Style Transfer: a brief view or... telling a good story

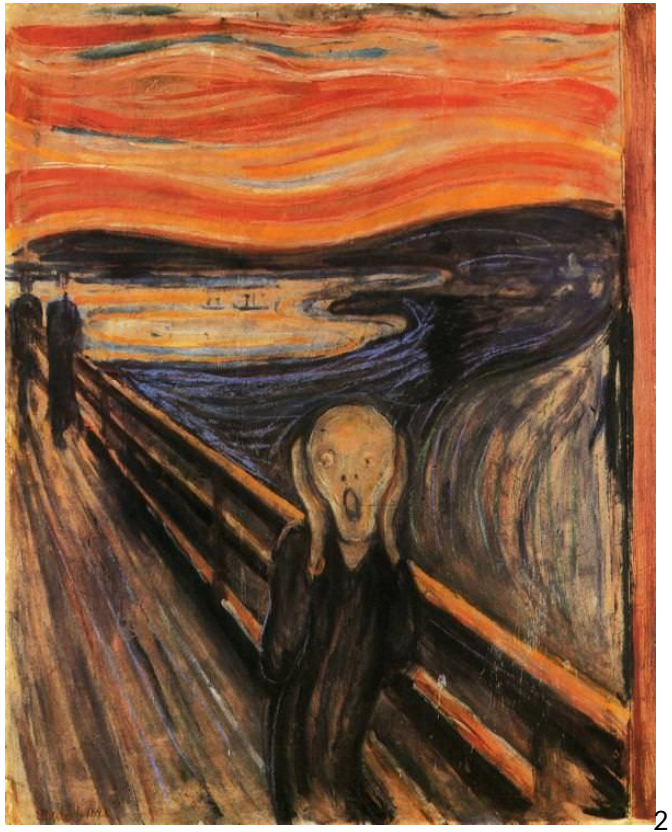
Victor Almeida
Fundamentals and trends in Vision and Image Processing, course at IMPA 2017
Professor: Luiz Velho

CC BY

Main goals of this presentation

1) Understand and explain the problem
2) How this topic research evolved
3) Examples: completed and intuitive.
4) Last but not least: discussion, opinions, directions, questions etc

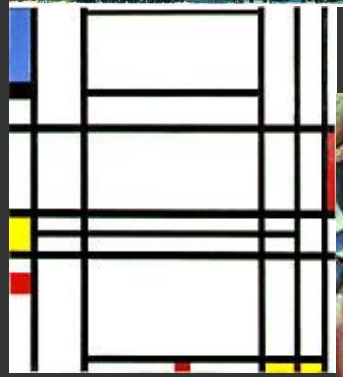
Just a remind: We're following, in some way, Gatys, L. A., Ecker, A. S., & Bethge, M. (2015). A Neural Algorithm of Artistic Style, 3–7. http://doi.org/10.1167/16.12.326

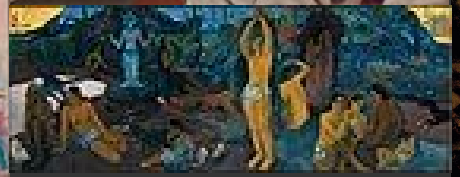


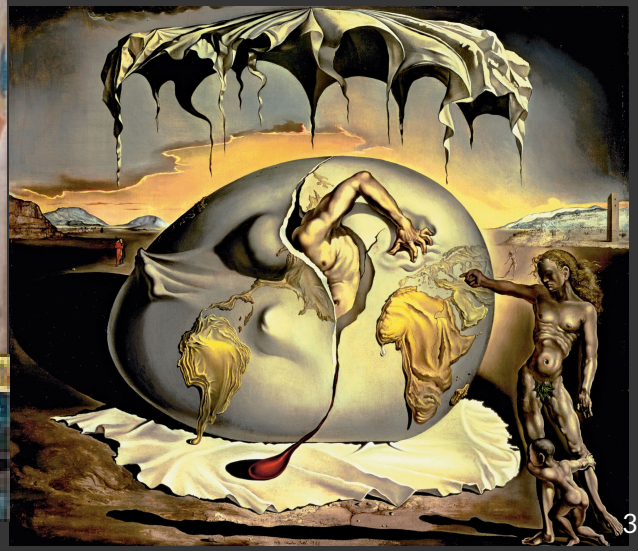
Edvard Munch, The Scream

– What is style?

How to quantify?







1. Evolution of the problem

"Classical" approach

- ➔ **Efros and Freeman, 2001: correspondence map that includes features of the target image to constrain the texture synthesis procedure.**
- ➔ **Hertzman et al., 2001: use image analogies to transfer the texture from an already stylised image onto a target image.**
- ➔ **Ashikhmin, 2001: focuses on transferring the high-frequency texture information while preserving the coarse scale of the target image.**
- ➔ **Lee et al., 2010: improve this algorithm by additionally informing the texture transfer with edge orientation information.**

Kyprianidis et al, 2013: "taxonomy of artistic stylization"

A review covering 2000's

Classical: pre-2000

Stroke-based Rendering (SBR)

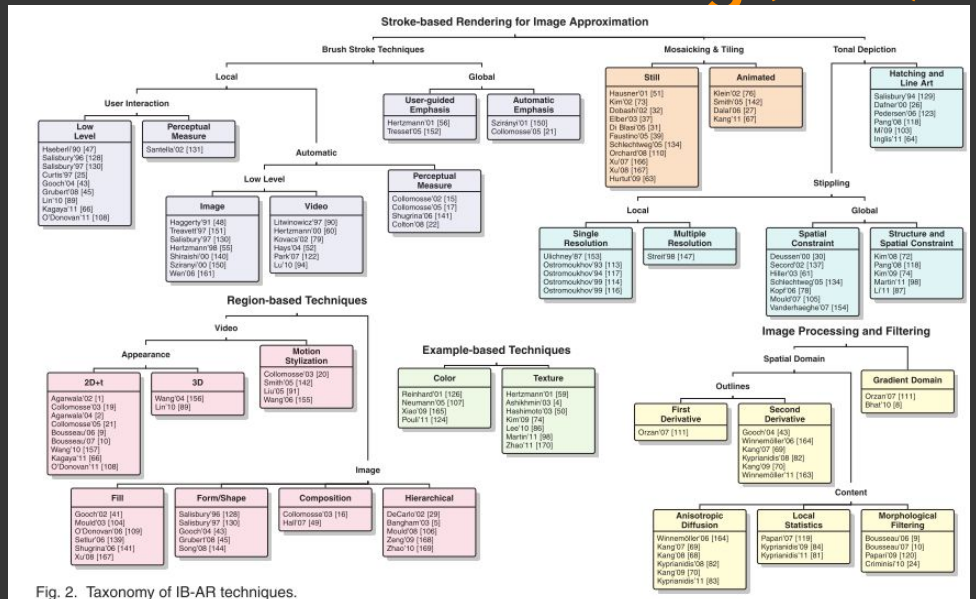


Fig. 2. Taxonomy of IB-AR techniques.

1) **Algorithmic Aesthetics.** How can we quantify success and how can aesthetic 'beauty' be defined?
2) **Abstraction.** How do you capture the 'essence' of an object, image, or painterly style? The reduction of an image to a sketch or structure was discussed.
3) **Visualization.** How can you use NPR to communicate information most effectively?
4) **Interactivity.** How can you develop automated tools that support creativity and the artist?
5) **Artistic Turing Test.** Can you make artistic computer images and animations that are indistinguishable from those created by hand?
6) **New Art Forms.** Can NPR be used to create some entirely new and original forms of art?
7) **Naming the Field.** In a closing aside, Salesin voiced the views of many that NPR is an unfortunately broad term. A decade later, nevertheless, it is still the dominant label for the field [40]. Artistic

– Kyprianidis et al, 2013

A review covering 2000's

Collomosse, 2003

1) **Algorithmic Aesthetics.** How can we quantify success and how can aesthetic 'beauty' be defined?
2) **Abstraction.** How do you capture the 'essence' of an object, image, or painterly style? The reduction of an image to a sketch or structure was discussed.
3) **Visualization.** How can you use NPR to communicate information most effectively?
4) **Interactivity.** How can you develop automated tools that support creativity and the artist?
5) **Artistic Turing Test.** Can you make artistic computer images and animations that are indistinguishable from those created by hand?
6) **New Art Forms.** Can NPR be used to create some entirely new and original forms of art?
7) **Naming the Field.** In a closing aside, Salesin voiced the views of many that NPR is an unfortunately broad term. A decade later, nevertheless, it is still the dominant label for the field [40]. Artistic

– Ashikhmin, 2001:

synthesis of natural textures

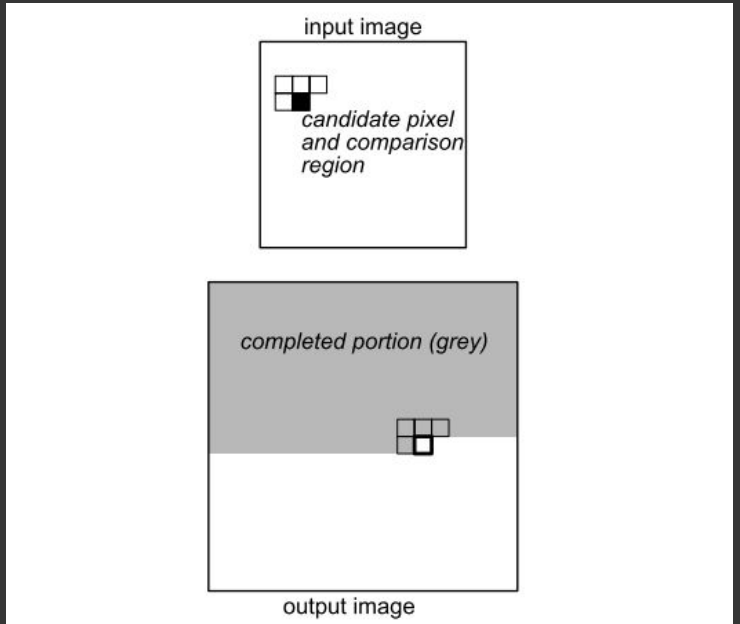


Figure 2: *The Wei and Levoy (WL) texture synthesis process. Pixels are generated in scanline order. The value of a pixel is determined by choosing the best pixel in a candidate list. For WL this list includes all pixels in the input image. Best pixel is the one whose L-shaped neighborhood most closely resembles the neighborhood of the pixel currently being synthesized. For clarity, a smaller than usual (only 3x3) neighborhood is shown on all figures.*

Wei and Levoy algorithm, 2000

- The output image is initialized to random noise with a histogram equal that of the input image.
- For each pixel in the output image, in scanline order, do:
  - in the output image, an L-shaped neighborhood of current pixel of a specific (fixed) size is considered, see Figure 2.
  - a search is performed in the input sample for a pixel with a neighborhood most similar to the one identified in the previous step.
  - the current pixel value in the output image is copied from the position in the input sample identified as the most similar by this search.

$$D(N_1, N_2) = \sum_{p\ in N} \{(R_1(p) - R_2(p))^2 + (G_1(p) - G_2(p))^2 + (B_1(p) - B_2(p))^2\}.$$

– Ashikhmin, 2001:

synthesis of natural textures

What can speed up the algorithm?

