ABSTRACT

In low light or short-exposure photography the image is often corrupted by noise. While longer exposure helps reduce the noise, it can produce blurry results due to the object and camera motion. The reconstruction of a noise-less image is an ill posed problem. Recent approaches for image denoising aim to predict kernels which are convolved with a set of successively taken images (burst) to obtain a clear image. We propose a deep neural network based approach called Multi-Kernel Prediction Networks (MKPN) for burst image denoising. MKPN predicts kernels of not just one size but of varying sizes and performs fusion of these different kernels resulting in one kernel per pixel. The advantages of our method are two fold: (a) the different sized kernels help in extracting different information from the image which results in better reconstruction and (b) kernel fusion assures retaining of the extracted information while maintaining computational efficiency. Experimental results reveal that MKPN outperforms state-of-the-art on our synthetic datasets with different noise levels.

Index Terms— Burst Image Denoising, Kernel Fusion, Deep Learning, Kernel Prediction Network.

1. INTRODUCTION

Image denoising is a long-standing problem finding applications in fields ranging from astronomical imaging to hand-held photography. With the development of digital photography and smartphone technology, it has recently become possible to take high-quality photos using relatively inexpensive equipment. However, there are still major differences between the imaging capabilities of hand-held devices such as smartphones and professional equipment such as DSLRs. One of the most important factors for taking noise-free photos is to collect as much light as possible. Professional cameras contain several hardware solutions such as large aperture lenses, sensors with large photosites and high-quality A/D converters for increased light collection [1]. However, for the sake of compactness, smartphones contain smaller and less expensive variants of these hardware elements which can result in noisy imaging, especially in low light conditions. To combat this problem, image denoising algorithms are implemented as a software solution in most smartphones [2].

Classical methods for image denoising developed in the early 1990s like anisotropic diffusion [3] and total variation denoising [4] use analytical priors and non-linear optimization to recover a clear image. More recently, plain neural networks [5], convolutional neural networks [6] and neural networks with auto-encoder architectures were used for single image denoising [7,8,9]. The combination of the last two, a convolutional auto-encoder, showed promising results in medical image denoising [10]. The same architecture was further improved by addition of skip connections [11] placed between the encoder and the decoder. This architecture was used both in single image denoising [12,13] as well as in denoising of the images captured in quick succession, i.e. burst image denoising [14].

Burst image denoising methods operate on a set of successive, rapidly taken images to compute a single, noise-free result. Since the noise is usually randomly distributed and a set of images obtained by the same camera have similar characteristics, it is reasonable to expect that burst image denoising tends to work better than single image denoising in most cases. Burst denoising methods commonly first align the successive images as a pre-processing step, and then fuse and denoise the aligned images [15,2,1,16]. Many of the state-of-the-art approaches in burst image denoising are based on fully convolutional neural network architectures [14,16,17].

Neural network based image denoising methods commonly operate on the whole image and predict the denoised image directly. Another approach is to design a network that can learn to predict spatially variant kernels for each pixel in the input. These per-pixel kernels are then convolved over the input to produce the final output. This approach worked very well in video frame interpolation [18], denoising of renderings [19,20] and burst image denoising [14].

In this paper we propose a novel neural network based method burst image denoising method. Our method is conceptually similar to [14] but it operates in a fundamentally different way. Specifically, instead of predicting a single, fixed-size kernel for each pixel, our method predicts kernels of multiple sizes for each pixel taking into account the spatially varying image structures. This allows our method to adapt to the properties of image structures with different properties (e.g., textured vs. non-textured, homogeneous areas) and successfully denoise a wide range of images without having to manually tweak the kernel size depending on the characteristics of the image.

2. PROPOSED METHOD

2.1. Multi-Kernel Prediction Network

Our goal is to perform denoising on burst images corrupted by noise due to low light or shot exposure photography. Given a noisy set of input burst images, kernels are predicted for each pixel using a deep neural network. These kernels are then convolved with their respective input pixels in the respective burst image which are then averaged to reconstruct the denoised image. [14] performed this denoising by predicting kernels of a predetermined size as given in [1]:

$$\hat{I}(x, y) = \frac{1}{N} \sum_{i=1}^{N} K_i(x, y) * P_i(x, y).$$  \hspace{1cm} (1)
by a linear combination of separable 1D kernels [22, 18]. In this
enhancements. In order to reduce the amount of computations, we apply two
sizes are used. However, convolving each kernel with its correspond-
characteristics. Ideally, MKPN will work well when many kernels of different
sizes are used. However, convolving each kernel with its correspond-
patch significantly increases the required amount of computa-
tions. In order to reduce the amount of computations, we apply two
enhancements:

Separable Kernel Estimation: We approximate the 2D kernels
by a linear combination of separable 1D kernels [22, 18]. In this
way the number of learnable parameters is reduced from \( n^2 \) to \( 2n \)
for each kernel of size \( n \times n \), significantly reducing the computation
cost.

