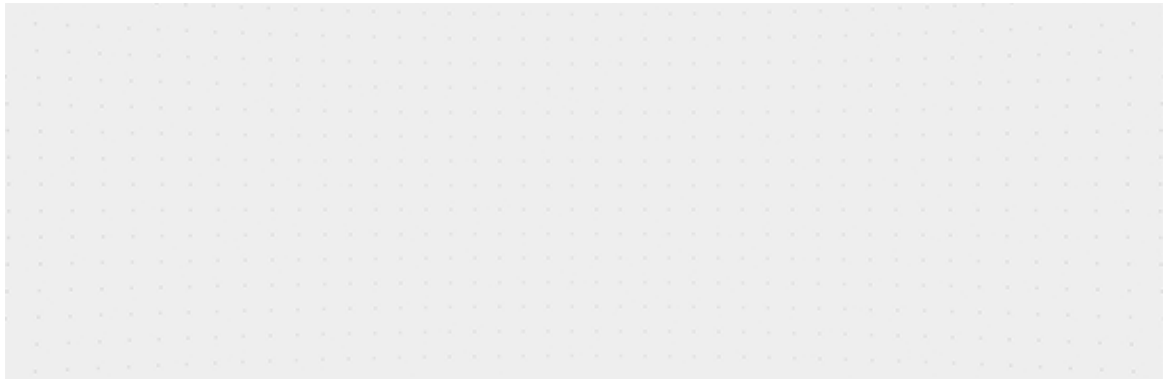


实操教程 | Pytorch-lightning的使用

CV开发者都爱看的 极市平台 2021-05-17 22:00:00 手机阅读 𑀓

↑ 点击蓝字 关注极市平台



作者 | Caliber@知乎 (已授权)

来源 | <https://zhuanlan.zhihu.com/p/370185203>

编辑 | 极市平台

极市导读

Pytorch-lightning可以非常简洁得构建深度学习代码。但是其实大部分人用不到很多复杂得功能，并且用的时候稍微有一些不灵活。本文作者分享了自己在使用时的一些心得，附有代码链接。 >>加入极市CV技术交流群，走在计算机视觉的最前沿

Pytorch-lightning(以下简称pl)可以非常简洁得构建深度学习代码。但是其实大部分人用不到很多复杂得功能。而pl有时候包装得过于深了，用的时候稍微有一些不灵活。通常来说，在你的模型搭建好之后，大部分的功能都会被封装在一个叫trainer的类里面。一些比较麻烦但是需要的功能通常如下：

1. 保存checkpoints
2. 输出log信息
3. resume training 即重载训练，我们希望可以接着上一次的epoch继续训练
4. 记录模型训练的过程(通常使用tensorboard)
5. 设置seed，即保证训练过程可以复制

好在这些功能在pl中都已经实现。

由于doc上的很多解释并不是很清楚，而且网上例子也不是特别多。下面分享一点我自己的使用心得。

首先关于设置全局的种子：

```
1 from pytorch_lightning import seed_everything
2
3 # Set seed
4 seed = 42
5 seed_everything(seed)
```

只需要import如上的seed_everything函数即可。它应该和如下的函数是等价的：

```
1 def seed_all(seed_value):
2     random.seed(seed_value) # Python
3     np.random.seed(seed_value) # cpu vars
4     torch.manual_seed(seed_value) # cpu vars
5
6     if torch.cuda.is_available():
7         print ('CUDA is available')
8         torch.cuda.manual_seed(seed_value)
9         torch.cuda.manual_seed_all(seed_value) # gpu vars
10        torch.backends.cudnn.deterministic = True #needed
11        torch.backends.cudnn.benchmark = False
12
13 seed=42
14 seed_all(seed)
```

但经过我的测试，好像pl的seed_everything函数应该更全一点。

下面通过一个具体的例子来说明一些使用方法：

先下载、导入必要的包和下载数据集：

```
1 !pip install pytorch-lightning
2 !wget https://download.pytorch.org/tutorial/hymenoptera_data.zip
```

```
3 !unzip -q hymenoptera_data.zip
4 !rm hymenoptera_data.zip
5
6 import pytorch_lightning as pl
7 import os
8 import numpy as np
9 import random
10 import matplotlib.pyplot as plt
11
12 import torch
13 import torch.nn.functional as F
14 import torchvision
15 import torchvision.transforms as transforms
```

以下代码种加入!的代码是在terminal中运行的。在google colab中运行linux命令需要在之前加!

