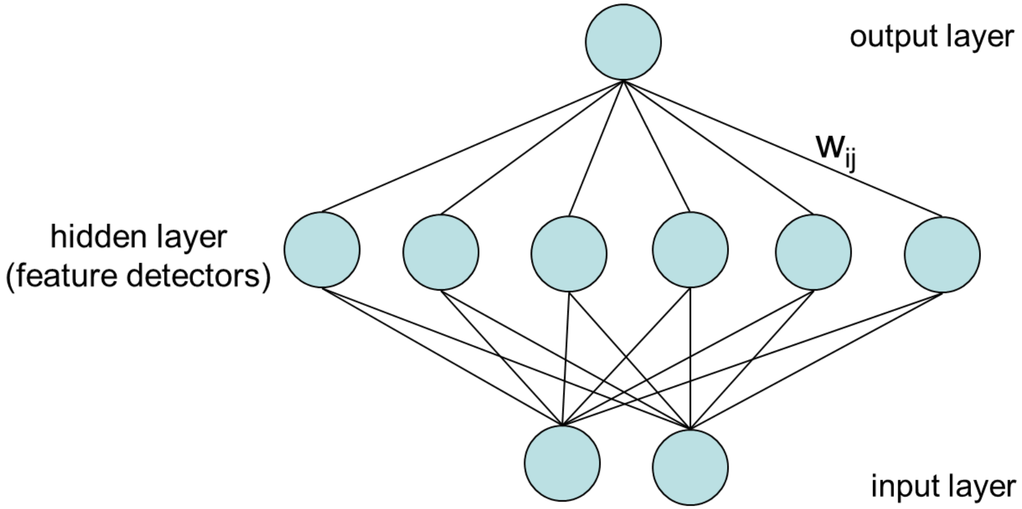
CS535: Deep Learning
1. Introduction

Winter 2018

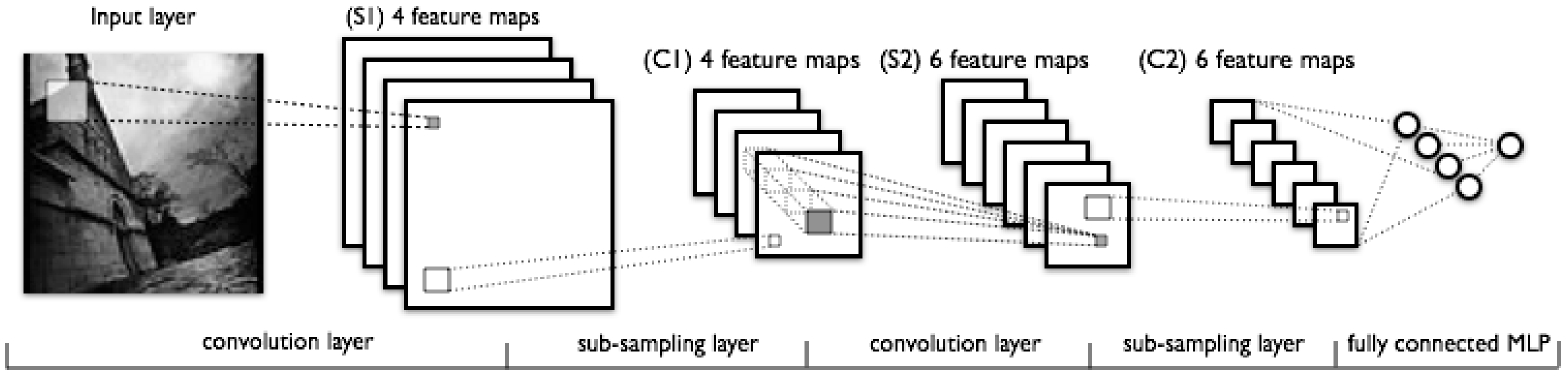
Fuxin Li

With materials from Pierre Baldi, Geoffrey Hinton, Andrew Ng, Honglak Lee, Aditya Khosla, Joseph Lim

Cutting Edge of Machine Learning: Deep Learning in Neural Networks

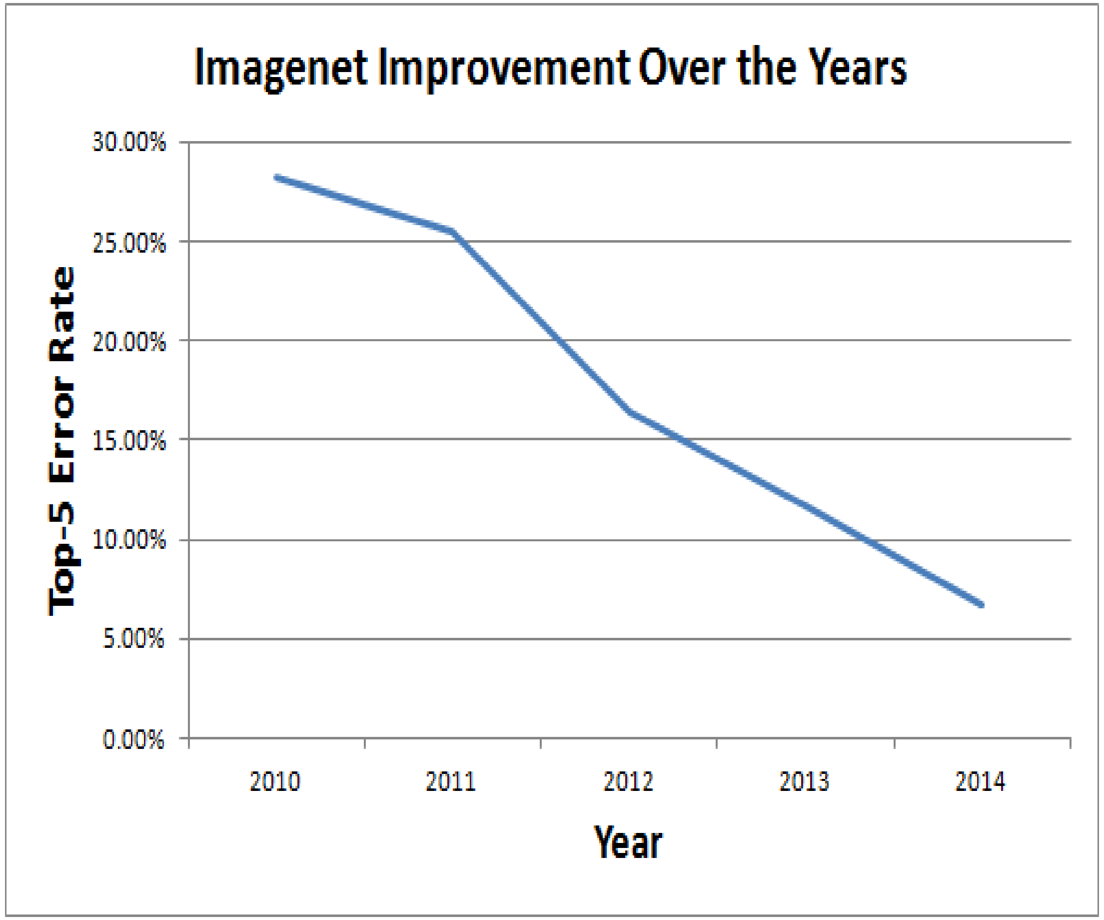


Engineering applications:
- Computer vision
- Speech recognition
- Natural Language Understanding
- Robotics



Computer Vision – Image Classification

- Imagenet
  - Over 1 million images, 1000 classes, different sizes, avg 482x415, color
- 16.42% Deep CNN dropout in 2012
- 6.66% 22 layer CNN (GoogLeNet) in 2014
- 3.6% (Microsoft Research Asia) super-human performance in 2015



Sources: Krizhevsky et al ImageNet Classification with Deep Convolutional Neural Networks, Lee et al Deeply supervised nets 2014, Szegedy et al, Going Deeper with convolutions, ILSVRC2014, Sanchez & Perronnin CVPR 2011, http://www.clarifai.com/ Benenson, http://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results.html

Speech recognition on Android (2013)

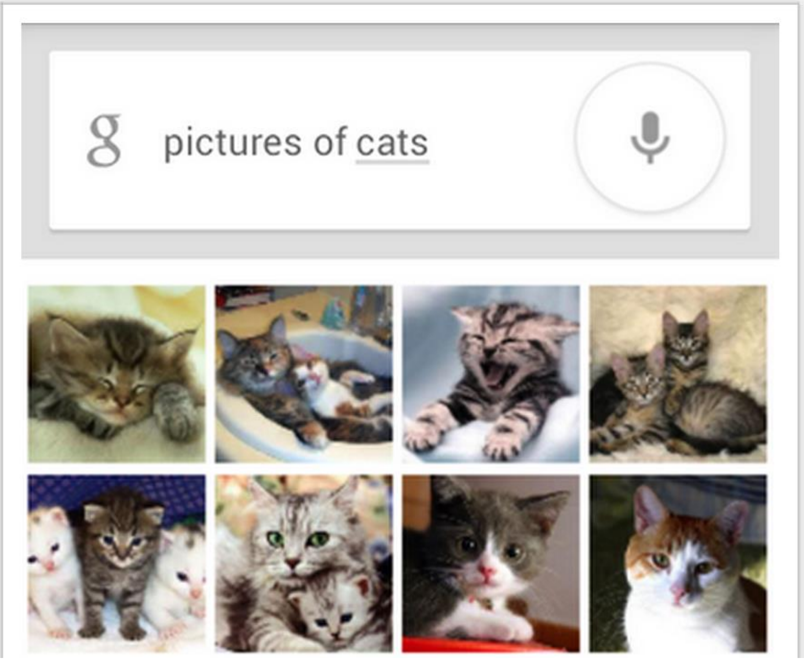
AUG
6

Speech Recognition and Deep Learning

Posted by Vincent Vanhoucke, Research Scientist, Speech Team

The New York Times recently published an article about Google's large scale deep learning project, which learns to discover patterns in large datasets, including... cats on YouTube!

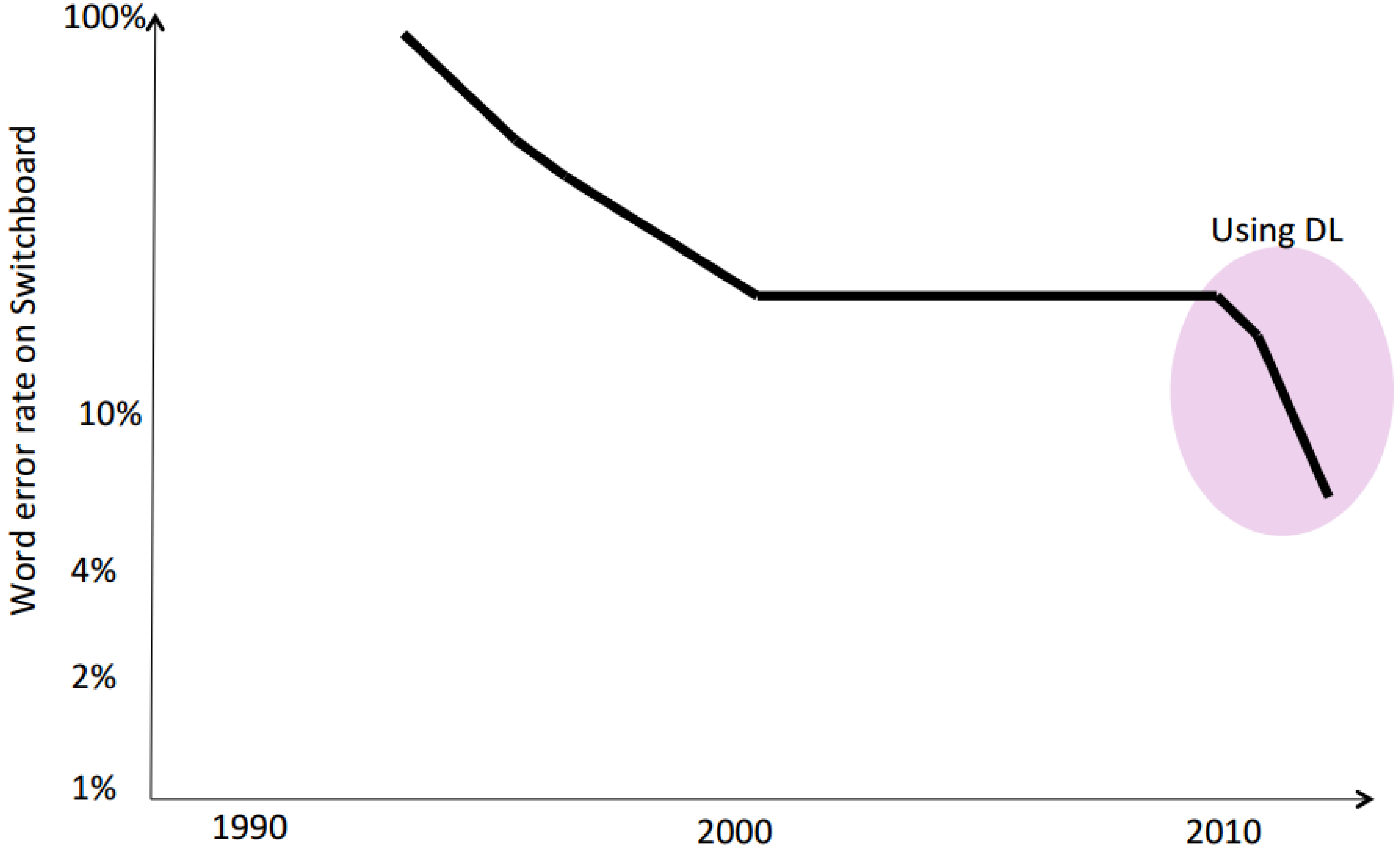
What's the point of building a gigantic cat detector you might ask? When you combine large amounts of data, large-scale distributed computing and powerful machine learning algorithms, you can apply the technology to address a large variety of practical problems.



With the launch of the latest Android platform release, Jelly Bean, we've taken a significant step towards making that technology useful: when you speak to your Android phone, chances are, you are talking to a neural network trained to recognize your speech.

Using neural networks for speech recognition is nothing new: the first proofs of concept were developed in the late

Impact on speech recognition



10th Community Wide Experiment on the
Critical Assessment of Techniques for Protein Structure Prediction

Menu

- Home
- FORCASP Forum
- PC Login
- PC Registration
- CASP Experiments
  - CASP ROLL
    - Home
    - My CASP ROLL profile
    - Targets
      - Target List
      - Target Submission
  - CASP10 (2012)
    - Home
    - My CASP10 profile
    - Targets
    - Results
    - CASP10 in numbers
  - CASP9 (2010)
  - CASP8 (2008)
  - CASP7 (2006)
  - CASP6 (2004)
  - CASP5 (2002)
  - CASP4 (2000)
  - CASP3 (1998)
  - CASP2 (1996)
  - CASP1 (1994)
- Initiatives
- Data Archive
- Local Services
- Proceedings
- Feedback
- Assessors
- People
- Community Resources

RR Analysis

Results Home | Table Browser | Quality Assessment Results | RR Assessment Results

Summary | Detailed Analysis | Help

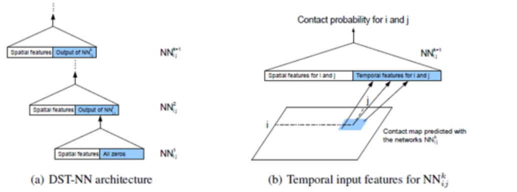
The table summarizes the evaluation of predictions in 'RR' category.
The analysis was performed at per domains basis; only predictions for domains classified as "FM", "TBM/FM", "TBM hard" were considered.
The groups were ranked according to sum of average Z-scores for two measures Acc and Xd.
The per target Z-scores were recalculated from the "cleaned" distributions, where the outlier predictions (below mean - 2 std dev) were eliminated.

- **Domain classification:**
  - FM
  - TBM/FM
  - TBM hard (max gdt_ts < 50)
- **Contact Range:** long
- **List Size:** L/5

#	GR#	GR Name	Count domains	AVG Acc	AVG Zscore Acc	AVG Xd	AVG Zscore Xd	Zscore Acc + Zscore Xd
1.	222 s	MULTICOM-CONSTRUCT	14	19.41	0.58	12.08	0.77	1.35
2.	305 s	IGBteam	15	19.22	0.72	10.19	0.58	1.30
3.	424 s	MULTICOM-NOVEL	14	20.39	0.50	10.32	0.72	1.22
4.	125 s	MULTICOM-REFINE	14	21.35	0.51	10.29	0.70	1.21
5.	413 s	ZHOU-SPARKS-X	12	12.26	0.62	8.26	0.59	1.21
6.	113 s	SAM-T08-server	11	16.13	0.72	9.44	0.47	1.19
7.	358 s	RaptorX-Roll	8	12.07	0.58	8.23	0.55	1.13
8.	314 s	ProC_S4	14	17.91	0.59	9.76	0.47	1.05
9.	087 s	Distill_roll	15	13.97	0.60	8.57	0.36	0.96
10.	489	MULTICOM	14	12.96	0.43	8.19	0.40	0.83
11.	184 s	ICOS	14	17.03	0.40	9.72	0.39	0.78
12.	396 s	ProC_S5	14	16.51	0.36	9.10	0.36	0.72
13.	381 s	SAM-T06-	10	10.98	0.37	7.94	0.31	0.68

Deep Learning

P. Di Lena, K. Nagata, and P. Baldi.
Deep Architectures for Protein Contact Map Prediction.
Bioinformatics, 28, 2449-2457, (2012)



(a) DST-NN architecture (b) Temporal input features for NN^k_{ij}

Figure 1: DST-NN architecture. (a) Overview. Each NN^k_{ij} represents a feed-forward neural network trainable by back-propagation. (b) For a pair of residues (i, j), the temporal inputs into NN^{k+1}_{ij} consist of the contact probabilities produced by the network at the previous level over a neighborhood of (i, j).

Deep Learning Applications

- Engineering:
  - Computer Vision (e.g. image classification, segmentation)
  - Speech Recognition
  - Natural Language Processing (e.g. sentiment analysis, translation)

- Science:
  - Biology (e.g. protein structure prediction, analysis of genomic data)
  - Chemistry (e.g. predicting chemical reactions)
  - Physics (e.g. detecting exotic particles)

- and many more

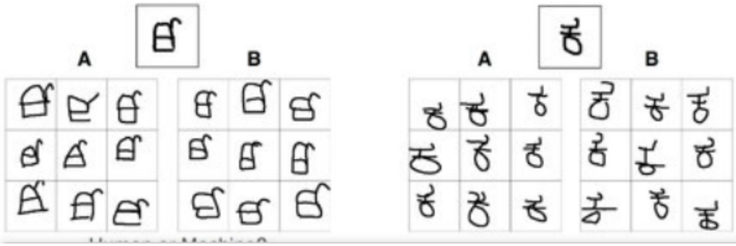
Penetration into mainstream media

The New York Times

SCIENCE

A Learning Advance in Artificial Intelligence Rivals Human Abilities

By JOHN MARKOFF DEC. 10, 2015



Computer researchers reported artificial-intelligence advances on Thursday that surpassed human capabilities for a narrow set of vision-related tasks.

Aha…

The advances reflect the intensifying focus in Silicon Valley and elsewhere on artificial intelligence.

Last month, the Toyota Motor Corporation announced a five-year, billion-dollar investment to create a research center based next to Stanford University to focus on artificial intelligence and robotics.

Also, a formerly obscure academic conference, Neural Information Processing Systems, underway this week in Montreal, has doubled in size since the previous year and has attracted a growing list of brand-name corporate sponsors, including Apple for the first time.

"There is a sellers' market right now — not enough talent to fill the demand from companies who need them," said Terrence Sejnowski, the director of the Computational Neurobiology Laboratory at the Salk Institute for Biological Studies in San Diego. "Ph.D. students are getting hired out of graduate schools for salaries that are higher than faculty members who are teaching them."

Machine learning before Deep Learning

Typical goal of machine learning

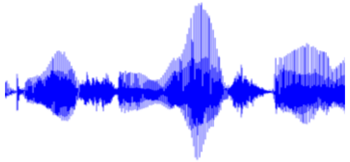
Input: X | **Output: Y**

images/video

ML →

Label: "Motorcycle"
Suggest tags
Image search
…

audio



ML →

Speech recognition
Music classification
Speaker identification
…

text

ML →

Web search
Anti-spam
Machine translation
…

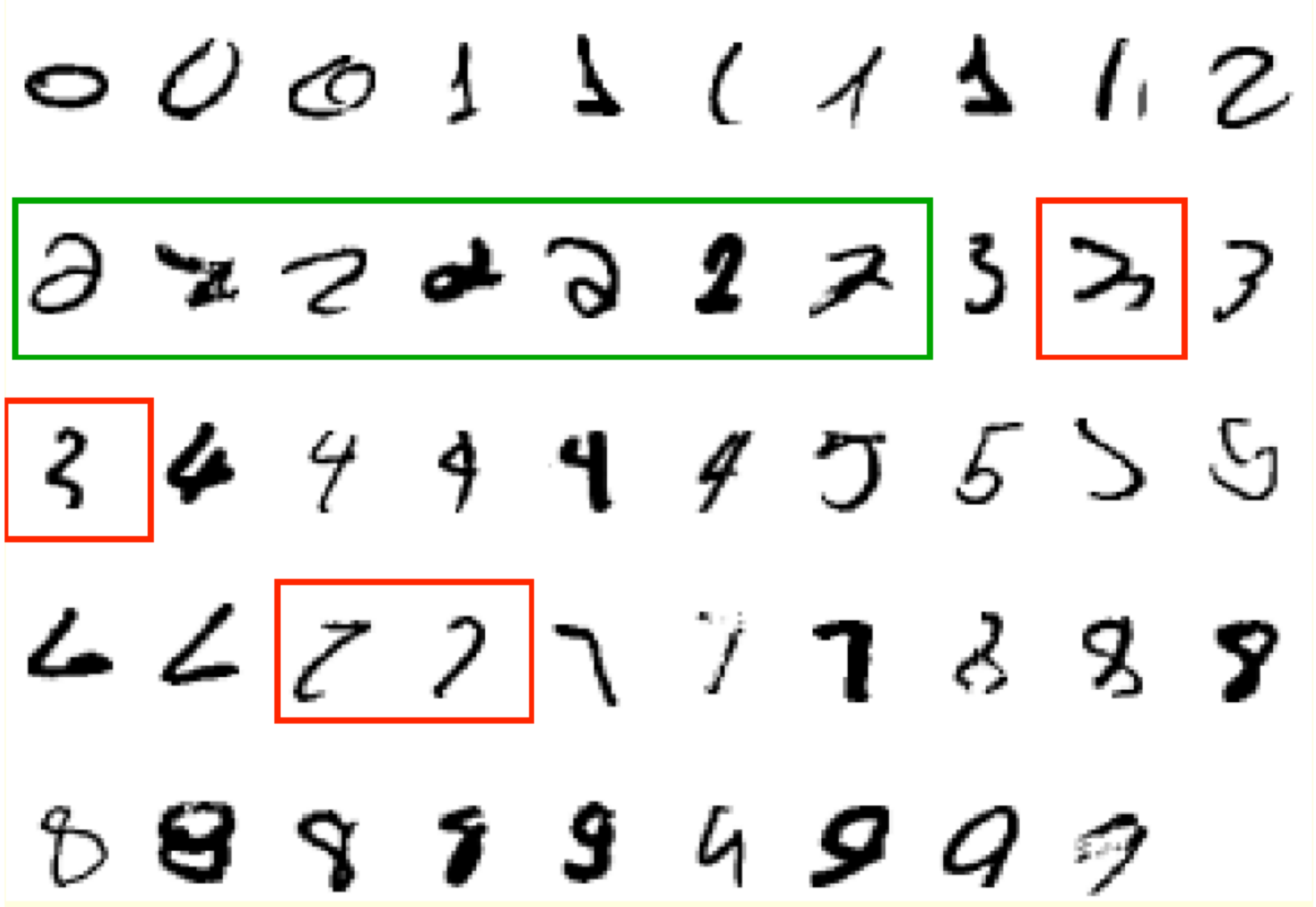
(Supervised)
Machine learning:

Find f, so that $f(X) \approx Y$

e.g.

→ ML → “motorcycle”

e.g.

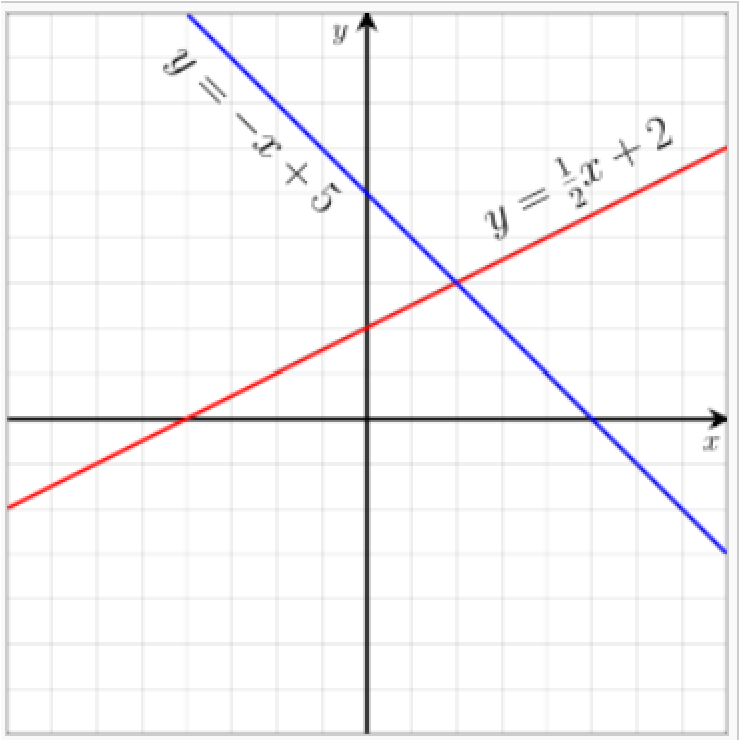


Basic ideas

- Turn every input into a vector $\boldsymbol{x}$
- Use function estimation tools to estimate the function $f(\boldsymbol{x})$
- Use observations $(x_1, y_1), (x_2, y_2), (x_3, y_3), \ldots (x_n, y_n)$ to train

Linear classifiers:

$$y = mx + b$$



- Our model is: $f(\mathbf{x_i}, \mathbf{w}, \mathbf{b}) = \mathbf{w}^{\mathbf{T}}\mathbf{x_i} + \mathbf{b}$

