Fast Image Processing with Fully-Convolutional Networks

Qifeng Chen* Jia Xu* Vladlen Koltun
Intel Labs

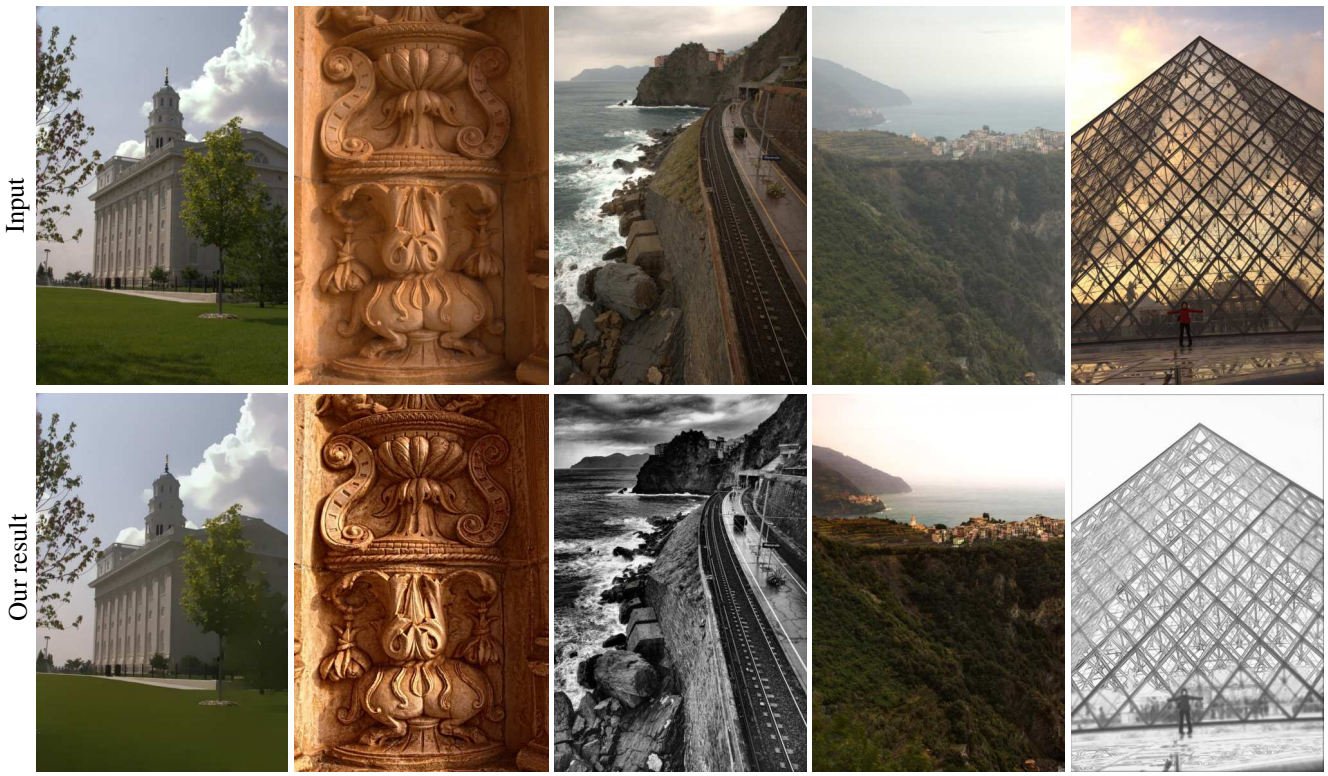

Figure 1. We present an approach to approximating image processing operators. This figure shows the results for five operators: L_0 gradient minimization, multiscale tone manipulation, photographic style transfer, nonlocal dehazing, and pencil drawing. All operators are approximated by the same model, with the same set of parameters and the same flow of computation.

Abstract

We present an approach to accelerating a wide variety of image processing operators. Our approach uses a fully-convolutional network that is trained on input-output pairs that demonstrate the operator's action. After training, the original operator need not be run at all. The trained network operates at full resolution and runs in constant time. We investigate the effect of network architecture on approximation accuracy, runtime, and memory footprint, and identify a specific architecture that balances these considerations. We evaluate the presented approach on ten advanced image processing operators, including multiple variational models, multiscale tone and detail manipulation, photographic style transfer, nonlocal dehazing, and nonphotorealistic stylization. All operators are approximated by the same model. Experiments demonstrate that the presented approach is significantly more accurate than prior approximation schemes. It increases approximation accuracy as measured by PSNR across the evaluated operators by 8.5 dB on the MIT-Adobe dataset (from 27.5 to 36 dB) and reduces DSSIM by a multiplicative factor of 3 compared to the most accurate prior approximation scheme, while being the fastest. We show that our models generalize across datasets and across resolutions, and investigate a number of extensions of the presented approach.

*Joint first authors

1. Introduction

Research in image processing has yielded a variety of advanced operators that produce visually striking effects. Techniques developed in the last decade can dramatically enhance detail [24, 69, 26, 28, 60], transform the image by applying a master photographer's style [7, 5], smooth the image for the purpose of abstraction [73, 76, 79], and eliminate the effects of atmospheric scattering [25, 35, 27, 9]. This is accomplished by a variety of algorithmic approaches, including variational methods, gradient-domain processing, high-dimensional filtering, and manipulation of multiscale representations.

The computational demands and running times of existing operators vary greatly. Some operators, such as bilateral filtering, have benefitted from more than a decade of concerted investment in their acceleration. Others still take seconds or even minutes for high-resolution images. While most existing techniques can be accelerated by experts given sufficient research and development time, such acceleration schemes often require significant expertise and may not generalize across operators.

One general approach to accelerating a broad range of image processing operators is well-known: downsample the image, execute the operator at low resolution, and upsample [45, 34, 14]. This approach suffers from two significant drawbacks. First, the original operator must still be evaluated on a lower-resolution image. This can be a severe handicap because some operators are slow and existing implementations cannot be executed at interactive rates even at low resolution. Second, since the operator is never evaluated at the original resolution, its effects on the high-frequency content of the image may not be modeled properly. This can limit the accuracy of the approximation.

