Discriminative Learning in Image Restoration

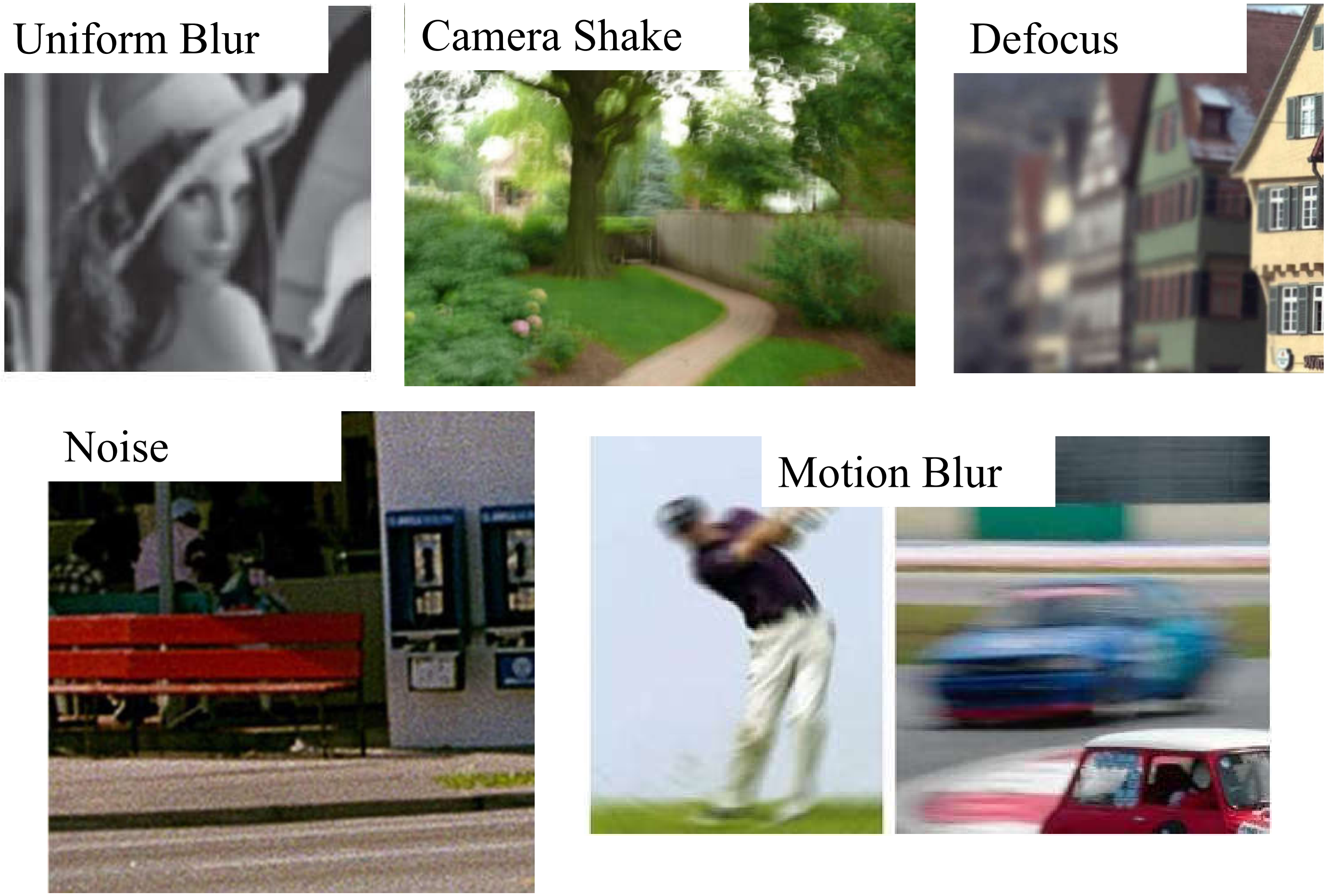
Wangmeng Zuo

Vision Perception and Cognition Group
Harbin Institute of Technology

Content

- Image Restoration
- Can we learn more from a set of degraded/clean image pairs?
- Can we directly learn a plain CNN for image denoising?
- Can we extend denoising CNN for general image restoration?
- Can we learn more for MAP-based restoration: efficiency and flexibility?

Image Restoration



Modeling Image Degradation

- Uniform blur + Noise
  - Space invariant PSF -> convolution

$$\mathbf{y} = \mathbf{x} \otimes \mathbf{k} + e$$

- Nonuniform blur + Noise
  - Image defocus
  - Camera shake

$$\mathbf{y} = \mathbf{A}\mathbf{x} + e$$

- Reflection/background separation

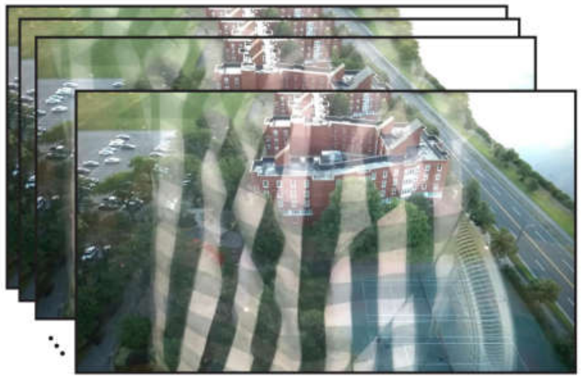
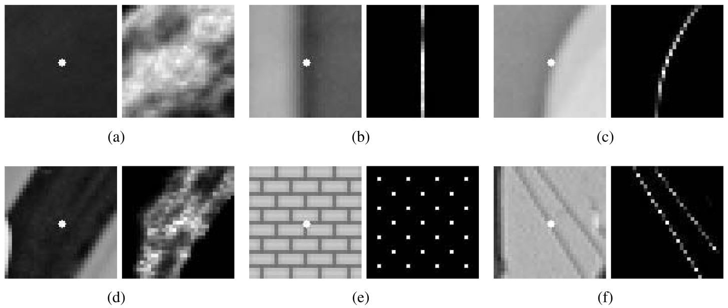
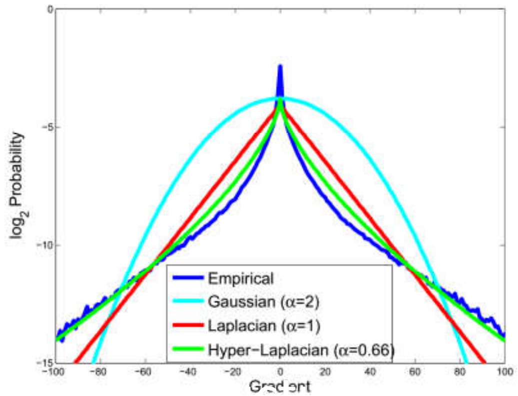
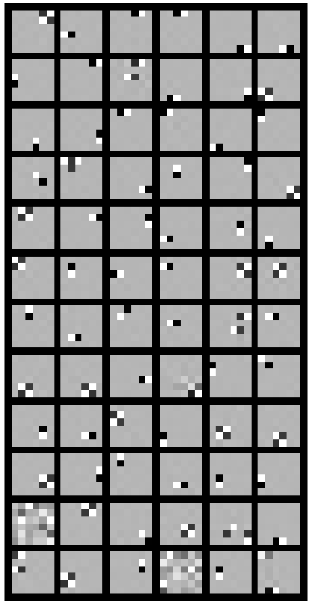


Image Priors

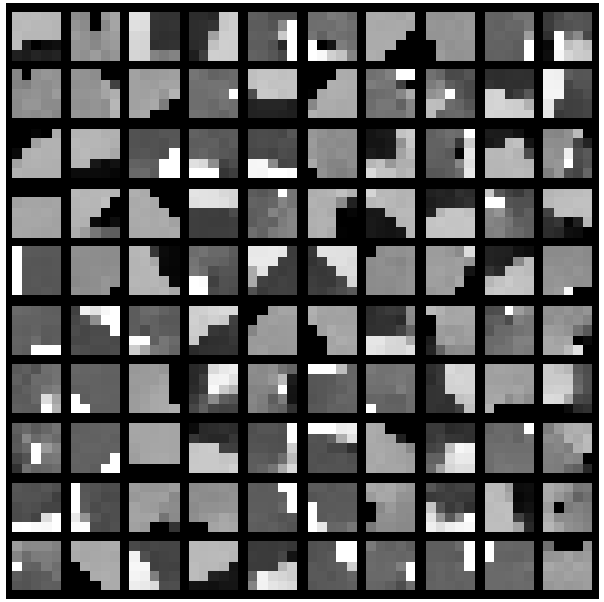
- Gradient-based model
- Dictionary-based model
  - Analysis vs. Synthesis
  - Filter vs. Patch
- Non-local similarity-based models
  - Low rank, group sparsity







Analysis Dictionary



Synthesis Dictionary

MAP Estimation

- Bayes: P(x|y) ∝ P(y|x)P(x)
- MAP Model

$$\mathbf{x} = \arg\min_{\mathbf{x}} -\log P(\mathbf{y} \mid \mathbf{x}) - \log P(\mathbf{x}) \quad \Leftrightarrow \quad \mathbf{x} = \arg\min_{\mathbf{x}} L(\mathbf{x}, \mathbf{y}) + R(\mathbf{x})$$

  - Fidelity term based on degradation model: $L(x,y)$
  - Regularization term based on prior: $R(x)$

- MAP-based restoration: Optimal MSE
  - Degradation model: generally is known in advance
  - Regularizer: Key issue
  - Optimization methods

Content

- Image Restoration
- Can we learn more from a set of degraded/clean image pairs?
  - Discriminative blind deconvolution
- Can we directly learn a plain CNN for image denoising?
- Can we extend denoising CNN for general image restoration?
- Can we learn more for MAP-based restoration: efficiency and flexibility?

Benefit of Task driven discriminative learning

- MAP: Optimal MSE $\mathbf{x} = \arg\min_{\mathbf{x}} L(\mathbf{x}, \mathbf{y}) + R(\mathbf{x})$

- Given a set of clear images, what can we do:
  - Modeling image prior
  - Non-convex
- Given a training set, what can we do:
  - Better optimization

$\Rightarrow$ Joint learning of both model and optimization method

$$\min_{\Theta} \left\| x_i^* - x_i(\Theta) \right\|^2$$

$$s.t.\ x_i(\Theta) = \arg\min_x \ \|x_i - y_i\|^2 + \mathrm{R}(x_i; \Theta)$$

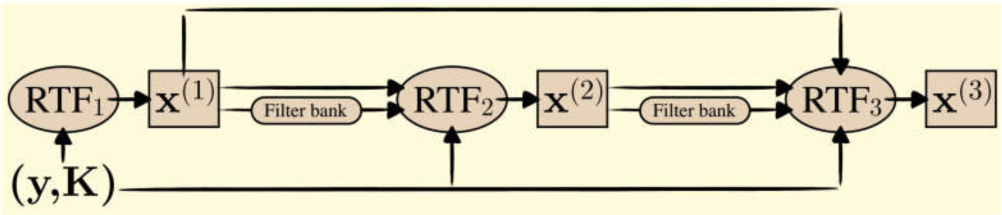
Discriminative non-blind deconvolution (Schmidt et al., 2013)

- The parameters of priors may vary for different tasks, data distributions, and even iterations

$$\min_{\{\Theta_k\}} \left\| x_i^* - x_i^K(\{\Theta_k\}) \right\|^2$$

$$s.t.\ x_i^k(\Theta_k) = F(x_i^k, y; \Theta_k(\mathrm{y}, \{x_i^1, \dots, x_i^{k-1}\}))$$

- Parameter learning: Random Tree Field (RTF)
    - But remains inefficient



Cascade of shrinkage fields (Schmidt & Roth, CVPR 2014)

- Rather than simply learning image priors, unfold the inference process as an iterative algorithm, where stage-wise model parameters are learned from training data
- Model:

$$E(\mathbf{x}|\mathbf{y}) = \frac{\lambda}{2}\|\mathbf{y} - \mathbf{K}\mathbf{x}\|^2 + \sum_{i=1}^{N}\sum_{c\in\mathcal{C}} \rho_i(\mathbf{f}_i^{\mathsf{T}}\mathbf{x}_{(c)})$$

- Half-Quadratic Optimization

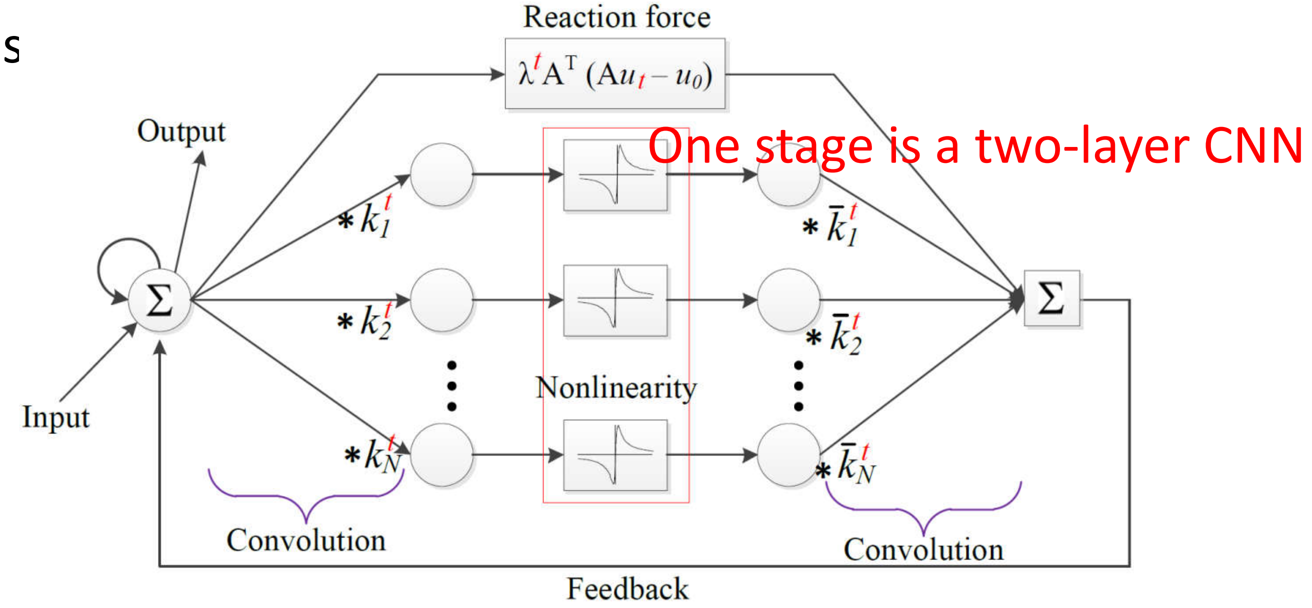
$$E_\beta(\mathbf{x},\mathbf{z}|\mathbf{y}) = \frac{\lambda}{2}\|\mathbf{y}-\mathbf{K}\mathbf{x}\|^2 + \sum_{i=1}^{N}\sum_{c\in\mathcal{C}}\left(\frac{\beta}{2}\left(\mathbf{f}_i^{\mathsf{T}}\mathbf{x}_{(c)} - z_{ic}\right)^2 + \rho_i(z_{ic})\right).$$

