# Tutorial 12:
# Deep losses and priors

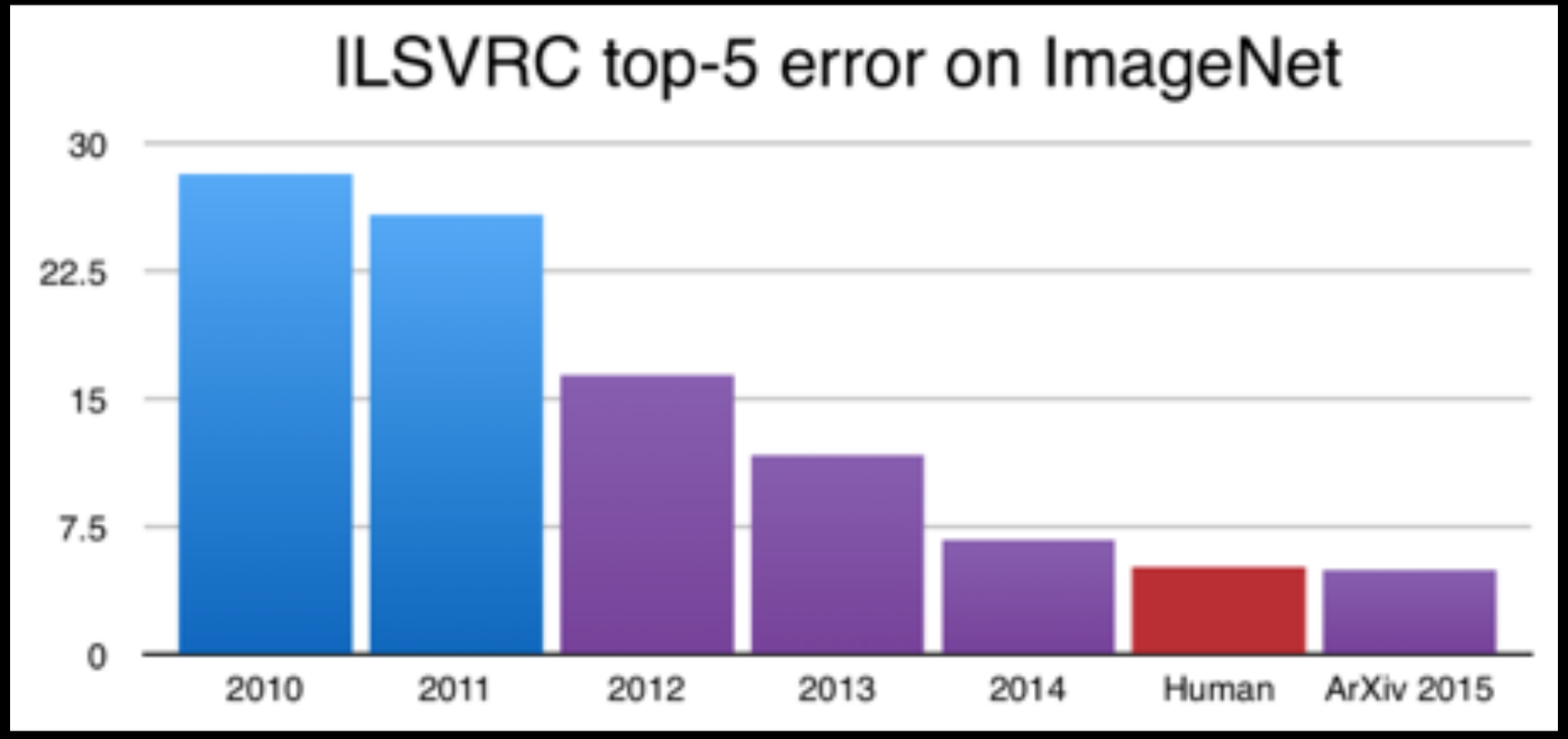
Digital Image Processing (236860)

# Agenda

- Common architectures: AlexNet, VGGNet
- Perceptual losses
- Deep image prior

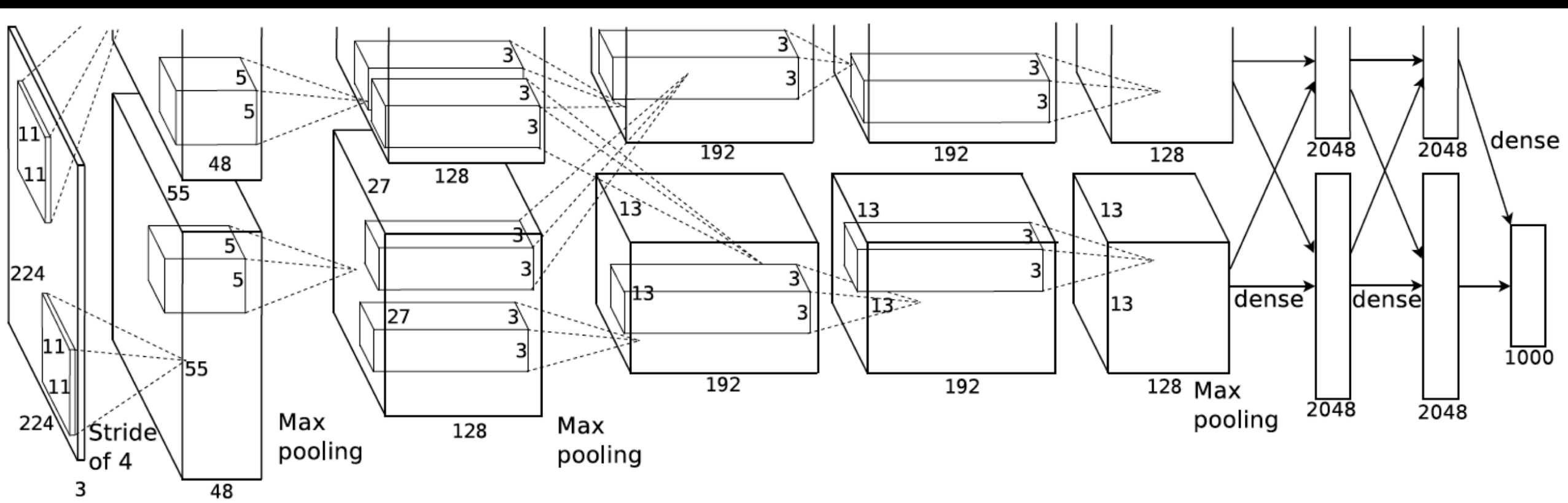
# Advances in image classification

- ImageNet: a dataset of 1M images with 1000 categories.



# AlexNet

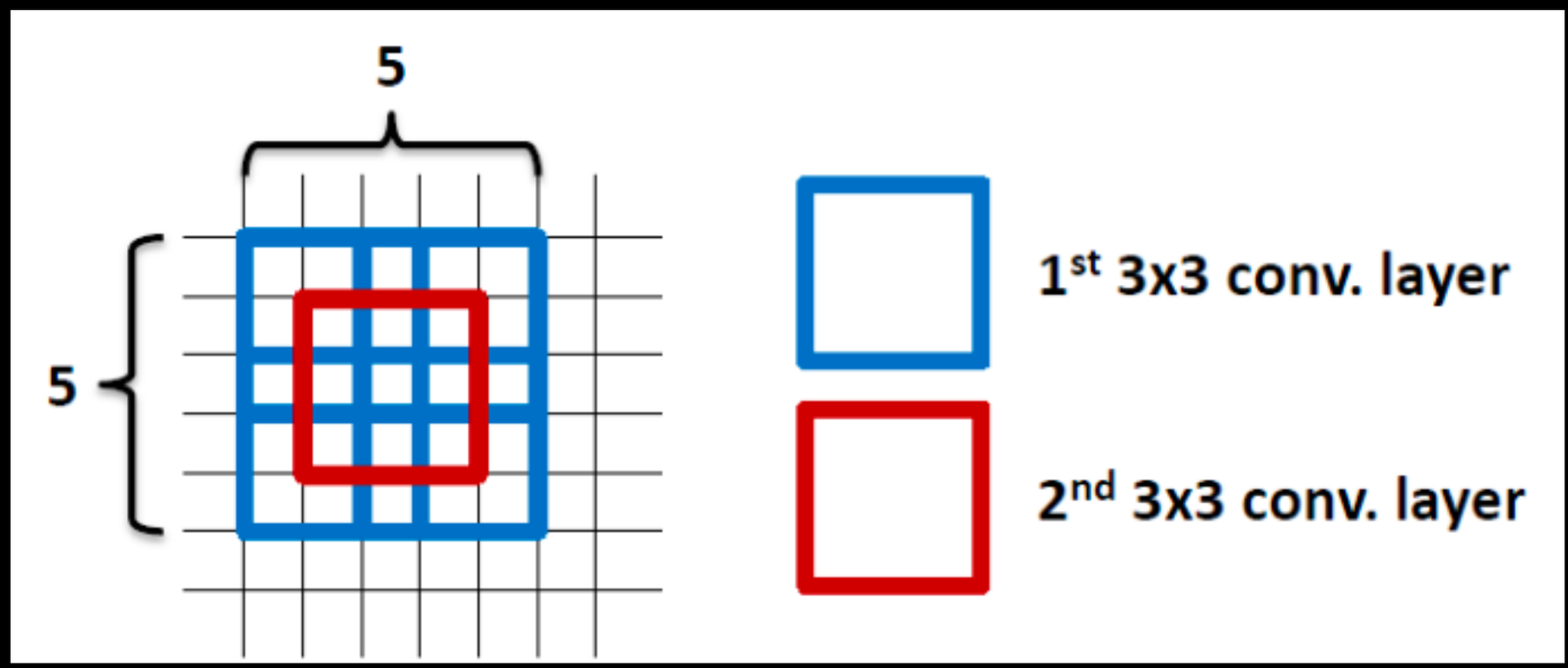
- Primary result: the depth of the model is essential to its high performance.
- The utilisation of GPUs made this model feasible.




