今天声学了吗

音频信号处理
及基于音频的深度学习教程

Audio Signal Processing
Audio-based Deep Learning Tutorials

b站：今天声学了吗

公众号：今天声学了吗

邮箱：1319560779@qq.com

编辑器的介绍

- **环境安装**

1. 在系统上安装Anaconda/Conda/CUDA/CUdnn/Pytorch
2. 参考视频 https://www.bilibili.com/video/BV1ov41137Z8/?share_source=copy_web&vd_source=bc17cd59a8134948a793a47630875aa1
3. 电脑系统： 独立显卡CPU 处理器：Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz 2.21 GHz，系统类型：win10 x64 的处理器
4. 环境版本：Anaconda2020.02/Conda22.9.0/CUDA11.3/CUdnn11.3/Python3.7.6/Tensorflow2.6.0 /pytorch 1.12.1
5. 测试：a = torch.cuda.is_available() a输出False则安装失败

- **注意事项：**

1. 一定不能忽略每一个步骤，要保证每次测试运行都是正确的，否则在后续写代码的时候会产生比较大的麻烦
2. 版本匹配问题，建议不要下载最新的版本，并且每个软件的版本都有匹配

编辑器的介绍

- **环境介绍**

1. 什么是python？python是一种面向对象的解释型计算机程序设计语言。（面向对象的语言）
2. 什么是Anaconda？Anaconda 可以完成同时对不同环境进行统一管理。Anaconda包含了Conda、python在内的超过180个科学包及其依赖项。（各种工具包的管理平台）
3. 什么是Conda？Conda是环境的管理工具。用于快速安装、运行和升级包及其依赖项。在计算机中便捷地创建、保存、加载和切换环境
4. 什么是pip？pip是用于安装和管理软件包的包管理器。（仅适用于pycharm。）`“pip Installs Packages”`（“pip安装包”）`“pip Installs Python”`（“pip安装python”）
5. Pip 与 Conda 比较：1） 依赖项检查。pip：不一定会展示所需的其他依赖包。Conda：列出所需其他依赖包。安装包时自动安装其依赖项。可以便捷地在包的不同版本中自由切换。 2）环境管理。Conda：在不同环境之间进行切换，环境管理较为简单。 3） 对系统自带python的影响。pip：在系统自带python中包的更新/回退版本/卸载将影响其他程序。Conda：不会影响系统自带python。

   4） 适用语言。pip：仅适用于python。

Conda：适用于python, R, Ruby, Lua, Scala, Java, JavaScript, C/C++, FORTRAN

编辑器的介绍

- **环境介绍**

1. 什么是CUDA？CUDA（Compute Unified Device Architecture），是一种由NVIDIA推出的**通用并行计算架构**，该架构使GPU能够解决复杂的计算问题。因为深度学习通常是利用tensor类型，需要大量计算，GPU的并行计算能力，因此一般在CUDA这个平台处理数据。主流的深度学习框架也都是基于CUDA进行GPU并行加速的，几乎无一例外。（CPU也是可以计算的，但对于大数据集，处理很慢）
2. 什么是CUDNN？ CUDNN： 是针对深度卷积神经网络的加速库。NVIDIA CUDNN可以在GPU上实现高性能现代并行计算。
3. CUDA与CUDNN的关系

CUDNN是基于CUDA的深度学习GPU加速库，有了它才能在GPU上完成深度学习的计算。它就相当于工作的工具，比如它就是个扳手。想要在CUDA上运行深度神经网络，就要安装CUDNN ， 就像你想要拧个螺帽就要把扳手买回来。这样才能使GPU进行深度神经网络的工作，工作速度相较CPU快很多。

