BATCH NORMALIZATION

SOURANGSHU BHATTACHARYA

CSE, IIT KHARAGPUR

WEB: HTTPS://CSE.IITKGP.AC.IN/~SOURANGSHU/

EMAIL: SOURANGSHU@CSE.IITKGP.AC.IN

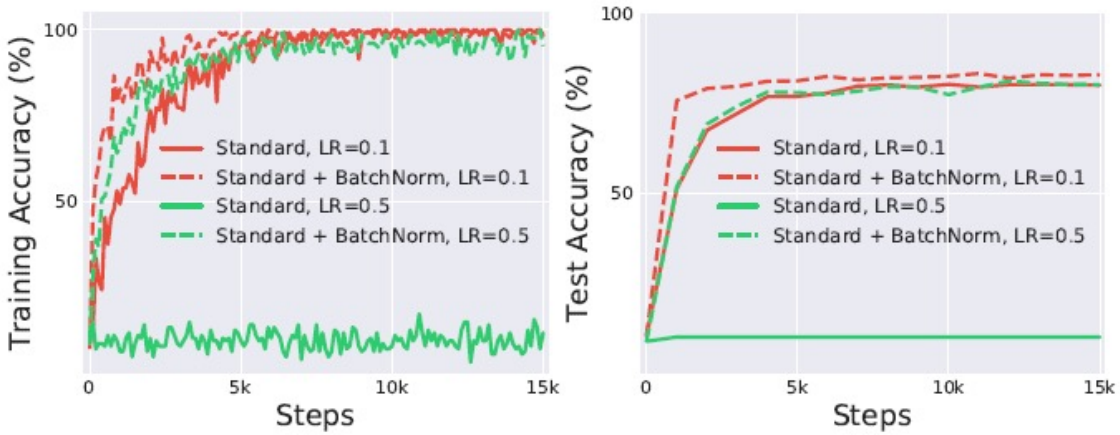
BATCH NORMALIZATION

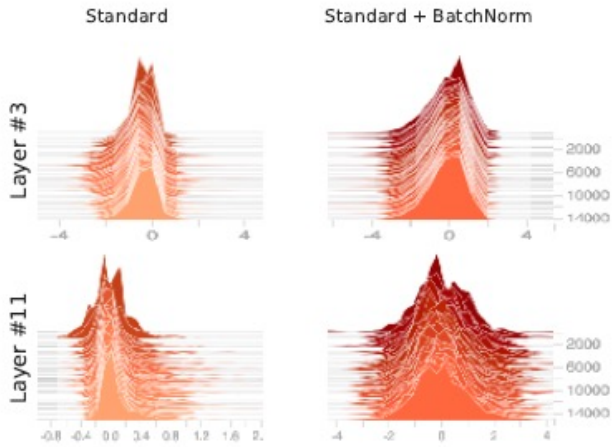
Slides taken from Jude Shavlik: http://pages.cs.wisc.edu/~shavlik/cs638_cs838.html

BATCH NORMALIZATION: OTHER BENEFITS IN PRACTICE

- BN reduces training times. (Because of less Covariate Shift, less exploding/vanishing gradients.)
- BN reduces demand for regularization, e.g. dropout or L2 norm.
  - Because the means and variances are calculated over batches and therefore every normalized value depends on the current batch. I.e. the network can no longer just memorize values and their correct answers.)
- BN allows higher learning rates. (Because of less danger of exploding/vanishing gradients.)
- BN enables training with saturating nonlinearities in deep networks, e.g. sigmoid. (Because the normalization prevents them from getting stuck in saturating ranges, e.g. very high/low values for sigmoid.)

INTERNAL COVARIATE SHIFT





WHY THE NAÏVE APPROACH DOES NOT WORK?

- Normalizes layer inputs to zero mean and unit variance. *whitening*.
- Naive method: Train on a batch. Update model parameters. Then normalize. Doesn't work: Leads to exploding biases while distribution parameters (mean, variance) don't change.

  - If we do it this way gradient always ignores the effect that the normalization for the next batch would have
  - i.e. : "The issue with the above approach is that the gradient descent optimization does not take into account the fact that the normalization takes place"

DOING IT THE "CORRECT WAY" IS TOO EXPENSIVE!

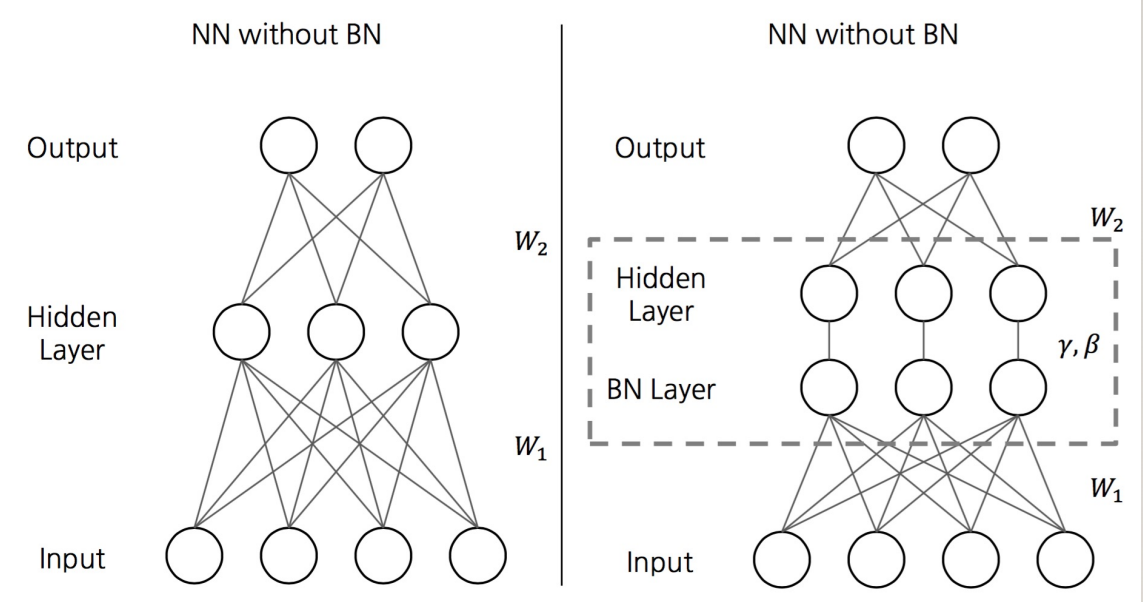
- A proper method has to include the current example batch *and* somehow all previous batches (all examples) in the normalization step.

- This leads to calculating in covariance matrix and its inverse square root. That's expensive. The authors found a faster way!

THE PROPOSED SOLUTION: TO ADD AN EXTRA REGULARIZATION LAYER

we introduce, for each activation $x^{(k)}$, a pair of parameters $\gamma^{(k)}, \beta^{(k)}$, which scale and shift the normalized value:

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



A new layer is added so the gradient can "see" the normalization and make adjustments if needed.

ALGORITHM SUMMARY: NORMALIZATION VIA MINI-BATCH STATISTICS

- Each feature (component) is normalized individually
- Normalization according to:
  - componentNormalizedValue = (componentOldValue - E[component]) / sqrt(Var(component))
- A new layer is added so the gradient can "see" the normalization and made adjustments if needed.
  - The new layer has the power to learn the identity function to de-normalize the features if necessary!
  - Full formula: newValue = gamma * componentNormalizedValue + beta (gamma and beta learned per component)
- E and Var are estimated for each mini batch.
- BN is fully differentiable.

THE BATCH TRANSFORMATION: FORMALLY FROM THE PAPER.

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

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

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

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

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

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

THE FULL ALGORITHM AS PROPOSED IN THE PAPER

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

Output: Batch-normalized network for inference, $N_{\text{BN}}^{\text{inf}}$

1: $N_{\text{BN}}^{\text{tr}} \leftarrow N$ // Training BN network

2: **for** $k = 1 \ldots K$ **do**

3: Add transformation $y^{(k)} = \text{BN}_{\gamma^{(k)},\beta^{(k)}}(x^{(k)})$ to $N_{\text{BN}}^{\text{tr}}$ (Alg. 1)

4: Modify each layer in $N_{\text{BN}}^{\text{tr}}$ with input $x^{(k)}$ to take $y^{(k)}$ instead

5: **end for**

6: Train $N_{\text{BN}}^{\text{tr}}$ to optimize the parameters $\Theta \cup \{\gamma^{(k)}, \beta^{(k)}\}_{k=1}^{K}$

7: $N_{\text{BN}}^{\text{inf}} \leftarrow N_{\text{BN}}^{\text{tr}}$ // Inference BN network with frozen // parameters

8: **for** $k = 1 \ldots K$ **do**

9: // For clarity, $x \equiv x^{(k)}, \gamma \equiv \gamma^{(k)}, \mu_{\mathcal{B}} \equiv \mu_{\mathcal{B}}^{(k)}$, etc.

10: Process multiple training mini-batches $\mathcal{B}$, each of size m, and average over them:

$$\text{E}[x] \leftarrow \text{E}_{\mathcal{B}}[\mu_{\mathcal{B}}]$$

$$\text{Var}[x] \leftarrow \frac{m}{m-1}\text{E}_{\mathcal{B}}[\sigma_{\mathcal{B}}^2]$$

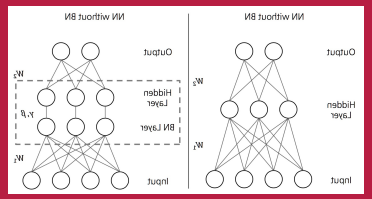
11: In $N_{\text{BN}}^{\text{inf}}$, replace the transform $y = \text{BN}_{\gamma,\beta}(x)$ with $y = \frac{\gamma}{\sqrt{\text{Var}[x]+\epsilon}} \cdot x + \left(\beta - \frac{\gamma\,\text{E}[x]}{\sqrt{\text{Var}[x]+\epsilon}}\right)$

12: **end for**

Algorithm 2: Training a Batch-Normalized Network

Alg 1 (previous slide)

Architecture modification



Note that BN(x) is different during test…

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2$$

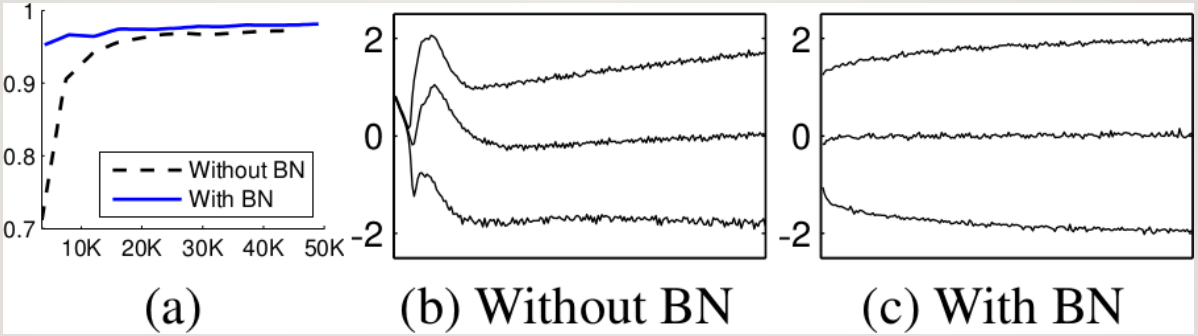
Vs.

$$\text{Var}[x] \leftarrow \frac{m}{m-1}\text{E}_{\mathcal{B}}[\sigma_{\mathcal{B}}^2]$$

BATCH NORMALIZATION: OTHER BENEFITS IN PRACTICE

- BN reduces training times. (Because of less Covariate Shift, less exploding/vanishing gradients.)
- BN reduces demand for regularization, e.g. dropout or L2 norm.
  - Because the means and variances are calculated over batches and therefore every normalized value depends on the current batch. I.e. the network can no longer just memorize values and their correct answers.)
- BN allows higher learning rates. (Because of less danger of exploding/vanishing gradients.)
- BN enables training with saturating nonlinearities in deep networks, e.g. sigmoid. (Because the normalization prevents them from getting stuck in saturating ranges, e.g. very high/low values for sigmoid.)

BATCH NORMALIZATION: BETTER ACCURACY , FASTER.



BN applied to MNIST (a), and activations of a randomly selected neuron over time (b, c), where the middle line is the median activation, the top line is the 15th percentile and the bottom line is the 85th percentile.

THANKS

QUESTIONS?

Email: sourangshu@cse.iitkgp.ac.in