重新思考图像超分辨率数据增强:综合分析与新策略

一. 阐述思路

1. 提出问题:

平常，在视觉效果的处理上，我们都是通过数据增强来应用于高级视觉任务（如人脸识别等分类任务），而对于低级视觉任务（比如图像恢复等），数据增强方式可能并不适用。出现此现象，原因在于：图像恢复需要保证原图像的空间关系，而在日常处理图像的过程中，为了图像的分辨率等因素，我们往往会对图像进行加工，而在此过程中，不可避免地我们会对图像的空间关系破坏。因此，本文将对超分辨率图像处理方式进行思考，如何正确运用超分辨率。

2. 五个模块分析:

(1) 数据增强（DA）介绍

a. 数据增强(DA)是在测试阶段提高模型性能而不增加计算成本的最实用的方法之一。如今的研究主要侧重多个高水平视觉任务的研究，低水平的DA分析研究较少。对于多数图像恢复研究，大多数基于一种合成数据集（$DIV2K$）。但是使用合成数据集和使用真是数据集毕竟不是同一件事。同时，部分人提出收集真实世界的数据来为DA分析服务，无疑这件事是消耗时间和非常昂贵的，因此，我们需要考虑其他方式处理。

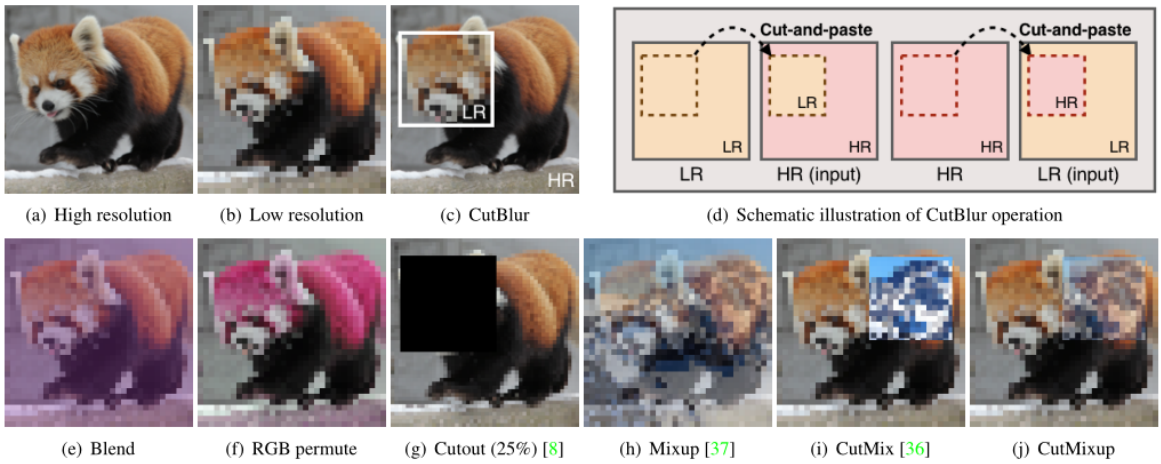
b. 在DA上，部分学者对其进行简单的图像处理，并没有深入使用其中的技术，典型人物如下：

① $Radu$等人第一个研究各种技术来提高基于示例的单图像超分辨率($SISR$)的性能，其中就使用了数据增强。简单来说，他们通过对图像的旋转和翻转，对模型和数据集进行一致性的改进。具体改进时，使用传统的SR模型和基于学习的简单$SRCNN$进行几何操作的研究。（这里提供一个$SRCNN$的介绍文章https://blog.csdn.net/xu_fu_yong/article/details/96481490）