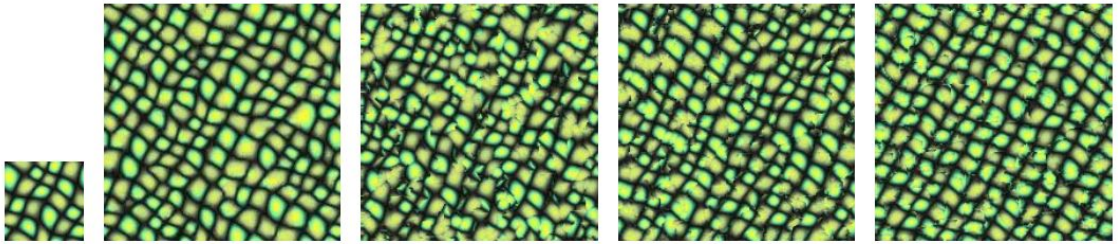


Figure 8: *Effect of the neighborhood size for a smooth texture. Input texture size is 64x64 pixels, all results are 192x192. Left to right: input sample, WL result, and our results with 5x5, 9x9 and 21x21 neighborhoods.*

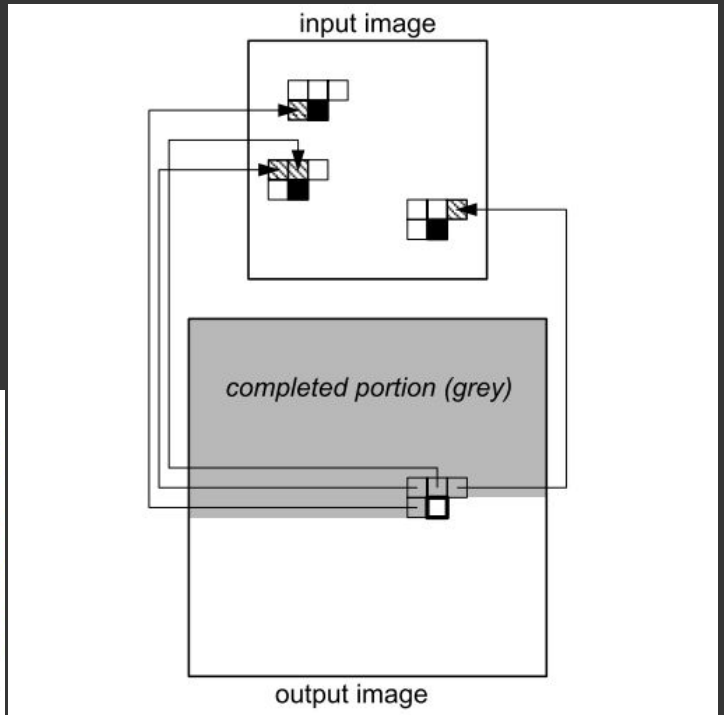
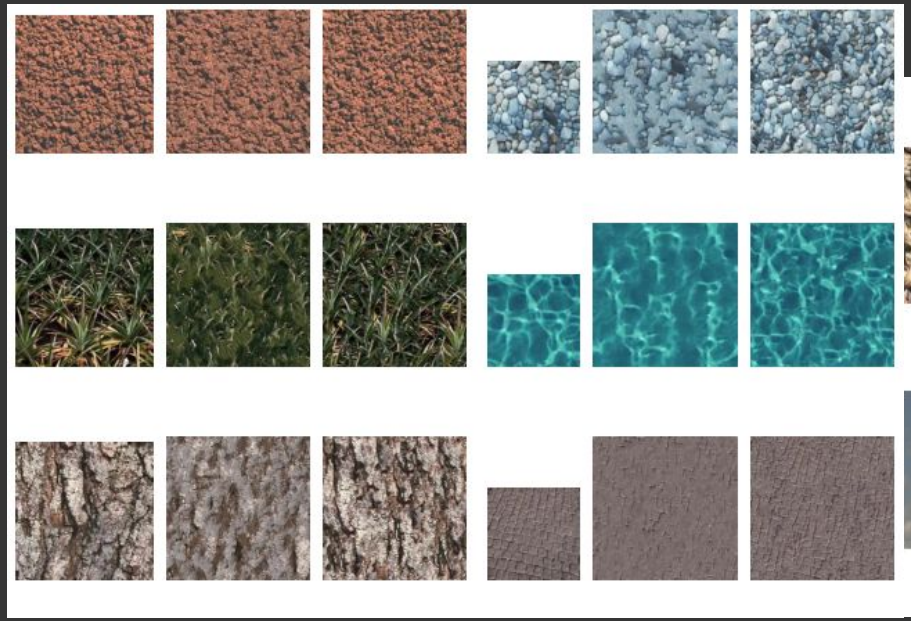


Figure 4: *Candidate pixels for our algorithm. Each pixel in the current L-shaped neighborhood generates a "shifted" candidate pixel (black) according to its original position (hatched) in the input texture. The best pixel is chosen among these candidates only. Several different pixels in the current neighborhood can generate the same candidate.*

– Ashikhmin, 2001:

synthesis of natural textures

What can speed up the algorithm?



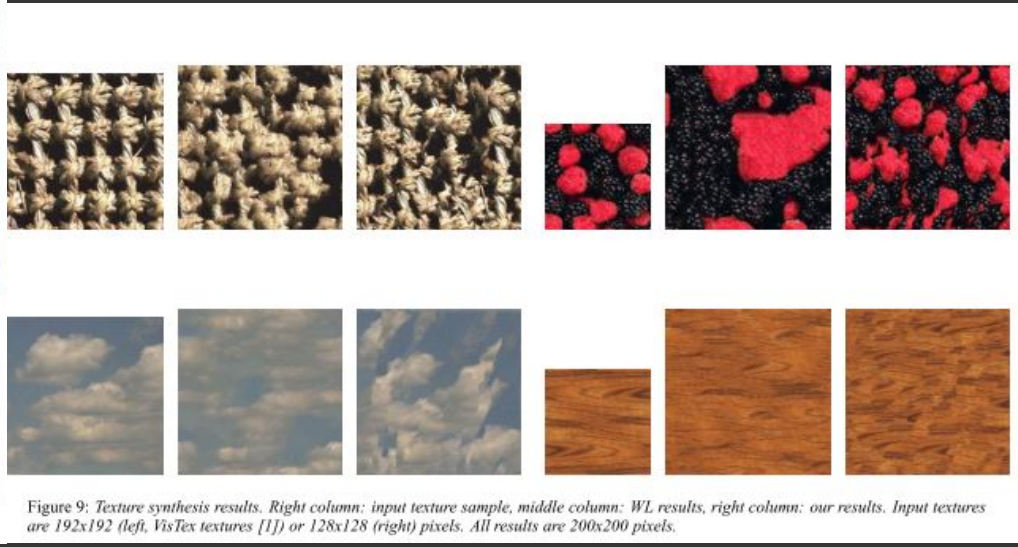
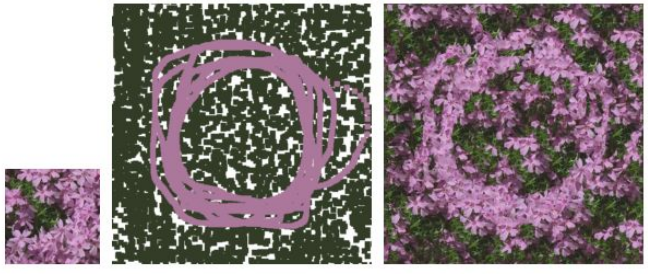


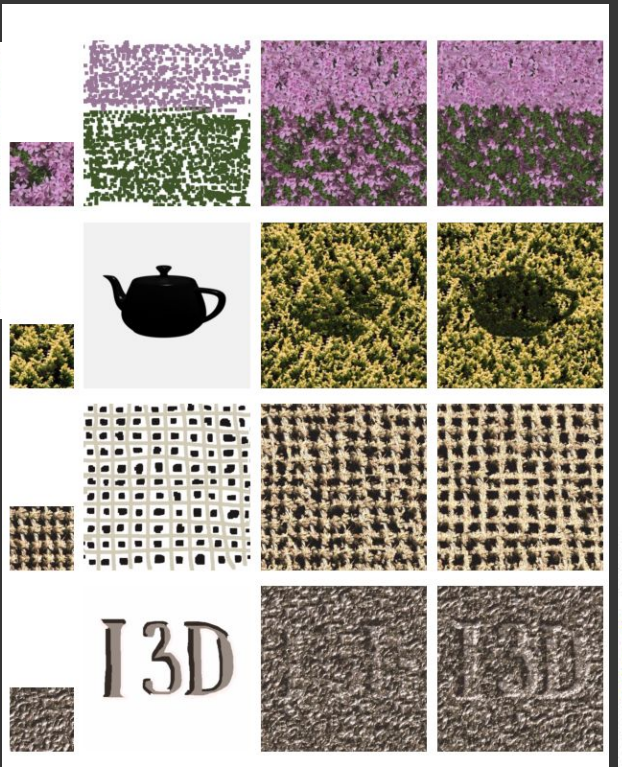
Figure 9: *Texture synthesis results. Right column: input texture sample, middle column: WL results, right column: our results. Input textures are 192x192 (left, VisTex textures [1]) or 128x128 (right) pixels. All results are 200x200 pixels.*

– Ashikhmin, 2001:

synthesis of natural textures

User “controlled” algorithm





“Most texture synthesis algorithms attempt to deal with as large class of textures as possible. This generality is certainly desirable but we are not hopeful about the discovery of such an ideal algorithm.”

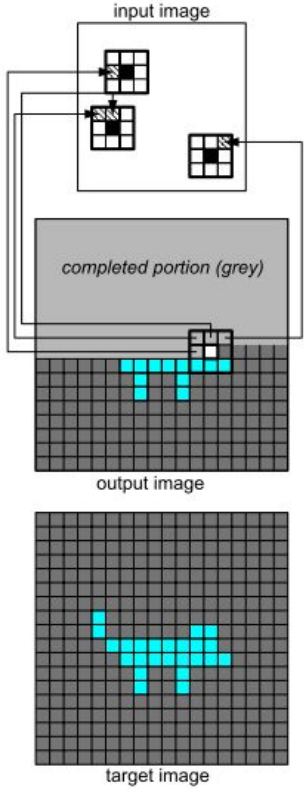


Figure 7: *User controlled version of the algorithm (first pass). Candidates are chosen as before but a square neighborhood in the input is now used to choose among them. This square of pixels is compared with a special neighborhood, top L-shaped half of the which is taken from the output image while the bottom half is from the target image (with only pixel which received user input being considered). The target image can be thought of as is being overwritten as the synthesis progresses. For additional passes the procedure is the same but candidates are created based on the full square neighborhood*

–Hertzman et al, 2001

Image analogies: simple, but powerful

-Filtering, style (artistic filter),texture transfer, super-resolution…

-Better than previou NPR, not restricted to one style.
-Each pixel one feature vector (RGB, luminance, filters)
→ Multiscale of input constructed;

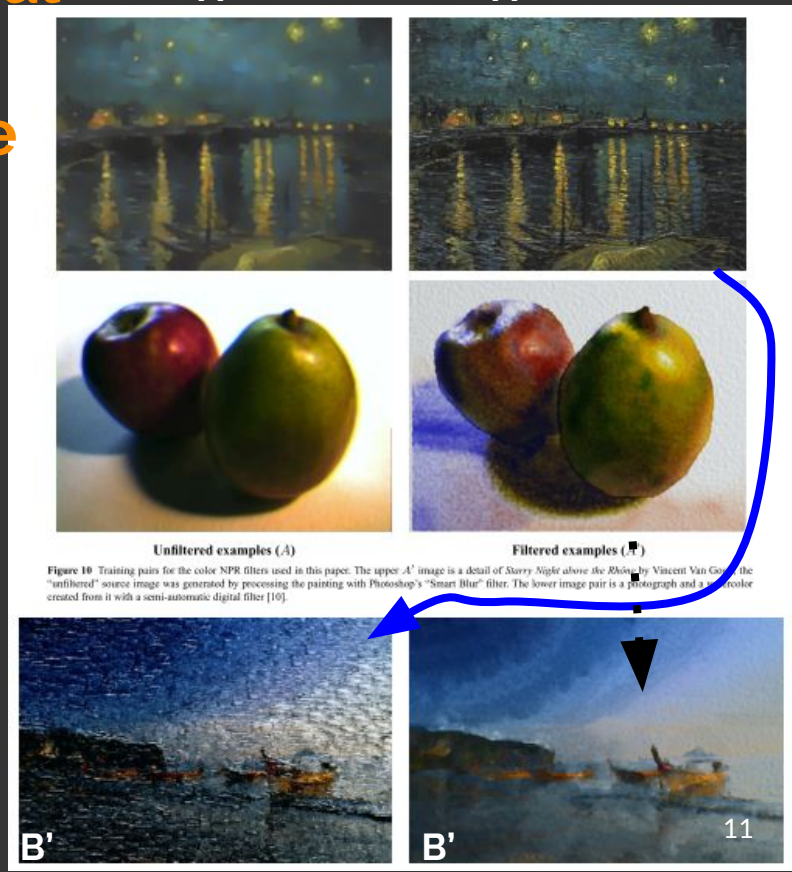


Figure 10 Training pairs for the color NPR filters used in this paper. The upper A' image is a detail of *Starry Night above the Rhône* by Vincent Van Gogh. The "unfiltered" source image was generated by processing the painting with Photoshop's "Smart Blur" filter. The lower image pair is a photograph and a watercolor created from it with a semi-automatic digital filter [10].

