# Project #5: **Artistic Style Transfer**

Team Members:
Fernando Velasco Salazar
Pablo Eduardo Gomez Romo
MiTIC UAG
Prof. Juan Antonio Vega
**Source Code: https://github.com/egomez99/ArtisticStyleTransfer**

## Abstract

*El Objetivo de este proyecto es transferir el estilo artístico de un cuadro de pintura famoso a una fotografía cualquiera tomada recientemente. Se deben utilizar al menos 5 fotografías (contenido) y 5 cuadros famosos (estilo artístico). Además, se deben realizar tres experimentos que varíen el resultado. Para los tres experimentos, puede variar los parámetros de la red neuronal, los filtros, o la topología, pero les recomiendo leer algo de la literatura sobre el tema para buscar variaciones que tengan un efecto artístico sobre la imagen resultante.*

## Introduction

[1] (**A Neural Algorithm of Artistic Style (**Leon A. Gatys, Alexander S. Ecker, Matthias Bethge *(Submitted on 26 Aug 2015 (v1), last revised 2 Sep 2015 (this version, v2))*) SDS Data Science Group implemented https://github.com/dsgiitr/Neural-Style-Transfer

***To represent Content, it activates features (transfer contours):*** *Given content layer **l**, the content loss is defined as the Mean Squared Error between the feature map **F** of our content image **C** and the feature map **P** of our generated image **Y**. When content-loss is minimized, it means that the mixed-image has feature activation in the given layers that are very similar to the activation of the content-image.*

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

***To represent style:*** *we measure which features in the style-layers activate simultaneously for the style-image. Copy this activation-pattern to the mixed-image. Gramm matrix of correlated features. Dot product if zero in given layer do not activate. Otherwise a large value, then it means the two features do activate simultaneously for the given style-image. Mixed image replicates this activation.*

$$G_{ij} = \sum_{k} F_{ik}F_{jk}$$

***How to optimize****: With images for Content, Style and Output it could minimize losses in the network. The output image generated from such a network, resembled the input image and had the stylist attributes of the style image. The total loss can then be written as a weighted sum of the both the style and content losses.*

$$\mathcal{L}_{total} = \alpha\mathcal{L}_{content} + \beta\mathcal{L}_{style}$$

**For VGG16 arquitecture u**sed the Keras model-level library running "**Neural-Style-Transfer-1/Keras_StyleTransfer.py**", provides high-level building blocks for developing deep learning model. Input tensor passes 3 arrays to Model VGG16 (with pre-trained weights file vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5) which are:

```
input_tensor = backend.concatenate(
   [content_image, style_image, combination_image], axis=0)
```

3 parameters adjust the final result for our styled transfer image:

```
content_weight = 0.05
style_weight = 5.0
total_variation_weight = 1.0
```

The best results are achieved by a combination of many different layers from the network, which capture both the finer textures and the larger elements of the original image.

Calculate a matrix comprising of correlated features for the tensors output by the style layers. Then the Mean Squared Error for the Gram-matrices instead of the raw tensor-outputs from the layers. One could minimize the losses in the network such that the style loss (loss between the output image style and style of 'style image'. For the content loss (loss between the content image and the output image) its also computed by the script.

Evaluated gradient loss is minimized with a function using the L-BFGS-B algorithm (L-BFGS stores only a few vectors that represent the approximation implicitly. Due to its resulting linear memory requirement, the L-BFGS method is particularly well suited for optimization problems with a large number of variables).

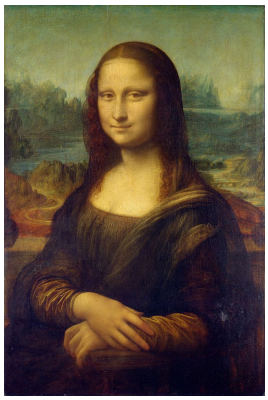


*Figure 1 Content Image, UAG building. Figure 2 Style graphics for a picture from Van Go. Figure 3 The combination result after 10 iterations using parameters from left.*

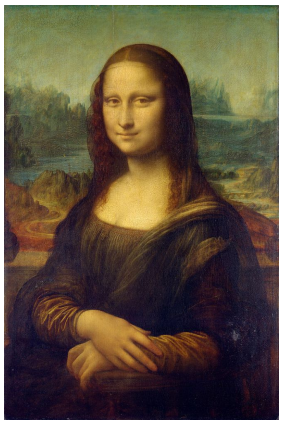
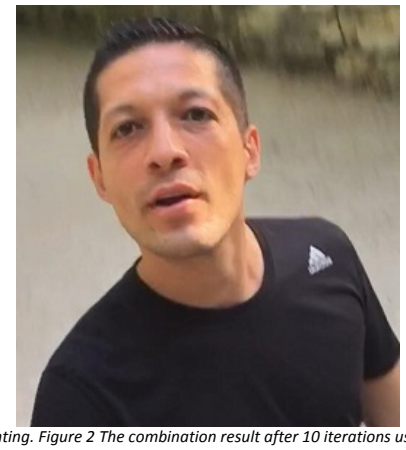




*Figure 1 Style graphic based on Monalisa painting. Figure 2 The combination result after 10 iterations using parameters from left*
*Figure 1 Content Image, my crappy selfie*