Kernel Fusion: Instead of convolving each kernel separately
with each image in the burst, MKPN performs in-place kernel addi-
tion as described in (3), before the convolution operation:

\[
\hat{K}_i(x, y) = \frac{1}{|S|} \sum_{s \in S} K_i^s(x, y).
\]

The accumulated kernels \( \hat{K}_i(x, y) \) are then convolved with the
corresponding image patches in a single operation, as shown in (4):

\[
\hat{I}(x, y) = \frac{1}{N} \sum_{i=1}^{N} \hat{K}_i(x, y) * P_i(x, y).
\]

This essentially brings the number of convolution operations equal
up to that of [14]. The computational cost of the in-place additions
is negligible. Note that model compression [23, 24] and efficient
representations [25] can further reduce the computational costs.

2.2. Network Architecture

Fig. 1 shows an overview of the MKPN architecture. MKPN has
a typical U-Net [26] shape resembling the architectures of [14, 18].
The network consists of convolutional layers, ReLU activation func-
tions, average pooling layers and bilinear upsampling layers. The
convolutional layers use 3 x 3 filters with zero padding and stride
one, and are always followed by a ReLU activation function. The
average pooling layers have a pool size of 2 x 2 and stride two,
which in effect decreases the spatial resolution by a factor of two. On
the contrary, the bilinear upsampling layer increases the spatial
resolution by a factor of two.

A series of convolutional and average pooling layers encode the
input features into a latent representation. Series of bilinear upsam-
pling and convolutional layers decode these features to predict the
per pixel kernels. After each upsampling layer, the high resolution
features from the encoder side of the architecture are concatenated
to the decoder side.

The channel dimension of the layers of the last convolution
block is \( 2p \cdot N \), where \( p \) is the sum of the predefined sizes of differ-
ent kernels and \( N \) is the number of images in the input burst. For

\[
\tilde{I}(x, y) = \frac{1}{N} \sum_{i=1}^{N} I_i(x, y) + P_i(x, y)
\]

\[
\hat{I}(x, y) = \frac{1}{N} \sum_{i=1}^{N} \hat{K}_i(x, y) * P_i(x, y)
\]

This is the first attempt to utilize multiple kernels of different sizes
in a burst. The input of the network is a burst sequence of length \( N = 8 \), \( p \) is the sum of predefined kernel sizes and \( S \) is the set of these
kernels. Each of the burst images is deconvolved by the predicted per pixel kernels as seen in Fig. 2 and averaged to the final output. The
numbers below the blocks represent the 3-dimensional data structure at different levels of the architecture. The dimensions correspond to the
height x width x channel.

However, images consist of heterogeneous patches where every pixel
in a patch is different compared to its surrounding pixels. Flat re-
regions like sky in an image would provide for a more certain pre-
diction while densely cluttered leaves in an image would provide for
more uncertainties in the prediction. The predicted kernels should be
able accommodate to the different regions and pixels in the image.
Naturally, kernels of predetermined size cannot successfully adapt
to the diversity of pixels in the images.

Our method, Multi-Kernel Prediction Networks (MKPN) is a
direct extension of the [14] and [18]. To the best of our knowledge
this is the first attempt to utilize multiple kernels of different sizes
predicted by a neural network for image denoising. Although [21]
combine different kernels, they assume a fixed size of the kernel and
do not use a deep learning based approach. MKPN predicts kernels
of different sizes for each pixel in the image belonging to the set of
burst images instead of predicting fixed-size kernels. The predicted
kernels of different sizes are then convolved with each pixel in the
input image from the set of noisy burst images which are then aver-
eged to obtain the final reconstruction as described in [2].

\[
\tilde{I}(x, y) = \frac{1}{N \cdot |S|} \sum_{i=1}^{N} \sum_{s \in S} K_i(x, y) * P_i(x, y)
\]

Here \( N \) is the length of the input burst, \( S \) is the set of kernel sizes,
\( K_i(x, y) \) is a kernel with size \( s \) for pixel located at \( I_i(x, y) \) and
\( P_i(x, y) \) is a patch with size \( s \) in \( I_i \) centered at \( (x, y) \). \( I_i \) is the \( i \)-th
image from the input burst. Kernels of different sizes extract and
accumulate information from image structures with different char-
acteristics.

