Blur-Invariant Deep Learning for Blind-Deblurring

T M Nimisha ee13d037@ee.iitm.ac.in
Akash Kumar Singh ee12b072@ee.iitm.ac.in
A N Rajagopalan raju@ee.iitm.ac.in
Indian Institute of Technology, Madras, India

Abstract

In this paper, we investigate deep neural networks for blind motion deblurring. Instead of regressing for the motion blur kernel and performing non-blind deblurring outside of the network (as most methods do), we propose a compact and elegant end-to-end deblurring network. Inspired by the data-driven sparse-coding approaches that are capable of capturing linear dependencies in data, we generalize this notion by embedding non-linearities into the learning process. We propose a new architecture for blind motion deblurring that consists of an autoencoder that learns the data prior, and an adversarial network that attempts to generate and discriminate between clean and blurred features. Once the network is trained, the generator learns a blur-invariant data representation which when fed through the decoder results in the final deblurred output.

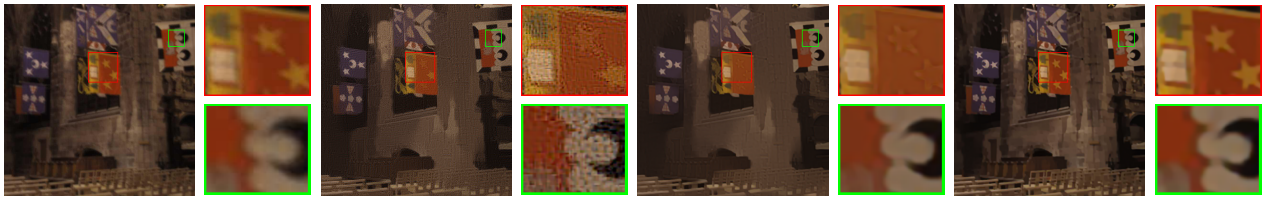
1. Introduction

Motion blur is an inevitable phenomenon under long exposure times. With mobile cameras becoming ubiquitous, there is an increasing need to invert the blurring process to recover a clean image. However, it is well-known that the problem of blur inversion is quite ill-posed. Many methods exist [40] that rely on information from multiple frames captured using video or burst mode and work by harnessing the information from these frames to solve for the underlying original (latent) image. Single image blind-deblurring is considerably more challenging as the blur kernel as well as the latent image must be estimated from just one observation. It is this problem that we attempt to solve here.

Early works [5, 27, 19] assumed space-invariant blur and iteratively solved for the latent image and blur kernel. Although these convolutional models are simple and straight forward to analyze using FFTs, they fail to account for space-variant blur caused by non-linear camera motion or dynamic objects or depth-varying scenes. Nevertheless, even in such situations local patch-wise convolutional model can be employed to achieve deblurring. Instead of using a patch-wise model, works such as [32, 15] take the space-variant blur formation model itself into consideration. But the deblurring process becomes highly ill-posed as it must now estimate blur kernel at each pixel position along with the underlying image intensities. For planar scenes or under pure camera rotations, the methods in [32, 10] circumvent this issue by modeling the global camera motion using homographies.

Major efforts have also gone into designing priors that are apt for the underlying clean image and the blur kernel to regularize the inversion process and ensure convergence during optimization. The most widely used priors are total variational regularizer [4, 25], sparsity prior on image gradients, l_1/l_2 image regularization [17], the unnatural l_0 prior [37] and the very recent dark channel prior [23] for images. Even though such prior-based optimization schemes have shown promise, the extent to which a prior is able to perform under general conditions is questionable [17]. Some priors (such as the sparsity prior on image gradient) even tend to favor blurry results [19]. In a majority of situations, the final result requires a judicious selection of the prior, its weightage, as well as tuning of other parameters. Depending on the amount of blur, these values need to be adjusted so as to strike the right balance between over-smoothing and ringing in the final result. Such an effect is depicted in Fig. 1. Note that the results fluctuate with the weightage selected for the prior. These results correspond to the method of [23] with varying weights for dark channel prior (λ), l_0 prior (μ) and the TV prior (λ_{TV}). Furthermore, these methods are iterative and quite time-consuming.

Dictionary learning is a data-driven approach and has shown good success for image restoration tasks such as denoising, super-resolution and deblurring [1, 39, 38]. Research has shown that sparsity helps to capture higher-order correlations in data, and sparse codes are well-suited for natural images [20]. Lou et al. [38] have proposed a dictionary replacement technique for deblurring of images blurred with a Gaussian kernel of specific variance. The authors of [33] adopt this concept to learn a pair of dictionaries jointly from blurred as well as clean image patches with the constraint that the sparse code be invariant to blur. They were able to show results for space-invariant motion deblur-



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

Figure 1: Effect of image prior. (a) Input blurred image. (b-d) Deblurred result using [23] with ($\lambda = 0.04$, $\mu = 0.007$, $\lambda_{TV} = 0.001$), ($\lambda = 0.001$, $\mu = 0.01$, $\lambda_{TV} = 0.01$) and ($\lambda = 0.004$, $\mu = 0.005$, $\lambda_{TV} = 0.01$), respectively.

ring but were again constrained to a single kernel. For multiple kernels, they learn different dictionaries and choose the one for which the reconstruction error is the least. Even though sparse coding models perform well in practice, they share a *shallow* linear structure and hence are limited in their ability to generalize to different types of blurs.

