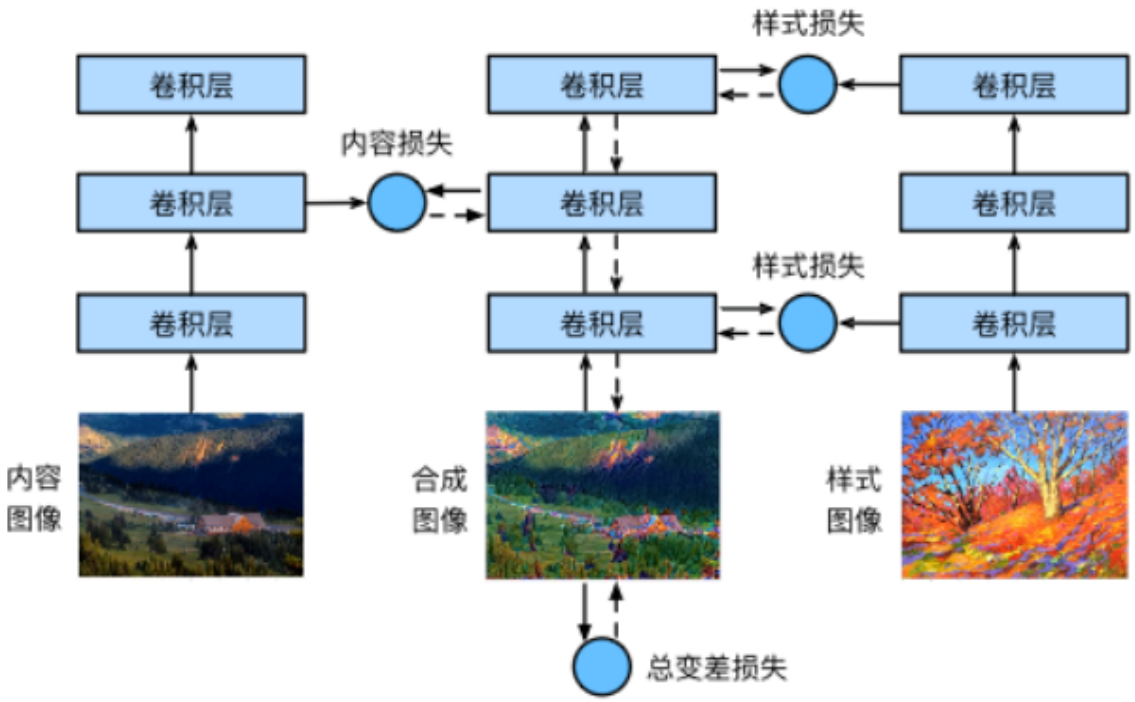
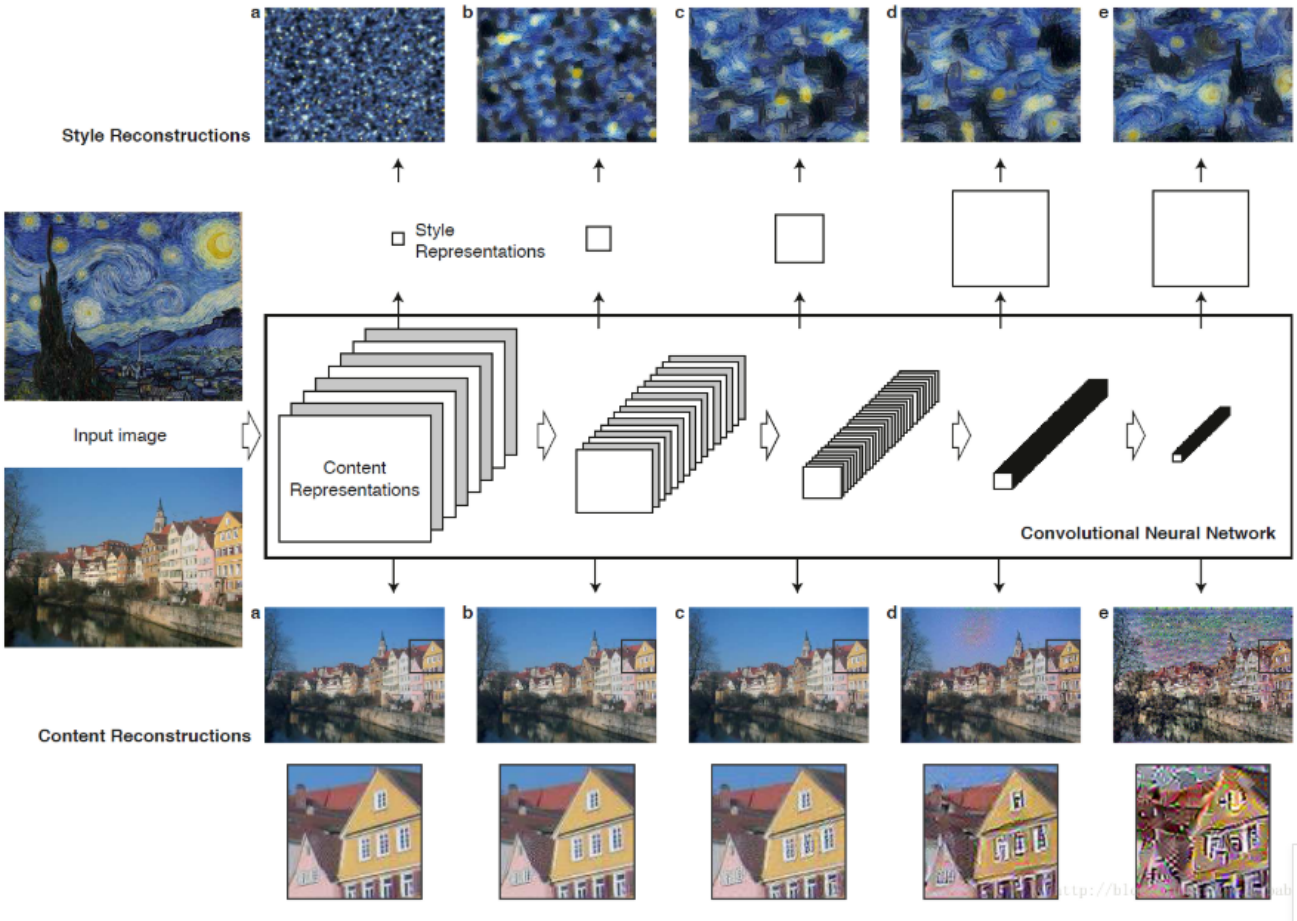
神经网络风格迁移