In this paper, we investigate an alternative approach to accelerating image processing operators. Like the downsample-evaluate-upsample approach, the presented method approximates the original operator. Unlike the downsampling approach, the method operates on full-resolution images, is trained end-to-end to maximize accuracy, and does not require running the original operator at all. To approximate the operator, we use a convolutional network that is trained on input-output pairs that demonstrate the action of the operator. After training, the network is used in place of the original operator, which need not be run at all.

We investigate the effects of different network architectures in terms of three properties that are important for accelerating image processing operators: approximation accuracy, runtime, and compactness. We identify a specific architecture that satisfies all three criteria and show that it approximates a wide variety of standard image processing operators extremely accurately. We evaluate the presented approach on ten advanced image processing operators, including multiple forms of variational image smoothing, adaptive detail enhancement, photographic style transfer, and dehazing. All operators are approximated using an identical architecture with no hyperparameter tuning. Five of the trained approximators are demonstrated in Figure 1, which shows their action on images from the MIT-Adobe 5K test set (not seen during training).

For all evaluated operators, the presented approximation scheme outperforms the downsampling approach. For example, the PSNR of our approximators across the ten considered operators on the MIT-Adobe test set is 36 dB, compared to 25 dB for the high-accuracy variant of bilateral guided upsampling [14]. At the same time, our approximators are faster than the fastest variant of that scheme. Our approximators run in constant time, independent of the runtime of the original operator.

We conduct extensive experiments that demonstrate that our simple approach outperforms a large number of recent and contemporary baselines, and that trained approximators generalize across datasets and to image resolutions not seen during training. We also investigate a number of extensions and show that the presented approach can be used to create parameterized networks that expose parameters that can be used to interactively control the effect of the image processing operator at test time; to train a single network that can emulate many diverse image processing operators and combine their effects; and to process video.

2. Related Work

Many schemes have been developed for accelerating image processing operators. The bilateral filter in particular has benefitted from long-term investment in its acceleration [21, 72, 15, 59, 2, 1, 29, 8]. Another family of dedicated acceleration schemes addresses the median filter and its variants [72, 61, 54, 80]. Other work has examined the acceleration of variational methods [6, 62, 13, 17], gradient-domain techniques [46], convolutions with large spatial support [23], and local Laplacian filters [5]. (Deep mathematical connections between these families of operators exist [57].) While many of these schemes successfully accelerate their intended families of operators, they do not have the generality we seek.

A general approach to accelerating image processing operators is to downsample the image, evaluate the operator at low resolution, and upsample [45, 34, 14]. This approach accelerates a broad range of operators by approximating them. It is largely agnostic to the operator but requires that the operator avoid spatial transformation so that the original image can be used to guide the upsampling. (E.g., no spatial warping such as perspective correction.) Our method shares a number of characteristics with the downsampling approach: it targets a broad range of operators, uses an approximation, and assumes that the spatial layout of the im-

age is preserved. However, our approximation has a much richer parameterization that can model the operator's effect on the high-frequency content of the image. Once trained, the approximator does not need to execute the original operator at all. We will show that our method is more accurate than the downsampling approach on a wide range of tasks, while being faster.

Other work on accelerating image processing considers the system infrastructure and programming language. Given a powerful cloud backend and a bandwidth-limited network connection, high-resolution processing can be offloaded to the cloud [32]. Domain-specific languages can be used to schedule image processing pipelines to better utilize available hardware resources [63, 36]. Our work is complementary and provides an approach to approximating a wide variety of operators with a uniform parameterization. Such uniform parameterization and predictable flow of computation can assist further acceleration using dedicated hardware.

The closest works to ours are due to Xu et al. [75], Liu et al. [51], and Yan et al. [77]. We review each in turn. Xu et al. [75] used deep networks to approximate a variety of edge-preserving filters. Our work also uses deep networks, but differs in key technical decisions, leading to substantially broader scope and better performance. Specifically, the approach of Xu et al. operates in the gradient domain and requires reconstructing the output image by integrating the gradient field produced by the network. Since their networks produce non-integrable gradient fields, the authors had to constrain the final image reconstruction by introducing an additional data term that forces the output to be similar to the input. For this and other reasons, the approach of Xu et al. only applies to edge-preserving smoothing, has limited approximation accuracy, exhibits high running times (seconds for 1 MP images), and requires operator-specific hyperparameter tuning. In comparison, we train an approximator end-to-end, pixels to pixels, using a parameterization that is deeper and more context-aware while being more compact. We will demonstrate experimentally that the presented approach yields higher accuracy and lower runtimes while fitting a much bigger family of operators.

Liu et al. [51] combined a convolutional network and a set of recurrent networks to approximate a variety of image filters. This approach is quite flexible and outperforms the approach of Xu et al. on some operators, but does not achieve the approximation accuracy and speed we seek. We will show that a single convolutional network can achieve higher accuracy, while being faster and more compact.

Yan et al. [77] also applied deep networks to image adjustment. This work is also related to ours in its idea of approximating image transformations by deep networks. However, our work differs substantially in scope, technical approach, and results. Yan et al. use a fully-connected network that operates on each pixel separately. The network itself has a receptive field of a single pixel. Contextual information is only provided by hand-crafted input features, instead of being collected adaptively by the network. This places a substantial burden on manual feature design. In contrast, our approximator is a single convolutional network that is trained end-to-end, aggregates spatial context from the image as needed, and does not rely on extraneous modules or preprocessing. This leads to much greater generality, higher accuracy, and faster runtimes.

Deep networks have been used for denoising [39, 11, 3], super-resolution [10, 19, 40, 42, 41, 48, 50], deblurring [74], restoration of images corrupted by dirt or rain [22], example-based non-photorealistic stylization [30, 70, 40], joint image filtering [49], dehazing [64], and demosaicking [31]. None of the approaches described in these works were intended as broadly applicable replacements for the standard downsample-evaluate-upsample approach to image processing acceleration. Indeed, our experiments have shown that many approaches lack in either *speed*, *accuracy*, or *compactness* when applied across a broad range of operators. These criteria will be explored further in the next section.

