Learning Recursive Filters for Low-Level Vision via a Hybrid Neural Network

Sifei Liu[1] Jinshan Pan[1,2] Ming-Hsuan Yang[1]

[1]University of California at Merced [2]Dalian University of Technology
sliu32@ucmerced.edu mhyang@ucmerced.edu sdluran@gmail.com

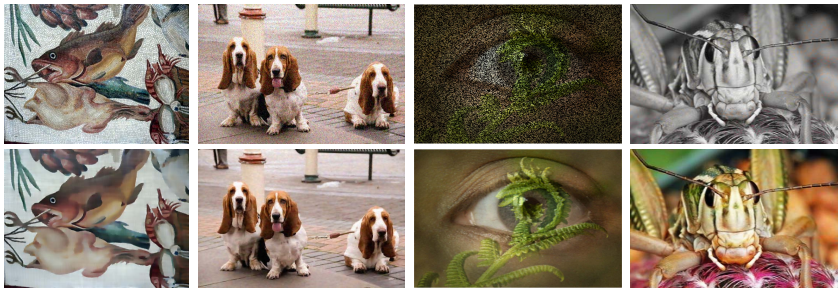
Abstract. In this paper, we consider numerous low-level vision problems (e.g., edge-preserving filtering and denoising) as recursive image filtering via a hybrid neural network. The network contains several spatially variant recurrent neural networks (RNN) as equivalents of a group of distinct recursive filters for each pixel, and a deep convolutional neural network (CNN) that learns the weights of RNNs. The deep CNN can learn regulations of recurrent propagation for various tasks and effectively guides recurrent propagation over an entire image. The proposed model does not need a large number of convolutional channels nor big kernels to learn features for low-level vision filters. It is significantly smaller and faster in comparison with a deep CNN based image filter. Experimental results show that many low-level vision tasks can be effectively learned and carried out in real-time by the proposed algorithm.

1 Introduction

Recursive filters, also called Infinite Impulse Response (IIR) filters, are efficient algorithms that account for signals with infinite duration. As such, recursive implementations are commonly exploited to accelerate image filtering methods, such as spatially invariant/variant Gaussian filters [1,2,3], bilateral filters [4] and domain transforms [5]. However, few methods are developed based on recursive formulations for low-level vision tasks mainly due to the difficulty in filter design.

Recently, several deep CNN based methods have been proposed for low-level vision tasks [6,7,8,9,10]. A convolutional filter can be considered equivalent to a finite impulse response (FIR) filter. Unlike IIR filters, it is easier to design FIR filters at the expense of using more parameters to support non-local dependency. In deep CNNs, Xu et al. [7] approximate a number of edge-preserving filters using a data-driven approach which can utilize hundreds of convolutional channels to support spatially variant filtering or large (up to 16×16) kernels to support global convolution. In spite of using a large number of parameters, this model does not present local image structures well. Furthermore, it is difficult to extend the deep CNN model to other low-level vision problems such as colorization and image completion.

Fig. 1 shows a number of low-level vision tasks, e.g., denoising and inpainting, which can be efficiently carried out by the proposed algorithm. In this work,



(a) smoothing (b) denoising (c) inpainting (d) color interpolation

Fig. 1. Several applications of the proposed algorithm. (a) Approximation of relative total variation (RTV) [11] for edge-preserving smoothing. (b) Denoising. (c) Restoration of an image with random 50% pixels occluded. (d) Restoration of an image with only 3% color informations retained.

we incorporate a group of RNNs as an equivalent of a recursive filter. As an important class of neural networks, RNNs have been used for modeling contextual information in sequential data [12,13,14]. The linear formulation of a RNN is equivalent to a first order recursive filter, and the weight matrix corresponds to the coefficients. In addition, higher order recursive filters can be formulated with several RNNs integrated either in cascade, or in parallel. To design a data-driven RNN filter, a straightforward approach is to take each pixel as a hidden recurrent node in a two-dimensional (2D) spatial sequence [15,16,17], and use the recurrent structure to learn the propagation weight matrix. However, a standard RNN uses an invariant weight matrix, which makes all pixels share one single recursive filter. Thus, this approach cannot be directly applied to filters that are conditioned on an input image with spatially variant structures, e.g. edge-preserving smoothing.

To address these issues, we propose a spatially variant RNN by introducing a weight map conditioned on the input image. The map has a set of distinct values for each node which control the node-wise recurrent propagation, or equivalently, each node has a distinct recursive filter. The weight map is associated with an image representation that reveals important structures e.g., salient edges (useful for edge-preserving smoothing and denoising). It can be jointly trained through a deep CNN that is combined with RNNs in an end-to-end fashion. The proposed hybrid network is shown in Fig. 3, which exhibits significant differences from existing pure data-driven CNN models [6,7,8,9,10]. It is worth emphasizing that the CNN is not used to extract hierarchical image features, but to learn the coefficients of RNNs. We show that a variety of low-level vision tasks can be carried out as recursive image filtering by the proposed neural network.

The contributions of this work are summarized as: (a) A hybrid neural network is proposed to learn recursive filters for low-level vision tasks. The network contains several spatially variant RNNs as equivalents of a group of distinct recursive filters for each pixel, and a deep CNN that learns the weights of the RNNs. (b) The deep CNN effectively guides the propagation of RNNs through learned regulations in a data-driven fashion. Specifically, the weight map from

the CNN is highly correlated to the corresponding image structures, which plays an important role in low-level vision problems. (c) The proposed model achieves promising results without any special design, regularization of the coefficients, pre-training or post-processing, and is suitable for real-time applications.

