Deep Learning for Computer Vision with MATLAB

By Jon Cherrie
Deep learning is getting a lot of attention

"Dahl and his colleagues won $22,000 with a deep-learning system. 'We improved on Merck's baseline by about 15%.'"

- Nature 2014

"When Google adopted deep-learning-based speech recognition in its Android smartphone operating system, it achieved a 25% reduction in word errors."

- Nature 2014

"[Baidu's] system has achieved the best result to date, with a top-5 error rate of 4.58% and exceeding the human recognition performance."

- HPCWire 2015
Agenda

- What is deep learning
- Demo – object recognition
- Challenges with deep learning
- Why MATLAB?
What is deep learning?
Example Problem – Image Classification

Model

Tractor

Bicycle
Typical Computer Vision Model

LOAD DATA

PREPROCESSING

FEATURE EXTRACTION

SUPERVISED LEARNING

TRAINING

MODEL

Support Vector Machine
Deep neural network

- Some of these layers will be detecting “features”
- Other layers will do classification
- All the layers are trained together
Deep learning ≈ convolutional neural network

A convolutional neural network (ConvNet or CNN) is made up of different types of layers:

- Convolution
- Rectified linear unit (ReLU)
- Pooling
- Fully connected layers
Convolution

- A convolutional layer operates on a three-dimensional array, i.e., an image with red, green, and blue channels.
Convolutions tend to act as edge filters
ReLU

- Repeating linear layers does not add complexity to our network
- Thus we insert nonlinear layers

The current popular choice:

\[ x \mapsto \max(x, 0). \]

Apply this to each entry in the array.
Pooling

- Need to decrease the size of our layers, as we eventually want to arrive at $n$ neuron outputs
- Pooling is one way of doing this. It is essentially just downsampling

There are two main types of pooling layer: average and max. Both apply to 3D array inputs.
## Average Pooling

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

3 × 3 Average Pooling

<table>
<thead>
<tr>
<th>0.9</th>
<th>0.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3.6</td>
</tr>
</tbody>
</table>
Max Pooling

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

3 × 3 Max Pooling

\[
\begin{pmatrix}
2 & 1 \\
3 & 4
\end{pmatrix}
\]
Fully connected layers

- Flatten the output of the network into a column vector
- Treat each entry as a feature that the network has learned
- Feed this into a feed-forward neural network to classify the features
Layers

- Convolution
- ReLU
- Pooling
- Fully connected
- Softmax
- Local response normalization
- …
A deep network might be …
Demo – Object Recognition
Training

- Trained to perform classification on the ImageNet ILSVRC challenge data
  - 1.2 million images of varying size, cropped to 224x224
  - Each image falls into one of 1000 categories

- Training takes approximately a week
  - This demo doesn’t show training

- We will use a pre-trained network: vgg-f
Object Recognition using Deep Learning

<table>
<thead>
<tr>
<th>Training (using GPU)</th>
<th>Millions of images from 1000 different categories</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prediction</td>
<td>Real-time object recognition using a webcam connected to a laptop</td>
</tr>
</tbody>
</table>
Challenges with deep learning
Large number of parameters to find

<table>
<thead>
<tr>
<th>Layer</th>
<th>Details</th>
<th>Output Size</th>
<th>Number of Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>224x224x3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Conv 1</td>
<td>64 filters @ 11x11 Stride: 4; Pad: 0</td>
<td>54x54x64</td>
<td>64<em>11</em>11*3 = 23,232</td>
</tr>
<tr>
<td>LRN</td>
<td>54x54x64</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Max Pool</td>
<td>x2 downsample</td>
<td>27x27x64</td>
<td></td>
</tr>
<tr>
<td>Conv 2</td>
<td>256 @ 5x5 Stride 1; Pad 1</td>
<td>25x25x256</td>
<td>256<em>5</em>5*64 = 409,600</td>
</tr>
<tr>
<td>LRN</td>
<td>25x25x256</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Max Pool</td>
<td>x2 downsample</td>
<td>12x12x256</td>
<td></td>
</tr>
<tr>
<td>Conv 3</td>
<td>256 @ 3x3 Stride 1; Pad 1</td>
<td>12x12x256</td>
<td>256<em>3</em>3*256 = 589,824</td>
</tr>
<tr>
<td>Conv 4</td>
<td>256 @ 3x3 Stride 1; Pad 1</td>
<td>12x12x256</td>
<td>256<em>3</em>3*256 = 589,824</td>
</tr>
<tr>
<td>Conv 5</td>
<td>256 @ 3x3 Stride 1; Pad 1</td>
<td>12x12x256</td>
<td>256<em>3</em>3*256 = 589,824</td>
</tr>
<tr>
<td>Max Pool</td>
<td>x2 downsample</td>
<td>6x6x256</td>
<td></td>
</tr>
<tr>
<td>Full Connect 6</td>
<td>4096</td>
<td>4096x1</td>
<td>6<em>6</em>256*4096 = 37748736 (38 million)</td>
</tr>
<tr>
<td>Dropout</td>
<td></td>
<td>4096x1</td>
<td></td>
</tr>
<tr>
<td>Full connect 7</td>
<td>4096</td>
<td>4096x1</td>
<td>4096*4096 = 16777216 (16.8 million)</td>
</tr>
<tr>
<td>Dropout</td>
<td></td>
<td>4096x1</td>
<td></td>
</tr>
<tr>
<td>Full connect 8</td>
<td>1000</td>
<td>1000x1</td>
<td>4096*1000 = 4096000 (4 million)</td>
</tr>
<tr>
<td>Softmax</td>
<td></td>
<td>1000x1</td>
<td></td>
</tr>
<tr>
<td>TOTAL</td>
<td></td>
<td></td>
<td>61 million</td>
</tr>
</tbody>
</table>
Need many images in training set
Tools for pre- and post-processing

