데이터프로그래밍 기초 10일차

2026-1 DS Bootcamp

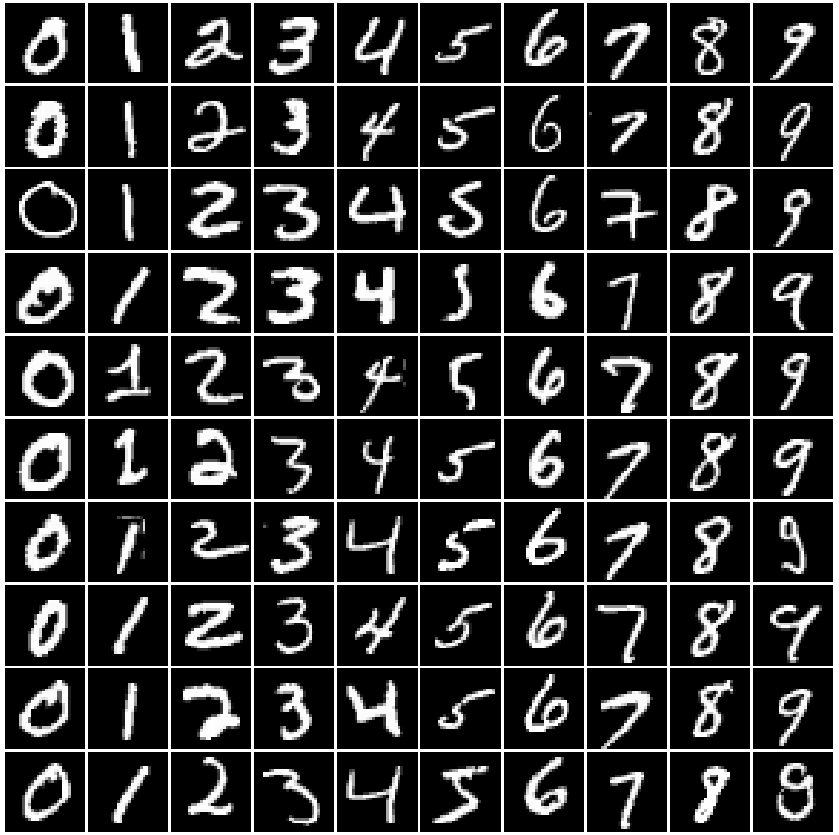
부산대학교
데이터사이언스전문대학원
석사과정 박민서

PUSAN NATIONAL UNIVERSITY
Graduate School of Data Science

CONTENTS

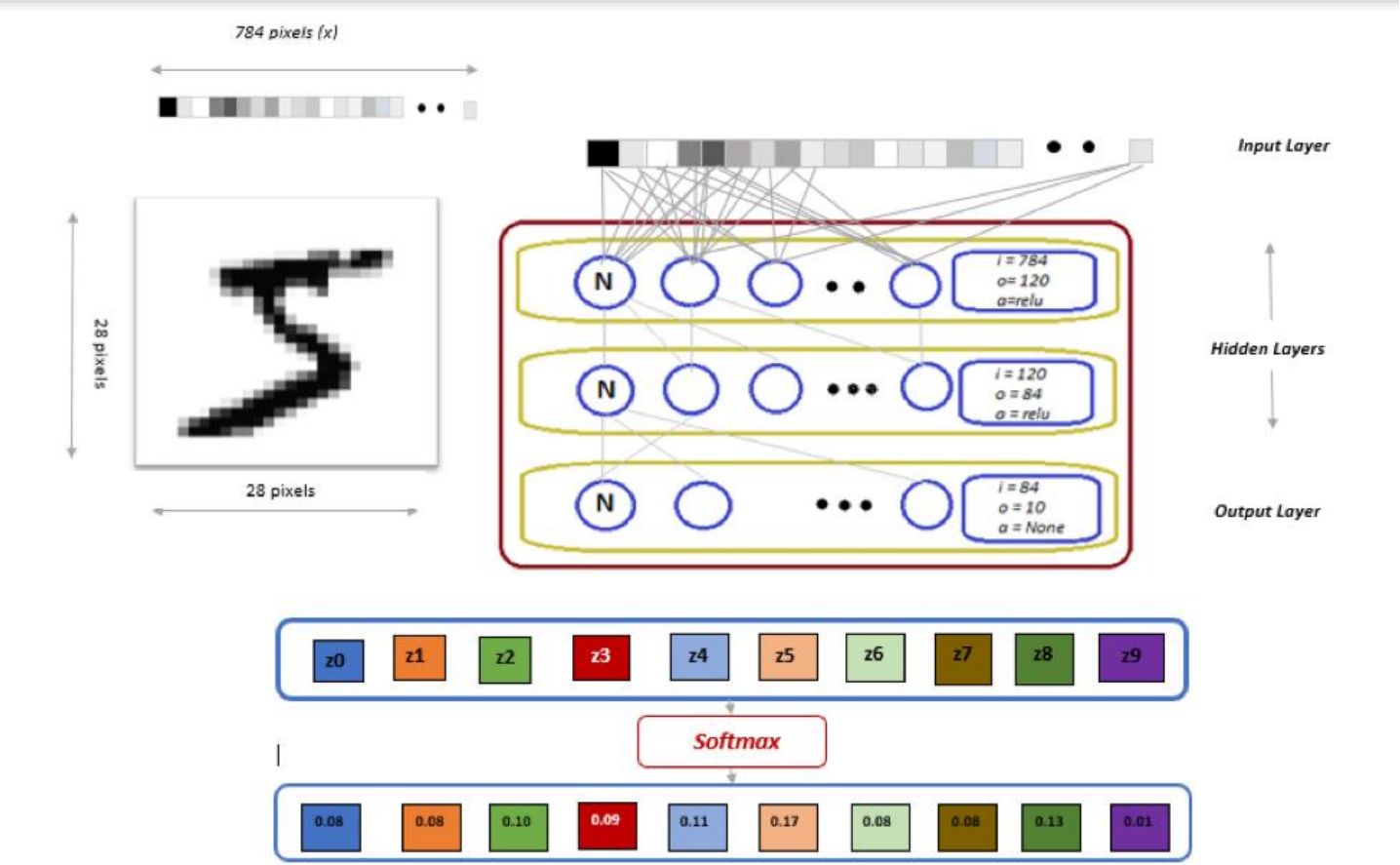
MNIST Dataset

- 딥러닝 처음 배울 때 사용하는 공개 데이터셋
- 미국의 NIST에서 이미지 처리 시스템을 위해 모은 0부터 9까지의 숫자 이미지 데이터셋



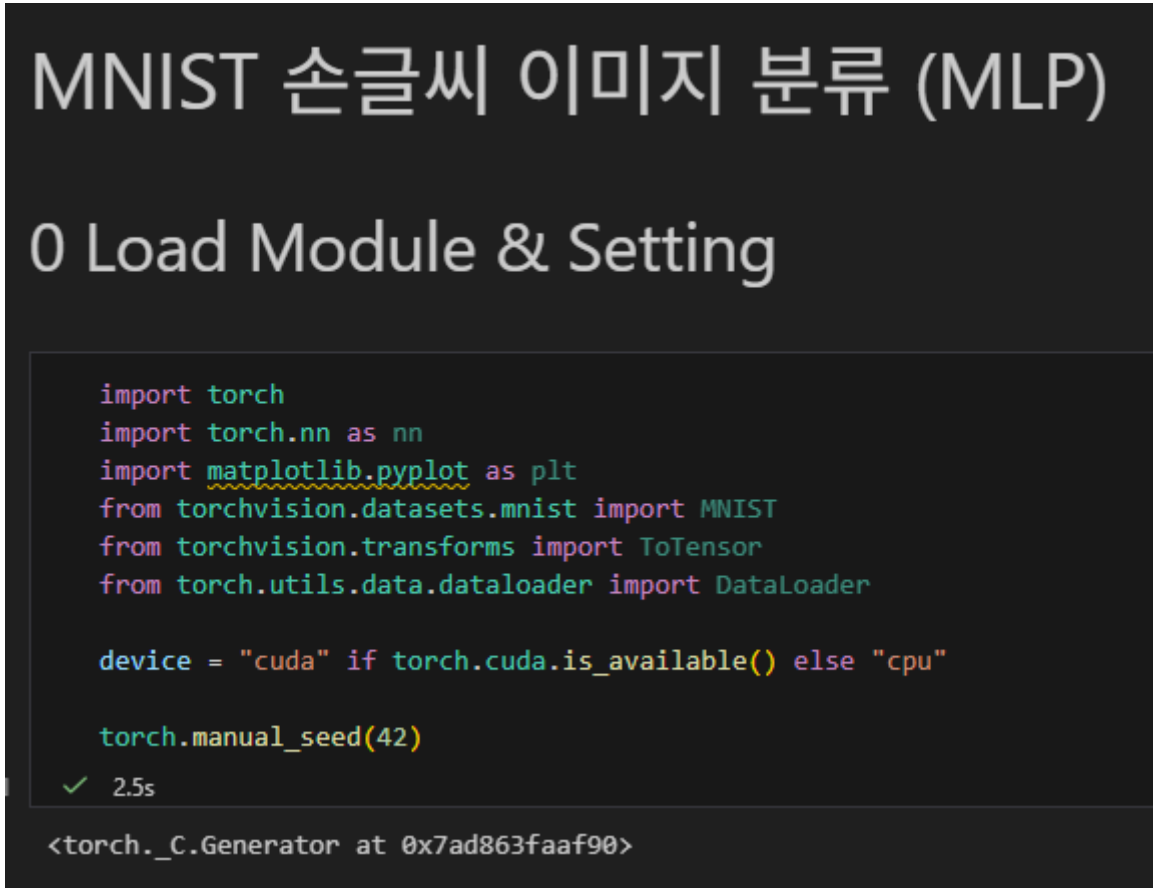
이종민, 「텐초의 파이토치 딥러닝 특강」, 골든래빗, 2022,

- 손글씨 이미지 분류 모델 만들기 (MLP, Multi-Layered Perceptron)



이종민, 「텐초의 파이토치 딥러닝 특강」, 골든래빗, 2022,

