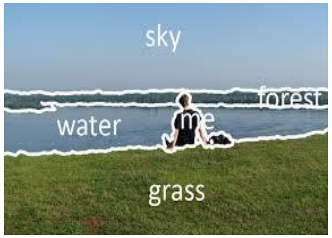
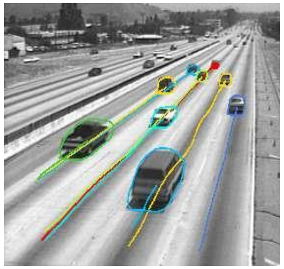
COMPUTER VISION

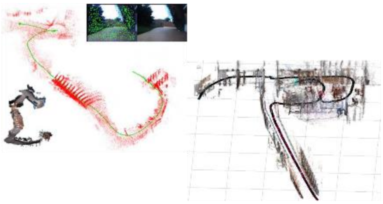
<table>
<tr><td colspan="5" rowspan="4"></td><th colspan="6">Tasks</th></tr>
<tr><th colspan="5">ADAS</th><th></th></tr>
<tr><th colspan="3">Self Driving</th><th></th><th></th><th></th></tr>
<tr><th>Localization</th><th>Perception</th><th>Planning/ Control</th><th>Driver state</th><th>Vehicle Diagnosis</th><th>Smart factory</th></tr>
<tr><td rowspan="7">Methods</td><td rowspan="2">Traditional</td><td colspan="3">Non-machine Learning</td><td>GPS, SLAM</td><td></td><td>Optimal control</td><td></td><td></td><td></td></tr>
<tr><td rowspan="6">Machine-Learning based method</td><td rowspan="4">Supervised</td><td>SVM
MLP</td><td></td><td>Pedestrian detection (HOG+SVM)</td><td></td><td></td><td></td><td></td></tr>
<tr><td rowspan="5">Deep-Learning based</td><td>CNN</td><td></td><td>Detection/ Segmentation/Classification</td><td>End-to-end Learning</td><td></td><td></td><td></td></tr>
<tr><td>RNN
(LSTM)</td><td></td><td>Dry/wet road classification</td><td>End-to-end Learning</td><td>Behavior Prediction/ Driver identification</td><td></td><td>*</td></tr>
<tr><td>DNN</td><td></td><td></td><td></td><td></td><td>*</td><td>*</td></tr>
<tr><td colspan="2">Reinforcement</td><td></td><td></td><td>*</td><td></td><td></td><td></td></tr>
<tr><td colspan="2">Unsupervised</td><td></td><td></td><td></td><td></td><td></td><td>*</td></tr>
</table>

Vision tasks

Object recognition	Object detection	Semantic segmentation	Object tracking	Visual SLAM







Semantic segmentation

- Building/road/sky/object/grass/water/tree

Clement Farabet, Camille Couprie, Laurent Najman and Yann LeCun: Learning Hierarchical Features for Scene Labeling, IEEE Transactions on Pattern Analysis and Machine Intelligence, August, 2013

Object tracking

Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang, "Object Tracking Benchmark",
IEEE Transactions on Pattern Analysis and Machine Intelligence, 2015

Visual SLAM

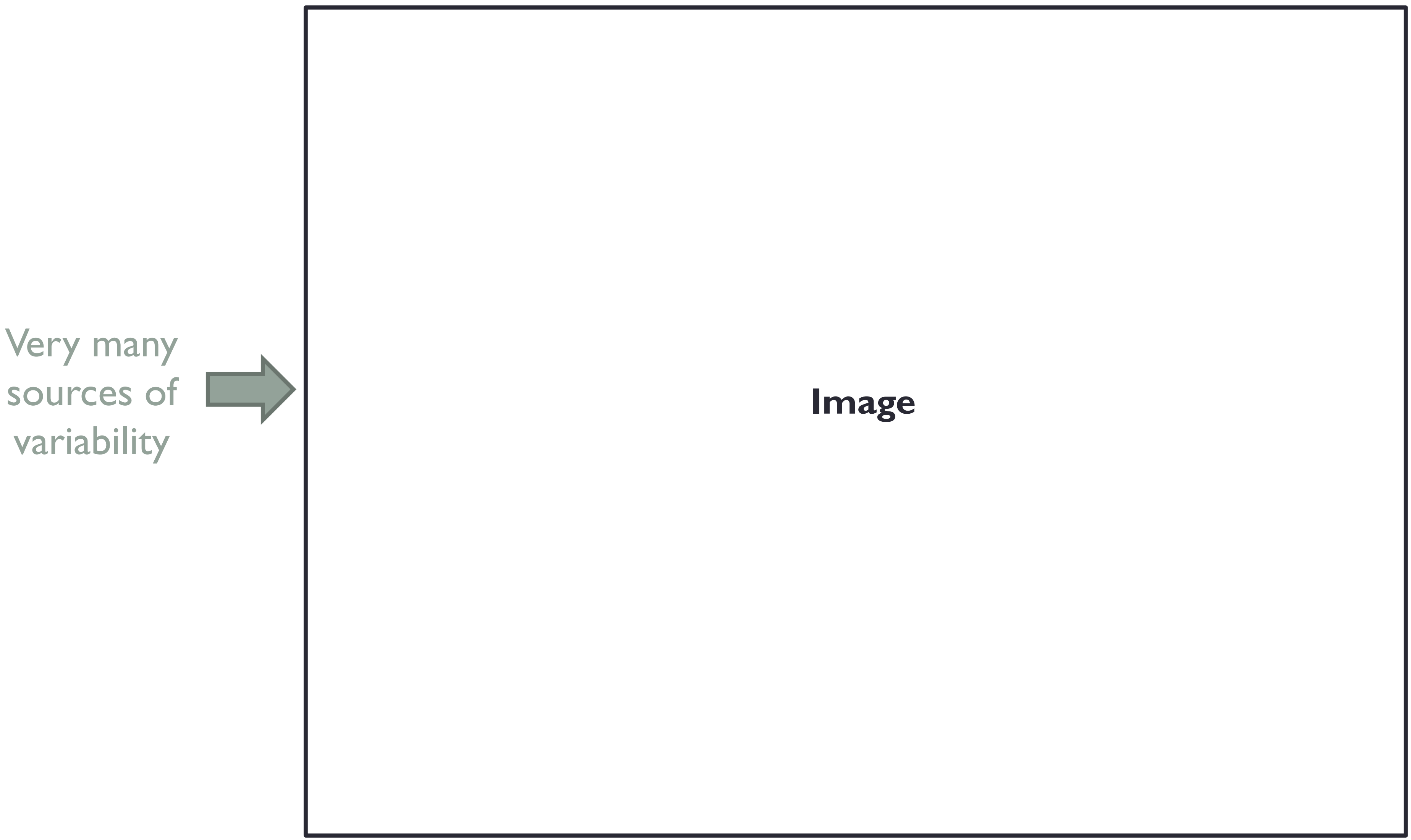
					Tasks					
					ADAS					
					Self Driving					
					Localization	Perception	Planning/ Control	Driver state	Vehicle Diagnosis	Smart factory
Methods	Traditional	Non-machine Learning			GPS, SLAM		Optimal control			
Methods	Traditional	Machine-Learning based method	Supervised	SVM MLP		Pedestrian detection (HOG+SVM)				
Methods	Deep-Learning based	Machine-Learning based method	Supervised	CNN		Detection/ Segmentation/Classification	End-to-end Learning			
Methods	Deep-Learning based	Machine-Learning based method	Supervised	RNN (LSTM)		Dry/wet road classification	End-to-end Learning	Behavior Prediction/ Driver identification		*
Methods	Deep-Learning based	Machine-Learning based method	Supervised	DNN					*	*
Methods	Deep-Learning based	Machine-Learning based method	Reinforcement				*			
Methods	Deep-Learning based	Machine-Learning based method	Unsupervised							*

ORB-SLAM in the KITTI dataset

- ORB-SLAM2 is a real-time SLAM library for **Monocular**, **Stereo** and **RGB-D** cameras that computes the camera trajectory and a sparse 3D reconstruction

