데이터프로그래밍 기초 10일차

2026-1 DS Bootcamp

부산대학교
데이터사이언스전문대학원
석사과정 박민서

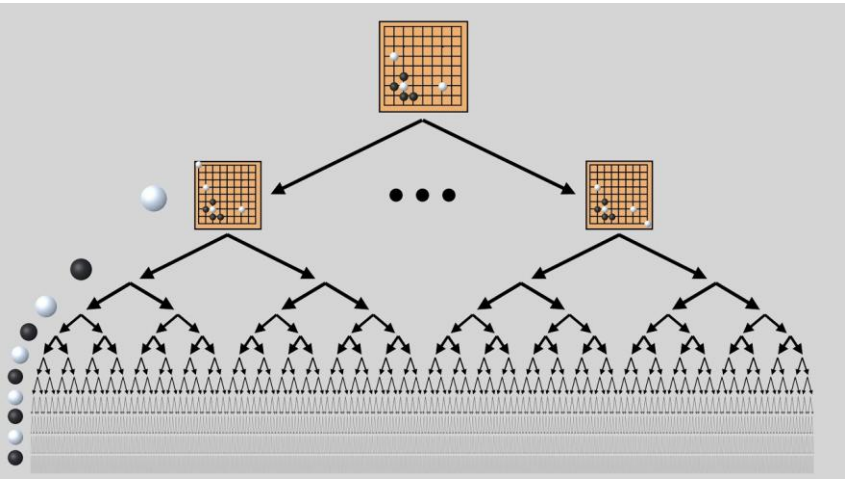
PUSAN NATIONAL UNIVERSITY
Graduate School of Data Science

CONTENTS

▪ 머신러닝 (Machine Learning) & 딥러닝 (Deep Learning)

- 인공지능: 인간의 지능을 모방한 기계 혹은 그에 준하는 컴퓨터 기술
- 머신러닝 (ML): 반복적으로 자신의 성능을 개선하는 컴퓨터 알고리즘
- 딥러닝 (DL): 인공 신경망을 활용한 머신러닝 알고리즘