如果是使用google colab，由于它创建的是一个虚拟机，不能及时保存，所以如果需保存，挂载自己google云盘也是有必要的。使用如下的代码：

```
1 from google.colab import drive
2 drive.mount('./content/drive')
3
4 import os
5 os.chdir("/content/drive/My Drive/")
```

先如下定义如下的LightningModule和main函数。

```
1 class CoolSystem(pl.LightningModule):
2
3     def __init__(self, hparams):
4         super(CoolSystem, self).__init__()
5
6         self.params = hparams
7
8         self.data_dir = self.params.data_dir
```

```
9         self.num_classes = self.params.num_classes
10
11         ##### define the model #####
12         arch = torchvision.models.resnet18(pretrained=True)
13         num_fters = arch.fc.in_features
14
15         modules = list(arch.children())[:-1] # ResNet18 has 10 children
16         self.backbone = torch.nn.Sequential(*modules) # [bs, 512, 1, 1]
17         self.final = torch.nn.Sequential(
18             torch.nn.Linear(num_fters, 128),
19             torch.nn.ReLU(inplace=True),
20             torch.nn.Linear(128, self.num_classes),
21             torch.nn.Softmax(dim=1))
22
23     def forward(self, x):
24         x = self.backbone(x)
25         x = x.reshape(x.size(0), -1)
26         x = self.final(x)
27
28         return x
29
30     def configure_optimizers(self):
31         # REQUIRED
32         optimizer = torch.optim.SGD([
33             {'params': self.backbone.parameters()},
34             {'params': self.final.parameters(), 'lr': 1e-2}
35         ], lr=1e-3, momentum=0.9)
36
37         exp_lr_scheduler = torch.optim.lr_scheduler.StepLR(optimizer, st
38
39         return [optimizer], [exp_lr_scheduler]
40
41     def training_step(self, batch, batch_idx):
42         # REQUIRED
43         x, y = batch
44         y_hat = self.forward(x)
45
46         loss = F.cross_entropy(y_hat, y)
47
48         _, preds = torch.max(y_hat, dim=1)
```

```
49         acc = torch.sum(preds == y.data) / (y.shape[0] * 1.0)
50
51         self.log('train_loss', loss)
52         self.log('train_acc', acc)
53
54         return {'loss': loss, 'train_acc': acc}
55
56
57     def validation_step(self, batch, batch_idx):
58         # OPTIONAL
59         x, y = batch
60         y_hat = self.forward(x)
61         loss = F.cross_entropy(y_hat, y)
62         _, preds = torch.max(y_hat, 1)
63         acc = torch.sum(preds == y.data) / (y.shape[0] * 1.0)
64
65         self.log('val_loss', loss)
66         self.log('val_acc', acc)
67
68         return {'val_loss': loss, 'val_acc': acc}
69
70
71     def test_step(self, batch, batch_idx):
72         # OPTIONAL
73         x, y = batch
74         y_hat = self.forward(x)
75         loss = F.cross_entropy(y_hat, y)
76         _, preds = torch.max(y_hat, 1)
77         acc = torch.sum(preds == y.data) / (y.shape[0] * 1.0)
78
79         return {'test_loss': loss, 'test_acc': acc}
80
81
82     def train_dataloader(self):
83         # REQUIRED
84
85         transform = transforms.Compose([
86             transforms.RandomResizedCrop(224),
87             transforms.RandomHorizontalFlip(),
88             transforms.ToTensor(),
```

```
89         transforms.Normalize([0.485, 0.456, 0.40
90     ])
91
92     train_set = torchvision.datasets.ImageFolder(os.path.join(self.d
93     train_loader = torch.utils.data.DataLoader(train_set, batch_size=
94
95     return train_loader
96
97     def val_data_loader(self):
98         transform = transforms.Compose([
99             transforms.Resize(256),
100             transforms.CenterCrop(224),
101             transforms.ToTensor(),
102             transforms.Normalize([0.485, 0.456, 0.40
103     ])
104
105     val_set = torchvision.datasets.ImageFolder(os.path.join(self.data_
106     val_loader = torch.utils.data.DataLoader(val_set, batch_size=32, s
107
108     return val_loader
109
110     def test_data_loader(self):
111         transform = transforms.Compose([
112             transforms.Resize(256),
113             transforms.CenterCrop(224),
114             transforms.ToTensor(),
115             transforms.Normalize([0.485, 0.456, 0.406]
116     ])
117
118     val_set = torchvision.datasets.ImageFolder(os.path.join(self.data_
119     val_loader = torch.utils.data.DataLoader(val_set, batch_size=8, sh
120
121     return val_loader
122
123
124
125
126
127     def main(hparams):
128         model = CoolSystem(hparams)
```

```
129
130
131     trainer = pl.Trainer(
132         max_epochs=hparams.epochs,
133         gpus=1,
134         accelerator='dp'
135     )
136
137     trainer.fit(model)
138
```

