



# Computer Vision; Image Transformation; Super-Resolution, Denoising, and Colorization



[YouTube Playlist](#)

**Maziar Raissi**

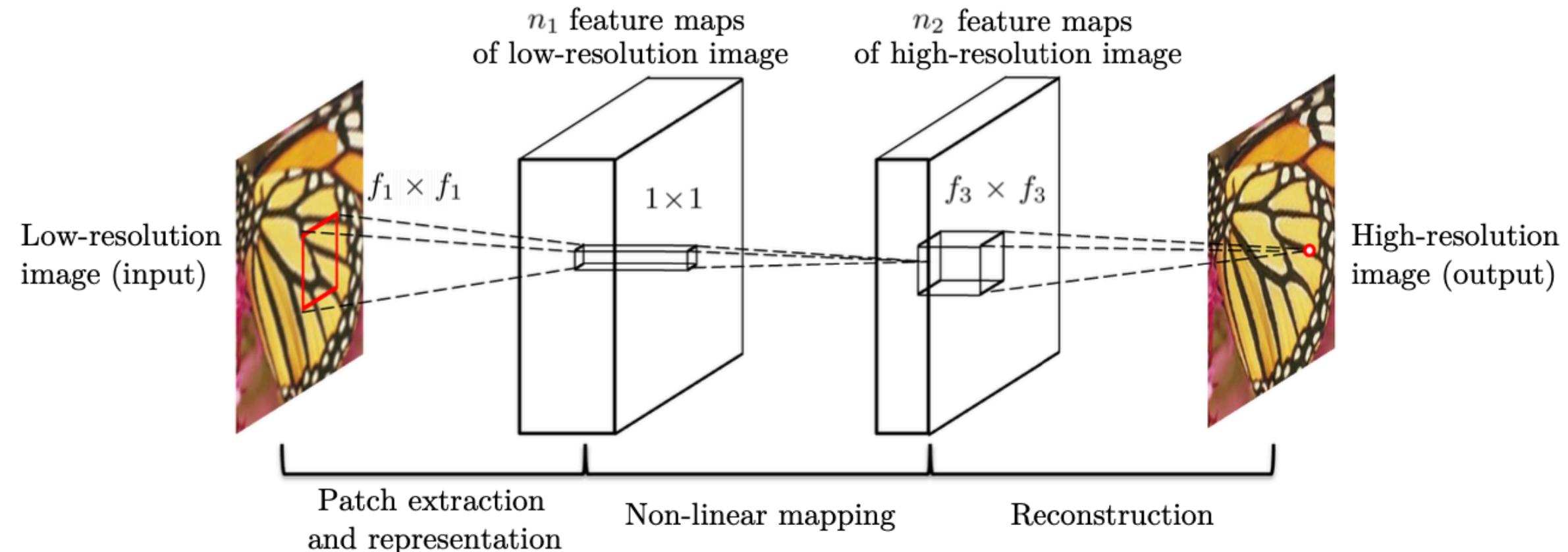
**Assistant Professor**

Department of Applied Mathematics

University of Colorado Boulder

[maziar.raissi@colorado.edu](mailto:maziar.raissi@colorado.edu)

# Learning a Deep Convolutional Network for Image Super-Resolution


[YouTube Video](#)


low resolution image  $\triangleright$  bicubic interpolation  
 $Y \rightarrow$  interpolated image  
 $F(Y) \rightarrow$  an image as similar as possible to  $X$   
 $X \rightarrow$  high resolution image

$$F_1(Y) = \max(0, W_1 * Y + B_1)$$

$$W_1 \in \mathbb{R}^{c \times f_1 \times f_1 \times n_1} \quad B_1 \in \mathbb{R}^{n_1}$$

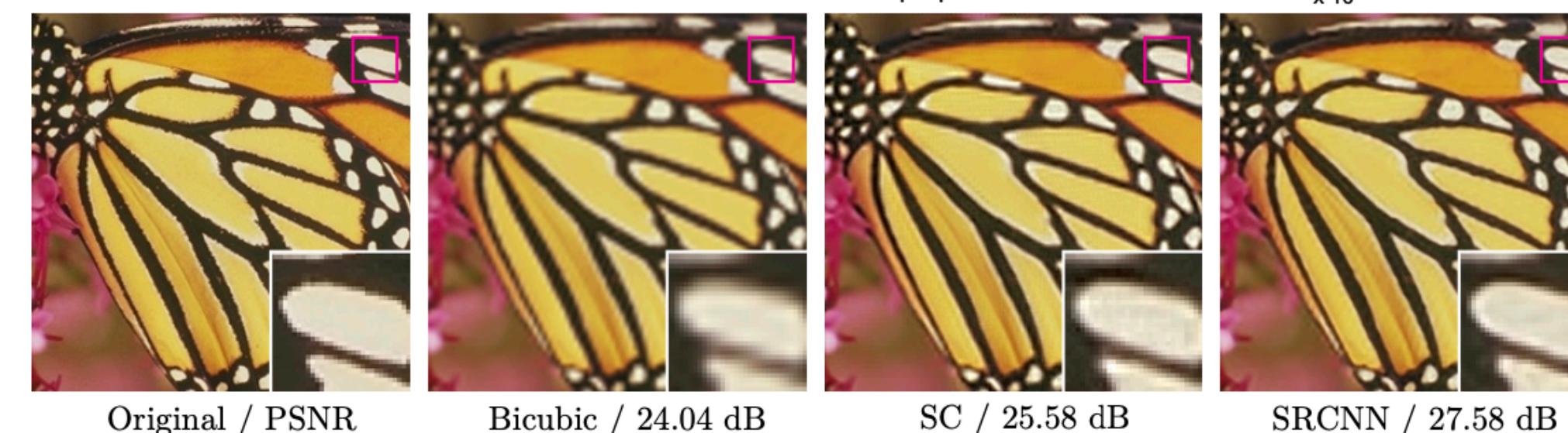
$$F_2(Y) = \max(0, W_2 * F_1(Y) + B_2)$$

$$W_2 \in \mathbb{R}^{n_1 \times 1 \times 1 \times n_2} \quad B_2 \in \mathbb{R}^{n_2}$$

$$F_3(Y) = W_3 * F_2(Y) + B_3$$

$$W_3 \in \mathbb{R}^{n_2 \times f_3 \times f_3 \times c} \quad B_3 \in \mathbb{R}^c$$

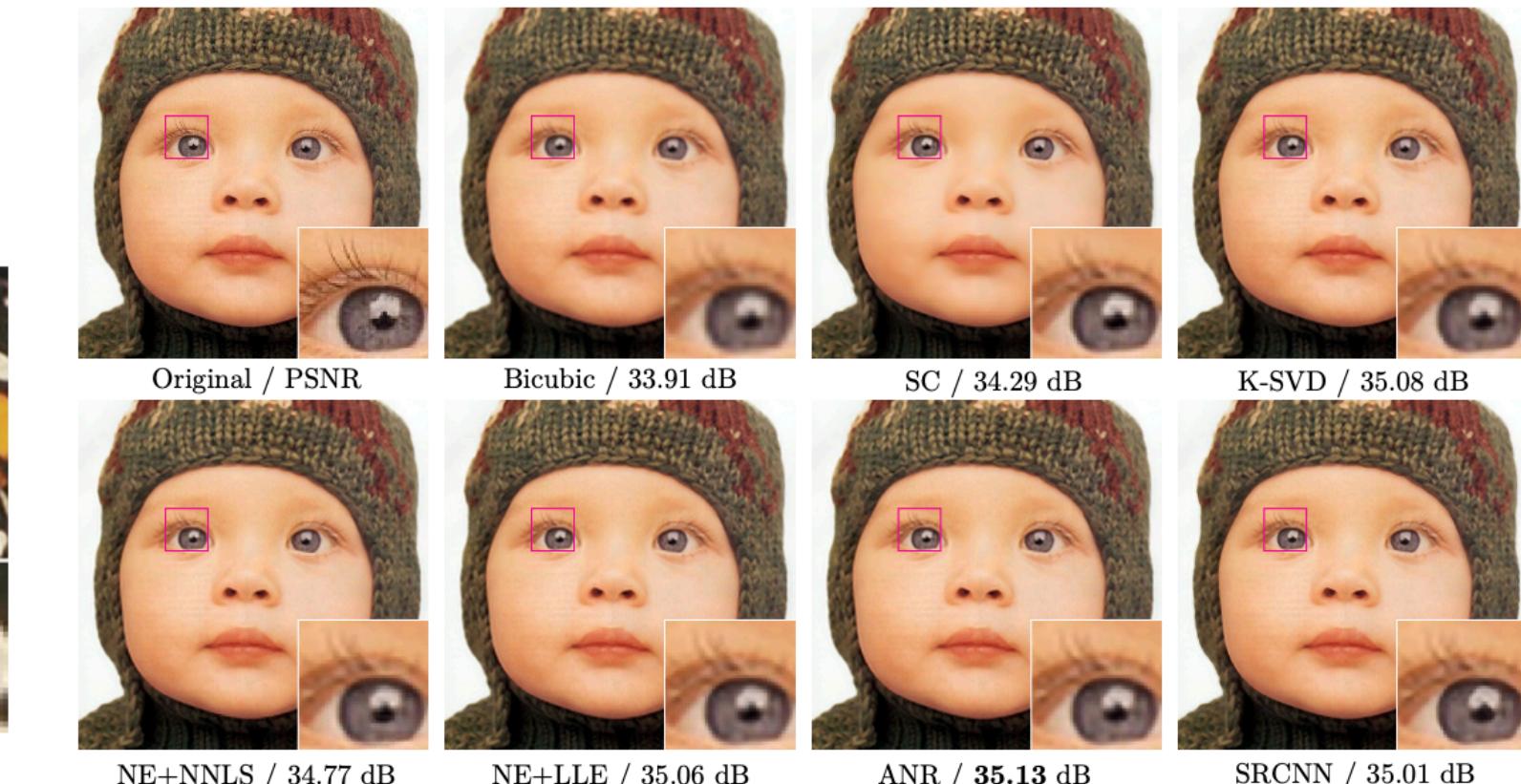
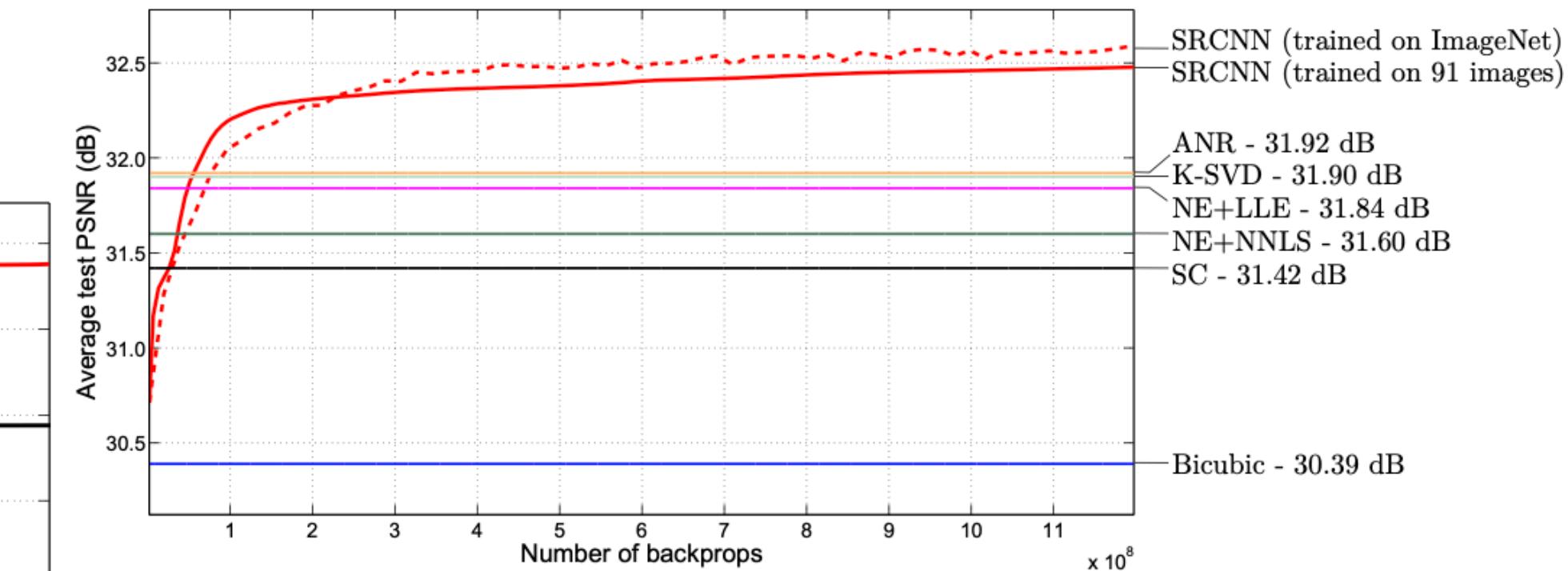
$$L(\Theta) = \frac{1}{n} \sum_{i=1}^n \|F(Y_i; \Theta) - X_i\|^2$$



Super-Resolution Convolutional Neural Network (SRCNN)  
Sparse Coding (SC) based method  
Peak Signal-to-Noise Ratio (PSNR)

Set5 [2] images	$n_1 = 128, n_2 = 64$		$n_1 = 64, n_2 = 32$		$n_1 = 32, n_2 = 16$	
	PSNR	Time	PSNR	Time	PSNR	Time
	32.60	0.60	32.52	0.18	32.26	0.05

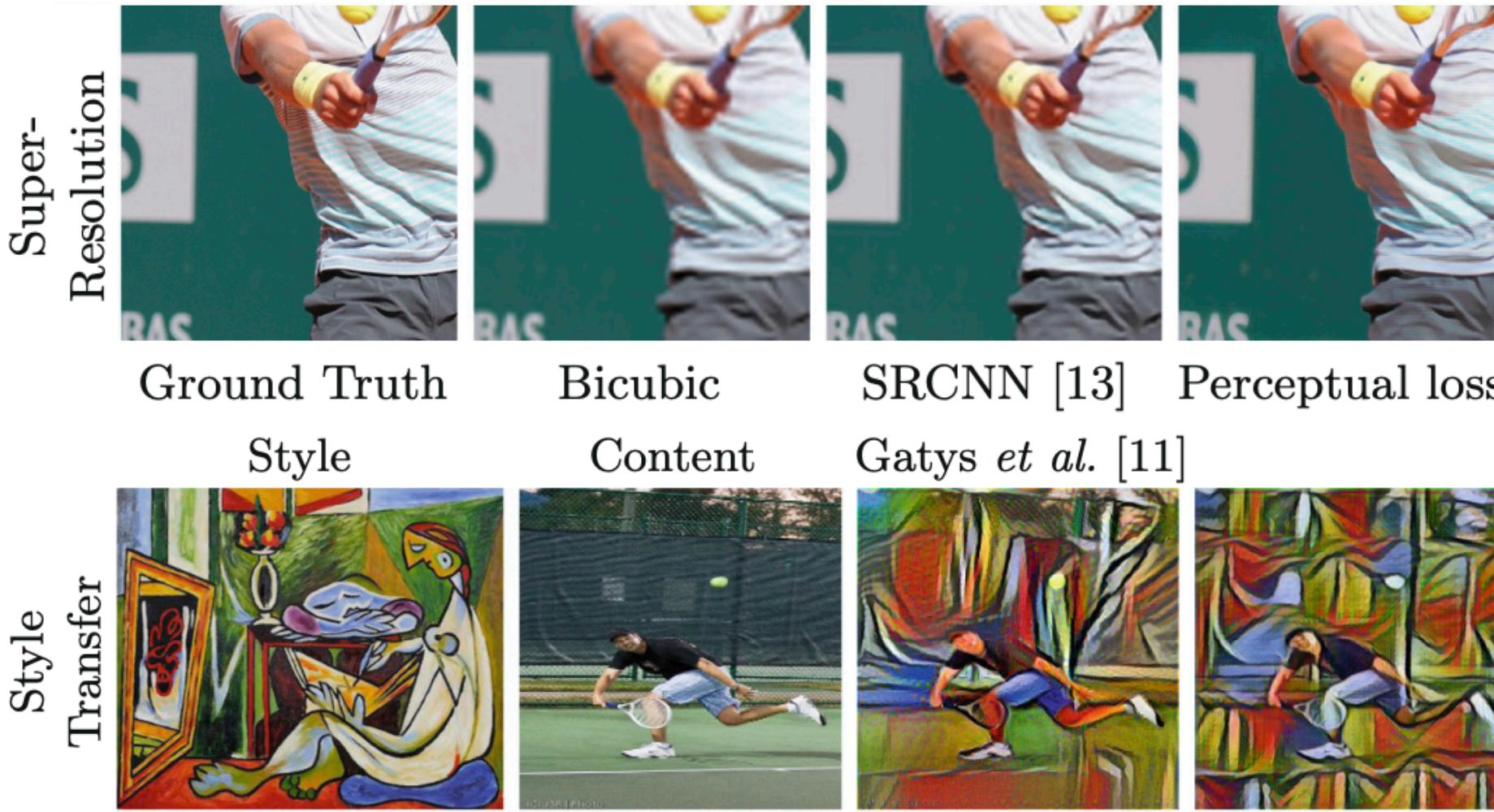
$$\begin{aligned} PSNR &= 10 \cdot \log_{10} \left( \frac{MAX_I^2}{MSE} \right) \\ &= 20 \cdot \log_{10} \left( \frac{MAX_I}{\sqrt{MSE}} \right) \\ &= 20 \cdot \log_{10}(MAX_I) - 10 \cdot \log_{10}(MSE) \\ MAX_I &= 255 \end{aligned}$$



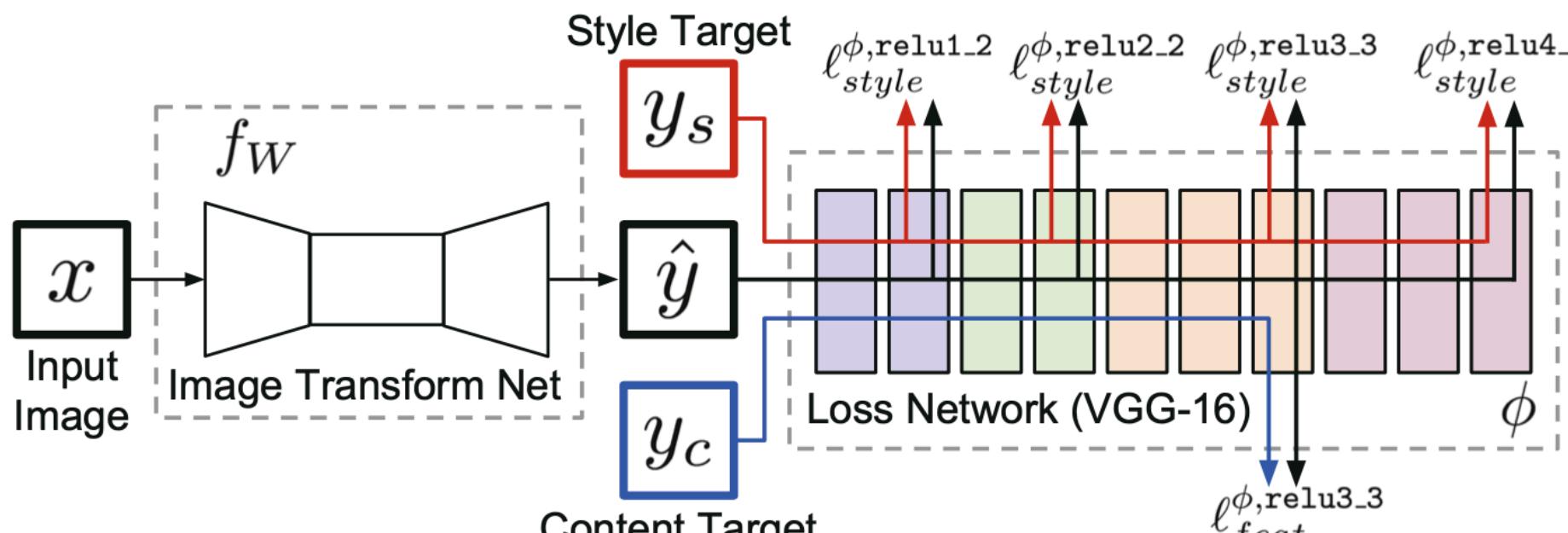
Dong, Chao, et al. "Learning a deep convolutional network for image super-resolution." *European conference on computer vision*. Springer, Cham, 2014.

