Learning Face Deblurring Fast and Wide

Meiguang Jin
University of Bern
Switzerland
jin@inf.unibe.ch

Michael Hirsch†
Amazon Research
Germany
hirsch@amazon.com

Paolo Favaro
University of Bern
Switzerland
favaro@inf.unibe.ch

Abstract

Portrait images and photos containing faces are ubiquitous on the web and the predominant subject of images shared via social media. Especially selfie images taken with lightweight smartphone cameras are susceptible to camera shake. Despite significant progress in the field of image deblurring over the last decade, the performance of state-of-the-art deblurring methods on blurry face images is still limited. In this work, we present a novel deep learning architecture that is designed to be computationally fast and exploits a very wide receptive field to return sharp face images even in challenging scenarios. Our network features an effective resampling convolution operation that ensures a wide receptive field from the very first layers, while at the same time being highly computationally efficient. We also show that batch normalization prevents networks from yielding high-quality image results and introduce instance normalization instead. We demonstrate our architecture on face deblurring as well as other more general scenes. Extensive experiments with state-of-the-art methods demonstrate the effectiveness of our proposed network, in terms of run-time, accuracy, and robustness to ISO levels as well as gamma correction.

1. Introduction

Friends and family are the most popular subjects for photographs according to a recent study in the UK. [1] Moreover in 2015, 93 million digital self-portraits have been taken every day, and 74% of the images shared on Snapchat are selfie images [2]. Hence, photos containing human faces are ubiquitous on the web and have enjoyed widespread attention in the research community enabling numerous applications such as face recognition, face tracking, face swapping, or facial emotion detection [25, 33, 2, 35].

Unfortunately, close-up portraits and selfie images captured with lightweight smartphone cameras are prone to blur stemming from unintended object motion or camera shake during exposure. The problem of recovering the sharp latent image from just a blurry one, without any further information about the blur kernel, is called blind deblurring (BD) or blind deconvolution. Without any prior or additional information, this problem is inherently ill-posed, as there exists an infinite number of latent image and kernel pairs that explain equally well a given blurry image. BD is an important research topic in low level vision that has enjoyed a tremendous research effort especially in the last decade [6, 29, 5, 37, 18, 17, 41, 32, 26, 27, 23].

Many state-of-the-art methods for BD produce high-quality results on images that contain sufficiently strong gradients. However, these deblurring algorithms typically fail on images with little texture and weak edges. Especially for object categories with gradual changes in surface orientation, such as faces, existing deblurring methods do not perform well, but require class-specific prior knowledge in order to obtain satisfactory restoration results [22, 1].

In this paper, we propose a novel convolutional neural network (CNN) to perform blind deconvolution on blurred images of faces. Our proposed network is not only able to handle large blurs, but also takes much less memory and a shorter execution time than competing methods. The success of our network is based on the observation that a large receptive field (RF) is necessary in order to handle large blurs. Moreover, we find that it is important to have a large RF from the very first layer. However, designing a network with such RF is computationally challenging, because the memory footprint and the computations increase as the RF grows larger. To address these challenges our network features an effective resampling convolution operation, akin to strided convolutions, that ensures a wide RF from the start while at the same time allowing a tradeoff between computational efficiency and accuracy. This approach is inspired by recent work in superresolution [30] and demosaicking [7].

Our network is trained in an end-to-end manner by using

† The scientific idea and code were developed prior to joining Amazon.

[1] https://www.ofcom.org.uk/__data/assets/pdf_file/0022/20668/cmr_uk_2015.pdf

[2] http://www.rawhide.org/blog/infographics/

synthetically generated pairs of sharp and blurred example images with a fixed gamma correction and ISO level. We show that our network is very robust as it can generalize to real blurry images and handle different gamma corrections and ISO levels. Several experiments and extensive evaluations with state-of-the-art methods demonstrate the effectiveness of our proposed network for high-quality face image deblurring.

2. Prior Work

General deblurring. Blind deconvolution has been studied for several decades and today many efficient and effective methods are available [6, 29, 5, 37]. Most methods explore the sparsity of image gradients [37, 14, 38, 40, 23, 17, 41, 32, 26, 27]. Some methods also exploit patch similarities [18] and, recently, a dark channel prior [24]. These methods have been designed for shift-invariant blur. However, real blur may vary from one pixel to another. While some shift-invariant methods may be robust to non-uniform blur [37, 38, 32, 26, 23] methods designed for the more general case of camera shake can perform better in those cases [11, 36].

Face deblurring. While current methods work well on natural scenes, they tend to not generalize well on some specific image categories, such as faces. Indeed, methods that are specific to faces have been proposed [9, 22, 1]. [9] requires an additional sharp reference example that contains shared content with the blurry photo. [22] deblurs a face image by exploring facial structures from an exemplar dataset. Using a dataset presents several challenges due to the high complexity of real images, which might contain multiple faces, have occlusions due to glasses, scarfs, hands, etc. [1] uses a class-specific image prior for recovering spatial frequencies attenuated by the blurring process. However, it does not generalize well for non-uniform blur.

Neural networks for blind deblurring. Due to the tremendous development of deep learning, convolutional neural network (CNN) based approaches are becoming mainstream. [31, 8] predict the motion field from a blurry input image. However, this method only describes locally linear blur. [28] uses a CNN to learn features that are informative about the unknown blur kernel and implements quotient layers to enable end-to-end training. [3] builds a CNN to estimate Fourier coefficients for each patch and then estimates a global blur kernel from the assembled patches. Very recently, [19] proposed a multi-scale Gaussian pyramid CNN to deal with the very general case of dynamic scene deblurring (blur is generated by generic object and camera motion). The success of their model is due to the multi-scale architecture, which aims to break down the complexity of the problem by ensuring that each level in the pyramid receives as input an image from the previous level with an amount of blur that is small enough to be handled by the CNN.