Usually refer $[\mathbf{w}, b]$ as $\mathbf{w}$

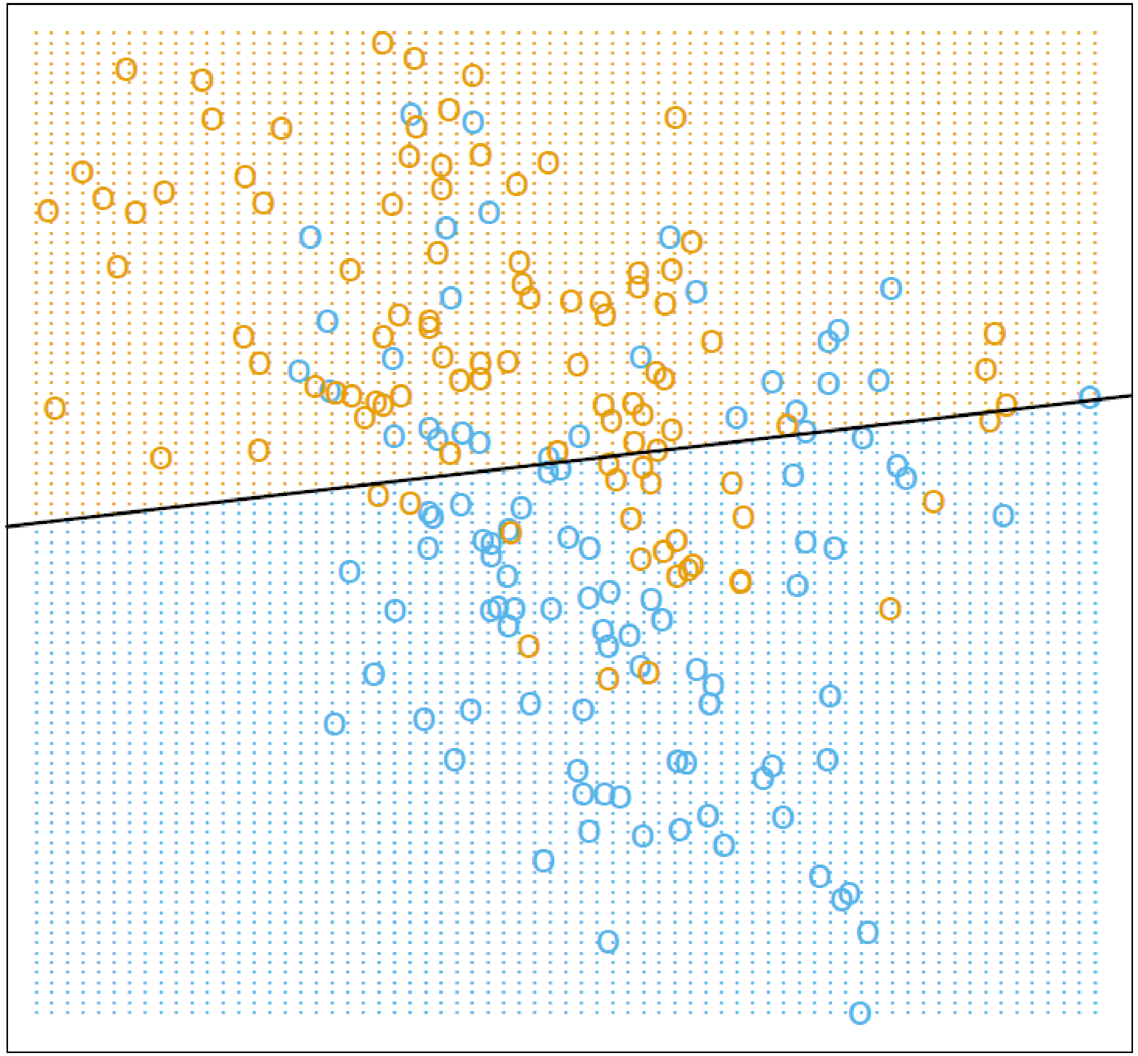
Classifier Result [1 x 1]

Parameters Vector [d x 1]

Input [d x 1]

Bias (scalar)

Linear Classifiers



What does this classifier do?

- Scores input based on linear combination of features
  - > 0 above hyperplane
  - < 0 below hyperplane

- Changes in weight vector (per classifier)
  - Rotate hyperplane

- Changes in Bias
  - Offset hyperplane from origin

Optimization of parameters

- Want to find **w** that achieves best result
- Empirical Risk Minimization principle
  - Find w that

$$\min_{\mathbf{w}} \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i; \mathbf{w}))$$

- Real goal (Bayes classifier):
  - Find **w** that $\min_{\mathbf{w}} \mathbf{E}[L_c(y_i, f(\mathbf{x}_i; \mathbf{w}))]$

$$L_c\colon \begin{cases} 1, y \neq f(x) \\ 0, y = f(x) \end{cases}$$

  - Bayes error: Theoretically optimal error

Loss Function: Some examples

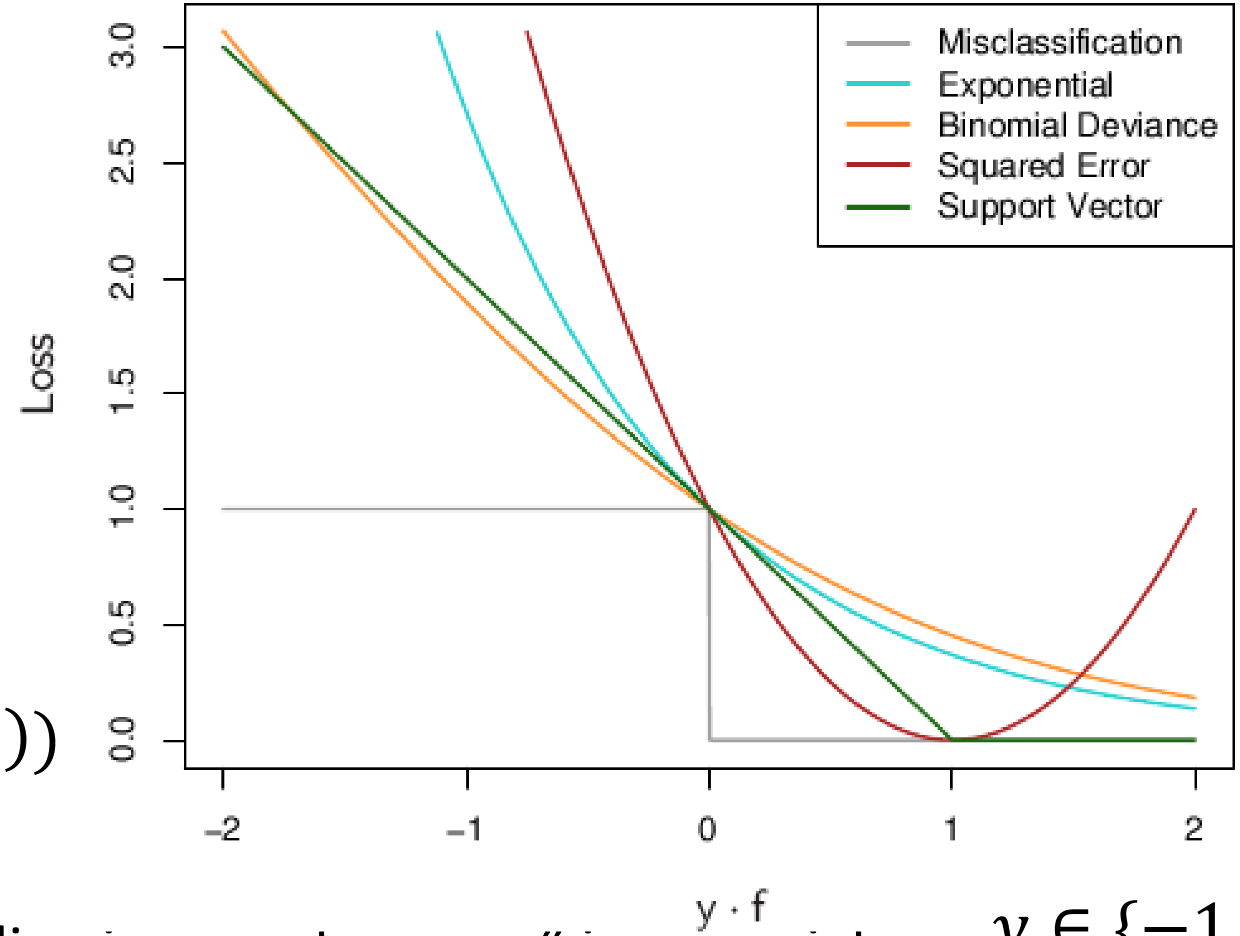
- Binary: $y \in \{-1, 1\}$
- L1/L2

  $$L_i = |y_i - \boldsymbol{w}^\top \boldsymbol{x}_i|$$

  $$L_i = \left(y_i - \boldsymbol{w}^\top \boldsymbol{x}_i\right)^2$$
- Logistic

  $$L_i = \log(1 + e^{y_i f(x_i)})$$
- Hinge (SVM)

  $$L_i = \max(0, 1 - y_i f(x_i))$$

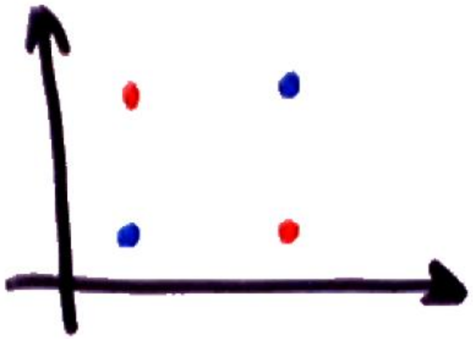


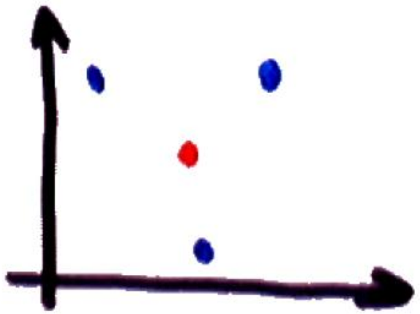
- Lots more
  - e.g. treat "most offending incorrect answer" in a special way

$y \in \{-1,1\}$

Is linear sufficient?

- Many interesting functions (as well as some non-interesting functions) not linearly separable





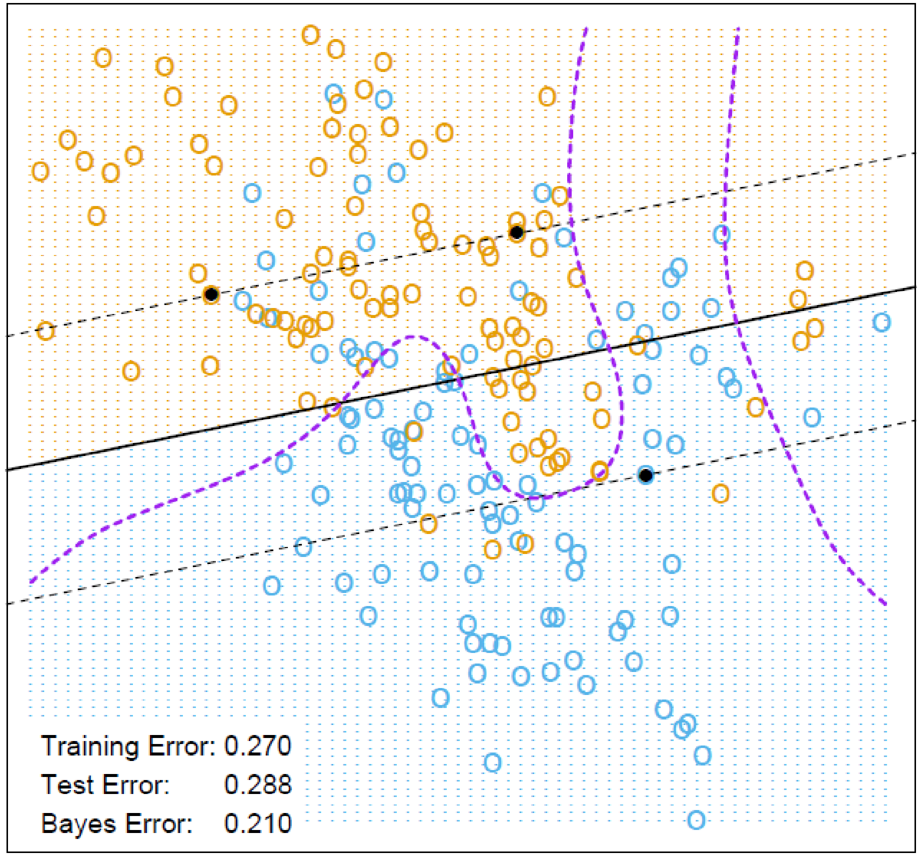
Model: Expansion of Dimensionality

- Representations:
  - Simple idea: Quadratic expansion
    $[x_1, x_2, \dots, x_d] \mapsto [x_1^2, x_2^2, \dots, x_d^2, x_1 x_2, x_1 x_3, \dots, x_{d-1} x_d]$
  - A better idea: Kernels
    $K(x, x_i) = \exp(-\beta ||x_i - x||^2)$ $\qquad f(x) = \sum_i \alpha_i K(x, x_i)$
  - Another idea: Fourier domain representations (Rahimi and Recht 2007)
    $\cos(\mathbf{w}^\top \mathbf{x} + b)$, $\mathbf{w} \sim N^d(0, \beta I), b \sim U[0,1]$
  - Another idea: Sigmoids (early neural networks)

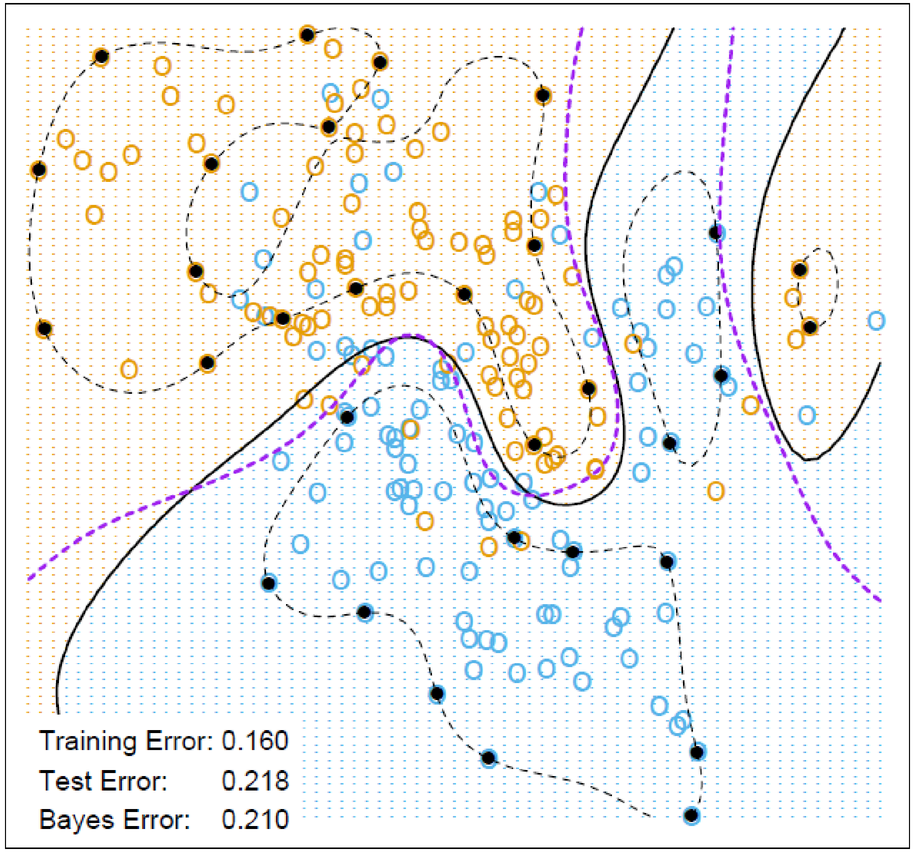
    $sigmoid(\mathbf{w}^\top \mathbf{x} + b)$, optimized $\mathbf{w}$

