Survey of DNN Development Resources

CICS/MTL Tutorial (2017)

Website: http://eyeriss.mit.edu/tutorial.html

Joel Emer, Vivienne Sze, Yu-Hsin Chen
Popular DNNs

- LeNet (1998)
- AlexNet (2012)
- OverFeat (2013)
- VGGNet (2014)
- GoogleNet (2014)

ImageNet: Large Scale Visual Recognition Challenge (ILSVRC)

Accuracy (Top 5 error)

2012 2013 2014 2015 Human

O. Russakovsky et al., IJCV 2015
LeNet-5

CONV Layers: 2
Fully Connected Layers: 2
Weights: 60k
MACs: 341k
Sigmoid used for non-linearity

Digit Classification!

[Y. Lecun et al, Proceedings of the IEEE, 1998]
AlexNet

CONV Layers: 5
Fully Connected Layers: 3
Weights: 61M
MACs: 724M
ReLU used for non-linearity

ILSCVR12 Winner
Uses Local Response Normalization (LRN)

[Krizhevsky et al., NIPS, 2012]
# Large Sizes with Varying Shapes

## AlexNet Convolutional Layer Configurations

<table>
<thead>
<tr>
<th>Layer</th>
<th>Filter Size (RxS)</th>
<th># Filters (M)</th>
<th># Channels (C)</th>
<th>Stride</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>11x11</td>
<td>96</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>5x5</td>
<td>256</td>
<td>48</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3x3</td>
<td>384</td>
<td>256</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>3x3</td>
<td>384</td>
<td>192</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>3x3</td>
<td>256</td>
<td>192</td>
<td>1</td>
</tr>
</tbody>
</table>

![Diagram of AlexNet layers with parameters](image_url)

- **Layer 1**: 34k Params, 105M MACs
- **Layer 2**: 307k Params, 224M MACs
- **Layer 3**: 885k Params, 150M MACs

[Krizhevsky et al., NIPS, 2012]
VGG-16

CONV Layers: 13
Fully Connected Layers: 3
Weights: 138M
MACs: 15.5G

Also, 19 layer version
Reduce # of weights

More Layers $\rightarrow$ Deeper!

Image Source: http://www.cs.toronto.edu/~frossard/post/vgg16/

[Simonyan et al., arXiv 2014, ICLR 2015]
GoogLeNet (v1)

CONV Layers: 21 (depth), 57 (total)
Fully Connected Layers: 1
Weights: 7.0M
MACs: 1.43G

Also, v2, v3 and v4 ILSVRC14 Winner

parallel filters of different size has the effect of processing image at different scales

Inception Module

1x1 ‘bottleneck’ to reduce number of weights

[Szegedy et al., arXiv 2014, CVPR 2015]
GoogLeNet (v1)

CONV Layers: 21 (depth), 57 (total)
Fully Connected Layers: 1
Weights: 7.0M
MACs: 1.43G

Also, v2, v3 and v4
ILSVRC14 Winner

9 Inception Layers

3 CONV layers

1 FC layer

[Szegedy et al., arXiv 2014, CVPR 2015]
ResNet-50

CONV Layers: 49
Fully Connected Layers: 1
Weights: 25.5M
MACs: 3.9G

Also, 34, 152 and 1202 layer versions
ILSVRC15 Winner

Short Cut Module

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

Helps address the vanishing gradient challenge for training very deep networks

[He et al., arXiv 2015, CVPR 2016]
Revolution of Depth

# Summary of Popular DNNs

<table>
<thead>
<tr>
<th>Metrics</th>
<th>LeNet-5</th>
<th>AlexNet</th>
<th>VGG-16</th>
<th>GoogLeNet (v1)</th>
<th>ResNet-50</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top-5 error</td>
<td>n/a</td>
<td>16.4</td>
<td>7.4</td>
<td>6.7</td>
<td>5.3</td>
</tr>
<tr>
<td>Input Size</td>
<td>28x28</td>
<td>227x227</td>
<td>224x224</td>
<td>224x224</td>
<td>224x224</td>
</tr>
<tr>
<td># of CONV Layers</td>
<td>2</td>
<td>5</td>
<td>16</td>
<td>21 (depth)</td>
<td>49</td>
</tr>
<tr>
<td>Filter Sizes</td>
<td>5</td>
<td>3, 5, 11</td>
<td>3</td>
<td>1, 3, 5, 7</td>
<td>1, 3, 7</td>
</tr>
<tr>
<td># of Channels</td>
<td>1, 6</td>
<td>3 - 256</td>
<td>3 - 512</td>
<td>3 - 1024</td>
<td>3 - 2048</td>
</tr>
<tr>
<td># of Filters</td>
<td>6, 16</td>
<td>96 - 384</td>
<td>64 - 512</td>
<td>64 - 384</td>
<td>64 - 2048</td>
</tr>
<tr>
<td>Stride</td>
<td>1</td>
<td>1, 4</td>
<td>1</td>
<td>1, 2</td>
<td>1, 2</td>
</tr>
<tr>
<td># of Weights</td>
<td>2.6k</td>
<td>2.3M</td>
<td>14.7M</td>
<td>6.0M</td>
<td>23.5M</td>
</tr>
<tr>
<td># of MACs</td>
<td>283k</td>
<td>666M</td>
<td>15.3G</td>
<td>1.43G</td>
<td>3.86G</td>
</tr>
<tr>
<td># of FC layers</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td># of Weights</td>
<td>58k</td>
<td>58.6M</td>
<td>124M</td>
<td>1M</td>
<td>2M</td>
</tr>
<tr>
<td># of MACs</td>
<td>58k</td>
<td>58.6M</td>
<td>124M</td>
<td>1M</td>
<td>2M</td>
</tr>
<tr>
<td>Total Weights</td>
<td>60k</td>
<td>61M</td>
<td>138M</td>
<td>7M</td>
<td>25.5M</td>
</tr>
<tr>
<td>Total MACs</td>
<td>341k</td>
<td>724M</td>
<td>15.5G</td>
<td>1.43G</td>
<td>3.9G</td>
</tr>
</tbody>
</table>