Recently, deep learning and generative networks have made forays into computer vision and image processing, and their influence and impact are growing rapidly by the day. Neural networks gained in popularity with the introduction of Alexnet [18] that showed a huge reduction in classification error compared to traditional methods. Following this, many regression networks based on Convolutional Neural Networks (CNNs) were proposed for image restoration tasks. With increasing computational speeds provided by GPUs, researchers are investigating deep networks for the problem of blur inversion as well. Xu et al. [36] proposed a deep deconvolutional network for non-blind single image deblurring (i.e, the kernel is fixed and known apriori). Schuler et al. [26] came up with a neural architecture that mimics traditional iterative deblurring approaches. Chakrabarti [3] trained a patch-based neural network to estimate the kernel at each patch and employed a traditional non-blind deblurring method in the final step to arrive at the deblurred result. Since these methods estimate a single kernel for the entire image, they work for the space-invariant case alone. The most relevant work to handle space-variant blur is a method based on CNN for patch-level classification of the blur type [28], which focuses on estimating the blur kernel at all locations from a single observation. They parametrize the kernels (using length and angle) and estimate these parameters at each patch using a trained network. However, such a parametric model is too restrictive to handle general camera motion blur.

The above-mentioned methods attempt to estimate the blur kernel using a deep network but finally perform non-blind deblurring exterior to the network to get the deblurred result. Any error in the kernel estimate (due to poor edge content, saturation or noise in the image) will impact deblurring quality. Moreover, the final non-blind deblurring step typically assumes a prior (such as sparsity on the gradient of latent image) which again necessitates a judicious selection of prior weightage; else the deblurred result will be imperfect as already discussed (Fig. 1). Hence, kernel-free approaches are very much desirable.

In this work, we propose a deep network that can perform single image blind-deblurring without the cumbersome need for prior modeling and regularization. The core idea is to arrive at a blur-invariant representation learned using deep networks that facilitates end-to-end deblurring. Performance-wise, our method is at par with conventional methods which use regularized optimization, and outperforms deep network-based methods. While conventional methods can only handle specific types of space-variant blur such as blur due to camera motion or object motion or scene with depth variations, our network does not suffer from these limitations. Most importantly, the run-time for our method is very small compared to conventional methods. The key strength of our network is that it does end-to-end deblurring with performance quality at par or better than competing methods while being computationally efficient.

2. Blur-invariant Feature Learning

It is well-known that most sensory data, including natural images, can be described as a superposition of small number of atoms such as edges and surfaces [20]. Dictionary-based methods exploit this information and learn the atoms that can represent data in sparse forms for various image restoration tasks (including deblurring). With an added condition that these representations should be invariant to the blur content in the image, dictionary methods have performed deblurring by learning coupled dictionaries [33]. However, constrained by the fact that dictionaries can capture only linearities in the data and blurring process involves non-linearities (high frequencies are suppressed more), their deblurring performance does not generalize across blurs.

In this paper, we extend the notion of blur-invariant representation to deep networks that can capture non-linearities in the data. We are not the first one to approach deep learning as a generalization of dictionary learning for sparse coding. The work in [34] combines sparse coding and denois-

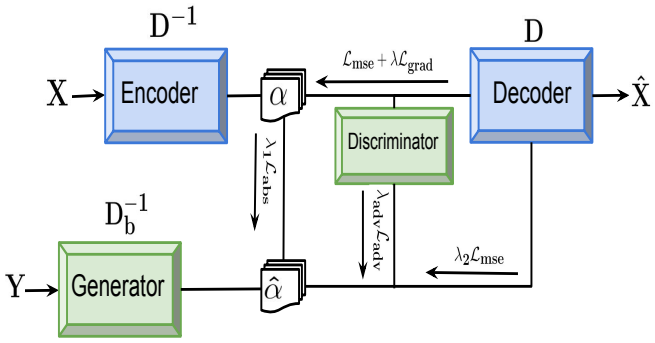


Figure 2: Illustration of our architecture.

ing encoders for the task of denoising and inpainiting. Deep neural networks, in general, have yielded good improvements over conventional methods for various low-level image restoration problems including super-resolution [7], inpainting and denoising [24, 34]. These networks are learned end-to-end by exposing them to lots of example-data from which the network learns the mapping to undo distortions. We investigate the possibility of such a deep network for the task of single image motion deblurring.

For blind-deblurring, we first require a good feature representation that can capture image-domain information. Autoencoders have shown great success in unsupervised learning by encoding data to a compact form [12] which can be used for classification tasks. This motivated us to train an autoencoder on clean image patches for learning the feature representation. Once a good representation is learned for clean patches, the next step is to produce a blur-invariant representation (as in [33]) from blurred data. We propose to use a generative adversarial network (GAN) for this purpose which involves training of a generator and discriminator that attempt to compete with each other. The purpose of the generator is to confuse the discriminator by producing clean features from blurred data that are similar to the ones produced by the autoencoder so as to achieve blur-invariance. The discriminator, on the other hand, tries to beat the generator by identifying the clean and blurred features.

