An Introduction to Deep Learning With Python

[8.3] Neural style transfer

Prof. Yuzo Iano

pgs: 289 - 294

Neural style transfer in Keras

In [1]:
```
# Defining initial variables
from keras.preprocessing.image import load_img, img_to_array
```

In [2]:
```
target_image_path = 'portrait.PNG'
style_reference_image_path = 'style_reference.PNG'

width, height = load_img(target_image_path).size
img_height = 400
img_width = int(width * img_height / height)
```

Auxiliary functions

In [3]:
```
import numpy as np
from keras.applications import vgg19
```

In [4]:
```
def preprocess_image(image_path):
    img = load_img(image_path, target_size=(img_height, img_width))
    img = img_to_array(img)
    img = np.expand_dims(img, axis=0)
    img = vgg19.preprocess_input(img)
    return img
```

In [5]:
```
def deprocess_image(x):
    x[:, :, 0] += 103.939
    x[:, :, 1] += 116.779
    x[:, :, 2] += 123.68
    x = x[:, :, ::-1]
    x = np.clip(x, 0, 255).astype('uint8')
    return x
```

Loading the pretrained VGG19 network and applying it to the three images

In [6]:
```
from keras import backend as K
```

In [7]:
```
target_image = K.constant(preprocess_image(target_image_path))
style_reference_image = K.constant(preprocess_image(style_reference_image_path))
combination_image = K.placeholder((1, img_height, img_width, 3))

input_tensor = K.concatenate([target_image,
                              style_reference_image,
                              combination_image], axis=0)

model = vgg19.VGG19(input_tensor = input_tensor,
                    weights = 'imagenet',
                    include_top = False)

print('Model loaded.')
```

```
WARNING:tensorflow:From C:\Users\pablo\AppData\Roaming\Python\Python36\site-packages\tensorflow\python\framework\o
p_def_library.py:263: colocate_with (from tensorflow.python.framework.ops) is deprecated and will be removed in a
future version.
Instructions for updating:
Colocations handled automatically by placer.
Downloading data from https://github.com/fchollet/deep-learning-models/releases/download/v0.1/vgg19_weights_tf_dim
_ordering_tf_kernels_notop.h5
80142336/80134624 [==============================] - 10s 0us/step
Model loaded.
```

Content loss

In [8]:
```
def content_loss(base, combination):
    return K.sum(K.square(combination - base))
```

Style loss

In [10]:
```
def gram_matrix(x):
    features = K.batch_flatten(K.permute_dimensions(x, (2, 0, 1)))
    gram = K.dot(features, K.transpose(features))
    return gram

def style_loss(style, combination):
    S = gram_matrix(style)
    C = gram_matrix(combination)
    channels = 3
    size = img_height * img_width
    return K.sum(K.square(S - C)) / (4. * (channels ** 2) * (size ** 2))
```

Total variation loss

In [11]:
```
def total_variation_loss(x):
    a = K.square(
        x[:, :img_height - 1, :img_width - 1, :] -
        x[:, 1:, :img_width - 1, :])
    b = K.square(
        x[:, :img_height - 1, :img_width - 1, :] -
        x[:, :img_height - 1, 1:, :])
    return K.sum(K.pow(a + b, 1.25))
```

Defining the final loss that you'll minimize

In [12]:
```
outputs_dict = dict([(layer.name, layer.output) for layer in model.layers])
content_layer = 'block5_conv2'
style_layers = ['block1_conv1',
                'block2_conv1',
                'block3_conv1',
                'block4_conv1',
                'block5_conv1']

total_variation_weight = 1e-4
style_weight = 1.
content_weight = 0.025
loss = K.variable(0.)
layer_features = outputs_dict[content_layer]
target_image_features = layer_features[0, :, :, :]
combination_features = layer_features[2, :, :, :]
loss += content_weight * content_loss(target_image_features,
                                      combination_features)

for layer_name in style_layers:
    layer_features = outputs_dict[layer_name]
    style_reference_features = layer_features[1, :, :, :]
    combination_features = layer_features[2, :, :, :]
    sl = style_loss(style_reference_features, combination_features)
    loss += (style_weight / len(style_layers)) * sl

loss += total_variation_weight * total_variation_loss(combination_image)
```

```
WARNING:tensorflow:Variable += will be deprecated. Use variable.assign_add if you want assignment to the variable value or 'x = x + y' if you want a new python Tensor object.
```

Setting up the gradient-descent process

In [13]:
```
grads = K.gradients(loss, combination_image)[0]
fetch_loss_and_grads = K.function([combination_image], [loss, grads])

class Evaluator(object):
    def __init__(self):
        self.loss_value = None
        self.grads_values = None

    def loss(self, x):
        assert self.loss_value is None
        x = x.reshape((1, img_height, img_width, 3))
        outs = fetch_loss_and_grads([x])
        loss_value = outs[0]
        grad_values = outs[1].flatten().astype('float64')
        self.loss_value = loss_value
        self.grad_values = grad_values
        return self.loss_value

    def grads(self, x):
        assert self.loss_value is not None
        grad_values = np.copy(self.grad_values)
        self.loss_value = None
        self.grad_values = None
        return grad_values

evaluator = Evaluator()
```