- 인공지능을 주목하게 된 사건

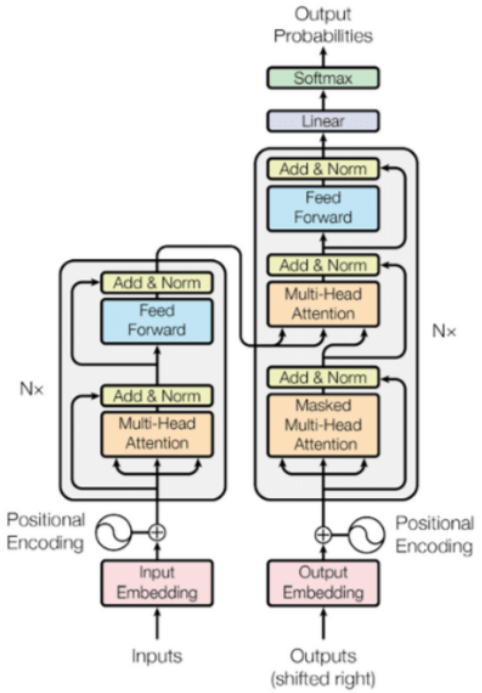


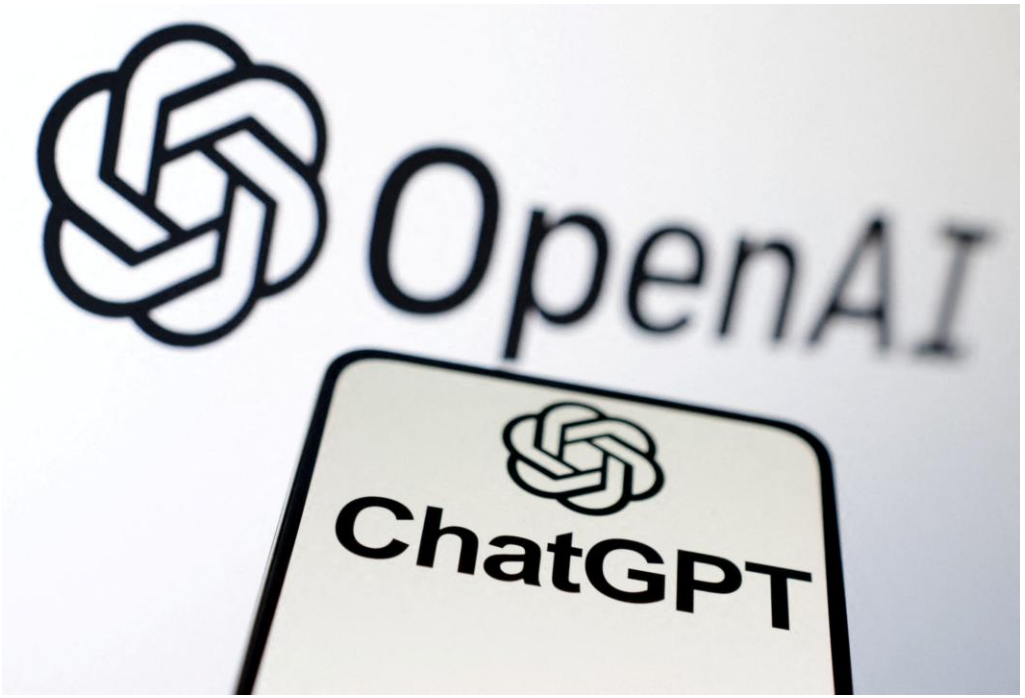
이종민, 「텐초의 파이토치 딥러닝 특강」, 골든래빗, 2022,

- 인공지능을 주목하게 된 사건

Transformer

Attention Is All You Need





이종민, 「텐초의 파이토치 딥러닝 특강」, 골든래빗, 2022,

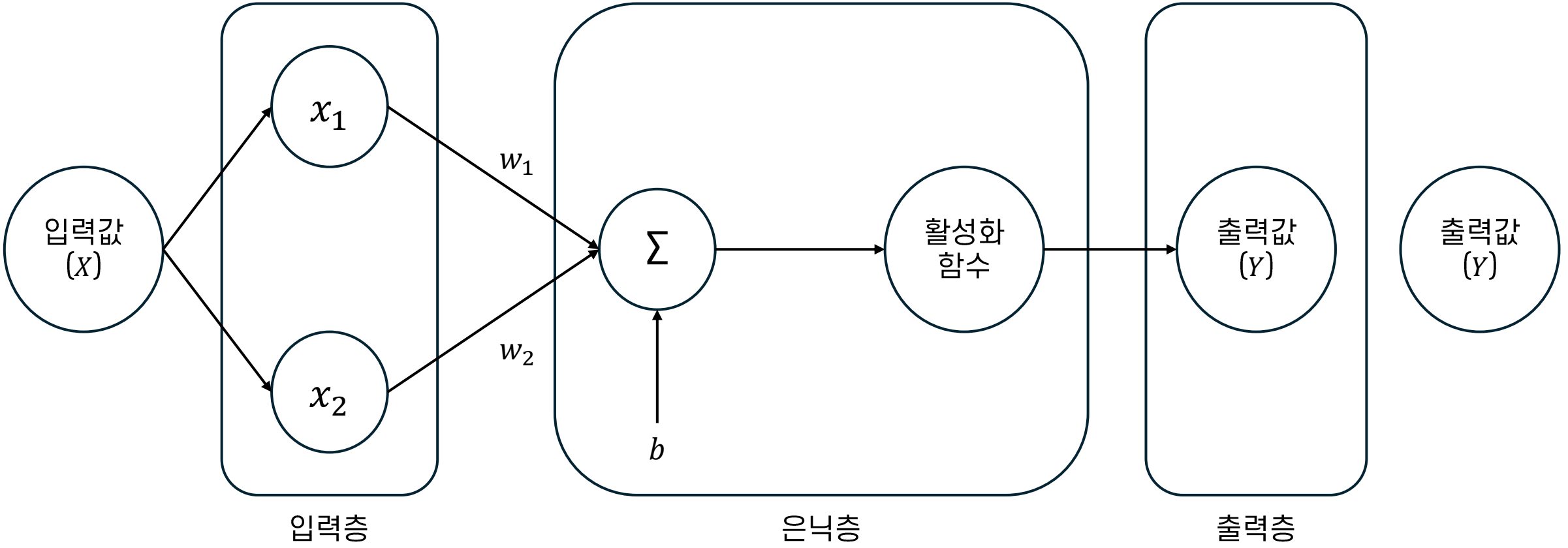
▪ 퍼셉트론 (Perceptron), 인공신경망 (ANN)

- 인간의 뉴런 구조를 수학적으로 구현한 알고리즘

이종민, 「텐초의 파이토치 딥러닝 특강」, 골든래빗, 2022,

▪ 퍼셉트론 (Perceptron), 인공신경망 (ANN)

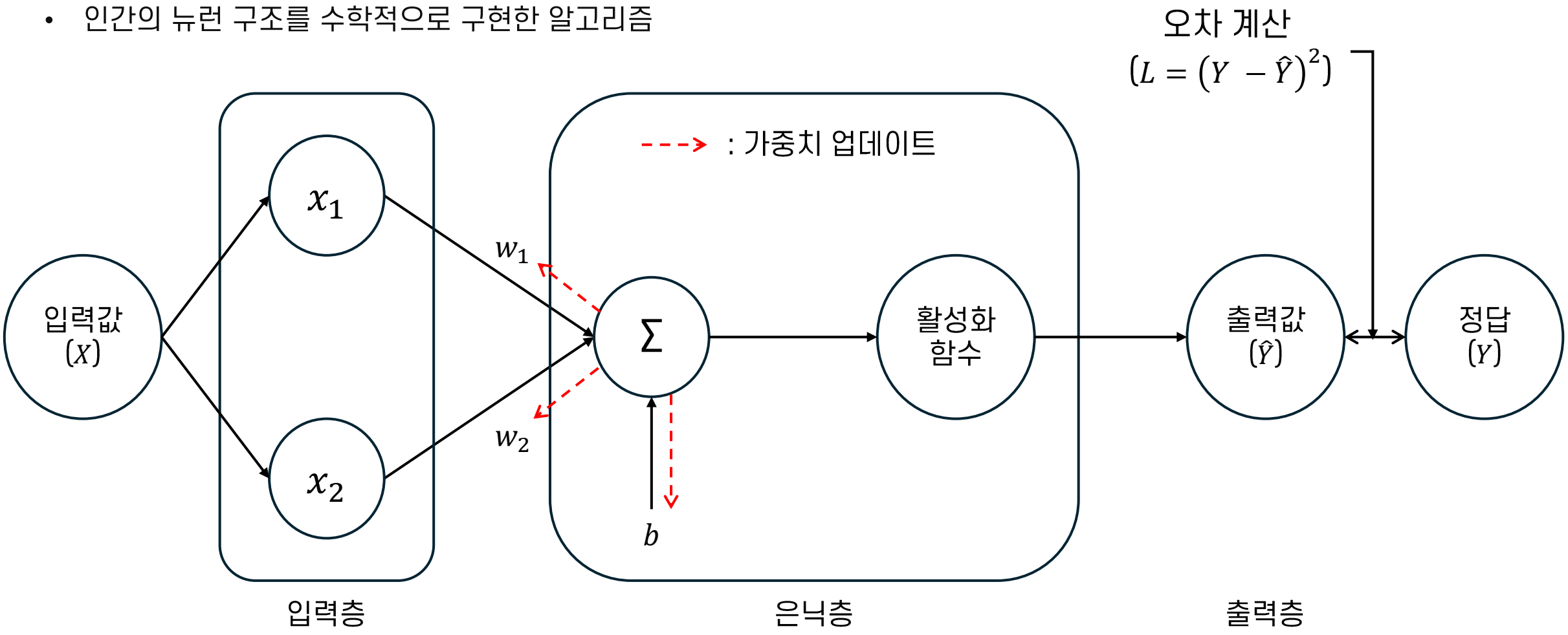
- 인간의 뉴런 구조를 수학적으로 구현한 알고리즘



이종민, 「텐초의 파이토치 딥러닝 특강」, 골든래빗, 2022,

▪ 퍼셉트론 (Perceptron), 인공신경망 (ANN)

