实践教程｜十个PyTorch最常用的转换函数

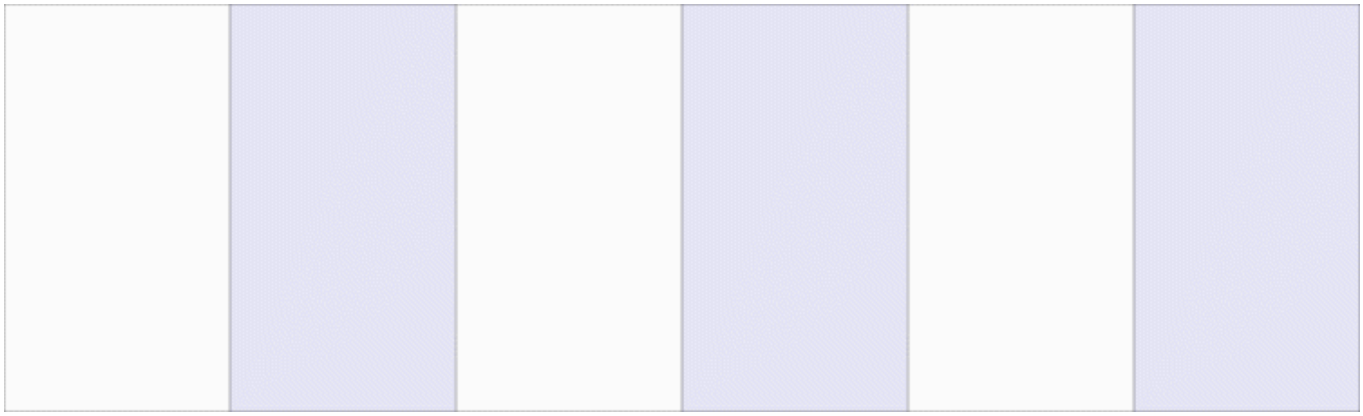
极市平台 2023-03-10 22:00:10 发表于广东 手机阅读

以下文章来源于深度学习与计算机视觉 ，作者磐怼怼

深度学习与计算机视觉

深度学习与计算机视觉碰撞出了新的火花，本公众号将坚持分享原创计算机视觉技术相...

↑ 点击**蓝字** 关注极市平台



作者｜磐怼怼

来源｜深度学习与计算机视觉

编辑｜极市平台

极市导读

常用的转换函数以及代码介绍。 >>加入极市CV技术交流群，走在计算机视觉的最前沿

介绍

Pytorch是一个深度学习框架，广泛用于图像分类、分割、目标识别等各种任务。在这种情况下，我们必须处理各种类型的数据。很可能在大多数情况下，数据可能不是我们所需要的格式。PyTorch转换就是救星。

torchvision.transforms模块提供了可以使用的各种图像转换。我们使用变换对数据进行一些操作，使其适合于训练torchvision模块，PyTorch为常见的图像变换提供变换有关的函数。这些变换可以使用Compose链接在一起。

1. ToTensor

这是一个非常常用的转换。在PyTorch中，我们主要处理张量形式的数据。如果输入数据是NumPy数组或PIL图像的形式，我们可以使用ToTensor将其转换为张量格式。

最后一个张量的形式是（C * H * W）。同时，还执行从0–255到0–1的范围内的缩放操作。

让我们用一个例子来更好地理解它。在这个博客中，我将使用Ragnar（我最喜欢的虚构角色）的图像来执行转换。

```
image
```

```
import numpy as np
from torchvision import transforms
transform = transforms.Compose([transforms.ToTensor()])
tensor_img = transform(image)
tensor_img.shape
```

```
torch.Size([3, 800, 1200])
```

2. Normalize

此操作将获取张量图像，并使用平均值和标准差对其进行归一化。它有3个参数：mean, std, inplace。我们需要为3个通道提供一系列平均值，作为参数“mean”，“std”类似。如果将“inplace”设为True，则将计算得到的值覆盖之前的值。

```
torchvision.transforms.Normalize\(\[meanOfChannel1, meanOfChannel2, meanOfChannel3\], \[st
#Example:
transforms.Normalize\(\(0.5, 0.5, 0.5\), \(0.5, 0.5, 0.5\)\)
```

3. CenterCrop

这将在中心裁剪给定的张量图像。你可以以（高度、宽度）的形式向transforms.CenterCrop（）提供要裁剪的大小作为输入。让我们在图像上实现这个并进行检查。

```
transform = transforms.Compose\(\[transforms.ToTensor\(\),transforms.CenterCrop\(\(200,100
tensor\_img = transform\(image\)
tensor\_img.shape
Output: torch.Size\(\[3, 200, 100\]\)
```

如果只提供一个尺寸标注而不是两个尺寸标注，会发生什么情况?

它将假设它是一个正方形，并且将生成一个（size, size)）的裁剪。

如果给定的尺寸比原来的尺寸大呢?

沿着这些边，图像将填充0!

4. RandomHorizontalFlip

此变换将以给定的概率水平（随机）翻转图像。你可以通过参数“p”来设置这个概率。p的默认值为0.5。

检查我下面的例子来理解。

```
transform = transforms.Compose\(\[transforms.RandomHorizontalFlip\(p=0.9\)\]\)
tensor\_img = transform\(image\)
tensor\_img
```

查看原始图像和翻转的图像!

