio of point annotations to image pixels is used in another branch to regularize the overall density estimation. To assist both the density estimation and the focus from segmentation, we also introduce an improved kernel size estimator for the point annotations. Experiments on six datasets show that all our contributions reduce the counting error, regardless of the base network, resulting in state-of-the-art accuracy using only a single network. Finally, we are the first to count on WIDER FACE, allowing us to show the benefits of our approach in handling varying object scales and crowding levels.

Chapter 5: Three Things for Improving Density-Based Counting. This paper considers the problem of counting arbitrary entities in images by learning from density maps. Leading approaches start from point annotations per object from which they construct the density maps. Their training objective then transforms images to density maps through neural networks. In this work, we take a closer look at density-based counting and identify three things that limit every counting approach, and for each we propose a simple network plug-in module as a mitigation. Specifically, (i) we find that predicting densities on the background induces over half the error rate in counting and we outline a cascade to limit the effect of counting on background pixels; (ii) occlusions are persistent in counting, yet do not occur often enough in training images to learn to handle them properly. We propose an augmentation on both input and density images to learn to be more robust to occlusions; (iii) constructing density maps from point annotations with Gaussian convolutions is suboptimal for counting. We propose an alternative that learns to distill density maps from an auxiliary density prediction network. Such distilled maps are smoother and more robust to noise than their Gaussian counterparts. All three proposals are simple, can be plugged into any density-based counting network and when combined achieves state-of-the-art results on ShanghaiTech Part_A and Part_B, JHU-Crowd++, and TRANCOS, with the first mean average error below 50 on ShanghaiTech Part A.

6.2 CONCLUSIONS

This thesis has shown the importance of uncovering and exploiting inductive biases for pixel representation learning. The focus of this thesis has been on three inductive biases for three specific pixel-level tasks. As such, a logical next step involves to uncover and exploit more inductive biases for different tasks. For example, the frequency bias for generative models needs to be explored to better learn the high-frequency distribution. Another interesting direction is about exploring how the inductive biases human's exploit can be used as priors in representation learning. For example, we humans pay attention to what moves while current video representation learning models rely on still image domain to recognize actions in videos. Exploiting the motion bias of the human mind would be promising for deep learning to better understand the activities in videos.

BIBLIOGRAPHY

[1] B. AlBahar and J.-B. Huang. Guided image-to-image translation with bi-directional feature transformation. In *ICCV*, 2019.

[2] R. Arandjelović and A. Zisserman. Three things everyone should know to improve object retrieval. In *CVPR*, 2012.

[3] P. Arias, G. Facciolo, V. Caselles, and G. Sapiro. A variational framework for exemplar-based image inpainting. *International Journal of Computer Vision*, 93(3):319–347, 2011.

[4] D. Arpit, S. Jastrzebski, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. Courville, Y. Bengio, et al. A closer look at memorization in deep networks. In *ICML*, 2017.

[5] S. Arridge, P. Maass, O. Öktem, and C.-B. Schönlieb. Solving inverse problems using data-driven models. *Acta Numerica*, 28:1–174, 2019.

[6] M. Asim, F. Shamshad, and A. Ahmed. Patchdip exploiting patch redundancy in deep image prior for denoising. In *NeurIPS Workshop on Solving Inverse Problems with Deep Networks*, 2019.

[7] S. P. Awate and R. T. Whitaker. Unsupervised, information-theoretic, adaptive image filtering for image restoration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(3):364–376, 2006.

[8] T. C. Aysal and K. E. Barner. Quadratic weighted median filters for edge enhancement of noisy images. *IEEE Transactions on Image Processing*, 15(11):3294–3310, 2006.

[9] D. Babu Sam, N. N. Sajjan, R. Venkatesh Babu, and M. Srinivasan. Divide and grow: Capturing huge diversity in crowd images with incrementally growing cnn. In *CVPR*, 2018.

[10] D. Babu Sam, S. Surya, and R. Venkatesh Babu. Switching convolutional neural network for crowd counting. In *CVPR*, 2017.

[11] K. Bahrami and A. C. Kot. A fast approach for no-reference image sharpness assessment based on maximum local variation. *IEEE Signal Processing Letters*, 21(6):751–755, 2014.

[12] S. Bai, Z. He, Y. Qiao, H. Hu, W. Wu, and J. Yan. Adaptive dilated network with self-correction supervision for counting. In *CVPR*, 2020.

[13] M. R. Banham and A. K. Katsaggelos. Spatially adaptive wavelet-based multiscale image restoration. *IEEE Transactions on Image Processing*, 5(4):619–634, 1996.

[14] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[15] V. Belagiannis, C. Rupprecht, G. Carneiro, and N. Navab. Robust optimization for deep regression. In *ICCV*, 2015.

[16] M. Bertero and P. Boccacci. *Introduction to inverse problems in imaging*. IOP Publishing, 1998.

[17] M. Bevilacqua, A. Roumy, C. Guillemot, and M. L. Alberi-Morel. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. In *BMVC*, 2012.

[18] M. Brown and S. Süsstrunk. Multispectral SIFT for scene category recognition. In *CVPR*, 2011.

[19] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *ECCV*, 2012.

[20] X. Cao, Z. Wang, Y. Zhao, and F. Su. Scale aggregation network for accurate and efficient crowd counting. In *ECCV*, 2018.

[21] P. Chakrabarty and S. Maji. The spectral bias of the deep image prior. In *NeurIPS Workshop on Bayesian Deep Learning*, 2019.

[22] B. Chen, Z. Yan, K. Li, P. Li, B. Wang, W. Zuo, and L. Zhang. Variational attention: Propagating domain-specific knowledge for multi-domain learning in crowd counting. In *ICCV*, 2021.

[23] D. Chen, Q. Fan, J. Liao, A. Aviles-Rivero, L. Yuan, N. Yu, and G. Hua. Controllable image processing via adaptive filterbank pyramid. *IEEE Transactions on Image Processing*, 29:8043–8054, 2020.

[24] L. Chen, H. Zhang, J. Xiao, L. Nie, J. Shao, W. Liu, and T.-S. Chua. Sca-cnn: Spatial and channel-wise attention in convolutional networks for image captioning. In *CVPR*, 2017.

[25] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):834–848, 2017.

[26] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018.

[27] Y. Chen and T. Pock. Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1256–1272, 2016.

[28] Y.-C. Chen, C. Gao, E. Robb, and J.-B. Huang. Nas-dip: Learning deep image prior with neural architecture search. In *ECCV*, 2020.

[29] J. Cheng, H. Xiong, Z. Cao, and H. Lu. Decoupled two-stage crowd counting and beyond. *IEEE Transactions on Image Processing*, 30:2862–2875, 2021.

[30] Z. Cheng, M. Gadelha, S. Maji, and D. Sheldon. A bayesian perspective on the deep image prior. In *CVPR*, 2019.

