Lecture 1: Introduction

Machine Learning (BBWL)

Michael Mommert, University of St. Gallen

Today's lecture

What this course is about…

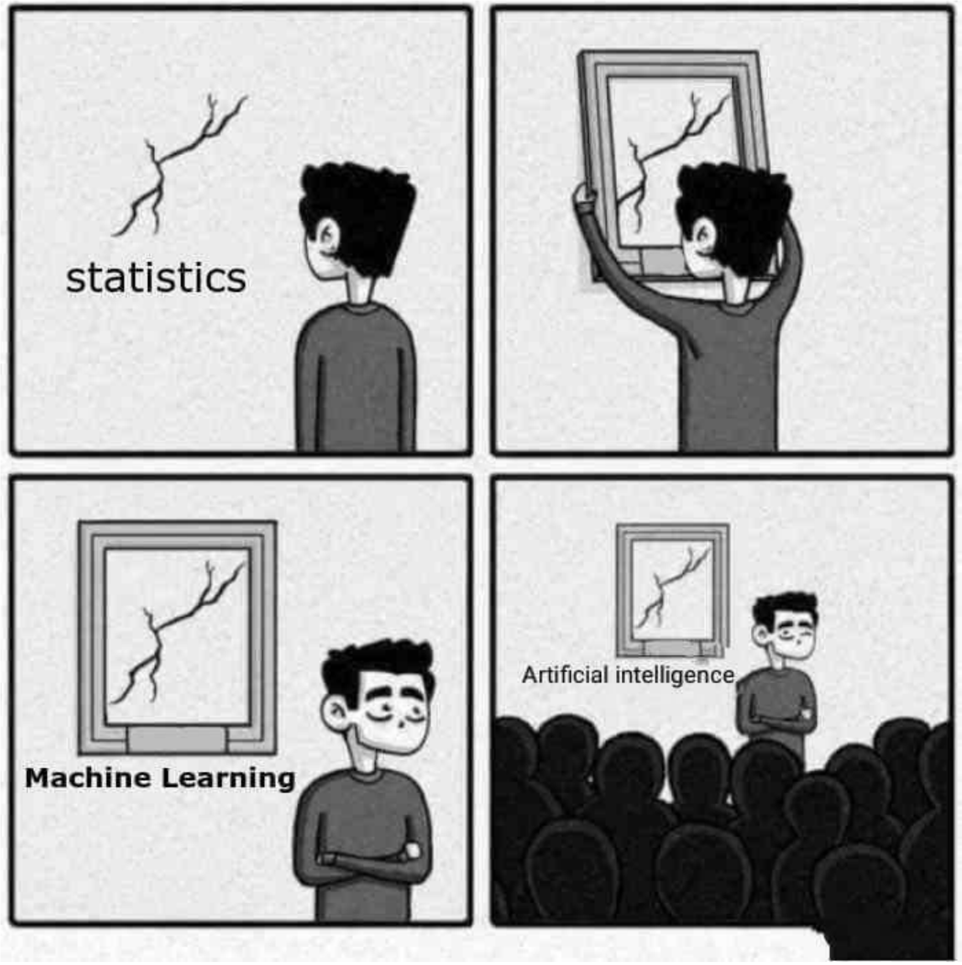
Course modalities

Course syllabus

About myself

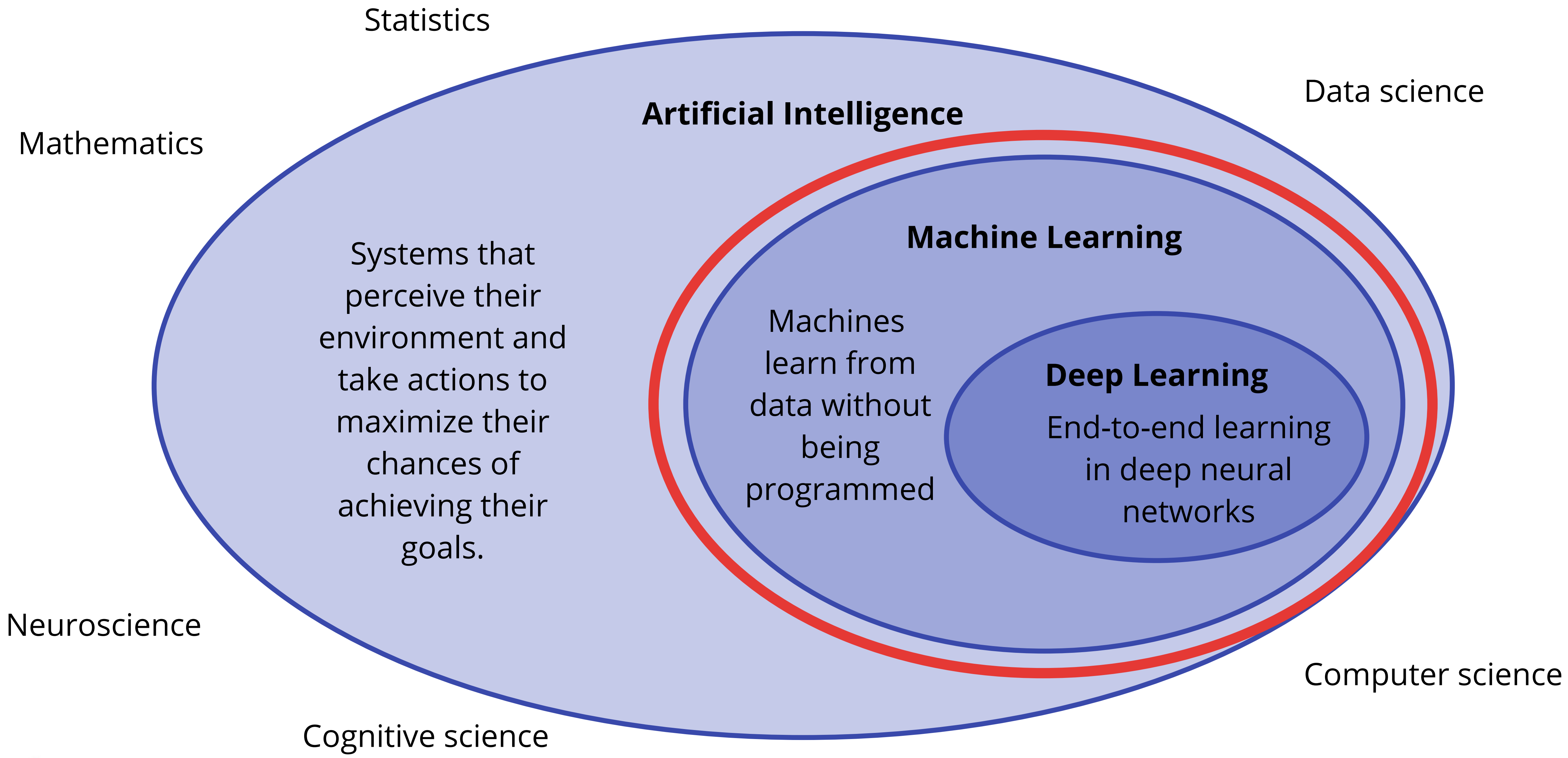
How did we get here?

What this course is about...



sandserif

Mapping terminology



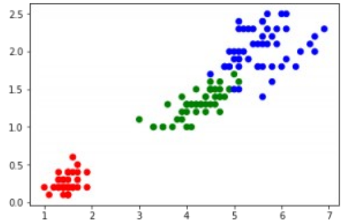
What is Machine Learning (ML)?

"The field of study that gives computers the ability to learn without being explicitly programmed."
- Arthur Samuel (1959)

Different approaches:

Iris Versicolor

- **Supervised learning**
  Find a function that relates input data to output data by learning a specific task.



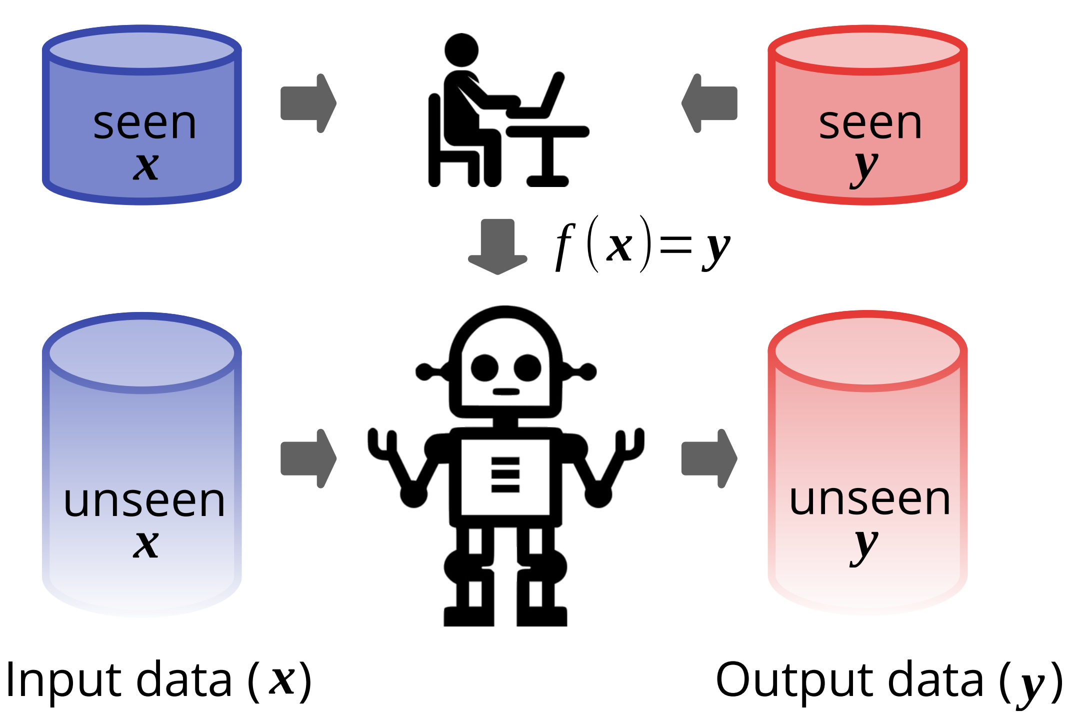
- **Unsupervised learning**
  Find structure within a data set.

- **Reinforcement learning**
  Learn a task in a dynamic and responsive environment.