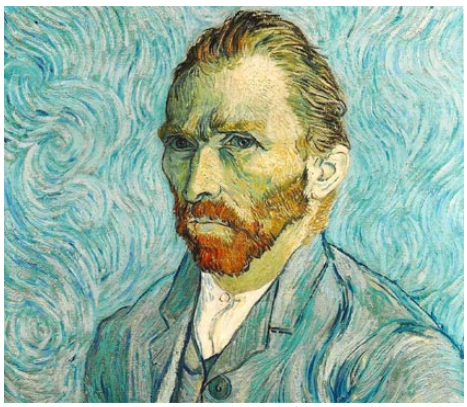
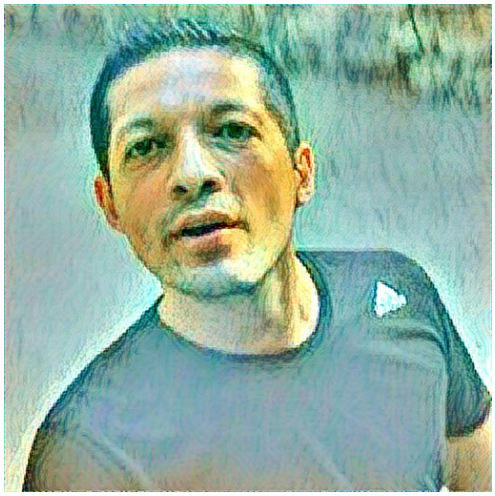
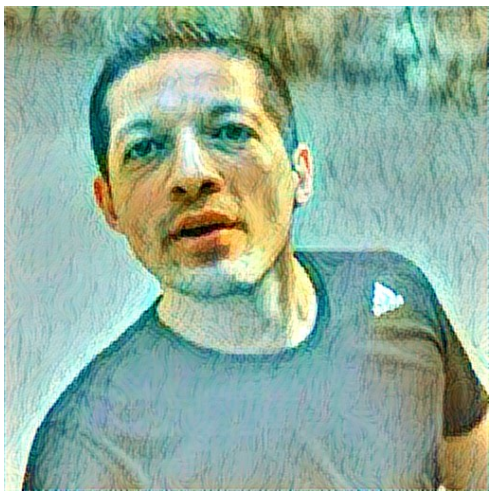
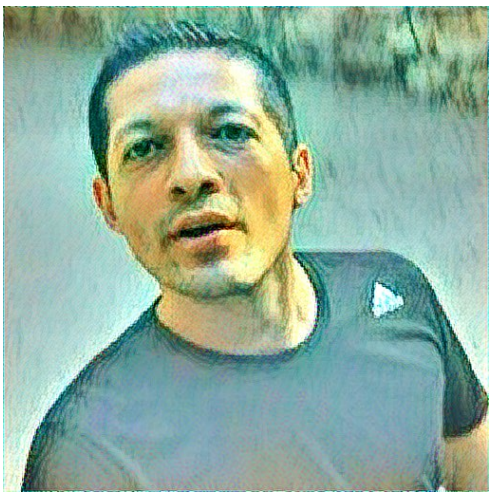
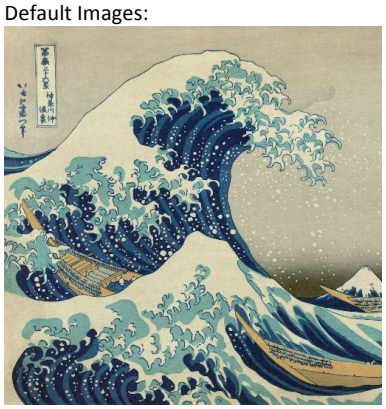


*Figure 1 Style graphic based on Van Go painting.*
*Figure 2 Style Weight 5 after 10 iterations*
*Figure 3 Style Weight 9 after 10 iterations*
*Figure 4 Style Weight 15 after 20 iterations*

Default Images:



A second **VGG19 Architecture** was also tested to create styled transfer images. Convolutional Layers and activations uses a back propagation network to make styled image:

16 convolutional layers and 5 pooling layers. There are no Fully Connected layers. For image synthesis we found that replacing the max-pooling operation by average pooling improves the gradient flow and one obtains slightly more appealing results, which is why the images shown were generated with average pooling.

```
VGG19_LAYERS = (
    'conv1_1', 'relu1_1', 'conv1_2', 'relu1_2', 'pool1',

    'conv2_1', 'relu2_1', 'conv2_2', 'relu2_2', 'pool2',

    'conv3_1', 'relu3_1', 'conv3_2', 'relu3_2', 'conv3_3',
    'relu3_3', 'conv3_4', 'relu3_4', 'pool3',

    'conv4_1', 'relu4_1', 'conv4_2', 'relu4_2', 'conv4_3',
    'relu4_3', 'conv4_4', 'relu4_4', 'pool4',

    'conv5_1', 'relu5_1', 'conv5_2', 'relu5_2', 'conv5_3',
    'relu5_3', 'conv5_4', 'relu5_4'
)
```

These parameters can make the difference when creating the images:

```
# default arguments
CONTENT_WEIGHT = 5e0
CONTENT_WEIGHT_BLEND = 1
STYLE_WEIGHT = 5e2
TV_WEIGHT = 1e2
STYLE_LAYER_WEIGHT_EXP = 1
LEARNING_RATE = 1e1
BETA1 = 0.9
BETA2 = 0.999
EPSILON = 1e-08
STYLE_SCALE = 1.0
ITERATIONS = 100
VGG_PATH = 'imagenet-vgg-verydeep-19.mat'
POOLING = 'max'
```

## Results and Discussion

The second network can be more effective although works on a similar way to the VGG16. In the “stylize.py” the style and the content loss are calculated and the Adam Optimizer trains the network with the learning rate, beta1, beta2, and epsilon parameters. Minimizes loss and runs a TF session evaluating loss vs best loss to improve on accuracy. Production time is also lower than the original implementation posed at the beginning (VGG16). The data after running 100 iterations last 12 minutes exhibiting these data:

| VGG16 – L-BGFS | VGG19 – Adam Optimizer |
|---|---|
| axon:Neural-Style-Transfer-1 egomez$ python3 Keras_StyleTransfer.py<br>Using TensorFlow backend.<br>/Users/egomez/miniconda3/envs/cifar100/lib/python3.5/importlib/_bootstrap.py:222: RuntimeWarning: compiletime version 3.6 of module 'tensorflow.python.framework.fast_tensor_util' does not match runtime version 3.5<br>  return f(*args, **kwds)<br>(1, 512, 512, 3)<br>(1, 512, 512, 3)<br>2018-04-10 19:15:10.219327: I tensorflow/core/platform/cpu_feature_guard.cc:137] Your CPU supports instructions that this TensorFlow binary was not compiled to use: SSE4.2<br>/Users/egomez/dev/MiTiC UAG/5 Cuatri/Deep Learning/deeplearning_repo/Proyecto #5/Neural-Style-Transfer-1/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5<br>Ya existen los pesos para esta configuración<br><br>Start of iteration 0<br>Position of the minimum [-68.66584358 -10.25413373 72.396128 ... -17.31963327 -71.1011142<br>  97.88485415] and its value 32279478000.0<br>Info dict (info): {'funcalls': 21, 'warnflag': 1, 'nit': 13, 'task': b'STOP: TOTAL NO. of f AND g EVALUATIONS EXCEEDS LIMIT', 'grad': array([ -675.45251465, -1070.33093262, 376.13977051, ..., | Optimization started...<br>content loss: 1.90956e+07<br>style loss: 2.36408e+09<br>tv loss: 26.3094<br>total loss: 2.38318e+09<br><br>**Iteration 1/ 100**<br>content loss: 1.8345e+07<br>style loss: 2.19076e+09<br>tv loss: 23932.6<br>total loss: 2.20913e+09<br><br>**Iteration 11/ 100**<br>content loss: 2.29778e+07<br>style loss: 6.68548e+08<br>tv loss: 120200<br>total loss: 6.91646e+08 |