A schematic of our proposed architecture is shown in Fig. 2. Akin to dictionary learning that represents any data X as a sparse linear combination of dictionary atoms D i.e, $X = D\alpha$, our encoder-decoder module performs this in non-linear space. Hence, the encoder can be thought of as an inverse dictionary D^{-1} that projects the incoming data into a sparse representation. The decoder acts as the dictionary D that reconstructs the input from the sparse representation. Generator training can be treated as learning the blur dictionary that can project the blurred data Y into the same sparse representation of X i.e, $\alpha = D^{-1}X = D_b^{-1}Y$. Once training is done, the input blurry image (Y) is passed through the generator to get a blur-invariant feature which when projected to the decoder yields the deblurred result as $\hat{X} = D\alpha = DD_b^{-1}Y$.

Thus, by associating the feature representation learned by the autoencoder with GAN training, our model is able to perform single image blind deblurring in an end-to-end manner. Ours is a kernel-free approach and does away with the tedious task of selecting priors, a serious bottleneck of conventional methods. Unlike other deep learning methods, our network directly regresses for the clean image.

The main contributions of our work are as follows :

- We propose a compact end-to-end regression network that directly estimates the clean image from a single blurred frame without the need for optimal prior selection and weighting, as well as blur kernel estimation.
- The proposed architecture is new and consists of an autoencoder in conjunction with a generative network for producing blur-invariant features to guide the deblurring process.
- Our method is computationally efficient and can restore both space-invariant and space-variant blur due to camera motion.
- The network is even able to account for blur caused by object motion/depth changes (to an extent) although it was not trained explicitly for such a situation.

3. Network Architecture

Our network consists of an autoencoder that learns the clean image domain and a generative adversarial network that generates blur-invariant features. We train our network in two stages. We first train an autoencoder to learn the clean image manifold. This is followed by the training of a generator that can produce clean features from a blurred image which when fed to the decoder gives the deblurred output. Note that instead of combining the task of data-representation and deblurring into a single network, we relegate the task of data-learning to the autoencoder and use this information to guide image deblurring. Details of the architecture and the training procedure are explained next.

3.1. Encoder-Decoder

Autoencoders were originally proposed for the purpose of unsupervised learning [12] and have since been extended to a variety of applications. An autoencoder projects the input data into a low-dimensional space and recovers the input from this representation. When not modeled properly, it is likely that the autoencoder learns to just compress the data without learning any useful representation. Denoising encoders [30] were proposed to overcome this issue by corrupting the data with noise and letting the network undo this effect and get back a clean output. This ensures that the autoencoder learns to correctly represent clean data. Deepak

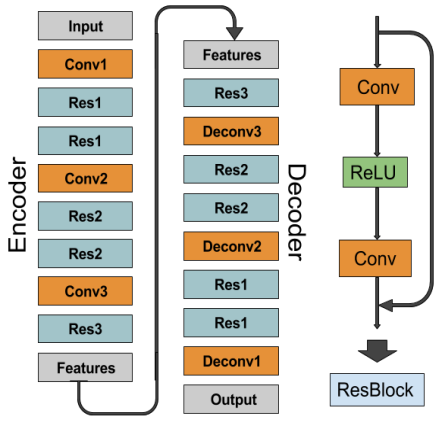


Figure 3: Autoencoder architecture with residual networks.

et al. [24] extended this idea from mere data representation to context representation for the task of inpainiting. In effect, it learns a meaningful representation that can capture domain information of data.

We investigated different architectures for the autoencoder and observed that including residual blocks (ResNet) [11] helped in achieving faster convergence and in improving the reconstructed output. Residual blocks help by bypassing the higher-level features to the output while avoiding the gradient vanishing problem. The training data was corrupted with noise (30% of the time) to ensure encoder reliability and to avoid learning an identity map. The architecture used in our paper along with the ResNet block is shown in Fig. 3. A detailed description of the filter and feature map sizes along with the stride values used are as given below.

Encoder: $C^{5}_{3\to 8} \downarrow 2 \to R^{5(2)}_{8} \to C^{5}_{8\to 16} \downarrow 2 \to R^{5(2)}_{16} \to C^{3}_{16\to 32} \downarrow 2 \to R^{3}_{32}$

Decoder: $R^{3}_{32} \to C^{2}_{32\to 16} \uparrow 2 \to R^{5(2)}_{16} \to C^{4}_{16\to 8} \uparrow 2 \to R^{5(2)}_{8} \to C^{4}_{8\to 3} \uparrow 2$

where $C^{c}_{a\to b} \downarrow d$ represents convolution mapping from a feature dimension of a to b with a stride of d and filter size of c, $\downarrow$ represents down-convolution, $\uparrow$ stands for up-convolution, and $R^{b(c)}_{a}$ represents the residual block which consists of a convolution and a ReLU block with output feature size a, filter size b, and c represents the number of repetitions of residual blocks.

Fig. 4 shows the advantage of the ResNet block. Fig. 4(a) is the target image and Figs. 4(c) and (d) are the output of autoencoders with and without ResNet block for the same number of iterations for the input noisy image in Fig. 4(b). Note that the one with ResNet converges faster and preserves the edges due to skip connections that pass on the information to deeper layers.

