Neural style transfer

NOTE for DEV.to: copy from TensorFlow DRC 2.4 TpUNC format

Description

The tutorial uses deep learning to compose one image in the style of another image (ever wish you could paint like Picasso or van Gogh?). Technically speaking, it takes three images, a content image, a style reference image (such as an artwork), and an output image, and blends the content image into the style of the style reference image, in such a way that the output image looks like the content image, but "painted" in the style of the style reference image.

This implementation follows the original style transfer algorithm. It optimizes the image content to a particular style. Between iterations over training, the output image is updated to minimize the actual image stylization loss. The approach is much slower than a TensorFlow Hub model, but it produces the best results.

For a detailed application of neural transfer check out this notebook to learn more about how to run a pre-trained tensorflow/models/research/magenta/arbitrary-image-stylization-v1-256 model in eager mode and in TensorFlow Lite.

What is Neural Style

Neural style transfer is an optimization technique used to take two images—a content image and a style reference image (such as an artwork by a famous painter)—and blend them together so the output image looks like the content image, but "painted" in the style of the style reference image.

This is accomplished by optimizing the output image to match the content statistics of the content image and the style statistics of the style reference image. These statistics are extracted from the images using a convolutional network.

For example, let's take an image of this dog and Wassily Kandinsky's Composition 7:

Then we would like to see how this dog looks if Kandinsky had painted it like this (presumably with this style? Something like this?

Setup

Apt-drive

```
$ apt-get install libhdf5-dev libc6-dev libmagickcore-dev libmagickwand-dev libblas-dev liblapack-dev gfortran
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
libc6-dev is set to manually installed.
The following additional packages will be installed: libaec0 libblas3 libcblas3 libcfitsio10 libgfortran-8-dev libgfortran5 libhdf5-103 libhdf5-103-1v5 libhdf5-dev libhdf5-mpi-dev libhdf5-mpi-1v5 libhdf5-openmpi-103 libhdf5-openmpi-103-1v5 libhdf5-serial-dev libhdf5-serial-1v5 libhdf5-103-1v5 libhdf5-mpi-1v5 libhdf5-openmpi-103-1v5 libhdf5-serial-1v5 libjpeg-dev libjpeg62-dev libjpeg62-dev:amd64 libjpeg8-dev libjpeg8-dev:amd64 libjpeg-turbo8-dev libjpeg-turbo8-dev:amd64 libjpeg-turbo8-dev:amd64 libjpeg-turbo8-dev:amd64 
```

Re-run the optimization

Choose a weight for the `total_variation_loss`:

In [34]:
```
total_variation_weight=30
```

Now include it in the `train_step` function:

In [35]:
```
@tf.function()
def train_step(image):
  with tf.GradientTape() as tape:
    outputs = extractor(image)
    loss = style_content_loss(outputs)
    loss += total_variation_weight*tf.image.total_variation(image)

  grad = tape.gradient(loss, image)
  opt.apply_gradients([(grad, image)])
  image.assign(clip_0_1(image))
```

Reinitialize the optimization variable:

In [36]:
```
image = tf.Variable(content_image)
```

And run the optimization:

In [37]:
```
import time
start = time.time()

epochs = 10
steps_per_epoch = 100

step = 0
for n in range(epochs):
  for m in range(steps_per_epoch):
    step += 1
    train_step(image)
    print(".", end='', flush=True)
  display.clear_output(wait=True)
  display.display(tensor_to_image(image))
  print("Train step: {}".format(step))

end = time.time()
print("Total time: {:.1f}".format(end-start))
```

```
Train step: 1000
Total time: 27.6
```

Finally, save the result:

In [38]:
```
file_name = 'stylized-image.png'
tensor_to_image(image).save(file_name)

try:
  from google.colab import files
except ImportError:
   pass
else:
  files.download(file_name)
```

Extensions

- Repeat with your own style and content image.
  - What would be an interesting combination of style and content, and why?
- Extend the training steps? how does this change the result?
- Change the Content and Style weights above. How does this change the results?

References

This tutorial was adapted from a TF example on colab

It demonstrates the original style-transfer algorithm. For a simple application of style transfer check out this tutorial to learn more about how to use the arbitrary image style transfer model from TensorFlow Hub.