An Overview of Deep Residual Learning

Semih Yagcioglu

01.03.2016
Deep Residual Learning

• Microsoft Research Asia (MSRA)


• ILSVRC & COCO 2015 competitions
MSRA @ ILSVRC & COCO 2015 Competitions

• *1st places* in all five main tracks
  • ImageNet Classification: “Ultra-deep” 152-layer nets
  • ImageNet Detection: 16% better than 2nd
  • ImageNet Localization: 27% better than 2nd
  • COCO Detection: 11% better than 2nd
  • COCO Segmentation: 12% better than 2nd

*improvements are relative numbers*
Microsoft researchers win ImageNet computer vision challenge

Jian Sun, a principal research manager at Microsoft Research, led the image understanding project. Photo: Craig Tuschhoff/Microsoft.

Posted December 10, 2015 By Allison Linn
The researchers say even they weren’t sure this new approach was going to be successful — until it was.

“We even didn’t believe this single idea could be so significant,” said Jian Sun, a principal research manager at Microsoft Research who led the image understanding project along with teammates Kaiming He, Xiangyu Zhang and Shaoqing Ren in Microsoft’s Beijing research lab.

The major leap in accuracy surprised others as well. Peter Lee, a corporate vice president in charge of Microsoft Research’s NExT labs, said he was shocked to see such a major breakthrough.

“It sort of destroys some of the assumptions I had been making about how the deep neural networks work,” he said.
Deep Residual Learning for Image Recognition

Latest commit 16a8453 4 days ago

File uploading is now available
You can now drag and drop files into your repositories. Learn more

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>caffe @ b590f1d</td>
<td>add submodule of caffe version at 2016/2/3</td>
<td>25 days ago</td>
</tr>
<tr>
<td>prototxt</td>
<td>prototxt</td>
<td>24 days ago</td>
</tr>
<tr>
<td>.gitmodules</td>
<td>add submodule of caffe version at 2016/2/3</td>
<td>25 days ago</td>
</tr>
<tr>
<td>LICENSE</td>
<td>Initial commit</td>
<td>26 days ago</td>
</tr>
<tr>
<td>README.md</td>
<td>Update README.md</td>
<td>4 days ago</td>
</tr>
</tbody>
</table>
Deep Residual Learning for Image Recognition

26 commits, 1 branch, 0 releases, 3 contributors

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>caffe</td>
<td>add submodule of caffe version at 2016/2/3</td>
<td>25 days ago</td>
</tr>
<tr>
<td>prototxt</td>
<td>prototxt</td>
<td>24 days ago</td>
</tr>
<tr>
<td>.gitmodules</td>
<td>add submodule of caffe version at 2016/2/3</td>
<td>25 days ago</td>
</tr>
<tr>
<td>LICENSE</td>
<td>Initial commit</td>
<td>26 days ago</td>
</tr>
<tr>
<td>README.md</td>
<td>Update README.md</td>
<td>4 days ago</td>
</tr>
</tbody>
</table>
Revolution of Depth

ImageNet Classification top-5 error (%)

- ILSVRC’15 ResNet: 3.57
- ILSVRC’14 GoogleNet: 6.7
- ILSVRC’14 VGG: 7.3
- ILSVRC’13: 11.7
- ILSVRC’12 AlexNet: 16.4
- ILSVRC’11: 25.8
- ILSVRC’10: 28.2

Revolution of Depth

152 layers
Revolution of Depth

Engines of visual recognition

HOG, DPM
AlexNet (RCNN)
VGG (RCNN)
ResNet (Faster RCNN)*

PASCAL VOC 2007 Object Detection mAP (%)

34 shallow
58 8 layers
66 16 layers
86 101 layers

* w/ other improvements & more data

Slide Credit: He et al. (MSRA)
Residual learning reformulates the learning procedure and redirects the information flow in deep neural networks.
Revolution of Depth

AlexNet, 8 layers (ILSVRC 2012)

11x11 conv, 96, /4, pool/2

5x5 conv, 256, pool/2

3x3 conv, 384

3x3 conv, 384, 3x3

conv, 256, pool/2

fc, 4096

fc, 4096

fc, 1000
Revolution of Depth

AlexNet, 8 layers (ILSVRC 2012)
- 11x11 conv, 96, /4, pool/2
- 5x5 conv, 256, pool/2
- 3x3 conv, 256, pool/2
- 3x3 conv, 384
- 3x3 conv, 384
- 3x3 conv, 256, pool/2
- fc, 4096
- fc, 4096
- fc, 1000

