Dong-A Univ. (ISPL)

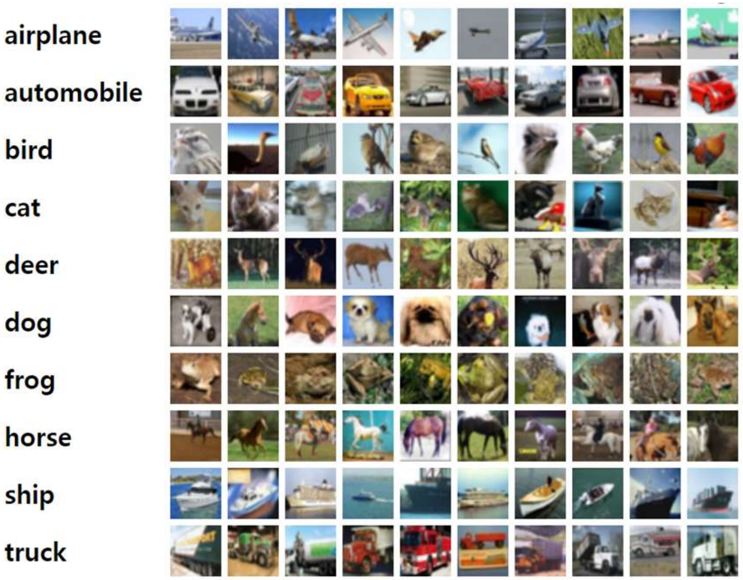
동아대학교
DONG-A UNIVERSITY

MLP 기반 CIFAR10 Classification

컴퓨터AI공학부 AI학과
2024년 1학기 인공지능

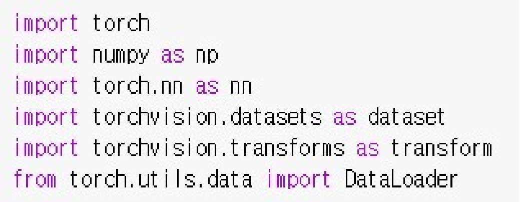
MLP 기반 CIFAR10 Classification

- CIFAR-10 dataset 형상
  - 32x32x3 (RGB) 이미지, 10개의 클래스
  - Train: 50,000개, Test: 10,000개



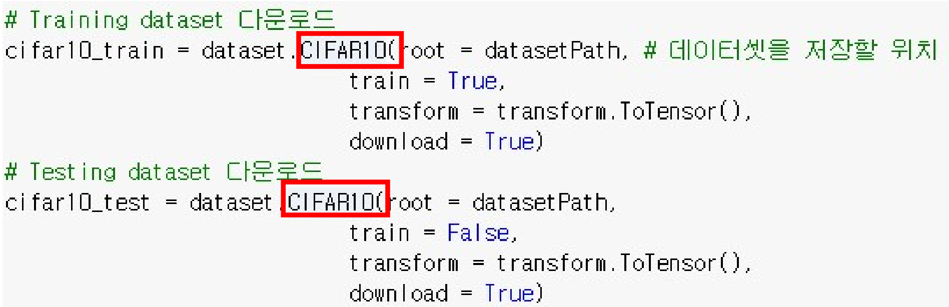
MLP 기반 CIFAR10 Classification

- 패키지 선언



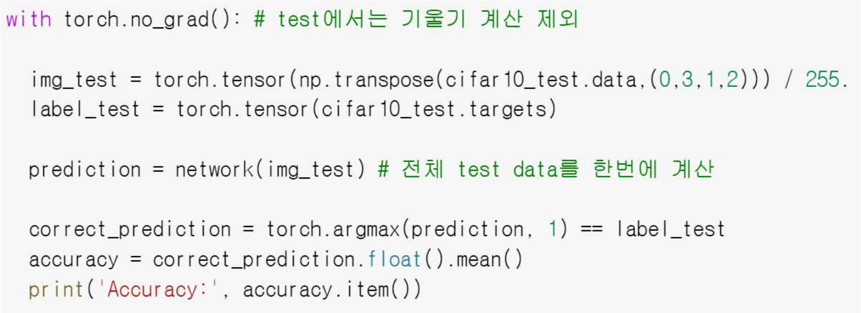
```
import torch
import numpy as np
import torch.nn as nn
import torchvision.datasets as dataset
import torchvision.transforms as transform
from torch.utils.data import DataLoader
```

