UNPAIRED IMAGE DENOISING

A Project Report

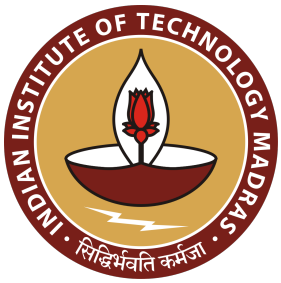
submitted by

PRIYATHAM KATTAKINDA

in partial fulfilment of the requirements

for the award of the degree of

MASTER OF TECHNOLOGY



DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS

JUNE 2020

THESIS CERTIFICATE

This is to certify that the thesis titled **UNPAIRED IMAGE DENOISING**, submitted by **PRIYATHAM KATTAKINDA**, to the Indian Institute of Technology, Madras, for the award of the degree of **Master of Technology**, is a bona fide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. A N Rajagopalan
Research Guide
Professor
Dept. of Electrical Engineering
IIT Madras, 600036

Place: Chennai

Date: 7 June, 2020

ACKNOWLEDGEMENTS

First of all, I would like to thank my guide, *Prof. A N Rajagopalan*. It was his insight that there was more work to be done in the area of unsupervised denoising, that prompted me to work on this problem in the first place. Throughout my work, he has patiently listened to my ideas, raised valid criticisms, suggested improvements. Without his constant guidance and support, I would have found it extremely difficult to make significant progress on this problem statement. For this and more, I am forever grateful to him.

I would also like to thank my friends from the Image Processing and Computer Vision Lab, Dept. of Electrical Engineering, IIT Madras. They have been a constant source of creativity and have provided me much needed motivation.

Finally, I am indebted to my family for their belief in me and their invaluable support.

ABSTRACT

KEYWORDS: Image denoising ; Unsupervised methods ; Deep learning.

Image denoising is a well-tackled problem in the domain of image processing. It has applications in many areas ranging from astronomical imaging, microscopy to present-day smartphone cameras. Traditional methods for image denoising have relied on filtering either in the spatial or a transform domain. As a result, they are fully unsupervised. The phenomenal success of deep learning in high-level computer vision tasks has led to its application in low-level image processing tasks, including image denoising. However, in stark contrast to traditional methods, deep learning approaches in image processing predominantly resort to supervised learning. A majority of methods for image denoising are no exception to this rule and hence demand pairs of noisy and corresponding clean images. Only recently has there been the emergence of methods such as *Noise2Void*, where a deep neural network learns to denoise solely from noisy images. However, when clean images that do not directly correspond to any of the noisy images are actually available, there is room for improvement as these clean images contain useful information that fully unsupervised methods do not exploit. In this work, we propose a method for image denoising in this setting. First, we use a flow-based generative model to learn a prior from clean images. We then use it to train a denoising network without the need for any clean targets. We demonstrate the efficacy of our method through extensive experiments and comparisons.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS i

ABSTRACT ii

LIST OF TABLES v

LIST OF FIGURES vii

ABBREVIATIONS viii

NOTATION ix

1 INTRODUCTION 1

2 RELATED WORK 3

2.1 Traditional methods 3

2.1.1 Non-local means (NLM) 3

2.1.2 Block-matching and 3D filtering (BM3D) 4

2.2 Deep learning based methods 5

2.2.1 Denoising convolutional neural network (DnCNN) 5

2.2.2 Noise2Noise (N2N) 6

2.2.3 Noise2Void (N2V) 7

3 BACKGROUND 9

3.1 Flow-based generative models 9

3.1.1 Additive coupling layer 10

3.1.2 Actnorm layer 10

3.1.3 Invertible 1 x 1 convolution 11

3.1.4 Affine coupling layer 11

3.1.5 Squeeze layer 11

4 OUR METHOD 13

4.1 Stage 1: Training the Flow model 13
4.2 Stage 2: Training the Denoiser 14

5 EXPERIMENTS **16**
5.1 Training Details 16
5.1.1 Stage 1 16
5.1.2 Stage 2 16
5.2 Results 17

6 CONCLUSION **20**

LIST OF TABLES

5.1 Quantitative results. We show PSNR (dB) of various unsupervised denoising methods, namely, BM3D Dabov *et al.* (2007), Noise2Void Krull *et al.* (2019), Deep image prior Ulyanov *et al.* (2018) and our method. 19

LIST OF FIGURES

1.1 Sample result from our method. Observe that the fine details in the tree are restored without any noticeable blur even when the noise level in the input is high ($\sigma = 35$). Image taken from BSD68 (Roth and Black, 2009) 2

2.1 A sample result from NLM denoising. The image on the left is the original image. The one in the middle is a noisy image ($\sigma = 15$) and the image on the right is the denoised image. Image taken from Buades *et al.* (2011). 4

2.2 A sample result from BM3D. The image on the left is the noisy image (σ = 25) and the one on the right is the denoised image. Image taken from Dabov *et al.* (2007). 5

2.3 A sample result from DnCNN. The image on the left is the original image. The one in the middle is a noisy image ($\sigma = 35$) and the image on the right is the denoised image. Image taken from Zhang *et al.* (2017). 6

2.4 A sample result from N2N. The image on the left is the original image. The one in the middle is a noisy image ($\sigma = 25$) and the image on the right is the denoised image. Image taken from Lehtinen *et al.* (2018). 7

2.5 A sample result from N2V. The image on the left is the original image. The one in the middle is a noisy image and the image on the right is the denoised image. Image taken from the supplementary material of Krull *et al.* (2019). 8

3.1 Illustration of the architecture used for learning a prior. Image taken Kingma and Dhariwal (2018). 12

4.1 An illustration of our method. In the first stage, we train a flow-based model to learn a prior distribution on clean images. Next, we use this prior along with weak supervision (see subsection 3.2) to train a denoising network. 13

5.1 Qualitative results. Here, we show the ground truth, the noisy input (Gaussian noise, $\sigma = 25$) and the denoised outputs from BM3D Dabov *et al.* (2007), Noise2Void Krull *et al.* (2019), Deep image prior Ulyanov *et al.* (2018) and finally, our method. None of these methods need supervision. 17

5.2 More qualitative results. Here, we show the ground truth, the noisy input (Gaussian noise, $\sigma = 25$) and the denoised outputs from BM3D Dabov *et al.* (2007), Noise2Void Krull *et al.* (2019), Deep image prior Ulyanov *et al.* (2018) and finally, our method. None of these methods need supervision. 18

5.3 Even more qualitative results. Here, we show the ground truth, the noisy input (Gaussian noise, $\sigma = 25$) and the denoised outputs from BM3D Dabov *et al.* (2007), Noise2Void Krull *et al.* (2019), Deep image prior Ulyanov *et al.* (2018) and finally, our method. None of these methods need supervision. 19

ABBREVIATIONS

BSD	Berkeley Segmentation Dataset
BM3D	Block-Matching and 3D Filtering
CNN	Convolutional Neural Network
dB	Decibel
DNN	Deep Neural Network
MSE	Mean Squared Error
N2N	Noise2Noise
N2V	Noise2Void
NLM	Non-Local Means
PSNR	Peak Signal-to-Noise Ratio

NOTATION

σ	Standard deviation of noise
$\boldsymbol{X}$	Clean image
$\boldsymbol{Y}$	Noisy image
$\hat{\boldsymbol{X}}$	Denoised image

CHAPTER 1

INTRODUCTION

Noise corrupts virtually any image captured through a camera. The degradation due to noise is typically captured in the equation:

$$\boldsymbol{Y} = \boldsymbol{X} + \boldsymbol{N}$$

where $\boldsymbol{X}$ is a clean image, $\boldsymbol{N}$ is noise and $\boldsymbol{Y}$ is the corresponding noisy version of $\boldsymbol{X}$. The noise $\boldsymbol{N}$ can be signal-independent, i.e., independent of $\boldsymbol{X}$ (in which case it is typically modeled as a Gaussian random variable) or signal-dependent (in this case, it is typically modeled as a Poisson random variable).