- 인간의 뉴런 구조를 수학적으로 구현한 알고리즘

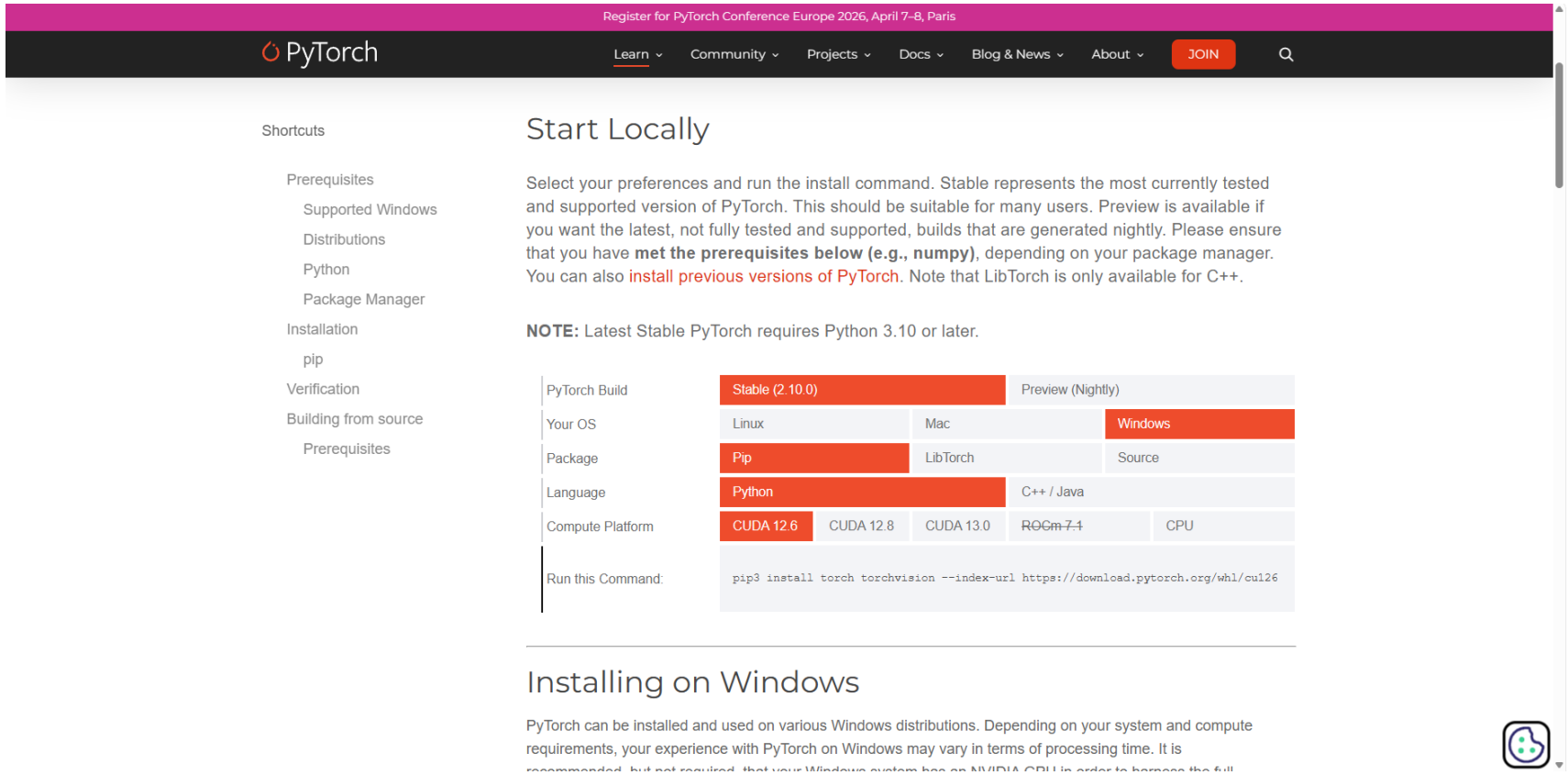


이종민, 「텐초의 파이토치 딥러닝 특강」, 골든래빗, 2022,

Pytorch & Colab

▪ 파이토치 (Pytorch)