Trainable Nonlinear Reaction Diffusion

- A nonlinear reaction and diffusion model for image restoration

$$\min_{\mathbf{x}} \frac{\lambda}{2}\|\mathbf{A}\mathbf{x}-\mathbf{y}\|^2 + \sum_{k=1}^{K}\sum_{p=1}^{P}\phi_k\left(\left(\mathbf{f}_k * \mathbf{x}\right)_p\right)$$

Do a gradient des

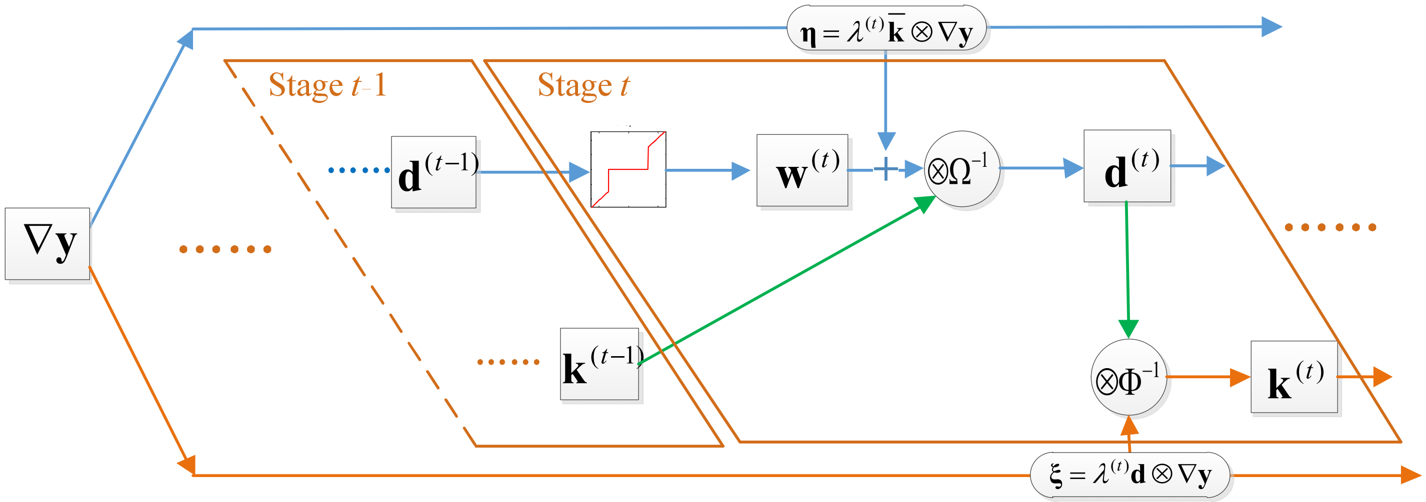


(Chen et al., CVPR 2015)

Learning Iteration-wise GST for blind deconvolution

- Blind deconvolution: both blur kernel and latent clear image are unknown.
  - More challenging and ill-posed than non-blind deconvolution.

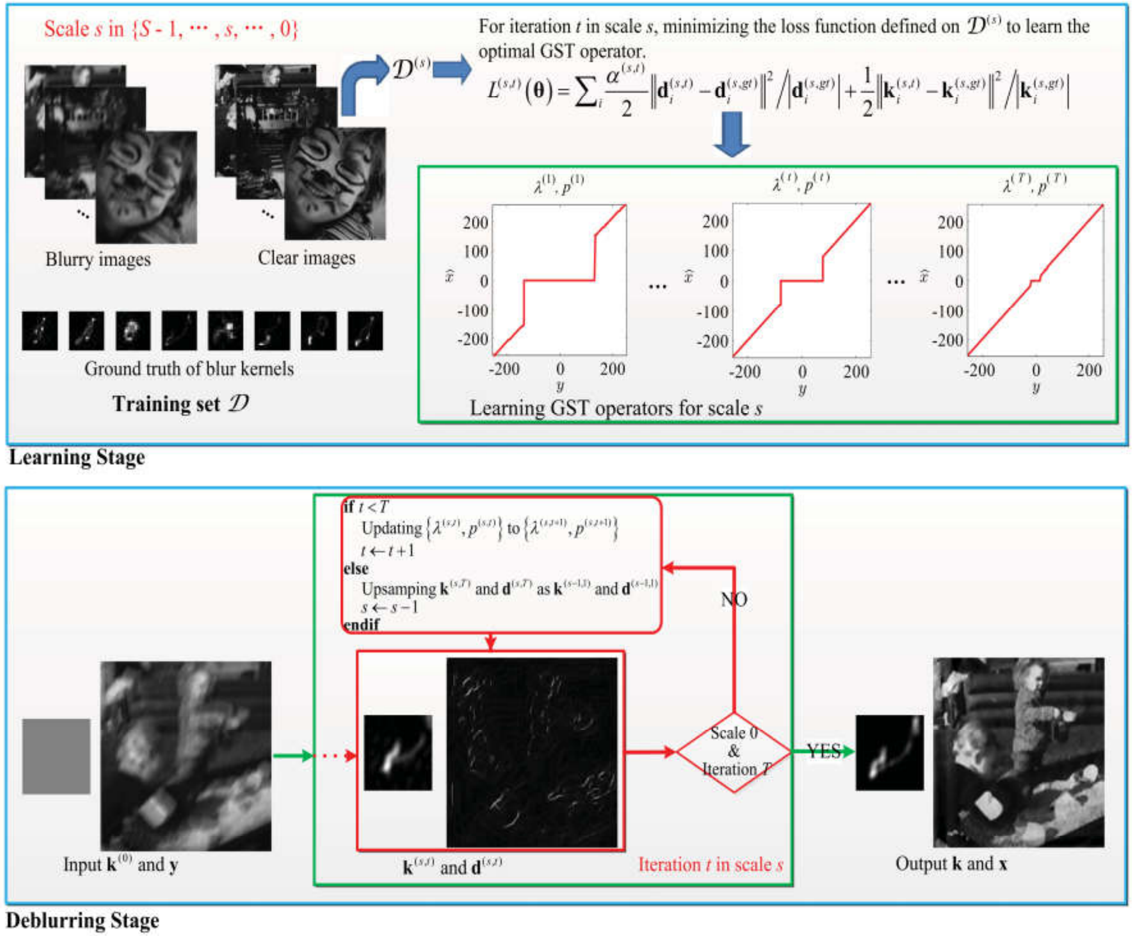
- Our blind deconvolution model:



(Zuo et al., CVPR 2015 & TIP 2016)

Illustration of the framework (Zuo et al., CVPR 2015, TIP 2016)

- Training:
  - Unfold the inference process
  - Empirical evidence
  - Iteration-wise loss
- Test:
  - Iteration-wise GST



Discriminative Learning Model

- Bi-level optimization model:

$$L^{(t)}(\boldsymbol{\theta}) = \sum_{i=1}^{N} \frac{\alpha^{(t)}}{2} \left\| \mathbf{d}_i^{(t)} - \mathbf{d}_i^{gt} \right\|^2 \Big/ \left| \mathbf{d}_i^{gt} \right| + \frac{1}{2} \left\| \mathbf{k}_i^{(t)} - \mathbf{k}_i^{gt} \right\|^2 \Big/ \left| \mathbf{k}_i^{gt} \right|$$

s.t.

$$(\mathbf{d}_i^{(t)}, \mathbf{k}_i^{(t)}) = \arg\min_{\mathbf{d},\mathbf{k}} \frac{\lambda^{(t)}}{2\sigma_n^2} \left\| \mathbf{k} \otimes \mathbf{d} - \nabla \mathbf{y} \right\|^2 + \left\| \mathbf{d} \right\|_{p^{(t)}}^{p^{(t)}} + \mu \left\| \mathbf{k} \right\|_{0.5}^{0.5}$$

$$\nabla_h \mathbf{d}_v = \nabla_v \mathbf{d}_h, \sum_i k_i = 1, k_i \geq 0, \forall i$$

Unfolding the iterative solution to lower optimization problem

- Fixing **k**, update **d** with one-step hybrid ALM

$$\min_{\mathbf{d}} \frac{\lambda^{(t)}}{2\sigma_n^2}\left\|\mathbf{k}^{(t-1)} \otimes \mathbf{d} - \nabla\mathbf{y}\right\|^2 + \left\|\mathbf{d}\right\|_{p^{(t)}}^{p^{(t)}} \text{ s.t. } \nabla_h \mathbf{d}_v = \nabla_v \mathbf{d}_h$$

  - One-step generalized shrinkage / thresholding (GST) operator (Zuo et al., 2013)

- Fixing **d**, update **k** with one-step ALM

$$\min_{\mathbf{k}} \frac{\lambda^{(t)}}{2\sigma_n^2}\left\|\mathbf{k} \otimes \mathbf{d}^{(t)} - \nabla\mathbf{y}\right\|^2 + \mu\left\|\mathbf{k}\right\|_{0.5}^{0.5}$$
$$\text{s.t.} \sum_i k_i = 1, k_i \geq 0, \forall i$$

Discriminative learning

- Partial derivative

$$\frac{\partial L_i^{(t)}}{\partial \boldsymbol{\theta}} = \left(\alpha^{(t)} \frac{\partial L_{\mathbf{d}_i}^{(t)}}{\partial p}, \alpha^{(t)} \frac{\partial L_{\mathbf{d}_i}^{(t)}}{\partial \lambda} + \frac{\partial L_{\mathbf{k}_i}^{(t)}}{\partial \lambda} \right)$$

- Optimization:
  - Gradient-based L-BFGS
  - Stage-wise learning

Incorporating Empirical Evidence and Extension

- Reduce the learned parameters to only λ and *p*
- Estimate **k** using a smaller *p* and estimate **x** using a relatively higher *p* (Xu et al., 2012)
  - Extend the GST operator to p < 0

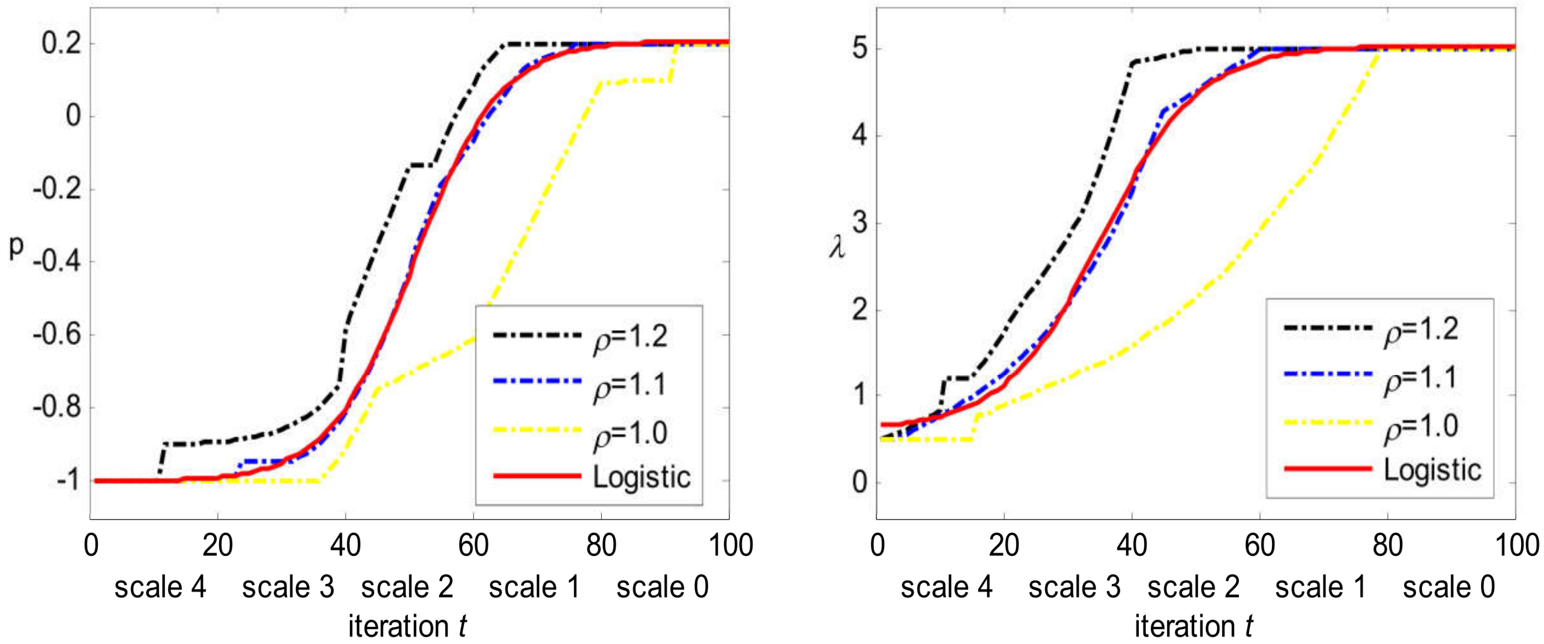
$$\hat{x} = \begin{cases} 0, & \text{if } |y| \leq \tau_p^{GST}(\lambda) \\ \operatorname{sgn}(y)\, S_p^{GST}(|y|;\lambda), & \text{if } |y| > \tau_p^{GST}(\lambda) \end{cases}$$

  - Non-decreasing constraints on λ and *p*

$$\lambda^{(s,t+1)} \geq \lambda^{(s,t)} \qquad p^{(s,t+1)} \geq p^{(s,t)}$$

- Multi-scale scheme (Krisnan et al., 2012)

Results: the learned iteration-wise parameters



Results (Training on Levin and test on Sun)

