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

Sergey Ioffe, Christian Szegedy
Google

Reviewed by Zhao Song

December 17, 2015
Stochastic gradient descent (\textit{SGD})

- Minimize the expected loss over the training set:
  \[
  \hat{\theta} = \arg \min_{\theta} \mathbb{E}_{x \sim D}[\ell(x, \theta)]
  \]
- Parameters update according to the gradient of mini-batches
  \[
  \theta \leftarrow \theta - \frac{\alpha}{m} \sum_{i=1}^{m} \frac{\partial \ell(x_i, \theta)}{\partial \theta}
  \]
- Careful tuning of learning rates and initial parameters.
Covariate shift: Changes of input distribution to a learning system

\[ \ell = F(x, \theta) \]

Internal covariate shift: Extension to the deep network

\[
\ell = F_2(F_1(u, \theta_1), \theta_2) \\
= F_2(x, \theta_2)
\]
Reducing internal covariate shift

- Whitening the inputs to each layer:

\[ x \leftarrow x - \frac{E[x]}{\sqrt{\text{Var}[x]}} \]

However, gradient descent does not take into account the normalization [Ioffe and Szegedy 2015].

- Mean and variance of an activation depend on model parameters

\[ \frac{\partial E[x]}{\partial \theta} \text{ and } \frac{\partial \text{Var}[x]}{\partial \theta} \]
Batch normalizing transform

**Input:** Values of \( x \) over a mini-batch: \( \mathcal{B} = \{x_1...m\} \);
Parameters to be learned: \( \gamma, \beta \)

**Output:** \( \{y_i = \text{BN}_{\gamma,\beta}(x_i)\} \)

\[
\begin{align*}
\mu_\mathcal{B} &\leftarrow \frac{1}{m} \sum_{i=1}^{m} x_i \quad \text{// mini-batch mean} \\
\sigma^2_\mathcal{B} &\leftarrow \frac{1}{m} \sum_{i=1}^{m} (x_i - \mu_\mathcal{B})^2 \quad \text{// mini-batch variance} \\
\hat{x}_i &\leftarrow \frac{x_i - \mu_\mathcal{B}}{\sqrt{\sigma^2_\mathcal{B} + \epsilon}} \quad \text{// normalize} \\
y_i &\leftarrow \gamma\hat{x}_i + \beta \equiv \text{BN}_{\gamma,\beta}(x_i) \quad \text{// scale and shift}
\end{align*}
\]

**Algorithm 1:** Batch Normalizing Transform, applied to activation \( x \) over a mini-batch.
Backpropagation with batch normalization

(a)

(b)

\[\frac{\partial l}{\partial x_i} = \frac{\partial l}{\partial y_i} \cdot \gamma\]

\[\frac{\partial l}{\partial \sigma_B^2} = \sum_{i=1}^{m} \frac{\partial l}{\partial x_i} \cdot (x_i - \mu_B) \cdot \frac{-1}{2} (\sigma_B^2 + \epsilon)^{-3/2}\]

\[\frac{\partial l}{\partial \mu_B} = \sum_{i=1}^{m} \frac{\partial l}{\partial x_i} \cdot \frac{-1}{\sqrt{\sigma_B^2 + \epsilon}}\]

\[\frac{\partial l}{\partial x_i} = \frac{\partial l}{\partial x_i} \cdot \frac{1}{\sqrt{\sigma_B^2 + \epsilon}} + \frac{\partial l}{\partial \sigma_B^2} \cdot \frac{2(x_i - \mu_B)}{m} + \frac{\partial l}{\partial \mu_B} \cdot \frac{1}{m}\]

\[\frac{\partial l}{\partial y_i} = \sum_{i=1}^{m} \frac{\partial l}{\partial y_i} \cdot \hat{x}_i\]

\[\frac{\partial l}{\partial \beta} = \sum_{i=1}^{m} \frac{\partial l}{\partial y_i}\]
Training a batch-normalized network

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

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

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: $N_{\text{BN}}^{\inf} \leftarrow N_{\text{BN}}^{\text{tr}}$ // Inference BN network with frozen parameters
8: for $k = 1 \ldots K$ do
9: // For clarity, $x \equiv x^{(k)}$, $\gamma \equiv \gamma^{(k)}$, $\mu_B \equiv \mu_B^{(k)}$, etc.
10: Process multiple training mini-batches $\mathcal{B}$, each of size $m$, and average over them:
    \begin{align*}
    E[x] &\leftarrow E_B[\mu_B] \\
    \text{Var}[x] &\leftarrow \frac{m}{m-1} E_B[\sigma_B^2]
    \end{align*}
11: In $N_{\text{BN}}^{\inf}$, replace the transform $y = \text{BN}_{\gamma, \beta}(x)$ with
    \begin{align*}
    y &\leftarrow \frac{\gamma}{\sqrt{\text{Var}[x]+\epsilon}} \cdot x + \left( \beta - \frac{\gamma E[x]}{\sqrt{\text{Var}[x]+\epsilon}} \right)
    \end{align*}
12: end for

**Algorithm 2:** Training a Batch-Normalized Network
Experiments

- **MNIST**
  - 3 fully-connect hidden layers with 100 nodes in each layer.
  - Sigmoid activation function.
  - Mini-batch size to be 60.

- **ImageNet**
  - The Inception network [Szegedy et al. 2014].
  - SGD with momentum [Sutskever et al. 2013].
  - Mini-batch size to be 32.
Learning curve on MNIST [Ioffe 2015]
Learning curve on ImageNet with single networks
### Classification results on ImageNet

<table>
<thead>
<tr>
<th>Model</th>
<th>Resolution</th>
<th>Crops</th>
<th>Models</th>
<th>Top-1 error</th>
<th>Top-5 error</th>
</tr>
</thead>
<tbody>
<tr>
<td>GoogLeNet ensemble</td>
<td>224</td>
<td>144</td>
<td>7</td>
<td>-</td>
<td>6.67%</td>
</tr>
<tr>
<td>Deep Image low-res</td>
<td>256</td>
<td>-</td>
<td>1</td>
<td>-</td>
<td>7.96%</td>
</tr>
<tr>
<td>Deep Image high-res</td>
<td>512</td>
<td>-</td>
<td>1</td>
<td>24.88%</td>
<td>7.42%</td>
</tr>
<tr>
<td>Deep Image ensemble</td>
<td>up to 512</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>5.98%</td>
</tr>
<tr>
<td>MSRA multicrop</td>
<td>up to 480</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>5.71%</td>
</tr>
<tr>
<td>MSRA ensemble</td>
<td>up to 480</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>4.94%*</td>
</tr>
<tr>
<td>BN-Inception single crop</td>
<td>224</td>
<td>1</td>
<td>1</td>
<td>25.2%</td>
<td>7.82%</td>
</tr>
<tr>
<td>BN-Inception multicrop</td>
<td>224</td>
<td>144</td>
<td>1</td>
<td>21.99%</td>
<td>5.82%</td>
</tr>
<tr>
<td>BN-Inception ensemble</td>
<td>224</td>
<td>144</td>
<td>6</td>
<td>20.1%</td>
<td>4.82%*</td>
</tr>
</tbody>
</table>