内容图像
合成图像
样式图像


样式损失
卷积层
卷积层
卷积层
内容损失
卷积层
卷积层
卷积层
样式损失
卷积层
卷积层
卷积层
内容
图像
合成
图像
样式
图像
总变差损失

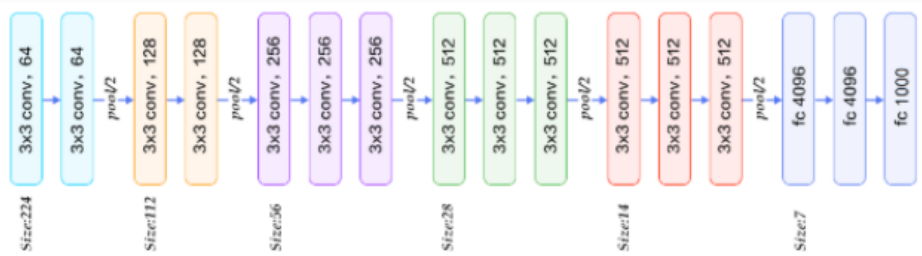


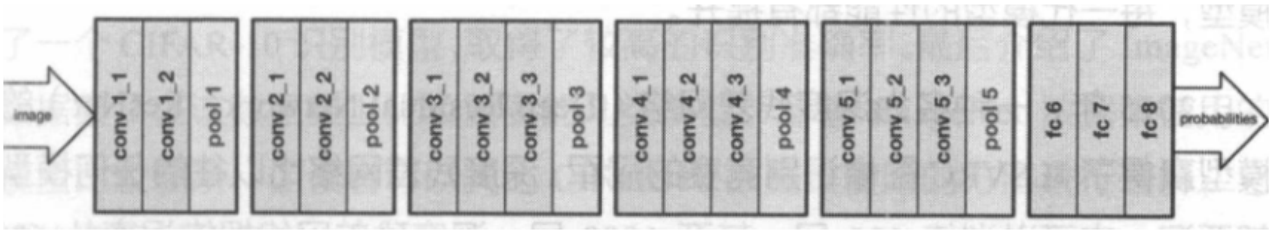
5张图分别来自VGG的`conv1 1 (a)`，`conv2 1 (b)`，`conv3 1 (c)`，`conv4 1 (d)`，`conv5 1(e)`。

