딥러닝

✅ 1. 비정형 데이터

데이터 유형	예시	특징
이미지	사진, 의료영상 등	2D 배열 (픽셀)
텍스트	뉴스, SNS, 리뷰 등	길이 가변, 순서 중요
오디오	음성 파일, 음악 등	1D 시계열
동영상	연속된 이미지 + 오디오	3D 시공간 정보

유형	예시	처리 도구
이미지	사진, 의료 영상	`torchvision`
텍스트	뉴스, 리뷰, 대화	`torchtext` , `tokenizers`
오디오	음성 파일	`torchaudio`

✅ 2. 전처리 및 정제

📷 이미지

- 크기 통일 (resize)
- 정규화 (0~1로 나누기)
- 증강 (rotation, crop 등)

📝 텍스트

- 토큰화 (단어/문장 → 숫자 인덱스)
- 패딩 (길이 맞추기)
- 정규화 (소문자화, 특수문자 제거)

이미지(예시)

```
from torchvision import transforms

transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
```

```
    transforms.Normalize(mean=[0.5]*3, std=[0.5]*3)
])
```

텍스트 (예시)

```
from torchtext.data.utils import get_tokenizer
tokenizer = get_tokenizer("basic_english")
tokens = tokenizer("This is a sample sentence.")
```

✅ 3. 데이터셋 구성

- PyTorch: `Dataset`, `DataLoader` 클래스로 정의
- 학습용 / 검증용 / 테스트용 분리

```
class Dataset(Dataset):
    def init(self, data, labels, transform=None):
        self.data = data
        self.labels = labels
        self.transform = transform

    def __len__(self):
        return len(self.data)

    def __getitem__(self, idx):
        x = self.data[idx]
        y = self.labels[idx]
        if self.transform:
            x = self.transform(x)
        return x, y
```

✅ 4. 데이터로더 구성

```
from torch.utils.data import DataLoader

train_loader = DataLoader(train_dataset, batch_size=32, shuffle=True)
val_loader = DataLoader(val_dataset, batch_size=32)
```

✅ 5. 모델 정의

```
import torch.nn as nn

class MyModel(nn.Module):
    def __init__(self):
        super().__init__()
        self.fc = nn.Linear(784, 10)

    def forward(self, x):
        return self.fc(x)
```

> 이미지: CNN
>
> 텍스트: RNN, Transformer
>
> 오디오: 1D CNN, WaveNet 등

✅ 6. 학습 설정

```
import torch.optim as optim

model = MyModel().to(device)
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)
```

문제 유형	추천 손실 함수	이유
이진 분류	`BCEWithLogitsLoss`	Sigmoid 포함, 수치 안정성
다중 클래스 분류	`CrossEntropyLoss`	softmax + log 내장, 효율적
멀티라벨 분류	`BCEWithLogitsLoss`	클래스별 독립 예측
일반 회귀	`MSELoss` , `SmoothL1Loss`	안정성 / 이상치 대응
이상치가 많은 회귀	`L1Loss` , `SmoothL1Loss`	이상치에 강한 손실
임베딩 거리 학습	`CosineEmbeddingLoss`	임베딩 유사도 최적화
순위 학습	`MarginRankingLoss`	쌍(pair) 간 순서 학습
지식증류 / 분포 예측	`KLDivLoss`	분포 간 차이 최소화

✅ 7. 학습

예시

```
for epoch in range(epochs):
    model.train()
    for x_batch, y_batch in train_loader:
        x_batch, y_batch = x_batch.to(device), y_batch.to(device)

        optimizer.zero_grad()
        output = model(x_batch)
        loss = criterion(output, y_batch)
        loss.backward()
        optimizer.tep()
```

✅ 8. 검증 및 테스트

예시

```
model.eval()
correct = 0
total = 0

with torch.no_grad():
    for x_batch, y_batch in val_loader:
        x_batch, y_batch = x_batch.to(device), y_batch.to(device)
        outputs = model(x_batch)
        _, predicted = torch.max(outputs, 1)
        correct += (predicted == y_batch).sum().item()

print(f"Accuracy: {100 * correct / total:.2f}%")
```

✅ 9. 결과 저장

📌 유형별 평가지표 요약

유형	대표 지표	비고
이미지 분류	Accuracy, F1-score, Top-k Accuracy	불균형 클래스면 F1-score 사용
객체 탐지	mAP, IoU	박스 정확도 + 클래스 평가
텍스트 분류	Accuracy, F1-score	감정 분석, 뉴스 분류 등
텍스트 생성	BLEU, ROUGE, METEOR, BERTScore	번역, 요약, 챗봇 등
음성 인식	WER, CER	TTS/ASR 평가 기준
음성 생성	MOS, STOI, PESQ	정량/정성 혼합 평가