Non-blind Deblurring using CNN

A Project Report

submitted by

Venkatesh Reddy Maligireddy
(EE13B041)

in partial fulfilment of the requirements

for the award of the degree of

BACHELOR OF TECHNOLOGY

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS

APRIL 2017

THESIS CERTIFICATE

This is to certify that the thesis titled **Non-blind Deblurring using CNN**, submitted by **Venkatesh Reddy Maligireddy**, to the Indian Institute of Technology, Madras, for the award of the degree of **Bachelor of Technology**, is a bonafide 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:

ACKNOWLEDGEMENTS

This work would not have been possible without the guidance and the help of several people. I take this opportunity to extend my sincere gratitude to all those who made this thesis possible.

First, I would like to thank all my teachers who bestowed me with good academic knowledge. I am indebted to my advisor Prof. A.N.Rajagopalan whose expertise, generous guidance and support made it possible for me to work on a topic that was of great interest to me. I would also like to thank my lab mate and dear friend Subeesh Vasu for sharing his valuable ideas and helping me whenever I am stuck with some problem.

I would like to thank my family for giving support and guidance all through my life. I would also like to thank all my friends and well-wishers for helping me in difficult times and being a good source of support and guidance.

ABSTRACT

Proper handling of outliers is an important problem that needs to be handled in case of non-blind deblurring, which is important in avoiding ringing artifacts in the restored image. This artifacts are a result of noisy kernels that come with most of the natural images. Blind motion deblurring methods are primarily responsible for recovering an accurate estimate of the blur kernel. Non-blind deblurring (NBD) methods, on the other hand, attempt to faithfully restore the original image, given the blur estimate. However, NBD is quite susceptible to errors in blur kernel. Many of the existing works tried to solve this problem but are limited to a single noisy kernal. In this work, we present a convolutional neural network-based approach to handle kernel uncertainty in non-blind motion deblurring. We provide multiple latent image estimates corresponding to different prior strengths obtained from the given blurry observation in order to exploit the complementarity of these inputs for improved learning. To generalize the performance to tackle arbitrary kernel noise, we train our network with a large number of real and synthetic noisy blur kernels. Our network mitigates the effects of kernel noise so as to yield detail-preserving and artifact-free restoration.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Motion blur is a common and unpleasant corruption that occurs in hand-held photography and is very hard to undo. Image deblurring has been an active area of research in terms of both blind and non-blind de-blurring. Blind image deblurring methods aim to recover the blur kernel as well as the clean image. Since the dimension of the kernel is much smaller than image size, one can better constrain the estimation of the blur kernel rather than the image [18]. Hence, most existing blind deblurring (BD) approaches [5, 10, 15, 21, 32, 34, 37, 39] try to recover an accurate motion estimate from the blurred image [16]. This is eventually used to recover the latent image using an off-the-shelf non-blind deblurring (NBD) method.

The objective of NBD is to recover a sharp latent image from a known blurred image and a given blur estimate. Over the last decade, there has been significant progress in this direction. Recent works [14, 18, 26, 27, 40] have come up with new image priors that can model the statistics of natural images which helps in suppressing ringing artifacts. Few works have even attempted to handle outliers such as noise [6, 38], saturated regions [6, 34, 38], and compression artifacts [38] to improve the quality of the deblurred result. Most of the existing non-blind methods are tailored to perform well under the assumption that the motion is known accurately. Consequently, they tend to underperform when the motion estimate is noisy. Because of the highly ill-posed nature of BD, an approach that can deliver accurate (close to ground truth) motion estimate is still far from reality [16]. The blur estimates from BD come with varying degrees of noise which can introduce serious artifacts in the restored output. This necessitates the need to tune NBD methods for optimal performance in the presence of noisy blur estimates which is a very cumbersome task.

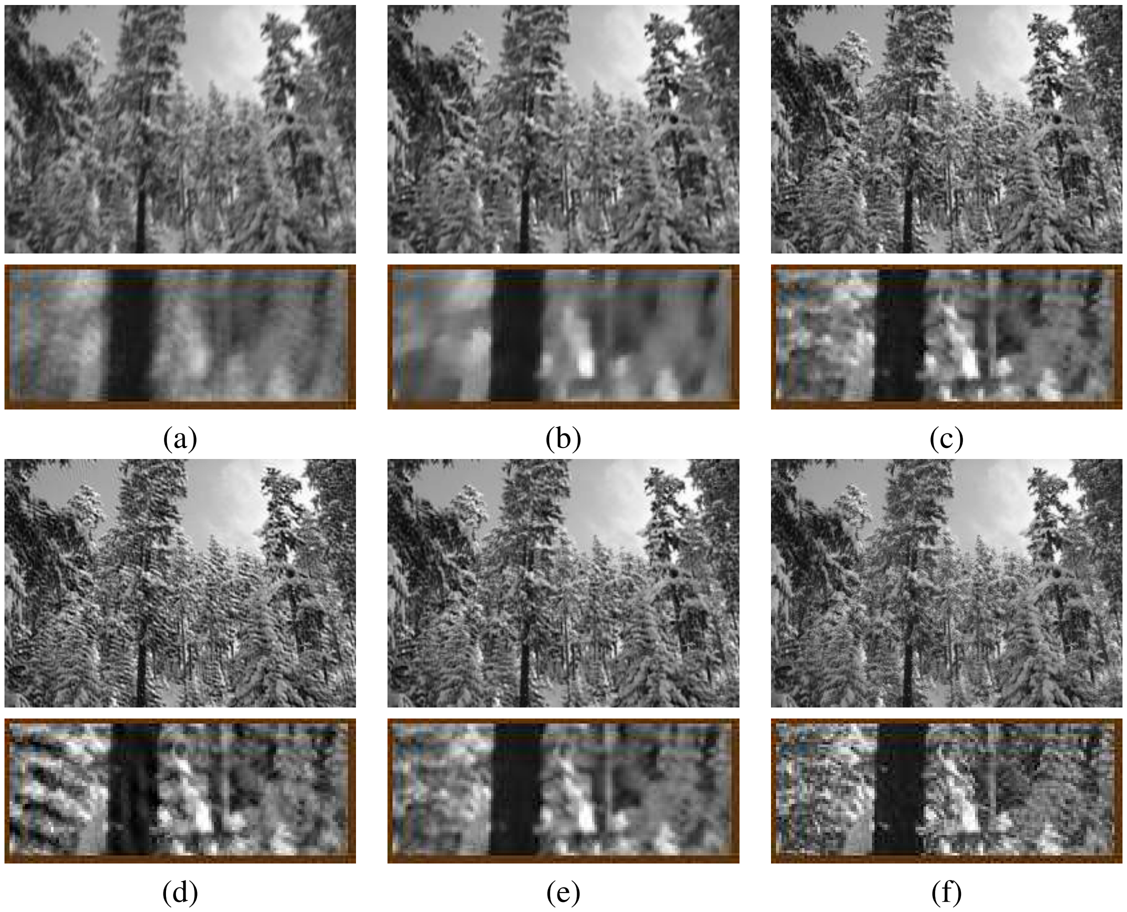

Figure 1.1: Preserving details in non-blind deconvolution. (a) Input blurry image. (b-d) Inputs to our network representing restored images with different prior weights. Restored results from (e) EPLL [40], and (f) our approach.

In this work, our focus is on improving restoration quality in the presence of a noisy blur kernel. The approach that we propose consists of a conventional non-blind deconvolution unit followed by a deep convolutional neural network (CNN) to remove undesired artifacts caused by errors in motion estimate. Our approach is based on the premise that deblurred images obtained with different prior strengths carry complementary information. The complementarity lies in the fact that restored images with low prior weight preserve details but suffer from artifacts. On the other hand, a large prior weight helps in artifact removal but at the cost of image details. Hence, we are motivated to use multiple images obtained from the same blurred image-kernel pair, but with different prior strengths as inputs to our network. This we believe allows the network to perform better feature discrimination and restoration as compared to the single input image case. Recent works on single image based restoration ([8, 9, 38]) have already revealed the potential of CNN based feature learning. The fact that we need discriminatory feature learning from multiple inputs renders CNN as a natural choice.

We also propose an approach to generate synthetic noisy kernels that can closely mimic the behavior of noise in real kernel estimates. We employ thousands of such synthetic noise kernels to improve the performance of our network and its generalization capability.

1.1 Summary of contributions

The following is the summary of the major contributions of this thesis:

- In chapter 3, we discuss a deep learning based approach to improve the restoration quality of motion blurred images over existing non-blind deblurring methods. The core idea of the method including use of multiple restored images as input to the network, and generation of the right kind of training data which we mention in this chapter was not our contribution and it was done by another member in our lab. However we mention it for completeness. Our contribution was mainly to try out different architectures and tune the hyper-parameters of the network so as to come up with a working network architecture that can perform the restoration task well.

1.2 Related works

1.2.1 Non-blind deblurring

Classical methods such as Wiener filter [35] and Richardson-Lucy deconvolution [24] are known to cause ringing artifacts. Most of the works on NBD resort to maximum a posteriori (MAP) estimation, with differences in the type of the image prior they employ. While most existing works use global image priors (typically in the form of $||\nabla l||^{\alpha}$, where ∇l represents image gradient) [14,17,33], the use of local (patch-based) priors [40] has been more effective in non-blind deconvolution. Earlier works have attempted to use Gaussian distributions ($\alpha = 2$) on the gradients of image. For the Gaussian prior, the presence of closed-form solution in the frequency domain allowed to recover the image very quickly. However, the assumption of Gaussian prior is very often violated by natural images leading to results of mediocre quality. The work in [33] has shown that Laplacian prior ($\alpha = 1$) is more effective in image restoration, and can produce good results in a reasonable time. However, recent studies have shown that the gradients of natural images have significantly heavier tails than a Laplacian, and can be well modeled by a hyper-Laplacian [17] $(0.5 \leq \alpha \leq 0.8)$. Both [17] and [14] have used sparse (hyper-Laplacian) prior on image gradients to perform non-blind deconvolution. While [17] uses an iteratively reweighed least squares (IRLS) algorithm to enforce the hyper-Laplacian prior on image gradients, [14] have proposed to speed up the computation using either look up tables or analytic formulas. The state-of-the-art work in [40] uses a patch prior learned from natural images to perform state-of-the-art image restoration. They have employed a Gaussian mixture model (GMM) with 200 components over patches of size 8×8 to learn the patch prior.