3. Method

3.1. Preliminaries

Let $\mathbf{I}$ be an image, represented in the RGB color space. Let f be an operator that transforms the content of an image without modifying its dimensions: that is, $\mathbf{I}$ and $f(\mathbf{I})$ have the same resolution. We will consider a variety of operators f that use a broad range of algorithmic techniques. Our goal is to approximate f with another operator $\hat{f}$, such that $\hat{f}(\mathbf{I}) \approx f(\mathbf{I})$ for all images $\mathbf{I}$. Note that the resolution of $\mathbf{I}$ is not restricted: both the operator f and its approximation $\hat{f}$ are assumed to operate on variable-resolution images. Furthermore, we will consider many operators $\{f_i\}$ but require that our corresponding approximations $\{\hat{f}_i\}$ all share the same parameterization: same set of parameters, same flow of computation. The approximations will differ only in their parameters, which will be fit for each operator during training.

Our goal is to find a broadly applicable approach to accelerating image processing operators. We have identified three desirable criteria for such an approach. *Accuracy*: We seek an approach that provides high approximation accuracy across a broad range of popular image processing operators. *Speed*: The approach must be fast, ideally achieving interactive rates on HD images. *Compactness*: We seek an approach that can potentially be deployed within the constraints of mobile devices. An ideal network would have a very compact parameterization that can fit into on-chip SRAM, and a small memory footprint [33].

Our basic approach is to approximate the operator using a convolutional network [47]. The network must operate on variable-resolution images and must produce an output image at the same resolution as the input. This is known as dense prediction [52]. In principle, any fully-convolutional network architecture can be used for this purpose. Specifically, any network that has been used for a pixelwise classification problem such as semantic segmentation can instead be trained with a regression loss to produce continuous color rather than a discrete label per pixel. However, not all network architectures will yield high accuracy in this regime and most are not compact.

We have experimented with a large number of network architectures derived from prior work in high-level vision, specifically on semantic segmentation. We found that when some of these high-level networks are applied to low-level image processing problems, they generally outperform dedicated architectures previously designed for these image processing problems. The key advantage of architectures designed for high-level vision is their large receptive field. Many image processing operators are based on global optimization over the entire image, analysis of global image properties, or nonlocal information aggregation. To model such operators faithfully, the network must collect data from spatially distributed locations, aggregating information at multiple scales that are ultimately large enough to provide a global view of the image.

In Section 3.2 we describe an architecture that strikes the best balance between the different desiderata according to our experiments. Three alternative fully-convolutional architectures are described in the supplement.

3.2. Context aggregation networks

Our primary architecture is the multi-scale context aggregation network (CAN), developed in the context of semantic image analysis [78]. Its intermediate representations and its output have the same resolution as the input. Contextual information is gradually aggregated at increasingly larger scales, such that the computation of each output pixel takes into account all input pixels within a window of size exponential in the network's depth. This accomplishes global information aggregation for high-resolution images with a very compact parameterization. We will see that this architecture fulfills all of the desiderata outlined above.

We now describe the parameterization in detail. The data is laid out over multiple consecutive layers: $\{\mathbf{L}^0, \ldots, \mathbf{L}^d\}$. The first and last layers $\mathbf{L}^0, \mathbf{L}^d$ have dimensionality $m \times n \times 3$. These represent the input and output images. The resolution $m \times n$ varies and is not given in advance.

Each intermediate layer $\mathbf{L}^s$ $(1 \leq s \leq d-1)$ has dimensionality $m \times n \times w$, where w is the width of (i.e., the number of feature maps in) each layer. The content of intermediate layer $\mathbf{L}^s$ is computed from the content of the previous layer $\mathbf{L}^{s-1}$ as follows:

$$\mathbf{L}_i^s = \Phi\left(\Psi^s\left(b_i^s + \sum_j \mathbf{L}_j^{s-1} *_{r_s} \mathbf{K}_{i,j}^s\right)\right). \quad (1)$$

Here $\mathbf{L}_i^s$ is the i^{th} feature map of layer $\mathbf{L}^s$, $\mathbf{L}_j^{s-1}$ is the j^{th} feature map of layer $\mathbf{L}^{s-1}$, b_i^s is a scalar bias, and $\mathbf{K}_{i,j}^s$ is a 3×3 convolution kernel. The operator $*_{r_s}$ is a dilated convolution with dilation r_s. The dilated convolution operator is the means by which the network aggregates long-range contextual information without losing resolution. Specifically, for image coordinates $\mathbf{x}$:

$$(\mathbf{L}_j^{s-1} *_{r_s} \mathbf{K}_{i,j}^s)(\mathbf{x}) = \sum_{\mathbf{a}+r_s\mathbf{b}=\mathbf{x}} \mathbf{L}_j^{s-1}(\mathbf{a})\mathbf{K}_{i,j}^s(\mathbf{b}). \quad (2)$$

The effect of dilation is that the filter is tapped not at adjacent locations in the feature map, but at locations separated by the factor r_s. The dilation is increased exponentially with depth: $r_s = 2^{s-1}$ for $1 \leq s \leq d-2$. For $\mathbf{L}^{d-1}$, we do not use dilation. For the output layer $\mathbf{L}^d$ we use a linear transformation (1×1 convolution with no nonlinearity) that projects the final layer into the RGB color space.

For the pointwise nonlinearity Φ, we use the leaky rectified linear unit (LReLU) [55]: $\Phi(x) = \max(\alpha x, x)$, where $\alpha = 0.2$. Ψ^s is an adaptive normalization function, described in Section 3.3. Additional specification of the CAN architecture is provided in the supplement.

The network aggregates global context via full-resolution intermediate layers. It has a large receptive field while being extremely compact. It also has a small memory footprint during the forward pass. Since no skip connections across non-consecutive layers are employed, only two layers need to be kept in memory at any one time. Since the layers are all structurally identical, two fixed memory buffers are sufficient, with data flowing back and forth between them.

3.3. Adaptive normalization

We have found that using batch normalization improves approximation accuracy on challenging image processing operators such as style transfer and pencil drawing, but degrades performance on other image processing operators. We thus employ adaptive normalization that combines batch normalization and the identity mapping:

$$\Psi^s(x) = \lambda_s x + \mu_s BN(x), \quad (3)$$

where $\lambda_s, \mu_s \in \mathbb{R}$ are learned scalar weights and BN is the batch normalization operator [37]. The weights $\{\lambda_s, \mu_s\}$ are learned by backpropagation alongside all other parameters of the network [67]. Learning these weights allows the model to adapt to the characteristics of the approximated operator, adjusting the strengths of the identity branch and the batch normalization branch as needed.

