Datawhale零基础入门CV赛事
街景字符编码识别
数据读取和数据扩增

分享人：小武

目录

contents

赛事资源

天池新人赛由天池与Datawhale联合发起，并提供学习内容和组织学习：

- Datawhale是一个专注于数据科学与AI领域的开源组织；
- CV直播PPT 可关注Datawhale公众号，回复关键词 **CV直播** 下载；
- 同时可以加入Datawhale数据竞赛交流群，一起组队参赛，交流学习；

个人介绍

小武

- ✓ Datawhale成员，开源贡献者
- ✓ 图像处理、数学建模爱好者
- ✓ 博客：https://blog.csdn.net/weixin_40647819

Datawhale CV小组开源项目：动手学CV-Pytorch版
https://github.com/datawhalechina/dive-into-cv-pytorch

Datawhale 组队学习：
图像处理（上）：https://github.com/datawhalechina/team-learning
图像处理（下）：敬请关注

Part 1 图像读取

图像读取

常用图像读取方法：

- OpenCV-python
  https://opencv.org/
- Pillow
  https://pillow.readthedocs.io/en/stable/
- matplotlib.image
  http://matplotlib.org/index.html
- scipy.misc
  https://www.scipy.org/
- skimage
  https://scikit-image.org/

图像读取-OpenCV

■ 直方图均衡操作

```python
import cv2
import numpy as np

#读入图片
img = cv2.imread('dog.jpg')
#通道分离
(B, G, R) = cv2.split(img)
#直方图均衡
BH = cv2.equalizeHist(B)
GH = cv2.equalizeHist(G)
RH = cv2.equalizeHist(R)
#合并每一个通道
result = cv2.merge((BH, GH, RH))
cv2.imshow('src ',img )
cv2.imshow('dst ',result )
print(img.shape) # (h,w,c)
print(img.size) # 像素总数目
print(img.dtype)
cv2.waitKey()
```

图像读取-Pillow

Datawhale

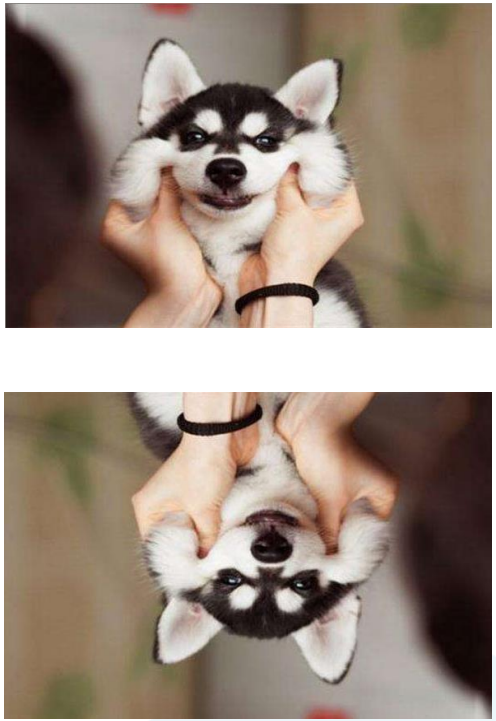
- 旋转操作

```
from PIL import Image
import numpy as np

img = Image.open('dog.jpg')
result=img.transpose(Image.ROTATE_180)

print(img.format)#格式
print(img.size) #省略了通道 ,为(w, h)
print(img.mode)
arr_img= np.array(img) #需要转换为数组
print (arr_img)

result.show() #显示图片
```



图像读取-matplotlib

- 和OpenCV结合（通道顺序转换）

```
import matplotlib.pyplot as plt
import numpy as np
import cv2

im2 = cv2.imread('dog.jpg')
plt.imshow(im2)
plt.axis('off')
plt.show()

#通道顺序转换
im2 = cv2.cvtColor(im2,cv2.COLOR_BGR2RGB)
plt.imshow(im2)
plt.axis('off')
plt.show()

print(im2.shape) # (h,w,c)
print(im2.size)
print(im2.dtype)
print(im2)
```

图像读取-scipy

Datawhale

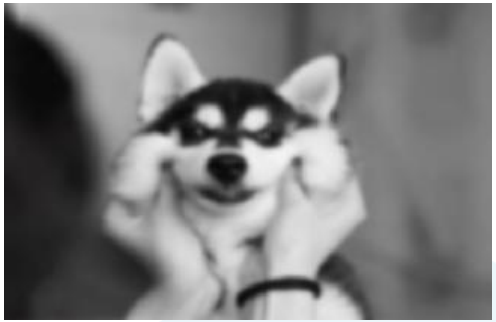
- 图像模糊

```
from scipy import misc
import scipy.ndimage as nd
import matplotlib.pyplot as plt

im = misc.imread('dog.jpg')

im = nd.gaussian_filter(im, 3)
#im=misc.imfilter(im, 'find_edges')
plt.axis('off')
plt.imshow(im)
plt.show()

print(sob.dtype)
print(im.size)
print(im.shape)
print(im)
```



图像读取-skimage

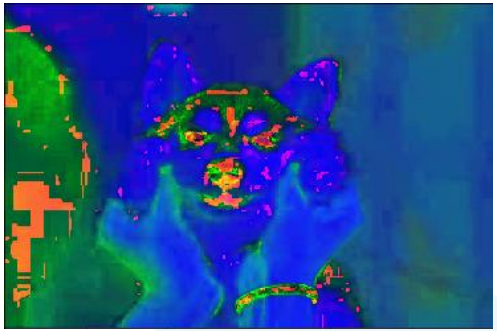
- 色彩转换

```
from skimage import color
from skimage import io

im = io.imread('dog.jpg')
io.imshow(im)
io.show()

#颜色转换
img_hsv = color.convert_colorspace(im, 'RGB',
                                   'HSV')
io.imshow(img_hsv)
io.show()

print(im.shape) # numpy矩阵, (h,w,c)
print(im.size)
print(im)
```



图像读取-总结

- **Pillow**只提供最基础的数字图像处理，功能有限，但很方便轻巧
- **Scikit-image**是基于**scipy**的一款图像处理包，功能也很强大
- **OpenCV**是一个非常全面的图像处理、计算机视觉库，功能全面强大
- **Pillow读入的图片是img类，其他库读进来的图片都是numpy 矩阵**
- **OpenCV读入的彩色图像通道顺序是BGR，其他图像库读入彩色图像都是RGB顺序存储**

Part 2 Pytorch数据读取

Pytorch数据读取

在模型训练之前，我们需要先读取和加载数据，Pytorch的torchvision中已经包含了很多常用数据集，如Imagenet，MNIST，CIFAR10、VOC、SVHN等，利用torchvision可以很方便地读取;另外，在实际应用中，我们可能还需要从各种不同的数据集或自己构建的数据集中读取图像。所以，我们从常见数据集读取方法和自定义读取数据方法两个方面介绍Pytorch数据读取方法。

- 常用数据集读取
- 自定义数据集读取

Pytorch常用数据集读取

Datawhale

对于常用的数据集，可以通过torchvision.datasets读取，所有datasets继承torch.utils.data.Dataset，也就是说，它们实现了 _getitem_ 和_len_ 方法。

那么，pytorch支持哪些常用数据加载呢，可以参见：
https://pytorch.org/docs/stable/torchvision/datasets.html

所有datasets读取方法的 API 均十分相似，以CIFAR10为例：

```
torchvision.datasets.CIFAR10(root, train=True, transform=None, target_transform=None, download=False)
```

参数：
root：存放数据集的路径。
train（bool，可选）–如果为True，则从训练集创建数据集，否则从测试集创建。
transform：数据预处理(数据增强)，如transforms.RandomRotation。
target_transform：标注的预处理。
download：是否下载，若为True则从互联网下载，如果在root已经存在，就不会再次下载。

