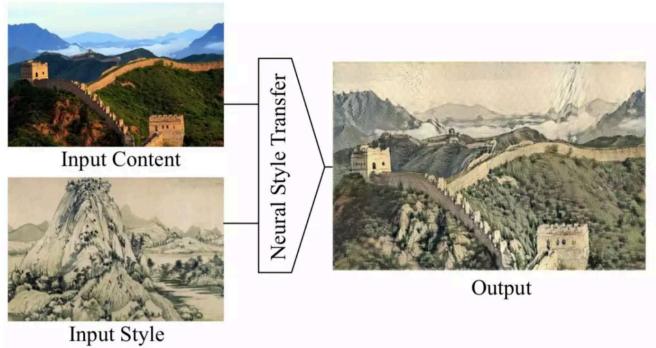


看了几篇神经风格迁移的文章，简单记录。



神经风格迁移的输入是一个内容图像，一个风格图像希望输出带有目标风格、保留了原图像内容的图。早期的图像风格迁移往往会对每一种风格建立一个统计模型，然后让内容图片更好的符合模型，这样的效果就是学到的模型完全不通用。直到 **gatys** 开创性的利用 **vgg** 网络对图像进行特征提取，才产生了更加通用的风格迁移模型。

1. 基于优化方法的慢速迁移

神经风格迁移最初源于纹理建模，纹理基于视觉上有重复结构并伴随着一些随机变化的单元，纹理生成问题是通过对纹理图像进行建模，来生成相似的纹理图案。分为两类方法：基于统计分布的参数化纹理建模和基于 MRF 的非参数化纹理建模，前者是将纹理建模为 N 阶统计量，后者是将图片分为小的 patch，利用 patch 相似度进行逐点合成。

当时 **vgg** 网络横空出世，在物体识别领域大获成功，**gatys** 认为除了对图像本身提取各种统计量，也可以直接用训练好的 **vgg** 网络作为特征提取器，在生成的不同深度的 **feature map** 上学习 **channel** 之间的相关性，具体来说，对每一层都得到一个 **gram** 矩阵 $\{G^1, G^2, \dots, G^L\}$ ，其中第 l 层的第 i、j 个 **feature map** 之间的内积即为 **gram** 矩阵的元素，**gram** 矩阵可以看做 **feature** 之间的偏心协方差矩阵，即将每一个位置点看做一个样本，不同 **channel** 的值看做是不同的维度，计算不同维度两两之间的协方差，这反应了两个维度之间是否容易被同时激活：

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

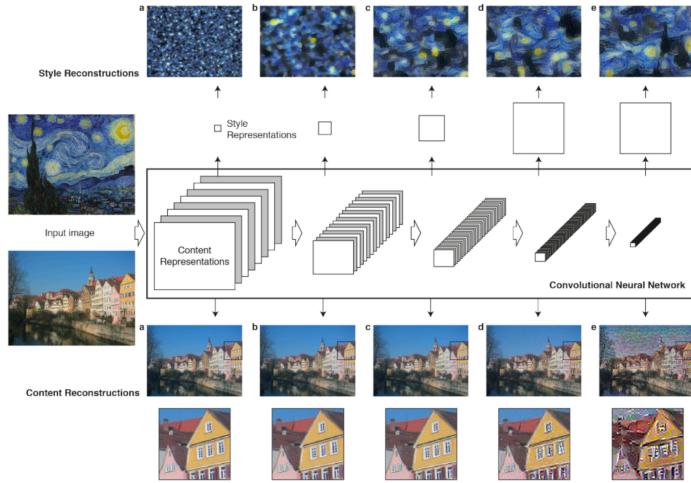
如果认为这些 **gram** 矩阵可以代表纹理的特性，那么我们只要从一张白噪音图片开始，令其输入 **vgg** 网络中得到的 **gram** 矩阵与目标纹理图片的 **gram** 矩阵尽可能接近即可，目标如下，可以直接用梯度下降解这个优化问题：

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G_{ij}^l - \hat{G}_{ij}^l)^2$$