Few other works have tried to handle the outliers present in the blurred image, while doing non-blind deconvolution. The works in [6, 34] has shown that improper handling of outliers results in artifacts in the restored images. The work in [6] tried to handle the presence of noise as well as saturation using a variational expectation maximization scheme. [34] employed auxiliary variables in the Richardson-Lucy method to handle saturation regions present in the blurred images.

1.2.2 Learning based non-blind deblurring

The work in [29] showed that better outlier-robust image restoration can be done by learning to predict the latent image from L_2 norm based solution. But they were unable to generalize for arbitrary kernels as well as noisy kernels since the performance scope of their network was limited only to a single kernel. Xu et al. [38] have employed 1D filter based deep network to perform an outlier-robust non-blind deconvolution entirely within a single network but was designed to work only for a single kernel. Some of the recent works on non-blind deblurring have employed learning frameworks based on integrated Bayesian model [28], Gaussian conditional random fields [27], [25] and shrinkage fields [26], but can handle only small level of noises in the blur kernels.

1.2.3 Learning based restoration

Recent works using deep neural networks have been showing promising results in other image restoration tasks as well. [2] shows that decent results on denoising can be achieved using plain multi-layer perceptrons. [36] uses a stacked denoise autoencoder to obtain promising results both for denoising and inpainting. The work in [9] uses a convolutional neural network (CNN) architecture to handle strong noises such as raindrop and lens dirt. The works in [7, 8] achieve state-of-the-art results on super-resolution using CNNs. The work in [23] propose Shepard networks to achieve performance improvement for both inpainting and super-resolution. While the works in [4, 30] have used deep networks for blind-deblurring, [3] perform dehazing.

Chapter 2

Neural Networks and Deep learning

Neural networks are a class of machine learning models that are used to approximate real-valued, discrete valued and vector-valued functions. Motivated by the biological neural networks which are a set of interconnected neurons that enable information processing in organisms, artificial neural networks are presented as a set of interconnected nodes (also called neurons) which exchange messages with each other. The connections have numeric weights that can be tuned based on experience, making neural nets adaptive to inputs and capable of learning.

A deep neural network (DNN) is an artificial neural network with multiple hidden layers of units between input and output layers. DNNs attempt to model highlevel abstraction in data by using multiple processing layers, with complex structures composed of multiple non-linear transformations. Deep learning is a part of a family of machine learning methods based on learning representations of data. An observation (e.g., an image) can be represented in many ways such as a vector of intensity values per pixel, or in a more abstract way as a set of edges, regions of particular shape, etc. Some representations are better than the others at simplifying the learning task from examples. One of the promises of deep learning methods is replacing hand-crafted features with better feature representations that are automatically learnt for a given task.

Deep learning methods have seen a recent surge in success and popularity due to advances in Graphics Processing Units (GPU) hardware as well as new and improved methods for training them. A type of feed-forward neural network called convolutional neural network (CNN) is extensively used in computer vision and has shown state-of-the-art results in many problems such as object detection, image calssification, image denoising, etc. In this work, we explore the use of CNNs in non-bind deblurring of images.

2.1 Modeling one neuron

One of the popular models for neurons in an Artificial Neural Network is the sigmoid model. The perceptrons (neurons) with out this sigmoid function at the end are essentially linear classifiers. The perceptron training rule will converge if the data is linearly separable. However, for overlapping data, convergence is not assured. Here we require our neural network to learn non-linear functions.

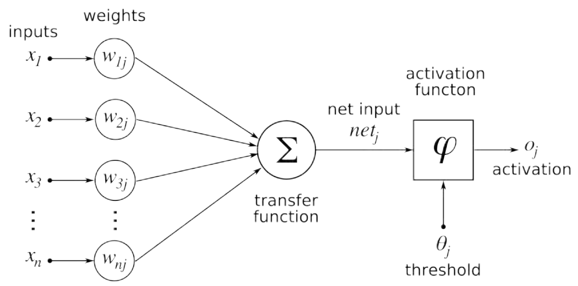

Figure 2.1: Artificial Neuron model

A new model of neuron (the sigmoid model) was introduced to circumvent the problem. The sigmoid model is illustrated in Fig.2.1. The sigmoid unit first computes a linear combination of its inputs, then applies a threshold to the result. In the case of sigmoid unit, however, the threshold output is a continuous function of its input. More precisely, the sigmoid unit computes its output *O* as

$$o(\vec{x}) = \sigma(\vec{w}, \vec{x})$$

$$\sigma(y) = \frac{1}{1 + e^{-y}}$$

The sigmoid function has a very useful property that its derivative is easily expressed in terms of its output.

$$\frac{d\sigma(y)}{dy} = \sigma(y)(1 - \sigma(y))$$

Other activation functions which are generally used include tanh function.

2.1.1 Training sigmoid neurons

For the sigmoid neurons to predict output given an input vector, it becomes necessary to estimate the weights $\{w_i\}_0^n$ based on a set of training examples D. Let us denote t_d to be the target output for the training example d. The set D is essentially the set of ordered pairs (d,t_d). In order to optimise w_i's, we need to specify a measure of training error relative to the training examples. One common measure of error between the predicted and the true output is the mean square error between them.

$$E(\vec{w}) = \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

Here $\vec{w} = (\vec{w_0}, \vec{w_1} ... \vec{w_n})$ is the weight vector to be learnt. We can optimize this error using the Gradient Descent Algorithm. In gradient descent, the error function is minimized by starting with an arbitrary weight vector, then iteratively updating the weight vector by moving in the direction of steepest descent along the error surface. The direction of the steepest descent is essentially the vector derivative of E with respect to $\vec{w}$.

$$\Delta E(\vec{w}) = \left(\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1} \cdots \frac{\partial E}{\partial w_n} \right)$$

$$\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i} \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2 = \frac{\partial}{\partial w_i} \frac{1}{2} \sum_{d \in D} (t_d - \sigma(\vec{w}, \vec{x_d}))^2$$

$$\frac{\partial E}{\partial w_i} = \sum_{d \in D} (t_d - o_d)(o_d(1 - o_d)(-x_{id}))$$

Here x_{id} is the i^{th} component of the vector $\vec{x_d}$. Now the iterative training rule can be written as

$$w_i \longleftarrow w_i + \eta \sum_{d \in D} (t_d - o_d)(o_d(1 - o_d)(-x_{id}))$$

2.2 Multilayer network

In the previous section, we have discussed a prominent model for neuron which is the fundamental unit in an ANN. By combining multiple units in a cascade, we obtain a model that will learn complex functions. Fig 2.2 shows one such network composed of 4 layers - one input layer, followed by 2 hidden layers which is then connected to an output layer. For the rest of this chapter, we shall consider only sigmoid neurons.

2.2.1 Backpropagation Algorithm

In 2.1.1 we have discussed gradient descent algorithm for training single sigmoid unit. To train multilayered network, we follow a similiar approach, but apply gradient updates in a layered fashion - for layer j, gradient output from layer j+1 is taken as input, gradient is then computed with respect to this output and then it is passed in to layer j-1. Since gradient

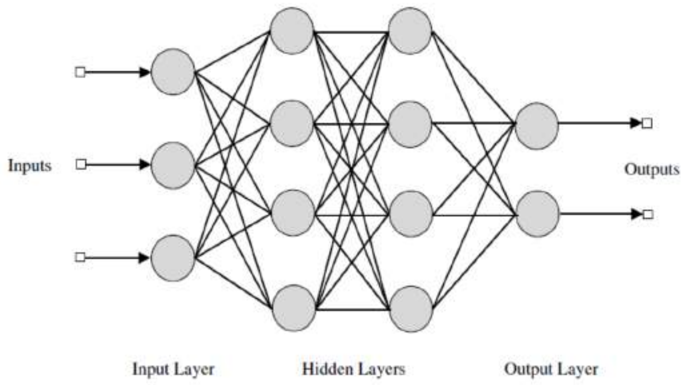

Figure 2.2: A neural network

propagates backwards, this algorithm is called ”Backpropagation Algorithm”.

Let us now derive the update rules for the backpropagation algorithm. The notation adopted for this section is given below.