Alex Krizhevsky

# VGGNet

- Only uses 3x3 filters.
- 1 layer of 5x5 filter => 5x5=25 parameters.
- 2 layers of 3x3 filter => 3x3x2=18 parameters.
- Number of parameters is reduced by 28%!




https://medium.com/coinmonks/paper-review-of-vggnet-1st-runner-up-of-ilsvlc-2014-image-classification-d02355543a11

# VGGNet


*Image: VGG group, Oxford, 2014; © Zieler & Fergus, 2013.*

# VGGNet

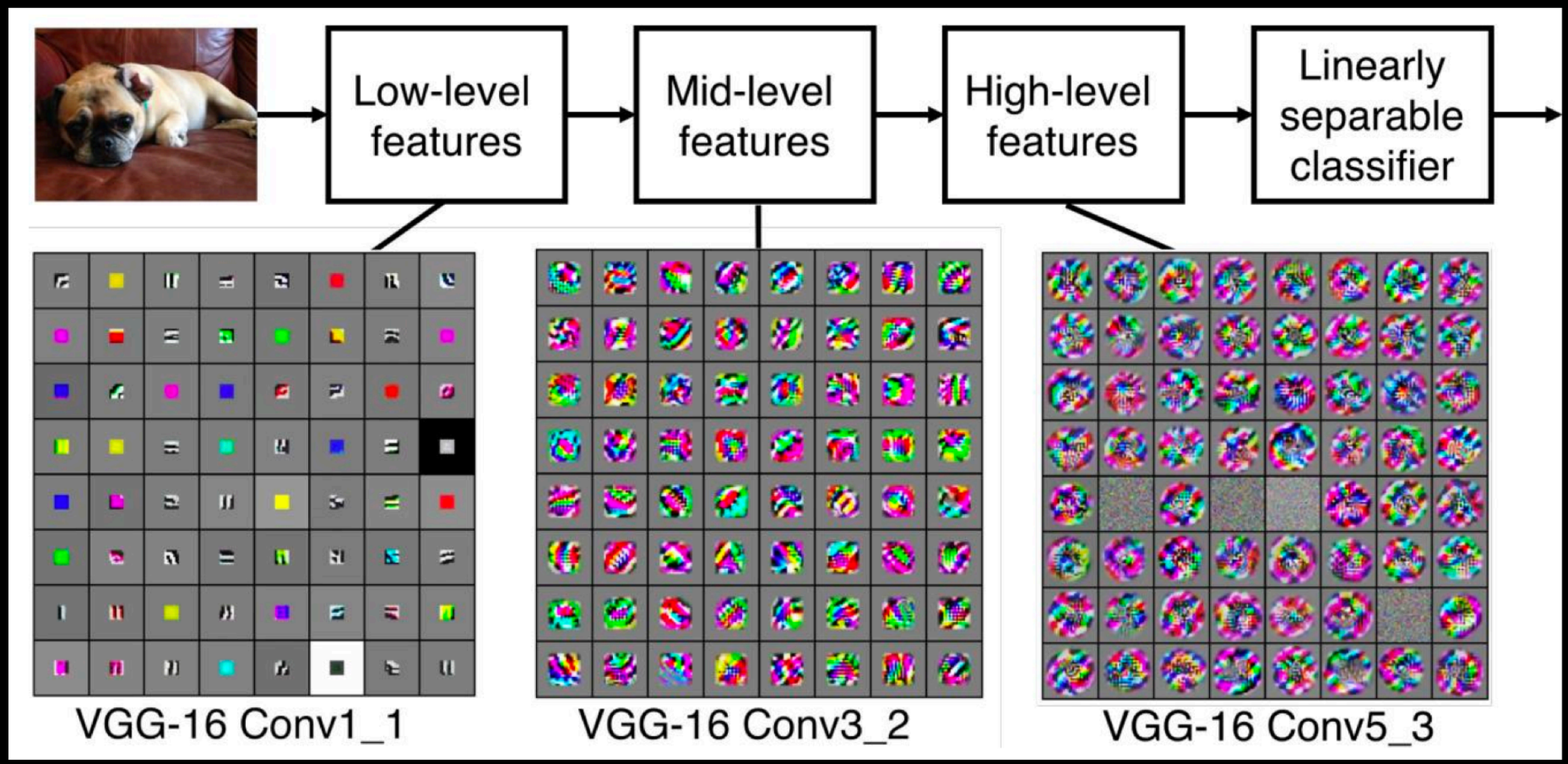


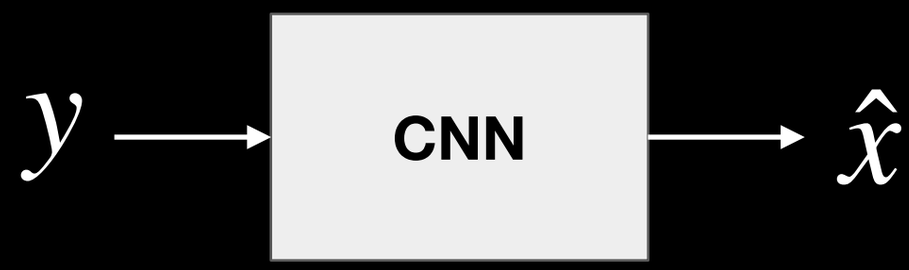

*Image: VGG group, Oxford, 2014; © Zieler & Fergus, 2013.*

# Perceptual losses

- Instead of minimising per pixel discrepancy $\| \hat{x} - x \|$, minimise a loss that measures perception.

- Assume we have a pre-trained VGG classification network $\phi$.

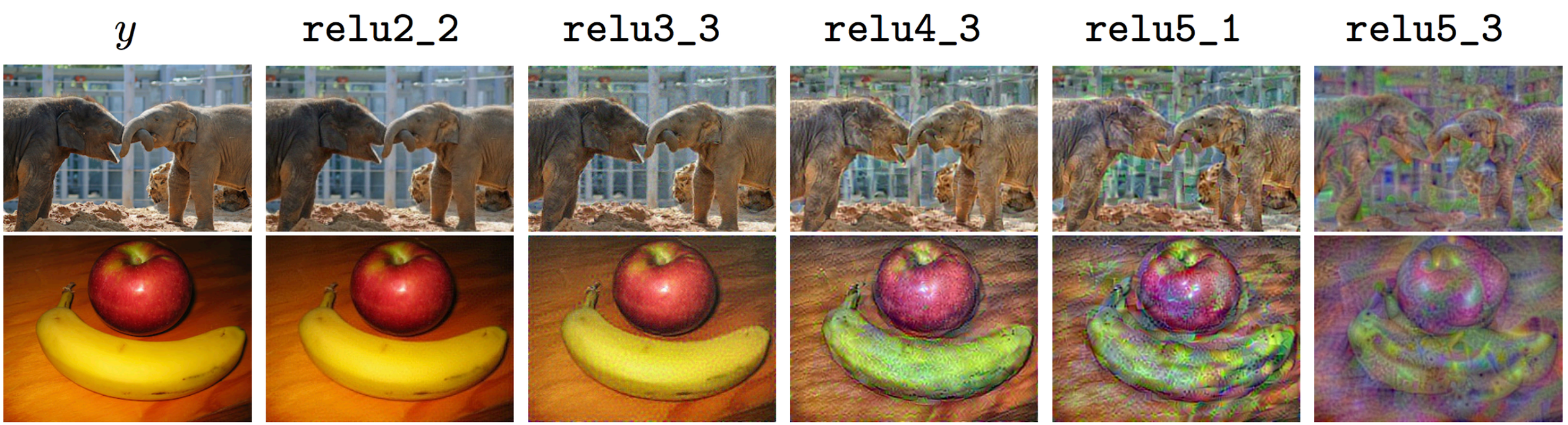
**GT:** $x$




*Johnson et al., ECCV 2016*

# Feature construction loss

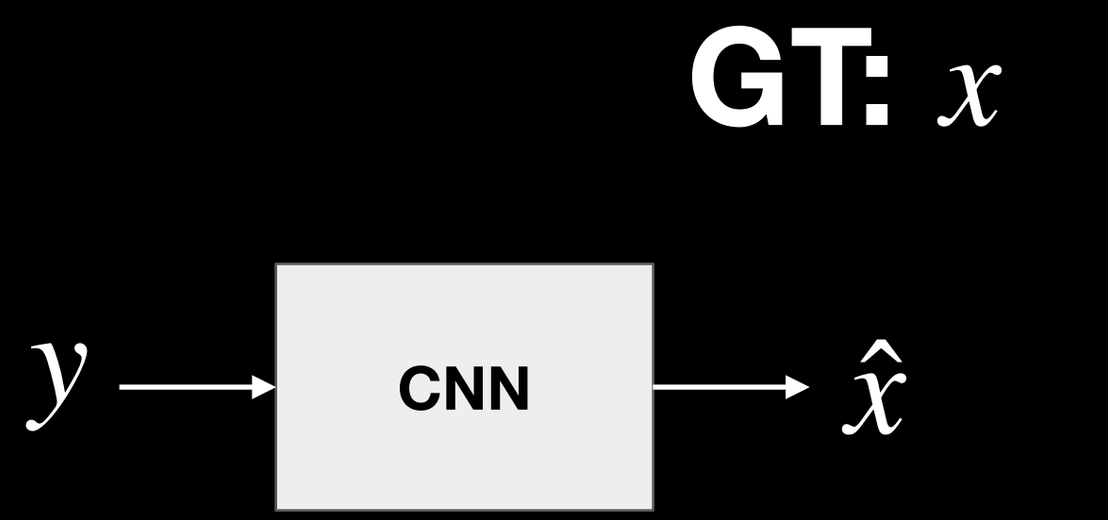
- VGG features contain semantic information.