2 Related Work

Low-Level Vision. The recent years have witnessed significant advances in numerous low-level vision problems due to the use of designed priors and propagation methods under the guidance of image structures. In edge-preserving image smoothing, the key problem is to design structural priors to preserve sharp edges. Some explicit weight-averaging filters, e.g., bilateral filters [18] and guided image filters [19] exploit internal or guided image structures to preserve the edges of filtered images. Most energy-based edge-preserving methods explicitly or implicitly design adaptive weight maps through image structures (e.g., image gradients), such as edge-preserving decompositions [20], relative total variation [11], to achieve this goal. In PDE-based image processing, the edge-preserving effect is achieved by hand-craft anisotropic diffusion operators [21]. These adaptive weight maps control whether the image regions should be smoothed or not. Similar ideas have been used in image denoising and inpainting.

Numerous recent image processing methods, e.g., colorization [22,23] and image matting [24,23], involve propagation that is equivalent to implicitly filtering an image according to its structure. Although significant progress has been made, solving any of these problems is not a trivial task as specific operations are required. Furthermore, it is difficult to solve them in a unified framework.

Deep Learning Models. Several data-driven deep learning methods for low-level vision have been explored in recent years [6,7,8,9,10]. One significant advantage is that these data-driven models are good approximations to multiple conventional filters/enhancers via one learning paradigm. The uniform edge-preserving CNN filter [7] is able to achieve 200 times acceleration against some conventional methods. In addition, CNNs have been applied to image denoising [25,26,27], super resolution [28], and deconvolution [6], among others. However, there are two factors that limit the performance of deep CNN based models. First, these models are generally large due to numerous convolutional operations. Second, it is difficult to generalize CNN based models to a variety of low-level vision problems. As another class of neural networks, RNNs have been recently exploited for high-level vision problems such as object recognition [14] and scene labeling [13], through applying recurrent propagations over the spatial domain. In this work, we show that recurrent structures can be better exploited for effective and efficient image filtering for low-level vision.

3 Recursive Filter via RNNs

The proposed model contains two parts: a deep CNN, and a set of RNNs that take the output of the CNN as their input. Different from existing CNN based

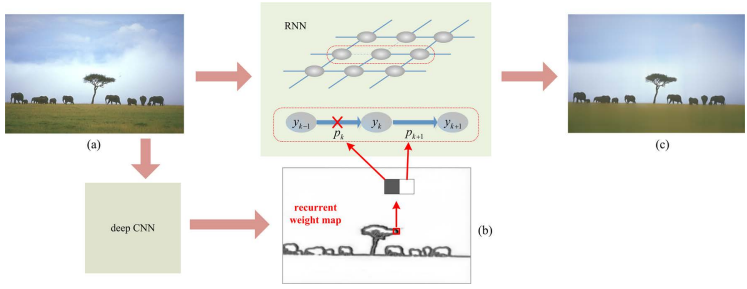


Fig. 2. An illustrative example of the proposed model for edge-preserving image smoothing with a single RNN. The deep CNN generates a weight map (b) that guides the propagation of the RNN. We consider an image as a group of sequences, and take the left-to-right recurrent propagation in 1D as an example, where k denotes a spatial location. For a single RNN, the weight map corresponds to the edges of an image and can be clearly visualized. When p_k is close to zero, it cuts off the propagations from $k-1$ to k so that the edge is preserved (i.e., near boundary). On the other hand, p_{k+1} maintains the propagation from k to $k+1$ so that the image is smoothed at any non-edge location. The CNN and RNN are jointly trained and the proposed network can be generalized to many other applications such as colorization, inpainting and denoising (see Fig. 1).

methods [6,7,8], the filtered images are generated only through the set of RNNs. The deep CNN, on the other hand, does not contribute any features or outputs for the filtered result. Instead, it learns the internal regulations (see Fig. 2, an example of a single RNN for edge-preserving smoothing) to guide the propagation process for each hidden node. In terms of the network structure, the deep CNN does not need to have a large number of channels or large kernels, since it focuses on learning the guidance for recurrent propagation instead of kernels for low-level filters. In comparison to recent deep CNN models for [28,6,7], the proposed model is much more efficient and light-weighted.

In this section, we describe the algorithmic details of the low-level part in the proposed network. We show that a recursive filter can be equally expressed by a set of RNNs, with its coefficients corresponding to the weight matrices of RNNs. We present two schemes to combine a group of RNNs for constructing a recursive filter, and show how to ensure the stability of the system.

3.1 Preliminaries of Recursive Filters

We first review recursive IIR filters [29] before presenting the hybrid neural network. For illustration, we use a one-dimensional (1D) convolution FIR filter, in which the output $y[k]$ is composed of a weighed sum of the input signal $x[k-i]$, expressed in the causal, discrete-time formulation:

$$y[k] = \sum_{i=0}^{M} a_i x[k-i], \quad k = 0, \ldots, N, \tag{1}$$

where N is the range of the sequence to be filtered, k is one point in the signal which practically corresponds to a frame, character, or pixel in the sequential

data. A 1D IIR filter is different in the sense that the output also contains the previously computed values:

$$y\left[k\right]=\sum_{i=0}^{P}a_{i}x\left[k-i\right]+\sum_{j=1}^{Q}b_{j}y\left[k-j\right],\ k=0,\ldots,N, \tag{2}$$

where $x\left[k-i\right]$ is the input and $y\left[k\right]$ is the output sequence, $\{a_i, b_i\} \in \mathbb{R}$ are filter coefficients, P and M are the order of convolutional filters, and Q is the order of the recursive filter. A 0-th order IIR filter is reduced to a FIR filter. An IIR filter (2) is equivalent to a FIR filter (1) by recursively expanding its second term. For an impulse input, the expanded terms can be infinitely long with exponentially decaying coefficients. That is, an IIR filter bypasses a long convolution, with only a few coefficients involved. The causal IIR system from (2) is equivalently described in the z-domain by its transfer function $H(z)$ [29]:

$$H\left(z\right)=\frac{\sum_{i=0}^{P}a_{i}z^{-i}}{1-\sum_{j=1}^{Q}b_{j}z^{-j}}. \tag{3}$$