- x_{ij} = i^{th} input to unit *j*
- w_{ij} = weight associated with i^{th} input to unit *j*
- net_j = the weighted sum of inputs for unit $j(\sum_i w_{ji}x_{ji})$
- o_j = output computed by unit *j*
- t_j = target output for unit *j*

As before, we use mean squared error between the target and output units as our error function. But now, since the output layer consist of multiple units, we need to compute the average error over all the units.

$$E = \frac{1}{2} \sum_{d \in D} \sum_{k \in outputs} (t_d - o_d)^2$$

Let us derive the update rule for a single training example- the overall gradient is the summation of gradients corresponding to all training examples.

$$E_d = \frac{1}{2} \sum_{k \in outputs} (t_d - o_d)^2$$

$$\begin{aligned} \frac{\partial E_d}{\partial w_{ij}} &= \frac{\partial E_d}{\partial net_j} \frac{\partial net_j}{\partial w_{ij}} \\ &= \frac{\partial E_d}{\partial net_j} x_{ij} \end{aligned}$$

Case 1: Training rule for output weights

In case of output weights, w_{ij} can influence the network only through net_j and net_j can influence the network only through o_j. Hence,

$$\frac{\partial E_d}{\partial net_j} = \frac{\partial E_d}{\partial o_j} \frac{\partial o_j}{\partial net_j}$$

$$\frac{\partial E_d}{\partial o_j} = \frac{\partial}{\partial o_j} \frac{1}{2} \sum_{k \in outputs} (t_d - o_d)^2$$

$$= -(t_j - o_j)$$

$$\frac{\partial o_j}{\partial net_j} = \frac{\partial \sigma(net_j)}{\partial net_j}$$

$$= o_j(1 - o_j)$$

Summing everything, we get

$$\frac{\partial E_d}{\partial net_j} = -(t_j - o_j)(o_j(1 - o_j)$$

Case 2: Training rule for hidden unit weights

In case of hidden units, the derivative of the training rule for w_{ji} must take into account the indirect ways in which w_{ji} can influence the network outputs and hence E_d. For this reason, we will find it useful to refer to the set of all units immediately downstream of unit *j* in the network. We can then write

$$\frac{\partial E_d}{\partial net_j} = \sum_{k \in Downstream(j)} \frac{\partial E_d}{\partial net_k} \frac{\partial net_k}{\partial net_j}$$

Let δ_k denote $-\frac{\partial E_d}{\partial net_j}$ Then

$$\begin{aligned} \delta_k &= - \sum_{k \in Downstream(j)} \frac{\partial E_d}{\partial net_k} \frac{\partial net_k}{\partial net_j} \\ &= - \sum_{k \in Downstream(j)} -\delta_k \frac{\partial net_k}{\partial o_j} \frac{\partial o_j}{\partial net_j} \\ &= o_j(1 - o_j) \sum_{k \in Downstream(j)} \delta_k w_{kj} \end{aligned} \tag{2.1}$$

We can combine all these derived update rules to obtain the Backpropagation algorithm.

Chapter 3

The Proposed Method

Our attempt is to improve the performance of non-blind image deblurring task, from the knowledge of a noisy kernel estimate and blurred image. Although this is the most common scenario to occur at the final stage of a blind-deblurring problem, a little attention has been given to resolve this issue. We propose to use a deep learning network to remove kernel noise specific characteristics present in the deconvolved images such as ringing artifacts thereby improving the image restoration quality. Consider the commonly used convolution model of blurring where the latent image l is related to the blurred image b through a blur kernel k as

$$b = l * k \tag{3.1}$$

where '*' refers to the convolution operation. Our approach for image restoration consists of two units, a conventional non-blind deconvolution unit followed by the use of a deep CNN to remove the undesired artifacts. In our experiments, we use the hyper-Laplacian prior based method in [14] , to obtain the restore image from first unit. The choice of this method was motivated by the following reasons; among the works that employ global priors, [14] gives the state-of-the-art results, and as compared to the state-of-the-art work in [40] this method yields close results while consuming significantly lesser computational time. [14] estimates the latent image from known values of b and k using the following optimization framework.

$$\hat{l} = \arg\min \lambda ||l * k - b||^2 + ||\nabla l||^\alpha \tag{3.2}$$

where λ is a scalar which determines the weight of the data cost over the prior term, ∇ represent the gradient operator, and α (< 1) refers to the norm value of the prior term. For the hyper-Laplacian prior, we use $\alpha = 2/3$, the value at which the restoration quality was found to be the best as reported in [14].

3.1 Motivation

Our idea of using deep CNN to improve the output from such a deconvolution is motivated by the fact that, λ plays a crucial role in the final image restoration quality. Figure 3.1 illustrates the differences in the restored images while we vary the value of λ over a range of values. We have used an example image (Fig. 3.1(a)) and a ground truth kernel (Fig. 3.1(b)) to produce the blurred image in Fig. 3.1(c). The restored image using the noisy kernel in Fig. 3.1(d) is shown in Figs. 3.1 (e-g). While higher value of λ results in restored image with more details, it is also affected by serious level of ringing artifacts since the weight for the prior term is negligible. Decrease in λ increases the influence of prior term reducing the ringing artifacts but leading to loss of high frequency details. Although, for ground truth kernel higher value of λ can produce results with no ringing artifacts, most of the time the availability of ground truth kernel is not guaranteed. In practice, the non-blind deblurring is used as the final image restoration stage of a blind-deblurring problem, and the blur kernels returned by deblurring problems are typically noisy.

Our attempt is to improve the quality of latent image estimate under this scenario, by making use of a deep learning network. Note that, unlike the kind of artifacts that have been

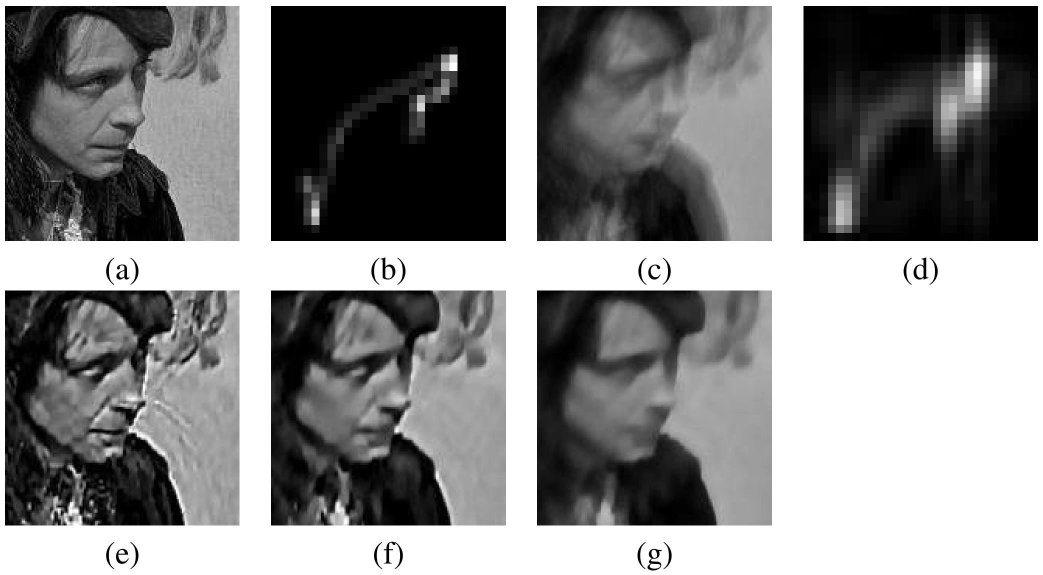

Figure 3.1: Impact on the restored image quality with varying λ. (a) Latent image. (b) Ground truth kernel. (c) Blurred image generated using (a) and (b). (d) Noisy kernel obtained from [32]. Estimated images using the kernel in (d) with λ = (e) 2e4, (f) 2e3, and (g) 2e2.

addressed in previous works on denoising [2, 36] or dirt removal [9], the variability of the artifacts produced by kernel noises are much more complicated. Hence coming up with a deep network which can generalize such a task for arbitrary kernels is not easy. This is one of the main reason why the earlier works ([29, 38]) which tried to handle outliers in non-blind deconvolution was unable to generalize the performance for arbitrary kernels. Our attempt is to train the network such that the performance improvement is applicable to arbitrary kernels also.

By giving the restored images instead of the blurry images, we can easily remove the kernel dependency of the network, since the artifacts in them is introduced by the noise in the estimates alone. However this alone does not solve the problem entirely. This is because the variabilities introduced by the noise in kernels have more complex patterns making it difficult to learn it directly through a deep CNN. To begin with, let us assume that we are trying to improve the estimated image with high λ. Now the artifacts in the input image have very complex patterns and hence a network which was learned with such inputs cannot promise high performance when it comes to generalization for arbitrary noises in kernel. In contrary, if we use the estimated image with lower λ, the details lost cannot be retrieved back since the mapping is much more complicated as compared to the other restoration tasks like super-resolution [7]. Hence our attempt to use a single image input based restoration was a partial failure, since the improvement that was able to achieve was minor. We have also observed that, image restoration by removing the artifacts (i.e., using high λ input) gives better performance as compared to attempting to recover the details (i.e., using low λ input).

Our key finding is that, giving multiple restored images corresponding to different values of λ can lead to significant improvement in restoration quality. For example consider the restored images in Fig. 3.1 (e) and Fig. 3.1 (g). By comparing with Fig. 3.1 (g) it is more easy to identify the regions in Fig. 3.1 (e) corresponding to artifacts and high frequency details. The same intuition work with deep networks also. Giving more inputs enable the network to easily differentiate between desirable and undesirable features leading to significant improvement over the case of single image based training.

3.2 Network Architecture

We initially started experimenting with the network as shown in Fig. 3.3 where we give multiple inputs concatanated with varying λ. In this case we feed the network with multiple inputs and expect it to learn the necessary features blindly. Later we moved onto a more intelligent and intuitive network where we divided the whole network into *Feature extraction* unit followed by *Feature discrimination and recombination* unit as shown in the Fig. 3.2.

Since we divided the feature extraction unit and reconstruction unit, there is less work on the network in the feature extraction phase and it can take the necessary features from the corresponding input images and combine them to get the estimated image. First, we use the blurry image and noisy kernel estimate to generate multiple estimates of the latent

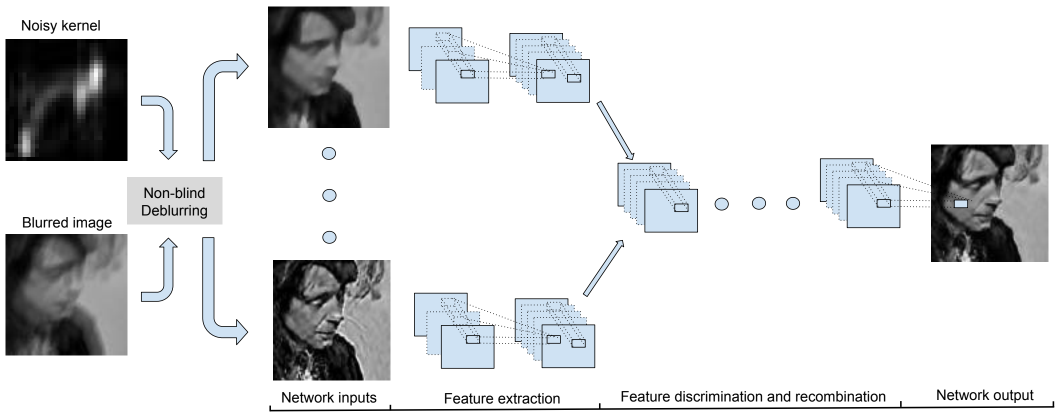

Figure 3.2: Our network structure. We use the deblurred images obtained from a standard NBD scheme but with different prior strengths as inputs to our network. Our FCN has a feature extraction unit followed by feature discrimination and recombination to yield the final result.

image using a conventional NBD scheme [14]. These multiple image estimates are passed through individual feature extraction units formed of two convolutional layers. The extracted features are then combined and passed through a number of convolutional layers which act as a feature discrimination unit using which the desired artifact-free features are integrated to yield the final restored image.

The feature extraction unit corresponding to each input consists of two convolutional layers, with 64 and 128 filters in the first and second layer, respectively. Thus, when we use a network with m inputs, the feature extraction units will output $128m$ number of feature channels. The feature discrimination unit takes $128m$ feature maps from the feature extraction unit as input. Our feature discrimination unit consists of 7 convolutional layers with the number of filters in each layer being 512, 512, 512, 512, 128, 64, and 1, respectively. At the output of every convolution layer (except the last layer), we apply batch-normalization followed by ReLu [11]. For all the layers, we use filters of size 3×3 with a stride of 1 and zero-padding by 1, to maintain spatial resolution over the entire network. The hyper-parameter settings that we use are partially motivated from the encoder-decoder architecture used in [12], although our network has significant differences in terms of structure. Before arriving at our final network, we conducted experiments to tune the parameters (such as the number of layers, number of filters in each layer, filter size, number of feature extraction units etc.). A detailed discussion on the various architectures that were attempted is provided in the next section.

3.3 Analyzing Performance of Different Network Structures

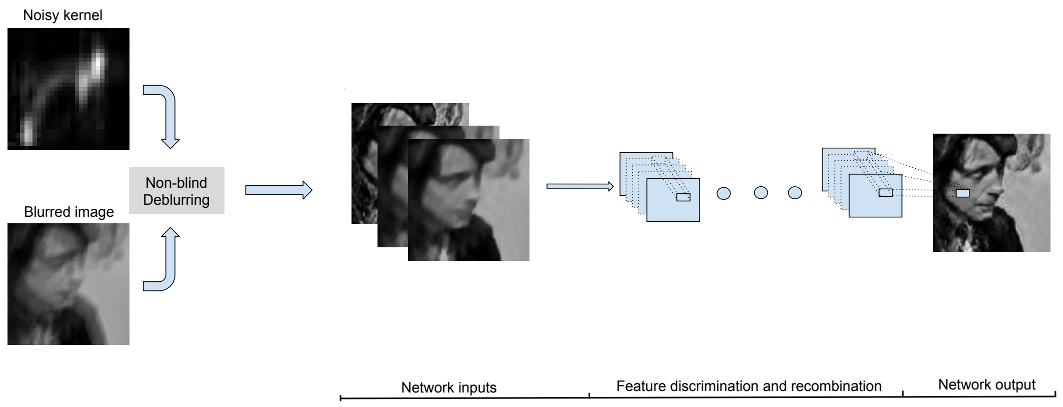

Figure 3.3: Initial network architecture we tried. We use the deblurred images obtained from a standard NBD scheme but with different prior strengths as inputs to our network. This network does not include a feature extraction unit that is present in our final architecture

Before arriving at our final network architecture, we had attempted different variations of our network of which one was mentioned in the previous section (Fig.3.3). After changing the network into *Feature Extraction* and *Recombination* layers there are few variations that we tried on the final network to improve performance, details of which are provided here. Their structure is described in the format of 'number of filters (filter size)' for each layer. In the following description, the layers inside curly braces represent feature extraction units for a single input, C refers to 64 filters, C_q refers to $64q$ filters, and A_n represent n^{th} network structure.

$\boldsymbol{A_1}$) 2 layers of feature extraction for each input (our final network), over all network structure: $3 \times \{C(3 \times 3) - C_2(3 \times 3)\} - C_8(3 \times 3) - C_8(3 \times 3) - C_8(3 \times 3) - C_8(3 \times 3) - C_2(3 \times 3) - C_1(3 \times 3) - 1(3 \times 3)$

$\boldsymbol{A_2}$) 3 layers of feature extraction for each input: $3 \times \{C_1(3 \times 3) - C_2(3 \times 3) - C_2(3 \times 3)\} - C_8(3 \times 3) - C_8(3 \times 3) - C_8(3 \times 3) - C_2(3 \times 3) - C_1(3 \times 3) - 1(3 \times 3)$

$\boldsymbol{A_3}$) 1 layer of feature extraction for each input: $3 \times \{C_2(3 \times 3)\} - C_8(3 \times 3) - C_8(3 \times 3) - C_8(3 \times 3) - C_8(3 \times 3) - C_8(3 \times 3) - C_2(3 \times 3) - C_1(3 \times 3) - 1(3 \times 3)$

$\boldsymbol{A_4}$) NO individual feature extraction units: $C_3(3 \times 3) - C_6(3 \times 3) - C_8(3 \times 3) - C_8(3 \times 3) - C_8(3 \times 3) - C_8(3 \times 3) - C_2(3 \times 3) - C_1(3 \times 3) - 1(3 \times 3)$

$\boldsymbol{A_5}$) 2 layers of feature extraction for each input, with larger filter size at input layer: $3 \times \{C_1(5 \times 5) - C_2(3 \times 3)\} - C_8(3 \times 3) - C_8(3 \times 3) - C_8(3 \times 3) - C_8(3 \times 3) - C_2(3 \times 3) - C_1(3 \times 3) - 1(3 \times 3)$

$\boldsymbol{A_6}$) 2 layers of feature extraction for each input, with larger filter size at output layer: $3 \times \{C_1(3 \times 3) - C_2(3 \times 3)\} - C_8(3 \times 3) - C_8(3 \times 3) - C_8(3 \times 3) - C_8(3 \times 3) - C_2(3 \times 3) - C_1(3 \times 3) - 1(5 \times 5)$

$\boldsymbol{A_7}$) 2 layers of feature extraction for each input, with one layer removed from the middle: $3 \times \{C_1(3 \times 3) - C_2(3 \times 3)\} - C_8(3 \times 3) - C_8(3 \times 3) - C_8(3 \times 3) - C_2(3 \times 3) - C_1(3 \times 3) - 1(3 \times 3)$

$\boldsymbol{A_8}$) 2 layers of feature extraction for each input, with one layer added at the middle: $3 \times \{C_1(3 \times 3) - C_2(3 \times 3)\} - C_8(3 \times 3) - C_8(3 \times 3) - C_8(3 \times 3) - C_8(3 \times 3) - C_8(3 \times 3) - C_2(3 \times 3) - C_1(3 \times 3) - 1(3 \times 3)$

A network with no individual feature extraction units (A_4) will try to discriminate the features from multiple inputs at the first layer itself. Whereas, for the networks with individual feature extraction units, the feature discrimination starts from the junction of all individual feature extraction units. Fig. 3.4 displays the PSNR values of test data for each of these networks. As can be observed, the inclusion of feature extraction units ($A_1 - A_3$) for each input resulted in improvement in PSNR. However, the performance improvement

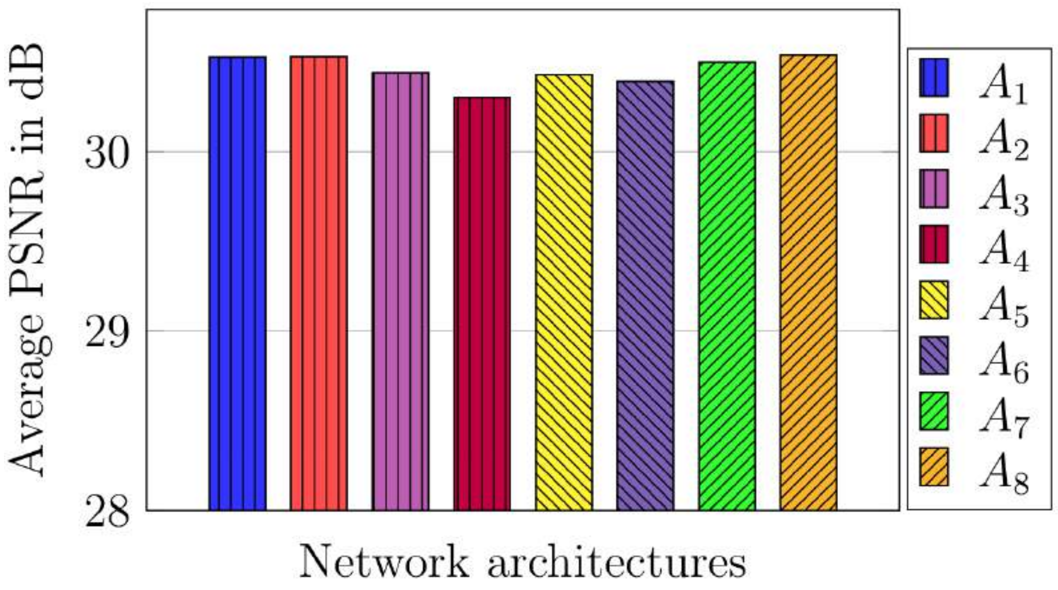

Figure 3.4: Performance of different network architectures.

begins to narrow down when we use more than 2 feature extraction layers. To explore the scope for performance improvement by using filters of larger spatial extent, we tried to increase the filter size at input and output layer, respectively ($A_5 - A_6$). However, for both cases, increasing the filter size lead to minor reduction in performance. The performance of the network had a direct influence on the total number of layers used. To display the sensitivity of network performance with respect to our final network, we compare the performance of two networks obtained by removing and adding one layer each, with respect to our final network ($A_7 - A_8$). As is evident, the performance is inferior for the network with one layer less. Although there exists minor improvement in PSNR for network with more layers, we chose to proceed with our final architecture as a trade-off between computational complexity and performance. Scope exists to improve the performance with additional layers.

