max planck institut informatik

SIC Saarland Informatics Campus

High Level Computer Vision - Introduction @ April 9, 2025

Bernt Schiele

https://cms.sic.saarland/hlcvss25/

Max Planck Institute for Informatics & Saarland University,
Saarland Informatics Campus Saarbrücken

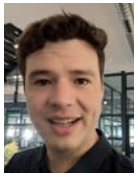
max planck institut informatik

SIC Saarland Informatics Campus

Computer Vision and Machine Learning Group @ Max-Planck-Institute for Informatics

Bernt Schiele
Computer Vision &
Machine Learning

Jonas Fischer
Explainable Machine
Learning



Jan Eric Lenssen
Geometric Representation
Learning

Margret Keuper
Robust Visual Learning
U Mannheim

High Level Computer Vision

- Lecturer:
  - Bernt Schiele (schiele@mpi-inf.mpg.de)

- Assistant:
  - Amin Parchami-Araghi (mparcham@mpi-inf.mpg.de)
  - Nhi Pham (nhipham@mpi-inf.mpg.de)
  - Sophia Wiedmann
  - Mostafa Abdelgawad

- Language:
  - English

- mailing list for announcements etc.
  - register online: https://cms.sic.saarland/hlcvss25/

Lecture & Tutorial

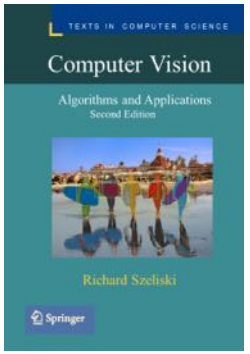
- Officially: 2V (lecture) + 2Ü (exercise/tutorial) - SWS 002
  - Lecture: Wed: 10:15am - 12pm
  - Tutorial: Mon: 10:15am - 12pm

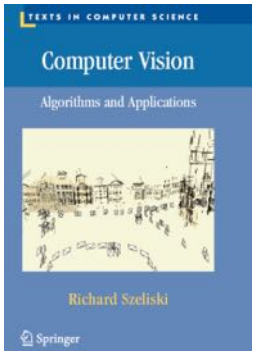
- typically 1 exercise sheet every 1-2 weeks
  - working in **groups of 2 students**
  - some pencil and paper, mostly practical including a larger project
  - larger project: we/you propose projects, mentoring, final presentation
  - Mon, April 14th: Python tutorial

- Exam
  - written exam
  - exercise & project participation **mandatory** to register for exam
  - up to 1 **grade improvement** (50% project, 50% exercises)

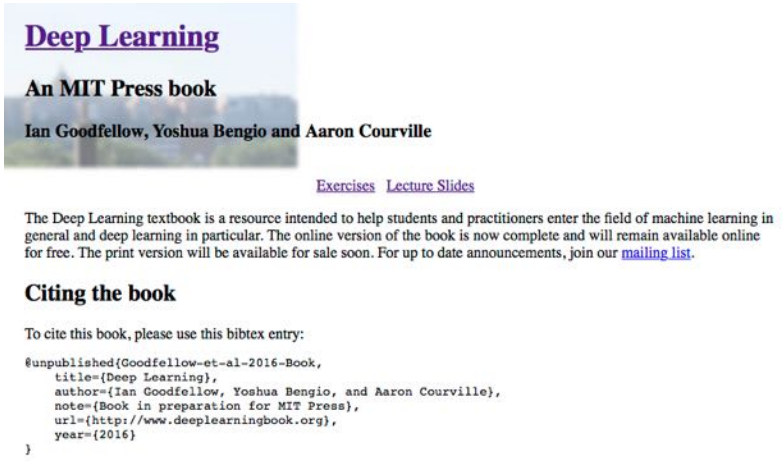
Material

- For "non-deep-learning" parts of the lecture:
  - available online http://szeliski.org/Book





- Background on deep learning: Deep Learning Book
  - available online http://deeplearningbook.org



Deep Learning

An MIT Press book

Ian Goodfellow, Yoshua Bengio and Aaron Courville

Exercises Lecture Slides

The Deep Learning textbook is a resource intended to help students and practitioners enter the field of machine learning in general and deep learning in particular. The online version of the book is now complete and will remain available online for free. The print version will be available for sale soon. For up to date announcements, join our mailing list.

Citing the book

To cite this book, please use this bibtex entry:

```
@unpublished{Goodfellow-et-al-2016-Book,
    title={Deep Learning},
    author={Ian Goodfellow, Yoshua Bengio, and Aaron Courville},
    note={Book in preparation for MIT Press},
    url={http://www.deeplearningbook.org},
    year={2016}
}
```

Overview Lecture (preliminary)

- Convolutional Neural Networks (CNNs) for High-Level Computer Vision Tasks:
  - Image Classification, Object Detection (Recognition, 2D Localization), Semantic Segmentation
  - Data preprocessing & Batch normalization, CNN architectures
- Transformer Architectures for High-Level Vision Tasks
  - Vision Transformers for image classification,
  - DETR for object detection, Semantic Segmentation
- Generative Models
  - Generative Adversarial Networks (GANs), Variational Auto-Encoders (VAEs), (Stable) Diffusion, …
- Self-Supervised Learning
  - How to learn with less or even no supervision
- Foundational Models
  - Vision-Language Models, CLIP, Segment Anything Model (SAM), …
- Recent Trends and Works
  - B-Cos Networks (inherently interpretable), Flamingo, ConvNeXt, …

Goals of today's lecture

- First intuitions about
  - What is computer vision?
  - What does it mean to see and how do we (as humans) do it?
  - How can we make this computational?

- Applications & Appetizers

- Case Study
  - Object Recognition — intuition from human vision…

- Image Classification
  - using data-driven approaches (machine learning)
  - K-nearest neighbor classifier

Why Study Computer Vision

- Science — How do Humans “See”
  - Foundations of perception. How do WE as humans see?
  - computer vision to explore “computational model of human vision”

- Engineering — How can We Make Machines “See”
  - How do we build systems that perceive the world
  - computer vision to solve real-world problems
    (e.g. self-driving cars to detect pedestrians)

- Applications
  - medical imaging (computer vision to support medical diagnosis, visualization)
  - car-industry (lane-keeping, pre-crash intervention, …)
  - security & surveillance (to follow/track people at the airport, train-station, ...)
  - entertainment (vision-based interfaces for games)
  - graphics (image-based rendering, vision to support realistic graphics)
  - …

Some Applications

- License Plate Recognition
  - London Congestion Charge
  - http://en.wikipedia.org/wiki/London_congestion_charge
- Security & Surveillance
  - Face Recognition
  - Airport Security (People Tracking)
- Medical Imaging
  - (Semi-)automatic segmentation and measurements
- Autonomous Driving & Robotics
  - Waymo, Chrysler, etc.

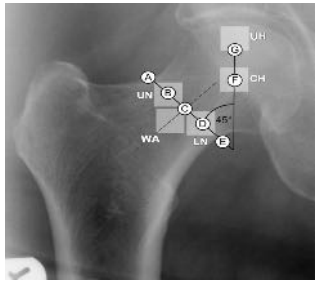
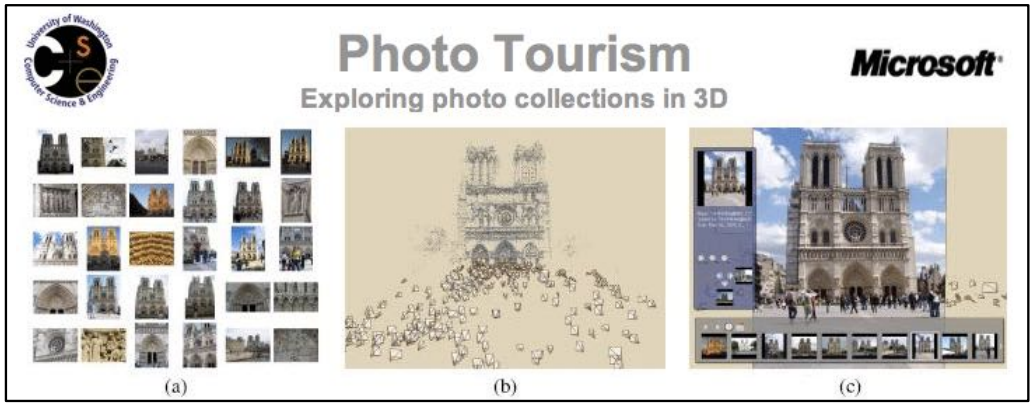


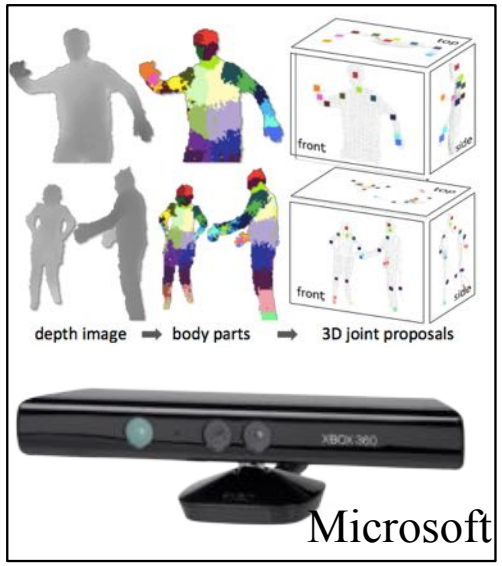
image: digital trends

More Applications

- Vision on Cellphones:
  - e.g. Google Lens, …
- Vision for Interfaces:
  - e.g. Microsoft Kinect
- Reconstruction

Google Lens



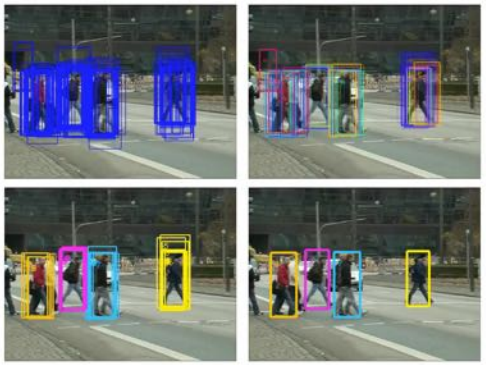


Applications & Appetizers

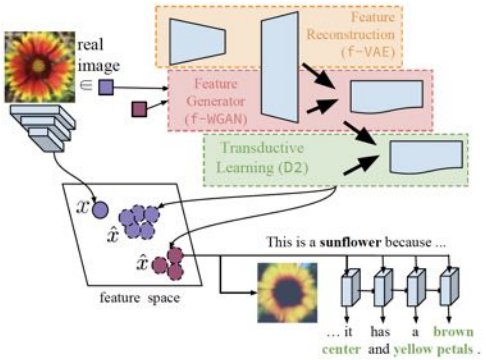
... work from our group

Some Research Highlights from the Past

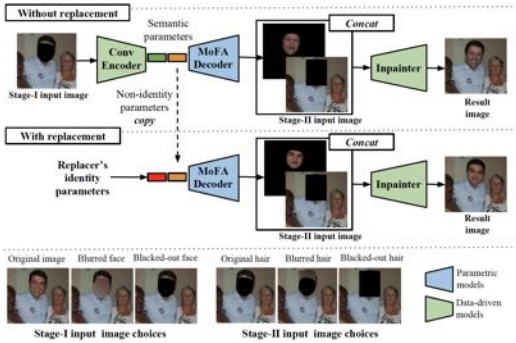
Multi-Person Tracking



Zero- and Few-Shot Learning



Visual Privacy



Multi-Person Human Pose Estimation & Tracking

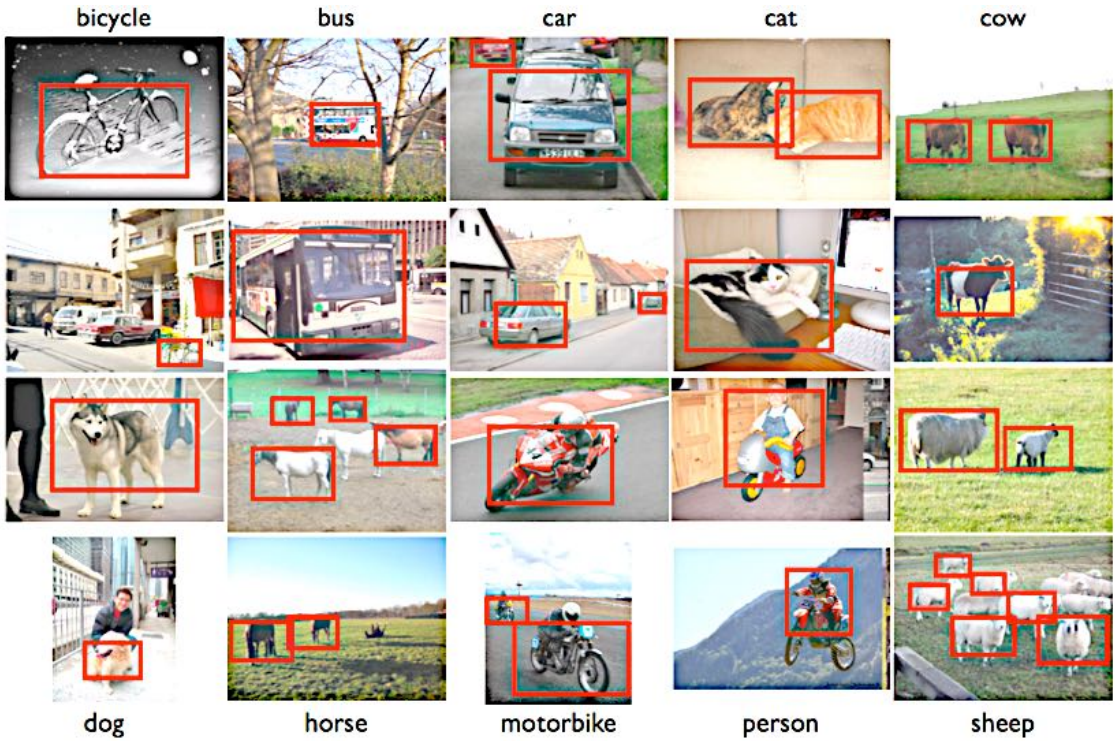
Video Segmentation

Datasets…

Caltech Pedestrians

CITYSCAPES DATASET

Detection & Recognition of Visual Categories

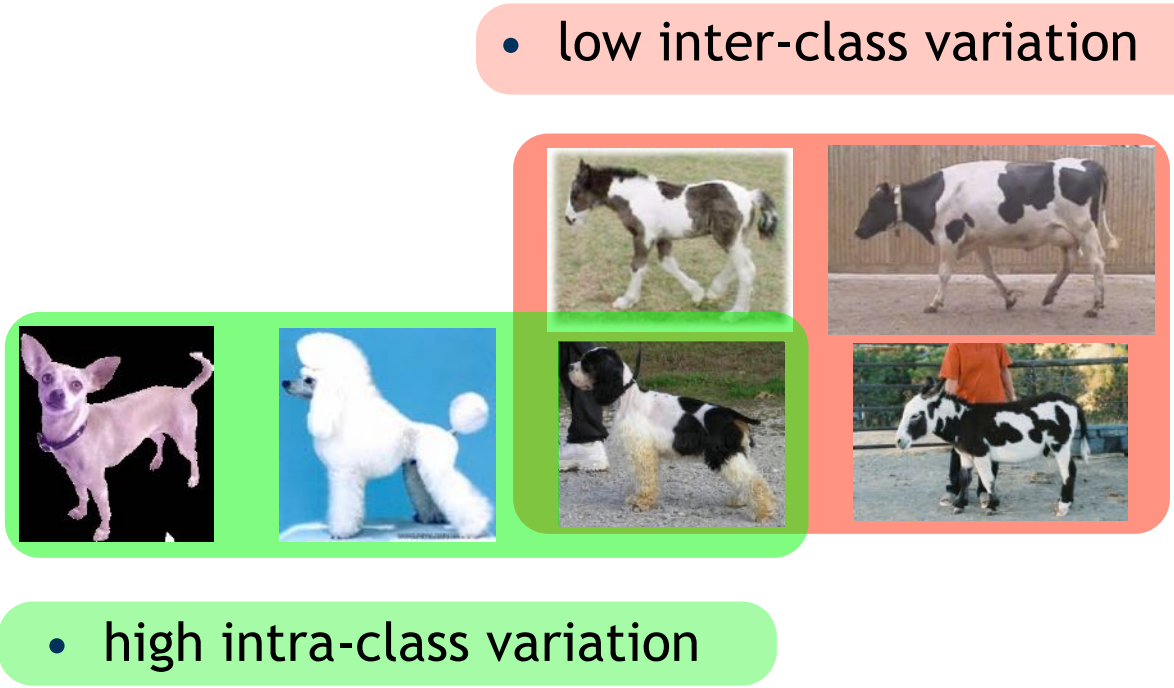


Challenges:

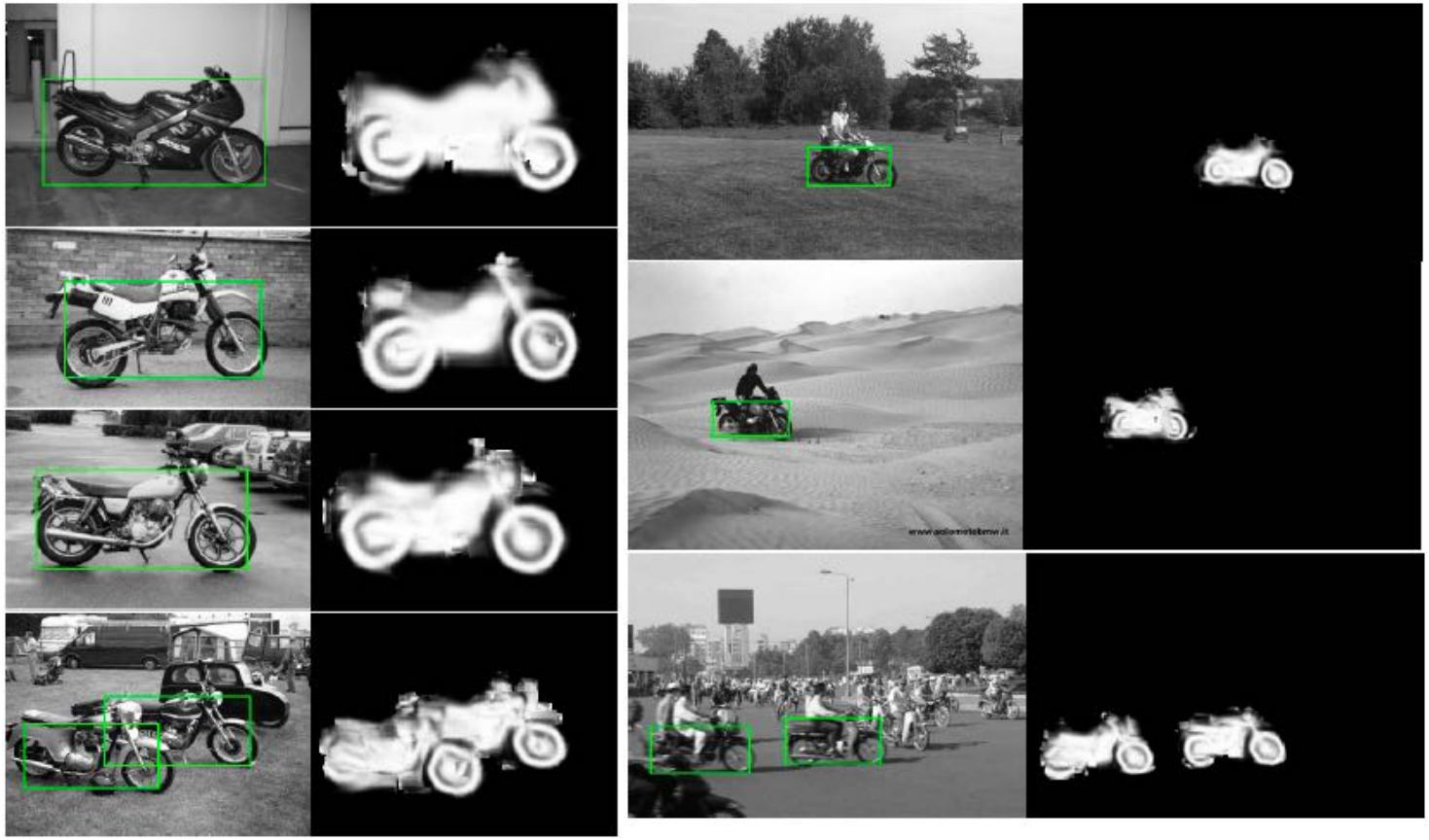
- multi-scale
- multi-view
- multi-class
- varying illumination
- occlusion
- cluttered background
- articulation
- high intraclass variance
- low interclass variance

Challenges of Visual Categorization

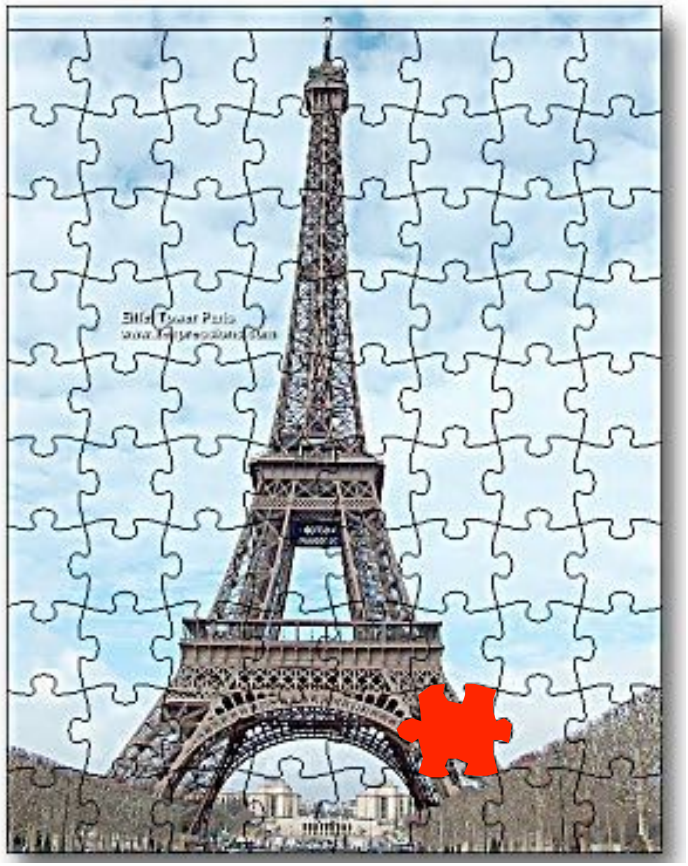
- high intra-class variation

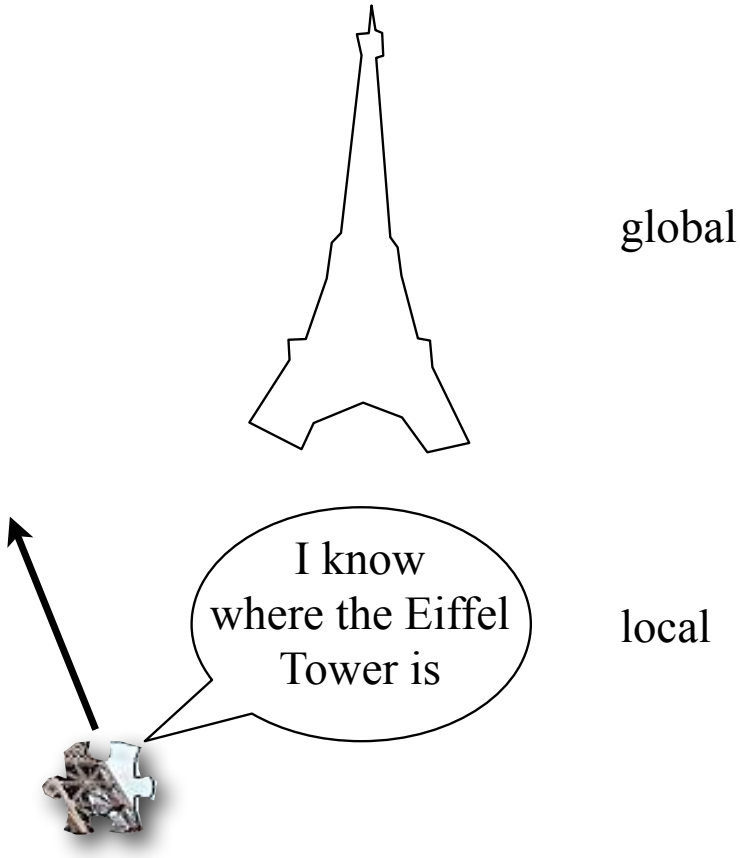


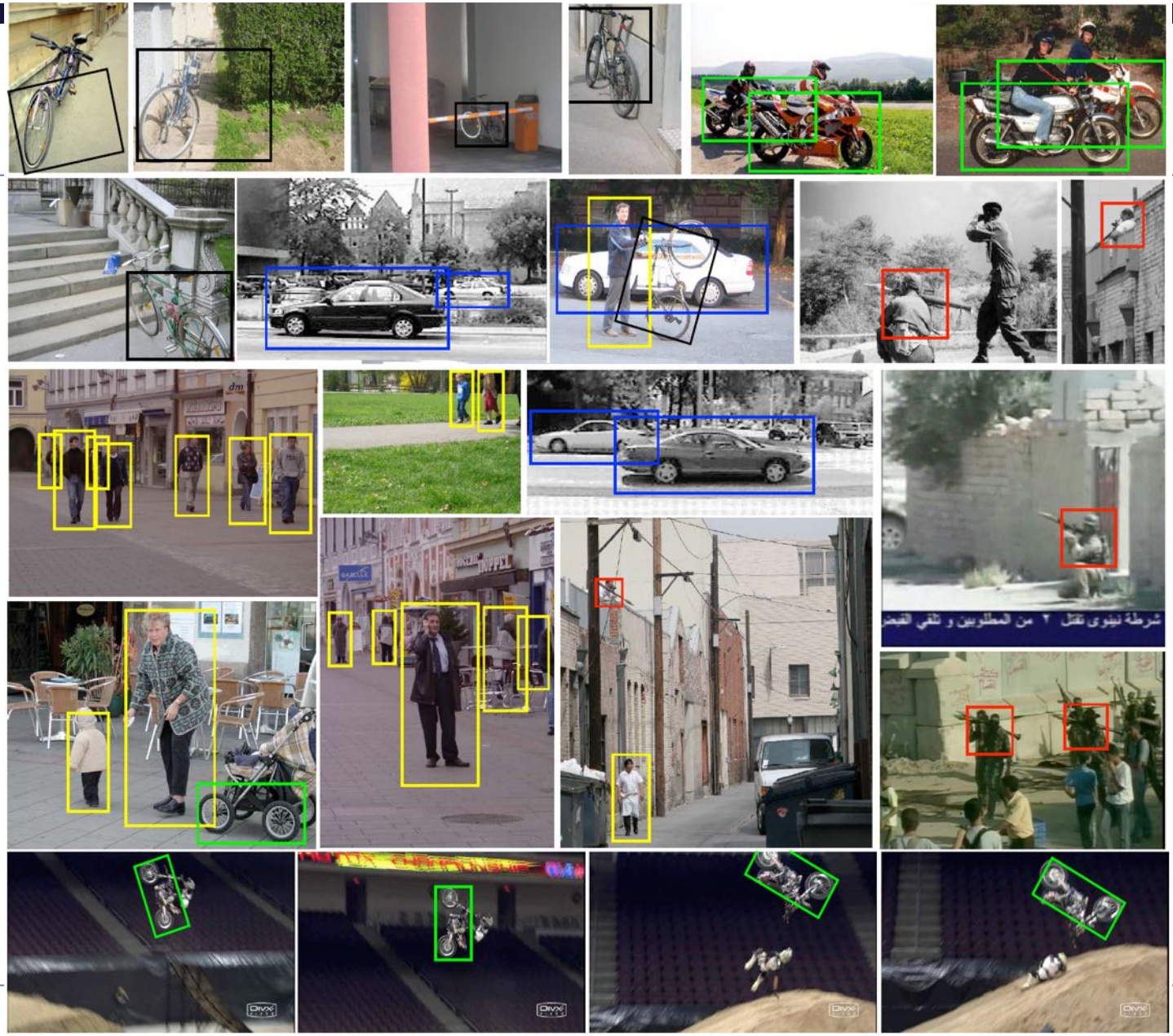
Sample Category: Motorbikes



Basic Idea



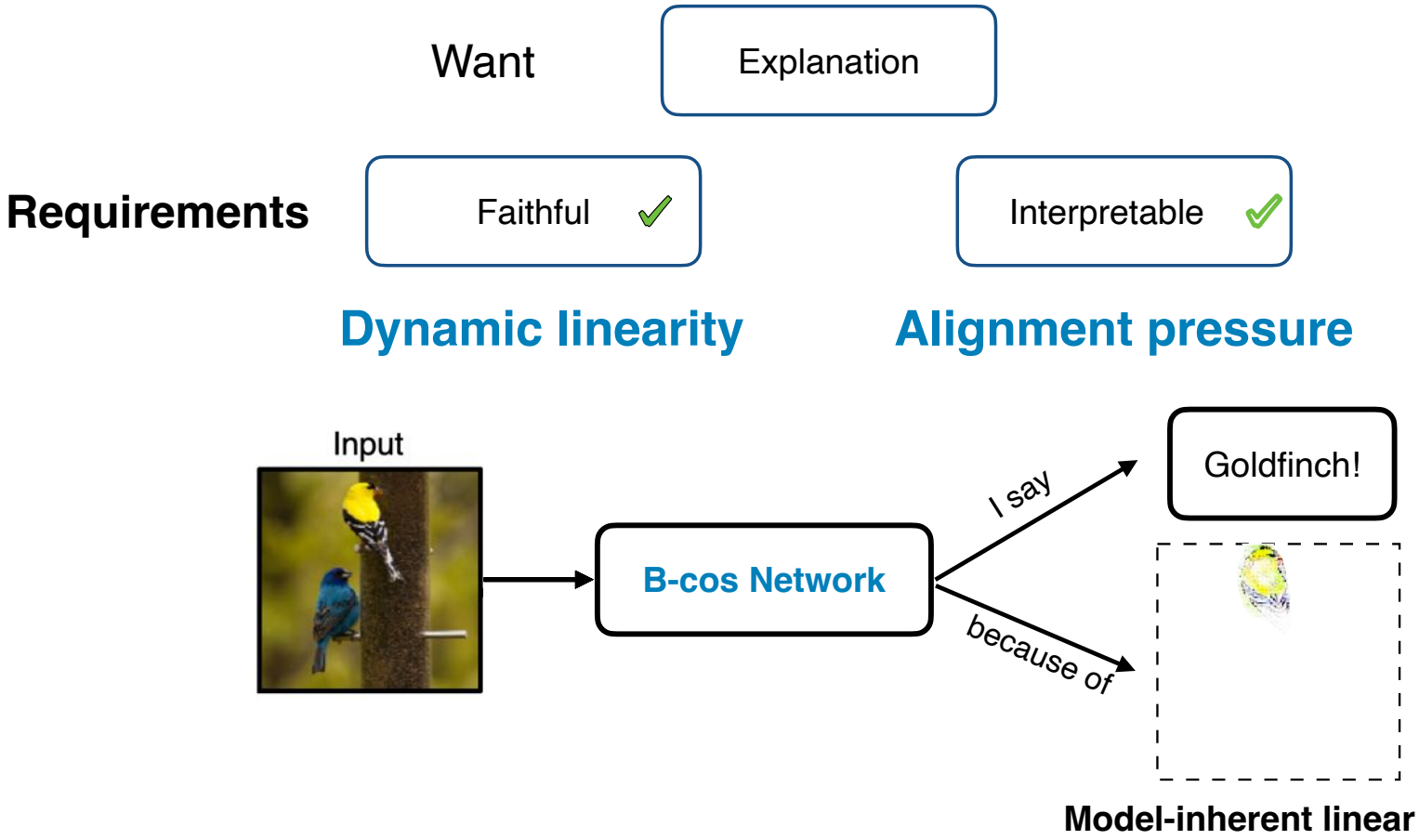




Computer Vision & Machine Learning: Some Research Themes

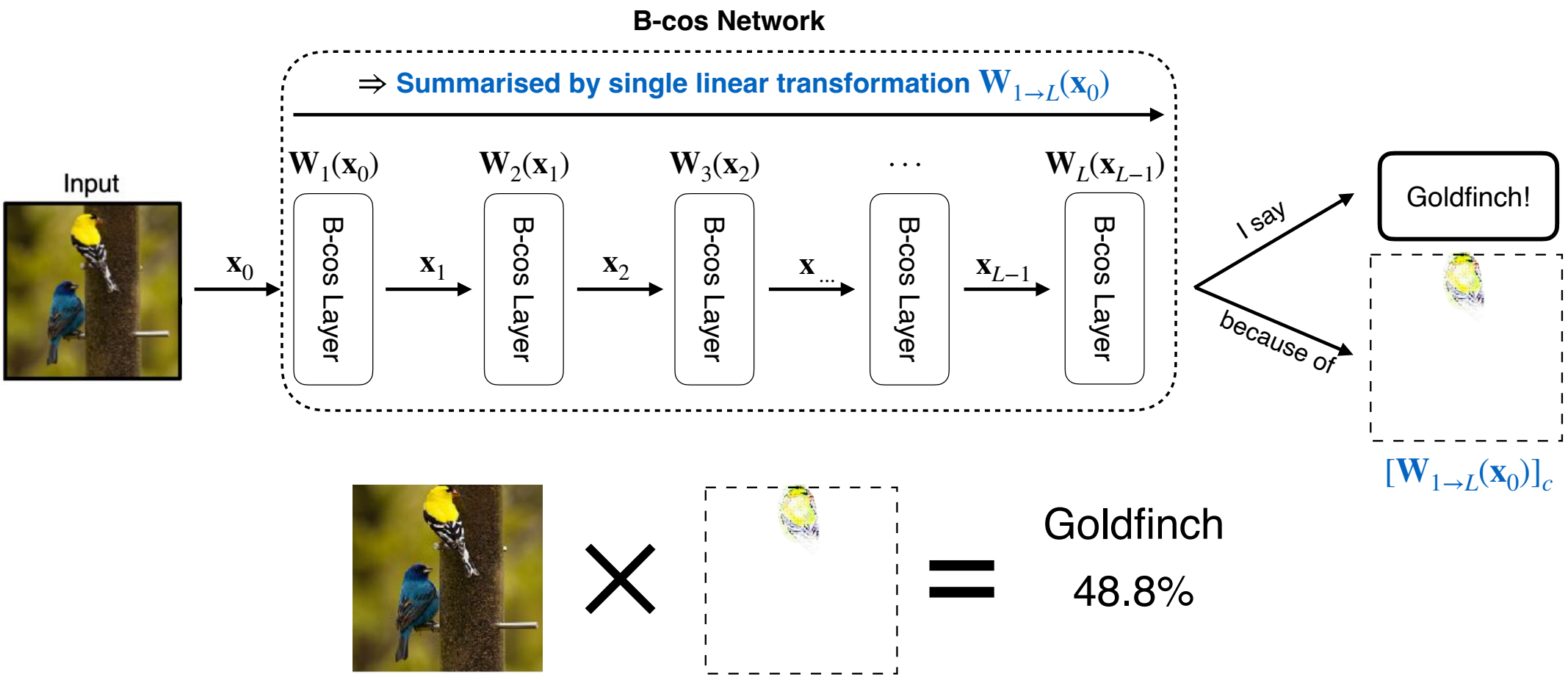
- **Interpretability, Robustness** and **Security** of Deep Learning in Computer Vision
  - **Inherently Interpretable** Deep Neural networks — CVPR'21, CVPR'22, PAMI'23, PAMI'24
  - **Robustness** of Deep Models:
    Bright and **Dark** Side of **Scene Context —** NeurIPS'18, CVPR'19, ECCV'20
  - **Security** of Deep Models
    Reverse Engineering and **Stealing** of Deep Models — ICLR'18, CVPR'19, ICLR'20

Motivation: we aim for Inherent Interpretability

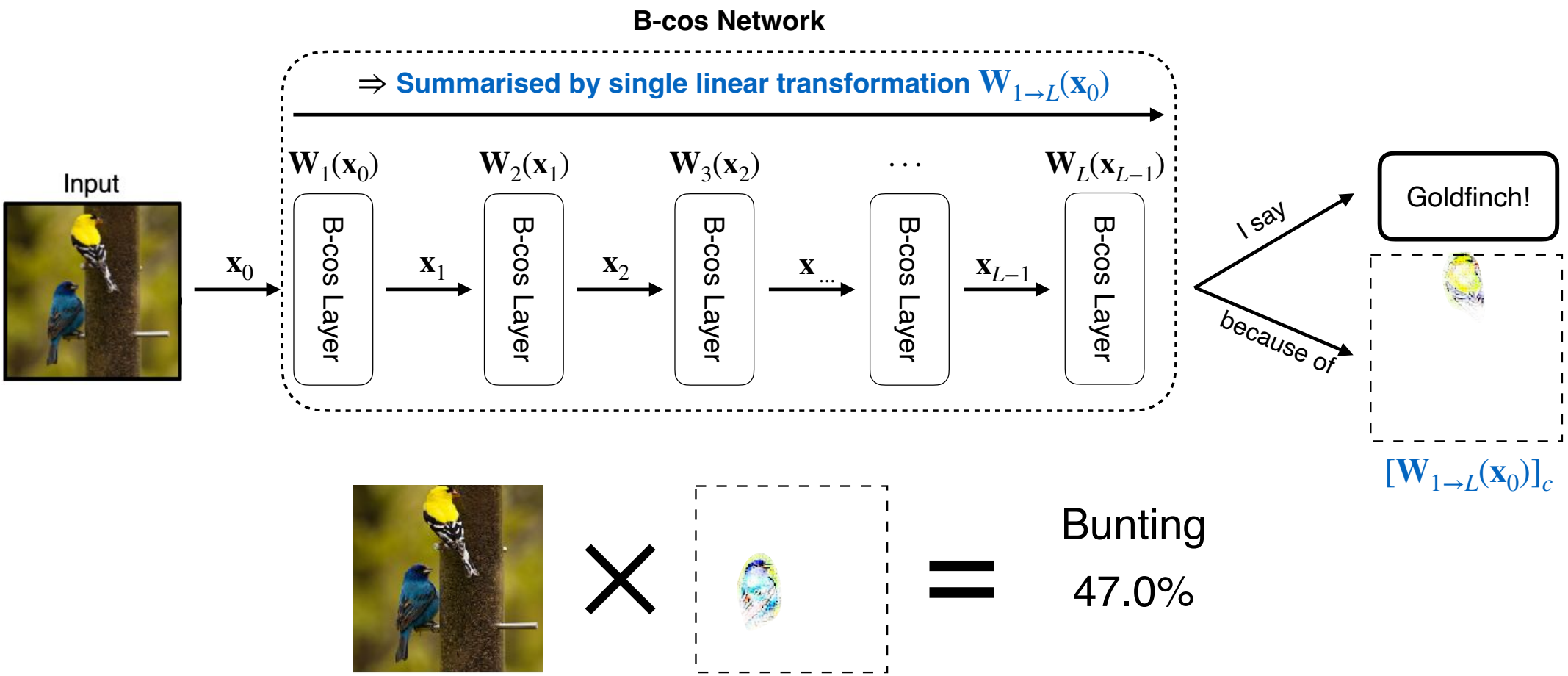


References: 'Requirements' (Gilpin et al., 2018), VGG-11 (Simonyan et al., 2014), Grad (Baehrens et al., 2010), Guided Backpropagation (Springenberg et al., 2014), Sanity check (Adebayo et al., 2018)

Dynamic linearity



Dynamic linearity



slide credit: Moritz Böhle

Visualisations of $\mathbf{W}_{1\rightarrow L}(\mathbf{x})$

Computer Vision & Machine Learning: Some Research Themes

- **Interpretability, Robustness** and **Security** of Deep Learning in Computer Vision
  - **Inherently Interpretable** Deep Neural networks — CVPR'21, CVPR'22
  - **Robustness** of Deep Models:
    Bright and **Dark** Side of **Scene Context —** NeurIPS'18, CVPR'19, ECCV'20
  - **Security** of Deep Models
    Reverse Engineering and **Stealing** of Deep Models — ICLR'18, CVPR'19, ICLR'20

Recognition: the Role of Context

- Antonio Torralba

[Shetty, Fritz, Schiele; NeurIPS19, ECCV20]

The Bright and the Dark Side of Scene Context

- Current models heavily rely on scene context:
  - Original image with cars on the left side:

original ($\mathcal{I}$) Upernet

  - Same image without those cars:

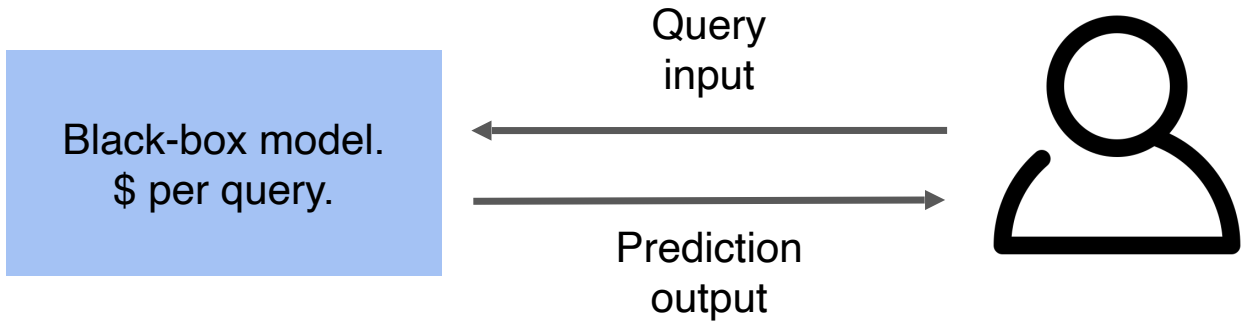
Computer Vision & Machine Learning: Some Research Themes

- **Interpretability, Robustness** and **Security** of Deep Learning in Computer Vision
  - **Inherently Interpretable** Deep Neural networks — CVPR'21, CVPR'22
  - **Robustness** of Deep Models:
    Bright and **Dark** Side of **Scene Context —** NeurIPS'18, CVPR'19, ECCV'20
  - **Security** of Deep Models
    Reverse Engineering and **Stealing** of Deep Models — ICLR'18, CVPR'19, ICLR'20

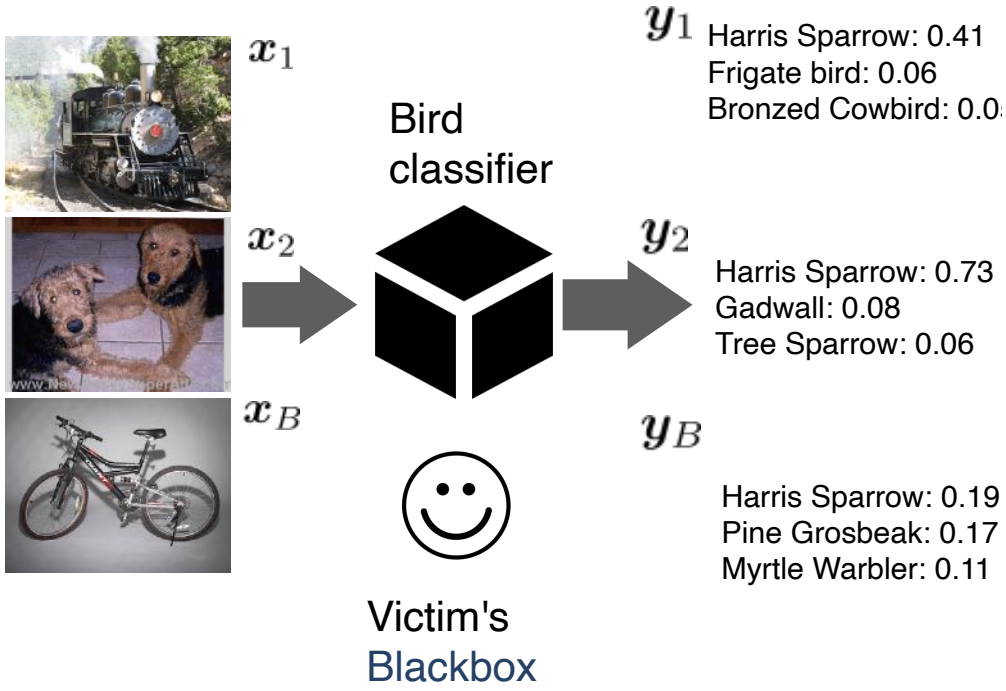
Providing ML Models is a Business Model

- **Input in, prediction out**. Ask $ per query.
  - ML models are **black boxes** !
  - not shared: **architecture**, **parameters**, **hyperparameter** details (IPs)

- **Research question**:
  - can an adversary **steal the functionality of the model** ?

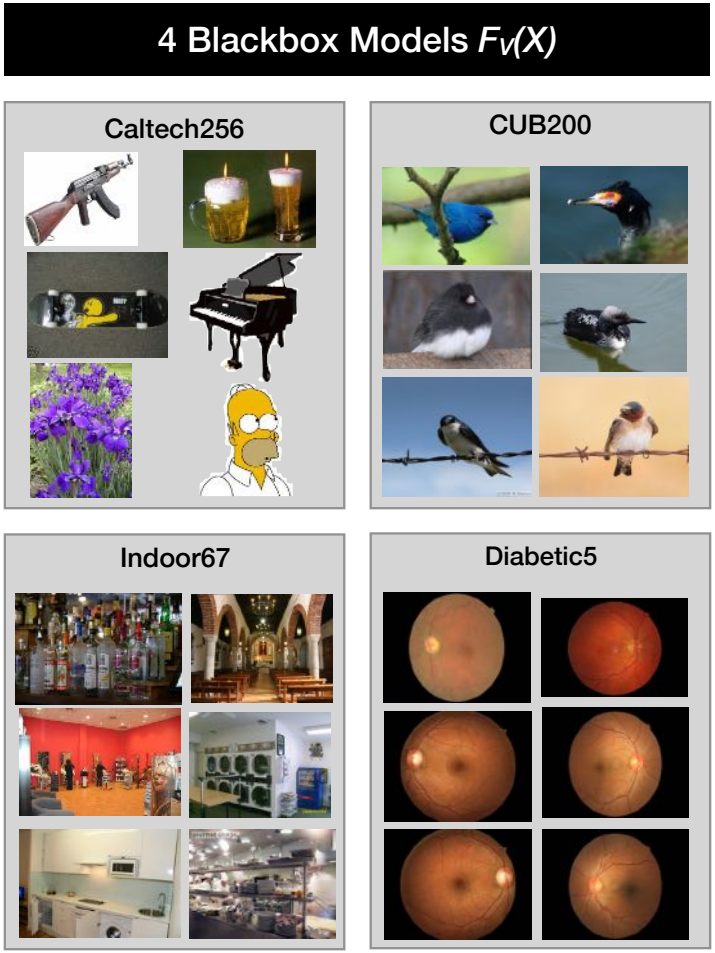


Functionality Stealing: Knock-Off Nets



Transfer Set Construction: $x_i \overset{\pi}{\sim} P_A(X)$

- Simple method: π = random
  - sample images randomly (without replacement)
  - prone to querying irrelevant images



Can we Learn with π = Random? Yes!

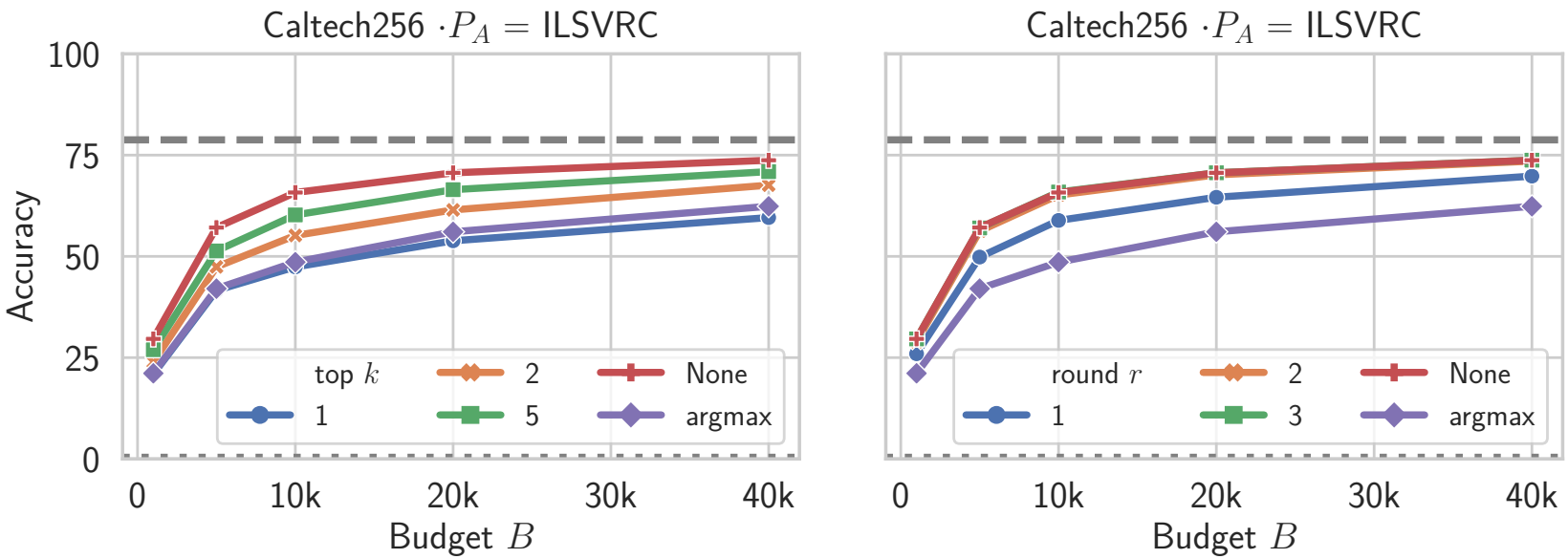
		random			
	P_A	Caltech256	CUBS200	Indoor67	Diabetic5
	P_V (F_V)	78.8 (1×)	76.5 (1×)	74.9 (1×)	58.1 (1×)
	P_V (KD)	82.6 (1.05×)	70.3 (0.92×)	74.4 (0.99×)	54.3 (0.93×)
Closed	D^2	76.6 (0.97×)	68.3 (0.89×)	68.3 (0.91×)	48.9 (0.84×)
Open	ILSVRC	75.4 (0.96×)	68.0 (0.89×)	66.5 (0.89×)	47.7 (0.82×)
Open	OpenImg	73.6 (0.93×)	65.6 (0.86×)	69.9 (0.93×)	47.0 (0.81×)

accuracy(victim blackbox)

accuracy(knockoff)

$\Rightarrow$ > 0.81× accuracy of blackbox recovered

Learning with Less Information? Yes!



⇒ Robust to various **passive** defense mechanisms:

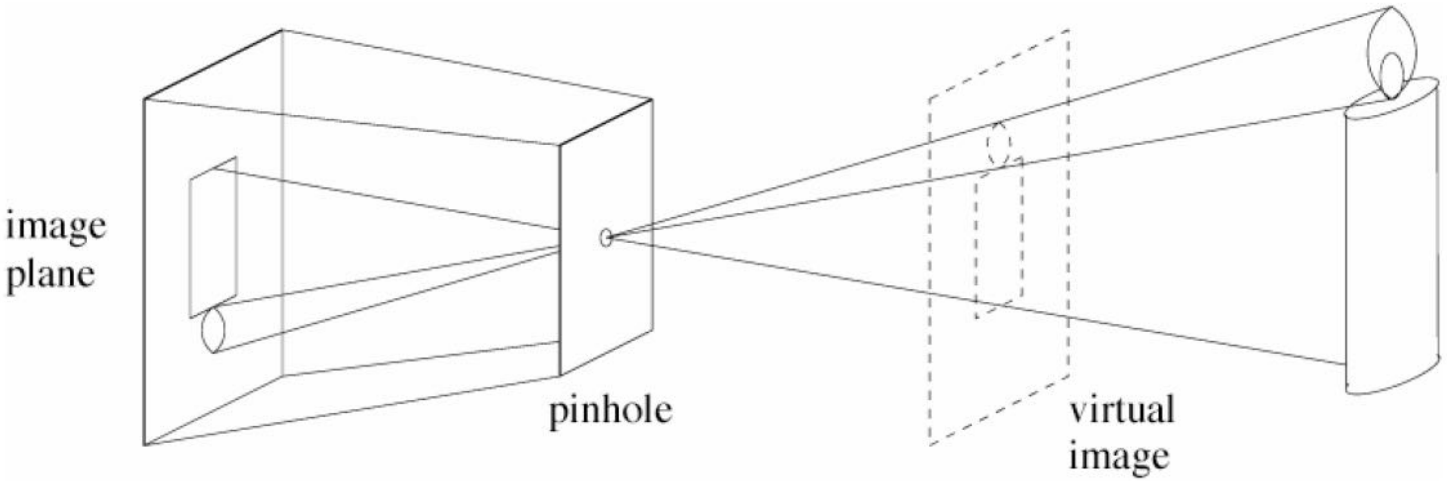
e.g. argmax, top-k, rounding, …

Basic Concepts and Terminology

Computer Vision vs. Computer Graphics

Pinhole Camera (Model)

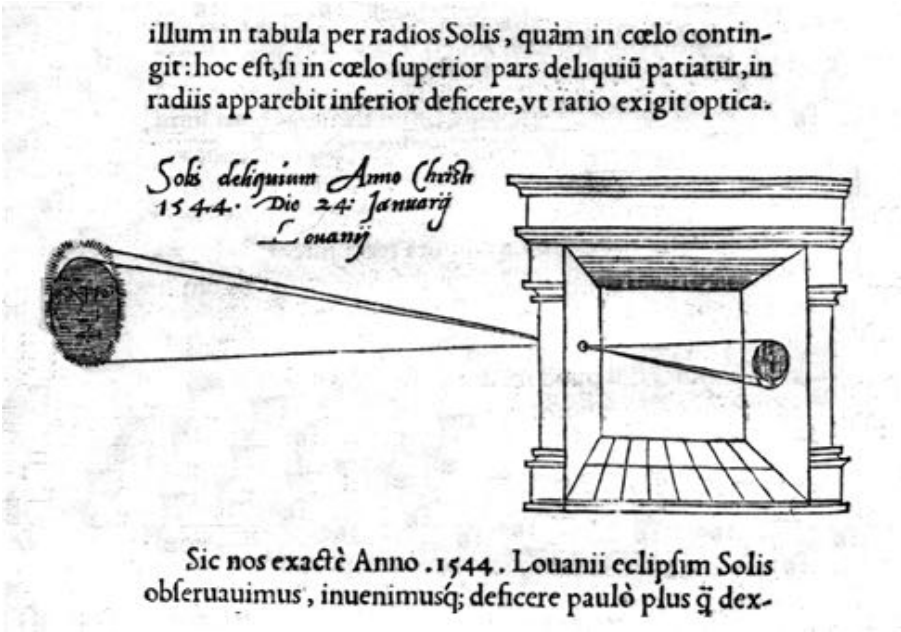
- (simple) standard and abstract model today
  - box with a small hole in it



Camera Obscura

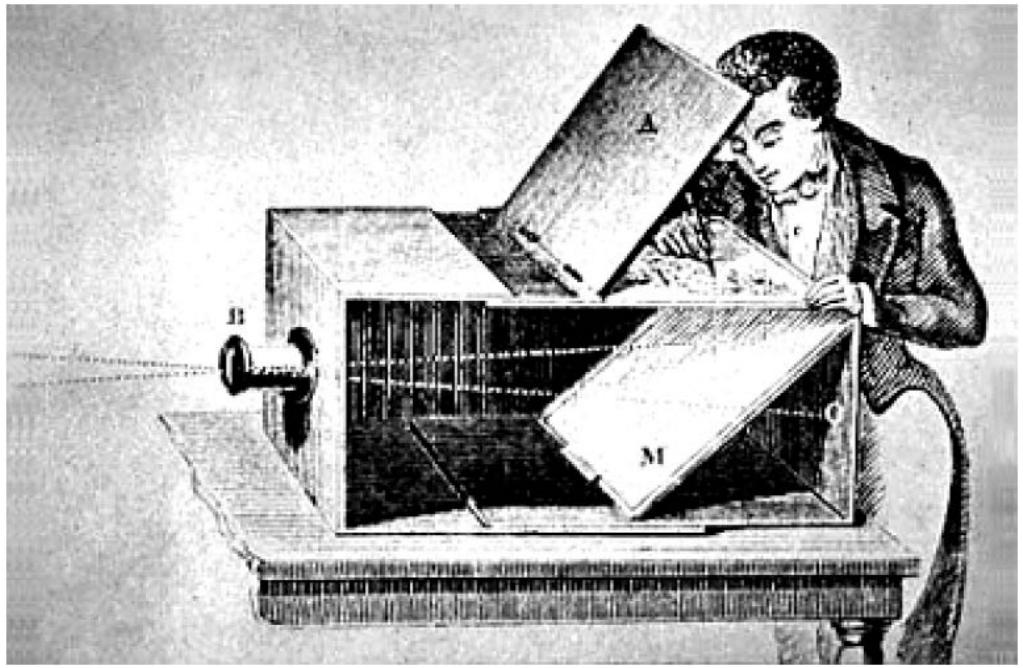
- around 1519, Leonardo da Vinci (1452 - 1519)
  - http://www.acmi.net.au/AIC/CAMERA_OBSCURA.html