Image denoising methods attempt to recover the clean image from its noisy counterpart. Traditional methods such as BM3D (Dabov *et al.*, 2007), NSCR (Dong *et al.*, 2012), WNNM (Gu *et al.*, 2014) rely on the self-similarity of image patches to denoise solely from noisy images. Methods such as Zhang *et al.* (2017); Zhang *et al.* (2017); Lefkimmiatis (2017) that use deep learning have been proposed for image denoising. Although they achieve state-of-the-art performance, they are all discriminative models and hence require pairs of noisy images and their corresponding clean images.

Recently, deep learning methods like Noise2Noise (Lehtinen *et al.*, 2018) and Noise2Void (Krull *et al.*, 2019) have been proposed that use statistical properties of noisy image patches to eliminate noise. While these methods do not need any clean images, in situations where they are available, they cannot utilize the valuable information available in the clean images.

Another important class of methods are prior based. Priors are crucial for obtaining a reasonable answer out of all the possible solutions for an ill-posed problem such as image denoising. With handcrafted priors, these methods can be used when clean images are not available. However, these priors have been criticized as they are often chosen for their computational or analytical convenience rather than accuracy. Deep learning has allowed for constructing more accurate priors. Deep image prior (Ulyanov

et al., 2018) claims that the architecture of a convolutional neural network alone can act as a prior for natural images. Though the results are good, it is surprising as there is no mathematical justification for why this prior works. Going further, Chen *et al.* (2018) have used a GAN (Goodfellow *et al.*, 2014) to explicitly construct a prior for realistic noise which they use for denoising.

In this thesis, we propose an approach for denoising using another class of generative models, called flow-based generative models (Dinh *et al.*, 2014, 2016). These models learn an invertible transformation from a complex distribution like images to a simple one like the Gaussian distribution. Unlike GANs, flow-based models can explicitly and accurately capture the likelihood function of clean images. As a consequence, they are excellent candidates for learning a realistic prior which is essential for superior denoising performance. Also, they do not suffer from the unstable training dynamics that GANs are notorious for.

Figure 1.1 shows a sample result from our method. We train a flow-model on clean images alone while a different network is trained to denoise using only the likelihood specified by the flow-based model. As a result, our method can be used even when there is no pairing between noisy and clean images.

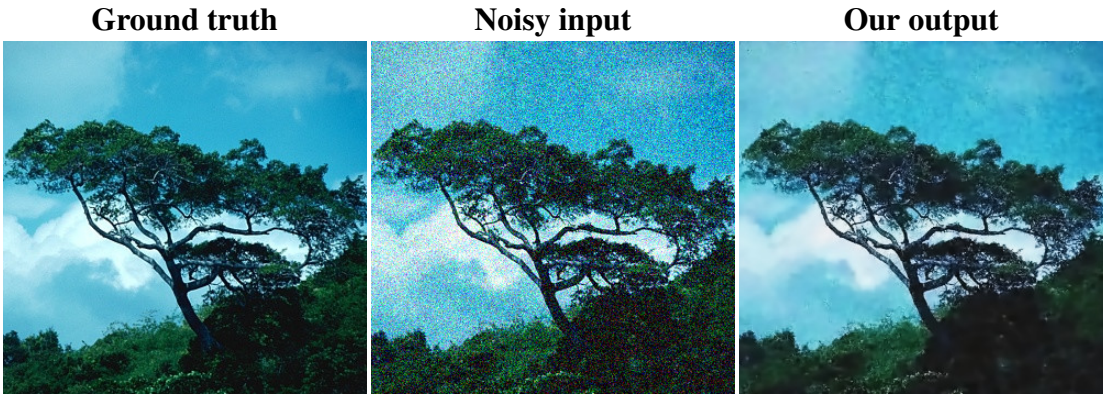


Figure 1.1: Sample result from our method. Observe that the fine details in the tree are restored without any noticeable blur even when the noise level in the input is high ($\sigma = 35$). Image taken from BSD68 (Roth and Black, 2009)

The remainder of this thesis is organized as follows: chapter 2 provides a summary of prior, related work on image denoising; chapter 3 contains the requisite background for our method; in chapter 4, we describe our method in detail and justify it; finally, in chapter 5, we show and discuss experimental results from our method.

CHAPTER 2

RELATED WORK

2.1 Traditional methods

2.1.1 Non-local means (NLM)

Non-local means filtering (Buades *et al.*, 2011) exploits the zero-mean nature of noise in images. In this method, denoising is achieved by averaging out similar pixels in the noisy image. More concretely, given a noisy image $\boldsymbol{Y}$, the denoised image $\hat{\boldsymbol{X}}$ is estimated as follows:

$$\hat{\boldsymbol{X}}(i) = \sum_{j \in N(i)} w(i,j)\boldsymbol{Y}(j) \tag{2.1}$$

where $N(i)$ is a neighborhood surrounding the pixel i and the weights $w(i,j)$ are chosen such that $0 \leq w(i,j) \leq 1$ and $\sum_j w(i,j) = 1$. The most common choice for $w(i,j)$ is the following:

$$w(i,j) = \frac{1}{W(i)} e^{-\frac{\|n(i)-n(j)\|}{k^2}}$$

Here, $n(i), n(j)$ are patches around pixels i and j respectively. The hyperparameter k controls how sensitive the weights are with respect to the differences in pixel intensities and $W(i)$ is a normalizing constant.

The summation in Equation 2.1 is only over a neighborhood rather than the whole image for purely computational reasons. The size of this neighborhood depends on the noise. For effective denoising, a larger neighborhood is required for higher noise levels. A sample output from NLM is shown in Figure 2.1.

Figure 2.1: A sample result from NLM denoising. The image on the left is the original image. The one in the middle is a noisy image ($\sigma = 15$) and the image on the right is the denoised image. Image taken from Buades *et al.* (2011).

2.1.2 Block-matching and 3D filtering (BM3D)

Block-matching and 3D filtering (Dabov *et al.*, 2007) is a very effective denoising approach that has stood the test of time in terms of denoising performance and is still widely used. Broadly speaking, the BM3D algorithm proceeds as follows:

1. Process the input image to extract reference blocks from the image.
2. Find blocks that are similar to each reference block and stack all these blocks together into a 3D array. This step is called *grouping* and each such array is called a *group*.
3. For each group, perform collaborative filtering and use the filtered blocks (from potentially several groups) to estimate the denoised image.

To get into the details, BM3D uses the Block-Matching (BM) scheme to find similar blocks. BM is very effective for motion estimation and is extensively used in video compression (such as MPEG 1, 2, 4 and H.26x) which justifies its usage here.

