Deep Convolutional Neural Networks for Image Classification

Slides from Lana Lazebnik
(Many slides from Rob Fergus, Andrej Karpathy)
Deep learning

• Learn a *feature hierarchy* all the way from pixels to classifier
• Each layer extracts features from the output of previous layer
• Train all layers jointly
Linear classifiers revisited

• When the data is linearly separable, there may be more than one separator (hyperplane)
Perceptron
Perceptron

Input

Weights

\[ x_1 \]
\[ x_2 \]
\[ x_3 \]
\[ \ldots \]
\[ x_D \]

\[ w_1 \]
\[ w_2 \]
\[ w_3 \]
\[ \ldots \]
\[ w_D \]

Output: \( \text{sgn}(w \cdot x + b) \)

Can incorporate bias as component of the weight vector by always including a feature with value set to 1.
Loose inspiration: Human neurons
NEW NAVY DEVICE LEARNS BY DOING

Psychologist Shows Embryo of Computer Designed to Read and Grow Wiser

WASHINGTON, July 7 (UPI) — The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.

The embryo — the Weather Bureau’s $2,000,000 “704” computer — learned to differentiate between right and left after fifty attempts in the Navy’s demonstration for newsmen.

The service said it would use this principle to build the first of its Perceptron thinking machines that will be able to read and write. It is expected to be finished in about a year at a cost of $100,000.

Dr. Frank Rosenblatt, designer of the Perceptron, conducted the demonstration. He said the machine would be the first device to think as the human brain. As do human beings, Perceptron will make mistakes at first, but will grow wiser as it gains experience, he said.

Dr. Rosenblatt, a research psychologist at the Cornell Aeronautical Laboratory, Buffalo, said Perceptrons might be fired to the planets as mechanical space explorers.

Without Human Controls

The Navy said the perceptron would be the first non-living mechanism “capable of receiving, recognizing and identifying its surroundings without any human training or control.”

The “brain” is designed to remember images and information it has perceived itself. Ordinary computers remember only what is fed into them on punch cards or magnetic tape.

Later Perceptrons will be able to recognize people and call out their names and instantly translate speech in one language to speech or writing in another language, it was predicted.

Mr. Rosenblatt said in principle it would be possible to build brains that could reproduce themselves on an assembly line and which would be conscious of their existence.

In today’s demonstration, the “704” was fed two cards, one with squares marked on the left side and the other with squares on the right side.

Learns by Doing

In the first fifty trials, the machine made no distinction between them. It then started registering a “Q” for the left squares and “O” for the right squares.

Dr. Rosenblatt said he could explain why the machine learned only in highly technical terms. But he said the machine had undergone a “self-induced change in the wiring diagram.”

The first Perceptron will have about 1,000 electronic “association cells” receiving electrical impulses from an eye-like scanning device with 400 photo-cells. The human brain has 10,000,000,000 responsive cells, including 100,000,000 connections with the eyes.
Perceptron update rule

- Initialize weights randomly
- Cycle through training examples in multiple passes (epochs)
- For each training instance \( \mathbf{x} \) with label \( y \):
  - Classify with current weights: \( y' = \text{sgn}(\mathbf{w} \cdot \mathbf{x}) \)
  - Update weights: \( \mathbf{w} \leftarrow \mathbf{w} + \alpha(y-y')\mathbf{x} \)
  - \( \alpha \) is a learning rate that should decay as \( 1/t \) (\( t \) is the epoch)
  - What happens if \( y' \) is correct?
  - Otherwise, consider what happens to individual weights \( w_i \leftarrow w_i + \alpha(y-y')x_i \)
    - If \( y = 1 \) and \( y' = -1 \), \( w_i \) will be increased if \( x_i \) is positive or decreased if \( x_i \) is negative \( \Rightarrow \mathbf{w} \cdot \mathbf{x} \) will get bigger
    - If \( y = -1 \) and \( y' = 1 \), \( w_i \) will be decreased if \( x_i \) is positive or increased if \( x_i \) is negative \( \Rightarrow \mathbf{w} \cdot \mathbf{x} \) will get smaller
Convergence of perceptron update rule

- **Linearly separable data:** converges to a perfect solution
- **Non-separable data:** converges to a minimum-error solution assuming learning rate decays as $O(1/t)$ and examples are presented in random sequence
Multi-Layer Neural Networks

• Network with a hidden layer:

• Can represent nonlinear functions (provided each perceptron has a nonlinearity)
Multi-Layer Neural Networks

Multi-Layer Neural Networks

• Beyond a single hidden layer:

Figure source: http://cs231n.github.io/neural-networks-1/
Training of multi-layer networks

- Find network weights to minimize the error between true and estimated labels of training examples:

\[
E(w) = \sum_{j=1}^{N} \left( y_j - f_w(x_j) \right)^2
\]

- Update weights by gradient descent:

\[
w \leftarrow w - \alpha \frac{\partial E}{\partial w}
\]
Training of multi-layer networks

• Find network weights to minimize the error between true and estimated labels of training examples:

$$E(w) = \sum_{j=1}^{N} (y_j - f_w(x_j))^2$$

• Update weights by **gradient descent**:  
  $$w \leftarrow w - \alpha \frac{\partial E}{\partial w}$$

• This requires perceptrons with a differentiable nonlinearity

Sigmoid:  
$$g(t) = \frac{1}{1 + e^{-t}}$$

Rectified linear unit (ReLU):  
$$g(t) = \max(0, t)$$
Training of multi-layer networks

• Find network weights to minimize the error between true and estimated labels of training examples:

\[ E(w) = \sum_{j=1}^{N} (y_j - f_w(x_j))^2 \]

• Update weights by gradient descent: \( w \leftarrow w - \alpha \frac{\partial E}{\partial w} \)

• Back-propagation: gradients are computed in the direction from output to input layers and combined using chain rule

• Stochastic gradient descent: compute the weight update w.r.t. one training example (or a small batch of examples) at a time, cycle through training examples in random order in multiple epochs
Multi-Layer Network Demo

http://playground.tensorflow.org/
Neural networks: Pros and cons

- **Pros**
  - Flexible and general function approximation framework
  - Can build extremely powerful models by adding more layers

- **Cons**
  - Hard to analyze theoretically (e.g., training is prone to local optima)
  - Huge amount of training data, computing power may be required to get good performance
  - The space of implementation choices is huge (network architectures, parameters)
Neural networks for images

- Image
- Convolutional layer
- Feature map
- Weight mask
Neural networks for images

image

convolutional layer
Convolution as feature extraction
Convolutional Neural Networks

- Neural network with specialized connectivity structure
- Stack multiple stages of feature extractors
- Higher stages compute more global, more invariant features
- Classification layer at the end

Biological inspiration

  
  Visual cortex consists of a hierarchy of *simple*, *complex*, and *hyper-complex* cells
Convolutional Neural Networks

- Input Image
- Convolution (Learned)
- Non-linearity
- Spatial pooling
- Normalization
- Feature maps
Convolutional Neural Networks

- Feature maps
- Normalization
- Spatial pooling
- Non-linearity
- Convolution (Learned)

Input Image → Feature Map
Convolutional Neural Networks

Diagram:
- Feature maps
- Normalization
- Spatial pooling
- Non-linearity
- Convolution (Learned)
- Input Image

Graph:
- Function $y = f(x)$
Convolutional Neural Networks

- Feature maps
- Normalization
- Spatial pooling
- Non-linearity
- Convolution (Learned)
- Input Image

Max
Convolutional Neural Networks

- Feature maps
- Normalization
- Spatial pooling
- Non-linearity
- Convolution (Learned)
- Input Image

Feature Maps
Feature Maps After Contrast Normalization
Convolutional Neural Networks

Convolutional filters are trained in a supervised manner by back-propagating classification error.
Simplified architecture

\[ P(c \mid x) = \frac{\exp(w_c \cdot x)}{\sum_{k=1}^{C} \exp(w_k \cdot x)} \]
Compare: SIFT Descriptor

Image Pixels → Apply oriented filters

Take max filter response (L-inf normalization)

Spatial pool (Sum), L2 normalization

Feature Vector

Lowe [IJCV 2004]
AlexNet

• Similar framework to LeCun’98 but:
  • Bigger model (7 hidden layers, 650,000 units, 60,000,000 params)
  • More data ($10^6$ vs. $10^3$ images)
  • GPU implementation (50x speedup over CPU)
    • Trained on two GPUs for a week