Our proposed network brings about several innovations despite its apparent simplicity. In the case of dynamic scene deblurring, we address a limitation of the pyramid structure used by [19]. This structure limits the receptive field via downsampling and upsampling operations between the pyramid levels. In our network, however, we show that allowing for a large receptive field is extremely beneficial, as the last layers of the network can exploit a higher level of abstraction, better understand the image content and ultimately determine a better deblurring output. This is shown in particular in Fig. 5. This ability of better interpreting the images is also reflected in a strong robustness to several variations in the data that we do not explicitly train for. For example, we show that our network can handle different gamma corrections and ISO levels, although all image pairs in the training set share the same settings. As in the recent dilated convolution networks [39] and Atrous convolution networks [4], our network aims at achieving a wide receptive field. However, in contrast to dilated and Atrous convolutions, we rely on combining resampling with convolutions, an operation equivalent to strided convolutions. The resampling scheme we use relates also to [30]. However, we only rearrange the input instead of downsampling or upsampling it.

3. Blind Deconvolution and Inverse Filtering

Blind deconvolution is the problem of recovering a sharp image $\mathbf{u}$ and a blur kernel $\mathbf{k}$ from a blurry and noisy image $\mathbf{g}$ such that

$$\mathbf{g} \simeq \mathbf{k} * \mathbf{u} \tag{1}$$

where $*$ denotes the convolution operator. Alternative formulations of this problem ask for just the recovery of the blur or the sharp image, since the estimation of the other unknown is typically easier given the former. In fact, the vast majority of algorithms developed in the last decade follow the approach of estimating first the blur kernel $\mathbf{k}$ and then the sharp image $\mathbf{u}$ in a second step. Moreover, given the blur $\mathbf{k}$, the estimation of $\mathbf{u}$ can be formulated as a convex optimization problem and solved very efficiently [16].

In recent years, however, several methods based on neural networks have sought to solve directly for the sharp image $\mathbf{u}$. Understanding how these networks achieve this goal seems still elusive, and this makes the design of better performing networks a challenge. In this paper, we focus our attention on the RF of a neural network and argue that it has a strong impact on its performance. Our motivation to look at the RF stems from considerations about how blur and

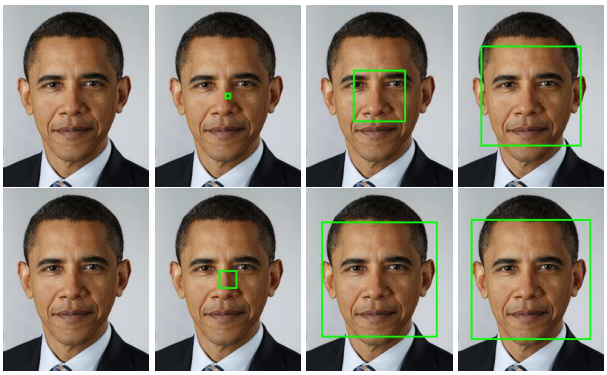

Figure 1. Visualization of receptive fields for each step of [19]'s network and our proposed network. The first column shows the inputs to both networks. On the first row from the second position from the left we show the receptive field of Nah's network (at the highest pyramid level) after the first convolutional layer, after 10 residual blocks, and at the output. On the second row from the second position from the left we show the receptive field of our network after the first convolutional layer, after the deblurring network, and at the output.

sharp images can be recovered from a blurry image. Our first observation is that the identification of the blur and the sharp image from a single blurry input is an ambiguous task. In fact, for any blurry image $\mathbf{g}$ there exist infinite $\mathbf{k}$ and $\mathbf{u}$ pairs that satisfy Eq. (1). However, most pairs will have a sharp image $\mathbf{u}$ with artifacts that make it unlike a real image. Thus, disambiguation could be achieved by restricting the space of admissible sharp images. Intuitively, the space of small sharp images (or image patches) will have very little or no structure, so that almost all points in this space will be admissible. Conversely, the space of large images will have strong structural constraints so that one can hope to better identify the correct $\mathbf{k}$ and $\mathbf{u}$ pair. Therefore, we argue that an algorithm tasked with the blind deconvolution problem should produce better results if each output pixel is the result of processing a large region of the input image. This corresponds to designing a network with a large RF.

4. Network Architecture Design

To break down the complexity of achieving a large RF, recent neural networks for blind deconvolution use a pyramidal scheme [19]. In this scheme one builds a network split in a number of blocks (see Fig. 3.b), each operating at a different resolution, starting with the smallest resolution at the input and gradually growing it to the final output resolution in the last network block. The first driving principle is that deblurring a downsampled version of the input image should be an easier task, since its blur has a smaller extent than in the original input. The second principle is that a network block should benefit from an input image that is more similar to the desired output, the deblurred image.

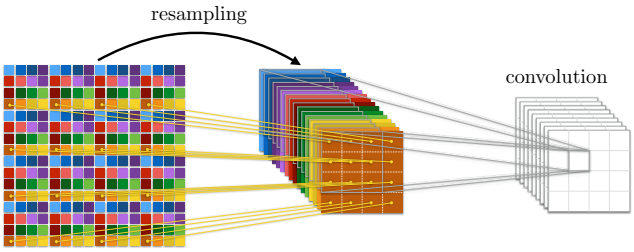

Figure 2. **Resampling and filtering**. From left to right: input image, resampled input (input pixels are only rearranged), filtering with convolutional layers applied to the resampled input.