0 if converged, 1 if too many function evaluations or too many iterations, 2 if stopped for another reason, given in info
task
d[task] b'STOP: TOTAL NO. of f AND g EVALUATIONS EXCEEDS LIMIT'
**Grad at Minimum: [ -675.45251465 -1070.33093262 376.13977051 ... 294.19558716**
**389.9755249 478.76654053]**
Function Calls Made: 21
Number Iterations: 13
Iteration 0 completed in 1140s

Start of iteration 1
Position of the minimum [ -49.39201019 23.88398481 63.0785059 ... -36.52706501 -121.46432216
72.17448815] and its value 23164158000.0
Info dict (info): {'funcalls': 21, 'warnflag': 1, 'nit': 15, 'task': b'STOP: TOTAL NO. of f AND g EVALUATIONS EXCEEDS LIMIT',
'grad': array([ 227.08477783, 217.57481384, 370.39434814, ..., 738.38470459,
1423.44445801, 1277.71679688])}
0 if converged, 1 if too many function evaluations or too many iterations, 2 if stopped for another reason, given in info
task
d[task] b'STOP: TOTAL NO. of f AND g EVALUATIONS EXCEEDS LIMIT'
**Grad at Minimum: [ 227.08477783 217.57481384 370.39434814 ... 738.38470459 1423.44445801**
**1277.71679688]**
Function Calls Made: 21
Number Iterations: 15
Iteration 1 completed in 1104s

Start of iteration 2
Position of the minimum [ -55.85668395 17.86585007 46.03015378 ... -74.32203742 -207.06309269
4.19603496] and its value 19720935000.0
Info dict (info): {'funcalls': 21, 'warnflag': 1, 'nit': 16, 'task': b'STOP: TOTAL NO. of f AND g EVALUATIONS EXCEEDS LIMIT',
'grad': array([1939.12890625, 2392.44775391, 2381.34960938, ..., 362.62756348,
-97.92776489, 373.46325684])}
0 if converged, 1 if too many function evaluations or too many iterations, 2 if stopped for another reason, given in info
task
d[task] b'STOP: TOTAL NO. of f AND g EVALUATIONS EXCEEDS LIMIT'
**Grad at Minimum: [1939.12890625 2392.44775391 2381.34960938 ... 362.62756348 -97.92776489**
**373.46325684]**
Function Calls Made: 21
Number Iterations: 16
Iteration 2 completed in 1036s

Start of iteration 3
Position of the minimum [-101.08828104 -37.41344224 -6.23103497 ... -91.59684325 -220.02011092
-15.07572329] and its value 18326129000.0
Info dict (info): {'funcalls': 21, 'warnflag': 1, 'nit': 16, 'task': b'STOP: TOTAL NO. of f AND g EVALUATIONS EXCEEDS LIMIT',
'grad': array([-270.04354858, -609.76098633, -562.50750732, ..., 360.78839111,
615.01849365, 424.4987793 ])}
0 if converged, 1 if too many function evaluations or too many iterations, 2 if stopped for another reason, given in info
task
d[task] b'STOP: TOTAL NO. of f AND g EVALUATIONS EXCEEDS LIMIT'
**Grad at Minimum: [-270.04354858 -609.76098633 -562.50750732 ... 360.78839111 615.01849365**
**424.4987793 ]**
Function Calls Made: 21
Number Iterations: 16
Iteration 3 completed in 1152s

Start of iteration 4
Position of the minimum [-103.37011404 -32.33264045 2.36608699 ... -110.57165799 -241.78954383
-32.06569418] and its value 17692176000.0
Info dict (info): {'funcalls': 21, 'warnflag': 1, 'nit': 17, 'task': b'STOP: TOTAL NO. of f AND g EVALUATIONS EXCEEDS LIMIT',
'grad': array([331.88415527, 119.89001465, 264.16094971, ..., 298.3112793 ,
11.80232239, 172.39199829])}
0 if converged, 1 if too many function evaluations or too many iterations, 2 if stopped for another reason, given in info
task
d[task] b'STOP: TOTAL NO. of f AND g EVALUATIONS EXCEEDS LIMIT'
**Grad at Minimum: [331.88415527 119.89001465 264.16094971 ... 298.3112793 11.80232239**
**172.39199829]**
Function Calls Made: 21
Number Iterations: 17
Iteration 4 completed in 1093s

content loss: 2.4294e+07
style loss: 2.83796e+08
tv loss: 116418
total loss: 3.08207e+08

**Iteration 31/ 100**
content loss: 2.43518e+07
style loss: 1.55095e+08
tv loss: 101371
total loss: 1.79548e+08

**Iteration 41/ 100**
content loss: 2.40891e+07
style loss: 9.73539e+07
tv loss: 91224.6
total loss: 1.21534e+08

**Iteration 51/ 100**
content loss: 2.34863e+07
style loss: 7.10214e+07
tv loss: 86200.5
total loss: 9.45938e+07

**Iteration 61/ 100**
content loss: 2.32527e+07
style loss: 6.06712e+07
tv loss: 85330.9
total loss: 8.40093e+07

**Iteration 71/ 100**
content loss: 2.33189e+07
style loss: 4.82494e+07
tv loss: 83883
total loss: 7.16522e+07

**Iteration 81/ 100**
content loss: 2.28495e+07
style loss: 3.9202e+07
tv loss: 83106.5
total loss: 6.21346e+07

**Iteration 91/ 100**
content loss: 2.23493e+07
style loss: 3.53853e+07
tv loss: 83076.3
total loss: 5.78176e+07

**FINAL DATA:**
content loss: 2.21624e+07
style loss: 3.31683e+07
tv loss: 83012.9
total loss: 5.54137e+07

Position of the minimum [-113.32474789 -39.42320419 -4.97539947 ... -121.89474416 -245.51627799
-37.68455472] and its value 17367845000.0
Info dict (info): {'funcalls': 21, 'warnflag': 1, 'nit': 17, 'task': b'STOP: TOTAL NO. of f AND g EVALUATIONS EXCEEDS LIMIT',
'grad': array([ 100.59750366, 35.20730591, -114.04934692, ..., 342.80822754,
358.15374756, 201.22393799])}
0 if converged, 1 if too many function evaluations or too many iterations, 2 if stopped for another reason, given in info
task
d[task] b'STOP: TOTAL NO. of f AND g EVALUATIONS EXCEEDS LIMIT'
**Grad at Minimum: [ 100.59750366 35.20730591 -114.04934692 ... 342.80822754 358.15374756**
**201.22393799]**
Function Calls Made: 21
Number Iterations: 17
Iteration 5 completed in 1876s

Start of iteration 6
Position of the minimum [-117.02547955 -41.29912714 -4.85882713 ... -131.76278103 -253.56069049
-42.51138712] and its value 17190126000.0
Info dict (info): {'funcalls': 21, 'warnflag': 1, 'nit': 17, 'task': b'STOP: TOTAL NO. of f AND g EVALUATIONS EXCEEDS LIMIT',
'grad': array([ 44.59740067, -68.46583557, 62.04046631, ..., 240.44769287,
44.74942398, 132.7640686 ])}
0 if converged, 1 if too many function evaluations or too many iterations, 2 if stopped for another reason, given in info
task
d[task] b'STOP: TOTAL NO. of f AND g EVALUATIONS EXCEEDS LIMIT'
**Grad at Minimum: [ 44.59740067 -68.46583557 62.04046631 ... 240.44769287 44.74942398**
**132.7640686 ]**
Function Calls Made: 21
Number Iterations: 17
Iteration 6 completed in 2039s

Start of iteration 7
Position of the minimum [-120.00946995 -42.94373914 -7.22849155 ... -137.91404657 -255.2243112
-45.27784916] and its value 17075969000.0
Info dict (info): {'funcalls': 21, 'warnflag': 1, 'nit': 17, 'task': b'STOP: TOTAL NO. of f AND g EVALUATIONS EXCEEDS LIMIT',
'grad': array([178.74102783, 151.26974487, 110.90898132, ..., 182.41041565,
229.19296265, 109.55125427])}
0 if converged, 1 if too many function evaluations or too many iterations, 2 if stopped for another reason, given in info
task
d[task] b'STOP: TOTAL NO. of f AND g EVALUATIONS EXCEEDS LIMIT'
**Grad at Minimum: [178.74102783 151.26974487 110.90898132 ... 182.41041565 229.19296265**
**109.55125427]**
Function Calls Made: 21
Number Iterations: 17
Iteration 7 completed in 1858s

Start of iteration 8
Position of the minimum [-123.55312255 -46.57460526 -8.80377646 ... -142.78911805 -259.56395453
-47.12933532] and its value 16995633000.0
Info dict (info): {'funcalls': 21, 'warnflag': 1, 'nit': 18, 'task': b'STOP: TOTAL NO. of f AND g EVALUATIONS EXCEEDS LIMIT',
'grad': array([ 105.2568512 , 55.44471741, 93.53216553, ..., 122.27689362,
-143.40956116, -24.1235714 ])}
0 if converged, 1 if too many function evaluations or too many iterations, 2 if stopped for another reason, given in info
task
d[task] b'STOP: TOTAL NO. of f AND g EVALUATIONS EXCEEDS LIMIT'
Grad at Minimum: [ 105.2568512 55.44471741 93.53216553 ... 122.27689362 -143.40956116
-24.1235714 ]
Function Calls Made: 21
Number Iterations: 18
Iteration 8 completed in 3396s

Start of iteration 9
Position of the minimum [-125.57150896 -48.30599047 -10.30681979 ... -147.06024784 -260.57034116
-48.52544293] and its value 16934557000.0
Info dict (info): {'funcalls': 21, 'warnflag': 1, 'nit': 18, 'task': b'STOP: TOTAL NO. of f AND g EVALUATIONS EXCEEDS LIMIT',
'grad': array([ 95.91752625, 146.02241516, 81.50404358, ..., 255.88401794,
335.1182251 , 173.59873962])}
0 if converged, 1 if too many function evaluations or too many iterations, 2 if stopped for another reason, given in info
task
d[task] b'STOP: TOTAL NO. of f AND g EVALUATIONS EXCEEDS LIMIT'
**Grad at Minimum: [ 95.91752625 146.02241516 81.50404358 ... 255.88401794 335.1182251**

| | |
|---|---|
| Function Calls Made: 21<br>Number Iterations: 18<br>Iteration 9 completed in 3259s<br>Exception ignored in: <bound method BaseSession.__del__ of <tensorflow.python.client.session.Session object at 0x11cf35dd8>><br>Traceback (most recent call last):<br>File "/Users/egomez/miniconda3/envs/cifar100/lib/python3.5/site-packages/tensorflow/python/client/session.py", line 702, in __del__<br>TypeError: 'NoneType' object is not callable | |

TensorFlow doesn't support L-BFGS (which is what the original authors used), so we use Adam. This may require a little bit more hyperparameter tuning to get nice results. https://github.com/anishathalye/neural-style

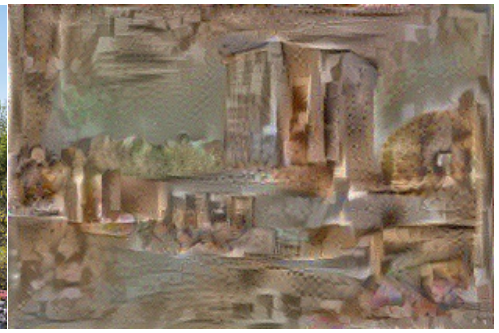
For each iteration it relies on specialized, well-optimized parameter estimation in machine learning.

Start of iteration 0
Position of the minimum [ -8.16304657 20.27224516 90.45626733 ... -96.68206586 137.27733763
-73.72396086] and its value 807178600000.0
Info dict (info): {'nit': 10, 'grad': array([11720.52734375, 5337.93310547, 8558.05859375, ...,
-4518.48828125, -6197.40527344, -1386.80834961]), 'warnflag': 1, 'funcalls': 21, 'task': b'STOP: TOTAL NO. of f AND g EVALUATIONS EXCEEDS LIMIT'}
0 if converged, 1 if too many function evaluations or too many iterations, 2 if stopped foranother reason, given in info task
d[task] b'STOP: TOTAL NO. of f AND g EVALUATIONS EXCEEDS LIMIT'
Grad at Minimum: [11720.52734375 5337.93310547 8558.05859375 ... -4518.48828125
-6197.40527344 -1386.80834961]
Function Calls Made: 21
Number Iterations: 10
Iteration 0 completed in 1148s
Iteration 1 completed in 1104s
Iteration 2 completed in 1036s
Iteration 3 completed in 1152s
Iteration 4 completed in 1093s
Iteration 5 completed in 1876s
Iteration 6 completed in 2039s
Iteration 7 completed in 1858s
Iteration 8 completed in 3396s
Iteration 9 completed in 3259s
**Total Time**: 17961 secs / 60 = 299.35 mins / 60 = 4.9891 hrs … X_X
Exception ignored in: <bound method BaseSession.__del__ of <tensorflow.python.client.session.Session object at 0x11cf35dd8>>
Traceback (most recent call last):
File "/Users/egomez/miniconda3/envs/cifar100/lib/python3.5/site-packages/tensorflow/python/client/session.py", line 702, in __del__
TypeError: 'NoneType' object is not callable i..e. https://github.com/tensorflow/tensorflow/issues/3388

