Style Transfer - Keras

Import pacakges

In [1]:
```
from scipy.misc import imsave
from IPython.display import Image
from IPython.core.display import display, HTML
import numpy as np
from scipy.optimize import fmin_l_bfgs_b
import time
import argparse


# Keras Components
from keras.preprocessing.image import load_img, img_to_array
from keras.applications import vgg19

from helper_functions import *
```

```
Using TensorFlow backend.
```

Inputs

In [2]:
```
base_image_path = 'inputs/cat.jpg'
style_reference_image_path = 'inputs/flowers.jpg'

display(HTML(  '<img src="%s" alt="Drawing" style="height: 320px; float:left; margin:0px; margin-left:10px"/>'*2 % (base_image_
path,style_reference_image_path) ))
```


What is Style Transfer?

The aim of style transfer is to generate a "combination" image with the same "content" as a base image, but with the "style" of a different picture. This is achieved by optermising the weighted sum of 3 loss functions

Style Loss

The style loss is where the deep learning keeps in --that one is defined using a deep convolutional neural network. Precisely, it consists in a sum of L2 distances between the Gram matrices of the representations of the base image and the style reference image, extracted from different layers of a convnet (trained on ImageNet). The general idea is to capture color/texture information at different spatial scales (fairly large scales --defined by the depth of the layer considered).

In [3]:
```
def style_loss(style, combination):
    assert K.ndim(style) == 3
    assert K.ndim(combination) == 3
    S = gram_matrix(style)
    C = gram_matrix(combination)
    channels = 3
    size = img_nrows * img_ncols
    return K.sum(K.square(S - C)) / (4. * (channels ** 2) * (size ** 2))
```

Content Loss

The content loss is a L2 distance between the features of the base image (extracted from a deep layer) and the features of the combination image, keeping the generated image close enough to the original one.

In [4]:
```
def content_loss(base, combination):
    return K.sum(K.square(combination - base))
```

Total Variation Loss

Imposes local spatial continuity between the pixels of the combination image, giving it visual coherence.

In [5]:
```
def total_variation_loss(x):
    assert K.ndim(x) == 4
    if K.image_data_format() == 'channels_first':
        a = K.square(x[:, :, :img_nrows - 1, :img_ncols - 1] - x[:, :, 1:, :img_ncols - 1])
        b = K.square(x[:, :, :img_nrows - 1, :img_ncols - 1] - x[:, :, :img_nrows - 1, 1:])
    else:
        a = K.square(x[:, :img_nrows - 1, :img_ncols - 1, :] - x[:, 1:, :img_ncols - 1, :])
        b = K.square(x[:, :img_nrows - 1, :img_ncols - 1, :] - x[:, :img_nrows - 1, 1:, :])
    return K.sum(K.pow(a + b, 1.25))
```

Initialise

Output Image Size

In [6]:
```
width, height = load_img(base_image_path).size
img_nrows = 400
img_ncols = int(width * img_nrows / height)
```

Tensor Representations of Images

In [7]:
```
base_image = K.variable(preprocess_image(base_image_path, img_nrows, img_ncols ))
style_reference_image = K.variable(preprocess_image(style_reference_image_path, img_nrows, img_ncols ))
```

In [8]:
```
# placeholder that will contain our generated image
if K.image_data_format() == 'channels_first':
    combination_image = K.placeholder((1, 3, img_nrows, img_ncols))
else:
    combination_image = K.placeholder((1, img_nrows, img_ncols, 3))
```

Model Representations of Tensors (vgg19)

In [9]:
```
# combine the 3 images into a single Keras tensor
input_tensor = K.concatenate([base_image, style_reference_image, combination_image], axis=0)
```

In [10]:
```
# build the VGG16 network with our 3 images as input
# the model will be loaded with pre-trained ImageNet weights
model = vgg19.VGG19(input_tensor=input_tensor,weights='imagenet', include_top=False)
# get the symbolic outputs of each "key" layer (we gave them unique names).
outputs_dict = { layer.name : layer.output for layer in model.layers }
```

Loss/gradient functions

Set the weights of each of each loss function

In [11]:
```
total_variation_weight, style_weight, content_weight = 1.0, 1.0, 0.025
```