3.4. Training

The network is trained on a set of input-output pairs that contain images before and after the application of the original operator: $\mathcal{D} = \{\mathbf{I}_i, f(\mathbf{I}_i)\}$. The parameters of the network are the kernel weights $\mathcal{K} = \{\mathbf{K}^s_{i,j}\}_{s,i,j}$ and the biases $\mathcal{B} = \{b^s_i\}_{s,i}$. These parameters are optimized to fit the action of the operator f across all images in the training set. We train with an image-space regression loss:

$$\ell(\mathcal{K}, \mathcal{B}) = \sum_i \frac{1}{N_i} \left\| \hat{f}(\mathbf{I}_i; \mathcal{K}, \mathcal{B}) - f(\mathbf{I}_i) \right\|^2, \qquad (4)$$

where N_i is the number of pixels in image $\mathbf{I}_i$. This loss minimizes the mean-squared error (MSE) in the RGB color space across the training set. Although MSE is known to have limited correlation with perceptual image fidelity [71], experiments will demonstrate that training an approximator to minimize MSE will also yield high accuracy in terms of other measures such as PSNR and SSIM.

We have also experimented with more sophisticated losses, including perceptual losses that match feature activations in a visual perception network [10, 20, 40, 48, 16] and adversarial training [20, 38, 48]. We found that the higher-level feature matching losses did not increase approximation accuracy in our tasks; the image processing operators we target are not semantic in nature and can be approximated well by directly fitting the operator's action on the photographic content of the image. Adversarial training is known to be unstable [4, 56, 16] and we found that it also did not increase the already excellent results that we were able to obtain with an appropriate network architecture and a direct image-space loss.

Creating the training set $\mathcal{D}$ only requires running the original operator f on a set of images. Training can thus be conducted on extremely large datasets that can be generated automatically without human intervention, although we found that training on a few thousand images already produces approximators that generalize well.

In order to expose the training to the effects of the operator f on images of different resolutions, we use images of varying resolution for training. Specifically, given a set of high-resolution images, each is automatically resized to a random resolution between 320p and 1440p (e.g., 517p) while preserving its aspect ratio. These resized images are used for training. Training uses the Adam solver [43] and proceeds for 500K iterations (one randomly sampled image per iteration). This takes roughly one day on our test workstation.

4. Experiments

Experimental setup. We evaluate the presented approach on ten image processing operators: Rudin-Osher-Fatemi [66], TV-L^1 image restoration [58], L_0 smoothing [73], relative total variation [76], image enhancement by multiscale tone manipulation [24], multiscale detail manipulation based on local Laplacian filtering [5, 60], photographic style transfer from a reference image [5], dark-channel dehazing [35], nonlocal dehazing [9], and pencil drawing [53]. The operators, their effect on images, and our reference implementations are described in the supplement.

We use two image processing datasets: MIT-Adobe 5K and RAISE [12, 18]. MIT-Adobe 5K contains 5,000 high-resolution photographs covering a broad range of scenes, subjects, and lighting conditions. We use the default 2.5K/2.5K training/test split. The RAISE dataset contains 8,156 high-resolution RAW images captured by four photographers over a period of three years, depicting different scenes and moments across Europe. We use 2.5K randomly sampled images for training and 1K other randomly sampled images for testing.

We ran all ten operators on all images from the training and test sets of both datasets. For each operator, the input-output pairs from the MIT-Adobe training set were used for training. The same models and training procedures were used for all operators. The only difference between the ten approximators is in the output images that were provided in the training set. For each architecture, this procedure yielded ten identically parameterized models, trained to approximate the respective operators. These approximators are used for most of the experiments, which are conducted on the MIT-Adobe test set.

The same procedure was performed using the RAISE training set. This yielded models trained to approximate the same operators on the RAISE dataset. These models will be used to test cross-dataset generalization.

Main results. Our primary baseline is bilateral guided upsampling (BGU) [14], the state-of-the-art form of the downsample-evaluate-upsample scheme for accelerating image processing operators. There are two variants of the BGU approach, both with publicly available implementations. The first uses global optimization and is designed to approximate the original operator as closely as possible. The second is an approximation scheme designed to maximize speed, which was implemented in Halide [63] with specific attention to parallelization, vectorization, and data locality. We will compare to both variants of BGU, referred to respectively as BGU-opt and BGU-fast. We use the public implementations with the default parameters.

We also compare to a large number of baseline approaches that have used deep networks for related problems. The closest of these are the deep edge-aware filters of Xu et al. [75] and the recursive filters of Liu et al. [51]. Beyond this, we also evaluate the image transformation approach of Johnson et al. [40], which was developed for style transfer and superresolution but can be applied more broadly. Finally, we compare to the contemporaneous work of Isola

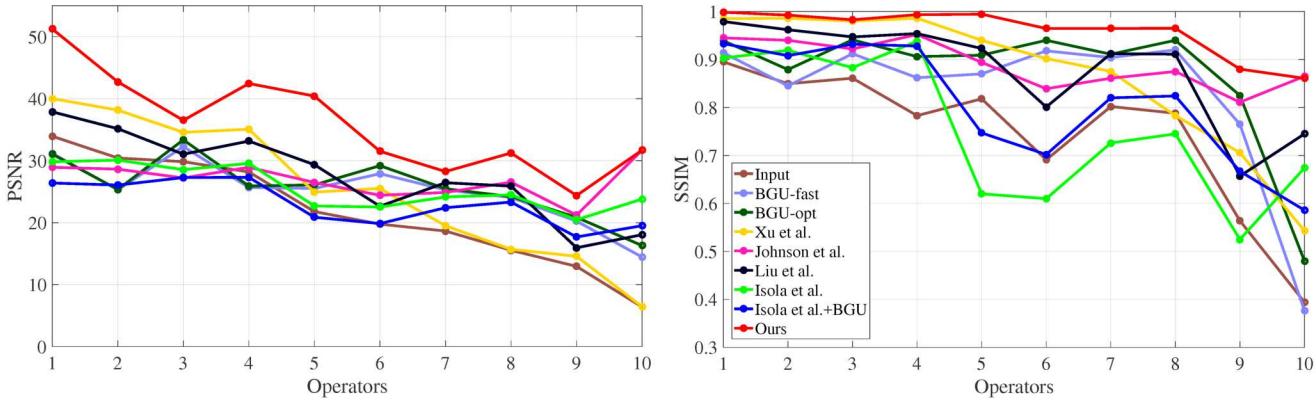

Figure 2. Approximation accuracy on the MIT-Adobe test set. Operators are arranged along the horizontal axis. From 1 to 10: Rudin-Osher-Fatemi [66], TV-L^1 image restoration [58], L_0 smoothing [73], relative total variation [76], image enhancement by multiscale tone manipulation [24], multiscale detail manipulation based on local Laplacian filtering [5, 60], nonlocal dehazing [9], dark-channel dehazing [35], photographic style transfer from a reference image [5], and pencil drawing [53].

