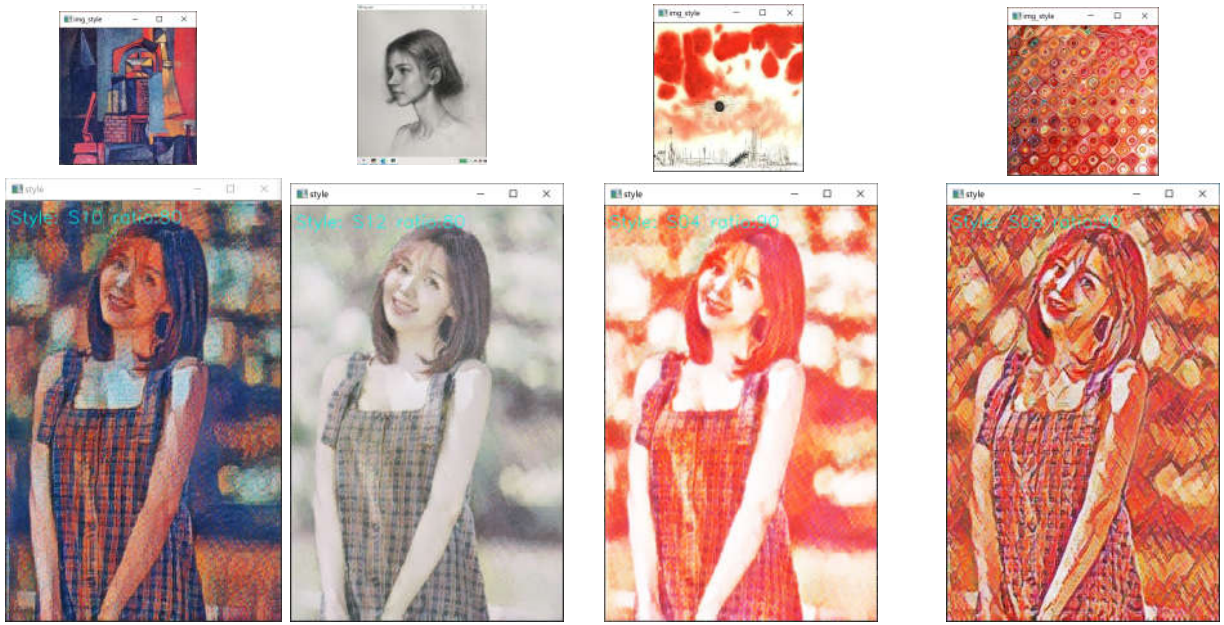
# Python编程与人工智能实践

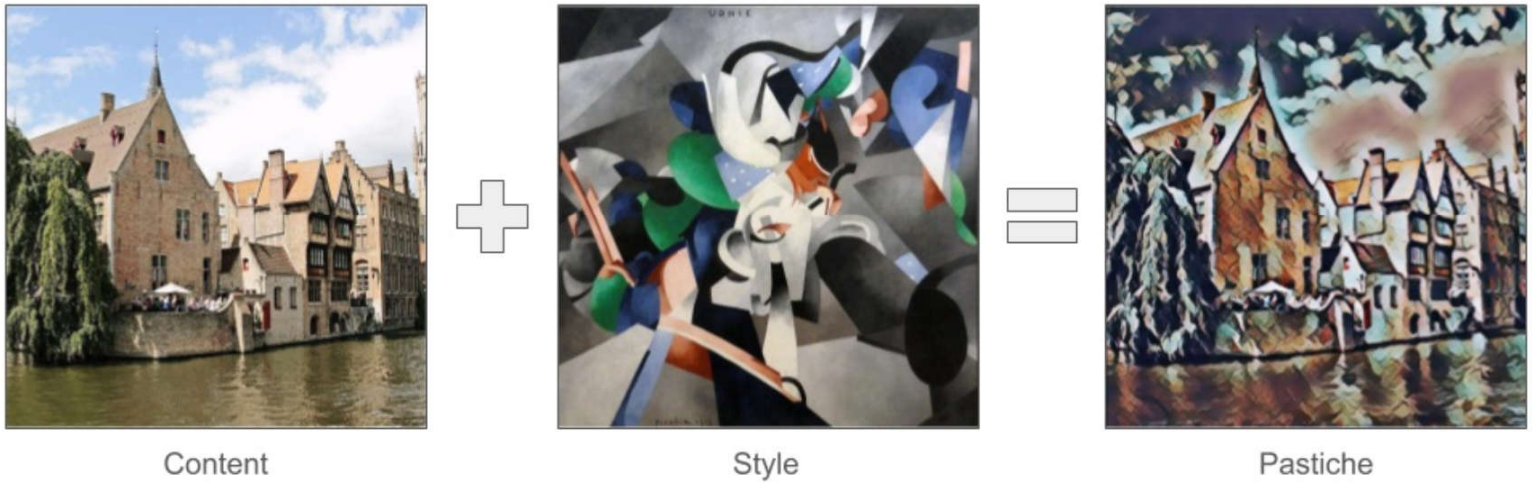
# 应用篇：基于tflite的图像风格迁移



于泓
鲁东大学
信息与电气工程学院
2021.5.22

# 风格迁移

图像风格迁移是近年来深度神经网络技术发展的一个有趣的新方向，利用深度神经网络可以将一张