AN ALGORITHM UNROLLING APPROACH TO DEEP IMAGE DEBLURRING

Yuelong Li[†]*, Mohammad Tofighi*[†]*, Vishal Monga*[†] *and Yonina C. Eldar*[‡]

[†]Department of Electrical Engineering, Pennsylvania State University, USA
[†]Department of Electrical Engineering, Technion-Israel Institute of Technology

ABSTRACT

While neural networks have achieved vastly enhanced performance over traditional iterative methods in many cases, they are generally empirically designed and the underlying structures are difficult to interpret. The algorithm unrolling approach has helped connect iterative algorithms to neural network architectures. However, such connections have not been made yet for blind image deblurring. In this paper, we propose a neural network architecture that advances this idea. We first present an iterative algorithm that may be considered a generalization of the traditional total-variation regularization method on the gradient domain, and subsequently unroll the half-quadratic splitting algorithm to construct a neural network. Our proposed deep network achieves significant practical performance gains while enjoying interpretability at the same time. Experimental results show that our approach outperforms many state-of-the-art methods.

1. INTRODUCTION

Blind image deblurring refers to the process of recovering a sharp image from its blurred observation. Among various deblurring problems, motion deblurring is an important topic because camera shaking is common during photography. Assuming a planar scene and translational camera motion, the blurring process is typically modeled as [1]: $\mathbf{y} = \mathbf{k} * \mathbf{x} + \mathbf{n}$ where $\mathbf{y}$ is the observed blur image, $\mathbf{x}$ is the latent sharp image, $\mathbf{k}$ is the blur kernel, and $\mathbf{n}$ is noise which is often modelled as Gaussian. When $\mathbf{k}$ is unknown the corresponding estimation problem is commonly called blind deconvolution.

The majority of existing blind motion deblurring methods rely on iterative optimization. These methods usually hinge on sparsity-inducing regularizers, either in the gradient domain [2, 3, 4, 5, 6, 7, 8, 9] or more general sparsifying transformation domains [10, 11, 12]. Variants of such methods may arise indirectly from a statistical estimation perspective, such as [13, 14, 15]. While these methods are typically physically interpretable, their performance depends heavily on appropriate selection of parameters and careful design of regularizers/priors, which are difficult to determine analytically. Furthermore, hundreds of iterations are usually required to achieve an acceptable performance level, and thus these algorithms can be computationally expensive.

Complementary to the aforementioned approaches, learning based methods for determining a non-linear mapping that deblurs the image while adapting parameter choices to an underlying training image set have been developed. Principally important in this class are techniques that employ deep neural networks, including [16, 17, 18, 19]. Although they offer practical promises in certain scenarios such as video deblurring and achieves substantial performance gains in some cases, these works commonly regard neural networks as abstract function approximators. The structures of the networks are typically empirically determined and the actual functionality of the neural networks is hard to interpret.

In the seminal work of Gregor *et al.* [20], a novel technique called algorithm unrolling was proposed that provides a neural network interpretation of iterative sparse coding algorithms. Passing through the network is equivalent to executing the iterative algorithm a finite number of times, and the trained network can be naturally interpreted as a parameter optimized algorithm. In blind deblurring, Schuler *et al.* [21] employ neural networks as feature extraction modules and integrate it into a trainable deblurring system. However, the network portions are still empirical and the whole system remains hard to interpret. The link between traditional iterative algorithms and neural networks remains largely unexplored for the problem of blind deblurring.

In this paper, we develop a neural network approach for blind motion deblurring in the spirit of algorithm unrolling, called Deblurring via Algorithm Unrolling (DAU). Parameters of the algorithm are optimized by training the network and performance gains are achieved without sacrificing interpretability. We experimentally verify its superior performance, both over best-known iterative algorithms and more recent neural network approaches.