[31] F. Crete, T. Dolmiere, P. Ladret, and M. Nicolas. The blur effect: Perception and estimation with a new no-reference perceptual blur metric. In *SPIE HVEI*, 2007.

[32] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Transactions on Image Processing*, 16(8):2080–2095, 2007.

[33] T. Dai, Y. Feng, D. Wu, B. Chen, J. Lu, Y. Jiang, and S.-T. Xia. DIPDefend: Deep image prior driven defense against adversarial examples. In *ACM MM*, 2020.

[34] I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics*, 57(11):1413–1457, 2004.

[35] G. Deng. A generalized unsharp masking algorithm. *IEEE Transactions on Image Processing*, 20(5):1249–1261, 2010.

[36] X. Deng and P. L. Dragotti. Deep coupled ista network for multi-modal image super-resolution. *IEEE Transactions on Image Processing*, 29:1683–1698, 2019.

[37] X. Deng and P. L. Dragotti. Deep convolutional neural network for multi-modal image restoration and fusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[38] T. DeVries and G. W. Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

[39] C. Dong, Y. Deng, C. C. Loy, and X. Tang. Compression artifacts reduction by a deep convolutional network. In *ICCV*, pages 576–584, 2015.

[40] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(2):295–307, 2015.

[41] W. Dong, G. Shi, Y. Ma, and X. Li. Image restoration via simultaneous sparse coding: Where structured sparsity meets gaussian scale mixture. *International Journal of Computer Vision*, 114(2):217–232, 2015.

[42] D. Dunn and W. E. Higgins. Optimal gabor filters for texture segmentation. *IEEE Transactions on Image Processing*, 4(7):947–964, 1995.

[43] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *ICCV*, 1999.

[44] M. Elad and A. Feuer. Superresolution restoration of an image sequence: adaptive filtering approach. *IEEE Transactions on Image Processing*, 8(3):387–395, 1999.

[45] M. Elad, M. A. Figueiredo, and Y. Ma. On the role of sparse and redundant representations in image processing. *Proceedings of the IEEE*, 98(6):972–982, 2010.

[46] H. W. Engl, M. Hanke, and A. Neubauer. *Regularization of inverse problems*, volume 375. Springer Science & Business Media, 1996.

[47] Z. Farbman, R. Fattal, D. Lischinski, and R. Szeliski. Edge-preserving decompositions for multi-scale tone and detail manipulation. *ACM Transactions on Graphics*, 27(3):1–10, 2008.

[48] A. Foi, V. Katkovnik, and K. Egiazarian. Pointwise shape-adaptive dct for high-quality deblocking of compressed color images. In *ESPC*, pages 1–5, 2006.

[49] J. Fu, H. Zheng, and T. Mei. Look closer to see better: recurrent attention convolutional neural network for fine-grained image recognition. In *CVPR*, 2017.

[50] Y. Gandelsman, A. Shocher, and M. Irani. Double-dip: Unsupervised image decomposition via coupled deep-image-priors. In *CVPR*, 2019.

[51] Y. Gao, O. Beijbom, N. Zhang, and T. Darrell. Compact bilinear pooling. In *CVPR*, 2016.

[52] M. Gharbi, J. Chen, J. T. Barron, S. W. Hasinoff, and F. Durand. Deep bilateral learning for real-time image enhancement. *ACM Transactions on Graphics*, 36(4):1–12, 2017.

[53] R. Girdhar and D. Ramanan. Attentional pooling for action recognition. In *NeurIPS*, 2017.

[54] R. Guerrero-Gómez-Olmedo, B. Torre-Jiménez, R. López-Sastre, S. Maldonado-Bascón, and D. Onoro-Rubio. Extremely overlapping vehicle counting. In *IbPRIA*, 2015.

[55] S. Gunasekar, J. Lee, D. Soudry, and N. Srebro. Characterizing implicit bias in terms of optimization geometry. In *ICML*, pages 1832–1841. PMLR, 2018.

[56] X. Guo, Y. Li, and H. Ling. Lime: Low-light image enhancement via illumination map estimation. *IEEE Transactions on Image Processing*, 26(2):982–993, 2016.

[57] X. Guo, Y. Li, J. Ma, and H. Ling. Mutually guided image filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(3):694–707, 2020.

[58] J. Hahn, X.-C. Tai, S. Borok, and A. M. Bruckstein. Orientation-matching minimization for image denoising and inpainting. *International Journal of Computer Vision*, 92(3):308–324, 2011.

[59] B. Ham, M. Cho, and J. Ponce. Robust guided image filtering using nonconvex potentials. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(1):192–207, 2018.

[60] K. He, J. Sun, and X. Tang. Guided image filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(6):1397–1409, 2012.

[61] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015.

[62] S. He and R. W. Lau. Saliency detection with flash and no-flash image pairs. In *ECCV*, 2014.

[63] R. Heckel and P. Hand. Deep decoder: Concise image representations from untrained non-convolutional networks. In *ICLR*, 2019.

[64] R. Heckel and M. Soltanolkotabi. Denoising and regularization via exploiting the structural bias of convolutional generators. In *ICLR*, 2020.

[65] F. Heide, W. Heidrich, and G. Wetzstein. Fast and flexible convolutional sparse coding. In *CVPR*, 2015.

[66] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[67] K. Ho, A. Gilbert, H. Jin, and J. Collomosse. Neural architecture search for deep image prior. *ArXiv:2001.04776*, 2020.

[68] M. Hossain, M. Hosseinzadeh, O. Chanda, and Y. Wang. Crowd counting using scale-aware attention networks. In *WACV*, 2019.

[69] Q. Hou, M.-M. Cheng, X. Hu, A. Borji, Z. Tu, and P. H. Torr. Deeply supervised salient object detection with short connections. In *CVPR*, 2017.

[70] Y. Hu, X. Jiang, X. Liu, B. Zhang, J. Han, X. Cao, and D. Doermann. Nas-count: Counting-by-density with neural architecture search. In *ECCV*, 2020.

[71] T.-W. Hui, C. C. Loy, and X. Tang. Depth map super-resolution by deep multi-scale guidance. In *ECCV*, 2016.

[72] H. Idrees, M. Tayyab, K. Athrey, D. Zhang, S. Al-Maadeed, N. Rajpoot, and M. Shah. Composition loss for counting, density map estimation and localization in dense crowds. In *ECCV*, 2018.

[73] S. Iizuka, E. Simo-Serra, and H. Ishikawa. Let there be color! joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification. *ACM Transactions on Graphics*, 35(4):1–11, 2016.

[74] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

[75] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017.

[76] H. L. Issam, R. Negar, O. P. Pedro, V. David, and S. Mark. Where are the blobs: Counting by localization with point supervision. In *ECCV*, 2018.

[77] M. Jacob and M. Unser. Design of steerable filters for feature detection using canny-like criteria. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(8):1007–1019, 2004.

[78] V. Jain and S. Seung. Natural image denoising with convolutional networks. In *NeuIPS*, 2008.