It describes the frequency properties of IIRs independent of specific input signals. The output sequence $y\left[k\right]$ can be obtained from the z-transform of the input signal $X\left(z\right)$ and $H\left(z\right)$ by computing the inverse z-transform of $H\left(z\right)X\left(z\right)$. Note that for causal filters, we need to define the initial conditions of the input signal $x\left[-i\right]$ where $i=1,...,P$, and the output signal $y\left[-j\right]$ where $j=1,...,Q$. In this work, we set the initial conditions to zero in the training process since we only use up to the second order ($Q \leq 2$). Similarly, we obtain the testing results by padding image borders.

3.2 Recursive Decomposition

The Q-th order IIR filter can be decomposited into a set of first order filters in two different forms.

Cascade Decomposition. A recursive filter can be described in the z-plane with poles and zeros [29]. Denoting the poles by $\{p_j\}_{j=1}^{Q}$ and the nonzero zeros by $\{q_i\}_{i=1}^{P}$, we have

$$\begin{aligned}&H\left(z\right)=H_{r}\left(z\right)H_{c}\left(z\right),\\&H_{r}=\textstyle\prod_{j=1}^{Q}\frac{g_{j}}{1-p_{j}z^{-1}},\ H_{c}=\textstyle\prod_{i=1}^{P}h_{i}(1-q_{i}z^{-1}),\end{aligned} \tag{4}$$

where H_r and H_c are recursive and convolutional parts, g_i and h_j are their coefficients respectively, $\{g, h, p, q\} \in \mathbb{C}$. While H_c is equivalent to an ordinary 0-th order FIR that can be constructed through a convolutional layer, H_r is a cascade of Q first order IIR units. The spatial domain formulation with respect to the j-th unit from sequences of input $x^r\left[k\right]$ and output $y^r\left[k\right]$ is:

$$y_{j}^{r}\left[k\right]=g_{j}x_{j}^{r}\left[k\right]+p_{j}y_{j}^{r}\left[k-1\right]. \tag{5}$$

We denote this formulation as a cascade decomposition.

Parallel Decomposition. In [30], it is shown that $H(z)$ can be decomposed into a sum of Q first order recursive filters:

$$\begin{aligned} &H(z) = H_r(z) + H_c(z), \\ &H_r = \textstyle\sum_{j=1}^{Q} \frac{g_j}{1-p_j z^{-1}},\ H_c = \textstyle\sum_{i=0}^{P-Q} h_i z^{-i}, \end{aligned} \tag{6}$$

where $\{g, h, p\} \in \mathbb{C}$. Similar to the cascade formulation, the parallel decomposition also contains a FIR H_c with different kernel size $(P - Q + 1)$ of a convolutional layer, as well as Q summed first order IIR units. Each one shares the same formulation as in (5). We refer to this formulation as a parallel decomposition.

To simplify the framework, we do not apply H_c from (4) and (6) in this work. Therefore, the parallel way has $P = Q - 1$, which is greater than the cascade one with $P = 0$ when $Q > 1$. It is more amenable to be designed as a high-pass filter (e.g., for enhancement effect) compared to the cascade connection [29].

3.3 Constructing Recursive Filter via Linear RNNs

Single Linear RNN is 1st Order Filter. RNNs have been used to learn sequential data of varying length for various tasks. Let $x \in X$ be the input signal, $h \in H$ be the hidden state, and $\{W_x, W_h\}$ be the weight matrices, then the recurrent relation over spatial or time is modeled by

$$h[k] = f\{W_x x[k] + W_h(h[k-1] + b)\}. \tag{7}$$

The formulation (7) is slightly different from the first order recursive filter, as expressed in (5), where the sigmoid is often used for f to ensure the output is bounded and the recurrent system is stable in transition.

To model the recursive filter (5), we set f as an identity function $f(x) = x$, and $\{W_x, W_h\}$ as diagonal matrixes. We refer to this neural network as the Linear Recurrent Neural Network (LRNN) in this paper. With this method, we ignore the bias term in (7) and formulate LRNN using the dot product:

$$h[k] = g \cdot x[k] + p \cdot h[k-1], \tag{8}$$

where $x[k] \in \mathbb{R}^{n\times 1}$. The $\{g, p\} \in \mathbb{R}^{n\times 1}$ can be regarded as the diagonal values of W_x and W_h, where $\cdot$ is a dot product operator.

We further formulate (8) in a normalized filter, which has unit gain at some specified frequency. For example, a low-pass filter commonly has unit gain at $z = 1$, which implies that its discrete impulse response should sum to one. Normalizing a filter is carried out by scaling its impulse response by an appropriate factor, where (8) is computed by setting $g = 1 - p$ such that the prediction of coefficients is reduced to estimating the parameter p only:

$$h[k] = (1-p) \cdot x[k] + p \cdot h[k-1]. \tag{9}$$

Its backward pass can be generalized by back propagation thorough time (BPTT) used in RNNs [31]. The derivations with respect to $h[k]$, denoted as $\theta[k]$ is,

$$\theta[k] = \delta[k] + p \cdot \theta[k+1]. \tag{10}$$

The stability of LRNN (9) is different from the standard RNN (7) because the range of $h[k]$ is not controlled through some nonlinear functions (e.g., sigmoid). The output sequence is likely to go to infinity when p is greater than one. According to z-transform [29], the causal recursive system can be stabilized by regularizing p inside the unit circle $|p| < 1$, which we discuss in the next section. In addition, the propagation of (9) can reach to a long range when p is close to one, thereby enabling global propagation over an entire image.