- Python 기반의 오픈 소스 ML & DL 라이브러리
- https://pytorch.org/get-started/locally/

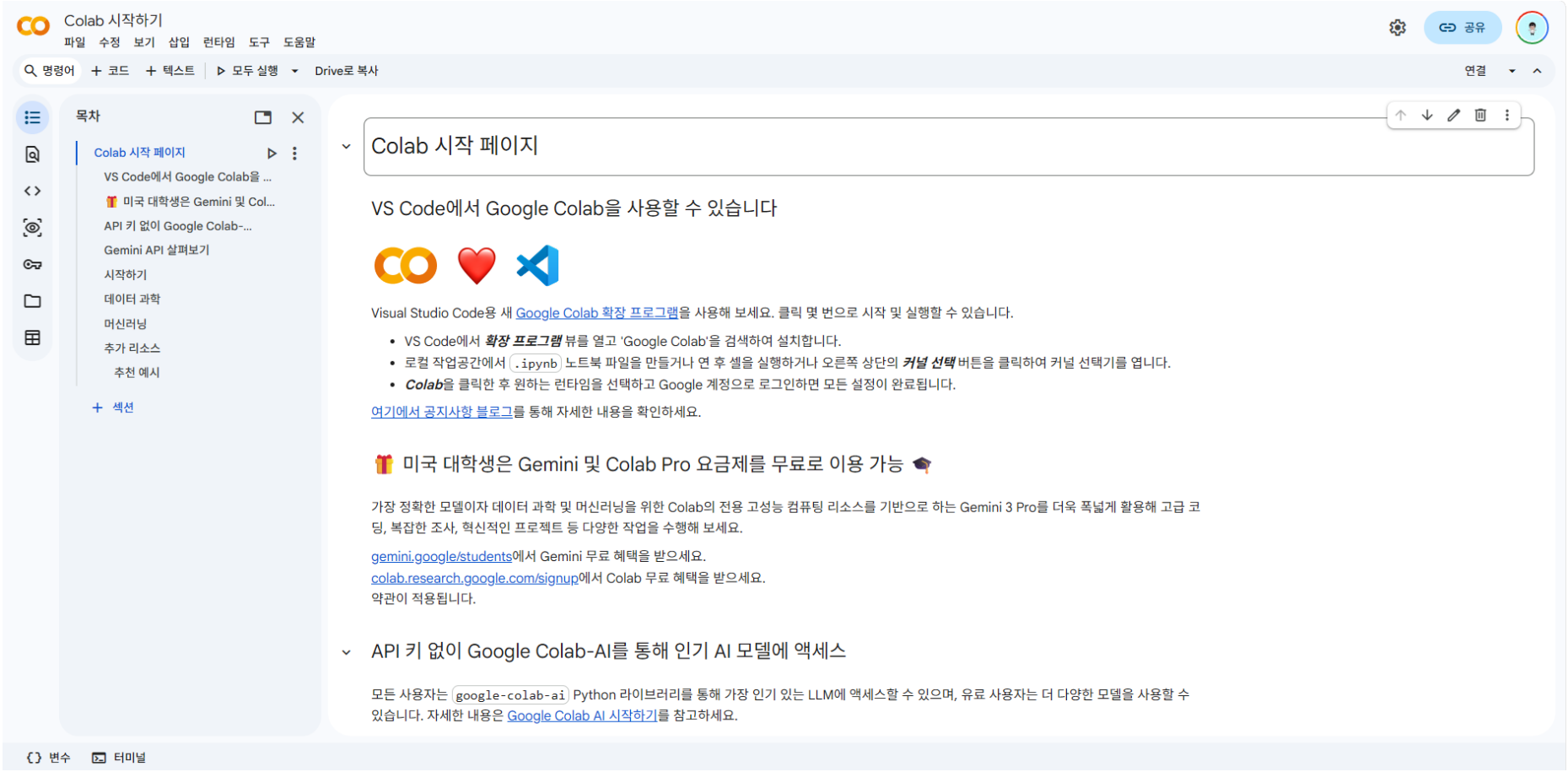


이종민, 「텐초의 파이토치 딥러닝 특강」, 골든래빗, 2022,

Pytorch & Colab

▪ 구글 코랩 (Google Colab.)