3.4 Training

We started our experiments using the code from [12] and modified it to our requirements. We implemented multiple input network architecture and modified the data loading file particular to our use case. We used L_2 norm of the difference between the network output and ground truth sharp image to define the loss for training. We used ADAM [13] solver for training with *Batch size* of 4, *Learning rate* of 0.0002 and *Momentum* of 0.5 as solver parameters.

Before giving the inputs to the network a simple normalization of inputs by multiplying the image pixel values with 2 and subtracting the result by 1 was done to make the input pixel values between *[-1,1]* as a preprocessing step. To nullify the effect of this normalization, a post-processing step is done on the output image from network at the testing phase. In most of our cases an intensity compensation of the output image is also required as another post-processing step with any of the inputs to get the best result. Because of the GPU memory requirements, we tested the images by splitting them patch wise and later combining them to reconstruct the total image after doing the post-processing steps mentioned above.

3.4.1 Data Generation

As we have already discussed, we use the images estimated by [14] for different values of λ as inputs to our network. Images from the BSD 500 dataset [1] are used as latent images for generation of training and test data. While the first 400 images were used for training, the remaining 100 images were used for testing. In order to learn a network which can generalize well to arbitrary noises in the kernels, we need to use a large number of realistic noisy kernels to generate the training data. This formed part of work of another student in the lab. As noisy kernel estimates, we used 3200 kernels returned by the BD methods in [4, 15, 19, 32] while deblurring images from the dataset in [32]. We chose a subset of these noisy kernels for generation of test data. Note that the set of kernels and images used for generation of test data is kept different from the ones used for generation of training data. For generation of training as well as test data, we use ground truth kernels to form the corresponding blurred images, and the noisy kernel estimates to obtain image estimates. We followed patch-wise training in which randomly cropped patches of size 101×101 from the estimated images were used as inputs to our network.

3.5 Identifying the input for optimal performance

In order to identify the input images that can deliver best restoration quality, we trained our network with different input combinations. We began our experiments by training with single image inputs and then proceeded to add more inputs to the network. Based on visual inspection of image quality, we chose λ = 2e4, 2e3, and 2e2 in [14] to generate multiple inputs. This is because: (i) [14] has shown that $\lambda = 2e3$ yields optimal restoration quality, and (ii) the restored results corresponding to $\lambda = 2e2$ and $2e4$ had significant differences with respect to the optimal λ.

Fig. 3.5 illustrates differences in the restoration quality when we train the network with the best performing 1, 2, and 3 input cases. Fig. 3.6 displays the average peak signal-to-noise ratio (PSNR) value (averaged over all images in the test data) of the network outputs while training with different input combinations. Out of all possible single-input cases, the input corresponding to λ = 2e4 yielded the highest PSNR. Among all the input pairs, it was found that 2e2+2e3 performed the worst (mainly due to lack of details in both the inputs) whereas 2e3+2e4 performed the best with a marginal improvement over 2e2+2e4.

When we use a single input corresponding to $\lambda = 2e4$, the network was able to achieve only partial artifact removal (Fig. 3.5 (d)). Whereas with 2 inputs (2e3+2e4) most of the artifacts went away (Fig. 3.5(e)). While the PSNR improvement for the 2-input case was significant, this was not true when we moved from 2 to 3 inputs. We also observed that

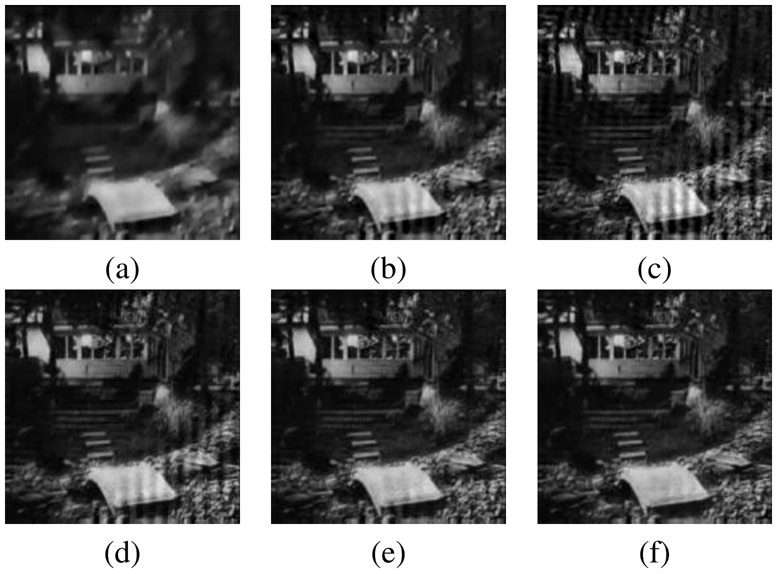

Figure 3.5: Performance of our network with different input combinations. Inputs to the network with (a) $\lambda = 2e2$, (b) $\lambda = 2e3$, and (c) $\lambda = 2e4$. Output of the network trained using (d) single input with $\lambda = 2e4$, (e) two inputs with $\lambda = 2e3, 2e4$, and (f) three inputs.

further addition of inputs yields only a marginal improvement in PSNR. A close inspection of the result in Fig. 3.5 (f) reveals the importance of the third input ($\lambda = 2e2$) in removing artifacts in Fig. 3.5 (e). Among all possible input combinations, the 3-input network was observed to yield best performance (Fig. 3.5 (f)) and was hence adopted for analyzing our network further.

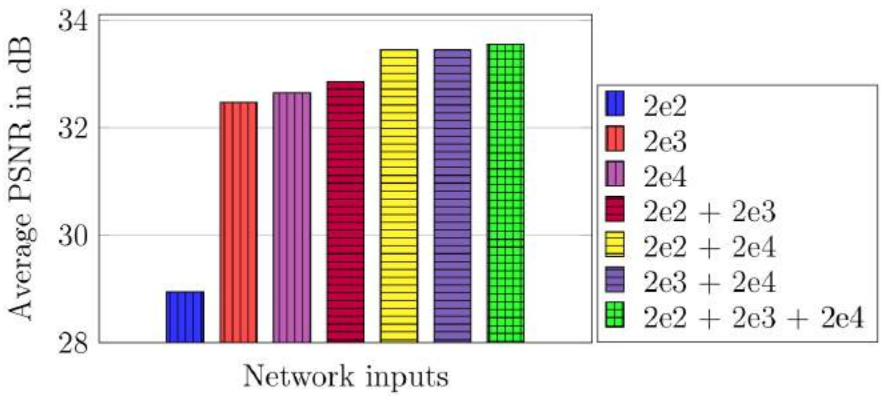

Figure 3.6: Network performance as a function of inputs.

3.6 Comparisons with existing methods

In this section, we use network trained with synthetic and real, as well as low and high-noise kernels. Training of our network was done using gray scale images. To obtain results for color images, following others, we recombined the restored results of each channel. To evaluate the performance of our proposed approach, we use the publicly available datasets in [16, 19, 32]. The dataset in [19] contains 32 low resolution blurred images formed from 8 kernels and 4 latent gray-scale images. [32] contains 640 high resolution blurred images formed from 8 kernels and 80 latent gray-scale images. The dataset in [16] contains 100 high resolution blurred color images.

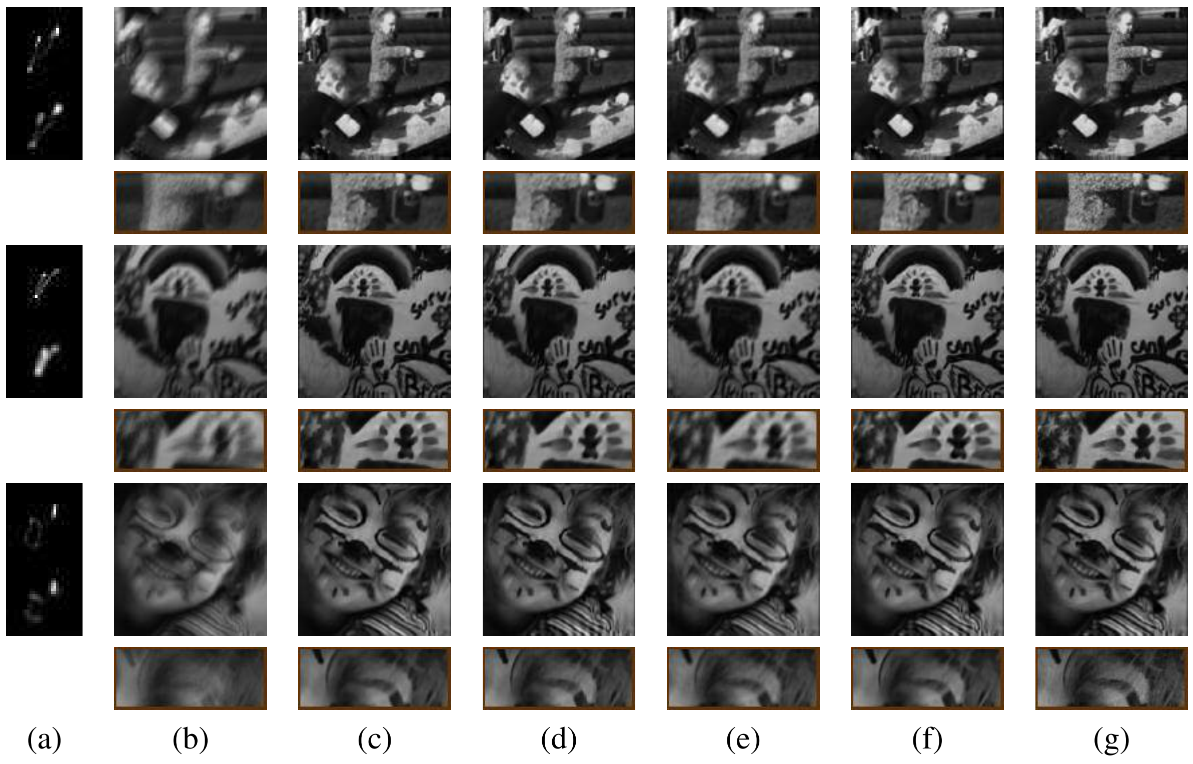

Figure 3.7: Examples from the dataset in [19]. (a) Ground truth (top) and estimated kernels (bottom). (b) Input blurred images. Restored images using (c) [14] (d) [17], (e) [26], (f) [40], and (g) proposed approach.

