Data Science and Scientific Computing - University of Trieste
Deep Learning Final Project

Universal Style Transfer via Feature Transforms: an implementation

Andrea Barbieri - Anna Pederzani Samarati

A.Y. 2022/2023

Outline

Introduction:
the Style Transfer technique

What is Style Transfer?

- **Style Transfer** is a computer vision technique used to manipulate a given image adopting the aesthetic of another one.
- The goal is to keep the **content** of the original image, while applying the **visual style** of another image.
- A seminal paper for this study field was *A Neural Algorithm of Artistic Style*, published in 2015 by Gatys et. al., introducing the use of convolutional networks to transfer styles.
- Since then, several models for style transfer have been proposed expanding also in the videos and text domain.

ST applications

Style transfer has some interesting applications:

- **Art generation**
- **Texture Synthesis**: mixing random noise with styles to obtain textures (see Results)
- **Data Augmentation**: use stylized images in both training and test phases can help improve the robustness of a model.[1]

[1] *Style Augmentation: Data Augmentation via Style Randomization*

Approaches to Style Transfer

Given the various models proposed, one could divide them into three main categories:

- **Image optimization**: networks based on *fixed parameters* which create a new image which will be iteratively updated to match respectively content and style.
- **Model optimization**: networks with parameters *tuned* to approximate the image optimization process, usually trained for one or a limited number of styles.
- **Universal Style Transfer**: networks trained for plain image reconstruction, "stylization" is achieved by means of *feature transformation*.

So, why WCT?

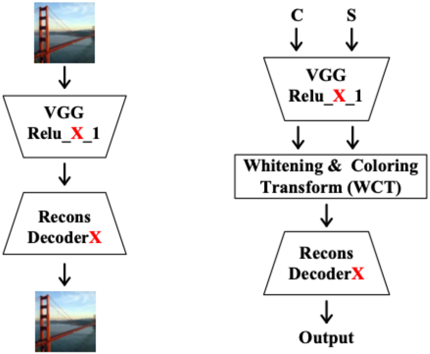
- The model proposed in the paper we chose, *Universal Style Transfer via Features Transforms*, published by Li et al. in 2017 belongs to the latter category presented.
- The underlying idea is to transfer the style by means of **whitening** and **coloring** transformation of both the content and style **latent representations**.
- The main advantage of the model is its **universality**: it is a learning-free scheme in which style is extracted during the coloring transformation and not during training.

WCT and Autoencoders

Proposed model

- The transfer style task is formulated as an **image reconstruction process** coupled with **feature transformation**.
- The model is composed by an **autoencoder**: the encoder resembles the VGG-19 network and the decoder is symmetric to the encoder.
- The feature transformation is performed in the **latent space**.

Proposed model



Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang, "Universal style transfer via feature transforms, " in Advances in Neural Information Processing Systems, 2017, pp. 386–396.

Loss function

$$L = \textit{reconstruction loss} + \textit{feature loss},$$

$$L = \|I_o - I_i\|_2^2 + \lambda\|\Phi(I_o) - \Phi(I_i)\|_2^2,$$

where I_i is the input image, I_o is the output image and Φ is the VGG encoder.

WCT (I)

Goal of WCT: to transform the content features such that they exhibit the same statistical characteristics as the style features.

- Firstly, the VGG feature maps of the content image and of the style image are extracted.
- Then WCT is applied to content features.
- The transformed features are then fed forward into the decoder.

WCT (II)

- The **whitening step** helps to remove information related to styles, while preserving the global content structure.
- The **coloring step** introduces style to the content image.

Whitening transform

Let f_c be the *centered* vectorized VGG feature map of the content image.

The **whitening transform** linearly transforms f_c in order to obtain an **uncorrelated** feature map $\hat{f}_c$ (i.e. $\mathbb{E}(\hat{f}_c) = \underline{0}$, $\mathbb{V}(\hat{f}_c) = I$).

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

where D_c is the diagonal eigenvalues matrix of $f_c f_c^T$ and E_c is the corresponding orthogonal matrix of eigenvectors.

Coloring transform

Let f_s be the *centered* vectorized VGG feature map of the style image.

