ESRGAN-进击的超分辨率复原

标签（空格分隔）： 陈扬

github:https://github.com/xinntao/ESRGAN、https://github.com/xinntao/BasicS

arxiv:https://arxiv.org/pdf/1809.00219v2.pdf

当超分辨率复原碰上GAN

前言

> **超分辨率成像**（Super-resolution imaging，缩写SR），是一种提高影片分辨率的技术。在一些称为“光学SR”的SR技术中，系统的衍射极限被超越；而在其他所谓的“几何SR”中，数位感光元件的分辨率因而提高。超分辨率成像技术用于一般图像处理和超高分辨率显微镜。

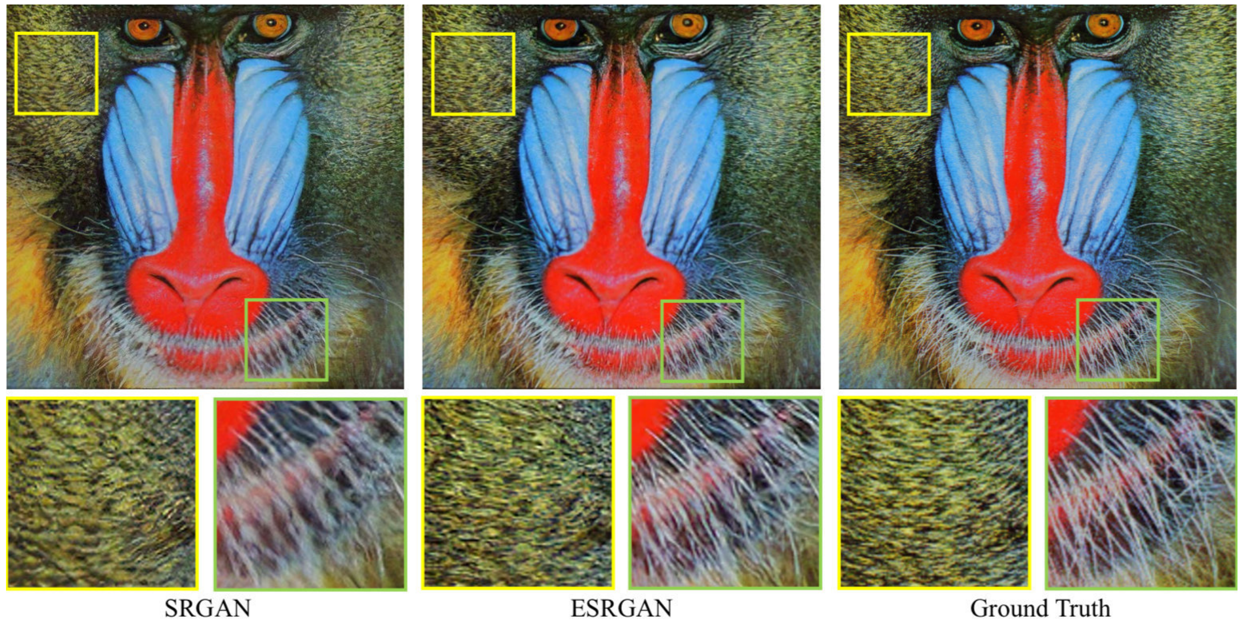
> **生成对抗网络**（英语：**G**enerative **A**dversarial **N**etwork，简称GAN）是非监督式学习的一种方法，通过让两个神经网络相互博弈的方式进行学习。该方法由伊恩·古德费洛等人于2014年提出。[1]

