Photo Filters Using Neural Style Transfer

Sahib Athwal, Myrelia I. Villa
University of California San Diego
9500 Gilman Dr, La Jolla, CA 92093
sathwal@ucsd.edu, mivilla@ucsd.edu

Abstract

In this course, we have learned the basics of implementing Machine Learning Models using Python. This prompted us to create our own unique project that would utilize some of the libraries including Keras and Tensorflow in order to manipulate pictures with a Neural Style Transfer. Our objective for this project was to create a software that was able to take the style of one photo and project that style onto another content photo, while maintaining the features of the content photo. In this project, it was important to take each of the photos' individual layers, so that the Convolution Neural Network could properly analyze the respective layers of each of photos and combine them together. The optimization of the loss functions allotted us to yield better results in terms of training each of the features from the respective photos. Overall, the results were fairly optimal and going forward the project can be built upon more so by integrating a real time neural transfer style onto a person that might be using a webcam.

Introduction/ Motivation/ Problem Statement

We have seen a large variety of pictures and works of art; from baroque paintings to sepia photographs to abstract art, the different styles give a variety of expressionism. Today, numerous apps can manipulate photos to fit various styles and aesthetics, and it has become a symbol of modern culture with apps such as Instagram and Snapchat. Our hope was to use the Machine Learning concept of Neural Style Transfer to make those style photos without having to manually create that art. Our Neural Style Transfer allows the user to pull any two images, one style and one content, and combine together in such a way that the result image takes the style of the style image while retaining the qualities of the content image. Because we used a pre-trained Convolutional Neural Network model called the VGG-19, the layers of the photos can be seen at different depths with ease. Thus, the more accurate the training from the layers becomes; the better our outcome is in terms of the style portrayal. It is important to note that using the VGG-19 model will train the output to be a more pastel style photo as noted within the documentation.

Methodology/Design

In modern society, we see filters on our phones that allow us to convert a normal photo into an image that looks like a painting or an abstract piece of art in another style. The question is: how does that work? First, a person needs to show the style of the style image, while making sure that the result is still like the original content photo. It is important to know that making the result photo coherent internally is an important objective, but not necessarily the priority for the base case.

The Python imports include the following: Keras, Tensorflow, Numpy, Pandas, and Matplotlib. Keras and Tensorflow allow the access to the neural networks that we use to drive the heart of our project. Numpy and Pandas are used for data manipulation, while Matplotlib is used for data visualization.

Neural Style Transfer Implementation:

1. Import photos through Google Drive as our dataset
2. Then, we read the files into a trained Convolutional Neural Network known as VGG19 in order to classify the layers of the images.
3. After, we used loss functions to be able to properly adjust for the features of each of the respective layers to quantify the style and content. We can account for the adjustment with the Gram Matrix because it shows similarity between three fillers that determine a layer by flattening the layers of the photo. After, we calculated the respective loss functions by using the Gram Matrix, we were able to optimize the photo and train it to be the best output in terms of style that maintains the original content
4. After all of this is done, we can get the output

Discussion

The methodology towards this project was not as simple as the writeup may seem because within the process there were several renditions along the way. The approach to the lab was to understand a specific portion of Machine Learning technology and then implement it in our own unique way. There were several hours and sessions dedicated to going through excellently written Python documentation, which helped facilitate our creation.

We know also based on our results in **Figure 1.** That the Neural Style Transfer is not perfect in cases where the pictures do not match in size. The technology works optimally, when the gradients are near each other and the pictures are of same size **Figure 1.1**.

Completing this project allowed us to see the possibilities of modern machine learning software that can be used in a creative manner. There is still a way to go with how perfect the images can be interpreted, but it's interesting enough that the model we used to train the image interprets the data as a oil pastel image in terms of the result. Other documentation had noted that use of a different model might result in a more pencil like drawing. We believe this project has potential going forward and can be expanded upon more so.

Figure 1 **Results:**

1.1

Best Result

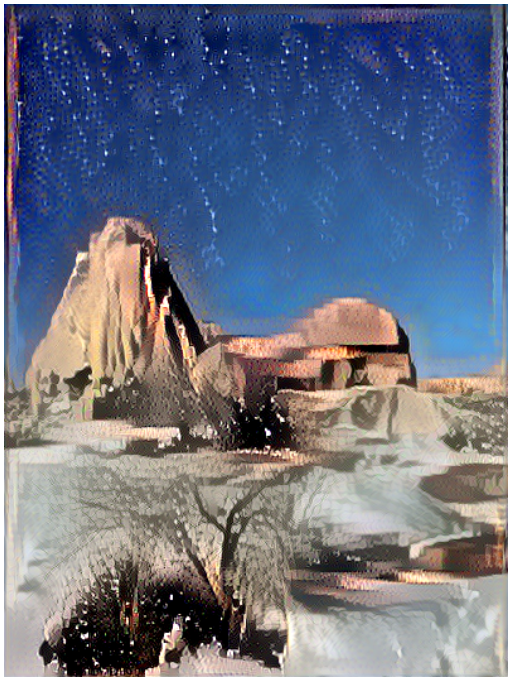
Other Results with mismatched Images

1.2

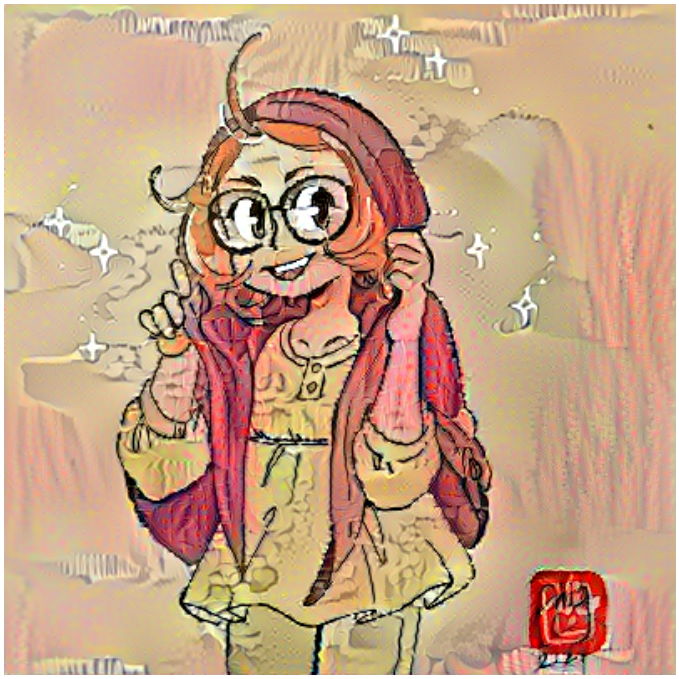
1.3

1.5

1.4



1.6



Conclusion

Overall, our results were solid and exceeded our own personal expectations. This project definitely can be taken further for future endeavors. Other optimizations would include formatting and making the pictures the similar resolution at processing time. On the other hand, some features that would be cool would be to edit the degree of style being applied and other effects to the photo for editing. As well as, the feature to project the image style on to a realitiem projection of your face via a camera of some sort. There is practical application in photo manipulation that Machine Learning algorithms can continue to solve making the creative process much simpler.

References

- TA videos and Class Material

Documentation of Python Libraries

https://keras.io/guides/distributed_training/

https://keras.io/api/applications/vgg/

https://keras.io/api/applications/

https://keras.io/examples/generative/neural_style_transfer/

https://keras.io/ko/applications/

https://www.tensorflow.org/api_docs/python/tf/keras

https://www.tensorflow.org/guide/keras/sequential_model

In [1]:

```
import h5py
import numpy as np
import pandas as pd
import os
import glob
import cv2
import torch
from keras import Model
import subprocess
from IPython.display import Image
from IPython.display import display
import tensorflow as tf
from keras.utils import get_file
import matplotlib.pyplot as plt
from google.colab import drive
import sys
from tensorflow.keras.applications import vgg19
from keras.utils import plot_model
import keras
from datetime import datetime
from keras.optimizers import SGD
import matplotlib.image as mpimg
import matplotlib.pyplot as plt


device_name = tf.test.gpu_device_name()
if device_name != '/device:GPU:0':
  raise SystemError("GPU device not found!")
print('Found GPU at: {}'.format(device_name))

# Define the path to the working directory
drive.mount('/content/drive')



#Show the content images
#Show the style images

#def load_image in order to reshape to make it compatible

#import VGG19 model
```

```
Found GPU at: /device:GPU:0
Mounted at /content/drive
```

In [2]:

```
sys.path.append('/content/drive/MyDrive/AI Picture Style Transfer Examples')
```

In [3]:

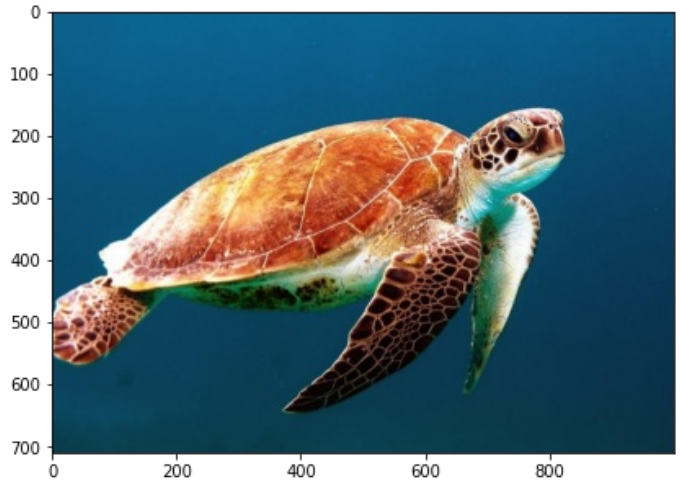
```
result_prefix = "Photo_generated"

style_image_path = get_file(fname="/content/drive/MyDrive/AI Picture Style Transfer Examples/Style/GTA Style.jpg",
                            origin="https://drive.google.com/file/d/1pqy9N7Uaucev3c1oaUBydyGSk6gQe0db/view?usp=sharing")
content_image_path = get_file(fname="/content/drive/MyDrive/AI Picture Style Transfer Examples/Content/Turtle Content.jpg",
                              origin="https://drive.google.com/file/d/1kJBRVsotv3jgvTd4zRm8AsXYv8hskIWc/view?usp=sharing")
```

Turtle Content Vs GTA Style Photograph

In [4]:

In [4]:

```
# read the image file in a numpy array
a = plt.imread(content_image_path)
b = plt.imread(style_image_path)
f, axarr = plt.subplots(1,2, figsize=(15,15))
axarr[0].imshow(a)
axarr[1].imshow(b)
plt.show()
```



In [5]:

```
#Neural Style Transfer

#The Gram Matrix allows us to calculate the style loss on a layer.
#It shows the similarity between the filters and is obtained by
#calculating the dot product between the vectors. Each of the
#calculations will not change based on the size of the layer.
def gram_matrix(x):
    x = tf.transpose(x, (2, 0, 1))
    features = tf.reshape(x, (tf.shape(x)[0], -1))
    gram = tf.matmul(features, tf.transpose(features))
    return gram

#Loss Function of the style
def loss_function_Style(style, combination):
    S = gram_matrix(style)
    C = gram_matrix(combination)
    channels = 3
    size = img_nrows * img_ncols
    return tf.reduce_sum(tf.square(S - C)) / (4.0 * (channels ** 2) * (size ** 2))

#Content Loss function
def content_loss_Function(base, combination):
    return tf.reduce_sum(tf.square(combination - base))

#Load The VGG19 Model
#This is a classification convolutional neural network. The reason already
#created neural networks are used. is because the VGG19 network is trained
#with the ImageNet dataset, which is a neural network offered by Keras.

model = vgg19.VGG19(weights="imagenet", include_top=False)
model.summary()

# Function that extracts the values of that model for some given layers
# so it can used it both for the content error and the style error.
outputs_dict= dict([(layer.name, layer.output) for layer in model.layers])
feature_extractor = Model(inputs=model.inputs, outputs=outputs_dict)

# The Keras package allows us to define: which layers are going to be used to
# calculate the loss function of the style and which layer we are going to use
# to calculate the loss function of the content.


#1. Combine all the images in the same tensioner.
```

```
#2. Get the values in all the layers for the three images.
#3. Initialize the loss vector where we will add the results.
#4. Extract the content layers for the base image and merge and calculate the content los
s function.
#5. Extract the style layers for the style image and the combination image and calculate
the style loss function.

convolution_blocks = [
    "block1_conv1",
    "block2_conv1",
    "block3_conv1",
    "block4_conv1",
    "block5_conv1",
]

convolution_Block_Content = "block5_conv2"

content_weight = 2.5e-8
style_weight = 1e-6

def loss_function(combination_image, base_image, style_reference_image):

    # 1. Combine all the images in the same tensioner.
    input_tensor = tf.concat(
        [base_image, style_reference_image, combination_image], axis=0
    )

    # 2. Get the values in all the layers for the three images.
    features = feature_extractor(input_tensor)

    #3. Inicializar the loss

    loss = tf.zeros(shape=())

    # 4. Extract the content layers + content loss
    layer_features = features[convolution_Block_Content]
    base_image_features = layer_features[0, :, :, :]
    combination_features = layer_features[2, :, :, :]

    loss = loss + content_weight * content_loss_Function(
        base_image_features, combination_features
    )
    # 5. Extract the style layers + style loss
    for layer_name in convolution_blocks:
        layer_features = features[layer_name]
        style_reference_features = layer_features[1, :, :, :]
        combination_features = layer_features[2, :, :, :]
        sl = loss_function_Style(style_reference_features, combination_features)
        loss += (style_weight / len(convolution_blocks)) * sl

    return loss
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg19/
vgg19_weights_tf_dim_ordering_tf_kernels_notop.h5
80142336/80134624 [==============================] - 0s 0us/step
Model: "vgg19"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, None, None, 3)]   0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, None, None, 64)    1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, None, None, 64)    36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, None, None, 64)    0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, None, None, 128)   73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, None, None, 128)   147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, None, None, 128)   0
```

```
block3_conv1 (Conv2D)         (None, None, None, 256)   295168
_________________________________________________________________
block3_conv2 (Conv2D)         (None, None, None, 256)   590080
_________________________________________________________________
block3_conv3 (Conv2D)         (None, None, None, 256)   590080
_________________________________________________________________
block3_conv4 (Conv2D)         (None, None, None, 256)   590080
_________________________________________________________________
block3_pool (MaxPooling2D)    (None, None, None, 256)   0
_________________________________________________________________
block4_conv1 (Conv2D)         (None, None, None, 512)   1180160
_________________________________________________________________
block4_conv2 (Conv2D)         (None, None, None, 512)   2359808
_________________________________________________________________
block4_conv3 (Conv2D)         (None, None, None, 512)   2359808
_________________________________________________________________
block4_conv4 (Conv2D)         (None, None, None, 512)   2359808
_________________________________________________________________
block4_pool (MaxPooling2D)    (None, None, None, 512)   0
_________________________________________________________________
block5_conv1 (Conv2D)         (None, None, None, 512)   2359808
_________________________________________________________________
block5_conv2 (Conv2D)         (None, None, None, 512)   2359808
_________________________________________________________________
block5_conv3 (Conv2D)         (None, None, None, 512)   2359808
_________________________________________________________________
block5_conv4 (Conv2D)         (None, None, None, 512)   2359808
_________________________________________________________________
block5_pool (MaxPooling2D)    (None, None, None, 512)   0
=================================================================
Total params: 20,024,384
Trainable params: 20,024,384
Non-trainable params: 0
_________________________________________________________________
```

In [6]:

```
#Optimization and Gradients

# With the cost function calculate the deltas, which are what gradient descent
# (or any other optimizer) uses to find our optimal values.
#Tensorflow with the GradientTape
# Gradients will only be calculated with the base image
@tf.function
def compute_loss_and_grads(combination_image, base_image, style_reference_image):
    with tf.GradientTape() as tape:
        loss = loss_function(combination_image, base_image, style_reference_image)
    grads = tape.gradient(loss, combination_image)
    return loss, grads
```

In [7]:

```
# Preprocessing of the images consists of giving the images the format that our network r
equires
def preprocess_image(image_path):
    # Util function to open, resize and format pictures into appropriate tensors
    img = keras.preprocessing.image.load_img(
        image_path, target_size=(img_nrows, img_ncols)
    )
    img = keras.preprocessing.image.img_to_array(img)
    img = np.expand_dims(img, axis=0)
    img = vgg19.preprocess_input(img)
    return tf.convert_to_tensor(img)
```

