MP4 Q1

1. BaseNet:

Network Structure:

```
BaseNet(
  (conv1): Conv2d(1, 6, kernel_size=(5, 5), stride=(1, 1))
  (relu1): ReLU()
  (pool1): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv2): Conv2d(6, 12, kernel_size=(5, 5), stride=(1, 1))
  (relu2): ReLU()
  (pool2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (fc1): Linear(in_features=192, out_features=12, bias=True)
  (relu3): ReLU()
  (fc2): Linear(in_features=12, out_features=10, bias=True)
)
```

Final Accuracy on validation images:

```
Accuracy of the final network on the val images: 63.5 %
Accuracy of T-shirt/top : 71.8 %
Accuracy of Trouser : 87.9 %
Accuracy of Pullover : 33.1 %
Accuracy of Dress : 76.6 %
Accuracy of  Coat : 47.8 %
Accuracy of Sandal : 38.8 %
Accuracy of Shirt : 16.3 %
Accuracy of Sneaker : 85.8 %
Accuracy of   Bag : 85.2 %
Accuracy of Ankle boot : 89.1 %
```

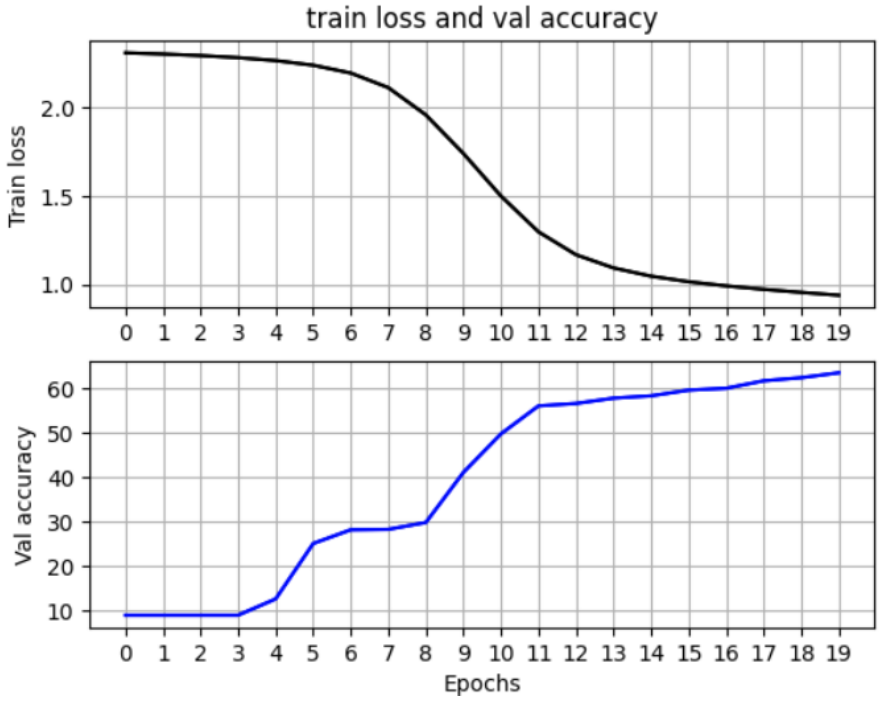
2. Improved Basenet

After a few adjustments, tried several layer combinations. The improved network is able to reach 90% accuracy, in general, I made some adjustments about the input channels, the original channels was 6 to 12 in the first convolution layer, I changed it to 1(from grayscale images) to 12 output channels. Similarly, I expanded the output channels for the next convolutional layer, this helped me to improve the performance of the BaseNet. A possible reason for this improvement is that with more filters, the network can learn a richer set of features from the input image, allowing it to capture more patterns and details that might be helpful in distinguishing between classes. After the two convolutional layers, I added two fully connected layers. One is fully connected layer with 384 input features and 192 output features, mapping the high-dimensional feature maps to a lower-dimensional space, and another fully connected (linear) layer, this time with 192 input features and 12 output features. Between them I added a regularization (drop out) layer to avoid overfitting. Each of the Linear layer were also followed by a Relu activation layer.

Final accuracy:

On Local machine:

```
Accuracy of the final network on the val images: 89.8 %
Accuracy of T-shirt/top : 85.8 %
Accuracy of Trouser : 97.2 %
Accuracy of Pullover : 82.4 %
Accuracy of Dress : 94.3 %
Accuracy of  Coat : 83.7 %
Accuracy of Sandal : 96.2 %
Accuracy of Shirt : 65.7 %
Accuracy of Sneaker : 97.6 %
Accuracy of   Bag : 97.9 %
Accuracy of Ankle boot : 96.0 %
```



Ablation table:

Components Adjusted	Validation Accuracy
Baseline	63.5%
Added the BatchNorm to the convo layers	70.1%
Added the first linear layer	82.3%
Added the second linear layer	88%
Increase the channel for the convo layers	90.2% on Gradescope/89.8% on local

Final architecture:

The Network structure is defined in the following table:

Layer No.	Layer Type	Kernel Size	Input Dim	Output Dim	Input Channels	Output Channels
1	conv2d	5	28	24	1	12
2	BatchNorm2d	-	24	24	12	12
3	relu	-	24	24	12	12
4	maxpool2d	2	24	12	12	12
5	conv2d	5	12	8	12	24
6	BatchNorm2d	-	8	8	24	24
7	relu	-	8	8	24	24
8	maxpool2d	2	8	4	24	24
9	linear	-	384	192	-	-

Layer No.	Layer Type	Kernel Size	Input Dim	Output Dim	Input Channels	Output Channels
10	dropout	-	192	192	-	-
11	relu	-	192	192	-	-
12	linear	-	192	12	-	-
13	dropout	-	12	12	-	-
14	relu	-	12	12	-	-
15	linear	-	12	10	-	-

```
print(net)
```

```
BaseNet(
  (conv1): Conv2d(1, 12, kernel_size=(5, 5), stride=(1, 1))
  (bn1): BatchNorm2d(12, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (relu1): ReLU()
  (pool1): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv2): Conv2d(12, 24, kernel_size=(5, 5), stride=(1, 1))
  (bn2): BatchNorm2d(24, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (relu2): ReLU()
  (pool2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (fc1): Linear(in_features=384, out_features=192, bias=True)
  (dropout1): Dropout(p=0.5, inplace=False)
  (relu3): ReLU()
  (fc2): Linear(in_features=192, out_features=12, bias=True)
  (dropout2): Dropout(p=0.5, inplace=False)
  (relu4): ReLU()
  (fc3): Linear(in_features=12, out_features=10, bias=True)
)
```

3. Final Submission On Gradescope:

```
Q1 evaluation results:
Accuracy: 90.2 %
Accuracy of T-shirt/top : 88.9 %
Accuracy of Trouser : 97.9 %
Accuracy of Pullover : 78.8 %
Accuracy of Dress : 91.8 %
Accuracy of  Coat : 85.7 %
Accuracy of Sandal : 96.9 %
Accuracy of Shirt : 72.5 %
Accuracy of Sneaker : 96.6 %
Accuracy of   Bag : 98.9 %
Accuracy of Ankle boot : 96.4 %
```

MP4 Q2:

1. Implement the training cycle

I did not change too much on the training cycle, I kept the original CrossEntrophy as the criterion, and in general kept the original training cycle skeleton, added a scheduler to adjust the learning rate.

2. Build on top of ImageNet pre-trained Model

I first located the classifier and avgpooling layers from the Resnet, which are last two layers, and removed them. Then I added one convolutional layers after the Resnet layers and a dropout layer to avoid overfitting, eventually upsample the output with factor=32. I used nn.Upsample(factor = 32) to increase my final resolution size.

Model Architecture:

I am kind of confused by the definition in the problem about "stack an additional layer", I added a simple convolution layer after the ResNet18 along with a dropout layer to avoid overfitting. Here's my architecture. Note that this only shows the Architecture after the ResNet18:

Layer No.	Layer Type	Kernel Size	Input Dim	Output Dim	Input Channels	Output Channels
1	dropout	-	-	-	-	-

Layer No.	Layer Type	Kernel Size	Input Dim	Output Dim	Input Channels	Output Channels
2	conv2d	3	7	7	512	256
3	relu	-	7	7	256	256
4	upsample	-	7	14	256	256

Training details:

I tried several learning rate in this section, and kept Epoch = 20, optimizer = Adam.

The scheduler will adjust the learning rate to be the 90% of the original learning rate every 10 epochs. I also tried to make the Epoch=30, but the performance stopped to improve after Epoch=20

Learning Rate	AP/IOU
1e-3	0.66/0.44
1e-4	0.70/0.48
1e-5	0.65/0.44

When learning rate start at 1e-4, both the accuracy and the IoU reached highest without further adjusting the network design.

```
poch 20, Loss: 1.0001
00%|████████████| 20/20 [01:37<00:00,  4.89s/it]          background: AP: 0.87, IoU: 0.63
              sports: AP: 0.42, IoU: 0.24
           accessory: AP: 0.52, IoU: 0.33
              animal: AP: 0.88, IoU: 0.61
             vehicle: AP: 0.73, IoU: 0.52
              person: AP: 0.79, IoU: 0.55
                mean: AP: 0.70, IoU: 0.48
his is validation during the train
```

3. Improve model performance

Starting from what I got in Part2. I choose to add additional fully convolutional layers to improve the model performance. In total I added 3 fully convolutional layers. But I wasn't able to get full mark on Gradescope (29/30),

```
 95%|████████████ | 19/20 [01:23<00:03,  3.80s/it]Epoch 19, Loss: 0.9564
Epoch 20, Loss: 0.9648
100%|████████████| 20/20 [01:35<00:00,  4.77s/it]          background: AP: 0.90, IoU: 0.70
              sports: AP: 0.52, IoU: 0.33
           accessory: AP: 0.57, IoU: 0.39
              animal: AP: 0.89, IoU: 0.65
             vehicle: AP: 0.75, IoU: 0.58
              person: AP: 0.81, IoU: 0.57
                mean: AP: 0.74, IoU: 0.54
This is validation during the train
```

The structure of the final network is as follows, I only adjusted the structure after the ResNet, I did not include the Resnet structure here:

Layer No.	Layer Type	Kernel Size	Input Dim	Output Dim	Input Channels	Output Channels
1	conv2d	3	7	7	512	256
2	batchnorm2d	-	7	7	256	256
3	relu	-	7	7	256	256
4	upsample	-	7	14	256	256
5	conv2d	3	14	14	256	128
6	batchnorm2d	-	14	14	128	128
7	relu	-	14	14	128	128
8	upsample	-	14	28	128	128
9	conv2d	3	28	28	128	n_classes
10	batchnorm2d	-	28	28	n_classes	n_classes
11	relu	-	28	28	n_classes	n_classes
12	upsample	-	28	224	n_classes	n_classes

I first implemented the dilation convolution by adjusting the 7th layer within the ResNet by:

```
        dilation = 4
        self.features[7][1].conv1 = nn.Conv2d(512, 512, kernel_size=3, stride=1, padding=dilation, dilation=dilation, bias=False)
        self.features[7][1].conv2 = nn.Conv2d(512, 512, kernel_size=3, padding=dilation, dilation=dilation, bias=False)
```

And found the dilation did not boost the performance of network but decreased the performance, after several adjustments, I decided to add more fully convolutional layers to boost my network. I tried to add 2/3/4/5 convolutional layers, after testing and validating, I managed to find that when num of convolutional layers == 3, the model performed the best:

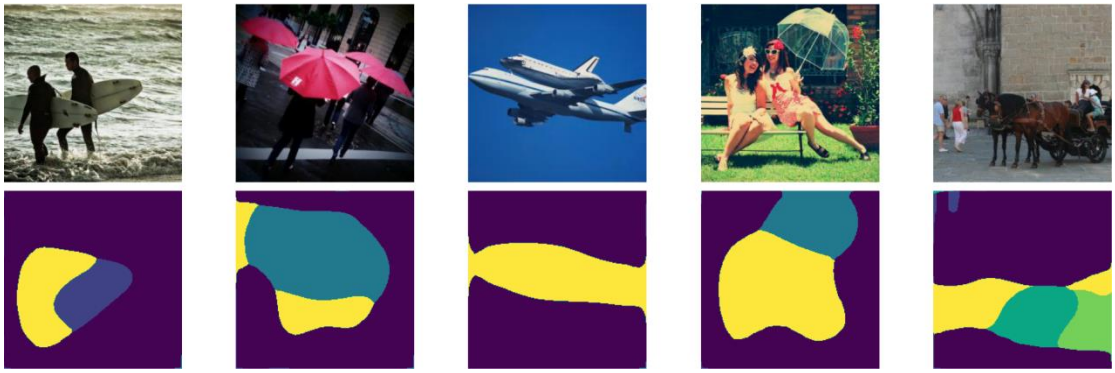
Ablation table:

Components Adjusted	AP/IoU
Baseline(num of convolutional layers ==1)	0.70/0.48
num of convolutional layers == 2	0.70/0.49
num of convolutional layers == 3	0.74/0.54
num of convolutional layers == 4	0.73/0.52

num of convolutional layers == 5	0.72/0.53
Dilation strides =4	0.70/0.47
Dilation strides = 8	0.68/0.45
Dilation strides = 16	0.67/0.44

Note that all of the output was upsampled factor = 32 after the convolutional layers.

Visualization of the best model:



Gradescope autograder results(29/30):

```
Q2 evaluation results:
mean: IoU: 0.53
mean: AP: 0.74
background: AP: 0.92, IoU: 0.76
sports: AP: 0.43, IoU: 0.23
accessory: AP: 0.58, IoU: 0.38
animal: AP: 0.86, IoU: 0.64
vehicle: AP: 0.78, IoU: 0.56
person: AP: 0.85, IoU: 0.62
```