*Johnson et al., ECCV 2016*

# Feature construction loss

- Minimise $\left\| \phi_j\left(\hat{x}\right) - \phi_j(x) \right\|$, where $\phi_j$ are the activations of the $j$-th layer of the network.




*Johnson et al., ECCV 2016*

# Feature reconstruction loss for super resolution

| Ground Truth | Bicubic | Ours ($\ell_{pixel}$) | SRCNN [11] | Ours ($\ell_{feat}$) |
|---|---|---|---|---|
| This image | 31.78 / 0.8577 | 31.47 / 0.8573 | 32.99 / 0.8784 | 29.24 / 0.7841 |
| Set5 mean | 28.43 / 0.8114 | 28.40 / 0.8205 | 30.48 / 0.8628 | 27.09 / 0.7680 |

**PSNR/SSIM are not good perceptual measures!**

*Johnson et al., ECCV 2016*

# Neural style transfer

Content Image

+

Style Image

=

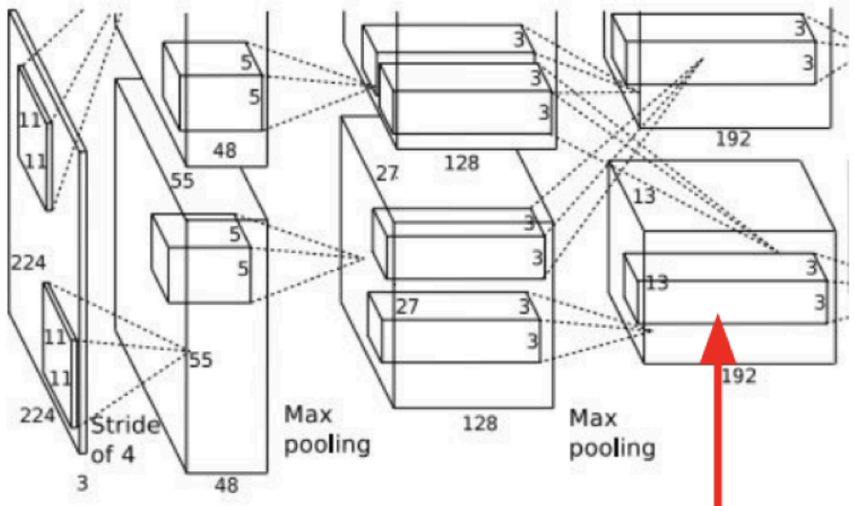
Style Transfer!

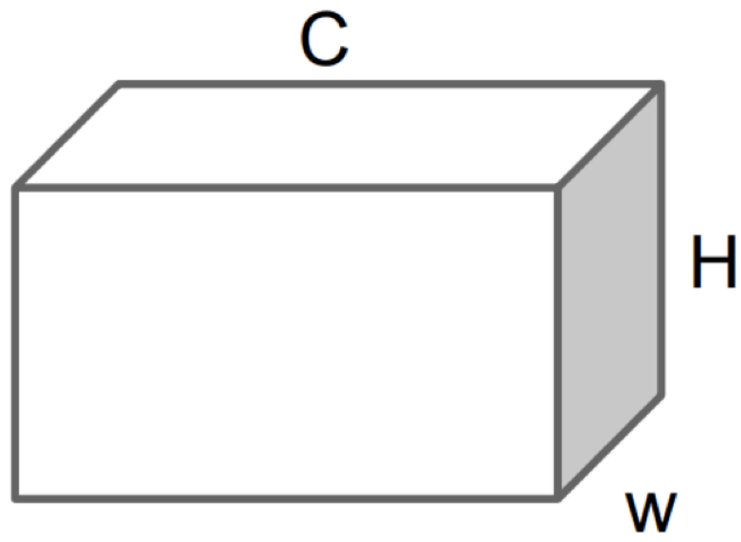
feature reconstruction loss

How can we measure “style”?

# Style reconstruction loss

This image is in the public domain.

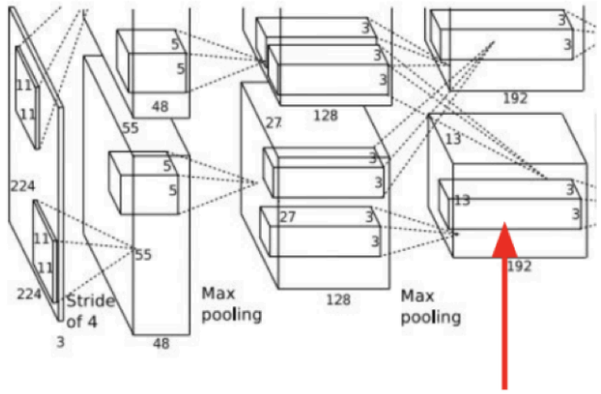


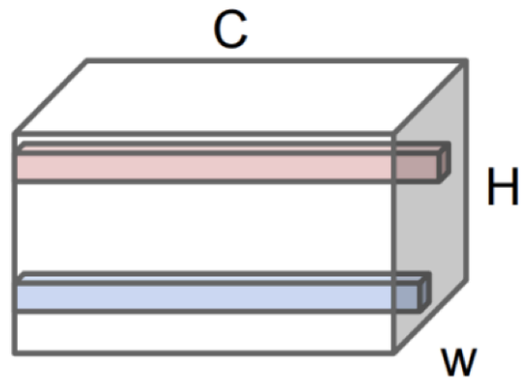



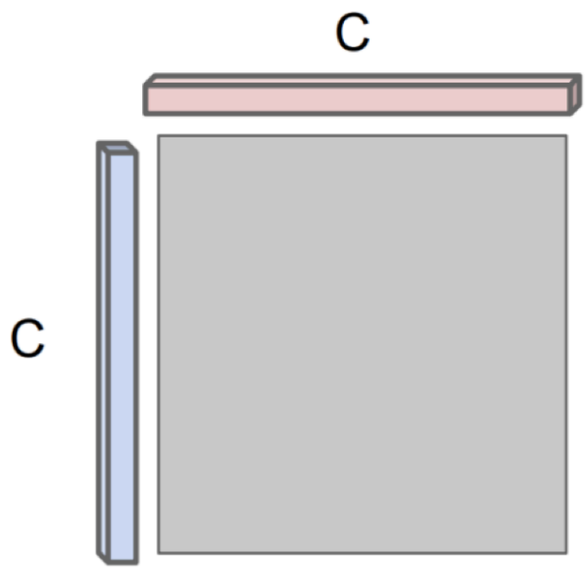

Each layer of CNN gives C x H x W tensor of features; H x W grid of C-dimensional vectors

# Style reconstruction loss

This image is in the public domain.



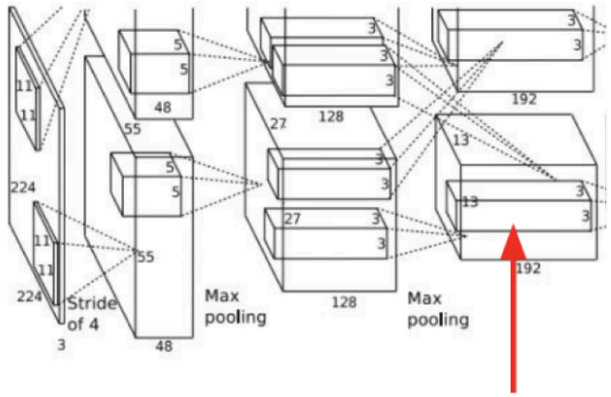



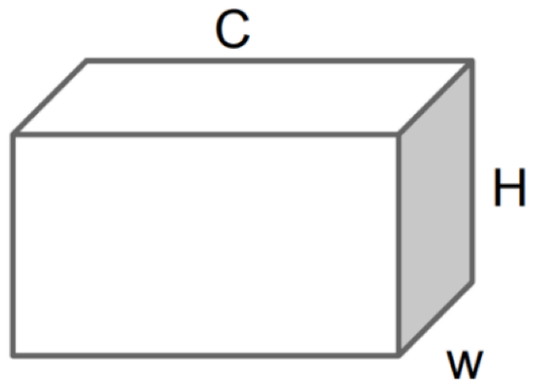

Each layer of CNN gives C x H x W tensor of features; H x W grid of C-dimensional vectors

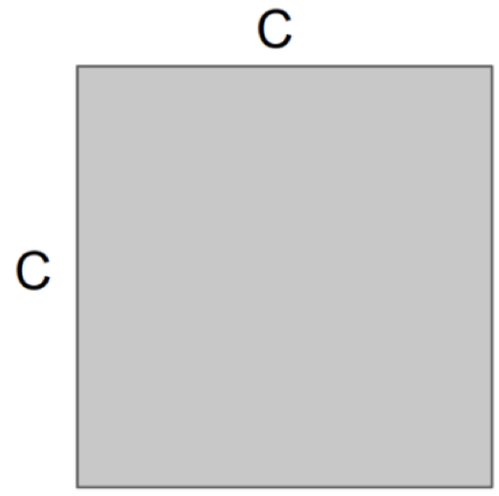
Outer product of two C-dimensional vectors gives C x C matrix measuring co-occurrence

# Style reconstruction loss

This image is in the public domain.