Style-transfer loop

In [14]:

```
from scipy.optimize import fmin_l_bfgs_b
from scipy.misc import imsave
import time

result_prefix = 'style_transfer_result'
iterations = 20

x = preprocess_image(target_image_path)
x = x.flatten()
for i in range(iterations):
    print('Start of iteration', i)
    start_time = time.time()
    x, min_val, info = fmin_l_bfgs_b(evaluator.loss, x,
                                     fprime=evaluator.grads, maxfun=20)
    print('Current loss value:', min_val)
    img = x.copy().reshape((img_height, img_width, 3))
    img = deprocess_image(img)
    fname = result_prefix + '_at_iteration_%d.png' % i
    imsave(fname, img)
    end_time = time.time()
    print('Image saved as', fname)
    print('Iteration %d completed in %ds' % (i, end_time - start_time))
```

```
C:\Users\pablo\Python\envs\DAVID\lib\importlib\_bootstrap.py:219: RuntimeWarning: numpy.ufunc size changed, may in
dicate binary incompatibility. Expected 192 from C header, got 216 from PyObject
  return f(*args, **kwds)

Start of iteration 0
Current loss value: 2915246600.0
Image saved as style_transfer_result_at_iteration_0.png
Iteration 0 completed in 177s
Start of iteration 1

C:\Users\pablo\Python\envs\DAVID\lib\site-packages\ipykernel_launcher.py:19: DeprecationWarning: `imsave` is depre
cated!
`imsave` is deprecated in SciPy 1.0.0, and will be removed in 1.2.0.
Use ``imageio.imwrite`` instead.
```

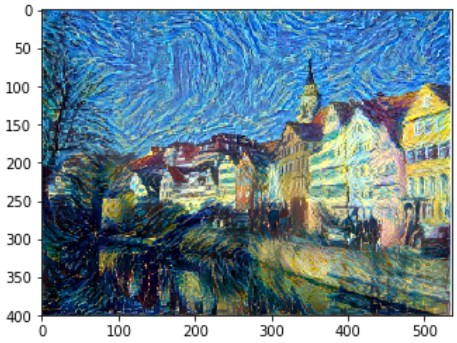
```
Current loss value: 829947900.0
Image saved as style_transfer_result_at_iteration_1.png
Iteration 1 completed in 176s
Start of iteration 2
Current loss value: 385688130.0
Image saved as style_transfer_result_at_iteration_2.png
Iteration 2 completed in 175s
Start of iteration 3
Current loss value: 257417090.0
Image saved as style_transfer_result_at_iteration_3.png
Iteration 3 completed in 175s
Start of iteration 4
Current loss value: 196536300.0
Image saved as style_transfer_result_at_iteration_4.png
Iteration 4 completed in 178s
Start of iteration 5
Current loss value: 165855780.0
Image saved as style_transfer_result_at_iteration_5.png
Iteration 5 completed in 176s
Start of iteration 6
Current loss value: 146676720.0
Image saved as style_transfer_result_at_iteration_6.png
Iteration 6 completed in 175s
Start of iteration 7
Current loss value: 131603780.0
Image saved as style_transfer_result_at_iteration_7.png
Iteration 7 completed in 176s
Start of iteration 8
Current loss value: 113504280.0
Image saved as style_transfer_result_at_iteration_8.png
Iteration 8 completed in 178s
Start of iteration 9
Current loss value: 102006100.0
Image saved as style_transfer_result_at_iteration_9.png
Iteration 9 completed in 176s
Start of iteration 10
Current loss value: 97229090.0
Image saved as style_transfer_result_at_iteration_10.png
Iteration 10 completed in 173s
Start of iteration 11
Current loss value: 91167720.0
Image saved as style_transfer_result_at_iteration_11.png
Iteration 11 completed in 181s
Start of iteration 12
Current loss value: 86615690.0
Image saved as style_transfer_result_at_iteration_12.png
Iteration 12 completed in 165s
Start of iteration 13
Current loss value: 83729360.0
Image saved as style_transfer_result_at_iteration_13.png
Iteration 13 completed in 179s
Start of iteration 14
Current loss value: 81145860.0
Image saved as style_transfer_result_at_iteration_14.png
Iteration 14 completed in 177s
Start of iteration 15
Current loss value: 77914830.0
Image saved as style_transfer_result_at_iteration_15.png
Iteration 15 completed in 169s
Start of iteration 16
Current loss value: 75213780.0
Image saved as style_transfer_result_at_iteration_16.png
Iteration 16 completed in 187s
Start of iteration 17
Current loss value: 70693940.0
Image saved as style_transfer_result_at_iteration_17.png
Iteration 17 completed in 181s
Start of iteration 18
Current loss value: 67415390.0
Image saved as style_transfer_result_at_iteration_18.png
Iteration 18 completed in 162s
Start of iteration 19
Current loss value: 65714612.0
Image saved as style_transfer_result_at_iteration_19.png
Iteration 19 completed in 166s
```

In [16]:
```
from matplotlib import pyplot as plt

# Content image
plt.imshow(load_img(target_image_path, target_size=(img_height, img_width)))
plt.figure()

# Style image
plt.imshow(load_img(style_reference_image_path, target_size=(img_height, img_width)))
plt.figure()

# Generate image
plt.imshow(img)
plt.show()
```



Pablo Minango

- pablodavid218@gmail.com