微调（Fine-tune）原理

- 在自己的数据集上训练一个新的深度学习模型时，一般采取在预训练好的模型上进行微调的方法。
- VGGNet16

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

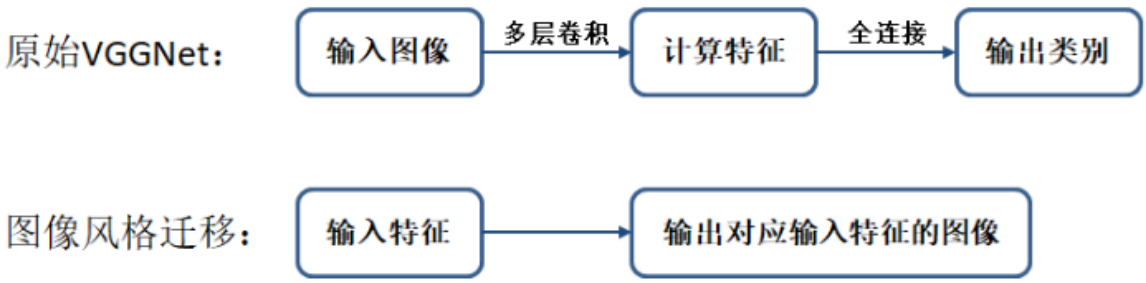




- VGG16的结构为卷积+全连接 = 16层。卷积层分为5个部分共13层，即图中的conv1~conv5。还有3层是全连接层，即图中的fc6、fc7、fc8。如果要将VGG16的结构用于一个新的数据集，首先要去掉fc8这一层。原因是fc8层的输入是fc7的特征，输出是1000类的概率，这1000类正好对应ImageNet模型中的1000个类别。采用符合数据集类别数的全连接层作为新的fc8。
- 在训练的时候，网络参数的初始值采用VGG16在ImageNet上已经训练好的参数作为训练的初始值。
- 载入VGG16的参数后，可以开始训练了。一般来说，可以选择以下几种范围进行训练：
    - 只训练fc8，好处是训练速度快，但往往性能不会太好。
    - 训练所有参数。训练速度慢，但是能取得较高的性能，可以充分发挥深度模型的威力。
    - 训练部分参数。通常是固定浅层参数不变，训练深层参数。如固定coonv1、conv2部分的参数不训练，只训练conv3、conv4、conv5、fc6、fc7、fc8的参数。
- 微调的原理大致就是先看懂网络的结构图，然后把网络的一部分修改成子集需要的模型。这种训练方法就是所谓的对神经网络模型做微调。借助微调，可以从预训练模型出发，将神经网络应用到自己的数据集上。

原始图像风格迁移

- VGGNet是输入图像，提取特征，并输出图像类别。图像风格迁移正好与其相反，输入的是特征，输出对应这种特征的图片。



- 还原图像的方法是梯度下降法。

还原图像的方法是梯度下降法。设原始图像为$\vec{p}$，期望还原的图像为$\vec{x}$（即自动生成的图像）。使用的卷积是第l层，原始图像$\vec{p}$在第l层的卷积特征为P_{ij}^{l}。i表示卷积的第i个通道，j表示卷积的第j个位置。通常卷积的特征是三维的，三维坐标分别对应（高、宽、通道）。此处不考虑具体的高和宽，只考虑位置j，相当于把卷积“压扁”了。比如一个10x10x32的卷积特征，对应$1 \leqslant i \leqslant 32$，$1 \leqslant j \leqslant 100$。对于生成图像$\vec{x}$，同样定义它在$l$层的卷积特征为$F_{ij}^{l}$。

有了上面这些符号后，可以写出“内容损失”（Content Loss）。内容损失$L_{content}(\vec{p},\vec{x},l)$的定义是：

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

$L_{content}(\vec{p},\vec{x},l)$描述了原始图像$\vec{p}$和生成图像$\vec{x}$在内容上的“差异”。内容损失越小，说明它们的内容越接近；内容损失越大，说明它们的内容差距也越大。先使用原始图像$\vec{p}$计算出它的卷积特征P_{ij}^{l}，同时随机初始化$\vec{x}$。接着，以内容损失$L_{content}(\vec{p},\vec{x},l)$为优化目标，通过梯度下降法逐步改变$\vec{x}$。经过一定步数后，得到的$\vec{x}$是希望的还原图像了。在这个过程中，内容损失$L_{content}(\vec{p},\vec{x},l)$应该是越来越小的。

