1. 创建数据集

numpy在处理大量数据的场景下,读取速度比torch更快;

1. ndarray如何转化成tensor
2. batch\_size可以理解为是len(triples)的一部分
3. BatchNorm1d 只能接受2D张量

Linear->Relu->BatchNorm->Dropout

1. 把模型放到gpu上跑
2. Sequential和ModuleList的区别
3. Vit采用不统一图像尺寸，得到的二维特征第一维度通常是不一样的;

D: 通常是768、1024

**图像尺寸 224x224，补丁大小 16x16**:

* 图像分割后得到 14×14=19614 \times 14 = 19614×14=196 个补丁。
* 特征矩阵的形状是 (196,D)(196, D)(196,D)，其中 DDD 是每个补丁的特征维度（例如 768）。

**图像尺寸 128x128，补丁大小 16x16**:

* 图像分割后得到 8×8=648 \times 8 = 648×8=64 个补丁。
* 特征矩阵的形状是 (64,D)(64, D)(64,D)。

**图像尺寸 384x384，补丁大小 16x16**:

* 图像分割后得到 24×24=57624 \times 24 = 57624×24=576 个补丁。
* 特征矩阵的形状是 (576,D)(576, D)(576,D)。

1. label\_smoothing的作用

nn.CrossEntropyLoss(label\_smoothing=args.smoothing)label\_smoothing的作用:

当使用 label\_smoothing 时，目标标签会被修改成一个更平滑的分布。如果使用标签平滑，真实标签 [1, 0, 0] 可能会被修改为 [0.9, 0.05, 0.05]。其中，正确类别的概率（0.9）稍微降低，分配一些概率（0.05）给其他类别。

1. weight\_decay的作用:

optimizer = optim.Adam(model.parameters(), lr=args.lr, weight\_decay=args.weight\_decay)

假设损失函数为 L，weight\_decay 为 λ，则在使用 L2 正则化时，损失函数会变为：

Ltotal=Loriginal+λ∑iwi2L\_{\text{total}} = L\_{\text{original}} + \lambda \sum\_{i} w\_i^2Ltotal​=Loriginal​+λi∑​wi2​

1. 学习率调度器

scheduler = torch.optim.lr\_scheduler.CosineAnnealingWarmRestarts(optimizer, 50, T\_mult=2)

**1. 第一个周期（衰减周期）：**

* 在 **前 5 个 epoch** 内，学习率从最大值逐渐按照余弦函数衰减到最小值。
  + 初始学习率为 lr\_max，在第 1 个 epoch 时最大，到了第 5 个 epoch 时，学习率衰减到 eta\_min（最小值）。

**2. 第一次重启：**

* 在第 5 个 epoch 结束后，学习率 **重启**，恢复为最大值 lr\_max。
* 由于 T\_mult=2，**第二个周期**的长度是 **5 \* 2 = 10 个 epoch**。

**3. 第二个周期（衰减周期）：**

* 在 **接下来的 10 个 epoch** 内，学习率再次从最大值 lr\_max 按照余弦函数衰减到最小值 eta\_min。
  + 这次学习率衰减的时间更长，从第 6 个 epoch 到第 15 个 epoch。

**4. 第二次重启：**

* 在第 15 个 epoch 结束后，学习率 **再次重启**，恢复为最大值 lr\_max。
* 由于 T\_mult=2，**第三个周期**的长度是 **10 \* 2 = 20 个 epoch**，但是因为总训练时间只有 20 个 epoch，所以这个周期将覆盖所有剩余的训练步骤。

**5. 第三个周期（衰减周期）：**

* 从第 16 个 epoch 到第 20 个 epoch，学习率会从最大值逐渐衰减到最小值。