Batch Normalization

Betty Shea

UBC MLRG

03 - March - 2021

Paper

Santurkar, S., Tsipras, D., Ilyas, A. and Madry, A. **How Does Batch Normalization Help Optimization?** In *Advances in Neural Information Processing Systems* 2018.

`https://proceedings.neurips.cc/paper/2018/file/905056c1ac1dad141560467e0a99e1cf-Paper.pdf`

This term's MLRG

MLRG intro talk:
"[Batch normalization] works amazingly well. But we know almost nothing about it."

"Machine learning has become alchemy."

-Ali Rahimi, NeurIPS 2017 Test of Time Award speech[1]

[1] A transcript is here.

Today's talk

1. Batch Normalization (BN)

2. Santurkar et al. paper

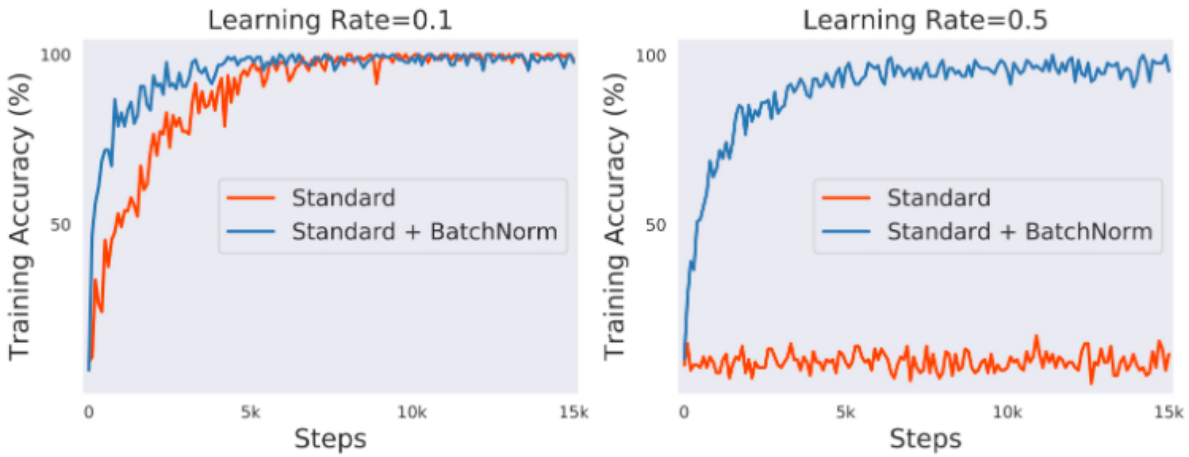
Batch Normalization (BN)

Why use BN?

Observed benefits:

- Trains faster (higher learning rates)
- Generalizes better (regularizes)
- Less sensitive to initialization

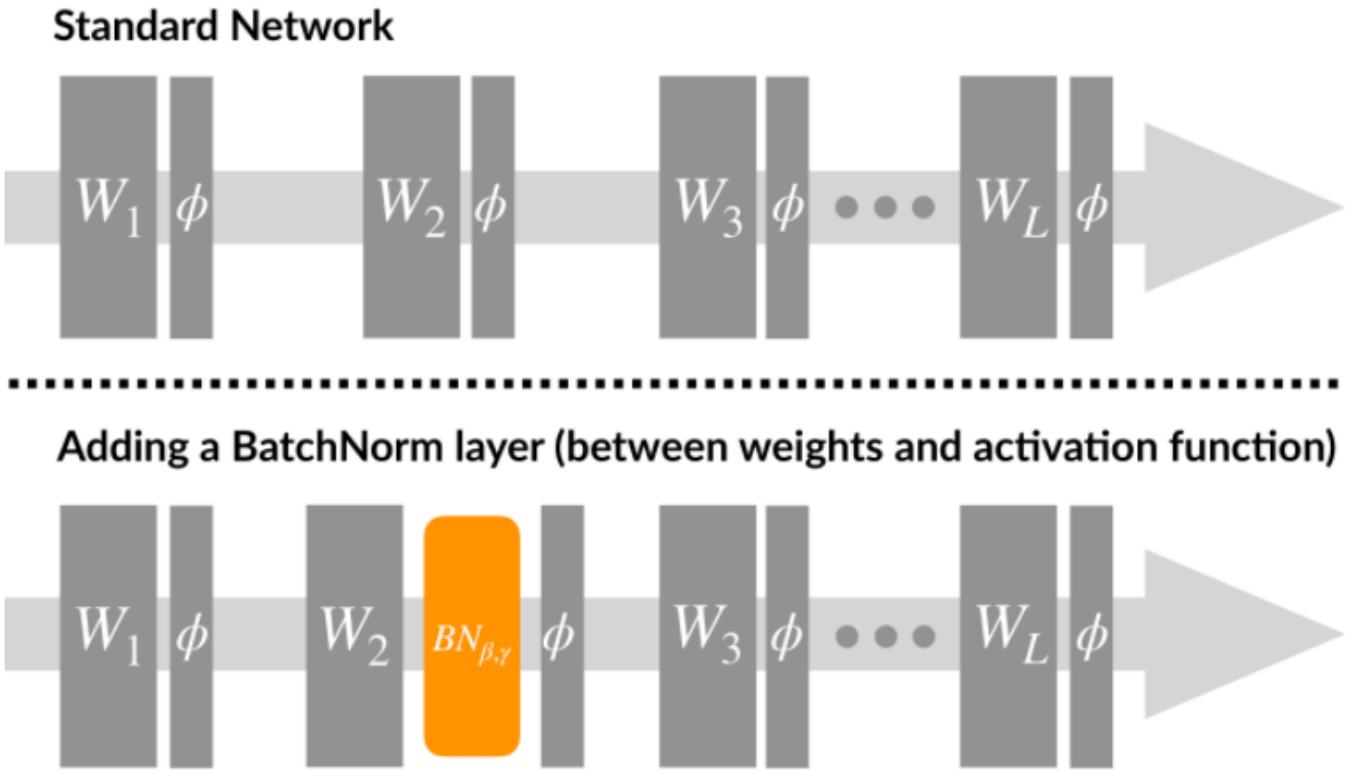
$\Longrightarrow$ Used more often than not in neural networks.



* https://gradientscience.org/batchnorm/

Ioffe and Szegedy 2015

Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *ICML*. `http://proceedings.mlr.press/v37/ioffe15.html`



* `https://gradientscience.org/batchnorm/`

BN transform

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.

BN transform

Normalizing each feature j independently is not costly

$$\hat{x}^{(j)} = \frac{x^{(j)} - E[x^{(j)}]}{\sqrt{\text{Var}[x^{(j)}]}}$$

Learn a pair of $(\gamma^{(j)}, \beta^{(j)})$ for every feature j used in the scale and shift step

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

Note

1. Just normalizing may change what a layer can represent.
2. Every minibatch contributes to every $\gamma^{(j)}, \beta^{(j)}$ pair.

Internal covariate shift (ICS)

Covariate shift: Inputs, or **covariate**, to a learning system change in distribution.

Internal covariate shift: Inputs to a part of the learning systems, e.g. a layer, change in distribution.

"We refer to the change in the distributions of internal nodes of a deep network, in the course of training, as *Internal Covariate Shift*."

- Ioffe and Szegedy 2015

Claims:

- ICS worse in deeper layers
- ICS slows down training
- BN reduces ICS

Some things are unclear

1. What is ICS?
2. How does ICS slow down training?
3. How does BN reduce ICS?

Santurkar et al. paper

Contributions:

- Defines ICS
- Show that ICS does not necessarily slow down training
- Shows that BN may not reduce ICS
- Give alternative explanation of how BN helps training

Focuses more on better training and less on better generalizability.

Both empirical and theoretical analysis.

Santurkar et al. 2018

Experimental setup

Task: image classification problem on CIFAR-10 data.

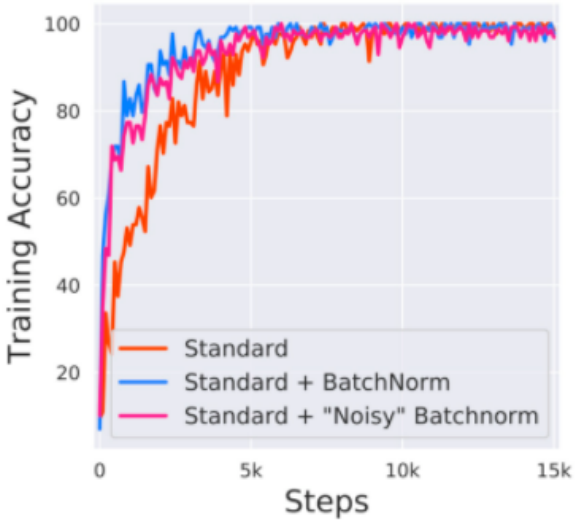
Two types of network architecture

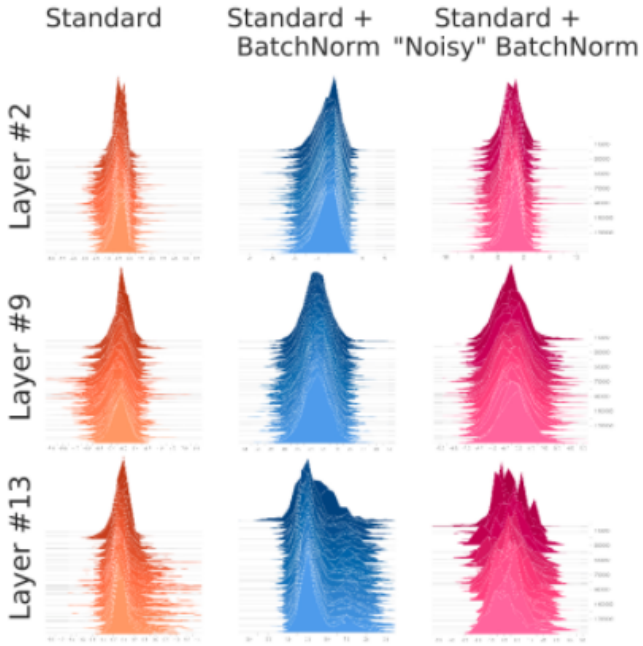
- VGG with stochastic gradient descent
- deep linear network (DLN) trained with full-batch gradient descent

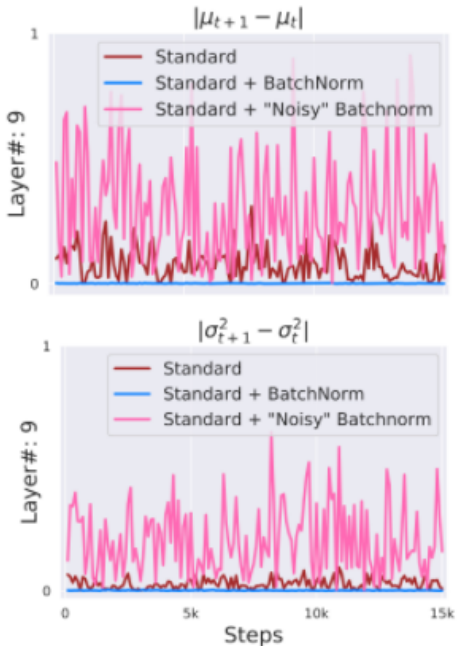
Three varieties within each architecture type:

- Standard
- BatchNorm
- Noisy BatchNorm

ICS does not slow down training







- Noisy BN had higher ICS and higher training accuracy than Standard
- Same noise injected into Standard meant training was impossible

Proposes ICS definition [2]

Definition (Internal Covariate Shift)

Let $\mathcal{L}$ be the loss, $W_1^{(t)}, \ldots, W_k^{(t)}$ be the parameters of each of the k layers and $(x^{(t)}, y^{(t)})$ be the batch of input-label pairs used to train the network at time t. We define *internal covariate shift (ICS) of activation i* at time t to be the difference $\|G_{t,i} - G'_{t,i}\|_2$ where

$$G_{t,i} = \nabla_{W_i^{(t)}} \mathcal{L}\left(W_1^{(t)}, \ldots, W_k^{(t)}; x^{(t)}, y^{(t)}\right)$$

$$G'_{t,i} = \nabla_{W_i^{(t)}} \mathcal{L}\left(W_1^{(t+1)}, \ldots, W_{i-1}^{(t+1)}, W_i^{(t)}, \ldots, W_k^{(t)}; x^{(t)}, y^{(t)}\right)$$

- Optimization-oriented definition: networks trained by first-order methods
- Not really distribution change of layer inputs
- Reflects the change in the optimization landscape of a layer

[2] Definition 2.1 in paper

BN does not reduce ICS (new definition)

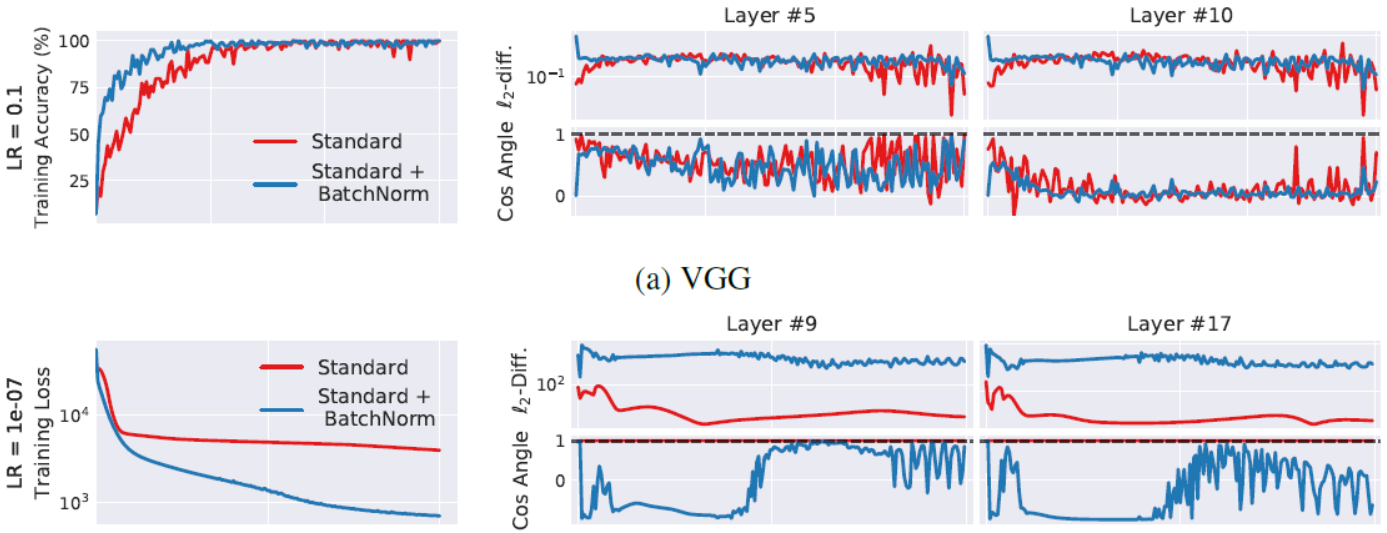


Figure 3: Measurement of ICS (as defined in Definition 2.1) in networks with and without BatchNorm layers. For a layer we measure the cosine angle (ideally 1) and ℓ_2-difference of the gradients (ideally 0) before and after updates to the preceding layers (see Definition 2.1). Models with BatchNorm have similar, or even worse, internal covariate shift, despite performing better in terms of accuracy and loss. (Stabilization of BatchNorm faster during training is an artifact of parameter convergence.)

Why does BN work?

Reparameterizes optimization problem to make its landscape more smooth.

Theoretical analysis:

- First order properties. (Theorem 4.1)
- Second order properties. (Theorem 4.2)

Effect of BN on smoothness

Theorem 4.1 (Increases L-smoothness.) For a BN network with loss $\hat{\mathcal{L}}$ and an identical non-BN network with (identical) loss $\mathcal{L}$,

$$\|\nabla_{y_j}\hat{L}\|^2 \leq \frac{\gamma^2}{\sigma^2}\left(\|\nabla_{y_j}\mathcal{L}\|^2 - \frac{1}{m}\langle 1, \nabla_{y_j}\mathcal{L}\rangle^2 - \frac{1}{\sqrt{m}}\langle \nabla_{y_j}\mathcal{L}, \hat{y}_j\rangle^2\right)$$

Theorem 4.2. (Reduces the second order term in the Taylor expansion.)

$$\left(\nabla_{y_j}\hat{\mathcal{L}}\right)^\intercal \frac{\delta\hat{\mathcal{L}}}{\delta y_j \delta y_j}\left(\nabla_{y_j}\hat{\mathcal{L}}\right) \leq \frac{\gamma^2}{\sigma^2}\left[\left(\nabla_{y_j}\mathcal{L}\right)^\intercal \frac{\partial\mathcal{L}}{\partial y_j \partial y_j}\left(\nabla_{y_j}\mathcal{L}\right) - \frac{1}{m\gamma}\langle \nabla_{y_j}\mathcal{L}, \hat{y}_j\rangle\|\nabla_{y_j}\hat{\mathcal{L}}\|^2\right]$$

- Both additive and multiplicative effect
- σ tends to be large in practice and so multiplicative effect may be significant
- Even if $\gamma = \sigma$, there are still benefits from the additive effect

Takeaways

1. Batch normalization does not necessarily reduce internal covariate shift.
2. Lower internal covariate shift does not mean better training.
3. The reason why BN works well may have to do with making the optimization landscape smoother.

Thank you

Whitening

How about whitening, i.e. unit diagonal covariance (jointly on features)?

Too costly to calculate the covariance matrix or its inverse. And mini-batches / stochastic optimization setting adds complexity because you want every batch to contribute to the whole covariance matrix for all features. Simpler when dealing with each feature separately.

Normalizing features independently shown to help even if features are correlated. Section 1.4.3 Normalizing the Inputs. *Neural Networks: Tricks of the Trade, 2nd edition, 2012*

Why not just normalize?

Just normalizing may result in a layer losing expressivity.

"For instance, normalizing the inputs of a sigmoid $\left[f(x) = 1/(1 + e^{-x})\right]$ would constrain them to the linear regime of the nonlinearity."

Make sure that the transformation inserted in the network can represent the identity transform.

Can recover original parameters by setting $\gamma^{(j)} = \sqrt{\text{Var}[x^{(j)}]}$ and $\beta^{(j)} = E[x^{(k)}]$.

Covariate shift

"The situation where the training input points and test input points follow *different* distributions but the conditional distribution of output values given input points is unchanged is called the *covariate shift*."[3]

Given covariate x, response y and conditional density $q(y|x)$. Let $q_1(x)$ be the density of x in the validation data and $q_0(x)$ be the density x in the training data. "The situation $q_0(x) \neq q_1(x)$ will be called covariate shift in distribution...."[4]

[3] Sugiyama, M, Krauledat, M & Müller, K. Covariate shift adaptation by importance weighted cross validation. *JMLR* 2007

[4] H. Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference* 2000.