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

进一步，**gatys** 发现纹理实际上也可以描述一张图片的风格，那么如果想要进行风格迁移，除了要学习目标图片的风格还要保留原图片的内容。由于 **vgg** 网络是在物体识别的任务上训练的，随着网络的加深，网络会逐渐更加关心输入图片的内容。如果对 **vgg** 每层进行可视化（如下图），发现低层几乎保留了所有的

像素信息，而高层捕捉到了整体的物体和排列布局，但是丧失了像素细节，所以我们可以直接将高层的 feature map 作为图片的内容信息。



如何利用 feature map 来重建图像呢，定义 p 和 x 分别为原图和迁移后产生的图像，则我们只需要令 x 第 l 层的 feature map 与 p 的尽可能接近即可：

$$\mathcal{L}_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} (F_{ij}^l - P_{ij}^l)^2, \quad F^l \in \mathcal{R}^{N_l \times M_l}$$

其中 N_l 是 l 层卷积核的数量（即 feature map 的数量）， M_l 为 feature map 展成一维的长度。

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

总结来说，这两篇文章的贡献就在于发现了 **style** 和 **content** 可以分开表示分别利用 vgg 来提取，甚至可以控制图片保留风格还是内容的相对的权重。卷积得到的 feature map 会保留图片的内容这很容易理解，但是为什么 feature 之间的相关性会反应图片的纹理\风格呢？

在 ijcai17 中的这篇文章，严格的证明了匹配 gram 矩阵相当于最小化 MMD (maximum mean discrepancy)，风格迁移可以看做是一个 domain adaptation 问题，(即在源 domain 和目标 domain 数据分布不一致时，如何迁移在源 domain 训练出来的模型，来适应目标 domain，对目标 domain 的 label 进行预测)

MMD 是利用两个分布的样本来对两个分布的差异进行估计的统计量，平方 mmd 为：

$$\begin{aligned} & \text{MMD}^2[X, Y] \\ &= \|\mathbf{E}_x[\phi(\mathbf{x})] - \mathbf{E}_y[\phi(\mathbf{y})]\|^2 \\ &= \left\| \frac{1}{n} \sum_{i=1}^n \phi(\mathbf{x}_i) - \frac{1}{m} \sum_{j=1}^m \phi(\mathbf{y}_j) \right\|^2 \\ &= \frac{1}{n^2} \sum_{i=1}^n \sum_{i'=1}^n \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_{i'}) + \frac{1}{m^2} \sum_{j=1}^m \sum_{j'=1}^m \phi(\mathbf{y}_j)^T \phi(\mathbf{y}_{j'}) \\ &\quad - \frac{2}{nm} \sum_{i=1}^n \sum_{j=1}^m \phi(\mathbf{x}_i)^T \phi(\mathbf{y}_j), \end{aligned}$$

可以引入 kernel function，则：

$$\begin{aligned}
& \text{MMD}^2[X, Y] \\
&= \frac{1}{n^2} \sum_{i=1}^n \sum_{i'=1}^n k(\mathbf{x}_i, \mathbf{x}_{i'}) + \frac{1}{m^2} \sum_{j=1}^m \sum_{j'=1}^m k(\mathbf{y}_j, \mathbf{y}_{j'}) \\
&\quad - \frac{2}{nm} \sum_{i=1}^n \sum_{j=1}^m k(\mathbf{x}_i, \mathbf{y}_j).
\end{aligned}$$

经过推导，发现 **style loss** 可以转化为二阶多项式 **kernel** 的最大均值差异，其中样本点是 **feature maps** 的每一个点所有 **channel** 看做一个向量。