Hertzman et al, 2001

Image analogies

function CREATEIMAGEANALOGY(A, A', B):
  Compute Gaussian pyramids for A, A', and B
  Compute features for A, A', and B
  Initialize the search structures (e.g., for ANN)
  for each level ℓ, from coarsest to finest, **do**:
    for each pixel $q \in B'_\ell$, in scan-line order, **do**:
      $p \leftarrow$ BESTMATCH(A, A', B, B', s, ℓ, q)
      $B'_\ell(q) \leftarrow A'_\ell(p)$
      $s_\ell(q) \leftarrow p$
  return B'_L

function BESTMATCH(A, A', B, B', s, ℓ, q):
  $p_{app} \leftarrow$ BESTAPPROXIMATEMATCH(A, A', B, B', ℓ, q)
  $p_{coh} \leftarrow$ BESTCOHERENCEMATCH(A, A', B, B', s, ℓ, q)
  $d_{app} \leftarrow \|F_\ell(p_{app}) - F_\ell(q)\|^2$
  $d_{coh} \leftarrow \|F_\ell(p_{coh}) - F_\ell(q)\|^2$
  if $d_{coh} \leq d_{app}(1 + 2^{\ell - L}\kappa)$ **then**
    return p_{coh}
  else
    return p_{app}

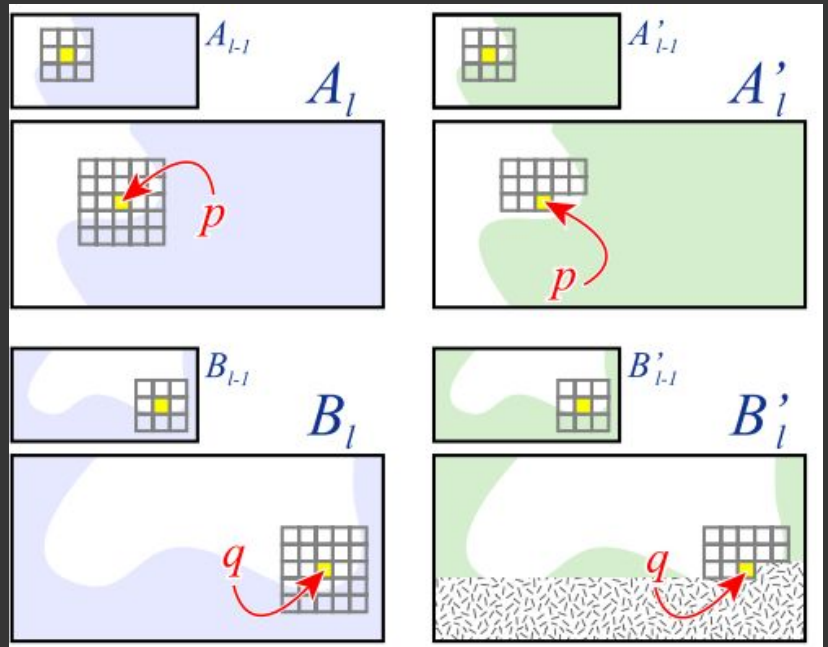


Figure 2 Neighborhood matching. In order to synthesize the pixel value at q in the filtered image B'_ℓ, we consider the set of pixels in B'_ℓ, B_ℓ, $B'_{\ell-1}$, and $B_{\ell-1}$ around q in the four images. We search for the pixel p in the A images that give the closest match. The synthesis proceeds in scan-line ordering in B'_ℓ.

–Hertzman et al, 2001

Image analogies

→ A first insight in Deep Learning?

- "In the design (or training) phase, a designer (possibly an expert) creates a filter by selecting the training images A and A' (for example, from scanned imagery), annotating the images if desired, and (directly or indirectly) selecting parameters that control how various types of image features will be weighted in the image analogy. The filter can then be stored away in a library. Later, in the application phase, a user (possibly someone with no expertise at all in creating image filters) applies the filter to some target image B."

→ Where is the learned filter?!

```
% TODO: Remove this -- it's just faster for testing.
A_scale = 0.5;
B_scale = 0.5;
A = imresize(A, A_scale);
A_prime = imresize(A_prime, A_scale);
B = imresize(B, B_scale);

% Make image analogy
B_prime = create_image_analogy(A, A_prime, B);
```

```
function [ B_prime ] = create_image_analogy( A, A_prime, B )
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here

global N_BIG;
global NUM_FEATURES;
global NNF;
```

Matlab:
https://github.com/jmecom/image-analogies

-Is classical methods dead?

No! Elad, M., & Milanfar, P. (2017). Style Transfer Via Texture Synthesis. IEEE Transactions on Image Processing, 26(5), 2338–2351.

Fig. 15. **Failure due to Poorly Chosen Style Image:** These two results are obtained by our algorithm, and in both the chosen style image is not rich enough to provide with a proper transfer.

Fig. 16. **Failure due to No-Segmentation:** These two results are obtained by our algorithm, and in both we applied no segmentation. As can be seen, the important content parts are treated badly and lost.

We start by defining the core objective of our algorithm: We are given a content image[1] $\underline{C} \in \mathbf{R}^{3N_c}$, and a style image $\underline{S} \in \mathbf{R}^{3N_s}$. These two images are accompanied by a segmentation mask $\mathbf{W} \in [0,\infty)^{N_c}$ that marks the importance of pixels in the content image, in terms of its parts to be preserved. More on this mask will be brought later in this Section. Our goal is the creation of the image[2] $\underline{X}$ that would minimize the following series of energy functionals:

$$E_{L,n}\{\underline{X}\} = \frac{1}{c}\sum_{(i,j)\in\Omega_{L,n}} \underset{(k,l)}{\text{Min}} \left\|\mathbf{R}_{ij}^{n}\underline{X} - \mathbf{Q}_{kl}^{n}\mathbf{D}_{L}^{S}\underline{S}\right\|_{2}^{r} \quad (1)$$
$$+ \|\mathbf{D}_{L}^{C}\underline{C} - \underline{X}\|_{\mathbf{W}}^{2} + \lambda\rho\{\underline{X}\}.$$

Objective: Create the style transfer image, $\underline{X}$.
Input: Use the following ingredients and parameters:

- $\underline{C}$ and $\underline{S}$ - Content and Style images
- $\mathbf{W}$ - Content segmentation map
- L_{max} - Number of resolution layers
- $n_1, n_2, \ \dots\ , n_m$ - patch sizes
- $d_1, d_2, \ \dots\ , d_m$ - Ω subsampling gaps
- I_{IRLS} - number of IRLS iterations (= 10)
- I_{alg} - number of update iterations per patch-size
- r - robust statistics value to use.

Initialization:

- Apply color transfer from $\underline{S}$ to $\underline{C}$
- Build the Gaussian pyramids of $\underline{C}$, $\underline{S}$, and $\mathbf{W}$
- Optional (for fast Approximate NN): Prepare the patches from the style image for each resolution layer and patch size as tree structure
- Initialize $\underline{X} = \mathbf{D}_{L}^{C}\underline{C} + V$, where $V \sim \mathcal{N}(0, 50)$

Loop Over Scales: For $L = L_{max}, \ \dots\ , 1$ do:
  Loop Over Patch-Sizes: For $n = n_1, \dots$, minimize $E_{L,n}$ w.r.t. $\underline{X}$
  Iterate: For $k = 1, 2 \ \dots\ , I_{alg}$ do:

1) *Patch Matching*: For the current image $\underline{X}$, match NN from the style image by solving (16).
2) *Robust Aggregation*: Compute $\tilde{\underline{X}}$ as the robust patch aggregation result, by solving (14) using I_{IRLS} iterations.
3) *Content Fusion*: Combine the content image $\mathbf{D}_{L}^{C}\underline{C}$ to $\tilde{\underline{X}}$ using Equation (15) to obtain the updated $\underline{X}$.
4) *Color Transfer*: Apply color-transfer from $\underline{S}$ to $\underline{X}$.
5) *Denoise*: Filter the obtained $\underline{X}$ by the domain-transform.

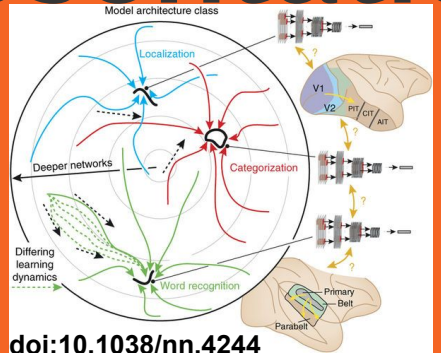
  Scale-Up: As we move to the next resolution layer, scale-up $\underline{X}$.
Result: The output of the above algorithm is $\underline{X}$.

Fig. 3. The overall Style Transfer Algorithm.

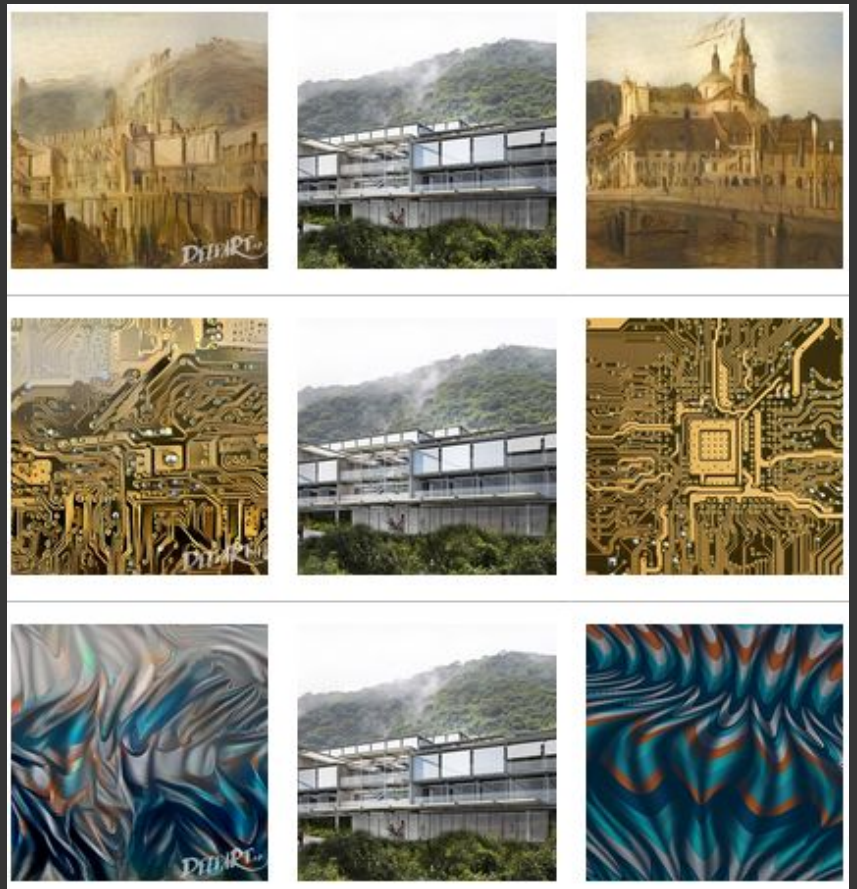
– Why should we use ANN?

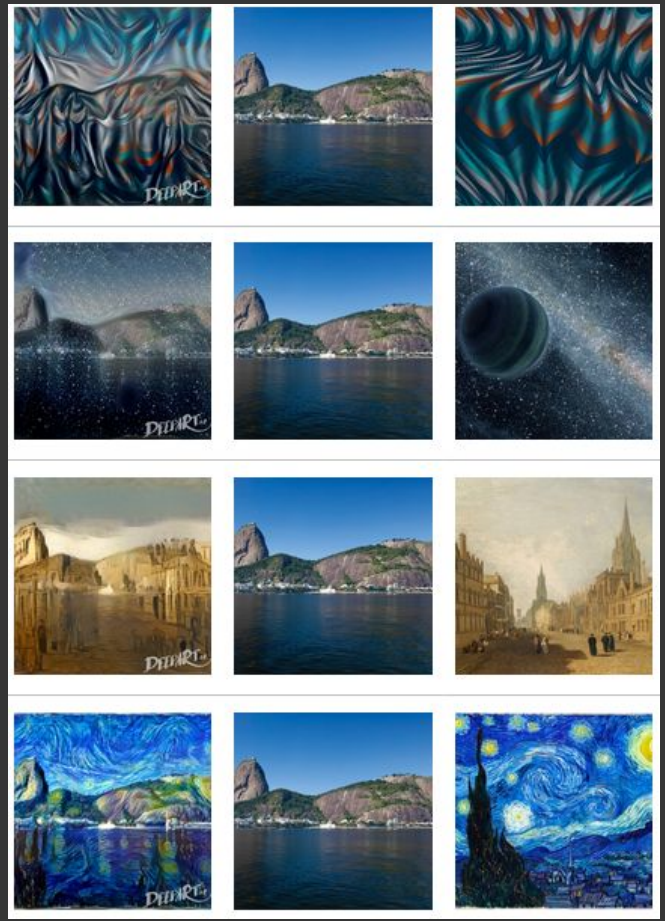
A motivation according to Gatys:

“Similarity between artificial NN and neural representation in the human brain.”



– Deepart.io (Gatys website)



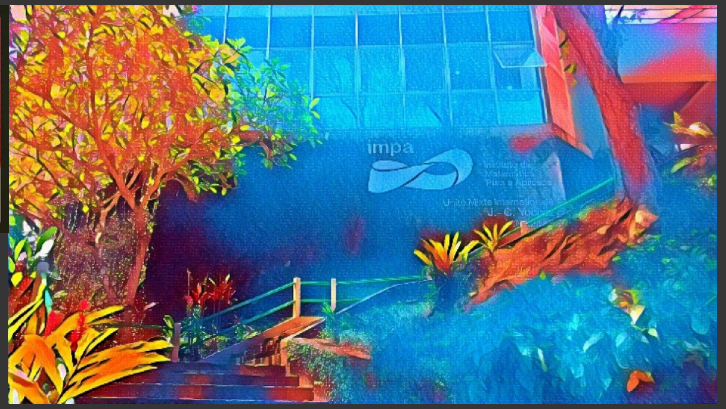


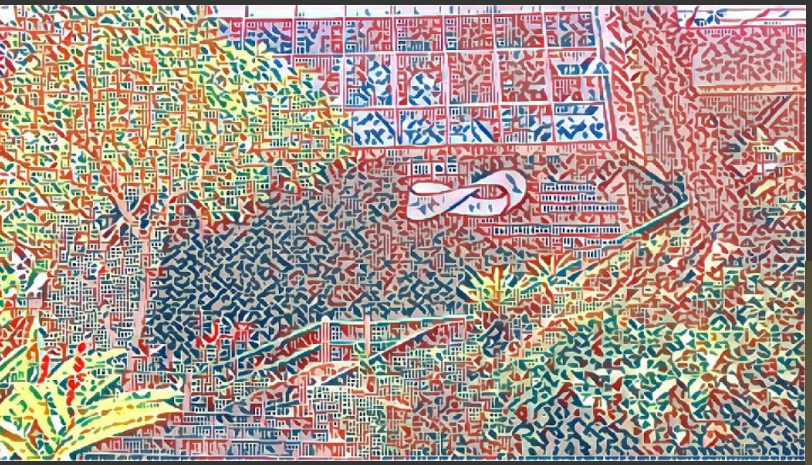
– Prisma App

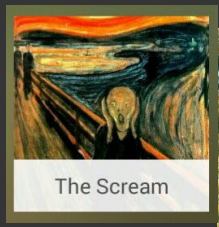
– Prisma App


impa
Instituto de Matemática Pura e Aplicada
Unité Mixte Internationale


Coloured


impa




The Scream

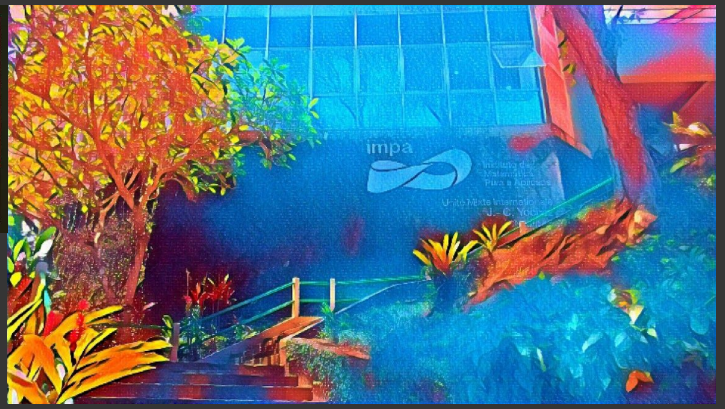

impa


DAILY
Toucan

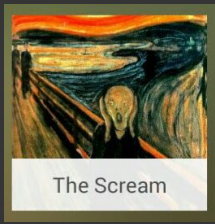
– Prisma App


impa
Instituto de Matemática Pura e Aplicada
Unité Mixte Internationale


Coloured


impa

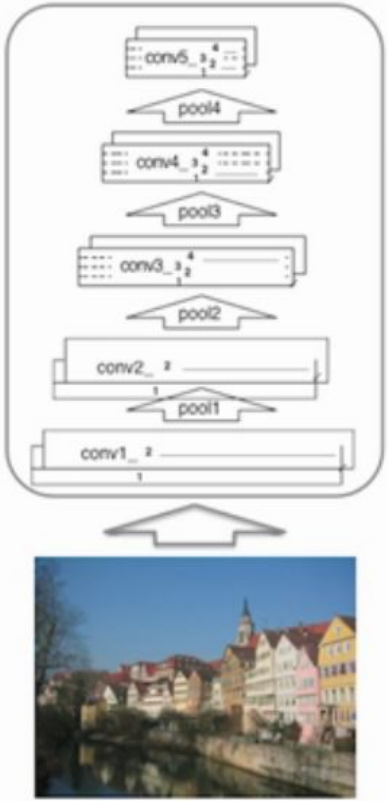

Mondrian


The Scream


impa

– “Ideal solution to the problem: Extract semantic content from the image and give it to the texture transfer procedure to render the semantic content of the target image in the style of the source image.”

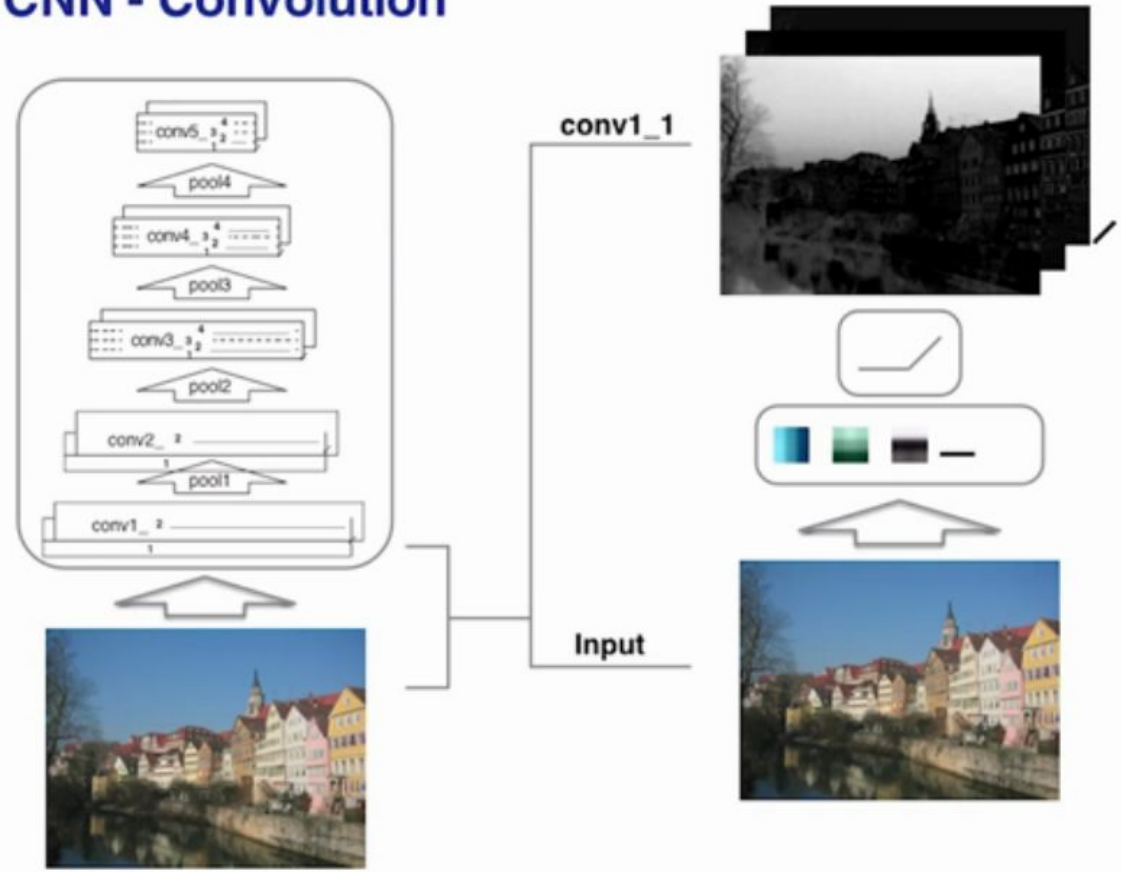
Convolutional Neural Network (CNN)



- Use VGG-19 network (Simonyan & Zissermann 2014)
- 2nd Place ImageNet 2014 object recognition challenge
- Consists of only 2 operations:
  - 3 x 3 x k rectified convolution
  - 2 x 2 max-pooling

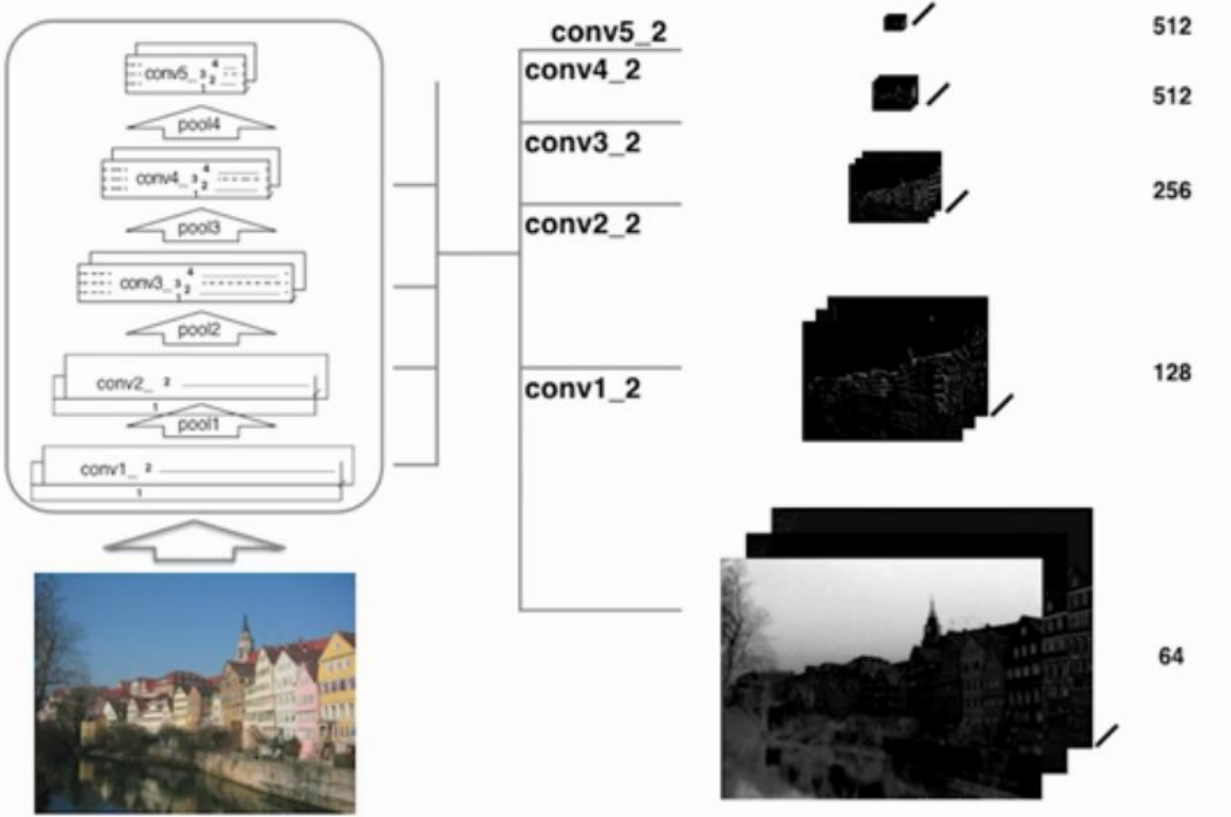
Leon gatys presentation from the 2016 DeepDream

CNN - Convolution



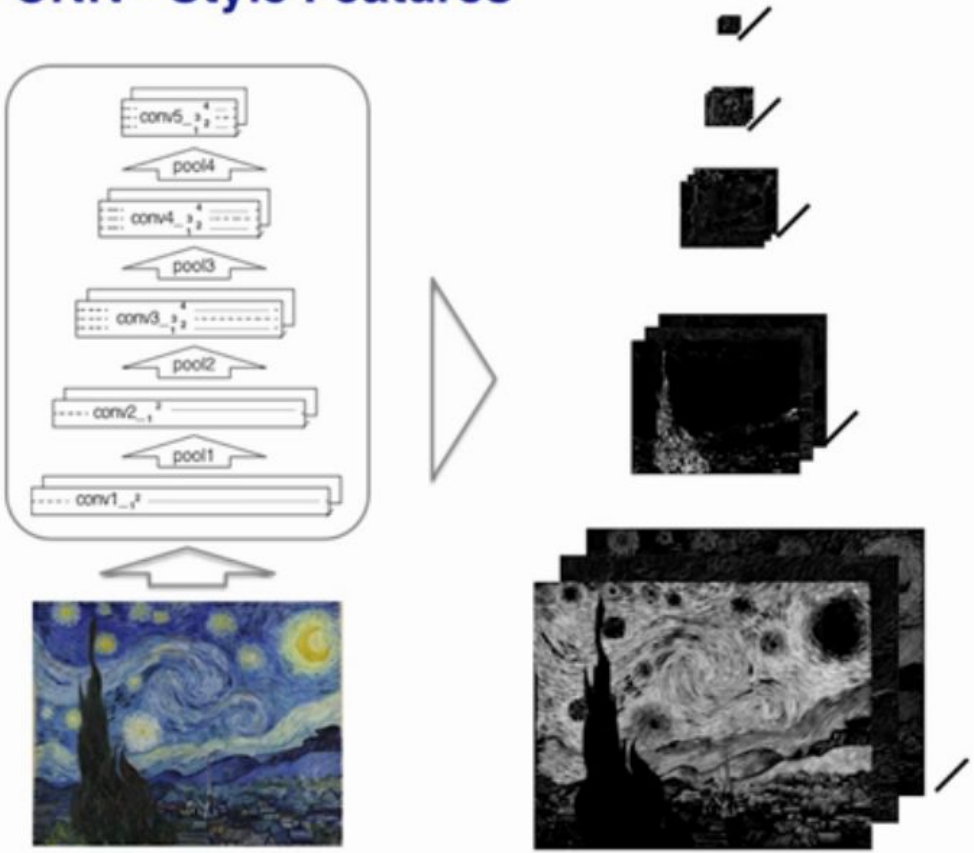
Leon gatys presentation from the 2016 DeepDream

CNN - Multiscale Nonlinear Filter Bank



Leon gatys presentation from the 2016 DeepDream

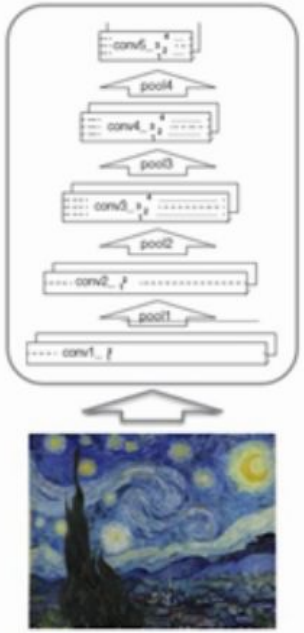
CNN - Style Features



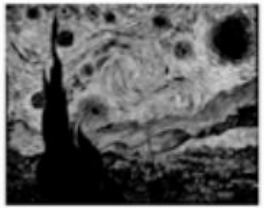
Gatys et al. (NIPS 2015)

Leon gatys presentation from the 2016 DeepDream

CNN - Style Features



X

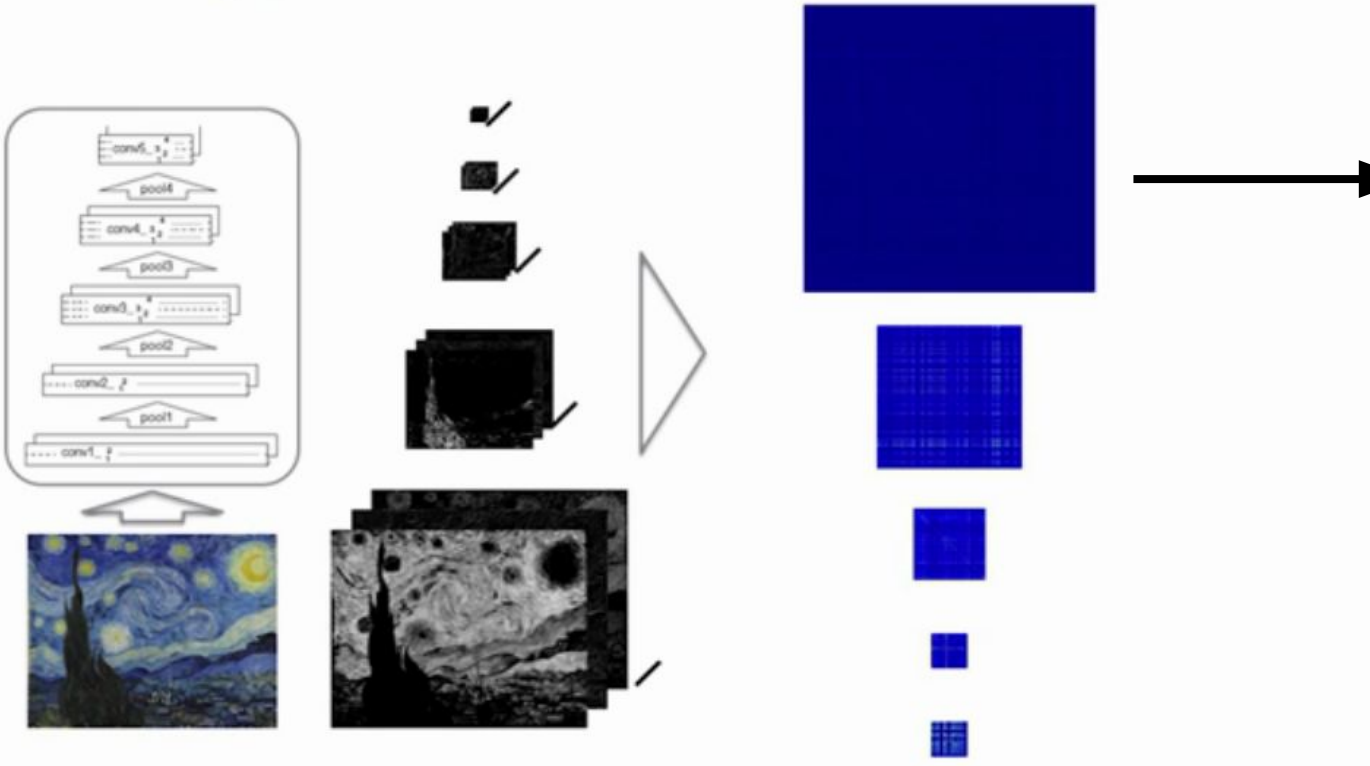


$$\sum = \left(\quad \langle \bar{f}_2, \bar{f}_1 \rangle \quad \right)$$

Gatys et al. (NIPS 2015)

Leon gatys presentation from the 2016 DeepDream

CNN - Style Features

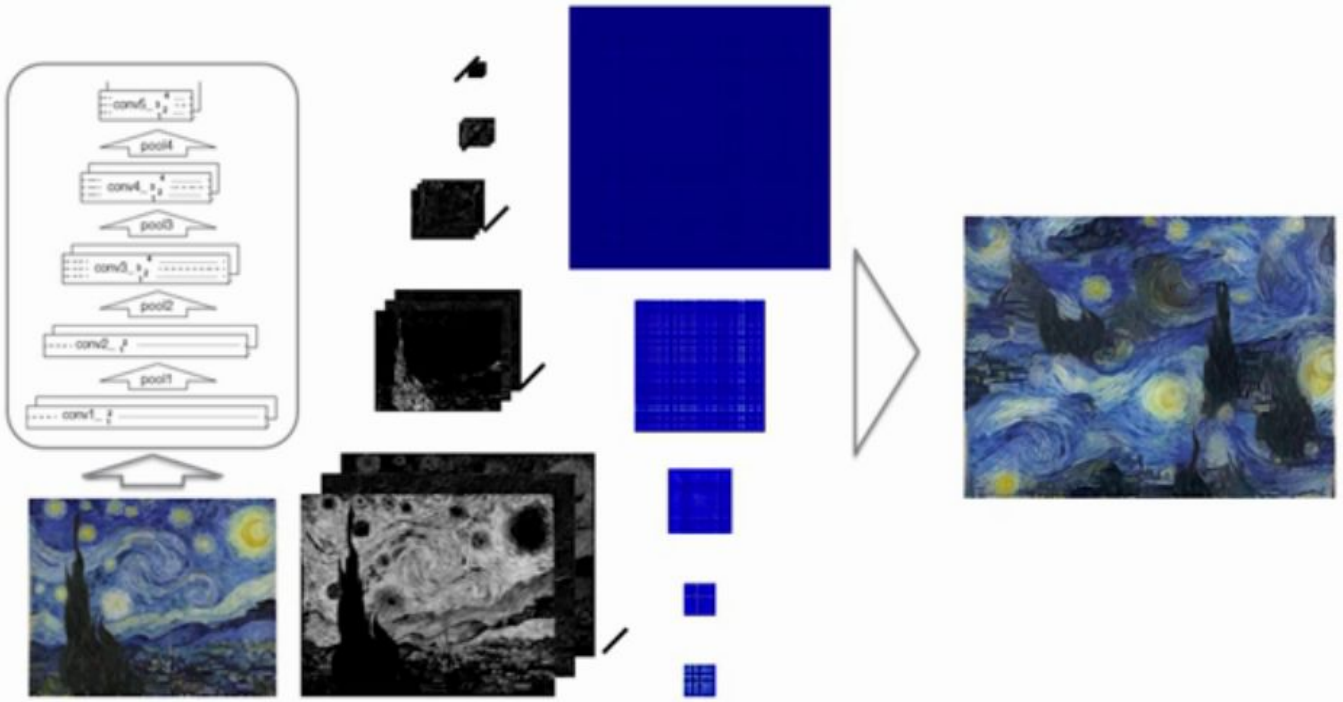


Gatys et al. (NIPS 2015)

arXiv:
1606.01286

Leon gatys presentation from the 2016 DeepDream

CNN - Style Features



Gatys et al. (NIPS 2015)

Leon gatys presentation from the 2016 DeepDream

Keep in mind

VGG-19 CNN

- 19 layers weight
- VGG-19 CNN
- Average pooling used

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

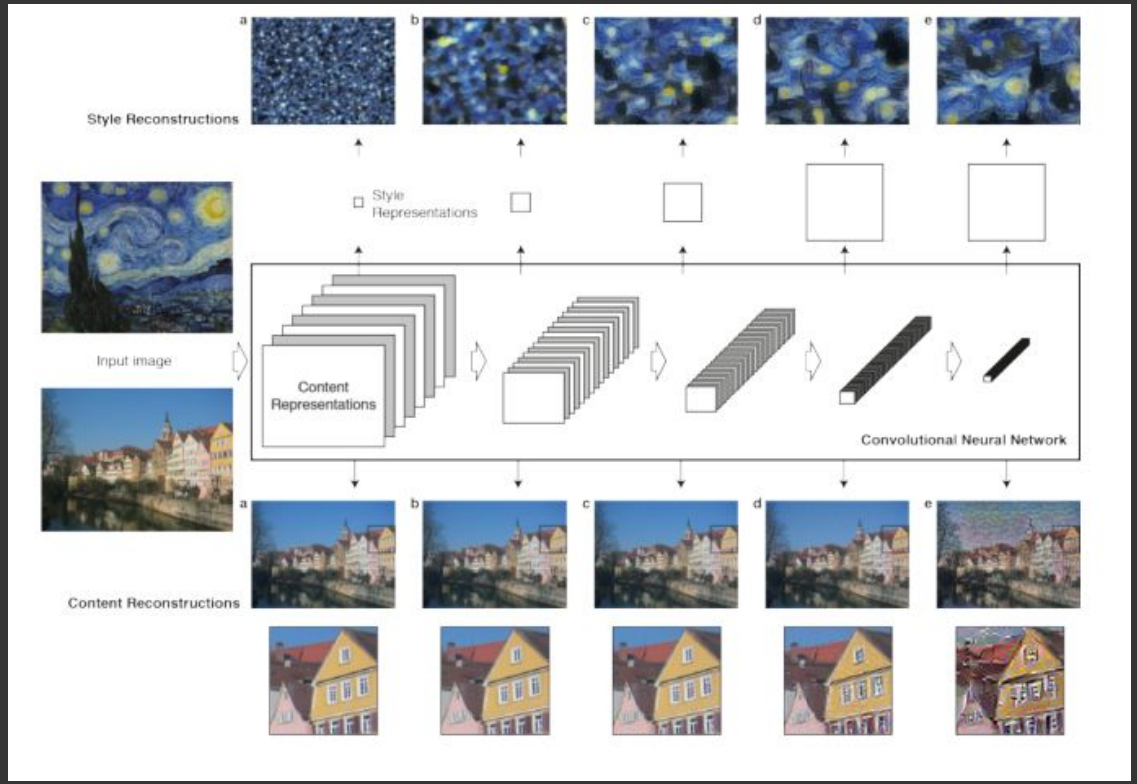
Table 2: **Number of parameters** (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

Karen Simonyan & Andrew Zisserman, VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION, 2015, arXiv:1409.1556

– Mathematical background

(In Gatys 2015 paper)



Mahendran, A., & Vedaldi, A. (2015). Understanding deep image representations by inverting them.

– Mathematical background

(In Gatys 2015 paper)

Feature representation

Original

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

Generated

$$\frac{\partial \mathcal{L}_{content}}{\partial F_{ij}^l} = \begin{cases} \left(F^l - P^l\right)_{ij} & \text{if } F_{ij}^l > 0 \\ 0 & \text{if } F_{ij}^l < 0 \end{cases}$$

$$\frac{\partial E_l}{\partial F_{ij}^l} = \begin{cases} \frac{1}{N_l^2 M_l^2} \left(\left(F^l\right)^{\mathrm{T}} \left(G^l - A^l\right)\right)_{ji} & \text{if } F_{ij}^l > 0 \\ 0 & \text{if } F_{ij}^l < 0 \end{cases}$$

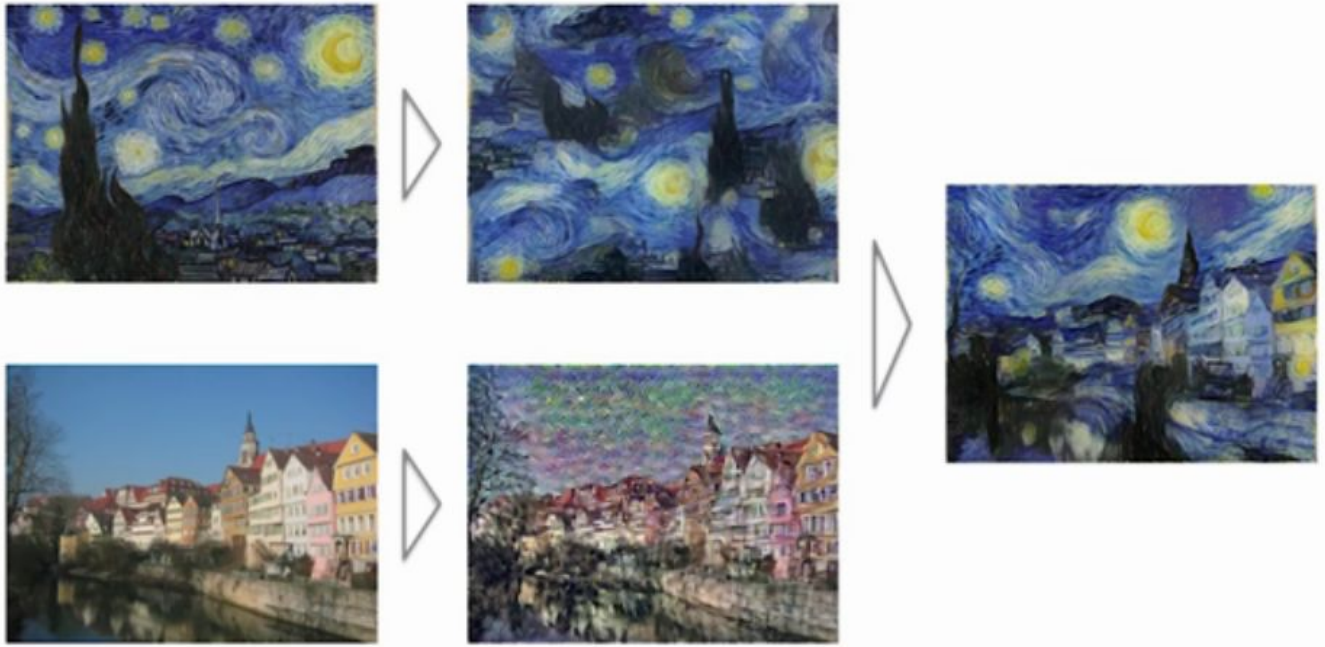
Style representation of original

$$G_{ij}^l = \sum_k F_{ik}^l F_{jk}^l$$

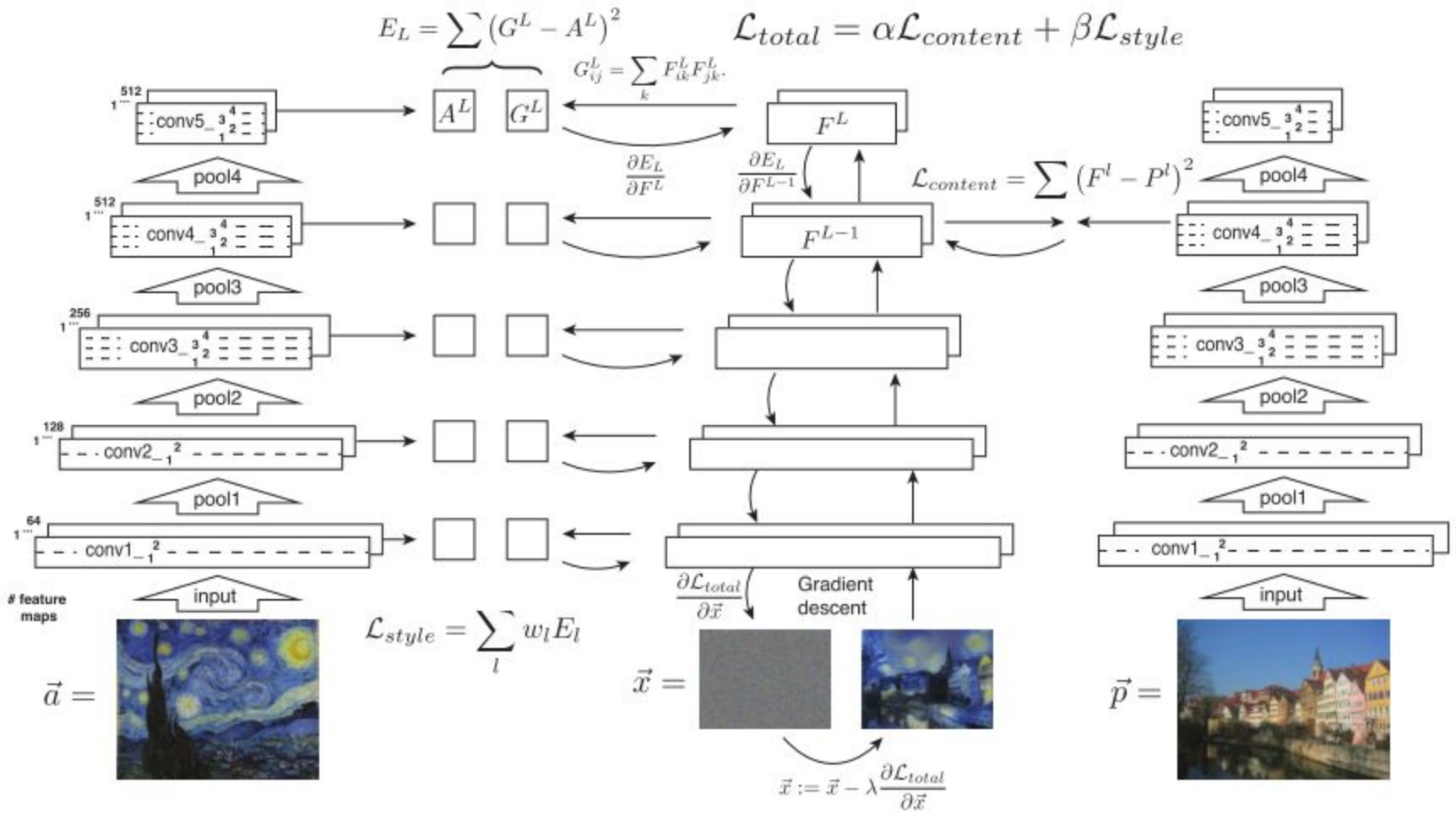
$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(G_{ij}^l - A_{ij}^l\right)^2$$

$$\mathcal{L}_{style}(\vec{a}, \vec{x}) = \sum_{l=0}^{L} w_l E_l$$

Artistic Style Transfer



Leon gatys presentation from the 2016 DeepDream


$E_L = \sum (G^L - A^L)^2$
$\mathcal{L}_{total} = \alpha \mathcal{L}_{content} + \beta \mathcal{L}_{style}$
$G^L_{ij} = \sum_k F^L_{ik} F^L_{jk}$
$\mathcal{L}_{content} = \sum (F^l - P^l)^2$
$\mathcal{L}_{style} = \sum_l w_l E_l$
$\vec{x} := \vec{x} - \lambda \frac{\partial \mathcal{L}_{total}}{\partial \vec{x}}$
conv5_1
conv4_1
conv3_1
conv2_1
conv1_1
pool4
pool3
pool2
pool1
input
feature maps
Gradient descent

Style weights

Content weight

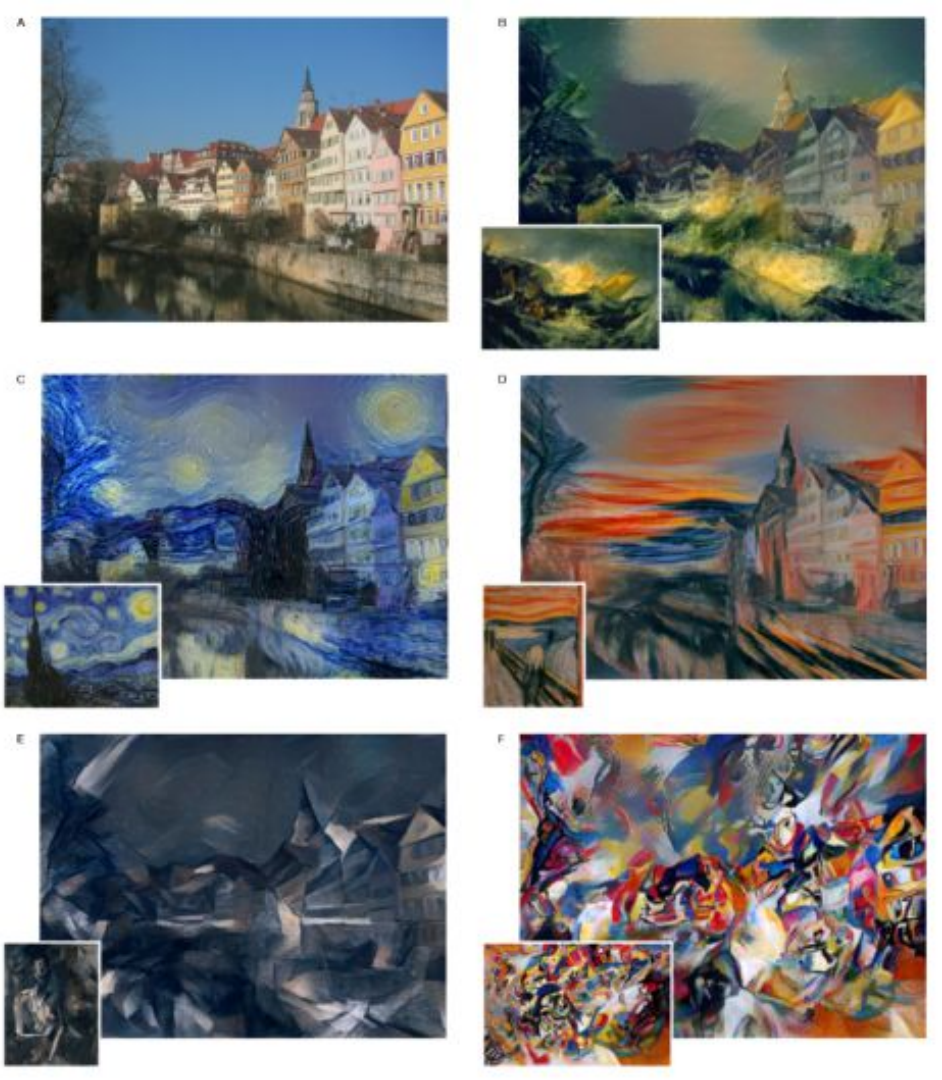
$$\mathcal{L}_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathcal{L}_{content}(\vec{p}, \vec{x}) + \beta \mathcal{L}_{style}(\vec{a}, \vec{x})$$

Style weights

Content weight

Key finding?

