

15 图像增广，微调和样式迁移

VCG

概要

- 图像增广
- 微调
 - 低层特征可通用
 - 高层特征重新初始化
- 样式迁移
 - 内容损失
 - 样式损失
 - 总损失函数

图像增广

为什么要数据增广 - CES'19 真实故事

- 一家初创公司的演示：一款智能自动售货机，通过相机识别客户挑选的物品
- 但演示失败，因为展厅有不同的
 - 光温
 - 桌子上的光反射
- 他们整夜工作
 - 重新收集数据并训练新模型
 - 订购了桌布



为什么要数据增广

➤ 图像增广的原因

- 生成相似但不同的训练样本，从而扩大了训练集的规模
- 随机改变训练样本可以减少模型对某些属性的依赖，从而提高模型的泛化能力

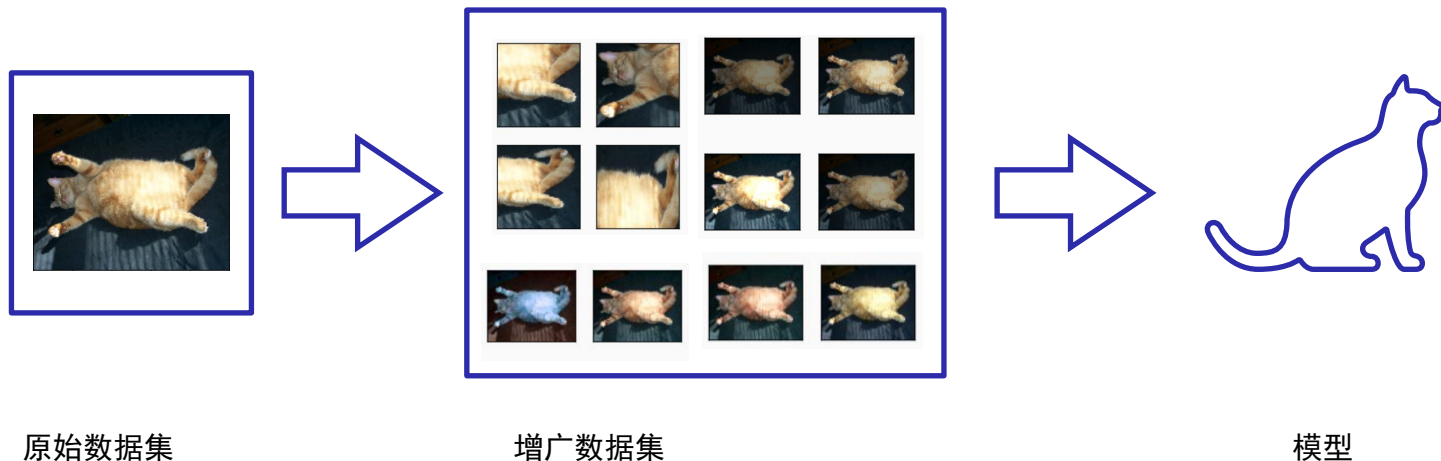
➤ 现有数据集基础上，增广为具有更多多样性的数据集

- 在训练数据集中添加各种背景噪音
- 转换为其他图像：改变颜色，改变形状



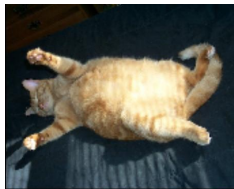
图像增广

➤ 工作模式

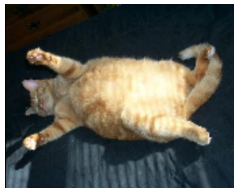


反转

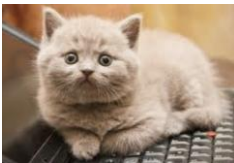
➤ 左右反转



➤ 上下反转

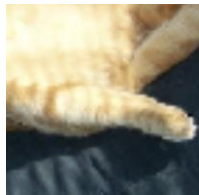
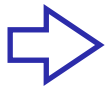
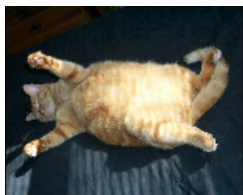


