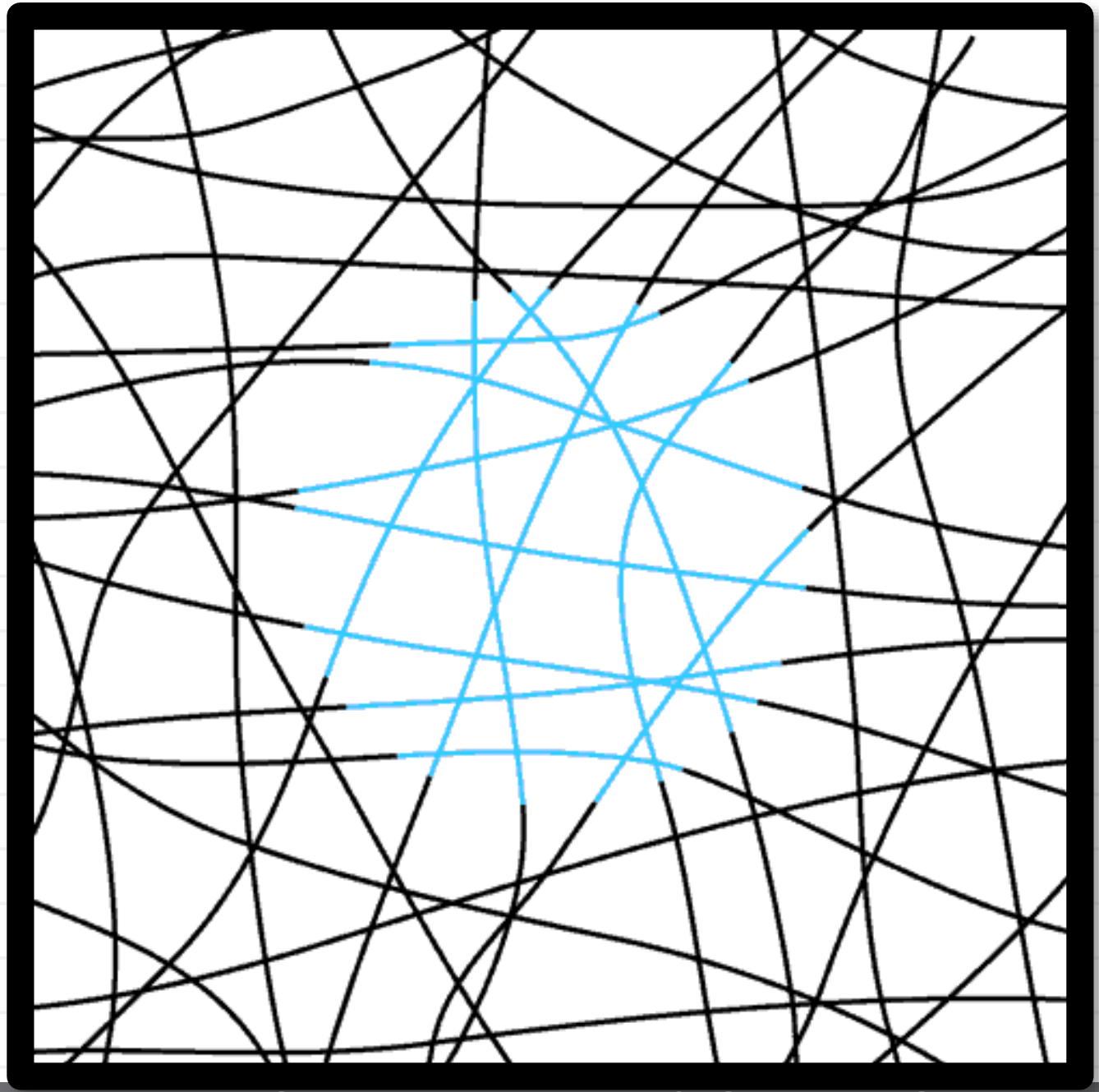
Lecture 6: Generative ConvNets

Why generate images using ConvNets?

- Answer 1: to get "good" images (computer graphics, advanced image processing and image-based rendering)
- Answer 2: to get better understanding of ConvNets
- Answer 3: to get training data for recognition
- Answer 4: truly robust vision must be (partially) generative

Generative model in biological vision model



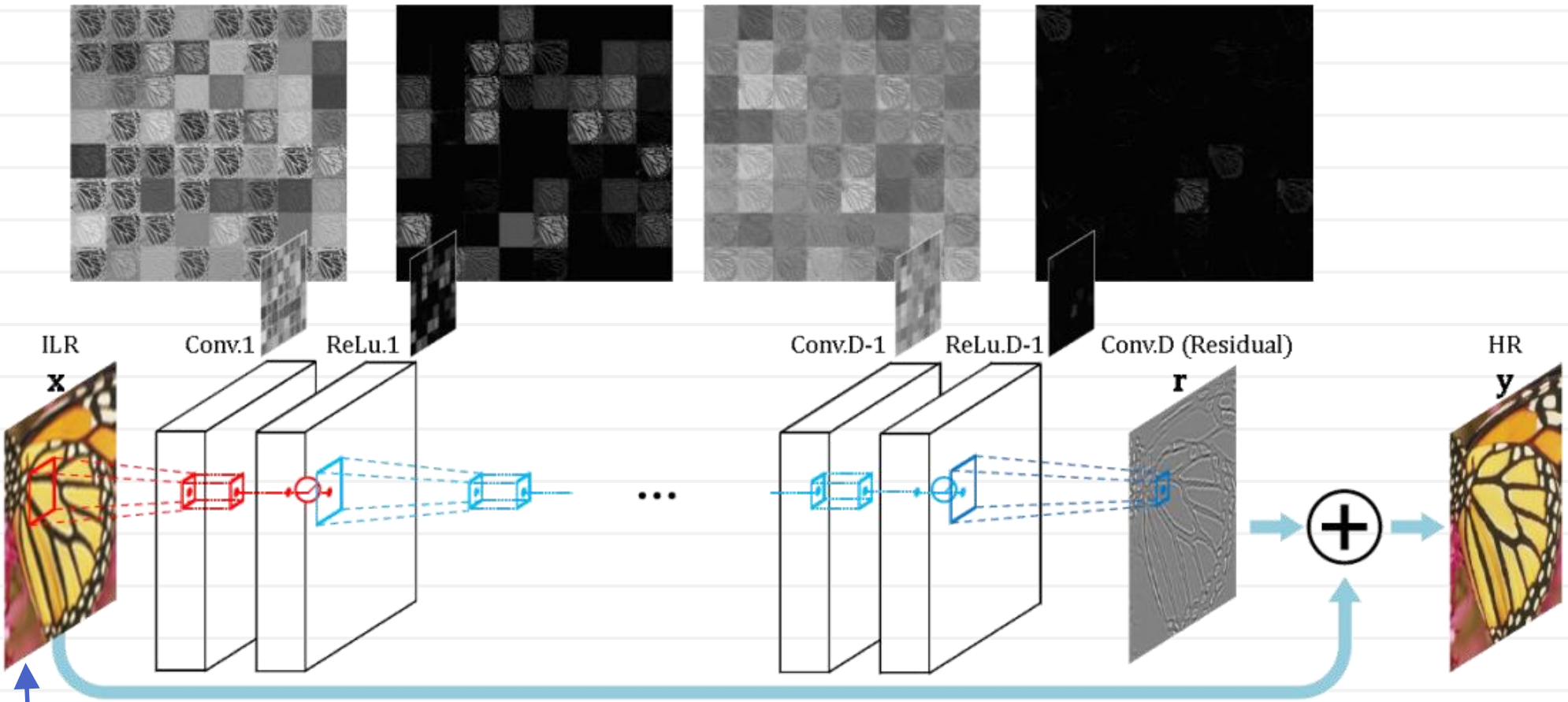
Single image superresolution

x_i

y_i

- ConvNet architecture that maps *x* to *y*.
- We have access to very good synthetic data

Image superresolution with ConvNets

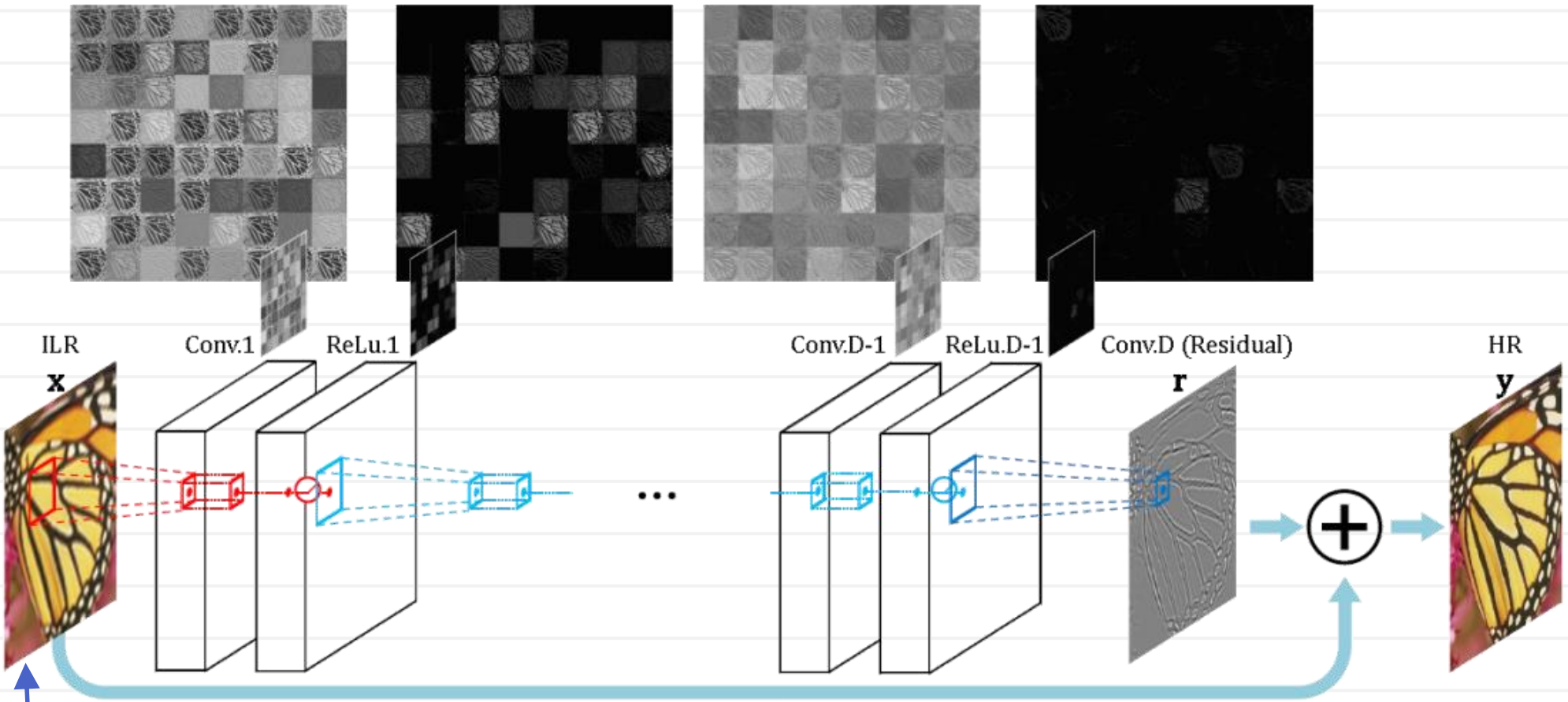


- Input is upscaled to the target resolution
- 19 layers with 3x3 filters (VGG inspired)

Interpolated low-res image

[Kim, Lee, Lee CVPR16]

Image superresolution with ConvNets

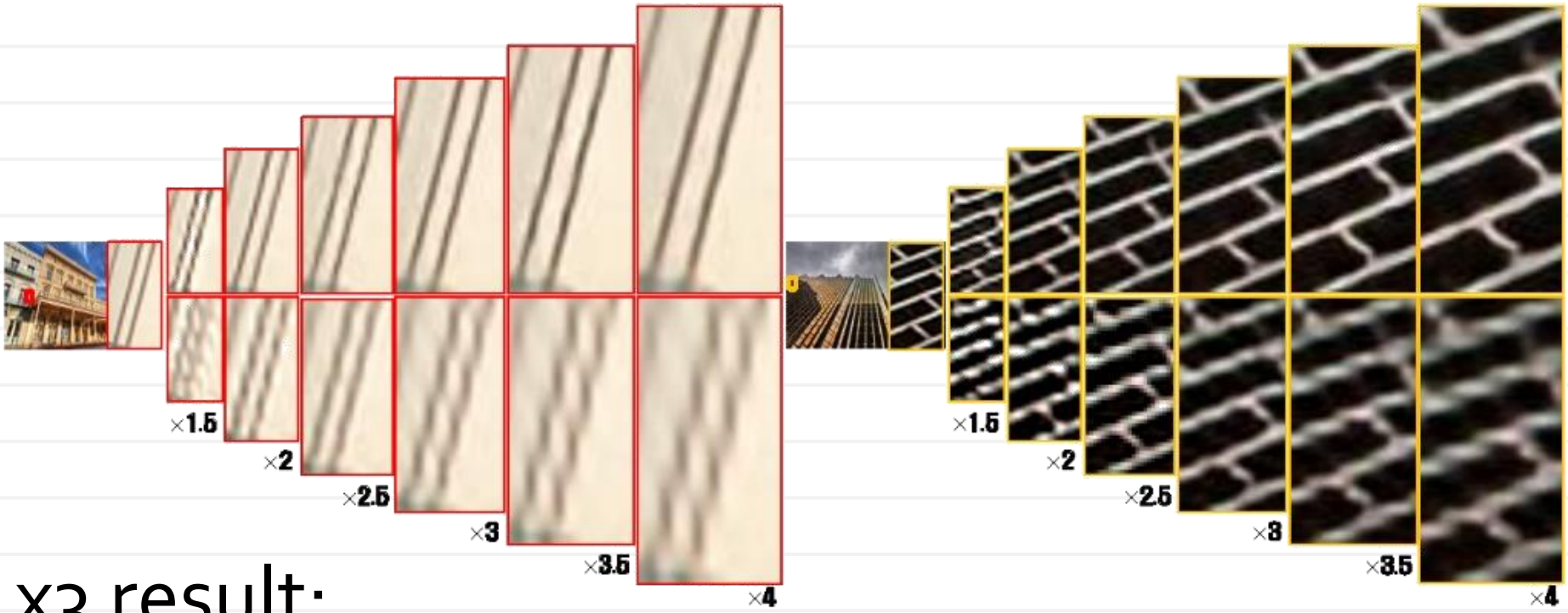


$$\hat{\theta} = \arg\min_{\theta} \sum_{i} \| f(x_i; \theta) - y_i \|^2$$

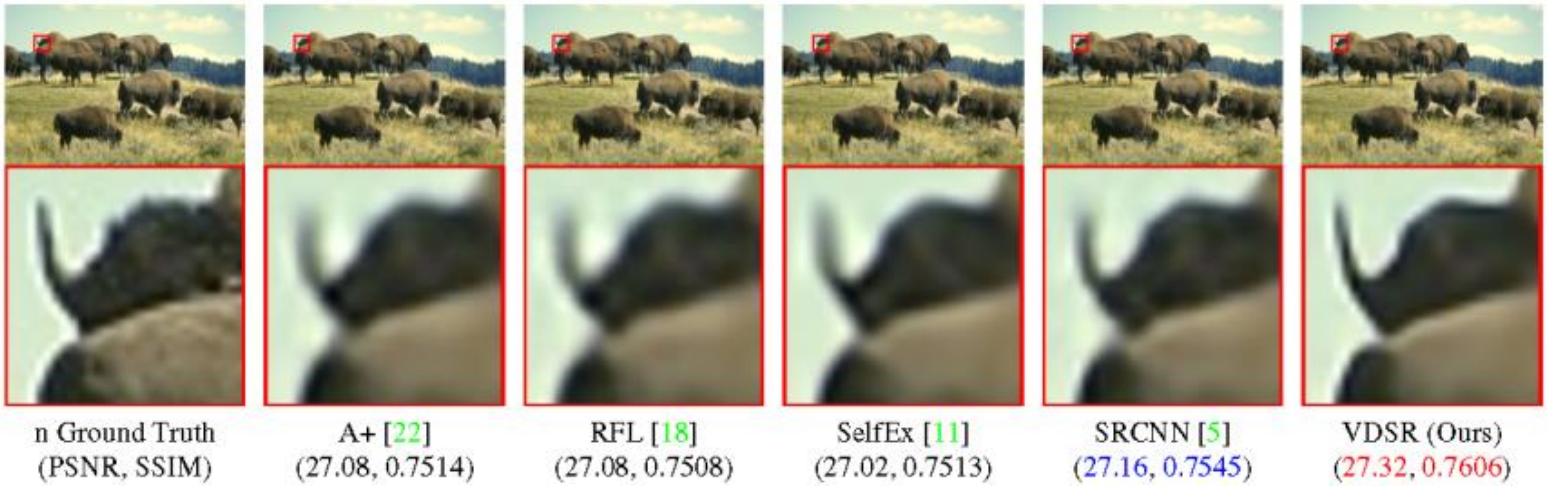
Interpolated low-res image

[Kim et al. CVPR16]

