

## 一 安装

## 1.1 命令获取

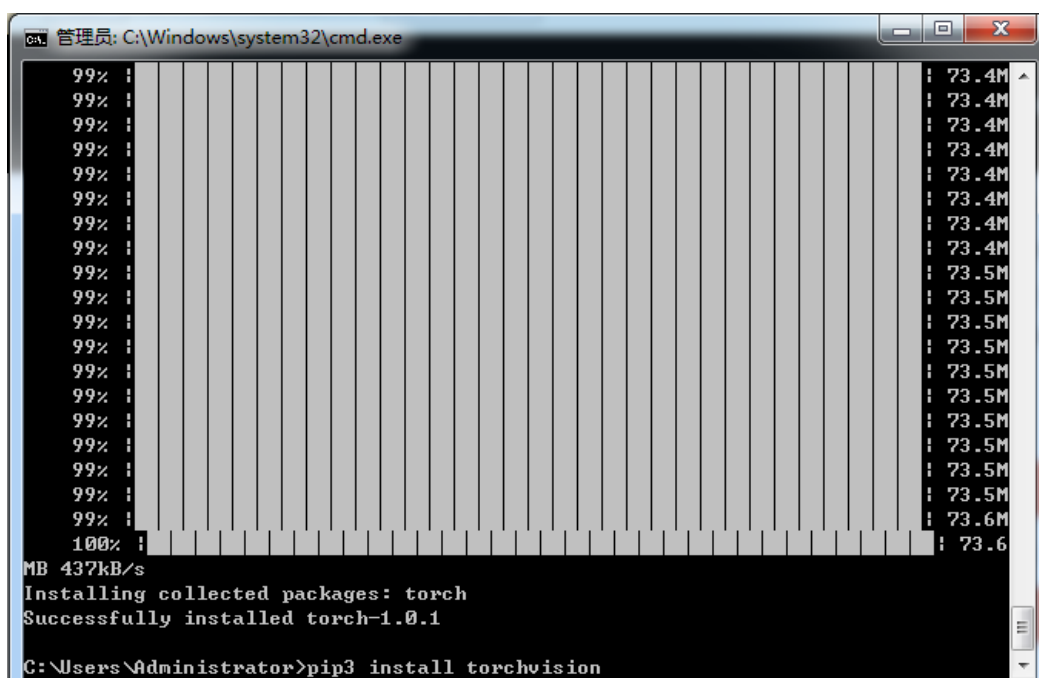
进入 PyTorch 官网，依次选择你电脑的配置（我这里已经下载了 python3.7），这里提供使用 pip 和 conda 两种环境下安装的步骤截图