➤ 不一定都有效



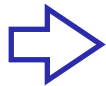
裁剪

- 从图像中裁剪一块区域，然后调整其大小
 - 随机宽高比（例如 $[3/4, 4/3]$ ）
 - 随机区域大小（例如 $[8\%, 100\%]$ ）
 - 一个随机的位置



改变颜色

► 调整色调，饱和度和亮度（例如 $[0.5, 1.5]$ ）



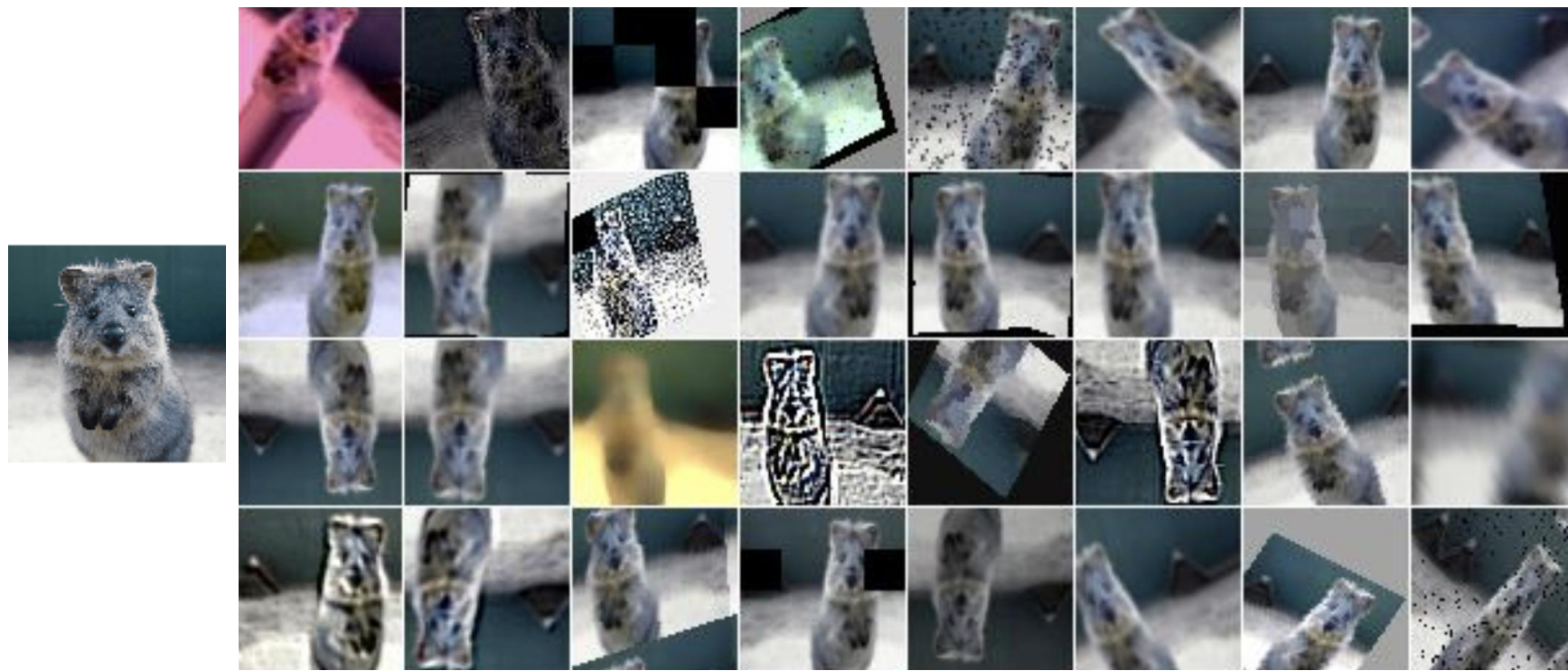
亮度



色调



更多方法



<https://github.com/aleju/imgaug>

微调

为什么要微调？

➤ 训练模型所需要的数据集，标注非常昂贵

现有数据集

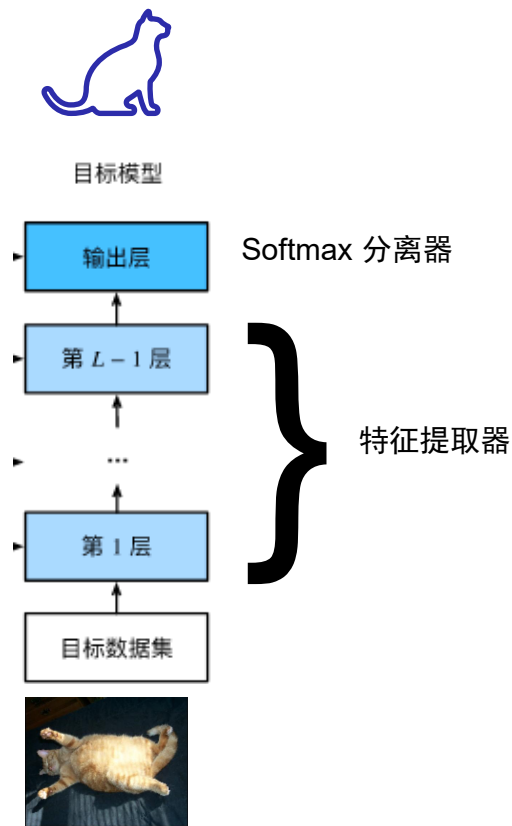


2 2 2 2 2 2 2 2 2 2 2 2
3 3 3 3 3 3 3 3 3 3 3 3
4 4 4 4 4 4 4 4 4 4 4 4
5 5 5 5 5 5 5 5 5 5 5 5
6 6 6 6 6 6 6 6 6 6 6 6

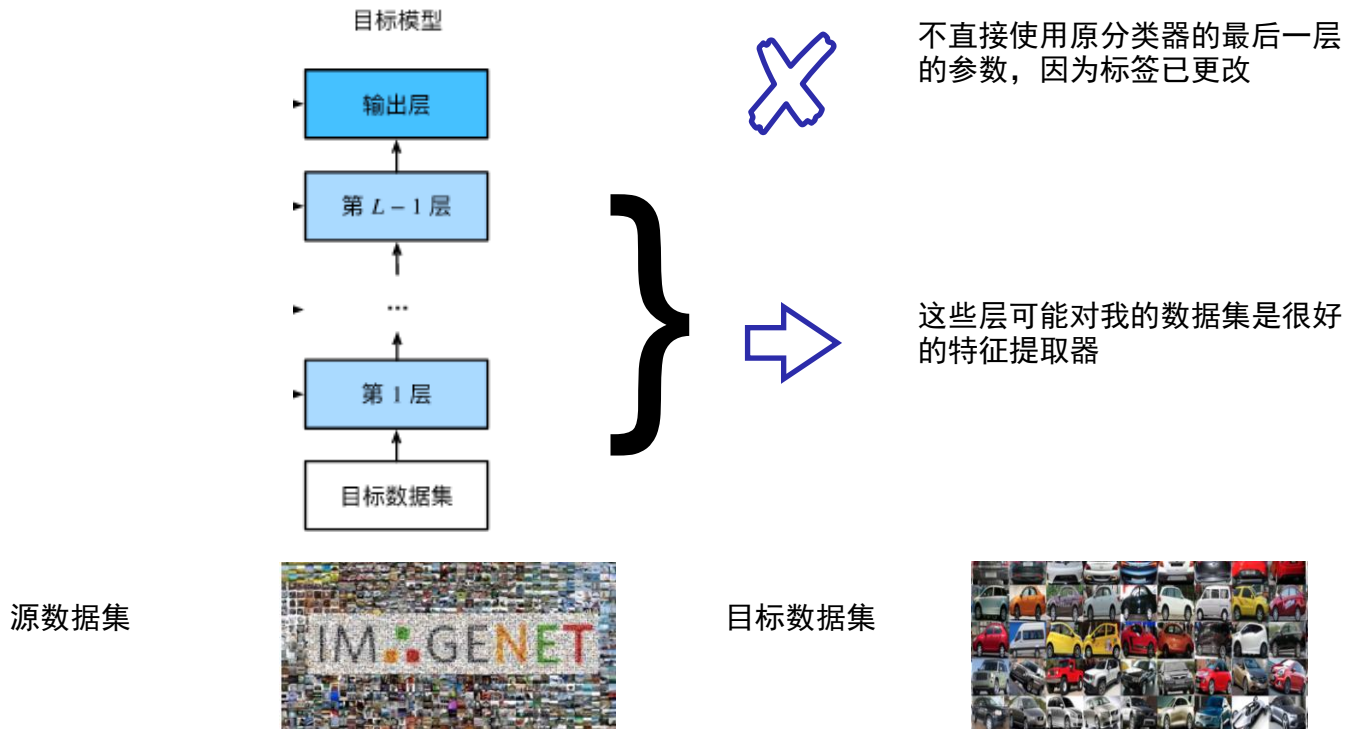
# 样本	1.2 M	50K	60 K
# 类别数	1,000	100	10

如何微调？

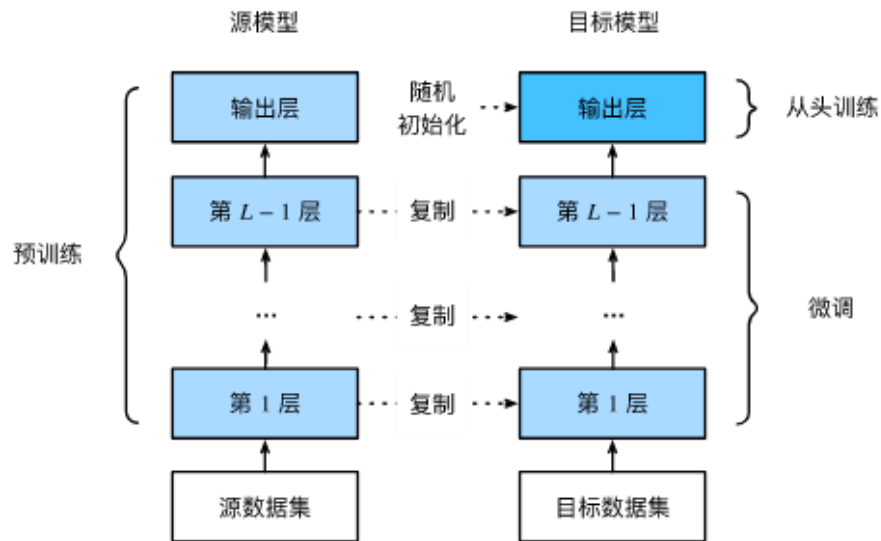
- 神经网络可以大致分为两部分
 - 特征提取器将原始像素映射为线性可分离的特征
 - 用线性分类做决定



如何微调？



微调中的权重初始化



源数据集



目标数据集



微调训练

- 在已有神经网络上微调目标数据集，但具有强大的正则化
 - 使用较小的学习率
 - 使用较少的迭代周期
- 如果源数据集比目标数据集更复杂，则微调通常会得到更高质量的模型

