In [1]:

```python
import torch
import torch.nn.functional
from torch import nn

from labml import experiment, tracker
from labml.configs import option
from labml_helpers.train_valid import BatchIndex
from labml_nn.distillation.large import LargeModel
from labml_nn.distillation.small import SmallModel
from labml_nn.experiments.cifar10 import CIFAR10Configs
```

In [2]:

```python
class Configs(CIFAR10Configs):
    model: SmallModel
    large: LargeModel
    kl_div_loss = nn.KLDivLoss(log_target=True)
    loss_func = nn.CrossEntropyLoss()
    temperature: float = 5.
    soft_targets_weight: float = 100.
    label_loss_weight: float = 0.5
```

In [3]:

```
def step(self, batch: any, batch_idx: BatchIndex):
    self.model.train(self.mode.is_train)
    self.large.eval()
    data, target = batch[0].to(self.device), batch[1].to(self.device)
    if self.mode.is_train:
        tracker.add_global_step(len(data))
    with torch.no_grad():
        large_logits = self.large(data)
    output = self.model(data)
    soft_targets = nn.functional.log_softmax(large_logits / self.temperature, dim=-1)
    soft_prob = nn.functional.log_softmax(output / self.temperature, dim=-1)
    soft_targets_loss = self.kl_div_loss(soft_prob, soft_targets)
    label_loss = self.loss_func(output, target)
    loss = self.soft_targets_weight * soft_targets_loss + self.label_loss_weight * label_loss
    tracker.add({"loss.kl_div.": soft_targets_loss,
                 "loss.nll": label_loss,
                 "loss.": loss})
    self.accuracy(output, target)
    self.accuracy.track()
    if self.mode.is_train:
        loss.backward()
        self.optimizer.step()
        if batch_idx.is_last:
            tracker.add('model', self.model)
        self.optimizer.zero_grad()
    tracker.save()
```

In [4]:

```
@option(Configs.large)
def _large_model(c: Configs):
    return LargeModel().to(c.device)
@option(Configs.model)
def _small_student_model(c: Configs):
    return SmallModel().to(c.device)
```

In [5]:

```
def get_saved_model(run_uuid: str, checkpoint: int):
    from labml_nn.distillation.large import Configs as LargeConfigs
    experiment.evaluate()
    conf = LargeConfigs()
    experiment.configs(conf, experiment.load_configs(run_uuid))
    experiment.add_pytorch_models({'model': conf.model})
    experiment.load(run_uuid, checkpoint)
    experiment.start()
    return conf.model
```