In [8]:

```
def deprocess_image(x):

    # Convert Tensor in Array
    x = x.reshape((img_nrows, img_ncols, 3))
```

```
    # Hacemos que no tengan promedio 0
    x[:, :, 0] += 103.939
    x[:, :, 1] += 116.779
    x[:, :, 2] += 123.68

    # Convert BGR to RGB.
    x = x[:, :, ::-1]

    # There should be no values outside
    x = np.clip(x, 0, 255).astype("uint8")

    return x
```

In [9]:

```
# Training of our Neural Style Transfer network
# Save the generated images
def result_saver(iteration):
  # Create name
  now = datetime.now()
  now = now.strftime("%Y%m%d_%H%M%S")
  model_name = str(i) + '_' + str(now)+"_model_" + '.h5'
  image_name = str(i) + '_' + str(now)+"_image" + '.png'

  # Save image
  img = deprocess_image(combination_image.numpy())
  keras.preprocessing.image.save_img(image_name, img)
```

Turtle GTA Style Image

In [10]:

```
#Run Program
width, height = keras.preprocessing.image.load_img(content_image_path).size
img_nrows = 400
img_ncols = int(width * img_nrows / height)

optimizer = SGD(
    keras.optimizers.schedules.ExponentialDecay(
        initial_learning_rate=100.0, decay_steps=100, decay_rate=0.96
    )
)

content_image = preprocess_image(content_image_path)
style_reference_image = preprocess_image(style_image_path)
combination_image = tf.Variable(preprocess_image(content_image_path))

iterations = 4000

for i in range(1, iterations + 1):
    loss, grads = compute_loss_and_grads(
        combination_image, content_image, style_reference_image
    )
    optimizer.apply_gradients([(grads, combination_image)])
    if i % 10 == 0:
        print("Iteration %d: loss=%.2f" % (i, loss))
        img = deprocess_image(combination_image.numpy())
        fname = result_prefix + "_at_iteration_%d.png" % i
        keras.preprocessing.image.save_img(fname, img)
        result_saver(i)

display(Image(result_prefix + "_at_iteration_4000.png"))
```

```
Iteration 10: loss=9063.15
Iteration 20: loss=5900.16
Iteration 30: loss=4595.76
Iteration 40: loss=3895.05
Iteration 50: loss=3453.85
Iteration 60: loss=3148.08
```

```
Iteration 70: loss=2921.96
Iteration 80: loss=2747.48
Iteration 90: loss=2607.59
Iteration 100: loss=2489.93
Iteration 110: loss=2390.50
Iteration 120: loss=2304.71
Iteration 130: loss=2228.44
Iteration 140: loss=2160.39
Iteration 150: loss=2099.05
Iteration 160: loss=2043.17
Iteration 170: loss=1991.65
Iteration 180: loss=1944.22
Iteration 190: loss=1900.20
Iteration 200: loss=1859.24
Iteration 210: loss=1821.05
Iteration 220: loss=1785.16
Iteration 230: loss=1751.33
Iteration 240: loss=1719.41
Iteration 250: loss=1689.28
Iteration 260: loss=1660.79
Iteration 270: loss=1633.69
Iteration 280: loss=1607.89
Iteration 290: loss=1583.44
Iteration 300: loss=1560.10
Iteration 310: loss=1537.70
Iteration 320: loss=1516.24
Iteration 330: loss=1495.64
Iteration 340: loss=1475.90
Iteration 350: loss=1456.93
Iteration 360: loss=1438.71
Iteration 370: loss=1421.19
Iteration 380: loss=1404.28
Iteration 390: loss=1388.01
Iteration 400: loss=1372.38
Iteration 410: loss=1357.25
Iteration 420: loss=1342.64
Iteration 430: loss=1328.50
Iteration 440: loss=1314.79
Iteration 450: loss=1301.54
Iteration 460: loss=1288.67
Iteration 470: loss=1276.20
Iteration 480: loss=1264.11
Iteration 490: loss=1252.39
Iteration 500: loss=1241.03
Iteration 510: loss=1230.00
Iteration 520: loss=1219.25
Iteration 530: loss=1208.79
Iteration 540: loss=1198.60
Iteration 550: loss=1188.65
Iteration 560: loss=1178.93
Iteration 570: loss=1169.47
Iteration 580: loss=1160.24
Iteration 590: loss=1151.25
Iteration 600: loss=1142.48
Iteration 610: loss=1133.91
Iteration 620: loss=1125.56
Iteration 630: loss=1117.40
Iteration 640: loss=1109.44
Iteration 650: loss=1101.66
Iteration 660: loss=1094.05
Iteration 670: loss=1086.60
Iteration 680: loss=1079.31
Iteration 690: loss=1072.19
Iteration 700: loss=1065.23
Iteration 710: loss=1058.40
Iteration 720: loss=1051.71
Iteration 730: loss=1045.17
Iteration 740: loss=1038.78
Iteration 750: loss=1032.53
Iteration 760: loss=1026.42
Iteration 770: loss=1020.41
Iteration 780: loss=1014.51
```

```
Iteration 790: loss=1008.73
Iteration 800: loss=1003.07
Iteration 810: loss=997.52
Iteration 820: loss=992.08
Iteration 830: loss=986.73
Iteration 840: loss=981.48
Iteration 850: loss=976.33
Iteration 860: loss=971.27
Iteration 870: loss=966.29
Iteration 880: loss=961.41
Iteration 890: loss=956.60
Iteration 900: loss=951.88
Iteration 910: loss=947.24
Iteration 920: loss=942.67
Iteration 930: loss=938.18
Iteration 940: loss=933.76
Iteration 950: loss=929.41
Iteration 960: loss=925.14
Iteration 970: loss=920.93
Iteration 980: loss=916.79
Iteration 990: loss=912.70
Iteration 1000: loss=908.68
Iteration 1010: loss=904.72
Iteration 1020: loss=900.82
Iteration 1030: loss=896.98
Iteration 1040: loss=893.20
Iteration 1050: loss=889.47
Iteration 1060: loss=885.79
Iteration 1070: loss=882.16
Iteration 1080: loss=878.59
Iteration 1090: loss=875.07
Iteration 1100: loss=871.60
Iteration 1110: loss=868.18
Iteration 1120: loss=864.82
Iteration 1130: loss=861.50
Iteration 1140: loss=858.22
Iteration 1150: loss=854.99
Iteration 1160: loss=851.81
Iteration 1170: loss=848.66
Iteration 1180: loss=845.56
Iteration 1190: loss=842.50
Iteration 1200: loss=839.48
Iteration 1210: loss=836.50
Iteration 1220: loss=833.56
Iteration 1230: loss=830.65
Iteration 1240: loss=827.79
Iteration 1250: loss=824.97
Iteration 1260: loss=822.18
Iteration 1270: loss=819.42
Iteration 1280: loss=816.69
Iteration 1290: loss=814.00
Iteration 1300: loss=811.34
Iteration 1310: loss=808.72
Iteration 1320: loss=806.14
Iteration 1330: loss=803.58
Iteration 1340: loss=801.04
Iteration 1350: loss=798.54
Iteration 1360: loss=796.07
Iteration 1370: loss=793.63
Iteration 1380: loss=791.21
Iteration 1390: loss=788.83
Iteration 1400: loss=786.46
Iteration 1410: loss=784.13
Iteration 1420: loss=781.82
Iteration 1430: loss=779.54
Iteration 1440: loss=777.29
Iteration 1450: loss=775.07
Iteration 1460: loss=772.86
Iteration 1470: loss=770.69
Iteration 1480: loss=768.53
Iteration 1490: loss=766.40
Iteration 1500: loss=764.28
```

```
Iteration 1510: loss=762.20
Iteration 1520: loss=760.13
Iteration 1530: loss=758.08
Iteration 1540: loss=756.06
Iteration 1550: loss=754.05
Iteration 1560: loss=752.08
Iteration 1570: loss=750.12
Iteration 1580: loss=748.18
Iteration 1590: loss=746.26
Iteration 1600: loss=744.37
Iteration 1610: loss=742.49
Iteration 1620: loss=740.63
Iteration 1630: loss=738.79
Iteration 1640: loss=736.97
Iteration 1650: loss=735.16
Iteration 1660: loss=733.38
Iteration 1670: loss=731.61
Iteration 1680: loss=729.86
Iteration 1690: loss=728.13
Iteration 1700: loss=726.41
Iteration 1710: loss=724.71
Iteration 1720: loss=723.03
Iteration 1730: loss=721.36
Iteration 1740: loss=719.71
Iteration 1750: loss=718.08
Iteration 1760: loss=716.46
Iteration 1770: loss=714.86
Iteration 1780: loss=713.27
Iteration 1790: loss=711.70
Iteration 1800: loss=710.14
Iteration 1810: loss=708.60
Iteration 1820: loss=707.07
Iteration 1830: loss=705.56
Iteration 1840: loss=704.06
Iteration 1850: loss=702.58
Iteration 1860: loss=701.12
Iteration 1870: loss=699.67
Iteration 1880: loss=698.23
Iteration 1890: loss=696.81
Iteration 1900: loss=695.40
Iteration 1910: loss=694.00
Iteration 1920: loss=692.61
Iteration 1930: loss=691.24
Iteration 1940: loss=689.88
Iteration 1950: loss=688.53
Iteration 1960: loss=687.19
Iteration 1970: loss=685.87
Iteration 1980: loss=684.55
Iteration 1990: loss=683.25
Iteration 2000: loss=681.95
Iteration 2010: loss=680.67
Iteration 2020: loss=679.40
Iteration 2030: loss=678.15
Iteration 2040: loss=676.90
Iteration 2050: loss=675.66
Iteration 2060: loss=674.43
Iteration 2070: loss=673.22
Iteration 2080: loss=672.01
Iteration 2090: loss=670.82
Iteration 2100: loss=669.63
Iteration 2110: loss=668.46
Iteration 2120: loss=667.30
Iteration 2130: loss=666.15
Iteration 2140: loss=665.00
Iteration 2150: loss=663.87
Iteration 2160: loss=662.74
Iteration 2170: loss=661.63
Iteration 2180: loss=660.52
Iteration 2190: loss=659.43
Iteration 2200: loss=658.34
Iteration 2210: loss=657.26
Iteration 2220: loss=656.19
```

```
Iteration 2230: loss=655.12
Iteration 2240: loss=654.07
Iteration 2250: loss=653.02
Iteration 2260: loss=651.99
Iteration 2270: loss=650.96
Iteration 2280: loss=649.94
Iteration 2290: loss=648.93
Iteration 2300: loss=647.93
Iteration 2310: loss=646.93
Iteration 2320: loss=645.95
Iteration 2330: loss=644.97
Iteration 2340: loss=644.00
Iteration 2350: loss=643.03
Iteration 2360: loss=642.08
Iteration 2370: loss=641.13
Iteration 2380: loss=640.19
Iteration 2390: loss=639.26
Iteration 2400: loss=638.33
Iteration 2410: loss=637.42
Iteration 2420: loss=636.51
Iteration 2430: loss=635.60
Iteration 2440: loss=634.70
Iteration 2450: loss=633.81
Iteration 2460: loss=632.93
Iteration 2470: loss=632.06
Iteration 2480: loss=631.19
Iteration 2490: loss=630.33
Iteration 2500: loss=629.47
Iteration 2510: loss=628.62
Iteration 2520: loss=627.78
Iteration 2530: loss=626.95
Iteration 2540: loss=626.12
Iteration 2550: loss=625.29
Iteration 2560: loss=624.48
Iteration 2570: loss=623.66
Iteration 2580: loss=622.86
Iteration 2590: loss=622.06
Iteration 2600: loss=621.27
Iteration 2610: loss=620.48
Iteration 2620: loss=619.70
Iteration 2630: loss=618.92
Iteration 2640: loss=618.15
Iteration 2650: loss=617.39
Iteration 2660: loss=616.63
Iteration 2670: loss=615.88
Iteration 2680: loss=615.13
Iteration 2690: loss=614.39
Iteration 2700: loss=613.65
Iteration 2710: loss=612.92
Iteration 2720: loss=612.20
Iteration 2730: loss=611.48
Iteration 2740: loss=610.77
Iteration 2750: loss=610.06
Iteration 2760: loss=609.36
Iteration 2770: loss=608.66
Iteration 2780: loss=607.96
Iteration 2790: loss=607.27
Iteration 2800: loss=606.59
Iteration 2810: loss=605.91
Iteration 2820: loss=605.24
Iteration 2830: loss=604.57
Iteration 2840: loss=603.90
Iteration 2850: loss=603.24
Iteration 2860: loss=602.59
Iteration 2870: loss=601.94
Iteration 2880: loss=601.29
Iteration 2890: loss=600.65
Iteration 2900: loss=600.01
Iteration 2910: loss=599.38
Iteration 2920: loss=598.75
Iteration 2930: loss=598.12
Iteration 2940: loss=597.50
```

```
Iteration 2950: loss=596.88
Iteration 2960: loss=596.27
Iteration 2970: loss=595.66
Iteration 2980: loss=595.06
Iteration 2990: loss=594.46
Iteration 3000: loss=593.86
Iteration 3010: loss=593.27
Iteration 3020: loss=592.68
Iteration 3030: loss=592.10
Iteration 3040: loss=591.52
Iteration 3050: loss=590.94
Iteration 3060: loss=590.37
Iteration 3070: loss=589.80
Iteration 3080: loss=589.24
Iteration 3090: loss=588.68
Iteration 3100: loss=588.12
Iteration 3110: loss=587.57
Iteration 3120: loss=587.02
Iteration 3130: loss=586.47
Iteration 3140: loss=585.93
Iteration 3150: loss=585.39
Iteration 3160: loss=584.86
Iteration 3170: loss=584.33
Iteration 3180: loss=583.80
Iteration 3190: loss=583.27
Iteration 3200: loss=582.75
Iteration 3210: loss=582.24
Iteration 3220: loss=581.72
Iteration 3230: loss=581.21
Iteration 3240: loss=580.70
Iteration 3250: loss=580.20
Iteration 3260: loss=579.70
Iteration 3270: loss=579.20
Iteration 3280: loss=578.71
Iteration 3290: loss=578.22
Iteration 3300: loss=577.73
Iteration 3310: loss=577.24
Iteration 3320: loss=576.76
Iteration 3330: loss=576.28
Iteration 3340: loss=575.81
Iteration 3350: loss=575.33
Iteration 3360: loss=574.86
Iteration 3370: loss=574.40
Iteration 3380: loss=573.93
Iteration 3390: loss=573.47
Iteration 3400: loss=573.01
Iteration 3410: loss=572.55
Iteration 3420: loss=572.10
Iteration 3430: loss=571.65
Iteration 3440: loss=571.20
Iteration 3450: loss=570.76
Iteration 3460: loss=570.32
Iteration 3470: loss=569.88
Iteration 3480: loss=569.44
Iteration 3490: loss=569.01
Iteration 3500: loss=568.58
Iteration 3510: loss=568.15
Iteration 3520: loss=567.72
Iteration 3530: loss=567.30
Iteration 3540: loss=566.87
Iteration 3550: loss=566.46
Iteration 3560: loss=566.04
Iteration 3570: loss=565.63
Iteration 3580: loss=565.21
Iteration 3590: loss=564.81
Iteration 3600: loss=564.40
Iteration 3610: loss=563.99
Iteration 3620: loss=563.59
Iteration 3630: loss=563.19
Iteration 3640: loss=562.80
Iteration 3650: loss=562.40
Iteration 3660: loss=562.01
```