(a) (b) 26.1 dB (c) 29.5 dB (d) 23.1 dB

Figure 4: Effect of ResNet on reconstruction. (a) The target image. (b) Noisy input to the encoder-decoder module. (c) Result of encoder-decoder module of Fig. 3. (d) Result obtained by removing ResNet for the same number of iterations. PSNR values are given under the respective figures. (*Enlarge for better viewing*).

3.2. GAN for feature mapping

The second stage of training constitutes learning a generator that can map from blurred image to clean features. For this purpose, we used a generative adversarial network. GANs were first introduced by Goodfellow [9] in 2014. Since then, they have been widely used for various image related tasks. GANs consists of two models: a Generator ($\mathcal{G}$) and a Discriminator ($\mathcal{D}$) which play a two-player mini-max game. $\mathcal{D}$ tries to discriminate between the samples generated by $\mathcal{G}$ and training data samples, while $\mathcal{G}$ tries to fool the discriminator by generating samples close to the actual data distribution. The mini-max cost function [9] for training GANs is given by

$$\min_{\mathcal{G}} \max_{\mathcal{D}} C(\mathcal{G},\mathcal{D}) = E_{x\sim P_{data}(x)}[\log \mathcal{D}(x)] + E_{z\sim P_z(z)}[\log(1-\mathcal{D}(\mathcal{G}(z)))]$$

where $\mathcal{D}(x)$ is the probability assigned by the discriminator to the input x for discriminating x as a real sample. P_{data} and P_z are the respective probability distributions of data x and the input random vector z. The main goal of [9] is to generate a class of natural images from z.

GANs that just accept random noise and attempt to model the probability distribution of data over noise are difficult to train. Sometimes their instability leads to artifacts in the generated image. Hence, instead of a vanila network for GAN, we used conditional GAN which was introduced by Mirza et al. [22] and which enables GANs to accomodate extra information in the form of a conditional input. The inclusion of adversarial cost in the loss function has shown great promise [24], [14]. Training conditional GANs is a lot more stable than unconditional GANs due to the additional guiding input. The modified cost function [14] is given by

$$\min_{\mathcal{G}} \max_{\mathcal{D}} C_{cond}(\mathcal{G},\mathcal{D}) = E_{x,y\sim P_{data}(x,y)}[\log \mathcal{D}(x,y)] + E_{x\sim P_{data}(x),z\sim P_z(z)}[\log(1-\mathcal{D}(x,\mathcal{G}(x,z)))] \quad (1)$$

where y is the clean target feature, x is the conditional image (the blurred input), and z is the input random vector.

(a) (b) (c)

Figure 5: Effect of direct regression using generative networks. (a) Input blurred image. (b-c) Output of the network and the expected output.

In conditional GANs, the generator tries to model the distribution of data over the joint probability distribution of x and z. When trained without z for our task, the network learns a mapping for x to a deterministic output y which is the corresponding clean feature.

[14] proposes an end-to-end network using a generative model to perform image-to-image translation that can be used in multiple tasks. Following this recent trend, we initially attempted regressing directly to the clear pixels using off-the-shelf generative networks. However, we observed that this can lead to erroneous results as shown in Fig. 5. The main reason for this could be that the network becomes unstable when trained on higher-dimensional data. Also GANs are quite challenging to train and have mainly shown results for specific class of images. When trained for large diverse datasets, training does not converge [31]. Hence, we used the apriori-learned features of the autoencoder for training GAN.

Training a perfect discriminator requires it's weights to be updated simultaneously along with the generator such that it is able to discriminate between the generated samples and data samples. This task becomes easy and viable for the discriminator in the feature space for two reasons:
i) In this space, the distance between blurred features and clean features is higher as compared to the image space. This helps in faster training in the initial stage.
ii) The dimensionality of the feature-space is much lower as compared to that of image-space. GANs are known to be quite effective in matching distributions in lower-dimensional spaces [6].

We train GAN using normal procedure but instead of asking the discriminator to discern between generated images and clean images, we ask it to discriminate between their corresponding features. The generator and the discriminator architectures are as given below.

Generator: $C^5_{3\to8} \downarrow 2 \to R^{5(2)}_8 \to C^5_{8\to16} \downarrow 2 \to R^{5(2)}_{16} \to C^3_{16\to32} \downarrow 2 \to R^{5(2)}_{32} \to C^3_{32\to128} \downarrow 2 \to R^{3(2)}_{128} \to C^3_{128\to32} \uparrow 2$

Discriminator: $C^5_{32\to32} \to C^5_{32\to32} \downarrow 2 \to C^5_{32\to16} \to C^5_{16\to16} \downarrow 2 \to C^5_{16\to8} \to C^3_{8\to8} \downarrow 2 \to C^3_{8\to1}$

Each convolution is followed by a Leaky ReLU and batch-normalization in the discriminator, and ReLU and batch-normalization in the generator.