[79] R. J. Jevnisek and S. Avidan. Co-occurrence filter. In *CVPR*, 2017.

[80] X. Jiang, Z. Xiao, B. Zhang, X. Zhen, X. Cao, D. Doermann, and L. Shao. Crowd counting and density estimation by trellis encoder-decoder networks. In *CVPR*, 2019.

[81] X. Jiang, L. Zhang, M. Xu, T. Zhang, P. Lv, B. Zhou, X. Yang, and Y. Pang. Attention scaling for crowd counting. In *CVPR*, 2020.

[82] K. H. Jin, M. T. McCann, E. Froustey, and M. Unser. Deep convolutional neural network for inverse problems in imaging. *IEEE Transactions on Image Processing*, 26(9):4509–4522, 2017.

[83] D. Kang and A. B. Chan. Crowd counting by adaptively fusing predictions from an image pyramid. In *BMVC*, 2018.

[84] Y. Kang, C. Roh, S.-B. Suh, and B. Song. A lidar-based decision-making method for road boundary detection using multiple kalman filters. *IEEE Transactions on Industrial Electronics*, 59(11):4360–4368, 2012.

[85] A. K. Katsaggelos. Iterative image restoration algorithms. *Optical Engineering*, 28(7):287735, 1989.

[86] A. Kattamis, T. Adel, and A. Weller. Exploring properties of the deep image prior. In *NeurIPS Workshop on Solving Inverse Problems with Deep Networks*, 2019.

[87] Y. Katznelson. *An introduction to harmonic analysis*. Cambridge University Press, 2004.

[88] S. Kindermann, S. Osher, and P. W. Jones. Deblurring and denoising of images by nonlocal functionals. *Multiscale Modeling & Simulation*, 4(4):1091–1115, 2005.

[89] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[90] J. Kopf, M. F. Cohen, D. Lischinski, and M. Uyttendaele. Joint bilateral upsampling. *ACM Transactions on Graphics*, 26(3):96, 2007.

[91] F. Kou, W. Chen, C. Wen, and Z. Li. Gradient domain guided image filtering. *IEEE Transactions on Image Processing*, 24(11):4528–4539, 2015.

[92] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, 2017.

[93] S. Lefkimmiatis. Universal denoising networks: a novel cnn architecture for image denoising. In *CVPR*, 2018.

[94] V. Lempitsky and A. Zisserman. Learning to count objects in images. In *NeurIPS*, 2010.

[95] J. Li, S. You, and A. Robles-Kelly. A frequency domain neural network for fast image super-resolution. In *IJCNN*, 2018.

[96] Y. Li, J.-B. Huang, N. Ahuja, and M.-H. Yang. Deep joint image filtering. In *ECCV*, 2016.

[97] Y. Li, J.-B. Huang, N. Ahuja, and M.-H. Yang. Joint image filtering with deep convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(8):1909–1923, 2019.

[98] Y. Li, X. Zhang, and D. Chen. Csrnet: dilated convolutional neural networks for understanding the highly congested scenes. In *CVPR*, 2018.

[99] Z. Li, K. Gavrilyuk, E. Gavves, M. Jain, and C. G. M. Snoek. VideoLSTM convolves, attends and flows for action recognition. *Computer Vision and Image Understanding*, 166:41–50, 2018.

[100] Z. Li, J. Zheng, Z. Zhu, W. Yao, and S. Wu. Weighted guided image filtering. *IEEE Transactions on Image Processing*, 24(1):120–129, 2014.

[101] T.-Y. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017.

[102] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *ICCV*, 2017.

[103] T.-Y. Lin, A. RoyChowdhury, and S. Maji. Bilinear cnn models for fine-grained visual recognition. In *ICCV*, 2015.

[104] Z. Lin, J. He, X. Tang, and C.-K. Tang. Limits of learning-based superresolution algorithms. *International Journal of Computer Vision*, 80(3):406–420, 2008.

[105] J. Liu, C. Gao, D. Meng, and A. G. Hauptmann. Decidenet: Counting varying density crowds through attention guided detection and density estimation. In *CVPR*, 2018.

[106] J. Liu, Y. Sun, X. Xu, and U. S. Kamilov. Image restoration using total variation regularized deep image prior. In *ICASSP*, 2019.

[107] L. Liu, J. Chen, H. Wu, T. Chen, G. Li, and L. Lin. Efficient crowd counting via structured knowledge transfer. In *ACM MM*, 2020.

[108] L. Liu, H. Lu, H. Xiong, K. Xian, Z. Cao, and C. Shen. Counting objects by blockwise classification. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(10):3513–3527, 2019.

[109] L. Liu, H. Lu, H. Zou, H. Xiong, Z. Cao, and C. Shen. Weighing counts: Sequential crowd counting by reinforcement learning. In *ECCV*, 2020.

[110] L. Liu, Z. Qiu, G. Li, S. Liu, W. Ouyang, and L. Lin. Crowd counting with deep structured scale integration network. In *ICCV*, 2019.

[111] L. Liu, H. Wang, G. Li, W. Ouyang, and L. Lin. Crowd counting using deep recurrent spatial-aware network. In *IJCAI*, 2018.

[112] M.-Y. Liu, O. Tuzel, and Y. Taguchi. Joint geodesic upsampling of depth images. In *CVPR*, 2013.

[113] N. Liu, Y. Long, C. Zou, Q. Niu, L. Pan, and H. Wu. Adcrowdnet: An attention-injective deformable convolutional network for crowd understanding. In *CVPR*, 2019.

[114] W. Liu, M. Salzmann, and P. Fua. Context-aware crowd counting. In *CVPR*, 2019.

[115] W. Liu, M. Salzmann, and P. Fua. Estimating people flows to better count them in crowded scenes. In *ECCV*, 2020.

[116] X. Liu, G. Li, Z. Han, W. Zhang, Y. Yang, Q. Huang, and N. Sebe. Exploiting sample correlation for crowd counting with multi-expert network. In *ICCV*, 2021.

[117] X. Liu, J. van de Weijer, and A. D. Bagdanov. Leveraging unlabeled data for crowd counting by learning to rank. In *CVPR*, 2018.

[118] X. Liu, J. van de Weijer, and A. D. Bagdanov. Leveraging unlabeled data for crowd counting by learning to rank. In *CVPR*, 2018.

[119] A. Lucas, M. Iliadis, R. Molina, and A. K. Katsaggelos. Using deep neural networks for inverse problems in imaging: beyond analytical methods. *IEEE Signal Processing Magazine*, 35(1):20–36, 2018.

[120] W.-Y. Ma and B. S. Manjunath. Edgeflow: a technique for boundary detection and image segmentation. *IEEE Transactions on Image Processing*, 9(8):1375–1388, 2000.

[121] Z. Ma, X. Hong, X. Wei, Y. Qiu, and Y. Gong. Towards a universal model for cross-dataset crowd counting. In *ICCV*, 2021.