Each layer of CNN gives C x H x W tensor of features; H x W grid of C-dimensional vectors

Outer product of two C-dimensional vectors gives C x C matrix measuring co-occurrence

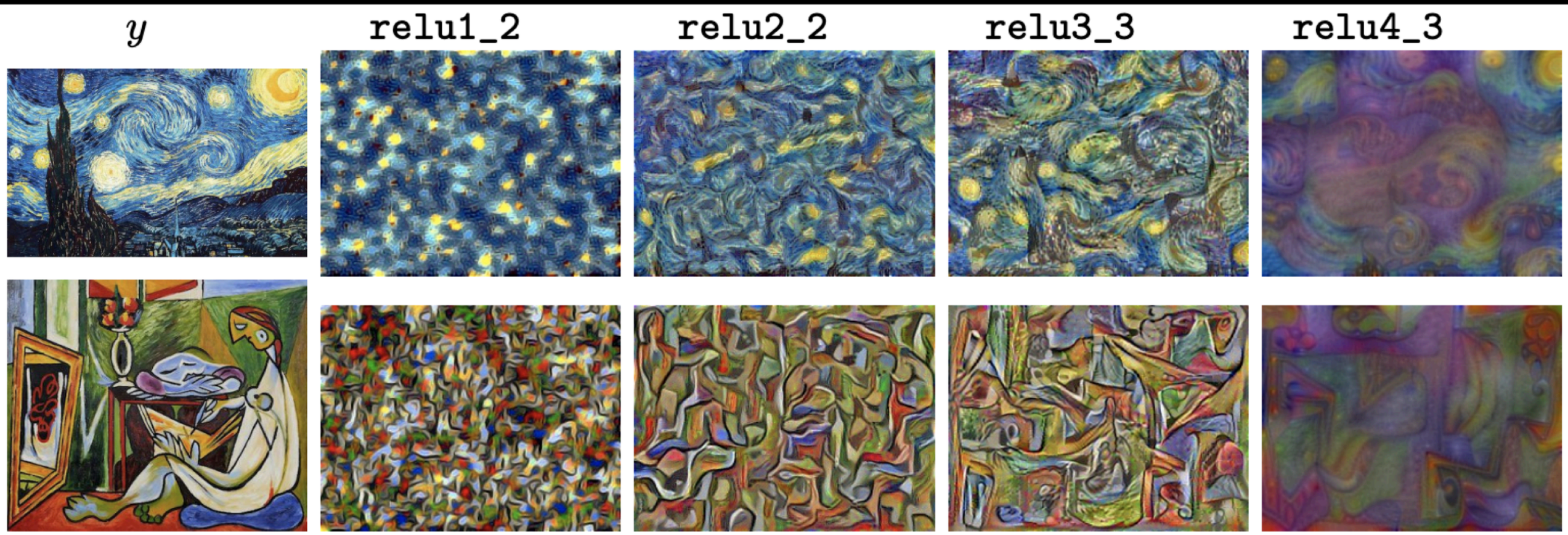
Average over all HW pairs of vectors, giving **Gram matrix** of shape C x C

