

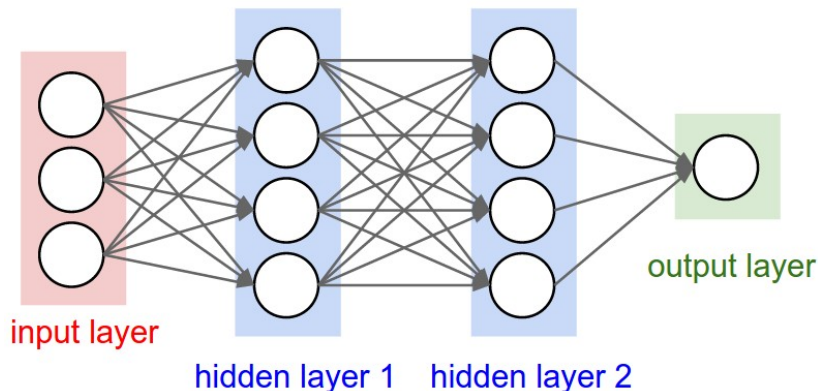
1000+ layers deep neural networks and how to train them

Daniil Polykovskiy

Feb. 12, 2016

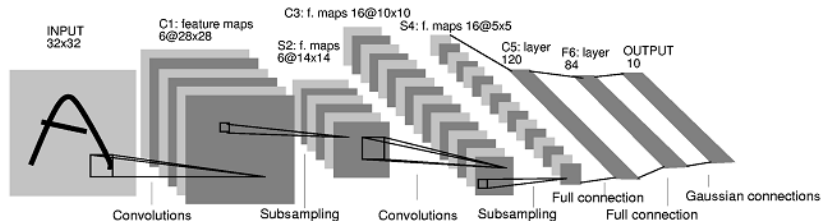
Quickly about neural networks

- ▶ Hierarchical ensemble of logistic regressions
- ▶ Output of a layer: $z^{i+1} = f(W^i \times z^i)$
- ▶ Universality theorem: $\forall F(x) \in C(X)$ exists a finite neural network with one hidden layer that computes approximation of $F(x)$



Convolutions

- ▶ $\hat{I}[i,j] = f(\sum_{k=1}^{channels} \sum_{p=-w}^w \sum_{q=-w}^w I[k, i+p, j+q] w[k,p,q])$
- ▶ Example architecture:



Residual networks

Proposed method:

- ▶ Insanely deep nets (**152** layers on ImageNet and **1202** layers on CIFAR-10)
- ▶ Lower complexity (VGG-19: 19.6×10^9 FLOP¹, ResNet-152: 11.3×10^9 FLOP)
- ▶ Faster training
- ▶ 1st place in ImageNet classification, ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation

¹Floating Point Operations

Problems with deep networks

- ▶ Vanishing / exploding gradients (better with ReLU)
- ▶ Degradation problem: quality drops on deeper nets

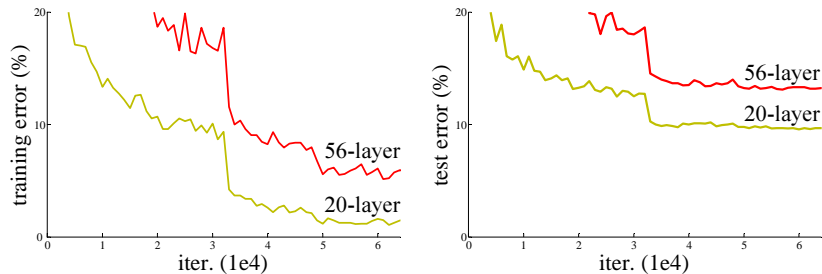


Figure: Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer networks.

- ▶ How to build a deeper model from shallow one?

Problems with deep networks

- ▶ Vanishing / exploding gradients (better with ReLU)
- ▶ Degradation problem: quality drops on deeper nets

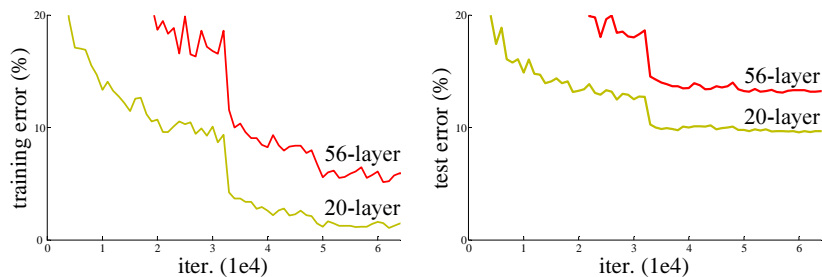


Figure: Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer networks.