- Google에서 제공하는 클라우드 기반 Jupyter Notebook 환경

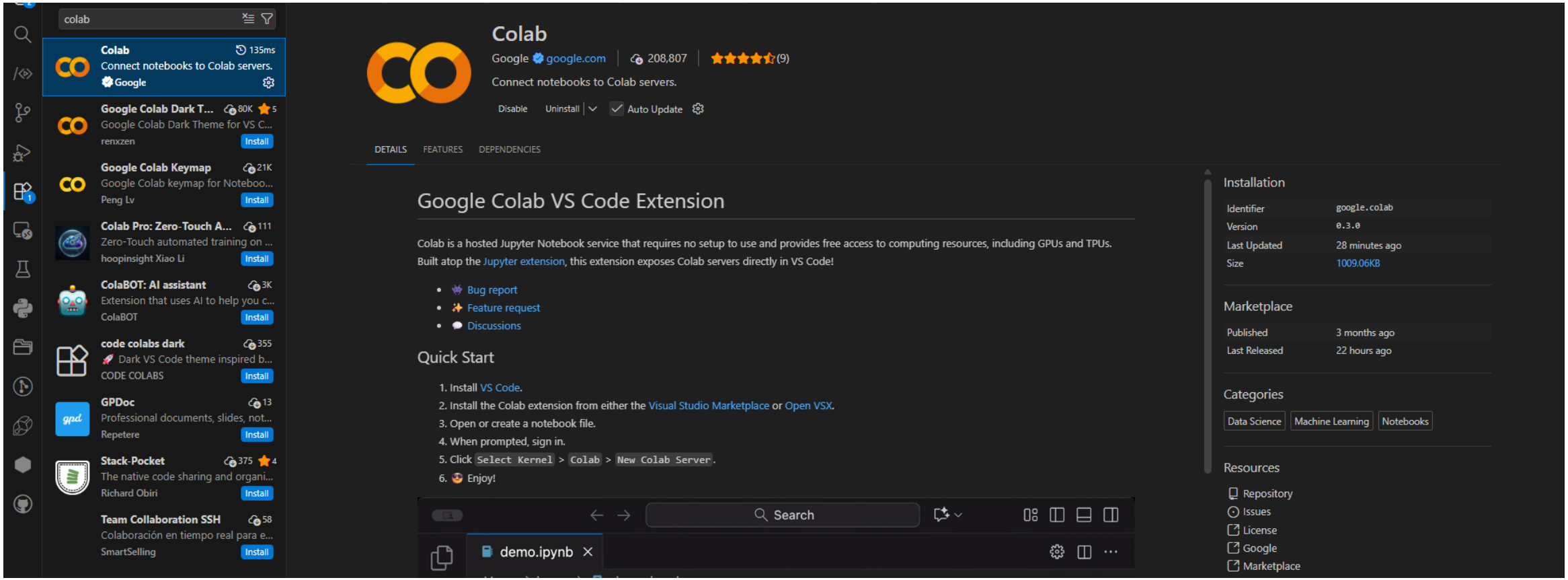


이종민, 「텐초의 파이토치 딥러닝 특강」, 골든래빗, 2022,

Pytorch & Colab

▪ VS Code에서 Colab 실행

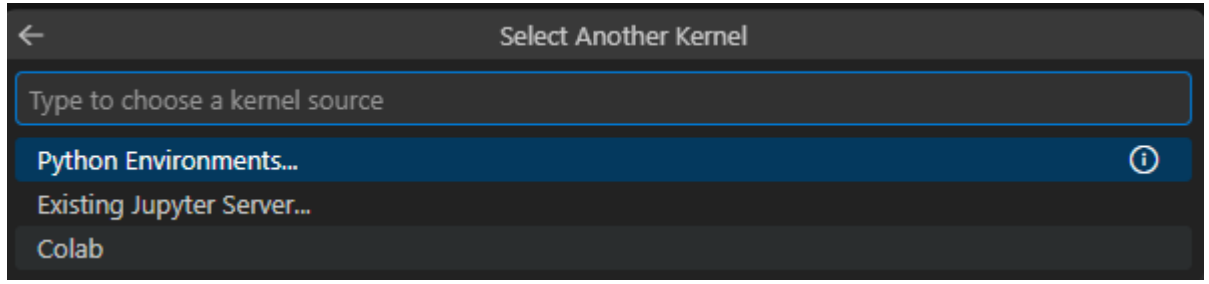
- Extension – ‘Colab’ 검색

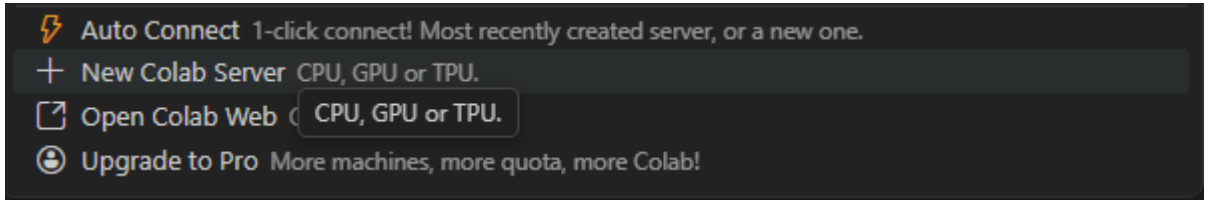


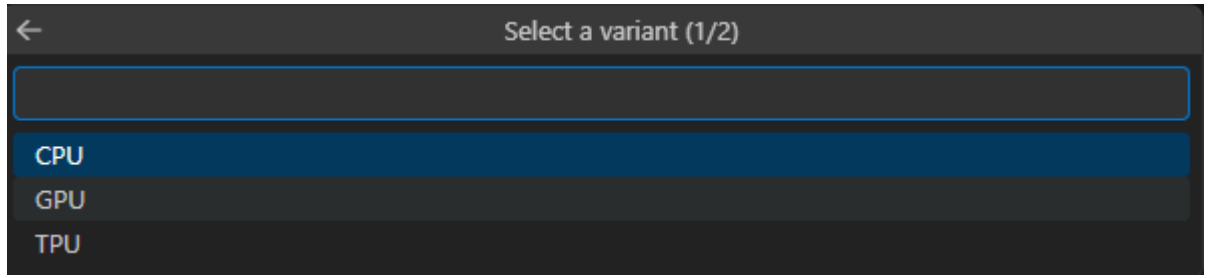
이종민, 「텐초의 파이토치 딥러닝 특강」, 골든래빗, 2022,

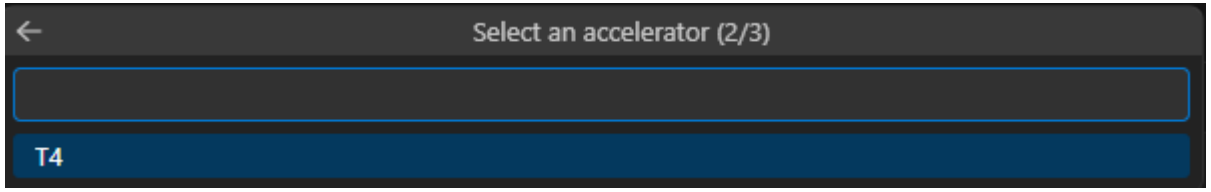
Pytorch & Colab

- **VS Code에서 Colab 실행**
  - Kernel 선택 – Colab – New Colab Server – ‘T4’ GPU 선택 – Enter









이종민, 「텐초의 파이토치 딥러닝 특강」, 골든래빗, 2022,

Pytorch & Colab

▪ VS Code에서 Colab 실행

- ‘nvidia-smi’: 현재 인식하고 있는 GPU 상태 확인
- ‘pip list’: 현재 Kernel에 설치되어 있는 모듈 확인

```
!nvidia-smi
✓ 0.1s
Thu Feb 12 20:00:18 2026
+-----------------------------------------------------------------------------------------+
| NVIDIA-SMI 580.82.07              Driver Version: 580.82.07      CUDA Version: 13.0     |
+-----------------------------------------+------------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Compute M. |
|                                         |                        |               MIG M. |
|=========================================+========================+======================|
|   0  Tesla T4                       Off |   00000000:00:04.0 Off |                    0 |
| N/A   36C    P8             10W /   70W |       0MiB /  15360MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+

+-----------------------------------------------------------------------------------------+
| Processes:                                                                              |
|  GPU   GI   CI              PID   Type   Process name                        GPU Memory |
|        ID   ID                                                               Usage      |
|=========================================================================================|
|  No running processes found                                                             |
+-----------------------------------------------------------------------------------------+
```

```
!pip list
✓ 2.1s
tokenizers                0.22.2
toml                      0.10.2
tomlkit                   0.13.3
toolz                     0.12.1
torch                     2.9.0+cu128
torchao                   0.10.0
torchaudio                2.9.0+cu128
torchcodec                0.8.0+cu128
torchdata                 0.11.0
torchsummary              1.5.1
torchtune                 0.6.1
torchvision               0.24.0+cu128
tornado                   6.5.1
tqdm                      4.67.3
traitlets                 5.7.1
traittypes                0.2.3
transformers              5.0.0
```