[122] Z. Ma, X. Wei, X. Hong, and Y. Gong. Bayesian loss for crowd count estimation with point supervision. In *ICCV*, 2019.

[123] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Non-local sparse models for image restoration. In *ICCV*, 2009.

[124] X. Mao, C. Shen, and Y.-B. Yang. Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. In *NeurIPS*, 2016.

[125] A. Marana, L. da Costa, R. Lotufo, and S. Velastin. On the efficacy of texture analysis for crowd monitoring. In *SIBGRAPI*, 1998.

[126] I. Marivani, E. Tsiligianni, B. Cornelis, and N. Deligiannis. Learned multimodal convolutional sparse coding for guided image super-resolution. In *ICIP*, 2019.

[127] I. Marivani, E. Tsiligianni, B. Cornelis, and N. Deligiannis. Multimodal image super-resolution via deep unfolding with side information. In *EUSIPCO*, 2019.

[128] I. Marivani, E. Tsiligianni, B. Cornelis, and N. Deligiannis. Joint image super-resolution via recurrent convolutional neural networks with coupled sparse priors. In *ICIP*, 2020.

[129] I. Marivani, E. Tsiligianni, B. Cornelis, and N. Deligiannis. Multimodal deep unfolding for guided image super-resolution. *IEEE Transactions on Image Processing*, 29:8443–8456, 2020.

[130] I. Marivani, E. Tsiligianni, B. Cornelis, and N. Deligiannis. Designing cnns for multimodal image super-resolution via the method of multipliers. In *EUSIPCO*, 2021.

[131] M. Marsden, K. McGuinness, S. Little, C. E. Keogh, and N. E. O'Connor. People, penguins and petri dishes: adapting object counting models to new visual domains and object types without forgetting. In *CVPR*, 2018.

[132] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *ICCV*, 2001.

[133] G. Mataev, P. Milanfar, and M. Elad. Deepred: Deep image prior powered by red. In *ICCV Workshop on Learning for Computational Imaging*, 2019.

[134] M. T. McCann, K. H. Jin, and M. Unser. Convolutional neural networks for inverse problems in imaging: A review. *IEEE Signal Processing Magazine*, 34(6):85–95, 2017.

[135] P. Mianjy, R. Arora, and R. Vidal. On the implicit bias of dropout. In *ICML*, pages 3540–3548. PMLR, 2018.

[136] D. Min, J. Lu, and M. N. Do. Depth video enhancement based on weighted mode filtering. *IEEE Transactions on Image Processing*, 21(3):1176–1190, 2011.

[137] T. M. Mitchell. *The need for biases in learning generalizations*. Department of Computer Science, Laboratory for Computer Science Research . . . , 1980.

[138] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. In *ICLR*, 2018.

[139] D. Modolo, B. Shuai, R. R. Varior, and J. Tighe. Understanding the impact of mistakes on background regions in crowd counting. In *WACV*, 2021.

[140] K. Morishita, S. Yamagata, T. Okabe, T. Yokoyama, and K. Hamatani. Unsharp masking for image enhancement, Dec. 27 1988. US Patent 4,794,531.

[141] A. Odena, V. Dumoulin, and C. Olah. Deconvolution and checkerboard artifacts. *Distill*, 1(10), 2016.

[142] B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996.

[143] D. Onoro-Rubio and R. J. López-Sastre. Towards perspective-free object counting with deep learning. In *ECCV*, 2016.

[144] J. Pan, J. Dong, J. S. Ren, L. Lin, J. Tang, and M.-H. Yang. Spatially variant linear representation models for joint filtering. In *CVPR*, 2019.

[145] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: feature learning by inpainting. In *CVPR*, 2016.

[146] G. Petschnigg, R. Szeliski, M. Agrawala, M. Cohen, H. Hoppe, and K. Toyama. Digital photography with flash and no-flash image pairs. *ACM Transactions on Graphics*, 23(3):664–672, 2004.

[147] C. C. Pham, S. V. U. Ha, and J. W. Jeon. Adaptive guided image filtering for sharpness enhancement and noise reduction. In *PSIVT*, 2011.

[148] A. Polesel, G. Ramponi, and V. J. Mathews. Image enhancement via adaptive unsharp masking. *IEEE Transactions on Image Processing*, 9(3):505–510, 2000.

[149] J. Portilla. Image restoration through l0 analysis-based sparse optimization in tight frames. In *ICIP*, 2009.

[150] M. Protter, M. Elad, H. Takeda, and P. Milanfar. Generalizing the nonlocal-means to super-resolution reconstruction. *IEEE Transactions on image processing*, 18(1):36–51, 2008.

[151] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. Hamprecht, Y. Bengio, and A. Courville. On the spectral bias of neural networks. In *ICML*, 2019.

[152] T. Randen and J. H. Husoy. Texture segmentation using filters with optimized energy separation. *IEEE Transactions on Image Processing*, 8(4):571–582, 1999.

[153] V. Ranjan, H. Le, and M. Hoai. Iterative crowd counting. In *ECCV*, 2018.

[154] B. Rasti, B. Koirala, P. Scheunders, and P. Ghamisi. Undip: Hyperspectral unmixing using deep image prior. *IEEE Transactions on Geoscience and Remote Sensing*, 2021.

[155] A. Ribes and F. Schmitt. Linear inverse problems in imaging. *IEEE Signal Processing Magazine*, 25(4):84–99, 2008.

[156] S. Roth and M. J. Black. Fields of experts. *International Journal of Computer Vision*, 82(2):205, 2009.

[157] D. L. Ruderman. The statistics of natural images. *Network: Computation in Neural Systems*, 5(4):517–548, 1994.

[158] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena*, 60(1-4):259–268, 1992.

[159] S. R. Safavian and D. Landgrebe. A survey of decision tree classifier methodology. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(3):660–674, 1991.

[160] D. B. Sam, S. Surya, and R. V. Babu. Switching convolutional neural network for crowd counting. In *CVPR*, 2017.

[161] D. H. Schenk. Exploiting the salience bias in designing taxes. *Yale J. on Reg.*, 28:253, 2011.

[162] S. C. Segerstrom. Optimism and attentional bias for negative and positive stimuli. *Personality and social psychology bulletin*, 27(10):1334–1343, 2001.

[163] H. R. Sheikh, M. F. Sabir, and A. C. Bovik. A statistical evaluation of recent full reference image quality assessment algorithms. *IEEE Transactions on image processing*, 15(11):3440–3451, 2006.

[164] X. Shen, C. Zhou, L. Xu, and J. Jia. Mutual-structure for joint filtering. In *ICCV*, 2015.

[165] Z. Shen, Y. Xu, B. Ni, M. Wang, J. Hu, and X. Yang. Crowd counting via adversarial cross-scale consistency pursuit. In *CVPR*, 2018.