We perform extensive quantitative evaluation using real kernels returned by BD methods on these three datasets so as to validate the wide scope and applicability of our method.

Table 3.1: Average PSNR in dB on dataset of [19]

NBD method	BD method for kernel estimation			
	[10]	[5]	[19]	Average
[17]	28.86	29.32	29.49	29.23
[14]	29.08	29.39	29.55	29.34
[26]	29.29	29.50	29.66	29.55
[40]	29.51	29.66	29.85	29.67
Ours (1 input)	29.50	29.64	29.91	29.68
Ours (2 inputs)	30.27	30.53	30.76	30.52
Ours (3 inputs)	**30.40**	**30.62**	**30.87**	**30.63**

Towards this end, we obtain kernel estimates for all the images in each dataset using a number of BD methods. For a particular BD method, we use the corresponding set of kernels to obtain the restored images using each NBD method, including ours. The quality of these deblurred images is used to assess performance. For the dataset in [19], we used kernel estimates obtained from [5, 10, 19], whereas for [32] we used kernel estimates returned by [5, 20, 37]. Since [16] comprises of many challenging examples, there exist examples on which BD algorithms ([22,32,37,39]) fail to recover the blur kernels. To avoid the damaging influence of such bad kernel estimates, we excluded them while testing performance in Table 3.2.

To evaluate the performance of our proposed approach, we compared with existing NBD approaches in [14, 17, 26, 40] and used PSNR, SSIM, and IFC [31] as metrics. We used the online-available implementation of these NBD methods with optimal parameters settings as specified in the respective works. For [14], we used $\lambda = 2e3$, as given therein. Use of IFC in the dataset of [16] is motivated by the fact that on the images from this dataset, IFC has the highest correlation with respect to human subject scores [16]. As can be observed from Tables 3.1 and 3.2, our method significantly outperforms the state-of-the-art works

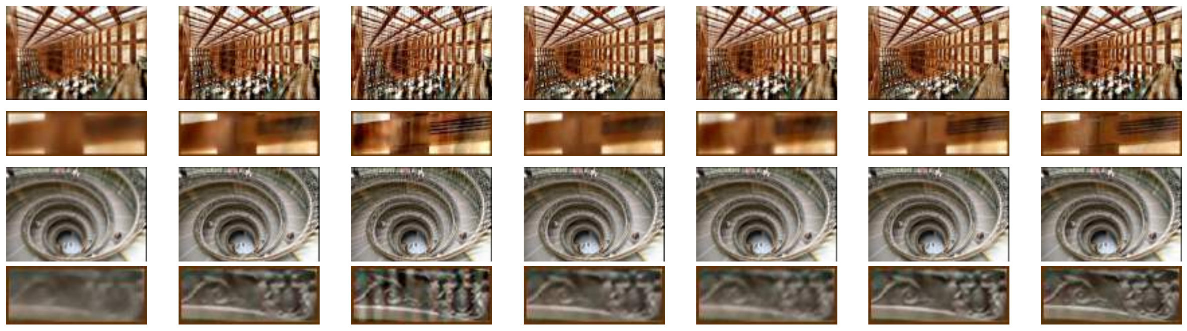

Figure 3.8: Examples from the dataset in [16]. Restored image using [14] with (a) $\lambda = 2e2$, (b) $\lambda = 2e3$, and (c) $\lambda = 2e4$. Results from (d) [17], (e) [26], (f) [40], and (g) proposed approach.

Table 3.2: Performance comparison on dataset of [32] and [16]

NBD method	BD method for kernel estimation						
	PSNR/SSIM for images from [32]			SSIM/IFC for images from [16]			
	[5]	[37]	[20]	[37]	[39]	[32]	[22]
[17]	28.02/0.81	29.56/0.83	28.87/0.80	0.72/2.17	0.72/2.00	0.69/1.92	0.69/1.91
[14]	28.29/0.83	30.15/0.85	29.27/0.82	0.73/2.40	0.73/2.16	0.71/2.08	0.71/2.13
[26]	27.22/0.78	28.34/0.81	27.90/0.78	0.66/1.70	0.67/1.60	0.65/1.60	0.63/1.50
[40]	28.46/0.83	30.51/0.86	29.59/0.82	0.75/2.52	0.74/2.28	0.72/2.20	0.72/2.20
Ours (3 inputs)	**29.54/0.88**	**32.60/0.91**	**30.69/0.85**	**0.78/2.58**	**0.76/2.30**	**0.75/2.31**	**0.75/2.41**

with respect to *all* the metrics. While with a single input (Table 3.1), the performance of our approach is only comparable to other works, the addition of more inputs results in significant improvement in PSNR.

For visual comparisons, we show few randomly chosen representative examples from all three datasets which we used in our quantitative evaluation. Fig. 1.1 illustrates the results of our detail-preserving non-blind deconvolution method while attempting to deblur the image in Fig. 1.1(a) using a noisy kernel estimate. We give the image estimates shown in Figs. 1.1(b-d) as input to our fully convolutional network (FCN) and it yields the high quality restoration result in Fig. 1.1(f). Note that our method clearly reveals significantly higher details as compared to the state-of-the-art [40] (Fig. 1.1(e)). Fig. 3.7 shows a set of low resolution image examples from the dataset in [19]. We have used the kernel estimates shown in Fig. 3.7(a) to restore the images from the corresponding blurry images (Fig. 3.7(b)). As is evident from Fig. 3.7(g), our approach restores the images without artifacts while achieving significant improvement in the recovery of details over competing methods. Fig. 1.1 and Fig. 3.8 reveal performance differences on high resolution images from [16, 32]. While our approach (Fig. 1.1(f) and Fig. 3.8(g)) is able to deliver an artifact-free and yet detail-preserving image by integrating desired features from the inputs (Figs. 1.1(b-d), Figs. 3.8(a-c)), the competing methods (Fig. 1.1(e), Figs. 3.8(d-f)) fail to recover these details.

Since we use [14] to obtain the inputs for our network, the performance difference between our network output and that of [14] can be treated as the performance gain achieved through our scheme. Note that for most of the cases, we could achieve more than 1dB gain in PSNR over the inputs. It should be possible to use image estimates obtained from even other prior-based NBD methods, which opens up the possibility to improve the performance further by employing better image priors.

3.7 Qualitative Comparisons

In this section, we provide visual comparisons of many randomly chosen examples from the datasets in [16, 18, 32]. The results shown here include images of a wide variety of scenes, and kernel estimates from various blind deblurring methods. The qualitative comparison that we present here is mainly to illustrate the wide-scope of our proposed approach in handling different kinds of scene contents, kernel noises, deconvolution artifacts etc.. In our comparisons we follow, four different ways to display the impact of results. We compare the results of proposed approach with all the inputs of our network, to visualize the variations in the kind of image artifacts that we can handle. Visual comparison with respect to blurry image,

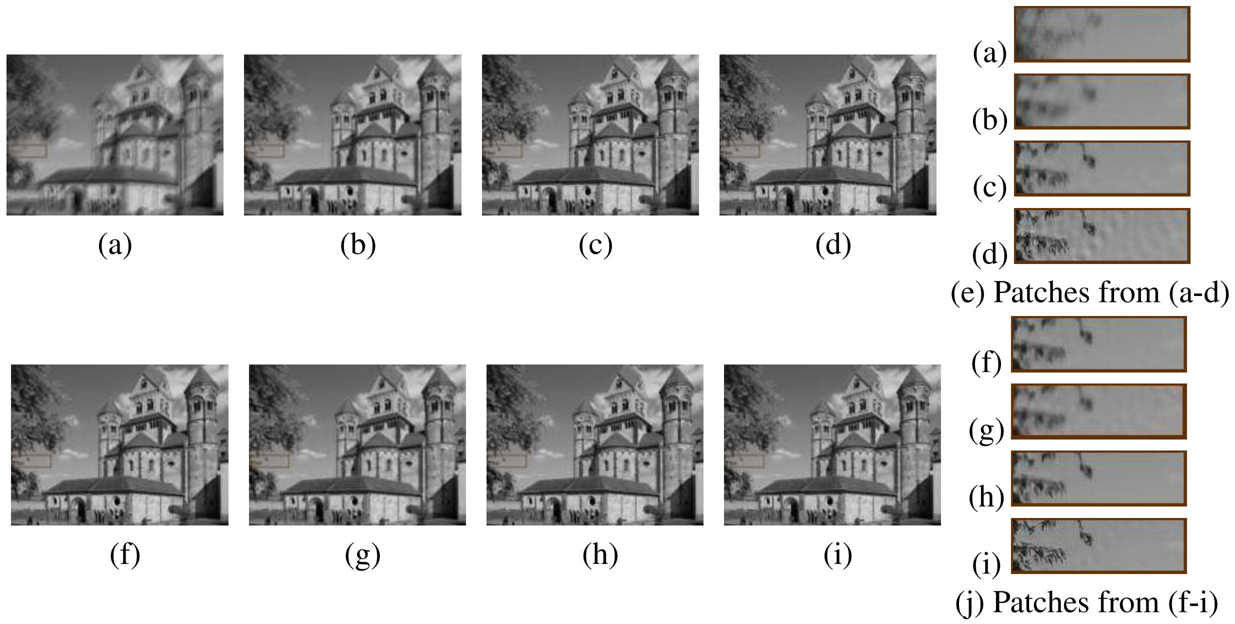

Figure 3.9: Image from [32] dataset, for a noisy kernel estimate returned by [37]. (a) Input blurred image. Restored image using [14] with (b) $\lambda = 2e2$, (c) $\lambda = 2e3$, and (d) $\lambda = 2e4$. Results from (f) [17], (g) [26], (h) [40], and (i) proposed approach.

and kernel estimates are provided to have an understanding of the level of blur, and the kinds of kernel noise that our approach can handle respectively. We also provide comparison with respect to ground truth image, which illustrates the closeness of the results (to ground truth) in preserving details. In all the examples, the detail preservation that can be achieved through our proposed approach as compared to other non-blind deblurring methods is clearly evident. We also provide few challenging scenarios (Figs. 3.12, 3.13) for which the estimated kernels are significantly noise-affected. For the kernel estimates with significant noises, although we are unable to remove the artifacts completely, the results of our approach are significantly better than those of competing methods.

