Going Deeper With Convolution
Christian Szegedy¹, Wei Liu², Yangqing Jia¹, Pierre Sermanet¹, Scott Reed³, Dragomir Anguelov¹, Dumitru Erhan¹, Vincent Vanhoucke¹, Andrew Rabinovich¹

[2] University of North Carolina, Chapel Hill
[3] University of Michigan

Presentation for:
MA-721: Topics in Numerical Analysis: Deep Learning

Presented by:
Sidrah Liaqat – PhD Candidate
Department of Electrical and Computer Engineering, UK
Inception (v1)

- Winner of ILSVRC14 → ImageNet Large-Scale Visual Recognition Challenge 2014
  - Task: Classification and Detection
ILSVRC 2012 winner: AlexNet[9]

ILSVRC 2013 winner: ZFNet[21]

ILSVRC14 winner: Inception (v1)

- Uses batch normalization, image augmentation and RMSprop
- Based on very small convolutions (1x1 convolutions) to drastically reduce the number of parameters
- Architecture consists of 22 layer deep CNN but with 12x less parameters as AlexNet

GoogLeNet Network with all the bells and whistles
Motivation and high-level consideration

• Simplest way to improve a CNN is to increase its size
  • Increasing depth – number of layers of the network
  • Increasing width – number of units at each level

• This comes with some drawbacks
  • Bigger networks require more parameters which raises the risk of overfitting if the data sets are not large enough
  • If two convolutional layers are chained together then the number of filters results in a quadratic increase of computation
  • If many of the connections have weights near zero then this computation is wasted
Motivation and high-level consideration

• To avoid these issues we may introduce sparsity even inside the convolutions
  • However today's computers are inefficient while calculating non uniform sparse data structures
  • Non uniform sparse models require more sophisticated engineering and computing infrastructures

Is there a network architecture that makes use of filter level sparsity but better utilizes the strengths of our hardware?
Inception Architecture

- **Aurora et al**[2]
  “Their main result states that if the probability distribution of the data-set is representable by a large, very sparse deep neural network, then the optimal network topology can be constructed layer by layer by analyzing the correlation statistics of the activations of the last layer and clustering neurons with highly correlated outputs.”

- **Hebbian Principle:**
  - “Neurons that fire together, wire together”

- **Network in Network Architecture**[12]
  - 1x1 Convolutions

---

In images, correlations tend to be local
Less spread out correlations

number of filters

1x1
Cover more spread out clusters by 3x3 convolutions.
Cover more spread out clusters by 5x5 convolutions
Cover more spread out clusters by 5x5 convolutions
A heterogeneous set of convolutions

Schematic view (naive version)

number of filters

1x1
3x3
5x5

Filter concatenation

1x1 convolutions
3x3 convolutions
5x5 convolutions

Previous layer

Naive idea

- Filter concatenation
- 1x1 convolutions
- 3x3 convolutions
- 5x5 convolutions
- Previous layer
Inception Architecture

• The goal of the “Inception” architecture is to consider how an optimal local sparse structure of convolutional vision network can be approximated by dense components
  • Clusters of different sizes expected throughout the image, filtering using 1x1, 3x3 and 5x5 convolutions are used.
  • Combination of these filters concatenated into a single output vector forming the input for the next stage
  • Because pooling has proven essential in other applications a pooling path is added

Does not work!
Inception Architecture

- These are stacked on top of each other
  - As the network moves to higher levels you need more 3x3 and 5x5 convolutions because spatial concentration decreases
  - An issue with this strategy is that at the highest levels even a small number of 5x5 convolutions would be very computationally expensive because the outputs increase in number from stage to stage
  - Computational cost would explode within a few stages
Inception Architecture

• This leads to the idea of reducing the dimensions within the Inception architectures
• This is based on embeddings because low dimensional embeddings may contain a lot of information
• We use 1x1 convolutions to compute some reductions to the inputs before we do the expensive 3x3 and 5x5 convolutions
Inception Architecture

• This allows for the number of units in each stage to increase while not blowing up the computational cost in the latest stages
• Because the layers are now far more efficient this allows us to increase the size of the network without running into computational limits
• This results in networks that are 3 to 10 times faster than similarly performing networks not using the Inception Module
Inception Architecture

(a) Inception module, naïve version
(b) Inception module with dimension reductions

Figure 2: Inception module
GoogLeNet

- This is the network used in the ILSVRC 2014 competition
  - The receptive field is 224x224 in the RGB color space
  - The Network is 22 layers deep when counting layers with parameters
  - The overall number of layers is around 100
GoogLeNet

