

第十二讲 可视化和理解神经网络

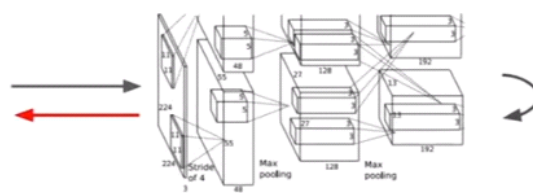
2019年5月24日 10:07

1. DeepDream and Style Transfer

1.1 DeepDream

DeepDream: Amplify existing features

Rather than synthesizing an image to maximize a specific neuron, instead try to **amplify** the neuron activations at some layer in the network



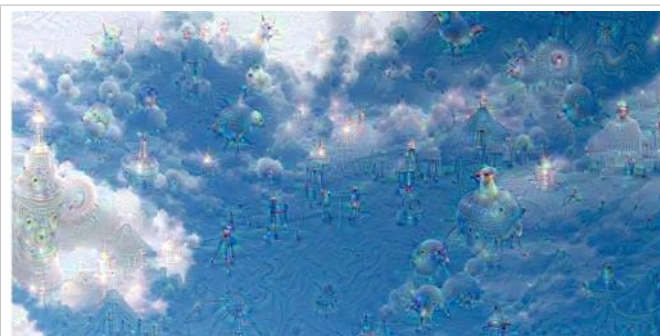
Choose an image and a layer in a CNN; repeat:

1. Forward: compute activations at chosen layer
2. Set gradient of chosen layer *equal to its activation*
3. Backward: Compute gradient on image
4. Update image

图12.1.1 DeepDream

将网络中一些层的梯度设置为它的激活函数所得到的值，反向传播，更新图片，这样试图从网络中发现网络所提取得图像特征是什么。

经过DeepDream的一些图片：



从图中可以看到其实图中所融合的特征展示了训练集所提取的一些图像，如第一张图，原始数据集中有两百多张狗的图片，所以网络所提取的特征与新图融合时里面出现了很多狗的图片。



1.2 Feature Inversion

Feature Inversion

Given a CNN feature vector for an image, find a new image that:

- Matches the given feature vector
- “looks natural” (image prior regularization)

$$\mathbf{x}^* = \underset{\mathbf{x} \in \mathbb{R}^{H \times W \times C}}{\operatorname{argmin}} \quad \ell(\Phi(\mathbf{x}), \Phi_0) + \lambda \mathcal{R}(\mathbf{x})$$

$\xrightarrow{\text{Given feature vector}}$
 $\xrightarrow{\text{Features of new image}}$

$$\ell(\Phi(\mathbf{x}), \Phi_0) = \|\Phi(\mathbf{x}) - \Phi_0\|^2$$

$$\mathcal{R}_{V^\beta}(\mathbf{x}) = \sum_{i,j} \left((x_{i,j+1} - x_{ij})^2 + (x_{i+1,j} - x_{ij})^2 \right)^{\frac{\beta}{2}}$$

$\xrightarrow{\text{Total Variation regularizer (encourages spatial smoothness)}}$

Feature Inversion

Reconstructing from different layers of VGG-16

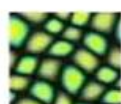


图12.1.2 从VGG16不同层重构的图像

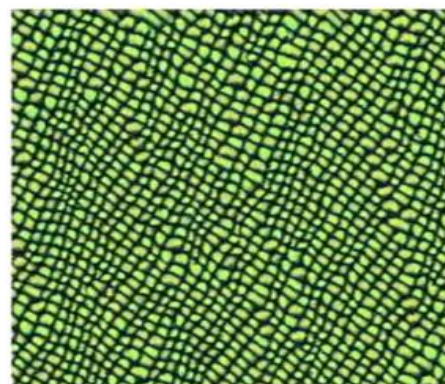
1.3 Texture Synthesis

Texture Synthesis

Given a sample patch of some texture, can we generate a bigger image of the same texture?



Input



Output

纹理合成常见的一种方法是最近邻截取，但是效果并不好，有人提出了另外一种纹理合成的方法如下 格拉姆矩阵：

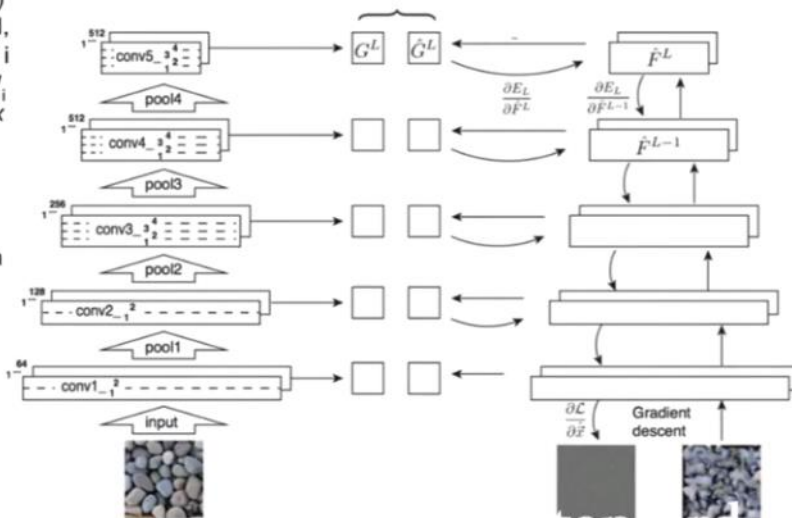
Neural Texture Synthesis

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G_{ij}^l - \hat{G}_{ij}^l)^2 \quad \mathcal{L}(\vec{x}, \hat{\vec{x}}) = \sum_{l=0}^L w_l E_l$$

1. Pretrain a CNN on ImageNet (VGG-19)
2. Run input texture forward through CNN, record activations on every layer; layer i gives feature map of shape $C_i \times H_i \times W_i$
3. At each layer compute the *Gram matrix* giving outer product of features:

$$G_{ij}^l = \sum_k F_{ik}^l F_{jk}^l \text{ (shape } C_l \times C_l \text{)}$$

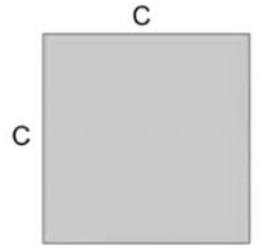
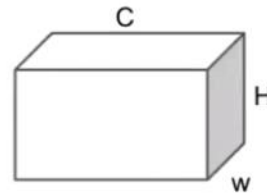
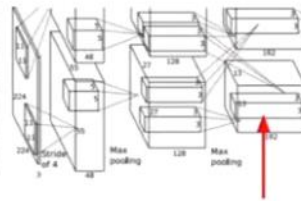
4. Initialize generated image from random noise
5. Pass generated image through CNN, compute Gram matrix on each layer
6. Compute loss: weighted sum of L2 distance between Gram matrices
7. Backprop to get gradient on image
8. Make gradient step on image
9. GOTO 5



Neural Texture Synthesis: Gram Matrix



This image is in the public domain.



Each layer of CNN gives $C \times H \times W$ tensor of features; $H \times W$ grid of C -dimensional vectors

Outer product of two C -dimensional vectors gives $C \times C$ matrix measuring co-occurrence

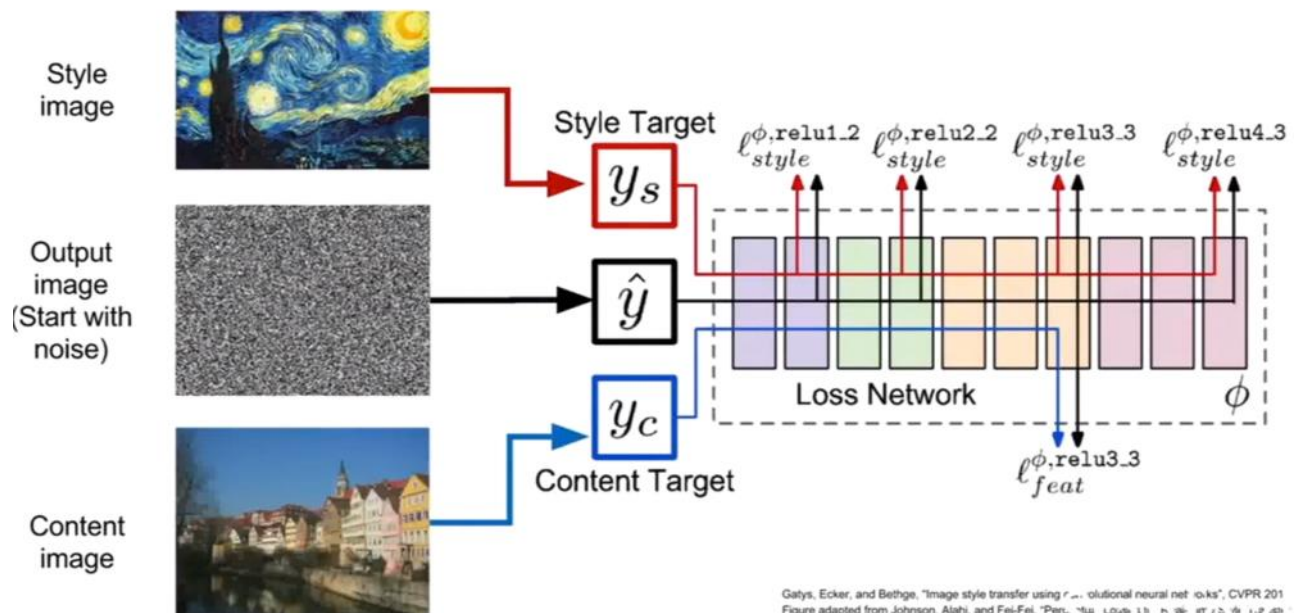
Average over all HW pairs of vectors, giving **Gram matrix** of shape $C \times C$

Efficient to compute; reshape features from

$C \times H \times W$ to $=C \times HW$

then compute $G = FF^T$

1.4 Style Transfer



Gatys, Ecker, and Bethge, "Image style transfer using convolutional neural networks", CVPR 2015
Figure adapted from Johnson, Alahi, and Fei-Fei, "Perceptual Loss for Deep Style Transfer", ECCV 2016

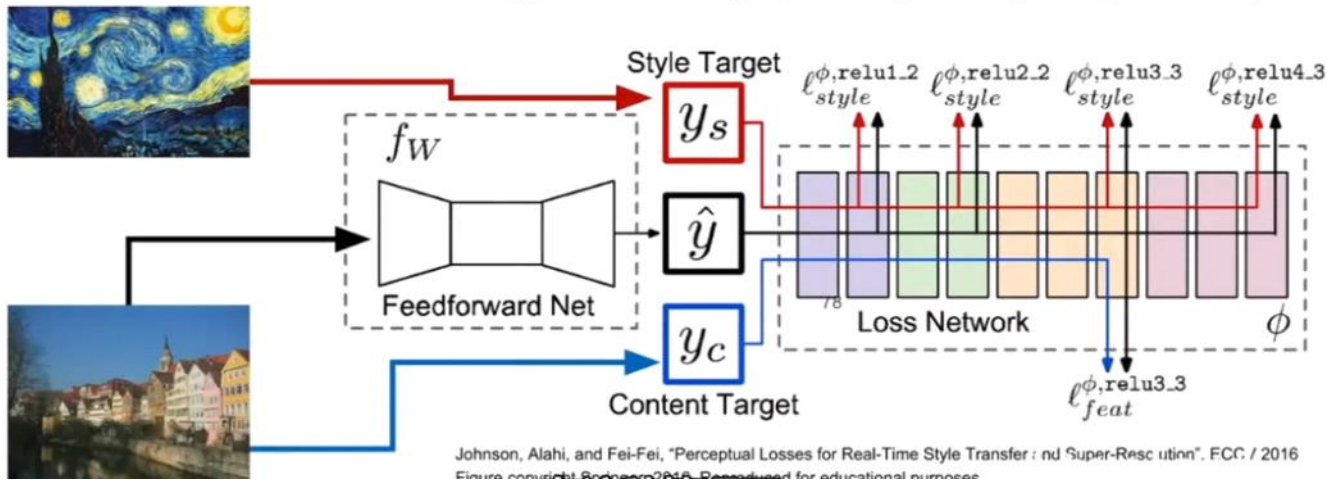
Neural Style Transfer

Resizing style image before running style transfer algorithm can transfer different types of features



Fast Style Transfer

- (1) Train a feedforward network for each style
- (2) Use pretrained CNN to compute same losses as before
- (3) After training, stylize images using a single forward pass



2卷积神经网络理解

1.what is going on inside convnets?

t-sne与pca