BM3D actually repeats the above procedure twice. In the first stage, a custom 3D transform is used. In the second stage, both the blocks from the original noisy image and from the basic estimate from the first stage are filtered together using the Wiener filter. In both stages, there can be multiple estimates for a single pixel, each estimate arising from a different group. These estimates are combined using weighted averaging. A sample output from BM3D is shown in Figure 2.2.

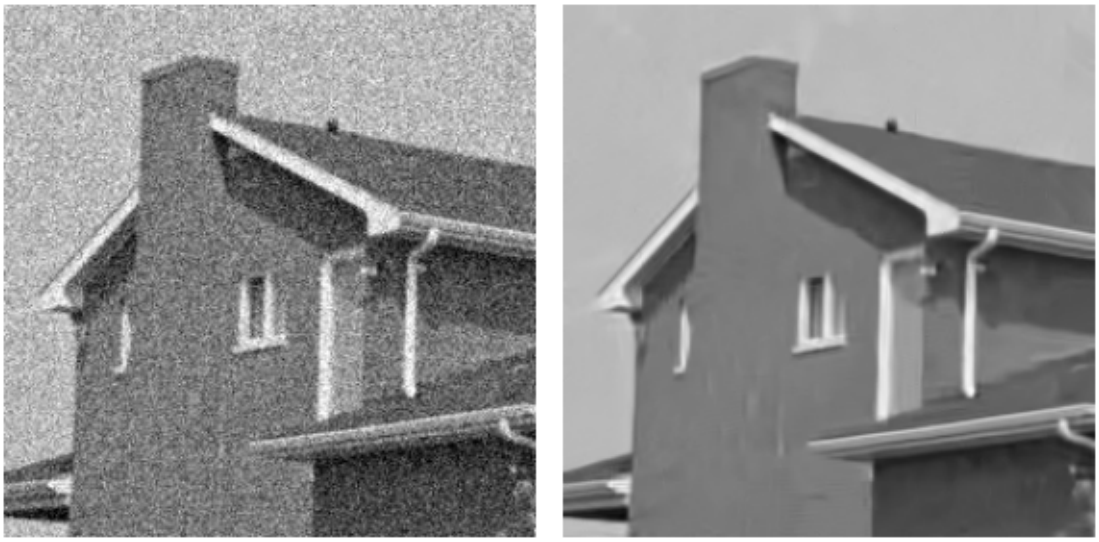


Figure 2.2: A sample result from BM3D. The image on the left is the noisy image (σ = 25) and the one on the right is the denoised image. Image taken from Dabov *et al.* (2007).

2.2 Deep learning based methods

2.2.1 Denoising convolutional neural network (DnCNN)

While using a deep neural network to denoise images was not a novel idea at that time, Zhang *et al.* (2017) were one of the first to carefully consider all the aspects of designing and training a DNN in the context of image denoising.

The authors of this paper based their network architecture on the VGG network (Simonyan and Zisserman, 2015) with some modifications. Specifically, they removed all the pooling layers and chose the depth of the network such that effective size of the receptive field of the network is large enough for denoising. They also used batch normalization to improve training speed and residual learning to improve denoising performance.

Previous works on denoising with deep learning trained different models for different noise levels. During test time, they estimated and denoised with the model that was specifically trained for that noise level. Zhang *et al.* (2017) were the first to train a single model for a wide range of noise levels. This improved both the training and testing times drastically. A sample output from DnCNN is shown in Figure 2.3

Perhaps the biggest drawback of this work was that, like the majority of discriminative

Figure 2.3: A sample result from DnCNN. The image on the left is the original image. The one in the middle is a noisy image ($\sigma = 35$) and the image on the right is the denoised image. Image taken from Zhang *et al.* (2017).

models, DnCNN also requires pairs of noisy and clean images. There are applications such as in astronomy and medical imaging where such a requirement cannot be met. This motivated the development of self-supervised/unsupervised models for image denoising some of which are summarized next.

2.2.2 Noise2Noise (N2N)

Noise2Noise (Lehtinen *et al.*, 2018) made significant progress in terms of relaxing the data requirements for image denoising using deep learning. Instead of needing pairs of noisy and corresponding clean images, N2N only needs pairs of noisy images each with different instances of noise.

It is well known that for the following optimization problem:

$$\arg\min_{\hat{X}} \mathbb{E}_X[(X - \hat{X})^2]$$

the solution is simply the expectation of X.

$$\hat{X} = \mathbb{E}_X[X]$$

The key idea in this work was to realize that, when training with an MSE loss, it is not necessary to supply clean images as targets to a DNN. As long as the expectation of the targets matches the expectation of the clean images, the network should learn to denoise. Whether the noise is Gaussian or Poisson, its mean is always zero. So, as long as the noise in an image is additive, its expectation value is, trivially, the clean image

corresponding to it.

Lehtinen *et al.* (2018) exploit this fact to train a CNN with a noisy image as input and a *different* noisy version of the same image as output. Note that it is very important for the target to be a different noisy version, as otherwise the network will simply learn an identity mapping. A sample output from N2N is shown in Figure 2.4.

Figure 2.4: A sample result from N2N. The image on the left is the original image. The one in the middle is a noisy image ($\sigma = 25$) and the image on the right is the denoised image. Image taken from Lehtinen *et al.* (2018).

2.2.3 Noise2Void (N2V)

In Noise2Void, Krull *et al.* (2019) go one step further and remove the need for a pair of images all together. Their algorithm learns to denoise just from singular noisy images, which are used as both inputs and targets while training a DNN.

Training is done patch-wise and the intensity of the center pixel of each *input* patch is replaced with the intensity of a randomly chosen nearby pixel. This effectively erases the information in the center pixel and prevents the network from learning a trivial identity mapping. Loss is also calculated only for the pixels whose intensity has been replaced. So now, during training, the same image can be used as both the input and the target. To summarize, the network learns to denoise a pixel using just the surrounding pixels. A sample output from N2V is shown in Figure 2.5.

As feeding and entire patch only to calculate loss on a single (center) pixel is very inefficient, the authors used a stratified sampling scheme that replaces the intensities

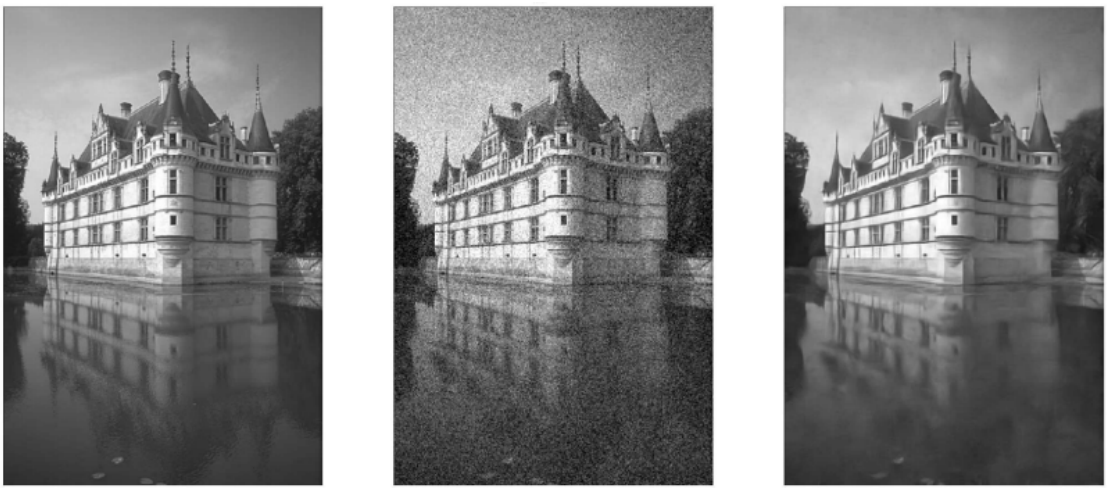


Figure 2.5: A sample result from N2V. The image on the left is the original image. The one in the middle is a noisy image and the image on the right is the denoised image. Image taken from the supplementary material of Krull *et al.* (2019).

of several randomly chosen pixels in a single patch. The scheme ensures that these pixels are not clustered together and hence the neighborhood of each of these pixels is undisturbed in the original noisy image.

