Batch Normalization:

Accelerating Deep Network Training by Reducing Internal Covariate Shift

Sergey Ioffe
Christian Szegedy

Google

Research at Google

Stochastic gradient optimization in deep models

- Minimize loss over the training data

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

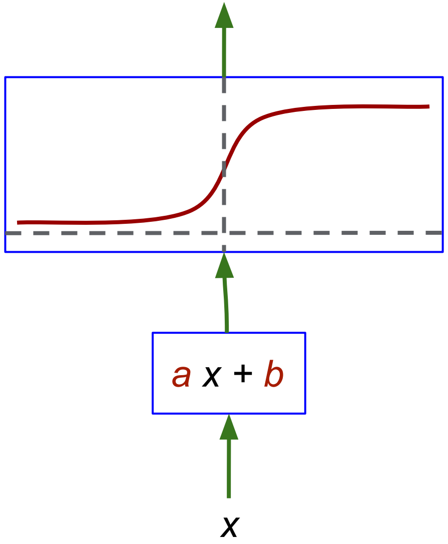
- Follow gradient for mini-batches

$$\Theta \leftarrow \Theta - \frac{\alpha}{m} \sum_{i=1}^{m} \frac{\partial \ell(\mathrm{x}_i, \Theta)}{\partial \Theta}$$

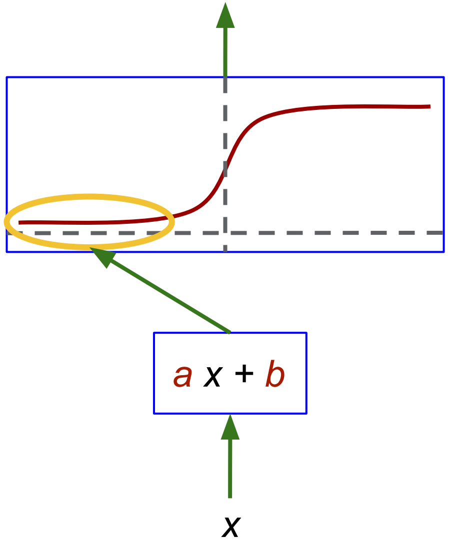
Outline

- 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

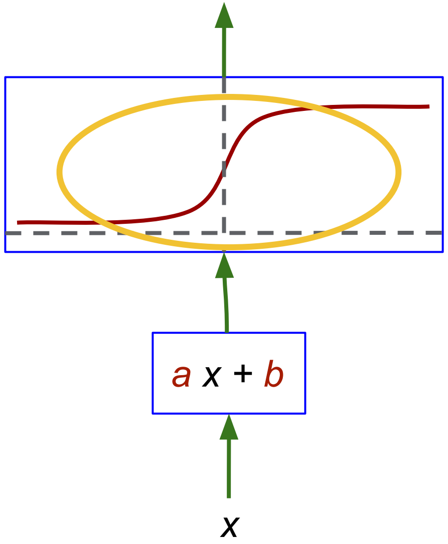
Care and training of deep models



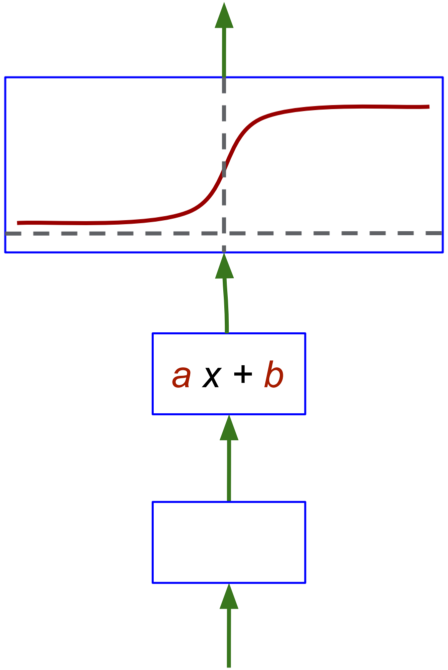
Care and training of deep models



Care and training of deep models

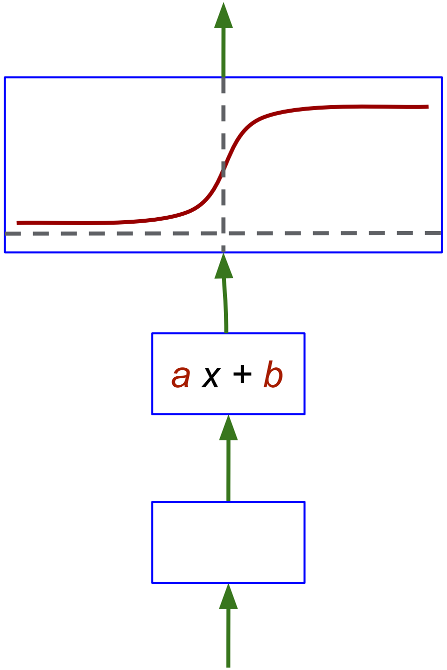


Care and training of deep models



Mitigating the effect of changing input distributions

- Careful initialization
- Small learning rates
- Rectifiers



Covariate shift

- Change in input distribution requires domain adaptation

$$\ell = F(\mathrm{x},\ \Theta)$$

Google

Internal covariate shift

- Layer input distributions change during training

$$\ell = F_2(\ F_1(\mathrm{u}, \Theta_1),\ \Theta_2)$$

- Change in internal activation distribution requires domain adaptation

Reducing internal covariate shift to speed up training

- Normalize each activation:

$$x \mapsto \frac{x - \mathrm{E}[x]}{\sqrt{\mathrm{Var}[x]}}$$

Normalization must participate in gradient optimization

- Mean and variance of an activation depend on model parameters
- Need $\frac{\partial \mathrm{E}[x]}{\partial \Theta}$ and $\frac{\partial \mathrm{Var}[x]}{\partial \Theta}$
- Cannot use population means and variances in mini-batch gradient optimization

Google

Batch Normalization

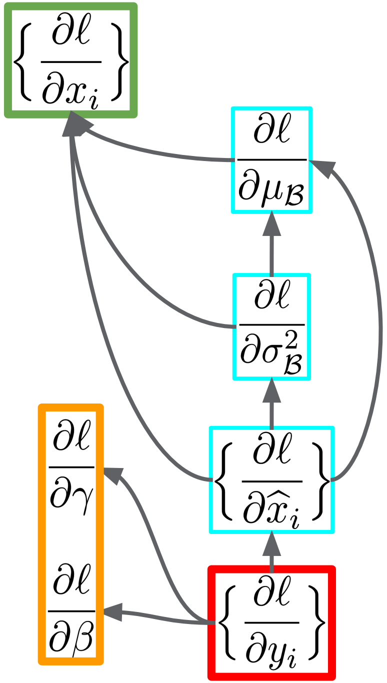
Mini-batch mean: $$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$$

Mini-batch variance: $$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2$$

Normalize: $$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}$$

Scale and shift: $$y_i \leftarrow \gamma\widehat{x}_i + \beta$$

Backprop with Batch Normalization



$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta$$

Inference with Batch Normalization

- Replace batch statistics with population statistics

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \quad \Rightarrow \quad \widehat{x} \leftarrow \frac{x - \mathrm{E}[x]}{\sqrt{\mathrm{Var}[x] + \epsilon}}$$

Batch Normalization in convolutional layers

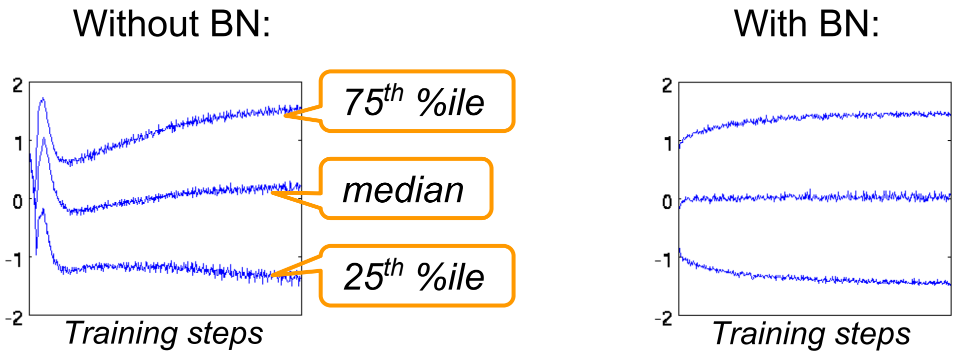
- Normalize over mini-batch examples *and* nodes
- Normalization before nonlinearity: $\mathrm{y} = g(\mathrm{BN}(W\mathrm{x}))$
    - Invariant to the scale of W

Experiments

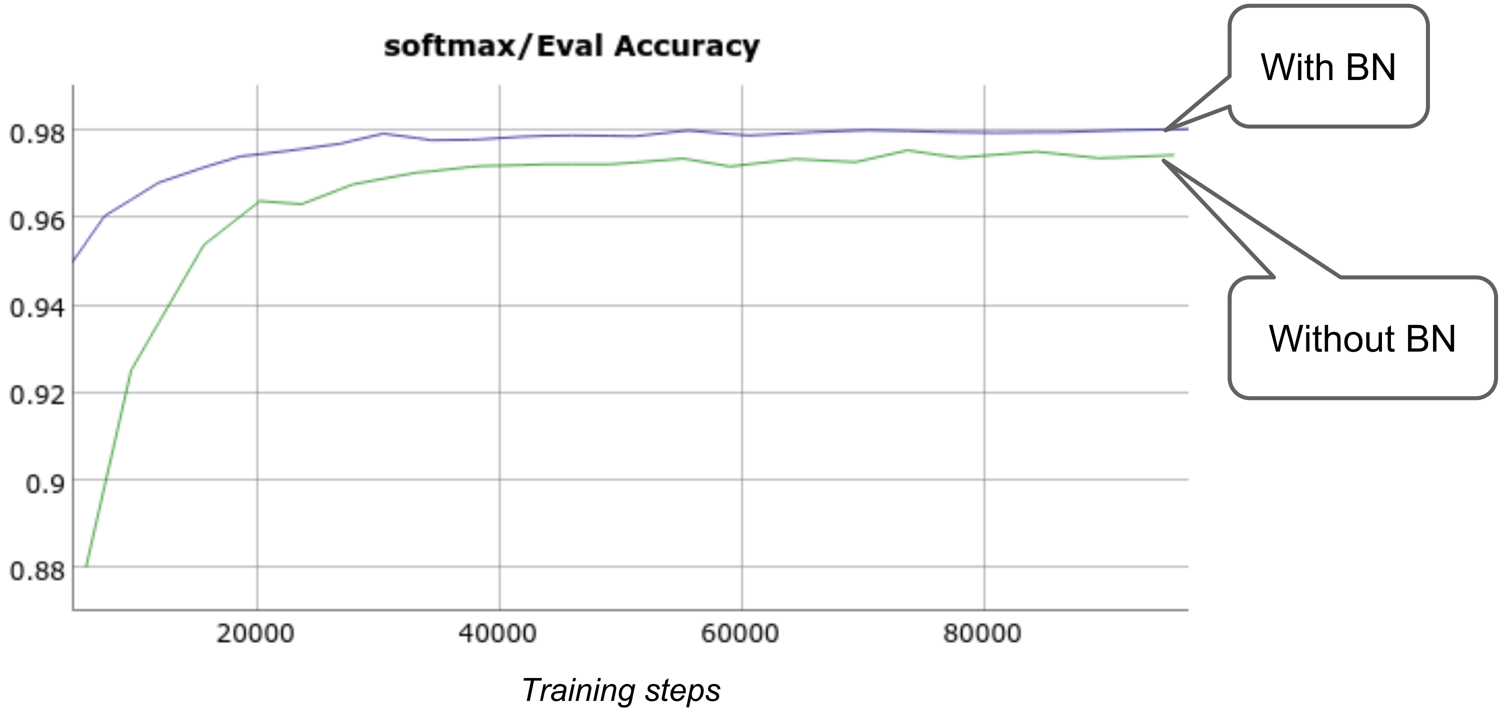
- Batch Normalization
  - reduces internal covariate shift
  - speeds up training of deep networks
  - sets state of the art in large-scale image recognition

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:

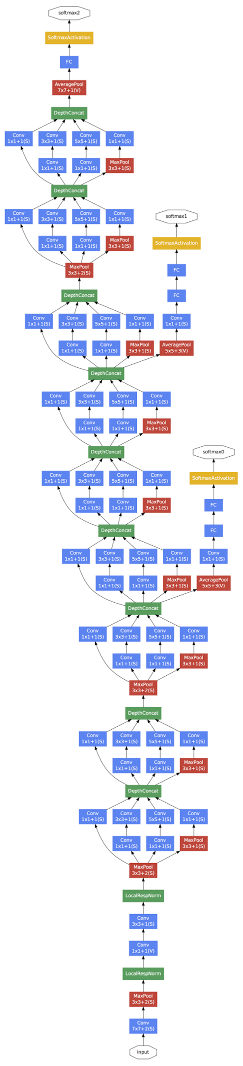


Batch Normalization reduces internal covariate shift

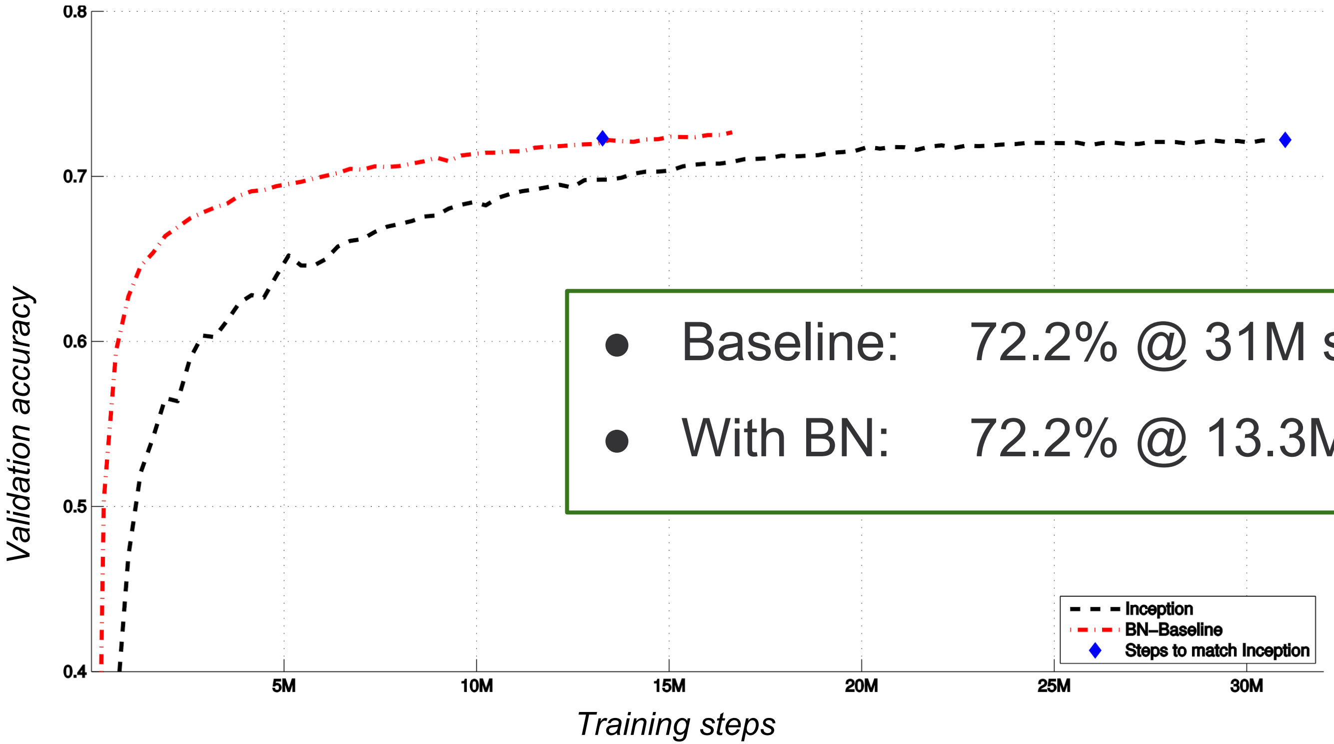


Experiment: ImageNet classification

- 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