② 最近对DA方法进行研究的是$Feng$等人，然而他们所做的工作也是有限的，仅仅使用一个单一的U-Net-like的架构以及单一的数据集测试DA方法。

c. 首先对高水平视觉DA进行分析。如今的数据增强方法可以分为两种：像素域和特征域。像素域是通过区域属性（例如灰度特征、或者频域分析等）比较其相似性。特征域则是建立两幅图像中特征点之间对应关系的过程。很明显，像素域的计算量明显大于特征域。但当我们使用这两种方法时，将会对源图像的空间信息丢失与混淆。但当我们使用基本操作，如RGB（颜色）排列，则不会造成这些情况。

d. 基于以上分析，文中提出一种新的处理方法：$Cutblur$。分析一下：研究员们从某张图片中扣出一块作为LR图像，将其与另一张图片进行组合训练，机器通过不断模型训练，以达到LR自动寻找到自己所应该在的位置，从而停止训练输出结果。当然，像素不同的情况下，我们需要考虑HR和LR的比例，以达到训练效果。在这种方式下，我们能获得额外的副产品：由于需要通过分辨率来定位图像位置，因此模型会给出每个局部的超分辨率特征，因此其自适应超分辨率图像。



(a) High resolution (b) Low resolution (c) CutBlur (d) Schematic illustration of CutBlur operation

(e) Blend (f) RGB permute (g) Cutout (25%) [8] (h) Mixup [37] (i) CutMix [36] (j) CutMixup

Figure 1. Data augmentation methods. **(Top)** An illustrative example of our proposed method, CutBlur. CutBlur generates an augmented image by cut-and-pasting the low resolution (LR) input image onto the ground-truth high resolution (HR) image region and vice versa (Section 3). **(Bottom)** Illustrative examples of the existing augmentation techniques and a new variation of CutMix and Mixup, CutMixup.

(2) DA分析

本节分析现有的数据增强方法。数据集：$DIV2K$或$RealSR$，模型：EDSR。（使用作者的代码进行分析）

a. 先验技术

① 像素空间中的DA方法：

很多研究在高级视觉视觉任务中增强图像，如下：

$Mixup$:混合两张图片，生成副本

$Cutout$:切掉图像中的某个部分，但这样会导致其无法充分利用数据集，因此我们可以在其空出的区域添加一个额外的图像。

$AutoAugment$:被认为是最佳增强策略，用于学习给定任务和数据集。其能够创建一个数据增强策略的搜索空间，利用搜索算法选取适合特定数据集的数据增强策略。此外，从一个数据集中学到的策略能够很好地迁移到其它相似的数据集上。（https://blog.csdn.net/pwtd_huran/article/details/80868435）

② 特征空间中的DA方法：

以及提出能够操纵CNN特征的DA方法，分为三类：特征混合、抖动、下降。

特征混合：此方法将输入图像和潜在特征混合（例如Java：拉普拉斯锐化、Sobel边缘检测、均值滤波、伽马变换）；

抖动：对特征执行随机仿射变换（其实就是利用双线性插值的方式对图像进行处理）

下降：对多余的特征有选择性的删除，提高模型泛化能力。

③ 超分辨率下的DA方法：

简单几何操作：翻转，旋转。多用于SR模型中。这里提出Mixup可以缓解SR模型的过拟合问题。

b. 现有DA技术分析

在低层次视觉学习中，图像之间的局部和全局关系显得尤为重要，然而DA方法却不重视图像的空间信息，这也限制了DA对图像的恢复能力。

我们通过观察下图，通过不同的方式随机挖去图片的一部分像素，通过输入恢复图象时，性能的降低也会有所不同。例如去除25%的矩形形状内容，其在Cutout下的性能降低%1，然而恢复后，像素 却增加0.01和0.06。这也让我们思考，如果不同的方式混合起来，将会对图像的像素产生何种效果?

下图的虚线下是对不同方法混合的描述，很明显我们发现，Cutout和Mixup混合使用将会有收益相对较大的提高。这也为我们提供了一种可行的方案。从这种方案中，我们提出一种新的方法：CutBlur

Table 1. PSNR (dB) comparison of different data augmentation methods in super-resolution. We report the baseline model (EDSR [18]) performance that is trained on DIV2K (×4) [2] and RealSR (×4) [5]. The models are trained from scratch. δ denotes the performance gap between with and without augmentation.