“content representation on layer ‘conv4_2’ and the style representations on layers ‘conv1_1’, ‘conv2_1’, ‘conv3_1’, ‘conv4_1’ and ‘conv5_1’ (wl = 1/5 in those layers, wl = 0 in all other layers) . The ratio α/β was either 1×10−3 (Fig 2 B,C,D) or 1×10−4 (Fig 2 E,F).”

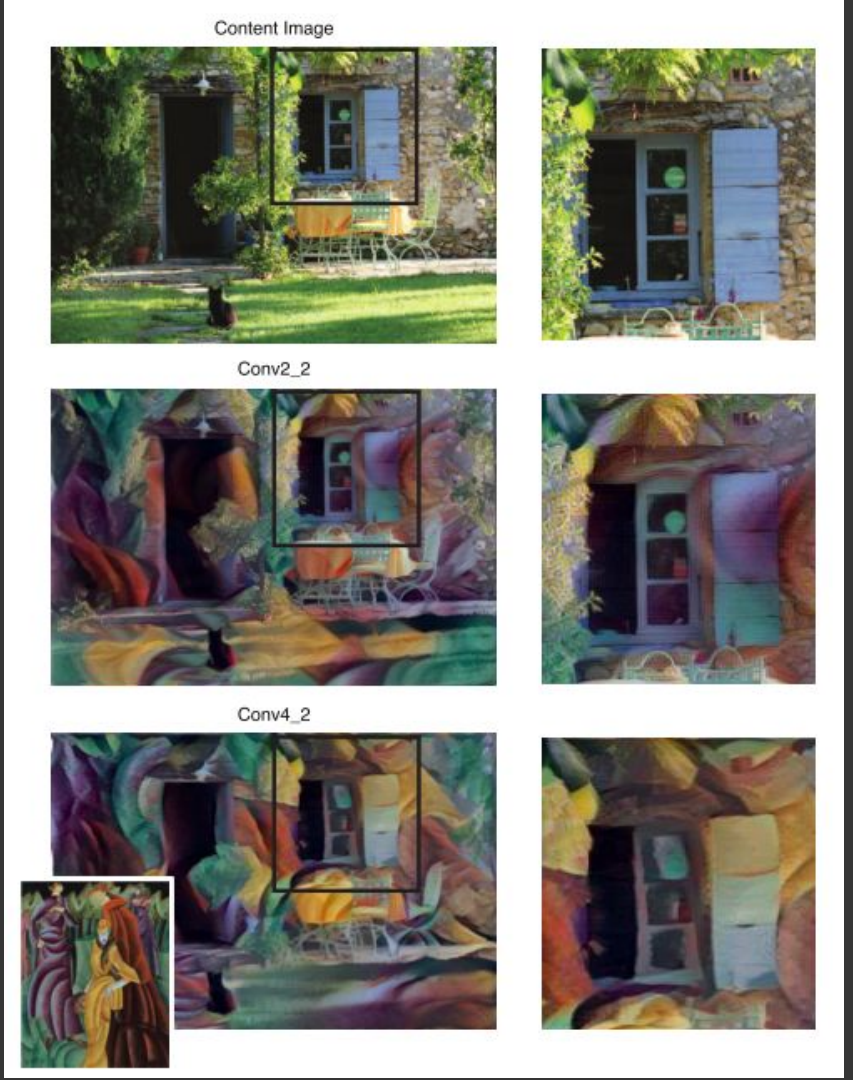


Style weights

Content weight

Key finding?

Content representation from different layers.



– A Faster Style Transfer Solution - Johnson/Ulyanov, 2016

Feed-forward generator network

$$W^* = \arg\min_W \mathbf{E}_{x,\{y_i\}} \left[\sum_{i=1} \lambda_i \ell_i(f_W(x), y_i) \right]$$

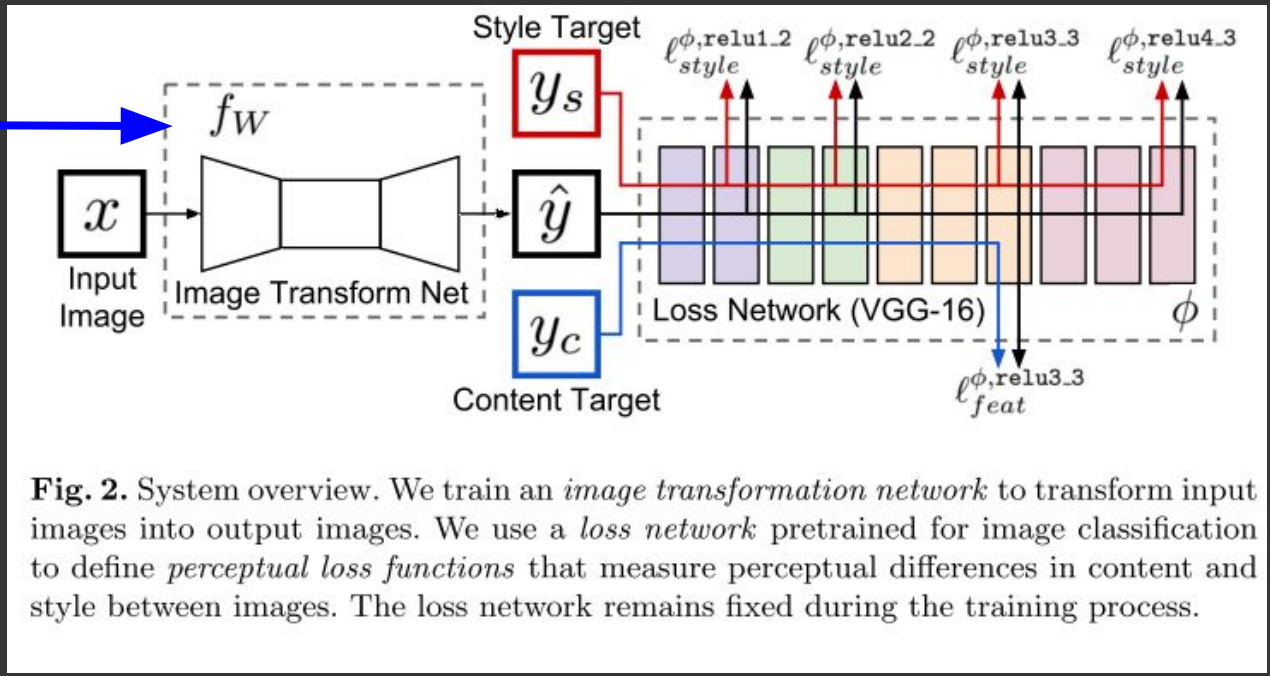


Fig. 2. System overview. We train an *image transformation network* to transform input images into output images. We use a *loss network* pretrained for image classification to define *perceptual loss functions* that measure perceptual differences in content and style between images. The loss network remains fixed during the training process.

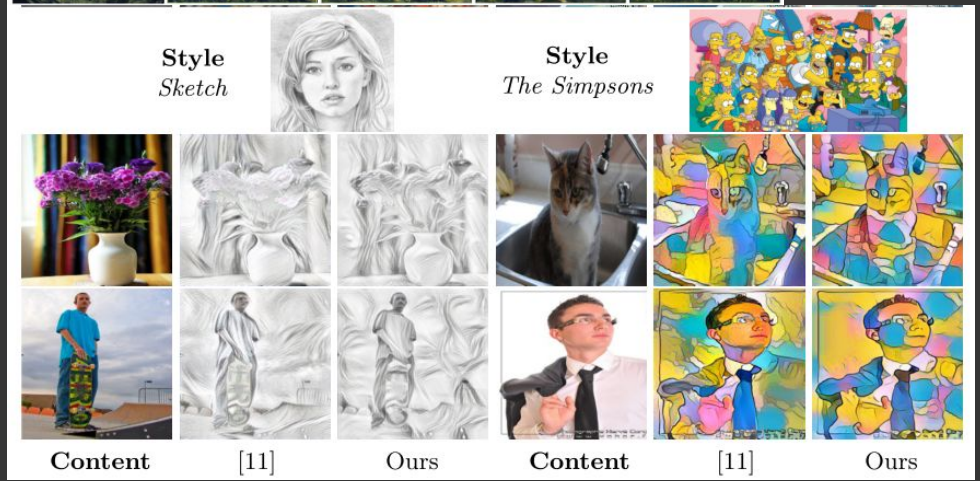
Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. In: ICLR. (2016)

– A Faster Style Transfer Solution - Johnson/Ulyanov, 2016

“hundreds of times faster (two order of magnitude)”

“Replacing batch normalization with instance normalization significantly improves the quality of feedforward style transfer models.” (Ulyanov, 2016)

Markovian Generative adversarial networks used in other work (Li & Wand, 2016).



– A Faster Style Transfer Solution - Johnson/Ulyanov,

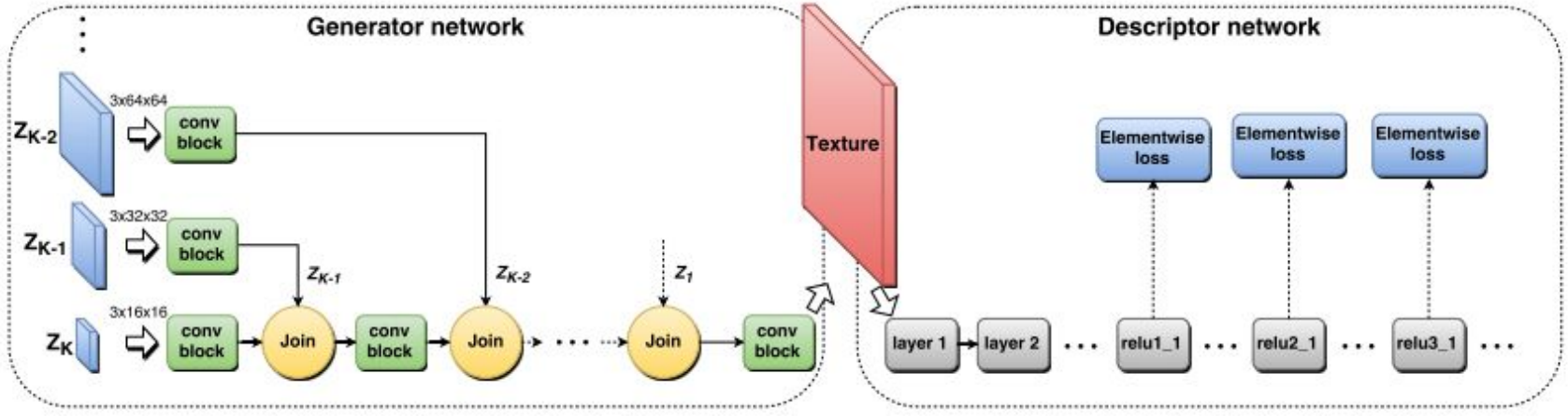


Figure 2. Overview of the proposed architecture (texture networks). We train a *generator network* (left) using a powerful loss based on the correlation statistics inside a fixed pre-trained *descriptor network* (right). Of the two networks, only the generator is updated and later used for texture or image synthesis. The `conv` block contains multiple convolutional layers and non-linear activations and the `join` block upsampling and channel-wise concatenation. Different branches of the generator network operate at different resolutions and are excited by noise tensors z_i of different sizes.

