Inception v2

Batch Normalization:

Accelerating Deep Network Training by Reducing Internal Covariate Shift

Sergey Ioffe & Christian Szegedy, 2 Mar 2015

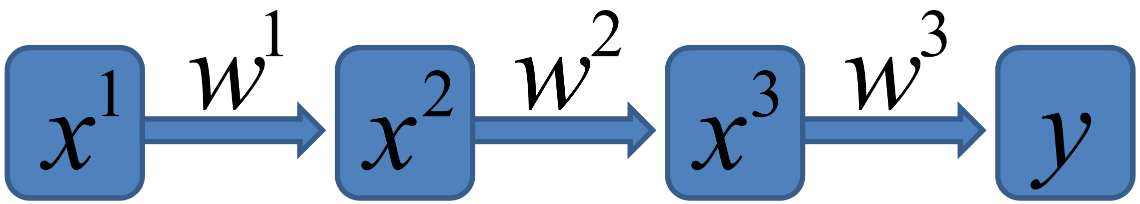
康仕承, 23 Aug 2018 at Taipei

Outline

- Internal Covariate Shift
- Batch Normalization
- Training and Testing
- Experiment (MNIST)
- Inception
- BN-Inception (Inception v2)
- Experiments

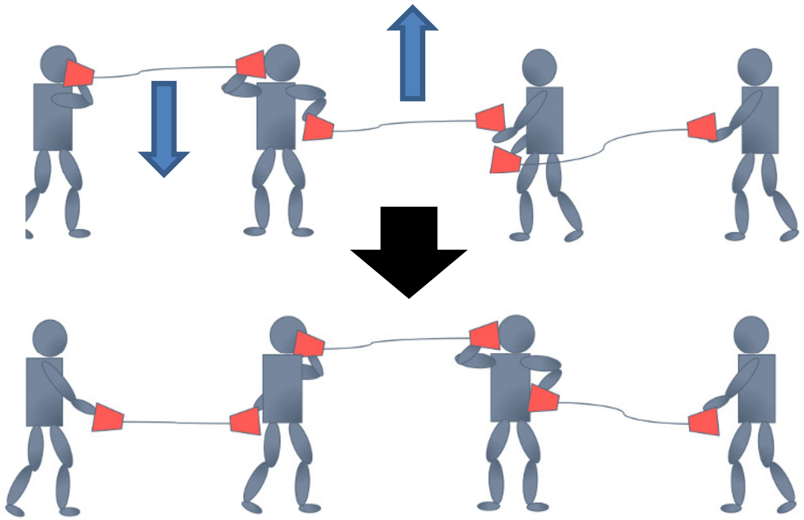
Internal Covariate Shift

- What is the "internal covariate shift"?
  - As the parameters of the previous layers change, the distribution of each layer's inputs changes.



Internal Covariate Shift

- What is the "internal covariate shift"?
  – As the parameters of the previous layers change, the distribution of each layer's inputs changes.



Internal Covariate Shift

(copy from Hung-yi Lee's ppt)

Internal Covariate Shift

- Problems:
  - Slow down the training by requiring lower learning rate and careful parameter initialization
  - Notoriously hard to train models with saturating non-linearities. (e.g. sigmoid or tanh)

Internal Covariate Shift

- Solutions:
  - ReLU, careful initialization, and small learning rate (but slow down the training)
  - Batch Normalization: make the distribution of non-linearity inputs remains more stable as the network trains, then the optimizer would be less likely to get stuck in the saturated regime, and the training would accelerate.

Batch Normalization

- Whitening transformation:
  - Transforms a vector of random variables with a known covariance matrix into a set of new variables whose covariance is the identity matrix.
- LeCun, 1998; Wiesler & Ney, 2011:
  - The network training converges faster if its inputs are whitened.

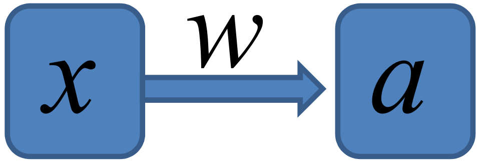
Batch Normalization

- Idea: full whitening of each layer's inputs
  - Costly and not everywhere differentiable
- Two simplifications:
  - Normalization
  - Mini-Batch

Batch Normalization

- Simplification:
  - Normalize:
    Let $x = \left(x^{(1)}, \ldots, x^{(d)}\right)$ be an input of a layer, then we set

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



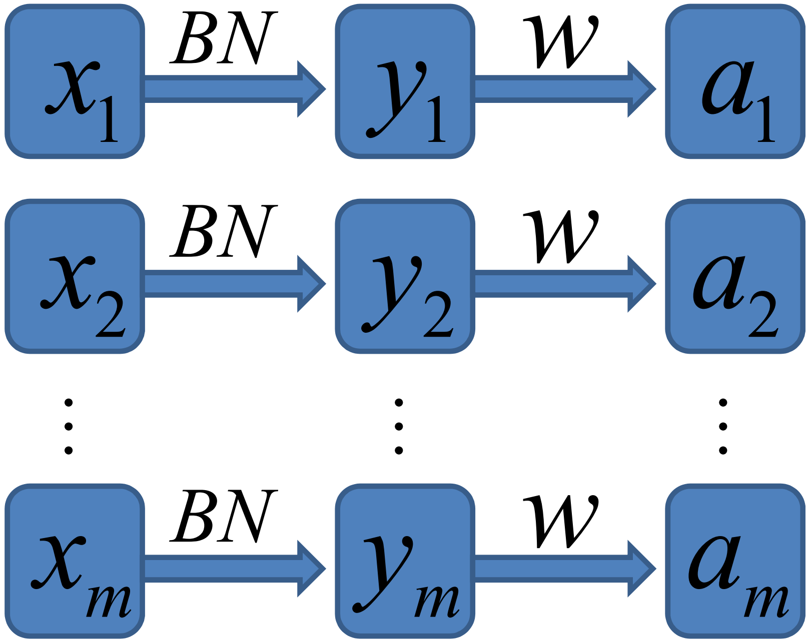
and

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

Batch Normalization

- Simplification:
  - Mini-Batches: consider a batch $\mathrm{B} = (x_1, \ldots, x_m)$

$$BN_{\gamma,\beta} : \{x_1, \ldots, x_m\} \rightarrow \{y_1, \ldots, y_m\}$$



Batch Normalization

- Simplification:
  - Mini-Batches: consider a batch $\mathrm{B} = \left(x_1, \ldots, x_m\right)$

$$\mu_{\mathrm{B}} = \frac{1}{m} \sum_{i=1}^{m} x_i$$ // mini-batch mean

$$\sigma_{\mathrm{B}}^2 = \frac{1}{m} \sum_{i=1}^{m} \left(x_i - \mu_B\right)^2$$ // mini-batch variance

$$\hat{x}_i = \frac{x_i - \mu_{\mathrm{B}}}{\sqrt{\sigma_{\mathrm{B}}^2 + \varepsilon}}$$ // normalize

$$y_i = \gamma \hat{x}_i + \beta \equiv BN_{\gamma,\beta}\left(x_i\right)$$ // scale and shift

Batch Normalization

- Remark
  - 此篇論文將BN層放在激活函數之前，因此加入scale(γ)和shift(β)一起訓練，而李宏毅老師的課程中提到，也可以將BN層放在激活函數之後，但沒測試過，故效果未知。
  - 為了讓mini-batch的平均數和標準差可代表母體，batch size不能太小，若法避免使用small batch size可考慮Group Normalization。

(Yuxin Wu & Kaiming He, 22 Mar 2018)

Batch Normalization

- Backpropagation:

$$\frac{\partial \ell}{\partial \hat{x}_i} = \frac{\partial \ell}{\partial y_i} \cdot \gamma$$

$$\frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial \hat{x}_i} \cdot (x_i - \mu_{\mathcal{B}}) \cdot \frac{-1}{2}(\sigma_{\mathcal{B}}^2 + \epsilon)^{-3/2}$$

$$\frac{\partial \ell}{\partial \mu_{\mathcal{B}}} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial \hat{x}_i} \cdot \frac{-1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}$$

$$\frac{\partial \ell}{\partial x_i} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial \hat{x}_i} \cdot \frac{-1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} + \frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{2(x_i - \mu_{\mathcal{B}})}{m} + \frac{\partial \ell}{\partial \mu_{\mathcal{B}}} \cdot \frac{1}{m}$$

$$\frac{\partial \ell}{\partial \gamma} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial y_i} \cdot \hat{x}_i$$

$$\frac{\partial \ell}{\partial \beta} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial y_i}$$

Training and Testing

- Input: Network N with trainable parameters Θ ;
  Subset of activations $\left\{x^{(k)}\right\}_{k=1}^{K}$
- Output: Batch-normalized network for inference,
  $N_{BN}^{\inf}$.
  - Set $N_{BN}^{tr} = N$
  - For $k = 1 \ldots K$ do
    Add transformation $y^{(k)} = BN_{\gamma^{(k)},\beta^{(k)}}\left(x^{(k)}\right)$ to N_{BN}^{tr}
    Modify each layer in N_{BN}^{tr} with input $x^{(k)}$ to take $y^{(k)}$ instead
    end for

Training and Testing

– Train N_{BN}^{tr} to optimize the parameters

$$\Theta \cup \left\{\gamma^{(k)}, \beta^{(k)}\right\}_{k=1}^{K}$$

– $N_{BN}^{\inf} = N_{BN}^{tr}$ // Inference BN network with forzen
// parameters

Training and Testing

– For $k = 1 \ldots K$ do // $x = x^{(k)}, \gamma = \gamma^{(k)}, \mu_{\mathrm{B}} = \mu_{\mathrm{B}}^{(k)}$, etc

Process multiple training mini-batches ,

each of size , and average over them:

$$E[x] = E_{\mathrm{B}}[\mu_{\mathrm{B}}]$$

$$Var[x] = \frac{m}{m-1} E_{\mathrm{B}}[\sigma_{\mathrm{B}}^2]$$

In $N_{BN}^{\inf}$, replace the transform $y = BN_{\gamma,\beta}(x)$

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

end for

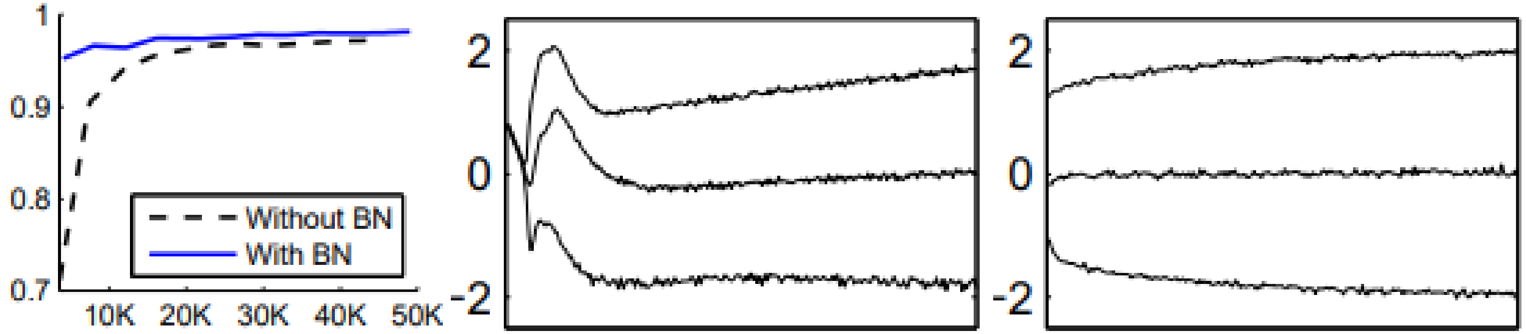
Training and Testing

- Remark.
  - Batch Normalization can be applied to any set of activations in the network.
  - In this paper, replace $z = g(Wu + b)$
    by $z = g(BN(Wu))$,
    the bias b can be ignored.

Experiment 1 (MNIST)

- Dataset: MNIST
  - inputs: 28x28 binary image
  - 3 FC hidden layers with 100 activations each
  - $y = g(Wu + b)$ with sigmoid nonlinearity
  - W initialized to a small random Gaussian values
  - outputs: FC layer with 10 activations
  - cross-entropy loss
  - 50000 steps
  - batch size: 60

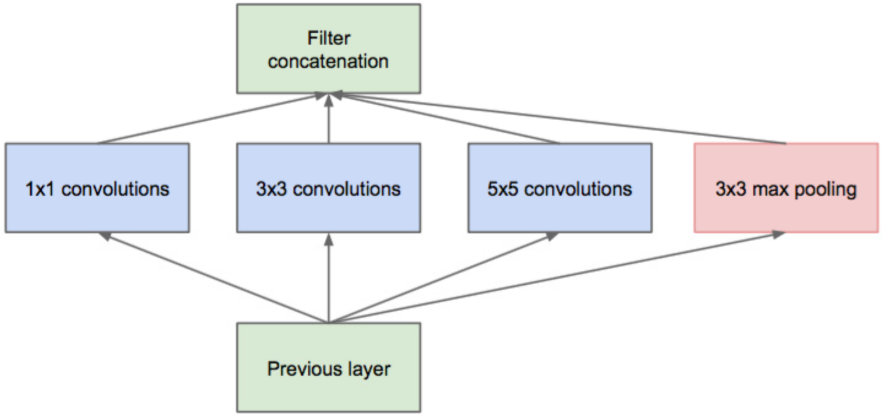
Experiment 1 (MNIST)



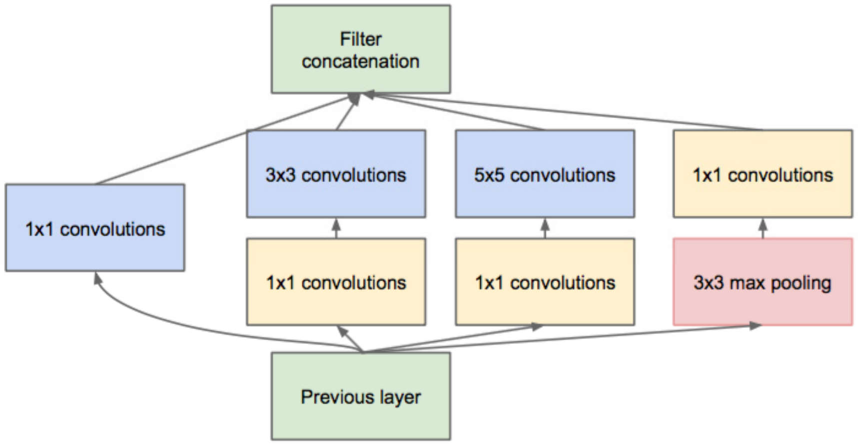
(a) (b) Without BN (c) With BN

Figure 1: (a) *The test accuracy of the MNIST network trained with and without Batch Normalization, vs. the number of training steps. Batch Normalization helps the network train faster and achieve higher accuracy.* (b, c) *The evolution of input distributions to a typical sigmoid, over the course of training, shown as* $\{15, 50, 85\}$*th percentiles. Batch Normalization makes the distribution more stable and reduces the internal covariate shift.*

Inception



(a) Inception module, naïve version



(b) Inception module with dimensionality reduction

Figure 2: Inception module

Inception

type	patch size/ stride	output size	depth	$\#1{\times}1$	$\#3{\times}3$ reduce	$\#3{\times}3$	$\#5{\times}5$ reduce	$\#5{\times}5$	pool proj	params	ops
convolution	$7{\times}7/2$	$112{\times}112{\times}64$	1							2.7K	34M
max pool	$3{\times}3/2$	$56{\times}56{\times}64$	0								
convolution	$3{\times}3/1$	$56{\times}56{\times}192$	2		64	192				112K	360M
max pool	$3{\times}3/2$	$28{\times}28{\times}192$	0								
inception (3a)		$28{\times}28{\times}256$	2	64	96	128	16	32	32	159K	128M
inception (3b)		$28{\times}28{\times}480$	2	128	128	192	32	96	64	380K	304M
max pool	$3{\times}3/2$	$14{\times}14{\times}480$	0								
inception (4a)		$14{\times}14{\times}512$	2	192	96	208	16	48	64	364K	73M
inception (4b)		$14{\times}14{\times}512$	2	160	112	224	24	64	64	437K	88M
inception (4c)		$14{\times}14{\times}512$	2	128	128	256	24	64	64	463K	100M
inception (4d)		$14{\times}14{\times}528$	2	112	144	288	32	64	64	580K	119M
inception (4e)		$14{\times}14{\times}832$	2	256	160	320	32	128	128	840K	170M
max pool	$3{\times}3/2$	$7{\times}7{\times}832$	0								
inception (5a)		$7{\times}7{\times}832$	2	256	160	320	32	128	128	1072K	54M
inception (5b)		$7{\times}7{\times}1024$	2	384	192	384	48	128	128	1388K	71M
avg pool	$7{\times}7/1$	$1{\times}1{\times}1024$	0								
dropout (40%)		$1{\times}1{\times}1024$	0								
linear		$1{\times}1{\times}1000$	1							1000K	1M
softmax		$1{\times}1{\times}1000$	0								

Table 1: GoogLeNet incarnation of the Inception architecture

Inception

- New Inception network:
  - Convolutional layers use ReLU as the nonlinearity
  - The 5×5 convolutional layers are replaced by two consecutive layers of 3×3 convolutions with up to 128 filters
  - Stochastic Gradient Descent with momentum
  - Batch size: 32

BN-Inception

- Apply BN to inputs of each nonlinearity and
  - Increase learning rate
  - Remove Dropout
  - Reduce the L2 weight regularization
  - Accelerate the learning rate decay
  - Remove Local Response Normalization
  - Shuffle training examples more thoroughly

Experiment 2

- Dataset: LSVRC2012 (1000 classes)
- Models:
  - Inception trained with initial LR 0.0015
  - BN-Baseline: Inception with BN
  - BN-x5: BN-Baseline with initial LR 0.0075
  - BN-x30: BN-Baseline with initial LR 0.045
  - BN-x5-sigmoid

Experiment 2

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.*

Experiment 2

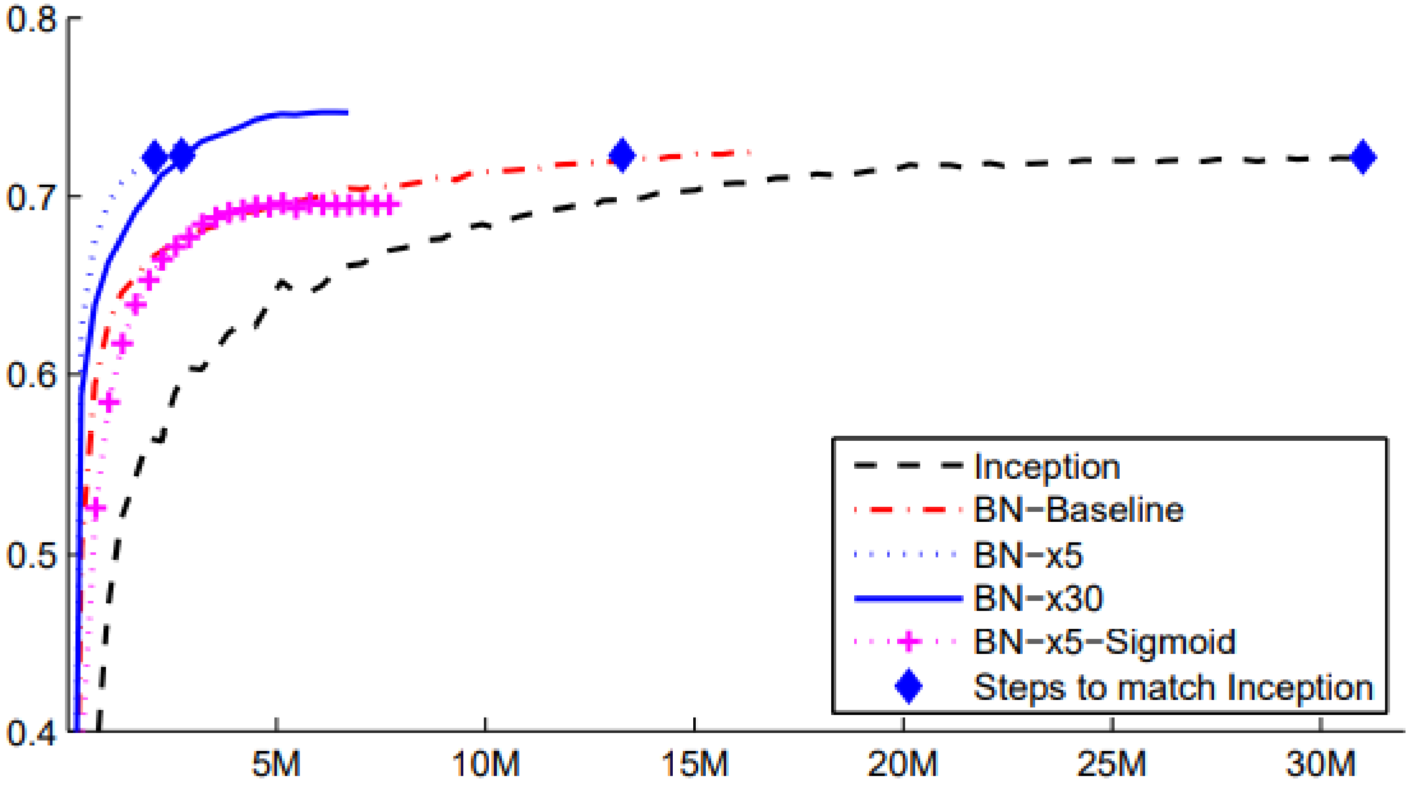


Figure 2: *Single crop validation accuracy of Inception and its batch-normalized variants, vs. the number of training steps.*

Experiment 3

- 6 networks, each was based on BN-x30
  - increased initial weights in the conv. Layers
  - using Dropout (5% or 10%)
  - using non-conv., per-activation BN with last hidden layers of the model
- Each network achieved its maximum accuracy after about 6×10^6 training steps

Experiment 3

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.*

Remark

- In face, BN does not really reduce internal covariate shift. There is a new paper: “How Does Batch Normalization Help Optimization? (No, It Is Not About Internal Covariate Shift)” Submitted on 29 May 2018 by Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas and Aleksander Madry.
- https://arxiv.org/abs/1805.11604

Thanks for your attention!