CS230

David Schmidt-Fellner
Stanford University, GSB MBA2020 - davidsf@stanford.edu

Neural Style Transfer for Low-Complexity Animated Inputs

Abstract

Neural Style Transfer may have promising commercial applications for artists and animators as a story-boarding tool, capable of instantly "stylizing" still-life photography into existing animation styles. However, common techniques and implementations of NST favor styles with more complex textures and color schemes than most animations, which are low-complexity. This paper examines the conventional wisdom that Style Loss should be derived from "deeper" layers in a Convolutional Neural Network. We find that for animated style images, earlier layers should be more highly weighted, and style weight should be greatly reduced relative to content weight.

Introduction

Neural Style Transfer is predominantly used to illustrate properties of Convolutional Neural Networks and provide a semantic definition of content and style. While they help describe the wide applicability of CNNs, NSTs themselves are not currently applied to commercial contexts[i].

However, animators and illustrators who use storyboards and sketches in the early stages of their work may find direct benefit from NST. While elements of style (color choice and texture) are often the final layer of an animator's process, NST allows style to be "borrowed" from other *animated* images and applied to images earlier in the creative process. Style images from animated sources, i.e. cartoons and video games, are predominantly low-granularity and simplistic in their color schemes. These properties can be applied to more nuanced photography (i.e. landscapes and figures), to see useful stylistic renderings earlier in their story-boarding process.

Typical implementations of NST are not always successful on lower-granularity style inputs (i.e. animated ones). Classic implementations involve architectures and best practices for tuning designed for high-granularity color-complex style, e.g. abstract oil paintings[2]. Our hypothesis entering the study is that the typical notion that Style Loss ought to be defined using *later* layers in the CNN may be inapplicable to low-granularity style images.

In this study, we modify a traditional NST which has been pre-trained on a VGG19 ImageNet. We gather our own content images from real-life stills of landscapes and figures, compressed to 400 by 300 pixels and RGB colors. We gather still-frames from various animated sources online, including Japanese animated film, classic video games, and modern cartoons, which are also compressed in advance to the specifications above. We use the NST to generate new stylized images and evaluate the success of each rendering before making changes. We make significant adjustments to how we define Style Loss by applying new weights to different Convolutional Layers of the VGG19. Finally, we tune the implementation using style vs. content parameters and number of iterations.

Related Works

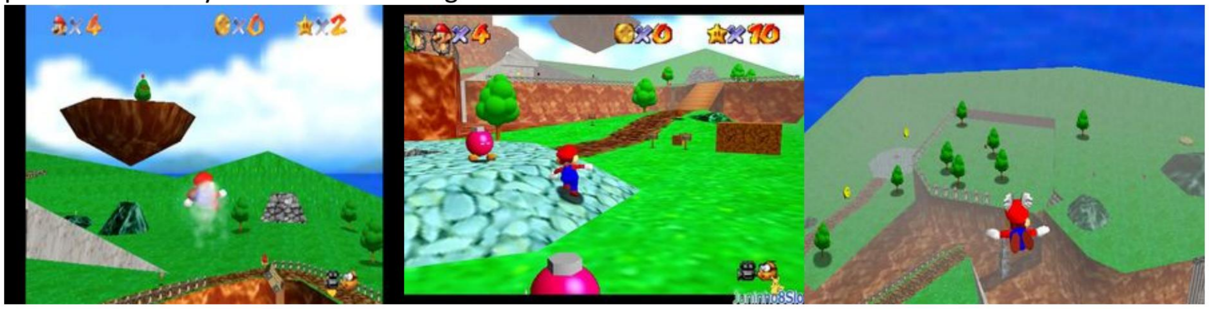
“A Neural Algorithm of Artistic Style” by Gatys, Ecker, Bethge (2015) established NST foundations in providing – primarily by identifying tenable definitions of content and style using CNN architecture. Even in early works, style is emphasized as a trait of “later” layers in a CNN, which this paper will examine. Further work by Gatys, and Leon. A. in “Controlling perceptual factors in neural style transfer" refined techniques to isolate the transfer of color from the transfer of “texture.” While the notion of texture-transfer has existed since 2003 (and has been applied to professional visual tools for animators like Adobe Illustrator and Photoshop), NST marked a significant increase in the thoroughness of image transfer generally. “Deep Photo Style Transfer” made significant strides in avoiding any low-granularity and “painting-like quality” in the generated image, emphasizing hyperrealism, which demonstrates a departure in focus away from applications for animation. While some highly cited papers have applied NST directly to animated style images with nice results (see “Simpsons” example in “Perceptual Losses for Real-Time Style Transfer and Super-Resolution”) (2016), little if any work has focused on this application in particular.

