Supplementary Materials for Iterative Normalization: Beyond Standardization towards Efficient Whitening

Lei Huang Yi Zhou Fan Zhu Li Liu Ling Shao
Inception Institute of Artificial Intelligence (IIAI), Abu Dhabi, UAE
{lei.huang, yi.zhou, fan.zhu, li.liu, ling.shao} @inceptioniai.org

1. Derivation of Back-propagation

We first show the forward pass for illustration. We follow the common matrix notation that the vectors are column vectors by default while their derivations are row vectors. Given the mini-batch inputs $\mathbf{X} \in \mathbb{R}^{d\times m}$, the forward pass of our Iterative Normalization (IterNorm) to compute the whitened output is described below:

$$\mu = \frac{1}{m}\mathbf{X}\cdot\mathbf{1} \tag{1}$$
$$\mathbf{X}_C = \mathbf{X} - \mu\cdot\mathbf{1}^T \tag{2}$$
$$\Sigma = \frac{1}{m}\mathbf{X}_C\mathbf{X}_C^T + \epsilon\mathbf{I} \tag{3}$$
$$\Sigma_N = \Sigma/tr(\Sigma) \tag{4}$$
$$\Sigma^{-\frac{1}{2}} = \mathbf{P}_T/\sqrt{tr(\Sigma)} \tag{5}$$
$$\widehat{\mathbf{X}} = \Sigma^{-\frac{1}{2}}\mathbf{X}_C \tag{6}$$

where $\mathbf{P}_T$ is calculated based on Newton's iterations as follows:

$$\begin{cases}\mathbf{P}_0 = \mathbf{I}\\ \mathbf{P}_k = \frac{1}{2}(3\mathbf{P}_{k-1} - \mathbf{P}_{k-1}^3\Sigma_N), \quad k = 1, 2, ..., T.\end{cases} \tag{7}$$

The back-propagation pass is based on the chain rule. Given $\frac{\partial L}{\partial\widehat{\mathbf{X}}}$, we can calculate $\frac{\partial L}{\partial\mathbf{X}}$ based on Eqn. 1 and 2:

$$\frac{\partial L}{\partial\mathbf{X}} = \frac{1}{m}\cdot(\mathbf{1}\cdot\frac{\partial L}{\partial\mu})^T + \frac{\partial L}{\partial\mathbf{X}_C}, \tag{8}$$

where $\frac{\partial L}{\partial\mu}$ is calculated based on Eqn. 2:

$$\frac{\partial L}{\partial\mu} = -\mathbf{1}^T\cdot\frac{\partial L}{\partial\mathbf{X}_C}^T \tag{9}$$

and $\frac{\partial L}{\partial\mathbf{X}_C}$ is calculated based on Eqn. 3 and 6:

$$\frac{\partial L}{\partial\mathbf{X}_C} = \Sigma^{-\frac{1}{2}}\frac{\partial L}{\partial\widehat{\mathbf{X}}} + \frac{2}{m}(\frac{\partial L}{\partial\Sigma})_s\mathbf{X}_C, \tag{10}$$

where $(\frac{\partial L}{\partial\Sigma})_s$ means symmetrizing $\frac{\partial L}{\partial\Sigma}$ by $\frac{\partial L}{\partial\Sigma} = \frac{1}{2}(\frac{\partial L}{\partial\Sigma} + \frac{\partial L}{\partial\Sigma}^T)$. Next, we will calculate $\frac{\partial L}{\partial\Sigma}$, given $\frac{\partial L}{\partial\Sigma^{-\frac{1}{2}}}$ as follows:

$$\frac{\partial L}{\partial\Sigma^{-\frac{1}{2}}} = \frac{\partial L}{\partial\widehat{\mathbf{X}}}\mathbf{X}_C^T \tag{11}$$

