

## 07. Batch Normalization

**AILab**  
**Hanyang Univ.**

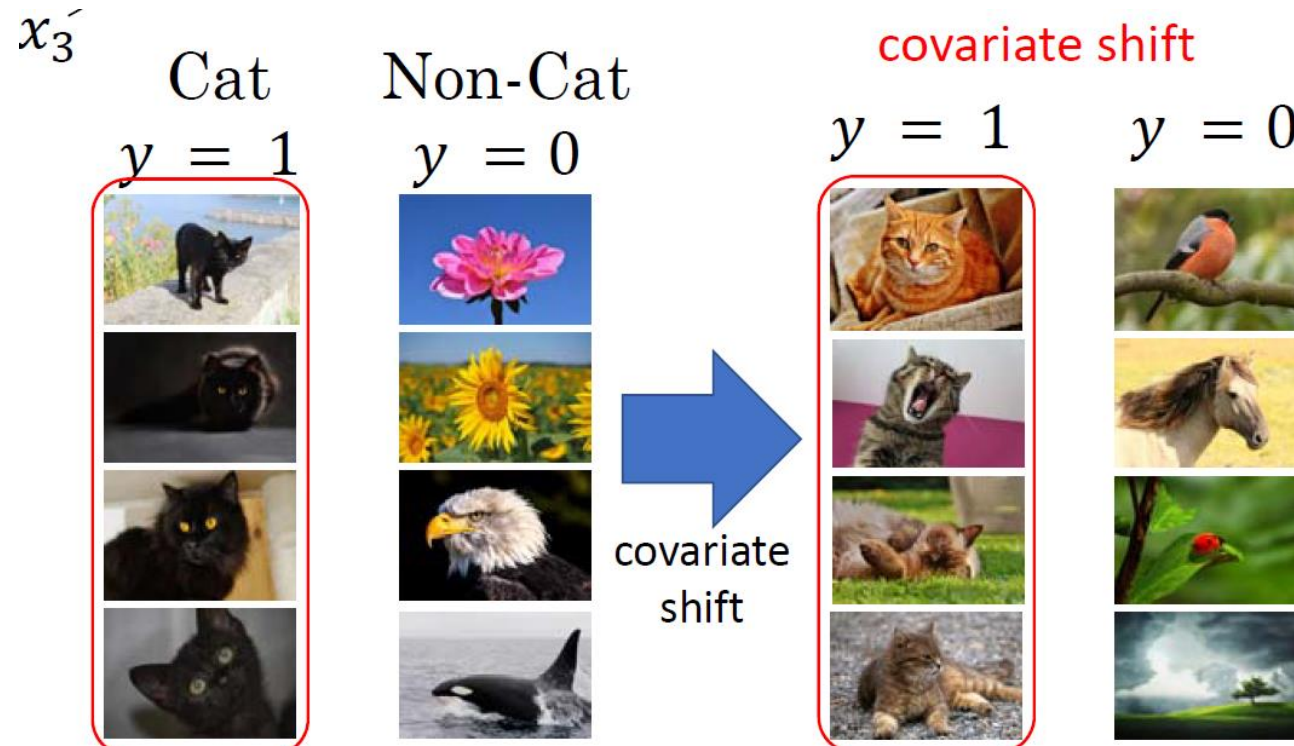
## 오늘 실습 내용

### 1. Batch Normalization

# 1. Batch Normalization

## Batch Normalization

- Covariate Shift
  - When the **input distribution** to a learning system changes, it is said to experience covariate shift

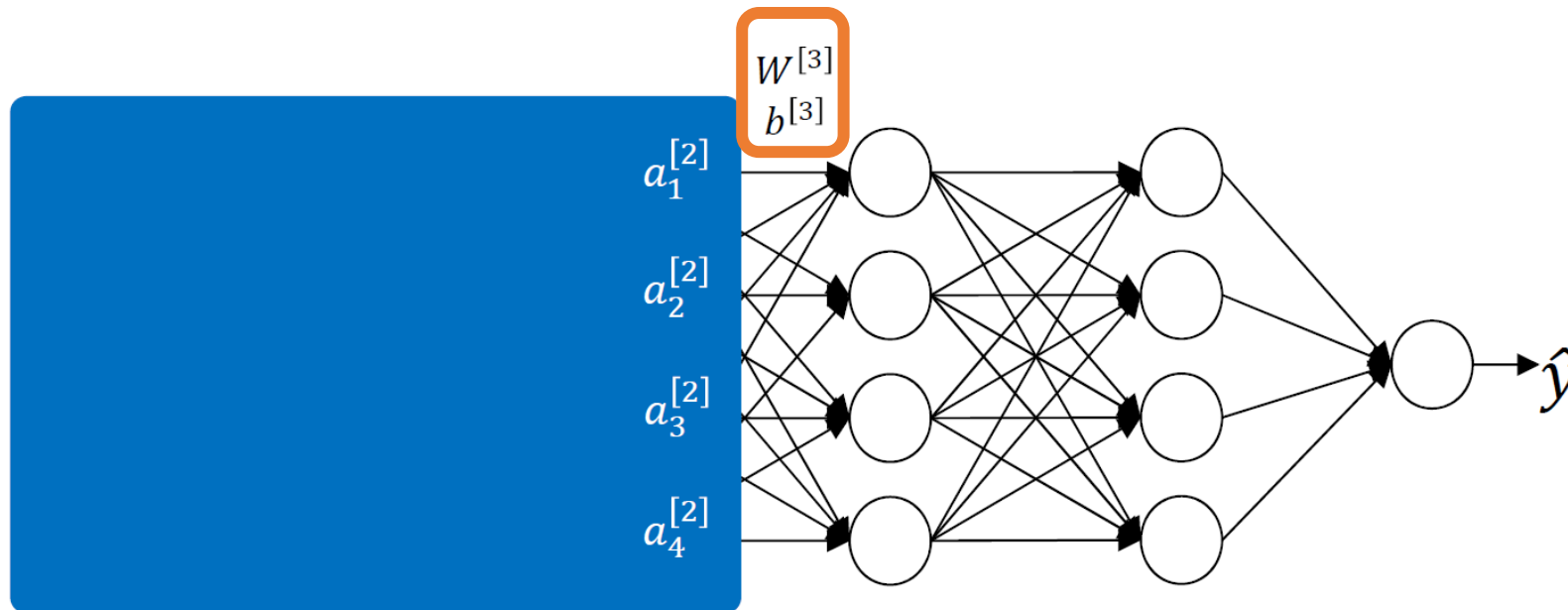


## Batch Normalization

- (Internal) Covariate Shift
  - Change in the **distribution of network activations** due to the change in **network parameters** during training.
  - 각 Layer의 input은 이전 Layer 들의 output에 영향을 받음
  - Layer가 깊어질 수록 covariate shift가 심해짐

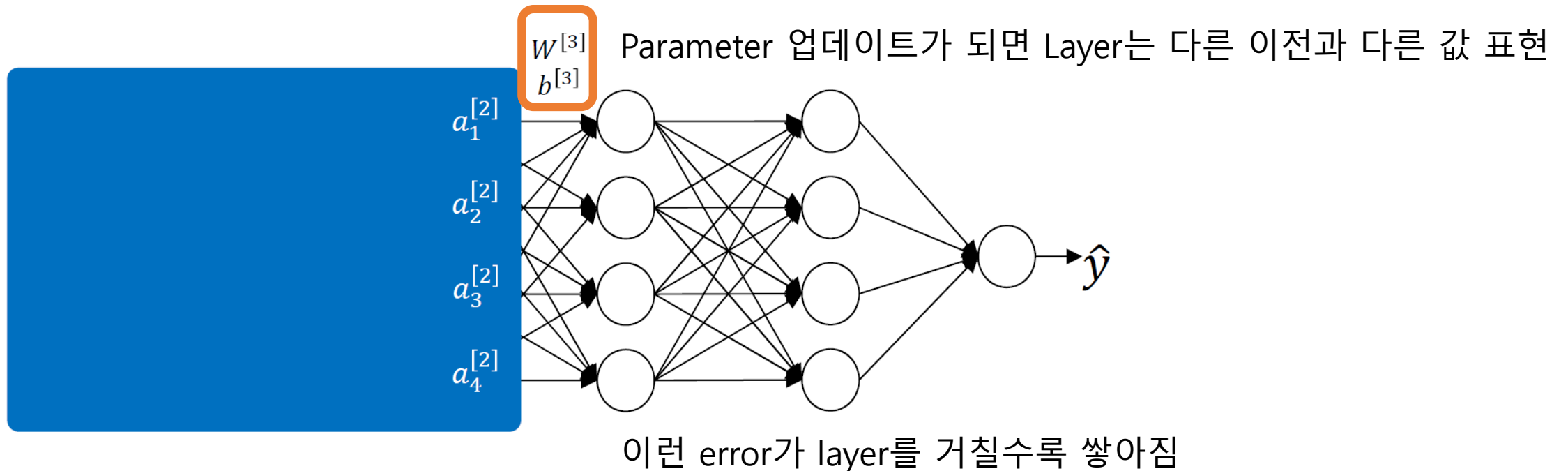
## Batch Normalization

- (Internal) Covariate Shift
  - Change in the **distribution of network activations** due to the change in **network parameters** during training.
  - 각 Layer의 input은 이전 Layer 들의 output에 영향을 받으므로 Layer가 깊어질 수록 covariate shift가 심해짐



## Batch Normalization

- (Internal) Covariate Shift
  - Change in the **distribution of network activations** due to the change in **network parameters** during training.
  - 각 Layer의 input은 이전 Layer 들의 output에 영향을 받으므로 Layer가 깊어질 수록 covariate shift가 심해짐



## Batch Normalization

- BATCHNORM1D

- <https://pytorch.org/docs/stable/generated/torch.nn.BatchNorm1d.html#torch.nn.BatchNorm1d>

`torch.nn.BatchNorm1d(num_features, eps=1e-05,` num\_features : batch normalization을 적용할 input의 dimension size    Input:  $(N, C)$  or  $(N, C, L)$

Applies Batch Normalization over a 2D or 3D input as described in the paper [Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift](#).

$$y = \frac{x - \mathbb{E}[x]}{\sqrt{\text{Var}[x] + \epsilon}} * \gamma + \beta$$

The mean and standard-deviation are calculated per-dimension over the mini-batches and  $\gamma$  and  $\beta$  are learnable parameter vectors of size  $C$  (where  $C$  is the number of features or channels of the input). By default, the elements of  $\gamma$  are set to 1 and the elements of  $\beta$  are set to 0. The standard-deviation is calculated via the biased estimator, equivalent to `torch.var(input, unbiased=False)`.



## Batch Normalization

- Model code
  - 각 layer의 activation 이전에 불러줌
  - num\_features에 올바른 값 선언

```
class Model(nn.Module):
    def __init__(self, drop_prob):
        super(Model, self).__init__()
        self.linear1 = nn.Linear(32*32*3, 256)
        self.linear2 = nn.Linear(256, 128)
        self.linear3 = nn.Linear(128, 10)
        self.dropout = nn.Dropout(drop_prob)
        self.activation = nn.Sigmoid()

        self.bn1 = nn.BatchNorm1d(256)
        self.bn2 = nn.BatchNorm1d(128)

    def forward(self, x):
        z1 = self.linear1(x)
        z1 = self.bn1(z1)
        a1 = self.activation(z1)
        a1 = self.dropout(a1)

        z2 = self.linear2(a1)
        z2 = self.bn2(z2)
        a2 = self.activation(z2)
        a2 = self.dropout(a2)

        z3 = self.linear3(a2)

        return z3
```

## Batch Normalization

- Model code
  - 각 layer의 activation 이전에 불러줌
  - num\_features에 올바른 값 선언

```
class Model(nn.Module):  
    def __init__(self, drop_prob):  
        super(Model, self).__init__()  
        self.linear1 = nn.Linear(32*32*3, 256)  
        self.linear2 = nn.Linear(256, 128)  
        self.linear3 = nn.Linear(128, 10)  
        self.dropout = nn.Dropout(drop_prob)  
        self.activation = nn.Sigmoid()
```

```
self.bn1 = nn.BatchNorm1d(256)  
self.bn2 = nn.BatchNorm1d(128)
```

```
def forward(self, x):  
    z1 = self.linear1(x)  
    z1 = self.bn1(z1)  
    a1 = self.activation(z1)  
    a1 = self.dropout(a1)  
  
    z2 = self.linear2(a1)  
    z2 = self.bn2(z2)  
    a2 = self.activation(z2)  
    a2 = self.dropout(a2)  
  
    z3 = self.linear3(a2)  
  
    return z3
```

## 오늘 실습 내용

Batch normalization을 이전 실습 코드에 적용하여 성능 차이 확인해보기