et al. [38], who proposed an approach to "image-to-image translation" based on adversarial training. The approach of Isola et al. differs from the other baselines in that it is not fully-convolutional and is set up to operate at fixed resolution (256×256). We report results for two versions of this baseline: one in which the output images are upsampled to the original resolution by bilinear interpolation, and one in which the output is upsampled using BGU-opt.

Approximation accuracy achieved by each approach for each of the ten operators is visualized in Figure 2. All the numerical results are listed in the supplement. Our default model is a CAN with adaptive normalization, using $d = 9$ and $w = 24$ for the depth and width, respectively. This is the model referred to as 'Ours' in Figure 2 and Table 1. For each image, the output of each approach is compared to the output of the original reference operator, and the distance between the two images is evaluated in terms of PSNR and SSIM [71]. For each operator, the results are averaged over the MIT-Adobe test set. We also use a trivial baseline for calibration, referred to as Input. This trivial baseline simply uses the input image with no modification and thus evaluates the distance between the input image and the output of the reference operator. The Input baseline shows how a trivial approximation scheme (doing nothing) would fare and also provides an indication of how strongly the reference operator alters the image.

Due to the high computational demands of some of the reference operators, all images were scaled to 1080p resolution (∼1.75 MP) for this comprehensive experiment. We will evaluate cross-resolution performance in a subsequent experiment. Note that a resolution of 1080p had no special significance during training: the models were trained on images with randomly sampled resolution.

Average accuracy and runtime for each approach across all ten operators is summarized in Table 1. The runtime of each approach on each specific operator is reported in the supplement. The CAN parameterization is extremely compact: the network has a total of 37K parameters. It approximates the reference operators extremely accurately, achieving SSIM above 0.99 on four of the operators and SSIM above 0.96 on eight of them. (See the supplement for detailed results on the individual operators.)

Method	MSE	PSNR	SSIM	Time (ms)	# of param
Reference	–	–	–	9,502	–
Input	2607.9	21.75	0.745	–	–
BGU-fast [14]	521.8	24.70	0.827	320	–
BGU-opt [14]	413.5	25.27	0.865	2,378	–
Xu et al. [75]	2347.3	25.45	0.869	5,493	312K
Johnson et al. [40]	215.0	26.89	0.890	203	1,678K
Liu et al. [51]	383.8	27.56	0.879	458	152K
Isola et al. [38]	279.5	25.62	0.754	198	57,184K
Isola et al. [38]+BGU	457.2	23.07	0.805	2,352	57,184K
Ours	**59.1**	**36.04**	**0.960**	**190**	**37K**

Table 1. Average accuracy, runtime, and number of parameters across all ten operators on the MIT-Adobe test set. Runtime is measured on images at 1080p resolution (∼1.75 MP).

Compared to our main baselines, BGU-opt and BGU-fast, our approach increases PNSR by 11 dB (from ∼25 to 36) and reduces DSSIM (=(1-SSIM)/2) by a multiplicative factor of 3. The downsampling approach does not perform well when the action of the operator at high resolution cannot be recovered from its output at low resolution. In contrast, our approach models the action of the operator directly at the original resolution. Our approach is also faster than BGU-fast and is more than an order of magnitude faster than BGU-opt. Runtime was measured on a workstation with an Intel i7-5960X 3.0GHz CPU and an Nvidia Titan X GPU. The runtime of BGU varies across operators, see the sup-

