deblocking

Part 1 Related Work

1. ARCNN

参考论文：http://de.arxiv.org/pdf/1504.06993
模型代码：https://github.com/yydlmzyz/deblocking/tree/master/ARCNN
模型结构：

Layer	conv1	conv2	conv3	conv4
FilterSize	9*9	7*7	1*1	5*5
channel	64	32	16	1(3)

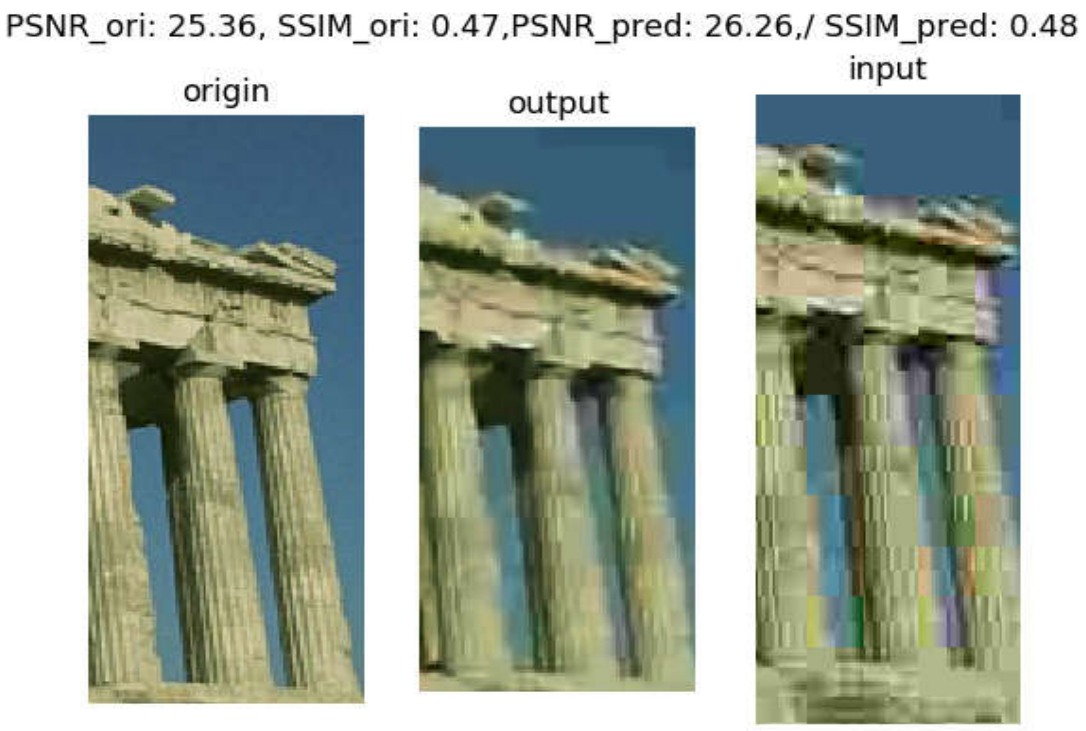
模型原理：Layer1 是进行特征提取，Layer2 进行特征增强（去噪），Layer3 非线性映射（？），Layer4 重建（将 Layer3 的一组特征图进行类似滤波操作？）
展示：

Layer1:4/64 feature extract

Layer2:4/32 feature enhancement

Layer3:4/32non-liner mapping

Result:



论文中可以将 Q10 的图像 PSNR 提高 1.2dB,实验对上图提高了 0.9dB

2 . L8

参考论文：https://arxiv.org/abs/1605.00366
模型结构：https://github.com/yydlmzyz/deblocking/tree/master/L8

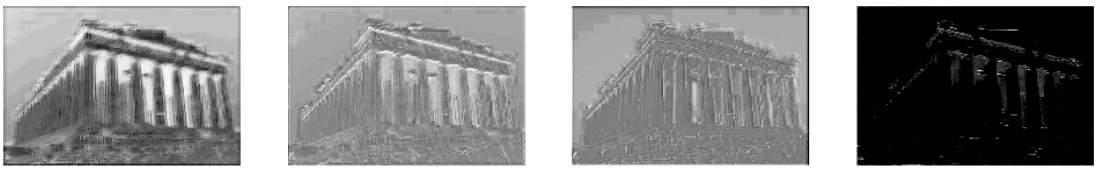
Layer	1	2	3	4（+1）	5	6（+1）	7	8
FilterSize	11*11	3*3	3*3	3*3	1*1	5*5	1*1	5*5
Channel	32	64	64	64（+32）	64	64（+32）	128	3（1）