编辑器的介绍

- **编译方式**

编译方式有三种，这三者的区别如下

1. PyCharm主工作台 - 每次运行都是从头开始，适用于大型项目。

- 优点：可阅读性高

1. Python console - 一行/一块运行，若有错误直接修改错误一行，并从错误的哪一行开始运行。

- 操作：enter是运行，shift enter是换行。

- 优点：可以显示每个变量的属性，便于修改

2. Jupyter notebook - 一块块运行，enter是换行，shift enter是运行，可以从一个**错误的块运行**

- 启动：打开Anaconda->输入conda activate pytorch -> jupyter notebook

- 操作：shift enter 换行，enter运行

我比较喜欢用第一种（运行整个文件的代码，阅读性比较高，用于存储一整个程序的代码）

和第二种（一块/一行运行，方便看各个变量的变化，用于调试），

所以我在以后的教程中会用这两个编译方式。

谢谢大家

Audio Signal Processing
Audio-based Deep Learning Tutorials

b站：今天声学了吗

公众号：今天声学了吗

邮箱：1319560779@qq.com

今天声学了吗

音频信号处理
及基于音频的深度学习教程

Audio Signal Processing
Audio-based Deep Learning Tutorials

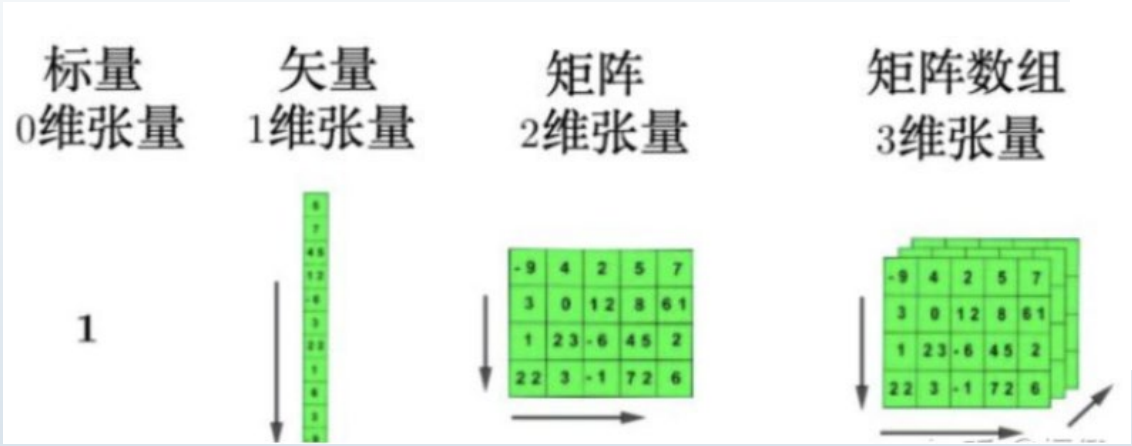
b站：今天声学了吗

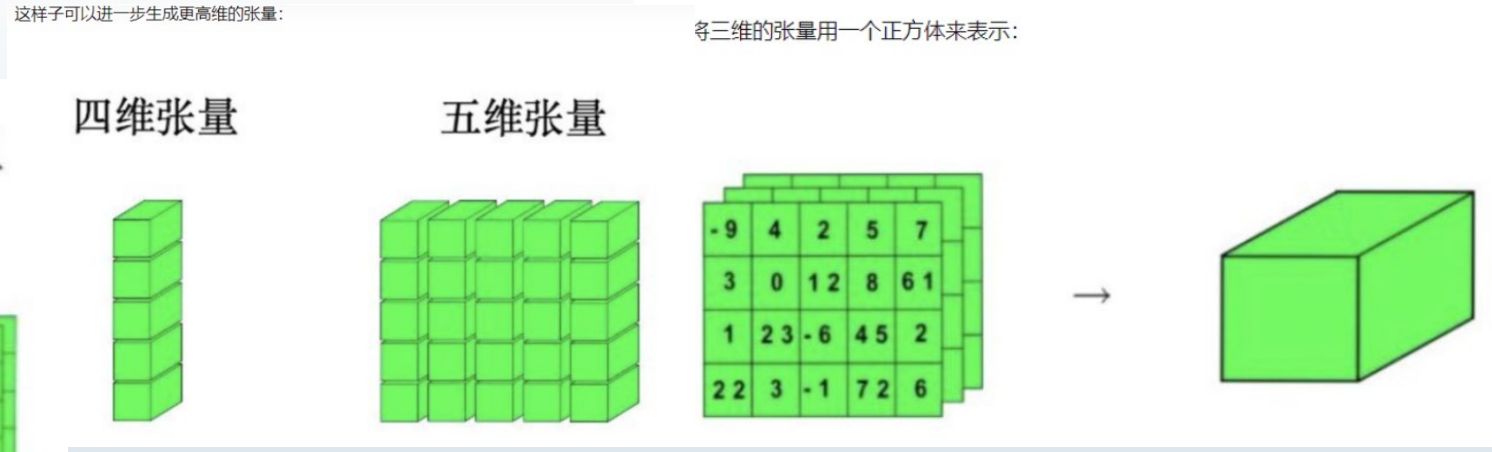
公众号：今天声学了吗

邮箱：1319560779@qq.com

张量Tensor

1. 什么是张量Tensor？张量是用张量积组合在一起的向量和共向量的集合，通俗理解是一个多维数组（multidimensional array）张量如同数组和矩阵一样，是一种特殊的数据结构。在PyTorch中，神经网络的输入、输出以及网络的参数等数据，都是使用张量来进行描述。
2. 操作：张量的使用和`Numpy`中的`ndarrays`很类似，区别在于张量可以在`GPU`或其它专用硬件上运行，这样可以得到更快的加速效果。





张量Tensor

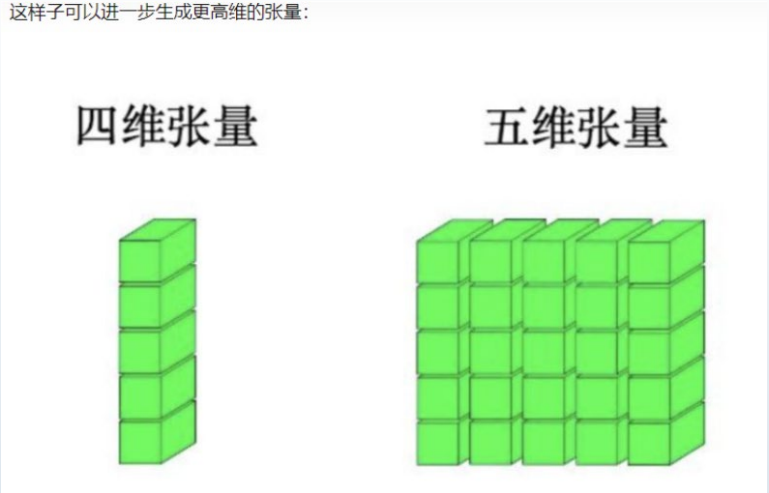
