LAB- 01

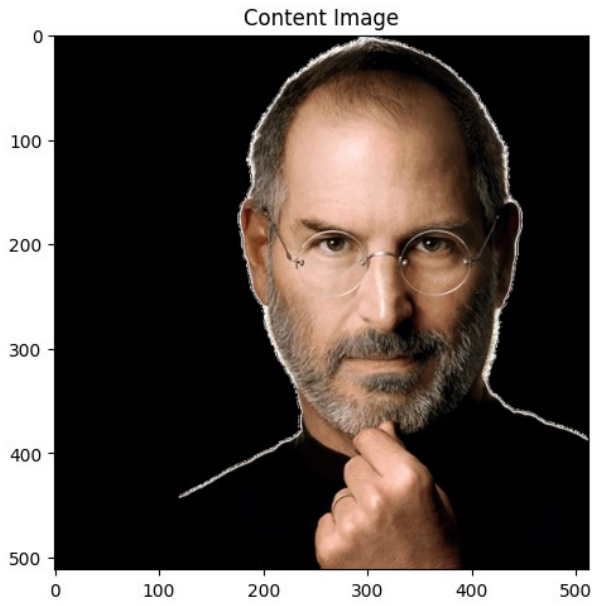
Aim: Neural Style Transfer for images

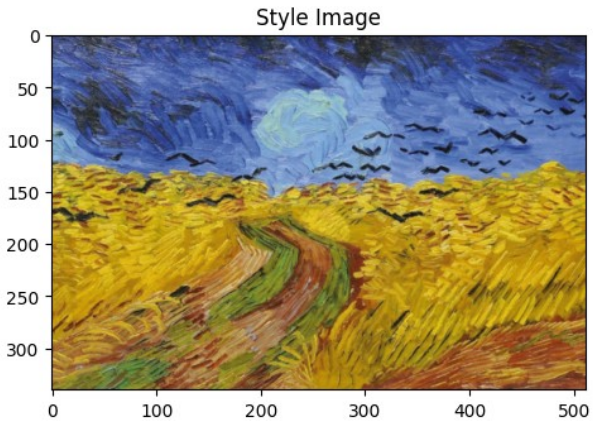
```
import os
import tensorflow as tf
os.environ['TFHUB_MODEL_LOAD_FORMAT'] = 'COMPRESSED'

import IPython.display as display

import matplotlib.pyplot as plt
import matplotlib as mpl
mpl.rcParams['figure.figsize'] = (12, 12)
mpl.rcParams['axes.grid'] = False

import numpy as np
import PIL.Image

def tensor_to_image(tensor):
  tensor = tensor*255
  tensor = np.array(tensor, dtype=np.uint8)
  if np.ndim(tensor)>3:
    assert tensor.shape[0] == 1
    tensor = tensor[0]
  return PIL.Image.fromarray(tensor)

content_path = tf.keras.utils.get_file('content.png',
'https://i.ibb.co/6mVpxGW/content.png')
style_path =
tf.keras.utils.get_file('style.jpg','https://i.ibb.co/30nz9Lc/style.jp
g')
```

```
Downloading data from https://i.ibb.co/6mVpxGW/content.png
424144/424144 [==============================] - 0s 0us/step
Downloading data from https://i.ibb.co/30nz9Lc/style.jpg
140844/140844 [==============================] - 0s 0us/step
```

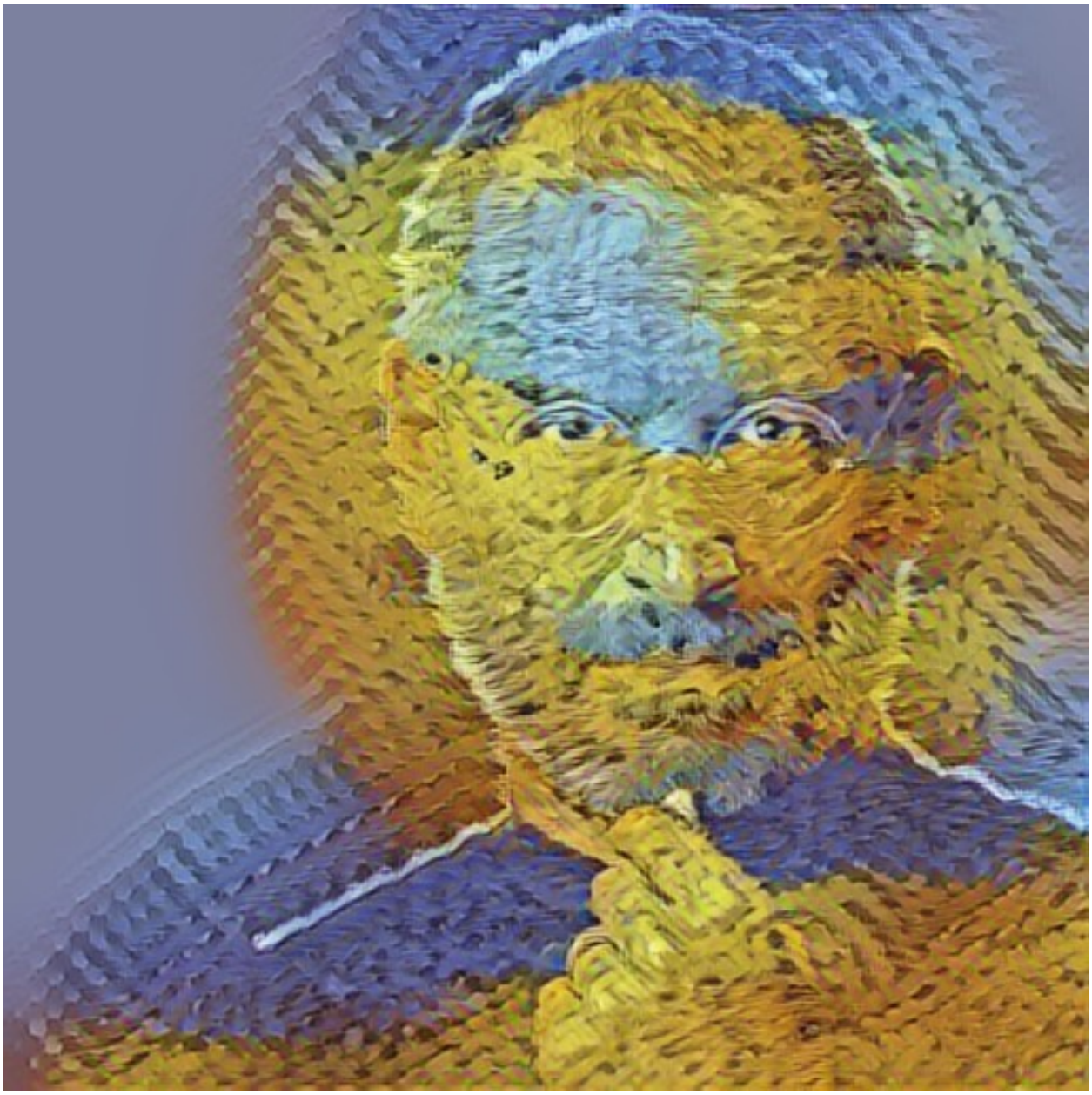
```
def load_img(path_to_img):
  max_dim = 512
  img = tf.io.read_file(path_to_img)
  img = tf.image.decode_image(img, channels=3)
  img = tf.image.convert_image_dtype(img, tf.float32)

  shape = tf.cast(tf.shape(img)[:-1], tf.float32)
  long_dim = max(shape)
  scale = max_dim / long_dim

  new_shape = tf.cast(shape * scale, tf.int32)
```

```
  img = tf.image.resize(img, new_shape)
  img = img[tf.newaxis, :]
  return img
def imshow(image, title=None):
  if len(image.shape) > 3:
    image = tf.squeeze(image, axis=0)

