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

Sergey Ioffe
Christian Szegedy
Stochastic gradient optimization in deep models

- Minimize loss over the training data

\[
\Theta = \arg \min_{\Theta} \mathbb{E}_{x \sim \mathcal{D}} [\ell(x, \Theta)]
\]

- Follow gradient for mini-batches

\[
\Theta \leftarrow \Theta - \alpha \frac{1}{m} \sum_{i=1}^{m} \frac{\partial \ell(x_i, \Theta)}{\partial \Theta}
\]
Outline

- Internal covariate shift
  - Distributions of activations in deep models change during training
  - Eliminating these changes speeds up training
- Batch Normalization
  - Normalize values using mini-batch mean and variance
  - Backprop through the transform enables gradient optimization
- Speedup >10x in ImageNet training
- Beats state of the art in ImageNet classification
Care and training of deep models

\[ a \cdot x + b \]
Care and training of deep models
Care and training of deep models

\[ a x + b \]
Care and training of deep models

\[ ax + b \]
Mitigating the effect of changing input distributions

- Careful initialization
- Small learning rates
- Rectifiers

\[ a x + b \]
Covariate shift

- Change in input distribution requires domain adaptation

\[ \ell = F(x, \Theta) \]
Internal covariate shift

- Layer input distributions change during training
  \[ \ell = F_2(F_1(u, \Theta_1), \Theta_2) \]

- Change in internal activation distribution requires domain adaptation
Reducing internal covariate shift to speed up training

- Normalize each activation:

\[ x \rightarrow \frac{x - \text{E}[x]}{\sqrt{\text{Var}[x]}} \]
Normalization must participate in gradient optimization

- Mean and variance of an activation depend on model parameters
  
- Need \( \frac{\partial E[x]}{\partial \Theta} \) and \( \frac{\partial \text{Var}[x]}{\partial \Theta} \)

- Cannot use population means and variances in mini-batch gradient optimization
Batch Normalization

Mini-batch mean: \( \mu_B \leftarrow \frac{1}{m} \sum_{i=1}^{m} x_i \)

Mini-batch variance: \( \sigma_B^2 \leftarrow \frac{1}{m} \sum_{i=1}^{m} (x_i - \mu_B)^2 \)

Normalize: \( \hat{x}_i \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \)

Scale and shift: \( y_i \leftarrow \gamma \hat{x}_i + \beta \)
Backprop with Batch Normalization

\[
\frac{\partial l}{\partial \mu_B} \leftarrow \frac{1}{m} \sum_{i=1}^{m} x_i
\]

\[
\frac{\partial l}{\partial \sigma^2_B} \leftarrow \frac{1}{m} \sum_{i=1}^{m} (x_i - \mu_B)^2
\]

\[
\hat{x}_i \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma^2_B + \epsilon}}
\]

\[
y_i \leftarrow \gamma \hat{x}_i + \beta
\]
Inference with Batch Normalization

- Replace batch statistics with population statistics

\[ \hat{x}_i \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma^2_B + \epsilon}} \quad \text{and} \quad \hat{x} \leftarrow \frac{x - \mathbb{E}[x]}{\sqrt{\text{Var}[x] + \epsilon}} \]
Batch Normalization in convolutional layers

- Normalize over mini-batch examples and nodes
- Normalization before nonlinearity: $y = g(BN(Wx))$
  - Invariant to the scale of $W$
Experiments

- Batch Normalization
  - reduces internal covariate shift
  - speeds up training of deep networks
  - sets state of the art in large-scale image recognition
Batch Normalization reduces internal covariate shift

- MNIST: 3 FC layers + softmax, 100 logistic units per hidden layer
- Distribution of inputs to a typical sigmoid, evolving over 100k steps:

![Graph showing distribution changes with and without Batch Normalization](image-url)
Batch Normalization reduces internal covariate shift

![Graph showing the effect of Batch Normalization on softmax/Eval Accuracy](image)

- **With BN**
- **Without BN**

*Training steps*
Experiment: ImageNet classification

- Inception: deep convolutional ReLU model
- Distributed SGD with momentum
- Batch Normalization applied at every convolutional layer
  - Extra cost (~30%) per training step
Inception with vs without Batch Normalization

- **Baseline:** 72.2% @ 31M steps
- **With BN:** 72.2% @ 13.3M steps
Further acceleration with Batch Normalization

- Batch Normalization enables higher learning rate
  - Increased 30x

- Removing dropout improves validation accuracy
  - Batch Normalization as a regularizer?
Higher learning rate, no dropout

- Baseline: 72.2% @ 31M steps
- Our best model: 72.2% @ 2.7M steps
  74.8% @ 6M steps
Saturating nonlinearities: Inception with logistic + BN

- Baseline: 0.1%
- With BN: 69.8%
Improving ImageNet classification

- Ensemble classifier
- Six Batch-Normalized Inception models
- Multi-crop, averaging over models and crops
## ImageNet classification: state of the art

<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><strong>4.82%</strong></td>
</tr>
</tbody>
</table>
Summary

- Reducing internal covariate shift speeds up training
- Batch Normalization using mini-batch mean and variance
- Preserve model expressivity
- Allows higher learning rates
- Reduces the need for dropout or careful parameter initialization
- Beats state of the art, and human accuracy, in ImageNet classification