Combine these loss functions into a single scalar

In [12]:
```
loss = K.variable(0.)
layer_features = outputs_dict['block5_conv2']
base_image_features = layer_features[0, :, :, :]
combination_features = layer_features[2, :, :, :]
loss += content_weight * content_loss(base_image_features, combination_features)

feature_layers = ['block1_conv1', 'block2_conv1','block3_conv1', 'block4_conv1','block5_conv1']

for layer_name in feature_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(feature_layers)) * sl
loss += total_variation_weight * total_variation_loss(combination_image)
```

get the gradients of the generated image wrt the loss

In [13]:
```
grads = K.gradients(loss, combination_image)
outputs = [loss] + grads if isinstance(grads, (list, tuple) ) else [loss, grads ]
f_outputs = K.function([combination_image], outputs)
```

In [14]:
```
def eval_loss_and_grads(x):
    if K.image_data_format() == 'channels_first':
        x = x.reshape((1, 3, img_nrows, img_ncols))
    else:
        x = x.reshape((1, img_nrows, img_ncols, 3))
    outs = f_outputs([x])
    loss_value = outs[0]
    if len(outs[1:]) == 1:
        grad_values = outs[1].flatten().astype('float64')
    else:
        grad_values = np.array(outs[1:]).flatten().astype('float64')
    return loss_value, grad_values
```

this Evaluator class makes it possible to compute loss and gradients in one pass while retrieving them via two separate functions, "loss" and "grads". This is done because scipy.optimize requires separate functions for loss and gradients, but computing them separately would be inefficient.

In [15]:
```
class Evaluator(object):

    def __init__(self):
        self.loss_value = None
        self.grads_values = None

    def loss(self, x):
        assert self.loss_value is None
        loss_value, grad_values = eval_loss_and_grads(x)
        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
```

In [ ]:
```
evaluator = Evaluator()
```

Generating the output

Run scipy-based optimization (L-BFGS) over the pixels of the generated image so as to minimize the neural style loss

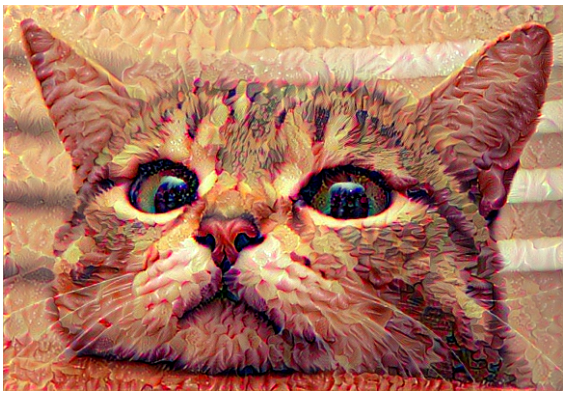
In [ ]:
```
## initialise with
x = preprocess_image(base_image_path , img_nrows, img_ncols )

iterations = 400

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.flatten(), fprime=evaluator.grads, maxfun=20)
    if i % 20 == 0:
        print('Current loss value:', min_val)
    # save current generated image
    img = deprocess_image(x.copy() , img_nrows, img_ncols )
    fname = 'outputs/combination_at_iteration_%d.png' % i
    imsave(fname, img)
    end_time = time.time()
    if i % 20 == 0:
        display(HTML(  '<img src="%s" alt="Drawing" style="height: 320px; margin:0px; margin-bottom:10px"/>' % fname ))
        print('Image saved as', fname)
```

```
Start of iteration 0
Current loss value: 1.14705e+10
```



```
Image saved as outputs/combination_at_iteration_0.png
Start of iteration 1
Start of iteration 2
Start of iteration 3
Start of iteration 4
Start of iteration 5
Start of iteration 6
Start of iteration 7
Start of iteration 8
Start of iteration 9
Start of iteration 10
Start of iteration 11
Start of iteration 12
Start of iteration 13
Start of iteration 14
Start of iteration 15
Start of iteration 16
Start of iteration 17
Start of iteration 18
Start of iteration 19
Start of iteration 20
Current loss value: 1.96545e+09
```

```
Image saved as outputs/combination_at_iteration_20.png
Start of iteration 21
Start of iteration 22
Start of iteration 23
Start of iteration 24
Start of iteration 25
Start of iteration 26
Start of iteration 27
Start of iteration 28
Start of iteration 29
Start of iteration 30
Start of iteration 31
Start of iteration 32
Start of iteration 33
Start of iteration 34
Start of iteration 35
Start of iteration 36
Start of iteration 37
Start of iteration 38
Start of iteration 39
Start of iteration 40
Current loss value: 1.83088e+09
```

```
Image saved as outputs/combination_at_iteration_40.png
Start of iteration 41
Start of iteration 42
Start of iteration 43
Start of iteration 44
Start of iteration 45
Start of iteration 46
Start of iteration 47
Start of iteration 48
Start of iteration 49
Start of iteration 50
Start of iteration 51
Start of iteration 52
Start of iteration 53
Start of iteration 54
Start of iteration 55
Start of iteration 56
Start of iteration 57
Start of iteration 58
Start of iteration 59
Start of iteration 60
Current loss value: 1.77367e+09
```

```
Image saved as outputs/combination_at_iteration_60.png
Start of iteration 61
Start of iteration 62
Start of iteration 63
Start of iteration 64
Start of iteration 65
Start of iteration 66
Start of iteration 67
Start of iteration 68
Start of iteration 69
Start of iteration 70
Start of iteration 71
Start of iteration 72
Start of iteration 73
Start of iteration 74
Start of iteration 75
Start of iteration 76
Start of iteration 77
Start of iteration 78
Start of iteration 79
Start of iteration 80
Current loss value: 1.74056e+09
```

```
Image saved as outputs/combination_at_iteration_80.png
Start of iteration 81
Start of iteration 82
Start of iteration 83
Start of iteration 84
Start of iteration 85
Start of iteration 86
Start of iteration 87
Start of iteration 88
Start of iteration 89
Start of iteration 90
Start of iteration 91
Start of iteration 92
Start of iteration 93
Start of iteration 94
Start of iteration 95
Start of iteration 96
Start of iteration 97
Start of iteration 98
Start of iteration 99
Start of iteration 100
Current loss value: 1.7179e+09
```

```
Image saved as outputs/combination_at_iteration_100.png
Start of iteration 101
Start of iteration 102
Start of iteration 103
Start of iteration 104
Start of iteration 105
Start of iteration 106
Start of iteration 107
Start of iteration 108
Start of iteration 109
Start of iteration 110
Start of iteration 111
Start of iteration 112
Start of iteration 113
Start of iteration 114
Start of iteration 115
Start of iteration 116
Start of iteration 117
Start of iteration 118
Start of iteration 119
Start of iteration 120
Current loss value: 1.70059e+09
```

```
Image saved as outputs/combination_at_iteration_120.png
Start of iteration 121
Start of iteration 122
Start of iteration 123
Start of iteration 124
Start of iteration 125
Start of iteration 126
Start of iteration 127
Start of iteration 128
Start of iteration 129
Start of iteration 130
Start of iteration 131
Start of iteration 132
Start of iteration 133
Start of iteration 134
Start of iteration 135
Start of iteration 136
Start of iteration 137
Start of iteration 138
Start of iteration 139
Start of iteration 140
Current loss value: 1.68749e+09
```

```
Image saved as outputs/combination_at_iteration_140.png
Start of iteration 141
Start of iteration 142
Start of iteration 143
Start of iteration 144
Start of iteration 145
Start of iteration 146
Start of iteration 147
Start of iteration 148
Start of iteration 149
Start of iteration 150
Start of iteration 151
Start of iteration 152
Start of iteration 153
Start of iteration 154
Start of iteration 155
Start of iteration 156
Start of iteration 157
Start of iteration 158
Start of iteration 159
Start of iteration 160
Current loss value: 1.67723e+09
```

```
Image saved as outputs/combination_at_iteration_160.png
Start of iteration 161
Start of iteration 162
Start of iteration 163
Start of iteration 164
Start of iteration 165
Start of iteration 166
Start of iteration 167
Start of iteration 168
Start of iteration 169
Start of iteration 170
Start of iteration 171
Start of iteration 172
Start of iteration 173
Start of iteration 174
Start of iteration 175
Start of iteration 176
```

In [ ]:

In [ ]: