北京大学
PEKING UNIVERSITY

The Legend of a Picture

报告人：李畅 李思哲 王珮衡

01

目录

图像风格化——架构

·论文summary

·参数调整

·不同风格的对比

图像风格化——应用

·服饰纹理设计

·图像风格统一化

图像风格化——拓展

·视频风格迁移

·UI界面的设计

·实时风格迁移

论文 summary

以下是Peggy的笔记 可以应用在衔接中

1、首先论文提及了一种无参的转化方式

2、其次提到许多很厉害的人用各种各样方法做风格转换

但是，都有limitation：用了低水平的图像特征处理纹理转换

提出要求即小**目标：能够获得content的语义信息 用style的方式表示出来**

故而

思路：用已经训练完成的网络分别独立处理content与style

将style transfer任务转化为一个最优解问题 即用一个神经网络最优化loss的问题

对于图片的处理大致分为两部分：

1）content：find image representation

- 这一部分在处理手写数据集等里面已经有很好的研究与方法

2）style

论文 summary

Content Representation:

一层N_l个大小为M_l的卷积 生成一个$N_l * M_l$的响应

计算新生成图片与原图之间的Loss

$$\mathcal{L}oss(\vec{p},\vec{x},\vec{l}) = \frac{1}{2}\sum_{i,j}\left(F_{ij}^{l} - P_{ij}^{l}\right)^2$$

$$\frac{\partial \mathcal{L}_{content}}{\partial F_{ij}^{l}} = \begin{cases} \left(F^{l} - P^{l}\right)_{ij} & \text{if} F_{ij}^{l} > 0 \\ 0 & \text{if} F_{ij}^{l} < 0 \end{cases}$$

代码中即为mse_loss()

高阶的layer能捕获更高级的content特征

选择在conv_4处计算得content_loss

论文 summary

Style Representation:

Consisting of the correlations between the different filter responses

一层N_l个大小为N_l的卷积 生成一个N_l*N_l的响应

生成Gram Matrix $G^l \in R^{N_l \times N_l}$ 记录内积

$$G^l_{ij} = \Sigma F^l_{ik} F^l_{jk}$$

用白噪音梯度下降 minimize mse_loss

各层权重论文中均取1/5 暂不作研究

论文 summary

$$\mathcal{L}_{total} = \alpha\, \mathcal{L}_{content}(\vec{p}, \vec{x}) + \beta\, \mathcal{L}_{\text{style}}(\vec{a}, \vec{x})$$

α， β 为各自权重

```
def get_input_optimizer(input_img):
    # 以梯度为参数
    optimizer = optim.LBFGS([input_img.requires_grad_()])
    return optimizer
```

采用L-BFGS算法找到最优解

L-BFGS 在有限内存中进行BFGS算法

牛顿迭代 -> 泰勒展开Hessi矩阵 -> 通过迭代逼近矩阵

初始化时需要：

将content与style转化为同样大小图片 在load_image()函数中处理

注： Image.LANCZOS 多相位插值放缩

论文 summary

整个过程中

论文给出的结论是 content 与 style 是完全**独立**的

其中，整个方法中对于最终风格迁移效果可能产生影响的可调参部分为：

1）α/β = content weight / style weight

2）获取 content loss 层数

3）用以初始化 output 的图片（content / style / white noise）

论文 summary

整个过程中

论文给出的结论是 content 与 style 是完全**独立**的

其中，整个方法中对于最终风格转换效果可能产生影响的可调参部分为：

1）α/β = content weight / style weight

2）获取 content loss 层数

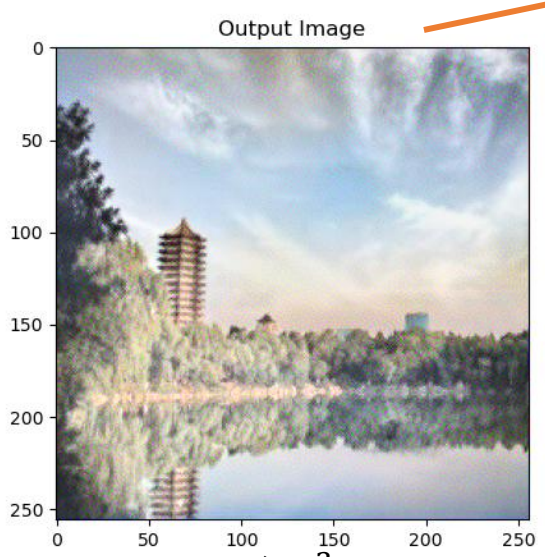
3）用以初始化 output 的图片（content / style / white noise）

