Deep Bilateral Learning for Real-Time Image Enhancement

Dezeming Family

2023 年 7 月 1 日

正常字体：表示论文的基本内容解释。

粗体：表示需要特别注意的内容。

红色字体：表示容易理解错误或者混淆的内容。

蓝色字体：表示额外增加的一些注释。

绿色字体：表示额外举的一些例子。

目录

一 引文

性能是移动端图像处理中的一个关键挑战。给定一个参考成像管线，甚至是人类调整的图像对，我们寻求再现增强并实现实时评估。为此，我们引入了一种新的神经网络架构，其灵感来自双边网格处理和局部仿射颜色变换。

使用输入/输出图像对，我们训练卷积神经网络来预测双边空间中局部仿射模型的系数。我们的架构学习做出局部、全局和内容相关的决策，以近似所需的图像转换。在运行时，神经网络输入图像的低分辨率版本，在双边空间中产生一组仿射变换，使用新的切片节点以边缘保持的方式对这些变换进行上采样，然后将这些上采样的变换应用于全分辨率图像。

我们的算法在几毫秒内处理智能手机上的高分辨率图像，提供 1080p 分辨率的实时取景器，并在一大类图像算子上与最先进的近似技术相匹配。

与之前的工作不同，我们的模型是根据数据离线训练的，因此不需要在运行时访问原始操作符。这使我们的模型能够学习复杂的、依赖场景的变换，而这些变换没有参考的实现方法，例如人修饰的照片编辑。

当代相机和移动设备产生的图像和视频的高分辨率给图像处理算法带来了巨大的性能压力，需要熟练的程序员进行复杂的代码优化。由于图像增强是主观的，因此通常希望直接从人调整的结果中学习增强模型。为此，我们提出了一种机器学习方法，其中通过深度网络学习参考滤波器、管线甚至主观手动照片调整的效果，该深度网络可以快速评估，并且成本与参考效果的复杂性无关（比如人手调很长时间，决策非常复杂，但是不影响网络学习的性能）。我们专注于不会在空间上扭曲图像或添加新边缘的照片增强。

我们分享了先前工作的动机，即通过使用远程服务器，或者通过处理低分辨率图像，然后使用低分辨率输出来近似高分辨率等效图像，来加速“黑匣子”图像处理操作。对于某些操作，这些方法可以实现较大的加速，但它们存在显著的局限性：基本的图像处理操作必须在一定程度上保持比例不变，并且必须在低分辨率下快速评估。此外，这些技术依赖于显式引用实现的可用性，因此不能用于从人工注释的输入/输出对的数据库中学习隐式定义的操作。

许多深度学习架构已被用于图像到图像的转换，然而，大多数先前的工作都会产生随着输入图像的大小线性缩放的沉重计算成本，这通常是因为必须以全分辨率评估大量的堆叠卷积和非线性。这种通用形式允许学习灵活的模型，但这种表现力是有代价的：这种架构对于实时取景器应用来说太慢了几个数量级，在最好的桌面 GPU 上处理 100 万像素的图像需要几秒钟的时间，比我们提出的模型慢 1000 倍多（GPU 上 2 毫秒）。我们的加速是通过专门针对摄影变换 (photographic transformations) 实现的，这些变换通常在双边空间中用线性运算很好地近似，并相应地学习我们在该空间中的模型。

我们提出了一种新的网络架构，它能够学习各种各样的照片图像增强，并可以在高分辨率输入上快速评估。我们通过三个关键策略来实现这一点：

1）我们在低分辨率的双边网格中执行大多数预测，其中每个像素的 x，y 坐标用作为像素颜色函数的第三维来扩充。为此，我们引入了一个用于深度学习的新节点，该节点执行数据相关查找。这实现了所谓的切片操作，即通过考虑每个像素的输入颜色以及其 x、y 位置，从 $3D$ 双边网格以全图像分辨率重建输出图像。

2）我们遵循了之前的工作，该工作观察到，预测从输入到输出的转变通常比直接预测输出更简单。这就是为什么我们的架构被设计为学习作为中间表示的局部仿射颜色变换，该变换将通过新的乘法节点应用于输入。

3）虽然我们的大多数学习和推理都是在低分辨率下进行的，但训练过程中使用的损失函数是在全分辨率下评估的，这导致我们学习的低分辨率变换直接针对其对高分辨率图像的影响进行优化。

总之，这三种策略（切片、仿射颜色变换和全分辨率损失）允许我们以低分辨率执行大部分处理（从而节省大量计算成本），同时再现参考算子的高频行为。我们在 7 个应用程序的基准上展示了我们模型的表现力，包括：近似已发布的图像过滤器，逆向工程黑盒 Photoshop 操作，以及从一组手动校正的照片中学习摄影师的修饰风格。

