Cosimo Gonnelli, Connor Robetorye, Prof. Anthony Rhodes

CS 545 - Machine Learning

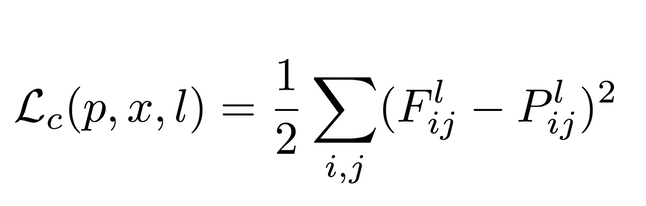
ARTISTIC STYLE TRANSFER

**Background and Methods:**

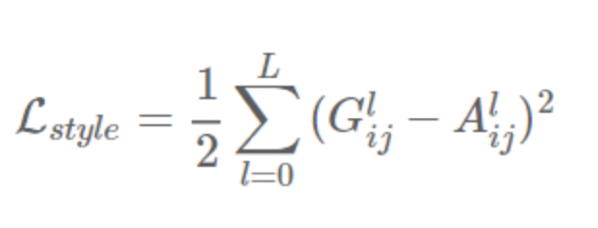
Though creativity is often pointed to as the unique spark of the human mind separating us from the other animals and certainly differentiating us from machines, myriad recent innovations In the world of AI and Machine Learning have created a fertile environment for researchers and students to explore the realm of computational creativity and machine-generated art. Among these recent innovations are the 2014 invention of Generative Adversarial Networks (GAN’s)[[1]](#footnote-0), as well as the application of deep Convolutional Neural Networks (CNN’s) to image classification[[2]](#footnote-1). Each of these innovations, have yielded previously unimagined creative capabilities to machines when they are turned towards the artistic space: much like a human artist might develop their own style through synthesizing the styles of artists who have influenced them while deviating from these norms to add a touch of their own flair, a GAN (or, more specifically, a CAN: a Creative Adversarial Network[[3]](#footnote-2) (2017)) can generate novel artwork by creating a feature map of artistic styles learned from established painters, and then sampling un-tested styles from within this mapped space to create novel and interesting works of art, such works being rated highly by practicing artists and even selling for hundreds of thousands at public auction.[[4]](#footnote-3) An alternative pathway towards visual creativity came through the use of use CNN’s in the ImageNet image classification challenge. These networks demonstrated the ability to extract high-level semantic information from a variety of content images while maintaining generalizability across 100’s of image classes, and the world was captivated when the strikingly human and psychedelic images produced by one such network, Google’s now famous DeepDream, became public in 2015. Given CNN’s proven ability to understand images in a human-like way, through a hierarchical composition of low-level features into high-level abstractions, it perhaps should be unsurprising that another group leveraged their visual potential towards the problem of capturing a particularly elusive sort of high-level abstraction: artistic style.

Enter Gatys et al. with their ground-breaking 2016 paper and the basis for our project: “Image Style Transfer using Convolutional Neural Networks”.[[5]](#footnote-4) In this paper, the authors demonstrated a technique for using a CNN trained on image classification for artistic style transfer. Artistic style transfer is a process of transferring a style from an image (style image) to another image (content image). This specific technique relies on the hierarchical properties of CNN’s. To briefly supply some relevant background information, a CNN is a class of deep NN that consists of layers of small computational units that process visual information hierarchically in a feed-forward manner. Each layer of units can be understood as a collection of image filters, each of which extracts a certain feature from the input image. The output of a given layer consists of feature maps: differently filtered versions of the input image. As it turns out, CNN’s trained on image-classification are remarkably good at recognizing and combining image features in a way that mirrors human visual composition. Thus, using a Convolutional Neural Network, Gatys et al. were able to represent an image in terms of features maps in the different layers of the network, and use these maps to transfer style between images.

To achieve the goal of transferring style it was necessary however to extract both the content and the style from the images’ feature representations. To do so, the authors represented style as a sum of gram matrices of feature maps of varying layers, and evaluated content using the feature map of a particular layer high in the network. Doing so allows style to be viewed as non-localized combination of high and low-level features, where image content is viewed as simply a somewhat abstracted version of the original image (a version with fewer particular details, allowing flexibility in the application of style). Thus, transferring style can be viewed as an optimization problem, wherein two losses are simultaneously minimized in an iteratively generated image: the difference between the style representation of the generated image and the style image, and the difference in content representation between the generated image and the content image. The purpose of the content loss in this optimization is to make sure that the generated image retains some of the “global” characteristics of the content image. This means for example, that shapes in the content image should be recognizable in the generated image, such as a lion’s ears, eyes, and facial structure. The content loss function is defined as below (where F contains the feature representation of the content image for convolutional layer l, and P contains the feature representation of the generated image for l):