重用已有分类器的参数

- 源数据集可能包含目标数据集中的某些类别
- 在初始化期间使用来自预训练模型的相应权重



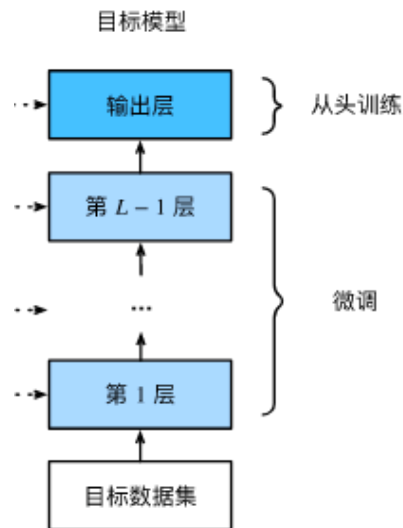
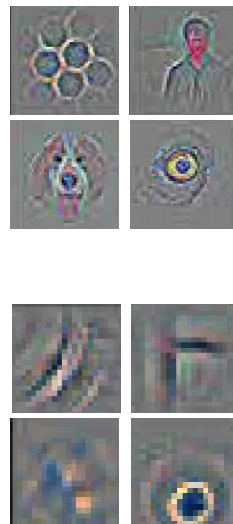
Racer, race car, racing car

A fast car that competes in races



修复一些层

- 神经网络学习分层特征表示
 - 低层特征是通用的
 - 高层特征与数据集中的对象更相关
- 在微调期间修复底层的参数
 - 另一个强有力的正则



风格迁移



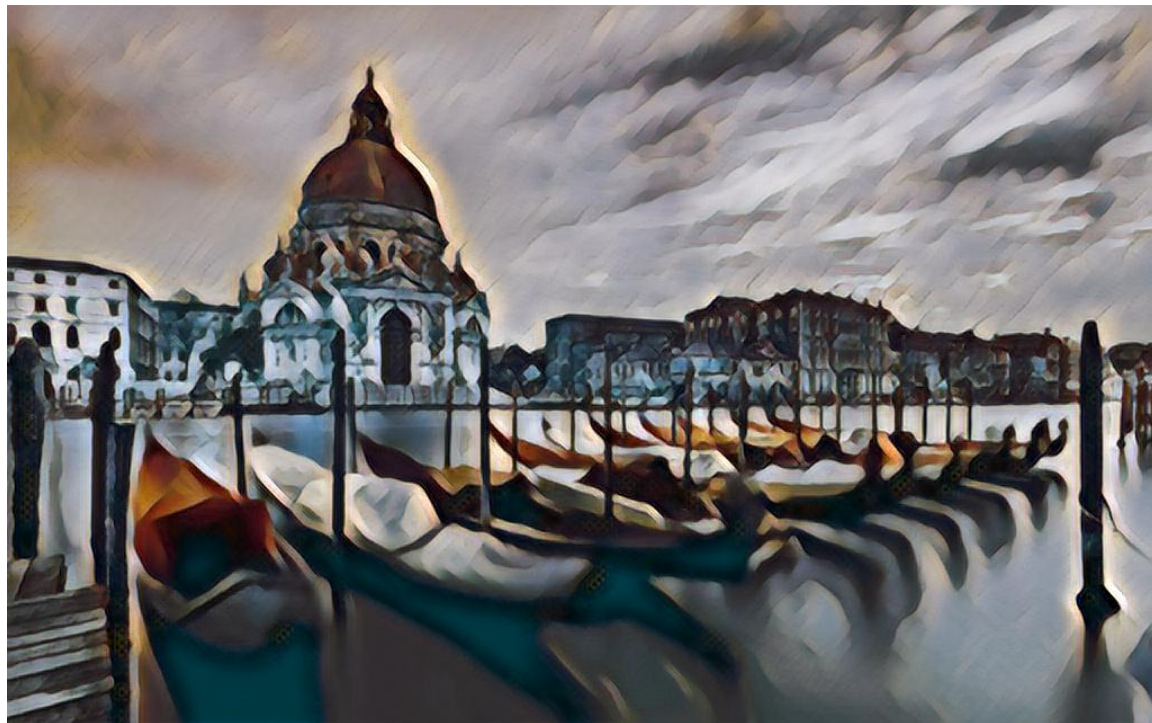
+



<https://github.com/zhanghang1989/MXNet-Gluon-Style-Transfer/>



+





+





+





+



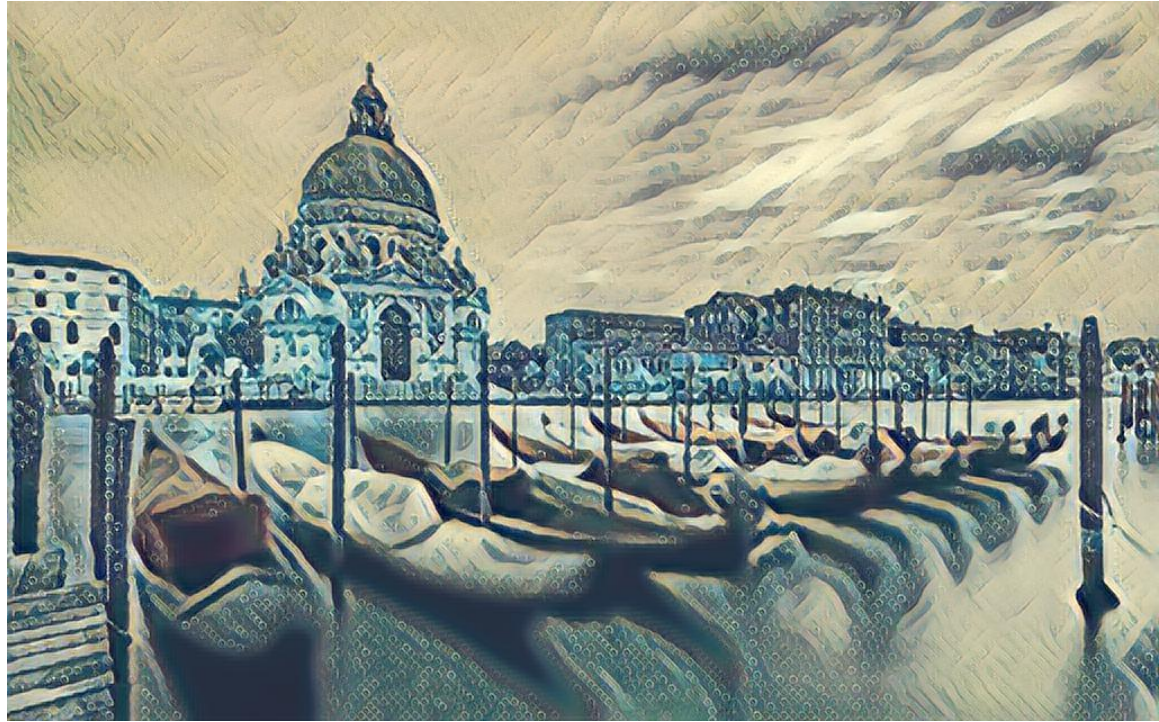


+





+



神经元风格

➤ 学习合成图像

➤ 内容图像中的内容

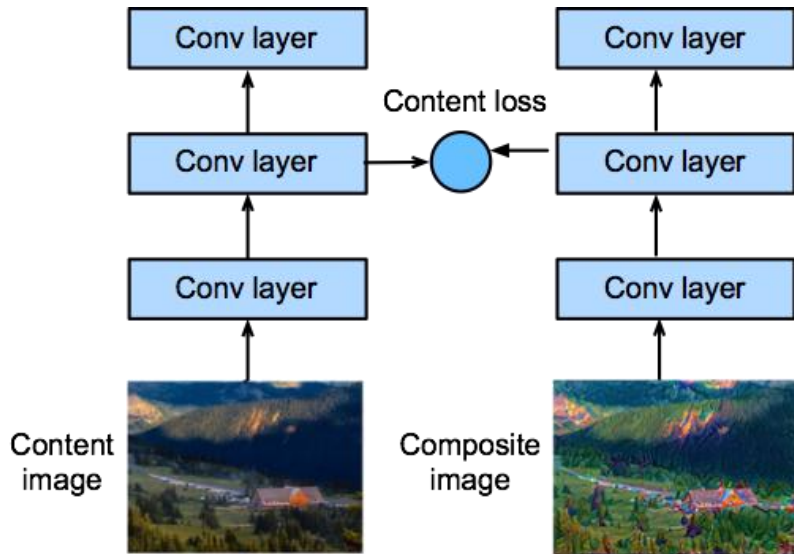
➤ 样式图像中的样式

$$\operatorname{argmin}_I w_1 \ell_{\text{content}}(I_{\text{content}}, I) + w_2 \ell_{\text{style}}(I_{\text{style}}, I) + w_3 \ell_{\text{noise}}(I)$$