Construction of High Order Filters. High order recursive filters [30] can be constructed by combining a group of LRNNs in cascade or parallel schemes as discussed in Section 3.2. In the cascade decomposition, LRNNs are stacked with the input signal passing through one to the next. In the parallel approach, each LRNN receives the input signal respectively, where the outputs are integrated with node-wise operations. The FIR terms (which we do not use in this work) can be implemented by convolutional layers that are integrated in the same way.

Two Dimensional Image. To filter an image we need to extend the 1D LRNN in (9) to 2D. We adopt a strategy similar to the 4-way directional propagation for two-dimensional data in [32]. First, the 1D LRNN is processed respectively along left-to-right, top-to-bottom and their reverse directions, as shown in Fig. 3. In any direction, we treat each row or column as 1D sequence. Taking the left-to-right case as an example, the LRNN scans each row from left to right. As a result, four hidden activation maps are generated. We integrate the four maps through selecting the optimal direction based on the maximum response at each location. This is carried out by a node-wise max pooling, which effectively selects the maximally responded direction as the desired information to be propagated and rejects noisy information from other directions. We note that the four directions can be executed in parallel for acceleration as they are independent.

4 Learning Spatially Variant Recursive Filters

One problem with the standard or linear RNN in (7) and (9) is that it takes a group of fixed weights for every point k. Filtering an image in such a way means that each pixel is processed with the same recursive filter, which is not effective for many low-level tasks, e.g., edge-preserving smoothing, where the edge and texture areas need to be processed differently.

4.1 Spatially Variant LRNN

Therefore, we propose a spatially variant recurrent network by extending the fixed parameter p to $p[k]$, so that each pixel has a distinct recursive filter. Taking edge-preserving smoothing as an example (see Fig. 2), and considering the first order recursive filter (a single LRNN), $\{p[k]\}$, namely the weight map, is supposed to be associated with an "edge map". Specifically, the weights that lie on the edge regions should be close to zero such that the input $x[k]$ is preserved,

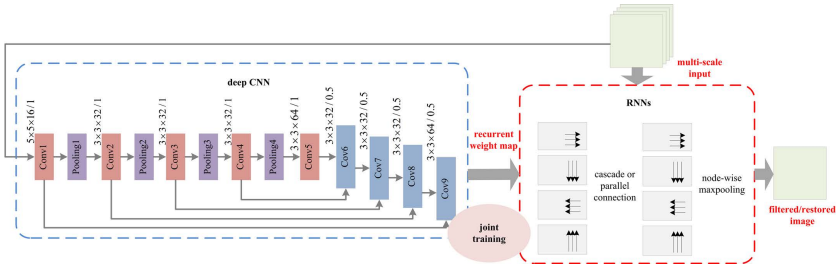


Fig. 3. Proposed hybrid network that contains a group of RNNs to filter/restore an image and a deep CNN to learn to propagate the RNNs. The process of filtering/restoration is carried out through RNNs with two inputs and one output result, denoted in red. Both parts are trained jointly in an end-to-end fashion.

and one otherwise so that the other regions can be smoothed out via recurrent propagation (as in (9)). For higher order recursive filters and some other tasks, e.g., inpainting, the weight maps are more complex and do not correspond to some explicit image structures. However, they reveal the propagation regulations with respect to specific tasks, which are conditioned on the input image.

We have two types of input to a LRNN, i.e., an image X and a weight map P. Given a hidden node $h\,[k]$ and similar to (9), the spatially variant LRNN is:

$$h\,[k] = (1 - p\,[k]) \cdot x\,[k] + p\,[k] \cdot h\,[k-1]\,. \tag{11}$$

In the back propagation pass, the derivative $\sigma\,[k]$ with respect to $p\,[k]$ is:

$$\sigma\,[k] = \theta\,[k] \cdot (h\,[k-1] - x\,[k])\,, \tag{12}$$

such that the weight map $p\,[k]$ of a spatially variant recursive filter can be learned.

4.2 Learning LRNN Weight Maps via CNN

We propose to learn the weight maps through a deep CNN, which takes an image to be filtered as its input. The CNN can be small and deep, since it learns the guidance of propagation instead of learning convolutional filters. The proposed network is equipped with 10 convolutional layers. The first five layers are followed by a max pooling, while the other five ones are followed by a bilinear upsampling. The RELUs are used between adjacent convolutional layers. In addition, 4 links between corresponding downsampling and upsampling units connect feature maps of the same size at different levels in order to learn better representations, where similar settings can be found in [33]. We use 3×3 kernels with the number of channels ranging from 16 to 64, as shown in Fig. 3.

To connect with the LRNNs of different directions (4 distinct hidden layers, see Fig. 3), the weight map can be equally split into 4 parts for the 4 directions. To simplify the network implementation, each axis is allowed to share the same part (e.g., the left-to-right and right-to-left directions share a common horizontal map). Thus, for each LRNN, we have two parts in a weight map for the x

and y-axis, respectively. We find that better results can be obtained by linearly transferring the RGB input of LRNN into a feature space, e.g., through one convolutional layer, and then perform LRNN on the proposed transform space. We are then able to select a best direction at each point on the feature space using a node-wise integration strategy, which combines the fore directions. The combined maps can be transferred back to a 3-channel image through another convolutional layer. We configure both of the transform convolutional layers using 3×3 kernels. We set the number of channels in each hidden layer of LRNNs to $m = 16$ in all experiments so that each $x[k]$ and $p[k]$ in (11) are vectors with dimension of 16. The number of output channels for CNN is $2 \times m \times R$, where R denotes the order of recursive filter (or equivalently the number of LRNNs), e.g., it should be set to 64 with a network configured with a $2nd$ order recursive filter. It is important that we equip a hyperbolic tangent function as the topmost layer of the CNN, so that the weight map is restricted to $(-1, 1)$ to stabilize the LRNN, as introduced in Section 3.3.