- Dataset 선언 → **CIFAR-10 dataset으로 변경**



```
# Training dataset 다운로드
cifar10_train = dataset.CIFAR10(root = datasetPath, # 데이터셋을 저장할 위치
                                train = True,
                                transform = transform.ToTensor(),
                                download = True)
# Testing dataset 다운로드
cifar10_test = dataset.CIFAR10(root = datasetPath,
                               train = False,
                               transform = transform.ToTensor(),
                               download = True)
```

- Accuracy 측정 코드



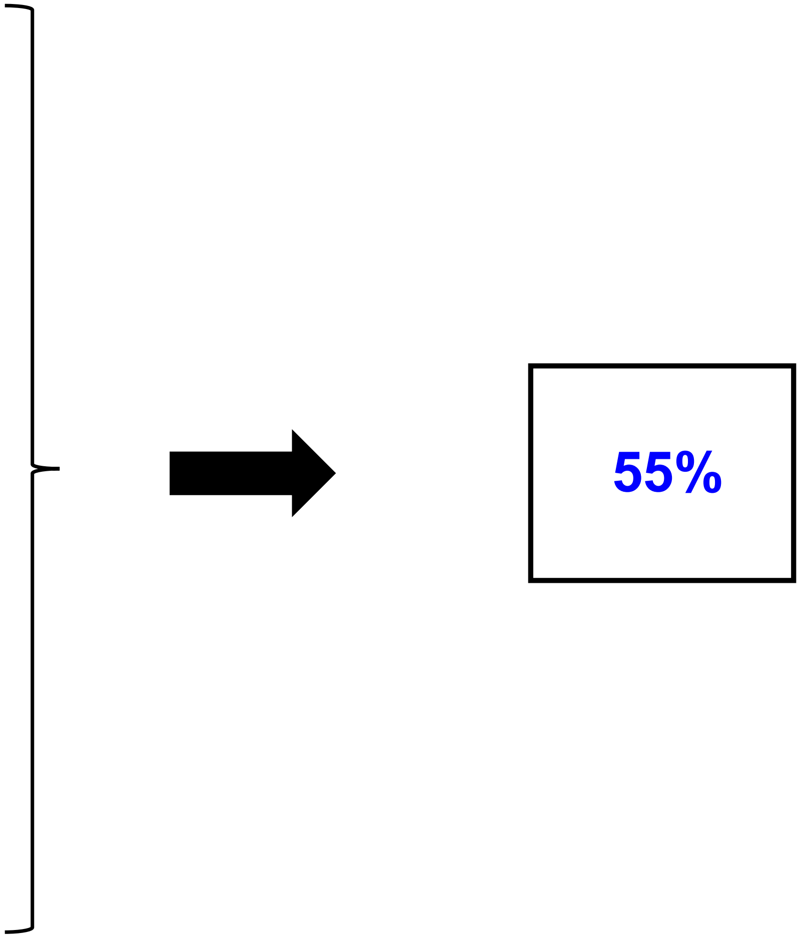
```
with torch.no_grad(): # test에서는 기울기 계산 제외

    img_test = torch.tensor(np.transpose(cifar10_test.data,(0,3,1,2))) / 255.
    label_test = torch.tensor(cifar10_test.targets)

    prediction = network(img_test) # 전체 test data를 한번에 계산

    correct_prediction = torch.argmax(prediction, 1) == label_test
    accuracy = correct_prediction.float().mean()
    print('Accuracy:', accuracy.item())
```

Requirement

- Batch size: ~200
- Learning rate: ~0.1
- Learning rate decay
- Activation function: ReLU
- Loss function: Cross Entropy
- Node: ~500
- Layer: ~6
- Epoch: ~20
- Batch normalization
- Drop-out
- Weight initialization
- Optimization