On the other hand, the purpose of style loss is to preserve the stylistic characteristics of the style image. To this end, style loss is defined as the summed difference between Gram matrices from selected layers L, where G is the Gram matrix for the generated image and A is the Gram matrix for the style image:



The loss function being minimized is then just a simple weighted sum of the content and style loss as defined above.[[6]](#footnote-5)

The goals of our project were to replicate the work of Gatys et al. and to investigate the relative advantages of different techniques for optimizing this loss function. For our project, we used the Visual Geometry Group 19 (VGG19)[[7]](#footnote-6) as the CNN. This publically available model was pre-trained on image classification for the imageNet data set, on which it achieves 92.7% top-5 test accuracy.[[8]](#footnote-7) For style transfer however, only the first 16 layers were used, as the three highest, fully connected and softmax layers used in the image classifications themselves were not relevant to our problem. The particular loss optimization algorithms compared in our project were ADAM and L-BFGS. The Adaptive Moment Estimation (ADAM) algorithm is a combination of gradient descent with momentum and RMSprop algorithm. ADAM calculates an exponentially weighted average of past gradients of the objective function (1), along with an exponentially weighted average of the squares of the past gradients (2), each stored in their own sets of variables and stores it in different variables after bias corrections. It then updates parameters in a direction based on combining information from (1) and (2).[[9]](#footnote-8) The Limited Memory-Broyden Fletcher Goldfarb Shanno (L-BFGS) is instead an optimization algorithm in the family of quasi-Newton methods that approximates the standard BFGS under some memory constraints. L-BFGS approximates the objective function locally with a quadratic function, and computes an approximation of the Hessian at this point (hence the *quasi* in quasi-Newton). It then uses a Newtonian method to update the weights under optimization: it uses stored information about the most recently computed Hessians (not all previously computed Hessians, hence the *L* in L-BFGS), along with the gradient of the function and the local quadratic approximation to do a line search and get the direction in which to update the weights.[[10]](#footnote-9) This quadratic approximation, Hessian estimation, and line-search is computationally expensive. However, because of the non-stochasticity of this particular optimization problem (every subsequent generated image is not a random training example, but completely determined by the prior image), the computational cost of a more accurate, second-order estimation is warranted and can yield to quicker convergence in the overall optimization when compared with an algorithm more suited to general, stochastic gradient descent like ADAM.

L-BFGS was the algorithm used by Gatys et al in the original paper, however other researchers have chosen ADAM for this task for its lower computational demands (in terms of memory usage). “A Comprehensive Comparison between Neural Style Transfer and Universal Style Transfer”[[11]](#footnote-10) in particular cited L-BFGS as producing subjectively more appealing images, but greater overall memory usage in comparison with ADAM. The aims of our project were thus, beyond replicating the functional results of the Gatys paper in Tensorflow, to give a rough quantification in the difference in image quality between these two algorithms in terms of evaluated image loss over time, and also to instrument the chosen optimizing algorithms to measure their relative memory usage. We hypothesized that L-BFGS would yield lower loss over time in comparison to ADAM, corresponding to the higher subjective appeal of the images produced, but owing to its higher memory demands, the relative advantage of L-BFGS over ADAM would diminish as image size grew and memory became more constrained. We chose image sizes 64, 128, 256, 512, and 1024 as test cases, and ran a selection of 4 content and style image pairing at each size for 1000 iterations per each optimization function. The trials sized 64 were designed to demonstrate the effect of content image with style held constant, and those sized 128 to demonstrate the effect of style image with content image held constant. Our pre-experiment results showed content image to have little effect on run-time, but that style image was more significant, so the remaining sizes were controlled for style image. All experiments were run on PDX’s linux server over night, in four terminals, with the footnoted commands.[[12]](#footnote-11)

**Results and Discussion:**

With respect to the preliminary experiments of size 64 and 128, our results showed that varying content image had little appreciable effect on either optimizer’s performance, execution time, or loss, but that varying style image did affect execution time, though only for L-BFGS. That is, while the memory usage for each algorithm was relatively constant for all size 64 trials measuring the effect of content (~1GiB for L-BFGS, and 900MiB for ADAM), the time spent by the ADAM optimizer varied only between ~275 and 325 seconds but, somewhat unexpectedly, L-BFGS varied between 175 and 315 seconds on the same images. Three of these L-BFGS times were quite clustered, and the low, outlying time for L-BFGS was the lone black and white image. Our size 128 trials, in which style was varied but content held constant, ran 3 to 6 times longer than the size 64 trials, and also showed greater collective variability than those at size 64, particularly, again, for L-BFGS. Taken in conjunction, these results seem to indicate that choice of images has a much greater effect on run-time for L-BFGS than ADAM, particularly choice of style image. We presume that the relatively stronger effect of style on L-BFGS comes from the fact that it comprises a much larger percentage of our loss function (style to content weights were set in ratio 2,000 : 1, following Gatys et al.), and that L-BFGS is particularly affected because time to calculate a local quadratic approximation for the loss function and approximating a Hessian at that point shows more variability with loss-function complexity in comparison with the relatively constant gradient calculations performed by ADAM, hence the relative simplicity of the black and white image leading to low compute time, but only for L-BFGS. It is also worth noting that at these small image sizes, we saw significant over-fitting to style in the generated images, seemingly brought on by the low dimensionality of the content input. With all this in mind, trials 256 through 1024 were done with the same four styles, but as content images have less of an effect on our measured variables, we took the scientific liberty of varying them for the sake of variety in our presentation.

