Recognition: Overview

Textbook

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 — July 7, 1966
Vision Memo. No. 100.

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".

[Slide credit: A. Torralba]

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?

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?

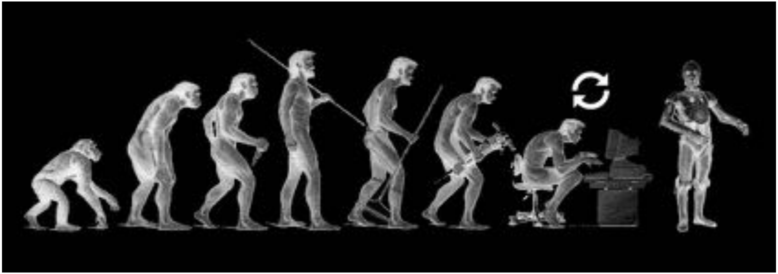
This Lecture

- 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?

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?



http://www.futurebuff.com/wp-content/uploads/2014/06/singularity-c3po.jpg

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

http://realitypod.com/wp-content/uploads/2013/08/Wall-E.jpg

The Recognition Tasks

- Let's take some typical tourist picture. What all do we want to recognize?

[Adopted from S. Lazebnik]

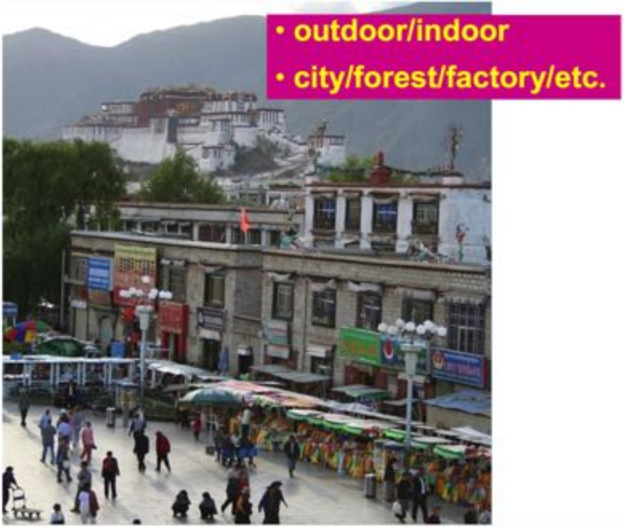
The Recognition Tasks

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

[Adopted from S. Lazebnik]

The Recognition Tasks

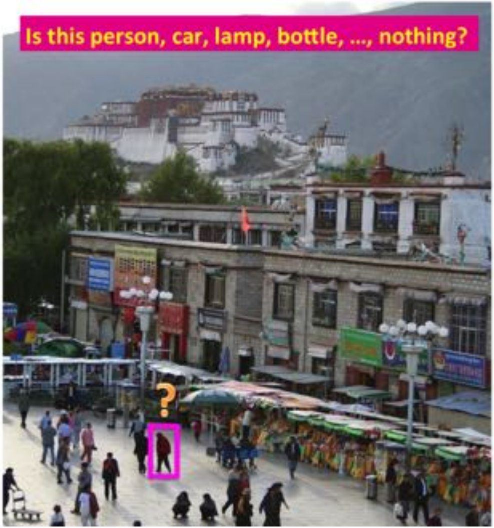
- Scene classification: what type of scene is the picture showing?



[Adopted from S. Lazebnik]

The Recognition Tasks

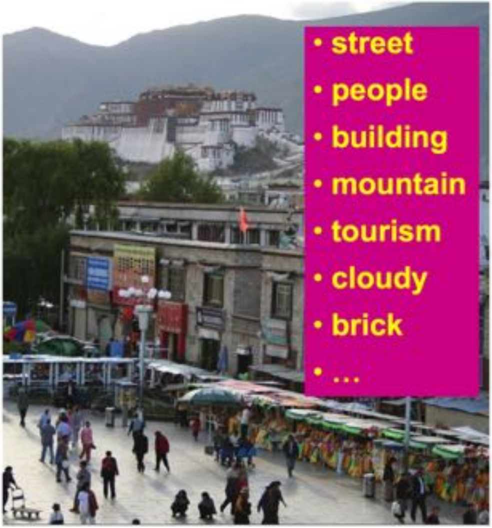
- Classification: Is the object in the window a person, a car, etc



[Adopted from S. Lazebnik]

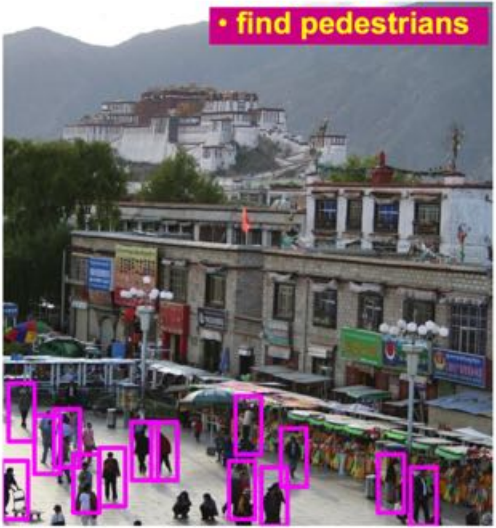
The Recognition Tasks

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



[Adopted from S. Lazebnik]

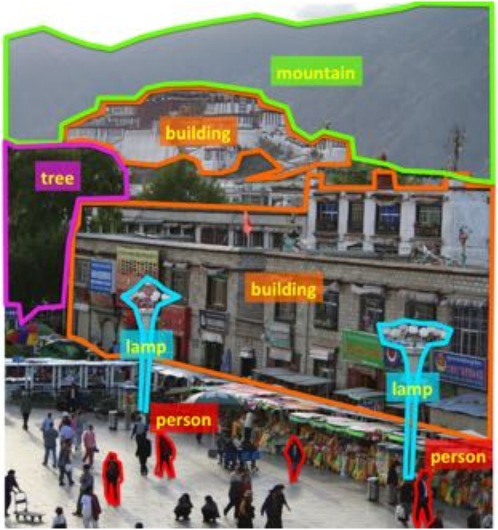
- Detection: Where are all objects of a particular class?



[Adopted from S. Lazebnik]

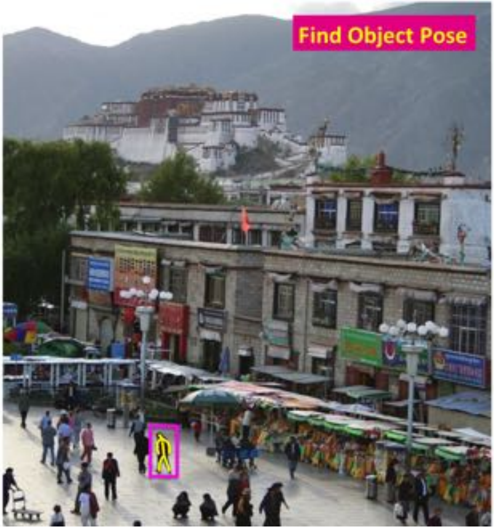
The Recognition Tasks

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



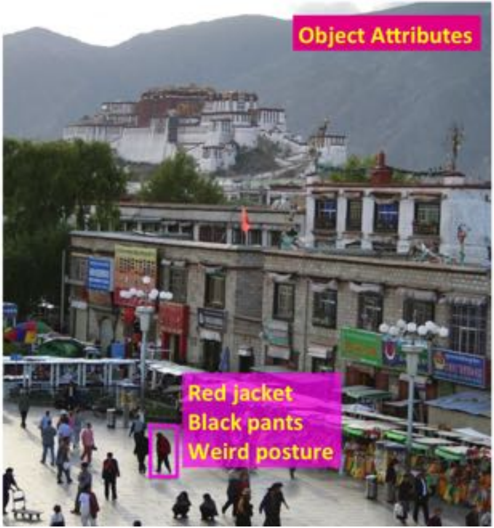
The Recognition Tasks

- 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 ;)

The Recognition Tasks

- Action recognition: What is happening in the image?

The Recognition Tasks

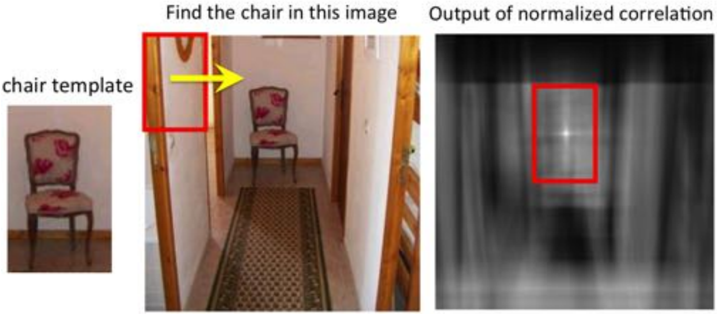
- Surveillance: Why is something happening?

Try Before Listening to the Next 8 Classes

- 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)



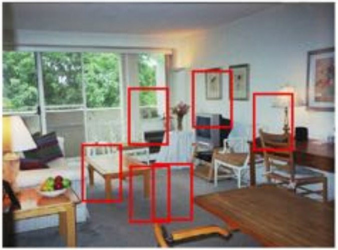
[Slide from: A. Torralba]

Template Matching

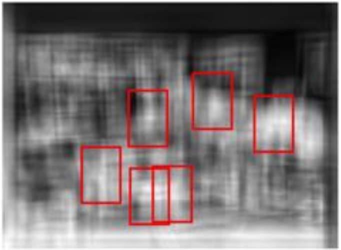
- Template matching: normalized cross-correlation with a template (filter)

template

Find the chair in this image



Pretty much garbage
Simple template matching is
not going to make it



My biggest concern while making this slide was:
how do I justify 50 years of research, and this course, if this experiment did work?

[Slide from: A. Torralba]

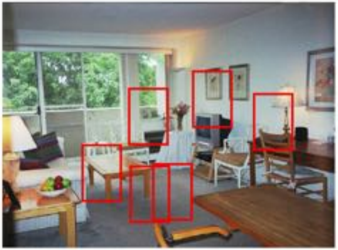
Template Matching

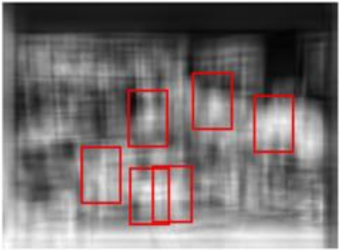
- Template matching: normalized cross-correlation with a template (filter)

template

Pretty much garbage
Simple template matching is
not going to make it

Find the chair in this image





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

query

Recognition via Retrieval by Similarity

- Upload a photo to Google image search
- Pretty reasonable, both are Golden Gate Bridge

query

Recognition via Retrieval by Similarity

- Upload a photo to Google image search
- Let's try a typical bathtub object

query

Recognition via Retrieval by Similarity

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

query

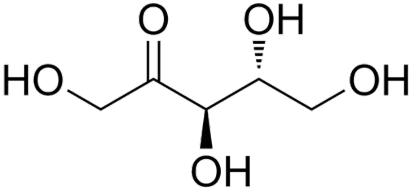
Recognition via Retrieval by Similarity

- Make a beautiful drawing and upload to Google image search
- Can you recognize this object?

query

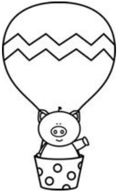
Recognition via Retrieval by Similarity

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



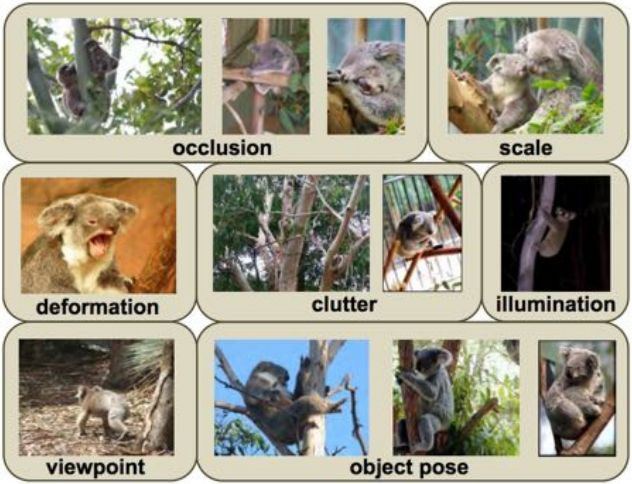
query

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

Overview

- 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**

Overview

- 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.

Overview

- 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

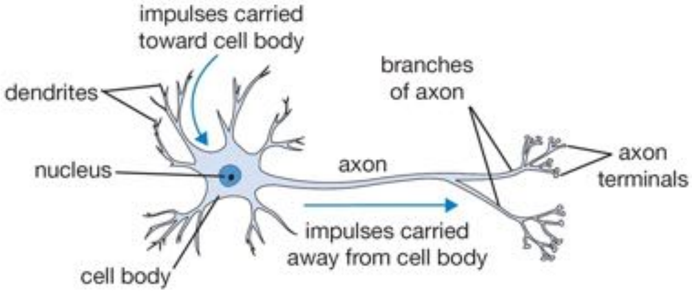


Figure: The basic computational unit of the brain: Neuron

[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*

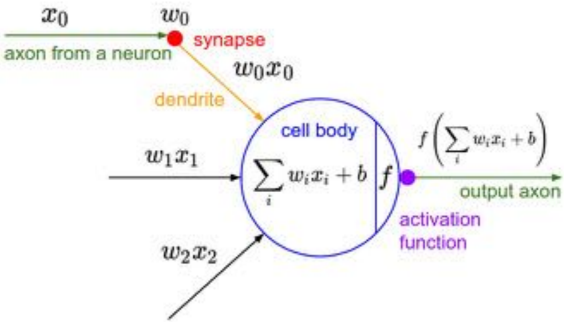


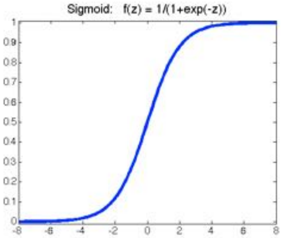
Figure: A mathematical model of the neuron in a neural network

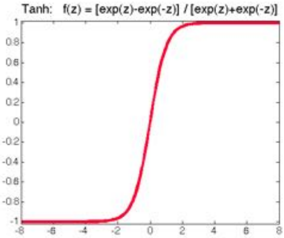
[Pic credit: http://cs231n.github.io/neural-networks-1/]

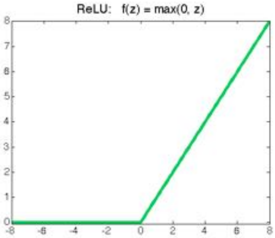
Activation Functions

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): $\mathrm{ReLU}(z) = \max(0, z)$







Neuron in Python

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

```
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:

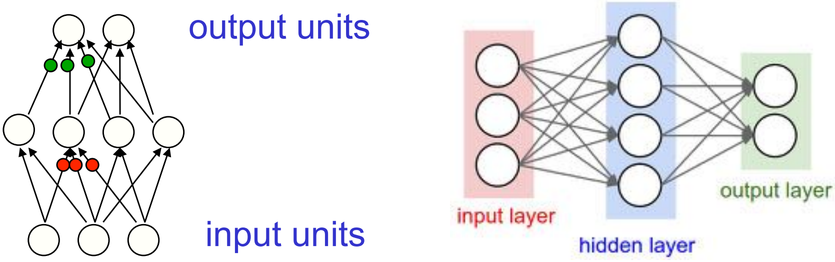


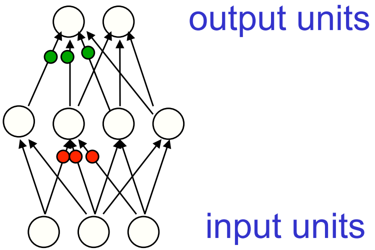
Figure: Two different visualizations of a 2-layer neural network. In this example: 3 input units, 4 hidden units and 2 output units

- 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:



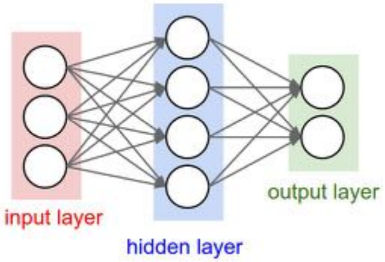


Figure: Two different visualizations of a 2-layer neural network. In this example: 3 input units, 4 hidden units and 2 output units

- 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/]

Neural Network Architecture (Multi-Layer Perceptron)

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

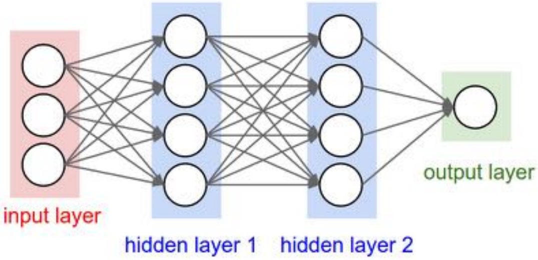


Figure: A 3-layer neural net with 3 input units, 4 hidden units in the first and second hidden layer and 1 output unit

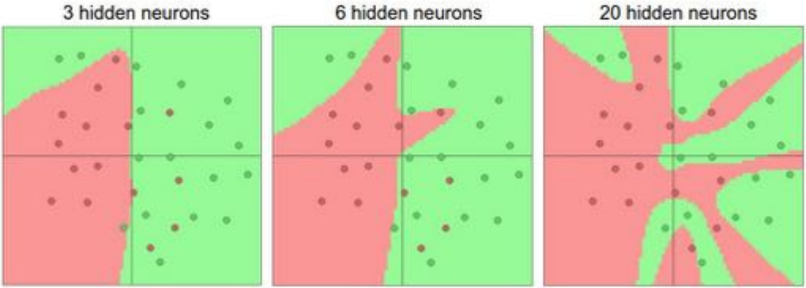
- Naming conventions; a N-layer neural network:
  - $N-1$ layers of hidden units
  - One output layer

[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

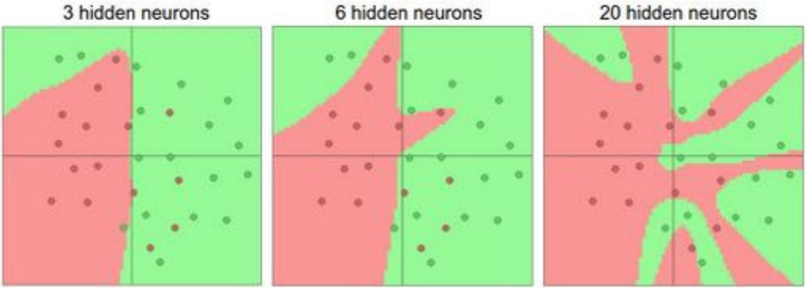


- The capacity of the network increases with more hidden units and more hidden layers

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



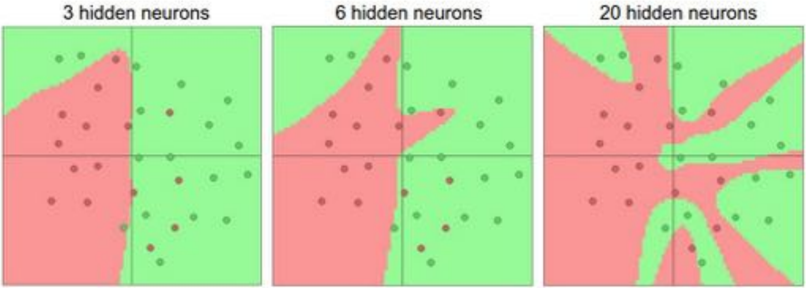
- The capacity of the network increases with more hidden units and more hidden layers
- Why go deeper? Read eg: Do Deep Nets Really Need to be Deep? Jimmy Ba, Rich Caruana, Paper: paper]

[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



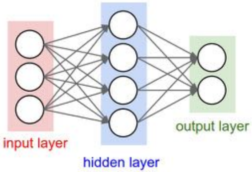
- The capacity of the network increases with more hidden units and more hidden layers
- Why go deeper? Read eg: Do Deep Nets Really Need to be Deep? Jimmy Ba, Rich Caruana, Paper: paper]

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

Neural Networks

- We only need to know two algorithms
  - *Forward pass:* performs inference
  - *Backward pass:* performs learning

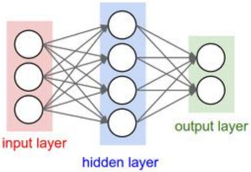
Forward Pass: What does the Network Compute?



- Output of the network can be written as:

$$h_j(\mathbf{x}) \quad = \quad f(v_{j0} + \sum_{i=1}^{D} x_i v_{ji})$$

Forward Pass: What does the Network Compute?

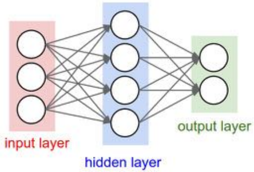


- Output of the network can be written as:

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

(j indexing hidden units, k indexing the output units, D number of inputs)

Forward Pass: What does the Network Compute?



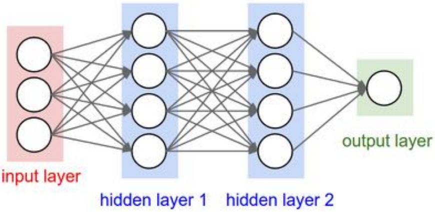
- Output of the network can be written as:

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

(j indexing hidden units, k indexing the output units, D number of inputs)

Forward Pass in Python

- Example code for a forward pass for a 3-layer network in Python:



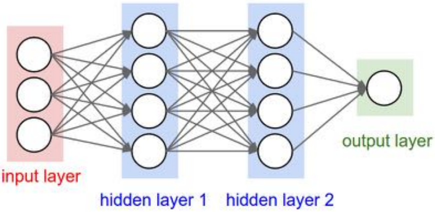

```
# forward-pass of a 3-layer neural network:
f = lambda x: 1.0/(1.0 + np.exp(-x)) # activation function (use sigmoid)
x = np.random.randn(3, 1) # random input vector of three numbers (3x1)
h1 = f(np.dot(W1, x) + b1) # calculate first hidden layer activations (4x1)
h2 = f(np.dot(W2, h1) + b2) # calculate 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 4×3, W_2 is 4×4. What about biases and W_3?

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

Forward Pass in Python

- 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 4×3, W_2 is 4×4. What about biases and W_3?

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

Training Neural Networks

- Find weights:

$$\mathbf{w}^* = \underset{\mathbf{w}}{\text{argmin}} \sum_{n=1}^{N} \text{loss}(\mathbf{o}^{(n)}, \mathbf{t}^{(n)})$$

  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:

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{n=1}^{N} \operatorname{loss}(\mathbf{o}^{(n)}, \mathbf{t}^{(n)})$$

  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 η is the learning rate (and E is error/loss)

Training Neural Networks

- Find weights:

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{n=1}^{N} \mathrm{loss}(\mathbf{o}^{(n)}, \mathbf{t}^{(n)})$$

  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 η 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{l-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
```

28

This code has a few bugs with indices...

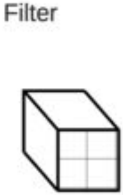
Ranzato

Convolutional Neural Networks (CNN)

- To work with images we typically use Neural Networks with special architecture

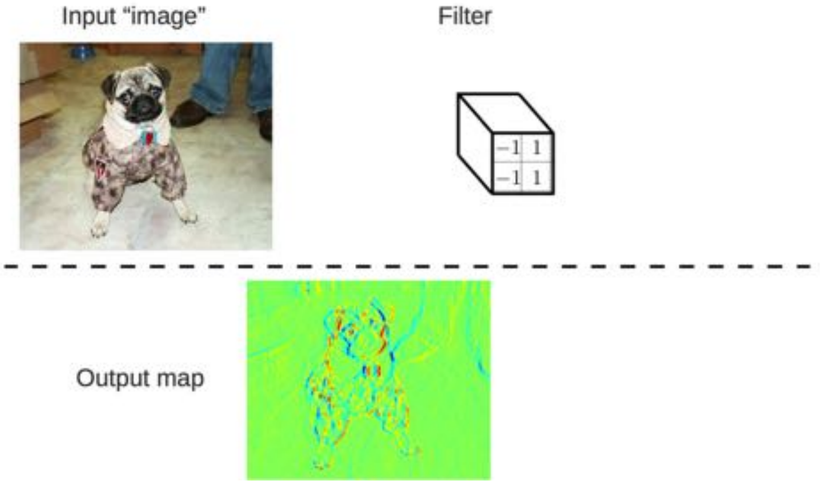
Convolutional Neural Networks (CNN)

- Remember our Lecture 2 about filtering?



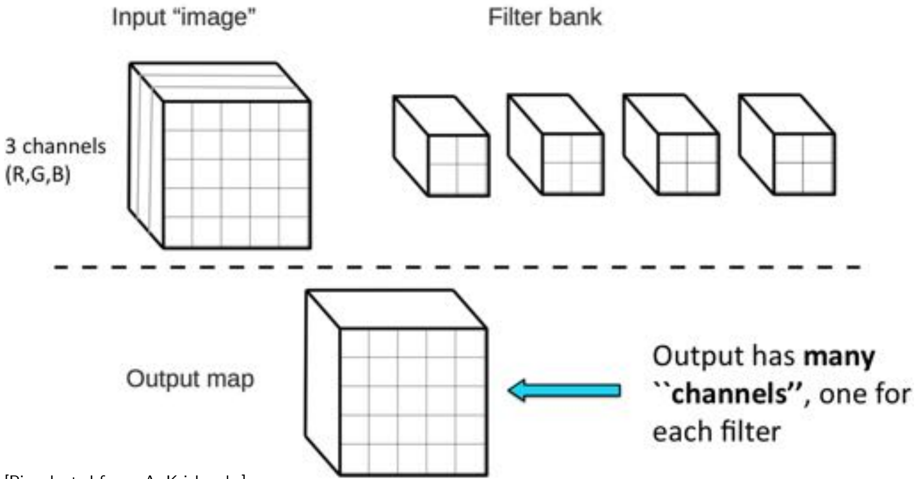
Convolutional Neural Networks (CNN)

- If our filter was $[-1, 1]$, we got a vertical edge detector



Convolutional Neural Networks (CNN)

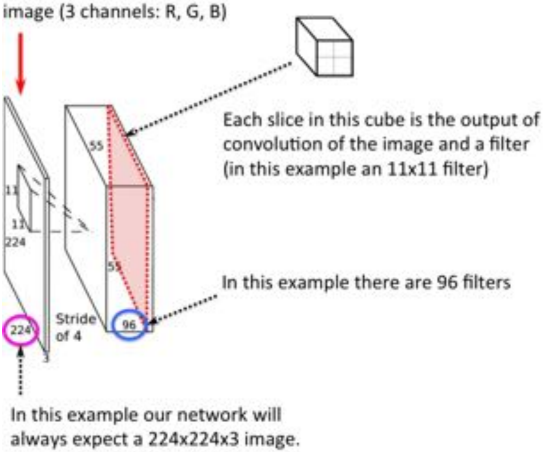
- Now imagine we didn't only want a vertical edge detector, but also a horizontal one, and one for corners, one for dots, etc. We would need to take many filters. A **filterbank**.



[Pic adapted from: A. Krizhevsky]

Convolutional Neural Networks (CNN)

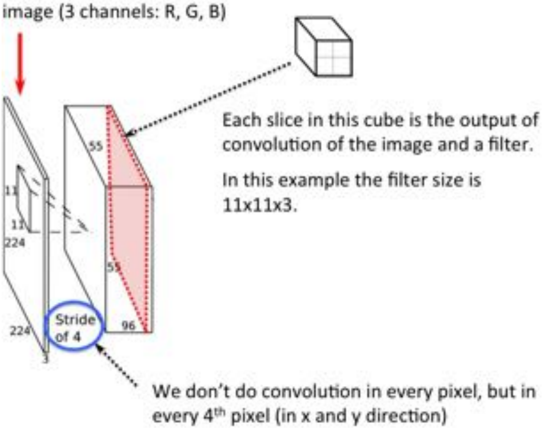
- Applying a filterbank to an image yields a cube-like output, a 3D matrix in which each slice is an output of convolution with one filter, and an activation function.



[Pic adopted from: A. Krizhevsky]

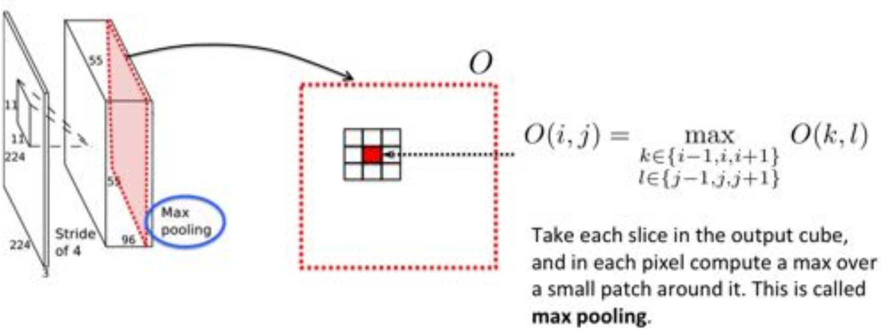
Convolutional Neural Networks (CNN)

- 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.



[Pic adopted from: A. Krizhevsky]

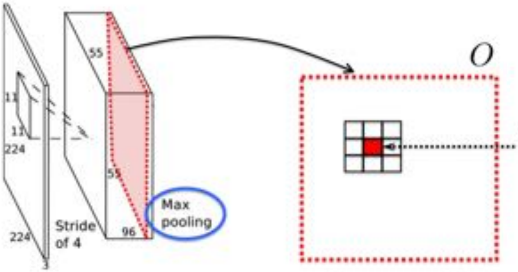
- Do some additional tricks. A popular one is called **max pooling**. Any idea why you would do this?



[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**.

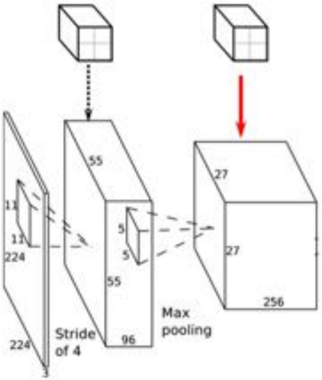


$$O(i,j) = \max_{\substack{k \in \{i-1,i,i+1\} \\ l \in \{j-1,j,j+1\}}} O(k,l)$$

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]

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



Add one more layer of filters

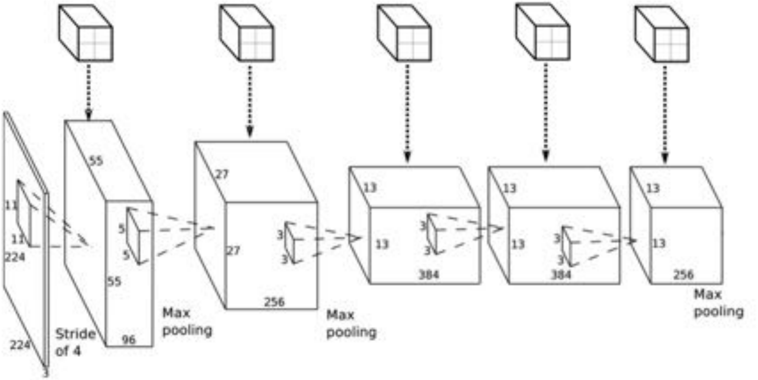
These filters are convolved with the output of the previous layer. The results of each convolution is again a slice in the cube on the right.

What is the dimension of each of these filters?

[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?



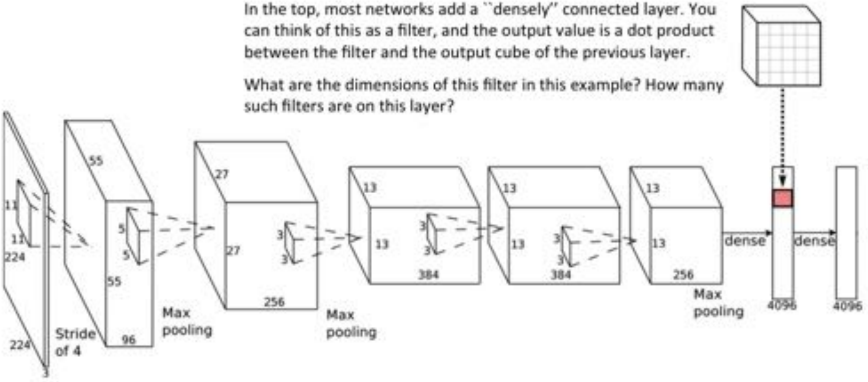
Do it recursively

Have multiple ``layers''

[Pic adopted from: A. Krizhevsky]

Convolutional Neural Networks (CNN)

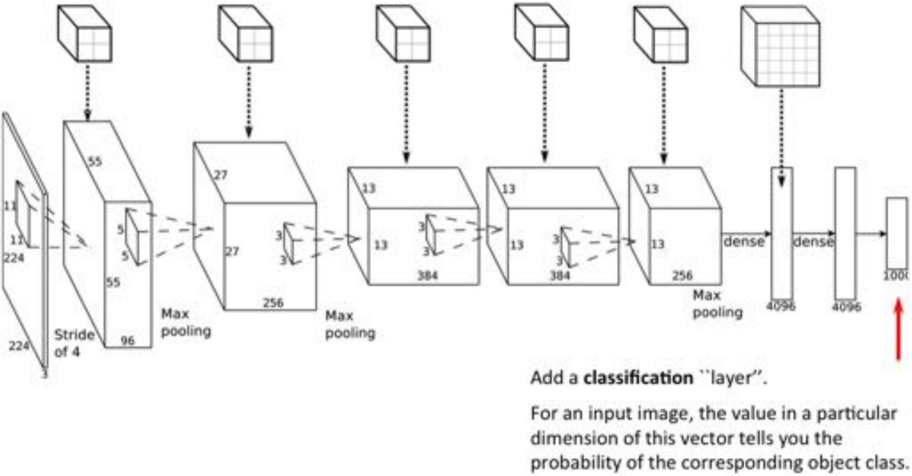
- In the end add one or two **fully** (or **densely**) connected layers. In this layer, we don't do convolution we just do a dot-product between the "filter" and the output of the previous layer.



[Pic adapted from: A. Krizhevsky]

Convolutional Neural Networks (CNN)

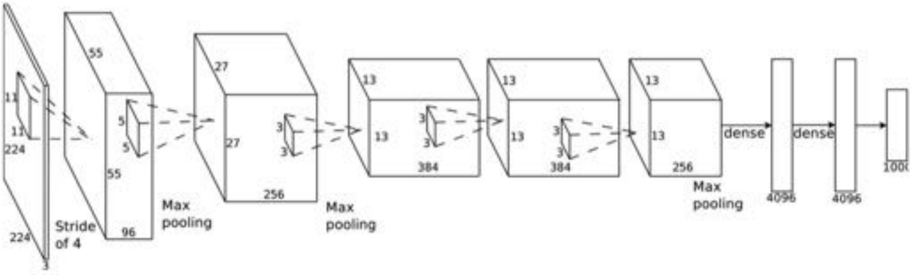
- Add one final layer: a **classification** layer. Each dimension of this vector tells us the probability of the input image being of a certain class.



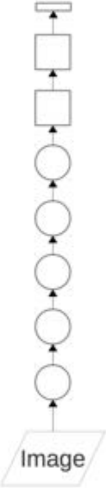
[Pic adopted from: A. Krizhevsky]

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

Convolutional layer: convolves its input with a bank of 3D filters, then applies point-wise non-linearity

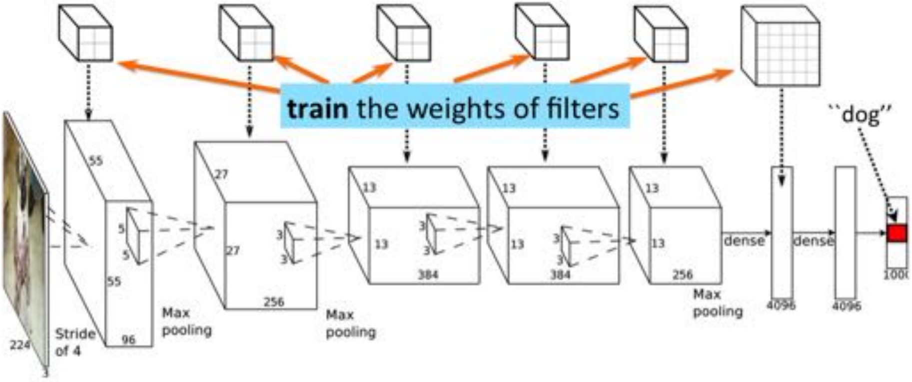
Fully-connected layer: applies linear filters to its input, then applies point-wise non-linearity

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

[Pic adopted from: A. Krizhevsky]

Convolutional Neural Networks (CNN)

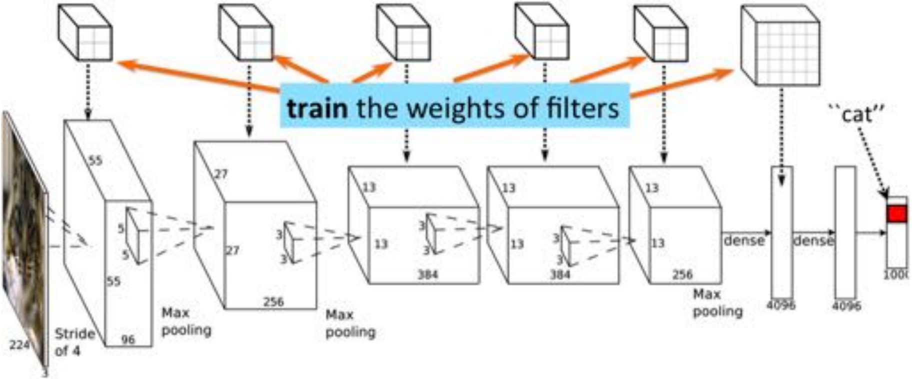
- The trick is to not hand-fix the weights, but to **train** them. Train them such that when the network sees a picture of a dog, the last layer will say "dog".



[Pic adopted from: A. Krizhevsky]

Convolutional Neural Networks (CNN)

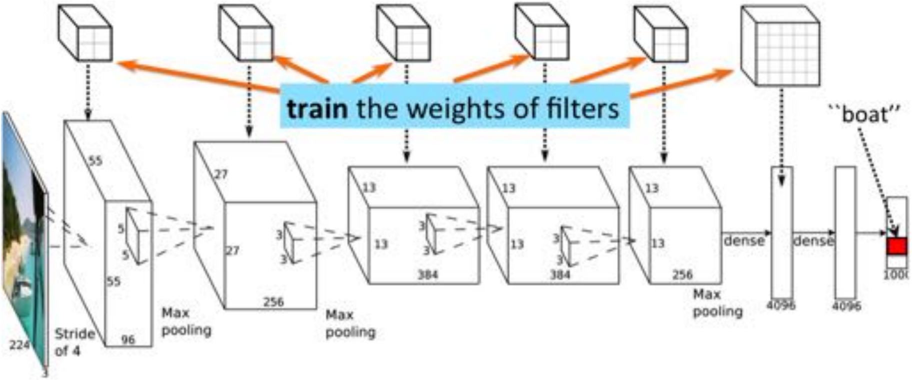
- Or when the network sees a picture of a cat, the last layer will say "cat".



[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.



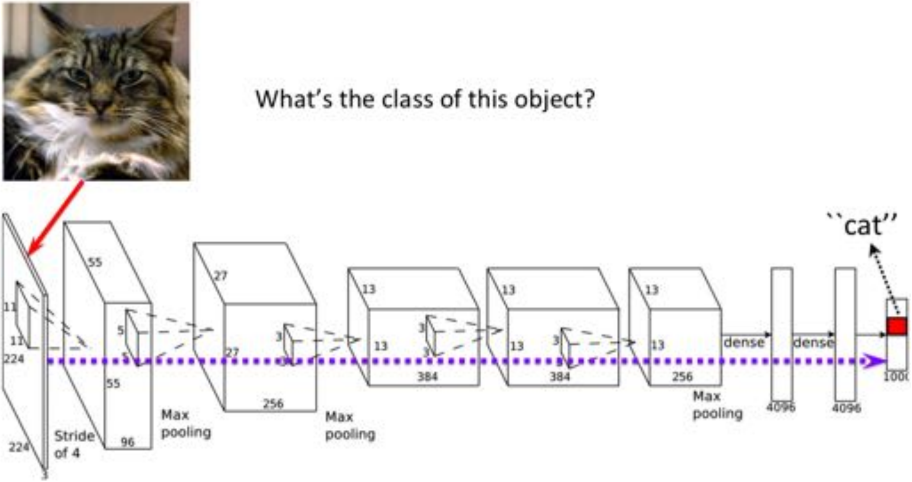
Train on **lots** of examples. Millions. Tens of millions. Wait a week for training to finish.

Share your network (the weights) with others who are not fortunate enough with GPU power.

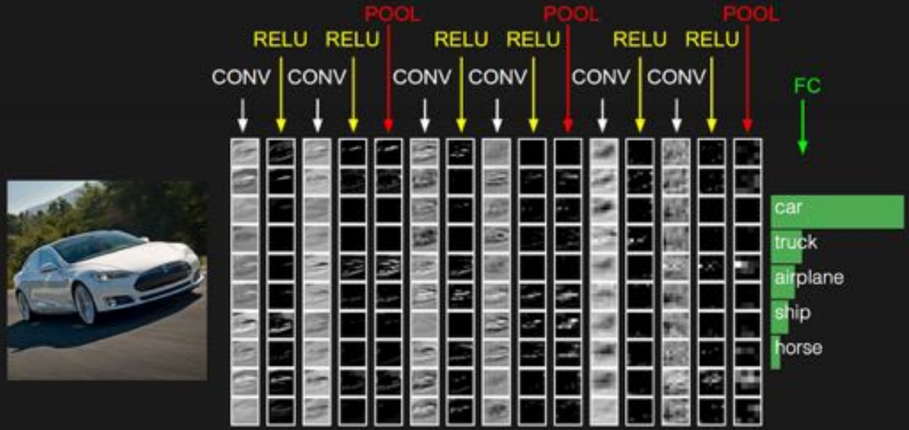
[Pic adopted from: A. Krizhevsky]

Classification

- 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



[http://cs231n.github.io/convolutional-networks/]

Classification Performance

- Imagenet, main challenge for object classification: `http://image-net.org/`
- 1000 classes, 1.2M training images, 150K for test

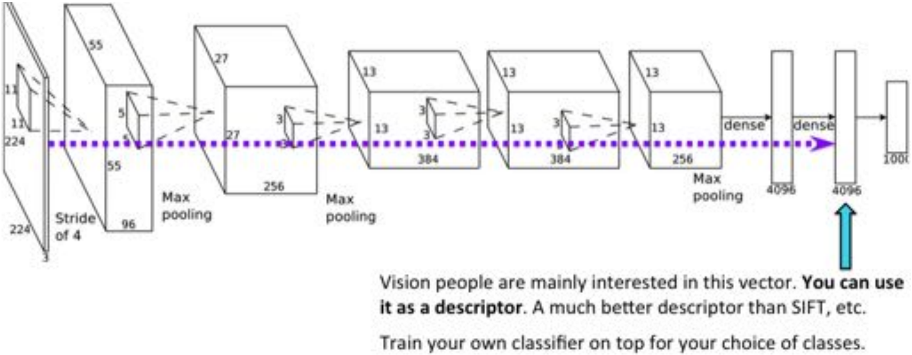
Classification Performance Three Years Ago (2012)

- A. Krizhevsky, I. Sutskever, and G. E. Hinton rock the Imagenet Challenge

Team name	Filename	Error (5 guesses)	Description
SuperVision	test-preds-141-146.2009-131-137-145-146.2011-145f.	0.15315	Using extra training data from ImageNet Fall 2011 release
SuperVision	test-preds-131-137-145-135-145f.txt	0.16422	Using only supplied training data
ISI	pred_FVs_wLACs_weighted.txt	0.26172	Weighted sum of scores from each classifier with SIFT+FV, LBP+FV, GIST+FV, and CSIFT+FV, respectively.
ISI	pred_FVs_weighted.txt	0.26602	Weighted sum of scores from classifiers using each FV.
ISI	pred_FVs_summed.txt	0.26646	Naive sum of scores from classifiers using each FV.
ISI	pred_FVs_wLACs_summed.txt	0.26952	Naive sum of scores from each classifier with SIFT+FV, LBP+FV, GIST+FV, and

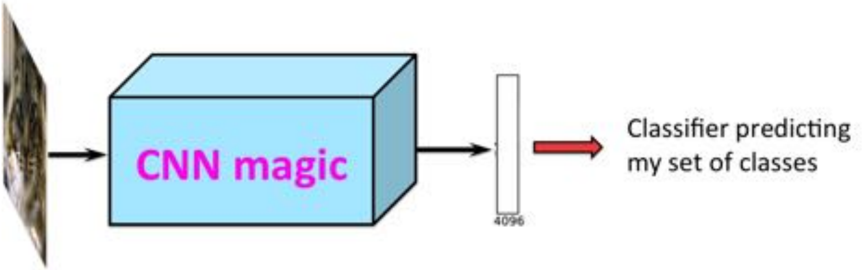
Neural Networks as Descriptors

- 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).



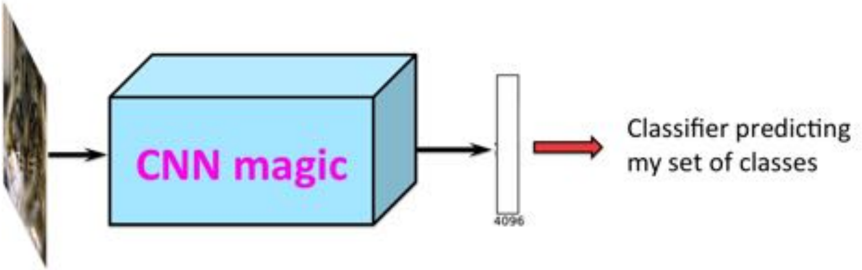
Neural Networks as Descriptors

- 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.



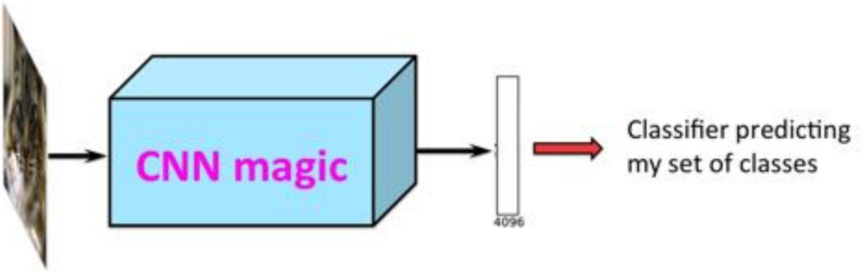
Neural Networks as Descriptors

- 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.



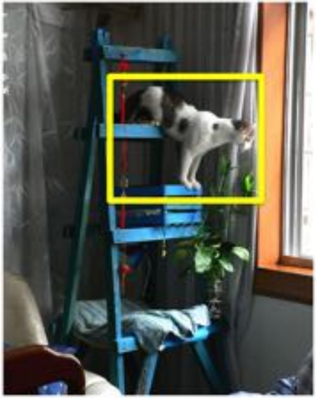
Neural Networks as Descriptors

- 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

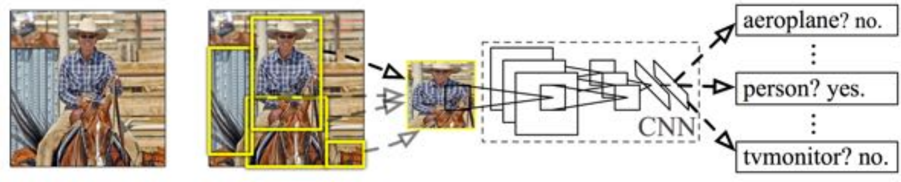


Figure: R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation, CVPR'14

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 (224×224) and run through the network

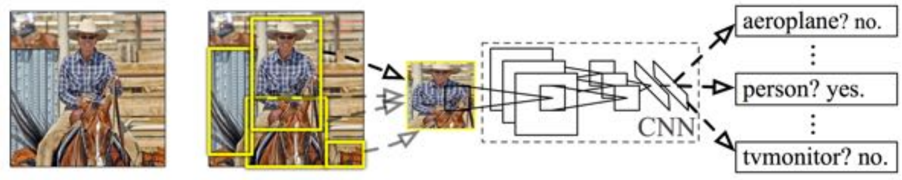


Figure: R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation, CVPR'14

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 (224×224) 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.

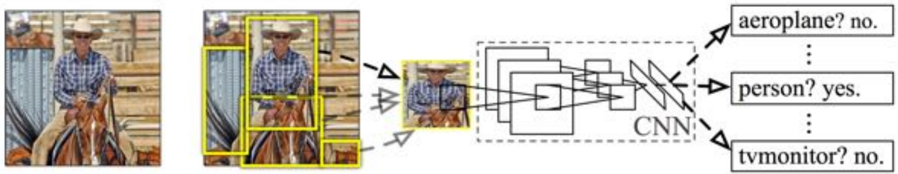


Figure: R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation, CVPR'14

And Detection?

- 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.

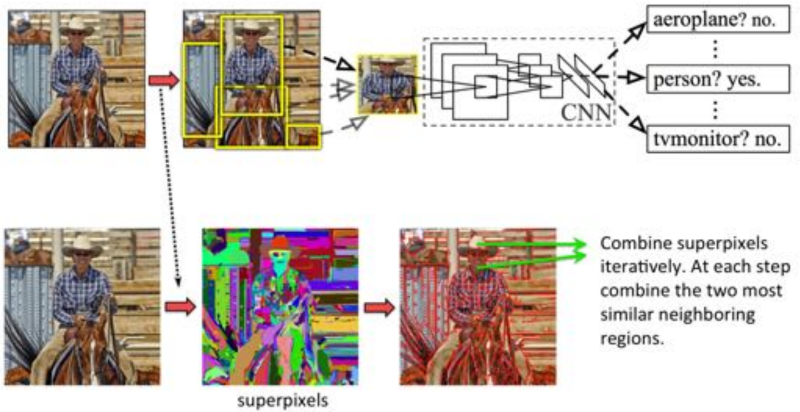


Figure: Bottom: J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, A. W. M. Smeulders, Selective Search for Object Recognition, IJCV 2013

And Detection?

- 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.

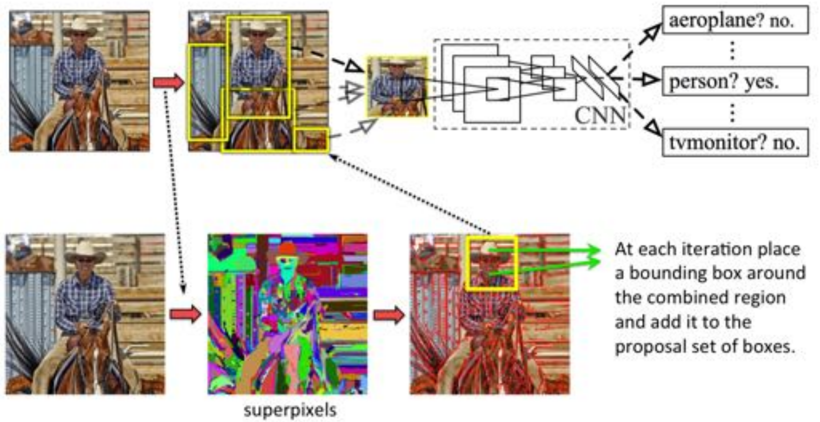


Figure: Bottom: J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, A. W. M. Smeulders, Selective Search for Object Recognition, IJCV 2013

Detection Datasets

- **PASCAL VOC challenge**: http://pascallin.ecs.soton.ac.uk/challenges/VOC/.

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**.

	mean	aero plane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	dining table	dog	horse	motor bike	person	potted plant	sheep	sofa	train	tv/ monitor	submission date
segDPM [?]	40.4	61.4	53.4	25.6	25.2	35.5	51.7	50.6	50.8	19.3	33.8	26.8	40.4	48.3	54.4	47.1	14.8	38.7	35.0	52.8	43.1	24-Feb-2013
Boosted HOG-LBP and multi-context (LC, EGC, HLC) [?]	36.8	53.3	55.3	19.2	21.0	30.0	54.5	46.7	41.2	20.8	31.5	20.8	30.3	48.6	55.3	46.5	10.2	34.4	26.6	50.3	40.3	29-Aug-2010
MITUCLA_Hierarchy [?]	36.0	54.3	48.5	15.7	19.2	29.2	55.6	43.5	41.7	16.9	28.5	26.7	30.9	48.3	55.0	41.7	9.7	35.8	30.8	47.2	40.8	30-Aug-2010
HOGLBP_context_classification_rescore_v2 [?]	34.2	49.1	52.4	17.8	12.0	30.6	53.5	32.8	37.3	17.7	30.6	27.7	29.5	51.9	56.3	44.2	9.6	14.8	27.9	49.5	38.4	30-Aug-2010
LSVM-MDPM [?]	33.7	52.4	54.3	13.0	15.6	35.1	54.2	49.1	31.8	15.5	26.2	13.5	21.5	45.4	51.6	47.5	9.1	35.1	19.4	46.6	38.0	26-Aug-2010
UOCTTI_LSVM_MDPM [?]	33.4	49.2	53.8	13.1	15.3	35.5	53.4	49.7	27.0	17.2	28.8	14.7	17.8	46.4	51.2	47.7	10.8	34.2	20.7	43.8	38.3	21-May-2012
Detection Monkey [?]	32.9	56.7	39.8	16.8	12.2	13.8	44.9	36.9	47.7	12.1	26.9	26.5	37.2	42.1	51.9	25.7	12.1	37.8	33.0	41.5	41.7	30-Aug-2010
RM^2C [?]	32.8	49.8	50.6	15.1	15.5	28.5	51.1	42.2	30.3	17.3	28.3	12.4	26.0	45.6	51.8	41.4	12.6	30.4	26.1	44.0	37.6	29-Oct-2010
UOCTTI_LSVM_MDPM [?]	32.2	48.2	52.2	14.8	13.8	28.7	53.2	44.9	26.0	18.4	24.4	13.7	23.1	45.8	50.5	43.7	9.8	31.1	21.5	44.4	35.7	11-May-2012
GroupLoc [?]	31.9	58.4	39.6	18.0	13.3	11.1	46.4	37.8	43.9	10.3	27.5	20.8	36.0	39.4	48.5	22.9	13.0	36.9	30.5	41.2	41.9	30-Aug-2010
UOCTTI_LSVM_MDPM [?]	29.6	45.6	49.0	11.0	11.6	27.2	50.5	43.1	23.6	17.2	23.2	10.7	20.5	42.5	44.5	41.3	8.7	29.0	18.7	40.0	34.5	21-Mar-2012
Bonn_FGT_Segm [?]	26.1	52.7	33.7	13.2	11.0	14.2	43.2	31.9	35.6	5.8	25.4	14.4	20.6	38.1	41.7	25.0	5.8	26.3	18.1	37.6	28.1	30-Aug-2010
HOG-LBP + DHOG bag of words, SVM [?]	23.3	40.4	34.7	2.7	8.4	26.0	43.1	33.8	17.2	11.2	14.3	14.5	14.9	31.8	37.3	30.0	6.4	25.2	11.6	30.0	35.7	30-Aug-2010
Svr-Segm [?]	23.4	30.5	24.5	17.1	13.3	10.9	39.5	32.9	36.5	5.6	16.0	6.6	22.3	24.8	29.0	29.8	6.7	28.4	13.3	32.1	27.2	30-Aug-2010
HOG-LBP Linear SVM [?]	22.1	37.9	33.7	2.7	6.5	25.3	37.5	33.1	15.1	10.9	12.3	12.5	13.7	29.7	34.5	33.8	7.2	22.9	9.9	28.9	34.1	29-Aug-2010
HOG+LBP+LTP+PLS2ROOTS [?]	17.3	32.7	29.7	0.8	1.1	19.9	39.4	27.5	8.6	4.5	8.1	6.3	11.0	22.8	34.1	24.6	3.1	24.0	2.0	23.5	27.0	31-Aug-2010
RandomParts [?]	14.2	23.8	33.7	1.2	3.4	11.1	29.7	19.5	14.2	6.8	11.3	7.8	4.7	16.4	31.5	16.0	1.1	15.6	10.2	14.7	21.0	25-Aug-2010
SIFT-GMM-MKL2 [?]	8.3	20.0	14.5	3.8	1.2	0.5	17.6	8.1	28.3	0.1	2.8	3.1	17.5	7.2	18.8	3.3	0.8	2.9	6.3	7.6	1.3	30-Aug-2010
UC3M_Generative_Discriminative [?]	6.3	15.8	5.5	5.6	2.3	0.3	10.2	5.4	12.6	0.5	5.8	4.5	7.7	11.3	12.6	5.3	1.5	2.0	5.9	9.1	3.2	30-Aug-2010
SIFT-GMM-MKL [?]	2.3	10.6	1.6	1.2	0.9	0.1	2.8	1.6	6.7	0.1	2.0	0.4	3.0	2.0	4.4	2.0	0.3	1.1	1.2	2.1	1.9	30-Aug-2010

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

Detection Performance in 2014: 53.7%

In 2014, networks:

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

	mean	aero plane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	dining table	dog	horse	motor bike	person	potted plant	sheep	sofa	train	tv/ monitor	submission date
R-CNN (bbox reg)	53.7	71.8	65.8	53.0	36.8	35.9	59.7	60.0	69.9	27.9	50.6	41.4	70.0	62.0	69.0	58.1	29.5	59.4	39.3	61.2	52.4	2014-Mar-13
R-CNN	50.2	67.1	64.1	46.7	32.0	30.5	56.4	57.2	65.9	27.0	47.3	40.9	66.6	57.8	65.9	53.6	26.7	56.5	38.1	52.8	50.2	2014-Jan-30

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

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

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

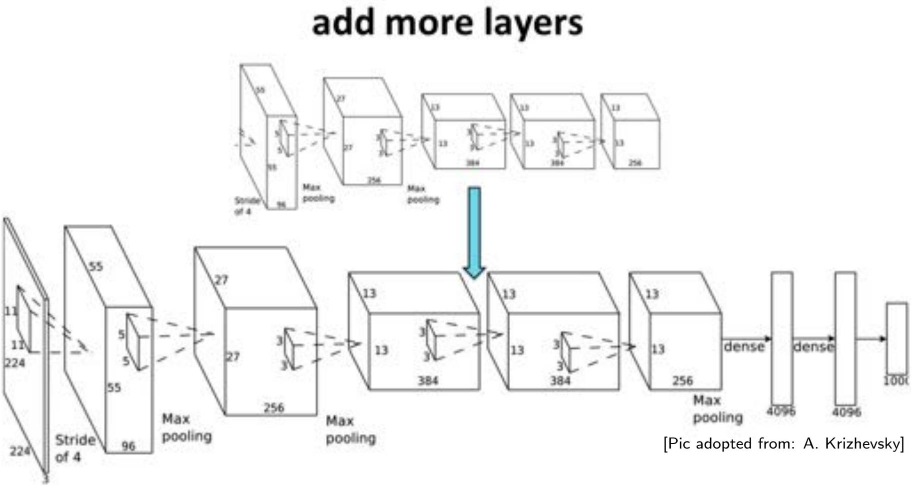
- But to train the networks you need quite a bit of computational power. So what do you do?

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.

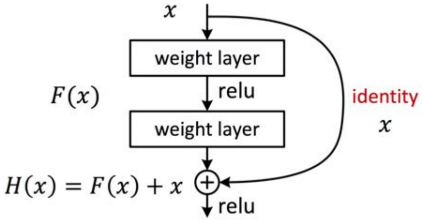


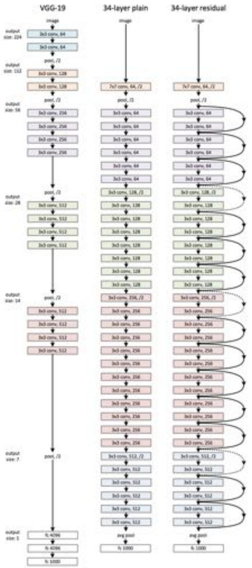
[Pic adopted from: A. Krizhevsky]

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

150 Layers!

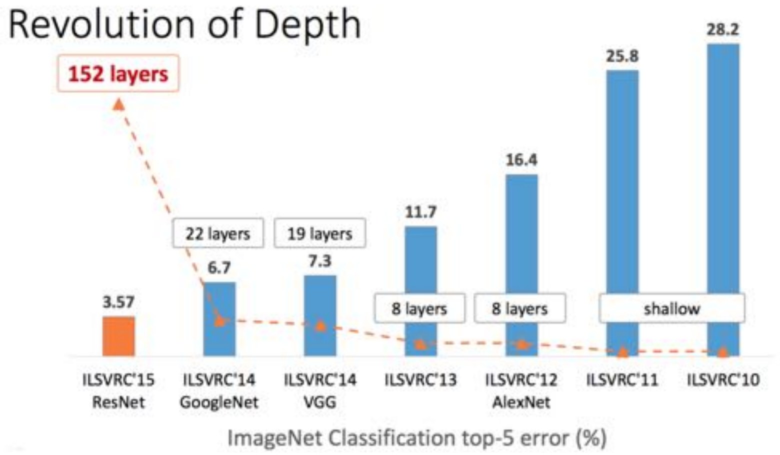
- Networks are now at 150 layers
- They use a skip connections with special form
- In fact, they don't fit on this screen
- Amazing performance!
- A lot of "mistakes" are due to wrong ground-truth





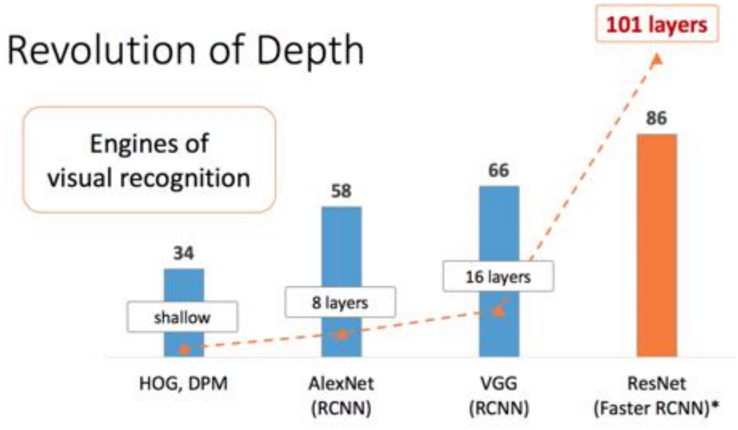
[He, K., Zhang, X., Ren, S. and Sun, J., 2015. Deep Residual Learning for Image Recognition. arXiv:1512.03385, 2016]

Results: Object Classification



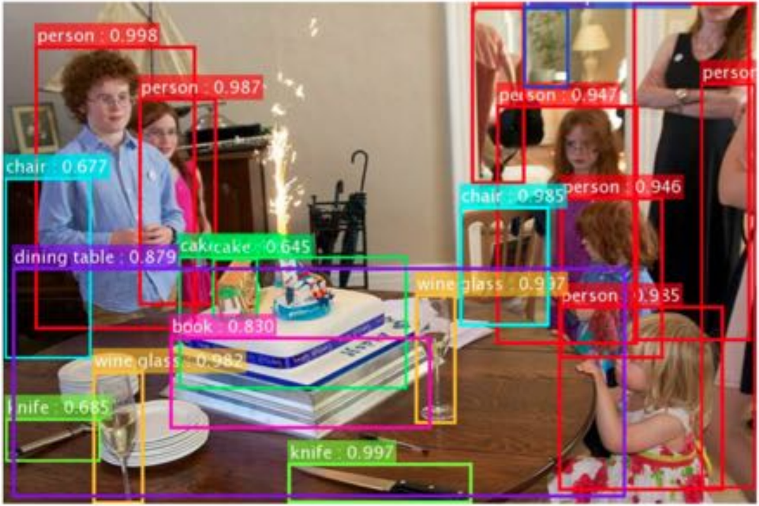
Slide: R. Liao, Paper: [He, K., Zhang, X., Ren, S. and Sun, J., 2015. Deep Residual Learning for Image Recognition. arXiv:1512.03385, 2016]

Results: Object Detection



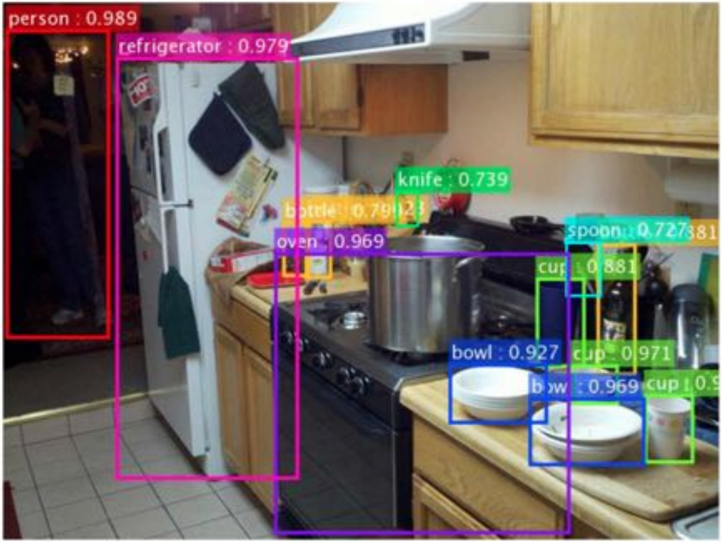
Slide: R. Liao, Paper: [He, K., Zhang, X., Ren, S. and Sun, J., 2015. Deep Residual Learning for Image Recognition. arXiv:1512.03385, 2016]

Results: Object Detection

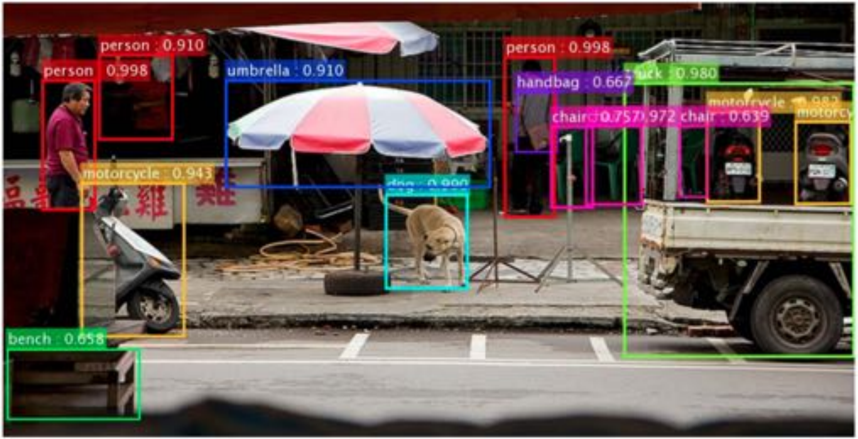


Slide: R. Liao, Paper: [He, K., Zhang, X., Ren, S. and Sun, J., 2015. Deep Residual Learning for Image Recognition. arXiv:1512.03385, 2016]

Results: Object Detection



Results: Object Detection



Slide: R. Liao, Paper: [He, K., Zhang, X., Ren, S. and Sun, J., 2015. Deep Residual Learning for Image Recognition. arXiv:1512.03385, 2016]

What do CNNs Learn?

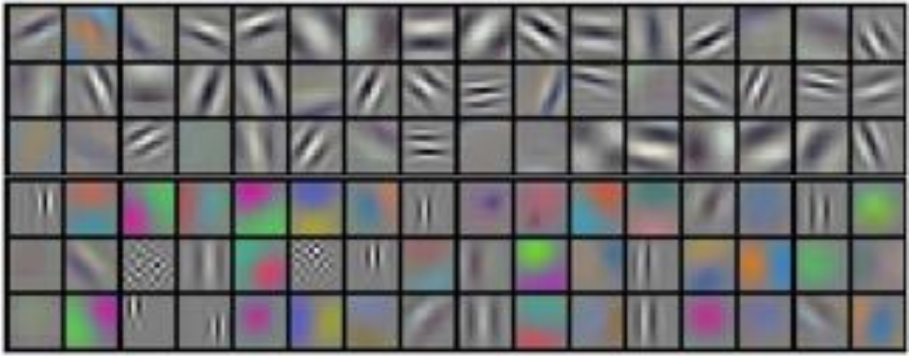


Figure: Filters in the first convolutional layer of Krizhevsky et al

What do CNNs Learn?

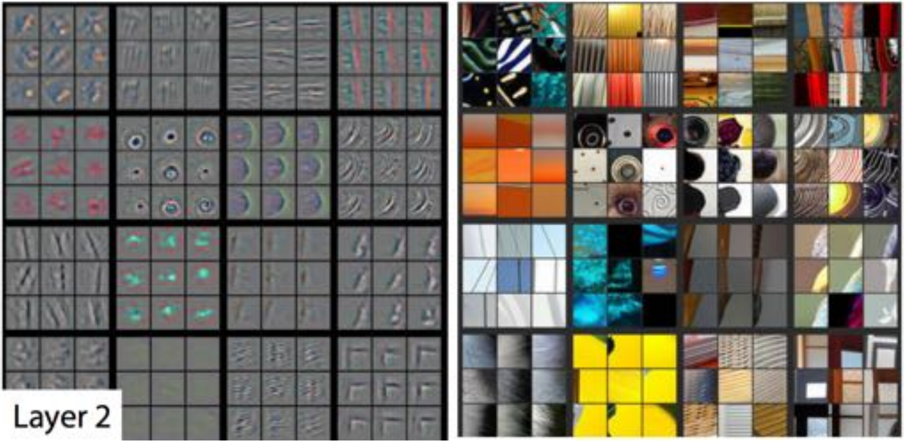


Figure: Filters in the second layer

[http://arxiv.org/pdf/1311.2901v3.pdf]

What do CNNs Learn?

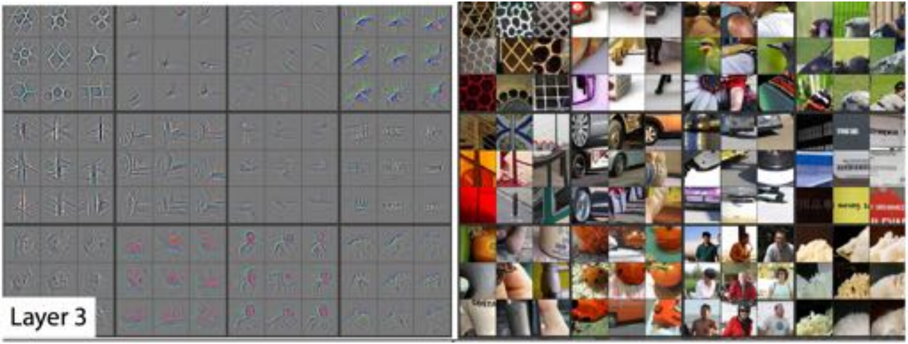
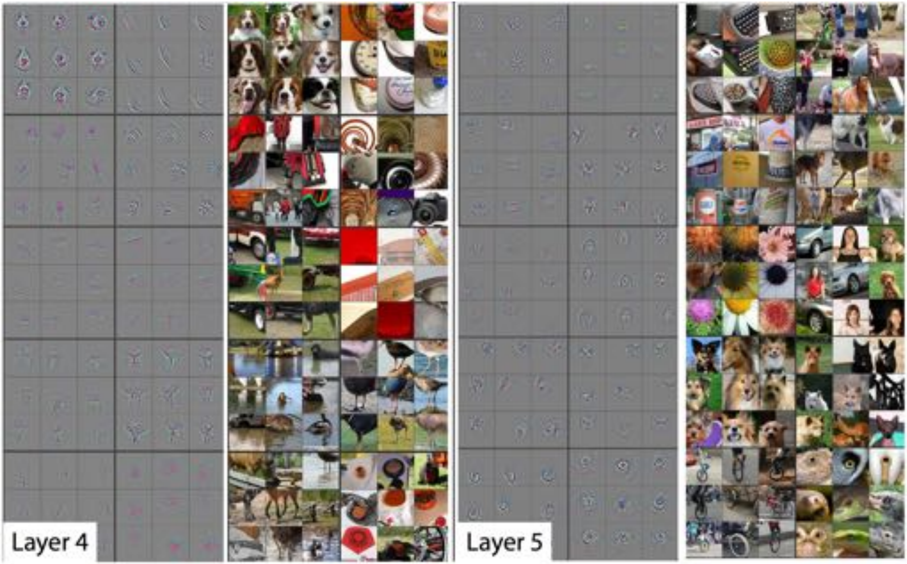


Figure: Filters in the third layer

[http://arxiv.org/pdf/1311.2901v3.pdf]

What do CNNs Learn?



[http://arxiv.org/pdf/1311.2901v3.pdf]

Neural Networks – Can Do Anything

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

How would you use them for these tasks?

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

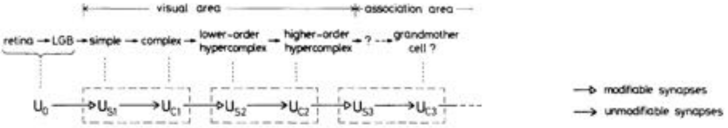
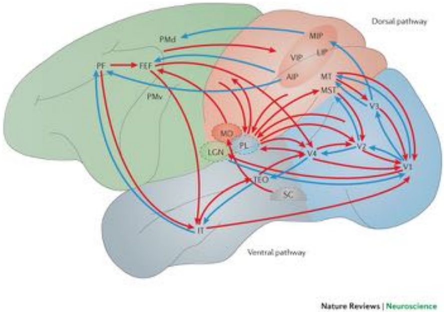


Fig. 1. Correspondence between the hierarchy model by Hubel and Wiesel, and the neural network of the neocognitron

Figure: Fukushima, Neocognitron. Biol. Cybernetics, 1980



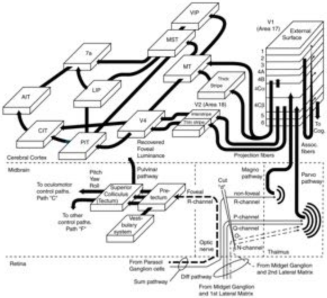


Figure: http://www.nature.com/nrn/journal/v14/n5/figs/recognition/nrn3476-f1.jpg,

Neuroscience

- V1: selective to direction of movement (Hubel & Wiesel)

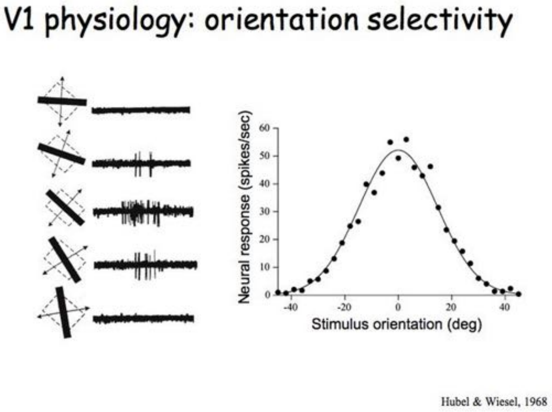


Figure: Pic from:
http://www.cns.nyu.edu/~david/courses/perception/lecturenotes/V1/LGN-V1-slides/Slide15.jpg

Neuroscience

- V2: selective to combinations of orientations

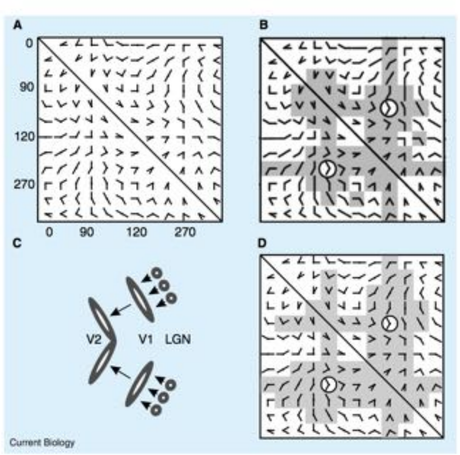


Figure 1. The selectivity of a V2 neuron can be explained by two V1 inputs.

(A) Angle stimuli, consisting of two line segments, used by Ito and Komatsu [9] to study the selectivity of V2 neurons. The orientation of one line segment varies along the rows and the orientation of the other line segment varies along the columns. (B) The pattern of responses for an example neuron. Circles surround the stimulus that evoked the maximal response, and stimuli that evoked more than half this maximum are shaded in gray. (C) Our model V2 neuron sums the responses from two orientation-selective V1 neurons that sum the inputs from LGN cells with center-surround receptive fields [11]. (D) Predicted response from our model neuron to the stimulus set. Like the example V2 neuron, the model neuron responds to angle stimuli containing oriented line segments that match the preferred orientation of either of the two V1 input neurons.

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)

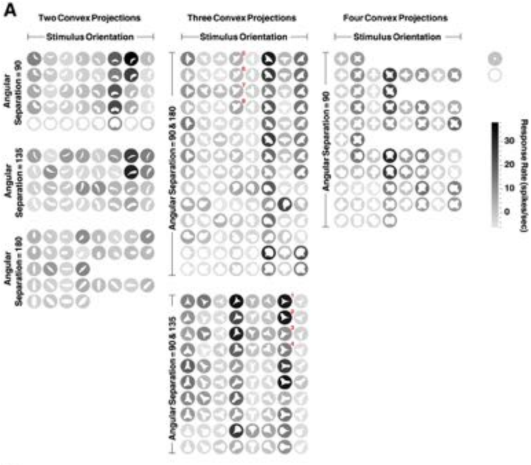


Figure: A. Pasupathy , C. E. Connor, Shape Representation in Area V4: Position-Specific Tuning for Boundary Conformation, Journal of Neurophysiology, 2001

- IT: Seems to be category selective

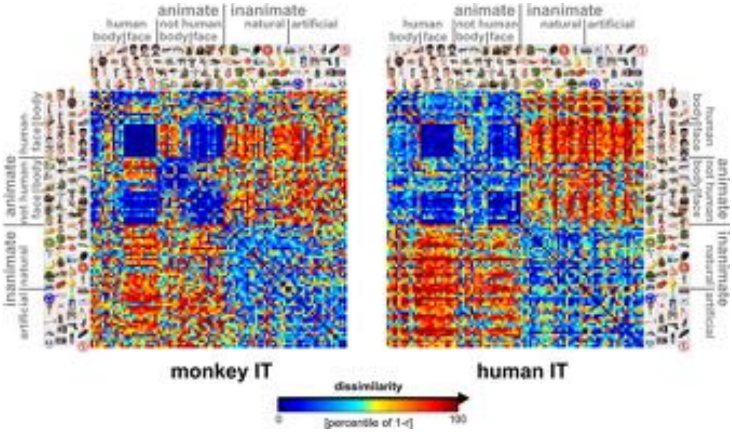


Figure: N. Kriegeskorte, M. Mur, D. A. Ruff, R. Kiani, J. Bodurka, H. Esteky, K. Tanaka, P. A. Bandettini, Matching Categorical Object Representations in Inferior Temporal Cortex of Man and Monkey, Neuron, 2008

Neuroscience

- Grandmother / Jennifer Aniston cell?

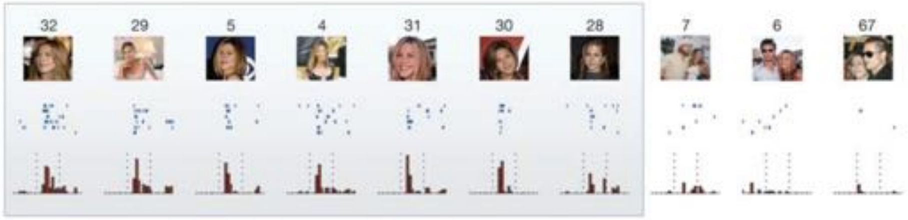


Figure: R. Q. Quiroga, L. Reddy, G. Kreiman, C. Koch, I. Fried, Invariant visual representation by single-neurons in the human brain. Nature, 2005

Neuroscience

- Grandmother / Jennifer Aniston cell?

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 minuscule. 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.

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

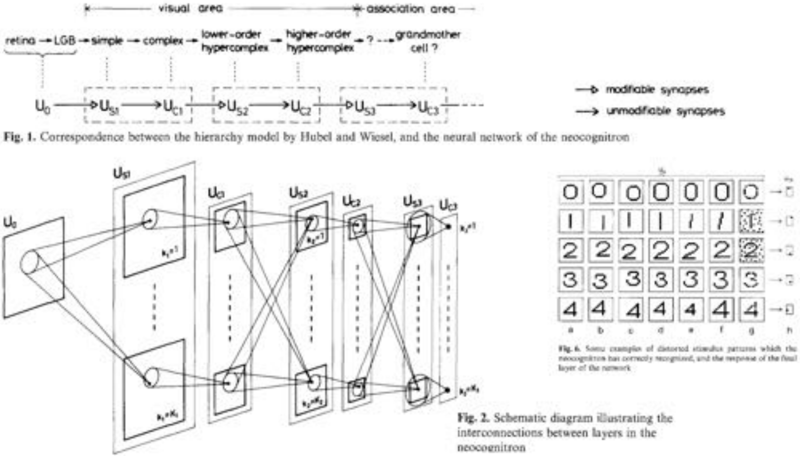


Fig. 1. Correspondence between the hierarchy model by Hubel and Wiesel, and the neural network of the neocognitron

Fig. 2. Schematic diagram illustrating the interconnections between layers in the neocognitron

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?

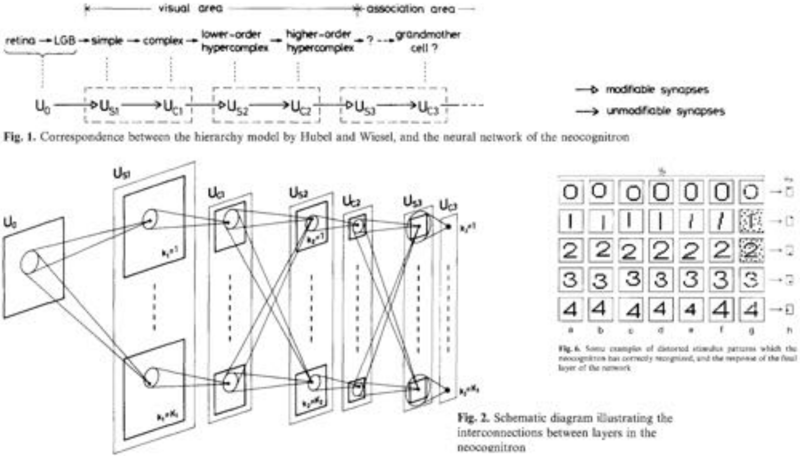


Figure: Fukushima, Neocognitron. Biol. Cybernetics, 1980

Neural Networks – Why Do They Work?

- Some cool tricks in design and training:

  A. Krizhevsky, I. Sutskever, G. E. Hinton, *ImageNet Classification with Deep Convolutional Neural Networks*, NIPS 2012

- 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

Code

Main code:

- Neural network packages:

  `http://caffe.berkeleyvision.org/`, Tensorflow, Theano, Torch, PyTorch

- Object detection:

  `https://github.com/rbgirshick/rcnn`
  `https://github.com/weiliu89/caffe/tree/ssd`

Summary – Stuff Useful to Know

- 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.