ML2022-2023 Spring HW13 Report

Report Questions

Question 1

1-1

Please copy&paste your student model architecture code to the HW13 GradeScope.

```
def dwpw_conv(in_channels, out_channels, kernel_size, stride=1, padding=1,bias=False):
    return nn.Sequential(
        nn.Conv2d(in_channels, in_channels, kernel_size, stride=stride, padding=padding,bias=bias, groups=in_channels), #depthwise convolution
        nn.BatchNorm2d(in_channels),
        nn.ReLU(inplace=True),
        nn.Conv2d(in_channels, out_channels, 1,  bias= bias,), # pointwise convolution
        nn.BatchNorm2d(out_channels),
        nn.ReLU(inplace=True)
    )

class StudentNet(nn.Module):
    def __init__(self, inplanes = 64):
        super().__init__()
        self.inplanes = inplanes
        self.conv1 = nn.Conv2d(3, 64, kernel_size=7, stride=2, padding=3, bias=False)
        self.bn1 = nn.BatchNorm2d(self.inplanes)
        self.relu = nn.ReLU(inplace=True)
        self.maxpool = nn.MaxPool2d(kernel_size=3, stride=2, padding=1)

        self.layer1 = dwpw_conv(inplanes, inplanes, kernel_size=3)
        self.layer2 = dwpw_conv(inplanes, 128, kernel_size=3, stride=2)
        self.layer3 = dwpw_conv(128, 256, kernel_size=3, stride=2)
        self.layer4 = dwpw_conv(256, 141, kernel_size=3, stride=2)

        self.avgpool = nn.AdaptiveAvgPool2d((1, 1))
        self.fc = nn.Linear(141, 11)

    def forward(self, x):
        x=self.conv1(x)
        x=self.bn1(x)
        x=self.relu(x)
        x=self.maxpool(x)

        x=self.layer1(x)
        x=self.layer2(x)
        x=self.layer3(x)
        x=self.layer4(x)

        x=self.avgpool(x)
        x = torch.flatten(x, 1)
        x=self.fc(x)

        return x
```

1-2

Copy&Paste the torchsummary result of your student model to HW13 GradeScope. The total params should not exceed 60,000

Answer:

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 64, 112, 112]           9,408
       BatchNorm2d-2         [-1, 64, 112, 112]             128
              ReLU-3         [-1, 64, 112, 112]               0
         MaxPool2d-4           [-1, 64, 56, 56]               0
            Conv2d-5           [-1, 64, 56, 56]             576
       BatchNorm2d-6           [-1, 64, 56, 56]             128
              ReLU-7           [-1, 64, 56, 56]               0
            Conv2d-8           [-1, 64, 56, 56]           4,096
       BatchNorm2d-9           [-1, 64, 56, 56]             128
             ReLU-10           [-1, 64, 56, 56]               0
           Conv2d-11           [-1, 64, 28, 28]             576
      BatchNorm2d-12           [-1, 64, 28, 28]             128
             ReLU-13           [-1, 64, 28, 28]               0
           Conv2d-14          [-1, 128, 28, 28]           8,192
      BatchNorm2d-15          [-1, 128, 28, 28]             256
             ReLU-16          [-1, 128, 28, 28]               0
           Conv2d-17          [-1, 128, 14, 14]           1,152
      BatchNorm2d-18          [-1, 128, 14, 14]             256
             ReLU-19          [-1, 128, 14, 14]               0
           Conv2d-20          [-1, 256, 14, 14]          32,768
      BatchNorm2d-21          [-1, 256, 14, 14]             512
             ReLU-22          [-1, 256, 14, 14]               0
```

Number of total parameters: 0.38M ≈ 38000

Question 2

2-1

Please copy and paste your KL divergence loss function (loss_fn_kd , choose alpha=0.5, temperature=1.0) to HW13 GradeScope

```python
# Implement the loss function with KL divergence loss for knowledge distillation.
# You also have to copy-paste this whole block to HW13 GradeScope.
def loss_fn_kd(student_logits, labels, teacher_logits, alpha=0.5, temperature=1.0):
  kl_loss = nn.KLDivLoss(reduction="batchmean", log_target=True)
  CE_loss = nn.CrossEntropyLoss()
  p = F.log_softmax(student_logits / temperature, dim=1)
  q = F.log_softmax(teacher_logits / temperature, dim=1)
  loss=alpha * (temperature**2) * kl_loss(p, q) + (1-alpha) * CE_loss(student_logits,labels)

  return loss
```

2-2

Which is true about the hyperparameter T (temperature) of KL divergence loss function for knowledge distillation ?

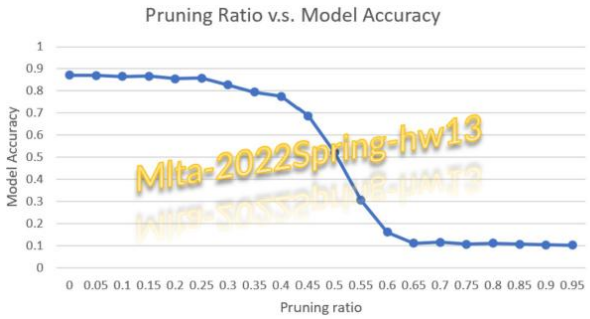
Answer:
(a). Using a higher value for T produces a softer probability distribution over classes.

Reason: In Knowledge Distillation, the role of the temperature parameter T is to smooth the output of the model. Specifically, the temperature T is introduced into the softmax function and is used to generate soft targets.

Question 3

3-1

Please reference to this pytorch network pruning tutorial and the provided sample code to adopt network pruning on your teacher network (you can also use the provided one). Plot the graph like below to indicate the relationship between pruning ratio and accuracy (use validation set). Please checkout HW13 GradeScope for more details about this question .



3-2

Is there a difference in inference speed between two versions of the same model, one using structured pruning and the other using unstructured pruning? Why?

Answer:
Yes. Structured pruning can more effectively improve the inference speed of the model because the pruned model it generates is more computationally efficient on the hardware, while unstructured pruning reduces the number of parameters but suffers from the computational complexity of sparse matrices. and hardware utilization, the inference speed is not significantly improved. Therefore, models using structured pruning are generally faster in inference speed than models using unstructured pruning.