Neural style transfer

Victor Kitov

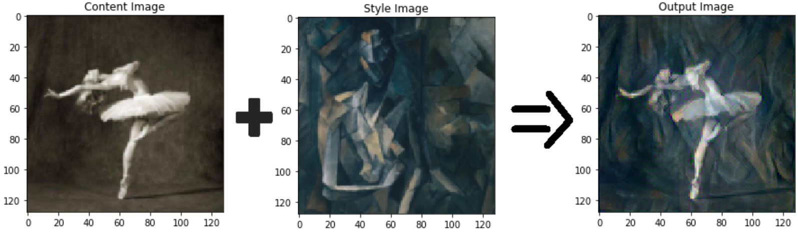
v.v.kitov@yandex.ru

Neural style transfer

- Input: content image, style image.
- Style transfer - application of artistic style from style image to content image.
- Easy to do with convolutional neural networks.

Example

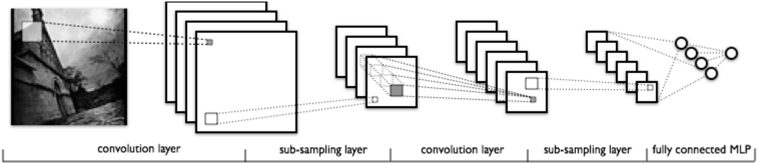
Example from PyTorch tutorial



Applications

- Enhancing social communication
  - adding personality
  - adding emotions
- User-assisted creation Tools
  - 2D drawings for painters
  - CAD drawings for designers, architects.
- Cheap cartoon creation from filmed scenes with actors.
- Applying special effects to
  - movies
  - computer games (interactive!)

Convolution network



What layers learn?[1]

- Image $x_0 \in \mathbb{R}^{HxWxC}$ produces at some level representation $\Phi_0 = \Phi(x_0)$.
- Find reconstructed image $x^* \in \mathbb{R}^{HxWxC}$ from

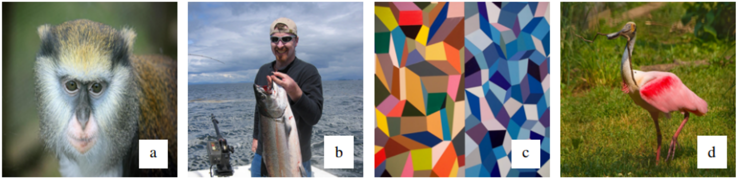
$$x^* = \arg\min_{x \in \mathbb{R}^{HxWxC}} \|\Phi(x) - \Phi_0\|_2^2 / \|\Phi_0\|_2^2 + \lambda_\alpha R_\alpha(x) + \lambda_\beta R_\beta(x)$$

- Regularizers:
    - $R_\alpha(x) = \|x\|_\alpha^\alpha$ for vectorized and mean subtracted x
    - $R_\beta(x) = \sum_{i,j} \left((x_{i,j+1} - x_{i,j})^2 + (x_{i+1,j} - x_{i,j})^2\right)^\beta$ (total variation)
- AlexNet taken.

[1] Mahendran et. al. 2015. Understanding Deep Image Representations by Inverting Them.

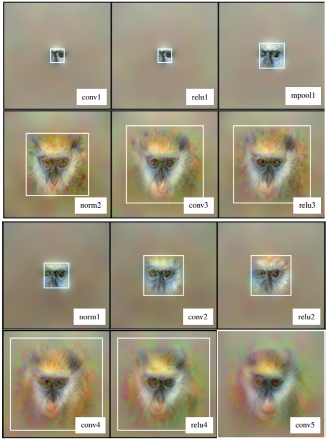
Original images

Original images



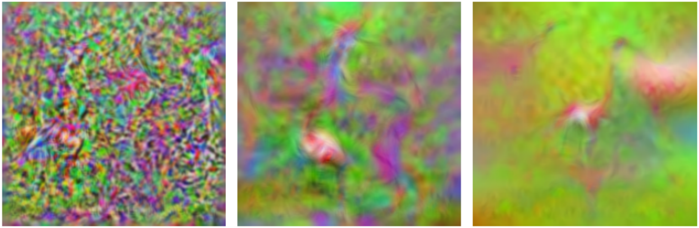
Receptive field

- Receptive field of central 5x5 patch grows for deeper (later) layers:



Without regularization reconstructed image is not interpretable

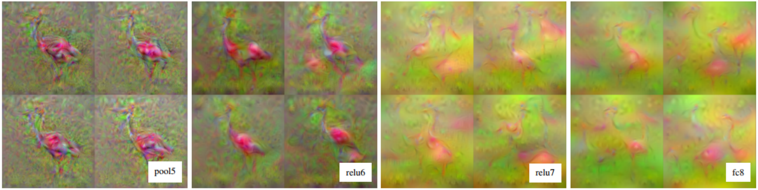
Reconstruction with increasing λ_β



Reconstructions from different initial conditions

- Take flamingo picture, calculate inner representations.
- Reconstruct original image from fixed inner representation and 4 random initial white noise approximations of the original image.