Image superresolution with ConvNets



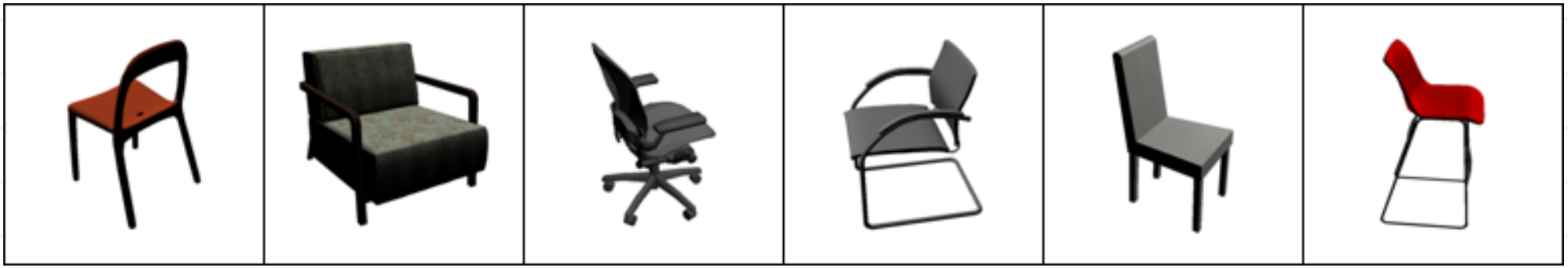
x3 result:



[Kim et al. CVPR16]

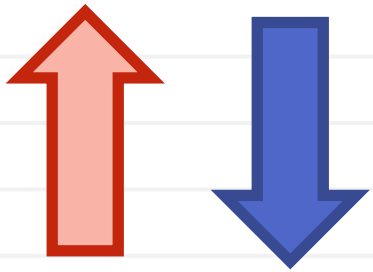
Neural rendering: 3D graphics via ConvNets

Source images:



x_i = chair ID and camera parameters

Recognition
(*computer vision*)



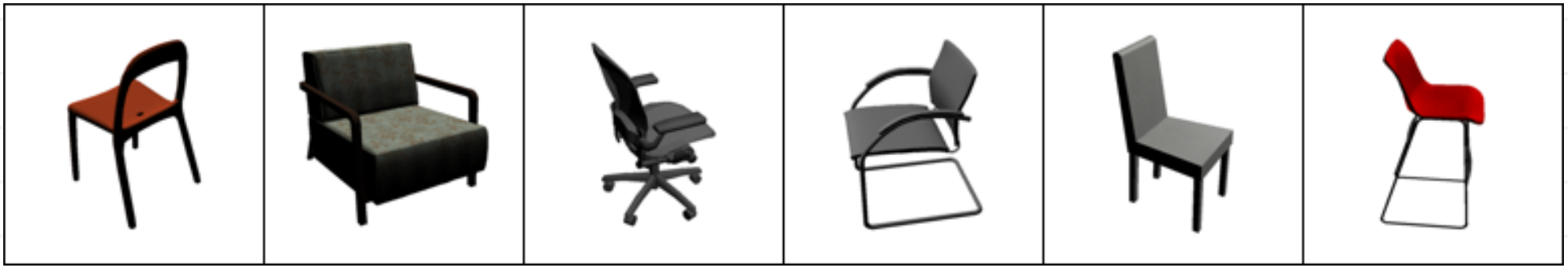
Rendering
(*computer graphics*)

y_i = RGB image + Binary foreground mask

[Dosovitskiy, Springerber, Brox CVPR15]

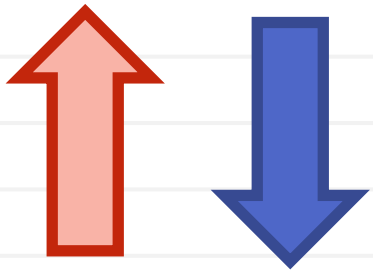
Neural rendering: 3D graphics via ConvNets

Source images:



x_i = chair ID and camera parameters

Recognition
(*computer vision*)



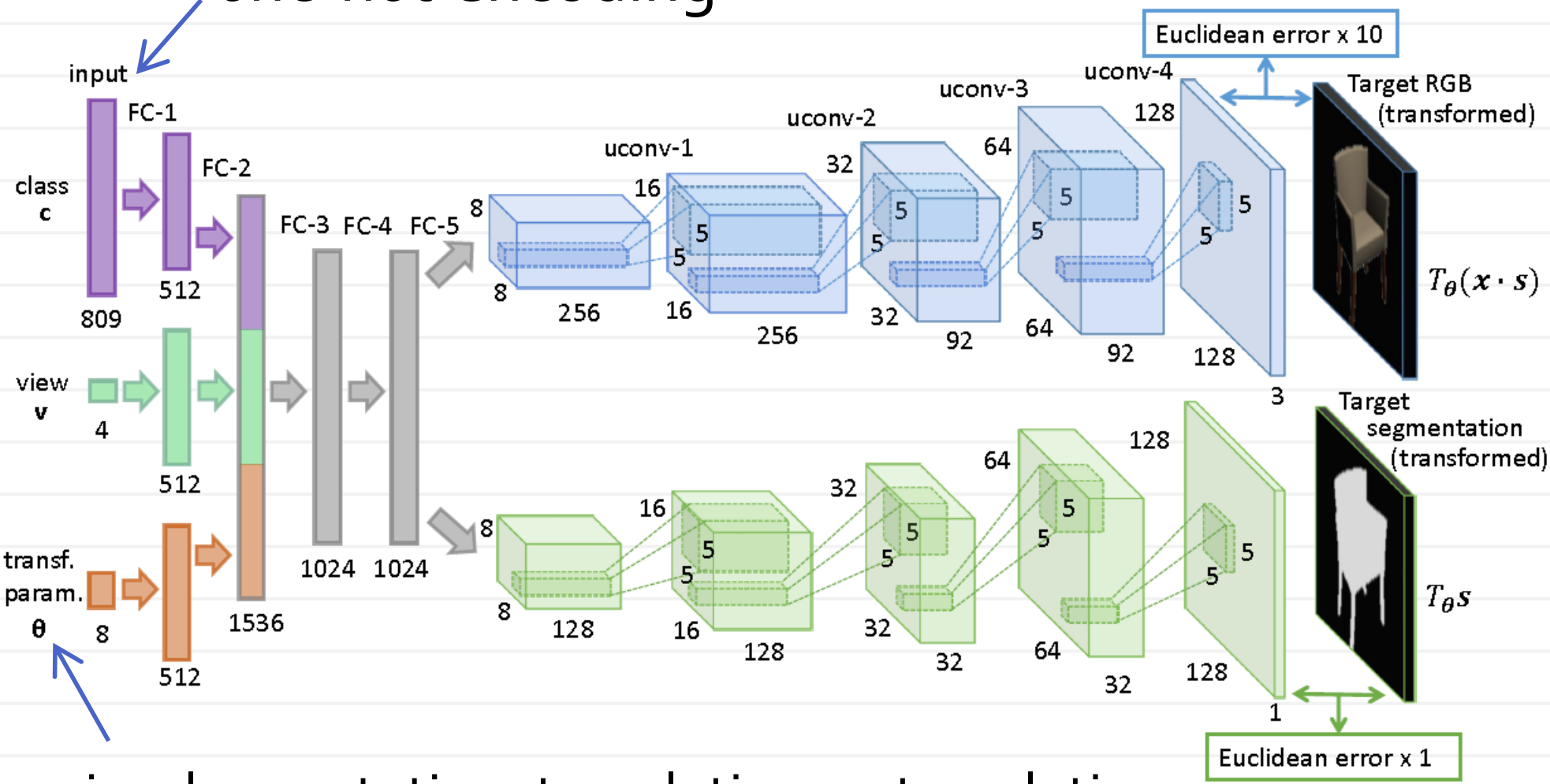
Generative ConvNet

y_i = RGB image + Binary foreground mask

[Dosovitskiy, Springerber, Brox CVPR15]

Feed-forward conditional generation

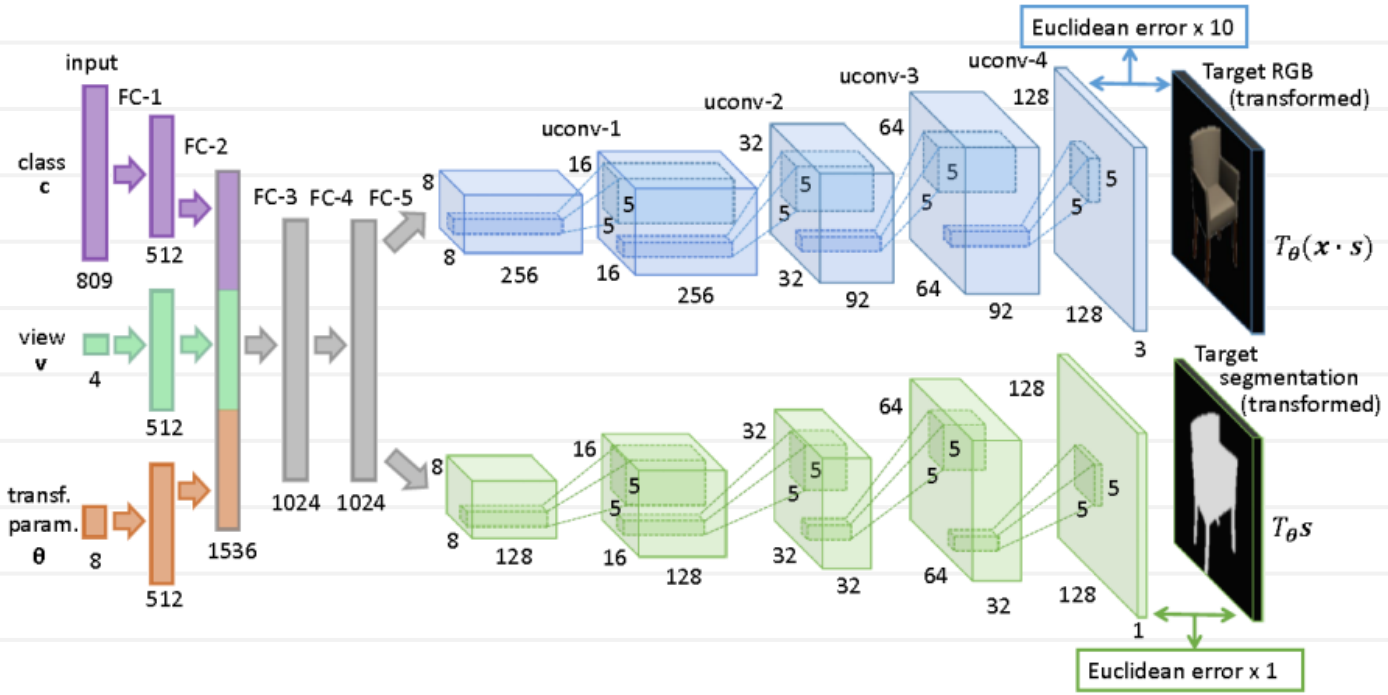
one-hot encoding



in-plane rotation, translation_x, translation_y, zoom_x, zoom_y, hue, saturation, brightness

[Dosovitskiy et al. CVPR15]

Feed-forward conditional generation



x_i = chair ID and camera parameters

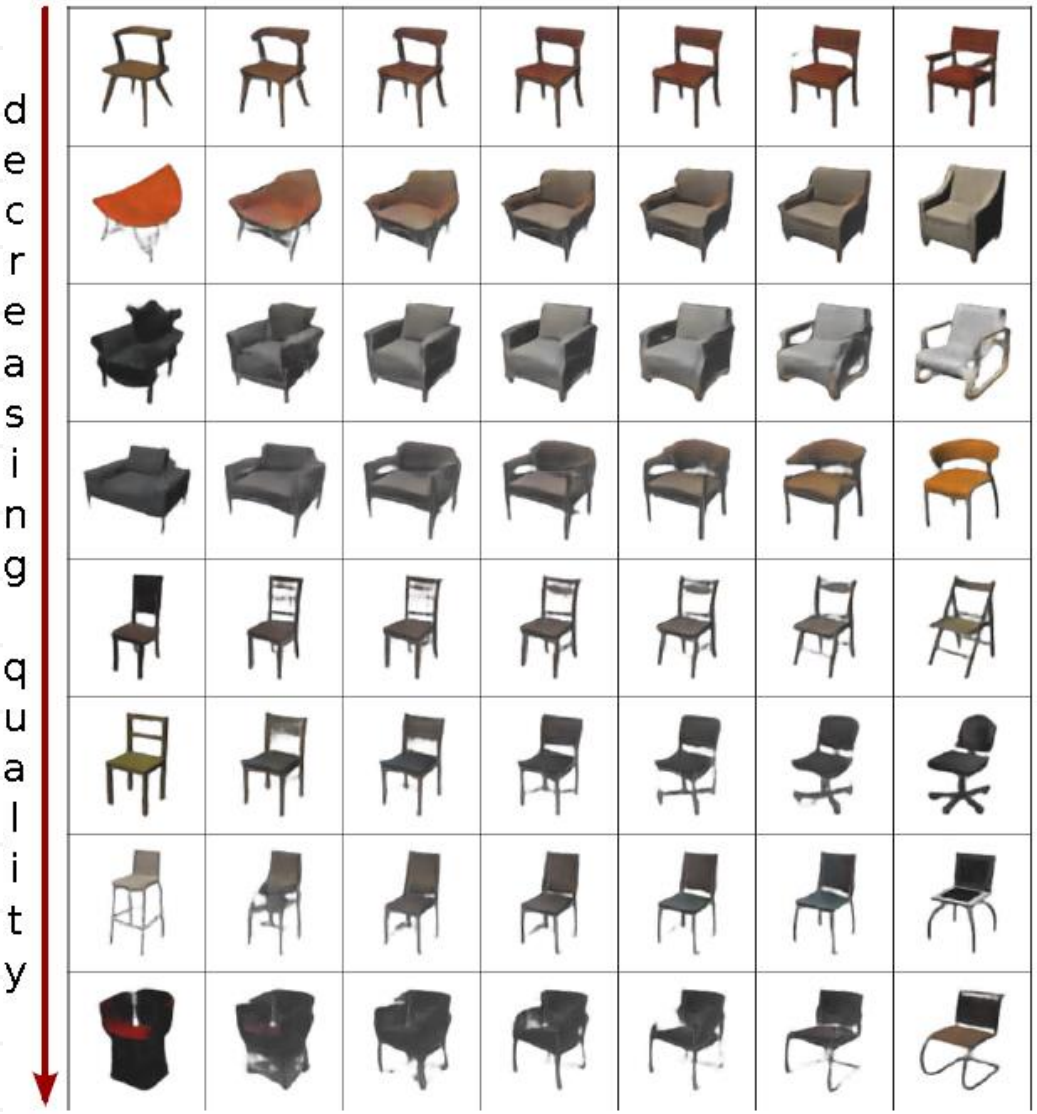
y_i = RGB image + Binary foreground mask

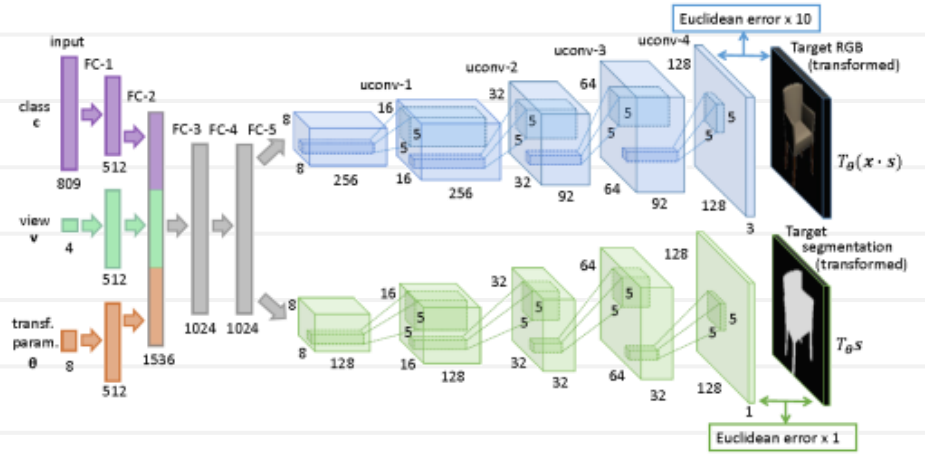
Training with MSE loss:

$$\hat{\theta} = \arg\min_{\theta} \sum_{i} \|f(x_i;\theta) - y_i\|^2$$

[Dosovitskiy et al. CVPR15]

Feed-forward conditional generation

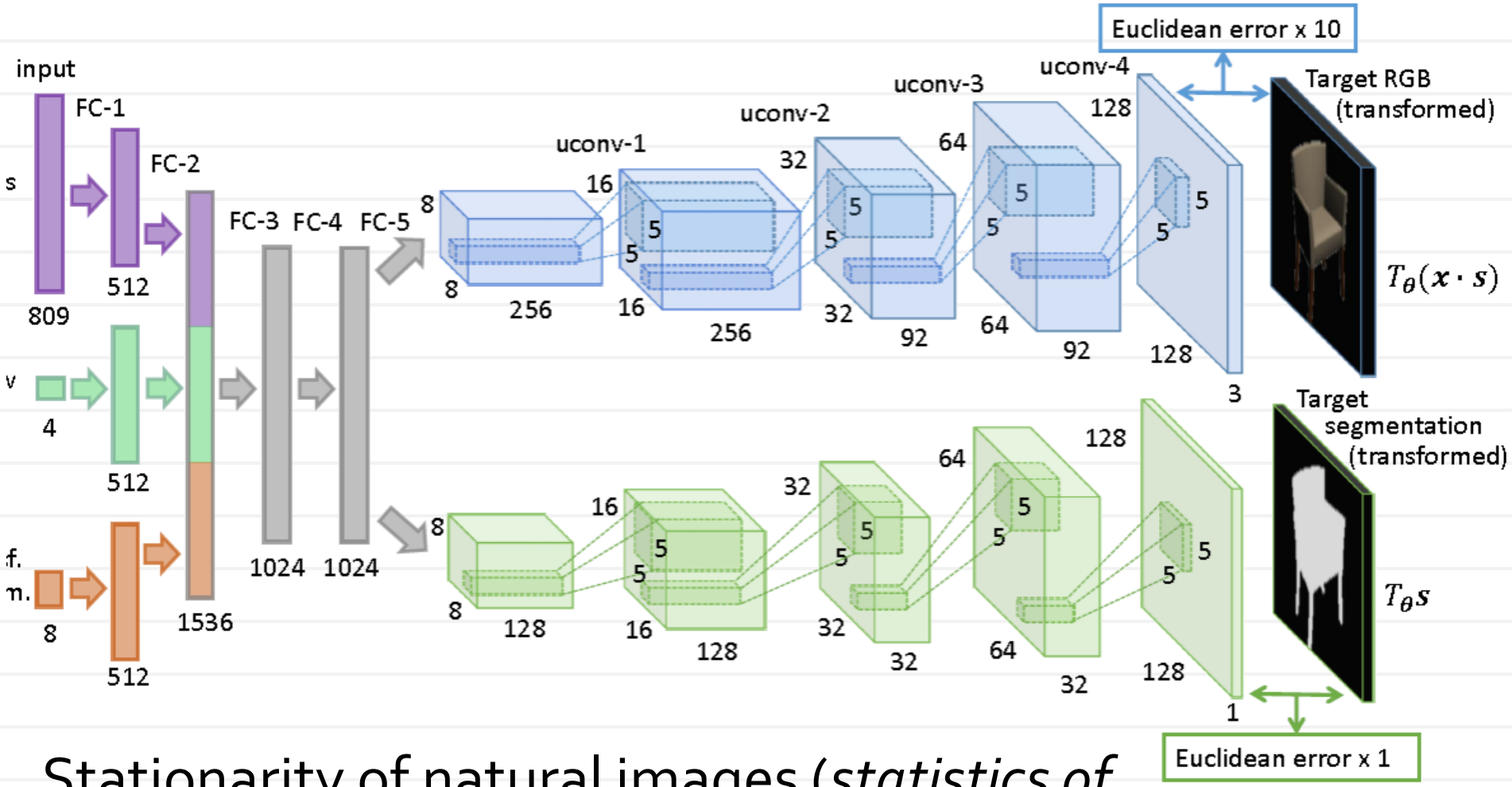




Morphing between chairs

[Dosovitskiy et al. CVPR15]

Why does it work?