Dong, Chao, et al. "Image super-resolution using deep convolutional networks." *IEEE transactions on pattern analysis and machine intelligence* 38.2 (2015): 295-307.

# Perceptual Losses for Real-Time Style Transfer and Super-Resolution


[YouTube Video](#)


"For example, consider two identical images offset from each other by one pixel; despite their perceptual similarity they would be very different as measured by per-pixel losses."



$$W^* = \arg \min_W \mathbb{E}_{x,y} \left[ \sum_i \lambda_i \ell_i(\underbrace{f_W(x)}_{\hat{y}}, y) \right]$$

**Style transfer:**  $y_c = x$  & train one network per style target  $y_s$

**Super-resolution:**  $y_c = \text{high-resolution image}$  & style reconstruction loss is not used

## Feature Reconstruction Loss

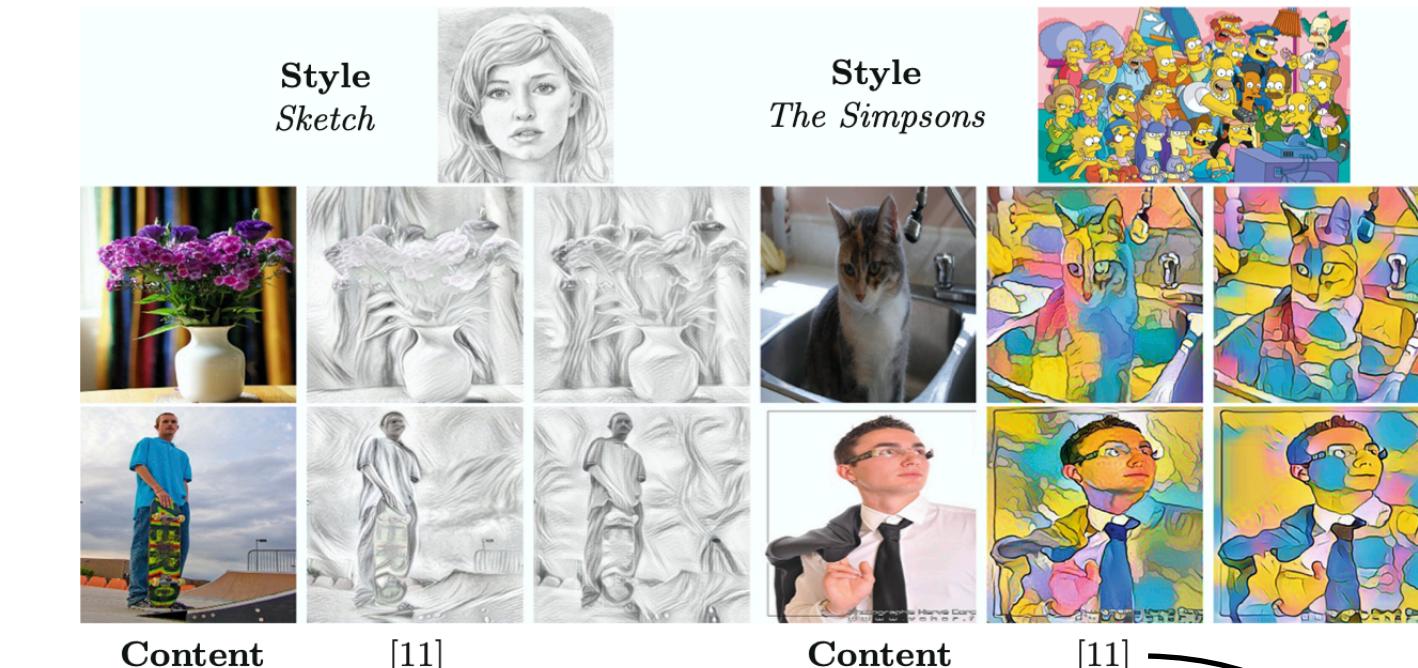
$\phi_j(x) \in \mathbb{R}^{C_j \times H_j \times W_j} \rightarrow \text{activations of the } j\text{-th layer of network } \phi$

$$\ell_{feat}^{\phi,j}(\hat{y}, y) = \frac{1}{C_j H_j W_j} \|\phi_j(\hat{y}) - \phi_j(y)\|_2^2$$

## Style Reconstruction Loss

$$G_j^\phi(x)_{c,c'} = \frac{1}{C_j H_j W_j} \sum_{h=1}^{H_j} \sum_{w=1}^{W_j} \phi_j(x)_{h,w,c} \phi_j(x)_{h,w,c'}$$

$G_j^\phi(x) \in \mathbb{R}^{C_j \times C_j} \rightarrow \text{Gram Matrix}$



Gram matrix captures information about which features tend to activate together.

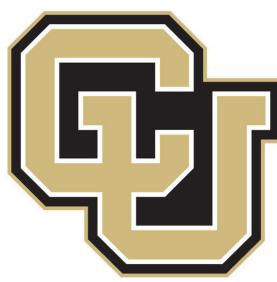
$$\ell_{style}^{\phi,j}(\hat{y}, y) = \|G_j^\phi(\hat{y}) - G_j^\phi(y)\|_F^2$$

$$\hat{y} = \arg \min_y \lambda_c \ell_{feat}^{\phi,j}(y, y_c) + \lambda_s \ell_{style}^{\phi,j}(y, y_s) + \lambda_{TV} \ell_{TV}(y)$$



Speedup over [11] at 100, 300, 500 optimization iterations

Image Size	Speedup		
	100	300	500
256 × 256	212x	636x	<b>1060x</b>
512 × 512	205x	615x	<b>1026x</b>
1024 × 1024	208x	625x	<b>1042x</b>



Boulder

# Image Style Transfer Using Convolutional Neural Networks

## Content Representation

$p \rightarrow$  photograph

$x \rightarrow$  generated image (randomly initialized)

$P^l \rightarrow$  feature representation of  $p$  at layer  $l$

$F^l \rightarrow$  feature representation of  $x$  at layer  $l$

$$\mathcal{L}_{\text{content}}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} (F_{ij}^l - P_{ij}^l)^2$$

$F_{i,j}^l \rightarrow$  activation of  $i$ -th filter at position  $j$  in layer  $l$

$$F^l \in \mathbb{R}^{N_l \times M_l}$$

$N_l \rightarrow$  number of feature maps

$M_l \rightarrow$  height times width of the feature map

## Style Representation

$$G^l \in \mathbb{R}^{N_l \times N_l} \rightarrow \text{Gram matrix}$$

$$G_{ij}^l = \sum_k F_{ik}^l F_{jk}^l \rightarrow \text{inner product between the vectorized feature maps } i \text{ and } j \text{ in layer } l$$

captures texture but not global arrangement

$a \rightarrow$  artwork image     $A^l \rightarrow$  style representation of  $a$

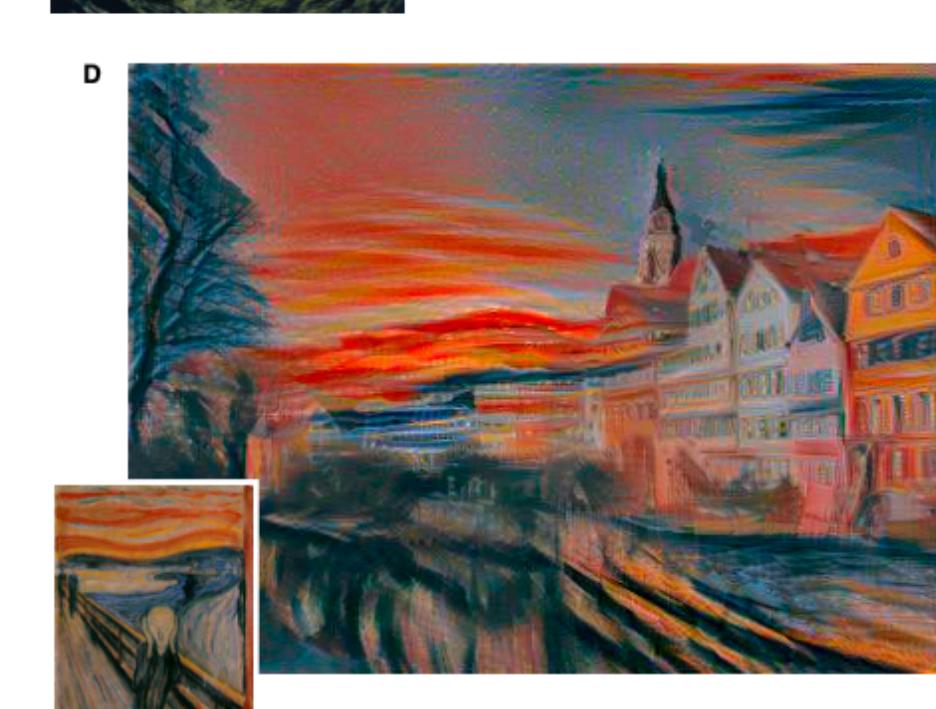
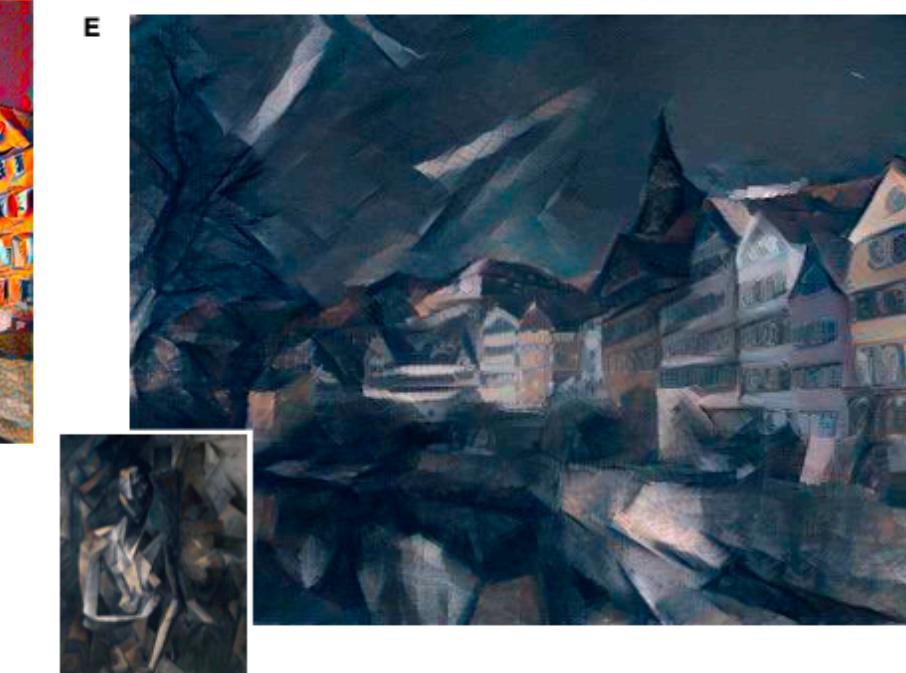
$G^l \rightarrow$  style representation of  $x$

$$\mathcal{L}_{\text{style}}(\vec{a}, \vec{x}) = \sum_{l=0}^L w_l E_l \quad E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G_{ij}^l - A_{ij}^l)^2 \quad \boxed{\text{The ratio } \alpha/\beta}$$

## Style Transfer

$$\mathcal{L}_{\text{total}}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathcal{L}_{\text{content}}(\vec{p}, \vec{x}) + \beta \mathcal{L}_{\text{style}}(\vec{a}, \vec{x})$$

$$\frac{\partial \mathcal{L}_{\text{total}}}{\partial \vec{x}}$$



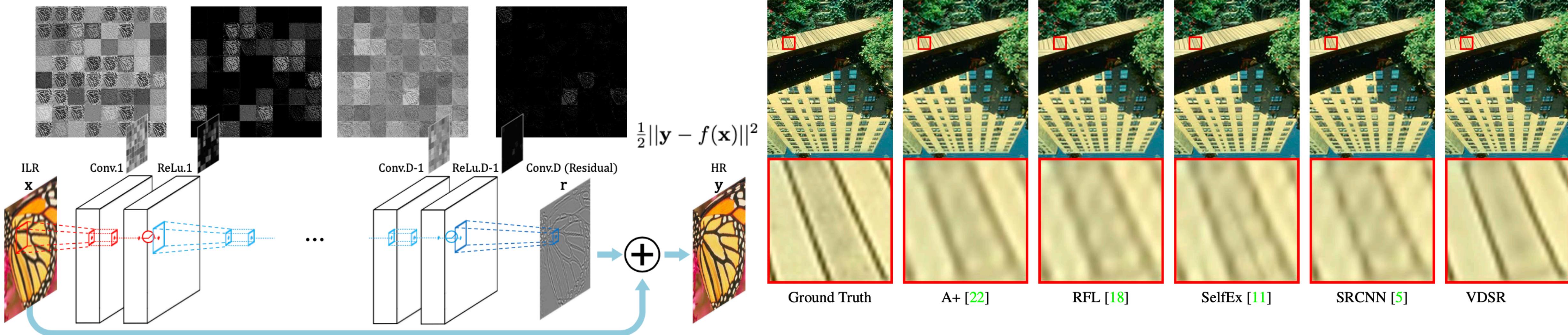
Gatys, Leon A., Alexander S. Ecker, and Matthias Bethge. "Image style transfer using convolutional neural networks." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

Gatys, Leon A., Alexander S. Ecker, and Matthias Bethge. "A neural algorithm of artistic style." *arXiv preprint arXiv:1508.06576* (2015).

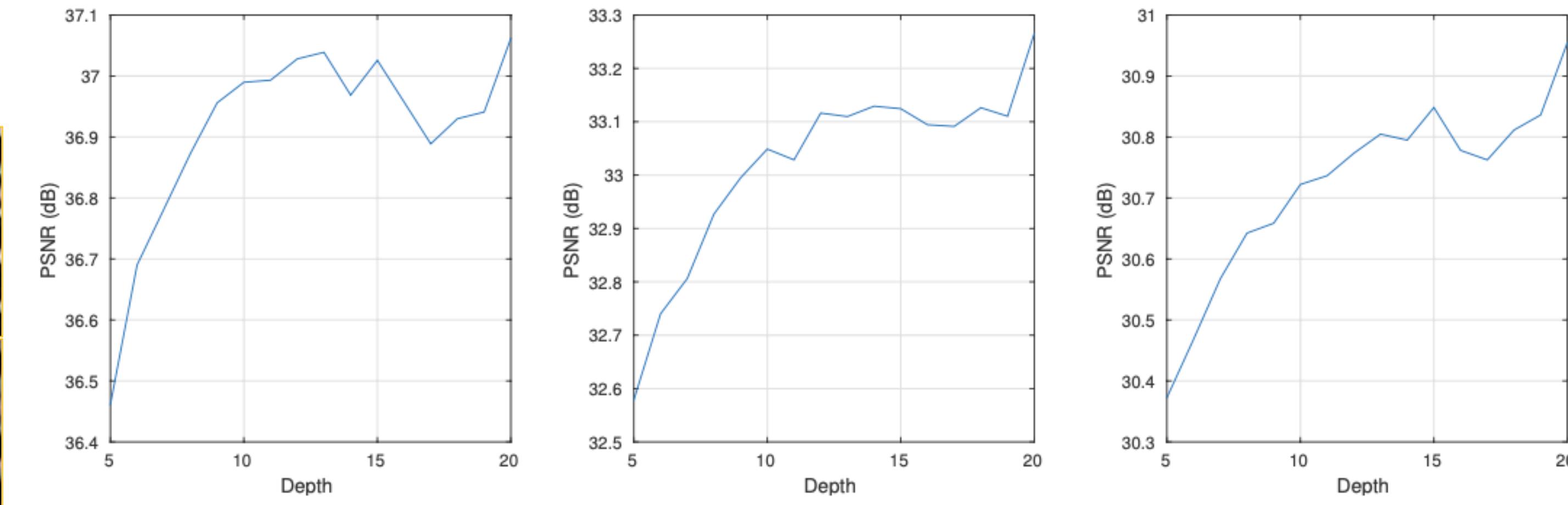
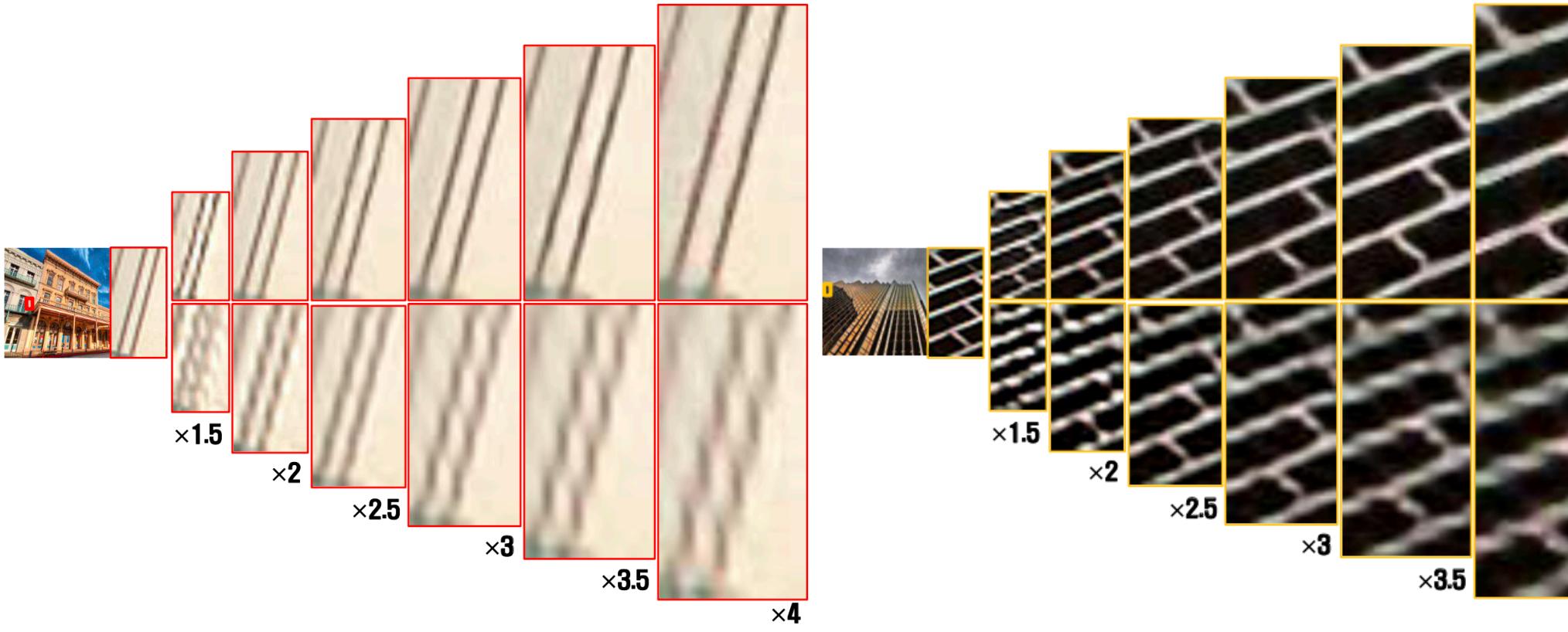
# Accurate Image Super-Resolution Using Very Deep Convolutional Networks


