Outline

1. Recap MLP
2. Convolutional Neural Networks
3. Large deep convnets
   - GoogLeNet
   - VGG
   - ResNet
Outline

1. Recap MLP
2. Convolutional Neural Networks
3. Large deep convnets
   - GoogLeNet
   - VGG
   - ResNet
Recap AlexNet: What’s next?

How to improve AlexNet architecture?
+++Deep?
+++Convolutional?
+++Fully connected?
All?
⇒A lot of empirical studies
⇒Tuning various design parameters
⇒what really works?
⇒Winners: VGG, GoogleNets, ResNet
Outline

1. Recap MLP
2. Convolutional Neural Networks
3. Large deep convnets
   • GoogLeNet
GoogLeNet (2014)

Winner of ILSVRC - 2014. Very deep network with 22 layers:
- Network-in-network-in-network
- Removed fully connected layers → small # of parameters (5M weights)
GoogLeNet (2014)

Inception layer
GoogLeNet (2014)

Auxiliary classifiers

Main classifier
Outline

1. Recap MLP
2. Convolutional Neural Networks
3. Large deep convnets
   - GoogLeNet
   - VGG
VGG Net: Archi post-2012 revolution

K. Simonyan, A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, ICLR 2015
VGG Net

Basic Idea: Investigate the **effect of depth** in large scale image recognition

- **Fix other parameters** of architecture, and steadily increase depth
Fixed configuration:

- Convolutional Layers: from 8 to 16
- Fully Connected Layers: 3
- Stride: 1
- ReLu: Follow all hidden layers
- Max-Pooling: 2x2 window
- Padding: s/t spatial resolution is preserved
- #Convolutional filters: Starting from 64, double after each max-pooling layer until 512
- Filter sizes: 3x3 and 1x1
# ConvNet Configuration

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>A-LRN</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>11 weight layers</td>
<td>11 weight layers</td>
<td>13 weight layers</td>
<td>16 weight layers</td>
<td>16 weight layers</td>
<td>19 weight layers</td>
</tr>
</tbody>
</table>

## Input (224 x 224 RGB Image)

<table>
<thead>
<tr>
<th></th>
<th>conv3-64</th>
<th>conv3-64</th>
<th>conv3-64</th>
<th>conv3-64</th>
<th>conv3-64</th>
<th>conv3-64</th>
</tr>
</thead>
<tbody>
<tr>
<td>maxpool</td>
<td>conv3-128</td>
<td>conv3-128</td>
<td>conv3-128</td>
<td>conv3-128</td>
<td>conv3-128</td>
<td>conv3-128</td>
</tr>
</tbody>
</table>

## Maxpool

<table>
<thead>
<tr>
<th></th>
<th>conv3-256</th>
<th>conv3-256</th>
<th>conv3-256</th>
<th>conv3-256</th>
<th>conv3-256</th>
<th>conv3-256</th>
</tr>
</thead>
<tbody>
<tr>
<td>maxpool</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
</tr>
</tbody>
</table>

## Maxpool

<table>
<thead>
<tr>
<th></th>
<th>conv3-512</th>
<th>conv3-512</th>
<th>conv3-512</th>
<th>conv3-512</th>
<th>conv3-512</th>
<th>conv3-512</th>
</tr>
</thead>
<tbody>
<tr>
<td>maxpool</td>
<td>FC-4096</td>
<td>FC-4096</td>
<td>FC-1000</td>
<td>soft-max</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**Table Credit:** Very Deep Convolutional Networks for Large-Scale Image Recognition, ICLR2015
VGG Net

Results:

- First place in localization (25.3% error), second in classification (7.3% error) in ILSVRC 2014 using ensemble of 7 networks
- Outperforms Szegedy et.al (GoogLeNet) in terms of single network classification accuracy (7.1% vs 7.9%)
Observations with VGG testing:

- Deepnets with small filters outperform shallow networks with large filters
  - Shallow version of B: 2 layers of 3x3 replaced with single 5x5 performs worse
- LRN does not improve performance
- Classification error decreases with increases ConvNet depth
- Important to capture more spatial context (config D vs C)
- Error rate saturated at 19 layers
- Scale jittering at training helps capturing multiscale statistics and leads to better performance
Outline

1. Recap MLP
2. Convolutional Neural Networks
3. Large deep convnets
   • GoogLeNet
   • VGG
   • ResNet
Deep ConvNets for image classification

- **ResNet** 152 layers, 60M parameters

Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun
Deep Residual Learning for Image Recognition.
In *CVPR*, 2016.
Deep ConvNets for image classification