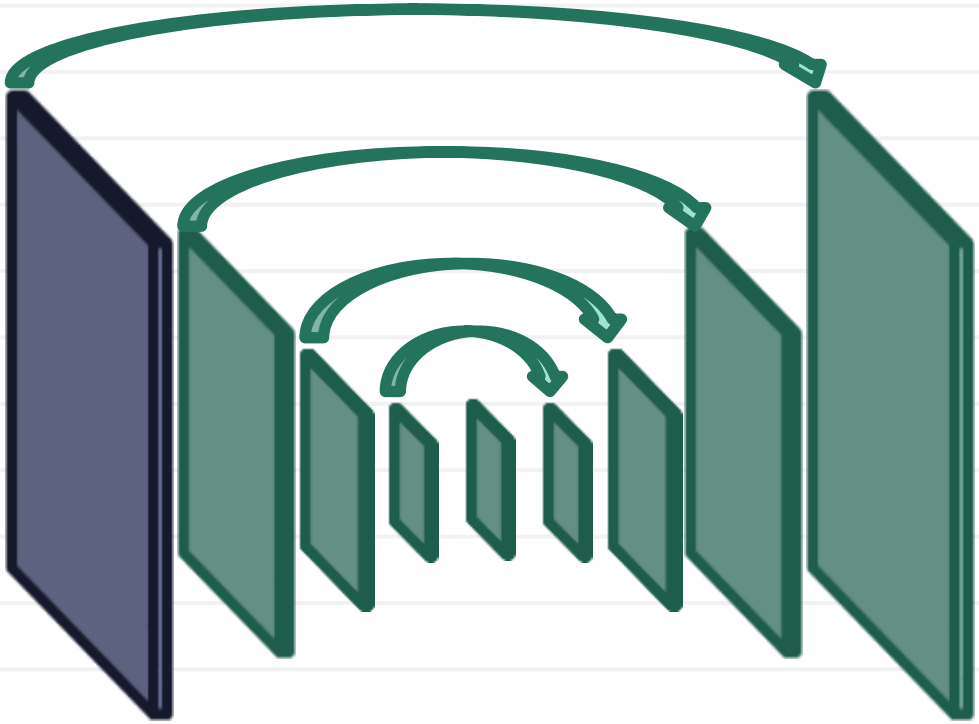


- Stationarity of natural images (*statistics of image appearance are similar across the field of view*)
- Abundance of supervision

Deep inpainting without training

Deep inpainting without training

Fixed noise maps *z*



Output *x*

- U-net with thin skip connections (containing convolution blocks)
- Convolution block: conv + batch norm + leaky ReLU + (up/down-sample)
- ~2 millions of parameters
- Fit to known pixel values

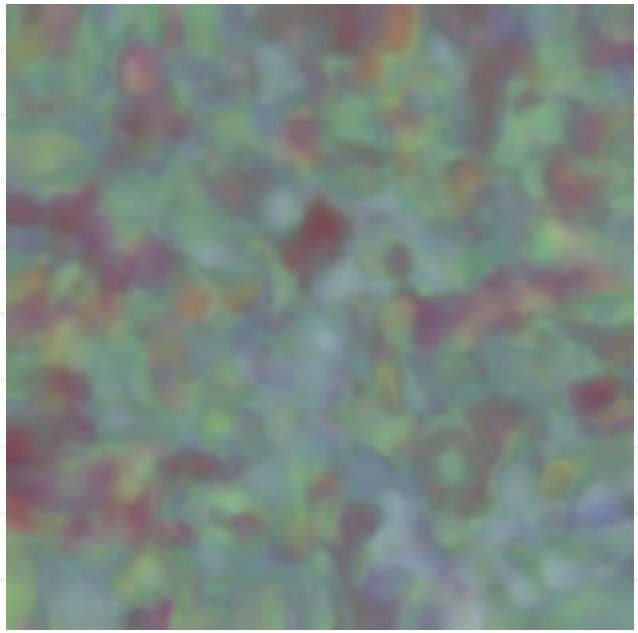
$$\min_{\theta} \|(f(z;\theta) - x_0) \odot m\|^2$$

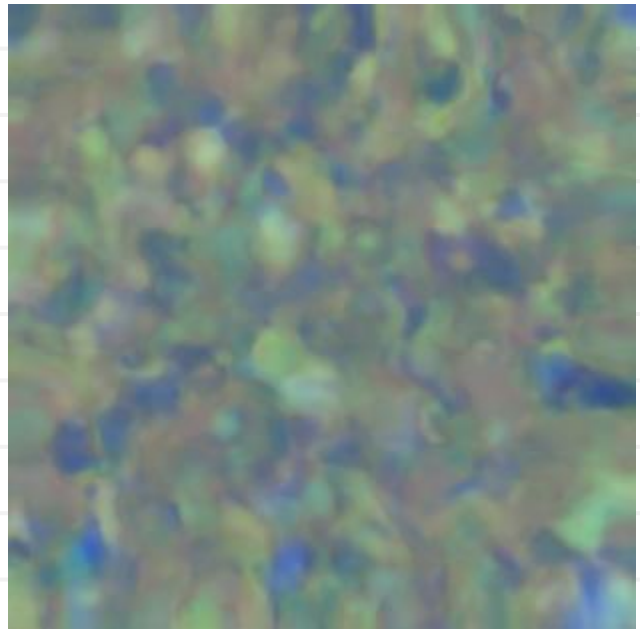
Deep inpainting without training

[Ulyanov et al. CVPR18]

"Samples" from the Deep Image Prior

Hourglass:





UNet:

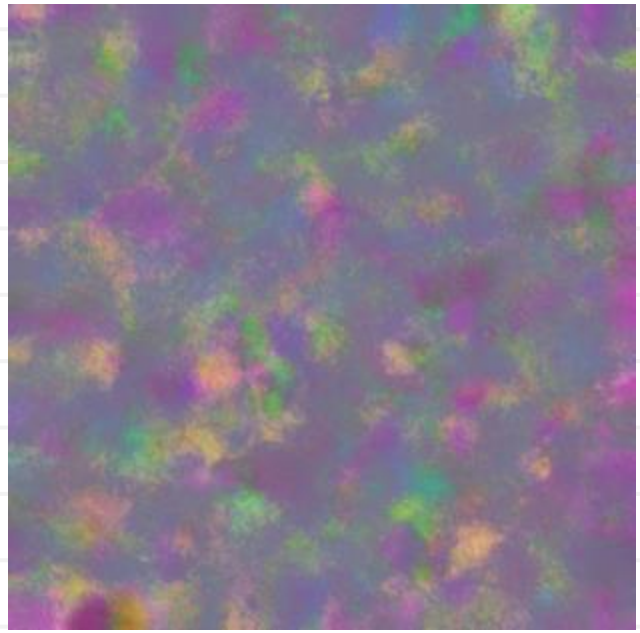


Image restoration with *Deep Image Prior*

Consider all images obtained from a random signal *z* via a convolutional network with a certain architecture :

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

Fixed input

Convolutional network with parameters θ

Perform the reconstruction by solving (*optionally: use fixed number of iterations*):

$$\min_\theta E(f_\theta(z); x_0)$$

Task-specific likelihood

"Find the most likely image that can be generated by a ConvNet from z"

[Ulyanov et al. CVPR18]

Image restoration: standard approach

- x – Recovered image
- x_0 – Corrupted image (observed)

Regularization / Prior term

$$\arg\min_x E(x; x_0) + R(x)$$

- **Denoising:** $E(x; x_0) = \|x - x_0\|^2$
- **Inpainting:** $E(x; x_0) = \|(x - x_0) \odot m\|^2$
- **Super-Res.:** $E(x; x_0) = \|d(x) - x_0\|^2$
- **Feature inv.:** $E(x; x_0) = \|\Phi(x) - \Phi(x_0)\|^2$

Deep superresolution *without* dataset learning

[Ulyanov et al. CVPR18]

Image restoration

$$E(x; x_0) = \|(x - x_0) \odot m\|^2$$

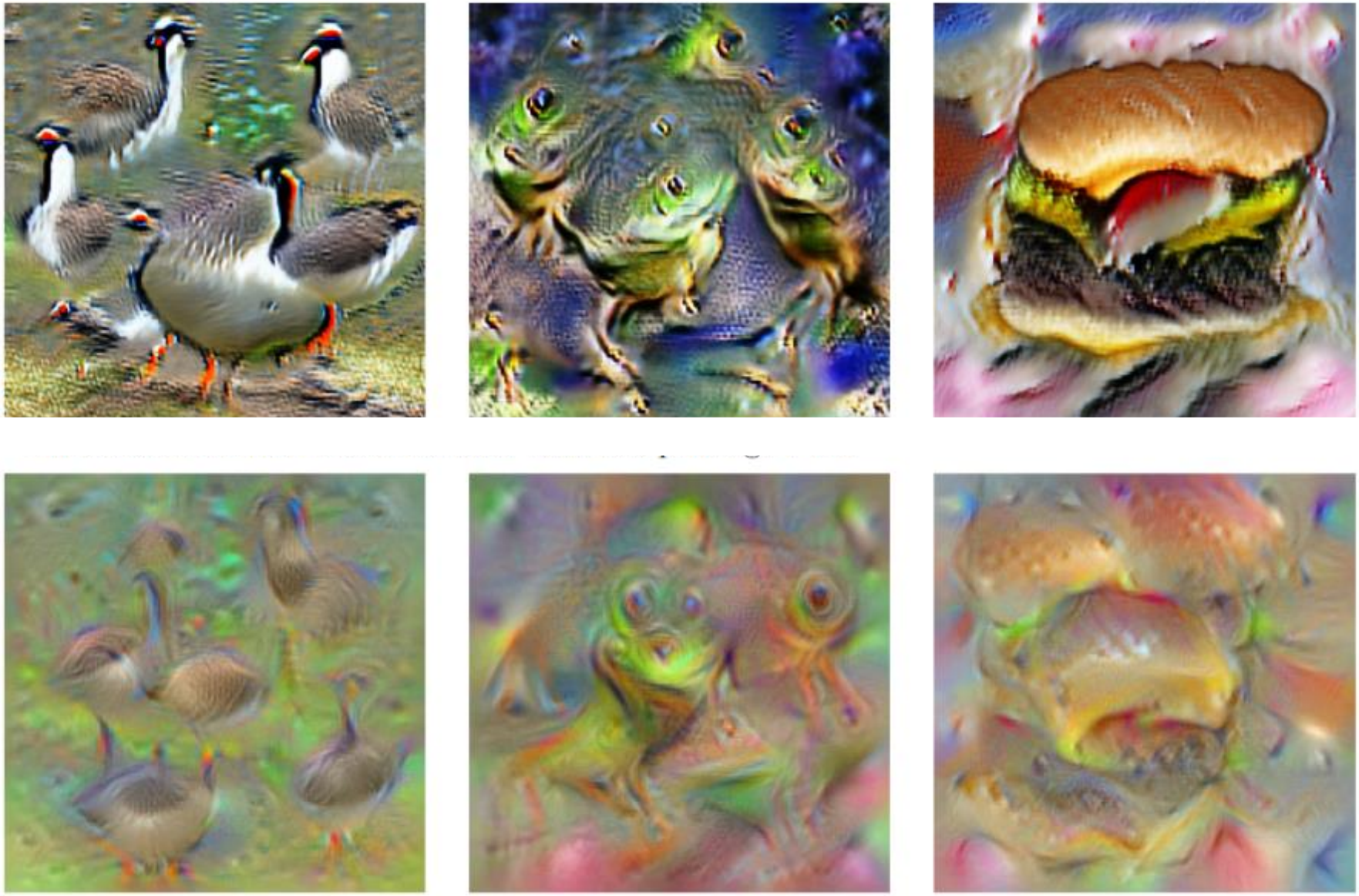
Corrupted

Deep image prior

[Ulyanov et al. CVPR18]

Activation maximization

deep image prior



total variation prior

The pitfall of the pixel-wise losses

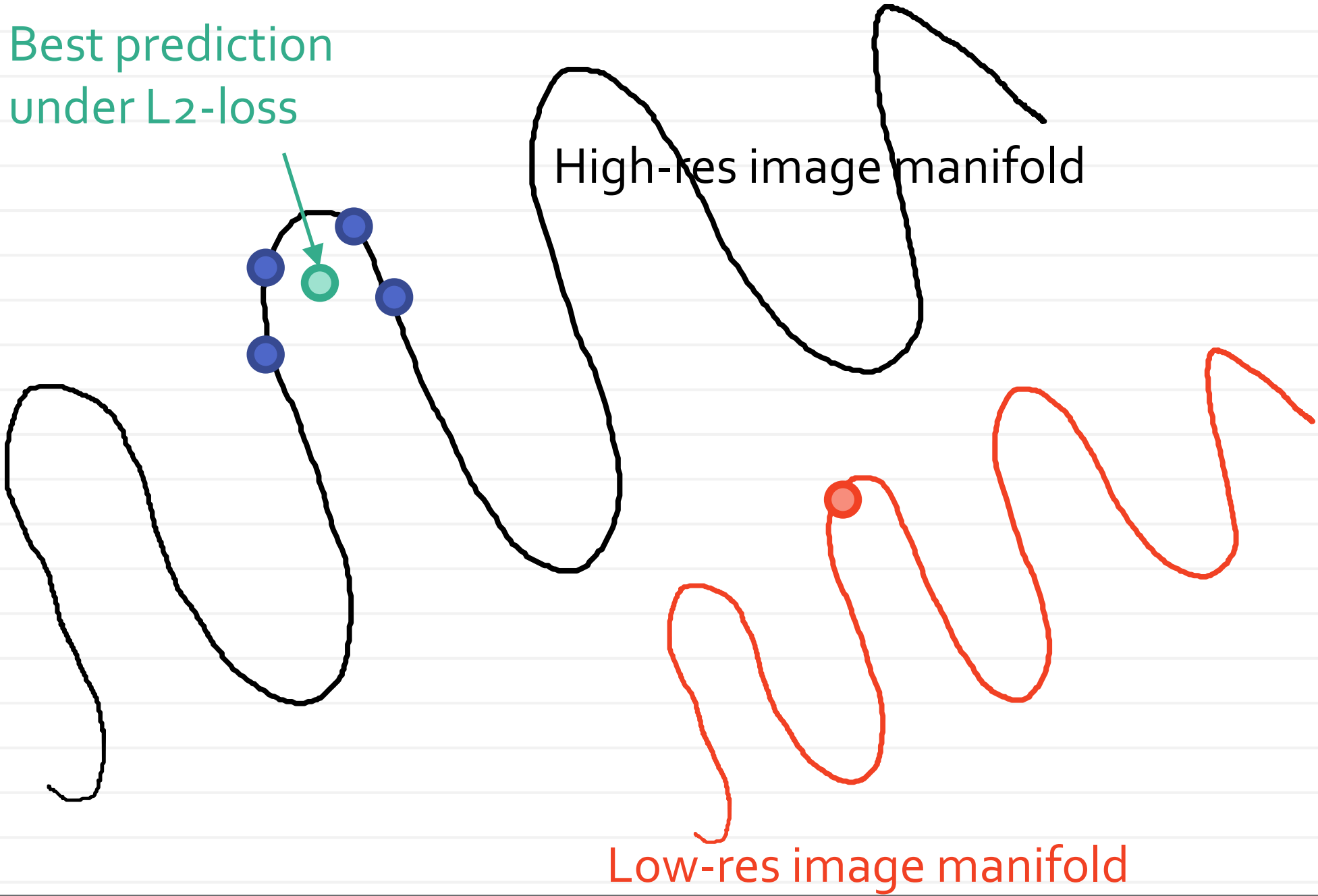
x_i

y_i

$$\hat{\theta} = \arg\min_{\theta} \sum_{i} \|f(x_i;\theta) - y_i\|^2$$

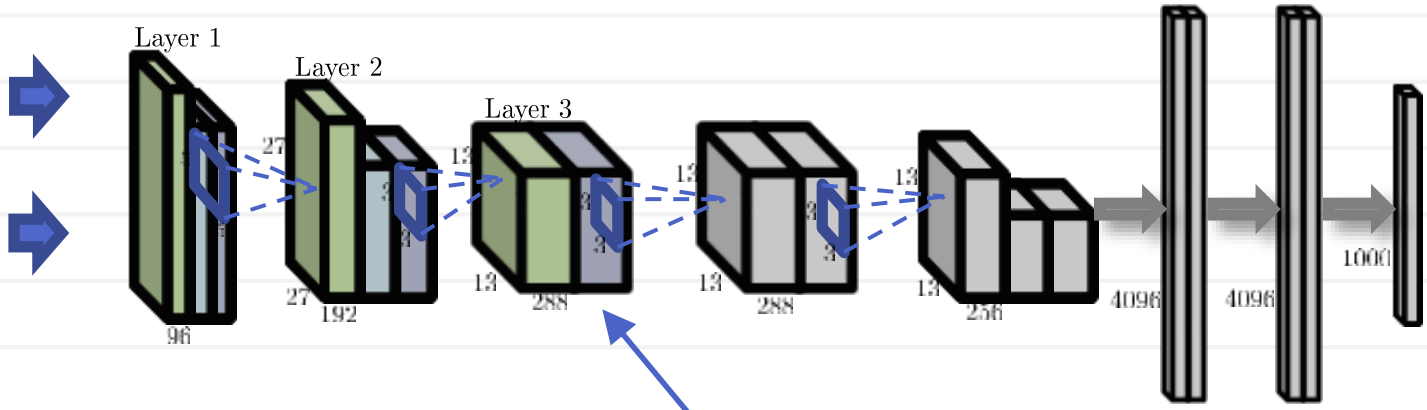
Averaging over jittered versions = blurriness