[YouTube Video](#)

## Single Image Super-Resolution (SISR)



- Residual-Learning
- High Learning Rates
- Adjustable Gradient Clipping
- Multi-scale





# Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network

$I^{\text{HR}}$  → high-resolution image

$I^{\text{LR}}$  → low-resolution image

$I^{\text{LR}} \leftarrow I^{\text{HR}}$   
    ▷ Gaussian filter  
    ▷ Downsampling

$I^{\text{SR}}$  → super-resolved image

$r$  → upscaling ratio

$I^{\text{LR}} \in \mathbb{R}^{H \times W \times C}$

$I^{\text{HR}} \in \mathbb{R}^{rH \times rW \times C}$

avoid upscaling  $I^{\text{LR}}$

(i.e., bicubic interpolation)

before feeding into the network

$$f^1(I^{\text{LR}}; W_1, b_1) = \tanh(W_1 * I^{\text{LR}} + b_1)$$

$$f^l(I^{\text{LR}}; W_{1:l}, b_{1:l}) = \tanh(W_l * f^{l-1}(I^{\text{LR}}) + b_l), l = 2, \dots, L-1$$

$$W_l \in \mathbb{R}^{n_{l-1} \times n_l \times k_l \times k_l}, b_l \in \mathbb{R}^{n_l}$$

$n_l$  → number of features at layer  $l$

$n_0 = C, k_l$  → filter size at layer  $l$

## Efficient sub-pixel convolution layer

$$\mathbf{I}^{\text{SR}} = f^L(\mathbf{I}^{\text{LR}}) = \mathcal{PS}(W_L * f^{L-1}(\mathbf{I}^{\text{LR}}) + b_L)$$

$\mathcal{PS}$  → periodic shuffling operator

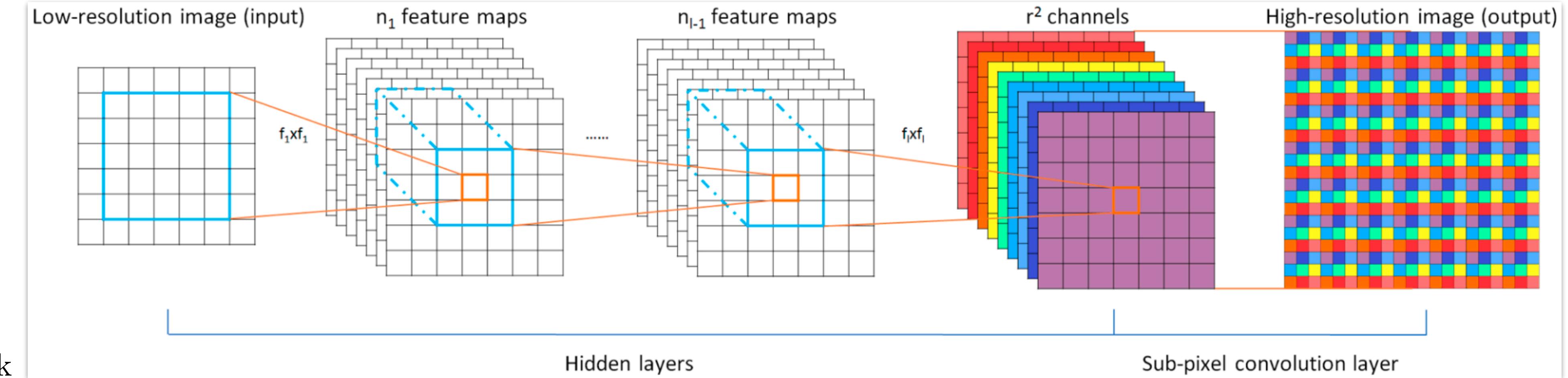
$$H \times W \times C \cdot r^2 \mapsto rH \times rW \times C$$

$$\mathcal{PS}(T)_{x,y,c} = T_{\lfloor x/r \rfloor, \lfloor y/r \rfloor, c \cdot r \cdot \text{mod}(y,r) + c \cdot \text{mod}(x,r)}$$

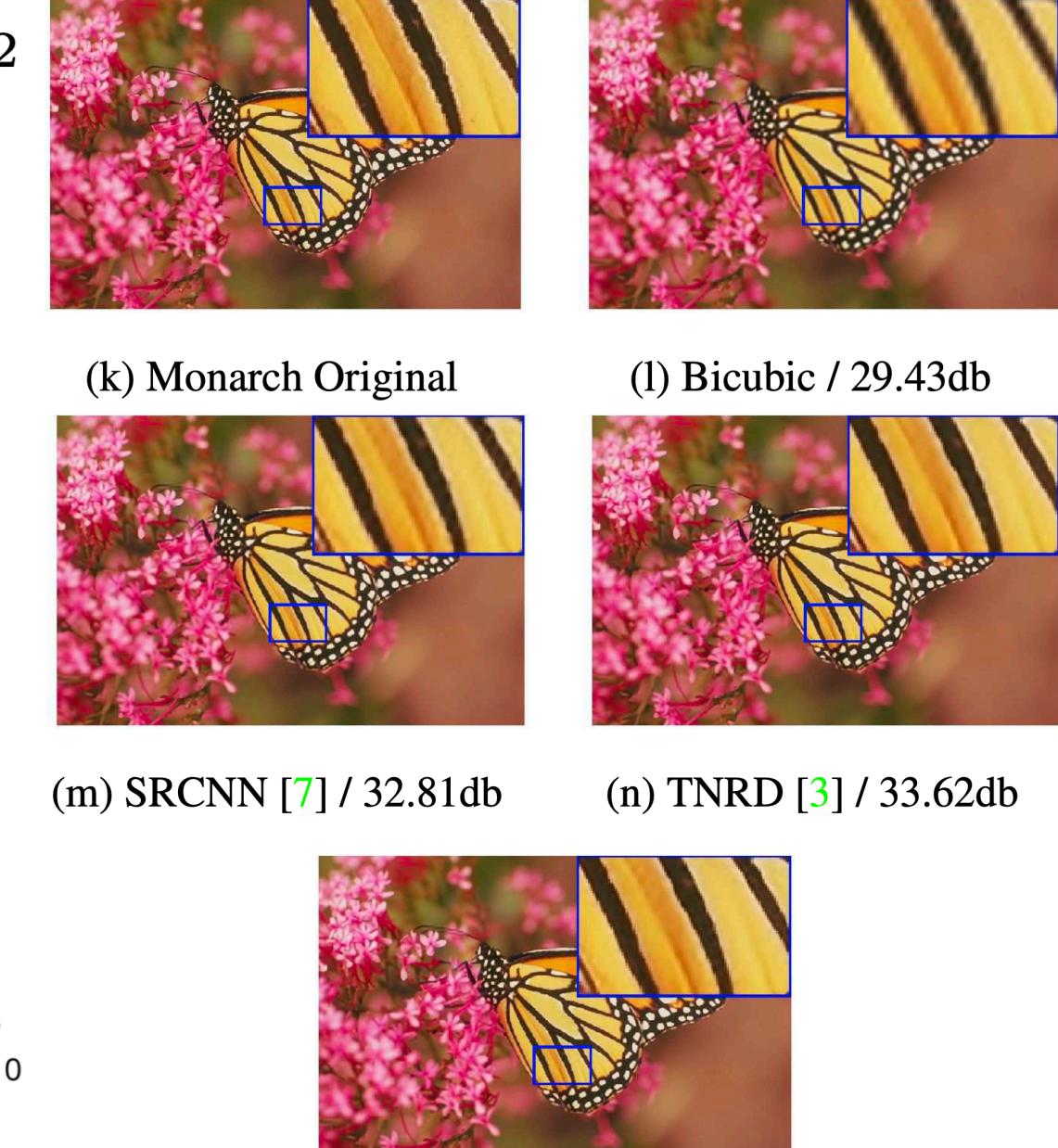
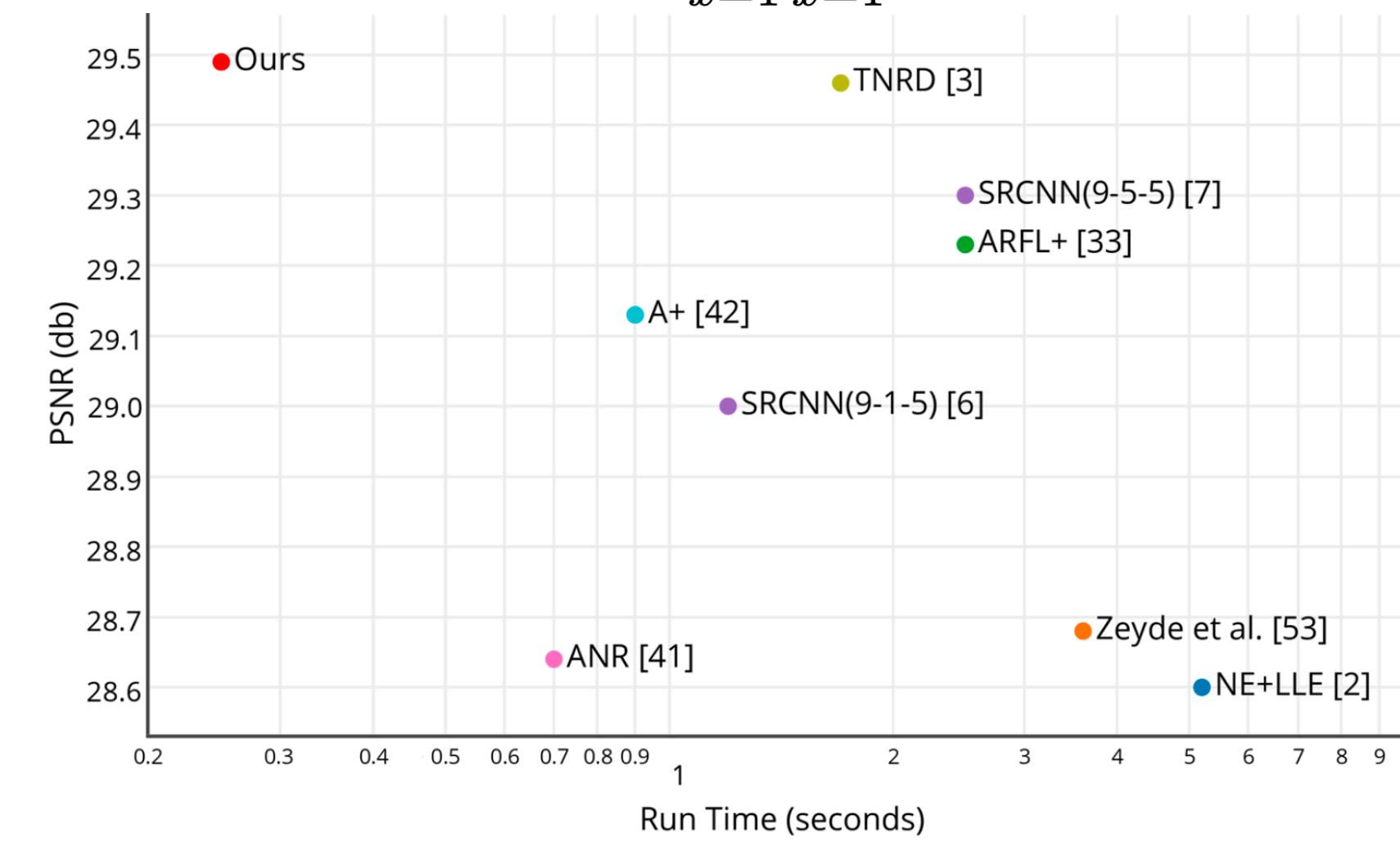
equivalent to a fractionally-strided convolution

convolution with stride  $1/r$  in the LR space

with a filter  $W_s$  of size  $k_s = rk_L$ , but more efficient!



$$\ell(W_{1:L}, b_{1:L}) = \frac{1}{r^2 H W} \sum_{x=1}^{rH} \sum_{y=1}^{rW} (\mathbf{I}_{x,y}^{\text{HR}} - f_{x,y}^L(\mathbf{I}^{\text{LR}}))^2$$



Shi, Wenzhe, et al. "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network."

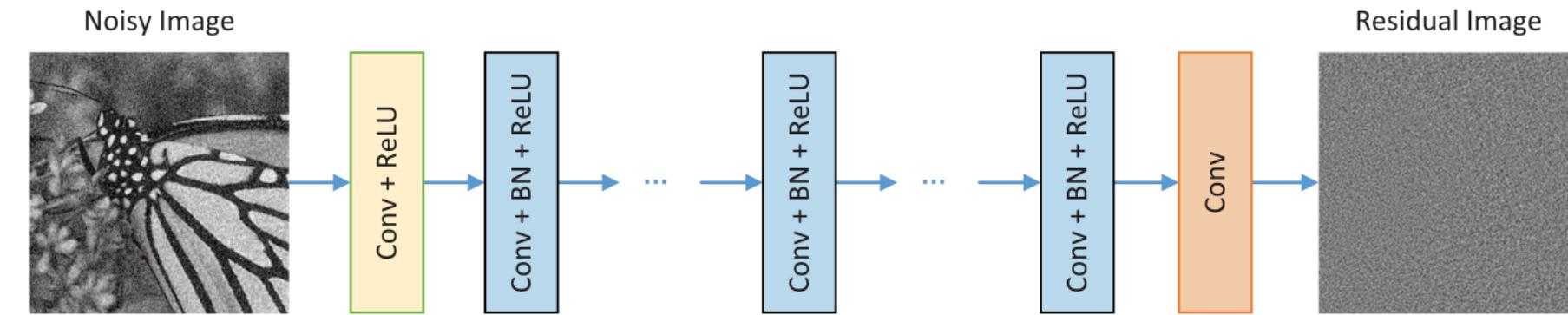
Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

33.66db

# Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising


[YouTube Video](#)

## Denoising Convolutional Neural Networks (DnCNNs)



$x \rightarrow$  clean image

$y \rightarrow$  noisy observation

$y = x + v \rightarrow$  image degradation model

AWGN (Additive White Gaussian Noise)  
with standard deviation  $\sigma$

Receptive Field of DnCNN ( $3 \times 3$  conv)

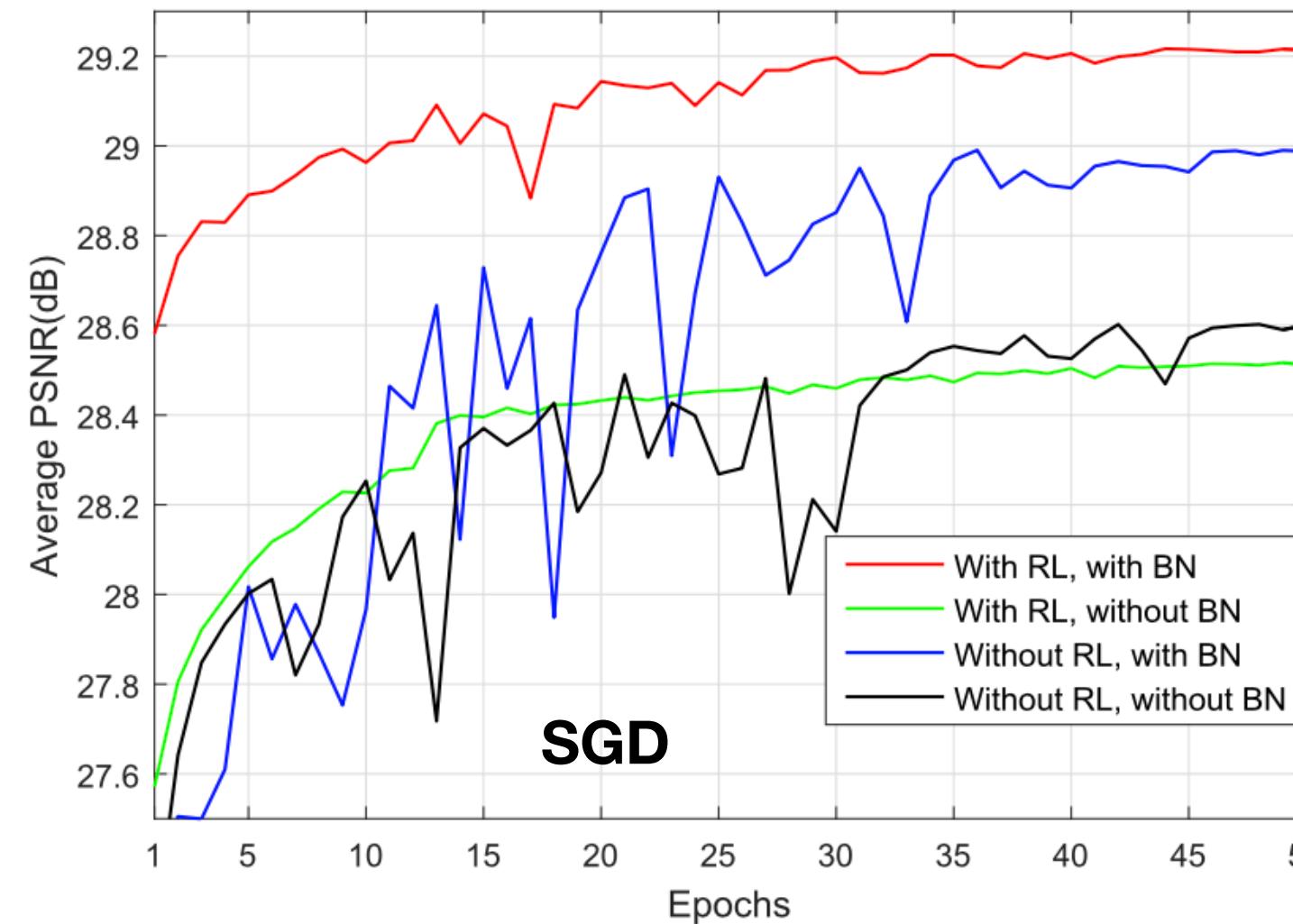
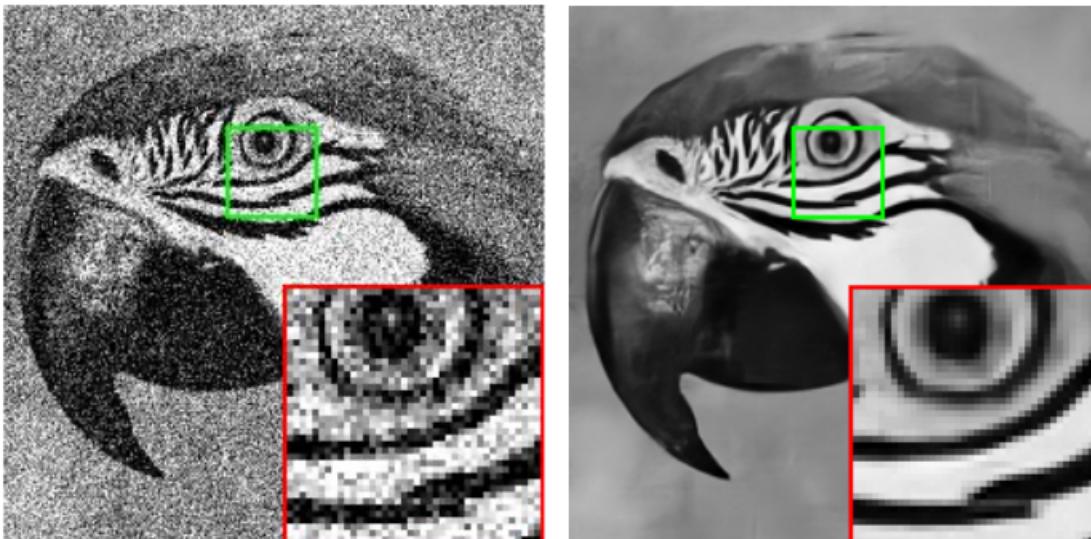
with depth  $d$  is  $(2d + 1) \times (2d + 1)$

$\mathcal{F}(y) = x$

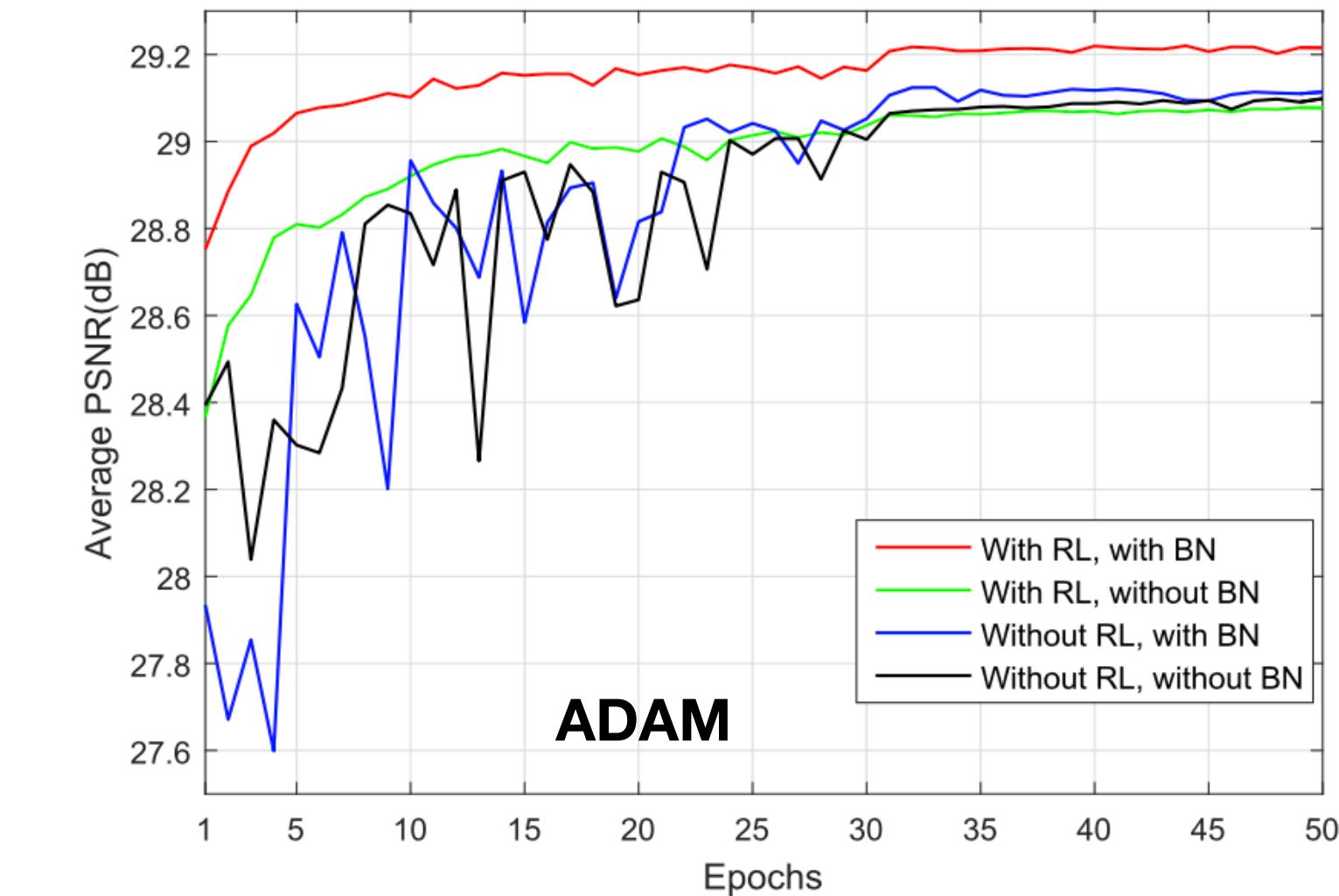
$\mathcal{R}(y) \approx v \rightarrow$  residual mapping

$\Rightarrow x = y - \mathcal{R}(y)$

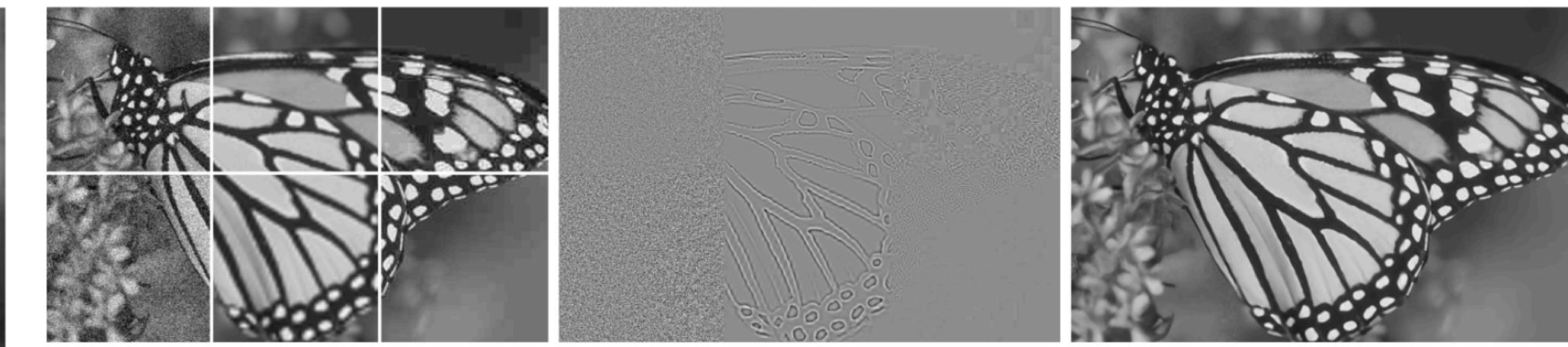
$$\ell(\theta) = \frac{1}{2N} \sum_{i=1}^N \|R(y_i; \theta) - (y_i - x_i)\|_F^2$$



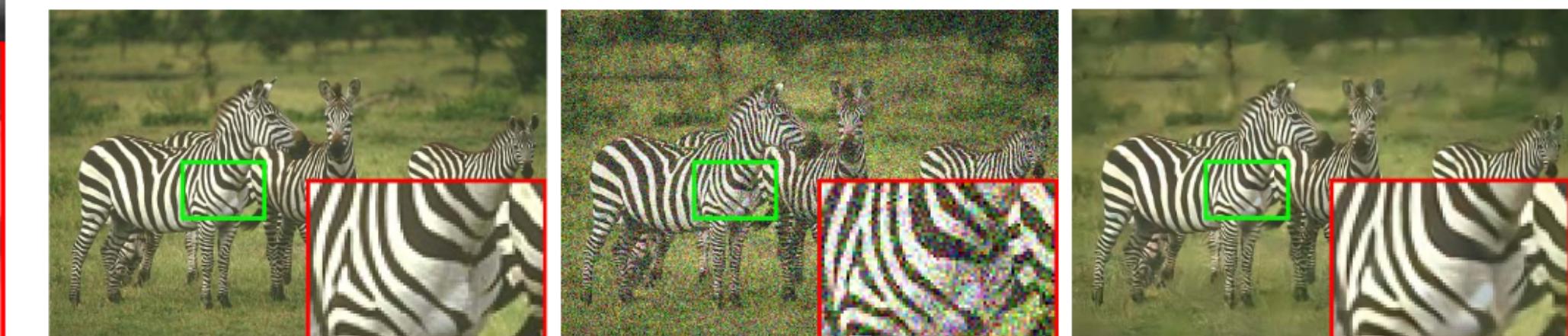
SGD



ADAM



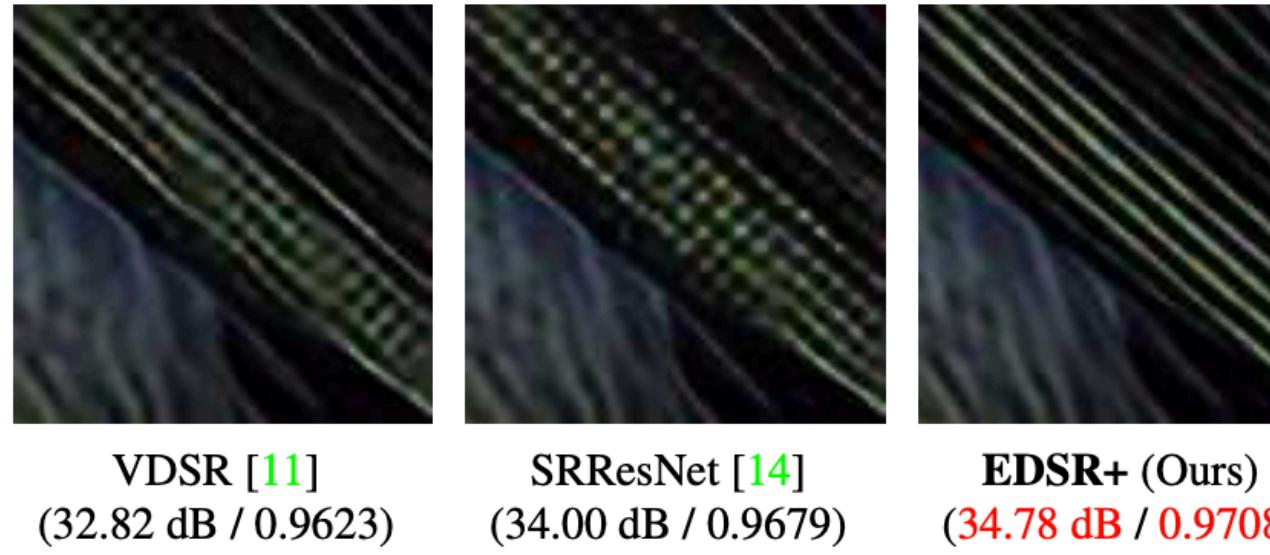
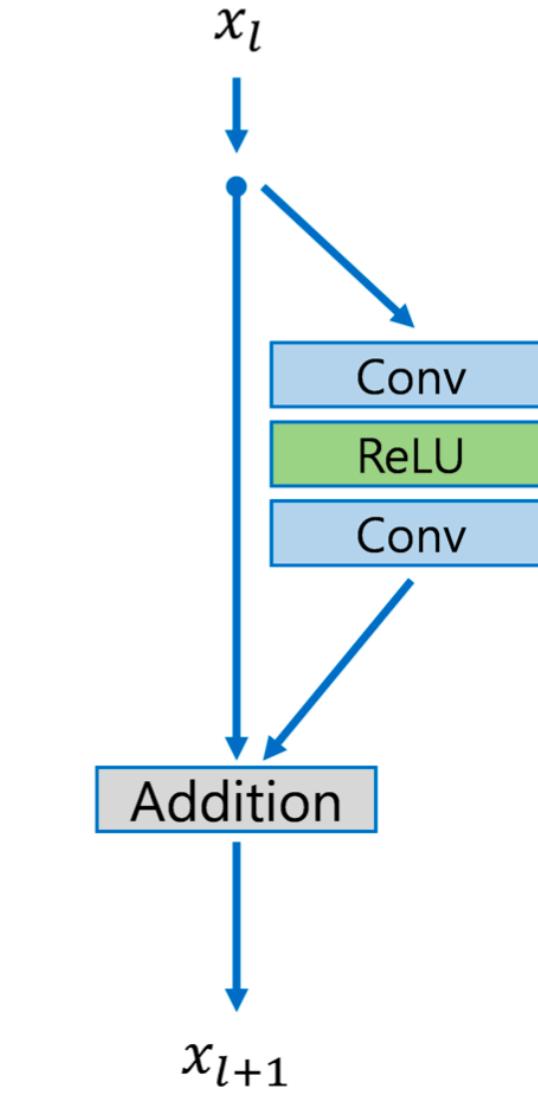
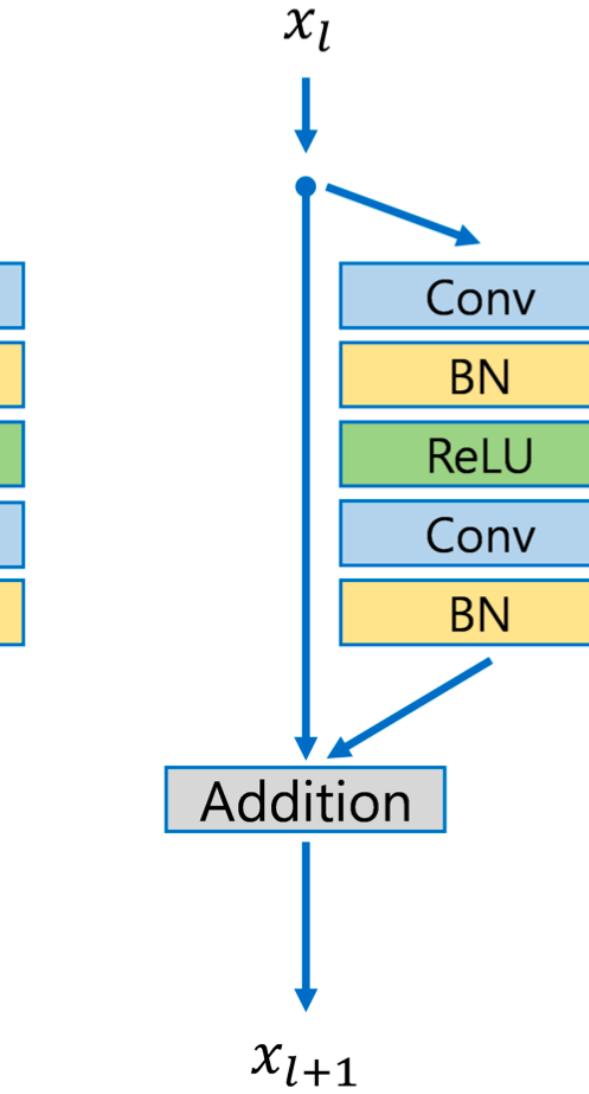
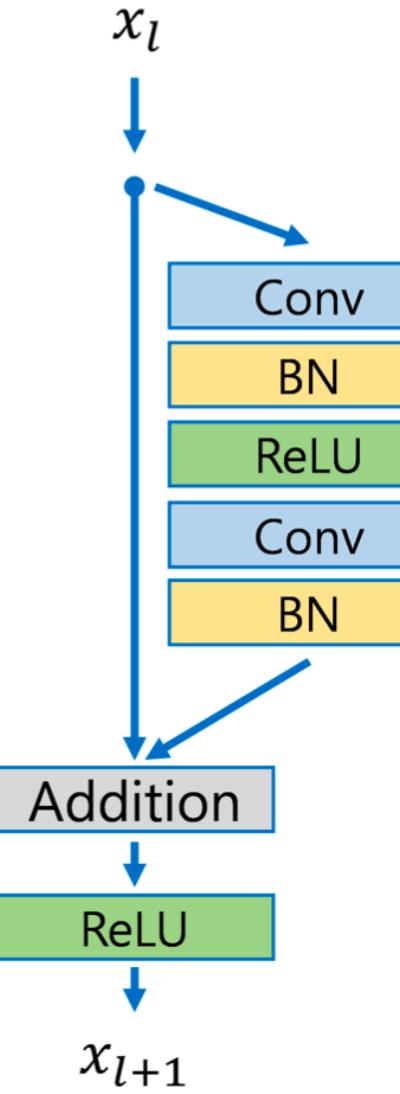
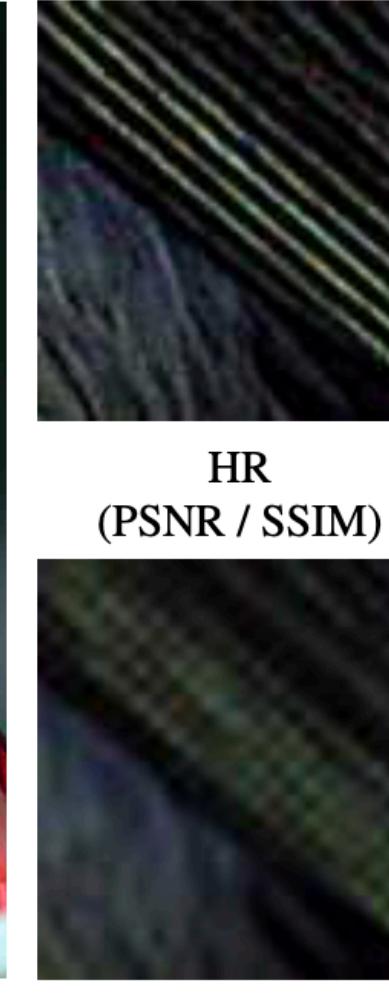
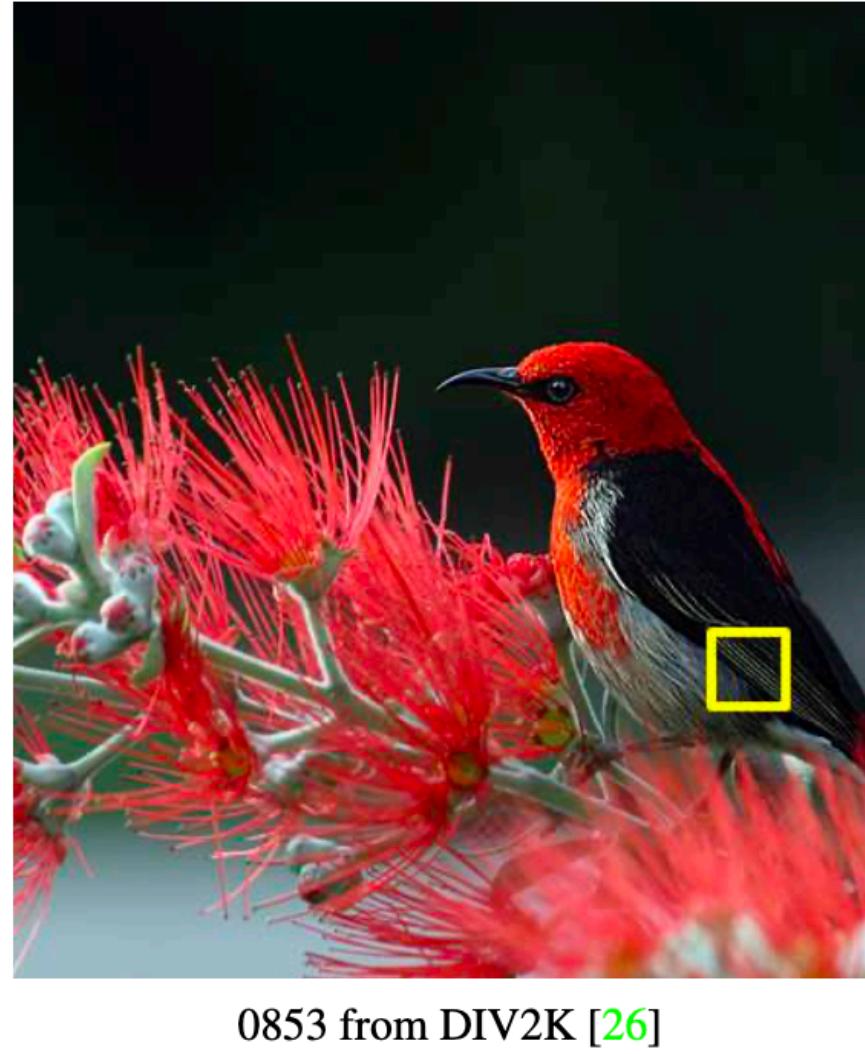
JPEG image de-blocking  
Single Image Super-resolution  
Gaussian Denoising





