Learning Deep Representations for Visual Recognition

CVPR 2018 Tutorial

Kaiming He
Facebook AI Research (FAIR)
Deep Learning is Representation Learning

Representation Learning: worth a conference name 😊 (ICLR)

Represent (raw) data for machines to perform tasks:
• Vision: pixels, ...
• Language: letters, ...
• Speech: waves, ...
• Games: status, ...
Representation Learning: AlphaGo

$3^{361}$ states?
Representation Learning: AlphaGo

$3^{361}$ states?

$256^{3 \times 640 \times 480}$?
Representation Learning: AlphaGo

$3^{361}$ states?

256$^{3*640*480}$?

Bad representations

Good representations

models (now, neural nets)
Representation Learning: AlphaGo

20-block to 40-block ResNet

12-layer VGG to 20-block ResNet

“Mastering the game of Go without human knowledge”, Silver et al. Nature 2017
How was an image represented?

But what’s next?

Learning to represent

Specialized components, domain knowledge required

Generic components, less domain knowledge

Repeat **elementary** layers: going deeper

• End-to-end by BackProp
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 Methods

• Analytical formulations of normalizing forward/backward signals
• Based on strong assumptions (like Gaussian distributions)

• Xavier Init (linear): $n \cdot Var[w] = 1$
• MSRA Init (ReLU): $n \cdot Var[w] = 2$

“Understanding the difficulty of training deep feedforward neural networks” Glorot & Bengio, 2010
“Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification” Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun, ICCV 2015
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, v2, v3, ...

More templates, but the same 3 main properties are kept:
• Multiple branches
• Shortcuts (1x1, concate.)
• Bottleneck

Batch Normalization (BN)

• 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 normalization, for each layer, for each mini-batch
  • Greatly accelerate training
  • Less sensitive to initialization
  • Improve regularization

Batch Normalization (BN)

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

- \( \mu \): 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)

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

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!)

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

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

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

\[ F(x) \]

weight layer

relu

weight layer

identity

\[ x \]

\[ H(x) \text{ is any desired mapping, hope the small subnet fit } H(x) \]

\[ \text{hope the small subnet fit } F(x) \]

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

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

\[
F(x) = \text{weight layer} \circ \text{relu} \circ \text{weight layer} \circ \text{relu} + x
\]

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

• 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

<table>
<thead>
<tr>
<th>Model</th>
<th>10-crop testing, top-5 val 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>

ResNet beyond computer vision

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

ResNet beyond computer vision

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

ResNet beyond computer vision

• **AlphaGo Zero**: 40 Residual Blocks

"Mastering the game of Go without human knowledge", Silver et al. Nature 2017
ResNeXt

• 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 a baseline ResNet

• Better trade-off for high-capacity models
Competition winners using ResNeXt

ResNeXt is a good trade-off for high-capacity:

- ImageNet Classification 2017, 1\textsuperscript{st} place
  - SE-ResNeXt
- COCO Object Detection 2017, 1\textsuperscript{st} place
  - MegDet + ResNeXt
- COCO Instance Segmentation 2017, 1\textsuperscript{st} place
  - PANet + ResNeXt
- COCO Stuff Segmentation 2017, 1\textsuperscript{st} place
  - FPN + ResNetXt
- ...

ResNeXt: higher capacity for billion-scale images

Fig. 5: Classification accuracy on val-IN-1k using ResNeXt-101 32×{4, 8, 16, 32, 48}d with and without pretraining on the IG-940M-1.5k dataset.

Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten.
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
- ....
Teaser: Group Normalization (GN)

• Independent of batch size

• Robust to small batches

• Enable new scenarios: e.g.: 41 AP on COCO trained from scratch

Yuxin Wu and Kaiming He. “Group Normalization”. arXiv 2018
Conclusion

• Deep Learning is Representation Learning

• Represent data for machines to perform tasks (this talk)

• Represent data for machines to perform tasks (next talks)