我们的技术产生的输出质量与之前的工作相当或更好，同时通过不需要近似的图像操作的一些参考实现、从输入/输出图像对中端到端可学习以及在移动硬件上实时运行而更广泛地应用。我们网络的前向传输需要 14 毫秒在谷歌像素手机上处理 1920×1080 分辨率的全屏图像，从而实现 50 赫兹的实时取景器效

果。

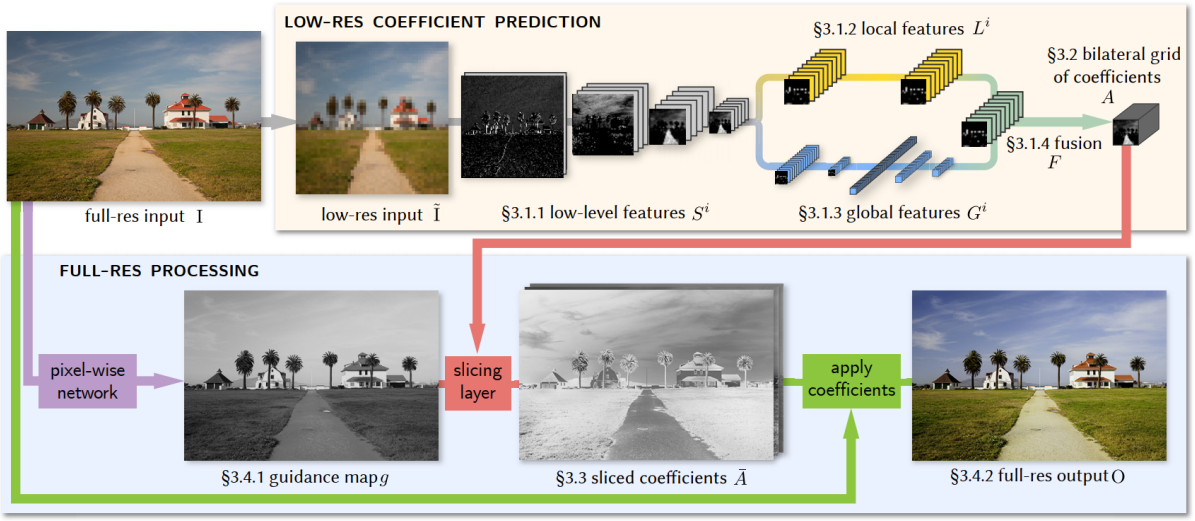
二　相关工作

以前，图像增强的加速操作的一种方法是简单地以低分辨率应用它并对结果进行上采样。简单的上采样通常会导致不可接受的模糊输出，但通常可以通过使用尊重原始图像边缘的更复杂的上采样技术来改善这一问题。比如，联合双边上采样通过在高分辨率引导图上使用双边滤波器来产生分段平滑边缘感知上采样。双边空间优化通过解决双边网格内的紧凑优化问题，产生最大程度平滑的上采样结果，以此为基础。

Gharbi 等人专注于学习从输入到输出的转换，而不是输出本身。它们通过一组简单的局部模型来近似一大类复杂的空间变化算子——这是一种针对给定输入/输出对量身定制的变换配方 (recipe)。算子和拟合配方的任务被卸载到云端，而移动设备只需要应用配方，从而节省时间和精力。类似地，Chen 等人在双边空间中用局部仿射模型网格近似图像算子，其参数以类似于引导滤波器的方式拟合到输入/输出对。通过在低分辨率图像对上执行该模型拟合，该技术实现了设备上的实时计算。我们建立在这种双边空间表示的基础上，但我们构建了一个丰富的类似 CNN 的模型，该模型被训练为将算子应用于任何看不见的输入，而不是拟合一个模型来从一对图像中近似算子的单个实例。这绕过了在运行时对原始算子的需求，并为学习非算法变换（即手动调整的输入/输出图像对）提供了机会。这也使我们能够优化仿射系数，以对以全分辨率运行的算子进行建模，这对于随尺度变化的滤波器来说很重要。

