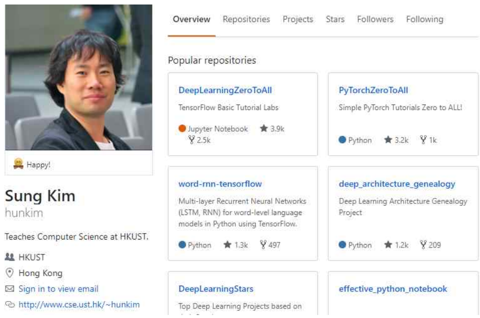
# 심층학습
# [실습04] 심층신경망 훈련

SW융합학부 양희경

# GitHub 로 실습코드 관리하길 추천합니다

- AWS SageMaker 5GB 제약
- 포트폴리오 작성법 익힘(미래의 나의 재산)
- 오픈소스에 기여
- 참고 GitHub
  - https://github.com/hunkim
  - https://github.com/hwalsuklee
  - https://github.com/sjchoi86



Happy!

Sung Kim
hunkim

Teaches Computer Science at HKUST.

HKUST
Hong Kong
Sign in to view email
http://www.cse.ust.hk/~hunkim

Overview Repositories Projects Stars Followers Following

Popular repositories

DeepLearningZeroToAll
TensorFlow Basic Tutorial Labs
● Jupyter Notebook ★ 3.9k 2.5k

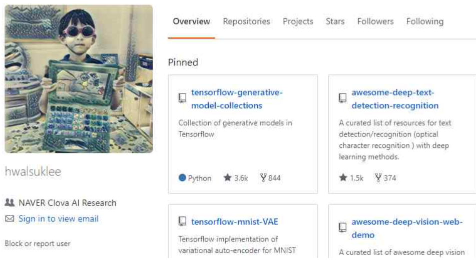
PyTorchZeroToAll
Simple PyTorch Tutorials Zero to ALL!
● Python ★ 3.2k 1k

word-rnn-tensorflow
Multi-layer Recurrent Neural Networks (LSTM, RNN) for word-level language models in Python using TensorFlow.
● Python ★ 1.3k 497

deep_architecture_genealogy
Deep Learning Architecture Genealogy Project
● Python ★ 1.2k 209

DeepLearningStars
Top Deep Learning Projects based on

effective_python_notebook



hwalsuklee

NAVER Clova AI Research
Sign in to view email

Block or report user

Overview Repositories Projects Stars Followers Following

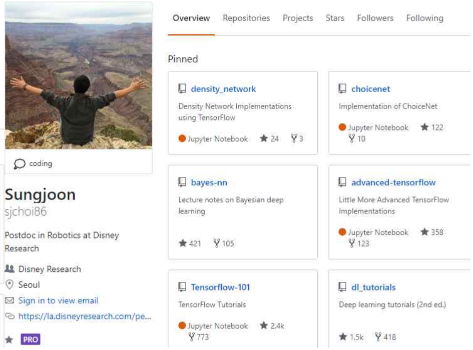
Pinned

tensorflow-generative-model-collections
Collection of generative models in Tensorflow
● Python ★ 3.6k 844

awesome-deep-text-detection-recognition
A curated list of resources for text detection/recognition (optical character recognition ) with deep learning methods.
★ 1.5k 374

tensorflow-mnist-VAE
Tensorflow implementation of variational auto-encoder for MNIST

awesome-deep-vision-web-demo
A curated list of awesome deep vision



coding

Sungjoon
sjchoi86

Postdoc in Robotics at Disney Research

Disney Research
Seoul
Sign in to view email
https://la.disneyresearch.com/pe...

★ PRO

Overview Repositories Projects Stars Followers Following

Pinned

density_network
Density Network Implementations using TensorFlow
● Jupyter Notebook ★ 24 3

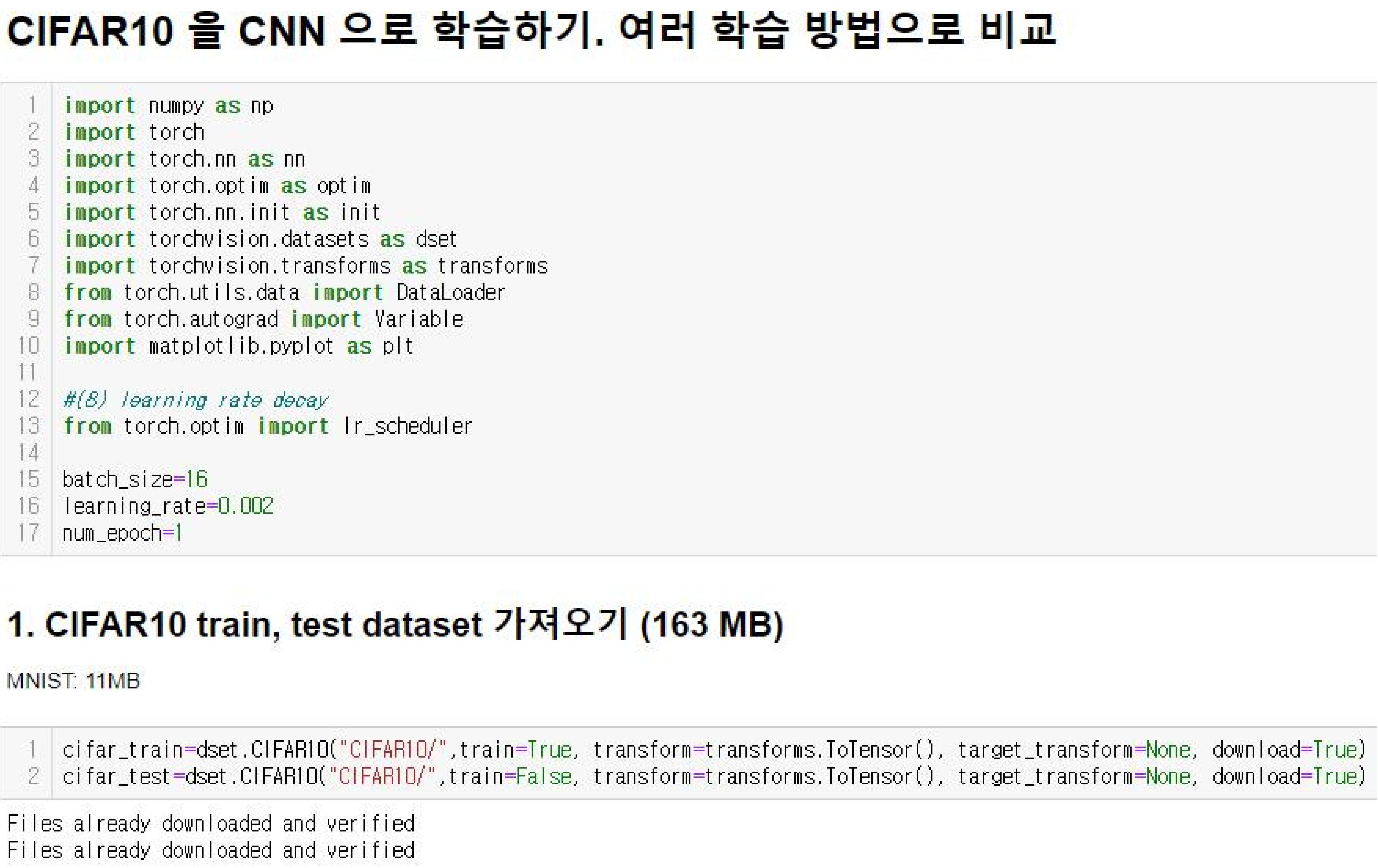
choicenet
Implementation of ChoiceNet
● Jupyter Notebook ★ 122 10

bayes-nn
Lecture notes on Bayesian deep learning
★ 421 105

advanced-tensorflow
Little More Advanced TensorFlow Implementations
● Jupyter Notebook ★ 358 123