  - "when images of illuminated objects … penetrate through a small hole into a very dark room … you will see [on the opposite wall] these objects in their proper form and color, reduced in size … in a reversed position owing to the intersection of the rays"



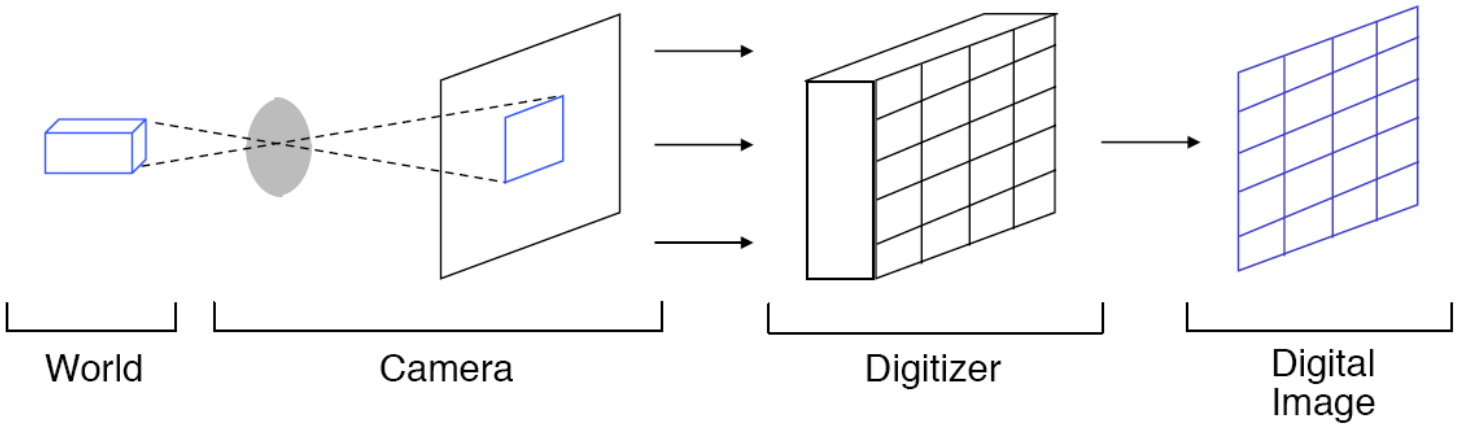
Principle of pinhole....

- ...used by artists
  - (e.g. Vermeer 17th century, dutch)
- and scientists



Digital Images

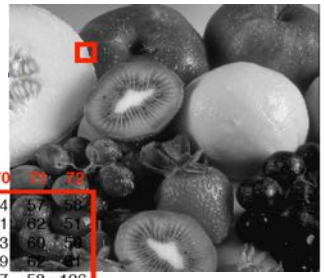
- Imaging Process:
  - (pinhole) camera model
  - digitizer to obtain digital image



(Grayscale) Image

- ‘Goals’ of Computer Vision
  - how can we recognize fruits from an array of (gray-scale) numbers?
  - how can we perceive depth from an array of (gray-scale) numbers?
  - …
- computer vision = the problem of ‘inverse graphics’ …?

- ‘Goals’ of Graphics
  - how can we generate an array of (gray-scale) numbers that looks like fruits?
  - how can we generate an array of (gray-scale) numbers so that the human observer perceives depth?
  - …



x =	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72
y = 41	210	209	204	202	197	247	143	71	64	80	84	54	54	57	58
42	206	196	203	197	195	210	207	56	63	58	53	53	61	62	51
43	201	207	192	201	198	213	156	69	65	57	55	52	53	60	58
44	216	206	211	193	202	207	208	57	69	60	55	77	49	62	61
45	221	206	211	194	196	197	220	56	63	60	55	46	97	58	106
46	209	214	224	199	194	193	204	173	64	60	59	51	62	56	48
47	204	212	213	208	191	190	191	214	60	62	66	76	51	49	55
48	214	215	215	207	208	180	172	188	69	72	55	49	56	52	56
49	209	205	214	205	204	196	187	196	86	62	66	87	57	60	48
50	208	209	205	203	202	186	174	185	149	71	63	55	55	45	56
51	207	210	211	199	217	194	183	177	209	90	62	64	52	93	52
52	208	205	209	209	197	194	183	187	187	239	58	68	61	51	56
53	204	206	203	209	195	203	188	185	183	221	75	61	58	60	60
54	200	203	199	236	188	197	183	190	183	196	122	63	58	64	66
55	205	210	202	203	199	197	196	181	173	186	105	62	57	64	63

Case Study

... object recognition

Case Study:
Computer Vision & Object Recognition

- how do you recognize
  - the banana?
  - the glas?
  - the towel?

- how can we make computers to do this?

- ill posed problem:
  - missing data
  - ambiguities
  - multiple possible explanations

Image Edges:
What are edges? Where do they come from?

- Edges are changes in pixel brightness

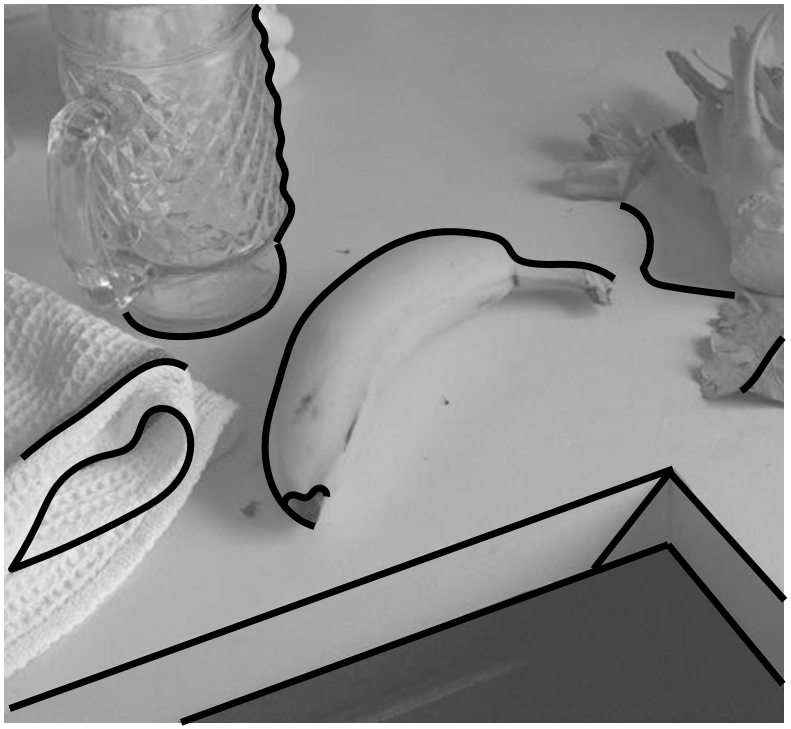
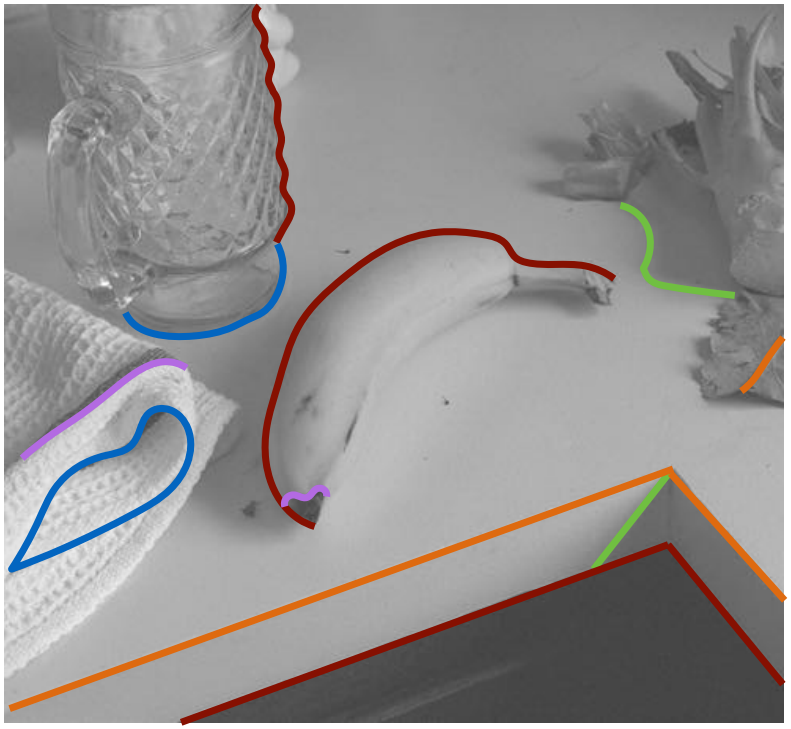


Image Edges:
What are edges? Where do they come from?



- Edges are changes in pixel brightness
  - **Foreground/Background Boundaries**
  - **Object-Object-Boundaries**
  - **Shadow Edges**
  - **Changes in Albedo or Texture**
  - **Changes in Surface Normals**

Line Drawings:
Good Starting Point for Recognition?

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

PROJECT MAC

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

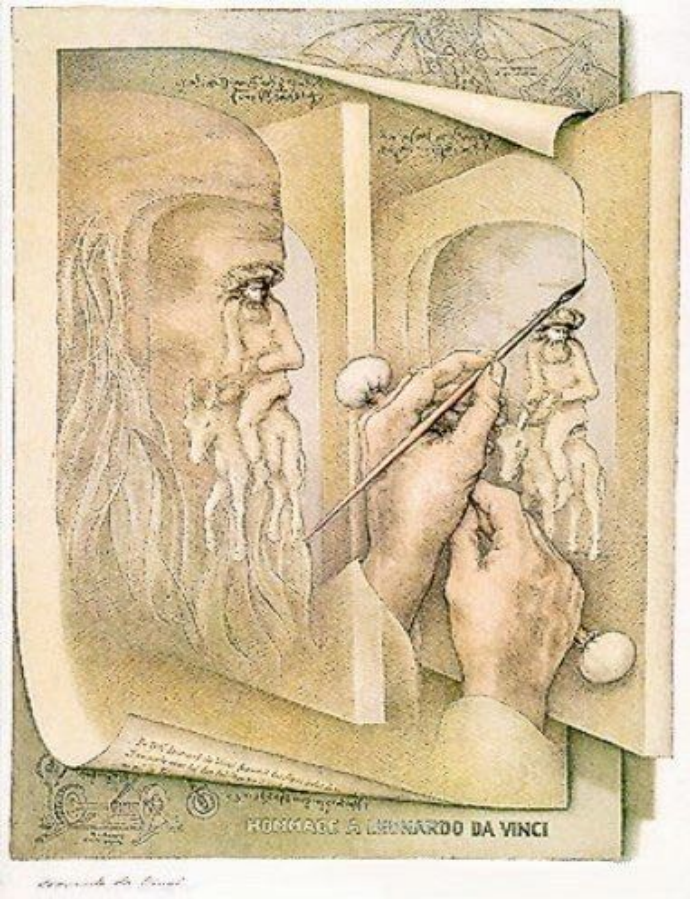
THE SUMMER VISION PROJECT

Seymour Papert

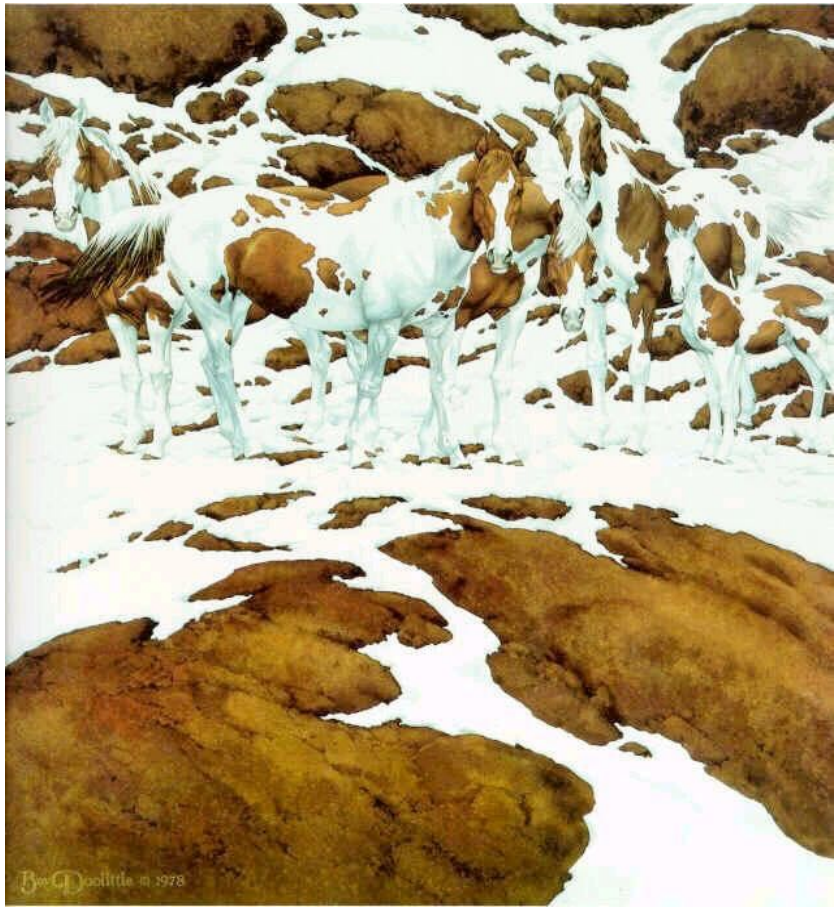
The summer vision project is an attempt to use our summer workers effectively in the construction of a significant part of a visual system. The particular task was chosen partly because it can be segmented into sub-problems which will allow individuals to work independently and yet participate in the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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

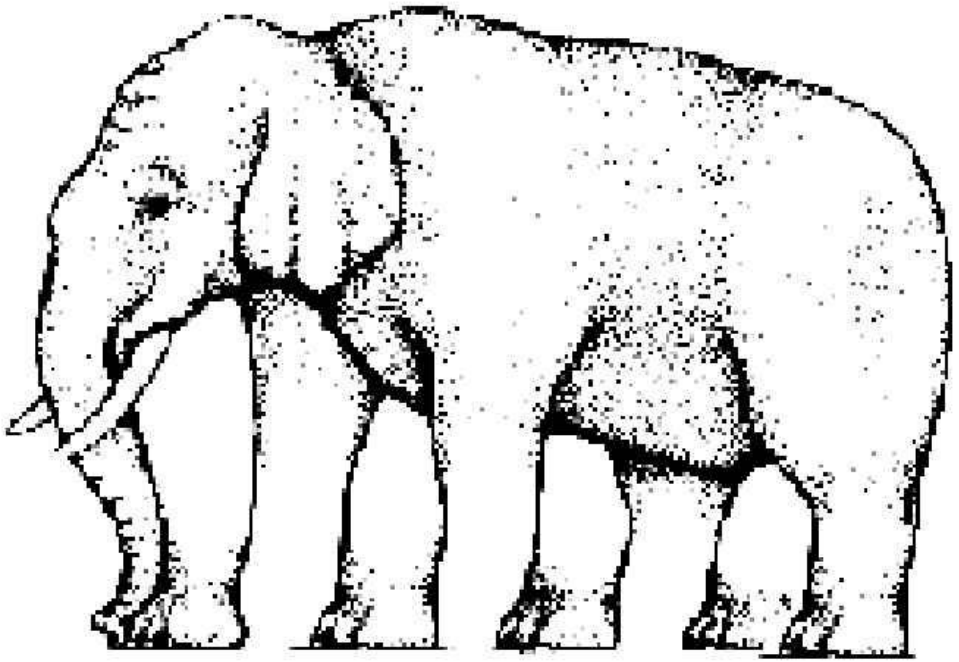
Complexity of Recognition



Complexity of Recognition



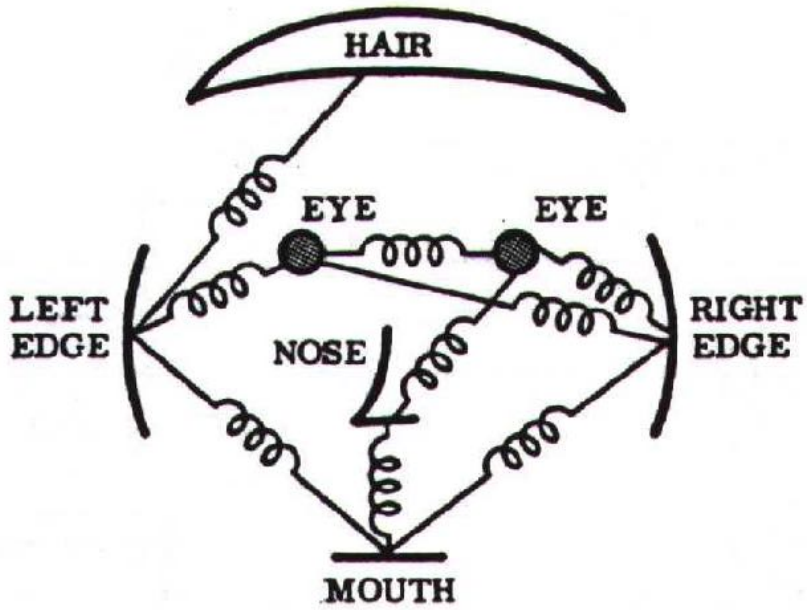
Complexity of Recognition



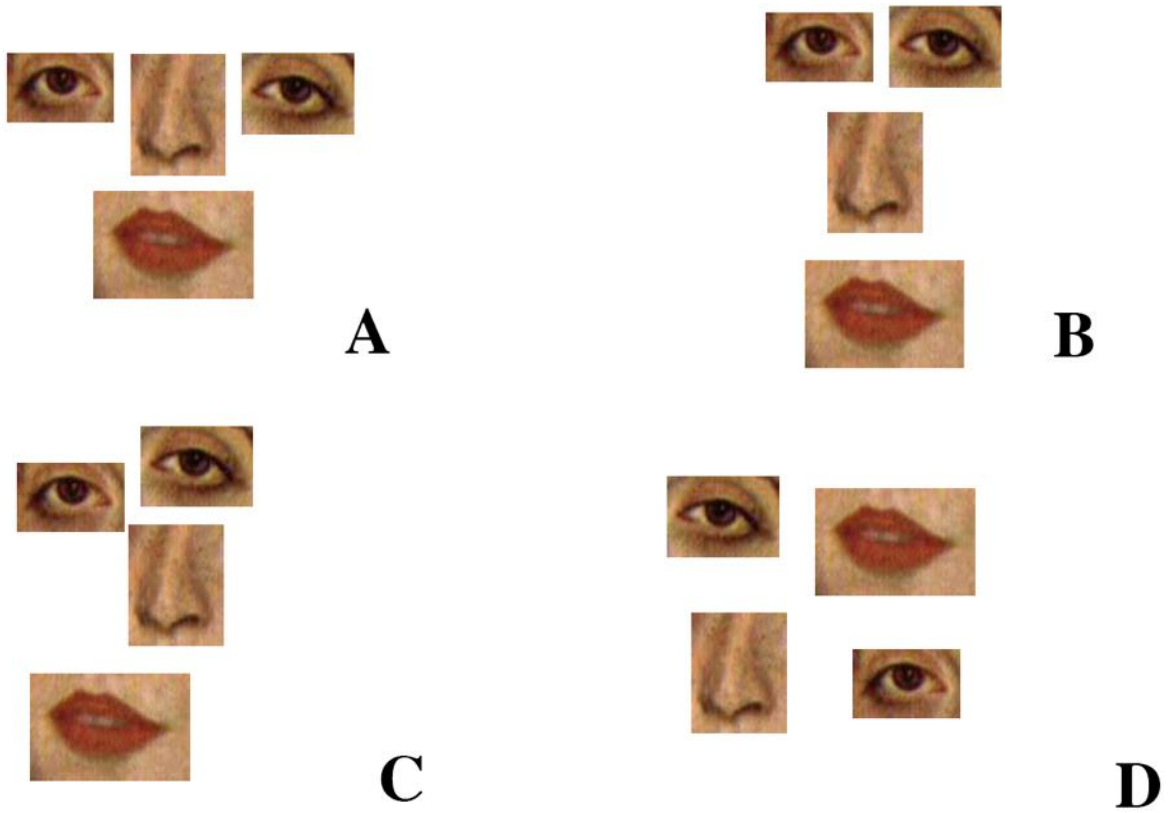
Complexity of Recognition

Class of Models: Pictorial Structure

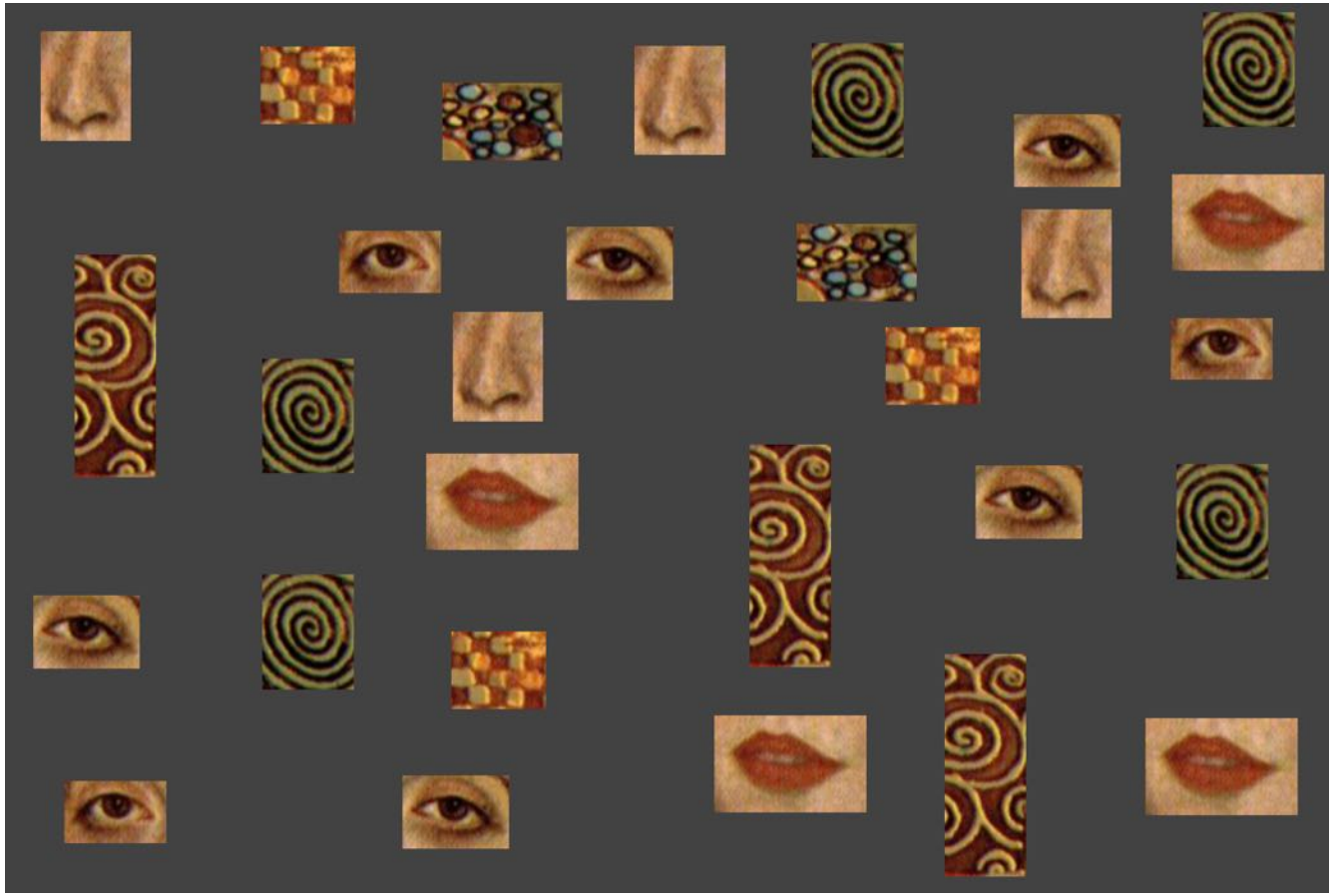
- Fischler & Elschlager 1973
- Model has two components
  - parts
    (2D image fragments)
  - structure
    (configuration of parts)



Deformations



Clutter



Example

Recognition, Localization, and Segmentation

a few terms

… let’s briefly define what we mean by that

Object Recognition:
First part of this Computer Vision class

- Different Types of Recognition Problems:
  - Object **Identification**
    - recognize your pencil, your dog, your car
  - Object **Classification**
    - recognize any pencil, any dog, any car
    - also called: generic object recognition, object categorization, …

- Recognition and
  - **Segmentation**: separate pixels belonging to the foreground (object) and the background
  - **Localization/Detection**: position of the object in the scene, pose estimate (orientation, size/scale, 3D position)

Object Recognition:
First part of this Computer Vision class

- Different Types of Recognition Problems:
  - Object **Identification**
    - recognize your apple, your cup, your dog
  - Object **Classification**
    - recognize any apple, any cup, any dog
    - also called: generic object recognition, object categorization, …
    - typical definition: ‘basic level category’

Which Level is right for Object Classes?

- Basic-Level Categories
  - the highest level at which category members have **similar perceived shape**
  - the highest level at which a **single mental image** can reflect the entire category
  - the highest level at which a person uses similar **motor actions** to interact with category members
  - the level at which human subjects are usually **fastest** at identifying category members
  - the first level named and understood by **children**

  - (while the definition of basic-level categories depends on culture there exist a remarkable consistency across cultures...)

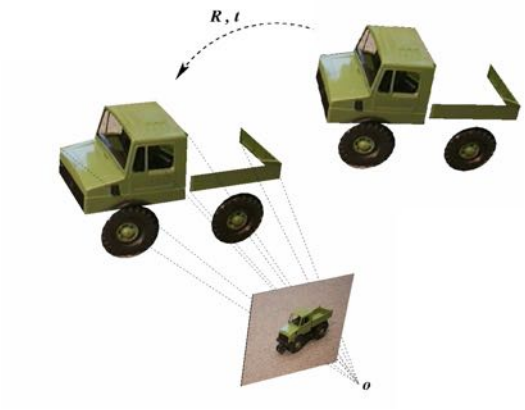
- Most recent work in object recognition has focused on this problem
  - we will discuss several of the most successful methods in the lecture :-)

Object Recognition & Segmentation

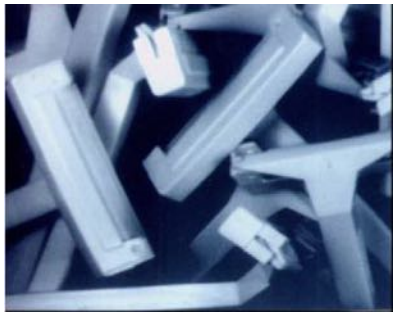
- Recognition and
  - **Segmentation**: separate pixels belonging to the foreground (object) and the background

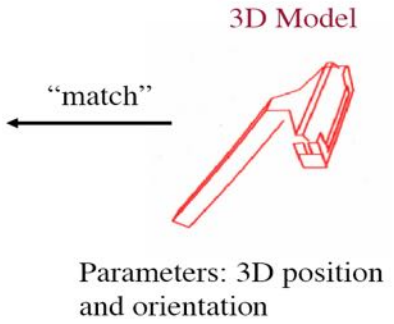
Object Recognition & Localization

- Recognition and
  - **Localization in 3D**: to position the object in the scene, estimate the object's pose (orientation, size/scale, 3D position)



  - Example from David Lowe:





Object Recognition

- Different Types of Recognition Problems:
  - Object **Identification**
    - recognize your pencil, your dog, your car
  - Object **Classification**
    - recognize any pencil, any dog, any car
    - also called: generic object recognition, object categorization, …

- Recognition and
  - **Segmentation**: separate pixels belonging to the foreground (object) and the background
  - **Localization**: position the object in the scene, estimate pose of the object (orientation, size/scale, 3D position)

Goals of today's lecture

- First intuitions about
  - What is computer vision?
  - What does it mean to see and how do we (as humans) do it?
  - How can we make this computational?

