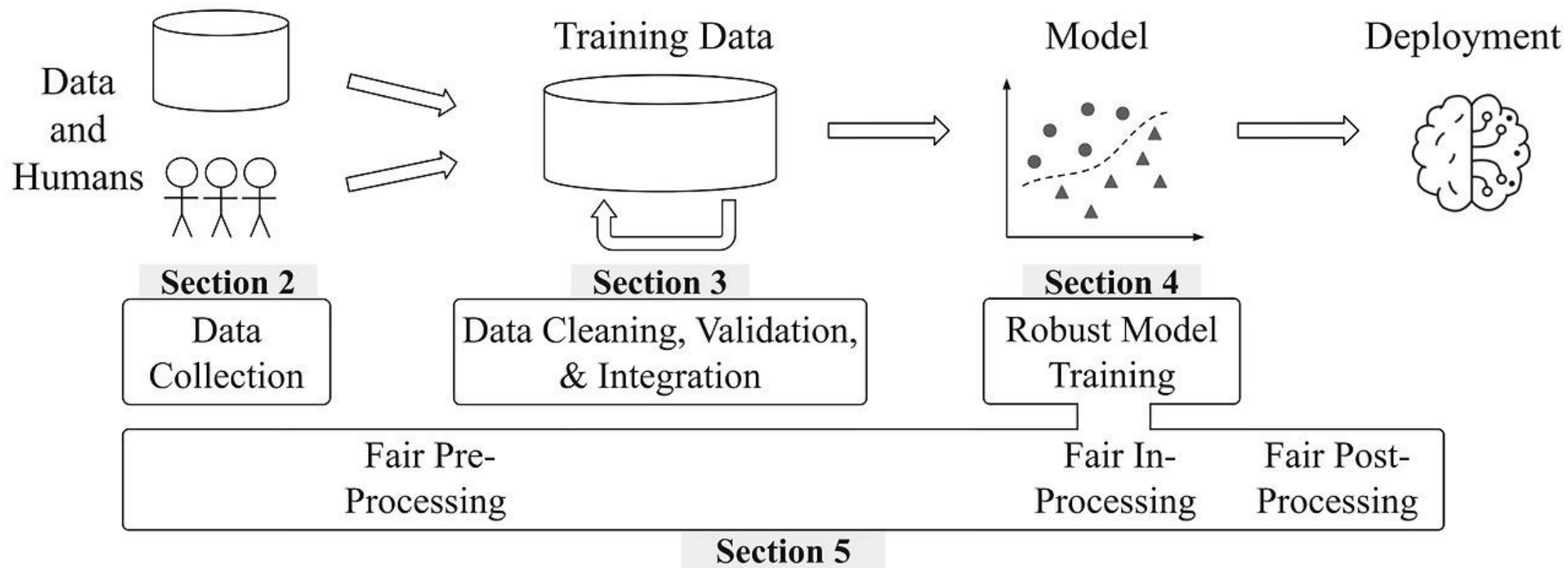


AI 실전 2주차

나만의 모델 설계 과정 및 진행

Feat. PyTorch

모델 제작 과정



Pytorch install

Start Locally

Shortcuts

Prerequisites

Supported Windows

Distributions

Python

Package Manager

Installation

pip

Verification

Building from source

Prerequisites

Select your preferences and run the install command. Stable represents the most currently tested and supported version of PyTorch. This should be suitable for many users. Preview is available if you want the latest, not fully tested and supported, builds that are generated nightly. Please ensure that you have **met the prerequisites below (e.g., numpy)**, depending on your package manager. You can also **install previous versions of PyTorch**. Note that LibTorch is only available for C++.