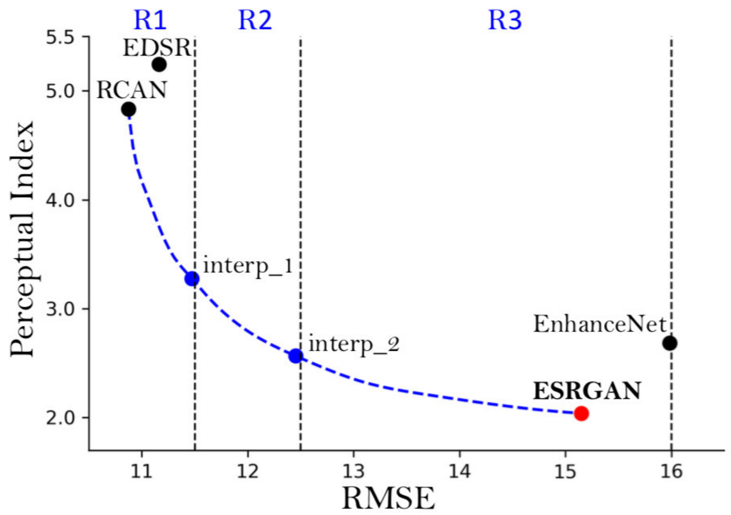
直入主题

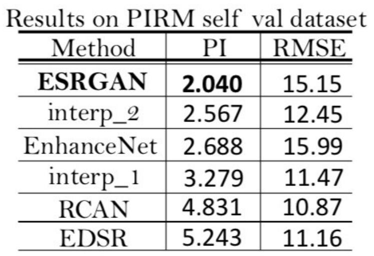
ESRGAN的全名叫 **Enhanced Super-Resolution Generative Adversarial Networks**,发表于ECCV2018,它是基于SRGAN改进而来到,相比于SRGAN它在三个方面进行了改进

1. 改进了网络结构,对抗损失,感知损失
2. 引入**Residual-in-Residu Dense Block**(RRDB)
3. 使用激活前的VGG特征来改善感知损失

在开始讲这个ESRGAN的具体实现之前,我先来看一下他和他的前辈SRGAN的对比效果:






Results on PIRM self_val dataset

Method	PI	RMSE
ESRGAN	**2.040**	15.15
interp_2	2.567	12.45
EnhanceNet	2.688	15.99
interp_1	3.279	11.47
RCAN	4.831	10.87
EDSR	5.243	11.16

我们可以从上图看出,

1. ESRGAN在锐度和边缘信息上优于SRGAN,且去除了"伪影"
2. 从PI和PMSE两个指标来看,ESRGAN也可以当之无愧地称得上是超分辨率复原任务中的the State-of-the-Art

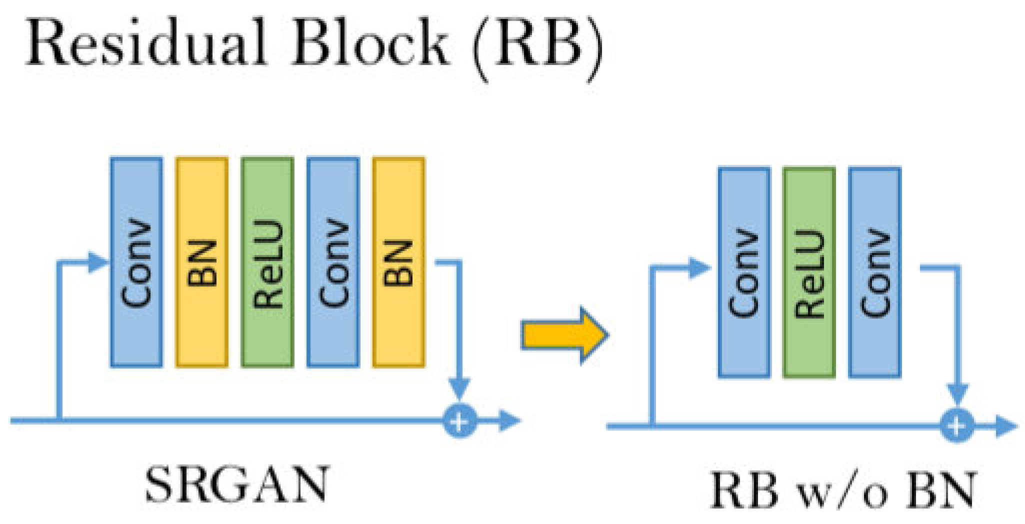
庖丁解牛

RRDB,对residual blocks的改进:

我们可以看出这个残差块是很传统的Conv-BN-relu-Conv-BN的结构,而作者在文章中是这么说到的:

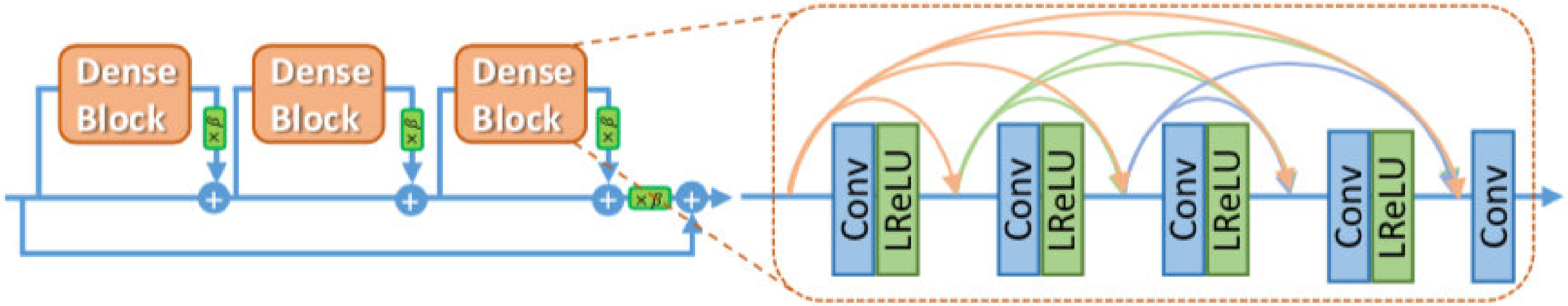
> We empirically observe that BN layers tend to bring artifacts. These artifacts,namely BN artifacts, occasionally appear among iterations and different settings,violating the needs for a stable performance over training. In this section, wepresent that the network depth, BN position, training dataset and training loss have impact on the occurrence of BN artifacts and show corresponding visual

