

Introduction to Computer Vision: Style Transfer

Navasardyan Shant



December 3, 2019

- 1 Style Transfer as an Image Optimization Problem
- 2 Style Transfer: One Style Image to Many Content Images
- 3 Style Transfer: Many Style Images to Many Content Images

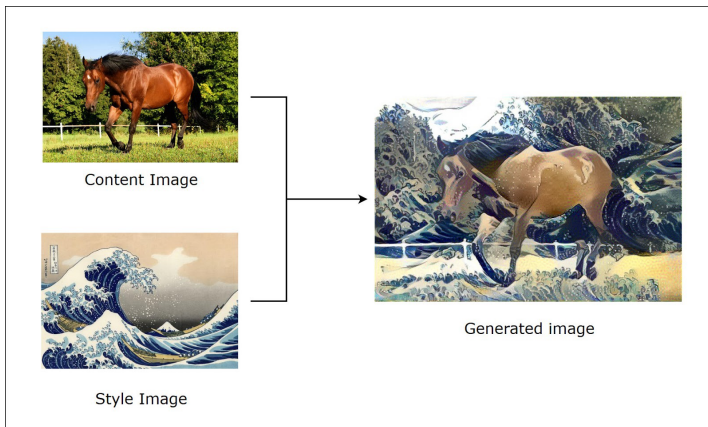
- 1 Style Transfer as an Image Optimization Problem
- 2 Style Transfer: One Style Image to Many Content Images
- 3 Style Transfer: Many Style Images to Many Content Images

Introduction

Style transfer is a process of **stylization** the *given image with an image of the given style*. This is illustrated in the image below.

Introduction

Style transfer is a process of **stylization** the *given image with an image of the given style*. This is illustrated in the image below.



Introduction

So, the style transfer framework takes as input two images, which we call **content** and **style** images, and returns an image with the content from the content image and the style from the style image.

Introduction


So, the style transfer framework takes as input two images, which we call **content** and **style** images, and returns an image with the content from the content image and the style from the style image.



Figure: More examples: the content image - in the up-left position, the style images - at the corners of the results

Style Transfer as an Optimization Problem


Convolutional neural networks, designed for classification tasks, have the ability to describe the semantic content of an image. And how much we go deeper into layers, so much the high-level content is captured.

¹Originally described in the paper: Gatys, L.A., Ecker, A.S., Bethge, M. (2016). Image Style Transfer Using Convolutional Neural Networks. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2414-2423. 

Style Transfer as an Optimization Problem

Convolutional neural networks, designed for classification tasks, have the ability to describe the semantic content of an image. And how much we go deeper into layers, so much the high-level content is captured.

So, in some sense, the content of the image is encoded in the deep layers of classification networks.


¹Originally described in the paper: Gatys, L.A., Ecker, A.S., Bethge, M. (2016). Image Style Transfer Using Convolutional Neural Networks. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2414-2423. 

Style Transfer as an Optimization Problem

Convolutional neural networks, designed for classification tasks, have the ability to describe the semantic content of an image. And how much we go deeper into layers, so much the high-level content is captured.

So, in some sense, the content of the image is encoded in the deep layers of classification networks.

As for styles, it turns out that the *connections* between the feature maps of deep layers describe style features of the image.

¹Originally described in the paper: Gatys, L.A., Ecker, A.S., Bethge, M. (2016). Image Style Transfer Using Convolutional Neural Networks. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2414-2423. 

Style Transfer as an Optimization Problem

Convolutional neural networks, designed for classification tasks, have the ability to describe the semantic content of an image. And how much we go deeper into layers, so much the high-level content is captured.

So, in some sense, the content of the image is encoded in the deep layers of classification networks.

As for styles, it turns out that the *connections* between the feature maps of deep layers describe style features of the image.

The algorithm we describe below¹ uses the classification network *Vgg* – 16 (or *Vgg* – 19) to "extract" the content and style informations from the content and style images respectively and to transfer them into a new image.

¹Originally described in the paper: Gatys, L.A., Ecker, A.S., Bethge, M. (2016). Image Style Transfer Using Convolutional Neural Networks. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2414-2423.

Style Transfer as an Optimization Problem

Let's denote by $S \in \mathbb{R}^{H_s \times W_s \times 3}$ *the style image*, by $C \in \mathbb{R}^{H_c \times W_c \times 3}$ *the content image*. Then we want to *generate* an image G , which has the content of C and the style of S .

Style Transfer as an Optimization Problem

Let's denote by $S \in \mathbb{R}^{H_s \times W_s \times 3}$ *the style image*, by $C \in \mathbb{R}^{H_c \times W_c \times 3}$ *the content image*. Then we want to *generate* an image G , which has the content of C and the style of S .

Note

Here we allow the style and content images to have different sizes. However, the output image G will have the same size as the content image C , i.e. $G \in \mathbb{R}^{H_c \times W_c \times 3}$.

Style Transfer as an Optimization Problem

The approach of generating such image G is the following.