Backbone Network

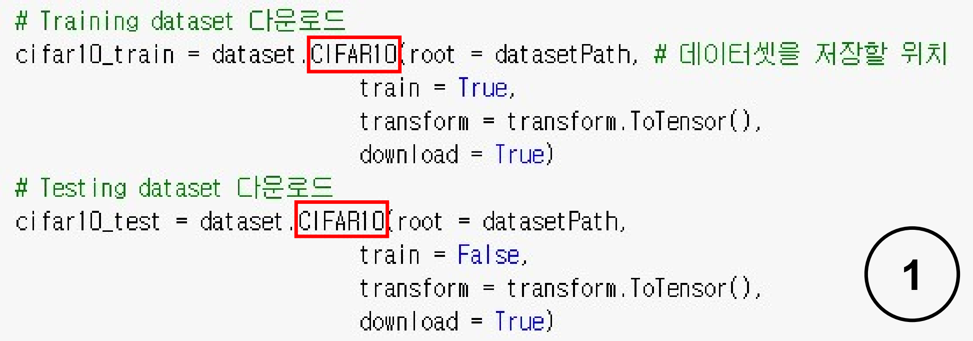
- Batch size: 200
- Learning rate: 0.1
- Learning rate decay : 10Epoch → x0.1
- Activation function: ReLU
- Loss function: Cross Entropy
- Node: 500
- Layer: 6
- Epoch: 20
- Batch normalization: o (ALL)
- Drop-out: o (0.1)
- Weight initialization: o (ALL)
- Optimization: SGD