α/β = content weight / style weight

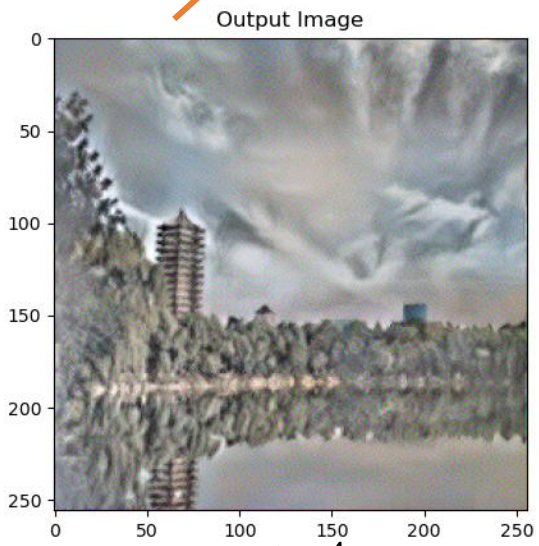
Style权重偏小
色彩保留较明显

content

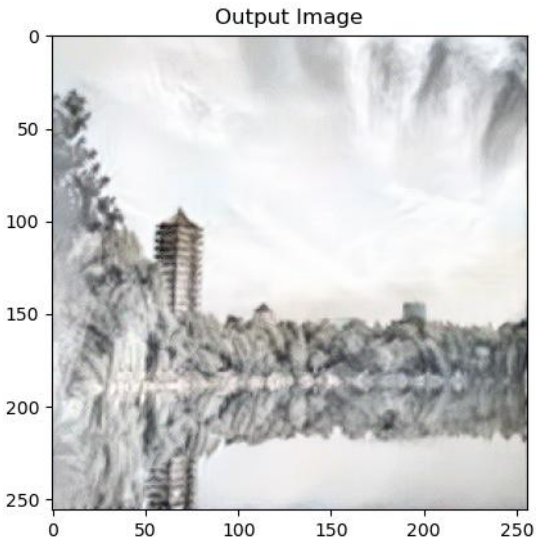
style



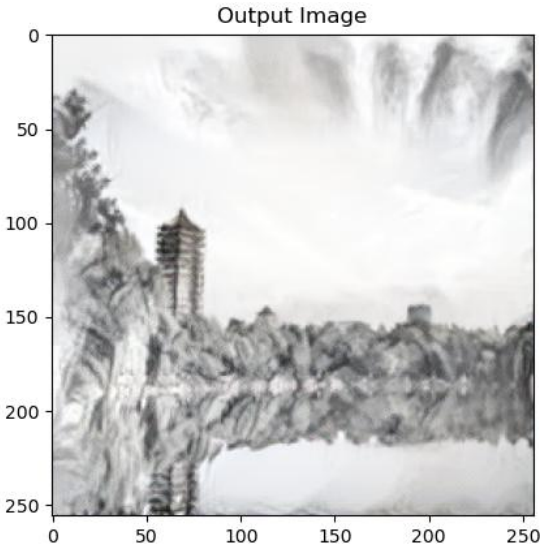
$1/10^3$



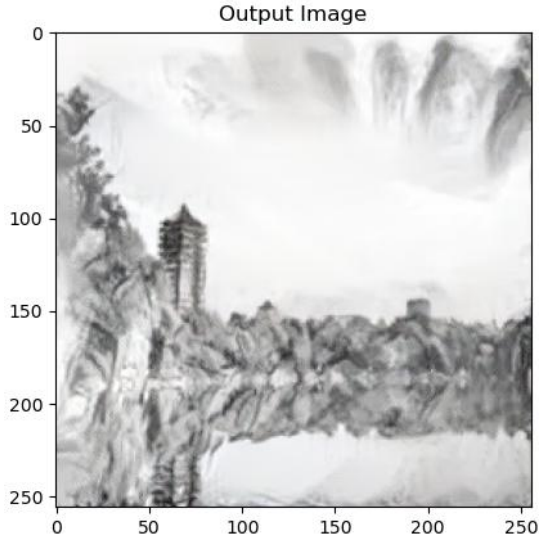
$1/10^4$



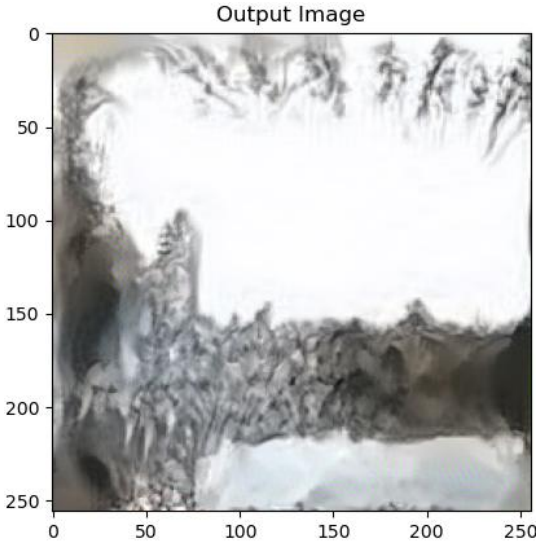
$1/10^5$



$1/10^6$



$1/10^7$

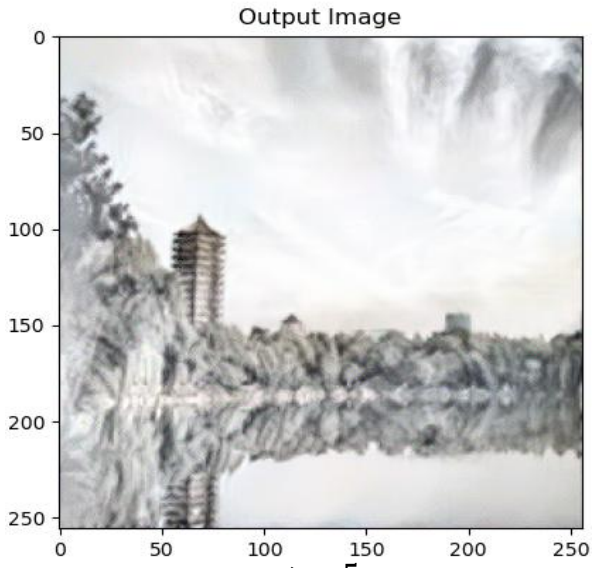


$1/10^8$

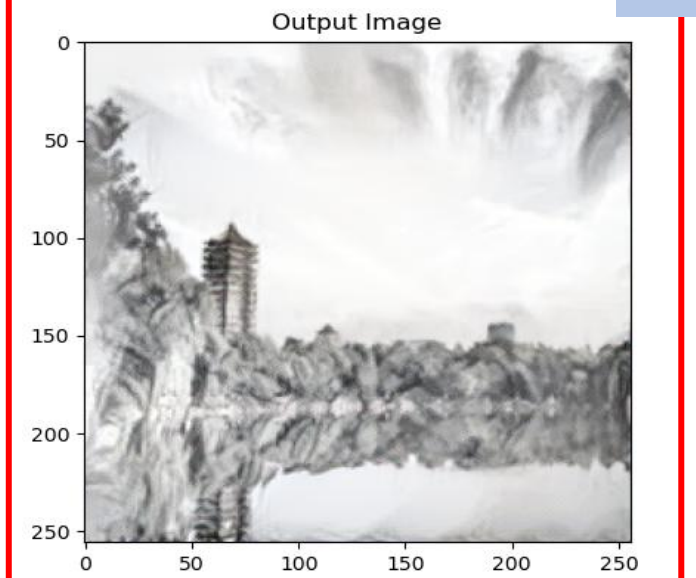
α/β = content weight / style weight

content

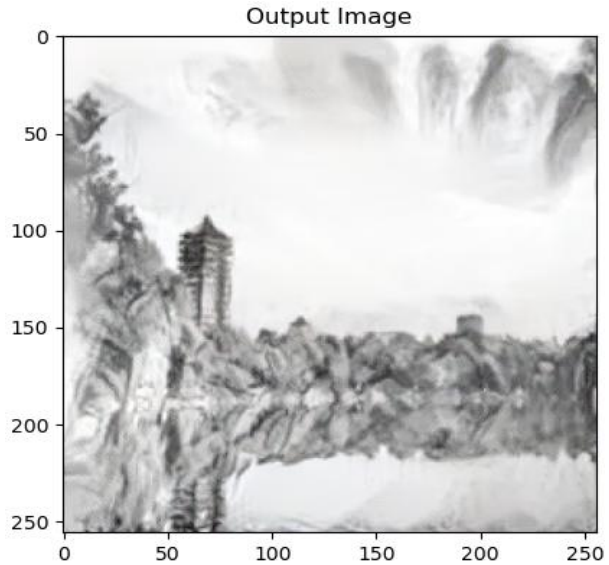
style



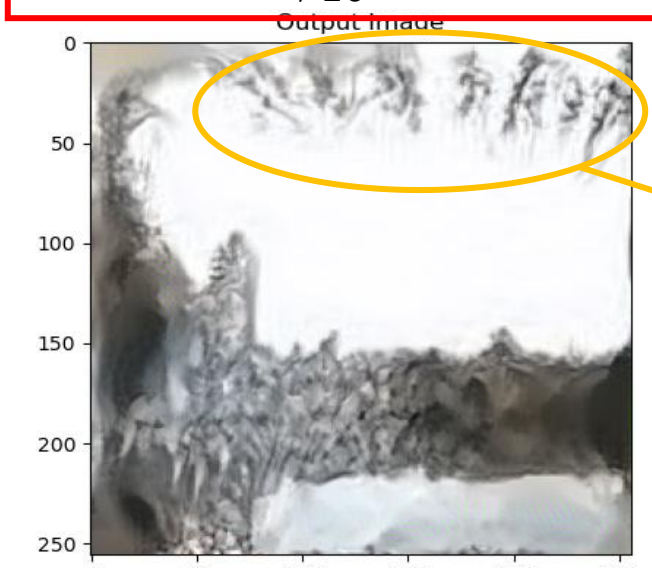
$1/10^5$



$1/10^6$



$1/10^7$



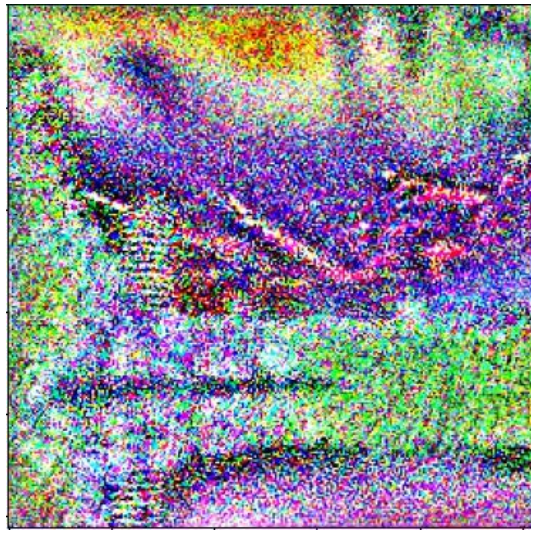
$1/10^8$

Style过大
保留内容特征

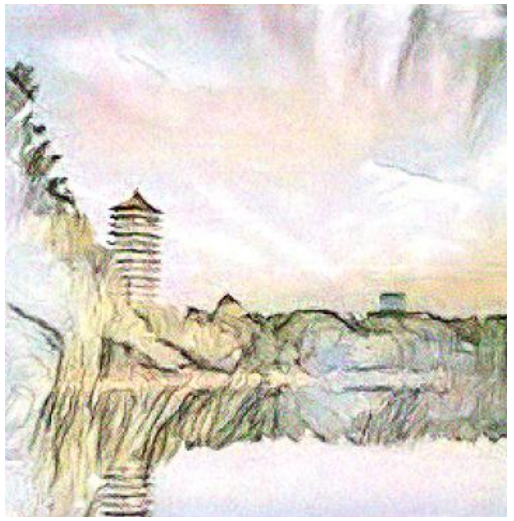
P 1.1

α/β = content weight / style weight

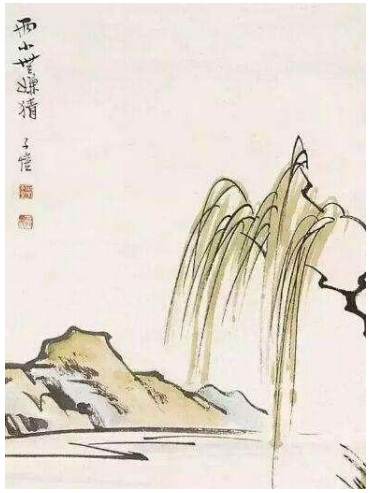
content



$1/10^3$



$1/10^4$



style

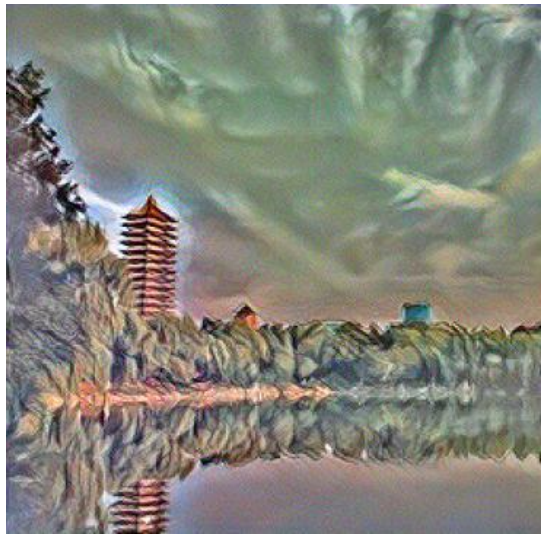
$1/10^5$

$1/10^6$

P 1.1

α/β = content weight / style weight

content



$1/10^3$

$1/10^4$

style

1/50000

$1/10^5$

$1/10^6$

P 1.1

α/β = content weight / style weight

content

style

1/5000

1/50000

$1/10^4$

$1/10^5$

α/β = content weight / style weight 总结

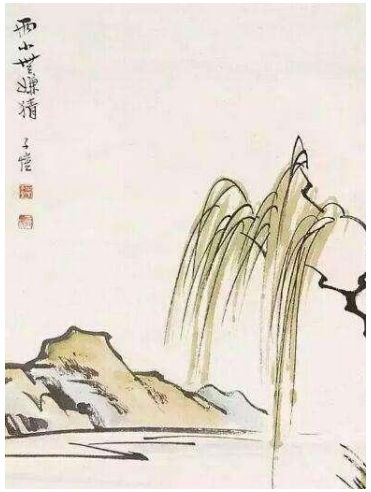
a. style风格笔触越小 style最终所占权重应相应调大，达到较为理想的风格保留效果

b. style 过大易导致 content 内容丢失严重 生成图片中混入较多 style 中特征

分类：水墨

特征：笔触较小 略微块状