Distance-based Learners (Gaussian SVM)

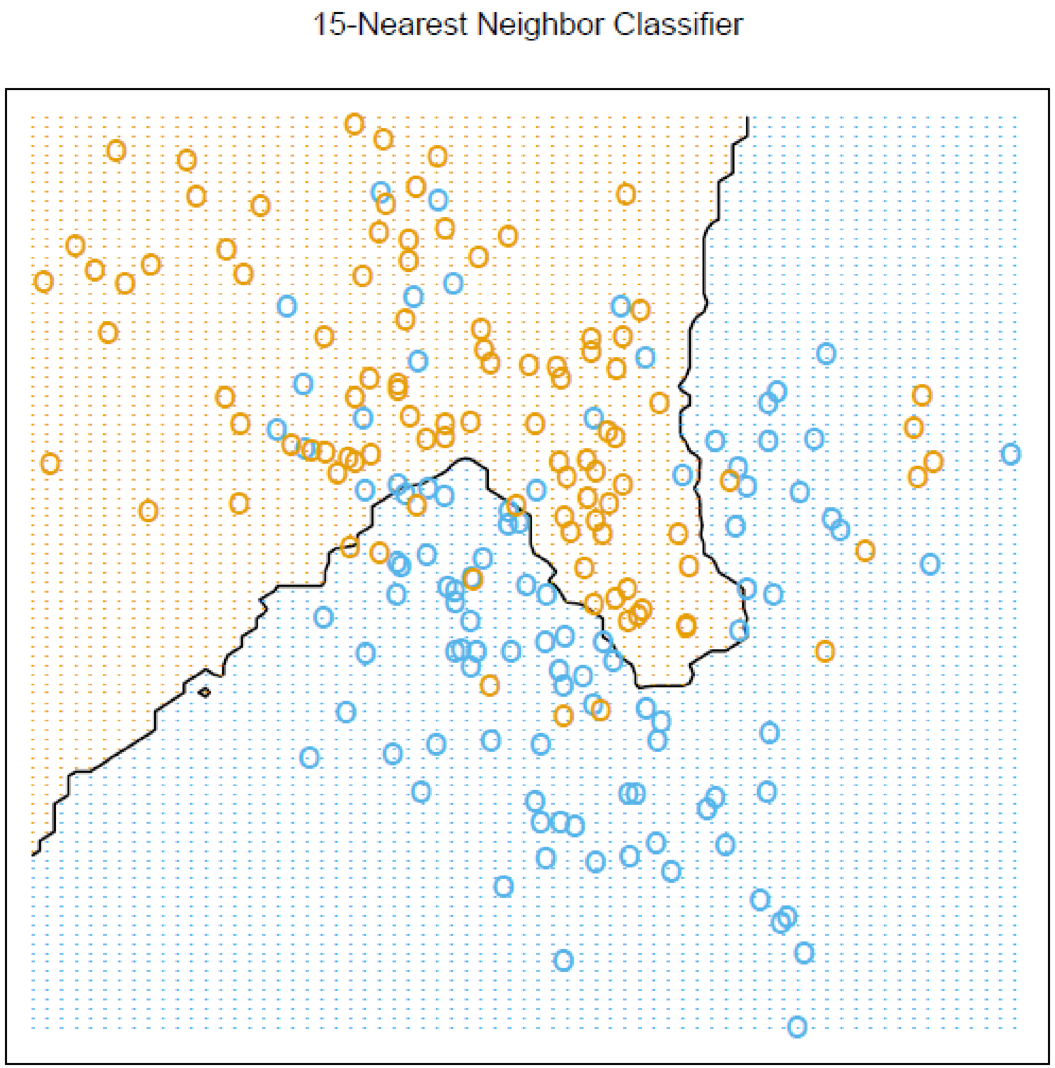
SVM: Linear

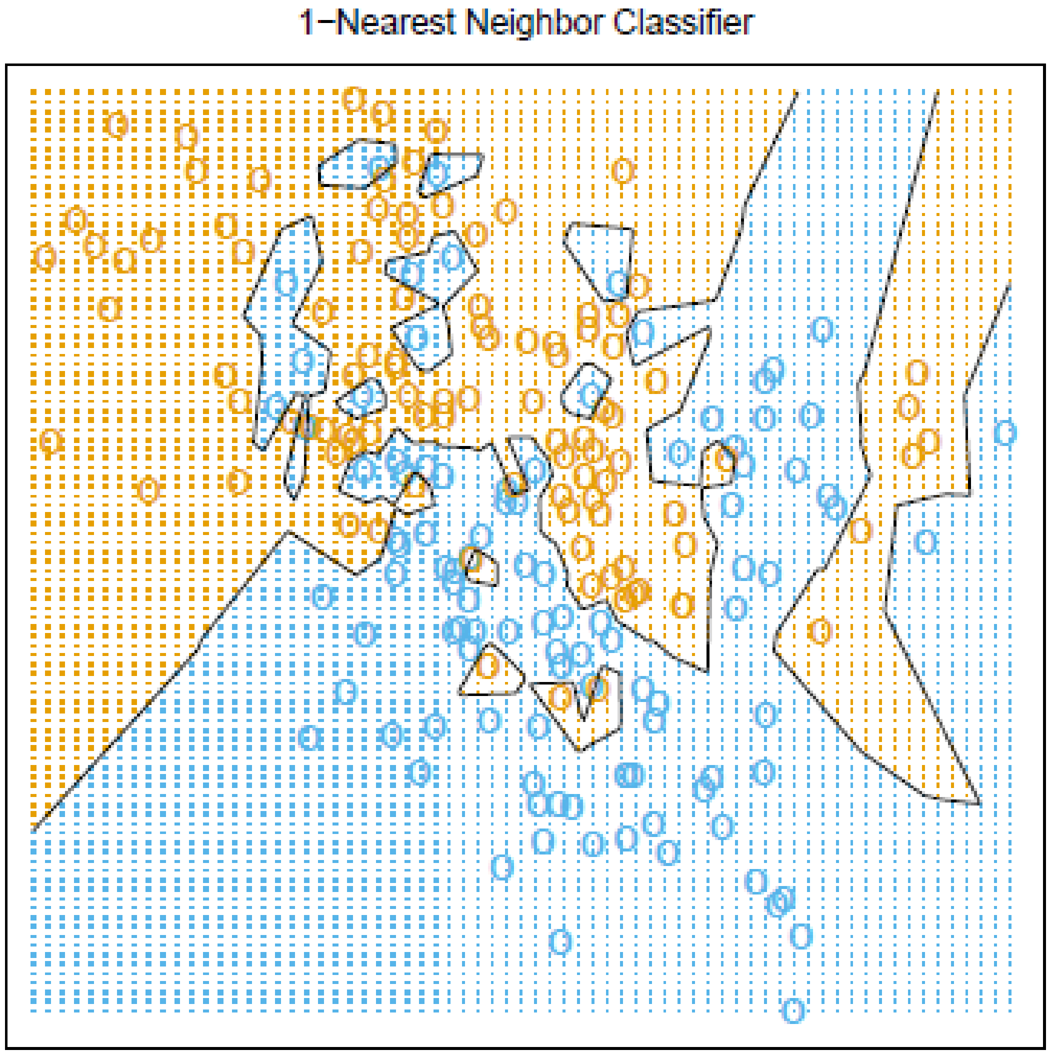


SVM - Radial Kernel in Feature Space



Distance-based Learners (kNN)





“Universal Approximators”

- Many non-linear function estimators are proven as “universal approximators”
  - Asymptotically (training examples -> infinity), they are able to recover the true function with a low error
  - They also have very good learning rates with finite samples
  - For almost all sufficiently smooth functions
- This includes:
  - Kernel SVMs
  - 1-Hidden Layer Neural Networks
- Essentially means we are “**done**” with machine learning

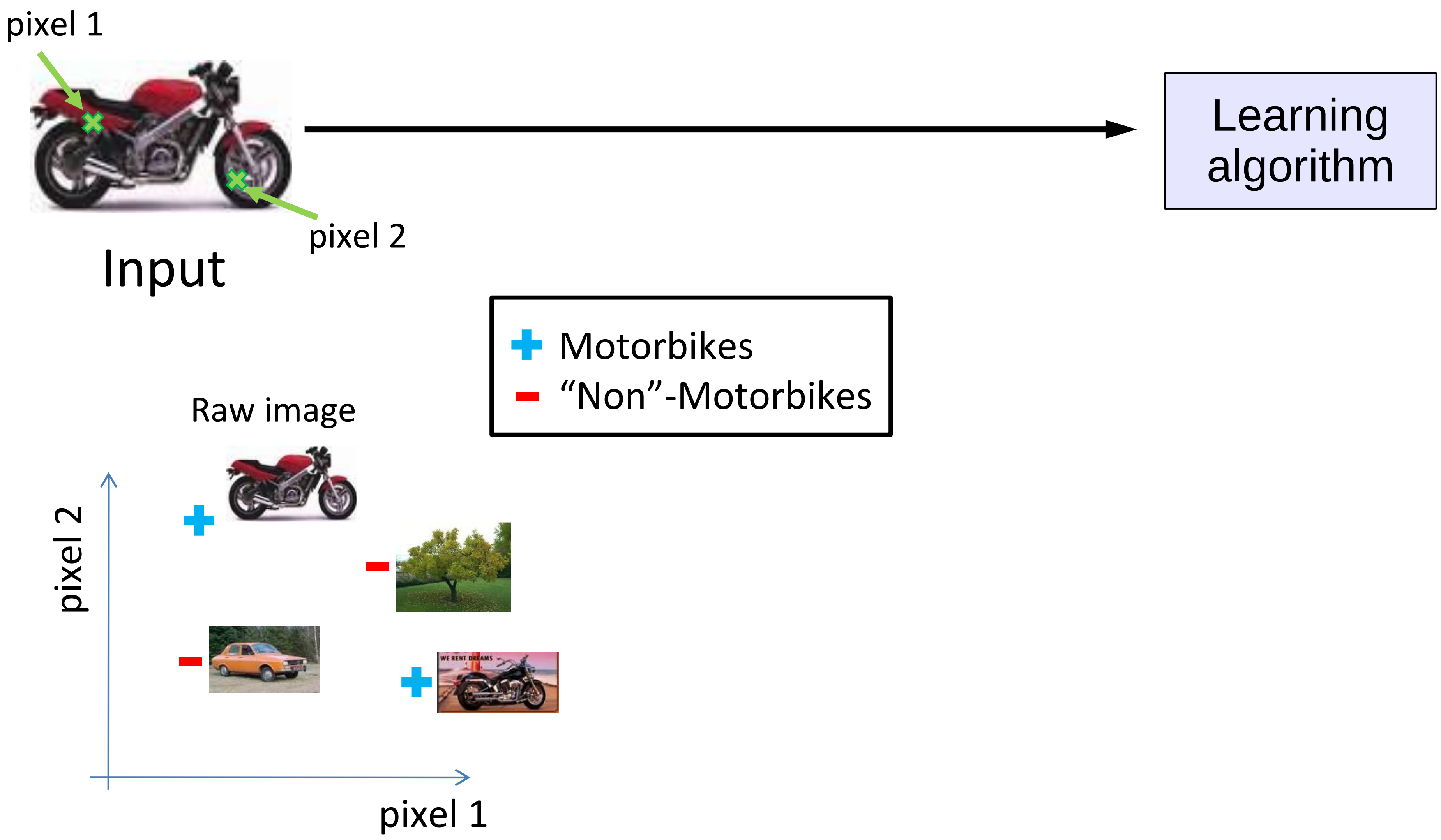
Why is machine learning hard to work in real applications?

You see this:

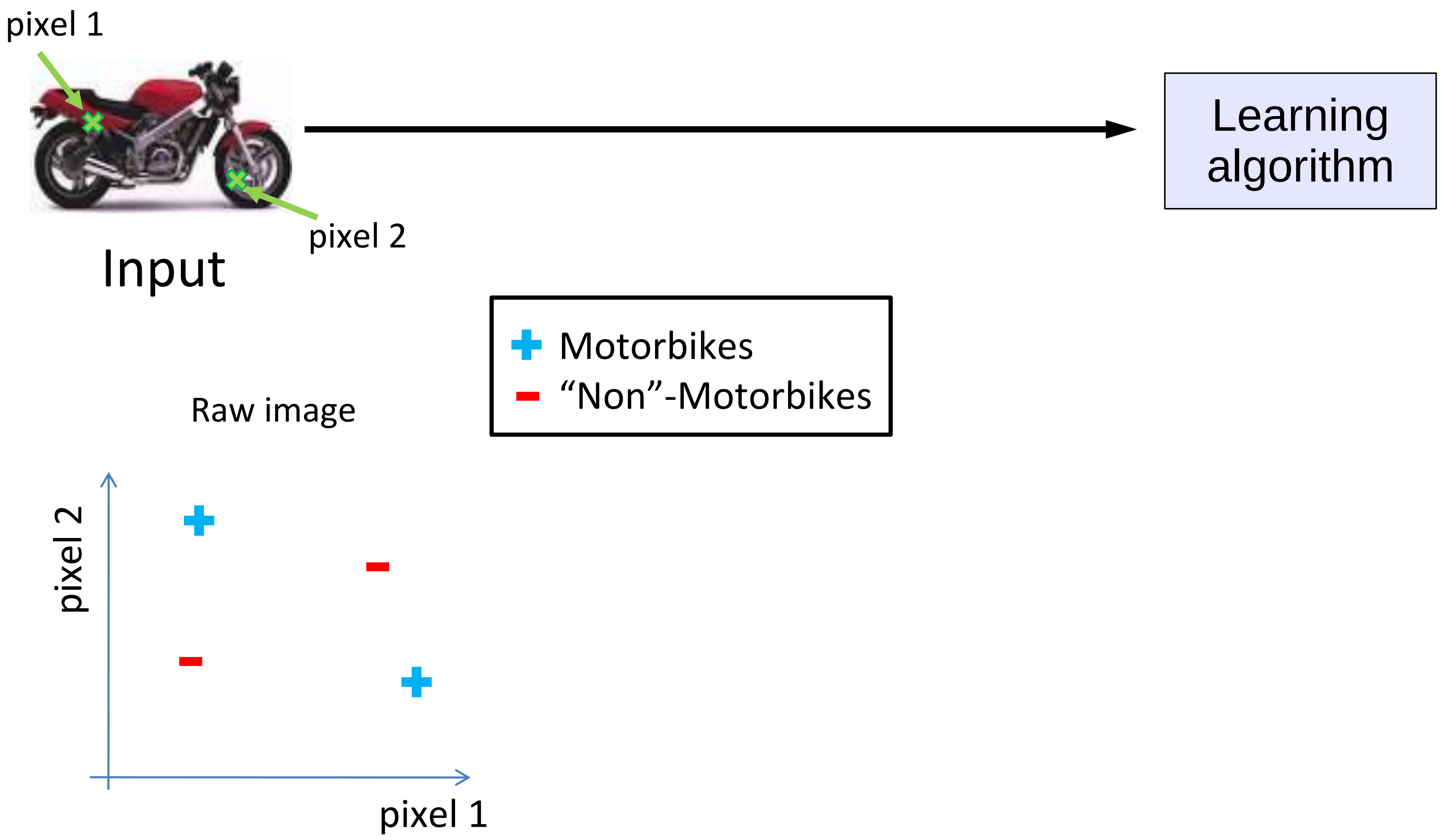
But the camera sees this:

194	210	201	212	199	213	215	195	178	158	182	209
180	189	190	221	209	205	191	167	147	115	129	163
114	126	140	188	176	165	152	140	170	106	78	88
87	103	115	154	143	142	149	153	173	101	57	57
102	112	106	131	122	138	152	147	128	84	58	66
94	95	79	104	105	124	129	113	107	87	69	67
68	71	69	98	89	92	98	95	89	88	76	67
41	56	68	99	63	45	60	82	58	76	75	65
20	43	69	75	56	41	51	73	55	70	63	44
50	50	57	69	75	75	73	74	53	68	59	37
72	59	53	66	84	92	84	74	57	72	63	42
67	61	58	65	75	78	76	73	59	75	69	50

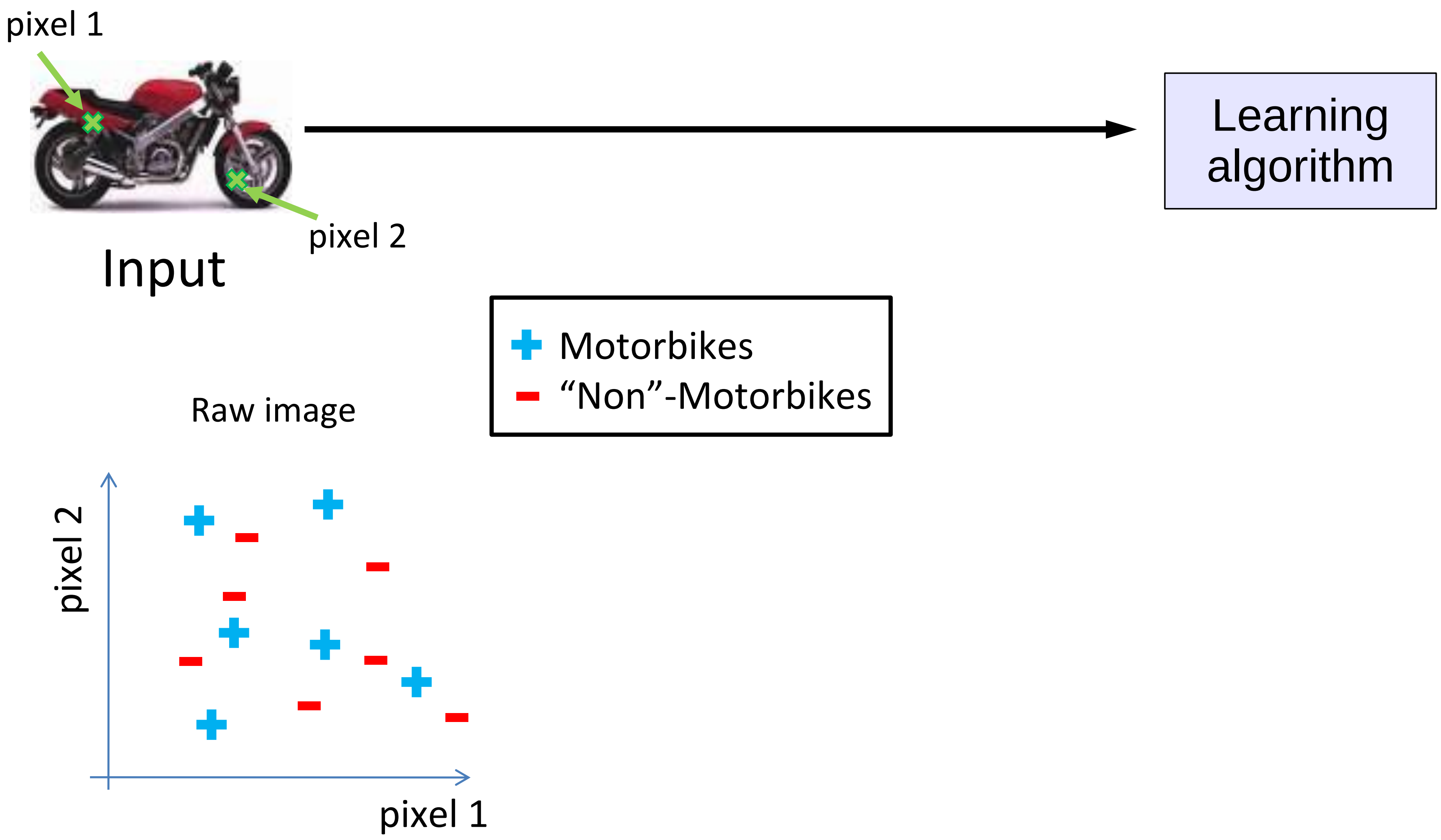
Raw representation



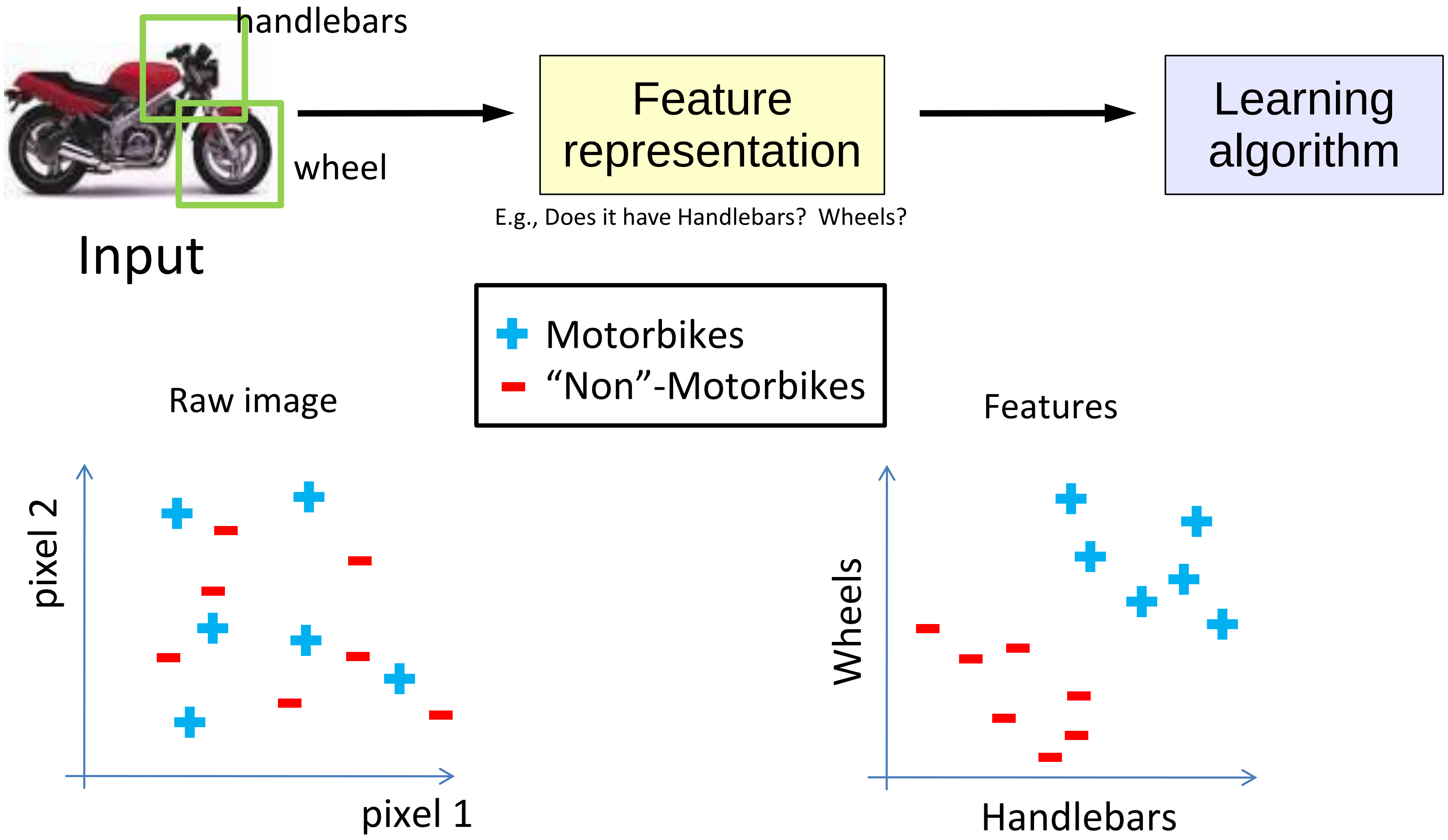
Raw representation



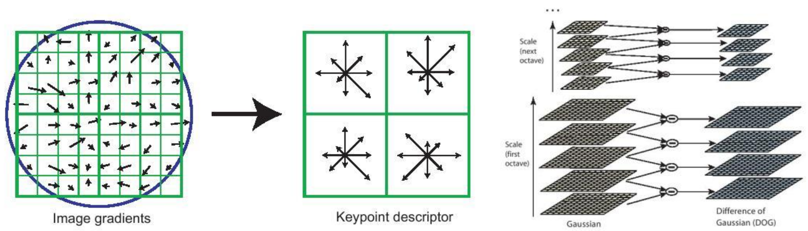
Raw representation



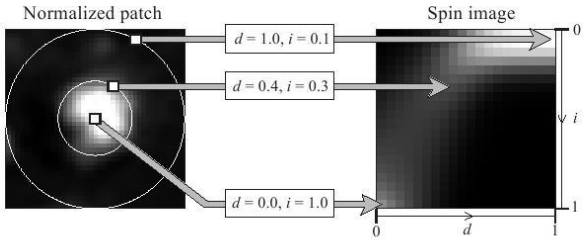
What we want


handlebars
wheel
Input
Feature representation
E.g., Does it have Handlebars? Wheels?
Learning algorithm
Motorbikes
"Non"-Motorbikes
Raw image
pixel 2
pixel 1
Features
Wheels
Handlebars