이종민, 「텐초의 파이토치 딥러닝 특강」, 골든래빗, 2022,

Pytorch & Colab

▪ VS Code에서 Colab 실행

- torch.cuda.is_available( ): 현재 Pytorch에서 GPU를 인식하고 있는지 확인

```
import torch
✓ 4.4s

print(torch.cuda.is_available())
✓ 0.0s
True
```

Pytorch & Colab

▪ Pytorch 실습

- 텐서 (tensor): NumPy의 array와 유사한 api 제공 + GPU 계산 가능

```
data = [[1, 2], [3, 4]]

my_arr = np.array(data)
my_tensor = torch.tensor(data)

print(my_arr)
print()
print(my_tensor)
```

✓ 0.0s

```
[[1 2]
 [3 4]]

tensor([[1, 2],
        [3, 4]])
```

이종민, 「텐초의 파이토치 딥러닝 특강」, 골든래빗, 2022,

▪ Pytorch 실습

- 텐서 (tensor): NumPy의 array와 유사한 api 제공 + GPU 계산 가능

```
zeros = torch.zeros((3, 5))
ones = torch.ones((3, 5))

print(zeros)
print()
print(ones)
```

✓ 0.0s

```
tensor([[0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0.]])

tensor([[1., 1., 1., 1., 1.],
        [1., 1., 1., 1., 1.],
        [1., 1., 1., 1., 1.]])
```

이종민, 「텐초의 파이토치 딥러닝 특강」, 골든래빗, 2022,

Pytorch & Colab

▪ Pytorch 실습

- 텐서 (tensor): NumPy의 array와 유사한 api 제공 + GPU 계산 가능

```
# GPU에 데이터 저장
data = [[1, 2], [3, 4]]

my_tensor = torch.tensor(data)  # 현재는 CPU에 저장
my_tensor_gpu = torch.tensor(data, device = "cuda")

print(my_tensor)
print()
print(my_tensor_gpu)
```

✓ 0.0s

```
tensor([[1, 2],
        [3, 4]])

tensor([[1, 2],
        [3, 4]], device='cuda:0')
```

이종민, 「텐초의 파이토치 딥러닝 특강」, 골든래빗, 2022,

Pytorch & Colab

▪ Pytorch 실습

- 텐서 (tensor): NumPy의 array와 유사한 api 제공 + GPU 계산 가능

```
device_cpu = torch.device("cpu")
device_gpu = torch.device("cuda" if torch.cuda.is_available() else "cpu")

print("GPU available:", torch.cuda.is_available())

N = 10000

a_cpu = torch.randn(N, N, device=device_cpu)
b_cpu = torch.randn(N, N, device=device_cpu)

a_gpu = a_cpu.to(device_gpu)
b_gpu = b_cpu.to(device_gpu)

start = time.time()
c_cpu = torch.matmul(a_cpu, b_cpu)
end = time.time()

print("CPU time:", end - start)

torch.cuda.synchronize()

start = time.time()
c_gpu = torch.matmul(a_gpu, b_gpu)
torch.cuda.synchronize()
end = time.time()

print("GPU time:", end - start)
```

```
CPU time: 26.4783556461334 23
GPU time: 0.7940359115600586
```

이종민, 「텐초의 파이토치 딥러닝 특강」, 골든래빗, 2022,

▪ Pytorch 실습

- Random tensor

```
# 랜덤 수 추출
# 0 ~ 1 사이의 uniform dist.
t1 = torch.rand(3, 4)
print(t1)

# 평균 0, 표준편차 1의 normal dist.
t2 = torch.randn(3, 4)
print(t2)

# 1 ~ 5 사이의 랜덤 정수
t3 = torch.randint(1, 5, size=((3, 4)))
print(t3)
```

✓ 0.0s

```
tensor([[0.8025, 0.2862, 0.6644, 0.7825],
        [0.4079, 0.6928, 0.4896, 0.2173],
        [0.7089, 0.2987, 0.0536, 0.8324]])
tensor([[ 1.3579, -0.1727, -0.6199,  0.3728],
        [-0.4731, -0.6131,  2.5039,  0.2900],
        [ 0.5445, -0.0106, -1.1620,  0.6384]])
tensor([[3, 3, 1, 1],
        [1, 3, 3, 2],
        [3, 1, 1, 2]])
```

이종민, 「텐초의 파이토치 딥러닝 특강」, 골든래빗, 2022,

▪ Pytorch 실습

- 이외의 NumPy와 같은 기능

```
# 이외의 numpy와 동일한 기능
t1 = torch.arange(10)
print(t1)

t2 = torch.linspace(0, 10, 5)
print(t2)

t3 = torch.eye(4)   # 대신 정사각행렬만
print(t3)
```

✓ 0.0s

```
tensor([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
tensor([ 0.0000,  2.5000,  5.0000,  7.5000, 10.0000])
tensor([[1., 0., 0., 0.],
        [0., 1., 0., 0.],
        [0., 0., 1., 0.],
        [0., 0., 0., 1.]])
```

▪ Pytorch 실습

- Broadcast도 NumPy와 똑같이 지원

```
# broadcast
x = torch.tensor([1, 2, 3, 4, 5])
y = torch.tensor([6, 7, 8, 9, 10])

print(x + y)
print(x - y)
print(x * y)
print(x / y)
```

