Learning Deep Features for Visual Recognition

CVPR 2017 Tutorial

Kaiming He
Facebook AI Research (FAIR)

covering joint work with:
Xiangyu Zhang, Shaoqing Ren, Jian Sun, Saining Xie, Zhuowen Tu, Ross Girshick, Piotr Dollar
Outline

- Introduction
- Convolutional Neural Networks: Recap
  - LeNet, AlexNet, VGG, GoogleNet; Batch Norm
- ResNet
- ResNeXt

slides will be available online
Revolution of Depth

ILSVRC'15
ResNet

ILSVRC'14
GoogleNet
8 layers
6.7

ILSVRC'14
VGG
19 layers
7.3

ILSVRC'13
8 layers
11.7

ILSVRC'12
AlexNet
8 layers
16.4

ILSVRC'11
shallow
25.8

ILSVRC'10
28.2

ImageNet Classification top-5 error (%)

Engine of Visual Recognition

- HOG, DPM
  - Shallow (34 layers)

- AlexNet (RCNN)
  - 8 layers (58 mAP)

- VGG (RCNN)
  - 16 layers (66 mAP)

- ResNet (Faster RCNN)*
  - 101 layers (86 mAP)

PASCAL VOC 2007 Object Detection mAP (%)

*with other improvements & more data

Engine of Visual Recognition

ResNets/extensions are leading models on popular benchmarks

- Detection: COCO/VOC
- Segmentation: COCO/VOC/ADE/Cityscape
- Visual Reasoning: VQA/CLEVR
- Video: UCF101/HMDB
- ...

Search “ResNet” on ILSVRC2016 result page returns 226 entries

Source: Ross Girshick
How did computer recognize an image?

But what’s next?

[pixels → classifier → “bus”?]

[edges → classifier → “bus”?]

[SIFT/HOG → edges → histogram → classifier → “bus”?]

[K-means sparse code FV/VLAD → classifier → “bus”?]

[Lazebnik et al 2006], [Perronnin & Dance 2007], [Yang et al 2009], [Jégou et al 2010], …
Learning Deep Features

Specialized components, domain knowledge required

- edges
- histogram
- K-means sparse code FV/VLAD
- classifier
- “bus”?

Generic components/“layers”, less domain knowledge

Repeat elementary layers: going deeper

- “bus”?

- Richer solution space
- End-to-end learning by BackProp
Convolutional Neural Networks: Recap

LeNet, AlexNet, VGG, GoogleNet; Batch Norm,...
LeNet

• Convolution:
  • locally-connected
  • spatially weight-sharing
    • weight-sharing is a key in DL (e.g., RNN shares weights temporally)

• Subsampling

• Fully-connected outputs

• Train by BackProp

• All are still the basic components of modern ConvNets!

“Gradient-based learning applied to document recognition”, LeCun et al. 1998
“Backpropagation applied to handwritten zip code recognition”, LeCun et al. 1989
AlexNet

LeNet-style backbone, plus:

• **ReLU** [Nair & Hinton 2010]
  • “RevoLUtion of deep learning”*
  • Accelerate training; better grad prop (vs. tanh)

• **Dropout** [Hinton et al 2012]
  • In-network ensembling
  • Reduce overfitting (might be instead done by BN)

• **Data augmentation**
  • Label-preserving transformation
  • Reduce overfitting

*Quote Christian Szegedy

“ImageNet Classification with Deep Convolutional Neural Networks”, Krizhevsky, Sutskever, Hinton. NIPS 2012
VGG-16/19

“16 layers are beyond my imagination!”
-- after ILSVRC 2014 result was announced.

Simply “Very Deep”!
• Modularized design
  • 3x3 Conv as the module
  • Stack the same module
  • Same computation for each module (1/2 spatial size => 2x filters)

• Stage-wise training
  • VGG-11 => VGG-13 => VGG-16
  • We need a better initialization...

Initialization

Input $X$  $\Rightarrow$  Output $Y = WX$

Weight $W$

1-layer:
$$Var[y] = (n^{in}Var[w])Var[x]$$

Multi-layer:
$$Var[y] = \left(\prod_d n_d^{in}Var[w_d]\right)Var[x]$$

If:
• Linear activation
• $x, y, w$: independent

Then:

LeCun et al 1998 “Efficient Backprop”
Glorot & Bengio 2010 “Understanding the difficulty of training deep feedforward neural networks”
Initialization

Both forward (response) and backward (gradient) signal can vanish/explode

Forward:
\[ \text{Var}[y] = (\prod_d n_d^{\text{in}} \text{Var}[w_d]) \text{Var}[x] \]

Backward:
\[ \text{Var} \left[ \frac{\partial}{\partial x} \right] = (\prod_d n_d^{\text{out}} \text{Var}[w_d]) \text{Var} \left[ \frac{\partial}{\partial y} \right] \]

LeCun et al 1998 “Efficient Backprop”
Glorot & Bengio 2010 “Understanding the difficulty of training deep feedforward neural networks”
Initialization: “Xavier”

• Initialization under \textbf{linear} assumption

\[
\prod_d n_d^{in} \text{Var}[w_d] = const_{fw} \text{ (healthy forward)} \quad \text{and} \quad \prod_d n_d^{out} \text{Var}[w_d] = const_{bw} \text{ (healthy backward)}
\]

\[
\begin{align*}
n_d^{in} \text{Var}[w_d] &= 1 \\
\text{or} \\
n_d^{out} \text{Var}[w_d] &= 1
\end{align*}
\]

LeCun et al 1998 “Efficient Backprop”
Glorot & Bengio 2010 “Understanding the difficulty of training deep feedforward neural networks”
Initialization: “MSRA”

- Initialization under **ReLU**

\[
\prod_d \frac{1}{2} n_d^{\text{in}} \text{Var}[w_d] = \text{const}_{\text{fw}} \text{ (healthy forward)}
\]

and

\[
\prod_d \frac{1}{2} n_d^{\text{out}} \text{Var}[w_d] = \text{const}_{\text{bw}} \text{ (healthy backward)}
\]

\[
\frac{1}{2} n_d^{\text{in}} \text{Var}[w_d] = 1
\]

\[
\text{or}
\]

\[
\frac{1}{2} n_d^{\text{out}} \text{Var}[w_d] = 1
\]

With \( D \) layers, a factor of 2 per layer has exponential impact of \( 2^D \)

---

Initialization

Xavier/MSRA init

• Required for training VGG-16/19 from scratch
• Deeper (>20) VGG-style nets can be trained w/ MSRA init
  • but deeper plain nets are not better (see ResNets)
• Recommended for newly initialized layers in fine-tuning
  • e.g., Fast/er RCNN, FCN, etc.

\[ \sqrt{\frac{1}{n}} \] or \[ \sqrt{\frac{2}{n}} \] doesn’t directly apply to multi-branch nets (e.g., GoogleNet)
  • but the same derivation methodology is applicable
  • does not matter, if BN is applicable...

GoogleNet/Inception

Accurate with small footprint.

My take on GoogleNets:

- Multiple branches
  - e.g., 1x1, 3x3, 5x5, pool
- Shortcuts
  - stand-alone 1x1, merged by concat.
- Bottleneck
  - Reduce dim by 1x1 before expensive 3x3/5x5 conv

GoogleNet/Inception v1-v3

More templates, but the same 3 main properties are kept:

• Multiple branches
• Shortcuts (1x1, concate.)
• Bottleneck

Batch Normalization (BN)

- Recap: Xavier/MSRA init are not directly applicable for multi-branch nets

- Optimizing multi-branch ConvNets largely benefits from BN
  - including all Inceptions and ResNets

Batch Normalization (BN)

• Recap: Normalizing image input (LeCun et al 1998 “Efficient Backprop”)

• Xavier/MSRA init: Analytic normalizing each layer

• BN: data-driven normalizing each layer, for each mini-batch
  • Greatly accelerate training
  • Less sensitive to initialization
  • Improve regularization

Batch Normalization (BN)

\[ \hat{x} = \frac{x - \mu}{\sigma} \Rightarrow y = \gamma \hat{x} + \beta \]

- \( \mu \): mean of \( x \) in mini-batch
- \( \sigma \): std of \( x \) in mini-batch
- \( \gamma \): scale
- \( \beta \): shift

- \( \mu, \sigma \): functions of \( x \), analogous to responses
- \( \gamma, \beta \): parameters to be learned, analogous to weights

Batch Normalization (BN)

\[
\text{layer} \rightarrow x \rightarrow \hat{x} = \frac{x - \mu}{\sigma} \rightarrow y = \gamma \hat{x} + \beta
\]

2 modes of BN:

- **Train mode:**
  - $\mu, \sigma$ are functions of a batch of $x$
- **Test mode:**
  - $\mu, \sigma$ are pre-computed* on training set

**Caution:** make sure your BN usage is correct!
(this causes many of my bugs in my research experience!)

*: by running average, or post-processing after training

Batch Normalization (BN)

Figure credit: Ioffe & Szegedy

ResNets
Simply stacking layers?

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

Simply stacking layers?