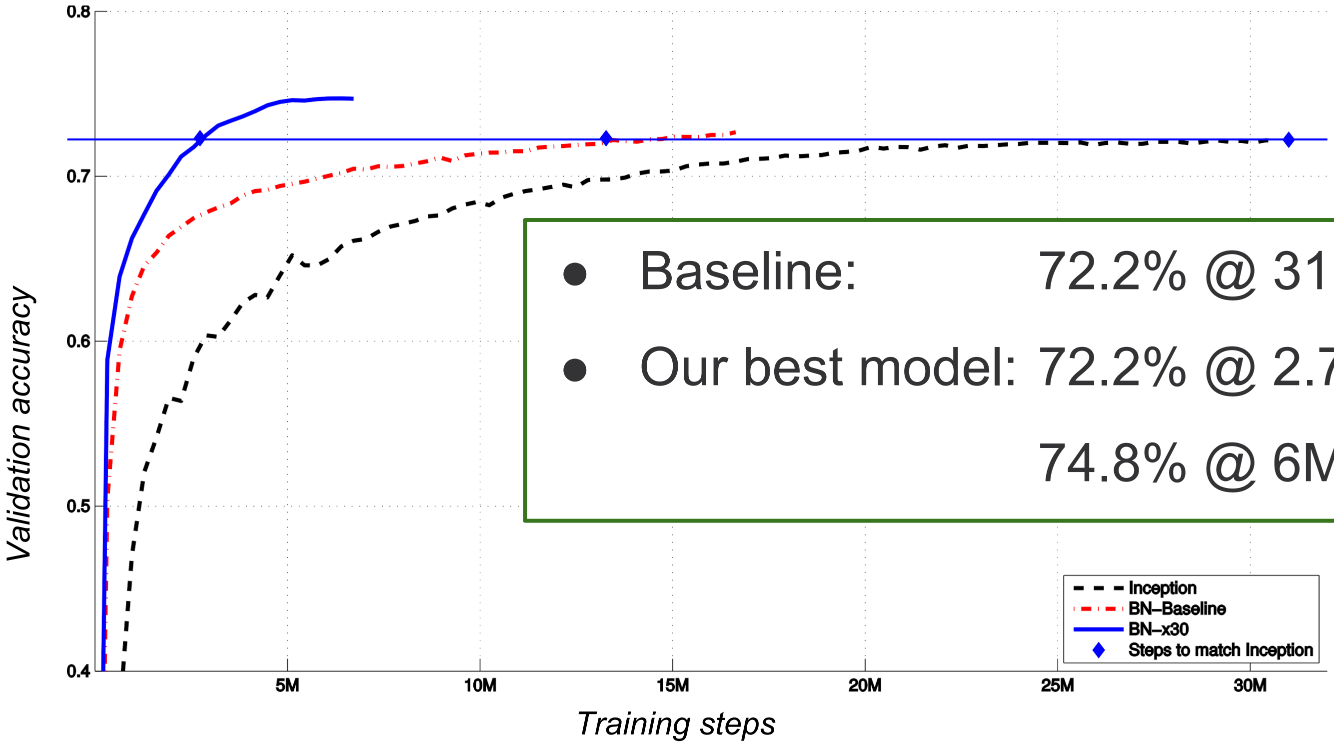


- Baseline: 72.2% @ 31M steps
- With BN: 72.2% @ 13.3M steps

Further acceleration with Batch Normalization

- Batch Normalization enables higher learning rate
    - Increased 30x
- Removing dropout improves validation accuracy
    - Batch Normalization as a regularizer?