- ▶ How to build a deeper model from shallow one?
- ▶ Add identity layers to the end of a trained net: $H(x) = x$

Residual connection

- ▶ Very deep nets are unable to represent identity mapping
- ▶ Example with ReLU: $\max(x + \infty, 0) - \infty \simeq x$
- ▶ **It may be easier to fit $F(x) = 0$ and add input to output:**
 $y = F(x) + x$

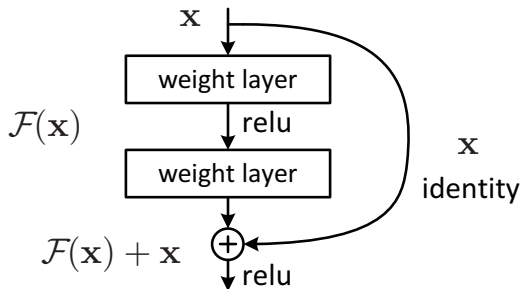
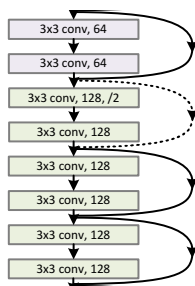


Figure: Residual learning: a building block.

Different dimensions



If input and output dimensions are different, $F(x) + x$ doesn't make sense. Ways to fix it:

- ▶ Don't use residual connection
- ▶ Add extra zeros at the end of vector x
- ▶ Projection shortcut (done by 1x1 convolutions):

$$y = F(x) + W_p \times x$$

Third method shows better results, but brings extra parameters.

Deeper nets

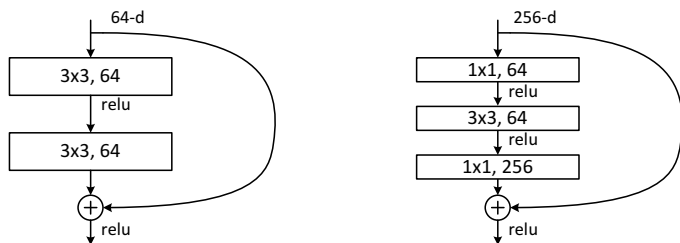


Figure: Left: a building block for ResNet-34. Right: a “bottleneck” building block for ResNet-50/101/152

How to go deeper:

- ▶ Reduce number of parameters with bottleneck block
- ▶ Stack a lot of bottleneck blocks
- ▶ 152-layer model still has less parameters than VGG-16!
- ▶ Important not to use projection shortcut (otherwise time x2)

Experiments

method	top-1 err.	top-5 err.
VGG (ILSVRC'14)	-	8.43
GoogLeNet(ILSVRC'14)	-	7.89
VGG	24.4	7.1
PReLU-net	21.59	5.71
BN-inception	21.99	5.81
ResNet-34 ²	21.84	5.71
ResNet-34 ³	21.53	5.60
ResNet-50	20.74	5.25
ResNet-101	19.87	4.60
ResNet-152	19.38	4.49

Table: Error rates (%) of **single-model** results on the ImageNet validation set.

²Projections when dimension changes

³Projections on each shortcut

Experiments

method	top-5 err. (test)
VGG(ILSVRC'14)	7.32
GoogLeNet (ILSVRC'14)	6.66
VGG (v5)	6.8
PReLU-net	4.94
BN-inception	4.82
ResNet (ILSVRC'15)	3.57

Table: Error rates (%) of **ensembles**. The top-5 error is on the test set of ImageNet and reported by the test server.

Batch normalization

Motivation

- ▶ Faster convergence with whitened inputs
- ▶ Whitening: $\hat{\mathbf{x}} = \text{Cov}[\mathbf{x}]^{-1/2}(\mathbf{x} - E[\mathbf{x}])$
- ▶ Normalization: $\hat{x}^{(k)} = \frac{x^{(k)} - E[x^{(k)}]}{\sqrt{\text{Var}[x^{(k)}]}}$ for each dimension

Batch normalization

- ▶ Covariate shift: changes of input distribution to a learning system

$$L = F(x, \theta)$$

- ▶ Internal covariate shift: Extension to the deep network

$$L = F_2(F_1(u, \theta_1), \theta_2) = F_2(x, \theta_2)$$

- ▶ Goal: reduce hidden layer activations' covariance shift
- ▶ In intermediate layers: $\hat{x}^{(k)} = \frac{x^{(k)} - E[x^{(k)}]}{\sqrt{\text{Var}[x^{(k)}]}}$

Problems with batch normalization

- 1 Computing $E[x]$ and $Var[x]$ is expensive

For each minibatch $\hat{x}^{(k)} = \frac{x^{(k)} - E_{\mathcal{B}}[x^{(k)}]}{\sqrt{Var_{\mathcal{B}}[x^{(k)}]}}$

- 2 Sigmoids would become almost linear

We want to be able to represent identity mapping

$$y^{(k)} = \gamma^{(k)} \hat{x}^{(k)} + \beta^{(k)}$$

Algorithm

Inputs: Values of x over a mini-batch $\mathcal{B} = \{x_{1\dots m}\}$;

Parameters: γ, β

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

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^m x_i \quad // \text{ mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2 \quad // \text{ mini-batch variance}$$

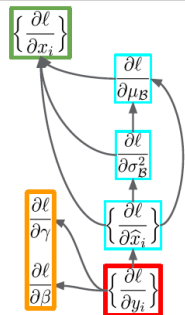
$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \quad // \text{ normalize}$$

$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma,\beta}(x_i) \quad // \text{ scale and shift}$$

Gradient

Apply chain rule

Note that μ_B and σ_B^2 aren't considered to be constants



$$\frac{\partial \ell}{\partial \hat{x}_i} = \frac{\partial \ell}{\partial y_i} \cdot \gamma$$

$$\frac{\partial \ell}{\partial \sigma_B^2} = \sum_{i=1}^m \frac{\partial \ell}{\partial \hat{x}_i} \cdot (x_i - \mu_B) \cdot \frac{-1}{2} (\sigma_B^2 + \epsilon)^{-3/2}$$

$$\frac{\partial \ell}{\partial \mu_B} = \left(\sum_{i=1}^m \frac{\partial \ell}{\partial \hat{x}_i} \cdot \frac{-1}{\sqrt{\sigma_B^2 + \epsilon}} \right) + \frac{\partial \ell}{\partial \sigma_B^2} \cdot \frac{\sum_{i=1}^m -2(x_i - \mu_B)}{m}$$

$$\frac{\partial \ell}{\partial x_i} = \frac{\partial \ell}{\partial \hat{x}_i} \cdot \frac{1}{\sqrt{\sigma_B^2 + \epsilon}} + \frac{\partial \ell}{\partial \sigma_B^2} \cdot \frac{2(x_i - \mu_B)}{m} + \frac{\partial \ell}{\partial \mu_B} \cdot \frac{1}{m}$$

$$\frac{\partial \ell}{\partial \gamma} = \sum_{i=1}^m \frac{\partial \ell}{\partial y_i} \cdot \hat{x}_i$$

$$\frac{\partial \ell}{\partial \beta} = \sum_{i=1}^m \frac{\partial \ell}{\partial y_i}$$

Inference

Deterministic input-output mapping at test time:

$$\hat{x} = \frac{x - E[x]}{\sqrt{\text{Var}[x] + \epsilon}}$$

$$y = \gamma \cdot \hat{x} + \beta$$

$$y = \frac{\gamma}{\sqrt{\text{Var}[x] + \epsilon}} \cdot x + \left(\beta - \frac{\gamma E[x]}{\sqrt{\text{Var}[x] + \epsilon}} \right)$$

$E[x]$ and $\text{Var}[x]$ are computed over the whole training set
exponential averaging in practice: $E_{i+1} = (1 - \alpha)E_i + \alpha E_{\mathcal{B}}$

Tips

After applying BN:

- ▶ Remove biases
- ▶ Increase learning rate
- ▶ Remove Dropout
- ▶ Reduce L_2 weight regularization
- ▶ Accelerate learning rate decay
- ▶ Shuffle training examples