Tensorflow-101
TensorFlow Tutorials
● Jupyter Notebook ★ 2.4k 773

dl_tutorials
Deep learning tutorials (2nd ed.)
★ 1.5k 418



# CIFAR10 을 CNN 으로 학습하기. 여러 학습 방법으로 비교

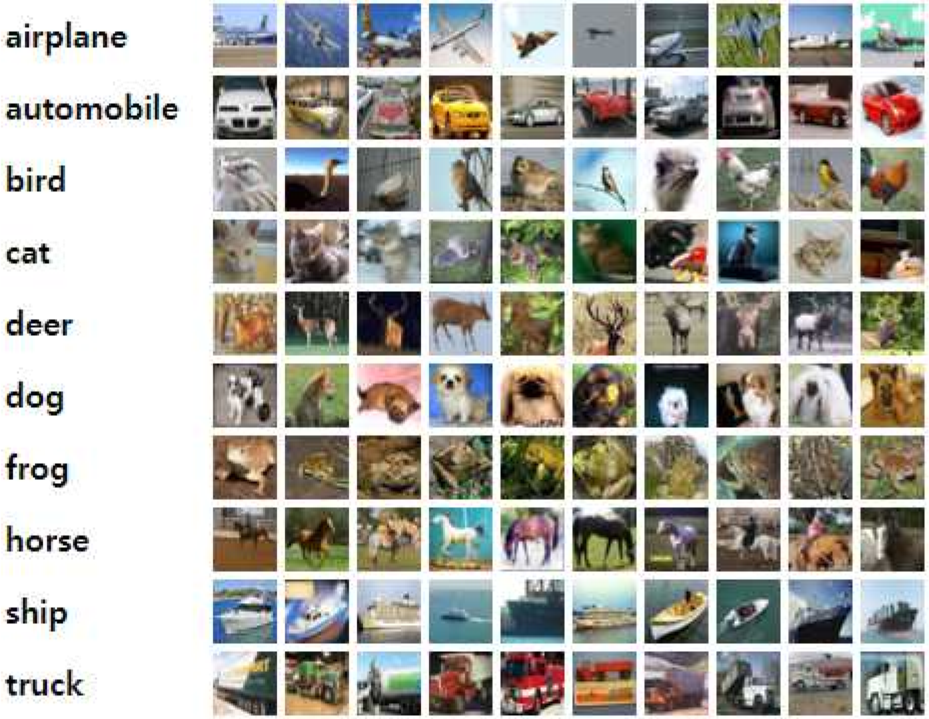
```
import numpy as np
import torch
import torch.nn as nn
import torch.optim as optim
import torch.nn.init as init
import torchvision.datasets as dset
import torchvision.transforms as transforms
from torch.utils.data import DataLoader
from torch.autograd import Variable
import matplotlib.pyplot as plt

#(8) learning rate decay
from torch.optim import lr_scheduler

batch_size=16
learning_rate=0.002
num_epoch=1
```

## 1. CIFAR10 train, test dataset 가져오기 (163 MB)

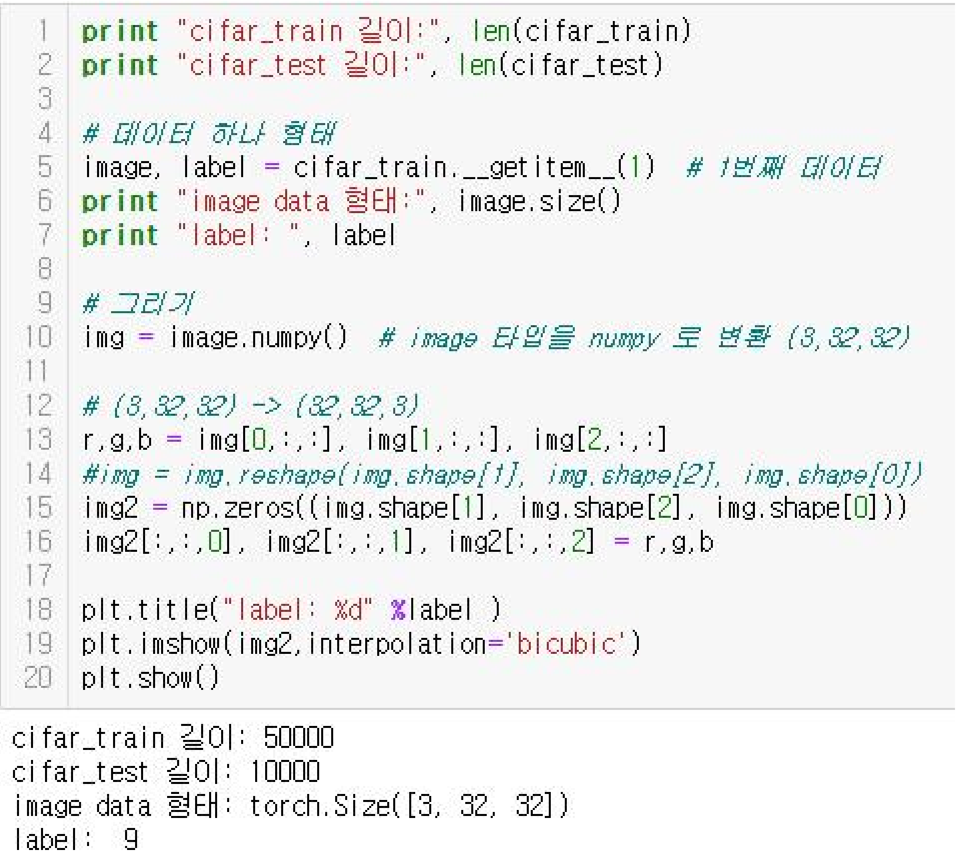
MNIST: 11MB

```
cifar_train=dset.CIFAR10("CIFAR10/",train=True, transform=transforms.ToTensor(), target_transform=None, download=True)
cifar_test=dset.CIFAR10("CIFAR10/",train=False, transform=transforms.ToTensor(), target_transform=None, download=True)
```

```
Files already downloaded and verified
Files already downloaded and verified
```