Method	DIV2K (δ)	RealSR (δ)
EDSR	29.21 (+0.00)	28.89 (+0.00)
Cutout [8] (0.1%)	29.22 (+0.01)	28.95 (+0.06)
CutMix [36]	29.22 (+0.01)	28.89 (+0.00)
Mixup [37]	29.26 (+0.05)	28.98 (+0.09)
CutMixup	29.27 (+0.06)	29.03 (+0.14)
RGB perm.	29.30 (+0.09)	29.02 (+0.13)
Blend	29.23 (+0.02)	29.03 (+0.14)
CutBlur	**29.26 (+0.05)**	**29.12 (+0.23)**
All DA's (random)	**29.30 (+0.09)**	**29.16 (+0.27)**

(3) CutBlur介绍——一种为超分辨率任务设计的新的增强方法。

a. 算法:

首先我们给出两个图像补丁$LR: x_{LR} \varepsilon R^{W\times H\times C}, LR: x_{HR} \varepsilon R^{sW\times sH\times C}$，其中表示SR中的比例因子。如图1所示，因为CutBlur需要匹配x_{LR}和x_{HR}的分辨率，我们首先使用双三次内核来查询x_{LR} s次，即x^s_{LR}。CutBlur的目标是生成一对新的训练样本($\hat{x}_{HR} \rightarrow LR$，$\hat{x}_{LR} \rightarrow HR$)，通过剪切和粘贴$x_{HR}$的随机区域到相应的$x^s_{LR}$中，反之亦然:

$$\begin{aligned} \hat{x}_{HR\rightarrow LR} &= \mathbf{M} \odot x_{HR} + (\mathbf{1} - \mathbf{M}) \odot x^s_{LR} \\ \hat{x}_{LR\rightarrow HR} &= \mathbf{M} \odot x^s_{LR} + (\mathbf{1} - \mathbf{M}) \odot x_{HR} \end{aligned} \tag{1}$$

where $\mathbf{M} \in \{0,1\}^{sW\times sH}$ denotes a binary mask indicating where to replace, $\mathbf{1}$ is a binary mask filled with ones, and $\odot$ is element-wise multiplication. For sampling the mask and its coordinates, we follow the original CutMix [36].

b. 讨论（问题）：

① 为什么CutBlur能够用于SR?

CutBlur在具有相同内容的LR和HR图像补丁之间进行剪切和粘贴。通过将LR缝补到HR上，而由于图像内容不匹配，cutblur能够最小化边界效应。同时，cutblur利用的是整个图像的信息，同时随机的HR比和位置的样本不同，它具有正则化效果。

② 通过CutBlur模型学到了什么?

CutBlur防止SR模型过度锐化图像，并帮助它只超解析必要的区域。防止模型过分自信的做出决定。主要通过人工给出HR和经过CutBlur的LR。通过使用cutblur，我们可以从中发现其能够解决过度锐化的问题(由于本人水平有限，只能从原作代码中截取部分运行，结果如下图，与原作图进行对比) 由于正则化效应，它提高了其他LR区域的SR性能(图3)：