- **손글씨 이미지 분류 모델 만들기 (MLP, Multi-Layered Perceptron)**



이종민, 「텐초의 파이토치 딥러닝 특강」, 골든래빗, 2022,

- **손글씨 이미지 분류 모델 만들기 (MLP, Multi-Layered Perceptron)**

1. Data Preparation

```
# 이미지를 tensor로 변환 (파이토치는 텐서만을 입력으로 받음)
training_data = MNIST(root="./", train=True, download=True, transform=ToTensor())
test_data = MNIST(root="./", train=False, download=True, transform=ToTensor())

print(len(training_data))
print(len(test_data))
```

✓ 0.0s

```
60000
10000
```

- 손글씨 이미지 분류 모델 만들기 (MLP, Multi-Layered Perceptron)

1. Data Preparation

```
# 이미지를 tensor로 변환 (파이토치는 텐서만을 입력으로 받음)
training_data = MNIST(root="./", train=True, download=True, transform=ToTensor())
test_data = MNIST(root="./", train=False, download=True, transform=ToTensor())

print(len(training_data))
print(len(test_data))
```

✓ 0.0s

```
60000
10000
```

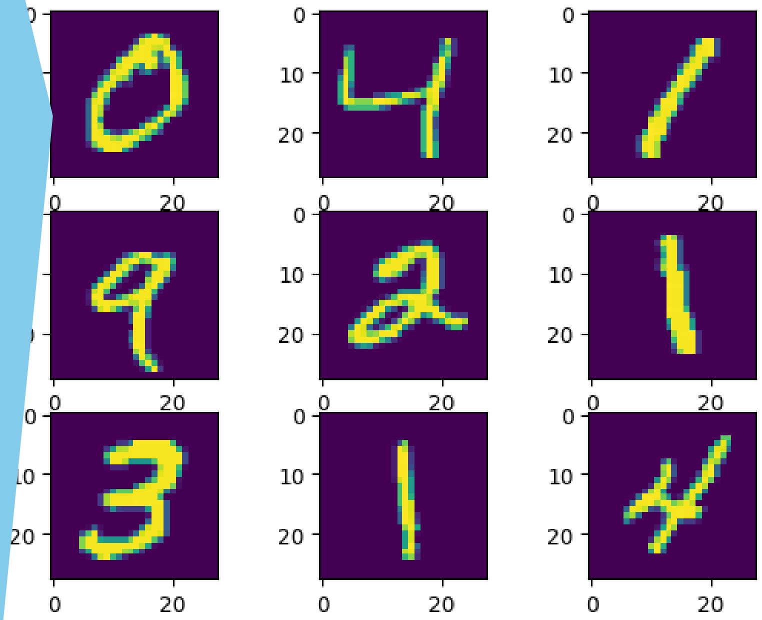
- **손글씨 이미지 분류 모델 만들기 (MLP, Multi-Layered Perceptron)**