$$\begin{aligned}
\mathcal{L}_{style}^l &= \frac{1}{4N_l^2 M_l^2} \sum_{i=1}^{N_l} \sum_{j=1}^{N_l} \left(\sum_{k=1}^{M_l} F_{ik}^l F_{jk}^l - \sum_{k=1}^{M_l} S_{ik}^l S_{jk}^l \right)^2 \\
&= \frac{1}{4N_l^2 M_l^2} \sum_{i=1}^{N_l} \sum_{j=1}^{N_l} \left(\left(\sum_{k=1}^{M_l} F_{ik}^l F_{jk}^l \right)^2 + \left(\sum_{k=1}^{M_l} S_{ik}^l S_{jk}^l \right)^2 - 2 \left(\sum_{k=1}^{M_l} F_{ik}^l F_{jk}^l \right) \left(\sum_{k=1}^{M_l} S_{ik}^l S_{jk}^l \right) \right) \\
&= \frac{1}{4N_l^2 M_l^2} \sum_{i=1}^{N_l} \sum_{j=1}^{N_l} \sum_{k_1=1}^{M_l} \sum_{k_2=1}^{M_l} (F_{ik_1}^l F_{jk_1}^l F_{ik_2}^l F_{jk_2}^l + S_{ik_1}^l S_{jk_1}^l S_{ik_2}^l S_{jk_2}^l - 2F_{ik_1}^l F_{jk_1}^l S_{ik_2}^l S_{jk_2}^l) \\
&= \frac{1}{4N_l^2 M_l^2} \sum_{k_1=1}^{M_l} \sum_{k_2=1}^{M_l} \sum_{i=1}^{N_l} \sum_{j=1}^{N_l} (F_{ik_1}^l F_{jk_1}^l F_{ik_2}^l F_{jk_2}^l + S_{ik_1}^l S_{jk_1}^l S_{ik_2}^l S_{jk_2}^l - 2F_{ik_1}^l F_{jk_1}^l S_{ik_2}^l S_{jk_2}^l) \\
&= \frac{1}{4N_l^2 M_l^2} \sum_{k_1=1}^{M_l} \sum_{k_2=1}^{M_l} \left(\left(\sum_{i=1}^{N_l} F_{ik_1}^l F_{ik_2}^l \right)^2 + \left(\sum_{i=1}^{N_l} S_{ik_1}^l S_{ik_2}^l \right)^2 - 2 \left(\sum_{i=1}^{N_l} F_{ik_1}^l S_{ik_2}^l \right)^2 \right) \\
&= \frac{1}{4N_l^2 M_l^2} \sum_{k_1=1}^{M_l} \sum_{k_2=1}^{M_l} \left((\mathbf{f}_{k_1}^l)^T \mathbf{f}_{k_2}^l)^2 + (\mathbf{s}_{k_1}^l)^T \mathbf{s}_{k_2}^l)^2 - 2(\mathbf{f}_{k_1}^l)^T \mathbf{s}_{k_2}^l)^2 \right), \\
\mathcal{L}_{style}^l &= \frac{1}{4N_l^2 M_l^2} \sum_{k_1=1}^{M_l} \sum_{k_2=1}^{M_l} \left(k(\mathbf{f}_{k_1}^l, \mathbf{f}_{k_2}^l) + k(\mathbf{s}_{k_1}^l, \mathbf{s}_{k_2}^l) - 2k(\mathbf{f}_{k_1}^l, \mathbf{s}_{k_2}^l) \right) \\
&= \frac{1}{4N_l^2} \text{MMD}^2[\mathcal{F}^l, \mathcal{S}^l],
\end{aligned}$$

由此我们可以得到的结论是，图片的风格可以由 CNN 得到的不同层的特征的分布来表示，对匹配 **gram** 矩阵相当于是在对齐生成图片和风格图片的分布。既然如此本文提出了可以换不同的 **kernel** 来作为 **style loss**。

此外，本文还提出了一个非常有意义的 **style loss**: BN 层统计量匹配(**channel-wise** 的均值和方差)。这个思路来源于在 **domain adaptation** 领域，有工作发现，BN 层包含着不同 **domain** 的的分布规律，因此对源 **domain** 和目标 **domain** 使用不同的 BN 统计量来解决两个域分布不一致问题。既然图像风格迁移做的也是 **domain adaptation** 问题，我们可以直接对齐 **domain** 统计量：

$$\mathcal{L}_{style}^l = \frac{1}{N_l} \sum_{i=1}^{N_l} \left((\mu_{F^l}^i - \mu_{S^l}^i)^2 + (\sigma_{F^l}^i - \sigma_{S^l}^i)^2 \right).$$