	PSNR	SSIM	Error Ratio	Time
Known **k**	32.35	0.9536	1.0000	—
Krishnan et al. [15]	22.76	0.8136	6.8351	159.29
Cho & Lee [12]	26.13	0.8624	5.0731	10.518
Levin et al. [2]	24.64	0.8606	4.5798	518.59
Xu & Jia [16]	28.11	0.9016	3.2843	6.2940
Sun et al. [21]	29.32	0.9200	2.4036	3911.1
Ours(-1)	28.03	0.9032	3.2083	99.193
Ours(0.2)	28.58	0.9152	2.9802	98.231
Ours	29.35	0.9231	2.3901	98.071

	PSNR	SSIM	Error Ratio	Time
Known **k**	32.31	0.9385	1.0000	—
Krishnan et al. [15]	28.26	0.8547	2.3746	8.9400
Cho & Lee [12]	28.83	0.8801	1.5402	1.3951
Levin et al. [2]	28.79	0.8922	1.5592	78.263
Xu & Jia [16]	29.45	0.9000	1.4071	1.1840
Sun et al. [21]	30.85	0.9191	1.2244	191.03
Ours(-1)	28.63	0.8899	1.6235	10.403
Ours(0.2)	29.08	0.9057	1.4181	10.830
Ours(Logistic)	30.89	0.9214	1.2228	10.549
Ours(Re-train)	30.80	0.9188	1.2257	10.981
Ours	30.91	0.9238	1.2210	10.998

Results

Ground truth | Cho & Lee [12] | Krishnan et al. [15] | Levin et al. [2] | Xu & Jia [16] | Sun et al. [21] | Ours

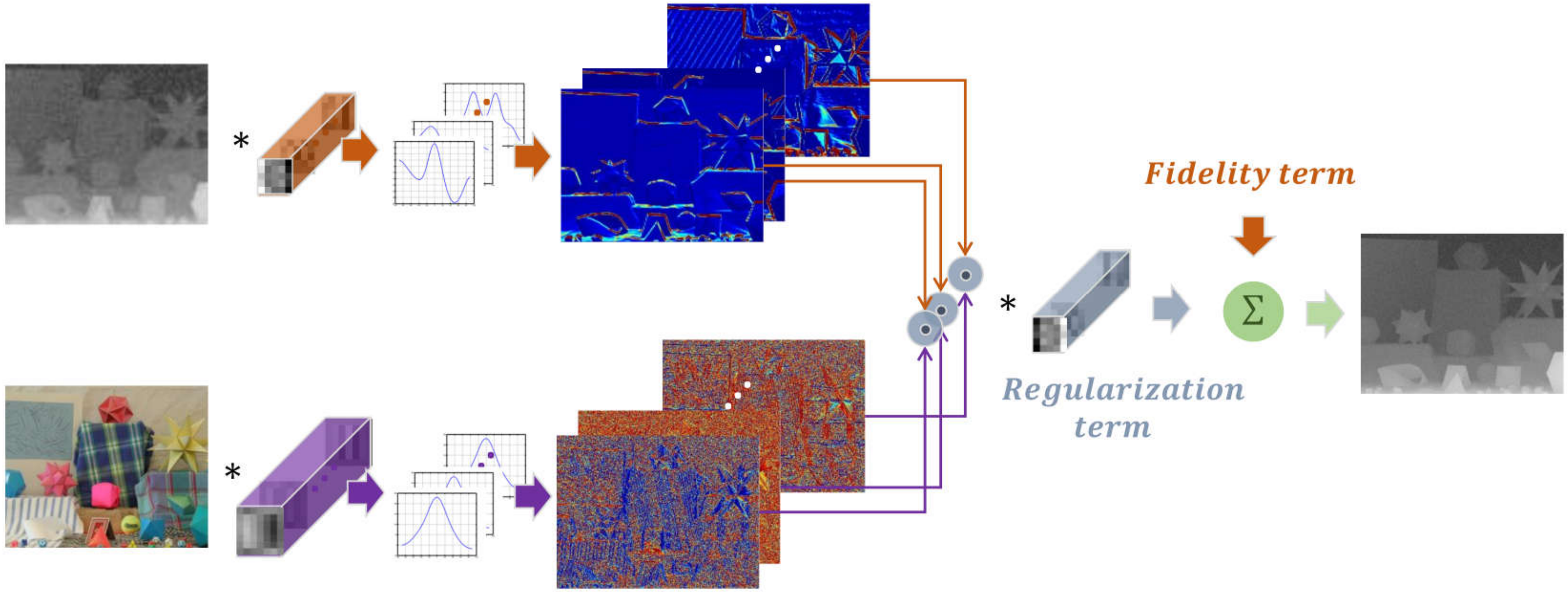
Blurry input | Xu & Jia [16] | Sun et al. [21] | Ours

Learning Dynamic Guidance for Depth Image Enhancement (CVPR 2017)

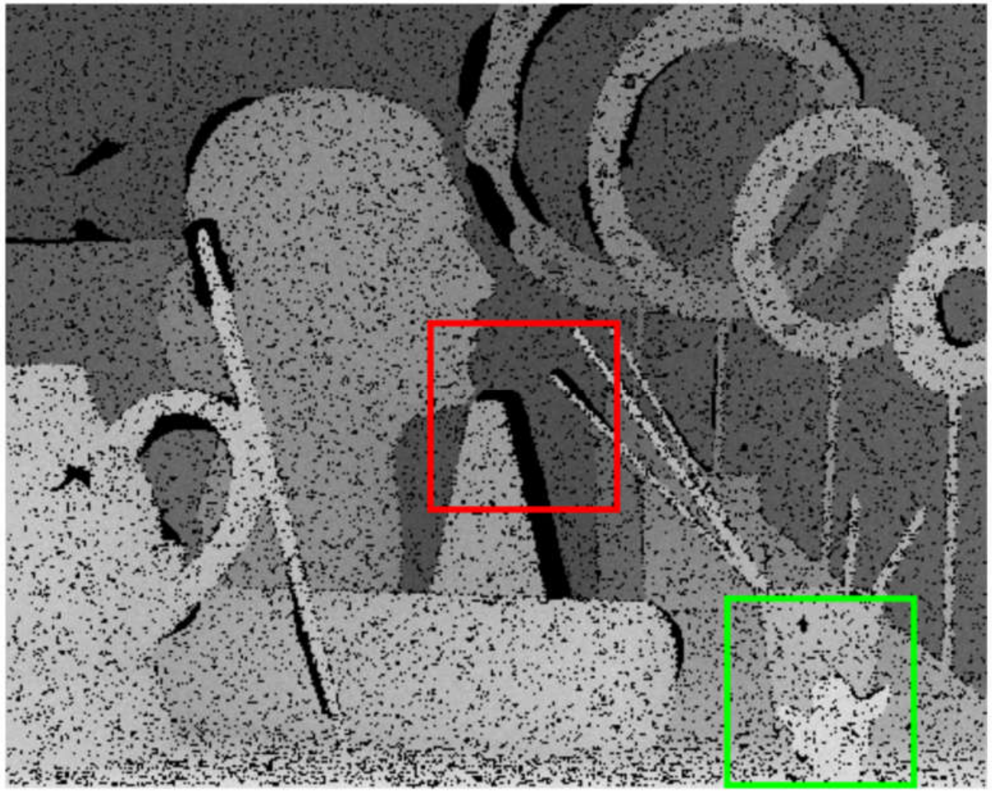
- Task driven learning
- Dynamic guidance

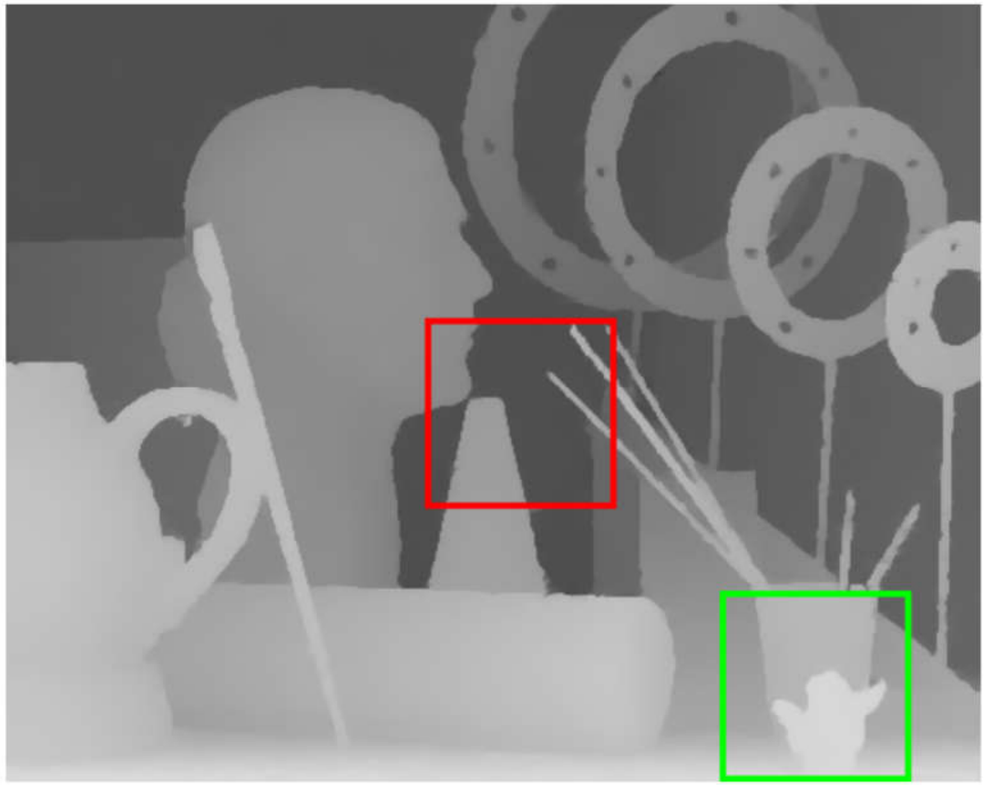
$$\{\rho_l^*, \boldsymbol{\beta}_l^*, \boldsymbol{k}_l^*\}_{l=1}^{L} = \arg \min_{\{\rho_l, \boldsymbol{\beta}_l, \boldsymbol{k}_l\}_{l=1}^{L}} \sum_{s=1}^{S} \|\boldsymbol{x}_{gt}^s - \boldsymbol{x}^s\|_2^2$$

$$s.t.\ \boldsymbol{x}^s = \arg\min_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{y}^s) + \sum_l \sum_i w_{l,i}(\boldsymbol{g}^s)\rho_l((\boldsymbol{k}_l \otimes \boldsymbol{x})_i)$$



Results





Results

	NYU		
	$\times 4$	$\times 8$	$\times 16$
MRF [4]	4.29	7.54	12.32
GF [13]	4.04	7.34	12.23
JBU [16]	2.31	4.12	6.98
TGV [6]	3.83	6.46	13.49
Park [28]	3.00	5.05	9.73
SDF [11]	3.04	5.67	9.97
DJF [22]	1.97	3.39	5.63
Ours	**1.56**	**2.99**	**5.24**

Upsampling on NYU

	Lu et al. [24]	Shen et al. [34]	Ours
Art	6.77	5.65	**4.96**
Books	2.24	2.24	**1.66**
Moebius	2.18	2.27	**1.76**

Denoising and inpainting

Content

- Image Restoration
- Can we learn more from a set of degraded/clean image pairs?
- Can we directly learn a plain CNN for image denoising?
- Can we extend denoising CNN for general image restoration?
- Can we learn more for MAP-based restoration: efficiency and flexibility?

CNN-based image denoising

- Understanding and extension of the existing image model
- Benefit from the existing deep CNN model and architecture
  - VGGnet, ResNet, GoogLeNet
- Better learning/regularization strategies
  - ReLU
  - Batch Normalization
- Efficiency in training and testing

Connect TNRD with ResNet

Revisit the basic model of TNRD

$$\min_{\mathbf{x}} D(\mathbf{y}-\mathbf{x}) + \lambda \sum_{k=1}^{K} \sum_{p=1}^{P} \phi_k\left((\mathbf{f}_k * \mathbf{x})_p \right)$$

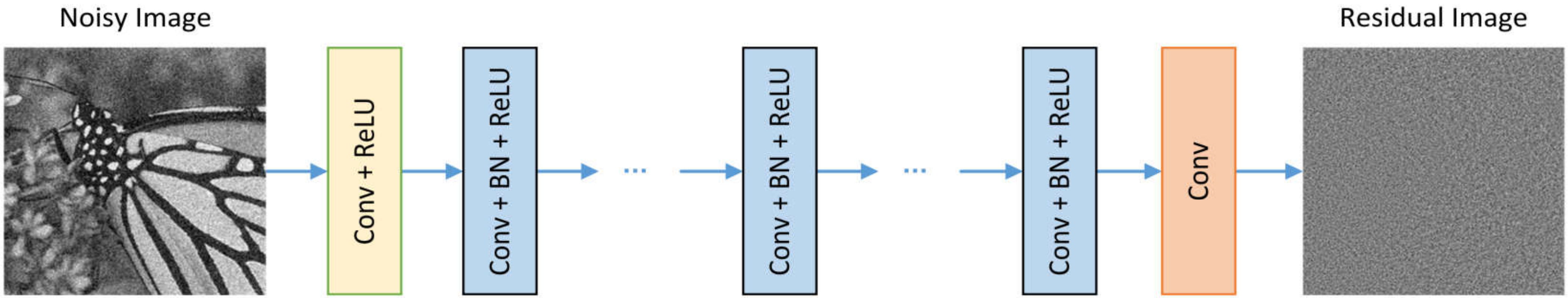
For the first stage with **x** initialized as **y** (**z** = **y** - **x**)

$$\mathbf{x}_1 = \mathbf{y} - \alpha \sum_{k=1}^{K} \overline{\mathbf{f}_k} * \rho_k(\mathbf{f}_k * \mathbf{x}) - \alpha \left. \frac{\partial D(\mathbf{z})}{\partial \mathbf{z}} \right|_{\mathbf{z}=0}$$

$\left. \frac{\partial D(\mathbf{z})}{\partial \mathbf{z}} \right|_{\mathbf{z}=0} = 0$ indicates that TNRD stage is a residual learning

$$\mathbf{v}_1 = \mathbf{x}_1 - \mathbf{y} = \alpha \sum_{k=1}^{K} \overline{\mathbf{f}_k} * \phi_k(\mathbf{f}_k * \mathbf{y})$$

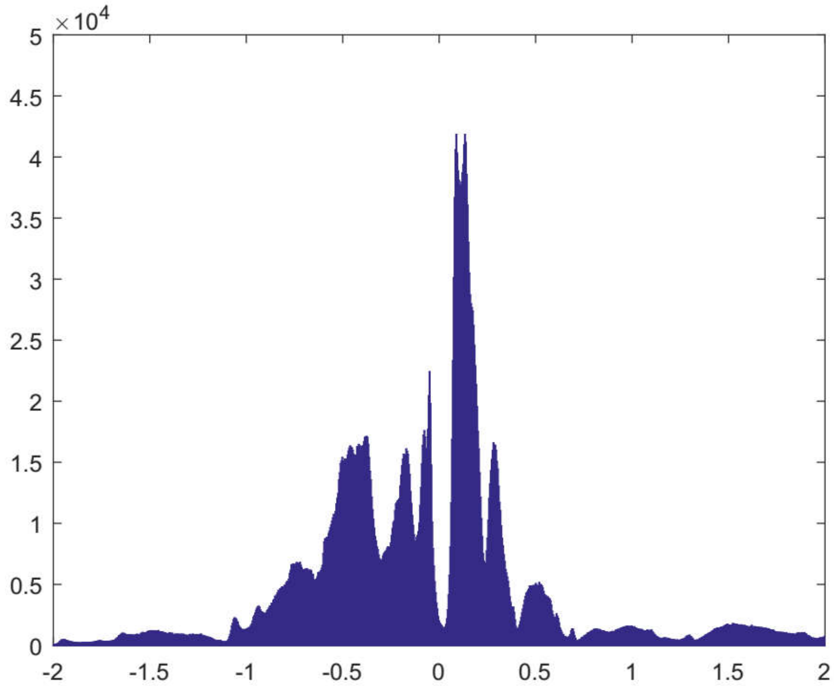
DnCNN: Extension of TNRD (Zhang et al., TIP 2017)



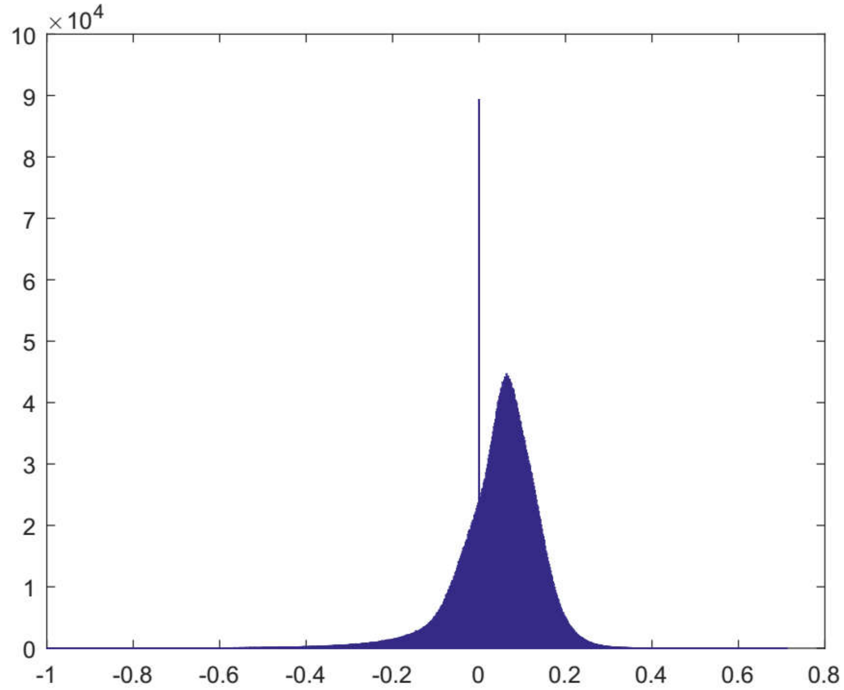
- Replacing the influence function with ReLU
- Increasing the CNN depth
- Incorporating with batch normalization

Why we use BN

- Layer 6, Monarch



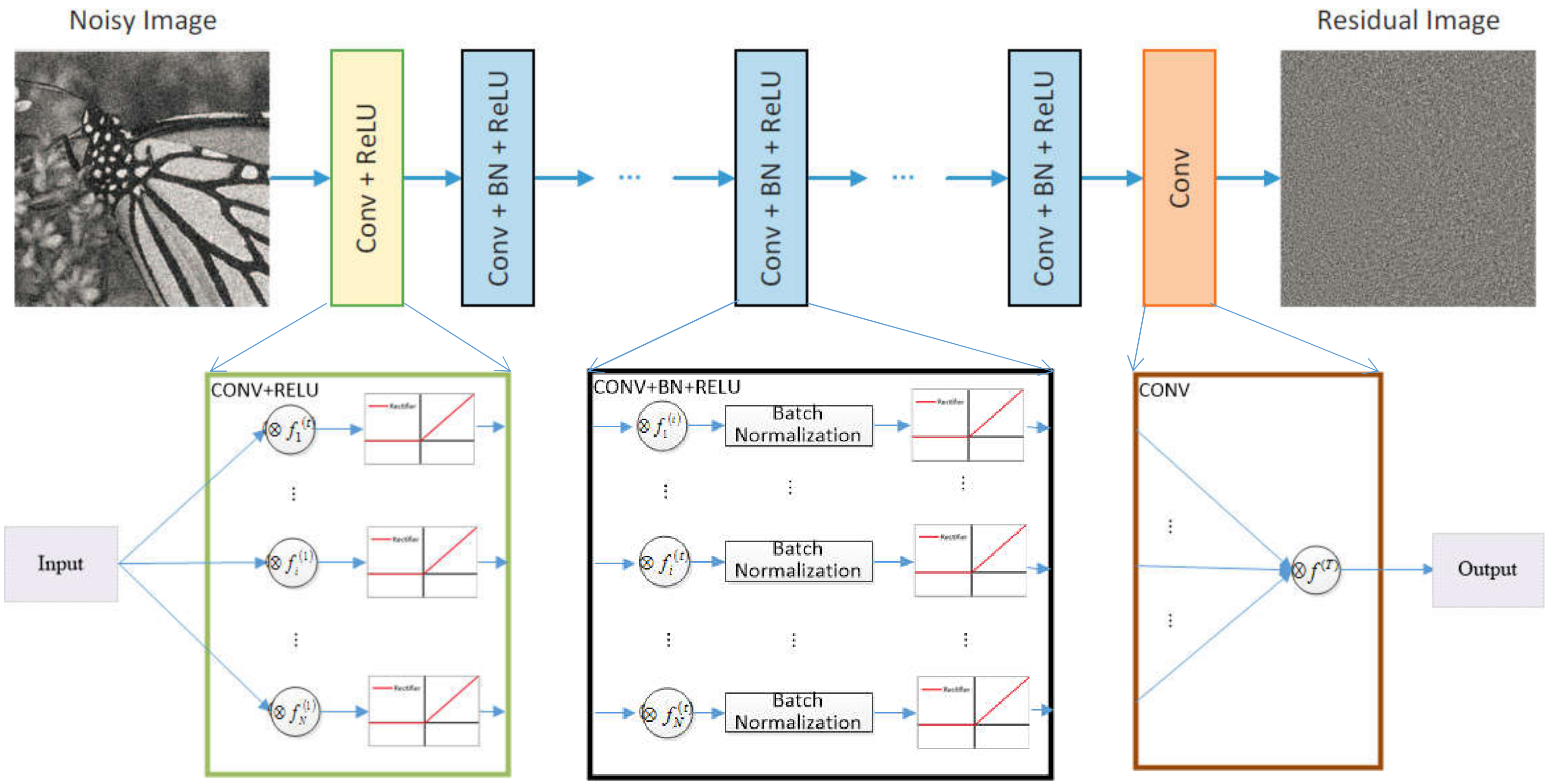
w/o BN



with BN

Deep CNN for Denoising: DnCNN

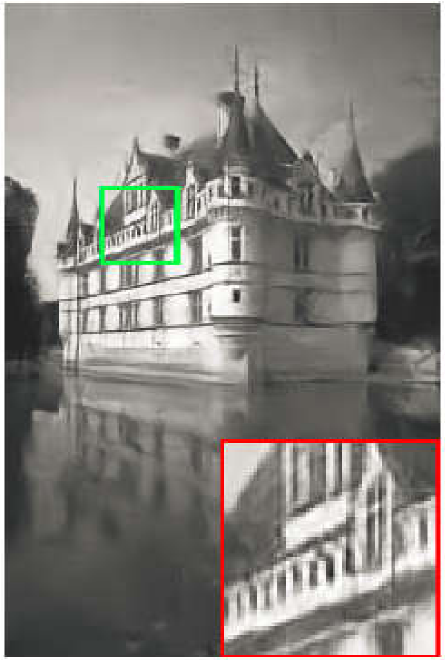
Even for non-Gaussian noise, if $\frac{\partial \mathrm{D}(\mathbf{n})}{\partial \mathbf{n}}|_{\mathbf{n}=\mathbf{n}_0} = 0$ holds, DnCNN also works well.



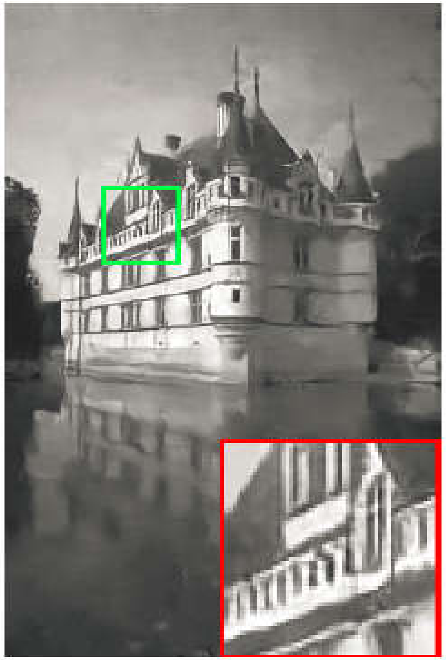
DnCNN for denoising

Results on BSD68 dataset

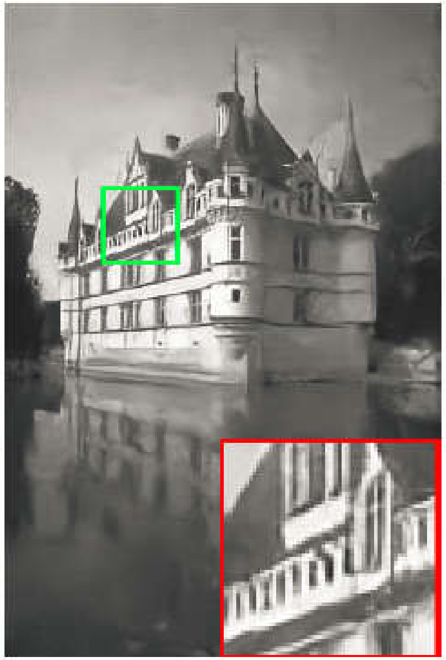
Methods	BM3D	WNNM	EPLL	MLP	CSF	TNRD	DnCNN-S	DnCNN-B
$\sigma = 15$	31.07	31.37	31.21	-	31.24	31.42	**31.73**	31.61
$\sigma = 25$	28.57	28.83	28.68	28.96	28.74	28.92	**29.23**	29.16
$\sigma = 50$	25.62	25.87	25.67	26.03	-	25.97	**26.23**	**26.23**



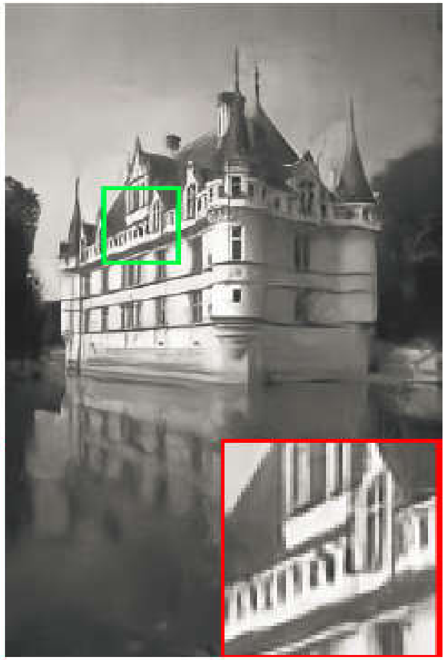
(e) MLP / 26.54dB



(f) TNRD / 26.59dB



(g) DnCNN-S / 26.90dB



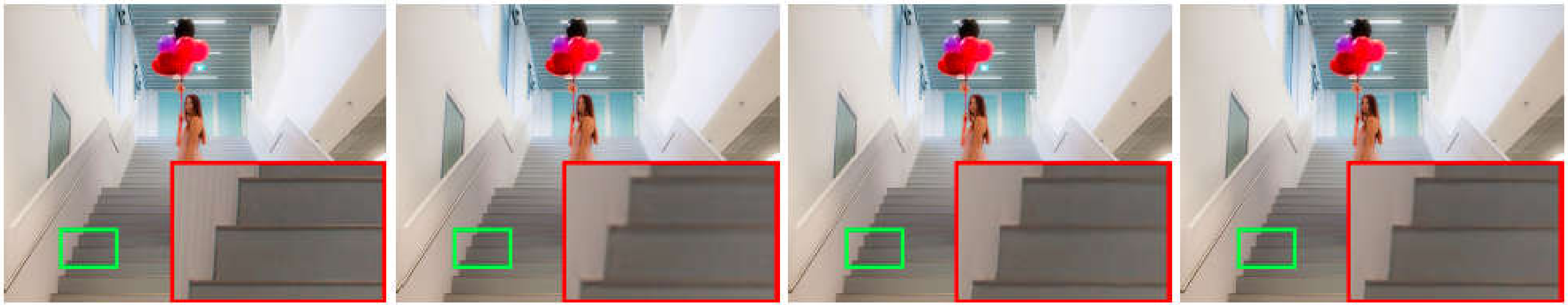
(h) DnCNN-B / 26.92dB

Denoising on real images

DnCNN for SISR

Averaged PSNR/SSIM comparison

Single Image Super-Resolution				
Dataset	Upscaling Factor	TNRD	VDSR	DnCNN-3
		PSNR / SSIM	PSNR / SSIM	PSNR / SSIM
Set5	2	36.86 / 0.9556	37.56 / **0.9591**	**37.58** / 0.9590
	3	33.18 / 0.9152	33.67 / 0.9220	**33.75** / **0.9222**
	4	30.85 / 0.8732	31.35 / **0.8845**	**31.40** / **0.8845**
Set14	2	32.51 / 0.9069	33.02 / **0.9128**	**33.03** / **0.9128**
	3	29.43 / 0.8232	29.77 / 0.8318	**29.81** / **0.8321**
	4	27.66 / 0.7563	27.99 / 0.7659	**28.04** / **0.7672**

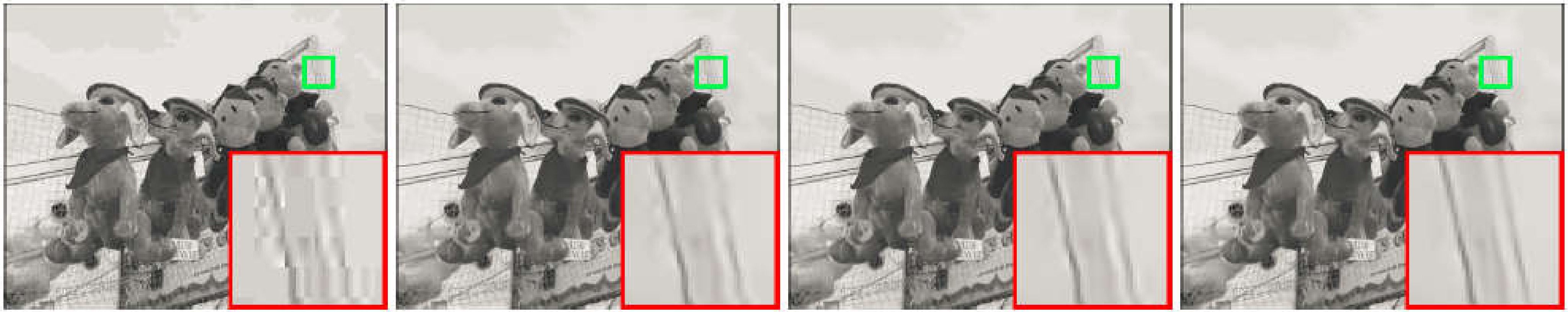


(a) Ground-truth (b) TNRD / 32.00dB (c) VDSR / 32.58dB (d) DnCNN-3 / 32.73dB

DnCNN for JPEG deblocking

Averaged PSNR/SSIM comparison

JPEG Image Deblocking				
Dataset	Quality Factor	AR-CNN	TNRD	DnCNN-3
		PSNR / SSIM	PSNR / SSIM	PSNR / SSIM
Classic5	10	29.03 / 0.7929	29.28 / 0.7992	**29.40 / 0.8026**
	20	31.15 / 0.8517	31.47 / 0.8576	**31.63 / 0.8610**
	30	32.51 / 0.8806	32.78 / 0.8837	**32.91 / 0.8861**
	40	33.34 / 0.8953	-	**33.77 / 0.9003**



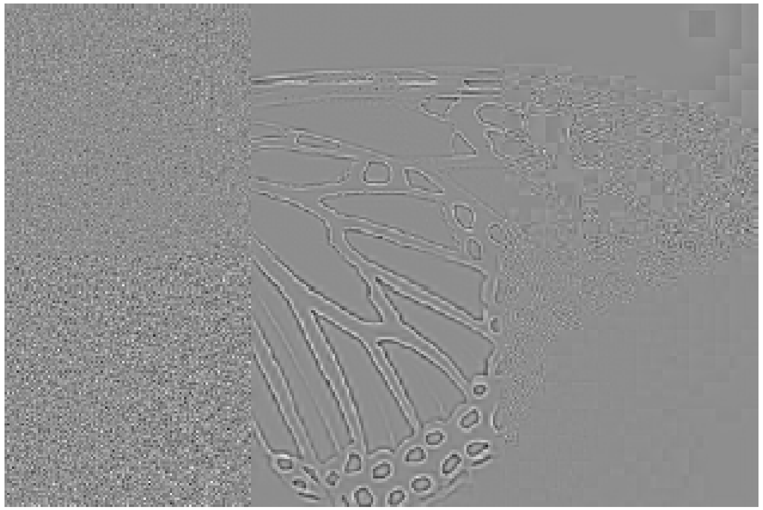
(a) JPEG / 28.10dB (b) AR-CNN / 28.85dB (c) TNRD / 29.54dB (d) DnCNN-3 / 29.70dB

DnCNN for hybrid noises

- DnCNN is robust to hybrid noises

noise level | SR factor | JPEG factor

15 | 2 | 10



25 | 3 | 30

NTIRE 2017 Challenge on SISR

		Track 1: bicubic downscaling						Track 2: unknown downscaling					
		×2		×3		×4		×2		×3		×4	
Team	User	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
SNU_CVLab [1]	limbee	$34.93_{(1)}$	0.948	$31.13_{(1)}$	0.889	$26.91^{*}_{(14)}$	0.752*	$34.00_{(1)}$	0.934	$30.78_{(1)}$	0.881	$28.77_{(1)}$	0.826
SNU_CVLab [2]	sanghyun	$34.83_{(2)}$	0.947	$31.04_{(2)}$	0.888	$29.04_{(1)}$	0.836	$33.86_{(2)}$	0.932	$30.67_{(2)}$	0.879	$28.62_{(2)}$	0.821
helloSR	sparkfirer	$34.47_{(4)}$	0.944	$30.77_{(4)}$	0.882	$28.82_{(3)}$	0.830	$33.67_{(3)}$	0.930	$30.51_{(3)}$	0.876	$28.54_{(3)}$	0.819
Lab402	iorism	$34.66_{(3)}$	0.946	$30.83_{(3)}$	0.884	$28.83_{(2)}$	0.830	$32.92_{(7)}$	0.921	$30.31_{(4)}$	0.871	$28.14_{(6)}$	0.807
VICLab	SJChoi	$34.29_{(5)}$	0.943	$30.52_{(5)}$	0.880	$28.55_{(5)}$	0.845						
UIUC-IFP	fyc0624	$34.19_{(6)}$	0.942	$30.44_{(7)}$	0.877	$28.49_{(6)}$	0.821	$28.54_{(14)}$	0.840	$28.11_{(14)}$	0.816	$24.96_{(15)}$	0.717
HIT-ULSee	chenyunjin	$34.07_{(7)}$	0.941	$30.21_{(9)}$	0.871	$28.49_{(6)}$	0.822	$33.40_{(4)}$	0.927	$30.21_{(6)}$	0.871	$28.30_{(4)}$	0.812
I hate mosaic	tzm1003306213	$34.05_{(8)}$	0.940	$30.47_{(6)}$	0.878	28.59	0.824						

HIT-ULSee solution is ***the most efficient***, it gives the best trade-off between runtime and quality of the results.

Team	Track 1: bicubic downscaling			Track 2: unknown down						
	×2	×3	×4	×2	×3					
SNU_CVLab [1]	67.240	28.720	20.050	8.778	4.717	2.602	Torch (Lua)		GTX TITAN X	36 ResBlocks
SNU_CVLab [2]	14.070	7.340	5.240	4.600	2.310	1.760	Torch (Lua)		GTX TITAN X	80 ResBlocks
helloSR	27.630	27.970	18.470	11.540	19.260	15.360	Torch (Lua)		GTX TITAN X	stacked ResNets
Lab402	4.080	5.120	5.220	4.120	1.880	1.120	Matconvnet+Matlab		GTX 1080ti	wavelet+41 conv. layers
VICLab	0.539	0.272	0.186				Matconvnet		TITAN X Pascal	22 layers
UIUC-IFP	1.683	1.497	1.520	1.694	1.474	1.523	TensorFlow+Python		8×GPUs	6+4 ResBlocks
HIT-ULSee	0.370	0.160	0.100	0.370	0.160	0.100	Matlab		Titan X Pascal	20 (sub-pixel) layers
I hate mosaic	10.980	8.510	8.150				TensorFlow+Python		Titan X Maxwell	Joint ResNets

Content

- Image Restoration
- Can we learn more from a set of degraded/clean image pairs?
- Can we directly learn a plain CNN for image restoration?
- Can we extend denoising CNN for general image restoration?
- Can we learn more for MAP-based restoration: efficiency and flexibility?

Revisting the MAP framework

- MAP

$$\mathbf{x} = \arg\min_{\mathbf{x}} \frac{1}{2}\left\|\mathcal{A}(\mathbf{x}) - \mathbf{y}\right\|_2^2 + \Phi(\mathbf{x})$$

  - The degradation model is known
  - Once the regularization term (i.e. prior) is given, optimization can be used to solve any image restoration task.

- Then, the problem becomes:
  - Learn image prior from a set of high-quality images
  - Desigin proper optimization methods to solve the MAP model

Let's take the optimization into account

$$\mathbf{x} = \arg\min_{\mathbf{x}} \frac{1}{2}\left\|\mathcal{A}(\mathbf{x}) - \mathbf{y}\right\|_2^2 + \Phi(\mathbf{x})$$

- Alternating direction method of multipliers
  - Reformulation

$$\mathcal{L}_\mu(\mathbf{x}, \mathbf{z}) = \frac{1}{2}\|\mathbf{y} - \mathbf{H}\mathbf{x}\|^2 + \lambda\Phi(\mathbf{z}) + \frac{\mu}{2}\|\mathbf{z} - \mathbf{x}\|^2$$

  - Optimization

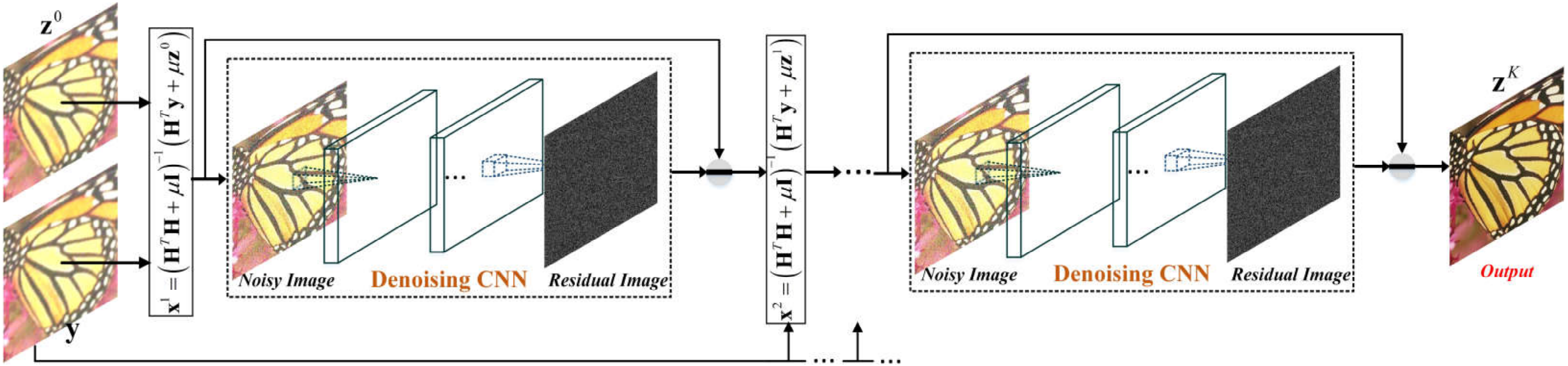
$$\begin{cases} \mathbf{x}_{k+1} = \arg\min_{\mathbf{x}} \|\mathbf{y} - \mathbf{H}\mathbf{x}\|^2 + \mu\|\mathbf{x} - \mathbf{z}_k\|^2 \\ \mathbf{z}_{k+1} = \arg\min_{\mathbf{z}} \frac{\mu}{2}\|\mathbf{z} - \mathbf{x}_{k+1}\|^2 + \lambda\Phi(\mathbf{z}) \end{cases}$$

$$\mathbf{z}_{k+1} = \arg\min_{\mathbf{z}} \frac{1}{2(\sqrt{\lambda/\mu})^2}\|\mathbf{x}_{k+1} - \mathbf{z}\|^2 + \Phi(\mathbf{z})$$

  - Actually, what we need is not the explicit form of Φ(x) but a denoiser

$$\mathbf{z}_{k+1} = Denoiser(\mathbf{x}_{k+1}, \sqrt{\lambda/\mu})$$

Incorporating denoising CNN with ADMM



ISTA and FISTA

- ISTA:

$$\mathbf{x}^{k+0.5} = \mathbf{x}^k - \frac{1}{L}\mathbf{A}^T(\mathbf{A}\mathbf{x}^k - \mathbf{y})$$

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \frac{L}{2}\left\|\mathbf{x} - \mathbf{x}^{k+0.5}\right\|_2^2 + \Phi(\mathbf{x})$$

- FISTA

$$t_{k+1} = \frac{1 + \sqrt{1 + 4t_k^2}}{2}$$

$$\mathbf{y}_{k+1} = \mathbf{x}_k + \left(\frac{t_k - 1}{t_{k+1}}\right)(\mathbf{x}_k - \mathbf{x}_{k-1})$$

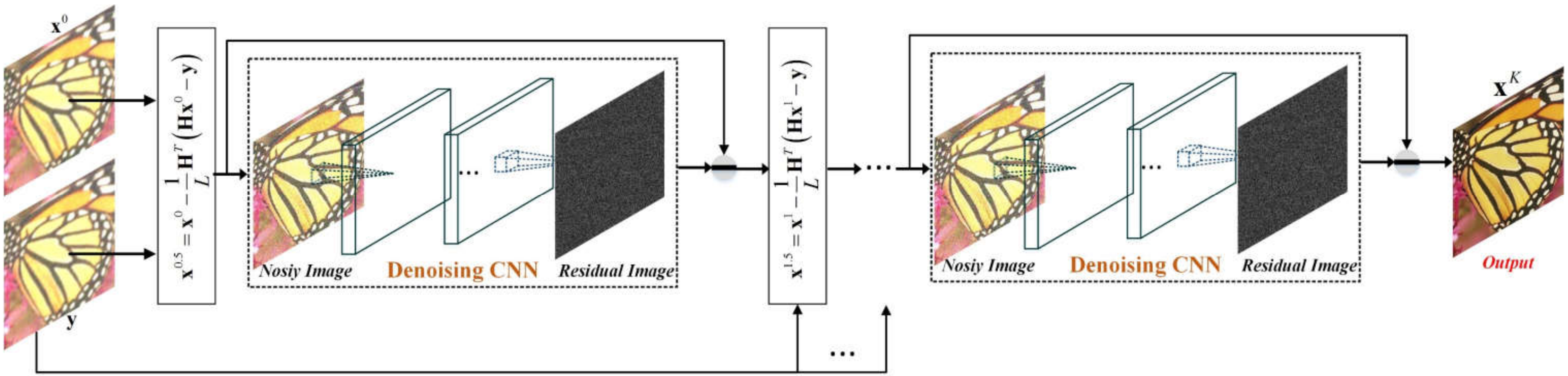
$$\mathbf{x}^{k+0.5} = \mathbf{y}_{k+1} - \frac{1}{L}\mathbf{A}^T(\mathbf{A}\mathbf{y}_{k+1} - \mathbf{y})$$

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \frac{L}{2}\left\|\mathbf{x} - \mathbf{x}^{k+0.5}\right\|_2^2 + \Phi(\mathbf{x})$$

- Also, what we need is not the explicit form of $\Phi(\mathbf{x})$ but a denoiser

$$\mathbf{z}_{k+1} = Denoiser(\mathbf{x}_{k+1}; \sqrt{1/L})$$

Incorporating denoising CNN with ISTA



And we have two choices

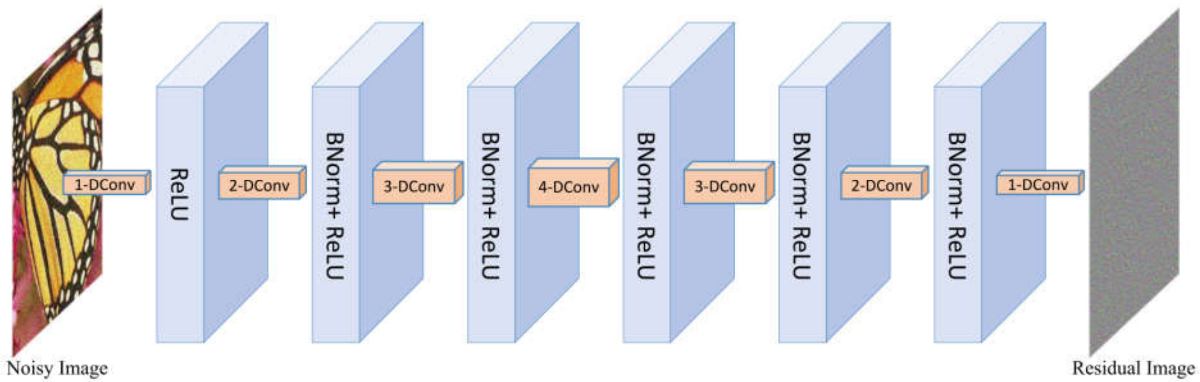
- Choice 1: End-to-end training of ADMM-CNN or ISTA-CNN
    - When the optimization method is changed, the network should be re-trained.
    - When the restoration task is changed (e.g. from deblurring to super-resolution), the network might also be re-trained.

- Choice 2:
    - Train a set of CNN denoisers for a set of noise levels
    - Deploy the denoiser CNNs to any models and tasks as we need

    - Flexible, no re-training is required.

Denoising CNNs (CVPR 2017)

- 25 denoising CNNs for noise level [0. 50]



  - Residual learning + BN
  - Dilated filter

  - Code: https://github.com/cszn/ircnn

Image Denoising

- BSD68: gray

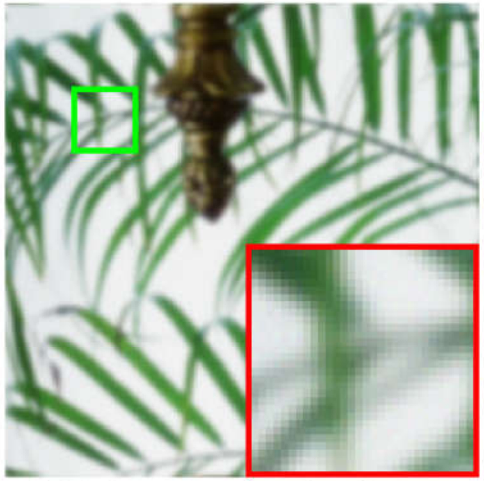
Methods	BM3D	WNNM	TNRD	MLP	Proposed
$\sigma = 15$	31.07	31.37	31.42	-	31.63
$\sigma = 25$	28.57	28.83	28.92	28.96	29.15
$\sigma = 50$	25.62	25.87	25.97	26.03	26.19

- BSD68: color

Noise Level	5	15	25	35	50
CBM3D	40.24	33.52	30.71	28.89	27.38
Proposed	40.36	33.86	31.16	29.50	27.86

Image Deblurring

Methods	σ	*C.man*	*House*	*Lena*	*Monar.*	*Leaves*	*Parrots*
Gaussian blur with standard deviation 1.6							
IDDBM3D	2	27.08	32.41	30.28	27.02	26.95	30.15
NCSR		27.99	33.38	30.99	28.32	27.50	30.42
MLP		27.84	33.43	31.10	28.87	28.91	31.24
Proposed		28.12	33.80	31.17	30.00	29.78	32.07
Kernel 1 (19×19) [38]							
EPLL	2.55	29.43	31.48	31.68	28.75	27.34	30.89
Proposed		32.07	35.17	33.88	33.62	33.92	35.49
EPLL	7.65	25.33	28.19	27.37	22.67	21.67	26.08
Proposed		28.11	32.03	29.51	29.20	29.07	31.63
Kernel 2 (17×17) [38]							
EPLL	2.55	29.67	32.26	31.00	27.53	26.75	30.44
Proposed		31.69	35.04	33.53	33.13	33.51	35.17
EPLL	7.65	24.85	28.08	27.03	21.60	21.09	25.77
Proposed		27.70	31.94	29.27	28.73	28.63	31.35



(a) Blurry and noisy image

(b) IDDBM3D (26.95dB)

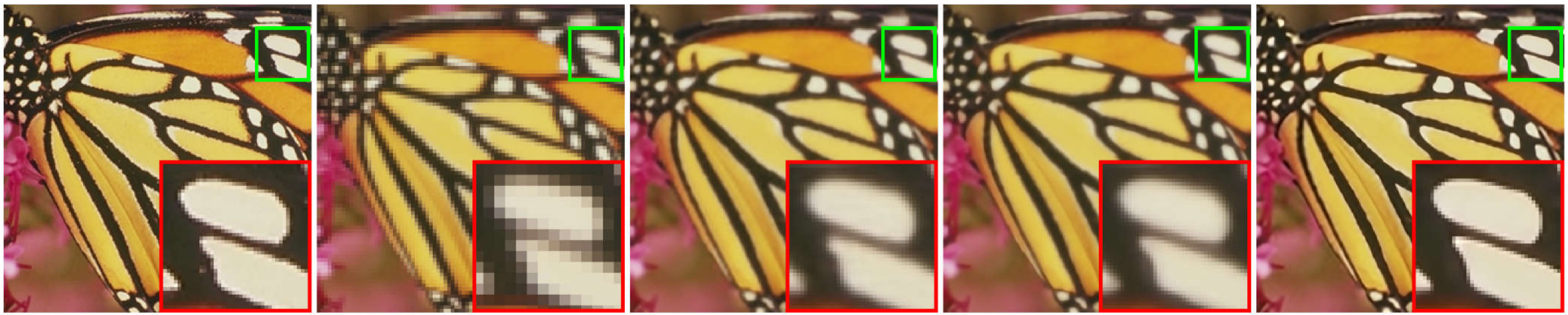
(c) NCSR (27.50dB)

(d) MLP (28.91dB)

(e) Proposed (29.78dB)

Super-resolution

Dataset	Scale	Kernel	Channel	SRCNN	VDSR	NCSR	SPMSR	SRBM3D	SRBM3D$_G$	SRBM3D$_C$	Proposed$_G$	Proposed$_C$
Set5	2	Bicubic	Y	36.65	37.56	-	36.11	37.10	36.34	36.25	37.43	37.22
			RGB	34.45	35.16	-	33.94	-	34.11	34.22	35.05	35.07
	3	Bicubic	Y	32.75	33.67	-	32.31	33.30	32.62	32.54	33.39	33.18
			RGB	30.72	31.50	-	30.32	-	30.57	30.69	31.26	31.25
	3	Gaussian	Y	30.42	30.54	33.02	32.27	-	32.66	32.59	33.38	33.17
			RGB	28.50	28.62	30.00	30.02	-	30.31	30.74	30.92	31.21
Set14	2	Bicubic	Y	32.43	33.02	-	31.96	32.80	32.09	32.25	32.88	32.79
			RGB	30.43	30.90	-	30.05	-	30.15	30.32	30.79	30.78
	3	Bicubic	Y	29.27	29.77	-	28.93	29.60	29.11	29.27	29.61	29.50
			RGB	27.44	27.85	-	27.17	-	27.32	27.47	27.72	27.67
	3	Gaussian	Y	27.71	27.80	29.26	28.89	-	29.18	29.39	29.63	29.55
			RGB	26.02	26.11	26.98	27.01	-	27.24	27.60	27.59	27.70



(a) Ground-truth (b) Zoomed LR image (c) SRCNN (24.46dB) (d) VDSR (24.73dB) (e) Proposed$_G$ (29.32dB)

Content

- Image Restoration
- Can we learn more from a set of degraded/clean image pairs?
- Can we directly learn a plain CNN for image restoration?
- Can we extend denoising CNN for general image restoration?
- Can we learn more for MAP-based restoration: efficiency and flexibility?

Limitation of Denoising CNNs for Restoration

- Multiple denoising CNNs
- Computational bottleneck
- Hard to end-to-end training

Using Gaussian Denoising as an Example

- Conventional CNN for restoration,
  - CNN aims to learn an explicit mapping for each setting on A and σ

$$\hat{\mathbf{x}} = F(\mathbf{y}; \mathbf{A}, \sigma^2)$$

- Now let's return to MAP model

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \frac{1}{2\sigma^2} \|\mathbf{A}\mathbf{x} - \mathbf{y}\|_2^2 + \Phi(\mathbf{x})$$

  - The solution actually defines an implicit function

$$\hat{\mathbf{x}} = F(\mathbf{y}, \mathbf{A}, \sigma^2)$$

- As for Gaussian denoising

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \frac{1}{2\sigma^2} \|\mathbf{y} - \mathbf{x}\|^2 + \lambda\Phi(\mathbf{x}) \Rightarrow \hat{\mathbf{x}} = f(\mathbf{y}, \sigma)$$

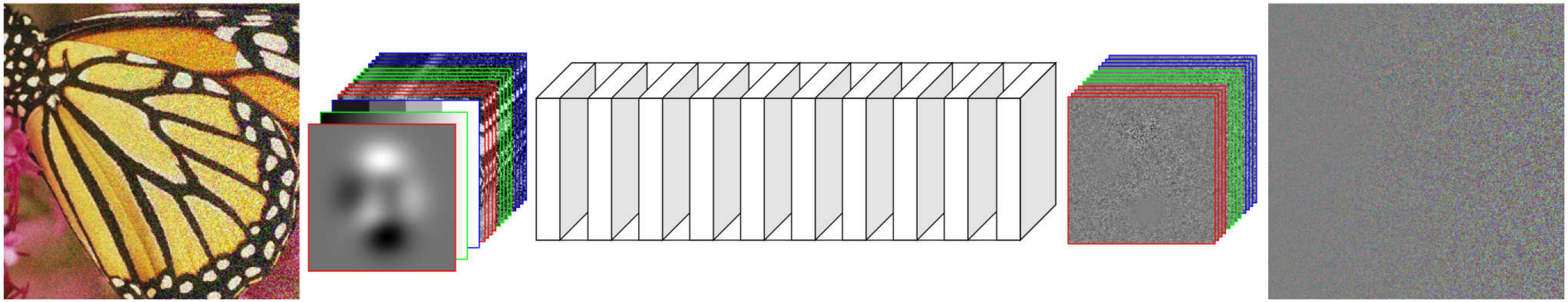
FFDNet (Zhang et al., Arxiv 2017)

- Flexibility: a single model to handle noisy image with different noise levels or even spatially variant noise.

- Fast speed: highly efficient without sacrificing denoising performance.

- Robustness: robust to the estimation error of noise levels.

FFDNet (Zhang et al., Arxiv 2017)



- Taking noise map as input
- Denoising on sub-images
- Orthogonal Regularization on Convolution Filters

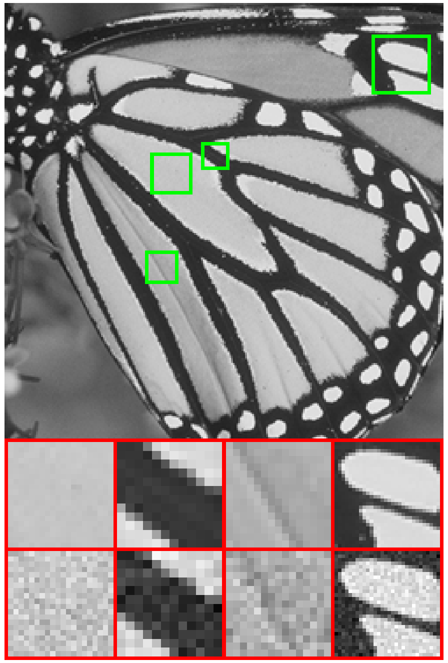
- Test code: https://github.com/cszn/FFDNet

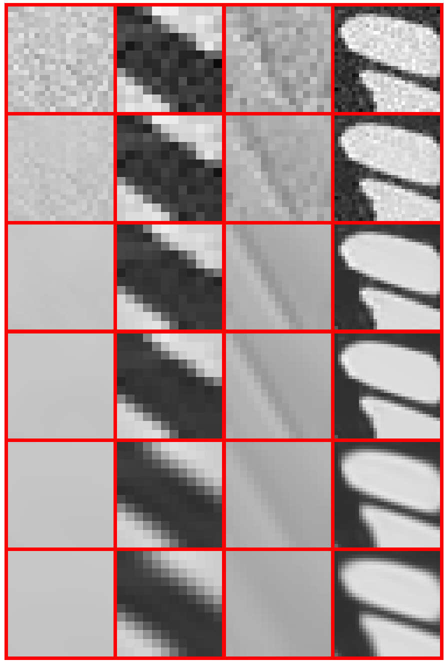
Denoising performance and run time

Methods	BM3D	WNNM	MLP	TNRD	DnCNN	FFDNet
$\sigma = 15$	31.07	31.37	–	31.42	31.72	31.62
$\sigma = 25$	28.57	28.83	28.96	28.92	29.23	29.19
$\sigma = 35$	27.08	27.30	27.50	–	27.69	27.73
$\sigma = 50$	25.62	25.87	26.03	25.97	26.23	26.30
$\sigma = 75$	24.21	24.40	24.59	–	24.64	24.78