Fig. 1: The proposed Multi-Kernel Prediction Network architecture. It predicts per pixel kernels of various sizes for each of the input images
in a burst. The input of the network is a burst sequence of length \( N = 8 \), \( p \) is the sum of predefined kernel sizes and \( S \) is the set of these
kernels. Each of the burst images is deconvolved by the predicted per pixel kernels as seen in Fig. 2 and averaged to the final output. The
numbers below the blocks represent the 3-dimensional data structure at different levels of the architecture. The dimensions correspond to the
height x width x channel.
example, if the selected kernel sizes are $5 \times 5$ and $11 \times 11$, then $p = 5 + 11 = 16$. In addition, another bilinear upsampling layer is employed which scales up the learned linear combinations of 1D per pixel kernels of various sizes to match the input shape. Outer products of 1D kernels are computed to produce the 2D kernels. These kernels are applied on the input burst using local convolution as shown in Fig. 2. Lastly, the deconvolved burst images are averaged to produce the final output.

### 2.3. Training Parameters

All experimental models were trained in end-to-end fashion using backpropagation. Total loss consists of basic and annealing loss as devised in [14]. Basic loss is composed of mean squared error on pixel intensities and L1 loss on gradient intensities between the denoised image $\hat{I}$ and the ground truth $I$:

$$\ell(\hat{I}, I) = \lambda_1 \left\| \hat{I} - I \right\|_2^2 + \lambda_2 \left\| \nabla \hat{I} - \nabla I \right\|_1.$$  \hspace{1cm} (5)

The basic loss tries to make the average of all estimations $\hat{I}_s$ close to the ground truth $I$. However, using only the basic loss can lead to convergence at an undesirable local minima that does not utilize all the images in the burst effectively [14]. Therefore, we add a second loss term annealing loss to (5) that attempts to make each estimation $\hat{I}_s$ close to the ground truth $I$ independently. The total loss is obtained as follows:

$$\mathcal{L}(\hat{I}, I) = \ell(\hat{I}, I) + \beta \alpha \sum_{s=1}^{N} \sum_{i=5}^{8} \ell(\hat{I}_s, I),$$  \hspace{1cm} (6)

where $N$ is the number of images in the burst, $S$ is the set of kernel sizes, $\beta$ and $\alpha$ are the hyperparameters controlling the weight decay, $t$ is the training step, and $\lambda_1 + \lambda_2 = 1$. Please note that during training we discard in-place addition of kernels to enable better convergence. However, once the network is well trained, the in-place addition can help speed up inference.

### 3. EXPERIMENTS

We followed the same procedure for dataset generation and noise estimation as in [14]. We trained our models and evaluated their performance on synthetically generated datasets from the Open Images dataset [27].

#### 3.1. Data Generation

The images from the Open Images dataset were $4 \times$ downsampled in each dimension using a box filter to reduce noise and compression artifacts. Random patches of size $128 \times 128$ were sampled from the images and those were used for both creating the ground truth and the remaining $N - 1$ burst images, where $N$ is the total number of images in the burst. These burst images are offset from the first image by $x_i$ and $y_i$, where $x_i$ and $y_i$ are the offsets of image $i$ in horizontal and vertical directions, respectively. The offsets simulate misalignments between consecutive frames caused by hand movements that may occur during handheld photography. Values for $(x_i, y_i)$ are sampled with probability $n/N$ from a 2D uniform integer distribution between $[-16, 16]$, otherwise from a 2D uniform integer distribution between $[-2, 2]$, where $n \sim$ Poisson($\lambda$).

The burst images are also considered to be noisy, and hence a signal dependent Gaussian noise is added to the burst:

$$x_p \sim N(y_p, \sigma_r^2 + \sigma_s y_p),$$  \hspace{1cm} (7)

where $x_p$ is the noisy measurement of true intensity $y_p$ at pixel $p$. Read and shot noise parameters $\sigma_r$ and $\sigma_s$ are sampled uniformly from $[10^{-3}, 10^{-1.5}]$ and $[10^{-2}, 10^{-3}]$, respectively. These ranges were selected from the real observed data. Synthetic train dataset is generated on the fly, while the test datasets are pre-generated using different gains (noise levels) that correspond to a fixed set of read and shot noise parameters. The selected values simulate the light sensitivities that correspond to the ISO settings on a real camera. The read and shot noise for each gain is as given:

- Gain $\propto 1$: $\sigma_r = 10^{-2.1}, \sigma_s = 10^{-2.6}$,
- Gain $\propto 2$: $\sigma_r = 10^{-1.8}, \sigma_s = 10^{-2.3}$,
- Gain $\propto 4$: $\sigma_r = 10^{-1.4}, \sigma_s = 10^{-1.9}$,
- Gain $\propto 8$: $\sigma_r = 10^{-1.1}, \sigma_s = 10^{-1.5}$.