[166] J. Shi, Q. Yan, L. Xu, and J. Jia. Hierarchical image saliency detection on extended cssd. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(4):717–729, 2015.

[167] M. Shi, Z. Yang, C. Xu, and Q. Chen. Revisiting perspective information for efficient crowd counting. In *CVPR*, 2019.

[168] Z. Shi, Y. Chen, E. Gavves, P. Mettes, and C. G. M. Snoek. Unsharp mask guided filtering. *IEEE Transactions on Image Processing*, 30:7472–7485, 2021.

[169] Z. Shi, P. Mettes, S. Maji, and C. G. M. Snoek. On measuring and controlling the spectral bias of the deep image prior. *International Journal of Computer Vision*, 2021.

[170] Z. Shi, P. Mettes, and C. G. M. Snoek. Counting with focus for free. In *ICCV*, 2019.

[171] Z. Shi, L. Zhang, Y. Liu, X. Cao, Y. Ye, M.-M. Cheng, and G. Zheng. Crowd counting with deep negative correlation learning. In *CVPR*, 2018.

[172] Z. Shi, L. Zhang, Y. Sun, and Y. Ye. Multiscale multitask deep netvlad for crowd counting. *IEEE Transactions on Industrial Informatics*, 14(11):4953–4962, 2018.

[173] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[174] E. P. Simoncelli and B. A. Olshausen. Natural image statistics and neural representation. *Annual Review of Neuroscience*, 24(1):1193–1216, 2001.

[175] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[176] V. A. Sindagi and V. M. Patel. Generating high-quality crowd density maps using contextual pyramid cnns. In *ICCV*, 2017.

[177] V. A. Sindagi and V. M. Patel. Multi-level bottom-top and top-bottom feature fusion for crowd counting. In *ICCV*, 2019.

[178] V. A. Sindagi, R. Yasarla, and V. M. Patel. Jhu-crowd++: Large-scale crowd counting dataset and a benchmark method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[179] P. Song, X. Deng, J. F. Mota, N. Deligiannis, P. L. Dragotti, and M. R. Rodrigues. Multimodal image super-resolution via joint sparse representations induced by coupled dictionaries. *IEEE Transactions on Computational Imaging*, 6:57–72, 2019.

[180] Q. Song, C. Wang, Z. Jiang, Y. Wang, Y. Tai, C. Wang, J. Li, F. Huang, and Y. Wu. Rethinking counting and localization in crowds: A purely point-based framework. In *ICCV*, 2021.

[181] D. Soudry, E. Hoffer, M. S. Nacson, S. Gunasekar, and N. Srebro. The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research*, 19(1):2822–2878, 2018.

[182] H. Su, V. Jampani, D. Sun, O. Gallo, E. Learned-Miller, and J. Kautz. Pixel-adaptive convolutional neural networks. In *CVPR*, 2019.

[183] Z. Sun, B. Han, J. Li, J. Zhang, and X. Gao. Weighted guided image filtering with steering kernel. *IEEE Transactions on Image Processing*, 29:500–508, 2019.

[184] Y. Tai, J. Yang, and X. Liu. Image super-resolution via deep recursive residual network. In *CVPR*, pages 3147–3155, 2017.

[185] D. Titterington. General structure of regularization procedures in image reconstruction. *Astronomy and Astrophysics*, 144:381, 1985.

[186] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Deep image prior. In *CVPR*, 2018.

[187] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Deep image prior. *International Journal of Computer Vision*, 128(7), 2020.

[188] T. Vu, A. DiSpirito, D. Li, Z. Wang, X. Zhu, M. Chen, L. Jiang, D. Zhang, J. Luo, Y. S. Zhang, Q. Zhou, R. Horstmeyer, and J. Yao. Deep image prior for undersampling high-speed photoacoustic microscopy. *Photoacoustics*, 22:100266, 2021.

[189] J. Wan and A. Chan. Adaptive density map generation for crowd counting. In *ICCV*, pages 1130–1139, 2019.

[190] J. Wan and A. Chan. Modeling noisy annotations for crowd counting. In *NeurIPS*, 2020.

[191] J. Wan, Z. Liu, and A. B. Chan. A generalized loss function for crowd counting and localization. In *CVPR*, 2021.

[192] J. Wan, Q. Wang, and A. B. Chan. Kernel-based density map generation for dense object counting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[193] Z. Wan, B. Zhang, D. Chen, P. Zhang, D. Chen, J. Liao, and F. Wen. Bringing old photos back to life. In *CVPR*, 2020.

[194] B. Wang, H. Liu, D. Samaras, and M. Hoai. Distribution matching for crowd counting. In *NeurIPS*, 2020.

[195] C. Wang, Q. Song, B. Zhang, Y. Wang, Y. Tai, X. Hu, C. Wang, J. Li, J. Ma, and Y. Wu. Uniformity in heterogeneity: diving deep into count interval partition for crowd counting. In *ICCV*, 2021.

[196] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang, et al. Deep high-resolution representation learning for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[197] P. Wang, P. Chen, Y. Yuan, D. Liu, Z. Huang, X. Hou, and G. Cottrell. Understanding convolution for semantic segmentation. In *WACV*, 2018.

[198] Q. Wang, J. Gao, W. Lin, and Y. Yuan. Pixel-wise crowd understanding via synthetic data. *International Journal of Computer Vision*, 129(1):225–245, 2021.

[199] X. Wang, F. Dai, Y. Ma, J. Guo, Q. Zhao, and Y. Zhang. Near-infrared image guided neural networks for color image denoising. In *ICASSP*, 2019.

[200] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[201] S. Weisberg. *Applied linear regression*, volume 528. John Wiley & Sons, 2005.

[202] T. P. Weldon, W. E. Higgins, and D. F. Dunn. Efficient gabor filter design for texture segmentation. *Pattern Recognition*, 29(12):2005–2016, 1996.

[203] H. Wu, S. Zheng, J. Zhang, and K. Huang. Fast end-to-end trainable guided filter. In *CVPR*, 2018.

[204] J. Xie, R. S. Feris, and M.-T. Sun. Edge-guided single depth image super resolution. *IEEE Transactions on Image Processing*, 25(1):428–438, 2015.

[205] C. Xu, K. Qiu, J. Fu, S. Bai, Y. Xu, and X. Bai. Learn to scale: Generating multipolar normalized density maps for crowd counting. In *ICCV*, 2019.

[206] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2015.

[207] X. Xu, Y. Ma, and W. Sun. Learning factorized weight matrix for joint filtering. In *ICML*, 2020.

[208] Z.-Q. J. Xu, Y. Zhang, T. Luo, Y. Xiao, and Z. Ma. Frequency principle: Fourier analysis sheds light on deep neural networks. *Communications in Computational Physics*, 2020.

[209] Q. Yan, X. Shen, L. Xu, S. Zhuo, X. Zhang, L. Shen, and J. Jia. Cross-field joint image restoration via scale map. In *ICCV*, 2013.