9 Inception modules
## GoogLeNet

<table>
<thead>
<tr>
<th>type</th>
<th>patch size/stride</th>
<th>output size</th>
<th>depth</th>
<th>#1×1</th>
<th>#3×3 reduce</th>
<th>#5×5 reduce</th>
<th>#5×5</th>
<th>pool proj</th>
<th>params</th>
<th>ops</th>
</tr>
</thead>
<tbody>
<tr>
<td>convolution</td>
<td>7×7/2</td>
<td>112×112×64</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2.7K</td>
<td>34M</td>
</tr>
<tr>
<td>max pool</td>
<td>3×3/2</td>
<td>56×56×64</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>112K</td>
<td>360M</td>
</tr>
<tr>
<td>convolution</td>
<td>3×3/1</td>
<td>56×56×192</td>
<td>2</td>
<td>64</td>
<td>192</td>
<td></td>
<td></td>
<td></td>
<td>380K</td>
<td>304M</td>
</tr>
<tr>
<td>max pool</td>
<td>3×3/2</td>
<td>28×28×192</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>364K</td>
<td>73M</td>
</tr>
<tr>
<td>inception (3a)</td>
<td>28×28×256</td>
<td>159K</td>
<td>128</td>
<td>64</td>
<td>96</td>
<td>128</td>
<td>32</td>
<td>32</td>
<td>437K</td>
<td>88M</td>
</tr>
<tr>
<td>inception (3b)</td>
<td>28×28×480</td>
<td>380K</td>
<td>128</td>
<td>128</td>
<td>192</td>
<td>32</td>
<td>96</td>
<td>64</td>
<td>159K</td>
<td>128M</td>
</tr>
<tr>
<td>max pool</td>
<td>3×3/2</td>
<td>14×14×480</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>580K</td>
<td>119M</td>
</tr>
<tr>
<td>inception (4a)</td>
<td>14×14×512</td>
<td>364K</td>
<td>192</td>
<td>96</td>
<td>208</td>
<td>16</td>
<td>48</td>
<td>64</td>
<td>840K</td>
<td>170M</td>
</tr>
<tr>
<td>inception (4b)</td>
<td>14×14×512</td>
<td>437K</td>
<td>160</td>
<td>112</td>
<td>224</td>
<td>24</td>
<td>64</td>
<td>64</td>
<td>1072K</td>
<td>54M</td>
</tr>
<tr>
<td>inception (4c)</td>
<td>14×14×512</td>
<td>463K</td>
<td>128</td>
<td>128</td>
<td>256</td>
<td>24</td>
<td>64</td>
<td>64</td>
<td>1388K</td>
<td>71M</td>
</tr>
<tr>
<td>inception (4d)</td>
<td>14×14×528</td>
<td>580K</td>
<td>112</td>
<td>144</td>
<td>288</td>
<td>32</td>
<td>64</td>
<td>64</td>
<td>159K</td>
<td>128M</td>
</tr>
<tr>
<td>inception (4e)</td>
<td>14×14×832</td>
<td>580K</td>
<td>256</td>
<td>160</td>
<td>320</td>
<td>32</td>
<td>128</td>
<td>128</td>
<td>1072K</td>
<td>54M</td>
</tr>
<tr>
<td>max pool</td>
<td>3×3/2</td>
<td>7×7×832</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>463K</td>
<td>100M</td>
</tr>
<tr>
<td>inception (5a)</td>
<td>7×7×832</td>
<td>1072K</td>
<td>256</td>
<td>160</td>
<td>320</td>
<td>32</td>
<td>128</td>
<td>128</td>
<td>1388K</td>
<td>71M</td>
</tr>
<tr>
<td>inception (5b)</td>
<td>7×7×1024</td>
<td>1388K</td>
<td>384</td>
<td>192</td>
<td>384</td>
<td>48</td>
<td>128</td>
<td>128</td>
<td>1072K</td>
<td>54M</td>
</tr>
<tr>
<td>avg pool</td>
<td>7×7/1</td>
<td>1×1×1024</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1000K</td>
<td>1M</td>
</tr>
<tr>
<td>dropout (40%)</td>
<td>1×1×1024</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1000K</td>
<td>1M</td>
</tr>
<tr>
<td>linear</td>
<td>1×1×1000</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1000K</td>
<td>1M</td>
</tr>
<tr>
<td>softmax</td>
<td>1×1×1000</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1000K</td>
<td>1M</td>
</tr>
</tbody>
</table>

Table 1: GoogLeNet incarnation of the Inception architecture
Training Methodology
(An honest description of how it really happened – and it was not straightforward)