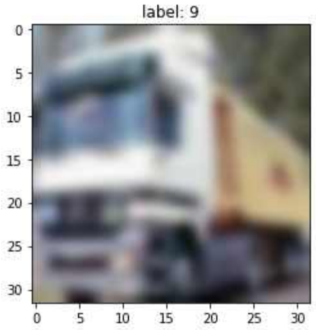

airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck

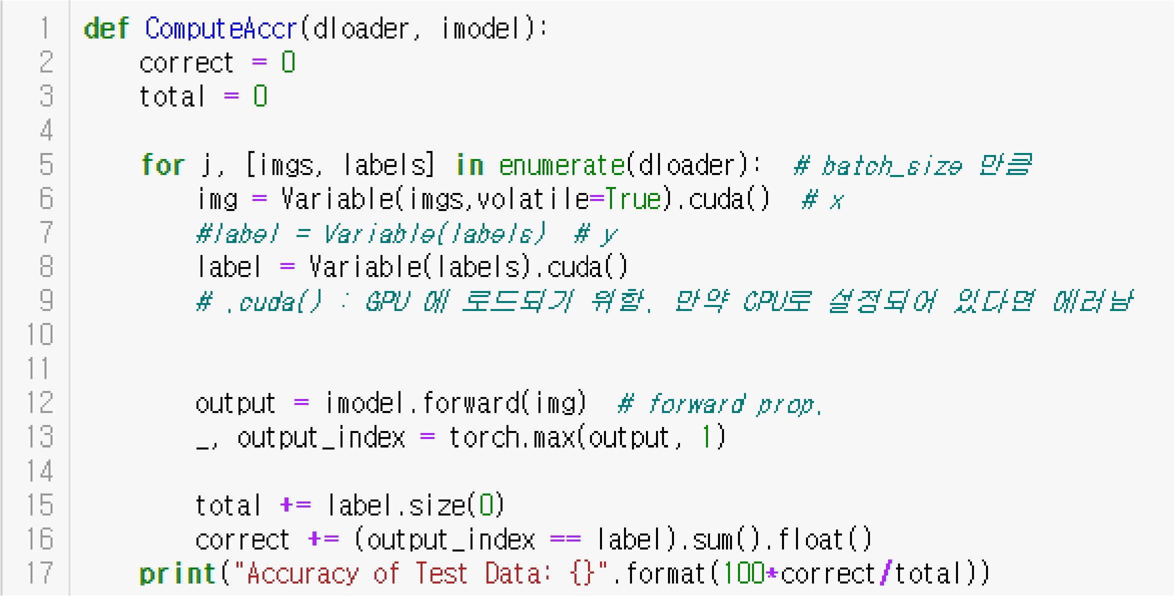
## 2. 대략적인 데이터 형태



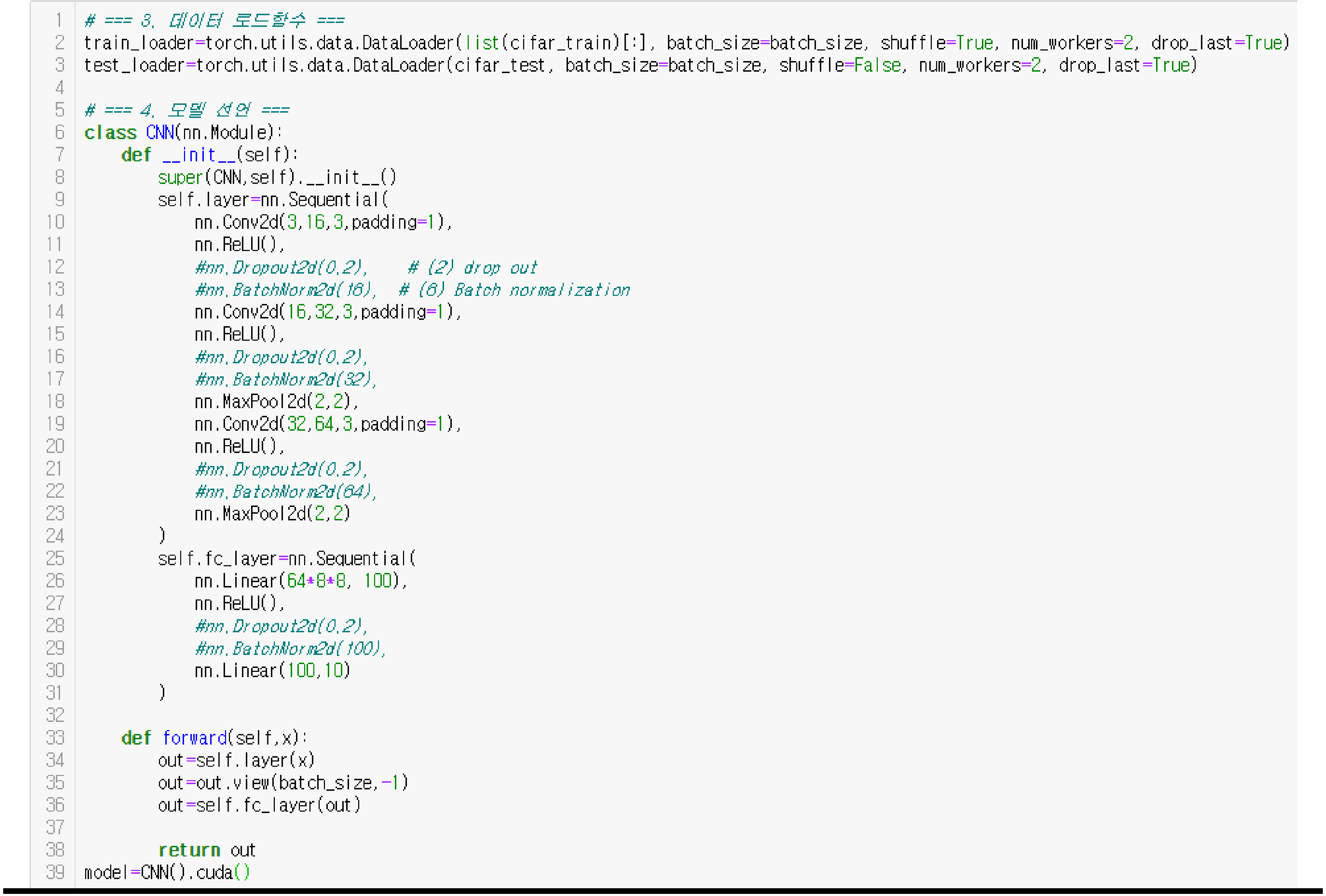
```
print "cifar_train 길이:", len(cifar_train)
print "cifar_test 길이:", len(cifar_test)

# 데이터 하나 형태
image, label = cifar_train.__getitem__(1)  # 1번째 데이터
print "image data 형태:", image.size()
print "label: ", label

# 그리기
img = image.numpy()  # image 타입을 numpy 로 변환 (3,32,32)

# (3,32,32) -> (32,32,3)
r,g,b = img[0,:,:], img[1,:,:], img[2,:,:]
#img = img.reshape(img.shape[1], img.shape[2], img.shape[0])
img2 = np.zeros((img.shape[1], img.shape[2], img.shape[0]))
img2[:,:,0], img2[:,:,1], img2[:,:,2] = r,g,b

plt.title("label: %d" %label )
plt.imshow(img2,interpolation='bicubic')
plt.show()
```

```
cifar_train 길이: 50000
cifar_test 길이: 10000
image data 형태: torch.Size([3, 32, 32])
label:  9
```

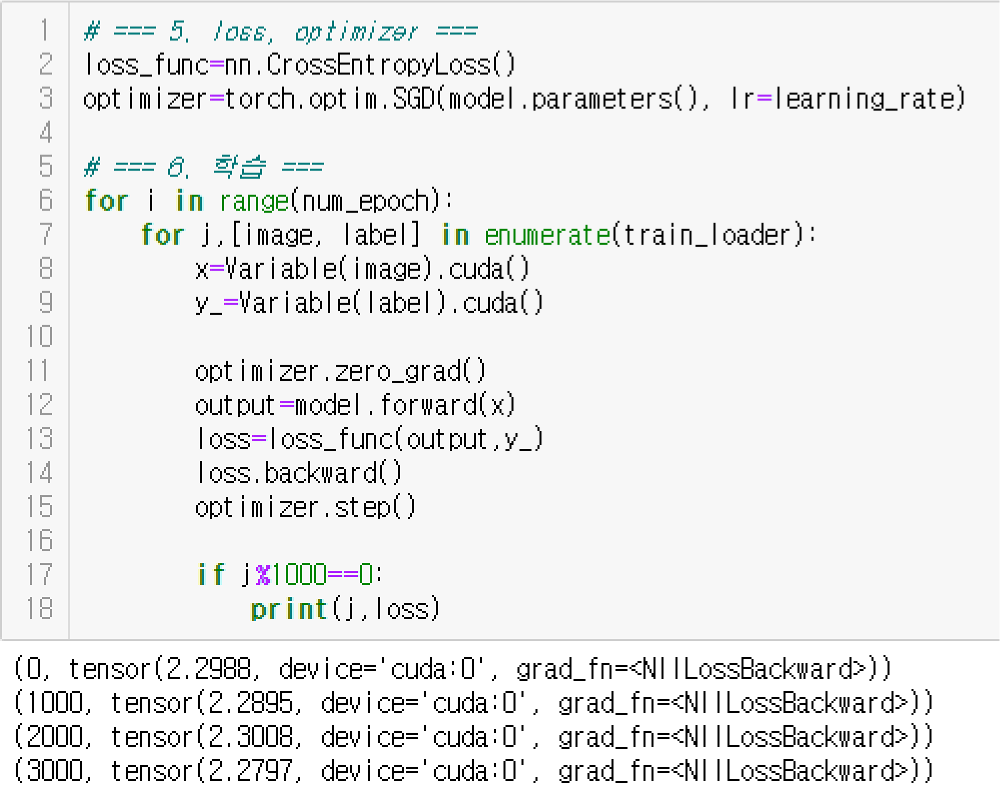




```python
def ComputeAccr(dloader, imodel):
    correct = 0
    total = 0

    for j, [imgs, labels] in enumerate(dloader):  # batch_size 만큼
        img = Variable(imgs,volatile=True).cuda()  # x
        #label = Variable(labels)  # y
        label = Variable(labels).cuda()
        # .cuda() : GPU 에 로드되기 위함. 만약 CPU로 설정되어 있다면 에러남


        output = imodel.forward(img)  # forward prop.
        _, output_index = torch.max(output, 1)

        total += label.size(0)
        correct += (output_index == label).sum().float()
    print("Accuracy of Test Data: {}".format(100*correct/total))
```

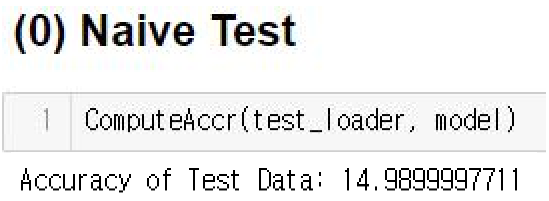


```
# === 3. 데이터 로드함수 ===
train_loader=torch.utils.data.DataLoader(list(cifar_train)[:], batch_size=batch_size, shuffle=True, num_workers=2, drop_last=True)
test_loader=torch.utils.data.DataLoader(cifar_test, batch_size=batch_size, shuffle=False, num_workers=2, drop_last=True)

# === 4. 모델 선언 ===
class CNN(nn.Module):
    def __init__(self):
        super(CNN,self).__init__()
        self.layer=nn.Sequential(
            nn.Conv2d(3,16,3,padding=1),
            nn.ReLU(),
            #nn.Dropout2d(0.2),     # (2) drop out
            #nn.BatchNorm2d(16),  # (6) Batch normalization
            nn.Conv2d(16,32,3,padding=1),
            nn.ReLU(),
            #nn.Dropout2d(0.2),
            #nn.BatchNorm2d(32),
            nn.MaxPool2d(2,2),
            nn.Conv2d(32,64,3,padding=1),
            nn.ReLU(),
            #nn.Dropout2d(0.2),
            #nn.BatchNorm2d(64),
            nn.MaxPool2d(2,2)
        )
        self.fc_layer=nn.Sequential(
            nn.Linear(64*8*8, 100),
            nn.ReLU(),
            #nn.Dropout2d(0.2),
            #nn.BatchNorm2d(100),
            nn.Linear(100,10)
        )

    def forward(self,x):
        out=self.layer(x)
        out=out.view(batch_size,-1)
        out=self.fc_layer(out)

        return out
model=CNN().cuda()
```



```
# === 5. loss, optimizer ===
loss_func=nn.CrossEntropyLoss()
optimizer=torch.optim.SGD(model.parameters(), lr=learning_rate)

# === 6. 학습 ===
for i in range(num_epoch):
    for j,[image, label] in enumerate(train_loader):
        x=Variable(image).cuda()
        y_=Variable(label).cuda()

        optimizer.zero_grad()
        output=model.forward(x)
        loss=loss_func(output,y_)
        loss.backward()
        optimizer.step()

        if j%1000==0:
            print(j,loss)
```