其中均值为在某个 **channel** 所有位置的点的均值：

$$\mu_{F^l}^i = \frac{1}{M_l} \sum_{j=1}^{M_l} F_{ij}^l, \quad \sigma_{F^l}^i = \frac{1}{M_l} \sum_{j=1}^{M_l} (F_{ij}^l - \mu_{F^l}^i)^2$$

实验表明，不同的 **kernel** 捕捉了风格的不同方面，这些 **loss** 的提出为风格迁移提供了丰富的选择。其中 **BN loss** 直接启发了后续的单模型多风格的迁移方法

的提出。

除了基于统计分布的纹理建模，还存在一种基于 MRF 的纹理建模方法，也衍生了一个图像风格迁移的分支。马尔科夫随机场即满足成对、局部或者全局马尔科夫性的联合分布，也称为概率无向图模型。局部马尔科夫性是说，对于该无向图中某一个点，不于它相连的随机变量集合和该点在与它相连的变量集合的条件下是独立的。如果我们将图像看做是一个马尔科夫随机场，则图像中某一个像素点可能的概率值分布，只和这个像素点周围的空间像素点信息有关系，而和该图像中剩余的像素点没有关系，也就是这个像素点对除了它周围的像素点以外的该图像的其他像素点是独立的。

因此我们可以将图像划分为以每个像素点为中心的 patch，我们可以定义该点关于周围点的条件概率分布。对想要生成的图片的每一个像素点所在 patch，首先定义块之间的距离 d ，找到纹理图片中最为接近的 patch，然后用该 patch 通过最大化条件概率分布得到生成图片中对应像素点的值。

如果想要结合 MRF 方法和 CNN，则我们可以选择不再原图上而是在图片经过 vgg 网络提取的特征上定义 MRF loss：

$$E_s(\Phi(\mathbf{x}), \Phi(\mathbf{x}_s)) = \sum_{i=1}^m \|\Psi_i(\Phi(\mathbf{x})) - \Psi_{NN(i)}(\Phi(\mathbf{x}_s))\|^2$$
$$NN(i) := \arg \min_{j=1, \dots, m_s} \frac{\Psi_i(\Phi(\mathbf{x})) \cdot \Psi_j(\Phi(\mathbf{x}_s))}{|\Psi_i(\Phi(\mathbf{x}))| \cdot |\Psi_j(\Phi(\mathbf{x}_s))|}$$

其中 $\Phi(\mathbf{x})$ 是 \mathbf{x} 在某一层的 feature maps， Ψ 是该 feature map 上划分的 $k*k*C$ 的 patch， C 是 channel 数。也就是说对于一个 patch，寻找所有 style image 中与之最接近的 patch，然后 min 两者之间的欧式距离。通过这样的方式我们可以得到局部的纹理特征，而为了保留内容，同样使用上一篇文章提出的内容损失。

$$E_c(\Phi(\mathbf{x}), \Phi(\mathbf{x}_c)) = \|\Phi(\mathbf{x}) - \Phi(\mathbf{x}_c)\|^2$$

最后，由于我们都是在深层的特征空间中操作，浅层的像素信息有所损失，所以文章提出了一个正则化项，使得生成的图片更加平滑：

$$\Upsilon(\mathbf{x}) = \sum_{i,j} ((x_{i,j+1} - x_{i,j})^2 + (x_{i+1,j} - x_{i,j})^2)$$

比起匹配 gram 矩阵的额方法，基于 MRF 的方法可以更好的保留 style image 中局部的结构和纹理，除了可以进行艺术风格图片的生成，还可以生成真实的摄影图片。