VGG, 19 layers (ILSVRC 2014)
- 3x3 conv, 64
- 3x3 conv, 64, pool/2
- 3x3 conv, 128
- 3x3 conv, 128, pool/2
- 3x3 conv, 256
- 3x3 conv, 256
- 3x3 conv, 256
- 3x3 conv, 256
- 3x3 conv, 256, pool/2
- 3x3 conv, 512
- 3x3 conv, 512
- 3x3 conv, 512
- 3x3 conv, 512
- 3x3 conv, 512, pool/2
- 3x3 conv, 512
- 3x3 conv, 512
- 3x3 conv, 512
- 3x3 conv, 512, pool/2
- fc, 4096
- fc, 4096
- fc, 1000

GoogleNet, 22 layers (ILSVRC 2014)
- 3x3 conv, 64
- 3x3 conv, 64, pool/2
- 3x3 conv, 128
- 3x3 conv, 128, pool/2
- 3x3 conv, 256
- 3x3 conv, 256
- 3x3 conv, 256
- 3x3 conv, 256
- 3x3 conv, 256
- 3x3 conv, 256, pool/2
- 3x3 conv, 512
- 3x3 conv, 512
- 3x3 conv, 512
- 3x3 conv, 512
- 3x3 conv, 512
- 3x3 conv, 512
- 3x3 conv, 512
- 3x3 conv, 512, pool/2
- fc, 4096
- fc, 4096
- fc, 1000

Slide Credit: He et al. (MSRA)
Revolution of Depth

AlexNet, 8 layers (ILSVRC 2012)

VGG, 19 layers (ILSVRC 2014)

ResNet, 152 layers (ILSVRC 2015)
Revolution of Depth

ResNet, 152 layers
Revolution of Depth

ResNet, 152 layers
Revolution of Depth

ResNet, 152 layers
Revolution of Depth

ResNet, 152 layers

Slide Credit: He et al. (MSRA)
Is learning better networks as simple as stacking more layers?
Simply stacking layers?

- *Plain* nets: stacking 3x3 conv layers...
- 56-layer net has higher *training error* and test error than 20-layer net

---

CIFAR-10

**train error (%)**

**test error (%)**

<table>
<thead>
<tr>
<th>Iter. (1e4)</th>
<th>Train Error (%)</th>
<th>Test Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>20-layer</td>
<td>56-layer</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

Slide Credit: He et al. (MSRA)
Simply stacking layers?

- “Overly deep” plain nets have higher training error
- A general phenomenon, observed in many datasets

Slide Credit: He et al. (MSRA)
A shallower model (18 layers)

A deeper counterpart (34 layers)

- A deeper model should not have higher training error
- A solution by construction:
  - original layers: copied from a learned shallower model
  - extra layers: set as identity
  - at least the same training error
- Optimization difficulties: solvers cannot find the solution when going deeper...
Deep Residual Learning

• Plain net

\[ x \rightarrow \text{weight layer} \rightarrow \text{relu} \rightarrow \text{weight layer} \rightarrow \text{relu} \rightarrow H(x) \]

H(x) is any desired mapping, hope the 2 weight layers fit F(x)
Deep Residual Learning

• Residual net

$H(x) = F(x) + x$

$H(x)$ is any desired mapping, hope the 2 weight layers fit $H(x)$

let $H(x) = F(x) + x$
Deep Residual Learning

- $F(x)$ is a residual mapping w.r.t. identity

- If identity were optimal, easy to set weights as 0

- If optimal mapping is closer to identity, easier to find small fluctuations

$$H(x) = F(x) + x$$
Building Block Oversimplified

So currently they have "skip" layers with zero parameters...

ie. if you have layers like this:

<table>
<thead>
<tr>
<th>Input</th>
<th>→</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>→</th>
<th>Output</th>
</tr>
</thead>
</table>

- A takes as input the Input
- B takes as input A
- C takes as input B
- D takes as input A + C
- E takes as input D
- F takes as input E
- G takes as input D + F
Related Works – Residual Representations

- **VLAD & Fisher Vector** [Jegou et al 2010], [Perronnin et al 2007]
  - Encoding *residual* vectors; powerful shallower representations.

- **Product Quantization (IVF-ADC)** [Jegou et al 2011]
  - Quantizing *residual* vectors; efficient nearest-neighbor search.

  - Solving *residual* sub-problems; efficient PDE solvers.
More Related Work


  Introduce a new architecture designed to ease gradient-based training of very deep networks. This architecture allows unimpeded information flow across several layers on "information highways". Use a soft gate that depends on the data, so $0 < \alpha < 1$ times the activations go through the layer, and $1 - \alpha$ is directly forwarded to the next layer (where $\alpha$ is a function of the activation).