Once the second stage is trained, we have a generator module to which we pass the blurred input during the test phase. The generator produces features which correspond to clean image features which when passed through the decoder deliver the final deblurred result. It may be noted that our network is compact with 34 convolutional layers (generator and decoder put together) despite performing end-to-end deblurring.

3.3. Loss function

We trained our network using the following loss functions. For autoencoder training, we used $\mathcal{L}_{\text{mse}} + \lambda\mathcal{L}_{\text{grad}}$. Adding the gradient-loss helps in preserving edges and recovering sharp images as compared to $\mathcal{L}_{\text{mse}}$ alone. We use normalized l_2 distance on the expected and observed image as our loss function i.e.

$$\mathcal{L}_{\text{mse}} = \|\mathcal{D}e(\mathcal{E}(I + \mathcal{N})) - I\|_2^2 \tag{2}$$

where $\mathcal{D}e$ is the decoder, $\mathcal{E}$ the encoder, N is noise and I is the target (clean) image. The MSE error captures overall image content but tends to prefer a blurry solution. Hence, training only with MSE loss results in loss of edge details. To overcome this, we used gradient loss as it favours edges as discussed in [21] for video-prediction.

$$\mathcal{L}_{\text{grad}} = \|\nabla\mathcal{D}e(\mathcal{E}(I + \mathcal{N})) - \nabla I\|_2^2 \tag{3}$$

where ∇ is the gradient operator.

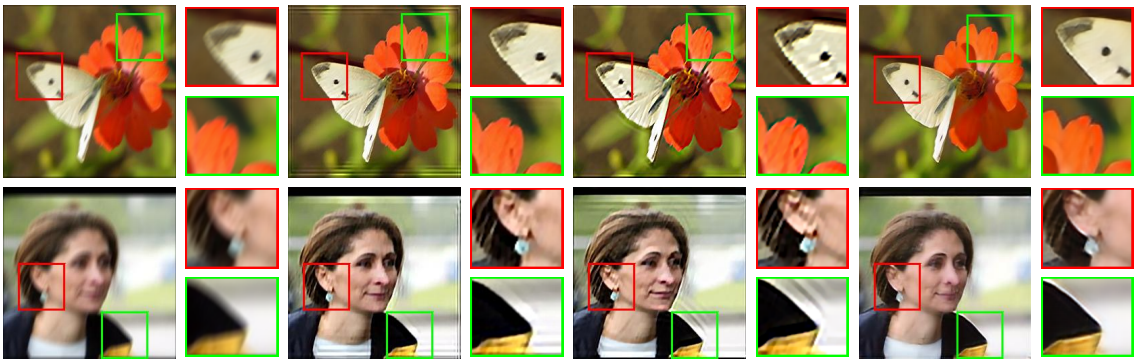
GAN is trained with the combined cost given by $\lambda_{\text{adv}}\mathcal{L}_{\text{adv}} + \lambda_1\mathcal{L}_{\text{abs}} + \lambda_2\mathcal{L}_{\text{mse}}$ in the image and feature space. Even though l_2 loss is simple and easy to back-propagate, it under performs on sparse data. Hence, we used l_1 loss for feature back-propagation i.e.

$$\mathcal{L}_{\text{abs}} = \|\mathcal{G}(B) - \mathcal{E}(I)\|_1 \tag{4}$$

where B is the blurred image. The adversarial loss function $\mathcal{L}_{\text{adv}}$ (given in Eq. (1)) requires that the samples output by the generator should be indistinguishable to the discriminator. This is a strong condition and forces the generator to produce samples that are close to the underlying data distribution. As a result, the generator outputs features that are close to the clean feature samples. Another advantage of this loss is that it helps in faster training (especially during the initial stages) as it provides strong gradients. Apart from adversarial and l_1 cost on the feature space, we also used MSE cost on the recovered clean image after passing the generated features through the decoder. This helps in fine-tuning the generator to match with the decoder. Fig. 2 shows the direction of error back-propagation along with the network modules.

Dataset [29]	Xu & Jia [35]	Xu [37]	Pan [23]	Whyte et al. [32]	Ours
PSNR	28.21	28.11	31.16	26.335	30.54
MSSIM	0.9226	0.9177	0.9623	0.8528	0.9553

Table 1: Average quantitative performance on the dataset [29].



(a) Input (b) [35] (c) [23] (d) Ours

Figure 6: Comparisons for space-invariant deblurring. (a) Input blurred image. (b-c) Deblurred output using methods in [35] and [23], respectively. (d) Our result.

Method	Run time
Ours (Torch, GPU/CPU)	3.4 sec/2 min
Xu & Jia [35] (Executable)	3 min (800 $\times$ 600 pixels)
Xu [37] (Matlab, CPU)	34 sec
Pan [23] (Matlab, CPU)	40 min
Whyte [32] (Matlab, CPU)	4 min

Table 2: Run-time for each method for average image size of 1024 $\times$ 700.

3.4. Training

We trained our autoencoder using clean patches of size 128 $\times$ 128 from the Pascal VOC 2012 dataset [8]. The inputs were randomly corrupted with Gaussian noise (standard deviation = 0.2) 30% of the time to ensure learning of useful data representation. For learning, we used Adam [16] with an initial learning rate of 0.0002 and momentum 0.9 with batch-size of 8. The training took around 10^5 iterations to converge. The gradient cost was scaled by $\lambda = 0.1$ to ensure that the final results are not over-sharpened.