二. 基于离线模型优化的快速风格迁移

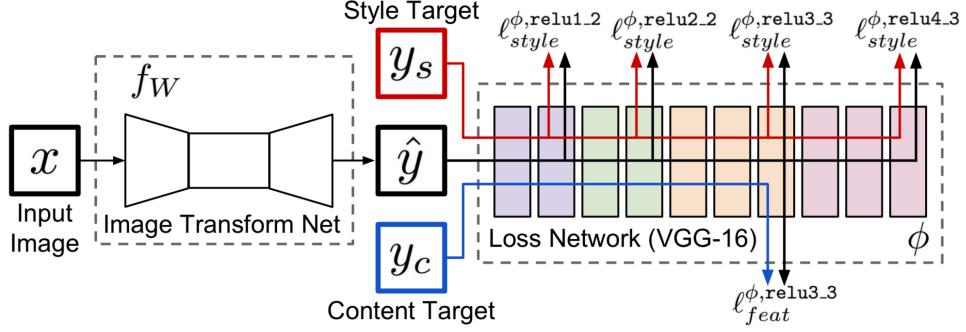
在线模型优化的速度非常慢，为了对图像进行快速重建，用一个前向网络来实时得到 gatys 提出的优化目标的解，根据单个网络可以学习多少种风格可以分为单风格、多风格、任意风格的快速风格迁移。

1. 单模型单风格

有两篇文章 Perceptual Losses for Real-Time Style Transfer and Super-Resolution, Texture Networks: Feed-forward Synthesis of Textures and Stylized Images，思路是相似的，只是前馈网络的结构有所不同。

我们知道图像风格迁移是纹理建模+图像重建，前面着重讲了纹理建模的方法，而图像重建也有两种方法，第一种就是上一节描述的，从一个白噪音图像开始，不断改变图像的每所有像素值使得我们定义的 loss—一般是定义重建图像的

某些特征上，比如高阶统计量或者 **perceptual loss**，每次重建都要从噪音图形迭代优化很多次；第二种方式是训练一个前向网络，给定一批的输入输入来训练这个网络，比如对低分辨率图像重建任务。第二种方法最原始的 **loss** 是 **per-pixel loss**。于是就有工作提出了，能不能也用前馈网络同时结合 **perceptual loss** 来进行风格迁移。框架如下：



有两个网络一个是 **image transformation network**，一个是预训练好的 **vgg** 网络作为 **loss network**。对于一个输入图片 x ，有多个 **target** 图片（内容、风格），网络的 **loss** 由内容 **loss** 和风格 **loss** 的加权和得到：

$$W^* = \arg \min_W \mathbf{E}_{x, \{y_i\}} \left[\sum_{i=1} \lambda_i \ell_i(f_W(x), y_i) \right]$$

上图给出了一个统一的框架，对于风格迁移问题，内容图片同时作为输入，对于低分辨率重建问题，没有 **style target**。

$$\ell_{feat}^{\phi,j}(\hat{y}, y) = \frac{1}{C_j H_j W_j} \|\phi_j(\hat{y}) - \phi_j(y)\|_2^2$$

$$\ell_{style}^{\phi,j}(\hat{y}, y) = \|G_j^\phi(\hat{y}) - G_j^\phi(y)\|_F^2$$

$$G_j^\phi(x)_{c,c'} = \frac{1}{C_j H_j W_j} \sum_{h=1}^{H_j} \sum_{w=1}^{W_j} \phi_j(x)_{h,w,c} \phi_j(x)_{h,w,c'}$$

2. 单模型多风格

利用前馈网络解决了单张图片风格迁移都要进行很多次优化训练的问题，我们已经可以用一个网络保存一种风格，只需要将待迁移的图片输进去即可。但我们需要为每一个风格训练一个模型，这仍是开销很大的。于是有一些工作又提出了用一个网络训练多个风格的方法，第一种方法作者称为 **stylebank network**，结构如下图：

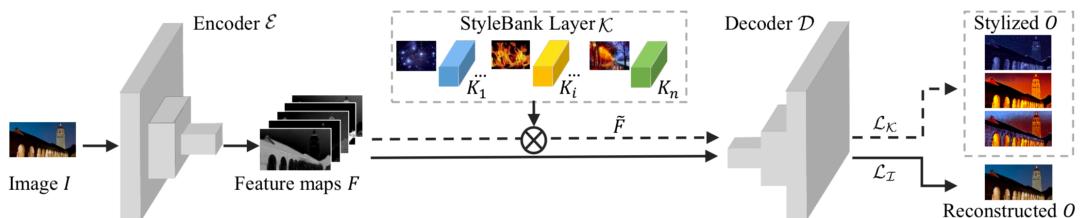


Figure 1. Our network architecture consists of three modules: image encoder \mathcal{E} , StyleBank layer \mathcal{K} and image decoder \mathcal{D}