  plt.imshow(image)
  if title:
    plt.title(title)
content_image = load_img(content_path)
style_image = load_img(style_path)

plt.subplot(1, 2, 1)
imshow(content_image, 'Content Image')

plt.subplot(1, 2, 2)
imshow(style_image, 'Style Image')
```

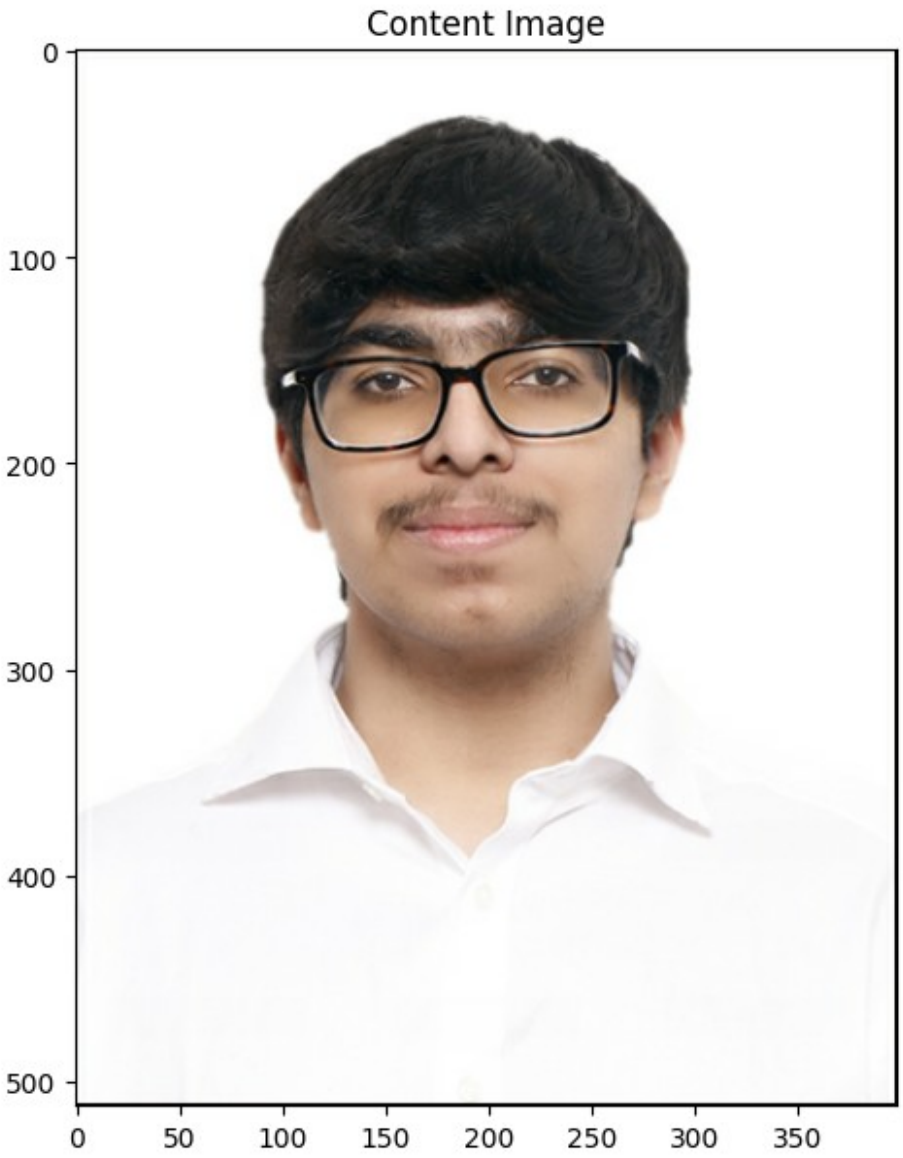





```
import tensorflow_hub as hub
hub_model = hub.load('https://tfhub.dev/google/magenta/arbitrary-
image-stylization-v1-256/2')

stylized_image = hub_model(tf.constant(content_image),
tf.constant(style_image))[0]
tensor_to_image(stylized_image)
```



```
content_image = load_img('/content/Passport Size Photo.jpg')
plt.subplot(1, 2, 1)
imshow(content_image, 'Content Image')
```




```
stylized_image = hub_model(tf.constant(content_image),
tf.constant(style_image))[0]
tensor_to_image(stylized_image)
```