The second stage of training involved learning a blur-invariant representation from blurred data. For creating the blurred data, we used around 10^5 kernels generated using the code provided by Chakrabarthi et al. [3]. The input images from PASCAL dataset were blurred using these kernels and patches of size 128 $\times$ 128 were extracted. Around 35 $\times$ 10^5 training data was used to train the generator-discriminator pair. The Adam optimizer with initial setting as before was used with a batch-size of 16. To improve GAN stability, we also used smooth labeling of blur and clean features as discussed in [2]. For around 2 $\times$ 10^5 iterations, the training was done with feature costs alone with $\lambda_{\text{adv}} = 0.01$ and $\lambda_1 = 1$. Fine tuning of the generator was subsequently done by adding the MSE cost and weighing down the adversarial cost ($\lambda_2 = 1$, $\lambda_1 = 1$ and $\lambda_{\text{adv}} = 0.001$).

4. Experiments

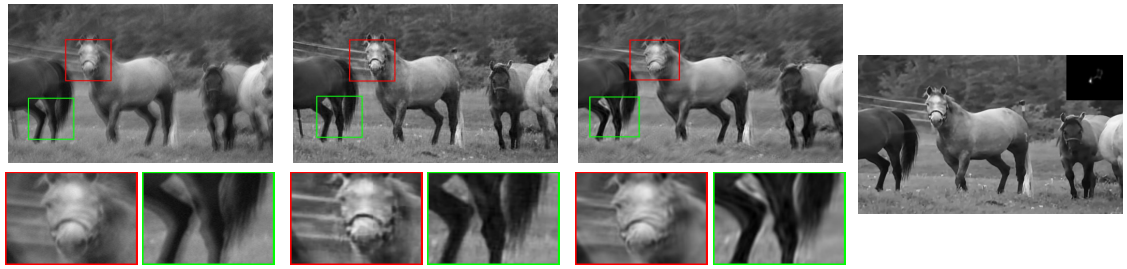
We evaluated the efficacy of our network for deblurring, both quantitatively and qualitatively. We also compared performance with conventional as well as deep networks. For conventional methods, we selected the very recent dark channel prior of [23], l_0 unnatural prior by Xu et al. [37], deblurring method of Whyte et al. [32] and the two-phase kernel method of Xu & Jia [35]. Codes were downloaded from the authors' websites. We also did comparisons with deep learning-based approaches [3] and [28]. In addition to the above, we also tested on images affected by object motion and compared our results with the generalized video deblurring method of [13].

4.1. Comparisons with Conventional Methods

Quantitative evaluation: We performed quantitative comparisons of our method with state-of-the-art methods [23, 37, 32, 35] on the dataset of [29]. The dataset consists of 80 images and 8 kernels totaling to 640 images each of size 1024 $\times$ 700 (on average). The average PSNR and MSSIM (Mean SSIM) measures obtained on each of these images is provided in Table 1. It can be observed from the quantitative measures in Table 1 that our method performs at par with the state-of-the-art. Importantly, our method offers a significant advantage in terms of run-time. The images tested here had an average size of 1024 $\times$

(a) Input (b) [37] (c) [32] (d) [23] (e) Ours

Figure 7: Examples for space-variant deblurring and comparisons with conventional state-of-the-art methods. (a) Input blurred image. (b-e) Results obtained by the methods in [37, 32, 23] and our result, respectively.



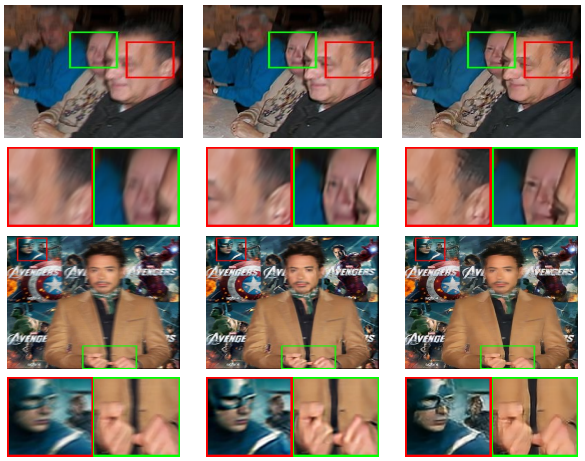
(a) Input (b) Ours (c) Network output of [3] (d) Final output of [3]

Figure 8: Comparison with [3]. (a) Input blurred image. (b) Output of our network. (c-d) Network output and final non-blind deblurred output of [3].

700 and we have reported the run-time for each method (run on Intel I-7 3.4GHz CPU with 8 cores) in Table 2. Note that our method implemented in Torch and run using a Titan-X GPU is at least an order faster than the fastest competitive method [37]. The CPU runtime of our method is 2 minutes which is still faster than most of the comparative methods. Traditional methods are sequential in nature with kernel and image estimated in an alternating fashion. This inherently limits the extent to which they can utilize the parallel processing capability of GPU. Even a GPU re-implementation of these methods will not provide much time savings.