COMPUTER VISION
IMAGE UNDERSTANDING …

Why understanding images is hard

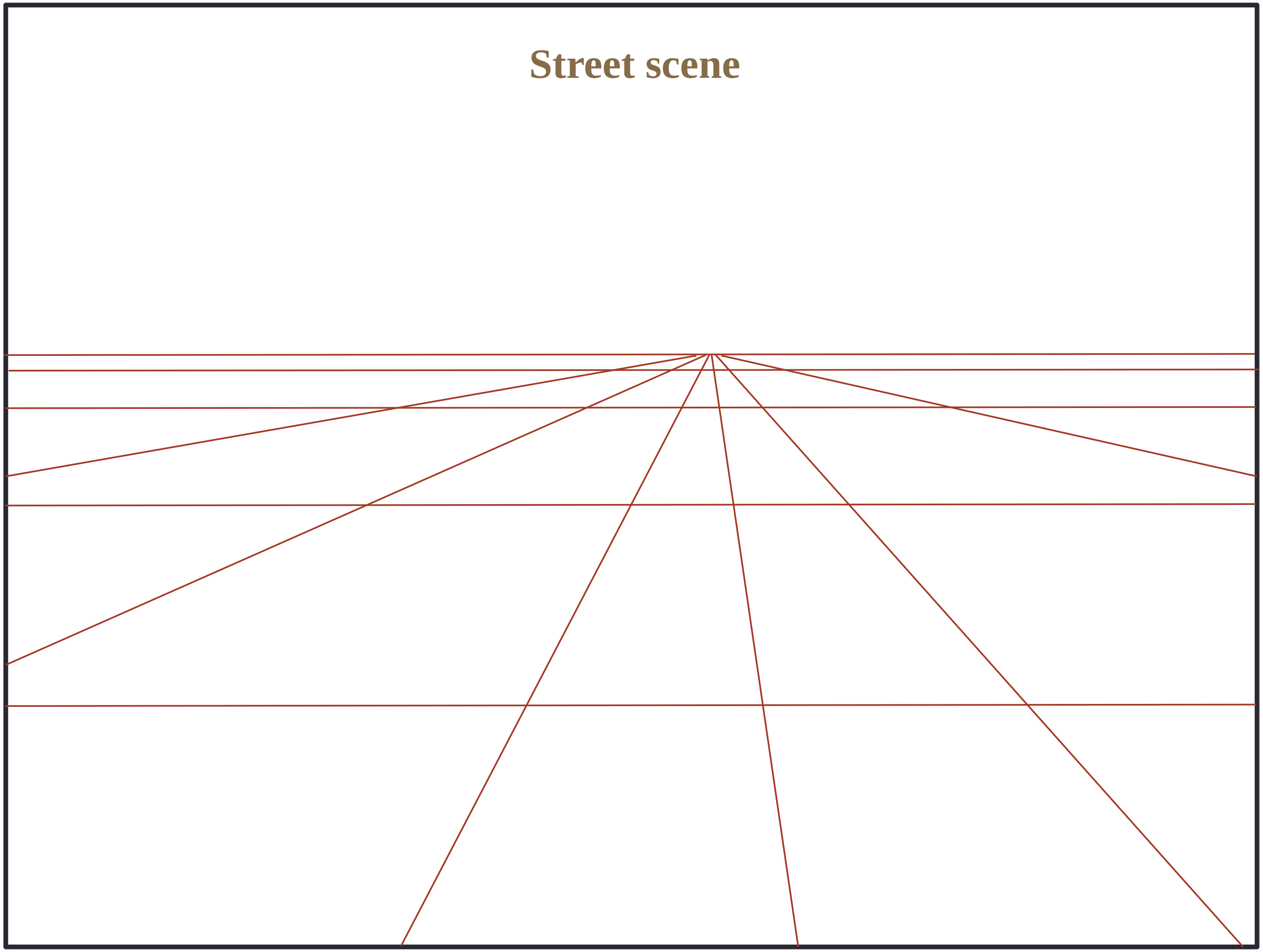


From J. Winn, MSR

Sources of image variability

Scene type

Scene geometry



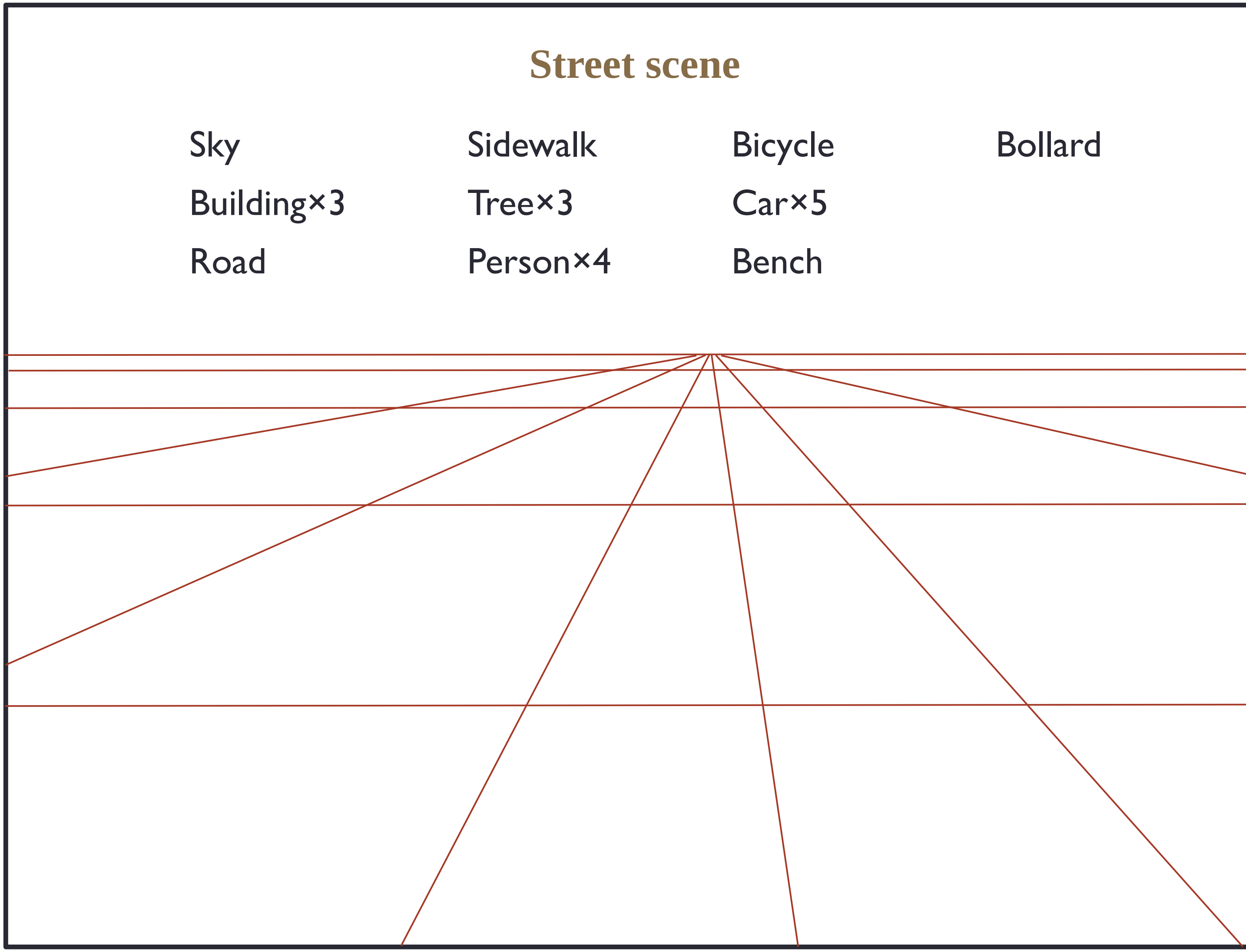
From J. Winn, MSR

Sources of image variability

Scene type

Scene geometry

Object classes



From J. Winn, MSR

Sources of image variability

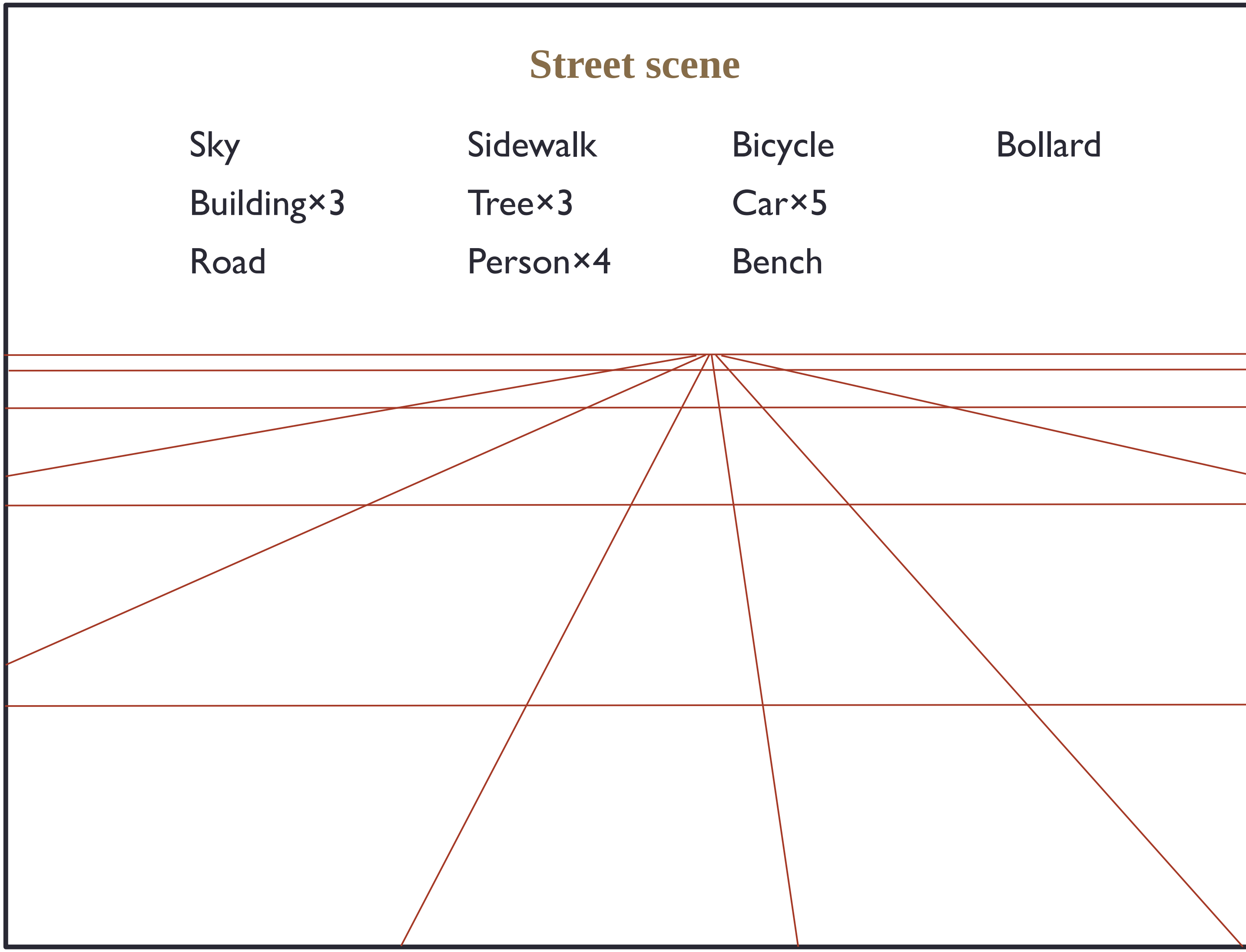
Scene type

Scene geometry

Object classes

Object position

Object orientation



From J. Winn, MSR

Sources of image variability

Scene type

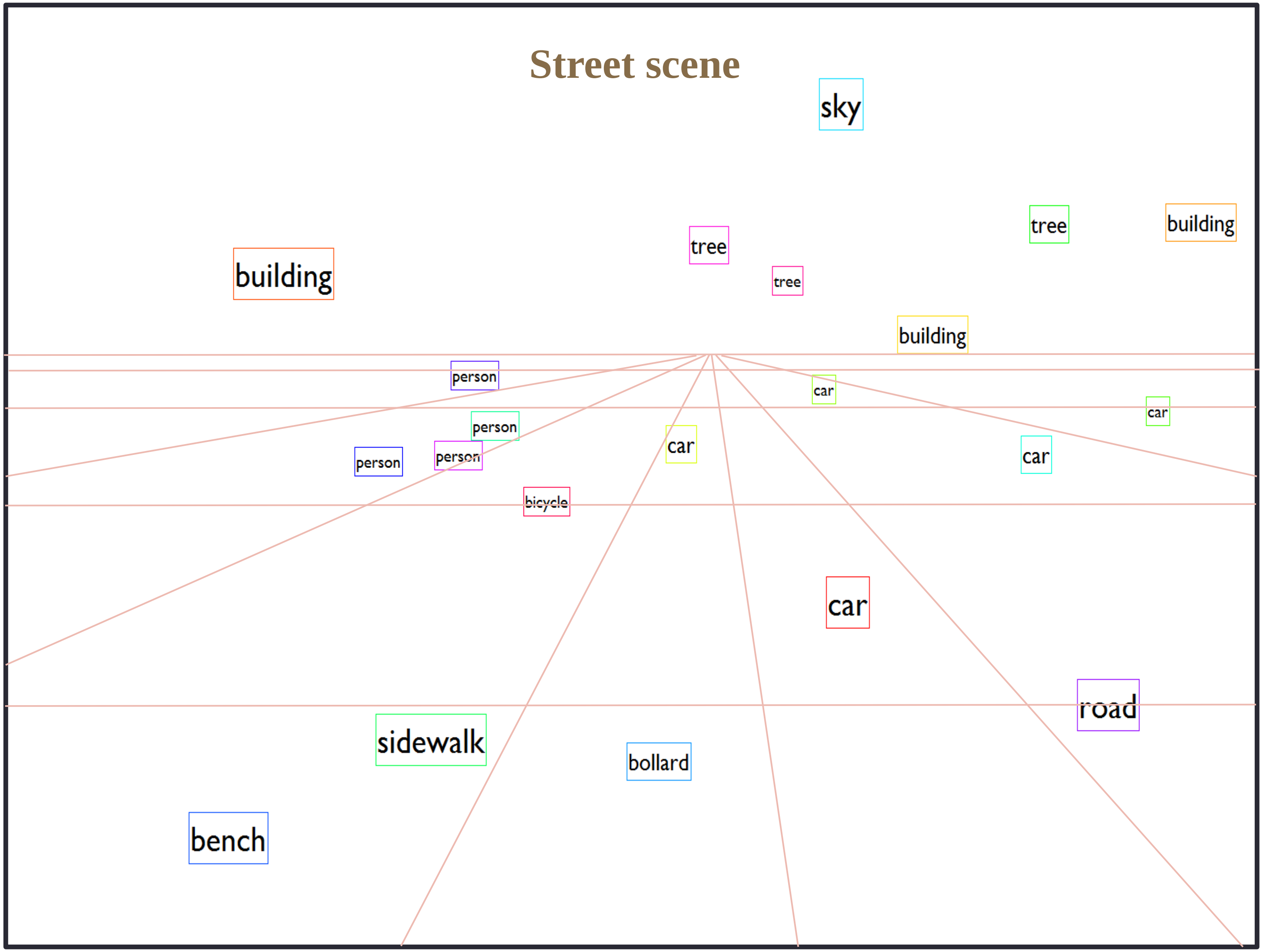
Scene geometry

Object classes

Object position

Object orientation

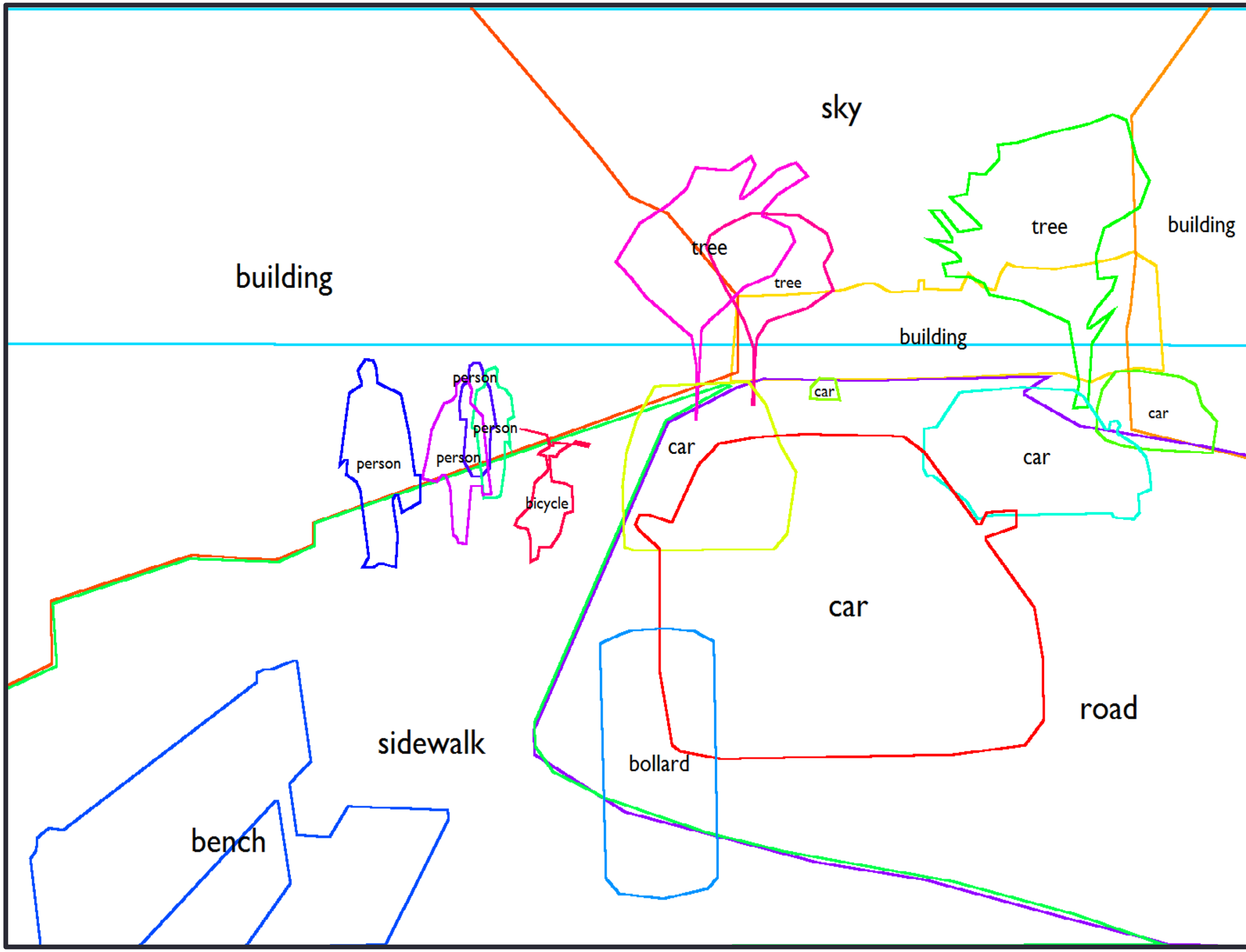
Object shape



From J. Winn, MSR

Sources of image variability

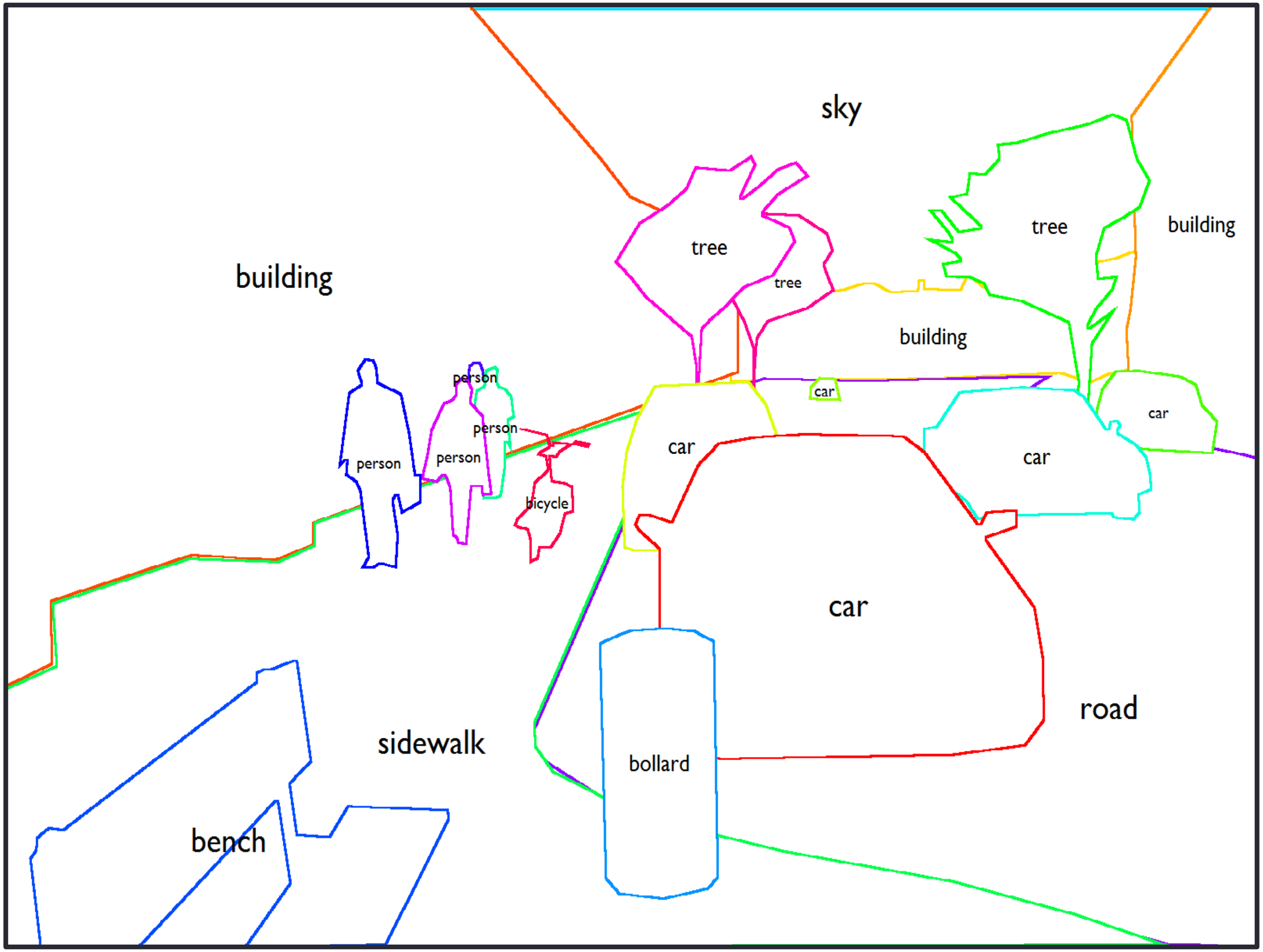
Scene type
Scene geometry
Object classes
Object position
Object orientation
Object shape
Depth/occlusions



From J. Winn, MSR

Sources of image variability

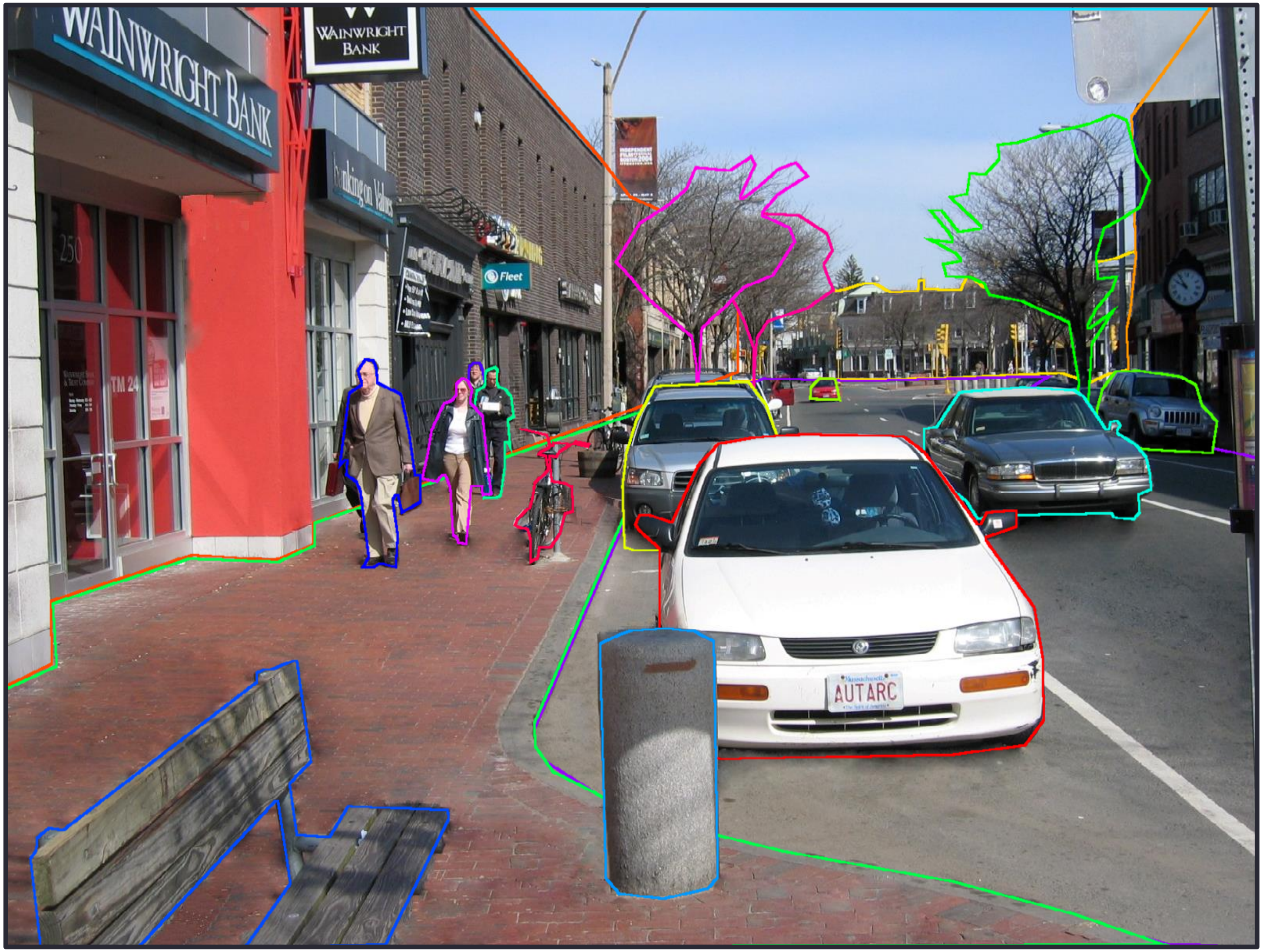
Scene type
Scene geometry
Object classes
Object position
Object orientation
Object shape
Depth/occlusions
Object appearance



From J. Winn, MSR

Sources of image variability

Scene type
Scene geometry
Object classes
Object position
Object orientation
Object shape
Depth/occlusions
Object appearance
Illumination
Shadows



From J. Winn, MSR

Sources of image variability

Scene type
Scene geometry
Object classes
Object position
Object orientation
Object shape
Depth/occlusions
Object appearance
Illumination
Shadows

From J. Winn, MSR

Sources of image variability

Scene type
Scene geometry
Object classes
Object position
Object orientation
Object shape
Depth/occlusions
Object appearance
Illumination
Shadows
Motion blur
Camera effects

From J. Winn, MSR

Computer vision problems

Scene type

Scene geometry

Object classes

Object position

Object orientation

Object shape

Depth/occlusions

Object appearance

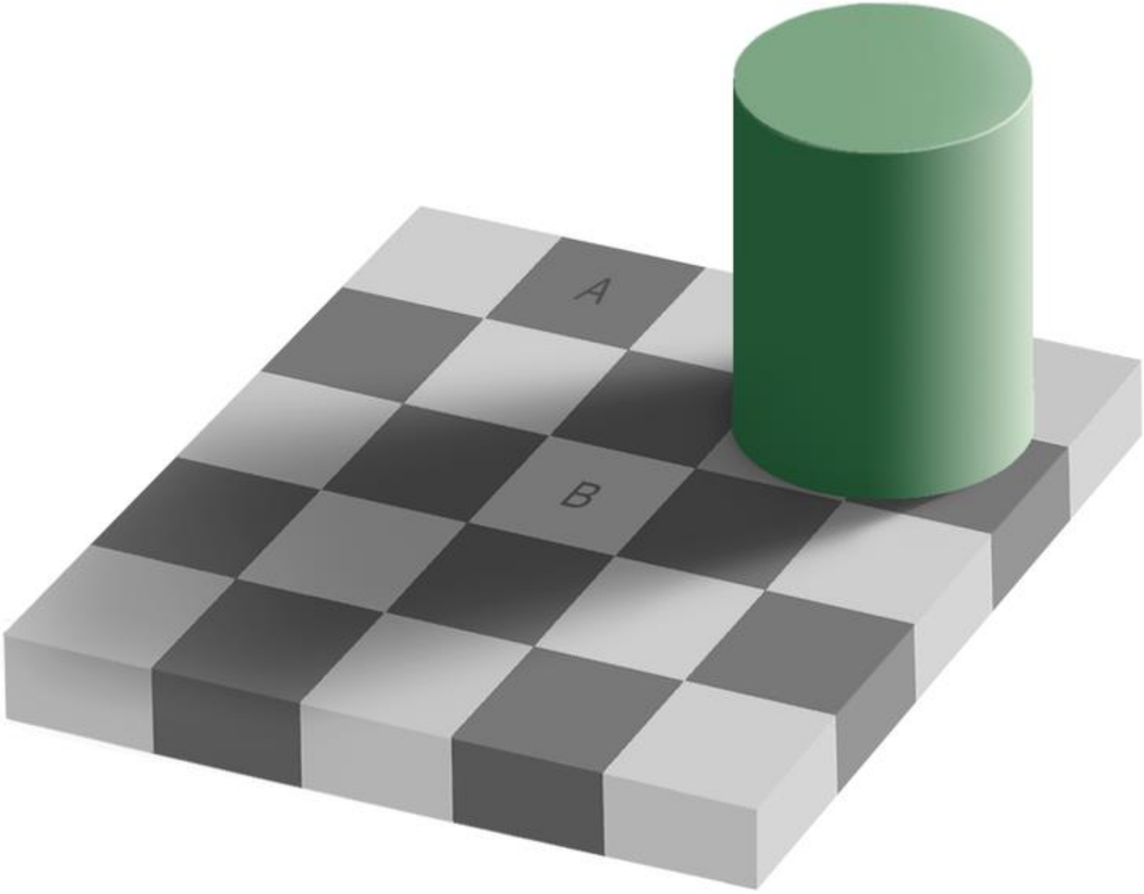
Illumination

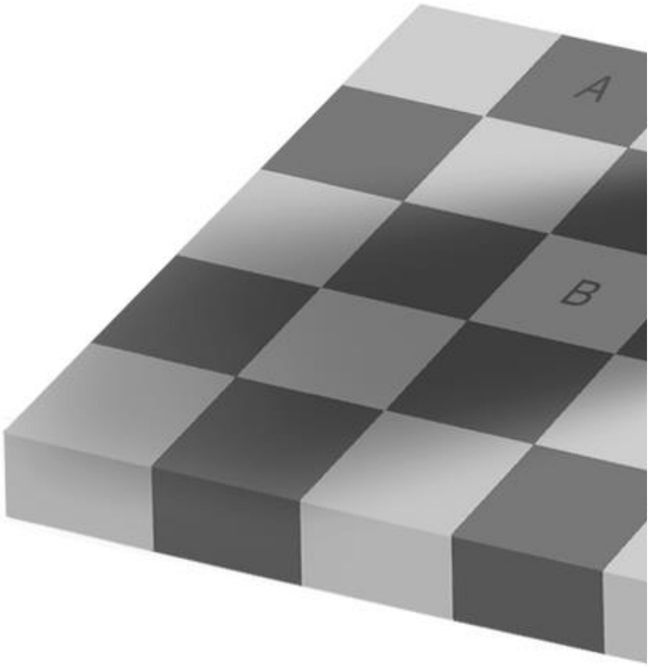
Shadows

Motion blur

Camera effects

Now you see me


A
B


A
B

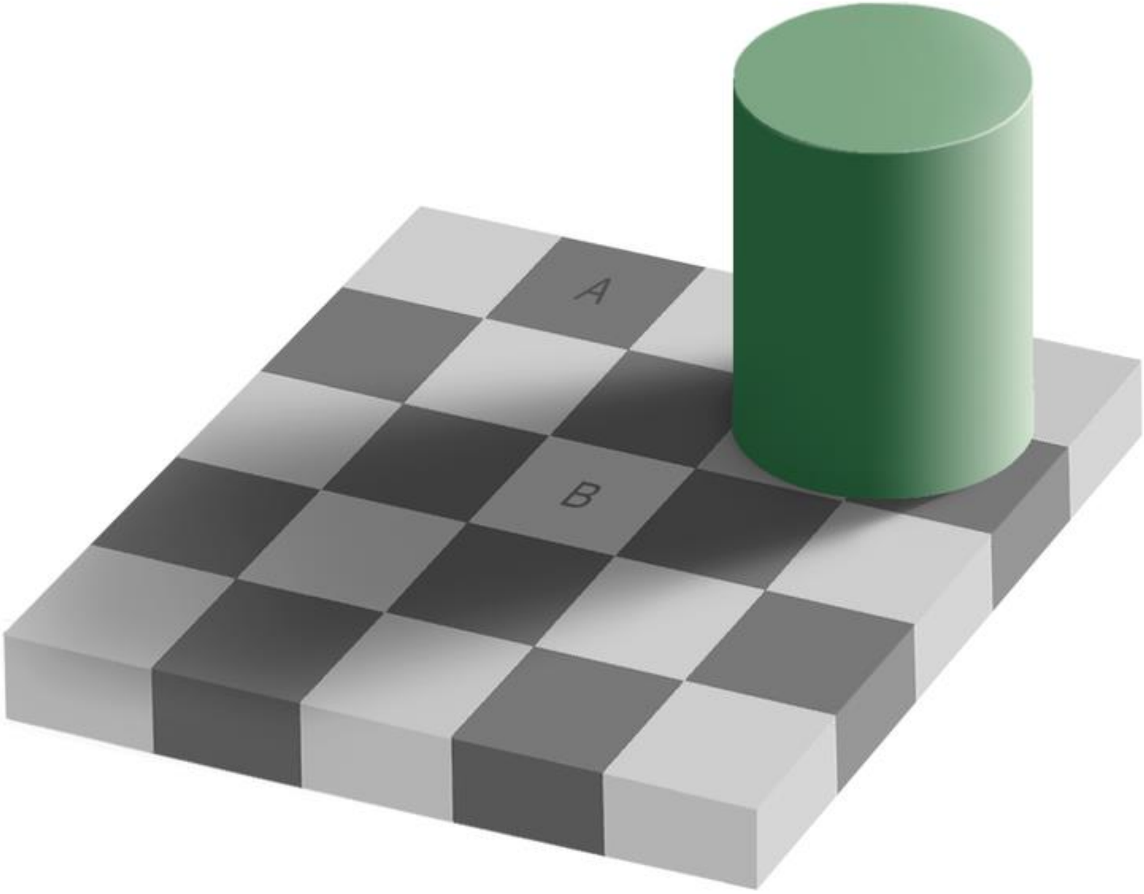

A
B


A
B

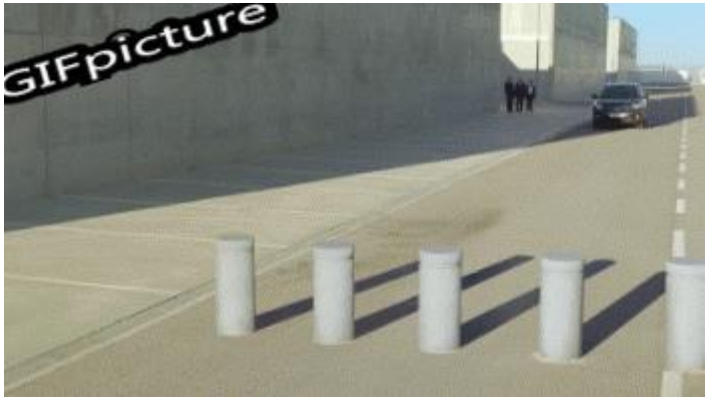

A
B


A


B


A
B


© swiked/Tumblr


GIFpicture

Moravec's Paradox

- The main lesson of 35 years of AI research is that **the hard problems are easy and the easy problems are hard**. The mental abilities of a four-year-old that we take for granted – recognizing a face, lifting a pencil, walking across a room, answering a question – in fact solve some of the hardest engineering problems ever conceived... As the new generation of intelligent devices appears, it will be the stock analysts and petrochemical engineers and parole board members who are in danger of being replaced by machines. The gardeners, receptionists, and cooks are secure in their jobs for decades to come.
  - Pinker, Steven (September 4, 2007) [1994], The Language Instinct, Perennial Modern Classics, Harper, ISBN 0-06-133646-7

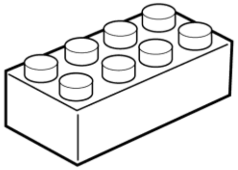
MACHINE LEARNING

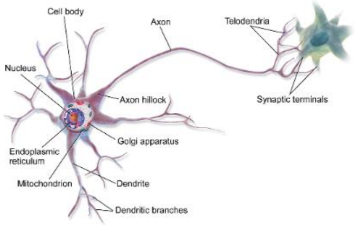
Neural network을 중심으로

					Tasks					
					ADAS					
					Self Driving					
					Localization	Perception	Planning/ Control	Driver state	Vehicle Diagnosis	Smart factory
Methods	Traditional	Non-machine Learning			GPS, SLAM		Optimal control			
Methods	Traditional	Machine-Learning based method	Supervised	MLP		Pedestrian detection (HOG+SVM)				
Methods	Deep-Learning based	Machine-Learning based method	Supervised	CNN		Detection/ Segmentation/Classification	End-to-end Learning			
Methods	Deep-Learning based	Machine-Learning based method	Supervised	RNN (LSTM)		Dry/wet road classification	End-to-end Learning			
Methods	Deep-Learning based	Machine-Learning based method	Supervised	DNN						
Methods	Deep-Learning based	Machine-Learning based method	Reinforcement							
Methods	Deep-Learning based	Machine-Learning based method	Unsupervised							

LINEAR PERCEPTRON

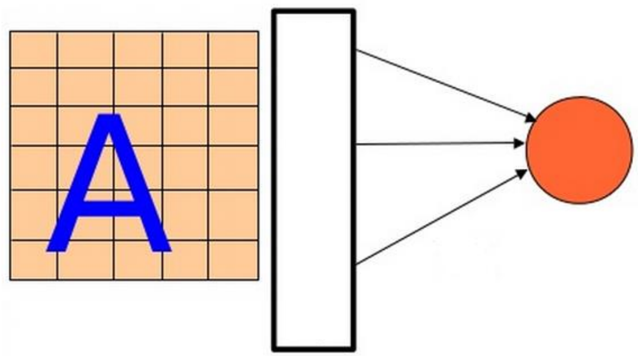
뉴런: 신경망의 기본 단위



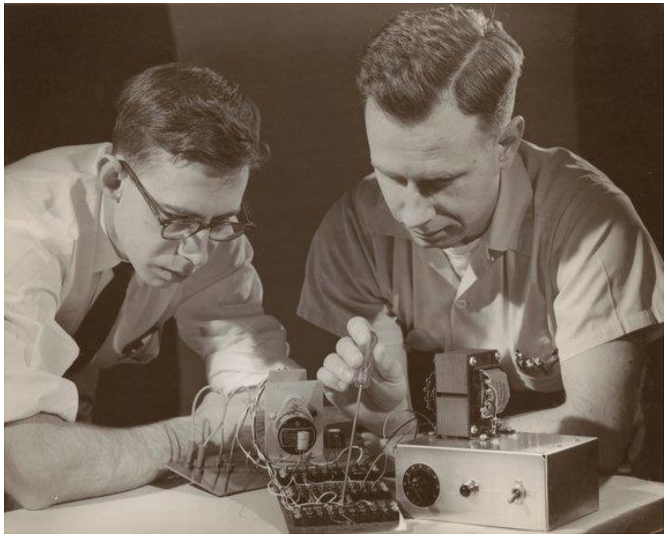

Cell body
Axon
Telodendria
Nucleus
Axon hillock
Synaptic terminals
Golgi apparatus
Endoplasmic reticulum
Mitochondrion
Dendrite
Dendritic branches

Basic model

- The first learning machine: the Perceptron (built in 1960)
- The perceptron was a linear classifier

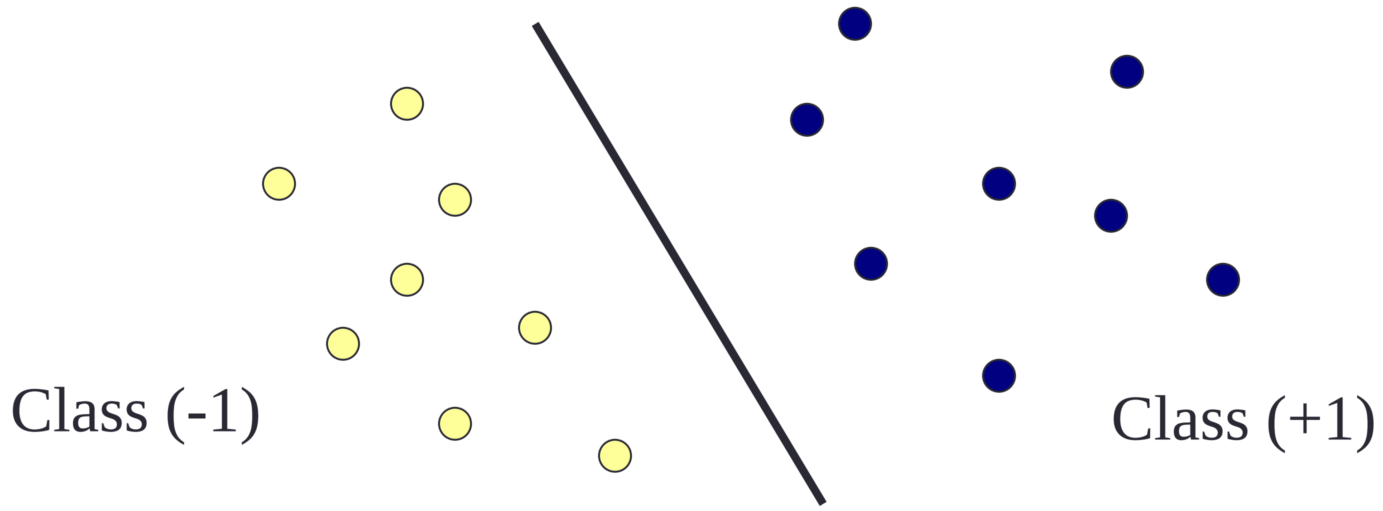


$$y = \mathrm{sign}(\mathrm{w}^{\mathrm{T}}\mathrm{x} + \mathrm{b})$$



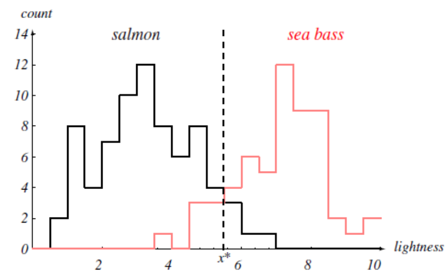
$$y = \begin{cases} +1 & \text{if } w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b > 0 \\ -1 & \text{otherwise} \end{cases}$$

