A. Algorithm Outline

Algorithm 1 presents an outline of our proposed Cross-Iteration Batch Normalization (CBN).

Algorithm 1: Cross-Iteration Batch Normalization(CBN)

```
Input: Feature responses of a network node of the
       $l$-th layer at the $t$-th iteration $\{x^l_{t,i}(\theta_t)\}^m_{i=1}$,
       network weights $\{\theta^l_{t-\tau}\}^{k-1}_{\tau=0}$, statistics
       $\{\mu^l_{t-\tau}(\theta_{t-\tau})\}^{k-1}_{\tau=1}$ and $\{\nu^l_{t-\tau}(\theta_{t-\tau})\}^{k-1}_{\tau=1}$,
       and gradients $\{\partial\mu_{t-\tau}(\theta_{t-\tau})/\partial\theta^l_{t-\tau}\}^{k-1}_{\tau=1}$
       and $\{\partial\nu_{t-\tau}(\theta_{t-\tau})/\partial\theta^l_{t-\tau}\}^{k-1}_{\tau=1}$ from most
       recent $k-1$ iterations
Output: $\{y^l_{t,i}(\theta_t) = \text{CBN}(x^l_{t,i}(\theta_t))\}$
$\mu_t(\theta_t) \leftarrow \frac{1}{m}\sum^m_{i=1} x_{t,i}(\theta_t)$,
  $\nu_t(\theta_t) \leftarrow \frac{1}{m}\sum^m_{i=1} x^2_{t,i}(\theta_t)$        //statistics on the
  current iteration
for $\tau \in \{1,\dots,k\}$ do
    $\mu^l_{t-\tau}(\theta_t) \leftarrow$
      $\mu^l_{t-\tau}(\theta_{t-\tau}) + \frac{\partial\mu^l_{t-\tau}(\theta_{t-\tau})}{\partial\theta^l_{t-\tau}}(\theta^l_t - \theta^l_{t-\tau})$
      //approximation from recent iterations
    $\nu^l_{t-\tau}(\theta_t) \leftarrow \nu^l_{t-\tau}(\theta_{t-\tau}) + \frac{\partial\nu^l_{t-\tau}(\theta_{t-\tau})}{\partial\theta^l_{t-\tau}}(\theta^l_t - \theta^l_{t-\tau})$
      //approximation from recent iterations
end
$\bar{\mu}^l_{t,k}(\theta_t) \leftarrow \frac{1}{k}\sum^{k-1}_{\tau=0}\mu^l_{t-\tau}(\theta_t)$        //averaging over
  recent iterations
$\bar{\nu}^l_{t,k}(\theta_t) \leftarrow \frac{1}{k}\sum^{k-1}_{\tau=0}\max\left[\nu^l_{t-\tau}(\theta_t), \mu^l_{t-\tau}(\theta_t)^2\right]$
  //validation and averaging over recent iterations
$\bar{\sigma}^l_{t,k}(\theta_t)^2 \leftarrow \bar{\nu}^l_{t,k}(\theta_t) - \bar{\mu}^l_{t,k}(\theta_t)^2$
$\hat{x}^l_{t,i}(\theta_t) = \frac{x^l_{t,i}(\theta_t) - \bar{\mu}^l_{t,k}(\theta_t)}{\sqrt{\bar{\sigma}^l_{t,k}(\theta_t)^2 + \epsilon}}$        //normalize
$y^l_{t,i}(\theta_t) \leftarrow \gamma\hat{x}^l_{t,i}(\theta_t) + \beta$        //scale and shift
```

B. Efficient Implementation of $\partial\mu^l_{t-\tau}(\theta_{t-\tau})/\partial\theta^l_{t-\tau}$ and $\partial\nu^l_{t-\tau}(\theta_{t-\tau})/\partial\theta^l_{t-\tau}$

Let C_{out} and C_{in} denote the output and input channel dimension of the l-th layer, respectively, and K denotes the kernel size of $\theta^l_{t-\tau}$. $\mu^l_{t-\tau}$ and $\nu^l_{t-\tau}$ are thus of C_{out} dimensions in channels, and $\theta^l_{t-\tau}$ is a $C_{out} \times C_{in} \times K$ dimensional tensor. A naive implementation of $\partial\mu^l_{t-\tau}(\theta_{t-\tau})/\partial\theta^l_{t-\tau}$ and $\partial\nu^l_{t-\tau}(\theta_{t-\tau})/\partial\theta^l_{t-\tau}$ involves computational overhead of $O(C_{out} \times C_{out} \times C_{in} \times K)$. Here we find that the operations of μ and ν can be implemented efficiently in $O(C_{in} \times K)$ and $O(C_{out} \times C_{in} \times K)$, respectively, thanks to the averaging of feature responses in μ and ν.

Here we derive the efficient implementation of $\partial\mu^l_{t-\tau}(\theta_{t-\tau})/\partial\theta^l_{t-\tau}$. That of $\partial\nu^l_{t-\tau}(\theta_{t-\tau})/\partial\theta^l_{t-\tau}$ is about the same. Let us first simplify the notations a bit. Let μ^l and θ^l denote $\mu^l_{t-\tau}(\theta_{t-\tau})$ and $\theta^l_{t-\tau}$ respectively, by removing the irrelevant notations for iterations. The element-wise computation in the forward pass can be computed as

$$\mu^l_j = \frac{1}{m}\sum^m_{i=1} x^l_{i,j}, \tag{13}$$

where μ^l_j denotes the j-th channel in μ^l, and $x^l_{i,j}$ denotes the j-th channel in the i-th example. $x^l_{i,j}$ is computed as

$$x^l_{i,j} = \sum^{C_{in}}_{n=1}\sum^{K}_{k=1}\theta^l_{j,n,k} \cdot y^{l-1}_{i+\text{offset}(k),n}, \tag{14}$$

where n and k enumerate the input feature dimension and the convolution kernel index, respectively, offset(k) denotes the spatial offset in applying the k-th kernel, and y^{l-1} is the output of the $(l-1)$-th layer.

The element-wise calculation of $\partial\mu^l/\partial\theta^l \in \mathbb{R}^{C_{out}\times C_{out}\times C_{in}\times K}$ is as follows, taking Eq. (13) and Eq. (14) into consideration:

$$\begin{aligned}
[\frac{\partial\mu^l}{\partial\theta^l}]_{j,q,p,\eta} &= \frac{\partial\mu^l_j}{\partial\theta^l_{q,p,\eta}} \\
&= \frac{\partial\frac{1}{m}\sum^m_{i=1}x^l_{i,j}}{\partial\theta^l_{q,p,\eta}} \\
&= \frac{\partial\frac{1}{m}\sum^m_{i=1}\sum^{C_{in}}_{n=1}\sum^K_{k=1}\theta^l_{j,n,k}\cdot y^{l-1}_{i+\text{offset}(k),n}}{\partial\theta^l_{q,p,\eta}} \\
&= \begin{cases} \frac{1}{m}\sum^m_{i=1} y^{l-1}_{i+\text{offset}(\eta),p} & , j = q \\ 0 & , j \neq q \end{cases} .
\end{aligned} \tag{15}$$

Thus, $[\frac{\partial\mu^l}{\partial\theta^l}]_{j,q,p,\eta}$ takes non-zero values only when $j = q$. This operation can be implemented efficiently in $O(C_{in} \times K)$. Similarly, the calculation of $\partial\nu^l/\partial\theta^l$ can be obtained in $O(C_{out} \times C_{in} \times K)$.

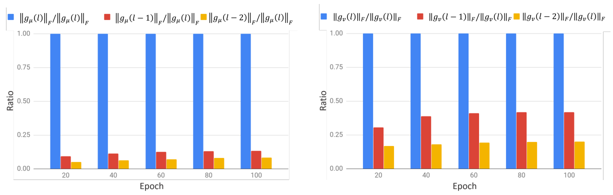
C. Observation of the gradients diminishing

The key assumption in Eq. (7) and Eq. (8) is that for a node at the l-th layer, the gradient of its statistics with respect to the network weights at the l-th layer is larger than that of weights from the prior layers, i.e.,

$$\begin{aligned}
&||g_\mu(l|l,t,\tau)||_F \gg ||g_\mu(r|l,t,\tau)||_F \\
&||g_\nu(l|l,t,\tau)||_F \gg ||g_\nu(r|l,t,\tau)||_F, \qquad r < l
\end{aligned}$$

where $g_\mu(r|l,t,\tau)$ denotes $\frac{\partial\mu^l_{t-\tau}(\theta_{t-\tau})}{\partial\theta^r_{t-\tau}}$, $g_\nu(r|l,t,\tau)$ denotes $\frac{\partial\nu^l_{t-\tau}(\theta_{t-\tau})}{\partial\theta^r_{t-\tau}}$, and $||\cdot||_F$ denotes the Frobenius norm.

Here, we examine this assumption empirically for networks trained on ImageNet image recognition. Both



(a) The gradients of μ (b) The gradients of ν

Figure 6. Comparison of gradients of statistics w.r.t. current layer vs. that w.r.t. previous layers on ImageNet.

$||g_\mu(r)||_F/||g_\mu(l)||_F$ and $||g_\nu(r)||_F/||g_\nu(l)||_F$ for $r \in \{l-1, l-2\}$ are averaged over all CBN layers of the network at different training epochs (Figure 6). The results suggest that the key assumption holds well, thus validating the approximation in Eq. (7) and Eq. (8).

We also study the gradients of non-ResNet models. The ratios of $||g_\mu||_F$ and $||g_\nu||_F$ are (0.20 and 0.41) for VGG-16 and (0.15 and 0.37) for Inception-V3, which is similar to ResNet (0.12 and 0.39), indicating that the assumption should also hold for the VGG and Inception series.