◆ CIFAR-10 with different Network

In practice 07, you might probably get accuracy 7x% by using LeNET with data augmentation. In this practice, you are asked to build other models to improve the accuracy. At last, show the result compared with LeNET you trained in practice 07. Follow hints below to finish it.

▫ Practice：

■ Load model：

PyTorch provides solutions to a variety of use cases regarding the saving and loading of models. When it comes to saving and loading models, there are some core functions to be familiar with:

1. **torch.save/torch.load**

   e.g.：

Save/Load Entire Model	
Save	**Load**
torch.save(model, PATH)	*# Model class must be defined somewhere* **model = torch.load(PATH)** **model.eval()**

2. **torch.nn.Module.load_state_dict** (use in practice 07)

   state_dict objects are Python dictionaries, they can be easily saved, updated, altered, and restored, adding a great deal of modularity to PyTorch models and optimizers.

Save/Load state_dict *(Recommend)*	
Save	**Load**
torch.save(model.state_dict(), PATH)	model = TheModelClass(*args, **kwargs) model.load_state_dict(torch.load(PATH)) model.eval()

■ AlexNet (2012)：

AlexNet is a convolutional neural network designed by Alex Krizhevsky. AlexNet contained eight layers：the first five were convolutional layers, some of them followed by max-pooling layers, and the last three were fully connected layers. It used the non-saturating ReLU activation function, which showed improved training performance over tanh and sigmoid. ReLUs have the desirable property that they do not require intput normalization to prevent them from saturating. AlexNet use two method to decrease overfitting problem, one

is “Dropout” and another one is “Data Augmentation”.

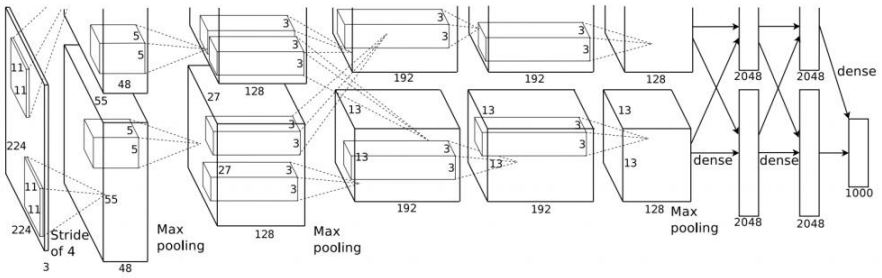
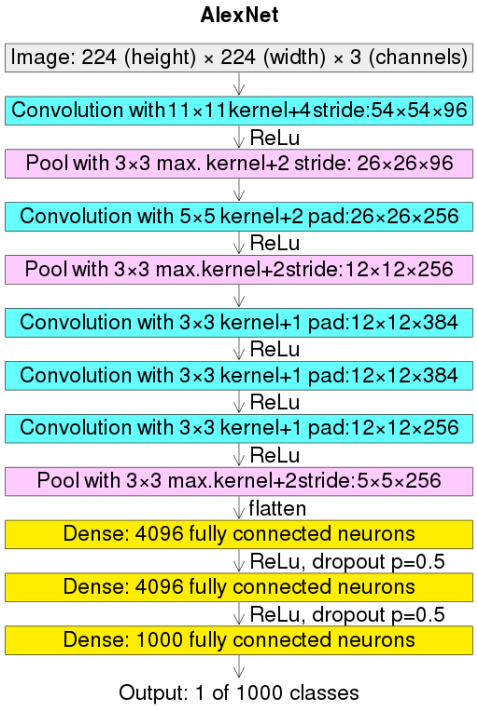


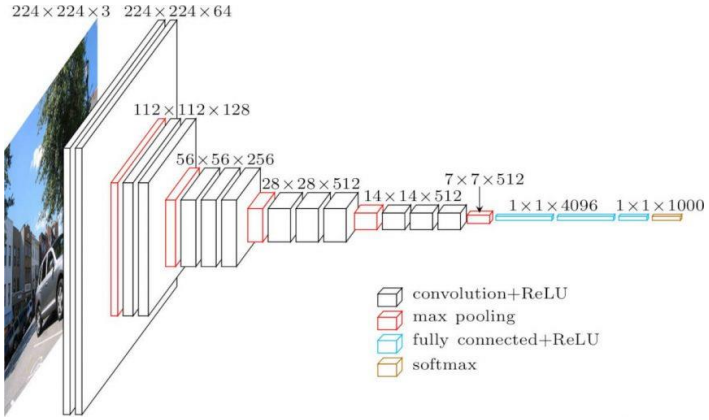
Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network’s input is 150,528-dimensional, and the number of neurons in the network’s remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

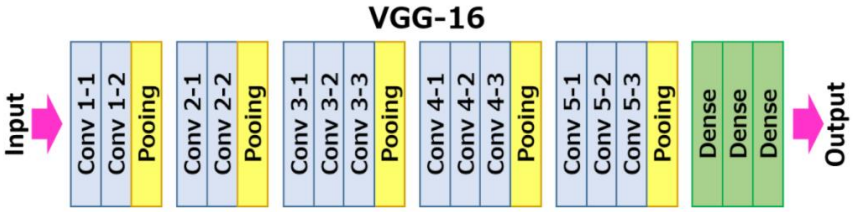


■ VGGNet (2014)：

VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper “Very Deep Convolutional Networks for Large-Scale Image Recognition”. It makes the improvement over AlexNet by replacing large kernel-sized filters (11 and 5 in the first and second convolutional layer, respectively) with multiple 3×3 kernel-sized filters one after another.

- **VGG16**





- **ConvNet Configuration**

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

[Please find reference for more details]

[Hint]

Hint 01：Import library you needed.

Hint 02：Check and set GPU.

Hint 03：Load the CIFAR-10 datasets.

Hint 04：Load LeNET which trained in practice 07 and obtain the testing accuracy.

```
model = LeNet().to(device)
model.load_state_dict(torch.load('./model/Cifar-10_LeNet_best_with_aug1.pkl'))
```

```
1 _, _, _, tmp_high_aug2 = evaluate(model, _, _, classes, test_loader, 0.0, None)
```

```
Now for eval!
 Test result --> Acc: 72.860001%
```

Hint 05：Define Hyper parameters by yourself.

Recommend initial parameters：

- learning_rate = 0.001
- batch_size = 64~128
- EPOCH：30~50
- Optimizer：Adam
- Loss function：CrossEntrpyLoss()

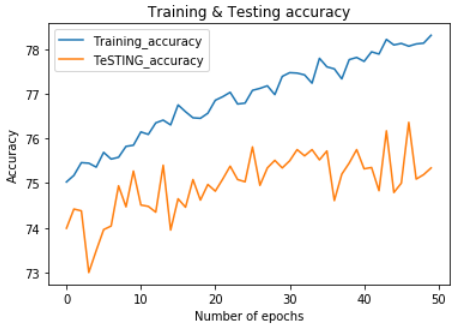
(You can tune by yourself.)

Hint 05：Construct AlexNet through structure provided below, then show the **loss-epoch** curve and **accuracy-epoch** curve. At the last epoch, please show the classified result and highest testing accuracy as shown below.

[You CANNOT use torchvision.models.AlexNet]

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 6, 32, 32]             168
              ReLU-2          [-1, 6, 32, 32]               0
         MaxPool2d-3          [-1, 6, 16, 16]               0
            Conv2d-4         [-1, 16, 16, 16]             880
              ReLU-5         [-1, 16, 16, 16]               0
         MaxPool2d-6           [-1, 16, 8, 8]               0
            Conv2d-7           [-1, 32, 8, 8]           4,640
              ReLU-8           [-1, 32, 8, 8]               0
         MaxPool2d-9           [-1, 32, 4, 4]               0
           Conv2d-10           [-1, 64, 4, 4]          18,496
             ReLU-11           [-1, 64, 4, 4]               0
        MaxPool2d-12           [-1, 64, 2, 2]               0
           Conv2d-13          [-1, 128, 2, 2]          73,856
             ReLU-14          [-1, 128, 2, 2]               0
        MaxPool2d-15          [-1, 128, 1, 1]               0
           Linear-16                [-1, 120]          15,480
             ReLU-17                [-1, 120]               0
           Linear-18                 [-1, 84]          10,164
             ReLU-19                 [-1, 84]               0
           Linear-20                 [-1, 10]             850
================================================================
```

```
--TESTING!!--
 Test result --> Acc: 75.339996%
 Come on, keep going!
 Accuracy of each classes
 Test Accuracy of plane: 77.9%, [779/1000]
 Test Accuracy of car: 88.5%, [885/1000]
 Test Accuracy of bird: 64.0%, [640/1000]
 Test Accuracy of cat: 49.8%, [498/1000]
 Test Accuracy of deer: 66.3%, [663/1000]
 Test Accuracy of dog: 68.4%, [684/1000]
 Test Accuracy of frog: 88.4%, [884/1000]
 Test Accuracy of horse: 80.0%, [800/1000]
 Test Accuracy of ship: 87.5%, [875/1000]
 Test Accuracy of truck: 82.6%, [826/1000]
 --> Highest testing value so far :  76.36000061035156
```

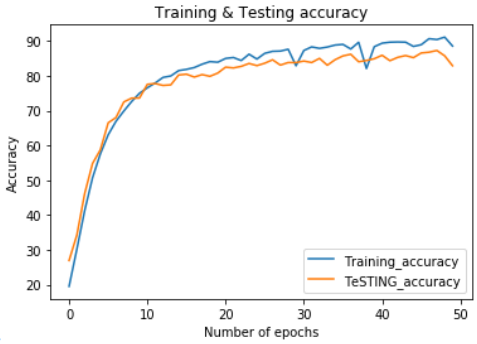


Hint 06：Construct VGG-net through structure provided below, then show the **loss-epoch** curve and **accuracy-epoch** curve. At the last epoch, please show the classified result and highest testing accuracy as shown below.

[You CANNOT use torchvision.models.vgg]

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 64, 32, 32]           1,792
              ReLU-2          [-1, 64, 32, 32]               0
            Conv2d-3          [-1, 64, 32, 32]          36,928
              ReLU-4          [-1, 64, 32, 32]               0
         MaxPool2d-5          [-1, 64, 16, 16]               0
            Conv2d-6         [-1, 128, 16, 16]          73,856
              ReLU-7         [-1, 128, 16, 16]               0
            Conv2d-8         [-1, 128, 16, 16]         147,584
              ReLU-9         [-1, 128, 16, 16]               0
        MaxPool2d-10           [-1, 128, 8, 8]               0
           Conv2d-11           [-1, 256, 8, 8]         295,168
             ReLU-12           [-1, 256, 8, 8]               0
           Conv2d-13           [-1, 256, 8, 8]         590,080
             ReLU-14           [-1, 256, 8, 8]               0
           Conv2d-15           [-1, 256, 8, 8]         590,080
             ReLU-16           [-1, 256, 8, 8]               0
        MaxPool2d-17           [-1, 256, 4, 4]               0
           Conv2d-18           [-1, 512, 4, 4]       1,180,160
             ReLU-19           [-1, 512, 4, 4]               0
           Conv2d-20           [-1, 512, 4, 4]       2,359,808
             ReLU-21           [-1, 512, 4, 4]               0
           Conv2d-22           [-1, 512, 4, 4]       2,359,808
             ReLU-23           [-1, 512, 4, 4]               0
        MaxPool2d-24           [-1, 512, 2, 2]               0
           Conv2d-25           [-1, 512, 2, 2]       2,359,808
             ReLU-26           [-1, 512, 2, 2]               0
           Conv2d-27           [-1, 512, 2, 2]       2,359,808
             ReLU-28           [-1, 512, 2, 2]               0
           Conv2d-29           [-1, 512, 2, 2]       2,359,808
             ReLU-30           [-1, 512, 2, 2]               0
        MaxPool2d-31           [-1, 512, 1, 1]               0
           Linear-32                 [-1, 4096]       2,101,248
             ReLU-33                 [-1, 4096]               0
           Linear-34                 [-1, 1024]       4,195,328
             ReLU-35                 [-1, 1024]               0
           Linear-36                   [-1, 10]          10,250
================================================================
```

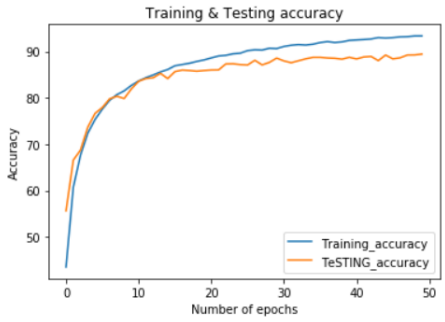
```
 Test result --> Acc: 82.909996%
 Come on, keep going!
 Accuracy of each classes
 Test Accuracy of plane: 82.7%, [827/1000]
 Test Accuracy of car: 92.2%, [922/1000]
 Test Accuracy of bird: 71.2%, [712/1000]
 Test Accuracy of cat: 57.7%, [577/1000]
 Test Accuracy of deer: 87.3%, [873/1000]
 Test Accuracy of dog: 79.6%, [796/1000]
 Test Accuracy of frog: 87.6%, [876/1000]
 Test Accuracy of horse: 88.5%, [885/1000]
 Test Accuracy of ship: 89.1%, [891/1000]
 Test Accuracy of truck: 93.2%, [932/1000]
 --> Highest testing value so far :  87.29000091552734
```



Hint 8：Construct a three layers CNN through structure provided below, then show the **loss-epoch** curve and **accuracy-epoch** curve. At the last epoch, please show the classified result and highest testing accuracy as shown below.

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 32, 32, 32]             896
       BatchNorm2d-2          [-1, 32, 32, 32]              64
              ReLU-3          [-1, 32, 32, 32]               0
            Conv2d-4          [-1, 64, 32, 32]          18,496
              ReLU-5          [-1, 64, 32, 32]               0
         MaxPool2d-6          [-1, 64, 16, 16]               0
            Conv2d-7         [-1, 128, 16, 16]          73,856
       BatchNorm2d-8         [-1, 128, 16, 16]             256
              ReLU-9         [-1, 128, 16, 16]               0
           Conv2d-10         [-1, 128, 16, 16]         147,584
             ReLU-11         [-1, 128, 16, 16]               0
        MaxPool2d-12           [-1, 128, 8, 8]               0
        Dropout2d-13           [-1, 128, 8, 8]               0
           Conv2d-14           [-1, 256, 8, 8]         295,168
      BatchNorm2d-15           [-1, 256, 8, 8]             512
             ReLU-16           [-1, 256, 8, 8]               0
           Conv2d-17           [-1, 256, 8, 8]         590,080
             ReLU-18           [-1, 256, 8, 8]               0
        MaxPool2d-19           [-1, 256, 4, 4]               0
          Dropout-20                 [-1, 4096]               0
           Linear-21                 [-1, 1024]       4,195,328
             ReLU-22                 [-1, 1024]               0
           Linear-23                  [-1, 512]         524,800
             ReLU-24                  [-1, 512]               0
          Dropout-25                  [-1, 512]               0
           Linear-26                   [-1, 10]           5,130
================================================================
```

```
Test result --> Acc: 88.669998%
-> Save the best model & value so far ~
Accuracy of each classes
Test Accuracy of plane: 86.6%, [866/1000]
Test Accuracy of car: 92.5%, [925/1000]
Test Accuracy of bird: 83.3%, [833/1000]
Test Accuracy of cat: 80.4%, [804/1000]
Test Accuracy of deer: 85.4%, [854/1000]
Test Accuracy of dog: 86.8%, [868/1000]
Test Accuracy of frog: 92.6%, [926/1000]
Test Accuracy of horse: 92.7%, [927/1000]
Test Accuracy of ship: 92.8%, [928/1000]
Test Accuracy of truck: 93.6%, [936/1000]
--> Highest testing value so far :  88.66999816894531
```



Hint 9：Plot the **compare result table** which contains testing accuracy of LeNet, AlexNet, VGG-16 and three layers-CNN.

Compare Reult with 50 epoch

Version	LeNet with augmentation	AlexNet	VGG-16	3 layers CNN
Best Accuracy (%)	72.860	76.360	87.290	88.670

[Please share your result and perspective of these Networks]

Reference：

- **Sample code** [Link]
- PyTorch model save/load *(Chinese)* [Link]
- PyTorch model save/load [Link]
- **AlexNet**
  - AlexNet overview [Link]
  - Deep Convolutional Neural Networks (AlexNet) [Link]
  - AlexNet: The First CNN to win Image Net [Link]
- **VGG Net**
  - VGG_深度學習_原理 [Link]
  - What is the VGG neural network? [Link]
  - VGG Neural Networks: The Next Step After AlexNet [Link]

Reference paper：[Link]

- Krizhevsky, A., Ilya Sutskever and Geoffrey E. Hinton. “ImageNet classification with deep convolutional neural networks.” *Communications of the ACM* 60 (2012): 84 - 90. *[Link] [中文分享]*

- Simonyan, Karen & Zisserman, Andrew. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv 1409.1556. *[Link] [中文分享]*