Our primary results were that, for *all* image sizes tested, L-BFGS dominated ADAM in terms of loss over time, but ADAM dominated in terms of memory usage. That is, for all sizes, L-BFGS achieved lower total loss than ADAM while terminating more quickly, but both the mean and peak memory usages of ADAM were lower than those of L-BFGS for all trials (with one exception). Thus, the hypothesis that L-BFGS would dominate in terms of image quality, as measured by loss, was confirmed, but the hypothesis that increasing image size and memory demands would dampen this effect was not confirmed: In fact, as image size grew, the relative disparity in memory demands between the two algorithms grew, but the disparity in loss over time remained relatively constant (see footnote for loss-versus time and memory usage graphs for each image size and for the generated images).[[13]](#footnote-12) This might be explained by a failure in experimental design, whereby neither algorithm ever ran into ‘memory scarcity’: Even at the 1024x1024 trial size, when L-BFGS images were using upwards of 6GiB of memory with ADAM using around 4GiB, the environment running the algorithm had enough free memory to feed the algorithm. Perhaps if memory were constrained to, for example 2GiB, such that each algorithm were not able to run cleanly in main memory, we might have seen the performance of L-BFGS in terms of time of execution take a hit relatively greater than that of ADAM as we expected.

Collectively, the dominance with respect to time and loss-evaluated performance of L-BFGS mean it should be the preferred algorithm for those who wish to generate high-quality images relatively quickly. Not only does L-BFGS produce subjectively more detailed and well-stylized images, it diminishes objectively evaluated loss by a significantly greater factor than does ADAM, and does so without the need to tune any hyperparameters (as one must with the ADAM optimizer). However, particularly as image dimension grows, the memory hit of L-BFGS over ADAM is significant, so for memory-limited systems or batch processing, ADAM could be the preferred algorithm. Indeed, though we were not able to directly measure this, ADAM’s rate of loss optimization over time might exceed that of L-BFGS when the program’s demands exceed what can be supplied at once by main memory, as with very large images. Though Gatys et al.’s iterative optimization technique for style transfer has fallen out of favor to the real-time style transfer technique since developed by Google in 2017, which achieves style transfer in a single feed-forward pass using a pre-trained style recognition network,[[14]](#footnote-13) ADAM-based iterative optimization might still find a use in its superior ability over L-BFGS (at least in our tensorflow implementation) to stop and generate intermediate outputs during the optimization process, which one could use to and create visualizations of the style transfer process that are lost in a single feed-forward pass, but future research is rightly directed towards the latter method.

1. <https://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf> [↑](#footnote-ref-0)
2. <http://www.robots.ox.ac.uk/~vgg/research/deep_eval/> [↑](#footnote-ref-1)
3. <https://arxiv.org/abs/1706.07068> [↑](#footnote-ref-2)
4. <https://www.nytimes.com/2018/10/25/arts/design/ai-art-sold-christies.html> [↑](#footnote-ref-3)
5. <https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Gatys_Image_Style_Transfer_CVPR_2016_paper.pdf> [↑](#footnote-ref-4)
6. <https://medium.com/data-science-group-iitr/artistic-style-transfer-with-convolutional-neural-network-7ce2476039fd> [↑](#footnote-ref-5)
7. [Very Deep Covolutional Networks for Large-scale image Recognition](https://arxiv.org/pdf/1409.1556.pdf) [↑](#footnote-ref-6)
8. <http://image-net.org/> [↑](#footnote-ref-7)
9. <https://arxiv.org/pdf/1412.6980.pdf> [↑](#footnote-ref-8)
10. <http://curtis.ml.cmu.edu/w/courses/index.php/L-BFGS> [↑](#footnote-ref-9)
11. <https://arxiv.org/pdf/1806.00868.pdf> [↑](#footnote-ref-10)
12. <https://github.com/inordirection/style-transfer/blob/master/experiments.txt> [↑](#footnote-ref-11)
13. <https://github.com/inordirection/style-transfer/tree/master/results> [↑](#footnote-ref-12)
14. <https://arxiv.org/pdf/1705.06830.pdf> [↑](#footnote-ref-13)