Distilling the Knowledge in a Neural Network 实验

Code：https://github.com/yanggggjie/knowledge_distillation_mnist_pytorch

一、 设计思路：

问题：

在分类问题中，标签使用 one-hot 编码，神经网络过 sofamax 激活后输出，再用 cross entropy 计算 loss，这会丢失类间和类内关于相似性的额外信息。例如：mnist 中一张为 2 的图片，我们的优化目标（标签）one-hot 编码为 [0 0 1 0 0 0 0 0 0 0] 其中类别 2 为 1，其他类别为 0，这就丢失了 2 和 3 具有一定相似度的信息。

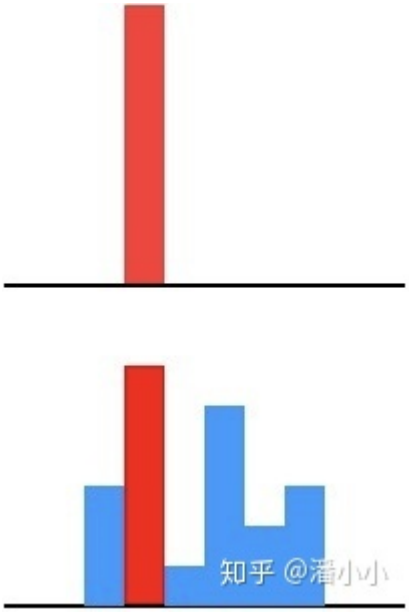
方法：

目的在于改变 one-hot 标签，使得优化目标具有更加丰富的信息可以学习。

因此作者使用温度系数 T 来“软化“softmax：

$$q_i = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}$$

为什么要引入温度 T? 什么是“软化“？下图形象地说明了：上半图为 one-hot label，下半图为 soft label；



引入温度 T，当 T 大于 1 时，可以丰富各类别的信息量。软化 softmax，使得其他类的信息得到保留。

关键：

传统方法使用 one-hot 编码作为标签，这样算出来的 loss 称 hard_loss。

KD 方法使用 teacher 网络的输出过软化的 softmax 作为标签，这样算出来的 loss 称 soft_loss。

本实验的目的就是探究 soft_loss 和 hard_loss 对网络训练的指导作用。

设计

第一组：直接用 hard_loss 训练 student 网络：（hard 组）

```
# NOTE:第一组: 直接用hard_loss训练student网络
loss = F.cross_entropy(student_logits, y)
```

第二组：用 soft_loss 训练 student 网络：（soft 组）

```
# NOTE:第二组: 用soft_loss训练student网络
loss = nn.KLDivLoss()(F.log_softmax(student_logits / self.temperature),
                      F.softmax(teacher_logits / self.temperature)) * (
               self.alpha * self.temperature * self.temperature)
```

第三组：用 hard_loss+soft_loss 训练 student 网络：（hard_soft 组）

```
# NOTE:第三组: 用hard_loss+soft_loss训练student网络
soft_loss = nn.KLDivLoss()(F.log_softmax(student_logits / self.temperature),
                           F.softmax(teacher_logits / self.temperature)) * (
                    self.alpha * self.temperature * self.temperature)
hard_loss = F.cross_entropy(student_logits, y) * (1.0 - self.alpha)
loss = hard_loss + soft_loss
```

二、 网络结构：

Teacher：

```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
TeacherNet                               --                        --
├─Conv2d: 1-1                            [1, 32, 26, 26]           320
├─Conv2d: 1-2                            [1, 64, 24, 24]           18,496
├─Dropout: 1-3                           [1, 64, 12, 12]           --
├─Linear: 1-4                            [1, 128]                  1,179,776
├─Dropout: 1-5                           [1, 128]                  --
├─Linear: 1-6                            [1, 10]                   1,290
==========================================================================================
Total params: 1,199,882
Trainable params: 1,199,882
Non-trainable params: 0
Total mult-adds (M): 12.05
==========================================================================================
Input size (MB): 0.00
Forward/backward pass size (MB): 0.47
Params size (MB): 4.80
Estimated Total Size (MB): 5.27
==========================================================================================
```

