

꼼꼼한 딥러닝 논문 리뷰와 코드 실습

Deep Learning Paper Review and Code Practice

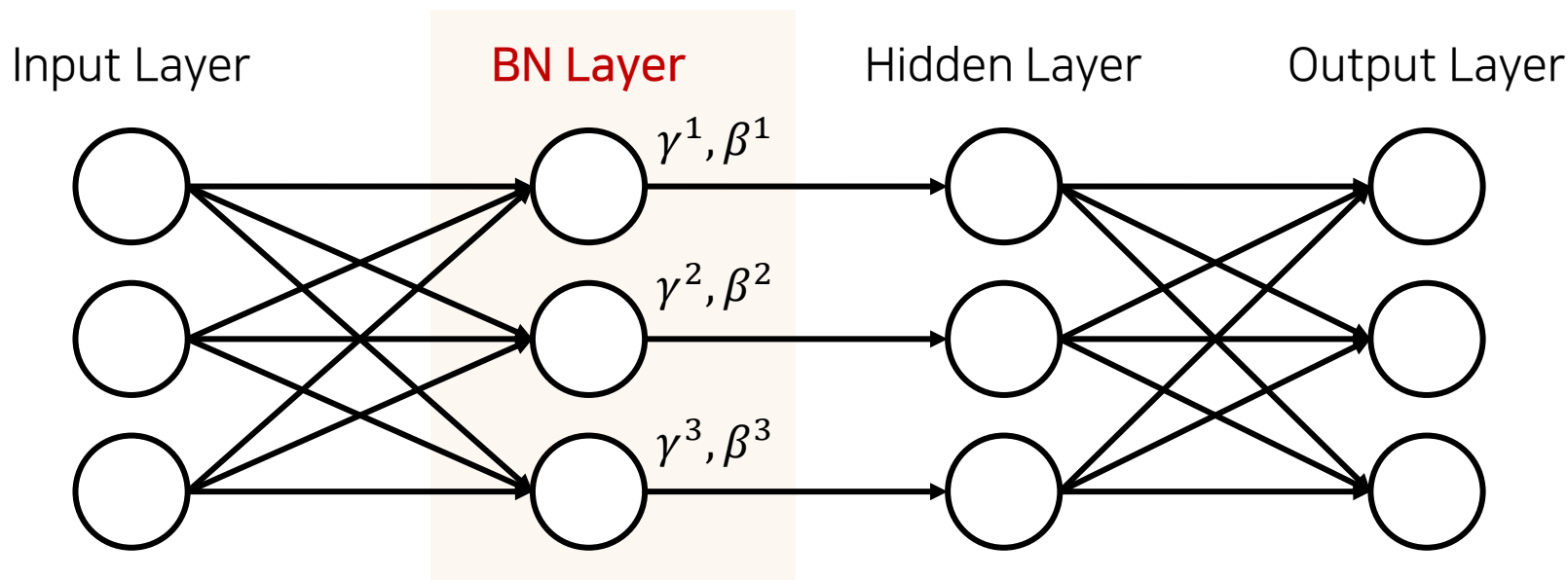
나동빈(dongbinna@postech.ac.kr)

Pohang University of Science and Technology

배치 정규화 (Batch Normalization)

배치 정규화(Batch Normalization)

- 배치 정규화의 잘 알려진 **장점**은 다음과 같습니다.
 - ① 학습 속도(training speed)를 빠르게 할 수 있습니다.
 - ② 가중치 초기화(weight initialization)에 대한 민감도를 감소시킵니다.
 - ③ 모델의 일반화(regularization) 효과가 있습니다.

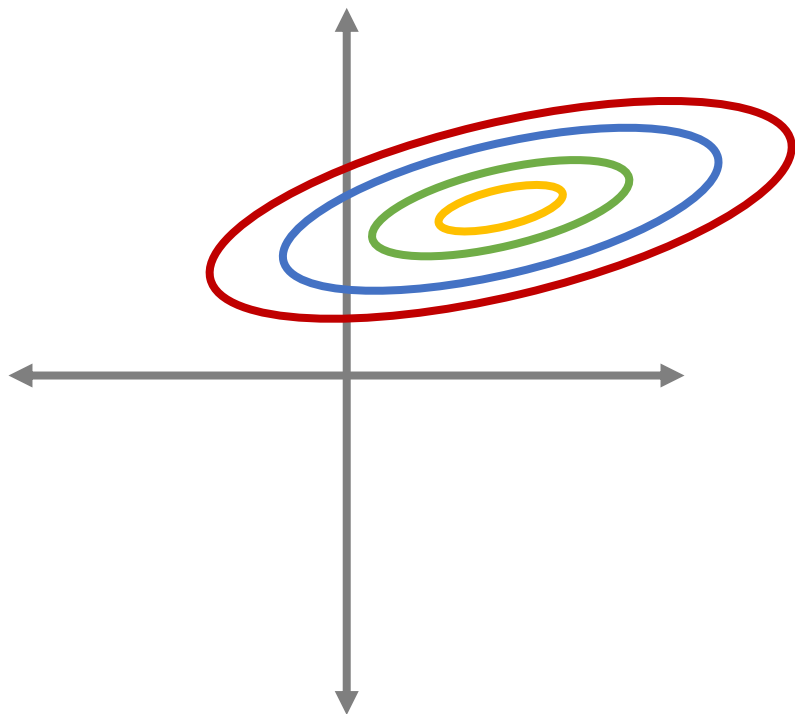


Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift (PMLR 2015)

관련 연구: 입력 정규화(Normalization)

- 입력 데이터를 정규화하여 학습 속도(training speed)를 개선할 수 있습니다.

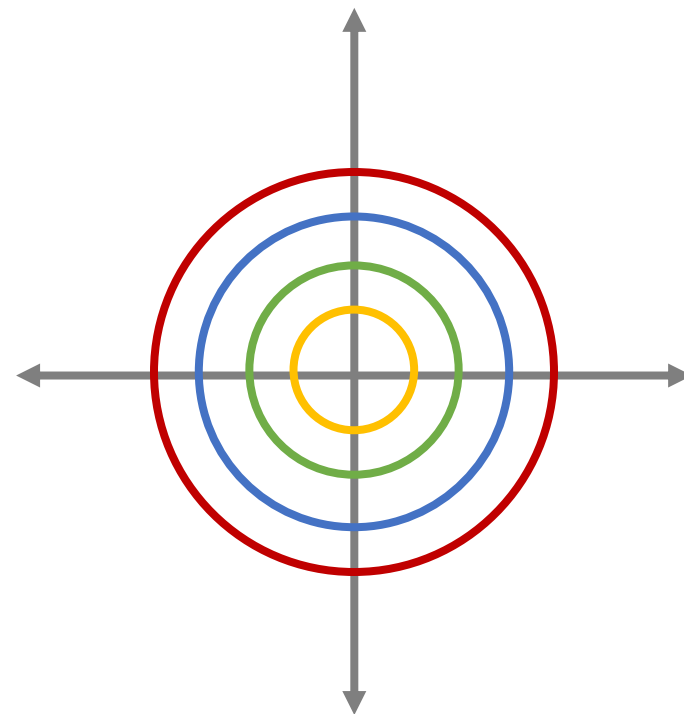
[정규화 전]



Large Learning Rate 사용 **어려움**



[정규화 후]



Large Learning Rate 사용 **가능**

관련 연구: 입력 표준화(Standardization)

- 입력 데이터를 $N(0, 1)$ 분포를 따르도록 표준화하는 예제는 다음과 같습니다.

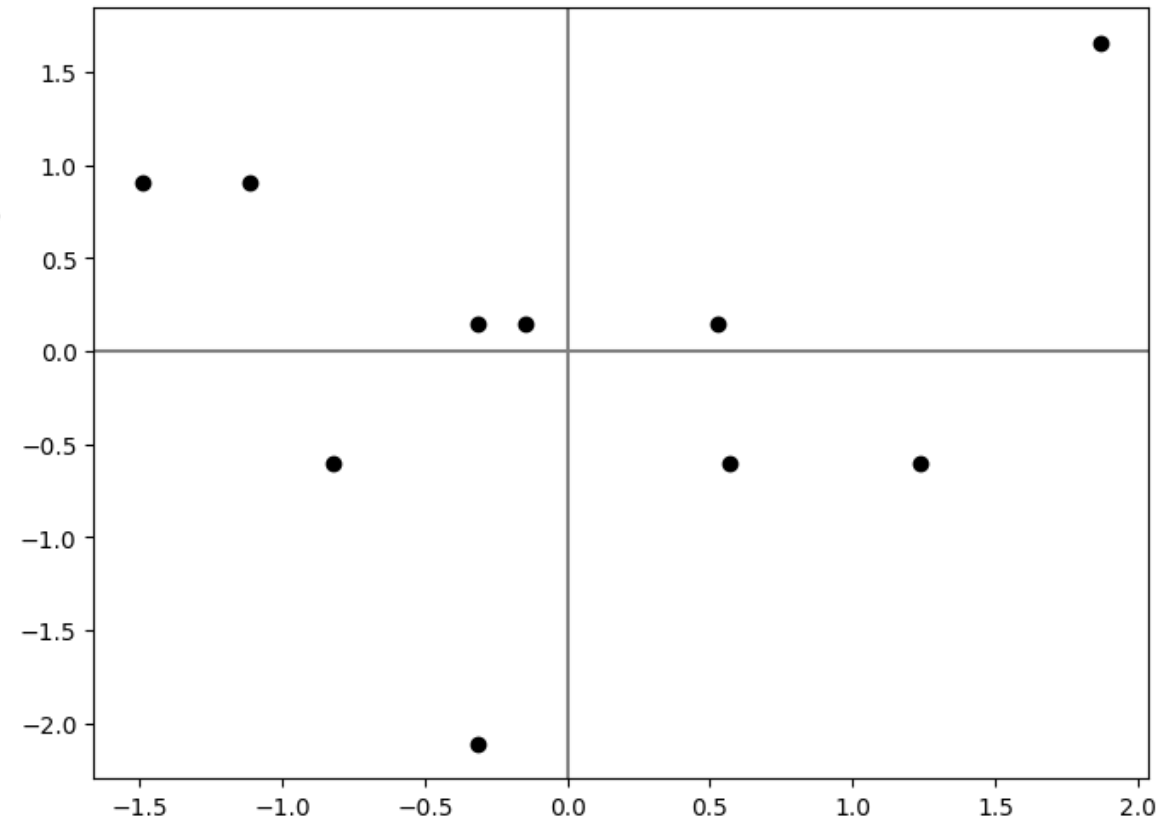
$$\hat{x} = \frac{x - E[x]}{\sqrt{Var[x]}}$$

```
x1 = np.asarray([33, 72, 40, 104, 52, 56, 89, 24, 52, 73])
x2 = np.asarray([9, 8, 7, 10, 5, 8, 7, 9, 8, 7])
```

```
normalized_x1 = (x1 - np.mean(x1)) / np.std(x1)
normalized_x2 = (x2 - np.mean(x2)) / np.std(x2)
```

```
plt.axvline(x=0, color='gray')
plt.axhline(y=0, color='gray')
plt.scatter(normalized_x1, normalized_x2, color='black')
plt.show()
```

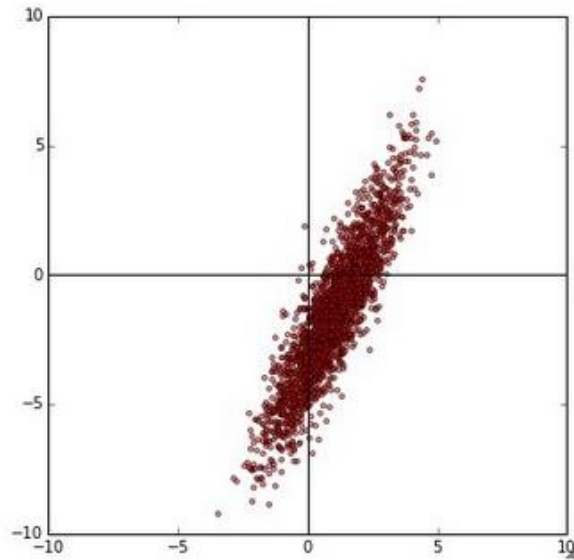
[평균(mean) = 0, 분산(variance) = 1]



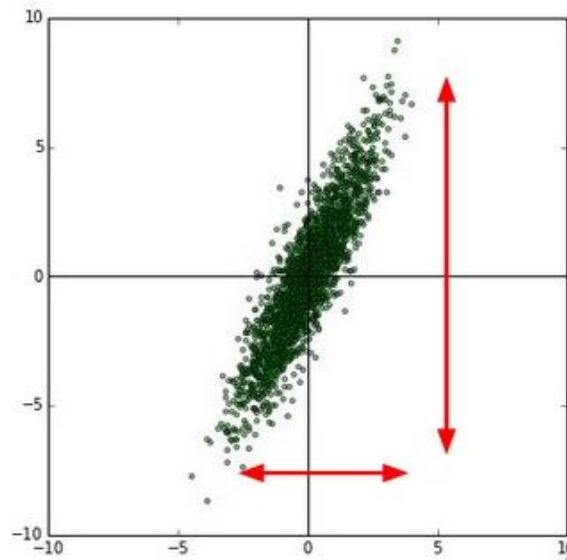
입력 정규화(Normalization) VS 화이트닝(Whitening)

- **입력 정규화**를 이용해 각 차원의 데이터가 동일한 범위 내의 값을 가지도록 만들 수 있습니다.
- 모든 특성(feature)에 대하여 각각 평균만큼 빼고 특정 범위의 값을 갖도록 조절할 수 있습니다.

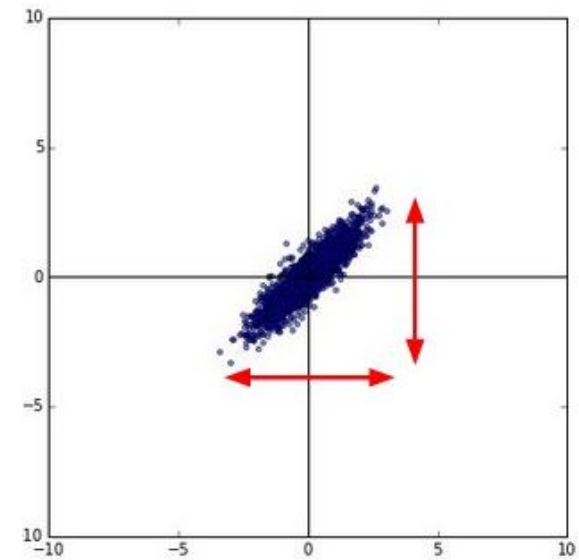
[2개의 특성(feature)으로 구성된 데이터셋의 정규화 예시]



원본 데이터셋



평균이 0이 된 데이터셋

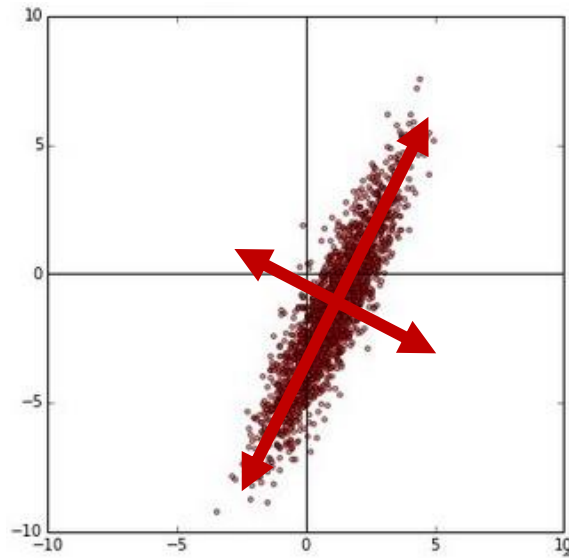


정규화된 데이터셋

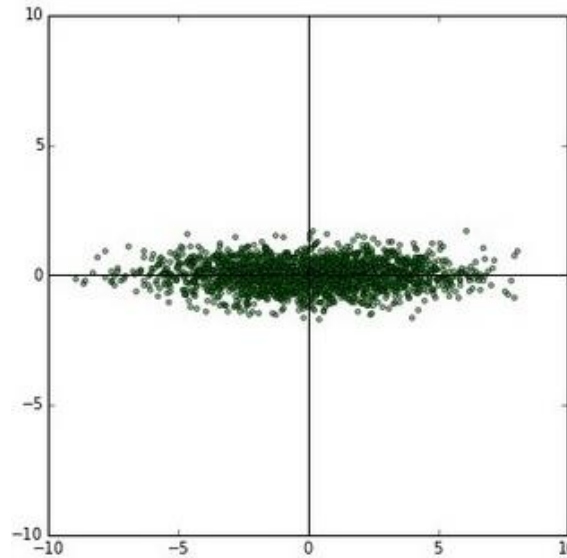
입력 정규화(Normalization) VS 화이트닝(Whitening)

- **화이트닝**은 평균이 0이며 공분산이 단위행렬인 정규분포 형태의 데이터로 변환하는 기법입니다.
- 일반적으로 PCA나 화이트닝보다는 정규화가 더 많이 사용됩니다.

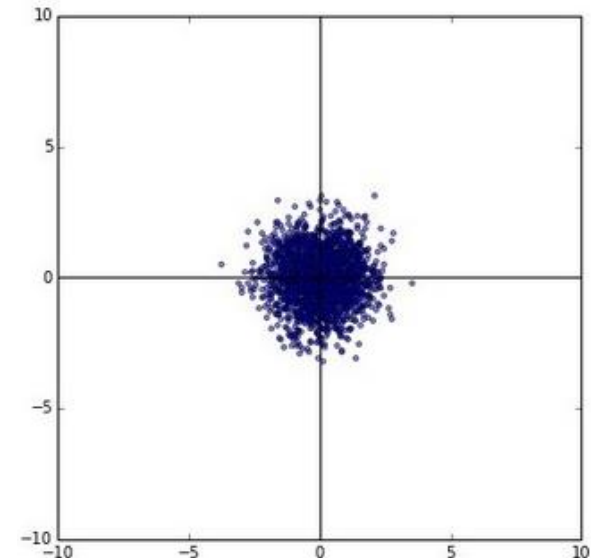
[2개의 특성(feature)으로 구성된 데이터셋의 화이트닝 예시]



원본 데이터셋



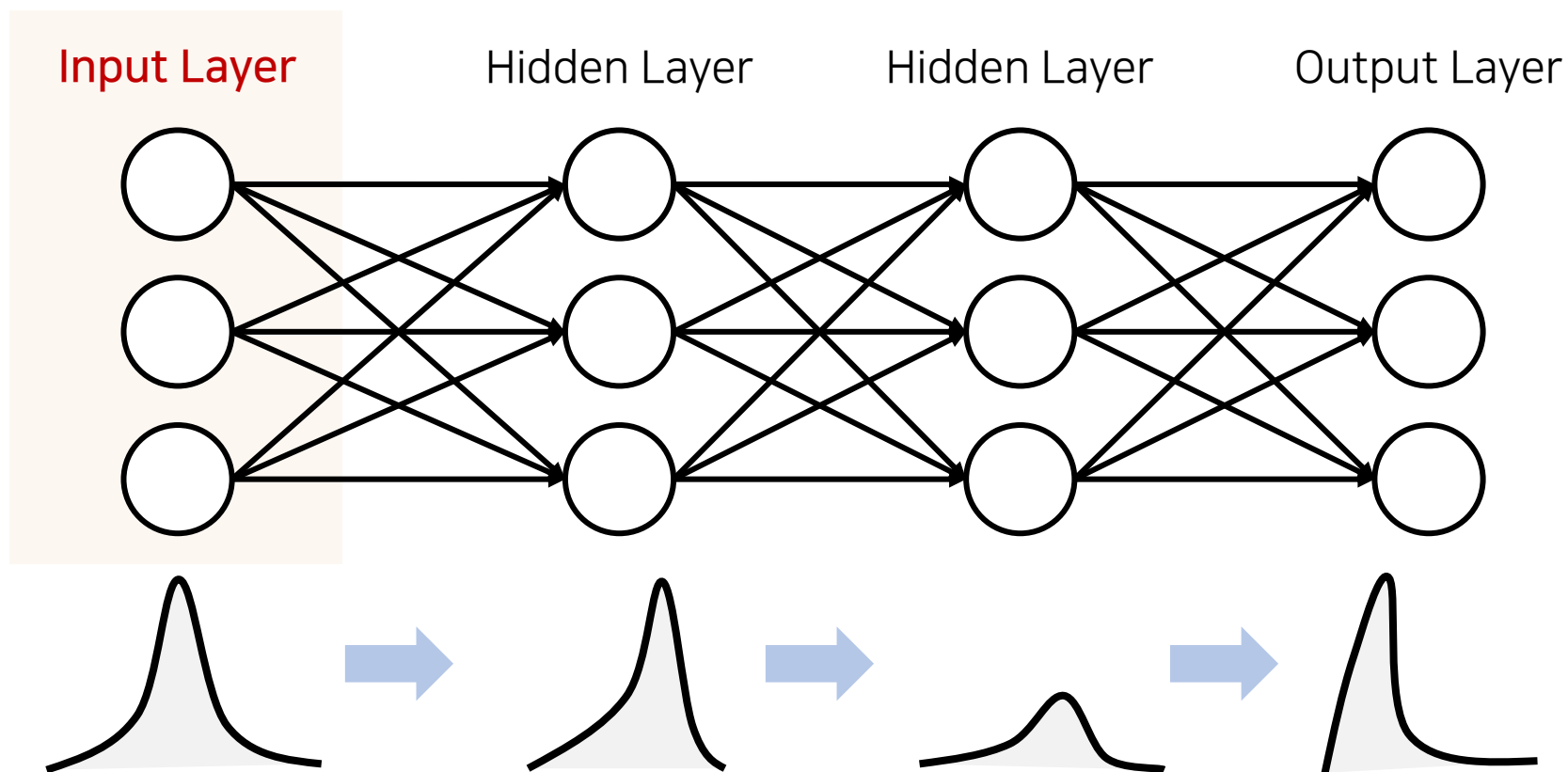
Decorrelated 데이터셋



화이트닝된 데이터셋

각 레이어에 대한 입력 분포

- 초기 입력 레이어의 데이터를 정규화하는 것은 상대적으로 간단합니다.
- 하지만 히든 레이어의 입력은 어떻게 정규화할 수 있을까요?



배치 정규화(Batch Normalization)

- 입력(Input)
 - A mini-batch: $Batch = \{x_1, x_2, \dots, x_m\}$
 - Parameters to be learned: γ, β
- 출력(Output)
 - $\{y_i = BN_{\gamma, \beta}(x_i)\}$

레이어의 입력 차원이 k 일 때, 학습할 두 개의 파라미터 γ 와 β 또한 k 차원을 가집니다.

$$\mu_{Batch} \leftarrow \frac{1}{m} \sum_{i=1}^m x_i \quad // \text{평균}$$

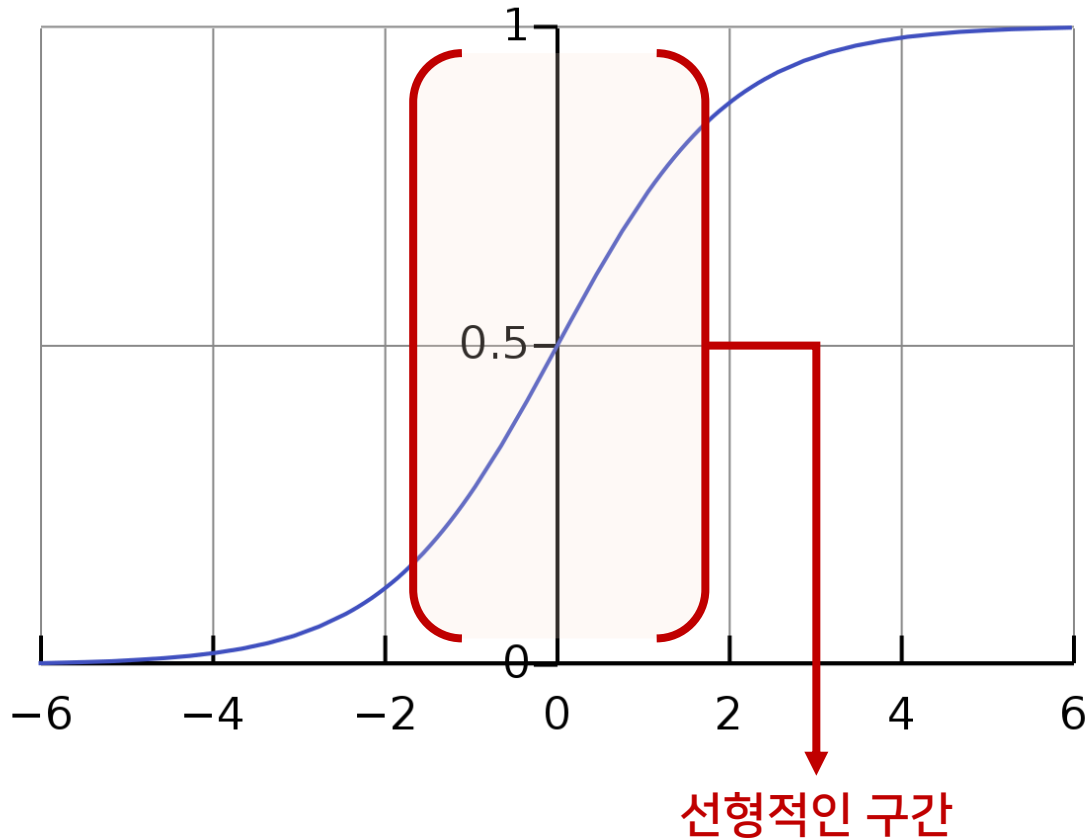
$$\sigma_{Batch}^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{Batch})^2 \quad // \text{분산}$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_{Batch}}{\sqrt{\sigma_{Batch}^2 + \epsilon}} \quad // \text{정규화}$$

$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv BN_{\gamma, \beta}(x_i)$$

레이어 입력을 정규화 할 때 유의할 점

- 각 레이어를 단순히 $N(0, 1)$ 로 정규화하면 비선형(non-linear) 활성화 함수의 영향력이 감소할 수 있습니다.
- Sigmoid 함수 예시



입력 데이터가 $N(0, 1)$ 로 정규화되므로 대부분의 입력에 대하여 매우 선형적으로 동작합니다.

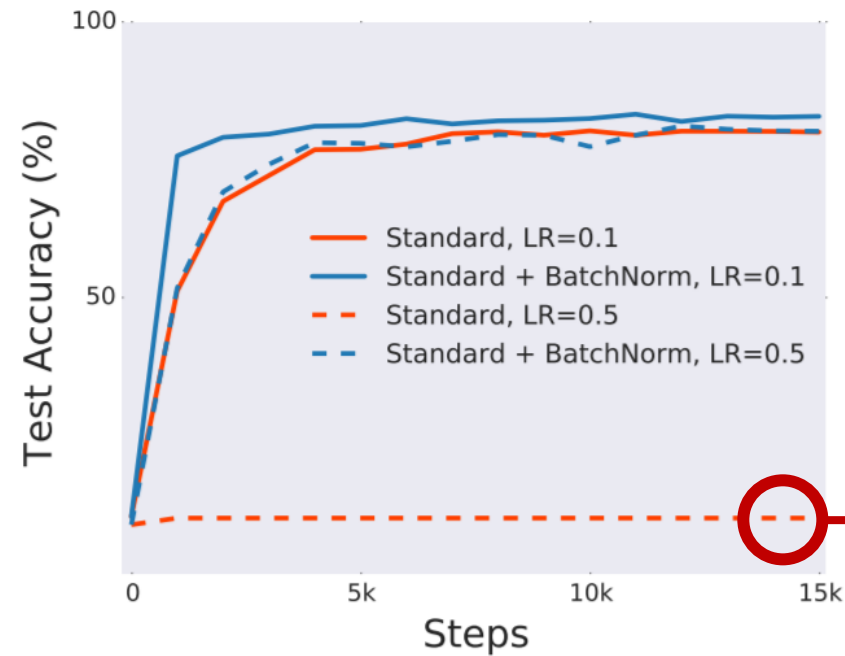
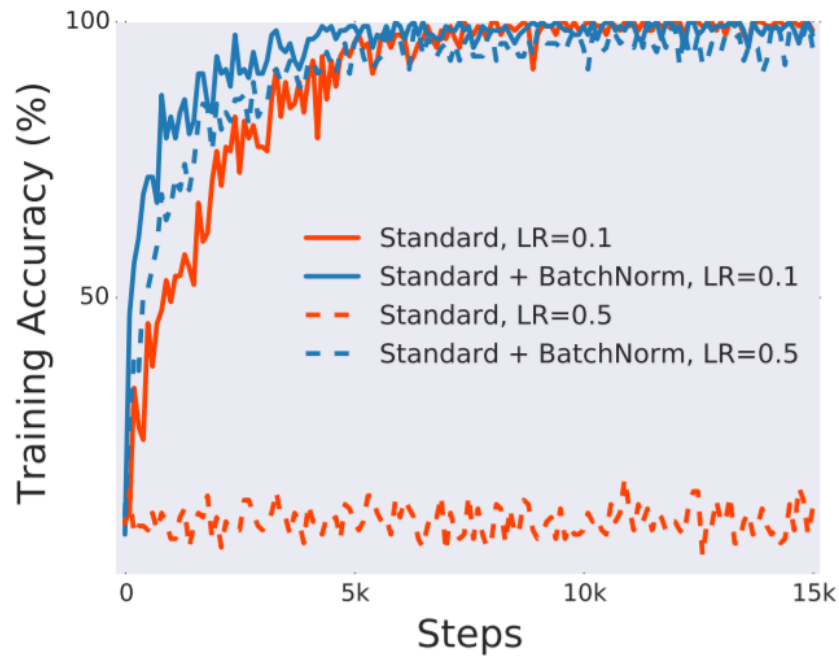


정규화 이후에 사용하는 감마(γ)와 베타(β)는 non-linearity를 유지할 수 있도록 해줍니다.

$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv BN_{\gamma, \beta}(x_i)$$

배치 정규화의 성능 향상

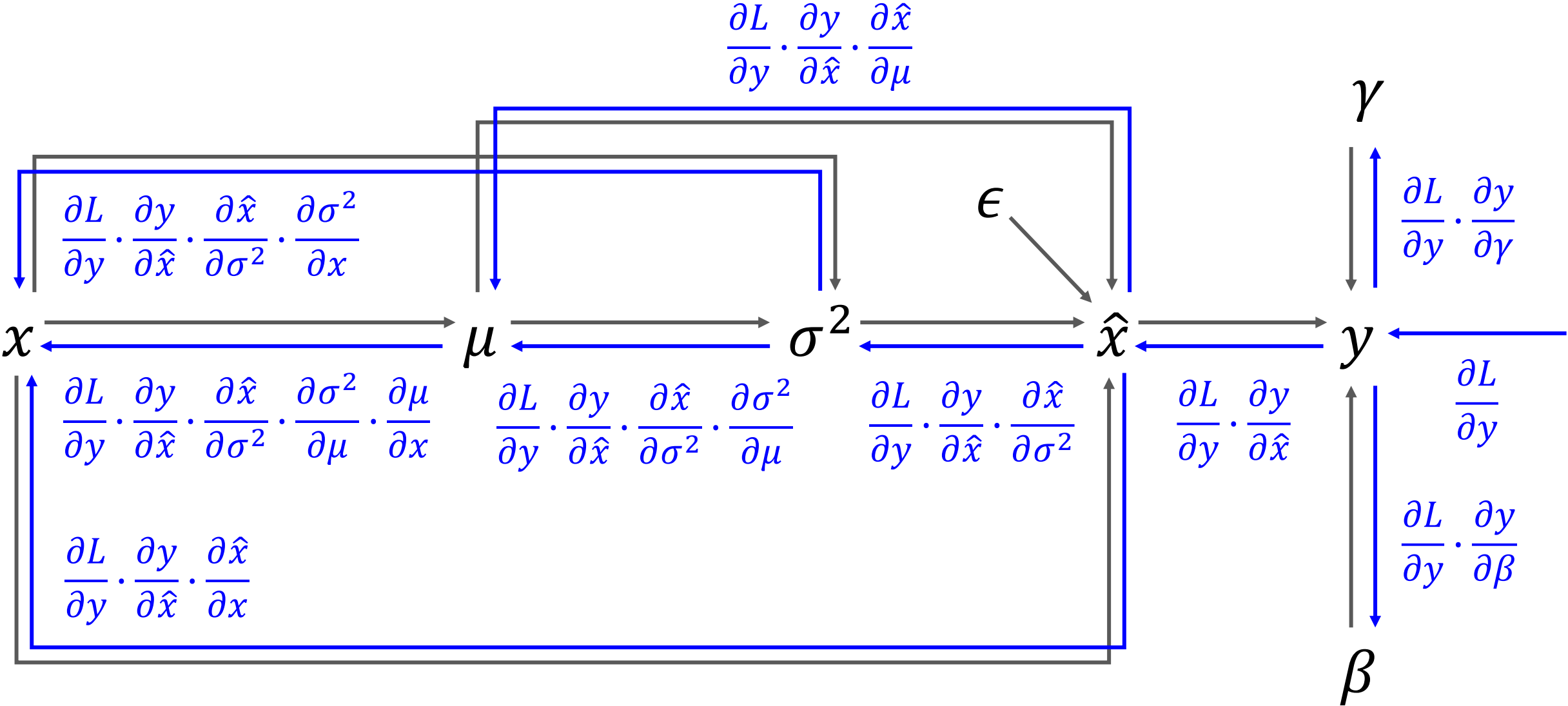
- 배치 정규화를 이용함으로써 얻을 수 있는 성능 향상 효과는 반박의 여지가 없습니다.
 - 학습을 위한 하이퍼 파라미터 설정으로부터 더 자유로우며, 학습이 빠르게 수행됩니다.



