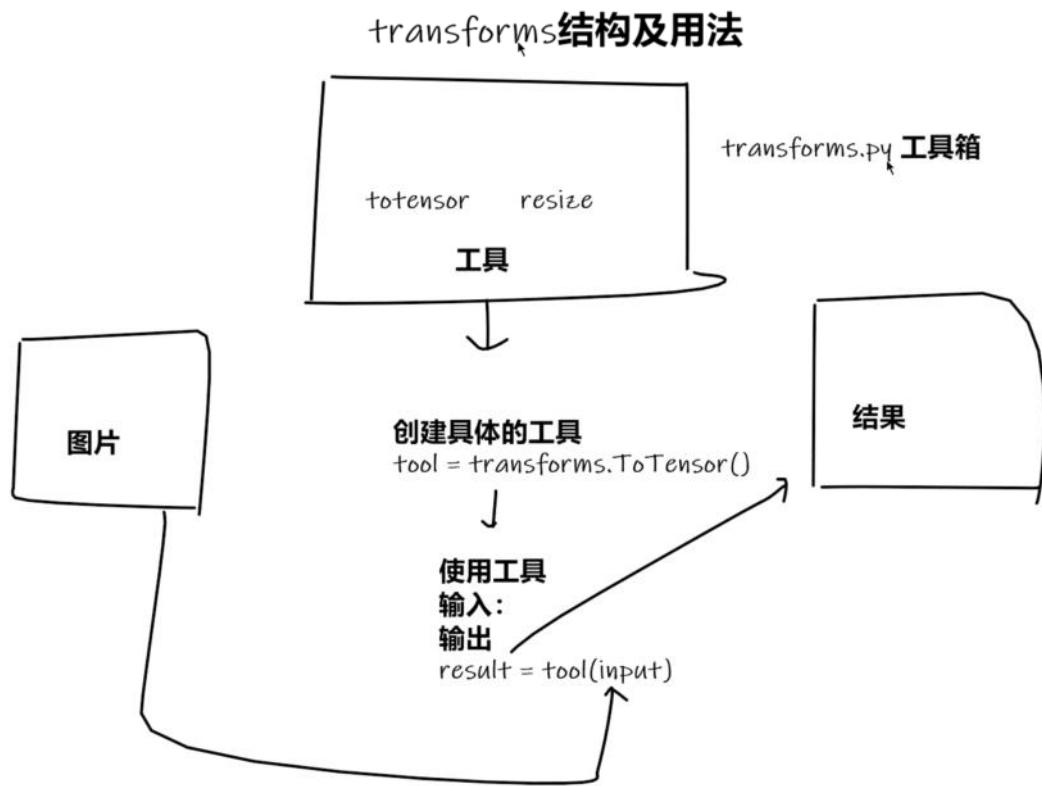


transform

2025年11月15日 19:26



为什么需要Tensor数据类型

包含了深度学习所要用到的一些数据，方便后面进行一些训练

学会查看官方文件

The screenshot shows a PyCharm interface with the following details:

- Project Structure:** The left sidebar displays the project structure under "Project 1". A red circle labeled "2" highlights the "transform" folder, which contains files like "transform.py", "test.py", and "webp transform.py".
- Code Editor:** The main window shows the content of "transforms.py". A red circle labeled "1" is positioned at the top right of the editor tab. The code imports various modules and defines a class with a list of transforms.
- Structure View:** On the right, a "Structure" view is open, showing a tree of available transforms. A red circle labeled "3" is on the "Structure" tab. The tree includes nodes for "Compose", "ToTensor", "PILToTensor", "ConvertImageDtype", "ToPILImage", "Normalize", "Resize", "CenterCrop", "Pad", "Lambda", "RandomTransforms", "RandomApply", "RandomOrder", "RandomChoice", "RandomCrop", and "RandomHorizontalFlip".

```
1 import math
2 import numbers
3 import random
4 import warnings
5 from collections.abc import Sequence
6 from typing import List, Optional, Tuple, Union
7
8 import torch
9 from torch import Tensor
10
11 try:
12     import accimage
13 except ImportError:
14     accimage = None
15
16 from ..utils import _log_api_usage_once
17 from . import functional as F
18 from .functional import _interpolation_modes_from_int, I
19
20 __all__ = [
21     "Compose",
22     "ToTensor",
23     "PILToTensor",
24     "ConvertImageDtype",
25     "ToPILImage",
26     "Normalize",
27     "Resize",
28     "CenterCrop",
29     "Pad",
30     "Lambda",
31     "RandomApply",
32     "RandomChoice",
33     "RandomOrder",
34     "RandomCrop",
35     "RandomHorizontalFlip",
36     "RandomVerticalFlip",
37     "RandomResizedCrop",
38     "FiveCrop",
```