- Applications & Appetizers

- Case Study
  - Object Recognition — intuition from human vision…

- Image Classification
  - using data-driven approaches (machine learning)
  - K-nearest neighbor classifier

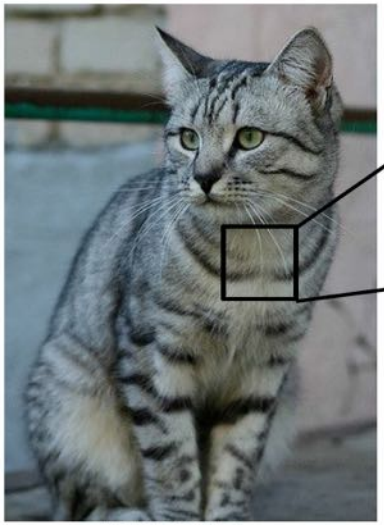
Image Classification: a core task in computer vision

(assume given set of discrete labels)
{dog, cat, truck, plane, ...}

→ cat

Image Classification

The Problem: Semantic Gap




```
[[105 112 108 111 104  99 106  99  96 103 112 119 104  97  93  87]
 [ 91  98 102 106 104  79  98 103  99 105 123 136 110 105  94  85]
 [ 76  85  90 105 128 105  87  96  95  99 115 112 106 103  99  85]
 [ 99  81  81  93 120 131 127 100  95  98 102  99  96  93 101  94]
 [106  91  61  64  69  91  88  85 101 107 109  98  75  84  96  95]
 [114 108  85  55  55  69  64  54  64  87 112 129  98  74  84  91]
 [133 137 147 103  65  81  80  65  52  54  74  84 102  93  85  82]
 [128 137 144 140 109  95  86  70  62  65  63  63  60  73  86 101]
 [125 133 148 137 119 121 117  94  65  79  80  65  54  64  72  90]
 [127 125 131 147 133 127 126 131 111  96  89  75  61  64  72  84]
 [115 114 109 123 150 148 131 118 113 109 100  92  74  65  72  78]
 [ 89  93  90  97 108 147 131 118 113 114 113 109 106  95  77  80]
 [ 63  77  86  81  77  79 102 123 117 115 117 125 125 130 115  87]
 [ 62  65  82  89  78  71  80 101 124 126 119 101 107 114 131 119]
 [ 63  65  75  88  89  71  62  81 120 138 135 105  81  98 110 118]
 [ 87  65  71  87 106  95  69  45  76 130 126 107  92  94 105 112]
 [118  97  82  86 117 123 116  66  41  51  95  93  89  95 102 107]
 [164 146 112  80  82 120 124 104  76  48  45  66  88 101 102 109]
 [157 170 157 120  93  86 114 132 112  97  69  55  70  82  99  94]
 [130 128 134 161 139 100 109 118 121 134 114  87  65  53  69  86]
 [128 112  96 117 150 144 120 115 104 107 102  93  87  81  72  79]
 [123 107  96  86  83 112 153 149 122 109 104  75  80 107 112  99]
 [122 121 102  80  82  86  94 117 145 148 153 102  58  78  92 107]
 [122 164 148 103  71  56  78  83  93 103 119 139 102  61  69  84]]
```

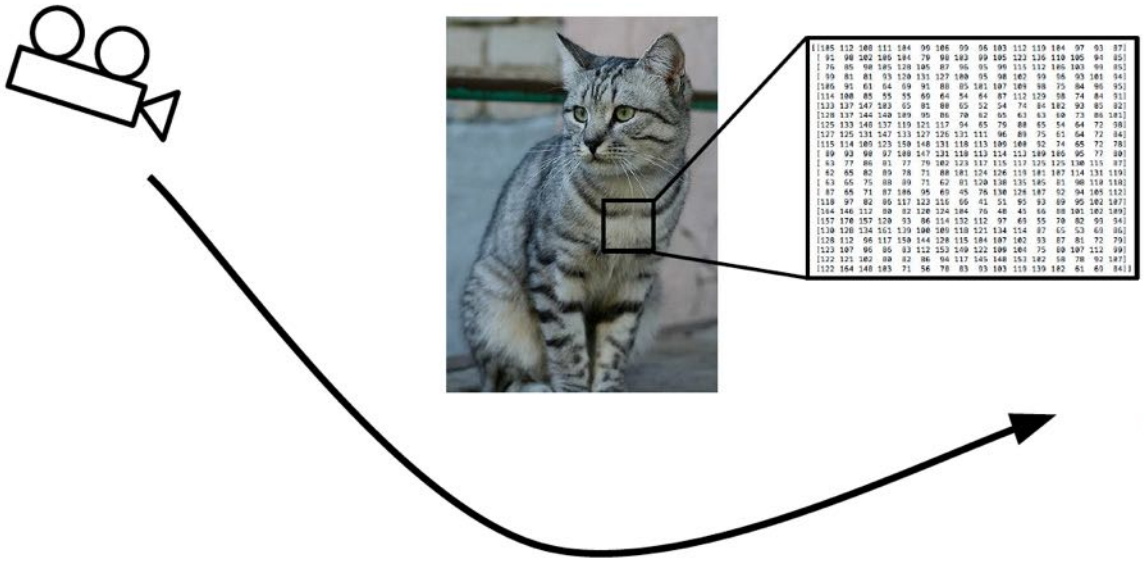
What the computer sees

An image is just a big grid of numbers between [0, 255]:

e.g. 800 x 600 x 3
(3 channels RGB)

Image Classification

Challenges: Viewpoint variation



All pixels change when the camera moves!

Image Classification

Challenges: Illumination


This image is CC0 1.0 public domain


This image is CC0 1.0 public domain


This image is CC0 1.0 public domain

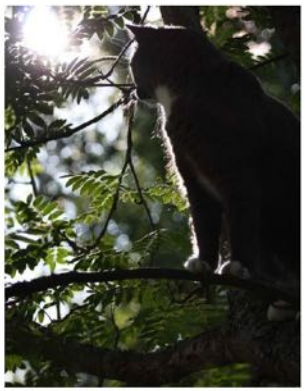
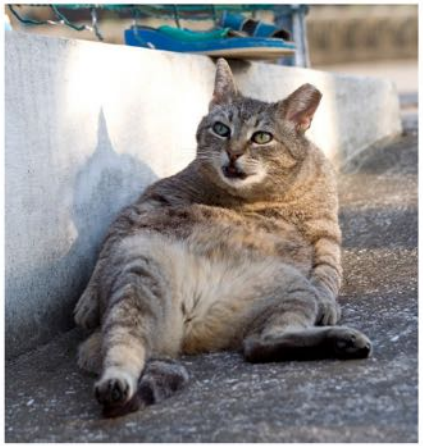
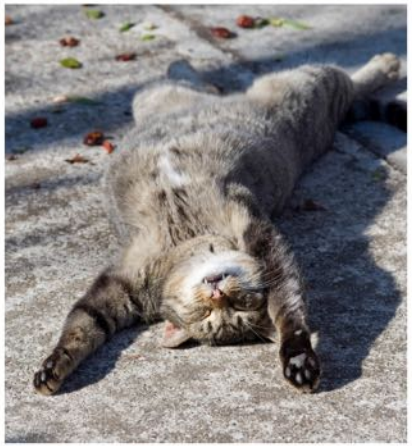

This image is CC0 1.0 public domain

Image Classification

Challenges: Deformation

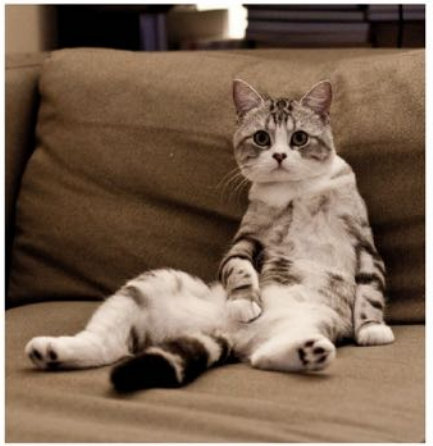


This image by Umberto Salvagnin is licensed under CC-BY 2.0



This image by Umberto Salvagnin is licensed under CC-BY 2.0

This image by sare bear is licensed under CC-BY 2.0



This image by Tom Thai is licensed under CC-BY 2.0

Image Classification

Challenges: Occlusion

This image is CC0 1.0 public domain

This image is CC0 1.0 public domain

This image by jonsson is licensed under CC-BY 2.0

Image Classification

Challenges: Background Clutter

This image is CC0 1.0 public domain

This image is CC0 1.0 public domain

Image Classification

Challenges: Intraclass variation

This image is CC0 1.0 public domain

Image Classification

An image classifier

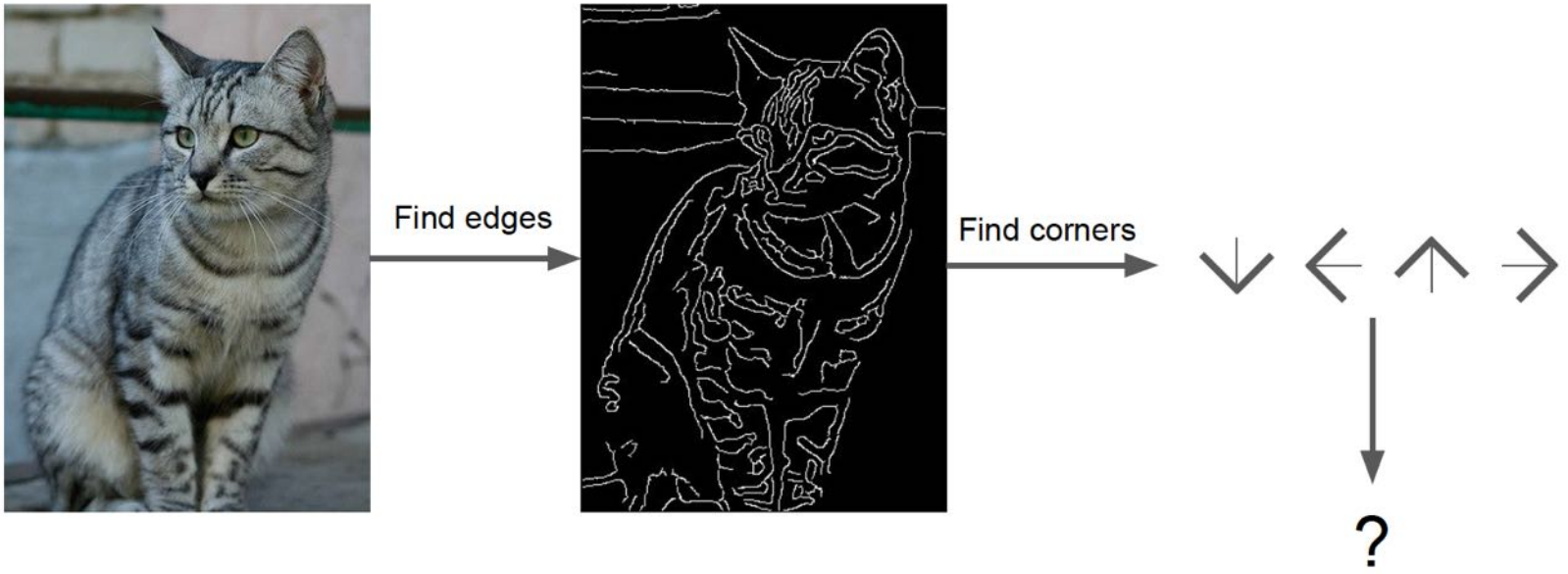
```
def classify_image(image):
  # Some magic here?
  return class_label
```

Unlike e.g. sorting a list of numbers,

no obvious way to hard-code the algorithm for recognizing a cat, or other classes.

Image Classification

Attempts have been made



John Canny, "A Computational Approach to Edge Detection", IEEE TPAMI 1986

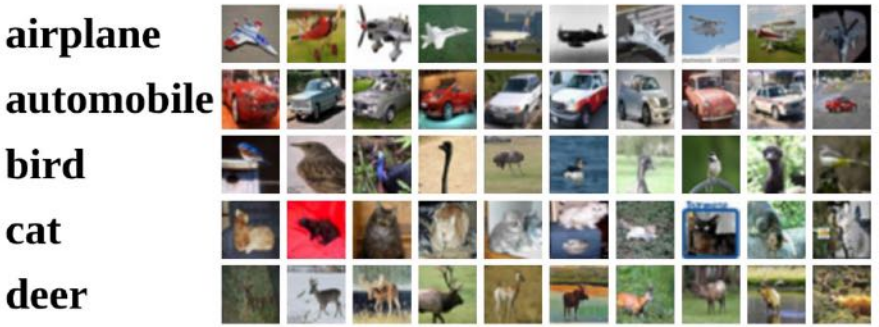
Machine Learning: Data-Driven Approach

1. Collect a dataset of images and labels
2. Use Machine Learning to train a classifier
3. Evaluate the classifier on new images

```
def train(images, labels):
  # Machine learning!
  return model
```

```
def predict(model, test_images):
  # Use model to predict labels
  return test_labels
```

Example training set



First Classifier: Nearest Neighbor

```
def train(images, labels):
  # Machine learning!
  return model
```

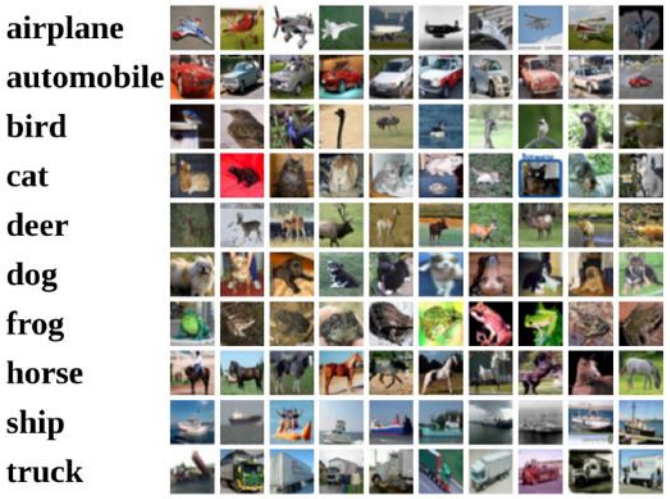
→ Memorize all data and labels

```
def predict(model, test_images):
  # Use model to predict labels
  return test_labels
```

→ Predict the label of the most similar training image

Example Dataset: CIFAR10

10 classes
50,000 training images
10,000 testing images



Test images and nearest neighbors

Alex Krizhevsky, "Learning Multiple Layers of Features from Tiny Images", Technical Report, 2009.