²In the original paper, authors have taken the ratio $\frac{\alpha}{\beta}$ is equal to 10^{-3} and 10^{-4}

Style Transfer as an Optimization Problem

The approach of generating such image G is the following.

The Main Idea

We take a **randomly initialized** image $G \in \mathbb{R}^{H_c \times W_c \times 3}$ and optimize it to be *similar* to the content image C by its content and to the style image S by its style. This optimization is just minimizing a cost function (we will give its detailed description later)

$$L = L(G, C, S)$$

w.r.t. the elements of $G = (g_{i,j,k})^{H_c \times W_c \times 3}$.

²In the original paper, authors have taken the ratio $\frac{\alpha}{\beta}$ is equal to 10^{-3} and 10^{-4}

Style Transfer as an Optimization Problem

The approach of generating such image G is the following.

The Main Idea

We take a **randomly initialized** image $G \in \mathbb{R}^{H_c \times W_c \times 3}$ and optimize it to be *similar* to the content image C by its content and to the style image S by its style. This optimization is just minimizing a cost function (we will give its detailed description later)

$$L = L(G, C, S)$$

w.r.t. the elements of $G = (g_{i,j,k})^{H_c \times W_c \times 3}$.

The cost function L is composed of two component, namely the content components, which we call **the content loss** and the style component, which we call **the style loss**. L is a weighted sum² of this two losses:

$$L = \alpha L_{\text{content}} + \beta L_{\text{style}}.$$

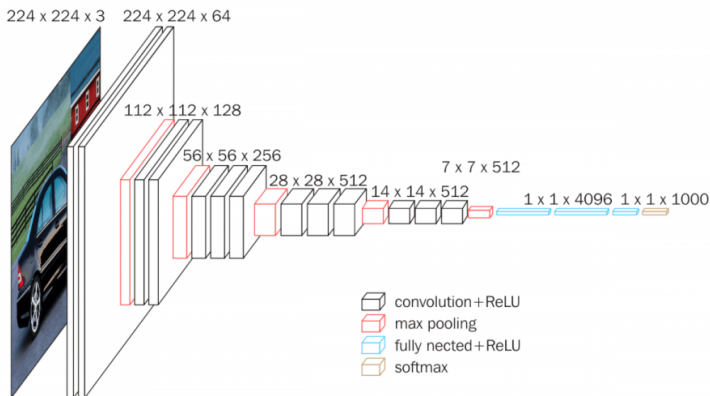
²In the original paper, authors have taken the ratio $\frac{\alpha}{\beta}$ is equal to 10^{-3} and 10^{-4}

Style and Content Losses via Vgg Network

For defining the content and style losses, we need the feature representations of content, style and generated (initially filled with random values) images. These representations are done with the *Vgg* networks (*Vgg* – 16 or *Vgg* – 19).

Style and Content Losses via Vgg Network

For defining the content and style losses, we need the feature representations of content, style and generated (initially filled with random values) images. These representations are done with the Vgg networks (*Vgg* – 16 or *Vgg* – 19). Recall the *Vgg* – 16 architecture:



The Content Loss

Let's consider the l^{th} layer of the network. Let C^l and G^l be the tensors which are obtained from l^{th} layer of *Vgg* network by passing the images C and G respectively. Since C and G has the same size, the tensors C^l and G^l also will have the same size, assume $C^l, G^l \in \mathbb{R}^{H^l \times W^l \times D^l}$ (here D^l is the number of channels of the l^{th} layer of the network). Then we can define³

³In the original paper there is the coefficient $\frac{1}{2}$.

The Content Loss

Let's consider the l^{th} layer of the network. Let C^l and G^l be the tensors which are obtained from l^{th} layer of Vgg network by passing the images C and G respectively. Since C and G has the same size, the tensors C^l and G^l also will have the same size, assume $C^l, G^l \in \mathbb{R}^{H^l \times W^l \times D^l}$ (here D^l is the number of channels of the l^{th} layer of the network). Then we can define³

$$L_{content} = L_{content}(G, C, l) = \sum_{i,j,k} (C_{ijk}^l - G_{ijk}^l)^2.$$

³In the original paper there is the coefficient $\frac{1}{2}$.

The Content Loss

Let's consider the l^{th} layer of the network. Let C^l and G^l be the tensors which are obtained from l^{th} layer of Vgg network by passing the images C and G respectively. Since C and G has the same size, the tensors C^l and G^l also will have the same size, assume $C^l, G^l \in \mathbb{R}^{H^l \times W^l \times D^l}$ (here D^l is the number of channels of the l^{th} layer of the network). Then we can define³

$$L_{content} = L_{content}(G, C, l) = \sum_{i,j,k} (C_{ijk}^l - G_{ijk}^l)^2.$$

Definition (Content Loss)

$L_{content}$ is called **the content loss** at the l^{th} layer. More often the `relu_3_3` is taken (the relu after 7th convolution).

³In the original paper there is the coefficient $\frac{1}{2}$.

The Content Loss

Let's consider the l^{th} layer of the network. Let C^l and G^l be the tensors which are obtained from l^{th} layer of Vgg network by passing the images C and G respectively. Since C and G has the same size, the tensors C^l and G^l also will have the same size, assume $C^l, G^l \in \mathbb{R}^{H^l \times W^l \times D^l}$ (here D^l is the number of channels of the l^{th} layer of the network). Then we can define³

$$L_{content} = L_{content}(G, C, l) = \sum_{i,j,k} (C_{ijk}^l - G_{ijk}^l)^2.$$

Definition (Content Loss)

$L_{content}$ is called **the content loss** at the l^{th} layer. More often the `relu_3_3` is taken (the relu after 7th convolution).

So the content loss is responsible for capturing the content from the content image.

³In the original paper there is the coefficient $\frac{1}{2}$.

The Style Loss

As we have already mentioned, for capturing the style we will consider the "connections" between channels of the network. These "connections" are given by the gram matrices of these layers.

The Style Loss

As we have already mentioned, for capturing the style we will consider the "connections" between channels of the network. These "connections" are given by the gram matrices of these layers. More precisely, let us consider the l^{th} layer of the network. Let S^l and G^l be the tensors which are obtained from l^{th} layer of the network by passing the images S and G respectively. Recall that the sizes of S^l and G^l can be different. Assume that $S^l \in \mathbb{R}^{H_s^l \times W_s^l \times D^l}$ and $G^l \in \mathbb{R}^{H^l \times W^l \times D^l}$.

The Style Loss

As we have already mentioned, for capturing the style we will consider the "connections" between channels of the network. These "connections" are given by the gram matrices of these layers. More precisely, let us consider the l^{th} layer of the network. Let S^l and G^l be the tensors which are obtained from l^{th} layer of the network by passing the images S and G respectively. Recall that the sizes of S^l and G^l can be different. Assume that $S^l \in \mathbb{R}^{H_s^l \times W_s^l \times D^l}$ and $G^l \in \mathbb{R}^{H^l \times W^l \times D^l}$.

Question

Why the resolutions of the tensors S^l and G^l can be different but the numbers of their feature maps are the same?

The Style Loss

As we have already mentioned, for capturing the style we will consider the "connections" between channels of the network. These "connections" are given by the gram matrices of these layers. More precisely, let us consider the l^{th} layer of the network. Let S^l and G^l be the tensors which are obtained from l^{th} layer of the network by passing the images S and G respectively. Recall that the sizes of S^l and G^l can be different. Assume that $S^l \in \mathbb{R}^{H_s^l \times W_s^l \times D^l}$ and $G^l \in \mathbb{R}^{H^l \times W^l \times D^l}$.

Question

Why the resolutions of the tensors S^l and G^l can be different but the numbers of their feature maps are the same?

We can reshape each channel of S^l and G^l and make these 2-dimensional arrays 1-dimensional, then combine them into a matrix. As a result we will get two matrices $\hat{S}^l \in \mathbb{R}^{H_s^l \cdot W_s^l \times D^l}$ and $\hat{G}^l \in \mathbb{R}^{H^l \cdot W^l \times D^l}$.

The Style Loss

Now let's define the gram matrix of a matrix $A \in \mathbb{R}^{N \times M}$. Let's denote the i^{th} column of the matrix A by A^i .

⁴In the original paper there is also the factor $1/(H^i \cdot W^i)^2$.

The Style Loss

Now let's define the gram matrix of a matrix $A \in \mathbb{R}^{N \times M}$. Let's denote the i^{th} column of the matrix A by A^i .

Definition (Gram Matrix)

The matrix $G(A) = (\langle A^i, A^j \rangle)_{i,j=1}^M$ is called the **gram matrix** of A , where $\langle x, y \rangle$ is the dot product of the vectors x and y .

⁴In the original paper there is also the factor $1/(H^l \cdot W^l)^2$.

The Style Loss

Now let's define the gram matrix of a matrix $A \in \mathbb{R}^{N \times M}$. Let's denote the i^{th} column of the matrix A by A^i .

Definition (Gram Matrix)

The matrix $G(A) = (< A^i, A^j >)_{i,j=1}^M$ is called the **gram matrix** of A , where $< x, y >$ is the dot product of the vectors x and y .

Note

The gram matrix $G(A)$ is a square-matrix of size $M \times M$ and its size does not depend on N (the number of the rows of A).

⁴In the original paper there is also the factor $1/(H' \cdot W')^2$.

The Style Loss

Now let's define the gram matrix of a matrix $A \in \mathbb{R}^{N \times M}$. Let's denote the i^{th} column of the matrix A by A^i .

Definition (Gram Matrix)

The matrix $G(A) = (< A^i, A^j >)_{i,j=1}^M$ is called the **gram matrix** of A , where $< x, y >$ is the dot product of the vectors x and y .

Note

The gram matrix $G(A)$ is a square-matrix of size $M \times M$ and its size does not depend on N (the number of the rows of A).

Let's consider the gram matrices $G(\hat{S}^l)$ and $G(\hat{G}^l)$, and define the style loss at l^{th} layer as follows⁴.

⁴In the original paper there is also the factor $1/(H^l \cdot W^l)^2$.

The Style Loss

Now let's define the gram matrix of a matrix $A \in \mathbb{R}^{N \times M}$. Let's denote the i^{th} column of the matrix A by A^i .

Definition (Gram Matrix)

The matrix $G(A) = (\langle A^i, A^j \rangle)_{i,j=1}^M$ is called the **gram matrix** of A , where $\langle x, y \rangle$ is the dot product of the vectors x and y .

Note

The gram matrix $G(A)$ is a square-matrix of size $M \times M$ and its size does not depend on N (the number of the rows of A).

Let's consider the gram matrices $G(\hat{S}^l)$ and $G(\hat{G}^l)$, and define the style loss at l^{th} layer as follows⁴.

$$L_{style}(S, G, l) = \frac{1}{4(D^l)^2} \|G(\hat{S}^l) - G(\hat{G}^l)\|_F^2 = \frac{1}{4(D^l)^2} \sum_{ij} (G(\hat{S}^l)_{ij} - G(\hat{G}^l)_{ij})^2.$$

⁴In the original paper there is also the factor $1/(H^l \cdot W^l)^2$.

The Style Loss

Finally, we define the complete **style loss** as follows:

$$L_{style} = L_{style}(S, G) = \sum_{l=1}^L \omega_l L_{style}(S, G, l),$$

where L is the number of layers of Vgg network, and ω_l are hyper-parameters.

The Style Loss

Finally, we define the complete **style loss** as follows:

$$L_{style} = L_{style}(S, G) = \sum_{l=1}^L \omega_l L_{style}(S, G, l),$$

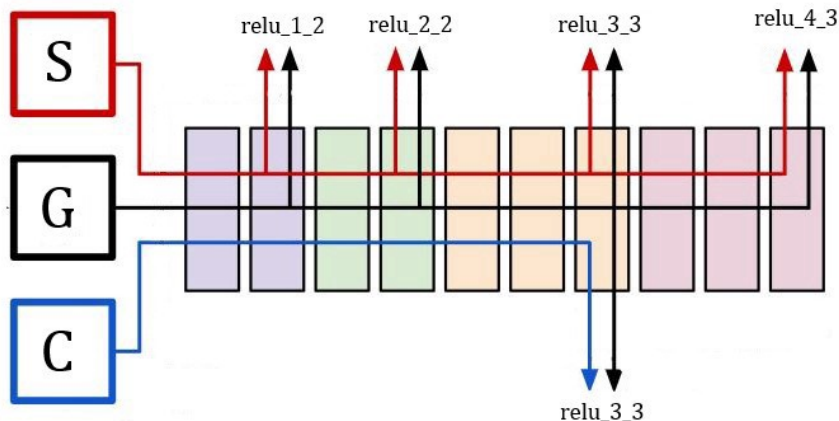
where L is the number of layers of Vgg network, and ω_l are hyper-parameters.

Conclusion

So, in order to generate the image G , we initialize it randomly, then minimize the cost function $L = \alpha L_{content} + \beta L_{style}$ w.r.t. G .

Style Transfer as an Optimization Problem

Here is the illustration of the style transfer algorithm described above:



- 1 Style Transfer as an Image Optimization Problem
- 2 Style Transfer: One Style Image to Many Content Images
- 3 Style Transfer: Many Style Images to Many Content Images

Training a Neural Network for Style Transfer

As you can notice, a drawback of the style transfer method described above is the necessity of optimizing the cost function for **a fixed pair of images: style and content**. When one wants to apply a particular style to many images, this kind of optimization problem becomes computationally heavy to solve. Hence more efficient style transfer algorithms are devised.

⁵Johnson, J.M., Alahi, A., Fei-Fei, L. (2016). Perceptual Losses for Real-Time Style Transfer and Super-Resolution. ECCV.

Training a Neural Network for Style Transfer

As you can notice, a drawback of the style transfer method described above is the necessity of optimizing the cost function for **a fixed pair of images: style and content**. When one wants to apply a particular style to many images, this kind of optimization problem becomes computationally heavy to solve. Hence more efficient style transfer algorithms are devised. The following algorithm⁵ is designed to deal with the cases, when the fixed style image is given, and we want to efficiently transfer this style to any content image with a **feed-forward function**.

⁵Johnson, J.M., Alahi, A., Fei-Fei, L. (2016). Perceptual Losses for Real-Time Style Transfer and Super-Resolution. ECCV.

Training a Neural Network for Style Transfer

The main idea of the algorithm is to generate a **stylized** image G by feeding the **content** image C to a **convolutional network** instead of directly optimizing a cost function w.r.t. G .