Refinement of AlexNet architecture

Figure 3. Architecture of our 8 layer convnet model. A 224 by 224 crop of an image (with 3 color planes) is presented as the input. This is convolved with 96 different 1st layer filters (red), each of size 7 by 7, using a stride of 2 in both x and y. The resulting feature maps are then: (i) passed through a rectified linear function (not shown), (ii) pooled (max within 3x3 regions, using stride 2) and (iii) contrast normalized across feature maps to give 96 different 55 by 55 element feature maps. Similar operations are repeated in layers 2,3,4,5. The last two layers are fully connected, taking features from the top convolutional layer as input in vector form (6*6*256 = 9216 dimensions). The final layer is a C-way softmax function, C being the number of classes. All filters and feature maps are square in shape.

Layer 1 Filters

M. Zeiler and R. Fergus, "Visualizing and Understanding Convolutional Networks," ECCV 2014 (Best Paper Award winner)
Layer 1: Top-9 Patches
Layer 2: Top-9 Patches

- Patches from validation images that give maximal activation of a given feature map
Layer 2: Top-9 Patches
Layer 3: Top-9 Patches
Layer 3: Top-9 Patches
Layer 4: Top-9 Patches
Layer 4: Top-9 Patches
Layer 5: Top-9 Patches
ImageNet Challenge

- ~14 million labeled images, 20k classes
- Images gathered from Internet
- Human labels via Amazon MTurk
- Challenge: 1.2 million training images, 1000 classes

www.image-net.org/challenges/LSVRC/
# ImageNet Challenge 2012-2014

<table>
<thead>
<tr>
<th>Team</th>
<th>Year</th>
<th>Place</th>
<th>Error (top-5)</th>
<th>External data</th>
</tr>
</thead>
<tbody>
<tr>
<td>SuperVision – Toronto</td>
<td>2012</td>
<td>-</td>
<td>16.4%</td>
<td>no</td>
</tr>
<tr>
<td>(7 layers)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SuperVision</td>
<td>2012</td>
<td>1st</td>
<td>15.3%</td>
<td>ImageNet 22k</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clarifai – NYU</td>
<td>2013</td>
<td>-</td>
<td>11.7%</td>
<td>no</td>
</tr>
<tr>
<td>(7 layers)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clarifai</td>
<td>2013</td>
<td>1st</td>
<td>11.2%</td>
<td>ImageNet 22k</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VGG – Oxford</td>
<td>2014</td>
<td>2nd</td>
<td>7.32%</td>
<td>no</td>
</tr>
<tr>
<td>(16 layers)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GoogLeNet</td>
<td>2014</td>
<td>1st</td>
<td>6.67%</td>
<td>no</td>
</tr>
<tr>
<td>(19 layers)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Human expert*</td>
<td></td>
<td></td>
<td>5.1%</td>
<td></td>
</tr>
</tbody>
</table>

http://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet/
THIS GUY BEAT GOOGLE’S SUPER-SMART AI—BUT IT WASN’T EASY

Andrej Karpathy
Stanford Computer Science Ph.D. student
karpathy_at_cs.stanford.edu
ImageNet Challenge 2015
Baidu admits cheating in international supercomputer competition

Baidu recently apologised for violating the rules of an international supercomputer test in May, when the Chinese search engine giant claimed to beat both Google and Microsoft on the ImageNet image-recognition test.

The New York Times

Computer Scientists Are Astir After Baidu Team Is Barred From A.I. Competition

Baidu caught gaming recent supercomputer performance test
Date: June 2, 2015

Dear ILSVRC community,

This is a follow up to the announcement on May 19, 2015 with some more details and the status of the test server.

During the period of November 28th, 2014 to May 18th, 2015, there were at least 36 accounts used by a team from Baidu to submit to the test server at least 200 times, far exceeding the specified limit of two submissions per week. This includes short periods of very high usage, for example with more than 40 submissions over 5 days from March 16th, 2015 to March 19th, 2015. Figure A below shows submissions from ImageNet accounts known to be associated with the team in question. Figure B shows a comparison to the activity from all other accounts.

The results obtained during this period are reported in a recent arXiv paper. Because of the violation of the regulations of the test server, these results may not be directly comparable to results obtained and reported by other teams. To make this clear, by exploiting the ability to test many slightly different solutions on the test server it is possible to 1) select the best out of a set of very similar solutions based on test performance and achieve a small but potentially significant advantage and 2) choose methods for further research and development based directly on the test data instead of using only the training and validation data for such choices.

<table>
<thead>
<tr>
<th>Team</th>
<th>Year</th>
<th>Place</th>
<th>Error (top-5)</th>
<th>External data</th>
</tr>
</thead>
<tbody>
<tr>
<td>SuperVision – Toronto</td>
<td>2012</td>
<td>-</td>
<td>16.4%</td>
<td>no</td>
</tr>
<tr>
<td>(7 layers)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SuperVision</td>
<td>2012</td>
<td>1st</td>
<td>15.3%</td>
<td>ImageNet 22k</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clarifai – NYU</td>
<td>2013</td>
<td>-</td>
<td>11.7%</td>
<td>no</td>
</tr>
<tr>
<td>(7 layers)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clarifai</td>
<td>2013</td>
<td>1st</td>
<td>11.2%</td>
<td>ImageNet 22k</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VGG – Oxford</td>
<td>2014</td>
<td>2nd</td>
<td>7.32%</td>
<td>no</td>
</tr>
<tr>
<td>(16 layers)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GoogLeNet</td>
<td>2014</td>
<td>1st</td>
<td>6.67%</td>
<td>no</td>
</tr>
<tr>
<td>(19 layers)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>??</td>
<td>2015</td>
<td>1st</td>
<td>??</td>
<td></td>
</tr>
</tbody>
</table>
Deep Residual Nets

Revolution of Depth

AlexNet, 8 layers (ILSVRC 2012)  VGG, 19 layers (ILSVRC 2014)  ResNet, 152 layers (ILSVRC 2015)

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, Deep Residual Learning for Image Recognition, arXiv 2015
Deep Residual Nets

Revolution of Depth

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, Deep Residual Learning for Image Recognition, arXiv 2015
Deep learning packages

- Caffe
- Torch
- Theano
- TensorFlow
- Matconvnet
- ...

http://deeplearning.net/software_links/
Breaking CNNs

Take a correctly classified image (left image in both columns), and add a tiny distortion (middle) to fool the ConvNet with the resulting image (right).

http://arxiv.org/abs/1312.6199
http://karpathy.github.io/2015/03/30/breaking-convnets/
Breaking CNNs

<table>
<thead>
<tr>
<th>centipede</th>
<th>peacock</th>
<th>jackfruit</th>
<th>bubble</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>king penguin</th>
<th>starfish</th>
<th>baseball</th>
<th>electric guitar</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>freight car</th>
<th>remote control</th>
<th>peacock</th>
<th>African grey</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

http://arxiv.org/abs/1412.1897
http://karpathy.github.io/2015/03/30/breaking-convnets/
What is going on?

• Recall gradient descent training: modify the weights to reduce classifier error

\[ \mathbf{w} \leftarrow \mathbf{w} - \alpha \frac{\partial E}{\partial \mathbf{w}} \]

• Adversarial examples: modify the image to increase classifier error

\[ \mathbf{x} \leftarrow \mathbf{x} + \alpha \frac{\partial E}{\partial \mathbf{x}} \]

http://arxiv.org/abs/1412.6572
http://karpathy.github.io/2015/03/30/breaking-convnets/
What is going on?

\[
x + 0.007 \times \frac{\partial E}{\partial x} = \alpha \frac{\partial E}{\partial x}
\]

“panda”
57.7% confidence

“nematode”
8.2% confidence

“gibbon”
99.3% confidence

http://arxiv.org/abs/1412.6572
http://karpathy.github.io/2015/03/30/breaking-convnets/
Fooling a linear classifier

• Perceptron weight update: add a small multiple of the example to the weight vector:

\[ w \leftarrow w + \alpha x \]

• To fool a linear classifier, add a small multiple of the weight vector to the training example:

\[ x \leftarrow x + \alpha w \]
Fooling a linear classifier

http://karpathy.github.io/2015/03/30/breaking-convnets/
Google DeepDream

- Modify the image to maximize activations of units in a given layer

https://github.com/google/deepdream/blob/master/dream.ipynb
More examples


http://deepdriving.cs.princeton.edu


http://www.visualqa.org