2. DEBLURRING VIA ALGORITHM UNROLLING

The total-variation regularization approach in the gradient domain [22] solves the following optimization problem:

$$\min_{\mathbf{k},\mathbf{g}_1,\mathbf{g}_2} \frac{1}{2}\left(\|D_x\mathbf{y} - \mathbf{k} * \mathbf{g}_1\|_2^2 + \|D_y\mathbf{y} - \mathbf{k} * \mathbf{g}_2\|_2^2\right) + \lambda_1\|\mathbf{g}_1\|_1 + \lambda_2\|\mathbf{g}_2\|_1 + \frac{\epsilon}{2}\|\mathbf{k}\|_2^2,$$
$$\text{subject to } \mathbf{1}^T\mathbf{k} = 1, \quad \mathbf{k} \geq 0, \tag{1}$$

where $D_x\mathbf{y}, D_y\mathbf{y}$ are the partial derivates of $\mathbf{y}$ in horizontal and vertical directions respectively, $\mathbf{1}$ is a vector whose entries are all ones, and $\|\cdot\|_p$ denotes the ℓ^p vector norm. The parameters $\lambda_1, \lambda_2, \varepsilon$ are positive constants which balance the contributions of each term. The $\geq$ sign acts elementwise.

In practice, $D_x\mathbf{y}$ and $D_y\mathbf{y}$ are usually computed using discrete filters, such as the Prewitt and Sobel filters. From this viewpoint, a straightforward generalization of (1) is to use more than two filters. We formulate the generalized optimization problem as the following:

$$\min_{\mathbf{k},\{\mathbf{g}_i\}_i} \sum_{i=1}^{C} \left(\frac{1}{2} \|\mathbf{f}_i * \mathbf{y} - \mathbf{k} * \mathbf{g}_i\|_2^2 + \lambda_i \|\mathbf{g}_i\|_1 \right) + \frac{\epsilon}{2}\|\mathbf{k}\|_2^2,$$
$$\text{subject to } \|\mathbf{k}\|_1 = 1, \quad \mathbf{k} \geq 0, \tag{2}$$

where $\{\mathbf{f}_i\}_{i=1}^C$ is a collection of C filters that will be determined subsequently through learning.

Algorithm 1 Half-quadratic Splitting Algorithm

Input: Blurred image $\mathbf{y}$, filter banks $\{\mathbf{f}_i^l\}_{i,l}$, positive constant parameters $\{\zeta_i^l, \lambda_i^l\}_{i,l}, \varepsilon$, number of iterations L.

Output: Estimated kernel $\widetilde{\mathbf{k}}$, feature maps $\{\widetilde{\mathbf{g}_i}\}_{i=1}^C$.

Initialize $\mathbf{k} \leftarrow \delta; \mathbf{z}_i \leftarrow 0, i = 1, \ldots, C$.
for $l = 1$ **to** L **do**
  for $i = 1$ **to** C **do**
    $\mathbf{y}_i^l \leftarrow \mathbf{f}_i^l * \mathbf{y}$,
    $\mathbf{g}_i^{l+1} \leftarrow \mathcal{F}^{-1}\left\{ \frac{\zeta_i^l \widehat{\mathbf{k}^l}^* \odot \widehat{\mathbf{y}_i^l} + \widehat{\mathbf{z}_i^l}}{\zeta_i^l \left|\widehat{\mathbf{k}^l}\right|^2 + 1} \right\}$,
    $\mathbf{z}_i^{l+1} \leftarrow \mathcal{S}_{\lambda_i^l \zeta_i^l}\left\{\mathbf{g}_i^{l+1}\right\}$,
  end for
  $\mathbf{k}^{l+\frac{1}{3}} \leftarrow \mathcal{F}^{-1}\left\{ \frac{\sum_{i=1}^C \widehat{\mathbf{z}_i^{l+1}}^* \odot \widehat{\mathbf{y}_i^l}}{\sum_{i=1}^C \left|\widehat{\mathbf{z}_i^{l+1}}\right|^2 + \epsilon} \right\}$,
  $\mathbf{k}^{l+\frac{2}{3}} \leftarrow \left[\mathbf{k}^{l+\frac{1}{3}}\right]_+, \mathbf{k}^{l+1} \leftarrow \frac{\mathbf{k}^{l+\frac{2}{3}}}{\left\|\mathbf{k}^{l+\frac{2}{3}}\right\|_1}$,
end for