1. 什么是TensorFlow： 图的每个节点的输入输出都是 Tensor数据类型，而连接节点的有向线段表示从一个 Tensor 状态到另一个 Tensor 状态，是Flow的状态。在计算图网络中，连接每个节点(操作)的边是TensorFlow。
2. 代码实现：

```
# 0.定义tensor的数据类型
# 使用不同类型 torch.IntTensor()/ torch.DoubleTensor()
# 只定义形状未赋值 torch.Tensor(1,3)
# 定义全为0的tensor，形状与input相似
# 定义元素为一组在[0,1)内分布的随机数
# 定义元素为一组满足标准正态分布(均值为0,方差为1)的随机数
# 定义返回形状为1*n，元素为0~n内不重复的整型Tensor

# 1.Tensor的形状调整
# torch.view(3,1)
# torch.resize_(3,1)

# 2.Tensor的数据类型，任意位置赋值
# torch.int()
# torch[index] = 1
```

```
# 3.tensor的运算
# torch.add(),torch.sub(),torch.mul(),torch.div()
# 幂次 torch.pow
# 各元素单独相乘*与矩阵的乘法 matmul

# 4.Tensor的拼接，横向、纵向
torch.cat([,],dim)

# 5.将列表，数组类型转换为tensor
# np->tensor torch.tensor(np_data) 与 torch.from_numpy(np_data)
# tensor->np tensor.numpy()
```

```
"""
1.使用函数注意大小写，torch.Tensor()
2.直接使用该tensor操作，操作符后加_ a.add_(b)
1.张量维度的判断——最中间的括号是一行的个数
2.矩阵的乘法 对应维度要相同
"""
```

这样子可以进一步生成更高维的张量：



类Class

什么是类Class？存储对象和函数的集合，调用Class中的函数，完成我们需要实现的功能。

类的定义

- 定义自己的类（无继承）——定义可调用的类
- 利用已存在的类，定义自己的类（模板）——定义数据集类
- 利用已存在的类，定义自己的类（模板）——定义网络结构类

注意：

- 一般情况在python中函数，函数类都是可调用的。至于由类生成的对象，是否是可调用的，主要是看类中是否实现魔术方法`__call__`。
- 调用时注意输入的参数类型
- 多看文档，多调试