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

a shallower model (18 layers)

a deeper counterpart (34 layers)

- Richer solution space
- 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 small subnet fit \( H(x) \)

Deep Residual Learning

- **Residual net**

\[
H(x) = F(x) + x
\]

\(H(x)\) is any desired mapping, hope the small subnet fit \(H(x)\)

Hope the small subnet fit \(F(x)\)

Let \(H(x) = F(x) + x\)
Deep Residual Learning

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

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

- If identity were optimal, easy to set weights as 0
- If optimal mapping is closer to identity, easier to find small fluctuations

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

ImageNet experiments

- Deeper ResNets have lower error


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

- 10-crop testing, top-5 val error (%)
ResNets beyond computer vision

- **Neural Machine Translation (NMT):** 8-layer LSTM!

ResNets beyond computer vision

- **Speech Synthesis** (WaveNet): Residual CNNs on 1-d sequence

ResNets beyond computer vision

- **Speech Recognition** – Residual CNNs on 1-d sequence

ResNeXt

to be presented in CVPR 2017
“Aggregated Residual Transformations for Deep Neural Networks”
Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He.
Multi-branch

• (Recap): shortcut, bottleneck, and **multi-branch**

**Inception:**
heterogeneous multi-branch

**ResNeXt:**
uniform multi-branch

ResNeXt

- **Concatenation and Addition are interchangeable**
  - General property for DNNs; not only limited to ResNeXt
- **Uniform multi-branching can be done by group-conv**

ResNeXt

• Better accuracy
  • when having the same FLOPs/#params as ResNet

• Better trade-off of larger models

ResNeXt for Mask R-CNN

<table>
<thead>
<tr>
<th>Faster R-CNN+++ [19]</th>
<th>ResNet-101-C4</th>
<th>AP^{bb}</th>
<th>AP^{bb}_{50}</th>
<th>AP^{bb}_{75}</th>
<th>AP^{bb}_S</th>
<th>AP^{bb}_M</th>
<th>AP^{bb}_L</th>
</tr>
</thead>
<tbody>
<tr>
<td>Faster R-CNN w FPN [27]</td>
<td>ResNet-101-FPN</td>
<td>34.9</td>
<td>55.7</td>
<td>37.4</td>
<td>15.6</td>
<td>38.7</td>
<td>50.9</td>
</tr>
<tr>
<td>Faster R-CNN by G-RMI [21]</td>
<td>Inception-ResNet-v2 [37]</td>
<td>36.2</td>
<td>59.1</td>
<td>39.0</td>
<td>18.2</td>
<td>39.0</td>
<td>48.2</td>
</tr>
<tr>
<td>Faster R-CNN w TDM [36]</td>
<td>Inception-ResNet-v2-TDM</td>
<td>34.7</td>
<td>55.5</td>
<td>36.7</td>
<td>13.5</td>
<td>38.1</td>
<td>52.0</td>
</tr>
<tr>
<td>Faster R-CNN, RoIAlign</td>
<td>ResNet-101-FPN</td>
<td>36.8</td>
<td>57.7</td>
<td>39.2</td>
<td>16.2</td>
<td>39.8</td>
<td>52.1</td>
</tr>
<tr>
<td>Mask R-CNN</td>
<td>ResNet-101-FPN</td>
<td>37.3</td>
<td>59.6</td>
<td>40.3</td>
<td>19.8</td>
<td>40.2</td>
<td>48.8</td>
</tr>
<tr>
<td>Mask R-CNN</td>
<td>ResNeXt-101-FPN</td>
<td><strong>39.8</strong></td>
<td><strong>62.3</strong></td>
<td><strong>43.4</strong></td>
<td><strong>22.1</strong></td>
<td><strong>43.2</strong></td>
<td><strong>51.2</strong></td>
</tr>
</tbody>
</table>

ResNeXt improves 1.6 bbox AP (and 1.4 mask AP) on COCO

Feature still matters!
More architectures (not covered in this tutorial)

- Inception-ResNet [Szegedy et al 2017]
  - Inception as transformation + residual connection
- DenseNet [Huang et al CVPR 2017]
  - Densely connected shortcuts w/ concat.
- Xception [Chollet CVPR 2017], MobileNets [Howard et al 2017]
  - DepthwiseConv (i.e., GroupConv with group=#channel)
- ShuffleNet [Zhang et al 2017]
  - More Group/DepthwiseConv + shuffle
- ……
Training ImageNet in 1 Hour

- 256 GPUs
- 8,192 mini-batch size
- ResNet-50
- No loss of accuracy

Key factors
- Linear scaling learning rate in minibatch size
- Warmup
- Implement things correctly in multiple GPUs/machines!

Figure 1. ImageNet top-1 validation error vs. minibatch size.

Conclusion: Features Matter!

Deep features empower amazing visual recognition results (Mask R-CNN w/ ResNet101; more in next talk)