Q1 Architecture Design
1 分數

Q1.1 Student Model Architecture
0.5 分數

Please copy&paste your student model architecture code block below. Note that the function get_student_model() must be included.

For example:

```
class StudentNet(nn.Module):
    def __init__(self):
      super().__init__()
      self.cnn = nn.Sequential(
        nn.Conv2d(3, 32, 3),
        nn.BatchNorm2d(32),
        nn.ReLU(),
        nn.Conv2d(32, 32, 3),
        nn.BatchNorm2d(32),
        nn.ReLU(),
        nn.MaxPool2d(2, 2, 0),

        nn.Conv2d(32, 64, 3),
        nn.BatchNorm2d(64),
        nn.ReLU(),
        nn.MaxPool2d(2, 2, 0),

        nn.Conv2d(64, 100, 3),
        nn.BatchNorm2d(100),
        nn.ReLU(),
        nn.MaxPool2d(2, 2, 0),
        # Here we adopt Global Average Pooling for various input size.
        nn.AdaptiveAvgPool2d((1, 1)),
      )
      self.fc = nn.Sequential(
        nn.Linear(100, 11),
      )

    def forward(self, x):
      out = self.cnn(x)
      out = out.view(out.size()[0], -1)
      return self.fc(out)

def get_student_model():
    return StudentNet()
```

Your student model architecture **code**(text only):

```
class BasicBlock(nn.Module):
    expansion = 1
    def __init__(self, in_planes, planes, stride=1):
        super(BasicBlock, self).__init__()
        self.conv1 = deepwise_pointwise_conv(in_planes, planes, kernel_size=3,
stride=stride, padding=1)
        self.bn1 = nn.BatchNorm2d(planes)
        self.conv2 = deepwise_pointwise_conv(planes, planes, kernel_size=3,
stride=1, padding=1)
        self.bn2 = nn.BatchNorm2d(planes)

        self.shortcut = nn.Sequential()
        if stride != 1 or in_planes != self.expansion*planes:
            self.shortcut = nn.Sequential(
                deepwise_pointwise_conv(in_planes, self.expansion*planes,
kernel_size=1, stride=stride),
                nn.BatchNorm2d(self.expansion*planes)
            )

    def forward(self, x):
        out = F.relu(self.bn1(self.conv1(x)))
        out = self.bn2(self.conv2(out))
        out += self.shortcut(x)
        out = F.relu(out)
        return out

class Bottleneck(nn.Module):
    expansion = 4
    def __init__(self, in_planes, planes, stride=1):
        super(Bottleneck, self).__init__()
        self.conv1 = deepwise_pointwise_conv(in_planes, planes, kernel_size=1)
        self.bn1 = nn.BatchNorm2d(planes)
        self.conv2 = deepwise_pointwise_conv(planes, planes, kernel_size=3,
stride=stride, padding=1)
        self.bn2 = nn.BatchNorm2d(planes)
        self.conv3 = deepwise_pointwise_conv(planes, self.expansion*planes,
kernel_size=1)
        self.bn3 = nn.BatchNorm2d(self.expansion*planes)

        self.shortcut = nn.Sequential()
        if stride != 1 or in_planes != self.expansion*planes:
            self.shortcut = nn.Sequential(
                deepwise_pointwise_conv(in_planes, self.expansion*planes,
kernel_size=1, stride=stride),
                nn.BatchNorm2d(self.expansion*planes)
            )
```

```
    def forward(self, x):
        out = F.relu(self.bn1(self.conv1(x)))
        out = F.relu(self.bn2(self.conv2(out)))
        out = self.bn3(self.conv3(out))
        out += self.shortcut(x)
        out = F.relu(out)
        return out

class StudentNet(nn.Module):
    def __init__(self, block, num_blocks, num_classes=11):
        super().__init__()

        self.in_planes = 64
        self.conv1 = nn.Conv2d(3, 64, kernel_size=3, stride=1, padding=1,
bias=False)
        self.bn1 = nn.BatchNorm2d(64)
        self.layer1 = self._make_layer(block, 64, num_blocks[0], stride=1)
        self.layer2 = self._make_layer(block, 128, num_blocks[1], stride=2)
        self.linear = nn.Linear(128*block.expansion, num_classes)

    def _make_layer(self, block, planes, num_blocks, stride):
        strides = [stride] + [1]*(num_blocks-1)
        layers = []
        for stride in strides:
            layers.append(block(self.in_planes, planes, stride))
            self.in_planes = planes * block.expansion
        return nn.Sequential(*layers)

    def forward(self, x):
        out = F.relu(self.bn1(self.conv1(x)))
        out = self.layer1(out)
        out = self.layer2(out)
        out = F.adaptive_avg_pool2d(F.avg_pool2d(out, 4), 1)
        out = out.view(out.size(0), -1)
        out = self.linear(out)
        return out

def get_student_model():
    return StudentNet(BasicBlock, [2, 2])
```