我们的模型可以被训练来自动校正来自人工修饰提供的输入/输出图像对的照片。这是 Bychkovsky 等人提出的任务，他们估计了 5 名训练有素的摄影师的个人风格的全局亮度/对比度调整。他们用手工制作的特征训练回归模型，这些特征在 5000 张原始图像的数据集上捕捉低级信息和语义内容（例如人脸）。Hwang 等人通过从粗到细搜索最佳匹配场景来解决这个问题，这对于 500×333 的图像来说需要一分钟以上的时间。考夫曼等人从硬编码特征（人脸、蓝天、云层、曝光不足区域）学习局部颜色和对比度操作，VGA 图像运行时间超过 2 分钟。最近，Yan 等人使用了紧凑的像素神经网络和手工制作的特征。他们的网络处理 100 万像素的图像需要 1.5 秒（除了物体检测、密集图像分割和特征中使用的场景识别所需的时间之外）。我们的模型可以学习类似的全局色调调整，并推广到更复杂的效果，包括颜色校正和局部编辑，此外速度要快得多。

三　架构设计



上图第一行是用于低分辨率处理的网络，第二行是原分辨率处理的网络。

在第一行中，首先先降分辨率得到 $\tilde{I}$，然后用卷积网络来学习特征 S^i（也用于降低分辨率），然后分到两个网络，上面的 L^i 只有卷积网络，用于学习局部特征，下面的 G^i 有卷积和全连接网络，用于学习全局特征，然后两种特征混合到一起得到 F。之后使用一个逐点线性层得到 A，A 可以理解为双边网格的仿射系数。因为这个双边网格是根据学习来得到的，所以叫做“learned splatting”。

得到的 A 是一个 $16*16*8$ 的 feature map（双边网格），每个网格包含了 12 个数字，是 $3*4$ 仿射颜色变换矩阵。从某种意义上说，在整个过程中保持 2D 卷积公式，并仅将最后一层解释为双边网格（并不是真的双边网格，只是解释为双边网格），也就是说，其实相当于我们让网络决定何时 2D 到 3D 转换是最佳的。

如果 A 不作用到网络中就没有意义，难点在于如何将低分辨率网格作用到原图中，这也算是本文一个很大的创新。本文介绍了一种基于双边网格 slicing 操作的层（splatting 和 slicing 是双边网格算法最基本的步骤），该层将单通道引导图 g 和空间分辨率远低于引导图的特征图 A（视为双边网格）作为输入，它在最终的特征图 A 中执行依赖于数据的查找。该层对于 A 和 g 都是底层可微的，这使我们能够在训练时通过它进行反向传播。

slicing 操作的结果是得到跟 g 一样分辨率的新特征图 $\overline{A}$，这是通过在由 g 定义的位置对 A 进行三线性插值来得到的。这个公式对理解整个架构非常有用，所以就重点了解一下论文里的公式 (5)：

$$\overline{A}_c[x,y] = \sum_{i,j,k} \tau(s_x x - i)\tau(s_y y - j)\tau(d \cdot g[x,y] - k)A_c[i,j,k] \tag{三.1}$$

s_x 和 s_y 分别是网格在长宽的维度比例，$x \in [0, W-1]$，$y \in [0, H-1]$，$[W, H]$ 是原图的分辨率。

我们假设原图长宽分别是 $512*512$，然后求 $\overline{A}_c[121, 145]$，此时，s_x 和 s_y 分别是 $\frac{16}{512} = 0.03125$ 和 $\frac{16}{512} = 0.03125$，$s_x x$ 和 $s_y y$ 分别是 3.78125 和 4.53125。$g[x,y]$ 是灰度尺度值，范围在 $[0,1]$ 之间。

最后输出的还是 12 通道的 $\overline{A}_c[x,y]$。

由于 slicing 操作是无参数的，在训练好之后的实际运行中，可以在 OpenGL 着色器中有效地实现。它充当了将神经网络的表示约束到低维空间的瓶颈层。这既简化了学习问题，又加快了处理时间。至关重要的是，在双边网格内进行推理会迫使我们的模型的预测遵循 g 中的边缘，从而使我们的预测规则化为边缘感知解决方案（与基于转置卷积或“去卷积层”的标准网络不同）。这种设计决策往往有利于像我们这样的摄影操作任务，并且由于 A 的低维性，使我们能够显著加快比更通用的模型的速度。

g 的获取在论文 3.4.1 节提到，是原图做通道变换为单通道引导图（这个过程也是网络学习学到的）。

当求出 $\overline{A}_c[x,y]$ 以后，使用该值和原图做线性变换，得到仿射变换以后的结果。损失函数就是该结果和目标结果的 L_2 距离之和。

四　疑问

关于这个架构，我有一点不是很理解，就是为什么矩阵可以进行插值。对于几何变换来说，直接对两个矩阵之间的值插值，无法得到准确的变换过渡（尤其是对于旋转矩阵）；而且，如果一个较大的区域的很多像素都有比较相似的仿射矩阵，总觉得会影响效果。

参考文献

[1] xxx