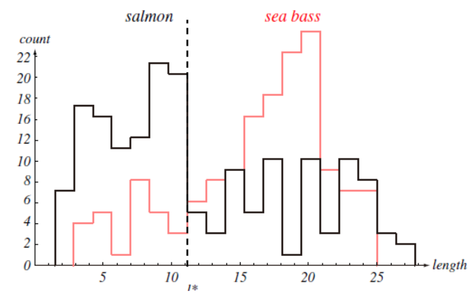
Linear Perceptron



- The goal: Find the best line (or hyper-plane) to separate the training data.
  - In two dimensions, the equation of the line is given by a line:
    - $ax + by + c = 0$
  - A better notation for *n* dimensions: treat each data point and the coefficients as vectors. Then the equation is given by:
    - $\mathrm{w}^{\mathrm{T}}\mathrm{x} + \mathrm{b} = 0$

예시: 연어와 농어의 구별


count
salmon
sea bass
lightness
x*
0
2
4
6
8
10
12
14


count
salmon
sea bass
length
l*
0
2
4
5
6
8
10
12
15
16
18
20
22
25

예시: 연어와 농어의 구별

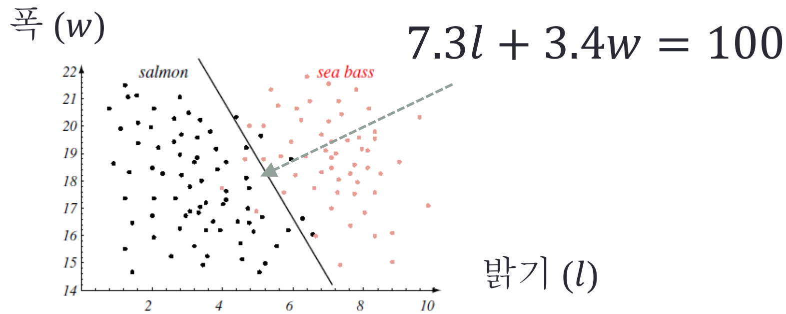
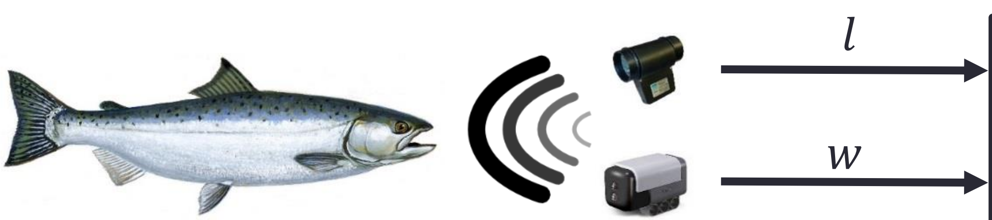
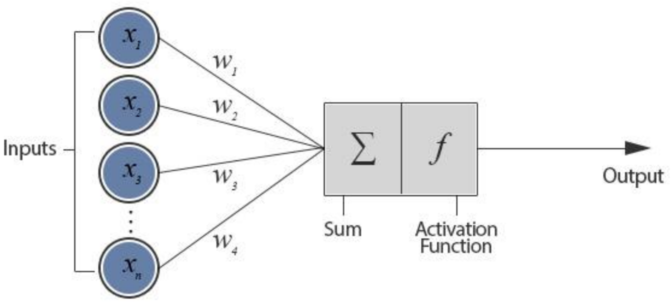


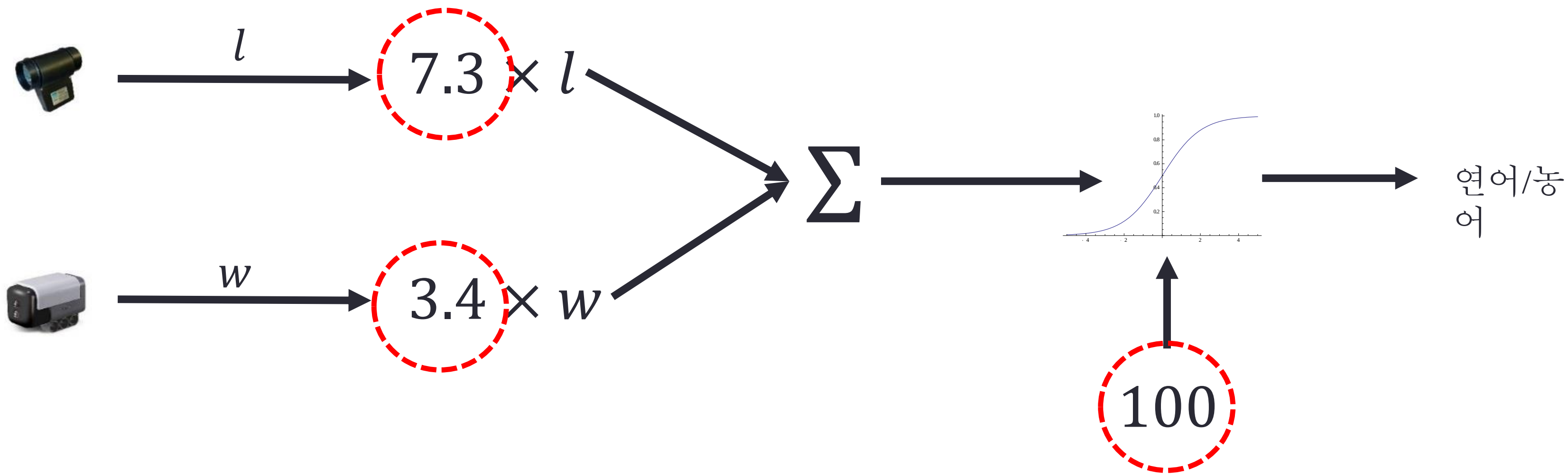
FIGURE 1.4. The two features of lightness and width for sea bass and salmon. The dark line could serve as a decision boundary of our classifier. Overall classification error on the data shown is lower than if we use only one feature as in Fig. 1.3, but there will still be some errors. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.



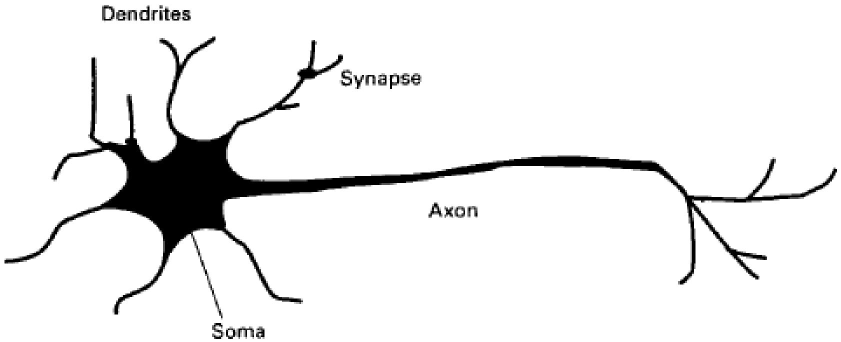
농어	$7.3l + 3.4w \geq 100$
연어	$7.3l + 3.4w < 100$

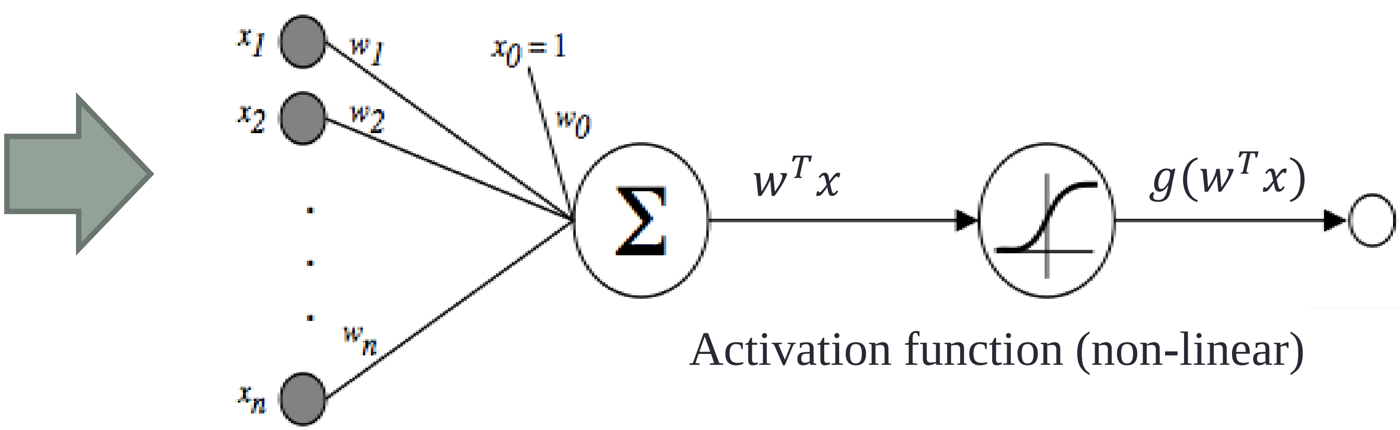
예시: 연어와 농어의 구별


Inputs
x_1
x_2
x_3
x_n
w_1
w_2
w_3
w_4
Σ
f
Output
Sum
Activation Function


l
$7.3 \times l$
w
$3.4 \times w$
Σ
100
연어/농어

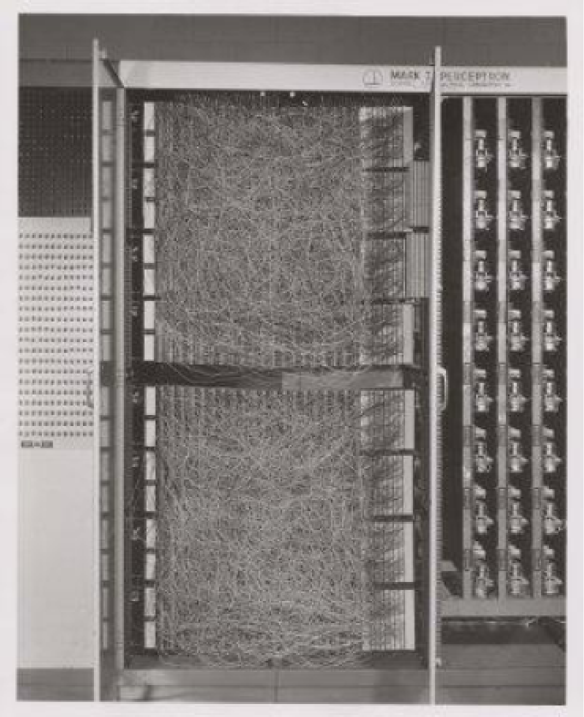
Artificial Neuron





Activation function (non-linear)

Mark I Perceptron



- Frank Rosenblatt
- 400 pixel image input
- Weights encoded in potentiometers
- Weight updated by electric motors

The New York Times

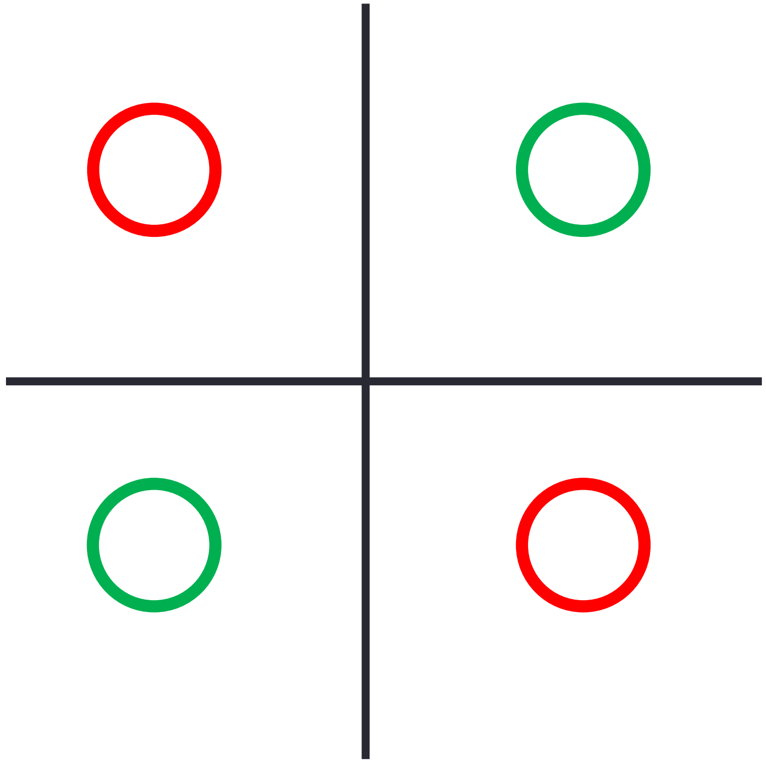
NEW NAVY DEVICE LEARNS BY DOING

July 8, 1958

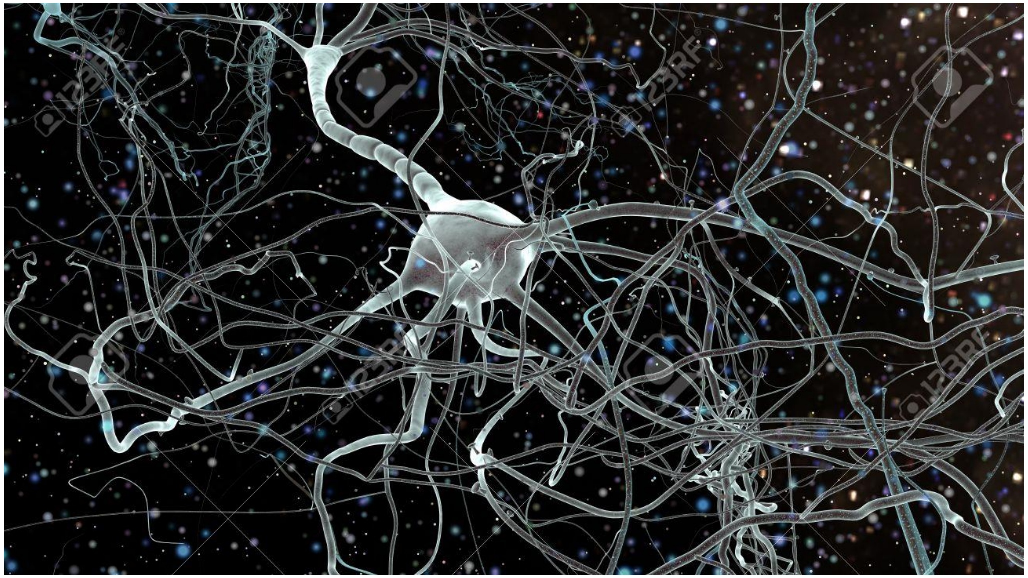
"The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence… Dr. Frank Rosenblatt, a research psychologist at the Cornell Aeronautical Laboratory, Buffalo, said Perceptrons might be fired to the planets as mechanical space explorers"

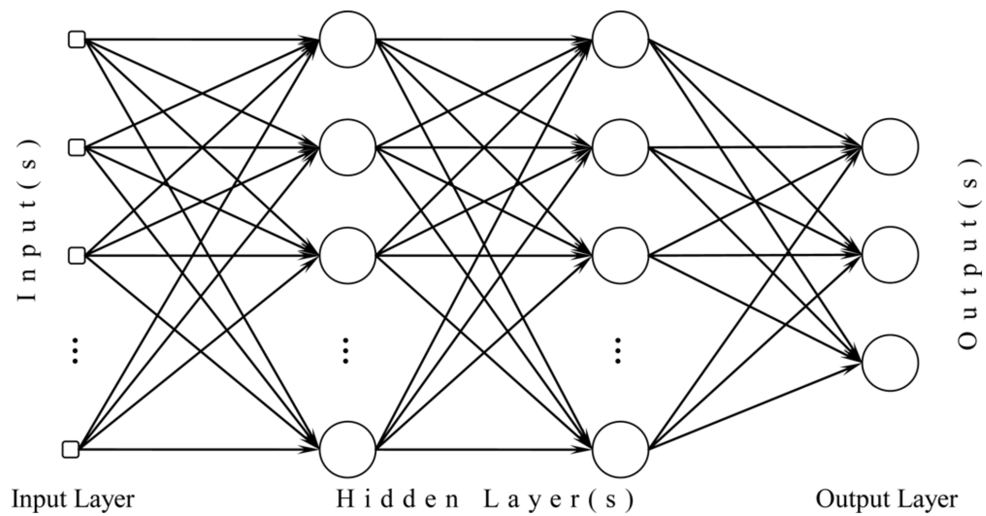
Artificial Neuron

- However, it cannot solve non-linearly-separable problems



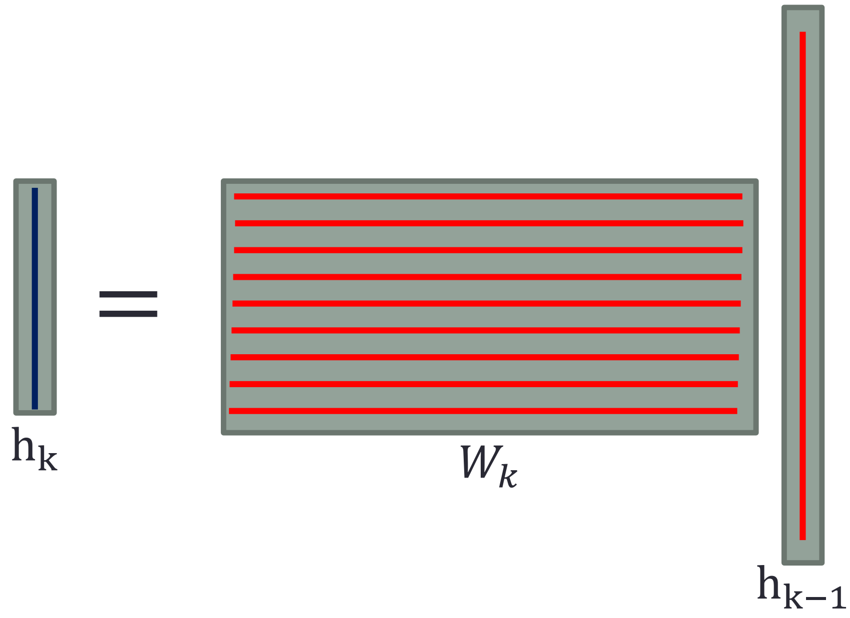
MULTI-LAYER PERCEPTRON




Input(s)
Output(s)
Input Layer
Hidden Layer(s)
Output Layer

Multi-layer Neural Network

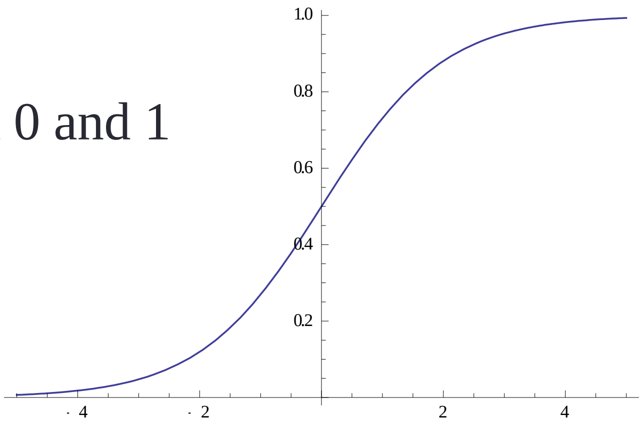
- 1^{st} Layer
  - $h_1 = g(W_1 x + b_1)$
- 2^{nd} Layer
  - $h_2 = g(W_2 h_1 + b_2)$
- …..
- Output layer
  - $o = \mathrm{softmax}(W_n h_{n-1} + b_n)$



Activation function $g(\cdot)$

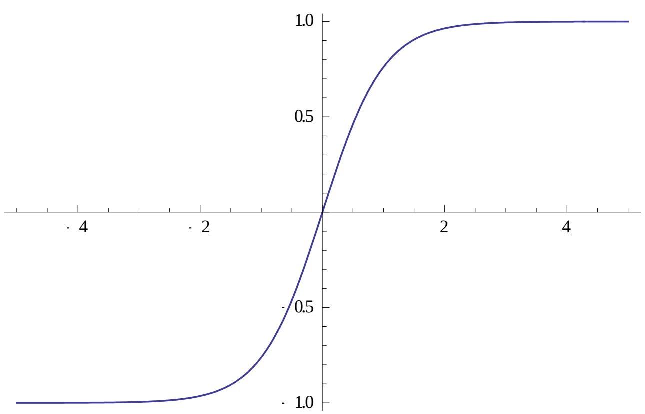
- Sigmoid activation function
  - Squashes the neuron's pre-activation between 0 and 1
  - Always positive/Bounded/Strictly increasing

$$g(x) = \frac{1}{1 + \exp(-x)}$$



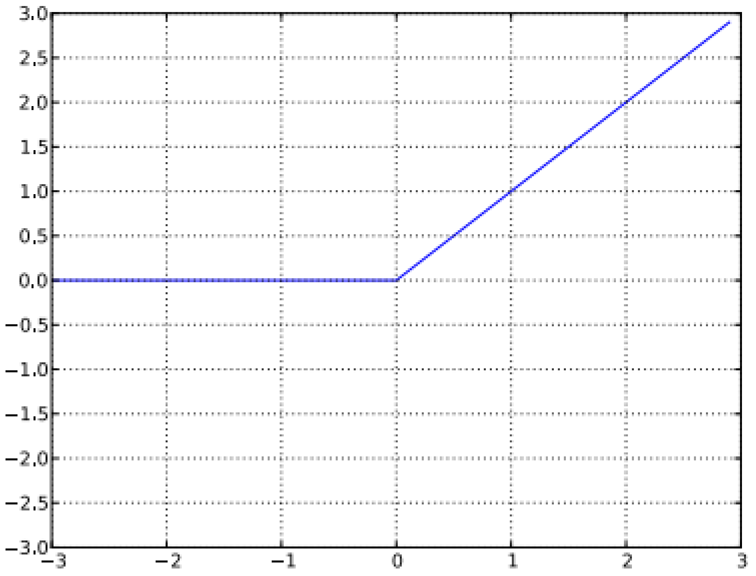
- Hyperbolic tangent ("tanh") activation function
  - Squashes the neuron's pre-activation between -1 and 1
  - Bounded/Strictly increasing

$$g(x) = \tanh(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}$$



Activation function $g(\cdot)$

- Rectified linear activation function (ReLU)
  - Bounded below by 0
  - Not upper bounded
  - Strictly increasing



$$g(a) = \text{rectlin}(a) = \max(0, a)$$

Soft-max activation function at the output

- For multi-class classification
  - We need multiple outputs (1 output per class)

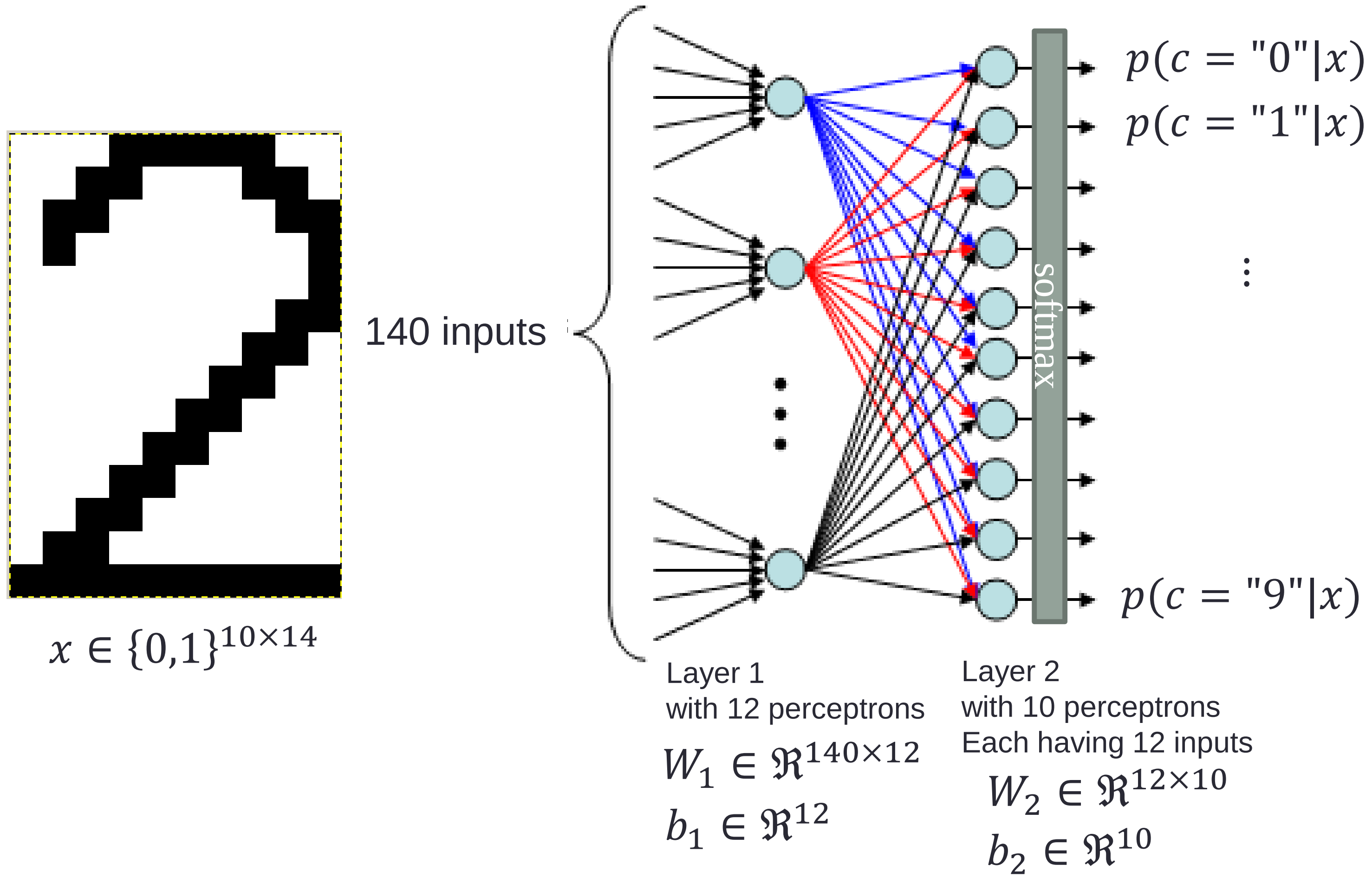
- We use the softmax activation function at the output

$$O(\mathbf{a}) = \operatorname{softmax}(\mathbf{a}) = \begin{bmatrix} \frac{\exp(a_1)}{\sum_c \exp(a_c)} \\ \frac{\exp(a_2)}{\sum_c \exp(a_c)} \\ \vdots \\ \vdots \\ \frac{\exp(a_c)}{\sum_c \exp(a_c)} \end{bmatrix}$$

  - strictly positive
  - sums to one

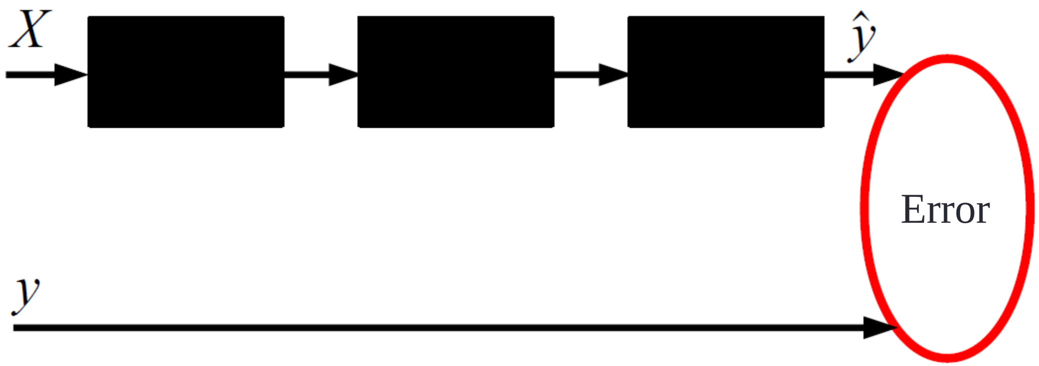
예시: $(a, b, c) \rightarrow \left(\frac{e^a}{e^a+e^b+e^c}, \frac{e^b}{e^a+e^b+e^c}, \frac{e^c}{e^a+e^b+e^c} \right)$

Example (character recognition example)



TRAINING OF MULTI-LAYER PERCEPTRON

Training: Loss function



- Cross entropy (classification)
  - $y, \hat{y} \in [0,1]^N, \sum_{i=1} y_i = 1, \sum_{i=1} \hat{y}_i = 1$
  - $L = -\sum y_i \log \hat{y}_i$

- Square Euclidean distance (regression)
  - $y, \hat{y} \in \Re^N$
  - $L = \frac{1}{2} \sum (y_i - \hat{y}_i)^2$

Cross Entropy (예시)

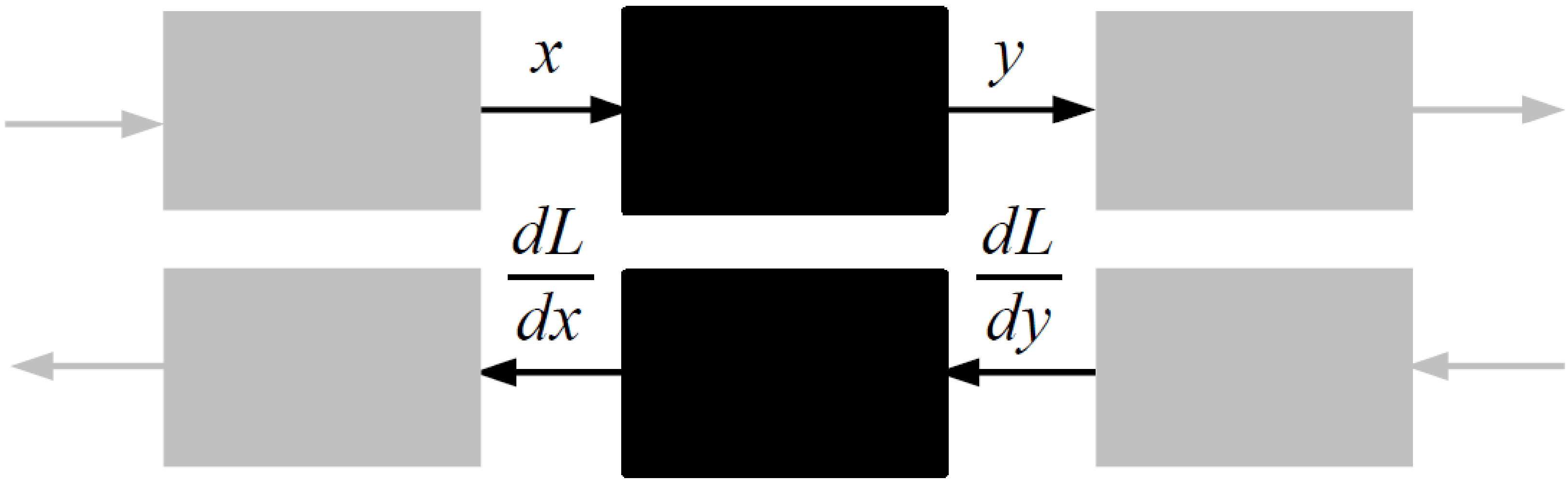
- Label:
  - $[y_1 \ \ y_2 \ \ y_3]$ = [1,0,0] : class 1
  - $[y_1 \ \ y_2 \ \ y_3]$ = [0,1,0] : class 2
  - $[y_1 \ \ y_2 \ \ y_3]$ = [0,0,1] : class 3

$$L = -\sum y_i \log \hat{y}_i$$

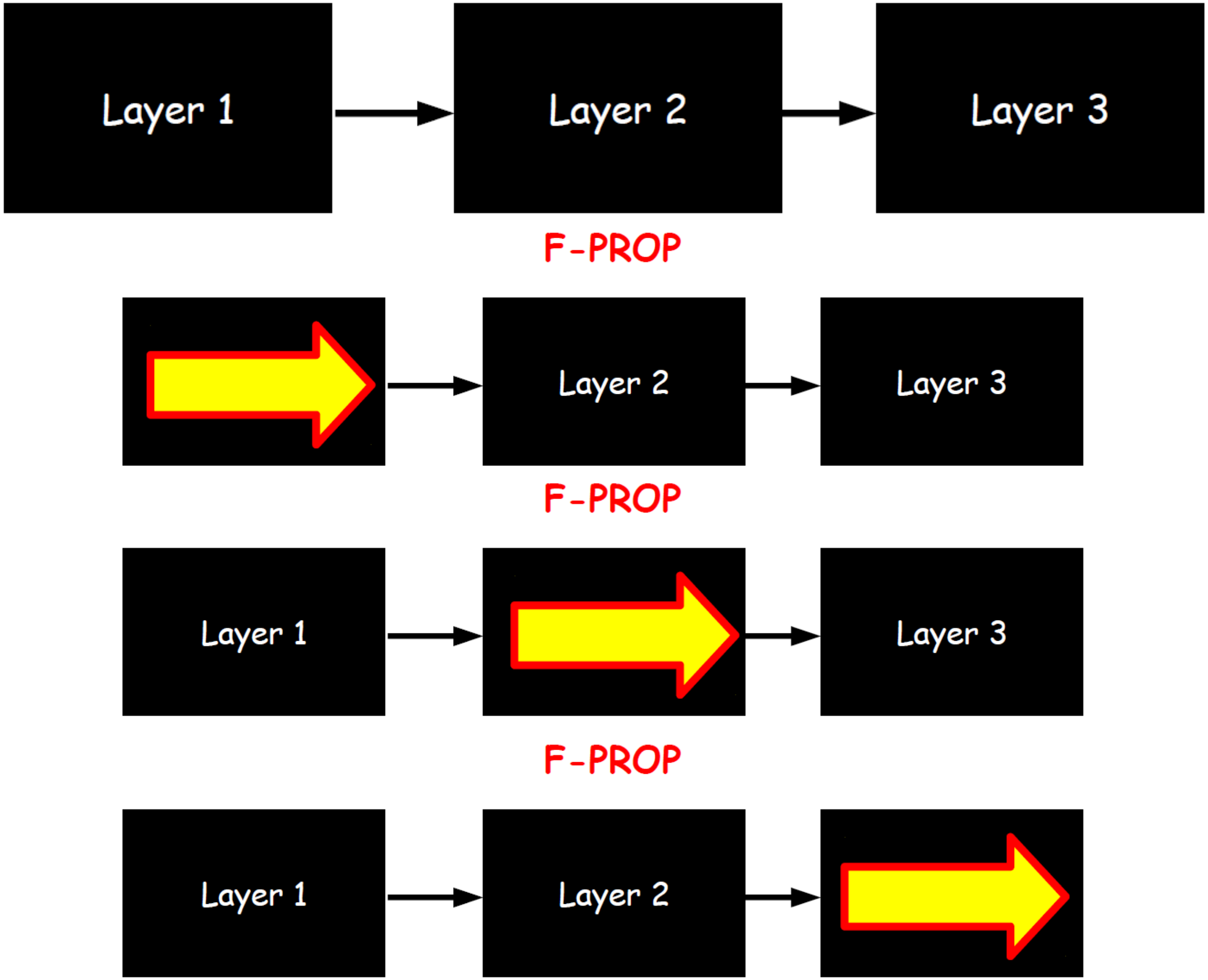
- 예시
  - Network output: $\hat{y} = [\hat{y}_1 \ \ \hat{y}_2 \ \ \hat{y}_3] = [0.3, 0.6, 0.1]$
    - Loss
      - If ground truth is class 1 (i.e., y = [1,0,0]) → $-\log 0.3 = 1.204$
      - If ground truth is class 2 (i.e., y = [0,1,0]) → $-\log 0.6 = 0.511$
      - If ground truth is class 3 (i.e., y = [0,0,1]) → $-\log 0.1 = 2.303$
  - Network output: $\hat{y} = [\hat{y}_1 \ \ \hat{y}_2 \ \ \hat{y}_3]$= [0.01, 0.98, 0.01]
    - Loss
      - If ground truth is class 1 (i.e., y = [1,0,0]) → $-\log 0.01 = 4.605$
      - If ground truth is class 2 (i.e., y = [0,1,0]) → $-\log 0.98 = 0.020$
      - If ground truth is class 3 (i.e., y = [0,0,1]) → $-\log 0.01 = 4.605$

Forward/Backward propagation

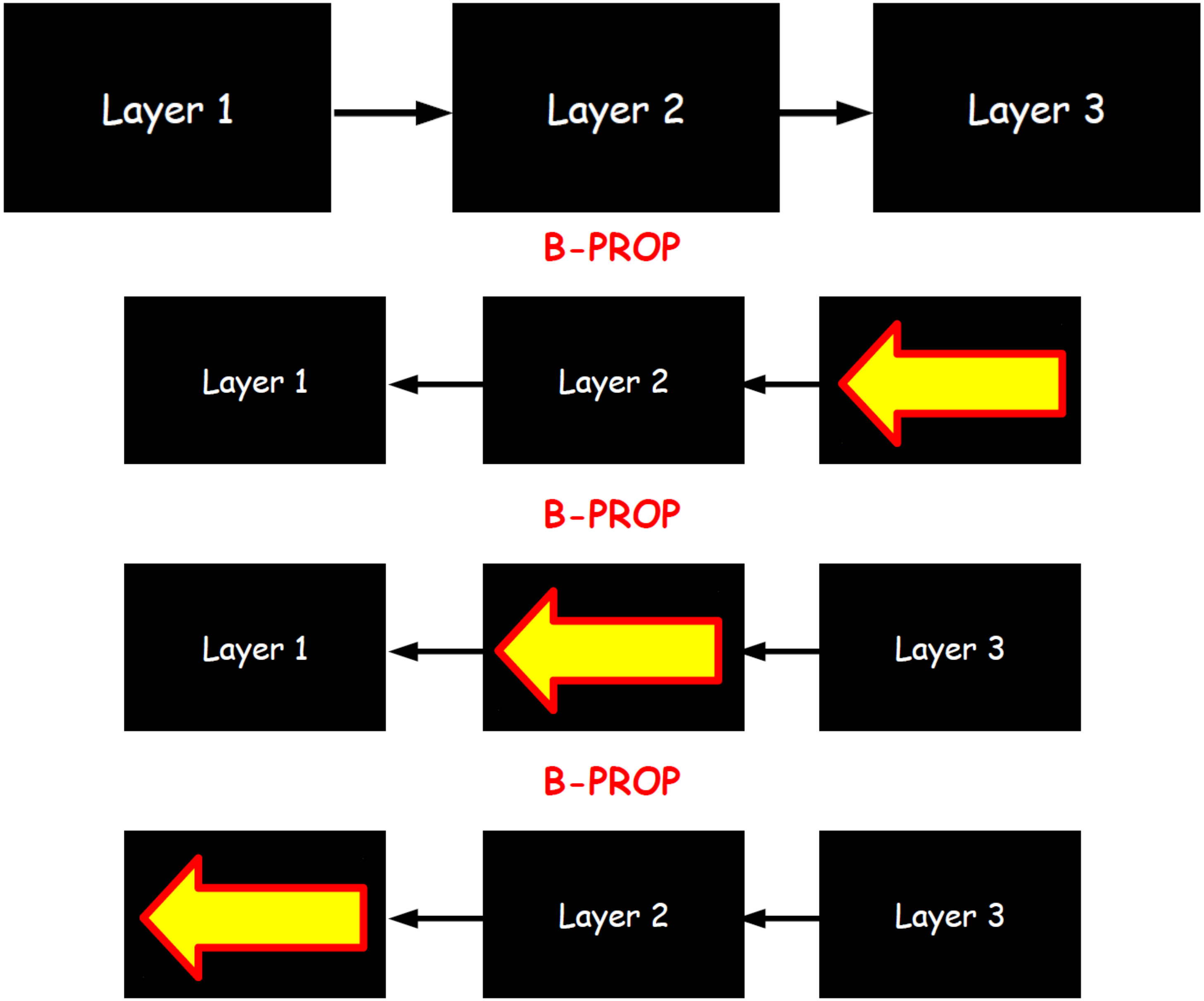
- Chain rule



$$W^{new} = W^{old} - \eta \frac{dL}{dW}$$

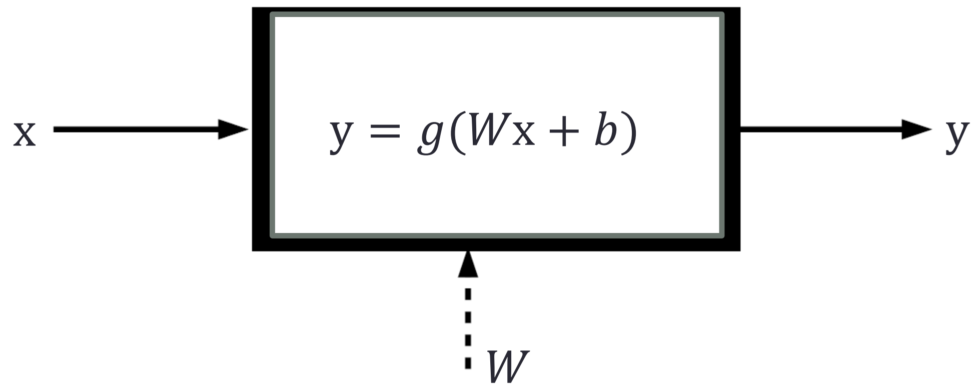


Layer 1
Layer 2
Layer 3
F-PROP
Layer 2
Layer 3
F-PROP
Layer 1
Layer 3
F-PROP
Layer 1
Layer 2



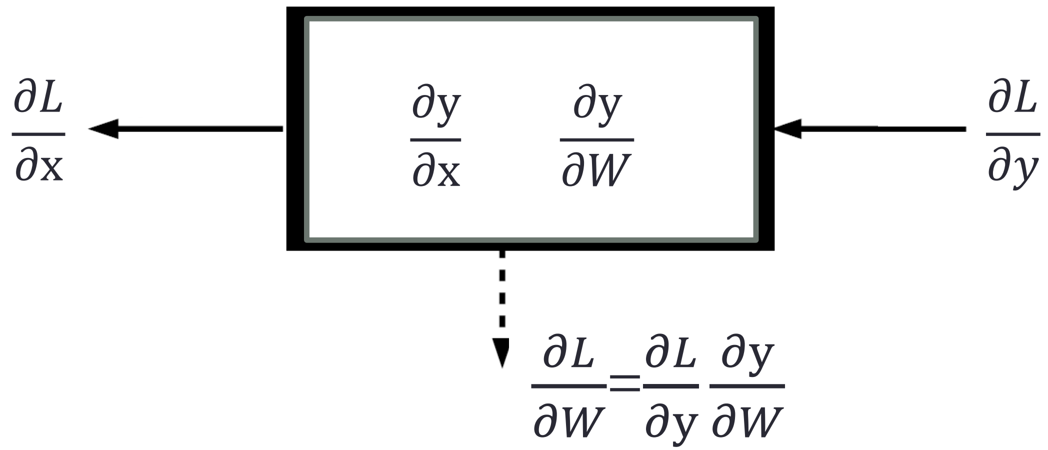
Layer 1
Layer 2
Layer 3
B-PROP
Layer 1
Layer 2
B-PROP
Layer 1
Layer 3
B-PROP
Layer 2
Layer 3

Forward/Backward propagation



Forward propagation

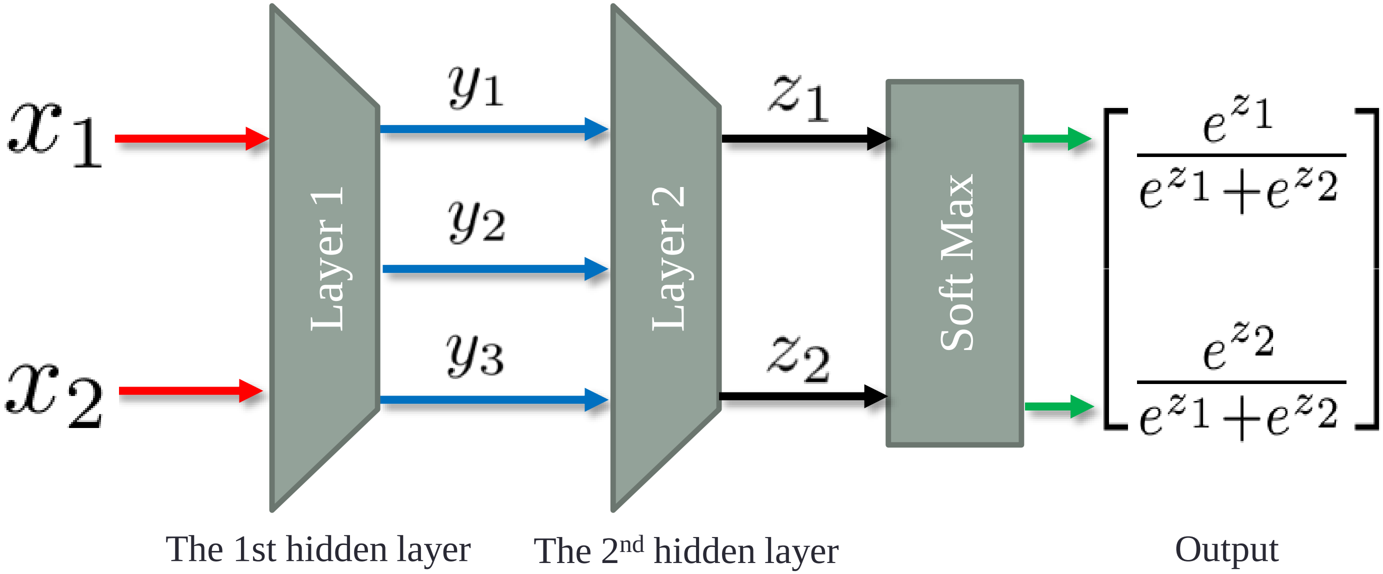
$$\frac{\partial L}{\partial \mathrm{x}} = \frac{\partial L}{\partial \mathrm{y}} \cdot \frac{\partial \mathrm{y}}{\partial \mathrm{x}}$$



Backward propagation

FEED-FORWARD NEURAL NETWORK (예시)

Forward propagation



$$y_1 = \varphi(w_{11}x_1 + w_{12}x_2 + b_1)$$

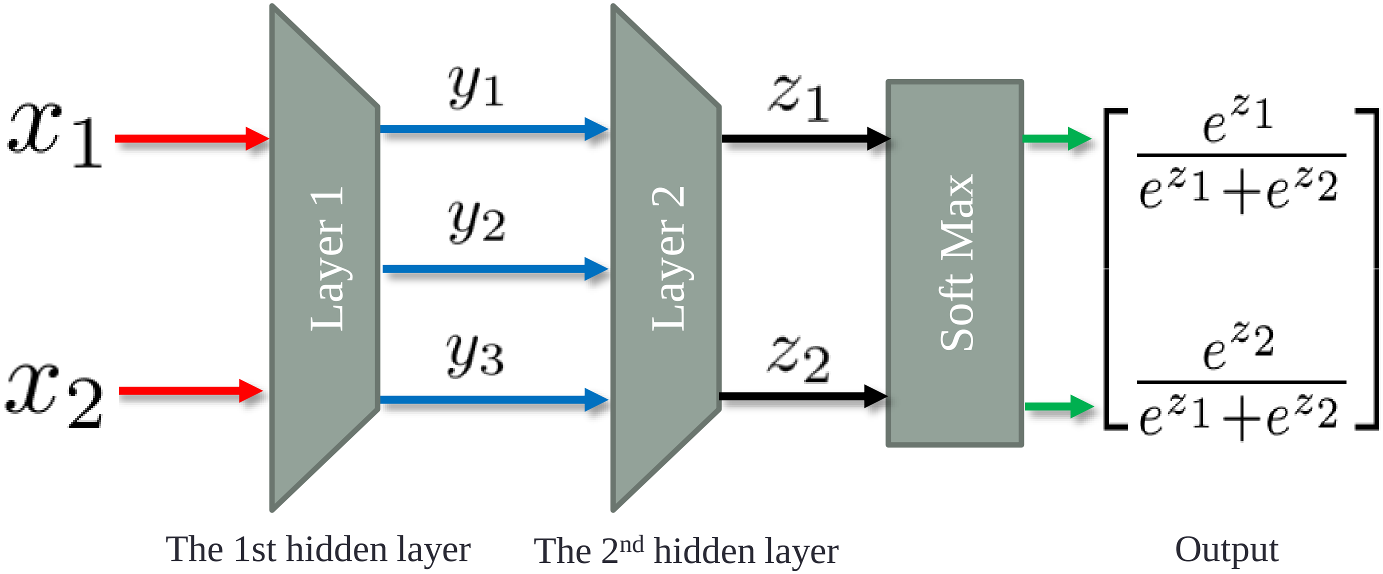
$$y_2 = \varphi(w_{21}x_1 + w_{22}x_2 + b_2)$$

$$y_3 = \varphi(w_{31}x_1 + w_{32}x_2 + b_3)$$

$$z_1 = u_{11}y_1 + u_{12}y_2 + u_{13}y_3 + c_1$$

$$z_2 = u_{21}y_1 + u_{22}y_2 + u_{23}y_3 + c_2$$

Forward propagation matrix repr.

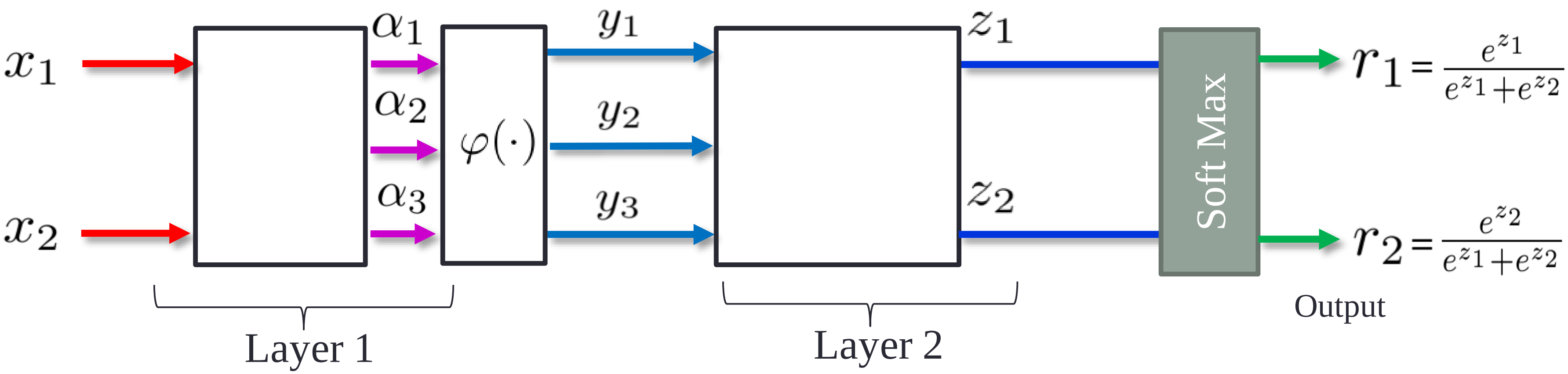


$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \varphi\left(\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \right)$$

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} u_{11} & w_{12} & u_{13} \\ u_{21} & w_{22} & u_{23} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} + \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}$$

BACK-PROPAGATION ALGORITHM (예시)

Forward propagation (block-based representation)



Layer 1

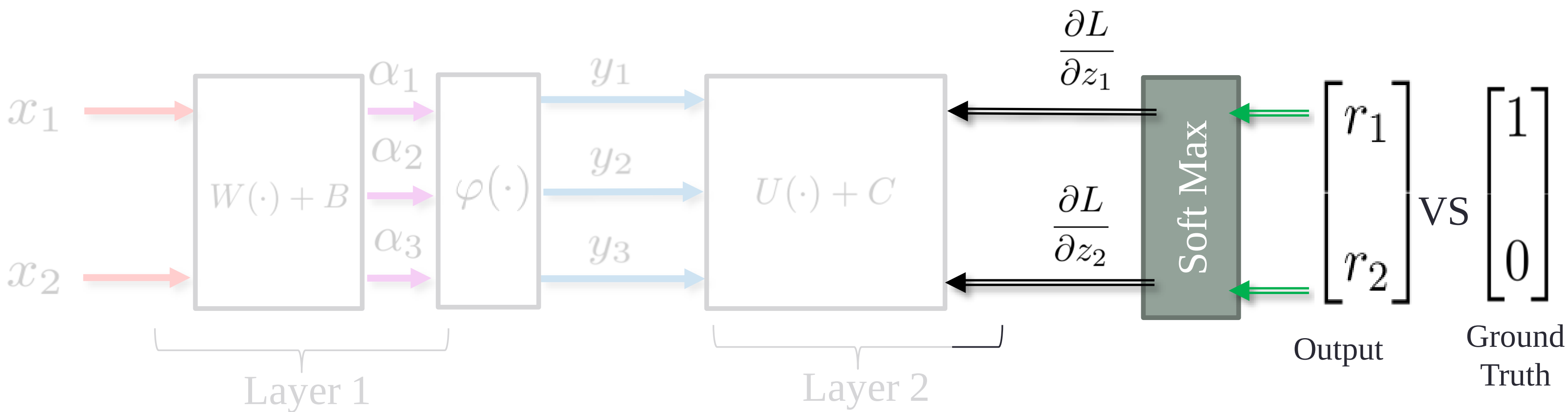
$$\begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} \varphi(\alpha_1) \\ \varphi(\alpha_2) \\ \varphi(\alpha_3) \end{bmatrix}$$

Layer 2

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} u_{11} & w_{12} & u_{13} \\ u_{21} & w_{22} & u_{23} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} + \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}$$

Backward propagation; 2nd layer

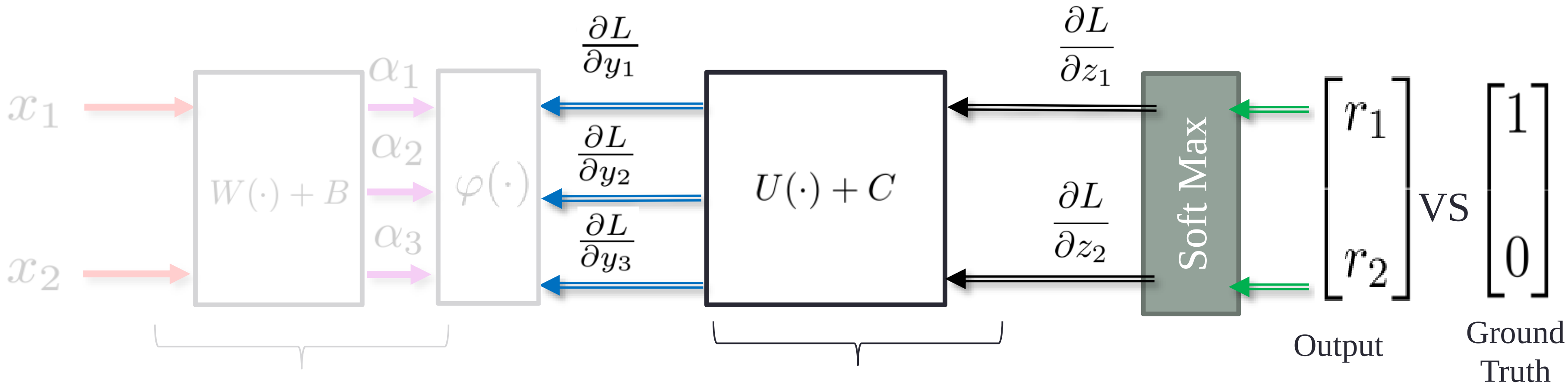


- Error propagation

$$\frac{\partial L}{\partial z_1} = -1 + r_1$$

$$\frac{\partial L}{\partial z_2} = r_2$$

Backward propagation; 2nd layer



- Error propagation

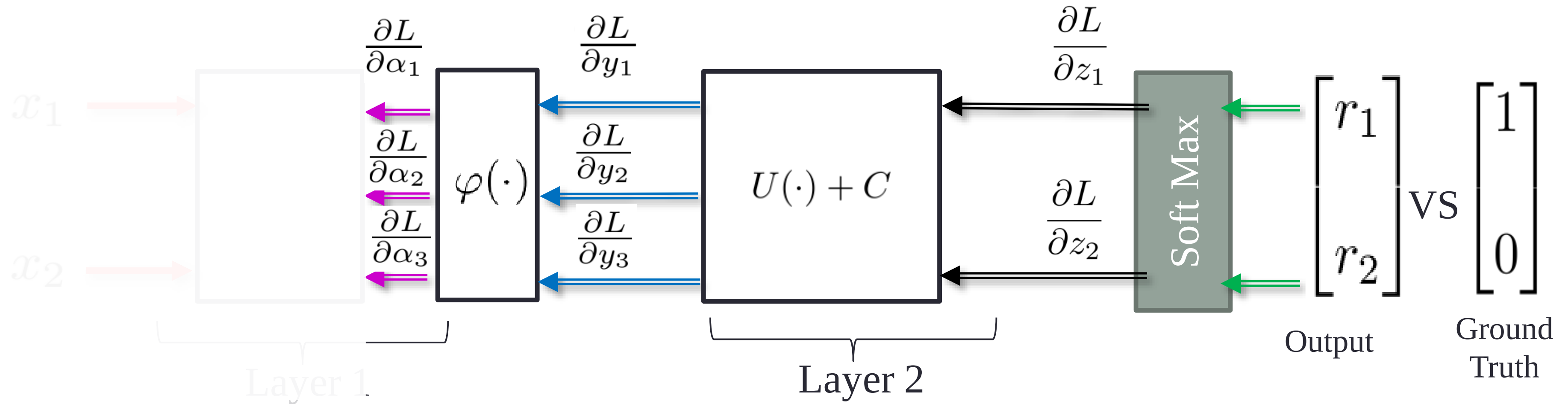
$$\begin{bmatrix} \frac{\partial L}{\partial y_1} \\ \frac{\partial L}{\partial y_2} \\ \frac{\partial L}{\partial y_3} \end{bmatrix} = \begin{bmatrix} u_{11} & u_{21} \\ u_{12} & u_{22} \\ u_{13} & u_{23} \end{bmatrix} \begin{bmatrix} \frac{\partial L}{\partial z_1} \\ \frac{\partial L}{\partial z_2} \end{bmatrix}$$

- Weight update

$$\begin{bmatrix} \frac{\partial L}{\partial u_{11}} & \frac{\partial L}{\partial u_{12}} & \frac{\partial L}{\partial u_{13}} \\ \frac{\partial L}{\partial u_{21}} & \frac{\partial L}{\partial u_{22}} & \frac{\partial L}{\partial u_{23}} \end{bmatrix} = \begin{bmatrix} \frac{\partial L}{\partial z_1} \\ \frac{\partial L}{\partial z_2} \end{bmatrix} \begin{bmatrix} y_1 & y_2 & y_3 \end{bmatrix}$$

$$\begin{bmatrix} \frac{\partial L}{\partial c_1} \\ \frac{\partial L}{\partial c_2} \end{bmatrix} = \begin{bmatrix} \frac{\partial L}{\partial z_1} \\ \frac{\partial L}{\partial z_2} \end{bmatrix}$$

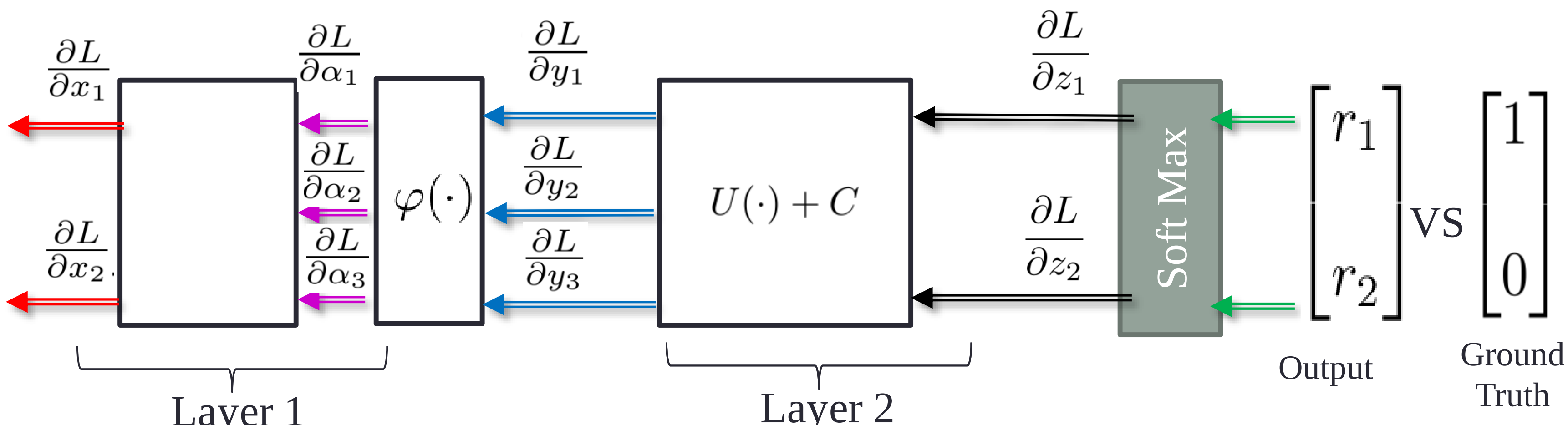
Backward propagation; 1st layer



- Error propagation

$$\begin{bmatrix} \frac{\partial L}{\partial \alpha_1} \\ \frac{\partial L}{\partial \alpha_2} \\ \frac{\partial L}{\partial \alpha_3} \end{bmatrix} = \begin{bmatrix} \varphi'(\alpha_1) & 0 & 0 \\ 0 & \varphi'(\alpha_2) & 0 \\ 0 & 0 & \varphi'(\alpha_3) \end{bmatrix} \begin{bmatrix} \frac{\partial L}{\partial y_1} \\ \frac{\partial L}{\partial y_2} \\ \frac{\partial L}{\partial y_3} \end{bmatrix}$$

Backward propagation; 1st layer



- Error propagation

$$\begin{bmatrix} \frac{\partial L}{\partial x_1} \\ \frac{\partial L}{\partial x_2} \end{bmatrix} = \begin{bmatrix} w_{11} & w_{21} & w_{31} \\ w_{12} & w_{22} & w_{32} \end{bmatrix} \begin{bmatrix} \frac{\partial L}{\partial \alpha_1} \\ \frac{\partial L}{\partial \alpha_2} \\ \frac{\partial L}{\partial \alpha_3} \end{bmatrix}$$

- Weight update

$$\begin{bmatrix} \frac{\partial L}{\partial w_{11}} & \frac{\partial L}{\partial w_{12}} \\ \frac{\partial L}{\partial w_{21}} & \frac{\partial L}{\partial w_{22}} \\ \frac{\partial L}{\partial w_{31}} & \frac{\partial L}{\partial w_{32}} \end{bmatrix} = \begin{bmatrix} \frac{\partial L}{\partial \alpha_1} \\ \frac{\partial L}{\partial \alpha_2} \\ \frac{\partial L}{\partial \alpha_3} \end{bmatrix} \begin{bmatrix} x_1 & x_2 \end{bmatrix}$$

$$\begin{bmatrix} \frac{\partial L}{\partial b_1} \\ \frac{\partial L}{\partial b_2} \\ \frac{\partial L}{\partial b_3} \end{bmatrix} = \begin{bmatrix} \frac{\partial L}{\partial \alpha_1} \\ \frac{\partial L}{\partial \alpha_2} \\ \frac{\partial L}{\partial \alpha_3} \end{bmatrix}$$

TENSORFLOW 실습

TENSORFLOW INTRODUCTION

What is TensorFlow?

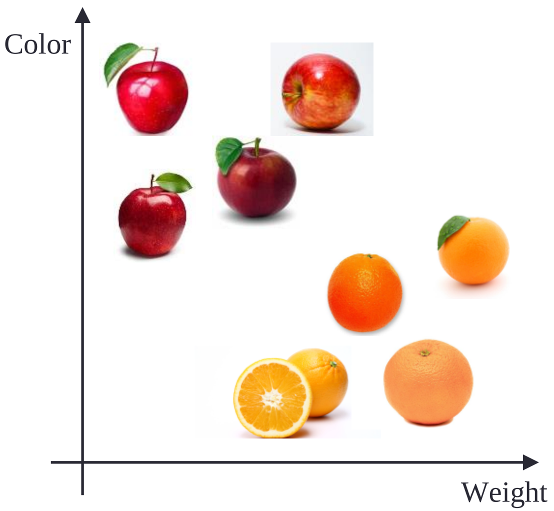
- TensorFlow is a deep learning library open-sourced by Google.
- TensorFlow provides primitives for defining functions on tensors and automatically computing their derivatives.
- Tensor is a multidimensional array of numbers

Design Choice

- Network structures
  - The mathematical relationship between inputs and outputs
- Loss function
- Optimization
  - Optimization methods
  - Hyper-parameters (Batch size, Learning rate, …)

Classification vs Regression

Classification



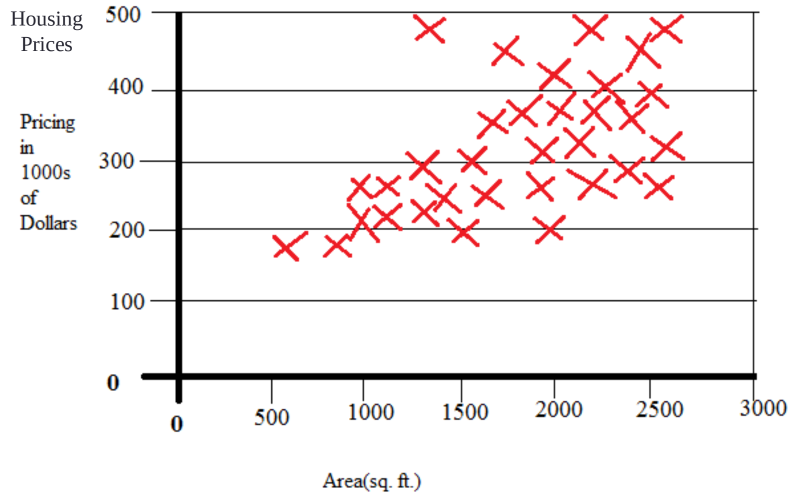
?

Apple

Orange

The variable we are trying to predict is
DISCRETE

Regression



1600 sq ft

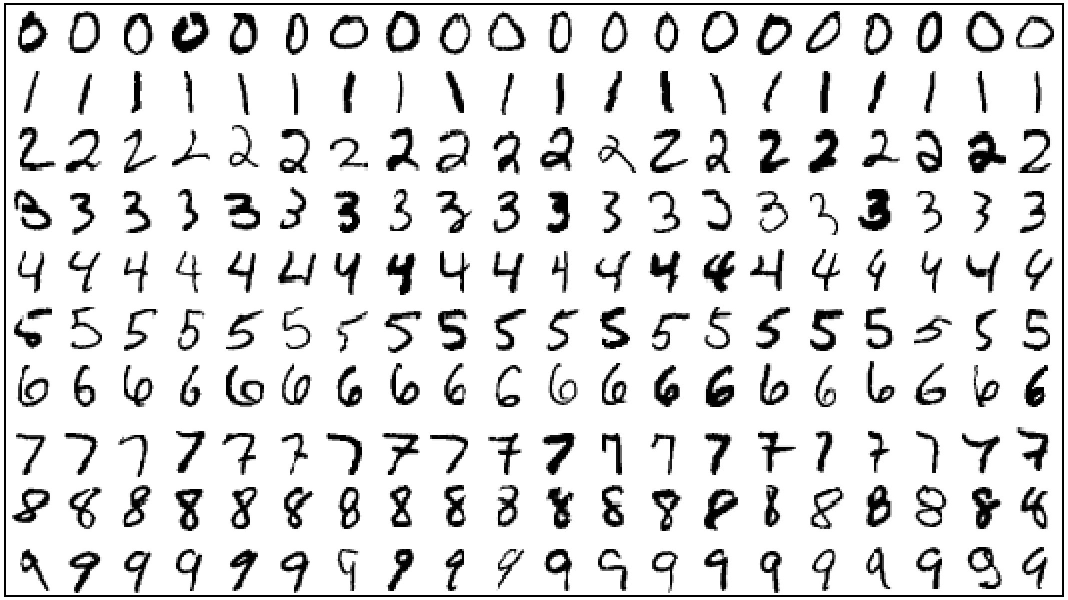
?

Expected Price

The variable we are trying to predict is
CONTINUOUS

MNIST dataset (classification example)

- handwritten digits
- a training set of 60,000 examples
- 28x28 images

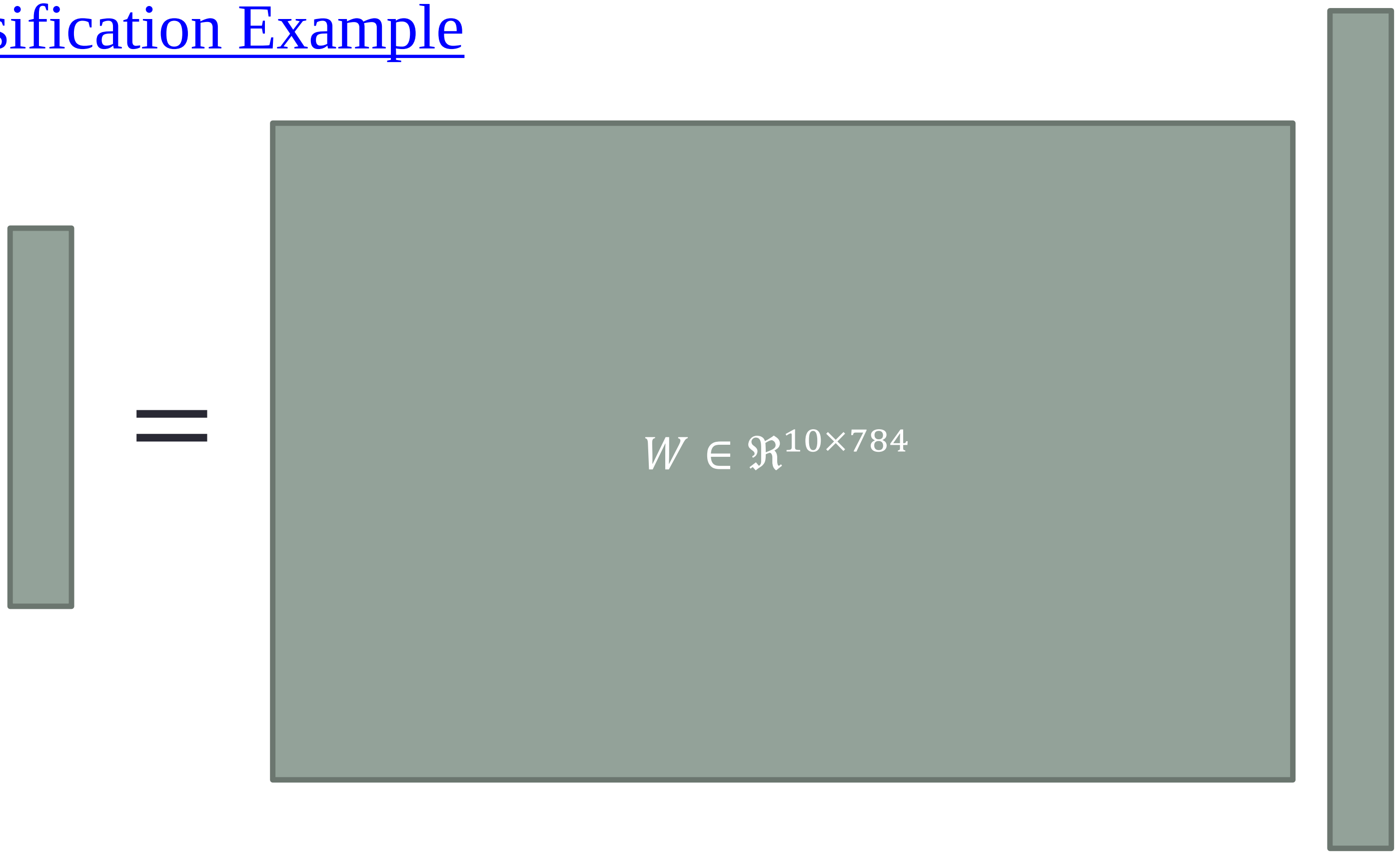


CLASSIFIER	PREPROCESSING	TEST ERROR RATE (%)	Reference
Linear Classifiers			
linear classifier (1-layer NN)	none	12.0	LeCun et al. 1998
linear classifier (1-layer NN)	deskewing	8.4	LeCun et al. 1998
pairwise linear classifier	deskewing	7.6	LeCun et al. 1998
Non-Linear Classifiers			
40 PCA + quadratic classifier	none	3.3	LeCun et al. 1998
1000 RBF + linear classifier	none	3.6	LeCun et al. 1998
SVMs			
SVM, Gaussian Kernel	none	1.4	
SVM deg 4 polynomial	deskewing	1.1	LeCun et al. 1998
Reduced Set SVM deg 5 polynomial	deskewing	1.0	LeCun et al. 1998
Virtual SVM deg-9 poly [distortions]	none	0.8	LeCun et al. 1998
Virtual SVM, deg-9 poly, 1-pixel jittered	none	0.68	DeCoste and Scholkopf, MLJ 2002
Virtual SVM, deg-9 poly, 1-pixel jittered	deskewing	0.68	DeCoste and Scholkopf, MLJ 2002
Virtual SVM, deg-9 poly, 2-pixel jittered	deskewing	0.56	DeCoste and Scholkopf, MLJ 2002
Neural Nets			
2-layer NN, 300 hidden units, mean square error	none	4.7	LeCun et al. 1998
2-layer NN, 300 HU, MSE, [distortions]	none	3.6	LeCun et al. 1998
2-layer NN, 300 HU	deskewing	1.6	LeCun et al. 1998
2-layer NN, 1000 hidden units	none	4.5	LeCun et al. 1998
2-layer NN, 1000 HU, [distortions]	none	3.8	LeCun et al. 1998
3-layer NN, 300+100 hidden units	none	3.05	LeCun et al. 1998
3-layer NN, 300+100 HU [distortions]	none	2.5	LeCun et al. 1998
3-layer NN, 500+150 hidden units	none	2.95	LeCun et al. 1998
3-layer NN, 500+150 HU [distortions]	none	2.45	LeCun et al. 1998
3-layer NN, 500+300 HU, softmax, cross entropy, weight decay	none	1.53	Hinton, unpublished, 2005
2-layer NN, 800 HU, Cross-Entropy Loss	none	1.6	Simard et al., ICDAR 2003
2-layer NN, 800 HU, cross-entropy [affine distortions]	none	1.1	Simard et al., ICDAR 2003
2-layer NN, 800 HU, MSE [elastic distortions]	none	0.9	Simard et al., ICDAR 2003