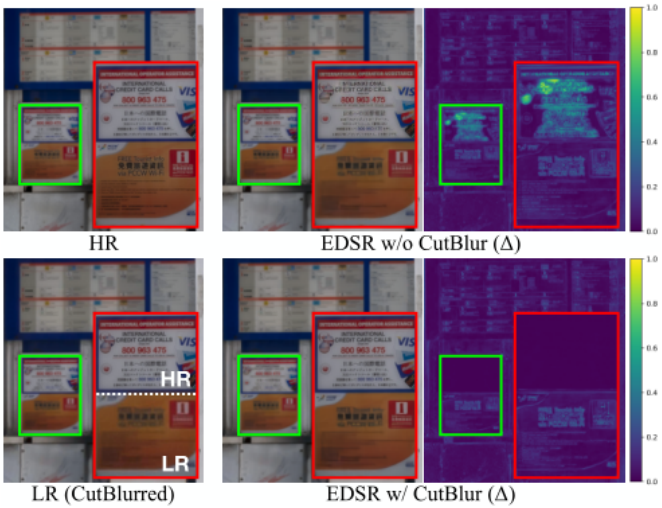
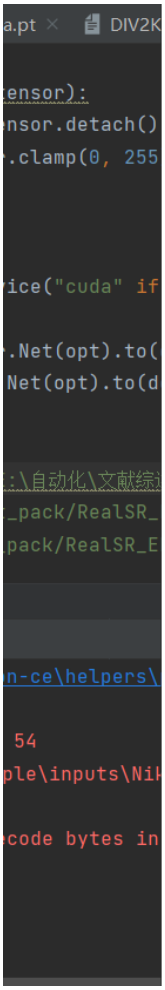


Figure 3. Qualitative comparison of the baseline and CutBlur model outputs when the input is augmented by CutBlur. Δ is the absolute residual intensity map between the network output and the ground-truth HR image. Unlike the baseline (**top right**), CutBlur model not only resolves the HR region but reduces Δ of the other LR input area as well (**bottom right**).



代码：

```
# options for EDSR
class Opt:
    scale = 4
    num_blocks = 32
    num_channels = 256
    res_scale = 0.1

def im2tensor(im):
    np_t = np.ascontiguousarray(im.transpose((2, 0, 1)))
    tensor = torch.from_numpy(np_t).float()
    return tensor

def tensor2im(tensor):
    tensor = tensor.detach().squeeze(0)
    im = tensor.clamp(0, 255).round().cpu().byte().permute(1, 2, 0).numpy()
    return im

opt = Opt()
dev = torch.device("cuda" if torch.cuda.is_available() else "cpu")

net_base = edsr.Net(opt).to(dev)
net_moa = edsr.Net(opt).to(dev)

#引入参数
path_image = "E:\自动化\文献综述\复现代码\cutblur-main\example\inputs\\0869.png"
path_base = "pt_pack/RealSR_EDSR_X4_base.pt"
path_moa = "pt_pack/RealSR_EDSR_X4_moa.pt"

state_base = torch.load(path_base, map_location=lambda storage, loc: storage)
state_moa = torch.load(path_moa, map_location=lambda storage, loc: storage)
net_base.load_state_dict(state_base)
net_moa.load_state_dict(state_moa)

LR = io.imread(path_image)
LR_tensor = im2tensor(LR).unsqueeze(0).to(dev)

with torch.no_grad():
    #引入参数比例以及pt源
    SR_base = tensor2im(net_base(LR_tensor))
    SR_moa = tensor2im(net_moa(LR_tensor))
    LR_plot = LR[100:400, 750:1100] / 255
    SR_base_plot = SR_base[100:400, 750:1100] / 255
    SR_moa_plot = SR_moa[100:400, 750:1100] / 255

    diff_SR_base = (LR_plot - SR_base_plot).mean(2) * 10
    diff_SR_moa = (LR_plot - SR_moa_plot).mean(2) * 10

    f, axarr = plt.subplots(3, 2, figsize=(10, 12))
    axarr[0, 0].imshow(LR_plot)
    axarr[0, 0].set_title("Input (HR)", fontsize=18)
    axarr[0, 0].axis("off")

    axarr[0, 1].axis("off")

    axarr[1, 0].imshow(SR_base_plot)
    axarr[1, 0].set_title("EDSR w/o MoA", fontsize=18)
    axarr[1, 0].axis("off")
```

