An Algorithm Unrolling Approach to Deep Image Deblurring

Yuelong Li[†], Mohammad Tofighi[†],
Vishal Monga[†], Yonina C. Eldar[‡]
[†]Pennsylvania State University [‡]Technion-Israel Institute of Technology

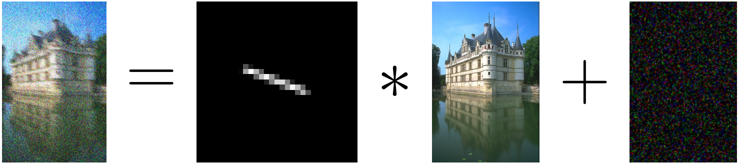
IEEE International Conference on Acoustics, Speech and Signal Processing

May 15, 2019

Outline

Background and Motivations

- Image deblurring is an essential topic in image reconstruction
  - Blurred images suffer from degraded visual quality
  - Image recognition algorithms may perform poorly when working on blurred images
- Uniform motion blur can be modeled as spatial convolutions

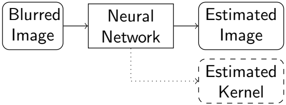


$\mathbf{y}$ Blurred Image $=$ $\mathbf{k}$ Blur Kernel $*$ $\mathbf{x}$ Sharp Image $+$ $\mathbf{n}$ Random Noise

- Blind motion deblurring: joint estimation of **k** and **x** given **y**
  - Very challenging inverse problem
  - But more practical as **k** is often unknown in reality

Approaches for Recovering Deblurred Image

- Two typical categories with complementary merits

	Iterative algorithms[1,2]	Neural networks[3,4]
General idea	$\min_{\mathbf{k},\mathbf{x}} \frac{1}{2}\|\mathbf{y}-\mathbf{k}*\mathbf{x}\|_2^2 + \lambda_1\phi(\mathbf{x}) + \lambda_2\psi(\mathbf{k})$	
Efficiency	Low	High
Interpretability	High	Low

- Q: is it possible to develop an approach that enjoys the merits of both worlds?

[1] L. Xu *et al.*, ECCV, 2010

[2] D. Perrone *et al.*, TPAMI, 2016

[3] A. Chakrabarti *et al.*, ECCV, 2016

[4] X. Xu, TIP, 2018

Algorithm Unrolling

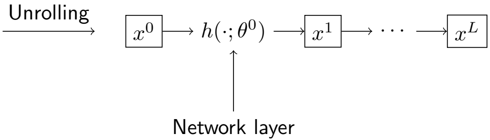
- Neural networks can be used to approximate sparse coding[5]
- The same idea may be used for generic iterative algorithms[6,7]

Algorithm: Input x^0, **Output** x^L

for $l = 1, 2, \ldots, L$ **do**

$x^{l+1} \leftarrow h(x^l; \theta^l)$,

end for



- For instance, $h(.)$ in [5] involves a soft-thresholding followed by a shrinking function.
- The unrolled network can thus be trained end-to-end
  - Optimize the algorithm parameters towards real world datasets $\longrightarrow$ enhance performance
  - Reduce the number of required iterations $\longrightarrow$ improve computational efficiency

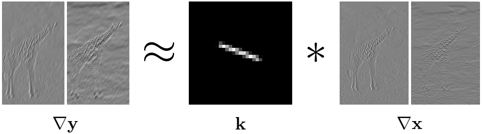
[5] K. Gregor *et al.*, NIPS, 2010.

[6] Z. Wang *et al.*, ICCV, 2015

[7] K. Jin *et al.*, TIP, 2017

Generalizing Total Variation Blind Deblurring

- Deblurring can be solved in the gradient domain[8]
  - By enforcing gradient sparsity (total variation regularization)



- Image gradients are commonly computed by linear filtering
- Then the following optimization problem is solved:

$$\min_{\mathbf{k},\mathbf{g}_1,\mathbf{g}_2} \frac{1}{2}\left(\|D_x\mathbf{y} - \mathbf{k} * \mathbf{g}_1\|_2^2 + \|D_y\mathbf{y} - \mathbf{k} * \mathbf{g}_2\|_2^2\right)$$

$$+ \lambda_1\|\mathbf{g}_1\|_1 + \lambda_2\|\mathbf{g}_2\|_1 + \frac{\epsilon}{2}\|\mathbf{k}\|_2^2,$$