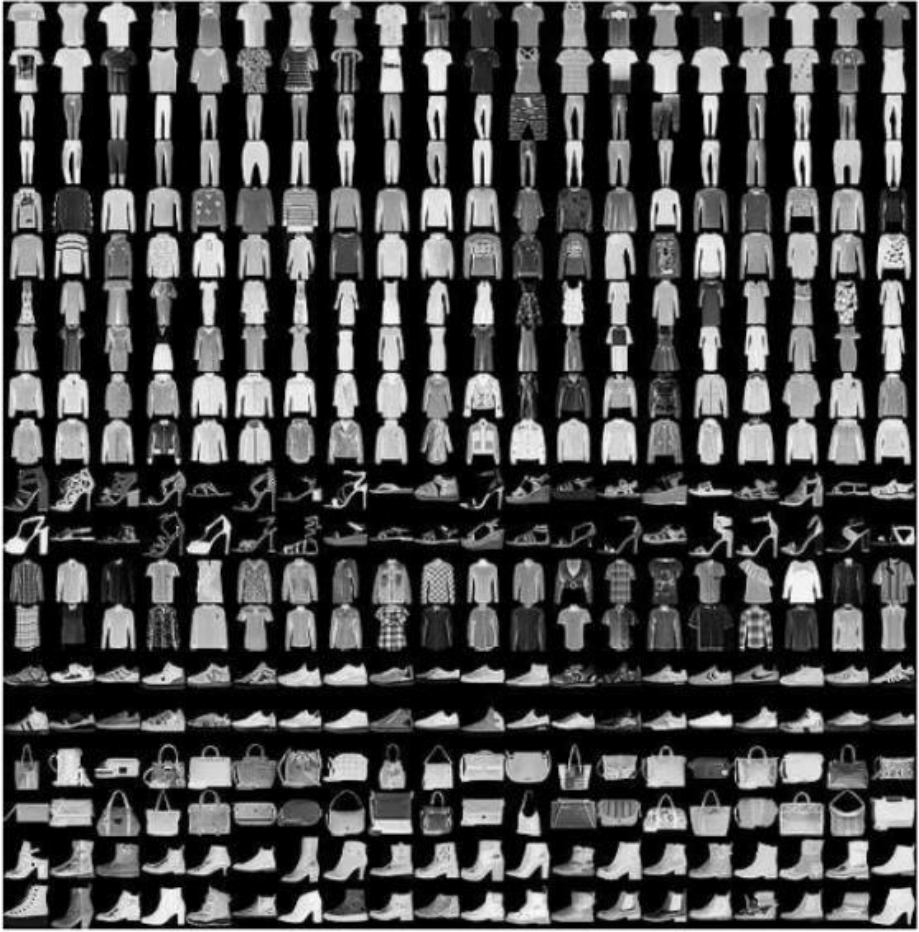
NOTE: Latest PyTorch requires Python 3.9 or later.

PyTorch Build	Stable (2.7.0)			Preview (Nightly)	
Your OS	Linux		Mac	Windows	
Package	Pip		LibTorch	Source	
Language	Python			C++ / Java	
Compute Platform	CUDA 11.8	CUDA 12.6	CUDA 12.8	ROCm 6.3	CPU
Run this Command:	pip3 install torch torchvision torchaudio				

- pip3 install torch torchvision torchaudio

<https://pytorch.org/get-started/locally/>

Fashion MNIST DataSet

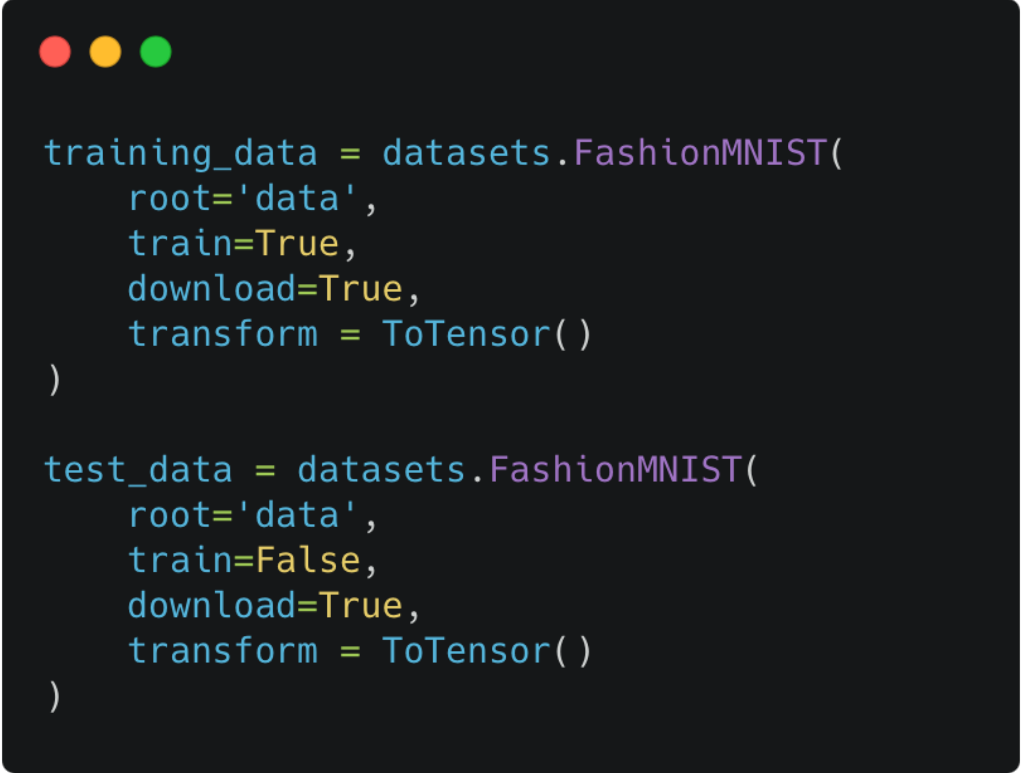
Label	Description	Examples
0	T-Shirt/Top	
1	Trouser	
2	Pullover	
3	Dress	
4	Coat	
5	Sandals	
6	Shirt	
7	Sneaker	
8	Bag	
9	Ankle boots	