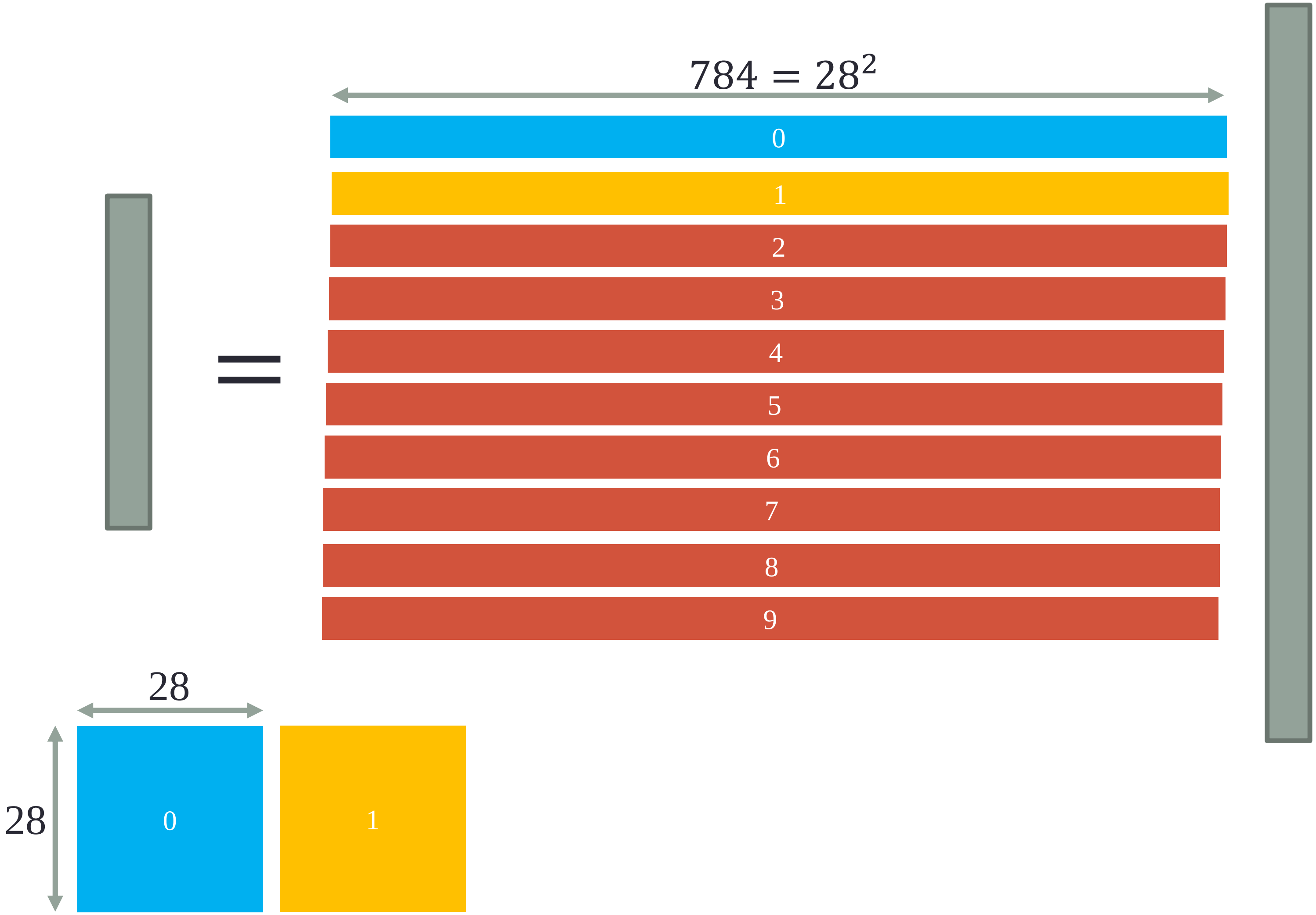
Convolutional nets			
Convolutional net LeNet-1	subsampling to 16x16 pixels	1.7	LeCun et al. 1998
Convolutional net LeNet-4	none	1.1	LeCun et al. 1998
Convolutional net LeNet-4 with K-NN instead of last layer	none	1.1	LeCun et al. 1998
Convolutional net LeNet-4 with local learning instead of last layer	none	1.1	LeCun et al. 1998
Convolutional net LeNet-5, [no distortions]	none	0.95	LeCun et al. 1998
Convolutional net LeNet-5, [huge distortions]	none	0.85	LeCun et al. 1998
Convolutional net LeNet-5, [distortions]	none	0.8	LeCun et al. 1998
Convolutional net Boosted LeNet-4, [distortions]	none	0.7	LeCun et al. 1998
Trainable feature extractor + SVMs [no distortions]	none	0.83	Lauer et al., Pattern Recognition 40-6, 2007
Trainable feature extractor + SVMs [elastic distortions]	none	0.56	Lauer et al., Pattern Recognition 40-6, 2007
Trainable feature extractor + SVMs [affine distortions]	none	0.54	Lauer et al., Pattern Recognition 40-6, 2007
unsupervised sparse features + SVM, [no distortions]	none	0.59	Labusch et al., IEEE TNN 2008
Convolutional net, cross-entropy [affine distortions]	none	0.6	Simard et al., ICDAR 2003
Convolutional net, cross-entropy [elastic distortions]	none	0.4	Simard et al., ICDAR 2003
large conv. net, random features [no distortions]	none	0.89	Ranzato et al., CVPR 2007
large conv. net, unsup features [no distortions]	none	0.62	Ranzato et al., CVPR 2007
large conv. net, unsup pretraining [no distortions]	none	0.60	Ranzato et al., NIPS 2006
large conv. net, unsup pretraining [elastic distortions]	none	0.39	Ranzato et al., NIPS 2006
large conv. net, unsup pretraining [no distortions]	none	0.53	Jarrett et al., ICCV 2009
large/deep conv. net, 1-20-40-60-80-100-120-120-10 [elastic distortions]	none	0.35	Ciresan et al. IJCAI 2011
committee of 7 conv. net, 1-20-P-40-P-150-10 [elastic distortions]	width normalization	0.27 +-0.02	Ciresan et al. ICDAR 2011
committee of 35 conv. net, 1-20-P-40-P-150-10 [elastic distortions]	width normalization	0.23	Ciresan et al. CVPR 2012

Classification Example Code

- [Classification Example]



Classification Example Code



Regression Example Code

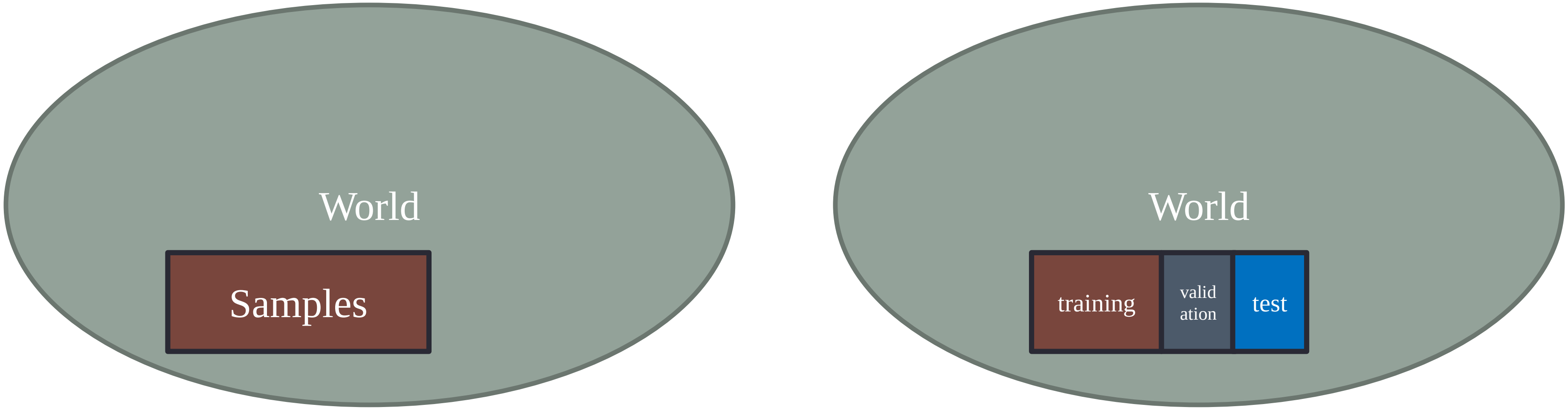
- Regression Example

VALIDATION

<table>
<tr><td colspan="5" rowspan="4"></td><th colspan="6">Tasks</th></tr>
<tr><th colspan="5">ADAS</th><th></th></tr>
<tr><th colspan="3">Self Driving</th><th></th><th></th><th></th></tr>
<tr><th>Localization</th><th>Perception</th><th>Planning/ Control</th><th>Driver state</th><th>Vehicle Diagnosis</th><th>Smart factory</th></tr>
<tr><td rowspan="7">Methods</td><td rowspan="2">Traditional</td><td colspan="3">Non-machine Learning</td><td>GPS, SLAM</td><td></td><td>Optimal control</td><td></td><td></td><td></td></tr>
<tr><td rowspan="6">Machine-Learning based method</td><td rowspan="4">Supervised</td><td>MLP</td><td></td><td>Pedestrian detection (HOG+SVM)</td><td></td><td></td><td></td><td></td></tr>
<tr><td rowspan="5">Deep-Learning based</td><td>CNN</td><td></td><td>Detection/ Segmentation/Classification</td><td>End-to-end Learning</td><td></td><td></td><td></td></tr>
<tr><td>RNN (LSTM)</td><td></td><td>Dry/wet road classification</td><td>End-to-end Learning</td><td></td><td></td><td></td></tr>
<tr><td>DNN</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td colspan="2">Reinforcement</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td colspan="2">Unsupervised</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

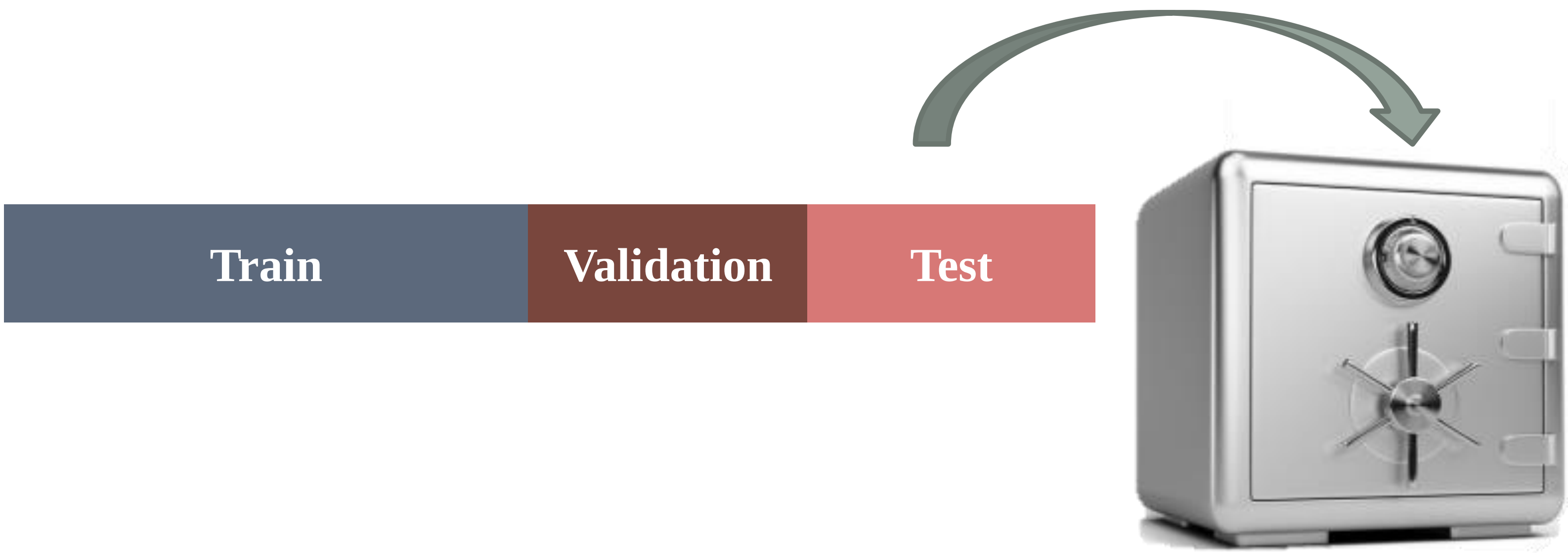
Validation set approach

- Divide the data in three parts:
  - training, validation (development), and test. We use the train and validation data to select the best model and the test data to assess the chosen model.



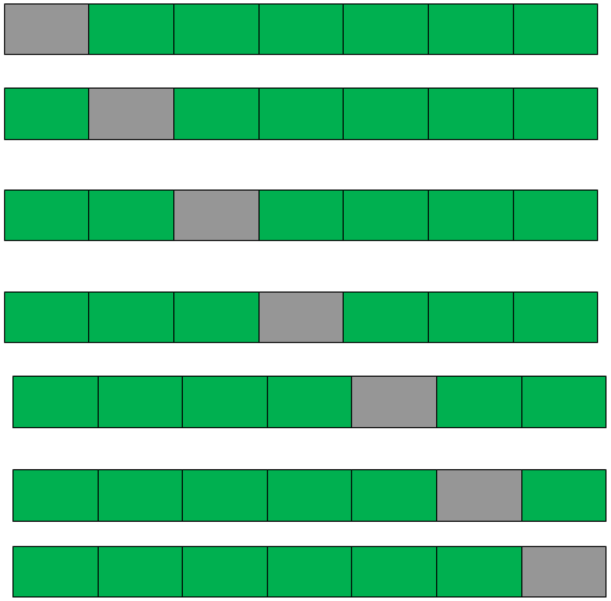
Validation set approach

- Training set
  - To fit the models
- Validation set
  - To estimate prediction error for model selection
- Test set
  - To assess of the generalization error of the final chose model



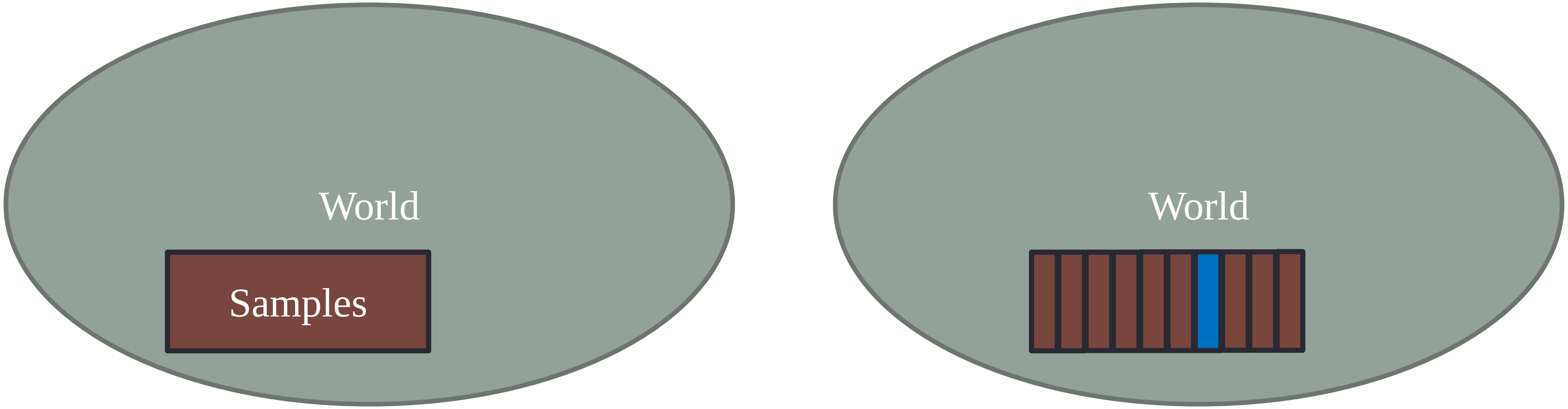
k-fold cross validation

- We partition the data into K parts. For the k –th part, we fit the model to the other $K-1$ parts of the data, and calculate the prediction error of the fitted model when predicting the kth part of the data. We do this for $k = 1,2,\cdots,K$ and combine the K estimates

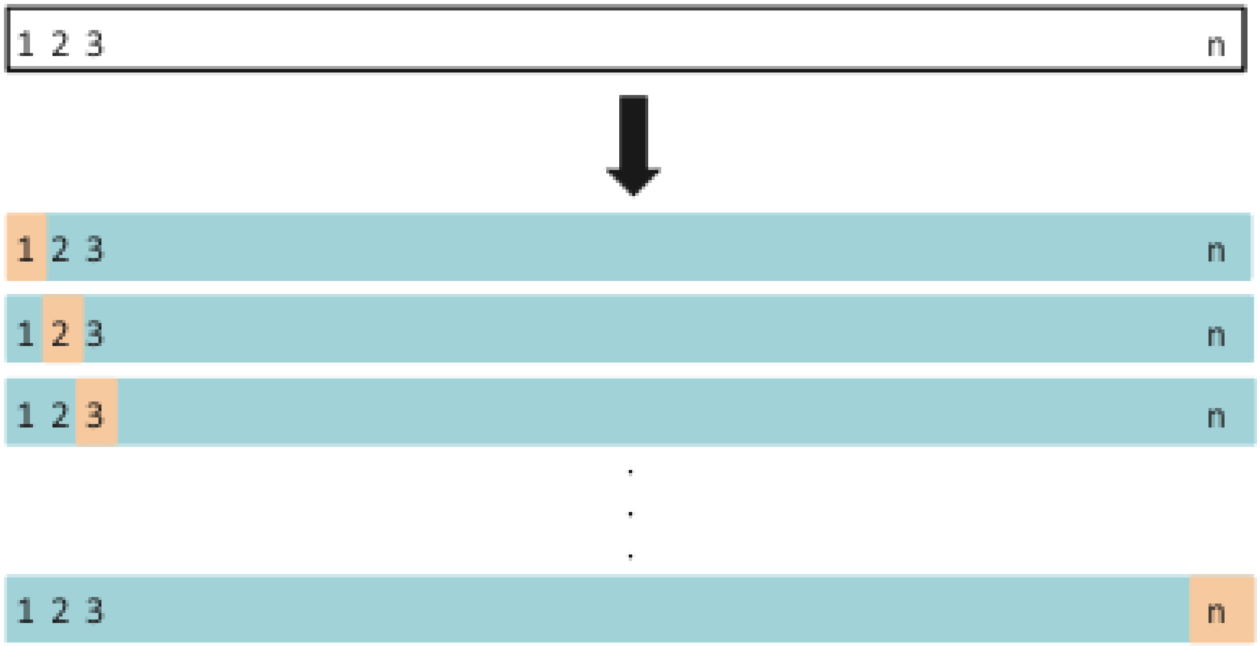


$K = 7$

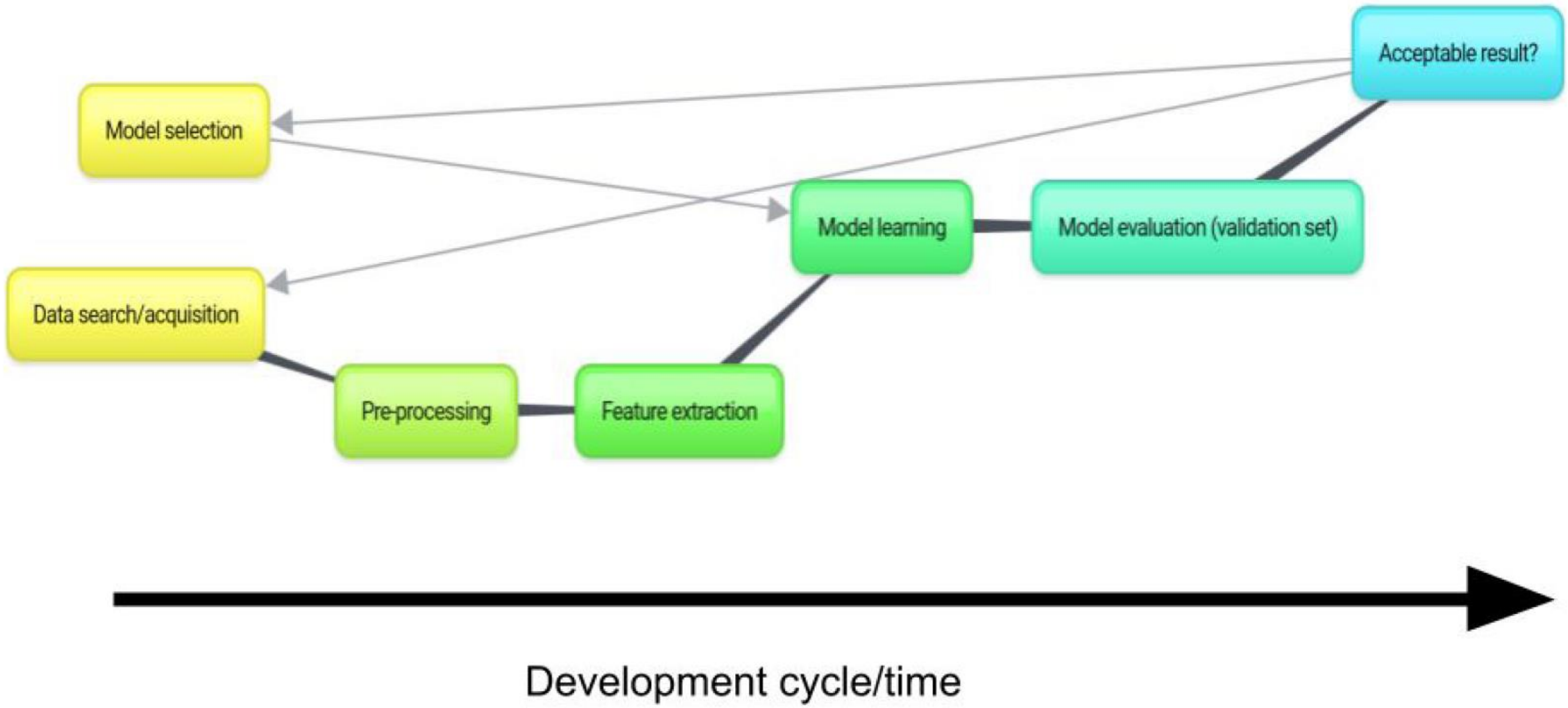
k-Fold cross validation



Leave-one-out cross validation



Development Cycle



전통적인 접근법

<table>
<tr><td colspan="5" rowspan="4"></td><th colspan="6">Tasks</th></tr>
<tr><th colspan="5">ADAS</th><th></th></tr>
<tr><th colspan="3">Self Driving</th><th></th><th></th><th></th></tr>
<tr><th>Localization</th><th>Perception</th><th>Planning/ Control</th><th>Driver state</th><th>Vehicle Diagnosis</th><th>Smart factory</th></tr>
<tr><td rowspan="7">Methods</td><td rowspan="2">Traditional</td><td colspan="3">Non-machine Learning</td><td>GPS, SLAM</td><td></td><td>Optimal control</td><td></td><td></td><td></td></tr>
<tr><td rowspan="6">Machine-Learning based method</td><td rowspan="4">Supervised</td><td>MLP</td><td></td><td>Pedestrian detection (HOG+SVM)</td><td></td><td></td><td></td><td></td></tr>
<tr><td rowspan="5">Deep-Learning based</td><td>CNN</td><td></td><td>Detection/ Segmentation/Classification</td><td>End-to-end Learning</td><td></td><td></td><td></td></tr>
<tr><td>RNN (LSTM)</td><td></td><td>Dry/wet road classification</td><td>End-to-end Learning</td><td></td><td></td><td></td></tr>
<tr><td>DNN</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td colspan="2">Reinforcement</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td colspan="2">Unsupervised</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

Conventional approach

- Image classification

“Motocycle”

Slides from “Andrew Ng”

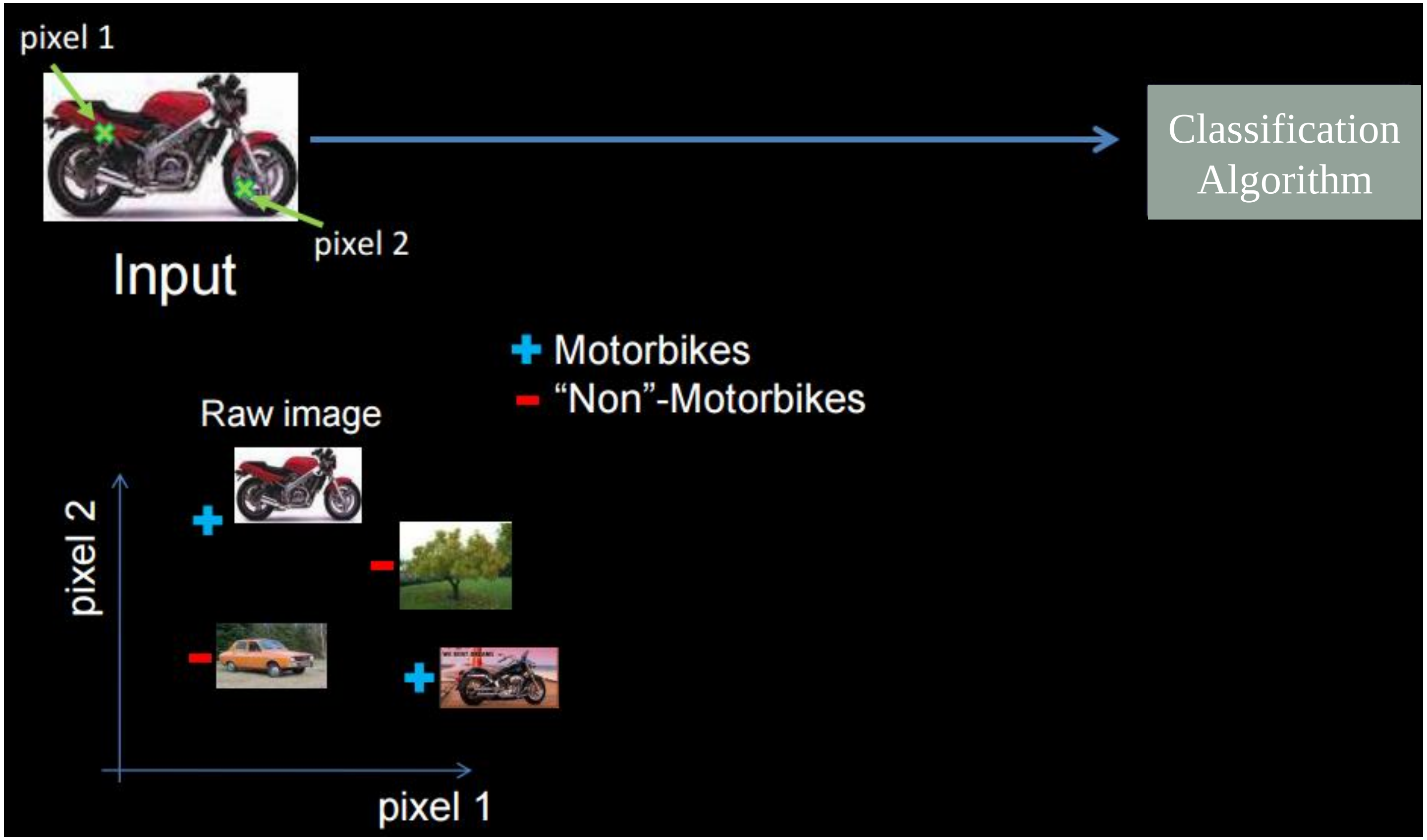
Why is this hard?