• Asynchronous stochastic gradient descent with 0.9 momentum [17]
• Fixed learning rate schedule (decreasing the learning rate by 4% every 8 epochs)
• Polyak averaging [13] was used to create the final model used at inference time
• Sampling of various sized patches of the image whose size is distributed evenly between 8% and 100% of the image area and whose aspect ratio is chosen randomly between 3=4 and 4=3
• Photometric distortions by Andrew Howard [8] were useful to combat overfitting to some extent.
## ILSVRC 2014 Classification Challenge

**Setup and Results**

<table>
<thead>
<tr>
<th>Team</th>
<th>Year</th>
<th>Place</th>
<th>Error (top-5)</th>
<th>Uses external data</th>
</tr>
</thead>
<tbody>
<tr>
<td>SuperVision</td>
<td>2012</td>
<td>1st</td>
<td>16.4%</td>
<td>no</td>
</tr>
<tr>
<td>SuperVision</td>
<td>2012</td>
<td>1st</td>
<td>15.3%</td>
<td>Imagenet 22k</td>
</tr>
<tr>
<td>Clarifai</td>
<td>2013</td>
<td>1st</td>
<td>11.7%</td>
<td>no</td>
</tr>
<tr>
<td>Clarifai</td>
<td>2013</td>
<td>1st</td>
<td>11.2%</td>
<td>Imagenet 22k</td>
</tr>
<tr>
<td>MSRA</td>
<td>2014</td>
<td>3rd</td>
<td>7.35%</td>
<td>no</td>
</tr>
<tr>
<td>VGG</td>
<td>2014</td>
<td>2nd</td>
<td>7.32%</td>
<td>no</td>
</tr>
<tr>
<td>GoogLeNet</td>
<td>2014</td>
<td>1st</td>
<td>6.67%</td>
<td>no</td>
</tr>
</tbody>
</table>

Table 2: Classification performance
# ILSVRC 2014 Classification Challenge

## Setup and Results

<table>
<thead>
<tr>
<th>Number of models</th>
<th>Number of Crops</th>
<th>Cost</th>
<th>Top-5 error</th>
<th>compared to base</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>10.07%</td>
<td>base</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>10</td>
<td>9.15%</td>
<td>-0.92%</td>
</tr>
<tr>
<td>1</td>
<td>144</td>
<td>144</td>
<td>7.89%</td>
<td>-2.18%</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>7</td>
<td>8.09%</td>
<td>-1.98%</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>70</td>
<td>7.62%</td>
<td>-2.45%</td>
</tr>
<tr>
<td>7</td>
<td>144</td>
<td>1008</td>
<td>6.67%</td>
<td>-3.45%</td>
</tr>
</tbody>
</table>

Table 3: GoogLeNet classification performance break down
ILSVRC 2014 Detection Challenge Setup and Results

<table>
<thead>
<tr>
<th>Team</th>
<th>Year</th>
<th>Place</th>
<th>mAP</th>
<th>external data</th>
<th>ensemble</th>
<th>approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>UvA-Euvision</td>
<td>2013</td>
<td>1st</td>
<td>22.6%</td>
<td>none</td>
<td>?</td>
<td>Fisher vectors</td>
</tr>
<tr>
<td>Deep Insight</td>
<td>2014</td>
<td>3rd</td>
<td>40.5%</td>
<td>ImageNet 1k</td>
<td>3</td>
<td>CNN</td>
</tr>
<tr>
<td>CUHK DeepID-Net</td>
<td>2014</td>
<td>2nd</td>
<td>40.7%</td>
<td>ImageNet 1k</td>
<td>?</td>
<td>CNN</td>
</tr>
<tr>
<td>GoogLeNet</td>
<td>2014</td>
<td>1st</td>
<td>43.9%</td>
<td>ImageNet 1k</td>
<td>6</td>
<td>CNN</td>
</tr>
</tbody>
</table>

Table 4: Detection performance
# ILSVRC 2014 Detection Challenge Setup and Results

<table>
<thead>
<tr>
<th>Team</th>
<th>mAP</th>
<th>Contextual model</th>
<th>Bounding box regression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trims-Soushen</td>
<td>31.6%</td>
<td>no</td>
<td>?</td>
</tr>
<tr>
<td>Berkeley Vision</td>
<td>34.5%</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>UvA-Euvision</td>
<td>35.4%</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>CUHK DeepID-Net2</td>
<td>37.7%</td>
<td>no</td>
<td>?</td>
</tr>
<tr>
<td>GoogLeNet</td>
<td>38.02%</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Deep Insight</td>
<td>40.2%</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>

Table 5: Single model performance for detection
Conclusion

• Reducing with using 1x1 convolutions before passing it to 3x3 and 5x5 convolutions has proven efficient and effective.

• These results show that the Inception module approximating optimal sparse structure is a viable method for improving Neural Networks

• The advantage of this network is a significant improvement in quality with a modest increase in computational requirements