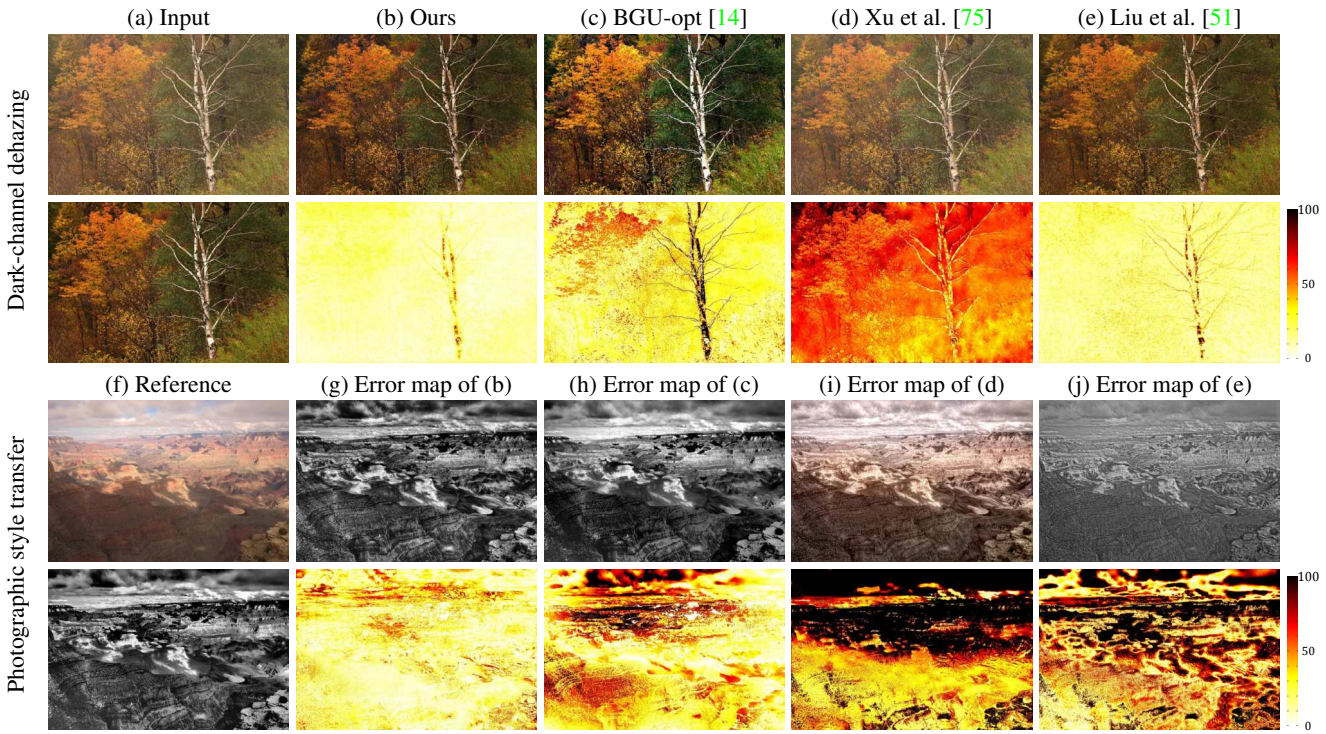

Figure 3. Qualitative results on images from the MIT-Adobe test set. For each operator, we show the input image, the result of the original reference operator, the result produced by our approximator, and results produced by BGU-opt [14], Xu et al. [75], and Liu et al. [51]. The error maps show per-pixel error, measured by Euclidean distance in 0-255 RGB space. Black indicates error of 100 or higher. Additional visualizations are provided in the supplement.

plement for detailed results. The runtime of our approach is constant. It is 40 ms (25 fps) for 480p images, 190 ms for 1080p images, and scales linearly in the number of pixels. We used a standard deep learning library (TensorFlow) with no additional performance tuning.

Of the prior approaches that use deep networks, Liu et al. [51] and Johnson et al. [40] achieve the best approximation accuracy. Our approach outperforms these baselines by 8.5 dB in PNSR, and reduces DSSIM by a multiplicative factor of 3. Our approach is also the fastest and has the most compact parameterization. Qualitative results are shown in Figure 3 and in the supplement.

Additional experiments. We now compare a number of different CAN configurations to alternative fully-convolutional architectures. These alternative architectures – Plain, Encoder-decoder [65], and FCN-8s [52, 68] – are described in detail in the supplement. All these models are trained by the same procedure as the CAN.

The results are summarized in Table 2. Here CAN24+AN is our primary model, referred to as 'Ours' in Table 1 ($d = 9$, $w = 24$, adaptive normalization). CAN32+AN is a more accurate but slower configuration ($d = 10$, $w = 32$, adaptive normalization). This configuration benefits from a receptive field of 513×513 versus the 257×257 receptive field of CAN24. We also evaluate two other variants of CAN32, controlling for the effect of adaptive normalization: CAN32 (no normalization) and CAN32+BN (BatchNorm). Finally, Table 2 also reports the performance of a single network (CAN32+AN) that represents all ten operators; this network is described in Section 5.

Method	MSE	PSNR	SSIM	Time (ms)	# of param
FCN-8s	344.1	26.36	0.808	150	30,510K
Encoder-decoder	177.9	34.90	0.950	139	7,760K
Plain	369.7	32.05	0.920	**118**	75K
CAN32	133.4	35.52	0.956	162	75K
CAN32+BN	129.9	28.64	0.929	243	75K
CAN24+AN	59.1	36.04	0.960	190	**37K**
CAN32+AN	**36.0**	**37.59**	**0.966**	277	75K
Single network	110.3	29.86	0.931	385	78K

Table 2. Average accuracy, running time, and number of parameters of different network architectures over all ten operators on the MIT-Adobe test set. Running time is measured on images at 1080p resolution (∼1.75 MP).

Cross-resolution generalization. We now test how the trained approximators generalize across resolutions. To

keep the time of the experiment manageable, we focus on the L_0 smoothing operator for this purpose. Recall that our approximator was trained on images resized to random resolutions between 320p and 1440p. We now compare the trained model to baselines on a set of specific resolutions: 320p, 480p, 720p, 1080p, 1440p, and 2160p. For this purpose, the MIT-Adobe test set was resized to each of these resolutions, the reference operator was executed on these images, and all methods were evaluated at each resolution. The results are shown in the supplement. They indicate that the accuracy of our approximator is stable and outperforms the other approaches across resolutions. Note that the 2160p condition ($\sim$7 MP) tests the generalization of our model to resolutions never seen during training. (The maximal resolution used during training was 1440p.)

Cross-dataset generalization. We have also evaluated how the trained operators generalize across datasets. To this end, for each operator, we tested two models on the MIT-Adobe test set: one trained on the MIT-Adobe training set and one trained on the RAISE training set. Similarly, for each operator, we tested two models on the RAISE test set: one trained on the RAISE training set and one trained on the MIT-Adobe training set. The detailed results are given in the supplement. They indicate that the trained approximators generalize extremely well and effectively represent the underlying action of the reference operators. The accuracy in corresponding conditions (e.g., MIT $\rightarrow$ MIT and RAISE $\rightarrow$ MIT) is virtually identical.

Ablation studies. Additional controlled experiments on network depth and width are reported in the supplement.

5. Extensions

We now describe three extensions of the presented approach: representing parameterized operators, representing multiple operators by a single network, and video processing.

Parameterized operators. An image processing operator can have parameters that control its action. For example, variational image smoothing operators [66, 58, 73] commonly have a parameter λ that controls the relative strength of the regularizer: higher λ leads to more aggressive smoothing. Other operators, such as multiscale tone manipulation, have multiple meaningful parameters that can be used to control the operator's effect [24]. Our approach extends naturally to creating parameterized approximators that expose these degrees of freedom at test time. To this end, we add channels to the input layer. For each parameter we wish to expose, we add an input channel that is used to communicate the parameter's value to the network. During training, we apply the operator with randomly sampled parameter values, thus showing the network the effect of the parameter on the operator. Quantitative results are reported in the supplement and qualitative results are shown in the video.

