风格迁移（style transfer）

论文 ："A Neural Algorithm of Artistic Style"

汇报人：蔡浩

CONTENTS

汇报流程

PART 01

风格迁移是什么意思

PART 02

如何实现风格迁移的

PART 03

Q&A

1

PART 01

风格迁移展示

PART 01

PART 01

PART 01

PART 01

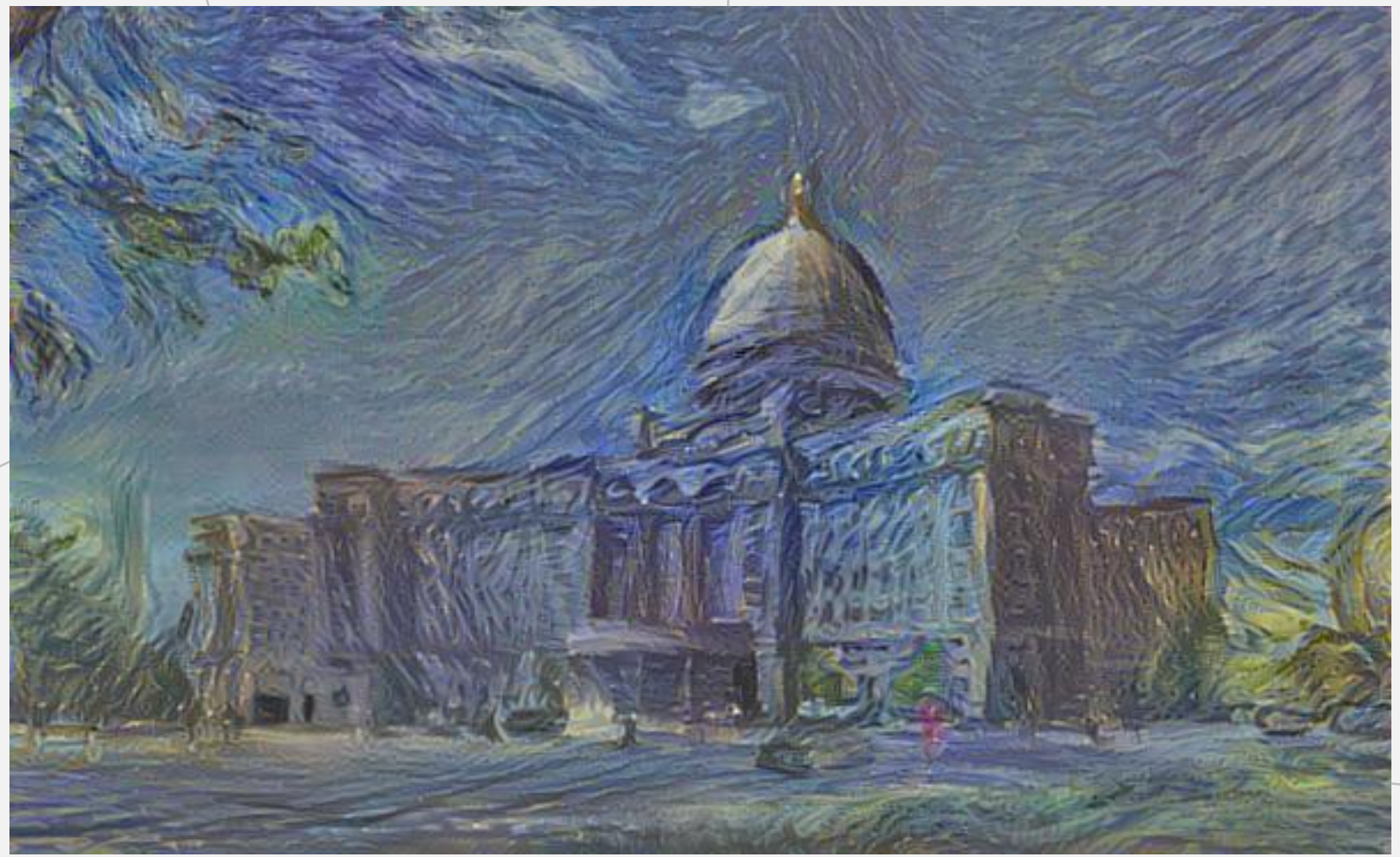
- 梵高的星空

PART 01

- 梵高的主楼

PART 01

- 梵高的宋健



1

PART 01

风格迁移解释

PART 01

风格迁移就是在两张图片中，取其中一张图片的风格与另一张图片的内容生成一张新的图片。