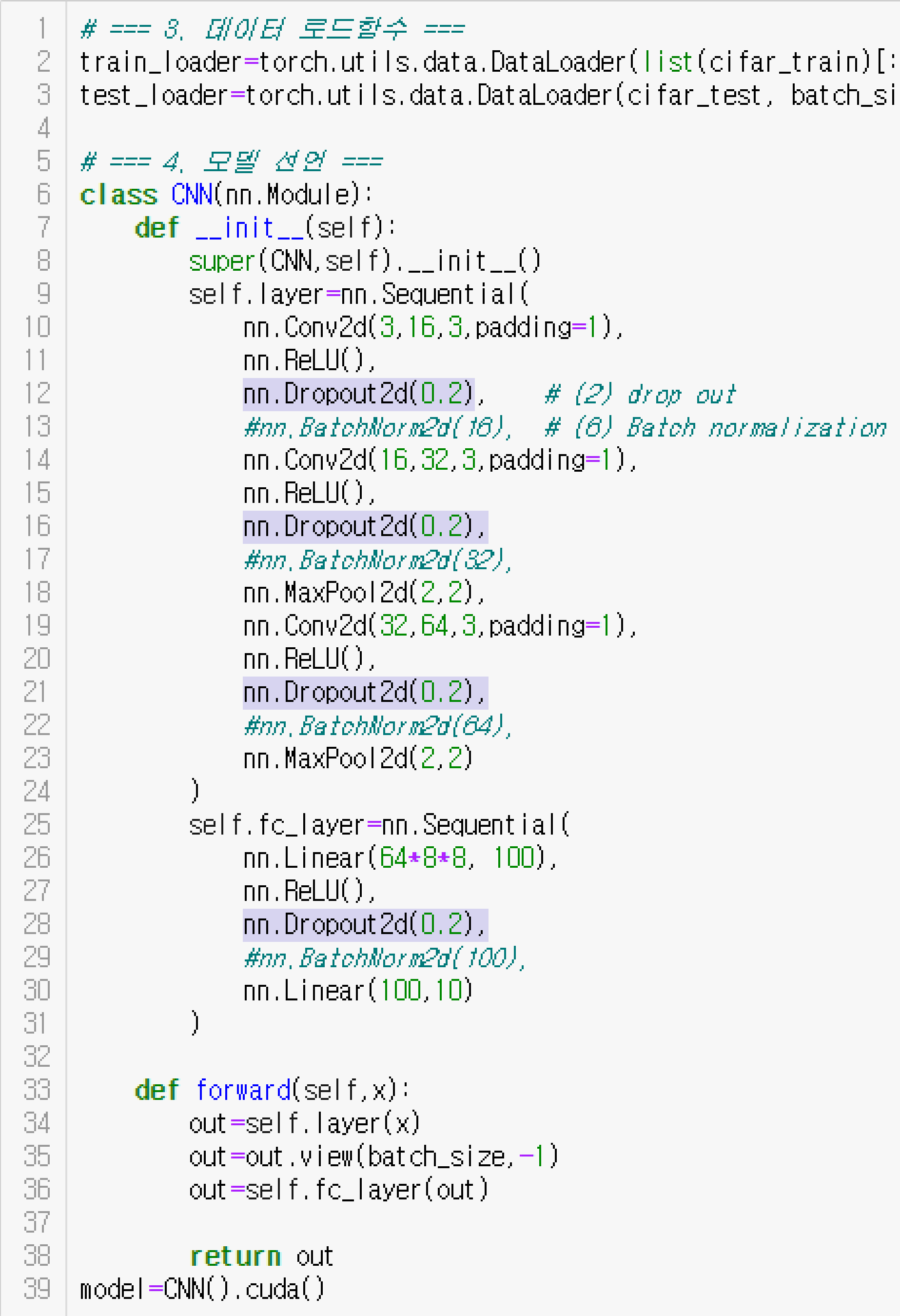
```
(0, tensor(2.2988, device='cuda:0', grad_fn=<NllLossBackward>))
(1000, tensor(2.2895, device='cuda:0', grad_fn=<NllLossBackward>))
(2000, tensor(2.3008, device='cuda:0', grad_fn=<NllLossBackward>))
(3000, tensor(2.2797, device='cuda:0', grad_fn=<NllLossBackward>))
```



## (0) Naive Test

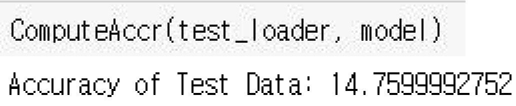
```
ComputeAccr(test_loader, model)
```

```
Accuracy of Test Data: 14.9899997711
```



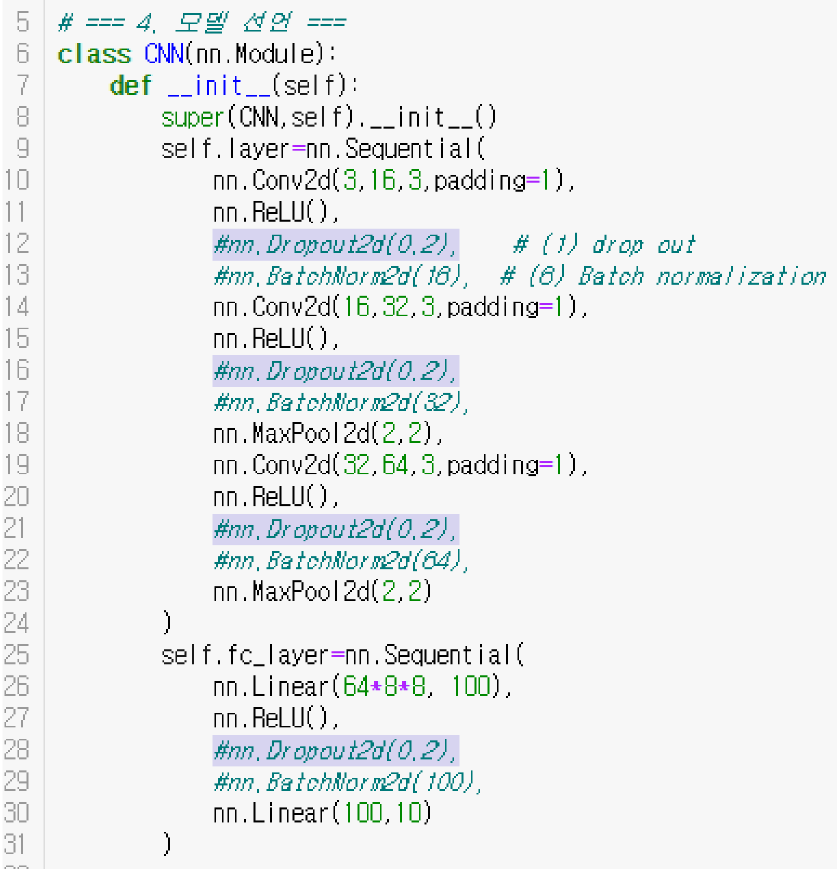
```
# === 3. 데이터 로드함수 ===
train_loader=torch.utils.data.DataLoader(list(cifar_train)[:
test_loader=torch.utils.data.DataLoader(cifar_test, batch_si

# === 4. 모델 선언 ===
class CNN(nn.Module):
    def __init__(self):
        super(CNN,self).__init__()
        self.layer=nn.Sequential(
            nn.Conv2d(3,16,3,padding=1),
            nn.ReLU(),
            nn.Dropout2d(0.2),     # (2) drop out
            #nn.BatchNorm2d(16),   # (6) Batch normalization
            nn.Conv2d(16,32,3,padding=1),
            nn.ReLU(),
            nn.Dropout2d(0.2),
            #nn.BatchNorm2d(32),
            nn.MaxPool2d(2,2),
            nn.Conv2d(32,64,3,padding=1),
            nn.ReLU(),
            nn.Dropout2d(0.2),
            #nn.BatchNorm2d(64),
            nn.MaxPool2d(2,2)
        )
        self.fc_layer=nn.Sequential(
            nn.Linear(64*8*8, 100),
            nn.ReLU(),
            nn.Dropout2d(0.2),
            #nn.BatchNorm2d(100),
            nn.Linear(100,10)
        )

    def forward(self,x):
        out=self.layer(x)
        out=out.view(batch_size,-1)
        out=self.fc_layer(out)

        return out
model=CNN().cuda()
```

