Zurich University
of Applied Sciences

# Artificial Intelligence
# V08: Learning Agents

Introduction to supervised machine learning
Decision trees
Doing machine learning

Based on material by
- Stuart Russell, UC Berkeley
- Andreas Krause, ETH Zurich

Zurich University of Applied Sciences and Arts
InIT Institute of  Applied Information Technology (stdm)

# Educational objectives

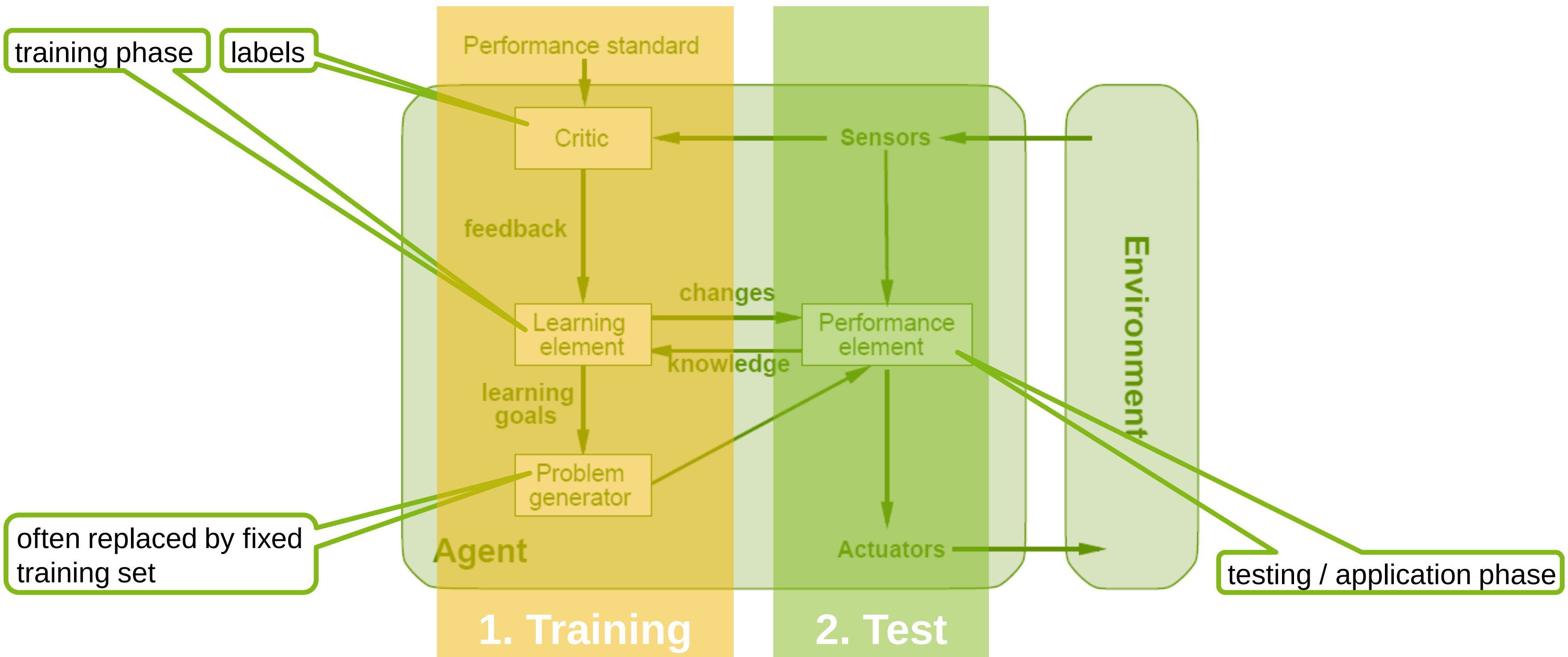
- **Remember** the basic **decision tree training algorithm**
- **Explain machine learning using** the correct **technical terms**
- **Defend** your **own view on** the existence of good **general learners**
- **Build decision tree**-based **models** for labeled data sets **using** the **ML development process**

*In which we describe agents that can improve their behaviour through diligent study of their own experiences.*

➔ Reading: AIMA, ch. 18-18.6

Zurich University
of Applied Sciences

# 1. INTRODUCTION TO SUPERVISED MACHINE LEARNING



Zurich University
of Applied Sciences

# The discipline of machine learning – mapped

*«…gives computers the ability to learn **without being explicitly programmed**.»*
A. Samuel, 1959

**ML**

Famous: used in most **human** learning, definition of **scientific method**

**Types of:**

- inductive (example-based)
- transductive (example → example)
- deductive (logic-based)

**Subtypes of:**

- supervised (learn concepts / predict values)
- reinforcement (learn to act)
- unsupervised (find structure)