(1) 使用 pip : windows+pip+python3.7+None

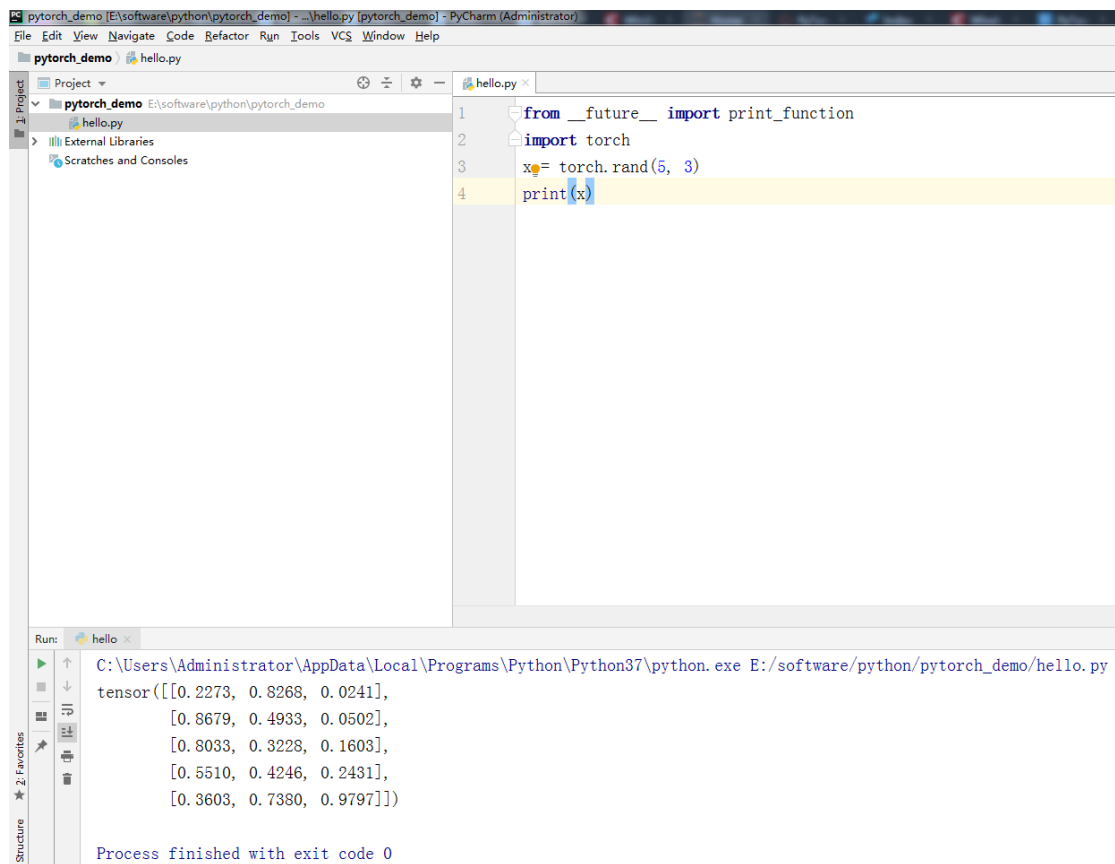
PyTorch Build	Stable (1.0)		Preview (Nightly)	
Your OS	Linux	Mac	Windows	
Package	Conda	Pip	LibTorch	Source
Language	Python 2.7	Python 3.5	Python 3.6	Python 3.7
CUDA	8.0	9.0	10.0	None
Run this Command:	<pre>pip3 install https://download.pytorch.org/whl/cpu/torch-1.0.1-cp37-cp37m-w in_amd64.whl pip3 install torchvision</pre>			

Previous versions of PyTorch

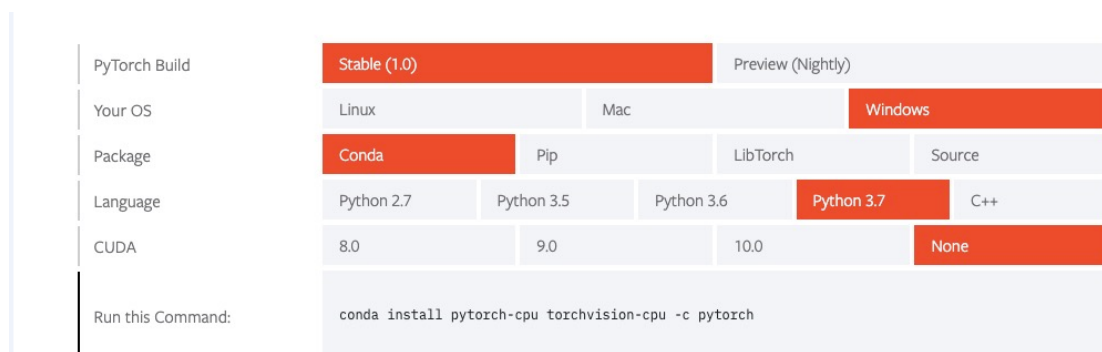
拷贝给出的命令在 cmd 下运行



安装成功后检验是否安装成功，打开 pycharm 运行一个小 demo：



(2) 使用 conda : windows+conda+python3.7+None



拷贝给出的命令在 cmd 下运行

```
C:\Windows\system32\cmd.exe - conda install pytorch-cpu torchvision-cpu -c pyto...

pycurl:          7.43.0.2-py37h74b6da3_0      --> 7.43.0.2-py37h7a1dbc1_0

qt:              5.9.6-vc14h1e9a669_2        --> 5.9.7-vc14h73c81de_0

sqlite:          3.24.0-h7602738_0           --> 3.28.0-he774522_0

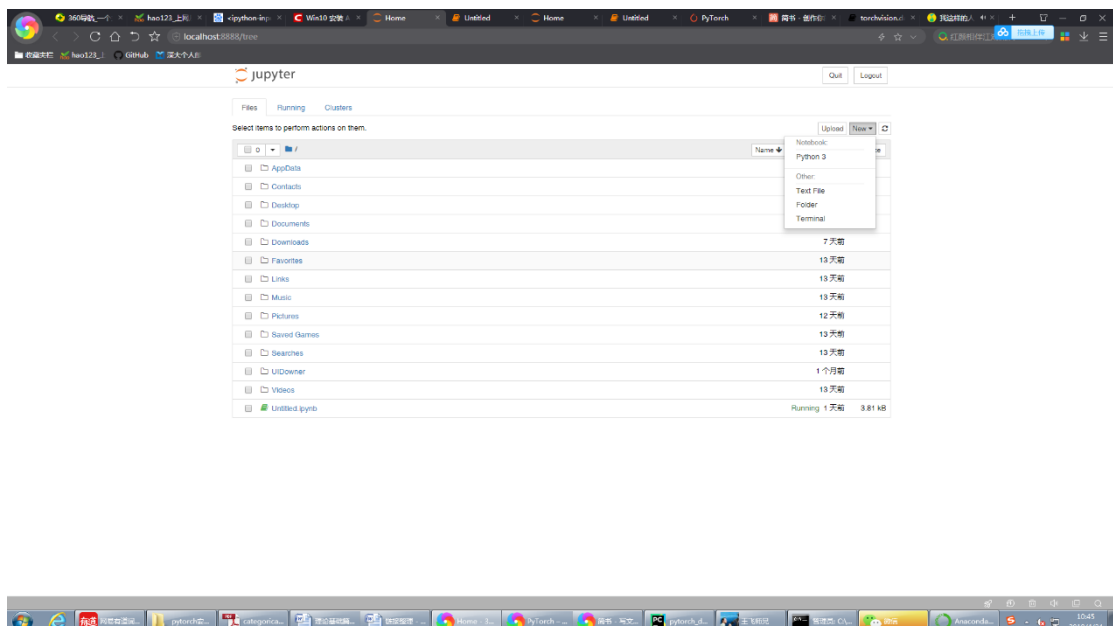
The following packages will be DOWNGRADED:

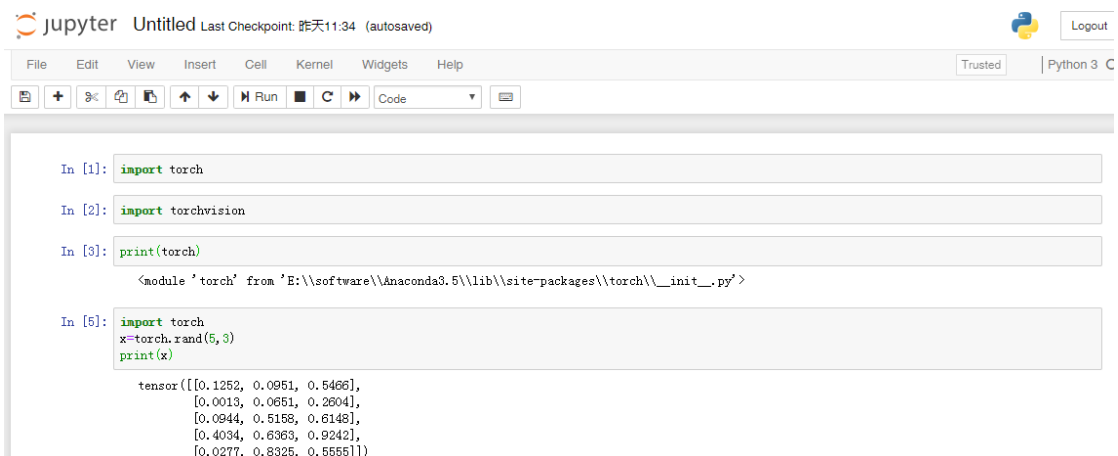
mkl:             2019.0-118                  --> 2018.0.3-1

Proceed <[y]/n>? y

Downloading and Extracting Packages
ca-certificates-2019 ! 158 KB ! ##### ! 100%
qt-5.9.7              ! 92.3 MB ! ##### ! 100%
mkl-2018.0.3          ! 178.1 MB ! ##### ! 100%
cryptography-2.6.1   ! 561 KB ! ##### ! 100%
sqlite-3.28.0         ! 945 KB ! ##### ! 100%
mkl_fft-1.0.6         ! 120 KB ! ##### ! 100%
curl-7.64.1           ! 120 KB ! ##### ! 100%
libpng-1.6.37         ! 598 KB ! ##### ! 100%
pytorch-cpu-1.0.1     ! 59.0 MB ! ##### ! 14%
```