什么意思呢?就是说作者认为SRGAN之所以会产生伪影,就是因为使用了Batchnormalization,所以作者做出了去除BN的改进



而且我们再来看,SRGAN的残差块是顺序连接的,而作者可能哎,受denseNet的启发,他就把这些残差块用密集连接的方式连在一起.那么他的生成器里的特征提取部分最终变成了这样子:

Residual in Residual Dense Block (RRDB)



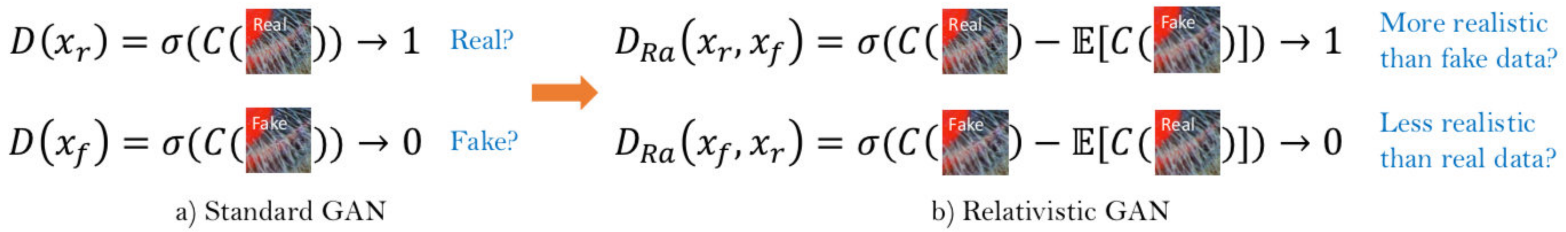
既然我们知道他的网络结构是这样子设计的,那么他的实现其实就很简单:

```
class RRDB_Net(nn.Module):
    def __init__(self, in_nc, out_nc, nf, nb, gc=32, upsca
            mode='CNA', res_scale=1, upsample_mode='upconv
        super(RRDB_Net, self).__init__()
        n_upscale = int(math.log(upscale, 2))
        if upscale == 3:
            n_upscale = 1
        fea_conv = B.conv_block(in_nc, nf, kernel_size=3,
        rb_blocks = [B.RRDB(nf, kernel_size=3, gc=32, str
            norm_type=norm_type, act_type=act_type, mode=
        LR_conv = B.conv_block(nf, nf, kernel_size=3, norm
        if upsample_mode == 'upconv':
            upsample_block = B.upconv_blcok
        elif upsample_mode == 'pixelshuffle':
            upsample_block = B.pixelshuffle_block
        else:
            raise NotImplementedError('upsample mode [%s]
        if upscale == 3:
            upsampler = upsample_block(nf, nf, 3, act_type
        else:
            upsampler = [upsample_block(nf, nf, act_type=
        HR_conv0 = B.conv_block(nf, nf, kernel_size=3, no
        HR_conv1 = B.conv_block(nf, out_nc, kernel_size=3
        self.model = B.sequential(fea_conv, B.ShortcutBloc
            *upsampler, HR_conv0, HR_conv1)
    def forward(self, x):
        x = self.model(x)
```

对损失函数的改进

说到损失函数啊,我们之前在SRGAN的文章里我也介绍过,这个判别器它判断的是你输入的图片是"真的"高清图像,还是"假的"高清图像,而且作者他就提出一种新的思考模式,就是说我的判别器是来**估计真实图像相对来说比fake图像更逼真的概率。**

怎么来理解这句话呢?