2.1. Efficient Minimization via Half-quadratic Splitting

A common approach to solve (1) and more generally (2) is the half-quadratic splitting algorithm [23]. The basic idea is to perform variable-splitting and then alternating minimization on the penalty function. To this end, we first cast (2) into the following approximation model:

$$\min_{\mathbf{k},\{\mathbf{g}_i,\mathbf{z}_i\}_i} \sum_{i=1}^{C} \left(\frac{1}{2}\|\mathbf{f}_i * \mathbf{y} - \mathbf{k} * \mathbf{g}_i\|_2^2 + \lambda_i \|\mathbf{z}_i\|_1 + \frac{1}{2\zeta_i}\|\mathbf{g}_i - \mathbf{z}_i\|_2^2 \right) + \frac{\epsilon}{2}\|\mathbf{k}\|_2^2,$$
$$\text{subject to } \|\mathbf{k}\|_1 = 1, \quad \mathbf{k} \geq 0, \tag{3}$$

by introducing auxiliary variables $\{\mathbf{z}_i\}_{i=1}^C$ and constant parameters $\zeta_i, i = 1, \ldots, C$. We then alternately minimize over $\{\mathbf{x}_i\}_i, \{\mathbf{z}_i\}_i$ and $\mathbf{k}$ and iterate until convergence.

In practice, a common strategy is to alter the parameters per iteration [23, 7, 22, 9]. In numerical analysis and optimization, this strategy is formally called continuation method. By adopting this strategy, we choose different parameters $\{\zeta_i^l, \lambda_i^l\}_{i,l}$ across the iterations l. We take this idea one step further by varying the filters $\{\mathbf{f}_i\}_i$ as well. The complete algorithm is summarized in Algorithm 1. We let $\widehat{\cdot}$ denote the Discrete Fourier Transform (DFT) and $\mathcal{F}^{-1}$ be the inverse DFT. We define $[x]_+ = \max\{x, 0\}$, δ is the unit impulse function, $\cdot^*$ is the complex conjugation and $\odot$ is the Hadamard product operator. Finally, $\mathcal{S}_\lambda(\cdot)$ is the soft-thresholding operator: $\mathcal{S}_\lambda(x) = \text{sgn}(x) \cdot \max\{|x| - \lambda, 0\}$. Operations matrices and vectors act elementwise.

After algorithm 1 converges, we obtain the estimated feature maps $\{\widetilde{\mathbf{g}_i}\}_i$ and the estimated kernel $\widetilde{\mathbf{k}}$. When $\widetilde{\mathbf{k}}$ approximates $\mathbf{k}$, $\widetilde{\mathbf{g}_i}$ should approximate $\mathbf{f}_i * \mathbf{x}$. Therefore, we retrieve the image $\mathbf{x}$ by solving the following optimization problem:

$$\widetilde{\mathbf{x}} \leftarrow \arg\min_{\mathbf{x}} \frac{1}{2}\left\|\mathbf{y} - \widetilde{\mathbf{k}} * \mathbf{x}\right\|_2^2 + \sum_{i=1}^{C} \frac{\eta_i}{2}\|\mathbf{f}_i * \mathbf{x} - \widetilde{\mathbf{g}_i}\|_2^2$$
$$= \mathcal{F}^{-1}\left\{ \frac{\widehat{\widetilde{\mathbf{k}}}^* \odot \widehat{\mathbf{y}} + \sum_{i=1}^C \eta_i \widehat{\mathbf{f}_i}^* \odot \widehat{\widetilde{\mathbf{g}_i}}}{\left|\widehat{\widetilde{\mathbf{k}}}\right|^2 + \sum_{i=1}^C \eta_i \left|\widehat{\mathbf{f}_i}\right|^2} \right\}, \tag{4}$$

where η_i's are positive constant parameters.