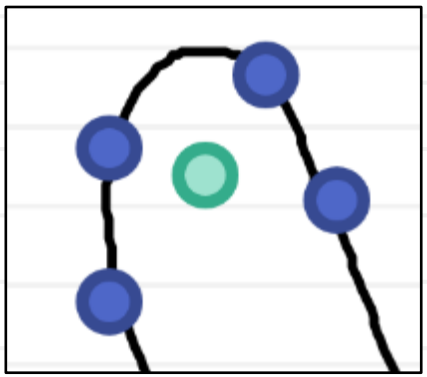
The pitfall of the pixel-wise losses



Perceptual loss functions

[Gatys, Ether, Bethge CVPR16]

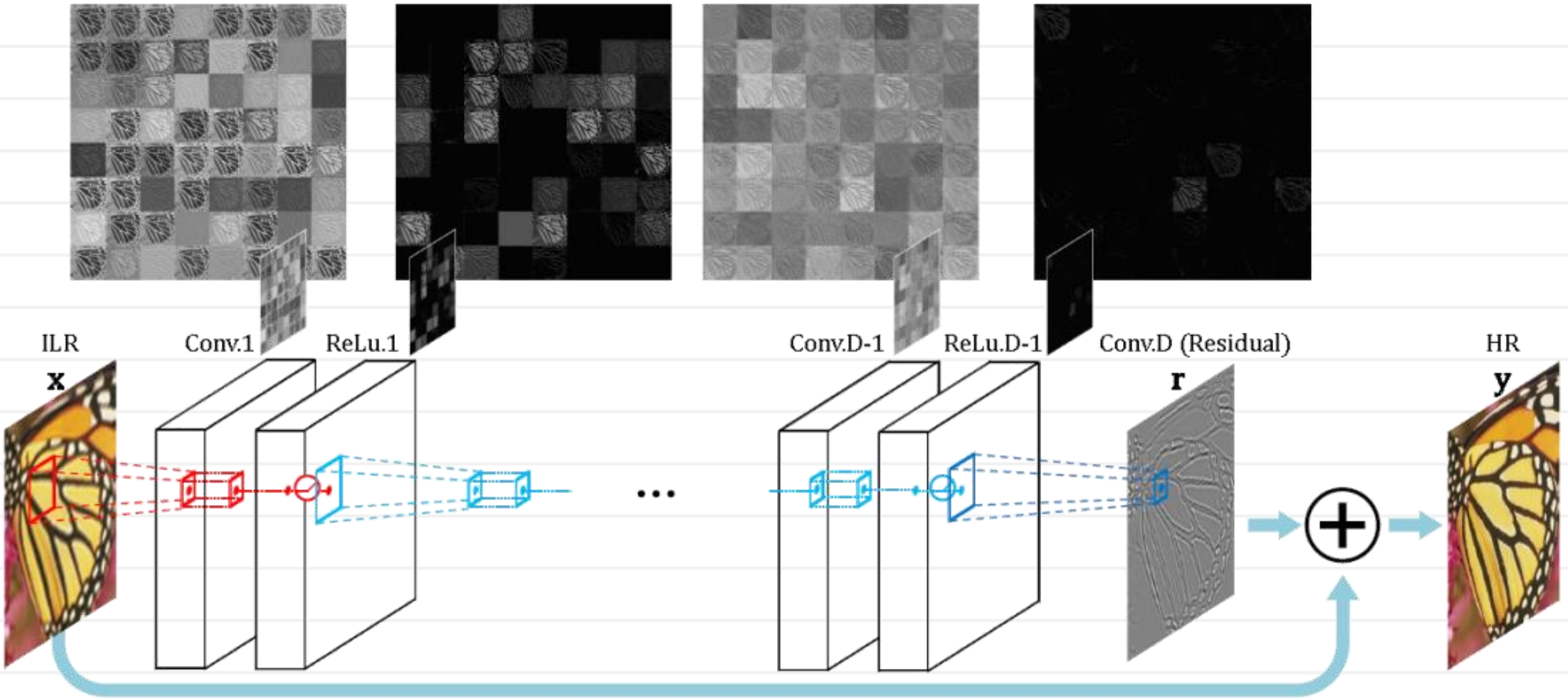




$$L(y, y_0) = \sum_{l \in L_C} \sum_m \|F_m^l(y) - F_m^l(y_0)\|_2^2$$

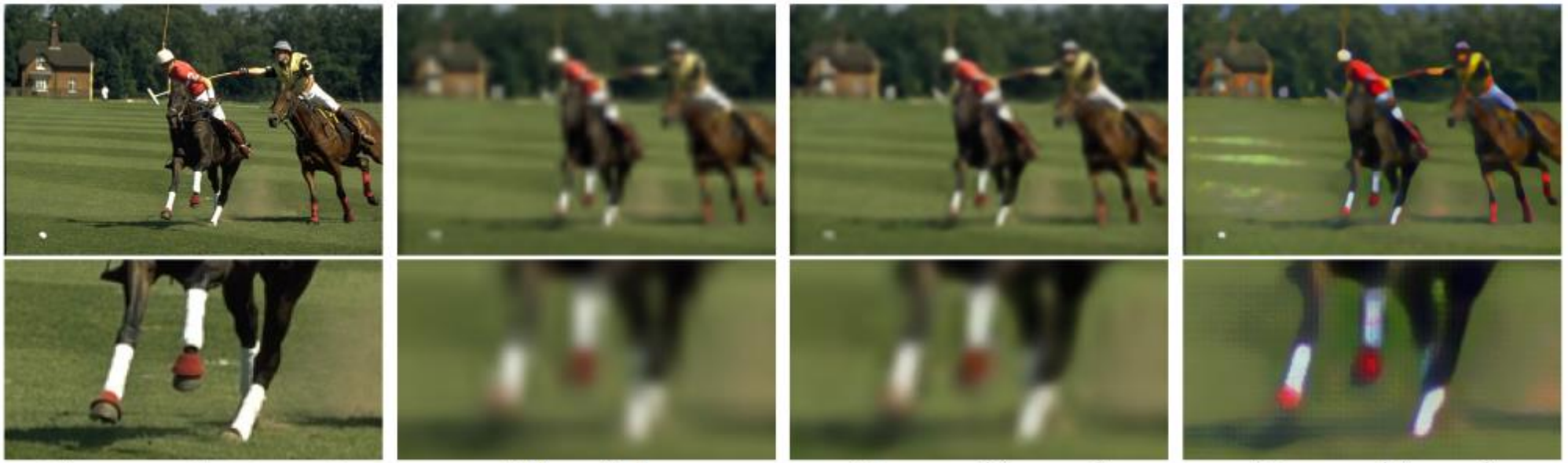
- Averaged image is far from all blue
- Blue points are similar to each other

Recap: superresolution using ConvNets



[Kim et al. CVPR16]

Superresolution with perceptual losses



Ground Truth	Bicubic	Ours (ℓ_{pixel})	Ours (ℓ_{feat})
This image	22.75 / 0.5946	23.42 / 0.6168	21.90 / 0.6083
Set5 mean	23.80 / 0.6455	24.77 / 0.6864	23.26 / 0.7058
Set14 mean	22.37 / 0.5518	23.02 / 0.5787	21.64 / 0.5837
BSD100 mean	22.11 / 0.5322	22.54 / 0.5526	21.35 / 0.5474

[Johnson, Alahi, Fei-Fei ECCV2016]

Superresolution with perceptual losses

Ground Truth	Bicubic	Ours (ℓ_{pixel})	SRCNN [13]	Ours (ℓ_{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

[Johnson et al. ECCV2016]

High-res generation with perceptual losses

[Chen and Koltun ICCV17]

High-res generation with perceptual losses

[Chen and Koltun ICCV17]

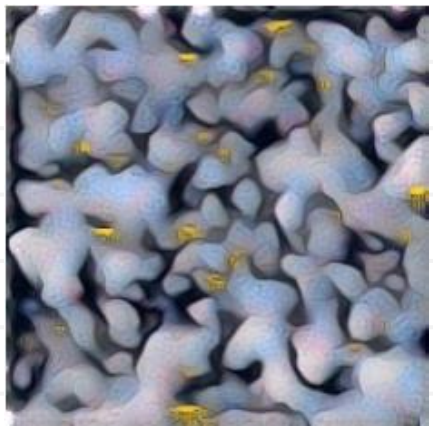
High-res generation with perceptual losses

- Cascade of refinement modules starting from 4x8 upto 1024x2048
- Simple variant outputs one variant and uses perceptual loss
- **Multiple-choice learning** improves results: outputting several answers and scoring a composite of these answers ("easy" version of probabilistic output)

[Chen and Koltun ICCV17]

Texture synthesis

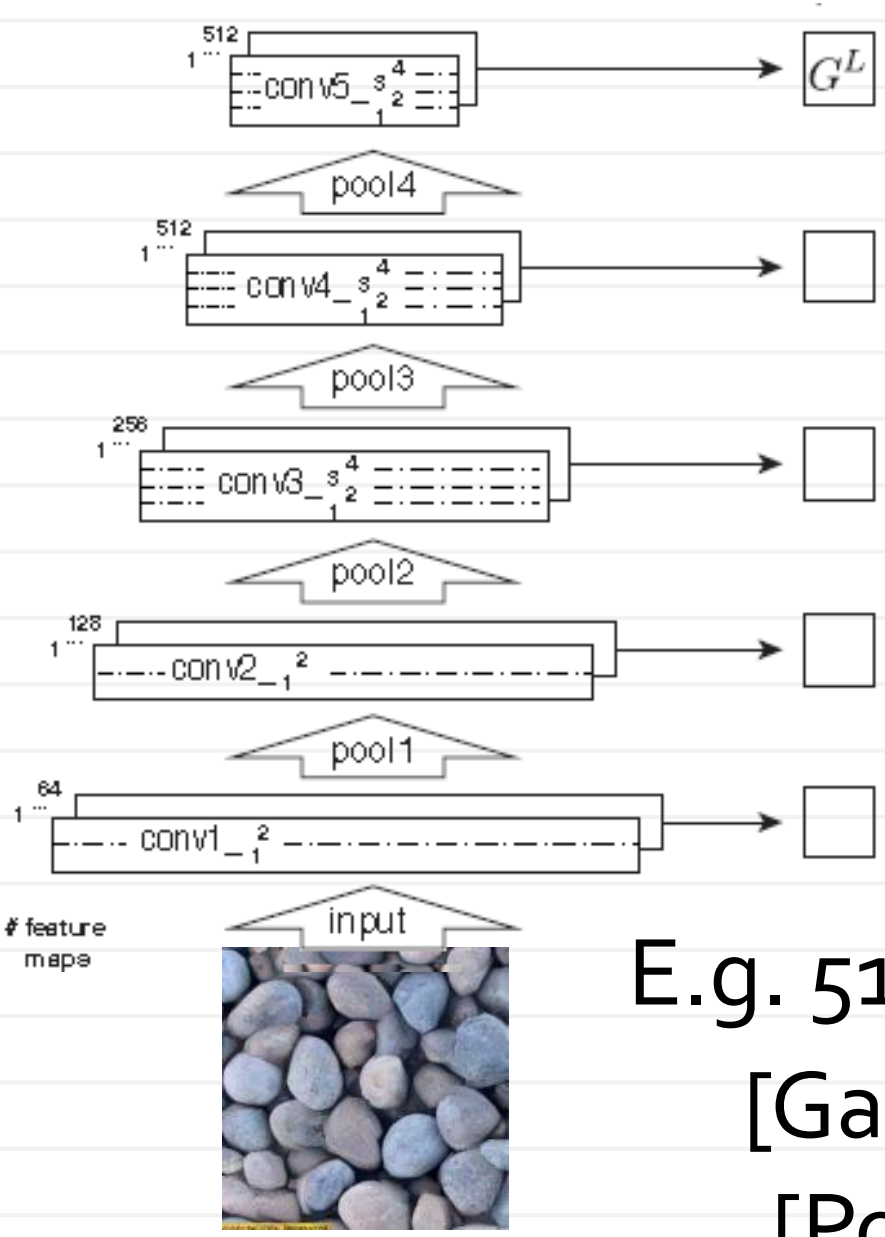
- How can we measure the similarity of *textures (i.e. spatially-stationary natural visual patterns)*?



similar different

- Pixel-wise measures are useless
- Histograms are not too useful
- Long history of research

What describes a texture?



Gram matrix:

$$G^l$$

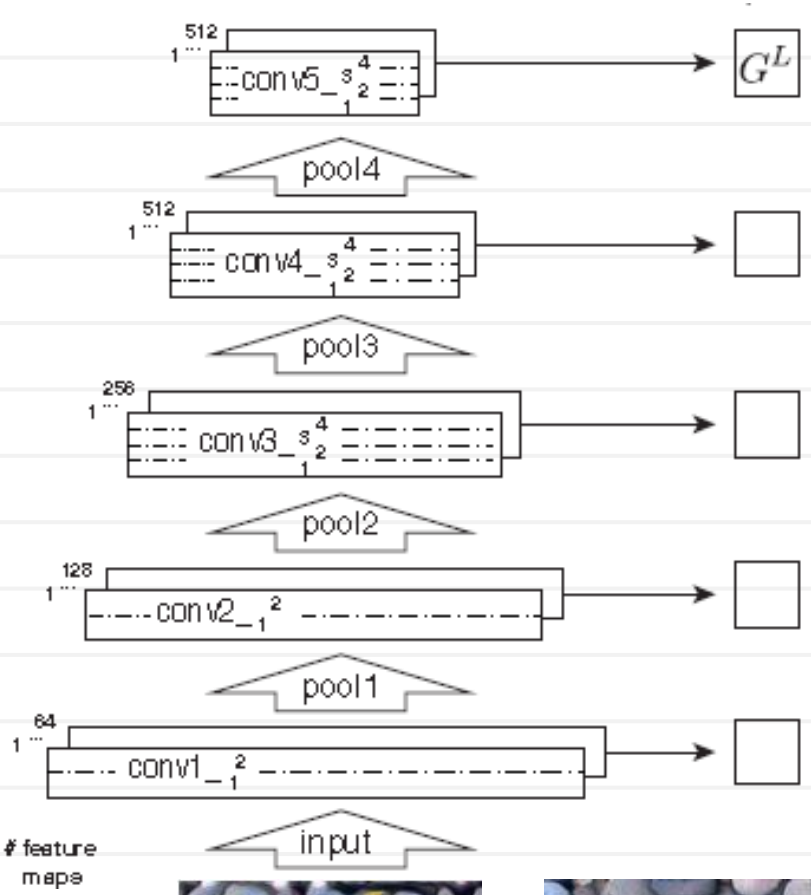
$$G^l_{ij} = \sum_k F^l_{i,k} \cdot F^l_{j,k}$$

E.g. 512x512-dim descriptor

[Gatys, Ecker, Bethge 2015]

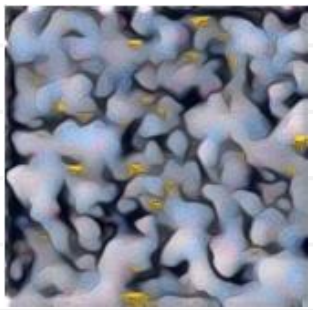
[Portilla Simoncelli 2000]