Another way to interpret the benefit of pyramidal schemes is that they make the same convolutional filters work at a larger scale (*i.e.*, with a larger RF) when the input images are downsampled. So, instead of making the filters larger, this approach makes the images smaller. The flip side of this strategy is that these filters (in the initial blocks) are equivalent to very smooth large kernels convolved with the original image and such that their output is undersampled. Moreover, these networks require upsampling layers between each block, which are known to introduce gridding artifacts [21]. Finally, another limitation is that each scale increases the RF very gradually by starting with a very small RF. As shown on the first row of Fig. 1, the RF of the highest pyramid level of this network is initially 5×5 pixels and then gradually reaches 161×161 pixels. The RFs of the lower pyramid levels are larger due to downsampling. However, the filters of these levels correspond to coarse filters at the highest pyramid level.

In our network (shown in Fig. 3), we address these limitations by avoiding downsampling and upsampling altogether. We propose to approximate convolutions with large kernels by exploiting convolutions applied to resampled data. The resampled data is obtained by taking all possible undersamplings of the d-dimensional input in the spatial coordinates (see the resampling mapping in Fig. 2) and stacking them as separate channels in a d+1-dimensional tensor. In this way a convolution applied to one of the channels corresponds to an Atrous (or dilated) convolution [4, 39], while a convolution applied to all channels corresponds to a strided convolution with a very large filter. The immediate advantage of this arrangement is that the RF can be large from the start (see the second row of Fig. 1) without a major impact on the computational load and memory footprint.

4.1. Resampling

Consider an input image $\mathbf{u}$ with $M \times N \times C$ pixels (the leftmost image in Fig. 2 shows $\mathbf{u}$ with $C = 1$). The resampled image $\hat{\mathbf{u}}$ will have $M/s \times N/s \times s^2C$ pixels, where $s \geq 1$ is the chosen resampling stride (center image in Fig. 2, with $s = 4$). $\hat{\mathbf{u}}$ has a smaller spatial extent, but

more channels than $\mathbf{u}$. A convolutional filter after resampling can then access pixels far away in the original image $\mathbf{u}$ (see rightmost image in Fig. 2). Resampling and a convolutional layer are the first components in our network (see Fig. 3, where we have chosen $s = 5$) .

4.2. Evaluation of the receptive field

Here, we briefly discuss the receptive field size Ω on the input image of each layer of the proposed network. To facilitate the ablation studies later on, we let the resampling stride s and the number of residual blocks b be free variables. Because the filters and resampling use the same quantities along both the horizontal and vertical axes, we evaluate just one side of the RF.

We start from the first convolutional layer (conv_1) in Fig. 3. The RF $\Omega_{\text{conv}_1} = 5s$. One residual block adds $4s$ pixels to the previous RF. Thus, after b blocks we have an RF of $\Omega_{\text{8resblock}} = 5s + 4bs$. After the deblurring block we have an un-resampling layer and 4 convolutions with 3×3 kernels. This results in a final RF of $\Omega_{\text{output}} = 5s + 4bs + 8$. In the case of $s = 5$ and $b = 8$, we have an initial RF of $\Omega_{\text{conv}_1} = 25$ and an output RF of $\Omega_{\text{output}} = 193$ (see Fig. 1).

4.3. Residual blocks and instance normalization

An important component of our architecture is the residual block structure and the use of instance normalization [34]. The design of networks that reconstruct residual terms (*i.e.*, the difference between the desired output and the given input) has been introduced by [10] for object classification and detection, and later used also in deblurring and other image restoration problems [19]. Residual learning has been shown to increase performance with an increasing depth of the network and to improve training.

In our deblurring network we employ $b = 8$ residual blocks with 96 input/output channels as shown in Fig. 3 (a). Each residual block uses the recently proposed instance normalization (IN) [34], where the reparametrization does not depend on the batch, but just on the input sample itself. This can be written as

$$\text{IN}(\mathbf{x}) = \gamma \frac{\mathbf{x} - \mu(\mathbf{x})}{\sigma_\epsilon(\mathbf{x})} + \beta \qquad (2)$$

where γ and β are two layer parameters, μ is the mean and σ_ϵ the (regularized) standard deviation computed across the spatial dimensions of the input $\mathbf{x}$, but independently for each feature channel. The benefits of instance normalization are illustrated in Fig. 4. We compare the output of 3 different normalization choices for the residual blocks shown in Fig. 3 (b): With conventional batch normalization (BN), with instance normalization (IN) or with neither of them. One can observe a dramatic improvement in the amount of detail and overall accuracy when IN is used.

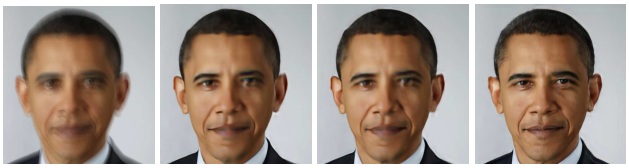

Figure 4. Visual comparisons between different normalizations. From left to right are an input from [15], the result of batch normalization, no normalization and instance normalization.