安装完毕后，验证是否安装成功，打开 Anaconda 的 Jupyter 新建 python 文件，运行 demo：





The image shows a Jupyter Notebook interface with the title 'Untitled' and a last checkpoint from '昨天11:34 (autosaved)'. The top bar includes a 'Logout' button and a 'Python 3' environment indicator. The notebook contains five input cells. The first three cells import 'torch' and 'torchvision', and print the 'torch' module path. The fourth cell imports 'torch', generates a random tensor of size (5, 3), and prints it. The output shows a tensor with five rows and three columns of floating-point values.

```
In [1]: import torch

In [2]: import torchvision

In [3]: print(torch)

<module 'torch' from 'E:\\software\\Anaconda3.5\\lib\\site-packages\\torch\\__init__.py'>

In [5]: import torch
x=torch.rand(5,3)
print(x)

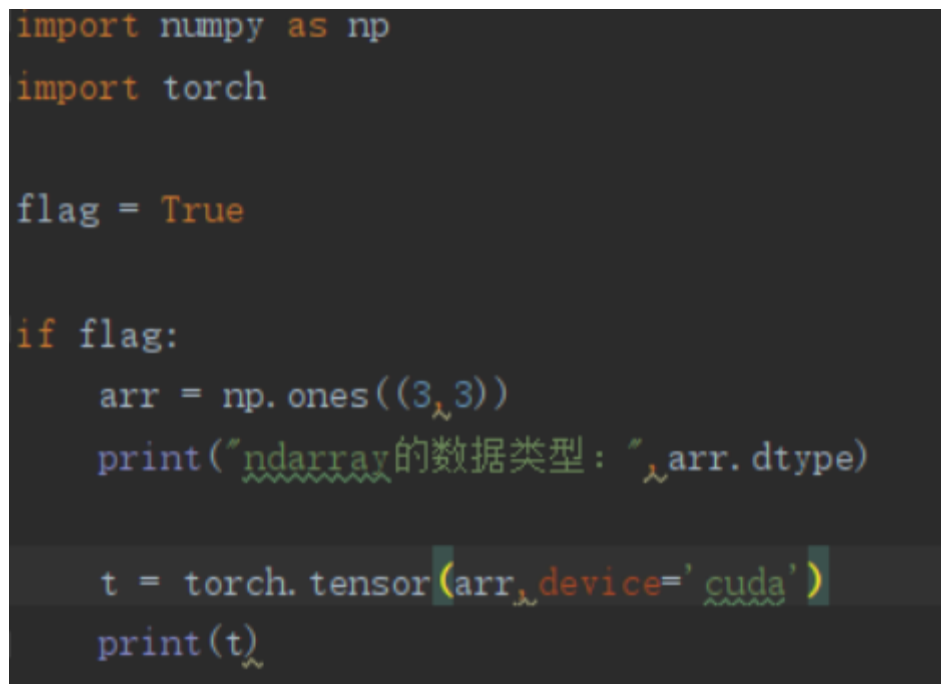
tensor([[0.1252, 0.0951, 0.5466],
        [0.0013, 0.0651, 0.2604],
        [0.0944, 0.5158, 0.6148],
        [0.4034, 0.6363, 0.9242],
        [0.0277, 0.8325, 0.5555]])
```