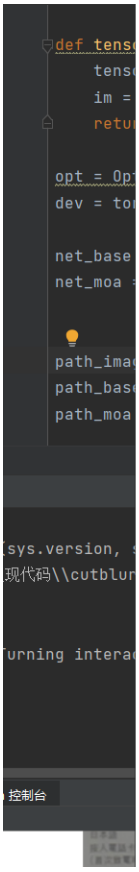
```
    axarr[1, 1].imshow(diff_SR_base, vmin=0, vmax=1, cmap="viridis")
    axarr[1, 1].set_title("EDSR w/o MoA (Δ)", fontsize=18)
    axarr[1, 1].axis("off")

    axarr[2, 0].imshow(SR_moa_plot)
    axarr[2, 0].set_title("EDSR w/ MoA", fontsize=18)
    axarr[2, 0].axis("off")

    axarr[2, 1].imshow(diff_SR_moa, vmin=0, vmax=1, cmap="viridis")
    axarr[2, 1].set_title("EDSR w/ MoA (Δ)", fontsize=18)
    axarr[2, 1].axis("off")

    plt.show()
```

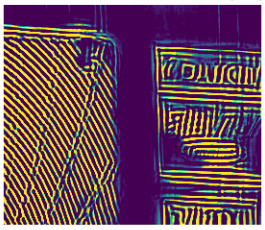
c. CutBlur vs 投入训练的人力资源:

d. 混合增强（MOA）：

用于提高性能增益的极限，将各种DA方法集成到单一框架里

e. 以下是其余图片的运行结果

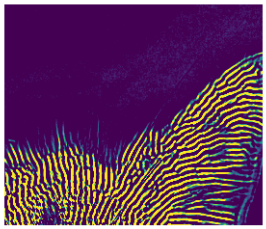




Input (HR)

EDSR w/o MoA

EDSR w/o MoA (Δ)



EDSR w/ MoA

EDSR w/ MoA (Δ)

(4) 实验分析

主要实现内容已包含在（2）（3）中，可参考。

a. 研究不同的模型和数据集

b. 对不同基准数据集进行比较

c. 公众视野里的CutBlur

d. 其他低层次视觉任务

(5) 总结

a. 实现细节：

使用作者的结论表：

Table 6. A description of data augmentations that are used in our final proposed method.

Name	Description	Default α
Cutout [8]	Erase (zero-out) randomly sampled pixels with probability α. Cutout-ed pixels are discarded when calculating loss by masking removed pixels.	0.001
CutMix [36]	Replace randomly selected square-shape region to sub-patch from other image. The coordinates are calculated as: $r_x = \text{Unif}(0, W)$, $r_w = \lambda W$, where $\lambda \sim \text{N}(\alpha, 0.01)$ (same for r_y and r_h).	0.7
Mixup [37]	Blend randomly selected two images. We use default setting of Feng *et al*. [10] which is: $I' = \lambda I_i + (1 - \lambda) I_j$, where $\lambda \sim \text{Beta}(\alpha, \alpha)$.	1.2
CutMixup	CutMix with the Mixup-ed image. CutMix and Mixup procedure use hyper-parameter α_1 and α_2 respectively.	0.7 / 1.2 (α_1 / α_2)
RGB permutation	Randomly permute RGB channels.	-
Blend	Blend image with vector $\mathbf{v} = (v_1, v_2, v_3)$, where $v_i \sim \text{Unif}(\alpha, 1)$.	0.6
CutBlur	Perform CutMix with same image but different resolution, producing $\hat{x}_{HR \to LR}$ and $\hat{x}_{LR \to HR}$. Randomly choose $\hat{x}$ from the [$\hat{x}_{HR \to LR}$, $\hat{x}_{LR \to HR}$], then provided selected one as input of the network.	0.7
MoA (Mixture of Augmentations)	Use all data augmentation method described above. Randomly select single augmentation from the augmentation pool then apply it.	-

表6描述了研究者使用的每个扩展的详细描述和设置。在这里，CutMixup,CutBlur和moaare是在论文中新提出的策略。超参数的描述遵循原始论文的符号。

b. 注意事项：

在运行代码的过程中，我尝试将blend与Cutup等合成使用，但是并不是很理想，仍需在思考如何正确实现。