推荐α/β参数： $1/10^6$



分类：工笔水彩

特征：笔触简练 线条特征

推荐α/β参数： $1/10^5$

分类：水粉、油画

特征：笔触较大 连贯条状

推荐α/β参数：$1/10^5 \sim 1/10^4$

论文 summary

整个过程中

论文给出的结论是 content 与 style 是完全**独立**的

其中，整个方法中对于最终风格转换效果可能产生影响的可调参部分为：

1）α/β = content weight / style weight

2）获取 content loss 层数

3）用以初始化 output 的图片（content / style / white noise）

content 获取特征层数

content

相同α/β权重下，改变获得 loss 的 content 卷积层数

可以验证论文中，从更深卷积层中获取更多的语义信息，使得content图中信息得以保留

可以发现conv_3,conv_4,conv_5处计算效果较好

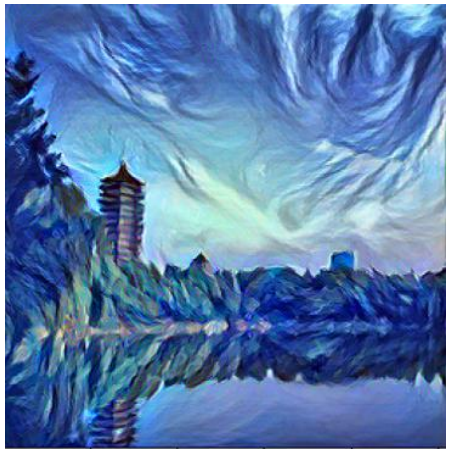
style

conv_1

conv_2

conv_3

conv_4



conv_5

采用先前得到的$\alpha/\beta = 10^5$以及conv4/conv5

content

conv_4

conv_5

style

论文 summary

整个过程中

论文给出的结论是 content 与 style 是完全**独立**的

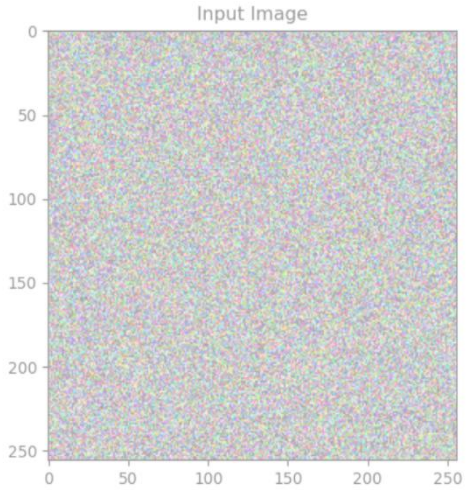
其中，整个方法中对于最终风格转换效果可能产生影响的可调参部分为：

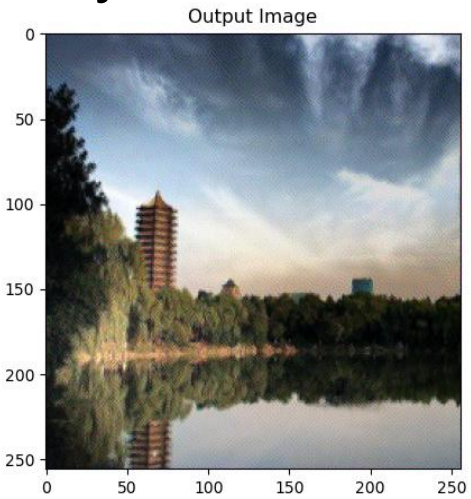
1）α/β = content weight / style weight

2）content获取特征层数

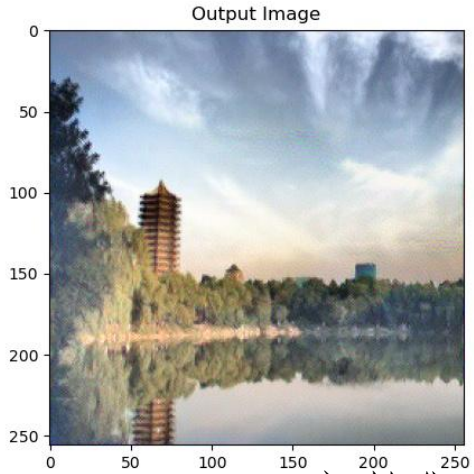
3）用以初始化 output 的图片（content / style / white noise）

用以初始化 output 的图片（content / style / white noise）

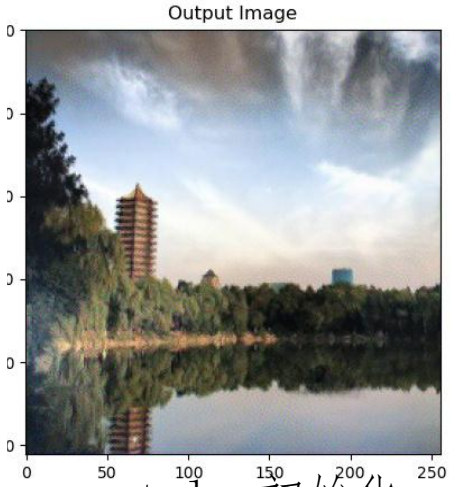




white noise 初始化



content 初始化



style 初始化

论文 summary

整个过程中

论文给出的结论是 content 与 style 是完全**独立**的

其中，整个方法中对于最终风格转换效果可能产生影响的可调参部分为：

1）α/β = content weight / style weight

2）content获取特征层数

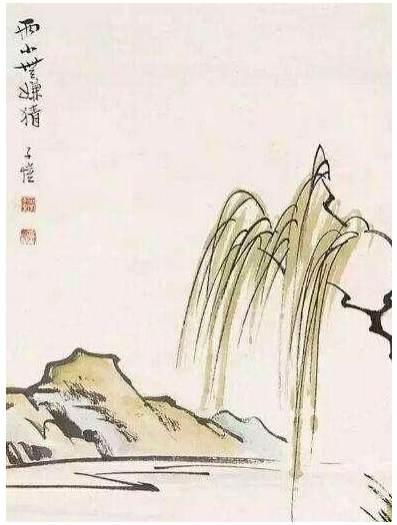
3）用以初始化 output 的图片（content / style / white noise）

Answer for Project One

our output

图片由神经网络生成后略调整饱和度得到

style



02

目录

图像风格化——架构

·论文summary

·参数调整

·不同风格的对比

图像风格化——应用

·服饰纹理设计

·图像风格统一化

图像风格化——拓展

·视频风格迁移

·UI界面的设计

·实时风格迁移

P2

应用：服饰纹理设计

手稿 ＋ 材质/纹样图片 ➡ 该类材质/纹样的成品图

方法：将目标纹样图片作为style 手稿作为content导入



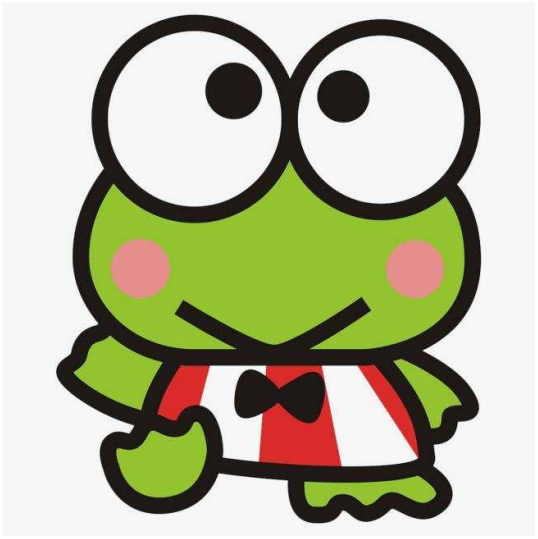



交互：手工选择需要转换的手稿区域、纹样区域 进行转换
改进：先识别出画面主体即手稿服装所在位置，仅对该位置进行风格迁移操作

图像风格统一化

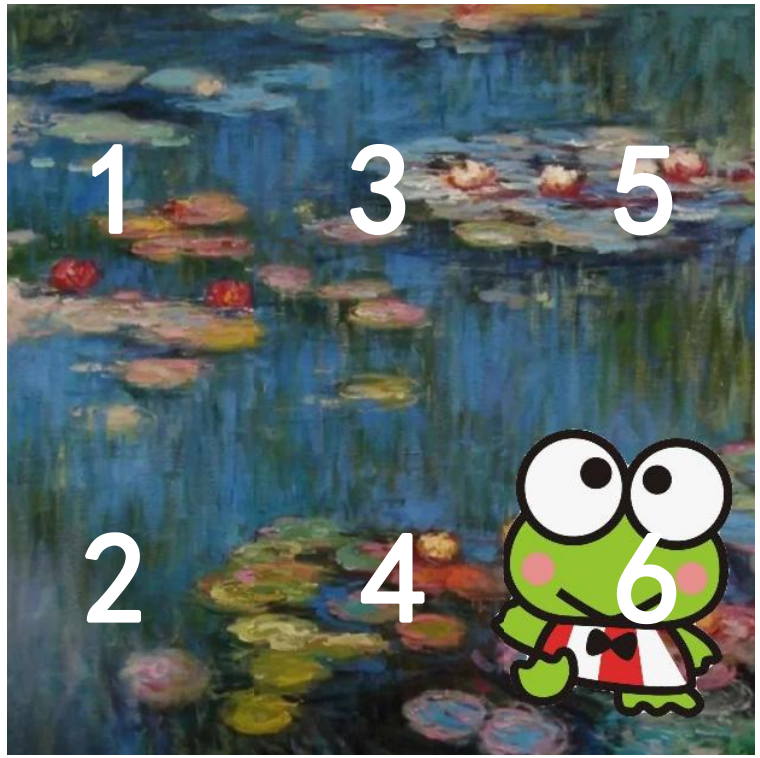
Content:

《池塘·睡莲》——莫奈



Style: ?

图像风格统一化



style_loss	1	2	3	4	5	6
1	\	103	101	104	102	106
2	103	\	98	98	102	102
3	101	98	\	100	100	104
4	104	98	100	\	101	103
5	102	102	100	101	\	105
6	106	102	104	103	105	\
total_loss	516	503	503	506	510	520

$$G_{ij} = \sum_k F_{ik} \cdot F_{jk} = F \cdot F^T; \qquad L_{Style} = \sum \frac{1}{w^2 \cdot h^2} \left(G_{ij}^1 - G_{ij}^2 \right)^2$$