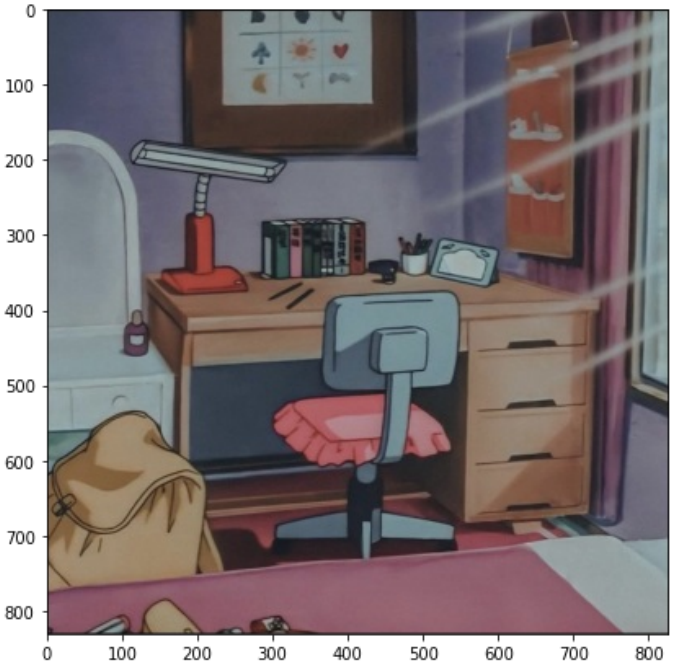
```
Iteration 3670: loss=561.62
Iteration 3680: loss=561.24
Iteration 3690: loss=560.85
Iteration 3700: loss=560.47
Iteration 3710: loss=560.09
Iteration 3720: loss=559.71
Iteration 3730: loss=559.33
Iteration 3740: loss=558.96
Iteration 3750: loss=558.59
Iteration 3760: loss=558.22
Iteration 3770: loss=557.85
Iteration 3780: loss=557.48
Iteration 3790: loss=557.12
Iteration 3800: loss=556.76
Iteration 3810: loss=556.40
Iteration 3820: loss=556.04
Iteration 3830: loss=555.68
Iteration 3840: loss=555.33
Iteration 3850: loss=554.98
Iteration 3860: loss=554.63
Iteration 3870: loss=554.28
Iteration 3880: loss=553.94
Iteration 3890: loss=553.59
Iteration 3900: loss=553.25
Iteration 3910: loss=552.91
Iteration 3920: loss=552.58
Iteration 3930: loss=552.24
Iteration 3940: loss=551.91
Iteration 3950: loss=551.57
Iteration 3960: loss=551.24
Iteration 3970: loss=550.91
Iteration 3980: loss=550.59
Iteration 3990: loss=550.26
Iteration 4000: loss=549.94
```

Hobbit House Lowfi Style Image

In [11]:

```
style_image_path2 = get_file(fname="/content/drive/MyDrive/AI Picture Style Transfer Examples/Style/LowFiStyle.jpg",
                             origin="https://drive.google.com/file/d/1h0MQBr6VO7QyBa-pQbF34Hm3E0Gfw6j2/view?usp=sharing")
content_image_path2 = get_file(fname="/content/drive/MyDrive/AI Picture Style Transfer Examples/Content/HobbitHouse.jpg",
```

```
                                origin="https://drive.google.com/file/d/1xkB0pKCRUqssUKB3u
EG8MI76-EnW744p/view?usp=sharing")

# read the image file in a numpy array
a = plt.imread(content_image_path2)
b = plt.imread(style_image_path2)
f, axarr = plt.subplots(1,2, figsize=(15,15))
axarr[0].imshow(a)
axarr[1].imshow(b)
plt.show()

#Run Program
width, height = keras.preprocessing.image.load_img(content_image_path2).size
img_nrows = 400
img_ncols = int(width * img_nrows / height)

optimizer = SGD(
    keras.optimizers.schedules.ExponentialDecay(
        initial_learning_rate=100.0, decay_steps=100, decay_rate=0.96
    )
)

content_image = preprocess_image(content_image_path2)
style_reference_image = preprocess_image(style_image_path2)
combination_image = tf.Variable(preprocess_image(content_image_path2))

iterations = 4000

for i in range(1, iterations + 1):
    loss, grads = compute_loss_and_grads(
        combination_image, content_image, style_reference_image
    )
    optimizer.apply_gradients([(grads, combination_image)])
    if i % 10 == 0:
        print("Iteration %d: loss=%.2f" % (i, loss))
        img = deprocess_image(combination_image.numpy())
        fname = result_prefix + "_at_iteration_%d.png" % i
        keras.preprocessing.image.save_img(fname, img)
        result_saver(i)

display(Image(result_prefix + "_at_iteration_4000.png"))
```




```
Iteration 10: loss=4917.41
Iteration 20: loss=3025.51
Iteration 30: loss=2269.87
Iteration 40: loss=1846.49
Iteration 50: loss=1572.60
Iteration 60: loss=1378.43
Iteration 70: loss=1232.78
Iteration 80: loss=1118.91
```

Iteration 80: loss=1118.91
Iteration 90: loss=1027.27
Iteration 100: loss=951.67
Iteration 110: loss=888.21
Iteration 120: loss=834.28
Iteration 130: loss=787.76
Iteration 140: loss=747.20
Iteration 150: loss=711.41
Iteration 160: loss=679.64
Iteration 170: loss=651.28
Iteration 180: loss=625.67
Iteration 190: loss=602.46
Iteration 200: loss=581.33
Iteration 210: loss=562.04
Iteration 220: loss=544.35
Iteration 230: loss=528.05
Iteration 240: loss=512.98
Iteration 250: loss=498.97
Iteration 260: loss=485.92
Iteration 270: loss=473.72
Iteration 280: loss=462.30
Iteration 290: loss=451.61
Iteration 300: loss=441.56
Iteration 310: loss=432.10
Iteration 320: loss=423.18
Iteration 330: loss=414.75
Iteration 340: loss=406.79
Iteration 350: loss=399.24
Iteration 360: loss=392.07
Iteration 370: loss=385.23
Iteration 380: loss=378.71
Iteration 390: loss=372.49
Iteration 400: loss=366.55
Iteration 410: loss=360.88
Iteration 420: loss=355.47
Iteration 430: loss=350.28
Iteration 440: loss=345.31
Iteration 450: loss=340.52
Iteration 460: loss=335.93
Iteration 470: loss=331.51
Iteration 480: loss=327.25
Iteration 490: loss=323.14
Iteration 500: loss=319.18
Iteration 510: loss=315.36
Iteration 520: loss=311.67
Iteration 530: loss=308.11
Iteration 540: loss=304.68
Iteration 550: loss=301.35
Iteration 560: loss=298.13
Iteration 570: loss=295.02
Iteration 580: loss=292.00
Iteration 590: loss=289.06
Iteration 600: loss=286.21
Iteration 610: loss=283.44
Iteration 620: loss=280.75
Iteration 630: loss=278.14
Iteration 640: loss=275.60
Iteration 650: loss=273.13
Iteration 660: loss=270.73
Iteration 670: loss=268.40
Iteration 680: loss=266.12
Iteration 690: loss=263.91
Iteration 700: loss=261.75
Iteration 710: loss=259.64
Iteration 720: loss=257.58
Iteration 730: loss=255.58
Iteration 740: loss=253.62
Iteration 750: loss=251.71
Iteration 760: loss=249.85
Iteration 770: loss=248.02
Iteration 780: loss=246.24
Iteration 790: loss=244.51
Iteration 800: loss=242.80

Iteration 800: loss=242.80
Iteration 810: loss=241.14
Iteration 820: loss=239.51
Iteration 830: loss=237.92
Iteration 840: loss=236.36
Iteration 850: loss=234.83
Iteration 860: loss=233.34
Iteration 870: loss=231.88
Iteration 880: loss=230.44
Iteration 890: loss=229.04
Iteration 900: loss=227.67
Iteration 910: loss=226.33
Iteration 920: loss=225.01
Iteration 930: loss=223.72
Iteration 940: loss=222.46
Iteration 950: loss=221.22
Iteration 960: loss=220.01
Iteration 970: loss=218.81
Iteration 980: loss=217.64
Iteration 990: loss=216.49
Iteration 1000: loss=215.36
Iteration 1010: loss=214.25
Iteration 1020: loss=213.15
Iteration 1030: loss=212.08
Iteration 1040: loss=211.03
Iteration 1050: loss=209.99
Iteration 1060: loss=208.97
Iteration 1070: loss=207.96
Iteration 1080: loss=206.98
Iteration 1090: loss=206.01
Iteration 1100: loss=205.06
Iteration 1110: loss=204.12
Iteration 1120: loss=203.19
Iteration 1130: loss=202.28
Iteration 1140: loss=201.39
Iteration 1150: loss=200.51
Iteration 1160: loss=199.64
Iteration 1170: loss=198.79
Iteration 1180: loss=197.95
Iteration 1190: loss=197.12
Iteration 1200: loss=196.31
Iteration 1210: loss=195.51
Iteration 1220: loss=194.72
Iteration 1230: loss=193.94
Iteration 1240: loss=193.17
Iteration 1250: loss=192.42
Iteration 1260: loss=191.67
Iteration 1270: loss=190.94
Iteration 1280: loss=190.21
Iteration 1290: loss=189.50
Iteration 1300: loss=188.80
Iteration 1310: loss=188.10
Iteration 1320: loss=187.41
Iteration 1330: loss=186.74
Iteration 1340: loss=186.07
Iteration 1350: loss=185.41
Iteration 1360: loss=184.76
Iteration 1370: loss=184.12
Iteration 1380: loss=183.49
Iteration 1390: loss=182.87
Iteration 1400: loss=182.25
Iteration 1410: loss=181.64
Iteration 1420: loss=181.05
Iteration 1430: loss=180.45
Iteration 1440: loss=179.87
Iteration 1450: loss=179.29
Iteration 1460: loss=178.72
Iteration 1470: loss=178.16
Iteration 1480: loss=177.60
Iteration 1490: loss=177.05
Iteration 1500: loss=176.51
Iteration 1510: loss=175.97
Iteration 1520: loss=175.44

Iteration 1520: loss=175.44
Iteration 1530: loss=174.92
Iteration 1540: loss=174.40
Iteration 1550: loss=173.89
Iteration 1560: loss=173.38
Iteration 1570: loss=172.88
Iteration 1580: loss=172.39
Iteration 1590: loss=171.90
Iteration 1600: loss=171.41
Iteration 1610: loss=170.94
Iteration 1620: loss=170.47
Iteration 1630: loss=170.00
Iteration 1640: loss=169.54
Iteration 1650: loss=169.08
Iteration 1660: loss=168.63
Iteration 1670: loss=168.19
Iteration 1680: loss=167.75
Iteration 1690: loss=167.31
Iteration 1700: loss=166.88
Iteration 1710: loss=166.46
Iteration 1720: loss=166.04
Iteration 1730: loss=165.62
Iteration 1740: loss=165.21
Iteration 1750: loss=164.81
Iteration 1760: loss=164.41
Iteration 1770: loss=164.01
Iteration 1780: loss=163.62
Iteration 1790: loss=163.23
Iteration 1800: loss=162.84
Iteration 1810: loss=162.46
Iteration 1820: loss=162.09
Iteration 1830: loss=161.72
Iteration 1840: loss=161.35
Iteration 1850: loss=160.98
Iteration 1860: loss=160.62
Iteration 1870: loss=160.27
Iteration 1880: loss=159.91
Iteration 1890: loss=159.56
Iteration 1900: loss=159.22
Iteration 1910: loss=158.87
Iteration 1920: loss=158.53
Iteration 1930: loss=158.20
Iteration 1940: loss=157.86
Iteration 1950: loss=157.54
Iteration 1960: loss=157.21
Iteration 1970: loss=156.89
Iteration 1980: loss=156.57
Iteration 1990: loss=156.25
Iteration 2000: loss=155.94
Iteration 2010: loss=155.63
Iteration 2020: loss=155.32
Iteration 2030: loss=155.02
Iteration 2040: loss=154.72
Iteration 2050: loss=154.42
Iteration 2060: loss=154.12
Iteration 2070: loss=153.83
Iteration 2080: loss=153.54
Iteration 2090: loss=153.25
Iteration 2100: loss=152.97
Iteration 2110: loss=152.69
Iteration 2120: loss=152.41
Iteration 2130: loss=152.13
Iteration 2140: loss=151.86
Iteration 2150: loss=151.58
Iteration 2160: loss=151.32
Iteration 2170: loss=151.05
Iteration 2180: loss=150.78
Iteration 2190: loss=150.52
Iteration 2200: loss=150.26
Iteration 2210: loss=150.00
Iteration 2220: loss=149.75
Iteration 2230: loss=149.50
Iteration 2240: loss=149.24

Iteration 2240: loss=149.24
Iteration 2250: loss=149.00
Iteration 2260: loss=148.75
Iteration 2270: loss=148.51
Iteration 2280: loss=148.26
Iteration 2290: loss=148.02
Iteration 2300: loss=147.79
Iteration 2310: loss=147.55
Iteration 2320: loss=147.32
Iteration 2330: loss=147.09
Iteration 2340: loss=146.86
Iteration 2350: loss=146.63
Iteration 2360: loss=146.40
Iteration 2370: loss=146.18
Iteration 2380: loss=145.96
Iteration 2390: loss=145.74
Iteration 2400: loss=145.52
Iteration 2410: loss=145.30
Iteration 2420: loss=145.09
Iteration 2430: loss=144.88
Iteration 2440: loss=144.67
Iteration 2450: loss=144.46
Iteration 2460: loss=144.25
Iteration 2470: loss=144.04
Iteration 2480: loss=143.84
Iteration 2490: loss=143.64
Iteration 2500: loss=143.44
Iteration 2510: loss=143.24
Iteration 2520: loss=143.04
Iteration 2530: loss=142.85
Iteration 2540: loss=142.65
Iteration 2550: loss=142.46
Iteration 2560: loss=142.27
Iteration 2570: loss=142.08
Iteration 2580: loss=141.89
Iteration 2590: loss=141.71
Iteration 2600: loss=141.52
Iteration 2610: loss=141.34
Iteration 2620: loss=141.16
Iteration 2630: loss=140.98
Iteration 2640: loss=140.80
Iteration 2650: loss=140.62
Iteration 2660: loss=140.44
Iteration 2670: loss=140.27
Iteration 2680: loss=140.10
Iteration 2690: loss=139.92
Iteration 2700: loss=139.75
Iteration 2710: loss=139.58
Iteration 2720: loss=139.42
Iteration 2730: loss=139.25
Iteration 2740: loss=139.08
Iteration 2750: loss=138.92
Iteration 2760: loss=138.76
Iteration 2770: loss=138.59
Iteration 2780: loss=138.43
Iteration 2790: loss=138.28
Iteration 2800: loss=138.12
Iteration 2810: loss=137.96
Iteration 2820: loss=137.80
Iteration 2830: loss=137.65
Iteration 2840: loss=137.50
Iteration 2850: loss=137.35
Iteration 2860: loss=137.19
Iteration 2870: loss=137.04
Iteration 2880: loss=136.90
Iteration 2890: loss=136.75
Iteration 2900: loss=136.60
Iteration 2910: loss=136.45
Iteration 2920: loss=136.31
Iteration 2930: loss=136.17
Iteration 2940: loss=136.02
Iteration 2950: loss=135.88
Iteration 2960: loss=135.74

Iteration 2960: loss=135.74
Iteration 2970: loss=135.60
Iteration 2980: loss=135.46
Iteration 2990: loss=135.32
Iteration 3000: loss=135.19
Iteration 3010: loss=135.05
Iteration 3020: loss=134.91
Iteration 3030: loss=134.78
Iteration 3040: loss=134.65
Iteration 3050: loss=134.51
Iteration 3060: loss=134.38
Iteration 3070: loss=134.25
Iteration 3080: loss=134.12
Iteration 3090: loss=133.99
Iteration 3100: loss=133.87
Iteration 3110: loss=133.74
Iteration 3120: loss=133.61
Iteration 3130: loss=133.49
Iteration 3140: loss=133.37
Iteration 3150: loss=133.24
Iteration 3160: loss=133.12
Iteration 3170: loss=133.00
Iteration 3180: loss=132.88
Iteration 3190: loss=132.76
Iteration 3200: loss=132.64
Iteration 3210: loss=132.52
Iteration 3220: loss=132.40
Iteration 3230: loss=132.29
Iteration 3240: loss=132.17
Iteration 3250: loss=132.05
Iteration 3260: loss=131.94
Iteration 3270: loss=131.83
Iteration 3280: loss=131.71
Iteration 3290: loss=131.60
Iteration 3300: loss=131.49
Iteration 3310: loss=131.38
Iteration 3320: loss=131.27
Iteration 3330: loss=131.16
Iteration 3340: loss=131.05
Iteration 3350: loss=130.94
Iteration 3360: loss=130.84
Iteration 3370: loss=130.73
Iteration 3380: loss=130.62
Iteration 3390: loss=130.52
Iteration 3400: loss=130.42
Iteration 3410: loss=130.31
Iteration 3420: loss=130.21
Iteration 3430: loss=130.11
Iteration 3440: loss=130.01
Iteration 3450: loss=129.90
Iteration 3460: loss=129.80
Iteration 3470: loss=129.70
Iteration 3480: loss=129.61
Iteration 3490: loss=129.51
Iteration 3500: loss=129.41
Iteration 3510: loss=129.31
Iteration 3520: loss=129.22
Iteration 3530: loss=129.12
Iteration 3540: loss=129.02
Iteration 3550: loss=128.93
Iteration 3560: loss=128.84
Iteration 3570: loss=128.74
Iteration 3580: loss=128.65
Iteration 3590: loss=128.56
Iteration 3600: loss=128.47
Iteration 3610: loss=128.37
Iteration 3620: loss=128.28
Iteration 3630: loss=128.19
Iteration 3640: loss=128.10
Iteration 3650: loss=128.02
Iteration 3660: loss=127.93
Iteration 3670: loss=127.84
Iteration 3680: loss=127.75

