# Week10 Pre-Report: Image Style Transfer Using Convolutional Neural Network and Perceptual Losses for Real-Time Style Transfer and Super-Resolution

Euijin Hong[1]
[1]Department of Electrical and Electronic Engineering, Yonsei University.

## 1 Introduction

First we will discover the idea of image style transfer with Convolutional Neural Networks, which is stated in [9]. The existing non-parametric algorithms [3, 4, 13, 19] have a limitation of using only low-level features of the target image to inform the texture transfer. Ideally, a style transfer algorithm should be able to separate the semantic image content from its style and then render the semantic content of the target image in the style of the source image. Deep Convolutional Neural Networks have produced powerful computer vision systems that can extract high-level semantic information from natural images. The authors propose a new algorithm called *A Neural Algorithm of Artistic Style* that uses feature representations learned by Convolutional Neural Networks to independently process and manipulate the content and style of natural images for style transfer. This algorithm combines a parametric texture model based on Convolutional Neural Networks with a method to invert their image representations.

Then we will discover about *perceptual loss* from [12] in which discusses how classic image problems like denoising, super-resolution, and colorization can be solved using a feed-forward convolutional neural network. While per-pixel loss functions have traditionally been used to measure the difference between output and ground-truth images[1, 2, 5, 6, 14], recent work has shown that using high-level image feature representations from a pretrained convolutional neural network can produce higher quality images[7, 8, 15, 17, 20]. The research proposes a combination of these approaches by training the transformation networks using *perceptual loss functions* that depend on high-level features from a pretrained loss network. This allows the transfer of semantic knowledge from the loss network to the transformation network. The approach is tested on style transfer and single-image super-resolution tasks and produces visually pleasing results in real-time.

## 2 Theory and Configuration

### 2.1 Image Style Transfer Using CNN

The results are based on pretrained and normalized (by re-scaling the weights that have their mean activation as one) VGG19 network [16], using the feature space of 16 convolutional layers and 5 pooling layers while not using the fully connected layers. Average pooling is used since it shows slightly better results than max pooling.

**Content representation** Content loss function (equation 1) is defined as a squared-error loss between two feature representations: feature representation of original image $P^l$ and that of the generated image $F^l$.

$$\mathcal{L}_{content}(\vec{p},\vec{x},l) = \frac{1}{2}\sum_{i,j}(F^l_{ij} - P^l_{ij})^2 \tag{1}$$

The derivative of the loss w.r.t. the activations in layer $l$ is expressed in equation 2, where gradients can be computed by standard error back-propagation.

$$\frac{\partial \mathcal{L}_{content}}{\partial F^l_{ij}} = \begin{cases} (F^l_{ij} - P^l_{ij}) & \text{if } F^l_{ij} > 0. \\ 0 & \text{if } F^l_{ij} < 0. \end{cases} \tag{2}$$

So, we can change the initially random image $\vec{x}$ until it creates an identical response as the original image $\vec{p}$ in a certain layer of the CNN.

When training Convolutional Neural Networks for object recognition, the image is transformed into representations that become more sensitive to its content and less dependent on its appearance along the processing hierarchy. The higher layers capture the high-level content of the input image in terms of objects and their arrangement, while the lower layers reproduce the exact pixel values. The feature responses in higher layers are referred to as the *content representation*.

**Style representation** To represent an input image's style, we create a feature space that captures texture information using filter responses from any network layer. The space is based on correlations between the filter responses, with the Gram matrix $G^l \in \mathcal{R}^{N_l \times N_l}$, representing the inner product between vectorized feature maps $i$ and $j$ in layer $l$.

$$G^l_{ij} = \sum_k F^l_{ik} F^l_{jk} \tag{3}$$

Using feature correlations from multiple layers, we can create a stationary, multi-scale representation of an input image that captures its texture information but not its global arrangement. To visualize this representation, we can generate an image that matches the style of the input by minimizing the mean-squared distance between the original image's Gram matrices and those of a white noise image using gradient descent.