4.4. Blending it all together

Since the processing of the channels of the resampled output is carried out independently (there is no direct link between the filters in the convolutional layers), the final texture reconstruction after the *un-resampling* could show some pattern artifacts (see image on the un-resampling layer in Fig. 3 (a)). One can see the 5×5 pattern due to the stride $s = 5$. We therefore introduce a blending subnetwork at the end of the deblurring subnetwork, that can reweigh and smooth out these artifacts. Firstly, we undo the resampling of the processed image to restore the original input image size and then apply a few convolutional layers (with very small filters, so that processing is local). The result of the blending network is shown in Fig. 3 (a) at the output layer.

5. Experiments

We evaluate our proposed network both quantitatively and qualitatively for various scenarios. Firstly, we carry out ablation studies to demonstrate experimentally the impact of each choice in our design. Then, we show that our proposed scheme achieves state-of-the-art performance on the blind deconvolution task on the synthetic face dataset of [15]. We achieve this performance while using fewer parameters and with a significantly smaller runtime than competing methods. We also test our network on real images: one dataset consists of face images with unknown (generic) motion blur and gamma correction and a second dataset is the standard camera shake dataset of [13].

Training dataset and implementation details. In our network, we choose a resampling factor $s = 5$ and use $b = 8$ residual blocks with 96 channels within the deblurring network and another residual block with 32 channels in the blending network (see Fig. 3). All convolutional layers use 3×3 filters except the first convolutional layer in the deblurring network, where 5×5 filters are used. The whole network contains $18bC^2 + 34Cs^2 + 19008 \simeq 1.4$ million trainable parameters with $C = 96$. For training, we collect face images from the FaceScrub dataset [20] and crop 110K sharp images of size 320×320. Additionally, we generate 10K synthetic random motion kernels as implemented in [3]. Blurry images are generated on the fly by convolving a sharp image with a kernel and by adding white

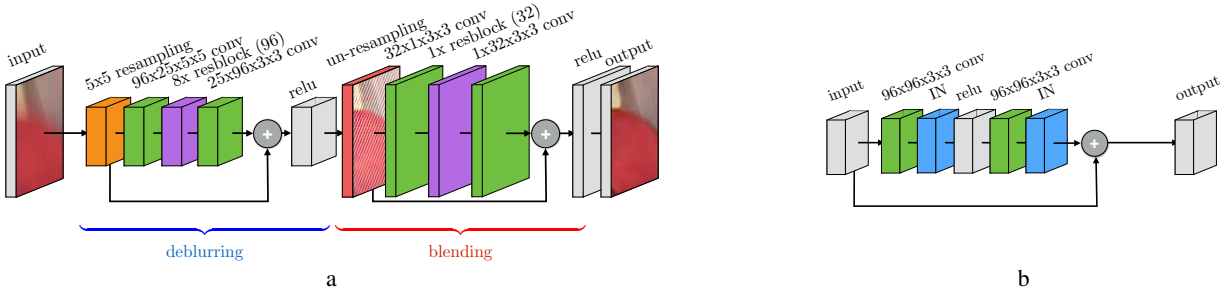

Figure 3. (a) Our wide receptive field network architecture. It is made of two main parts: the deblurring network and the blending network. Within the deblurring network we repeat 8 times the residual block shown in (b). (b) Our proposed residual block in the case of 96 input and output channels, with instance normalization (IN) [34], and 3×3 filters.

Gaussian noise with standard deviation 2.55, thereby generating enough training data to avoid overfitting. Notice that we train with gray scale images, but our network is able to handle color images by processing each color channel separately. The network is implemented using torch featuring CUDA 8.0 and cuDNN 5.1. The training is done on two Titan X Pascal GPUs with mini batch size 48 and 120K iterations. We use the Adam solver [12] with an initial learning rate of 2×10^{-3} and a 10% decrease after every 12K iterations.

Synthetic blurry face images benchmark. We quantitatively evaluate our face deblurring network by testing our network on the dataset of [15], which includes 20 synthetic blurry face images generated by convolving 5 different face images with 4 different non-uniform blur kernels obtained from 6D inertial measurement sensor data. In Table 1 we show the performance (the peak signal to noise ratio, or PSNR, and the structural similarity index measure, or SSIM) of our *wide receptive field network (WRFN)* along with several other competing methods. The WRFN performs better in terms of average PSNR and on the worst case than previous state of the art, which evidences the robustness of our network. To better demonstrate the capabilities of our network we also train Nah's network [19] on our blurred FaceScrub dataset. We can see that the performance of WRFN is still significantly better. All results will be available on the project page [3].

Ablation studies. We evaluate the impact of the normalization layers in the residual blocks on the synthetic dataset of [15] and show the performance in Table 1. We train the WRFN with batch normalization (BN), instance normalization (IN), or without any normalization (NO). We observe that batch normalization may be harmful to the WRFN and make little difference compared to not using normalization. Visual comparison between different normalization schemes can be found in Fig. 4.

Table 1. Results on synthetic images from [15].