Pytorch常用数据集读取

数据下载完成后，我们还需要做数据装载操作，加快我们准备数据集的速度。datasets继承至torch.utils.data.Dataset，而torch.utils.data.DataLoader对Dataset进行了封装，所以我们可以利用DataLoader进行多线程批量读取。

```
from PIL import Image
import torch
import torchvision
from torch.utils.data.dataset import Dataset
import torchvision.transforms as transforms

train_data=torchvision.datasets.CIFAR10('../../../dataset', train=True,
                                        transform=None,
                                        target_transform=None,
                                        download=True)

train_loader = torch.utils.data.DataLoader(train_data,
                                           batch_size=2,
                                           shuffle=True,
                                           num_workers=4)
```

Pytorch自定义数据集读取

pytorch自定义读取数据的方式 主要涉及到两个类：

- torch.utils.data.Dataset
- torch.utils.data.DataLoader

Dataset和DataLoder究竟有何区别？

Dataset：对数据集的封装，提供索引方式的对数据样本进行读取
DataLoder：对Dataset进行封装，提供批量读取的迭代读取

Pytorch自定义数据集读取

Dataset类的基本结构：需要实现 _getitem_ 和 _len_ 方法

```python
from torch.utils.data.dataset import Dataset

class MyDataset(Dataset):#继承Dataset

    def __init__(self):
        #  初始化文件路径或文件名列表
        pass

    def __getitem__(self, index):

        #1。从文件中读取一个数据（例如，使用PIL.Image.open，cv2.imread）
        #2。预处理数据（例如torchvision.Transform）。
        #3。返回数据对（例如图像和标签）。
        pass

    def __len__(self):
        return count
```

Part 3 数据扩增

常用数据扩增技术分类：

- 基于图像处理的数据扩增
- 基于深度学习的数据扩增

推荐论文阅读：

《A survey on Image Data Augmentation for Deep Learning》-2019
https://link.springer.com/article/10.1186/s40537-019-0197-0#Sec3