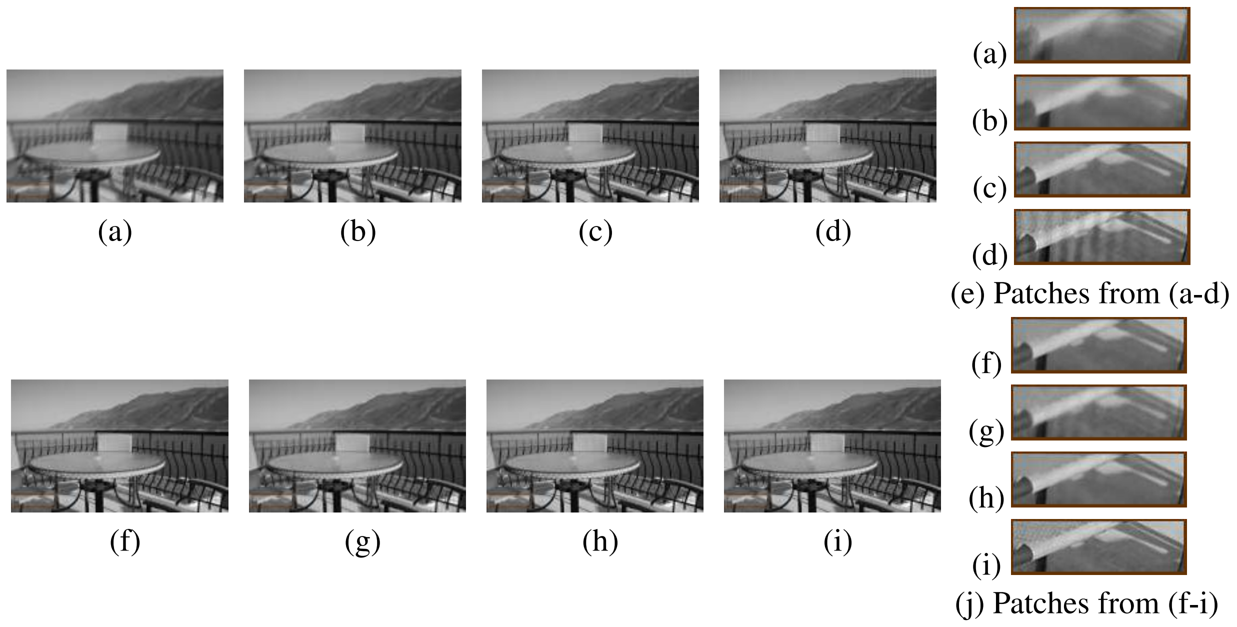

Figure 3.10: Image from [32] dataset, for a noisy kernel estimate returned by [37]. (a) Input blurred image. Restored image using [14] with (b) $\lambda = 2e2$, (c) $\lambda = 2e3$, and (d) $\lambda = 2e4$. Results from (f) [17], (g) [26], (h) [40], and (i) proposed approach.

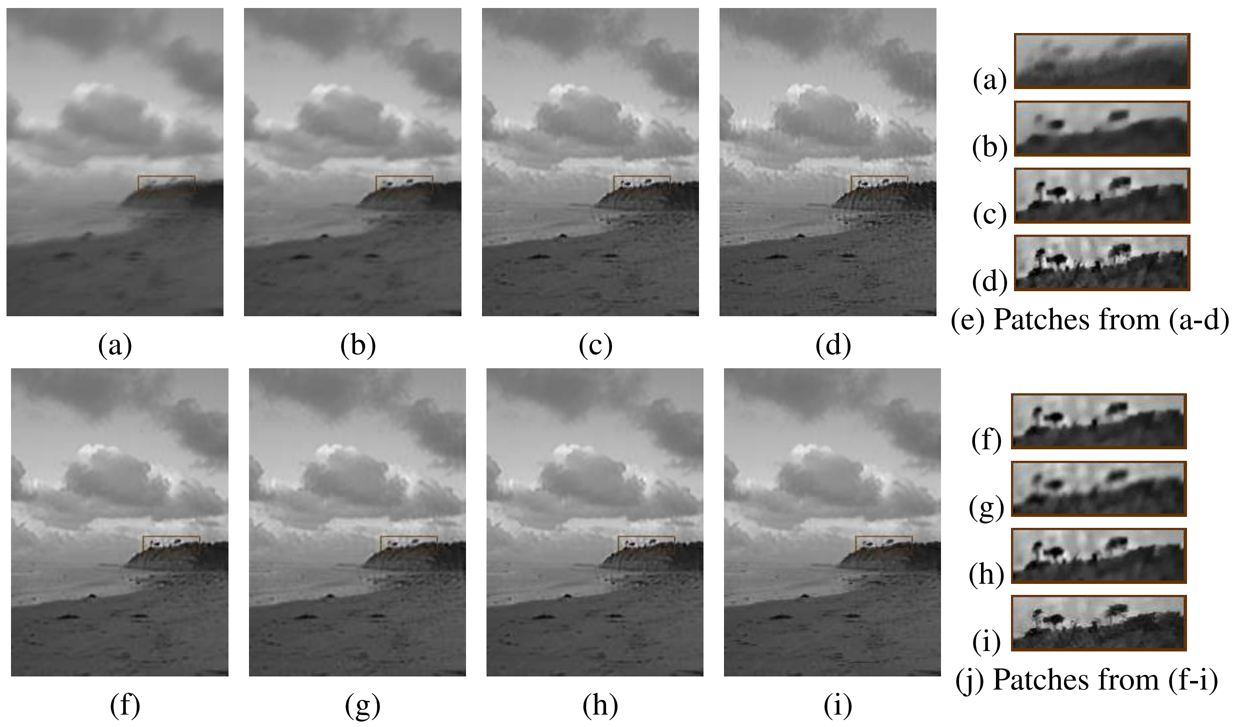

Figure 3.11: Image from [32] dataset, for a noisy kernel estimate returned by [37]. (a) Input blurred image. Restored image using [14] with (b) $\lambda = 2e2$, (c) $\lambda = 2e3$, and (d) $\lambda = 2e4$. Results from (f) [17], (g) [26], (h) [40], and (i) proposed approach.

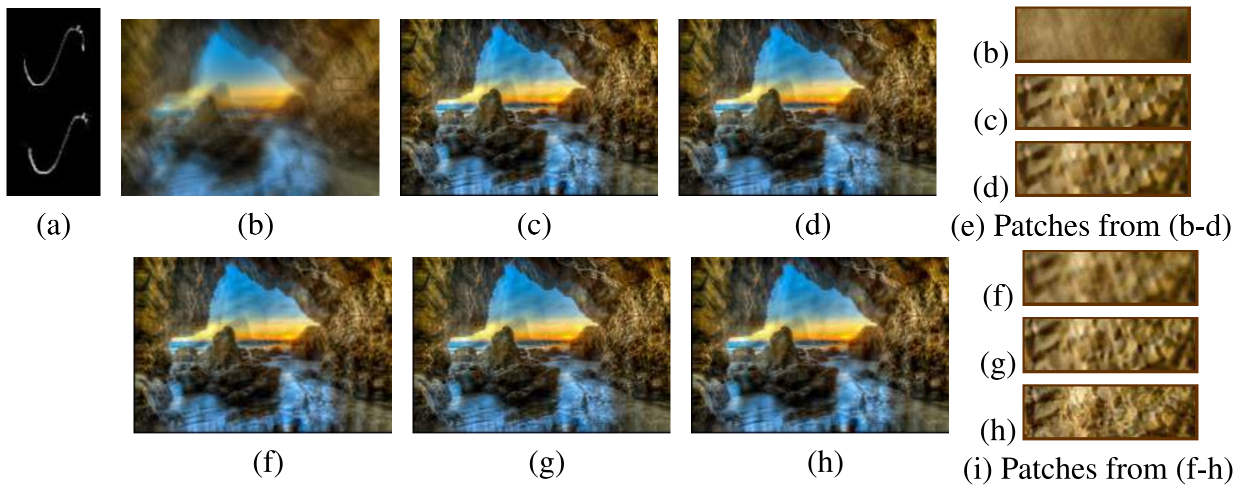

Figure 3.12: Worst case scenario example 1: Image from [16] dataset, for a severely noisy kernel estimate returned by [32]. (a) Ground truth (top) and estimated kernels (bottom). (b) Input blurred image. Restored images using (c) [14] (d) [17], (f) [26], (g) [40], and (h) proposed approach.