**CONV Layers increasingly important!**
Summary of Popular DNNs

- **AlexNet**
  - First CNN Winner of ILSVRC
  - Uses LRN (deprecated after this)

- **VGG-16**
  - Goes Deeper (16+ layers)
  - Uses only 3x3 filters (stack for larger filters)

- **GoogLeNet (v1)**
  - Reduces weights with Inception and only one FC layer
  - Inception: 1x1 and DAG (parallel connections)
  - Batch Normalization

- **ResNet**
  - Goes Deeper (24+ layers)
  - Shortcut connections
Frameworks

Caffe
Berkeley / BVLC
(C, C++, Python, MATLAB)

TensorFlow
Google
(C++, Python)

Theano
U. Montreal
(Python)

Torch
Facebook / NYU
(C, C++, Lua)

Also, CNTK, MXNet, etc.
More at: https://developer.nvidia.com/deep-learning-frameworks
Example: Layers in Caffe

Convolution Layer
layer {
  name: "conv1"
  type: "Convolution"
  bottom: "data"
  top: "conv1"
... 
  convolution_param {
    num_output: 20
    kernel_size: 5
    stride: 1
  }
... 

Non-Linearity
layer {
  name: "relu1"
  type: "ReLU"
  bottom: "conv1"
  top: "conv1"
}

Pooling Layer
layer {
  name: "pool1"
  type: "Pooling"
  bottom: "conv1"
  top: "pool1"
  pooling_param {
    pool: MAX
    kernel_size: 2
    stride: 2 ...
  }

http://caffe.berkeleyvision.org/tutorial/layers.html
Benefits of Frameworks

• Rapid development
• Sharing models
• Workload profiling
• Network hardware co-design
Image Classification Datasets

- **Image Classification/Recognition**
  - Given an entire image → Select 1 of N classes
  - No localization (detection)

Datasets affect difficulty of task
MNIST

Digit Classification
28x28 pixels (B&W)
10 Classes
60,000 Training
10,000 Testing

LeNet in 1998
(0.95% error)

ICML 2013
(0.21% error)

http://yann.lecun.com/exdb/mnist/
Object Classification
~256x256 pixels (color)
1000 Classes
1.3M Training
100,000 Testing (50,000 Validation)

Image Source: http://karpathy.github.io/

http://www.image-net.org/challenges/LSVRC/
Fine grained Classes (120 breeds)

Top-5 Error
Winner 2012 (16.42% error)
Winner 2016 (2.99% error)

Image Source: http://www.image-net.org/challenges/LSVRC/

http://www.image-net.org/challenges/LSVRC/
## Image Classification Summary

<table>
<thead>
<tr>
<th></th>
<th>MNIST</th>
<th>IMAGENET</th>
</tr>
</thead>
<tbody>
<tr>
<td>Year</td>
<td>1998</td>
<td>2012</td>
</tr>
<tr>
<td>Resolution</td>
<td>28x28</td>
<td>256x256</td>
</tr>
<tr>
<td>Classes</td>
<td>10</td>
<td>1000</td>
</tr>
<tr>
<td>Training</td>
<td>60k</td>
<td>1.3M</td>
</tr>
<tr>
<td>Testing</td>
<td>10k</td>
<td>100k</td>
</tr>
<tr>
<td>Accuracy</td>
<td>0.21% error (ICML 2013)</td>
<td>2.99% top-5 error (2016 winner)</td>
</tr>
</tbody>
</table>

http://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results.html
Next Tasks: Localization and Detection

<table>
<thead>
<tr>
<th>Image classification</th>
<th>Single-object localization</th>
<th>Object detection</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Steel drum" /></td>
<td><img src="image" alt="Steel drum" /></td>
<td><img src="image" alt="Steel drum" /></td>
</tr>
<tr>
<td>Ground truth</td>
<td>Accuracy: 1</td>
<td>Accuracy: 1</td>
</tr>
<tr>
<td><img src="image" alt="Steel drum" /></td>
<td>Accuracy: 1</td>
<td>Accuracy: 0</td>
</tr>
<tr>
<td><img src="image" alt="Steel drum" /></td>
<td>Accuracy: 0</td>
<td>Accuracy: 0</td>
</tr>
</tbody>
</table>

[Russakovsky et al., IJCV, 2015]
Others Popular Datasets

- **Pascal VOC**
  - 11k images
  - Object Detection
  - 20 classes

- **MS COCO**
  - 300k images
  - Detection, Segmentation
  - Recognition in context

[Links to Pascal VOC and MS COCO datasets]
Recently Introduced Datasets

• Google Open Images (~9M images)
  – https://github.com/openimages/dataset
• Youtube-8M (8M videos)
  – https://research.google.com/youtube8m/
• AudioSet (2M sound clips)
  – https://research.google.com/audioset/index.html
Summary

- Development resources presented in this section enable us to evaluate hardware using the appropriate DNN model and dataset
  - Difficult tasks typically require larger models
  - Different datasets for different tasks
  - Number of datasets growing at a rapid pace