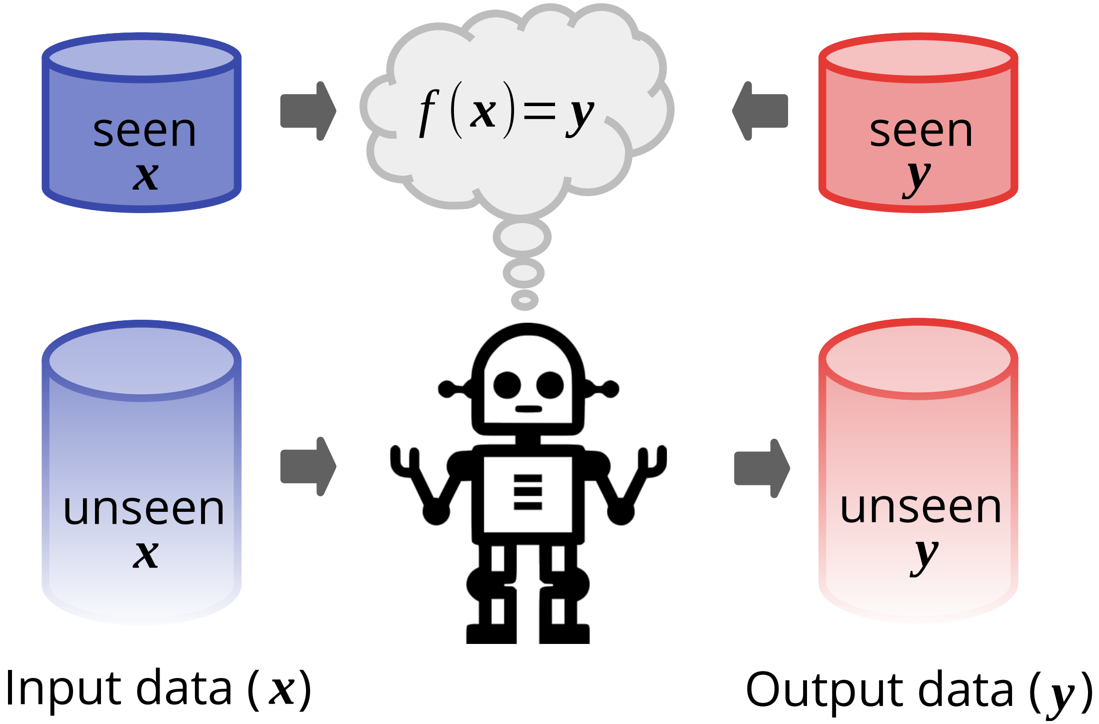
Supervised ML

General goal for supervised problems:
Find a function (“task”) that relates input data ($\boldsymbol{x}$) to output data ($\boldsymbol{y}$) such that: $f(\boldsymbol{x}) = \boldsymbol{y}$

Traditional (Rule-based) Approach:



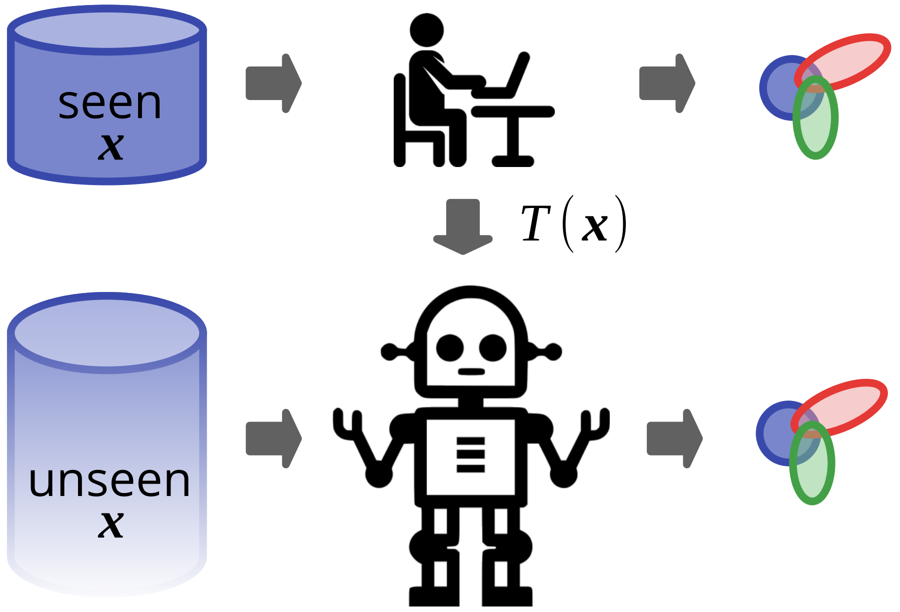
Machine-Learning Approach:



Unsupervised ML

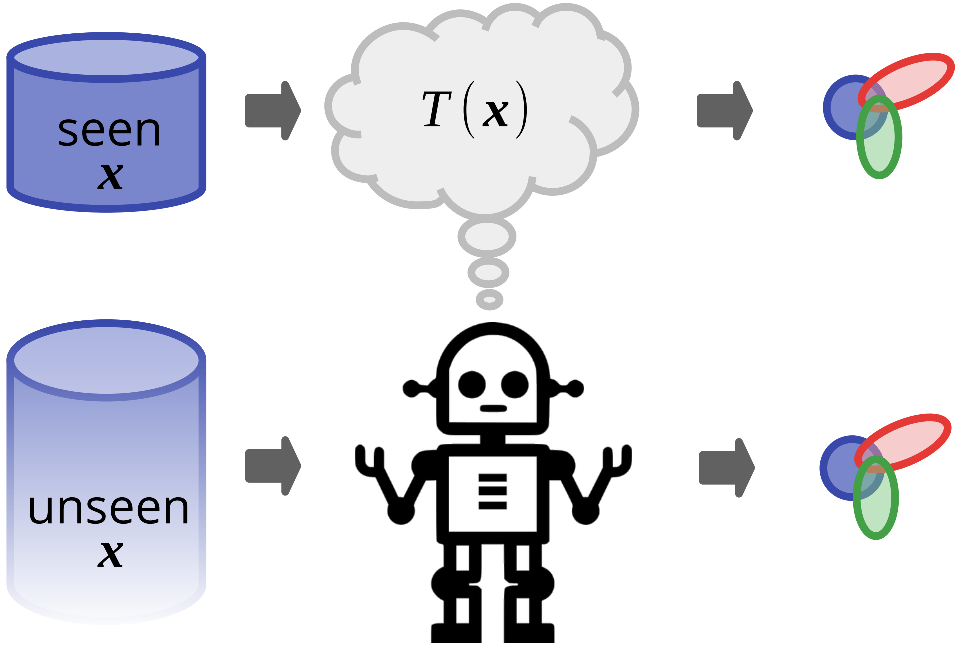
General goal for unsupervised problems:
Find a transformation (T) that builds a compact internal representation of unlabeled data ($\boldsymbol{x}$) to unveil its internal structure.

Traditional Approach:



Input data ($\boldsymbol{x}$)

Machine-Learning Approach:

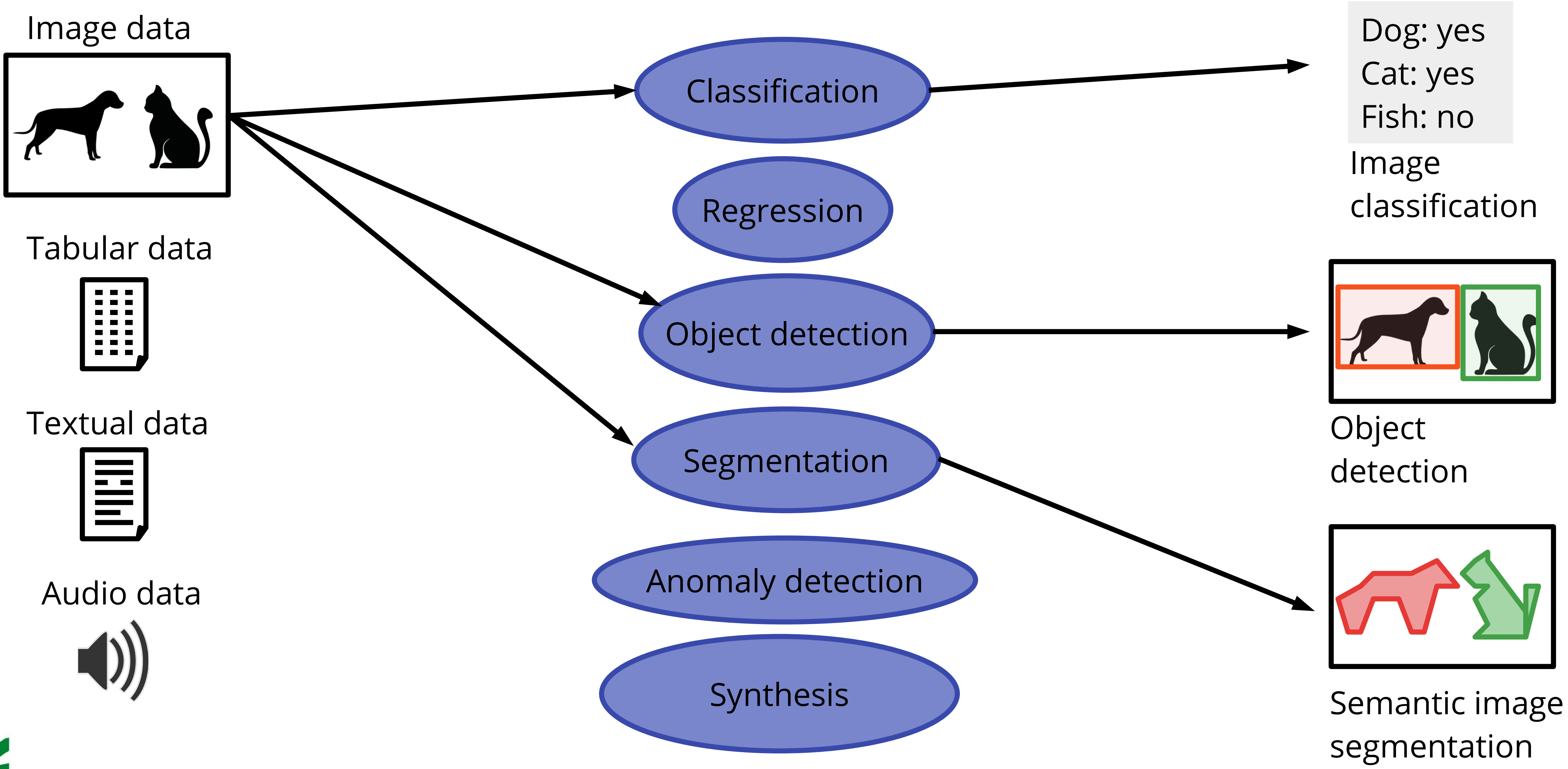


Input data ($\boldsymbol{x}$)

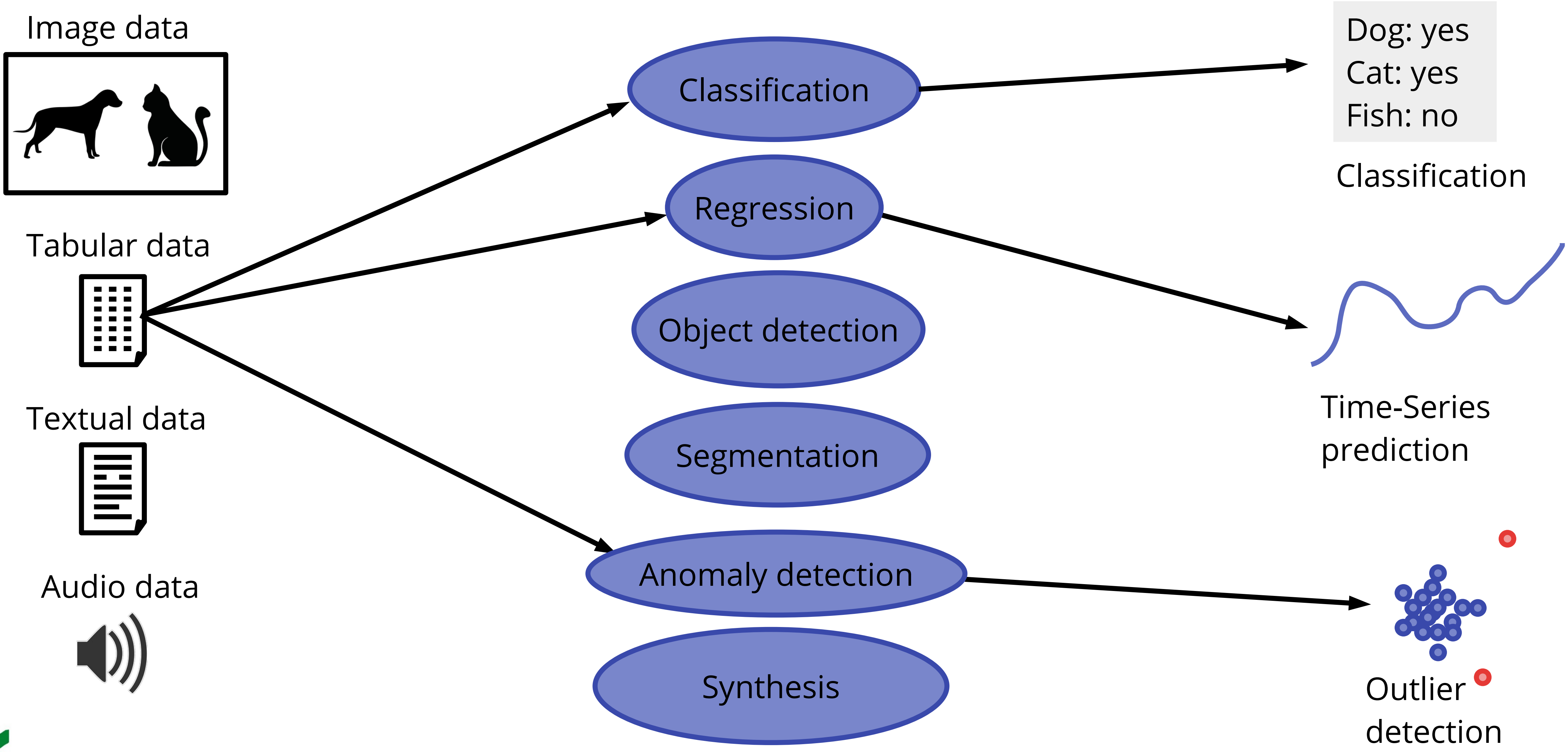
What tasks can ML learn?

- Classification
- Regression
- Object detection
- Segmentation
- Anomaly detection
- Synthesis

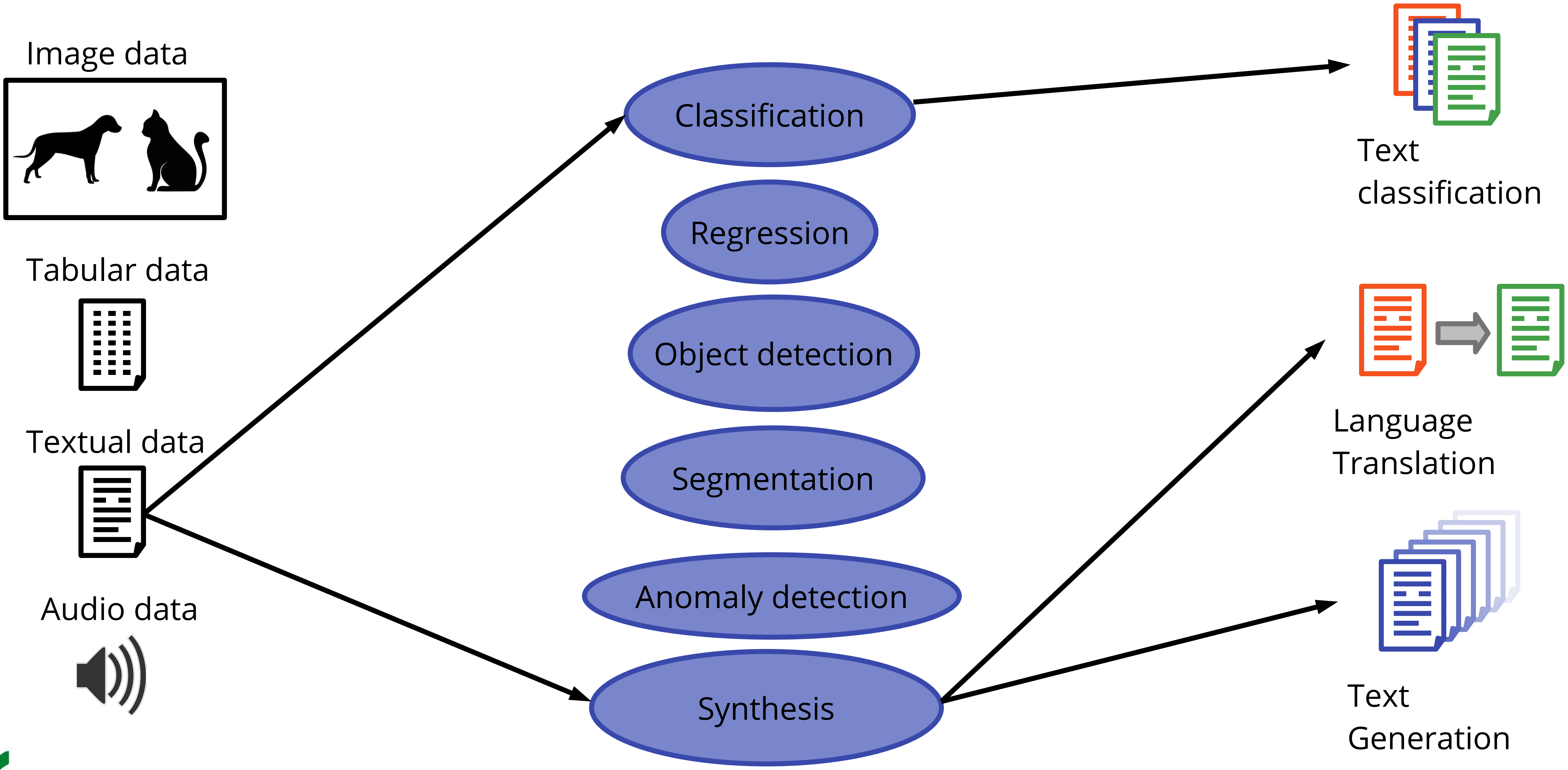
What tasks can ML learn? That depends on the data modality...



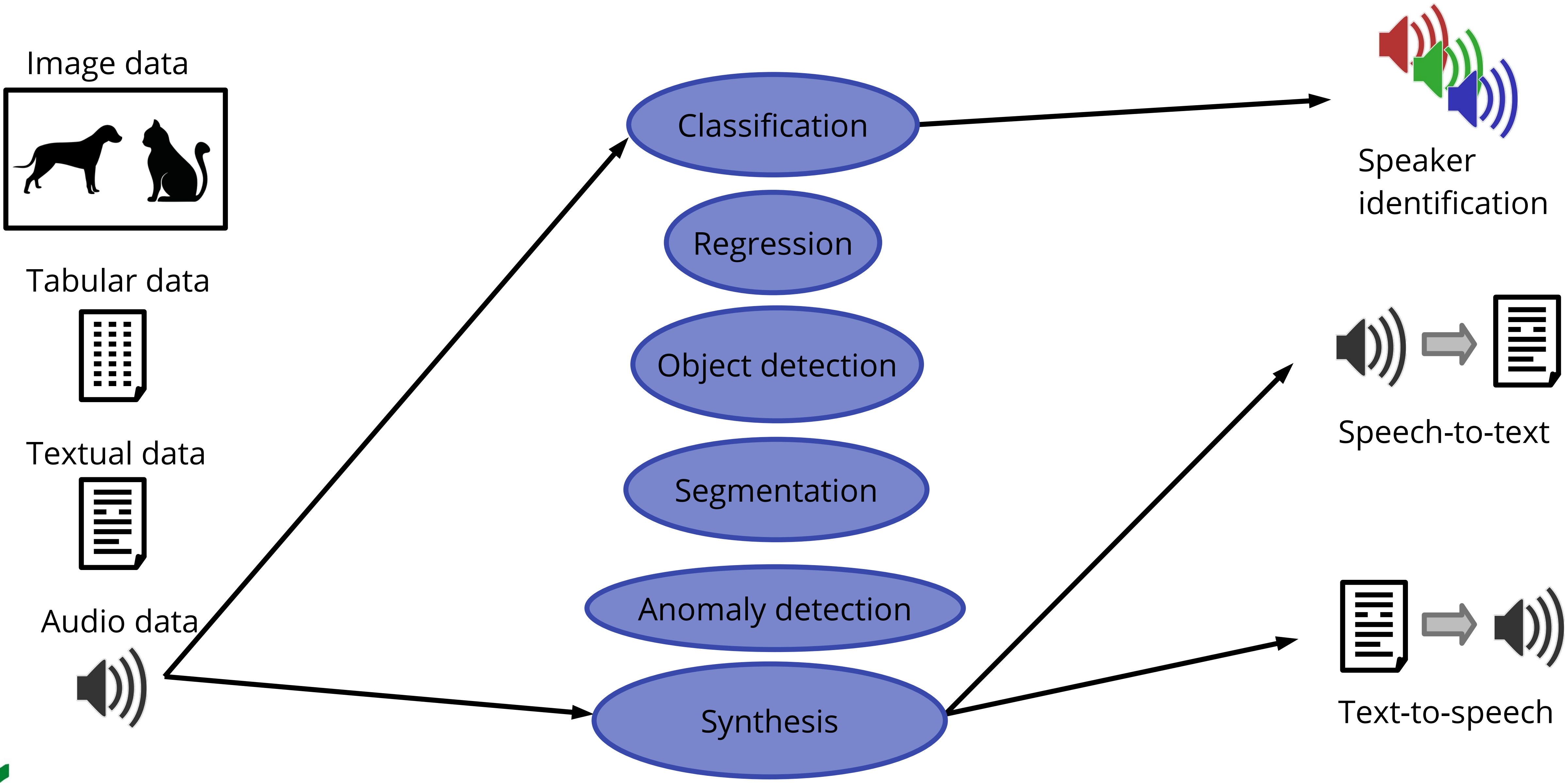
What tasks can ML learn? That depends on the data modality...



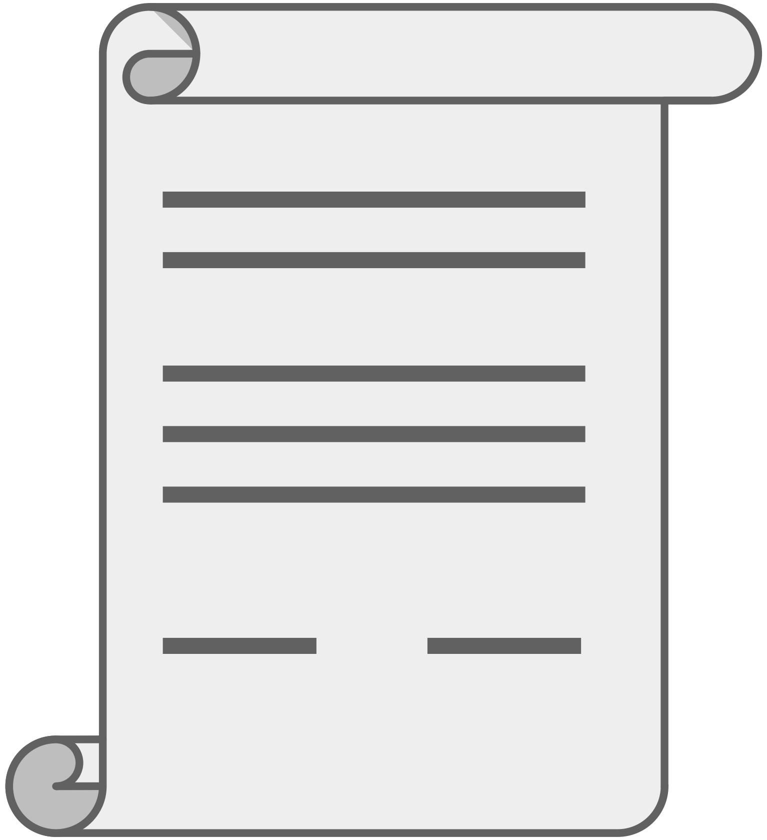
What tasks can ML learn? That depends on the data modality...



What tasks can ML learn? That depends on the data modality…



Course modalities



Course modalities

- **Goal** of this course:
  To understand and be able to implement and utilize traditional Machine Learning and Deep Learning models.
- **Requirements**: Math/statistics, Python programming (successful participation in *Fundamentals of Computer Science* course), English language skills
- **Setup**: Combination of lectures and hands-on lab courses + (3) home-work coding assignments
- **Lab courses**:
  Python, Jupyter Notebooks, Pytorch, scikit-learn, Google Colab

Lab courses

- Lab courses provide practical examples focused on lecture topics and they can be used to discuss questions from the lecture
- Code will be provided (and used) in the form of **Jupyter Notebooks**.
- We will use **Google Colab** for running our Notebooks (they offer free GPUs!). If you don't have a Google account, please let me know as soon as possible!
- All code elements from the lab courses can be used for your coding assignments.

Grading

- **Exam (70%)**:
  - Written decentralized examination (90 min): **22 May 2023, 12:15-14:00**, room 01-114
  - In general, **all lecture slides and lab course materials** are relevant for the exam (unless specifically excluded); no additional literature (e.g., books) is required for the exam; you will not have to write code, but may have to interpret code snippets
  - Exam goes beyond simply memorizing the slides: you have to understand the content
- **3 Coding Assignments (30%)**:
  - Related to supervised learning (with traditional ML), unsupervised learning and Deep Learning
  - Assignments in the form of Jupyter Notebooks that must be submitted before the deadline
  - Due dates: see course syllabus

Coding assignments

- **Coding assignment grades** will be based on the following aspects (in this order):
  - **Code implementation** (clean and well-structured code, inline comments where useful)
  - **Documentation** (explanations as markdown cells, analysis of the results, plots)
  - **Approach** (how novel is the approach compared to what was introduced in the corresponding lab course?)
  - **Results** (how good are the results of the method?)

Will it hurt?

If you want to use ML, you have to understand ML...

$$S(i,j) = (I*K)(i,j) = \sum_m \sum_n I(m,n) K(i-m, j-n)$$

$$\sigma(\mathbf{z})_i = \frac{\exp(z_i)}{\sum_{j=1}^{K} \exp(z_j)}$$

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip} + \epsilon$$

```
[ ]  class CIFAR10Net(nn.Module):

         def __init__(self):
             # call super class constructor
             super(CIFAR10Net, self).__init__()

             self.conv1 = nn.Conv2d(in_channels=3, out_channels=6, kernel_size=5, stride=1, padding=0)
             self.pool1 = nn.MaxPool2d(kernel_size=2, stride=2)
             self.conv2 = nn.Conv2d(in_channels=6, out_channels=16, kernel_size=5, stride=1, padding=0)
             self.pool2 = nn.MaxPool2d(kernel_size=2, stride=2)

             self.linear1 = nn.Linear(16 * 5 * 5, 120, bias=True)
             self.relu1 = nn.ReLU(inplace=True)
             self.linear2 = nn.Linear(120, 84, bias=True)
             self.relu2 = nn.ReLU(inplace=True)
             self.linear3 = nn.Linear(84, 10)
             self.logsoftmax = nn.LogSoftmax(dim=1)

         # define network forward pass
         def forward(self, images):
             x = self.pool1(self.conv1(images))
             x = self.pool2(self.conv2(x))
             x = x.view(-1, 16 * 5 * 5)
             x = self.relu1(self.linear1(x))
             x = self.relu2(self.linear2(x))
             x = self.logsoftmax(self.linear3(x))
             return x
```

Slides

- Slides will be uploaded to Canvas **before each lecture**.
- Slides should be **self-explanatory**; your final exam will be based on the slide deck content (but goes beyond simply memorizing facts)
- I am using a few symbols that will guide you through the slides:

There is a **Jupyter notebook** available with some related code.

This content will be discussed in more detail **in the future**.

Content in an orange dotted box is generally a bit more complex and only shown for completeness. For the exam you should be **aware of its existence** and be able to **describe it roughly**, but you will not have to memorize every detail.

Github, Canvas and Zoom

Github:

- Coding resources are stored at github.com/HSG-AIML-Teaching/MLBBWL-2023FS

Canvas:

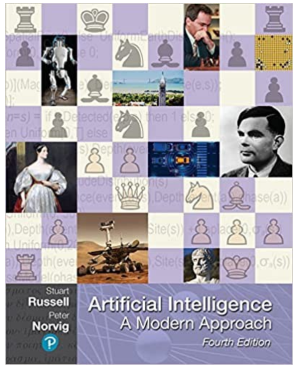
- All resources (slides, lab notebooks) will be accessible through Canvas
- Feel free to utilize the Canvas discussion board to ask questions on lectures and lab sessions.

Zoom:

- By all default, **lectures and lab courses are in-person** (unless the university decides otherwise)
- **Hybrid format** (in-person + zoom) or recordings are only available for select students under **special circumstances** (sickness, unforeseen family situation, military service, etc.)
- Job-related collisions or other courses taking place at the same time do **not** count as special circumstances!

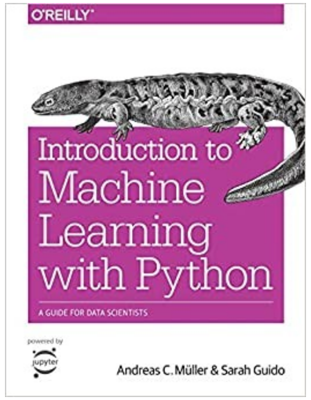
Literature resources

- Stuart Russell, Peter Norvig: **Artificial Intelligence: A Modern Approach** (2020 and earlier versions, MIT Press)
  Part V ("Learning") is especially relevant to this course and provides good introductions

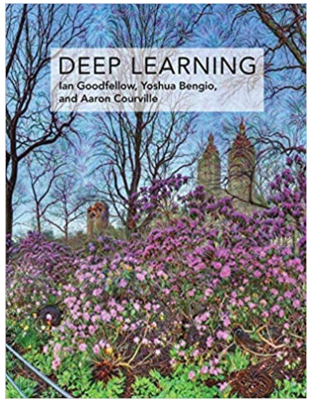


ebook@HSG

- Andreas Müller & Sarah Guido: **Introduction to Machine Learning with Python** (2017, O'Reilly)
  Easy-to-understand introduction to Python for ML, uses scikit-learn



- Ian Goodfellow, Yoshua Bengio, Aaron Courville: **Deep Learning** (2016, MIT Press)
  All you need to know about Deep Learning



free online

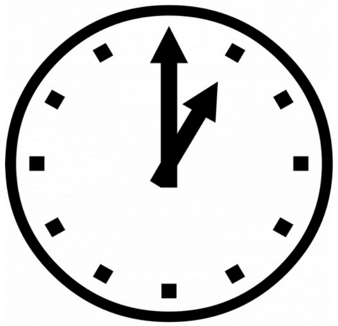
How (and when) to reach me?

- In person:
  - before/after the lecture/lab course
  - at ICS (please make an appointment via email first)

- Via Email: michael.mommert@unisg.ch
  (please start the header with **MLBBWL:** so I immediately see that the email is related to this lecture)

- One personal note: Please be aware that I am generally unable to reply to emails outside office hours.

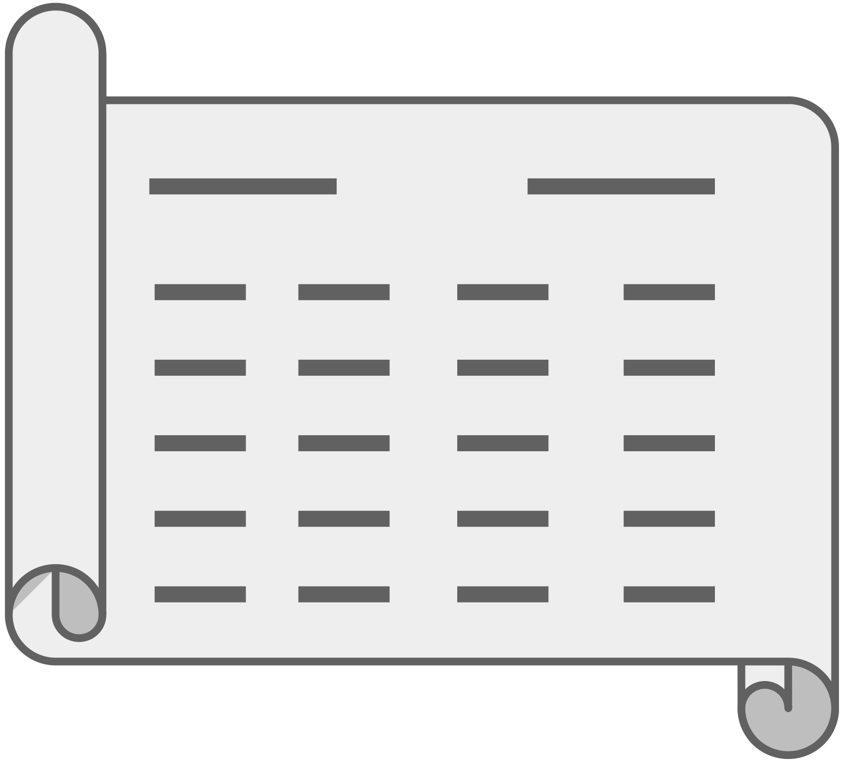
osm.org



Questions?

?

Course syllabus



Course Syllabus

Date	Lecture	Lab Course
20 Feb	Intro	Prep: Python Intro + Numpy
27 Feb	Data and Features	-
6 Mar	Supervised Learning	-
13 Mar	-	Supervised Learning
20 Mar	Unsupervised Learning	-
27 Mar	-	Unsupervised Learning
Spring Break		
17 Apr	Neural Networks	-
24 Apr	-	Neural Networks
1 May	CNNs and Computer Vision	-
8 May	-	CNNs
15 May	Deep Learning & Ethics	-
22 May	**Exam!**	

Assignment 1

Assignment 2

Assignment 3

Course Syllabus

Date	Lecture	Lab Course
20 Feb	Intro	Prep: Python Intro + Numpy
27 Feb	Data and Features	-
6 Mar	Supervise earning	-
13 Mar	-	Supervised Learning
20 Mar	Unsupervis	
27 Mar		arning
17 Apr	Neu	
24 Apr		rks
1 May	CNNs and	
8 May	-	CNNs
15 May	Deep Learning & Ethics	-
22 May	**Exam!**	

- Data types
- Features and feature engineering
- Data scaling

Assignment 1

Assignment 2

Assignment 3

Course Syllabus

Date	Lecture	Lab Course
20 Feb	Intro	Prep: Python Intro + Numpy
27 Feb	Data and Features	-
6 Mar	Supervised Learning	-
13 Mar	-	Supervised Learning
20 Mar	Unsupervised Lea	-
27 Mar	-	ning
17 Apr	Neural N	
24 Apr		
1 May	CNNs and Co	
8 May	-	
15 May	Deep Learning & Ethics	-
22 May	**Exam!**	

- Supervised learning setup and concepts
- Benchmarking and metrics
- Linear models
- Nearest neighbor models
- Tree-based models

Assignment 1

Assignment 2

Assignment 3

Course Syllabus

Date	Lecture	Lab Course
20 Feb	Intro	Prep: Python Intro + Numpy
27 Feb	Data and Features	-
6 Mar	Supervised Learning	-
13 Mar	-	Supervised Learning
20 Mar	Unsupervised Learning	-
27 Mar	-	Unsupervised Learning
17 Apr		-
24 Apr		ural Networks
1 May		-
8 May		CNNs
15 May		-
22 May	Exam!	

Assignment 1

Assignment 2

Assignment 3

- Supervised learning with scikit-learn
- Iris dataset
- K-NN
- Fashion-MNIST dataset

Course Syllabus

Date	Lecture	Lab Course
20 Feb	Intro	Prep: Python Intro + Numpy
27 Feb	Data and Features	-
6 Mar	Supervised Learning	-
13 Mar	-	Supervised Learning
20 Mar	Unsupervised Learning	-
27 Mar	-	Unsupervised Learning
	Spring Brea	
17 Apr	Neural Network	
24 Apr	-	
1 May	CNNs and Computer	
8 May	-	
15 May	Deep Learning & E	
22 May	**Exam:**	

- Unsupervised learning setup
- Hierarchical clustering
- K-means clustering
- Expectation Maximization Clustering
- Principal component analysis

Assignment 1

Assignment 2

Assignment 3

Course Syllabus

Date	Lecture	Lab Course
20 Feb	Intro	Prep: Python Intro + Numpy
27 Feb	Data and Features	-
6 Mar	Supervised Learning	-
13 Mar	-	Supervised Learning
20 Mar	Unsupervised Learning	-
27 Mar	-	Unsupervised Learning
	Spring Bre	
17 Apr		-
24 Apr		Neural Networks
1 May		-
8 May		CNNs
15 May		-
22 May	Exam.	

- Unsupervised learning with scikit-learn
- k-means
- Agglomerative clustering
- PCA

Assignment 1

Assignment 2

Assignment 3

Course Syllabus

Date	Lecture	Lab Course
20 Feb	In...	...umpy
27 Feb	Data and...	
6 Mar	Supervise...	
13 Mar		...g
20 Mar	Unsupervis...	
27 Mar	-	Unsupervised Learning
	S... g Break	
17 Apr	Neural Networks	-
24 Apr	-	Neural Networks
1 May	CNNs and Computer Vision	-
8 May	-	CNNs
15 May	Deep Learning & Ethics	-
22 May	**Exam!**	

- Perceptron and neurons
- Activation functions
- Loss functions
- Backpropagation
- Multilayer Perceptron

Assignment 1

Assignment 2

Assignment 3

Course Syllabus

Date	Lecture	Lab Course
20 Feb	Intro	Prep: Python Intro + Numpy
27 Feb		-
6 Mar		-
13 Mar		rvised Learning
20 Mar		-
27 Mar		ervised Learning
	Spring Bre	
17 Apr	Neural Networks	-
24 Apr	-	Neural Networks
1 May	CNNs and Computer Vision	-
8 May	-	CNNs
15 May	Deep Learning & Ethics	-
22 May	**Exam!**	

- MLP with Pytorch
- Image classification with MLP

Assignment 1

Assignment 2

Assignment 3

Course Syllabus

Date	Lecture	Lab Course
20 Feb	Intro	Prep: Python Intro + Numpy
27 Feb	Data and Features	-
6 Mar	Super	
13 Mar		rning
20 Mar	Unsupe	
27 Mar		arning
17 Apr	Neural Netw	-
24 Apr	-	Neural Networks
1 May	CNNs and Computer Vision	-
8 May	-	CNNs
15 May	Deep Learning & Ethics	-
22 May	**Exam!**	

- Convolutional neural networks
- Semantic segmentation
- Object detection

Assignment 1

Assignment 2

Assignment 3

Course Syllabus

Date	Lecture	Lab Course
20 Feb	Intro	Prep: Python Intro + Numpy
27 Feb	Data and Features	-
6 Mar	Supervised Learning	-
13 Mar		ed Learning
20 Mar	U	-
27 Mar		ised Learning
17 Apr		-
24 Apr		eural Networks
1 May	CNNs and Computer Vision	-
8 May	-	CNNs
15 May	Deep Learning & Ethics	-
22 May	**Exam!**	

- Convolutional neural networks with PyTorch
- Image classification

Assignment 1

Assignment 2

Assignment 3

Course Syllabus

Date	Lecture	Lab Course
20 Feb	Intro	Prep: Python Intro + Numpy
27 Feb	Data and Features	-
6 Mar	Supervised Learning	-
13 Mar	-	Supervised Learning
20 Mar	Unsupervised L	
27 Mar	-	
17 Apr	Neural Netw	
24 Apr	-	
1 May	CNNs and Comput	
8 May	-	CNNs
15 May	Deep Learning & Ethics	-
22 May	**Exam!**	

- How to train large models
- Different learning approaches
- Ethics in AI

Assignment 1

Assignment 2

Assignment 3

Course Syllabus

Date	Lecture	Lab Course
20 Feb	Intro	Prep: Python Intro + Numpy
27 Feb	Data and Features	-
6 Mar	Supervised Learning	-
13 Mar	-	Supervised Learning
20 Mar	Unsupervised Learning	-
27 Mar	-	Unsupervised Learning
	Spring Break	
17 Apr	Neural Networks	-
24 Apr	-	Neural Networks
1 May	CNNs and Computer Vision	-
8 May	-	CNNs
15 May	Deep Learning & Ethics	-
22 May	**Exam!**	

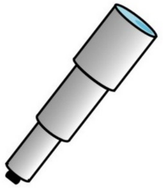
Assignment 1

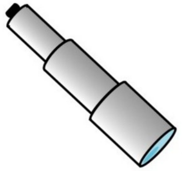
Assignment 2

Assignment 3

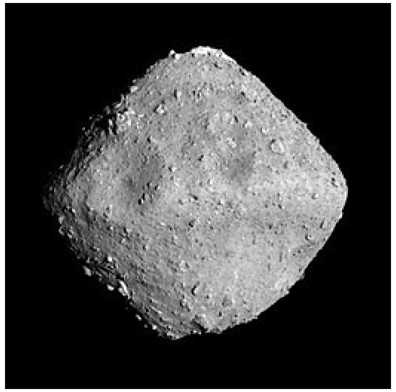
About myself

About myself

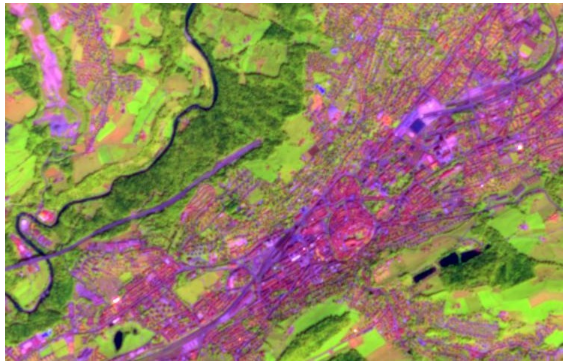


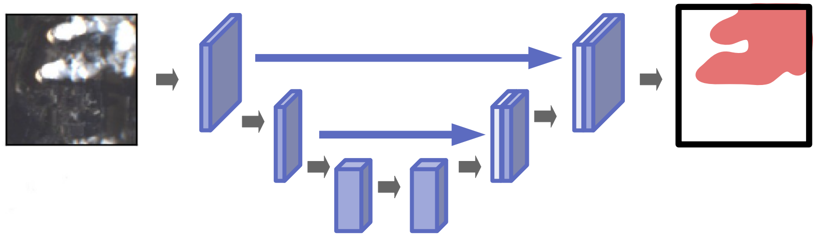


ISAS/JAXA



Gerald Rhemann





Universität Heidelberg – Zukunft seit 1386

DLR – Deutsches Zentrum für Luft- und Raumfahrt – German Aerospace Center

Freie Universität Berlin

NAU – Northern Arizona University

Lowell Observatory – 125 Years | 1894 - 2019

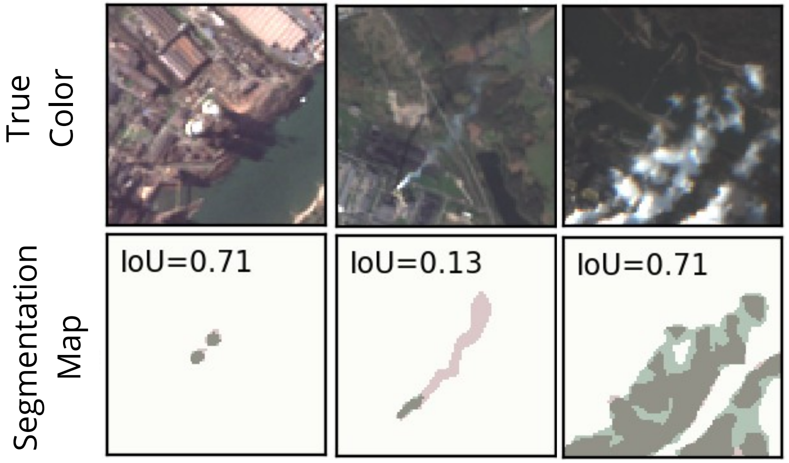
Universität St.Gallen

2009	Physics
2013	Dr. rer. nat. (Earth Sciences)
2020	Postdoc @HSG-AIML
2022	Asst. Prof. Computer Vision

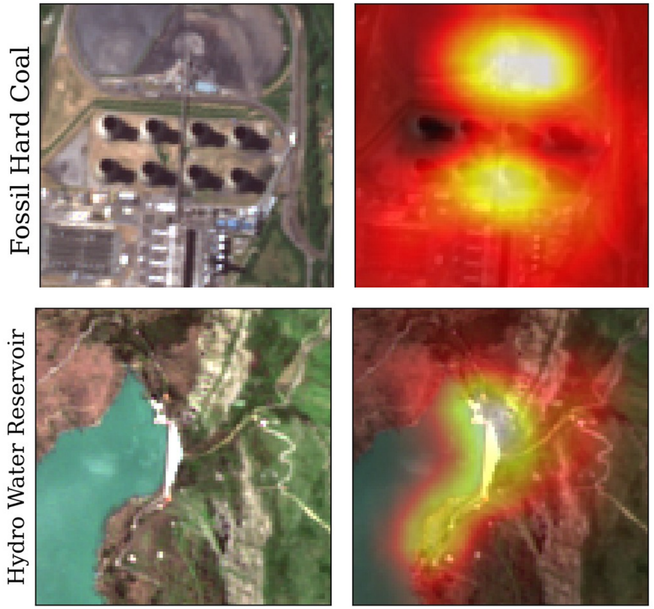
What I work on...

Commercial Vehicle Traffic Monitoring
(*Blattner et al. 2021*)

Characterization of Industrial Smoke Plumes from Remote Sensing Data
(*Mommert et al. 2020, Hanna et al. 2021*)

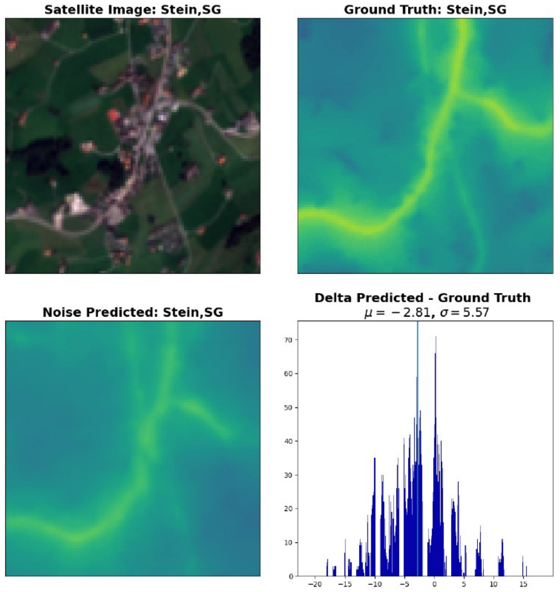


R: ground-truth, G: prediction



Power Plant Classification from Remote Imaging with Deep Learning
(*Mommert et al. 2021*)

Road Traffic Noise Estimation from Remote Imaging Data
(*Eicher et al. submitted*)

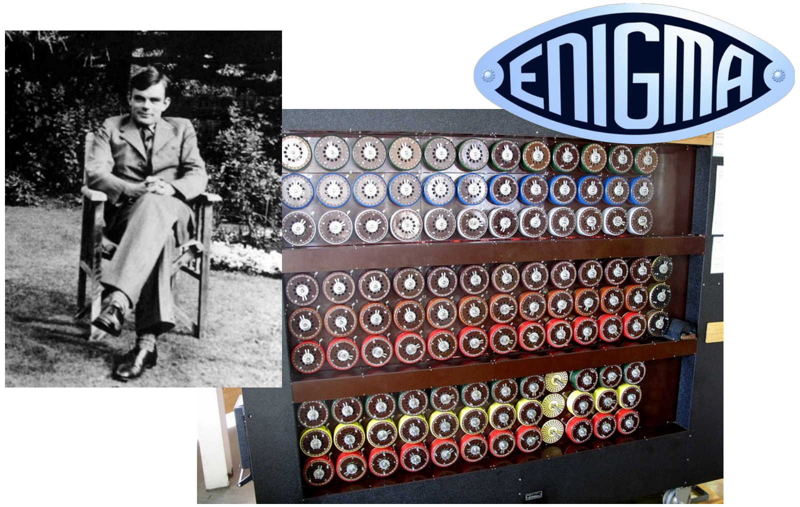


Looking for a Bachelor thesis topic?
→ hsg.ai

How did we get here?

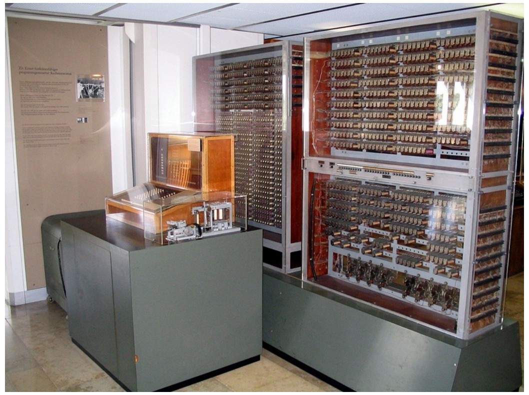
jcstudio

A little history



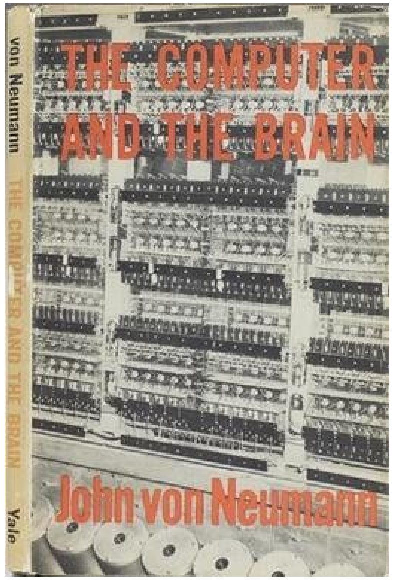
TedColes @ wikipedia

Alan Turing’s work



Venusianer @ wikipedia

1941: Z3 (first digital computer)



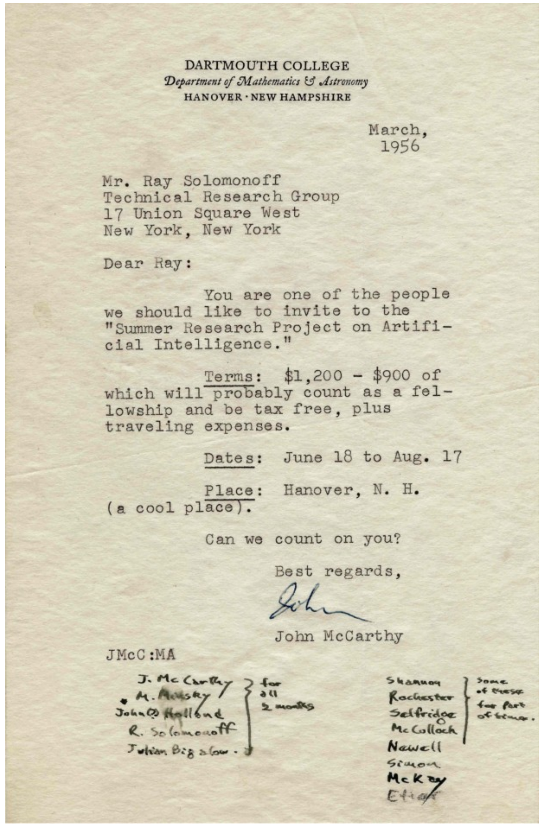
Computer science
+ Neuroscience
= “Cybernetics”

Mathematics: λ
- Logic
- Information Theory

1950s

1956: Dartmouth Workshop

- 6-week workshop of leading researchers: Minsky, McCarthy, Shannon, Rochester…
- Birth of the term “Artificial Intelligence”
- Objective of AI: *To write a calculator program which can solve intellectual problems as well as or better than a human being.*



DARTMOUTH COLLEGE
Department of Mathematics & Astronomy
HANOVER · NEW HAMPSHIRE

March,
1956

Mr. Ray Solomonoff
Technical Research Group
17 Union Square West
New York, New York

Dear Ray:

You are one of the people we should like to invite to the "Summer Research Project on Artificial Intelligence."

Terms: $1,200 - $900 of which will probably count as a fellowship and be tax free, plus traveling expenses.

Dates: June 18 to Aug. 17

Place: Hanover, N. H. (a cool place).

Can we count on you?

Best regards,

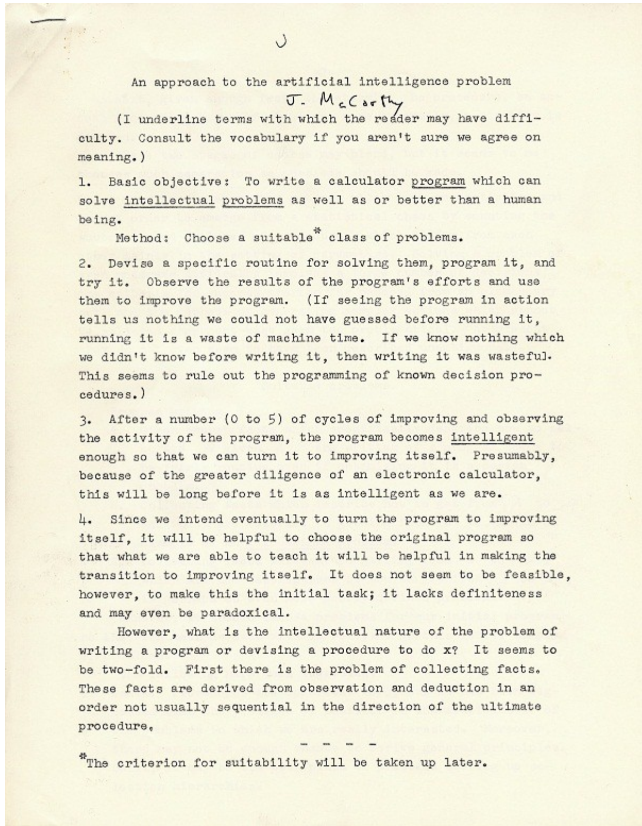
John

John McCarthy

JMcC:MA

J. McCarthy
M. Minsky
John Holland
R. Solomonoff
Julian Bigelow
for all 2 months

Shannon
Rochester
Selfridge
McCulloch
Newell
Simon
McKay
Some of these for part of time.



An approach to the artificial intelligence problem

J. McCarthy

(I underline terms with which the reader may have difficulty. Consult the vocabulary if you aren't sure we agree on meaning.)

1. Basic objective: To write a calculator program which can solve intellectual problems as well as or better than a human being.
   Method: Choose a suitable* class of problems.

2. Devise a specific routine for solving them, program it, and try it. Observe the results of the program's efforts and use them to improve the program. (If seeing the program in action tells us nothing we could not have guessed before running it, running it is a waste of machine time. If we know nothing which we didn't know before writing it, then writing it was wasteful. This seems to rule out the programming of known decision procedures.)

3. After a number (0 to 5) of cycles of improving and observing the activity of the program, the program becomes intelligent enough so that we can turn it to improving itself. Presumably, because of the greater diligence of an electronic calculator, this will be long before it is as intelligent as we are.

4. Since we intend eventually to turn the program to improving itself, it will be helpful to choose the original program so that what we are able to teach it will be helpful in making the transition to improving itself. It does not seem to be feasible, however, to make this the initial task; it lacks definiteness and may even be paradoxical.

However, what is the intellectual nature of the problem of writing a program or devising a procedure to do x? It seems to be two-fold. First there is the problem of collecting facts. These facts are derived from observation and deduction in an order not usually sequential in the direction of the ultimate procedure.

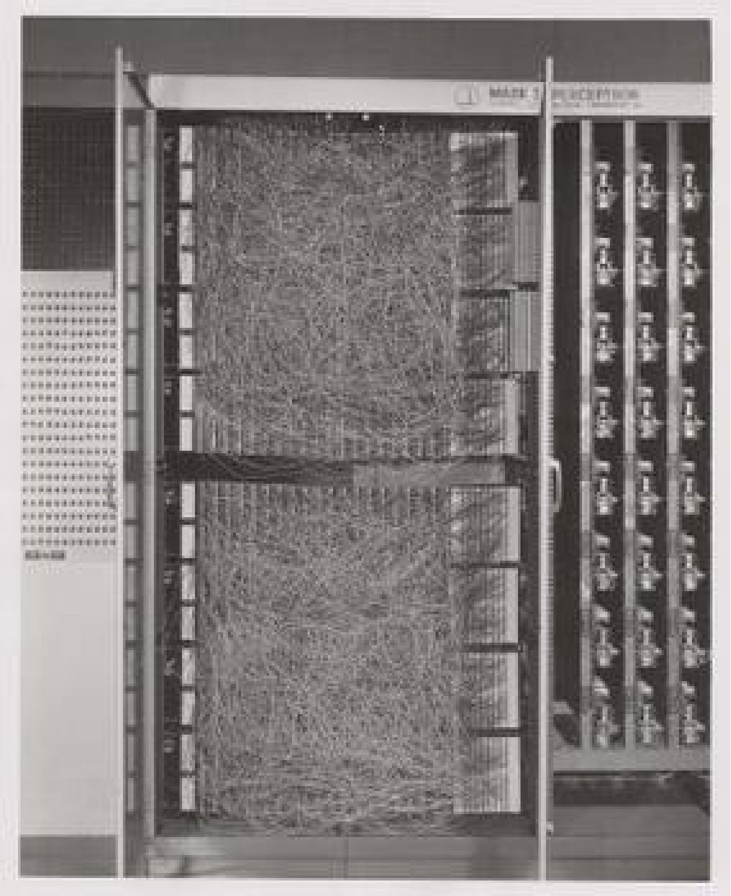
\- - - -

*The criterion for suitability will be taken up later.

1956

Connectionism:
Use of Artificial Neural Networks (ANNs) as function approximators

1958: The Perceptron
First ANN implementation (as a physical device with manually tunable ”weights”); able to solve many problems.

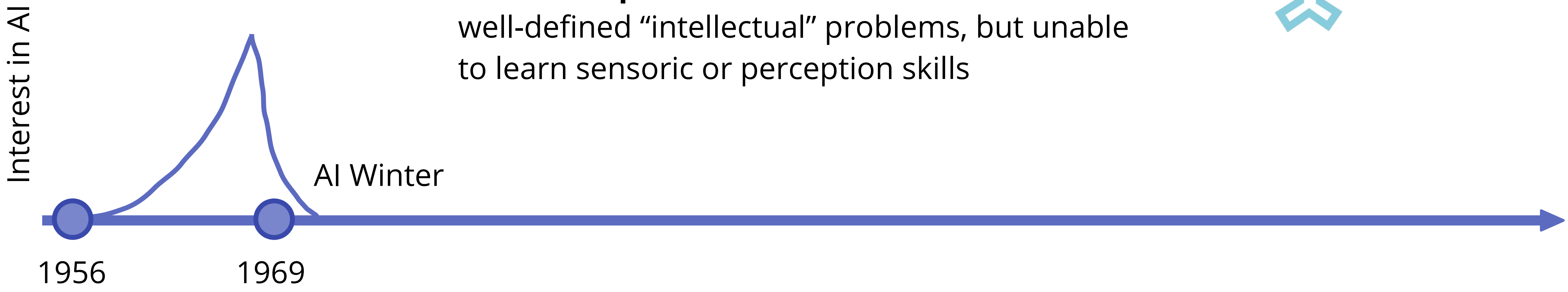

Cornell University Library

A little history of AI

1969: *Perceptrons* book (Minsky and Papert):
Major limitations of ANNs revealed (cannot approximate XOR function)

AI Winter:

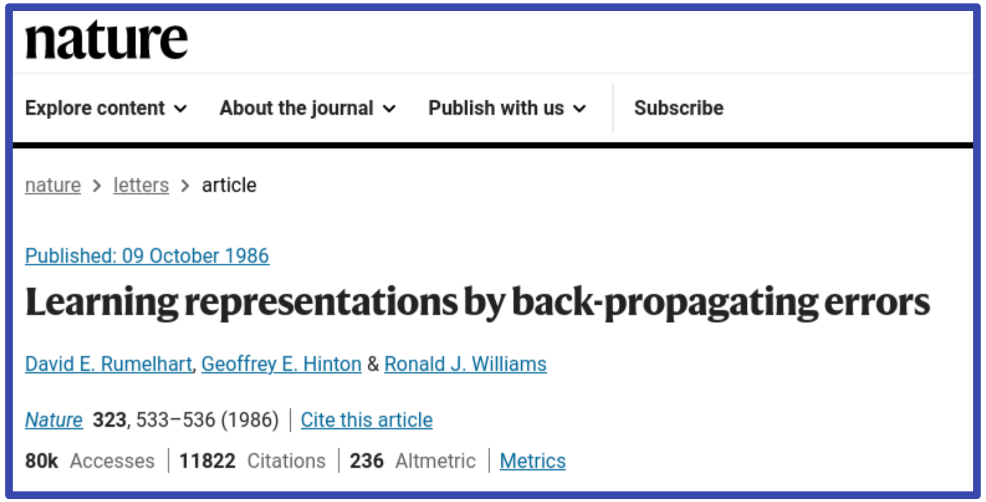
- High expectations of AI not met
- Lack of computational resources
- Limitations of ANNs become clear:
  Moravec's paradoxon: AI able to solve well-defined "intellectual" problems, but unable to learn sensoric or perception skills



A little history of AI

1980s: success of **expert systems** (utilize pre-programmed domain knowledge to define rules to solve tasks)

First commercial wave of AI only short-lived: limited applicability and inability to learn = no commercial success


nature

Explore content ⌄ About the journal ⌄ Publish with us ⌄ Subscribe

nature > letters > article

Published: 09 October 1986

Learning representations by back-propagating errors

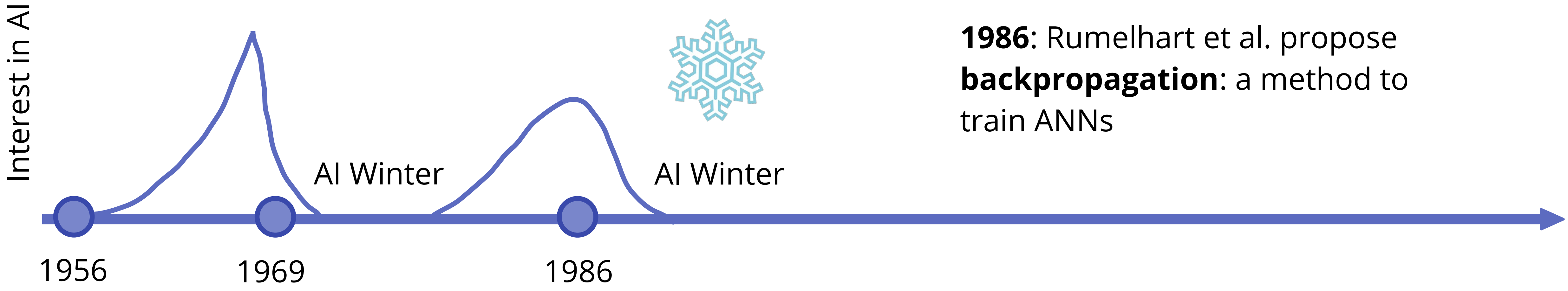
David E. Rumelhart, Geoffrey E. Hinton & Ronald J. Williams

Nature **323**, 533–536 (1986) | Cite this article

80k Accesses | **11822** Citations | **236** Altmetric | Metrics

Nature

1986: Rumelhart et al. propose **backpropagation**: a method to train ANNs



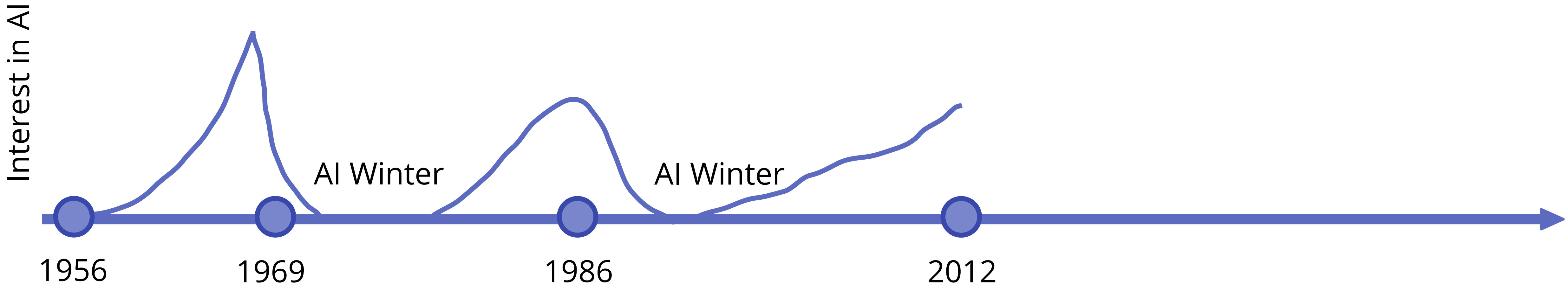
A little history of AI

James the Photographer @ wikipedia

1990s and 2000s: slow progress due to improved computational resources

- DeepBlue beats Gary Kasparov
- Watson defeats Jeopardy champions

Jeopardy (2011)

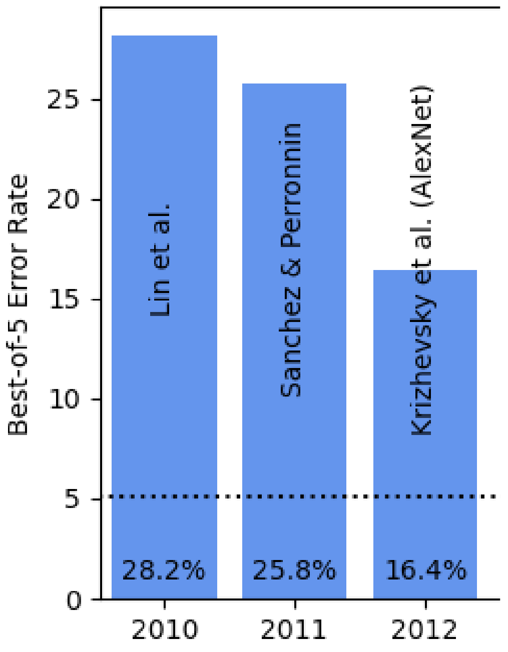


A little history of AI

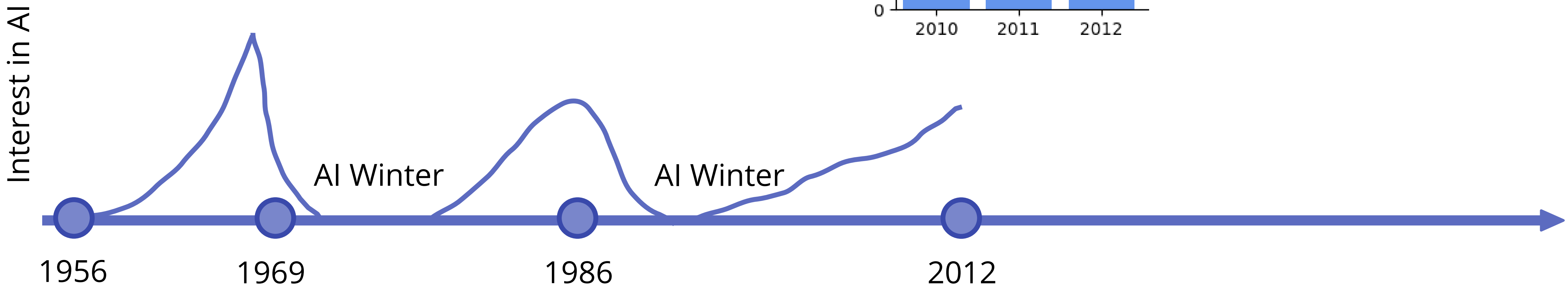
2012: **AlexNet** (Krizhevsky et al. 2012) Convolutional Neural Network trained on two GPUs with backpropagation beats all challengers on the ImageNet Challenge with a wide margin.

IMAGENET

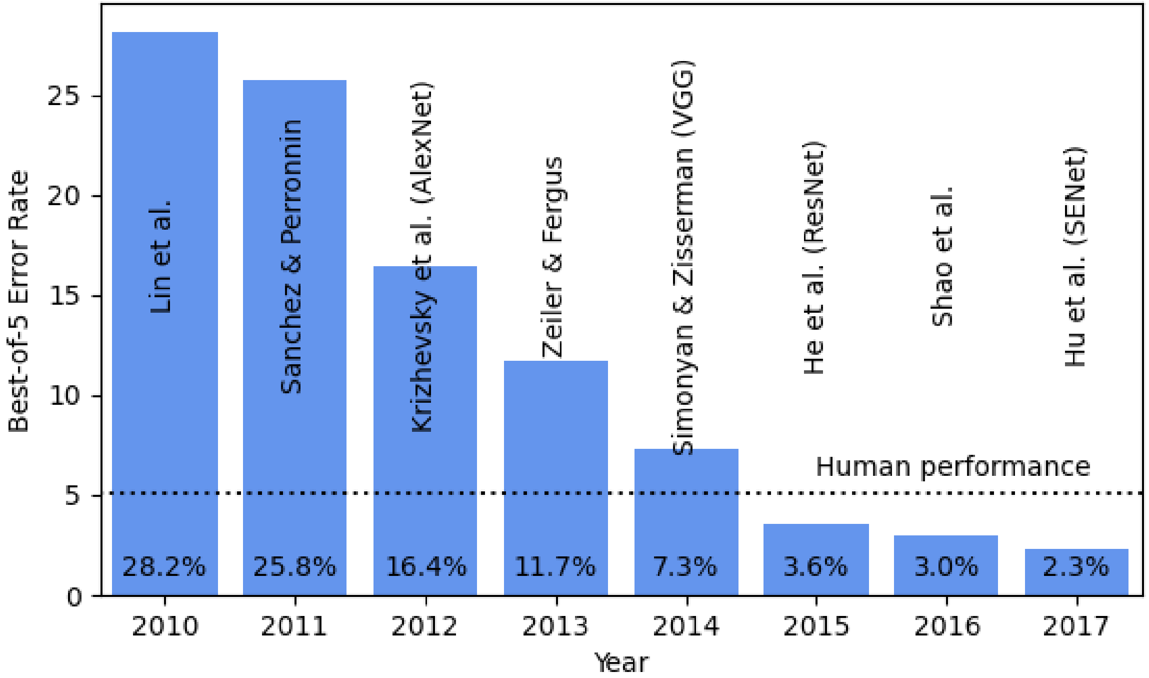
(150,000 images in 1,000 categories)



• • •

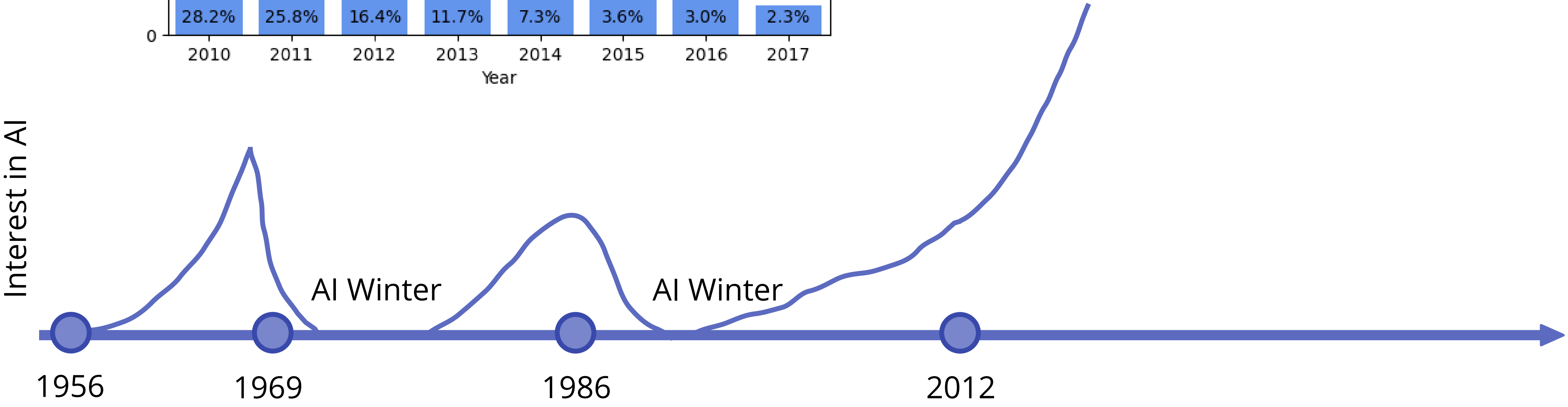


A little history of AI

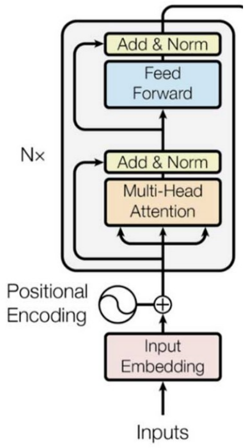


2010s: Deep Learning
Deep Neural Networks outperform almost all other ML methods based on two factors:

- Training on GPUs is highly efficient
- Vast amounts of data is available to train and validate these models



A little history of AI



Vaswani et al. (2017)

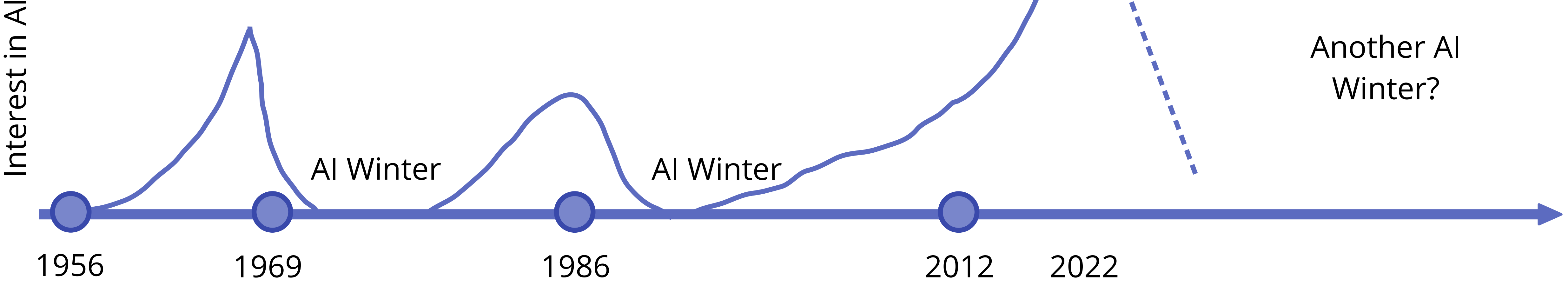
Large-scale language models based on Attention/ Transformer architectures

General Artificial Intelligence (GAI)

Autonomous driving

2020s+: What's next?

Another AI Winter?



That’s all folks!

Today's lecture

1- Introduction

What this course is about...

Course modalities

Course syllabus

About myself

How did we get here?

Next lecture (27th Feb)

2 – Data and Features

Types of Data

Features and feature engineering

Data scaling