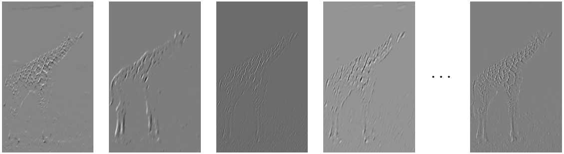
$$\text{subject to} \quad 1^T\mathbf{k} = 1, \mathbf{k} \geq 0,$$

- The solutions $\mathbf{g}_1$ and $\mathbf{g}_2$ are estimates of the gradients of the sharp image $\mathbf{x}$, i.e., we may expect $\mathbf{g}_1 \approx D_x\mathbf{x}$ and $\mathbf{g}_2 \approx D_y\mathbf{x}$

[8] Y. Yang *et al.*, SIAM 2008

A Filter Domain Regularization Perspective

- Our idea: increase the number of filters to $C > 2$ and learn their coefficients $\{\mathbf{f}_i\}_{i=1}^{C}$
  - Use more filters to capture richer image features
  - Learn the filter coefficients by unrolling



$\mathbf{f}_1 * \mathbf{x}$ $\quad$ $\mathbf{f}_2 * \mathbf{x}$ $\quad$ $\mathbf{f}_3 * \mathbf{x}$ $\quad$ $\mathbf{f}_4 * \mathbf{x}$ $\quad \cdots \quad$ $\mathbf{f}_C * \mathbf{x}$

- Formulation in the filtered domain

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

$$\text{subject to} \quad 1^T \mathbf{k} = 1, \mathbf{k} \geq 0.$$

The Iterative Minimization Algorithm

- Then we cast it into the following approximation model:

$$\min_{\mathbf{k},\{\mathbf{g}_i,\mathbf{z}_i\}_{i=1}^C} \sum_{i=1}^{C} \left(\frac{1}{2}\|\mathbf{f}_i * \mathbf{y} - \mathbf{k} * \mathbf{g}_i\|_2^2 + \lambda_i \|\mathbf{z}_i\|_1 + \frac{1}{2\zeta_i}\|\mathbf{g}_i - \mathbf{z}_i\|_2^2 \right) + \frac{\epsilon}{2}\|\mathbf{k}\|_2^2,$$

$$\text{subject to} \quad 1^T\mathbf{k} = 1, \mathbf{k} \geq 0,$$

- Minimization by half-quadratic splitting

$$\mathbf{g}_i^{l+1} \leftarrow \arg\min_{\mathbf{g}_i} \frac{1}{2}\left\|\mathbf{f}_i^l * \mathbf{y} - \mathbf{k}^l * \mathbf{g}_i\right\|_2^2 + \frac{1}{2\zeta_i^l}\left\|\mathbf{g}_i - \mathbf{z}_i^l\right\|_2^2, \quad \forall i,$$

$$\mathbf{z}_i^{l+1} \leftarrow \arg\min_{\mathbf{z}_i} \frac{1}{2\zeta_i^l}\left\|\mathbf{g}_i^{l+1} - \mathbf{z}_i\right\|_2^2 + \lambda_i^l \|\mathbf{z}_i\|_1, \quad \forall i,$$

$$\mathbf{k}^{l+1} \leftarrow \arg\min_{\mathbf{k}} \sum_{i=1}^{C} \frac{1}{2}\left\|\mathbf{f}_i^l * \mathbf{y} - \mathbf{k} * \mathbf{g}_i^{l+1}\right\|_2^2 + \frac{\epsilon}{2}\|\mathbf{k}\|_2^2,$$

$$\text{subject to} \quad 1^T\mathbf{k} = 1, \mathbf{k} \geq 0.$$

- For **g** and **z**, there exist close-from solutions. The problem for **k** is also solved through series of three close-form solutions.

Iterative Minimization with Dynamic Parameters

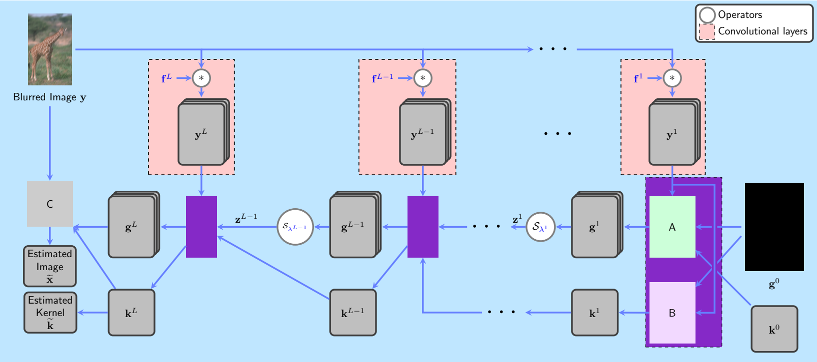
- Reconstruct image from feature maps $\{\mathbf{g}_i\}_{i=1}^{C}$