共 8 个卷积层，特色是用了 Residual objective，用 skip architecture 将第 1 层的输出串接（concatenate）到第 4 层和第 6 层。目的是为了在 input 和 output 之间更容易的传递几何信息，同时使中间层可以更复杂，处理更多信息。

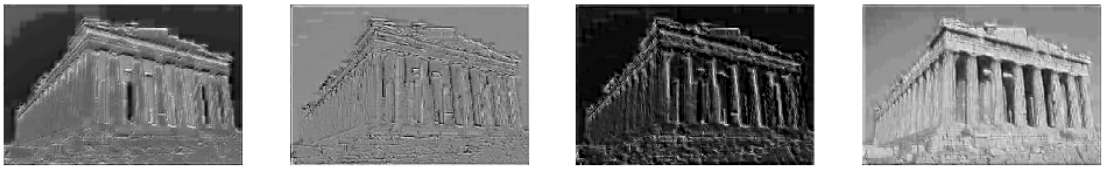
它没有使用 Batch Normalization，因为它认为尽管 Batch Normalization 可以加速收敛、控制过拟合，允许使用更大 learning rate，但是会在训练中会引入噪声。为此它在初始化上进行了改进（？）

Layer1:

Layer2:



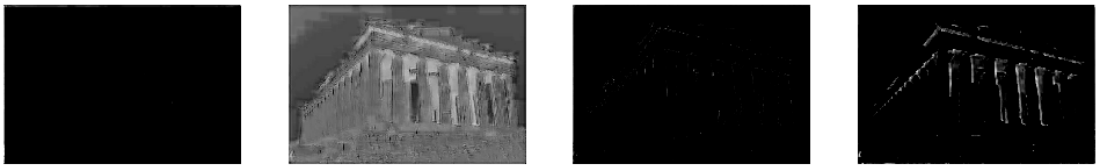
Layer3:



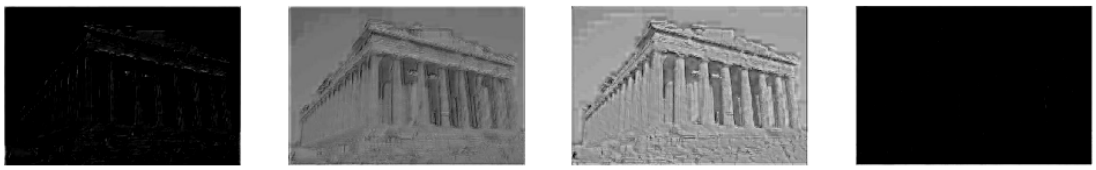
Layer4:

Layer5:

Layer6:



Layer7:



Result：

论文中可以将 Q10 的 PSNR 提高约 1.2dB，实验之后提高了 1.15dB

3.CAS-CNN

参考论文：https://arxiv.org/pdf/1611.07233.pdf

代码：（尚未实现）？

模型结构：

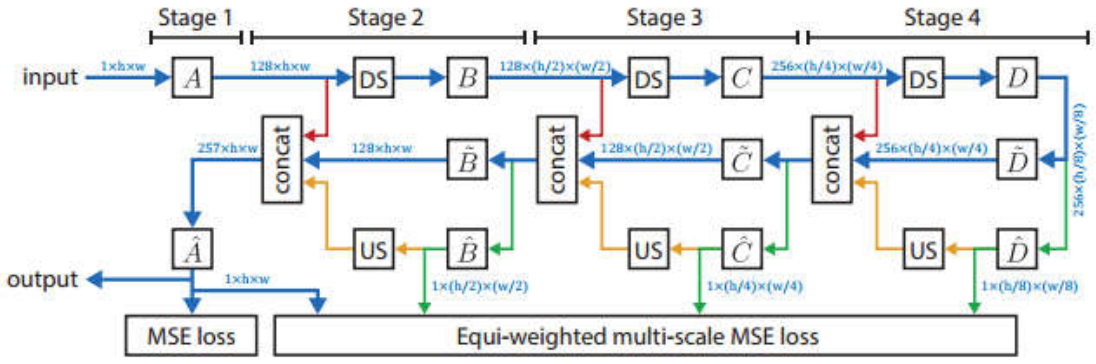


Fig. 1: Structure of the proposed ConvNet. The paths are color coded: main path (bold), concatenation of lower-level features, multi-scale output paths, re-use of multi-scale outputs.