Deeper layers reconstruct more general concepts.



- Deeper representations capture progressively larger deformations of the original object.
- For example layer 8 reconstructs multiple flamingos at different positions.

Rich information is saved in deep layers

Deep layers (mpool5 here) reconstruct most informative parts of the original picture:

Seminal work[2]

Consider convolutional network (VGG)

Denote:

- Images:
  - x_c - content image;
  - x_s - style image
  - x - stylized image (to be found)
- $\Phi_{ij}^l(x)$ - output of i-th filter on j-th position on layer l.
  - N_l filters and $M_l = H_l \times W_l$ spatial positions.

[2] Gatys et al (2015). A Neural Algorithm of Artistic Style.

Definitions

- Gram matrices $G_{ij} = \sum_k \Phi_{ik}(x_c)\Phi_{jk}(x_c)$, $A_{ij} = \sum_k \Phi_{ik}(x)\Phi_{jk}(x)$
- Content loss:

$$E_c(x, x_c) = \frac{1}{2}\sum_{i,j}\left\|\Phi^l_{ij}(x) - \Phi^l_{ij}(x_c)\right\|^2$$

- Style loss for one layer:

$$E^l_s = \frac{1}{4N_l^2 M_l^2}\sum_{i,j}\left(G^l_{ij} - A^l_{ij}\right)^2$$

  - inter-channels correlations="style"
  - spatial components ignored (stands for content).
  - higher l=>higher order style.

Losses

- Total style loss:

$$E_s(x, x_s) = \sum_{l=0}^{L} w_l E_l$$

- x is found from

$$x = \arg\min_x \{\alpha E_c(x, x_c) + \beta E_s(x, x_s)\} \qquad (1)$$

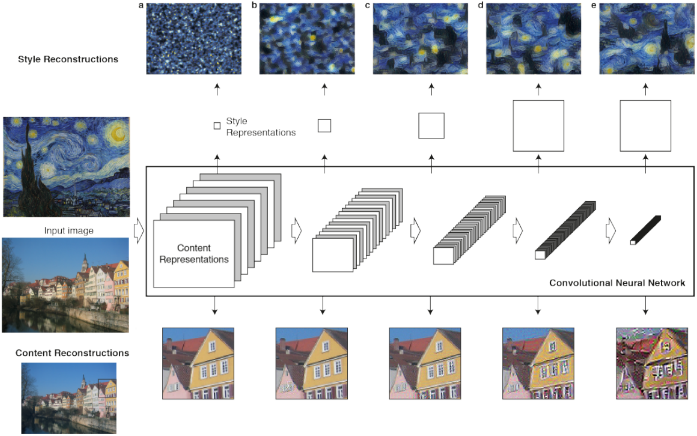
1. initialize x randomly (or $x = x_c$ - works better)
2. use back-propagation to update x

Style transfer algorithm

1. Pretrain CNN
2. Compute features for content image
3. Compute Gram matrices for style image
4. Randomly initialize new image (from content image also possible)
5. repeat until convergence:
    1. Forward new image through CNN
    2. Compute style loss (L2 distance between Gram matrices) and content loss (L2 distance between features)
    3. Loss is weighted sum of style and content losses
    4. Backprop to image

Deeper layers contain more abstract information

- Content is reconstructed from its activations on deep layer
- Style is reconstructed from correlations of its activations on deep layer



Visualizations

Increasing α/β (relative importance of content to style): content more visible.

Reconstructed style (small $\frac{\alpha}{\beta}$) with increasing number of layers in E_s: more abstract reconstruction.