✓ 0.0s

```
tensor([ 7,  9, 11, 13, 15])
tensor([-5, -5, -5, -5, -5])
tensor([ 6, 14, 24, 36, 50])
tensor([0.1667, 0.2857, 0.3750, 0.4444, 0.5000])
```

▪ Pytorch 실습

- 행렬 곱 (Matrix Multiplication)

```
# 행렬 곱
x = torch.tensor([[1, 2], [3, 4]])
y = torch.tensor([[5, 6], [7, 8]])

print(x @ y)
print(torch.matmul(x, y))
# print(torch.dot(x, y))   # 1차원 내적일때만 사용
```

✓ 0.0s

```
tensor([[19, 22],
        [43, 50]])
tensor([[19, 22],
        [43, 50]])
```

▪ Pytorch 실습

- 기초 통계

```
# 기초 통계
data = [[1, 2], [3, 4]]

my_tensor = torch.tensor(data, dtype=float)
print(f"Sum: {my_tensor.sum()}")
print(f"Average: {my_tensor.mean()}")
print(f"Max: {my_tensor.max()}")
print(f"Min: {my_tensor.min()}")
```

✓ 0.0s

```
Sum: 10.0
Average: 2.5
Max: 4.0
Min: 1.0
```

▪ Pytorch 실습

- masking

```
x = torch.arange(1, 10)
print(x)

# 텐서 재배열
print(x.reshape((3, 3)))    # 3 * 3 tensor로 변경

# masking
x = x.reshape((3, 3))
mask = x > 5
print(x[mask])
# mask에 걸리면 0으로 변경
x[mask] = 0
print(x)
```

✓ 0.0s

```
tensor([1, 2, 3, 4, 5, 6, 7, 8, 9])
tensor([[1, 2, 3],
        [4, 5, 6],
        [7, 8, 9]])
tensor([6, 7, 8, 9])
tensor([[1, 2, 3],
        [4, 5, 0],
        [0, 0, 0]])
```

이종민, 「텐초의 파이토치 딥러닝 특강」, 골든래빗, 2022,

Q&A

▪ Pytorch 실습

- 차원 조작
- unsqueeze(늘릴 차원): 해당 차원을 사이즈 1로 추가

```
# 차원 조작 (unsqueeze)
x = torch.arange(12).reshape((3, 4))
print(x)

# unsqueeze-> 해당 축의 차원 늘림
print(f"unsqueeze(0): {x.unsqueeze(0).shape}")
print(f"unsqueeze(1): {x.unsqueeze(1).shape}")
print(f"unsqueeze(2): {x.unsqueeze(2).shape}")
```

✓ 0.0s

```
tensor([[ 0,  1,  2,  3],
        [ 4,  5,  6,  7],
        [ 8,  9, 10, 11]])
unsqueeze(0): torch.Size([1, 3, 4])
unsqueeze(1): torch.Size([3, 1, 4])
unsqueeze(2): torch.Size([3, 4, 1])
```

▪ Pytorch 실습

- 차원 조작
- squeeze( ): 사이즈 1인 차원 제거

```
# 차원 조작 (squeeze)
x = torch.arange(12).reshape((1, 3, 4, 1))
print(x)

t = x.squeeze()
print(t)
print(t.shape)
```

✓ 0.0s

```
tensor([[[[ 0],
          [ 1],
          [ 2],
          [ 3]],

         [[ 4],
          [ 5],
          [ 6],
          [ 7]],

         [[ 8],
          [ 9],
          [10],
          [11]]]])
tensor([[ 0,  1,  2,  3],
        [ 4,  5,  6,  7],
        [ 8,  9, 10, 11]])
torch.Size([3, 4])
```

▪ Pytorch 실습

- Auto Gradient: 기울기 자동 계산

```
# Auto Gradient
x = torch.tensor([2], dtype=float, requires_grad=True)
y = torch.tensor([6], dtype=float, requires_grad=True)

Q = 3 * x ** 3 - y ** 2
# 역전파 계산
Q.backward()

# 기울기 계산
# dQ/dx = 9 * x ** 2 = 36
print(f"x.gradient: {x.grad}")
# dQ/dy = - 2 * y = -12
print(f"y.gradient: {y.grad}")
```

✓ 0.0s

```
x.gradient: tensor([36.], dtype=torch.float64)
y.gradient: tensor([-12.], dtype=torch.float64)
```

이종민, 「텐초의 파이토치 딥러닝 특강」, 골든래빗, 2022,

▪ Pytorch 실습

- 매 backpropagation 시행 시 이전에 계산한 gradient 초기화

```
# Auto Gradient는 기울기가 누적
# (Optimizer가) 매 Gradient 계산 시, 계산 값 초기화 해야함
# grad.zero_(): Gradient 초기화

x = torch.tensor([2], dtype=float, requires_grad=True)
y = torch.tensor([6], dtype=float, requires_grad=True)

Q = 3 * x ** 3 - y ** 2 # 예측 모델 Q
Q.backward() # 역전파 계산
print(f"x.gradient: {x.grad}") # 36

# 2번 계산 시
Q = 3 * x ** 3 - y ** 2
Q.backward()
print(f"x.gradient: {x.grad}") # 36 + 36 = 72 (누적)

# 기울기 초기화
x.grad.zero_()
Q = 3 * x ** 3 - y ** 2
Q.backward()
print(f"x.gradient: {x.grad}") # 36
```

✓ 0.0s

```
x.gradient: tensor([36.], dtype=torch.float64)
x.gradient: tensor([72.], dtype=torch.float64)
x.gradient: tensor([36.], dtype=torch.float64)
```

Pytorch & Colab

▪ Pytorch 실습

- 선형 회귀 구현

0. Load Module & Setting

```
import torch
import torch.nn as nn
import torch.optim as optim
import matplotlib.pyplot as plt
```

✓ 1.6s

```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print("Using device:", device)

torch.manual_seed(42) # 시드 고정
```