网络有两条训练分支：autoencoder 和风格化分支。这两条分支共享 encoder 和 decoder 网络结构，在风格化分支中在 AE 之间插入 stylebank 层，对应某一个 style。

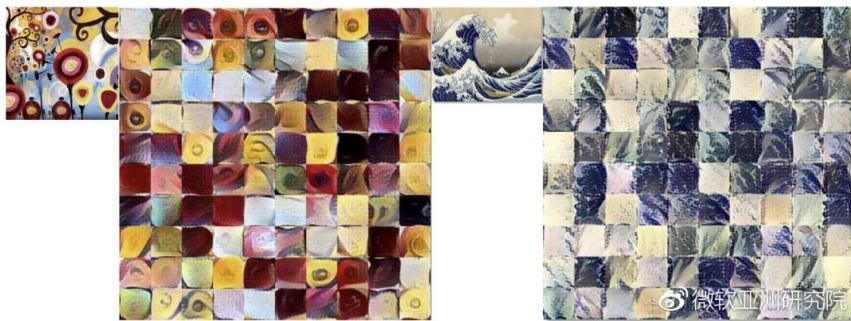
AE 分支的 loss 为 MSE loss：

$$\mathcal{L}_{\mathcal{I}}(I, O) = \|O - I\|^2$$

风格化分支为 perceptual loss：

$$\mathcal{L}_{\mathcal{K}}(I, S_i, O_i) = \alpha \mathcal{L}_c(O_i, I) + \beta \mathcal{L}_s(O_i, S_i) + \gamma \mathcal{L}_{tv}(O_i)$$

在训练的时候迭代的训练这两部分。训练完成后，当有新的风格输入时，我们可以固定 AE 的部分只训练一个新的 stylebank 层；通过对各个 style 卷积核进行可视化，我们发现它表达了风格图片中的不同纹理单元。



A LEARNED REPRESENTATION FOR ARTISTIC STYLE 则对不同的 style 进行了更高程度的耦合。这篇文章认为不同的 style 图像可能共享了一些相同东西，比如说呐喊和星夜都是油画，没有必要用两个单独的网络来生成。我们可以完全通过一个网络来生成不同的风格，只要将想要转化的 style 作为一个 id 当成网络的输入。

首先简单介绍下 BN，REVISITING BATCH NORMALIZATION FOR PRACTICAL DOMAIN ADAPTATION 这篇文章发现，通过调整 BN 统计量，或学习的参数 beta 和 gamma，BN 可以用来做 domain adaptation。提出了一个叫做 adaBN 的方法来做 domain adaptation。adaBN 做的事情看上去非常 trivial：在拓展到未知的 domain 的时候，将 BN 的统计量在这个 domain 的 unlabel data 上重新计算一遍。在 CNN 中，BN 和 IN 的区别为：

an input batch $x \in \mathbb{R}^{N \times C \times H \times W}$, BN normalizes the mean and standard deviation for each individual feature channel:

$$BN(x) = \gamma \left(\frac{x - \mu(x)}{\sigma(x)} \right) + \beta \quad (1)$$

where $\gamma, \beta \in \mathbb{R}^C$ are affine parameters learned from data; $\mu(x), \sigma(x) \in \mathbb{R}^C$ are the mean and standard deviation, computed across batch size and spatial dimensions independently for each feature channel:

$$\mu_c(x) = \frac{1}{NHW} \sum_{n=1}^N \sum_{h=1}^H \sum_{w=1}^W x_{nchw} \quad (2)$$

$$\sigma_c(x) = \sqrt{\frac{1}{NHW} \sum_{n=1}^N \sum_{h=1}^H \sum_{w=1}^W (x_{nchw} - \mu_c(x))^2 + \epsilon} \quad (3)$$

