Recognition: Overview
This book has a lot of material:

K. Grauman and B. Leibe

*Visual Object Recognition*

Synthesis Lectures On Computer Vision, 2011
How It All Began...

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
PROJECT MAC

Artificial Intelligence Group
Vision Memo. No. 100.

July 7, 1966

THE SUMMER VISION PROJECT
Seymour Papert

The summer vision project is an attempt to use our summer workers effectively in the construction of a significant part of a visual system. The particular task was chosen partly because it can be segmented into sub-problems which will allow individuals to work independently and yet participate in the construction of a system complex enough to be a real landmark in the development of "pattern recognition".
What are the recognition tasks that we need to solve in order to finish Papert’s summer vision project?

How did thousands of computer vision researchers kill time in order to not finish the project in 50 summers?
This Lecture

- What are the recognition tasks that we need to solve in order to finish Papert’s summer vision project?
- How did thousands of computer vision researchers kill time in order to not finish the project in 50 summers?
- What’s still missing?
What are the recognition tasks that we need to solve in order to finish Papert’s summer vision project?

How did thousands of computer vision researchers kill time in order to not finish the project in 50 summers?

What’s still missing?
This Lecture

- What are the recognition tasks that we need to solve in order to finish Papert’s summer vision project?
- How did thousands of computer vision researchers kill time in order to not finish the project in 50 summers?
- What’s still missing?
- What happens if we solve it?

**Figure:** Singularity?

What are the recognition tasks that we need to solve in order to finish Papert’s summer vision project?

How did thousands of computer vision researchers kill time in order to not finish the project in 50 summers?

What’s still missing?

What happens if we solve it?

Figure: Nah... Let’s start by having a more intelligent Roomba.

The Recognition Tasks

- Let’s take some typical tourist picture. What all do we want to recognize?
The Recognition Tasks

- Identification: we know this one (like our DVD recognition pipeline)
The Recognition Tasks

- Scene classification: what type of scene is the picture showing?

[Image: Outdoor/indoor, city/forest/factory/etc.]

[Adopted from S. Lazebnik]
The Recognition Tasks

- Classification: Is the object in the window a person, a car, etc

[Adopted from S. Lazebnik]
The Recognition Tasks

- Image Annotation: Which types of objects are present in the scene?

[Adopted from S. Lazebnik]
The Recognition Tasks

- Detection: Where are all objects of a particular class?

[Adopted from S. Lazebnik]
The Recognition Tasks

- Segmentation: Which pixels belong to each class of objects?
The Recognition Tasks

- Pose estimation: What is the pose of each object?
The Recognition Tasks

- Attribute recognition: Estimate attributes of the objects (color, size, etc)
The Recognition Tasks

- Commercialization: Suggest how to fix the attributes ;)

What should he wear to get a girlfriend?
The Recognition Tasks

- Action recognition: What is happening in the image?
The Recognition Tasks

- Surveillance: Why is something happening?
Before we proceed, let’s first give a shot to the techniques we already know

Let’s try object class detection

These techniques are:

- Template matching (remember Waldo in Lecture 3-5?)
- Large-scale retrieval: store millions of pictures, recognize new one by finding the most similar one in database. This is a Google approach.
Template Matching

- Template matching: normalized cross-correlation with a template (filter)
Template Matching

- Template matching: normalized cross-correlation with a template (filter)

[Slide from: A. Torralba]
Template matching: normalized cross-correlation with a template (filter)

A popular method is that of template matching, by point to point correlation of a model pattern with the image pattern. These techniques are inadequate for three-dimensional scene analysis for many reasons, such as occlusion, changes in viewing angle, and articulation of parts.” Nevatia & Binford, 1977.

[Slide from: A. Torralba]
Recognition via Retrieval by Similarity

- Upload a photo to Google image search and check if something reasonable comes out

![Google Images Query](image-url)
Recognition via Retrieval by Similarity

- Upload a photo to Google image search
- Pretty reasonable, both are Golden Gate Bridge
Recognition via Retrieval by Similarity

- Upload a photo to Google image search
- Let’s try a typical bathtub object
Recognition via Retrieval by Similarity

- Upload a photo to Google image search
- A bit less reasonable, but still some striking similarity

query
Recognition via Retrieval by Similarity

- Make a beautiful drawing and upload to Google image search
- Can you recognize this object?
Recognition via Retrieval by Similarity

- Make a beautiful drawing and upload to Google image search
- Not a very reasonable result

other retrieved results:
Why is it a Problem?

- Difficult scene conditions

[From: Grauman & Leibe]
Why is it a Problem?

- Huge within-class variations. Recognition is mainly about modeling variation.

[Pic from: S. Lazebnik]
Why is it a Problem?

- Tones of classes

~10,000 to 30,000
What if I tell you that you can do all these tasks with fantastic accuracy (enough to get a D+ in Papert’s class) with a single concept?

This concept is called **Neural Networks**
What if I tell you that you can do all these tasks with fantastic accuracy (enough to get a D+ in Papert’s class) with a single concept?

This concept is called Neural Networks

And it is quite simple.
What if I tell you that you can do all these tasks with fantastic accuracy (enough to get a D+ in Papert’s class) with a single concept?

This concept is called **Neural Networks**

And it is quite simple.
Inspiration: The Brain

- Many machine learning methods inspired by biology, eg the (human) brain
- Our brain has $\sim 10^{11}$ neurons, each of which communicates (is connected) to $\sim 10^4$ other neurons

Figure: The basic computational unit of the brain: Neuron

[Pic credit: http://cs231n.github.io/neural-networks-1/]
Mathematical Model of a Neuron

- Neural networks define functions of the inputs (*hidden features*), computed by neurons
- Artificial neurons are called *units*

**Figure:** A mathematical model of the neuron in a neural network

[Pic credit: http://cs231n.github.io/neural-networks-1/]
Activation Functions

Most commonly used activation functions:

- **Sigmoid**: \( \sigma(z) = \frac{1}{1+\exp(-z)} \)
- **Tanh**: \( \tanh(z) = \frac{\exp(z)-\exp(-z)}{\exp(z)+\exp(-z)} \)
- **ReLU (Rectified Linear Unit)**: \( \text{ReLU}(z) = \max(0, z) \)
Neuron in Python

- Example in Python of a neuron with a sigmoid activation function

```python
class Neuron(object):
    # ...
    def forward(inputs):
        """ assume inputs and weights are 1-D numpy arrays and bias is a number """
        cell_body_sum = np.sum(inputs * self.weights) + self.bias
        firing_rate = 1.0 / (1.0 + math.exp(-cell_body_sum))  # sigmoid activation function
        return firing_rate
```

**Figure**: Example code for computing the activation of a single neuron

[http://cs231n.github.io/neural-networks-1/]
Neural Network Architecture (Multi-Layer Perceptron)

- Network with one layer of four hidden units:

![Diagram of a neural network with one layer of four hidden units.]

**Figure:** Two different visualizations of a 2-layer neural network. In this example: 3 input units, 4 hidden units and 2 output units

- Each unit computes its value based on linear combination of values of units that point into it, and an activation function

[http://cs231n.github.io/neural-networks-1/]
Neural Network Architecture (Multi-Layer Perceptron)

- Network with one layer of four hidden units:

![Diagram of a neural network with one layer of four hidden units and output units.](image)

**Figure:** Two different visualizations of a 2-layer neural network. In this example: 3 input units, 4 hidden units and 2 output units

- Naming conventions; a 2-layer neural network:
  - One layer of hidden units
  - One output layer
  (we do not count the inputs as a layer)

[http://cs231n.github.io/neural-networks-1/]

Sanja Fidler
CSC420: Intro to Image Understanding
Neural Network Architecture (Multi-Layer Perceptron)

- Going deeper: a 3-layer neural network with two layers of hidden units

![A 3-layer neural net with 3 input units, 4 hidden units in the first and second hidden layer and 1 output unit](figure)

**Figure**: A 3-layer neural net with 3 input units, 4 hidden units in the first and second hidden layer and 1 output unit

- Naming conventions; a N-layer neural network:
  - \( N - 1 \) layers of hidden units
  - One output layer

[http://cs231n.github.io/neural-networks-1/]
Representational Power

- Neural network with at least one hidden layer is a universal approximator (can represent any function).
  