PART 01

图片的风格（style）和内容（content）可以说都是图片的特征（feature），到底怎样才能将他们抽取出来呢？

PART 02

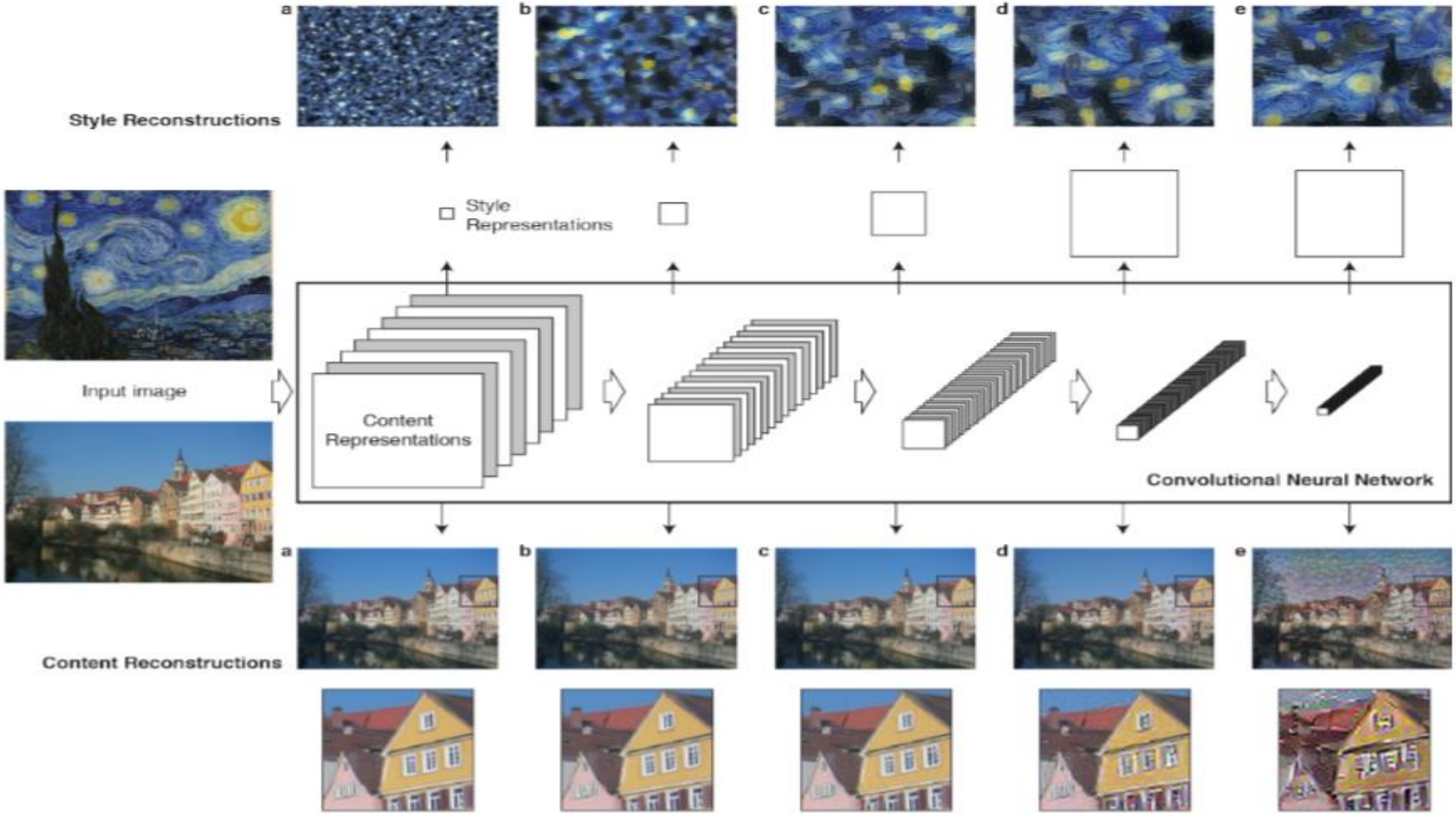
抽取图片特征的方法

PART 02

没错，就是使用ＣＮＮ

PART 02

CNN视角下的图片到底是什么样子呢


a
b
c
d
e
Style Reconstructions
Style
Representations
Input image
Content
Representations
Convolutional Neural Network
a
b
c
d
e
Content Reconstructions