## (1) drop out



```
ComputeAccr(test_loader, model)
Accuracy of Test Data: 14.7599992752
```

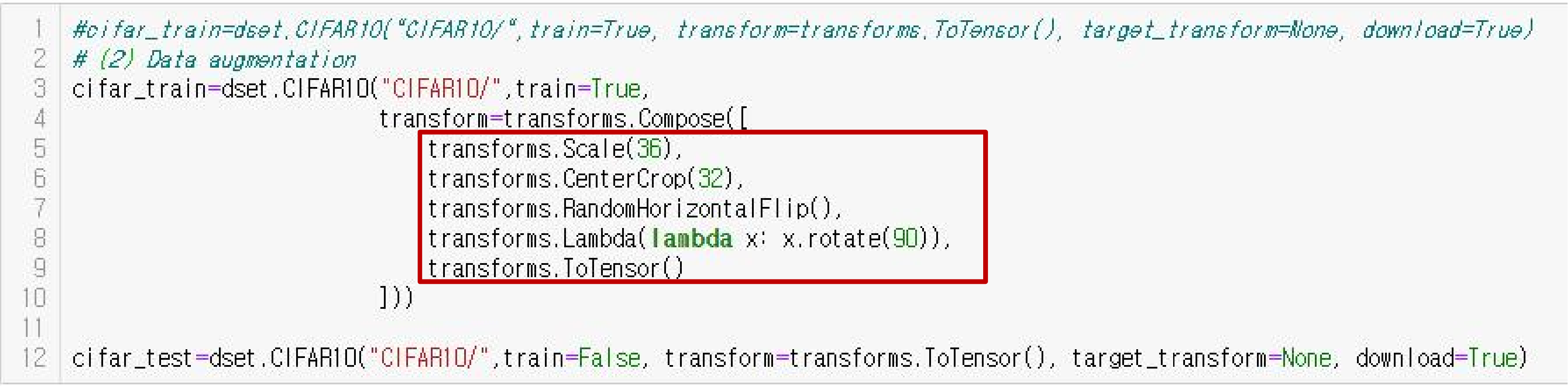
- (1) 에 바뀐 것 원상 복귀할 것
- (2)~마지막 까지 같은 방식으로 원상 복귀 후 실험 권장



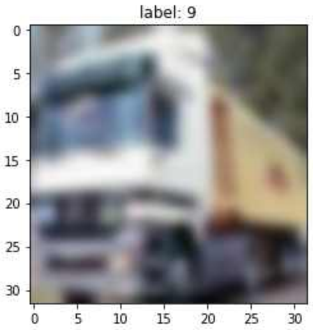
```
# === 4. 모델 선언 ===
class CNN(nn.Module):
    def __init__(self):
        super(CNN,self).__init__()
        self.layer=nn.Sequential(
            nn.Conv2d(3,16,3,padding=1),
            nn.ReLU(),
            #nn.Dropout2d(0.2),     # (1) drop out
            #nn.BatchNorm2d(16),  # (6) Batch normalization
            nn.Conv2d(16,32,3,padding=1),
            nn.ReLU(),
            #nn.Dropout2d(0.2),
            #nn.BatchNorm2d(32),
            nn.MaxPool2d(2,2),
            nn.Conv2d(32,64,3,padding=1),
            nn.ReLU(),
            #nn.Dropout2d(0.2),
            #nn.BatchNorm2d(64),
            nn.MaxPool2d(2,2)
        )
        self.fc_layer=nn.Sequential(
            nn.Linear(64*8*8, 100),
            nn.ReLU(),
            #nn.Dropout2d(0.2),
            #nn.BatchNorm2d(100),
            nn.Linear(100,10)
        )
```

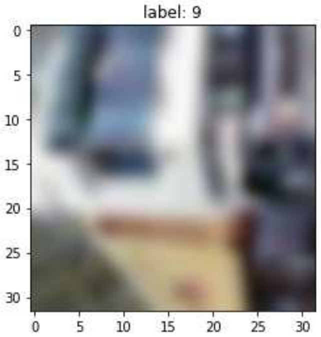
# 1. CIFAR10 train, test dataset 가져오기 (163 MB)

MNIST: 11MB

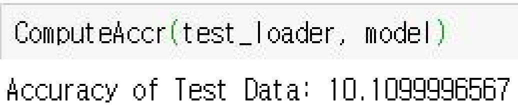


```
#cifar_train=dset.CIFAR10("CIFAR10/",train=True, transform=transforms.ToTensor(), target_transform=None, download=True)
# (2) Data augmentation
cifar_train=dset.CIFAR10("CIFAR10/",train=True,
                         transform=transforms.Compose([
                             transforms.Scale(36),
                             transforms.CenterCrop(32),
                             transforms.RandomHorizontalFlip(),
                             transforms.Lambda(lambda x: x.rotate(90)),
                             transforms.ToTensor()
                         ]))

cifar_test=dset.CIFAR10("CIFAR10/",train=False, transform=transforms.ToTensor(), target_transform=None, download=True)
```



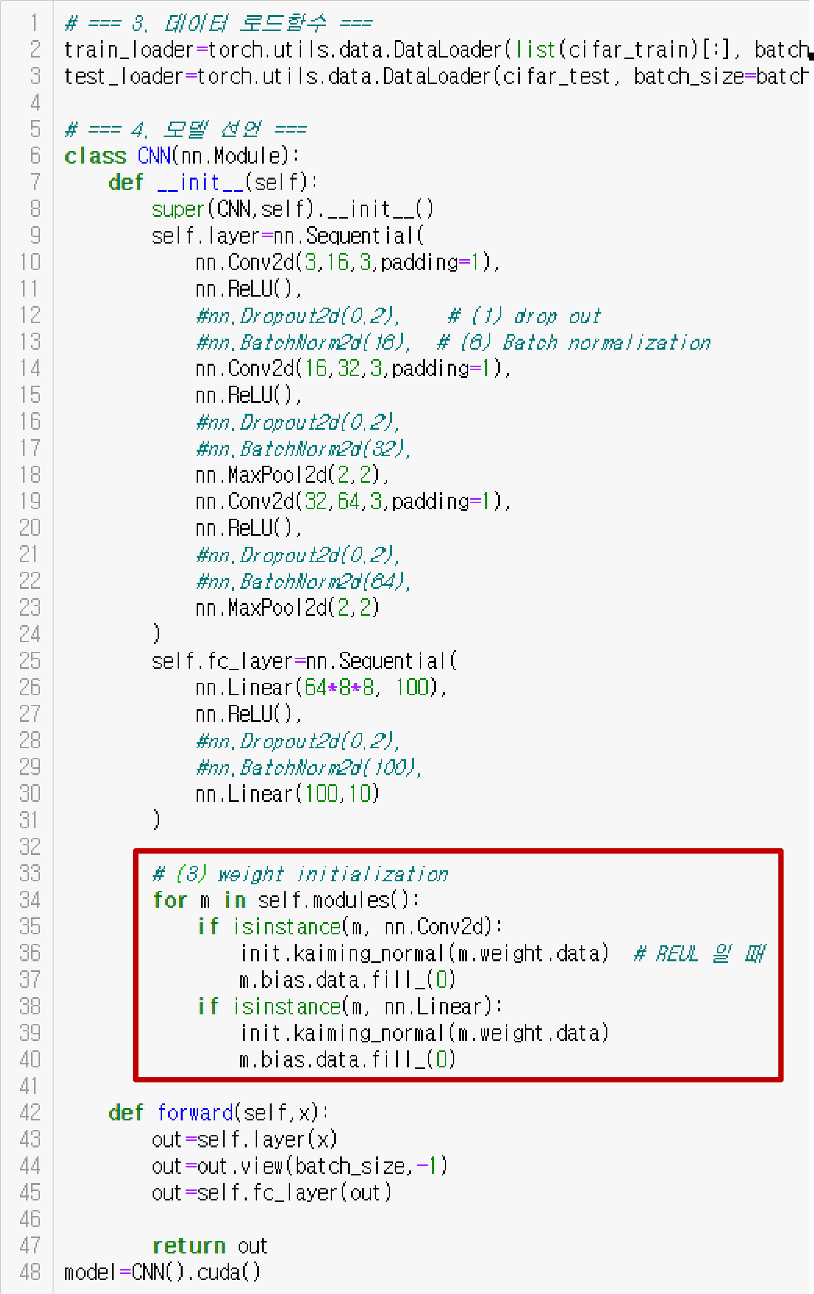


## (2) Data augmentation

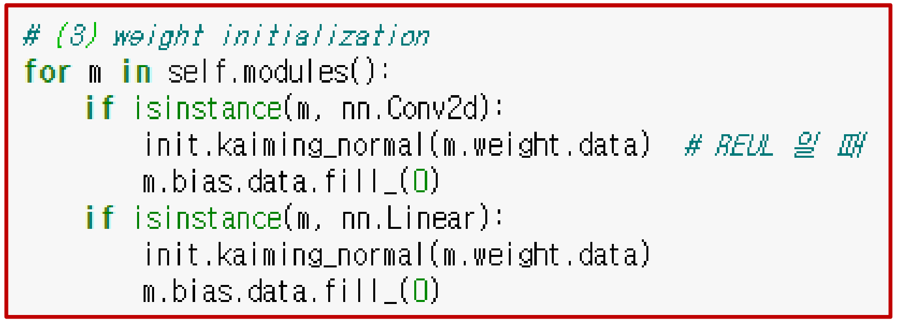


```
ComputeAccr(test_loader, model)
```

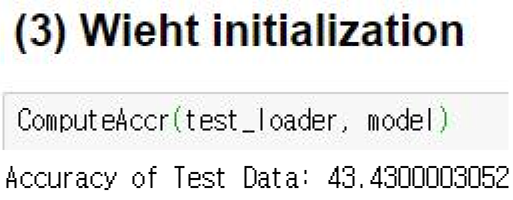
```
Accuracy of Test Data: 10.1099996567
```



```
# === 3. 데이터 로드함수 ===
train_loader=torch.utils.data.DataLoader(list(cifar_train)[:], batch
test_loader=torch.utils.data.DataLoader(cifar_test, batch_size=batch

# === 4. 모델 선언 ===
class CNN(nn.Module):
    def __init__(self):
        super(CNN,self).__init__()
        self.layer=nn.Sequential(
            nn.Conv2d(3,16,3,padding=1),
            nn.ReLU(),
            #nn.Dropout2d(0.2),    # (1) drop out
            #nn.BatchNorm2d(16),   # (6) Batch normalization
            nn.Conv2d(16,32,3,padding=1),
            nn.ReLU(),
            #nn.Dropout2d(0.2),
            #nn.BatchNorm2d(32),
            nn.MaxPool2d(2,2),
            nn.Conv2d(32,64,3,padding=1),
            nn.ReLU(),
            #nn.Dropout2d(0.2),
            #nn.BatchNorm2d(64),
            nn.MaxPool2d(2,2)
        )
        self.fc_layer=nn.Sequential(
            nn.Linear(64*8*8, 100),
            nn.ReLU(),
            #nn.Dropout2d(0.2),
            #nn.BatchNorm2d(100),
            nn.Linear(100,10)
        )

        # (3) weight initialization
        for m in self.modules():
            if isinstance(m, nn.Conv2d):
                init.kaiming_normal(m.weight.data)  # REUL 일 때
                m.bias.data.fill_(0)
            if isinstance(m, nn.Linear):
                init.kaiming_normal(m.weight.data)
                m.bias.data.fill_(0)

    def forward(self,x):
        out=self.layer(x)
        out=out.view(batch_size,-1)
        out=self.fc_layer(out)

        return out
model=CNN().cuda()
```



```
# (3) weight initialization
for m in self.modules():
    if isinstance(m, nn.Conv2d):
        init.kaiming_normal(m.weight.data)  # REUL 일 때
        m.bias.data.fill_(0)
    if isinstance(m, nn.Linear):
        init.kaiming_normal(m.weight.data)
        m.bias.data.fill_(0)
```



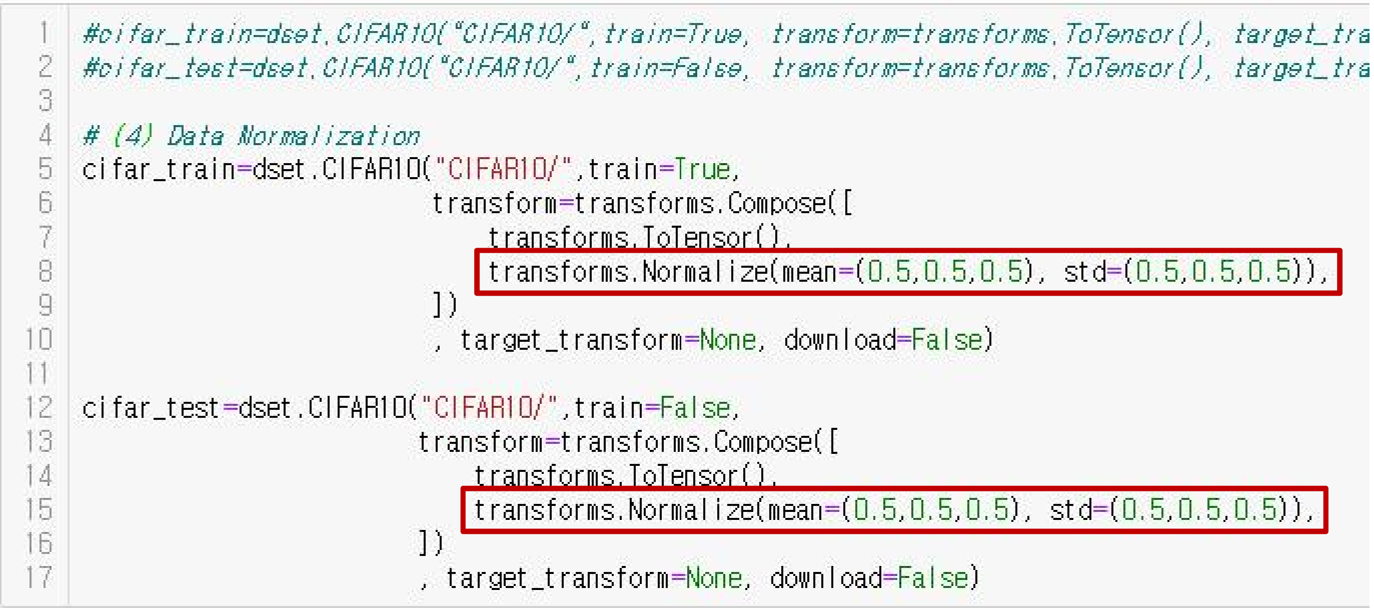
## (3) Wieht initialization

```
ComputeAccr(test_loader, model)
```

```
Accuracy of Test Data: 43.4300003052
```

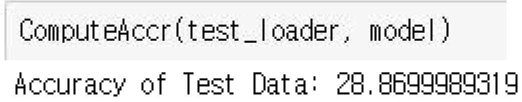
# 1. CIFAR10 train, test dataset 가져오기 (163 MB)