$$\widetilde{\mathbf{x}} \leftarrow \arg\min_{\mathbf{x}} \frac{1}{2} \|\mathbf{y} - \mathbf{k} * \mathbf{x}\|_2^2 + \sum_{i=1}^{C} \frac{\eta_i}{2} \|\mathbf{f}_i * \mathbf{x} - \mathbf{g}_i\|_2^2$$

$$= \mathcal{F}^{-1} \left\{ \frac{\widehat{\widetilde{\mathbf{k}}}^* \odot \widehat{\mathbf{y}} + \sum_{i=1}^{C} \eta_i \widehat{\mathbf{f}_i^L}^* \odot \widehat{\widetilde{\mathbf{g}}}_i}{\left|\widehat{\widetilde{\mathbf{k}}}\right|^2 + \sum_{i=1}^{C} \eta_i \left|\widehat{\mathbf{f}_i^L}\right|^2} \right\}$$

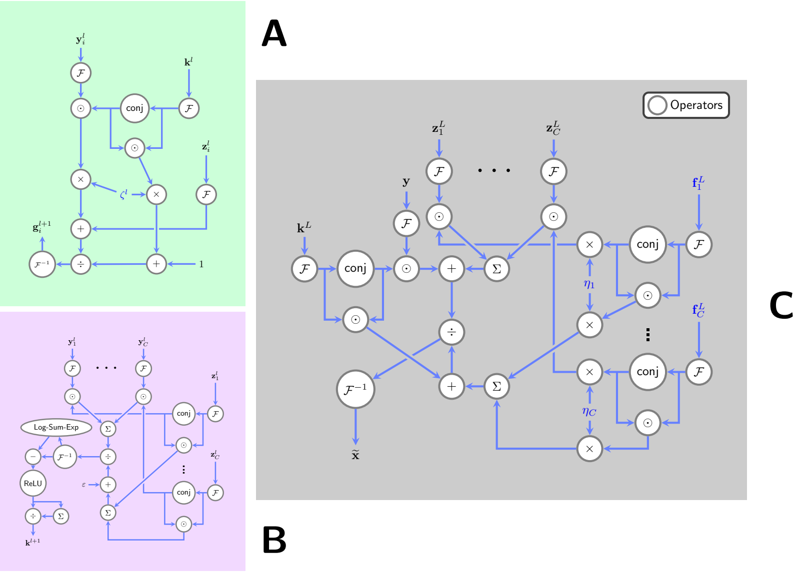
- It is beneficial to vary the parameters at different iteratons
  - In particular, λ_i's and ζ_i's may vary per iteration
- Decreasing ζ_i ($\zeta_i \to 0$) formally called continuation method
- We adopt the same strategy in network construction and use layer-specific parameters $\left\{\lambda_i^l, \zeta_i^l, \mathbf{f}_i^l\right\}_{i,l}$

Network Constructed through Unrolling[9]



[9] Every learnable parameters are colored in blue

Network Constructed through Unrolling



Training

- Collect training image samples $\{\mathbf{x}_t\}_{t=1}^{T}$ and $\{\mathbf{k}_t\}_{t=1}^{T}$
- Let the t-th estimated image and kernel be $\widetilde{\mathbf{x}}_t$, $\widetilde{\mathbf{k}}_t$
- Loss function

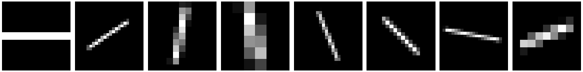
$$\min_{\{\mathbf{f}^l, b^l, \lambda^l\}_{l=1}^{L}, \eta} \sum_{t=1}^{T} \mathrm{MSE}\left(\mathbf{x}_t^{\mathrm{train}} - \widetilde{\mathbf{x}}_t\left(\{\mathbf{f}^l, b^l, \lambda^l\}_{l=1}^{L}, \eta\right)\right)$$
$$+\kappa\,\mathrm{MSE}\left(\mathbf{k}_t^{\mathrm{train}} - \widetilde{\mathbf{k}}_t\left(\left\{\mathbf{f}^l,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