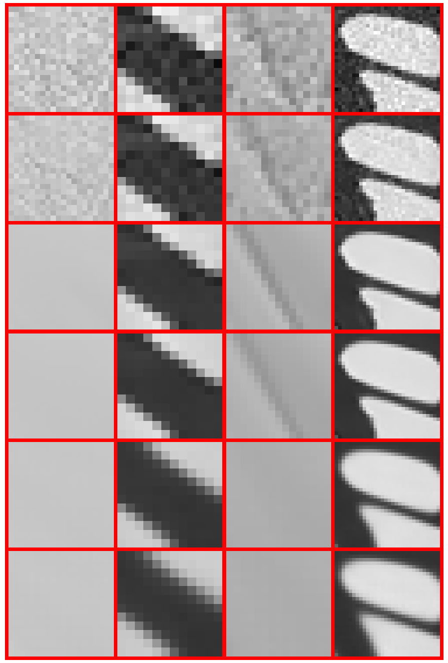
Methods	Device	256×256		512×512		1024×1024	
		Gray	Color	Gray	Color	Gray	Color
BM3D	CPU(ST)	0.59	0.98	2.52	3.57	10.77	20.15
DnCNN	CPU(ST)	2.14	2.44	8.63	9.85	32.82	38.11
	CPU(MT)	0.74	0.98	3.41	4.10	12.10	15.48
	GPU	0.011	0.014	0.033	0.040	0.124	0.167
FFDNet	CPU(ST)	0.44	0.52	1.81	2.14	7.24	8.51
	CPU(MT)	0.18	0.19	0.73	0.79	2.96	3.15
	GPU	0.006	0.007	0.012	0.016	0.038	0.054

Robustness to Noise Level Mismatching





BM3D



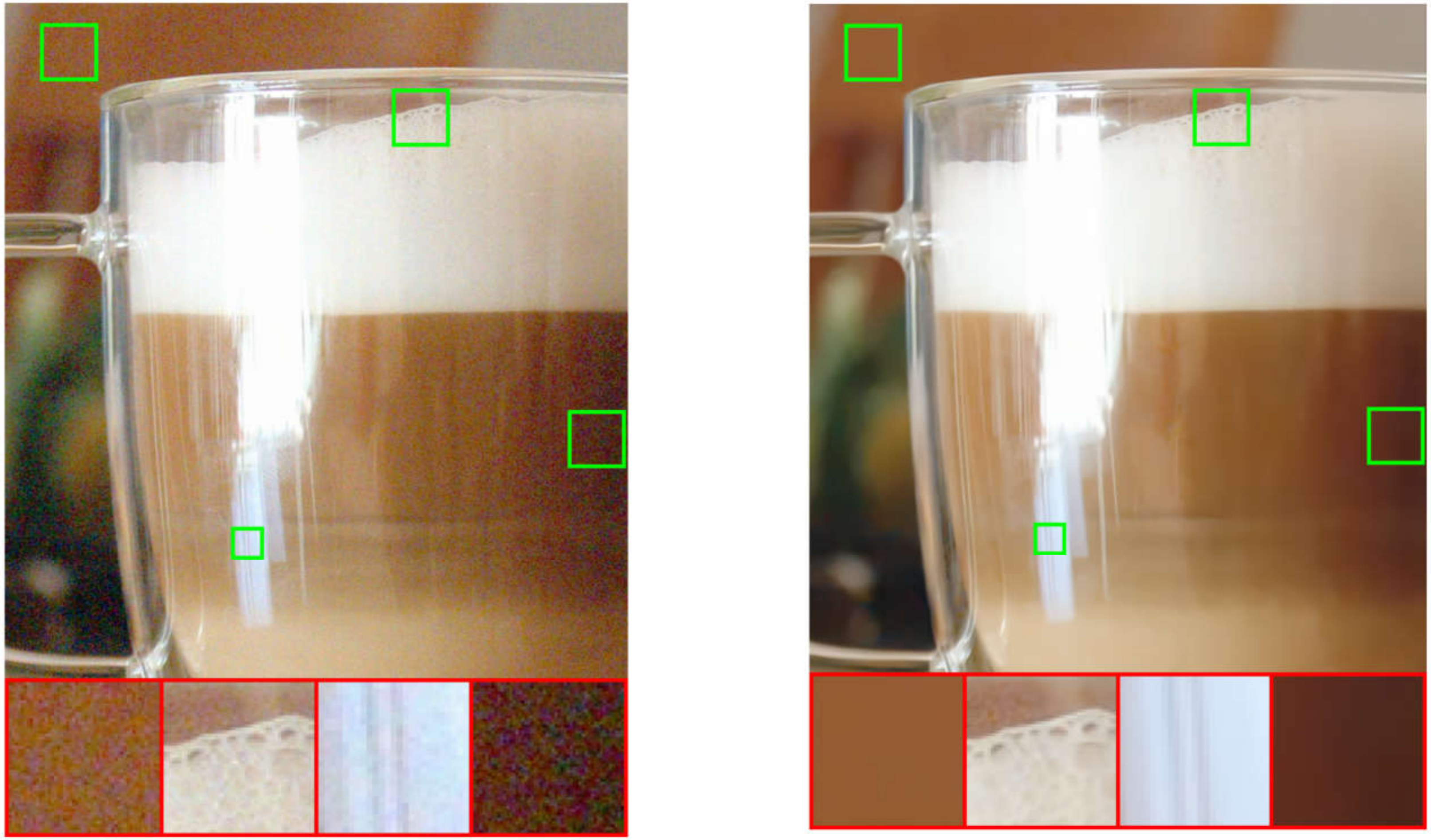
Proposed

From top to bottom: denoising results with input noise levels 5, 10, 15, 20, 50, and 75, respectively.

Denoising on real images

Denoising on real images

Denoising on real images

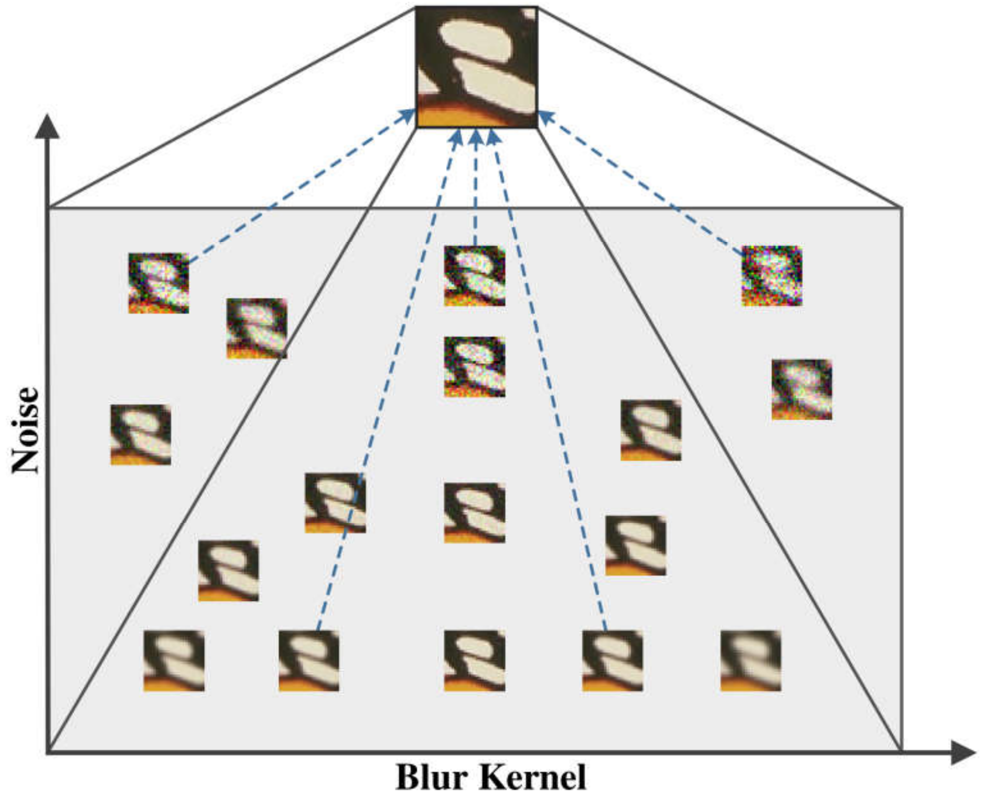


SRMD: From Denoising to Super-Resolution

- Multiple degradations
- Image downsampling

$$\boldsymbol{y} = (\boldsymbol{x} \otimes \boldsymbol{k}) \downarrow_s + \boldsymbol{n}$$

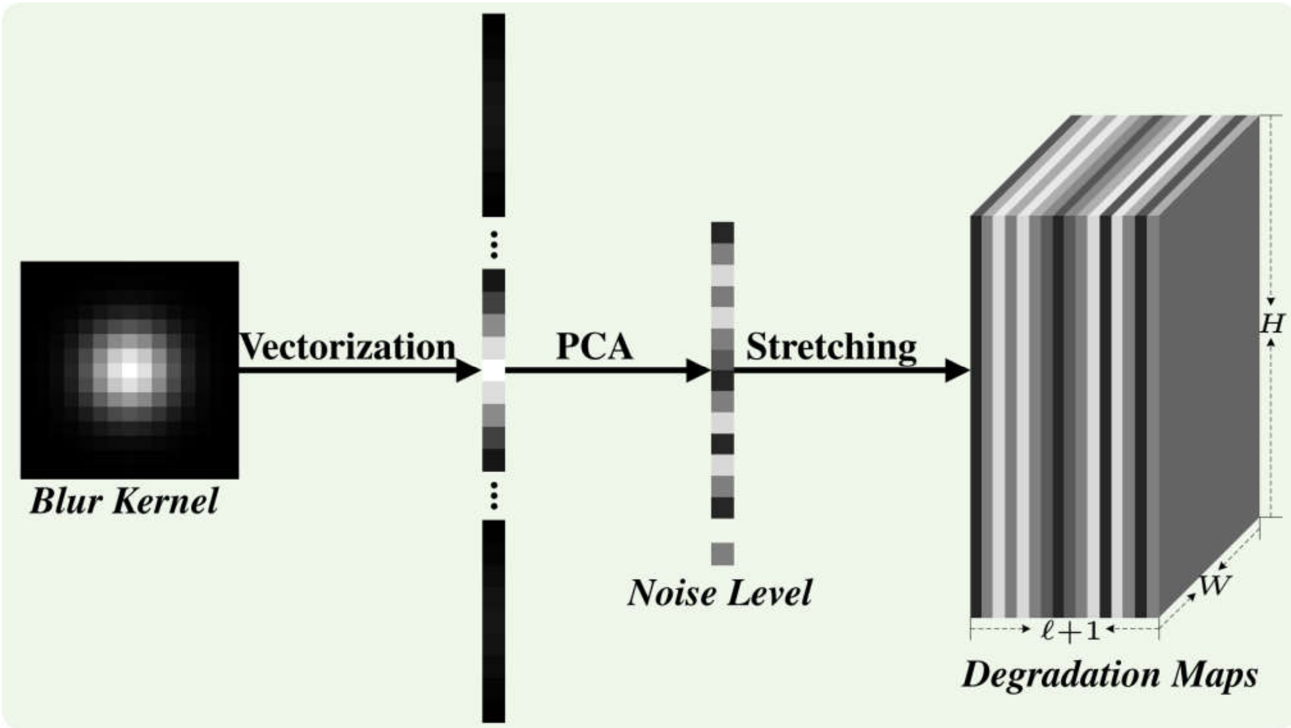
  - Blur kernel
  - Noise level
  - Downsampler