MNIST: 11MB



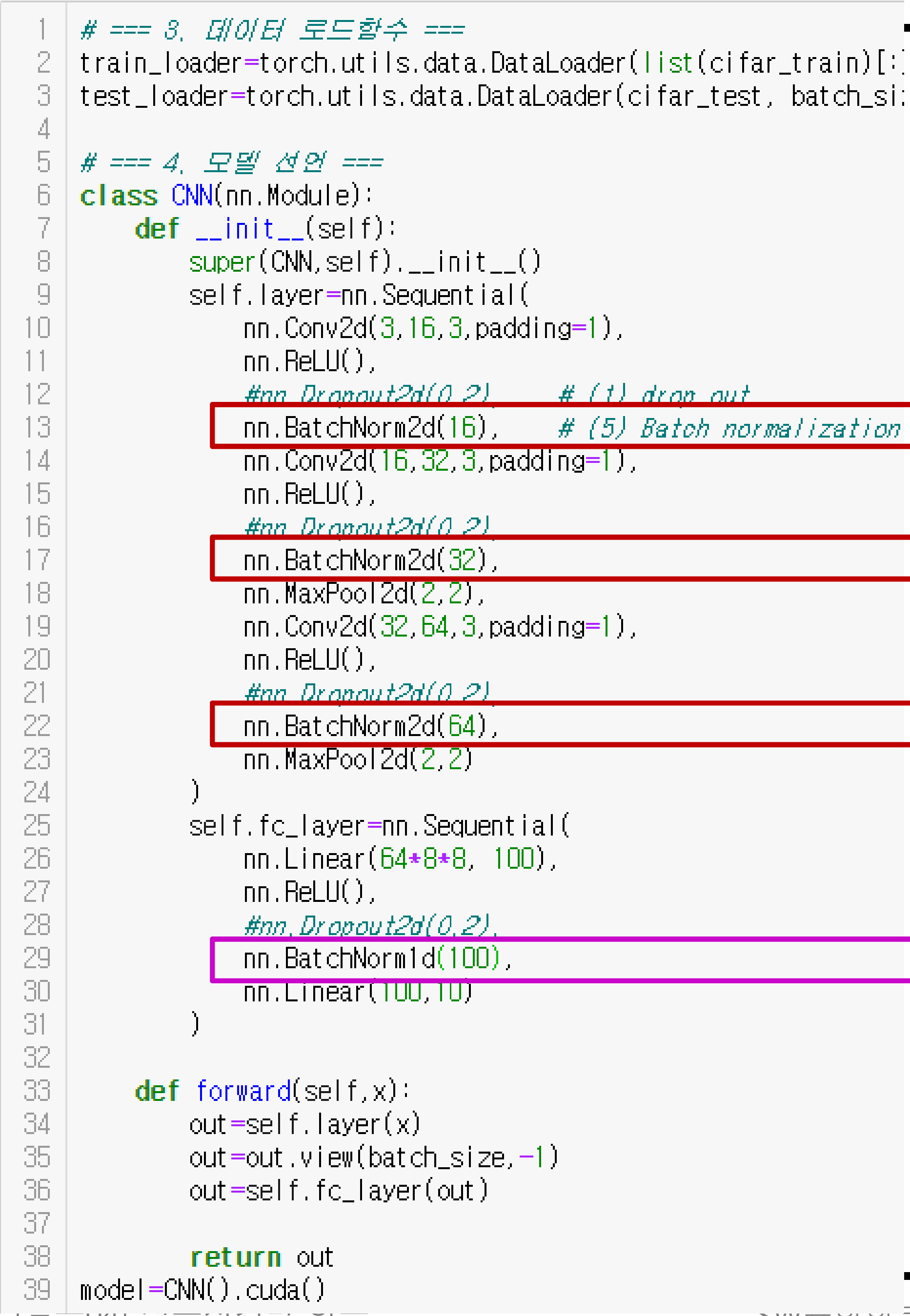
```
#cifar_train=dset.CIFAR10("CIFAR10/",train=True, transform=transforms.ToTensor(), target_tra
#cifar_test=dset.CIFAR10("CIFAR10/",train=False, transform=transforms.ToTensor(), target_tra

# (4) Data Normalization
cifar_train=dset.CIFAR10("CIFAR10/",train=True,
                         transform=transforms.Compose([
                             transforms.ToTensor(),
                             transforms.Normalize(mean=(0.5,0.5,0.5), std=(0.5,0.5,0.5)),
                         ])
                         , target_transform=None, download=False)

cifar_test=dset.CIFAR10("CIFAR10/",train=False,
                        transform=transforms.Compose([
                            transforms.ToTensor(),
                            transforms.Normalize(mean=(0.5,0.5,0.5), std=(0.5,0.5,0.5)),
                        ])
                        , target_transform=None, download=False)
```

## (4) Data Normalization



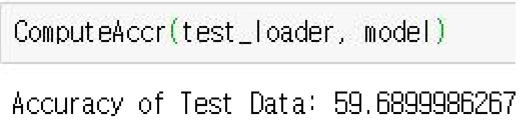
```
ComputeAccr(test_loader, model)
```

```
Accuracy of Test Data: 28.8699989319
```

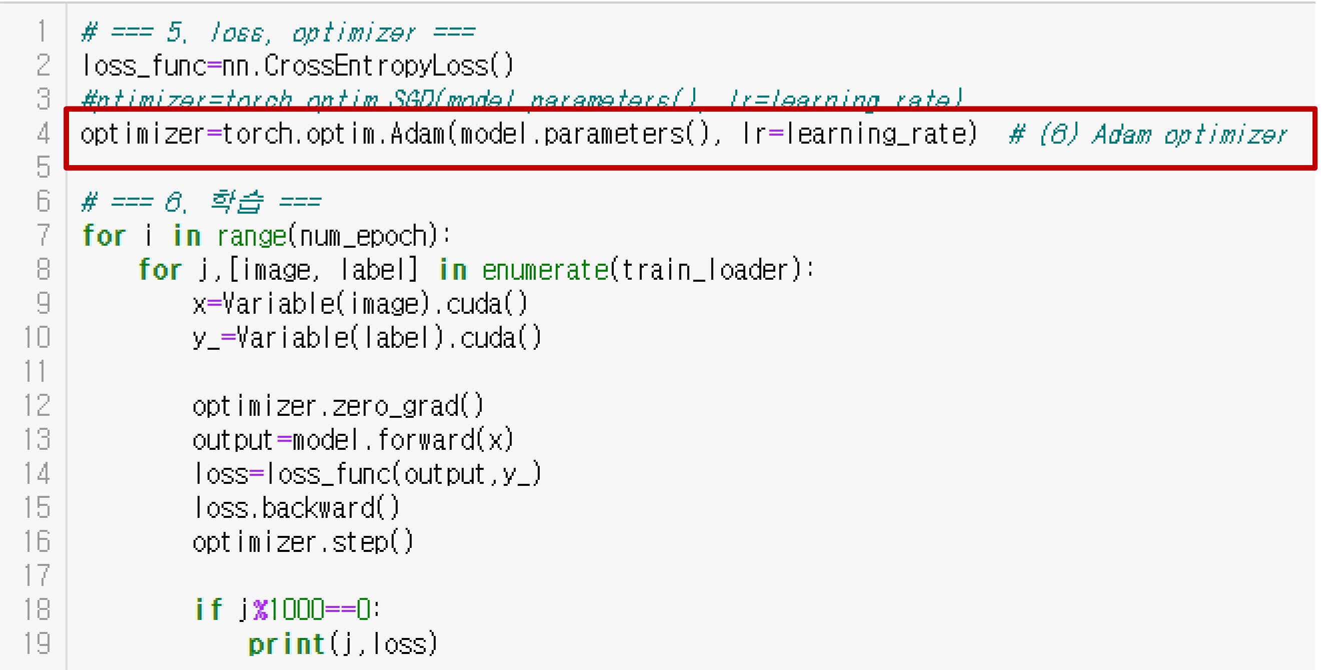


```
# === 3. 데이터 로드함수 ===
train_loader=torch.utils.data.DataLoader(list(cifar_train)[:
test_loader=torch.utils.data.DataLoader(cifar_test, batch_si

# === 4. 모델 선언 ===
class CNN(nn.Module):
    def __init__(self):
        super(CNN,self).__init__()
        self.layer=nn.Sequential(
            nn.Conv2d(3,16,3,padding=1),
            nn.ReLU(),
            #nn.Dropout2d(0.2)      # (1) drop out
            nn.BatchNorm2d(16),     # (5) Batch normalization
            nn.Conv2d(16,32,3,padding=1),
            nn.ReLU(),
            #nn.Dropout2d(0.2)
            nn.BatchNorm2d(32),
            nn.MaxPool2d(2,2),
            nn.Conv2d(32,64,3,padding=1),
            nn.ReLU(),
            #nn.Dropout2d(0.2)
            nn.BatchNorm2d(64),
            nn.MaxPool2d(2,2)
        )
        self.fc_layer=nn.Sequential(
            nn.Linear(64*8*8, 100),
            nn.ReLU(),
            #nn.Dropout2d(0.2),
            nn.BatchNorm1d(100),
            nn.Linear(100,10)
        )

    def forward(self,x):
        out=self.layer(x)
        out=out.view(batch_size,-1)
        out=self.fc_layer(out)

        return out
model=CNN().cuda()
```

## (5) Batch normalization

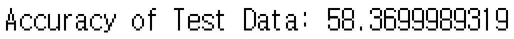


```
ComputeAccr(test_loader, model)

Accuracy of Test Data: 59.6899986267
```

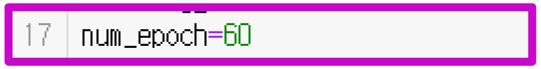


```
# === 5. loss, optimizer ===
loss_func=nn.CrossEntropyLoss()
#optimizer=torch.optim.SGD(model.parameters(), lr=learning_rate)
optimizer=torch.optim.Adam(model.parameters(), lr=learning_rate)  # (6) Adam optimizer

# === 6. 학습 ===
for i in range(num_epoch):
    for j,[image, label] in enumerate(train_loader):
        x=Variable(image).cuda()
        y_=Variable(label).cuda()

        optimizer.zero_grad()
        output=model.forward(x)
        loss=loss_func(output,y_)
        loss.backward()
        optimizer.step()

        if j%1000==0:
            print(j,loss)
```