- where $\mathbf{f}^l = (\mathbf{f}_i^l)_{i=1}^{C}$, $b^l = (b_i^l)_{i=1}^{C}$, $\lambda^l = (\lambda_i^l)_{i=1}^{C}$ and $\eta = (\eta_i)_{i=1}^{C}$
- Stochastic gradient descent with projection to ensure positiveness of b,η and λ

Experimental Results

Training and Testing Setup for Linear Kernel

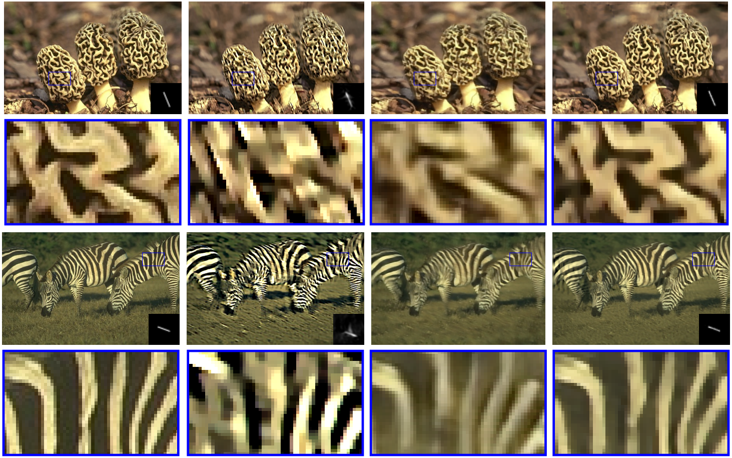
- Training set: 300 images from the training and validation subset of Berkeley Segmentation Data Set 500 (BSDS500)[10] out of 500 natural images.
- We generated 256 linear kernels by varying the length and angle of the kernels.



- Testing set: 200 images from the test portion of BSDS500.
- We randomly choose four kernels of different angle and length as test kernels.

[10] P. Arbelaez *et al.*, IEEE TPAMI 2011

Visual Results on Linear Motion



(a) Groundtruth (b) Perrone *et al.* (c) Nah *et al.* (d) DAU

Perrone *et al.*, IEEE TPAMI 2016
Nah *et al.*, CVPR 2017

Quantitative Comparisons on Linear Motion

Table: Quantitative comparison (average of 200 images and 4 kernels).

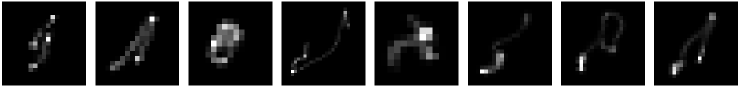
Metrics	DAU	Perrone *et al.*[13]	Nah *et al.*[14]	Xu *et al.*[15]	Kupyn *et al.*[16]
PSNR (dB)	**27.30**	22.23	24.82	24.02	23.98
ISNR (dB)	**4.45**	2.06	1.92	1.12	1.05
SSIM	**0.88**	0.76	0.80	0.78	0.78
RMSE $(\times 10^{-3})$	**1.67**	5.21	–	2.40	–

- RMSE is calculated with respect to the kernel.
- RMSE for Nah *et al.* and Kupyn *et al.*, is not reported since their methods directly recover deblurred image without estimating the kernel.

[13] Perrone *et al.*, IEEE TPAMI 2016
[14] Nah *et al.*, CVPR 2017
[15] Xu *et al.*, IEEE TIP 2018
[16] Kupyn *et al.*, CVPR 2018

Training and Testing Setup for Nonlinear Kernel

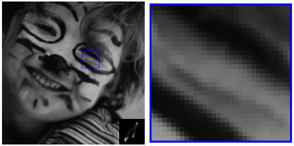
- We used 330K images from the Microsoft COCO[17] dataset.
- We generated around 30,000 real world kernels by recording camera motion trajectories.