PART 02

可以发现，对于图片的风格（style）和内容（content）。

1． CNN底层提取的图片特征更接近content

2． CNN高层提取的图片特征更接近style

PART 02

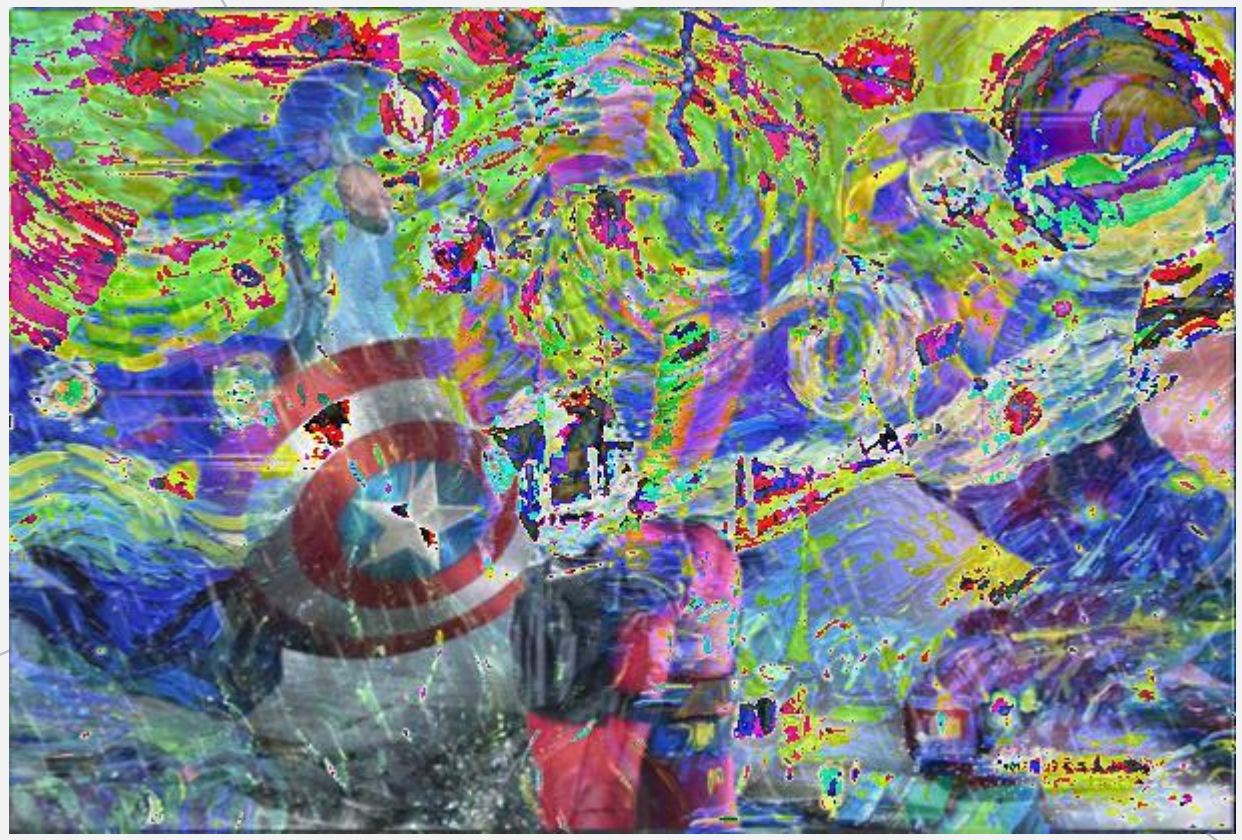
难道风格迁移的图像就是直接将网络顶层和底层的特征映射相加得到的吗？

PART 02

试试把原图相加：

+

PART 02



。。。。。。。。。

沉默，沉默是今晚的康桥

PART 02

具体算法

PART 02

对于内容图片 A，风格图片 B：

１． 给出一张随机图片，即图片的每个像素点都是随机值，图片的shape和内容图片一致

２．在每次迭代学习时，让图片的内容(content)逐渐接近A， 风格(style)逐渐接近B

PART 02

网络结构

由于ＣＮＮ网络的任务是提取图片特征，不需要特别设计一个结构。

所以选用在很多计算机视觉的比赛里取得很好成绩的ＶＧＧ－１９网络

```
# 用到的vgg的网络层， 丢弃了全连接层
layers = (
    'conv1_1', 'relu1_1', 'conv1_2', 'relu1_2', 'pool1',

    'conv2_1', 'relu2_1', 'conv2_2', 'relu2_2', 'pool2',

    'conv3_1', 'relu3_1', 'conv3_2', 'relu3_2', 'conv3_3',
    'relu3_3', 'conv3_4', 'relu3_4', 'pool3',

    'conv4_1', 'relu4_1', 'conv4_2', 'relu4_2', 'conv4_3',
    'relu4_3', 'conv4_4', 'relu4_4', 'pool4',

    'conv5_1', 'relu5_1', 'conv5_2', 'relu5_2', 'conv5_3',
    'relu5_3', 'conv5_4', 'relu5_4'
)
```

16层卷积(convolution)，3层池化(pooling)

不需要使用全连接层(全连接层不起到提取特征的作用)

PART 02

```
INPUT: [224x224x3]        memory:  224*224*3=150K   weights: 0
CONV3-64: [224x224x64]  memory:  224*224*64=3.2M   weights: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64]  memory:  224*224*64=3.2M   weights: (3*3*64)*64 = 36,864
POOL2: [112x112x64]  memory:  112*112*64=800K   weights: 0
CONV3-128: [112x112x128]  memory:  112*112*128=1.6M   weights: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128]  memory:  112*112*128=1.6M   weights: (3*3*128)*128 = 147,456
POOL2: [56x56x128]  memory:  56*56*128=400K   weights: 0
CONV3-256: [56x56x256]  memory:  56*56*256=800K   weights: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256]  memory:  56*56*256=800K   weights: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256]  memory:  56*56*256=800K   weights: (3*3*256)*256 = 589,824
POOL2: [28x28x256]  memory:  28*28*256=200K   weights: 0
CONV3-512: [28x28x512]  memory:  28*28*512=400K   weights: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512]  memory:  28*28*512=400K   weights: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512]  memory:  28*28*512=400K   weights: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512]  memory:  14*14*512=100K   weights: 0
CONV3-512: [14x14x512]  memory:  14*14*512=100K   weights: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512]  memory:  14*14*512=100K   weights: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512]  memory:  14*14*512=100K   weights: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512]  memory:  7*7*512=25K  weights: 0
FC: [1x1x4096]  memory:  4096  weights: 7*7*512*4096 = 102,760,448
FC: [1x1x4096]  memory:  4096  weights: 4096*4096 = 16,777,216
FC: [1x1x1000]  memory:  1000 weights: 4096*1000 = 4,096,000

TOTAL memory: 24M * 4 bytes ~= 93MB / image (only forward! ~*2 for bwd)
TOTAL params: 138M parameters
```

PART 02

使用已经训练好的网络模型，意味着在这个算法中，唯一需要训练的量就是我们随机生成的图片。

PART 02

算法核心

PART 02

loss 函数

content loss：衡量内容图片与所求图片内容之间的差距

content layer: conv4_4，使用vgg-19 中此层的数据来衡量content loss

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

PART 02

loss 函数

content loss：

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

F^L : *内容图片在VGG网络中计算时，在content layer(conv4_4)的值（即：特征映射）*

P^L : *所求图片在VGG网络中计算时，在content layer(conv4_4)的值*

i : 表示在L层的 i 号卷积核

j : 实际上是个坐标，表示 i 号卷积核的特征映射矩阵的某一点。

Ps： 由于内容图片和所求图片的shape是相等的，所以F和P的shape也是相等，所以在实现时，可直接将两个图片在content layer 的值相减

PART 02

loss 函数

style loss：衡量风格图片与所求图片之间风格的差距

style layers: conv1_1, conv2_1, conv3_1, conv4_1, conv5_1

使用vgg-19 的这些层的数据来衡量 style loss

$$G_{ij}^l = \sum_k F_{ik}^l F_{jk}^l.$$

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(G_{ij}^l - A_{ij}^l\right)^2$$

$$\mathcal{L}_{style}(\vec{a}, \vec{x}) = \sum_{l=0}^{L} w_l E_l$$

PART 02

loss 函数

style loss：

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(G_{ij}^l - A_{ij}^l\right)^2$$

$$G_{ij}^l = \sum_k F_{ik}^l F_{jk}^l.$$

$$\mathcal{L}_{style}(\vec{a}, \vec{x}) = \sum_{l=0}^{L} w_l E_l$$

G^L: 即 F 乘以 F 的转置矩阵。得到一个方阵 $N_L * N_L$

M_L: 是图像在经过VGG网络时，在L层的值 高 * 宽

N_L: VGG-19 网络在L层的卷积核数目

PART 02

loss 函数

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

实现代码：

https://github.com/haolang9527/Neural_style/

PART 03

实验中可能改进的方向

Q&A

PART 03

引入图像去噪

http://www.cnblogs.com/ccbb/archive/2011/01/06/1929033.html

PART 03

Q&A

THANKS