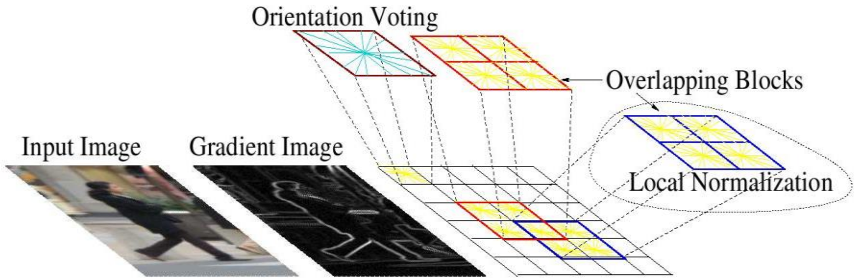
Some feature representations



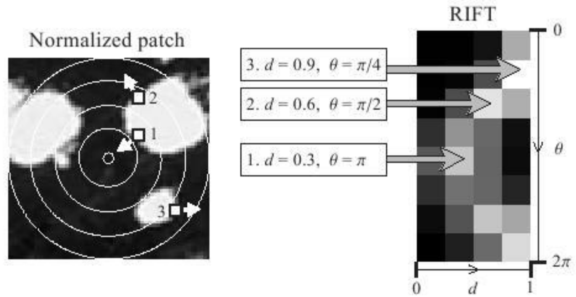
SIFT



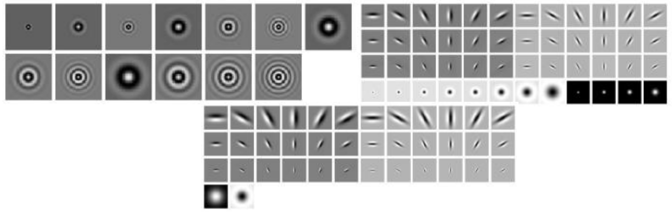
Spin image



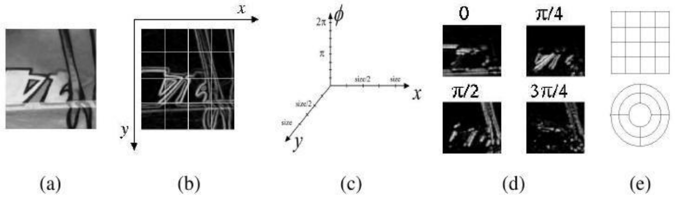
HoG



RIFT

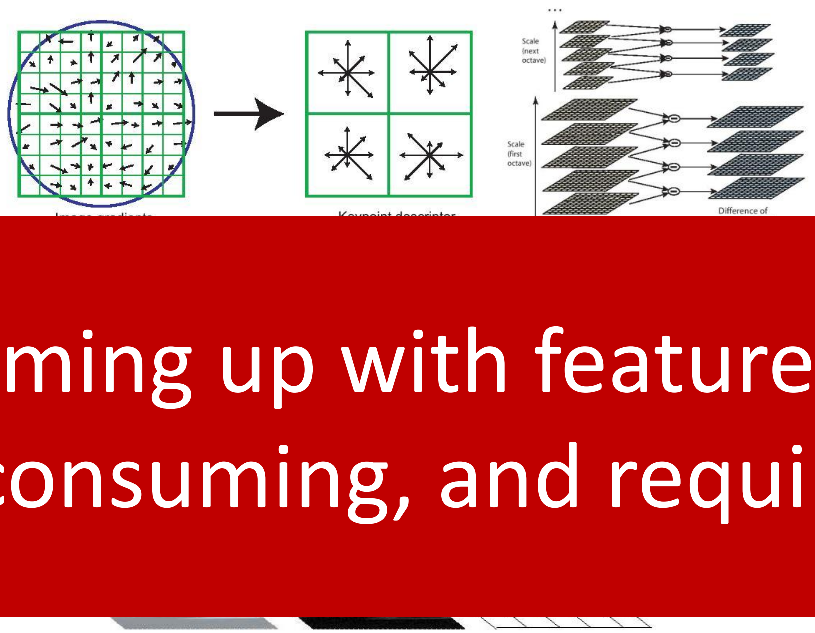


Textons



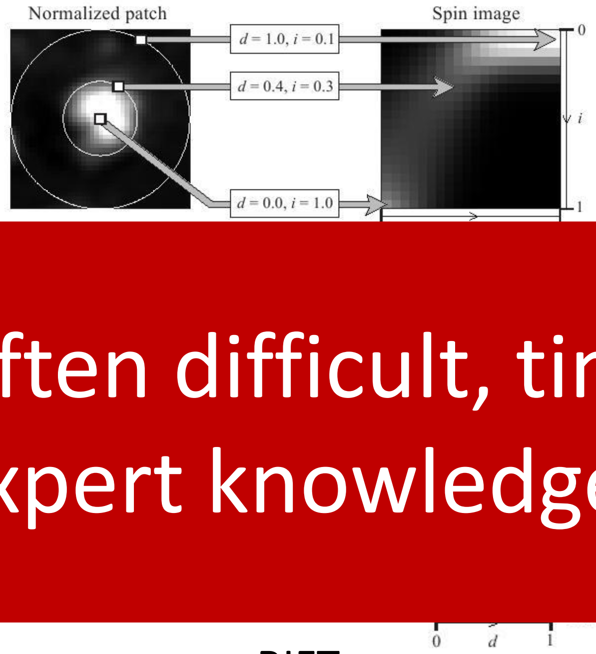
GLOH

Some feature representations

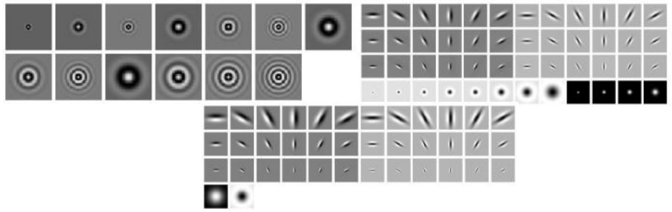


Coming up with features is often difficult, time-consuming, and requires expert knowledge.

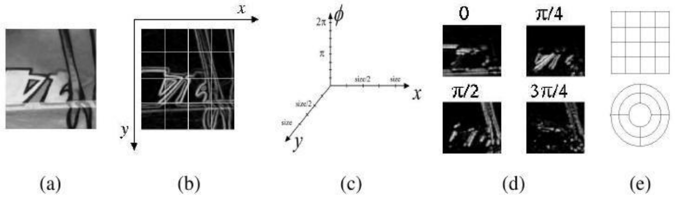
HoG



RIFT

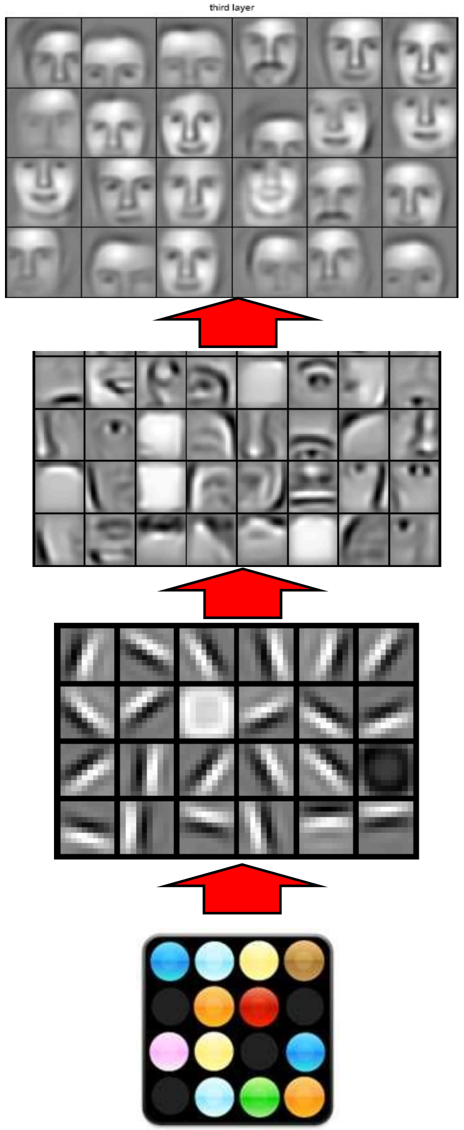


Textons



GLOH

Deep Learning: Let’s learn the representation!

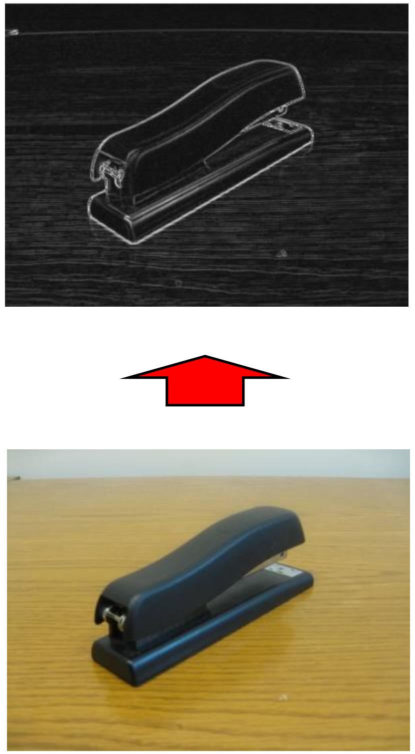


object models

object parts
(combination
of edges)

edges

pixels



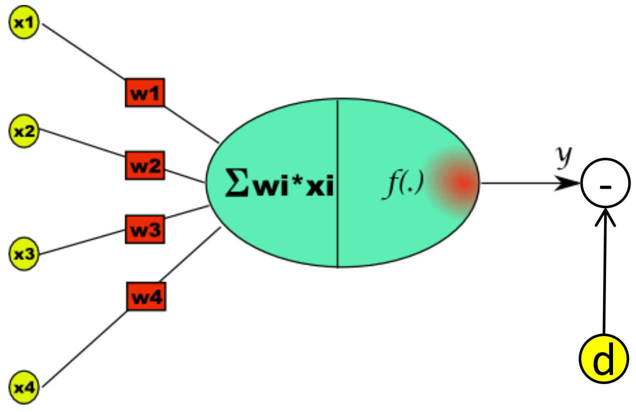
Historical Remarks

The high and low tides of neural networks

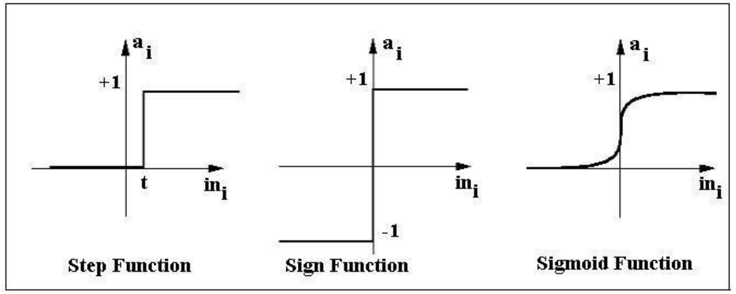
1950s – 1960s The Perceptron

- **The Perceptron was introduced in 1957 by Frank Rosenblatt.**

Perceptron:



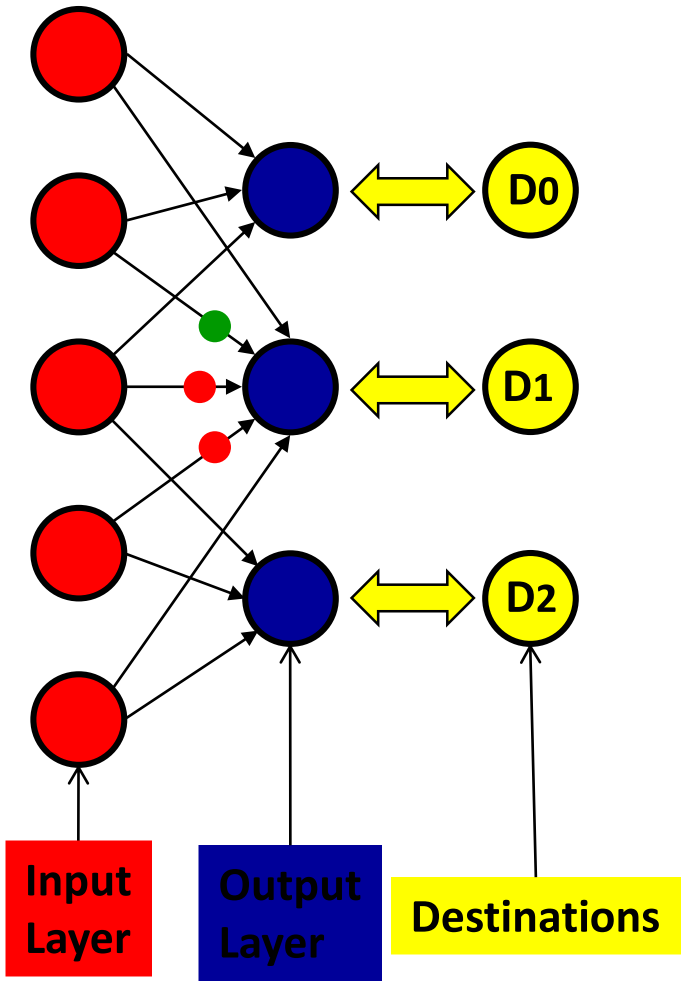
Activation functions:



Learning:

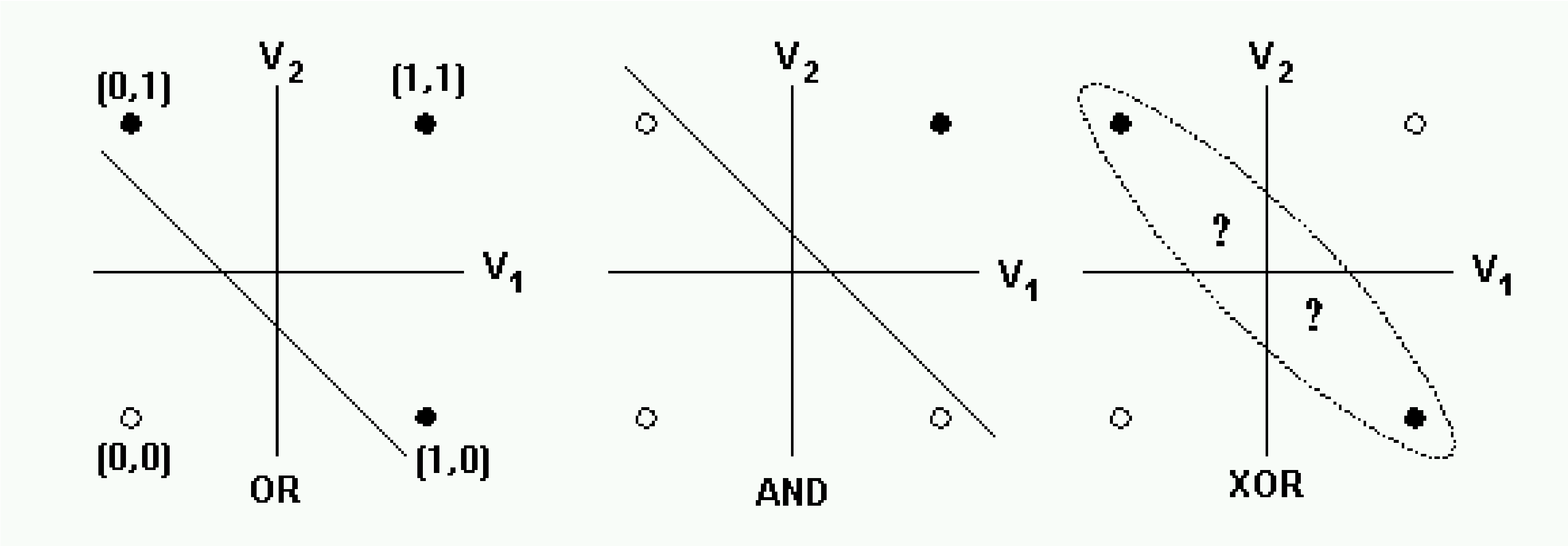
$$y^{(t)} = f\left\{\sum_i w_i^{(t)} x_i^{(t)}\right\}$$

$$\text{Update}\begin{cases} \Delta w_i^{(t)} = \varepsilon(d^{(t)} - y^{(t)})x_i^{(t)} \\ w_i^{(t+1)} = w_i^{(t)} + \Delta w_i^{(t)} \end{cases}$$



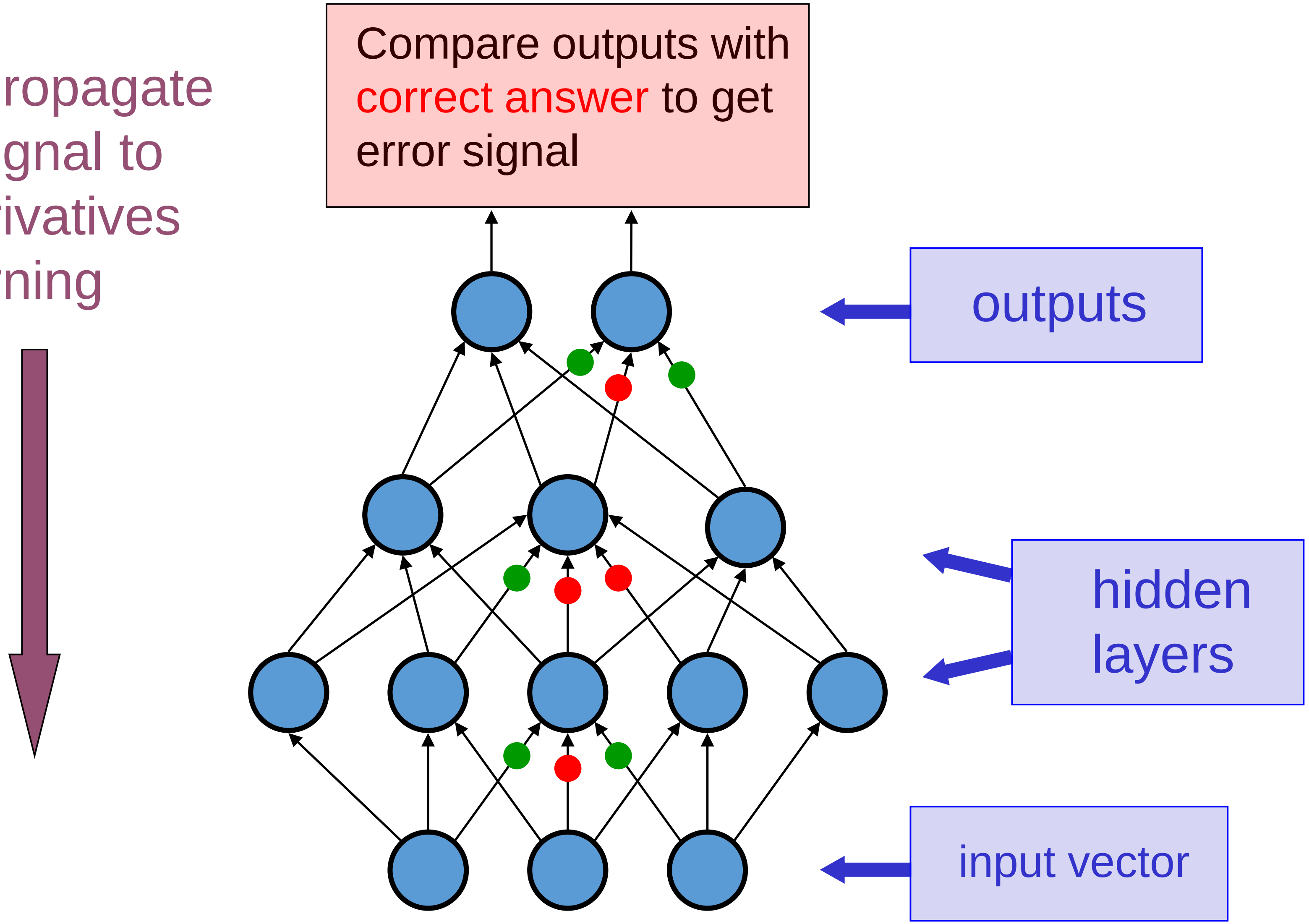
1970s -- Hiatus

- Perceptrons. Minsky and Papert. 1969
  - Revealed the fundamental difficulty in linear perceptron models
  - Stopped research on this topic for more than 10 years



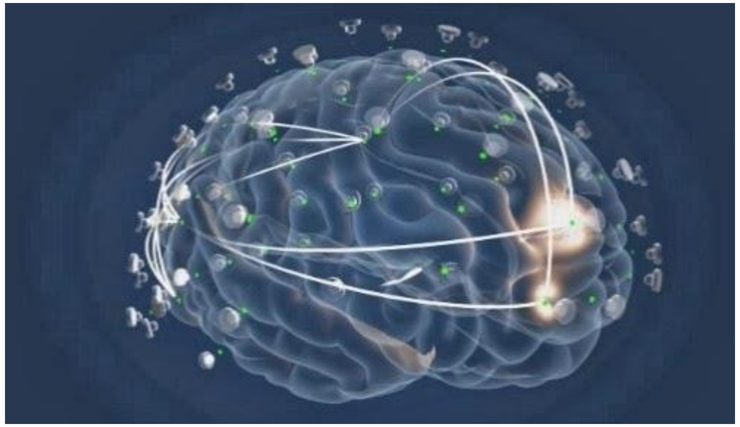
1980s, nonlinear neural networks (Werbos 1974, Rumelhart, Hinton, Williams 1986)

Back-propagate error signal to get derivatives for learning



1990s: Universal approximators

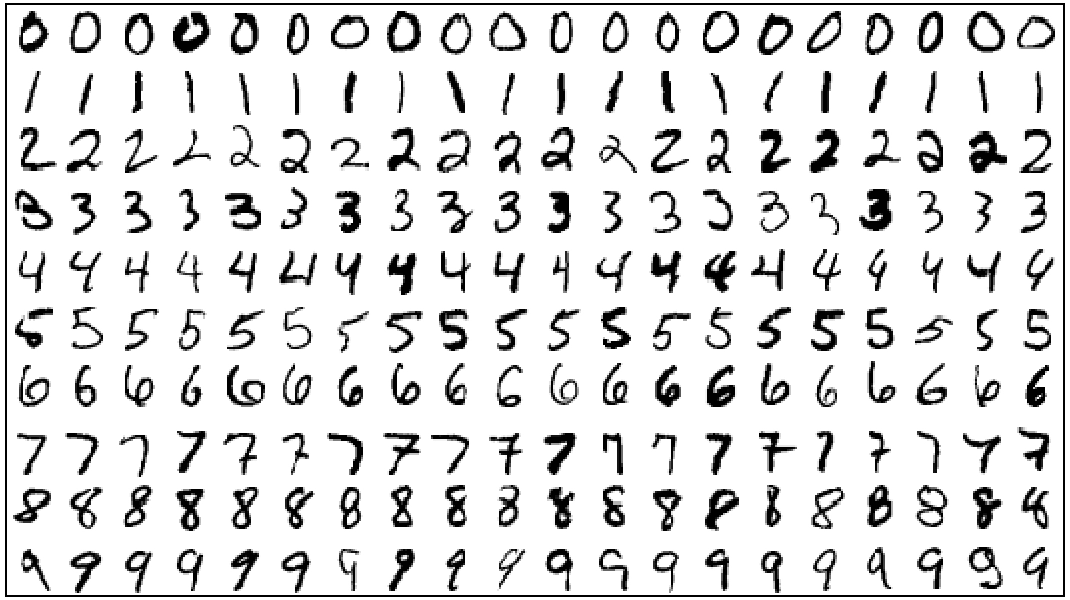
- Glorious times for neural networks (1986-1999):
  - Success in handwritten digits
  - Boltzmann machines
  - Network of all sorts
  - Complex mathematical techniques



- Kernel methods (1992 – 2010):
  - (Cortes, Vapnik 1995), (Vapnik 1995), (Vapnik 1998)
  - Fixed basis function
  - First paper is forced to publish under “Support Vector Networks”

Recognizing Handwritten Digits

- MNIST database
  - 60,000 training, 10,000 testing
  - Large enough for digits
  - Battlefield of the 90s



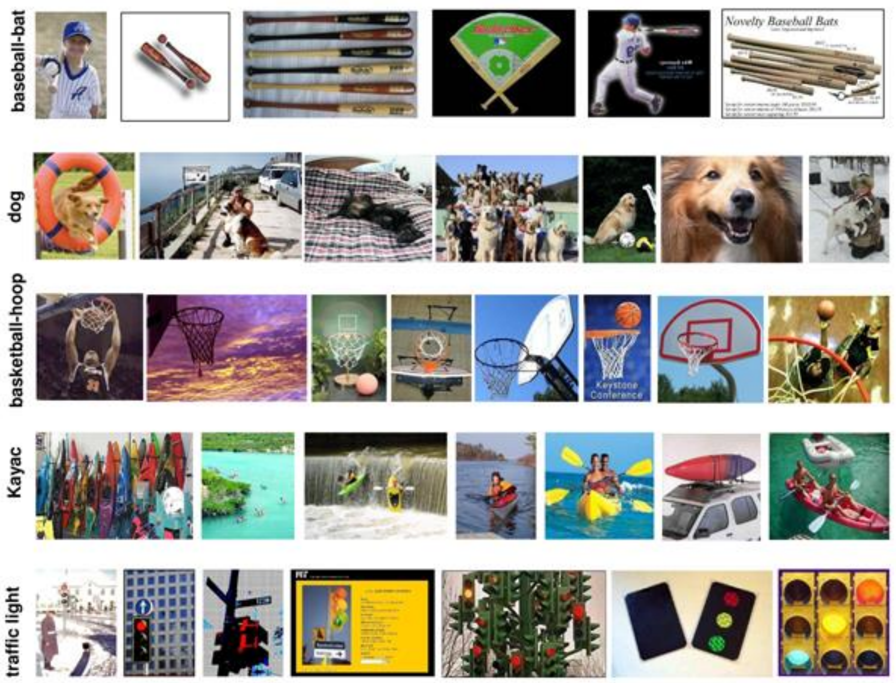
Algorithm	Error Rate (%)
Linear classifier (perceptron)	12.0
K-nearest-neighbors	5.0
Boosting	1.26
SVM	1.4
Neural Network	1.6
Convolutional Neural Networks	0.95
With automatic distortions + ensemble + many tricks	0.23

What’s wrong with backpropagation?

- It requires a lot of labeled training data
- The learning time does not scale well
- It is theoretically **the same** as kernel methods
  - Both are “universal approximators”
- It can get stuck in poor local optima
  - Kernel methods give **globally optimal** solution
- It overfits, especially with many hidden layers
  - Kernel methods have proven approaches to control overfitting

Caltech-101: Long-time computer vision struggles without enough data

- Caltech-101 dataset
  - Around 10,000 images
  - Certainly not enough!



~80% is widely considered to be the limit on this dataset

Algorithm	Accuracy (%)
SVM with Pyramid Matching Kernel (2005)	58.2%
Spatial Pyramid Matching (2006)	64.6%
SVM-KNN (2006)	66.2%
Sparse Coding + Pyramid Matching (2009)	73.2%
SVM Regression w object proposals (2010)	81.9%
Group-Sensitive MKL (2009)	84.3%
Deep Learning (pretrained on Imagenet) (2014)	91.4%

2010s: Deep representation learning

- Comeback: Make it deep!
  - Learn **many**, **many** layers simultaenously
  - How does this happen?
  - Max-pooling (Weng, Ahuja, Huang 1992)
  - Stochastic gradient descent (Hinton 2002)
  - ReLU nonlinearity (Nair and Hinton 2010), (Krizhevsky, Sutskever, Hinton 2012)
    - Better understanding of subgradients
  - Dropout (Hinton et al. 2012)
  - WAY more labeled data
    - Amazon Mechanical Turk (https://www.mturk.com/mturk/welcome)
    - 1 million+ labeled data
  - A lot better computing power
    - GPU processing

Convolutions: Utilize Spatial Locality

Sobel filter

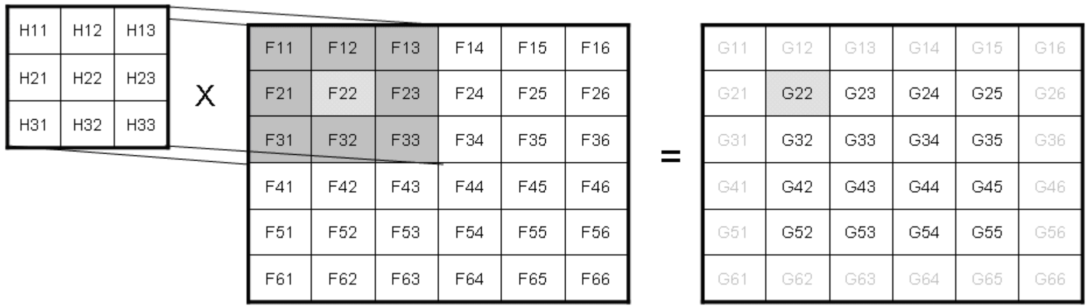
-1	0	+1
-2	0	+2
-1	0	+1

Gx

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

Gy

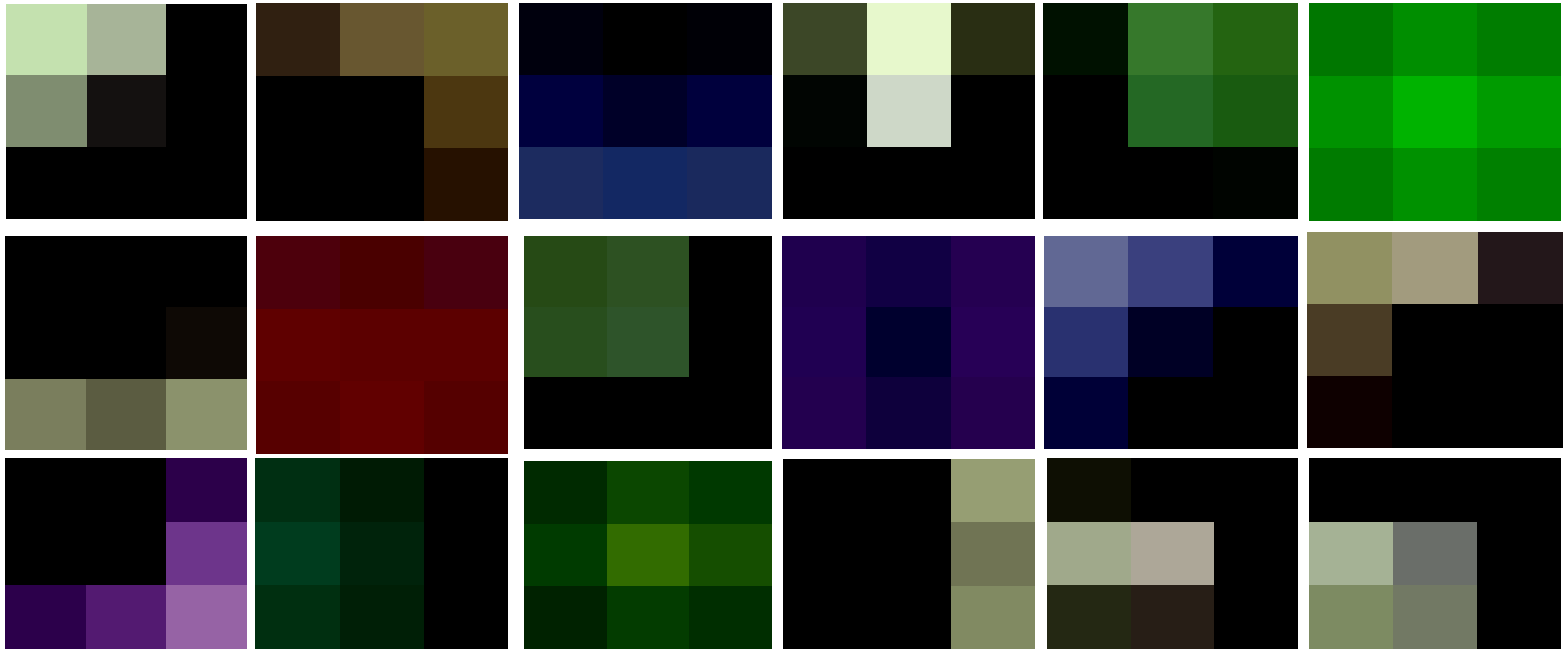
Convolution



Convolution

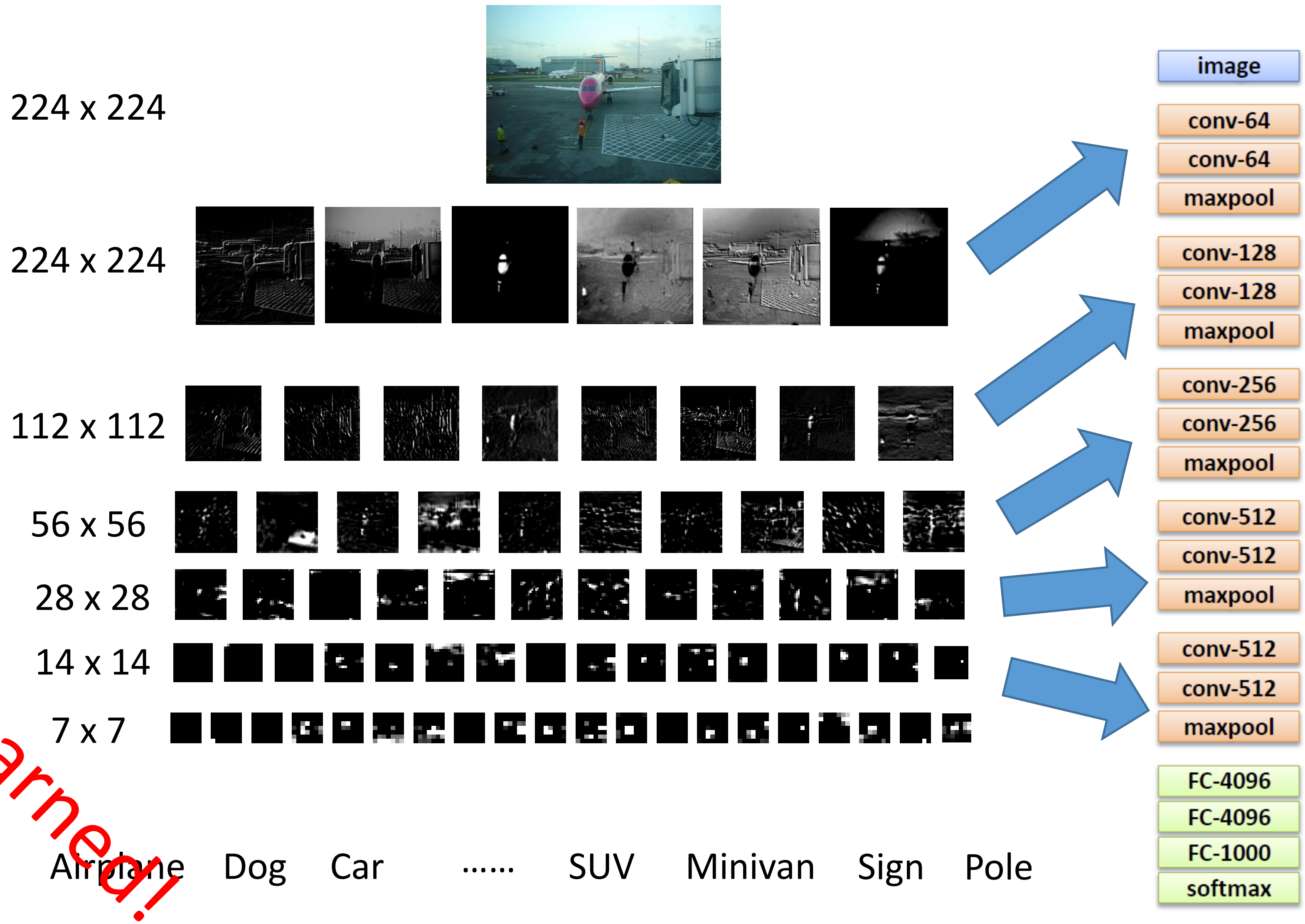
Convolutional Neural Networks

Learning filters:



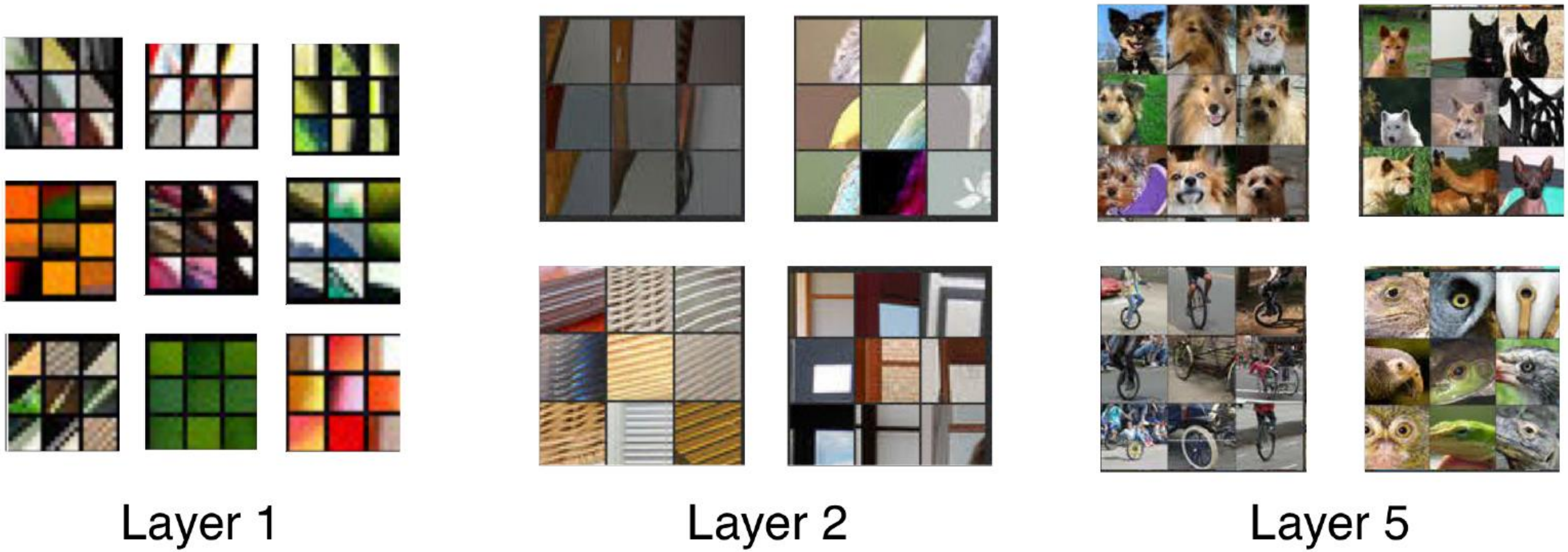
- CNN makes sense because ***locality*** is important for visual processing

A Convolutional Neural Network Model



Every filter is learned!

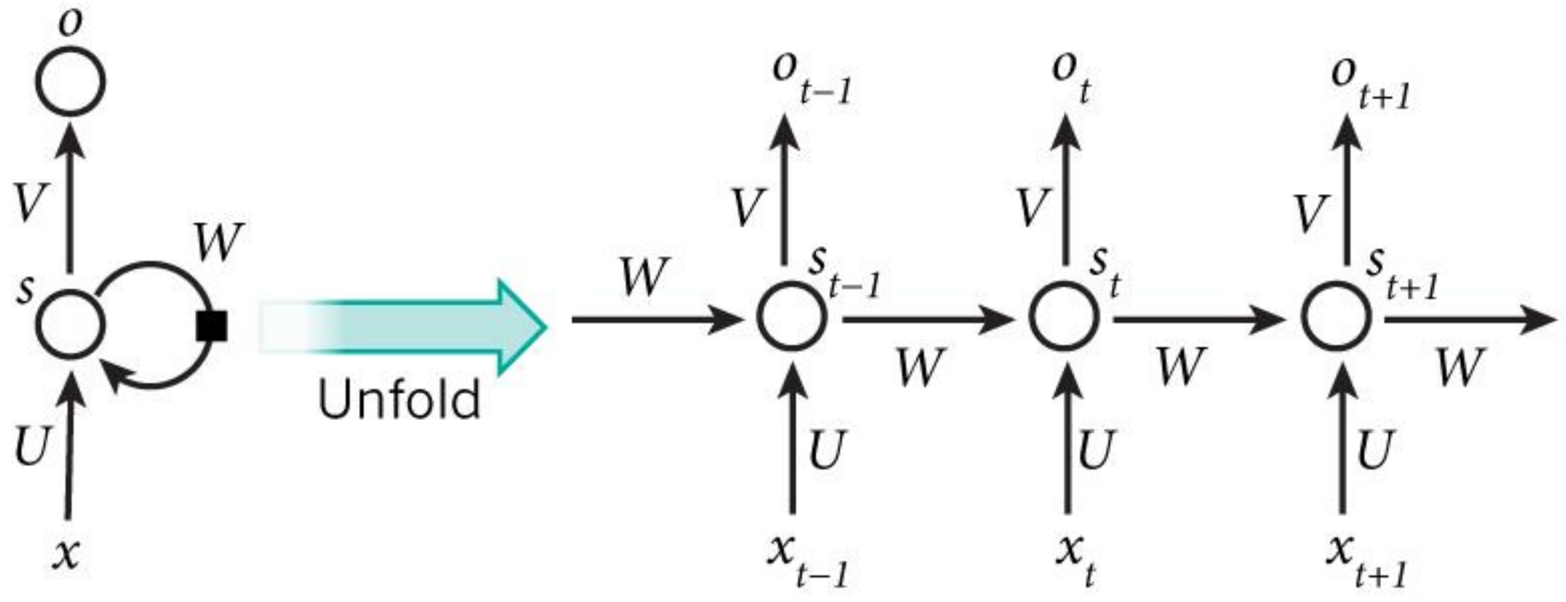
Images that respond to various filters



Zeiler and Fergus 2014

Recurrent Neural Network

- Temporal stability: *history always repeats itself*
  - Parameter sharing across time



What is the hidden assumption in your problem?

- Image Understanding: Spatial locality
- Temporal Models: Temporal (partial) stationarity
- How about your problem?

References

- (Weng, Ahuja, Huang 1992) J. Weng, N. Ahuja and T. S. Huang, "Cresceptron: a self-organizing neural network which grows adaptively," *Proc. International Joint Conference on Neural Networks*, Baltimore, Maryland, vol I, pp. 576-581, June, 1992.
- (Hinton 2002) Hinton, G. E..Training Products of Experts by Minimizing Contrastive Divergence. Neural Computation, 14, pp 1771-1800.
- (Hinton, Osindero and Teh 2006) Hinton, G. E., Osindero, S. and Teh, Y.. A fast learning algorithm for deep belief nets. Neural Computation 18, pp 1527-1554.
- (Cortes and Vapnik 1995) Support-vector networks. C Cortes, V Vapnik. Machine learning 20 (3), 273-297
- (Vapnik 1995) V Vapnik. The Nature of Statistical Learning Theory. Springer 1995
- (Vapnik 1998) V Vapnik. Statistical Learning Theory. Wiley 1998.
- (Krizhevsky, Sutskever, Hinton 2012). ImageNet Classification with Deep Convolutional Neural Networks. NIPS 2012
- (Nair and Hinton 2010) V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In Proc. 27th International Conference on Machine Learning, 2010
- (Hinton et al. 2012) G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. Arxiv 2012.
- (Zeiler and Fergus 2014) ***M.D. Zeiler, R. Fergus***. Visualizing and Understanding Convolutional Networks. ECCV 2014