$$IN(x) = \gamma \left(\frac{x - \mu(x)}{\sigma(x)} \right) + \beta \quad (4)$$

Different from BN layers, here $\mu(x)$ and $\sigma(x)$ are computed across spatial dimensions independently for each channel and each sample:

$$\mu_{nc}(x) = \frac{1}{HW} \sum_{h=1}^H \sum_{w=1}^W x_{nchw} \quad (5)$$

$$\sigma_{nc}(x) = \sqrt{\frac{1}{HW} \sum_{h=1}^H \sum_{w=1}^W (x_{nchw} - \mu_{nc}(x))^2 + \epsilon} \quad (6)$$

文章 Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. 发现只需将 BN 替换为 instance normalization (IN) 即可大幅提升 transfer 的收敛速度, 于是在后续一些的图像迁移工作中我们将 BN 替换为 IN。

我们想到, 既然 style transfer 是一个 domain adaption 的问题, 源 domain 和目标 domain 分别是两张图片, 图片上的每个点可以看做是一个样本。那么我们也可以通过调整参数 γ 和 β 来适应不同的 style! 我们对不同的 style 学习不同的参数即可;

$$z = \gamma_s \left(\frac{x - \mu}{\sigma} \right) + \beta_s$$

其中对于每一层我们有 $N \times C$ 个参数 γ , N 是 style 的数量, C 是 channel 的数量。这样参数量比起单独训练 N 个网络, 或者单独训练 N 个卷积核都会大大下降。我们相当于将每个 style 嵌入到了低维空间中, 而当需要训练新的 style 时, 只需要初始化新的缩放参数而固定已经训练好的 CNN 网络, 训练很少量的参数。

3. 单模型任意风格

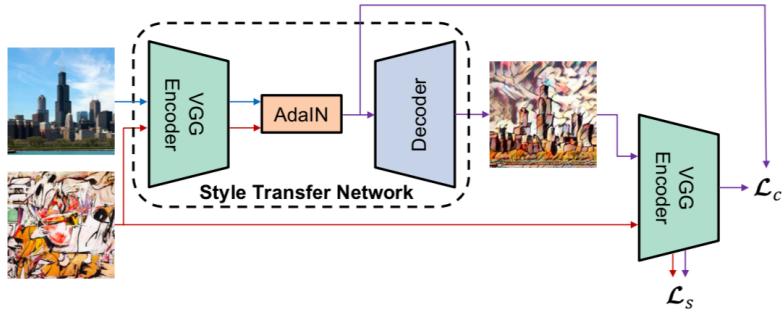
虽然可以实现单一网络产生多种风格迁移, 但是对于新的风格, 我们还是要额外的训练时间。那么有没有什么方法可以只学一个网络, 对于新的风格可以直接实现风格迁移呢? 也就是 zero-shot style transfer 问题。解决这个问题目前有以下几种思路

1. style prediction network, 基于上面的 CIN 模型, 既然对于一个风格图我们可以得到一个低维的参数, 那么可以直接训练一个网络, 用风格图作为输入, 将仿射变换的参数作为输出。
2. meta learning、不展开
3. Adaptive Instance Normalization
4. learning-free method

先讲第三个思路。我们前面讲到 CVPR17 的一篇文章发现将 BN 替换为 IN 会大幅度的提高 transfer 的收敛速度。Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization 这篇文章的作者考虑为什么 IN 会提高收敛速度, 按照之前那篇文章的说法, 因为对单独一张图片做 normalization 不会改变图片的对比度, 于是作者实验了将所有输入图片 normalize 为相同的对比度, 再去做风格迁移, 发现仍然 IN 比 BN 快很多。而如果将所有图片转换为同一种风格, BN 和 IN 的收敛速度差不多。于是作者得到一个重要的结论是: IN 是一种 style

normalization。

如果这个假设成立的话，BN 相当于把一批图片的标准化到一个中心的 style 上去，但是每张图片的 style 仍然是不同的，(每张图片的都有不同的均值和方差)。然而 IN 将所有的图片都标准化到均值为 0 方差为 1 的分布中去，统一了所有图片的 style。这也是 CIN 为什么会 work 的原因，每一个 style 对应着不同的均值和方差。于是文章提出了一种更神奇的方法，用一个 AdaIN 层，同时接收内容图和风格图(的 feature map)作为 input，然后用风格图的 channel-wise 的均值和方差去标准化内容图的的均值和方差，以使得内容图的风格迁移到目标风格上。