$D(x_r) = \sigma(C(\text{Real})) \to 1$
Real?
$D(x_f) = \sigma(C(\text{Fake})) \to 0$
Fake?
a) Standard GAN
$D_{Ra}(x_r, x_f) = \sigma(C(\text{Real}) - \mathbb{E}[C(\text{Fake})]) \to 1$
More realistic than fake data?
$D_{Ra}(x_f, x_r) = \sigma(C(\text{Fake}) - \mathbb{E}[C(\text{Real})]) \to 0$
Less realistic than real data?
b) Relativistic GAN

具体而言，作者把标准的判别器换成Relativistic average Discriminator（RaD），所以判别器的损失函数定义为：

$$L_D^{Ra} = -\mathbb{E}_{x_r}[\log(D_{Ra}(x_r, x_f))] - \mathbb{E}_{x_f}[\log(1 - D_{Ra}(x_f, x_r))]. \qquad (1)$$

对应的生成器的对抗损失函数为：

$$L_G^{Ra} = -\mathbb{E}_{x_r}[\log(1 - D_{Ra}(x_r, x_f))] - \mathbb{E}_{x_f}[\log(D_{Ra}(x_f, x_r))], \qquad (2)$$

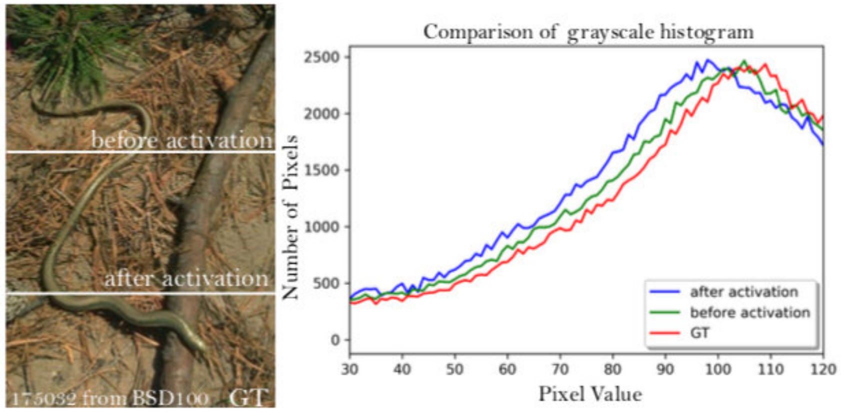
```python
# Extract validity predictions from discriminator
pred_real = discriminator(imgs_hr).detach()
pred_fake = discriminator(gen_hr)
# Adversarial loss (relativistic average GAN)
loss_GAN = criterion_GAN(pred_fake - \
        pred_real.mean(0, keepdim=True), valid)

pred_real = discriminator(imgs_hr)
pred_fake = discriminator(gen_hr.detach())
# Adversarial loss for real and fake images (relativistic
loss_real = criterion_GAN(pred_real - \
        pred_fake.mean(0, keepdim=True), valid)
loss_fake = criterion_GAN(pred_fake - \
        pred_real.mean(0, keepdim=True), fake)
# Total loss
loss_D = (loss_real + loss_fake) / 2
```

对感知损失的改进

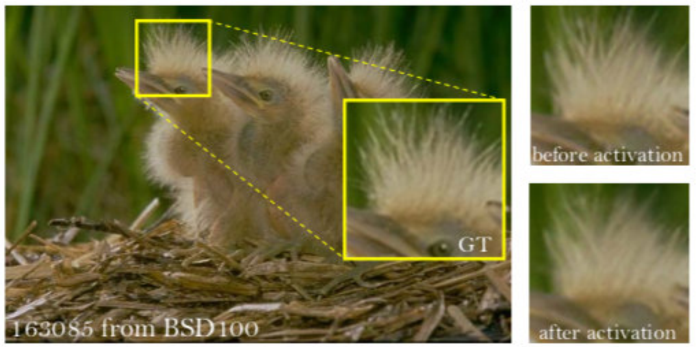
我们之前看SRGAN的时候看到哎,它是用来一个训练好的VGG16来给出超分辨率复原所需要的特征,作者通过对损失域的研究发现,激活前的特征，这样会克服两个缺点。

> Perceptual loss is previously defined on the activation layers of a pre-trained deep network, where the distance between two activated features is minimized. Contrary to the convention, we propose to use features before the activation layers, which will overcome two drawbacks of the original design.