CHAPTER 3

BACKGROUND

3.1 Flow-based generative models

Flow-based generative models (Dinh *et al.*, 2014, 2016) learn the bijective transformation from a high-dimensional, complicated random variable $\boldsymbol{X}$ to a latent random variable $\boldsymbol{Z}$. Typically, $\boldsymbol{X}$ represents images in a dataset while $\boldsymbol{Z}$ is assumed to be a standard normal random vector.

$$\boldsymbol{Z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}) \tag{3.1}$$

$$\boldsymbol{X} = h(\boldsymbol{Z}) \tag{3.2}$$

To learn the transformation h, the following unbiased estimate of the negative log-likelihood of $\boldsymbol{X}$ is minimized:

$$\frac{1}{N}\sum_{i=1}^{N} -\log P(\boldsymbol{x_i}) \tag{3.3}$$

Here, $\boldsymbol{x_i}$ are samples from the dataset. Using the standard rules of random variable transformation, $\log P(\boldsymbol{X})$ can be written as

$$\log P(\boldsymbol{X}) = \log P(\boldsymbol{Z}) - \log \left| \frac{\mathrm{d}h}{\mathrm{d}\boldsymbol{x}} \right| \tag{3.4}$$

where $\left|\frac{\mathrm{d}h}{\mathrm{d}\boldsymbol{x}}\right|$ is the determinant of the Jacobian of h. This term can be further decomposed when h is a composition of several other functions as is typical in a deep neural network.

$$\boldsymbol{X} = \boldsymbol{Z_0} \xrightarrow{h_1} \boldsymbol{Z_1} \xrightarrow{h_2} \boldsymbol{Z_2} \ldots \xrightarrow{h_n} \boldsymbol{Z_n} = \boldsymbol{Z} \tag{3.5}$$

$$\log P(\boldsymbol{X}) = \log P(\boldsymbol{Z}) - \sum_{i=1}^{n} \log \left| \frac{\mathrm{d}\boldsymbol{Z_i}}{\mathrm{d}\boldsymbol{Z_{i-1}}} \right| \tag{3.6}$$

To make the computation of the right hand side of (3.6) tractable, flow-based models restrict the class of transformations to those for which the Jacobian is a triangular (or

even a diagonal) matrix. We now describe various transformations that satisfy this criterion.

3.1.1 Additive coupling layer

Additive coupling layer (Dinh *et al.*, 2014) is characterized by the following transformation:

$$\boldsymbol{y}_{\boldsymbol{p_1}} = \boldsymbol{x}_{\boldsymbol{p_1}} \tag{3.7}$$

$$\boldsymbol{y}_{\boldsymbol{p_2}} = \boldsymbol{x}_{\boldsymbol{p_2}} + m(\boldsymbol{x}_{\boldsymbol{p_1}}) \tag{3.8}$$

where $\boldsymbol{x}, \boldsymbol{y}$ are the inputs and outputs of the layer respectively; p_1, p_2 is a partition of the features along the channel dimension and m is an arbitrary transformation. For this layer, it is easy to see that the Jacobian is

$$\begin{bmatrix} \boldsymbol{I}_{\boldsymbol{p_1}} & 0 \\ \frac{\mathrm{d}m(\boldsymbol{x}_{\boldsymbol{p_1}})}{\mathrm{d}\boldsymbol{x}_{\boldsymbol{p_1}}} & \boldsymbol{I}_{\boldsymbol{p_2}} \end{bmatrix} \tag{3.9}$$

where $\boldsymbol{I}_{\boldsymbol{p_1}}, \boldsymbol{I}_{\boldsymbol{p_2}}$ are identity matrices that are of the same size as the partitions p_1, p_2. Conveniently, the determinant of the matrix in (3.9) is simply 1 and hence it is ideal for use in a flow-based model.

3.1.2 Actnorm layer

In a regular network, batch normalization layers help stabilize training. However, they cannot be directly used in a flow-based model as their Jacobian is not trivial. Actnorm (short for activation normalization) layer is a modified version of batchnorm and it behaves as follows:

$$\boldsymbol{y} = \boldsymbol{s}\boldsymbol{x} + \boldsymbol{b} \tag{3.10}$$

where $\boldsymbol{x}, \boldsymbol{y}$ are the inputs and outputs of the layer respectively. Both $\boldsymbol{s}, \boldsymbol{b}$ are vectors whose dimension is the number of channels in $\boldsymbol{x}$ (or equivalently, $\mathbf{y}$).

The parameters $\boldsymbol{s}, \boldsymbol{b}$ are initialized such that $\boldsymbol{y}$ has zero mean and unit standard deviation. Note that the determinant of the Jacobian for the actnorm layer is simply the product of elements in $\boldsymbol{s}$:

$$\prod_i \boldsymbol{s_i} \tag{3.11}$$

3.1.3 Invertible 1 x 1 convolution

A convolution layer in general is not invertible. However, if the kernel size is set to 1 and the number of input and output channels is the same, then it is possible to invert the convolution. Care must however be taken that the convolution matrix is not singular. In practice, this matrix is initialized to a random invertible matrix and it is observed that it typically does not become singular during the course of training the model. The Jacobian for this layer is just the matrix itself

3.1.4 Affine coupling layer

Affine coupling layer is an extension to an additive coupling layer that acts as follows:

$$(\boldsymbol{s}, \boldsymbol{b}) = m(\boldsymbol{x_{p_2}}) \tag{3.12}$$

$$\boldsymbol{y_{p_1}} = \boldsymbol{s x_{p_1}} + \boldsymbol{b} \tag{3.13}$$

$$\boldsymbol{y_{p_2}} = \boldsymbol{x_{p_2}} \tag{3.14}$$

where $\boldsymbol{x}, \boldsymbol{y}$ are the inputs and outputs of the layer respectively; p_1, p_2 is a partition of the features along the channel dimension and m is an arbitrary transformation that is implemented as a shallow convolutional neural network.

3.1.5 Squeeze layer

Squeeze layer is exactly the same as pixel shuffle layer (Shi *et al.*, 2016). It is simply redistribution of pixels from spatial dimension to the channel dimension. Trivially, the determinant of Jacobian for this transformation is 1.