Dataset

Content Dataset: We gathered iPhone images from a personal photo library that we thought would lend themselves well to animators’ creative process. Ultimately, five images total were used – attempting to focus on A) backdrops (i.e. landscapes and city-settings) and B) human figures. Two were of landscape photography of the Scottish Highlands. One was of two figures (my siblings) resting by their bikes in Ireland. One was of two people playing on a rooftop in Lisbon, Portugal. And finally, one overlooked the Lisbon city-scape. All photos were cropped and compressed using a bulk-image modifier online to make sure they were 400 by 300 .jpeg files.

.

Style Dataset: Using Google Image Search, we gathered still images from a mix of low-granularity, animated images. Four came from screenshots of Super Mario 64, a classic Nintendo game released in 1996. Two came from Adventure Time, a modern cartoon series. Two came from Howl’s Moving Castle and The Wind Rises – both acclaimed animated films by Hayao Miyazaki. The style photos were pre-processed exactly as the content images were.



.

Methods

This implementation uses methods from *A Neural Algorithm of Artistic Style by Gatys, Ecker, and Bethge* (2015). The objective of Neural Style Transfer is to generate a new image while minimizing an overall loss function that combines Content Loss and Style Loss using this formula:

$$L_{total} = \beta \times L_{content} + \alpha \times L_{style}$$

β and α are simply weights to balance and tune the relative influence of Content Loss and Style Loss. In this implementation, the Content Loss (L_{content}) is defined according to the following equation:

$$\mathcal{L}_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} \left(F_{ij}^{l} - P_{ij}^{l} \right)^2 .$$

Where p is the original image, and P is its feature representation at layer l. x is the generated content image, and its feature representation at layer l. The only layer used to define Content Loss is convolutional layer 4_2 of VGG19 (the second convolutional layer after 3 batches of RELU and maxpooling), which has a proven record of adequately reflecting content as a standalone layer[1]. Style Loss is defined according to this equation, where w is a manual weight for each layer:

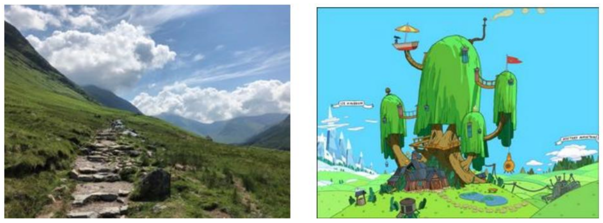
$$\mathcal{L}_{style}(\vec{a}, \vec{x}) = \sum_{l=0}^{L} w_l E_l \qquad E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(G_{ij}^{l} - A_{ij}^{l} \right)^2$$

E is given by the below, where G is a gram matrix (inner product) of the Feature representation of the generated image at a given layer. A is feature rep. of the original style image. The first term is a normalization term where N is the number of filters and M is the size of the filter.

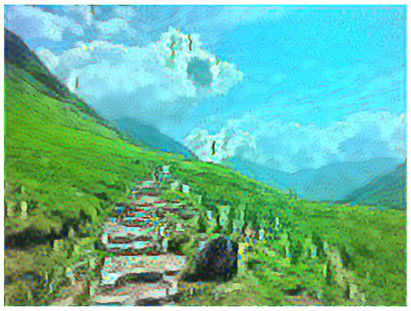
Results

Many permutations for each combination of content and style image were created while adjusting for the manual inputs of the NST: the weight matrix in the style loss (w), and the relative weight of β vs. α. There were two primary findings of the work. Firstly, the typical guidance that the Style Loss weight matrix should favor later convolutional layers over earlier ones did not seem to be true for low-granularity, animated style images.

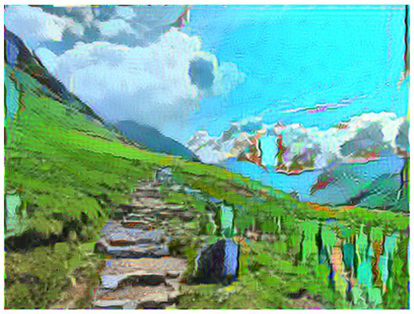
Best practices suggest, for a VGG19 network, applying weights of [.5, 1.0, 1.5, 3.0, 4.0] to convolutional layers [c1_1, c2_1, c3_1, c4_1, c5_1] when defining style loss (Fig (4)). It also recommends flat weighting across all five layers as a baseline of performance (Fig (5)). However, both these weightings performed worse (according to the experimenter's visual /aesthetic judgment) than weightings that favored earlier layers, namely **[0, 5, 5, 0, 0] (Fig (3))**. While this configuration seemed to perform best on average across content / style combinations, simply using the second layer only [0, 10, 0, 0, 0] (Fig (1)) or the third layer [0, 0, 10, 0, 0] (Fig (2)) still outperformed the recommended one. The typical recommendations seem to be failing because the fourth layer, commonly thought to be the most useful in the style loss, seems to perform very poorly on low granularity style images.



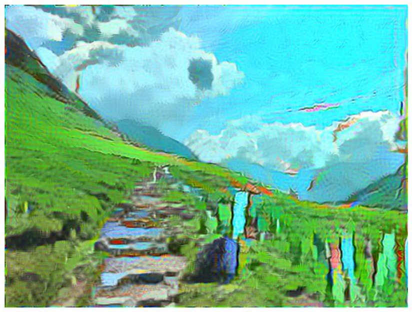
Testing style weightings on content image of Scottish Highlands and screenshot from "Adventure Time"



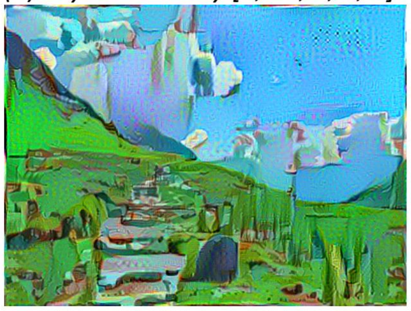
(1) Layer Two Only [0, 10, 0, 0, 0]



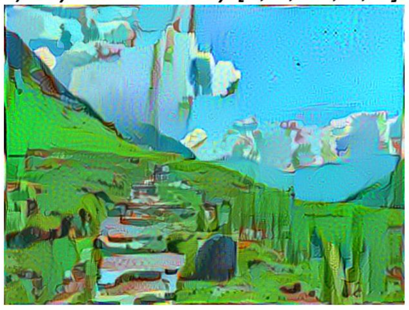
(2) Layer Three Only [0, 0, 10, 0, 0]



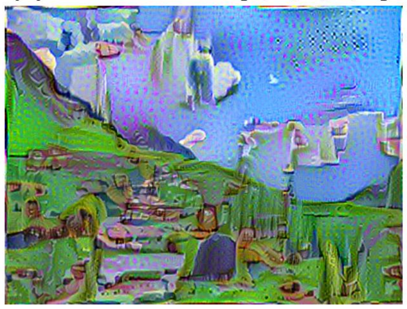
(3) Two and Three [0, 5, 5, 0, 0]



(4) Back-Weighted (Typical) [0, 10, 0, 0, 0]

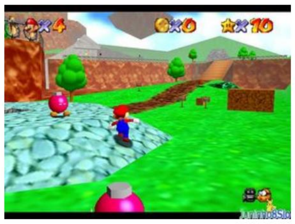


(5) Flat (Typical) [2, 2, 2, 2, 2]

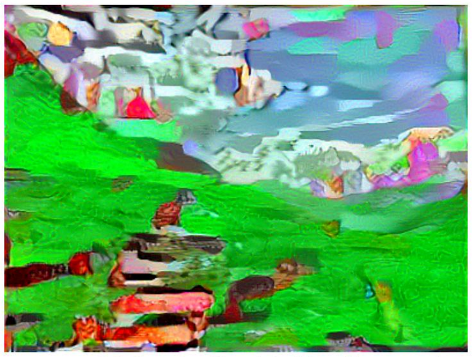


(6) Layer Four Only [0, 0, 0, 10, 0]