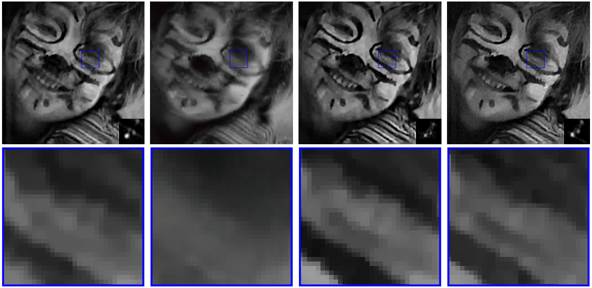
- Testing set: We test on *two* benchmark datasets specifically developed for evaluating blind deblurring with non-linear kernels:
  - 4 images and 8 kernels by Levin *et al.*[18]
  - 80 images and 8 kernels from Sun *et al.*[19]

[17] T-Y. Lin *et al.*, ECCV 2014
[18] Levin *et al.*, IEEE TPAMI, 2011
[19] Sun *et al.*, IEEE ICCP, 2013

Visual Results on Non-linear Motion



(a) Groundtruth



(b) Perrone *et al.* (c) Nah *et al.* (d) Xu *et al.* (e) DAU

Perrone *et al.*, IEEE TPAMI 2016
Nah *et al.*, CVPR 2017
Xu *et al.*, IEEE TIP 2018

Quantitative Comparisons on Non-linear Motion

	DAU	Perrone *et al.*[23]	Nah *et al.*[24]	Xu *et al.*[25]
PSNR (dB)	**27.15**	26.79	24.51	26.75
ISNR (dB)	**3.79**	3.63	1.35	3.59
SSIM	**0.89**	**0.89**	0.81	**0.89**
RMSE $(\times 10^{-3})$	3.87	**3.83**	–	3.98

[23] Perrone *et al.*, IEEE TPAMI 2016
[24] Nah *et al.*, CVPR 2017
[25] Xu *et al.*, IEEE TIP 2018

Running Time Comparisons

	DAU	Perrone *et al.*[26]	Nah *et al.*[27]	Xu *et al.*[28]	Kupyn *et al.*[29]
Runtime (GPU/CPU) (s)	**0.05/1.47**	−/1462.90	7.32/−	2.01/6.89	0.13/10.29
Number of Parameters	$\mathbf{2.3 \times 10^4}$	−	2.3×10^7	6.0×10^6	1.2×10^7

[26] Perrone *et al.*, IEEE TPAMI 2016
[27] Nah *et al.*, CVPR 2017
[28] Xu *et al.*, IEEE TIP 2018
[29] Kupyn *et al.*, CVPR 2018

Summary of Contributions

- An interpretable neural network architecture inspired by algorithm unrolling
- Superior performance, reduced number of parameters and computational benefits
- Code and datasets freely online for reproducibility[30]

[30] https://scholarsphere.psu.edu/concern/generic_works/8sf268475m

Thank you!

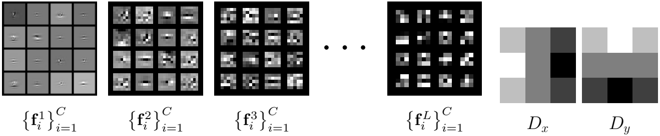
Questions?

Cascaded Filtering

- Capture large structures first, and let details emerge later
- Correspondingly, parameterization of filter coefficients

$$\text{size of } \mathbf{f}^1 > \text{ size of } \mathbf{f}^2 > \cdots > \text{ size of } \mathbf{f}^L$$

which can be difficult to learn in practice



- Re-parametrize with small (3×3) filters $\{\mathbf{w}_{ij}^l\}_{i,j,l}$:

$$\mathbf{f}_i^l \leftarrow \sum_{j=1}^{C} \mathbf{w}_{ij}^l * \mathbf{f}_j^{l+1}.$$

Network Flowchart

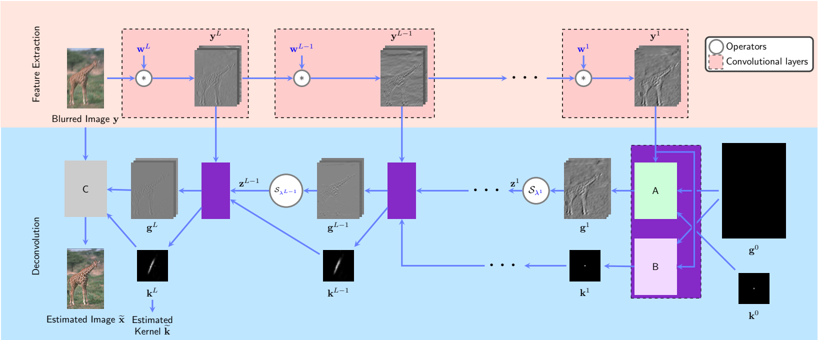


Figure: Neural network constructed through algorithm unrolling and cascaded filtering.