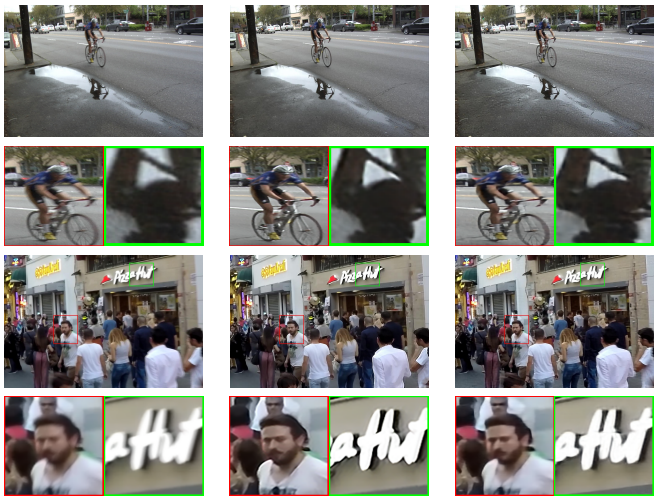
Qualitative evaluation: Figs. 6 and 7 provide qualitative performance of our network compared to conventional methods on space-invariant and space-variant blur, respectively. The results in Fig. 6 clearly reveal that our method (Fig. 6(d)) produces results devoid of any ringing artifacts when compared to [35] and [23]. The butterfly's wing and the lady's collar look sharp and clear. Though the result in Fig. 6(c) also appears to be sharp, there is ringing at the boundaries. The same issue is present in Fig. 6(b) as well.

The efficacy of our method to deal with space-variant blur due to non-uniform camera motion is given in Fig. 7. Here, the input images (first column) are affected by different blurs at different locations. We compared our method with that of [37, 32, 23]. It can be clearly observed that our method outperforms all others. The zoomed-in patch of the bag in the second row as well as the bottle in the fourth row are quite clear and sharp in our result (Fig. 7(e)), in comparison with other outputs. It has to be noted here that we ran all the comparisons using the default settings provided



(a) Input (b) [28] (c) Ours

Figure 9: Comparison with [28]. (a) Space-variantly blurred input images. (b) Result of [28] and (c) our result.



(a) Input (b) [13] (c) Ours

Figure 10: Object motion deblurring. (a) Input. (b) Deblurred result of [13], and (c) our output.

by the authors in their codes. The result of these methods could perhaps improve with parameter tuning (although the 'subway' comparison results are taken directly from [23]); but as already explained in section 1, this is very cumbersome exercise; more so, given the fact that the run-time for the competing methods is quite high. In stark contrast, our method elegantly avoids these pitfalls.

4.2. Comparisons with Deep networks

We also compared our deblurring results with that of [3] and [28]. These are deep network-based approaches but perform the task of kernel estimation. The deblurred results for these methods are obtained by using a non-blind deblurring scheme in the final step. For comparisons with [3] (shown in Fig. 8), we ran our network on the images provided in their paper. The results are compared here with the network output of [3] (Fig. 8(c)) and the final output (Fig. 8(d)) obtained post non-blind deblurring. It must be noted here that our network output (Fig. 8(b)) is significantly better than the network output of Fig. 8(c). Moreover, the method [3] can only handle space-invariant blur.

We also compared our network with [28] that performs parametrized estimation of space-variant blur using deep networks and then uses a conventional method for deblurring. We again ran our network on the images provided by the authors in their paper (Fig. 9 (a)). Note that our model produces results (Fig. 9(c)) that are comparable to that produced by final deblurring in [28] (Fig. 9(b)). The zoomed-in patches of the man and the woman's face in the second row as well as the masked man's face and reporter's hands in the forth row are much sharper in our result.

4.3. Object Motion Deblurring

We also tested on images with blur due to object motion and observed that our network is able to handle even such cases to a resonable extent although it was not trained with such examples. In contrast, conventional deblurring methods [23, 37, 32, 35] that model blur due to camera motion (alone) cannot handle blur due to object motion. Fig. 10 depicts examples of object motion deblurring where the inputs in Fig. 10(a) have blur in the background due to camera motion while the foreground (which is at a differnt depth from the background) incurs blur due to object as well as camera motion. We compared our results (Fig. 10(c)) with the video-based dynamic scene deblurring method of [13] (Fig. 10(b)). The results reveal that our method (Fig. 10(c)) (although single image based) is able to perform at par with the video-based method that uses information from multiple frames.

Additional results and comparisons are provided in the supplementary material.

5. Conclusions

In this paper, we proposed an end-to-end deep network for single image blind-deblurring using autoencoder and GAN. Instead of directly regressing for clean pixels, we perform regression over encoder-features to arrive at a blur-invariant representation which when passed through the decoder produces the desired clean output. Our network is kernel-free, does not require any prior modeling, and can handle both space-invariant and space-variant blur. When evaluated on standard datasets as well as on our own examples, our network performs at par or even better than competing methods while being faster by at least an order.