Pre-image texture synthesis



Texture loss:

$$\mathcal{L}(x) = \sum_l \|G^l(x) - G^l(x^*)\|^2$$



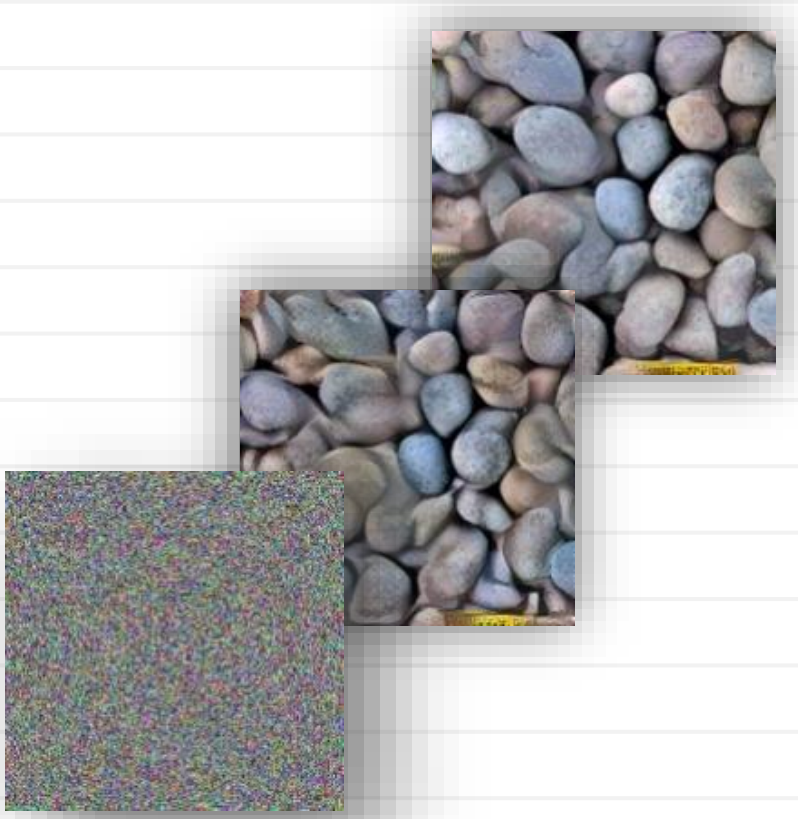
x

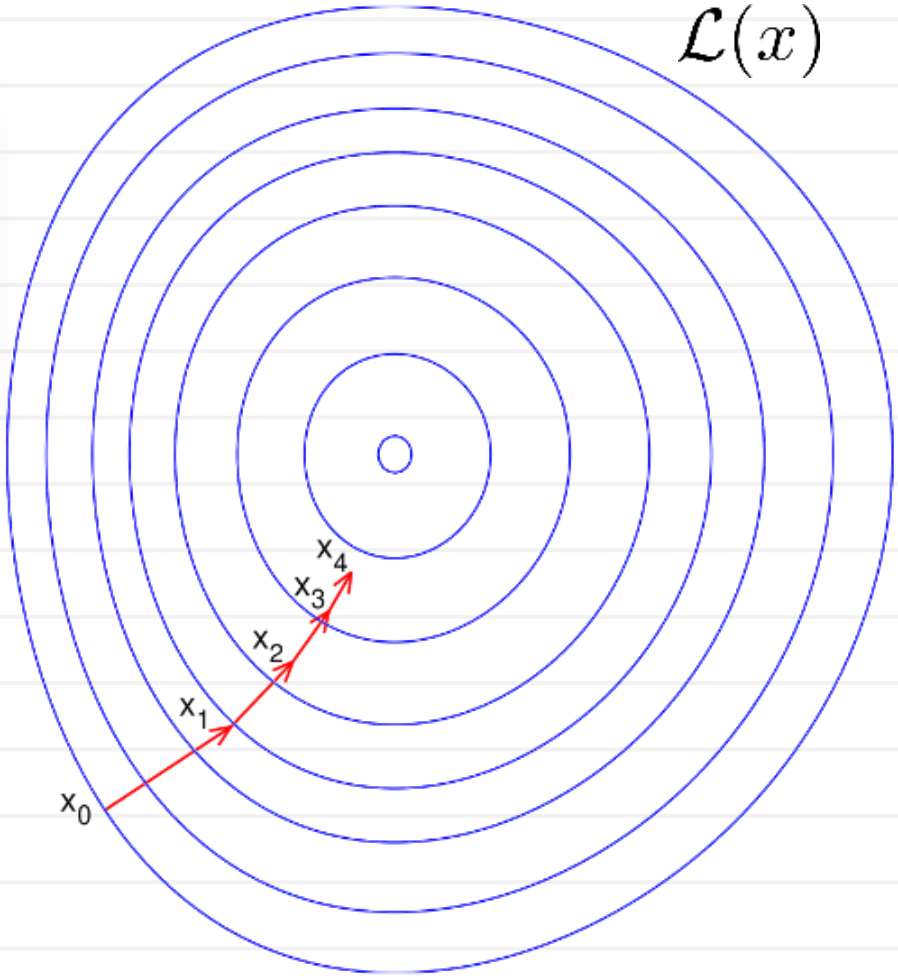
x^*

[Gatys, Ecker, Bethge 2015]

Image generation by optimization

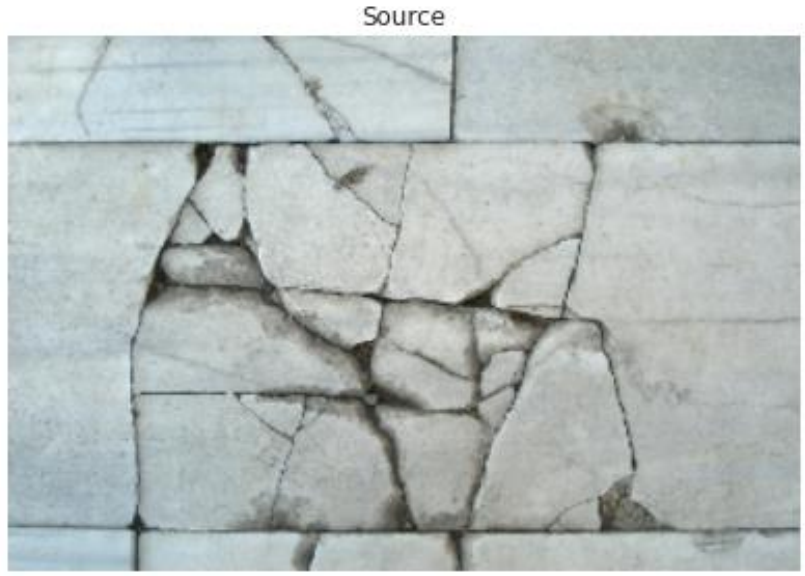
$$x^* = \arg\min_x \mathcal{L}(x)$$





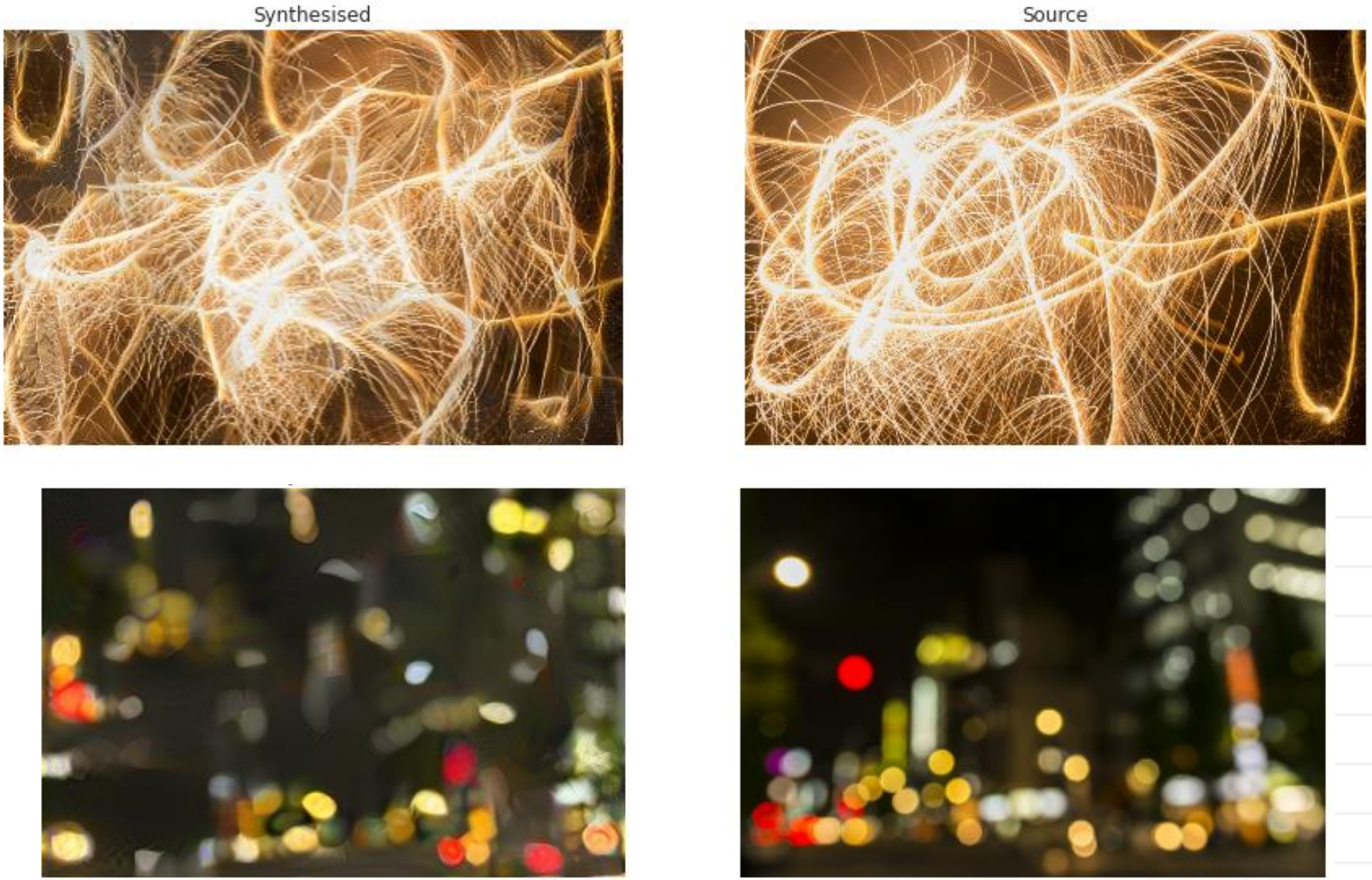
[Gatys, Ecker, Bethge 2015]

Pre-image texture synthesis



[Gatys, Ecker, Bethge 2015]

Pre-image texture synthesis



[Gatys, Ecker, Bethge 2015]

Pre-image texture synthesis

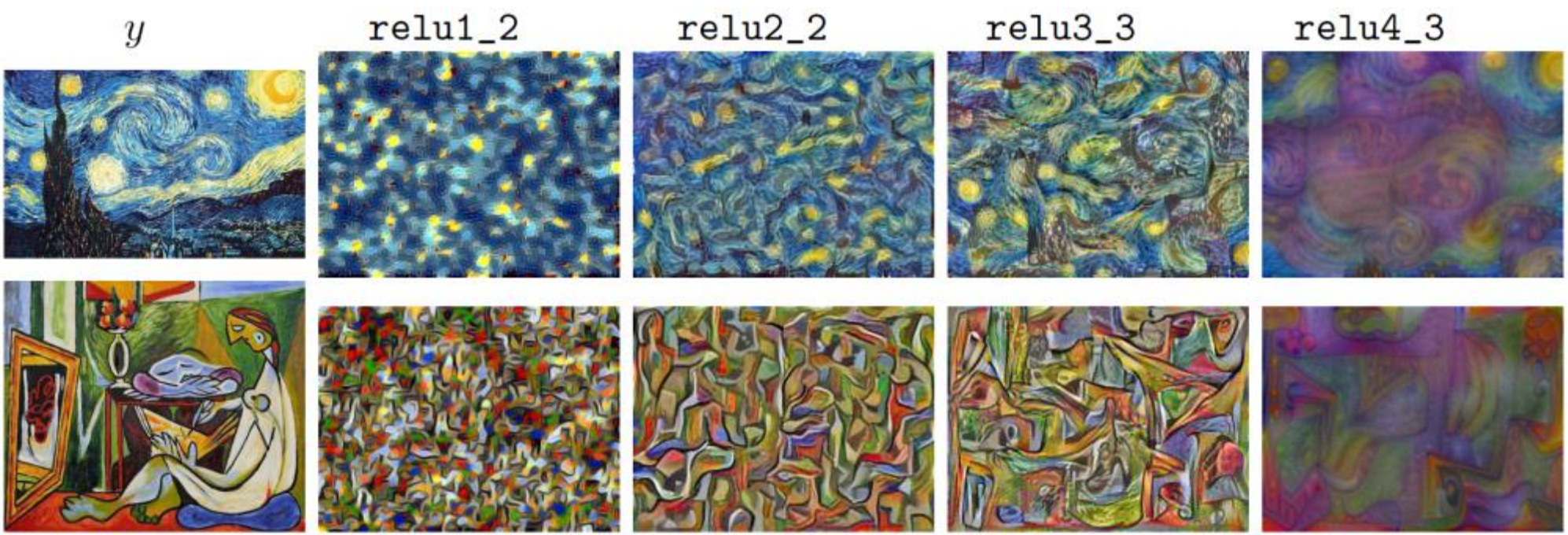
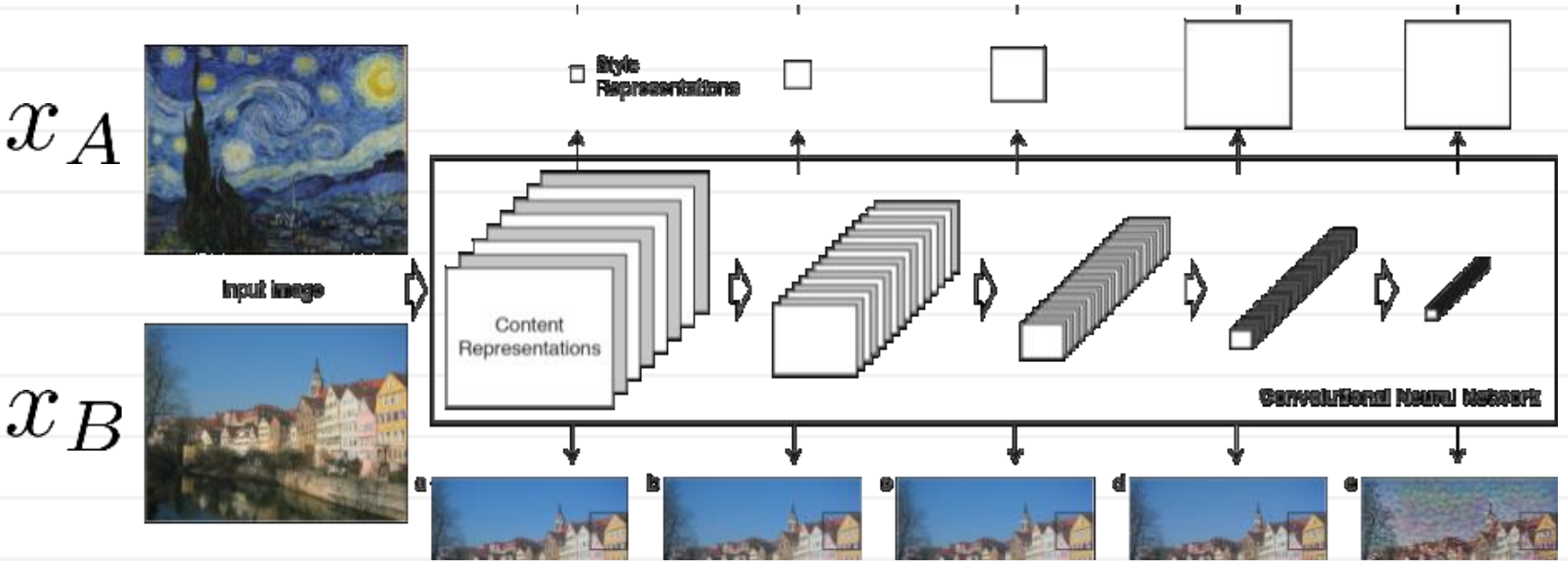


Image from [Johnson et al. ECCV16]

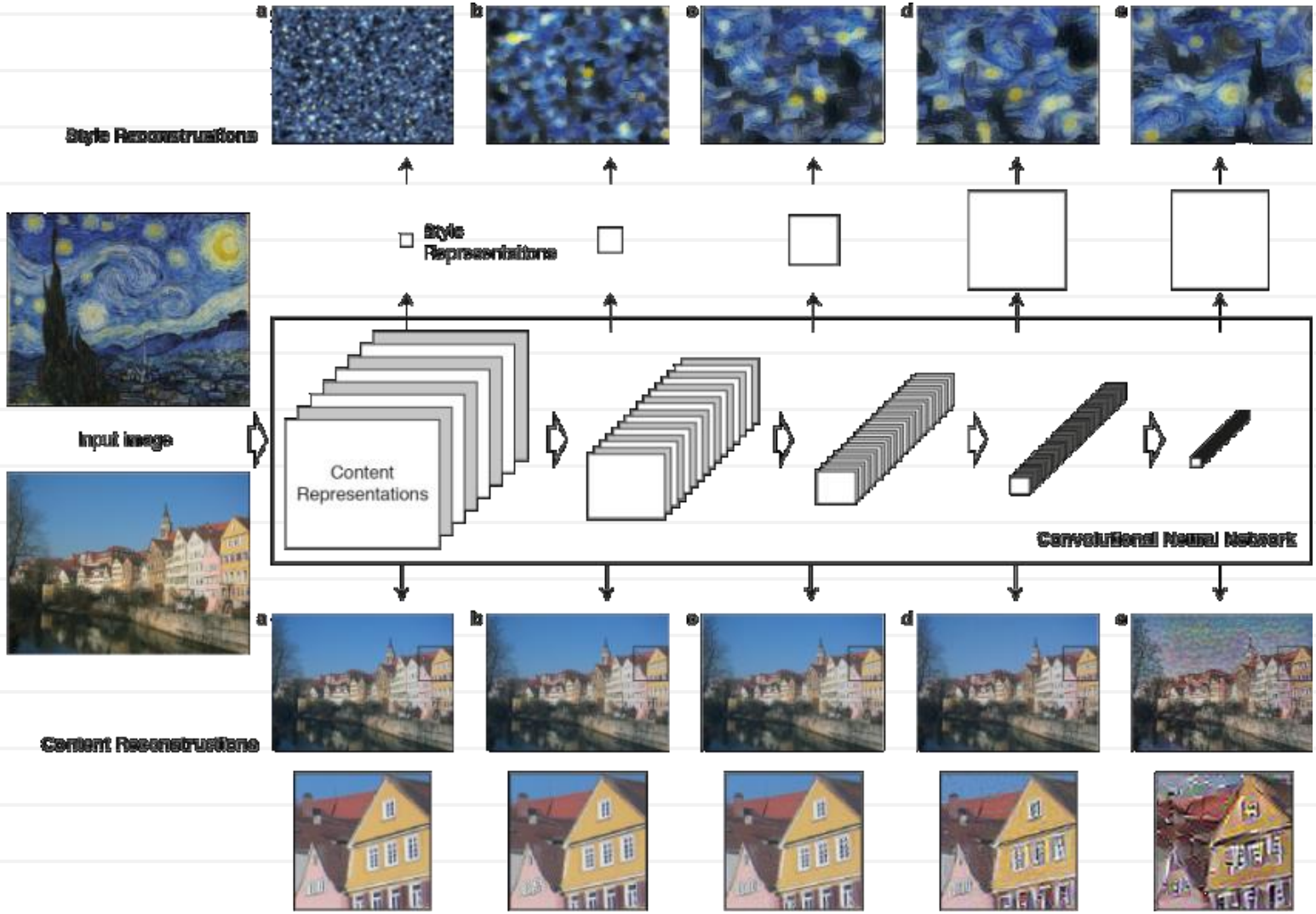
Stylization with texture loss



$$\mathcal{L}(x) = \lambda \sum_{l \in T} \|G^l(x) - G^l(x_A)\|^2 + \sum_{l \in C} \|F^l(x) - F^l(x_B)\|^2$$