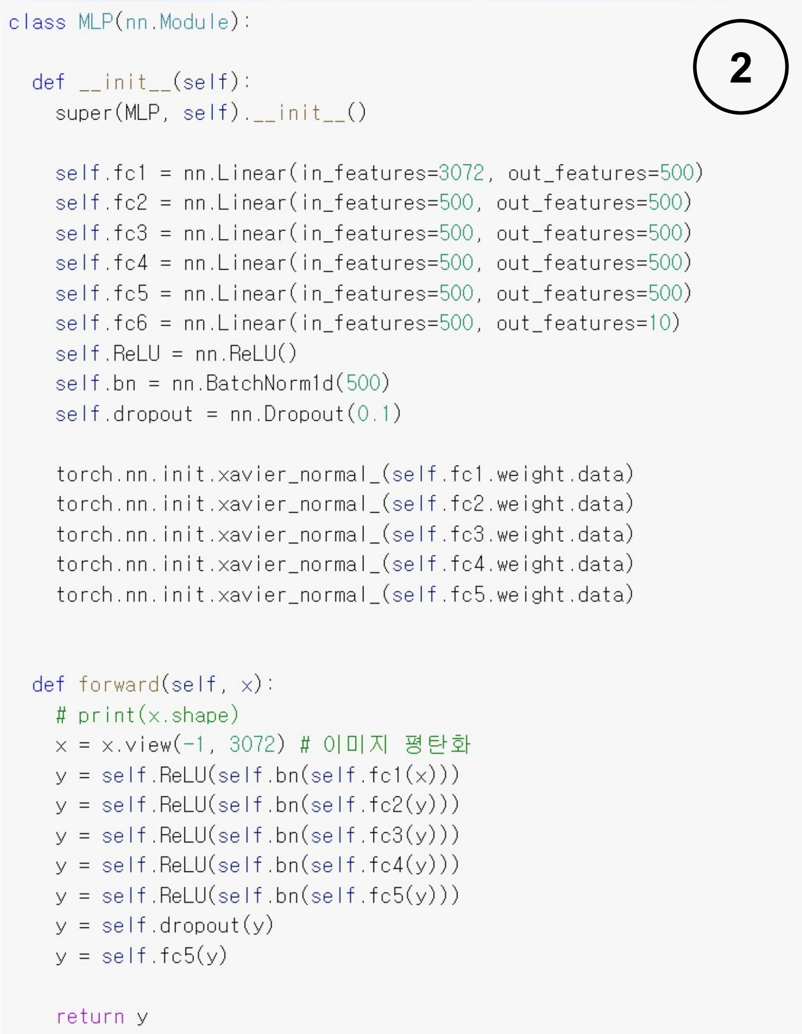
55%

Code

1. **CIFAR10 다운로드**
2. **모델 정의**

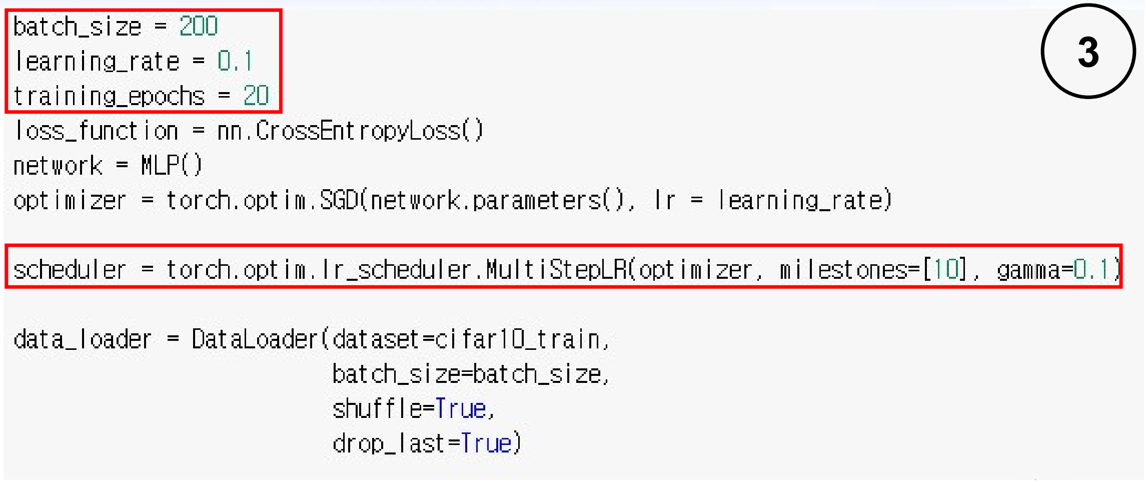


```
# Training dataset 다운로드
cifar10_train = dataset.CIFAR10(root = datasetPath, # 데이터셋을 저장할 위치
                                train = True,
                                transform = transform.ToTensor(),
                                download = True)
# Testing dataset 다운로드
cifar10_test = dataset.CIFAR10(root = datasetPath,
                               train = False,
                               transform = transform.ToTensor(),
                               download = True)
```



```
class MLP(nn.Module):

  def __init__(self):
    super(MLP, self).__init__()

    self.fc1 = nn.Linear(in_features=3072, out_features=500)
    self.fc2 = nn.Linear(in_features=500, out_features=500)
    self.fc3 = nn.Linear(in_features=500, out_features=500)
    self.fc4 = nn.Linear(in_features=500, out_features=500)
    self.fc5 = nn.Linear(in_features=500, out_features=500)
    self.fc6 = nn.Linear(in_features=500, out_features=10)
    self.ReLU = nn.ReLU()
    self.bn = nn.BatchNorm1d(500)
    self.dropout = nn.Dropout(0.1)

    torch.nn.init.xavier_normal_(self.fc1.weight.data)
    torch.nn.init.xavier_normal_(self.fc2.weight.data)
    torch.nn.init.xavier_normal_(self.fc3.weight.data)
    torch.nn.init.xavier_normal_(self.fc4.weight.data)
    torch.nn.init.xavier_normal_(self.fc5.weight.data)


  def forward(self, x):
    # print(x.shape)
    x = x.view(-1, 3072) # 이미지 평탄화
    y = self.ReLU(self.bn(self.fc1(x)))
    y = self.ReLU(self.bn(self.fc2(y)))
    y = self.ReLU(self.bn(self.fc3(y)))
    y = self.ReLU(self.bn(self.fc4(y)))
    y = self.ReLU(self.bn(self.fc5(y)))
    y = self.dropout(y)
    y = self.fc5(y)

    return y
```