  Proof in: Approximation by Superpositions of Sigmoidal Function, Cybenko, paper

- The capacity of the network increases with more hidden units and more hidden layers
Representational Power

- Neural network with at least one hidden layer is a universal approximator (can represent any function).
  
  Proof in: Approximation by Superpositions of Sigmoidal Function, Cybenko, paper

- The capacity of the network increases with more hidden units and more hidden layers
  

[http://cs231n.github.io/neural-networks-1/]
Representational Power

- Neural network with at least one hidden layer is a universal approximator (can represent any function).
  
  Proof in: Approximation by Superpositions of Sigmoidal Function, Cybenko, paper

- The capacity of the network increases with more hidden units and more hidden layers


[http://cs231n.github.io/neural-networks-1/]
Neural Networks

- We only need to know two algorithms
  - Forward pass: performs inference
  - Backward pass: performs learning
Forward Pass: What does the Network Compute?

Output of the network can be written as:

\[ h_j(x) = f(v_{j0} + \sum_{i=1}^{D} x_i v_{ji}) \]

- Output of the network can be written as:

\[ h_j(x) = f(v_{j0} + \sum_{i=1}^{D} x_i v_{ji}) \]
Forward Pass: What does the Network Compute?

- Output of the network can be written as:

\[
\begin{align*}
    h_j(x) &= f(v_{j0} + \sum_{i=1}^{D} x_i v_{ji}) \\
    o_k(x) &= g(w_{k0} + \sum_{j=1}^{J} h_j(x) w_{kj})
\end{align*}
\]

(j indexing hidden units, k indexing the output units, D number of inputs)
Forward Pass: What does the Network Compute?

Output of the network can be written as:

\[ h_j(x) = f(v_{j0} + \sum_{i=1}^{D} x_i v_{ji}) \]

\[ o_k(x) = g(w_{k0} + \sum_{j=1}^{J} h_j(x) w_{kj}) \]

(j indexing hidden units, k indexing the output units, D number of inputs)
Example code for a forward pass for a 3-layer network in Python:

```python
# forward-pass of a 3-layer neural network:
f = lambda x: 1.0/(1.0 + np.exp(-x))  # activation function (use sigmoid)
x = np.random.randn(3, 1)  # random input vector of three numbers (3x1)
h1 = f(np.dot(W1, x) + b1)  # calculate first hidden layer activations (4x1)
h2 = f(np.dot(W2, h1) + b2)  # calculate second hidden layer activations (4x1)
out = np.dot(W3, h2) + b3  # output neuron (1x1)
```

- Can be implemented efficiently using matrix operations
- Example above: $W_1$ is matrix of size $4 \times 3$, $W_2$ is $4 \times 4$. What about biases and $W_3$?
Example code for a forward pass for a 3-layer network in Python:

```python
# forward-pass of a 3-layer neural network:
f = lambda x: 1.0/(1.0 + np.exp(-x)) # activation function (use sigmoid)
x = np.random.randn(3, 1) # random input vector of three numbers (3x1)
h1 = f(np.dot(W1, x) + b1) # calculate first hidden layer activations (4x1)
h2 = f(np.dot(W2, h1) + b2) # calculate second hidden layer activations (4x1)
out = np.dot(W3, h2) + b3 # output neuron (1x1)
```

- Can be implemented efficiently using matrix operations
- Example above: $W_1$ is matrix of size $4 \times 3$, $W_2$ is $4 \times 4$. What about biases and $W_3$?

[http://cs231n.github.io/neural-networks-1/]
Training Neural Networks

- Find weights:

\[
\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_{n=1}^{N} \text{loss}(\mathbf{o}(n), \mathbf{t}(n))
\]

where \( \mathbf{o} = f(\mathbf{x}; \mathbf{w}) \) is the output of a neural network, \( \mathbf{t} \) is ground-truth

- Define a loss function, eg:
  - Squared loss: \( \sum_k \frac{1}{2} (o_k^{(n)} - t_k^{(n)})^2 \)
  - Cross-entropy loss: \( -\sum_k t_k^{(n)} \log o_k^{(n)} \)
Training Neural Networks

- Find weights:

\[ w^* = \arg\min_w \sum_{n=1}^{N} \text{loss}(o^{(n)}, t^{(n)}) \]

where \( o = f(x; w) \) is the output of a neural network, \( t \) is ground-truth

- Define a loss function, eg:
  - Squared loss: \( \sum_k \frac{1}{2} (o_k^{(n)} - t_k^{(n)})^2 \)
  - Cross-entropy loss: \( -\sum_k t_k^{(n)} \log o_k^{(n)} \)

- Gradient descent:

\[ w^{t+1} = w^t - \eta \frac{\partial E}{\partial w^t} \]

where \( \eta \) is the learning rate (and \( E \) is error/loss)
Training Neural Networks

- Find weights:

\[ \mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_{n=1}^{N} \text{loss}(\mathbf{o}^{(n)}, \mathbf{t}^{(n)}) \]

where \( \mathbf{o} = f(\mathbf{x}; \mathbf{w}) \) is the output of a neural network, \( \mathbf{t} \) is ground-truth.

- Define a loss function, eg:
  - Squared loss: \( \sum_k \frac{1}{2}(o_k^{(n)} - t_k^{(n)})^2 \)
  - Cross-entropy loss: \( -\sum_k t_k^{(n)} \log o_k^{(n)} \)

- Gradient descent:

\[ \mathbf{w}^{t+1} = \mathbf{w}^t - \eta \frac{\partial E}{\partial \mathbf{w}^t} \]

where \( \eta \) is the learning rate (and \( E \) is error/loss).
Toy Code (Matlab): Neural Net Trainer

% F-PROP
for i = 1 : nr_layers - 1
    [h{i}  jac{i}]  =  nonlinearity(W{i} * h{i-1} +  b{i});
end
h{nr_layers-1}  =  W{nr_layers-1} * h{nr_layers-2}  +   b{nr_layers-1};
prediction  =  softmax(h{1-1});

% CROSS ENTROPY LOSS
loss  =  -  sum(sum(log(prediction)  .*  target)) / batch_size;

% B-PROP
dh{l-1}  =  prediction  -  target;
for i = nr_layers – 1 : -1 : 1
    Wgrad{i}  =  dh{i} * h{i-1}';
    bgrad{i}  =  sum(dh{i}, 2);
    dh{i-1}  =  (W{i}' * dh{i})  .*  jac{i-1};
end

% UPDATE
for i = 1 : nr_layers - 1
    W{i}  =  W{i}  –  (lr / batch_size)  *  Wgrad{i};
    b{i}  =  b{i}  –  (lr / batch_size)  *  bgrad{i};
end

This code has a few bugs with indices...
To work with images we typically use Neural Networks with special architecture.
Convolutional Neural Networks (CNN)

- Remember our Lecture 2 about filtering?
Convolutional Neural Networks (CNN)

- If our filter was $[-1, 1]$, we got a vertical edge detector.
Now imagine we didn’t only want a vertical edge detector, but also a horizontal one, and one for corners, one for dots, etc. We would need to take many filters. A **filterbank**.

![Filterbank Diagram](Image)

- Input “image”
- Filter bank
- 3 channels (R,G,B)
- Output map
- Output has many “channels”, one for each filter

[Pic adopted from: A. Krizhevsky]
Convolutional Neural Networks (CNN)

- Applying a filterbank to an image yields a cube-like output, a 3D matrix in which each slice is an output of convolution with one filter, and an activation function.

In this example our network will always expect a 224x224x3 image.

[Pic adopted from: A. Krizhevsky]
Convolutional Neural Networks (CNN)

- Applying a filterbank to an image yields a cube-like output, a 3D matrix in which each slice is an output of convolution with one filter, and an activation function.

![Diagram showing convolution process and filter sizes](image)
Convolutional Neural Networks (CNN)

- Do some additional tricks. A popular one is called **max pooling**. Any idea why you would do this?

![Diagram showing max pooling in a convolutional neural network](Pic adopted from: A. Krizhevsky)
Convolutional Neural Networks (CNN)

- Do some additional tricks. A popular one is called **max pooling**. Any idea why you would do this? To get **invariance to small shifts in position**.

![Max pooling diagram](image-url)

\[
O(i, j) = \max_{k \in \{i-1, i, i+1\}, l \in \{j-1, j, j+1\}} O(k, l)
\]

Take each slice in the output cube, and in each pixel compute a max over a small patch around it. This is called **max pooling**.

[Pic adopted from: A. Krizhevsky]
Convolutional Neural Networks (CNN)

- Now add another “layer” of filters. For each filter again do convolution, but this time with the output cube of the previous layer.

![Diagram of CNN layers](image-url)

[Pic adopted from: A. Krizhevsky]
Convolutional Neural Networks (CNN)

- Keep adding a few layers. Any idea what’s the purpose of more layers? Why can’t we just have a full bunch of filters in one layer?

Do it recursively
Have multiple “layers”

[Pic adopted from: A. Krizhevsky]
Convolutional Neural Networks (CNN)

- In the end add one or two **fully** (or **densely** connected layers. In this layer, we don't do convolution we just do a dot-product between the “filter” and the output of the previous layer.

![Diagram of convolutional neural network](image-url)

In the top, most networks add a “densely” connected layer. You can think of this as a filter, and the output value is a dot product between the filter and the output cube of the previous layer.

What are the dimensions of this filter in this example? How many such filters are on this layer?
Add one final layer: a **classification** layer. Each dimension of this vector tells us the probability of the input image being of a certain class.
Convolutional Neural Networks (CNN)

- This fully specifies a network. The one below has been a popular choice in the fast few years. It was proposed by UofT guys: A. Krizhevsky, I. Sutskever, G. E. Hinton, *ImageNet Classification with Deep Convolutional Neural Networks*, NIPS 2012. This network won the Imagenet Challenge of 2012, and revolutionized computer vision.

- How many parameters (weights) does this network have?
Convolutional Neural Networks (CNN)

- Trained with stochastic gradient descent on two NVIDIA GPUs for about a week
- 650,000 neurons
- 60,000,000 parameters
- 630,000,000 connections
- Final feature layer: 4096-dimensional

**Figure:** From: http://www.image-net.org/challenges/LSVRC/2012/supervision.pdf

[Pic adopted from: A. Krizhevsky]
The trick is to not hand-fix the weights, but to \textbf{train} them. Train them such that when the network sees a picture of a dog, the last layer will say “dog”.

[Pic adopted from: A. Krizhevsky]
Convolutional Neural Networks (CNN)

- Or when the network sees a picture of a cat, the last layer will say “cat”.

[Pic adopted from: A. Krizhevsky]
Convolutional Neural Networks (CNN)

- Or when the network sees a picture of a boat, the last layer will say “boat”... The more pictures the network sees, the better.

Train on lots of examples. Millions. Tens of millions. Wait a week for training to finish.
Share your network (the weights) with others who are not fortunate enough with GPU power.

[Pic adopted from: A. Krizhevsky]
Once trained we can do classification. Just feed in an image or a crop of the image, run through the network, and read out the class with the highest probability in the last (classification) layer.
Example
Classification Performance

- Imagenet, main challenge for object classification: http://image-net.org/
- 1000 classes, 1.2M training images, 150K for test
A. Krizhevsky, I. Sutskever, and G. E. Hinton rock the Imagenet Challenge

<table>
<thead>
<tr>
<th>Team name</th>
<th>Filename</th>
<th>Error (5 guesses)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SuperVision</td>
<td>test-preds-131-137-145-135-145f.txt</td>
<td>0.16422</td>
<td>Using only supplied training data</td>
</tr>
<tr>
<td>ISI</td>
<td>pred_FVs_wLACs_weighted.txt</td>
<td>0.26172</td>
<td>Weighted sum of scores from each classifier with SIFT+FV, LBP+FV, GiST+FV, and CSIFT+FV, respectively.</td>
</tr>
<tr>
<td>ISI</td>
<td>pred_FVs_weighted.txt</td>
<td>0.26602</td>
<td>Weighted sum of scores from classifiers using each FV.</td>
</tr>
<tr>
<td>ISI</td>
<td>pred_FVs_summed.txt</td>
<td>0.26646</td>
<td>Naive sum of scores from classifiers using each FV.</td>
</tr>
<tr>
<td>ISI</td>
<td>pred_FVs_wLACs_summed.txt</td>
<td>0.26952</td>
<td>Naive sum of scores from each classifier with SIFT+FV, LBP+FV, GiST+FV, and CSIFT+FV, respectively.</td>
</tr>
</tbody>
</table>
Neural Networks as Descriptors

- What vision people like to do is take the already trained network (avoid one week of training), and remove the last classification layer. Then take the top remaining layer (the 4096 dimensional vector here) and use it as a descriptor (feature vector).

Vision people are mainly interested in this vector. You can use it as a descriptor. A much better descriptor than SIFT, etc. Train your own classifier on top for your choice of classes.
Neural Networks as Descriptors

- What vision people like to do is take the already trained network, and remove the last classification layer. Then take the top remaining layer (the 4096 dimensional vector here) and use it as a descriptor (feature vector).
- Now train your own classifier on top of these features for arbitrary classes.
Neural Networks as Descriptors

- What vision people like to do is take the already trained network, and remove the last classification layer. Then take the top remaining layer (the 4096 dimensional vector here) and use it as a descriptor (feature vector).
- Now train your own classifier on top of these features for arbitrary classes.
- This is quite hacky, but works miraculously well.

![Image of CNN magic]

Classifier predicting my set of classes
Neural Networks as Descriptors

- What vision people like to do is take the already trained network, and remove the last classification layer. Then take the top remaining layer (the 4096 dimensional vector here) and use it as a descriptor (feature vector).
- Now train your own classifier on top of these features for arbitrary classes.
- This is quite hacky, but works miraculously well.
- Everywhere where we were using SIFT (or anything else), you can use NNs.
And Detection?

- For classification we feed in the full image to the network. But how can we perform detection?

Find all objects of interest in this image!
And Detection?

- Generate lots of proposal bounding boxes (rectangles in image where we think any object could be)
- Each of these boxes is obtained by grouping similar clusters of pixels

**Figure:** R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation, CVPR'14
And Detection?

- Generate lots of proposal bounding boxes (rectangles in image where we think any object could be)
- Each of these boxes is obtained by grouping similar clusters of pixels
- Crop image out of each box, warp to fixed size \((224 \times 224)\) and run through the network

**Figure:** R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation, CVPR’14
And Detection?

- Generate lots of proposal bounding boxes (rectangles in image where we think any object could be)
- Each of these boxes is obtained by grouping similar clusters of pixels
- Crop image out of each box, warp to fixed size ($224 \times 224$) and run through the network.
- If the warped image looks weird and doesn’t resemble the original object, don’t worry. Somehow the method still works.
- This approach, called R-CNN, was proposed in 2014 by Girshick et al.

**Figure:** R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation, CVPR’14
One way of getting the proposal boxes is by hierarchical merging of regions. This particular approach, called Selective Search, was proposed in 2011 by Uijlings et al. We will talk more about this later in class.

**Figure:** Bottom: J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, A. W. M. Smeulders, Selective Search for Object Recognition, IJCV 2013
One way of getting the proposal boxes is by hierarchical merging of regions. This particular approach, called Selective Search, was proposed in 2011 by Uijlings et al. We will talk more about this later in class.

**Figure:** Bottom: J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, A. W. M. Smeulders, Selective Search for Object Recognition, IJCV 2013
Detection Datasets

- **PASCAL VOC challenge**: [http://pascallin.ecs.soton.ac.uk/challenges/VOC/](http://pascallin.ecs.soton.ac.uk/challenges/VOC/).

Figure: PASCAL has 20 object classes, 10K images for training, 10K for test.
Detection Performance in 2013: 40.4%

In 2013, no networks:

- Results on the main recognition benchmark, the **PASCAL VOC** challenge.

Figure: Leading method segDPM is by Sanja et al. Those were the good times...

S. Fidler, R. Mottaghi, A. Yuille, R. Urtasun, Bottom-up Segmentation for Top-down Detection, CVPR’13
In 2014, networks:

- Results on the main recognition benchmark, the PASCAL VOC challenge.

**Figure:** Leading method R-CNN is by Girshick et al.

So Neural Networks are Great

- So networks turn out to be great.
- At this point Google, Facebook, Microsoft, Baidu “steal” most neural network professors from academia.
So Neural Networks are Great

- But to train the networks you need quite a bit of computational power. So what do you do?
So Neural Networks are Great

- Buy even more.
So Neural Networks are Great

- And train **more layers**. 16 instead of 7 before. 144 million parameters.

**Figure:** K. Simonyan, A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv 2014

[Pic adopted from: A. Krizhevsky]
150 Layers!

- Networks are now at 150 layers
- They use a skip connections with special form
- In fact, they don’t fit on this screen
- Amazing performance!
- A lot of “mistakes” are due to wrong ground-truth

Results: Object Classification

Sanja Fidler

Results: Object Detection

Results: Object Detection

Results: Object Detection
Results: Object Detection

What do CNNs Learn?

Figure: Filters in the first convolutional layer of Krizhevsky et al
What do CNNs Learn?

![Filters in the second layer](http://arxiv.org/pdf/1311.2901v3.pdf)

**Figure:** Filters in the second layer
What do CNNs Learn?

Figure: Filters in the third layer

What do CNNs Learn?

What do CNNs Learn?

Neural Networks – Can Do Anything

- Classification / annotation
- Detection
- Segmentation
- Stereo
- Optical flow

How would you use them for these tasks?
Neural Networks – Years In The Making

- NNs have been around for 50 years. Inspired by processing in the brain.

**Figure:** Fukushima, Neocognitron. Biol. Cybernetics, 1980

**Figure:** http://www.nature.com/nrn/journal/v14/n5/figs/recognition/nrn3476-f1.jpg
V1: selective to direction of movement (Hubel & Wiesel)

Figure: Pic from:
http://www.cns.nyu.edu/~david/courses/perception/lecturenotes/V1/LGN-V1-slides/Slide15.jpg

Hubel & Wiesel, 1968
Neuroscience

- V2: selective to combinations of orientations

Figure: G. M. Boynton and Jay Hegde, Visual Cortex: The Continuing Puzzle of Area V2, Current Biology, 2004
V4: selective to more complex local shape properties (convexity/concavity, curvature, etc)

**Figure:** A. Pasupathy, C. E. Connor, Shape Representation in Area V4: Position-Specific Tuning for Boundary Conformation, Journal of Neurophysiology, 2001
Neuroscience

- IT: Seems to be category selective

Figure: N. Kriegeskorte, M. Mur, D. A. Ruff, R. Kiani, J. Bodurka, H. Esteky, K. Tanaka, P. A. Bandettini, Matching Categorical Object Representations in Inferior Temporal Cortex of Man and Monkey, Neuron, 2008
Neuroscience

- Grandmother / Jennifer Aniston cell?

**Figure:** R. Q. Quiroga, L. Reddy, G. Kreiman, C. Koch, I. Fried, Invariant visual representation by single-neurons in the human brain. Nature, 2005
GRANDMOTHER CELLS REVISITED
ARE NERVE CELLS SUCH AS THE JENNIFER ANISTON NEURON THE LONG-DEBATED GRANDMOTHER CELLS? TO ANSWER THAT QUESTION, WE HAVE TO BE MORE PRECISE ABOUT WHAT WE MEAN BY GRANDMOTHER CELLS. ONE EXTREME WAY OF THINKING ABOUT THE GRANDMOTHER CELL HYPOTHESIS IS THAT ONLY ONE NEURON RESPONDS TO ONE CONCEPT. BUT IF WE COULD FIND ONE SINGLE NEURON THAT FIRED TO JENNIFER ANISTON, IT STRONGLY SUGGESTS THAT THERE MUST BE MORE—THE CHANCE OF FINDING THE ONE AND ONLY ONE AMONG BILLIONS IS MINISCULE. MOREOVER, IF ONLY A SINGLE NEURON WOULD BE RESPONSIBLE FOR A PERSON’S ENTIRE CONCEPT OF JENNIFER ANISTON, AND IT WERE DAMAGED OR DESTROYED BY DISEASE OR ACCIDENT, ALL TRACE OF JENNIFER ANISTON WOULD DISAPPEAR FROM MEMORY, AN EXTREMELY UNLIKELY PROSPECT.

A LESS EXTREME DEFINITION OF GRANDMOTHER CELLS POSTULATES THAT MANY MORE THAN A SOLITARY NEURON RESPOND TO ANY ONE CONCEPT. THIS HYPOTHESIS IS PLAUSIBLE BUT VERY DIFFICULT, IF NOT IMPOSSIBLE, TO PROVE. WE CANNOT TRY EVERY POSSIBLE CONCEPT TO PROVE THAT THE NEURON FIRES ONLY TO JENNIFER ANISTON. IN FACT, THE OPPOSITE IS OFTEN THE CASE: WE OFTEN FIND NEURONS THAT RESPOND TO MORE THAN ONE CONCEPT. Thus, if a neuron fires only to one person during an experiment, we cannot rule out that it could have also fired to some other stimuli that we did not happen to show.

A single neuron that responded to Luke Skywalker and his written and spoken name also fired to the image of Yoda.
Neuroscience

- Take the whole brain processing business with a grain of salt. Even neuroscientists don’t fully agree. Think about computational models.
Neural Networks – Why Do They Work?

- NNs have been around for 50 years, and they haven’t changed much.
- So why do they work now?

Figure: Fukushima, Neocognitron. Biol. Cybernetics, 1980
Neural Networks – Why Do They Work?

- NNs have been around for 50 years, and they haven’t changed much.
- So why do they work now?

Figure: Fukushima, Neocognitron. Biol. Cybernetics, 1980
Neural Networks – Why Do They Work?

- Some cool tricks in design and training:

- Computational resources and tones of data

- NNs can **train millions** of parameters from tens of **millions of examples**

**Figure:** The Imagenet dataset: Deng et al. 14 million images, 1000 classes
Main code:

- Neural network packages:

- Object detection:
  - https://github.com/rbgirshick/rcnn
  - https://github.com/weiliu89/caffe/tree/ssd
Important tasks for visual recognition: classification (given an image crop, decide which object class or scene it belongs to), detection (where are all the objects for some class in the image?), segmentation (label each pixel in the image with a semantic label), pose estimation (which 3D view or pose the object is in with respect to camera?), action recognition (what is happening in the image/video)

Bottom-up grouping is important to find only a few rectangles in the image which contain objects of interest. This is much more efficient than exploring all possible rectangles.

Neural Networks are currently the best feature extractor in computer vision.

Mainly because they have multiple layers of nonlinear classifiers, and because they can train from millions of examples efficiently.

Going forward design computationally less intense solutions with higher generalization power that will beat 100 layers that Google can afford to do.