[Gatys et al. 2015]

Stylization with texture loss

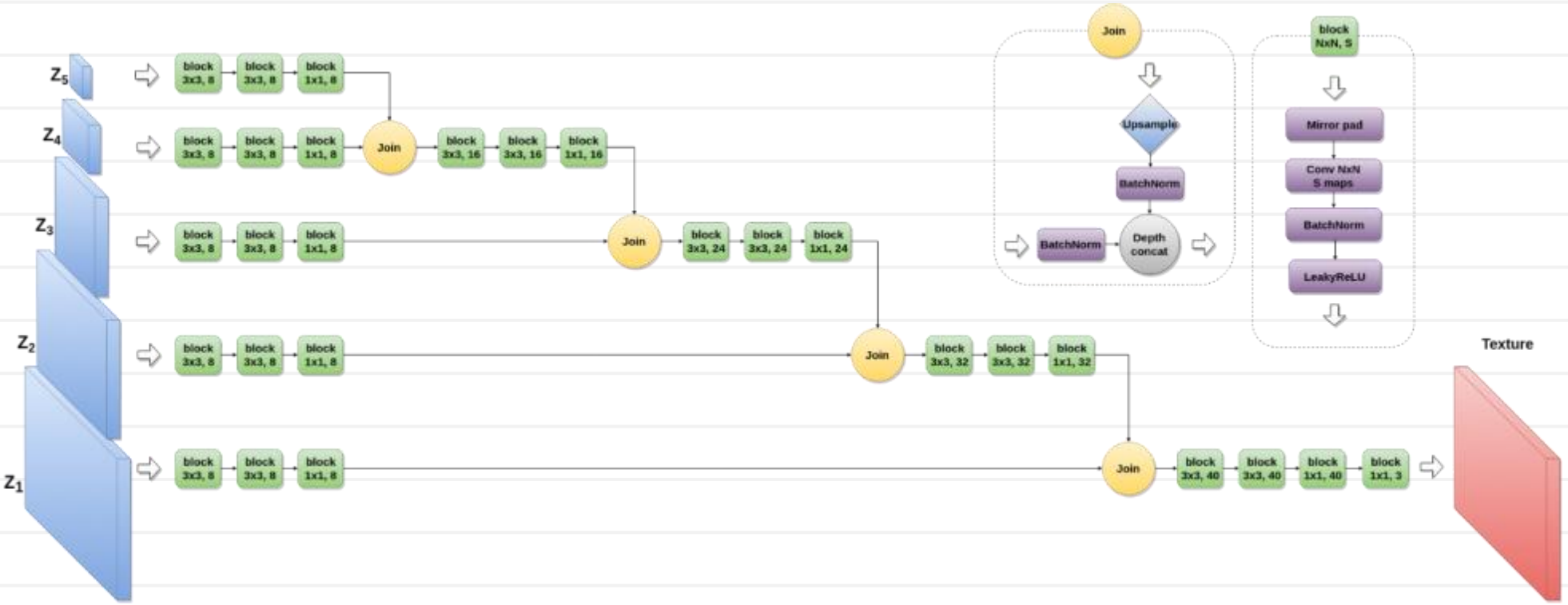


[Gatys et al. 2015]

Neural algorithm for artistic style

[Gatys et al. 2016]

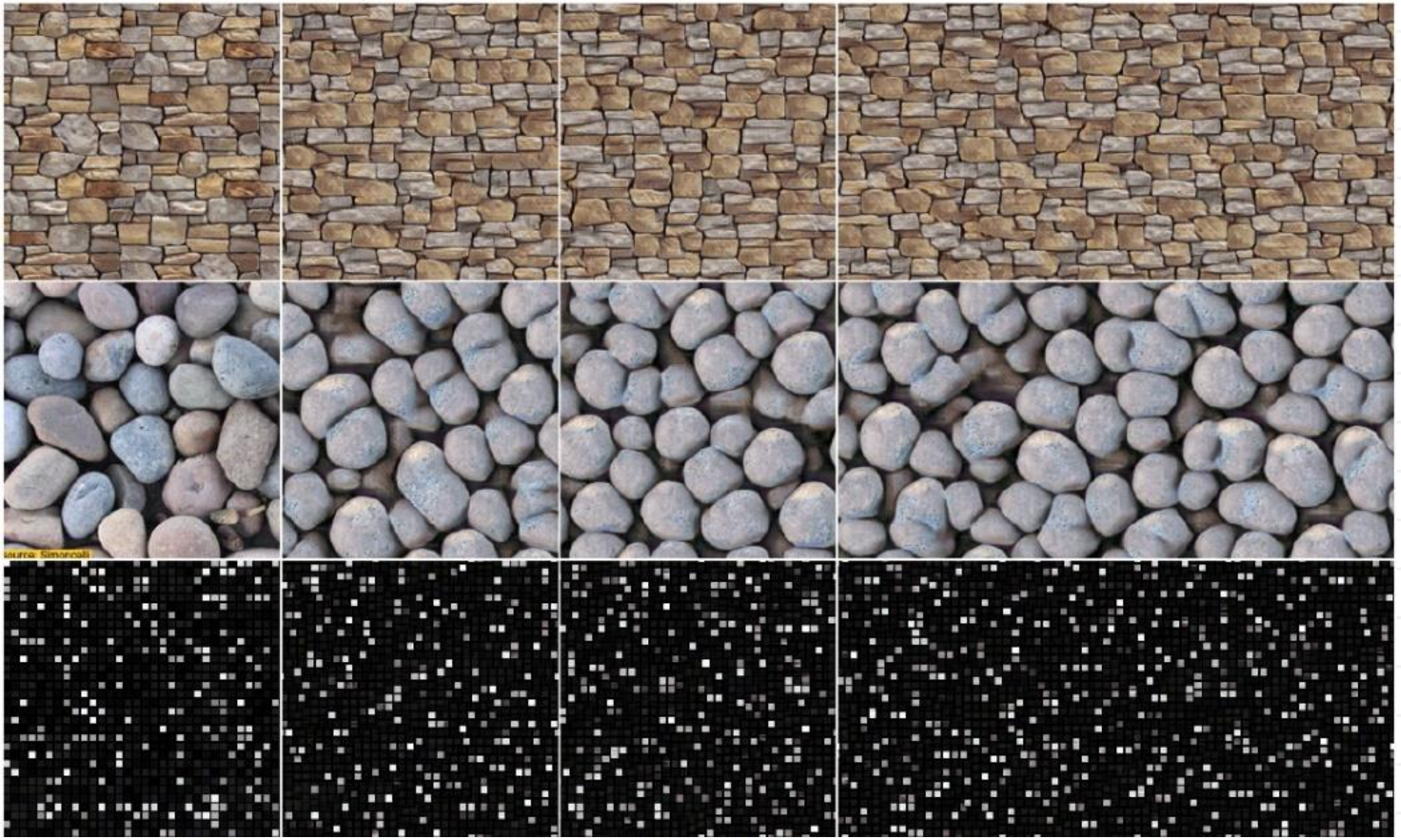
Generator networks details



- Multi-scale approach simplifies learning
- Looks scary, but still feedforward and fully convolutional

[Ulyanov et al. ICML15]

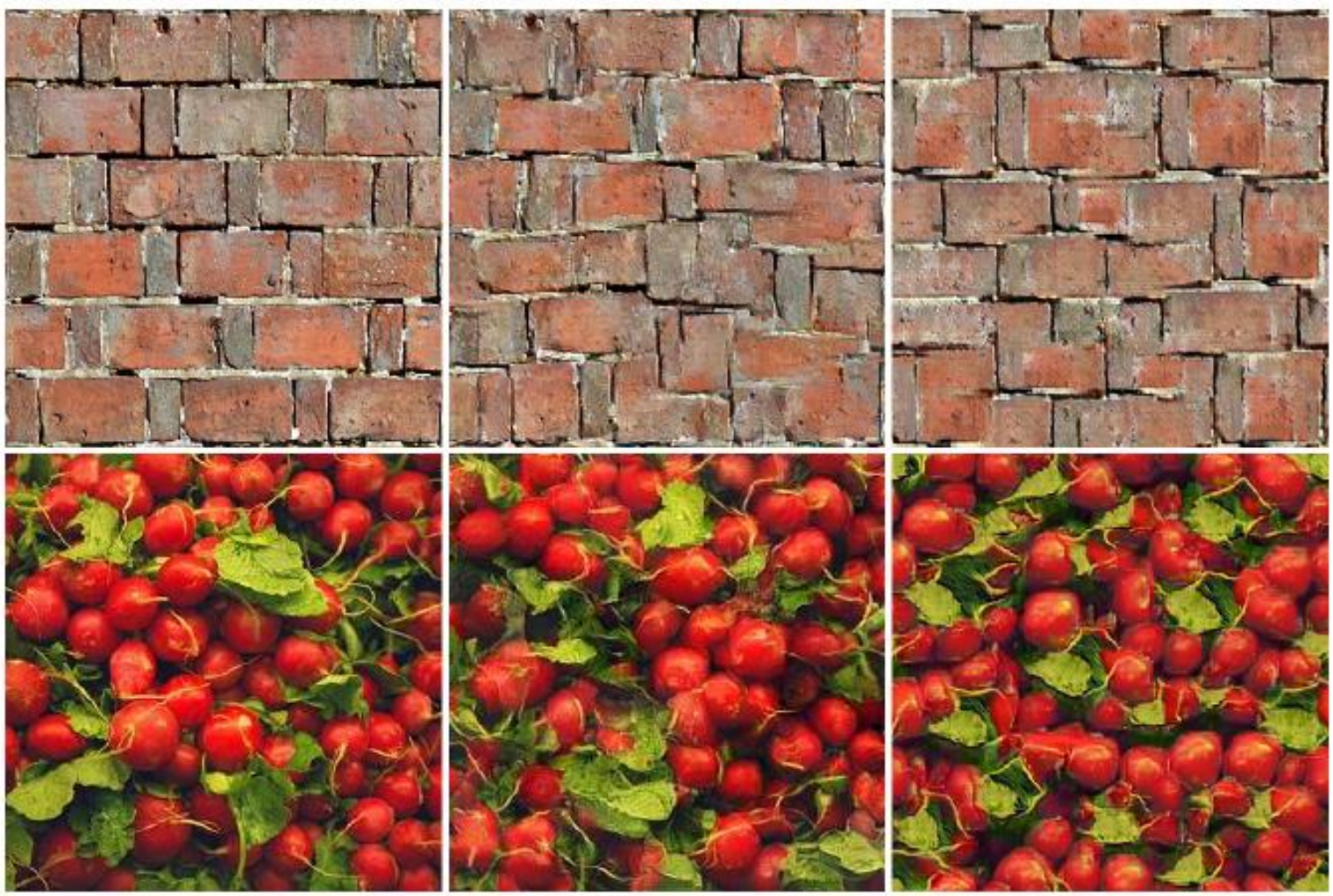
Feed-forward texture synthesis



sample

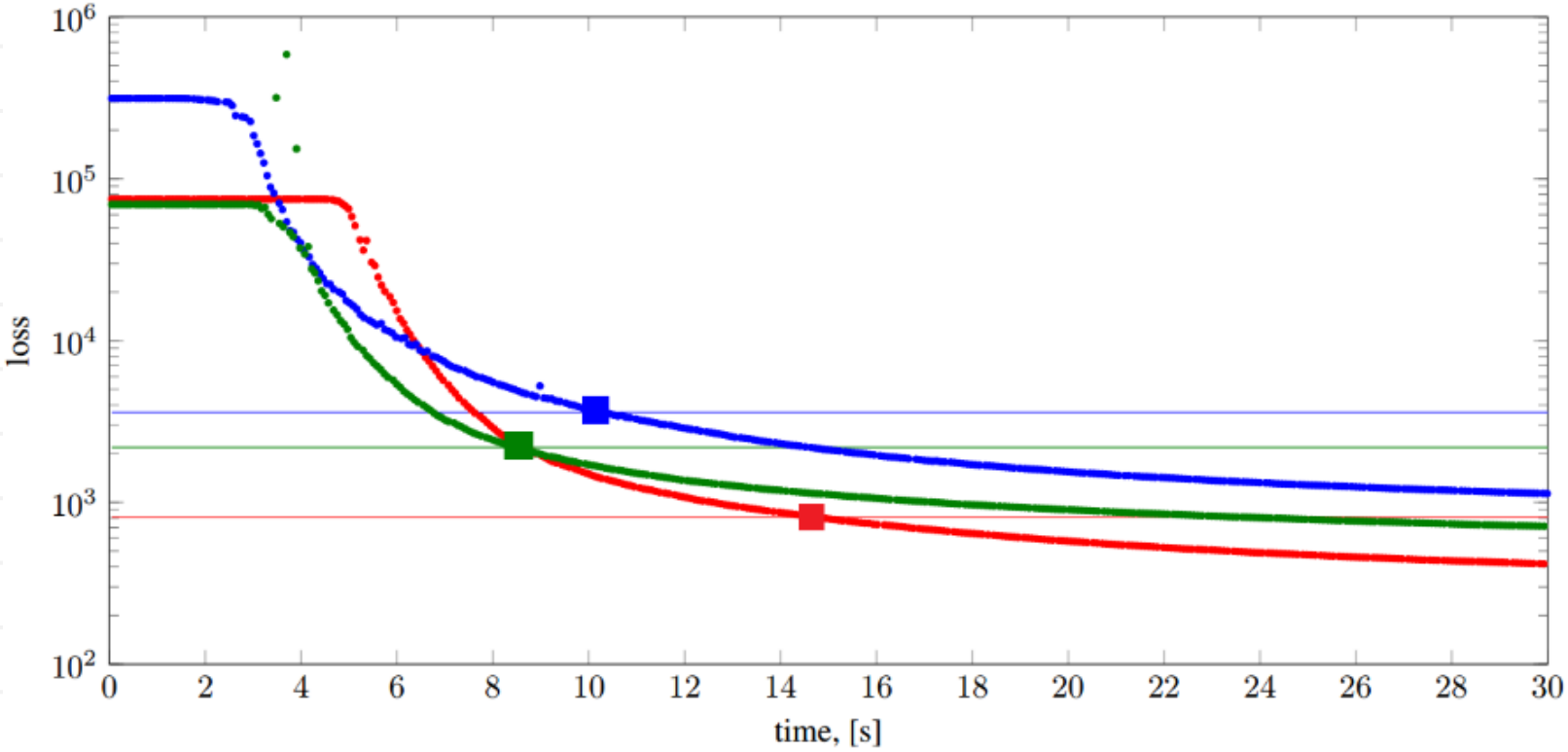
Results

Pre-image vs. feed-forward



sample Pre-image Feed-forward

Feed-forward ConvNet as a "pseudooptimizer"



- Horizontal lines: average sample loss
  - **0.06 second** to generate sample
- Dotted lines: *optimization-based* loss as a function of time
  - **10 seconds** to achieve the same loss values