基于图像处理的数据扩增—几何变换

- 旋转
- 缩放
- 翻转
- 裁剪
- 平移
- 仿射变换

作用：几何变换可以有效地对抗数据中存在的位置偏差、视角偏差、尺寸偏差，而且易于实现，非常常用。

基于图像处理的数据扩增—灰度和彩色空间变换

- 亮度调整
- 对比度、饱和度调整
- 颜色空间转换
- 色彩调整

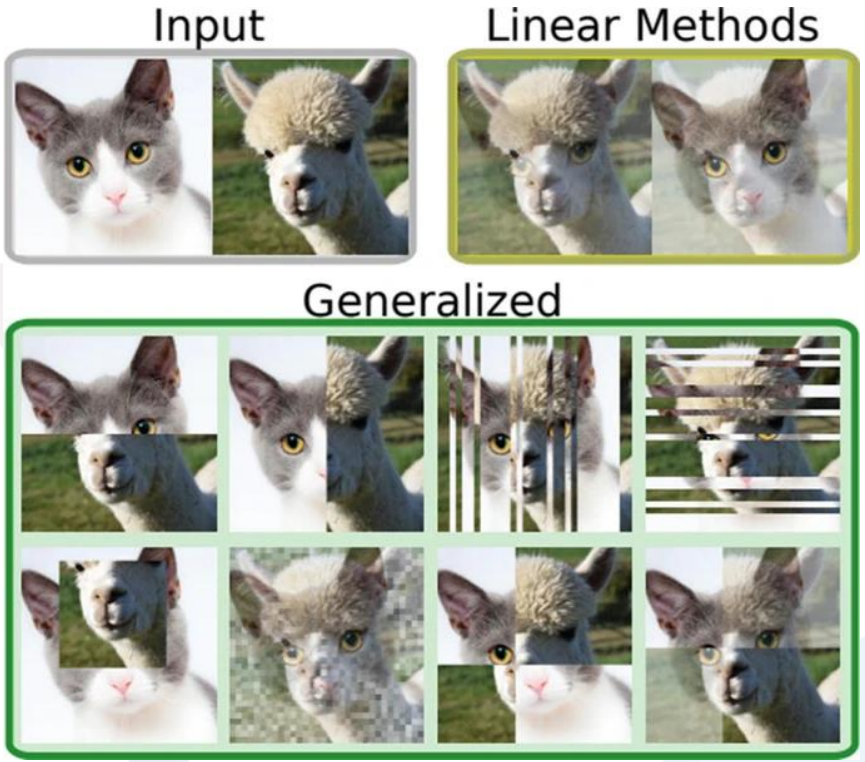
作用：对抗数据中存在的光照、色彩、亮度偏差。

基于图像处理的数据扩增—添加噪声和滤波

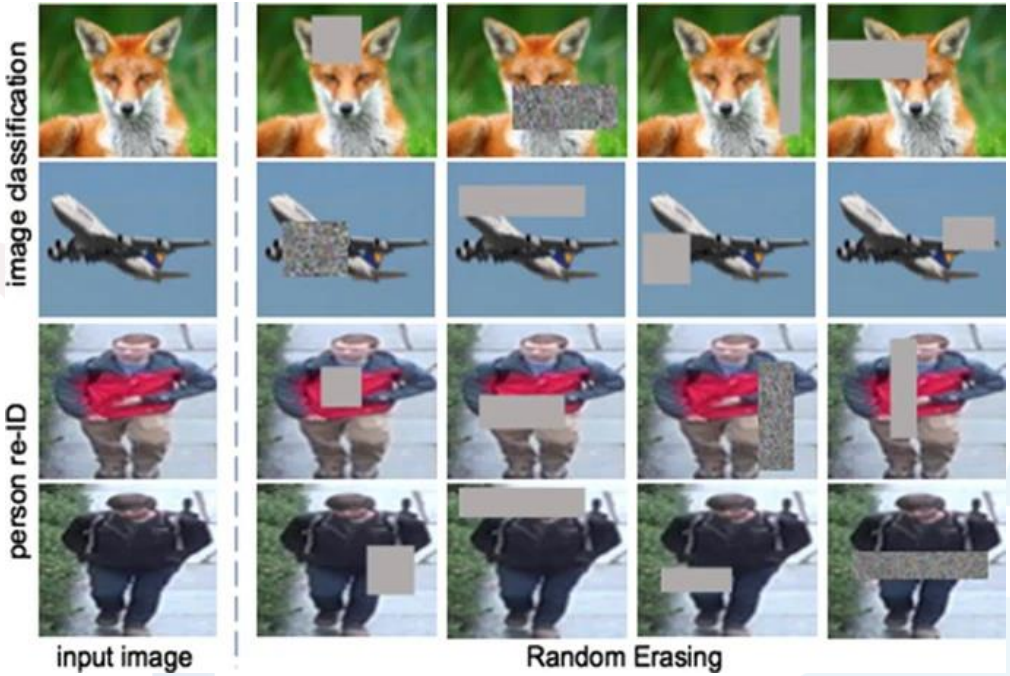
- 注入高斯噪声、椒盐噪声等
- 滤波：模糊、锐化

作用：应对噪声干扰、成像异常等特殊环境，帮助CNN学习更泛化的特征。

基于图像处理的数据扩增—Mixing images（图像混合）



基于图像处理的数据扩增—Random erasing（随机擦除）



基于深度学习的数据扩增

- 基于GAN的数据增强（GAN-based Data Augmentation）：使用 GAN 生成模型来生成更多的数据，可用作解决类别不平衡问题的过采样技术。
- 神经风格转换（Neural Style Transfer）：通过神经网络风格迁移来生成不同风格的数据，防止模型过拟合。
- AutoAugment

使用Pytorch进行数据增强——常用方法

Pytorch中，常用的数据增强的函数主要集成在了torchvision的transforms中,这里列出19种图像扩增强方法:

1.裁剪

- transforms.CenterCrop —对图片中心进行裁剪
- transforms.RandomCrop — 随机区域裁剪
- transforms.RandomResizeCrop — 随机长宽比裁剪
- transforms.FiveCrop —对图像四个角和中心进行裁剪得到五分图像
- transforms.TenCrop ——上下左右中心裁剪后翻转