- 还原图像的风格的方法是使用图像的卷积层特征的Gram矩阵。
  - Gram矩阵：是为了表达图像的纹理特征。Gram就是对两个feature map求内积，结果和位置没有关系。
    - 在feature map中，每个数字都来自于一个特点滤波器在特定位置的卷积，因此每个数字代表一个特征的强度，而Gram计算的实际是两两特征之间的相关性，哪两个特征是同时出现的，哪两个是此消彼长的。
    - 同时，Gram的对角线元素，体现了每个特征在图像中出现的量。
    - 有助于把握整个图像的大体风格。

Gram矩阵是关于一组向量的内积的对称矩阵，例如，向量组$\overrightarrow{x_1},\overrightarrow{x_2},...,\overrightarrow{x_n}$的Gram矩阵是

$$\begin{bmatrix}(\overrightarrow{x_1},\overrightarrow{x_1}) & (\overrightarrow{x_1},\overrightarrow{x_2}) & \dots & (\overrightarrow{x_1},\overrightarrow{x_n})\\(\overrightarrow{x_2},\overrightarrow{x_1}) & (\overrightarrow{x_2},\overrightarrow{x_2}) & \dots & (\overrightarrow{x_2},\overrightarrow{x_n})\\\dots & \dots & \dots & \dots\\(\overrightarrow{x_n},\overrightarrow{x_1}) & (\overrightarrow{x_n},\overrightarrow{x_2}) & \dots & (\overrightarrow{x_n},\overrightarrow{x_n})\end{bmatrix}$$

通常取内积为欧几里得空间上的标准内积，即$(\overrightarrow{x_i},\overrightarrow{x_j})=\overrightarrow{x_i}^{T}\overrightarrow{x_j}$。

设卷积层的输出为F_{ij}^{l}，那么这个卷积特征对应的Gram矩阵的第i行第j个元素定义为

$$G_{ij}^{l}=\sum_{k}F_{ik}^{l}F_{jk}^{l}$$

设在第l层中，卷积特征的通道数为N_l，卷积的高、宽乘积为M_l，那么F_{ij}^l满足$1 \leqslant i \leqslant N_l$，$1 \leqslant j \leqslant M_l$。G实际是向量组$F_1^l$，$F_2^l$，...，$F_i^l$，...，$F_{N_l}^l$的Gram矩阵，其中，其中$F_i^l = (F_{i1}^l, F_{i2}^l, \ldots, F_{ij}^l, \ldots, F_{iM_l}^l)$。

此处数学符号较多，因此再举一个例子来加深读者对此Gram矩阵的理解。假设某一层输出的卷积特征为10x10x32，即它是一个宽、高均为10，通道数为32的张量。F_1^l表示第一个通道的特征，它是一个100维的向量，F_2^l表示第二个通道的特征，它同样是一个100维的向量，它对应的Gram矩阵G是

$$\begin{bmatrix} (F_1^l)^T(F_1^l) & (F_1^l)^T(F_2^l) & \ldots & (F_1^l)^T(F_{32}^l) \\ (F_2^l)^T(F_1^l) & (F_2^l)^T(F_2^l) & \ldots & (F_2^l)^T(F_{32}^l) \\ \ldots & \ldots & \ldots & \ldots \\ (F_{32}^l)^T(F_1^l) & (F_{32}^l)^T(F_2^l) & \ldots & (F_{32}^l)^T(F_{32}^l) \end{bmatrix}$$

Gram矩阵可以在一定程度上反映原始图片中的“风格”。仿照“内容损失”，还可以定义一个“风格损失”（Style Loss）。设原始图像为$\vec{a}$，要还原的风格图像为$\vec{x}$，先计算出原始图像某一次卷积的Gram矩阵为A^l，要还原的图像$\vec{x}$经过同样的计算得到对应卷积层的Gram矩阵是G^l，风格损失定义为

$$L_{style}(\vec{p}, \vec{x}, l) = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (A_{ij}^l - G_{ij}^l)^2$$

分母上的$4N_l^2 M_l^2$是一个归一化项，目的是防止风格损失的数量级相比内容损失过大。在实际应用中，常常利用多层而非一层的风格损失，多层的风格损失是单层风格损失的加权累加，即$L_{style}(\vec{p}, \vec{x}) = \sum_i w_l L_{style}(\vec{p}, \vec{x}, l)$，其中$w_l$表示第$l$层权重。