The **coloring transform** linearly transforms $\hat{f}_c$ such that we obtain $\hat{f}_{cs}$ which has the **desired correlations** between its feature map ($\hat{f}_{cs}\hat{f}_{cs}^T = f_s f_s^T$).

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

where D_s is the diagonal eigenvalues matrix of $f_s f_s^T$ and E_s is the corresponding orthogonal matrix of eigenvectors.

Finally $\hat{f}_{cs}$ is *re-centered*.

Strength of stylization effects

- Another peculiarity of this model is that it provides **user controls** on the strength of stylization effects.
- Indeed the output of the WCT may be *blended* with the content feature before feeding it to the decoder.
- The blended $\hat{f}_{cs}$ is:

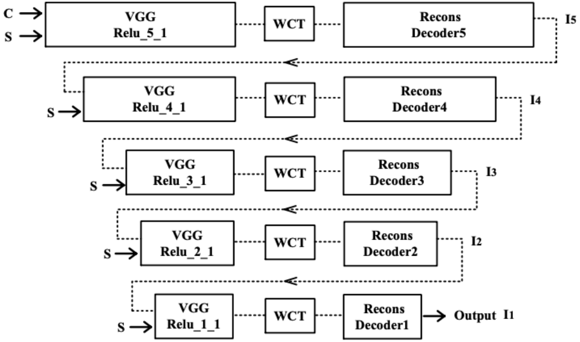
$$\hat{f}_{cs} = \alpha \, \hat{f}_{cs} + (1 - \alpha) \, f_c,$$

where α serves as the style weight for users to control the transfer effect.

WCT multi-level

- The entire model is composed of **5 sub-models**.
- Each sub-model is an autoencoder in which the encoder is made of the VGG architecture up to the **Relu_X_1** layer with X=1,..,5.
- The multi-level stylization is made of a **pipeline of the sub-models**, starting from the largest one to the smallest one one after the other.
- The WCT is applied in each sub-model.

WCT multi-level



Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang, "Universal style transfer via feature transforms, " in Advances in Neural Information Processing Systems, 2017, pp. 386–396.

How we did it: implementation details

Dataset

- The model has been trained using the **COCO dataset** (Common Object in COntext), a collection of images specifics for **object segmentation and recognition**.
- There are multiple COCO iterations published during the years, the last one (2017) counting $\sim$ 330k images.
- Due to the large size of the dataset we have chosen to use a **"small" sample** of it consisting of 10k images, 8k used for training and 2k for testing.
- For the styles, we have used a sample of images taken from WikiArt.

Training

- During the training phase we have first **loaded and fixed the encoders parameters**, leaving only the decoders ones to be optimized.
- Each autoencoder has been trained **separately** using the same training data for 500 epochs and fixed batch size of 128.
- The optimizer used was **ADAM** with default betas and 10^{-4} learning rate.
- Following the paper, we have set the λ parameter of the loss function to 1.

Speed up the training

To speed up the train we have used a couple of tricks, citing the most important:

- Train on **GPU**.
- **Distributed Data Parallelization**: the model was replicated in two GPUs and trained in parallel, the gradient was then averaged and distributed between the two models.
- **Automatic mixed precision training**: use `float16` instead of `float32` type to compute loss, saving space and computation time.

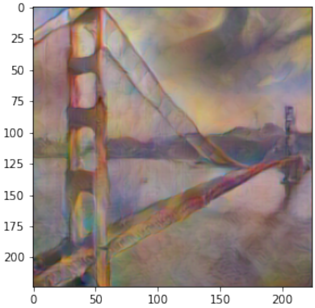
Loss values

Sub-model	Train loss (last epoch)	Test loss
Relu_1_1	1,338	1,494
Relu_2_1	28,854	29,695
Relu_3_1	103,753	108,963
Relu_4_1	220,565	301,522
Relu_5_1	20,111	48,045