图像的风格转换成另外一种风格，即所谓的模仿混合画（pastiche）。



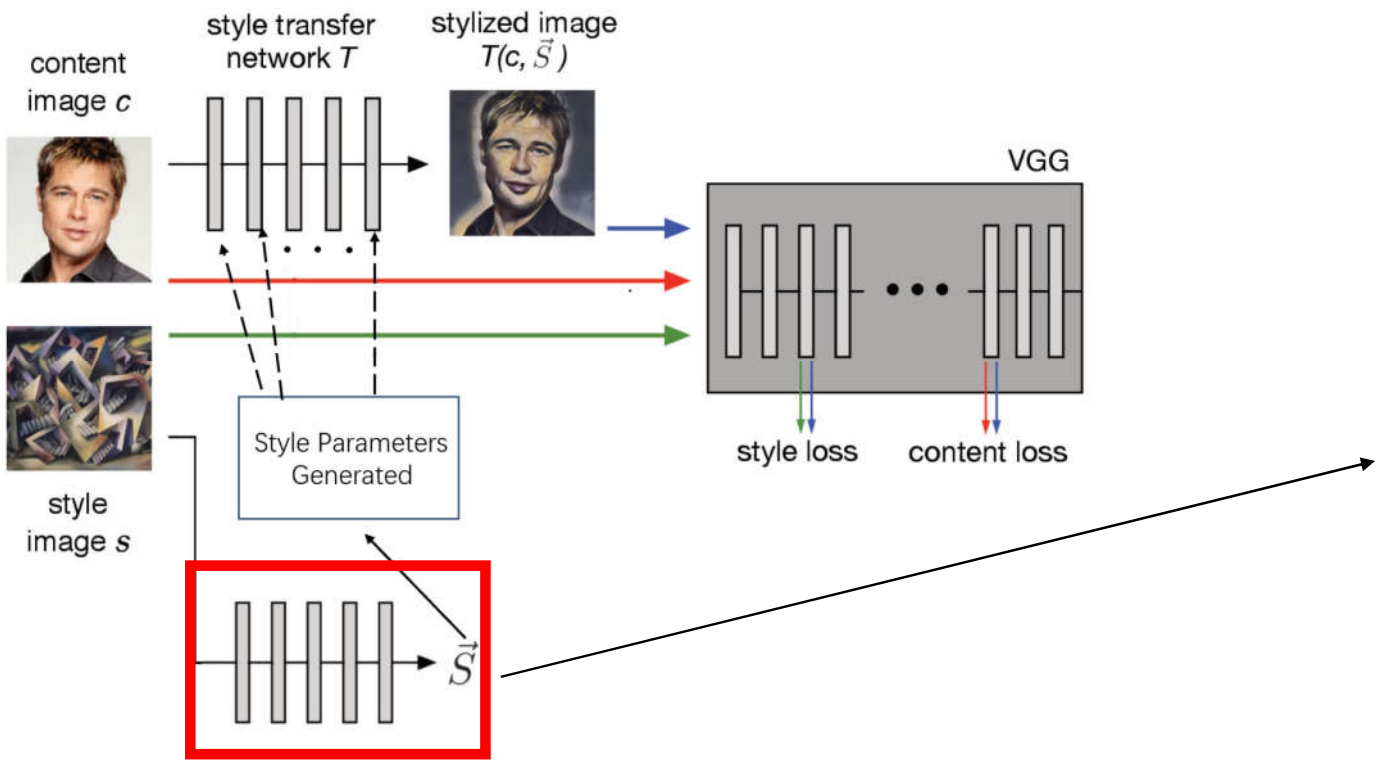

风格迁移的过程


不同风格迁移的效果

# 风格迁移的基本原理

参考文献：Exploring the structure of a real-time, arbitrary neural artistic stylization network