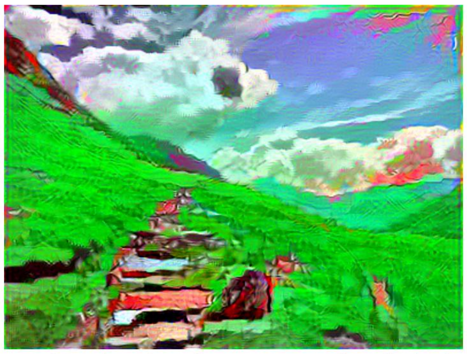
Secondly, these NSTs required an atypically low relative weighting of α vs. β. Style loss from the generated image, which is initiated as a weighted average of a noise image and the content image, is several orders of magnitude higher than content loss when using animated style inputs. When using a typical α like 100[6,7] (fig (7)) while β stays constant at 5, content is highly distorted. Only tuning α down several orders of magnitude produces nice results.



Testing relative alpha weight on content image of Scottish Highlands and screenshot from "Super Mario 64"



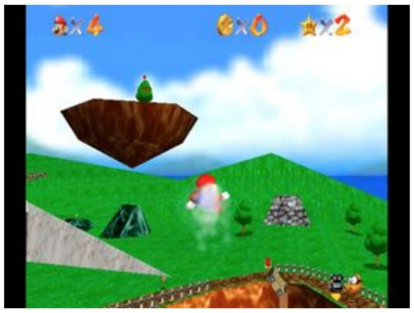
α = 100 (typical) – highly distorted



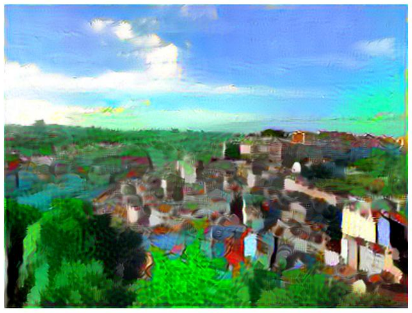
α = .001 (optimal) – content / style balanced

Finally, implementation guidance generally suggests that 5,000 iterations produce significantly better results than 1,000. However, using these images, there was hardly any visible difference between 500 iterations and anything larger (up to 10,000). This is likely a result of the low granularity of the style images, the extreme color palettes, and the low image quality (400 x 300). Below are other nice results using w = [0, 5, 5, 0, 0], *α = .001, iterations = 500:*

Content: Rooftop in Lisbon



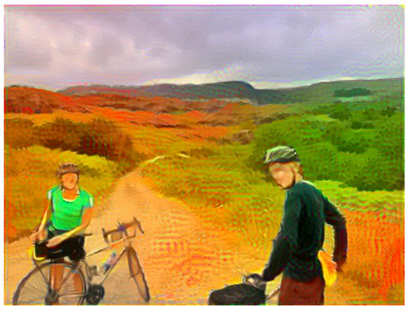
Style: Super Mario 64



Result

Content: Biking in Ireland

Style: "Adventure Time"



Result

References:

Contributions:
This implementation was primarily derived from an open-source github repository here: https://github.com/log0/neural-style-painting, and draws substantially on work founded in Gatys and Leon A.'s "A neural algorithm of artistic style"

Papers:
[1] Gatys, Leon A., Alexander S. Ecker, and Matthias Bethge. "A neural algorithm of artistic style."

[2] Gatys, Leon A., et al. "Controlling perceptual factors in neural style transfer." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017.

[3] Ashikhmin, N. "Fast texture transfer." IEEE Computer Graphics and Applications 23.4 (2003): 38-43.

[4] Luan, Fujun, et al. "Deep photo style transfer." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017.

[5] Johnson, Justin, Alexandre Alahi, and Li Fei-Fei. "Perceptual losses for real-time style transfer and super-resolution." European conference on computer vision. Springer, Cham, 2016.

Other Implementations (Code)
[6] Anis Athalye, Neural Style, https://github.com/anishathalye/neural-style/

[7] Coursera Deep Learning; "Art generation with Neural Style Transfer"