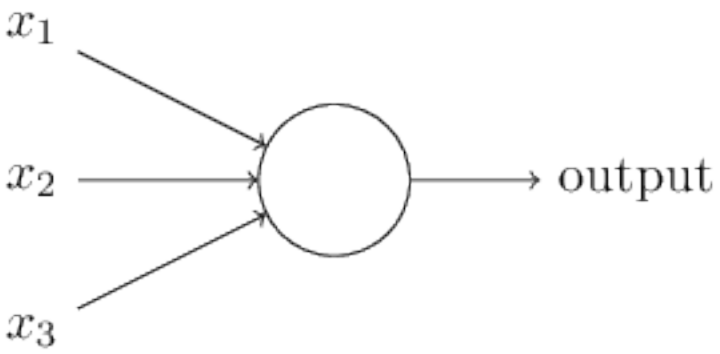
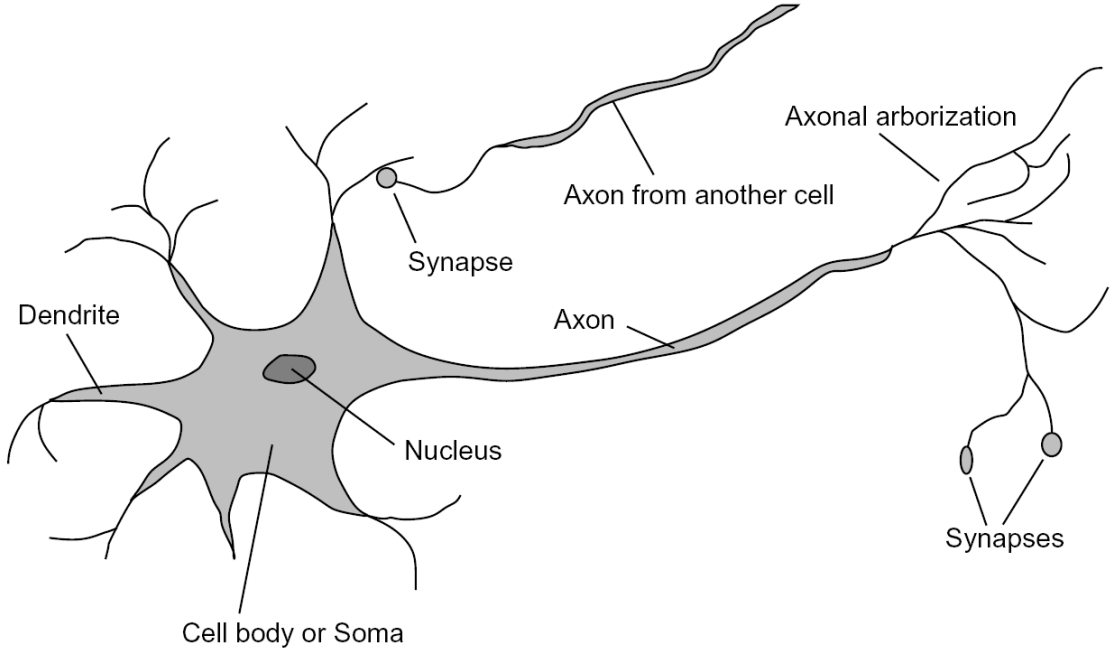
Neural Networks

This lecture is based on Michael Nielsen's *Neural Networks and Deep Learning,* a free online book (with code!).
Please read chapters 1-3.

Review: Perceptron

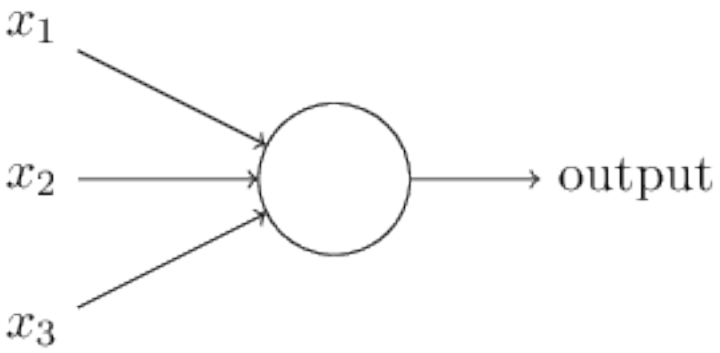
- Perceptrons were developed in the 1950s and 1960s loosely inspired by the neuron.





Review: Perceptron

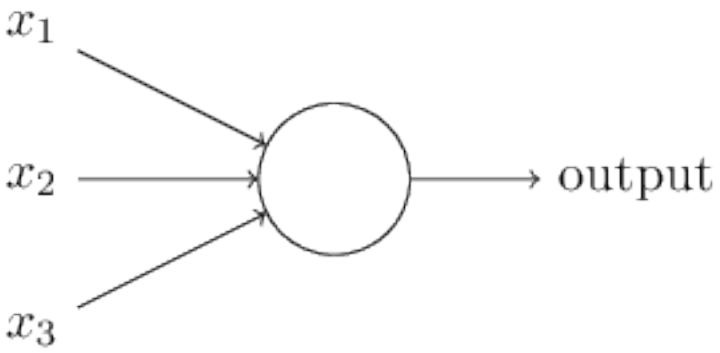
- Perceptron has inputs, $x_1, x_2, .. x_N$, and weights $w_1, w_2, .. w_N$
- The perceptron outputs 0 or 1, based on the weighted sum is less than or greater than a threshold value

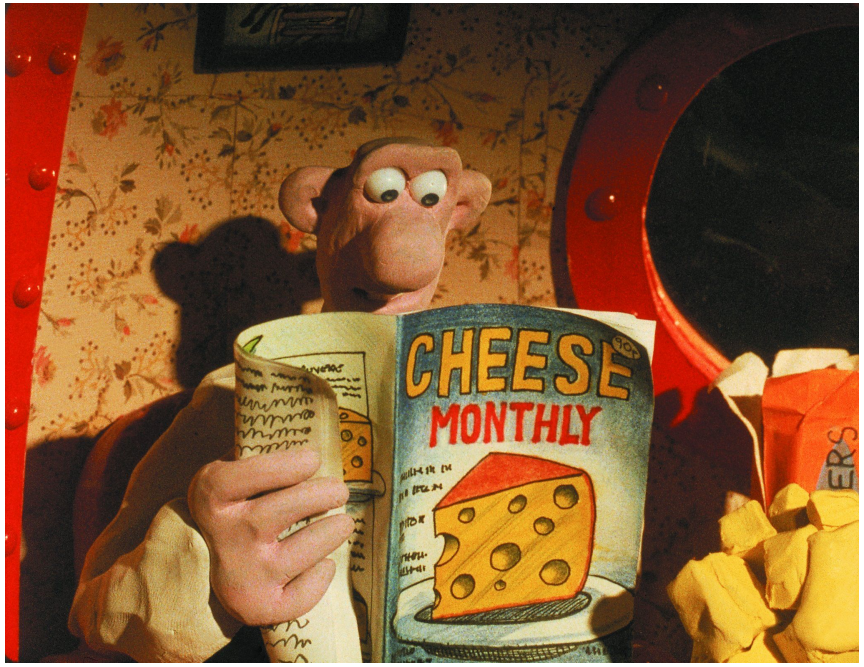


$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{threshold} \end{cases}$$

Perceptrons for decision making

- We can think about the perceptron as a device that makes decisions by weighing up evidence.
- Example: Suppose there's a cheese festival in your town. You like cheese.



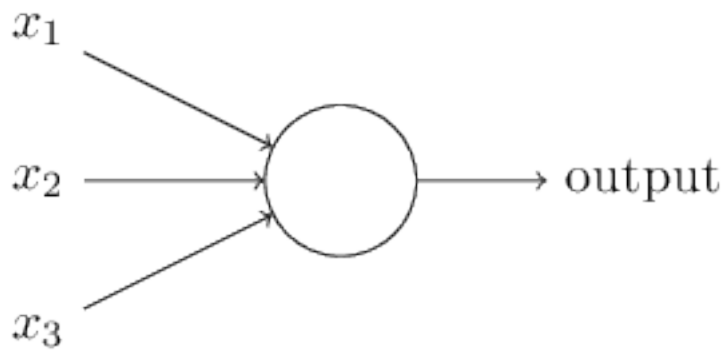


Perceptrons for decision making

- You might use 3 factors to decide whether to go.

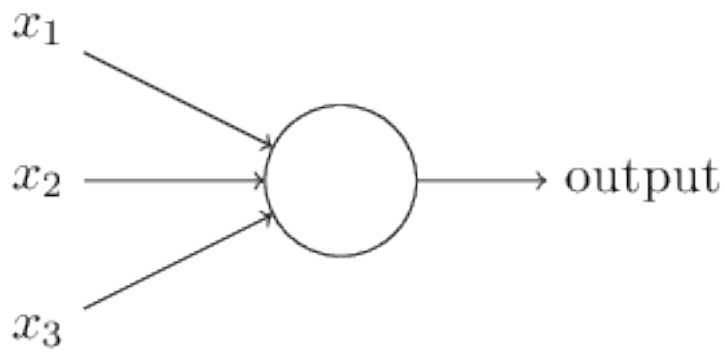
1. Is the weather good?
2. Can your loyal companion come with you?
3. Is the festival near public transit?

- These can be the binary input values to a perceptron



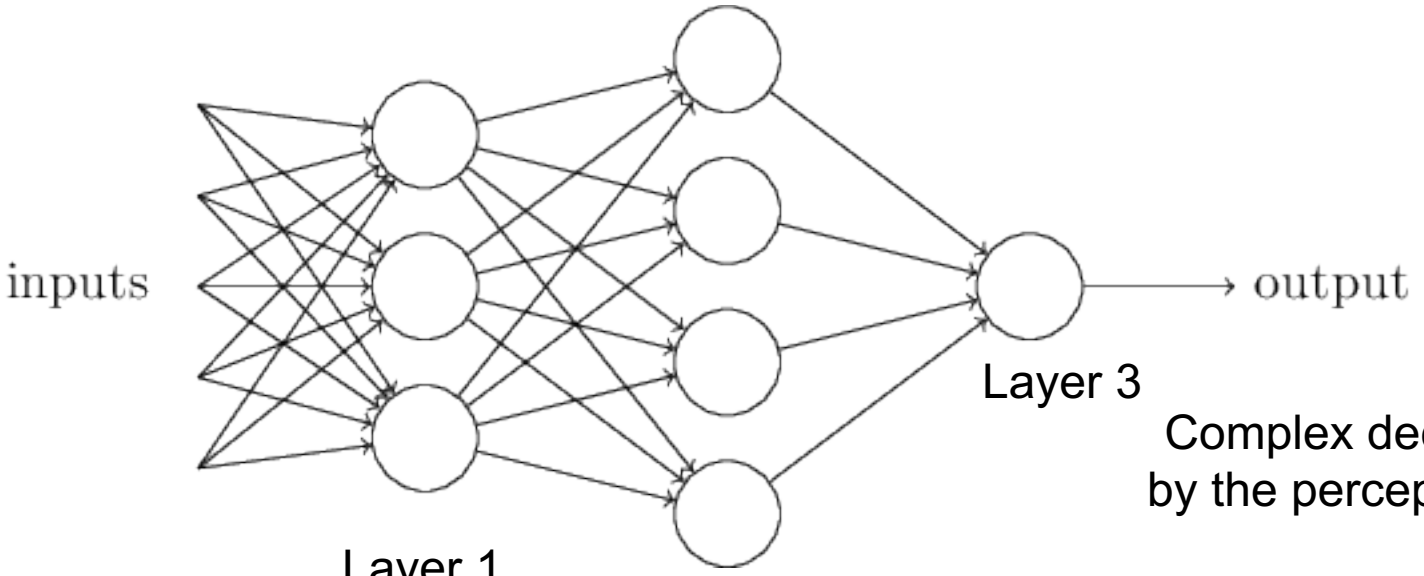
Perceptrons for decision making

- By varying weights and the threshold we get different models of decision making
- Example 1: $w_1 = 6$ $w_2 = 2$ $w_3 = 2$, threshold = 5
- Example 2: $w_1 = 6$ $w_2 = 2$ $w_3 = 2$, threshold = 3



Perceptrons for decision making

- A complex network of perceptrons could make quite subtle decisions:



1st layer makes 3 simple decisions by weighing the input evidence

2nd layer makes a decision by weighing up the results from the 1st layer of decision-making

Complex decisions can be made by the perceptron in the third layer

Weights, bias and dot products

- Two notational changes simplify the way that perceptrons are described.
- The first change is to replace the weighted sum as a dot product

$$w \cdot x \equiv \sum_j w_j x_j$$

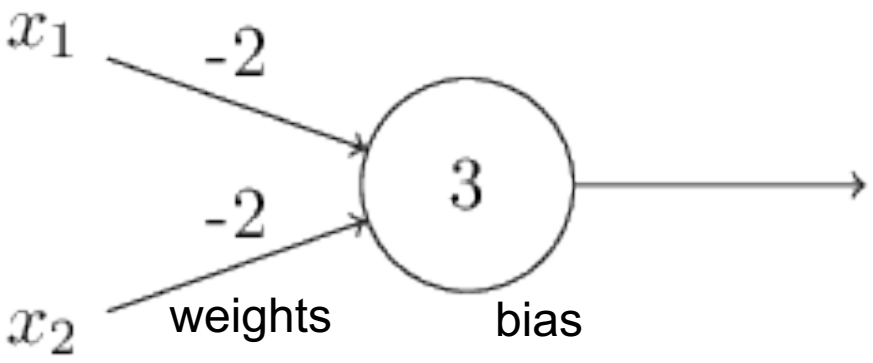
- The second change is to move the threshold to the other side of the inequality, and to replace it by a *bias*, b≡−threshold

$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{threshold} \end{cases}$$

$$\text{output} = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases}$$

Decision making OR logical functions

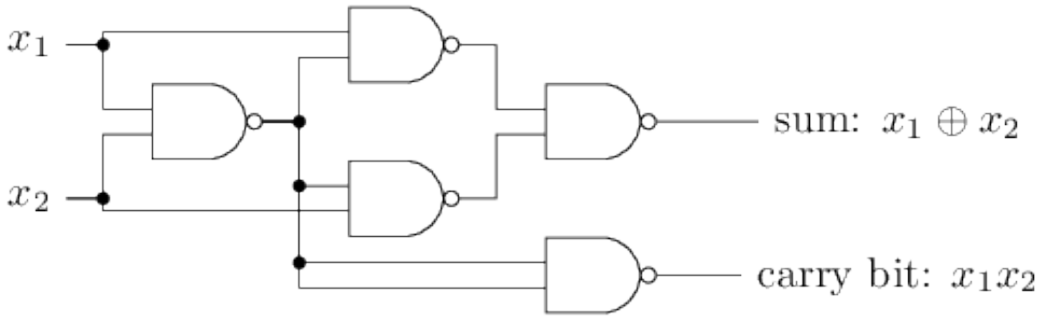
- Perceptrons can be used is to compute logical functions like AND, OR and NAND
- Example:

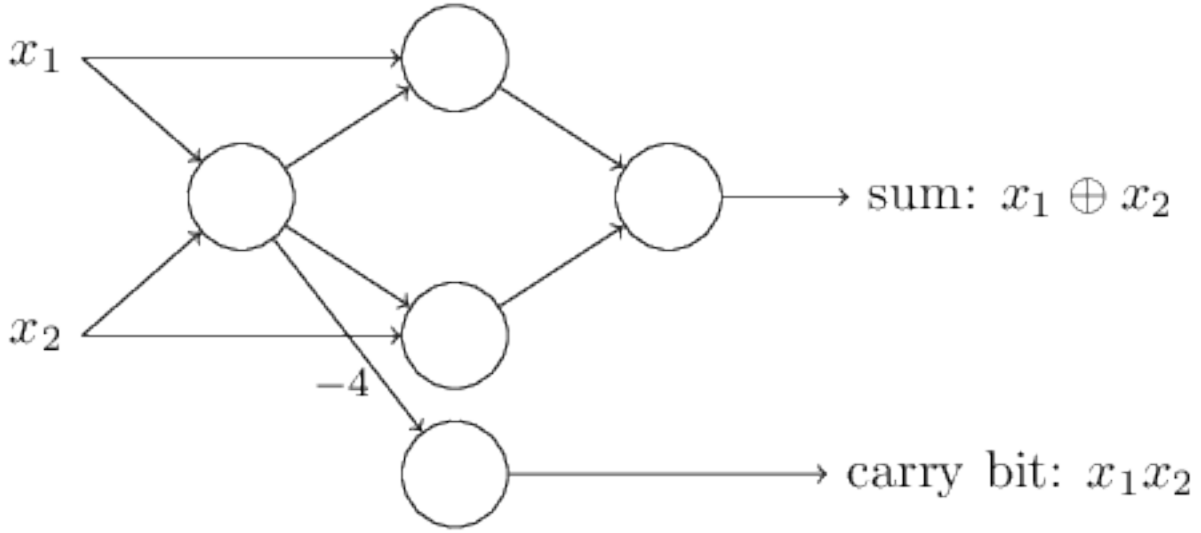


Input	Weighted sum	Output
00	-2*0 + -2*0 +3 = 3	1
10 or 01	-2*1 + -2*0 +3 = 1	1
11	-2*1 + -2*1 +3 = -1	0

Logical functions

- Networks of perceptrons to compute *any* logical function
- We can build any computation up out of NAND gates.
- For example, a circuit which adds two bits x_1 and x_2





All unlabeled weights are -2, all biases =3.

Power of Perceptrons

- Networks of Perceptrons are universal for computation, like NAND gates
- Perceptrons can be as powerful as any other computing device!

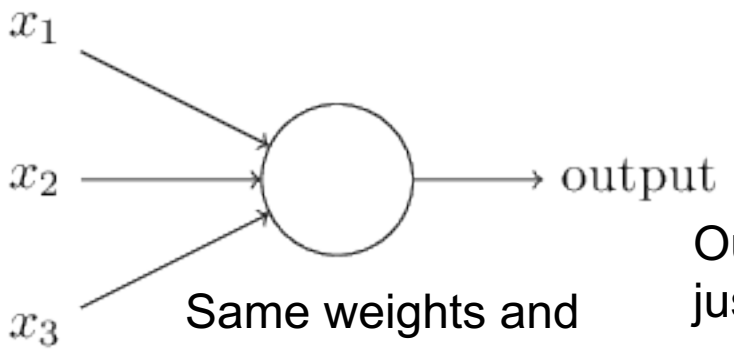
- We can devise *learning algorithms* to automatically tune the weights and biases of a network of artificial neurons
- Instead of laying out a circuit of NAND and other gates, neural networks can simply learn to solve problems

Sigmoid neurons

- Problem: a small change in the weights or bias of any single perceptron in the network can causes the output to completely flip from 0 to 1.
- Solution: sigmoid neuron

Perceptron

$$\text{output} = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases}$$



Same weights and bias as perceptron

Output is no longer just 1 or 0.

Sigmoid neuron

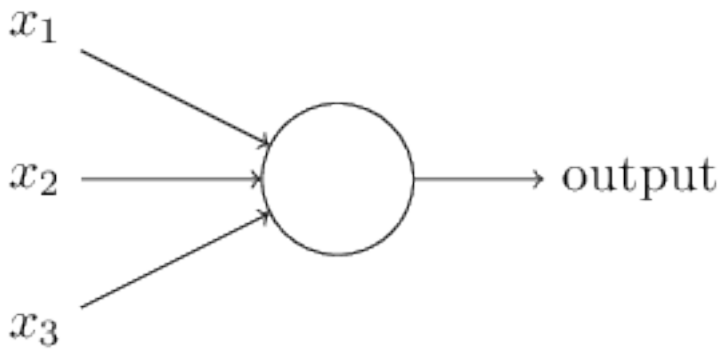
output = $\sigma(w \cdot x + b)$

Sigmoid function:

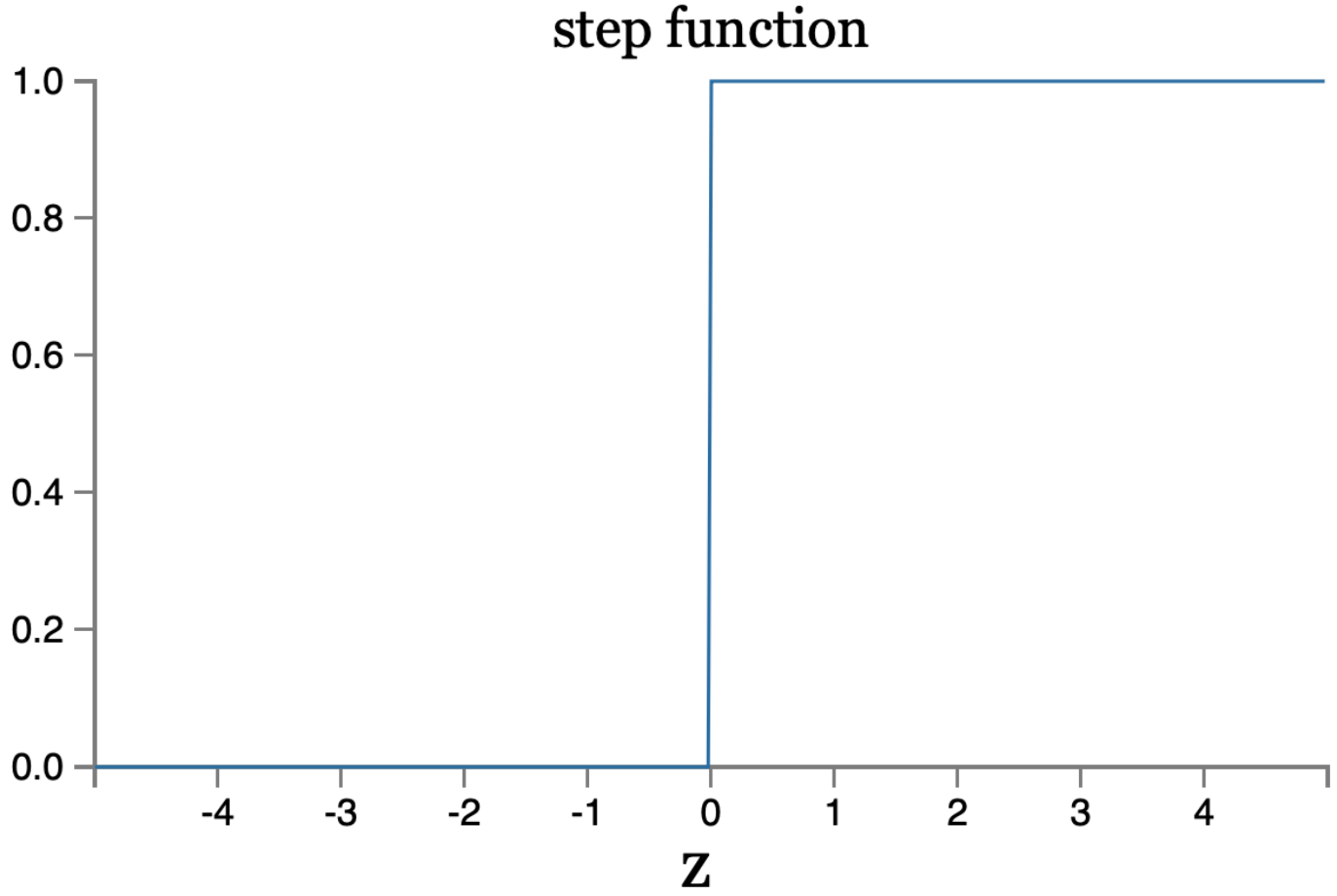
$$\sigma(z) \equiv \frac{1}{1 + e^{-z}}$$

Inputs: any real-valued number

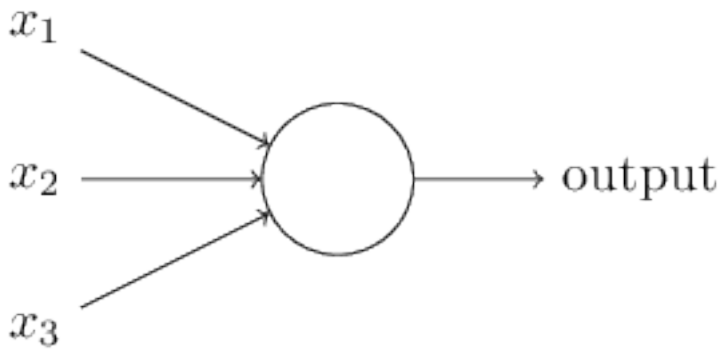
Perceptron



$$\text{output} = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases}$$

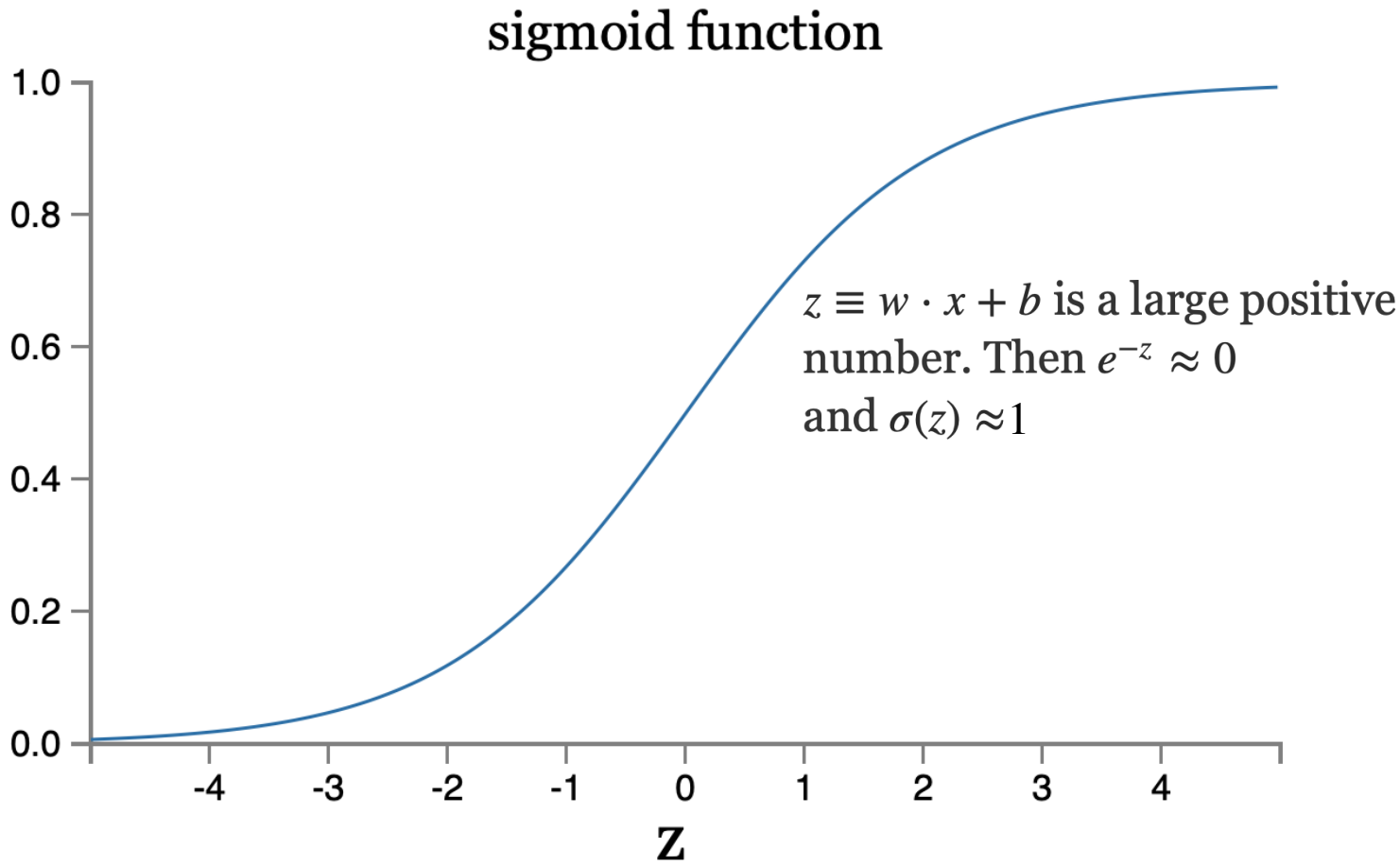


Sigmoid neurons



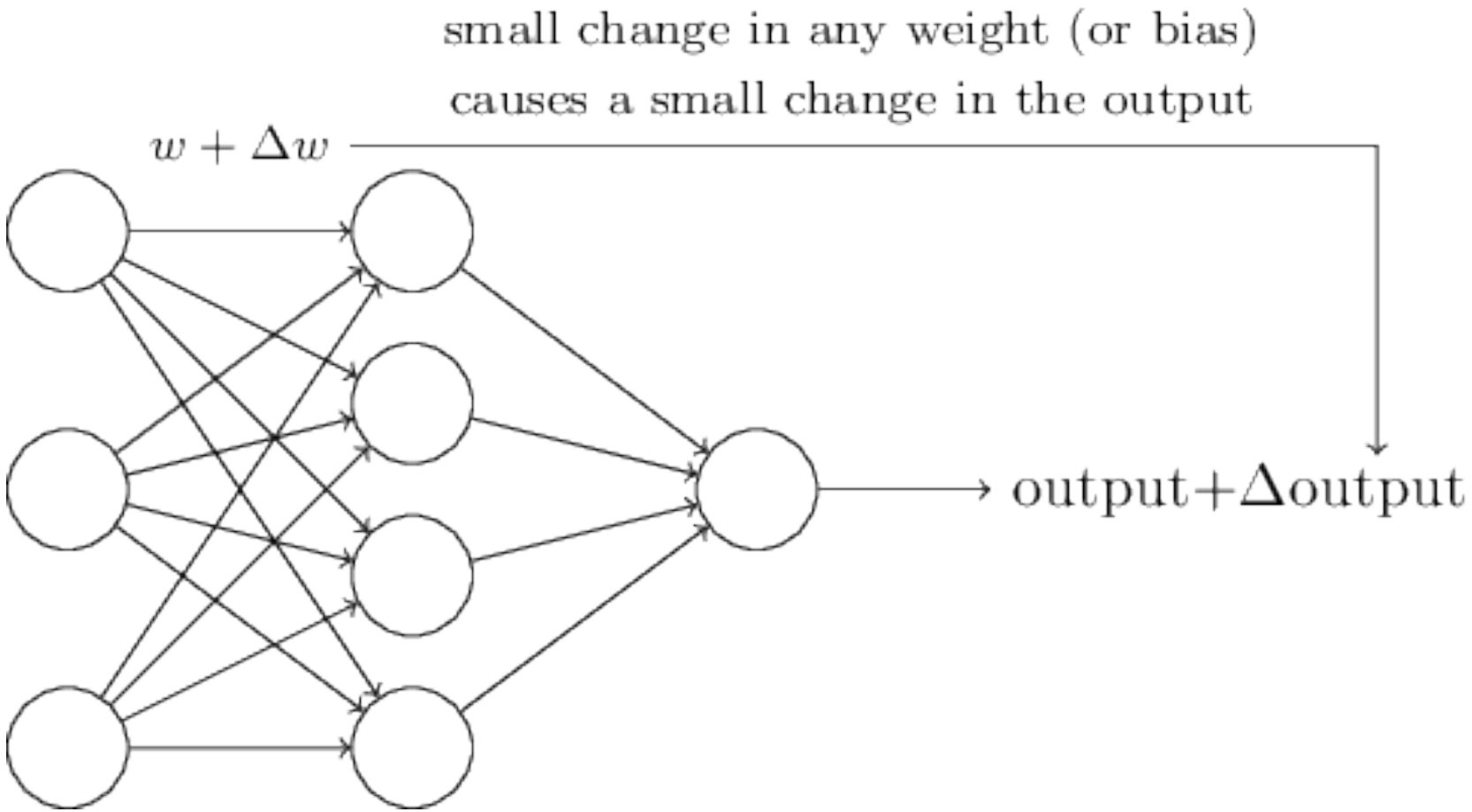
$$z \equiv w \cdot x + b$$

$$\sigma(z) \equiv \frac{1}{1 + e^{-z}}$$



$z = w \cdot x + b$ is very negative.
Then $e^{-z} \to \infty$, and $\sigma(z) \approx 0$

Sigmoid neurons are better for training



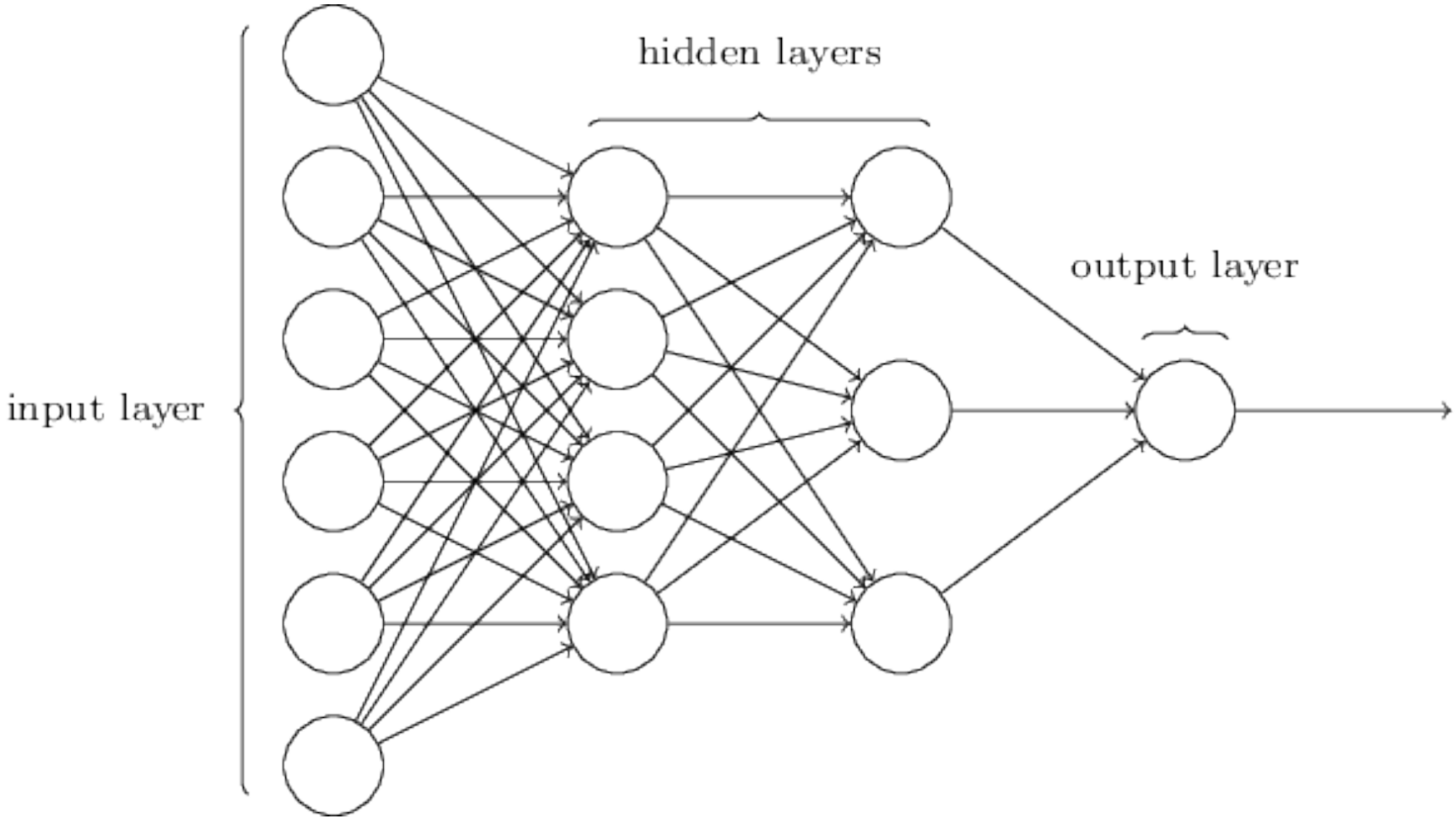
Smoothness is crucial

- Smoothness of σ means that small changes in the weights w_j and in the bias b will produce a small change the output from the neuron

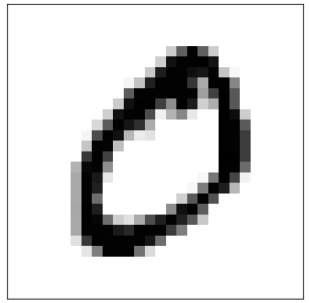
$$\Delta \text{output} \approx \sum_j \frac{\partial \, \text{output}}{\partial w_j} \Delta w_j + \frac{\partial \, \text{output}}{\partial b} \Delta b$$

- Δoutput is a *linear function* of the changes Δwj and Δb
- This makes it easy to choose small changes in the weights and biases to achieve any desired small change in the output

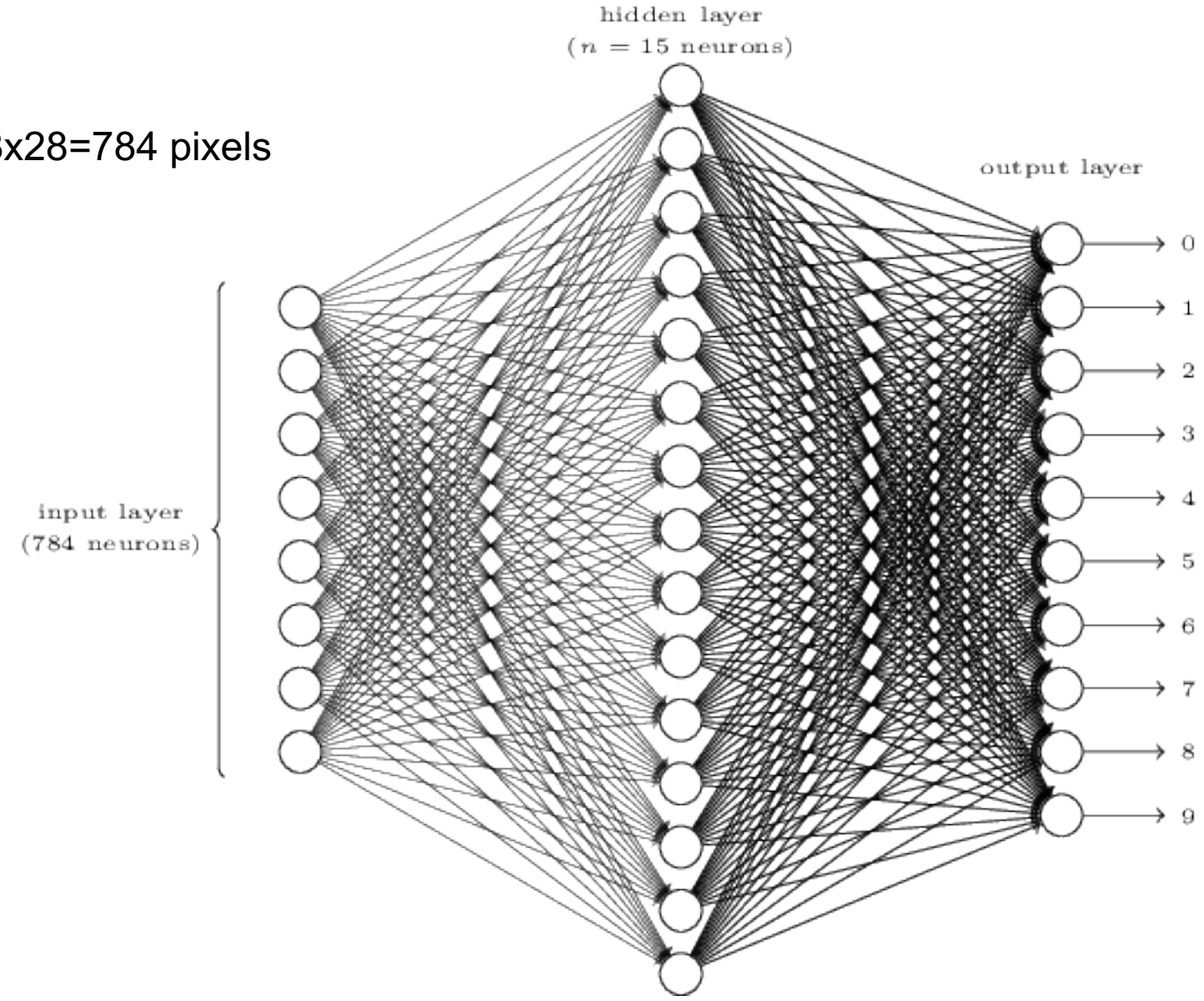
Neural Net Architecture



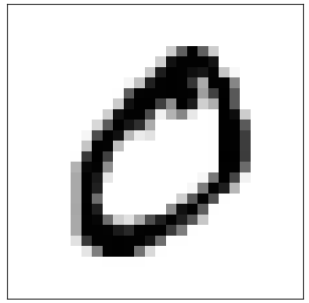
Neural Net Architecture



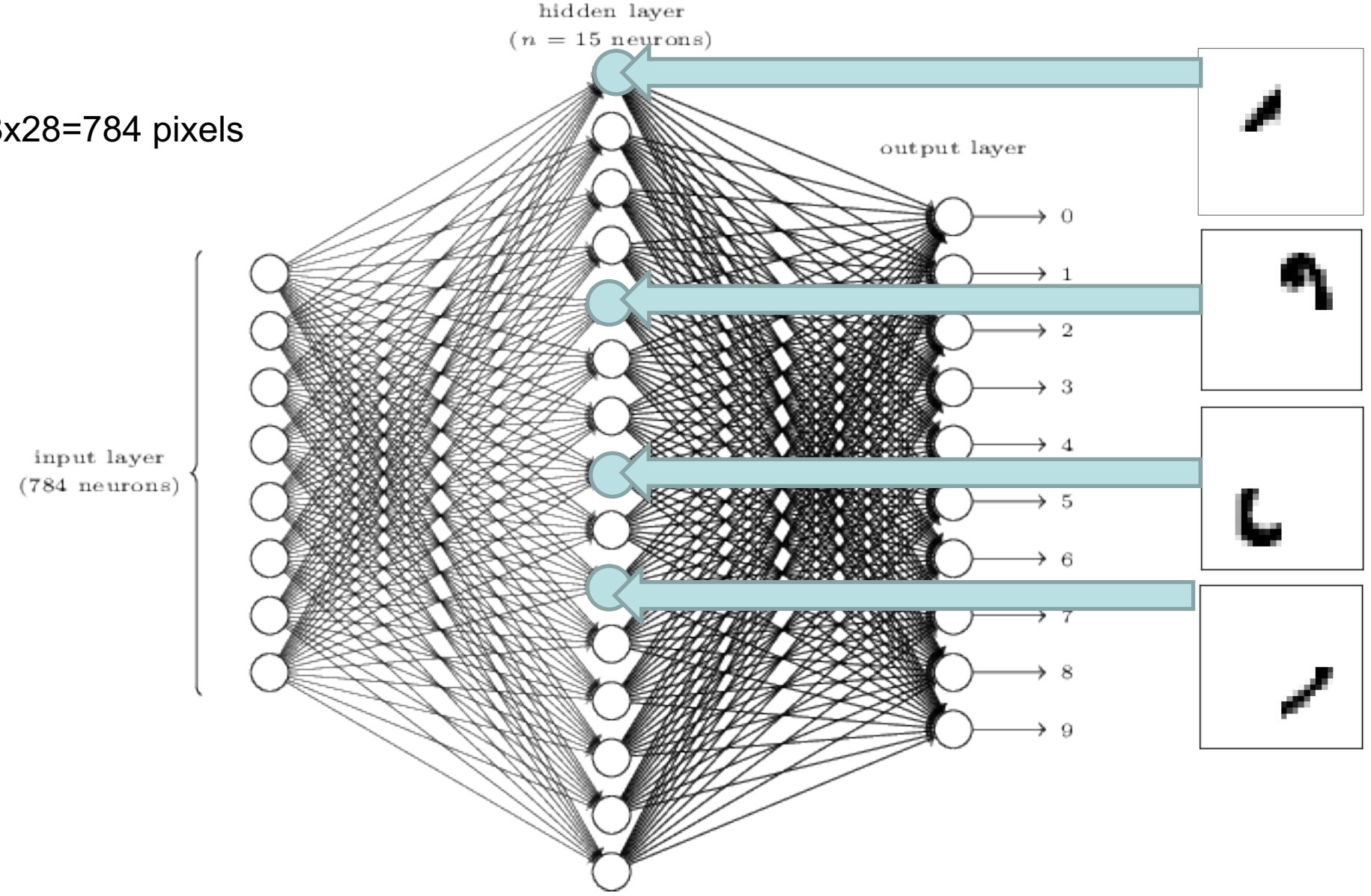
28x28=784 pixels



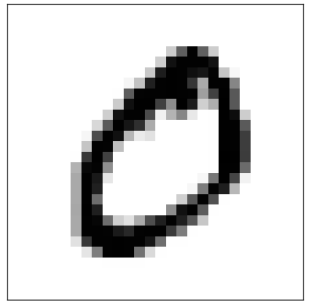
Neural Net Architecture



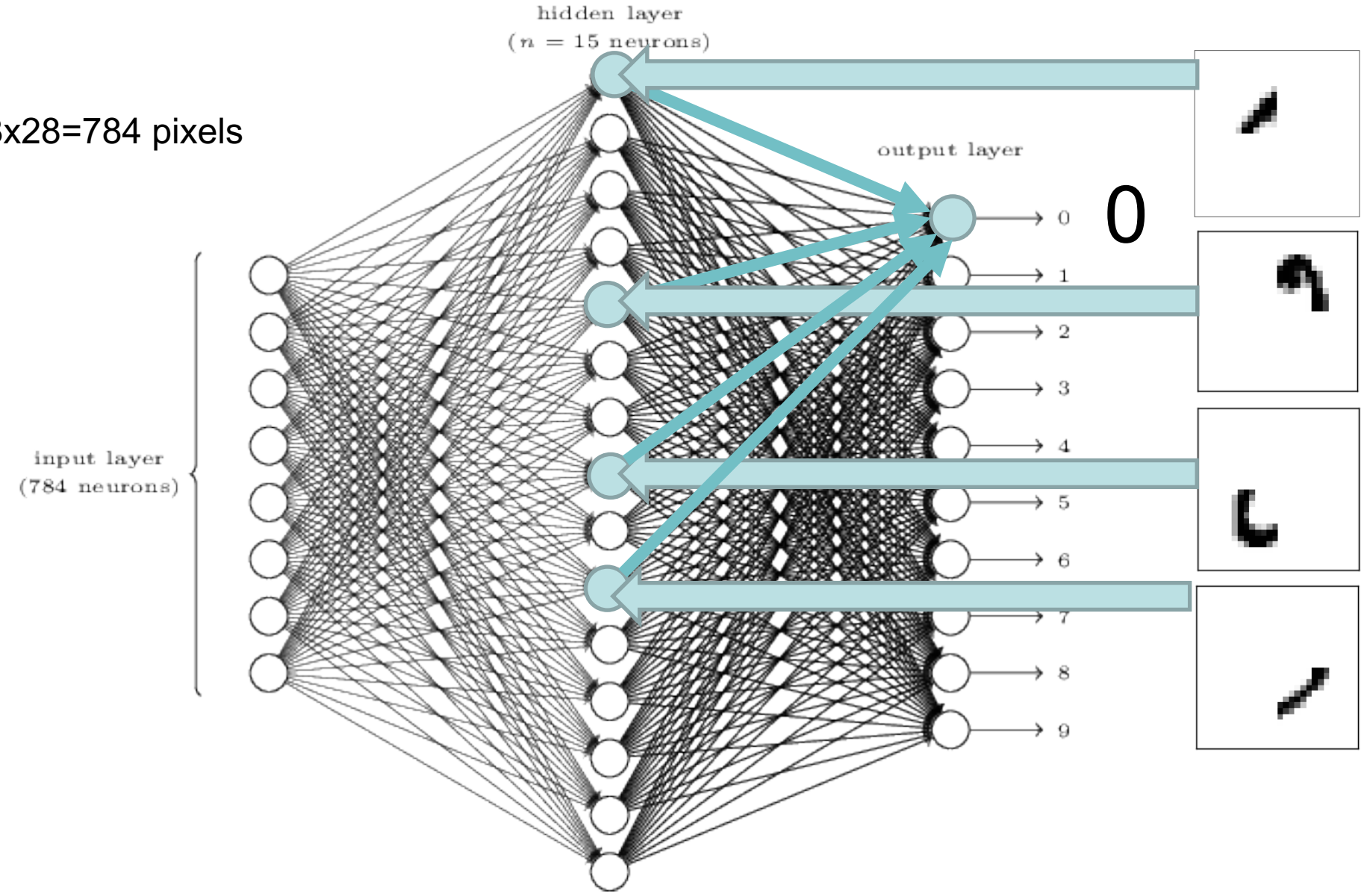
28x28=784 pixels



Neural Net Architecture



28x28=784 pixels



Cost Function

- AKA Loss function or Objective function
- Here's a common one called "Mean Squared Error"

$$C(w, b) \equiv \frac{1}{2n} \sum_x \|y(x) - a\|^2$$

Cost

weights, biases

n = number of training examples

y(x) = correct answer

a = vector of system's outputs

Cost Function

- AKA Loss function or Objective function
- Here's a common one called "Mean Squared Error"

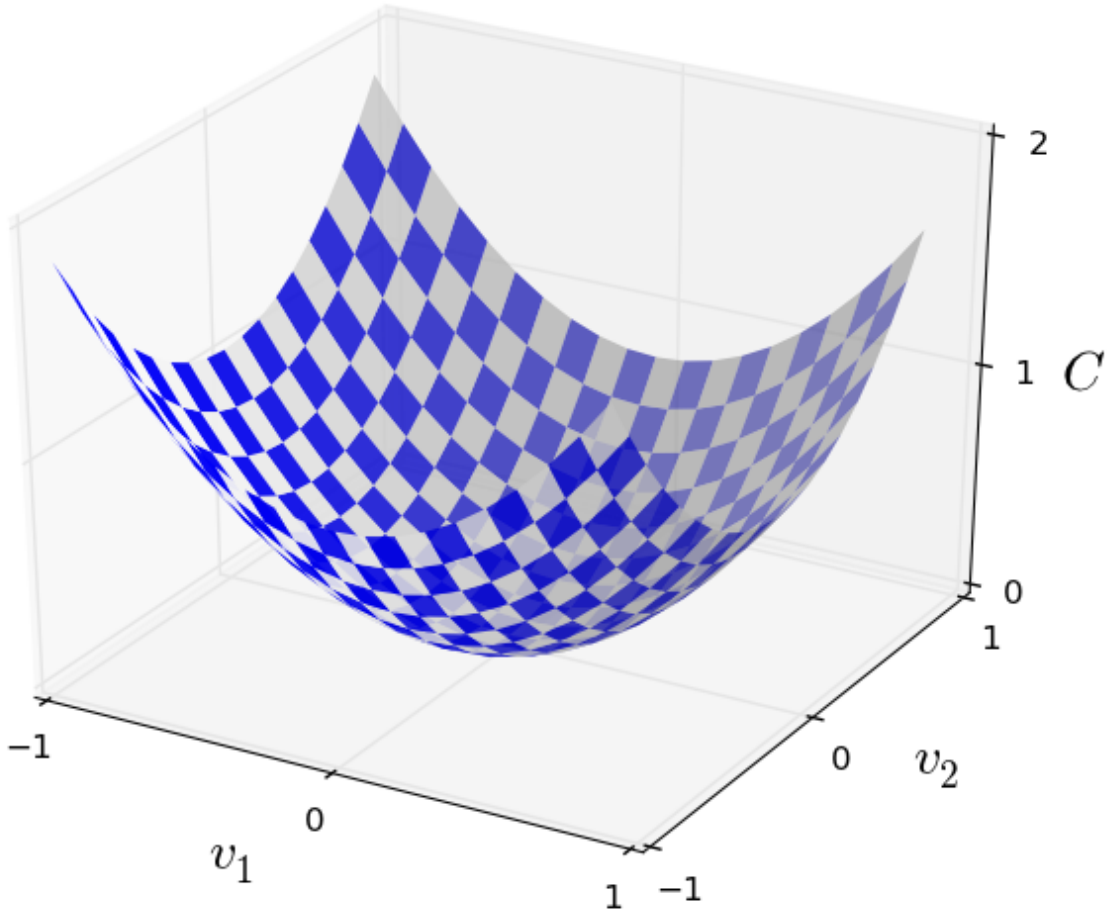
$$C(w,b) \equiv \frac{1}{2n} \sum_x \|y(x) - a\|^2$$

Cost | weights, biases | n = number of training examples | y(x) = correct answer | a = vector of system's outputs

$$\|\boldsymbol{x}\|_2 := \sqrt{x_1^2 + \cdots + x_n^2}$$

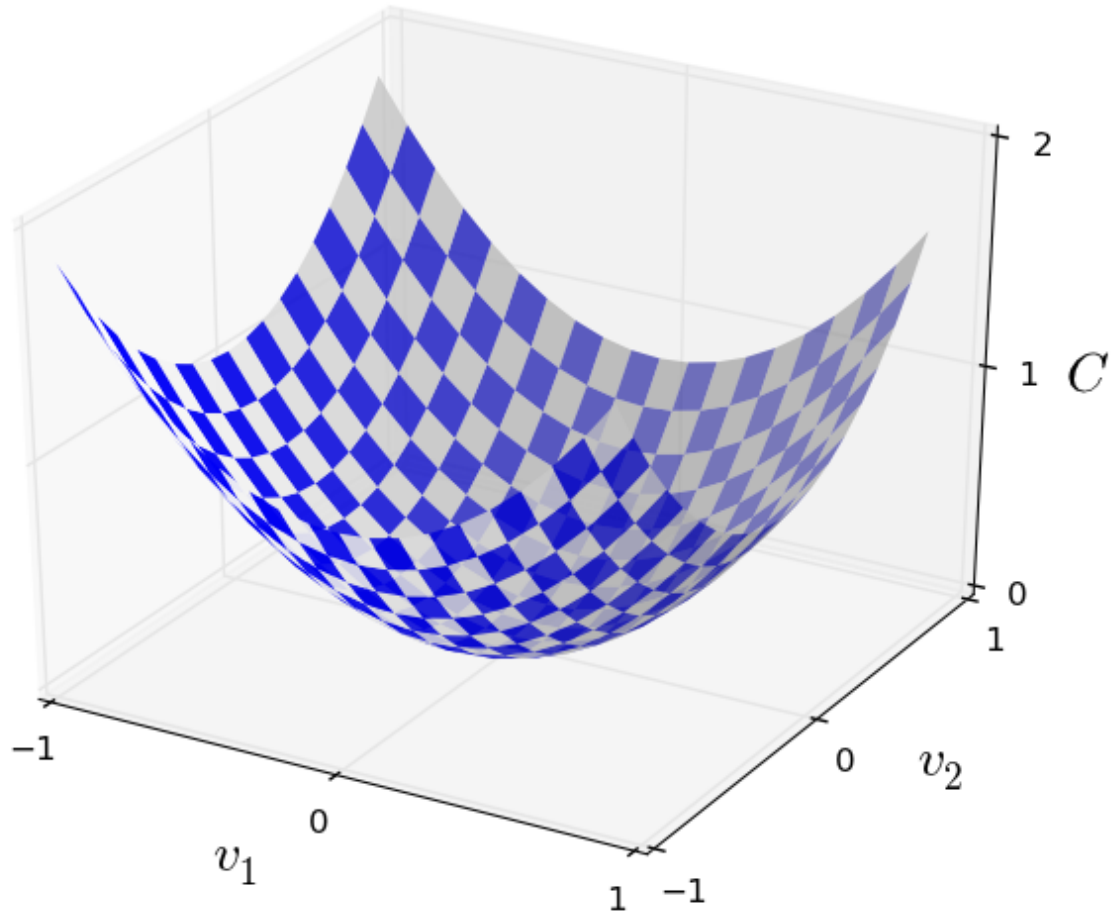
	y(x)	a
0	0	0.01
1	0	0.001
2	0	
3	0	
4	0	
5	0	
6	1	
7	0	
8	0	
9	0	

Minimize the loss function



Gradient Descent

Minimize the loss function



$$\Delta C \approx \frac{\partial C}{\partial v_1}\Delta v_1 + \frac{\partial C}{\partial v_2}\Delta v_2$$

(we want to minimize this)

vector of changes in v

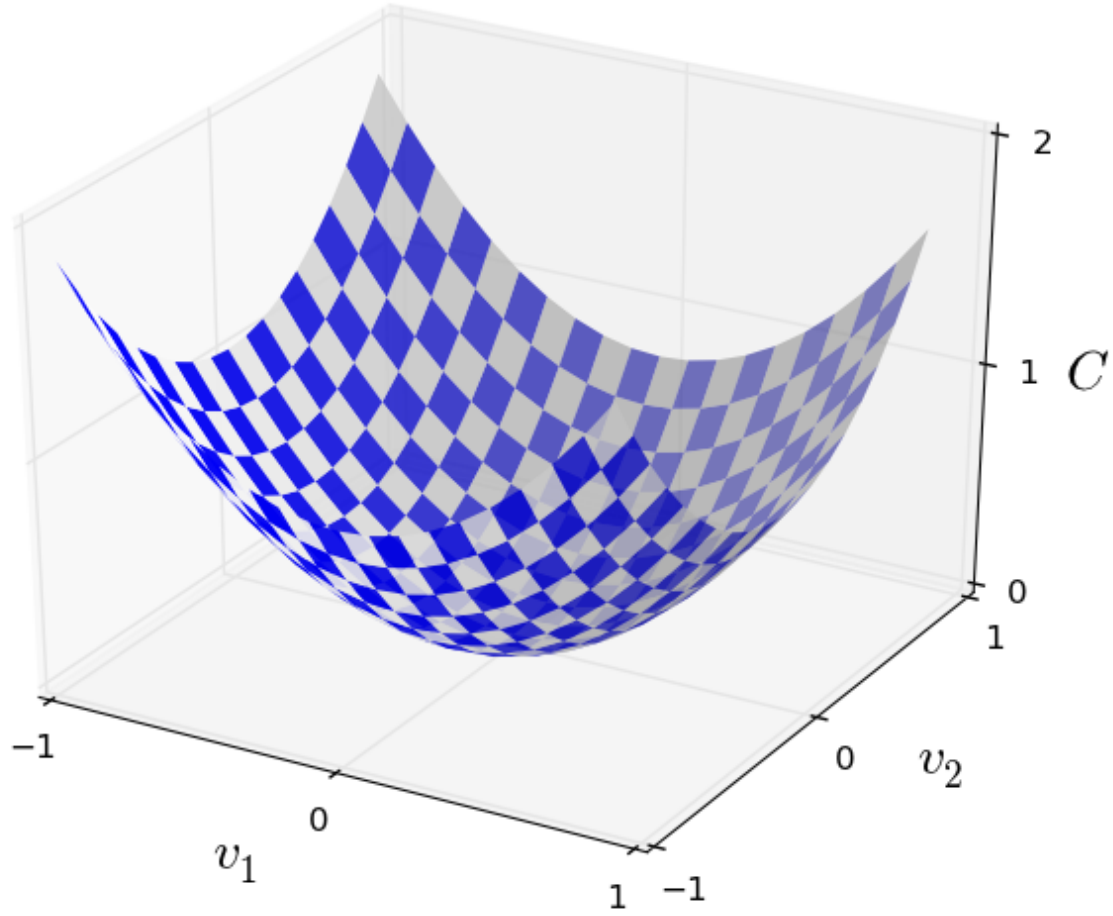
$$\Delta v \equiv (\Delta v_1, \Delta v_2)^T$$

gradient vector

$$\nabla C \equiv \left(\frac{\partial C}{\partial v_1}, \frac{\partial C}{\partial v_2}\right)^T$$

$$\Delta C \approx \nabla C \cdot \Delta v.$$

Minimize the loss function



$$\Delta v = -\eta \nabla C$$

η is a small, positive parameter known as the *learning rate*

$$\Delta C \approx \nabla C \cdot \Delta v.$$

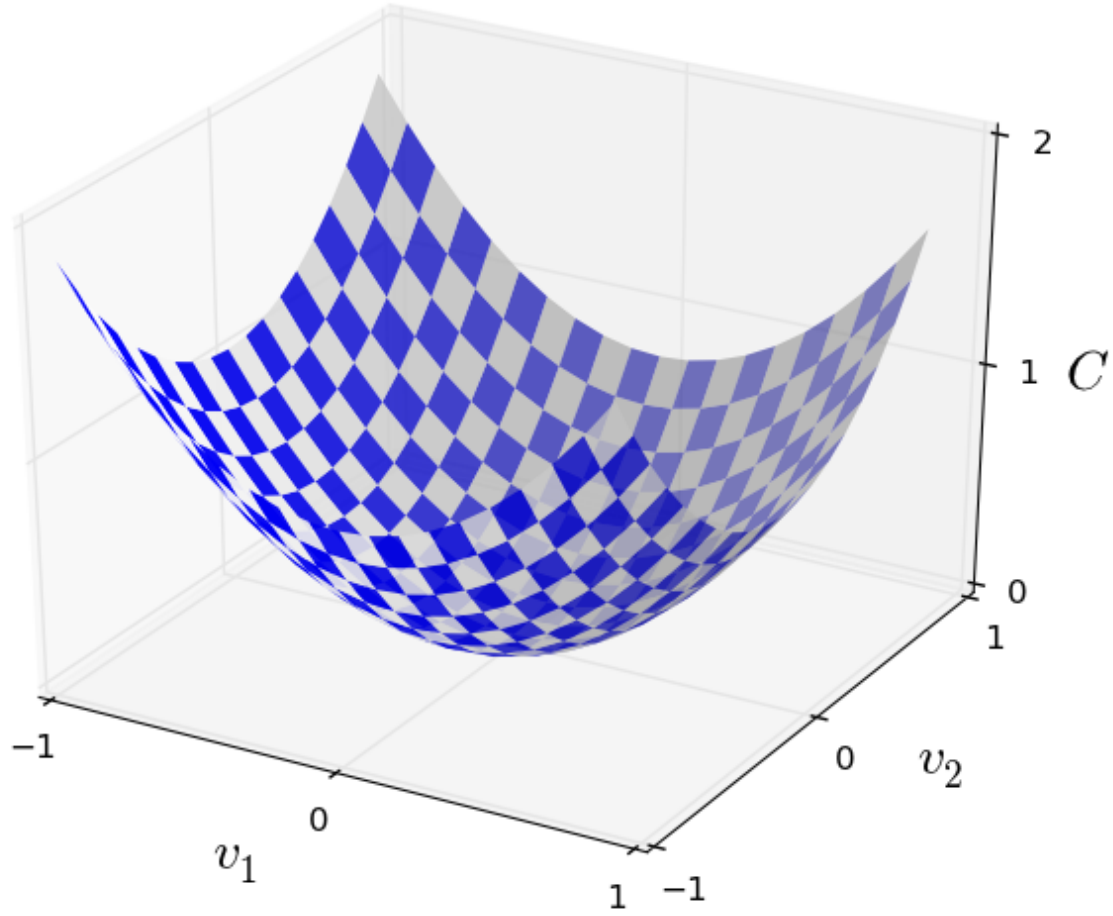
$$\Delta C \approx -\eta \nabla C \cdot \nabla C = -\eta \|\nabla C\|^2$$