下面是run的部分：

```
1  from argparse import Namespace
2
3  args = {
4      'num_classes': 2,
5      'epochs': 5,
6      'data_dir': "/content/hymenoptera_data",
7  }
8
9  hyperparams = Namespace(**args)
10
11
12  if __name__ == '__main__':
13      main(hyperparams)
```

如果希望重载训练的话，可以按如下方式：

```
1  # resume training
2
3  RESUME = True
4
5  if RESUME:
6      resume_checkpoint_dir = './lightning_logs/version_0/checkpoints/'
7      checkpoint_path = os.listdir(resume_checkpoint_dir)[0]
8      resume_checkpoint_path = resume_checkpoint_dir + checkpoint_path
9
```

```
10
11     args = {
12         'num_classes': 2,
13         'data_dir': "/content/hymenoptera_data"}
14
15     hparams = Namespace(**args)
16
17     model = CoolSystem(hparams)
18
19
20     trainer = pl.Trainer(gpus=1,
21                          max_epochs=10,
22                          accelerator='dp',
23                          resume_from_checkpoint = resume_checkpoint_path)
24
25     trainer.fit(model)
```

如果我们想要从checkpoint加载模型，并进行使用可以按如下操作来：

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3
4 # functions to show an image
5 def imshow(inp):
6     inp = inp.numpy().transpose((1, 2, 0))
7     mean = np.array([0.485, 0.456, 0.406])
8     std = np.array([0.229, 0.224, 0.225])
9     inp = std * inp + mean
10    inp = np.clip(inp, 0, 1)
11    plt.imshow(inp)
12    plt.show()
13
14    classes = ['ants', 'bees']
15
16    checkpoint_dir = 'lightning_logs/version_1/checkpoints/'
17    checkpoint_path = checkpoint_dir + os.listdir(checkpoint_dir)[0]
18
19    checkpoint = torch.load(checkpoint_path)
```



