##这是个非常关键的问题。在知识蒸馏（distillation）中引入 softmax 的温度系数 T（temperature）时，如果你希望梯度正确传播，训练时需要特别注意以下几点：
##
##🧠 背景回顾：什么是 Temperature T?
##softmax(logits / T) 用于控制输出分布的“平滑程度”：
##T = 1：正常 softmax。
##T > 1：输出分布更平滑，差异更小。
##T < 1：输出分布更尖锐。
##
##在知识蒸馏中，一般设置 T = 2 ~ 10。
##
##✅ 训练时计算蒸馏损失时的注意点
##你通常会有两个 loss：
##
##loss_ce：交叉熵损失（真实标签与 student 输出之间）
##
##loss_kd：蒸馏损失（student 与 teacher soft logits 对比）
##
##其中 蒸馏损失常见形式是 KL 散度：
##
##loss_kd = KLDivLoss(
log_softmax(student_logits / T),
softmax(teacher_logits / T)
##) * (T * T)
##✅ 为什么要乘 (T * T)?
##这一步非常重要，是为了确保梯度的 scale 保持一致。
##
##📌 数学推导核心
##当你对 softmax(logits / T) 做反向传播时，梯度会被缩小 1 / T^2。
##
##为了补偿这个梯度缩小，你需要在 loss 外层乘上 T^2。
##
##否则：
##Temperature 变大会让 loss 看起来更小；
##但实际影响了梯度方向，训练就会失效。
##
##✅ PyTorch 实现示例（可运行）
##
##import torch
##import torch.nn.functional as F
##
##def distillation_loss(student_logits, teacher_logits, labels, T=4.0, alpha=0.5):
Cross Entropy（真实标签）

```
##    loss_ce = F.cross_entropy(student_logits, labels)
##
##    # 蒸馏 loss：KL divergence between softened probs
##    student_soft = F.log_softmax(student_logits / T, dim=1)
##    teacher_soft = F.softmax(teacher_logits / T, dim=1)
##
##    loss_kd = F.kl_div(student_soft, teacher_soft,
reduction='batchmean') * (T * T)
##
##    # 总 loss = 蒸馏 + 真实标签交叉熵
##    return alpha * loss_kd + (1 - alpha) * loss_ce
##
##🧪 用法示例
##
##student_logits = student_model(input_ids)
##with torch.no_grad():
##    teacher_logits = teacher_model(input_ids)
##
##loss = distillation_loss(student_logits, teacher_logits, labels,
T=4.0, alpha=0.7)
##loss.backward()
##
##✅ 总结：训练时必须做的操作
##步骤   操作
##1     对 logits 做 softmax/log_softmax 时除以 T
##2     loss 外部乘以 T^2 进行梯度缩放补偿
##3     用 alpha 控制 soft loss 与 hard loss 的加权
##
###-----------------------------------------------------------------
-----
##distillation, softmax 中的tempreture T, 在inference时，需要做啥特别操作?
##
##结论（简洁版）：
##推理时 不要使用高温度 T，只使用正常的 softmax(logits) 或 argmax(logits) 来
做预测。
##
##🧠 为什么训练时要用 T，而推理时不用?
##阶段   Temperature T                    用途说明
##训练阶段        T > 1                   平滑 teacher/student 的 soft
label 分布，使 student 能学习 teacher 的知识结构（细粒度相似度）
##推理阶段        T = 1                   直接输出分类结果，不需要知识平滑，目
标是做最终决策
##
##✅ 推理代码对比：
##✅ 推理时：正常 softmax
```

```
##
##logits = student_model(input_ids)
##probs = torch.softmax(logits, dim=-1)
##pred = torch.argmax(probs, dim=-1)
##或者甚至更简单：
##
##pred = torch.argmax(logits, dim=-1)
##🚫 不要再做这个（训练用）：
##
### ❌ 这是训练用的高温 softmax，不适合 inference
##probs = torch.softmax(logits / T, dim=-1)
##🔍 举个例子
##训练时用 T = 4：
##
### 训练阶段
##teacher_probs = softmax(teacher_logits / 4.0)
##student_probs = softmax(student_logits / 4.0)
##但 推理阶段，只需要：
##
### 推理阶段
##pred = torch.argmax(student_logits, dim=-1)
##🧾 总结
##阶段            softmax(logits / T)              是否乘以 T²
用途
##训练（蒸馏）          ✅（T > 1）                ✅（为了梯度正确）
获取 soft target + 平滑训练
##推理                    ❌（直接用 logits）       ❌
输出预测类别

##-------------------------------------------------------------------
https://zhuanlan.zhihu.com/p/102038521
4.2. 温度代表了什么，如何选取合适的温度?
温度的高低改变的是Net-S训练过程中对负标签的关注程度：温度较低时，对负标签的关注，尤其是那些显著低于平均值的负标签的关注较少；而温度较高时，负标签相关的值会相对增大，Net-S会相对多地关注到负标签。

实际上，负标签中包含一定的信息，尤其是那些值显著高于平均值的负标签。但由于Net-T的训练过程决定了负标签部分比较noisy，并且负标签的值越低，其信息就越不可靠。因此温度的选取比较empirical，本质上就是在下面两件事之中取舍：

从有部分信息量的负标签中学习 --> 温度要高一些
防止受负标签中噪声的影响 -->温度要低一些
总的来说，T的选择和Net-S的大小有关，Net-S参数量比较小的时候，相对比较低的温度就可
```

以了（因为参数量小的模型不能capture all knowledge，所以可以适当忽略掉一些负标签的信息）

##----------
https://zhuanlan.zhihu.com/p/71986772

1.2 蒸馏时的softmax

比之前的softmax多了一个参数T（temperature），T越大产生的概率分布越平滑。

有两种蒸馏的目标函数：
1. 只使用soft targets：在蒸馏时teacher使用新的softmax产生soft targets；student使用新的softmax在transfer set上学习，和teacher使用相同的T。
2. 同时使用sotf和hard targets：student的目标函数是hard target和soft target目标函数的加权平均，使用hard target时T=1，soft target时T和teacher的一样。Hinton的经验是给hard target的权重小一点。另外要注意的是，因为在求梯度（导数）时新的目标函数会导致梯度是以前的
1/T^2 ，所以要再乘上 T^2 ，不然T变了的话hard target不减小（T=1），但soft target会变。
3. 直接用logits的MSE（是1的special case）