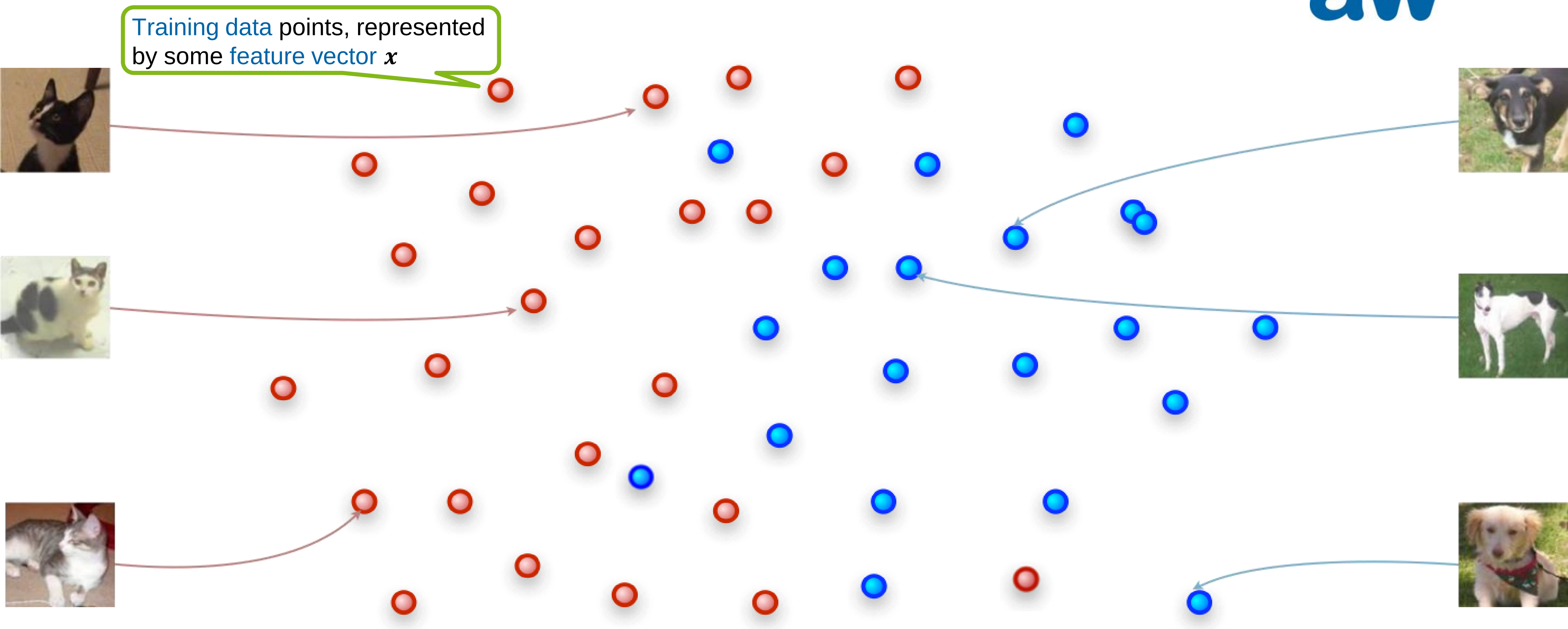
⋮


**Search** through **hypothesis space**, driven by **theory** and **empiricism**

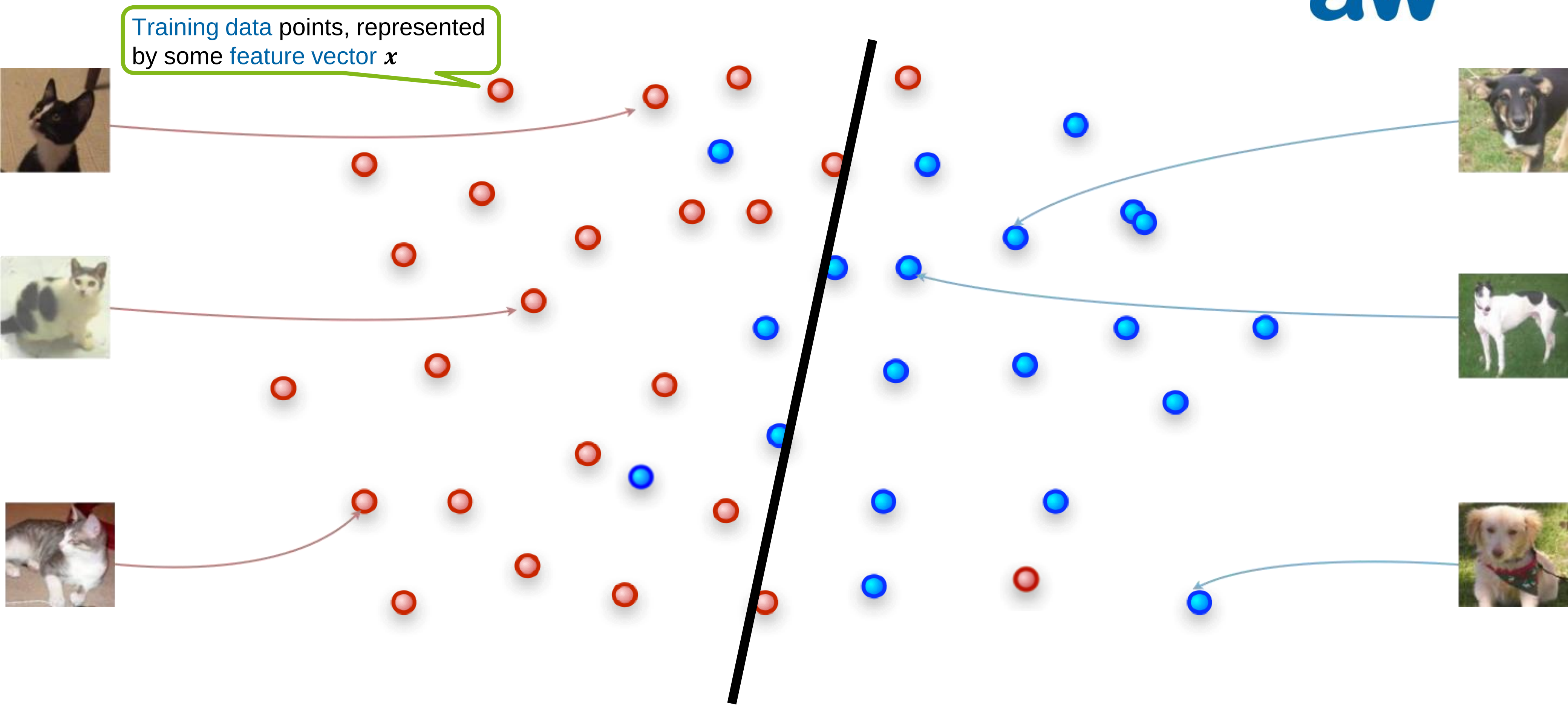
**Models:** … linear / non-linear | parametric / non-parametric | discriminative / generative | predictive / inferential | deep / shallow …

- fixed size / growing with data
- learn boundary / blueprint
- black box / explanatory

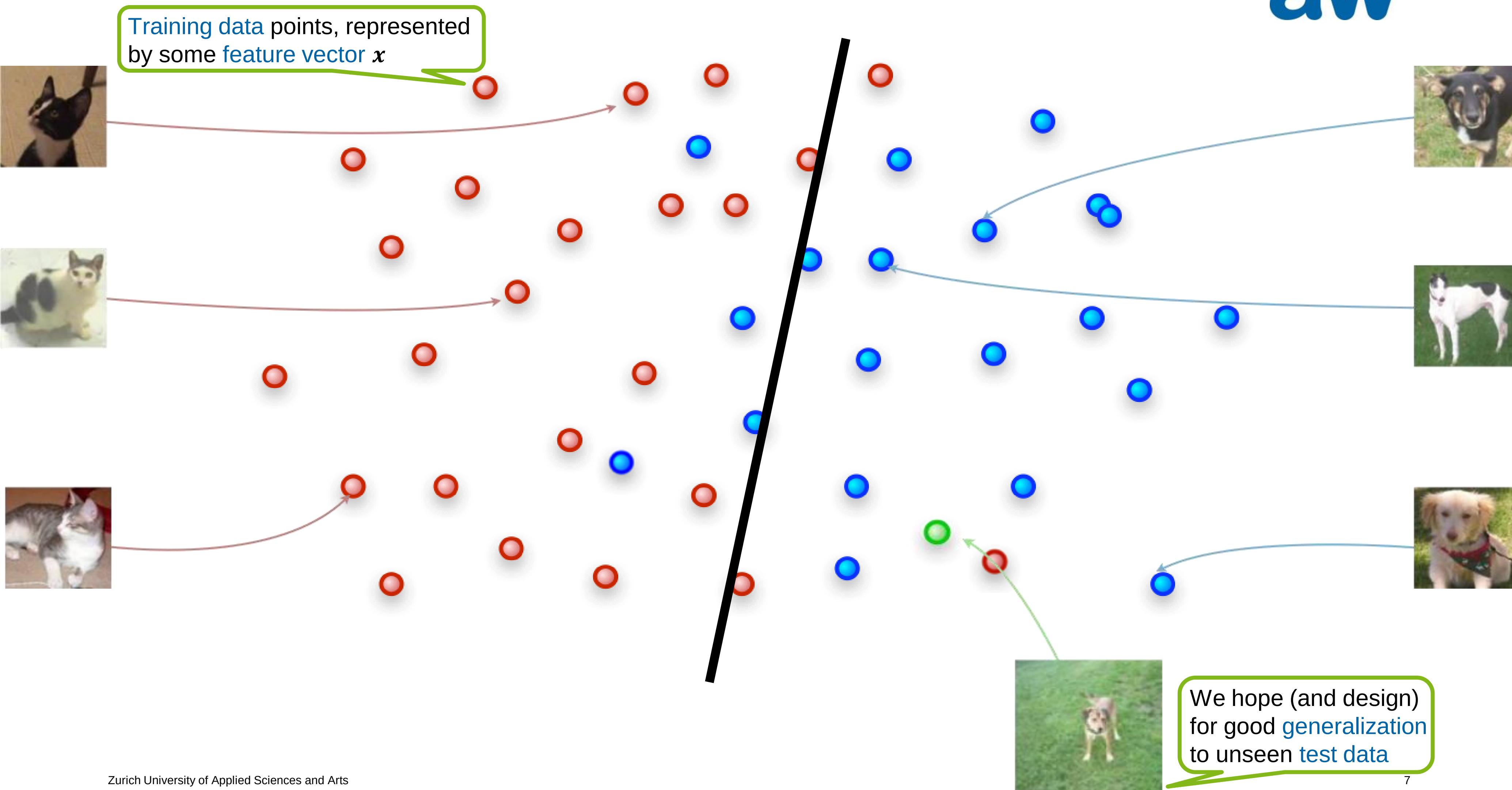
# Supervised machine learning in a nutshell