```

20 model_infer = CoolSystem(hparams)
21 model_infer.load_state_dict(checkpoint['state_dict'])
22
23 try_dataloader = model_infer.test_dataloader()
24
25 inputs, labels = next(iter(try_dataloader))
26
27 # print images and ground truth
28 imshow(torchvision.utils.make_grid(inputs))
29 print('GroundTruth: ', ' '.join('%5s' % classes[labels[j]] for j in range
30
31 # inference
32 outputs = model_infer(inputs)
33
34 _, preds = torch.max(outputs, dim=1)
35 # print (preds)
36 print (torch.sum(preds == labels.data) / (labels.shape[0] * 1.0))
37
38 print('Predicted: ', ' '.join('%5s' % classes[preds[j]] for j in range(8))

```



GroundTruth: bees ants ants bees ants bees bees bees
 tensor(1.)

Predicted: bees ants ants bees ants bees bees bees

知乎 @Caliber

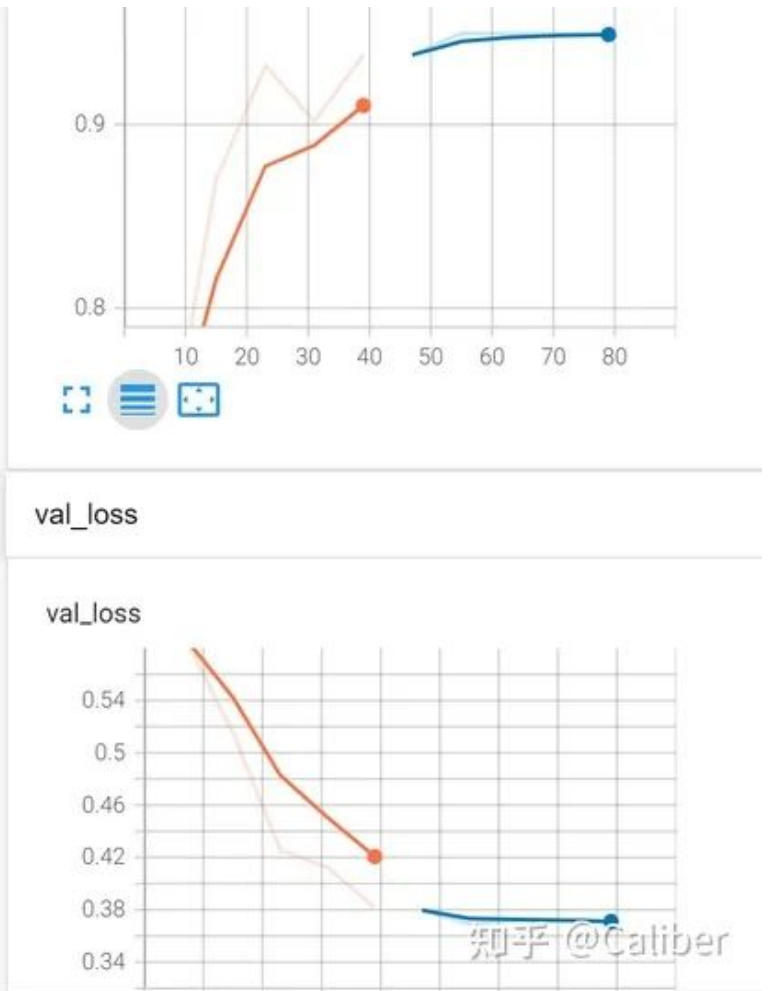
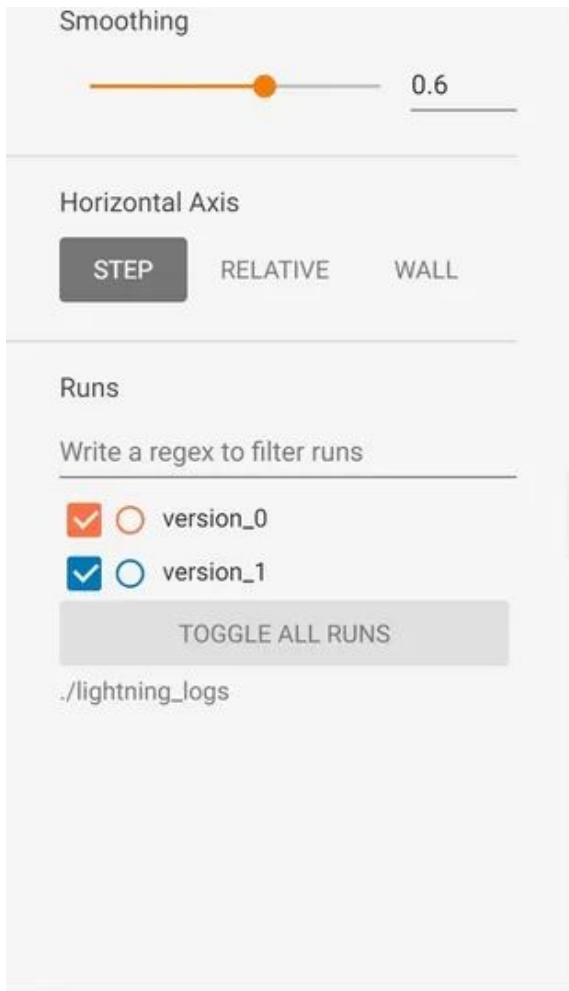
预测结果如上。

如果希望检测训练过程（第一部分+重载训练的部分），如下：