[210] Z. Yan, Y. Yuan, W. Zuo, X. Tan, Y. Wang, S. Wen, and E. Ding. Perspective-guided convolution networks for crowd counting. In *ICCV*, 2019.

[211] S. Yang, P. Luo, C. C. Loy, and X. Tang. Wider face: A face detection benchmark. In *CVPR*, 2016.

[212] W. Ye and K.-K. Ma. Blurriness-guided unsharp masking. *IEEE Transactions on Image Processing*, 27(9):4465–4477, 2018.

[213] H. Yin, Y. Gong, and G. Qiu. Side window filtering. In *CVPR*, 2019.

[214] D. Yu, J. Fu, T. Mei, and Y. Rui. Multi-level attention networks for visual question answering. In *CVPR*, 2017.

[215] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. *ICLR*, 2015.

[216] F. Yu, V. Koltun, and T. A. Funkhouser. Dilated residual networks. In *CVPR*, 2017.

[217] T. Yu, S. Simoff, and T. Jan. Vqsvm: a case study for incorporating prior domain knowledge into inductive machine learning. *Neurocomputing*, 73(13-15):2614–2623, 2010.

[218] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *ICCV*, 2019.

[219] R. Zeyde, M. Elad, and M. Protter. On single image scale-up using sparse-representations. In *ICCS*, 2010.

[220] A. Zhang, J. Shen, Z. Xiao, F. Zhu, X. Zhen, X. Cao, and L. Shao. Relational attention network for crowd counting. In *ICCV*, 2019.

[221] A. Zhang, L. Yue, J. Shen, F. Zhu, X. Zhen, X. Cao, and L. Shao. Attentional neural fields for crowd counting. In *ICCV*, 2019.

[222] C. Zhang, H. Li, X. Wang, and X. Yang. Cross-scene crowd counting via deep convolutional neural networks. In *CVPR*, 2015.

[223] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *Transactions on Image Processing*, 26(7):3142–3155, 2017.

[224] K. Zhang, W. Zuo, and L. Zhang. Ffdnet: Toward a fast and flexible solution for cnn-based image denoising. *IEEE Transactions on Image Processing*, 27(9):4608–4622, 2018.

[225] L. Zhang, Z. Shi, M.-M. Cheng, Y. Liu, J.-W. Bian, J. T. Zhou, G. Zheng, and Z. Zeng. Nonlinear regression via deep negative correlation learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[226] S. Zhang, G. Wu, J. P. Costeira, and J. M. F. Moura. Understanding traffic density from large-scale web camera data. In *CVPR*, 2017.

[227] S. Zhang, J. Yang, and B. Schiele. Occluded pedestrian detection through guided attention in cnns. In *CVPR*, 2018.

[228] Y. Zhang, D. Zhou, S. Chen, S. Gao, and Y. Ma. Single-image crowd counting via multi-column convolutional neural network. In *CVPR*, 2016.

[229] M. Zhao, J. Zhang, C. Zhang, and W. Zhang. Leveraging heterogeneous auxiliary tasks to assist crowd counting. In *CVPR*, 2019.

[230] Z. Zhu, W. Wu, W. Zou, and J. Yan. End-to-end flow correlation tracking with spatial-temporal attention. In *CVPR*, 2018.

[231] J. Zukerman, T. Tirer, and R. Giryes. Bp-dip: A backprojection based deep image prior. In *EUSIPCO*, 2020.

COMPLETE LIST OF PUBLICATIONS

- **Zenglin Shi**, Pascal Mettes, Cees G. M. Snoek. "Three Things Everyone Should Know to Improve Density-Based Counting", in submission to **European Conference on Computer Vision**, 2022.
- **Zenglin Shi**, Pascal Mettes, Subhransu Maji, Cees G. M. Snoek. "On Measuring and Controlling the Spectral Bias of the Deep Image Prior", **International Journal of Computer Vision**, https://doi.org/10.1007/s11263-021-01572-7, 2022.
- **Zenglin Shi**, Yunlu Chen, Efstratios Gavves, Pascal Mettes, Cees G. M. Snoek. "Unsharp Mask Guided Filtering", **IEEE Transactions on Image processing**, vol. 30, pp. 7472-7485, 2021.
- **Zenglin Shi**, Pascal Mettes, Guoyan Zheng, Cees G. M. Snoek. "Frequency-Supervised MR-to-CT Image Synthesis", **MICCAI workshop on Deep Generative Models**, 2021.
- Le Zhang*, **Zenglin Shi***, Joey Tianyi Zhou, Ming-Ming Cheng, Yun Liu, Jia-Wang Bian, Zeng Zeng, Chunhua Shen. "Ordered or Orderless: A Revisit for Video based Person Re-ID", **IEEE Transactions on Pattern Analysis and Machine Intelligence**, vol. 43, pp. 1460-1466, 2021. (* equal contributions)
- Le Zhang*, **Zenglin Shi***, Ming-Ming Cheng, Yun Liu, Jia-Wang Bian, Joey Tianyi Zhou, Guoyan Zheng, Zeng Zeng. "Nonlinear Regression via Deep Negative Correlation Learning", **IEEE Transactions on Pattern Analysis and Machine Intelligence**, vol. 43, pp. 982-998, 2021. (* equal contributions)
- Shuo Chen, **Zenglin Shi**, Pascal Mettes, Cees G. M. Snoek. "Social Fabric: Tubelet Compositions for Video Relation Detection", **IEEE/CVF International Conference on Computer Vision**, 2021.
- Shizhe Hu, **Zenglin Shi**, Yangdong Ye. "DMIB: Dual-correlated Multivariate Information Bottleneck for Multi-view Clustering", **IEEE Transactions on Cybernetics**, vol. 41, pp. 1261-1273, 2020.
- **Zenglin Shi**, Pascal Mettes, Cees G. M. Snoek. "Counting with Focus for Free", **IEEE/CVF International Conference on Computer Vision**, 2019.
- Xiaofeng Cao, Baozhi Qiu, Xiangli Li, **Zenglin Shi**, Guandong Xu, Jianliang Xu. "Multidimensional Balance-Based Cluster Boundary Detection for High-Dimensional Data", **IEEE Transactions on Neural Networks and Learning Systems**, vol. 30, pp. 1867-1880, 2019.
- **Zenglin Shi**, Le Zhang, Yun Liu, Xiaofeng Cao, Yangdong Ye, Ming-Ming Cheng, Guoyan Zheng. "Crowd Counting with Deep Negative Correlation Learning", **IEEE/CVF Conference on Computer Vision and Pattern Recognition**, 2018.
- **Zenglin Shi**, Guodong Zeng, Le Zhang, Xiahai Zhuang, Lei Li, Guang Yang, Guoyan Zheng. "Bayesian VoxDRN: A Probabilistic Deep Voxelwise Dilated Residual Network for WholeHeart Segmentation from 3D MR Images", **MICCAI**, 2018. (spotlight oral)

- **Zenglin Shi**, Le Zhang, Yibo Sun and Yandong Ye. "Multiscale Multitask Deep NetVLAD for Crowd Counting", **IEEE Transactions on Industrial Informatics**, vol. 14, pp. 4953-4962, 2018.
- Yunpeng Wu, Yangdong Ye, Chenyang Zhao, **Zenglin Shi**. "Collective Density Clustering for Coherent Motion Detection", **IEEE Transactions on MultiMedia**, vol. 20, pp. 1418-1431, 2018.
- **Zenglin Shi**, Yangdong Ye, Yunpeng Wu. "Rank-based Pooling for Deep Convolutional Neural Networks", **Neural Networks**, vol. 83, pp. 21-31, 2016.
- **Zenglin Shi**, Yangdong Ye, Yunpeng Wu, Zhengzheng Lou. "Crowd Counting Using Rank-based Spatial Pyramid Pooling Network", **Acta Automatica Sinica**, vol. 42, pp. 866-874, 2016.