Gram Matrix

**The gram matrix doesn't carry any spatial information!**

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

# Style reconstruction loss

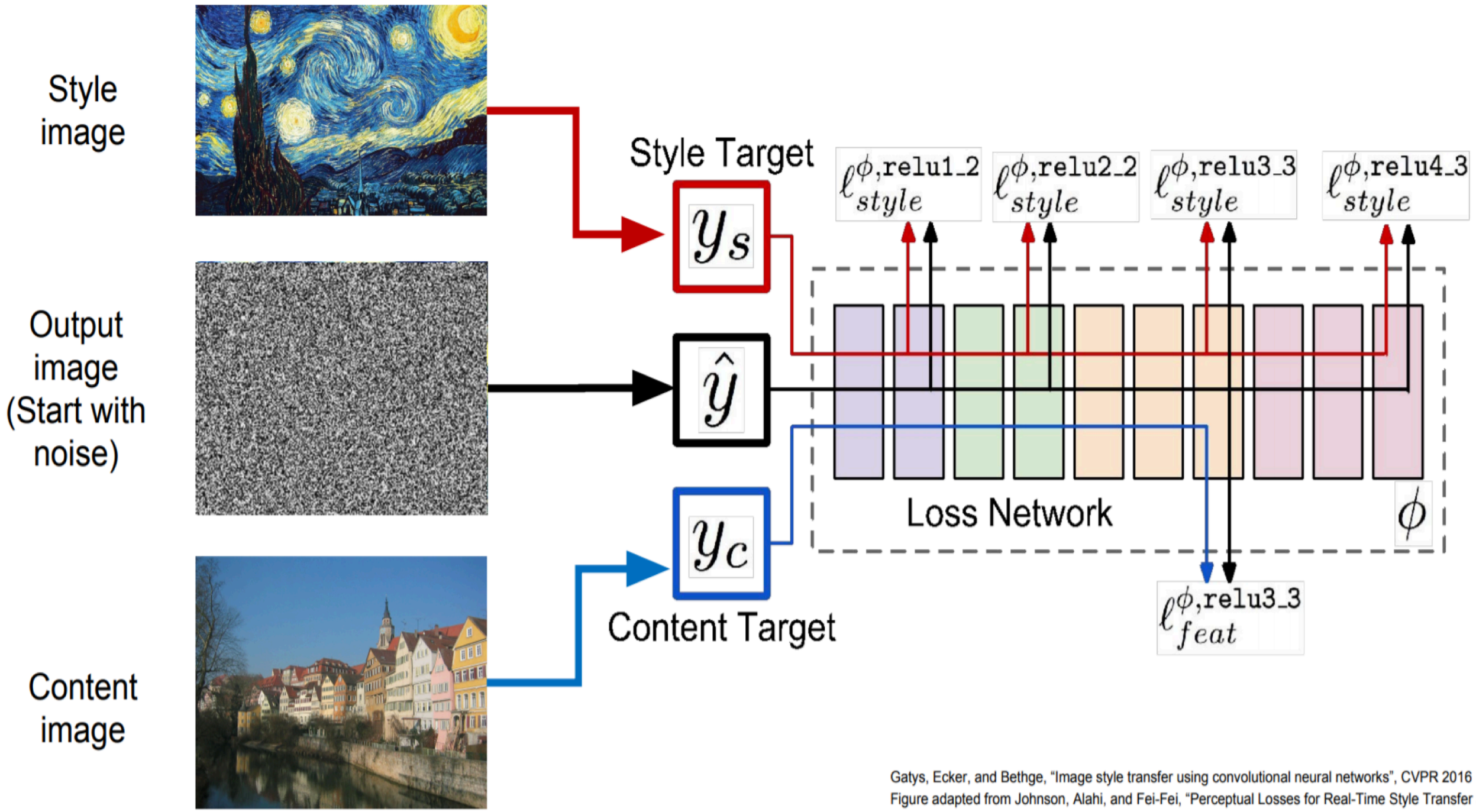



*Johnson et al., ECCV 2016*

# Summary: conceptual losses

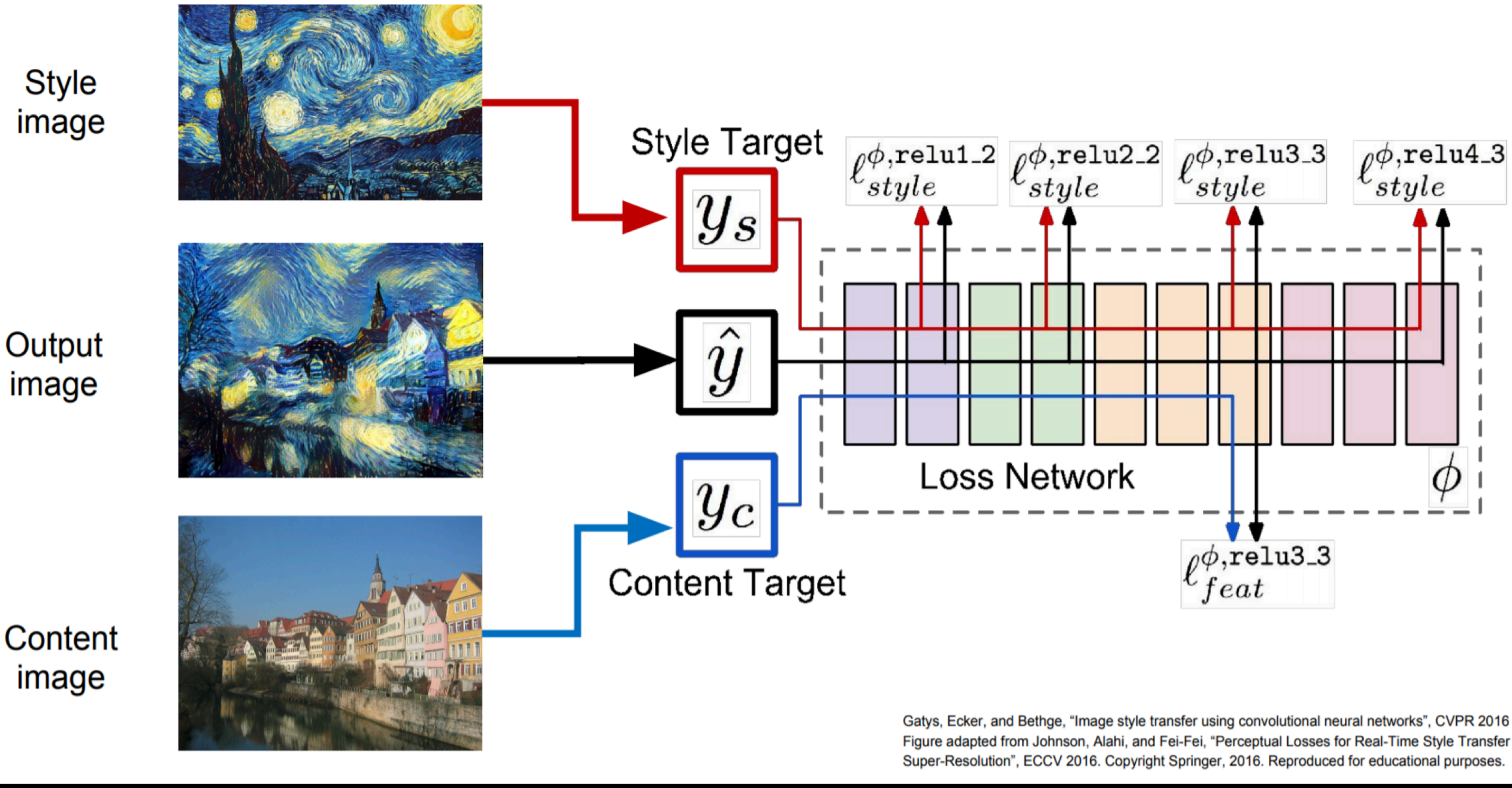
- Feature reconstruction loss: $\left\| \phi_j\left(\hat{x}\right) - \phi_j\left(x\right) \right\|$
- Style reconstruction loss: $\left\| G_j^{\phi}\left(\hat{x}\right) - G_j^{\phi}\left(x\right) \right\|$

# Neural style transfer



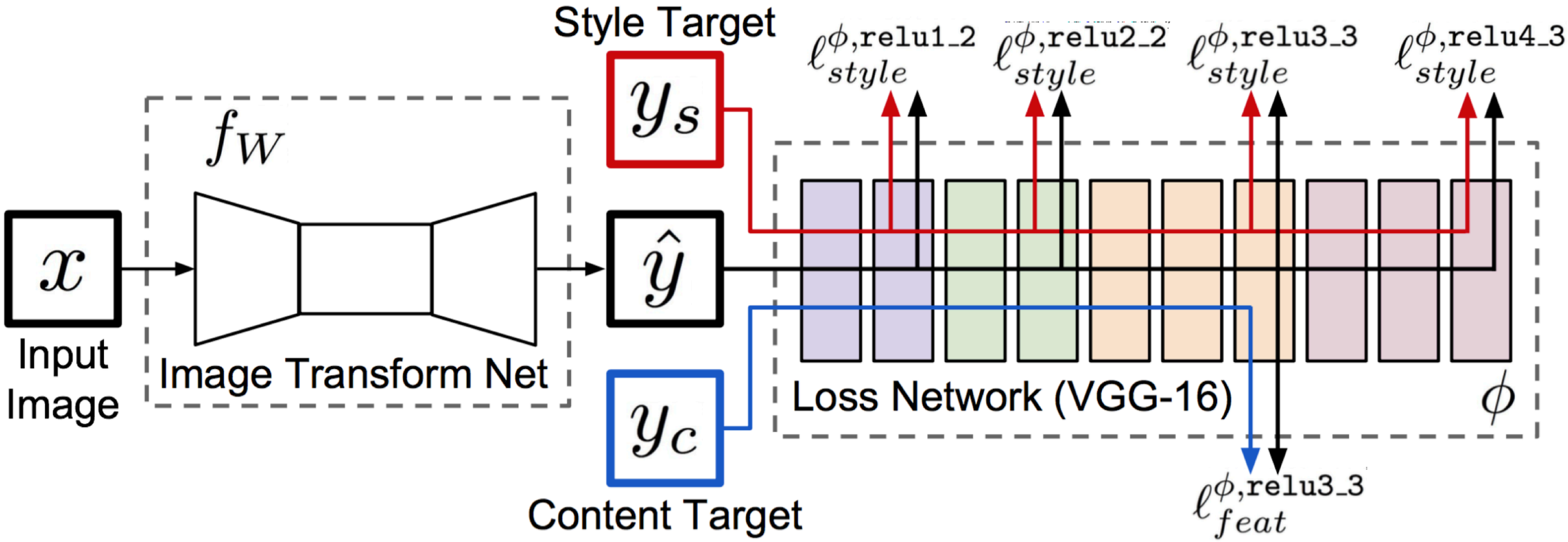

Gatys, Ecker, and Bethge, "Image style transfer using convolutional neural networks", CVPR 2016
Figure adapted from Johnson, Alahi, and Fei-Fei, "Perceptual Losses for Real-Time Style Transfer and Super-Resolution", ECCV 2016. Copyright Springer, 2016. Reproduced for educational purposes.

# Neural style transfer




Gatys, Ecker, and Bethge, "Image style transfer using convolutional neural networks", CVPR 2016
Figure adapted from Johnson, Alahi, and Fei-Fei, "Perceptual Losses for Real-Time Style Transfer and Super-Resolution", ECCV 2016. Copyright Springer, 2016. Reproduced for educational purposes.

# Neural style transfer

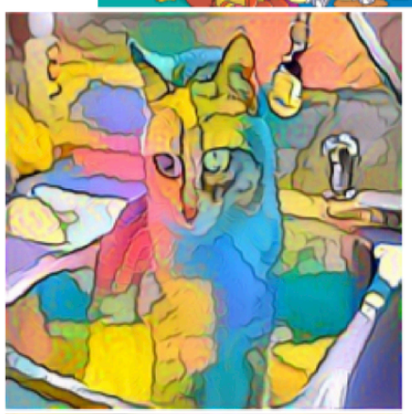


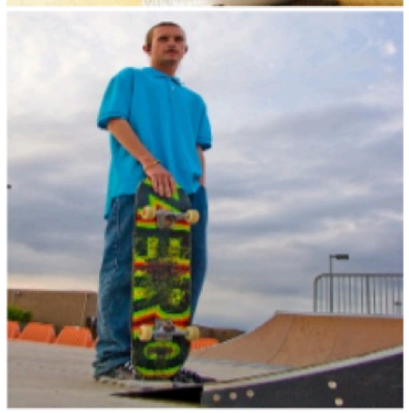

$$\hat{y} = \arg\min_y \lambda_c \ell^{\phi,j}_{feat}(y, y_c) + \lambda_s \ell^{\phi,J}_{style}(y, y_s) + \lambda_{TV} \ell_{TV}(y)$$

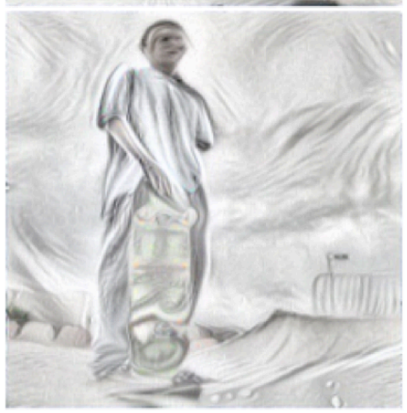
# Neural style transfer

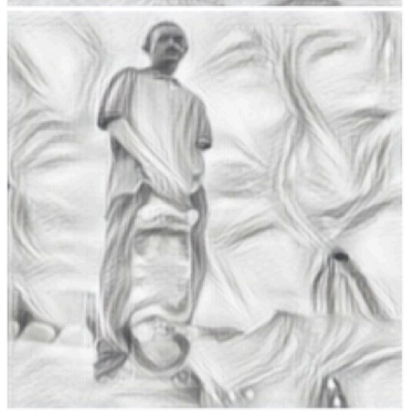
**Style**
*Sketch*

**Style**
*The Simpsons*









**Content** [10] Ours **Content** [10] Ours

*Johnson et al., ECCV 2016*

# Neural style transfer

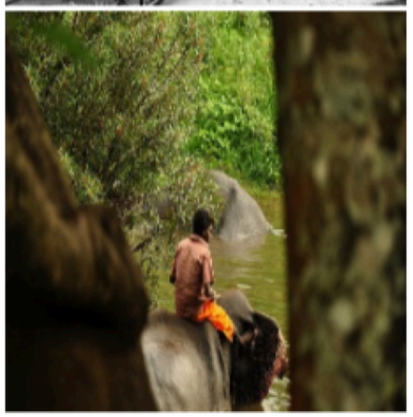
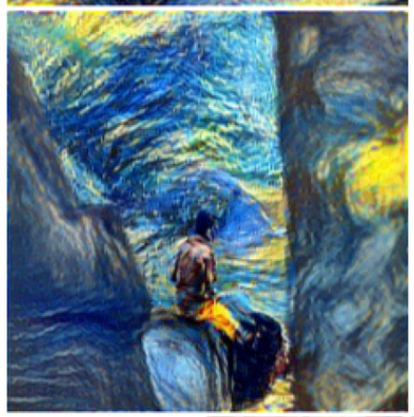
**Style**
*The Starry Night*,
Vincent van Gogh,
1889

**Style**
*The Muse*,
Pablo Picasso,
1935

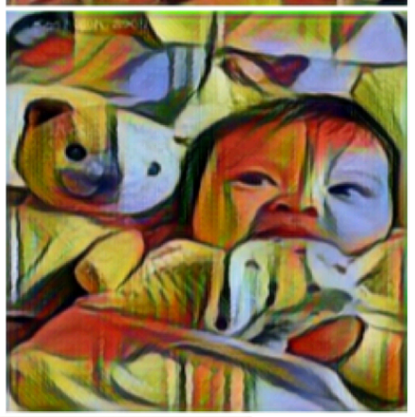








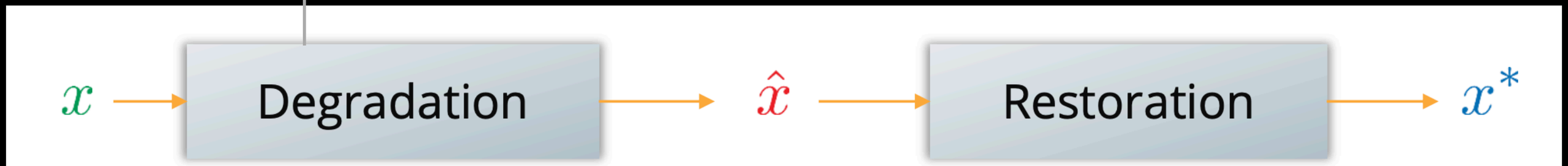


*Johnson et al., ECCV 2016*

# Image priors

denoising,
inpainting, blur etc.

- $x$ – Clean image
- $\hat{x}$ – Corrupted image (observed)
- $x^*$ – Restored image




**MAP:** $$x^* = \arg\max_x p(x|\hat{x})$$

$$p(x|\hat{x}) = \frac{p(\hat{x}|x)p(x)}{p(\hat{x})} \propto p(\hat{x}|x)p(x)$$

**Likelihood** **Prior**

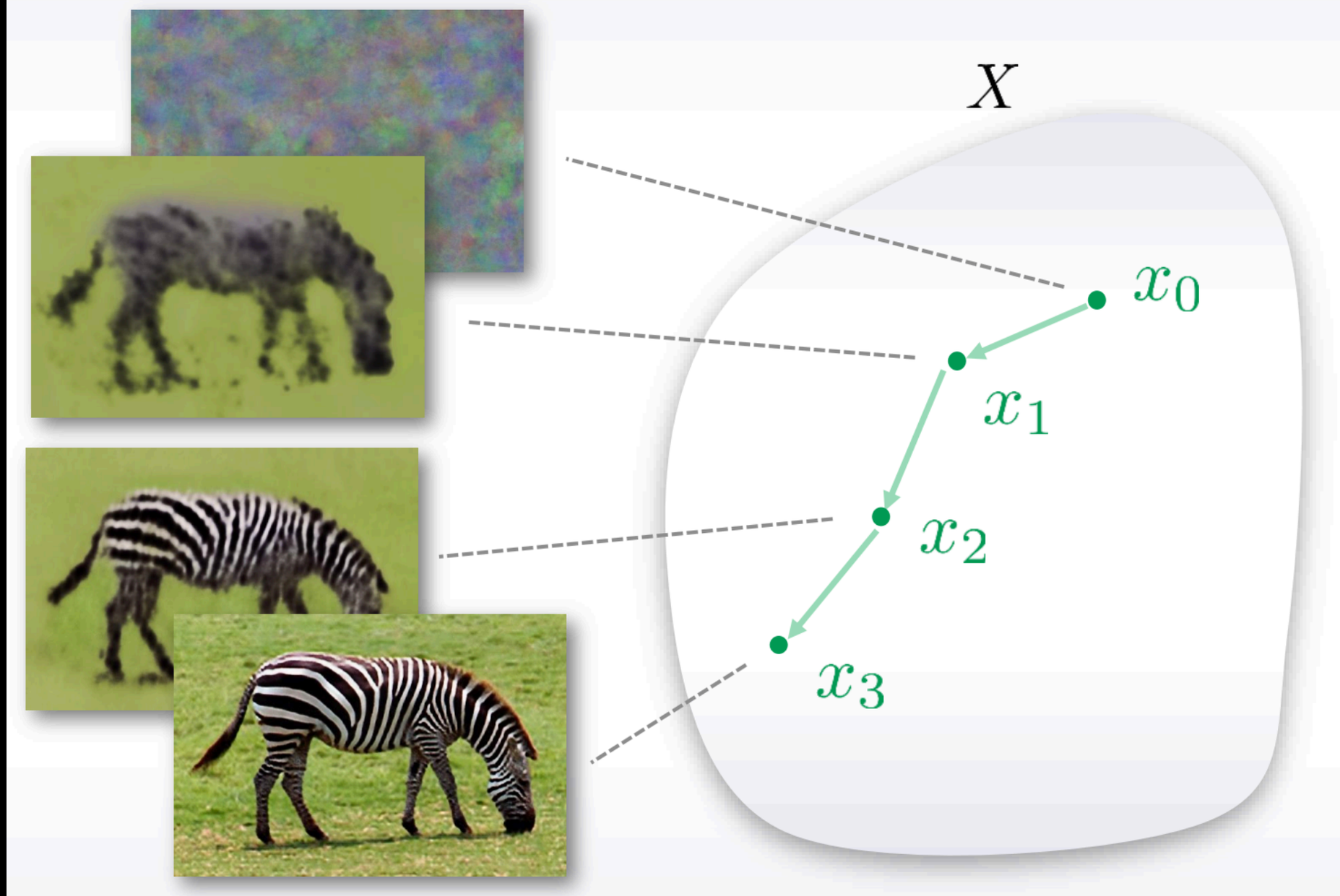
# Image priors

- $x$ – Clean image
- $\hat{x}$ – Corrupted image (observed)
- $x^*$ – Restored image

$$
\begin{aligned}
x^* &= \arg\max_x p(\hat{x}|x)p(x) \\
&= \arg\min_x -\log p(\hat{x}|x) - \log p(x) \\
&= \arg\min_x E(x;\hat{x}) + R(x)
\end{aligned}
$$

We've seen several instances of how to model priors.

# Image priors




**Regular:** $\arg\min_x E(x; \hat{x}) + R(x)$

# Parametrised image priors

- $x$ – Clean image
- $\hat{x}$ – Corrupted image (observed)
- $x^*$ – Restored image