忽视的小细节，导致运行错误

虽然是在transform.py的编辑界面下点的运行，但是上一个运行的控制台什么的都没有关闭，导致现在运行的还是上一个程序（如绿色部分）

路径的错误

D:\anaconda3\envs\pytorch\python.exe D:\pycharm_space\PycharmProjects\PythonProject1\tensorboard\t_b.py
<class 'numpy.ndarray'>
(334, 500, 3)
进程已结束，退出代码为 0

D:\anaconda3\envs\pytorch\python.exe D:\pycharm_space\PycharmProjects\PythonProject1\transform\transform.py
from PIL import Image
from torchvision import transforms

img_path = "transform/djsy.png"
img = Image.open(img_path) #把图片重新存起来 Image.open读取之后存的是PIL格式的
print(type(img))
tensor_trans = transforms.ToTensor()
tensor_img = tensor_trans(img)
print(tensor_img)

报错中显示程序在找 'transform/djsy.png'，这说明当前工作目录是 PythonProject1（项目根目录），而脚本误以为图片在 PythonProject1/transform/djsy.png。但实际情况是：
如果图片和脚本在同一个 transform 文件夹下，正确的相对路径应该是 djsy.png [脚本所在目录为基准]，而非 transform/djsy.png。

代码红了

按 alt + enter

```
# print(tensor_img)
# 使用tensorboard查看图片
SummaryWriter
    未解析的引用 'SummaryWriter'
    导入 'torch.utils.tensorboard.SummaryWriter' Alt+Shift+Enter 更多操作... Alt+Enter
```

就可直接导入需要的库

print(type(img))
使用tensorboard查看图片
SummaryWriter

导入 'torch.utils.tensorboard.SummaryWriter'
在模块 transform.py 中创建类 'SummaryWriter'
重命名引用
忽略 'transform.transform' 的所有未解析特性
忽略未解析的引用 'transform.transform.SummaryWriter'

重新设置文件格式
忽略这样的错误
编辑检查配置文件设置
按 Ctrl+Q 切换预览

查看需要传入的数据

直接按ctrl + P

print(type(img))
tensor_trans = transforms.ToTensor()
tensor_img = tensor_trans(img)
print(tensor_img) self, log_dir: str = None, comment, purge_step, max_queue, flush_secs, filename_suffix
使用tensorboard查看图片
SummaryWriter("logs")

使用tensorboard时要注意图片的格式



但是

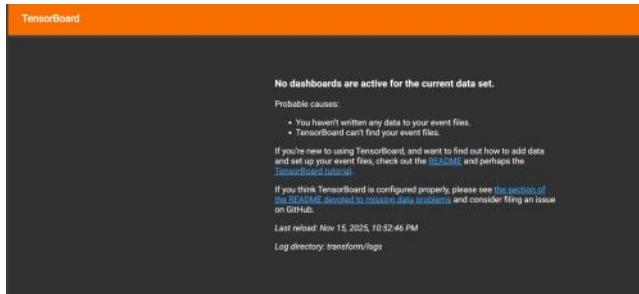
def add_images(
 self, tag, img_tensor, global_step=None, walltime=None, dataformats="NCHW"
):
 """Add batched image data to summary.

官方描述文件里是这个

所以

sor_img, dataformats = "CHW")

别忘记加writer.close()



我没有加writer.close()这个是干嘛的

writer.close() 是 SummaryWriter (TensorBoard 写入工具) 的一个重要方法，作用是**确保所有缓存的数据被强制写入到日志文件中，并释放相关资源**。具体来说：