内容损失

- 将内容图像和合成图像都放入 CNN
- 比较双方的平方损失
 - 底层匹配细节
 - 顶层匹配全局内容

➤ 第2层



样式损失

➤ 格拉姆矩阵 (Gram matrix) G

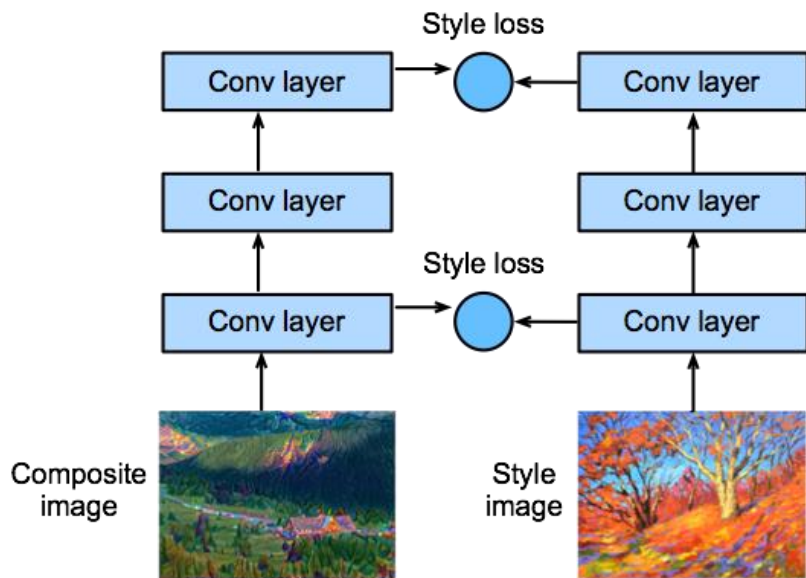
➤ $G_{i,j}$ 是通道 i 和 j 之间的内积

➤ 通过平方损失比较内部层输出的 G

➤ 底层匹配本地样式

➤ 顶层匹配全局样式

➤ 第1, 3层



噪声损失

- 学习的合成图像可能具有许多高频噪声
- 使用总变差损失（全变分损失）来降噪

$$\sum_{i,j} |x_{i,j} - x_{i+1,j}| + |x_{i,j} - x_{i,j+1}|$$

Original image