Q1.2 Torchsummary of Student Model
0.5 分數

Copy&Paste the torchsummary result of your student model. The total params should not exceed *100,000*.

For example:

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 32, 222, 222]             896
       BatchNorm2d-2         [-1, 32, 222, 222]              64
              ReLU-3         [-1, 32, 222, 222]               0
            Conv2d-4         [-1, 32, 220, 220]           9,248
       BatchNorm2d-5         [-1, 32, 220, 220]              64
              ReLU-6         [-1, 32, 220, 220]               0
         MaxPool2d-7         [-1, 32, 110, 110]               0
            Conv2d-8         [-1, 64, 108, 108]          18,496
       BatchNorm2d-9         [-1, 64, 108, 108]             128
             ReLU-10         [-1, 64, 108, 108]               0
        MaxPool2d-11           [-1, 64, 54, 54]               0
           Conv2d-12          [-1, 100, 52, 52]          57,700
      BatchNorm2d-13          [-1, 100, 52, 52]             200
             ReLU-14          [-1, 100, 52, 52]               0
        MaxPool2d-15          [-1, 100, 26, 26]               0
AdaptiveAvgPool2d-16            [-1, 100, 1, 1]               0
           Linear-17                   [-1, 11]           1,111
================================================================
Total params: 87,907
Trainable params: 87,907
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.57
Forward/backward pass size (MB): 99.72
Params size (MB): 0.34
Estimated Total Size (MB): 100.62
----------------------------------------------------------------
```

Torchsummary of your student network(text only).

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
======
            Conv2d-1         [-1, 64, 224, 224]           1,728
       BatchNorm2d-2         [-1, 64, 224, 224]             128
            Conv2d-3         [-1, 64, 224, 224]             640
       BatchNorm2d-4         [-1, 64, 224, 224]             128
              ReLU-5         [-1, 64, 224, 224]               0
            Conv2d-6         [-1, 64, 224, 224]           4,160
       BatchNorm2d-7         [-1, 64, 224, 224]             128
            Conv2d-8         [-1, 64, 224, 224]             640
       BatchNorm2d-9         [-1, 64, 224, 224]             128
             ReLU-10         [-1, 64, 224, 224]               0
           Conv2d-11         [-1, 64, 224, 224]           4,160
```

BatchNorm2d-12	[-1, 64, 224, 224]	128
BasicBlock-13	[-1, 64, 224, 224]	0
Conv2d-14	[-1, 64, 224, 224]	640
BatchNorm2d-15	[-1, 64, 224, 224]	128
ReLU-16	[-1, 64, 224, 224]	0
Conv2d-17	[-1, 64, 224, 224]	4,160
BatchNorm2d-18	[-1, 64, 224, 224]	128
Conv2d-19	[-1, 64, 224, 224]	640
BatchNorm2d-20	[-1, 64, 224, 224]	128
ReLU-21	[-1, 64, 224, 224]	0
Conv2d-22	[-1, 64, 224, 224]	4,160
BatchNorm2d-23	[-1, 64, 224, 224]	128
BasicBlock-24	[-1, 64, 224, 224]	0
Conv2d-25	[-1, 64, 112, 112]	640
BatchNorm2d-26	[-1, 64, 112, 112]	128
ReLU-27	[-1, 64, 112, 112]	0
Conv2d-28	[-1, 128, 112, 112]	8,320
BatchNorm2d-29	[-1, 128, 112, 112]	256
Conv2d-30	[-1, 128, 112, 112]	1,280
BatchNorm2d-31	[-1, 128, 112, 112]	256
ReLU-32	[-1, 128, 112, 112]	0
Conv2d-33	[-1, 128, 112, 112]	16,512
BatchNorm2d-34	[-1, 128, 112, 112]	256
Conv2d-35	[-1, 64, 112, 112]	128
BatchNorm2d-36	[-1, 64, 112, 112]	128
ReLU-37	[-1, 64, 112, 112]	0
Conv2d-38	[-1, 128, 112, 112]	8,320
BatchNorm2d-39	[-1, 128, 112, 112]	256
BasicBlock-40	[-1, 128, 112, 112]	0
Conv2d-41	[-1, 128, 112, 112]	1,280
BatchNorm2d-42	[-1, 128, 112, 112]	256
ReLU-43	[-1, 128, 112, 112]	0
Conv2d-44	[-1, 128, 112, 112]	16,512
BatchNorm2d-45	[-1, 128, 112, 112]	256
Conv2d-46	[-1, 128, 112, 112]	1,280
BatchNorm2d-47	[-1, 128, 112, 112]	256
ReLU-48	[-1, 128, 112, 112]	0
Conv2d-49	[-1, 128, 112, 112]	16,512
BatchNorm2d-50	[-1, 128, 112, 112]	256
BasicBlock-51	[-1, 128, 112, 112]	0
Linear-52	[-1, 11]	1,419

==

Total params: 96,587

Trainable params: 96,587

Non-trainable params: 0

```
----------------------------------------------------------------
Input size (MB): 0.57
Forward/backward pass size (MB): 882.00
Params size (MB): 0.37
Estimated Total Size (MB): 882.94
----------------------------------------------------------------
```

Q2 Knowledge Distillation
1 分數

Q2.1 Knowledge Distillation with KL Divergence Loss
0.5 分數

Copy&Paste the contents in the whole code block of your loss_fn_kd implementation. (Simply choose alpha=0.5, temperature=1.0 for the keyword arguments.)
For example:

```python
def loss_fn_kd(student_logits, labels, teacher_logits, alpha=0.5, temperature=1.0):
    # You need to implement this loss function by your own.
    pass
```

Your loss_fn_kd **code**(text only).

```python
def loss_fn_kd(student_logits, teacher_logits, y_a, y_b=None, lam=0, alpha=0.3, temperature=2.0):
    p = F.softmax(student_logits / temperature, dim=-1)
    q = F.softmax(teacher_logits / temperature, dim=-1)
    kl_loss = nn.KLDivLoss(reduction='batchmean')(p, q)
    loss_fn = nn.CrossEntropyLoss()
    if cfg['MIXUP'] and y_b is not None:
        ce_loss = mixup_criterion(loss_fn, student_logits, y_a, y_b, lam)
    else:
        ce_loss = loss_fn(student_logits, y_a)
    loss = (alpha*temperature**2) * kl_loss + (1 - alpha) * ce_loss
    return loss
```

Q2.2 Understanding Temperature in Knowledge Distillation
0.5 分數

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

Using a higher value for T produces a harder probability distribution over classes.

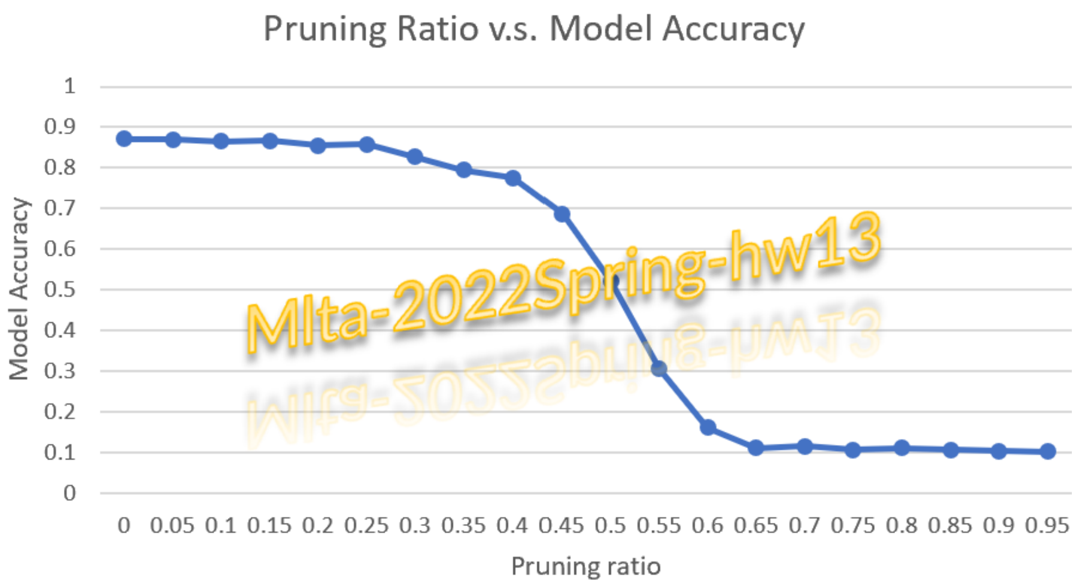
✓ Using a higher value for T produces a softer probability distribution over classes.

Q3 Network Pruning
2 分數

Q3.1 Pruning Ratio v.s. Model Accuracy
1 分數

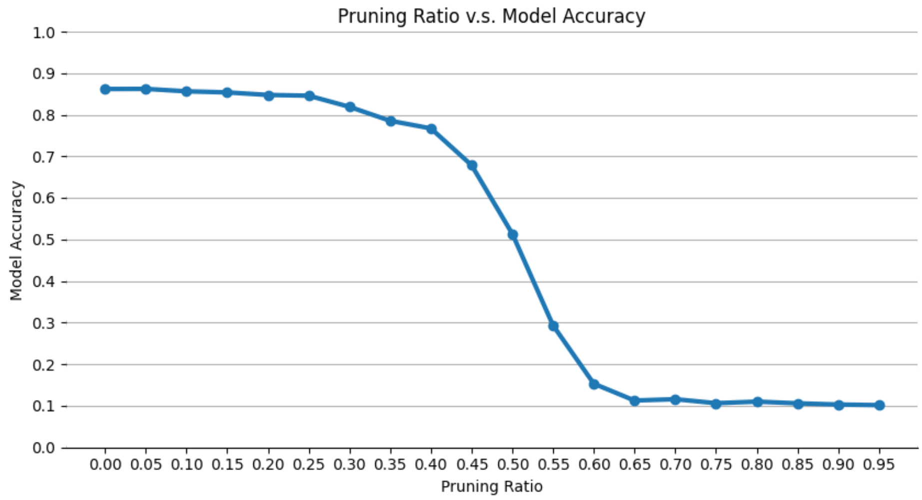
Please go through the official pytorch network pruning tutorial first.
In this question, you are asked to plot a graph to indicate the relationship between **pruning ratio** and **model accuracy**. You can use the provided teacher model or your own teacher model. And to be more specific, please prune the parameters named with `weight` in all of the `nn.Conv2d` layers, with different ratios. If you are still confused about how to implement it, we also have provided an example on google colab.

Example plot:



Please upload your .jpg/.png file below.

▾ Q3-1.png Download



> We encourage you to do the same thing on your student network and compare the results, but there is no need to submit the plot of the student network.

Q3.2 Network Pruning and Inference Speed
1 分數

According to your understanding of the network pruning methods in the tutorial (torch.nn.utils.prune), can this kind of approach reduce the model inference time? Choose your answer below.
If you have no idea, we encourage you to do some experiments about it. (e.g., measure the inference time of a non-pruned model and a 90%-pruned model.)

- Yes. After pruning, the neurons are removed so the amount of calculation is lower and the inference time is faster.
- ✓ No. Such pruning technique in the tutorial is just to add some masks to the modules. The amount of calculation is nearly the same so the inference time is also similar.

HW13

● 已批改

總分
3.75 / 4 pts

問題 1

Architecture Design 1 / 1 pt

1.1 Student Model Architecture 0.5 / 0.5 pts

1.2 Torchsummary of Student Model 0.5 / 0.5 pts

問題 2

Knowledge Distillation 0.75 / 1 pt

2.1 Knowledge Distillation with KL Divergence Loss 0.25 / 0.5 pts

2.2 Understanding Temperature in Knowledge Distillation 0.5 / 0.5 pts

問題 3

Network Pruning 2 / 2 pts

3.1 Pruning Ratio v.s. Model Accuracy 1 / 1 pt

3.2 Network Pruning and Inference Speed 1 / 1 pt