使用说明

测试数据和更多使用方法请联系QQ群173640919，在群文件中有 其他更多程序请访问 https://gitee.com/cangyeone/seismological-ai-tools

In [1]:

```
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:

```
import torch
import torch.nn as nn
```

In [3]:

```
from models.UNet import UNet, Loss as ULoss # PhaseNet
# 数据长度3000,6000 不是2的整数倍
# 建议数据长度3072， 6144
model = UNet()
model.train() # 调整为训练模式
model.load_state_dict(torch.load("ckpt/diting.unet.pt", map_location="cpu"))
```

Out[3]:

```
<All keys matched successfully>
```

In [4]:

```
wave = torch.randn([100, 3, 1024])#[100个样本，每个样本3个分量，6144采样点]
y = model(wave)
print(y.shape)
```

```
torch.Size([100, 3, 1024])
```

数据读取部分

- windows系统建议使用Thread
- thread并行加速效果不明显

In [5]:

```
from utils.data import DitingData, DitingDataThread # 多线程读取程序，ThreadWin兼容
```

In [6]:

```
datatool = DitingDataThread(file_name="data/diting.h5", n_length=6144, stride=16, pa
```

In [7]:

```
x1, x2, x3, x4 = datatool.batch_data()
# 波形数据，LPPN标签，PhaseNet标签，EQT标签
print(x1.shape, x2.shape, x3.shape, x4.shape)
```

```
(32, 3, 6144) (32, 2, 384) (32, 3, 6144) (32, 3, 6144)
```

In [8]:

```
plt.plot(x1[0, 1], c="k")
plt.plot(x3[0, 0], c="g")
plt.plot(x3[0, 1], c="b")
plt.plot(x3[0, 2], c="r")
```

Out[8]:

```
[<matplotlib.lines.Line2D at 0x2be3fa1c3a0>]
```

模型训练

- 将训练数据输入到模型中即可

In [9]:

```
from models.UNet import UNet, Loss as ULoss
model = UNet()
model.train() # 训练模式
model.load_state_dict(torch.load("ckpt/diting.unet.pt", map_location="cpu"))
lossfn = ULoss()
optim = torch.optim.Adam(model.parameters(), 1e-5)

for step in range(10):
    x1, x2, x3, x4 = datatool.batch_data()
    wave = torch.tensor(x1, dtype=torch.float32) # 波形
    label = torch.tensor(x3, dtype=torch.float32) # 标签
    y = model(wave) # 预测结果[N, 3, 6144]
    loss = lossfn(y, label)
    loss.backward()  # 反向传播计算梯度
    optim.step()     # 执行梯度下降法
    optim.zero_grad()# 将梯度置零
    if step %2==0:
        print(step, loss)
        torch.save(model.state_dict(), "ckpt/unet.temp.pt")
```

```
0 tensor(115957.5625, grad_fn=<NegBackward0>)
2 tensor(111537.0547, grad_fn=<NegBackward0>)
4 tensor(115598.4375, grad_fn=<NegBackward0>)
6 tensor(108046.5078, grad_fn=<NegBackward0>)
8 tensor(113523.4219, grad_fn=<NegBackward0>)
```

迭代次数问题

- 迭代5000次已经接近最终精度
- 迭代20000次最终精度

连续数据拾取

In [10]:

```
import torch
jitmodel = torch.jit.load("ckpt/china.rnn.jit")
```

In [11]:

```
x = torch.randn([8640000, 3])
y = jitmodel(x)
print(y)
```

```
tensor([], size=(0, 3))
```

实际处理流程

In [12]:

```
import torch
import torch.nn as nn
from models.UNet import UNet
class Picker(UNet):
    def __init__(self):
        super().__init__()
        self.n_stride = 1
    def forward(self, x):
        device = x.device
        with torch.no_grad():
            #print("数据维度", x.shape)
            T, C = x.shape
            seqlen = 6144
            batchstride = 6144 - 256
            batchlen = torch.ceil(torch.tensor(T / batchstride).to(device))
            idx = torch.arange(0, seqlen, 1, device=device).unsqueeze(0) + torch.ar
            idx = idx.clamp(min=0, max=T-1).long()
            x = x.to(device)
            wave = x[idx, :]
            wave = wave.permute(0, 2, 1)
            wave -= torch.mean(wave, dim=2, keepdim=True)
            max, maxidx = torch.max(torch.abs(wave), dim=2, keepdim=True)
            wave /= (max + 1e-6)
            x = wave.unsqueeze(3)
            x = self.inputs(x)
            x1 = self.layer0(x)
            x2 = self.layer1(x1)
            x3 = self.layer2(x2)
            x4 = self.layer3(x3)
            x5 = self.layer4(x4)
            x6 = self.layer5(x5)
            x6 = torch.cat([x4, x6], dim=1) # 加入skip connection
            x7 = self.layer6(x6)
            x7 = torch.cat([x3, x7], dim=1) # 加入skip connection
            x8 = self.layer7(x7)
            x8 = torch.cat([x2, x8], dim=1) # 加入skip connection
            x9 = self.layer8(x8)
            x9 = torch.cat([x1, x9], dim=1) # 加入skip connection
            x10 = self.layer9(x9)
            x10 = x10.softmax(dim=1)
            oc = x10.squeeze(dim=3)
            B, C, T = oc.shape
            tgrid = torch.arange(0, T, 1, device=device).unsqueeze(0) * self.n_stri
            oc = oc.permute(0, 2, 1).reshape(-1, C)
            ot = tgrid.squeeze()
            ot = ot.reshape(-1)
            output = []
            #print("NN处理完成", oc.shape, ot.shape)
            # 接近非极大值抑制（NMS）
            # .......P........S......
            for itr in range(2):
                pc = oc[:, itr+1]
                time_sel = torch.masked_select(ot, pc>0.3)
                score = torch.masked_select(pc, pc>0.3)
                _, order = score.sort(0, descending=True)    # 降序排列
                ntime = time_sel[order]
                nprob = score[order]
                #print(batchstride, ntime, nprob)
                select = -torch.ones_like(order)
                selidx = torch.arange(0, order.numel(), 1, dtype=torch.long, device=
                count = 0
                while True:
                    if nprob.numel()<1:
                        break
                    ref = ntime[0]
                    idx = selidx[0]
                    select[idx] = 1
                    count += 1
                    selidx = torch.masked_select(selidx, torch.abs(ref-ntime)>1000)
                    nprob = torch.masked_select(nprob, torch.abs(ref-ntime)>1000)
                    ntime = torch.masked_select(ntime, torch.abs(ref-ntime)>1000)
                p_time = torch.masked_select(time_sel[order], select>0.0)
                p_prob = torch.masked_select(score[order], select>0.0)
                p_type = torch.ones_like(p_time) * itr
                y = torch.stack([p_type, p_time, p_prob], dim=1)
                output.append(y)
            y = torch.cat(output, dim=0)
        return y

model = Picker()
torch.jit.save(torch.jit.script(model), "unet.jit.temp")
x = torch.randn([300000, 3])
```

```
y = model(x)
print(y)
```

```
tensor([[0.0000e+00, 2.3830e+05, 8.4935e-01],
        [0.0000e+00, 5.8863e+04, 8.4638e-01],
        [0.0000e+00, 8.4928e+04, 8.3707e-01],
        ...,
        [1.0000e+00, 2.6463e+05, 4.5571e-01],
        [1.0000e+00, 5.9627e+04, 4.2730e-01],
        [1.0000e+00, 2.3480e+05, 3.4589e-01]])
```

直接拾取示意

In [14]:

```
import torch # 机器学习库， conda install pytorch
import numpy as np # 矩阵计算
import matplotlib.pyplot as plt # 绘图
# 加载震相拾取模型
# ckpt/china.rnn.single.jit 为单分量拾取，当前为三分量拾取
model = torch.jit.load("ckpt/china.rnn.jit")
# 读取数据，使用obspy，必须要100Hz
import obspy # pip install obspy
st1 = obspy.read("data/waveform/X1.53085.01.BHE.D.20122080726235953.sac")
st2 = obspy.read("data/waveform/X1.53085.01.BHN.D.20122080726235953.sac")
st3 = obspy.read("data/waveform/X1.53085.01.BHZ.D.20122080726235953.sac")
data = [st1[0].data, st2[0].data, st3[0].data]
# 任意长度数据均可
data = np.stack(data, axis=1) #[N, 3]->一天 [8640000]100Hz
## 开始拾取
with torch.no_grad():# 不需要计算梯度
    x = torch.tensor(data, dtype=torch.float32) # 转化为Tensor
    y = model(x)#y-tensor, -> ndarray
    y = y.cpu().numpy()#[K, 3]-[类型，相对到时，置信度]
print(y)
plt.plot(data[:, 2], c="k") # 波形
for pha in y:
    if pha[0]==0:
        c = "r"
    else:
        c = "b"
    plt.axvline(pha[1], c=c)
plt.show()
```

```
[[0.0000000e+00 9.6320000e+03 4.5449382e-01]
 [1.0000000e+00 1.2247000e+04 3.2736662e-01]]
```

```
C:\Users\cangy\AppData\Local\Temp\ipykernel_17168\1765068556.py:28: UserWarning: Mat
plotlib is currently using agg, which is a non-GUI backend, so cannot show the figur
e.
  plt.show()
```