2.2. Network Construction via Algorithm Unrolling

Each step of Algorithm 1 is in analytic form and can be implemented using a series of basic functional operations. Therefore, each iteration of Algorithm 1 admits a layered representation, and repeating it L times yields an L-layer neural network (assuming L iterations). For notational brevity, we concatenate the parameters in each layer and let $\mathbf{f}^l = (\mathbf{f}_i^l)_{i=1}^C, \zeta^l = (\zeta_i^l)_{i=1}^C, \lambda^l = (\lambda_i^l)_{i=1}^C$ and $\eta = (\eta_i)_{i=1}^C$. We also concatenate $\mathbf{y}_i^l$'s, $\mathbf{z}_i^l$'s and $\mathbf{g}_i^l$'s by letting $\mathbf{y}^l = (\mathbf{y}_i^l)_{i=1}^C$, $\mathbf{z}^l = (\mathbf{z}_i^l)_{i=1}^C$ and $\mathbf{g}^l = (\mathbf{g}_i^l)_{i=1}^C$, respectively.

To handle large blur kernels, we alter the size of the filter banks $\{\mathbf{f}_i\}_i$ in different layers in the following way:

$$\text{size of } \mathbf{f}_i^1 > \text{ size of } \mathbf{f}_i^2 > \text{ size of } \mathbf{f}_i^3 > \ldots.$$

so that high-level representations features are captured first, and fine details emerge in later iterations. To facilitate training, we produce large filters by cascading small 3×3 filters, following the same principle as [24]. Formally speaking, we set $\mathbf{f}_i^L = \mathbf{w}_{i1}^L$ where $\{\mathbf{w}_{i1}^L\}_{i=1}^C$ is a collection of 3×3 filters, and recursively obtain $\mathbf{f}_i^l$ by: $\mathbf{f}_i^l \leftarrow \sum_{j=1}^C \mathbf{w}_{ij}^l * \mathbf{f}_j^{l+1}$. Using this representation, we obtain the network structure in Fig. 1. The parameters $\{\mathbf{w}^l, b^l, \lambda^l\}_{l=1}^L$ will be learned from the training data, as explained in the next Section.