具体来说，用一个 AE 作为图像的生成器，输入为内容图和风格图，其中 encoder 是 VGG 网络的前几层，固定住不进行训练，然后得到的 feature map 经过 AdaIN 层：

$$t = \text{AdaIN}(f(c), f(s))$$

$$T(c, s) = g(t)$$

decoder 层是唯一需要训练的结构，这里不加任何的 BN 或者 IN 结构，否则会将已经 transfer 的 style 重新改变。

Loss 同样是包括两部分，需要注意的是，关于内容的 loss 是去匹配内容图经过 AdaIN 层的 feature map 和生成图片相应层的 feature，因为作者实验发现这样会收敛的更快。第二部分作者分别试过匹配 gram 矩阵和匹配均值方差这些统计量，效果差不多。

$$\mathcal{L} = \mathcal{L}_c + \lambda \mathcal{L}_s$$

$$\mathcal{L}_c = \|f(g(t)) - t\|_2$$

$$\begin{aligned} \mathcal{L}_s = & \sum_{i=1}^L \|\mu(\phi_i(g(t))) - \mu(\phi_i(s))\|_2 + \\ & \sum_{i=1}^L \|\sigma(\phi_i(g(t))) - \sigma(\phi_i(s))\|_2 \end{aligned}$$

最后一篇 Universal Style Transfer via Feature Transforms 发表在 nips17，非常强的一篇，比起上一篇通过 IN 可以对齐 style，这一篇完全用统计学习的手段找出了到底什么是 style，如何完全去除和恢复 style。基于此提出了一种不需要训练的快速迁移方法。

整个过程只需要预训练一个 AE，encoder 任然是 VGG 的前几层，loss 如下：

$$L = \|I_o - I_i\|_2^2 + \lambda \|\Phi(I_o) - \Phi(I_i)\|_2^2$$

然后通过白化操作可以去除内容图片的风格，通过 **coloring transform**（实际上就是白化操作的逆，将不相关的各 **feature map** 重新建立相关性，协方差矩阵为 **style img** 各 **channel** 的协方差矩阵。）

Whitening transform. Before whitening, we first center f_c by subtracting its mean vector m_c . Then we transform f_c linearly as in (2) so that we obtain \hat{f}_c such that the feature maps are uncorrelated ($\hat{f}_c \hat{f}_c^\top = I$),

$$\hat{f}_c = E_c D_c^{-\frac{1}{2}} E_c^\top f_c, \quad (2)$$

where D_c is a diagonal matrix with the eigenvalues of the covariance matrix $f_c f_c^\top \in \mathbb{R}^{C \times C}$, and E_c is the corresponding orthogonal matrix of eigenvectors, satisfying $f_c f_c^\top = E_c D_c E_c^\top$.

Coloring transform. We first center f_s by subtracting its mean vector m_s , and then carry out the coloring transform [14], which is essentially the inverse of the whitening step to transform \hat{f}_c linearly as in (3) such that we obtain \hat{f}_{cs} which has the desired correlations between its feature maps ($\hat{f}_{cs} \hat{f}_{cs}^\top = f_s f_s^\top$),

$$\hat{f}_{cs} = E_s D_s^{\frac{1}{2}} E_s^\top \hat{f}_c, \quad (3)$$

where D_s is a diagonal matrix with the eigenvalues of the covariance matrix $f_s f_s^\top \in \mathbb{R}^{C \times C}$, and E_s is the corresponding orthogonal matrix of eigenvectors. Finally we re-center the \hat{f}_{cs} with the mean vector m_s of the style, i.e., $\hat{f}_{cs} = \hat{f}_{cs} + m_s$.