出现这个结果，那么恭喜你，至此 PyTorch1.0 & Anaconda3.5 已经安装成功。

## 二 . Tensors (张量)

### 2.1 创建张量

#### 1.直接创建



The image shows a code snippet for creating a tensor from a NumPy array. It imports 'numpy' as 'np' and 'torch'. A variable 'flag' is set to 'True'. An if-statement checks 'flag' and creates a NumPy array 'arr' of ones with shape (3, 3). It prints the dtype of 'arr'. Then, it creates a PyTorch tensor 't' from 'arr' with device='cuda' and prints it.

```
import numpy as np
import torch

flag = True

if flag:
    arr = np.ones((3,3))
    print("ndarray的数据类型: ", arr.dtype)

    t = torch.tensor(arr, device='cuda')
    print(t)
```

#### 2. 从一个 Numpy 数组来创建张量

```

flag = True
# flag = False

if flag:
    arr = np.array([[1,2,3],[4,5,6]])
    t = torch.from_numpy(arr)
    print("numpy array:", arr)
    print("tensor:", t)

    print("\n修改arr")
    arr[0,0] = 0
    print("numpy array:", arr)
    print("tensor :", t)

```

3. 依据数值创建 Numpy 数组

```

flag = True
# flag = False

if flag:
    out_t = torch.tensor([1])

    #out的功能是将下面这个函数生成的张量赋给out
    t = torch.zeros((3,3), out=out_t)

    print(t, '\n', out_t)
    print(id(t), id(out_t), id(t)==id(out_t)) #id的作用是求地址

```

2.2 基本运算

1. 加减乘除

```
torch.add(input, alpha=1, other, out=None)
torch.addcdiv(tensor, value=1, tensor1, tensor2, out=None)
torch.addcmul(tensor, value=1, tensor1, tensor2, out=None)
torch.sub(input, other, out=None)
torch.mul(input, other, out=None)
torch.div(input, other, out=None)
```

## 2. 对数，指数，幂函数

```
torch.log(input, out=None) ——以e为底
torch.log10(input, out=None)
torch.log2(input, out=None)
torch.exp(input, out=None)
torch.pow(input, exponent, out=None)
```

## 3. 三角函数

```
torch.acos(input, out=None)
torch.cosh(input, out=None)
torch.cos(input, out=None)
torch.asin(input, out=None)
torch.sinh(input, out=None)
torch.sin(input, out=None)
torch.atan(input, out=None)
torch.tanh(input, out=None)
torch.tan(input, out=None)
torch.atan2(input, other, out=None)
```

## 4. 绝对值

```
torch.abs(input, out=None)
```

## 三. 自动求导机制

Pytorch 会根据计算过程来自动生成动态计算图，然后可以根据动态图的创建过程进行反向传播，计算得到每个节点的梯度值。

### 3.1 张量本身 grad\_fn

为了能记录张量的梯度，首先需要在张量创建的时候设置 `requires_grad=True`。  
对于 pytorch 来说，每一个张量都有一个 `grad_fn` 方法，这个方法包含着创建该张量的运算的导数信息。本身携带计算图的信息，该方法还有一个 `next_functions` 属性，包含链接该张量的其他张量的 `grad_fn`。