特点: 可以实现对未经训练的风格图像进行迁移，并且可以人工的控制迁移的比重。

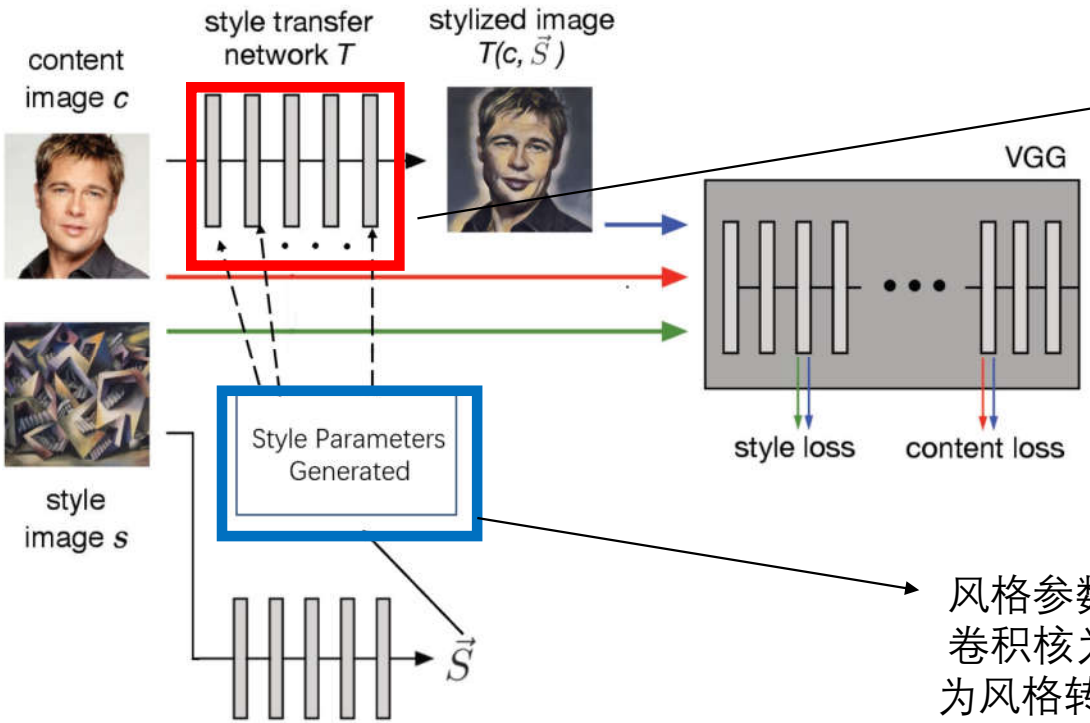



**风格特征生成网络**
用来生成风格特征矢量 $\vec{S}$

# 风格迁移的基本原理

参考文献：Exploring the structure of a real-time, arbitrary neural artistic stylization network

特点: 可以实现对未经训练的风格图像进行迁移，并且可以人工的控制迁移的比重。




**风格转换网络：CNN-deCNN**
用来生成风格转换后的图像

$$\tilde{z} = \gamma_s\left(\frac{z-\mu}{\delta}\right) + \beta_s$$

风格参数

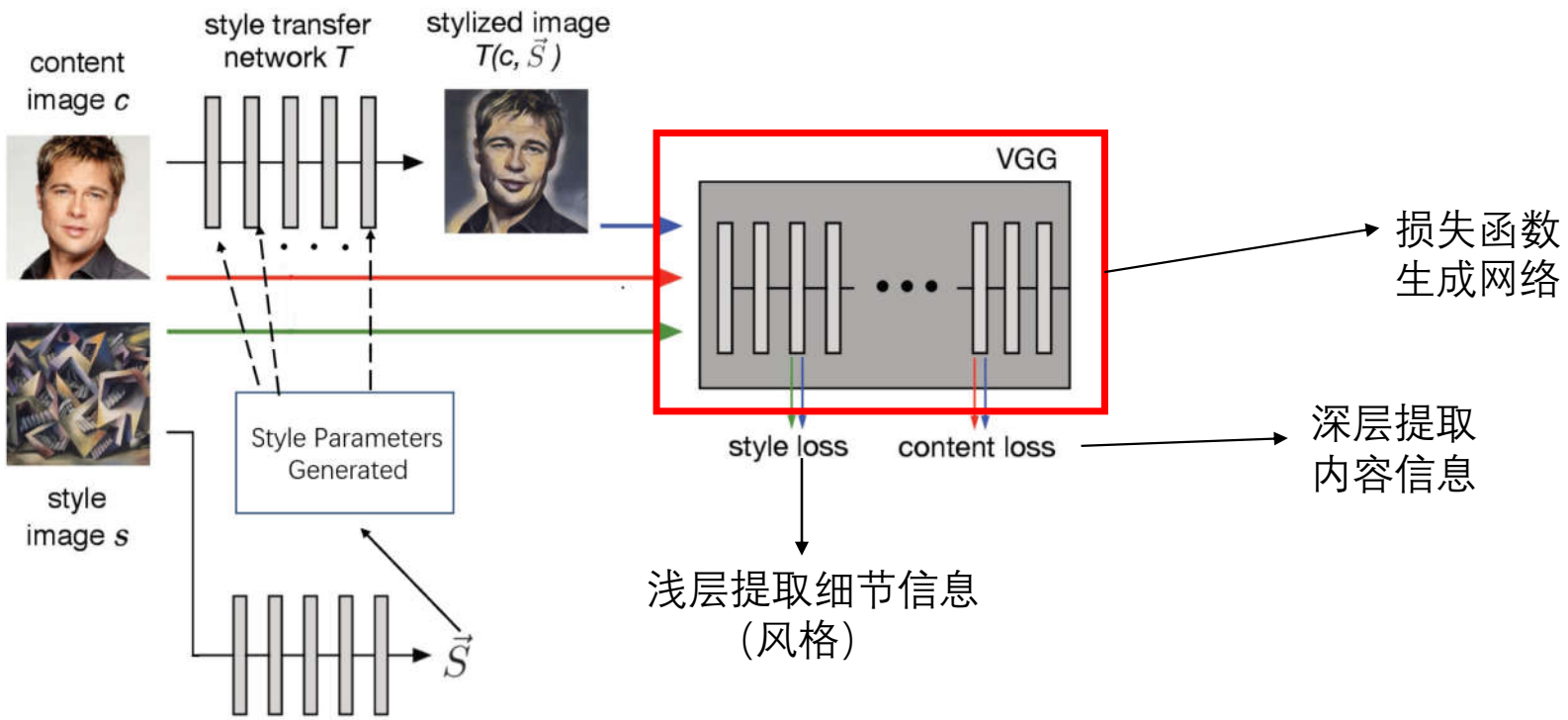
$z$ 为每一层的输出 $\tilde{z}$ 下一层输入

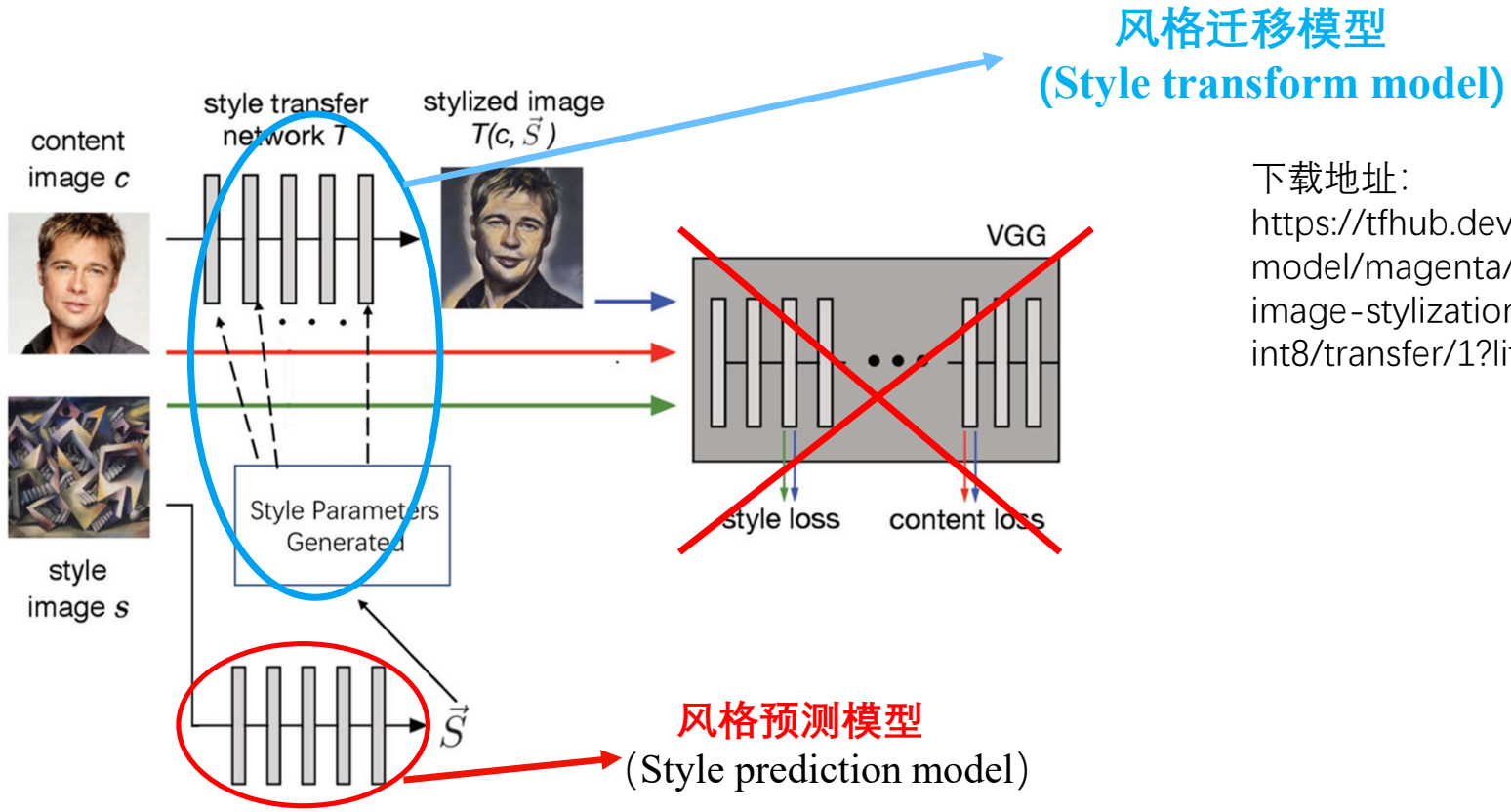
风格参数生成网络
卷积核为1×1的CNN，
为风格转换网络的每一层
生成转换参数

# 风格迁移的基本原理

参考文献：Exploring the structure of a real-time, arbitrary neural artistic stylization network