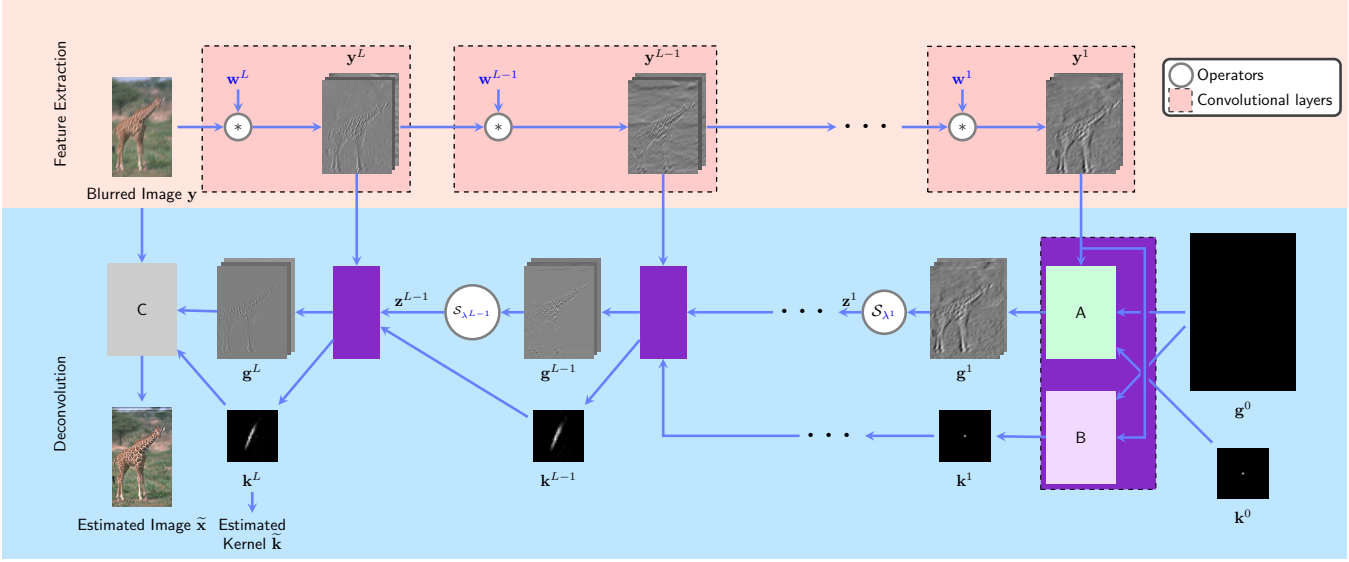


Fig. 1. Structure of the deep network constructed by algorithm unrolling and cascaded filtering. Block A and B implements Step 5 and Step 8 in Algorithm 1 respectively, while block C implements (4). A diagram representation can be found at `http://signal.ee.psu.edu/diagram.pdf`. Intermediate data (hidden layers) on the trained network are also shown. It can be observed that, as l increases, more details are extracted in $\mathbf{g}^l$ and finer kernel coefficients are recovered. The parameters that will be learned from real datasets are colored in blue.

Table 1. Quantitative comparison averaged over 200 images from the BSDS500 [26] set and 4 linear kernels. The RMSE values are computed over kernels. Best scores are in bold.

Metrics	DAU	[22]	[25]
PSNR (dB)	**27.21**	22.23	25.23
ISNR (dB)	**4.36**	2.06	1.88
SSIM	**0.88**	0.76	0.81
RMSE ($\times 10^{-3}$)	**2.21**	5.21	–

Table 2. Quantitative comparison on nonlinear motion (average over 4 images and 8 kernels from [27]). The RMSE values are computed over kernels. Best scores are in bold.

	DAU	[22]	[25]	[18]	[19]
PSNR (dB)	**27.15**	26.79	24.51	23.18	26.75
ISNR (dB)	**3.79**	3.63	1.35	0.02	3.59
SSIM	0.88	**0.89**	0.81	0.81	**0.89**
RMSE ($\times 10^{-3}$)	3.87	**3.83**	–	–	3.98