1. main path 中 A、B、C、D 都是由两层 Conv(3*3)组成的 block，channel 分别为 128、128、256、256；同时在 A、B、C、D 每两个 block 中间插入一个 Down Sampling 层（用 average-pooling（2*2）实现）逐步将分辨率降到(h/8 ,w/8)；然后是 3 个反卷积层（full-convolution/deconvolution/Conv2DTranspose）D` C` B`，（kernel size 是（4*4/2）？，输出 channel 分别是 256、128、128），最后是一个 3*3 的卷积层 A`，输出最终结果。（图中蓝色路径）

2. 由于 main path 共 12 个卷积层，比较深，给训练上带来了困难，为了减少平均路径

长度，论文中提出了一种办法：在每个 full-convolution D` C` B`后，将前面 A B C 3 个 block 提取到的低层特征（low_level feature）串接（concatenate），传递到下一层，好处有两个，一是达到增强特征的目的，同时也为中间层提供 bypassing，使其离输入输出更近，也更方便训练；（图中红色路径）

3.尽管 2 中提供的红色 bypassing 可以缩短路径长度，如 A->A`->output,但是对于更深一些的中间层，比如 D,路径还是比较长，导致梯度反向经过 A` B` C`传播到 D`时，可能产生梯度消失问题（vanishing gradient problem）？，导致难以优化，所以文中又提出了多尺度的优化标准（multi-scale optimization criterion），就是不仅比较 input、output，还重建了 3 个低分辨率的输出 B^ C^ D^（分辨率分别为 h/2*w/2 h/4*w/4 h/8*w/8），来计算损失（Equi-weighted multi-scale MSE loss），与从最终 output 产生的 Loss 一起用来优化（图中绿色路径）。

4.同时还将 3 中的输出再经过上采样（upsampling）放大后，再串接（concatenate）到 main path 中，进一步缩短路径。（图中棕色路径）

实验效果：

论文中显示其效果很好，可以将 Q10 的图像 PSNR 提高约 1.67dB。但是这种模型，参数很大，论文中说明它共有约 500 万个参数，计算量会很大。

4. SRResNet

参考论文：https://arxiv.org/abs/1609.04802v1

代码：https://github.com/yydlmzyz/deblocking/tree/master/SRResNet_original （？）

模型结构：

ResidualBlock	conv1	BN1	relu	Conv2	BN2	add
FilterSize	3*3	/	/	3*3	/	/
Channel	64	64	64	64	64	64

upscaleBlock	Conv1	UpSample1	Relu1	Conv2	upSample2	relu
FilterSize/size	3*3	(2,2)	/	3*3	(2,2)	/
Chnnel	128	128	128	128	128	128

Layer	Conv1	relu	ResidualBlock(5)	Conv2	BN1	Add(conv1)
FilterSize	9*9	/	/	3*3	/	/
Channel	64	64	64	64	64	64
Layer	upscaleBlock	Conv3				
FilterSize	/	9*9				
Channel	128	3				

使用 5 个 residual block，1 个 upscale block 将图像放大 4*4 倍，共 15 个卷积层，2 个上采样层，在 residual block 中共采用了 11 次 Batch Normalization，用 relu 激活。

(疑问：关于 upscale block，论文中是 pixel shuffle，代码中有的使用 upsampling，有的用 deconvolution，有的用 pytorch 中的 pixel shuffle，这些有什么区别？)

Result：

这个模型我没有训练，根据论文中数据：对于 Q10 的图像 PSNR 可以提高 2.56dB,根据其他人的测试，只提高了 1.2dB

Part 2　My Work

1.SRResNet_deep

代码：https://github.com/yydlmzyz/deblocking/tree/master/SRResNet_complex
模型结构：
保留了 SRResNet 的 Residual Block，舍弃 UpscaleBlock，用一个普通的卷积层代替，对于从第一个 conv1 特征提取层到最后一个 Add(conv1)之间的 skip connection 也舍弃了

Layer	Conv1	relu	ResidualBlock(5)	Conv2	BN1	Conv3	Conv4
FilterSize	9*9	/	/	3*3	/	3*3	9*9
Channel	64	64	64	64	64	64	3(1)

