

# Data Science(Machine Learning, Deep Learning)

- 실습 소스 깃 참고

<실전 머신러닝 활용 및 데이터 수집>

train.py 구현 및 실행

<train.py 소스 코드>

```
import argparse

import torch
import torch.nn as nn
import torch.optim as optim

from model import ImageClassifier
from trainer import Trainer

from utils import load_mnist
from utils import split_data
from utils import get_hidden_sizes

def define_argparser():
    p = argparse.ArgumentParser()

    p.add_argument('--model_fn', required=True)
    p.add_argument('--gpu_id', type=int, default=0 if torch.cuda.is_available() else -1)

    p.add_argument('--train_ratio', type=float, default=.8)

    p.add_argument('--batch_size', type=int, default=256)
    p.add_argument('--n_epochs', type=int, default=20)

    p.add_argument('--n_layers', type=int, default=5)
    p.add_argument('--use_dropout', action='store_true')
    p.add_argument('--dropout_p', type=float, default=.3)

    p.add_argument('--verbose', type=int, default=1)

    config = p.parse_args()

    return config

def main(config):
    # Set device based on user defined configuration.
    device = torch.device('cpu') if config.gpu_id < 0 else torch.device('cuda:%d' %
config.gpu_id)
```

```

x, y = load_mnist(is_train=True, flatten=True)
x, y = split_data(x.to(device), y.to(device), train_ratio=config.train_ratio)

print("Train:", x[0].shape, y[0].shape)
print("Valid:", x[1].shape, y[1].shape)

input_size = int(x[0].shape[-1])
output_size = int(max(y[0])) + 1

model = ImageClassifier(
    input_size=input_size,
    output_size=output_size,
    hidden_sizes=get_hidden_sizes(input_size,
                                   output_size,
                                   config.n_layers),
    use_batch_norm=not config.use_dropout,
    dropout_p=config.dropout_p,
).to(device)
optimizer = optim.Adam(model.parameters())
crit = nn.NLLLoss()

if config.verbose >= 1:
    print(model)
    print(optimizer)
    print(crit)

trainer = Trainer(model, optimizer, crit)

trainer.train(
    train_data=(x[0], y[0]),
    valid_data=(x[1], y[1]),
    config=config
)

# Save best model weights.
torch.save({
    'model': trainer.model.state_dict(),
    'opt': optimizer.state_dict(),
    'config': config,
}, config.model_fn)

if __name__ == '__main__':
    config = define_argparser()
    main(config)

```

main이 보통은 숨겨져 있음.

소스 코드를 파악할 때 main부분을 파악하는 것이 중요함.

arg = 인수, 매개변수라고도 함

함수를 호출할 때 전달되는 값을 인수라고 하고, 함수를 정의할 때 선언되는 변수를 매개변수라고 합니다.

arg는 인수와 매개변수의 개념을 모두 포함하는 포괄적인 용어

<train.py 활용>

Anaconda Prompt

<Pytorch 설치>

(base) C:\Users\admin\Desktop\BigData DeepLearning\DeepLearning  
소스코드\실습\15-practical\_exercise>conda install pytorch

The following packages will be downloaded:

package		build	
ca-certificates-2023.08.22		haa95532_0	123 KB
conda-content-trust-0.2.0		py311haa95532_0	82 KB
libuv-1.44.2		h2bfff1b_0	288 KB
ninja-1.10.2		haa95532_5	14 KB
ninja-base-1.10.2		h6d14046_5	255 KB
openssl-1.1.1w		h2bfff1b_0	5.5 MB
pytorch-2.0.1		cpu_py311hd080823_0	91.2 MB
-----			
Total:			97.5 MB

Package 설치

중간에 Proceed ([y]/n)?가 나오면 y입력

(base) C:\Users\admin\Desktop\BigData DeepLearning\DeepLearning  
소스코드\실습\15-practical\_exercise>python train.py

usage: train.py [-h] --model\_fn MODEL\_FN [--gpu\_id GPU\_ID] [--train\_ratio TRAIN\_RATIO]  
[--batch\_size BATCH\_SIZE]

                  [--n\_epochs N\_EPOCHS] [--n\_layers N\_LAYERS] [--use\_dropout]  
[--dropout\_p DROPOUT\_P] [--verbose VERBOSE]

train.py: error: the following arguments are required: --model\_fn

에러가 뜸

```
if __name__ == '__main__':  
    config = define_argparser()  
    main(config)
```

argparser로 매개변수(인수)를 쪼갬

(base) C:\Users\admin\Desktop\BigData DeepLearning\DeepLearning

소스코드\실습\15-practical\_exercise>python train.py --model\_fn tmp.pth --gpu\_id -1  
--batch\_size 256 --n\_epochs 20 --n\_layers 5

아래같이 에러가 뜨는 경우 conda install torchvision으로 TorchVision 설치해 줘야 함

(base) C:\Users\admin\Desktop\BigData DeepLearning\DeepLearning

소스코드\실습\15-practical\_exercise>python train.py --model\_fn tmp.pth --gpu\_id -1  
--batch\_size 256 --n\_epochs 20 --n\_layers 5