图像风格统一化

方案一：

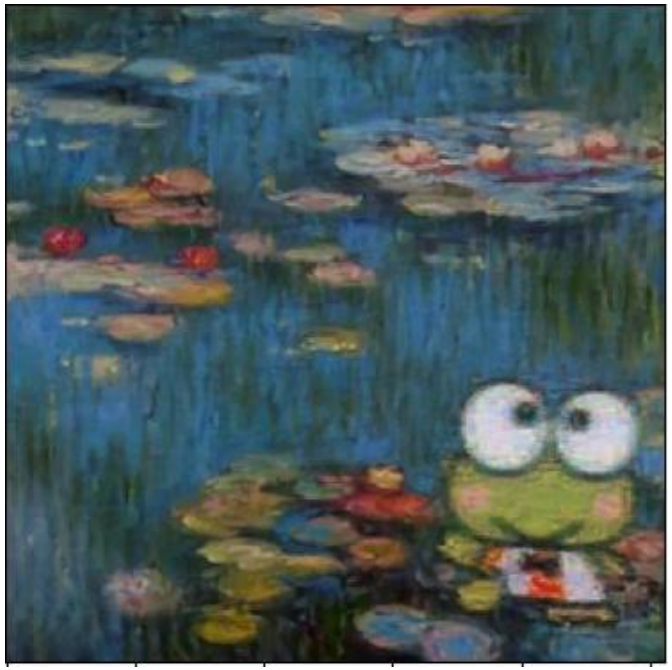
直接用图2作为Style图片，对内容图片进行风格化。

我们需要找到一张更合适的图片来作为Style！！

方案二：

使用图2替代图6，利用**PIL的paste函数**，重新拼合成与原图尺寸相同的图片作为输入的style图片。

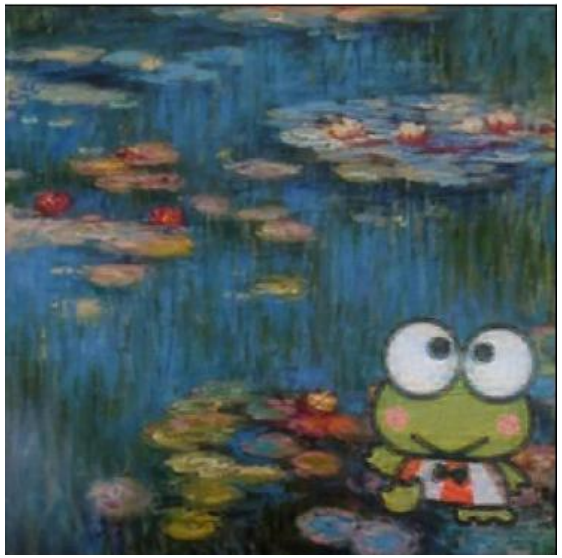
Style输入



Output

1.参数问题

实验发现，随style_loss的权重增大，风格统一化的效果逐渐变好。记style_loss/content_loss = α / β。



α / β = 10^7

α / β = 10^9

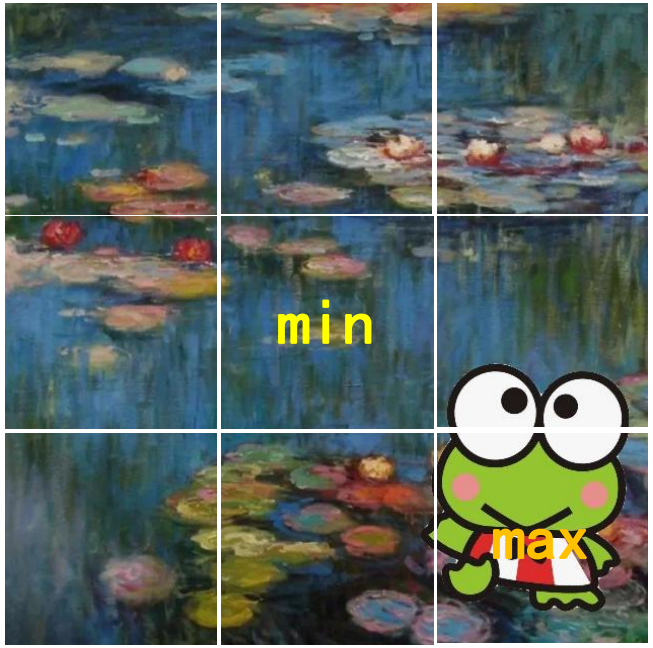
α / β = 10^11

图像风格统一化

2. 图片切割以及损失函数

图片切割是否是越细越好？损失函数是否合理？

2*2



3*3

4*4

图像风格统一化

3. 如何寻找更好的风格输入

Content

Style

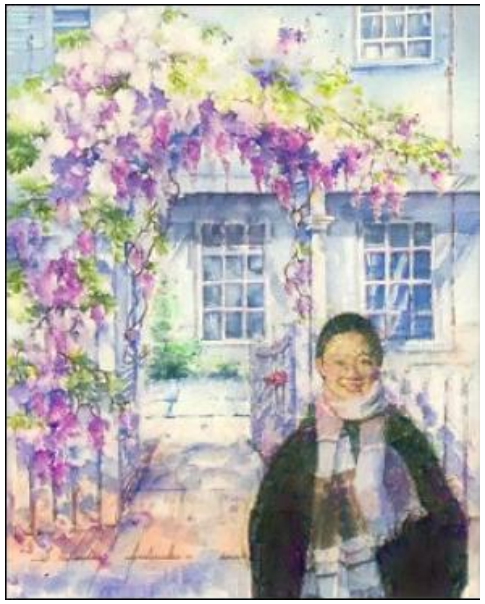
Output

图像风格统一化

3. 如何寻找更好的风格输入

Content

Style



Output

目录

图像风格化——架构

·论文summary

·参数调整

·不同风格的对比

图像风格化——应用

·服饰纹理设计

·图像风格统一化

图像风格化——拓展

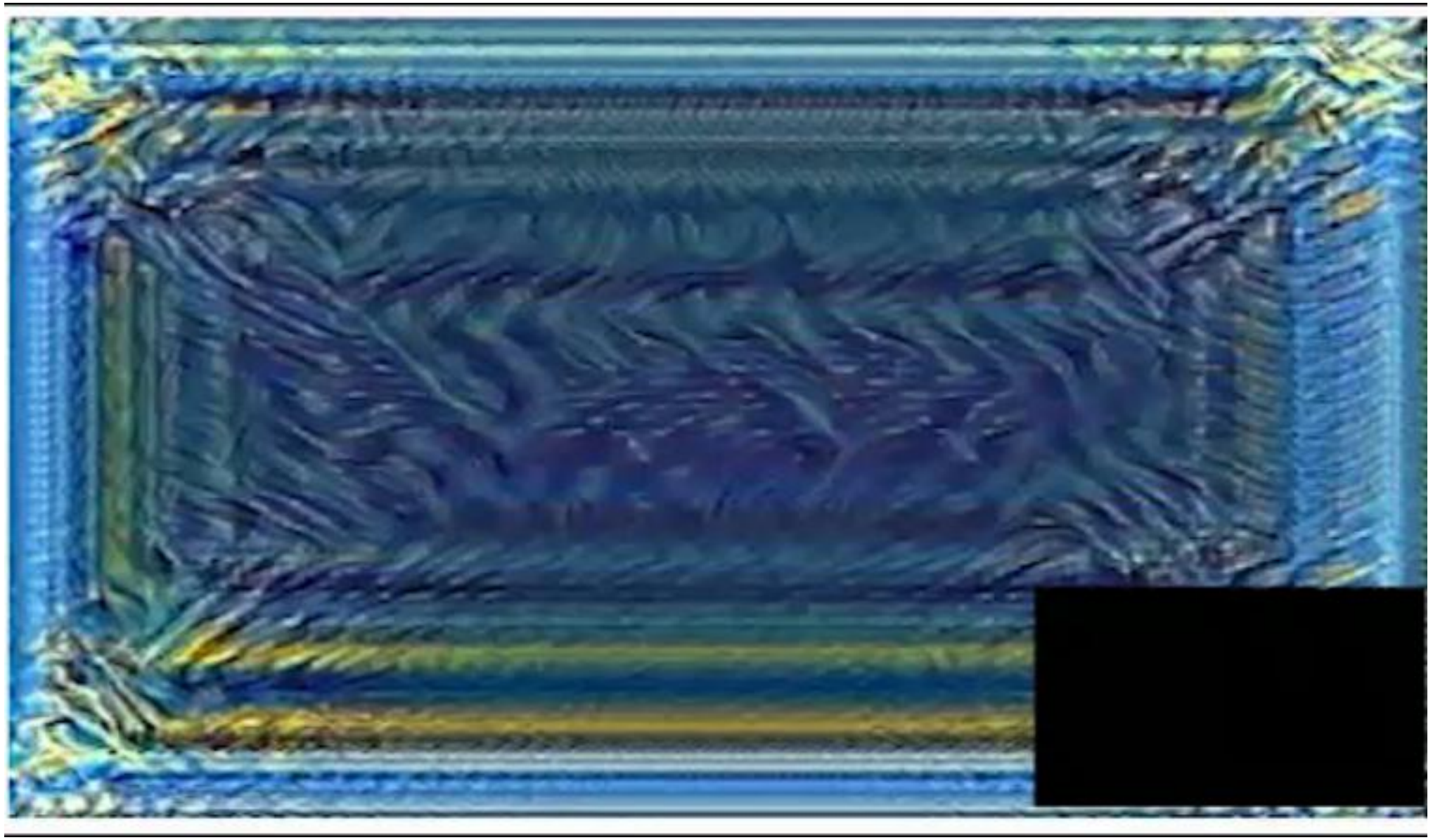
·视频风格迁移

·UI界面的设计

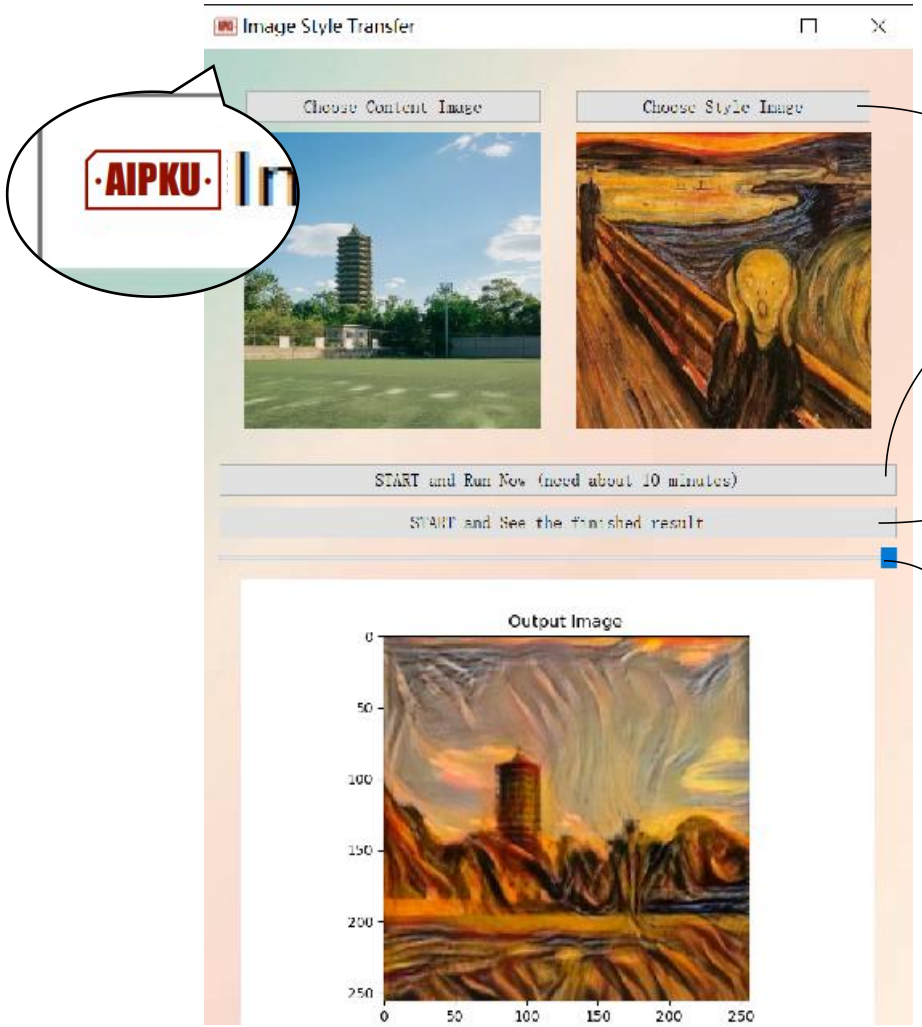
·实时风格迁移

P2

视频风格化处理



人机交互界面——PyQt5图形库



可以从本地选择文件

Start and Run now（现场运行）

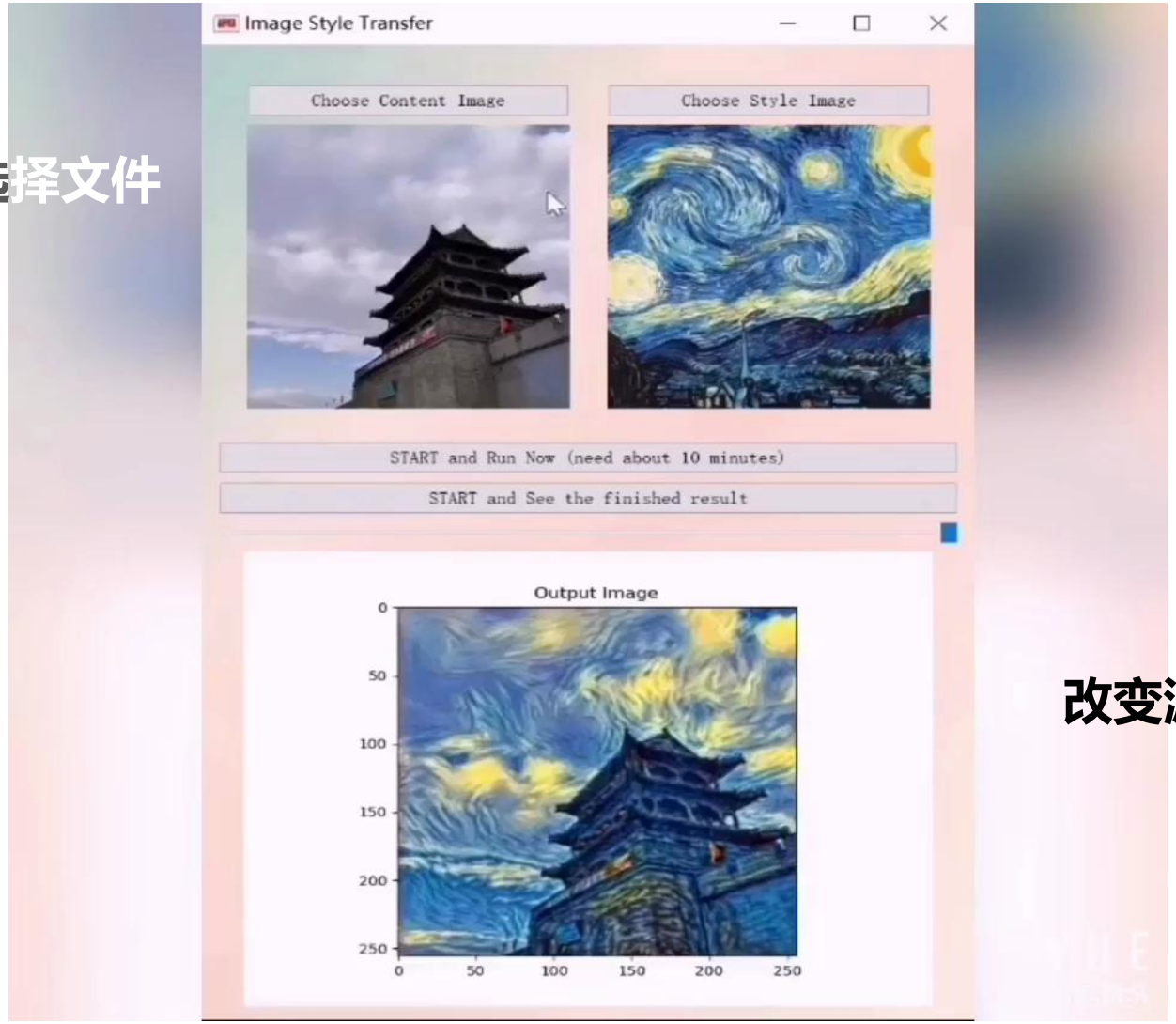
See the Result（查看运行后的结果）

使用滑动条调节滤镜浓度

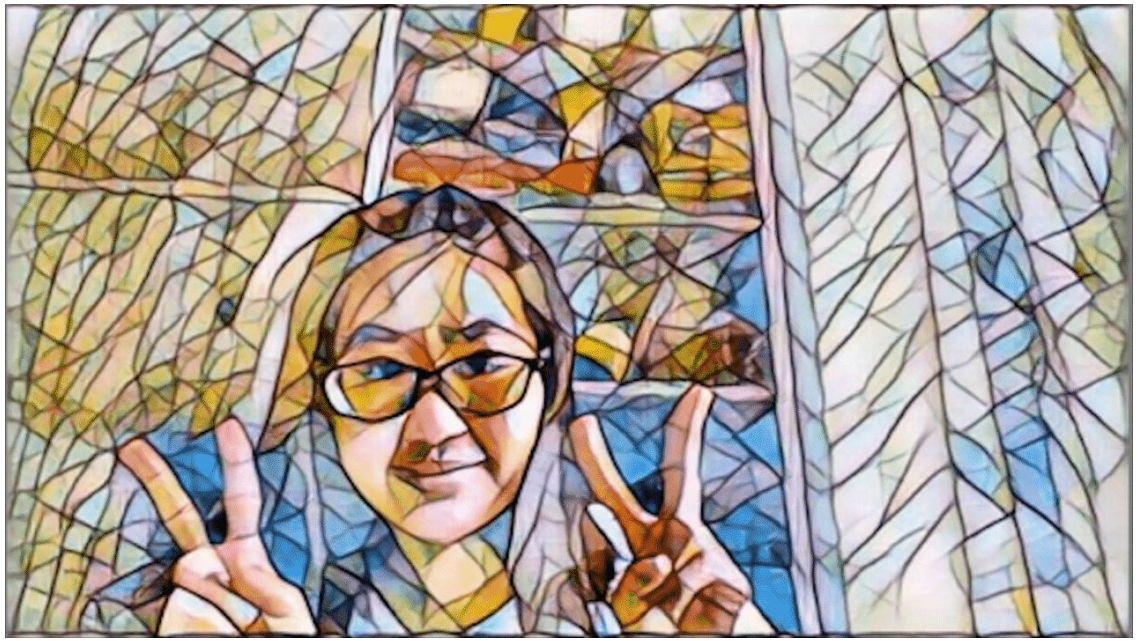
P2

人机交互界面——展示

从电脑选择文件



改变滤镜浓度



实时风格转换

通过调用训练好的神经网络

再逐帧处理网络摄像头视频流信息实现