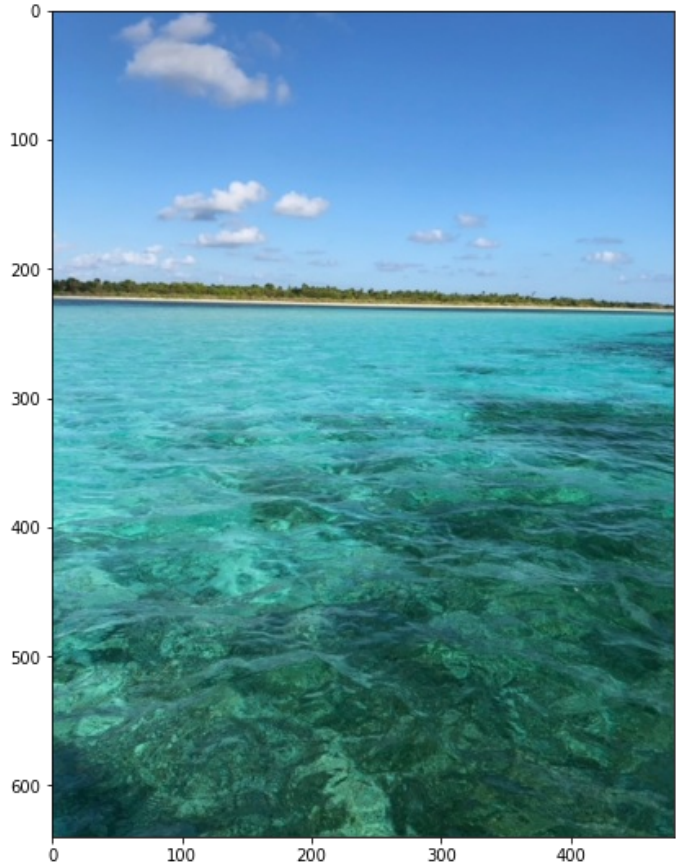
```
Iteration 3680: loss=127.75
Iteration 3690: loss=127.66
Iteration 3700: loss=127.58
Iteration 3710: loss=127.49
Iteration 3720: loss=127.41
Iteration 3730: loss=127.32
Iteration 3740: loss=127.24
Iteration 3750: loss=127.16
Iteration 3760: loss=127.07
Iteration 3770: loss=126.99
Iteration 3780: loss=126.91
Iteration 3790: loss=126.83
Iteration 3800: loss=126.74
Iteration 3810: loss=126.66
Iteration 3820: loss=126.58
Iteration 3830: loss=126.50
Iteration 3840: loss=126.42
Iteration 3850: loss=126.35
Iteration 3860: loss=126.27
Iteration 3870: loss=126.19
Iteration 3880: loss=126.11
Iteration 3890: loss=126.04
Iteration 3900: loss=125.96
Iteration 3910: loss=125.88
Iteration 3920: loss=125.81
Iteration 3930: loss=125.73
Iteration 3940: loss=125.66
Iteration 3950: loss=125.59
Iteration 3960: loss=125.51
Iteration 3970: loss=125.44
Iteration 3980: loss=125.37
Iteration 3990: loss=125.29
Iteration 4000: loss=125.22
```

Beach Orange Tree Style Image

In [ ]:

```
style_image_path3 = get_file(fname="/content/drive/MyDrive/AI Picture Style Transfer Examples/Style/OrangeTreeStyle.jpg",
                             origin="https://drive.google.com/file/d/1h0MQBr6VO7QyBa-pQbF34Hm3E0Gfw6j2/view?usp=sharing")
content_image_path3 = get_file(fname="/content/drive/MyDrive/AI Picture Style Transfer Examples/Content/Beach.jpg",
                               origin="https://drive.google.com/file/d/1xkB0pKCRUqssUKB3u
```

```
EG8MI76-EnW744p/view?usp=sharing")

# read the image file in a numpy array
a = plt.imread(content_image_path3)
b = plt.imread(style_image_path3)
f, axarr = plt.subplots(1,2, figsize=(15,15))
axarr[0].imshow(a)
axarr[1].imshow(b)
plt.show()

#Run Program
width, height = keras.preprocessing.image.load_img(content_image_path3).size
img_nrows = 400
img_ncols = int(width * img_nrows / height)

optimizer = SGD(
    keras.optimizers.schedules.ExponentialDecay(
        initial_learning_rate=100.0, decay_steps=100, decay_rate=0.96
    )
)

content_image = preprocess_image(content_image_path3)
style_reference_image = preprocess_image(style_image_path3)
combination_image = tf.Variable(preprocess_image(content_image_path3))

iterations = 4000

for i in range(1, iterations + 1):
    loss, grads = compute_loss_and_grads(
        combination_image, content_image, style_reference_image
    )
    optimizer.apply_gradients([(grads, combination_image)])
    if i % 10 == 0:
        print("Iteration %d: loss=%.2f" % (i, loss))
        img = deprocess_image(combination_image.numpy())
        fname = result_prefix + "_at_iteration_%d.png" % i
        keras.preprocessing.image.save_img(fname, img)
        result_saver(i)

display(Image(result_prefix + "_at_iteration_4000.png"))
```




```
Iteration 10: loss=17065.53
Iteration 20: loss=9407.05
```

```
Iteration 30: loss=6232.47
Iteration 40: loss=5657.21
Iteration 50: loss=5902.86
Iteration 60: loss=3903.02
Iteration 70: loss=3502.08
Iteration 80: loss=3481.25
Iteration 90: loss=3309.30
Iteration 100: loss=2817.99
Iteration 110: loss=2740.84
Iteration 120: loss=2659.12
Iteration 130: loss=2459.68
Iteration 140: loss=2392.93
Iteration 150: loss=2589.96
Iteration 160: loss=2252.67
Iteration 170: loss=2111.08
Iteration 180: loss=2035.72
Iteration 190: loss=2109.41
Iteration 200: loss=1956.87
Iteration 210: loss=1872.74
Iteration 220: loss=1851.56
Iteration 230: loss=1841.75
Iteration 240: loss=1743.47
Iteration 250: loss=1716.68
Iteration 260: loss=1716.77
Iteration 270: loss=1640.49
Iteration 280: loss=1609.76
Iteration 290: loss=1603.93
Iteration 300: loss=1556.26
Iteration 310: loss=1519.98
Iteration 320: loss=1508.53
Iteration 330: loss=1484.64
Iteration 340: loss=1446.16
Iteration 350: loss=1429.33
Iteration 360: loss=1415.87
Iteration 370: loss=1384.27
Iteration 380: loss=1364.40
Iteration 390: loss=1353.30
Iteration 400: loss=1329.80
Iteration 410: loss=1309.19
Iteration 420: loss=1297.34
Iteration 430: loss=1280.63
Iteration 440: loss=1260.58
Iteration 450: loss=1247.47
Iteration 460: loss=1234.43
Iteration 470: loss=1217.32
Iteration 480: loss=1202.78
Iteration 490: loss=1191.18
Iteration 500: loss=1178.12
Iteration 510: loss=1164.53
Iteration 520: loss=1152.84
Iteration 530: loss=1141.76
Iteration 540: loss=1129.74
Iteration 550: loss=1118.00
Iteration 560: loss=1107.39
Iteration 570: loss=1097.01
Iteration 580: loss=1085.77
Iteration 590: loss=1075.62
Iteration 600: loss=1066.43
Iteration 610: loss=1056.88
Iteration 620: loss=1047.09
Iteration 630: loss=1038.36
Iteration 640: loss=1029.54
Iteration 650: loss=1020.90
Iteration 660: loss=1012.43
Iteration 670: loss=1004.26
Iteration 680: loss=996.27
Iteration 690: loss=988.55
Iteration 700: loss=981.00
Iteration 710: loss=973.42
Iteration 720: loss=966.26
Iteration 730: loss=959.10
Iteration 740: loss=952.20
```

```
Iteration 750: loss=945.41
Iteration 760: loss=938.72
Iteration 770: loss=932.19
Iteration 780: loss=925.70
Iteration 790: loss=919.39
Iteration 800: loss=913.36
Iteration 810: loss=907.39
Iteration 820: loss=901.41
Iteration 830: loss=895.43
Iteration 840: loss=889.80
Iteration 850: loss=884.29
Iteration 860: loss=878.74
Iteration 870: loss=873.33
Iteration 880: loss=868.05
Iteration 890: loss=862.80
Iteration 900: loss=857.61
Iteration 910: loss=852.73
Iteration 920: loss=847.98
Iteration 930: loss=843.13
Iteration 940: loss=838.32
Iteration 950: loss=833.67
Iteration 960: loss=829.09
Iteration 970: loss=824.57
Iteration 980: loss=820.14
Iteration 990: loss=815.84
Iteration 1000: loss=811.54
Iteration 1010: loss=807.35
Iteration 1020: loss=803.27
Iteration 1030: loss=799.28
Iteration 1040: loss=795.24
Iteration 1050: loss=791.29
Iteration 1060: loss=787.40
Iteration 1070: loss=783.56
Iteration 1080: loss=779.80
Iteration 1090: loss=776.15
Iteration 1100: loss=772.51
Iteration 1110: loss=768.93
Iteration 1120: loss=765.39
Iteration 1130: loss=761.90
Iteration 1140: loss=758.50
Iteration 1150: loss=755.11
Iteration 1160: loss=751.78
Iteration 1170: loss=748.53
Iteration 1180: loss=745.29
Iteration 1190: loss=742.09
Iteration 1200: loss=738.96
Iteration 1210: loss=735.85
Iteration 1220: loss=732.79
Iteration 1230: loss=729.79
Iteration 1240: loss=726.83
Iteration 1250: loss=723.91
Iteration 1260: loss=721.02
Iteration 1270: loss=718.19
Iteration 1280: loss=715.39
Iteration 1290: loss=712.60
Iteration 1300: loss=709.87
Iteration 1310: loss=707.19
Iteration 1320: loss=704.56
Iteration 1330: loss=701.94
Iteration 1340: loss=699.35
Iteration 1350: loss=696.82
Iteration 1360: loss=694.32
Iteration 1370: loss=691.83
Iteration 1380: loss=689.39
Iteration 1390: loss=686.98
Iteration 1400: loss=684.59
Iteration 1410: loss=682.24
Iteration 1420: loss=679.92
Iteration 1430: loss=677.62
Iteration 1440: loss=675.33
Iteration 1450: loss=673.09
Iteration 1460: loss=670.88
```

```
Iteration 1470: loss=668.70
Iteration 1480: loss=666.54
Iteration 1490: loss=664.40
Iteration 1500: loss=662.29
Iteration 1510: loss=660.21
Iteration 1520: loss=658.16
Iteration 1530: loss=656.12
Iteration 1540: loss=654.11
Iteration 1550: loss=652.12
Iteration 1560: loss=650.15
Iteration 1570: loss=648.20
Iteration 1580: loss=646.28
Iteration 1590: loss=644.37
Iteration 1600: loss=642.49
Iteration 1610: loss=640.62
Iteration 1620: loss=638.78
Iteration 1630: loss=636.95
Iteration 1640: loss=635.15
Iteration 1650: loss=633.36
Iteration 1660: loss=631.59
Iteration 1670: loss=629.84
Iteration 1680: loss=628.11
Iteration 1690: loss=626.40
Iteration 1700: loss=624.71
Iteration 1710: loss=623.04
Iteration 1720: loss=621.39
Iteration 1730: loss=619.76
Iteration 1740: loss=618.14
Iteration 1750: loss=616.55
Iteration 1760: loss=614.97
Iteration 1770: loss=613.40
Iteration 1780: loss=611.86
Iteration 1790: loss=610.32
Iteration 1800: loss=608.81
Iteration 1810: loss=607.31
Iteration 1820: loss=605.82
Iteration 1830: loss=604.35
Iteration 1840: loss=602.90
Iteration 1850: loss=601.45
Iteration 1860: loss=600.02
Iteration 1870: loss=598.61
Iteration 1880: loss=597.20
Iteration 1890: loss=595.82
Iteration 1900: loss=594.44
Iteration 1910: loss=593.08
Iteration 1920: loss=591.72
Iteration 1930: loss=590.38
Iteration 1940: loss=589.05
Iteration 1950: loss=587.74
Iteration 1960: loss=586.43
Iteration 1970: loss=585.14
Iteration 1980: loss=583.86
Iteration 1990: loss=582.59
Iteration 2000: loss=581.34
Iteration 2010: loss=580.10
Iteration 2020: loss=578.87
Iteration 2030: loss=577.65
Iteration 2040: loss=576.45
Iteration 2050: loss=575.26
Iteration 2060: loss=574.07
Iteration 2070: loss=572.90
Iteration 2080: loss=571.74
Iteration 2090: loss=570.58
Iteration 2100: loss=569.44
Iteration 2110: loss=568.31
Iteration 2120: loss=567.19
Iteration 2130: loss=566.08
Iteration 2140: loss=564.98
Iteration 2150: loss=563.89
Iteration 2160: loss=562.80
Iteration 2170: loss=561.73
Iteration 2180: loss=560.67
```

```
Iteration 2190: loss=559.62
Iteration 2200: loss=558.57
Iteration 2210: loss=557.54
Iteration 2220: loss=556.51
Iteration 2230: loss=555.49
Iteration 2240: loss=554.49
Iteration 2250: loss=553.48
Iteration 2260: loss=552.49
Iteration 2270: loss=551.51
Iteration 2280: loss=550.53
Iteration 2290: loss=549.56
Iteration 2300: loss=548.60
Iteration 2310: loss=547.65
Iteration 2320: loss=546.70
Iteration 2330: loss=545.76
Iteration 2340: loss=544.83
Iteration 2350: loss=543.91
Iteration 2360: loss=542.99
Iteration 2370: loss=542.07
Iteration 2380: loss=541.17
Iteration 2390: loss=540.27
Iteration 2400: loss=539.38
Iteration 2410: loss=538.49
Iteration 2420: loss=537.62
Iteration 2430: loss=536.75
Iteration 2440: loss=535.88
Iteration 2450: loss=535.03
Iteration 2460: loss=534.18
Iteration 2470: loss=533.34
Iteration 2480: loss=532.50
Iteration 2490: loss=531.68
Iteration 2500: loss=530.86
Iteration 2510: loss=530.04
Iteration 2520: loss=529.23
Iteration 2530: loss=528.43
Iteration 2540: loss=527.63
Iteration 2550: loss=526.83
Iteration 2560: loss=526.05
Iteration 2570: loss=525.27
Iteration 2580: loss=524.49
Iteration 2590: loss=523.73
Iteration 2600: loss=522.96
Iteration 2610: loss=522.21
Iteration 2620: loss=521.46
Iteration 2630: loss=520.71
Iteration 2640: loss=519.97
Iteration 2650: loss=519.24
Iteration 2660: loss=518.51
Iteration 2670: loss=517.78
Iteration 2680: loss=517.06
Iteration 2690: loss=516.35
Iteration 2700: loss=515.64
Iteration 2710: loss=514.93
Iteration 2720: loss=514.23
Iteration 2730: loss=513.54
Iteration 2740: loss=512.85
Iteration 2750: loss=512.17
Iteration 2760: loss=511.49
Iteration 2770: loss=510.81
Iteration 2780: loss=510.14
Iteration 2790: loss=509.48
Iteration 2800: loss=508.82
Iteration 2810: loss=508.17
Iteration 2820: loss=507.51
Iteration 2830: loss=506.87
Iteration 2840: loss=506.23
Iteration 2850: loss=505.59
Iteration 2860: loss=504.96
Iteration 2870: loss=504.33
Iteration 2880: loss=503.70
Iteration 2890: loss=503.08
Iteration 2900: loss=502.47
```

```
Iteration 2910: loss=501.86
Iteration 2920: loss=501.25
Iteration 2930: loss=500.65
Iteration 2940: loss=500.05
Iteration 2950: loss=499.45
Iteration 2960: loss=498.86
Iteration 2970: loss=498.28
Iteration 2980: loss=497.69
Iteration 2990: loss=497.12
Iteration 3000: loss=496.54
Iteration 3010: loss=495.97
Iteration 3020: loss=495.40
Iteration 3030: loss=494.84
Iteration 3040: loss=494.28
Iteration 3050: loss=493.72
Iteration 3060: loss=493.17
Iteration 3070: loss=492.62
Iteration 3080: loss=492.08
Iteration 3090: loss=491.54
Iteration 3100: loss=491.00
Iteration 3110: loss=490.47
Iteration 3120: loss=489.93
Iteration 3130: loss=489.41
Iteration 3140: loss=488.88
Iteration 3150: loss=488.36
Iteration 3160: loss=487.84
Iteration 3170: loss=487.33
Iteration 3180: loss=486.82
Iteration 3190: loss=486.31
Iteration 3200: loss=485.80
Iteration 3210: loss=485.30
Iteration 3220: loss=484.80
Iteration 3230: loss=484.30
Iteration 3240: loss=483.81
Iteration 3250: loss=483.32
Iteration 3260: loss=482.83
Iteration 3270: loss=482.35
Iteration 3280: loss=481.87
Iteration 3290: loss=481.39
Iteration 3300: loss=480.91
Iteration 3310: loss=480.44
Iteration 3320: loss=479.97
Iteration 3330: loss=479.50
Iteration 3340: loss=479.04
Iteration 3350: loss=478.58
Iteration 3360: loss=478.12
Iteration 3370: loss=477.66
Iteration 3380: loss=477.21
Iteration 3390: loss=476.76
Iteration 3400: loss=476.32
Iteration 3410: loss=475.87
Iteration 3420: loss=475.43
Iteration 3430: loss=474.99
Iteration 3440: loss=474.56
Iteration 3450: loss=474.12
Iteration 3460: loss=473.69
Iteration 3470: loss=473.27
Iteration 3480: loss=472.84
Iteration 3490: loss=472.42
Iteration 3500: loss=472.00
Iteration 3510: loss=471.58
Iteration 3520: loss=471.17
Iteration 3530: loss=470.76
Iteration 3540: loss=470.35
Iteration 3550: loss=469.94
Iteration 3560: loss=469.54
Iteration 3570: loss=469.14
Iteration 3580: loss=468.74
Iteration 3590: loss=468.34
Iteration 3600: loss=467.95
Iteration 3610: loss=467.56
Iteration 3620: loss=467.17
```