5. RandomRotation

此变换将图像随机旋转一个角度。以度为单位的角度可以作为参数“degrees”的输入。

```
transform = transforms.Compose\(\[transforms.RandomRotation\(degrees=180\)\]\)
tensor\_img = transform\(image\)
tensor\_img
```

查看上述代码的转换！

旋转图像

6. Grayscale

此转换将把原始RGB图像更改为灰度（即黑白）。你可以提供你想要多少个通道作为参数“num_output_channels”的输入。

```
transform = transforms.Compose\(\[transforms.Grayscale\(num\_output\_channels=1\)\]\)
tensor\_img = transform\(image\)
tensor\_img
```

输出如下所示。

7. GaussianBlur

在这里，图像将被随机选择的高斯模糊所模糊。必须提供参数kernel_size。

```
transform = transforms.Compose\(\[transforms.GaussianBlur\(kernel\_size=501\)\]\)
tensor\_img = transform\(image\)
tensor\_img
```

8. RandomApply

此转换将随机应用给定的转换列表。

```
transform = transforms.RandomApply\(\[transforms.RandomSizedCrop\(200\),transforms.RandomH
tensor\_img = transform\(image\)
```

9. Compose

在本文中，我们一直在使用Compose() 。为了清楚地定义它，它将几个变换组合在一起。

```
transforms.Compose\(\[transforms.Grayscale\(1\),transforms.CenterCrop\(10\), transforms.To
```

一些转换将以所需格式处理数据。然而，对于图像数据增强，则需要灰度、随机水平翻转和随机旋转等变换。

10. 函数变换

在我们学习到的所有变换中，你可以注意到参数是随机生成的。这通常足以进行数据扩充。但是，有时你可能需要对转换管道进行更细粒度的控制。在这种情况下，可以使用函数变换。在这里，你可以指定或生成所有参数。一个附加的优点是，一个特定定义的函数变换可以应用于多个图像。

可以从torchvision.transforms.functional访问所有函数转换。

现在让我们深入了解PyTorch提供的不同功能转换。

A） 调整亮度：adjust_brightness

这主要是调整图像的亮度。它以PyTorch张量的形式将图像作为输入。它还有一个重要参数“亮度系数”。这将表示如何实际更改亮度。

例如，如果值为1，则会得到与输入相同的图像。如果该值大于1，将获得更亮的图像。如果它小于1，你会得到一个更暗的图像。可以相应地传递浮点值。返回的图像将是张量或PIL图像。

```
new\_img = transforms.functional.adjust\_brightness\(image,brightness\_factor=2\)
new\_img
```

B） 调整对比度：adjust_contrast

上面我们看到了如何调整亮度，这里我们有另一个用于调整图像对比度的变换。它需要两个输入参数：张量形式的图像和“对比度因子”。第二个参数将输入一个浮点值，它将告诉你如何调整对比度。但不能是负的。

```
new\_img = transforms.functional.adjust\_contrast\(image,contrast\_factor=3.8\)
new\_img
```

C） 调整色调：adjust_hue

色调是图像的一个重要属性。Pytorch允许你通过transforms.functional.adjust_hue进行调整。

想知道它是怎么工作的吗?

首先，图像将被转换成HSV（色调，饱和度，值）形式。将根据我们的参数在H通道中进行更改。更改后，图像将转换为其原始形式。重要的参数是“色调因子”。它可以是[-0.5.0.5]范围内的浮点值。尝试实现下面的示例。

```
new\_img = transforms.functional.adjust\_hue\(image,hue\_factor=0.3\)
new\_img
```

D） 调整饱和度：adjust_saturation

这是为了调整输入图像的颜色饱和度。与上述情况类似，我们有一个“饱和度系数”参数，它决定了饱和度的变化方式。这将输入一个浮点值。如果将其设置为0，则会得到黑白图像。

```
new\_img = transforms.functional.adjust\_saturation\(image,saturation\_factor=6\)
```

输出如下图所示！

E） 调整锐度：adjust_sharpness

你可以通过此变换调整图像的清晰度。它采用浮点值作为“锐度系数”参数的输入。此值可以是除负值以外的任何值。

在下面的代码中，我使用了10的锐度因子，这意味着变换后的图像将是原始图像的10倍锐度。

```
new\_img = transforms.functional.adjust\_sharpness\(image,sharpness\_factor=10\)
```

检查输出！

F） 均衡：equalize

这种变换将均衡图像的直方图。

怎么会这样？

它将对输入应用非线性映射，从而在输出中创建灰度值的均匀分布。

```
new\_img = transforms.functional.equalize\(image\)
```

这些是一些重要的函数转换，将有助于在图像预处理阶段。它们也可以组合使用。

点击阅读原文进入CV社区

收获更多技术干货

阅读原文

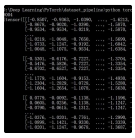
喜欢此内容的人还喜欢

YOLOv5帮助母猪产仔？南京农业大学研发母猪产仔检测模型并部署到Jetson Nano开发板

极市平台

实践教程｜PyTorch数据导入机制与标准化代码模板

极市平台



实践教程｜使用 OpenCV 进行特征提取（颜色、形状和纹理）

极市平台