Fast Face-swap Using Convolutional Neural Networks

Iryna Korshunova, Wenzhe Shi, Joni Dambre, Lucas Theis
{iryna.korshunova, joni.dambre}@ugent.be, {wshi, ltheis}@twitter.com

Twitter

#KeyIdea

Frame the face swapping problem in terms of **style transfer** [1,2] and make use of convolutional neural networks trained to capture the appearance of the target identity from an unstructured collection of his/her photographs.

Combine neural networks with simple pre- and post-processing steps, so the face swap works in real-time with no input from the user.

#Problem

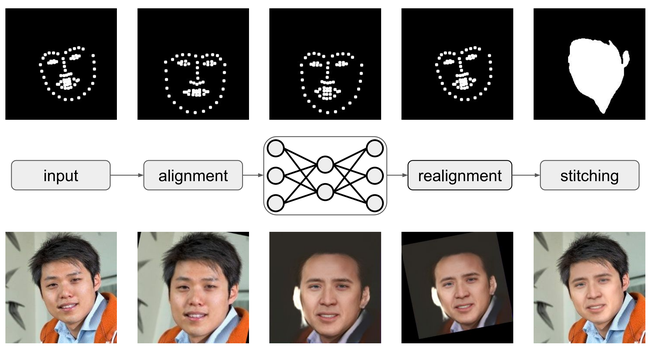
Having an image of person A, we would like to transform his/her identity into person B's identity while keeping head pose, expression and lighting conditions intact. In terms of style transfer, we can think of input image A's pose and expression as the content, and image B's identity as the style. We assume that we are given a **set of style images**, which describe the target identity.

Content: pose, expression and lighting

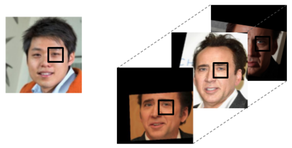
Style: identity

#Method

The identity replacement is done via a convolutional neural network with two extra pre- and post-processing components performing face alignment and background/hair/skin segmentation.

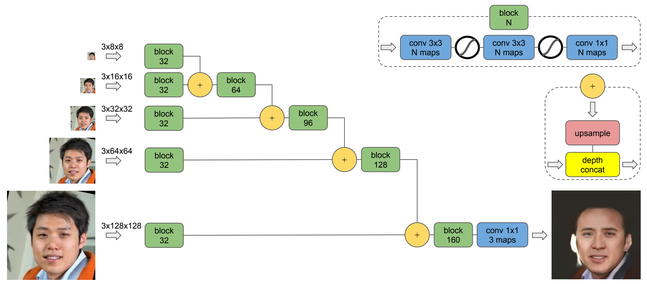


We align all images to a frontal-view reference face, such that we can easily **match neural patches** from content and style images which roughly correspond to the **same location** within a face.



#TransformationNetwork

The transformation network has a **multiscale architecture** with branches operating on different downsampled versions of the input image [3].



Content loss

$\mathbf{x}$ – input image
$\hat{\mathbf{x}}$ – generated image
$\Phi_l(\mathbf{x})$ – VGG representation of $\mathbf{x}$ on layer l
$|\Phi_l(\mathbf{x})|$ – cardinality of $\Phi_l(\mathbf{x})$

$$\mathcal{L}_{content}(\hat{\mathbf{x}}, \mathbf{x}, l) = \frac{1}{|\Phi_l(\mathbf{x})|}\|\Phi_l(\hat{\mathbf{x}}) - \Phi_l(\mathbf{x})\|_2^2$$

Style loss

$\mathbf{y}$–image from the set of style images $Y = \{\mathbf{y}_1, \dots \mathbf{y}_N\}$
$\Psi(\Phi_l(\mathbf{x}))$–list of neural patches generated by looping over spatial locations in $\Phi_l(\mathbf{x})$
M – number of neural patches
$d_c(\cdot, \cdot)$ – cosine distance
N_{best}–number of best matching style images from Y

$$\mathcal{L}_{style}(\hat{\mathbf{x}}, \mathbf{Y}, l) = \frac{1}{M}\sum_{i=1}^{M} d_c\left(\Psi_i(\Phi_l(\hat{\mathbf{x}})), \Psi_i(\Phi_l(\mathbf{y}_{NN(i)}))\right)$$

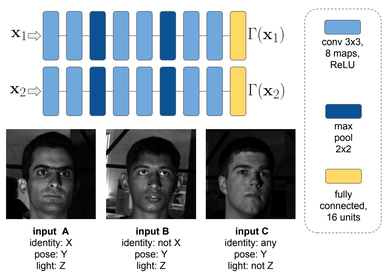
$$NN(i) = \underset{j=1,\dots,N_{best}}{\operatorname{argmin}}\ d_c\left(\Psi_i(\Phi_l(\hat{\mathbf{x}})), \Psi_i(\Phi_l(\mathbf{y}_j))\right)$$

Light loss

$\Gamma(\mathbf{x})$–representation of $\mathbf{x}$ in the feature space of the light network
$|\Gamma(\mathbf{x})|$ – dimensionality of $\Gamma(\mathbf{x})$

$$\mathcal{L}_{light}(\hat{\mathbf{x}}, \mathbf{x}) = \frac{1}{|\Gamma(\mathbf{x})|}\|\Gamma(\hat{\mathbf{x}}) - \Gamma(\mathbf{x})\|_2^2$$

The **light network** is a siamese network [4] trained to maximize a distance between images with different lighting conditions and to minimize it for pairs with equal illumination. The distance is an L2 norm in the feature space of the fully connected layer.



Total loss

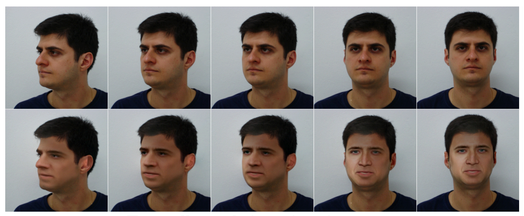
$$\mathcal{L}(\hat{\mathbf{x}}, \mathbf{x}, \mathbf{Y}) = \mathcal{L}_{content}(\hat{\mathbf{x}}, \mathbf{x}) + \alpha\mathcal{L}_{style}(\hat{\mathbf{x}}, \mathbf{Y}) + \beta\mathcal{L}_{light}(\hat{\mathbf{x}}, \mathbf{x}) + \gamma\mathcal{L}_{TV}(\hat{\mathbf{x}})$$

#ExperimentsAndResults

Top: original images. **Middle:** face swapping with Nicolas Cage and Taylor Swift.
Bottom: raw outputs of the Cage- and SwiftNet

original $\alpha = 80$ $\alpha = 120$

original $\beta = 10^{-22}$ $\beta = 0$



Top: effect of changing the pose.
Right: difficult cases.

#References

[1] L. A. Gatys, A. S. Ecker, and M. Bethge. *Image style transfer using convolutional neural networks*. In IEEE Conference on Computer Vision and Pattern Recognition, 2016
[2] C. Li and M. Wand. *Combining Markov random fields and convolutional neural networks for image synthesis*. In IEEE Conference on Computer Vision and Pattern Recognition, 2016
[3] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. Lempitsky. *Texture networks: Feed-forward synthesis of textures and stylized images*. In International Conference on Machine Learning, 2016
[4] S. Chopra, R. Hadsell, and Y. Lecun. *Learning a similarity metric discriminatively, with application to face verification*. In IEEE Conference on Computer Vision and Pattern Recognition, 2005