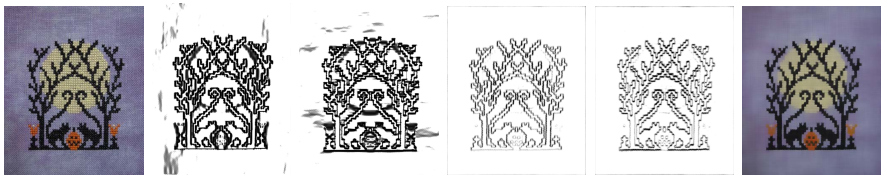
5 Experimental Results

We apply the proposed model to a variety of low-level vision problems including edge-preserving smoothing, enhancement, image denoising, inpainting and colorization. All the following applications share the same model size as well as the run-time. Specifically, our model reaches real-time performance on images of 320×240 pixels (QVGA) using a Nvidia Geforce GTX Ti GPU with 3 GB memory. Due to space limitations, we present some results in this section. More and large images are included in the supplemental materials. The trained models and source code will be made available at www.sifeiliu.net/project.

Experimental Settings. To obtain rich information from different scales of an image, we use multi-scale input through downsampling the color image with ratio of $\{1/2, 1/4, 1/8, 1/16\}$, resizing them to the original size, and concatenating them to be a single input. Therefore, nodes in a LRNN can reach to a more global range via processing on coarse scales, without increasing the number of coefficient maps to be learned. We use 96×96 image patches as the original inputs that are randomly cropped from training images, which are then processed as multi-scale input through average pooling and upsampling. All patches are augmented through perturbation using the similarity transform, so as to adapt to the scale-variant property for some existing filters. We use roughly $400,000$ image patches that are randomly cropped from the MS COCO dataset [34] in the training process with data augmentations. For all the following applications, the order of filter is set to 2 with specific structures shown in Fig. 3. The only difference lies in the integration manner with respect to these 2 LRNNs, e.g., in cascade or parallel way, which is specified in each application.

5.1 Edge-Preserving Smoothing

Xu et al. in [7] propose a CNN model to approximate various filters such that many conventional implementations can be accelerated significantly. We show



(a) original (b) RTV-x (c) RTV-y (d) weight-x (e) weight-y (f) our result

Fig. 4. Visualization of weight maps for approximating the RTV filter using first order recursive filter. (a) original image; (b) and (c): manually designed edge prior maps in RTV for x and y axes; (d) and (e): weight maps generated from the CNN for x and y; (f) our filtered result.

that the proposed algorithm is able to approximate various filters and performs favorably against [7] in terms of accuracy, run time, and model size. We selectively learn a group of local and global filters including bilateral filter (BLF) [18], weighted least square (WLS) [20], L0 smoothing [35], RTV texture smoothing [11], weighted median filter (WMF) [36], and rolling guidance filter (RGF) [37].

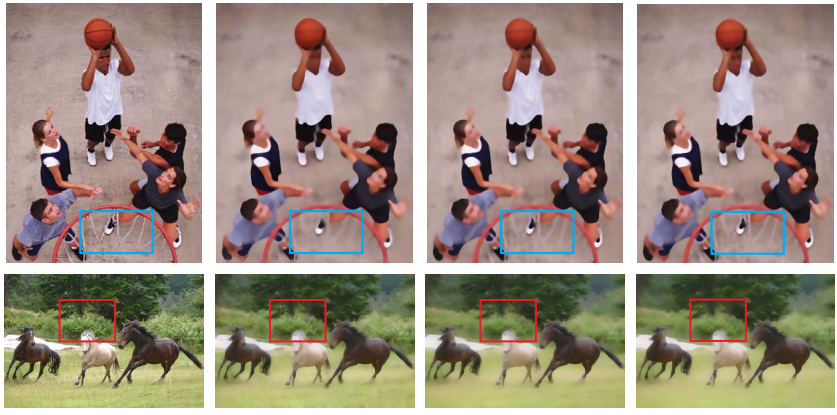
Visualization of Weight Maps. We first demonstrate through a first order recursive filter using a single scale RGB image without any linear transformation as the input to both CNN and LRNN, where the weight maps with respect to x and y axes accurately correspond to the edges of the image. This is carried out by setting the number of output channels of the CNN to 2, such that the maps for x and y axes, which are then shared by all channels of the hidden layers in the LRNN, can be obtained and visualized.

We note that some edge-preserving methods, e.g., RTV [11], focus on extracting the main structures of an image. The designed edge prior maps for RTV (Fig. 4(b) and (c)), which reflect the main structures of an image, determine whether the image regions should be smoothed or not in the propagation step [11]. Interestingly, the learned data-driven weight maps by our model (see Fig. 4(d) and (e)) have the similar effects to the hand-craft maps. They accurately locate the image edges with cleaner background, and effectively remove the grid-like texture in the input image, as shown in Fig. 4(f). As our method is data-driven, different weight maps can be generated for different tasks. The data-driven approach allows the proposed algorithm to be generalized to a variety of applications without hand-craft priors.

Quantitative Comparisons. We show the applications that are based on a second order filter. Specifically for edge-smoothing tasks (e.g., L0, WLS and RTV, etc.), the two LRNNs are connected in cascade since it is more amicable to low-pass filtering. On the other hand, we use the parallel integration scheme for learning shock filters [38] with enhancement effects. We quantitatively evaluate the proposed algorithm against [7] on the dataset used in [7]. Table 1 shows that our method generates high quality filtered images with significant improvements over the state-of-the-art CNN based method. In addition, the proposed model is much smaller and faster due to its hybrid structure, which can be used to accelerate more conventional algorithms, e.g., region covariance filter (RegCov) [39] and local laplacian filter (LLF) [40].

Table 1. Quantitative evaluations for learning various image filters.

Methods	L_0 [35]	BLF [18]	RTV [11]	RGF [37]	WLS [20]	WMF [36]	Shock filter [38]
PSNRs of [7]	32.8	38.4	32.1	35.9	36.2	31.6	30.0
Our PSNRs	30.9	**38.6**	**37.1**	**42.2**	**39.4**	**34.0**	**31.8**



(a) Input (b) Xu et al. [7] (c) Ours (d) Original filters

Fig. 5. Approximation of edge-preserving filters. (a) input images. (b) results by Xu et al. [7]. (c) results of our model. (d) results from the original filters. First row: Results by approximating RGF [37]. Second row: Results by approximating WLS smoothing [20].

Fig. 5 shows approximations of RGF [37] and WLS smoothing [20]. The results by our model preserve more accurate structures without including details that are supposed to be removed. The filtered images are visually the same as those generated by the original implementations. We note that the CNN based filter [7] misses important local structures by approximating the RGF, and includes some details that should be removed by approximating the WLS, as shown in Fig. 5(b). More results are included in the supplemental material.

Run Time and Model Size We evaluate all the following methods with the same computer introduced in the beginning of this section. The proposed method achieves favorable speed as shown in Table 2, and is significantly smaller than that of [7] (0.54 vs 5.60 MB). It can speed up a variety of conventional filters for denoising, inpainting and colorization, etc.

Table 2. Run-time (second) performance against [7] and some conventional methods at different resolutions of color images.

method	BLF [18]	WLS [20]	RTV [11]	WMF [36]	EPLL [41]	Levin [42]	Xu et al. [7]	**Ours**
QVGA	0.46	0.71	0.81	0.67	33.82	2.10	0.23	**0.05**
VGA	1.41	3.40	3.51	1.70	466.79	9.24	0.83	**0.16**
720p	3.18	11.38	9.94	3.80	1395.61	31.09	2.10	**0.37**

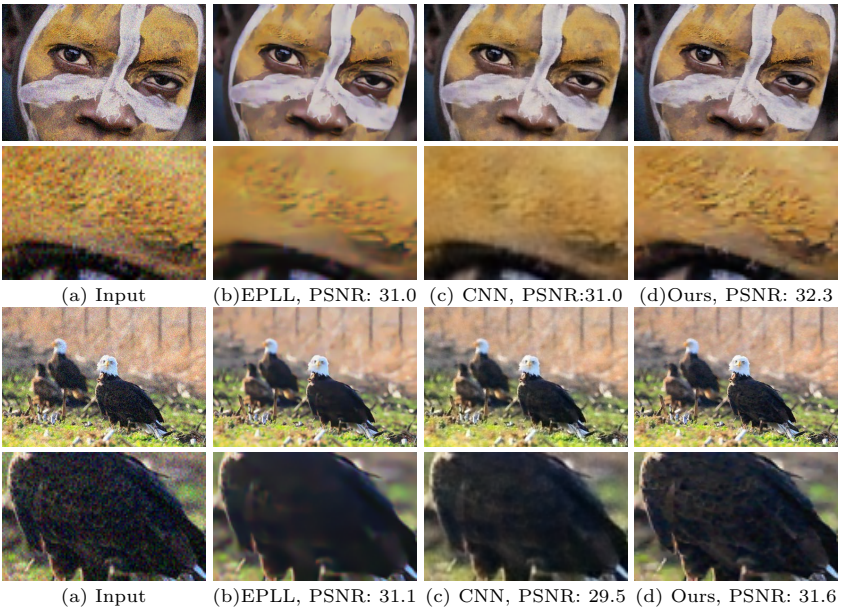


Fig. 6. Image denoising. (b) denotes the results of image patch prior based method EPLL [41]. (c) denotes the results by end-to-end trainable CNN method [9].

5.2 Image Denoising

The proposed method can be used to learn filters for image denoising. Specifically, we train the model with thousands of patches in which white Gaussian noise with the standard deviation of 0.01 is added. At the output end, the model is supervised by the original image patches. We apply the parallel connection to the two LRNNs to preserve more details. The other settings are the same as those used in Section 5.1.

Fig. 6 shows the results with two state-of-the-art algorithms including expected patch log likelihood (EPLL) [41] and a deep CNN based model [9]. The denoising method [41] is based on a prior of image patches, and the vectorization-based deep CNN [9] is based on a two-layer convolutional model. Although significant noise has been removed by both methods, some details are not preserved well and the restored results can be over-smoothed. The learned filter by the proposed model generates clear images with well preserved fine details, as shown in Fig. 6(d). It retains important image contents such as the brushstrokes of oil painting in the first row, or pattens of the feather in the second row.

The deep CNN method is likely to be slower in terms of run-time (was not specified in [9]) due to its large model size, while the EPLL takes more than hundreds of seconds to process one image. In contrast, the proposed method achieves several order of magnitude accelerations (see Section 5.1).

(a) occluded (b) restored (c) original

Fig. 7. Pixel interpolation. (a) input image. (b) restored image for masking half pixels in (a).