### 3.1.1 torch.autograd

Pytorch 提供了一个专门用来做自动求导的包，`torch.autograd`。

包含 2 个重要函数：

`torch.autograd.backward`

这个函数通过传入根节点张量，以及初始梯度张量，可以计算产生该根节点所对应的叶子节点的梯度。

### 3.1.2 torch.autograd.grad

在某些情况下，我们并不要求出当前张量对所有产生该张量的叶子节点的梯度，这时候我们可以使用 `torch.autograd.grad` 方法。

## 4 自定义激活函数和梯度

仅仅使用模块有时候是不能满足我们需要效果的。我们需要自定义激活函数，在激活函数中定义前向传播和反向传播的代码来实现自己的需求。

### 4.1 类及方法

Pytorch 自定义激活函数继承于 `torch.autograd.Function`，其内部有 2 个静态方法：`forward` 和 `backward`

```
class Func(torch.autograd.Function):
    @staticmethod
    def forward(ctx,input):
        return result

    @staticmethod
    def backward(ctx,grad_output):
        return grad_output
```

### 4.2 实例

```
swish =Swish.apply #获得激活函数
torch.autograd.gradcheck(
    swish,torch.randn(
        10,requires_grad =True,
        dtype =torch.double)
)
#测试反向传播，正常返回值为 True

class Swish(torch.autograd.Function):
    @staticmethod
    def forward(ctx,input):
        ctx.input =input
        return input*torch.sigmoid(1*input) #假设b=1
    @staticmethod
    def backward(ctx,grad_output):
        ctx.input =input
        tmp = torch.sigmoid(1*input)

        return grad_output*(tmp +1 *input*tmp(1-tmp))
```

## 5 损失函数和优化器

### 5.1 损失函数

`torch.nn` 中存在很多封装好的损失函数。比如均方差损失，用 `torch.nn.MSELoss()` 表示。

```
import torch
import torch.nn as nn

# 初始化数据集
X = torch.tensor([1, 2, 3, 4], dtype=torch.float32)
Y = torch.tensor([2, 4, 6, 8], dtype=torch.float32)
w = torch.tensor(0.0, dtype=torch.float32, requires_grad=True)

def forward(x):
    # 正向传播函数
    return w * x

# 这里使用均方差损失计算预测值和真实值之间的距离
loss = nn.MSELoss()
# 测试此时的损失
loss(forward(X), Y)

# 输出结果
# tensor(30., grad_fn=<MseLossBackward>)
```

### 5.2 优化器

优化器可以理解为一种利用梯度下降算法自动求解所需参数的工具包。在 PyTorch 中提供了 `torch.optim` 方法优化我们的模型。`torch.optim` 工具包中存在着各种梯度下降的改进算法，比如 SGD、Momentum、RMSProp 和 Adam 等。

例如 SGD 优化器定义如下：

```
optimizer = torch.optim.SGD([w], lr=learning_rate)
```

## 6 模型的保存与加载

当保存和加载模型时，需要熟悉三个核心功能：

1. `torch.save`：将序列化对象保存到磁盘。此函数使用 Python 的 `pickle` 模块进行序列化。使用此函数可以保存如模型、tensor、字典等各种对象。
2. `torch.load`：使用 `pickle` 的 `unpickling` 功能将 `pickle` 对象文件反序列化到内存。此功能还可以有助于设备加载数据。
3. `torch.nn.Module.load_state_dict`：使用反序列化函数 `state_dict` 来加载模型的参数字典。

### 6.1 保存/加载 `state_dict`（推荐使用）

保存：

```
torch.save(model.state_dict(), PATH)
```

加载：

```
model = TheModelClass(*args, **kwargs)
model.load_state_dict(torch.load(PATH))
model.eval()
```

### 6.2 保存/加载完整模型

保存：



```
torch.save(model, PATH)
```

加载：

```
# 模型类必须在此之前被定义
model = torch.load(PATH)
model.eval()
```

### 6.3 保存和加载 Checkpoint 用于推理/继续训练

保存：

```
torch.save({
    'epoch': epoch,
    'model_state_dict': model.state_dict(),
    'optimizer_state_dict': optimizer.state_dict(),
    'loss': loss,
    ...
}, PATH)
```

