김성수

This Week

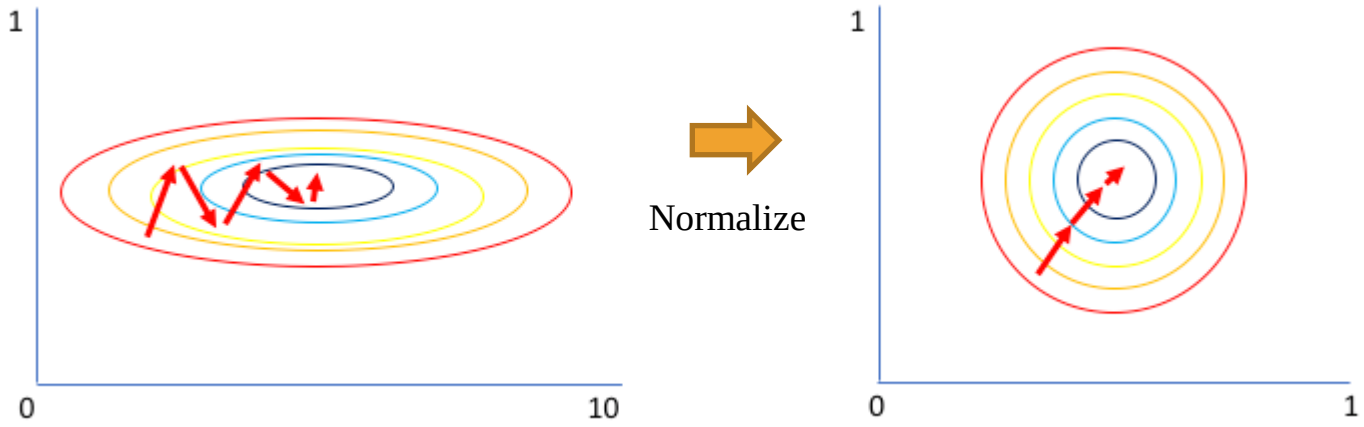
[1] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in Proceedings of the 32nd International Conference on Machine Learning, Lille, France, Jul. 2015, vol. 37, pp. 448-456

[2] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry, "How Does Batch Normalization Help Optimization?," in Advances in Neural Information Processing Systems, 2018, vol. 31

- Paper review
  - Batch Normalization : Accelerating Deep Network Training by Reducing Internal Covariate Shift[1]
  - How Does Batch Normalization Help Optimization?[2]

This Week

- Normalization
  - 목적: 데이터가 각 feature dimension(분포 상의 축)에 대해서 비슷한 값의 범위를 가지게 하는 것.
    - Weight의 update가 특정 feature에 치중되는 현상을 막음.
    - Training시 상대적으로 더 큰 learning rate를 사용할 수 있게 함.
  - Ex) Two feature dimension data



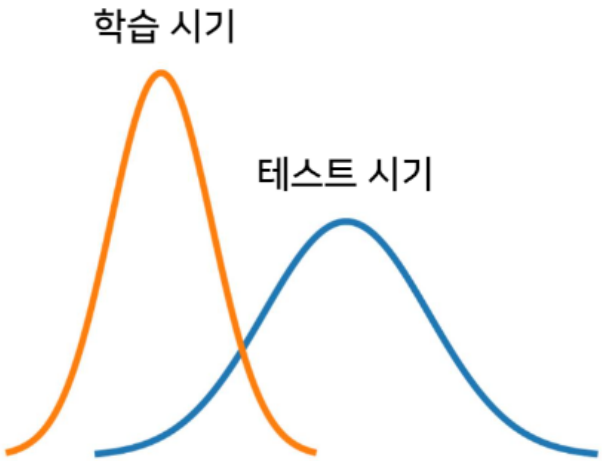
초기 Input 레이어의 입력 데이터를 정규화 하는것은 상대적으로 간단함.

그러나 입력 데이터가 정규 분포를 띈다 해도
매 step에서 batch 데이터의 분포가 hidden layer를 거치면서 예측하기 힘든 형태로 변함.(Internal covariate shift)

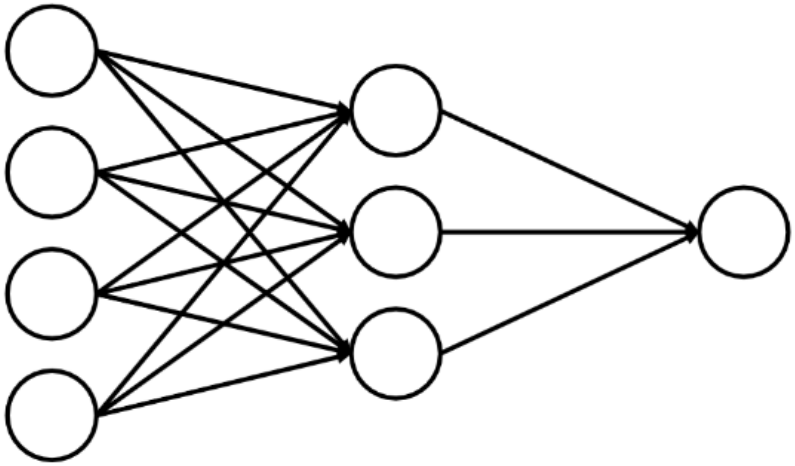
그렇다면 hidden 레이어의 입력은 어떻게 normalize 할수있는가? → Batch normalization

This Week

- Covariate shift
  - 학습 시기와는 다르게 테스트 시기에 입력 데이터의 분포가 변경되는 현상



Input data distribution

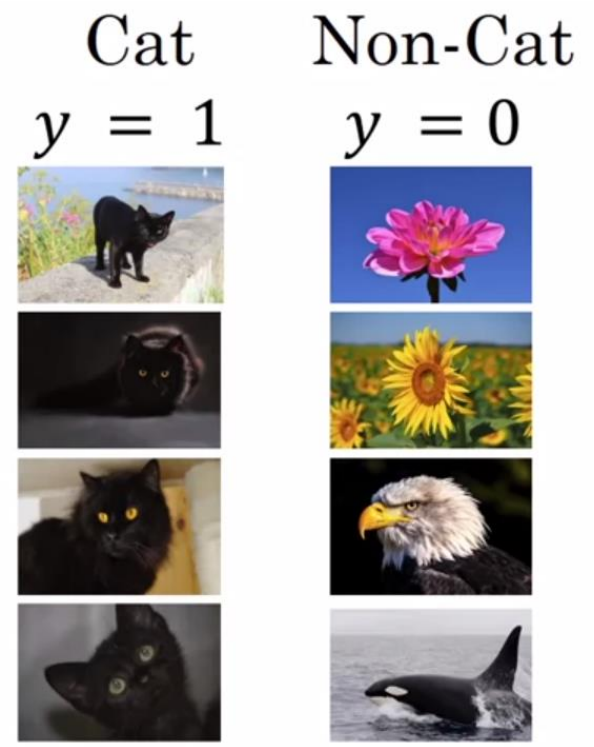


Network

This Week

https://www.coursera.org/lecture/deep-neural-network/why-does-batch-norm-work

- Covariate shift
  - 학습 시기와는 다르게 테스트 시기에 입력 데이터의 분포가 변경되는 현상



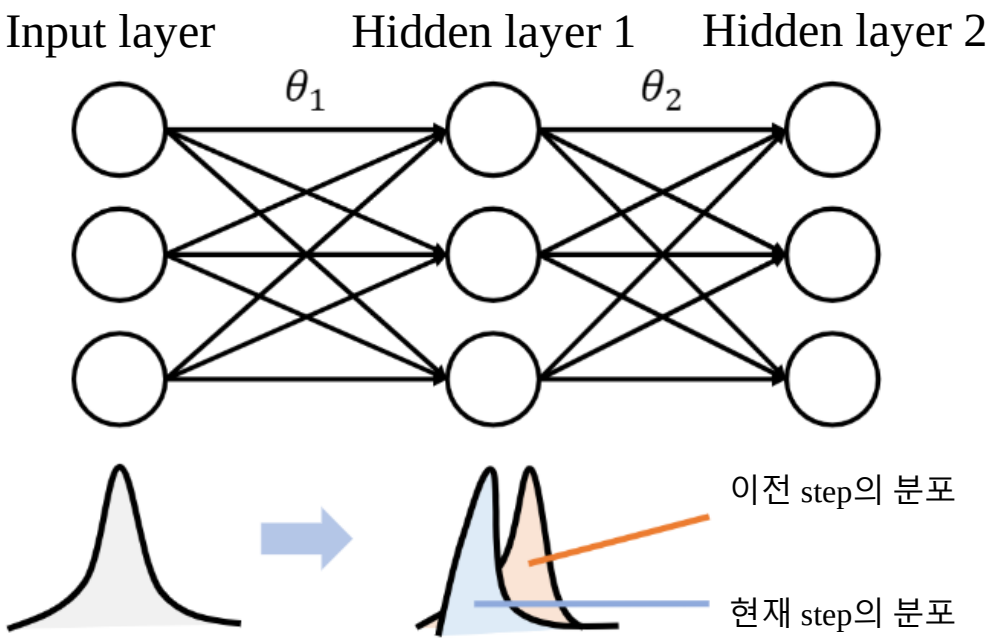
Train data

Covariate shift

Test data

This Week

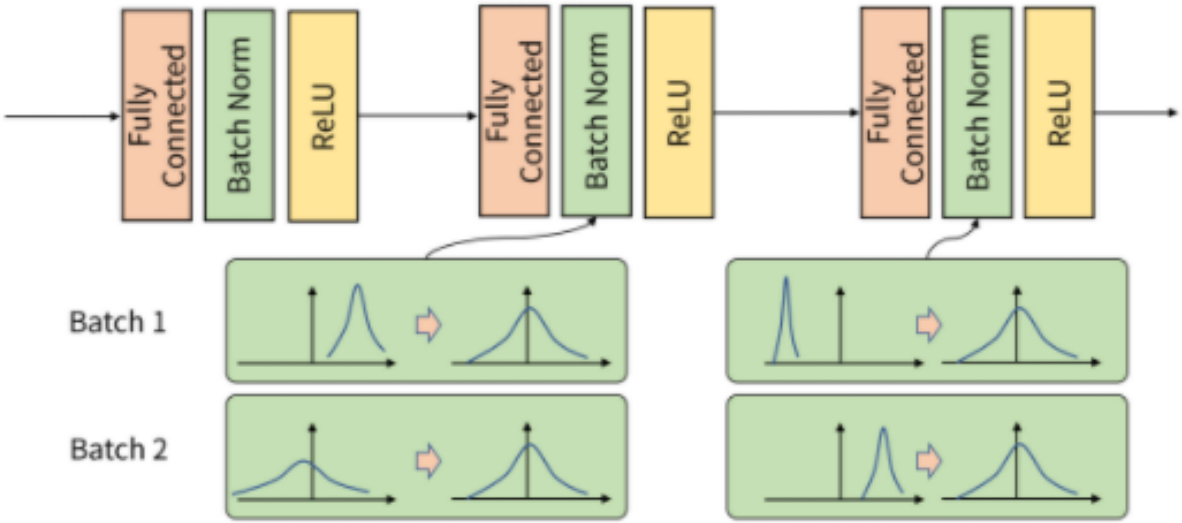
- Internal covariate shift (ICS)
  - Covariate shift가 네트워크 내부에서 발생하는 현상으로, Batch normalization 초창기 논문에서 해결하고자 한 문제점.



θ_1가 $update$됨에 따라 뒤쪽의 $hidden\ layer$의 $input\ distribution$이 변경됨. θ_2의 입장에서는 매 step 마다 input distribution이 바뀌는 것과 동일함.

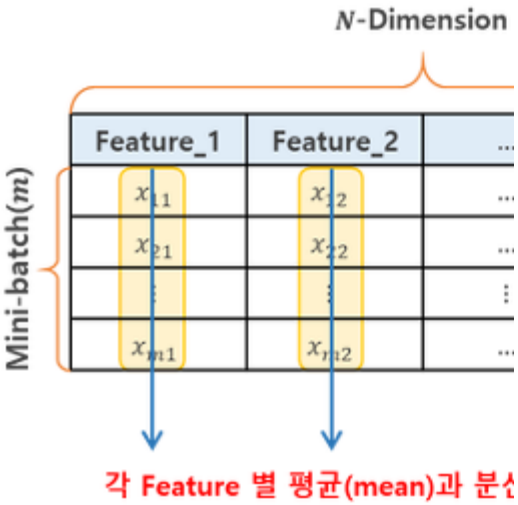
This Week

- Batch normalization
  - 학습 과정에서 이전 layer를 거침에 따라서 , 또는 batch의 데이터 분포가 변함에 따라서 다음 layer로의 입력이 다양한 분포를 가지는 것을 방지하기 위해 **batch의 평균과 분산을 이용해 layer의 출력을 정규화**하는 것. (mean = 0, std =1)



This Week

- Batch normalization
  - 학습 과정에서 이전 layer를 거침에 따라서 , 또는 batch의 데이터 분포가 변함에 따라서 다음 layer로의 입력이 다양한 분포를 가지는 것을 방지하기 위해 **batch의 평균과 분산을 이용해 layer의 출력을 정규화**하는 것. (mean = 0, std =1)



Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

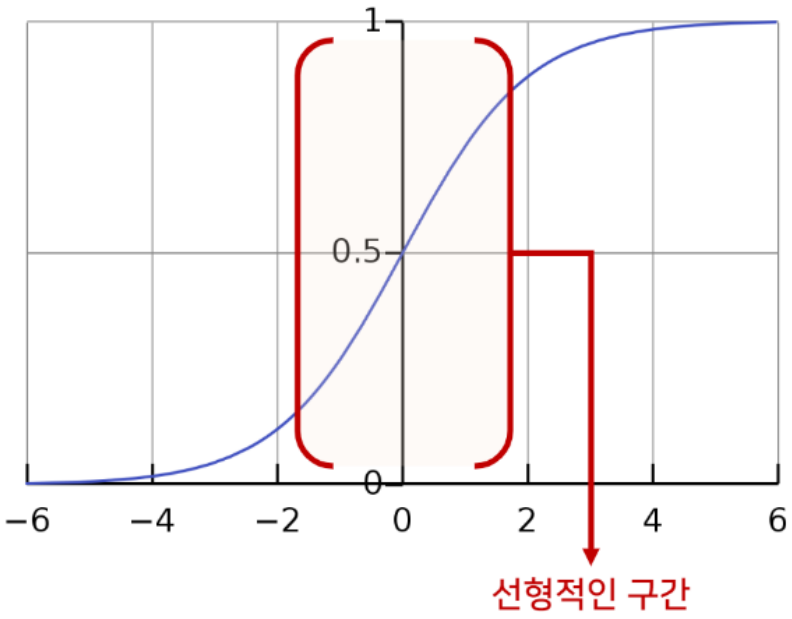
Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

This Week

- Batch normalization parameter γ, β
  - γ : Scaling factor
  - β : Translation factor
  - Learnable parameter를 두는 이유
    : 각 layer를 단순히 N(0,1)로 normalize 할 경우, non-linear activation function의 영향력이 감소할 수 있기 때문.

$$\hat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}} \qquad y^{(k)} = \gamma^{(k)}\hat{x}^{(k)} + \beta^{(k)}$$

Ex) Sigmoid activation function



Input data가 N(0,1)로 normalize 될 경우, 대부분의 입력에 대해 선형적인 activation output를 얻게 되어 network의 표현력이 감소.

γ , β를 학습하게 함으로써 network가 non-linearity를 유지할 수 있도록 해줌.

* Note

$$\gamma^{(k)} = \sqrt{\mathrm{Var}[x^{(k)}]}$$
$$\beta^{(k)} = \mathrm{E}[x^{(k)}]$$

Network가 parameter를 위와 같은 값으로 학습할 경우, 단순히 N(0,1)의 분포 또한 가질 수 있음.

This Week

- Dimension of γ, β
  - Multi layer perceptron

: Network가 multi layer perceptron 이라고 할 때, layer의 input dimension이 k라면 γ, β 의 dimension 또한 k임.

  - Convolutional neural network

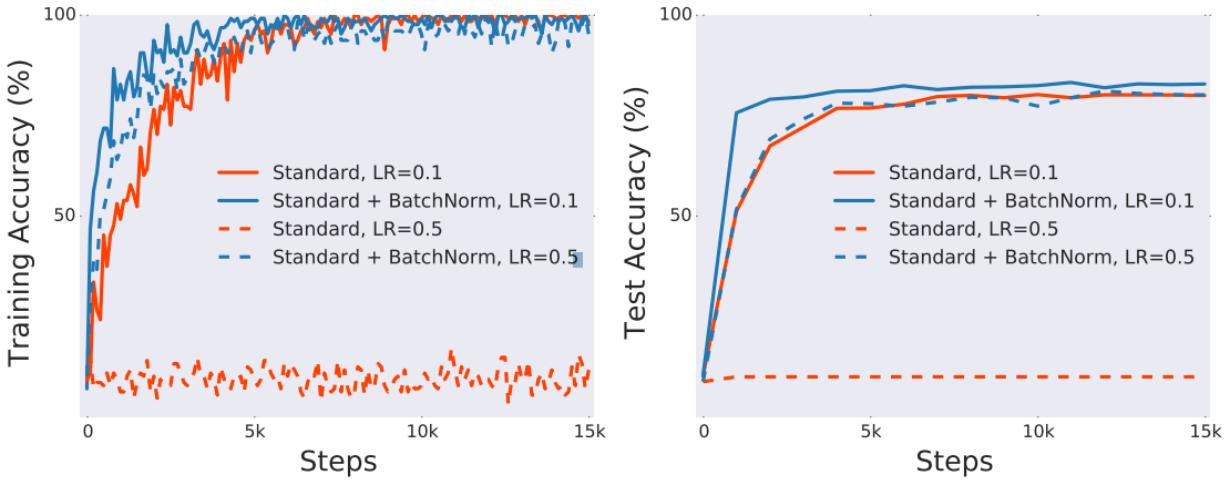
: CNN의 경우 convolution의 성질을 유지시키고 싶기 때문에, 각 channel을 기준으로 각각의 γ, β 를 학습시킴.
γ, β 의 dimension이 channel의 depth를 따르기 때문에 적은 개수의 parameter를 추가함으로써 성능을 비약적으로 올릴 수 있음.

또한, convolutional layer의 출력 (Wx+bias) 에서 bias 의 역할을 β 가 완벽히 대신 할 수 있기 때문에 batchnorm과 CNN을 함께 사용할 경우엔 convolutional layer에 bias parameter를 두지 않아도 됨.

```
layers += [nn.Conv2d(in_channels = in_channels,out_channels = out_channels, bias = False, kernel_size = (3,3), stride = (1,1), padding = (1,1)),
           nn.BatchNorm2d(x),
           nn.ReLU(),]
```

This Week

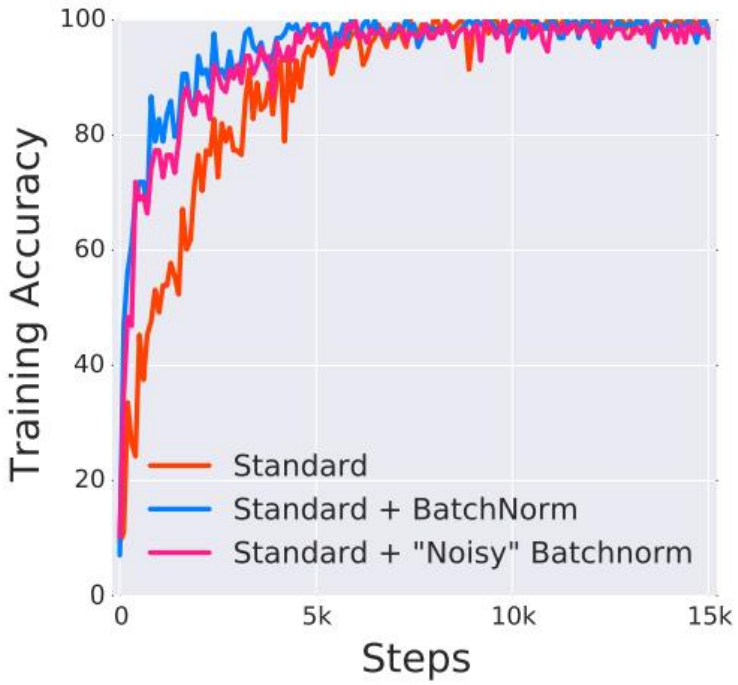
- Batch normalization의 잘 알려진 장점
  - Model의 학습 속도(training speed)를 증가 시킴.
  - Weight initialization에 대한 model의 민감도를 감소 시킴.
  - Model의 hyperparameter setting에 대한 부담을 감소 시킴. (Ex: 더 큰 learning rate를 사용할 수 있음)

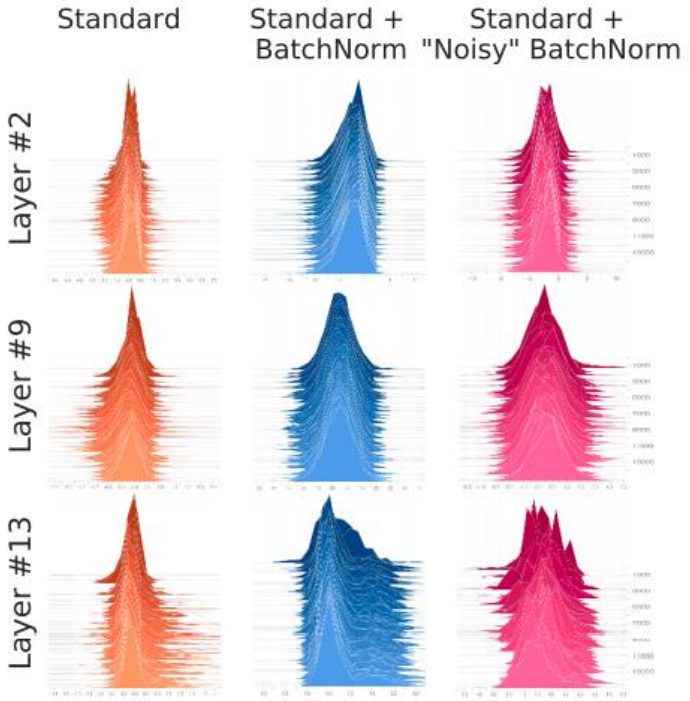


- " 정말 Batch normalization의 성능 향상은 ICS의 감소로 부터 기인하는가? "
  - How Does Batch Normalization Help Optimization?[2]
    : 기존의 Batch normalization에 대한 해석이 잘못되었음을 실험적으로 입증하고 새로운 해석을 제시함.

This Week

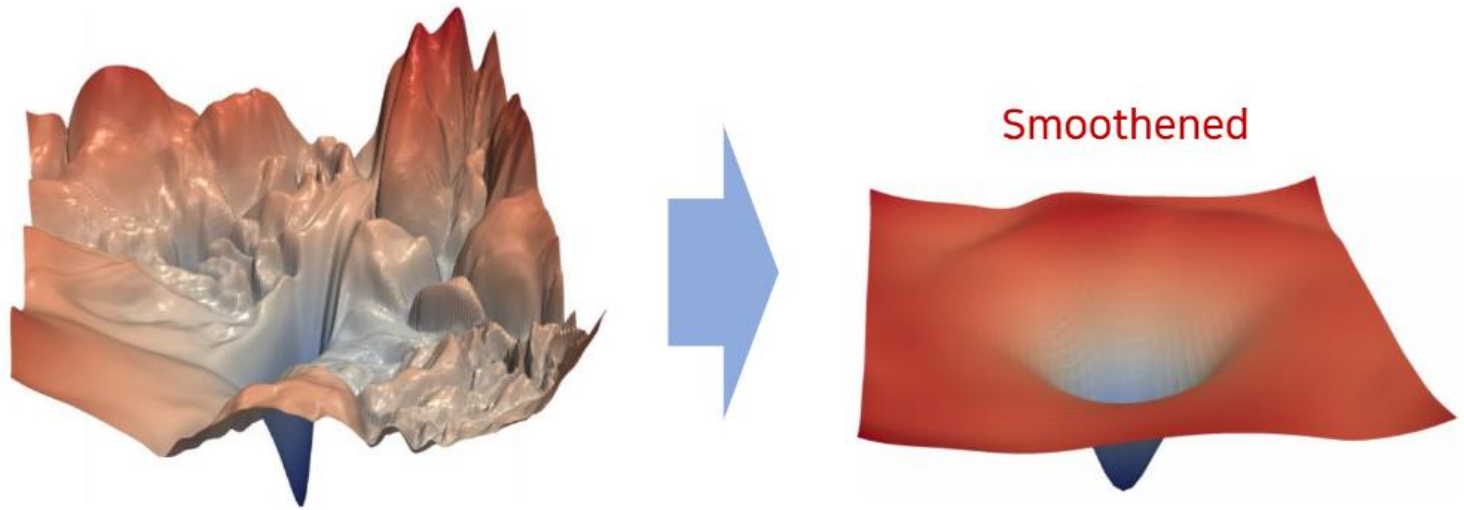
- How Does Batch Normalization Help Optimization?[2]
  - 각 layer에서 ICS를 강제로 발생시켜도 일반 network보다 성능이 우수할 수 있다는 점과, Batch norm이 ICS를 감소시키지 못함을 실험적으로 입증함
- Batch norm layer 직후에 random noise를 삽입하여 ICS를 강제로 발생시킴





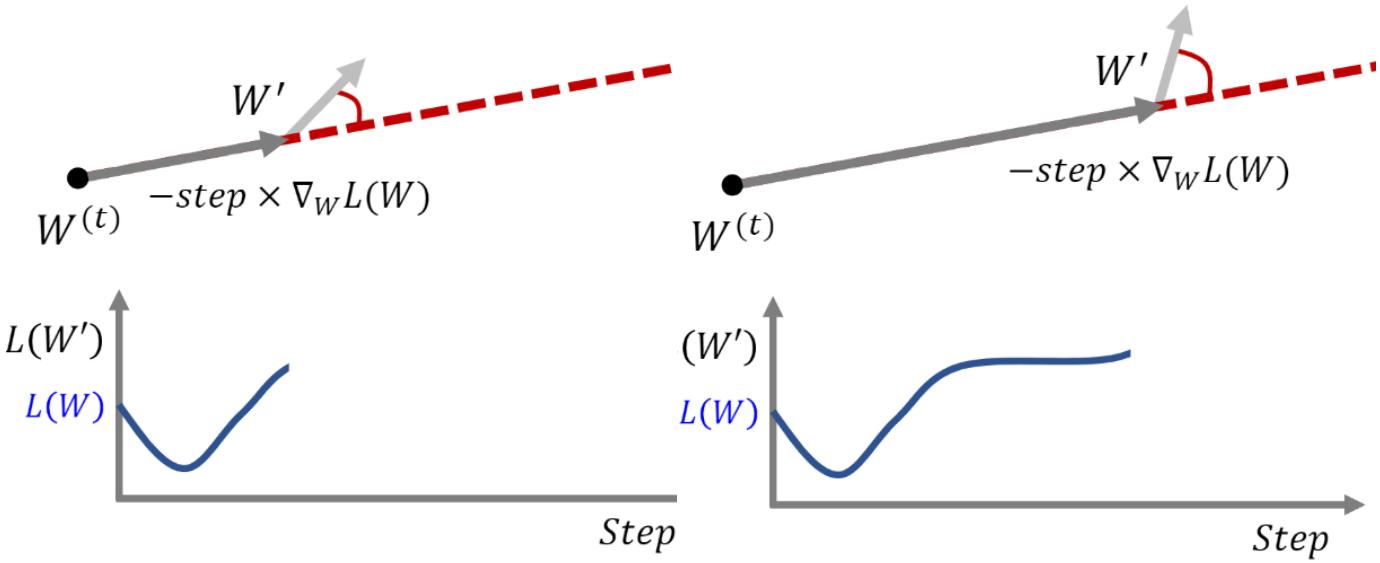
This Week

- How Does Batch Normalization Help Optimization?[2]
  - Batch normalization의 성능 향상은 optimization landscape의 smoothing 효과로 부터 기인한다.
  - Optimization landscape : Weight에 따라 loss를 visualize 한 것.



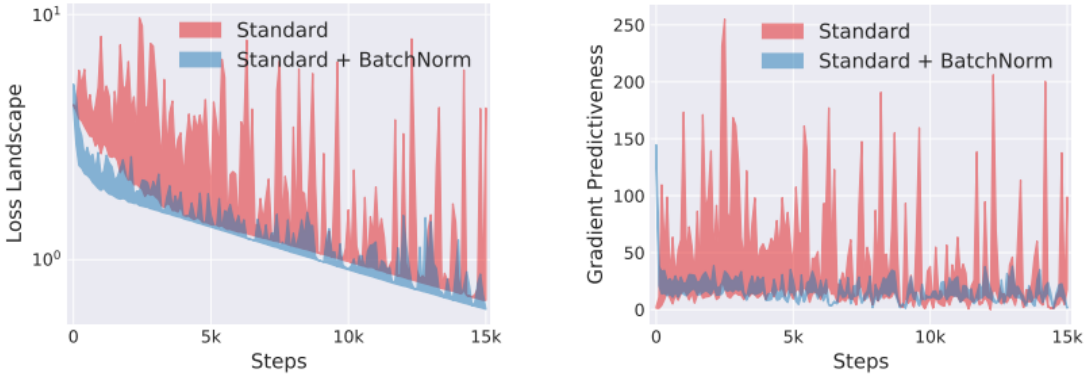
This Week

- Optimization landscape
  - 현 시점에서 weight의 update 방향에 대해 step의 크기를 바꿔가며 update를 진행한 후 loss의 변화와 향후 weight의 update 방향을 측정.
    - Changes in gradient : Big differences imply less reliable gradient.
    - Loss variation : Large fluctuations make optimization hard.



This Week

- Smoothing effect



(a) loss landscape (b) gradient predictiveness

- Gradient Predictiveness
  - 특정 시점에서 얻은 기울기에 대해 큰 step으로 weight update가 이루어져도, updat후의 gradien의 방향이 다르지 않다면 해당 gradien를 신뢰할 수 있고, 보다 안정적으로 학습을 진행 할 수 있음.
  - 즉, 현재의 기울기 방향으로 큰 스텝(step)만큼 이동 한 뒤에도 이동한 뒤의 기울기의 방향과 유사할 가능성에 따라 gradient predictiveness를 측정.
- Loss landscape
  - Smoothing effect로 인해 특정 시점에서 찾은 gradient의 방향대로 weight update가 많이 이루어진다고 해도 loss 값이 요동치지 않으며 일관성 있는 변화양상을 띔.

[1] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in Proceedings of the 32nd International Conference on Machine Learning, Lille, France, Jul. 2015, vol. 37, pp. 448-456
[2] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry, "How Does Batch Normalization Help Optimization?," in Advances in Neural Information Processing Systems, 2018, vol. 31

This week

- Batch normalization의 잘 알려진 장점

  - Model의 학습 속도(training speed)를 증가 시킴
    1.Normalization: 각 feature가 weight update에 끼치는 영향이 치중되지 않게 함으로써 optimization을 도움.
    2.Smoothing effect : Reliable한 gradient에 따라 weight를 update하기 때문에 적은 수의 update 만으로도 빠르게 global minimum에 도달할 수 있음.

  - Weight initialization에 대한 model의 민감도를 감소 시킴.
    : 이전 출력 layer에 대한 dependency가 감소하므로, weight initialization이 network에 주는 영향력 또한 감소함.

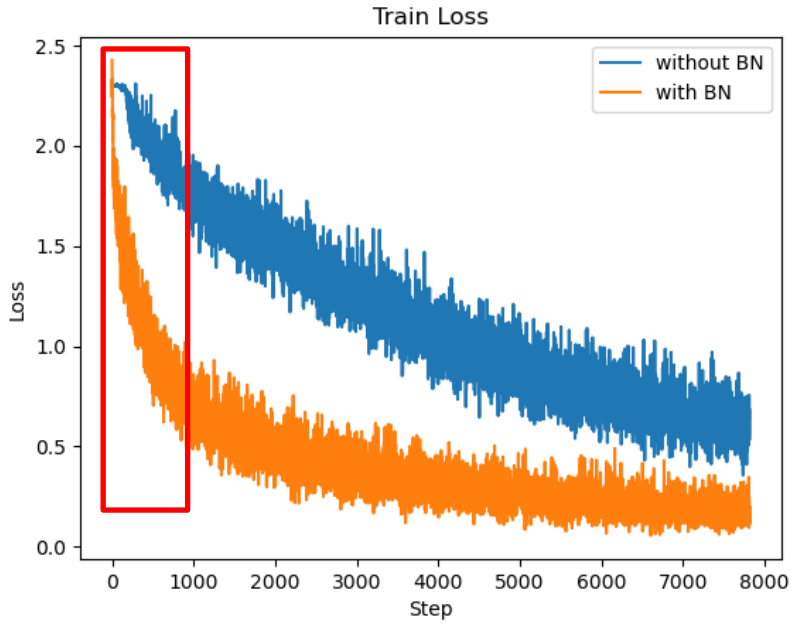
  - Model의 hyperparameter setting에 대한 부담을 감소 시킴. (Ex: 더 큰 learning rate를 사용할 수 있음)
    : Smoothing effect로 인해 더 큰 step(=더 큰 learning rate)을 사용해도 loss가 fluctuate하지 않고 global minimum에 도달할 수 있음.

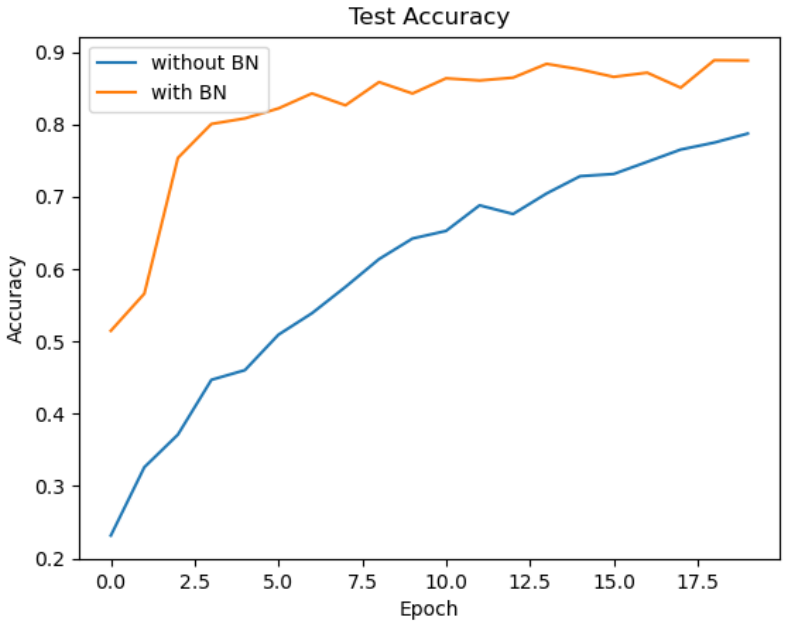
This Week

- Experimental result
  - Comparison between “Model with BN” and “Model without BN”
    - CIFAR10 classification
    - ResNet (ResNet-50), VGGnet (VGG11)

This Week

- ResNet50
  - epoch : 20 , optimizer: SGD, loss: CE, learning rate: 0.01





This Week

- VGGnet
  - epoch : 50 , optimizer: SGD, loss: CE, learning rate: 0.01