필요한 라이브러리 불러오기



```
import torch
from torch import nn
from torch.utils.data import DataLoader
from torchvision import datasets
from torchvision.transforms import ToTensor
```

0. Download Fashion MNIST



```
training_data = datasets.FashionMNIST(  
    root='data',  
    train=True,  
    download=True,  
    transform = ToTensor()  
)  
  
test_data = datasets.FashionMNIST(  
    root='data',  
    train=False,  
    download=True,  
    transform = ToTensor()  
)
```

1. Load DataSet



```
batch_size = 64
```

```
# 데이터로더를 생성합니다.
```

```
train_dataloader = DataLoader(training_data, batch_size=batch_size)
```

```
test_dataloader = DataLoader(test_data, batch_size=batch_size)
```

2. Model 생성

```
# 학습에 사용할 CPU나 GPU, MPS 장치를 얻습니다.
device = (
    "cuda"
    if torch.cuda.is_available()
    else "mps"
    if torch.backends.mps.is_available()
    else "cpu"
)
print(f"Using {device} device")

# 모델을 정의합니다.
class NeuralNetwork(nn.Module):
    def __init__(self):
        super().__init__()
        self.flatten = nn.Flatten()
        self.linear_relu_stack = nn.Sequential(
            nn.Linear(28*28, 512),
            nn.ReLU(),
            nn.Linear(512, 512),
            nn.ReLU(),
            nn.Linear(512, 10)
        )

    def forward(self, x):
        x = self.flatten(x)
        logits = self.linear_relu_stack(x)
        return logits

model = NeuralNetwork().to(device)
print(model)
```


3. 모델 최적화 설정



```
loss_fn = nn.CrossEntropyLoss()  
optimizer = torch.optim.SGD(model.parameters(), lr=1e-3)
```

4. 모델 학습 함수

```
def train(dataloader, model, loss_fn, optimizer):
    size = len(dataloader.dataset)
    for batch, (X, y) in enumerate(dataloader):
        X, y = X.to(device), y.to(device)

        # 예측 오류 계산
        pred = model(X)
        loss = loss_fn(pred, y)

        # 역전파
        loss.backward()
        optimizer.step()
        optimizer.zero_grad()

        if batch % 100 == 0:
            loss, current = loss.item(), (batch + 1) * len(X)
            print(f"loss: {loss:>7f} [{current:>5d}/{size:>5d}]" )
```

5. 테스트 데이터 정확도 확인

```
def test(dataloader, model, loss_fn):
    size = len(dataloader.dataset)
    num_batches = len(dataloader)
    model.eval()
    test_loss, correct = 0, 0
    with torch.no_grad():
        for X, y in dataloader:
            X, y = X.to(device), y.to(device)
            pred = model(X)
            test_loss += loss_fn(pred, y).item()
            correct += (pred.argmax(1) == y).type(torch.float).sum().item()
    test_loss /= num_batches
    correct /= size
    print(f"Test Error: \n Accuracy: {(100*correct):>0.1f}%, Avg loss: {test_loss:>8f} \n")
```

5. 모델 학습 시작



```
epochs = 5
for t in range(epochs):
    print(f"Epoch {t+1}\n-----")
    train(train_dataloader, model, loss_fn, optimizer)
    test(test_dataloader, model, loss_fn)
print("Done!")
```

6. Model 저장하기



```
torch.save(model.state_dict(), "model.pth")
```

7. 모델 불러오기



```
model = NeuralNetwork().to(device)
model.load_state_dict(torch.load("model.pth"))
```

8. 모델 예측하기

```
classes = [  
    "T-shirt/top",  
    "Trouser",  
    "Pullover",  
    "Dress",  
    "Coat",  
    "Sandal",  
    "Shirt",  
    "Sneaker",  
    "Bag",  
    "Ankle boot",  
]  
  
model.eval()  
x, y = test_data[0][0], test_data[0][1]  
with torch.no_grad():  
    x = x.to(device)  
    pred = model(x)  
    predicted, actual = classes[pred[0].argmax(0)], classes[y]  
    print(f'Predicted: "{predicted}", Actual: "{actual}"')
```

실습해보기
CNN 모델로 만들어보기

실습해보기
CIFAR-10 데이터로 학습