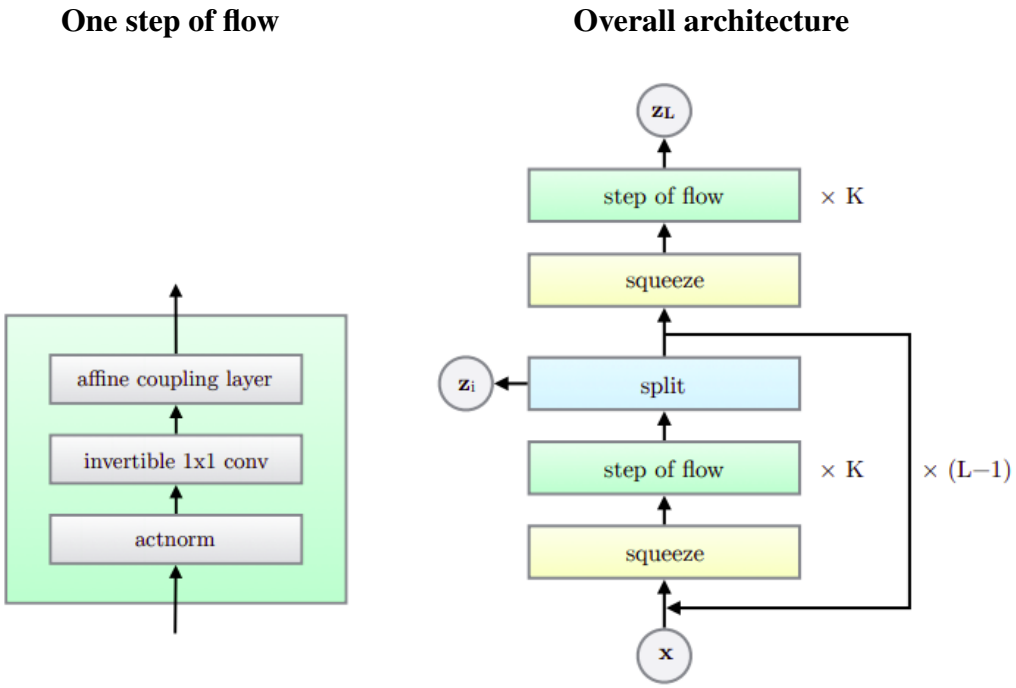


Figure 3.1: Illustration of the architecture used for learning a prior. Image taken Kingma and Dhariwal (2018).

Unlike in Dinh *et al.* (2014, 2016); Kingma and Dhariwal (2018), we do not require invertible transformations as there is no need for sampling when we are only learning a prior. Nevertheless, in our work we use the layers and formulation of flow-based models proposed in Kingma and Dhariwal (2018). An illustration of the final architecture is shown in Figure 3.1. We use $K = 32$, $L = 3$ in our work.

CHAPTER 4

OUR METHOD

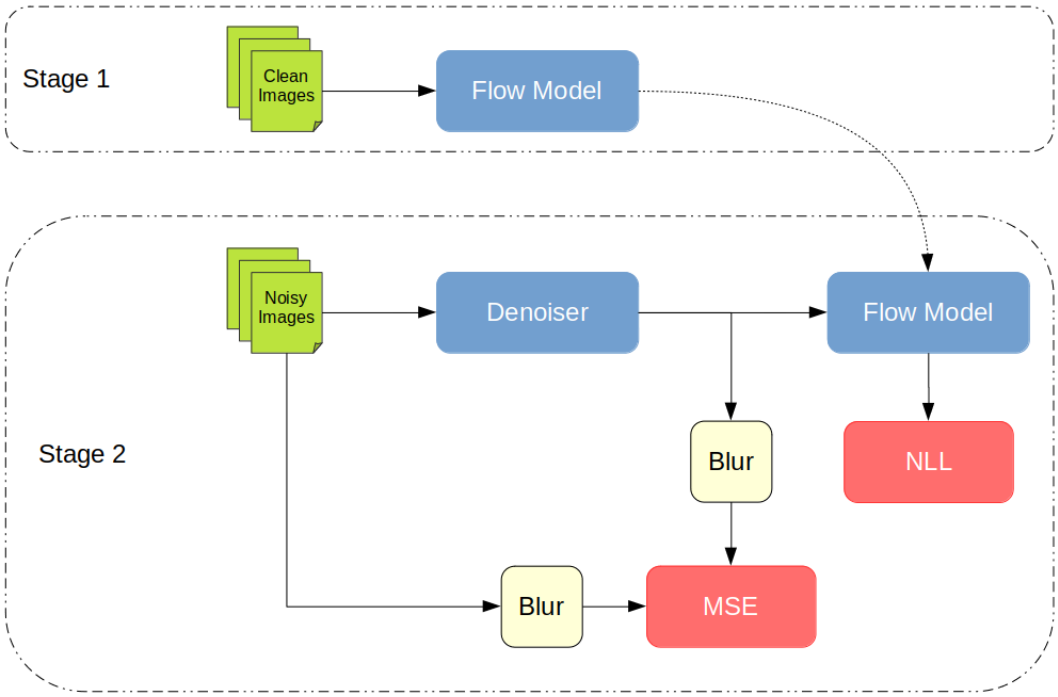


Figure 4.1: An illustration of our method. In the first stage, we train a flow-based model to learn a prior distribution on clean images. Next, we use this prior along with weak supervision (see subsection 3.2) to train a denoising network.

In this chapter we describe our two-stage approach (illustrated in Figure 4.1) to using the log-likelihood in (3.6) as a prior for image denoising.

4.1 Stage 1: Training the Flow model

First, we train a flow-based model based on clean images to learn a transformation from clean images to the standard multivariate Gaussian random variable. Due to structure of the flow-based model as described in (3.1) and the tractable probability density of a Gaussian random variable, we can evaluate (3.6) for any given image and obtain the likelihood that the image is clean.

Concretely, we train a flow-based model h to minimize the following objective:

$$-\log P(\boldsymbol{X}) = -\log P(\boldsymbol{Z}) + \sum_{i=1}^{n} \log \left| \frac{\mathrm{d}\boldsymbol{Z_i}}{\mathrm{d}\boldsymbol{Z_{i-1}}} \right| \tag{4.1}$$

$$= \frac{1}{2}\|\boldsymbol{Z}\|_2^2 + \sum_{i=1}^{n} \log \left| \frac{\mathrm{d}\boldsymbol{Z_i}}{\mathrm{d}\boldsymbol{Z_{i-1}}} \right| + C \tag{4.2}$$

where C is a constant that normalizes the Gaussian distribution. It has no bearing on the training and hence can be eliminated. Note that once the training in Stage 1 is complete, h is fixed during Stage 2.

4.2 Stage 2: Training the Denoiser

Given a noisy image $\boldsymbol{Y}$, the posterior distribution for the corresponding clean image $\boldsymbol{X}$ is

$$P(\boldsymbol{X} \mid \boldsymbol{Y}) = \frac{P(\boldsymbol{Y} \mid \boldsymbol{X})P(\boldsymbol{X})}{P(\boldsymbol{Y})} \tag{4.3}$$

To obtain the maximum a posteriori (MAP) estimate of the clean image, the denominator can be ignored and the numerator or equivalently its $\log$ value is maximized.

$$\arg\max_{\boldsymbol{X}} \log P(\boldsymbol{X} \mid \boldsymbol{Y}) = \arg\max_{\boldsymbol{X}} \log P(\boldsymbol{Y} \mid \boldsymbol{X}) + \log P(\boldsymbol{X}) \tag{4.4}$$

Assuming additive white Gaussian noise, $\log P(\boldsymbol{Y}|\boldsymbol{X})$ is simply the negative of the squared error between $\boldsymbol{Y}$ and $\boldsymbol{X}$. Using the flow model h trained in Stage 1, we can also compute the prior log-likelihood of $\boldsymbol{X}$.