The contribution of layer $l$ to the total loss is:

$$E_l = \frac{1}{4N_l^2 M_l^2}\sum_{ij}((G^l_{ij} - A^l_{ij})^2) \tag{4}$$

and the total style loss becomes:

$$\mathcal{L}_{style}(\vec{a},\vec{x}) = \sum_{l=0}^{L} w_l E_l \tag{5}$$

where $\vec{a},\vec{x}$ are original and generated image and $w_l$ are weighting factors of the contribution of each layer to the total loss.

The derivative of $E_l$ w.r.t. the activations in layer $l$ can be computed analytically, which means that gradients of $E_l$ can be computed by standard error back-propagation method.

$$\frac{\partial E_l}{\partial F^l_{ij}} = \begin{cases} \frac{1}{N_l^2 M_l^2}((F^l)^T(G^l - A^l))_{ji} & \text{if } F^l_{ij} > 0. \\ 0 & \text{if } F^l_{ij} < 0. \end{cases} \tag{6}$$

**Style transfer** To transfer an artwork's style onto a photograph, we synthesize a new image that matches both the content and style representations. We minimize a loss function that weights the distance of the feature representations of a white noise image from the content and style representations, with $\alpha$ and $\beta$ as weighting factors.

$$\mathcal{L}_{total}(\vec{p},\vec{a},\vec{x}) = \alpha\mathcal{L}_{content}(\vec{p},\vec{x}) + \beta\mathcal{L}_{style}(\vec{a},\vec{x}) \tag{7}$$

We use the L-BFGS algorithm to optimize the gradient with respect to the pixel values. We resize the style image to match the content image before computing its feature representations. We do not regularize with image priors, but lower-layer texture features could act as a specific image prior for the style image. Differences in image synthesis are expected due to the network architecture and optimization algorithm used.

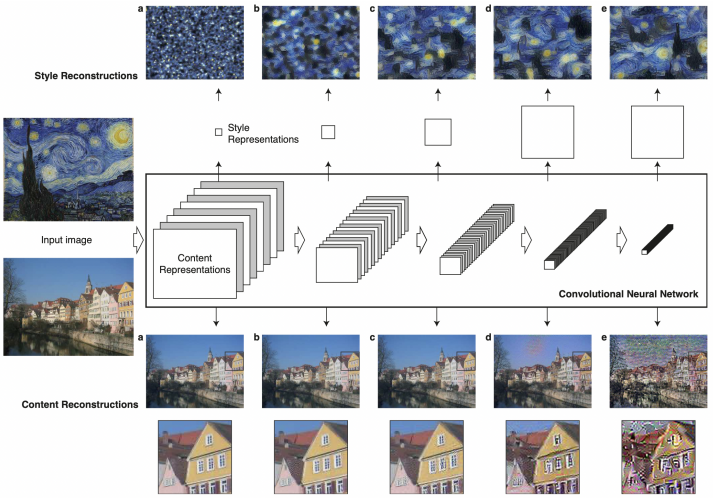

Figure 1: The overall style transfer algorithm.

This is an extended abstract. The full paper is available at the Image Style Transfer Using Convolutional Neural Networks, Perceptual Losses for Real-Time Style Transfer and Super-Resolution.

## 2.2 Perceptual Losses for Real-Time Style Transfer and SR

The system in this research contains of two components: *image transformation network* $f_W$ and a *loss network* $\phi$ that is used for defining several loss functions $l_1, ..., l_k$, which is shown in Figure 2.

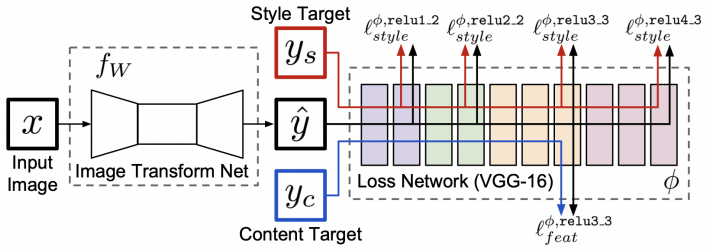


Figure 2: The overall system of a perceptual loss network.