#### 3.2. Noise Estimation

The camera noise is estimated from the first image in a burst. The noise estimate helps the model denoise beyond the noise levels of the training data [14]. It is defined as:

$$\hat{\sigma}_p = \sqrt{\sigma_r^2 + \sigma_s \max(x_p, 0)},$$  \hspace{1cm} (8)

where $x_p$ is the intensity of pixel $p$ in the first image of a burst. In real data the noise parameters $\sigma_r$ and $\sigma_s$ are available in the DNG raw image format [28]. This noise estimate is of the same dimension as the burst images and is appended to the end of the burst.

In all our experiments, we set the parameters as follows $N = 8$, $\lambda = 1.5$, $\beta = 100$ and $\alpha = .9998$.

#### 3.3. Results and Discussion

In the experiments we performed using MKPN, we define the sizes of the kernels in advance – $S \in \{1, 3, 5, 7, 9, 11\}$. The kernel sizes and the number of kernels however, are not limited to these and can be used in any combination to produce the best results for a given...
Table 1: Results on test datasets with different gains (noise levels) in measures of PSNR and SSIM, where Gain ∝ 8 is the noisiest. MKPN outperforms the state-of-the-art KPN [14] and other KPN variations.

| Model     | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM | PSNR | SSIM |
|-----------|------|------|------|------|------|------|------|------|------|------|
|            | Gain ∝ 1 |      | Gain ∝ 2 |      | Gain ∝ 4 |      | Gain ∝ 8 |      |
| MKPN [14]  | 35.10 | 0.925 | 32.22 | 0.878 | 28.61 | 0.781 | 25.78 | 0.692 |
| KPN L₂₅  | 34.28 | 0.914 | 31.46 | 0.862 | 27.86 | 0.756 | 24.90 | 0.654 |
| KPN L₉₅  | 34.96 | 0.922 | 32.04 | 0.871 | 28.44 | 0.770 | 25.66 | 0.686 |
| KPN L₁₃  | 34.71 | 0.918 | 31.77 | 0.870 | 28.15 | 0.772 | 25.34 | 0.683 |
| KPN L₁₁  | 34.67 | 0.919 | 31.78 | 0.871 | 28.15 | 0.773 | 25.26 | 0.679 |
| KPN L₉   | 34.57 | 0.917 | 31.63 | 0.867 | 28.07 | 0.769 | 25.34 | 0.681 |
| KPN L₇   | 34.54 | 0.918 | 31.65 | 0.866 | 28.10 | 0.765 | 25.37 | 0.679 |
| KPN L₅   | 34.30 | 0.906 | 31.36 | 0.855 | 27.76 | 0.754 | 24.88 | 0.658 |
| KPN L₃   | 33.65 | 0.897 | 30.79 | 0.844 | 27.26 | 0.735 | 24.55 | 0.632 |
| KPN L₁   | 31.66 | 0.837 | 28.50 | 0.759 | 24.32 | 0.607 | 21.30 | 0.486 |

Fig. 3: Example of denoising an image of a bear at Gain ∝ 4. The detailed fur is recovered best by MKPN. KPN L₂₅ that uses a large kernel oversmooths the details of the fur. Best viewed on a screen.

Fig. 4: Example of denoising an image of a grasshopper at Gain ∝ 8. The detailed legs and smooth background are best recovered by the proposed method MKPN. Best viewed on a screen.

4. CONCLUSION

Burst image denoising, with its inherent challenges, remains to be an open problem. In this work, we propose MKPN, a DNN based method for denoising of burst images captured by handheld cameras. The novelty of this method lies in predicting kernels of different sizes and performing kernel fusion by in-place addition before the convolution operation. MKPN effectively combines the best behavior of small and large kernels – it manages to denoise flat areas as well as preserve the detailed image structures. Kernel fusion ensures that MKPN is able to extract different information from the different kernels without compromising on computational efficiency.
We empirically show that MKPN outperforms state-of-the-art models quantitatively and provides visually pleasing results.

5. REFERENCES

[1] C. Godard, K. Matzen, and M. Uyttendaele, “Deep burst denoising,” in European Conference on Computer Vision (ECCV), 2018, pp. 538–554.

[2] S. W. Hasinoff, D. Sharlet, R. Geiss, A. Adams, J. T. Barron, F. Kainz, J. Chen, and M. Levoy, “Burst photography for high dynamic range and low-light imaging on mobile cameras,” ACM Transactions on Graphics (TOG), vol. 35, no. 6, 2016.