References

[1] M. Aharon, M. Elad, and A. Bruckstein. rmk-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *Transactions on signal processing*, 54(11):4311–4322, 2006. 1

[2] M. Arjovsky and L. Bottou. Towards principled methods for training generative adversarial networks. In *NIPS 2016 Workshop on Adversarial Training*, volume 2016, 2017. 6

[3] A. Chakrabarti. A neural approach to blind motion deblurring. In *ECCV*, pages 221–235. Springer, 2016. 2, 6, 7, 8

[4] T. F. Chan and C.-K. Wong. Total variation blind deconvolution. *TIP*, 7(3):370–375, 1998. 1

[5] S. Cho and S. Lee. Fast motion deblurring. In *TOG*, volume 28, page 145. ACM, 2009. 1

[6] J. Donahue, P. Krähenbühl, and T. Darrell. Adversarial feature learning. In *ICLR*, 2017. 5

[7] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *TPAMI*, 38(2):295–307, 2016. 3

[8] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html. 6

[9] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, pages 2672–2680, 2014. 4

[10] A. Gupta, N. Joshi, L. Zitnick, M. Cohen, and B. Curless. Single image deblurring using motion density functions. In *ECCV*, 2010. 1

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 4

[12] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006. 3

[13] T. Hyun Kim and K. Mu Lee. Generalized video deblurring for dynamic scenes. In *CVPR*, pages 5426–5434, 2015. 6, 8

[14] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint arXiv:1611.07004*, 2016. 4, 5

[15] N. Joshi, R. Szeliski, and D. J. Kriegman. Psf estimation using sharp edge prediction. In *CVPR*, pages 1–8. IEEE, 2008. 1

[16] D. Kingma and J. B. Adam. A method for stochastic optimisation. In *ICLR*. ICLR, 2015. 6

[17] D. Krishnan, T. Tay, and R. Fergus. Blind deconvolution using a normalized sparsity measure. In *CVPR*, pages 233–240. IEEE, 2011. 1

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012. 2

[19] A. Levin, Y. Weiss, F. Durand, and W. T. Freeman. Understanding blind deconvolution algorithms. *TPAMI*, 33(12):2354–2367, 2011. 1

[20] J. Mairal, J. Ponce, G. Sapiro, A. Zisserman, and F. R. Bach. Supervised dictionary learning. In *Advances in NIPS*, pages 1033–1040, 2009. 1, 2

[21] M. Mathieu, C. Couprie, and Y. LeCun. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*, 2015. 5

[22] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014. 4

[23] J. Pan, D. Sun, H. Pfister, and M.-H. Yang. Blind image deblurring using dark channel prior. In *CVPR*, pages 1628–1636, 2016. 1, 2, 6, 7, 8

[24] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *CVPR*, pages 2536–2544, 2016. 3, 4

[25] D. Perrone and P. Favaro. Total variation blind deconvolution: The devil is in the details. In *CVPR*, pages 2909–2916, 2014. 1

[26] C. J. Schuler, M. Hirsch, S. Harmeling, and B. Schölkopf. Learning to deblur. In *NIPS*, 2014. 2

[27] Q. Shan, J. Jia, and A. Agarwala. High-quality motion deblurring from a single image. In *TOG*, volume 27, page 73. ACM, 2008. 1

[28] J. Sun, W. Cao, Z. Xu, and J. Ponce. Learning a convolutional neural network for non-uniform motion blur removal. In *CVPR*, pages 769–777. IEEE, 2015. 2, 6, 8

[29] 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

[30] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th iICML*, pages 1096–1103. ACM, 2008. 3

[31] D. Warde-Farley and Y. Bengio. Improving generative adversarial networks with denoising feature matching. In *ICLR*. ICLR, 2017. 5

[32] O. Whyte, J. Sivic, A. Zisserman, and J. Ponce. Non-uniform deblurring for shaken images. *IJCV*, 98(2):168–186, 2012. 1, 6, 7, 8

[33] S. Xiang, G. Meng, Y. Wang, C. Pan, and C. Zhang. Image deblurring with coupled dictionary learning. *IJCV*, 114(2-3):248–271, 2015. 1, 2, 3

[34] J. Xie, L. Xu, and E. Chen. Image denoising and inpainting with deep neural networks. In *NIPS*, pages 341–349, 2012. 2, 3

[35] L. Xu and J. Jia. Two-phase kernel estimation for robust motion deblurring. In *ECCV*, pages 157–170. Springer, 2010. 6, 7, 8

[36] L. Xu, J. S. Ren, C. Liu, and J. Jia. Deep convolutional neural network for image deconvolution. In *NIPS*, pages 1790–1798, 2014. 2

[37] L. Xu, S. Zheng, and J. Jia. Unnatural l0 sparse representation for natural image deblurring. In *CVPR*, pages 1107–1114, 2013. 1, 6, 7, 8

[38] A. B. Y Lou and S. Soatto. Direct sparse deblurring. In *Technical Report, CAM-UCLA*, 2009. 1

[39] J. Yang, J. Wright, T. S. Huang, and Y. Ma. Image super-resolution via sparse representation. *TIP*, 19(11):2861–2873, 2010. 1

[40] H. Zhang and L. Carin. Multi-shot imaging: joint alignment, deblurring and resolution-enhancement. In *CVPR*, pages 2925–2932, 2014. 1