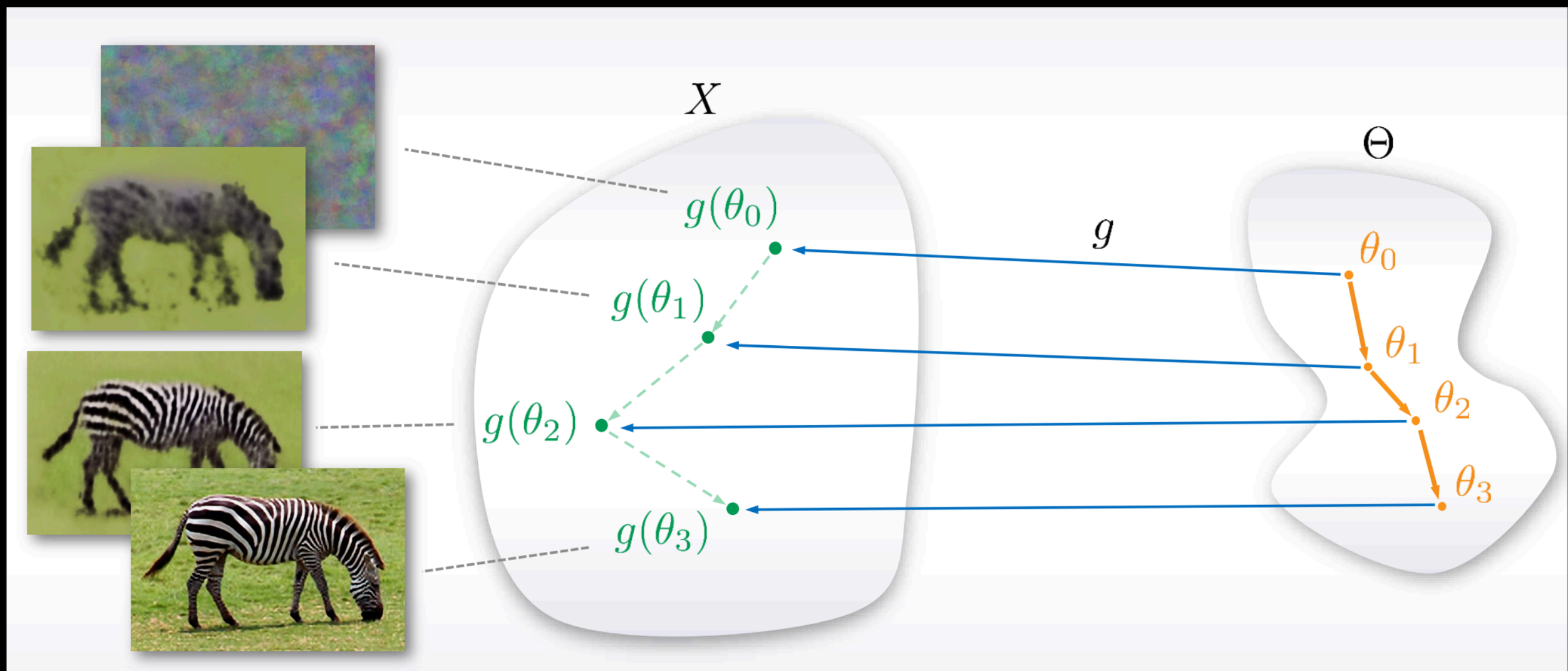
Search in the image space

**Regular:** $\arg\min_{x} E(x; \hat{x}) + R(x)$

**Parametrized:** $\arg\min_{\theta} E(g(\theta); \hat{x}) + R(g(\theta))$

Search in some other space

# Parameterised image priors




**Parametrized:** $\arg\min_{\theta} E(g(\theta); \hat{x}) + R(g(\theta))$

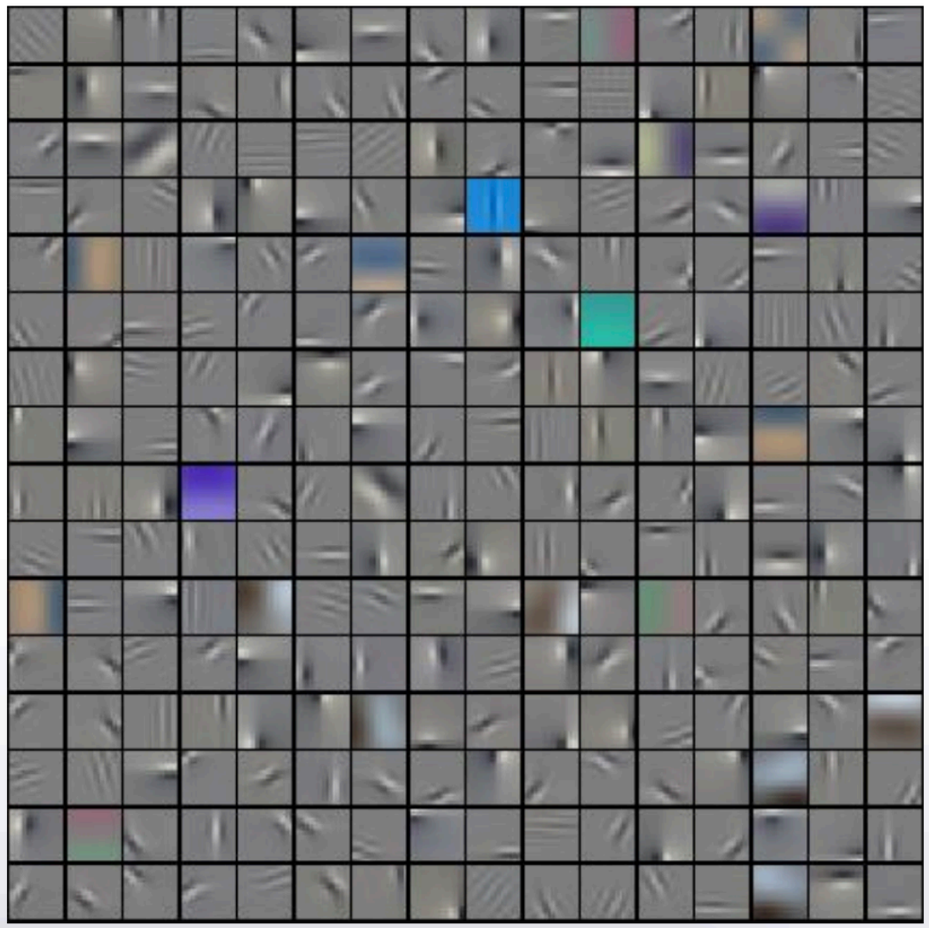
# Parameterised image priors

**Parametrized:** $\arg\min_{\theta} E(g(\theta); \hat{x}) + R(g(\theta))$

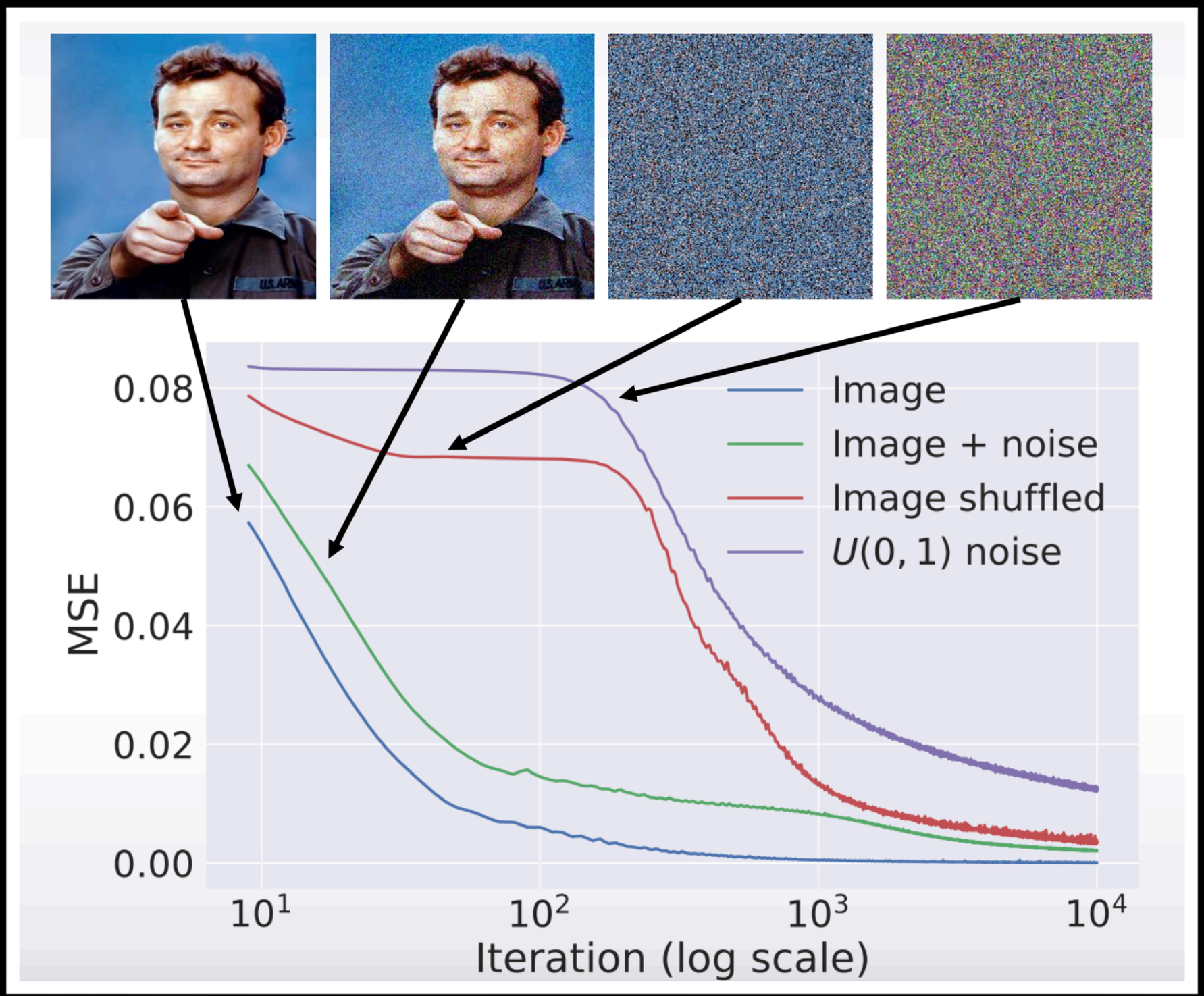
**Coefficients**

$$g(\theta) = D\theta$$

**Dictionary**



# Deep image prior



# Deep image prior

**Parametrized:** $\arg\min_{\theta} E(g(\theta); \hat{x}) + R(g(\theta))$

**Fixed input**

$$g(\theta) \equiv f_\theta(z)$$

**Convolutional network with parameters** $\theta$

**Optimize only data term:** $\arg\min_{\theta} E(f_\theta(z); \hat{x})$

# Deep image prior

- $\hat{x}$ – Corrupted image (observed)

1. **Initialize** $z$
   - For example fill it with uniform noise U(-1, 1)

2. **Solve** $$\arg\min_{\theta} E(f_\theta(z); \hat{x}))$$
   - With your favorite gradient-based method

$$\theta^{k+1} = \theta^k - \alpha \frac{\partial E(f_\theta(z); \hat{x})}{\partial \theta}$$