**Revolution of Depth**

- **ILSVRC'15**: ResNet
  - 3.57% error
- **ILSVRC'14**:GoogleNet
  - 6.7% error
- **ILSVRC'14**: VGG
  - 7.3% error
- **ILSVRC'13**: AlexNet
  - 11.7% error
- **ILSVRC'12**: Shallow
  - 16.4% error
- **ILSVRC'11**: Shallow
  - 25.8% error
- **ILSVRC'10**: Shallow
  - 28.2% error

ImageNet Classification top-5 error (%)
ResNet
The deeper, the better

+ Deeper network covers more complex problems
  • Receptive field size ↑
  • Non-linearity ↑
- Training deeper network more difficult because of vanishing/exploding gradients problem

@ Kaiming He ILSVRC & COCO 2015
Deeper VGG: 56 Plain Network

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

“Overly deep” plain nets have higher training error

A general phenomenon, observed in many datasets
Residual Network

Naïve solution
If extra layers identity mapping, training error not increase
Residual Network

• Deeper networks maintain the tendency of results
  Features in same level will be almost same
  An amount of changes is fixed
  Adding layers make smaller differences
  Optimal mappings closer to an identity
Residual Network

Plain block

Difficult to make identity mapping because of multiple non-linear layers
Residual Network

Residual block

If identity were optimal, easy to set weights as 0

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

→ Appropriate for treating perturbation as keeping a base information
Residual Network

• Difference between an original image and a changed image

Preserving base information

Some Network

residual

can treat perturbation
Residual Network

Deeper ResNets have lower training error
Residual Network

• Residual block
  • Very simple
  • Parameter-free

A naïve residual block “bottleneck” residual block
(for ResNet-50/101/152)
Residual Network

• Shortcuts connections
  • Identity shortcuts
    \[ y = \mathcal{F}(x, \{W_i\}) + x. \]
  • Projection shortcuts
    \[ y = \mathcal{F}(x, \{W_i\}) + W_s x. \]
Network Design

Basic design (VGG-style)
- All 3x3 conv (almost)
- Spatial size/2 => #filters x2
- Batch normalization
- Simple design, just deep

Other remarks
- No max pooling (almost)
- No hidden fc
- No dropout
Network Design

ResNet-152

Use bottlenecks

ResNet-152 (11.3 billion FLOPs) lower complexity than VGG-16/19 nets (15.3/19.6 billion FLOPs)

About 64M parameters
Results

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

• Deep Resnets can be trained “without difficulties”
• Deeper ResNets have lower training error, and also lower test error
Results

• 1st places in all five main tracks in “ILSVRC & COCO 2015 Competitions”
  • ImageNet Classification
  • ImageNet Detection
  • ImageNet Localization
  • COCO Detection
  • COCO Segmentation
Deep ConvNets for image classification

- **ResNeXt**
  - Multi-branch architecture

Saining Xie, Ross Girshick, Piotr Dollàr, Zhuowen Tu and Kaiming He
Aggregated Residual Transformations for Deep Neural Networks.
FractalNet: Ultra-Deep Neural Networks without Residuals

Comments from paper: “Our experiments demonstrate that residual representation is not fundamental to the success of extremely deep convolutional neural networks. A fractal design achieves an error rate of 22.85% on CIFAR-100, matching the state-of-the-art held by residual networks.”
Exploring type of deep modules in Neural Nets

Neural Architecture Search
Conclusion

• ResNet: currently the best archi for large scale image classification

• Not yet consensus about the design of the Net (cf. FractalNet), Neural Architecture Search

• Fully Convolutional Net (FCN) very interesting option

• To be studied
  • ResNet 50
  • Vision transformers
Batch normalization

- Improves gradient flow through the network
- Allows higher learning rates
- Reduces the strong dependence on initialization
- Reduces need for dropout

Un-normalization!! Re-compute and apply the optimal scaling and bias for each neuron!

Learn $\gamma$ and $\beta$ (same dims as $\mu$ and $\sigma^2$).
It can (should?) learn the identity mapping!

**Input:** Values of $x$ over a mini-batch: $\mathcal{B} = \{x_1...m\};$

Parameters to be learned: $\gamma, \beta$

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

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

$$\sigma_B^2 \leftarrow \frac{1}{m} \sum_{i=1}^{m} (x_i - \mu_B)^2 \quad \text{// mini-batch variance}$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \quad \text{// normalize}$$

$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma,\beta}(x_i) \quad \text{// scale and shift}$$
Batch Normalization Gradients [Ioffe and Szegedy, 2015]

\[
\frac{\partial \ell}{\partial x_i} = \frac{\partial \ell}{\partial y_i} \cdot \gamma
\]

\[
\frac{\partial \ell}{\partial \sigma_B^2} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial x_i} \cdot (x_i - \mu_B) \cdot \frac{-1}{2} \left( \frac{\sigma_B^2}{\sigma_B^2 + \epsilon} \right)^{-3/2}
\]

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

\[
\frac{\partial \ell}{\partial x_i} = \frac{\partial \ell}{\partial x_i} \cdot \frac{1}{\sqrt{\sigma_B^2 + \epsilon}} + \frac{\partial \ell}{\partial \sigma_B^2} \cdot \frac{2(x_i - \mu_B)}{m} + \frac{\partial \ell}{\partial \mu_B} \cdot \frac{1}{m}
\]

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

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

Don’t need these directly, they are subexpressions for the other gradients.

Think of this as backprop for the nodes $\hat{x}, \sigma_B^2, \mu_B$, which are all internal to the minibatch update.
Batch Normalization Gradients \cite{loffe2015}

\[
\frac{\partial l}{\partial x_i} = \frac{\partial l}{\partial y_i} \cdot \gamma
\]

\[
\frac{\partial l}{\partial \sigma_B^2} = \sum_{i=1}^{m} \frac{\partial l}{\partial x_i} \cdot (x_i - \mu_B) \cdot \frac{-1}{2} (\sigma_B^2 + \epsilon)^{-3/2}
\]

\[
\frac{\partial l}{\partial \mu_B} = \left( \sum_{i=1}^{m} \frac{\partial l}{\partial x_i} \cdot \frac{-1}{\sqrt{\sigma_B^2 + \epsilon}} \right) + \frac{\partial l}{\partial \sigma_B^2} \cdot \frac{\sum_{i=1}^{m} -2(x_i - \mu_B)}{m}
\]

Gradient to propagate to the input layer

\[
\frac{\partial l}{\partial x_i} = \frac{\partial l}{\partial x_i} \cdot \frac{1}{\sqrt{\sigma_B^2 + \epsilon}} + \frac{\partial l}{\partial \sigma_B^2} \cdot \frac{2(x_i - \mu_B)}{m} + \frac{\partial l}{\partial \mu_B} \cdot \frac{1}{m}
\]

\[
\frac{\partial l}{\partial \gamma} = \sum_{i=1}^{m} \frac{\partial l}{\partial y_i} \cdot \hat{x}_i
\]

\[
\frac{\partial l}{\partial \beta} = \sum_{i=1}^{m} \frac{\partial l}{\partial y_i}
\]
Batch Normalization Gradients [Ioffe and Szegedy, 2015]

\[
\frac{\partial l}{\partial x_i} = \frac{\partial l}{\partial y_i} \cdot \gamma
\]

\[
\frac{\partial l}{\partial \sigma^2_B} = \sum_{i=1}^{m} \frac{\partial l}{\partial x_i} \cdot (x_i - \mu_B) \cdot \frac{-1}{2} \left( \frac{\sigma^2_B}{\sigma^2_B + \epsilon} + \epsilon \right)^{3/2}
\]

\[
\frac{\partial l}{\partial \mu_B} = \left( \sum_{i=1}^{m} \frac{\partial l}{\partial x_i} \cdot \frac{-1}{\sqrt{\sigma^2_B + \epsilon}} \right) + \frac{\partial l}{\partial \sigma^2_B} \cdot \frac{\sum_{i=1}^{m} (x_i - \mu_B)}{m}
\]

\[
\frac{\partial l}{\partial x_i} = \frac{\partial l}{\partial x_i} \cdot \frac{1}{\sqrt{\sigma^2_B + \epsilon}} + \frac{\partial l}{\partial \sigma^2_B} \cdot \frac{2(x_i - \mu_B)}{m} + \frac{\partial l}{\partial \mu_B} \cdot \frac{1}{m}
\]

\[
\frac{\partial l}{\partial \gamma} = \sum_{i=1}^{m} \frac{\partial l}{\partial y_i} \cdot \hat{x}_i
\]

\[
\frac{\partial l}{\partial \beta} = \sum_{i=1}^{m} \frac{\partial l}{\partial y_i}
\]

Gradients for the learnable parameters $\gamma$ and $\beta$. 
Multitask learning / auxiliary loss in GoogLeNet