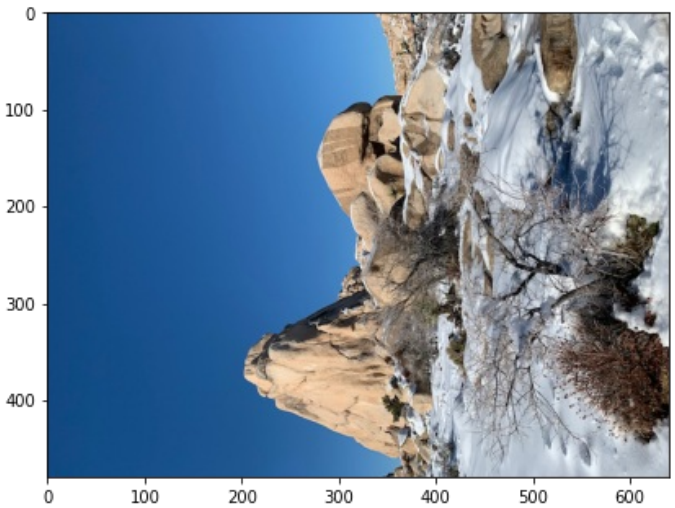
```
Iteration 3630: loss=466.78
Iteration 3640: loss=466.39
Iteration 3650: loss=466.01
Iteration 3660: loss=465.63
Iteration 3670: loss=465.25
Iteration 3680: loss=464.88
Iteration 3690: loss=464.50
Iteration 3700: loss=464.13
Iteration 3710: loss=463.76
Iteration 3720: loss=463.39
Iteration 3730: loss=463.03
Iteration 3740: loss=462.66
Iteration 3750: loss=462.30
Iteration 3760: loss=461.94
Iteration 3770: loss=461.59
Iteration 3780: loss=461.23
Iteration 3790: loss=460.88
Iteration 3800: loss=460.53
Iteration 3810: loss=460.18
Iteration 3820: loss=459.83
Iteration 3830: loss=459.48
Iteration 3840: loss=459.14
Iteration 3850: loss=458.80
Iteration 3860: loss=458.46
Iteration 3870: loss=458.12
Iteration 3880: loss=457.78
Iteration 3890: loss=457.45
Iteration 3900: loss=457.12
Iteration 3910: loss=456.79
Iteration 3920: loss=456.46
Iteration 3930: loss=456.13
Iteration 3940: loss=455.80
Iteration 3950: loss=455.48
Iteration 3960: loss=455.16
Iteration 3970: loss=454.84
Iteration 3980: loss=454.52
Iteration 3990: loss=454.20
Iteration 4000: loss=453.89
```

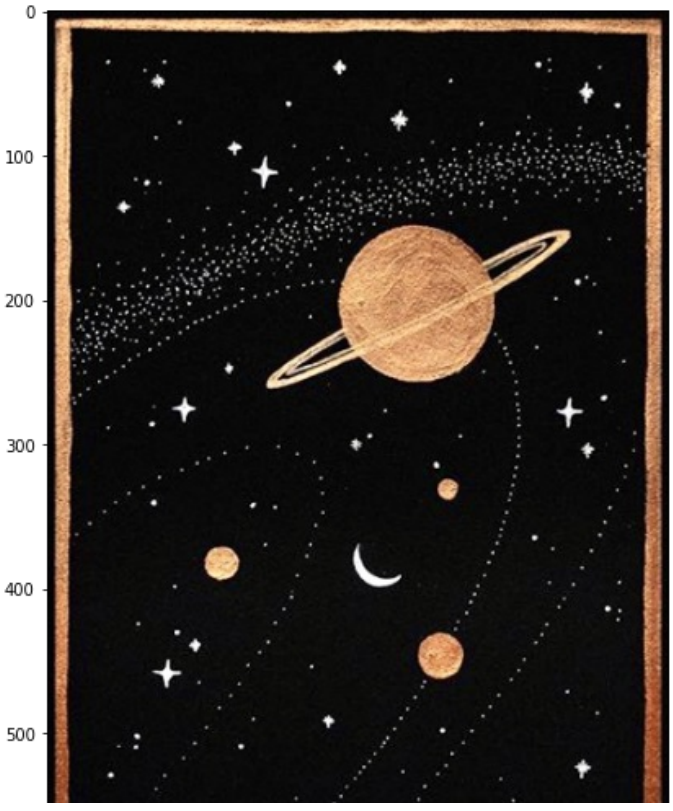
Rock Space Light Style Image

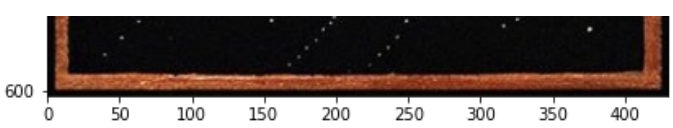
In [ ]:

```
style_image_path4 = get_file(fname="/content/drive/MyDrive/AI Picture Style Transfer Examples/Style/SpaceStyleLight.jpg",
```

```
                                origin="https://drive.google.com/file/d/1h0MQBr6VO7QyBa-pQbF
34Hm3E0Gfw6j2/view?usp=sharing")
content_image_path4 = get_file(fname="/content/drive/MyDrive/AI Picture Style Transfer Ex
amples/Content/Rock.jpg",
                               origin="https://drive.google.com/file/d/1xkB0pKCRUqssUKB3u
EG8MI76-EnW744p/view?usp=sharing")

# read the image file in a numpy array
a = plt.imread(content_image_path4)
b = plt.imread(style_image_path4)
f, axarr = plt.subplots(1,2, figsize=(15,15))
axarr[0].imshow(a)
axarr[1].imshow(b)
plt.show()

#Run Program
width, height = keras.preprocessing.image.load_img(content_image_path4).size
img_nrows = 400
img_ncols = int(width * img_nrows / height)

optimizer = SGD(
    keras.optimizers.schedules.ExponentialDecay(
        initial_learning_rate=100.0, decay_steps=100, decay_rate=0.96
    )
)

content_image = preprocess_image(content_image_path4)
style_reference_image = preprocess_image(style_image_path4)
combination_image = tf.Variable(preprocess_image(content_image_path4))

iterations = 4000

for i in range(1, iterations + 1):
    loss, grads = compute_loss_and_grads(
        combination_image, content_image, style_reference_image
    )
    optimizer.apply_gradients([(grads, combination_image)])
    if i % 10 == 0:
        print("Iteration %d: loss=%.2f" % (i, loss))
        img = deprocess_image(combination_image.numpy())
        fname = result_prefix + "_at_iteration_%d.png" % i
        keras.preprocessing.image.save_img(fname, img)
        result_saver(i)

display(Image(result_prefix + "_at_iteration_4000.png"))
```








```
Iteration 10: loss=21507.53
Iteration 20: loss=15577.63
Iteration 30: loss=12591.28
Iteration 40: loss=10844.58
Iteration 50: loss=9652.89
Iteration 60: loss=8785.48
Iteration 70: loss=8121.65
Iteration 80: loss=7591.17
Iteration 90: loss=7163.06
Iteration 100: loss=6801.87
Iteration 110: loss=6492.71
Iteration 120: loss=6223.74
Iteration 130: loss=5985.47
Iteration 140: loss=5774.69
Iteration 150: loss=5585.30
Iteration 160: loss=5417.79
Iteration 170: loss=5262.68
Iteration 180: loss=5122.17
Iteration 190: loss=4991.59
Iteration 200: loss=4871.12
Iteration 210: loss=4760.68
Iteration 220: loss=4656.48
Iteration 230: loss=4561.09
Iteration 240: loss=4470.98
Iteration 250: loss=4386.53
Iteration 260: loss=4308.19
Iteration 270: loss=4233.39
Iteration 280: loss=4163.32
Iteration 290: loss=4096.70
Iteration 300: loss=4033.77
Iteration 310: loss=3973.64
Iteration 320: loss=3916.19
Iteration 330: loss=3861.50
Iteration 340: loss=3809.40
Iteration 350: loss=3759.57
Iteration 360: loss=3710.50
Iteration 370: loss=3664.99
Iteration 380: loss=3621.92
Iteration 390: loss=3579.77
Iteration 400: loss=3539.43
Iteration 410: loss=3500.19
Iteration 420: loss=3462.84
Iteration 430: loss=3426.25
Iteration 440: loss=3391.17
Iteration 450: loss=3357.22
Iteration 460: loss=3324.34
Iteration 470: loss=3292.75
Iteration 480: loss=3262.18
Iteration 490: loss=3232.69
Iteration 500: loss=3204.14
Iteration 510: loss=3176.12
Iteration 520: loss=3149.35
Iteration 530: loss=3123.23
Iteration 540: loss=3097.84
Iteration 550: loss=3073.29
Iteration 560: loss=3049.11
Iteration 570: loss=3025.65
Iteration 580: loss=3003.01
Iteration 590: loss=2980.62
Iteration 600: loss=2959.11
Iteration 610: loss=2938.01
Iteration 620: loss=2917.40
Iteration 630: loss=2897.34
Iteration 640: loss=2877.58
Iteration 650: loss=2858.64
Iteration 660: loss=2840.07
Iteration 670: loss=2821.81
Iteration 680: loss=2804.17
```

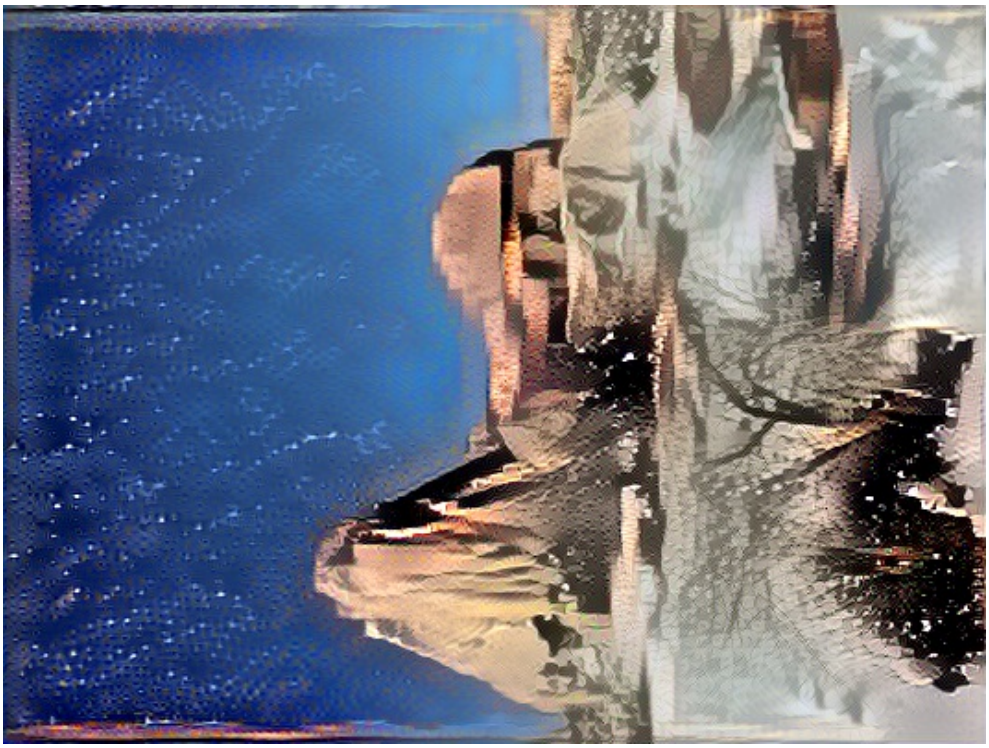
Iteration 680: loss=2804.17
Iteration 690: loss=2786.75
Iteration 700: loss=2769.73
Iteration 710: loss=2753.17
Iteration 720: loss=2736.92
Iteration 730: loss=2721.00
Iteration 740: loss=2705.49
Iteration 750: loss=2690.25
Iteration 760: loss=2675.29
Iteration 770: loss=2660.70
Iteration 780: loss=2646.42
Iteration 790: loss=2632.34
Iteration 800: loss=2618.64
Iteration 810: loss=2605.06
Iteration 820: loss=2591.87
Iteration 830: loss=2578.92
Iteration 840: loss=2566.08
Iteration 850: loss=2553.53
Iteration 860: loss=2541.31
Iteration 870: loss=2529.17
Iteration 880: loss=2517.16
Iteration 890: loss=2505.54
Iteration 900: loss=2494.02
Iteration 910: loss=2482.76
Iteration 920: loss=2471.68
Iteration 930: loss=2460.75
Iteration 940: loss=2450.09
Iteration 950: loss=2439.55
Iteration 960: loss=2429.28
Iteration 970: loss=2419.11
Iteration 980: loss=2409.21
Iteration 990: loss=2399.38
Iteration 1000: loss=2389.72
Iteration 1010: loss=2380.27
Iteration 1020: loss=2370.89
Iteration 1030: loss=2361.68
Iteration 1040: loss=2352.64
Iteration 1050: loss=2343.66
Iteration 1060: loss=2334.88
Iteration 1070: loss=2326.21
Iteration 1080: loss=2317.68
Iteration 1090: loss=2309.28
Iteration 1100: loss=2300.97
Iteration 1110: loss=2292.74
Iteration 1120: loss=2284.66
Iteration 1130: loss=2276.67
Iteration 1140: loss=2268.82
Iteration 1150: loss=2261.10
Iteration 1160: loss=2253.53
Iteration 1170: loss=2245.97
Iteration 1180: loss=2238.53
Iteration 1190: loss=2231.21
Iteration 1200: loss=2223.90
Iteration 1210: loss=2216.75
Iteration 1220: loss=2209.69
Iteration 1230: loss=2202.65
Iteration 1240: loss=2195.72
Iteration 1250: loss=2188.83
Iteration 1260: loss=2182.07
Iteration 1270: loss=2175.31
Iteration 1280: loss=2168.74
Iteration 1290: loss=2162.22
Iteration 1300: loss=2155.76
Iteration 1310: loss=2149.39
Iteration 1320: loss=2143.10
Iteration 1330: loss=2136.89
Iteration 1340: loss=2130.75
Iteration 1350: loss=2124.73
Iteration 1360: loss=2118.75
Iteration 1370: loss=2112.85
Iteration 1380: loss=2107.02
Iteration 1390: loss=2101.26
Iteration 1400: loss=2095.55

```
Iteration 1400: loss=2095.55
Iteration 1410: loss=2089.90
Iteration 1420: loss=2084.30
Iteration 1430: loss=2078.77
Iteration 1440: loss=2073.29
Iteration 1450: loss=2067.85
Iteration 1460: loss=2062.46
Iteration 1470: loss=2057.16
Iteration 1480: loss=2051.93
Iteration 1490: loss=2046.76
Iteration 1500: loss=2041.64
Iteration 1510: loss=2036.60
Iteration 1520: loss=2031.58
Iteration 1530: loss=2026.65
Iteration 1540: loss=2021.72
Iteration 1550: loss=2016.84
Iteration 1560: loss=2012.01
Iteration 1570: loss=2007.22
Iteration 1580: loss=2002.50
Iteration 1590: loss=1997.86
Iteration 1600: loss=1993.22
Iteration 1610: loss=1988.66
Iteration 1620: loss=1984.16
Iteration 1630: loss=1979.67
Iteration 1640: loss=1975.23
Iteration 1650: loss=1970.83
Iteration 1660: loss=1966.49
Iteration 1670: loss=1962.17
Iteration 1680: loss=1957.91
Iteration 1690: loss=1953.69
Iteration 1700: loss=1949.51
Iteration 1710: loss=1945.36
Iteration 1720: loss=1941.24
Iteration 1730: loss=1937.16
Iteration 1740: loss=1933.15
Iteration 1750: loss=1929.15
Iteration 1760: loss=1925.22
Iteration 1770: loss=1921.33
Iteration 1780: loss=1917.46
Iteration 1790: loss=1913.61
Iteration 1800: loss=1909.81
Iteration 1810: loss=1906.08
Iteration 1820: loss=1902.35
Iteration 1830: loss=1898.68
Iteration 1840: loss=1895.06
Iteration 1850: loss=1891.45
Iteration 1860: loss=1887.89
Iteration 1870: loss=1884.37
Iteration 1880: loss=1880.84
Iteration 1890: loss=1877.36
Iteration 1900: loss=1873.93
Iteration 1910: loss=1870.50
Iteration 1920: loss=1867.12
Iteration 1930: loss=1863.75
Iteration 1940: loss=1860.44
Iteration 1950: loss=1857.15
Iteration 1960: loss=1853.87
Iteration 1970: loss=1850.64
Iteration 1980: loss=1847.42
Iteration 1990: loss=1844.25
Iteration 2000: loss=1841.12
Iteration 2010: loss=1837.99
Iteration 2020: loss=1834.91
Iteration 2030: loss=1831.86
Iteration 2040: loss=1828.83
Iteration 2050: loss=1825.82
Iteration 2060: loss=1822.85
Iteration 2070: loss=1819.91
Iteration 2080: loss=1816.98
Iteration 2090: loss=1814.06
Iteration 2100: loss=1811.17
Iteration 2110: loss=1808.31
Iteration 2120: loss=1805.48
```