功能开发者：李畅

整体思路

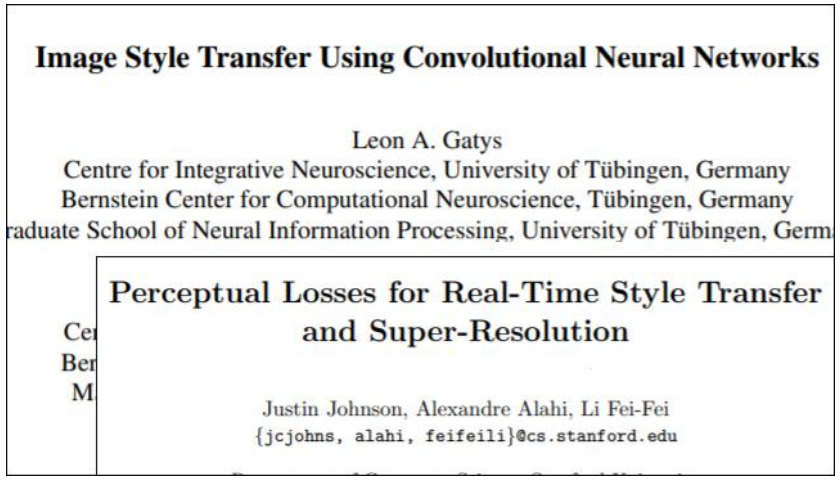
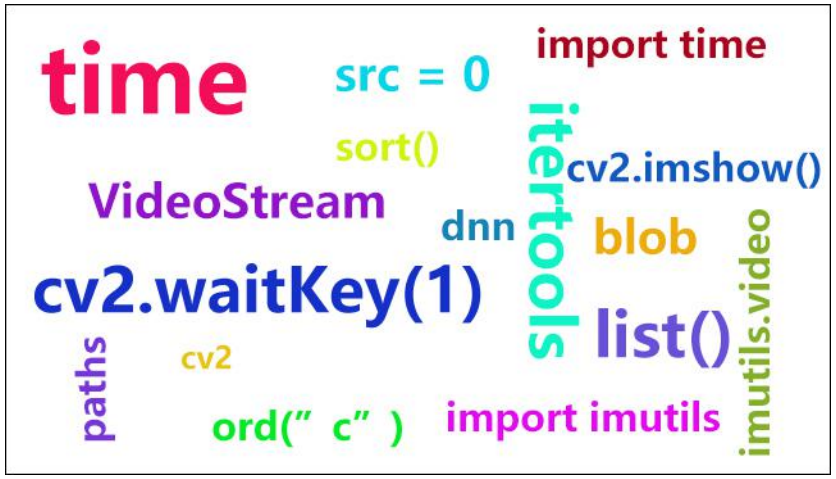


Image Style Transfer Using Convolutional Neural Networks

Leon A. Gatys
Centre for Integrative Neuroscience, University of Tübingen, Germany
Bernstein Center for Computational Neuroscience, Tübingen, Germany
raduate School of Neural Information Processing, University of Tübingen, Germ

Perceptual Losses for Real-Time Style Transfer and Super-Resolution

Justin Johnson, Alexandre Alahi, Li Fei-Fei
{jcjohns, alahi, feifeili}@cs.stanford.edu



网络的选取

采用Johnson 2017年提出的模型

网络的调用

视频流引入 逐帧处理

调用模型 切换

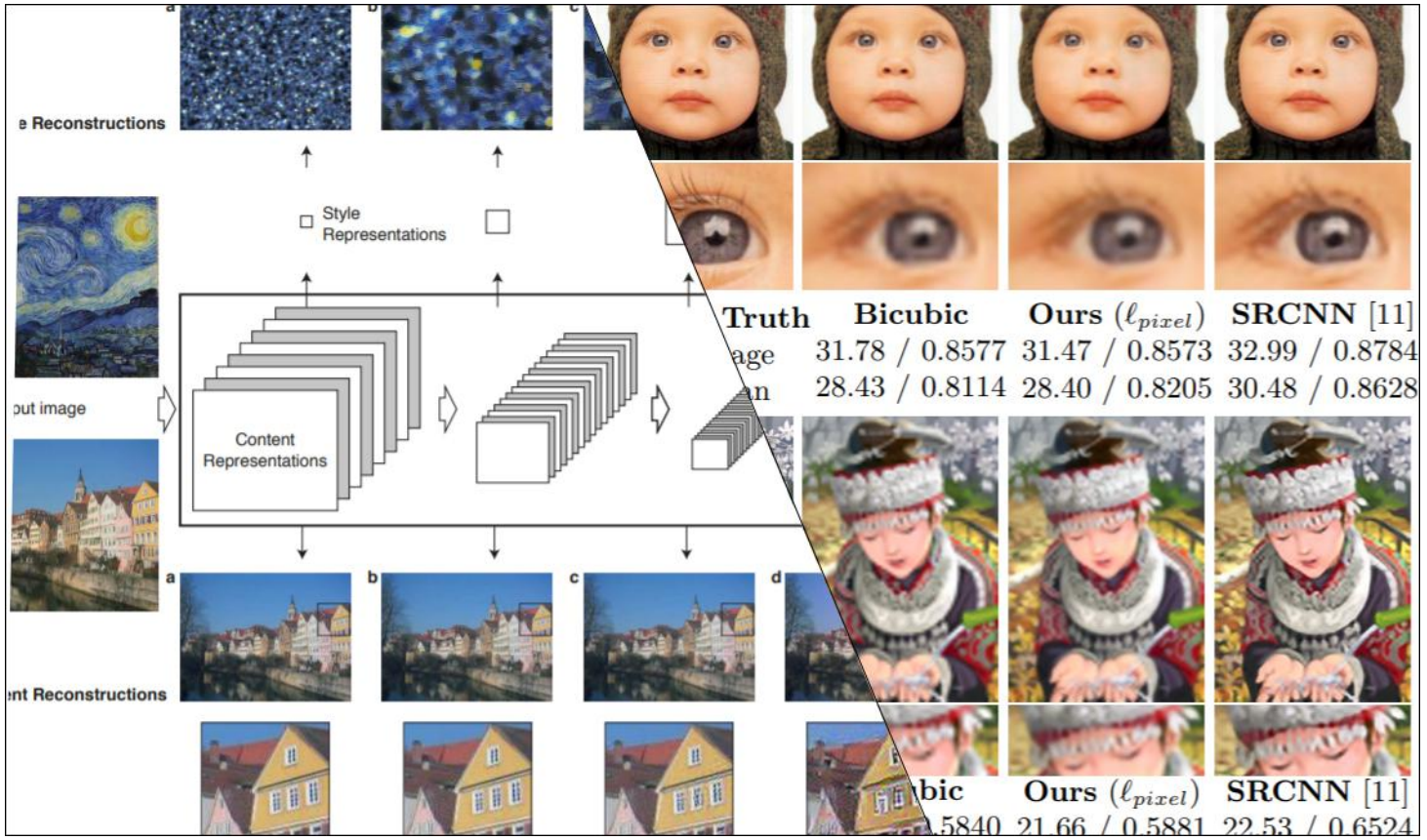
网络的选取

2016年Gatys:
A Neural Algorithm of Artistic Style

最初的神经风格迁移算法

2017年Johnson:
Perceptual Losses for Real-Time Style Transfer and Super-Resolution

大大提高了处理速度



实时风格转换

```
demo_final.py
import cv2
import argparse
import imutils
from imutils import paths
from imutils.video import VideoStream
import itertools
import time
```

```
model = list(zip(range(0, len(find_models
iterator_m = itertools.cycle(model)