Feed-forward stylization

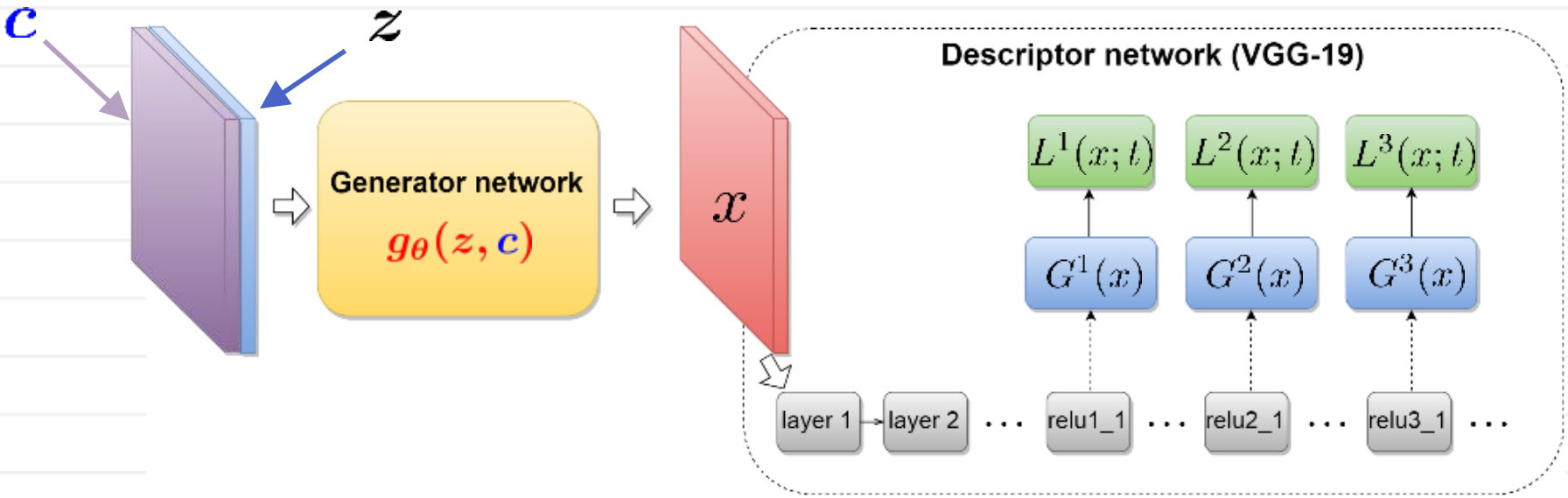


Image generation:

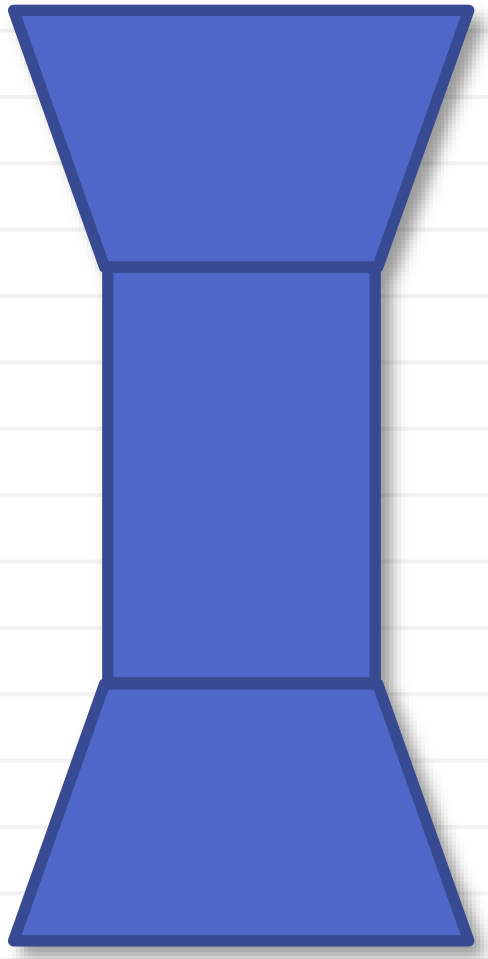
$$x = g_\theta(\boldsymbol{z}, \boldsymbol{c}), \quad \boldsymbol{z} \sim U(0,1)$$

Optimization task:

$$\min_\theta \mathbb{E}\mathcal{L}(g_\theta(\boldsymbol{z}, \boldsymbol{c}); c, t), \quad \boldsymbol{z} \sim U(0,1)$$

Good architectures for generative networks

Layer	Activation size
Input	$3 \times 256 \times 256$
Reflection Padding (40×40)	$3 \times 336 \times 336$
$32 \times 9 \times 9$ conv, stride 1	$32 \times 336 \times 336$
$64 \times 3 \times 3$ conv, stride 2	$64 \times 168 \times 168$
$128 \times 3 \times 3$ conv, stride 2	$128 \times 84 \times 84$
Residual block, 128 filters	$128 \times 80 \times 80$
Residual block, 128 filters	$128 \times 76 \times 76$
Residual block, 128 filters	$128 \times 72 \times 72$
Residual block, 128 filters	$128 \times 68 \times 68$
Residual block, 128 filters	$128 \times 64 \times 64$
$64 \times 3 \times 3$ conv, stride 1/2	$64 \times 128 \times 128$
$32 \times 3 \times 3$ conv, stride 1/2	$32 \times 256 \times 256$
$3 \times 9 \times 9$ conv, stride 1	$3 \times 256 \times 256$



1. Downsampling part
2. Residual blocks (usually with padding)
3. Upsampling block

[Johnson et al. CVPR16]

Instance normalization

$$y_{nijk} = \frac{x_{nijk} - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}},$$

$$\mu_i = \frac{1}{HWN} \sum_{n=1}^{N} \sum_{l=1}^{W} \sum_{m=1}^{H} x_{nilm},$$

$$\sigma_i^2 = \frac{1}{HWN} \sum_{n=1}^{N} \sum_{l=1}^{W} \sum_{m=1}^{H} (x_{nilm} - \mu_i)^2$$

Batch norm (train time)

$$y_{nijk} = \frac{x_{nijk} - \mu_{ni}}{\sqrt{\sigma_{ni}^2 + \epsilon}},$$

$$\mu_{ni} = \frac{1}{HW} \sum_{l=1}^{W} \sum_{m=1}^{H} x_{nilm},$$

$$\sigma_{ni}^2 = \frac{1}{HW} \sum_{l=1}^{W} \sum_{m=1}^{H} (x_{nilm} - \mu_{ni})^2$$

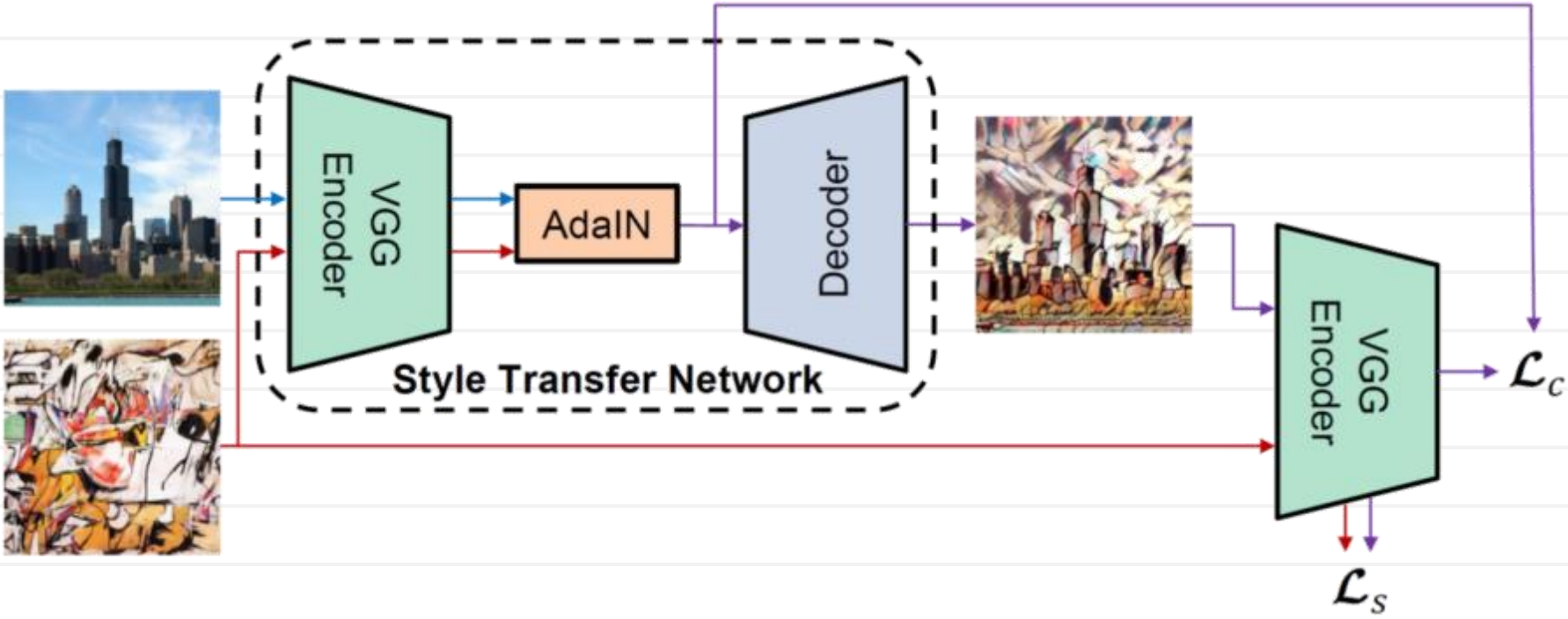
Instance norm

[Ulyanov et al. CVPR17]

Feed-forward stylization

[Ulyanov et al. CVPR17]

Fully feed-forward stylization

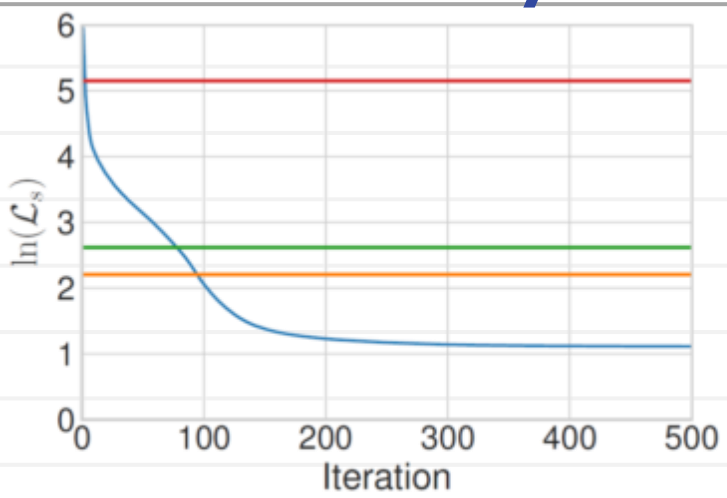


$$\text{AdaIN}(x, y) = \sigma(y)\left(\frac{x - \mu(x)}{\sigma(x)}\right) + \mu(y)$$

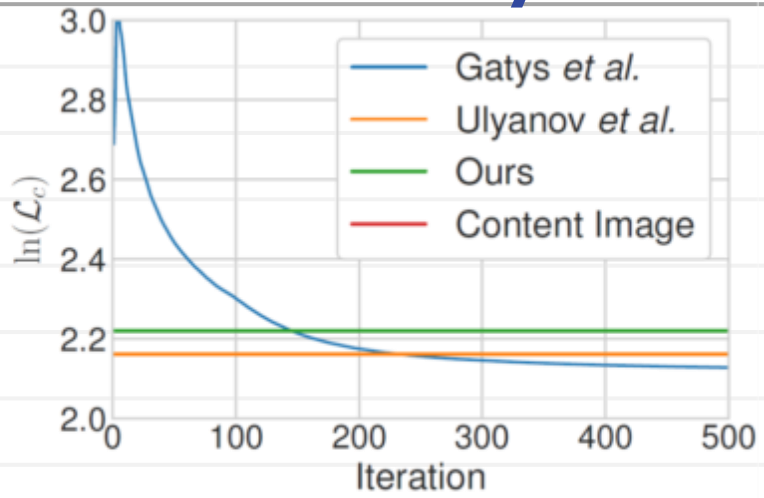
- Style image affects the result through a vector of params (a useful pattern)

[Huang and Belongie, ICCV17]

Fully feed-forward stylization



(a) Style Loss



(b) Content Loss

Style | Content | Ours | Chen and Schmidt | Ulyanov *et al.* | Gatys *et al.*

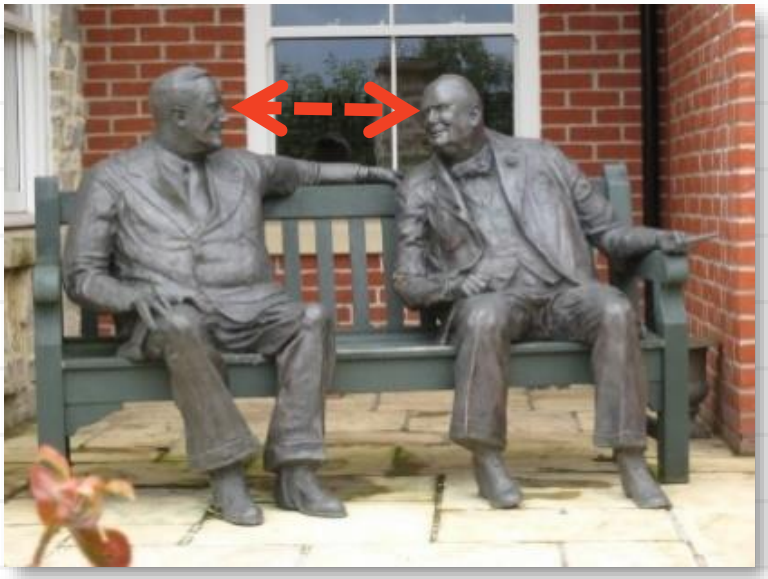
[Huang and Belongie, ICCV17]

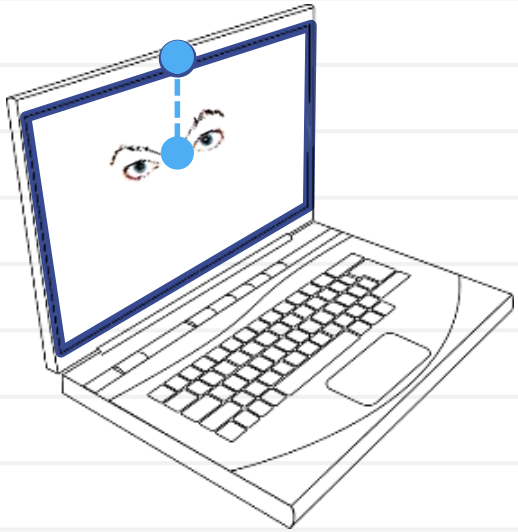
Spatial alignment

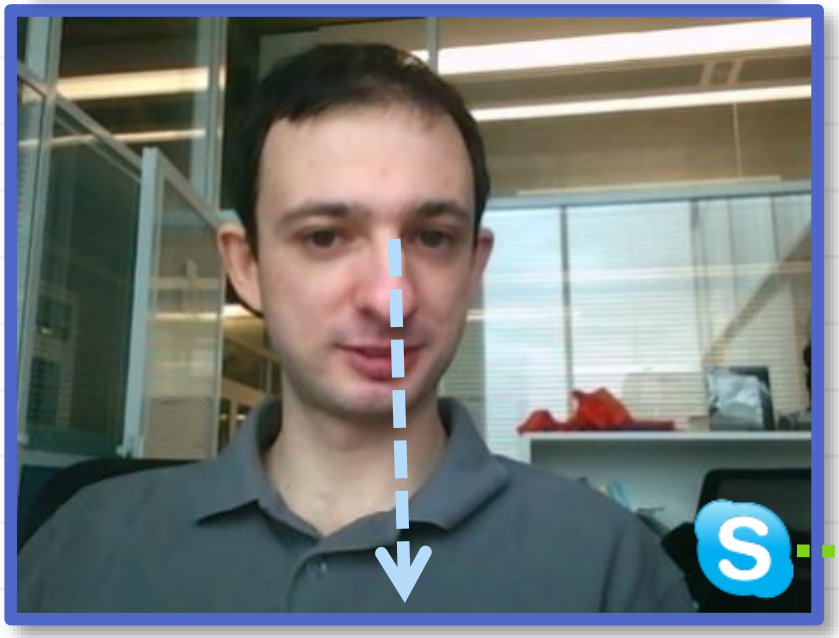



- Previous examples (superres, stylization, photorealistic synthesis): the input and the output are spatially aligned
- Many interesting/useful transforms require geometric deformations
- Such tasks can be solved by networks with warping layers

Problem: lack of eye contact





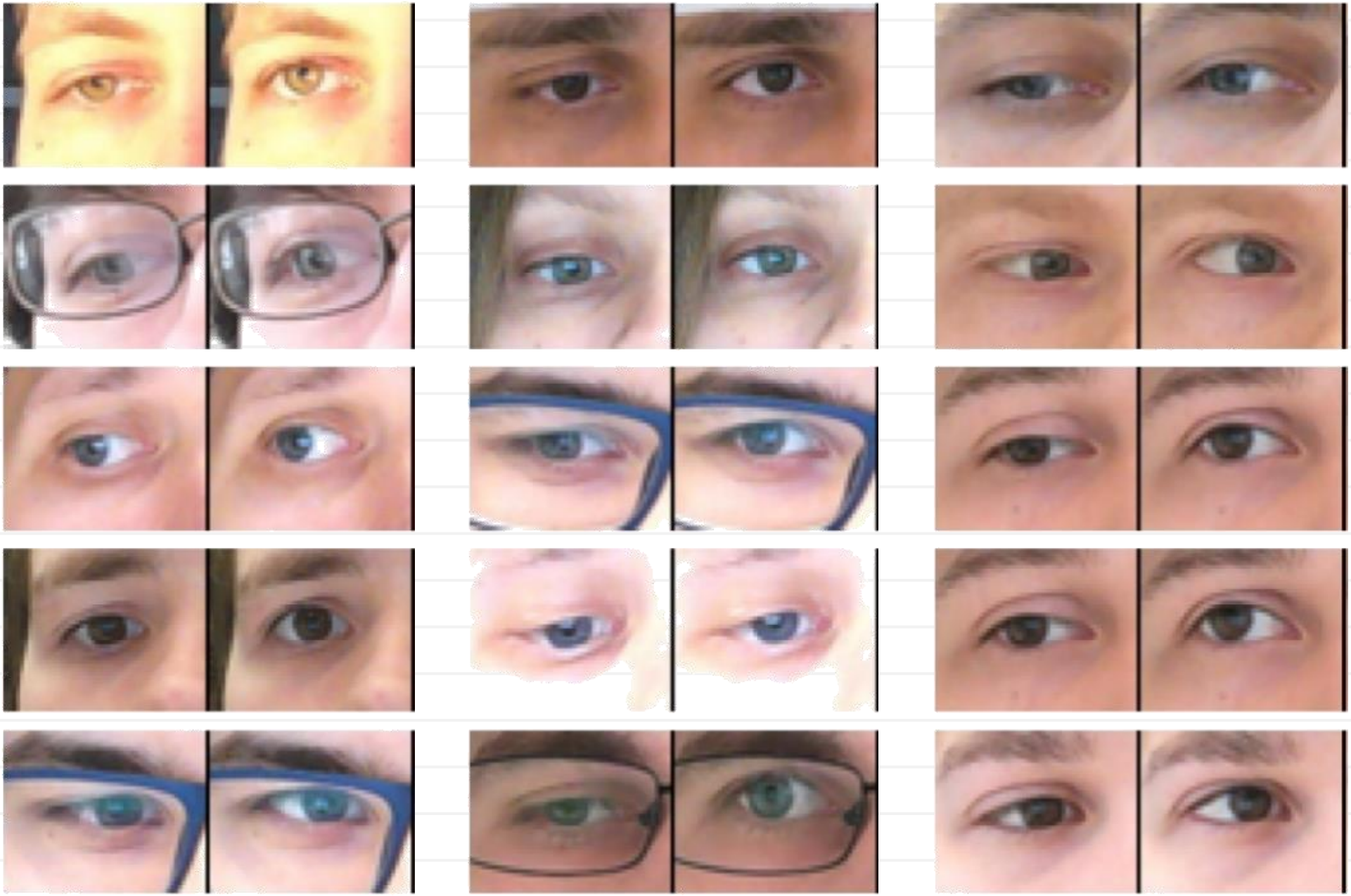


Supervised learning is hard

Goal: capture appearance changes from gaze redirection.