Iteration 2120: loss=1805.48
Iteration 2130: loss=1802.68
Iteration 2140: loss=1799.90
Iteration 2150: loss=1797.14
Iteration 2160: loss=1794.40
Iteration 2170: loss=1791.70
Iteration 2180: loss=1789.02
Iteration 2190: loss=1786.36
Iteration 2200: loss=1783.72
Iteration 2210: loss=1781.11
Iteration 2220: loss=1778.52
Iteration 2230: loss=1775.94
Iteration 2240: loss=1773.37
Iteration 2250: loss=1770.84
Iteration 2260: loss=1768.31
Iteration 2270: loss=1765.82
Iteration 2280: loss=1763.35
Iteration 2290: loss=1760.88
Iteration 2300: loss=1758.45
Iteration 2310: loss=1756.03
Iteration 2320: loss=1753.62
Iteration 2330: loss=1751.23
Iteration 2340: loss=1748.86
Iteration 2350: loss=1746.50
Iteration 2360: loss=1744.15
Iteration 2370: loss=1741.82
Iteration 2380: loss=1739.52
Iteration 2390: loss=1737.22
Iteration 2400: loss=1734.94
Iteration 2410: loss=1732.68
Iteration 2420: loss=1730.43
Iteration 2430: loss=1728.20
Iteration 2440: loss=1725.99
Iteration 2450: loss=1723.79
Iteration 2460: loss=1721.60
Iteration 2470: loss=1719.44
Iteration 2480: loss=1717.29
Iteration 2490: loss=1715.16
Iteration 2500: loss=1713.03
Iteration 2510: loss=1710.92
Iteration 2520: loss=1708.84
Iteration 2530: loss=1706.76
Iteration 2540: loss=1704.69
Iteration 2550: loss=1702.65
Iteration 2560: loss=1700.63
Iteration 2570: loss=1698.61
Iteration 2580: loss=1696.63
Iteration 2590: loss=1694.63
Iteration 2600: loss=1692.66
Iteration 2610: loss=1690.70
Iteration 2620: loss=1688.75
Iteration 2630: loss=1686.81
Iteration 2640: loss=1684.89
Iteration 2650: loss=1682.97
Iteration 2660: loss=1681.07
Iteration 2670: loss=1679.17
Iteration 2680: loss=1677.29
Iteration 2690: loss=1675.43
Iteration 2700: loss=1673.58
Iteration 2710: loss=1671.75
Iteration 2720: loss=1669.93
Iteration 2730: loss=1668.12
Iteration 2740: loss=1666.32
Iteration 2750: loss=1664.54
Iteration 2760: loss=1662.76
Iteration 2770: loss=1661.00
Iteration 2780: loss=1659.26
Iteration 2790: loss=1657.53
Iteration 2800: loss=1655.82
Iteration 2810: loss=1654.12
Iteration 2820: loss=1652.42
Iteration 2830: loss=1650.75
Iteration 2840: loss=1649.08

Iteration 2840: loss=1649.08
Iteration 2850: loss=1647.42
Iteration 2860: loss=1645.78
Iteration 2870: loss=1644.15
Iteration 2880: loss=1642.53
Iteration 2890: loss=1640.92
Iteration 2900: loss=1639.32
Iteration 2910: loss=1637.72
Iteration 2920: loss=1636.15
Iteration 2930: loss=1634.57
Iteration 2940: loss=1633.01
Iteration 2950: loss=1631.46
Iteration 2960: loss=1629.91
Iteration 2970: loss=1628.39
Iteration 2980: loss=1626.86
Iteration 2990: loss=1625.36
Iteration 3000: loss=1623.86
Iteration 3010: loss=1622.37
Iteration 3020: loss=1620.90
Iteration 3030: loss=1619.42
Iteration 3040: loss=1617.96
Iteration 3050: loss=1616.51
Iteration 3060: loss=1615.07
Iteration 3070: loss=1613.64
Iteration 3080: loss=1612.22
Iteration 3090: loss=1610.81
Iteration 3100: loss=1609.40
Iteration 3110: loss=1608.02
Iteration 3120: loss=1606.63
Iteration 3130: loss=1605.26
Iteration 3140: loss=1603.88
Iteration 3150: loss=1602.52
Iteration 3160: loss=1601.17
Iteration 3170: loss=1599.82
Iteration 3180: loss=1598.48
Iteration 3190: loss=1597.15
Iteration 3200: loss=1595.83
Iteration 3210: loss=1594.51
Iteration 3220: loss=1593.21
Iteration 3230: loss=1591.90
Iteration 3240: loss=1590.62
Iteration 3250: loss=1589.33
Iteration 3260: loss=1588.06
Iteration 3270: loss=1586.80
Iteration 3280: loss=1585.54
Iteration 3290: loss=1584.29
Iteration 3300: loss=1583.05
Iteration 3310: loss=1581.82
Iteration 3320: loss=1580.59
Iteration 3330: loss=1579.38
Iteration 3340: loss=1578.16
Iteration 3350: loss=1576.96
Iteration 3360: loss=1575.76
Iteration 3370: loss=1574.57
Iteration 3380: loss=1573.39
Iteration 3390: loss=1572.20
Iteration 3400: loss=1571.04
Iteration 3410: loss=1569.87
Iteration 3420: loss=1568.72
Iteration 3430: loss=1567.57
Iteration 3440: loss=1566.43
Iteration 3450: loss=1565.30
Iteration 3460: loss=1564.17
Iteration 3470: loss=1563.04
Iteration 3480: loss=1561.93
Iteration 3490: loss=1560.82
Iteration 3500: loss=1559.71
Iteration 3510: loss=1558.62
Iteration 3520: loss=1557.52
Iteration 3530: loss=1556.44
Iteration 3540: loss=1555.36
Iteration 3550: loss=1554.30
Iteration 3560: loss=1553.24

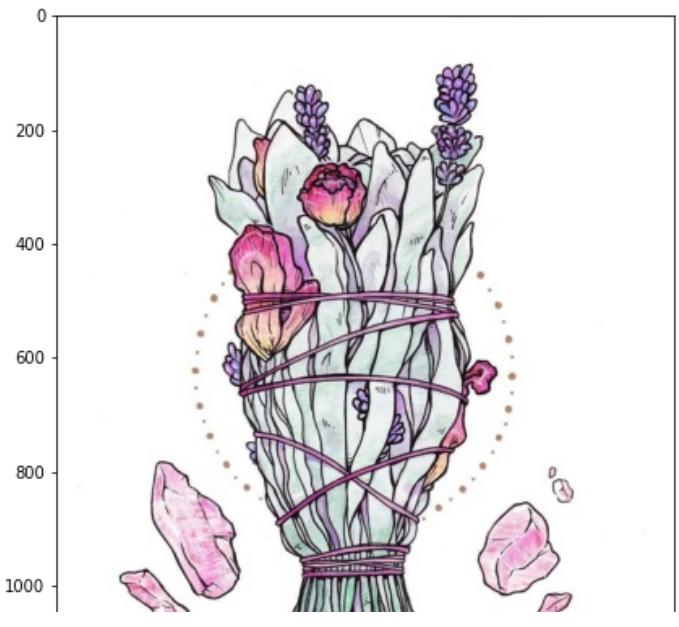
```
Iteration 3560: loss=1553.24
Iteration 3570: loss=1552.18
Iteration 3580: loss=1551.13
Iteration 3590: loss=1550.09
Iteration 3600: loss=1549.05
Iteration 3610: loss=1548.02
Iteration 3620: loss=1547.00
Iteration 3630: loss=1545.98
Iteration 3640: loss=1544.97
Iteration 3650: loss=1543.96
Iteration 3660: loss=1542.96
Iteration 3670: loss=1541.97
Iteration 3680: loss=1540.98
Iteration 3690: loss=1540.00
Iteration 3700: loss=1539.03
Iteration 3710: loss=1538.05
Iteration 3720: loss=1537.09
Iteration 3730: loss=1536.13
Iteration 3740: loss=1535.18
Iteration 3750: loss=1534.23
Iteration 3760: loss=1533.28
Iteration 3770: loss=1532.35
Iteration 3780: loss=1531.41
Iteration 3790: loss=1530.48
Iteration 3800: loss=1529.56
Iteration 3810: loss=1528.64
Iteration 3820: loss=1527.74
Iteration 3830: loss=1526.82
Iteration 3840: loss=1525.92
Iteration 3850: loss=1525.03
Iteration 3860: loss=1524.13
Iteration 3870: loss=1523.25
Iteration 3880: loss=1522.36
Iteration 3890: loss=1521.48
Iteration 3900: loss=1520.61
Iteration 3910: loss=1519.74
Iteration 3920: loss=1518.88
Iteration 3930: loss=1518.02
Iteration 3940: loss=1517.17
Iteration 3950: loss=1516.32
Iteration 3960: loss=1515.49
Iteration 3970: loss=1514.65
Iteration 3980: loss=1513.81
Iteration 3990: loss=1512.99
Iteration 4000: loss=1512.17
```