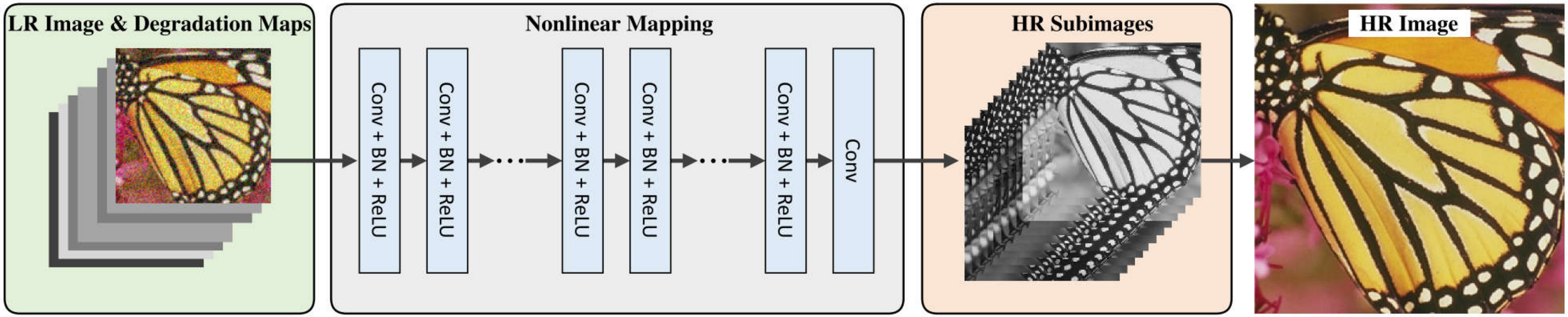
SRMD

$$\hat{\mathbf{x}} = F(\mathbf{y}, \mathbf{k}, \sigma^2)$$

- nonuniform noise
- non-uniform blur kernel



Network architecture

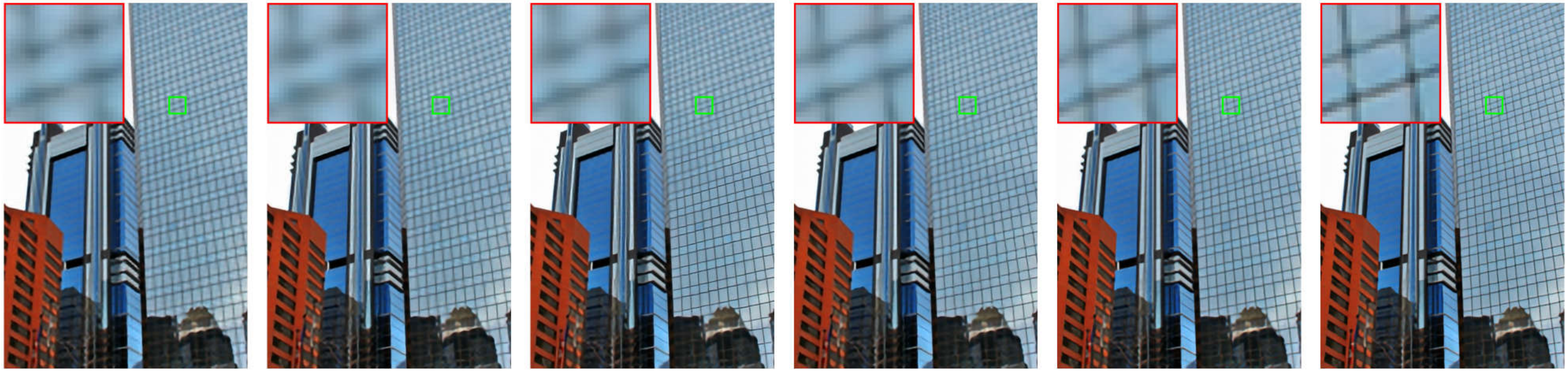


- Testing code: https://github.com/cszn/SRMD

Result

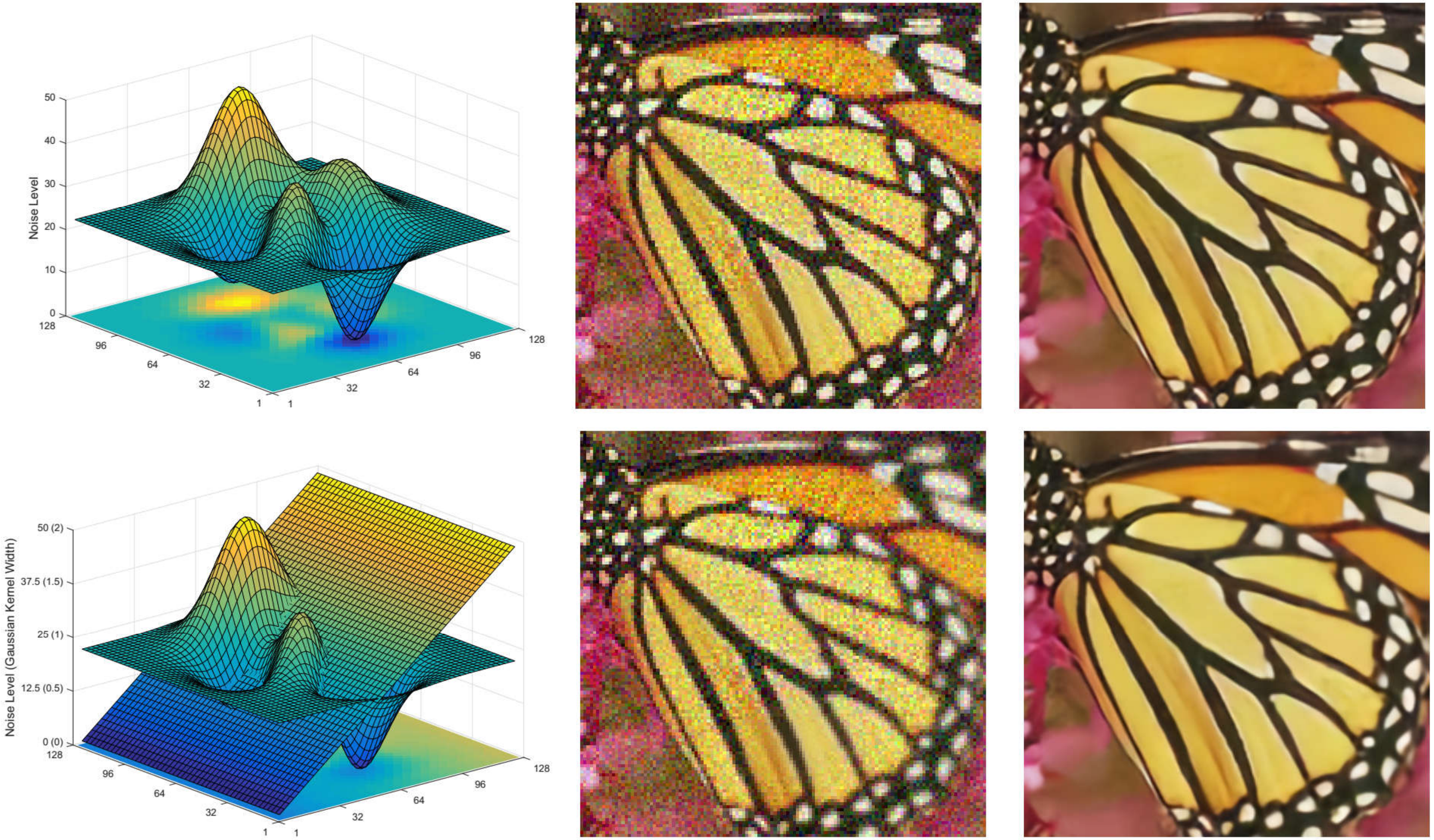
Degradation Settings			VDSR [22]	NCSR [10]	IRCNN [54]	DnCNN [53]+SRMDNF	SRMD	SRMDNF
Kernel Width	Down-sampler	Noise Level	PSNR (×2/×3/×4)					
0.2	Bicubic	0	37.56/33.67/31.35	– /23.82/–	37.43/33.39/31.02	–	37.53/33.76/31.59	37.75/34.09/31.96
0.2	Bicubic	15	26.02/25.40/24.70	–	32.60/30.08/28.35	32.47/30.07/28.31	32.72/30.38/28.76	–
0.2	Bicubic	50	16.02/15.72/15.46	–	28.20/26.25/24.95	28.20/26.27/24.93	28.48/26.45/25.18	–
1.3	Bicubic	0	30.57/30.24/29.72	– /21.81/–	36.01/33.33/31.01	–	36.68/33.64/31.48	37.40/34.10/31.99
1.3	Bicubic	15	24.82/24.70/24.30	–	29.96/28.68/27.71	27.68/28.78/27.71	30.95/29.39/28.18	–
1.3	Bicubic	50	15.89/15.68/15.43	–	26.69/25.20/24.42	24.35/25.19/24.39	27.41/25.79/24.78	–
2.6	Bicubic	0	26.37/26.31/26.28	– /21.46/–	32.07/31.09/30.06	–	32.81/32.23/31.01	34.10/32.88/31.77
2.6	Bicubic	15	23.09/23.07/22.98	–	26.44/25.67/24.36	– /21.33/23.85	28.44/27.53/26.80	–
2.6	Bicubic	50	15.58/15.43/15.23	–	22.98/22.16/21.43	– /19.03/21.15	25.80/24.72/24.01	–
1.6	Direct	0	– /30.54/ –	– /33.02/ –	– /33.38/ –	–	– /33.74/ –	– /34.01/ –

Result: Urban100

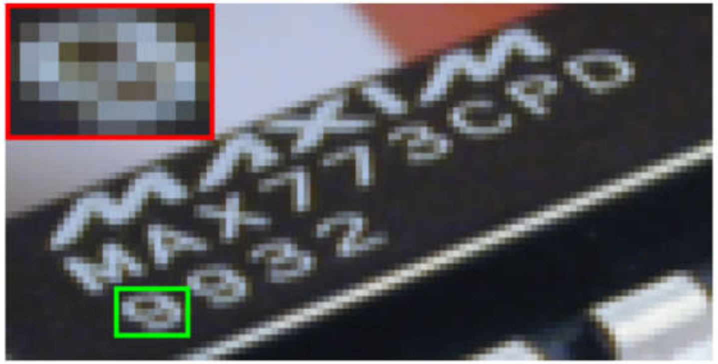


(a) SRCNN (23.78dB) (b) VDSR (24.20dB) (c) DRRN (25.11dB) (d) LapSR (24.47dB) (e) SRMD (24.74dB) (f)SRMDNF (25.38dB)

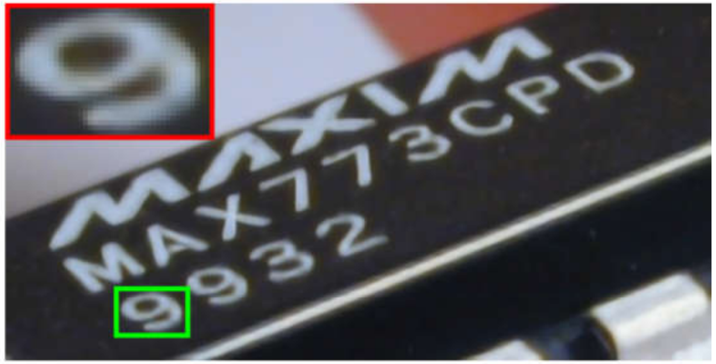
Result: nonuniform noise and blur



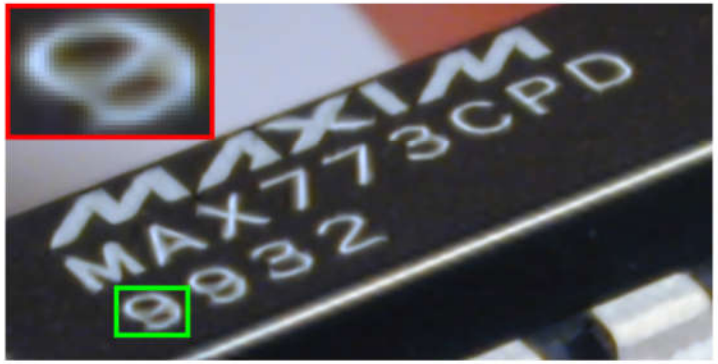
Results



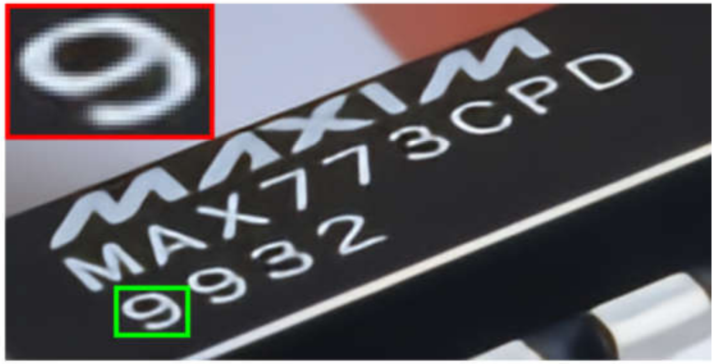
(a) LR image



(b) VDSR [22]

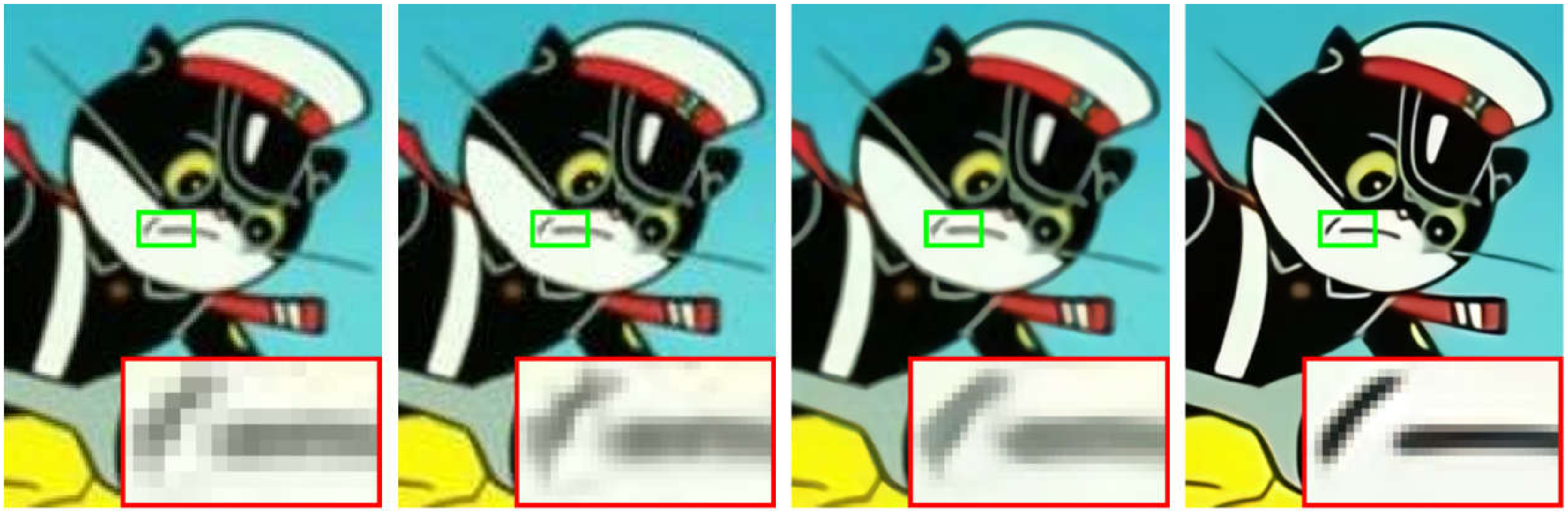


(c) SelfEx [18]



(d) SRMD

Results



(a) LR image (b) VDSR [22] (c) Waifu2x [47] (d) SRMD

Summary

- Model-guided Network design and learning: Discriminative learning of stage-wise parameters for blind deconvolution
- Task-specific CNN: Development of CNN-based models for image denoising
- Incorporation of traditional model and CNN for general image restoration
- Taking degradation model parameters as input to CNN
- Future work: More practical image restoration or generation

Related Publications

- W. Zuo, D. Ren, D. Zhang, S. Gu, L. Lin, L. Zhang, Discriminative Learning of Iteration-wise Priors for Blind Deconvolution, CVPR 2015.
- W. Zuo, D. Ren, D. Zhang, S. Gu, L. Zhang. Learning Iteration-wise Generalized Shrinkage-Thresholding Operators for Blind Deconvolution, IEEE Trans. Image Processing, 25(4): 1751 - 1764, 2016.
- S. Gu, W. Zuo, S. Guo, Y. Chen, C. Chen, L. Zhang. Learning Dynamic Guidance for Depth Image Enhancement, CVPR 2017.
- K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising, IEEE Trans. Image Processing, 2017
- K. Zhang, W. Zuo, S. Gu, and L. Zhang. Learning deep CNN denoiser prior for image restoration. In CVPR 2017.
- L. Zhang, W. Zuo, Image Restoration: From Sparse and Low-rank Priors to Deep Priors, IEEE Signal Processing Magazine, Sept. 2017.
- K. Zhang, W. Zuo, L. Zhang. FFDNet: Toward a Fast and Flexible Solution for CNN based Image Denoising, https://arxiv.org/abs/1710.04026
- K. Zhang, W. Zuo, L. Zhang. Learning a Single Convolutional Super-Resolution Network for Multiple Degradations, https://arxiv.org/abs/1712.06116