In [6]:

```
def main(run_uuid: str, checkpoint: int):
    large_model = get_saved_model(run_uuid, checkpoint)
    experiment.create(name='distillation', comment='cifar10')
    conf = Configs()
    conf.large = large_model
    experiment.configs(conf, {
        'optimizer.optimizer': 'Adam',
        'optimizer.learning_rate': 2.5e-4,
        'model': '_small_student_model',
    })
    experiment.add_pytorch_models({'model': conf.model})
    experiment.load(None, None)
    with experiment.start():
        conf.run()
```

```
In [7]: main('d46cd53edaec11eb93c38d6538aee7d6', 1_000_000)
```

```
--------------------------------------------------
LABML WARNING
.labml.yaml config file could not be found. Looking in path: /Users/jianyiyang/Desktop/work/deepLearni
ng/summerClass/final/distillation example
--------------------------------------------------

--------------------------------------------------
LABML WARNING
Not a valid git repository: /Users/jianyiyang/Desktop/work/deepLearning/summerClass/final/distillation
example
--------------------------------------------------

--------------------------------------------------
LABML ERROR
Couldn't find a previous run to load configurations: d46cd53edaec11eb93c38d6538aee7d6
--------------------------------------------------

Prepare model...
  Prepare device.device...
    Prepare device.device_info...[DONE] 2.69ms
  Prepare device.device...[DONE]        6.46ms
Prepare model...[DONE]  110.67ms
Couldn't find a previous run

Notebook Experiment: 83b07f462d4311ee9310acde48001122
[clean]: ""

--------------------------------------------------
LABML WARNING
Not a valid git repository: /Users/jianyiyang/Desktop/work/deepLearning/summerClass/final/distillation
example
--------------------------------------------------

Prepare model...[DONE]  10.28ms
No labml server url specified. Please start a labml server and specify the URL. Docs: https://github.c
```

```
om/labmlai/labml/tree/master/app

distillation: 83f598e22d4311ee9310acde48001122
        cifar10
[clean]: ""
~/labml/configs.yaml does not exist. Creating /Users/jianyiyang/.labml/configs.yaml
Initialize...
  Prepare mode...[DONE] 3.25ms
Initialize...[DONE]     77.61ms
Prepare validator...
  Prepare valid_loader...
    Prepare valid_dataset...
      Prepare dataset_transforms...[DONE]       3.33ms
    Prepare valid_dataset...[DONE]      63,081.73ms
  Prepare valid_loader...[DONE] 63,177.20ms
Prepare validator...[DONE]      63,277.03ms
Prepare trainer...
  Prepare train_loader...
    Prepare train_dataset...[DONE]      725.73ms
  Prepare train_loader...[DONE] 819.43ms
Prepare trainer...[DONE]        846.29ms
Prepare training_loop...
  Prepare loop_count...[DONE]   46.19ms
Prepare training_loop...[DONE]  229.86ms
       0:  Train:  ...  0ms    0ms  0:00m/  0:00m
Prepare optimizer...[DONE]      3.94ms
Prepare optimizer.optimizer...
  Prepare optimizer.weight_decay_obj...[DONE]   4.14ms
Prepare optimizer.optimizer...[DONE]    10.60ms
  50,000:  Train: 100% 245,426ms  Valid: 100% 11,634ms   loss.train: 1.051353 accuracy.train: 0.578660
loss.valid:  0.91503 accuracy.valid: 0.608100  321,307ms  0:05m/  0:48m
 100,000:  Train: 100% 233,280ms  Valid: 100% 11,383ms   loss.train: 0.858824 accuracy.train: 0.730740
loss.valid: 0.738977 accuracy.valid: 0.725500  284,795ms  0:09m/  0:37m
 150,000:  Train: 100%   241,689ms  Valid: 100% 11,771ms   loss.train: 0.730364 accuracy.train: 0.7914
20 loss.valid: 0.676379 accuracy.valid: 0.765300  275,545ms  0:14m/  0:31m
 200,000:  Train: 100%   244,112ms  Valid: 100% 11,960ms   loss.train: 0.626952 accuracy.train: 0.8314
00 loss.valid: 0.656888 accuracy.valid: 0.774500  279,666ms  0:18m/  0:27m
```

```
 250,000:  Train: 100%   392,136ms  Valid: 100% 15,118ms   loss.train: 0.272719 accuracy.train: 0.8658
40 loss.valid: 0.684196 accuracy.valid: 0.785900  304,412ms  0:24m/  0:26m
 300,000:  Train: 100%   338,520ms  Valid: 100% 13,535ms   loss.train: 0.121643 accuracy.train: 0.8973
00 loss.valid: 0.671173 accuracy.valid: 0.791400  330,315ms  0:30m/  0:24m
 350,000:  Train: 100%   312,880ms  Valid: 100% 13,134ms   loss.train: 0.031197 accuracy.train: 0.9220
80 loss.valid: 0.658405 accuracy.valid: 0.793400  324,037ms  0:35m/  0:18m
 400,000:  Train: 100%   344,714ms  Valid: 100% 13,458ms   loss.train: 0.328620 accuracy.train: 0.9393
20 loss.valid: 0.737392 accuracy.valid: 0.792600  338,564ms  0:41m/  0:14m
 450,000:  Train: 100%   270,469ms  Valid: 100% 11,885ms   loss.train: 0.240720 accuracy.train: 0.9529
20 loss.valid: 0.776116 accuracy.valid: 0.794700  335,637ms  0:47m/  0:08m
 500,000:   loss.train: 0.073790 accuracy.train: 0.963500 loss.valid: 0.805189 accuracy.valid: 0.79610
0

2023-07-28 20:37:34.349372: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binar
y is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appro
priate compiler flags.

Downloading https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz (https://www.cs.toronto.edu/~kriz/
cifar-10-python.tar.gz) to /Users/jianyiyang/Desktop/work/deepLearning/summerClass/final/distillation
example/data/cifar-10-python.tar.gz

100%|████████████████████████████████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████████████| 170498071/170
498071 [00:55<00:00, 3099570.55it/s]

Extracting /Users/jianyiyang/Desktop/work/deepLearning/summerClass/final/distillation example/data/cif
ar-10-python.tar.gz to /Users/jianyiyang/Desktop/work/deepLearning/summerClass/final/distillation exam
ple/data
Files already downloaded and verified
```

In [8]:

```
print(LargeModel())
```

```
LargeModel(
  (layers): Sequential(
    (0): Dropout(p=0.1, inplace=False)
    (1): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (2): BatchNorm()
    (3): ReLU(inplace=True)
    (4): Dropout(p=0.1, inplace=False)
    (5): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (6): BatchNorm()
    (7): ReLU(inplace=True)
    (8): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (9): Dropout(p=0.1, inplace=False)
    (10): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): BatchNorm()
    (12): ReLU(inplace=True)
    (13): Dropout(p=0.1, inplace=False)
    (14): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (15): BatchNorm()
    (16): ReLU(inplace=True)
    (17): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (18): Dropout(p=0.1, inplace=False)
    (19): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (20): BatchNorm()
    (21): ReLU(inplace=True)
    (22): Dropout(p=0.1, inplace=False)
    (23): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (24): BatchNorm()
    (25): ReLU(inplace=True)
    (26): Dropout(p=0.1, inplace=False)
    (27): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (28): BatchNorm()
    (29): ReLU(inplace=True)
    (30): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (31): Dropout(p=0.1, inplace=False)
    (32): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (33): BatchNorm()
    (34): ReLU(inplace=True)
    (35): Dropout(p=0.1, inplace=False)
    (36): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (37): BatchNorm()
    (38): ReLU(inplace=True)
    (39): Dropout(p=0.1, inplace=False)
    (40): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (41): BatchNorm()
    (42): ReLU(inplace=True)
    (43): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (44): Dropout(p=0.1, inplace=False)
    (45): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (46): BatchNorm()
    (47): ReLU(inplace=True)
    (48): Dropout(p=0.1, inplace=False)
    (49): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (50): BatchNorm()
    (51): ReLU(inplace=True)
    (52): Dropout(p=0.1, inplace=False)
    (53): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (54): BatchNorm()
    (55): ReLU(inplace=True)
    (56): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (fc): Linear(in_features=512, out_features=10, bias=True)
)
```

In [ ]: