PyTorch版和Tensorflow版对比

---

1. Shuffle

tf：设置1024大小的buffer size每次采用是随机从这个buffer中进行采样。

pt：对一个batch（256）的数据进行shuffle，然后采样。

2. Data Augmentation

tf:

在imagenet_preprocessing文件中，

训练增强策略是：随机翻转，然后resize到224，均值化；

验证的增强策略是：resize到224和center crop，均值化。

pt:

训练采用策略：

- 随机resizecrop到224
- 随机翻转
- 随机旋转15°以内
- Color Jitter
- Style PCA增强
- 均值化

验证集：

- resize到448
- center crop到224
- 均值化

3. Learning Rate / Optimizer

tf:

momentumOptimizer scheduler在epoch 30, 60, 80, and 90的时候lr/10，weight decay设置了5个值：decay_rates=[1, 0.1, 0.01, 0.001, 1e-4]

- 如果不使用warmup，那么base learning rate=0.1
- 使用warmup，那么base learning rate=0.128， warmup前5个epoch，当前global step在慢慢增加，知道5个epoch对应的step，所以learning rate会慢慢以线性的速度逼近原有base learning rate

$$\text{warmup learning rate} = \frac{\text{global step}}{\text{5 epoch step}} \times \text{base learning rate}$$

pt:

SGD momentum=0.9 weight_decay=1e-4 初始学习率0.1,scheduler在epoch 30,60,90的时候lr/10

scheduler warmup的bug已经修复，实现有问题。初始化learning rate是0.1。warmup结束后使用的是LambdaLR调节，具体方法是

$$\text{new learning rate} = \text{base learning rate} \times 0.1^{\lfloor \frac{epoch}{30} \rfloor}$$

相当于每30个epoch，lr变为原来10分之一

4. Loss

tf: cross entropy

pt: label smoothing

5. Batch Size

一般来说，0.1的learning rate对应的batch size为256

tf: 使用了等比例缩放的方法

$$\text{new learning rate} = \frac{\text{current batch}}{256} \times 0.1$$

pt: 64的batch size，accumulate 4次以后，也就是256个以后再loss.backward() ，设置初始leanring rate为0.1