210917 SPARK LAB Seminar

Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift

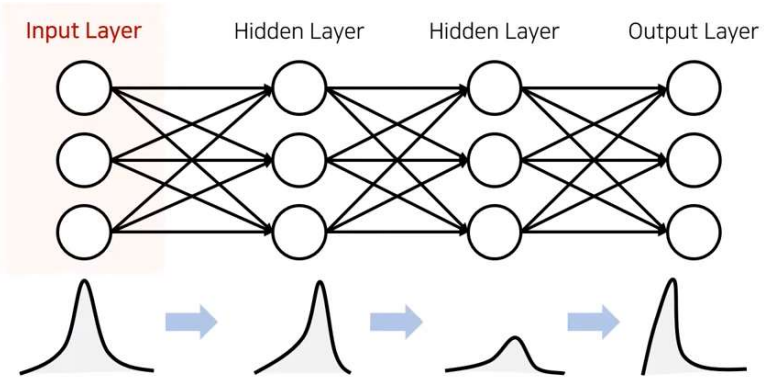
Sergey Ioffe SIOFFE@GOOGLE.COM
Christian Szegedy SZEGEDY@GOOGLE.COM
Google, 1600 Amphitheatre Pkwy, Mountain View, CA 94043

김정훈

Abstract

- Research background
  - **Distribution** of each layer's inputs **changes** during training, as the parameters of the previous layers change

- Internal covariate shift (ICS) : problems with the research background
  - Slows down the training
  - Requiring lower learning rates
    - Gradient Vanishing
    - Gradient Exploding
  - Careful parameter initialization



- The solution : normalizing layer inputs
  - normalization for each training mini-batch

Solve the problem!
(+ eliminate dropout)
14 times fewer training steps!!!

1. Introduction

- SGD(Stochastic Gradient Descent)

$$\theta = \arg min_{\theta} \frac{1}{N} \sum_{i=1}^{N} l(x_i, \theta)$$

  - $x_{1...N}$ is the training data set
  - Careful tuning, learning rate, and initial parameter value
  - Small changes to the network parameters amplify as the network becomes deeper

- Mini-batch
  - Batch : 1. 명사 (일괄적으로 처리되는) 집단[무리]
  - Mini-batch :

$$\frac{1}{\mathrm{m}} \sum_{\mathrm{i}=1}^{\mathrm{m}} \frac{\partial \mathrm{l}(\mathrm{x_i}, \theta)}{\partial \theta}$$

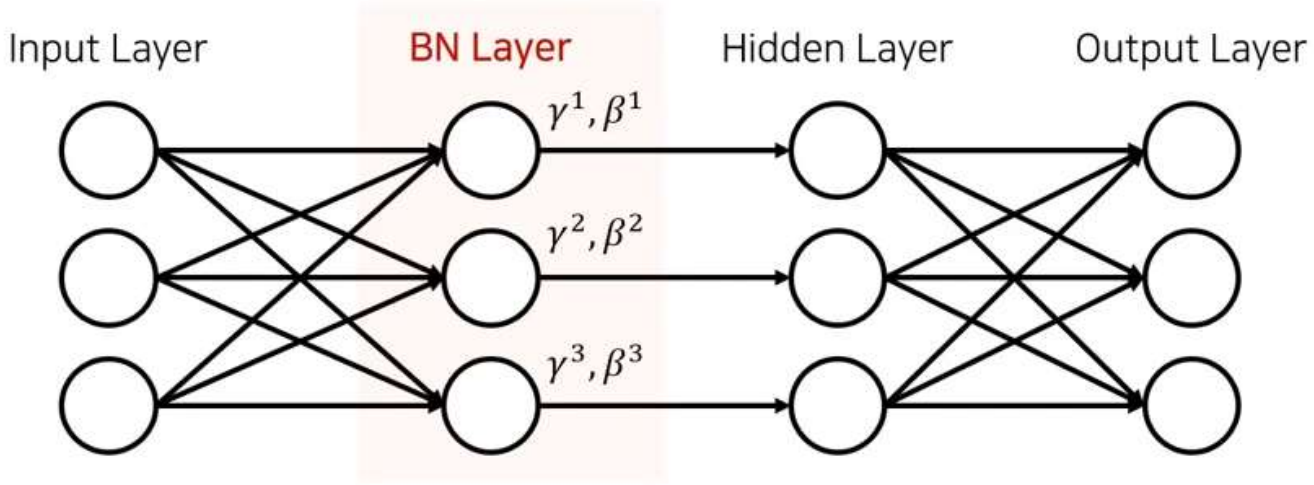
  - More efficient (**parallel computing**)

Jackson, S. R., et al. "A kinematic analysis of goal-directed prehension movements executed under binocular, monocular, and memory-guided viewing conditions." *Visual Cognition* 4.2 (1997): 113-142.

Internal Covariate Shift(ICS)

- Internal Covariate Shift(ICS)
  - Covariate shift (Shimodaira, 2000)
  - Change in the distributions of internal nodes of a deep network
  - Eliminating it offers a **promise** of faster training

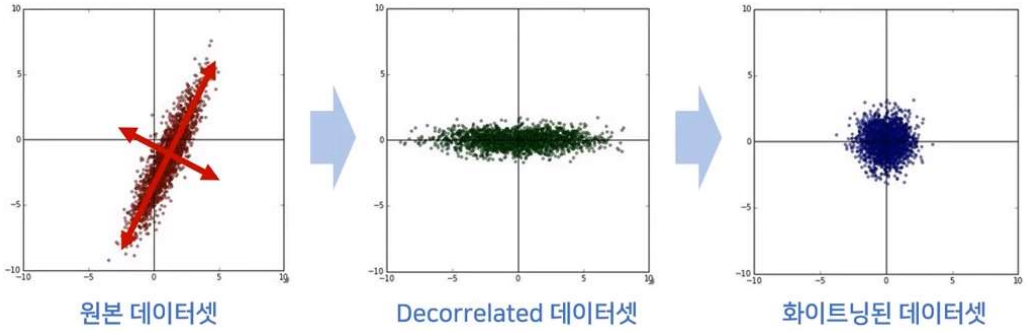
- Batch Normalization
  - Dramatically accelerates the training
  - Fixing the means and variances of layer inputs
  - Reducing the dependence of gradients on the scale and **initial values**
  - This allows us to use much **higher learning rates**
  - Reducing the need for Dropout
  - Preventing saturating



https://www.youtube.com/watch?v=58fuWVu5DVU

2. Towards Reducing Internal Covariate Shift

- Whitening (LeCun et al., 1998b)
  - Network training converges faster if its inputs are whitened
  - Linearly transformed to have zero means and unit variances and achieving the fixed distributions
  - However, Reducing the effect of the gradient step(**difficult to apply as an internal network area**)
  - Training data : $\hat{x} = x - E[x]$ where $x = u + b$
  - Then, it will update $b \leftarrow b + \Delta b$
  - The model blows up



- To address this issue
  - $\hat{x} = Norm(x, \chi)$
  - x be a layer input, χ : set of training data set
  - $Cov[x] = E_{x \in \chi}[xx^T] - E[x]E[x]^T$
  - inverse square root, $Cov[x]^{\frac{-1}{2}}(x - E[x])$
  - This motivates us to seek an alternative
  - We want to a preserve the information

LeCun, Y., Bottou, L., Orr, G., and Muller, K. Efficient backprop. In Orr, G. and K., Muller (eds.), Neural Networks: Tricks of the trade. Springer, 1998b.
https://cs231n.stanford.edu/

3. Normalization via Mini-Batch Statistics

- Normalizing the inputs of a sigmoid would **constrain** them to the linear regime of the **nonlinearity**
- To address this,
  - d-dimensional input $x = (x^{(1)} \dots x^{(d)})$, each activation $x^{(k)}$, a pair of parameters $\gamma^{(k)}$, $\beta^{(k)}$

$$\hat{x}^{(k)} = \frac{x^{(k)} - E[x^{(k)}]}{\sqrt{Var[x^{(k)}] + \epsilon}}$$

$$y^{(k)} = \gamma^{(k)} \hat{x}^{(k)} + \beta^{(k)}$$

- restore the representation power, recover the original activations

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

- Each mini-batch produces estimates of the mean and variance of each activation
- This way, fully participate in the gradient backpropagation

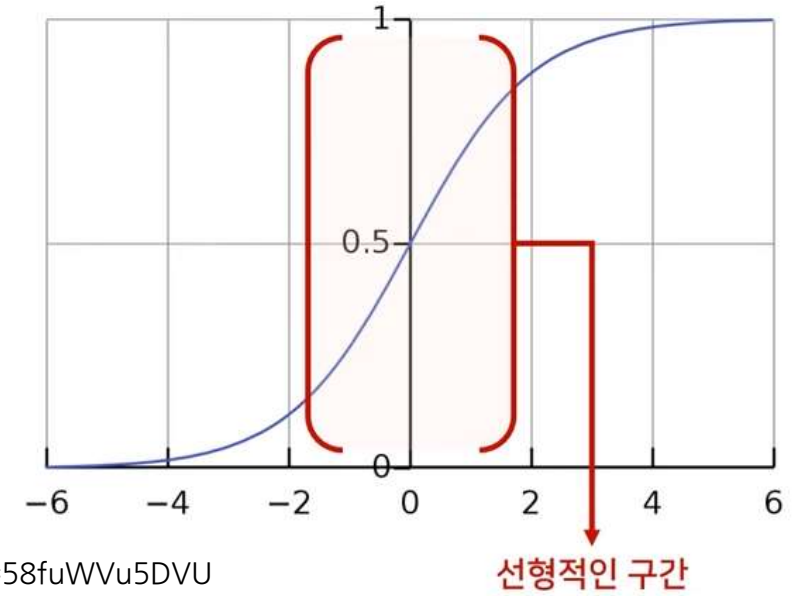
$$\mathrm{B} = \{x_{1 \dots m}\}$$

$$BN_{\gamma,\beta} : x_{1 \dots m} \rightarrow y_{1 \dots m}$$

Algorithm 1 : Batch Normalizing Transform

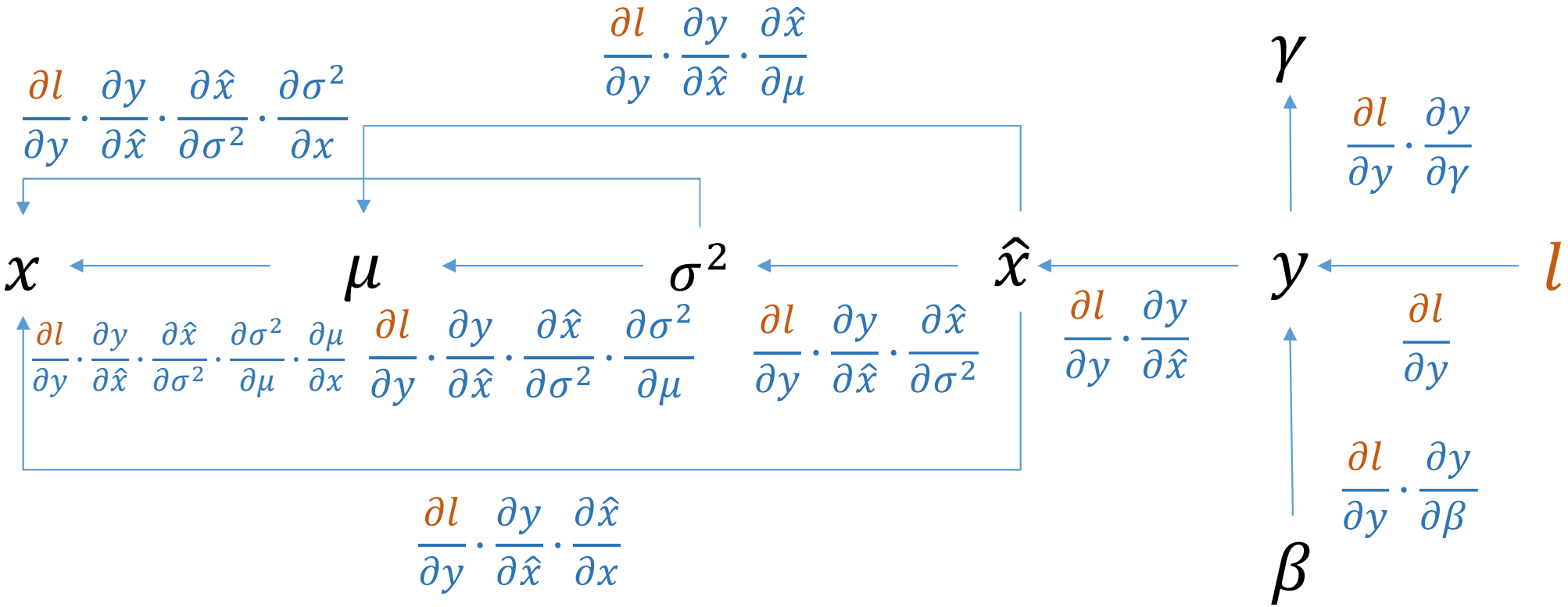
- Input: Values of x over a mini-batch: $\mathrm{B} = \{x_{1\ldots m}\}$
  Parameters to be learned: γ, β
- Output : $\{y_i = BN_{\gamma,\beta}\ (x_i)\}$

  - $\mu_{\mathrm{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$ // mini-batch mean
  - $\sigma_{\mathrm{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathrm{B}})^2$ // mini-batch variance
  - $\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathrm{B}}}{\sqrt{\sigma_{\mathrm{B}}^2 + \epsilon}}$ // normalize
  - $y_i \leftarrow \gamma\hat{x}_i + \beta \equiv BN_{\gamma,\beta}(x_i)$ // scale and shift



https://www.youtube.com/watch?v=58fuWVu5DVU

Chain rule


$\frac{\partial l}{\partial y}\cdot\frac{\partial y}{\partial \hat{x}}\cdot\frac{\partial \hat{x}}{\partial \mu}$
$\frac{\partial l}{\partial y}\cdot\frac{\partial y}{\partial \hat{x}}\cdot\frac{\partial \hat{x}}{\partial \sigma^2}\cdot\frac{\partial \sigma^2}{\partial x}$
γ
$\frac{\partial l}{\partial y}\cdot\frac{\partial y}{\partial \gamma}$
x
μ
σ^2
$\hat{x}$
y
l
$\frac{\partial l}{\partial y}\cdot\frac{\partial y}{\partial \hat{x}}\cdot\frac{\partial \hat{x}}{\partial \sigma^2}\cdot\frac{\partial \sigma^2}{\partial \mu}\cdot\frac{\partial \mu}{\partial x}$
$\frac{\partial l}{\partial y}\cdot\frac{\partial y}{\partial \hat{x}}\cdot\frac{\partial \hat{x}}{\partial \sigma^2}\cdot\frac{\partial \sigma^2}{\partial \mu}$
$\frac{\partial l}{\partial y}\cdot\frac{\partial y}{\partial \hat{x}}\cdot\frac{\partial \hat{x}}{\partial \sigma^2}$
$\frac{\partial l}{\partial y}\cdot\frac{\partial y}{\partial \hat{x}}$
$\frac{\partial l}{\partial y}$
$\frac{\partial l}{\partial y}\cdot\frac{\partial y}{\partial \beta}$
β
$\frac{\partial l}{\partial y}\cdot\frac{\partial y}{\partial \hat{x}}\cdot\frac{\partial \hat{x}}{\partial x}$

Algorithm 2: Training a Batch-Normalized Network

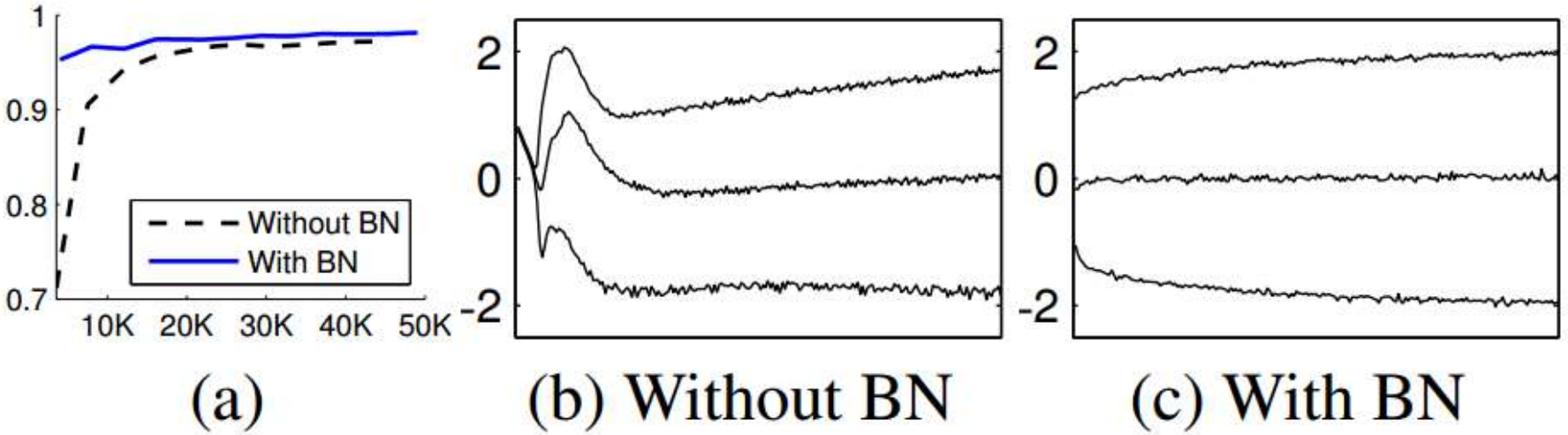
- Input: Network N with trainable parameters Θ;
  subset of activations $\{x^{(k)}\}_{k=1}^{K}$
- Output : Batch-normalized network for inference, N_{BN}^{inf}

1: $N_{BN}^{tr} \leftarrow N$ // **Training** BN network
2: for k = 1…K do
3: Add transformation $y^{(k)} = BN_{\gamma^{(k)},\beta^{(k)}}(x^{(k)})$ to N_{BN}^{tr} (Alg.1)
4: Modify each layer in N_{BN}^{tr} with input $x^{(k)}$ to take $y^{(k)}$ instead
5: end for
6: Train N_{BN}^{tr} to optimize the parameters $\Theta \cup \{\gamma^{(k)}, \beta^{(k)}\}_{k=1}^{K}$
7: $N_{BN}^{inf} \leftarrow N_{BN}^{tr}$ // Inference BN network with **frozen** parameters
8: for k = 1…K do
9: // Process multiple training mini-batches B, each of size m, and average over them:
10: $E[x] \leftarrow E_{\mathrm{B}}[\mu_{\mathrm{B}}]$
$\mathrm{Var}[x] \leftarrow \frac{m}{m-1} E_{\mathrm{B}}[\sigma_{\mathrm{B}}^2]$
11: In N_{BN}^{inf}, replace the transform $y = BN_{\gamma,\beta}(x)$ *with*

$$y = \frac{\gamma}{\sqrt{Var[x] + \epsilon}} \cdot x + (\beta - \frac{\gamma E[x]}{\sqrt{Var[x] + \epsilon}})$$

4. Experiments

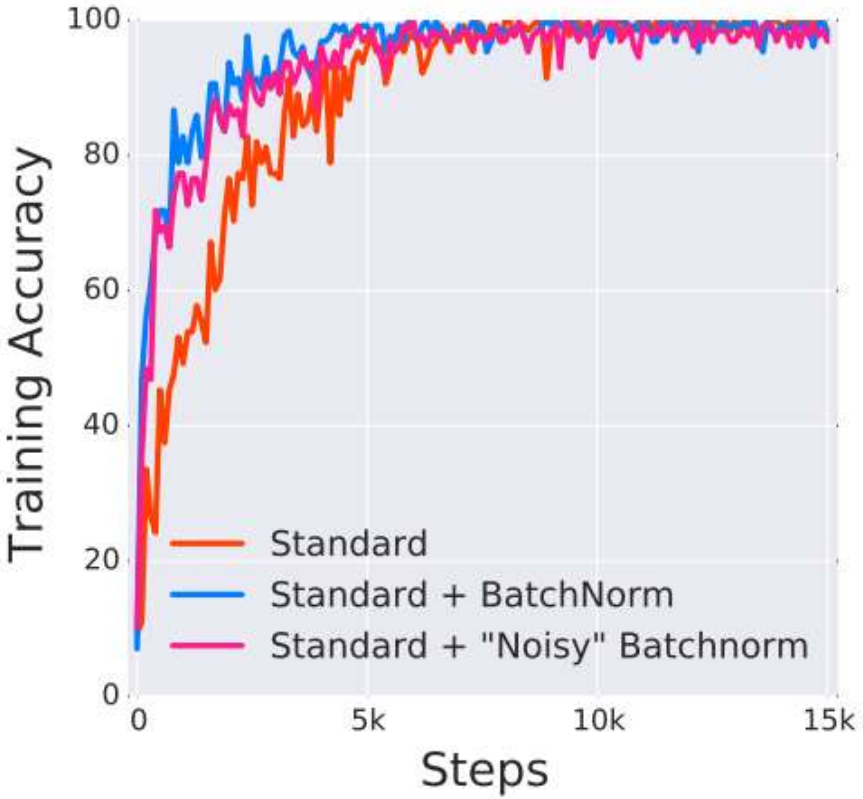
- (a) The test accuracy of the MNIST network trained with and without Batch Normalization
- (b,c) Batch Normalization makes the distribution more stable and reduces the internal covariate shift

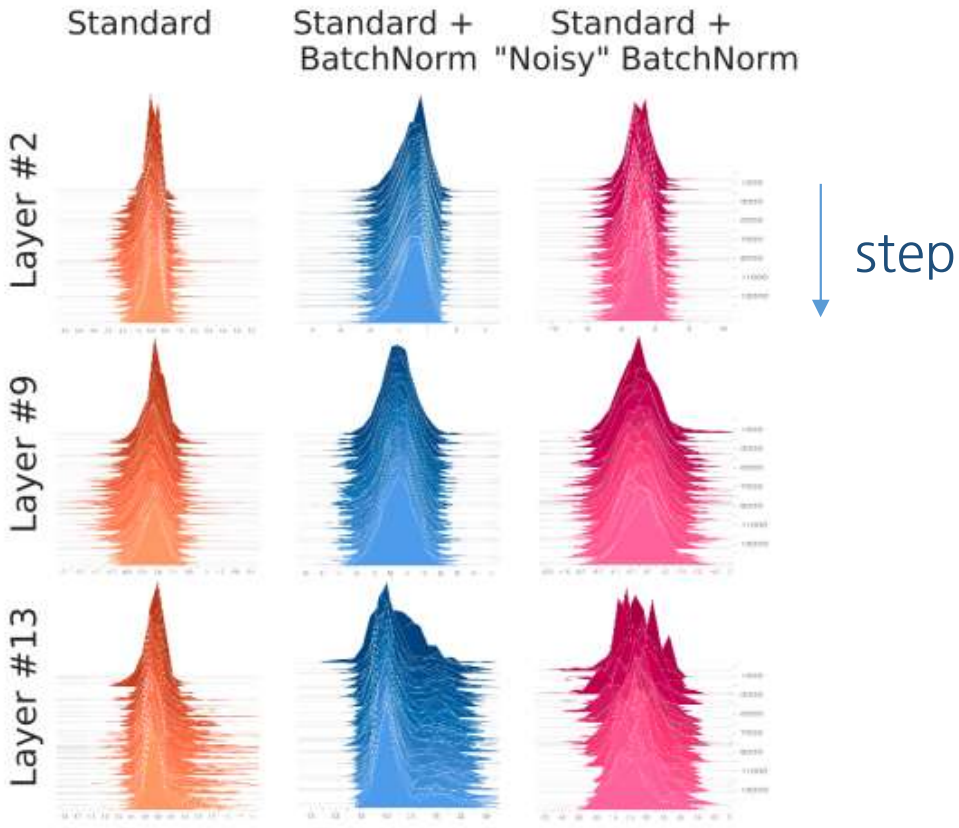


(a) (b) Without BN (c) With BN

How does batch normalization help optimization?

- The “noisy” BN model **nearly matches the performance** of standard BN model, despite complete distributional instability



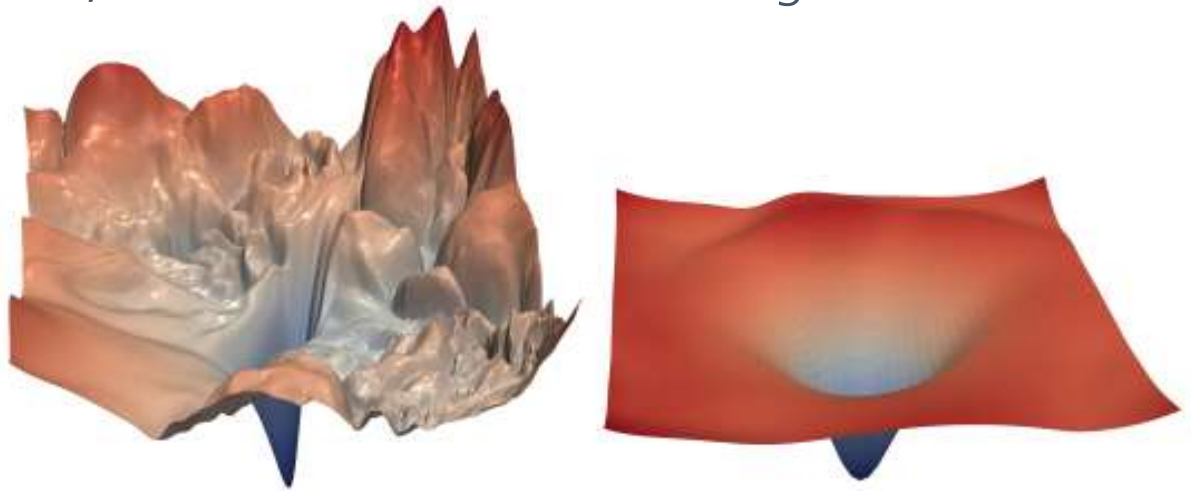


- Contribution of this paper is that there does not seem to be any link between the performance gain of BN and reduction of ICS.

Santurkar, Shibani, et al. "How does batch normalization help optimization?." Proceedings of the 32nd international conference on neural information processing systems. 2018.

Visualizing the loss landscape of neural nets

- Lipschitz-continuous function
  - A function that does not increase the distance between two points above a certain ratio
  - Ex) improved Lipschitzness of the Loss gradient
  - Widely believed connection between the performance of BN and ICS, but because of stable Lipschitzness.
- Residual connections (Smoothing effect)
  - Filter normalization scheme is used to enable comparisons of sharpness/flatness between the two figures



[The loss surfaces of ResNet-56]

Li, Hao, et al. "Visualizing the loss landscape of neural nets." arXiv preprint arXiv:1712.09913 (2017).

5. Conclusion

- Novel mechanism for **dramatically accelerating** the training of deep networks
  - Normalizing activations, normalization in the network architecture itself
  - BN adds only two extra parameters per feature map(The number of parameters does **not increase significantly**)
  - BN can be trained with saturating nonlinearities
  - Do not require Dropout(generalized performance has improved)
  - BN can learn stably even with a large running rate

- However, the batch normalization has seen the smoothing effect rather than decreasing the internal coverage shift(ICS).

- BN is still used as the main component of the network.