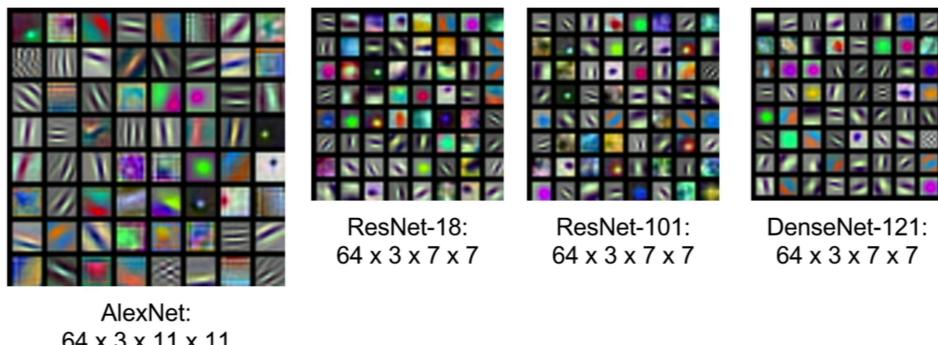


cs231n-5: Visualizing and Understanding And Style Transfer

1. 可视化某些层

1. 不管采用什么架构，第一层总是类似这种

1. 主要寻找一些边缘、角度、黑暗和光亮的颜色等特征



2. 中间层就很难解释了，尽管也是在寻找某些特征，但是无法直观的看出，可以用

1. **Maximally Activation Patches方法**

1. 如果中间某层的output（不是weight! !）size为 $128 \times 7 \times 7$ ，那么我们选择其中一层
2. 然后运行很多图片，找到这个output，然后排序，找到最大的几个patches
 1. 找到图片里面的一小片区域，使输出值达到最大（但我不知道如何实现，难道是卷积过程中找到最大的一片区域？）
 3. 输出这些最大的output对应的图片上的某个区域
 4. 可以发现他们都在寻找类似的结构
 1. 而且较低的层可能找类似眼镜鼻子嘴等部分
 2. 较高的层则试图寻找整张脸，整个人等等
 3. 这是因为层数越高，receptive field越大，所以patch越大

3. 最后一层FC层

1. 我们可以用**kNN的方法**

1. 在每张照片得到的FC层特征空间（在FC层会输出一个向量，比如维度为4096的向量）找到其k近邻
2. 可以发现尽管图像大不相同（比如大象朝左边站和右边站，甚至正面和侧面），但依然被当作近邻找出
3. 也就是说FC层显示了图片的一些语义内容（semantic contents），尽管我们没有刻意地去找到这种语义特征

2. 用降维（**Dimensionality Reduction**）方法去显示FC层的特征

1. PCA（主成分分析）或者说这里提到一个常用来可视化深度学习的特征的算法 **t-SNE**可以将图片计算得到FC层数据降到2维

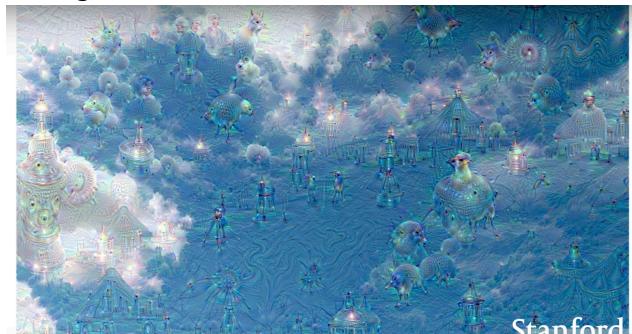
2. 我们可以将图片映射到这两个维度上对应的点，然后显示出来，发现它们附近都是自己的k近邻
2. Occlusion Experiments (阻塞实验)
 1. 挡住图片的某个区域（替换成平均像素值），来决定哪个区域使得Neural Network做出决定
 2. 最后输出一个热度图
3. Gradient Based Strategies
 1. Saliency Maps (特点映射)
 1. 计算图片上每个pixel的梯度，梯度越大说明对最后score的影响越大
 2. 这玩意还可以拿来做Segmentation! ! Cool! ! 而且是Unsupervised! !
 2. Intermediate Features
 1. naive地结合Saliency Maps: Compute gradient of neuron value with respect to image pixels
 2. Guided Back Propagation
 1. 只计算ReLU的正梯度，但是这里没有进行细节的讨论
 2. 可以找到更清晰的影响特定神经元的像素点
 3. $I^* = \operatorname{argmax}_I f(I) + R(I)$
 1. $R(I)$ 是正则项
 2. 用gradient ascent得到 I
 3. 最大化中间层某个neuron的输出值，甚至是FC层
 4. 可以看到它们在找寻什么特征，同样的，越高层的neuron会找到越大的接受域
 3. Fooling Images / Adversarial Examples
 1. 初始样本+增加扰动，找到一个可以让神经网络误分类的最小扰动
 2. 可能是GAN的原型？
 4. Deep Dream: Amplify Existing Features
 1. 具体过程：

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

Equivalent to:

$$I^* = \operatorname{arg} \max_I \sum_i f_i(I)^2$$
 2. 注意要对原图片加入一些扰动 (jitter)，最终最大化norm
 2. 如下图，底图天空上面出现了一些特征性的东西，较低的层出现的特征可能只是edges而已。



4. 接下来一个有趣的东西：Feature Inversion / Feature Reconstruction

- 给出一个图片的特征向量，找到的新图片的特征和特征向量匹配， \mathbf{I} 函数表示特征的区别

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

Given feature vector
Features of new image

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

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

Total Variation regularizer
(encourages spatial smoothness)

-
- 我们发现越高的层，重建效果越好（看上去就是原图啊...）
- 可以用来纹理合成 (Texture Synthesis)
 - Gram Matrix
 - 找到一层feature tensor，Size是CxHxW，变成一个有HW个Cx1的向量
 - 外积这个向量，得到CxC的矩阵
 - 取得HW个xCx矩阵的平均值，即Gram Matrix
 - 这个方法计算简单，真的找协方差矩阵的话，计算复杂度太高
 - Gram Matrix丢弃了空间上的信息，但是统计得到了二阶协方差信息
 - 然后我们用Gram Matrix作为特征向量，做Feature Inversion
 - 较高的层会找到类似的还原度相对高的图片，但因为不是特征向量，所以有较大差别
 - 较低的层就会出现纹理一样的东西了

5. 更有趣的东西：Style Transfer

- 输入两张图片，对一个随机的图片做Gradient Ascent，但是结合Feature Inversion和Gram Matrix Matching
 - 一个做Feature Inversion，一个做Gram Matrix Matching
 - 如果改变二者的权重、style图片的尺寸，还能改变图片的样式
- 也可以输入多张图片，找到混合的图片
- 问题是这个操作很慢，更快的可以这么做
 - 助教的Paper hah
 - (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
 - Google的论文
 - 将Batch Normalization改成Instance Normalization会提升性能
 - 并且一个Network可以提供多个Style（我也不知道为什么）
 - 也可以做Style Blending（风格混合）

6. 总结：

- Activation (基于激活值)
 - 最近邻

- 2. 降维
 - 3. 最大Patch
 - 4. 阻塞
2. Gradients (利用梯度建立新的图片来理解)
- 1. 特点映射 (Saliency Map)
 - 2. 可视化Class
 - 3. Fooling Images
 - 4. 特征重建
3. 函数 (应用Gradients方法Find More Fun 😂)
- 1. DeepDream
 - 2. Style Transfer