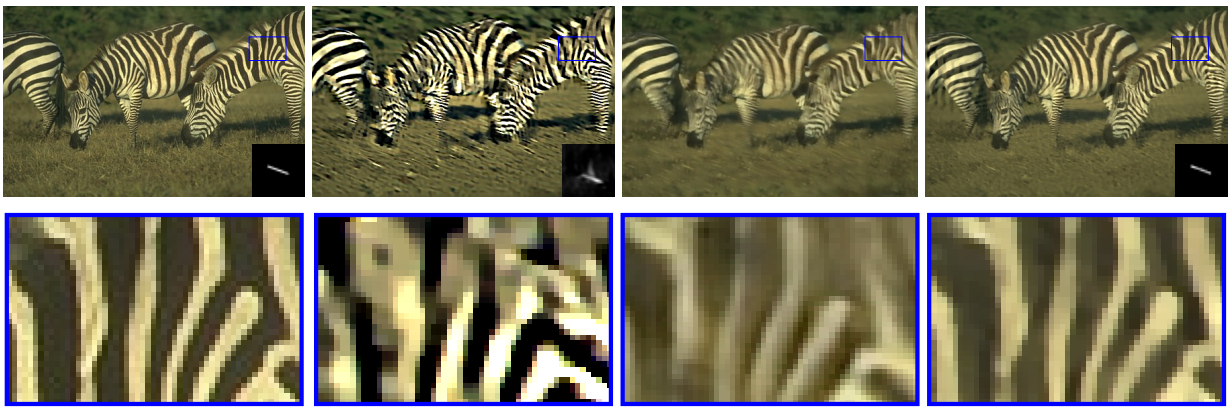
3. EXPERIMENTS

Training: We use the training and validation portions from the Berkeley Segmentation Dataset (BSDS500) [26] as training images. The linear motion kernels are generated by uniformly sampling 16 angles in $[0, \pi]$ and 16 lengths in $[5, 20]$. The images are convolved with each kernel and white Gaussian noise with standard deviation 0.01 (suppose the image intensity is in $[0, 1]$) is added. For each blurred image $\mathbf{y}_t^{\text{train}} (t = 1, \ldots, T)$, we let the corresponding sharp image and kernel be $\mathbf{x}_t^{\text{train}}$ and $\mathbf{k}_t^{\text{train}}$, respectively. We re-parametrize λ_i^l in step 6 of Algorithm 1 by letting $b_i^l = \lambda_i^l \zeta_i^l$ and let $b^l = (b_i^l)_{i=1}^C, l = 1, \ldots, L$. The network outputs $\widetilde{\mathbf{x}}_t, \widetilde{\mathbf{k}}_t$ corresponding to $\mathbf{y}_t^{\text{train}}$ depend on the network parameters $\mathbf{w}^l, b^l, \lambda^l, l = 1, 2, \ldots, L$, and $\widetilde{\mathbf{x}}_t$ further depends on η. We train the network to determine those parameters by minimizing:

$$\min_{\{\mathbf{w}^l, b^l, \lambda^l\}_{l=1}^L, \eta} \sum_{t=1}^{T} \text{MSE}\left(\mathbf{x}_t^{\text{train}} - \widetilde{\mathbf{x}}_t\left(\{\mathbf{w}^l, b^l, \lambda^l\}_{l=1}^L, \eta\right)\right) + \kappa \,\text{MSE}\left(\mathbf{k}_t^{\text{train}} - \widetilde{\mathbf{k}}_t\left(\{\mathbf{w}^l, b^l, \lambda^l\}_{l=1}^L\right)\right),$$
$$\text{subject to } b_i^l \geq 0,\ \lambda_i^l \geq 0, \quad l = 1, \ldots, L, i = 1, \ldots, C,$$

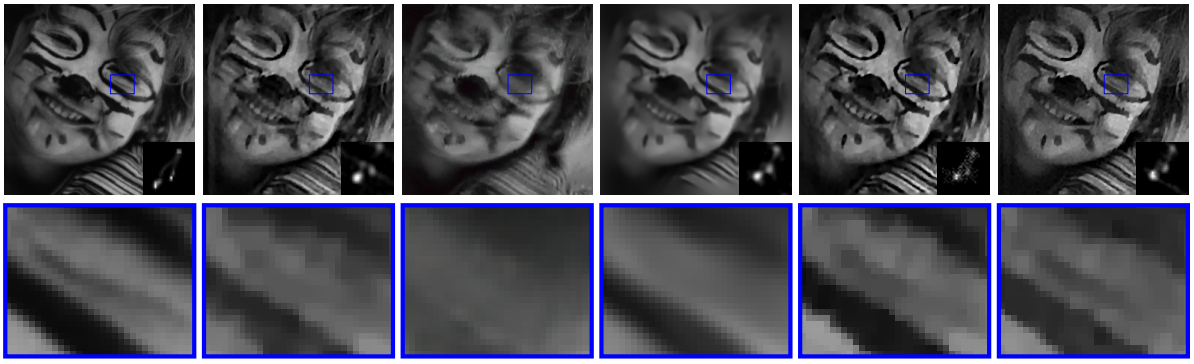
where $\kappa > 0$ is a constant parameter which is fixed to 10^5 and MSE is the Mean Squared Error. We choose $L = 10$ and $C = 16$ by cross-validation. The minimization is performed by stochastic gradient descent, followed by a gradient projection step to enforce the non-negative constraints. We use the Adam [28] solver for faster training. The learning rate is set to 1×10^{-3} initially and decayed by a factor of 0.9 per epoch. We terminate training after 20 epochs. The parameters $\{\lambda_i^l\}_{i,l}$ are initialized to zeros, $\{b_i^l\}_{i,l}$ to 1, and $\{\eta_i\}_i$ to 20, respectively. The weights are initialized according to [29].

Evaluation: We use 200 images from the test portion from the BSDS500 dataset [26] as test images. We randomly



(a) Groundtruth (b) Perrone *et al.* [22] (c) Nah *et al.* [25] (d) DAU

Fig. 2. Qualitative comparisons on the BSDS500 dataset [26]. The blur kernels are placed at the right below corner. DAU recovers the kernel at higher accuracy and therefore the estimated images are more faithful to the groundtruth.