Number of epochs = 500

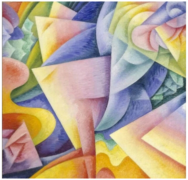
Results

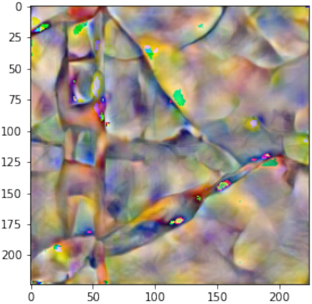
Single-level stylization



Single-level stylization with Relu_4_1 and $\alpha = 0.8$

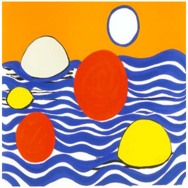
Multi-level stylization

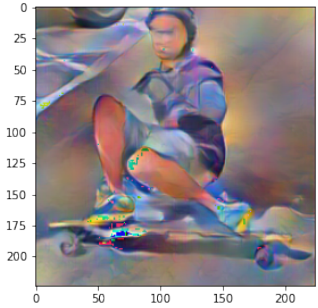




Multi-level stylization with mixed α

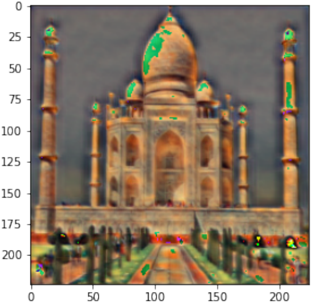
Single-level stylization





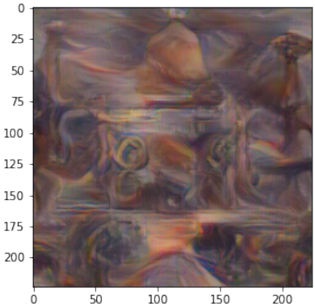
Single-level stylization with Relu_4_1 and $\alpha = 0.8$

Low-depth stylization



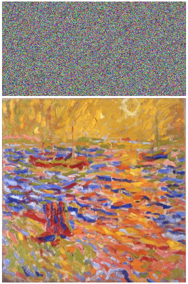
Multi-level stylization with Relu_1_1 and Relu_2_1 and $\alpha = 0.8$

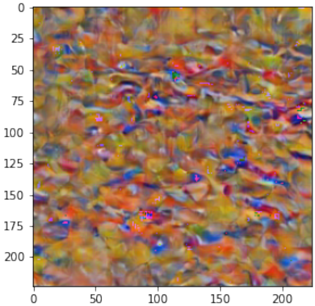
High-depth stylization



Multi-level stylization with Relu_4_1 and Relu_5_1 and $\alpha = 0.8$

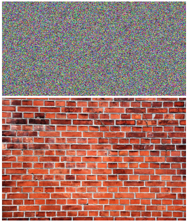
Texture Synthesis (1)

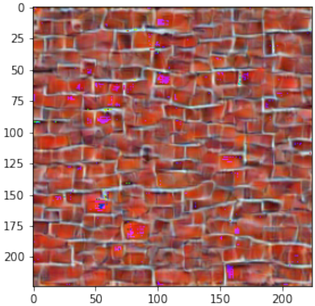




Multi-level texture synthesis with $\alpha = 1$

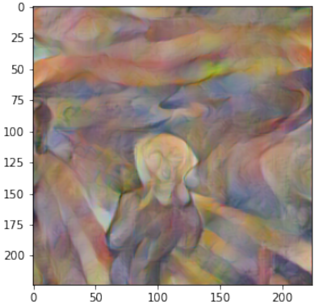
Texture Synthesis (2)





Multi-level texture synthesis with $\alpha = 1$

Mixture of styles



ReLU_4_1 style transfer with $\alpha = 1$

Conclusions

Conclusions

- To perform plain style transfer a good solution is to use single WCT with high alpha (.6 - .8) and large autoencoders (e.g. Relu_4_1).
- For texture synthesis purposes, multi-level WCT with high alphas (.7- 1) can be effectively used.
- Lastly, for data augmentation one can use the same parameters as the plain style transfer.

Further developements

- Due to the poor reconstruction quality color-wise of the models (especially the Relu_1_1) it would be effective to **train the autoencoders with a larger dataset**.
- **Spatial control of stylization**: extend the model in order to apply the style to custom portions of the content image.
- **Generalization to multiple styles**[2]: extend the model introducing multiple styles at the same time in the transformation process.

[2] *Multiple Style Transfer via Variational AutoEncoder*

Thank you for the attention!

And thanks for all the fish!