- **LSTM Networks** - Hochreiter & Schmidhuber (1997)

  Designed to avoid the long-term dependency problem. Remembering information for long periods of time. Each memory cell is associated with an input gate, an output gate and an internal state that feeds into itself unperturbed across time steps.
Network “Design”

• Keep it simple

• Our basic design (VGG-style)
  • all 3x3 conv (almost)
  • spatial size /2 => # filters x2
  • Simple design; just deep!

• Other remarks:
  • no max pooling (almost)
  • no hidden fc
  • no dropout
A Residual Block
Training

• All plain/residual nets are trained from scratch

• All plain/residual nets use Batch Normalization

• Standard hyper-parameters & augmentation
CIFAR-10 experiments

- Deep ResNets can be trained without difficulties
- Deeper ResNets have lower training error, and also lower test error
ImageNet experiments

- Deep ResNets can be trained without difficulties
- Deeper ResNets have **lower training error**, and also lower test error!
ImageNet experiments

• A practical design of going deeper

all-3x3

bottleneck (for ResNet-50/101/152)
ImageNet experiments

- Deeper ResNets have lower error

this model has lower time complexity than VGG-16/19

<table>
<thead>
<tr>
<th>Model</th>
<th>Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-152</td>
<td>5.7</td>
</tr>
<tr>
<td>ResNet-101</td>
<td>6.1</td>
</tr>
<tr>
<td>ResNet-50</td>
<td>6.7</td>
</tr>
<tr>
<td>ResNet-34</td>
<td>7.4</td>
</tr>
</tbody>
</table>

10-crop testing, top-5 val error (%)
ImageNet experiments

ImageNet Classification top-5 error (%)

ILSVRC'15
ResNet

ILSVRC'14
GoogleNet

ILSVRC'14
VGG

ILSVRC'13

ILSVRC'12
AlexNet

ILSVRC'11

ILSVRC'10
shallow

152 layers

3.57
6.7
7.3
11.7
16.4
25.8
28.2

Slide Credit: He et al. (MSRA)
Just classification?

A treasure from ImageNet is on learning features.
“Features matter.” (quote [Girshick et al. 2014], the R-CNN paper)

<table>
<thead>
<tr>
<th>task</th>
<th>2nd-place winner</th>
<th>MSRA</th>
<th>margin (relative)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ImageNet Localization (top-5 error)</td>
<td>12.0</td>
<td>9.0</td>
<td>27%</td>
</tr>
<tr>
<td>ImageNet Detection (mAP@.5)</td>
<td>53.6 <strong>absolute 8.5% better!</strong></td>
<td>62.1</td>
<td>16%</td>
</tr>
<tr>
<td>COCO Detection (mAP@.5:.95)</td>
<td>33.5</td>
<td>37.3</td>
<td>11%</td>
</tr>
<tr>
<td>COCO Segmentation (mAP@.5:.95)</td>
<td>25.1</td>
<td>28.2</td>
<td>12%</td>
</tr>
</tbody>
</table>

- Their results are all based on ResNet-101
- Their features are well transferrable
Figure 1: Left: Region Proposal Network (RPN). Right: Example detections using RPN proposals on PASCAL VOC 2007 test. Our method detects objects in a wide range of scales and aspect ratios.
Results on COCO - too many objects, let’s check carefully!
Third-party re-implementations

Deep residual networks are very easy to implement and train. We recommend to see also the following third-party re-implementations and extensions:

1. By Facebook AI Research (FAIR), with training code in Torch and pre-trained ResNet-18/34/50/101 models for ImageNet: blog, code
2. Torch, CIFAR-10, with ResNet-20 to ResNet-110, training code, and curves: code
3. Lasagne, CIFAR-10, with ResNet-32 and ResNet-56 and training code: code
5. Torch, MNIST, 100 layers: blog, code
6. A winning entry in Kaggle's right whale recognition challenge: blog, code
7. Neon, Place2 (mini), 40 layers: blog, code

In addition, this code by Ryan Dahl helps to convert the pre-trained models to TensorFlow.
Training at a larger scale: ImageNet

We trained variants of the 18, 34, 50, and 101-layer ResNet models on the ImageNet classification dataset. What’s notable is that we achieved error rates that were better than the published results by using a different data augmentation method.

We are also training a 152-layer ResNet model, but the model has not finished converging at the time of this post.