(a) Groundtruth (b) Perrone *et al.* [22] (c) Nah *et al.* [25] (d) Chakrabarti [18] (e) Xu *et al.* [19] (f) DAU

Fig. 3. Qualitative comparisons on the dataset from [27]. The blur kernels are placed at the right below corner. DAU generates fewer artifacts and preserves more details than competing state of the art methods.

choose angles from $[0, \pi]$ and lengths from $[5, 20]$ to generate 4 test kernels. We compare with state-of-the art algorithms, Perrone *et al.* [22] and Nah *et al.* [25], which are representatives of iterative algorithms and deep-learning approaches. We assess the performance using four commonly used evaluation metrics: Peak Signal-to-Noise-Ratio (PSNR), Improvement in Signal-to-Noise-Ratio (ISNR), Structural Similarity Index (SSIM) [30], and Root-Mean-Square Error (RMSE) between the estimated kernel and the groundtruth kernel. The average scores are in Table 1. Clearly, DAU outperforms state-of-the art algorithms by a significant margin.

Fig. 2 shows example images and kernels for a qualitative comparison. Although Perrone *et al.*'s method can roughly infer the directions of the blur kernels, the recovered coefficients are unsatisfactory. As a result, the recovered image contains clearly visible artifacts. Nah *et al.*'s method effectively removes most of the blurs, but blurring artifacts still remain locally and the details are not faithfully preserved. In contrast, the kernel recovered by DAU is closer to the ground truth and hence leads to a more accurate estimated image.

Additionally, we compare the performance of various methods on deblurring under non-linear motion kernels, which is a more realistic scenario as discussed in [27]. We collect training kernels by interpolating the paths provided by [31] and created by ourselves in the same manner. We further augment these kernels by scaling and rotations. We use the standard image set from [27] (comprising 4 images and 8 kernels) as the test set. The average scores are presented in Table 2. Again DAU outperforms state-of-the-art methods. A visual example is shown in Fig. 3,

4. CONCLUSION

We propose a neural network deblurring architecture built by unrolling an iterative algorithm. We show how a generalized TV-regularized algorithm can be recast into a neural network, and train it to optimize the parameters. Unlike most existing deblurring networks, our work has the benefit of interpretability, while exhibiting performance benefits that are shared with modern deep-nets and exceed state of the art performance.

5. REFERENCES

[1] D. Kundur and D. Hatzinakos, "Blind image deconvolution," *IEEE Signal Process. Mag.*, vol. 13, no. 3, pp. 43–64, May 1996.

[2] N. Joshi, R. Szeliski, and D. J. Kriegman, "PSF estimation using sharp edge prediction," in *Proc. IEEE Conf. CVPR*, June 2008.

[3] Qi Shan, Jiaya Jia, and Aseem Agarwala, "High-quality Motion Deblurring from a Single Image," in *Proc. ACM SIGGRAPH*, 2008.

[4] Sunghyun Cho and Seungyong Lee, "Fast Motion Deblurring," in *Proc. ACM SIGGRAPH Asia*, 2009.

[5] Li Xu and Jiaya Jia, "Two-phase kernel estimation for robust motion deblurring," in *Proc. ECCV*, 2010.

[6] Dilip Krishnan, Terence Tay, and Rob Fergus, "Blind deconvolution using a normalized sparsity measure," in *Proc. IEEE Conf. CVPR*, 2011.

[7] L. Xu, S. Zheng, and J. Jia, "Unnatural L0 Sparse Representation for Natural Image Deblurring," in *Proc. IEEE Conf. CVPR*, June 2013.

[8] L. Sun, S. Cho, J. Wang, and J. Hays, "Edge-based blur kernel estimation using patch priors," in *Proc. IEEE ICCP*, Apr. 2013.

[9] J. Pan, Z. Hu, Z. Su, and M. H. Yang, "L_0 -Regularized Intensity and Gradient Prior for Deblurring Text Images and Beyond," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 2, pp. 342–355, Feb. 2017.