```

1 # tensorboard
2
3 %load_ext tensorboard
4 %tensorboard --logdir = ./lightning_logs

```



训练过程在tensorboard里面记录，version0是第一次的训练，version1是重载后的结果。

完整的code在这里.

https://colab.research.google.com/gist/calibertytz/a9de31175ce15f384dead94c2a9fad4d/pl_tutorials_1.ipynb

如果觉得有用，就请分享到朋友圈吧！



极市平台

为计算机视觉开发者提供全流程算法开发训练平台，以及大咖技术分享、社区交流、竞赛...
848篇原创内容

公众号

▲点击卡片关注极市平台，获取最新CV干货

公众号后台回复“目标检测”获取目标检测算法综述盘点~

极市干货

YOLO教程：一文读懂YOLO V5 与 YOLO V4 | 大盘点 | YOLO 系目标检测算法总览 | 全面解析 YOLO V4网络结构

实操教程：PyTorch vs LibTorch：网络推理速度谁更快？ | 只用两行代码，我让Transformer推理加速了50倍 | PyTorch AutoGrad C++层实现

算法技巧 (trick)：深度学习训练tricks总结（有实验支撑） | 深度强化学习调参Tricks合集 | 长尾识别中的Tricks汇总（AAAI2021）

最新CV竞赛：2021 高通人工智能应用创新大赛 | CVPR 2021 | Short-video Face Parsing Challenge | 3D人体目标检测与行为分析竞赛开赛，奖池7万+，数据集达16671张！



Qualcomm | Qualcomm ventures

2021

高通人工智能应用创新大赛

2021 Qualcomm AI Innovation Challenge

创新赛道-垃圾分类识别

28万奖励
免费云算力
40081张实景数据

扫码报名
竞赛时间:3月24日-5月20日

联合主办方
极视角 | mi | ThunderSoft | CSDN
开创AI视觉算法商城

开源技术合作伙伴
TensorFlow Lite

CV技术社群邀请函

.....

△长按添加极市小助手

添加极市小助手微信 (ID : cvmart2)

备注: 姓名-学校/公司-研究方向-城市 (如: 小极-北大-目标检测-深圳)

即可申请加入极市目标检测/图像分割/工业检测/人脸/医学影像/3D/SLAM/自动驾驶/超分辨率/姿态估计/ReID/GAN/图像增强/OCR/视频理解等技术交流群

每月大咖直播分享、真实项目需求对接、求职内推、算法竞赛、干货资讯汇总、与 10000+ 来自港科大、北大、清华、中科院、CMU、腾讯、百度等名校名企视觉开发者互动交流~

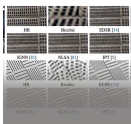
觉得有用麻烦给个在看啦~

阅读原文

喜欢此内容的人还喜欢

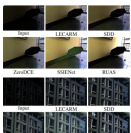
ICCV 2023 | 南开程明明团队提出适用于SR任务的新颖注意力机制 (已开源)

极市平台



ICCV23 | 将隐式神经表征用于低光增强, 北大张健团队提出NeRCo

极市平台



ICCV 2023 | Pixel-based MIM: 简单高效的多级特征融合自监督方法

极市平台