加载：

```
model = TheModelClass(*args, **kwargs)
optimizer = TheOptimizerClass(*args, **kwargs)

checkpoint = torch.load(PATH)
model.load_state_dict(checkpoint['model_state_dict'])
optimizer.load_state_dict(checkpoint['optimizer_state_dict'])
epoch = checkpoint['epoch']
loss = checkpoint['loss']

model.eval()
# - or -
model.train()
```

### 6.4 在一个文件中保存多个模型

保存：

```
torch.save({
    'modelA_state_dict': modelA.state_dict(),
    'modelB_state_dict': modelB.state_dict(),
    'optimizerA_state_dict': optimizerA.state_dict(),
    'optimizerB_state_dict': optimizerB.state_dict(),
    ...
}, PATH)
```

加载：

```

modelA = TheModelAClass(*args, **kwargs)
modelB = TheModelBClass(*args, **kwargs)
optimizerA = TheOptimizerAClass(*args, **kwargs)
optimizerB = TheOptimizerBClass(*args, **kwargs)

checkpoint = torch.load(PATH)
modelA.load_state_dict(checkpoint['modelA_state_dict'])
modelB.load_state_dict(checkpoint['modelB_state_dict'])
optimizerA.load_state_dict(checkpoint['optimizerA_state_dict'])
optimizerB.load_state_dict(checkpoint['optimizerB_state_dict'])

modelA.eval()
modelB.eval()
# - or -
modelA.train()
modelB.train()

```

## 7.数据可视化

TensorBoard 是一个数据可视化工具，能够直观地显示深度学习过程中张量的变化，从这个变化中就可以很容易地了解到模型在训练中的行为，包括但不限于损失函数的下降趋势是否合理、张量分量的分布是否在训练中发生变化以及输出训练过程中的图片等等。

### 7.1TensorBoard 安装

```
pip install tensorboard
```

注意：tensorboard 版本号要和 tensorflow 版本匹配

### 7.2 使用方法

以线性模型的训练为例，以下为完整代码：

```

from sklearn.datasets import load_boston
from torch.utils.tensorboard import SummaryWriter
import torch
import torch.nn as nn

boston = load_boston()
lm = nn.Linear(13, 1)
criterion = nn.MSELoss()
optim = torch.optim.SGD(lm.parameters(), lr=1e-6)
data = torch.tensor(boston['data'], requires_grad=True, dtype=torch.float32)
target = torch.tensor(boston['target'], dtype=torch.float32)

writer = SummaryWriter()

for step in range(10000):
    predict = lm(data)
    loss = criterion(predict, target)

    writer.add_scalar("loss/train", loss, step)
    if step and step % 1000:
        print("Loss : {:.3f}".format(loss.item()))

    optim.zero_grad()
    loss.backward()
    optim.step()

```

1.运行成功后，可以看到当前目录下多了一个文件夹 **runs**，其中记录的就是你添加的需要写入摘要的信息。如果此处运行失败，可以考虑两个方面的原因：

(1) tensorboard 和 tensorflow 的版本是否匹配

(2) 如果出现 `AttributeError: module 'tensorflow' has no attribute 'io'` 的问题，则可以尝试通过 `pip install tensorflow-io` 安装 tensorflow-io，之后再运行程序。

2.在当前目录下，执行 **tensorboard --logdir=runs** 命令，即可看到以下页面：

```
root@1c374f45274e:/tlliu/learn/tensorboard# tensorboard --logdir=runs
2021-02-21 02:47:18.344451: W tensorflow/stream_executor/platform/default/dso_loader.cc:60] Could not load dynamic library 'libnvidia-rtx.so.460.38': No such file or directory; LD_LIBRARY_PATH: /usr/local/lib:/usr/local/lib:/u
2021-02-21 02:47:18.344514: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cud
Serving TensorBoard on localhost; to expose to the network, use a proxy or pass --bind_all
TensorBoard 2.4.1 at http://localhost:6006/ (Press CTRL+C to quit)
```

3. 打开浏览器，输入 `http://localhost:6006`，可以看到如下界面，展示了训练过程中的 loss 变化。