2. Data EDA (?)

```
training_data.data[0]
```

✓ 0.0s

```
tensor([[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
           0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0],
        [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
           0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0],
        [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
           0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0],
        [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
           0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0],
        [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
           0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0],
        [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   3,  18,
          18,  18, 126, 136, 175,  26, 166, 255, 247, 127,   0,   0,   0,   0],
        [  0,   0,   0,   0,   0,   0,   0,   0,  30,  36,  94, 154, 170, 253,
         253, 253, 253, 253, 225, 172, 253, 242, 195,  64,   0,   0,   0,   0],
        [  0,   0,   0,   0,   0,   0,   0,  49, 238, 253, 253, 253, 253, 253,
         253, 253, 253, 251,  93,  82,  82,  56,  39,   0,   0,   0,   0,   0],
        [  0,   0,   0,   0,   0,   0,   0,  18, 219, 253, 253, 253, 253, 253,
         198, 182, 247, 241,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0],
        [  0,   0,   0,   0,   0,   0,   0,   0,  80, 156, 107, 253, 253, 205,
          11,   0,  43, 154,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0],
        [  0,   0,   0,   0,   0,   0,   0,   0,   0,  14,   1, 154, 253,  90,
           0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0],
        [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 139, 253, 190,
           2,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0],
        [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  11, 190, 253,
...
        [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
           0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0],
        [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
           0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0]],
       dtype=torch.uint8)
```



이종민, 「텐초의 파이토치 딥러닝 특강」, 골든래빗, 2022,

- 손글씨 이미지 분류 모델 만들기 (MLP, Multi-Layered Perceptron)

3. Define Dataset & Dataloader

```
# DataLoader() : 원하는 배치 크기, 데이터를 섞어서 사용할지, cpu 코어 몇개 사용할지 결정 가능
# 학습 데이터 섞는 이유
# 6000장씩 0, 1, 2, .. 이렇게 순서대로 나타내는 label일 경우
# 모델은 계속해서 첫 6000장에 대해서 0을 출력하게 됨 -> 학습에 영향이 안 좋다
# test_data는 안 섞어도 상관이 없음
train_loader = DataLoader(training_data, batch_size=32, shuffle=True)
test_loader = DataLoader(test_data, batch_size=32, shuffle=False)
```

✓ 0.0s

- 손글씨 이미지 분류 모델 만들기 (MLP, Multi-Layered Perceptron)

4. Define Model

```
# 이미지는 픽셀로 2차원 데이터이기 때문에 2차원 -> 1차원으로 변형
# 28 * 28 2D 데이터 -> 784 1D 데이터
# gradient descent 방법 Adam 선택
model = nn.Sequential(
    nn.Linear(784, 32), # 길이 784 배열의 픽셀값을 특징 추출해서 32개의 유닛으로 추출
    nn.ReLU(),
    nn.Linear(32, 32), # 추출한거 다시
    nn.ReLU(),
    nn.Linear(32, 10), # 10개 units으로 만들어서 가장 높은 값이 label이 됨
).to(device)

epochs = 20  # 에포크 수 20
lr = 1e-3    # 학습률 0.001
optim = torch.optim.Adam(model.parameters(), lr=lr)
```

✓ 0.3s

이종민, 「텐초의 파이토치 딥러닝 특강」, 골든래빗, 2022,

- 손글씨 이미지 분류 모델 만들기 (MLP, Multi-Layered Perceptron)

5. Train

```python
for epoch in range(epochs):
    for data, label in train_loader:
        optim.zero_grad() # 초기화
        data = torch.reshape(data, (-1, 784)).to(device)
        preds = model(data)

        loss = nn.CrossEntropyLoss()(preds, label.to(device)) # 분류문제니까 cross-entropy loss 사용
        loss.backward()
        optim.step() # 최적화 진행

    print(f"epoch {epoch + 1} loss:{loss}")
```

✓ 3m 2.6s

```
epoch 1 loss:0.24151277542114258
epoch 2 loss:0.502062976360321
epoch 3 loss:0.10415862500667572
epoch 4 loss:0.024286514148116112
epoch 5 loss:0.01421770267188549
epoch 6 loss:0.045864593237638474
epoch 7 loss:0.017290234565734863
epoch 8 loss:0.19971536099910736
epoch 9 loss:0.061532068997621536
epoch 10 loss:0.1832006275653839
epoch 11 loss:0.05878648906946182
epoch 12 loss:0.11025068163871765
epoch 13 loss:0.1963825225830078
epoch 14 loss:0.007888146676123142
epoch 15 loss:0.07111804187297821
epoch 16 loss:0.07704029977321625
epoch 17 loss:0.03500031307339668
epoch 18 loss:0.05503794923424721
epoch 19 loss:0.01288294792175293
epoch 20 loss:0.002667102264240384
```

- 손글씨 이미지 분류 모델 만들기 (MLP, Multi-Layered Perceptron)

6. Inference

```
num_corr = 0 # 분류에 성공한 전체 개수

with torch.no_grad():
    for data, label in test_loader:
        data = torch.reshape(data, (-1, 784)).to(device)

        output = model(data.to(device))
        preds = output.data.max(1)[1]

        corr = preds.eq(label.to(device).data).sum().item()
        num_corr += corr

    print(f"Accuracy:{num_corr/len(test_data)}")
```