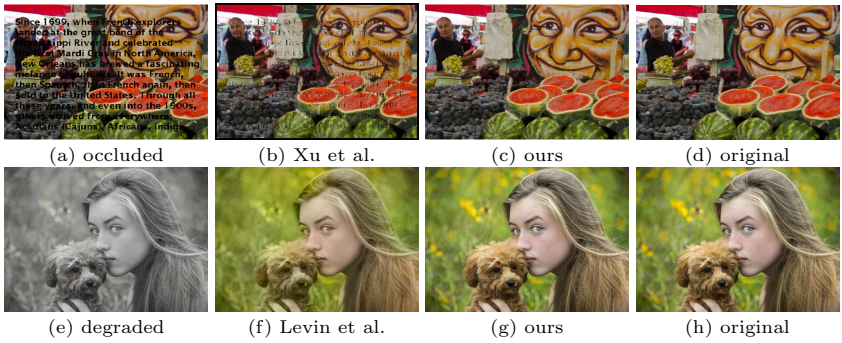
5.3 Image Propagation Examples

In this section, we validate the effectiveness of propagation-study of the network by restoring images from degraded frames with masks. The deep CNN here learns more complex rules than the edges that are used for smoothing. We apply the proposed model to two interesting applications for pixel and color interpolation (e.g., inpainting and colorizaiton). Specifically, we retain randomly 50% pixels for the image interpolation and 3% monochrome pixels for the color interpolation. The proposed model takes degraded images as well as masks as input channels, and learns the weight maps with the supervision of the original images. It learns complex regulations including identifying the occluded pixels and restoring them by propagating information from the other pixels, and identifying the image structures such that the restored pixels can naturally adapt to them.

Pixel Interpolation. The goal of pixel interpolation is to restore the values in missing regions according to a mask of pixels that are to be restored. In this model, the random mask is concatenated with the degraded image as the input, such that it learns the propagation rules according to all the visual information. The LRNNs filter the degraded image according to the learned rules and output an interpolated result. It does not require explicit regulations to compute the missing data, nor expensive optimizations for each test image. Therefore, it is accurate and fast to execute through forward propagation.

We show that the proposed algorithm can restore fine details (e.g., pattens on a butterfly) in Fig. 7 with randomly half pixels are masked. We discover that the proposed model trained for image interpolation can be directly applied to image inpainting with texts, as shown in the first row of Fig. 8. Both results are visually very similar to the original images, as shown in Fig. 7(b) and 8(c).

Color Interpolation. The proposed algorithm can be applied to color image restoration and editing despite providing little color information, e.g., user inputs. Given the brightness channels (y channel in the YCbCr color space), we retain only 3% color pixels, as shown in Fig. 8(e). Taking a degraded image and a mask as input, the proposed model learns to propagate the known colors to other regions to be restored. Specifically, the proposed model generates favorable results (visually the same with the original image) compared to the state-of-the-art method [42], which takes more than 3 seconds on a QVGA image.



(a) occluded (b) Xu et al. (c) ours (d) original

(e) degraded (f) Levin et al. (g) ours (h) original

Fig. 8. First row: image inpainting on the regions of texts with comparison to Xu et al. [8]. We directly apply the pixel interpolation model to inpainting. The model does not require any network finetuning on texts masks. Second row: color interpolation with comparison to Levin et al. [42].

(a) origin (b) reference (c) re-colored

Fig. 9. Re-colorization by applying the brightness channel of (a) and directly taking 3% color pixels from the monochrome channels in a reference image with the same size.

The proposed model can also be generalized to image re-colorization by applying the brightness channel of an input image, and directly taking 3% color pixels from the monochrome channels in a reference image of the same size. The re-colored image has the contents of the original image, but with the color style of the reference image. Fig. 9 shows one example of image re-colorization. More results are included in the supplemental material.

6 Conclusion

In this work, we propose a novel hybrid neural network for low-level vision tasks, based on the recursive filters whose coefficients can be learned by a deep CNN. We show that the proposed model is faster and significantly smaller than the deep CNN filters. It is also more generic, and can effectively and efficiently handle a variety of applications including image smoothing and enhancement, image denoising and pixel interpolation.

Acknowledgments. This work is supported by NSF CAREER grant 1149783, IIS grant 1152576, and gifts from Adobe as well as Nvidia. Preliminary work is carried out at Multimedia Laboratory in Chinese University of Hong Kong.