Traceback (most recent call last):

```
File "C:\Users\admin\Desktop\BigData DeepLearning\DeepLearning
소스코드\실습\15-practical_exercise\utils.py", line 5, in load_mnist
    from torchvision import datasets, transforms
ModuleNotFoundError: No module named 'torchvision'
```

```

        (0): Linear(in_features=784, out_features=630, bias=True)
        (1): LeakyReLU(negative_slope=0.01)
        (2): BatchNorm1d(630, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
)
(1): Block(
  (block): Sequential(
    (0): Linear(in_features=630, out_features=476, bias=True)
    (1): LeakyReLU(negative_slope=0.01)
    (2): BatchNorm1d(476, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  )
)
(2): Block(
  (block): Sequential(
    (0): Linear(in_features=476, out_features=322, bias=True)
    (1): LeakyReLU(negative_slope=0.01)
    (2): BatchNorm1d(322, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  )
)
(3): Block(
  (block): Sequential(
    (0): Linear(in_features=322, out_features=168, bias=True)
    (1): LeakyReLU(negative_slope=0.01)
    (2): BatchNorm1d(168, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  )
)
(4): Linear(in_features=168, out_features=10, bias=True)
(5): LogSoftmax(dim=-1)
)
)
Adam (
Parameter Group 0
  amsgrad: False
  betas: (0.9, 0.999)
  capturable: False
  differentiable: False
  eps: 1e-08
  foreach: None
  fused: None
  lr: 0.001
  maximize: False
  weight_decay: 0
)
NLLLoss()
Epoch(1/20): train_loss=1.9742e-01  valid_loss=1.0411e-01  lowest_loss=1.0411e-01
Epoch(2/20): train_loss=8.1424e-02  valid_loss=9.0457e-02  lowest_loss=9.0457e-02

```

Epoch(3/20):	train_loss=5.6666e-02	valid_loss=8.7683e-02	lowest_loss=8.7683e-02
Epoch(4/20):	train_loss=4.2112e-02	valid_loss=1.0279e-01	lowest_loss=8.7683e-02
Epoch(5/20):	train_loss=3.5553e-02	valid_loss=8.5411e-02	lowest_loss=8.5411e-02
Epoch(6/20):	train_loss=2.5702e-02	valid_loss=7.4233e-02	lowest_loss=7.4233e-02
Epoch(7/20):	train_loss=2.6314e-02	valid_loss=7.5723e-02	lowest_loss=7.4233e-02
Epoch(8/20):	train_loss=2.0976e-02	valid_loss=8.4059e-02	lowest_loss=7.4233e-02
Epoch(9/20):	train_loss=1.5663e-02	valid_loss=9.4074e-02	lowest_loss=7.4233e-02
Epoch(10/20):	train_loss=1.7997e-02	valid_loss=8.3264e-02	lowest_loss=7.4233e-02
Epoch(11/20):	train_loss=1.5647e-02	valid_loss=8.0157e-02	lowest_loss=7.4233e-02
Epoch(12/20):	train_loss=1.3690e-02	valid_loss=7.8018e-02	lowest_loss=7.4233e-02
Epoch(13/20):	train_loss=1.5982e-02	valid_loss=8.3138e-02	lowest_loss=7.4233e-02
Epoch(14/20):	train_loss=1.2972e-02	valid_loss=8.2686e-02	lowest_loss=7.4233e-02
Epoch(15/20):	train_loss=1.1763e-02	valid_loss=7.6221e-02	lowest_loss=7.4233e-02
Epoch(16/20):	train_loss=1.2345e-02	valid_loss=7.0978e-02	lowest_loss=7.0978e-02
Epoch(17/20):	train_loss=8.5329e-03	valid_loss=7.4744e-02	lowest_loss=7.0978e-02
Epoch(18/20):	train_loss=9.8661e-03	valid_loss=8.3837e-02	lowest_loss=7.0978e-02
Epoch(19/20):	train_loss=7.0544e-03	valid_loss=7.6451e-02	lowest_loss=7.0978e-02
Epoch(20/20):	train_loss=1.0674e-02	valid_loss=7.2226e-02	lowest_loss=7.0978e-02

에러가 안 뜨면 이와 같이 나오면서 tmp.pth가 생성이 됨.

#### <MNIST란?>

MNIST는 "Modified National Institute of Standards and Technology"의 약어로, 손으로 쓴 숫자(0에서 9까지)로 이루어진 대형 데이터 세트  
머신 러닝 및 딥 러닝에서 기본적으로 사용되는 벤치마크 데이터 세트

MNIST 데이터 세트는 컴퓨터 비전에서 이미지 분류 알고리즘을 개발하고 테스트하는 데 사용된다.  
많은 머신 러닝 및 딥 러닝 프레임워크와 라이브러리에서 쉽게 사용할 수 있습니다.  
MNIST 데이터 세트는 비교적 작고 간단하기 때문에, 머신 러닝 및 딥 러닝 모델을 처음 학습하는 데 좋은 출발점이 된다.

#### <predict.ipynb 구현>

Colab에서의 Linux 환경 구현

!ls로 현재 디렉토리 위치 확인

model.py를 content에 끌어다 넣고 !python model.py 실행  
앞서 tmp.pth 만든 것 역시 content에 복사

Jupyter Notebook에서는 환경이 다를 수 있음.