**Loss Network** Several loss functions $l_1, ..., l_k$ are defined by a loss network $\phi$, where each of them computes a scalar value $l_i(\hat{y}, y_i)$ which determines the difference between output image $\hat{y}$ and a target image $y_i$. The loss network makes use of a fixed VGG-16 network pretrained for image classification, since it already learned to encode the perceptual and semantic information we would like to measure in our loss functions. The loss network $\phi$ defines a *feature reconstruction loss* and a *style reconstruction loss* that measure differences in content and style between images.

**Image Transformation Network** An image transformation network $f_W$ is a deep residual convolutional neural network parameterized by weights $W$, in which the output image $\hat{y}$ can be represented as $\hat{y} = f_W(x)$. The network is trained with stochastic gradient descent to minimize a weighted combination of loss functions as:

$$W^* = argmin_W \mathbf{E}_{x,\{y_i\}}[\sum_{i=1} \lambda_i l_i(f_W(x), y_i)] \quad (8)$$

The network consists of five residual blocks [11] with each containing two $3 \times 3$ convolutional layers with modifications at [10] applied, which enables the network an appealing property to learn the identity function more easily.

**Feature Reconstruction Loss** We use a loss network $\phi$ to encourage the feature representations of the output image $\hat{y} = f_W(x)$ to be similar to those of the target image $y$, rather than matching the pixels exactly. The feature reconstruction loss measures the squared, normalized Euclidean distance between feature representations at layer $j$ of network $\phi$. Lower layer feature reconstruction produces visually indistinguishable images from the target, while higher layers preserve spatial structure but not color, texture, or exact shape. Using feature reconstruction loss encourages perceptual similarity between $\hat{y}$ and $y$, but not exact matching.

$$\ell^{\phi,j}_{feat}(\hat{y}, y) = \frac{1}{C_j H_j W_j} \|\phi_j(\hat{y}) - \phi_j(y)\|_2^2$$

**Style Reconstruction Loss** The feature reconstruction loss penalizes differences in content between the output and target images, while the style reconstruction loss penalizes differences in style such as colors, textures, and patterns. To achieve this, Gatys et al [7, 8] propose computing the Gram matrix of the feature maps at each layer, which captures information about which features activate together.

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

The style reconstruction loss is then defined by the squared Frobenius norm of the difference between the Gram matrices of the output and target images.

$$\ell^{\phi,j}_{style}(\hat{y}, y) = \|G^{\phi}_j(\hat{y}) - G^{\phi}_j(y)\|_F^2.$$

Reconstructing from higher layers transfers larger-scale structure from the target image. To perform style reconstruction from a set of layers $J$, we sum the losses for each layer $j \in J$.

**Simple Loss Functions** Researchers also defined two simple loss functions which depend on only low-level pixel information. Pixel Loss measures the normalized Euclidean distance between the output image $\hat{y}$ and the target $y$. It's defined as $l_{pixel}(\hat{y}, y) = ||\hat{y} - y||_2^2 / CHW$ and requires a ground-truth target $y$ for comparison. Total Variation Regularization, $l_{TV}(\hat{y})$, is used to encourage spatial smoothness in the output image, building upon previous work in feature inversion and super-resolution.

# 3 Results and Experiments

## 3.1 Image Style Transfer Using CNN

As we can see in Figure 3, there was a certain trade-off between the accuracy in content and style, following the ratio of $\alpha$ over $\beta$ in computing the total loss $L_{total} = \alpha L_{content} + \beta L_{style}$.

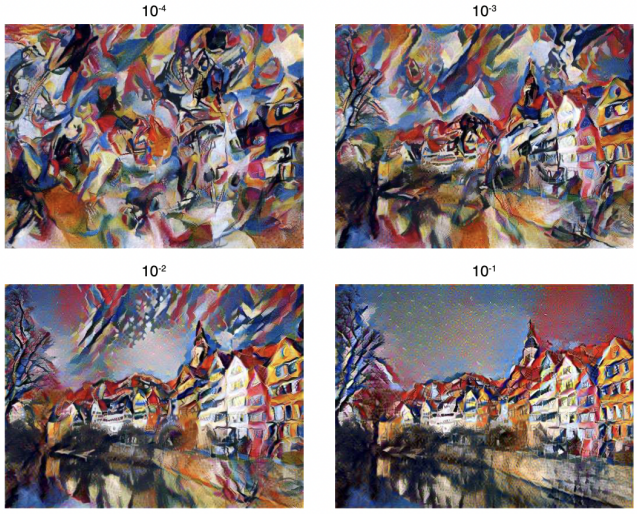


Figure 3: Relative weighting of matching content and style of the respective source images: ratio of $\alpha/\beta$ increases from top left to the bottom right.

Also, researchres observed that the effect of matching the content representation in different layers of the network turns out to be diverse, where the 'conv2_2' layer preserves more finer structural details, but meanwhile the content and the texture seems to merge in the result of matching the content in 'conv4_2' layer.

Moreover, the team have found out that initializing the image with white noise is not necessary, hence either content image or style image can be used for initializing, while arbitrary new images can be generated only when initialized by white noise. Finally, researchers found some limitations in style transferring in photorealistic images, where the resolution of the style image is relatively higher.

## 3.2 Perceptual Losses for Real-Time Style Transfer and SR

**Style Transfer** Feature reconstruction loss is computed at layer relu2_2 and style reconstruction loss is computed at layers relu1_2, relu2_2, relu3_3, and relu4_3 of the VGG-16 loss network. The trained style transfer network has an understanding of the semantic content of images, as evident from the transformed images of a beach and a cat, where the objects like people and animals are recognizable but the backgrounds and bodies are not. This could be because the VGG-16 loss network has features that are selective for such objects since they are part of the classification dataset on which it was trained. The proposed method is much faster than Image Style Transfer, with the ability to process 512x512 sized images at 20 FPS, which makes it suitable for real-time or video applications.


Figure 4: Comparison between the style transfer results of Gatys *et al* and the proposed image transformation network. The network shows much faster generating speed.

**Single Image Super Resolution** Researchers used the baseline model as SRCNN [2], while training their image transformation networks using both per-pixel loss $l_pixel$ and the feature loss $l_feat$. They found out that the proposed method obtains lower PSNR and SSIM [18], but with more pleasant images than SRCNN, since perceptual loss does not optimize PSNR/SSIM using l1/l2 loss, and because the $l_pixel$ loss gives fewer visual artifacts and higher PSNR values but the $l_feat$ loss does a better job at reconstructing fine details, leading to pleasing visual results.

[1] Zezhou Cheng, Qingxiong Yang, and Bin Sheng. Deep colorization. In *Proceedings of the IEEE international conference on computer vision*, pages 415–423, 2015.

[2] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):295–307, 2015.

[3] Alexei A Efros and William T Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 341–346, 2001.

[4] Alexei A Efros and Thomas K Leung. Texture synthesis by non-parametric sampling. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1033–1038. IEEE, 1999.

[5] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE international conference on computer vision*, pages 2650–2658, 2015.

[6] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. *Advances in neural information processing systems*, 27, 2014.

[7] Leon Gatys, Alexander S Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks. *Advances in neural information processing systems*, 28, 2015.

[8] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015.

[9] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016.

[10] Sam Gross and Michael Wilber. Training and investigating residual nets. *Facebook AI Research*, 6(3), 2016.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[12] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, pages 694–711. Springer, 2016.

[13] Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick. Graphcut textures: Image and video synthesis using graph cuts. *Acm transactions on graphics (tog)*, 22(3):277–286, 2003.

[14] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[15] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5188–5196, 2015.

[16] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[17] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[18] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[19] Li-Yi Wei and Marc Levoy. Fast texture synthesis using tree-structured vector quantization. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 479–488, 2000.

[20] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015.