Ulyanov, 2016 - Network architecture

– A Faster Style Transfer Solution - Johnson/Ulyanov, 2016

"hundreds of times faster (two order of magnitude)"

"Instance Normalization: The Missing Ingredient for Fast Stylization"
(Ulyanov, 2016)

Images from:
https://github.com/jcjohnson/fast-neural-style

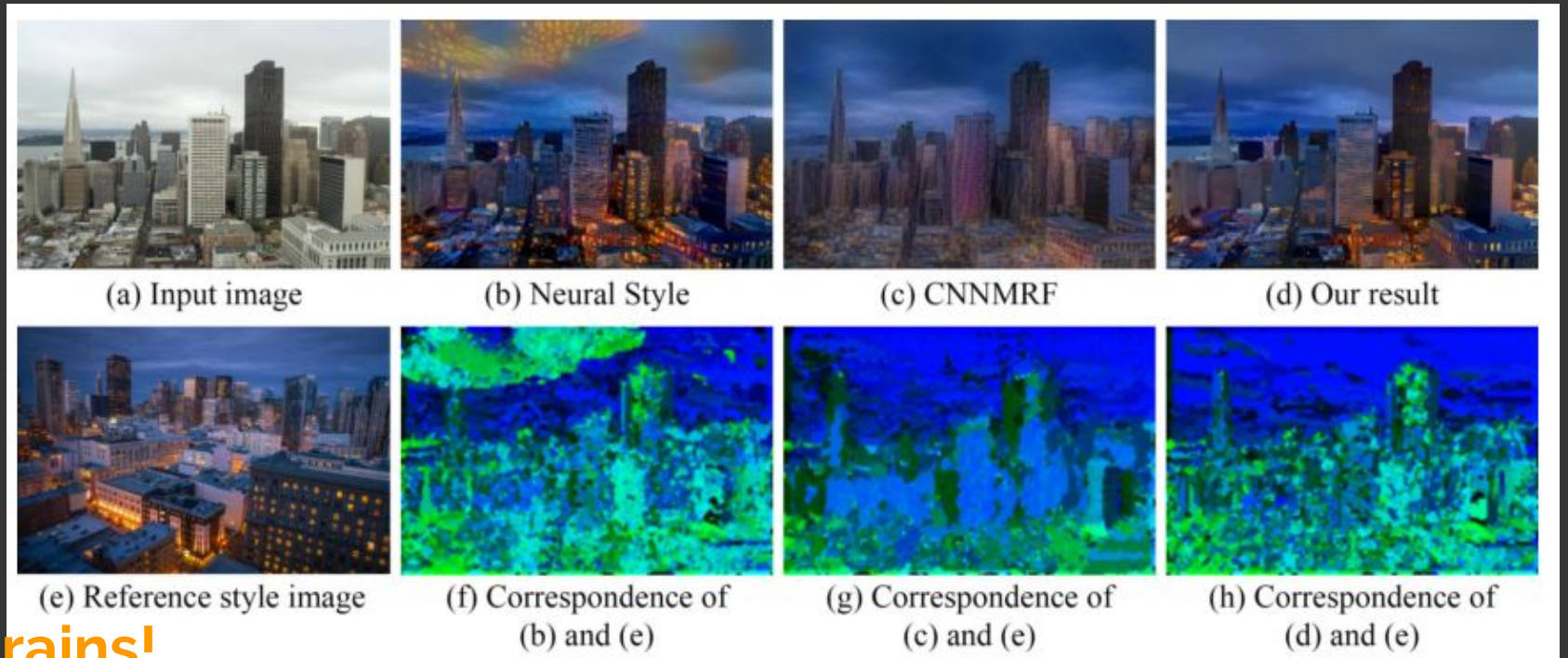
Markovian Generative adversarial networks used in other work (Li & Wand, 2016).

Beyond artistic transfer

Gatys, 2016

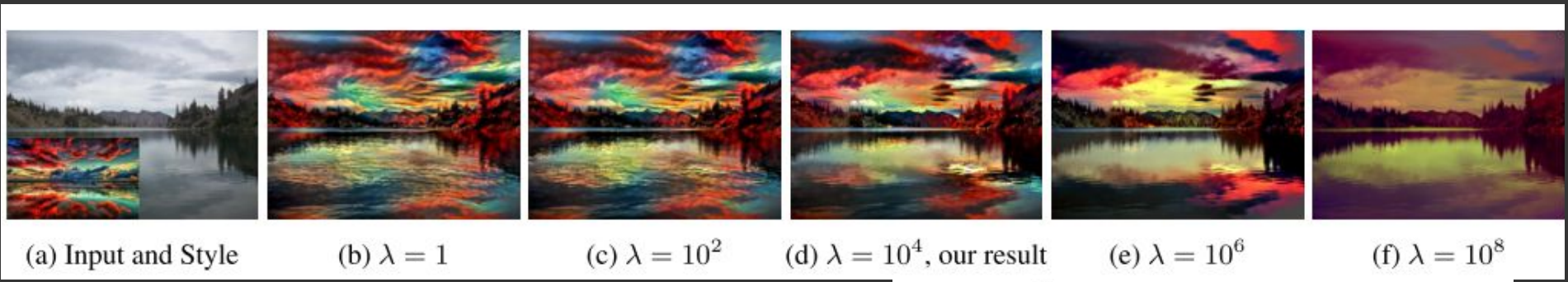
Figure 7. Photorealistic style transfer. The style is transferred from a photograph showing New York by night onto a picture showing London by day. The image synthesis was initialised from the content image and the ratio α/β was equal to 1×10^{-2}

– “Deep Photo Style Transfer”, Luan et al, 2017



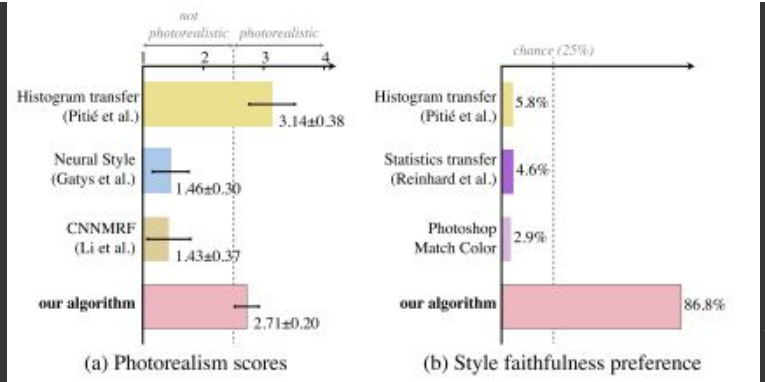
Constrains!

– “Deep Photo Style Transfer”,



(a) Input and Style (b) $\lambda = 1$ (c) $\lambda = 10^2$ (d) $\lambda = 10^4$, our result (e) $\lambda = 10^6$ (f) $\lambda = 10^8$

Constrains!
→ Painting effect
→ Context-semantic
→ Structure



(a) Photorealism scores (b) Style faithfulness preference

$$\mathcal{L}_m = \sum_{c=1}^{3} V_c[O]^T \mathcal{M}_I V_c[O] \quad (2)$$

Output (vector)

Input related

$$\mathcal{L}_{s+}^{\ell} = \sum_{c=1}^{C} \frac{1}{2N_{\ell,c}^2} \sum_{ij} (G_{\ell,c}[O] - G_{\ell,c}[S])_{ij}^2 \quad (3a)$$

$$F_{\ell,c}[O] = F_{\ell}[O] M_{\ell,c}[I] \quad F_{\ell,c}[S] = F_{\ell}[S] M_{\ell,c}[S] \quad (3b)$$

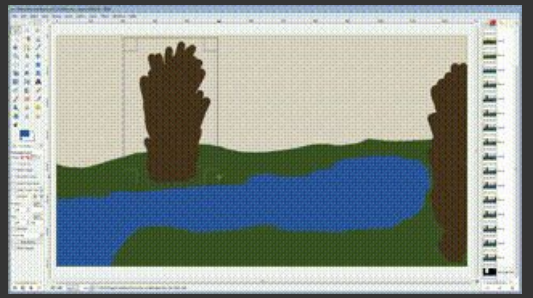
$$\mathcal{L}_{\text{total}} = \sum_{l=1}^{L} \alpha_{\ell} \mathcal{L}_c^{\ell} + \Gamma \sum_{\ell=1}^{L} \beta_{\ell} \mathcal{L}_{s+}^{\ell} + \lambda \mathcal{L}_m \quad (4)$$

(a) Input image (b) Reference style image (c) Neural Style (d) CNNMRF (e) Our result


Input
NS
Ours
(a) Reference style image
(b) Input image
(c) Neural Style (distortions)
(d) Our result
(e) Insets

Other trends

Semantic Style transfer (Champandard, 2016)



Faces

Videos (Ruder et al, 2016)

Typography; Natural language

Other trends

Semantic
(Champan
Faces

Videos (Ru
Typograp

- Applications

Apps (Prisma and similars)
Animations, movies...
New platforms
Art

Jing, et al Neural Style Transfer: A Review. (2017)

References

1) Ulyanov, D., Vedaldi, A., & Lempitsky, V. (2016). Instance Normalization: The Missing Ingredient for Fast Stylization, (2016). Retrieved from http://arxiv.org/abs/1607.08022
2) Ulyanov, D., Lebedev, V., Vedaldi, A., & Lempitsky, V. (2016). Texture Networks: Feed-forward Synthesis of Textures and Stylized Images. Retrieved from http://arxiv.org/abs/1603.03417
3) Jing, Y., Yang, Y., Feng, Z., Ye, J., & Song, M. (2017). Neural Style Transfer: A Review. Retrieved from http://arxiv.org/abs/1705.04058
4) Mahendran, A., & Vedaldi, A. (2015). Understanding deep image representations by inverting them. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 07–12–June, 5188–5196. http://doi.org/10.1109/CVPR.2015.7299155
5) Kyprianidis, J. E., Collomosse, J., Wang, T., & Isenberg, T. (2013). State of the 'Art: A taxonomy of artistic stylization techniques for images and video. IEEE Transactions on Visualization and Computer Graphics, 19(5), 866–885. http://doi.org/10.1109/TVCG.2012.160
6) Hertzmann, A., Jacobs, C. E., Oliver, N., Curless, B., & Salesin, D. H. (2001). Image analogies. Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques - SIGGRAPH '01, (August), 327–340. http://doi.org/10.1145/383259.383295
7) Ashikhmin, M. (2001). Synthesizing natural textures. Proceedings of the 2001 Symposium on Interactive 3D Graphics, 217–226. http://doi.org/10.1145/364338.364405
8) Efros, A. A., & Freeman, W. T. (2001). Image quilting for texture synthesis and transfer. Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques - SIGGRAPH '01, 341–346. http://doi.org/10.1145/383259.383296

Edvard Munch, The Scream

References

9) Gatys, L. A., Ecker, A. S., & Bethge, M. (2016). Image Style Transfer Using Convolutional Neural Networks. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 2414–2423). IEEE. http://doi.org/10.1109/CVPR.2016.265

10) Johnson, J., Alahi, A., & Fei-Fei, L. (2016). Perceptual Losses for Real-Time Style Transfer and Super-Resolution. Arxiv, 1–5. http://doi.org/10.1007/978-3-319-46475-6_43

11) Elad, M., & Milanfar, P. (2017). Style Transfer Via Texture Synthesis. IEEE Transactions on Image Processing, 26(5), 2338–2351. http://doi.org/10.1109/TIP.2017.2678168

12) Gatys, L. A., Ecker, A. S., & Bethge, M. (2015). A Neural Algorithm of Artistic Style, 3–7. http://doi.org/10.1167/16.12.326

13) Ruder, M., Dosovitskiy, A., & Brox, T. (2016). Artistic style transfer for videos. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 9796 LNCS, 26–36. http://doi.org/10.1007/978-3-319-45886-1_3

14) Luan, F., Paris, S., Shechtman, E., & Bala, K. (2017). Deep Photo Style Transfer. http://doi.org/10.1109/CVPR.2017.740

15) Gatys, L. A., Ecker, A. S., & Bethge, M. (2015). Texture Synthesis Using Convolutional Neural Networks, 1–10. http://doi.org/10.1109/CVPR.2016.265

16) Gatys, L. A., Ecker, A. S., Bethge, M., Hertzmann, A., & Shechtman, E. (2016). Controlling Perceptual Factors in Neural Style Transfer, 1(c). http://doi.org/10.1109/CVPR.2017.397

Edvard Munch, The Scream