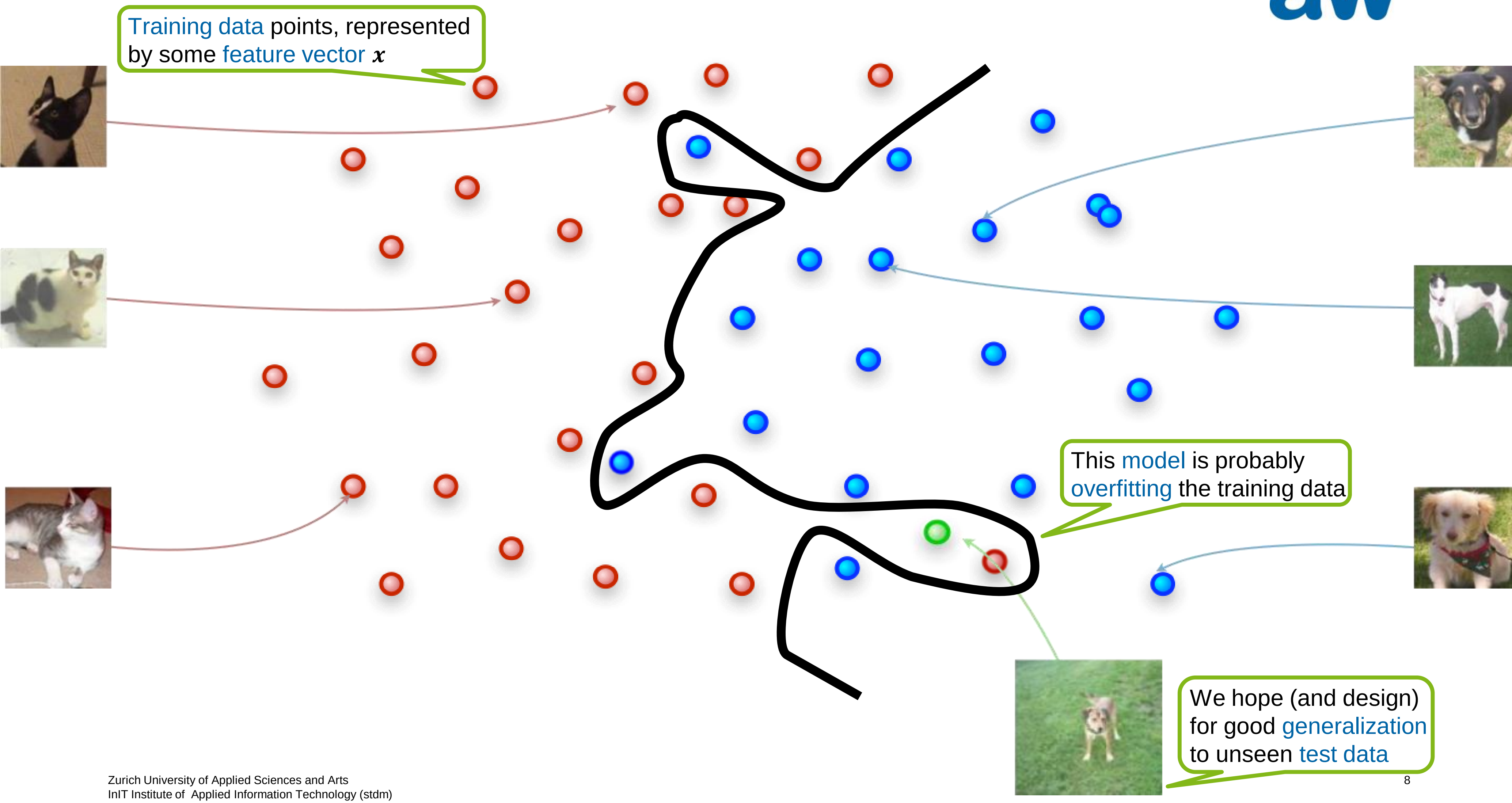
# Supervised machine learning in a nutshell



# Supervised machine learning in a nutshell

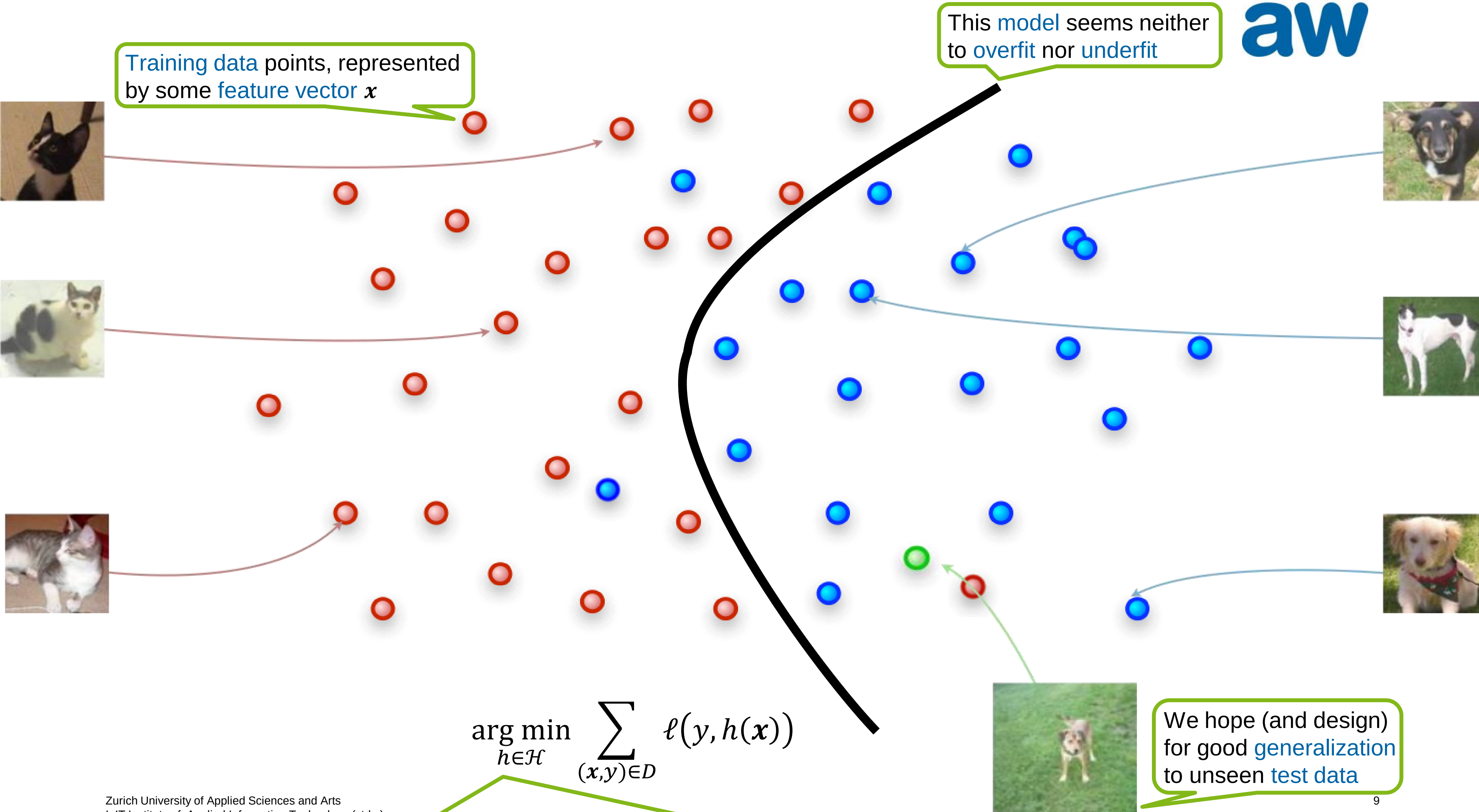


# Supervised machine learning in a nutshell



Zurich University
of Applied Sciences

# Supervised machine learning in a nutshell



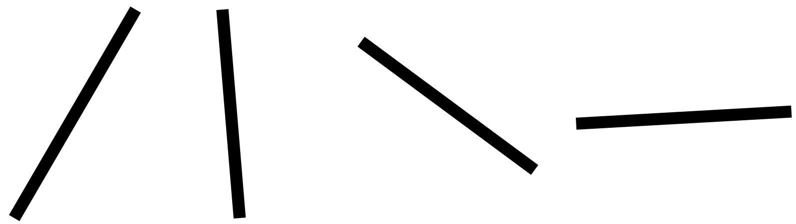

$$\underset{h \in \mathcal{H}}{\arg\min} \sum_{(\boldsymbol{x}, y) \in D} \ell\big(y, h(\boldsymbol{x})\big)$$

We search for models (functions) in a hypothesis space $\mathcal{H}$ by minimizing loss $\ell$ between label $y$ and result $h(\boldsymbol{x})$

# Learning as search through $\mathcal{H}$

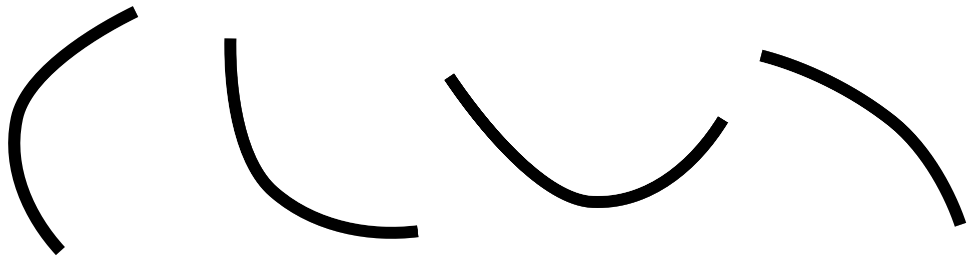
$$\mathcal{H} = \{ \qquad \}$$

# Learning as search through $\mathcal{H}$

$\mathcal{H} = \{$  $\ldots$ $\}$

# Learning as search through $\mathcal{H}$

$$\mathcal{H} = \{$$



$$\dots \}$$

Success is largely determined by **choosing the correct hypothesis space** for the problem:

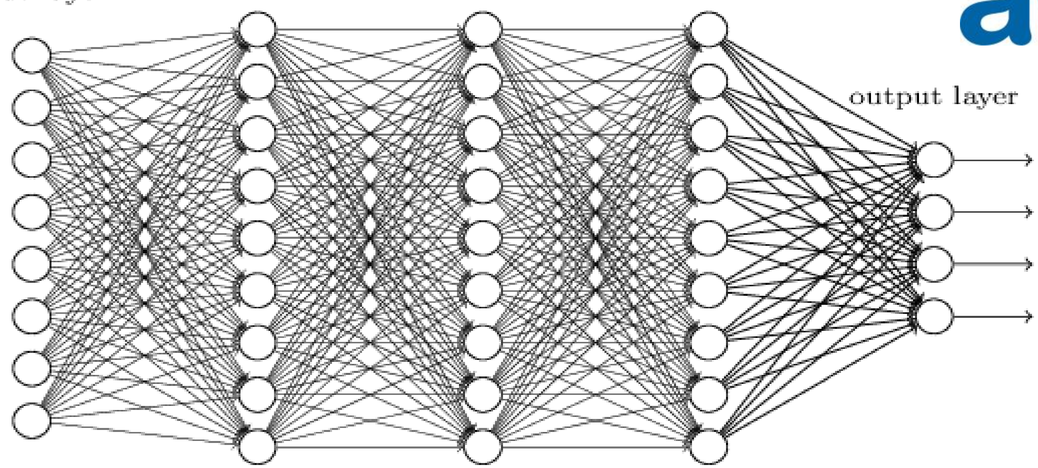
- Linear? Polynomial?
- Deep neural network? CNN?
- Ensemble of decision trees? …

$$h(\boldsymbol{x}) = h(\boldsymbol{x}, \boldsymbol{w})$$

Learning then means finding good parameters (sometimes called $\theta$)

# Learning as search through $\mathcal{H}$

$\mathcal{H} = \{$




$\}$

Success is largely determined by **choosing the correct hypothesis space** for the problem:

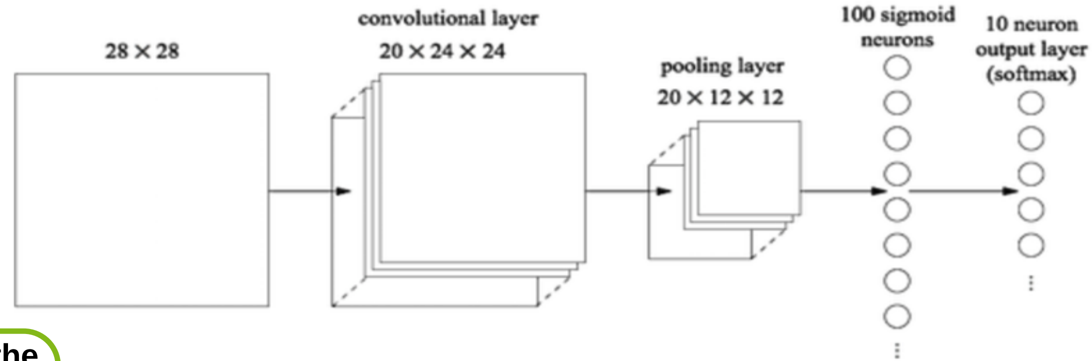
- Linear? Polynomial?
- Deep neural network? CNN?
- Ensemble of decision trees? …

$$h(\boldsymbol{x}) = h(\boldsymbol{x}, \boldsymbol{w})$$

Learning then means finding good parameters (sometimes called $\theta$)

# Learning as search through $\mathcal{H}$

$\mathcal{H} = \{$




$\}$

Success is largely determined by **choosing the correct hypothesis space** for the problem:

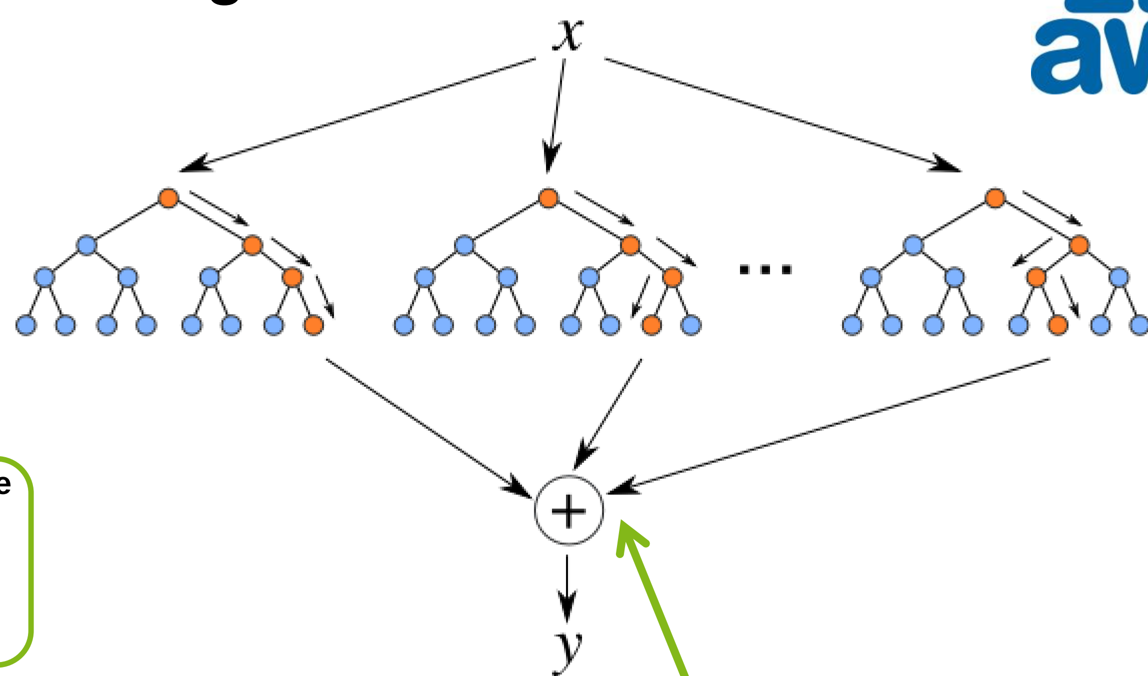
- Linear? Polynomial?
- Deep neural network? CNN?
- Ensemble of decision trees? …

$$h(\boldsymbol{x}) = h(\boldsymbol{x}, \boldsymbol{w})$$

Learning then means finding good parameters (sometimes called $\theta$)

# Learning as search through $\mathcal{H}$

$$\mathcal{H} = \{ \quad \cdots \quad \}$$




Success is largely determined by **choosing the correct hypothesis space** for the problem:
- Linear? Polynomial?
- Deep neural network? CNN?
- Ensemble of decision trees? …

$$h(\boldsymbol{x}) = h(\boldsymbol{x}, \boldsymbol{w})$$

A **good model** complies with Ockham's razor: **Maximize** a combination of **consistency and simplicity**

Learning then means finding good parameters (sometimes called $\theta$)

# What is this current hype about deep learning?

## Add depth (layers ➔ capability) to learn features automatically

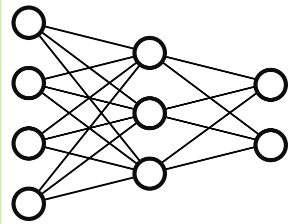
**Classic computer vision**

**Feature extraction
(SIFT, SURF, LBP, HOG, etc.)**

**(0.2, 0.4, …)**

**(0.4, 0.3, …)**

**Classification
(SVM, neuronal net, etc.)**



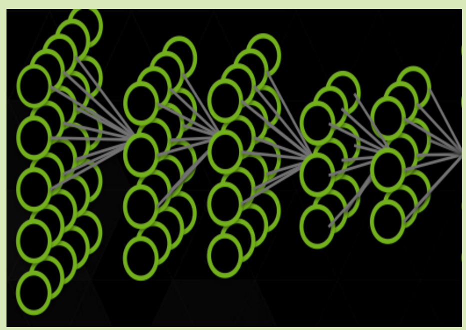
**container ship**

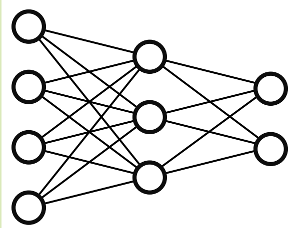
**tiger**

**…**

**Convolutional neural networks (CNNs)**

**Takes raw pixels as input, learns good features automatically!**





**container ship**

**tiger**

**…**

# Why study machine learning in general?

*«A learner that makes **no a priori assumptions** regarding the identity of the target concept has **no rational basis for classifying** any unseen instances»* [Mitchell, 1997, ch. 2.7.3]

## There's no single best algorithm

- No free lunch theorem (NFL) regarding the general equivalence of learners [Wolpert, 1996]:
  When all hypotheses $h$ are equally likely, the probability of observing an arbitrary sequence of cost values during training does not depend upon the learning algorithm $\mathcal{L}$
  ➔ there's **no universally best learner** (across problems)

- Empirical study [Caruana et al., 2006]:
  *«Even the best models sometimes perform poorly, and models with poor average performance occasionally perform exceptionally well»*
  ➔ **All** learning algorithms **have** advantages & **disadvantages**, **depending on the** current **data**

| | Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | Species |
|---|---|---|---|---|---|
| 1 | 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| 2 | 4.9 | 3.0 | 1.4 | 0.2 | setosa |
| 3 | 4.7 | 3.2 | 1.3 | 0.2 | setosa |
| 4 | 4.6 | 3.1 | 1.5 | 0.2 | setosa |
| 5 | 5.0 | 3.6 | 1.4 | 0.2 | setosa |
| 6 | 5.4 | 3.9 | 1.7 | 0.4 | setosa |
| 7 | 4.6 | 3.4 | 1.4 | 0.3 | setosa |
| 8 | 5.0 | 3.4 | 1.5 | 0.2 | setosa |

Examples of sensor data for pattern recognition tasks («Labeled faces in the wild» dataset) and tabular data («Iris» dataset)

## Ascertainment from kaggle.com

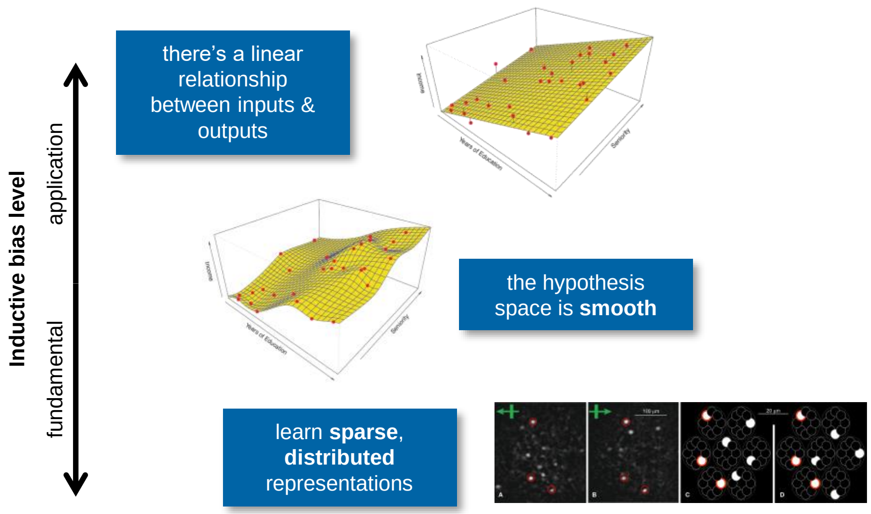
- **Tabular** data: do handcrafted feature engineering, followed by an **ensemble of decision trees**
- **Sensor** data (images, speech, …): use a *suitable* **deep neural network**

➔ See https://www.import.io/post/how-to-win-a-kaggle-competition/

# Why is there no *universally* best learner?
## Even if not, can there be a good *general* learner?

ML research unanimously states that *“there is no universally best learner”*. But a *general* learner doesn’t need to work for *all possible* kinds of data – it may suffice that it works well on *all data relevant* to human problem solving.

- [Optional] Conduct a quick search: What does the NFL theorem really claim (and what not)?
- Conduct a quick search on the concept of the “inductive bias” of a learning algorithm as its brought-in prior knowledge (e.g. Tom Mitchell’s work)
- Discuss: Are there more general forms of prior knowledge that universally guide learning?




# 2. DECISION TREES

# Attribute-based representations of data

## Valid for all kinds of data ( , | | Sepal.Length | |---|---| | 1 | 5.1 | | 2 | 4.9 | )

| | Sepal.Length |
|---|---|
| 1 | 5.1 |
| 2 | 4.9 |

Examples described by features

- Possible attribute values: Boolean, discrete, continuous, etc.
- Example: *"Situations where I will/won't **wait for a table**"*

Alternative nearby? | Has a bar to wait in? | Is it Friday? | Really hungry? | How crowded already? | Raining outside? | Did make reservation? | Minutes to wait | The label

| Example | Attributes | | | | | | | | | | Target |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | *Alt* | *Bar* | *Fri* | *Hun* | *Pat* | *Price* | *Rain* | *Res* | *Type* | *Est* | *WillWait* |
| $X_1$ | T | F | F | T | Some | \$\$\$ | F | T | French | 0–10 | T |
| $X_2$ | T | F | F | T | Full | \$ | F | F | Thai | 30–60 | F |
| $X_3$ | F | T | F | F | Some | \$ | F | F | Burger | 0–10 | T |
| $X_4$ | T | F | T | T | Full | \$ | F | F | Thai | 10–30 | T |
| $X_5$ | T | F | T | F | Full | \$\$\$ | F | T | French | >60 | F |
| $X_6$ | F | T | F | T | Some | \$\$ | T | T | Italian | 0–10 | T |
| $X_7$ | F | T | F | F | None | \$ | T | F | Burger | 0–10 | F |
| $X_8$ | F | F | F | T | Some | \$\$ | T | T | Thai | 0–10 | T |
| $X_9$ | F | T | T | F | Full | \$ | T | F | Burger | >60 | F |
| $X_{10}$ | T | T | T | T | Full | \$\$\$ | F | T | Italian | 10–30 | F |
| $X_{11}$ | F | F | F | F | None | \$ | F | F | Thai | 0–10 | F |
| $X_{12}$ | T | T | T | T | Full | \$ | F | F | Burger | 30–60 | T |

- Goal: classification of examples into positive ($T$) or negative ($F$) class

# Attribute-based representations of data

## Valid for all kinds of data ( ,  )

| | Sepal.Length |
|---|---|
| 1 | 5.1 |
| 2 | 4.9 |

Examples described by features

- Possible attribute values: Boolean, discrete, continuous, etc.
- Example: *“Situations where I will/won't* ***wait for a table****”*

Alternative nearby? | Has a bar to wait in? | Is it Friday? | Really hungry? | How crowded already? | Raining outside? | Did make reservation? | Minutes to wait

The label

| Example | Attributes | | | | | | | | | | Target |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | *Alt* | *Bar* | *Fri* | *Hun* | *Pat* | *Price* | *Rain* | *Res* | *Type* | *Est* | *WillWait* |
| $X_1$ | T | F | F | T | Some | $$$ | F | T | French | 0–10 | T |
| $X_2$ | T | F | F | T | Full | $ | F | F | Thai | 30–60 | F |
| $X_3$ | F | T | F | F | Some | $ | F | F | Burger | 0–10 | T |
| $X_4$ | T | F | T | T | Full | $ | F | F | Thai | 10–30 | T |
| $X_5$ | T | F | T | F | Full | $$$ | F | T | French | >60 | F |
| $X_6$ | F | T | F | T | Some | $$ | T | T | Italian | 0–10 | T |
| $X_7$ | F | T | F | F | None | $ | T | F | Burger | 0–10 | F |
| $X_8$ | F | F | F | T | Some | $$ | T | T | Thai | 0–10 | T |
| $X_9$ | F | T | T | F | Full | $ | T | F | Burger | >60 | F |
| $X_{10}$ | T | T | T | T | Full | $$$ | F | T | Italian | 10–30 | F |
| $X_{11}$ | F | F | F | F | None | $ | F | F | Thai | 0–10 | F |
| $X_{12}$ | T | T | T | T | Full | $ | F | F | Burger | 30–60 | T |

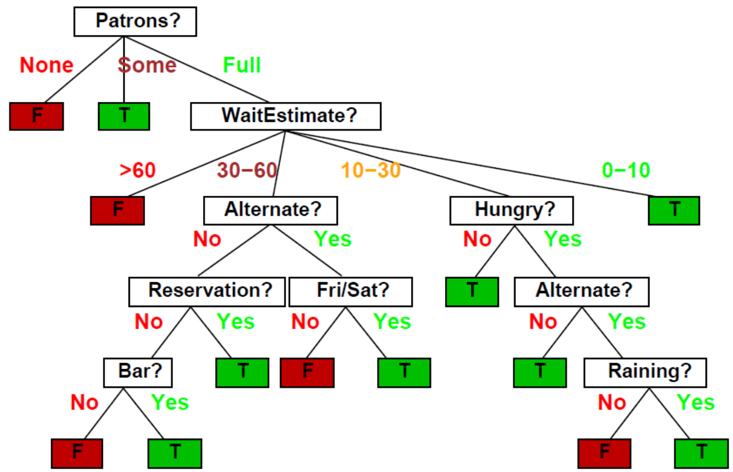
$\boldsymbol{x}$

$y$

- Goal: classification of examples into positive ($T$) or negative ($F$) class

# Decision tree representation of hypotheses

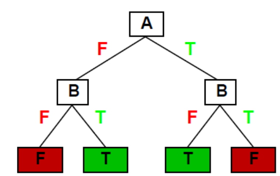
Example: Stuart Russell's *"true"* tree to **decide whether to wait** in a restaurant



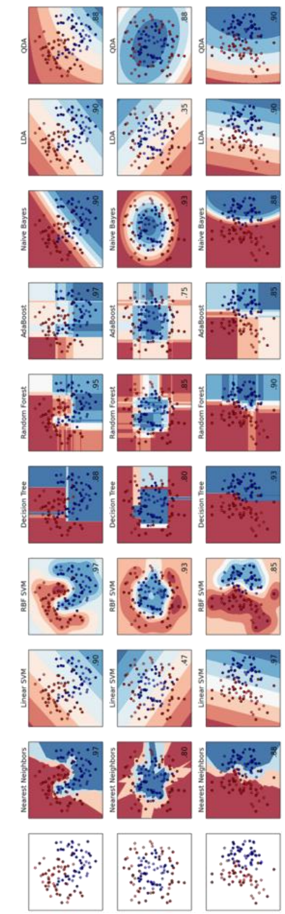

Expressiveness

- Decision trees can **express any function** of the input attributes
  E.g. for Boolean functions: truth table row ➔ path to leaf
- **Trivial** tree ∀ training sets: **one path** to leaf **for each example**
  But probably won't generalize to new examples
  ➔ Prefer to find more **compact** decision trees

| A | B | A xor B |
|---|---|---|
| F | F | F |
| F | T | T |
| T | F | T |
| T | T | F |



# Hypothesis spaces



Even a constrained hypothesis space is large

- **How many distinct** decision **trees** with $n$ Boolean attributes?
  - $=$ number of Boolean functions
  - $=$ number of distinct truth tables with $2^n$ rows $= 2^{2^n}$
  - Example: **6** Boolean **attributes** ➔ $18'446'744'073'709'551'616$ possible trees

- How many purely conjunctive hypotheses (e.g., $Hungry \wedge \neg Rain$)
  - Each attribute can be either positive, negative, or out of the hypothesis ➔ $3^n$

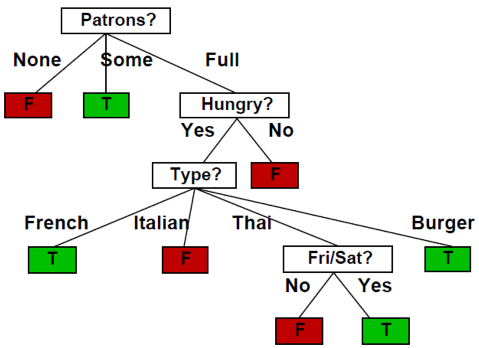
More expressive hypothesis spaces

- …increase chance that **target** function can be **expressed** ☺
- …increases **number** of hypotheses **consistent** w/ training set
  ➔ **may get worse** predictions ☹

Due to overfitting we have seen earlier

# Decision tree learning

Goal: find a **small** tree **consistent** with the training examples
Idea: (**recursively**) choose "**most significant**" **attribute as root** of (sub)tree




## Algorithm

- ```
  function LearnDecisionTree(examples, attributes) returns a tree
      return DecisionTreeLearning(examples, attributes, {})

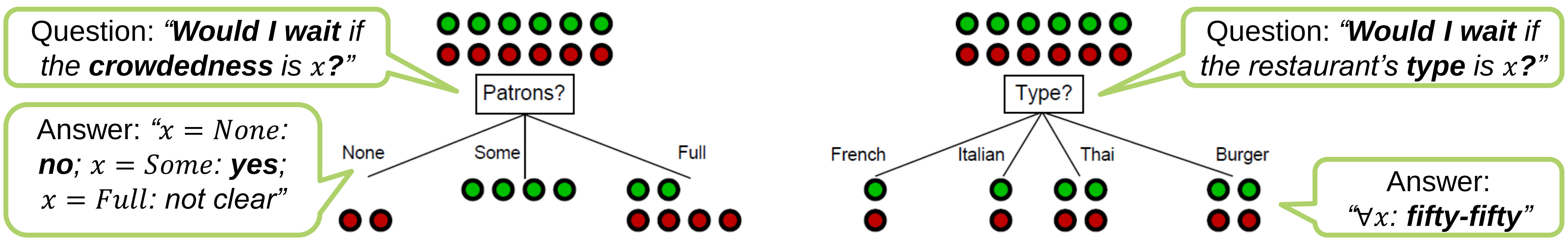
  function DecisionTreeLearning(examples, attributes, parent_examples) returns a tree
      if examples is empty then return PluralityValue(parent_examples)
      else if all examples have the same classification then return the classification
      else if attributes is empty then return PluralityValue(examples)
      else
          A ← argmax_{a∈attributes} Importance(a, examples)
          tree ← a new decision tree with root test A
          for each value v_k of A do #for categorical features
              exs ← {e: e∈examples and e.A=v_k}
              subtree ← DecisionTreeLearning(exs, attributes-A, examples)
              add a branch to tree with label (A=v_k) and subtree subtree
          return tree
  ```

- `PluralityValue(examples)` **selects** the **most common output** among examples
- `Importance(attribute, examples)` selects the **most important attribute**
- On ties, both functions choose randomly
  ```

# Choosing an attribute
## How to implement `Importance(attribute, examples)`

Idea: A **good attribute splits** examples into subsets that are (ideally) “**all pos**" **or** “all **neg**“

Example




- $Patrons$ is better choice: gives information about the classification

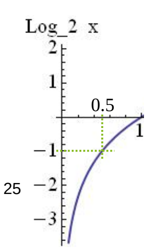
Recap: Information theory

- Information **answers questions**: The more **cluelessness** an observation **removes**, the more information it contains
- Inversely proportional to entropy (**uncertainty** of a random variable)
  - A Boolean answer with prior $< 0.5, 0.5 >$ has entropy$= 1\ bit$ (if we remove this uncertainty, we gain 1 bit of info.)
  - A coin giving heads $99\%$ of the time has entropy close to $0$ ($\approx 0.08\ bits$ → almost no info.-gain when observed)
  - **Entropy in** an **observation** (having prior $< P_1, \ldots, P_n >$): $H(\langle P_1, \ldots, P_n \rangle) = -\sum_{i=1}^{n} P_i \log_2 P_i$

Prior: Probabilities of all possible values of the random variable w.r.t. answer of question

Sum over all possible values

Bits needed to encode data, weighted by prob.



# Information gain as splitting criterion

Suppose we have $p$ positive and $n$ negative examples at the root

- $H\left(\left\langle\frac{p}{p+n},\frac{n}{p+n}\right\rangle\right)$ **bits needed** to **classify a** new **example**
- E.g., for the 12 restaurant examples, $p=n=6$, so we need overall 1 bit

An attribute $A$ splits the examples $E$ into **subsets** $E_i$ (one per possible value)

- Each of which (we hope) **need**s **less information** to complete the classification
- Let $E_i$ have $p_i$ positive and $n_i$ negative examples
  → $H\left(\left\langle\frac{p_i}{p_i+n_i},\frac{n_i}{p_i+n_i}\right\rangle\right)$ bits needed to classify a new example
- Expected number of necessary bits per example over all branches $i$ stemming from $A$ is

$$\text{Remainder(A)} = \sum_i \frac{p_i+n_i}{p+n} H\left(\left\langle\frac{p_i}{p_i+n_i},\frac{n_i}{p_i+n_i}\right\rangle\right)$$

Entropy of branch $i$, weighted by branch's size

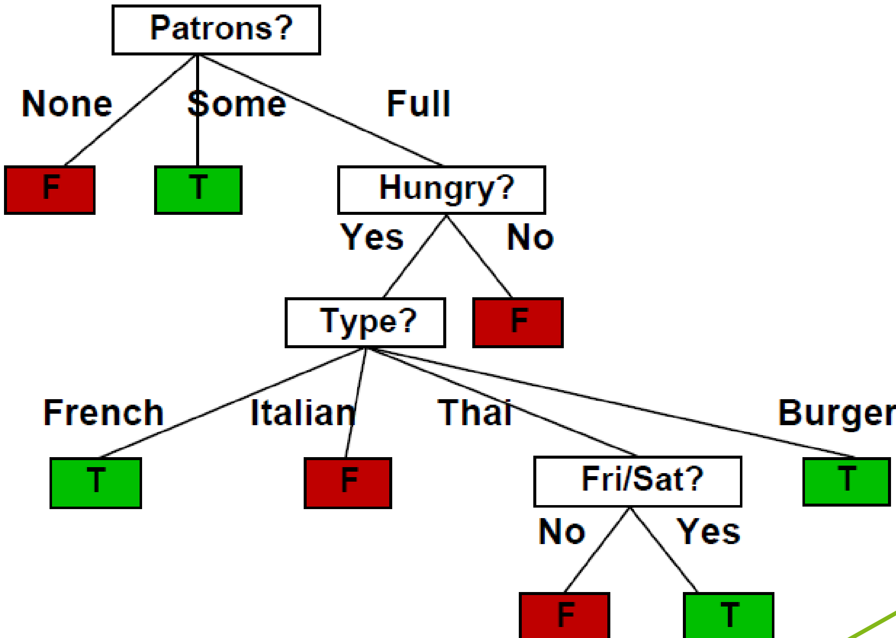
- For $Patrons$ this is $0.459\ bits$, for $Type$ this is (still) $1\ bit$
  ➔ **Choose** the attribute that **minimizes** the **remaining information need**ed, …
  ➔ i.e., maximizes information gain: $Gain(A) = H\left(\left\langle\frac{p}{p+n},\frac{n}{p+n}\right\rangle\right) - Remainder(A)$

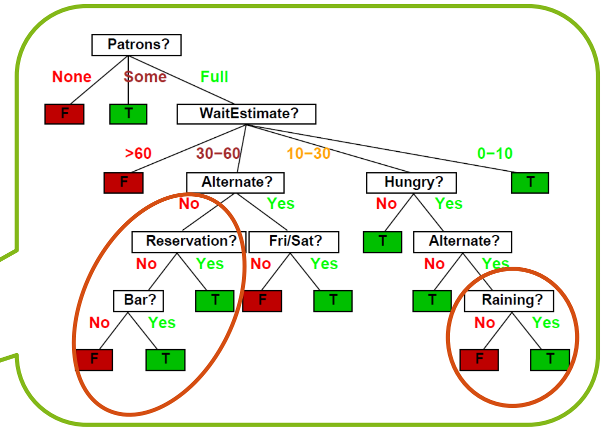
Entropy of original problem

Entropy remaining after splitting on $A$

# The learned decision tree

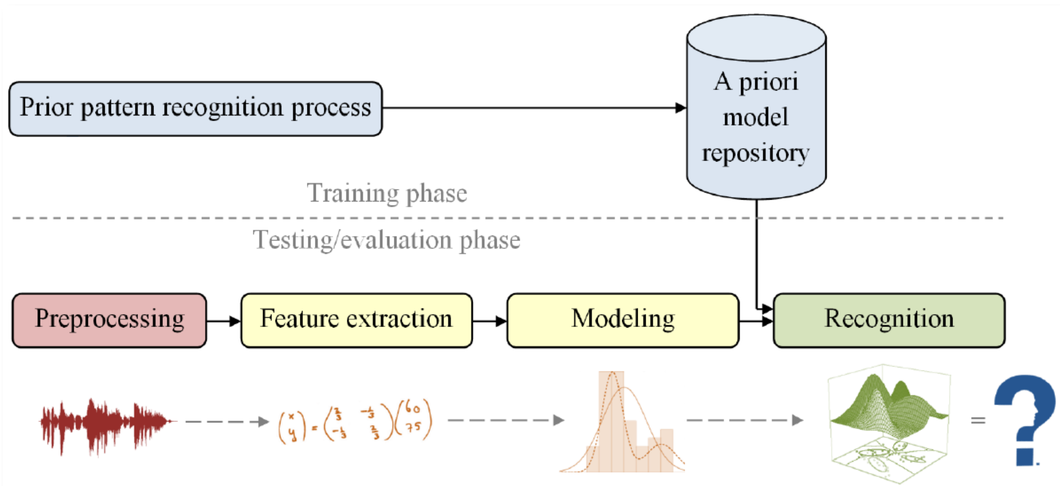
## Based on our 12 examples




- Substantially simpler than *"true"* tree
  → E.g., $Reservation$ and $Raining$ are not needed (perfect classification possible without)
- A more complex hypothesis isn't justified by the small amount of data
  → But **what makes one tree better than another?**

# 3. DOING MACHINE LEARNING



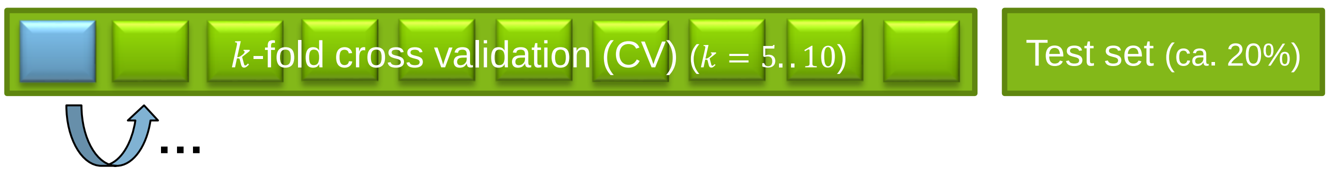
# Performance measurement

## The ML development process being an empirical science

Hume's *"Problem of Induction"* (1740): when is generalization admissible?

How do we know that $h \approx f$ (the true function)?

1. Use theorems of computational/statistical learning theory
2. **Try** $h$ on a **new** test set of **examples**
   - Prerequisite for inductive learning: generalizes (only) to **same distribution** as seen in training set!
   - Best practice: use cross-validation to train & validate on different sets before final test



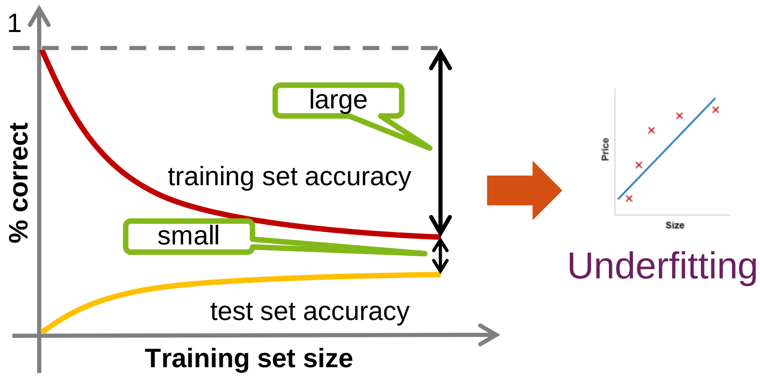

3. Report performance using recognized figures of merit
   - E.g. accuracy (or test set error) if all errors are equally costly: $\text{accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$
   - E.g. recall/precision if false alarms and misses differ in cost: $\text{recall} = \frac{TP}{TP+FN}$, $\text{precision} = \frac{TP}{TP+FP}$
   - Conduct **repeatable experiments** (i.e., fully scriptable, full documentation of inputs and results)

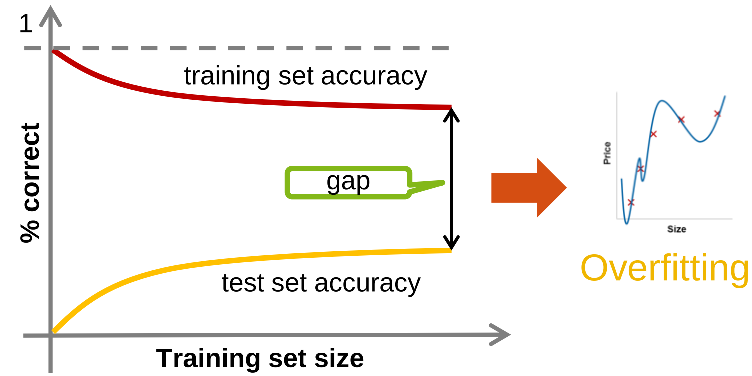
| **classification →<br>↓ label** | **1** | **0** |