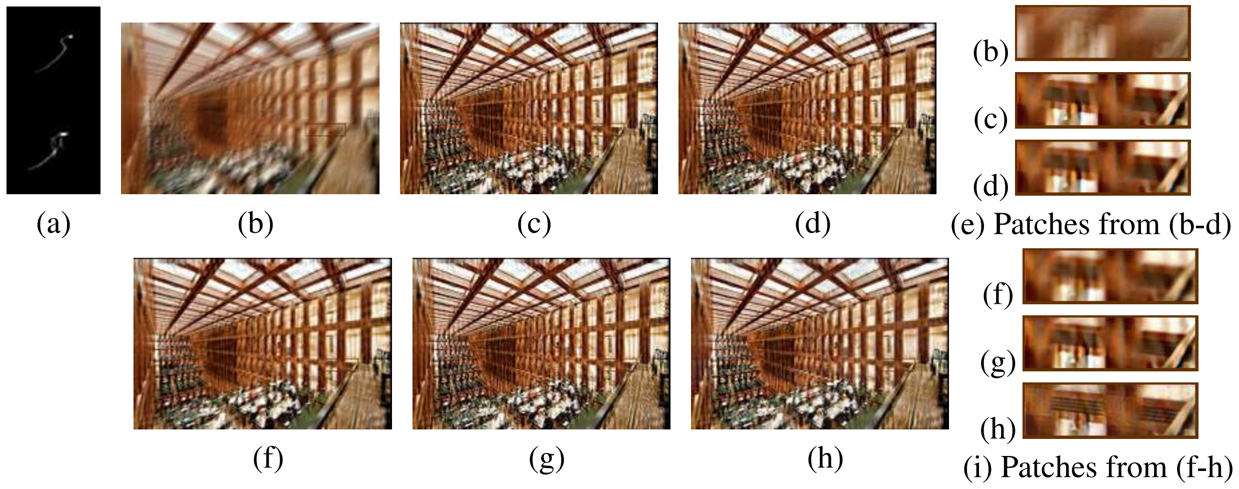

Figure 3.13: Worst case scenario example 2: Image from [16] dataset, for a severely noisy kernel estimate returned by [37]. (a) Ground truth (top) and estimated kernels (bottom). (b) Input blurred image. Restored images using (c) [14] (d) [17], (f) [26], (g) [40], and (h) proposed approach.

Chapter 4

Conclusion

We presented a deep CNN-based framework for non-blind restoration of motion blurred images. Unlike existing works, we investigated a very relevant scenario which is the unavailability of the exact ground truth kernel. By using multiple latent image estimates obtained with different prior strengths as inputs, our network exploits the complementarity present in the input data to yield high-quality restoration results. To remove kernel noise-specific artifacts in the deconvolved results, we trained our network with real kernels obtained from existing blind deblurring methods as well as synthetically generated noisy kernels. We also provided results of our network on high resolution images which are artifact-free and yet detail-preserving image by integrating desired features of the inputs, performing better than existing methods. Our method is able to deliver state-of-the-art performance in non-blind deblurring.

We trained our network with training data assuming images are noise free and kernels generated are noisy, but we didn't explicitly consider the case of noise present in the blurred images. We believe that our network can be able to learn this task and perform well if necessary training data is provided. As future work one can train the network with data from noisy images and noisy-kernel combined for a more general case.

Bibliography

[1] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *TPAMI*, 33(5):898–916, 2011. 30

[2] H. C. Burger, C. J. Schuler, and S. Harmeling. Image denoising: Can plain neural networks compete with bm3d? In *CVPR*, pages 2392–2399. IEEE, 2012. 14, 24

[3] B. Cai, X. Xu, K. Jia, C. Qing, and D. Tao. Dehazenet: An end-to-end system for single image haze removal. *TIP*, 25(11):5187–5198, 2016. 14

[4] A. Chakrabarti. A neural approach to blind motion deblurring. In *ECCV*, pages 221–235. Springer, 2016. 14, 30

[5] S. Cho and S. Lee. Fast motion deblurring. In *TOG*, volume 28, page 145. ACM, 2009. 10, 34, 35

[6] S. Cho, J. Wang, and S. Lee. Handling outliers in non-blind image deconvolution. In *ICCV*, pages 495–502. IEEE, 2011. 10, 13

[7] C. Dong, C. C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. In *ECCV*, pages 184–199. Springer, 2014. 14, 25

[8] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *TPAMI*, 38(2):295–307, 2016. 12, 14

[9] D. Eigen, D. Krishnan, and R. Fergus. Restoring an image taken through a window covered with dirt or rain. In *ICCV*, pages 633–640, 2013. 12, 14, 24

[10] R. Fergus, B. Singh, A. Hertzmann, S. T. Roweis, and W. T. Freeman. Removing camera shake from a single photograph. In *TOG*, volume 25, pages 787–794. ACM, 2006. 10, 34

[11] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 26

[12] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *arxiv*, 2016. 26, 30

[13] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 30

[14] D. Krishnan and R. Fergus. Fast image deconvolution using hyper-laplacian priors. In *NIPS*, pages 1033–1041, 2009. 7, 8, 10, 13, 22, 23, 26, 30, 31, 33, 34, 35, 36, 37, 38, 39

[15] D. Krishnan, T. Tay, and R. Fergus. Blind deconvolution using a normalized sparsity measure. In *CVPR*, pages 233–240. IEEE, 2011. 10, 30

[16] W.-S. Lai, J.-B. Huang, Z. Hu, N. Ahuja, and M.-H. Yang. A comparative study for single image blind deblurring. In *CVPR*, June 2016. 7, 8, 9, 10, 11, 33, 34, 35, 36, 39

[17] A. Levin, R. Fergus, F. Durand, and W. T. Freeman. Image and depth from a conventional camera with a coded aperture. *TOG*, 26(3):70, 2007. 7, 8, 13, 33, 34, 35, 37, 38, 39

[18] A. Levin, Y. Weiss, F. Durand, and W. T. Freeman. Understanding and evaluating blind deconvolution algorithms. In *CVPR*, pages 1964–1971. IEEE, 2009. 10, 36

[19] A. Levin, Y. Weiss, F. Durand, and W. T. Freeman. Efficient marginal likelihood optimization in blind deconvolution. In *CVPR*, pages 2657–2664. IEEE, 2011. 7, 9, 30, 33, 34, 36

[20] T. Michaeli and M. Irani. Blind deblurring using internal patch recurrence. In *ECCV*, pages 783–798. Springer, 2014. 34, 35

[21] J. Pan, D. Sun, H. Pfister, and M.-H. Yang. Blind image deblurring using dark channel prior. In *CVPR*, June 2016. 10

[22] D. Perrone and P. Favaro. Total variation blind deconvolution: The devil is in the details. In *CVPR*, pages 2909–2916, 2014. 34, 35

[23] J. S. Ren, L. Xu, Q. Yan, and W. Sun. Shepard convolutional neural networks. In *NIPS*, pages 901–909, 2015. 14

[24] W. H. Richardson. Bayesian-based iterative method of image restoration. *JOSA*, 62(1):55–59, 1972. 13

[25] U. Schmidt, J. Jancsary, S. Nowozin, S. Roth, and C. Rother. Cascades of regression tree fields for image restoration. *TPAMI*, 38(4):677–689, 2016. 14

[26] U. Schmidt and S. Roth. Shrinkage fields for effective image restoration. In *CVPR*, pages 2774–2781, 2014. 7, 8, 10, 14, 33, 34, 35, 37, 38, 39

[27] U. Schmidt, C. Rother, S. Nowozin, J. Jancsary, and S. Roth. Discriminative non-blind deblurring. In *CVPR*, pages 604–611, 2013. 10, 14

[28] U. Schmidt, K. Schelten, and S. Roth. Bayesian deblurring with integrated noise estimation. In *CVPR*, pages 2625–2632. IEEE, 2011. 14

[29] C. J. Schuler, H. Christopher Burger, S. Harmeling, and B. Scholkopf. A machine learning approach for non-blind image deconvolution. In *CVPR*, pages 1067–1074, 2013. 14, 24

[30] C. J. Schuler, M. Hirsch, S. Harmeling, and B. Schölkopf. Learning to deblur. *TPAMI*, 38(7):1439–1451, 2016. 14

[31] H. R. Sheikh, A. C. Bovik, and G. De Veciana. An information fidelity criterion for image quality assessment using natural scene statistics. *TIP*, 14(12):2117–2128, 2005. 34

[32] L. Sun, S. Cho, J. Wang, and J. Hays. Edge-based blur kernel estimation using patch priors. In *ICCP*, pages 1–8. IEEE, 2013. 6, 7, 8, 9, 10, 24, 30, 33, 34, 35, 36, 37, 38, 39

[33] Y. Wang, J. Yang, W. Yin, and Y. Zhang. A new alternating minimization algorithm for total variation image reconstruction. *SIAM Journal on Imaging Sciences*, 1(3):248–272, 2008. 13

[34] O. Whyte, J. Sivic, and A. Zisserman. Deblurring shaken and partially saturated images. *IJCV*, 110(2):185–201, 2014. 10, 13, 14

[35] N. Wiener. *Extrapolation, interpolation, and smoothing of stationary time series*, volume 7. MIT press Cambridge, MA, 1949. 13

[36] J. Xie, L. Xu, and E. Chen. Image denoising and inpainting with deep neural networks. In *NIPS*, pages 341–349, 2012. 14, 24

[37] L. Xu and J. Jia. Two-phase kernel estimation for robust motion deblurring. In *ECCV*, pages 157–170. Springer, 2010. 7, 8, 10, 34, 35, 37, 38, 39

[38] L. Xu, J. S. Ren, C. Liu, and J. Jia. Deep convolutional neural network for image deconvolution. In *NIPS*, pages 1790–1798, 2014. 10, 12, 14, 24

[39] L. Xu, S. Zheng, and J. Jia. Unnatural l0 sparse representation for natural image deblurring. In *CVPR*, pages 1107–1114, 2013. 10, 34, 35

[40] D. Zoran and Y. Weiss. From learning models of natural image patches to whole image restoration. In *ICCV*, pages 479–486. IEEE, 2011. 6, 7, 8, 10, 11, 13, 22, 33, 34, 35, 36, 37, 38, 39