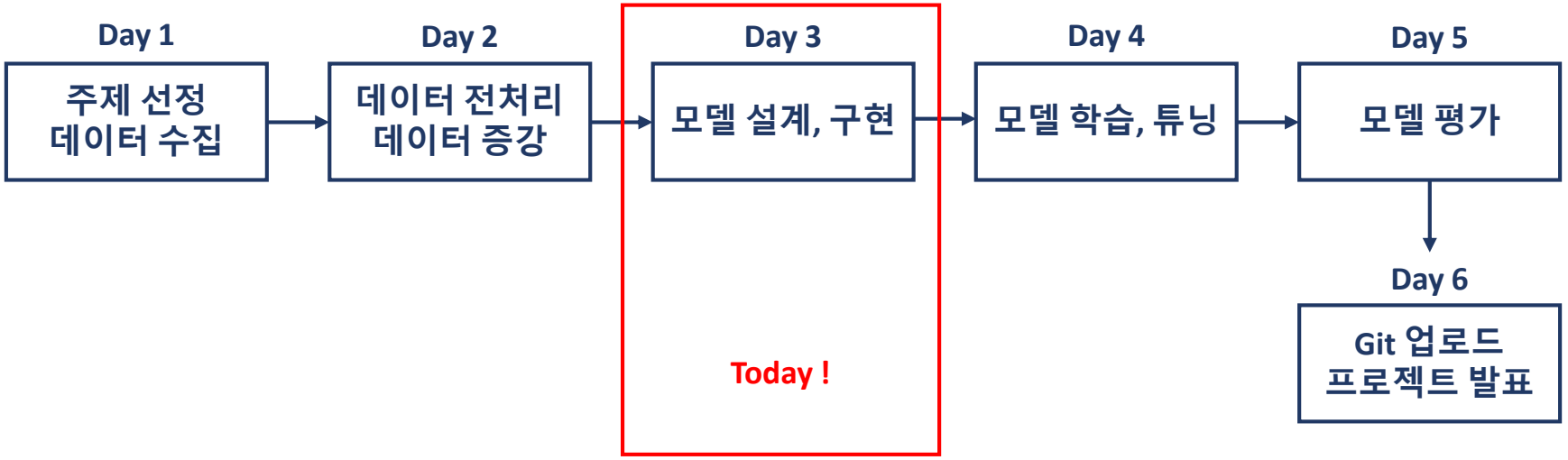
AI 기반 영상 데이터 분석 실습
- Day 3 -

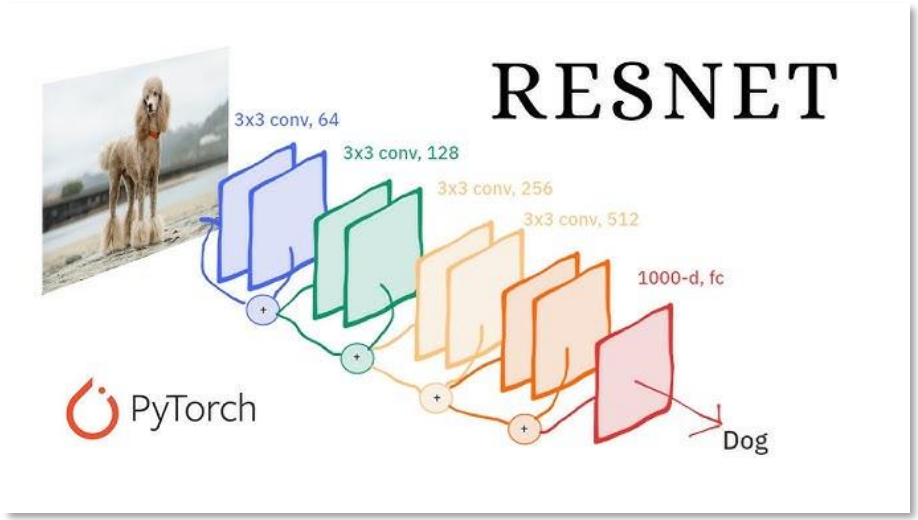
index

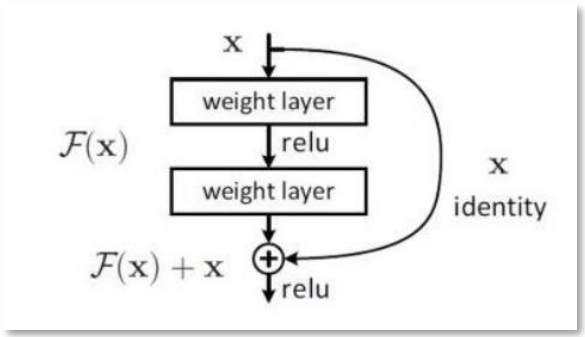

Day 1
주제 선정
데이터 수집
Day 2
데이터 전처리
데이터 증강
Day 3
모델 설계, 구현
Today !
Day 4
모델 학습, 튜닝
Day 5
모델 평가
Day 6
Git 업로드
프로젝트 발표

Afternoon Lab session (실습)

Day 3 : Course overview

- 학습시킬 모델(ResNet) 아키텍쳐 구현

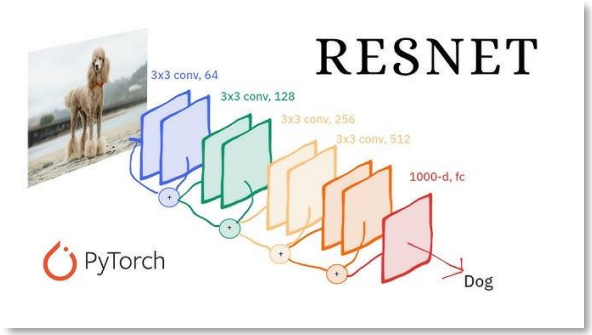


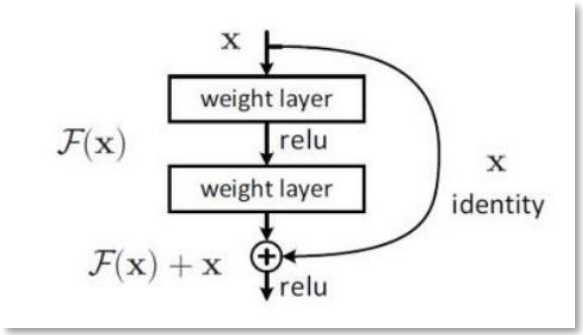


Afternoon Lab session (실습)

ResNet이란?

- CNN 기반의 딥러닝 모델 중 **가장 핵심적인 모델.**
- Residual Block의 Skip connection 구조로 gradient vanishing 문제를 극복하여 **100층 이상의 깊은 신경망을 학습**시킬 수 있는 길을 열음.
- 2015년도에 발표된 딥러닝 모델임에도 불구하고 현재까지도 **많은 연구들의 Baseline**으로 자주 언급됨.





Residual Block

Afternoon Lab session (실습)

코드 실습

- **def __init__(self) :**
  - 필요한 layer를 선언하는 공간

- **def forward(self) :**
  - 위에서 선언한 layer를 바탕으로 연산 흐름을 정의

```python
class BasicBlock(nn.Module):
    expansion_factor = 1
    def __init__(self, in_channels: int, out_channels: int, stride: int = 1):
        super(BasicBlock, self).__init__()
        self.conv1 = nn.Conv2d(in_channels, out_channels, kernel_size=3, stride=stride, padding=1, bias=False)
        self.bn1 = nn.BatchNorm2d(out_channels)
        self.relu1 = nn.ReLU()

        # ...

    def forward(self, x)
        out = x
        x = self.conv1(x)
        x = self.bn1(x)
        x = self.relu1(x)

        # ...

        return x
```

코드 예시

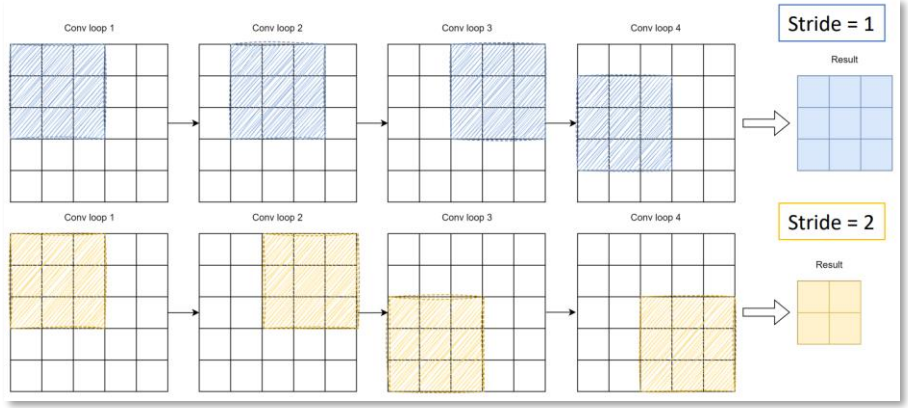
Afternoon Lab session (실습)

Layer 구성 모듈

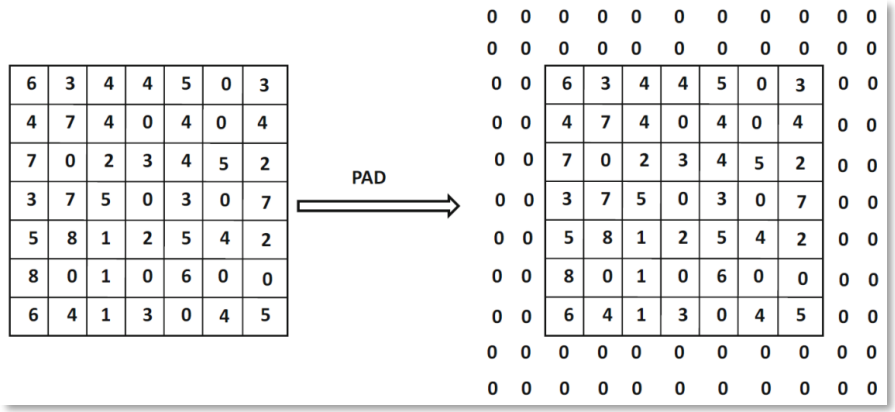
- **nn.Conv2d(in_channels, out_channels, kernel_size, stride, padding)**
  - in_channels : input의 channel 수
  - out_channels : output의 channel 수
  - kernel_size : convolution kernel size, (예시 – 3 or (3, 3))
  - stride : convolution 연산 간격
  - padding : 경계를 몇 칸까지 0으로 채울 지, (matrix 사이즈 보존의 역할)

```python
class BasicBlock(nn.Module):
    expansion_factor = 1
    def __init__(self, in_channels: int, out_channels: int, stride: int = 1):
        super(BasicBlock, self).__init__()
        self.conv1 = nn.Conv2d(in_channels, out_channels, kernel_size=3, stride=stride, padding=1, bias=False)
        self.bn1 = nn.BatchNorm2d(out_channels)
        self.relu1 = nn.ReLU()

        # ...

    def forward(self, x)
        out = x
        x = self.conv1(x)
        x = self.bn1(x)
        x = self.relu1(x)

        # ...

        return x
```

코드 예시



Stride



Padding

Afternoon Lab session (실습)

Layer 구성 모듈

- **nn.Sequential()**
  - **Layer를 연속적으로 한 번에 선언할 수 있는 함수**
- **nn.BatchNorm2d(channels)**
  - **Conv2d 뒤에 붙는 Batch Normalization 함수**
- **nn.ReLU()**
  - **Activation function**

```python
class BasicBlock(nn.Module):
    expansion_factor = 1
    def __init__(self, in_channels: int, out_channels: int, stride: int = 1):
        super(BasicBlock, self).__init__()
        self.conv1 = nn.Conv2d(in_channels, out_channels, kernel_size=3, stride=stride, padding=1, bias=False)
        self.bn1 = nn.BatchNorm2d(out_channels)
        self.relu1 = nn.ReLU()

        # ...

    def forward(self, x)
        out = x
        x = self.conv1(x)
        x = self.bn1(x)
        x = self.relu1(x)

        # ...

        return x
```

코드 예시

```python
class BasicBlock(nn.Module):
    def __init__(self, in_channels: int, out_channels: int, stride: int = 1):

        self.my_block = nn.Sequential(
            nn.Conv2d(in_channels, out_channels, kernel_size=3, stride=stride, padding=1, bias=False),
            nn.BatchNorm2d(out_channels),
            nn.ReLU()
        )

        # ...

    def forward(self, x):
        out = x
        x = self.conv1(x)
        x = self.bn1(x)
        x = self.relu1(x)

        # ...

        return x
```

- nn.Sequential() 활용

Afternoon Lab session (실습)

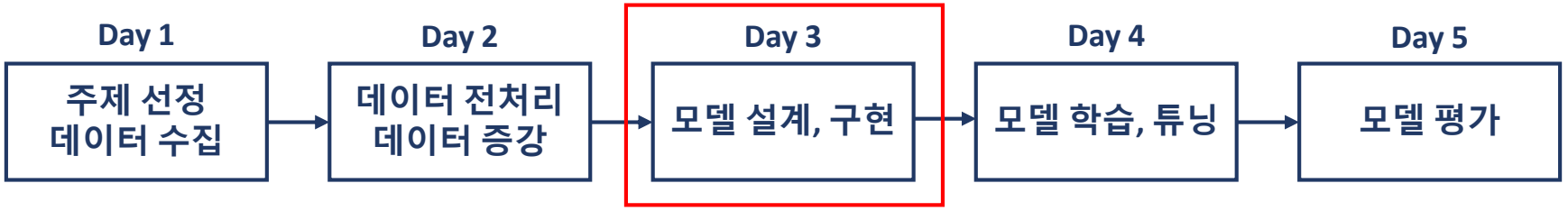
실습

- **목표 : 최종 모델 구현 후, Summary 정상 출력**

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

# 모델 선언
model = ResNet().to(device)

# 완성한 모델의 summary를 출력해보세요.
summary(model, (3, 224, 224))
```

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 64, 112, 112]           9,408
       BatchNorm2d-2         [-1, 64, 112, 112]             128
              ReLU-3         [-1, 64, 112, 112]               0
         MaxPool2d-4           [-1, 64, 56, 56]               0
            Conv2d-5           [-1, 64, 56, 56]          36,864
       BatchNorm2d-6           [-1, 64, 56, 56]             128
              ReLU-7           [-1, 64, 56, 56]               0
            Conv2d-8           [-1, 64, 56, 56]          36,864
       BatchNorm2d-9           [-1, 64, 56, 56]             128
             ReLU-10           [-1, 64, 56, 56]               0
       BasicBlock-11           [-1, 64, 56, 56]               0
            Conv2d-12           [-1, 64, 56, 56]          36,864
       BatchNorm2d-13           [-1, 64, 56, 56]             128
             ReLU-14           [-1, 64, 56, 56]               0
            Conv2d-15           [-1, 64, 56, 56]          36,864
       BatchNorm2d-16           [-1, 64, 56, 56]             128
             ReLU-17           [-1, 64, 56, 56]               0
       BasicBlock-18           [-1, 64, 56, 56]               0
            Conv2d-19          [-1, 128, 28, 28]          73,728
       BatchNorm2d-20          [-1, 128, 28, 28]             256
             ReLU-21          [-1, 128, 28, 28]               0
            Conv2d-22          [-1, 128, 28, 28]         147,456
       BatchNorm2d-23          [-1, 128, 28, 28]             256
            Conv2d-24          [-1, 128, 28, 28]           8,192
       BatchNorm2d-25          [-1, 128, 28, 28]             256
             ReLU-26          [-1, 128, 28, 28]               0
       BasicBlock-27          [-1, 128, 28, 28]               0
            Conv2d-28          [-1, 128, 28, 28]         147,456
       BatchNorm2d-29          [-1, 128, 28, 28]             256
             ReLU-30          [-1, 128, 28, 28]               0
            Conv2d-31          [-1, 128, 28, 28]         147,456
       BatchNorm2d-32          [-1, 128, 28, 28]             256
             ReLU-33          [-1, 128, 28, 28]               0
       BasicBlock-34          [-1, 128, 28, 28]               0
            Conv2d-35          [-1, 256, 14, 14]         294,912
       BatchNorm2d-36          [-1, 256, 14, 14]             512
             ReLU-37          [-1, 256, 14, 14]               0
            Conv2d-38          [-1, 256, 14, 14]         589,824
       BatchNorm2d-39          [-1, 256, 14, 14]             512
            Conv2d-40          [-1, 256, 14, 14]          32,768
       BatchNorm2d-41          [-1, 256, 14, 14]             512
             ReLU-42          [-1, 256, 14, 14]               0
       BasicBlock-43          [-1, 256, 14, 14]               0
            Conv2d-44          [-1, 256, 14, 14]         589,824
       BatchNorm2d-45          [-1, 256, 14, 14]             512
             ReLU-46          [-1, 256, 14, 14]               0
            Conv2d-47          [-1, 256, 14, 14]         589,824
       BatchNorm2d-48          [-1, 256, 14, 14]             512
             ReLU-49          [-1, 256, 14, 14]               0
       BasicBlock-50          [-1, 256, 14, 14]               0
            Conv2d-51            [-1, 512, 7, 7]       1,179,648
       BatchNorm2d-52            [-1, 512, 7, 7]           1,024
             ReLU-53            [-1, 512, 7, 7]               0
            Conv2d-54            [-1, 512, 7, 7]       2,359,296
       BatchNorm2d-55            [-1, 512, 7, 7]           1,024
            Conv2d-56            [-1, 512, 7, 7]         131,072
       BatchNorm2d-57            [-1, 512, 7, 7]           1,024
             ReLU-58            [-1, 512, 7, 7]               0
       BasicBlock-59            [-1, 512, 7, 7]               0
            Conv2d-60            [-1, 512, 7, 7]       2,359,296
       BatchNorm2d-61            [-1, 512, 7, 7]           1,024
             ReLU-62            [-1, 512, 7, 7]               0
            Conv2d-63            [-1, 512, 7, 7]       2,359,296
       BatchNorm2d-64            [-1, 512, 7, 7]           1,024
             ReLU-65            [-1, 512, 7, 7]               0
       BasicBlock-66            [-1, 512, 7, 7]               0
AdaptiveAvgPool2d-67            [-1, 512, 1, 1]               0
            Linear-68                  [-1, 10]           5,130
================================================================
Total params: 11,181,642
Trainable params: 11,181,642
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.57
Forward/backward pass size (MB): 62.79
Params size (MB): 42.65
Estimated Total Size (MB): 106.01
----------------------------------------------------------------
```

01/21 To-do



layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2 Basic_Block [3×3, 64; 3×3, 64] ×2	[3×3, 64; 3×3, 64] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3
conv3_x	28×28	[3×3, 128; 3×3, 128] ×2	[3×3, 128; 3×3, 128] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×8
conv4_x	14×14	[3×3, 256; 3×3, 256] ×2	[3×3, 256; 3×3, 256] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×23	[1×1, 256; 3×3, 256; 1×1, 1024] ×36
conv5_x	7×7	[3×3, 512; 3×3, 512] ×2	[3×3, 512; 3×3, 512] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Kernel size

out_planes

- 유의사항
  - Convolution 이후, Batch Normalization 적용
  - convolution 연산으로 인해 size가 증가하여 shortcut 연산이 불가능할 경우, 1x1 conv, BatchNorm으로 차원 맞추기

Thank you