1. 激活后的特征是非常稀疏的，特别是在很深的网络中。这种稀疏的激活提供的监督效果是很弱的，会造成性能低下；
2. 使用激活后的特征会导致重建图像与GT的亮度不一致。



(a) brightness influence



(b) detail influence

与此同时,作者还在loss函数中加入了$L_1 = E_{xi}||G(x_i) - y||_1$,也就是$L_1$损失,最终损失函数由三部分组成:

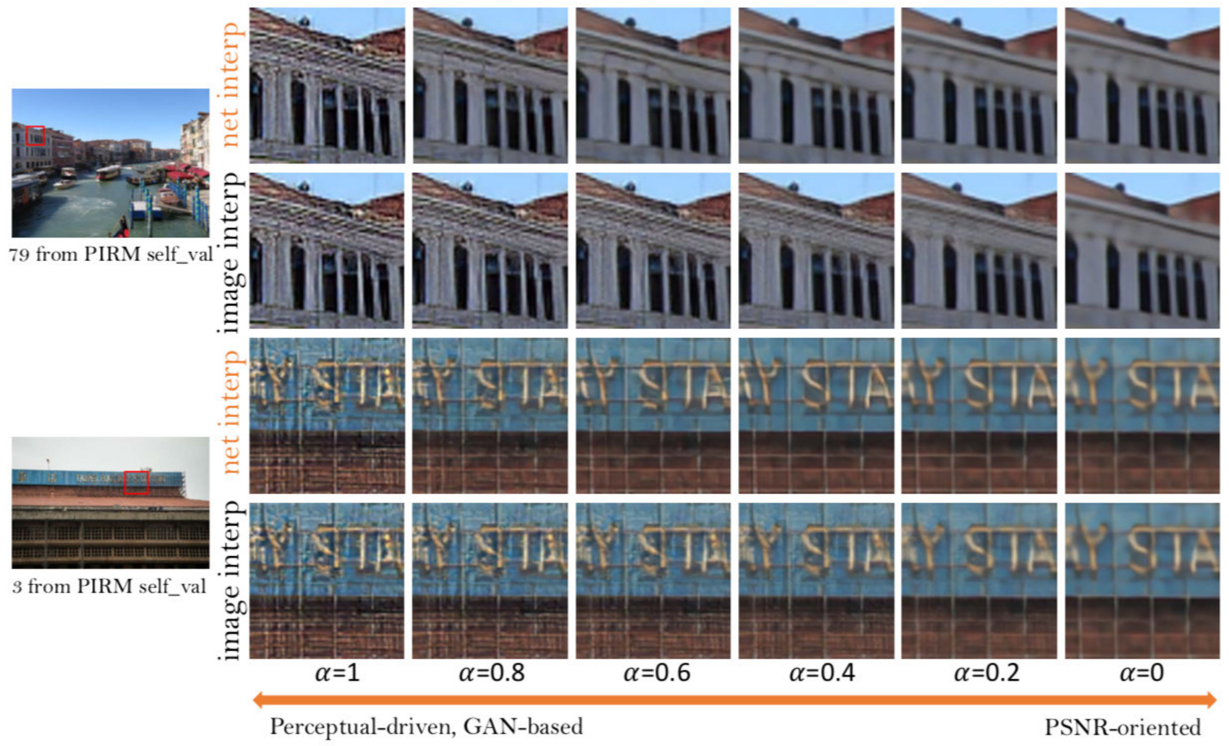
$$L_G = L_{\text{percep}} + \lambda L_G^{Ra} + \eta L_1, \tag{3}$$

```
        loss_G = loss_content + \
                opt.lambda_adv * loss_GAN + \
                opt.lambda_pixel * loss_pixel
```

网络插值

为了平衡感知质量和PSNR等评价值，作者提出了一个灵活且有效的方法---网络插值。具体而言，作者首先基于PSNR方法训练的得到的网络G_PSNR，然后再用基于GAN的网络G_GAN进行整合

$$\theta_G^{\text{INTERP}} = (1-\alpha)\ \theta_G^{\text{PSNR}} + \alpha\ \theta_G^{\text{GAN}}, \qquad (4)$$



它的具体实现就是:

```
for k, v_PSNR in net_PSNR.items():
    v_ESRGAN = net_ESRGAN[k]
    net_interp[k] = (1 - alpha) * v_PSNR + \
                    alpha * v_ESRGAN
```

超参数部分

ESRGAN可以实现放大4倍的效果

首先要训练一个基于PSRN指标的模型,如何根据这个模型的权重进行生成器的初始化.

作者使用了Adam作为优化器(β1 = 0.9, β2 = 0.999.)进行交替训练.

生成器有16个residual block和23个RRDB

总结:

文章提出的ESRGAN在SRGAN的基础上做出了改进,在大的框架上没有改动,但是通过各种各样的小技巧,也确实有不错的效果提升,而且很重要滴是作者很良心滴开源了代码,复现的要求也不是很难.