预训练：

参数如下：

parms.batch_size = 64

parms.test_batch_size = 1000

parms.epochs = 5

parms.lr = 1.0

parms.gamma = 0.7

parms.scheduler="StepLR(optimizer,step_size=1,gamma=parms.gamma)"

取 5 个 epoch 后的最终模型：

Test Average loss: 0.0299,

Test Accuracy: 9903/10000 (99%)

Student：

```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
StudentNet                               --                        --
├─Linear: 1-1                            [1, 16]                   12,560
├─Linear: 1-2                            [1, 10]                   170
==========================================================================================
Total params: 12,730
Trainable params: 12,730
Non-trainable params: 0
Total mult-adds (M): 0.01
==========================================================================================
Input size (MB): 0.00
Forward/backward pass size (MB): 0.00
Params size (MB): 0.05
Estimated Total Size (MB): 0.05
==========================================================================================
```

三、 进行实验

数据集参数：

dataset_args.seed = 42

dataset_args.train_batch_size = 64

dataset_args.test_batch_size = 1000

dataset_args.train_val_ratio = (0.8, 0.2)

训练参数：

train_args.learning_rate = 1e-3

train_args.max_epochs = 10

train_args.temperature = 2

train_args.alpha = 0.8

EarlyStopping(monitor='student_val_loss',

min_delta=1e-4, patience=2,

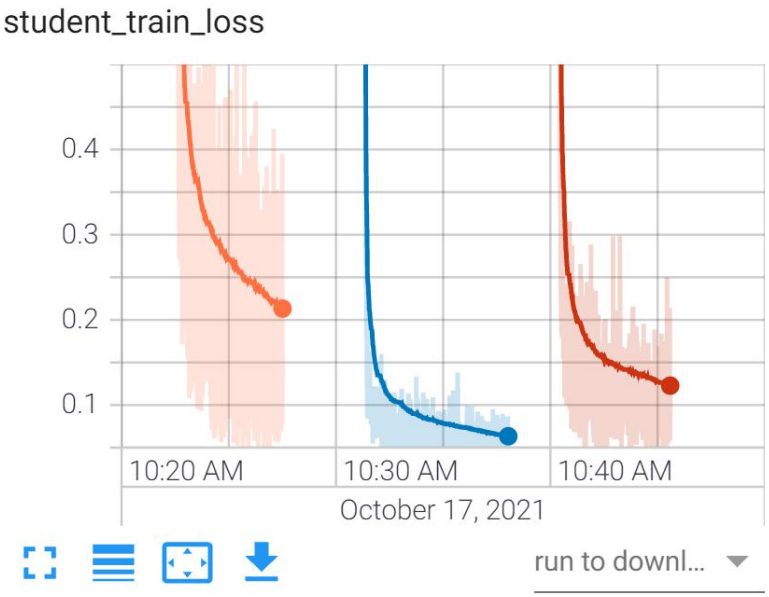
verbose=False, mode='min')