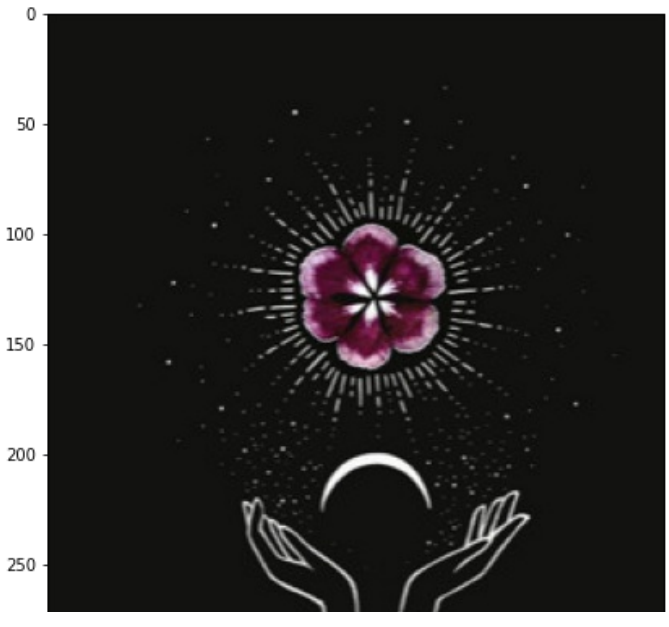


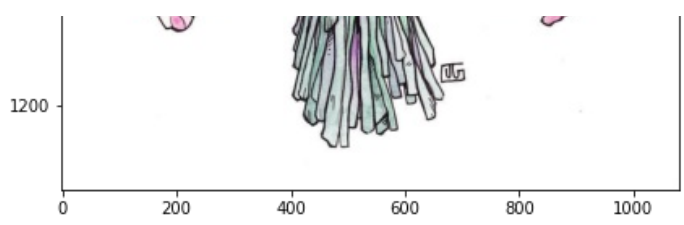
Rose Space Dark Style Image

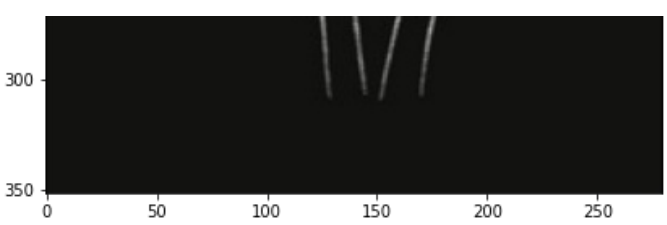
In [12]:

```
style_image_path5 = get_file(fname="/content/drive/MyDrive/AI Picture Style Transfer Examples/Style/SpaceStyleDark.jpg",
                             origin="https://drive.google.com/file/d/1h0MQBr6VO7QyBa-pQbF34Hm3E0Gfw6j2/view?usp=sharing")
content_image_path5 = get_file(fname="/content/drive/MyDrive/AI Picture Style Transfer Examples/Content/Rose.jpg",
                               origin="https://drive.google.com/file/d/1xkB0pKCRUqssUKB3uEG8MI76-EnW744p/view?usp=sharing")

# read the image file in a numpy array
a = plt.imread(content_image_path5)
b = plt.imread(style_image_path5)
f, axarr = plt.subplots(1,2, figsize=(15,15))
axarr[0].imshow(a)
axarr[1].imshow(b)
plt.show()

#Run Program
width, height = keras.preprocessing.image.load_img(content_image_path5).size
img_nrows = 400
img_ncols = int(width * img_nrows / height)

optimizer = SGD(
    keras.optimizers.schedules.ExponentialDecay(
        initial_learning_rate=100.0, decay_steps=100, decay_rate=0.96
    )
)

content_image = preprocess_image(content_image_path5)
style_reference_image = preprocess_image(style_image_path5)
combination_image = tf.Variable(preprocess_image(content_image_path5))

iterations = 4000

for i in range(1, iterations + 1):
    loss, grads = compute_loss_and_grads(
        combination_image, content_image, style_reference_image
    )
    optimizer.apply_gradients([(grads, combination_image)])
    if i % 10 == 0:
        print("Iteration %d: loss=%.2f" % (i, loss))
        img = deprocess_image(combination_image.numpy())
        fname = result_prefix + "_at_iteration_%d.png" % i
        keras.preprocessing.image.save_img(fname, img)
        result_saver(i)

display(Image(result_prefix + "_at_iteration_4000.png"))
```










```
Iteration 10: loss=24202.04
Iteration 20: loss=19522.03
Iteration 30: loss=17015.83
Iteration 40: loss=15303.97
Iteration 50: loss=14062.08
Iteration 60: loss=13064.05
Iteration 70: loss=12300.12
Iteration 80: loss=11690.30
Iteration 90: loss=11179.90
Iteration 100: loss=10751.63
Iteration 110: loss=10380.46
Iteration 120: loss=10075.23
Iteration 130: loss=9792.69
Iteration 140: loss=9537.26
Iteration 150: loss=9306.17
Iteration 160: loss=9106.95
Iteration 170: loss=8914.60
Iteration 180: loss=8735.80
Iteration 190: loss=8577.01
Iteration 200: loss=8436.92
Iteration 210: loss=8300.41
Iteration 220: loss=8164.44
Iteration 230: loss=8054.85
Iteration 240: loss=7936.95
Iteration 250: loss=7834.24
Iteration 260: loss=7731.37
Iteration 270: loss=7640.50
Iteration 280: loss=7546.98
Iteration 290: loss=7460.29
Iteration 300: loss=7382.40
Iteration 310: loss=7307.68
Iteration 320: loss=7230.73
Iteration 330: loss=7159.86
Iteration 340: loss=7097.00
Iteration 350: loss=7033.17
Iteration 360: loss=6969.27
Iteration 370: loss=6908.68
Iteration 380: loss=6855.38
Iteration 390: loss=6799.54
Iteration 400: loss=6746.51
Iteration 410: loss=6690.58
Iteration 420: loss=6647.28
Iteration 430: loss=6601.81
Iteration 440: loss=6555.78
Iteration 450: loss=6505.93
Iteration 460: loss=6467.21
Iteration 470: loss=6427.68
Iteration 480: loss=6380.91
Iteration 490: loss=6342.53
Iteration 500: loss=6305.68
Iteration 510: loss=6267.85
Iteration 520: loss=6229.62
Iteration 530: loss=6196.53
Iteration 540: loss=6165.67
Iteration 550: loss=6130.81
Iteration 560: loss=6101.28
Iteration 570: loss=6066.06
Iteration 580: loss=6036.99
Iteration 590: loss=6009.44
Iteration 600: loss=5977.04
Iteration 610: loss=5947.67
Iteration 620: loss=5917.97
Iteration 630: loss=5891.29
Iteration 640: loss=5864.34
```

```
Iteration 650: loss=5838.07
Iteration 660: loss=5813.15
Iteration 670: loss=5787.23
Iteration 680: loss=5762.53
Iteration 690: loss=5736.21
Iteration 700: loss=5713.30
Iteration 710: loss=5686.50
Iteration 720: loss=5664.86
Iteration 730: loss=5640.48
Iteration 740: loss=5617.30
Iteration 750: loss=5597.37
Iteration 760: loss=5574.56
Iteration 770: loss=5555.45
Iteration 780: loss=5534.25
Iteration 790: loss=5512.59
Iteration 800: loss=5493.68
Iteration 810: loss=5473.58
Iteration 820: loss=5455.13
Iteration 830: loss=5438.23
Iteration 840: loss=5417.02
Iteration 850: loss=5399.96
Iteration 860: loss=5380.09
Iteration 870: loss=5363.33
Iteration 880: loss=5344.01
Iteration 890: loss=5327.49
Iteration 900: loss=5310.41
Iteration 910: loss=5293.38
Iteration 920: loss=5277.11
Iteration 930: loss=5260.43
Iteration 940: loss=5244.03
Iteration 950: loss=5228.69
Iteration 960: loss=5211.89
Iteration 970: loss=5196.41
Iteration 980: loss=5181.48
Iteration 990: loss=5167.20
Iteration 1000: loss=5151.52
Iteration 1010: loss=5137.84
Iteration 1020: loss=5123.37
Iteration 1030: loss=5108.16
Iteration 1040: loss=5094.16
Iteration 1050: loss=5079.58
Iteration 1060: loss=5065.93
Iteration 1070: loss=5053.33
Iteration 1080: loss=5041.71
Iteration 1090: loss=5027.34
Iteration 1100: loss=5013.66
Iteration 1110: loss=4999.94
Iteration 1120: loss=4986.54
Iteration 1130: loss=4974.11
Iteration 1140: loss=4961.49
Iteration 1150: loss=4950.30
Iteration 1160: loss=4937.79
Iteration 1170: loss=4926.05
Iteration 1180: loss=4914.14
Iteration 1190: loss=4901.33
Iteration 1200: loss=4889.53
Iteration 1210: loss=4877.91
Iteration 1220: loss=4866.84
Iteration 1230: loss=4855.03
Iteration 1240: loss=4844.24
Iteration 1250: loss=4833.09
Iteration 1260: loss=4821.57
Iteration 1270: loss=4810.56
Iteration 1280: loss=4800.01
Iteration 1290: loss=4788.79
Iteration 1300: loss=4778.26
Iteration 1310: loss=4768.09
Iteration 1320: loss=4757.56
Iteration 1330: loss=4747.75
Iteration 1340: loss=4737.36
Iteration 1350: loss=4727.26
Iteration 1360: loss=4716.82
```

Iteration 1370: loss=4707.07
Iteration 1380: loss=4697.22
Iteration 1390: loss=4687.28
Iteration 1400: loss=4677.59
Iteration 1410: loss=4668.20
Iteration 1420: loss=4658.31
Iteration 1430: loss=4648.71
Iteration 1440: loss=4639.25
Iteration 1450: loss=4630.13
Iteration 1460: loss=4621.72
Iteration 1470: loss=4612.19
Iteration 1480: loss=4603.18
Iteration 1490: loss=4594.41
Iteration 1500: loss=4585.76
Iteration 1510: loss=4576.73
Iteration 1520: loss=4568.42
Iteration 1530: loss=4560.30
Iteration 1540: loss=4551.75
Iteration 1550: loss=4543.32
Iteration 1560: loss=4535.49
Iteration 1570: loss=4527.41
Iteration 1580: loss=4519.65
Iteration 1590: loss=4511.57
Iteration 1600: loss=4503.12
Iteration 1610: loss=4496.00
Iteration 1620: loss=4488.19
Iteration 1630: loss=4480.31
Iteration 1640: loss=4472.90
Iteration 1650: loss=4465.25
Iteration 1660: loss=4457.77
Iteration 1670: loss=4450.21
Iteration 1680: loss=4443.45
Iteration 1690: loss=4435.93
Iteration 1700: loss=4428.88
Iteration 1710: loss=4421.52
Iteration 1720: loss=4414.22
Iteration 1730: loss=4407.39
Iteration 1740: loss=4400.29
Iteration 1750: loss=4393.18
Iteration 1760: loss=4386.53
Iteration 1770: loss=4379.53
Iteration 1780: loss=4372.87
Iteration 1790: loss=4366.04
Iteration 1800: loss=4359.35
Iteration 1810: loss=4352.95
Iteration 1820: loss=4346.46
Iteration 1830: loss=4340.06
Iteration 1840: loss=4333.66
Iteration 1850: loss=4327.34
Iteration 1860: loss=4320.98
Iteration 1870: loss=4314.98
Iteration 1880: loss=4308.60
Iteration 1890: loss=4302.37
Iteration 1900: loss=4296.27
Iteration 1910: loss=4290.11
Iteration 1920: loss=4284.12
Iteration 1930: loss=4278.14
Iteration 1940: loss=4272.01
Iteration 1950: loss=4266.04
Iteration 1960: loss=4260.20
Iteration 1970: loss=4254.31
Iteration 1980: loss=4248.48
Iteration 1990: loss=4242.90
Iteration 2000: loss=4237.41
Iteration 2010: loss=4231.41
Iteration 2020: loss=4225.97
Iteration 2030: loss=4220.16
Iteration 2040: loss=4214.67
Iteration 2050: loss=4208.92
Iteration 2060: loss=4203.57
Iteration 2070: loss=4198.02
Iteration 2080: loss=4192.41

Iteration 2090: loss=4187.06
Iteration 2100: loss=4181.66
Iteration 2110: loss=4176.40
Iteration 2120: loss=4171.23
Iteration 2130: loss=4165.99
Iteration 2140: loss=4160.78
Iteration 2150: loss=4155.56
Iteration 2160: loss=4150.71
Iteration 2170: loss=4145.73
Iteration 2180: loss=4140.58
Iteration 2190: loss=4135.68
Iteration 2200: loss=4130.79
Iteration 2210: loss=4125.78
Iteration 2220: loss=4120.86
Iteration 2230: loss=4116.05
Iteration 2240: loss=4111.32
Iteration 2250: loss=4106.71
Iteration 2260: loss=4101.88
Iteration 2270: loss=4097.16
Iteration 2280: loss=4092.69
Iteration 2290: loss=4087.89
Iteration 2300: loss=4083.41
Iteration 2310: loss=4079.01
Iteration 2320: loss=4074.44
Iteration 2330: loss=4069.88
Iteration 2340: loss=4065.51
Iteration 2350: loss=4061.07
Iteration 2360: loss=4056.76
Iteration 2370: loss=4052.34
Iteration 2380: loss=4048.02
Iteration 2390: loss=4043.88
Iteration 2400: loss=4039.47
Iteration 2410: loss=4035.49
Iteration 2420: loss=4031.13
Iteration 2430: loss=4027.04
Iteration 2440: loss=4022.90
Iteration 2450: loss=4018.89
Iteration 2460: loss=4014.88
Iteration 2470: loss=4010.69
Iteration 2480: loss=4006.80
Iteration 2490: loss=4002.67
Iteration 2500: loss=3998.76
Iteration 2510: loss=3994.86
Iteration 2520: loss=3990.96
Iteration 2530: loss=3986.92
Iteration 2540: loss=3983.18
Iteration 2550: loss=3979.51
Iteration 2560: loss=3975.67
Iteration 2570: loss=3971.83
Iteration 2580: loss=3967.92
Iteration 2590: loss=3964.14
Iteration 2600: loss=3960.42
Iteration 2610: loss=3956.70
Iteration 2620: loss=3952.97
Iteration 2630: loss=3949.15
Iteration 2640: loss=3945.36
Iteration 2650: loss=3941.62
Iteration 2660: loss=3938.06
Iteration 2670: loss=3934.50
Iteration 2680: loss=3930.92
Iteration 2690: loss=3927.39
Iteration 2700: loss=3923.58
Iteration 2710: loss=3920.08
Iteration 2720: loss=3916.45
Iteration 2730: loss=3912.89
Iteration 2740: loss=3909.40
Iteration 2750: loss=3906.01
Iteration 2760: loss=3902.55
Iteration 2770: loss=3899.26
Iteration 2780: loss=3895.98
Iteration 2790: loss=3892.65
Iteration 2800: loss=3889.39

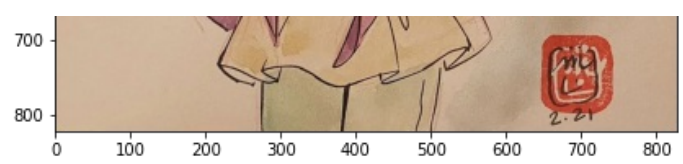
```
Iteration 2810: loss=3886.33
Iteration 2820: loss=3883.10
Iteration 2830: loss=3879.92
Iteration 2840: loss=3876.79
Iteration 2850: loss=3873.68
Iteration 2860: loss=3870.59
Iteration 2870: loss=3867.61
Iteration 2880: loss=3864.51
Iteration 2890: loss=3861.38
Iteration 2900: loss=3858.25
Iteration 2910: loss=3855.41
Iteration 2920: loss=3852.53
Iteration 2930: loss=3849.51
Iteration 2940: loss=3846.54
Iteration 2950: loss=3843.71
Iteration 2960: loss=3840.72
Iteration 2970: loss=3837.90
Iteration 2980: loss=3835.06
Iteration 2990: loss=3832.22
Iteration 3000: loss=3829.42
Iteration 3010: loss=3826.59
Iteration 3020: loss=3823.72
Iteration 3030: loss=3821.02
Iteration 3040: loss=3818.24
Iteration 3050: loss=3815.43
Iteration 3060: loss=3812.79
Iteration 3070: loss=3810.03
Iteration 3080: loss=3807.31
Iteration 3090: loss=3804.63
Iteration 3100: loss=3801.93
Iteration 3110: loss=3799.14
Iteration 3120: loss=3796.60
Iteration 3130: loss=3793.88
Iteration 3140: loss=3791.31
Iteration 3150: loss=3788.70
Iteration 3160: loss=3786.02
Iteration 3170: loss=3783.46
Iteration 3180: loss=3780.89
Iteration 3190: loss=3778.36
Iteration 3200: loss=3775.70
Iteration 3210: loss=3773.15
Iteration 3220: loss=3770.64
Iteration 3230: loss=3768.06
Iteration 3240: loss=3765.54
Iteration 3250: loss=3763.07
Iteration 3260: loss=3760.63
Iteration 3270: loss=3758.16
Iteration 3280: loss=3755.75
Iteration 3290: loss=3753.31
Iteration 3300: loss=3751.02
Iteration 3310: loss=3748.53
Iteration 3320: loss=3746.16
Iteration 3330: loss=3743.76
Iteration 3340: loss=3741.39
Iteration 3350: loss=3739.12
Iteration 3360: loss=3736.68
Iteration 3370: loss=3734.35
Iteration 3380: loss=3732.03
Iteration 3390: loss=3729.74
Iteration 3400: loss=3727.40
Iteration 3410: loss=3725.13
Iteration 3420: loss=3722.93
Iteration 3430: loss=3720.64
Iteration 3440: loss=3718.36
Iteration 3450: loss=3716.08
Iteration 3460: loss=3713.86
Iteration 3470: loss=3711.59
Iteration 3480: loss=3709.33
Iteration 3490: loss=3707.11
Iteration 3500: loss=3704.85
Iteration 3510: loss=3702.67
Iteration 3520: loss=3700.48
```

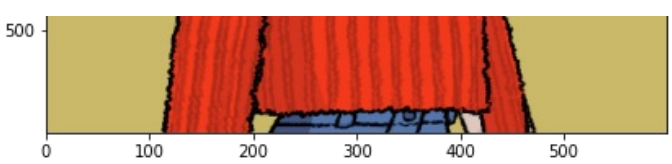
```
Iteration 3530: loss=3698.27
Iteration 3540: loss=3696.07
Iteration 3550: loss=3693.89
Iteration 3560: loss=3691.78
Iteration 3570: loss=3689.62
Iteration 3580: loss=3687.44
Iteration 3590: loss=3685.31
Iteration 3600: loss=3683.11
Iteration 3610: loss=3681.09
Iteration 3620: loss=3678.89
Iteration 3630: loss=3676.81
Iteration 3640: loss=3674.69
Iteration 3650: loss=3672.62
Iteration 3660: loss=3670.54
Iteration 3670: loss=3668.43
Iteration 3680: loss=3666.34
Iteration 3690: loss=3664.39
Iteration 3700: loss=3662.31
Iteration 3710: loss=3660.28
Iteration 3720: loss=3658.38
Iteration 3730: loss=3656.29
Iteration 3740: loss=3654.35
Iteration 3750: loss=3652.35
Iteration 3760: loss=3650.45
Iteration 3770: loss=3648.44
Iteration 3780: loss=3646.42
Iteration 3790: loss=3644.51
Iteration 3800: loss=3642.64
Iteration 3810: loss=3640.66
Iteration 3820: loss=3638.83
Iteration 3830: loss=3636.87
Iteration 3840: loss=3635.04
Iteration 3850: loss=3633.16
Iteration 3860: loss=3631.34
Iteration 3870: loss=3629.48
Iteration 3880: loss=3627.62
Iteration 3890: loss=3625.83
Iteration 3900: loss=3624.01
Iteration 3910: loss=3622.12
Iteration 3920: loss=3620.35
Iteration 3930: loss=3618.57
Iteration 3940: loss=3616.77
Iteration 3950: loss=3615.01
Iteration 3960: loss=3613.24
Iteration 3970: loss=3611.45
Iteration 3980: loss=3609.73
Iteration 3990: loss=3608.06
Iteration 4000: loss=3606.35
```