Based on (4.4), we can formulate a loss function (note the change of signs as by convention, we want to minimize this loss) for the denoiser d as follows:

$$(\boldsymbol{Y} - \boldsymbol{X})^2 - \lambda \log P(\boldsymbol{X}) \tag{4.5}$$

where λ is a hyperparameter that controls the relative importance of the conditional and the prior probability distributions. To be mathematically precise, λ depends on the noise level in the image.

To facilitate the use of a single λ for a range of noise levels, we modify the first term in (4.5) to instead measure the squared error between blurred versions of $\boldsymbol{X}$ and $\boldsymbol{Y}$.

Intuitively speaking, we are training d to copy only the low frequency information from the input $\boldsymbol{Y}$ while adding details that make the output $\boldsymbol{X}$ to look more clean. The flow model h dictates what details are added to $\boldsymbol{Y}$.

The final form of the loss function we use for the denoiser d is

$$(B(\boldsymbol{Y}) - B(\boldsymbol{X}))^2 - \lambda \log P(\boldsymbol{X}) \tag{4.6}$$

Here B is a local mean filter, the size of which is chosen to be 3×3, as that gave the best performance on the validation set.

CHAPTER 5

EXPERIMENTS

5.1 Training Details

We use the validation set of MS COCO (Lin *et al.*, 2014) for our training. Of the 41K images it contains, we use a subset of 20K images as our clean image dataset. We add Gaussian noise to another subset of 20K images to form our noisy image dataset. As we want our method to be agnostic to noise level, for each image, the standard deviation of the added noise is chosen uniformly in the interval $[0, 50]$. We set aside the remaining 1K images for validation to tune the hyperparameters λ and the size of the local mean filter.

5.1.1 Stage 1

We use the architecture described in Kingma and Dhariwal (2018) for the flow-based model. We feed patches of size 32 from images in the clean dataset as input to this model. Using the loss in (4.2), we train for 100 epochs using the Adam optimizer (Kingma and Ba, 2014) with learning rate $= 1 \times 10^{-3}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$.

5.1.2 Stage 2

We use the ResNet He *et al.* (2016) for our denoiser. Because the flow-based model only accepts fixed size inputs and the ResNet does not change input size, we use input patches of size 32. In this stage, however, they are extracted from noisy images. Using (4.6), we train only the denoiser, for 100 epochs using the Adam optimizer with the same parameter settings as in stage 1. Based on our experiments on the validation set, we choose $\lambda = 1.5 \times 10^{-6}$ and local mean filter of size 3×3.

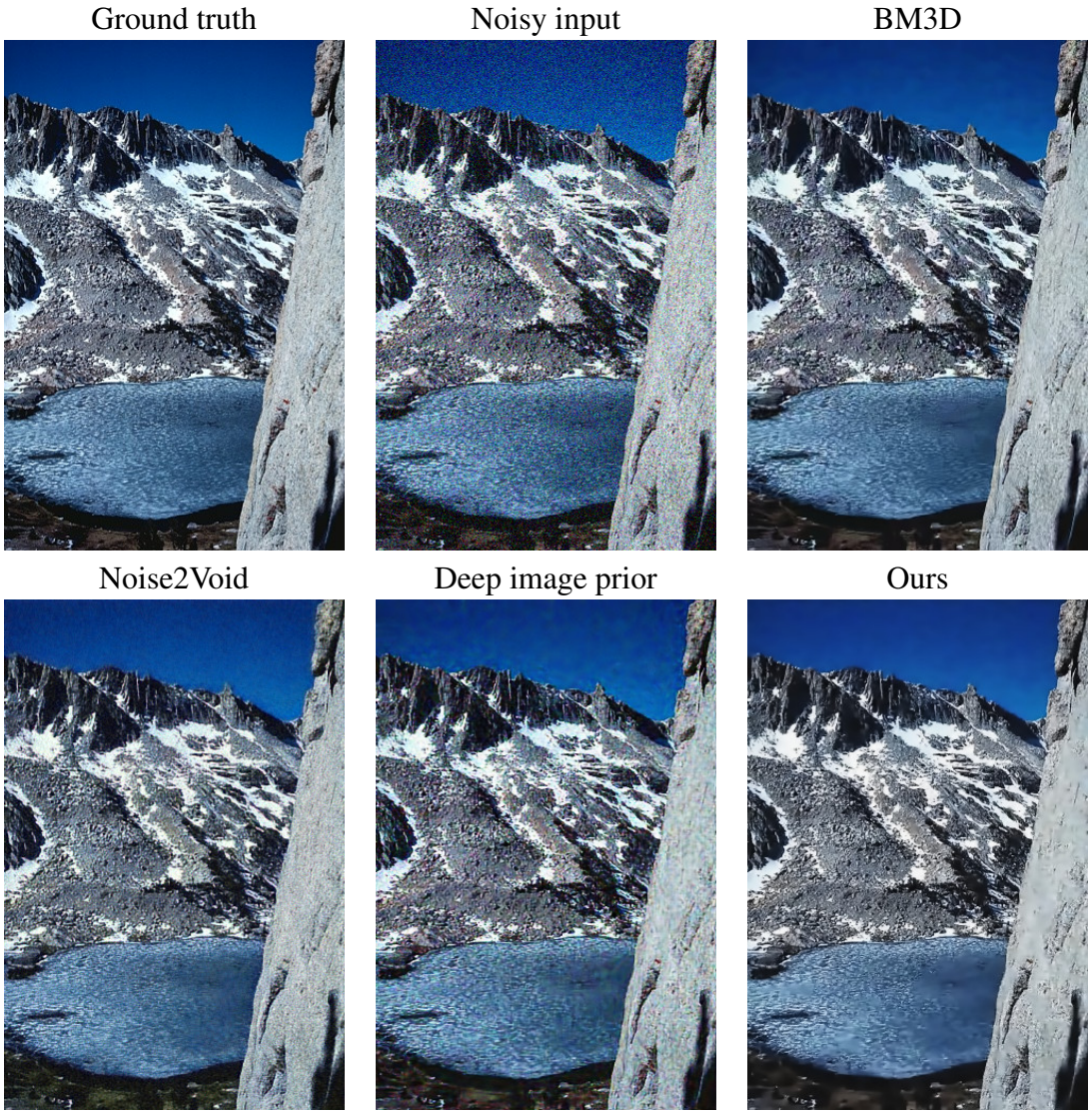


Figure 5.1: Qualitative results. Here, we show the ground truth, the noisy input (Gaussian noise, $\sigma = 25$) and the denoised outputs from BM3D Dabov *et al.* (2007), Noise2Void Krull *et al.* (2019), Deep image prior Ulyanov *et al.* (2018) and finally, our method. None of these methods need supervision.