1. 为什么需要 writer.close()？

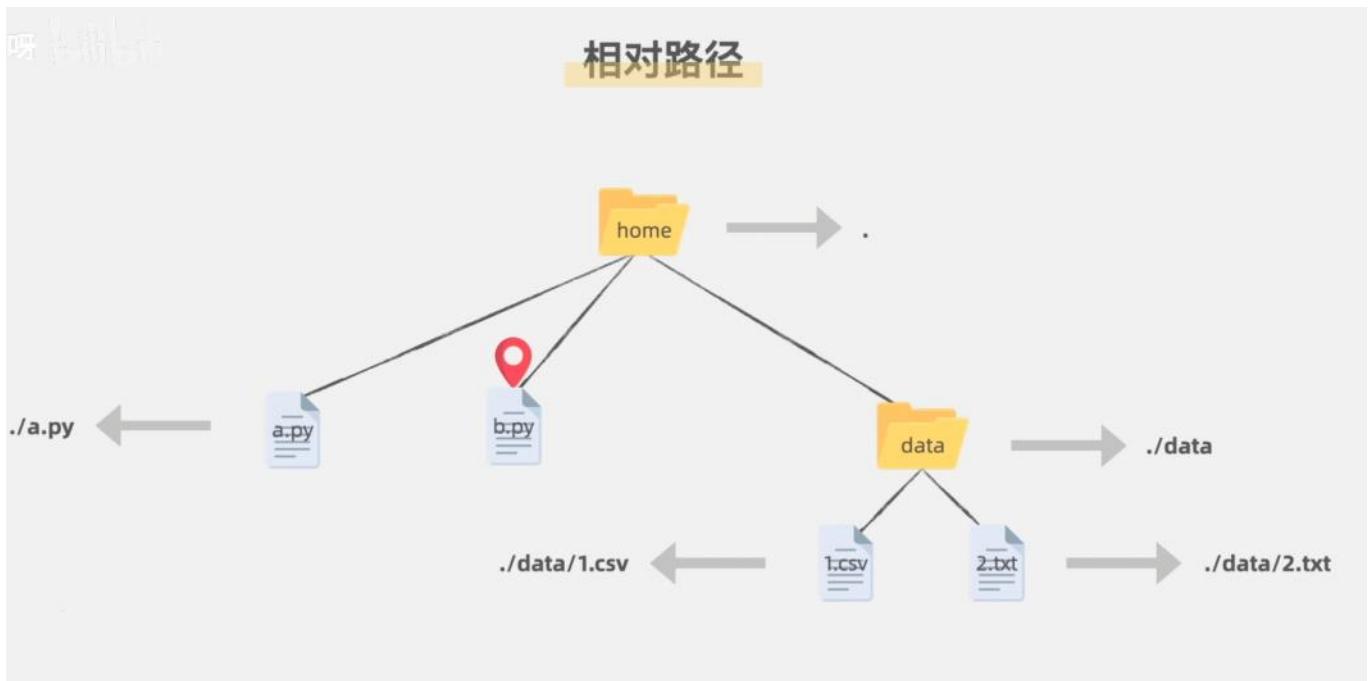
SummaryWriter 在写入数据 (如 add_scalar、add_image 等) 时，并不会立即把所有数据直接写入磁盘，而是先缓存在内存中，积累到一定量后再批量写入 (类似“缓冲区”机制，提升效率)。

如果代码中没有调用 writer.close()：

迅雷还能这样用



相对路径

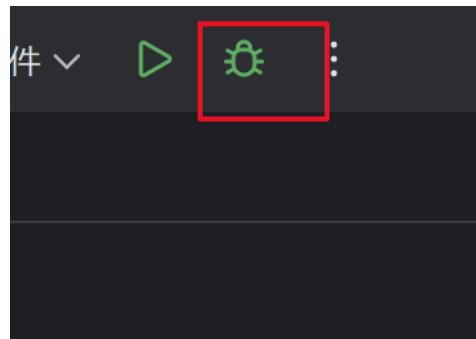


debug (调试)

```

5     img_path = r"D:\pycharm_sp
6     img = Image.open(img_path)
7     print(type(img))
8     tensor_trans = transforms.
9     tensor_img = tensor_trans(
10    print(tensor_img)
11    #使用tensorboard查看图片
12    writer = SummaryWriter("lo
13    writer.add_images( tag: "te
14    writer.cl
正在提取文档

```



打上断点

再按这个小虫

- 步入 (Step Into) :

当程序执行到某一行代码时，如果这行代码包含函数调用，执行“步入”操作会进入该函数内部，并停在函数的第一行代码处，方便调试函数内部的逻辑。

例如：若当前行是 `result = add(1, 2)`，步入后会进入 `add` 函数的定义体（如 `def add(a, b):` 后的第一行）。

- 步出 (Step Out) :

当程序正在函数内部执行时，执行“步出”操作会继续执行当前函数的剩余代码，直到函数执行完毕并返回调用处，然后停在函数调用语句的下一行。

例如：在 `add` 函数内部调试时，步出后会执行完 `add` 函数，回到 `result = add(1, 2)` 的下一行代码。

1. 步过 (Step Over)

当执行到包含函数调用的代码行时，“步过”会**执行整个函数但不进入函数内部**，直接停在该函数调用语句的下一行。

例如：当前行是 `result = add(1, 2)`，步过后会计算完 `add` 函数的结果，直接停在 `result = ...` 的下一行，适合确认函数整体返回结果是否正确，无需关注函数内部细节。