```
"""
学習済みのCNN（VGG19）を使用し、
ターゲット画像のコンテンツを維持した上で、
リファレンス画像のスタイルをターゲット画像に適用する
"""

import os
from keras.preprocessing.image import load_img, img_to_array
import numpy as np
from keras.applications import vgg19
from keras import backend as K
import time
from scipy.optimize import fmin_l_bfgs_b
from scipy.misc import imsave

dir_path = "/content/drive/My Drive/Python/Style Transfer/"
os.chdir(dir_path)

#  変数の定義
target_image_path = 'target.jpg'
style_image_path = 'style.jpg'

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

#  補助関数
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

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

#  学習済みのVGG19ネットワークを読み込み、3つの画像に適用
target_image = K.constant(preprocess_image(target_image_path))
style_image = K.constant(preprocess_image(style_image_path))
combination_image = K.placeholder((1, img_height, img_width, 3))

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

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

print('Model loaded')

#  コンテンツの損失関数
def content_loss(content, combination):
    return K.sum(K.square(content - combination))

#  スタイルの損失関数
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))

#  全変動損失関数
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))

#  最小化の対象となる最終的な損失関数を定義
outputs_dict = dict([(layer.name, layer.output) for layer in model.layers])

content_layer = 'block5_conv2'

style_layer = ['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]
content_features = layer_features[0, :, :, :]
combination_features = layer_features[2, :, :, :]
loss = loss + content_weight * content_loss(content_features,
                                    combination_features)

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

loss += total_variation_weight * total_variation_loss(combination_image)

#  勾配降下法のプロセスを定義
```

```
# 勾配降下法のステップを定義
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_value = 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]
        grads_value = outs[1].flatten().astype('float64')
        self.loss_value = loss_value
        self.grads_value = grads_value
        return self.loss_value

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

evaluator = Evaluator()

#  スタイル変換ループ
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 = '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))
```

Model: "vgg19"

Layer (type)	Output Shape	Param #
input_8 (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
_________________________________________________________________
Model loaded
Start of iteration 0
Current loss value: 3223519700.0
Image saved as iteration_0.png
Iteration 0 completed in 11s
Start of iteration 1
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:141: DeprecationWarning: `imsave` is deprecated!
`imsave` is deprecated in SciPy 1.0.0, and will be removed in 1.2.0.
Use ``imageio.imwrite`` instead.
Current loss value: 1093964400.0
Image saved as iteration_1.png
Iteration 1 completed in 4s
Start of iteration 2
Current loss value: 663668100.0
Image saved as iteration_2.png
Iteration 2 completed in 4s
Start of iteration 3
Current loss value: 513920500.0
Image saved as iteration_3.png
Iteration 3 completed in 4s
Start of iteration 4
Current loss value: 422785400.0
Image saved as iteration_4.png
Iteration 4 completed in 4s
Start of iteration 5
Current loss value: 366054660.0
Image saved as iteration_5.png
Iteration 5 completed in 4s
Start of iteration 6
Current loss value: 324812830.0
Image saved as iteration_6.png
Iteration 6 completed in 4s
```

```
Start of iteration 7
Current loss value: 286410620.0
Image saved as iteration_7.png
Iteration 7 completed in 5s
Start of iteration 8
Current loss value: 261980160.0
Image saved as iteration_8.png
Iteration 8 completed in 4s
Start of iteration 9
Current loss value: 239920220.0
Image saved as iteration_9.png
Iteration 9 completed in 5s
Start of iteration 10
Current loss value: 224403900.0
Image saved as iteration_10.png
Iteration 10 completed in 4s
Start of iteration 11
Current loss value: 207010130.0
Image saved as iteration_11.png
Iteration 11 completed in 5s
Start of iteration 12
Current loss value: 196127790.0
Image saved as iteration_12.png
Iteration 12 completed in 4s
Start of iteration 13
Current loss value: 186923180.0
Image saved as iteration_13.png
Iteration 13 completed in 4s
Start of iteration 14
Current loss value: 179837140.0
Image saved as iteration_14.png
Iteration 14 completed in 4s
Start of iteration 15
Current loss value: 174138350.0
Image saved as iteration_15.png
Iteration 15 completed in 4s
Start of iteration 16
Current loss value: 168209280.0
Image saved as iteration_16.png
Iteration 16 completed in 4s
Start of iteration 17
Current loss value: 162087870.0
```

```
Image saved as iteration_17.png
Iteration 17 completed in 5s
Start of iteration 18
Current loss value: 156716540.0
```