(modelID, find_model) = next(iterator_m)
net = cv2.dnn.readNetFromTorch(find_model
window_x = VideoStream(src = 0).start()
time.sleep(1.0)
while True:
    frame = window_x.read()
    frame = imutils.resize(frame, width =
```

视频流引入

导入imutils库

用VideoStream.start()调用摄像头

逐帧处理

resize

resize成需要的大小

同时获取长宽信息

减均值

用blob函数减均值

对抗亮度变化

投入网络

用setInput函数

将减均值结果投入网络

reshape

对output tensor进行reshape

加回均值

重新加回减掉的均值

展示

imshow函数展示结果

模型的调用

```
rt time
and_x = argparse.ArgumentParser()
and_x.add_argument("-m", "--models", required
and_final = vars(command_x.parse_args())
_models = paths.list_files(command_final["mod
_models = sorted(list(find_models))
l = list(zip(range(0, len(find_models)), (fin
ator_m = itertools.cycle(model)
elID, find_model) = next(iterator_m)
```

```
model = list(zip(range(0, len(find_models
iterator_m = itertools.cycle(model)
(modelID, find_model) = next(iterator_m)
net = cv2.dnn.readNetFromTorch(find_model
window_x = VideoStream(src = 0).start()
time.sleep(1.0)
while True:
    frame = window_x.read()
    frame = imutils.resize(frame, width =
```