[10] Jian-Feng Cai, Hui Ji, Chaoqiang Liu, and Zuowei Shen, "Framelet-Based Blind Motion Deblurring From a Single Image," *IEEE Trans. Image Process.*, vol. 21, no. 2, pp. 562–572, Feb. 2012.

[11] Shiming Xiang, Gaofeng Meng, Ying Wang, Chunhong Pan, and Changshui Zhang, "Image Deblurring with Coupled Dictionary Learning," *Int. J. Comput. Vis.*, vol. 114, no. 2-3, pp. 248–271, Sept. 2015.

[12] J. Pan, D. Sun, H. Pfister, and M. H. Yang, "Deblurring Images via Dark Channel Prior," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PP, no. 99, pp. 1–1, 2018.

[13] Rob Fergus, Barun Singh, Aaron Hertzmann, Sam T. Roweis, and William T. Freeman, "Removing Camera Shake from a Single Photograph," in *Proc. ACM SIGGRAPH*, New York, NY, USA, 2006.

[14] A. Levin, Y. Weiss, F. Durand, and W. T. Freeman, "Efficient marginal likelihood optimization in blind deconvolution," in *Proc. IEEE Conf. CVPR*, June 2011.

[15] S. Derin Babacan, Rafael Molina, Minh N. Do, and Aggelos K. Katsaggelos, "Bayesian Blind Deconvolution with General Sparse Image Priors," in *Proc. ECCV*, Oct. 2012.

[16] Li Xu, Jimmy SJ Ren, Ce Liu, and Jiaya Jia, "Deep convolutional neural network for image deconvolution," in *Proc. NIPS*, 2014.

[17] R. Yan and L. Shao, "Blind Image Blur Estimation via Deep Learning," *IEEE Trans. Image Process.*, vol. 25, no. 4, pp. 1910–1921, Apr. 2016.

[18] Ayan Chakrabarti, "A Neural Approach to Blind Motion Deblurring," in *Proc. ECCV*, Oct. 2016.

[19] X. Xu, J. Pan, Y. J. Zhang, and M. H. Yang, "Motion Blur Kernel Estimation via Deep Learning," *IEEE Trans. Image Process.*, vol. 27, no. 1, pp. 194–205, Jan. 2018.

[20] Karol Gregor and Yann LeCun, "Learning fast approximations of sparse coding," in *Proc. ICML*, 2010.

[21] Christian J. Schuler, Michael Hirsch, Stefan Harmeling, and Bernhard Scholkopf, "Learning to Deblur," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 7, pp. 1439–1451, July 2016.

[22] Daniele Perrone and Paolo Favaro, "A Clearer Picture of Total Variation Blind Deconvolution," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 6, pp. 1041–1055, June 2016.

[23] Y. Wang, J. Yang, W. Yin, and Y. Zhang, "A New Alternating Minimization Algorithm for Total Variation Image Reconstruction," *SIAM J. Imaging Sci.*, vol. 1, no. 3, pp. 248–272, Jan. 2008.

[24] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. ICLR*, 2015.

[25] Seungjun Nah, Tae Hyun Kim, and Kyoung Mu Lee, "Deep multi-scale convolutional neural network for dynamic scene deblurring," in *Proc. IEEE Conf. CVPR*, 2017, vol. 1, p. 3.

[26] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proc. IEEE ICCV*, July 2001.

[27] Anat Levin, Yair Weiss, Fredo Durand, and William T. Freeman, "Understanding and evaluating blind deconvolution algorithms," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. 2009, pp. 1964–1971, IEEE.

[28] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2015.

[29] Xavier Glorot and Yoshua Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. ICAIS*, Mar. 2010.

[30] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[31] Rolf Köhler, Michael Hirsch, Betty Mohler, Bernhard Schölkopf, and Stefan Harmeling, "Recording and playback of camera shake: Benchmarking blind deconvolution with a real-world database," in *European Conference on Computer Vision*. 2012, pp. 27–40, Springer.