Batch/Layer Normalization

2025-02-06

InHyeok Jeong

Content

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

- Layer Normalization

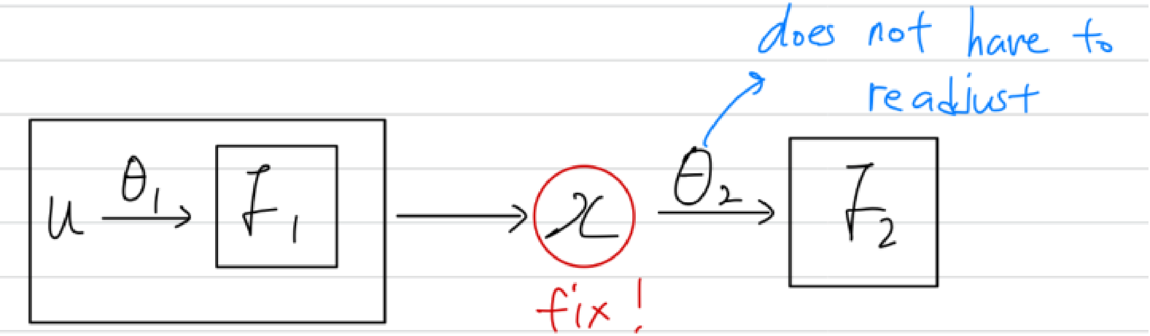
Index

- **What is Internal Covariate Shift?**
- **Goal**
- **How to reduce Internal Covariate Shift?**
  - Whiten
  - Batch Normalization
- **Advantages**
- **Accelerating BN Networks**
- **Experiments**
  - MNIST dataset
  - ImageNet classification
  - Ensemble classification
- **Conclusion**

What is Internal Covariate Shift?

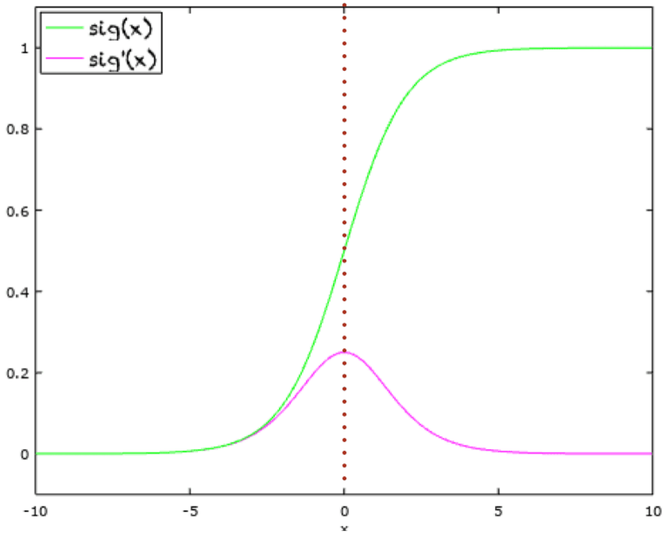
- **Covariate Shift**
  - Change in the distributions of layers' inputs
  - Require lower learning rates and careful parameter initialization
  - Can be applied to whole learning system and its part $e.g.,$ sub-network

- $l = F_2(F_1(u, \Theta_1), \Theta_2), x = F_1(u, \Theta_1) \rightarrow l = F_2(x, \Theta_2)$
- $\Theta_2 \leftarrow \Theta_2 - \frac{\alpha}{m} \sum_{i=1}^{m} \frac{\partial F_2(x_i, \Theta_2)}{\partial \Theta_2}$



What is Internal Covariate Shift?

- **Fixed distribution of inputs to a sub-network would have positive consequence for the layer outside the sub-network**
  - $e.g.,\ z = g(Wu + b),\ g(x) = \frac{1}{1+\exp(-x)}$
  - As $|x|$ increases, $g'(x)$ tends to zero
    $\rightarrow x$ moves to the saturated regime

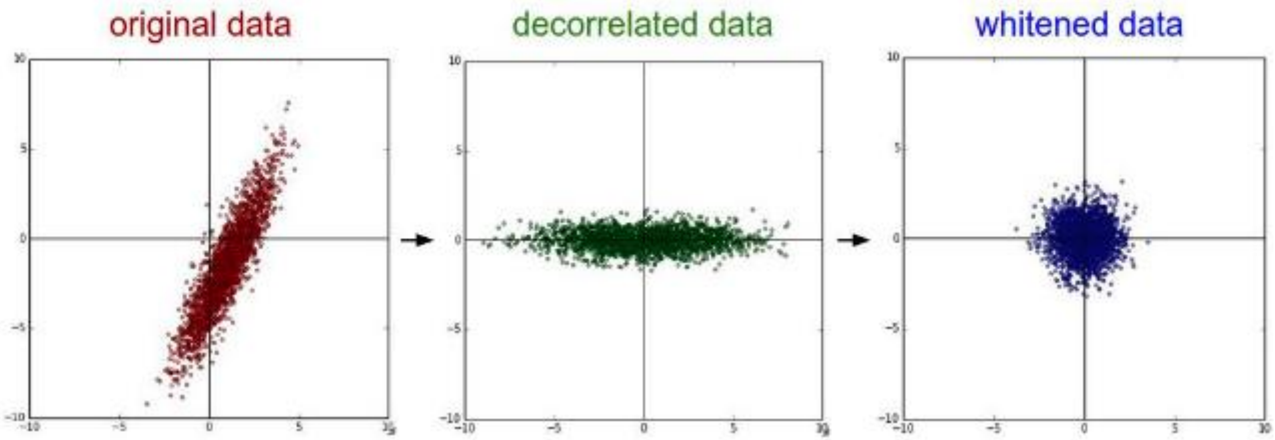


Goal

- **Reduce Internal Covariate Shift**
  - Make more stable during training
  - Accelerate training

How to reduce Internal Covariate Shift?

- **Whiten**
  - Make inputs have Zero means, unit variances
  - Decorrelate



How to reduce Internal Covariate Shift?

- **Whiten**
  - Key consideration
    - Gradient descent optimization take into account the normalization
    - $e.g., \ \hat{x} = x \ - E[x], \ x = u + b, \ X = \{x_1, \dots x_N\}, \ E[x] = \frac{1}{N}\sum_{i=1}^{N} x_i$
    - $b \leftarrow b + \ \Delta b, \ \Delta b \ \propto -\frac{\partial l}{\partial \hat{x}}$
    - $u + (b + \ \Delta b) - E[u + (b + \ \Delta b)] = u + b \ - E[u + b]$

→ The network *always* produces activations with the desired distribution!

How to reduce Internal Covariate Shift?

- **x : layer input, X : the set of inputs over the training data set**
  - $\hat{x} = Norm(x, X)$
  - For backpropagation,
    - We need to compute $\frac{\partial Norm(x,X)}{\partial \mathrm{x}}, \frac{\partial Norm(x,X)}{\partial X}$
  - Too expensive
    - Require computing Cov[x] and derivatives of transforms for backpropagation

How to reduce Internal Covariate Shift?

- **Batch Normalization**
  - Two necessary simplifications
    - Normalize each scalar feature independently
    - For a layer with d-dimensional input $x = \left(x^{(1)} \ldots x^{(d)}\right)$

$$\widehat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}$$

    - But simply normalizing each input of layer may change what the layer can represent
      → The transformation inserted in the network can represent the identity transform

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

How to reduce Internal Covariate Shift?

- **Batch Normalization**
  - Two necessary simplifications
    - *Each mini-batch produces estimates of the mean and variance* of each activation

How to reduce Internal Covariate Shift?

- **Batch Normalization**
  - Accelerate the training of the sub-network

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

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

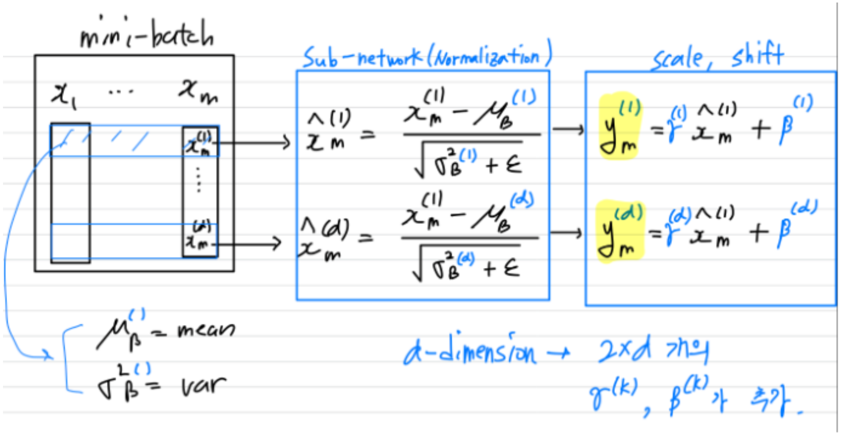
$$\mu_\mathcal{B} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

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

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

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

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



How to reduce Internal Covariate Shift?

- **Batch Normalization**
  - Can differentiate transformation

$$
\begin{aligned}
\frac{\partial \ell}{\partial \widehat{x}_i} &= \frac{\partial \ell}{\partial y_i} \cdot \gamma \\
\frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} &= \sum_{i=1}^{m} \frac{\partial \ell}{\partial \widehat{x}_i} \cdot (x_i - \mu_{\mathcal{B}}) \cdot \frac{-1}{2}(\sigma_{\mathcal{B}}^2 + \epsilon)^{-3/2} \\
\frac{\partial \ell}{\partial \mu_{\mathcal{B}}} &= \left(\sum_{i=1}^{m} \frac{\partial \ell}{\partial \widehat{x}_i} \cdot \frac{-1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \right) + \frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{\sum_{i=1}^{m} -2(x_i - \mu_{\mathcal{B}})}{m} \\
\frac{\partial \ell}{\partial x_i} &= \frac{\partial \ell}{\partial \widehat{x}_i} \cdot \frac{1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} + \frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{2(x_i - \mu_{\mathcal{B}})}{m} + \frac{\partial \ell}{\partial \mu_{\mathcal{B}}} \cdot \frac{1}{m} \\
\frac{\partial \ell}{\partial \gamma} &= \sum_{i=1}^{m} \frac{\partial \ell}{\partial y_i} \cdot \widehat{x}_i \\
\frac{\partial \ell}{\partial \beta} &= \sum_{i=1}^{m} \frac{\partial \ell}{\partial y_i}
\end{aligned}
$$

How to reduce Internal Covariate Shift?

- **Batch Normalization**
  - Once the network has been trained, we use the below normalization

$$\widehat{x} = \frac{x - \mathrm{E}[x]}{\sqrt{\mathrm{Var}[x] + \epsilon}}$$

using the population

How to reduce Internal Covariate Shift?

- Batch Normalization

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

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

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

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

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

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

end for

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

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

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

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

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

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

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

end for

Algorithm 2: Training a Batch-Normalized Network

Advantages

- **Make it possible to have a high learning rate**
  - It prevents small changes to the parameters from amplifying into larger and suboptimal changes in activations in gradients

- **Make training more resilient to the parameter scale**
  - Backpropagation through a layer is unaffected by the scale of its parameters
    - $BN(Wu) = BN\big((aW)u\big)$
    - $\frac{\partial BN\big((aW)u\big)}{\partial u} = \frac{\partial BN(Wu)}{\partial u}, \ \frac{\partial BN\big((aW)u\big)}{\partial (aW)} = \frac{1}{a} \cdot \frac{\partial BN(Wu)}{\partial W}$

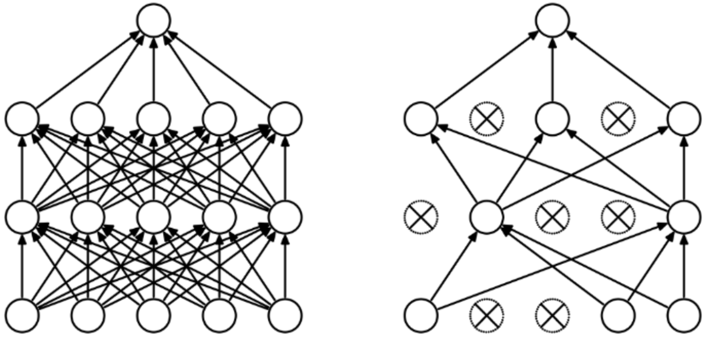
Advantages

- **Regularize the model**
  - A training example is seen in conjunction with other examples in the mini-batch
  - Training network no longer produces deterministic values for a given training example

Accelerating BN Networks

- **Increase learning rate**
  - We can achieve a training speedup from higher learning rates with no ill side effects

- **Remove or Reduce Dropout**
  - BN already fulfills some of the same goals as Dropout



(a) Standard Neural Net (b) After applying dropout.

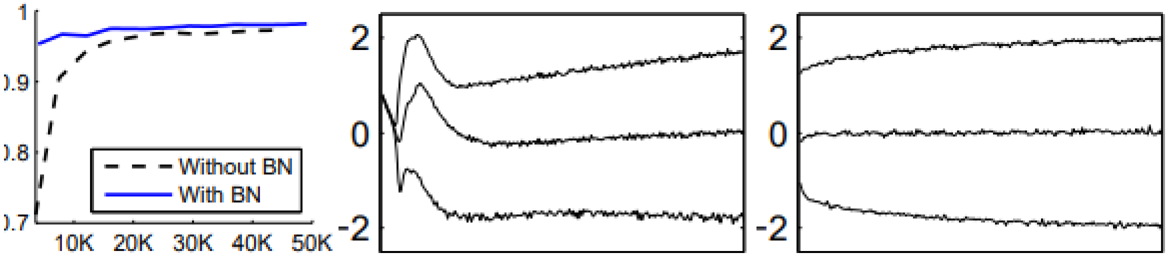
Accelerating BN Networks

- **Reduce the L_2 weight regularization**
  - $L_\lambda(w) = L(w) + \lambda||w||_2^2$
  - When L2 regularization is used with BN, the original regularization effect disappears, leaving only the effect of increasing the learning rate

- **Shuffle training examples more thoroughly**
  - Prevent the same examples from always appearing in a mini-batch together

Experiments

- MNIST dataset



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

Experiments

- ImageNet classification
  - Apply Batch Normalization to a new variant of the Inception network

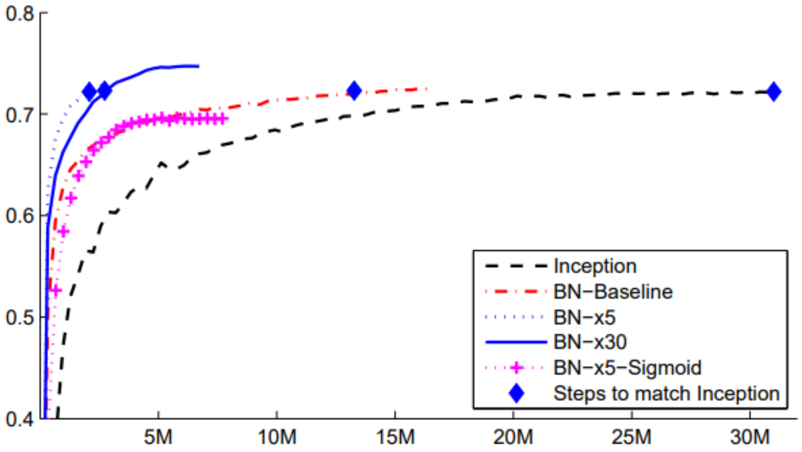


Figure 2: *Single crop validation accuracy of Inception and its batch-normalized variants, vs. the number of training steps.*

Model	Steps to 72.2%	Max accuracy
Inception	$31.0 \cdot 10^6$	72.2%
BN-Baseline	$13.3 \cdot 10^6$	72.7%
BN-x5	$2.1 \cdot 10^6$	73.0%
BN-x30	$2.7 \cdot 10^6$	74.8%
BN-x5-Sigmoid		69.8%

Figure 3: *For Inception and the batch-normalized variants, the number of training steps required to reach the maximum accuracy of Inception (72.2%), and the maximum accuracy achieved by the network.*

Experiments

- Ensemble classification

Model	Resolution	Crops	Models	Top-1 error	Top-5 error
GoogLeNet ensemble	224	144	7	-	6.67%
Deep Image low-res	256	-	1	-	7.96%
Deep Image high-res	512	-	1	24.88	7.42%
Deep Image ensemble	variable	-	-	-	5.98%
BN-Inception single crop	224	1	1	25.2%	7.82%
BN-Inception multicrop	224	144	1	21.99%	5.82%
BN-Inception ensemble	224	144	6	20.1%	**4.9%***

Figure 4: *Batch-Normalized Inception comparison with previous state of the art on the provided validation set comprising 50000 images. *BN-Inception ensemble has reached 4.82% top-5 error on the 100000 images of the test set of the ImageNet as reported by the test server.*

Conclusion

- Batch Normalization reduces Internal Covariate Shift, accelerating training and improving stability
- It enhances generalization, often eliminating the need for Dropout
- By further increasing the learning rates, removing Dropout, and applying other modifications afforded by Batch Normalization

Index

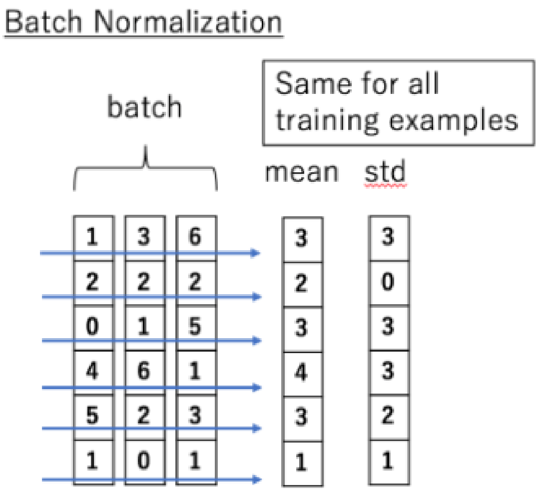
- **Disadvantages of Batch Normalization**
- **Layer Normalization**
  - Vs. Batch Normalization
  - Similarity
  - Difference
- **Experiments**
  - Order embeddings of images and language
  - Handwriting sequence generation
  - Permutation invariant MNIST
  - Convolutional Networks
- **Conclusion**

Disadvantages of Batch Normalization

- Dependent on the mini-batch size
- Not obvious how to apply it to recurrent neural networks

Layer Normalization

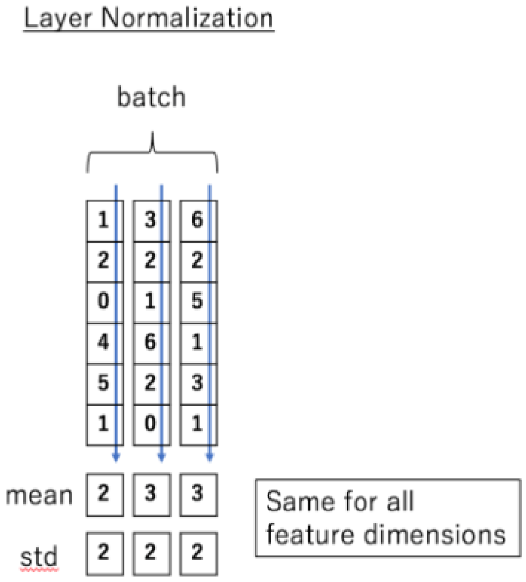
- Vs. Batch Normalization



$$\mu_i^l = \mathop{\mathbb{E}}_{\mathbf{x}\sim P(\mathbf{x})}\left[a_i^l\right] \qquad \sigma_i^l = \sqrt{\mathop{\mathbb{E}}_{\mathbf{x}\sim P(\mathbf{x})}\left[\left(a_i^l - \mu_i^l\right)^2\right]}$$

Layer Normalization

- Vs. Batch Normalization



$$\mu^l = \frac{1}{H}\sum_{i=1}^{H} a_i^l \qquad \sigma^l = \sqrt{\frac{1}{H}\sum_{i=1}^{H}\left(a_i^l - \mu^l\right)^2}$$

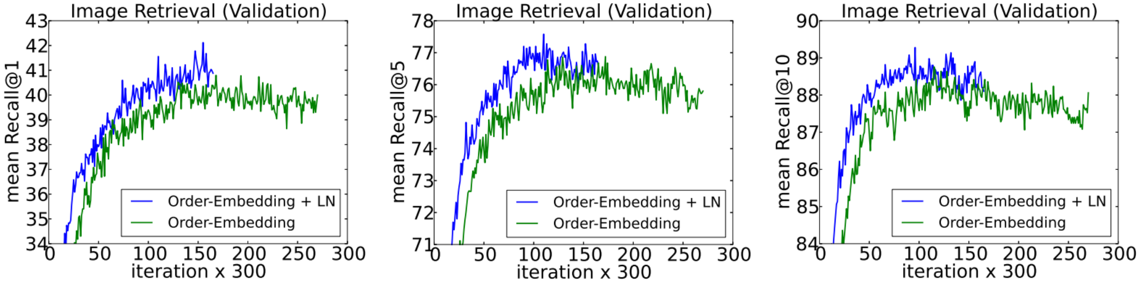
Layer Normalization

- **Similarity**
  - Give each neuron its own adaptive bias and gain

- **Difference**
  - Perform exactly the same computation at training and test times

Experiments

☐ Order embeddings of images and language



(a) Recall@1 (b) Recall@5 (c) Recall@10

Figure 1: Recall@K curves using order-embeddings with and without layer normalization.

MSCOCO								
	Caption Retrieval				Image Retrieval			
Model	**R@1**	**R@5**	**R@10**	**Mean** r	**R@1**	**R@5**	**R@10**	**Mean** r
Sym [Vendrov et al., 2016]	45.4		88.7	5.8	36.3		85.8	9.0
OE [Vendrov et al., 2016]	46.7		88.9	5.7	37.9		85.9	8.1
OE (ours)	46.6	79.3	89.1	5.2	37.8	73.6	85.7	7.9
OE + LN	**48.5**	**80.6**	**89.8**	**5.1**	**38.9**	**74.3**	**86.3**	**7.6**

Table 2: Average results across 5 test splits for caption and image retrieval. **R@K** is Recall@K (high is good). **Mean** r is the mean rank (low is good). Sym corresponds to the symmetric baseline while OE indicates order-embeddings.

Experiments

- Handwriting sequence generation

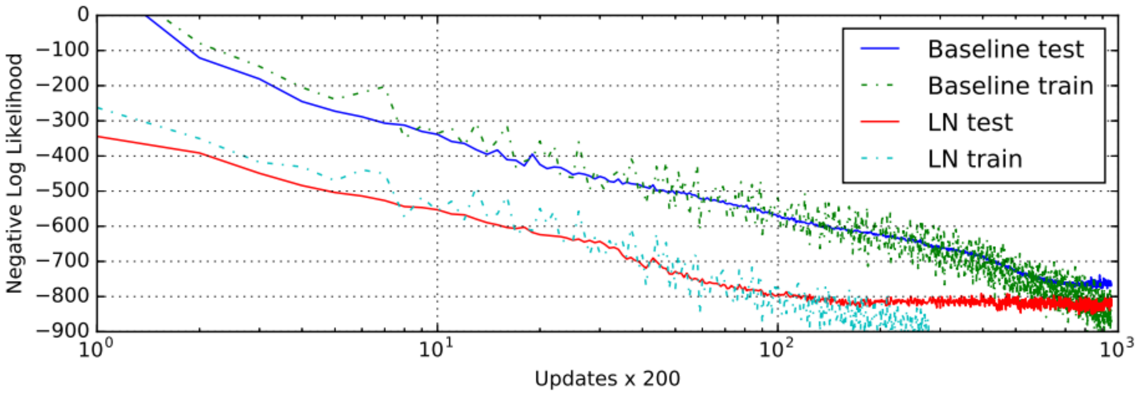


Figure 5: Handwriting sequence generation model negative log likelihood with and without layer normalization. The models are trained with mini-batch size of 8 and sequence length of 500,

Experiments

- Permutation invariant MNIST

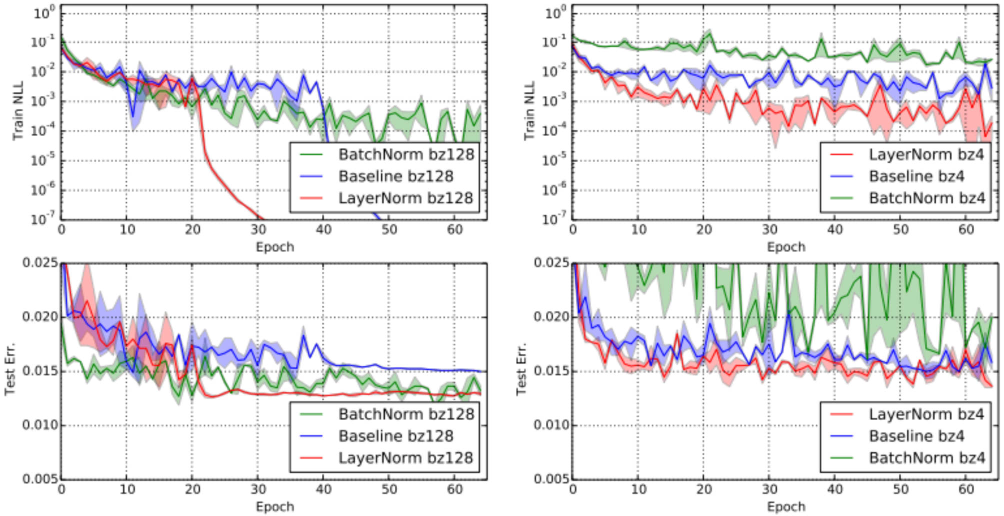


Figure 6: Permutation invariant MNIST 784-1000-1000-10 model negative log likelihood and test error with layer normalization and batch normalization. (Left) The models are trained with batch-size of 128. (Right) The models are trained with batch-size of 4.

Experiments

- **Convolutional Networks**
  - The large number of the hidden units whose receptive fields lie near the boundary of the image are rarely activated
  - Have very different statistics from the rest of the hidden units within the same layer
  - BN outperforms LN

Conclusion

- Not dependent on mini-batch size compared to Batch Normalization
- Applicable to RNN
- Further research is needed to make layer normalization work well in CNN