CHAPTER 2--PyCharm入门

元素处理中常用的类Class

- Dataset
- 来源：`torch.utils.data`
- 作用：数据集的定义与提取等初始化操作
- Dataset使用方法：继承Dataset定义数据集的类Class，注意：区分类Dataset与下载数据集的函数datasets
- 代码：

```
# 0.环境安装：下载 torchaudio -> pip install torchaudio
# import torchaudio
# from torch.utils.data import Dataset
# 1.下载数据集dataset_yesno
torchaudio.datasets.YESNO()
# 2.定义自己的数据类
# __init__(self,dir,labe) -> None:
# __getitem__(self, index)
# __len__(self)
# 3.调用数据类
# mydata(dir,label)
```

```
# __init__(self,dir,labe) -> None:
# __getitem__(self, index)
# __len__(self)
```

```
"""
1.数据集dataset 中的特殊函数__getitem__提取数据集中的项目
当传参使用mydata[index]->mydata.__getitem__(index)
2.YESNO一个人在希伯来语中说是或否的 60次； 每个记录长 8 个字
"""
```

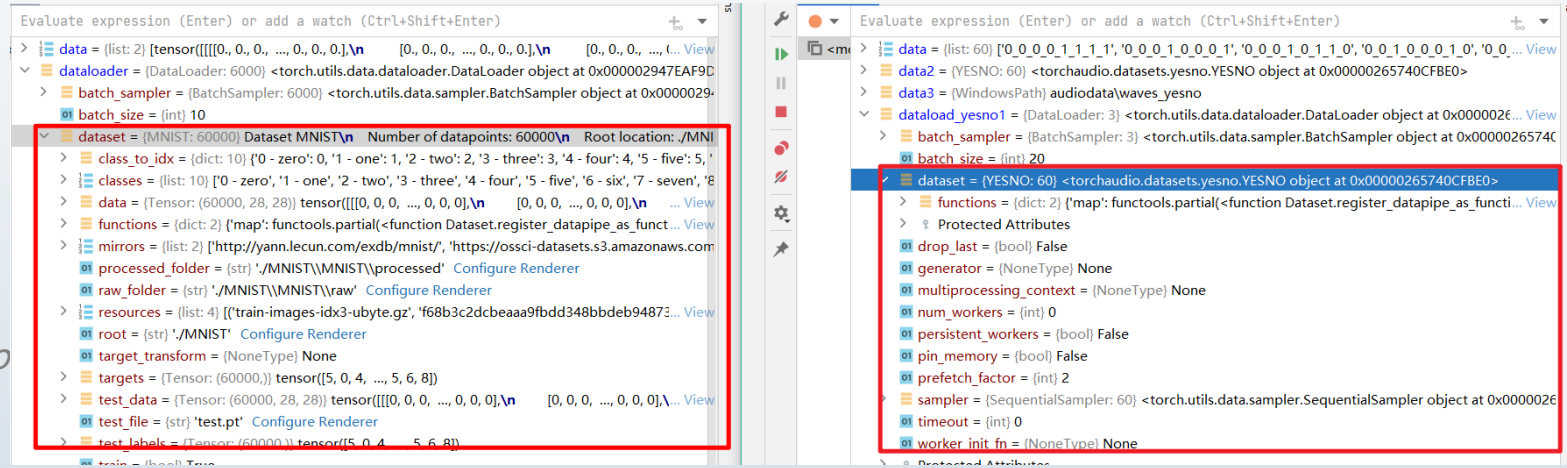
CHAPTER 2--PyCharm入门

元素处理中常用的类Class

- Dataloader
- 来源：`torch.utils.data`
- 作用：对数据集打包处理，因为GPU处理一批数据的速度=处理一个数据的速度，使用GPU处理会很快。
- Dataloader使用方法：定义*batch_size*，表示一次性读取数据集中元素的个数，并且集合在一起进行返回；shuffle表示对数据集的读取规则，=True则在不同的训练周期会将数据打乱顺序读取
- 代码：

batch_size表示一个batch中元素的个数，
batch_size越大表示一次训练中训练的样本个数越多，训练得到的效果越好，
因为一个包中包含的数据种类越多，结果越拟合于期望结果。但BZ太小，一个包只包含一种类型的数据集，
不会得到一个全域的最优解。

```
""" DataLoader的使用
# 0.下载数据集
# 1.设定batch_size的大小，加载数据集
#torchvision.datasets.MNIST(root="./MNIST",

train=True, transform=torchvision.transforms.ToTenso
# 2.读取不同batch_size的数据集中的数据
# DataLoader(batch_size=6_0000,20,1)
# iteration = samples / batch_size
# 3.读取不同shuffle模式的数据集中的数据
# DataLoader(shuffle=True)
"""
```



