About Neural Network Distiller:

Distiller is an open-source Python package which is used for neural network compression research. Network compression can reduce the memory footprint of a neural network, increase its inference speed and save energy. Distiller provides a PyTorch environment for prototyping and analyzing compression algorithms, such as sparsity-inducing methods and low-precision arithmetic.
For now, the distiller does not support the simultaneous training of teacher and student- so we have to first train the cumbersome teacher model and then we use knowledge distillation on the student models.

Teacher Models used:

1) Resnet 18, trained 260 epochs-
    1. Memory: 42.6 MB
    2. Parameters: 11,173,962
    3. Latency for 10000 test images: 0.00063 seconds (GPU)
       0.031 seconds (CPU)
    4. Training Set accuracy: 99.976 %
    5. Accuracy of the network on the 10000 test images: 94.59 %
        - Accuracy of plane : 97 %
        - Accuracy of car : 97 %
        - Accuracy of bird : 93 %
        - Accuracy of cat : 85 %
        - Accuracy of deer : 96 %
        - Accuracy of dog : 92 %
        - Accuracy of frog : 98 %
        - Accuracy of horse : 96 %
        - Accuracy of ship : 96 %
        - Accuracy of truck : 96 %
2) VGG16, trained 275 epochs-
    1. Memory: 56.2 MB
    2. Parameters: 14,728,266
    3. Average Latency: 0.00052 seconds (GPU)
       0.025 seconds (CPU)
    4. Training Set Accuracy: 99.834 %
    5. Accuracy of the network on the 10000 test images: 93.10 %
        - Accuracy of plane : 93 %
        - Accuracy of car : 96 %
        - Accuracy of bird : 91 %
        - Accuracy of cat : 83 %
        - Accuracy of deer : 92 %
        - Accuracy of dog : 88 %

- Accuracy of frog : 95 %
- Accuracy of horse : 94 %
- Accuracy of ship : 96 %
- Accuracy of truck : 95 %

3) Averaged Teacher [AvNet]: 0.72*Resnet18 + 0.28*VGG16-
   1. Memory: 98.9 MB
   2. Parameters: 25,902,228
   3. Average Latency: 0.00085 seconds (GPU)
      0.0490 seconds (CPU)
   4. Training set accuracy: 100 %
   5. Accuracy of the network on the 10000 test images: 95.44 %
      - Accuracy of plane : 97 %
      - Accuracy of   car : 97 %
      - Accuracy of bird : 93 %
      - Accuracy of   cat : 88 %
      - Accuracy of deer : 95 %
      - Accuracy of   dog : 93 %
      - Accuracy of frog : 97 %
      - Accuracy of horse : 96 %
      - Accuracy of ship : 96 %
      - Accuracy of truck : 96 %

4) JointNet, pre-trained models + 250 epochs-
   1. Memory: 98.9 MB
   2. Parameters: 25,902,558
   3. Average Latency: 0.000885 seconds (GPU)
      0.0492 seconds (CPU)
   4. Training set accuracy: 100 %
   5. Accuracy of the network on the 10000 test images: 95.99 %
      - Accuracy of plane : 97 %
      - Accuracy of   car : 98 %
      - Accuracy of bird : 94 %
      - Accuracy of   cat : 91 %
      - Accuracy of deer : 97 %
      - Accuracy of   dog : 93 %
      - Accuracy of frog : 98 %
      - Accuracy of horse : 97 %
      - Accuracy of ship : 98 %
      - Accuracy of truck : 98 %

*__**About AvNet-**__ In AvNet, we take two neural networks- VGG16 and Resnet18 and do a simple weighted average of the two outputs that we get from them. The weights (0.28 and 0.72) were found experimentally. Here, we used pre-trained Resnet18 and VGG16 models. So, no training is done here.

__****About JointNet-**__ In JointNet, we took two pre-trained neural networks- VGG16 and Resnet18, and did the following computation:

a1 = vgg16(x)
a2 = resnet18(x)
a3 = 0.28*a1 + 0.72*a2 (*As in AvNet*)
a4 = softmax(a3)
y = linear10(x) – *a layer with 10 units*

Now, we have to train this new layer to get better accuracy. However, we also train some of the final layers of the two networks. So, some of the last layers are 'unfreezed' and they are also trained. The loss function that we used was:

l1 = cross_entropy(a1, labels)
l2 = cross_entropy(a2, labels)
l3 = cross_entropy(a3, labels)
loss = l1 + l2 + l3

In this experiment, we unfroze last 4 layers of VGG16- i.e., 1 FC layer and 3 Convolutional layers. And for Resnet18, we unfroze the last FC layer and the last **basic block**.
Also, after training, the accuracy of both the networks increased:
Resnet18: 94.59 % -> 95.31 %
VGG16: 93.10 % -> 93.63 %

Student Models used:

We used two kinds of students- one is the 5-CNN as given in [1], and the other is a VGG variant we made and call VGG7.

VGG7 architecture:-
[64, 'M', 64, 'M', 128, 'M', 256, 'M', 512, 'M', 512 x 10 FC layer]
'M' => Max. pool layer, with stride = 2
Other layers are convolutional layers, with 3x3 kernels, stride = 1 and padding = 1

1) VGG7, trained 350 epochs:
    1. Memory: 6.07 MB
    2. Parameters: 1,595,082
    3. Average Latency: 0.000342 seconds (GPU)

0.0042 seconds (CPU)

4. Training set accuracy: 99.888 %
5. Accuracy of the network on the 10000 test images: 90.6400 %
   - Accuracy of plane : 92 %
   - Accuracy of car : 95 %
   - Accuracy of bird : 87 %
   - Accuracy of cat : 81 %
   - Accuracy of deer : 92 %
   - Accuracy of dog : 83 %
   - Accuracy of frog : 93 %
   - Accuracy of horse : 92 %
   - Accuracy of ship : 94 %
   - Accuracy of truck : 93 %

2) 5-CNN, trained 200 epochs:
   1) Memory: 1.37 MB
   2) Parameters: 357,514
   3) Average Latency: 0.000313 seconds (GPU)

      0.0024 seconds (CPU)

   4) Accuracy in [1] - 84.74 %
   5) Accuracy of the network on the 10000 test images: 82.85%
      - Accuracy of plane : 86 %
      - Accuracy of car : 90 %
      - Accuracy of bird : 74 %
      - Accuracy of cat : 68 %
      - Accuracy of deer : 85 %
      - Accuracy of dog : 71 %
      - Accuracy of frog : 88 %
      - Accuracy of horse : 84 %
      - Accuracy of ship : 91 %
      - Accuracy of truck : 87 %

Distillation Results:

1. Resnet18 on VGG7

alpha	Temperature	Accuracy
0.5	1	90.82%
0.5	1.5	90.94%
0.6	1.5	90.88%
0.4	1.5	90.84%
0.5	2	90.69%

Resnet18 accuracy: 94.59 %
VGG7 baseline accuracy: 90.64 %

2. VGG16 on VGG7

alpha	Temperature	Accuracy
0.5	1	91.03%
0.5	1.5	90.78%

VGG16 accuracy: 93.10 %
VGG7 baseline accuracy: 90.64 %

3. AvNet on VGG7

alpha	Temperature	Accuracy
0.5	1	90.83%
0.5	1.35	91.10%

AvNet accuracy: 95.44 %
VGG7 baseline accuracy: 90.64 %

4. AvNet on 5-CNN

alpha	Temperature	Accuracy
0.5	1.36	85.17%

AvNet accuracy: 95.44 %
5-CNN baseline accuracy: 82.85 % [84.74 %]

5. JointNet on 5-CNN

alpha	Temperature	Accuracy
0.5	1	85.28%
0.5	1.36	85.60%

JointNet accuracy: 95.99 %
VGG7 baseline accuracy: 82.85 % [84.74 %]

How to use the code- command lines:

To use the code, run the file main_KD_CIFAR10.py- along with the command lines according to the requirement:

For testing-

> python main_KD_CIFAR10.py Dataset_Location **tst** Network_Type Network_Location

Example: **> python main_KD_CIFAR10.py D: tst VGG16 VGG16.pt**

The following Network Types are available-
Resnet18, Resnet34, Resnet50, Resnet101, Resnet152, VGG6AS, VGG6AM, VGG6A, VGG6, VGG7, VGG8, VGG11, VGG13, VGG16, VGG16A, VGG19, 5-CNN, avg, jn.
avg is for AvNet, and jn is for JointNet.

For training without knowledge distillation-

> **python main_KD_CIFAR10.py** Dataset_Location **trn** aug. scheme total_epochs begin_epoch minibatch_size **n** Network_Type Network_Location

- aug- Augmetation on (**y**) or off (**n**)
- scheme- Training scheme used- **sgd** or **adam**
- total_epochs- Total no. of epochs
- begin_epoch- Starting epoch; epochs always begin with 0
- minibatch_size: Minibatch size

Example: **> python main_KD_CIFAR10.py D: trn n sgd 350 0 128 n VGG16 VGG16.pt**

For training with knowledge distillation-

> **python main_KD_CIFAR10.py** Dataset_Location **trn** aug. scheme total_epochs begin_epoch minibatch_size **y** alpha temperature Student_Type Student_Location Teahcer_Type Teacher_Location

Example: **> python main_KD_CIFAR10.py D: trn n sgd 350 0 128 y 0.5 1.5 VGG7 VGG7.pt VGG16 VGG16.pt**

***** For JointNet and AvNet, to specify model locations (and model types) the code in avNet.py and netJoin.py has to be modified.

***** As AvNet is not trained, to create AvNet, use the following code:

> python main_KD_CIFAR10.py D: trn n sgd 350 350 128 n avg avNet.pt

That is, just put begin_epoch >= total_epochs

References

[1]. Li, H.: Exploring Knowledge Distillation of Deep Neural Networks for Efficient Hardware solutions (2018)
http://cs230.stanford.edu/files_winter_2018/projects/6940224.pdf

[2]. Mishra, A., Marr, D.: Apprentice:Using knowledge distillation techniques to improve low-precision network accuracy. ICLR,2018.

[3]. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015)

Code References

Nervana Systems Distiller- https://github.com/NervanaSystems/distiller

Resnet and VGG models- https://github.com/kuangliu/pytorch-cifar