SC201 Lecture 15

PyTorch

< Data Pre-processing >

import torchvision.transforms as T

transform = T.Compose([T.Resize((64,64)), T.ToTensor( ), T.Normalize(mean=mean, std=std)])

< Load Data >

import torchvision.datasets as dset

train = dset.ImageFolder(path , transform=transform)

[(Tensor$_1$, int$_1$), (Tensor$_2$, int$_2$), … (Tensor$_m$, int$_m$)]

< Create Mini-batches >

from torch.utils.data import DataLoader

mini_trains = DataLoader(train, batch_size=BATCH_SIZE, shuffle=True)

< Model Building & ForwardProp >

model = nn.Sequential (

⋮

)

scores = model(input)

< Calculating Loss & BackProp >

loss_function = _____ → _____ = _____

loss_function = _____

loss = loss_function(scores, y) ⟶ loss = loss_function(_____)

< Optimizer >

optimizer = optim.Adam(model.parameters( ))

_____

loss.backward( )

_____

< Training >

model = model.to(device = _____)

for e in range(epochs):

    for _____, (x, y) in enumerate(iterator):

        model.train( )

< Summary >

from torchsummary import summary

summary(model, (3, 32, 32))

Data Augmentation

CNN is _____

- _____
- _____
- _____
- _____

- _____
- _____
- _____
- _____

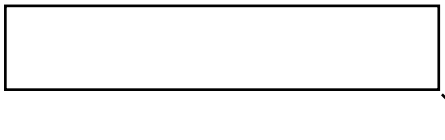
```
transform = T.Compose([
            T.ToTensor(),
            _____,
            ⋮
            T.Normalize(…)
        ])
```

_____ = nn.Conv2d(3, 64, 3, 1, 1)

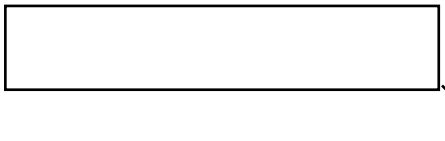
_____ = _____(_____)

_____

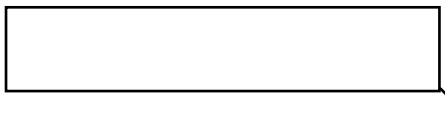
relu = _____



max_pool2d = _____



flatten = _____



```
model = nn.Sequential (
        nn.Conv2d(3, 64, 3, 1, 1),
        nn.ReLU(),
        nn.MaxPool2d(2, 2),

        nn.Conv2d(64, 128, 3, 1, 1),
        nn.ReLU(),
        nn.MaxPool2d(2, 2),

        nn.Flatten()
        nn.Linear( 8*8*128, 10)
    )
```

_____

```
class  MyCNN(nn.Module):
    def _ _init_ _(self):
        _____
        _____
        _____
        _____
    def foward(self):
        x = self.conv1(x)
        x = F.relu( )
        x = F.max_pool2d(x, (2, 2))
        x = self.conv2(x)
        x = F.relu( )
        x = F.max_pool2d(x, (2, 2))
        x = torch.flatten(x, 1)
        return self.fc(x)
```

model = _____

model.conv1.weight

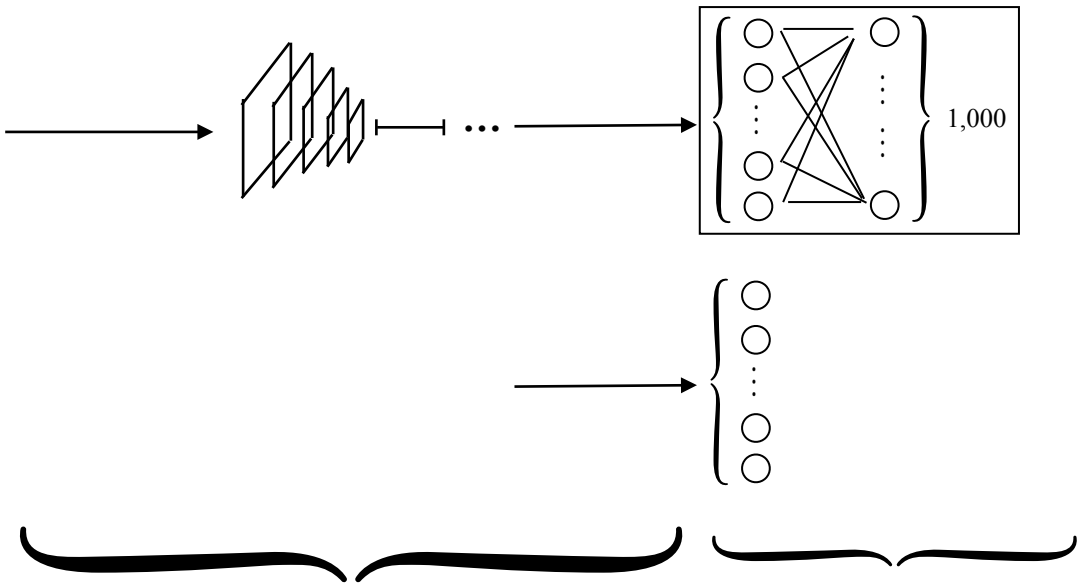
model.conv2.weight

model.fc.weight

_____

Transfer Learning

- Take a _____ model (VGGNet, GoogleNet, RestNet) / 比較：Human _____
- _____ all Conv Layers
- _____ affine layer (_____ layer)
- Train the affine layer



_____ _____

(_____)

< PyTorch >

```
from torchvision import models
resnet = models.resnet18(pretrained = True).cuda()
num_flatten = resnet.fc.in_features
resnet.fc = nn.Linear(num_flatten, 10)
model = resnet
optimizer = optim.Adam(model.parameters(), lr=1e-4)
train_part34(model, optimizer, epoch=1)
```

① mount to Google Drive

_____

② folder name

_____

③ import sys
sys.path.append(...)

④ %cd drive/MyDrive/$FOLDERNAME

< Load Existing Data >

```
import torchvision.datasets as dset

mnist_train = dset.MNIST(
                    './folder_train ',
                    train = True,
                    download = True
                )
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
mnist_train = dset.MNIST(
                    ‘./folder_test ',
                    train = False,
                    download = True
                )
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

img, label = mnist_train[0]
```