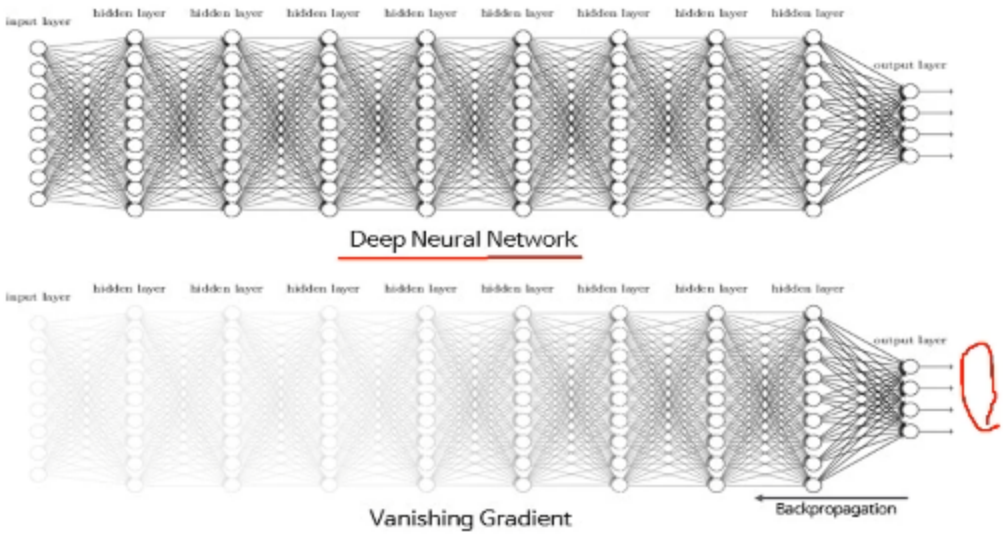
Batch Normalization

상태	DNN
담당자	

Gradient Vanishing / Exploding

Gradient Vanishing



Gradient가 앞 단으로 진행될수록 소실되는 문제

ReLu

앞서 sigmoid function의 문제점으로 gradient vanishing을 언급한 적 있었다.

Gradient Exploding

> 💡 Gradient을 진행할수록 미분값이 너무 커져버리는 문제

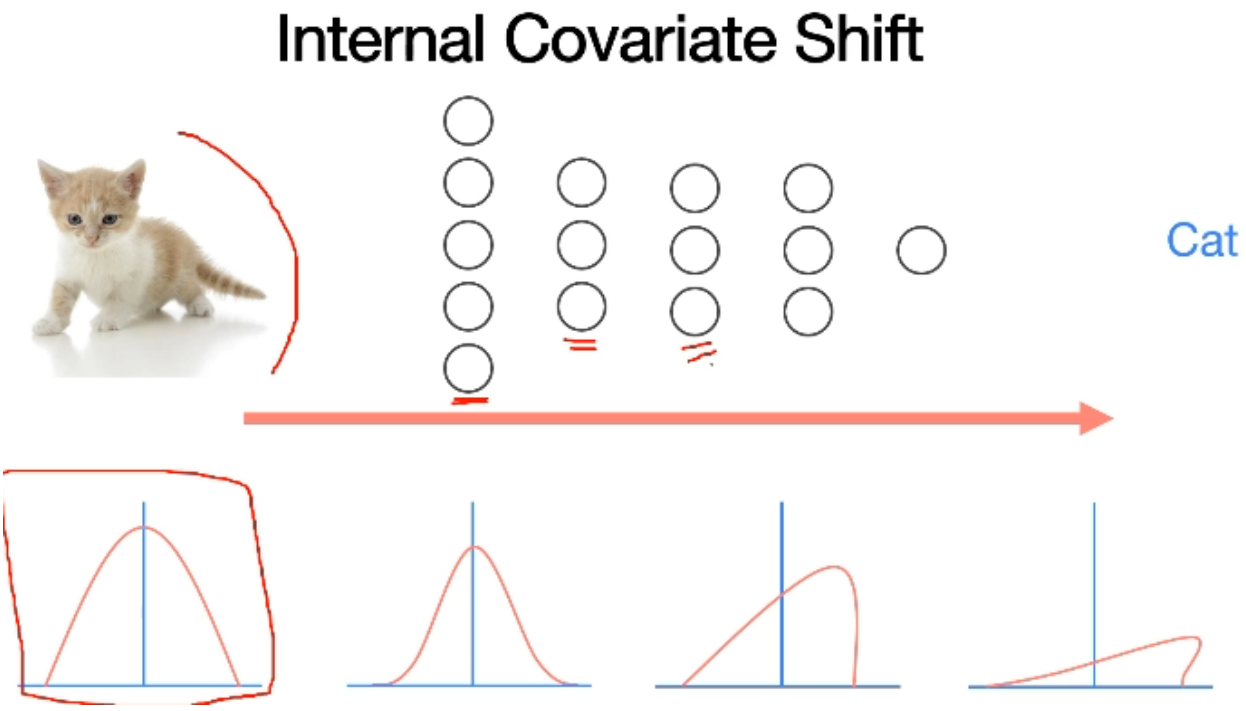
Gradient Vanishing이나 Gradient Exploding 문제가 있을 때 뉴럴 넷의 학습이 어려워진다. 이러한 문제점을 어떻게 극복해야 할까?

- Activation function 변경 (e.g. Sigmoid → ReLU)
- Weight initalization을 잘 하기 (e.g. Xavier, He)
- Learning rate를 작게 하기 → Gradient exploding 문제를 해결하는 데 도움이 될 수 있음

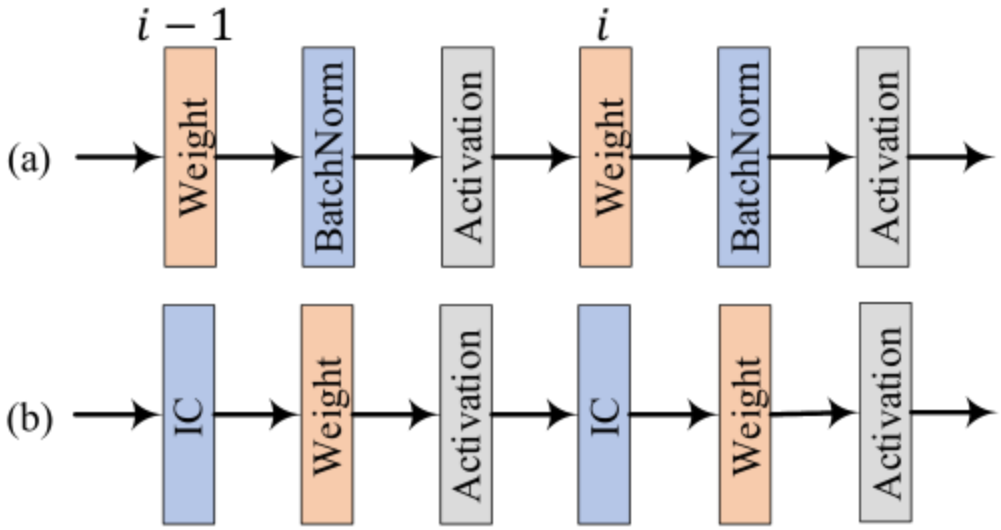
하지만 이러한 방법들은 모두 간접적인 방법이다. 보다 직접적으로 문제를 해결할 수 있는 방법이 없을까?

Batch Normalization

Weight initalization을 잘 하면 각 layer마다 활성화 값 분포가 적당히 퍼지면서 학습이 원활하게 수행되었다. 이를 강제해 볼 수는 없을까?



잘 만들어진 분포가 학습 과정에서 변화하게 된다. 그리고 이 문제는 layer가 많아질수록 심화되게 된다. 따라서, 매 미니배치의 각 레이어마다 데이터 분포를 정규화하는 ‘배치 정규화 계층’을 신경망에 삽입하고자 한다.



Batch Normalization은 다음과 같은 장점을 가지고 있다.

- 학습을 빨리 진행할 수 있다.
- 초깃값에 크게 의존하지 않는다.
- 오버피팅을 억제한다 (dropout의 필요성 감소).

```
class dnn_batchnorm(nn.Module):
    def __init__(self):
        super().__init__()
        self.linear1 = nn.Linear(784, 32, bias=True).to(device)
        self.linear2 = nn.Linear(32, 32, bias=True).to(device)
        self.linear3 = nn.Linear(32, 10, bias=True).to(device)
        self.relu = nn.ReLU()
        self.bn1 = nn.BatchNorm1d(32)
        self.bn2 = nn.BatchNorm1d(32)

        nn.init.xavier_normal_(self.linear1.weight)
        nn.init.xavier_normal_(self.linear2.weight)
        nn.init.xavier_normal_(self.linear3.weight)

        self.model = nn.Sequential(self.linear1, self.bn1, self.relu,
                                   self.linear2, self.bn2, self.relu,
                                   self.linear3).to(device)

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

일반적으로 batch normalization은 activation function 이전에 사용해 준다. 여기에서도 train 할 때에는 `model.train()`, 테스트할 때 `model.eval()`을 달고 하는 것을 잊지 말자.