2.翻转和旋转

- transforms.RandomHorizontalFlip — 依概率随机水平翻转
- transforms.RandomVerticalFlip —依概率随机垂直翻转
- transforms.RandomRotation — 随机旋转

3. 图像变换

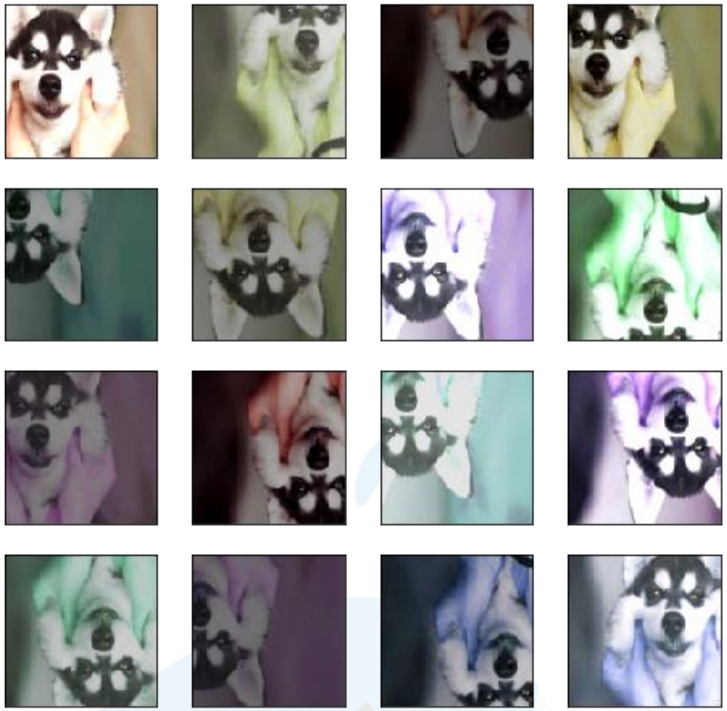
- transforms.Pad —使用固定值进行像素填充
- transforms.ColorJitter — 对图像颜色的对比度、饱和度、亮度、色相进行变换
- transforms.Grayscale — 对图像进行灰度变换
- transforms.RandomGrayscale — 依概率灰度化
- transforms.RandomAffine — 随机仿射变换
- transforms.LinearTransformation — 线性变换
- transforms.RandomErasing —随机选择图像中的矩形区域并擦除其像素
- transforms.Lambda — 用户自定义变换
- tuxiang transforms.Resize — 尺度缩放
- transforms.Totensor — 将 PIL Image 或者 numpy.ndarray 格式的数据转换成 tensor
- transforms.Normalize — 图像标准化

使用Pytorch进行数据增强——组合使用

```python
from PIL import Image
from torchvision import transforms as tfs
import matplotlib.pyplot as plt

im = Image.open('dog.jpg')
im_aug = tfs.Compose([
    tfs.Resize([200,200]),
    tfs.RandomVerticalFlip(),
    tfs.RandomCrop(110),
    tfs.ColorJitter(brightness=0.5, contrast=0.5, hue=0.5),
    #tfs.RandomRotation(5)
])

nrows = 4
ncols = 4
figsize = (8, 8)
_, figs = plt.subplots(nrows, ncols, figsize=figsize)
for i in range(nrows):
    for j in range(ncols):
        figs[i][j].imshow(im_aug(im))
        figs[i][j].axes.get_xaxis().set_visible(False)
        figs[i][j].axes.get_yaxis().set_visible(False)
plt.show()
```



Part 4 Q&A

如何提高得分?

- 改进baseline
- baseline过拟合问题
  尝试加入其他数据增强：噪声、图像模糊、图像锐化等等
- 用目标（字符）检测的思路

问答

- 你对比赛有什么问题?
- 你对学习有什么问题?
- 你对PPT内容有什么问题?

Datawhale

一个专注于AI领域的开源组织