SAMENVATTING

Dit proefschrift is gewijd aan het onderzoeken van inductieve biases voor het leren van pixelrepresentaties. Dit proefschrift richt zich specifiek op de onderzoeksvraag: *Hoe kunnen inductieve biases voor het leren van pixelrepresentaties worden ontdekt en benut?* We ontdekten eerst drie inductieve biases en onthulden hun belang voor verschillende taken op pixelniveau, waaronder spectrale bias voor de 'deep image prior', salience-bias voor begeleide filtering en aandachtsbias voor het tellen van objecten. Naast het blootleggen van verschillende inductieve biases, proberen we vervolgens nieuwe inductieve biases te ontwikkelen door gebruik te maken van voorkennis. We hebben drie inductieve biases ontwikkeld voor het tellen van de beste objecten in hun klasse door nieuwe kennis te ontdekken. Van elk hoofdstuk wordt als volgt een korte samenvatting gegeven:

Hoofdstuk 2: Spectrale Bias van de Deep Image Prior. De 'deep image prior' toonde aan dat een willekeurig geïnitialiseerd netwerk met een geschikte architectuur kan worden getraind om inverse beeldvormingsproblemen op te lossen door simpelweg de parameters te optimaliseren om een enkel verslechterd beeld te reconstrueren. Het heeft echter twee praktische beperkingen. Ten eerste blijft het onduidelijk hoe de prior kan worden bediend buiten de keuze van de netwerkarchitectuur om. Ten tweede vereist training een orakel-stopcriterium, omdat tijdens de optimalisatie de prestatie afneemt nadat een optimale waarde is bereikt. Om deze uitdagingen aan te gaan, introduceren we een maat voor de frequentieband correspondentie om de spectrale bias van de 'deep image prior' te karakteriseren, waarbij laagfrequente beeldsignalen sneller en beter worden geleerd dan hoogfrequente tegenhangers. Op basis van onze observaties stellen we technieken voor om de uiteindelijke prestatievermindering te voorkomen en de convergentie te versnellen. We introduceren een Lipschitz-gestuurde convolutielaag en een Gaussiaans-gestuurde upsampling-laag als plug-in vervangers voor lagen die worden gebruikt in de diepe architecturen. De experimenten laten zien dat met deze veranderingen de prestaties niet verslechteren tijdens optimalisatie, waardoor we geen orakel-stopcriterium nodig hebben. We schetsen verder een stopcriterium om overbodige berekeningen te vermijden. Ten slotte laten we zien dat onze aanpak gunstige resultaten behaalt in vergelijking met de huidige benaderingen voor verschillende taken op het gebied van ruisonderdrukking, deblokkering, inkleuring, superresolutie en detailverbetering.

Hoofdstuk 3: Onscherpe Masker-Gestuurde Filterin. Het doel van dit hoofdstuk is gestuurde beeldfiltering, die het belang van structuuroverdracht tijdens het filteren benadrukt door middel van een extra begeleidingsbeeld. Waar klassiek gestuurde filters structuren overdragen met behulp van met de hand ontworpen functies, zijn recente geleide filters aanzienlijk verbeterd door parametrisch leren van diepe netwerken. De state-of-the-art maakt gebruik van diepe netwerken om de twee kerncoëfficiënten van het gestuurde filter te schatten. In dit werk stellen we dat het gelijktijdig schatten van beide coëfficiënten suboptimaal is, wat resulteert in halo-artefacten en structuurinconsistenties. Geïnspireerd door onscherp maskeren, een klassieke techniek voor randverbetering die slechts een enkele coëfficiënt vereist, stellen we een nieuwe en vereenvoudigde formulering voor van het gestuurde filter. Onze formulering geniet van een filter prior van een laagdoorlaatfilter en maakt expliciete structuuroverdracht mogelijk door een enkele coëfficiënt te schatten. Op basis van onze voorgestelde formulering introduceren we een opeenvolgend gestuurd filternetwerk, dat meerdere filterresultaten biedt vanuit een enkel netwerk, waardoor een afweging tussen nauwkeurigheid en efficiëntie mogelijk is. Uitgebreide

ablaties, vergelijkingen en analyses tonen de effectiviteit en efficiëntie van onze formulering en ons netwerk, wat resulteert in state-of-the-art resultaten voor filtertaken zoals upsamplen, ruisonderdrukking en cross-modaliteitsfiltering.

Hoofdstuk 4: Gratis Tellen met Focus. Dit hoofdstuk heeft als doel om willekeurige objecten in afbeeldingen te tellen. De leidende telbenaderingen beginnen met puntannotaties per object waaruit ze dichtheidskaarten construeren. Vervolgens transformeert hun trainingsdoel input afbeeldingen naar dichtheidskaarten via diepe convolutionele netwerken. We stellen dat de puntannotaties meer toezichtdoeleinden dienen dan alleen het construeren van dichtheidskaarten. We introduceren manieren om de punten gratis opnieuw te gebruiken. Ten eerste stellen we gesuperviseerde focus voor vanuit segmentatie, waarbij punten worden omgezet in binaire kaarten. De binaire kaarten worden gecombineerd met een netwerktak en bijbehorende loss functie om zich te concentreren op relevante gebieden. Ten tweede stellen we gesuperviseerde focus van globale dichtheid voor, waarbij de verhouding van puntannotaties tot beeldpixels in een andere tak wordt gebruikt om de algehele dichtheidsschatting te regulariseren. Om zowel de dichtheidsschatting als de focus van segmentatie te ondersteunen, introduceren we ook een verbeterde schatting voor kernel-grootte voor de puntannotaties. Experimenten met zes datasets laten zien dat al onze bijdragen de telfouten verminderen, ongeacht het basisnetwerk, wat resulteert in state-of-the-art nauwkeurigheid met slechts een enkel netwerk. Ten slotte zijn we de eersten die tellen op WIDER FACE, waardoor we de voordelen van onze aanpak kunnen laten zien bij het omgaan met verschillende objectschalen en drukte niveaus.

Hoofdstuk 5: Drie Dingen Voor het Verbeteren van op Dichtheid Gebaseerd Tellen. Drie dingen voor het verbeteren van op dichtheid gebaseerd tellen. Dit artikel gaat in op het probleem van het tellen van willekeurige entiteiten in afbeeldingen door te leren van dichtheidskaarten. Toonaangevende benaderingen vertrekken van puntannotaties per object waaruit ze de dichtheidskaarten construeren. Hun trainingsdoel transformeert vervolgens afbeeldingen naar dichtheidskaarten via neurale netwerken. In dit werk bekijken we op dichtheid gebaseerd tellen en identificeren we drie dingen die elke telbenadering beperken, en voor elk stellen we een eenvoudige module als netwerk plug-in voor als een verbetering. Concreet, (i) we vinden dat het voorspellen van dichtheden in de achtergrond meer dan de helft van het foutenpercentage bij het tellen veroorzaakt en we schetsen een cascade om het effect van het tellen op achtergrondpixels te beperken; (ii) occlusies zijn hardnekkig bij het tellen, maar komen niet vaak genoeg voor in trainingsbeelden om te leren er goed mee om te gaan. We stellen een vergroting voor van zowel invoer- als dichtheidsbeelden om te leren robuuster te zijn tegen occlusies; (iii) het construeren van dichtheidskaarten van puntannotaties met Gaussiaanse convoluties is niet optimaal voor het tellen. We stellen een alternatief voor dat leert om dichtheidskaarten te destilleren uit een hulpdichtheidsvoorspellingsnetwerk. Dergelijke gedestilleerde kaarten zijn vloeiender en robuuster tegen ruis dan hun Gaussiaanse tegenhangers. Alle drie de voorstellen zijn eenvoudig, kunnen worden aangesloten op elk dichtheidsgebaseerd telnetwerk en leveren, wanneer ze worden gecombineerd, de allernieuwste resultaten op ShanghaiTech Deel_A en Deel_B, JHU-Crowd++ en TRANCOS, met de eerste mean average fout onder de 50 op ShanghaiTech Deel A.

ACKNOWLEDGMENTS

Completing a PhD is a beautiful thing, but there have been many challenges along the journey, such as isolation, stress, failure, and pandemic. I could not have overcome these challenges without the help and support of many great people beside me, and I therefore wish to express my sincere gratitude.

First and foremost, I would like to thank my supervisor, Cees Snoek for his support and guidance. Cees has taught me a good deal of research & communication knowledge and skills, which made me grow up to be an independent and confident researcher. To name just a few, I have learned how to write a paper, a rebuttal from a reader-based perspective, how to approach a problem scientifically, and how to work smarter. He was always patient with my English, my failure, and my negative emotions. He also was always there to encourage me and never lost faith in my ability when I got rejected from a paper submission or an internship. I have a host of memories of how kind and caring he'd been, for instance, he often came to my office to check if I might feel isolated when I was sitting in a new office alone in the first few months of my PhD; He called to comfort me when he heard I lost my backpack with my ID, passport, laptop and other personal items, although he was attending a conference abroad. Cees, thank you so much for everything that you have done for me.

I would also thank my co-supervisor, Pascal Mettes. He was always happy to listen, to discuss, to give intelligent advice and suggestion whenever I needed him. Pascal, to be honest, I felt sometimes frustrated because you always had better ideas on from structuring a paper to drawing a figure. Whereas, I felt much more appreciated because this is how I learned a lot of practical research skills from you. You always provided detailed feedbacks on my every *Soos talk* presentation, and this is how I improved my presentation skills. You were also always there to encourage and support me whenever I have been through rejections or other difficulties. Pascal, it was an honor and pleasure working with you, and I truly appreciate you helping me get here.

Next, I thank the committee members for my Ph.D. defense: prof. Arnold Smeulders, prof. Zeno Geradts, prof. Peter de With, dr. Subhransu Maji, and dr. Xiantong Zhen. Thanks for your time in reading and commenting on my thesis. A special thanks goes to Arnold for his insightful questions and suggestions provided during *Soos talks*. I also thank him for his kind to organize New Year's dinners for the whole group at his place. I'm also happy to have had the opportunity to collaborate with Subhransu, and thank him for his great help in completing the Chapter 2.

Now I would like to thank all my great and cheerful friends for leaving me with numerous beautiful memories by getting together to cook, cycling, travel, and enjoy the cozy life in Amsterdam. Special thanks to Kong Xue, Song Wei, Liu Yuefeng, Huang Yifan, who were the first few friends I made in Amsterdam and are still close and dear to me, thanks for teaching me how to enjoy life. To William Thong, Sarah Ibrahimi, Riaan Zoetmulder, Sadaf Gulshad, Inske Groenen, who are more of friends than colleagues, thanks for involving me in your personal life and taking caring of me in the last four years. To Huang Jiahong, Chen Shuo, Zhang David, Shen Yixian, Chen Yunlu, Wang Qi, Yang Pengwan, who were always there to share a drink, dinner and holiday with me, thanks for your great friendship. To Wang Shihan, Wang Biwen, Hu Tao, Zhao Jiaojiao, Zhang Yunhua, Wang Yuandou, Liu Jie, Liu Yongtuo, Liu Wenfeng, Shi Zeshun, Zuo Qianru, Wang wei, Zhang Yahui, Yin Ruihong, Chu Wenjing, Guo Xiaotian, Feng Xiaoyi, Xiao Jun, Sun Yiwei, Liu Hongyun, thanks for all the great moments shared. A big thanks also

goes to my Dutch friend Nancy Jurgens, who were always adding joy to my heart whenever we had a chance to meet.

Furthermore, I would like to thank all great colleagues in VIS/ISIS Lab. I have been so fortunate to join such a nice lab and work with many kind and intelligent people. A special thanks goes to Dennis Koelma, you have been always helpful, cheerful, and enthusiastic to share your opinions and stories with all of us. A big thanks to Virginie Mes and Petra Venema, who are always there to lend a helping hand for all of us. I also thank Efstratios Gavves, Iris Groen, Yuki M. Asano, Xiantong Zhen, Stevan Rudinac, and Nanne van Noord for setting good examples of how to become a successful researcher for me. Cheers to Mehmet Altinkaya, Artem Moskalev, Mina Ghadimiatigh, Sarah Rastegar, Clemens Georg Bartnik, Hazel Doughty, Mohammadreza Salehi, Fida Thoker, Miltiadis Kofinas, Melika Ayoughi, Amber Brands, Andrew Brown, Gjorgji Strezoski, Devanshu Arya, Tom van Sonsbeek for sharing your story, thought, value and culture with me.

My final gratitude goes to my mother, father and younger brother for their incredible and unconditional love and support, for enduring my absence in the past few years, and for always being there for me. I dedicate this thesis to them.

Zenglin