Noisy image

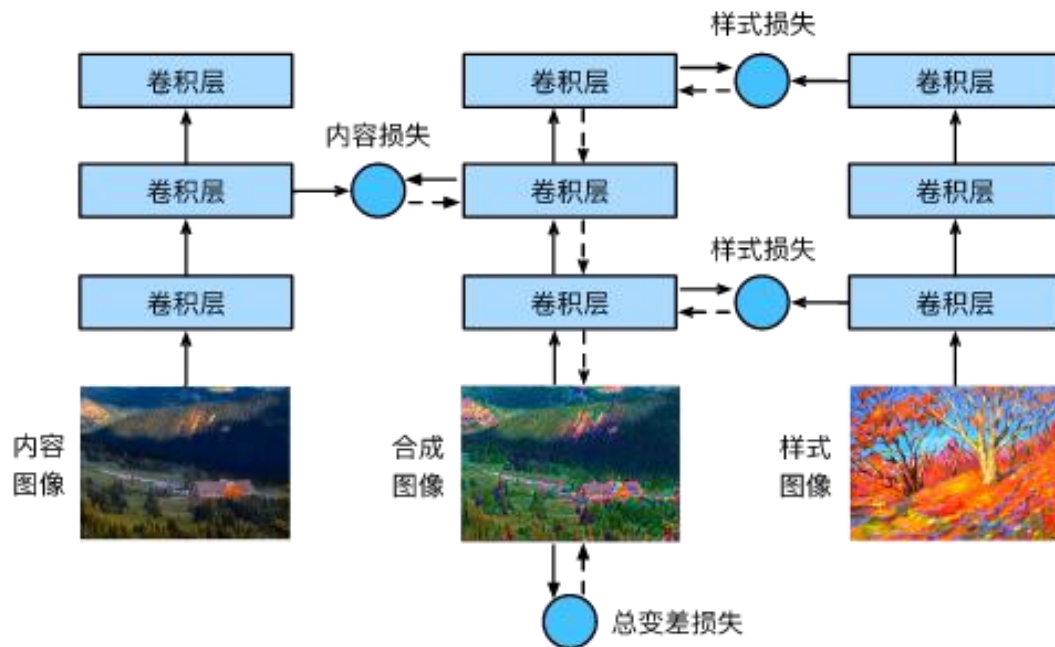


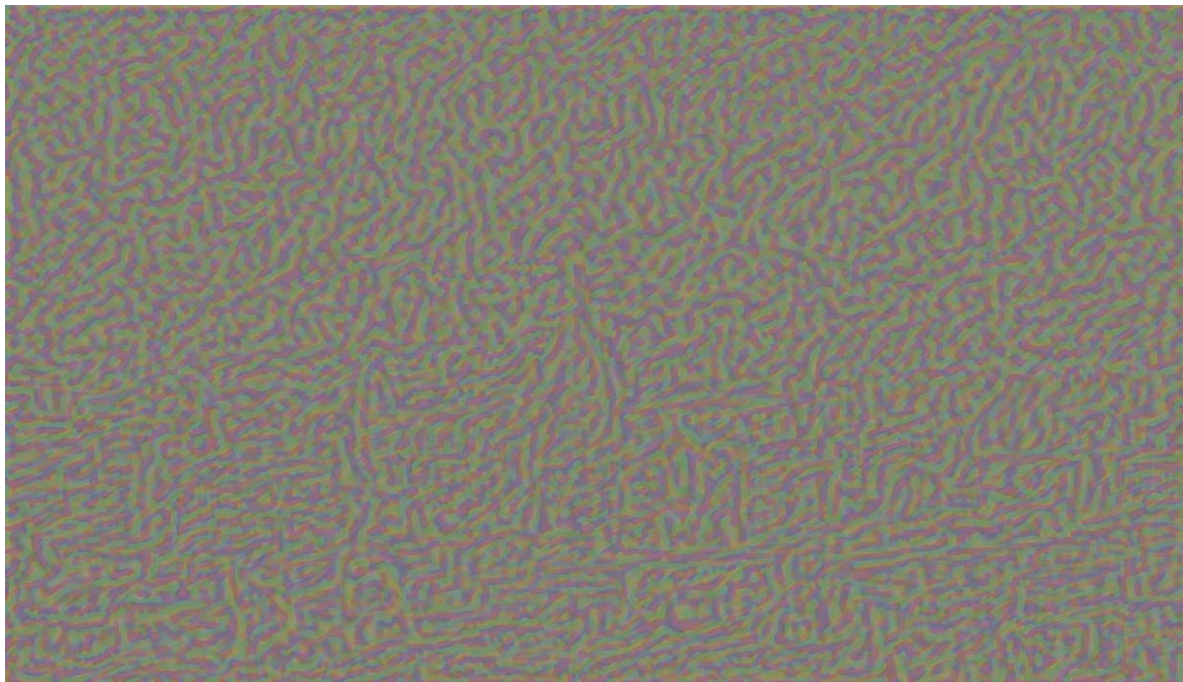
Denoised image



总损失函数

➤ $\arg \min_I \quad w_1 \ell_{\text{content}}(I_{\text{content}}, I) + w_2 \ell_{\text{style}}(I_{\text{style}}, I) + w_3 \ell_{\text{noise}}(I)$





总结

- 图像增广
- 微调
 - 低层特征可通用
 - 高层特征重新初始化
- 样式迁移
 - 内容损失
 - 样式损失
 - 总损失函数