We used the scale and aspect ratio augmentation described in "Going Deeper with Convolutions" instead of the scale augmentation described in the ResNet paper. With ResNet-34, this improved top-1 validation error by about 1.2% points. We also used the color augmentation described in "Some Improvements on Deep Convolutional Neural Network Based Image Classification," but found that had a very small effect on ResNet-34.
Implementation
-- The basic residual layer block for 18 and 34 layer network, and the
-- CIFAR networks
local function basicblock(n, stride)
    local nInputPlane = iChannels
    iChannels = n

    local s = nn.Sequential()
    s:add(Convolution(nInputPlane, n, 3, 3, stride, stride, 1, 1))
    s:add(SBatchNorm(n))
    s:add(ReLU(true))
    s:add(Convolution(n, n, 3, 3, 1, 1, 1, 1))
    s:add(SBatchNorm(n))

    return nn.Sequential()
        :add(nn.ConcatTable()
            :add(s)
            :add(shortcut(nInputPlane, n, stride)))
        :add(nn.CAddTable(true))
        :add(ReLU(true))
end
-- The shortcut layer is either identity or 1x1 convolution

local function shortcut(nInputPlane, nOutputPlane, stride)
local useConv = shortcutType == 'C' or
    (shortcutType == 'B' and nInputPlane ~= nOutputPlane)
if useConv then
    -- 1x1 convolution
    return nn.Sequential()
        :add(Convolution(nInputPlane, nOutputPlane, 1, 1, stride, stride))
        :add(SBatchNorm(nOutputPlane))
elseif nInputPlane ~= nOutputPlane then
    -- Strided, zero-padded identity shortcut
    return nn.Sequential()
        :add(nn.SpatialAveragePooling(1, 1, stride, stride))
        :add(nn.Concat(2)
            :add(nn.Identity())
            :add(nn.MulConstant(0)))
else
    return nn.Identity()
end
end
end
-- The bottleneck residual layer for 50, 101, and 152 layer networks

local function bottleneck(n, stride)
    local nInputPlane = iChannels
    iChannels = n * 4

    local s = nn.Sequential()
    s:add(Convolution(nInputPlane, n, 1, 1, 1, 0, 0))
    s:add(SBatchNorm(n))
    s:add(ReLU(true))
    s:add(Convolution(n, n, 3, 3, stride, stride, 1, 1))
    s:add(SBatchNorm(n))
    s:add(ReLU(true))
    s:add(Convolution(n, n*4, 1, 1, 1, 0, 0))
    s:add(SBatchNorm(n * 4))

    return nn.Sequential()
        :add(nn.ConcatTable())
        :add(s)
            :add(Shortcut(nInputPlane, n * 4, stride))
        :add(nn.CAddTable(true))
        :add(ReLU(true))
end
Experiments

- There are currently a few implementations of Deep Residual Learning
  - Torch (will be covered in detail in the upcoming weeks)
  - A few Python based implementation (mxnet, lasagne)
  - I have played with a few these implementations
    - Most of them are quite easy to grasp
    - But again, they are built upon existing frameworks
    - If you are not familiar with the abstractions, i.e. how things are implemented underneath, then it might not make sense much
    - Torch implementation is the best I have seen in terms of readability and understandability
How to Get Started with Lasagne Implementation

1. Install Requirements (sudo pip install -r https://raw.githubusercontent.com/Lasagne/Lasagne/master/requirements.txt)
2. Install Lasagne (sudo pip install https://github.com/Lasagne/Lasagne/archive/master.zip)
3. It will probably fail. Fix things accordingly.
4. Download this: https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
5. Extract the cifar-10 dataset into a folder
6. Download this folder and code: https://github.com/Lasagne/Recipes/tree/master/papers/deep_residual_learning
7. Run Deep_Residual_Learning_CIFAR-10.py
Conclusions

• Amazing results!
• Simple idea
• Deeper is better
• Features matter
• Their building block is so simple and effective that it will change the direction of the upcoming works!
  • Actually it just did!
• Inception-v4 (3.08% Top-5 error)
• Torch implementation reports better results than the paper!
Hot From the Oven

ImageNet Classification Error (Top 5)


Slide Credit: Fei-Fei Li & Andrej Karpathy & Justin Johnson
Inception-v4

Figure 9. The overall schema of the Inception-v4 network. For the detailed modules, please refer to Figures 3, 4, 5, 6, 7 and 8 for the detailed structure of the various components.

Figure 11. The schema for 17 × 17 grid (Inception-ResNet-B) module of Inception-ResNet-v1 network.

Final Remarks

- Seems quite easy to implement.
- Several third party frameworks are adopting this method.
- Torch implementation will be described in detail in the upcoming Torch tutorial!
- Finally the idea is quite simple and yields amazing results!
- Compared to previous works, requires less hardware and time!