$$\text{Because } \|\nabla C\|^2 \geq 0, \ \Delta C \leq 0$$

C will always decrease, if we change v according to $\Delta v = -\eta \nabla C$

$$v \to v' = v - \eta \nabla C.$$

Minimize the loss function



$$\Delta v = -\eta \nabla C$$

η is a small, positive parameter known as the *learning rate*

$$\Delta C \approx \nabla C \cdot \Delta v.$$

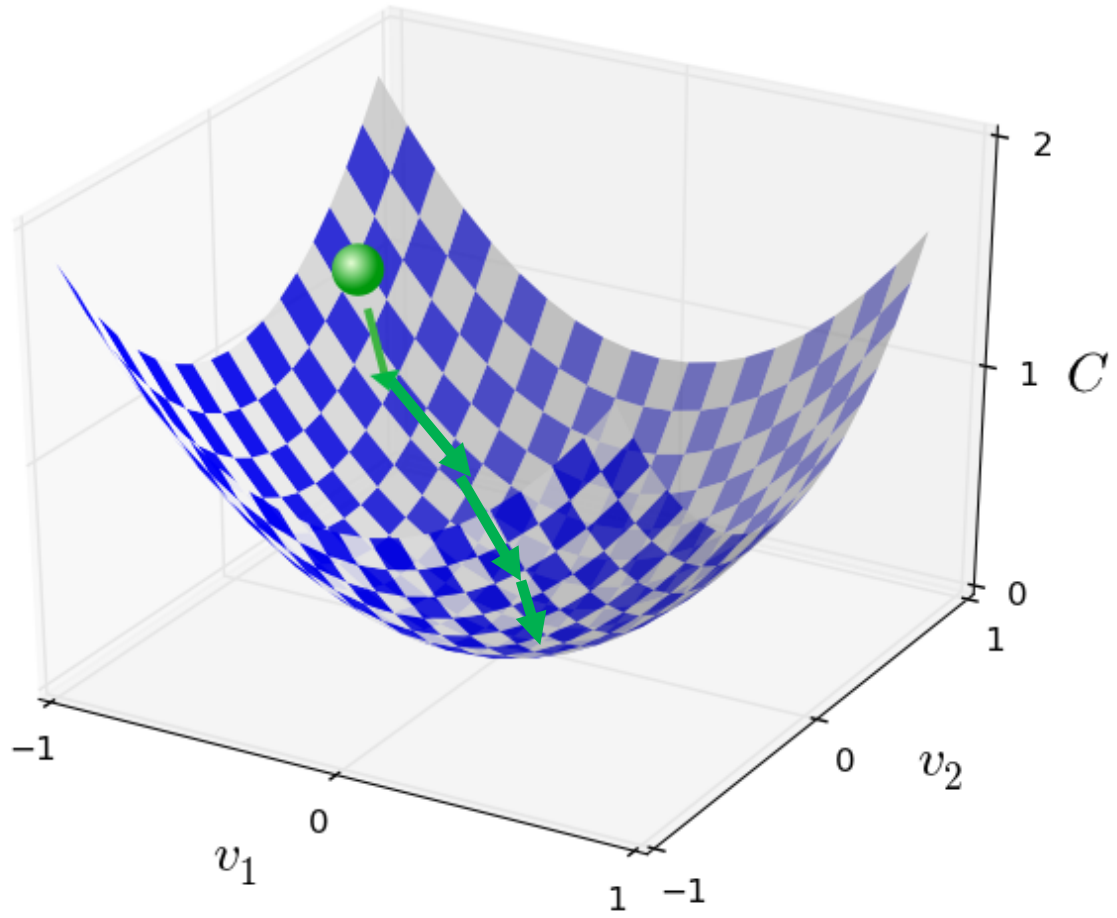
$$\Delta C \approx -\eta \nabla C \cdot \nabla C = -\eta \|\nabla C\|^2$$

$$\text{Because } \|\nabla C\|^2 \geq 0, \ \Delta C \leq 0$$

C will always decrease, if we change v according to $\Delta v = -\eta \nabla C$

$$v \to v' = v - \eta \nabla C.$$

Minimize the loss function



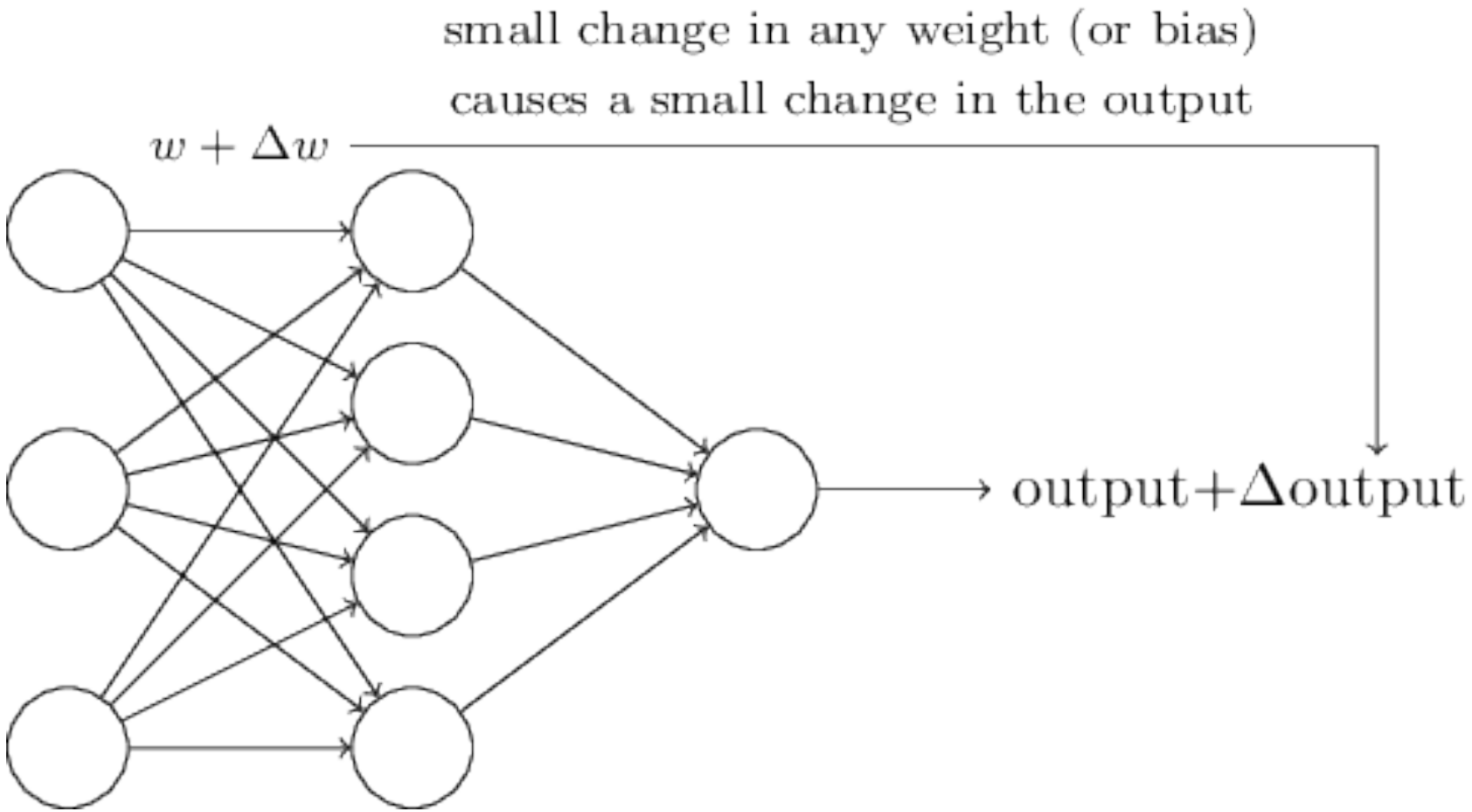
$$v \rightarrow v' = v - \eta \nabla C.$$

$$v \rightarrow v' = v - \eta \nabla C.$$

$$v \rightarrow v' = v - \eta \nabla C.$$

$$v \rightarrow v' = v - \eta \nabla C.$$

Sigmoid neurons are better for training



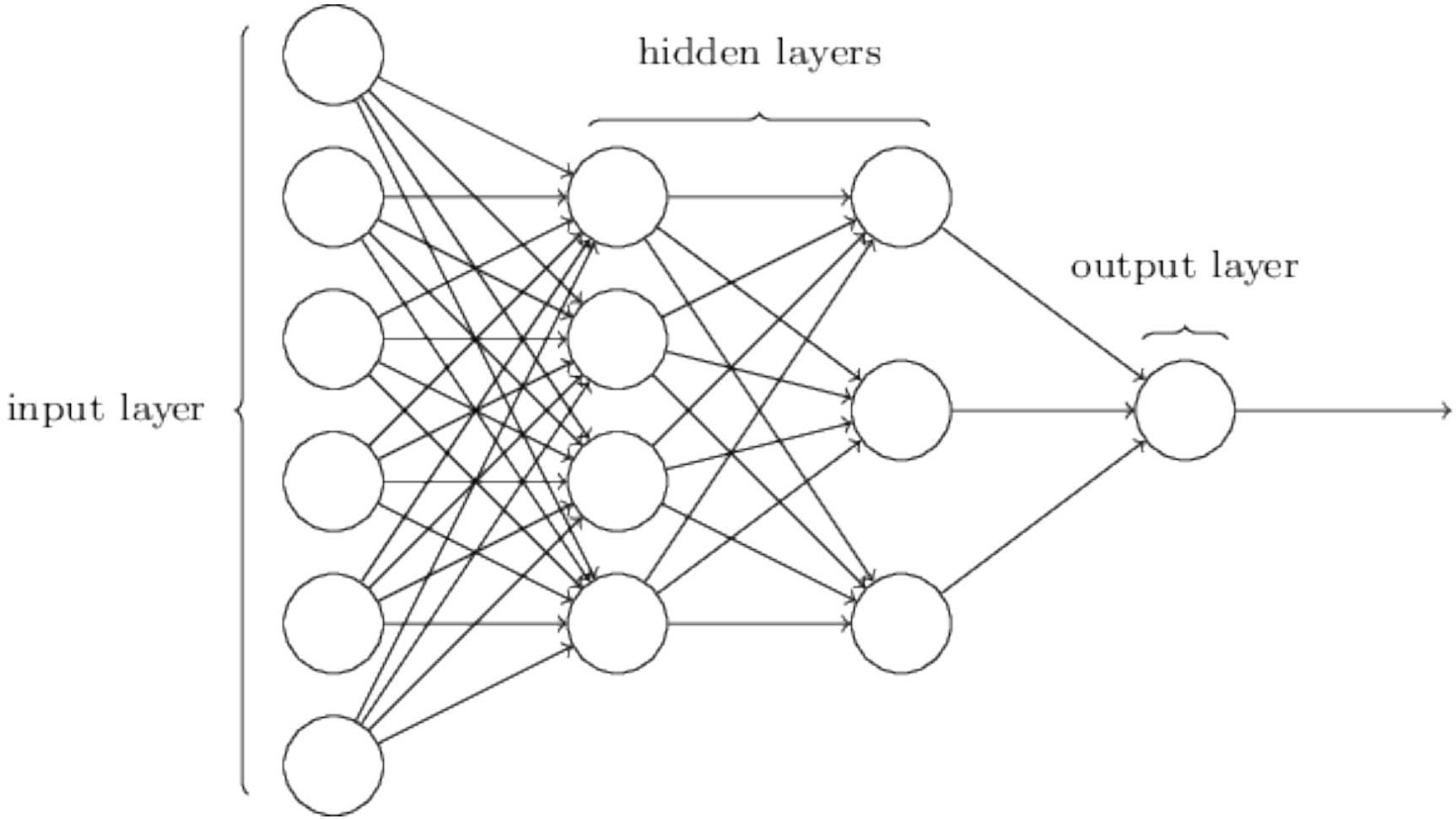
Smoothness is crucial

- Smoothness of σ means that small changes in the weights w_j and in the bias b will produce a small change the output from the neuron

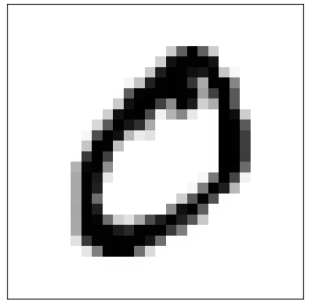
$$\Delta \text{output} \approx \sum_j \frac{\partial \, \text{output}}{\partial w_j} \Delta w_j + \frac{\partial \, \text{output}}{\partial b} \Delta b$$

- Δoutput is a *linear function* of the changes Δwj and Δb
- This makes it easy to choose small changes in the weights and biases to achieve any desired small change in the output

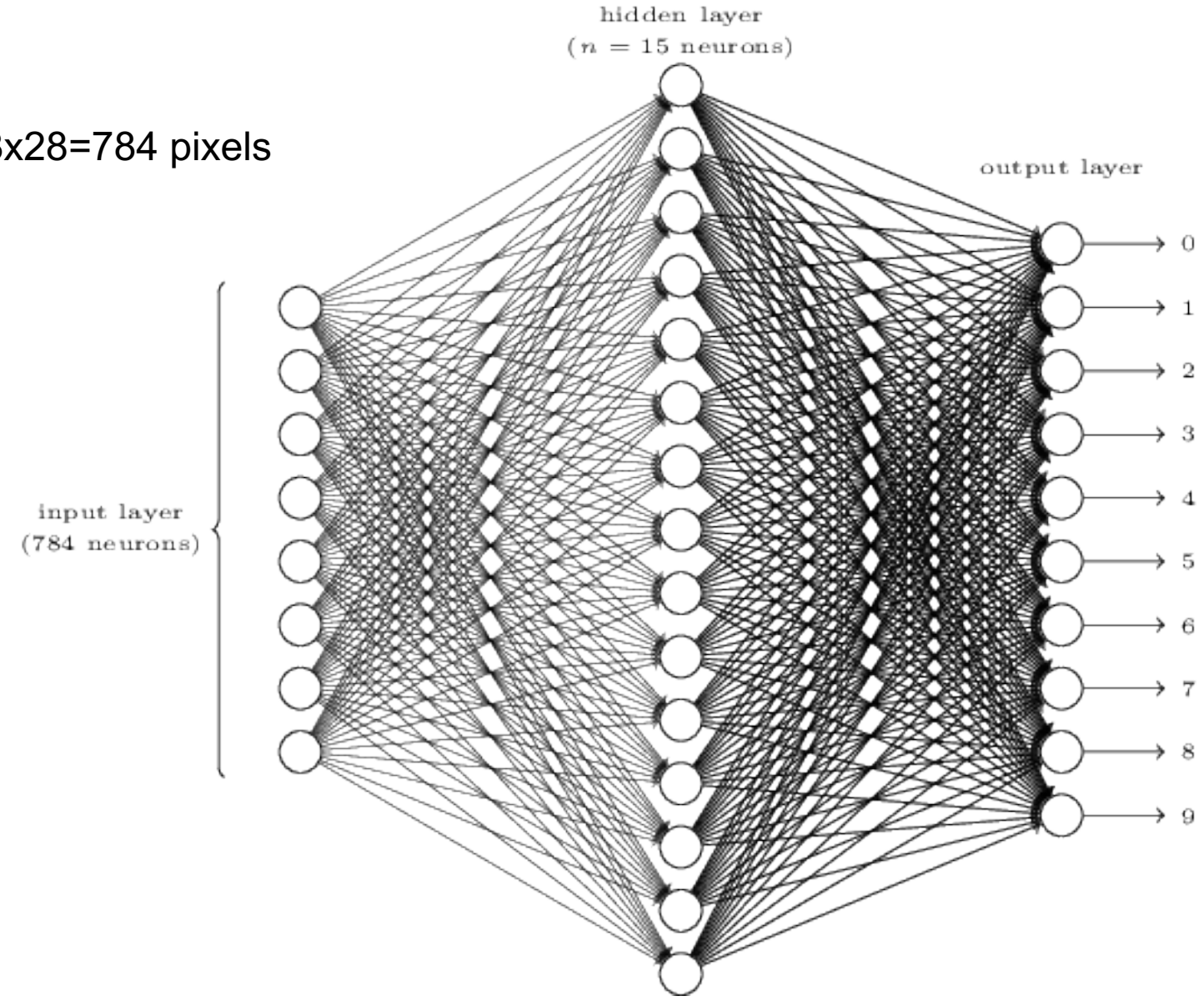
Neural Net Architecture



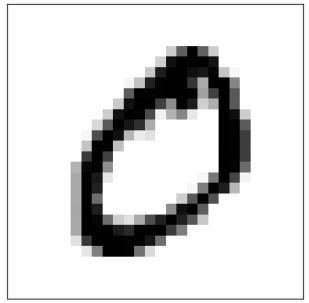
Neural Net Architecture



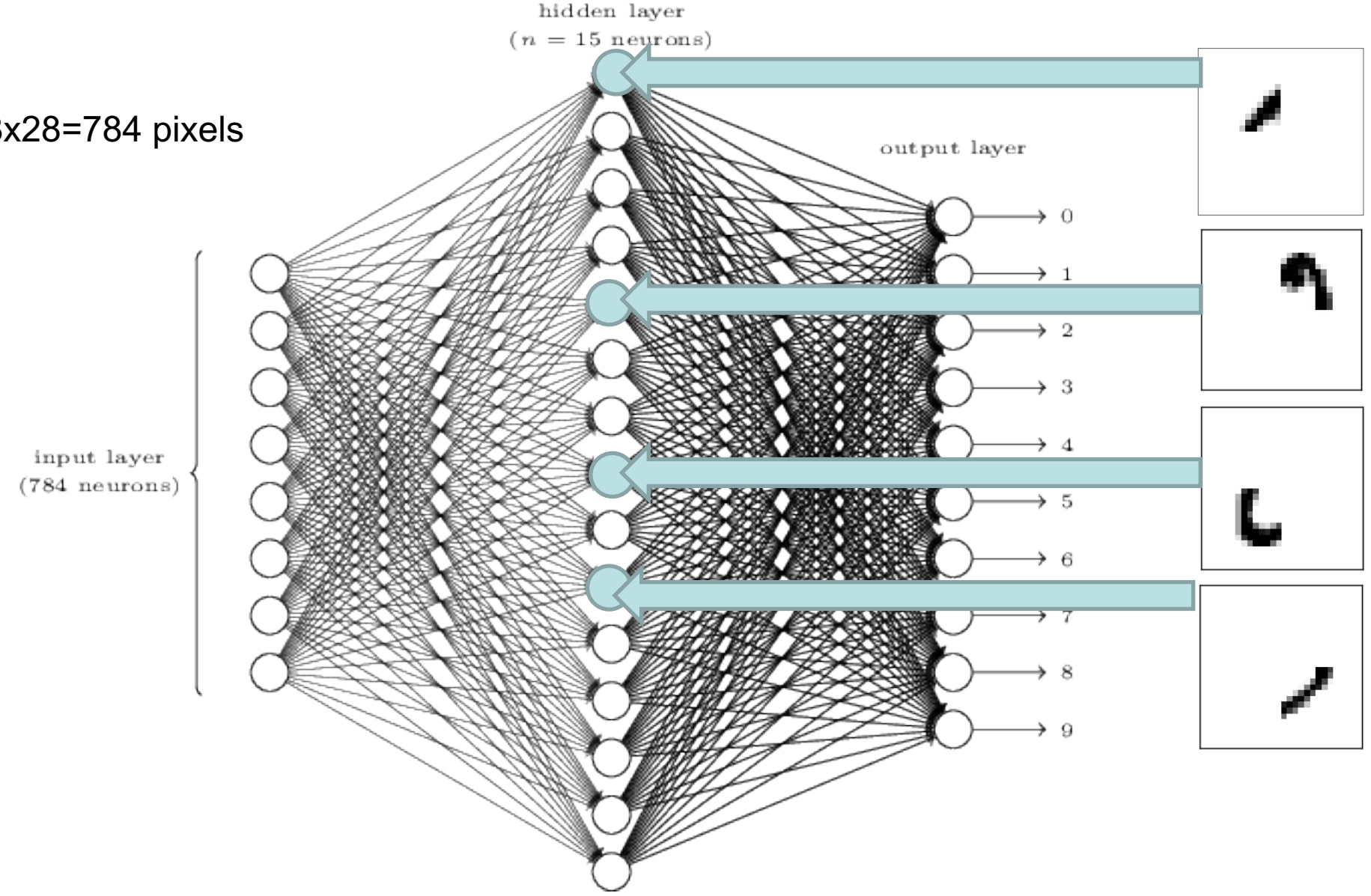
28x28=784 pixels



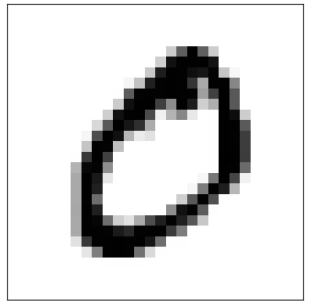
Neural Net Architecture



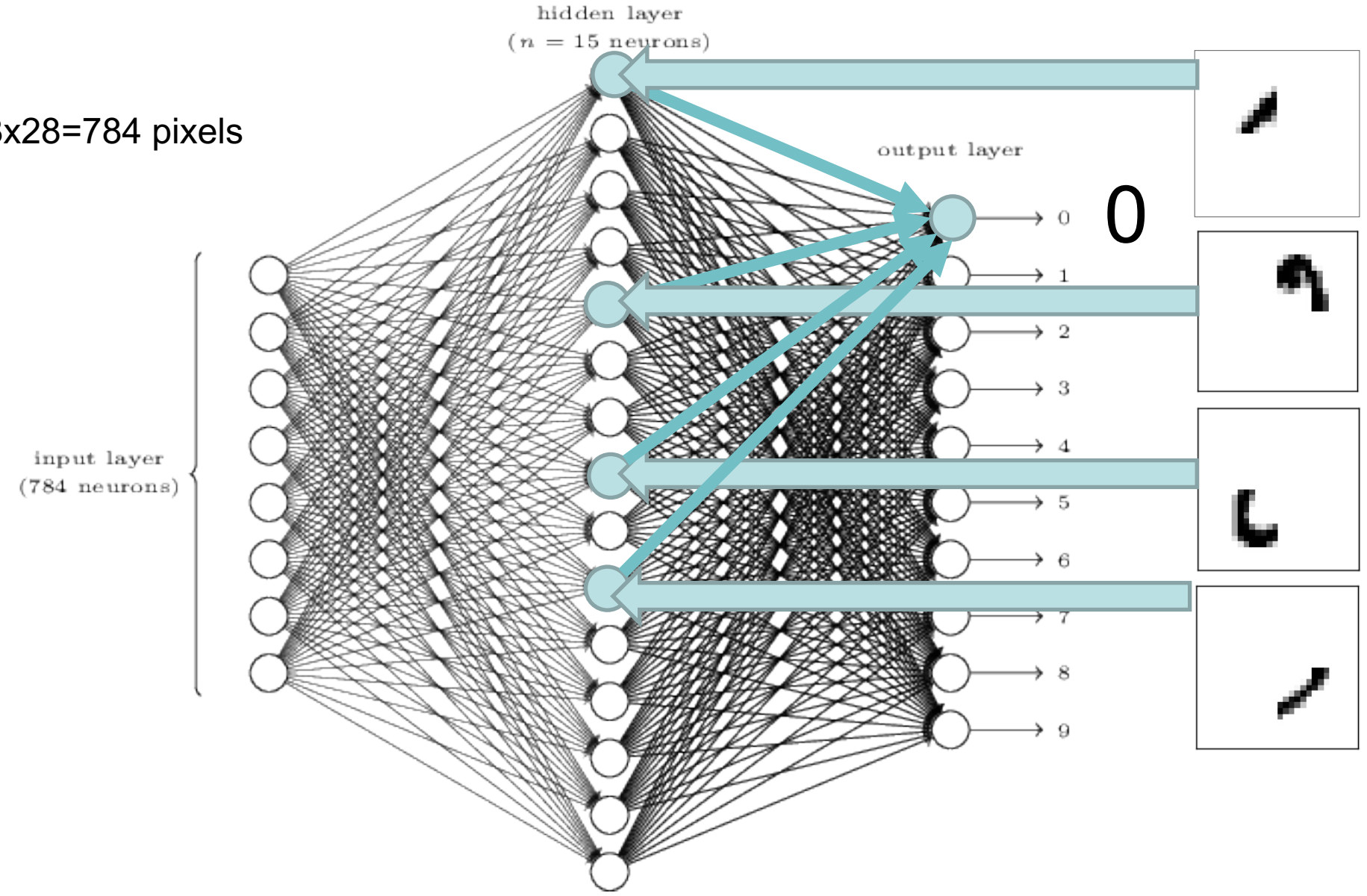
28x28=784 pixels



Neural Net Architecture



28x28=784 pixels



Cost Function

- AKA Loss function or Objective function
- Here's a common one called "Mean Squared Error"

$$C(w, b) \equiv \frac{1}{2n} \sum_x \|y(x) - a\|^2$$

Cost

weights, biases

n = number of training examples

y(x) = correct answer

a = vector of system's outputs

Cost Function

- AKA Loss function or Objective function
- Here's a common one called "Mean Squared Error"

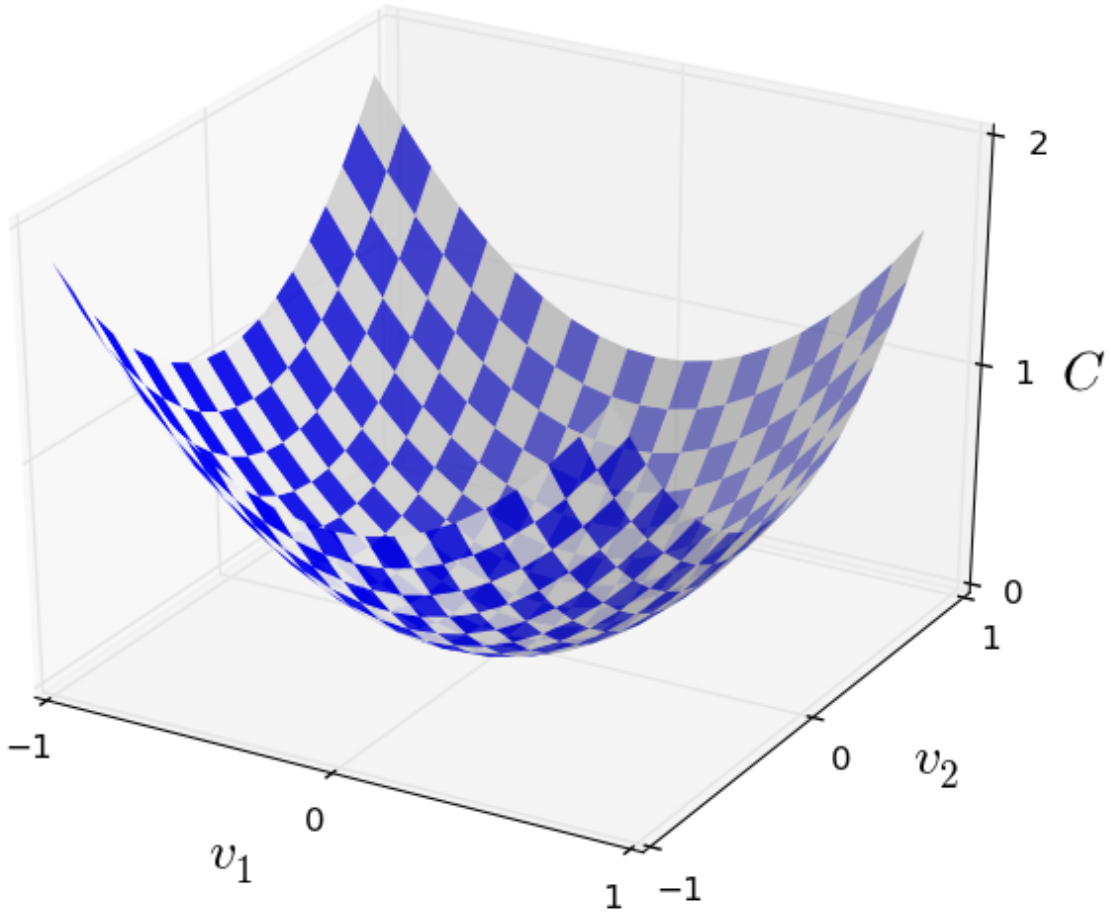
$$C(w,b) \equiv \frac{1}{2n} \sum_x \|y(x) - a\|^2$$

Cost | weights, biases | n = number of training examples | y(x) = correct answer | a = vector of system's outputs

$$\|\boldsymbol{x}\|_2 := \sqrt{x_1^2 + \cdots + x_n^2}$$

	y(x)	a
0	0	0.01
1	0	0.001
2	0	
3	0	
4	0	
5	0	
6	1	
7	0	
8	0	
9	0	

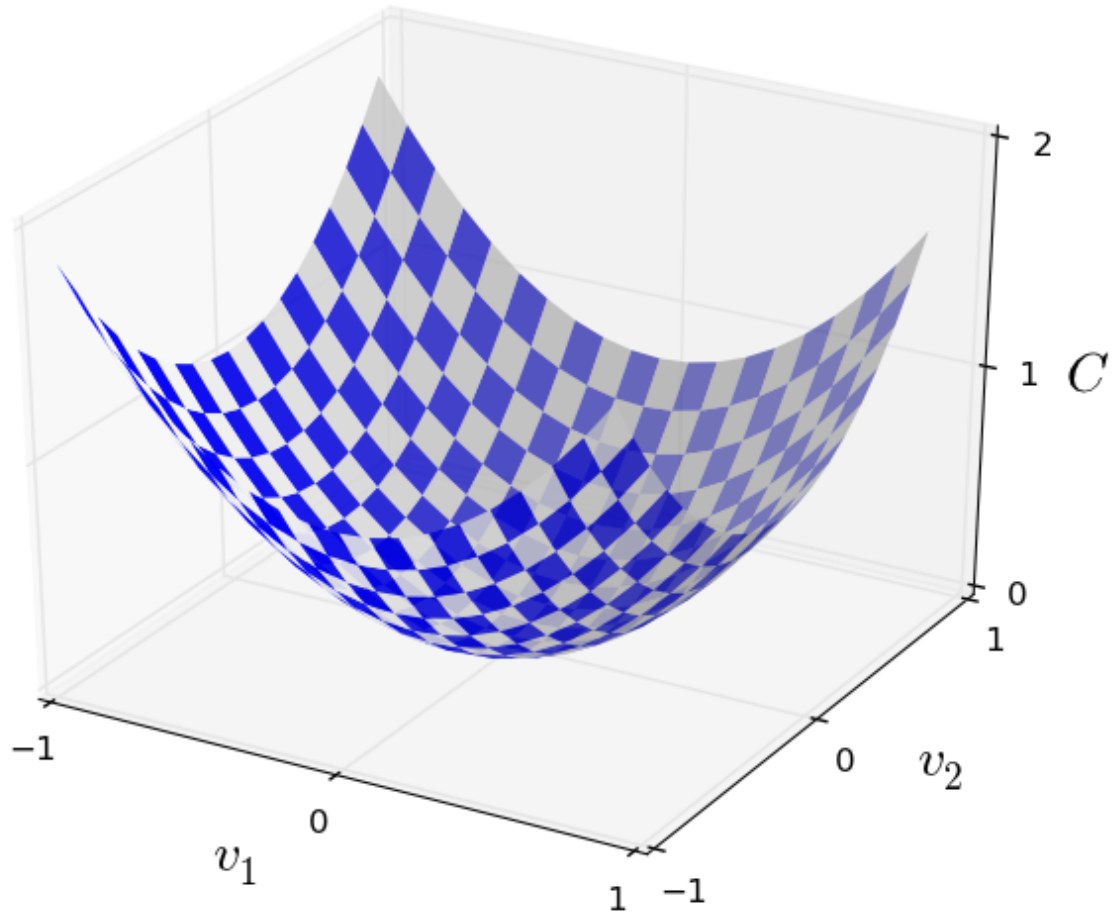
Minimize the loss function



Gradient Descent

Hill Climbing

Minimize the loss function



$$\Delta C \approx \frac{\partial C}{\partial v_1}\Delta v_1 + \frac{\partial C}{\partial v_2}\Delta v_2$$

(we want to minimize this)

vector of changes in v

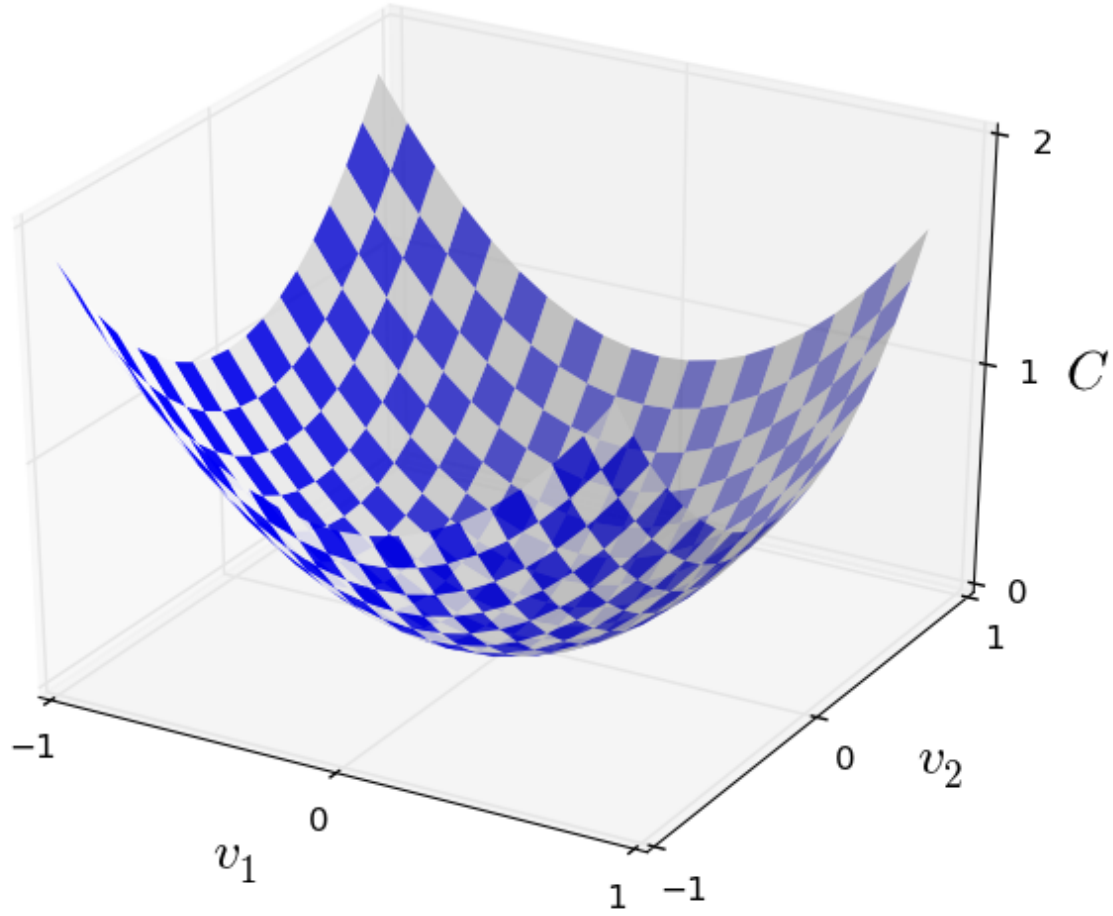
$$\Delta v \equiv (\Delta v_1, \Delta v_2)^T$$

gradient vector

$$\nabla C \equiv \left(\frac{\partial C}{\partial v_1}, \frac{\partial C}{\partial v_2}\right)^T$$

$$\Delta C \approx \nabla C \cdot \Delta v.$$

Minimize the loss function



$$\Delta v = -\eta \nabla C$$

η is a small, positive parameter known as the *learning rate*

$$\Delta C \approx \nabla C \cdot \Delta v.$$

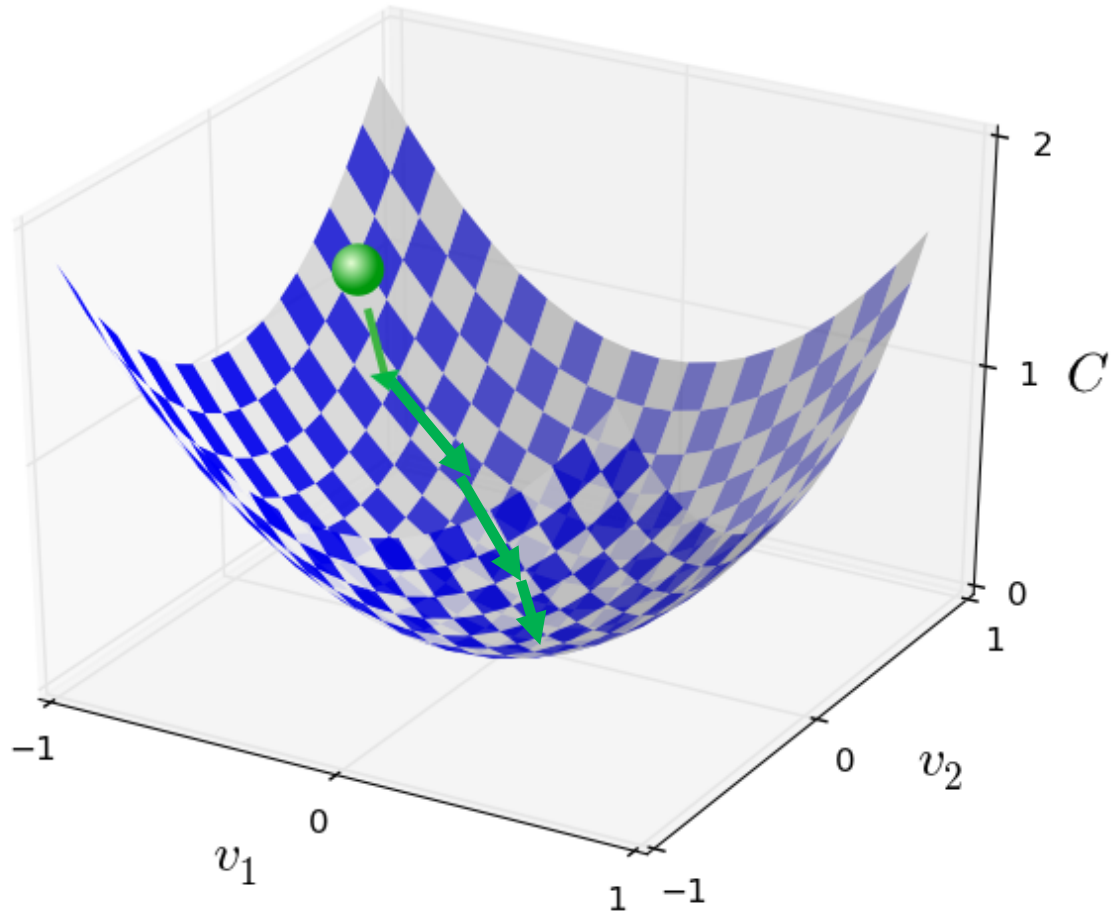
$$\Delta C \approx -\eta \nabla C \cdot \nabla C = -\eta \|\nabla C\|^2$$

$$\text{Because } \|\nabla C\|^2 \geq 0, \ \Delta C \leq 0$$

C will always decrease, if we change v according to $\Delta v = -\eta \nabla C$

$$v \to v' = v - \eta \nabla C.$$

Minimize the loss function



$$v \rightarrow v' = v - \eta \nabla C.$$

$$v \rightarrow v' = v - \eta \nabla C.$$

$$v \rightarrow v' = v - \eta \nabla C.$$

$$v \rightarrow v' = v - \eta \nabla C.$$

Backpropagation

Learning representations by back-propagating errors

David E. Rumelhart*, Geoffrey E. Hinton† & Ronald J. Williams*

* Institute for Cognitive Science, C-015, University of California, San Diego, La Jolla, California 92093, USA
† Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Philadelphia 15213, USA

We describe a new learning procedure, back-propagation, for networks of neurone-like units. The procedure repeatedly adjusts the weights of the connections in the network so as to minimize a measure of the difference between the actual output vector of the net and the desired output vector. As a result of the weight adjustments, internal 'hidden' units which are not part of the input or output come to represent important features of the task domain, and the regularities in the task are captured by the interactions of these units. The ability to create useful new features distinguishes back-propagation from earlier, simpler methods such as the perceptron-convergence procedure[1].

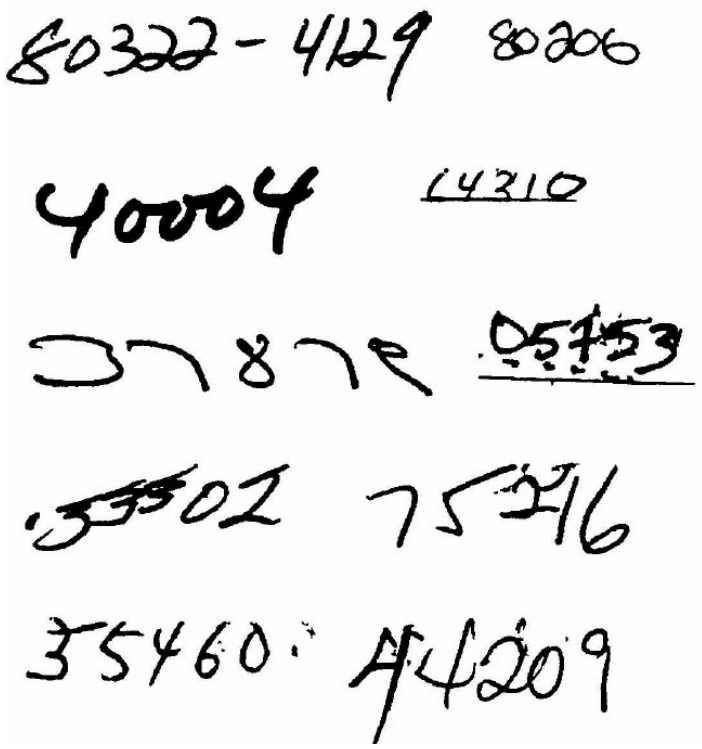
There have been many attempts to design self-organizing neural networks. The aim is to find a powerful synaptic modification rule that will allow an arbitrarily connected neural network to develop an internal structure that is appropriate for a particular task domain. The task is specified by giving the more difficult when we introduce hidden units whose actual or desired states are not specified by the task. (In perceptrons, there are 'feature analysers' between the input and output that are not true hidden units because their input connections are fixed by hand, so their states are completely determined by the input vector: they do not learn representations.) The learning procedure must decide under what circumstances the hidden units should be active in order to help achieve the desired input–output behaviour. This amounts to deciding what these units should represent. We demonstrate that a general purpose and relatively simple procedure is powerful enough to construct appropriate internal representations.

The simplest form of the learning procedure is for layered networks which have a layer of input units at the bottom; any number of intermediate layers; and a layer of output units at the top. Connections within a layer or from higher to lower layers are forbidden, but connections can skip intermediate layers. An input vector is presented to the network by setting the states of the input units. Then the states of the units in each layer are determined by applying equations (1) and (2) to the connections coming from lower layers. All units within a layer have their states set in parallel, but different layers have their states set sequentially, starting at the bottom and working upwards until the states of the output units are determined.

The total input, x_j, to unit j is a linear function of the outputs, y_i, of the units that are connected to j and of the weights, w_{ji}, on these connections

$$x_j = \sum_i y_i w_{ji} \qquad (1)$$

Digit classification

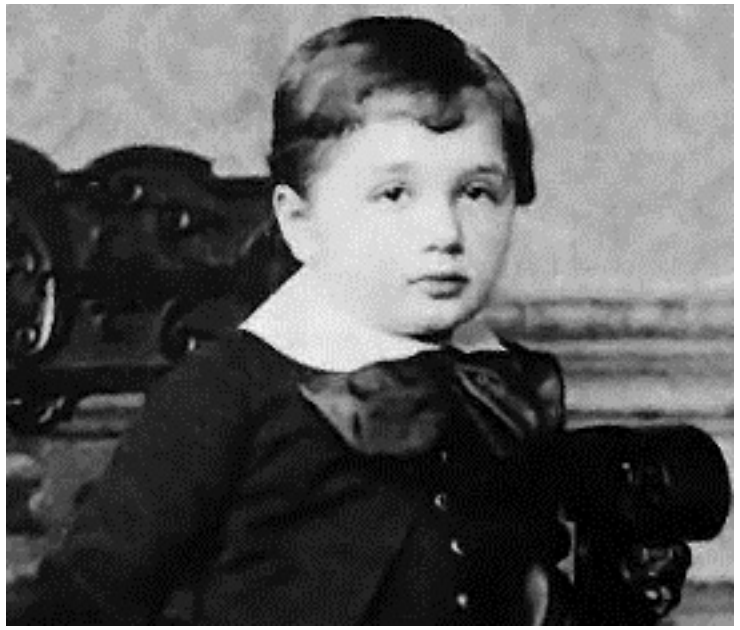


Zip codes

Backpropagation applied to handwritten zip code recognition,
Lecun et al., 1989

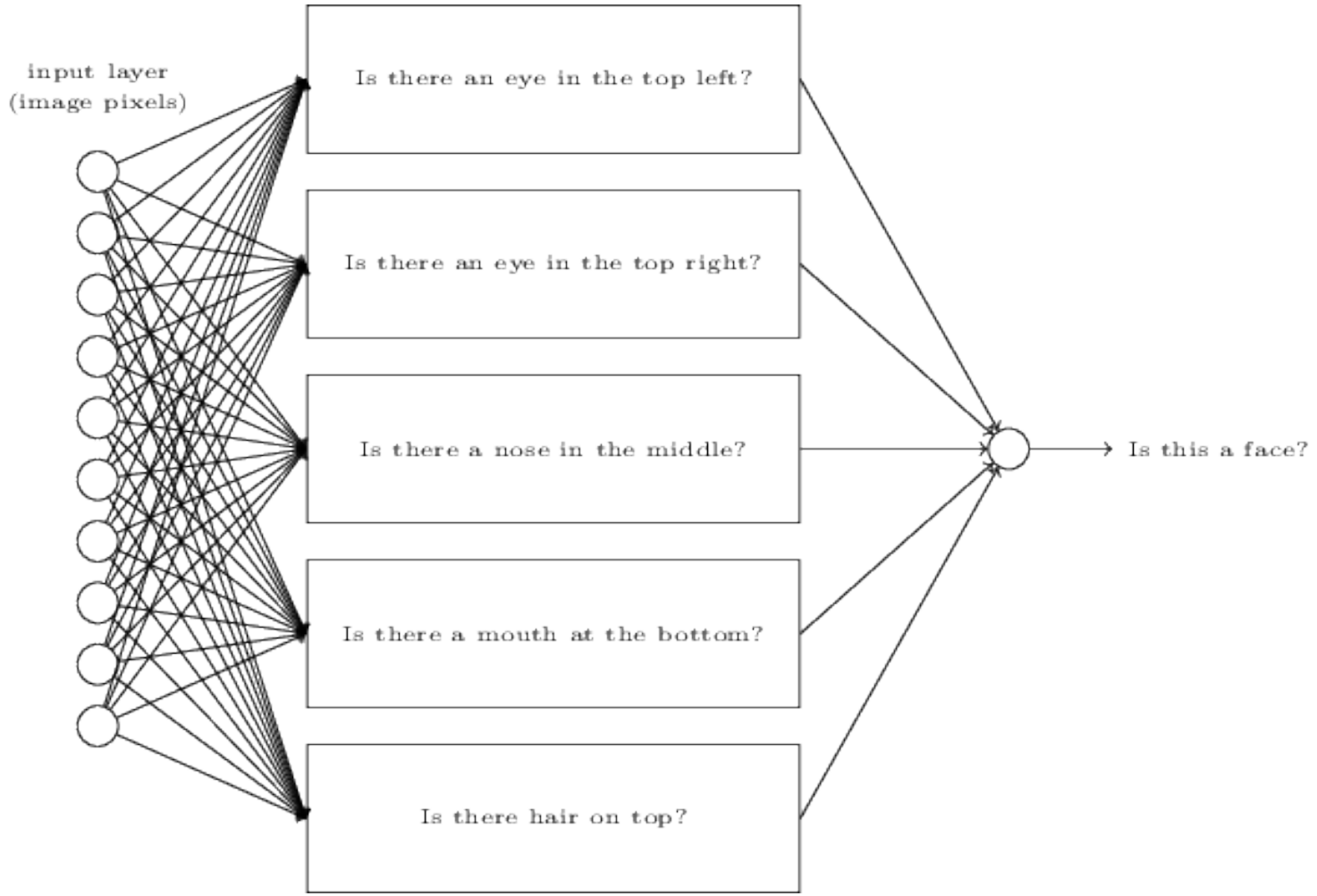
Toward Deep Learning

- Which of the following contain a face?

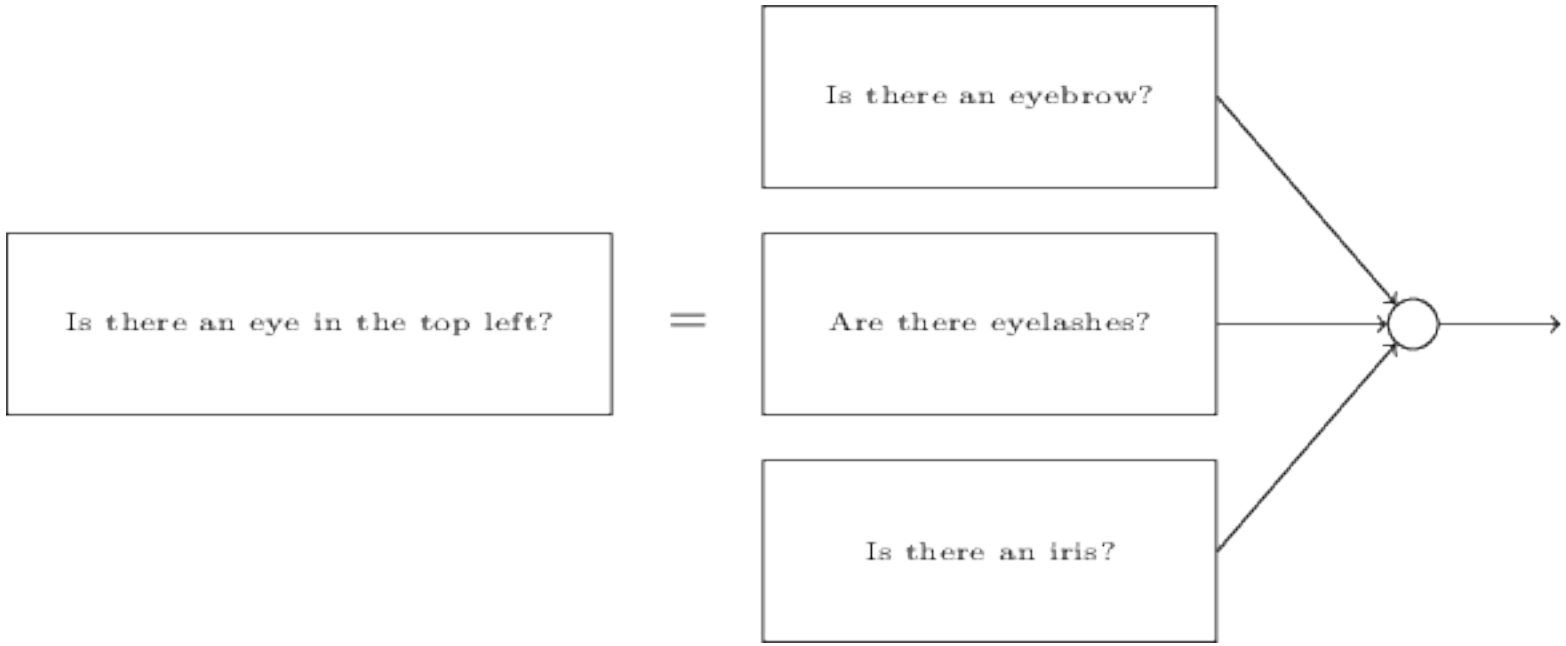


- Can we approach the problem the same way we do for digits?

Toward Deep Learning

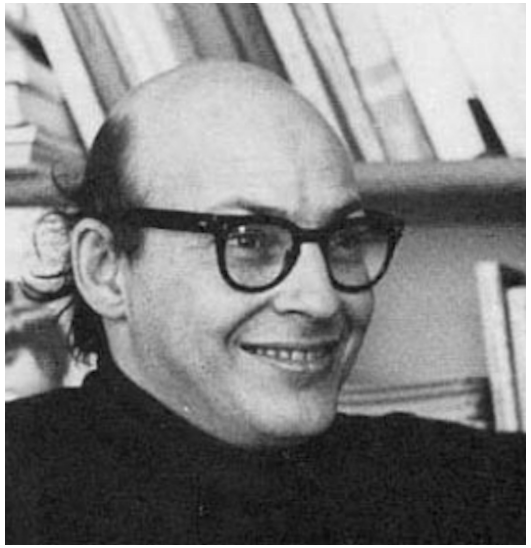


Toward Deep Learning



1966

"Connect a television camera to a computer and get the machine to describe what it sees."



Marvin Minsky
Turing award, 1969

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
PROJECT MAC

Artificial Intelligence Group — July 7, 1966
Vision Memo. No. 100.

THE SUMMER VISION PROJECT

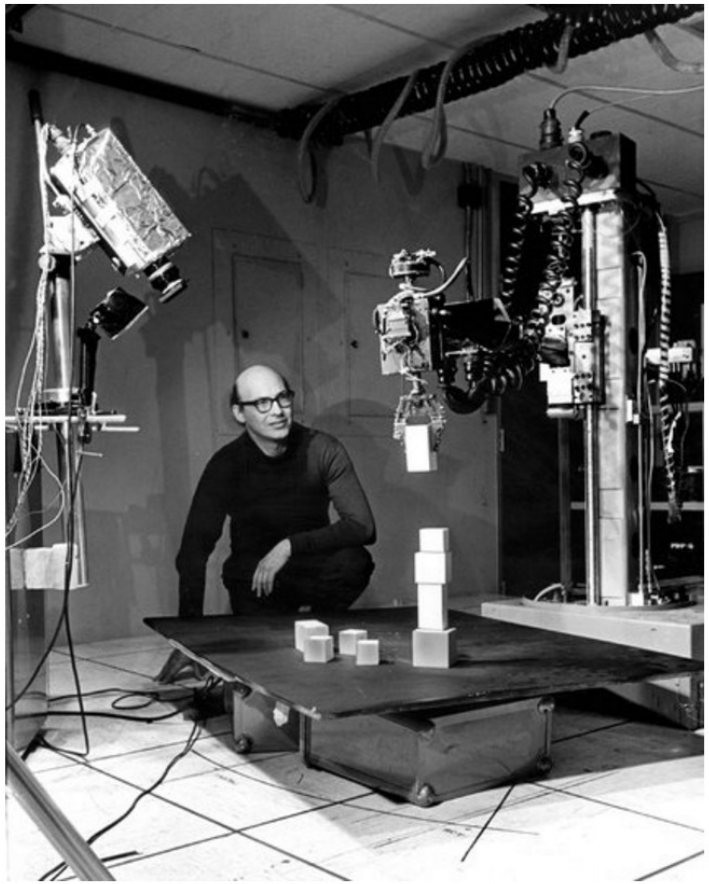
Seymour Papert

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".

The New York Times

Marvin Minsky, Pioneer in Artificial Intelligence, Dies at 88

By GLENN RIFKIN JAN. 25, 2016



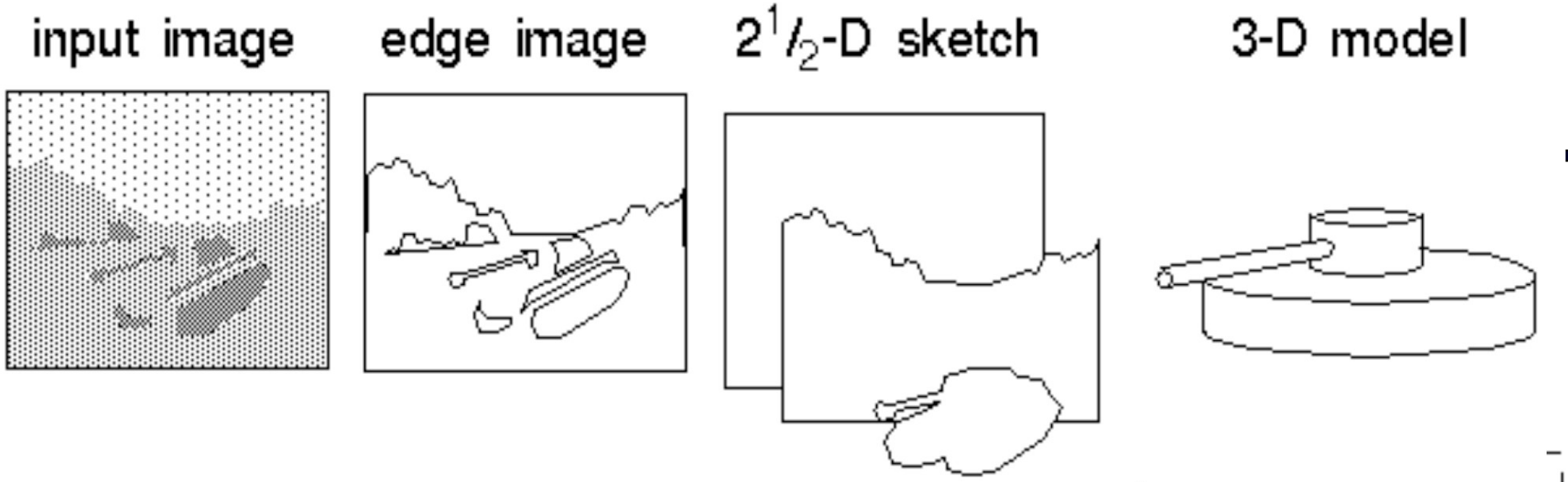
Marvin Minsky in a lab at M.I.T. in 1968. M.I.T.

Marvin Minsky, who combined a scientist's thirst for knowledge with a philosopher's quest for truth as a pioneering explorer of artificial intelligence, work that helped inspire the creation of the personal computer and the Internet, died on Sunday night in Boston. He was 88.

His family said the cause was a cerebral hemorrhage.

Well before the advent of the microprocessor and the supercomputer, Professor Minsky, a revered computer science educator at M.I.T., laid the foundation for the field of artificial intelligence by demonstrating the possibilities of imparting common-sense reasoning to computers.

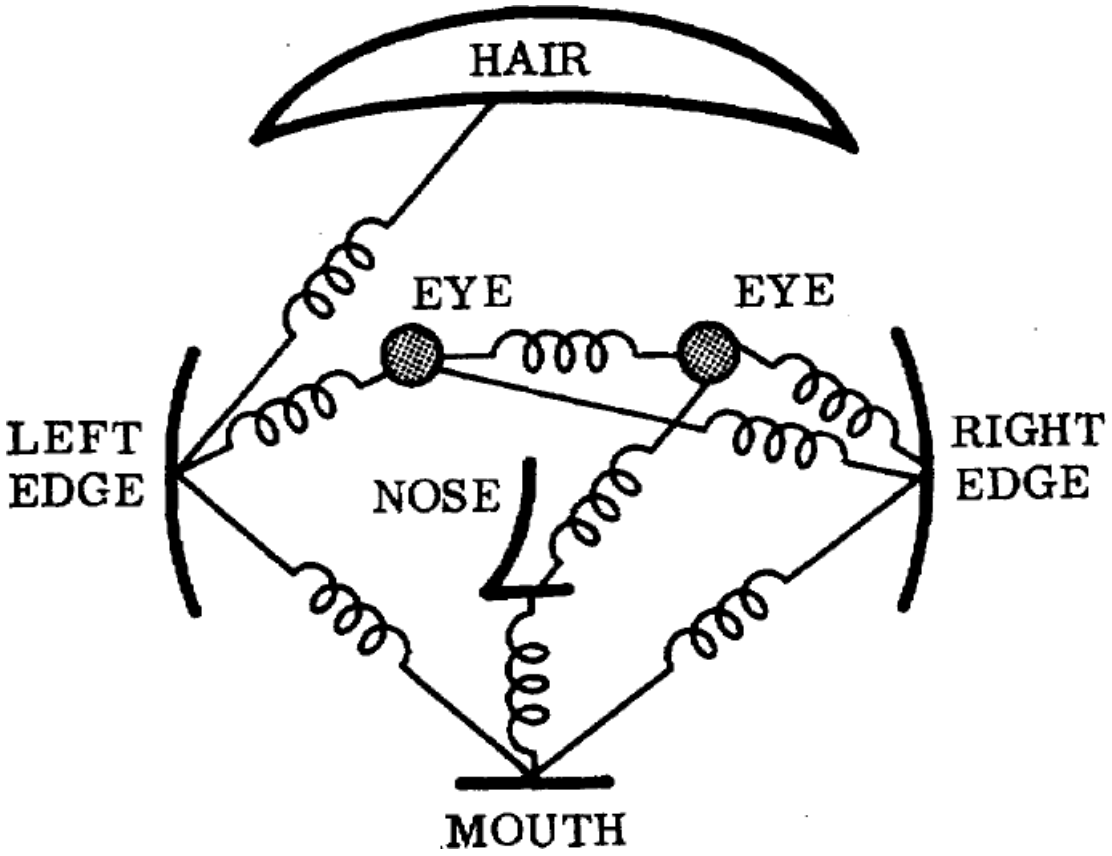
"Marvin was one of the very few people in computing whose visions and



Input Image	Primal Sketch	2 1/2-D Sketch	3-D Model Representation
Perceived intensities	Zero crossings, blobs,edges, bars, ends, virtual lines, groups, curves boundaries.	Local surface orientation and discontinuities in depth and in surface orientation	3-D models hierarchically organised in terms of surface and volumetric primitives

Stages of Visual Representation, David Marr, 1970

1973

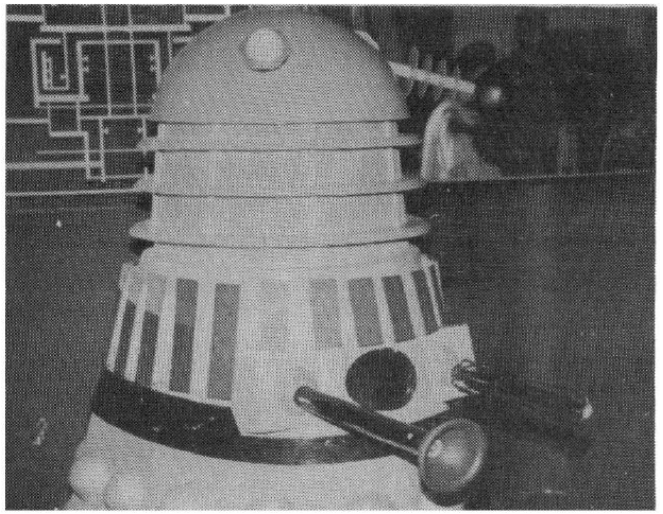


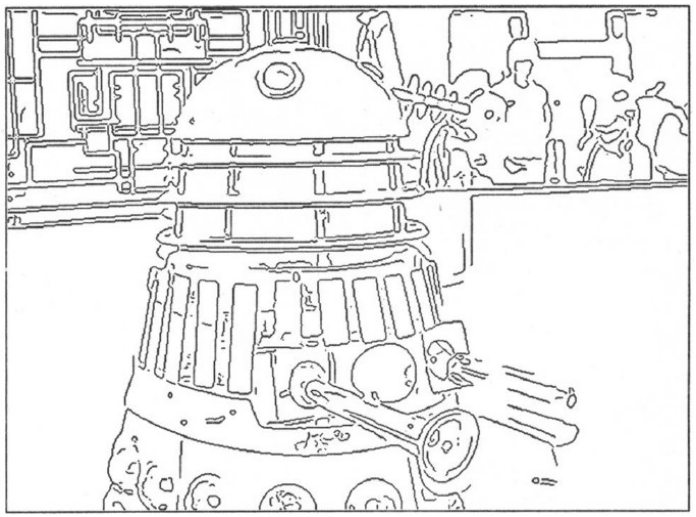
The constellation model

The representation and matching of pictorial structures,
Fischler and Elschlager, 1973

1980's

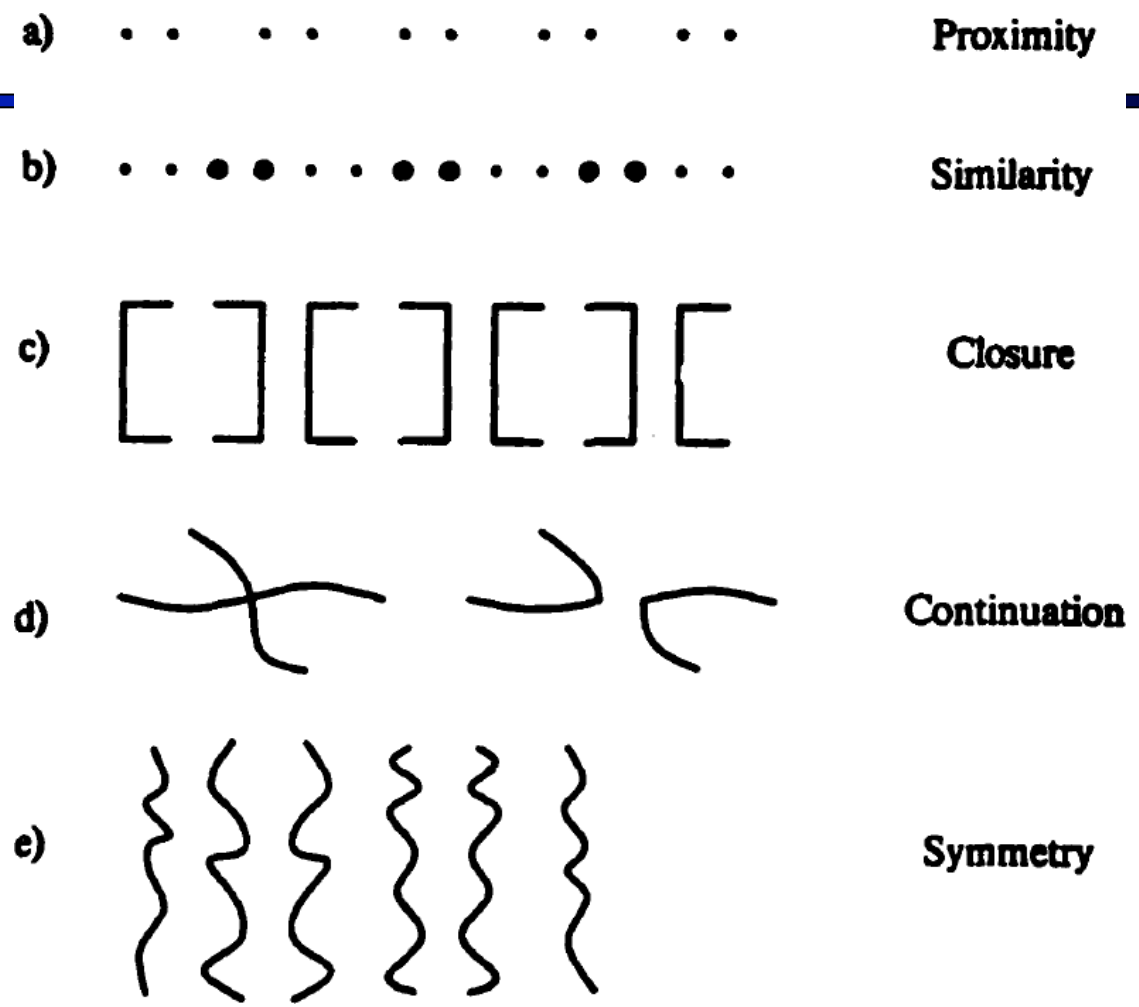
AI winter… …back to basics





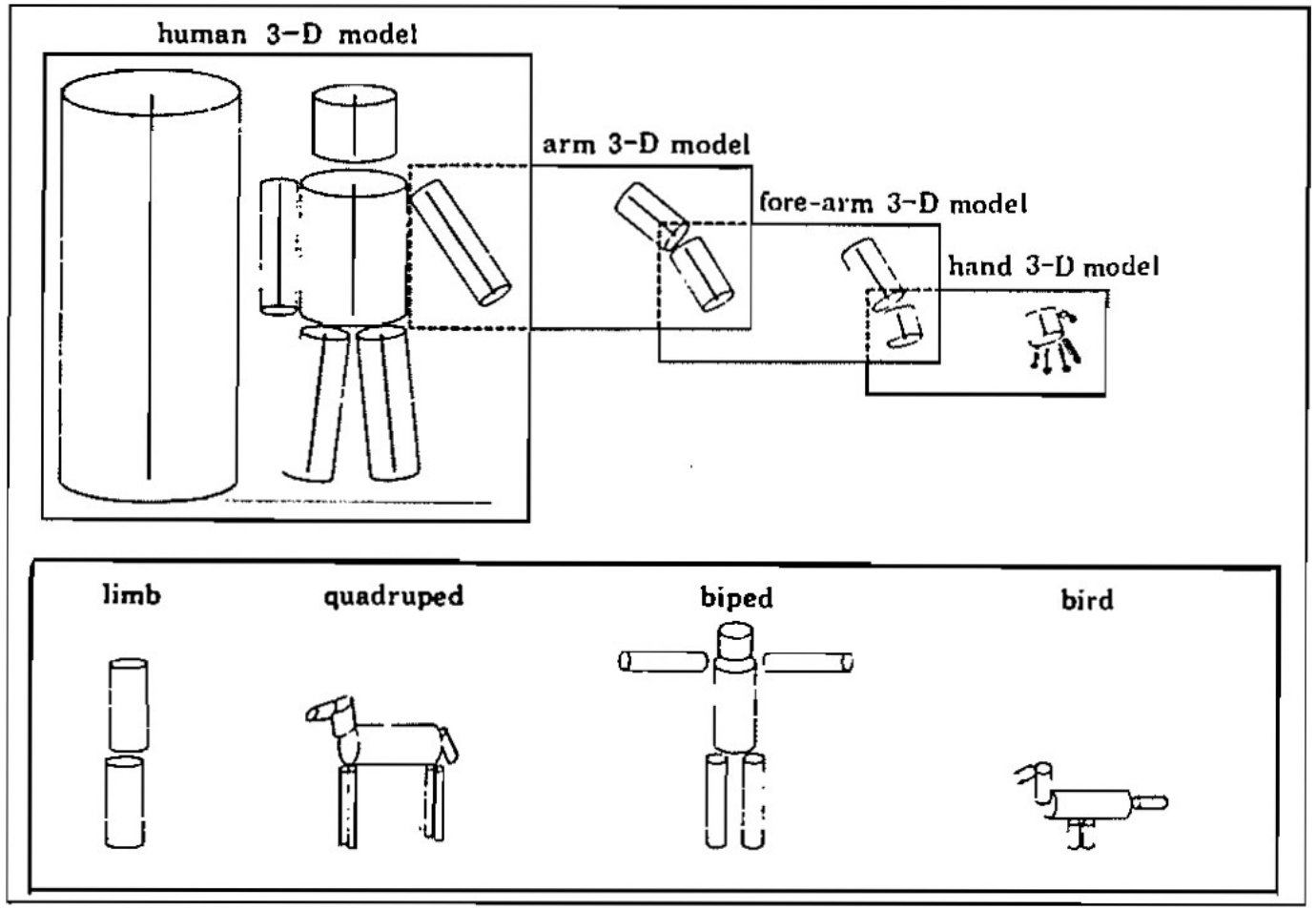
A Computational Approach to Edge Detection, Canny 1986

1984



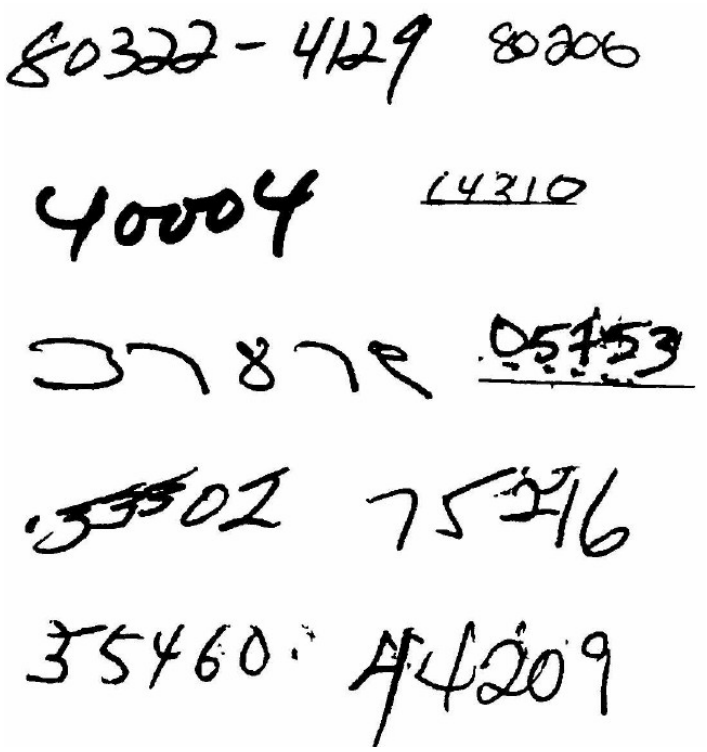
Perceptual Organization and Visual Recognition,
David Lowe, 1984

1986



Perceptual organization and the representation of natural form,
Alex Pentland, 1986

1989



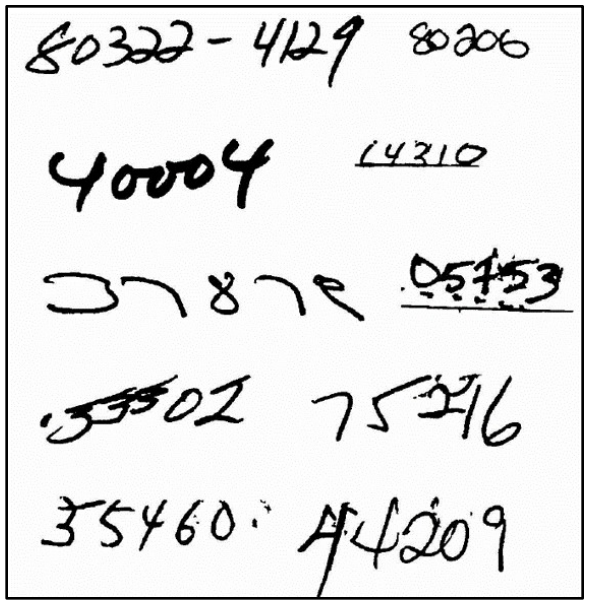
Zip codes

MNIST

Backpropagation applied to handwritten zip code recognition,
Lecun et al., 1989

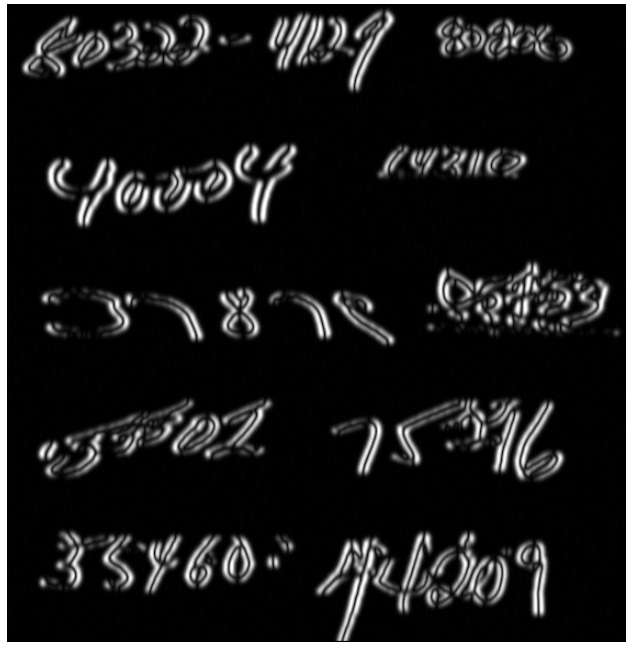
Filters

Input



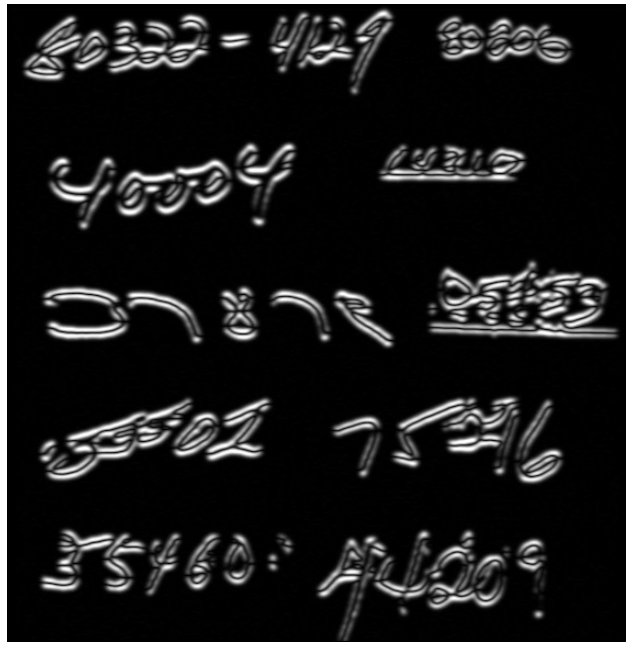
-1	0	+1
-2	0	+2
-1	0	+1

x filter

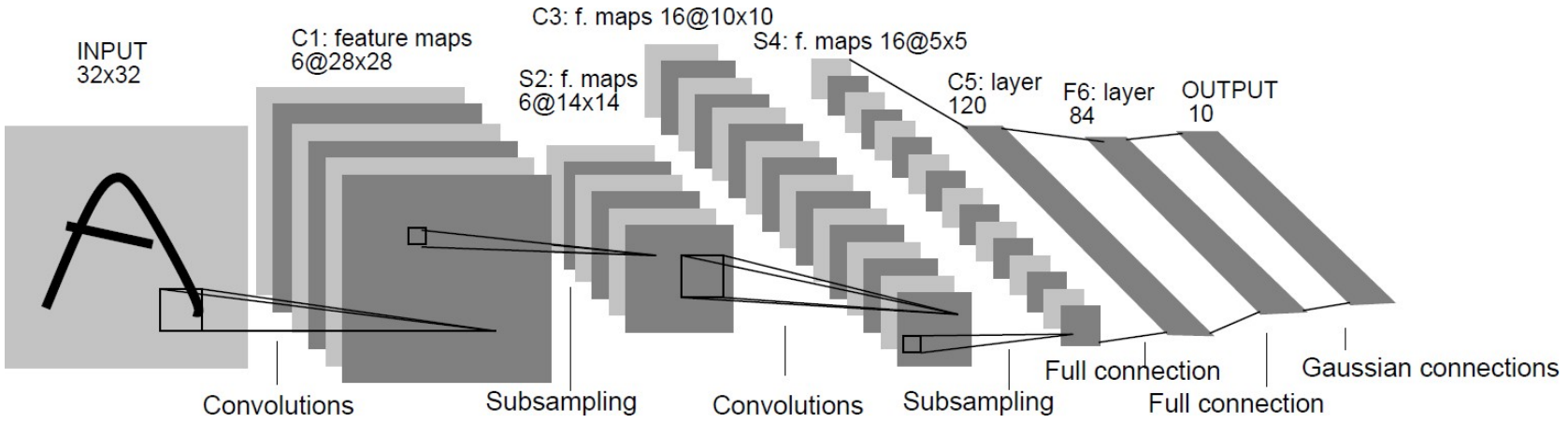


+1	+2	+1
0	0	0
-1	-2	-1

y filter

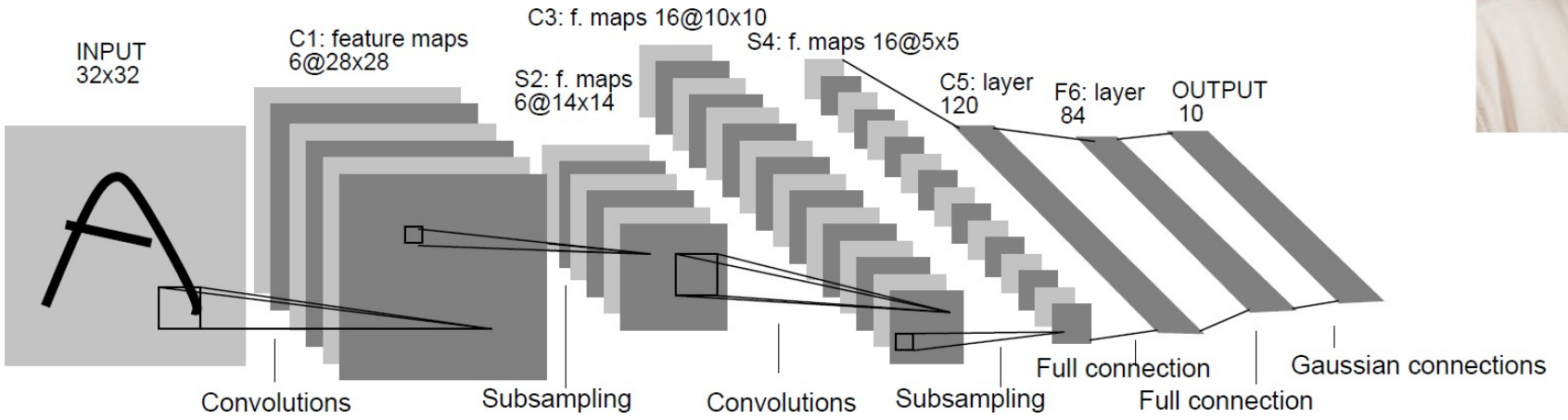


1989



Backpropagation applied to handwritten zip code recognition,
Lecun et al., 1989

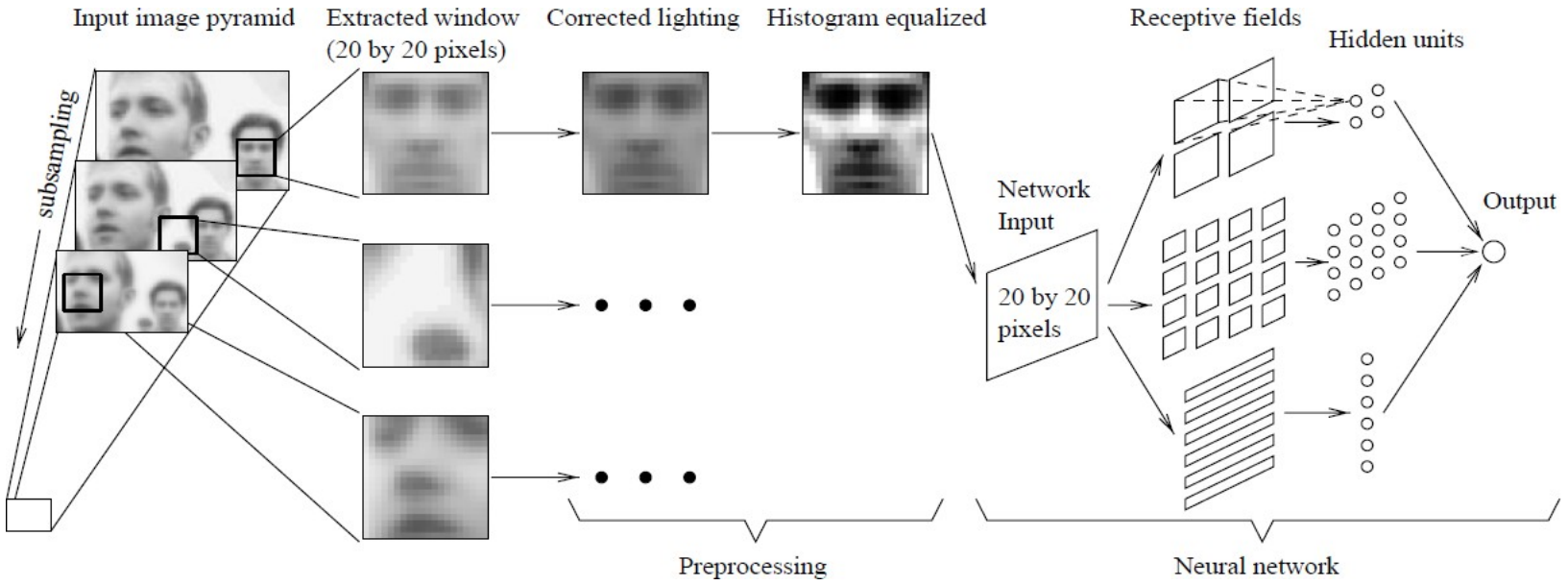
1989



Backpropagation applied to handwritten zip code recognition,
Lecun et al., 1989

1998

Faces

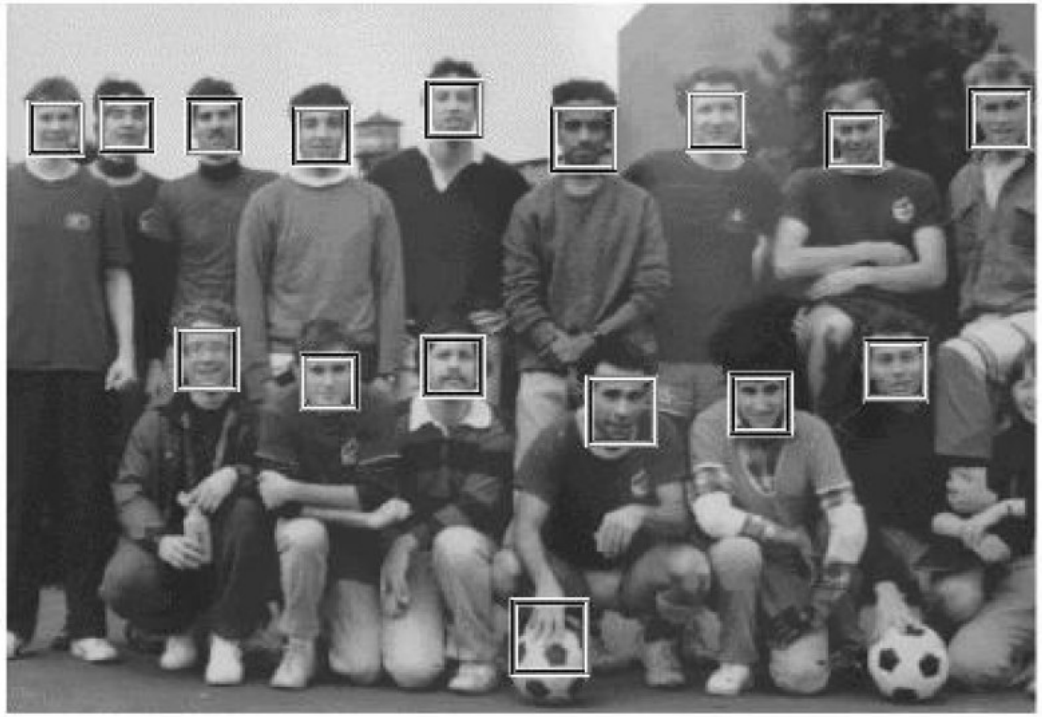


Neural Network--Based Face Detection, Rowley at al., PAMI 1998

2001

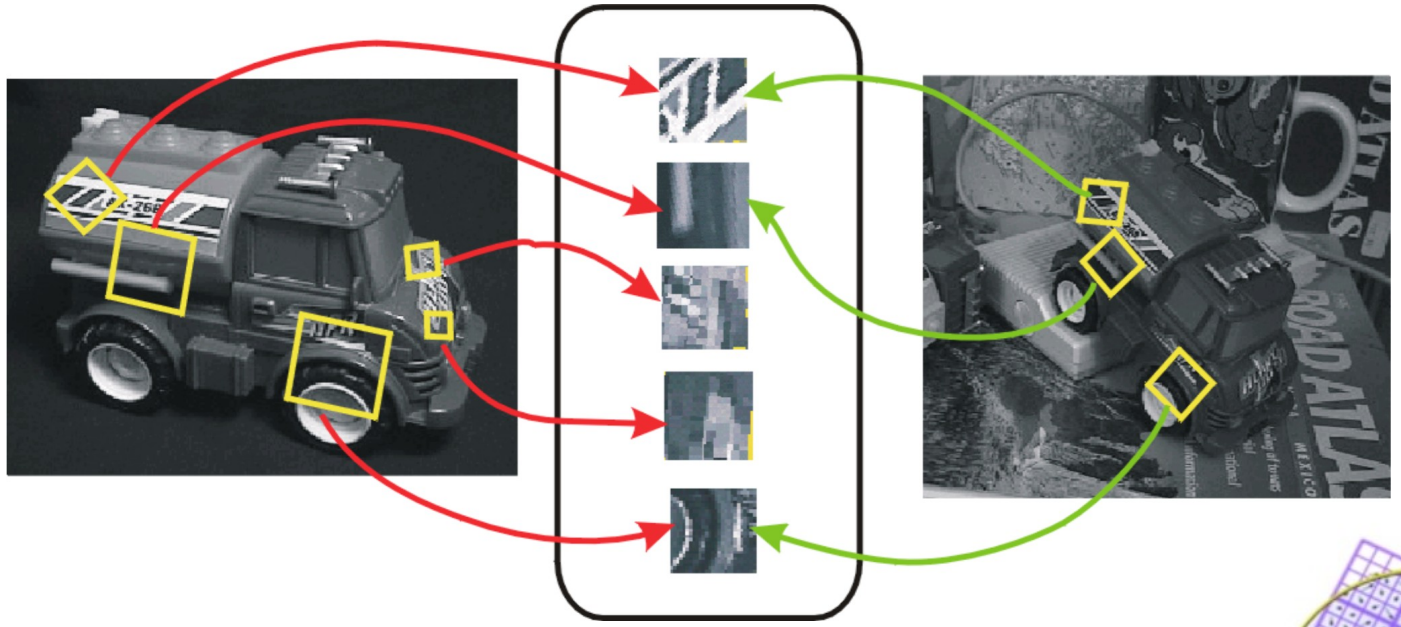
Sliding window in real time!

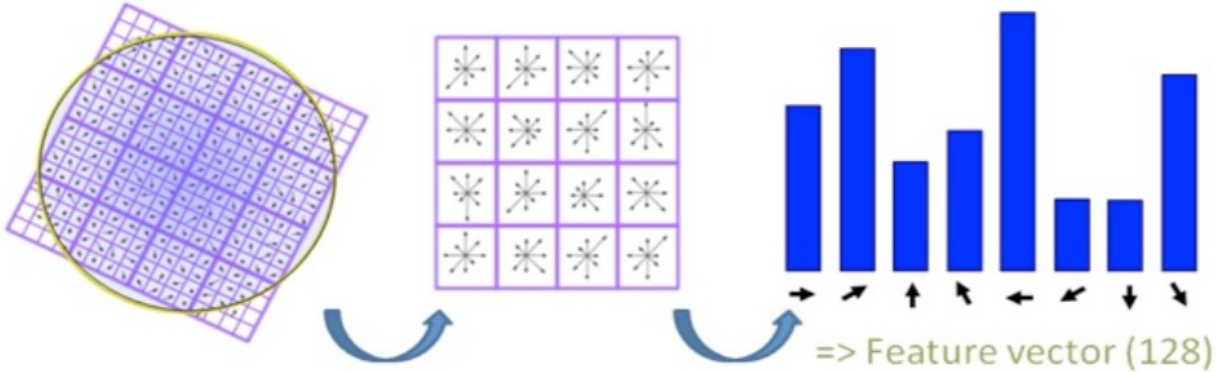
Boosting + Cascade =Speed



Rapid Object Detection using a Boosted Cascade of Simple Features,
Viola and Jones, CVPR 2001

1999 SIFT (Scale Invariant Feature Transform)

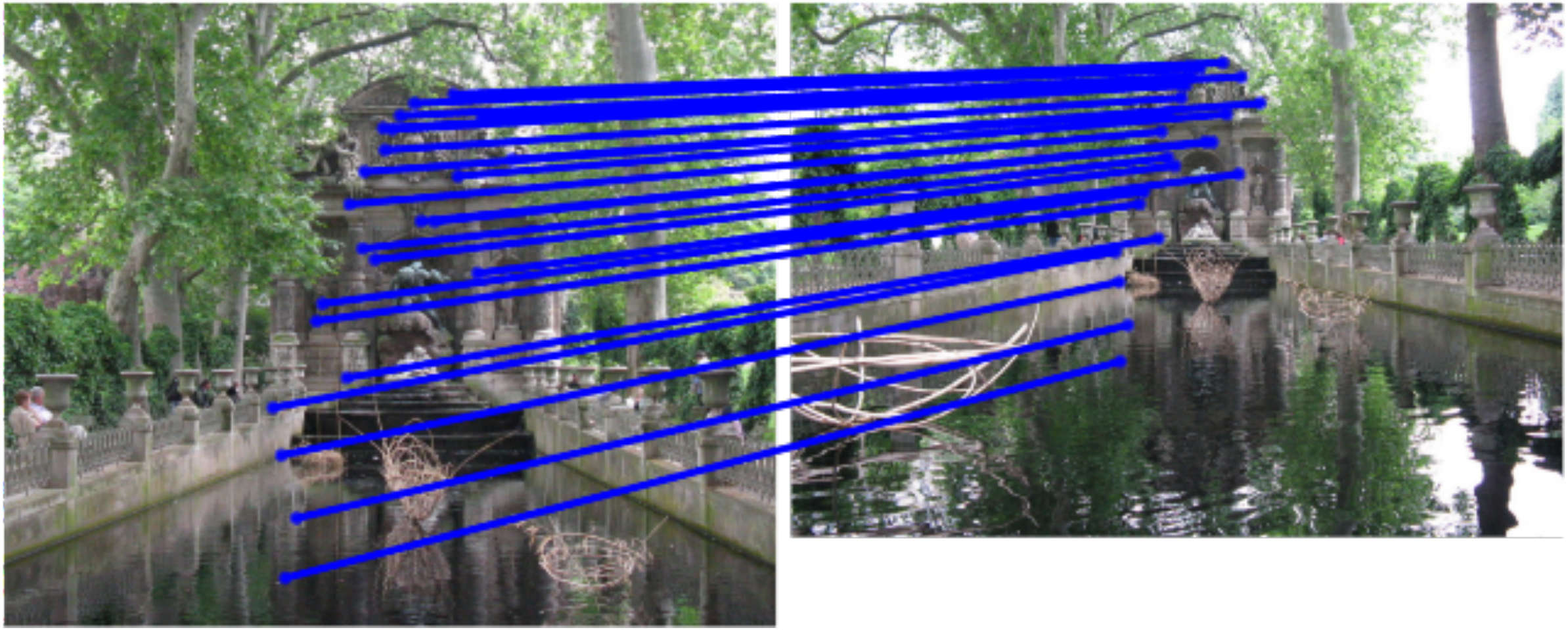




No more sliding windows (interest points)
Better features (use more computation)

Object Recognition from Local Scale--Invariant Features, Lowe, ICCV 1999.

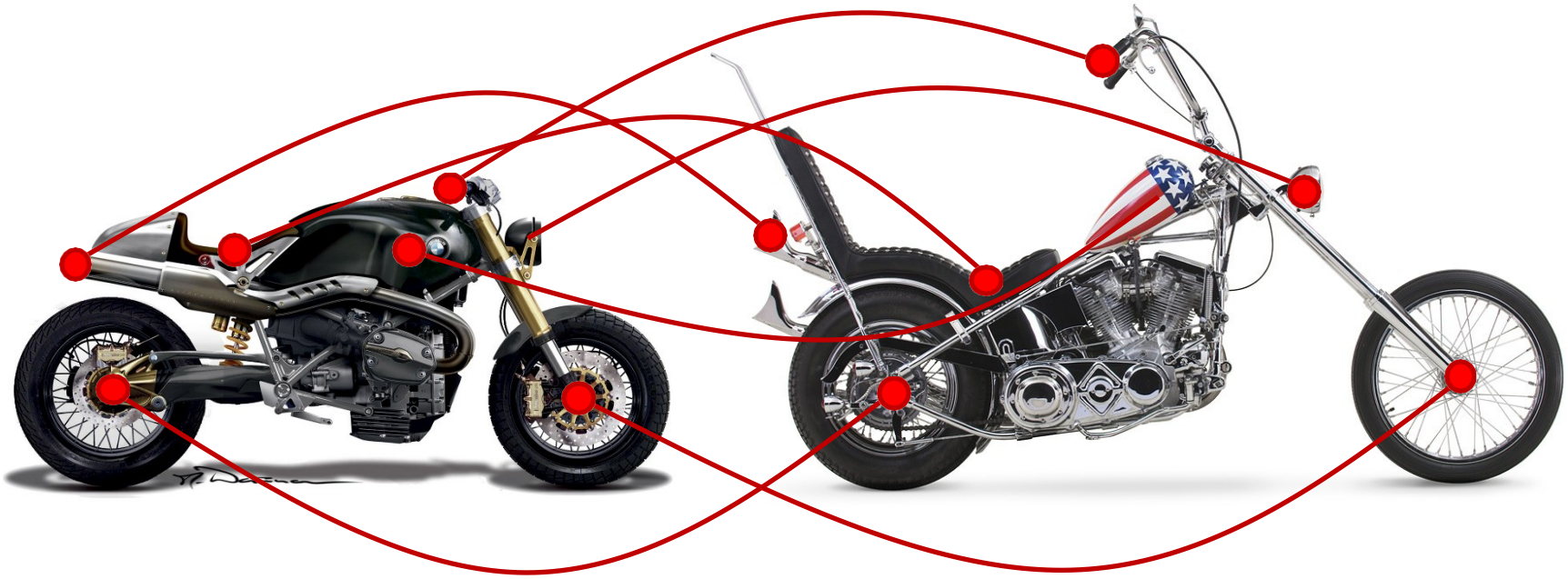
SIFT Matching



[SIFT: Lowe, 2004]

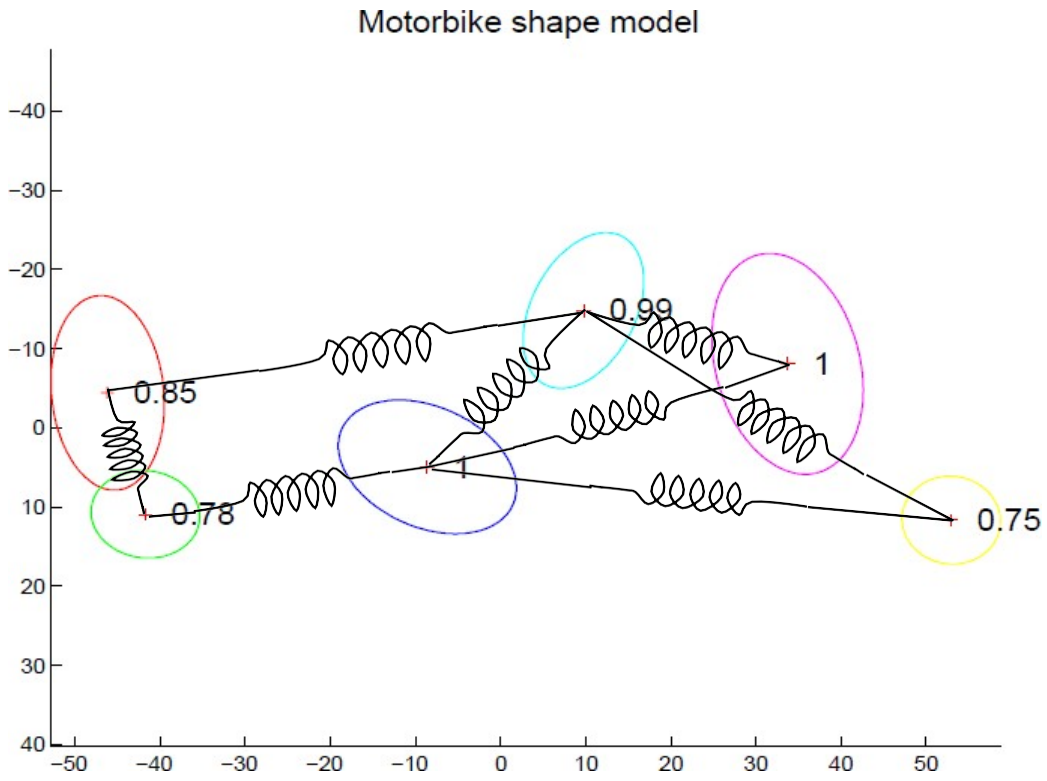
Interest points

2003 Constellation model (redux)

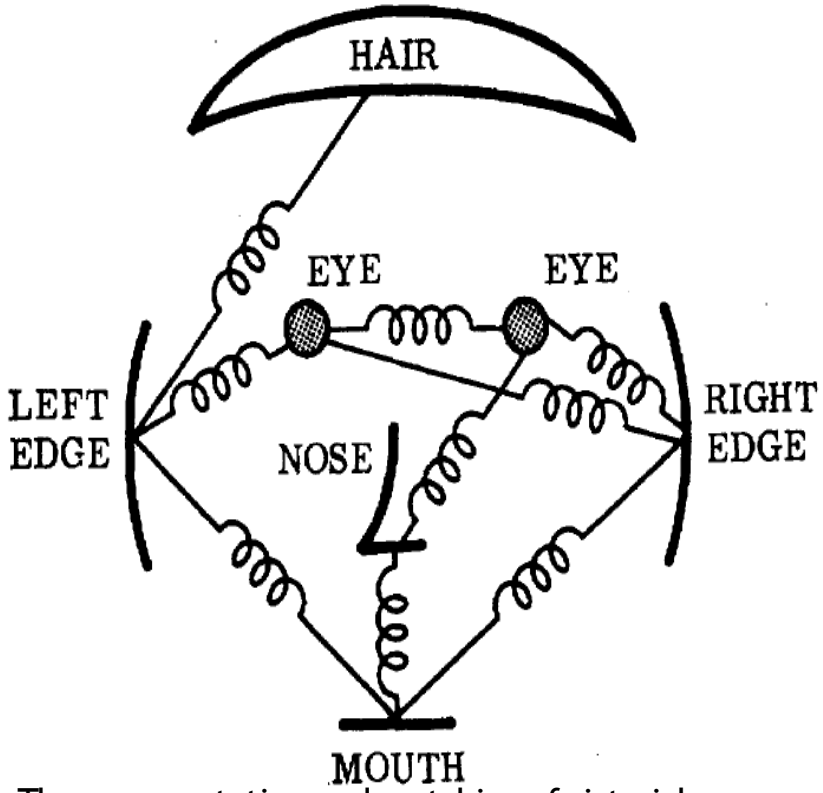


Object Class Recognition by Unsupervised Scale--Invariant Learning,
Fergus et al., *CVPR* 2003.

2003 Constellation model (redux)

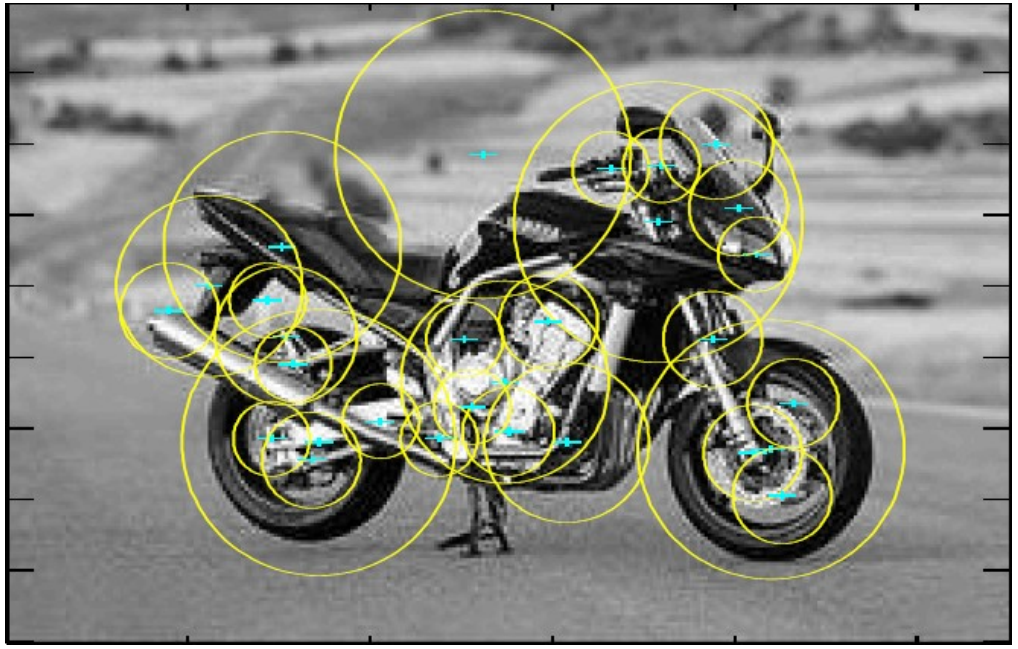


Joint Gaussian density



The representation and matching of pictorial structures, Fischler and Elschlager, 1973

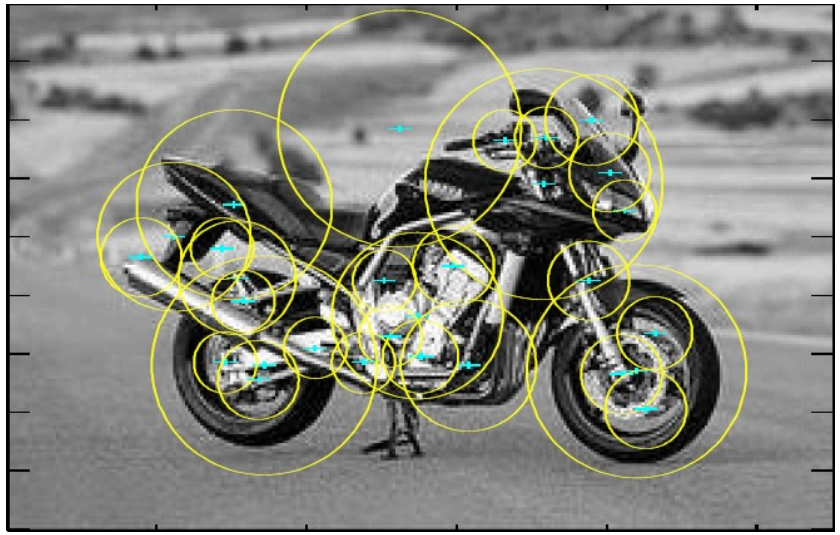
Interest points used to find parts:

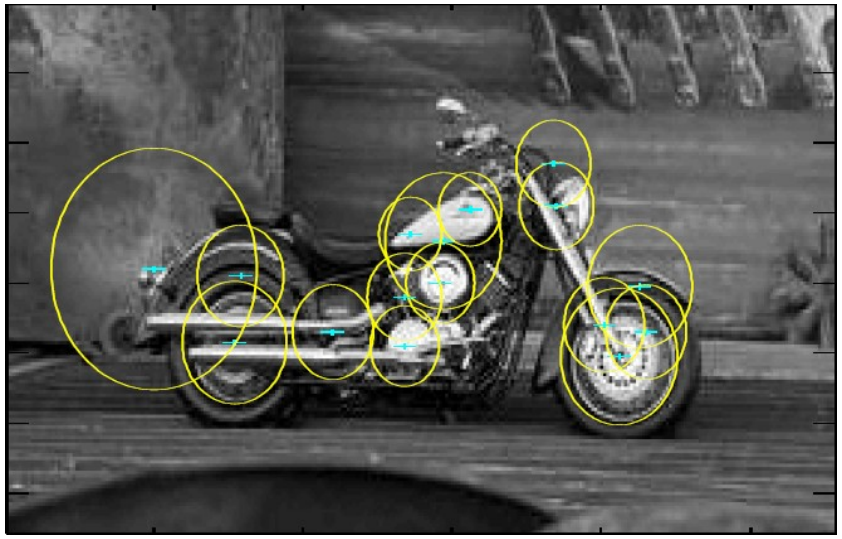


Smaller number of candidate parts allows for more complex spatial models.

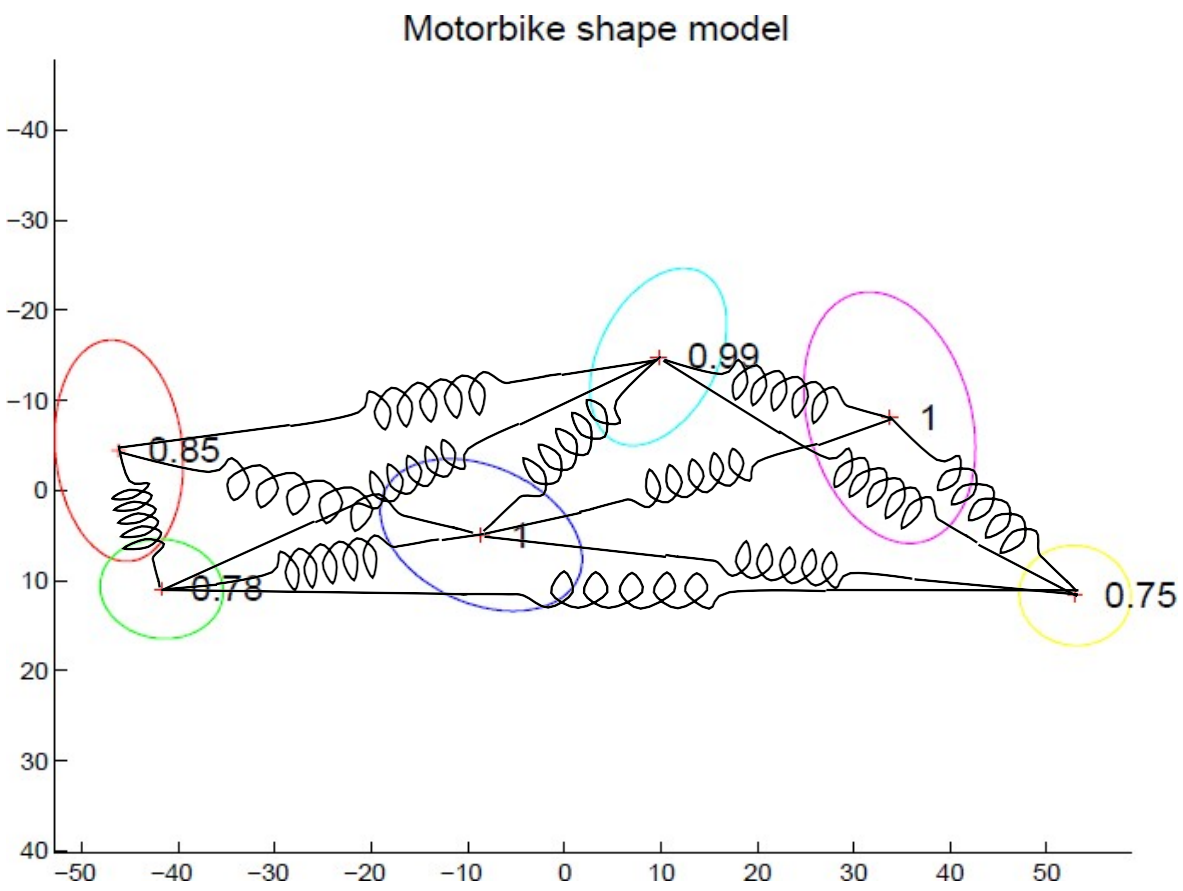
Why it fails

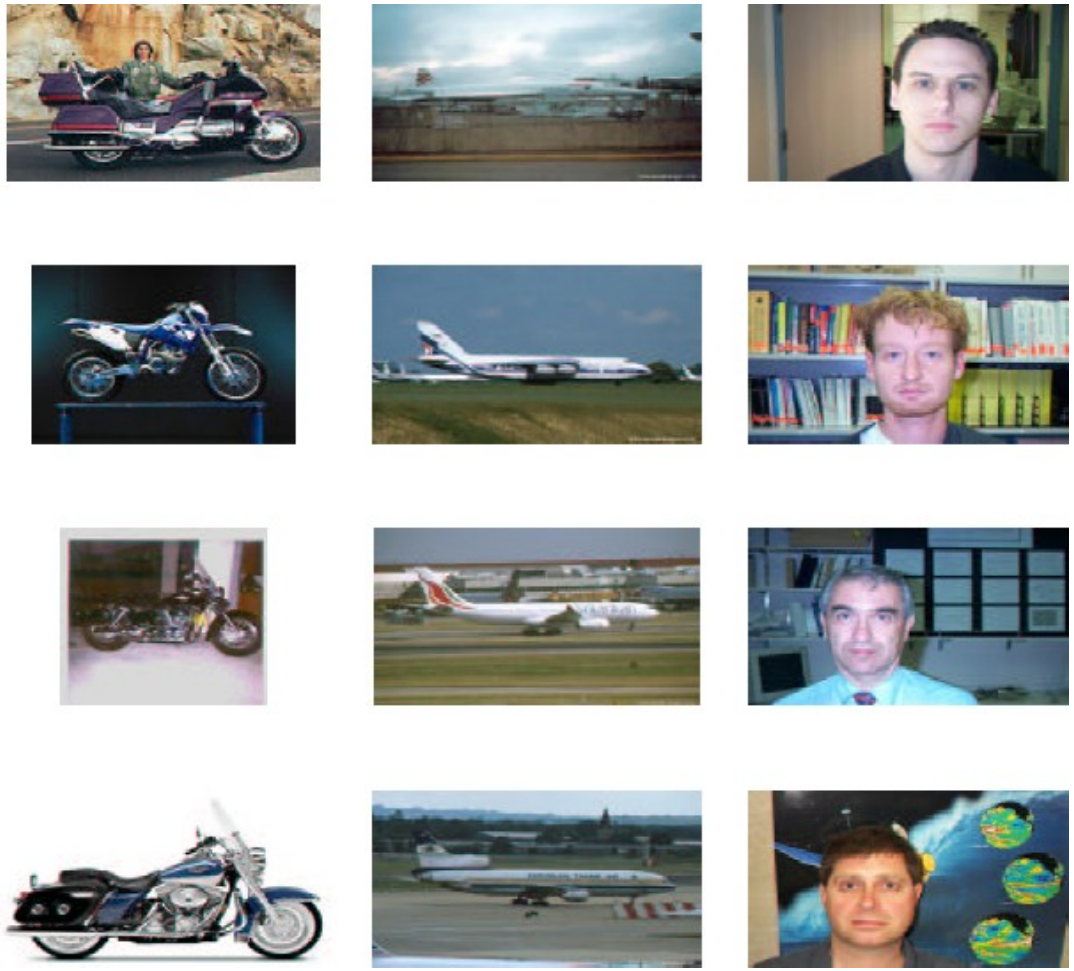
Interest points don't work for category recognition



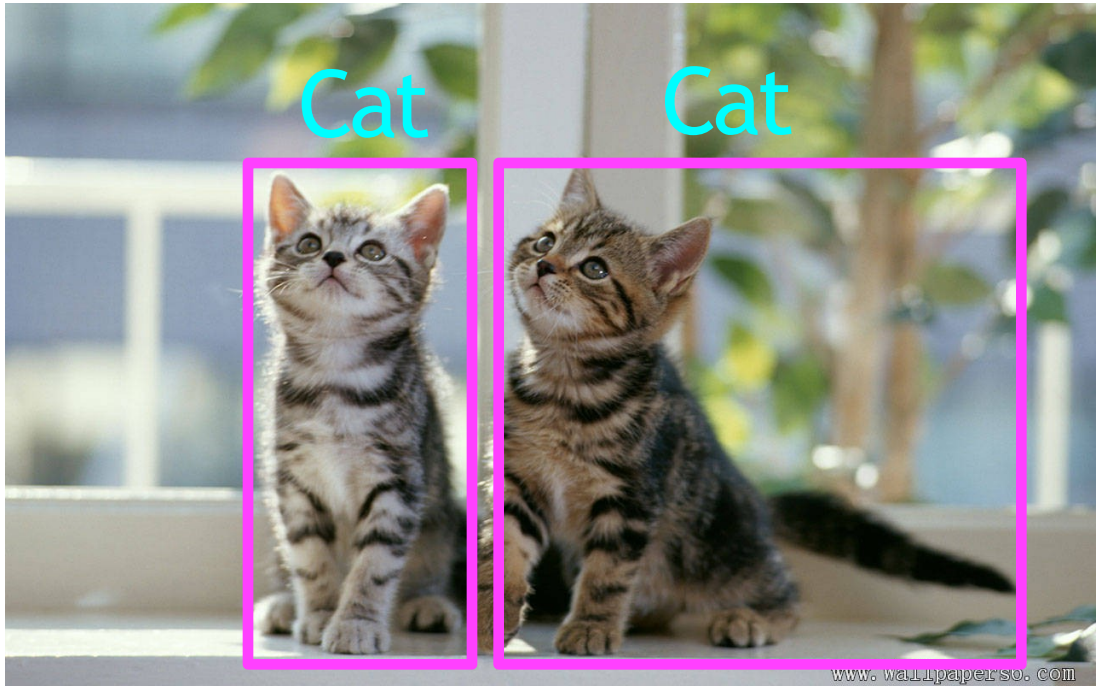


Too many springs…

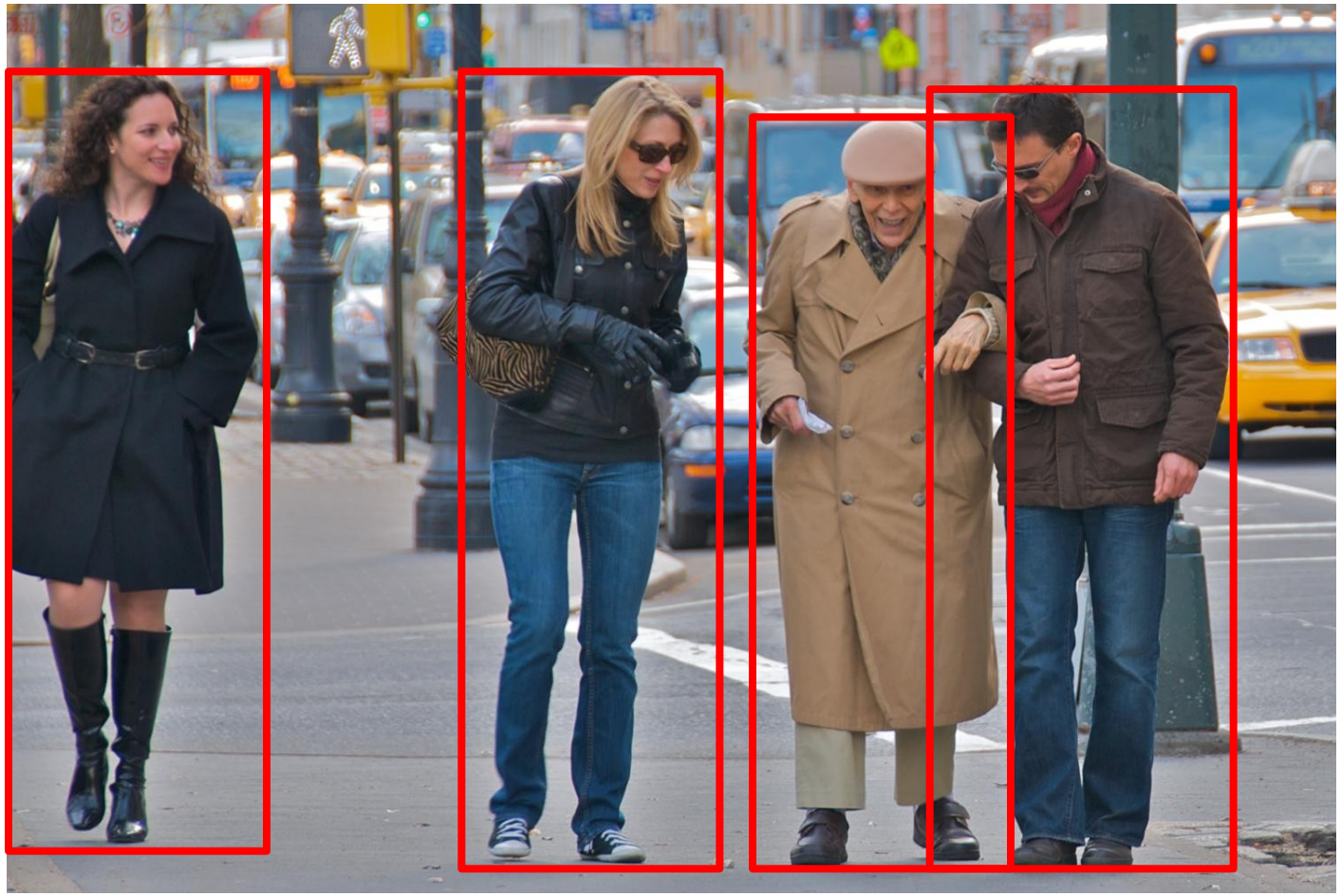




Classification Vs. Detection



2005 HOG (histograms of oriented gradients)



Histograms of oriented gradients for human detection,
Dalal and Triggs, CVPR 2005.

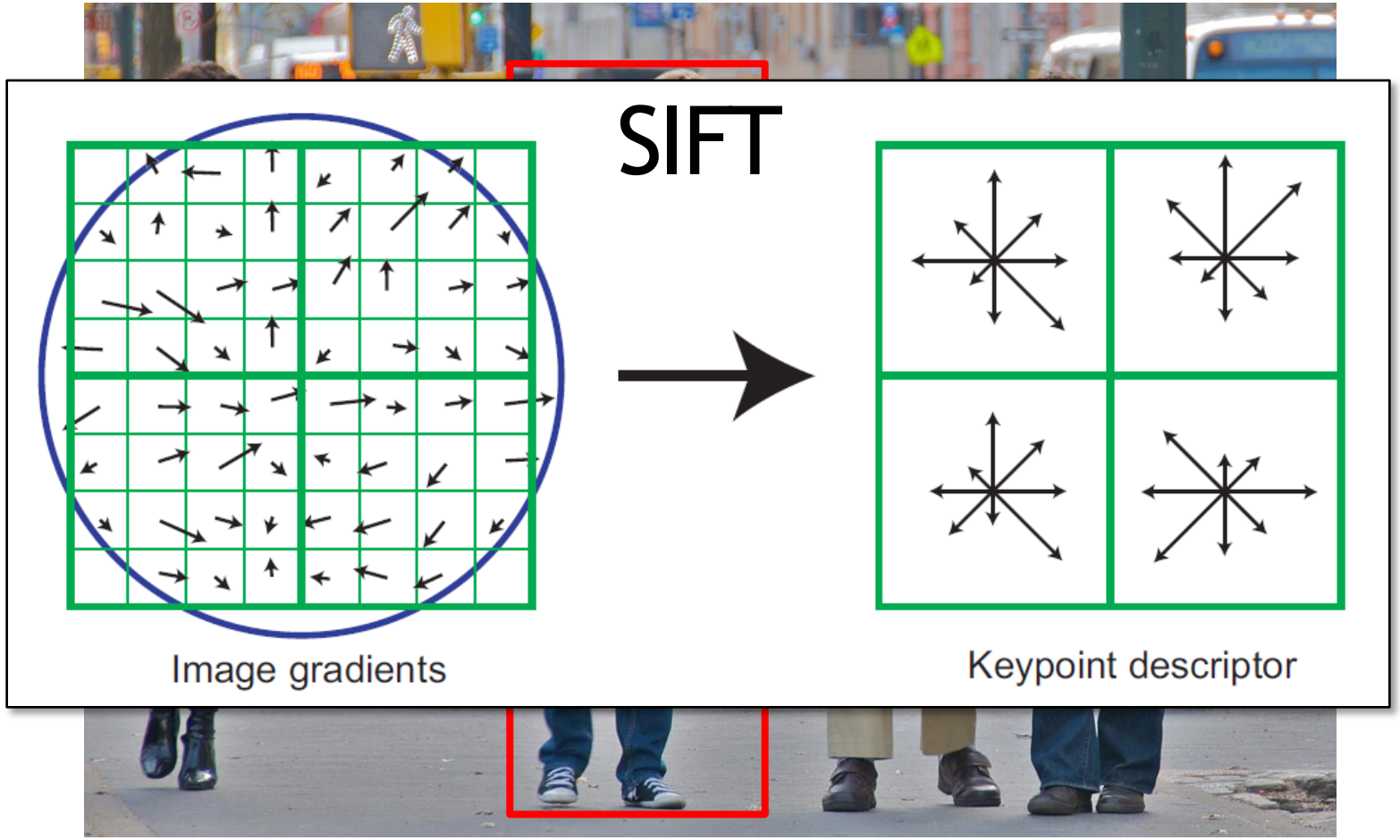
Pedestrians

- Defined by their contours
- Cluttered backgrounds
- Significant variance in texture

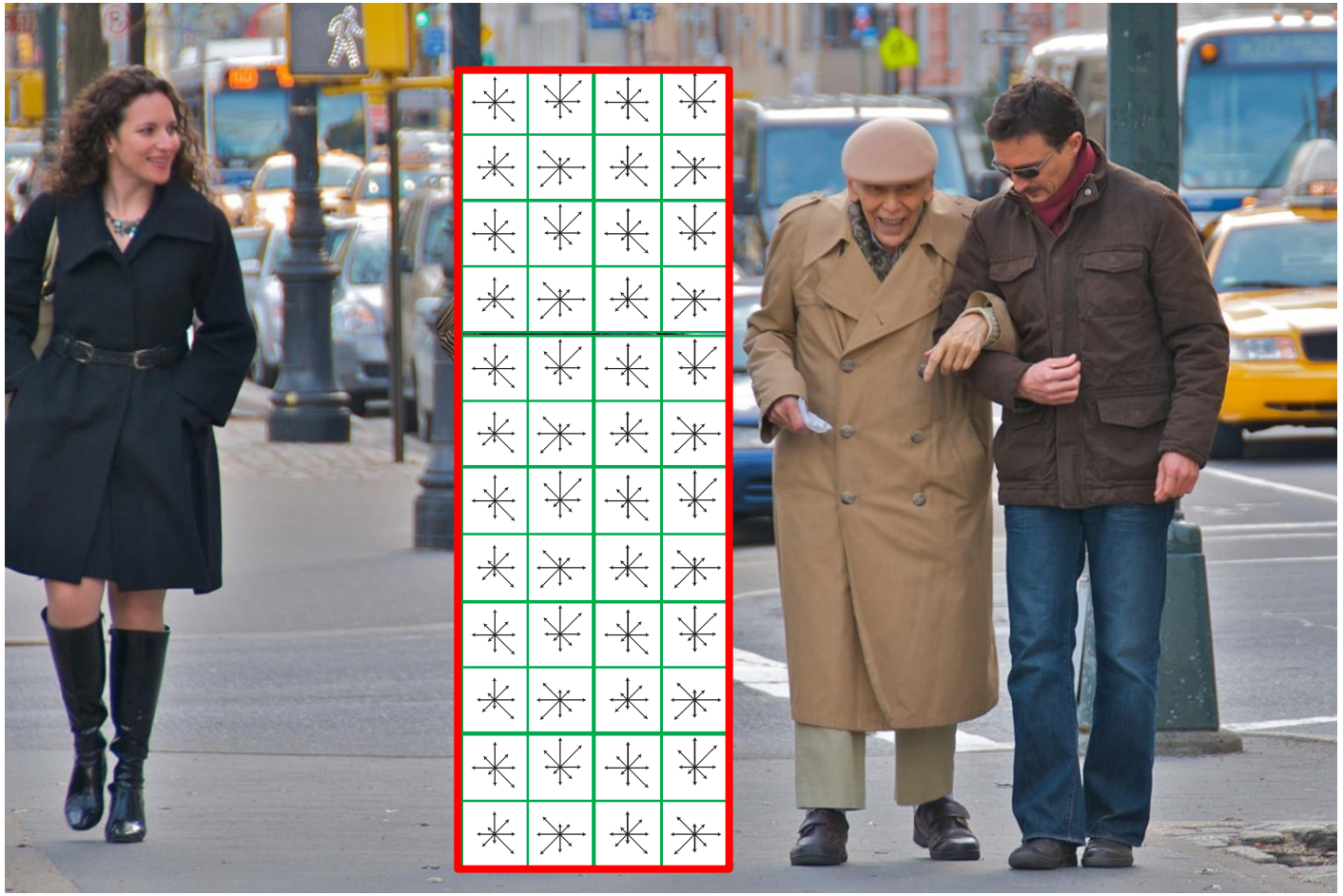
Interest points won't work...

...back to sliding window.

2005 HOG (histograms of oriented gradients)



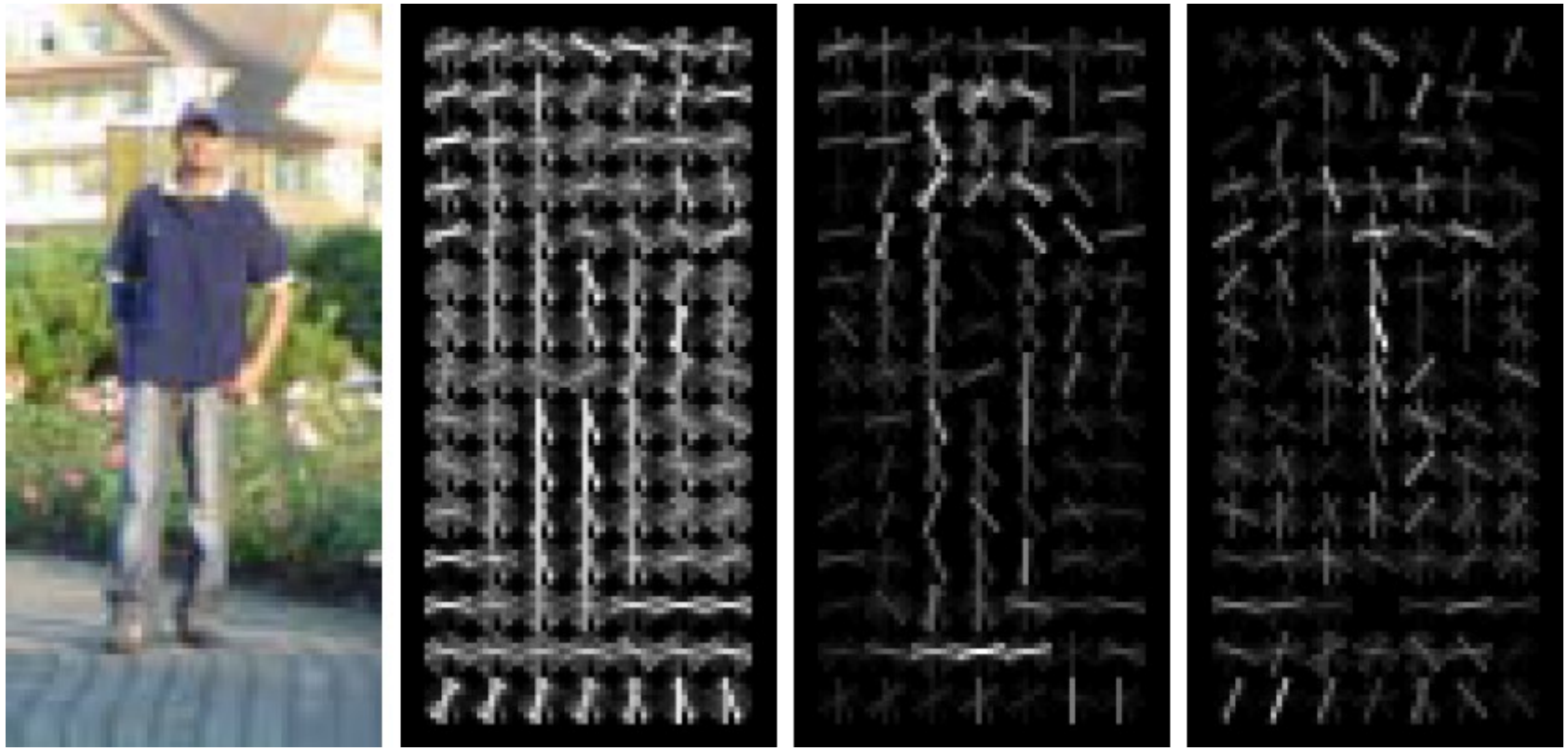
2005 HOG (histograms of oriented gradients)



Histograms of oriented gradients for human detection,
Dalal and Triggs, *CVPR* 2005.

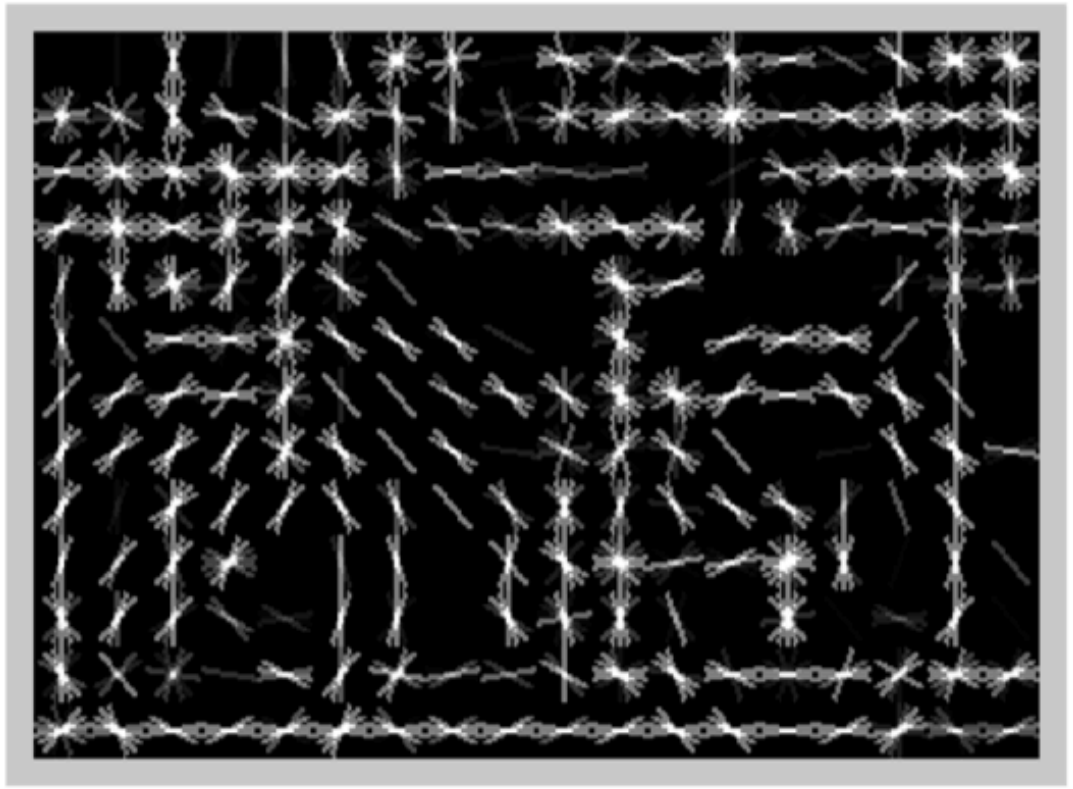
2005 HOG (histograms of oriented gradients)

Presence > Magnitude



✓✓ Normalization by a local window

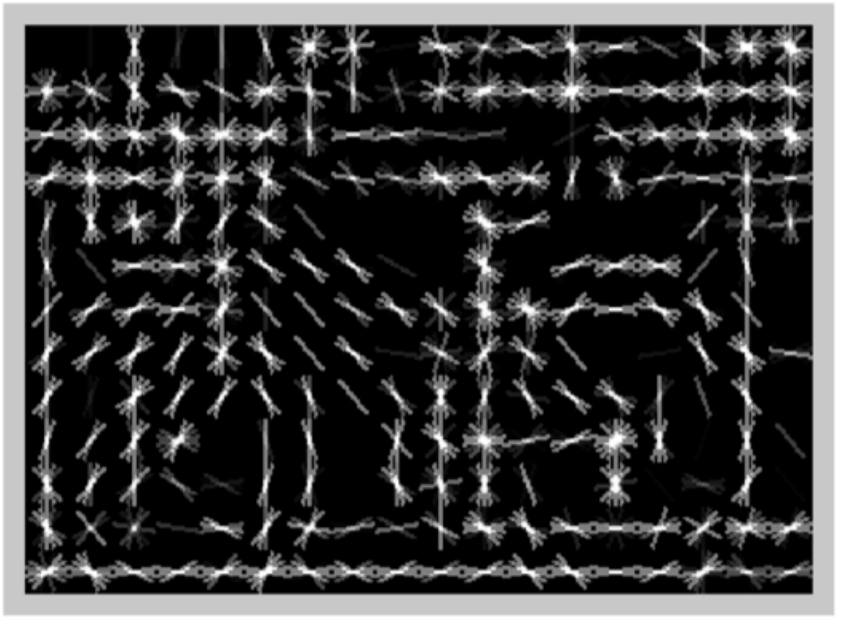
What's this?



[Dalal and Triggs, 2005]

Yup.

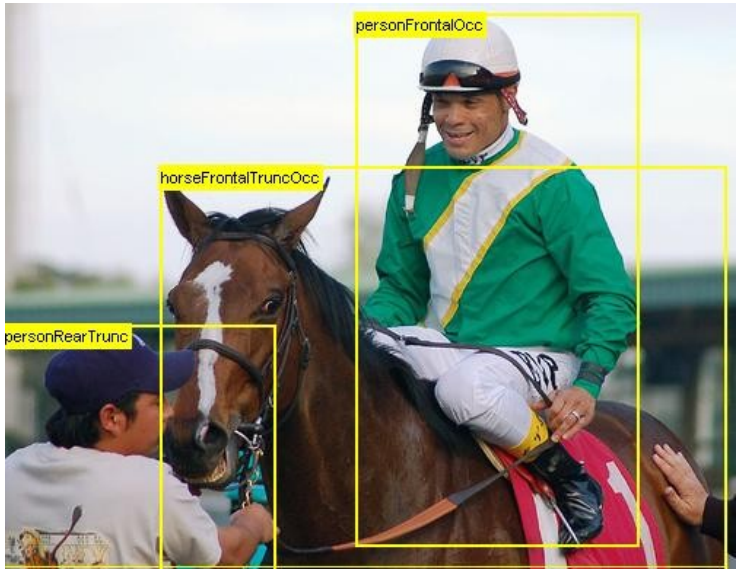
Image



HoG

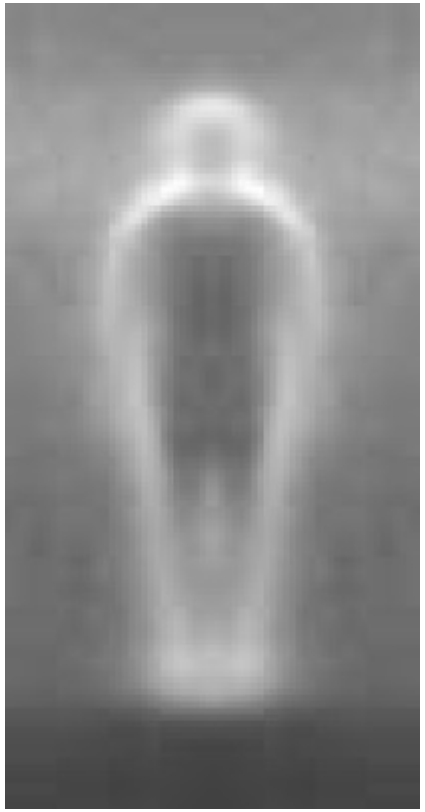
2007 PASCALVOC

20 classes

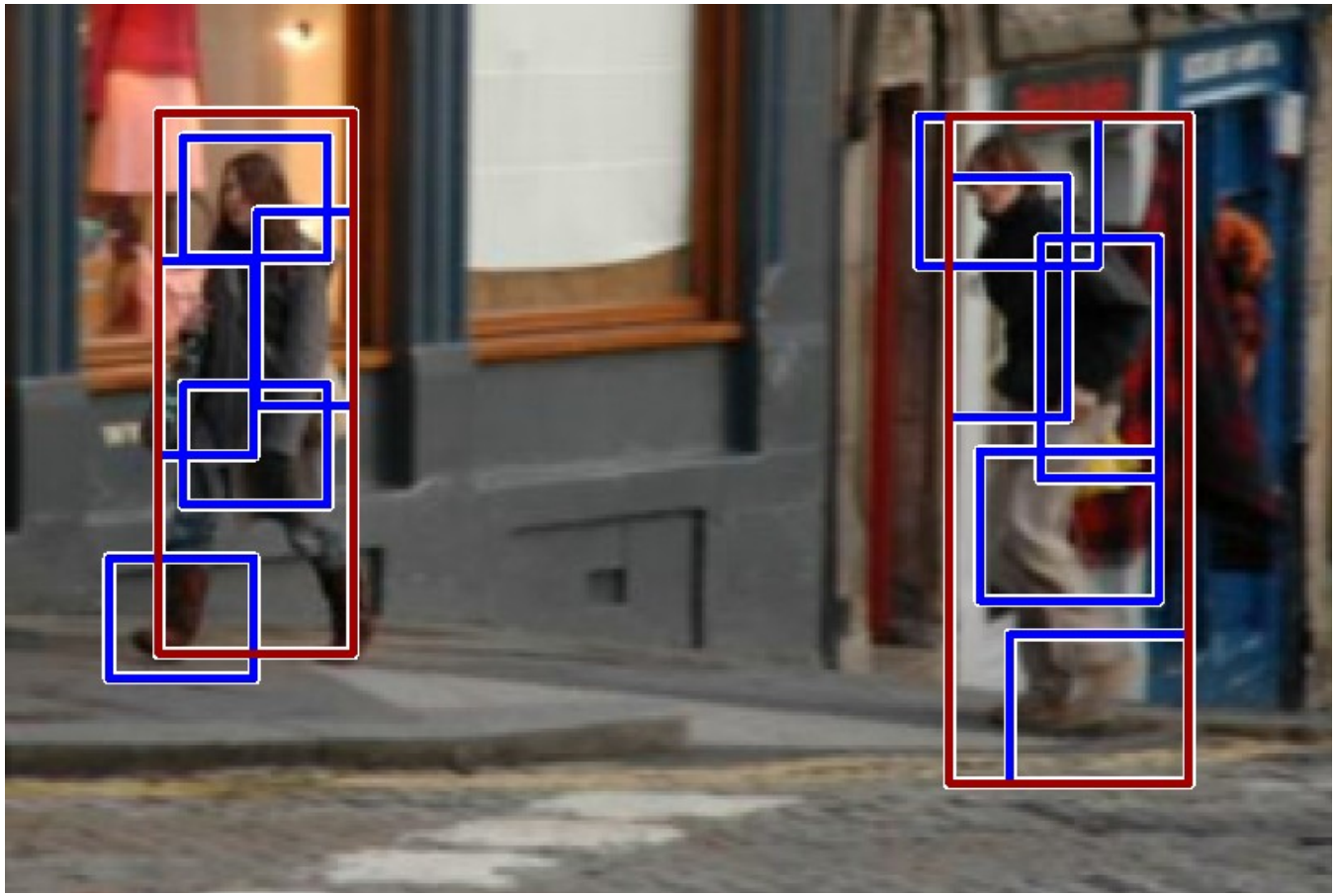


The PASCAL Visual Object Classes (VOC) Challenge, Everingham, Van Gool, Williams, Winn and Zisserman, *IJCV*, 2010

Why it is hard

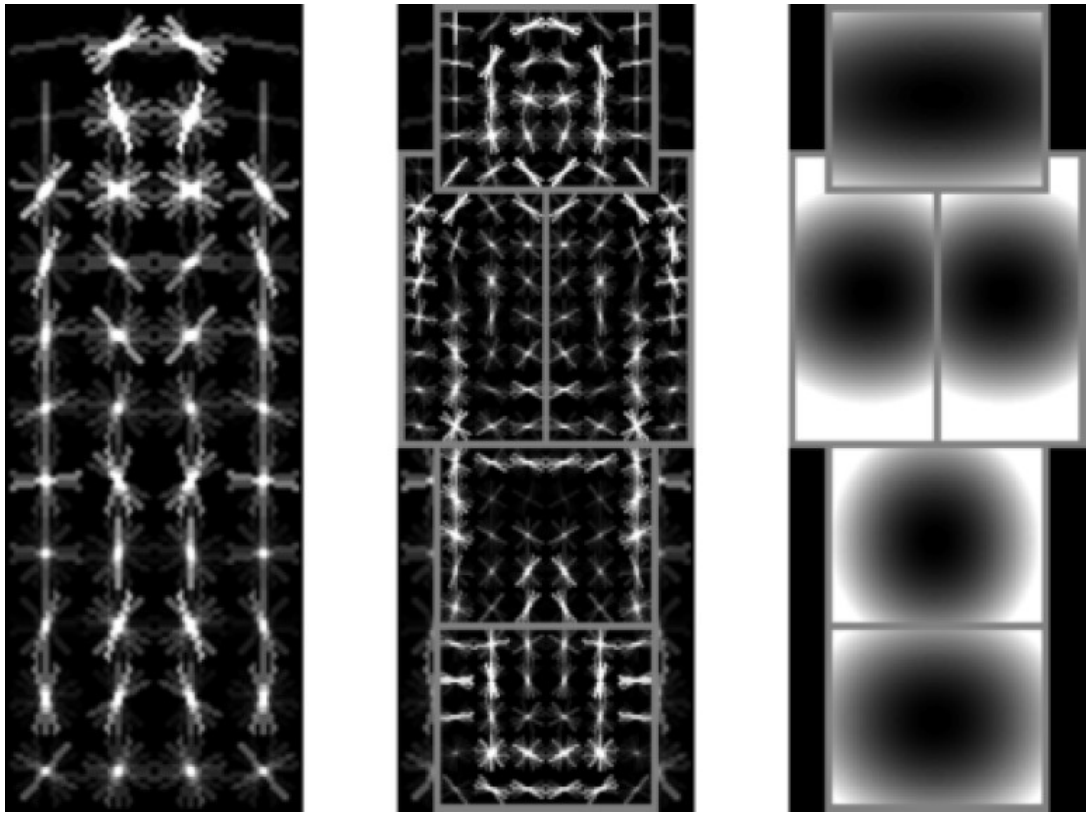


2008 DPM (Deformable parts model)



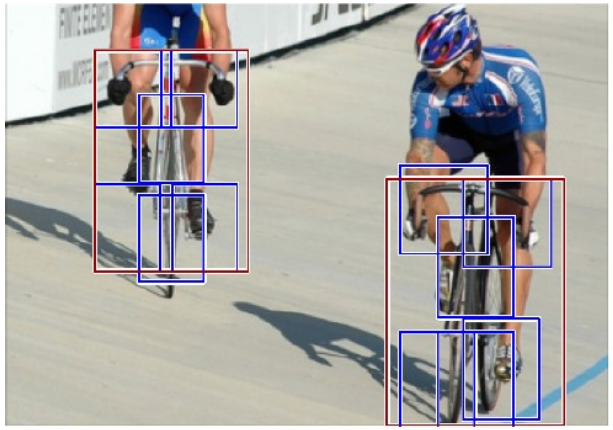
Object Detection with Discriminatively Trained Part Based Model,
Felzenszwalb, Girshick, McAllester and Ramanan, *PAMI*, 2010

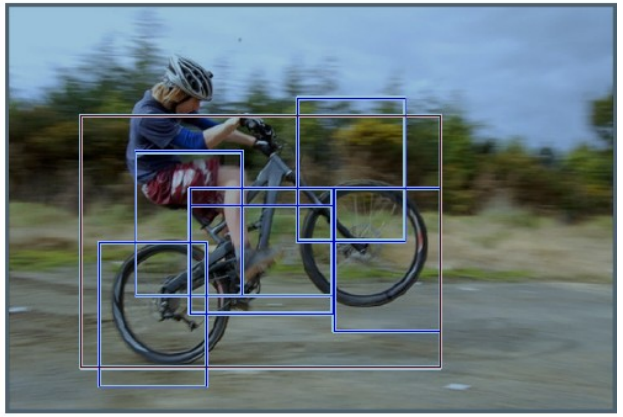
2008 DPM (Deformable parts model)

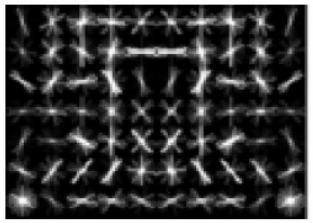


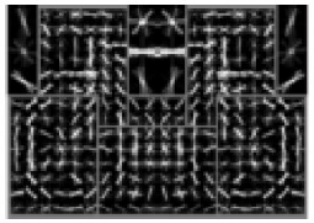
Object Detection with Discriminatively Trained Part Based Model,
Felzenszwalb, Girshick, McAllester and Ramanan, *PAMI*, 2010

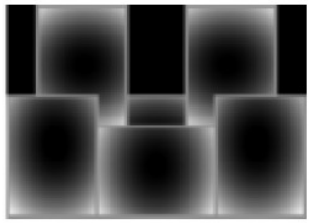
Multiple components

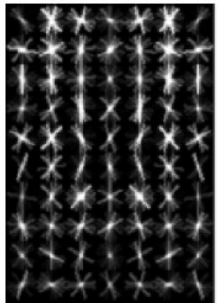


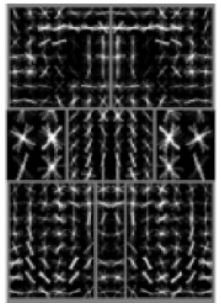


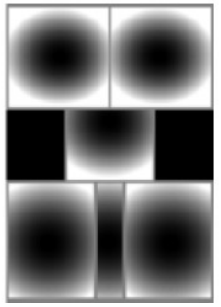












Problems with Visual Categories

- A lot of categories are functional

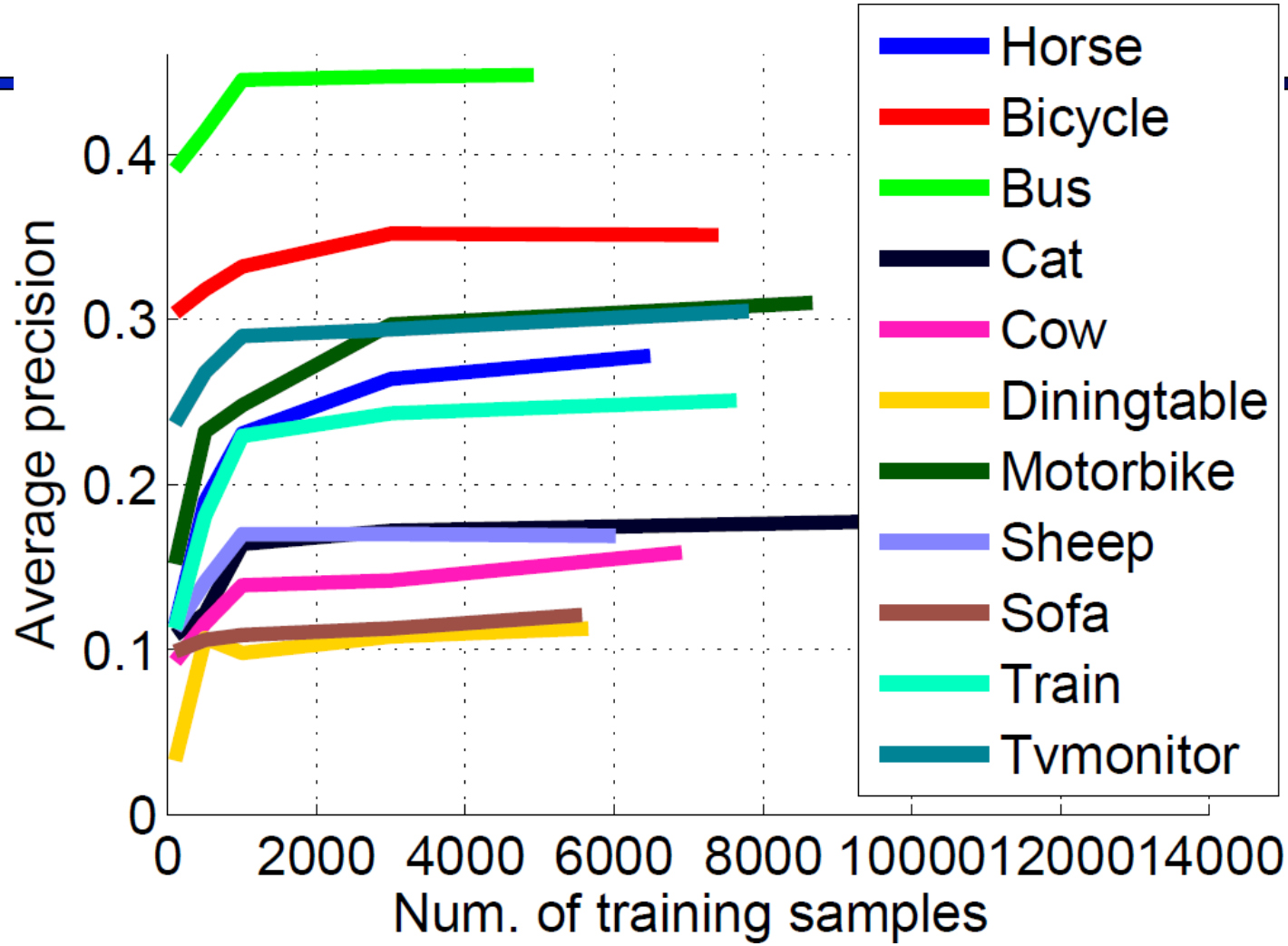
Char



- World is too varied


- Categories are 3D, but images are 2D

car

Do We Need More Training Data or Better Models for Object Detection?
Zhu, Vondrick, Ramanan, Fowlkes, *BMVC* 2012.

IMAGENET

www.image-net.org

22K categories and **14M** images

- Animals
  - Bird
  - Fish
  - Mammal
  - Invertebrate
- Plants
  - Tree
  - Flower
- Food
- Materials
- Structures
- Artifact
  - Tools
  - Appliances
  - Structures
- Person
- Scenes
  - Indoor
  - Geological Formations
- Sport Activities

Deng, Dong, Socher, Li, Li, & Fei-Fei, 2009

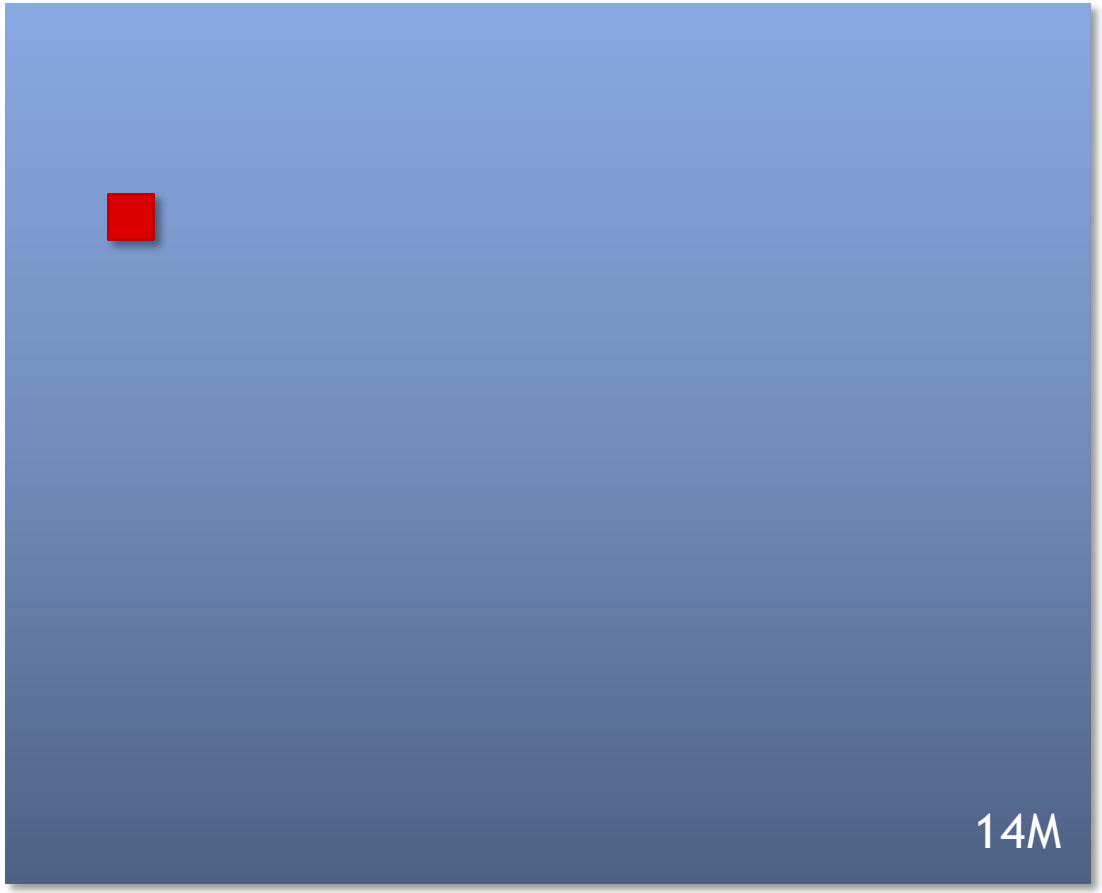
2009 ImageNet

22K categories, 14M images

ImageNet: A Large–Scale Hierarchical Image Database,
Deng, Dong, Socher, Li, Li and Fei-Fei, *CVPR*, 2009

Images

2009 2012



ImageNet

Categories

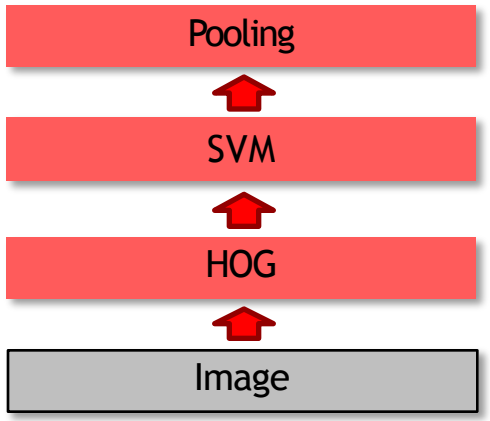
2009 2012

ImageNet

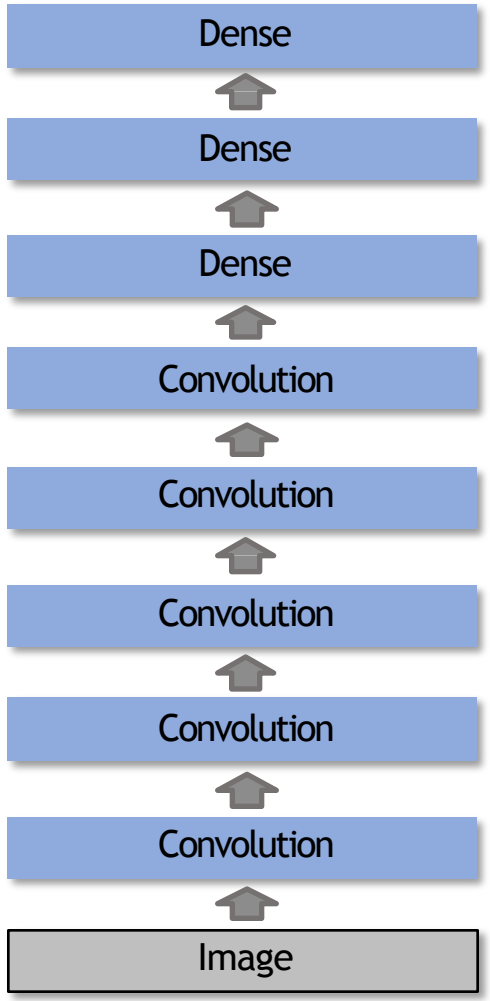
22K

Algorithms

2009



2012

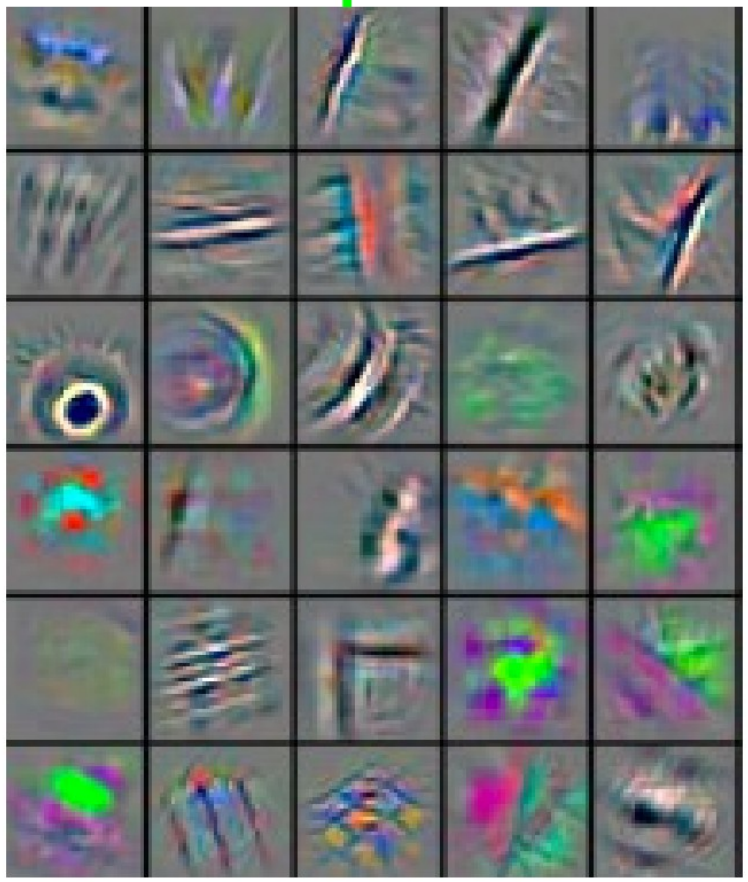


Low-Level Feature

Mid-Level Feature

High-Level Feature

Trainable Classifier



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

IMAGENET Large Scale Visual Recognition Challenge

Year 2010

NEC-UIUC

Dense grid descriptor: HOG, LBP

Coding: local coordinate, super-vector

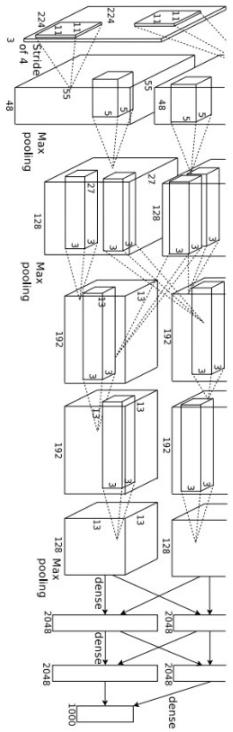
Pooling, SPM

Linear SVM

[Lin CVPR 2011]

Year 2012

SuperVision



[Krizhevsky NIPS 2012]

Year 2014

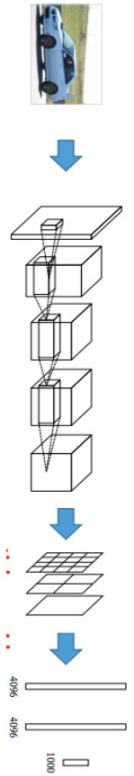
GoogLeNet

[Szegedy arxiv 2014]

VGG

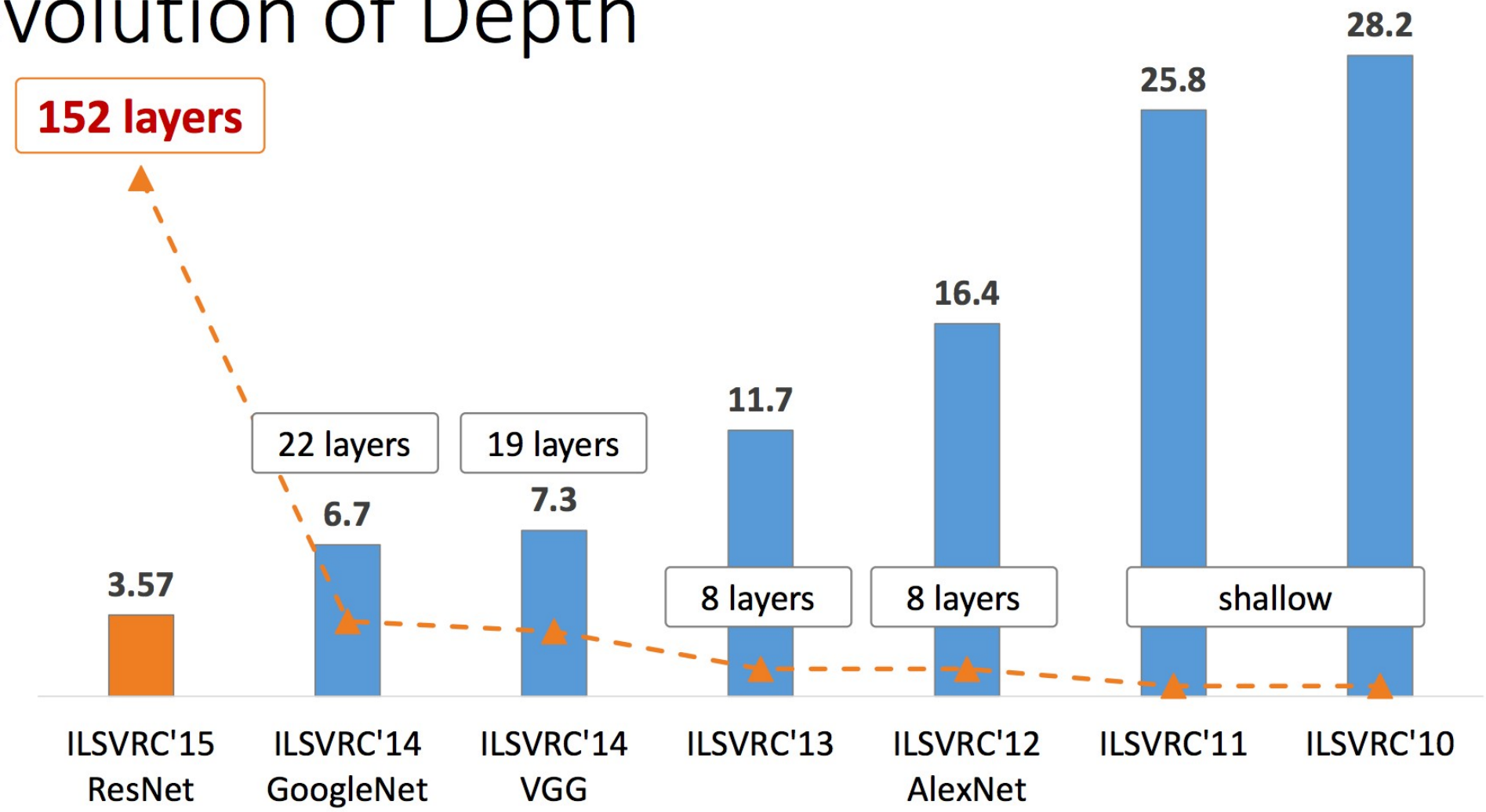
[Simonyan arxiv 2014]

MSRA



[He arxiv 2014]

Revolution of Depth

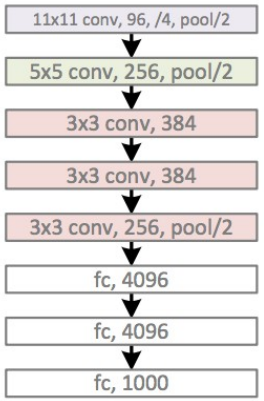


Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". arXiv 2015.

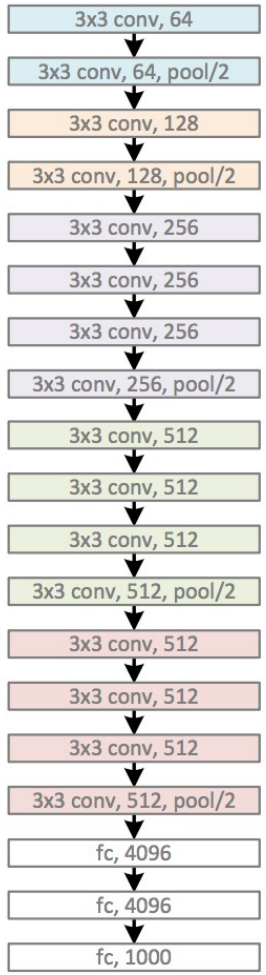
Microsoft Research

Revolution of Depth

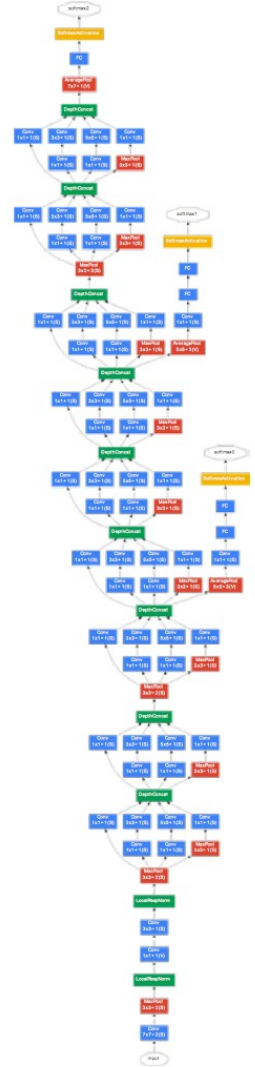
AlexNet, 8 layers
(ILSVRC 2012)



VGG, 19 layers
(ILSVRC 2014)



GoogleNet, 22 layers
(ILSVRC 2014)



ICCV15 International Conference on Computer Vision

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. arXiv 2015.

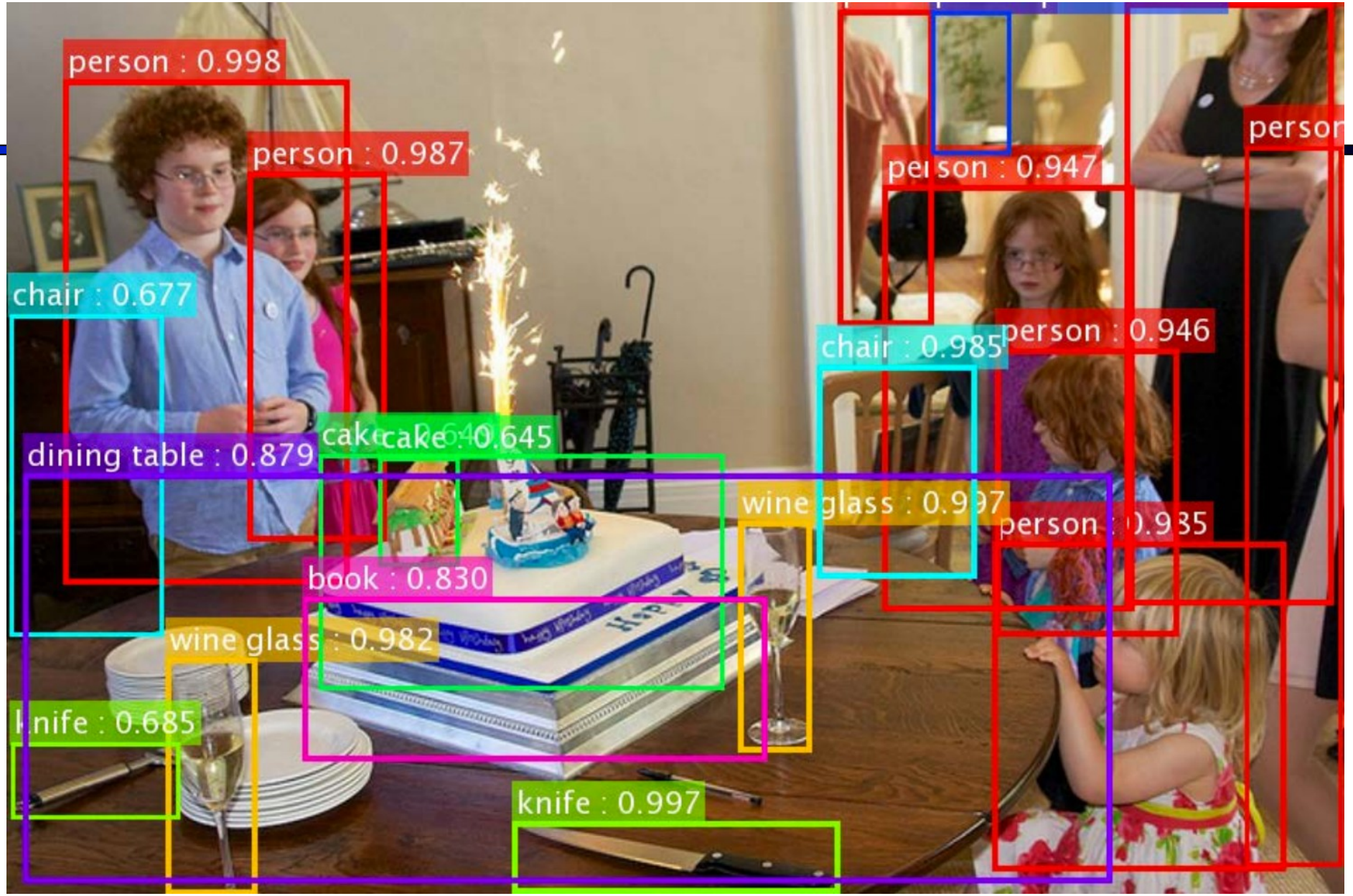
Revolution of Depth

AlexNet, 8 layers
(ILSVRC 2012)

VGG, 19 layers
(ILSVRC 2014)

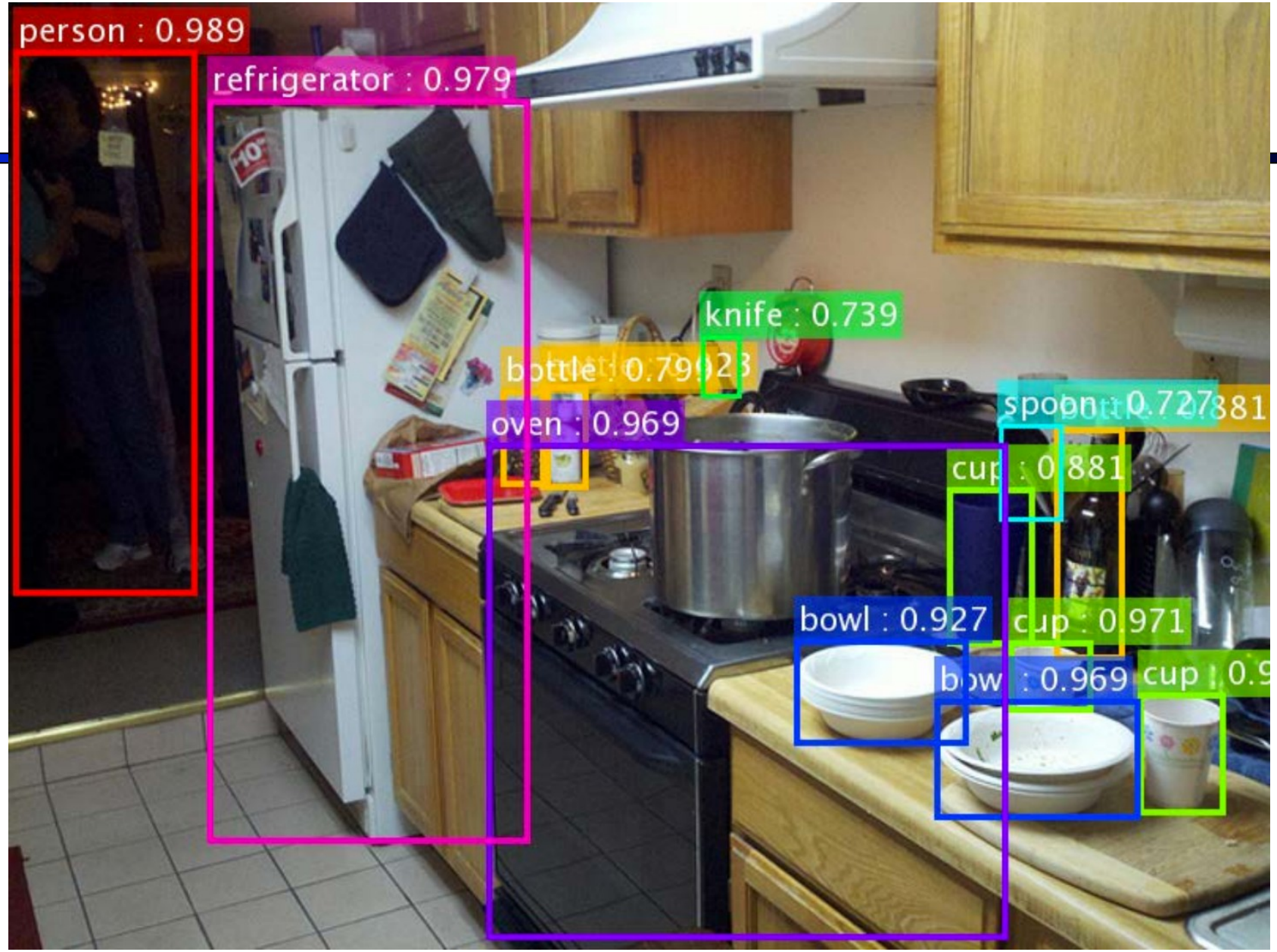
ResNet, 152 layers
(ILSVRC 2015)

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. arXiv 2015.



*the original image is from the COCO dataset

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. arXiv 2015.
Shaoqing Ren, Kaiming He, Ross Girshick, & Jian Sun. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. NIPS 2015.



*the original image is from the COCO dataset

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. arXiv 2015.
Shaoqing Ren, Kaiming He, Ross Girshick, & Jian Sun. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. NIPS 2015.

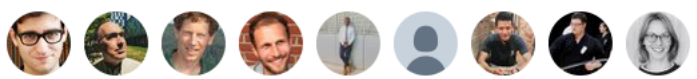
Oren Etzioni
@etzioni

Following

The winner of the 2014 ImageNet competition had 4 million parameters, while the winner of the 2017 challenge had 145.8 million parameters - a 36X increase in three years. source: @jackclarkSF Shall we increase parameters by another 36X, or solve more interesting problems?1

9:21 PM - 27 Nov 2018

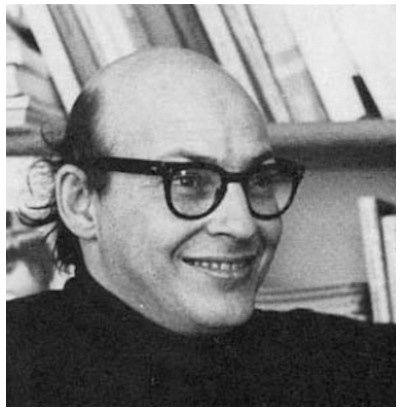
63 Retweets **222** Likes



12 63 222

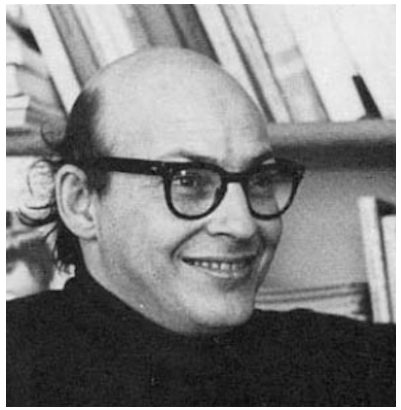
Going beyond categorization…

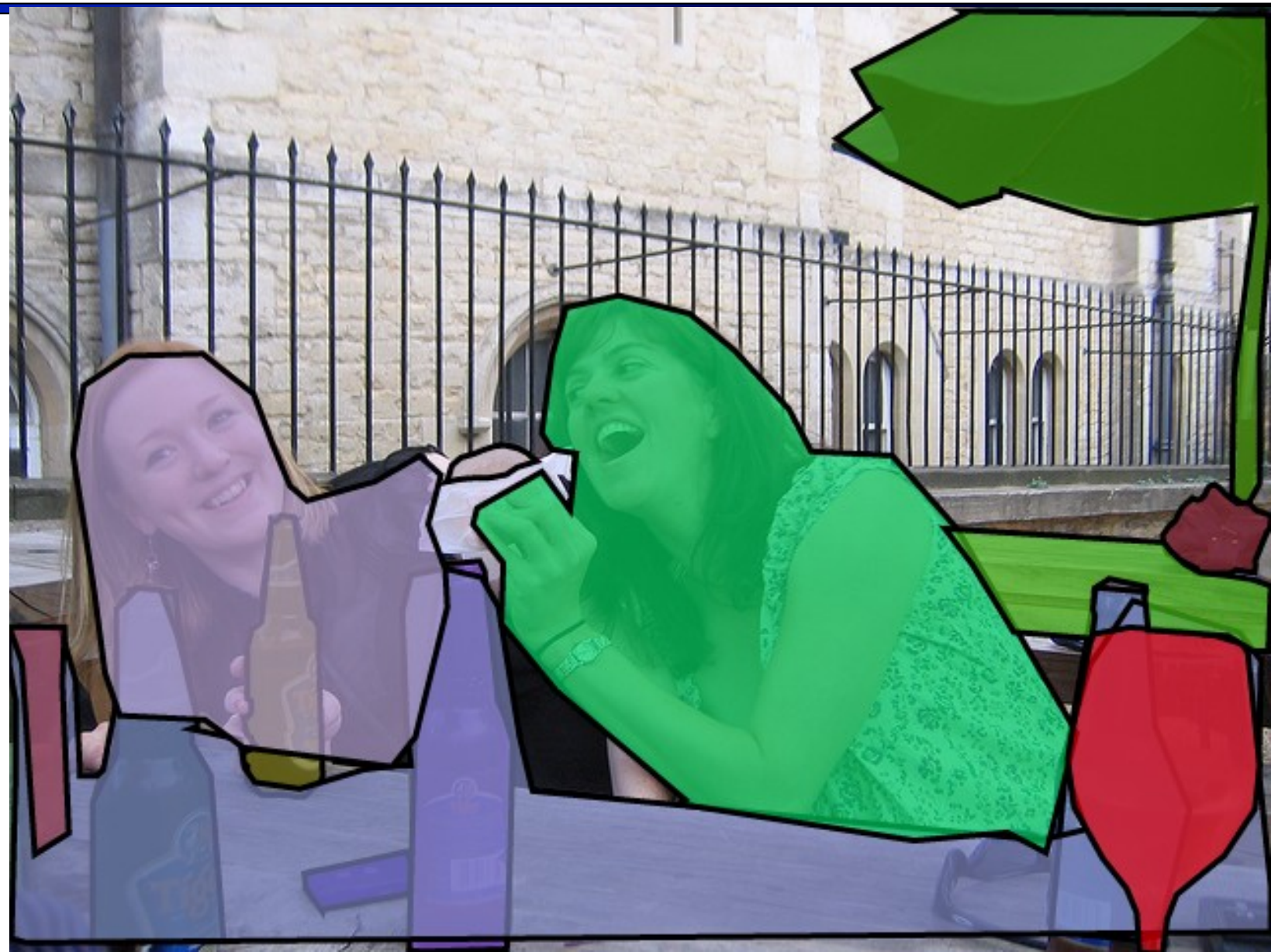
"Connect a television camera to a computer and **get the machine to describe what it sees.**"



Going beyond categorization…

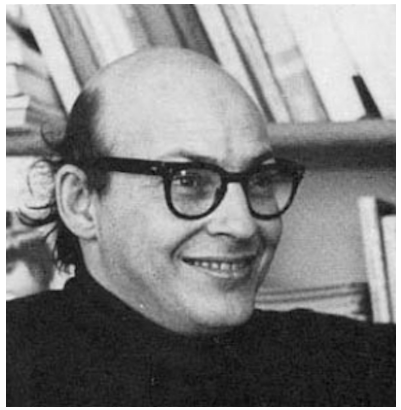
"Connect a television camera to a computer and **get the machine to describe what it sees."**





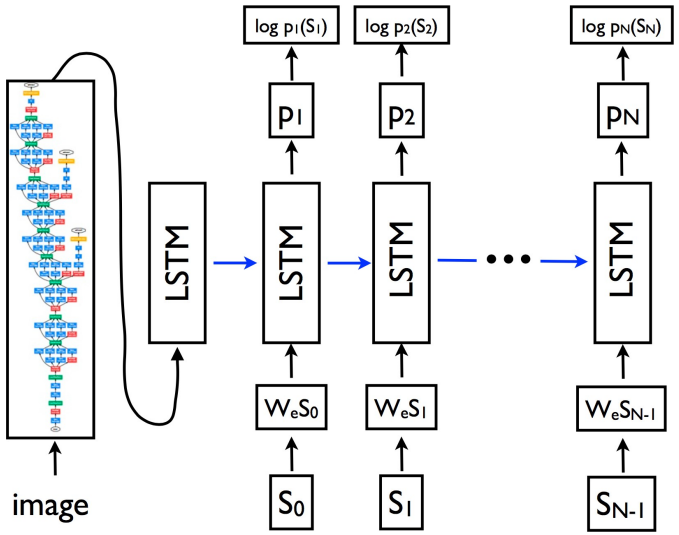
Going beyond categorization…

"Connect a television camera to a computer and **get the machine to describe what it sees.**"



two girls sitting at a table smiling and eating and drinking.
a woman is eating a doughnut and drinking beer.
there are two woman drinking beers and eating food
a woman leaning into another woman as she holds a sandwich towards her.
two ladies are enjoying beer and treats at the table.

Going beyond categorization…



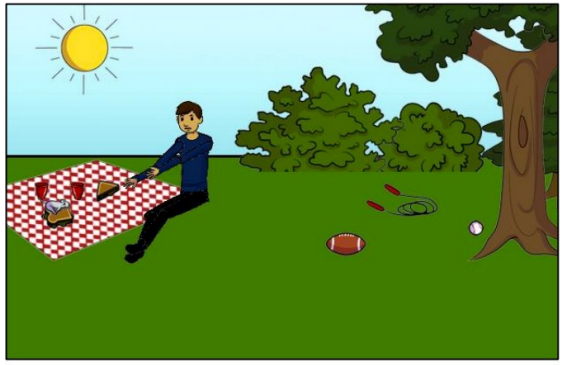
VQA: Visual Question Answering

www.visualqa.org

Stanislaw Antol*, Aishwarya Agrawal*, Jiasen Lu, Margaret Mitchell,
Dhruv Batra, C. Lawrence Zitnick, Devi Parikh

What color are her eyes?
What is the mustache made of?

How many slices of pizza are there?
Is this a vegetarian pizza?

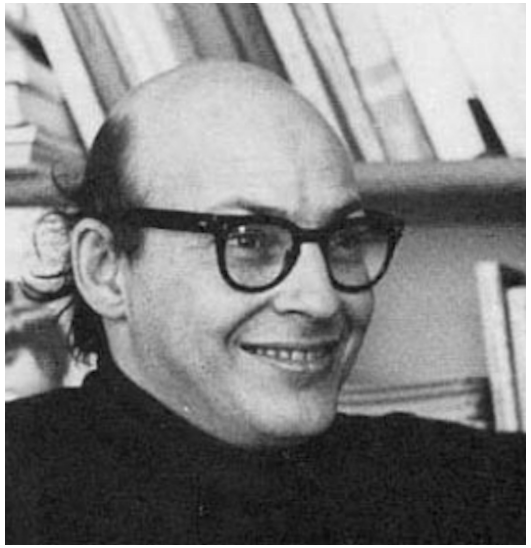


Is this person expecting company?
What is just under the tree?

Does it appear to be rainy?
Does this person have 20/20 vision?

1966

"Connect a television camera to a computer and get the machine to describe what it sees."



Marvin Minsky
Turing award, 1969

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".