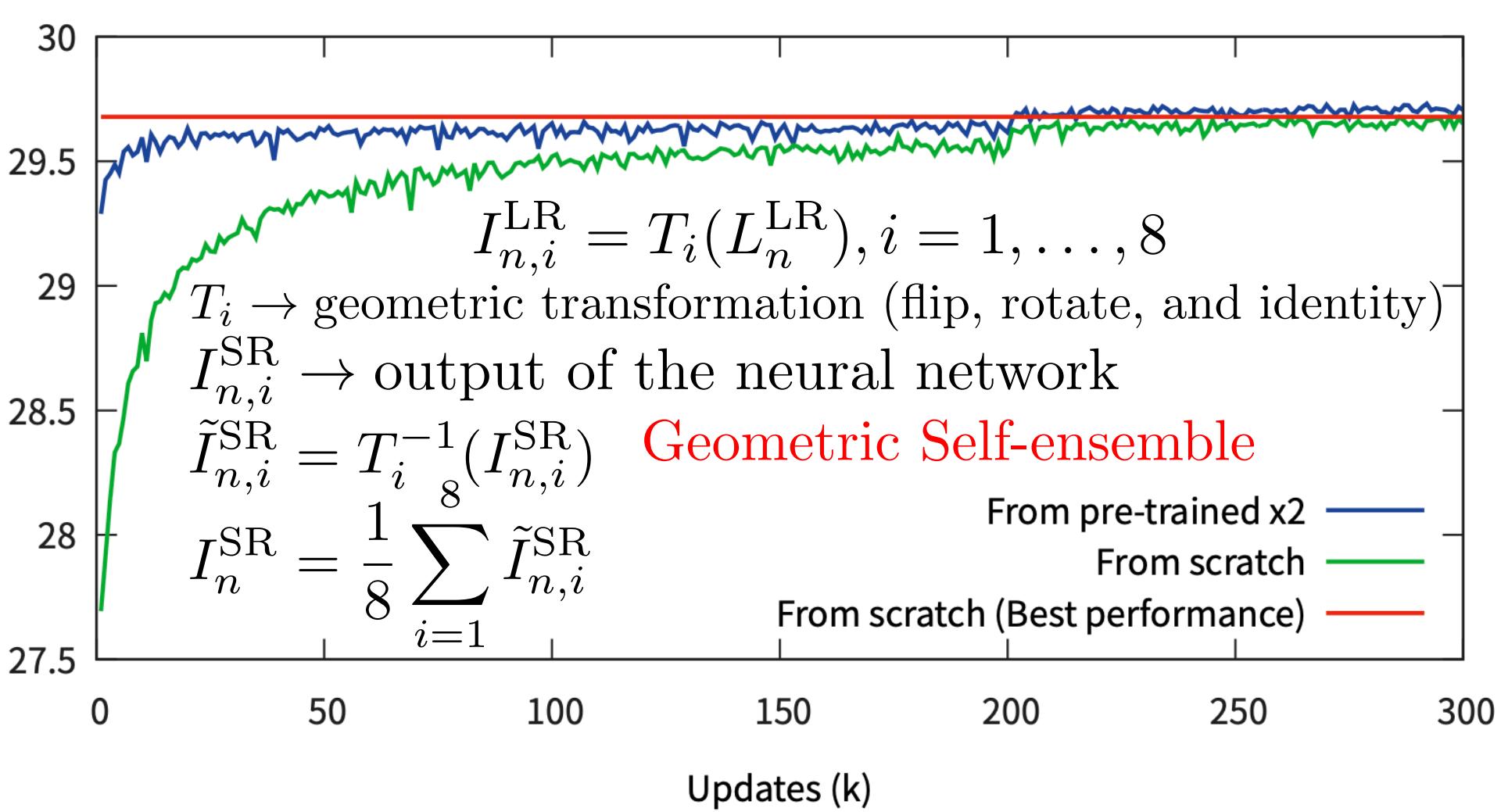
Boulder

# Enhanced Deep Residual Networks for Single Image Super-Resolution

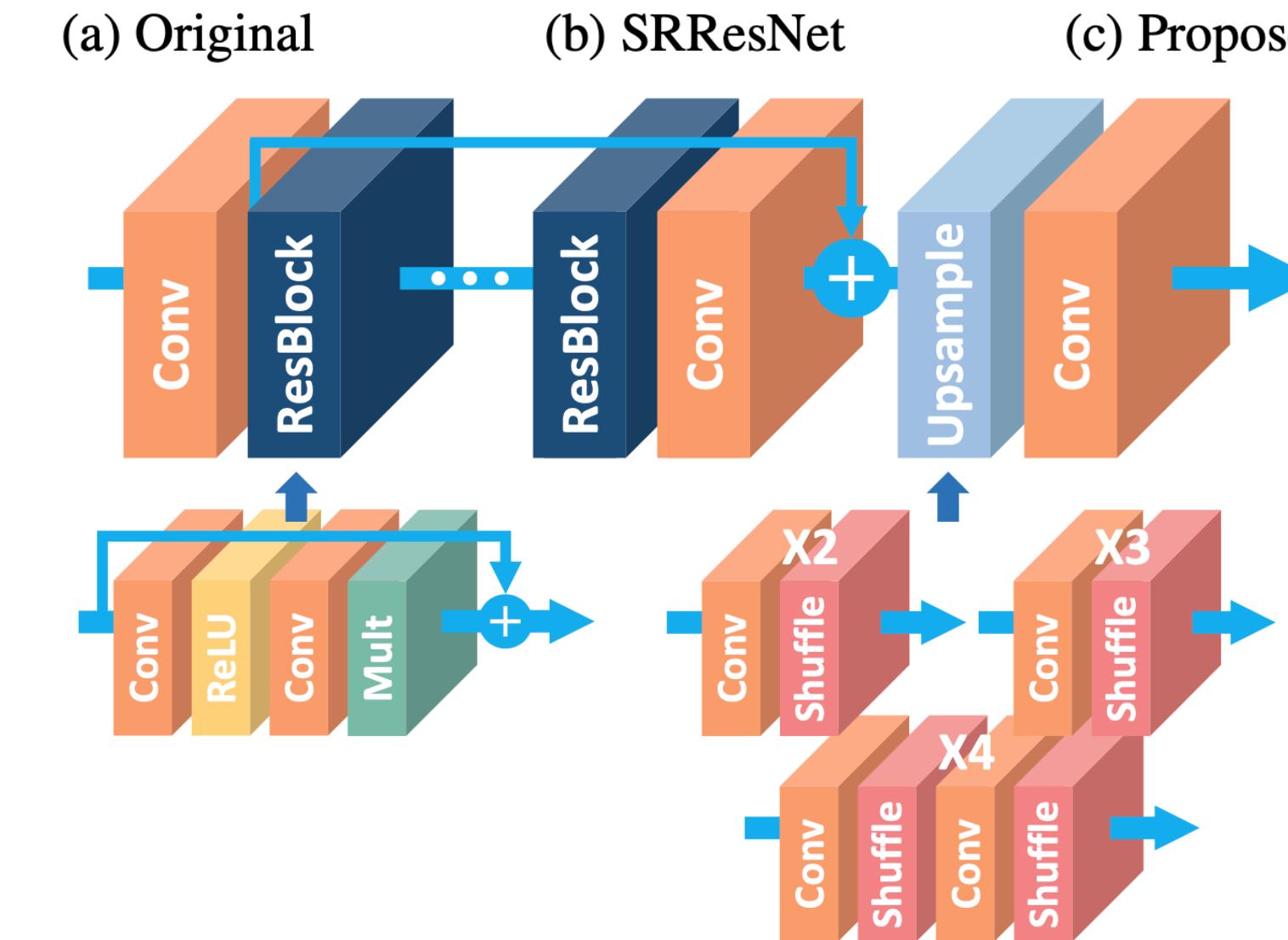


Options	SRResNet [14] (reproduced)	Baseline (Single / Multi)	EDSR	MDSR
# Residual blocks	16	16	32	80
# Filters	64	64	256	64
# Parameters	1.5M	1.5M / 3.2M	43M	8.0M
Residual scaling	-	-	0.1	-
Use BN	Yes	No	No	No
Loss function	L2	L1	L1	L1

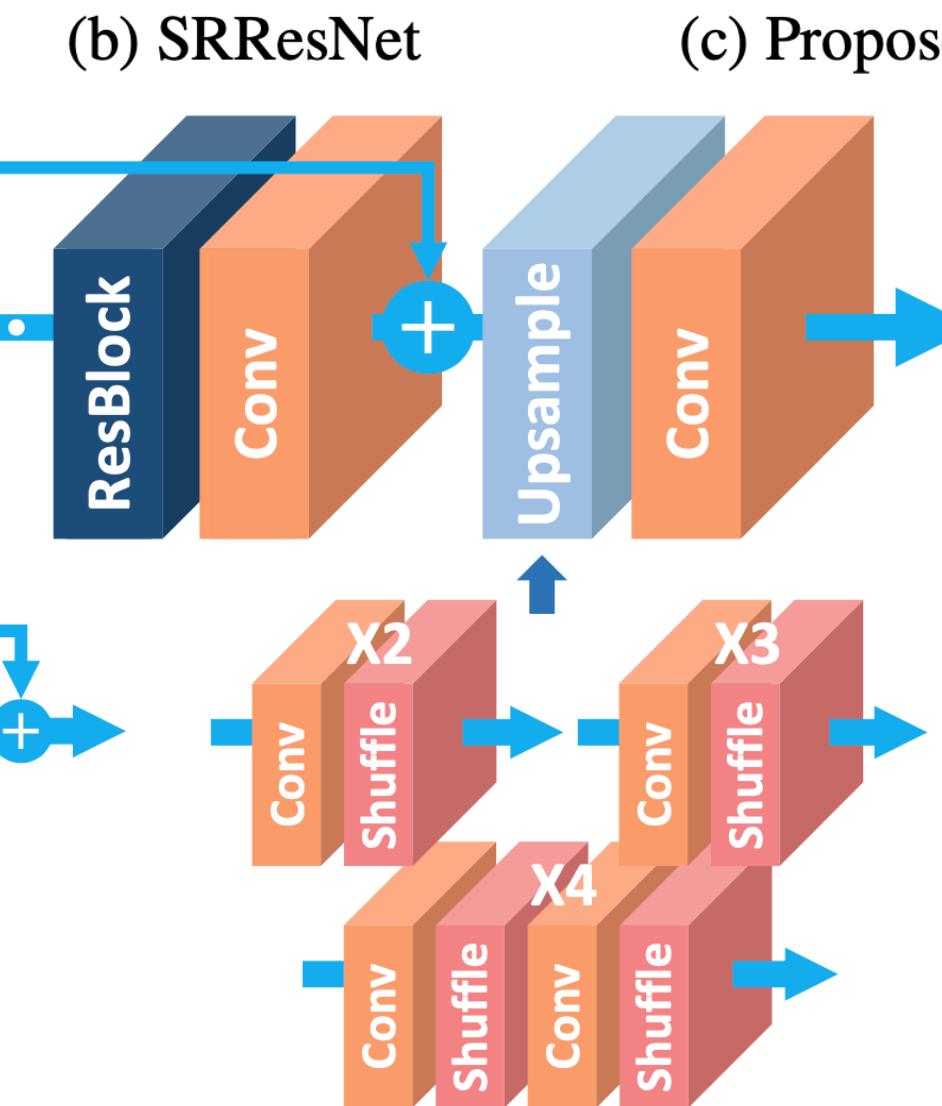
PSNR(dB) on DIV2K validation set (x4)



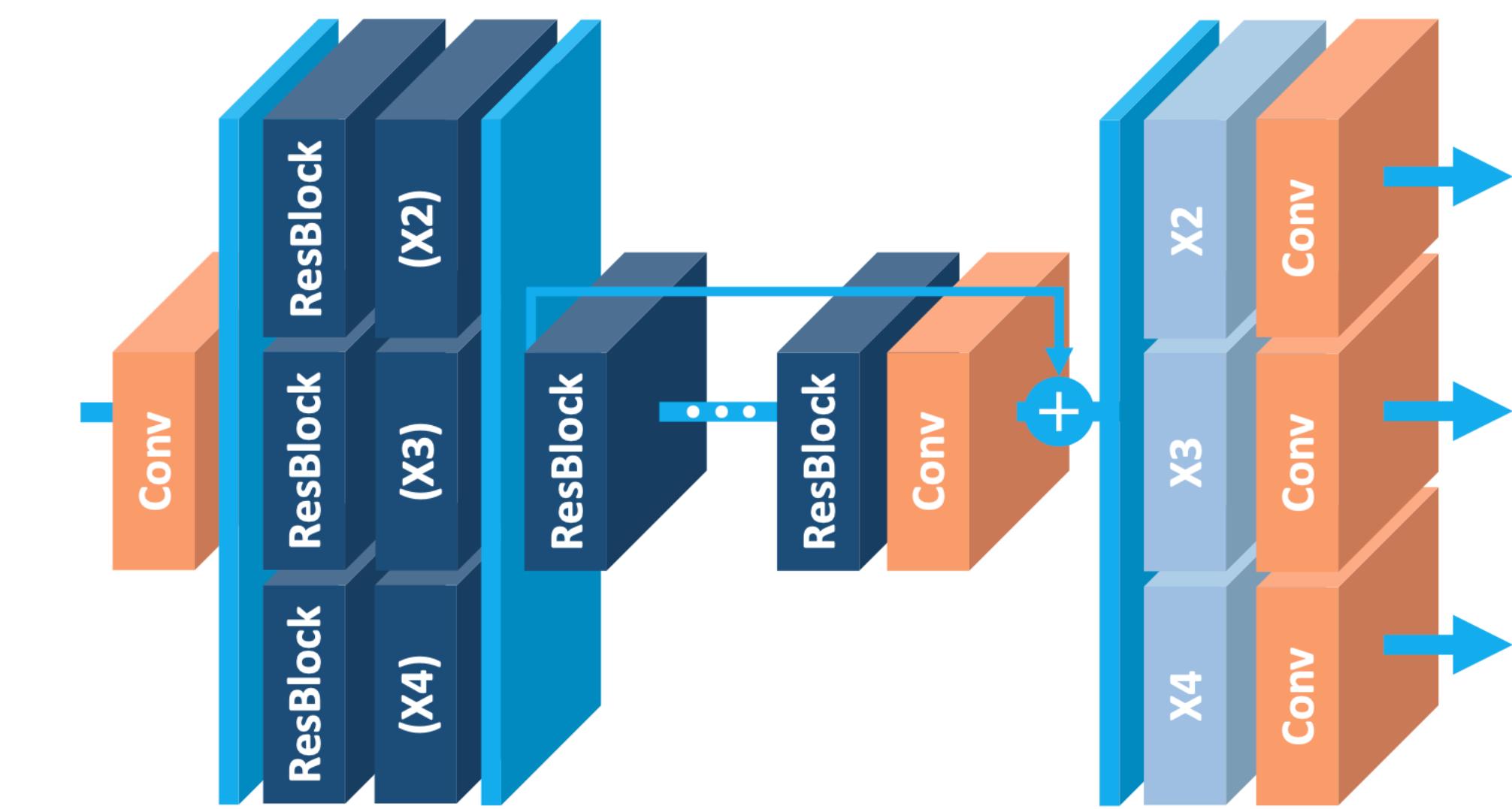
(a) Original



(b) SRResNet



(c) Proposed





Boulder

# Deep Image Prior

Zero training data, zero validation data, and one test data!

$$x^* = \arg \min_x \|d(x) - x_0\|^2 + R(x)$$

$x^* \in \mathbb{R}^{rH \times rW \times C}$  → super-resolved image

$x_0 \in \mathbb{R}^{H \times W \times C}$  → low resolution image

$x \in \mathbb{R}^{rH \times rW \times C}$

$d : \mathbb{R}^{rH \times rW \times C} \rightarrow \mathbb{R}^{H \times W \times C}$

└ downsampling operator

$R(x)$  → regularizer (e.g., total variation (TV) of the image)

Use a CNN as the regularizer!