✓ 1.1s

```
Accuracy:0.9686
```

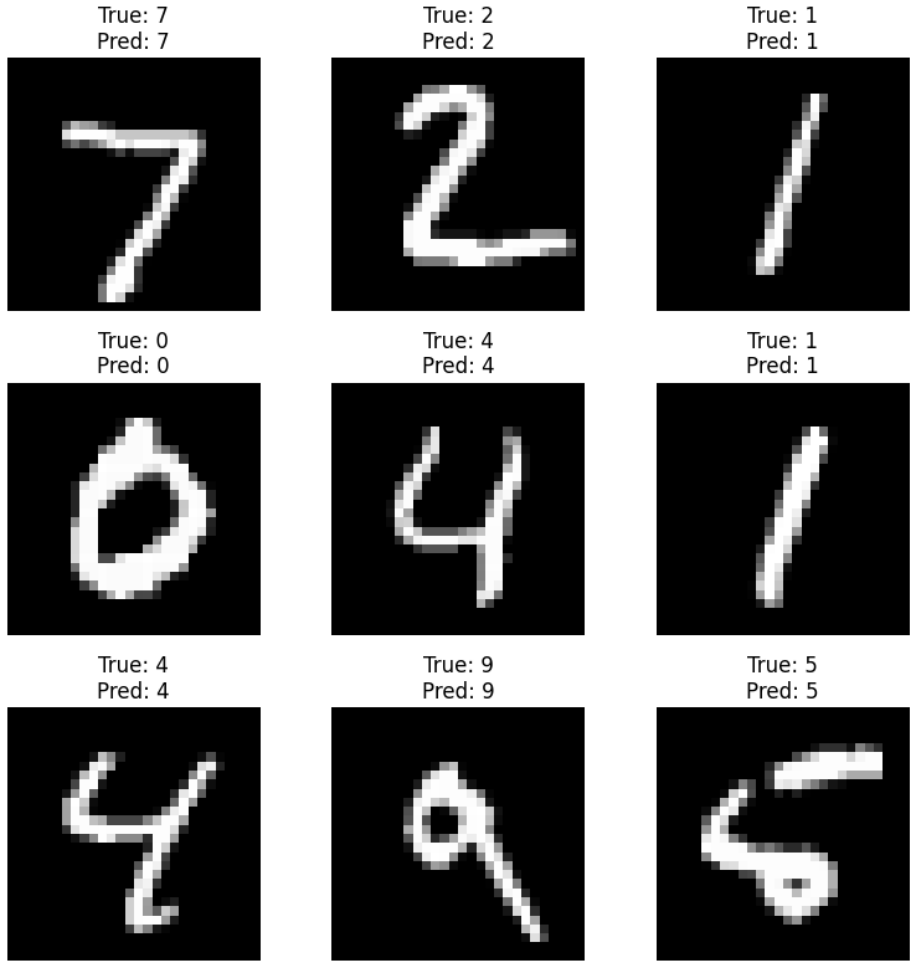
- 손글씨 이미지 분류 모델 만들기 (MLP, Multi-Layered Perceptron)

6. Inference

```
num_corr = 0 # 분류에 성공한 전체 개수

with torch.no_grad():
    for data, label in test_loader:
        data = torch.reshape(data, (-1, 784)).to(device)

        output = model(data.to(device))
        preds = output.data.max(1)[1]

        corr = preds.eq(label.to(device).data).sum().item()
        num_corr += corr

    print(f"Accuracy:{num_corr/len(test_data)}")
```

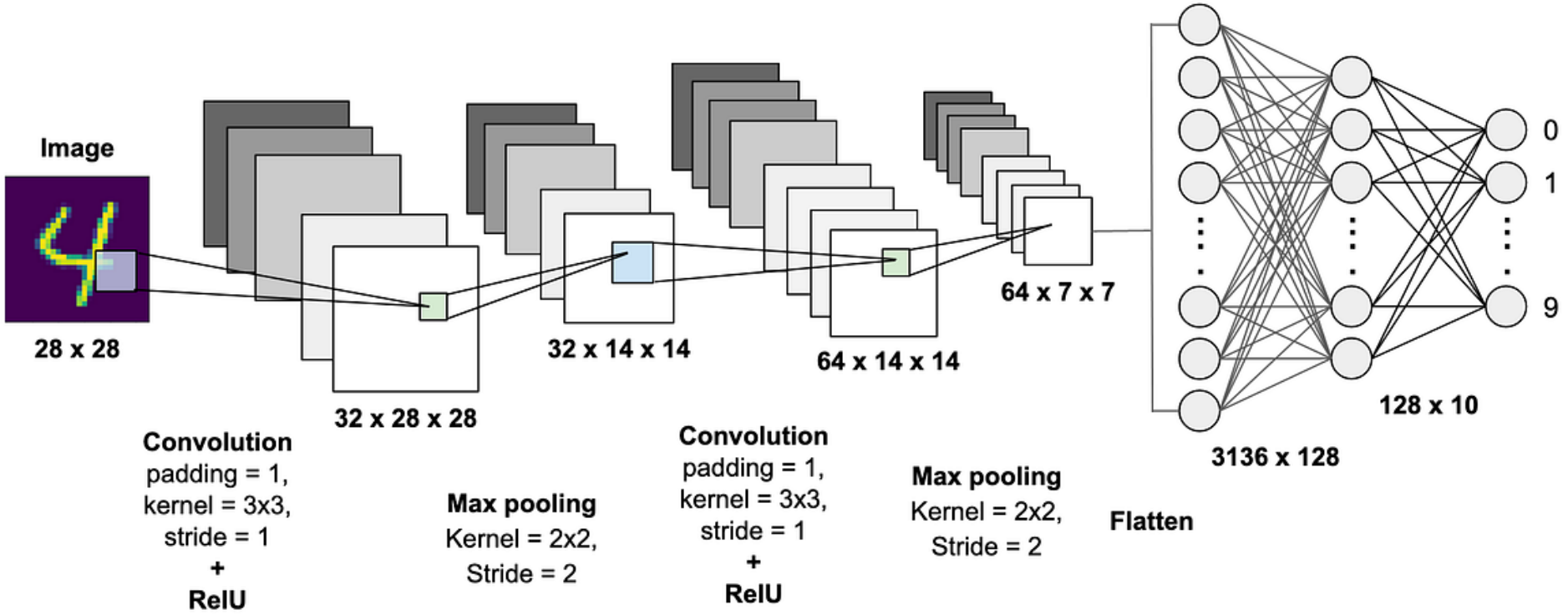
✓ 1.1s

```
Accuracy:0.9686
```



이종민, 「텐초의 파이토치 딥러닝 특강」, 골든래빗, 2022,

- 손글씨 이미지 분류 모델 만들기 (CNN, Convolutional Neural Network)



이종민, 「텐초의 파이토치 딥러닝 특강」, 골든래빗, 2022,

- **손글씨 이미지 분류 모델 만들기 (CNN, Convolutional Neural Network)**

MNIST 손글씨 이미지 분류 (CNN)

0 Load Module & Setting

```
import torch
import torch.nn as nn
import matplotlib.pyplot as plt
from torchvision.datasets import MNIST
from torchvision.transforms import ToTensor
from torch.utils.data import Dataset, DataLoader
```

✓ 0.0s

```
device = "cuda" if torch.cuda.is_available() else "cpu"
torch.manual_seed(42)
```

✓ 0.0s

```
<torch._C.Generator at 0x7ad863faaf90>
```

- **손글씨 이미지 분류 모델 만들기 (CNN, Convolutional Neural Network)**

1. Data Preparation & preprocessing(skip)

2. Define Dataset

```
class MNISTDataset(Dataset):
    def __init__(self, train=True):
        self.data = MNIST(
            root="./",
            train=train,
            download=True,
            transform=ToTensor()
        )

    def __len__(self):  # len() 적용시 출력할 것 반환
        return len(self.data)

    def __getitem__(self, idx): # idx로 접근 시 반환할 데이터
        image, label = self.data[idx]
        return image, label
```

✓ 0.0s

```
train_dataset = MNISTDataset(train=True)
test_dataset = MNISTDataset(train=False)

train_loader = DataLoader(train_dataset, batch_size=32, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=32, shuffle=False)

print(len(train_dataset))
print(len(test_dataset))
```

✓ 0.0s

```
60000
10000
```

이종민, 「텐초의 파이토치 딥러닝 특강」, 골든래빗, 2022,

- 손글씨 이미지 분류 모델 만들기 (CNN, Convolutional Neural Network)

3. Define Model

```python
class CNN(nn.Module):
    def __init__(self):
        super().__init__()
        # 2D Convolution
        self.conv1 = nn.Conv2d(1, 32, kernel_size=3, padding=1)
        # 2D MaxPooling
        self.pool = nn.MaxPool2d(2, 2)
        # 2D Convolution
        self.conv2 = nn.Conv2d(32, 64, kernel_size=3, padding=1)
        # Activation Function
        self.relu = nn.ReLU()
        # Fully-Connected Layer
        self.fc1 = nn.Linear(64 * 7 * 7, 128)
        self.fc2 = nn.Linear(128, 10)
```

```python
    def forward(self, x):
        x = self.conv1(x)   # 28 * 28
        x = self.relu(x)
        x = self.pool(x)    # 14 * 14

        x = self.conv2(x)   # 14 * 14
        x = self.relu(x)
        x = self.pool(x)    # 7 * 7

        x = x.view(x.size(0), -1) # flatten

        x = self.fc1(x)
        x = self.relu(x)
        x = self.fc2(x)
        return x
```

이종민, 「텐초의 파이토치 딥러닝 특강」, 골든래빗, 2022,

- 손글씨 이미지 분류 모델 만들기 (CNN, Convolutional Neural Network)

```
model = CNN().to(device)
epochs = 5
lr = 1e-3
optimizer = torch.optim.Adam(model.parameters(), lr=lr)
loss_func = nn.CrossEntropyLoss()
```

이종민, 「텐초의 파이토치 딥러닝 특강」, 골든래빗, 2022,

- 손글씨 이미지 분류 모델 만들기 (CNN, Convolutional Neural Network)

4. Train

```
for epoch in range(epochs):
    model.train()
    running_loss = 0

    for data, label in train_loader:
        data = data.to(device)
        label = label.to(device)

        optimizer.zero_grad()
        output = model(data)
        loss = loss_func(output, label)
        loss.backward()
        optimizer.step()

        running_loss += loss.item()

    print(f"Epoch {epoch + 1}, Loss: {running_loss/len(train_loader):.4f}")
```

```
Epoch 1, Loss: 0.1455
Epoch 2, Loss: 0.0449
Epoch 3, Loss: 0.0298
Epoch 4, Loss: 0.0219
Epoch 5, Loss: 0.0158
```

이종민, 「텐초의 파이토치 딥러닝 특강」, 골든래빗, 2022,

- 손글씨 이미지 분류 모델 만들기 (CNN, Convolutional Neural Network)

5. Inference

```
model.eval()
num_corr = 0

with torch.no_grad():
    for data, label in test_loader:
        data = data.to(device)
        label = label.to(device)

        output = model(data)
        preds = output.argmax(dim=1)
        num_corr += (preds == label).sum().item()

print(f"Accuracy: {num_corr / len(test_dataset):.4f}")
```

✓ 1.1s

```
Accuracy: 0.9905
```

이종민, 「텐초의 파이토치 딥러닝 특강」, 골든래빗, 2022,

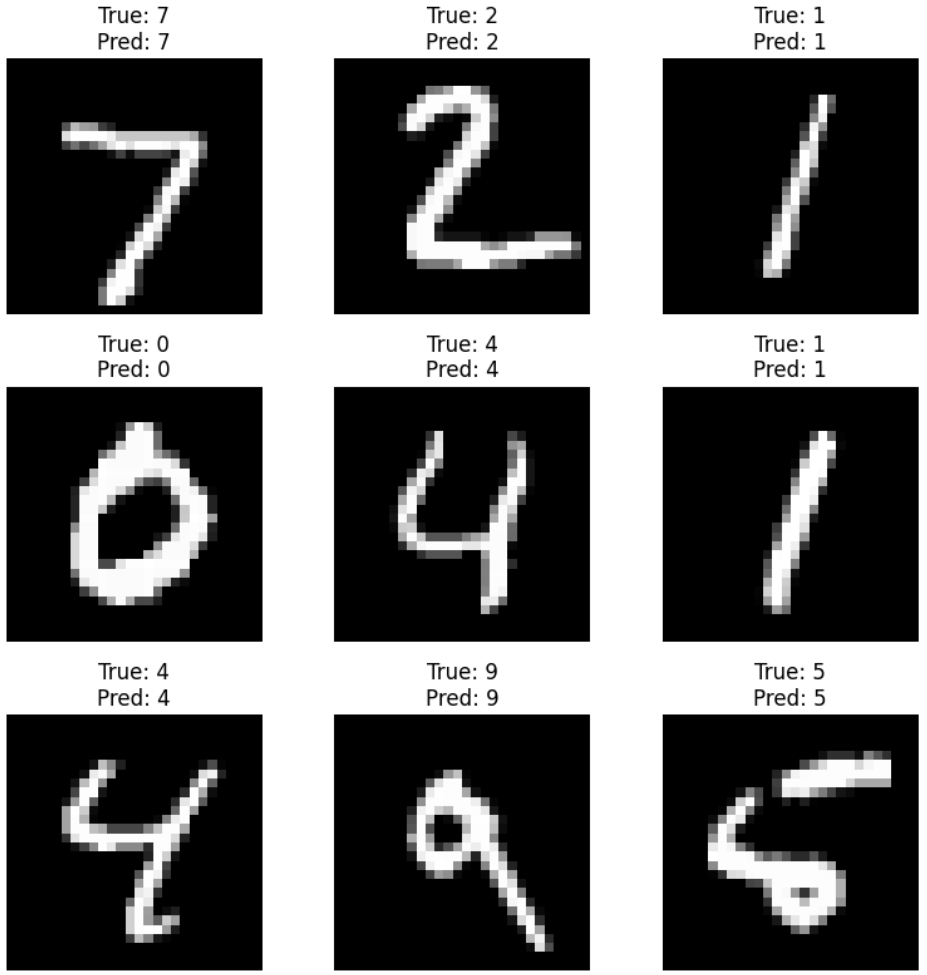
- 손글씨 이미지 분류 모델 만들기 (CNN, Convolutional Neural Network)

