Fast Style Transfer for Arbitrary Styles

Let's start with importing TF-2 and all relevant dependencies.

In [1]:
```
import functools
import os

from matplotlib import gridspec
import matplotlib.pylab as plt
import numpy as np
import tensorflow as tf
import tensorflow_hub as hub

print("TF Version: ", tf.__version__)
print("TF-Hub version: ", hub.__version__)
print("Eager mode enabled: ", tf.executing_eagerly())
print("GPU available: ", tf.test.is_gpu_available())
```

```
TF Version:  2.11.0
TF-Hub version:  0.12.0
Eager mode enabled:  True
WARNING:tensorflow:From /var/folders/yw/k75t7n390ts0pt4ymwsb9x_r0000gn/
T/ipykernel_31124/3324514280.py:13: is_gpu_available (from tensorflow.py
thon.framework.test_util) is deprecated and will be removed in a future
version.
Instructions for updating:
Use `tf.config.list_physical_devices('GPU')` instead.
GPU available:  True
Metal device set to: Apple M2
```

```
2023-03-15 16:29:55.502214: I tensorflow/core/common_runtime/pluggable_d
evice/pluggable_device_factory.cc:306] Could not identify NUMA node of p
latform GPU ID 0, defaulting to 0. Your kernel may not have been built w
ith NUMA support.
2023-03-15 16:29:55.502713: I tensorflow/core/common_runtime/pluggable_d
evice/pluggable_device_factory.cc:272] Created TensorFlow device (/devic
e:GPU:0 with 0 MB memory) -> physical PluggableDevice (device: 0, name:
METAL, pci bus id: <undefined>)
```

In [2]:
```
# @title Define image loading and visualization functions  { display-mode

def crop_center(image):
  """Returns a cropped square image."""
  shape = image.shape
  new_shape = min(shape[1], shape[2])
  offset_y = max(shape[1] - shape[2], 0) // 2
  offset_x = max(shape[2] - shape[1], 0) // 2
  image = tf.image.crop_to_bounding_box(
      image, offset_y, offset_x, new_shape, new_shape)
  return image

@functools.lru_cache(maxsize=None)
def load_image(image_url, image_size=(256, 256), preserve_aspect_ratio=Tr
  """Loads and preprocesses images."""
  # Cache image file locally.
  image_path = tf.keras.utils.get_file(os.path.basename(image_url)[-128:]
  # Load and convert to float32 numpy array, add batch dimension, and nor
```

```
  img = plt.imread(image_path).astype(np.float32)[np.newaxis, ...]
  if img.max() > 1.0:
    img = img / 255.
  if len(img.shape) == 3:
    img = tf.stack([img, img, img], axis=-1)
  img = crop_center(img)
  img = tf.image.resize(img, image_size, preserve_aspect_ratio=True)
  return img

def show_n(images, titles=('',)):
  n = len(images)
  image_sizes = [image.shape[1] for image in images]
  w = (image_sizes[0] * 6) // 320
  plt.figure(figsize=(w  * n, w))
  gs = gridspec.GridSpec(1, n, width_ratios=image_sizes)
  for i in range(n):
    plt.subplot(gs[i])
    plt.imshow(images[i][0], aspect='equal')
    plt.axis('off')
    plt.title(titles[i] if len(titles) > i else '')
  plt.show()
```

Let's get as well some images to play with.

In [3]:
```
# @title Load example images  { display-mode: "form" }

content_image_url = 'https://framemark.vam.ac.uk/collections/2006BH7789/f
style_image_url = 'https://upload.wikimedia.org/wikipedia/commons/0/0a/Th
output_image_size = 384  # @param {type:"integer"}

# The content image size can be arbitrary.
content_img_size = (output_image_size, output_image_size)
# The style prediction model was trained with image size 256 and it's the
# recommended image size for the style image (though, other sizes work as
# well but will lead to different results).
style_img_size = (256, 256)  # Recommended to keep it at 256.

content_image = load_image(content_image_url, content_img_size)
style_image = load_image(style_image_url, style_img_size)
style_image = tf.nn.avg_pool(style_image, ksize=[3,3], strides=[1,1], pad
show_n([content_image, style_image], ['Content image', 'Style image'])
```

```
Downloading data from https://framemark.vam.ac.uk/collections/2006BH778
9/full/1400,/0/default.jpg
   8192/Unknown - 0s 0us/step
```

```
2023-03-15 16:30:04.646274: I tensorflow/core/common_runtime/pluggable_d
evice/pluggable_device_factory.cc:306] Could not identify NUMA node of p
latform GPU ID 0, defaulting to 0. Your kernel may not have been built w
ith NUMA support.
2023-03-15 16:30:04.646303: I tensorflow/core/common_runtime/pluggable_d
evice/pluggable_device_factory.cc:272] Created TensorFlow device (/job:l
ocalhost/replica:0/task:0/device:GPU:0 with 0 MB memory) -> physical Plu
ggableDevice (device: 0, name: METAL, pci bus id: <undefined>)
```

Content image

Style image

Import TF-Hub module

In [4]:
```
# Load TF-Hub module.

hub_handle = 'https://tfhub.dev/google/magenta/arbitrary-image-stylizatio
hub_module = hub.load(hub_handle)
```

The signature of this hub module for image stylization is:

```
outputs = hub_module(content_image, style_image)
stylized_image = outputs[0]
```

Where `content_image`, `style_image`, and `stylized_image` are expected to be 4-D Tensors with shapes `[batch_size, image_height, image_width, 3]`.

In the current example we provide only single images and therefore the batch dimension is 1, but one can use the same module to process more images at the same time.

The input and output values of the images should be in the range [0, 1].

The shapes of content and style image don't have to match. Output image shape is the same as the content image shape.

Demonstrate image stylization

In [5]:
```
# Stylize content image with given style image.
# This is pretty fast within a few milliseconds on a GPU.

outputs = hub_module(tf.constant(content_image), tf.constant(style_image)
stylized_image = outputs[0]
```

```
2023-03-15 16:30:23.037459: W tensorflow/tsl/platform/profile_utils/cpu_
utils.cc:128] Failed to get CPU frequency: 0 Hz
2023-03-15 16:30:23.048204: I tensorflow/core/grappler/optimizers/custom
_graph_optimizer_registry.cc:114] Plugin optimizer for device_type GPU i
s enabled.
```

In [6]:
```
# Visualize input images and the generated stylized image.

show_n([content_image, style_image, stylized_image], titles=['Original co
```