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

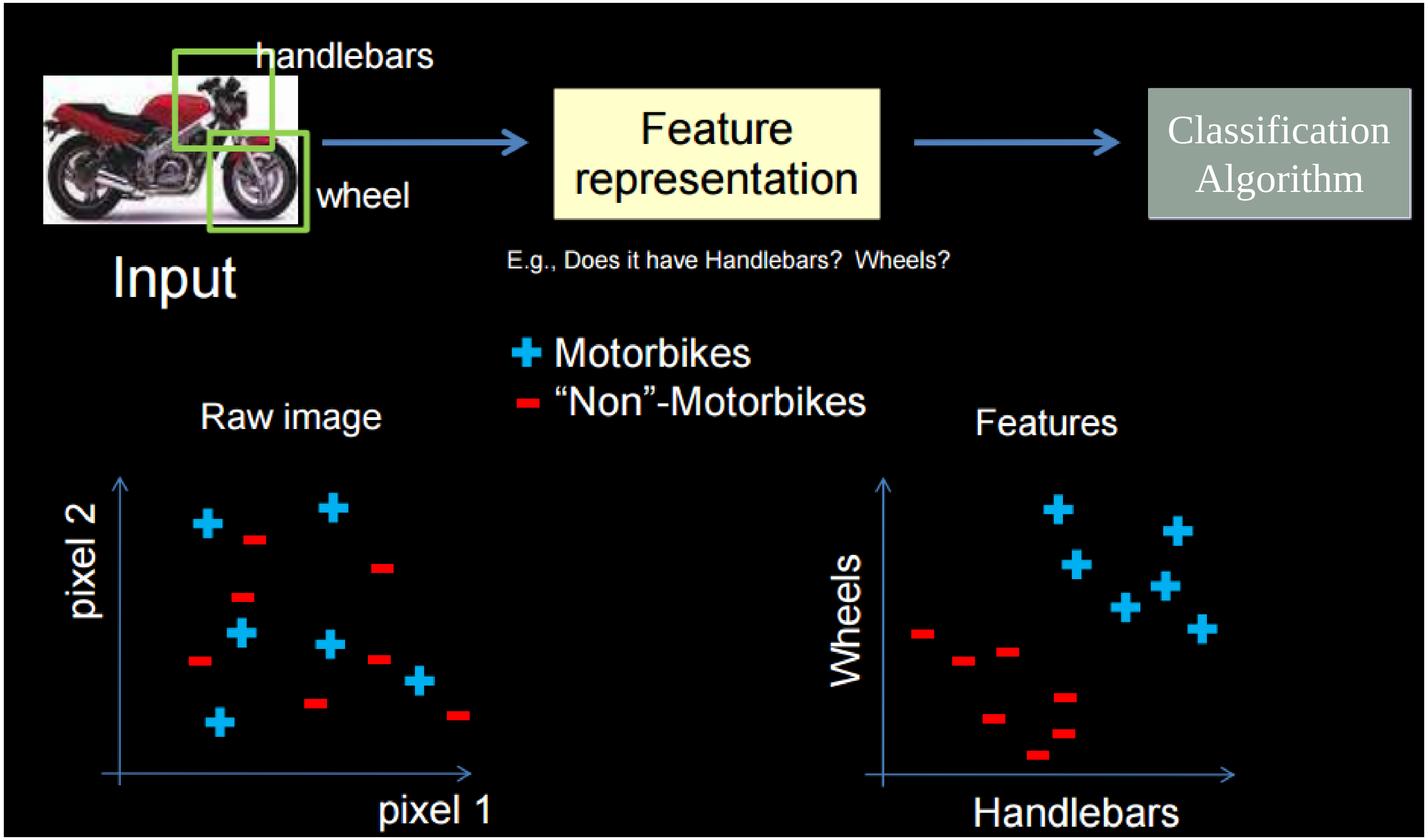
Slides from "Andrew Ng"

Feature representation



Slides from "Andrew Ng"

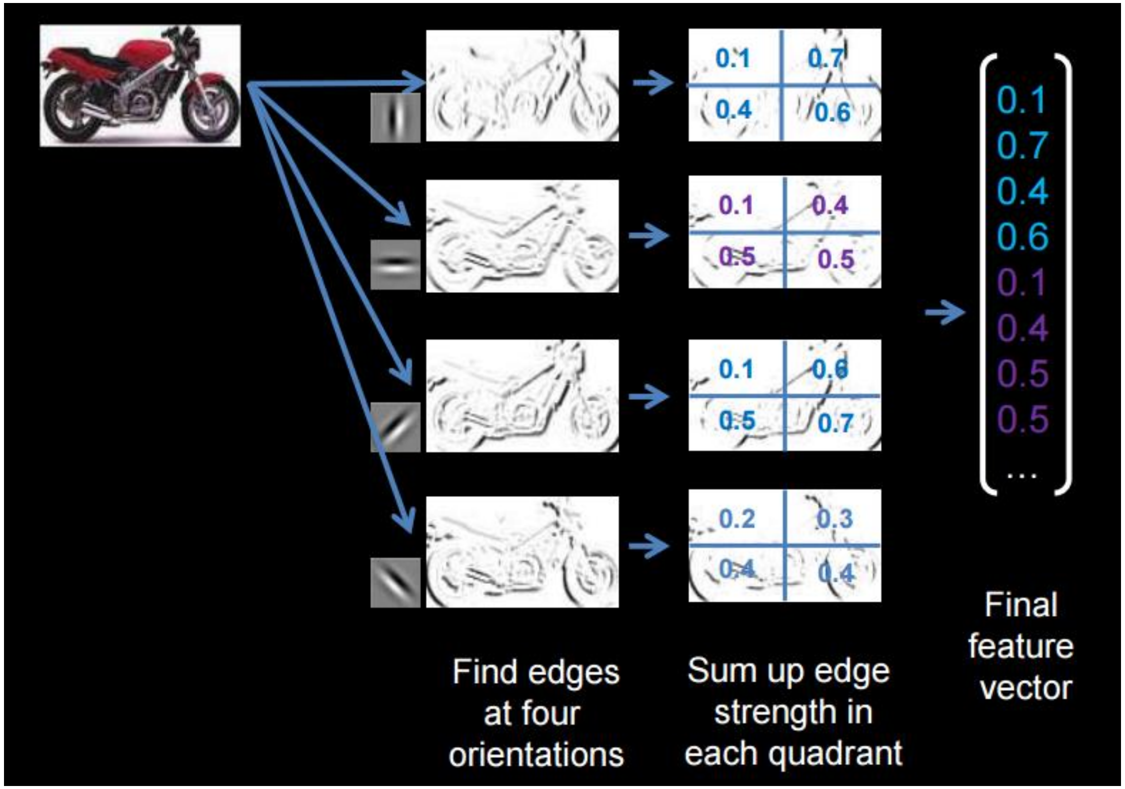
Feature representation



Slides from "Andrew Ng"

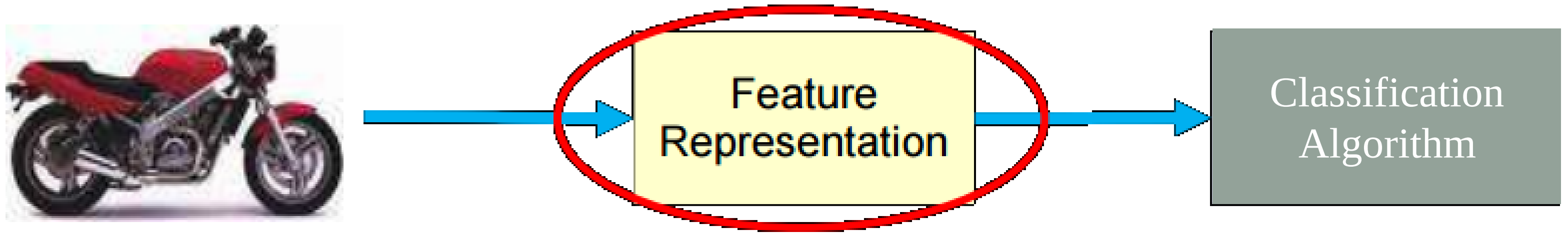
Example of Feature Representation

- But, … we don’t have a handlebars detector. So, researchers try to hand-design features to capture various statistical properties of the image



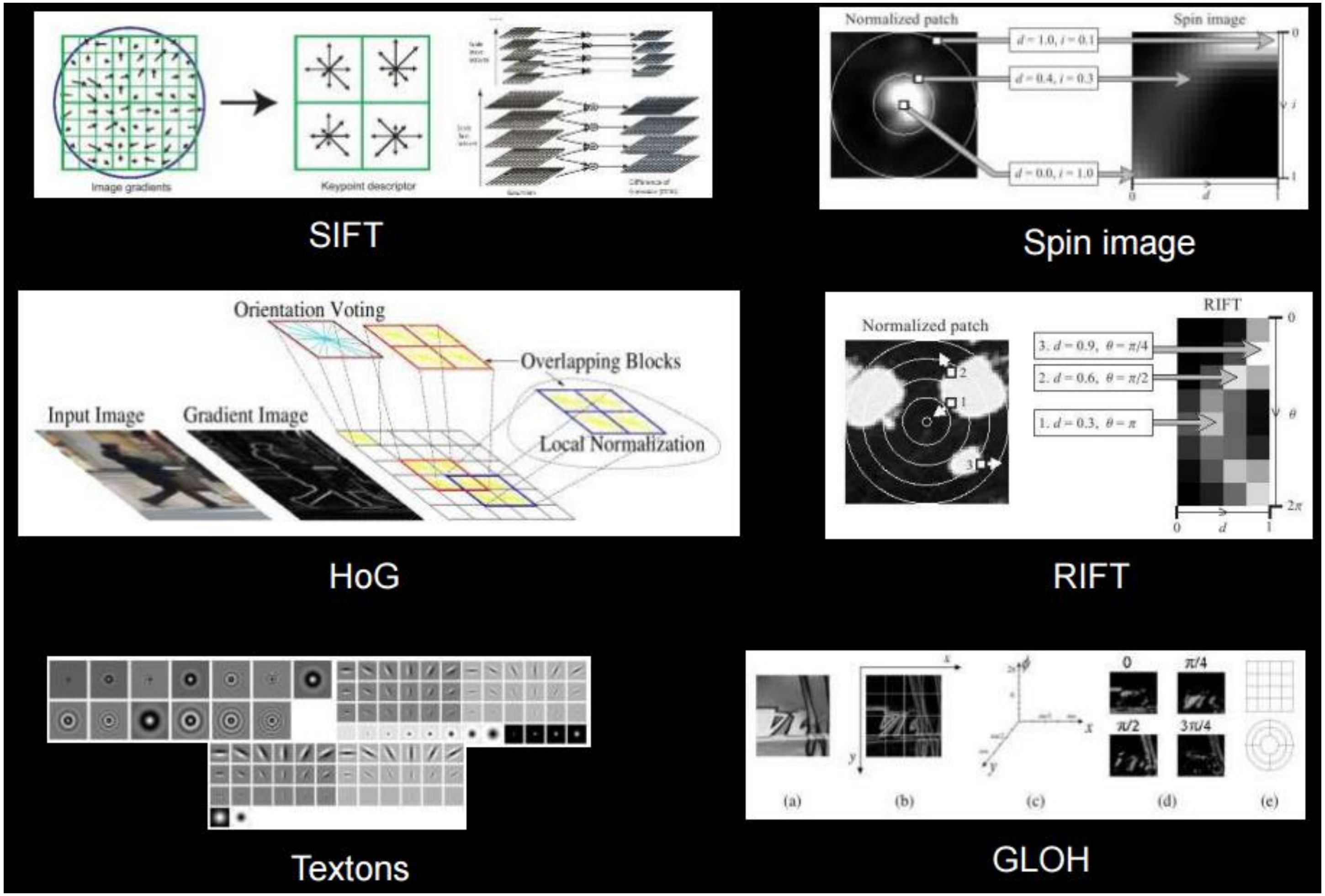
Slides from “Andrew Ng”

Feature representation



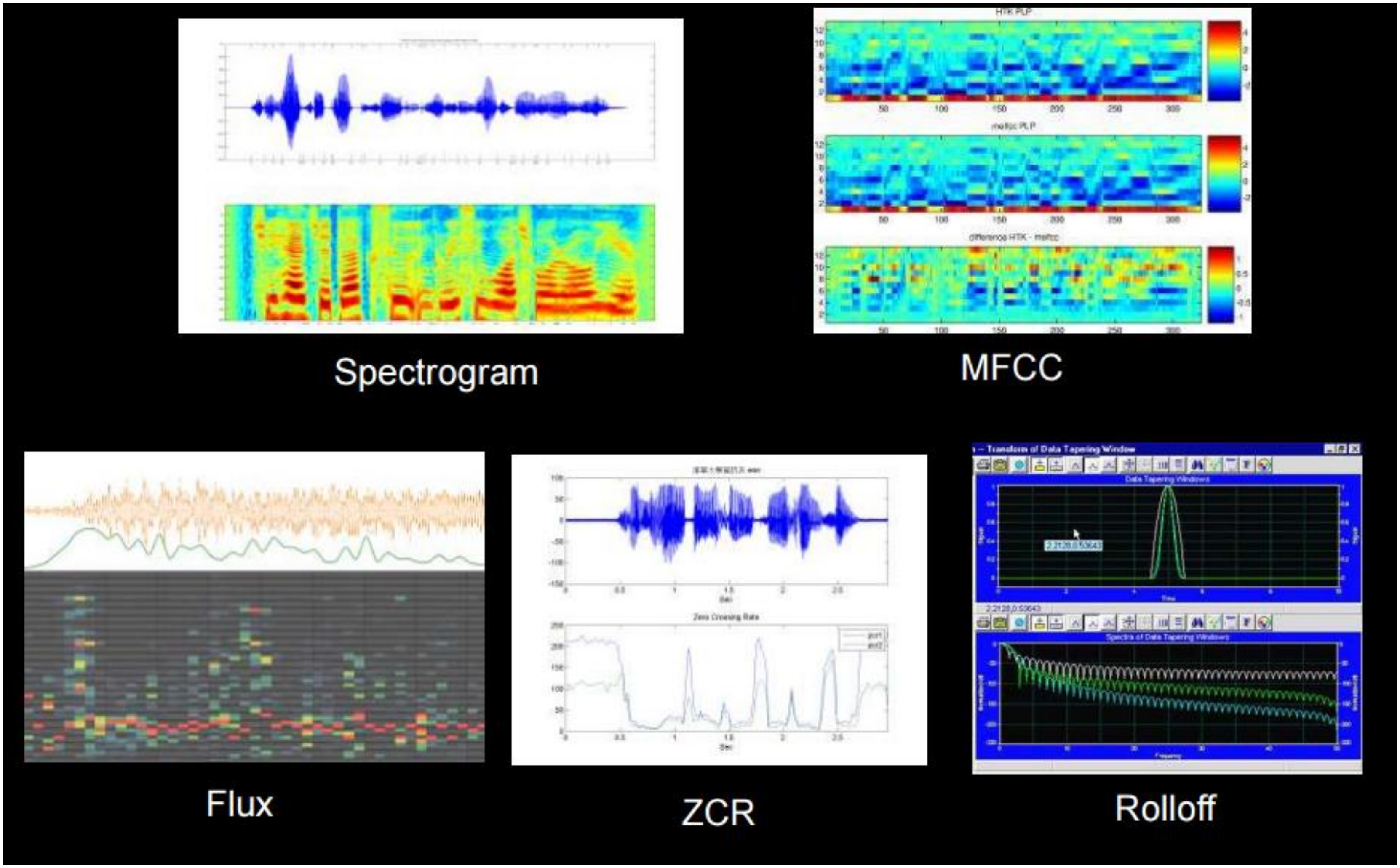
Slides from “Andrew Ng”

Computer vision features



Slides from “Andrew Ng”

Audio features



Traditional pattern recognition

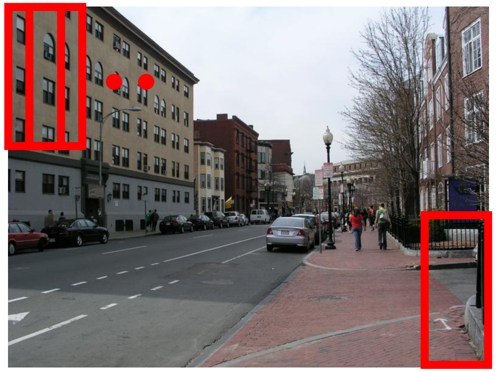
- Fixed/engineered feature + trainable classifier

CASE STUDY:
PEDESTRIAN DETECTOR

					Tasks					
					ADAS					
					Self Driving					
					Localization	Perception	Planning/ Control	Driver state	Vehicle Diagnosis	Smart factory
Methods	Traditional	Non-machine Learning			GPS, SLAM		Optimal control			
		Machine-Learning based method	Supervised	MLP		Pedestrian detection (HOG+SVM)				
	Deep-Learning based			CNN		Detection/ Segmentation/Classification	End-to-end Learning			
				RNN (LSTM)		Dry/wet road classification	End-to-end Learning			
				DNN						
			Reinforcement							
			Unsupervised							

Detection problem → (binary) classification problem

- Sliding window scheme



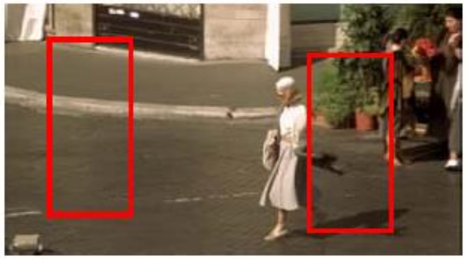
Each window is separately classified

Training data

- 64x128 images of humans cropped from a varied set of personal photos
  - Positive data – 1239 positive window examples (reflections->2478)




  - Negative data – 1218 person-free training photos (12180 patches)

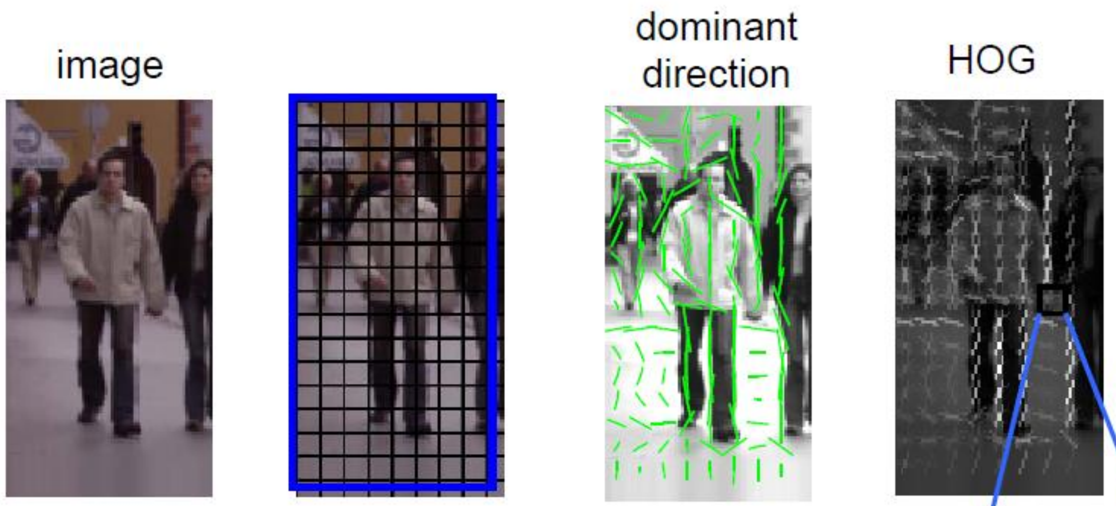



Training

- A preliminary detector
  - Trained with (2478) vs (12180) samples

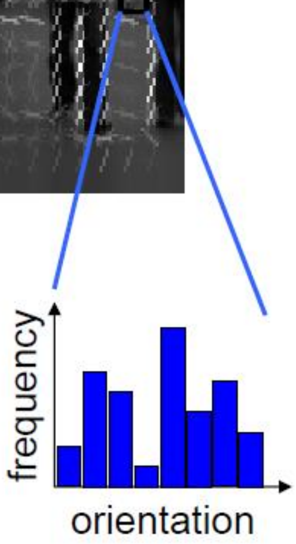
- Retraining
  - With augmented data set
    - initial 12180 + hard examples
  - Hard examples
    - 1218 negative training photos are searched exhaustively for false positive

Feature: histogram of oriented gradients (HOG)

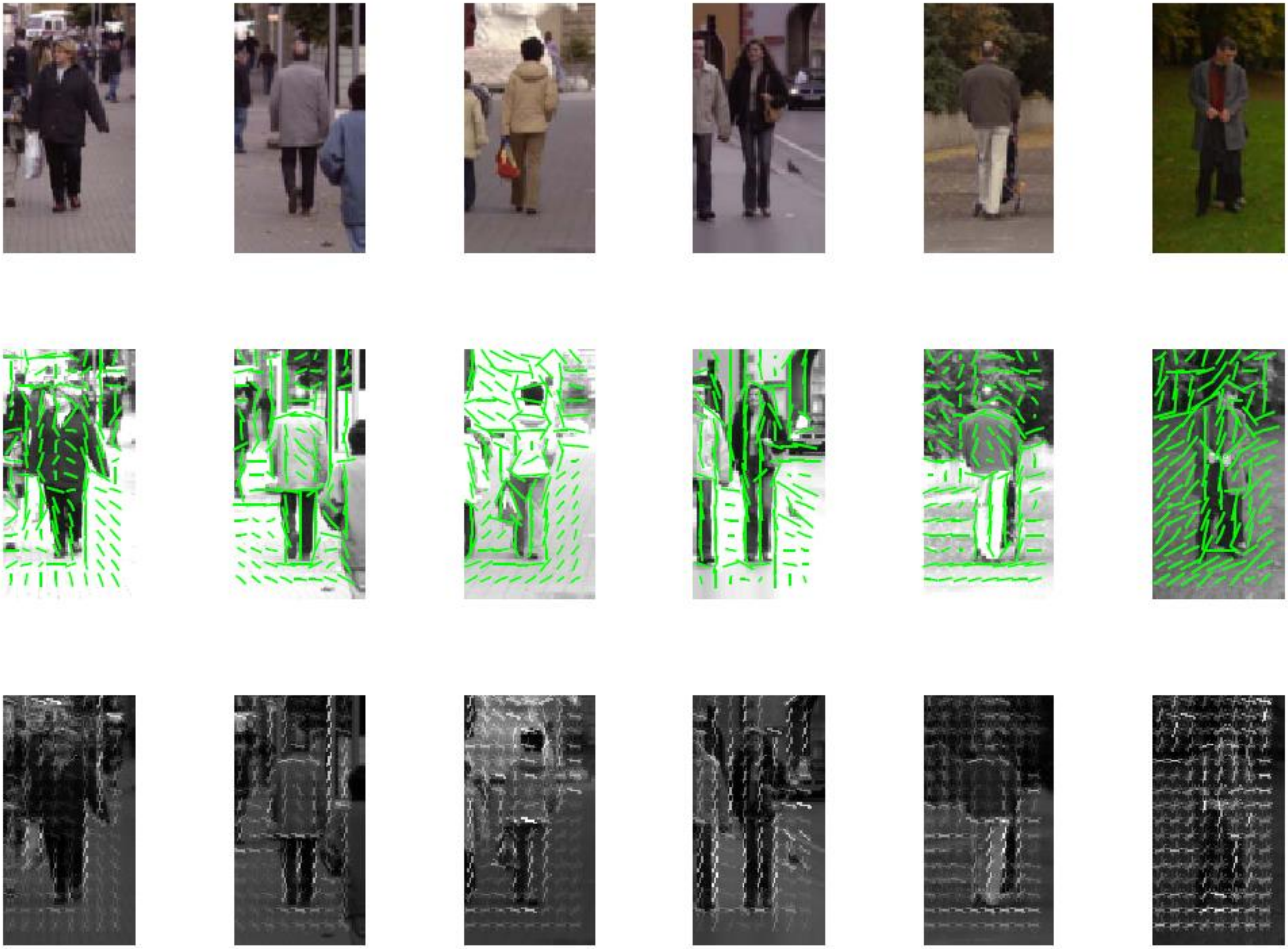


• tile window into 8 x 8 pixel cells

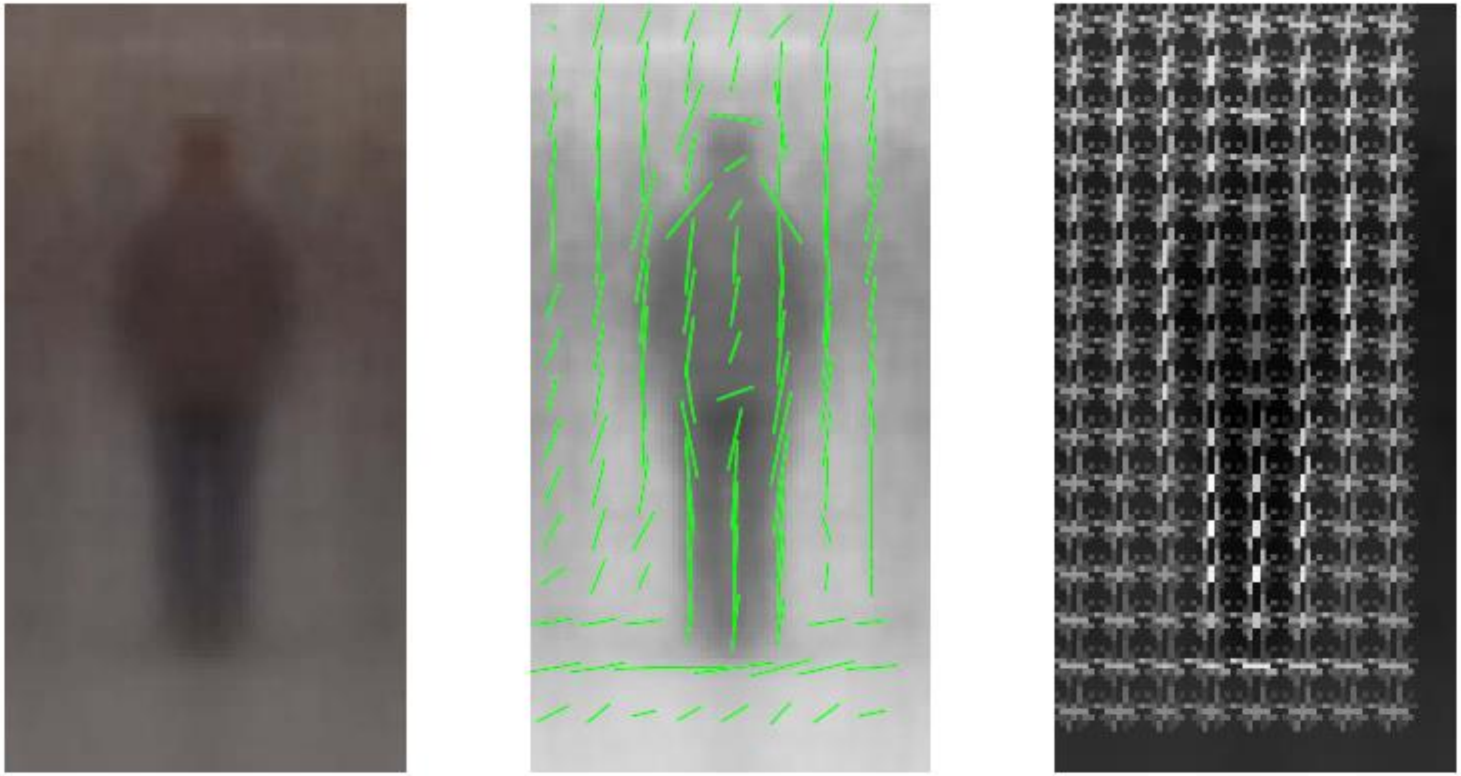
• each cell represented by HOG



Feature vector dimension = 16 x 8 (for tiling) x 8 (orientations) = 1024



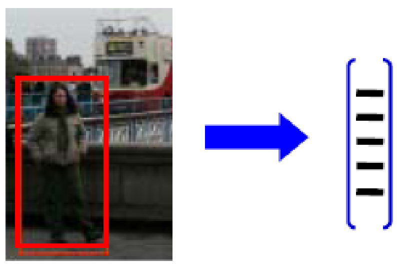
Averaged examples



Algorithm

Training (Learning)

- Represent each example window by a HOG feature vector



$$\mathbf{x}_i \in \mathbb{R}^d, \ \text{with } d = 1024$$

- Train a SVM classifier

Testing (Detection)

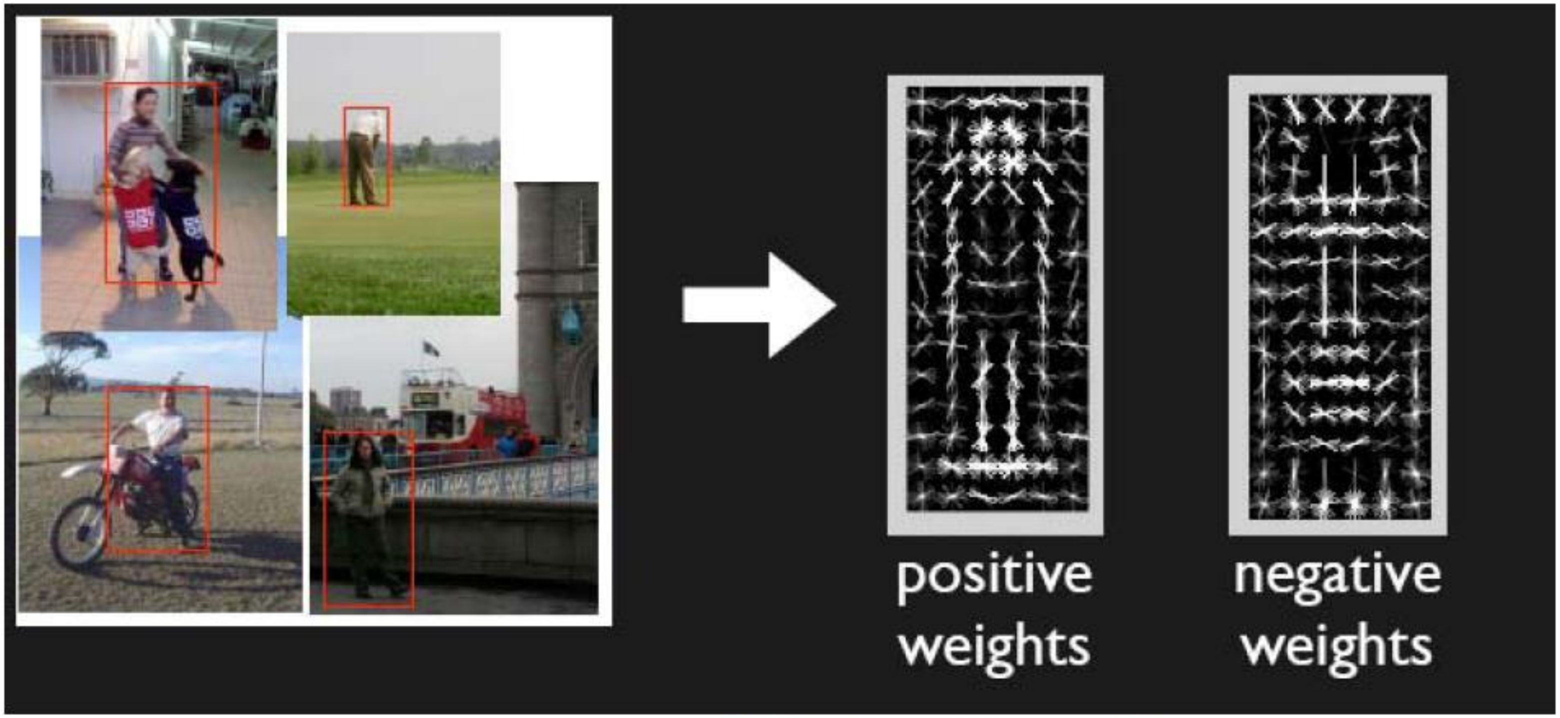
- Sliding window classifier

$$f(x) = \mathbf{w}^\top \mathbf{x} + b$$

Dalal, Navneet, and Bill Triggs. "Histograms of oriented gradients for human detection." *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 1. IEEE, 2005.

Learned model

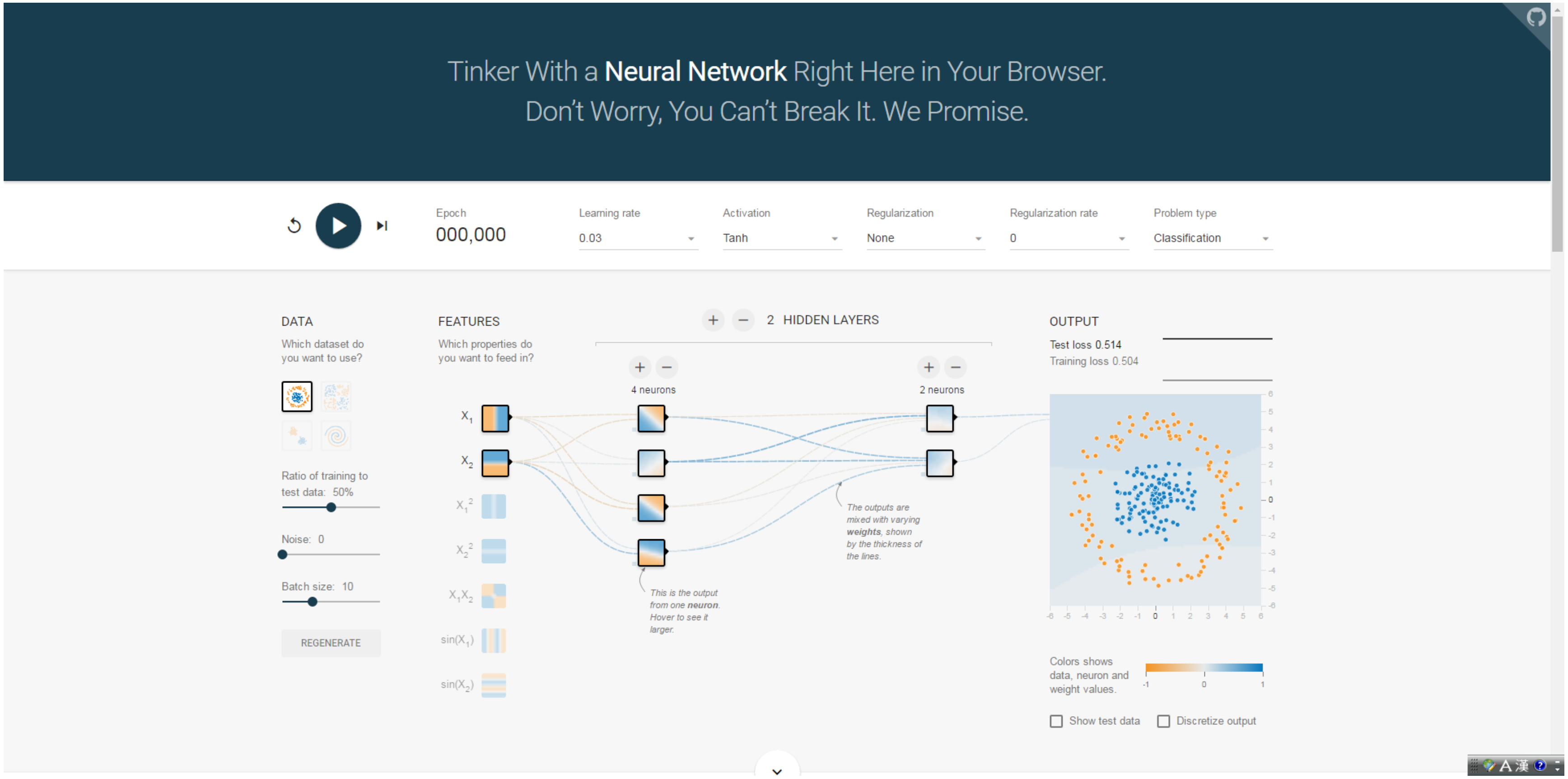
$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$$



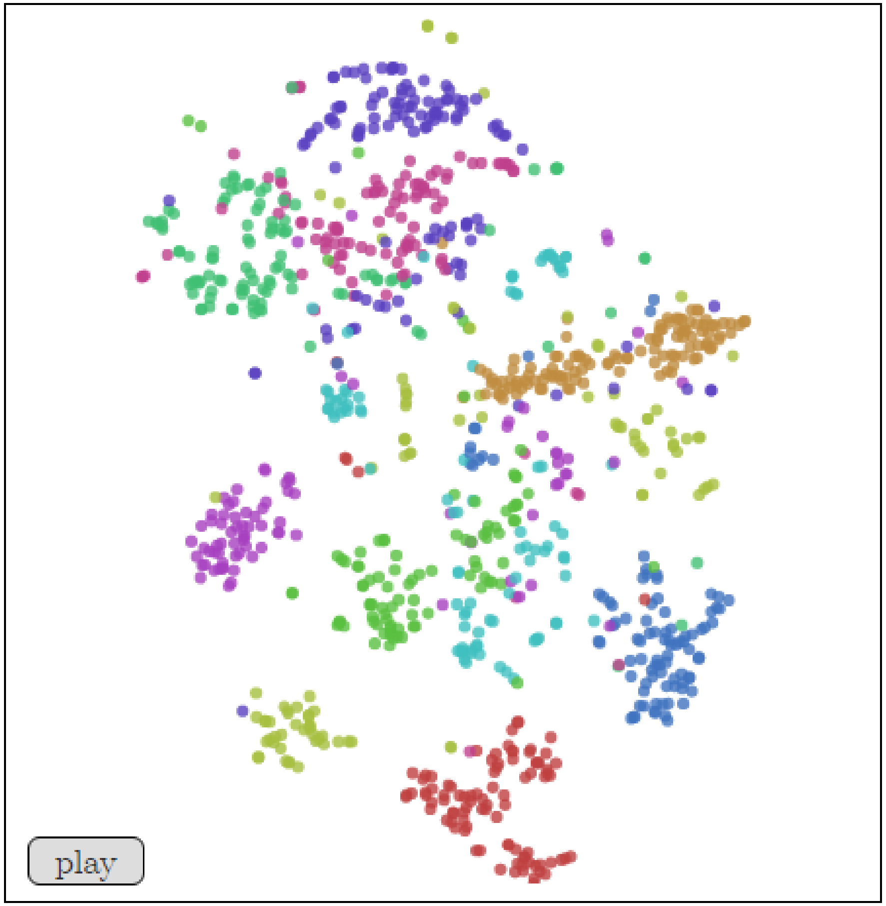
Slide from Deva Ramanan

BACKUPS

http://playground.tensorflow.org/



Visualization MNIST with t-SNE

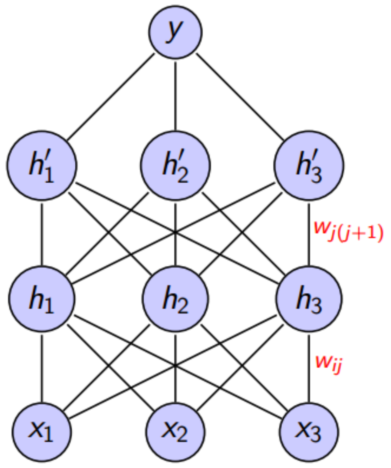


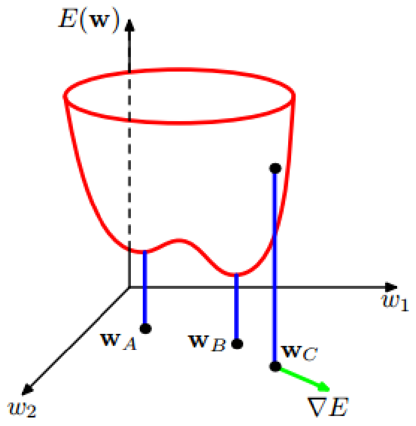
Visualizing MNIST with t-SNE

Why are Deep Architectures hard to train?

- Vanishing gradient problem in back-propagation.

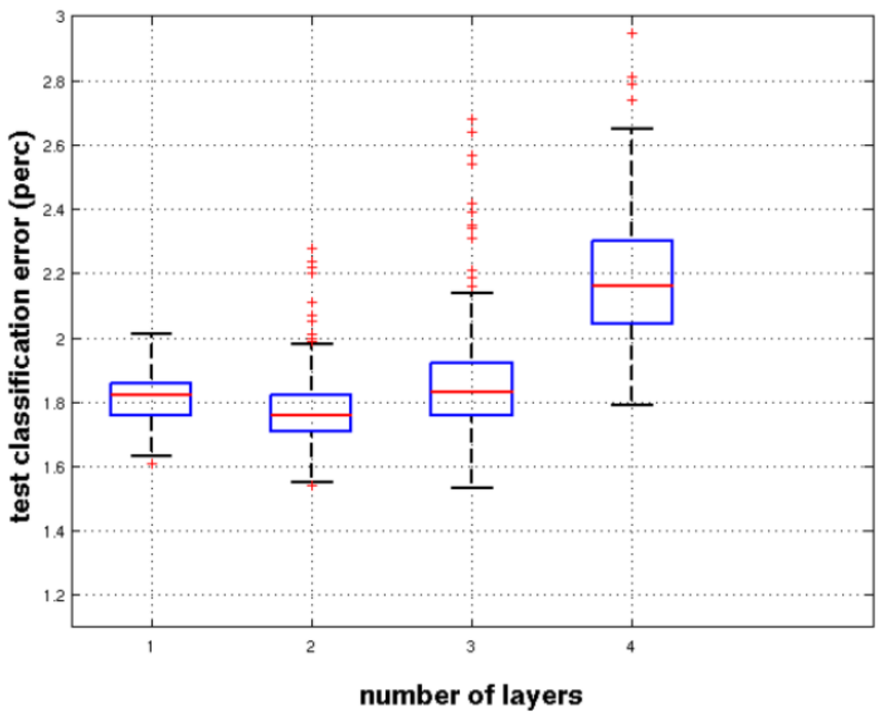
- Local Optimum (saddle points?) Issue in Neural Nets
  - For Deep Architectures, back-propagation is apparently getting a local optimum (saddle points?) that does not generalize well





Empirical Results: Poor performance of Backpropagation on Deep Neural Nets [Erhan et al., 2009]

- MNIST digit classification task; 400 trials (random seed)
- Each layer: initialize weights with random numbers
- Although $L+1$ layers is more expressive, worse error than L layers.



AI Winters

Two major episodes:

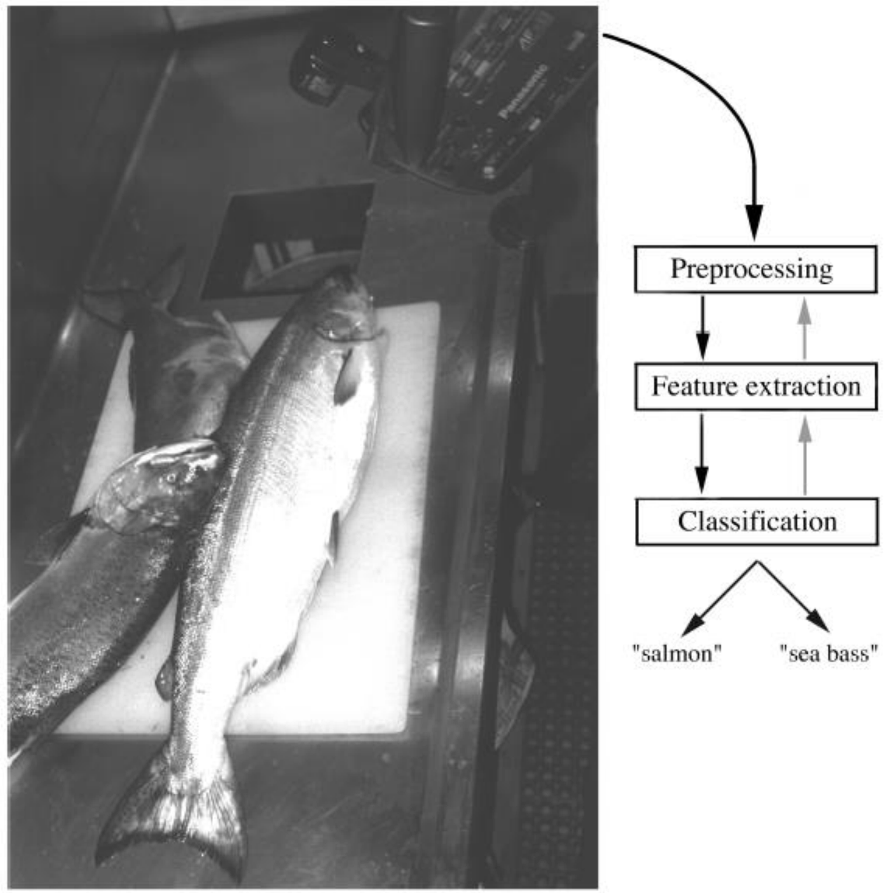
- 1974-80
- 1987-93

Smaller episodes:

- 1966: the failure of machine translation
- 1970: the abandonment of connectionism
- 1971-75: DARPA's frustration with the Speech Understanding Research program
- 1973: the large decrease in AI research in the UK in response to the Lighthill report.
- 1973–74: DARPA's cutbacks to academic AI research in general
- 1987: the collapse of the Lisp machine market
- 1988: the cancellation of new spending on AI by the Strategic Computing Initiative
- 1993: expert systems slowly reaching the bottom
- 1990s: the quiet disappearance of the fifth-generation computer project's original goals.

"In no part of the field have discoveries made so far produced the major impact that was then promised."

An example


Preprocessing
Feature extraction
Classification
"salmon"
"sea bass"

Feature extraction

Width feature

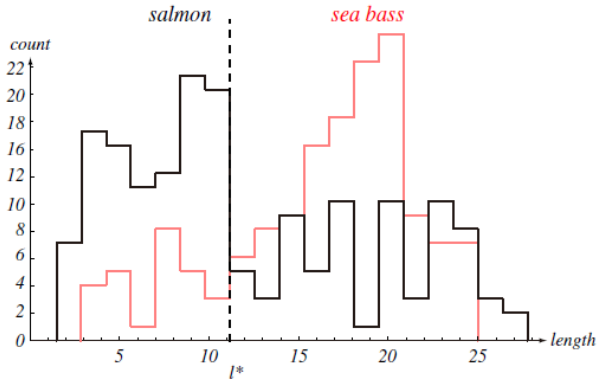


FIGURE 1.2. Histograms for the length feature for the two categories. No single threshold value of the length will serve to unambiguously discriminate between the two categories; using length alone, we will have some errors. The value marked l^* will lead to the smallest number of errors, on average. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Lightness feature

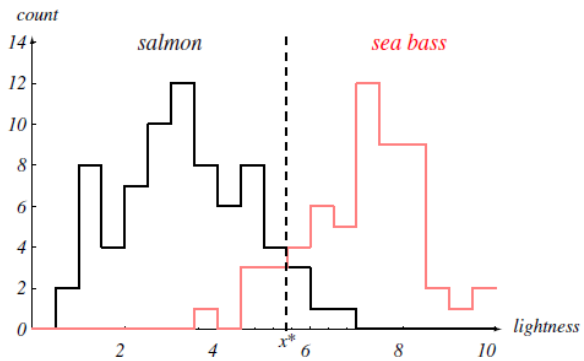
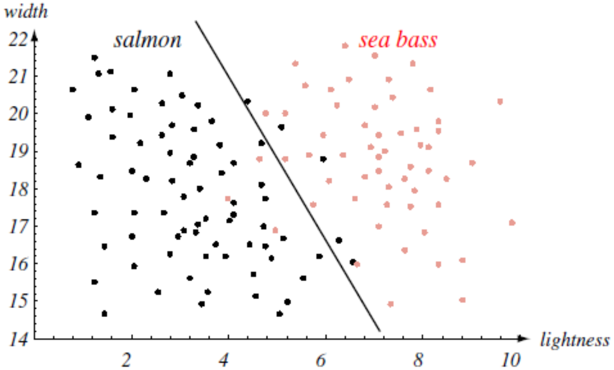


FIGURE 1.3. Histograms for the lightness feature for the two categories. No single threshold value x^* (decision boundary) will serve to unambiguously discriminate between the two categories; using lightness alone, we will have some errors. The value x^* marked will lead to the smallest number of errors, on average. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Classification

Simple model



Complex model