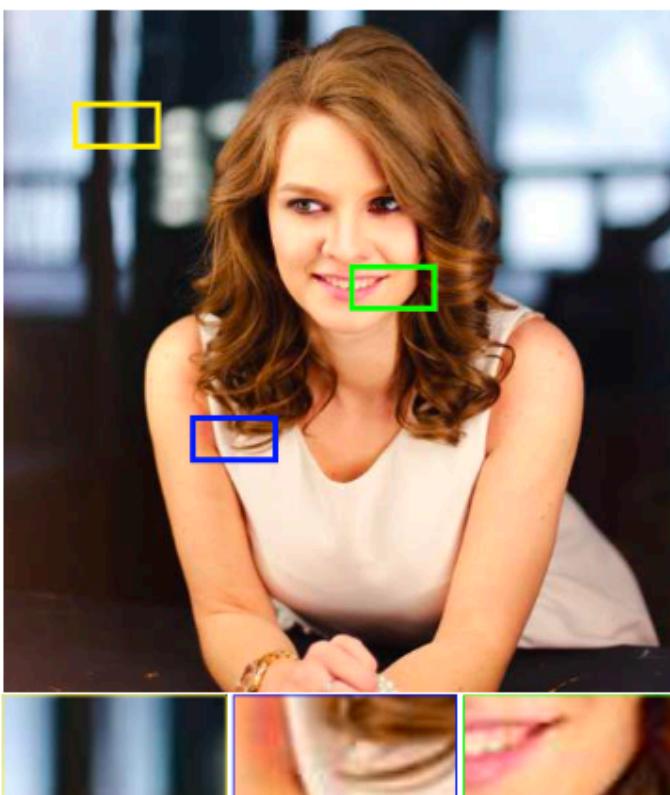
$R(x) = 0$  when  $x = f_\theta(z)$  and  $R(x) = +\infty$  otherwise

$z$  → a vector filled with uniform noise (fixed)

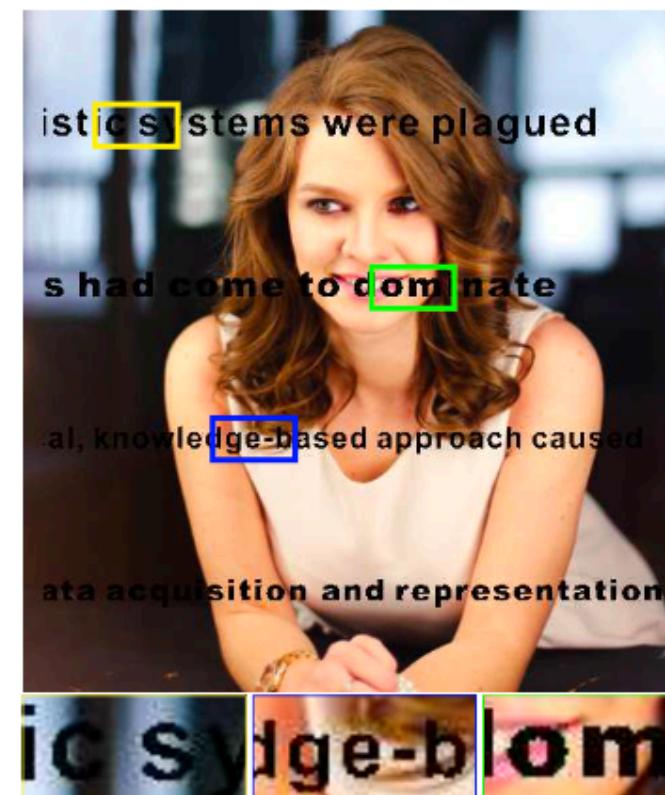
$f_\theta$  → CNN

$$\theta^* = \arg \min_\theta \|d(f_\theta(z)) - x_0\|^2$$

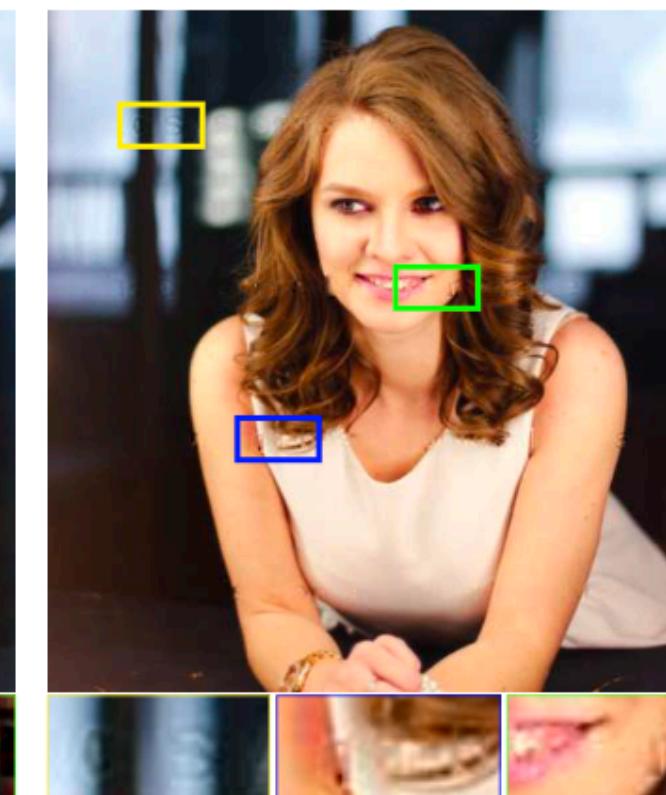
$$x^* = f_{\theta^*}(z)$$



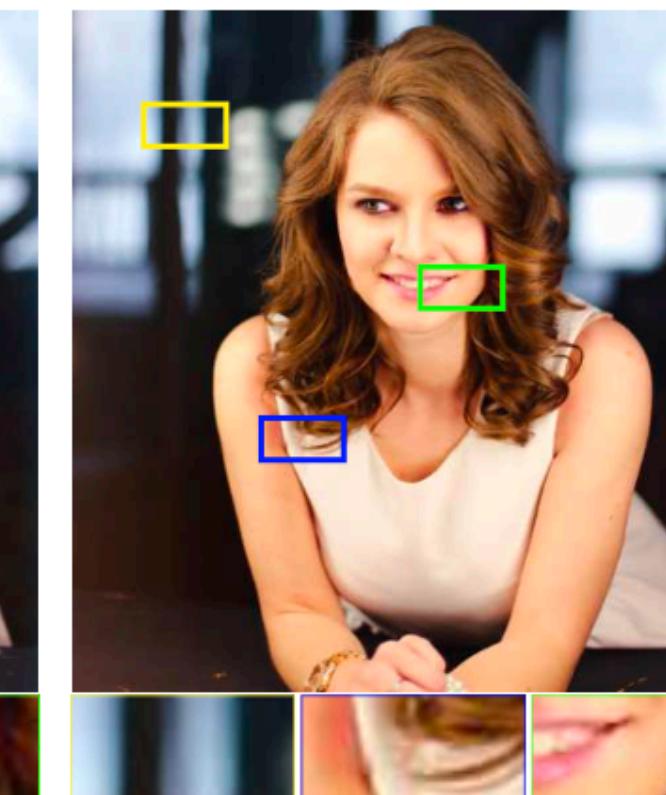
(a) Original image



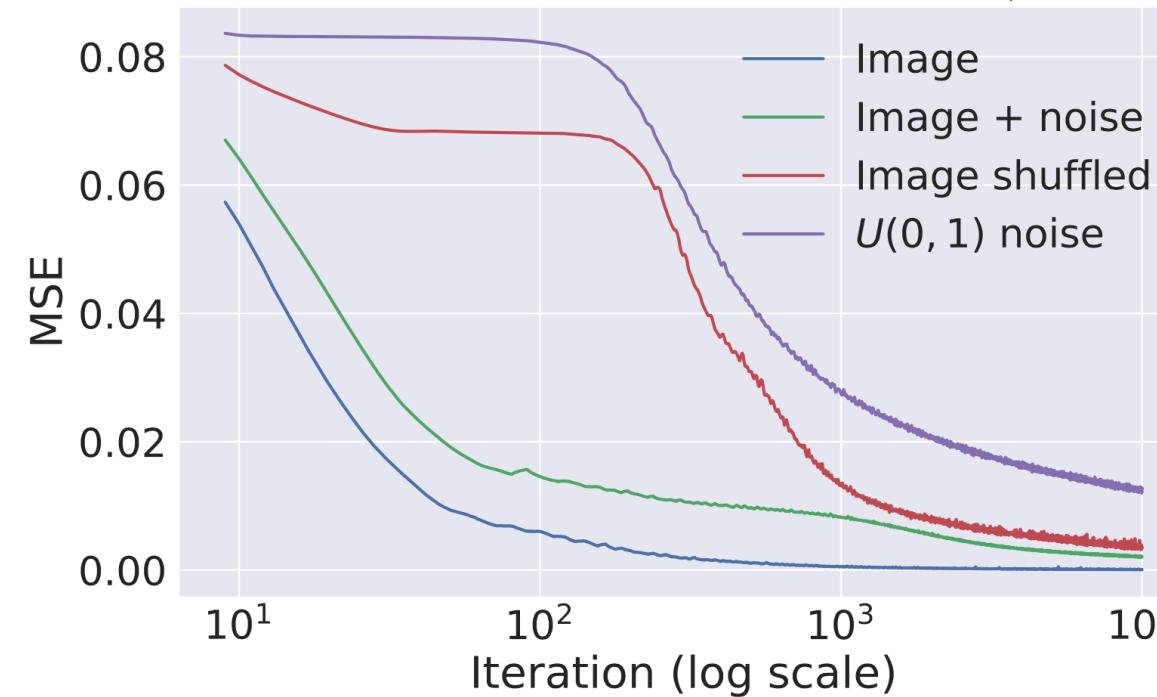
(b) Corrupted image



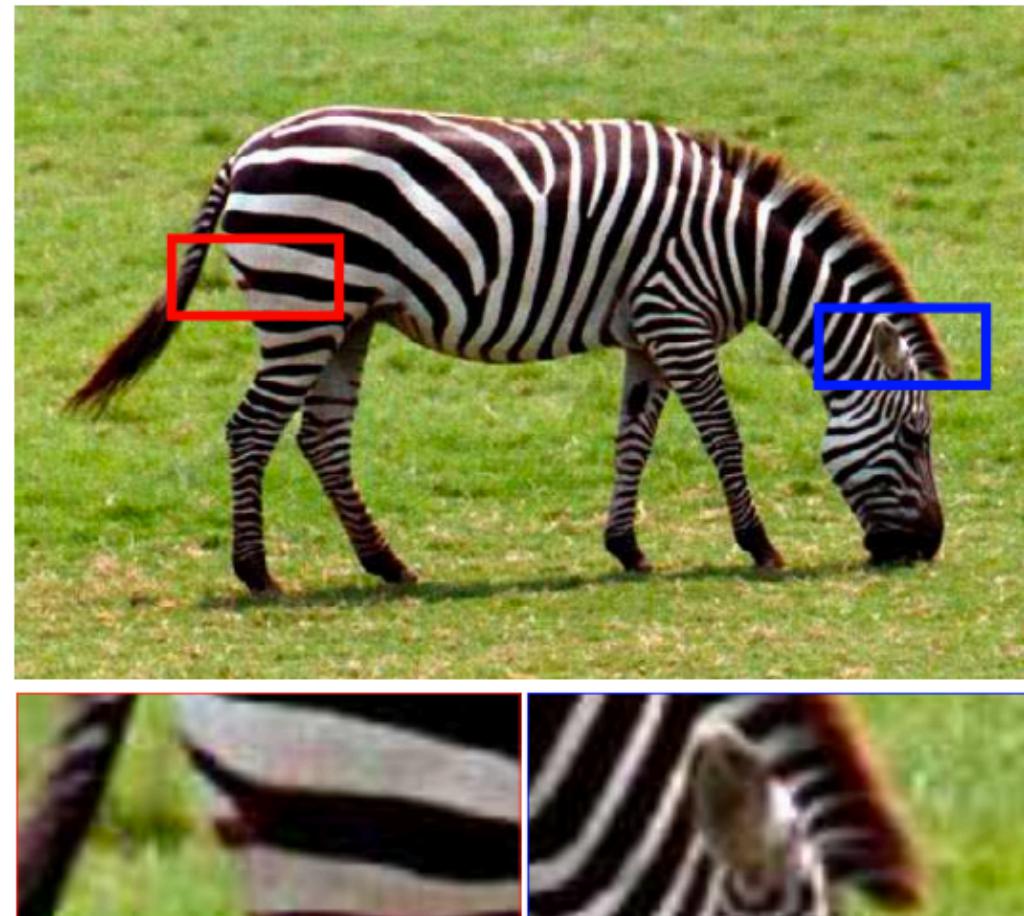
(c) Sheppard networks [27]



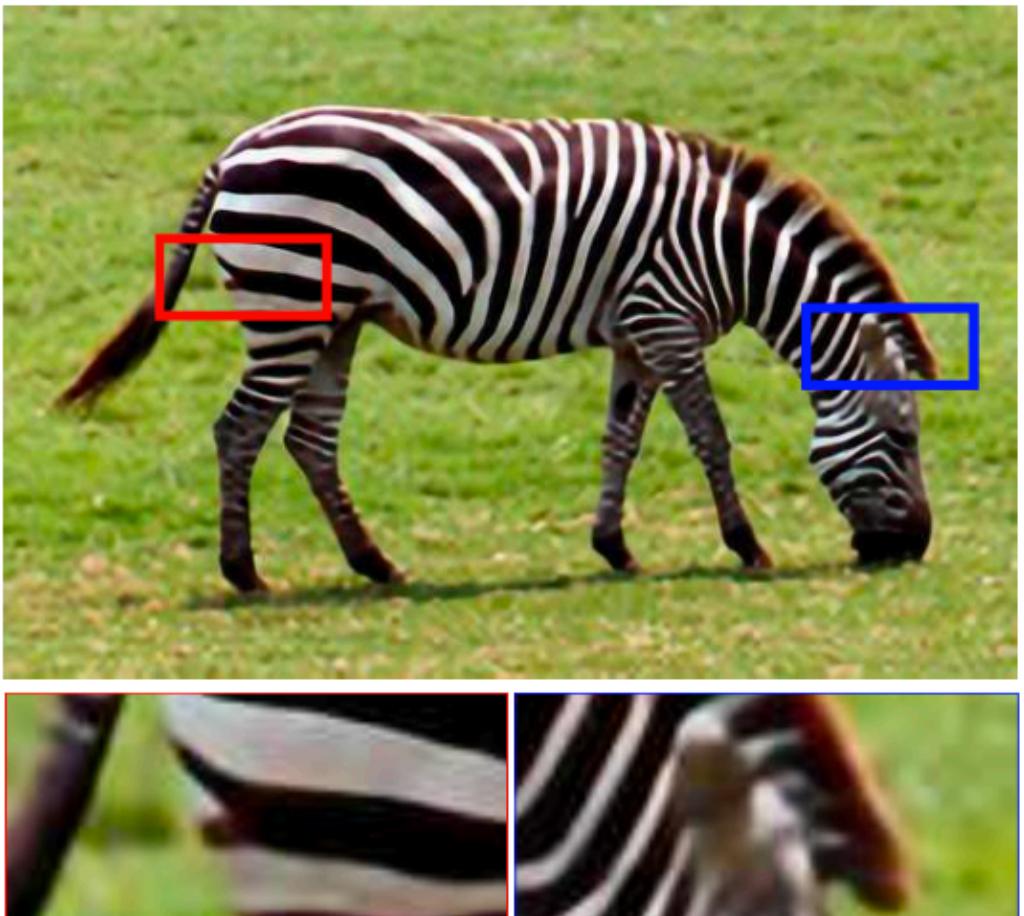
(d) Deep Image Prior



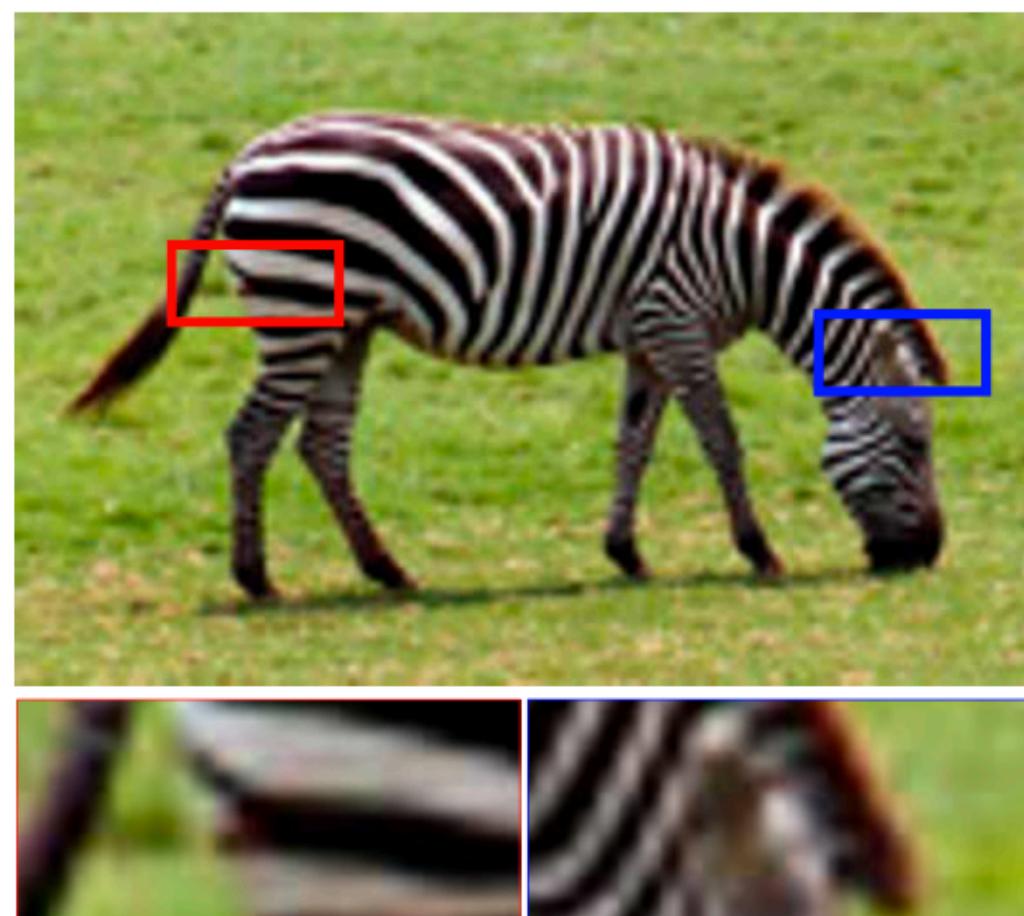
The network has a harder time overfitting noise than overfitting to natural images!