Spatial control[3]

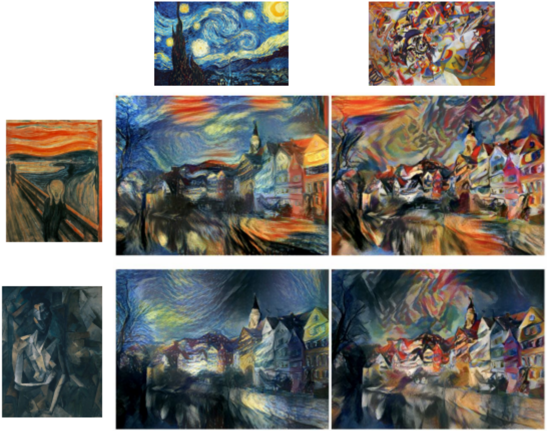
- Implemented by applying masks to content/modified image, $E_s^l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(G_{ij}^l - A_{ij}^l \right)^2$, M_l - # of points in the mask.
- On example below: best result is obtained when
  - style 1 is applied only to house
  - style 2 is applied only to the rest of the image

[3] Gatys et. al. (2017) Controlling Perceptual Factors in Neural Style Transfer.

Results without and with spatial control (house style from b, sky style from c)

May mix styles[4]

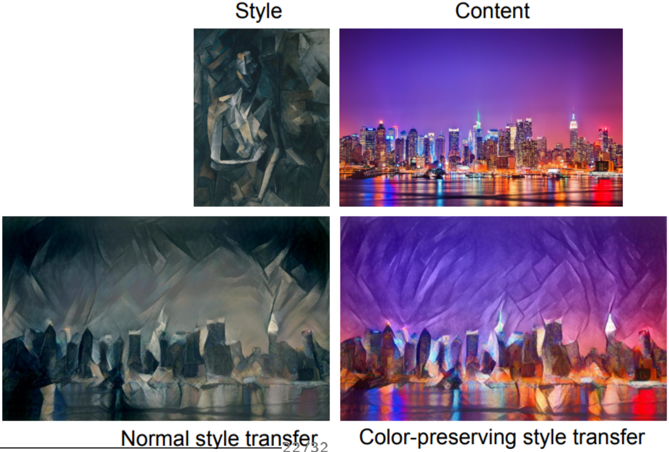
Mix style from multiple images by taking a weighted average of Gram matrices of their activations.



[4] https://github.com/jcjohnson/neural-style

Preserve color of content[5]

- Perform style transfer only on the luminance (brightness) channel (Y in YUV colorspace)
- Copy colors from content image



Generalization with other kernels[6]

- Consider 2 samples $X = \{x_i\}_{i=1}^{n}$, $Y = \{y_j\}_{j=1}^{m}$ transformed with $\phi(\cdot)$ and kernel $k(x, y) = \langle \phi(x), \phi(y) \rangle$.
- Are X and Y equally distributed?
- Can check that with *MMD* statistic:

$$
\begin{aligned}
&\mathrm{MMD}^2[X,Y] = \|\mathbf{E}_x[\phi(\mathbf{x})] - \mathbf{E}_y[\phi(\mathbf{y})]\|^2 \\
&= \|\frac{1}{n}\sum_{i=1}^{n}\phi(\mathbf{x}_i) - \frac{1}{m}\sum_{j=1}^{m}\phi(\mathbf{y}_j)\|^2 \\
&= \frac{1}{n^2}\sum_{i=1}^{n}\sum_{i'=1}^{n}\phi(\mathbf{x}_i)^T\phi(\mathbf{x}_{i'}) + \frac{1}{m^2}\sum_{j=1}^{m}\sum_{j'=1}^{m}\phi(\mathbf{y}_j)^T\phi(\mathbf{y}_{j'}) \\
&\quad - \frac{2}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m}\phi(\mathbf{x}_i)^T\phi(\mathbf{y}_j), \\
&= \frac{1}{n^2}\sum_{i=1}^{n}\sum_{i'=1}^{n}k(\mathbf{x}_i,\mathbf{x}_{i'}) + \frac{1}{m^2}\sum_{j=1}^{m}\sum_{j'=1}^{m}k(\mathbf{y}_j,\mathbf{y}_{j'}) \\
&\quad - \frac{2}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m}k(\mathbf{x}_i,\mathbf{y}_j).
\end{aligned}
$$

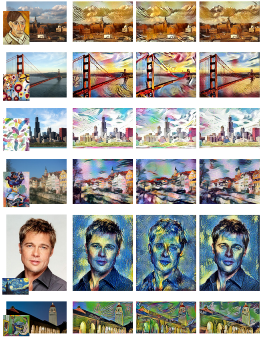
[6] Li et. al. (2017). Demystifying Neural Style Transfer.