[3] P. Perona and J. Malik, “Scale-space and edge detection using anisotropic diffusion,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 12, no. 7, pp. 629–639, 1990.

[4] L. I. Rudin, S. Osher, and E. Fatemi, “Nonlinear total variation based noise removal algorithms,” Physica D: Nonlinear Phenomena, vol. 60, no. 1–4, pp. 259–268, 1992.

[5] H. C. Burger, C. J. Schuler, and S. Harmeling, “Image denoising: Can plain networks compete with bm3d?” in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012, pp. 2392–2399.

[6] V. Jain and S. Seung, “Natural image denoising with convolutional networks,” in Advances in Neural Information Processing Systems (NIPS), 2009, pp. 769–776.

[7] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, “Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion,” Journal of Machine Learning Research, vol. 11, pp. 3371–3408, 2010.

[8] J. Xie, L. Xu, and E. Chen, “Image denoising and inpainting with deep neural networks,” in Advances in Neural Information Processing Systems (NIPS), 2012, pp. 341–349.

[9] F. Agostinelli, M. R. Anderson, and H. Lee, “Adaptive multi-column deep neural networks with application to robust image denoising,” in Advances in Neural Information Processing Systems (NIPS), 2013, pp. 1493–1501.

[10] L. Gondara, “Medical image denoising using convolutional denoising autoencoders,” in IEEE International Conference on Data Mining Workshops (ICDMW), 2016, pp. 241–246.

[11] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778.

[12] X. Mao, C. Shen, and Y.-B. Yang, “Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections,” in Advances in Neural Information Processing Systems (NIPS), 2016, pp. 2802–2810.

[13] T. Brooks, B. Mildenhall, T. Xue, J. Chen, D. Sharlet, and J. T. Barron, “Unprocessing images for learned raw denoising,” arXiv preprint arXiv:1811.11127, 2018.

[14] B. Mildenhall, J. T. Barron, J. Chen, D. Sharlet, R. Ng, and R. Carroll, “Burst denoising with kernel prediction networks,” in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 2502–2510.

[15] Z. Liu, L. Yuan, X. Tang, M. Uyttendaele, and J. Sun, “Fast burst images denoising,” ACM Transactions on Graphics (TOG), vol. 33, no. 6, pp. 232, 2014.

[16] C. Chen, Q. Chen, J. Xu, and V. Koltun, “Learning to see in the dark,” in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

[17] F. Kokkinos and S. Lefkimmiatis, “Iterative residual cnns for burst photography applications,” arXiv preprint arXiv:1811.12197, 2018.

[18] S. Niklaus, L. Mai, and F. Liu, “Video frame interpolation via adaptive convolution,” in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, vol. 1, p. 3.

[19] S. Bako, T. Vogels, B. Mcwilliams, M. Meyer, J. Novákov, A. Harvill, P. Sen, T. DeRose, and F. Rousselle, “Kernel-predicting convolutional networks for denoising monte carlo renderings,” ACM Transactions on Graphics (TOG), vol. 36, no. 4, 2017.

[20] T. Vogels, F. Rousselle, B. Mcwilliams, G. Röthlin, A. Harvill, D. Adler, M. Meyer, and J. Novákov, “Denoising with kernel prediction and asymmetric loss functions,” ACM Transactions on Graphics (TOG), vol. 37, no. 4, pp. 124, 2018.

[21] L. Mai and F. Liu, “Kernel fusion for better image deblurring,” in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 371–380.

[22] R. Rigamonti, A. Sironi, V. Lepetit, and P. Fua, “Learning separable filters,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 37, pp. 94–106, 2013.

[23] S. Han, H. Mao, and W. J. Dally, “Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding,” arXiv preprint arXiv:1510.00149, 2015.

[24] S. Wiedemann, A. Marban, K.-R. Müller, and W. Samek, “Entropy-constrained training of deep neural networks,” arXiv preprint arXiv:1812.07520, 2018.

[25] S. Wiedemann, K.-R. Müller, and W. Samek, “Compact and computationally efficient representation of deep neural networks,” arXiv preprint arXiv:1805.10692, 2018.

[26] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI), 2015, pp. 234–241.

[27] Google. Open Images Dataset V4, Google, 2018. https://storage.googleapis.com/openimages/web/index.html

[28] Adobe. Digital Negative (DNG) Specification, Adobe, 2012. https://www.adobe.com/content/dam/acom/en/products/photoshop/pdfs/dng_spec_1.4.0.0.pdf

[29] 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, vol. 13, no. 4, pp. 600–612, 2004.