Also:
- removing average
- distortions, e.g., rotation & flips
- etc.
Iterative design
Why MATLAB for deep learning?
Why MATLAB for Deep Learning?

- Ability to work with signal, images, financial, geospatial etc. data
- Library of algorithms for image, signal and computer vision
- Built-in GPU support for functions such as image rotation, convolution, transformation and filtering
- Visualization
- Lots of community packages, e.g., MatConvNet, Caffe, deep learning toolbox in File Exchange
MatConvNet: CNNs for MATLAB

MatConvNet is a MATLAB toolbox implementing Convolutional Neural Networks (CNNs) for computer vision applications. It is simple, efficient, and can run and learn state-of-the-art CNNs. Several example CNNs are included to classify and encode images.

Citing. If you use MatConvNet in your work, please cite: "MatConvNet - Convolutional Neural Networks for MATLAB", A. Vedaldi and K. Lenc, Proc. of the ACM Int. Conf. on Multimedia, 2015. [BibTex]

New: 1.0-beta15 adds a few new layers to DagNN to support the Fully-Convolutional Networks (FCN) for image segmentation. Pretrained models are also available here. Batch normalization (\texttt{v1\_nnb\_norm}) has also been improved adding features that will make it easier to remove the layer after training a model.

New: 1.0-beta14 adds a new object-oriented network wrapper DagNN, supporting arbitrary network topologies. This release also adds GoogLeNet as a pre-trained model, new building blocks such as
Start with a pre-trained network

Pretrained models

This section describes how pre-trained models can be downloaded and used in MatConvNet.

Remark: The following CNN models may have been imported from other reference implementations and are equivalent to the originals up to numerical precision. However, note that:

1. Images need to be pre-processed (resized and cropped) before being submitted to a CNN for evaluation. Even small differences in the preprocessing details can have a non-negligible effect on the results.
2. The example below shows how to evaluate a CNN, but does not include data augmentation or encoding normalization as for example provided by the VGG code. While this is easy to implement, it is not done automatically here.
3. These models are provided here for convenience, but please credit the original authors.

Using the pretrained models

In order to run, say, `imagenet-vgg-s` on a test image, start from the example code included in the quickstart guide.
Managing image datasets

- imageSet (new in R2014b)
- Automated file-based workflow
  - Labelling
  - Partition
  - Reading
  - Indexing
- Integrated in Computer Vision workflows
Image Acquisition Toolbox

- Support for
  - industry standards, including DCAM, Camera Link, and GigE Vision
  - Common OS interfaces for webcams, including Direct Show, QuickTime, and video4linux2
  - A range of industrial and scientific hardware vendors
  - Microsoft Kinect

- Built-in MATLAB support for
  - Webcams
  - IP Cameras
How MATLAB addresses challenges

- Large sets of images that don't fit in memory: `imageSet`
- Image Processing and Computer Vision tools for pre- and post-processing
- Long running training: built-in GPU support for over 200 MATLAB functions, 45 Image Processing function, 90 Statistics and Machine Learning functions, etc.
- MATLAB offers flexible architecture for customized workflows
- Community toolboxes for ConvNets
FIN