학습 자체가 진행되지 않음

How Does Batch Normalization Help Optimization? (NIPS 2018)

배치 정규화(Batch Normalization): 데이터 플로우 그래프



배치 정규화(Batch Normalization): 기울기(Gradient) 계산하기

$$\frac{\partial \ell}{\partial \hat{x}_i} = \frac{\partial \ell}{\partial y_i} \cdot \gamma$$

$$\frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} = \sum_{i=1}^m \frac{\partial \ell}{\partial \hat{x}_i} \cdot (x_i - \mu_{\mathcal{B}}) \cdot \frac{-1}{2} (\sigma_{\mathcal{B}}^2 + \epsilon)^{-3/2}$$

$$\frac{\partial \ell}{\partial \mu_{\mathcal{B}}} = \left(\sum_{i=1}^m \frac{\partial \ell}{\partial \hat{x}_i} \cdot \frac{-1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \right) + \frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{\sum_{i=1}^m -2(x_i - \mu_{\mathcal{B}})}{m}$$

$$\frac{\partial \ell}{\partial x_i} = \frac{\partial \ell}{\partial \hat{x}_i} \cdot \frac{1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} + \frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{2(x_i - \mu_{\mathcal{B}})}{m} + \frac{\partial \ell}{\partial \mu_{\mathcal{B}}} \cdot \frac{1}{m}$$

$$\left. \begin{aligned} \frac{\partial \ell}{\partial \gamma} &= \sum_{i=1}^m \frac{\partial \ell}{\partial y_i} \cdot \hat{x}_i \\ \frac{\partial \ell}{\partial \beta} &= \sum_{i=1}^m \frac{\partial \ell}{\partial y_i} \end{aligned} \right] \text{학습시킬 파라미터}$$

배치 정규화(Batch Normalization): 학습(Training) 및 추론(Inference)

Input: Network N with trainable parameters θ ; subset of activations $\{x^{(k)}\}_{k=1}^K$

Output: Batch-normalized network for inference, N_{BN}^{infer}

$N_{BN}^{train} \leftarrow N$ // Training BN network

for $k = 1 \dots K$ **do**

 Add transformation $y^{(k)} = BN_{\gamma^{(k)}, \beta^{(k)}}(x^{(k)})$ to N_{BN}^{train}

 Modify each layer in N_{BN}^{train} with input $x^{(k)}$ to take $y^{(k)}$ instead

end for

Train N_{BN}^{train} to optimize the parameters $\theta \cup \{\gamma^{(k)}, \beta^{(k)}\}_{k=1}^K$ **학습**

$N_{BN}^{infer} = N_{BN}^{train}$ // Inference BN network with frozen parameters

for $k = 1 \dots K$ **do**

 Process multiple training mini-batches $Batch$, each of size m , and average over them:

$$E[x] \leftarrow E_{Batch}[\mu_{Batch}]$$

$$Var[x] \leftarrow \frac{m}{m-1} E_{Batch}[\sigma_{Batch}^2]$$

 In N_{BN}^{infer} , replace the transform $y = BN_{\gamma, \beta}(x)$ with $y = \frac{\gamma}{\sqrt{Var[x] + \epsilon}} \cdot x + \left(\beta - \frac{\gamma E[x]}{\sqrt{Var[x] + \epsilon}} \right)$

end for

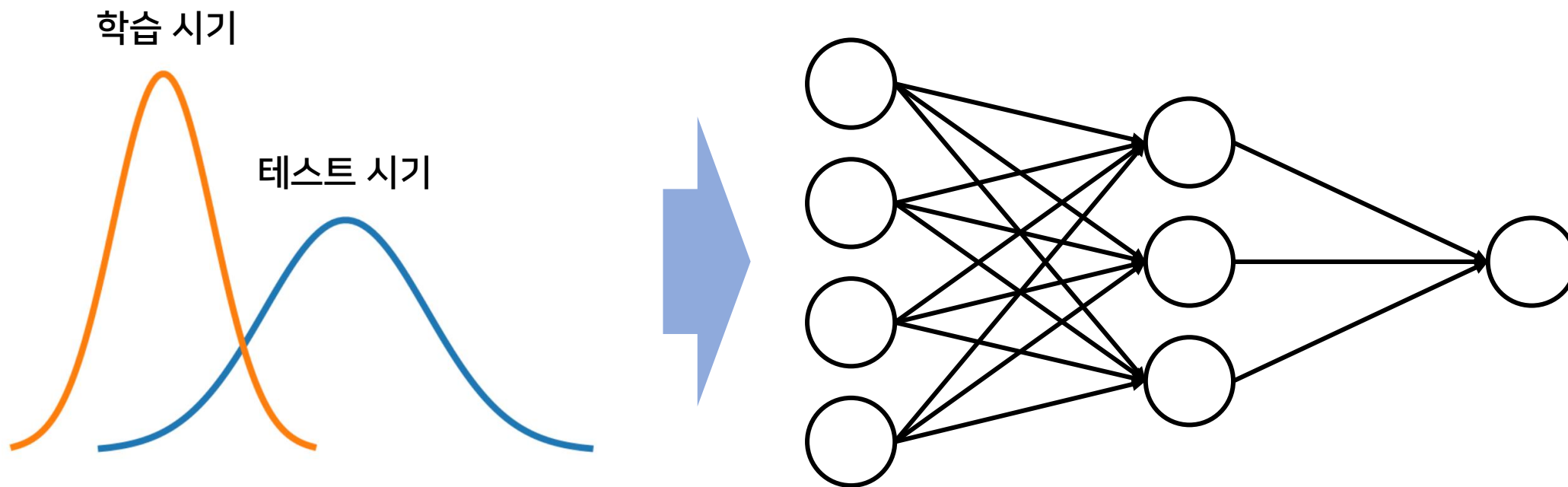
학습(Training) 단계

**추론(Inference)을 위한
준비 단계**

Internal Covariate Shift (ICS)

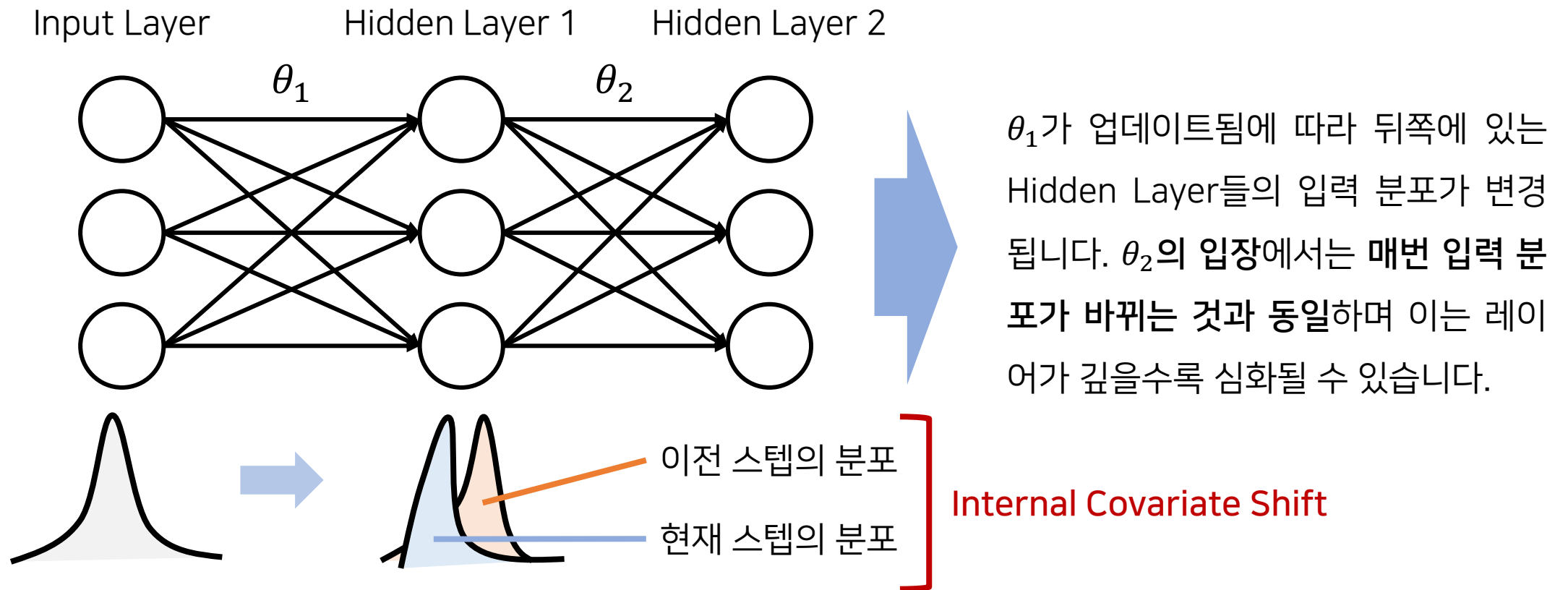
공변량 변화(Covariate Shift)

- 공변량 변화(covariate shift): 학습 시기와는 다르게 테스트 시기에 입력 데이터의 분포가 변경되는 현상
 - $P_{train}(y|x) = P_{test}(y|x)$ and $P_{train}(x) \neq P_{test}(x)$



Internal Covariate Shift (ICS) 가설

- 앞서 언급한 공변량 변화(covariate shift)가 네트워크 내부에서 발생하는 현상을 의미합니다.
 - 배치 정규화(batch normalization) 초창기 논문에서 해결하고자 했던 문제 상황입니다.



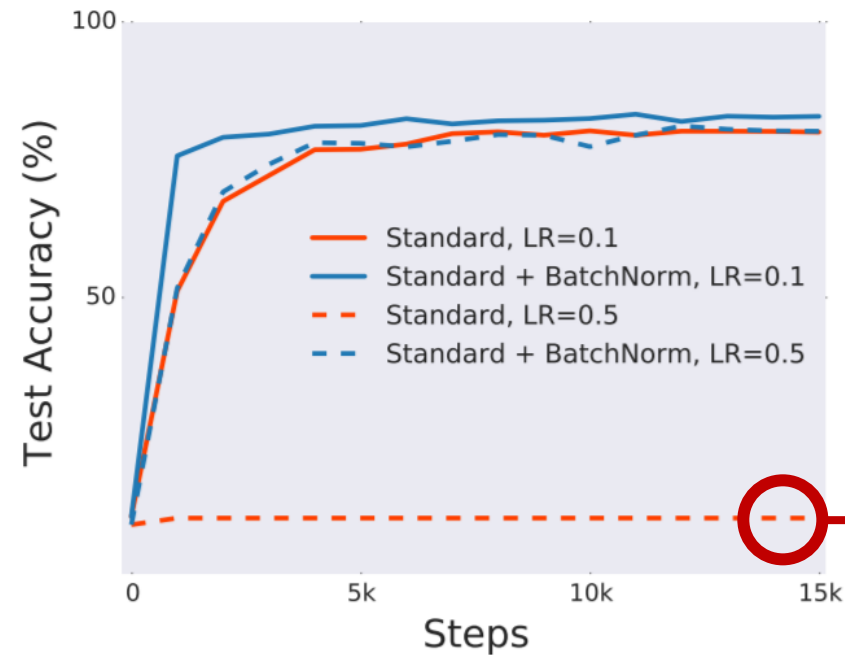
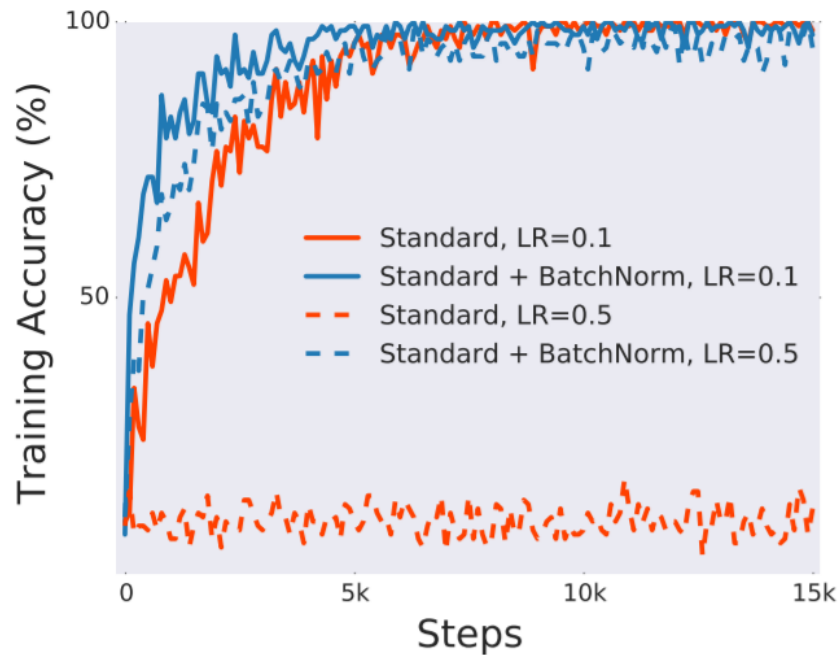
정말 배치 정규화의 성능 향상은 ICS의 감소로부터 기인한 것일까요?



후속 연구에 의하면 배치 정규화의 효과와 ICS의 감소는 큰 상관이 없다는 주장이 제기됩니다.

배치 정규화의 성능 향상

- 배치 정규화를 이용함으로써 얻을 수 있는 성능 향상 효과는 반박의 여지가 없습니다.
 - 학습을 위한 하이퍼 파라미터 설정으로부터 더 자유로우며, 학습이 빠르게 수행됩니다.

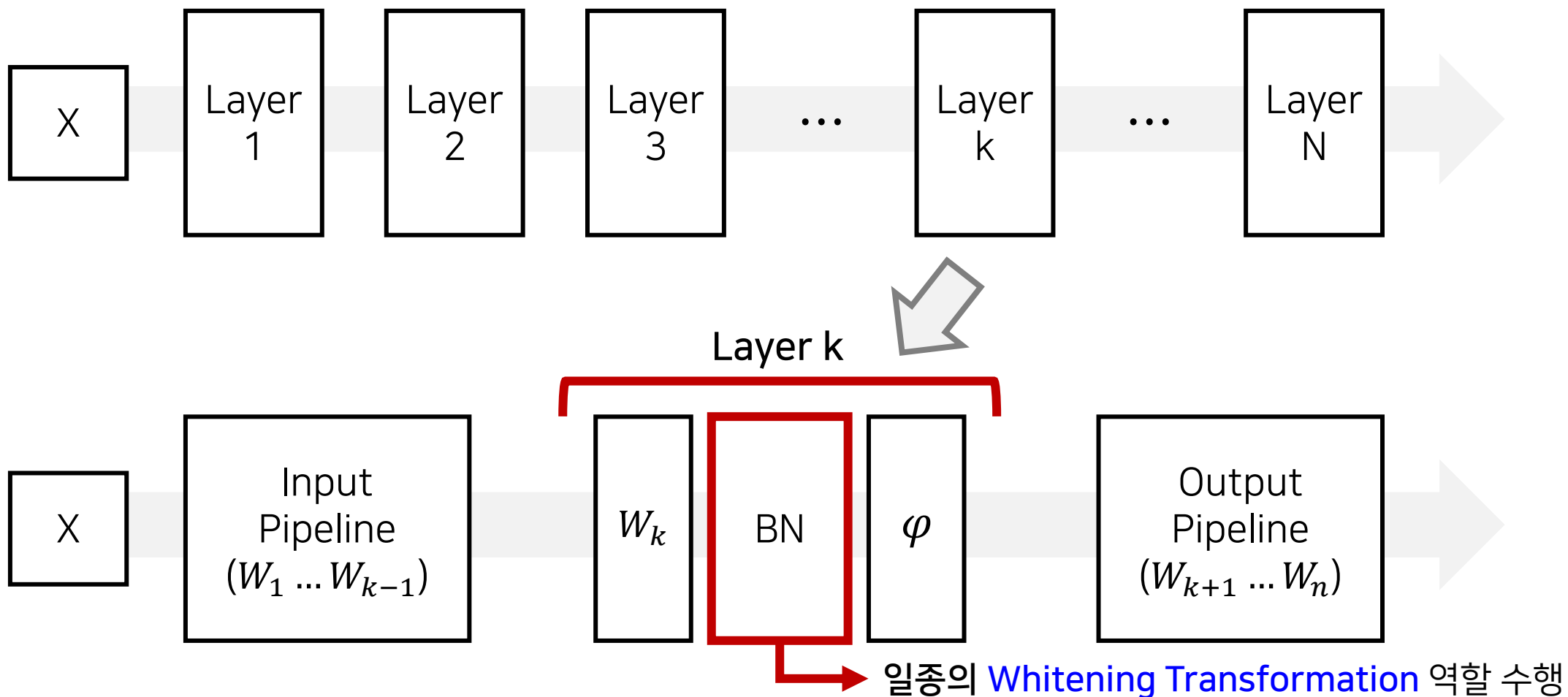


학습 자체가 진행되지 않음

How Does Batch Normalization Help Optimization? (NIPS 2018)

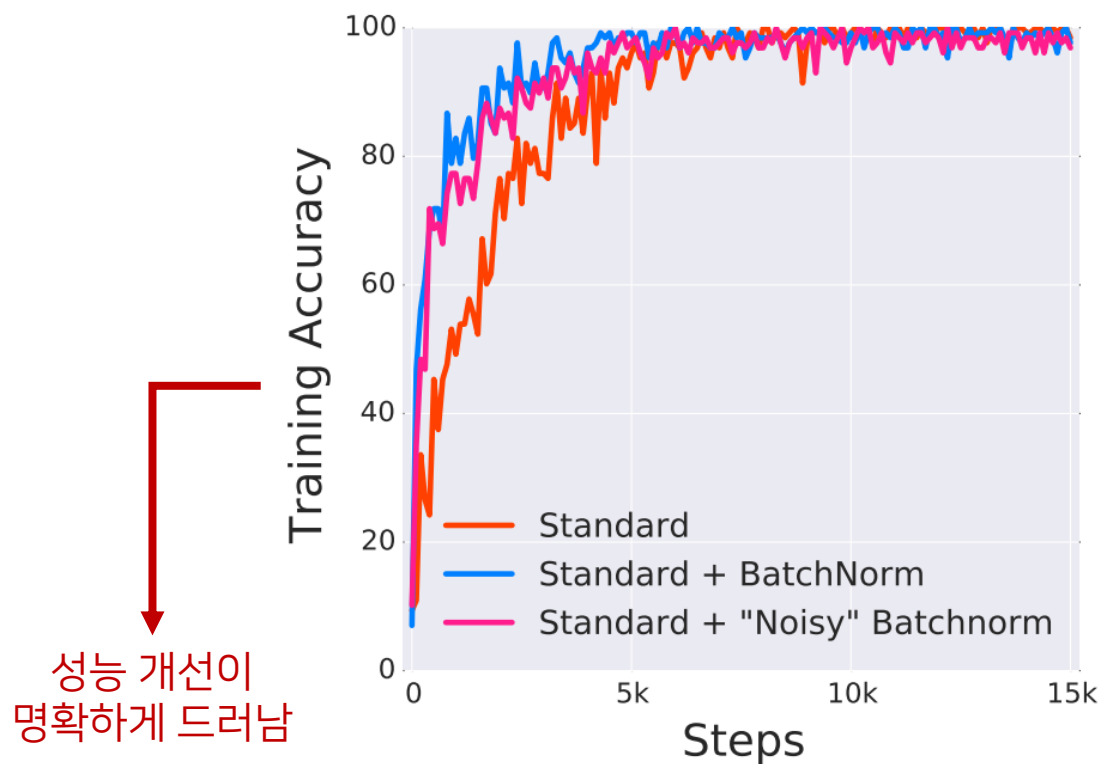
배치 정규화의 적용

- 배치 정규화를 적용한 형태를 도식화하면 다음과 같습니다.

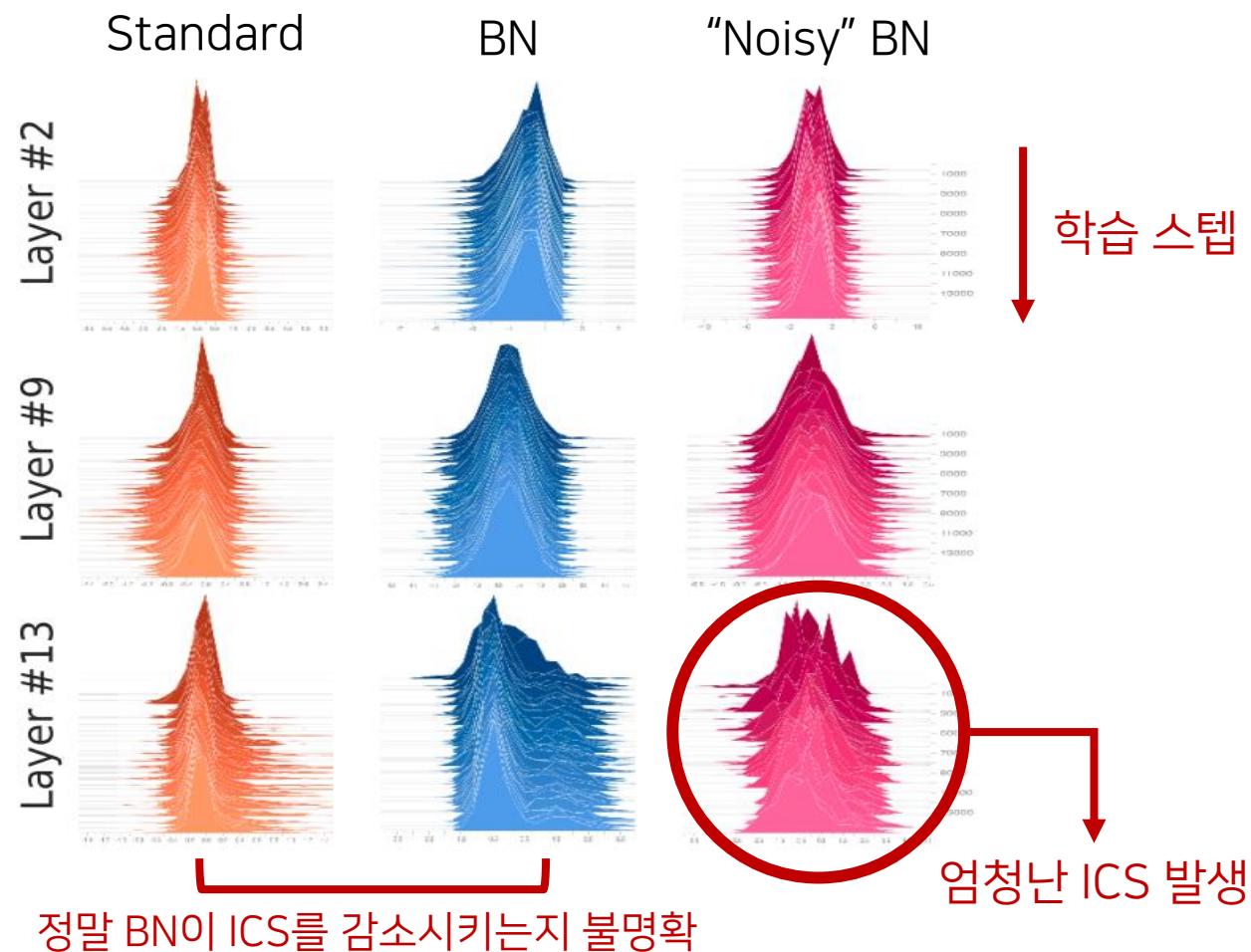


배치 정규화와 ICS와의 관계

- 배치 정규화 레이어 **직후에** 랜덤 노이즈를 삽입하여 ICS를 강제로 발생시켰을 때에도 여전히 일반적인 네트워크보다 성능이 우수했습니다.

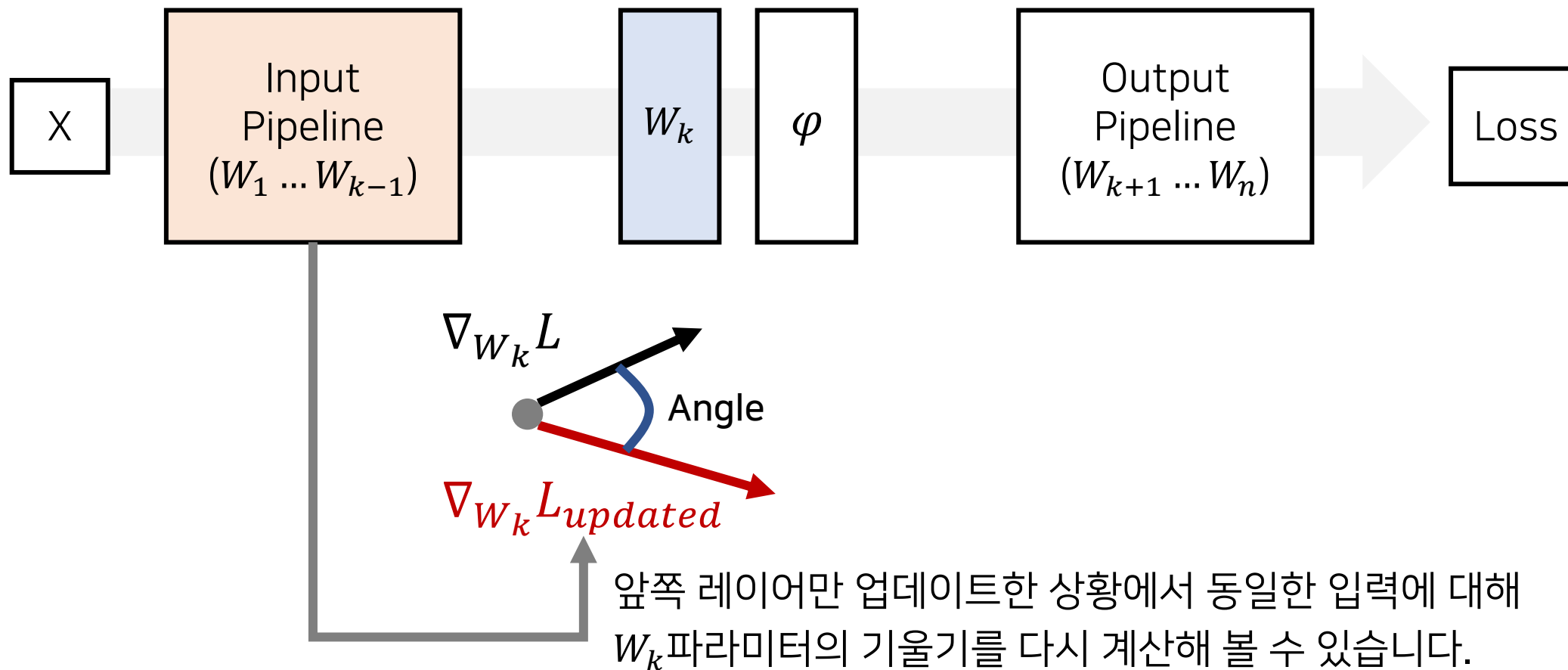


How Does Batch Normalization Help Optimization? (NIPS 2018)



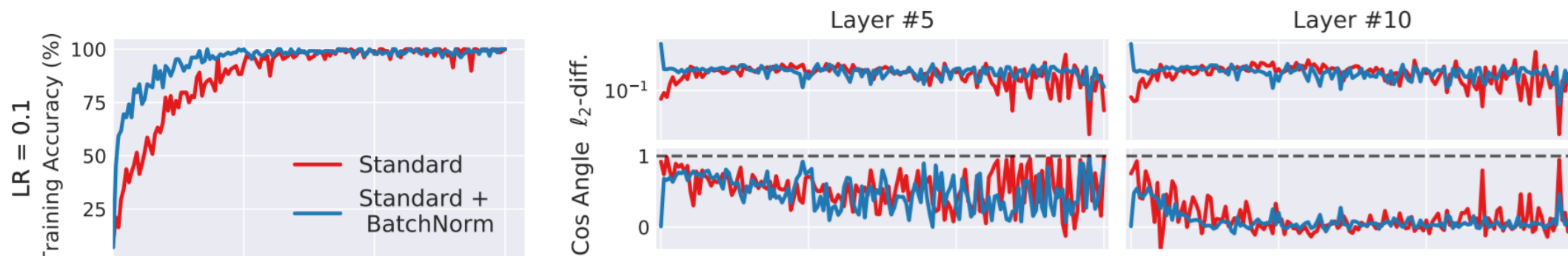
ICS를 계산하는 또 다른 방법

- ICS를 계산하기 위한 또 다른 방법으로 다음과 같은 방법을 사용할 수 있습니다.

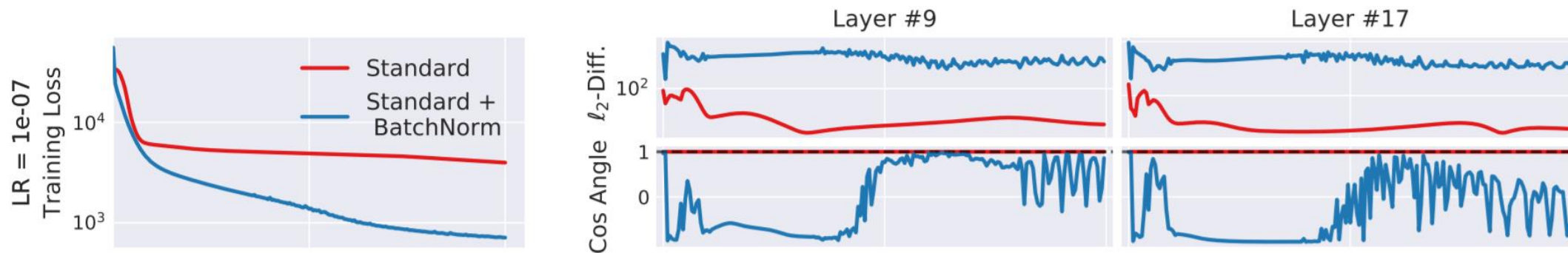


ICS를 계산하는 또 다른 방법

- 배치 정규화를 사용해도 ICS는 그대로이거나 오히려 증가하는 것을 확인할 수 있습니다.



(a) VGG



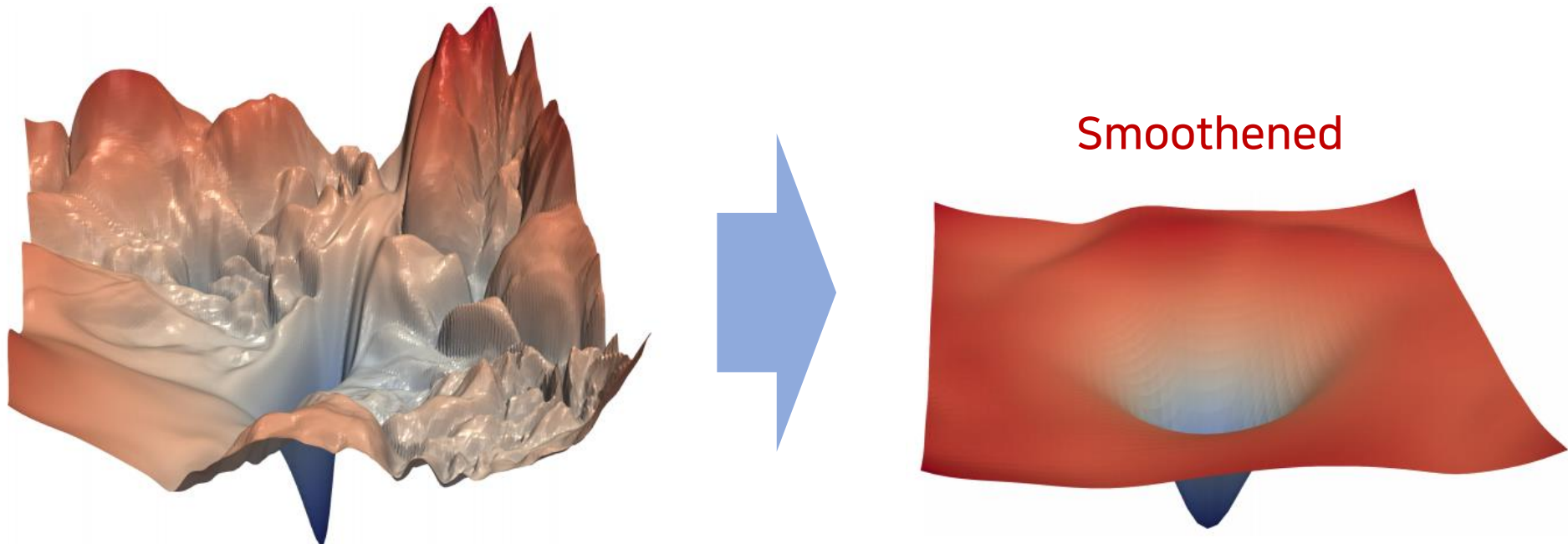
(b) DLN

How Does Batch Normalization Help Optimization? (NIPS 2018)

배치 정규화가 잘 동작하는 이유 분석

배치 정규화의 Smoothing 효과

- 배치 정규화는 Optimization Landscape를 부드럽게 만드는(Smoothing) 효과가 있습니다.
 - 흔히 말하는 Loss Landscape는 가중치 값에 따른 Loss 값을 시각화한 것을 의미합니다.
 - Smoothing을 이끌어낸다고 알려진 기법들: Batch Normalization, Residual Connection 등



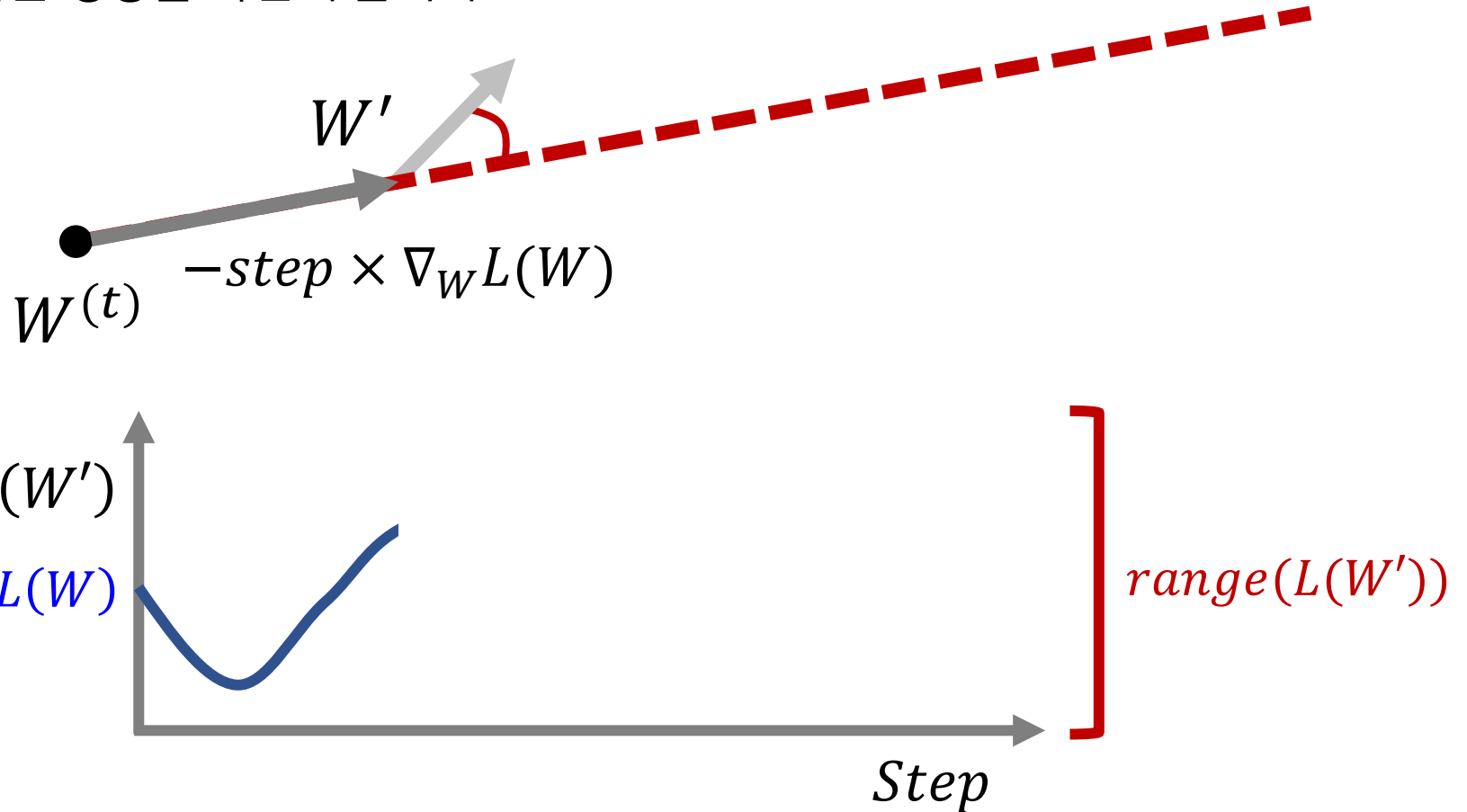
Visualizing the Loss Landscape of Neural Nets (NIPS 2018)

배치 정규화의 Smoothing 효과

- 초기 기울기 방향에 따른 Optimization Landscape를 확인할 수 있습니다.
 - Step의 크기를 크게 키움에 따른 영향을 확인해 봅시다.

Changes in gradient:

Big differences imply less reliable gradients



Loss Variation:

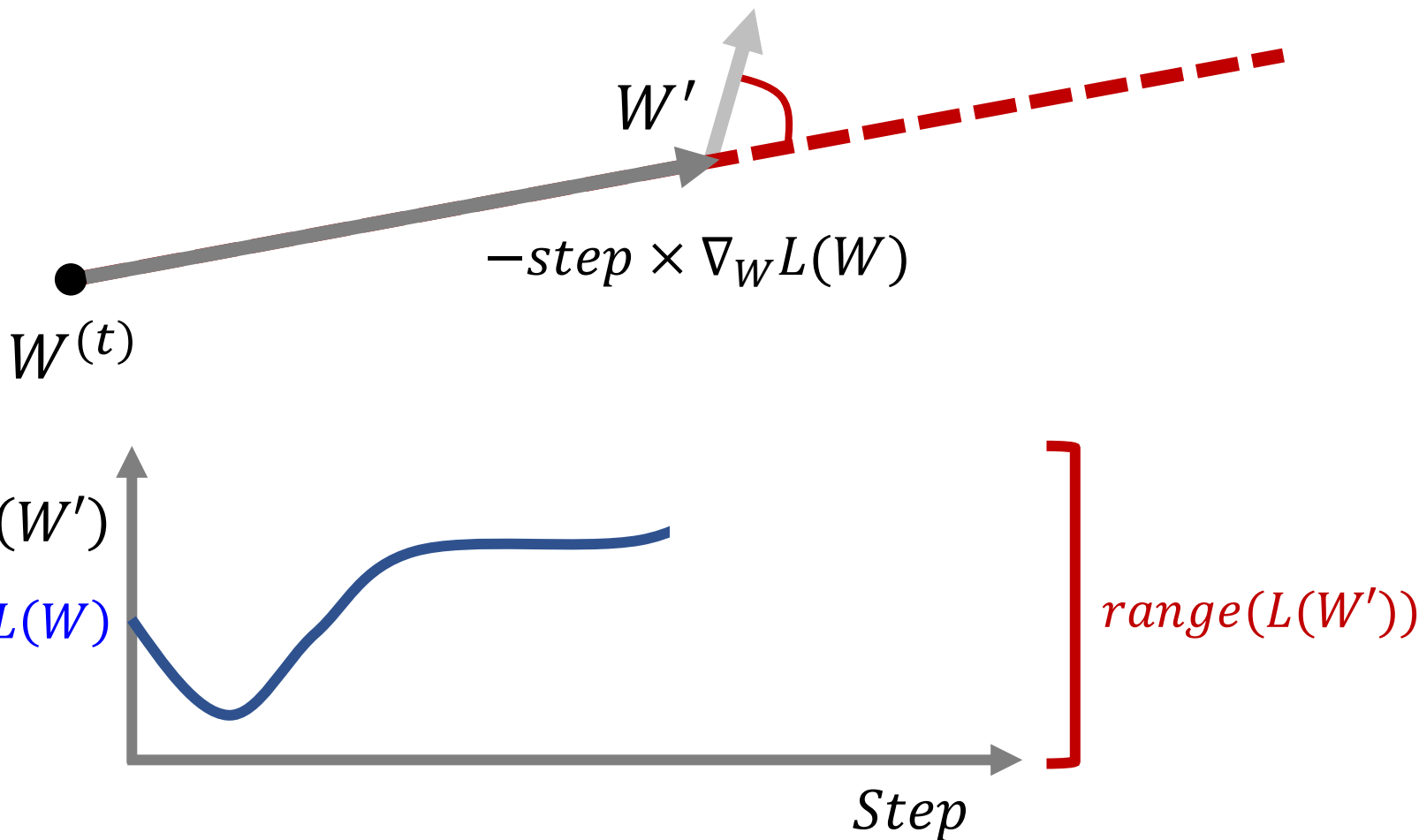
Large fluctuations make optimization hard.

배치 정규화의 Smoothing 효과

- 초기 기울기 방향에 따른 Optimization Landscape를 확인할 수 있습니다.

Changes in gradient:

Big differences imply less reliable gradients



Loss Variation:

Large fluctuations make optimization hard.

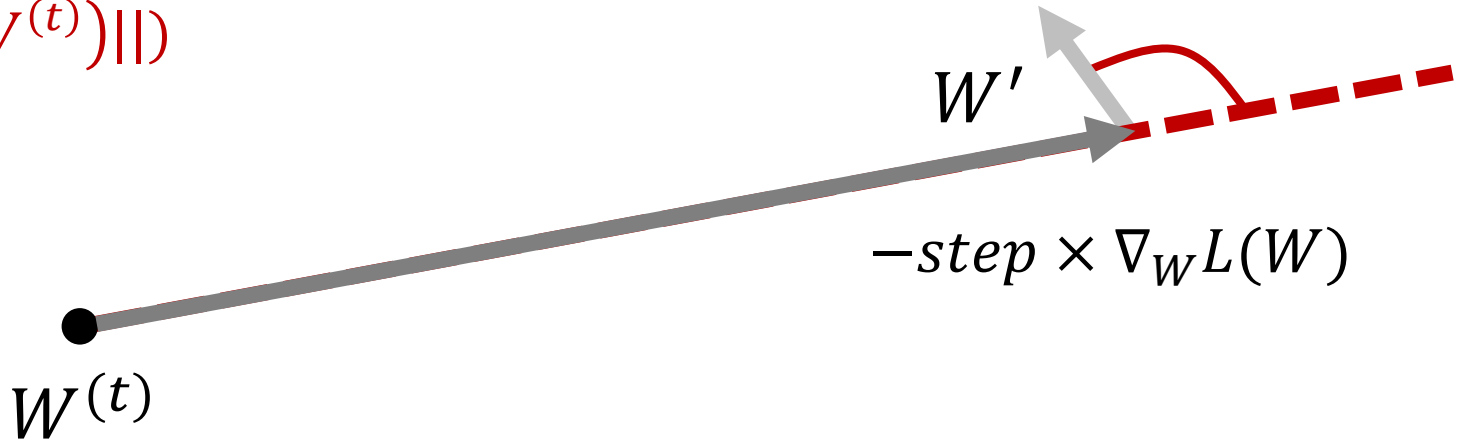
배치 정규화의 Smoothing 효과

- 초기 기울기 방향에 따른 Optimization Landscape를 확인할 수 있습니다.

$$\text{range}(\|\nabla_W L(W') - \nabla_W L(W^{(t)})\|)$$

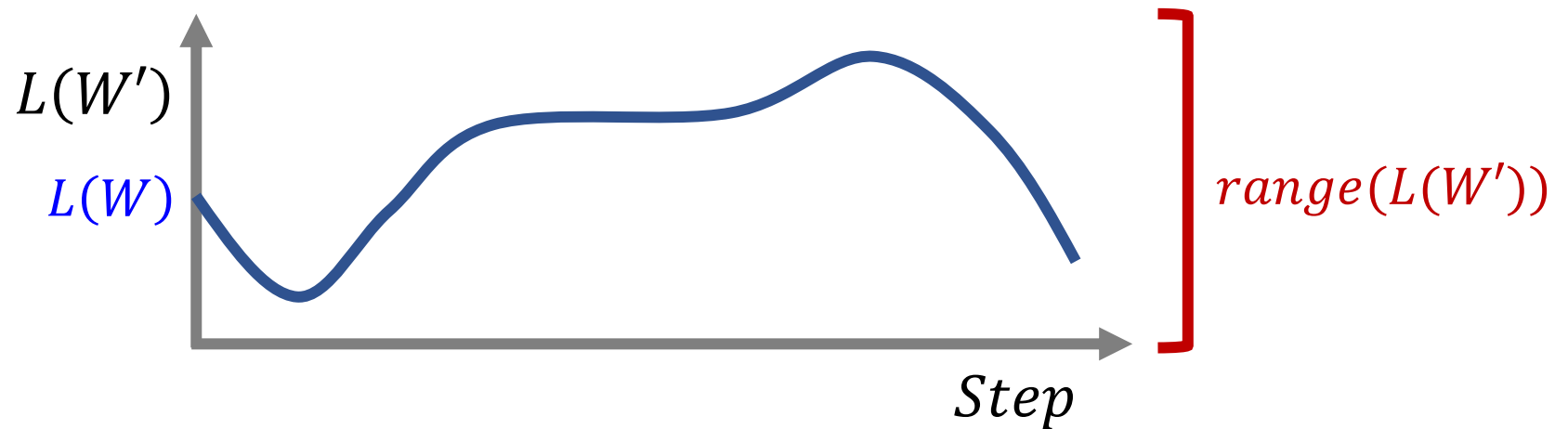
Changes in gradient:

Big differences imply less reliable gradients



Loss Variation:

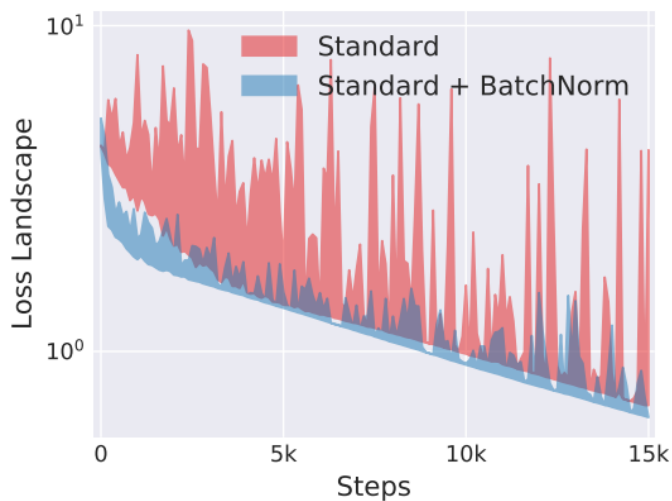
Large fluctuations make optimization hard.



배치 정규화의 Smoothing 효과

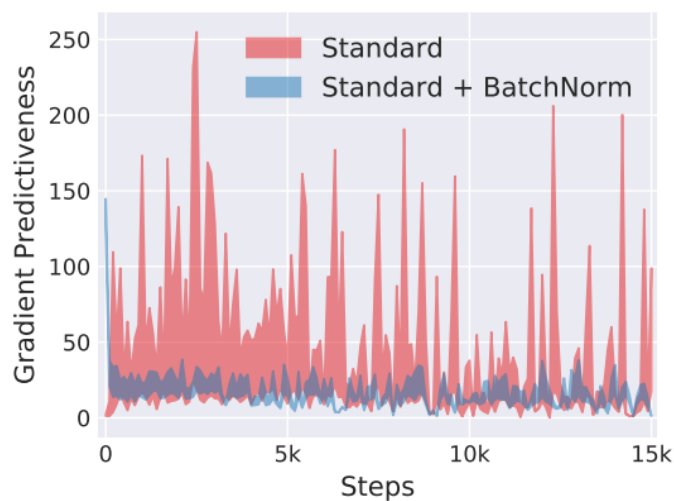
- 배치 정규화는 ICS를 해결하는 것과 큰 연관성이 없으며 그 성능 향상은 Smoothing에서 기인합니다.
 - 기울기 예측성(Predictiveness): 현재의 기울기 방향성을 얼마나 믿을 수 있는지

Variation in Loss ($L(W)$)

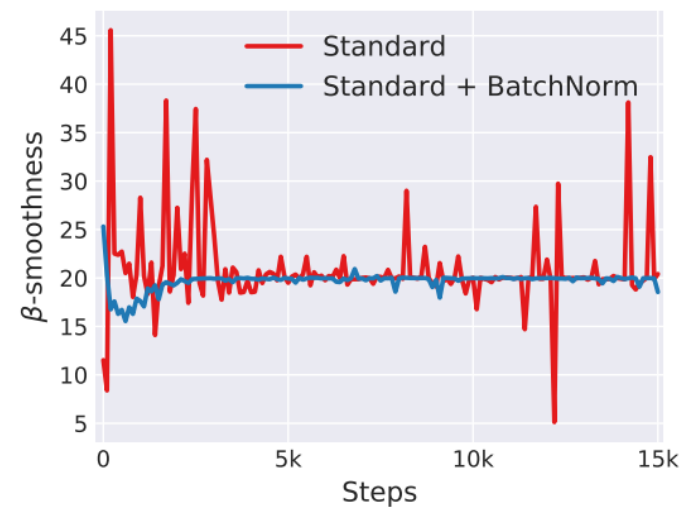


(a) Loss Landscape

Change in Gradient ($\nabla_W L(W)$)



(b) Gradient Predictiveness



(c) "effective" β -smoothness

립시츠 연속 함수(Lipschitz-Continuous Function)

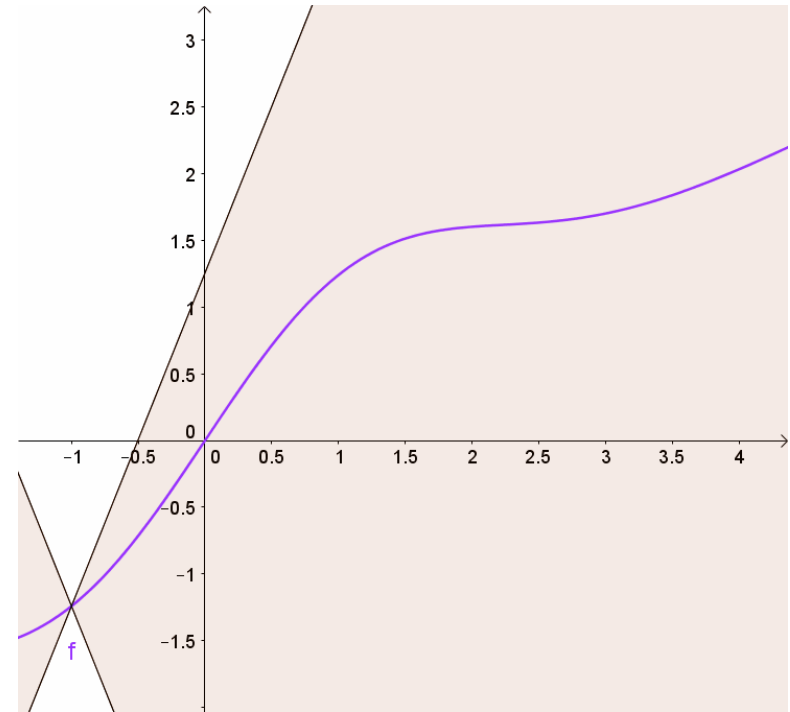
- 연속적이고 미분 가능하며 어떠한 두 점을 잡아도 기울기가 K 이하인 함수입니다.
 - 쉽게 생각하자면, 급격한 변화 없이 (K 만큼) 전반적으로 물 흐르듯이 완만한 기울기를 가지는 함수 형태
- K -Lipschitz 함수의 정의

$$\frac{|f(x_1) - f(x_2)|}{|x_1 - x_2|} \leq K$$

for all x_1 and x_2 .

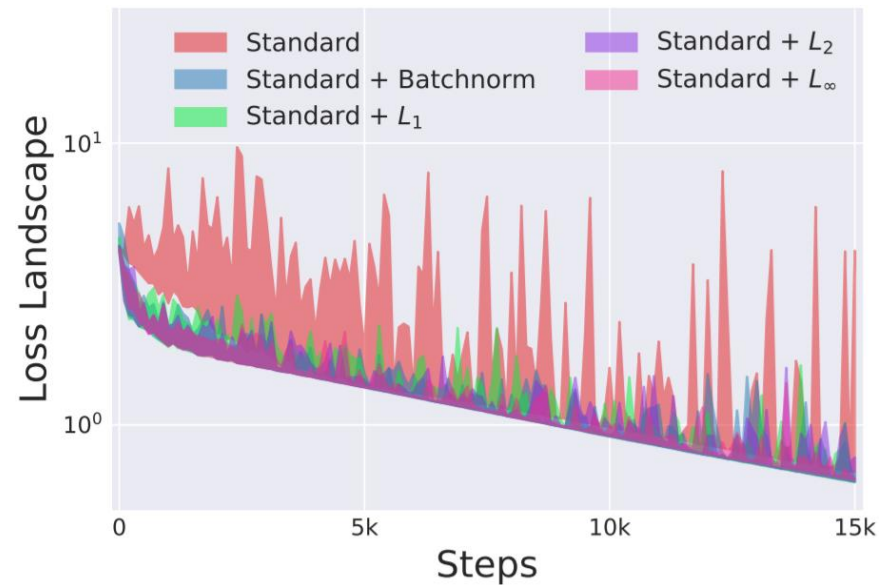
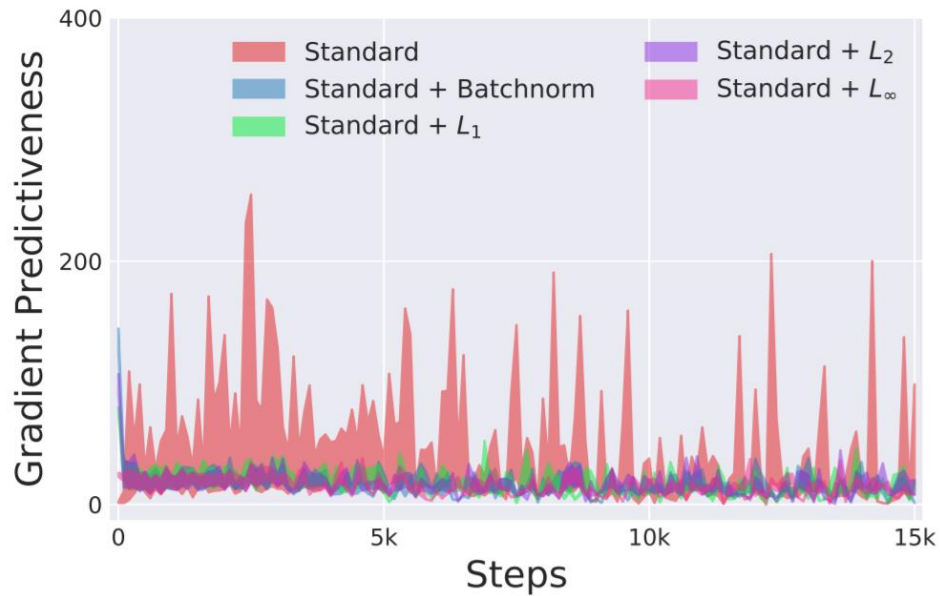


파라미터에 대한 비용함수가 립시츠 연속이라면
상대적으로 안정적인 학습을 진행할 수 있습니다.



배치 정규화가 잘 동작하는 이유

- 립시츠(Lipschitzness)가 향상된다는 것은 현재의 기울기 방향으로 큰 스텝(step)만큼 이동을 한 뒤에도 이동한 뒤의 기울기의 방향과 유사할 가능성이 높다는 것을 의미합니다.
- 배치 정규화가 아니더라도 Smoothing 효과가 있는 기법을 사용하면 성능이 향상될 수 있습니다.
 - 논문에서는 이를 실험적으로/이론적으로 증명하고 있습니다.



Visualizing the Loss Landscape of Neural Nets (NIPS 2018)