	PSNR		SSIM	
Method	Ave	Worst	Ave	Worst
Fergus	22.870	16.700	0.682	0.450
Cho	23.272	18.690	0.699	0.479
Xu10	25.586	20.939	0.773	0.603
Krishnan	23.070	19.216	0.716	0.618
Levin	21.855	17.878	0.651	0.551
Whyte	23.232	20.495	0.667	0.550
Sun	24.649	20.397	0.756	0.561
Xu13	25.319	20.060	0.765	0.629
Zhang	22.918	20.337	0.679	0.548
Zhong	23.440	18.895	0.723	0.518
Pan14	23.193	13.842	0.691	0.262
Michaeli	24.726	21.375	0.754	0.617
Perrone	24.843	21.265	0.754	0.607
Sun	24.309	21.256	0.708	0.564
Anwar	23.605	20.355	0.683	0.536
Chakrabarti	25.389	19.235	0.769	0.540
Pan16	24.460	18.101	0.754	0.574
Pan17	23.297	18.187	0.740	0.560
Nah	24.224	21.354	0.713	0.571
Gong	23.805	18.747	0.694	0.566
Nah (our dataset)	25.925	22.600	0.754	0.663
WRFN - BN	26.370	23.204	0.771	0.681
WRFN - NO	26.373	23.148	0.772	0.682
WRFN - IN	**27.136**	**23.967**	**0.803**	**0.707**
s=4, C=84, b=11	27.020	23.963	0.796	0.695
s=4, C=96, b=11	27.022	23.963	0.798	0.696
swap, s=5, C=96, b=8	26.947	23.739	0.796	0.692
crop, s=5, C=96, b=8	26.751	23.872	0.785	0.681

We then compare slight variations of the WRFN (with settings $s = 5$, $b = 8$ and $C = 96$) where we try to match the final receptive field $\Omega_{\text{output}} = 193$ and the total number of parameters 1,432,666 and vary instead the initial recep-

[3] http://www.cvg.unibe.ch/publications/projects/meiguang%20jin/faceDeblur.html

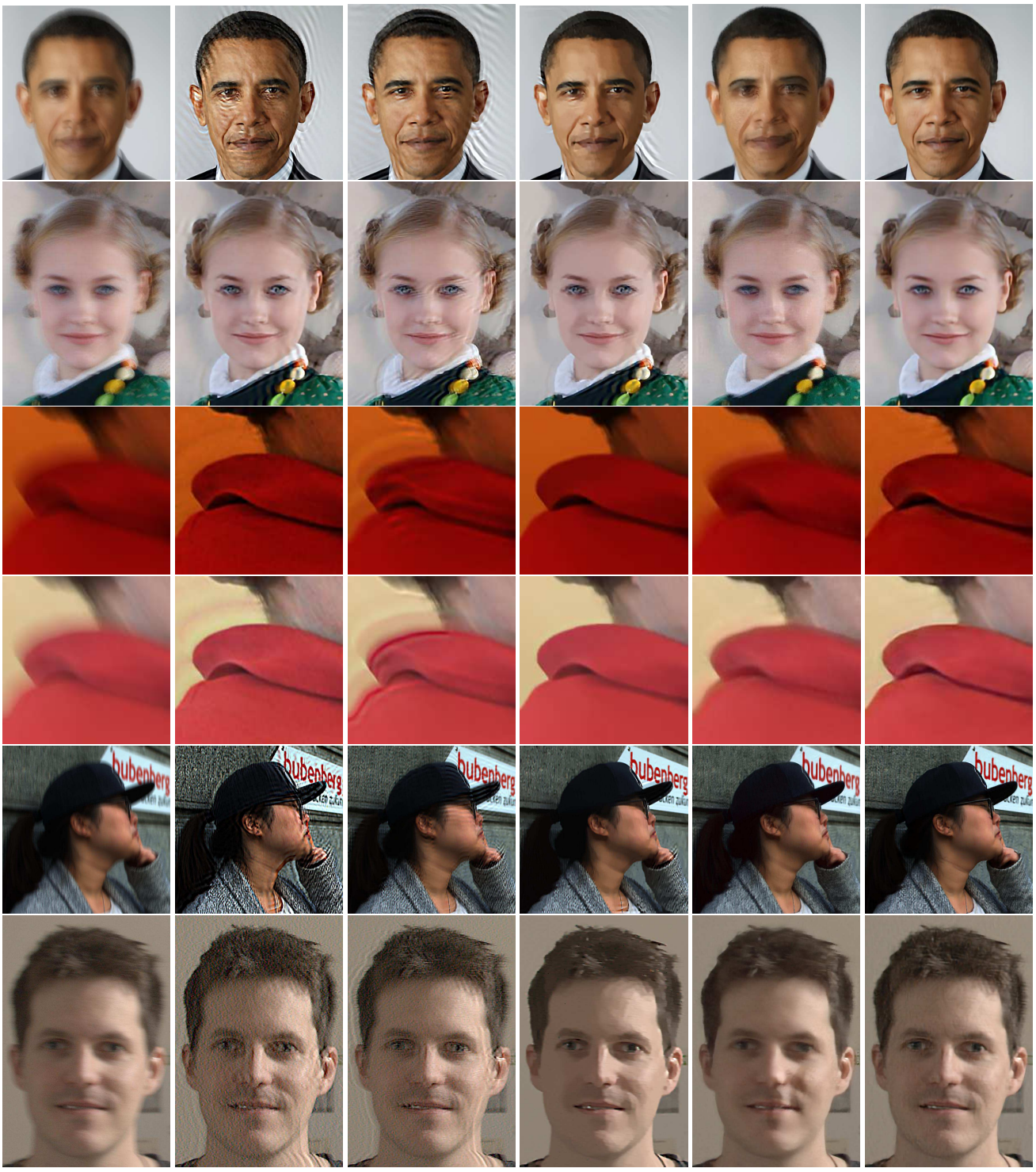

Figure 5. Visual comparisons on both synthetic and real face images. From left to right columns are, blurry input, results of [22], [3], [24], [19], and ours. The first two row inputs are from [15] and the other four inputs are captured with a DSLR.

tive field by changing the resampling stride s. To look at a smaller initial receptive field we choose $s = 4$. The corresponding number of blocks is $b = 11$ and the number of channels is $C = 84$. In this case we have a final receptive field of $\Omega_{\text{output}} = 204$ (slightly larger than the reference) and the total number of parameters is 1,467,661. Notice that despite the slightly larger receptive field and total number of parameters, the performance of this network is worse

than that of the reference WRFN. Also, in order to demonstrate that the loss of performance is not due to the number of channels in the residual blocks, we also train and test a variant with $s = 4$, $b = 11$ and $C = 96$, *i.e.*, with the same number of channels as in the reference WRFN.