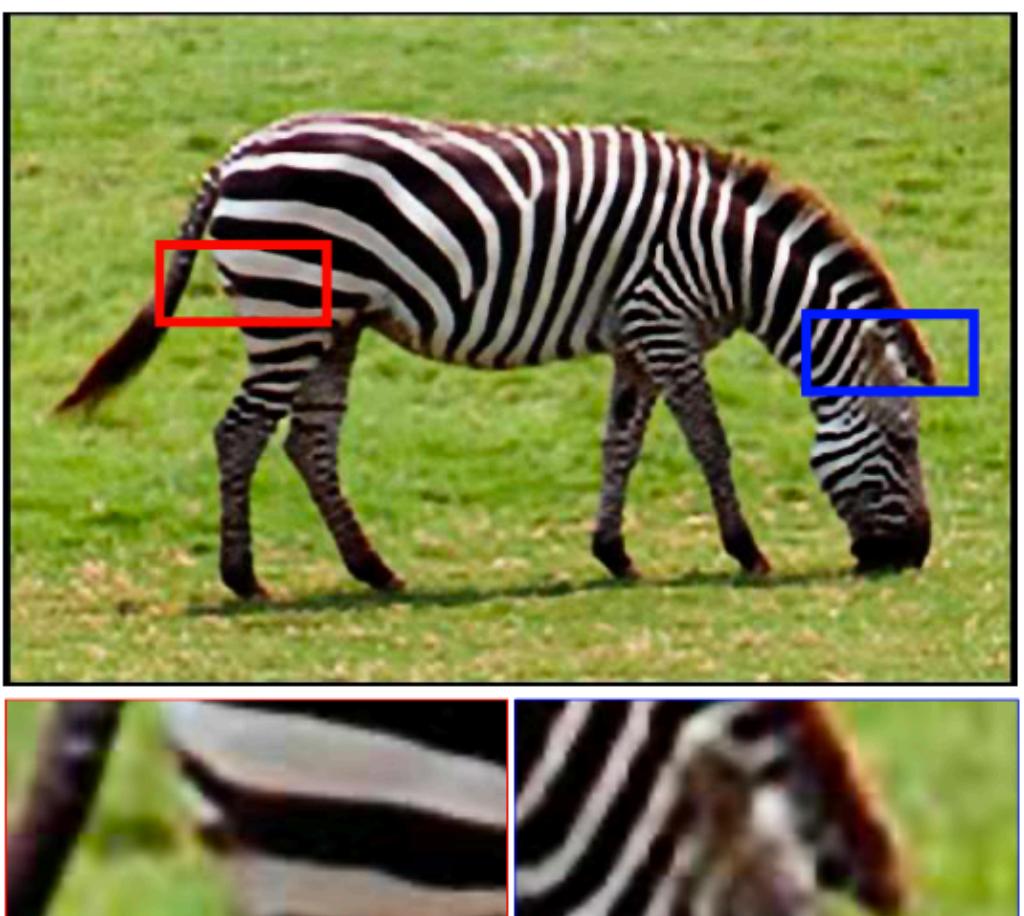
(a) Ground truth



(b) SRResNet [19], Trained



(c) Bicubic, Not trained



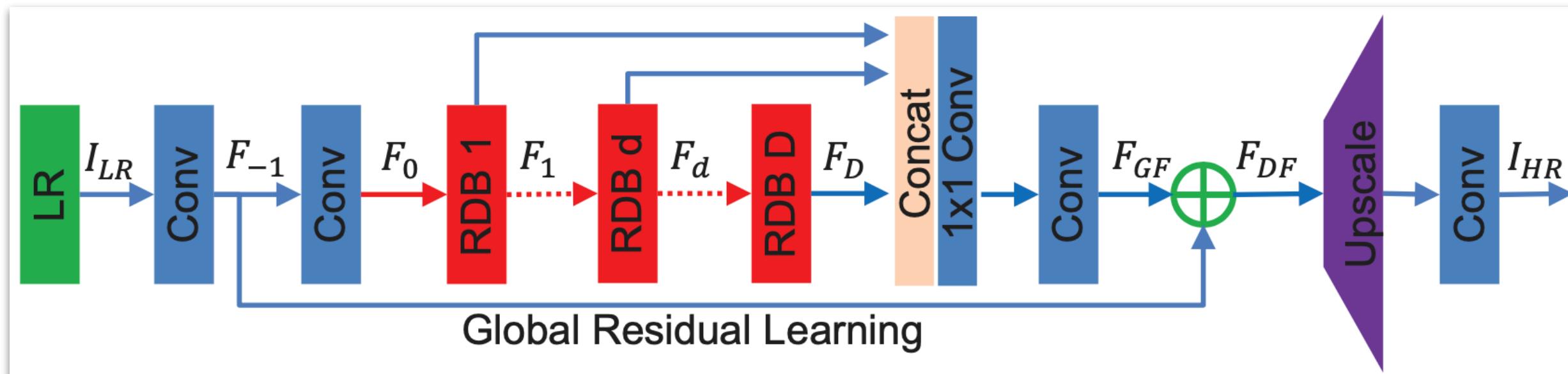
(d) Deep prior, Not trained



Boulder

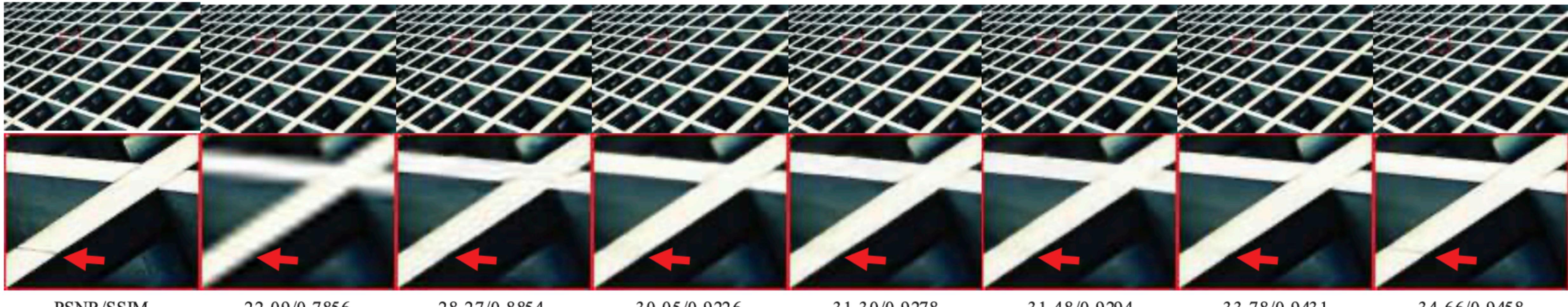
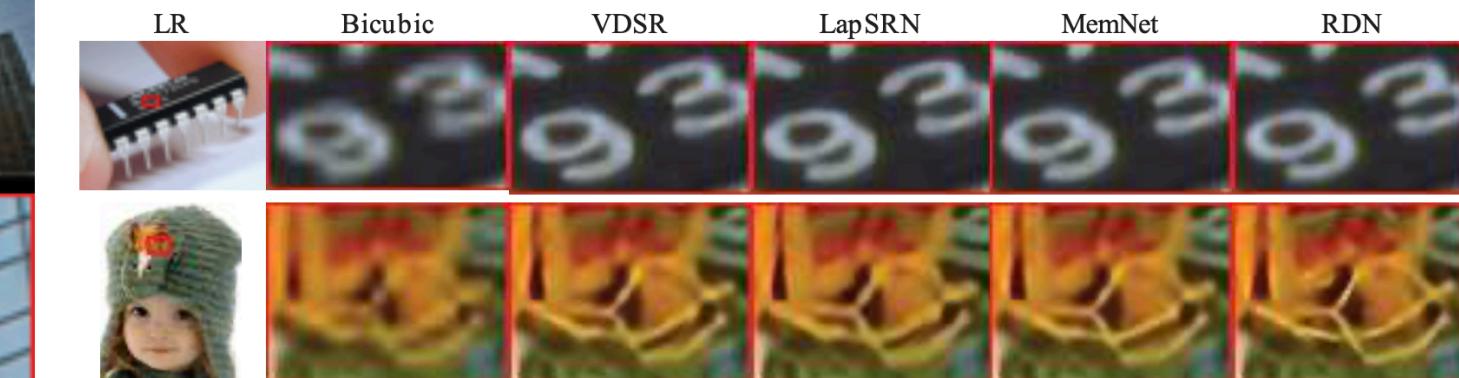
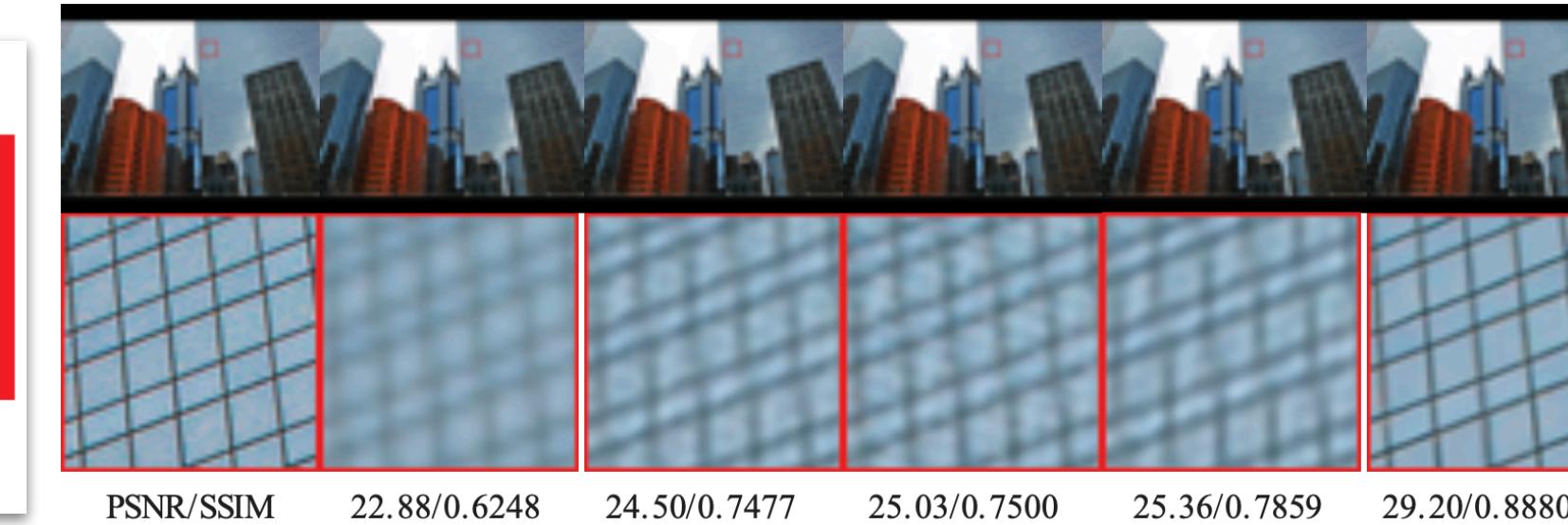
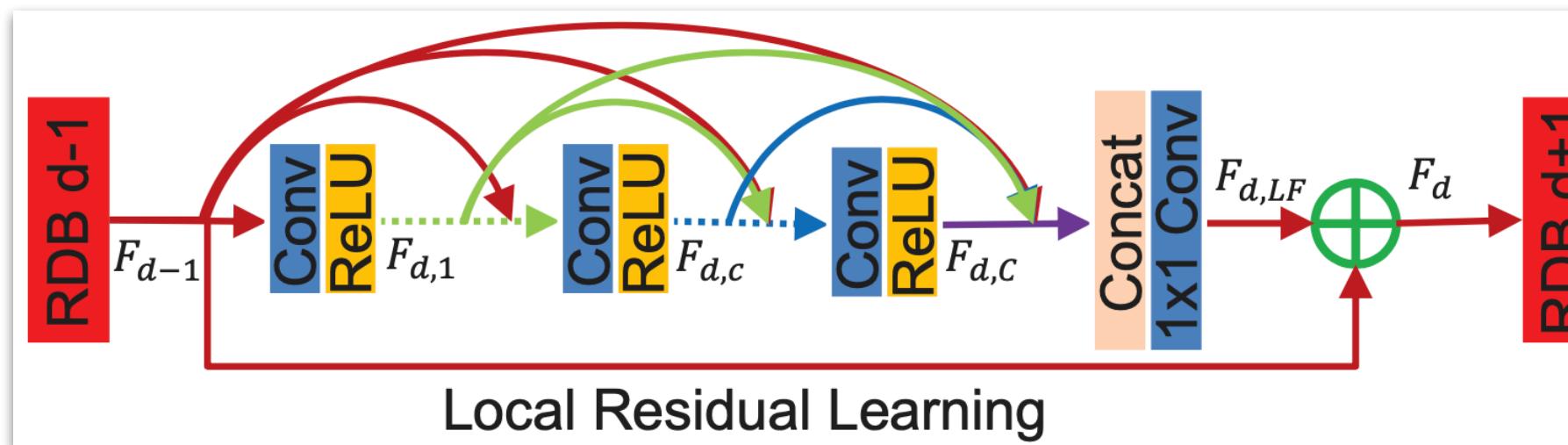
# Residual Dense Network for Image Super-Resolution

## Residual Dense Network (RDN)



Dataset	Scale	Bicubic	SRCCN [3]	LapSRN [13]	DRRN [25]	SRDenseNet [31]	MemNet [26]	MDSR [17]	RDN (ours)	RDN+ (ours)
Set5	$\times 2$	33.66/0.9299	36.66/0.9542	37.52/0.9591	37.74/0.9591	-/-	37.78/0.9597	38.11/0.9602	38.24/0.9614	<b>38.30/0.9616</b>
	$\times 3$	30.39/0.8682	32.75/0.9090	33.82/0.9227	34.03/0.9244	-/-	34.09/0.9248	34.66/0.9280	34.71/0.9296	<b>34.78/0.9300</b>
	$\times 4$	28.42/0.8104	30.48/0.8628	31.54/0.8855	31.68/0.8888	32.02/0.8934	31.74/0.8893	32.50/0.8973	32.47/0.8990	<b>32.61/0.9003</b>
Set14	$\times 2$	30.24/0.8688	32.45/0.9067	33.08/0.9130	33.23/0.9136	-/-	33.28/0.9142	33.85/0.9198	34.01/0.9212	<b>34.10/0.9218</b>
	$\times 3$	27.55/0.7742	29.30/0.8215	29.79/0.8320	29.96/0.8349	-/-	30.00/0.8350	30.44/0.8452	30.57/0.8468	<b>30.67/0.8482</b>
	$\times 4$	26.00/0.7027	27.50/0.7513	28.19/0.7720	28.21/0.7721	28.50/0.7782	28.26/0.7723	28.72/0.7857	28.81/0.7871	<b>28.92/0.7893</b>
B100	$\times 2$	29.56/0.8431	31.36/0.8879	31.80/0.8950	32.05/0.8973	-/-	32.08/0.8978	32.29/0.9007	32.34/0.9017	<b>32.40/0.9022</b>
	$\times 3$	27.21/0.7385	28.41/0.7863	28.82/0.7973	28.95/0.8004	-/-	28.96/0.8001	29.25/0.8091	29.26/0.8093	<b>29.33/0.8105</b>
	$\times 4$	25.96/0.6675	26.90/0.7101	27.32/0.7280	27.38/0.7284	27.53/0.7337	27.40/0.7281	27.72/0.7419	27.77/0.7434	<b>27.80/0.7434</b>
Urban100	$\times 2$	26.88/0.8403	29.50/0.8946	30.41/0.9101	31.23/0.9188	-/-	31.31/0.9195	32.84/0.9347	32.89/0.9353	<b>33.09/0.9368</b>
	$\times 3$	24.46/0.7349	26.24/0.7989	27.07/0.8272	27.53/0.8378	-/-	27.56/0.8376	28.79/0.8655	28.80/0.8653	<b>29.00/0.8683</b>
	$\times 4$	23.14/0.6577	24.52/0.7221	25.21/0.7553	25.44/0.7638	26.05/0.7819	25.50/0.7630	26.67/0.8041	26.61/0.8028	<b>26.82/0.8069</b>
Manga109	$\times 2$	30.80/0.9339	35.60/0.9663	37.27/0.9740	37.60/0.9736	-/-	37.72/0.9740	38.96/0.9769	39.18/0.9780	<b>39.38/0.9784</b>
	$\times 3$	26.95/0.8556	30.48/0.9117	32.19/0.9334	32.42/0.9359	-/-	32.51/0.9369	34.17/0.9473	34.13/0.9484	<b>34.43/0.9498</b>
	$\times 4$	24.89/0.7866	27.58/0.8555	29.09/0.8893	29.18/0.8914	-/-	29.42/0.8942	31.11/0.9148	31.00/0.9151	<b>31.39/0.9184</b>