5. Inference

```
model.eval()
num_corr = 0

with torch.no_grad():
    for data, label in test_loader:
        data = data.to(device)
        label = label.to(device)

        output = model(data)
        preds = output.argmax(dim=1)
        num_corr += (preds == label).sum().item()

print(f"Accuracy: {num_corr / len(test_dataset):.4f}")
```

✓ 1.1s

```
Accuracy: 0.9905
```



이종민, 「텐초의 파이토치 딥러닝 특강」, 골든래빗, 2022,

Q&A

▪ Data Science 연구 설정

- Introduction (서론) -> Related works & Background (기존 연구) -> Methods (실험 방법)
  -> Experiments (실험) -> Results (실험 결과 및 해석) -> Conclusion (결론) 순으로 논문 작성 (대부분)

- 본인이 알고 있는, 본인이 원하는 **Domain에 대한 Knowledge**를 탄탄히 쌓는 것이 중요
  - Domain Knowledge 없이 연구 설정을 진행할 시, 불분명한 목표가 생김

- Related works & Background에 중점을 둬 Domain에 대한 이해를 갖춘 뒤, 연구에 대한 문제 정의가 필요
  - Domain (ex. 제조, 헬스케어, 금융, …)에서 어떤 연구들이 진행되었는지
  - 어떤 문제 정의를 갖고 연구를 진행했는지
  - 실험 설계를 어떻게 해서 문제를 해결하려 했는지

▪ Data Science 연구 설정

- **Introduction (서론) -> Related works & Background (기존 연구)** -> Methods (실험 방법)
  -> Experiments (실험) -> Results (실험 결과 및 해석) -> Conclusion (결론) 순으로 논문 작성 (대부분)

- 기존 연구에 대한 많은 Survey가 필요
  → Google Scholar: https://scholar.google.com/
  → DBPia: https://www.dbpia.co.kr/
  → …

- 검색한 논문이 인용한 논문들도 연결해서 검색
  → 이전에 한계점이 있었던 논문이거나 더 이전에 같은 Domain의 연구

▪ Data Science 연구 설정

- Introduction (서론) -> Related works & Background (기존 연구) -> **Methods (실험 방법)**
  -> Experiments (실험) -> Results (실험 결과 및 해석) -> Conclusion (결론) 순으로 논문 작성 (대부분)

- **“Trendy한 방법 ≠ 만능”**

- 전체 실험에 대한 구조화가 필요
  - 입력과 출력은 어떤 것인지
  - 분류, 회귀, 추천, 생성, 강화학습, 이상 탐지 … (유형 결정)
  - 나의 연구와 비교할 기존에 있는 Baseline 모델 설정

- **데이터 수집 – 데이터 EDA – 데이터 Preprocessing** – 모델 선정 – 훈련 및 추론

▪ Data Science 연구 설정

- Introduction (서론) -> Related works & Background (기존 연구) -> Methods (실험 방법)
  -> Experiments (실험) -> **Results (실험 결과 및 해석) -> Conclusion (결론)** 순으로 논문 작성 (대부분)

- 실험 결과를 **정량적**으로 나타낼 평가지표
  → Accuracy, Recall, Precision, F1-Score
  → MAE, MSE, RMSE, …

- Ablation Study (★)
  → 나의 실험과 나의 실험에서 주요 설계 부분을 빼거나 대체한 다른 방법과 비교
  → 나의 실험 설계의 정당성을 입증

- 결론: '나의 실험이 기존의 한계점을 (어떤 식으로) 극복했다.', '기존과는 다른 Novelty(신선함)이 있다.',
  '나의 방법에도 한계점이 존재했다. 향후에 이런 점을 극복할 방법, 연구의 확장 제시'

Q&A