Training a Neural Network for Style Transfer

The main idea of the algorithm is to generate a **stylized** image G by feeding the **content** image C to a **convolutional network** instead of directly optimizing a cost function w.r.t. G .

This enables us to **once** train a network (so to get its weights and obtain the above-mentioned *feed-forward function*) and each time feed a content image to it and obtain a generated stylized image.

Training a Neural Network for Style Transfer

Here is an illustration of this method (training and inference):

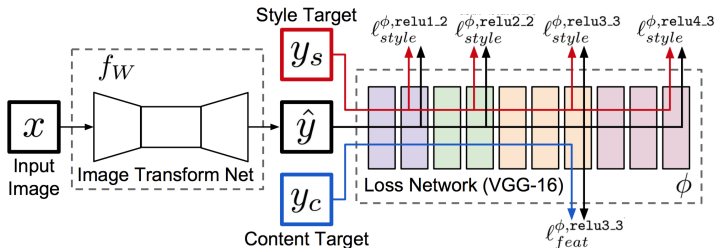


Figure: In the case of style transfer the input image x coincides with the content target y_c .

Training a Neural Network for Style Transfer

Here is an illustration of this method (training and inference):

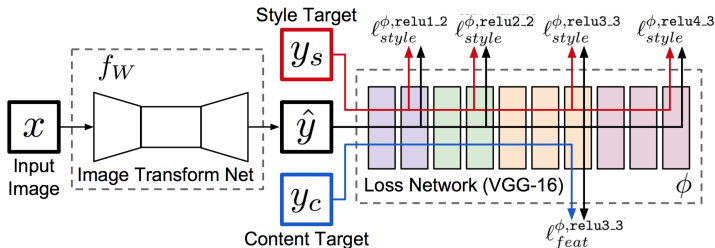


Figure: In the case of style transfer the input image x coincides with the content target y_c .

The *Image Transform Net* is a network which outputs any input image x to a \hat{y} - stylized version of the x with the style image y_s .

Emphasize that the style target y_s is fixed during the training (hence, also during the inference).

Training the Image Transform Network

For training the *Image Transform Network* we need

- a dataset to train on,

Training the Image Transform Network

For training the *Image Transform Network* we need

- a dataset to train on,
- a cost function, which will be optimized w.r.t. the weights W of the network

Training the Image Transform Network

For training the *Image Transform Network* we need

- a dataset to train on,
- a cost function, which will be optimized w.r.t. the weights W of the network

Note

For this algorithm we need a dataset just composed of many images. No labels are required. These images will serve as content images. As for style image, it is fixed during the whole training process.

Training the Image Transform Network

For training the *Image Transform Network* we need

- a dataset to train on,
- a cost function, which will be optimized w.r.t. the weights W of the network

Note

For this algorithm we need a dataset just composed of many images. No labels are required. These images will serve as content images. As for style image, it is fixed during the whole training process.

In the original paper, authors have taken the images from the *MS COCO* dataset.

We train the *Image Transform Net* by minimizing the same loss as in the previous method, but w.r.t. the weights W :

Training the Image Transform Network

For training the *Image Transform Network* we need

- a dataset to train on,
- a cost function, which will be optimized w.r.t. the weights W of the network

Note

For this algorithm we need a dataset just composed of many images. No labels are required. These images will serve as content images. As for style image, it is fixed during the whole training process.

In the original paper, authors have taken the images from the *MS COCO* dataset.

We train the *Image Transform Net* by minimizing the same loss as in the previous method, but w.r.t. the weights W :

$$L^W(y_c, y_s, \hat{y}) = \alpha L_{content}^W(y_c, \hat{y}) + \beta L_{style}^W(y_s, \hat{y}),$$

where W are the weights of the *Image Transform Network*.

Training the Image Transform Network

More precisely, our loss function is

$$L^W = \frac{\alpha}{N} \sum_{x \in \text{dataset}} L_{\text{content}}^W(x, \hat{y}) + \frac{\beta}{N} \sum_{x \in \text{dataset}} L_{\text{style}}^W(y_s, \hat{y}),$$

where $\hat{y} = f_W(x)$ and N is the number of the elements in the training dataset.

Training the Image Transform Network

More precisely, our loss function is

$$L^W = \frac{\alpha}{N} \sum_{x \in \text{dataset}} L_{\text{content}}^W(x, \hat{y}) + \frac{\beta}{N} \sum_{x \in \text{dataset}} L_{\text{style}}^W(y_s, \hat{y}),$$

where $\hat{y} = f_W(x)$ and N is the number of the elements in the training dataset.

Note

During the whole training the Vgg-network remains fixed.

Training the Image Transform Network

More precisely, our loss function is

$$L^W = \frac{\alpha}{N} \sum_{x \in \text{dataset}} L_{\text{content}}^W(x, \hat{y}) + \frac{\beta}{N} \sum_{x \in \text{dataset}} L_{\text{style}}^W(y_s, \hat{y}),$$

where $\hat{y} = f_W(x)$ and N is the number of the elements in the training dataset.