First Classifier: Nearest Neighbor Classifier

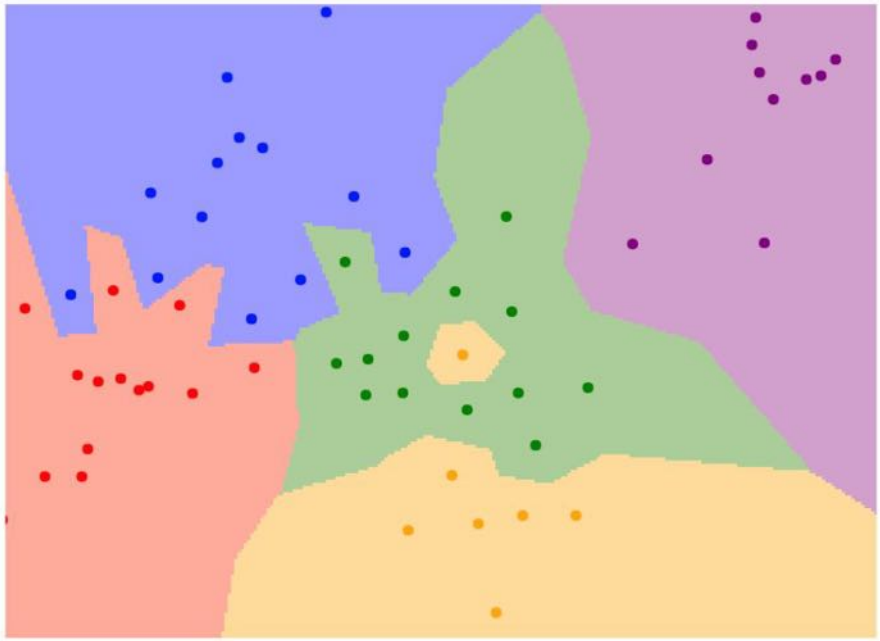
Distance Metric to compare images

L1 distance: $d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$

test image					training image					pixel-wise absolute value differences				
56	32	10	18	-	10	20	24	17	=	46	12	14	1	add → 456
90	23	128	133		8	10	89	100		82	13	39	33	
24	26	178	200		12	16	178	170		12	10	0	30	
2	0	255	220		4	32	233	112		2	32	22	108	

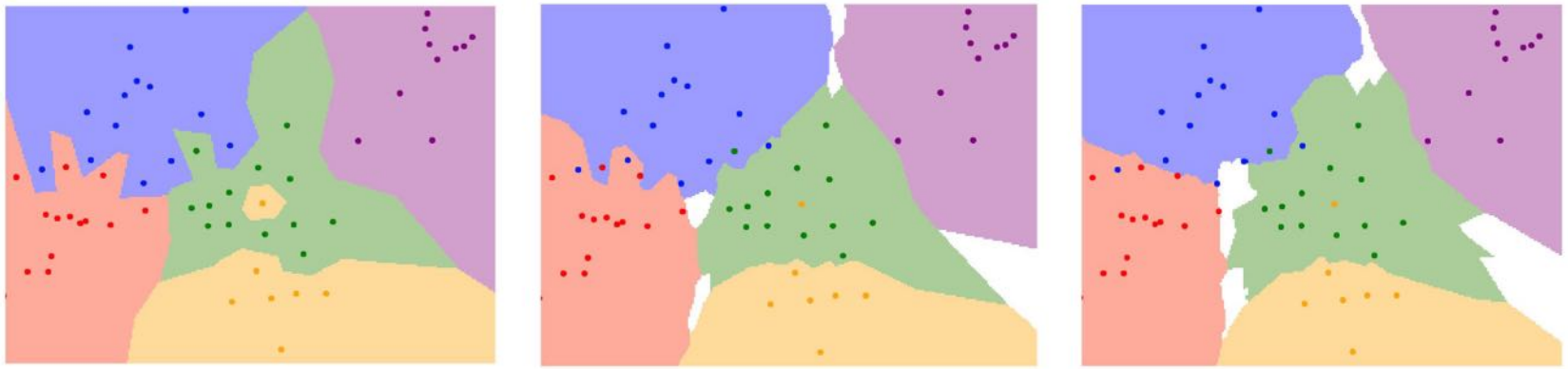
First Classifier: Nearest Neighbor Classifier

What does this look like?



K-Nearest Neighbor Classifier

Instead of copying label from nearest neighbor,
take **majority vote** from K closest points



K = 1 K = 3 K = 5

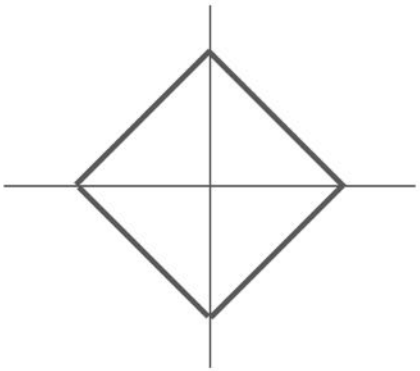
K-Nearest Neighbor Classifier

What does this look like?

K-Nearest Neighbor Classifier: Distance Metric

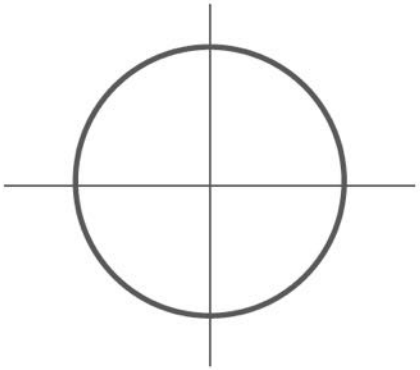
L1 (Manhattan) distance

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$



L2 (Euclidean) distance

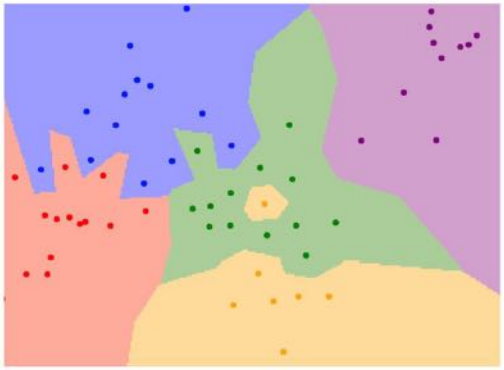
$$d_2(I_1, I_2) = \sqrt{\sum_p \left(I_1^p - I_2^p\right)^2}$$



K-Nearest Neighbor Classifier: Distance Metric

L1 (Manhattan) distance

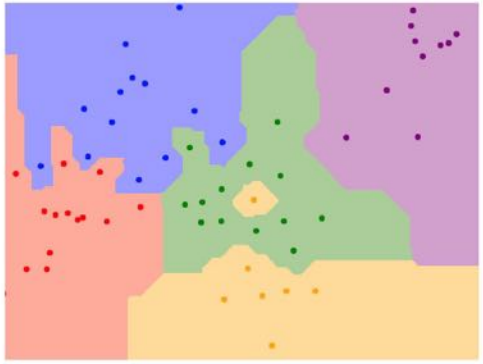
$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$



K = 1

L2 (Euclidean) distance

$$d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$$



K = 1

Hyperparameters

What is the best value of **k** to use?
What is the best **distance** to use?

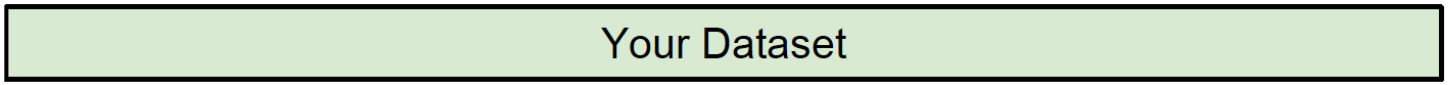
These are **hyperparameters**: choices about the algorithm that we set rather than learn

Very problem-dependent.
Must try them all out and see what works best.

Setting Hyperparameters

Idea #1: Choose hyperparameters that work best on the data

BAD: K = 1 always works perfectly on training data



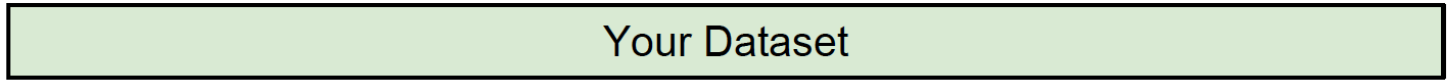
Idea #2: Split data into **train** and **test**, choose hyperparameters that work best on test data

BAD: No idea how algorithm will perform on new data

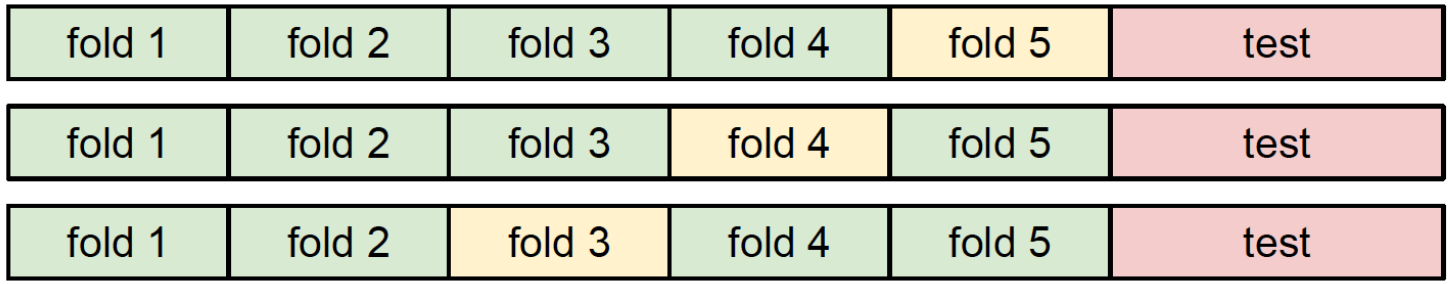
Idea #3: Split data into **train**, **val**, and **test**; choose hyperparameters on val and evaluate on test

Better!

Setting Hyperparameters

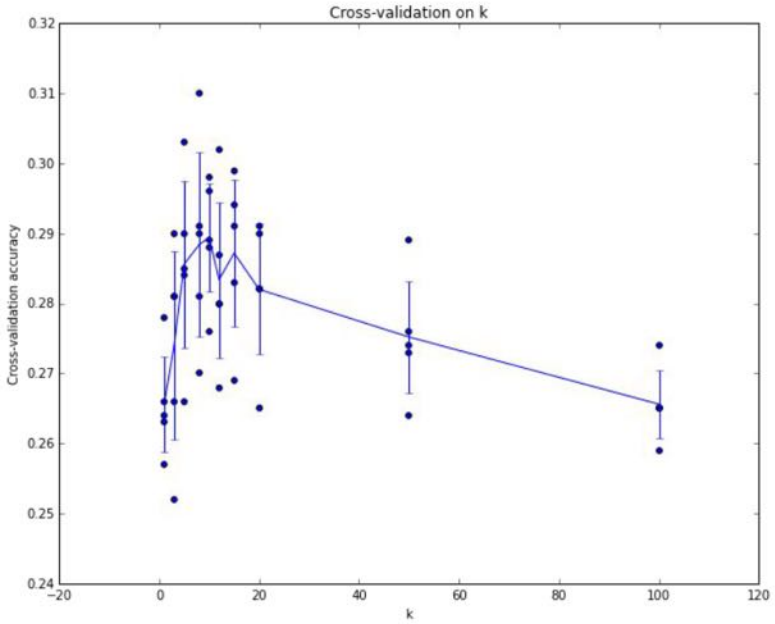


Idea #4: **Cross-Validation**: Split data into **folds**,
try each fold as validation and average the results



Useful for small datasets, but not used too frequently in deep learning

Setting Hyperparameters



Example of
5-fold cross-validation
for the value of **k.**

Each point: single
outcome.

The line goes
through the mean, bars
indicated standard
deviation

(Seems that k ~= 7 works best
for this data)

Nearest Neighbor - not used for images :-)

- Very slow at test time
- Distance metrics on pixels are not informative

Original image is CC0 public domain

(all 3 images have same L2 distance to the one on the left)

K-Nearest Neighbors: Summary

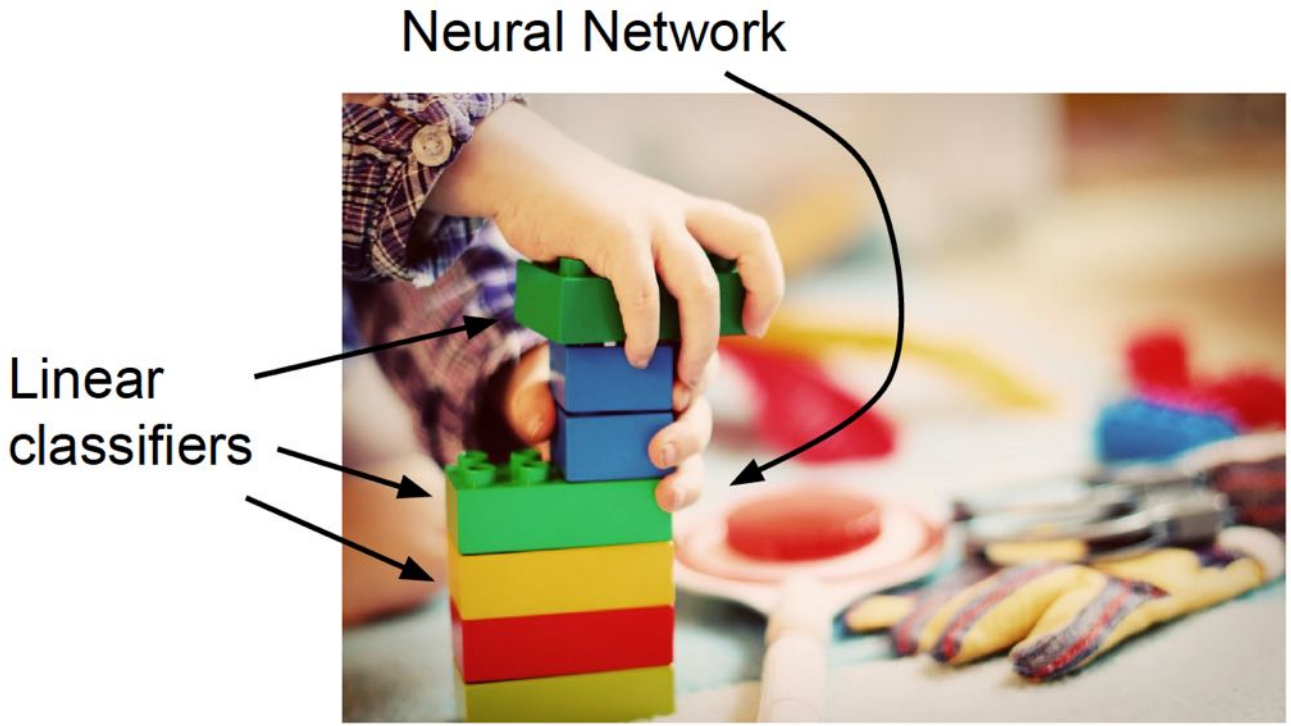
In **Image classification** we start with a **training set** of images and labels, and must predict labels on the **test set**

The **K-Nearest Neighbors** classifier predicts labels based on nearest training examples

Distance metric and K are **hyperparameters**

Choose hyperparameters using the **validation set**; only run on the test set once at the very end!

Linear Classification



This image is CC0 1.0 public domain