To further highlight the importance of starting with a large receptive field, we also swap the first (5×5 kernel) with the last (3×3 kernel) convolutional layer of our proposed deblurring network. By doing so, the initial receptive field is smaller than in the unswapped case, but the total number of parameters and the final receptive field are the same. As shown in Table 1 under the case `swap`, the performance drops.

Finally, we found important that the input data during training is larger than the output receptive field. To test the WRFN in this case, we train the reference WRFN with smaller input images by cropping them to 240×240 pixels. Such small input sizes make the boundary more significant than the central image region. We argue that this might cause a loss of performance. With cropping the WRFN observes less data at each iteration during training. To compensate for this loss we also iterate the stochastic gradient descent longer. This still results in a loss of performance.

Robustness on real face images. In this set of experiments we show that although our network has been trained only with shift invariant motion blur, it is still able to deal with non-uniform blur. The first and second rows of Fig. 5 show results on real images. We observe that our network is capable of handling large blurs without introducing artifacts, while other methods either fail to deblur or generate undesirable artifacts (*e.g.*, ringing). Additionally, we test our network for gamma correction, non-frontal pose, and strong noise. On the third and fourth rows of Fig. 5, we show deblurring results for images w/o gamma correction whereby a gamma of 2.2 is used. Here, we only include a comparison with the state-of-the-art general deblurring methods of [24], state-of-the-art face deblurring approach [22] and two deep learning based approaches [3, 19]. More visual comparisons with other methods and full view of the third and fourth rows images can be found in the supplementary material. From the results, we can see that our network is quite robust to gamma correction. Note that our network has neither been trained for different gamma values nor does it know the gamma value in advance. Interestingly, although our network has been trained on face images, it is capable to deblur the text in the background of the image as shown on the fifth row of Fig. 5. On the sixth row of Fig. 5, we show an example, which we captured under a large ISO(3200) setting. This case demonstrates that the WRFN is quite robust to strong noise, while the other two methods either generate noise artifacts or tend to oversmooth their output.

Generalization on real general scenes. To see the generalization ability of our network, we tested it on general scenes. In Fig. 6, we show the comparison between [19] and our network. On the first three rows we use input images from [15]. We also tested the WRFN on [13]'s dataset, which contains 48 camera shake blurry images. The WRFN achieves a PSNR of 27.7dB, while [19]'s network achieves 25.9dB. On the last four rows of Fig. 6 we show four examples from this dataset. In all cases we observe that our network generalizes better than [19].

Resampling factor. The choice of the resampling factor is in principle arbitrary. However, performance can be heavily impacted by it. We illustrate two extreme cases to show the trade-off between different choices. If a small factor, *e.g.*, 2 is chosen for training, then one has to build a deep enough network to ensure a large receptive field. This choice has two main limitations: 1) The system requires a large memory footprint during execution; 2) The execution time suffers from a deeper network. On the other hand, choosing a large resampling factor of 10 or more brings two advantages: 1) The network does not have to be deep and hence requires less memory; 2) The execution is faster. However, there is also a limitation: As the resampling factor increases, the number of trainable parameters increases quadratically. As a result, more data samples are required for training.

Resampling vs strided convolutions. Ideally, our resampling convolution produces identical results to strided convolutions. However, in practice, execution performance can be different depending on various filter sizes, strides, and input sizes. Strided convolutions are more efficient when filter size and resampling factor are small. We find that for a filter size between 5 and 7 and a resampling factor between 5 and 8, resampling convolution benefits more compared to other choices. At the same time we find that a large resampling factor will inevitably increase the overhead, and the speed gain will decrease. Comparisons between resampling and strided convolution can be found in Table 3.

Runtime comparison. In Table 2, we show the execution time of several competing methods for different image sizes. We report the average runtime for 10 runs. GPU based approaches are measured with an Nvidia Titan X Pascal. Our network is almost 2 magnitudes faster than existing CNN based approaches and $4 - 5$ magnitudes faster than non-CNN CPU based blind deblurring approaches. In our architecture, most computations are performed at the resampling stage, while other networks perform most of the computation at the original scale.