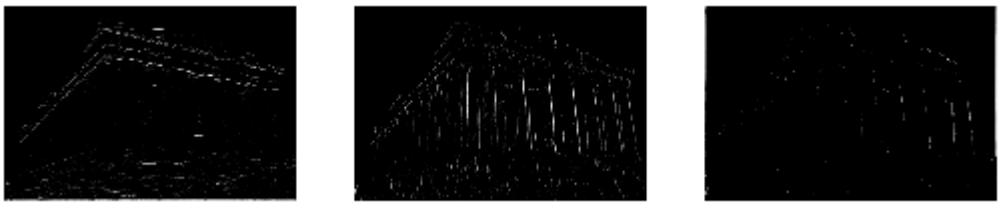
Layer1:3/64

Layer8:BN 4/64

Layer9:add_1 4/64

Layer33:add_5 3/64

Layer36 conv 4/64



1.从 Layer8 和 Layer9 之间对比看，Batch Normalization 的作用并不明显

2.随着 residual block 的增加，提取到的特征变少

从图中可以看出，随着层数的增加，各层提取的特征逐渐变少，另外很多特征都比较相似，说明网络层数的加深可以提取到更多的信息，从而相比浅层网络获得更好的效果。残差层提取的效果看起来好像差不多。

Result：

2.SRResNet_simple

代码地址：https://github.com/yydlmzyz/deblocking/tree/master/SRResNet_simple

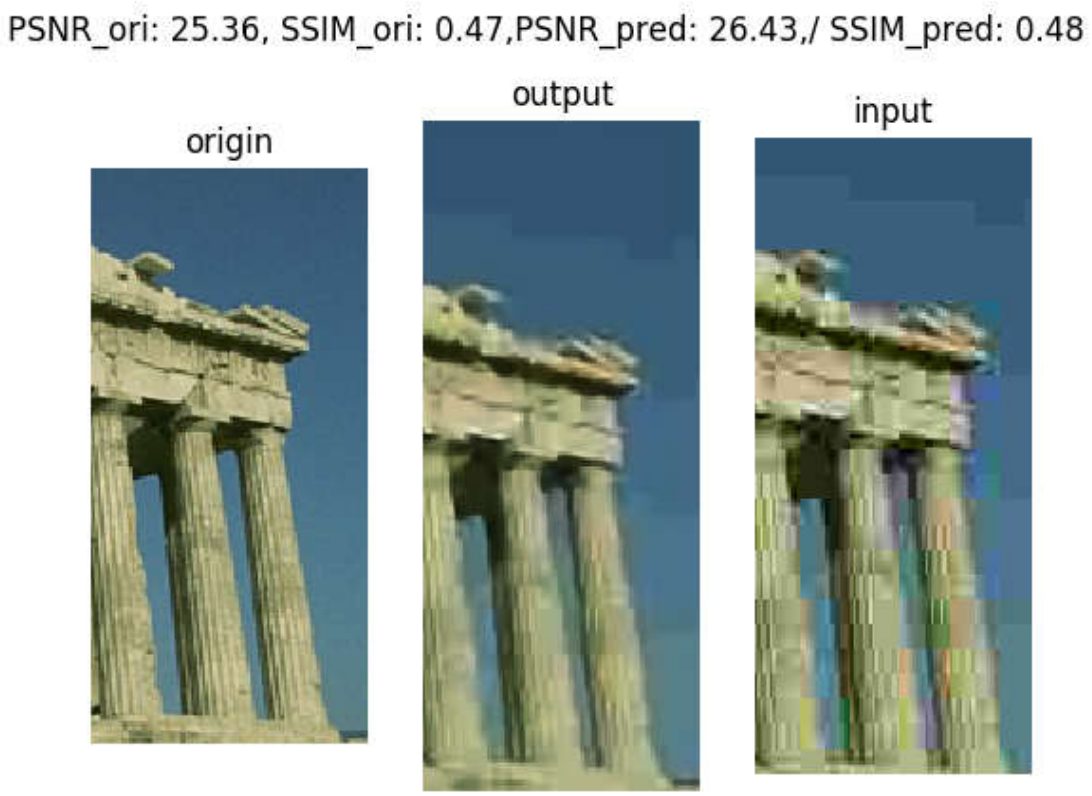
模型结构：

相比于前面的 SRResNet_deep，进行了简化，5 个 residual block 减少为 3 个，加上了从第一层到第三个 block 之间的 skip，在输出端增加了 128channel，1*1 的卷积层