特点: 可以实现对未经训练的风格图像进行迁移，并且可以人工的控制迁移的比重。

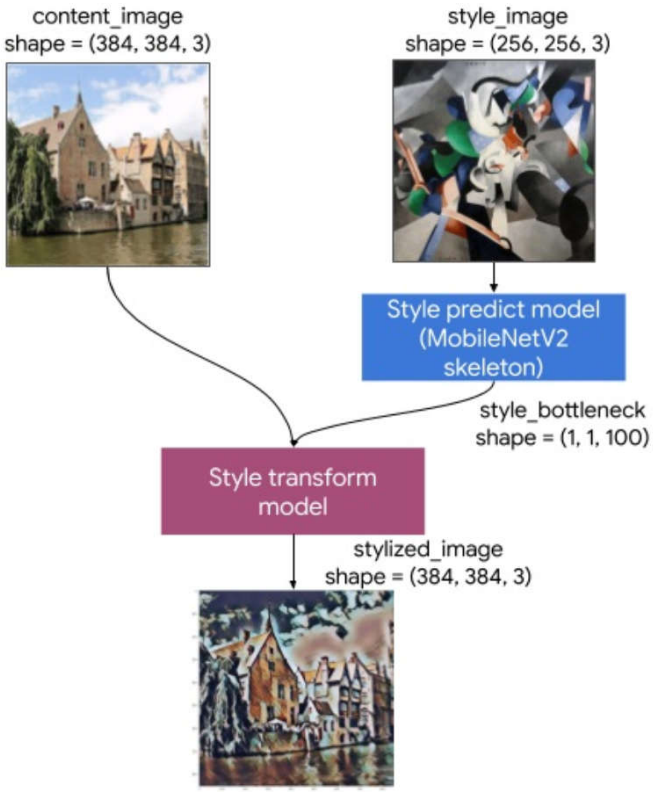


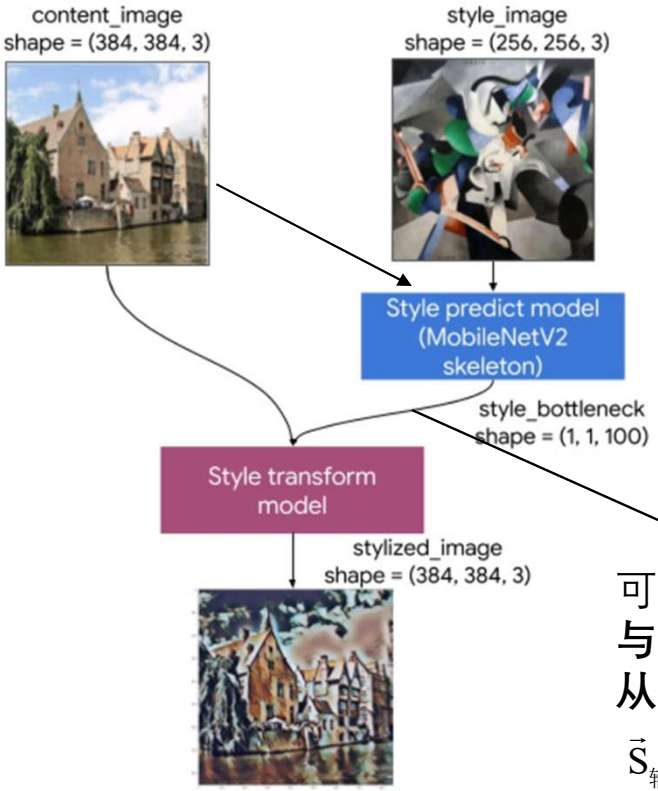



下载地址：
https://tfhub.dev/google/lite-model/magenta/arbitrary-image-stylization-v1-256/int8/transfer/1?lite-format=tflite

下载地址：https://tfhub.dev/google/lite-model/magenta/arbitrary-image-stylization-v1-256/int8/prediction/1?lite-format=tflite

## 风格转换过程




可以将内容的风格矢量与 风格的风格矢量融合从而调整风格的比重

$$\vec{S}_{输入} = \alpha \vec{S}_{风格} + (1-\alpha)\vec{S}_{内容}$$

# 模型信息提取

```
# 风格特征提取模型
file_model_prediction = "model/magenta_arbitrary-image-stylization-v1-256_int8_prediction_1.tflite"

# 风格转换模型
file_model_transfer = "model/magenta_arbitrary-image-stylization-v1-256_int8_transfer_1.tflite"

# 加载模型
print("model_prediction")
interpreter = tflite.Interpreter(model_path=file_model_prediction)
interpreter.allocate_tensors()
# 获取输入信息
input_details = interpreter.get_input_details()
print("input_details",input_details)
outputdetails = interpreter.get_output_details()
print("outputdetails",outputdetails)


# 加载模型
print("model_transfer")
interpreter = tflite.Interpreter(model_path=file_model_transfer)
interpreter.allocate_tensors()
# 获取输入信息
input_details = interpreter.get_input_details()
print("input_details",input_details)
outputdetails = interpreter.get_output_details()
print("outputdetails",outputdetails)
```

model_prediction 风格特征提取网络

输入图像大小

数据格式
取值范围 [0,1] data/255.0

```
input_details
[{'name': 'style_image', 'index': 172, 'shape': array([  1, 256, 256,   3]),
 'shape_signature': array([  1, 256, 256,   3]), 'dtype': <class 'numpy.float32'>,
 'quantization': (0.0, 0), 'quantization_parameters': {'scales': array([], dtype=float32),
 'zero_points': array([], dtype=int32), 'quantized_dimension': 0}, 'sparsity_parameters': {}}]

outputdetails
[{'name': 'mobilenet_conv/Conv/BiasAdd', 'index': 173, 'shape': array([  1,   1,   1, 100]),
'shape_signature': array([  1,   1,   1, 100]), 'dtype': <class 'numpy.float32'>,
'quantization': (0.0, 0), 'quantization_parameters': {'scales': array([], dtype=float32),
 'zero_points': array([], dtype=int32), 'quantized_dimension': 0}, 'sparsity_parameters': {}}]
```

输出特征 100维

model_transfer 风格转换网络

输入图像大小

数据格式
取值范围 [0,1] data/255.0

```
input_details
[{'name': 'content_image', 'index': 0, 'shape': array([  1, 384, 384,   3]),
  'shape_signature': array([  1, 384, 384,   3]), 'dtype': <class 'numpy.float32'>,
  'quantization': (0.0, 0), 'quantization_parameters': {'scales': array([], dtype=float32),
  'zero_points': array([], dtype=int32), 'quantized_dimension': 0}, 'sparsity_parameters': {}},

  {'name': 'mobilenet_conv/Conv/BiasAdd', 'index': 428, 'shape': array([  1,   1,   1, 100]),
  'shape_signature': array([  1,   1,   1, 100]), 'dtype': <class 'numpy.float32'>,
  'quantization': (0.0, 0), 'quantization_parameters': {'scales': array([], dtype=float32),
  'zero_points': array([], dtype=int32), 'quantized_dimension': 0}, 'sparsity_parameters': {}}]

outputdetails
[{'name': 'transformer/expand/conv3/conv/Sigmoid',
  'index': 429, 'shape': array([  1, 384, 384,   3]),
  'shape_signature': array([  1, 384, 384,   3]), 'dtype': <class 'numpy.float32'>,
  'quantization': (0.0, 0), 'quantization_parameters': {'scales': array([], dtype=float32),
  'zero_points': array([], dtype=int32), 'quantized_dimension': 0}, 'sparsity_parameters': {}}]
```

风格特征矢量

输出图像大小

输出值的范围为[0,1]
*255转为像素值

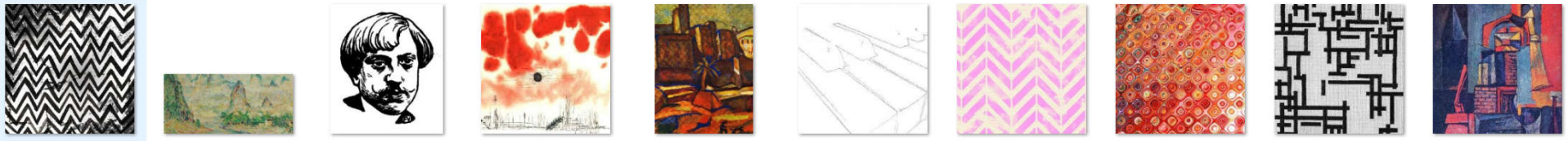
# 一　风格特征提取

fea_style
img_style
model

三个文件夹，分别用来存储，风格图片（img_style）
风格特征（fea_style）
下载的模型（model）

采集一些风格图片保存到img_style文件夹
(没有什么特别要求，可以到百度图片搜索纹理)

此电脑 › Data (D:) › 工作相关 › 我设计的课程 › python与人工智能课程设计 › 应用篇 › style_trans › img_style　搜索"img_style"




## collect_fea_style.py

```
import numpy as np
import cv2
import tflite_runtime.interpreter as tflite
import os

# 风格特征提取模型
file_model_prediction = "model/magenta_arbitrary-image-stylization-v1-256_int8_prediction_1.tflite"
base_path ="img_style"

def img_preprocessing(img,size_output):
    # 获取图像的尺寸
    imH,imW,_ = np.shape(img)

    # BGR 转RGB
    img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

    # 尺寸缩放适应网络输入要求
    img_resized = cv2.resize(img_rgb, size_output)

    # 维度扩张适应网络输入要求
    input_data = np.expand_dims(img_resized, axis=0)

    # 正则化变为 0-1之间
    input_data = input_data/255.0

    return (imW,imH), np.float32(input_data)
```

图像预处理

# 风格特征提取

```python
# 从风格图片中提取风格特征矢量.
def run_style_predict(file_model_prediction,img):

    # 加载模型
    interpreter = tflite.Interpreter(model_path=file_model_prediction)
    interpreter.allocate_tensors()

    # 获取输入的数据的信息
    input_details = interpreter.get_input_details()

    # 获取输入图像的高和宽
    height = input_details[0]['shape'][1]
    width = input_details[0]['shape'][2]

    # 对图像进行预处理
    t,preprocessed_style_image = img_preprocessing(img,(width,height))

    # 特征输入网络
    interpreter.set_tensor(input_details[0]["index"], preprocessed_style_image)

    # 获取模型矢量.
    interpreter.invoke()

    style_bottleneck = interpreter.tensor(interpreter.get_output_details()[0]["index"])()

    return style_bottleneck
```

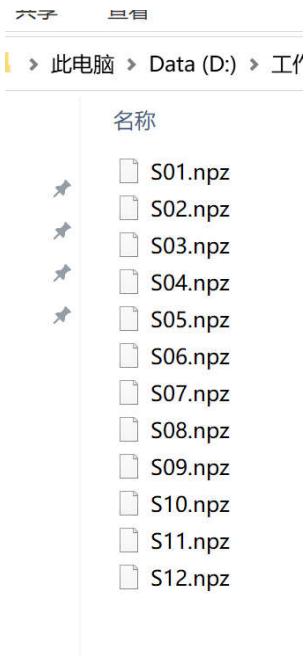
```python
if __name__ == "__main__":
    # 遍历文件夹
    for f_img in os.listdir(base_path):
        # 遍历所有以.jpg为后缀的文件
        if os.path.splitext(f_img)[-1] == ".jpg":
            str_style =  os.path.splitext(f_img)[0]
            print(str_style)
            f_img = os.path.join(base_path,f_img)

            # 提取风格特征
            style_bottleneck = run_style_predict(file_model_prediction,cv2.imread(f_img))

            # 特征保存
            print('save style feature of image %s as %s.npz'%(f_img,os.path.join('fea_style',str_style)))

            np.savez(os.path.join('fea_style',str_style),fea = style_bottleneck)
```

```
PS D:\工作相关\我设计的课程\python与人工智能课程设计\应用篇\style_trans> python collect_fea_style.py
S01
save style feature of image img_style\S01.jpg as fea_style\S01.npz
S02
save style feature of image img_style\S02.jpg as fea_style\S02.npz
S03
save style feature of image img_style\S03.jpg as fea_style\S03.npz
S04
save style feature of image img_style\S04.jpg as fea_style\S04.npz
S05
save style feature of image img_style\S05.jpg as fea_style\S05.npz
S06
save style feature of image img_style\S06.jpg as fea_style\S06.npz
S07
save style feature of image img_style\S07.jpg as fea_style\S07.npz
S08
save style feature of image img_style\S08.jpg as fea_style\S08.npz
S09
save style feature of image img_style\S09.jpg as fea_style\S09.npz
S10
save style feature of image img_style\S10.jpg as fea_style\S10.npz
S11
save style feature of image img_style\S11.jpg as fea_style\S11.npz
S12
save style feature of image img_style\S12.jpg as fea_style\S12.npz
```



# 二、风格转换

```
import numpy as np
import cv2
import tflite_runtime.interpreter as tflite
import os
from collect_fea_style import img_preprocessing,run_style_predict

# 风格特征提取模型
file_model_prediction = "model/magenta_arbitrary-image-stylization-v1-256_int8_prediction_1.tflite"
# 风格转换模型
file_model_transfer = "model/magenta_arbitrary-image-stylization-v1-256_int8_transfer_1.tflite"
# 进行风格转换 输入风格矢量 、 风格转换模型 、

def run_style_transform(file_model_transfer,style_bottleneck, content_image):
    im_H,im_W,_ = np.shape(content_image)

    # 加载模型
    interpreter = tflite.Interpreter(model_path=file_model_transfer)
    interpreter.allocate_tensors()

    # 获取输入信息
    input_details = interpreter.get_input_details()

    # 获取输入图像的高和宽
    height = input_details[0]['shape'][1]
    width = input_details[0]['shape'][2]

    # 对内容图像进行预处理
    t,preprocessed_content_image = img_preprocessing(content_image,(width,height))

    # 将数据送入模型.
    interpreter.set_tensor(input_details[0]["index"], preprocessed_content_image)
    interpreter.set_tensor(input_details[1]["index"], style_bottleneck)
    interpreter.invoke()
```

```
# 得到风格变换后的图像.
stylized_image = interpreter.tensor(interpreter.get_output_details()[0]["index"])()

# 将输出从0-1 转为 0-255 浮点转uint8
stylized_image = np.uint8(stylized_image*255)
stylized_image = np.squeeze(stylized_image)
# RGB 转 BGR
stylized_image = cv2.cvtColor(stylized_image, cv2.COLOR_RGB2BGR)
stylized_image = cv2.resize(stylized_image,(im_W,im_H))
return stylized_image


if __name__ == "__main__":
    # 加载风格特征
    path_fea_style= 'fea_style'
    path_img_style = "img_style"
    str_style = 'S10'
    style_fea = np.load(os.path.join(path_fea_style,str_style+'.npz'))['fea']

    # 显示风格图片
    img_style = cv2.imread(os.path.join(path_img_style,str_style+'.jpg'))
    cv2.imshow("img_style",img_style)

    # 读取原始图像
    img = cv2.imread("test.jpg")
    # 获取图像帧的尺寸
    imH,imW,_ = np.shape(img)
    # 适当缩放
    img = cv2.resize(img,(int(imW*0.8),int(imH*0.8)))
    cv2.imshow("img",img)
```

显示风格图像

显示原始图像

```
# 获取内容图像的风格特征
content_fea = run_style_predict(file_model_prediction,img)

# 设置风格比例
ratio =20
mix_fea = ratio*0.01*style_fea +(1-ratio*0.01)*content_fea

print("start processing Style=%s ratio=%d%%"%(str_style,ratio))
stylized_image = run_style_transform(file_model_transfer,mix_fea, img)
print("processing end")
cv2.putText(stylized_image,'Style: %s ratio:%d'%(str_style,ratio),
                              (5,30),cv2.FONT_HERSHEY_SIMPLEX,
                               0.8,(255,255,0),1,cv2.LINE_AA)
cv2.imshow('style', stylized_image)
```

内容图片风格与
风格图片风格融合

显示风格化后的图像

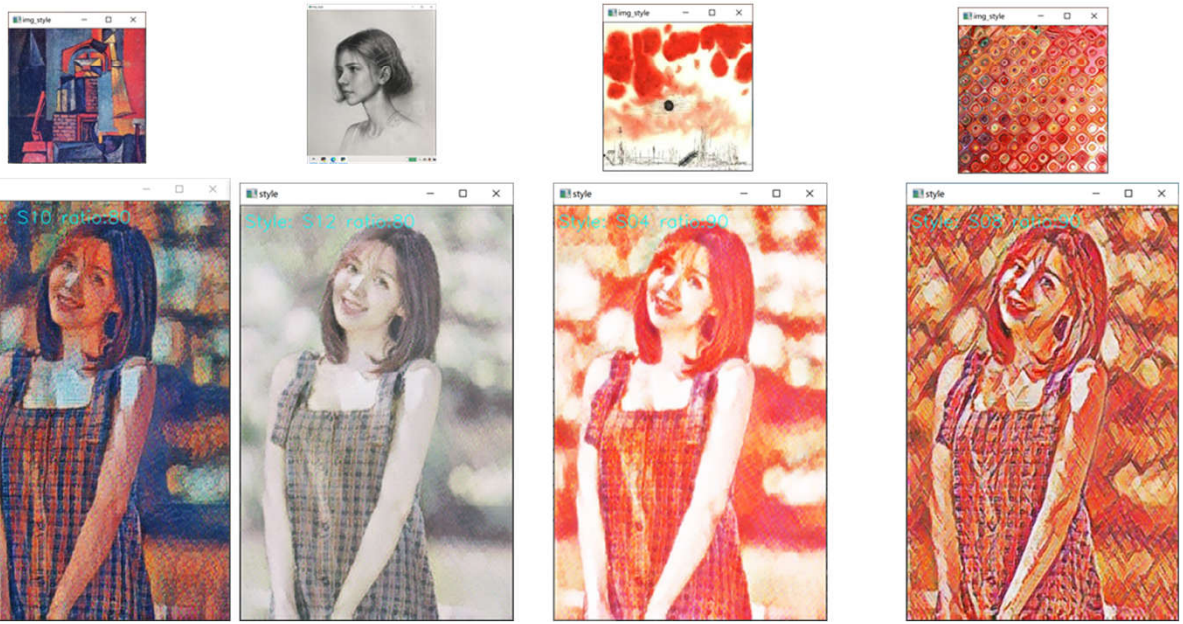
```
flag = 0
while True:
    if flag:
        # 进行风格转换
        print("start processing Style=%s ratio=%d%%"%(str_style,ratio))
        stylized_image = run_style_transform(file_model_transfer,mix_fea, img)
        cv2.putText(stylized_image,'Style: %s ratio:%d'%(str_style,ratio),(5,30),
                    cv2.FONT_HERSHEY_SIMPLEX,0.8,(255,255,0),1,cv2.LINE_AA)
        print("processing end")
        flag =0

    # 显示结果
    cv2.imshow('style', stylized_image)
    # 获取按键
    key=cv2.waitKey(10) & 0xFF

    # 按s 切换风格
    if key == ord('s'):
        str_style = input('Enter your style name:')
        file_style = os.path.join(path_fea_style,str_style+'.npz')
        if not os.path.exists(file_style):
            print('Can not find ' + file_style)
        else:
            img_style = cv2.imread(os.path.join(path_img_style,str_style+'.jpg'))
            cv2.imshow("img_style",img_style)

            style_fea = np.load(file_style)['fea']
            mix_fea = ratio*0.01*style_fea +(1-ratio*0.01)*content_fea
            flag = 1
        key =0
    elif key == ord('r'):
        str_ratio = input('Enter your style ratio:')
        ratio = float(str_ratio)
        mix_fea = ratio*0.01*style_fea +(1-ratio*0.01)*content_fea
        flag = 1
    elif key == ord('q'):
        break
```

原始图像



风格图像

风格转换
后图像