- **利用内容损失还原图像内容。**
- **利用风格损失还原图像风格。**
- 将内容损失和风格损失结合起来，在还原图像内容的同时还原图像风格。

设原始的内容图像为$\vec{p}$，原始的风格图像为$\vec{a}$，待生成的图像为$\vec{x}$。希望$\vec{x}$可以保持内容图像$\vec{p}$的内容，同时具备风格图像$\vec{a}$的风格。因此组合$\vec{p}$的内容损失和$\vec{a}$的风格损失，定义总的损失函数为

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

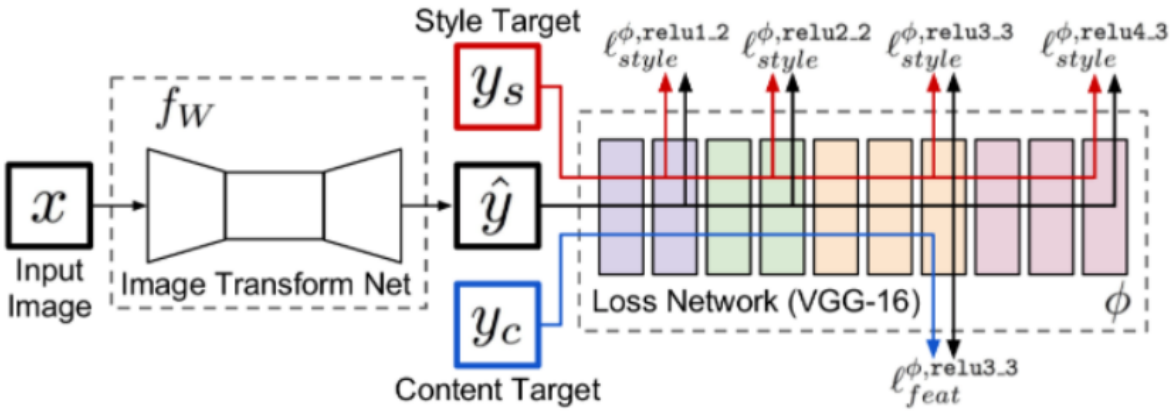
α，β是平衡两个损失的超参数。如果α偏大，还原的图像会更接近于$\vec{p}$中，如果β偏大，还原的图像会更接近$\vec{a}$。使用总的损失函数可以组合$\vec{p}$的内容和$\vec{x}$的风格，这实现了图像风格的迁移。部分还原的图像如下图所示

- 缺点：

一张较大的图片，这大大的限制了这项技术的使用场景。速度慢的原因在于，要使用总损失$L_{total}(\vec{p},\vec{a},\vec{x})$优化图片$\vec{x}$，这意味着生成一张图片需要几百步梯度下降法的迭代，而每一步的迭代都需要耗费大量的时间。从另一个角度看，优化$\vec{x}$可以看作是一个“训练模型”的过程，以往都是针对模型参数训练，而这里训练的目标是图片$\vec{x}$，而训练模型一般都比执行训练好的模型要慢很多。下面将会讲到快速图像风格迁移，它把原来的“训练”的过程变成了一个“执行”的过程，因此大大加快了生成风格话图片的过程。