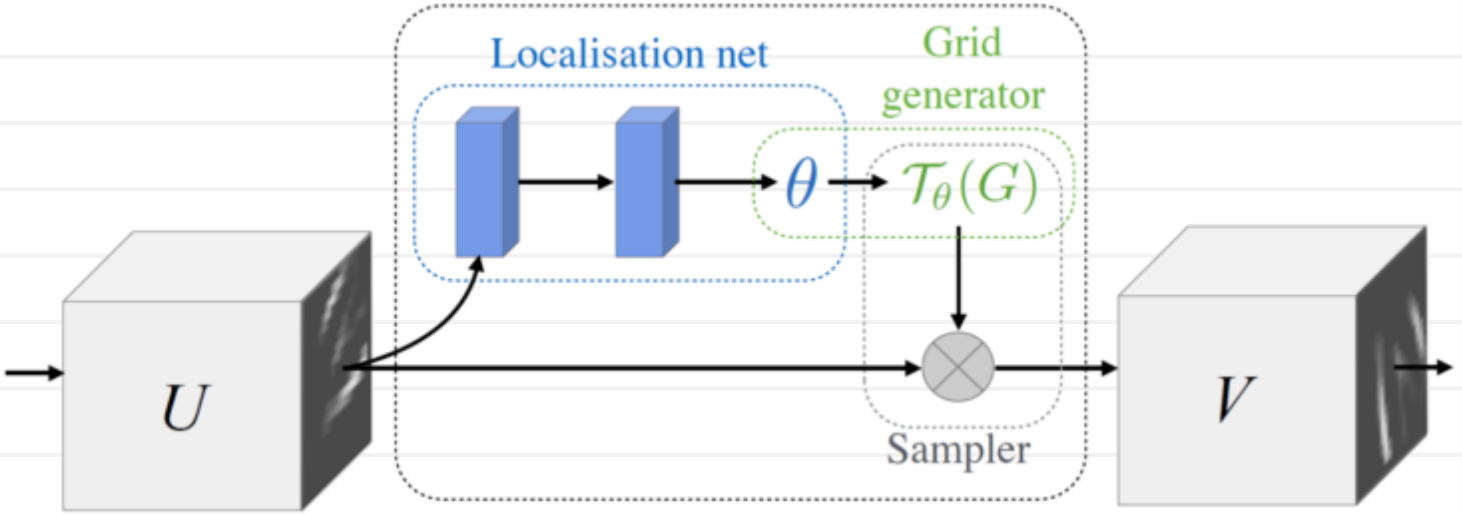
[Ganin et al. 2016]

Harnessing training examples

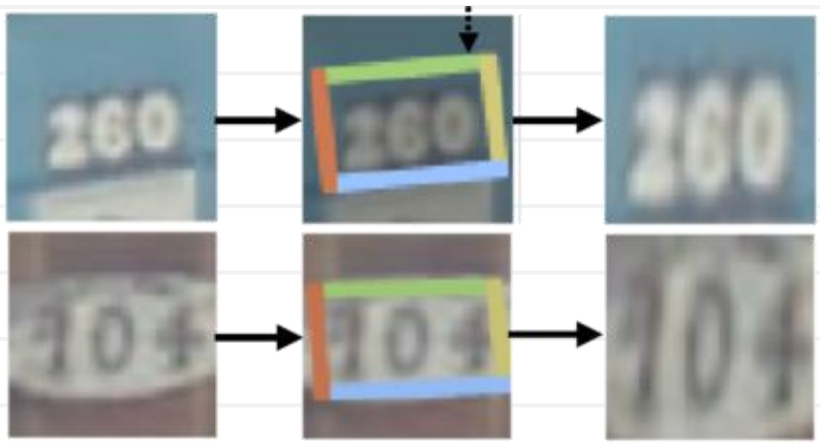


Spatial transformer networks

[Jadegberg et al. NIPS 2015]

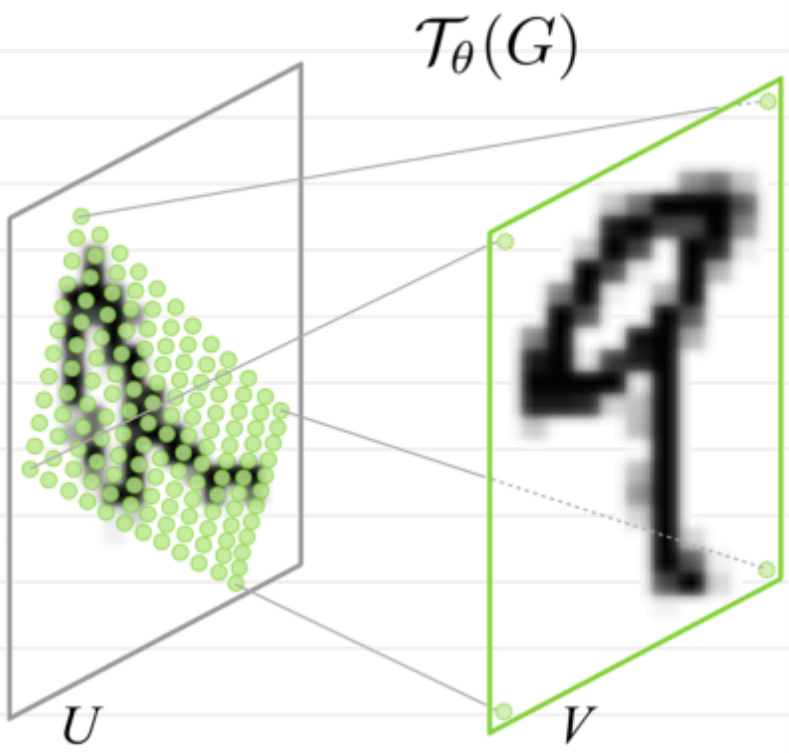


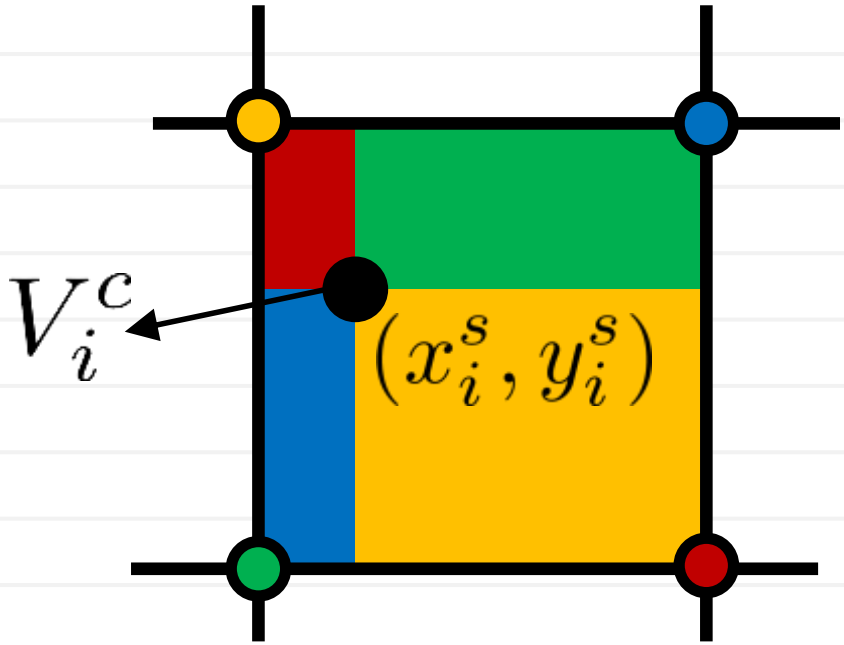
Spatial transformer in action:



Bilinear sampling layer

- Was introduced as part of *Spatial transformer*
- Very useful in generative networks





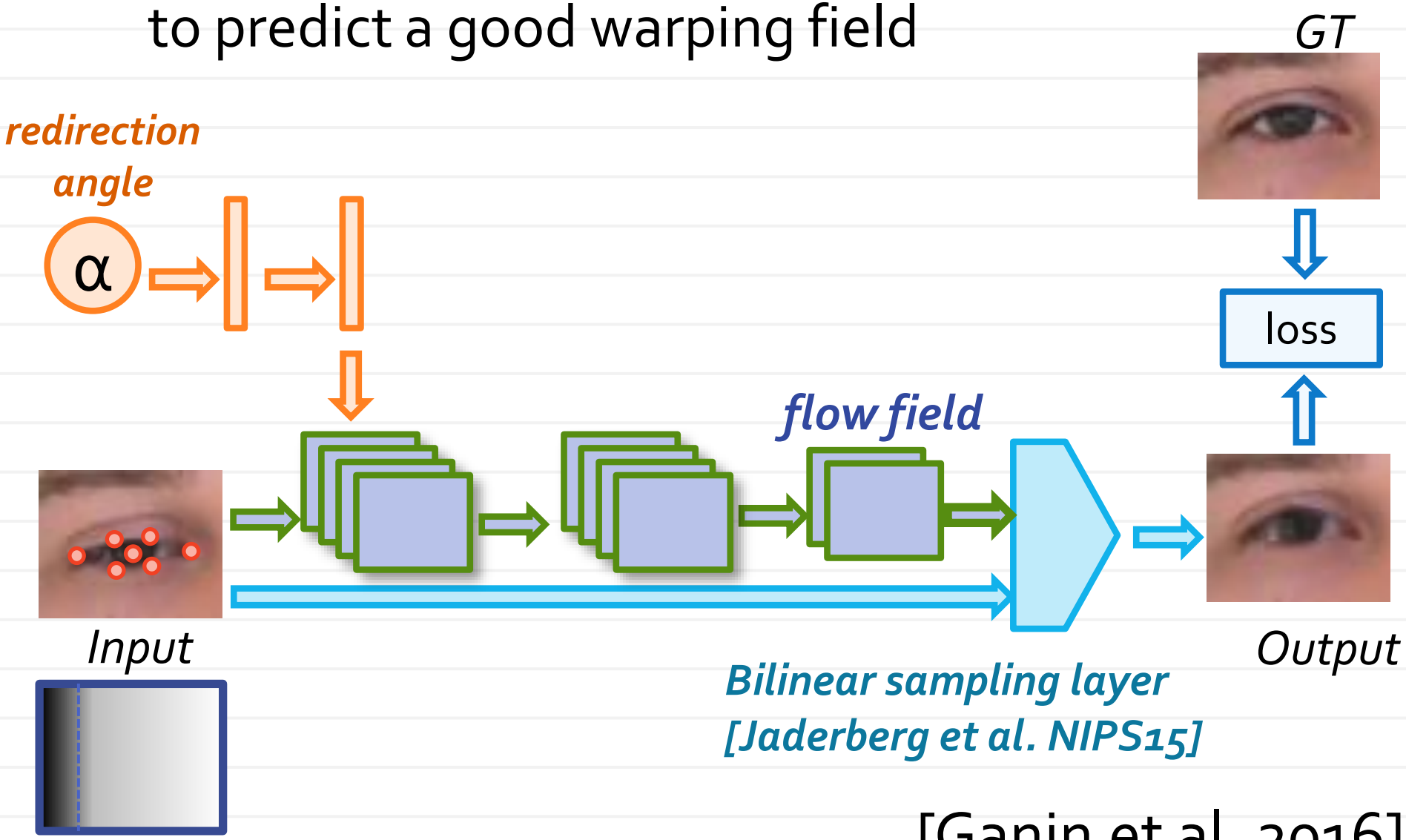
Piecewise-differentiable!

$$V_i^c = \sum_n^H \sum_m^W U_{nm}^c \max(0, 1 - |x_i^s - m|) \max(0, 1 - |y_i^s - n|)$$

[Jadegberg et al. NIPS 2015]

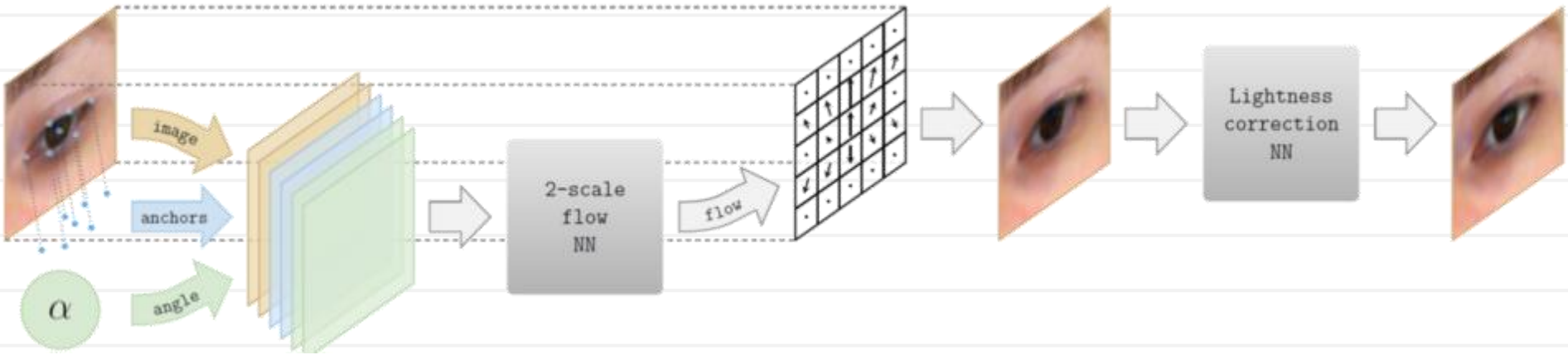
Deep warping

- High-level idea: make a network to predict a good warping field



[Ganin et al. 2016]

DeepWarp model overview



- All three layers can be trained "end-to-end"
- High-level idea: warping should be "coarse-to-fine" [Horn&Schunck 1981]

[Ganin et al. ECCV16]

Example results

15° up

15° down

[Ganin et al. 2016]

Results

Try yourself:

@MeduzaEyeBot on Telegram *(send any photo with large face, tick "Compress image" when sending)*

[Ganin et al. 2016]

Bibliography

J Kim, JK Lee, KM Lee, Accurate Image Super-Resolution Using Very Deep Convolutional Networks, CVPR 2016

Alexey Dosovitskiy, Jost Tobias Springenberg, Thomas Brox:
Learning to generate chairs with convolutional neural networks. CVPR 2015: 1538-1546

Dmitry Ulyanov, Andrea Vedaldi, Victor S. Lempitsky:
Deep Image Prior. CVPR 2018

Justin Johnson, Alexandre Alahi, Li Fei-Fei:
Perceptual Losses for Real-Time Style Transfer and Super-Resolution. ECCV (2) 2016: 694-711

Qifeng Chen, Vladlen Koltun:
Photographic Image Synthesis with Cascaded Refinement Networks. ICCV 2017: 1520-1529

Leon A. Gatys, Alexander S. Ecker, Matthias Bethge:
Texture Synthesis Using Convolutional Neural Networks. NIPS 2015: 262-270

Bibliography

Leon A. Gatys, Alexander S. Ecker, Matthias Bethge:
Image Style Transfer Using Convolutional Neural Networks. CVPR 2016: 2414-2423

Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, Victor S. Lempitsky:
Texture Networks: Feed-forward Synthesis of Textures and Stylized Images. CoRR abs/1603.03417 (2016)

Xun Huang, Serge J. Belongie:
Arbitrary Style Transfer in Real-Time with Adaptive Instance Normalization. ICCV 2017: 1510-1519

Dmitry Ulyanov, Andrea Vedaldi, Victor S. Lempitsky:
Deep Image Prior. CVPR 2018: 9446-9454

Dmitry Ulyanov, Andrea Vedaldi, Victor S. Lempitsky:
Improved Texture Networks: Maximizing Quality and Diversity in Feed-Forward Stylization and Texture Synthesis. CVPR 2017: 4105-4113

Yaroslav Ganin, Daniil Kononenko, Diana Sungatullina, Victor S. Lempitsky:
DeepWarp: Photorealistic Image Resynthesis for Gaze Manipulation. ECCV (2) 2016: 311-326