CS 1699: Intro to Computer Vision
Deep Learning

Prof. Adriana Kovashka
University of Pittsburgh
December 1, 2015
Today: Deep neural networks

- Background
- Architectures and basic operations
- Applications
- Visualizing deep neural networks
- “Breaking” deep learning
- Packages
Deep neural network

Figure from http://neuralnetworksanddeeplearning.com/chap5.html
Quick summary

• Take model trained on, e.g., ImageNet 2012 training set
• Take outputs of 6th or 7th layer
• Optional: fine-tune features and/or classifier on new dataset
• To train model from scratch, need lots of data
• Classify test set of new dataset

• Why now: We have lots of data, and deep nets can be trained in reasonable time with GPUs;
• Language: deep learning ~ deep neural nets ~ convolutional neural nets
Traditional Recognition Approach

- Features are key to recent progress in recognition, but flawed…
- Where next? Better classifiers? Or keep building more features?

Adapted from Lana Lazebnik
What about learning the features?

- Learn a *feature hierarchy* all the way from pixels to classifier.
- Each layer extracts features from the output of the previous layer.
- Train all layers jointly.

![Diagram showing layers of feature extraction](image)

*Image/Video Pixels* → *Layer 1* → *Layer 2* → *Layer 3* → *Simple Classifier*
“Shallow” vs. “deep” architectures

Traditional recognition: “Shallow” architecture

Image/Video Pixels \[\rightarrow\] Hand-designed feature extraction \[\rightarrow\] Trainable classifier \[\rightarrow\] Object Class

Deep learning: “Deep” architecture

Image/Video Pixels \[\rightarrow\] Layer 1 \[\rightarrow\] ... \[\rightarrow\] Layer N \[\rightarrow\] Simple classifier \[\rightarrow\] Object Class

Lana Lazebnik
**Background: Perceptrons**

![Diagram of a perceptron with inputs, weights, and a sigmoid function output.](image)

**Input**

- $x_1$
- $x_2$
- $x_3$
- $\ldots$
- $x_d$

**Weights**

- $w_1$
- $w_2$
- $w_3$
- $\ldots$
- $w_d$

**Output:** $\sigma(w \cdot x + b)$

**Sigmoid function:**

$$\sigma(t) = \frac{1}{1 + e^{-t}}$$

Lana Lazebnik
Background: Multi-Layer Neural Networks

- Nonlinear classifier
- **Training:** find network weights \( w \) to minimize the error between true training labels \( y_i \) and estimated labels \( f_w(x_i) \):

\[
E(w) = \sum_{i=1}^{N} (y_i - f_w(x_i))^2
\]

- Minimization can be done by gradient descent provided \( f \) is differentiable
  - This training method is called **back-propagation**
Deep neural network
Inspiration: Neuron cells

- Axon from another cell
- Axonal arborization
- Synapse
- Dendrite
- Nucleus
- Axon
- Cell body or Soma
- Synapses
Biological neuron and Perceptrons

A biological neuron

An artificial neuron (Perceptron) - a linear classifier

Jia-bin Huang
Hubel/Wiesel Architecture

  - Visual cortex consists of a hierarchy of *simple*, *complex*, and *hyper-complex* cells
  - **Hierarchy of feature detectors** in the visual cortex, with higher level features responding to patterns of activation in lower level cells, and propagating activation upwards to still higher level cells

Source

Lana Lazebnik, Jia-bin Huang
Hubel/Wiesel Architecture and Multi-layer Neural Network

Hubel and Weisel’s architecture

Multi-layer Neural Network
- A non-linear classifier

Jia-bin Huang
Today: Deep neural networks

• Background
• Architectures and basic operations
• Applications
• Visualizing deep neural networks
• “Breaking” deep learning
• Packages
Convolutional Neural Networks (CNN)

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


Adapted from Rob Fergus
Convolutional Neural Networks (CNN)

- **Feed-forward feature extraction:**
  1. Convolve input with learned filters
  2. Apply non-linearity
  3. Spatial pooling (downsample)
  4. Normalization (optional)
- **Supervised training of convolutional filters** by back-propagating classification error

Adapted from Lana Lazebnik
1. Convolution

- Apply learned filter weights
- One feature map per filter
- Stride can be greater than 1 (faster, less memory)

Adapted from Rob Fergus
2. Non-Linearity

- **Per-element (independent)**
- **Options:**
  - **Tanh**
  - **Sigmoid:** $1/(1+\exp(-x))$
  - **Rectified linear unit (ReLU)**
    - Simplifies backpropagation
    - Makes learning faster
    - Avoids saturation issues
    - Preferred option (works well)
3. Spatial Pooling

- Sum or max
- Non-overlapping / overlapping regions
- Role of pooling:
  - Invariance to small transformations
  - Larger receptive fields (see more of input)

Rob Fergus, figure from Andrej Karpathy
4. Normalization

- Within or across feature maps
- Before or after spatial pooling
Compare: SIFT Descriptor

Image Pixels → Apply oriented filters → Spatial pool (Sum) → Normalize to unit length → Feature Vector

Lowe [IJCV 2004]
A common CNN architecture (AlexNet)

Figure from http://www.mdpi.com/2072-4292/7/11/14680/htm
Training Convolutional Neural Networks

• Backpropagation + stochastic gradient descent
  – *Neural Networks: Tricks of the Trade*

• Initialization
  – Transfer learning

• Dropout

• Data augmentation

Adapted from Jia-bin Huang
Dropout

• Randomly turn off some neurons
• Allows individual neurons to independently be responsible for performance

Dropout: A simple way to prevent neural networks from overfitting [Srivastava JMLR 2014]

Adapted from Jia-bin Huang
Data Augmentation (Jittering)

• Create *virtual* training samples
  – Horizontal flip
  – Random crop
  – Color casting
  – Geometric distortion

Deep Image [Wu et al. 2015]
Today: Deep neural networks

- Background
- Architectures and basic operations
- Applications
- Visualizing deep neural networks
- “Breaking” deep learning
- Packages
Convnet Successes

- Handwritten text/digits
  - MNIST (0.17% error [Ciresan et al. 2011])
  - Arabic & Chinese  [Ciresan et al. 2012]

- Simpler recognition benchmarks
  - CIFAR-10 (9.3% error [Wan et al. 2013])
  - Traffic sign recognition
    - 0.56% error vs 1.16% for humans [Ciresan et al. 2011]

- But until recently, less good at more complex datasets
  - Caltech-101/256 (few training examples)
ImageNet Challenge 2012

~14 million labeled images, 20k classes

Images gathered from Internet

Human labels via Amazon Turk

Challenge: 1.2 million training images, 1000 classes

Deng et al. CVPR 2009

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
- Better regularization for training (DropOut)

ImageNet Challenge 2012

Krizhevsky et al. -- **16.4% error** (top-5)
Next best (non-convnet) – **26.2% error**
# ImageNet Challenge 2012-2014

<table>
<thead>
<tr>
<th>Team</th>
<th>Year</th>
<th>Place</th>
<th>Error (top-5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SuperVision – Toronto (7 layers)</td>
<td>2012</td>
<td>-</td>
<td>16.4%</td>
</tr>
<tr>
<td>Clarifai – NYU (7 layers)</td>
<td>2013</td>
<td>-</td>
<td>11.7%</td>
</tr>
<tr>
<td>VGG – Oxford (16 layers)</td>
<td>2014</td>
<td>2nd</td>
<td>7.32%</td>
</tr>
<tr>
<td>GoogLeNet (19 layers)</td>
<td>2014</td>
<td>1st</td>
<td>6.67%</td>
</tr>
<tr>
<td><strong>Human expert</strong> *</td>
<td></td>
<td></td>
<td><strong>5.1%</strong></td>
</tr>
</tbody>
</table>

Best non-convnet in 2012: 26.2%
Results on misc. benchmarks

### [1] Caltech-101 (30 samples per class)

<table>
<thead>
<tr>
<th>Method</th>
<th>DeCAF$_5$</th>
<th>DeCAF$_6$</th>
<th>DeCAF$_7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>LogReg</td>
<td>63.29 ± 6.6</td>
<td>84.30 ± 1.6</td>
<td>84.87 ± 0.6</td>
</tr>
<tr>
<td>LogReg with Dropout</td>
<td>-</td>
<td>86.08 ± 0.8</td>
<td>85.68 ± 0.6</td>
</tr>
<tr>
<td>SVM</td>
<td>77.12 ± 1.1</td>
<td>84.77 ± 1.2</td>
<td>83.24 ± 1.2</td>
</tr>
<tr>
<td>SVM with Dropout</td>
<td>-</td>
<td>86.91 ± 0.7</td>
<td>85.51 ± 0.9</td>
</tr>
</tbody>
</table>

Yang et al. (2009) 84.3
Jarrett et al. (2009) 65.5

### [1] SUN 397 dataset (DeCAF)

<table>
<thead>
<tr>
<th>Method</th>
<th>DeCAF$_6$</th>
<th>DeCAF$_7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>LogReg</td>
<td>40.94 ± 0.3</td>
<td>40.84 ± 0.3</td>
</tr>
<tr>
<td>SVM</td>
<td>39.36 ± 0.3</td>
<td>40.66 ± 0.3</td>
</tr>
</tbody>
</table>

Xiao et al. (2010) 38.0

### [1] Caltech-UCSD Birds (DeCAF)

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeCAF$_6$</td>
<td>58.75</td>
</tr>
<tr>
<td>DPD + DeCAF$_6$</td>
<td>64.96</td>
</tr>
<tr>
<td>DPD (Zhang et al., 2013)</td>
<td>50.98</td>
</tr>
<tr>
<td>POOF (Berg &amp; Belhumeur, 2013)</td>
<td>56.78</td>
</tr>
</tbody>
</table>

### [2] MIT-67 Indoor Scenes dataset (OverFeat)

<table>
<thead>
<tr>
<th>Method</th>
<th>mean Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROI + Gist</td>
<td>26.05</td>
</tr>
<tr>
<td>DPM</td>
<td>30.40</td>
</tr>
<tr>
<td>Object Bank</td>
<td>37.60</td>
</tr>
<tr>
<td>RBow</td>
<td>37.93</td>
</tr>
<tr>
<td>BoP</td>
<td>46.10</td>
</tr>
<tr>
<td>miSVM</td>
<td>46.40</td>
</tr>
<tr>
<td>D-Parts</td>
<td>51.40</td>
</tr>
<tr>
<td>IFV</td>
<td>60.77</td>
</tr>
<tr>
<td>MLrep</td>
<td>64.03</td>
</tr>
<tr>
<td>CNN-SVM</td>
<td>58.44</td>
</tr>
</tbody>
</table>


CNN features for detection

R-CNN: Regions with CNN features

Object detection system overview. Our system (1) takes an input image, (2) extracts around 2000 bottom-up region proposals, (3) computes features for each proposal using a large convolutional neural network (CNN), and then (4) classifies each region using class-specific linear SVMs. **R-CNN achieves a mean average precision (mAP) of 53.7% on PASCAL VOC 2010.** For comparison, Uijlings et al. (2013) report 35.1% mAP using the same region proposals, but with a spatial pyramid and bag-of-visual-words approach. The popular deformable part models perform at 33.4%.


Lana Lazebnik
Transfer Learning

- Improvement of learning in a **new** task through the **transfer of knowledge** from a **related** task that has already been learned

Learning and Transferring Mid-Level Image Representations using Convolutional Neural Networks [Oquab et al. CVPR 2014]
Beyond classification

- Detection
- Segmentation
- Regression
- Pose estimation
- Matching patches
- Synthesis

and many more...
Photographer identification

• Who took this photograph?

• Deep net features achieve 74% accuracy
  • Chance is less than 3%, human performance is 47%
• Method learns what proto-objects + scenes authors shoot
• Can use this to develop “field guides” for human use
• Can generate novel photographs by given author

Thomas and Kovashka, submitted to CVPR 2016
Photographer identification

- Some misclassifications
New photographs from dead photographers with convolutional neural networks

Martin Anderson Thu 12 Nov 2015 12.38pm

Researchers out of the University of Pittsburgh have used Convolutional Neural Networks (CNNs) to identify the strange obsessions and unique styles of 41 well-known photographers and even to generate new photographs which accord with their unique
Today: Deep neural networks

• Background
• Architectures and basic operations
• Applications

• Visualizing deep neural networks
• “Breaking” deep learning
• Packages
Layer 1

Visualizing and Understanding Convolutional Networks [Zeiler and Fergus, ECCV 2014]
Layer 2

- Patches from validation images that give maximal activation of a given feature map

Visualizing and Understanding Convolutional Networks [Zeiler and Fergus, ECCV 2014]
Layer 3

Visualizing and Understanding Convolutional Networks [Zeiler and Fergus, ECCV 2014]
Layer 3
Layer 4 and 5

Visualizing and Understanding Convolutional Networks [Zeiler and Fergus, ECCV 2014]
Evolution of Features During Training
Evolution of Features During Training
Diagnosing Problems

- Visualization of Krizhevsky et al.’s architecture showed some problems with layers 1 and 2
  - Large stride of 4 used
- Alter architecture: smaller stride & filter size
  - Visualizations look better
  - Performance improves
How important is depth?

Architecture of Krizhevsky et al.
8 layers total
Trained on ImageNet
18.1% top-5 error
How important is depth?

Remove top fully connected layer
  • Layer 7

Drop 16 million parameters

Only 1.1% drop in performance!
How important is depth?

Remove both fully connected layers
  • Layer 6 & 7

Drop ~50 million parameters

5.7% drop in performance
How important is depth?

Now try removing upper feature extractor layers:
- Layers 3 & 4

Drop ~1 million parameters

3.0% drop in performance
How important is depth?

Now try removing upper feature extractor layers & fully connected:

• Layers 3, 4, 6, 7

Now only 4 layers

33.5% drop in performance

→ Depth of network is key
Tapping off Features at each Layer

- Plug features from each layer into linear SVM
- Features are neuron activations at that level

<table>
<thead>
<tr>
<th></th>
<th>Cal-101 (30/class)</th>
<th>Cal-256 (60/class)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM (1)</td>
<td>44.8 ± 0.7</td>
<td>24.6 ± 0.4</td>
</tr>
<tr>
<td>SVM (2)</td>
<td>66.2 ± 0.5</td>
<td>39.6 ± 0.3</td>
</tr>
<tr>
<td>SVM (3)</td>
<td>72.3 ± 0.4</td>
<td>46.0 ± 0.3</td>
</tr>
<tr>
<td>SVM (4)</td>
<td>76.6 ± 0.4</td>
<td>51.3 ± 0.1</td>
</tr>
<tr>
<td>SVM (5)</td>
<td>86.2 ± 0.8</td>
<td>65.6 ± 0.3</td>
</tr>
<tr>
<td>SVM (7)</td>
<td>85.5 ± 0.4</td>
<td>71.7 ± 0.2</td>
</tr>
</tbody>
</table>

Adapted from Matt Zeiler
Breaking CNNs

Intriguing properties of neural networks [Szegedy ICLR 2014]

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).

Jia-bin Huang
Breaking CNNs

Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images [Nguyen et al. CVPR 2015]
Fooling a linear classifier

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

\[ x \leftarrow x + \alpha w \]
Today: Deep neural networks

• Background
• Architectures and basic operations
• Applications
• Visualizing deep neural networks
• “Breaking” deep learning
• Packages
Packages

- Caffe (new version of Decaf)
- cuda-convnet2
- Torch
- Theano
- MatConvNet
- Pylearn2
- Overfeat
Tutorials

- http://deeplearning.net/
- http://cs231n.stanford.edu
Things to remember

- **Overview**
  - Neuroscience, perceptron, multi-layer neural networks

- **Convolutional neural network (CNN)**
  - Convolution, nonlinearity, max pooling
  - CNN for classification and beyond

- **Understanding and visualizing CNN**
  - Find images that maximize some class scores; visualize individual neuron activation and input patterns; breaking CNNs

- **Training CNN**
  - Dropout; data augmentation; transfer learning

- **Using CNNs for your own task**
  - Basic first step: try the pre-trained CaffeNet fc6-fc8 layers as features

Adapted from Jia-bin Huang