快速图像风格迁移原理

- 快速图像风格迁移的方法是：不使用优化的方法来逐步迭代生成x，而是使用一个神经网络生成x。网络结构如图：

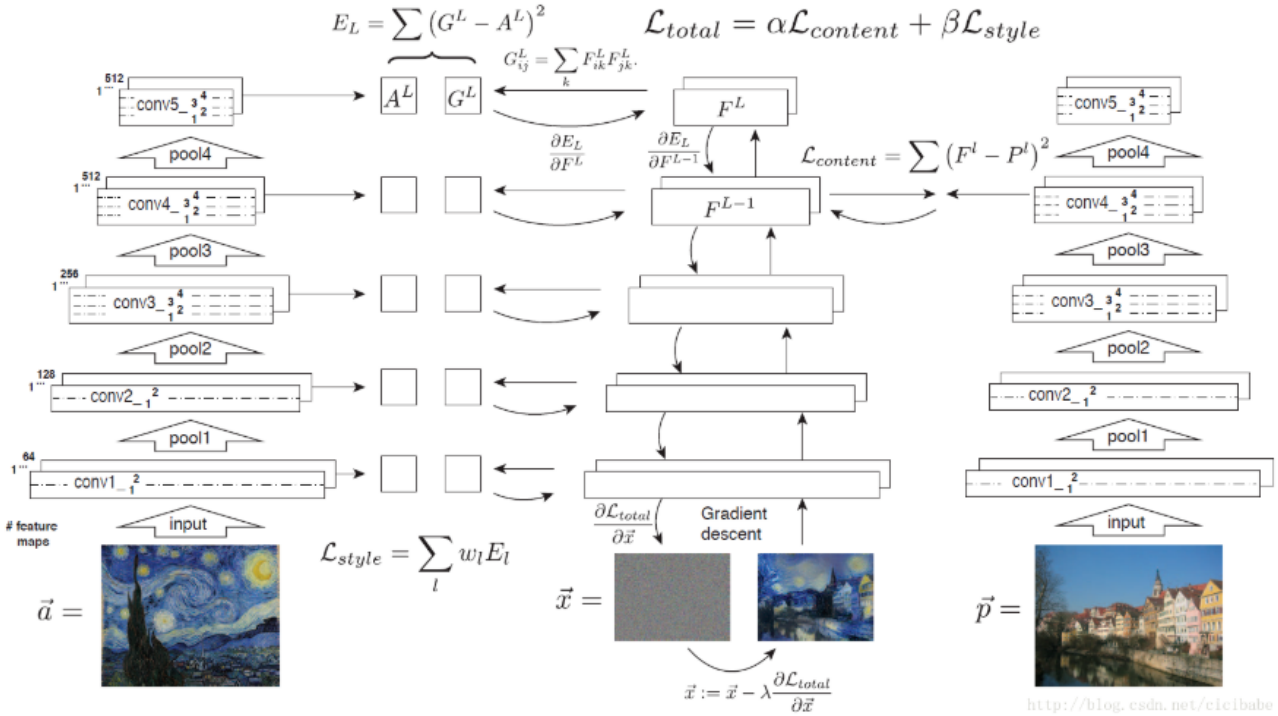


  - 整个系统由两个神经网络组成，他们在图中由两个休闲裤分别标出。左边是图像生成网络，右边是损失网络。损失网络实际是VGGNet。利用损失网络来定义内容损失、风格损失。这个损失用来实例图像生成网络。图像生成网络的职责是生成某一种风格的图像，它的输入是一张图像，输出同样是一张图像。由于输出图像只需要在网络中计算一遍，所以速度快。

同样使用数学符号严格地阐述上面地过程：设输入的图像为$\vec{x}$，经过图像生成网络生成的图像为$\vec{y}$。$\vec{y}$在内容上应该与原始的内容图像$\vec{y}_c$接近，因此可以利用损失网络定义内容损失$L_{content}(\vec{y},\vec{y}_c)$，内容损失使用的是VGG-16中的relu3_3层输出的特征，对应上图中的$l^{\phi,relu3_3}_{feat}$。另一方面，我们还希望$\vec{y}$具有目标风格图像$\vec{y}_s$的风格，因此又可以定义一个风格损失$L_{total}(\vec{y},\vec{y}_c,\vec{y}_s)$。定义风格损失时使用了VGG-16的四个中间层relu1_2，relu2_2，relu3_3，relu4_3，对应图中的$l^{\phi,relu1_2}_{style}$、$l^{\phi,relu2\_2}_{style}$、$l^{\phi,relu3\_3}_{style}$、$l^{\phi,relu4\_3}_{style}$。同样组合这两个损失得到一个总损失$L_{total}(\vec{y},\vec{y}_c,\vec{y}_s)$。利用总损失可以训练图像生成网络。训练完成后直接使用图像生成网络生成图像。值得一提的是，在整个训练过程中，一般只固定一种风格$\vec{y}_s$，而内容图像$\vec{y}_c$取和输入$\vec{x}$一样，即$\vec{y}_s$=$\vec{x}$。

- 原始图像风格迁移与快速图像风格迁移比较

类型	损失定义	是否需要训练新网络	生成图像的方法
原始图像风格迁移	组合内容损失$L_{content}$与风格损失L_{style}	否。只需要预训练好的VGGNet	利用损失，通过梯度下降法计算适合的图像
快速图像风格迁移		是。除了预训练好的VGGNet，还需要训练图像生成网络	利用训练好的图像生成网络之间生成



- 当在网络的低层上匹配内容，算法会匹配照片上的大部分像素细节信息，生成的图像似乎艺术图的纹理几乎不融合进照片中。相反，在网络高层上匹配内容特征，照片的像素细节信息没有很强的约束，艺术画的纹理和照片的内容恰当地融合在一起。也就是说，图像中明确的结构，比如边缘和颜色地图会被改变，使用艺术画的风格和照片的内容。