Covariance shift

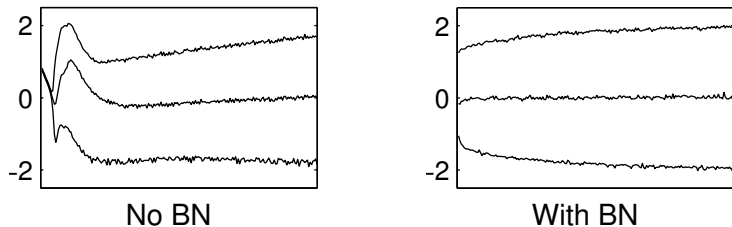
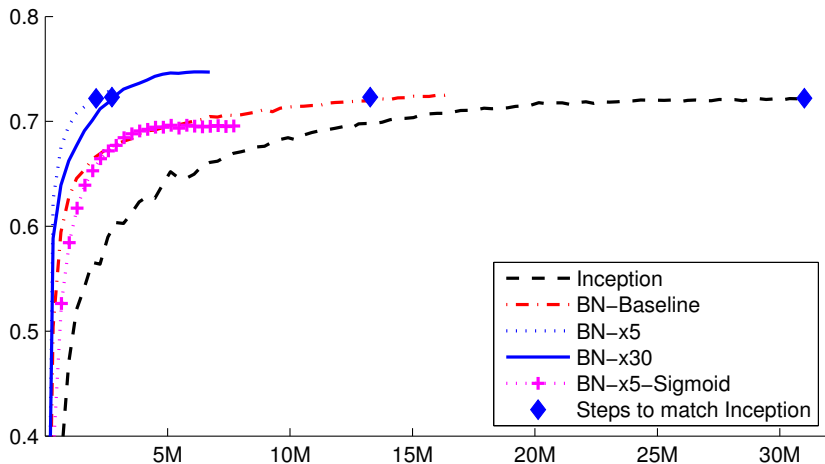


Figure: The evolution of input distributions to a typical sigmoid, over the course of training, shown as {15, 50, 85}th percentiles.

Learning



Single crop validation accuracy of Inception and its batch-normalized variants, vs. the number of training steps. (x30 means learning with 30 times larger learning rate)

Experiments

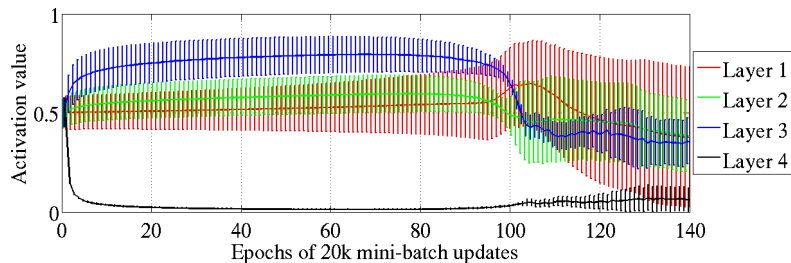
Model	Top-1 error	Top-5 error
GoogLeNet ensemble	-	6.67%
Deep Image low-res	-	7.96%
Deep Image high-res	24.88	7.42%
Deep Image ensemble	-	5.98%
BN-Inception single crop	25.2%	7.82%
BN-Inception multicrop	21.99%	5.82%
BN-Inception ensemble	20.1%	4.9%*

Figure: *Batch-Normalized Inception comparison with previous state of the art models*

Weight initialization

Motivation of tanh

Activations' evolution during training:



- ▶ After initialization it is better to ignore inputs
- ▶ With sigmoid $f(z) = \frac{1}{1+e^{-z}}$
- ▶ $f(z) = 0 \Rightarrow z = -\infty$
- ▶ Better with $\tanh(z) = \text{sh}(z)/\text{ch}(z) = (e^z - e^{-z})/(e^z + e^{-z})$
- ▶ $\tanh(z) = 0 \Rightarrow z = 0$

Xavier (Glorot)

Consider *tanh* activation

- ▶ Start in a non-saturated regime
- ▶ Avoid vanishing gradients

$$z^{i+1} = f(\underbrace{z^i W^i}_{s^i})$$

$$\text{Var}[z^i] = \text{Var}[x] \prod_{k=0}^{i-1} n_k \text{Var}[W^k]$$

$$\text{Var}\left[\frac{\partial L}{\partial s^i}\right] = \text{Var}\left[\frac{\partial L}{\partial s^d}\right] \prod_{k=i}^d n_{k+1} \text{Var}[W^k]$$

Where n_i is a dimension of i -th layer

Xavier (Glorot)

Good initialization:

$$\forall (i, j) \begin{cases} \text{Var}[z^i] = \text{Var}[z^j] \\ \text{Var}\left[\frac{\partial L}{\partial s^i}\right] = \text{Var}\left[\frac{\partial L}{\partial s^j}\right] \end{cases}$$

This is equivalent to

$$\forall i \begin{cases} n_i \text{Var}[W^i] = 1 \\ n_{i+1} \text{Var}[W^i] = 1 \end{cases}$$

Compromise: $\text{Var}[W^i] = \frac{2}{n_i + n_{i+1}}$

$$W^i \sim U\left[-\frac{\sqrt{6}}{\sqrt{n_i + n_{i+1}}}, \frac{\sqrt{6}}{\sqrt{n_i + n_{i+1}}}\right]$$

$$\text{Var}[U(a, b)] = \frac{1}{12}(b - a)^2$$

If activation is ReLU:

- ▶ Not symmetric
- ▶ Not differentiable at zero

$$\text{Var}[z^i] = \text{Var}[x] \left(\prod_{k=0}^{i-1} \frac{1}{2} n_k \text{Var}[W^k] \right) \Rightarrow \frac{1}{2} n_k \text{Var}[W^k] \Rightarrow \text{Var}[W^k] = \frac{2}{n_k}$$

$$\text{Var}\left[\frac{\partial L}{\partial s^i}\right] = \text{Var}\left[\frac{\partial L}{\partial s^d}\right] \left(\prod_{k=i}^d \frac{1}{2} n_{k+1} \text{Var}[W^k] \right) \Rightarrow \text{Var}[W^k] = \frac{2}{n_{k+1}}$$

It is sufficient to use first:

$$\text{Var}\left[\frac{\partial L}{\partial s^i}\right] = \text{Var}\left[\frac{\partial L}{\partial s^d}\right] \prod_{k=1}^d \frac{1}{2} n_{k+1} \text{Var}[W^k] = \frac{n_2}{n_d} \text{Var}\left[\frac{\partial L}{\partial s^d}\right]$$

n_2/n_d isn't big for convolution network

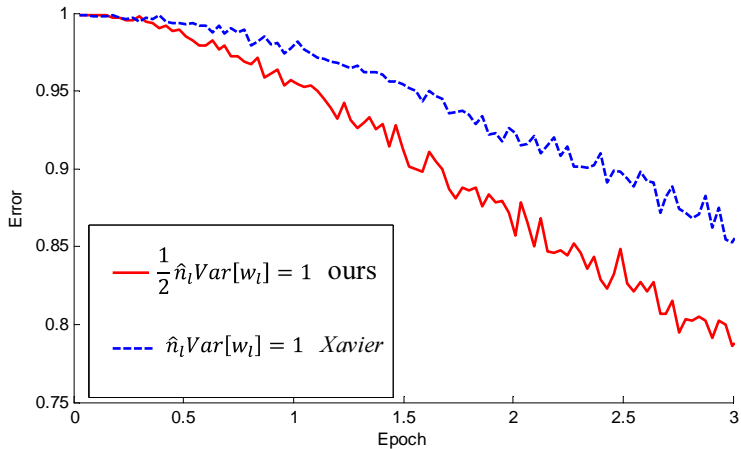
$$W^i \sim N\left(0, \frac{2}{n_i}\right)$$

or

$$W^i \sim N\left(0, \frac{2}{n_{i+1}}\right)$$

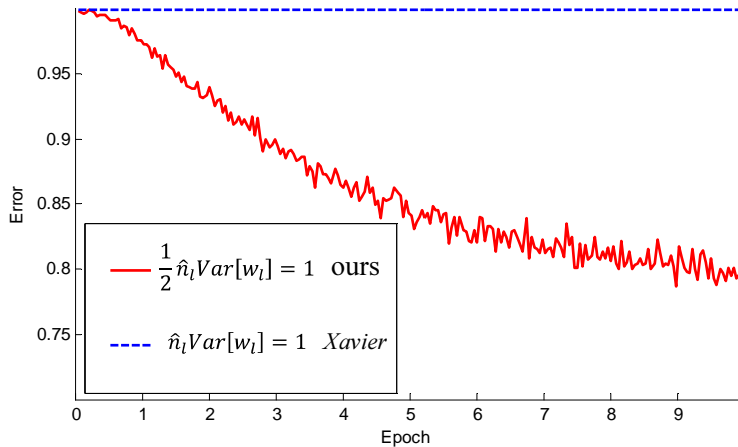
Xavier vs He for ReLU

22 layer network







Xavier vs He for ReLU

30 layer network



References

-  K. He, X. Zhang, S. Ren, J. Sun. *Deep Residual Learning for Image Recognition*
-  S. Ioffe, C. Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*
-  K. He, X. Zhang, S. Ren, J. Sun. *Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification*
-  X. Glorot, Y. Bengio. *Understanding the difficulty of training deep feedforward neural networks*

He, Var derivation

$$\text{Var}[s^l] = n_l \text{Var}[w_l x_l]$$

$\text{Var}[s^l] = n_l \text{Var}[w_l] E[x_l^2]$, if x_l and w_l have zero mean

$$x_l = \max(0, s^{l-1})$$

$$E[s^l] = 0, \text{ if } E[w_l] = 0$$

$$E[x_l^2] = \frac{1}{2} \text{Var}[s^{l-1}]$$

$$\text{Var}[s^l] = \frac{1}{2} n_l \text{Var}[w_l] \text{Var}[s^{l-1}]$$