Note

During the whole training the *Vgg*-network remains fixed.

Question

What if we train also the *Vgg*-network?

- 1 Style Transfer as an Image Optimization Problem
- 2 Style Transfer: One Style Image to Many Content Images
- 3 Style Transfer: Many Style Images to Many Content Images

The Case of Many Style Images

We have studied style transfer methods in the cases when either the style image or both style and content images are fixed. So the question arises: how we can generalize the algorithm as a feed-forward approach which takes a pair of images (style and content) and returns the stylized image? In this section we will consider several such approaches.

WCT (Whitening and Coloring Transformation)⁶

The idea of this algorithm is

⁶Li, Yijun, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu and Ming-Hsuan Yang.
“Universal Style Transfer via Feature Transforms.” NIPS (2017).

WCT (Whitening and Coloring Transformation)⁶

The idea of this algorithm is

- to **encode** both the style and the content images into a **feature** space,

⁶Li, Yijun, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu and Ming-Hsuan Yang. “Universal Style Transfer via Feature Transforms.” NIPS (2017).

WCT (Whitening and Coloring Transformation)⁶

The idea of this algorithm is

- to **encode** both the style and the content images into a **feature** space,
- obtain the corresponding feature for the desired stylized image by using the features of the style and the content images and a **transformation** which is called *whitening and coloring transformation*,

⁶Li, Yijun, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu and Ming-Hsuan Yang. “Universal Style Transfer via Feature Transforms.” NIPS (2017).

WCT (Whitening and Coloring Transformation)⁶

The idea of this algorithm is

- to **encode** both the style and the content images into a **feature** space,
- obtain the corresponding feature for the desired stylized image by using the features of the style and the content images and a **transformation** which is called *whitening and coloring transformation*,
- **decode** the obtained feature to get the stylized image.

⁶Li, Yijun, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu and Ming-Hsuan Yang. “Universal Style Transfer via Feature Transforms.” NIPS (2017).

WCT (Whitening and Coloring Transformation)⁶

The idea of this algorithm is

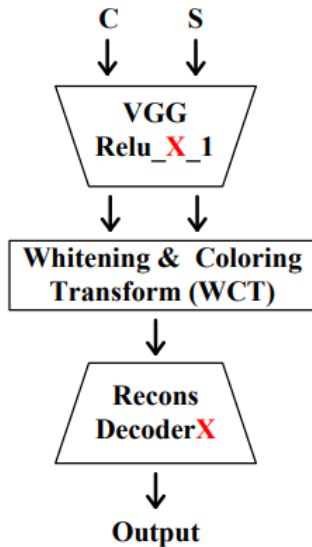
- to **encode** both the style and the content images into a **feature** space,
- obtain the corresponding feature for the desired stylized image by using the features of the style and the content images and a **transformation** which is called *whitening and coloring transformation*,
- **decode** the obtained feature to get the stylized image.

This process is illustrated in the figure below. As earlier C is the content image, S is the style image.

⁶Li, Yijun, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu and Ming-Hsuan Yang. “Universal Style Transfer via Feature Transforms.” NIPS (2017).

WCT (Whitening and Coloring Transformation)

The feature space is the space of the deep features from a particular layer of the *Vgg*-network. Then these features are fed to the transformation *WCT* which gives an output we expect to be the feature of the desired stylized image. Then the obtained feature is fed to the decoder, which returns the corresponding image (from which this feature can be obtained by passing the image to the *Vgg*-network)

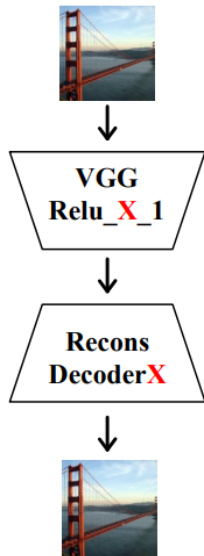


Decoder

In the process described above, one needs a **decoder** to decode back the images from their **deep feature representations**. In other words in the ideal world we want to obtain **the inverse mapping of the Vgg-network's particular layer** (Is it possible?).

For this we simply train a **decoder** for the particular layer of the *Vgg*-network. Decoder is just a deconvolutional neural network, it can be symmetric to the part (untill the particular layer of *Vgg*) of *Vgg*-network, except of the convolutional layers, which can be replaced by transposed-convolutions.

Emphasize that during the training the weights of the *Vgg*-network remain fixed, while the weights of the constructed decoder network are trained to invert the features ov the *Vgg*-network.



The Whitening and Coloring Transformation

The key idea of this algorithm is the transformation called **Whitening and Coloring Transformation** (WCT).

