Deep Residual Learning for Image Recognition (ResNet)

Journal club presentation

Christophe Ecabert

LTS5, EPFL

March 23rd, 2017

Reference

He *et al.* ***Deep Residual Learning for Image Recognition***
Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2016.

Recap

- Dropout
  - Learn better feature representation
  - Longer training time
- AlexNet
  - Depth matter
- GoogLeNet
  - *Inception* cell (Network in network)
  - 1x1 convolution for dimension reduction / adaptation
- Batch normalization
  - Accelerate training
  - Less sensitive to initialization
  - Improve regularization

Recap On Architecture

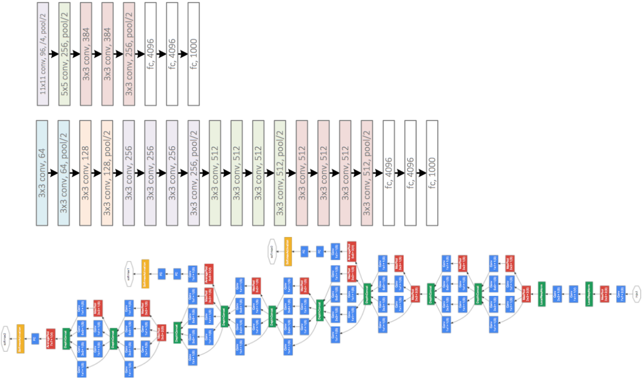

Figure 1: AlexNet (*8 layers*), VGG19 (*19 layers*), GoogLeNet (*22 layers*)

Exploding / Vanishing signals

- Single layer model

$$\mathbf{x}_l = f(\mathbf{y}_{l-1})$$

$$\mathbf{y}_l = \mathbf{W}_l \mathbf{x}_l + \mathbf{b}_l$$

- Single layer with ReLU activation function

$$Var[y_l] = \frac{1}{2} n_l Var[w_l] Var[y_{l-1}]$$

- With L layers

$$Var[y_l] = Var[y_1] \left(\prod_{l=2}^{L} \frac{1}{2} n_l Var[w_l] \right)$$

He *et al. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification* (2015)

Initialization

- Weight distribution requirements

$$\frac{1}{2} n_l Var[w_l] = 1, \quad \forall l$$

Therefore weight are initialized with zero mean gaussian noise with a standard deviation of $\sigma_l = \sqrt{2/n_l}$ and $\mathbf{b}_l = 0$. For the first layer, $n_1 Var[w_1] = 1$ should hold as well.

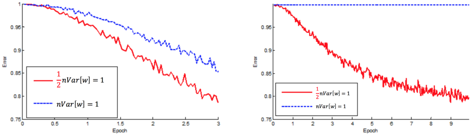

Figure 2: The convergence of a 22-layer and 30-layer model with ReLU.

Learning Better Network - Stacking layers

- Adding layers exposes a degradation problem, the accuracy decreases as the depth increases.
- Such degradation *is not caused by overfitting.*
- Considering the following experiment :
  - Train two networks, one shallow (*18 layers*) and one deep (*34 layers*).

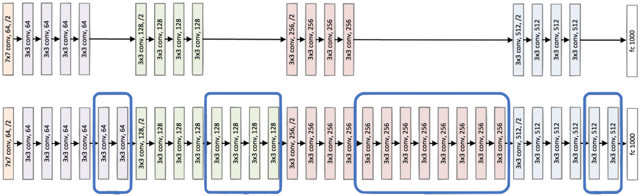

Figure 3: Experimental setup

Degradation problem

- Issues
  - Richer solution space
  - Solver can not find the solution when going deeper
- The deeper network should, in the worst case, have same performance as the shallow one since it exists a solution where the extra layers are identities (*i.e. same as shallow network*).