Layer	Conv1	Residual block(3)	Conv2	Add(conv1)	Conv3	Conv4
FilterSize	9*9	/	3*3	/	1*1	9*9
channel	64	64	64	64	128	3(1)

实验发现，虽然简化后可以减少计算量，但是去块能力也变差了。

Result：



Part 3 summary and discussion

1.experiment

GPU:GeForce GTX 1080 1.74GHz 8G

DataSet：BSD500 中 204 张 RGB 图片，都转换成 YCbCr 模式，以 16 的步长分割，生成（92534,42, 42, 3）的数据集

Metrics：PSNR、SSIM（？）

Loss Function：MSE

Model	JPEG	ARCNN	L8	CAS-CNN	SRResNet-deep	SRResNet-simple
Loss	/	/	1024e-3	/	1039e-3	1049e-3
PSNR/dB	25.36	26.26	26.51	27.08(估计)	26.52	26.43
Params	/	117K	220K	5144K	480K	315K
TrainTime		39s*?	93s*24	/	266s*50	167s*50

总结：兼顾去块效果和计算量，L8 最好

Conclusion

1.H.264 去块滤波：

块效应的产生主要是因为变换编码、运动补偿、量化损失。变换编码中 DCT 变换后，解码时由于运算精度造成误差；运动补偿中块的匹配不精确，复制后在边缘出现不连续，而且原来块中的不连续边界复制到下一帧中，产生累积误差；量化如果过粗，高频细节会受到损失，相邻块之间相关性减弱，产生块效应（？）

环路滤波：相比较于后处理滤波器在解码端进行滤波。在编码环中引入去块滤波器，在编码过程中对每帧进行滤波，完成后用于下一帧的参考，可以避免误差积累。

自适应滤波：对于图像（宏块？）不同的位置，滤波的强度也不同，要有选择的平滑误差。例如，如果块边界与块内像素绝对差值相对比较大，出现块效应的可能性就大，需要进行滤波，如果幅度很大，则反映了真实边界，不需要滤波。针对不同位置的像素值，滤波轻度也要不同。

2.模型结构的选择

结合 H.264 和前述论文，用神经网络进行去块也是遵循首先提取低级特征，然后增强特征，找到和块边界、图像细节等有关的特征，最后根据提取到的特征映射重建特征。

从不同模型的比较中反映出网络越深，处理能力越强。

但是网络加深后，不仅运算量变大，而且由于梯度减弱消失，给训练带来了困难。为了解决这个问题，这些模型都提出了不同的解决方案，它们的共同特点都是采用 skip connection 来跳过若干层，连接到更后面的层。

不同点是 SRResNet 用到的 ResNet 将相同 shape 的参数直接（add）加到后面的网络层，这样在输入输出相似的情况下，只用学习残差，训练时会变容易；而 L8 和 CAS-CNN 则是将前面的层串接（concentenate）到后面的层，这样既可以使中间层里路径长度变短，还可以使中间层变复杂。

由于输入输出图像分辨率相同，ARCNN、L8、SRResNet 并没有使用池化、上采样、反卷积等结构，但是 CAS-CNN 却都使用了，主要是为了得到不同分辨率的输出，以得到 multi-scale 的损失函数，来解决网络较深导致的梯度减弱，难以训练的问题。很值得借鉴。

3.视频的去块

论文中都是对单幅图像进行处理，对于视频处理时效果不好。原因主要在数据集和运动补偿上。

由于视频相邻帧之间区别很小，大部分地方都是相同的，在制作数据集时很多截取的子图片都是几乎相似的，冗余很大，难以训练。

另外由于视频图像的静止部分几乎没有块效应，而运动部分的块效应不仅受量化误差影响，而且由于运动补偿产生块的累积，复原时不仅要平滑块边界，还要消除累积，这个比较困难。

所以借鉴 H.264 中在编码时用环路滤波避免误差积累的方法，如果在编码环路中就进行去块，要比在解码时再去块好。

4.优化方案

1.mian path 保持在 10 个卷积层以内，也可以产生比较好的效果。

2.在 skip connection 上进行优化，除了论文中的 concatenate 和 ResNet，DensNet 也可能有用