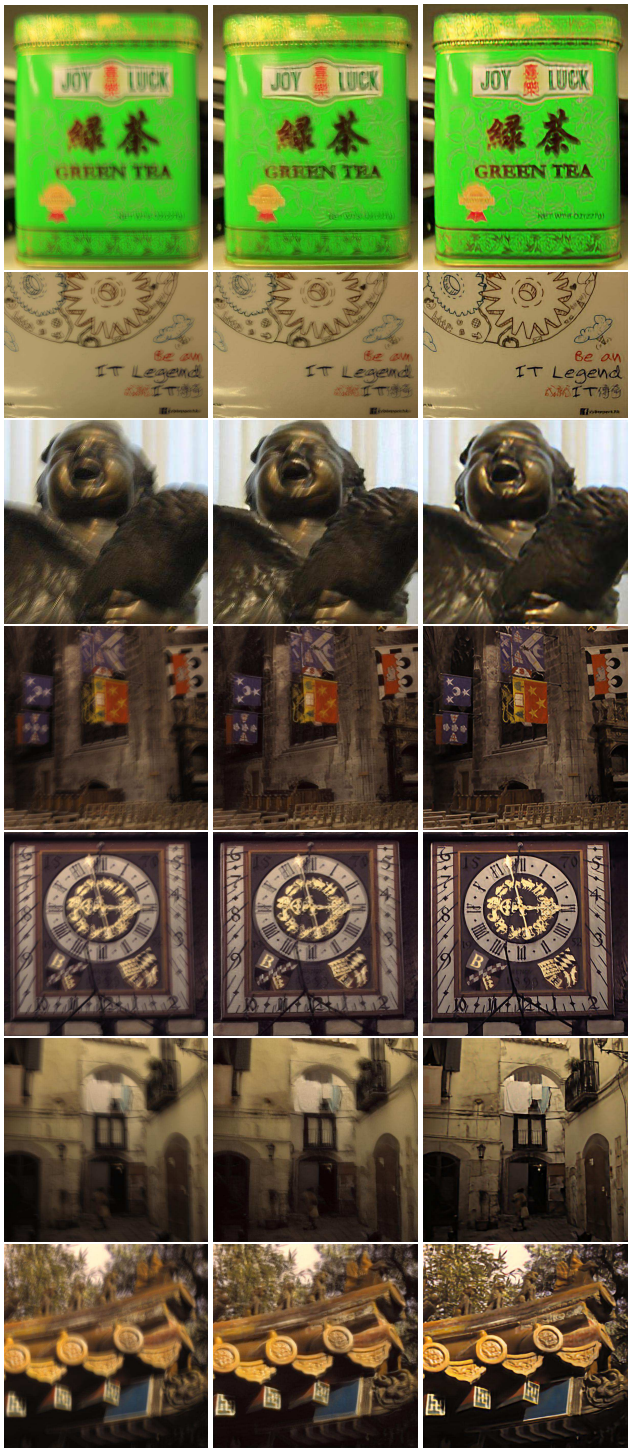

Figure 6. Visual comparisons on real general scenes. From left to right are inputs, results of [19]'s and ours. The first three row inputs are from [15] and the last four row blurry inputs are from [13].

Table 2. Runtime and parameter space comparisons on different color image sizes.

method		size	processing	#params
	500^2	1000^2		
Sun	>8m	>45m	GPU+CPU	>7M
Chakrabarti	>5m	-	GPU+CPU	>100M
Pan16	>10m	>1h	CPU	-
Pan17	>60s	>3m	GPU	-
Nah	2.9s	5.6s	GPU	11.7M
Ours	**0.023s**	**0.086s**	GPU	1.4M

Table 3. Execution runtime ratio between strided convolution and resampling convolution on different image sizes. Strided convolutions seem to thrive with small filters and small strides. Vice versa, resampling convolutions are more computationally efficient with larger filters and strides, which we use in our network.

Filter size	stride s	Forward			Backward		
		315^2	630^2	1260^2	315^2	630^2	1260^2
3×3	3	0.6	0.5	0.4	0.8	0.6	0.7
3×3	5	0.9	0.9	0.7	**1.0**	**1.1**	0.9
3×3	7	**1.1**	**1.2**	0.8	**1.1**	**1.1**	**1.0**
3×3	9	**1.2**	0.9	0.9	**1.1**	**1.1**	**1.1**
5×5	3	**1.0**	**1.1**	**1.2**	0.8	0.8	0.8
5×5	5	**1.1**	**1.3**	**1.2**	**1.0**	**1.2**	**1.3**
5×5	7	0.9	**1.5**	**1.9**	**1.1**	**1.3**	**1.6**
5×5	9	0.9	**1.1**	**1.4**	**1.2**	**1.3**	**1.4**
7×7	3	**1.3**	**1.3**	**1.5**	0.9	0.8	**1.0**
7×7	5	**1.3**	**1.5**	**1.4**	**1.0**	**1.3**	**1.2**
7×7	7	0.7	**1.9**	**1.6**	0.9	**1.4**	**1.6**
7×7	9	0.7	**1.2**	**1.6**	**1.1**	**1.2**	**1.3**

6. Conclusion

Zillions of digital self-portraits and photos of human faces are captured and shared online every day. Despite the tremendous progress in blind image deblurring, generic state-of-the-art methods fail to recover high-quality images from blurry photos due to the lack of strong salient edges. In this work we proposed a discriminative learning approach that enables high-quality restoration of blurry portrait photos under a wide variety of conditions. In particular, our method is able to handle images with unknown gamma. As a deblurring function we use a deep convolutional neural network featuring multi-channel resampling convolutions, enabling both efficient computation and a wide receptive field. In numerous synthetic and challenging real-world examples we demonstrate the efficacy of our approach.

Acknowledgements. MJ and PF acknowledge support from the Swiss National Science Foundation on project 200021_153324.