3. **Get the solution**

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

# Deep image prior

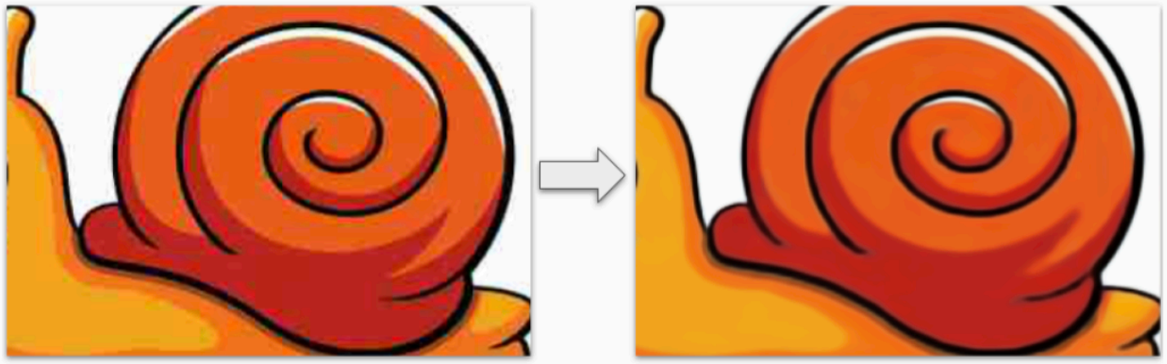
- $x$ – Clean image
- $\hat{x}$ – Corrupted image (observed)
- $m$ – Binary mask

**Objective:** $\arg\min_{\theta} E(f_\theta(z); \hat{x}))$

- **Denoising:** $E(x; \hat{x}) = \|x - \hat{x}\|^2$ **Needs early stopping!**
- **Inpainting:** $E(x; \hat{x}) = \|(x - \hat{x}) \cdot m\|^2$
- **Super-Res.:** $E(x; \hat{x}) = \|d(x) - \hat{x}\|^2$

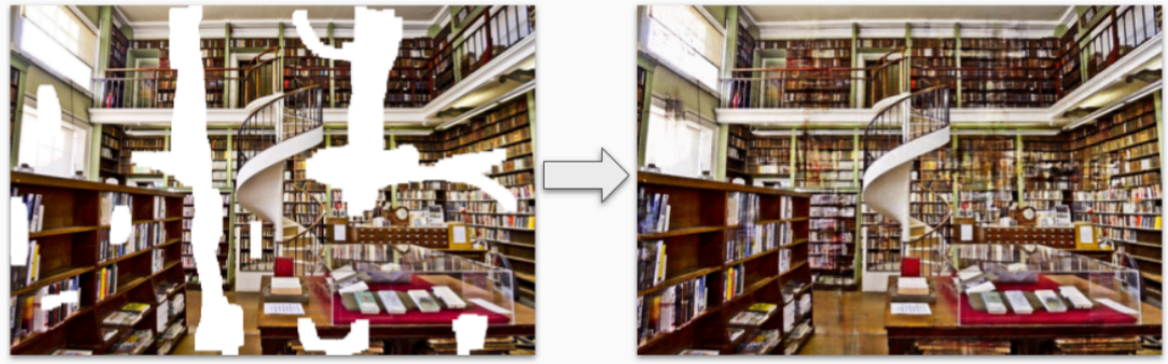
# Deep image prior

**JPEG Artifacts removal**



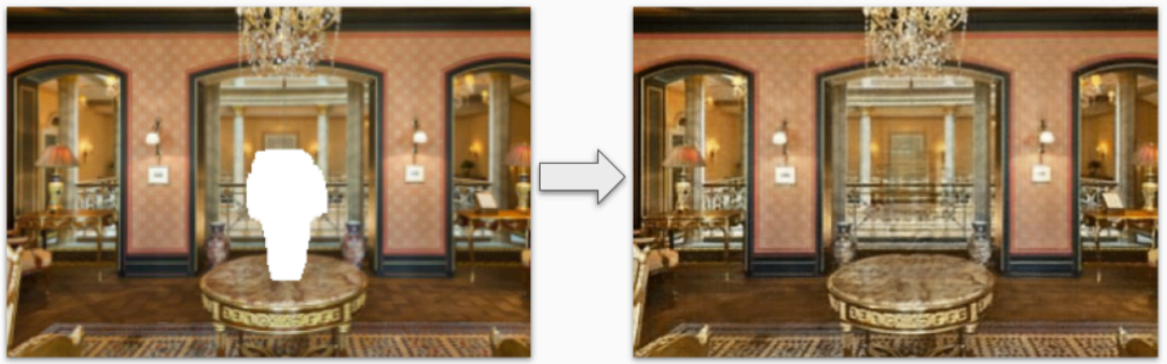
Corrupted | Deep image prior

**Inpainting**



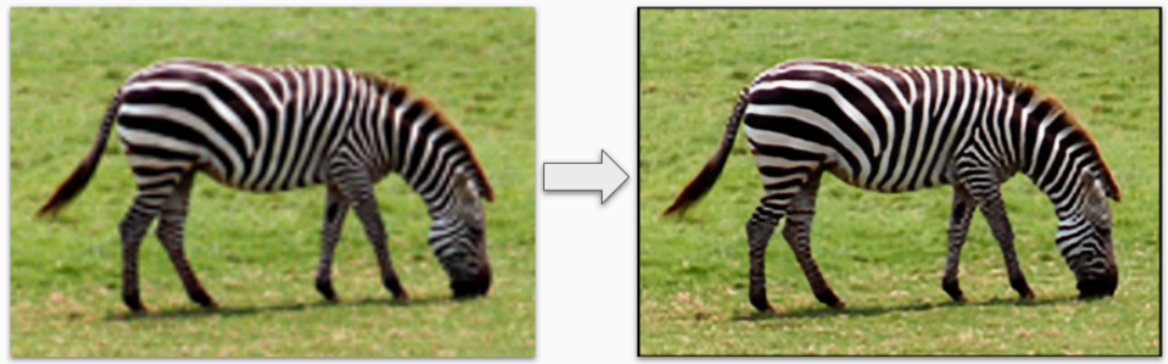
Corrupted | Deep image prior

**Inpainting**



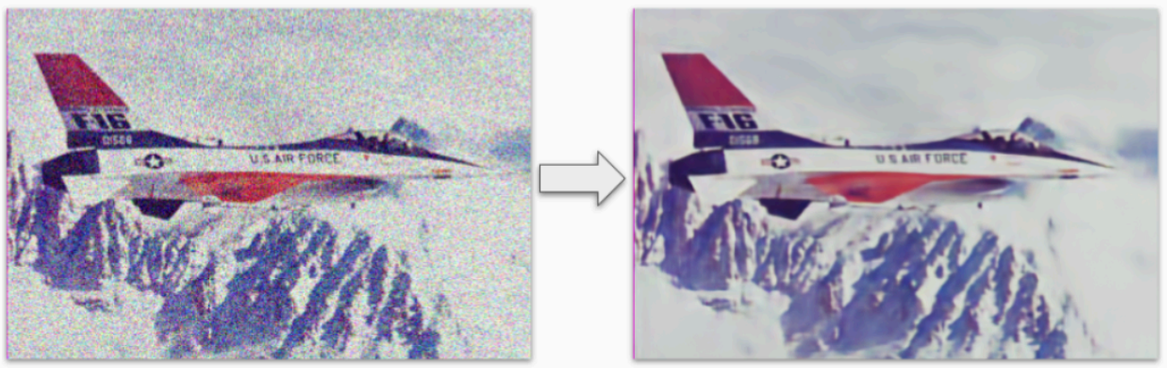
Corrupted | Deep image prior

**Super-resolution**



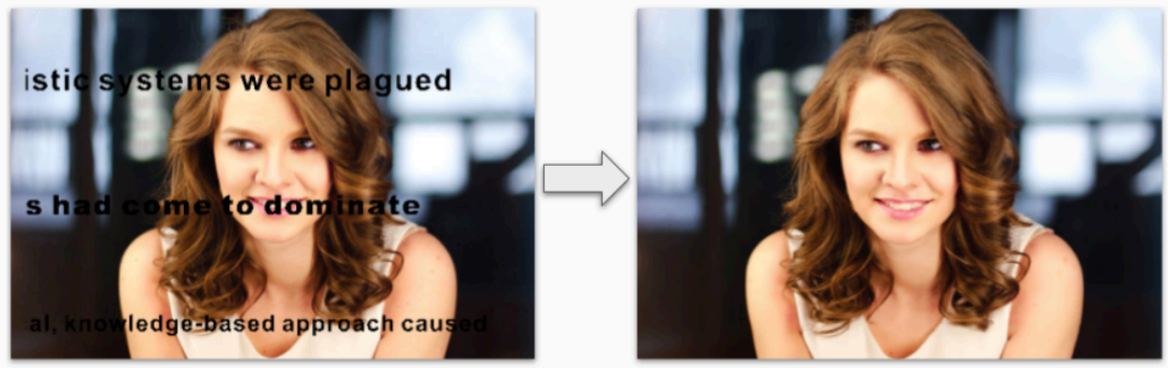
Corrupted | Deep image prior

**Denoising**




Corrupted | Deep image prior

**Inpainting**




Corrupted | Deep image prior