The Whitening and Coloring Transformation

The key idea of this algorithm is the transformation called **Whitening and Coloring Transformation** (WCT). Let's describe the WCT. Let I_C and I_S be a content and a style images respectively. Then let their features, taken from a particular layer of the Vgg-network, are $f_C \in \mathbb{R}^{H_C \times W_C \times C}$ and $f_S \in \mathbb{R}^{H_S \times W_S \times C}$. Let's consider them as vectors, i.e. $f_C \in \mathbb{R}^{H_C W_C \times C}$ and $f_S \in \mathbb{R}^{H_S W_S \times C}$.

The Whitening and Coloring Transformation

The key idea of this algorithm is the transformation called **Whitening and Coloring Transformation** (WCT). Let's describe the WCT. Let I_C and I_S be a content and a style images respectively. Then let their features, taken from a particular layer of the Vgg-network, are $f_C \in \mathbb{R}^{H_C \times W_C \times C}$ and $f_S \in \mathbb{R}^{H_S \times W_S \times C}$. Let's consider them as vectors, i.e. $f_C \in \mathbb{R}^{H_C W_C \times C}$ and $f_S \in \mathbb{R}^{H_S W_S \times C}$. WCT is composed of two parts, namely the whitening transform and the coloring transform.

The Whitening Transform

The whitening transform is done only on the features of the **content** image. The idea of this transform is to "*clear*" these features from their style (then, by the coloring transform, we will add the style of the style image). As the style of an image is characterized by the covariance matrix of deep feature maps of the network, after the whitening transformation the covariance matrix becomes the identity matrix.

The Whitening Transform

The whitening transform is done only on the features of the **content** image. The idea of this transform is to "*clear*" these features from their style (then, by the coloring transform, we will add the style of the style image). As the style of an image is characterized by the covariance matrix of deep feature maps of the network, after the whitening transformation the covariance matrix becomes the identity matrix.

Note

We have defined the style loss on the gram-matrices of the deep feature-maps. Note that in general the gram-matrix is not an estimation of the covariance matrix (when it is?). On the other hand, since $\mathbb{E}[XY] = \mathbb{E}[X]\mathbb{E}[Y] + \text{Cov}(X, Y)$ for any random variables X, Y , we can expect (and it can be shown empirically), that the covariance matrix of deep feature maps also contains an information about the style of the image.

The Whitening Transform

So for the feature $f_C \in \mathbb{R}^{H_C W_C \times C}$ we need to make its covariance matrix the identity (to "clear the style"). The method we use below is very similar to the *PCA* procedure.

The Whitening Transform

So for the feature $f_C \in \mathbb{R}^{H_C W_C \times C}$ we need to make its covariance matrix the identity (to "clear the style"). The method we use below is very similar to the *PCA* procedure.

Recall that *PCA* is a statistical procedure that uses an orthogonal transformation to convert a set of observations into a set of values of uncorrelated variables (isn't it exactly what we want? Why?).

The Whitening Transform

So for the feature $f_C \in \mathbb{R}^{H_C W_C \times C}$ we need to make its covariance matrix the identity (to "clear the style"). The method we use below is very similar to the *PCA* procedure.

Recall that *PCA* is a statistical procedure that uses an orthogonal transformation to convert a set of observations into a set of values of uncorrelated variables (isn't it exactly what we want? Why?).

We first center the f_C by subtracting the mean vector m_C , i.e.
$$f_C = f_C - m_C.$$

The Whitening Transform

So for the feature $f_C \in \mathbb{R}^{H_C W_C \times C}$ we need to make its covariance matrix the identity (to "clear the style"). The method we use below is very similar to the *PCA* procedure.

Recall that *PCA* is a statistical procedure that uses an orthogonal transformation to convert a set of observations into a set of values of uncorrelated variables (isn't it exactly what we want? Why?).

We first center the f_C by subtracting the mean vector m_C , i.e.
$$\tilde{f}_C = f_C - m_C.$$

Similar to the procedure of *PCA*, the whitening transform uses the fact that if the matrix $\tilde{f}_C^T \tilde{f}_C$ is a positive definite matrix, then its Jordan form is

$$\tilde{f}_C^T \tilde{f}_C = E_C^T D_C E_C,$$

where D_C is an invertible diagonal matrix and E_C is an orthogonal matrix.

The Whitening Transform

Definition (Whitening)

The transformation $\mathbb{R}^{H_C W_C \times C} \rightarrow \mathbb{R}^{H_C W_C \times C}$, $f_C \mapsto \hat{f}_C$

$$\hat{f}_C = f_C E_C^T D_C^{-\frac{1}{2}} E_C$$

is called **whitening transformation**.

The Whitening Transform

Definition (Whitening)

The transformation $\mathbb{R}^{H_C W_C \times C} \rightarrow \mathbb{R}^{H_C W_C \times C}$, $f_C \mapsto \hat{f}_C$

$$\hat{f}_C = f_C E_C^T D_C^{-\frac{1}{2}} E_C$$

is called **whitening transformation**.