Code



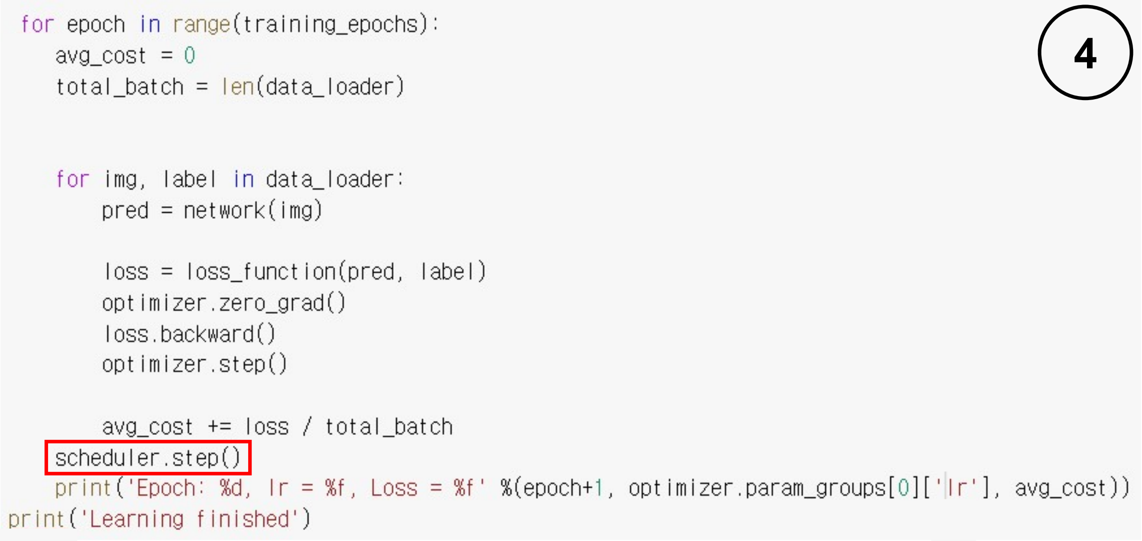
3

```
batch_size = 200
learning_rate = 0.1
training_epochs = 20
loss_function = nn.CrossEntropyLoss()
network = MLP()
optimizer = torch.optim.SGD(network.parameters(), lr = learning_rate)

scheduler = torch.optim.lr_scheduler.MultiStepLR(optimizer, milestones=[10], gamma=0.1)

data_loader = DataLoader(dataset=cifar10_train,
                         batch_size=batch_size,
                         shuffle=True,
                         drop_last=True)
```

3. Hyper-parameter 지정

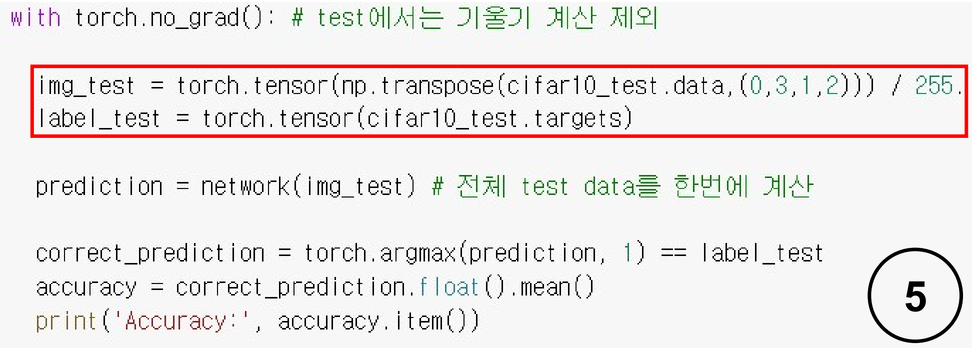
4. 학습을 위한 반복문 선언

5. 정답률 확인



4

```
for epoch in range(training_epochs):
   avg_cost = 0
   total_batch = len(data_loader)


   for img, label in data_loader:
       pred = network(img)

       loss = loss_function(pred, label)
       optimizer.zero_grad()
       loss.backward()
       optimizer.step()

       avg_cost += loss / total_batch
   scheduler.step()
   print('Epoch: %d, lr = %f, Loss = %f' %(epoch+1, optimizer.param_groups[0]['lr'], avg_cost))
print('Learning finished')
```



5

```
with torch.no_grad(): # test에서는 기울기 계산 제외

  img_test = torch.tensor(np.transpose(cifar10_test.data,(0,3,1,2))) / 255.
  label_test = torch.tensor(cifar10_test.targets)

  prediction = network(img_test) # 전체 test data를 한번에 계산

  correct_prediction = torch.argmax(prediction, 1) == label_test
  accuracy = correct_prediction.float().mean()
  print('Accuracy:', accuracy.item())
```

Questions & Answers

Dongsan Jun (dsjun@dau.ac.kr)
Image Signal Processing Laboratory (www.donga-ispl.kr)
Division of Computer･AI Engineering
Dong-A University, Busan, Rep. of Korea