One network to represent them all. So far, we have trained separate networks for different operators, albeit with identical parameterizations. We now show that all 10 operators can be represented by a single network, which can emulate any of the individual operators at test time. This shows that a single compact network can execute a large number of advanced image processing effects at high accuracy. To this end, we augment the input layer by adding 10 additional channels, where each channel is a binary indicator that corresponds to one of the 10 operators. During training, we randomly sample an operator and an input image in each iteration. Training proceeds for 500K iterations total, as in the other experiments. For this experiment we use the CAN32 configuration with adaptive normalization.

The approximation accuracy achieved by the trained network across the 10 operators is reported in Table 2. The accuracy on each individual operator is given in the supplement. Remarkably, a single compact network that represents all 10 operators achieves high accuracy, well above the most accurate prior approximation scheme (compare to the results in Table 1). The trained network is demonstrated in the supplementary video. As shown in the video, the network can also smoothly transition between the operators when it is given continuous values in the auxiliary input channels, even though it was trained with one-hot vectors only.

Video processing. We also apply the trained models to videos from the Tanks and Temples dataset [44]. This further demonstrates cross-dataset generalization. (The models were trained on the MIT-Adobe dataset.) We simply apply the approximator to each frame. Although no provisions are made for temporal coherence, the results are as coherent as the original operators. The results are shown in the supplementary video.

6. Conclusion

We have presented an approach to approximating a wide range of image processing operators. All operators are approximated with the same parameterization and the same flow of computation. We have shown that the presented approach significantly outperforms prior approximation schemes.

We see the uniform and regular flow of computation in the presented model as a strong advantage. While the model is already faster than baselines using a generic implementation, we expect that significant further acceleration can be achieved.

References

[1] A. Adams, J. Baek, and M. A. Davis. Fast high-dimensional filtering using the permutohedral lattice. *Computer Graphics Forum*, 29(2), 2010. 2

[2] A. Adams, N. Gelfand, J. Dolson, and M. Levoy. Gaussian KD-trees for fast high-dimensional filtering. *ACM Transactions on Graphics*, 28(3), 2009. 2

[3] F. Agostinelli, M. R. Anderson, and H. Lee. Adaptive multi-column deep neural networks with application to robust image denoising. In *NIPS*, 2013. 3

[4] M. Arjovsky and L. Bottou. Towards principled methods for training generative adversarial networks. In *ICLR*, 2017. 5

[5] M. Aubry, S. Paris, S. W. Hasinoff, J. Kautz, and F. Durand. Fast local Laplacian filters: Theory and applications. *ACM Transactions on Graphics*, 33(5), 2014. 2, 5, 6

[6] J. Aujol, G. Gilboa, T. F. Chan, and S. Osher. Structure-texture image decomposition – modeling, algorithms, and parameter selection. *IJCV*, 67(1), 2006. 2

[7] S. Bae, S. Paris, and F. Durand. Two-scale tone management for photographic look. *ACM Transactions on Graphics*, 25(3), 2006. 2

[8] J. T. Barron and B. Poole. The fast bilateral solver. In *ECCV*, 2016. 2

[9] D. Berman, T. Treibitz, and S. Avidan. Non-local image dehazing. In *CVPR*, 2016. 2, 5, 6

[10] J. Bruna, P. Sprechmann, and Y. LeCun. Super-resolution with deep convolutional sufficient statistics. In *ICLR*, 2016. 3, 5

[11] H. C. Burger, C. J. Schuler, and S. Harmeling. Image denoising: Can plain neural networks compete with BM3D? In *CVPR*, 2012. 3

[12] V. Bychkovsky, S. Paris, E. Chan, and F. Durand. Learning photographic global tonal adjustment with a database of input / output image pairs. In *CVPR*, 2011. 5

[13] A. Chambolle and T. Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 40, 2011. 2

[14] J. Chen, A. Adams, N. Wadhwa, and S. W. Hasinoff. Bilateral guided upsampling. *ACM Transactions on Graphics*, 35(6), 2016. 2, 5, 6, 7

[15] J. Chen, S. Paris, and F. Durand. Real-time edge-aware image processing with the bilateral grid. *ACM Transactions on Graphics*, 26(3), 2007. 2

[16] Q. Chen and V. Koltun. Photographic image synthesis with cascaded refinement networks. In *ICCV*, 2017. 5

[17] Y. Chen, W. Yu, and T. Pock. On learning optimized reaction diffusion processes for effective image restoration. In *CVPR*, 2015. 2

[18] D. Dang-Nguyen, C. Pasquini, V. Conotter, and G. Boato. RAISE: A raw images dataset for digital image forensics. In *Proc. ACM Multimedia Systems Conference*, 2015. 5

[19] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *PAMI*, 38(2), 2016. 3

[20] A. Dosovitskiy and T. Brox. Generating images with perceptual similarity metrics based on deep networks. In *NIPS*, 2016. 5

[21] F. Durand and J. Dorsey. Fast bilateral filtering for the display of high-dynamic-range images. *ACM Transactions on Graphics*, 21(3), 2002. 2

[22] D. Eigen, D. Krishnan, and R. Fergus. Restoring an image taken through a window covered with dirt or rain. In *ICCV*, 2013. 3

[23] Z. Farbman, R. Fattal, and D. Lischinski. Convolution pyramids. *ACM Transactions on Graphics*, 30(6), 2011. 2

[24] Z. Farbman, R. Fattal, D. Lischinski, and R. Szeliski. Edge-preserving decompositions for multi-scale tone and detail manipulation. *ACM Transactions on Graphics*, 27(3), 2008. 2, 5, 6, 8

[25] R. Fattal. Single image dehazing. *ACM Transactions on Graphics*, 27(3), 2008. 2

[26] R. Fattal. Edge-avoiding wavelets and their applications. *ACM Transactions on Graphics*, 28(3), 2009. 2

[27] R. Fattal. Dehazing using color-lines. *ACM Transactions on Graphics*, 34(1), 2014. 2

[28] E. S. L. Gastal and M. M. Oliveira. Domain transform for edge-aware image and video processing. *ACM Transactions on Graphics*, 30(4), 2011. 2

[29] E. S. L. Gastal and M. M. Oliveira. Adaptive manifolds for real-time high-dimensional filtering. *ACM Transactions on Graphics*, 31(4), 2012. 2

[30] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *CVPR*, 2016. 3

[31] M. Gharbi, G. Chaurasia, S. Paris, and F. Durand. Deep joint demosaicking and denoising. *ACM Transactions on Graphics*, 35(6), 2016. 3

[32] M. Gharbi, Y. Shih, G. Chaurasia, J. Ragan-Kelley, S. Paris, and F. Durand. Transform recipes for efficient cloud photo enhancement. *ACM Transactions on Graphics*, 34(6), 2015. 3

[33] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. In *ICLR*, 2016. 3

[34] K. He and J. Sun. Fast guided filter. *arXiv:1505.00996*, 2015. 2

[35] K. He, J. Sun, and X. Tang. Single image haze removal using dark channel prior. *PAMI*, 33(12), 2011. 2, 5, 6

[36] J. Hegarty, R. Daly, Z. DeVito, M. Horowitz, P. Hanrahan, and J. Ragan-Kelley. Rigel: Flexible multi-rate image processing hardware. *ACM Transactions on Graphics*, 35(4), 2016. 3

[37] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 4

[38] P. Isola, J. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017. 5, 6

[39] V. Jain and H. S. Seung. Natural image denoising with convolutional networks. In *NIPS*, 2008. 3

[40] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016. 3, 5, 6, 7

[41] J. Kim, J. K. Lee, and K. M. Lee. Accurate image super-resolution using very deep convolutional networks. In *CVPR*, 2016. 3

[42] J. Kim, J. K. Lee, and K. M. Lee. Deeply-recursive convolutional network for image super-resolution. In *CVPR*, 2016. 3

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

[44] A. Knapitsch, J. Park, Q. Zhou, and V. Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics*, 36(4), 2017. 8

[45] J. Kopf, M. F. Cohen, D. Lischinski, and M. Uyttendaele. Joint bilateral upsampling. *ACM Transactions on Graphics*, 26(3), 2007. 2

[46] D. Krishnan, R. Fattal, and R. Szeliski. Efficient preconditioning of Laplacian matrices for computer graphics. *ACM Transactions on Graphics*, 32(4), 2013. 2

[47] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4), 1989. 4

[48] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. P. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, 2017. 3, 5

[49] Y. Li, J. Huang, N. Ahuja, and M. Yang. Deep joint image filtering. In *ECCV*, 2016. 3

[50] R. Liao, X. Tao, R. Li, Z. Ma, and J. Jia. Video super-resolution via deep draft-ensemble learning. In *ICCV*, 2015. 3

[51] S. Liu, J. Pan, and M. Yang. Learning recursive filters for low-level vision via a hybrid neural network. In *ECCV*, 2016. 3, 5, 6, 7

[52] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 4, 7

[53] C. Lu, L. Xu, and J. Jia. Combining sketch and tone for pencil drawing production. In *Non-Photorealistic Animation and Rendering*, 2012. 5, 6

[54] Z. Ma, K. He, Y. Wei, J. Sun, and E. Wu. Constant time weighted median filtering for stereo matching and beyond. In *ICCV*, 2013. 2

[55] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *ICML*, 2013. 4

[56] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein. Unrolled generative adversarial networks. In *ICLR*, 2017. 5

[57] P. Milanfar. A tour of modern image filtering: New insights and methods, both practical and theoretical. *IEEE Signal Processing Magazine*, 30(1), 2013. 2

[58] M. Nikolova. A variational approach to remove outliers and impulse noise. *Journal of Mathematical Imaging and Vision*, 20, 2004. 5, 6, 8

[59] S. Paris and F. Durand. A fast approximation of the bilateral filter using a signal processing approach. *IJCV*, 81(1), 2009. 2

[60] S. Paris, S. W. Hasinoff, and J. Kautz. Local Laplacian filters: Edge-aware image processing with a Laplacian pyramid. *ACM Transactions on Graphics*, 30(4), 2011. 2, 5, 6

[61] S. Perreault and P. Hébert. Median filtering in constant time. *IEEE Transactions on Image Processing*, 16(9), 2007. 2

[62] T. Pock, M. Unger, D. Cremers, and H. Bischof. Fast and exact solution of total variation models on the GPU. In *CVPR Workshops*, 2008. 2

[63] J. Ragan-Kelley, A. Adams, S. Paris, M. Levoy, S. P. Amarasinghe, and F. Durand. Decoupling algorithms from schedules for easy optimization of image processing pipelines. *ACM Transactions on Graphics*, 31(4), 2012. 3, 5

[64] W. Ren, S. Liu, H. Zhang, J. Pan, X. Cao, and M. Yang. Single image dehazing via multi-scale convolutional neural networks. In *ECCV*, 2016. 3

[65] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 7

[66] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D*, 60(1), 1992. 5, 6, 8

[67] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323, 1986. 4

[68] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 7

[69] K. Subr, C. Soler, and F. Durand. Edge-preserving multiscale image decomposition based on local extrema. *ACM Transactions on Graphics*, 28(5), 2009. 2

[70] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. S. Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *ICML*, 2016. 3

[71] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4), 2004. 5, 6

[72] B. Weiss. Fast median and bilateral filtering. *ACM Transactions on Graphics*, 25(3), 2006. 2

[73] L. Xu, C. Lu, Y. Xu, and J. Jia. Image smoothing via L_0 gradient minimization. *ACM Transactions on Graphics*, 30(6), 2011. 2, 5, 6, 8

[74] L. Xu, J. S. J. Ren, C. Liu, and J. Jia. Deep convolutional neural network for image deconvolution. In *NIPS*, 2014. 3

[75] L. Xu, J. S. J. Ren, Q. Yan, R. Liao, and J. Jia. Deep edge-aware filters. In *ICML*, 2015. 3, 5, 6, 7

[76] L. Xu, Q. Yan, Y. Xia, and J. Jia. Structure extraction from texture via relative total variation. *ACM Transactions on Graphics*, 31(6), 2012. 2, 5, 6

[77] Z. Yan, H. Zhang, B. Wang, S. Paris, and Y. Yu. Automatic photo adjustment using deep neural networks. *ACM Transactions on Graphics*, 35(2), 2016. 3

[78] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2016. 4

[79] Q. Zhang, X. Shen, L. Xu, and J. Jia. Rolling guidance filter. In *ECCV*, 2014. 2

[80] Q. Zhang, L. Xu, and J. Jia. 100+ times faster weighted median filter (WMF). In *CVPR*, 2014. 2