Universal Style Transfer via Feature Transforms

Authors: Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang

Presented by: Ibrahim Ahmed and Trevor Chan
Problem

- Transfer arbitrary visual styles to content images

Image source: https://research.googleblog.com/2016/02/exploring-intersection-of-art-and.html
Related Work

1. **Not efficient during inference**: A Neural Algorithm of Artistic Style [1].
2. **Style Specific Networks**: Perceptual Losses for Real-Time Style Transfer and Super-Resolution [2].
3. **Poor generalizing abilities in terms of output quality**: Arbitrary Style Transfer in Real-Time with Instance Normalization[3]
Proposed Method

- Image reconstruction + Feature Transforms

Train autoencoder for image reconstruction, then fix it
Proposed Method

“bottleneck” hidden layer

input layer

output layer
(reconstruction of input layer)

all layers are fully connected but not drawn

Image source: http://nghiaho.com/?p=1765
Proposed Method

- Encoder - Train VGG-19 on Imagenet Classification task
- Decoder - Trained to reconstruct the image
- More than one decoder trained for reconstruction
  - 5 trained decoders

Image source: Li et al.
Loss function for Reconstruction Decoder

\[ L = \| I_o - I_i \|_2^2 + \lambda \| \Phi(I_o) - \Phi(I_i) \|_2^2 \]

- Note: no style image is used in process of training
Feature Transforms
Feature Transforms by Whitening/Coloring

- Content features are transformed at intermediate levels by statistics of the style features.

- In each layer, need content features to exhibit same characteristics of the style features of the same layer.

- WCT achieves this.
Data Whitening

- Transform a random vector to be uncorrelated and have unit variance
  1. decorrelate the components of original vector
  2. scale the different components so they have unit variance
Data Whitening (Step One)

- Eigendecomposition of covariance matrix by $\Sigma = \Phi \Lambda \Phi^{-1}$
- Let $Y = \Phi^T X$ where $Y$ has uncorrelated components

\[
\Sigma \quad \Phi \quad \Lambda \quad \Phi^{-1}
\]

\[
\begin{bmatrix}
\vdots \\
0 \\
\vdots \\
0 \\
\lambda_1 \\
\vdots \\
\lambda_2 \\
\vdots \\
\lambda_3
\end{bmatrix}
\begin{bmatrix}
v_1 \\
v_2 \\
v_3
\end{bmatrix}
\begin{bmatrix}
\lambda_1 & 0 & 0 \\
0 & \lambda_2 & 0 \\
0 & 0 & \lambda_3
\end{bmatrix}
\begin{bmatrix}
v_1 \\
v_2 \\
v_3
\end{bmatrix}^{-1}
\]
Data Whitening (Step Two)

- Scale the uncorrelated components of $Y$ by $W = \Lambda^{-1/2} * Y$
- $W$ is now the white noise vector
- Components are i.i.d with unit variance
- $W$'s covariance matrix is identity matrix
Data Coloring

- Coloring is the inverse of the whitening transform
- Transform white noise into random vector with desired covariance matrix
Applying WCT to Style Transfer

- Disassociate input image style and associate with the input image the style of the style image
- From content image $I_c$ and style image $I_s$, extract their vectorized feature maps $f_c$ and $f_s$
- WCT will directly transform the $f_c$ to match the covariance matrix of $f_s$

Image source: Li et. al
Whitening and Coloring Transform

Whitening:

\[ \hat{f}_c = E_c D_c^{-\frac{1}{2}} E_c^T f_c \]

\[ \hat{f}_c \hat{f}_c^T = I \]

Coloring:

\[ \hat{f}_{cs} = E_s D_s^{\frac{1}{2}} E_s^T \hat{f}_c \]

\[ \hat{f}_{cs} \hat{f}_{cs}^T = f_s f_s^T \]

Image source: Li et. al
Multi-level Stylization

Image source: Li et. al
Multi-level Stylization

Image source: Li et. al
Results

- Compared with other style transfer methods

- Other methods were inferior in terms of
  - Handling arbitrary styles
  - Efficiency
  - Learning-free

Image source: Li et. al
## Results

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Arbitrary</td>
<td>√</td>
<td>√</td>
<td>×</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Efficient</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>×</td>
<td>√</td>
</tr>
<tr>
<td>Learning-free</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>√</td>
<td>√</td>
</tr>
</tbody>
</table>
Parameters

- Style weight control
- Image size

\[ \hat{f}_{cs} = \alpha \hat{f}_{cs} + (1 - \alpha) f_c \]
Image Editing

- Edit different areas of a content image with different style images

Stylization Result

Mask 1

Mask 2

The Scream

Graffiti

Mask 3

Fish art

Image source: Li et. al
Texture Synthesis

- Input noise image as content image, and use texture example as the style image

\[ \hat{f}_{cs} = \beta \hat{f}_{cs1} + (1 - \beta) \hat{f}_{cs2} \]
Takeaways

- Works with any chosen style
- Don’t have to train on style images
- Scale and weight of style transfer can be changed on the fly
- Performs more consistently across a set of widely varying input styles