It can be easily checked that $\hat{f}_C^T \hat{f}_C = I$:

$$\begin{aligned}\hat{f}_C^T \hat{f}_C &= E_C^T D_C^{-\frac{1}{2}} E_C f_C^T f_C E_C^T D_C^{-\frac{1}{2}} E_C = \\ &E_C^T D_C^{-\frac{1}{2}} I D_C I D_C^{-\frac{1}{2}} E_C = E_C^T I E_C = I\end{aligned}$$

The Whitening Transform

To illustrate how the whitening transform works in practice, we pass an image to the encoder, then apply **only** the whitening transform, followed by decoding the obtained feature back to the image space. We expect the output with the content of the input but with "*no style*" (in some sense). Here are some results:



Figure: inverted from the layer ReLU 4_1

The Coloring Transform

The coloring transform takes the output of the whitening transform and the features f_S of the style image and outputs the features f_{CS} of the stylized image.

The Coloring Transform

The coloring transform takes the output of the whitening transform and the features f_S of the style image and outputs the features f_{CS} of the stylized image. First, as in case of the whitening transform, we center the features f_S by subtracting the mean vector m_S , i.e. $f_S = f_S - m_S$.

The Coloring Transform

The coloring transform takes the output of the whitening transform and the features f_S of the style image and outputs the features f_{CS} of the stylized image. First, as in case of the whitening transform, we center the features f_S by subtracting the mean vector m_S , i.e. $f_S = f_S - m_S$. Then we consider the matrices E_S and D_S in the same way as in the whitening transform.

The Coloring Transform

The coloring transform takes the output of the whitening transform and the features f_S of the style image and outputs the features f_{CS} of the stylized image. First, as in case of the whitening transform, we center the features f_S by subtracting the mean vector m_S , i.e. $f_S = f_S - m_S$. Then we consider the matrices E_S and D_S in the same way as in the whitening transform.

Definition (Coloring)

The transformation $\mathbb{R}^{H_C W_C \times C} \rightarrow \mathbb{R}^{H_C W_C \times C}$, $f_C \mapsto \hat{f}_C$

$$\hat{f}_{CS} = \hat{f}_C E_S^T D_S^{\frac{1}{2}} E_S$$

is called **coloring transformation**.

The Coloring Transform

The coloring transform takes the output of the whitening transform and the features f_S of the style image and outputs the features f_{CS} of the stylized image. First, as in case of the whitening transform, we center the features f_S by subtracting the mean vector m_S , i.e. $f_S = f_S - m_S$. Then we consider the matrices E_S and D_S in the same way as in the whitening transform.

Definition (Coloring)

The transformation $\mathbb{R}^{H_C W_C \times C} \rightarrow \mathbb{R}^{H_C W_C \times C}$, $f_C \mapsto \hat{f}_C$

$$\hat{f}_{CS} = \hat{f}_C E_S^T D_S^{\frac{1}{2}} E_S$$

is called **coloring transformation**.

Then we re-center the \hat{f}_{CS} , i.e. $\hat{f}_{CS} = \hat{f}_{CS} + m_S$.

The Whole Pipeline

Question

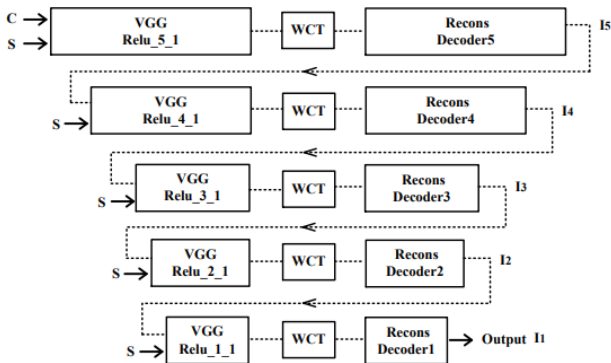
What output will give the whitening and coloring transformation, if the style and the content images are identical?

The Whole Pipeline

Question

What output will give the whitening and coloring transformation, if the style and the content images are identical?

This approach can be improved when we do it for several layers of Vgg-network sequentially, i.e. as described in the image below:



Some Results of WCT approach



Figure: Single-level pipelines for different layers of Vgg-network

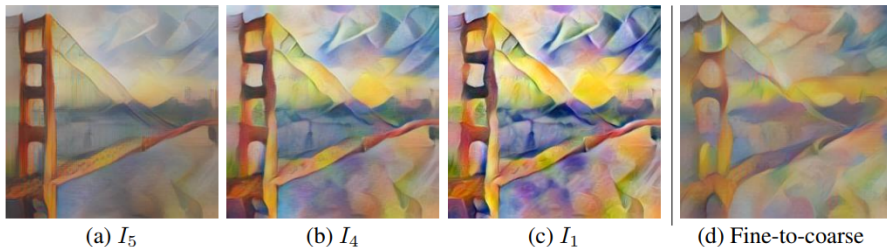


Figure: Intermediate results from the multi-level approach

Introduction to Computer Vision: Style Transfer

Navasardyan Shant



December 3, 2019