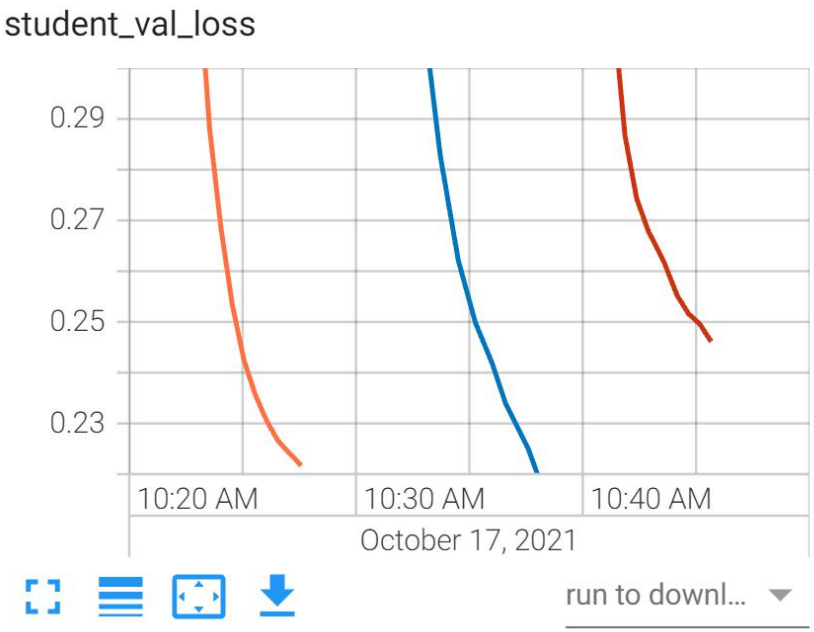
四、 实验结果：

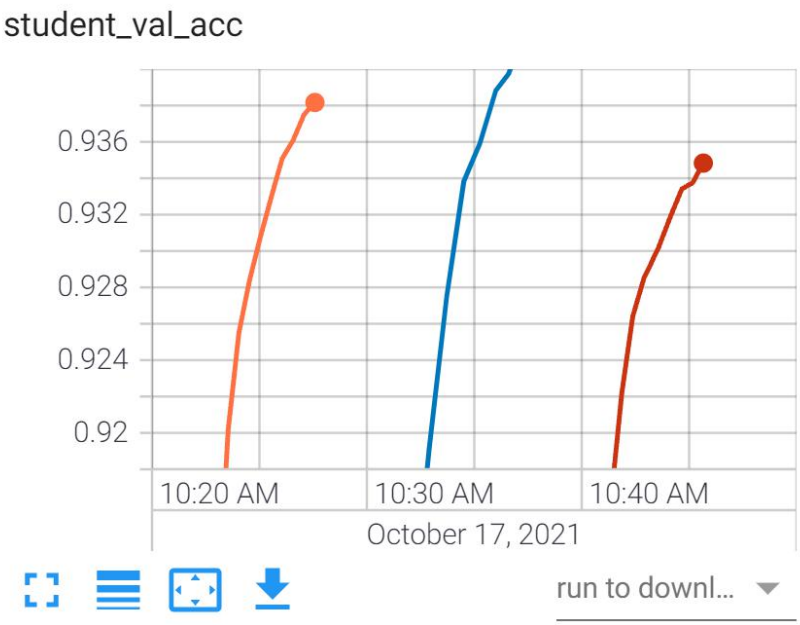
test_acc			
第一组（hard 组）	第二组（soft 组）	第三组（hard+soft 组）	Teacher
0.94559	0.94980	0.93910	0.99029

下图数据：

分别为 “第一组（hard 组） 第二组（soft 组） 第三组（hard+soft 组）”


student_train_loss
0.4
0.3
0.2
0.1
10:20 AM
10:30 AM
10:40 AM
October 17, 2021
run to downl...


student_val_loss
0.29
0.27
0.25
0.23
10:20 AM
10:30 AM
10:40 AM
October 17, 2021
run to downl...



五、 实验结论：

1. 使用 soft_target 能够极大的有助于训练，不但收敛速度加快，而且还趋于平稳，是一种很不错的训练小模型的方式。
    a) 原因在于 soft_target 使得各类的信息得到保留，信息量更大，学的就好。
2. 为什么 soft_target+hard_target 效果看上去最差？
    a) 原因在于这组 loss＝soft_target+ （1-α）*hard_target，这个α系数来平衡 hard 和 soft 比例，这个系数不是学到的，是需要调参的，需要进一步实验。
3. 使用 soft_target 方法，学到了类间相似的知识，应当比 hard_target 具有更高的精确度和更好的泛化性。
    a) 实验中没有明显的表现出这一现象，但有表现的更好的趋势，对比 hard_target，soft_target 的 train loss 有很大优势，并且 val loss 下降的趋势还非常大。10epochs 是不足够的，但受制于算力，没有继续实验。
4. 三组对比来看，当没有进一步调参，那么直接只使用 soft_targets 是最合适，最合算的方法。
5. 基于 softmax 软化的知识蒸馏对于小模型训练有很大好处，用来提升小模型的性能是一种非常好的选择。

六、 参考

1. https://github.com/peterliht/knowledge-distillation-pytorch
2. 【经典简读】知识蒸馏(Knowledge Distillation) 经典之作 - 知乎 (zhihu.com)