References

1. Deriche, R.: Recursively implementating the Gaussian and its derivatives. PhD thesis, INRIA (1993) 1
2. Young, I.T., Vliet, L.J.V.: Recursive implementation of the gaussian filter. Signal processing **44**(2) (1995) 139–151 1
3. Tan, S., Dale, J.L., Johnston, A.: Performance of three recursive algorithms for fast space-variant gaussian filtering. Real-Time Imaging **9**(3) (2003) 215–228 1
4. Yang, Q.: Recursive bilateral filtering. In: ECCV 1
5. Gastal, E.S., Oliveira, M.M.: Domain transform for edge-aware image and video processing. In: ACM TOG. Volume 30. (2011) 69 1
6. Xu, L., Ren, J.S., Liu, C., Jia, J.: Deep convolutional neural network for image deconvolution. In: NIPS. (2014) 1790–1798 1, 2, 3, 4
7. Xu, L., Ren, J.S., Yan, Q., Liao, R., Jia, J.: Deep edge-aware filters. In: ICML. (2015) 1669–1678 1, 2, 3, 4, 9, 10, 11
8. SJ, R.J., Li, X., Qiong, Y., Wenxiu, S.: Shepard convolutional neural networks. In: NIPS. (2015) 901–909 1, 2, 3, 4, 14
9. Ren, J.S.J., Xu, L.: On vectorization of deep convolutional neural networks for vision tasks. In: AAAI. (2015) 1840–1846 1, 2, 3, 12
10. Dong, C., Loy, C.C., He, K., Tang, X.: Learning a deep convolutional network for image super-resolution. In: ECCV. (2014) 184–199 1, 2, 3
11. Xu, L., Yan, Q., Xia, Y., Jia, J.: Structure extraction from texture via relative total variation. ACM TOG **31**(6) (2012) 139 2, 3, 10, 11
12. Graves, A., Fernndez, S., Schmidhuber, J.: Multi-dimensional recurrent neural networks. In: ICANN. (2007) 2
13. Byeon, W., Breuel, T.M., Raue, F., Liwicki, M.: Scene labeling with lstm recurrent neural networks. In: CVPR. (2015) 3547–3555 2, 3
14. Visin, F., Kastner, K., Cho, K., Matteucci, M., Courville, A., Bengio, Y.: Renet: A recurrent neural network based alternative to convolutional networks. arXiv preprint arXiv:1505.00393 (2015) 2, 3
15. Oord, A.V.D., Kalchbrenner, N., Kavukcuoglu, K.: Pixel recurrent neural networks. arXiv preprint arXiv:1601.06759 (2016) 2
16. Stollenga, M.F., Byeon, W., Liwicki, M., Schmidhuber, J.: Parallel multi-dimensional lstm, with application to fast biomedical volumetric image segmentation. arXiv preprint arXiv:1506.07452 (2015) 2
17. Kalchbrenner, N., Danihelka, I., Graves, A.: Grid long short-term memory. arXiv preprint arXiv:1507.01526 (2015) 2
18. Tomasi, C., Manduchi, R.: Bilateral filtering for gray and color images. In: ICCV. (1998) 839–846 3, 10, 11
19. He, K., Sun, J., Tang, X.: Guided image filtering. IEEE PAMI **35**(6) (2013) 1397–1409 3
20. Farbman, Z., Fattal, R., Lischinski, D., Szeliski, R.: Edge-preserving decompositions for multi-scale tone and detail manipulation. In: ACM TOG. Volume 27. (2008) 67 3, 10, 11
21. Weickert, J.: Anisotropic Diffusion in Image Processing. B.G. Teubner Stuttgart (1998) 3
22. Levin, A., Lischinski, D., Weiss, Y.: Colorization using optimization. ACM TOG **23**(3) (2004) 689–694 3
23. Xu, L., Yan, Q., Jia, J.: A sparse control model for image and video editing. ACM TOG **32**(6) (2013) 197:1–197:10 3

24. Levin, A., Lischinski, D., Weiss, Y.: A closed-form solution to natural image matting. IEEE PAMI **30**(2) (2008) 228–242 3
25. Burger, H.C., Schuler, C.J., Harmeling, S.: Image denoising: Can plain neural networks compete with bm3d? In: CVPR. (2012) 2392–2399 3
26. Xie, J., Xu, L., Chen, E.: Image denoising and inpainting with deep neural networks. In: NIPS. (2012) 341–349 3
27. Agostinelli, F., Anderson, M.R., Lee, H.: Adaptive multi-column deep neural networks with application to robust image denoising. In: NIPS. (2013) 1493–1501 3
28. Dong, C., Loy, C.C., He, K., Tang, X.: Learning a deep convolutional network for image super-resolution. In: ECCV. (2014) 184–199 3, 4
29. Proakis John, G., Manolakis Dimitris, G.: Digital signal processing, principles, algorithms, and applications. Pentice Hall (1996) 4, 5, 6, 7
30. Gastal, E.S., Oliveira, M.M.: High-order recursive filtering of non-uniformly sampled signals for image and video processing. In: Computer Graphics Forum. Volume 34. (2015) 81–93 6, 7
31. Graves, A., Schmidhuber, J.: Offline handwriting recognition with multidimensional recurrent neural networks. In: NIPS. (2009) 6
32. Chen, L., Barron, J.T., Papandreou, G., Murphy, K., Yuille, A.L.: Semantic image segmentation with task-specific edge detection using cnns and a discriminatively trained domain transform. arXiv preprint arXiv:1511.03328 (2015) 7
33. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: CVPR. (2015) 8
34. Lin, T., Maire, M., Belongie, S., Bourdev, L.D., Girshick, R.B., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: common objects in context. (2014) 9
35. Xu, L., Lu, C., Xu, Y., Jia, J.: Image smoothing via l-0 gradient minimization. In: ACM TOG. Volume 30. (2011) 174 10, 11
36. Zhang, Q., Xu, L., Jia, J.: 100+ times faster weighted median filter (wmf). In: CVPR. (2014) 2830–2837 10, 11
37. Zhang, Q., Shen, X., Xu, L., Jia, J.: Rolling guidance filter. In: ECCV. (2014) 815–830 10, 11
38. Osher, S., Rudin, L.I.: Feature-oriented image enhancement using shock filters. SIAM Journal on Numerical Analysis **27**(4) (1990) 919–940 10, 11
39. Karacan, L., Erdem, E., Erdem, A.: Structure-preserving image smoothing via region covariances. ACM TOG **32** (2013) 176 10
40. Paris, S., Hasinoff, S.W., Kautz, J.: Local laplacian filters: edge-aware image processing with a laplacian pyramid. ACM Trans. Graph. **30**(4) (2011) 68 10
41. Zoran, D., Weiss, Y.: From learning models of natural image patches to whole image restoration. In: ICCV. (2011) 479–486 11, 12
42. Levin, A., Lischinski, D., Weiss, Y.: Colorization using optimization. ACM TOG **23**(3) (2004) 689–694 11, 13, 14