Based on Eqn. 4 and 5, we obtain:

$$\begin{aligned}\frac{\partial L}{\partial\Sigma} &= \frac{\partial L}{\partial\Sigma_N}\frac{\Sigma_N}{\partial\Sigma} + \frac{\partial L}{\partial\Sigma^{-\frac{1}{2}}}\frac{\partial\Sigma^{-\frac{1}{2}}}{\partial\Sigma}\\ &= \frac{1}{tr(\Sigma)}\frac{\partial L}{\partial\Sigma_N} + tr(\frac{\partial L}{\partial\Sigma_N}^T\Sigma)\frac{\partial\frac{1}{tr(\Sigma)}}{\partial tr(\Sigma)}\frac{\partial tr(\Sigma)}{\partial\Sigma}\\ &\quad + tr(\frac{\partial L}{\partial\Sigma^{-\frac{1}{2}}}^T\mathbf{P}_T)\frac{\partial\frac{1}{\sqrt{tr(\Sigma)}}}{\partial tr(\Sigma)}\frac{\partial tr(\Sigma)}{\partial\Sigma}\\ &= \frac{1}{tr(\Sigma)}\frac{\partial L}{\partial\Sigma_N} - \frac{1}{(tr(\Sigma))^2}tr(\frac{\partial L}{\partial\Sigma_N}^T\Sigma)\mathbf{I}\\ &\quad - \frac{1}{2(tr(\Sigma))^{3/2}}tr((\frac{\partial L}{\partial\Sigma^{-1/2}})^T\mathbf{P}_T)\mathbf{I}.\end{aligned} \tag{12}$$

Based on Eqn. 7, we can derive $\frac{\partial L}{\partial\Sigma_N}$ as follows:

$$\frac{\partial L}{\partial\Sigma_N} = -\frac{1}{2}\sum_{k=1}^{T}(\mathbf{P}_{k-1}^3)^T\frac{\partial L}{\partial\mathbf{P}_k} \tag{13}$$

where $\{\frac{\partial L}{\partial\mathbf{P}_k}, k = 1, ..., T\}$ can be calculated based on Eqn. 5 and 7 by following iterations:

$$\begin{cases}\frac{\partial L}{\partial\mathbf{P}_T} = \frac{1}{\sqrt{tr(\Sigma)}}\frac{\partial L}{\partial\Sigma^{-\frac{1}{2}}}\\ \frac{\partial L}{\partial\mathbf{P}_{k-1}} = \frac{3}{2}\frac{\partial L}{\partial\mathbf{P}_k} - \frac{1}{2}\frac{\partial L}{\partial\mathbf{P}_k}(\mathbf{P}_{k-1}^2\Sigma_N)^T - \frac{1}{2}(\mathbf{P}_{k-1}^2)^T\frac{\partial L}{\partial\mathbf{P}_k}\Sigma_N^T\\ -\frac{1}{2}(\mathbf{P}_{k-1})^T\frac{\partial L}{\partial\mathbf{P}_k}(\mathbf{P}_{k-1}\Sigma_N)^T, \quad k = T, ..., 1.\end{cases} \tag{14}$$

Further, we can simplify the derivation of $\frac{\partial L}{\partial \mathbf{X}}$ as:

$$\begin{aligned}
\frac{\partial L}{\partial \mathbf{X}} &= \frac{1}{m}\cdot(\mathbf{1}\cdot\frac{\partial L}{\partial \mu})^T + \frac{\partial L}{\partial \mathbf{X}_C} \\
&= \frac{\partial L}{\partial \mathbf{X}_C}(\mathbf{I} - \frac{1}{m}\mathbf{1}\mathbf{1}^T) \\
&= (\Sigma^{-\frac{1}{2}}\frac{\partial L}{\partial \widehat{\mathbf{X}}} + \frac{2}{m}(\frac{\partial L}{\partial \Sigma})_s\mathbf{X}_C)(\mathbf{I} - \frac{1}{m}\mathbf{1}\mathbf{1}^T) \\
&= (\Sigma^{-\frac{1}{2}}\frac{\partial L}{\partial \widehat{\mathbf{X}}})(\mathbf{I} - \frac{1}{m}\mathbf{1}\mathbf{1}^T) + \frac{2}{m}(\frac{\partial L}{\partial \Sigma})_s\mathbf{X}_C \\
&\quad - \frac{2}{m}\frac{1}{m}(\frac{\partial L}{\partial \Sigma})_s(\mathbf{X}_C\mathbf{1})\mathbf{1}^T \\
&= \Sigma^{-\frac{1}{2}}\frac{\partial L}{\partial \widehat{\mathbf{X}}}(\mathbf{I} - \frac{1}{m}\mathbf{1}\mathbf{1}^T) + \frac{2}{m}(\frac{\partial L}{\partial \Sigma})_s\mathbf{X}_C + 0 \\
&= \Sigma^{-\frac{1}{2}}(\frac{\partial L}{\partial \widehat{\mathbf{X}}} - \mathbf{f}\mathbf{1}^T) + \frac{2}{m}(\frac{\partial L}{\partial \Sigma})_s\mathbf{X}_C \\
&= \Sigma^{-\frac{1}{2}}(\frac{\partial L}{\partial \widehat{\mathbf{X}}} - \mathbf{f}\mathbf{1}^T) + \frac{1}{m}(\frac{\partial L}{\partial \Sigma} + \frac{\partial L}{\partial \Sigma}^T)\mathbf{X}_C \qquad (15)
\end{aligned}$$

where $\mathbf{f} = \frac{1}{m}\frac{\partial L}{\partial \widehat{\mathbf{X}}}\cdot\mathbf{1}$.

2. Comparison of Wall Clock Time

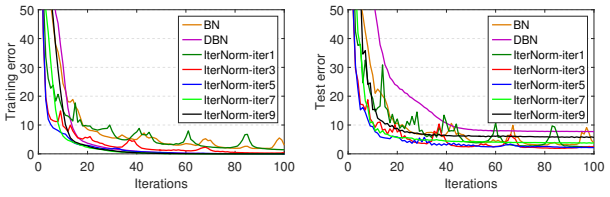
As discussed in Section 3.3 of the main paper, the computational cost of our method is comparable to the convolution operation. To be specific, given the internal activation $\mathbf{X}_C \in \mathbb{R}^{h\times w\times d\times m}$, the 3×3 convolution with the same input and output feature maps costs $9hwmd^2$, while our IterNorm costs $2hwmd^2 + Td^3$. The relative cost of IterNorm for 3×3 convolution is $2/9 + Td/mhw$.

Here, we compare the wall-clock time of IterNorm, Decorrelated Batch Normalization (DBN) [3] and 3×3 convolution. Our IterNorm is implemented based on Torch [2]. The implementation of DBN is from the released code of the DBN paper [3]. We compare 'IterNorm' to the 'nn', 'cudnn' convolution [1] and DBN [3] in Torch. The experiments are run on a TITAN Xp. We use the corresponding configurations of the input $\mathbf{X}_C \in \mathbb{R}^{h\times w\times d\times m}$ and convolution $\mathbf{W}_C \in \mathbb{R}^{3\times 3\times d\times d}$: $m = 64$, $h = w = 32$. We compare the results of $d = 64$ and $d = 128$, as shown in Table 1. We find that our unoptimized implementation of IterNorm (*e.g.*, 'IterNorm-iter5') is faster than the 'nn' convolution, and slightly slower than 'cudnn' convolution. Note that our IterNorm is implemented based on the API provided by Torch [2], it is thus more fair to compare IterNorm to 'nn' convolution. Besides, our IterNorm is significantly faster than DBN.

Besides, we also conduct additional experiments to compare the training time of IterNorm and DBN on the VGG architecture described in Section 5.1 of the main paper, with a batch size of 256. DBN (group size of 16) costs 1.366s per iteration, while IterNorm costs 0.343s.

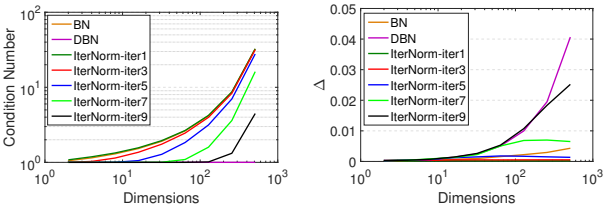
Module	d=64	d=128
'nn' convolution	8.89ms	17.46ms
'cudnn' convolution	5.65ms	13.62ms
DBN [3]	15.92ms	35.02ms
IterNorm-iter3	6.59ms	13.32ms
IterNorm-iter5	7.40ms	13.92ms
IterNorm-iter7	8.21ms	14.68ms

Table 1. Comparison of wall clock time (ms). 'IterNorm-iterN' (N=3, 5 and 7) indicates the proposed IterNorm with iteration number of N. We compute the time including the forward pass and the backward pass, averaged over 100 runs.



(a) (b)

Figure 1. Ablation study in training an MLP on MNIST. The experimental setup is the same as the setup in Section 4.2 of the main paper: We train a 4-layer MLP and the number of neurons in each hidden layer is 100; We use full batch gradient and report the best results with respect to the training loss among learning rates=$\{0.2, 0.5, 1, 2, 5\}$. (a) shows the training error with respect to the iterations, and (b) shows the test error with respect to the iterations.



(a) (b)

Figure 2. Comparison of different normalization operations in condition number of covariance matrix of normalized output (a) and SND (b). We sample 60,000 examples from Gaussian distribution and choose a batch size of 1024, and observe the results with respect to the dimensions from 2^1 to 2^9, averaged over 10 times.

3. Experiments of IterNorm with Different Iterations

Here, we show the results of IterNorm with different iteration numbers on the experiments described in Section 4.2 of the main paper. We also show the results of Batch Normalization (BN) [4] and Decorrelated Batch Normalization (DBN) [3] for comparison.

Figure 1 shows the results on MNIST dataset. We explore the effects of T on performance of IterNorm, for a range of $\{1, 3, 5, 7, 9\}$. We observe that the smallest (T = 1) and

input (32×32 RGB image)
conv3(3,64) conv3(64,64) maxpool(2,2)
conv3(64,128) conv3(128,128) maxpool(2,2)
conv3(128,256) conv3(256,256) $\times 3$ maxpool(2,2)
conv3(256,512) conv3(512,512) $\times 3$ maxpool(2,2)
conv3(512,512) $\times 4$ avepool(2,2)
FC(512,10)
soft-max

Table 2. The VGG network used in the experiment as shown in Section 5.1 in the main paper. 'conv3(d_{in}, d_{out})' indicates the 3×3 convolution with input channel number of d_{in} and output channel number of d_{out}.

the largest (T = 9) iteration number both have the worse performance in terms of training efficiency. Further, when T = 9, IterNorm has significantly worse test performance. These observations are consistent to the results on VGG network described in Section 5.1 of the main paper.

Figure 2 shows the results of SND and conditioning analysis. We observe that IterNorm has better conditioning and increasing SND, with increasing iteration T. The results show that the iteration number T can be effectively used to control the extent of whitening, therefore to obtain a good trade-off between the improved conditioning and introduced stochasticity.

4. Details of the VGG Network

As introduced in Section 5.1 of the main paper, we use the VGG networks [5] tailored for 32×32 inputs. Table 2 shows the details of the used VGG network.

References

[1] Sharan Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan Cohen, John Tran, Bryan Catanzaro, and Evan Shelhamer. cudnn: Efficient primitives for deep learning. *CoRR*, abs/1410.0759, 2014. 2

[2] R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, 2011. 2

[3] Lei Huang, Dawei Yang, Bo Lang, and Jia Deng. Decorrelated batch normalization. In *CVPR*, 2018. 2

[4] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 2

[5] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 3