## Residual Dense Block (RDB)

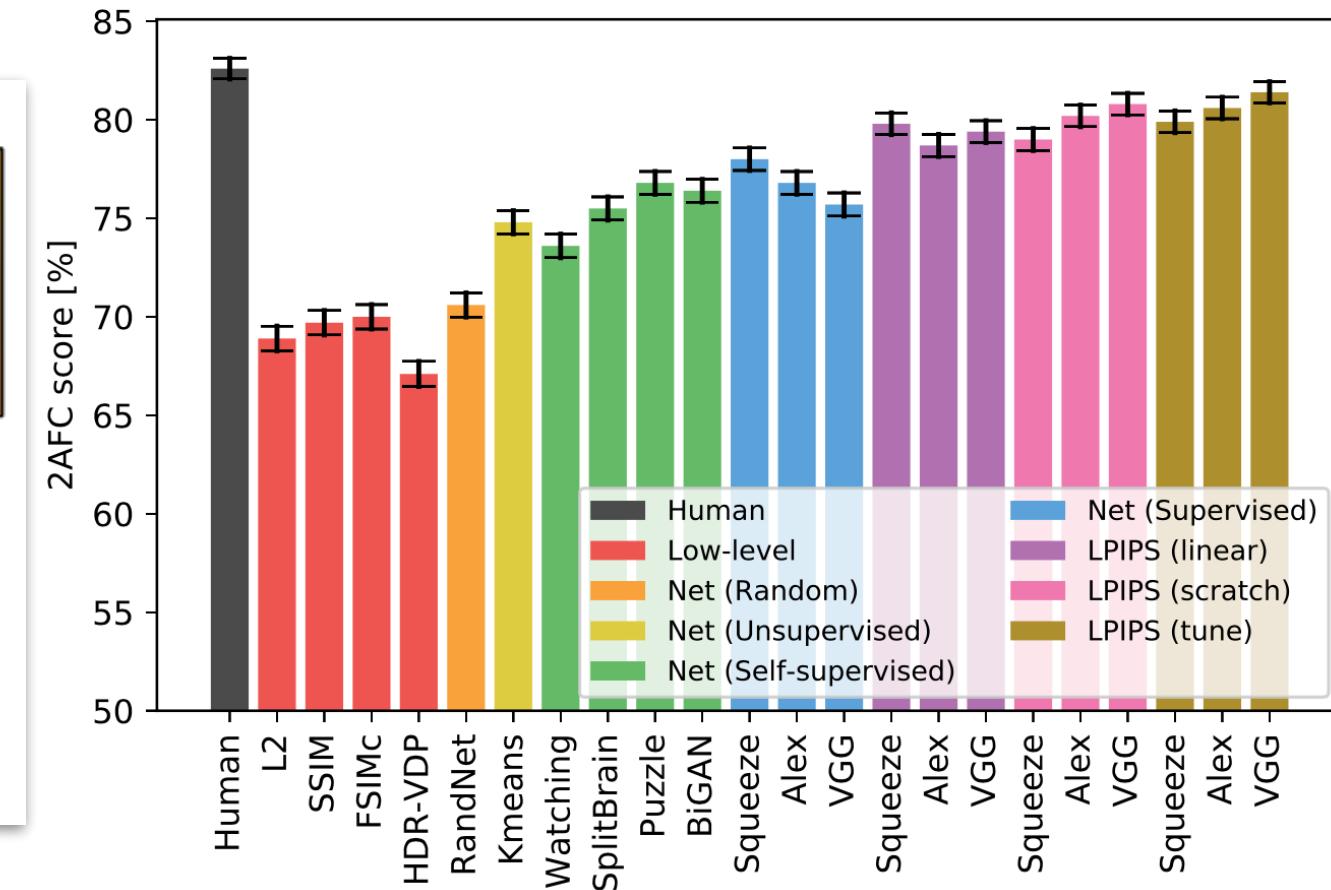
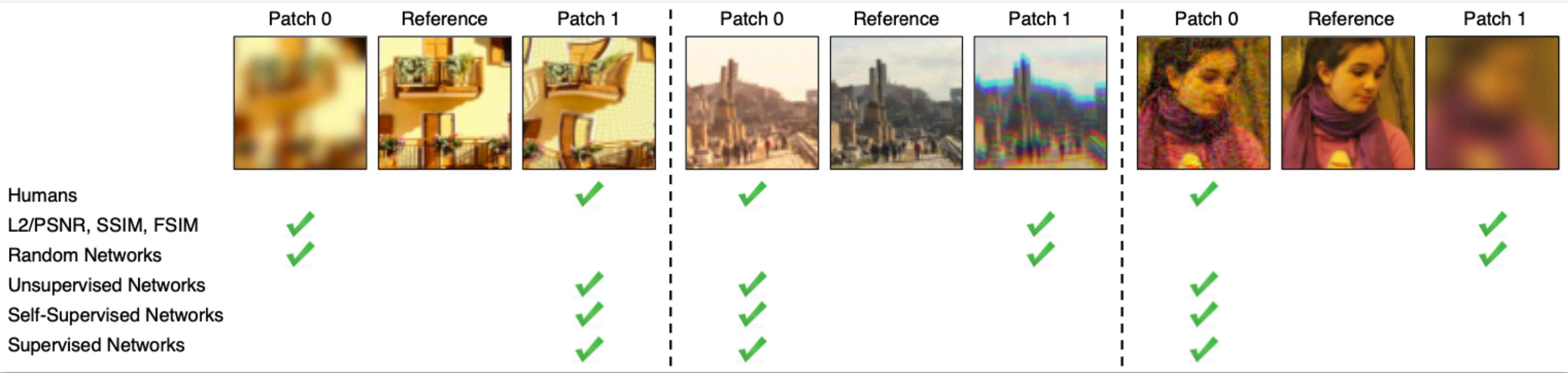




Boulder

# The Unreasonable Effectiveness of Deep Features as a Perceptual Metric

Which patch (left or right) is “closer” to the middle patch in these examples?



Perceptual Distance: Is a red circle more similar to a red square or to a blue circle?

Berkeley-Adobe Perceptual Patch Similarity (BAPPS) Dataset

– Two Alternative Forced Choice (2AFC): Which of two distortions is more similar to a reference?

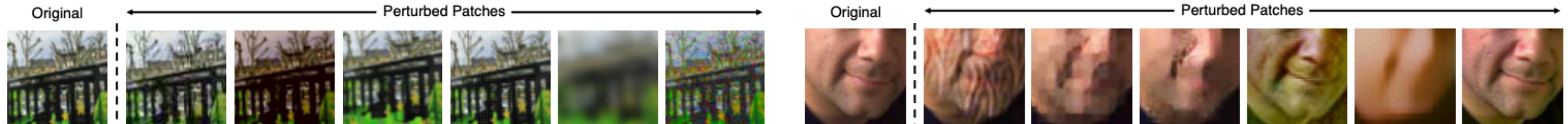
$x \rightarrow$  randomly selected image patch

$x_0, x_1 \rightarrow$  distorted patches

$h \in \{0, 1\} \rightarrow$  response of a human (which one is closer to the original patch?)

$\mathcal{T} \rightarrow$  dataset of patch triplets  $(x, x_0, x_1, h)$

– Just Noticeable Difference (JND): Are two patches (one reference and one distorted) the same?

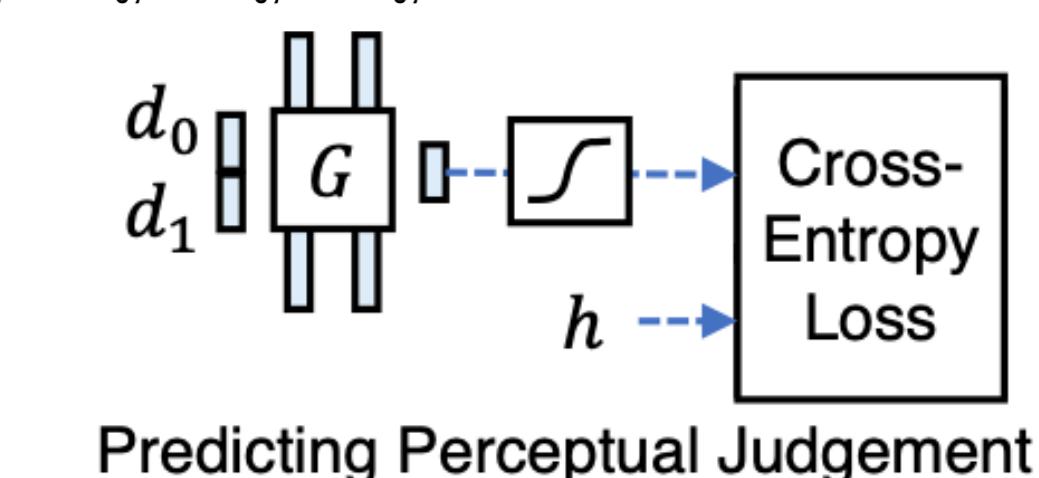
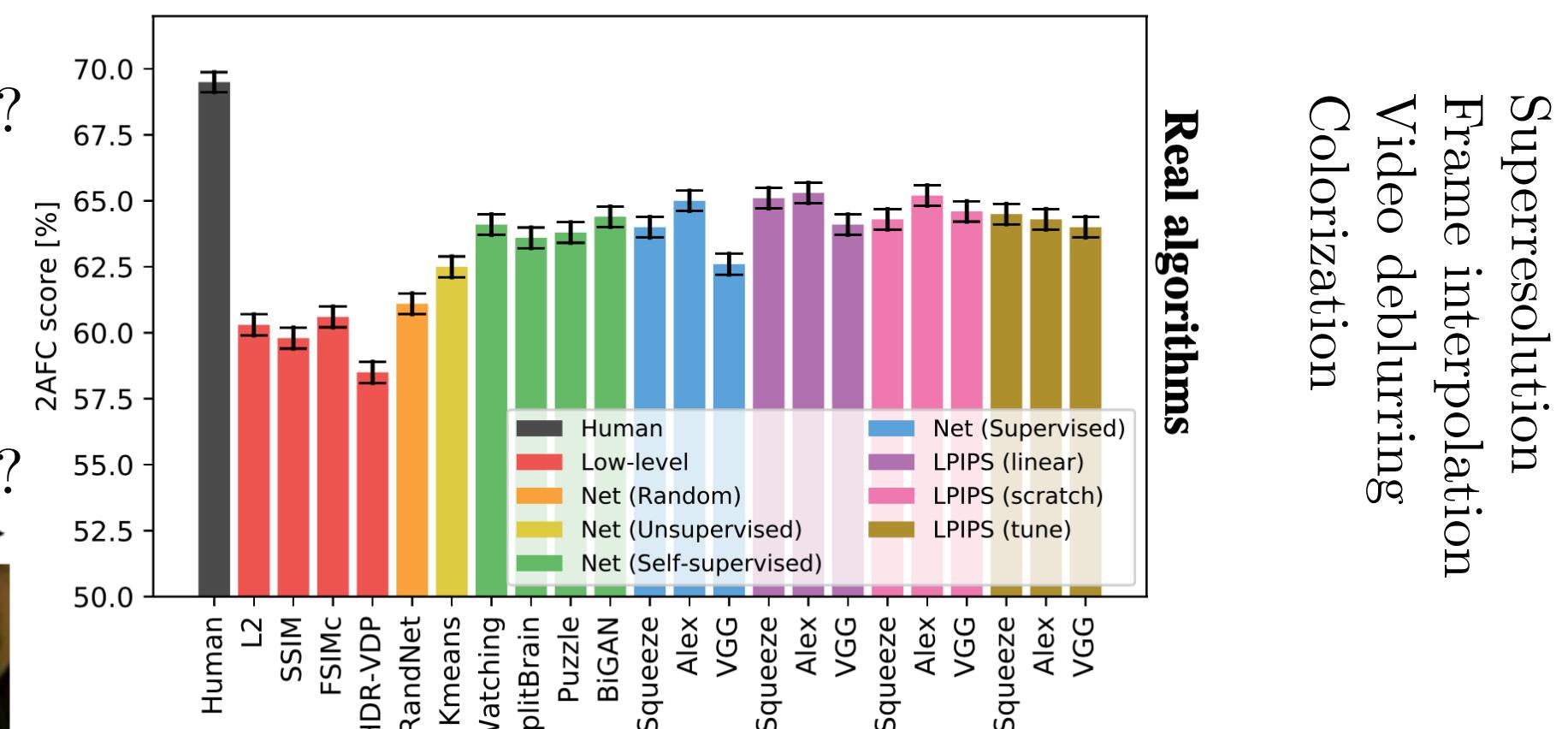


Learned Perceptual Image Patch Similarity (LPIPS) metric  
Distance between reference and distorted patches  $x, x_0$  with network  $\mathcal{F}$ :

$$d(x, x_0) = \sum_l \frac{1}{H_l W_l} \sum_{h,w} \|w_l \odot (\hat{y}_{hw}^l - \hat{y}_{0hw}^l)\|_2^2 \quad \hat{y}^\ell, \hat{y}_0^\ell \in \mathbb{R}^{H_\ell \times W_\ell \times C_\ell} \text{ for layer } \ell \quad w^\ell \in \mathbb{R}^C \rightarrow \text{scaling factor}$$

$$\|\hat{y}^\ell\| = \|\hat{y}_0^\ell\| = 1 \rightarrow \text{unit normalized in the channel dimension}$$

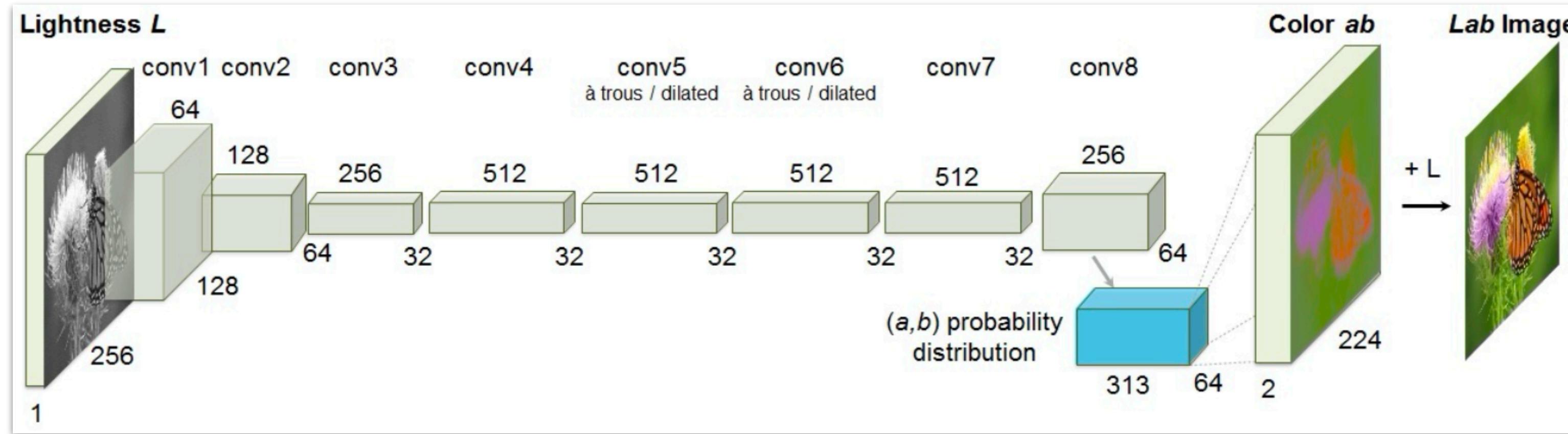
Zhang, Richard, et al. "The unreasonable effectiveness of deep features as a perceptual metric." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018.





Boulder

# Colorful Image Colorization



$X \in \mathbb{R}^{H \times W \times 1}$  → input lightness channel

$Y \in \mathbb{R}^{H \times W \times 2}$  → two associated color channels

$\hat{Y} = F(X) \in \mathbb{R}^{H \times W \times 2}$  → mapping (to be learned)

Distances in CIE Lab color space model perceptual distances

$$L_2(\hat{Y}, Y) = \frac{1}{2} \sum_{h,w} \|Y_{h,w} - \hat{Y}_{h,w}\|_2^2$$

Not robust to the inherent ambiguity and multimodal nature of the colorization problem.

For example, an apple is typically red, green, or yellow, but unlikely to be blue or orange.

Quantize the  $ab \in [-110, 110]^2$  output space into bins with grid size 10 and keep the  $Q = 313$  values which are in-gamut.

$\hat{Z} = G(X) \in [0, 1]^{H \times W \times Q}$  → probability distribution over colors

$Z = H_{gt}^{-1}(Y)$  → converts ground truth color  $Y$  to vector  $Z$ , using a soft-encoding scheme

Find the 5-nearest neighbors to  $Y_{h,w}$  in the output space and weight them proportionally to their distance from the ground truth using a Gaussian kernel with  $\sigma = 5$ .

$$L_{cl}(\hat{Z}, Z) = - \sum_{h,w} v(Z_{h,w}) \sum_q Z_{h,w,q} \log(\hat{Z}_{h,w,q})$$

$v(Z_{h,w})$  → weighting term used to rebalance the loss based on color-class rarity

$$\hat{Y} = H(\hat{Z}) \rightarrow \text{class probabilities to point estimates}$$

$$H(Z_{h,w}) = \mathbb{E}[f_T(Z_{h,w})]$$

$$f_T(z) = \frac{\exp(\log(z)/T)}{\sum_q \exp(\log(z_q)/T)}$$

$$T = 0.38$$



## Class Rebalancing

$p \in \Delta^Q$  → empirical probability of colors in the quantized  $ab$  space

$\tilde{p} \in \Delta^Q$  → smoothed empirical distribution

(smoothed version of  $p$  using Gaussian kernel  $G_\sigma$ )

$w \propto ((1 - \lambda)\tilde{p} + \lambda/Q)^{-1} \in \mathbb{R}^Q$  → weight factor

$1/Q$  → uniform distribution

$$\mathbb{E}[w] = \sum_q \tilde{p}_q w_q = 1 \rightarrow \text{normalization constraint}$$

$$q^* = \arg \max_q Z_{h,w,q} \rightarrow \text{closest } ab \text{ bin}$$

$$v(Z_{h,w,q}) = w_{q^*}$$

Colorization as a pretext task for representation learning

Acting as a cross-channel encoder!

PASCAL classification, detection, and segmentation



Boulder



# Questions?

[YouTube Playlist](#)

---