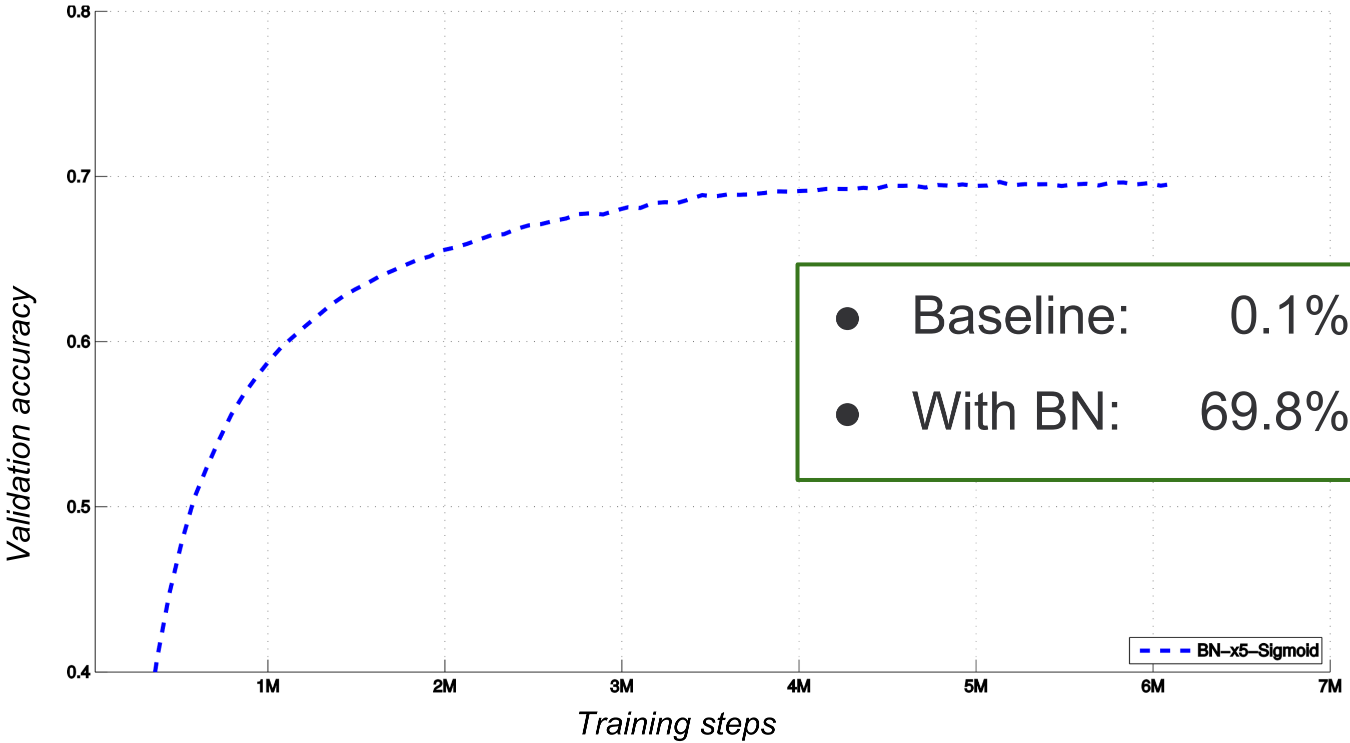
Higher learning rate, no dropout



- 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

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	up to 512	-	-	-	5.98%
MSRA multicrop	up to 480	-	-	-	5.71%
MSRA ensemble*	up to 480	-	-	-	4.94%
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%	**4.82%**

Google

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