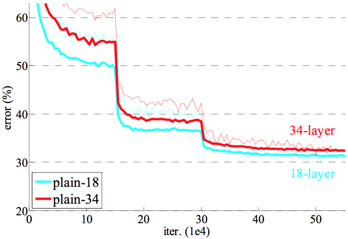

Figure 4: Training on *ImageNet*

Deep Residual Network

- **Plain** vs **Residuel** Network

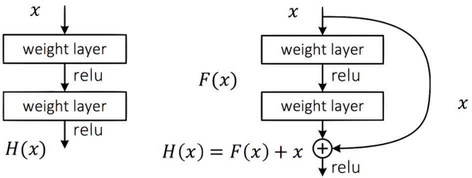

Figure 5: Mapping lerning : Plain vs Residual

- Design motivation
  - All *3x3* convolution or paired with *1x1*.
  - Feature maps size halfed, number of filter doubled (*preserves time complexity*).
  - No max-pooling, play with filter stride.
  - End with global average pooling layer + single fully connected.

Training

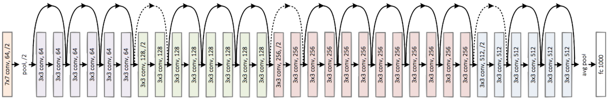

Figure 6: Residual Architecture

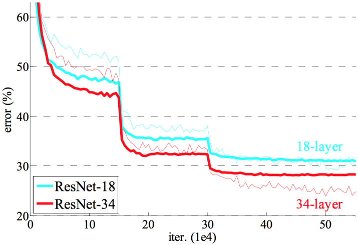

Figure 7: Training on *ImageNet*

Going Even Deeper

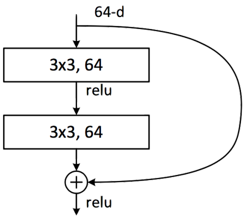

Figure 8: Deeper residual function $\mathcal{F}$ for *ImageNet*

ResNet Architecture

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 9: Deeper Architecture

Smooth Propagation Forward / Backward

- Plain network, multiplicative process.

$$x_L = \prod_{i=l}^{L-1} W_i x_l$$

- Residual network, cumulative process.

$$x_L = x_l + \sum_{i=l}^{L-1} F(x_i)$$

Results

- Training process
  - Data augmentation (*random crop, scale augmentation, ...*)
  - Per-pixel mean subtraction
  - Color augmentation (*PCA on RGB, add multipules of principal components*)
  - Batch Normalization after **each** convolution and **before** activation function
  - Weights initialization with proper standard deviation accroding to ReLU.
  - Train from scratch with standard *SGD*.

Results

Table 1: Error rate of **single-model** on the *ImageNet* validation set.

Method	Top 5% error
VGG (*ILSVRC14*)	8.43
GoogLeNet (*ILSVRC14*)	7.89
VGG (*v5*)	7.1
BN-Inception	5.81
ResNet-50	5.25
ResNet-101	4.60
ResNet-152	**4.49**

Results

Table 2: Error rate of ensembles on the *ImageNet* test set.

Method	Top 5% error
VGG (*ILSVRC14*)	7.32
GoogLeNet (*ILSVRC14*)	6.66
VGG (*v5*)	6.80
BN-Inception	4.82
ResNet (*ILSVRC15*)	**3.57**

Results

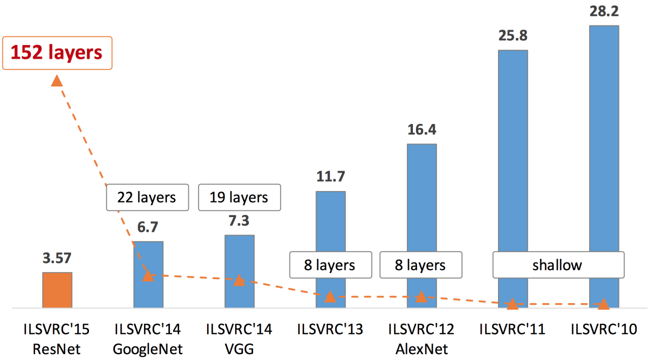

Figure 10: Results on *ImageNet*

Conclusions

- Residual architecture
  - Even with very deep structure, it has smaller complexity than plain network (*i.e. VGG*)
  - Features of any layers are additive outcomes
  - Enables smooth forward/backward propagation
  - Greatly eases the optimization of the model