Generalization with other kernels

- It's easy to show that $E_s^l = \frac{1}{4N_l^2} MMD^2[X, Y]$ where $x_j = \Phi^l_{\cdot j}(x_c)$, $y_j = \Phi^l_{\cdot j}(x)$ - vectors of features, j-spatial location.
- Extensions: take different kernel!
    - linear, multinomial, Gaussian.

Different kernels

Style transfer with different kernels



(a) Content / Style (b) linear (c) poly (d) Gaussian

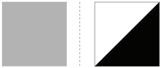
Adding histogram regularizer [7]

Gram matrix does not reveal all statistical properties of style!
Take random vector $X \in \mathbb{R}^D$, its Gram matrix is $\mathbb{E}XX^T$, $\mathbb{E}X = \mu$, $cov(x) = \Sigma$.

$$G = \mathbb{E}XX^T = \Sigma + \mu\mu^T$$

different combinations of μ and Σ give the same Gram matrix!

- For example, these 2 vectorized feature outputs would give identical Gram matrix (a scalar):



[7] Risser et. al. (2017). Stable and Controllable Neural Texture Synthesis and Style Transfer Using Histogram Losses.

Additional regularizers

- Add more regularizers to make optimization more stable.
- +total variation:
  $R_\beta(x) = \sum_{i,j}\left((x_{i,j+1} - x_{i,j})^2 + (x_{i+1,j} - x_{i,j})^2\right)^\beta$
- +histogram loss $\sum_{l=1}^{L} \gamma_l \left\| \tilde{O}_i^l - O_i^l \right\|$:
  - for each layer l and feature i calculate "grayscale image" of spatial outputs O_i^l.
  - convert O_i^l to $\tilde{O}_i^l$ having the same grayscale colors histogram [8] as style image (should be the same size).

[8] https://en.wikipedia.org/wiki/Histogram_matching

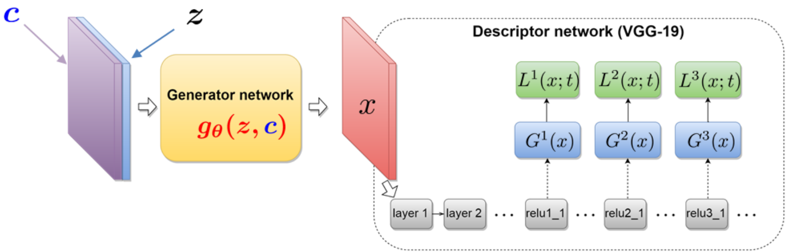
Table of Contents

Idea[9]

- Train generative network $g_\theta(z, c)$
    - c: content image
    - z: Gaussian random noise (for diversity of output results)
    - θ: trained parameters (weights)
- Loss from (1) - as in Gatys et al (2015).

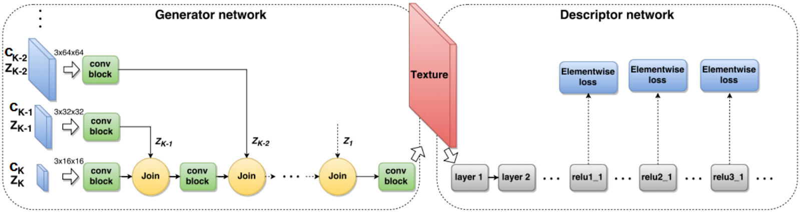
[9] Ulyanov et. al. (2016). Texture Networks: Feed-forward Synthesis of Textures and Stylized Images.

Proposed architecture



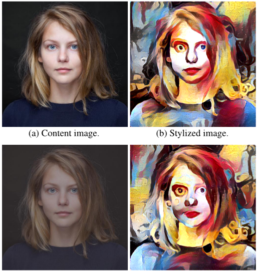
Generator network

- $c_1, c_2, c_3, \dots$ - downsampled of content image
- $z_1, z_2, z_3, \dots$ - random noise, generating randomness of different abstract levels.
- Only generator network is trained, descriptor network held fixed.
- In descriptor network (first layers of VGG) style transfer loss (1) is calculated.



Instance normalization[10]

- Instead of batch normalization use instance normalization (i.e. normalize features at different layers for single object) at training and test time.
- Allows to build transfers robust to brightness/contrast of the original image.



(a) Content image. (b) Stylized image.

(c) Low contrast content image. (d) Stylized low contrast image.

[10] Ulyanov et al, "Instance Normalization: The Missing Ingredient for Fast Stylization", ICML 2016