References

[1] S. Anwar, C. Phuoc Huynh, and F. Porikli. Class-specific image deblurring. In *ICCV*, 2015. 1, 2

[2] D. Bitouk, N. Kumar, S. Dhillon, P. N. Belhumeur, and S. K. Nayar. Face swapping: automatically replacing faces in photographs. *ACM Trans. Graph*, 2008. 1

[3] A. Chakrabarti. A neural approach to blind motion deblurring. In *ECCV*, 2016. 2, 4, 6, 7

[4] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE TPAMI*, 2017. 2, 3

[5] S. Cho and S. Lee. Fast motion deblurring. *ACM Trans. Graph*, 2009. 1, 2

[6] R. Fergus, B. Singh, A. Hertzmann, S. T. Roweis, and W. T. Freeman. Removing camera shake from a single photograph. *ACM Trans. Graph*, 2006. 1, 2

[7] M. Gharbi, G. Chaurasia, S. Paris, and F. Durand. Deep joint demosaicking and denoising. *ACM Trans. Graph*, 2016. 1

[8] D. Gong, J. Yang, L. Liu, Y. Zhang, I. Reid, C. Shen, A. v. d. Hengel, and Q. Shi. From motion blur to motion flow: a deep learning solution for removing heterogeneous motion blur. In *CVPR*, 2017. 2

[9] Y. HaCohen, E. Shechtman, and D. Lischinski. Deblurring by example using dense correspondence. In *ICCV*, 2013. 2

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

[11] M. Hirsch, C. J. Schuler, S. Harmeling, and B. Scholkopf. Fast removal of non-uniform camera shake. In *ICCV*, 2011. 2

[12] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 5

[13] R. Köhler, M. Hirsch, B. J. Mohler, B. Schölkopf, and S. Harmeling. Recording and playback of camera shake: Benchmarking blind deconvolution with a real-world database. In *ECCV*, 2012. 4, 7, 8

[14] D. Krishnan, T. Tay, and R. Fergus. Blind deconvolution using a normalized sparsity measure. In *CVPR*, 2011. 2

[15] W. Lai, J. Huang, Z. Hu, N. Ahuja, and M. Yang. A comparative study for single image blind deblurring. In *CVPR*, 2016. 4, 5, 6, 7, 8

[16] A. Levin, Y. Weiss, F. Durand, and W. T. Freeman. Understanding and evaluating blind deconvolution algorithms. In *CVPR*, 2009. 2

[17] A. Levin, Y. Weiss, F. Durand, and W. T. Freeman. Efficient marginal likelihood optimization in blind deconvolution. In *CVPR*, 2011. 1, 2

[18] T. Michaeli and M. Irani. Blind deblurring using internal patch recurrence. In *ECCV*, 2014. 1, 2

[19] S. Nah, T. H. Kim, and K. M. Lee. Deep multi-scale convolutional neural network for dynamic scene deblurring. In *CVPR*, 2017. 2, 3, 4, 5, 6, 7, 8

[20] H. Ng and S. Winkler. A data-driven approach to cleaning large face datasets. In *ICIP*, 2014. 4

[21] A. Odena, V. Dumoulin, and C. Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016. 3

[22] J. Pan, Z. Hu, Z. Su, and M. Yang. Deblurring face images with exemplars. In *ECCV*, 2014. 1, 2, 6, 7

[23] J. Pan, Z. Hu, Z. Su, and M. Yang. L_0-regularized intensity and gradient prior for deblurring text images and beyond. *IEEE TPAMI*, 2017. 1, 2

[24] J. Pan, D. Sun, H. Pfister, and M.-H. Yang. Blind image deblurring using dark channel prior. In *CVPR*, 2016. 2, 6, 7

[25] O. M. Parkhi, A. Vedaldi, and A. Zisserman. Deep face recognition. In *BMVC*, 2015. 1

[26] D. Perrone and P. Favaro. Total variation blind deconvolution: The devil is in the details. In *CVPR*, 2014. 1, 2

[27] D. Perrone and P. Favaro. A logarithmic image prior for blind deconvolution. *IJCV*, 2016. 1, 2

[28] C. J. Schuler, M. Hirsch, S. Harmeling, and B. Schölkopf. Learning to deblur. *IEEE TPAMI*, 2016. 2

[29] Q. Shan, J. Jia, and A. Agarwala. High-quality motion deblurring from a single image. *ACM Trans. Graph*, 2008. 1, 2

[30] W. Shi, J. Caballero, F. Huszar, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *CVPR*, 2016. 1, 2

[31] J. Sun, W. Cao, Z. Xu, and J. Ponce. Learning a convolutional neural network for non-uniform motion blur removal. In *CVPR*, 2015. 2

[32] L. Sun, S. Cho, J. Wang, and J. Hays. Edge-based blur kernel estimation using patch priors. In *ICCP*, 2013. 1, 2

[33] J. Thies, M. Zollhöfer, M. Stamminger, C. Theobalt, and M. Nießner. Face2Face: Real-time face capture and reenactment of RGB videos. In *CVPR*, 2016. 1

[34] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In *CVPR*, 2017. 4, 5

[35] Z. Wang, S. Wang, and Q. Ji. Capturing complex spatio-temporal relations among facial muscles for facial expression recognition. In *CVPR*, 2013. 1

[36] O. Whyte, J. Sivic, A. Zisserman, and J. Ponce. Non-uniform deblurring for shaken images. *IJCV*, 2012. 2

[37] L. Xu and J. Jia. Two-phase kernel estimation for robust motion deblurring. In *ECCV*, 2010. 1, 2

[38] L. Xu, S. Zheng, and J. Jia. Unnatural L_0 sparse representation for natural image deblurring. In *CVPR*, 2013. 2

[39] F. Yu, V. Koltun, and T. Funkhouser. Dilated residual networks. In *CVPR*, 2017. 2, 3

[40] H. Zhang, D. P. Wipf, and Y. Zhang. Multi-image blind deblurring using a coupled adaptive sparse prior. In *CVPR*, 2013. 2

[41] L. Zhong, S. Cho, D. N. Metaxas, S. Paris, and J. Wang. Handling noise in single image deblurring using directional filters. In *CVPR*, 2013. 1, 2