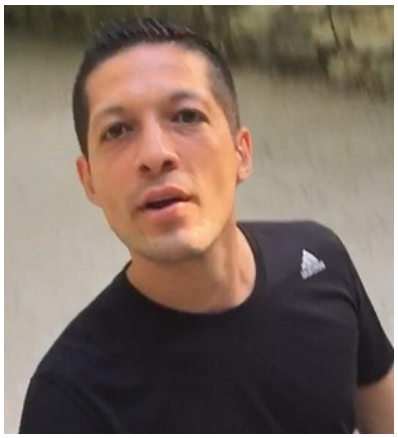
**DaVinci Style** **Content** **Results**

*Average* *Pooling Max*











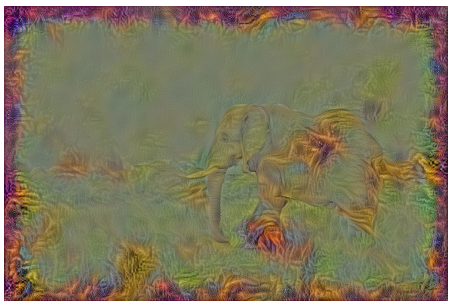


**DaVinci Style** **Content** **Results**

*Average* *Pooling Max*

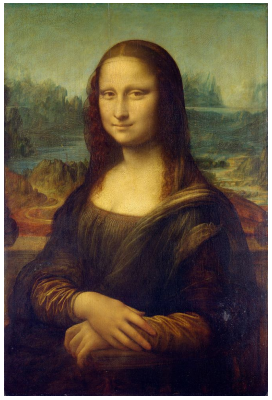


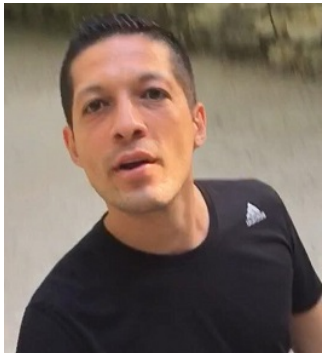


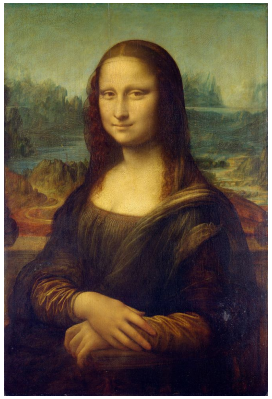




**Monalisa Style** **Content** **Results**

*Average* *Pooling Max*

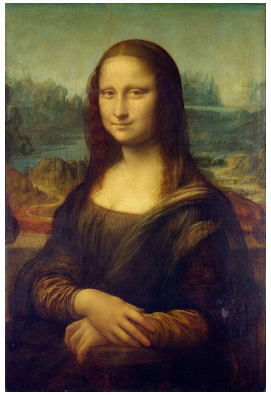


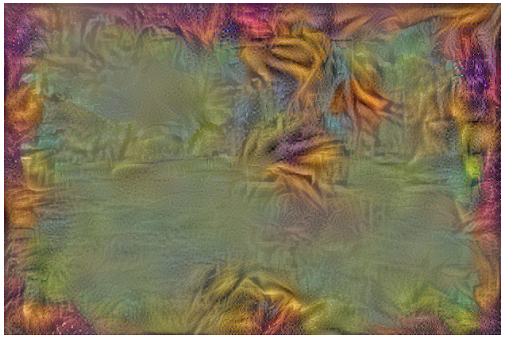


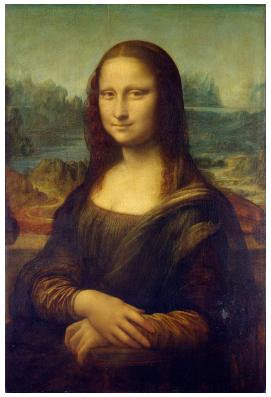


**Monalisa Style** **Content** **Results**

*Average* *Pooling Max*



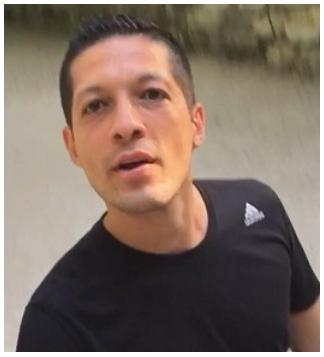






**Picasso Style** **Content** **Results**

*Average* *Pooling Max*







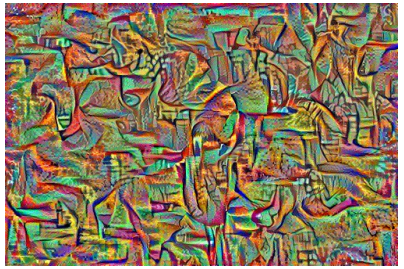
**Picasso Style** **Content** **Results**

*Average* *Pooling Max*





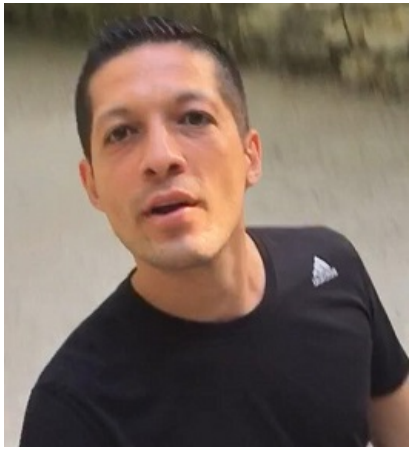
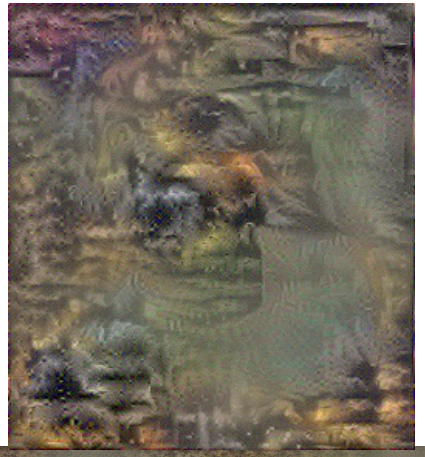




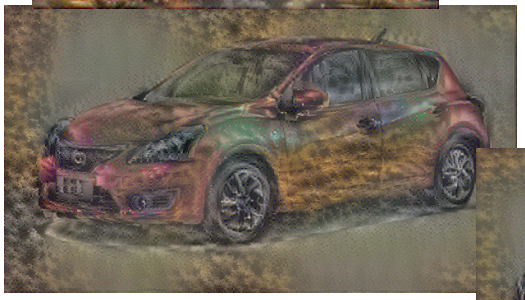
**Rembrandt Style** **Content** **Results**

*Average* *Pooling Max*



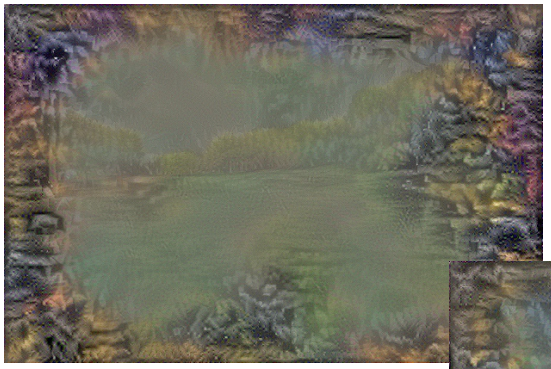
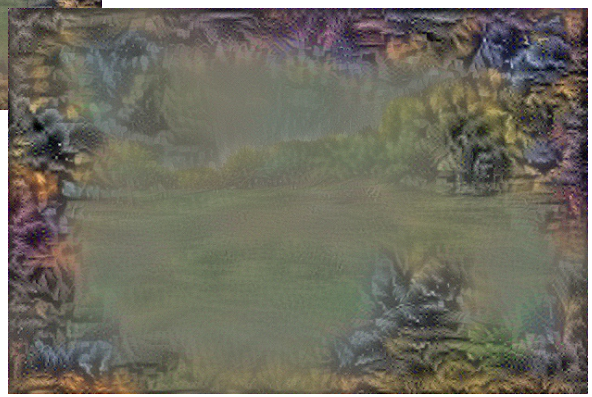





**Monalisa Style** **Content** **Results**

*Average* *Pooling Max*

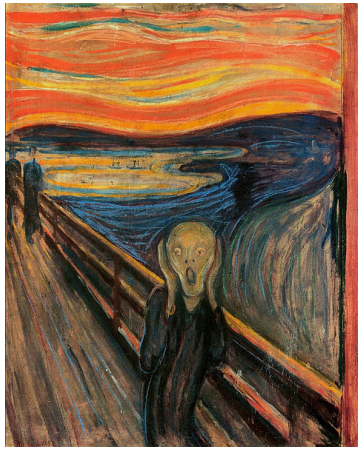


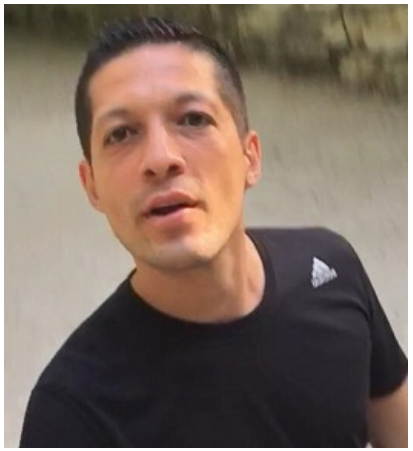



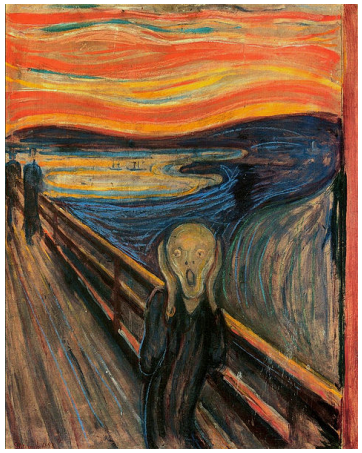


**Scream Style** **Content** **Results**

*Average* *Pooling Max*

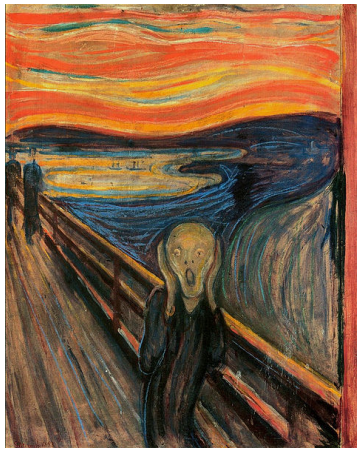


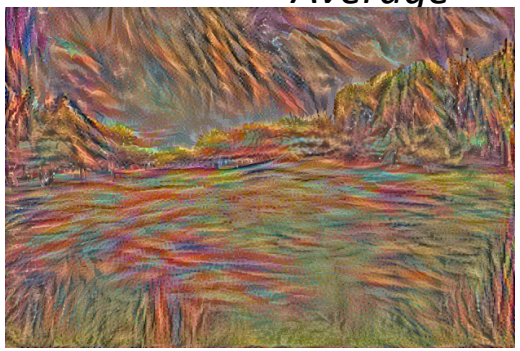
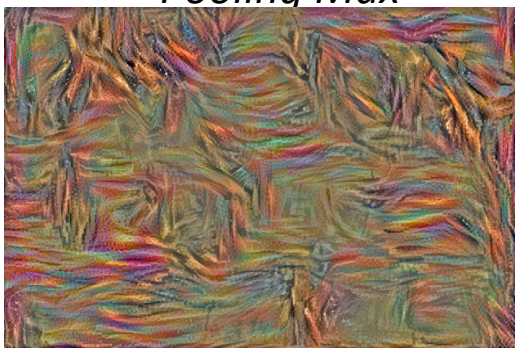
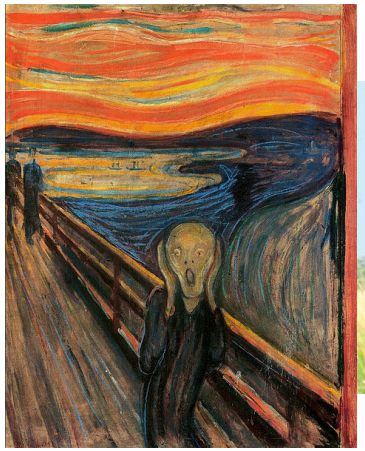


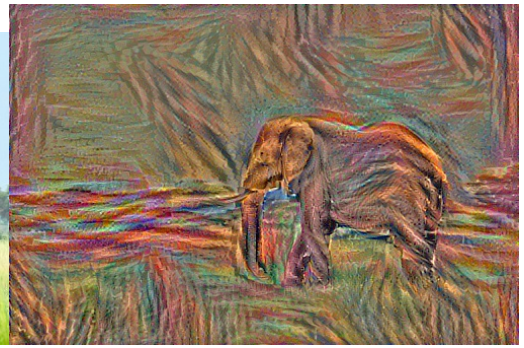
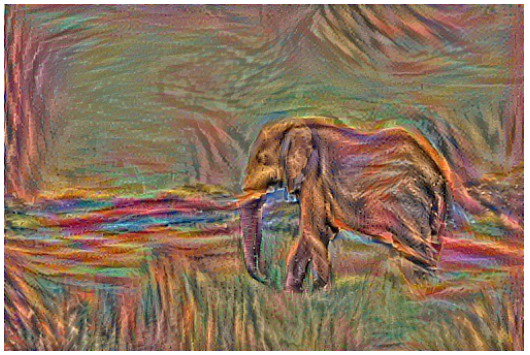


**Scream Style** **Content** **Results**

*Average* *Pooling Max*



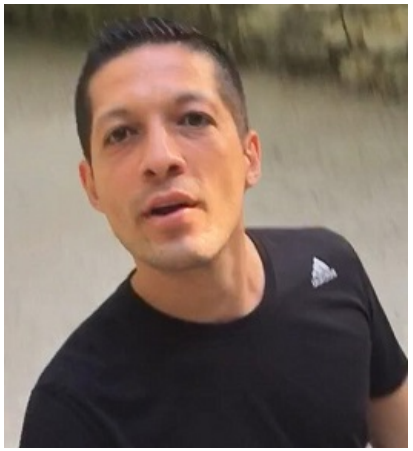
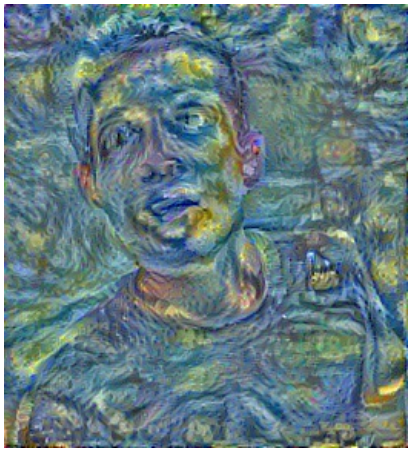




**Starry Night Style** **Content** **Results**

*Average* *Pooling Max*



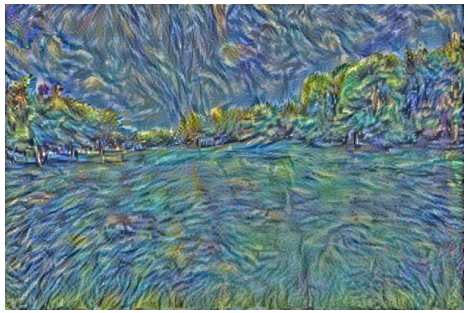
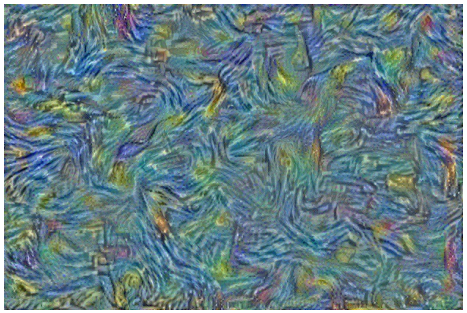






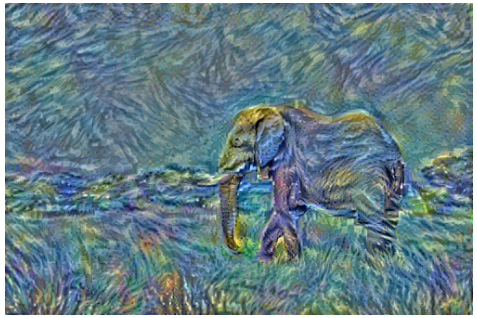
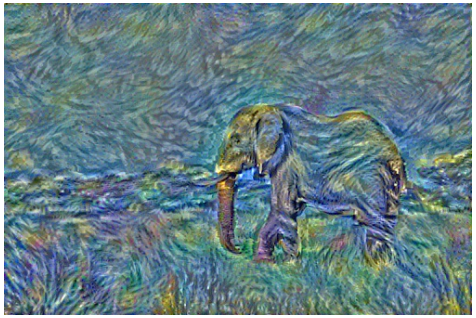

**Starry Night Style** **Content** **Results**

*Average* *Pooling Max*



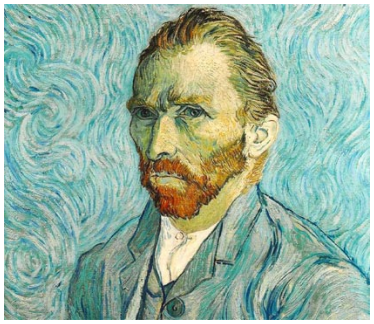



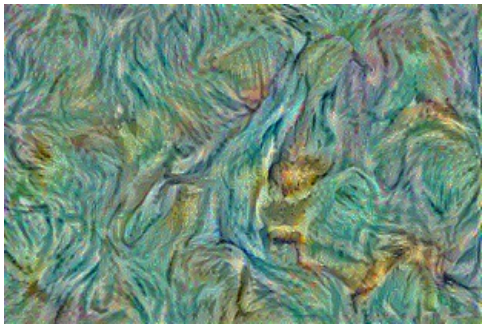
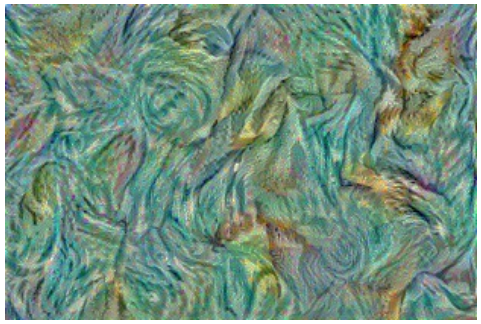


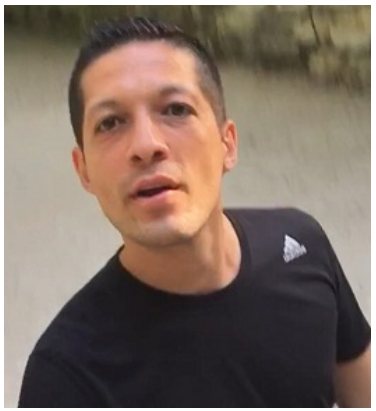



**Vincent Style** **Content** **Results**

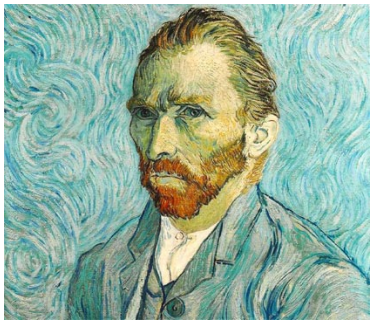
*Average* *Pooling Max*

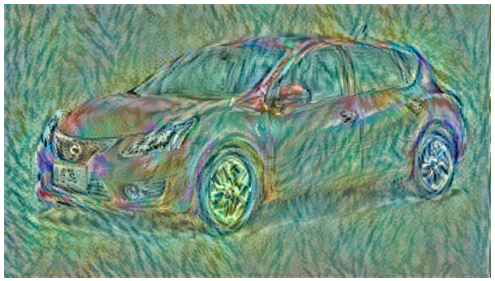




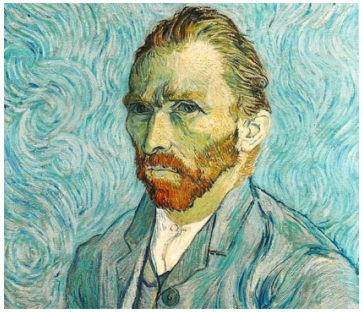

*100* *1000* *10000*

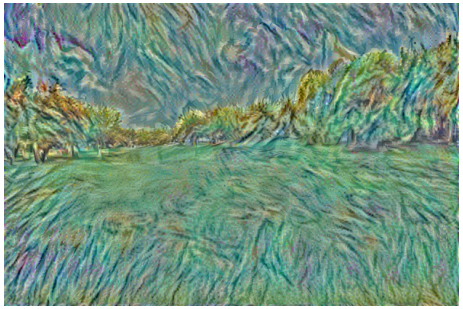
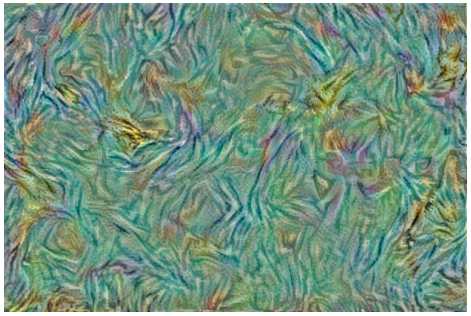


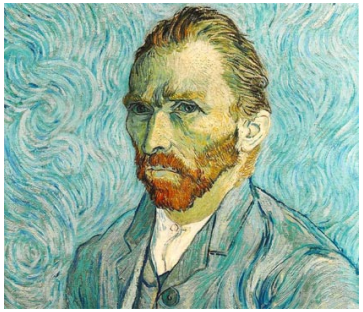


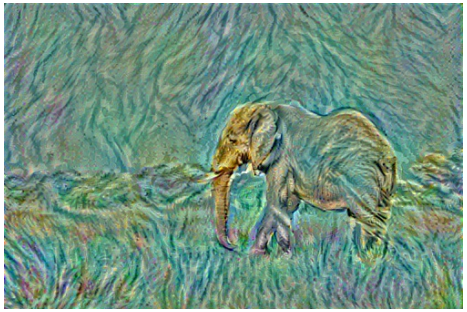
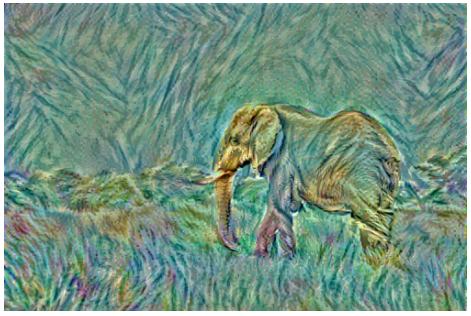

**Vincent Style** **Content** **Results**

*Average* *Pooling Max*





## Note

For VGG16 script could not get data above to compare results.

raise ValueError("Cannot evaluate tensor using `eval()`: No default "

ValueError: Cannot evaluate tensor using `eval()`: No default session is registered. Use `with sess.as_default()` or pass an explicit session to `eval(session=sess)`

## Hardware

All tests were ran with a Macbook Pro laptop: MacBook Pro (15-inch, Mid 2010), 2.4 GHz Intel Core i5, 8 GB 1067 MHz DDR3, NVIDIA GeForce GT 330M 256 MB Intel HD Graphics 288 MB.

## Issues

**Note:** There are known, accuracy-affecting numerical issues before macOS 10.12.6 (Sierra) that are described in GitHub#15933.

**Note:** As of version 1.2, TensorFlow no longer provides GPU support on macOS.

## Reference


[1] (**A Neural Algorithm of Artistic Style (**Leon A. Gatys, Alexander S. Ecker, Matthias Bethge *(Submitted on 26 Aug 2015 (v1), last revised 2 Sep 2015 (this version, v2))*) SDS Data Science Group implemented https://github.com/dsgiitr/Neural-Style-Transfer

VGG16 **Artistic Style Transfer with Convolutional Neural Network**

https://medium.com/data-science-group-iitr/artistic-style-transfer-with-convolutional-neural-network-7ce2476039fd

A Neural Algorithm of Artistic Style

Leon A. Gatys,1,2,3∗ Alexander S. Ecker,1,2,4,5 Matthias Bethge1,2,4

https://arxiv.org/pdf/1508.06576v2.pdf, 2000.

1Werner Reichardt Centre for Integrative Neuroscience

and Institute of Theoretical Physics, University of Tu¨bingen, for Neural Information Processing, Tu¨bingen, Germany 4Max Planck Institute for Biological Cybernetics, Tu¨bingen, Germany 5Department of Neuroscience, Baylor College of Medicine, Houston, TX, USA ∗To whom correspondence should be addressed; E-mail: leon.gatys@bethgelab.org [3] 1. Krizhevsky, A., Sutskever, I. & Hinton, G. E. Imagenet classification with deep convolu- tional neural networks. In *Advances in neural information processing systems*, 1097–1105 (2012). URL http://papers.nips.cc/paper/4824-imagenet.

2. Taigman, Y., Yang, M., Ranzato, M. & Wolf, L. Deepface: Closing the gap to human-level performance in face verification. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, 1701–1708 (IEEE, 2014). URL http://ieeexplore. ieee.org/xpls/abs_all.jsp?arnumber=6909616.