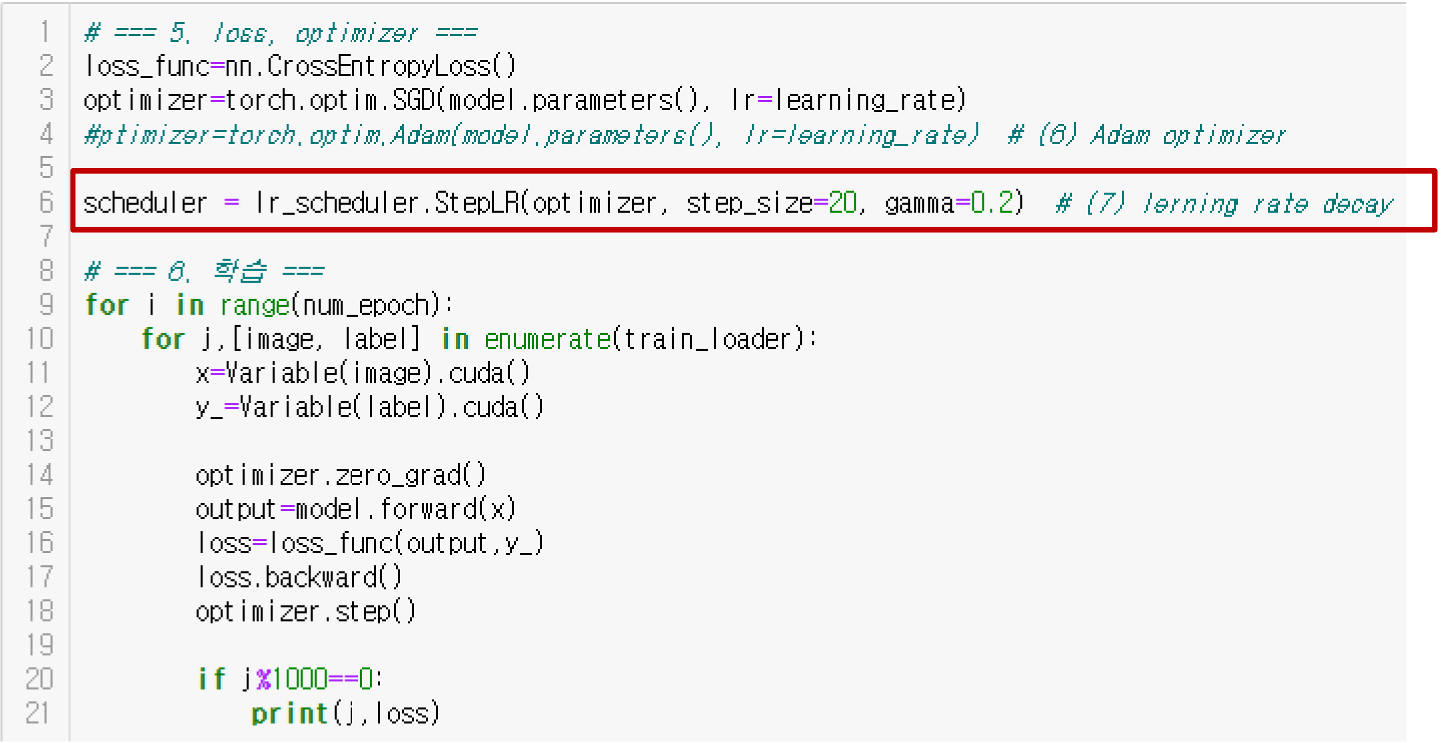
## (6) Adam optimizer



```
Accuracy of Test Data: 58.3699989319
```

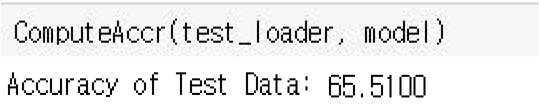


```
num_epoch=60
```



```
# === 5. loss, optimizer ===
loss_func=nn.CrossEntropyLoss()
optimizer=torch.optim.SGD(model.parameters(), lr=learning_rate)
#ptimizer=torch.optim.Adam(model.parameters(), lr=learning_rate)  # (6) Adam optimizer

scheduler = lr_scheduler.StepLR(optimizer, step_size=20, gamma=0.2)  # (7) lerning rate decay

# === 6. 학습 ===
for i in range(num_epoch):
    for j,[image, label] in enumerate(train_loader):
        x=Variable(image).cuda()
        y_=Variable(label).cuda()

        optimizer.zero_grad()
        output=model.forward(x)
        loss=loss_func(output,y_)
        loss.backward()
        optimizer.step()

        if j%1000==0:
            print(j,loss)
```

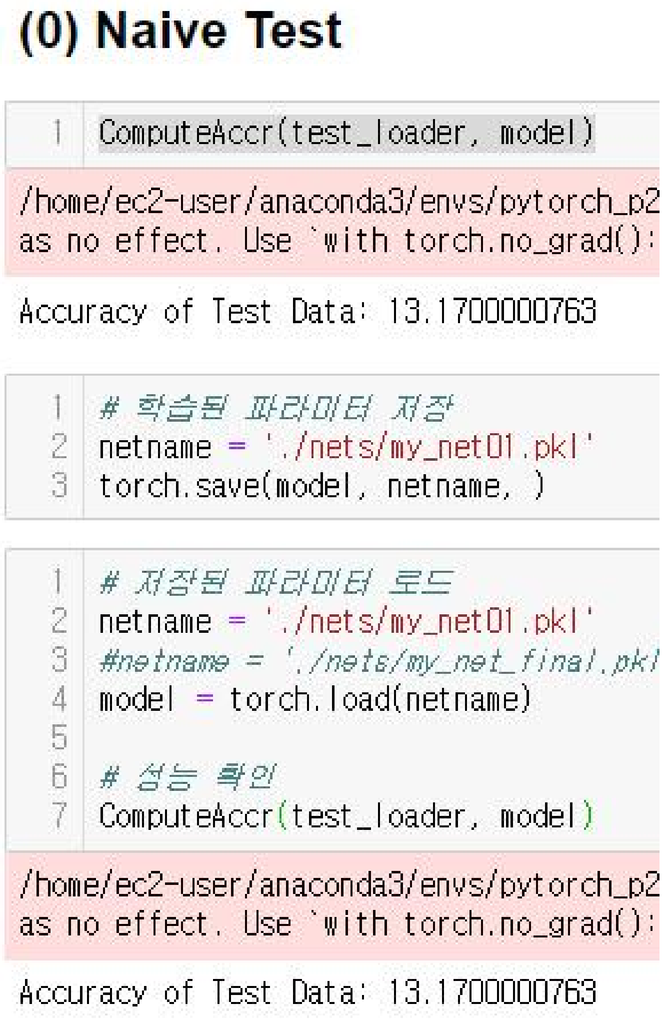
## (7) learning rate decay



```
ComputeAccr(test_loader, model)
Accuracy of Test Data: 65.5100
```

# 성능 측정 전/후 저장 & 로드 후 테스트

- 모델 파라미터 저장
  - 성능 측정 전 or 후에 파라미터 저장
- 저장된 모델 로드 후 성능 확인



(0) Naive Test

```
ComputeAccr(test_loader, model)
```

```
/home/ec2-user/anaconda3/envs/pytorch_p2
as no effect. Use `with torch.no_grad():
```

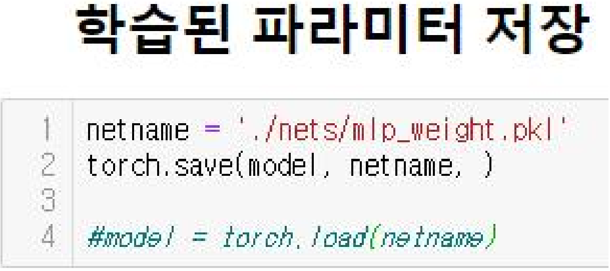
```
Accuracy of Test Data: 13.1700000763
```

```
# 학습된 파라미터 저장
netname = './nets/my_net01.pkl'
torch.save(model, netname, )
```

```
# 저장된 파라미터 로드
netname = './nets/my_net01.pkl'
#netname = './nets/my_net_final.pkl
model = torch.load(netname)

# 성능 확인
ComputeAccr(test_loader, model)
```

```
/home/ec2-user/anaconda3/envs/pytorch_p2
as no effect. Use `with torch.no_grad():
```

```
Accuracy of Test Data: 13.1700000763
```



학습된 파라미터 저장

```
netname = './nets/mlp_weight.pkl'
torch.save(model, netname, )

#model = torch.load(netname)
```