将模型的path做成列表

利用imutils包中的list_files函数

可以将所有的模型地址导成列表

再从地址中读取网络

利用cv2.dnn.readNetFromTorch(modelpath)

调用存储的网络

模型的切换

内部机制：迭代器

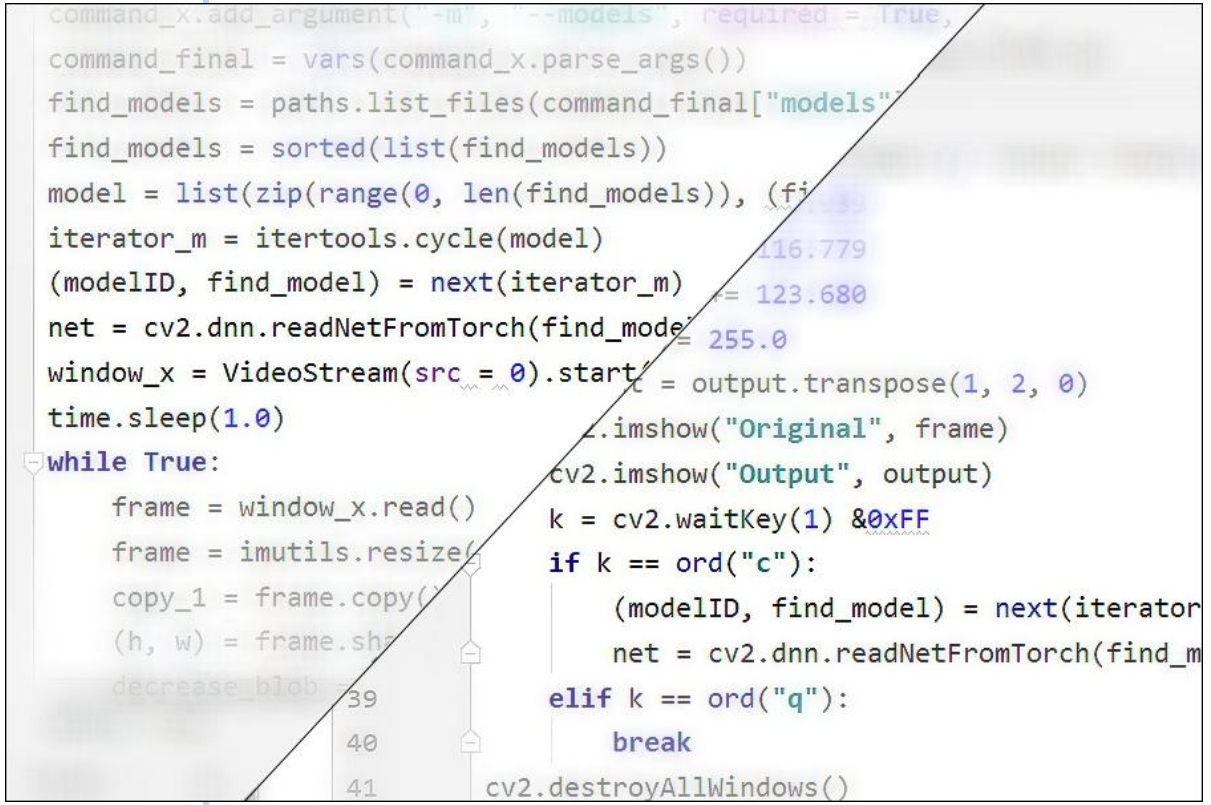
导入itertools包

迭代预先处理成列表形式存储的模型信息

外部交互：键盘键入

先用cv2.waitKey函数定义得到的信息

不同键入来进行不同操作



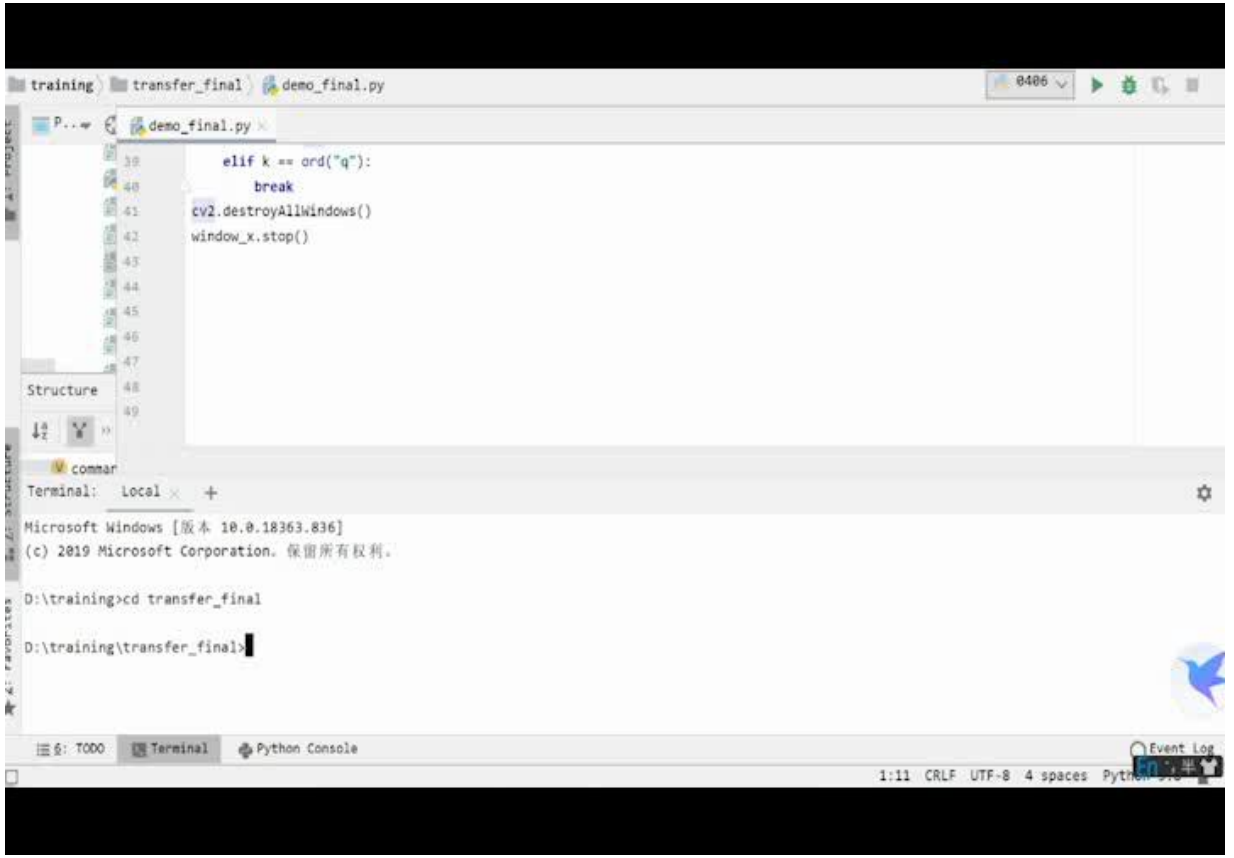
具体操作

具体操作

1.命令行调用

2.全屏

3.键入'c'或'q'



北京大学
PEKING UNIVERSITY

谢谢大家

图片、视频均来自网络，使用的人像照片均已经本人同意

待改进

1、论文：模型阶段提到，将max-pooling改为average-pooling效果更佳

2、Q：第二个代码中

Why normalize?

```
def gram_matrix(input):
    a, b, c, d = input.size()  # a = batch size(=1)
    # b = number of feature maps
    # (c,d) = dimensions of a f. map (N = c*d)

    features = input.view(a * b, c * d)  # resise F_XL into \hat F_XL

    G = torch.mm(features, features.t())  # compute the gram product

    # we 'normalize' the values of the gram matrix
    # by dividing by the number of element in each feature maps.
    return G.div(a * b * c * d)
```

3、要再确认一下代码中方法和实际论文方法的对应

4、试试可视化style loss在最后报告中

5、为了达到input的style和content一致 运用了Image.ANTIALIAS和Image.LANCZOS 多相位插值放缩会导致一定程度数据丢失 使得最后生成图片较为模糊

6、使用的已训练网络 （参数不同？）

7、关于L-BFGS

8、降噪