Girl Drawing Style Image

In [14]:

```
style_image_path7 = get_file(fname="/content/drive/MyDrive/AI Picture Style Transfer Examples/Style/GirlDrawingStyle.jpg",
                             origin="https://drive.google.com/file/d/1h0MQBr6VO7QyBa-pQbF34Hm3E0Gfw6j2/view?usp=sharing")
content_image_path7 = get_file(fname="/content/drive/MyDrive/AI Picture Style Transfer Examples/Content/GirlDrawing.jpg",
                               origin="https://drive.google.com/file/d/1xkB0pKCRUqssUKB3uEG8MI76-EnW744p/view?usp=sharing")

# read the image file in a numpy array
a = plt.imread(content_image_path7)
b = plt.imread(style_image_path7)
f, axarr = plt.subplots(1,2, figsize=(15,15))
axarr[0].imshow(a)
axarr[1].imshow(b)
plt.show()

#Run Program
width, height = keras.preprocessing.image.load_img(content_image_path7).size
img_nrows = 400
img_ncols = int(width * img_nrows / height)

optimizer = SGD(
    keras.optimizers.schedules.ExponentialDecay(
        initial_learning_rate=100.0, decay_steps=100, decay_rate=0.96
    )
)

content_image = preprocess_image(content_image_path7)
style_reference_image = preprocess_image(style_image_path7)
combination_image = tf.Variable(preprocess_image(content_image_path7))

iterations = 4000

for i in range(1, iterations + 1):
    loss, grads = compute_loss_and_grads(
        combination_image, content_image, style_reference_image
    )
    optimizer.apply_gradients([(grads, combination_image)])
    if i % 10 == 0:
        print("Iteration %d: loss=%.2f" % (i, loss))
        img = deprocess_image(combination_image.numpy())
        fname = result_prefix + "_at_iteration_%d.png" % i
        keras.preprocessing.image.save_img(fname, img)
        result_saver(i)

display(Image(result_prefix + "_at_iteration_4000.png"))
```






```
Iteration 10: loss=13205.87
Iteration 20: loss=7689.65
Iteration 30: loss=5828.55
Iteration 40: loss=4821.25
Iteration 50: loss=4176.09
Iteration 60: loss=3723.47
Iteration 70: loss=3381.08
Iteration 80: loss=3109.32
Iteration 90: loss=2887.75
Iteration 100: loss=2704.49
Iteration 110: loss=2548.59
Iteration 120: loss=2415.35
Iteration 130: loss=2299.05
Iteration 140: loss=2196.41
Iteration 150: loss=2106.18
Iteration 160: loss=2025.72
Iteration 170: loss=1953.79
Iteration 180: loss=1888.96
Iteration 190: loss=1829.91
Iteration 200: loss=1776.09
Iteration 210: loss=1726.69
Iteration 220: loss=1681.22
Iteration 230: loss=1639.20
Iteration 240: loss=1600.29
Iteration 250: loss=1564.18
Iteration 260: loss=1530.52
Iteration 270: loss=1498.90
Iteration 280: loss=1469.13
Iteration 290: loss=1441.21
Iteration 300: loss=1414.90
Iteration 310: loss=1390.03
Iteration 320: loss=1366.49
Iteration 330: loss=1344.14
Iteration 340: loss=1322.94
Iteration 350: loss=1302.81
Iteration 360: loss=1283.78
Iteration 370: loss=1265.59
Iteration 380: loss=1248.30
Iteration 390: loss=1231.84
Iteration 400: loss=1216.09
Iteration 410: loss=1201.00
Iteration 420: loss=1186.50
Iteration 430: loss=1172.64
Iteration 440: loss=1159.35
Iteration 450: loss=1146.60
Iteration 460: loss=1134.34
Iteration 470: loss=1122.54
Iteration 480: loss=1111.19
Iteration 490: loss=1100.20
Iteration 500: loss=1089.57
Iteration 510: loss=1079.29
Iteration 520: loss=1069.34
Iteration 530: loss=1059.73
Iteration 540: loss=1050.42
Iteration 550: loss=1041.39
Iteration 560: loss=1032.63
Iteration 570: loss=1024.14
Iteration 580: loss=1015.87
Iteration 590: loss=1007.86
Iteration 600: loss=1000.07
Iteration 610: loss=992.52
Iteration 620: loss=985.17
Iteration 630: loss=978.00
Iteration 640: loss=971.01
Iteration 650: loss=964.19
Iteration 660: loss=957.52
```

```
Iteration 670: loss=951.02
Iteration 680: loss=944.68
Iteration 690: loss=938.47
Iteration 700: loss=932.42
Iteration 710: loss=926.52
Iteration 720: loss=920.74
Iteration 730: loss=915.10
Iteration 740: loss=909.57
Iteration 750: loss=904.17
Iteration 760: loss=898.90
Iteration 770: loss=893.75
Iteration 780: loss=888.71
Iteration 790: loss=883.79
Iteration 800: loss=878.95
Iteration 810: loss=874.23
Iteration 820: loss=869.59
Iteration 830: loss=865.05
Iteration 840: loss=860.59
Iteration 850: loss=856.23
Iteration 860: loss=851.95
Iteration 870: loss=847.76
Iteration 880: loss=843.65
Iteration 890: loss=839.62
Iteration 900: loss=835.67
Iteration 910: loss=831.78
Iteration 920: loss=827.98
Iteration 930: loss=824.24
Iteration 940: loss=820.58
Iteration 950: loss=816.97
Iteration 960: loss=813.43
Iteration 970: loss=809.96
Iteration 980: loss=806.55
Iteration 990: loss=803.19
Iteration 1000: loss=799.90
Iteration 1010: loss=796.65
Iteration 1020: loss=793.45
Iteration 1030: loss=790.32
Iteration 1040: loss=787.23
Iteration 1050: loss=784.21
Iteration 1060: loss=781.24
Iteration 1070: loss=778.31
Iteration 1080: loss=775.43
Iteration 1090: loss=772.58
Iteration 1100: loss=769.79
Iteration 1110: loss=767.04
Iteration 1120: loss=764.33
Iteration 1130: loss=761.66
Iteration 1140: loss=759.02
Iteration 1150: loss=756.43
Iteration 1160: loss=753.87
Iteration 1170: loss=751.34
Iteration 1180: loss=748.85
Iteration 1190: loss=746.40
Iteration 1200: loss=743.99
Iteration 1210: loss=741.61
Iteration 1220: loss=739.26
Iteration 1230: loss=736.93
Iteration 1240: loss=734.63
Iteration 1250: loss=732.35
Iteration 1260: loss=730.11
Iteration 1270: loss=727.90
Iteration 1280: loss=725.71
Iteration 1290: loss=723.55
Iteration 1300: loss=721.42
Iteration 1310: loss=719.31
Iteration 1320: loss=717.23
Iteration 1330: loss=715.18
Iteration 1340: loss=713.15
Iteration 1350: loss=711.15
Iteration 1360: loss=709.18
Iteration 1370: loss=707.24
Iteration 1380: loss=705.32
```

```
Iteration 1390: loss=703.42
Iteration 1400: loss=701.55
Iteration 1410: loss=699.71
Iteration 1420: loss=697.88
Iteration 1430: loss=696.08
Iteration 1440: loss=694.30
Iteration 1450: loss=692.54
Iteration 1460: loss=690.81
Iteration 1470: loss=689.09
Iteration 1480: loss=687.40
Iteration 1490: loss=685.72
Iteration 1500: loss=684.06
Iteration 1510: loss=682.42
Iteration 1520: loss=680.80
Iteration 1530: loss=679.19
Iteration 1540: loss=677.60
Iteration 1550: loss=676.02
Iteration 1560: loss=674.47
Iteration 1570: loss=672.93
Iteration 1580: loss=671.41
Iteration 1590: loss=669.90
Iteration 1600: loss=668.41
Iteration 1610: loss=666.93
Iteration 1620: loss=665.47
Iteration 1630: loss=664.03
Iteration 1640: loss=662.60
Iteration 1650: loss=661.19
Iteration 1660: loss=659.79
Iteration 1670: loss=658.40
Iteration 1680: loss=657.03
Iteration 1690: loss=655.68
Iteration 1700: loss=654.34
Iteration 1710: loss=653.01
Iteration 1720: loss=651.70
Iteration 1730: loss=650.40
Iteration 1740: loss=649.11
Iteration 1750: loss=647.84
Iteration 1760: loss=646.57
Iteration 1770: loss=645.33
Iteration 1780: loss=644.09
Iteration 1790: loss=642.86
Iteration 1800: loss=641.65
Iteration 1810: loss=640.45
Iteration 1820: loss=639.25
Iteration 1830: loss=638.07
Iteration 1840: loss=636.90
Iteration 1850: loss=635.74
Iteration 1860: loss=634.59
Iteration 1870: loss=633.45
Iteration 1880: loss=632.32
Iteration 1890: loss=631.20
Iteration 1900: loss=630.10
Iteration 1910: loss=629.00
Iteration 1920: loss=627.92
Iteration 1930: loss=626.84
Iteration 1940: loss=625.78
Iteration 1950: loss=624.72
Iteration 1960: loss=623.68
Iteration 1970: loss=622.64
Iteration 1980: loss=621.61
Iteration 1990: loss=620.60
Iteration 2000: loss=619.59
Iteration 2010: loss=618.59
Iteration 2020: loss=617.59
Iteration 2030: loss=616.61
Iteration 2040: loss=615.63
Iteration 2050: loss=614.66
Iteration 2060: loss=613.70
Iteration 2070: loss=612.74
Iteration 2080: loss=611.79
Iteration 2090: loss=610.85
Iteration 2100: loss=609.91
```

```
Iteration 2110: loss=608.98
Iteration 2120: loss=608.06
Iteration 2130: loss=607.15
Iteration 2140: loss=606.24
Iteration 2150: loss=605.35
Iteration 2160: loss=604.46
Iteration 2170: loss=603.58
Iteration 2180: loss=602.71
Iteration 2190: loss=601.84
Iteration 2200: loss=600.99
Iteration 2210: loss=600.14
Iteration 2220: loss=599.30
Iteration 2230: loss=598.46
Iteration 2240: loss=597.63
Iteration 2250: loss=596.81
Iteration 2260: loss=596.00
Iteration 2270: loss=595.19
Iteration 2280: loss=594.39
Iteration 2290: loss=593.60
Iteration 2300: loss=592.81
Iteration 2310: loss=592.04
Iteration 2320: loss=591.26
Iteration 2330: loss=590.50
Iteration 2340: loss=589.73
Iteration 2350: loss=588.98
Iteration 2360: loss=588.23
Iteration 2370: loss=587.48
Iteration 2380: loss=586.74
Iteration 2390: loss=586.01
Iteration 2400: loss=585.28
Iteration 2410: loss=584.56
Iteration 2420: loss=583.84
Iteration 2430: loss=583.13
Iteration 2440: loss=582.43
Iteration 2450: loss=581.73
Iteration 2460: loss=581.04
Iteration 2470: loss=580.35
Iteration 2480: loss=579.66
Iteration 2490: loss=578.99
Iteration 2500: loss=578.31
Iteration 2510: loss=577.65
Iteration 2520: loss=576.98
Iteration 2530: loss=576.32
Iteration 2540: loss=575.67
Iteration 2550: loss=575.02
Iteration 2560: loss=574.38
Iteration 2570: loss=573.74
Iteration 2580: loss=573.11
Iteration 2590: loss=572.48
Iteration 2600: loss=571.86
Iteration 2610: loss=571.24
Iteration 2620: loss=570.63
Iteration 2630: loss=570.02
Iteration 2640: loss=569.42
Iteration 2650: loss=568.82
Iteration 2660: loss=568.23
Iteration 2670: loss=567.64
Iteration 2680: loss=567.05
Iteration 2690: loss=566.47
Iteration 2700: loss=565.89
Iteration 2710: loss=565.32
Iteration 2720: loss=564.75
Iteration 2730: loss=564.18
Iteration 2740: loss=563.62
Iteration 2750: loss=563.06
Iteration 2760: loss=562.50
Iteration 2770: loss=561.95
Iteration 2780: loss=561.41
Iteration 2790: loss=560.86
Iteration 2800: loss=560.32
Iteration 2810: loss=559.79
Iteration 2820: loss=559.26
```

```
Iteration 2830: loss=558.73
Iteration 2840: loss=558.20
Iteration 2850: loss=557.68
Iteration 2860: loss=557.16
Iteration 2870: loss=556.65
Iteration 2880: loss=556.14
Iteration 2890: loss=555.63
Iteration 2900: loss=555.12
Iteration 2910: loss=554.62
Iteration 2920: loss=554.13
Iteration 2930: loss=553.63
Iteration 2940: loss=553.14
Iteration 2950: loss=552.66
Iteration 2960: loss=552.17
Iteration 2970: loss=551.69
Iteration 2980: loss=551.22
Iteration 2990: loss=550.74
Iteration 3000: loss=550.27
Iteration 3010: loss=549.80
Iteration 3020: loss=549.34
Iteration 3030: loss=548.87
Iteration 3040: loss=548.41
Iteration 3050: loss=547.96
Iteration 3060: loss=547.50
Iteration 3070: loss=547.05
Iteration 3080: loss=546.61
Iteration 3090: loss=546.16
Iteration 3100: loss=545.72
Iteration 3110: loss=545.28
Iteration 3120: loss=544.85
Iteration 3130: loss=544.41
Iteration 3140: loss=543.98
Iteration 3150: loss=543.56
Iteration 3160: loss=543.13
Iteration 3170: loss=542.71
Iteration 3180: loss=542.29
Iteration 3190: loss=541.87
Iteration 3200: loss=541.46
Iteration 3210: loss=541.05
Iteration 3220: loss=540.64
Iteration 3230: loss=540.23
Iteration 3240: loss=539.83
Iteration 3250: loss=539.43
Iteration 3260: loss=539.03
Iteration 3270: loss=538.64
Iteration 3280: loss=538.25
Iteration 3290: loss=537.86
Iteration 3300: loss=537.47
Iteration 3310: loss=537.09
Iteration 3320: loss=536.70
Iteration 3330: loss=536.32
Iteration 3340: loss=535.95
Iteration 3350: loss=535.57
Iteration 3360: loss=535.20
Iteration 3370: loss=534.83
Iteration 3380: loss=534.46
Iteration 3390: loss=534.10
Iteration 3400: loss=533.73
Iteration 3410: loss=533.37
Iteration 3420: loss=533.01
Iteration 3430: loss=532.66
Iteration 3440: loss=532.30
Iteration 3450: loss=531.95
Iteration 3460: loss=531.60
Iteration 3470: loss=531.25
Iteration 3480: loss=530.90
Iteration 3490: loss=530.56
Iteration 3500: loss=530.22
Iteration 3510: loss=529.88
Iteration 3520: loss=529.54
Iteration 3530: loss=529.21
Iteration 3540: loss=528.87
```

```
Iteration 3550: loss=528.54
Iteration 3560: loss=528.21
Iteration 3570: loss=527.88
Iteration 3580: loss=527.55
Iteration 3590: loss=527.23
Iteration 3600: loss=526.91
Iteration 3610: loss=526.59
Iteration 3620: loss=526.27
Iteration 3630: loss=525.95
Iteration 3640: loss=525.63
Iteration 3650: loss=525.32
Iteration 3660: loss=525.01
Iteration 3670: loss=524.70
Iteration 3680: loss=524.39
Iteration 3690: loss=524.08
Iteration 3700: loss=523.78
Iteration 3710: loss=523.47
Iteration 3720: loss=523.17
Iteration 3730: loss=522.87
Iteration 3740: loss=522.57
Iteration 3750: loss=522.28
Iteration 3760: loss=521.98
Iteration 3770: loss=521.69
Iteration 3780: loss=521.40
Iteration 3790: loss=521.11
Iteration 3800: loss=520.82
Iteration 3810: loss=520.53
Iteration 3820: loss=520.25
Iteration 3830: loss=519.96
Iteration 3840: loss=519.68
Iteration 3850: loss=519.40
Iteration 3860: loss=519.12
Iteration 3870: loss=518.85
Iteration 3880: loss=518.57
Iteration 3890: loss=518.30
Iteration 3900: loss=518.02
Iteration 3910: loss=517.75
Iteration 3920: loss=517.49
Iteration 3930: loss=517.22
Iteration 3940: loss=516.95
Iteration 3950: loss=516.69
Iteration 3960: loss=516.43
Iteration 3970: loss=516.16
Iteration 3980: loss=515.90
Iteration 3990: loss=515.65
Iteration 4000: loss=515.39
```