元素处理中常用的类Class

- SummaryWriter
- 来源：`torch.utils.tensorboard`
- 作用：创建文件,并保存数据集到文件.利用这种方法可以在训练循环
- 使用方法: file = SummaryWriter(log_dir="log_func",)
- 文件中可以写入scalar , Img, audio, text
- 代码:

```
"""保存函数的文件
# 0.设置环境
# from torch.utils.tensorboard import SummaryWriter
# 1.定义文件
# 2.文件中写入函数
# 3.写入完毕
# 4.tensorboard查看
# 3.用tensorboard --logdir=D:\WZY\Pytorch_Audio\Chapter2\log_func 网页查看
# 最好使用绝对路径
```

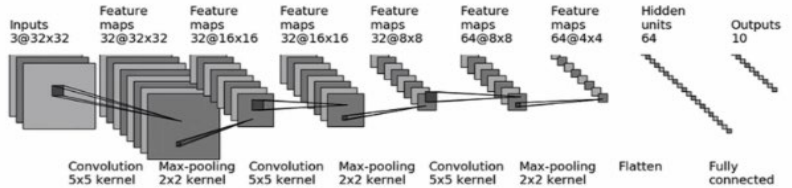
```
"""保存音频的文件
# 0.设置环境
# from torch.utils.tensorboard import SummaryWriter
# 1.定义文件
# 2.加载音频
# 3.文件中写入音频
# 3.写入完毕
# 4.tensorboard查看
# 用tensorboard --logdir=D:\WZY\Pytorch_Audio\Chapter2\log_audio --port=1000 网页查看
# 最好使用绝对路径
"""

# torchvision.datasets.MNIST
# 3.文件中写入图像
#img,target = dataset_MNIST.data[i],dataset_MNIST.targets[i]
# 3.写入完毕
# 4.tensorboard查看
# 用tensorboard --logdir=D:\WZY\Pytorch_Audio\Chapter2\log_img --port=1000 网页查看
# 最好使用绝对路径
"""
```

元素处理中常用的类Class

- nn.Module
- 来源:' torch.nn'
- 作用: 构建神经网络的模板
- 代码:

```
"""定义网络模型的类
# 0.环境设置
# from torch import nn
# 1.继承nn.Module定义网络模型
# 2.该网络模型包含卷积层、池化层和线性层
from torch.nn import Conv2d,MaxPool2d,Linear
# 3.调用该模型得到输出
"""
```

- nn.Sequential
- 来源:' torch.nn'
- 作用: 构建神经网络的模板



- 代码:

```
"""定义网络模型的类
# 0.环境设置
# from torch import nn
# 1.继承nn.Sequential定义网络模型
# 2.按照参考图构建该网络模型
from torch.nn import
Conv2d,MaxPool2d,Linear
# 3.调用该模型得到输出 test =
# 4.将该模型写入文件中
# 5.用tensorboard --
logdir=D:\WZY\Pytorch_Audio\Chapter2
\log_nn --port=1000 网页查看
"""
```

库Library

什么是库Library？存储对象和类class的集合。

注意：需要提前下载库 `teiminal`->`conda activate pytorch`-> `pip install xxx`，安装好后在structure查看该库含有的函数。

常用的库

- `PIL`Python Imaging Library是Python平台事实上的图像处理标准库。
- `NumPy`（Numerical Python的缩写）是数组和矩阵的计算库。
- `torch.utils` torch中的工具库
- `torch.utils.data` torch中的数据库
- `torch.utils.tensorboard` torch中的tensorboard库，可以在tensorboard打开
- `transforms` torchvision中的转换器工具库

官方数据库

- **Yes no数据集**http://www.openslr.org/1/
- The archive "waves_yesno.tar.gz" contains 60 .wav files, sampled at 8 kHz. All were recorded by the same male speaker, in Hebrew. In each file, the individual says 8 words; each word is either the Hebrew for "yes" or "no", so each file is a random sequence of 8 yes-es or noes. There is no separate transcription provided; the sequence is encoded in the filename, with 1 for yes and 0 for no
- Torchvision 图像数据集 https://pytorch.org/vision/stable/index.html
- Torchaudio 音频数据集https://pytorch.org/audio/stable/index.html