✓ 0.0s

```
Using device: cuda

<torch._C.Generator at 0x7936b773ef70>
```

Pytorch & Colab

▪ Pytorch 실습

- 선형 회귀 구현

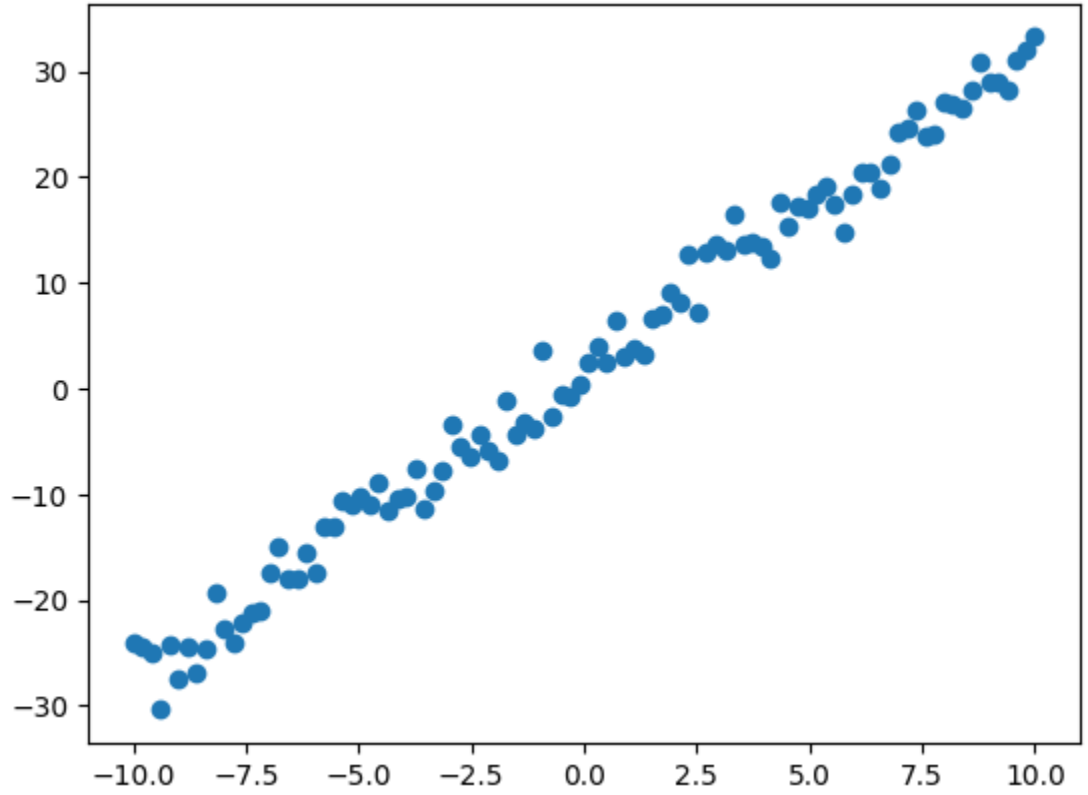
1. Data Preparation

```
X = torch.linspace(-10, 10, 100).unsqueeze(1)  # (100, 1)
y = 3 * X + 2 + torch.randn_like(X) * 2        # 노이즈 추가
```

✓ 0.0s

```
X = X.to(device)     # GPU로 데이터 이동
y = y.to(device)     # GPU로 데이터 이동
```

✓ 0.3s



이종민, 「텐초의 파이토치 딥러닝 특강」, 골든래빗, 2022,

Pytorch & Colab

▪ Pytorch 실습

- 선형 회귀 구현

3. Model Define

```
model = nn.Linear(1, 1).to(device)  # 선형 모델 y = w*x + b
loss_func = nn.MSELoss()    # 손실함수 L = (y - y_pred) ** 2
optimizer = optim.SGD(model.parameters(), lr=0.01)  # 기울기 optimizer -> stochastic GD
```

✓ 0.9s

▪ Pytorch 실습

- 선형 회귀 구현

4. Train

```
epochs = 200    # 학습 반복할 횟수 (에포크)

loss_history = []

for epoch in range(epochs):
    y_pred = model(X)   # 모델을 거친 예측값 저장
    loss = loss_func(y_pred, y) # loss 계산


    optimizer.zero_grad()   # 기울기 초기화
    loss.backward()         # 역전파
    optimizer.step()        # 가중치 업데이트

    loss_history.append(loss.item())

    if (epoch + 1) % 20 == 0:
        print(f"Epoch [{epoch+1}/{epochs}], Loss: {loss.item():.4f}")
```

✓ 0.2s

```
Epoch [20/200], Loss: 8.0268
Epoch [40/200], Loss: 5.7125
Epoch [60/200], Loss: 4.6810
Epoch [80/200], Loss: 4.2213
Epoch [100/200], Loss: 4.0164
Epoch [120/200], Loss: 3.9250
Epoch [140/200], Loss: 3.8843
Epoch [160/200], Loss: 3.8662
Epoch [180/200], Loss: 3.8581
Epoch [200/200], Loss: 3.8545
```

```
w = model.weight.item()
b = model.bias.item()

print("\nLearned parameters:")
print("w =", w)
print("b =", b)
```

✓ 0.0s

```
Learned parameters:
w = 2.9941048622131348
b = 2.066777467727661
```

Pytorch & Colab

▪ Pytorch 실습

- 선형 회귀 구현

5. Inference

```
model.eval()   # 모델의 평가 모드 (모델 구조 업데이트 X)

with torch.no_grad():   # gradient 계산 비활성화
    # 새 데이터 (test data)
    x_test = torch.tensor([[4.0]]).to(device)
    # 추론
    y_inference = model(x_test)

    print(f"\nInput: {x_test.item()}")
    print(f"Predicted Output: {y_inference.item():.4f}")
```

✓ 0.0s

```
Input: 4.0
Predicted Output: 14.0432
```

이종민, 「텐초의 파이토치 딥러닝 특강」, 골든래빗, 2022,

Q&A