5.2 Results

Following Dabov *et al.* (2007); Krull *et al.* (2019), we evaluate our method on the BSD68 dataset (Roth and Black, 2009) for different noise levels and compare it with BM3D (Dabov *et al.*, 2007), Noise2Void (Krull *et al.*, 2019), Deep image prior (Ulyanov *et al.*, 2018). All comparisons are made using either results reported in the respective papers or those obtained from running the code that the authors have generously shared. Table 5.1 shows the average PSNR values of different methods for images from BSD68. Although PSNR is not an accurate metric for perceptual quality, our method performs

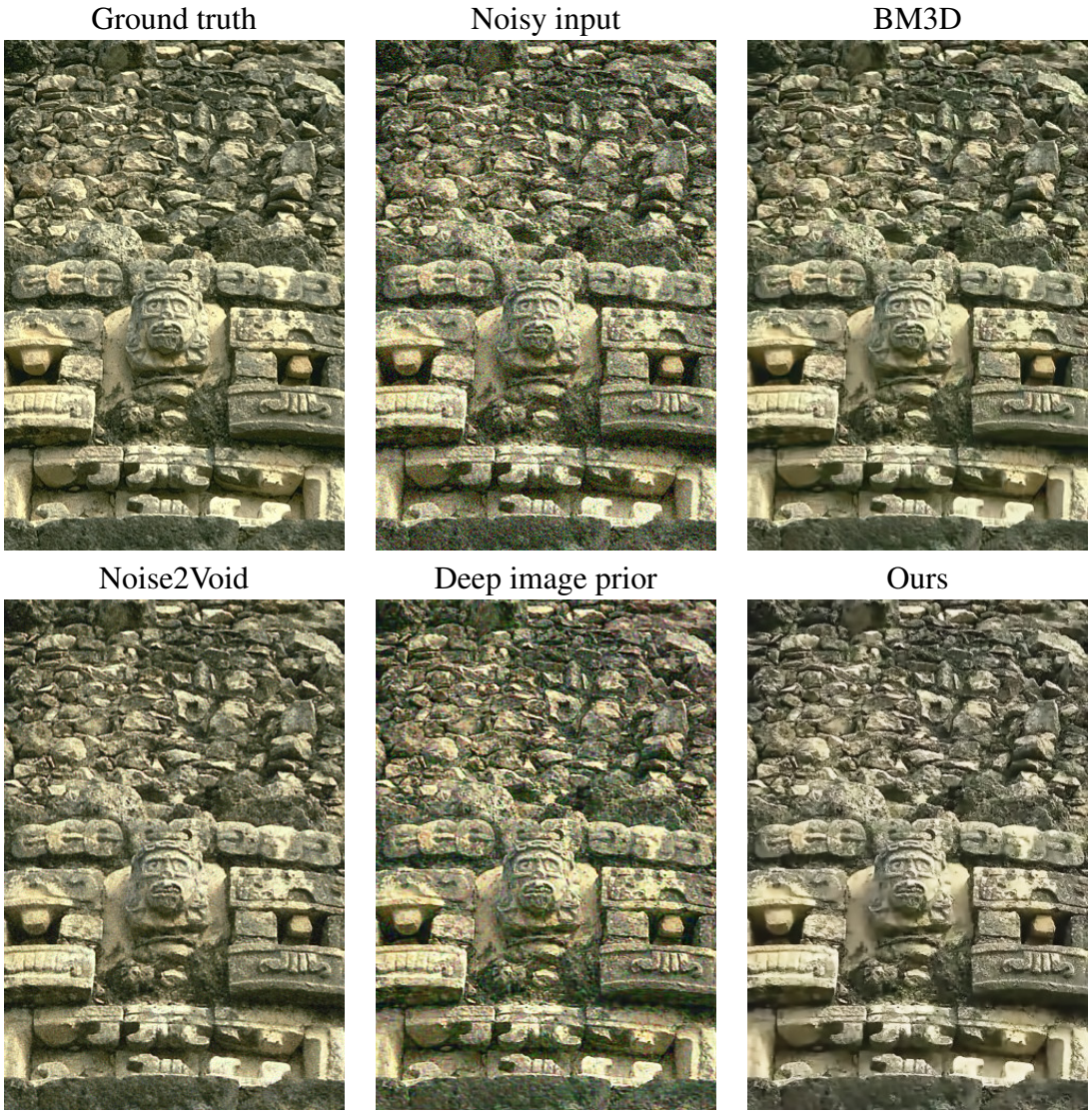


Figure 5.2: More qualitative results. Here, we show the ground truth, the noisy input (Gaussian noise, $\sigma = 25$) and the denoised outputs from BM3D Dabov *et al.* (2007), Noise2Void Krull *et al.* (2019), Deep image prior Ulyanov *et al.* (2018) and finally, our method. None of these methods need supervision.

competitively with Noise2Void and is better than Deep image prior.

Figure 5.1, Figure 5.2 and Figure 5.3 show qualitative comparison of our results with other methods. Our method is able to remove noise effectively without blurring any textures, details or sharp edges (this is obvious in the sky in the first set of images). Deep image prior produces outputs that still have visible noise. Noise2Void, although better than Deep image prior, fails in some cases. An example of this is the blades of grass in the third set of images where the output from Noise2Void is noticeably desaturated.

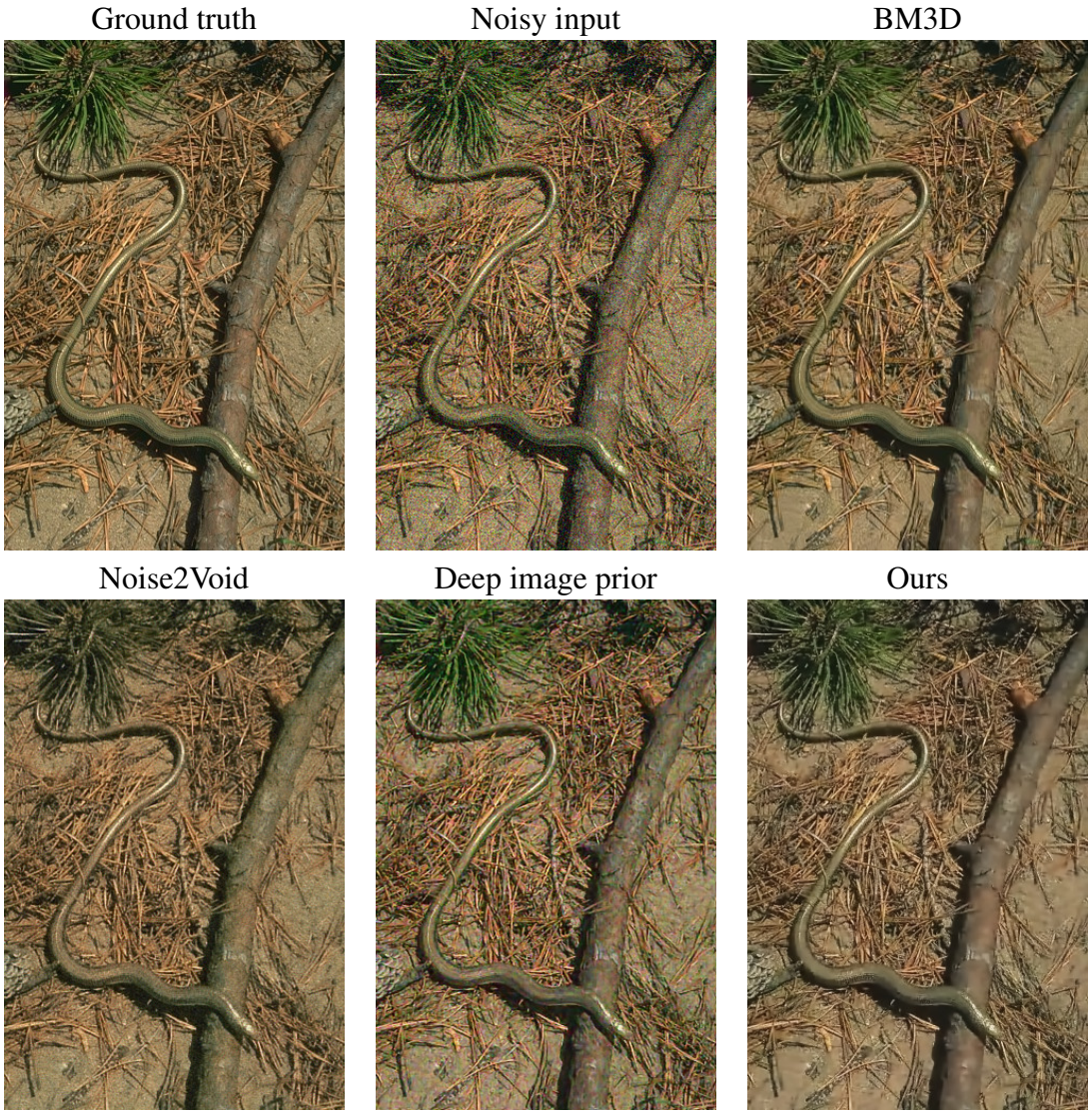


Figure 5.3: Even more qualitative results. Here, we show the ground truth, the noisy input (Gaussian noise, $\sigma = 25$) and the denoised outputs from BM3D Dabov *et al.* (2007), Noise2Void Krull *et al.* (2019), Deep image prior Ulyanov *et al.* (2018) and finally, our method. None of these methods need supervision.

Method	BM3D	N2V	DIP	Ours
$\sigma = 15$	33.14	28.92	27.58	29.10
$\sigma = 25$	30.22	27.68	26.6	28.61
$\sigma = 35$	28.25	26.51	25.97	26.2

Table 5.1: Quantitative results. We show PSNR (dB) of various unsupervised denoising methods, namely, BM3D Dabov *et al.* (2007), Noise2Void Krull *et al.* (2019), Deep image prior Ulyanov *et al.* (2018) and our method.

CHAPTER 6

CONCLUSION

We have proposed the use of flow-based model as a mathematically justifiable and realistic prior for image denoising. We have conducted qualitative and quantitative experiments on the BSD68 (Roth and Black, 2009) dataset that reveals the competitive performance of our method.

Motivated by our success, we conjecture that using a flow-based model prior should be effective for solving other image restoration tasks such as image deblurring and super-resolution in an unsupervised fashion.

REFERENCES

1. **Buades, A.**, **B. Coll**, and **J.-M. Morel** (2011). Non-Local Means Denoising. *Image Processing On Line*, **1**, 208–212.

2. **Chen, J.**, **J. Chen**, **H. Chao**, and **M. Yang**, Image blind denoising with generative adversarial network based noise modeling. *In 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018. ISSN 1063-6919.

3. **Dabov, K.**, **A. Foi**, **V. Katkovnik**, and **K. Egiazarian** (2007). Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Transactions on Image Processing*, **16**(8), 2080–2095. ISSN 1941-0042.

4. **Dinh, L.**, **D. Krueger**, and **Y. Bengio** (2014). NICE: Non-linear Independent Components Estimation. *arXiv e-prints*, arXiv:1410.8516.

5. **Dinh, L.**, **J. Sohl-Dickstein**, and **S. Bengio** (2016). Density estimation using Real NVP. *arXiv e-prints*, arXiv:1605.08803.

6. **Dong, W.**, **L. Zhang**, **G. Shi**, and **X. Li** (2012). Nonlocally centralized sparse representation for image restoration. *IEEE transactions on Image Processing*, **22**(4), 1620–1630.

7. **Goodfellow, I.**, **J. Pouget-Abadie**, **M. Mirza**, **B. Xu**, **D. Warde-Farley**, **S. Ozair**, **A. Courville**, and **Y. Bengio**, Generative adversarial nets. *In Advances in neural information processing systems*. 2014.

8. **Gu, S.**, **L. Zhang**, **W. Zuo**, and **X. Feng**, Weighted nuclear norm minimization with application to image denoising. *In Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014.

9. **He, K.**, **X. Zhang**, **S. Ren**, and **J. Sun**, Deep residual learning for image recognition. *In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

10. **Kingma, D. P.** and **J. Ba** (2014). Adam: a method for stochastic optimization. corr abs/1412.6980 (2014).

11. **Kingma, D. P.** and **P. Dhariwal**, Glow: Generative flow with invertible 1x1 convolutions. *In Advances in Neural Information Processing Systems*. 2018.

12. **Krull, A.**, **T. Buchholz**, and **F. Jug**, Noise2void - learning denoising from single noisy images. *In 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019. ISSN 1063-6919.

13. **Lefkimmiatis, S.**, Non-local color image denoising with convolutional neural networks. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017.

14. **Lehtinen, J.**, **J. Munkberg**, **J. Hasselgren**, **S. Laine**, **T. Karras**, **M. Aittala**, and **T. Aila**, Noise2Noise: Learning image restoration without clean data. *In* **J. Dy** and **A. Krause** (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*. PMLR, Stockholmsmässan, Stockholm Sweden, 2018. URL http://proceedings.mlr.press/v80/lehtinen18a.html.

15. **Lin, T.-Y.**, **M. Maire**, **S. Belongie**, **L. Bourdev**, **R. Girshick**, **J. Hays**, **P. Perona**, **D. Ramanan**, **C. L. Zitnick**, and **P. Dollár** (2014). Microsoft COCO: Common Objects in Context. *arXiv e-prints*, arXiv:1405.0312.

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

17. **Shi, W.**, **J. Caballero**, **F. Huszár**, **J. Totz**, **A. P. Aitken**, **R. Bishop**, **D. Rueckert**, and **Z. Wang**, Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. *In Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

18. **Simonyan, K.** and **A. Zisserman**, Very deep convolutional networks for large-scale image recognition. *In International Conference on Learning Representations*. 2015.

19. **Ulyanov, D.**, **A. Vedaldi**, and **V. Lempitsky**, Deep image prior. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018.

20. **Zhang, K.**, **W. Zuo**, **Y. Chen**, **D. Meng**, and **L. Zhang** (2017). Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing*, **26**(7), 3142–3155. ISSN 1941-0042.

21. **Zhang, K.**, **W. Zuo**, **S. Gu**, and **L. Zhang**, Learning deep cnn denoiser prior for image restoration. *In Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017.

LIST OF PAPERS BASED ON THESIS

1. **Priyatham Kattakinda** and **A N Rajagopalan**, Unpaired Image Denoising. *IEEE International Conference on Image Processing*. 2020.