|---|---|---|
| 1 | true positive (TP, "hit") | false negative (FN, "miss") |
| 0 | false positive (FP, "false alarm") | true negative (TN) |

# Debugging machine learning models

Learning curve: **%correct on train & test** set **as** a **function of training** set size

- Diagnostic: **reveal**s **over-** and **underfitting** as well as **realizability** (→ see appendix)




## What to try next when a given model generalizes poorly?

- Get **more training** examples → fixes overfitting
- Try **smaller sets** of **features** → fixes overfitting
- Try getting **additional features** → fixes underfitting
- Try **adding polynomial** features $x_1, x_2, x_1{}^2, x_2{}^2, \ldots$ → fixes underfitting
- Try **less regularization** → fixes underfitting
- Try **more** regularization → fixes overfitting
- Build **ensembles** → fixes overfitting, uses limited data best (→ see V09)

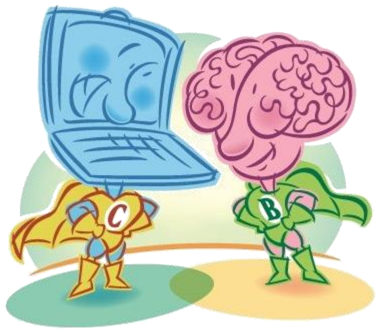
**Regularization**: Any method that **limits** the **expressiveness of** the **hypothesis space** by adding constraints to learning; e.g., pruning decision trees.

# Where's the intelligence?
## Man vs. machine

- Machine learning offers **general function approximations purely learned** from examples
- But: **Success depends on** a good fit of the algorithm's inductive bias to problem at hand
  → i.e., **clever algorithm choice** based on experience

- Learning is a **powerful principle of self-optimization**, **applicable to all** components of previously seen agent designs
- But: **General** (domain crossing, knowledge-linking) **learning must be** based on way better inclusion of **unsupervised** learning principles (besides general inductive biases)
  → current avant-garde deep learning research explores this route (→ see e.g. GANs in V11)

- **Decision trees** in principle **are simple** models (appreciated for their simplicity in formalism and interpretation), suitable only for Excel-like data
- But: **Combining multiple trees** (called an "ensemble") makes them **extremely powerful** for all but pattern recognition (i.e., sensor data-based) problems
  (and sometimes even there → see V09)



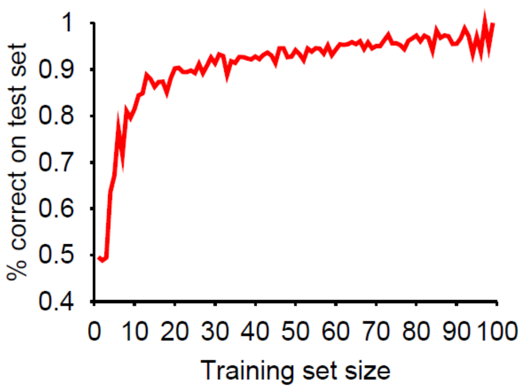
# Review

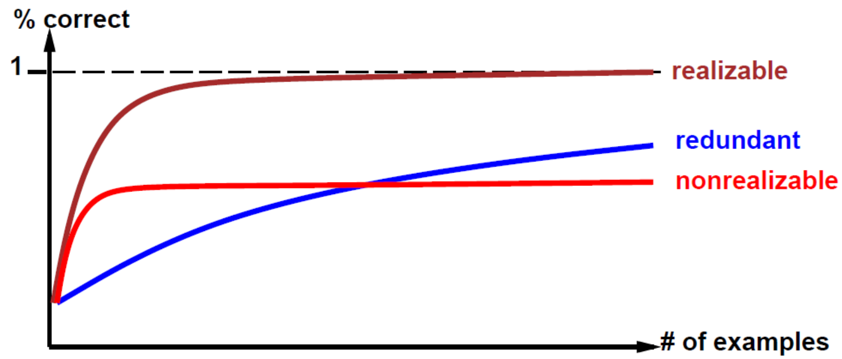
- Learning needed for unknown environments, “lazy designers”
- Learning agent = performance element (**testing** / application **phase**)
  + learning element (**training phase**)
- **Learning method** (algorithm) **depends** on…
  - type of performance element (classify? regress? control?),
  - available feedback (labels),
  - type of component to be improved (representation? utility function? action?),
  - and data representation (numerical or categorical data, logical clauses, raw pixels, …)
- For supervised learning, the **aim is** to find a **simple hypothesis** that is **approximately consistent** with training examples and **generalizes well**
- **Decision tree** learning **uses information gain**
  - **Popular** models because of easy interpretability
  - Many famous implementations (e.g. CART, C4.5®)
  - As ensembles: **very good general-purpose out-of-the-box** models (e.g. Random Forest®, XGBoost → see V09)
- Learning performance = prediction **accuracy** measured **on separate test set**
  - Development using 5-fold cross validation (without ever looking at test set!)
  - Systematic and repeatable experiments are paramount (e.g. using UNIX-style scripts)

# APPENDIX

# Learning curves
## Diagnosing learning problems

Learning curve, simplified: **%correct on test** set only **as** a **function of training** set size




Accuracy shown in learning curve depends on

- Realizability (target function expressible in chosen hypothesis space?)
  - **Non**-realizability can be due to **missing attributes**
  - or **restricted hypothesis class** (e.g., a thresholded linear function might be overly simplistic)
- **Redundant** features
  (e.g., loads of irrelevant attributes make learning difficult)

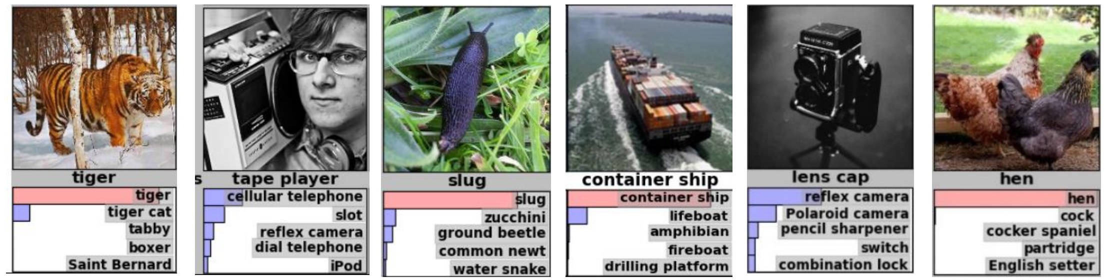
# Why is this current hype about deep learning?

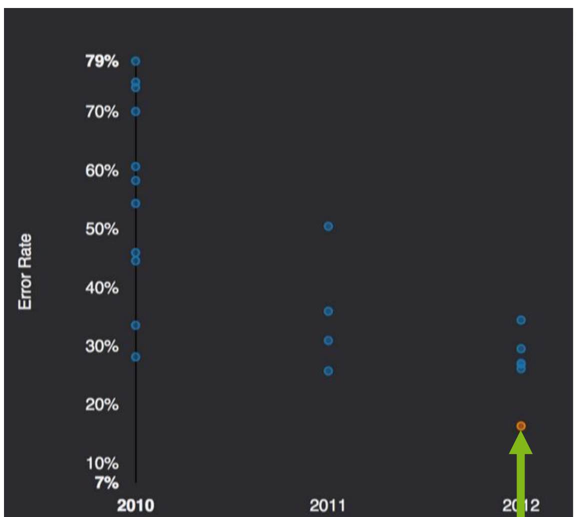
## The ImageNet Competition (more on deep learning → see appendix)

1000 categories

1 mio. training examples

…



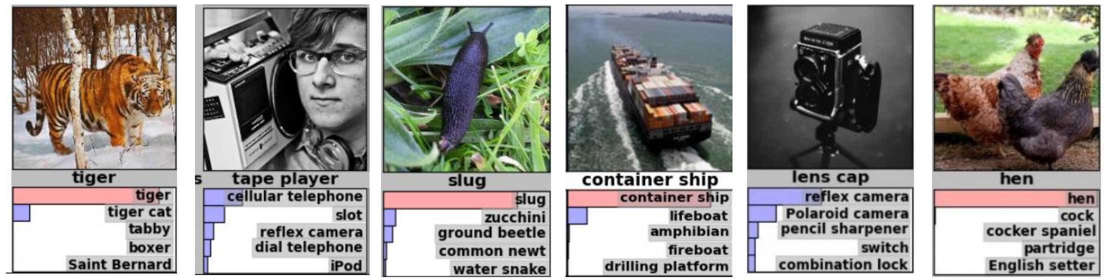



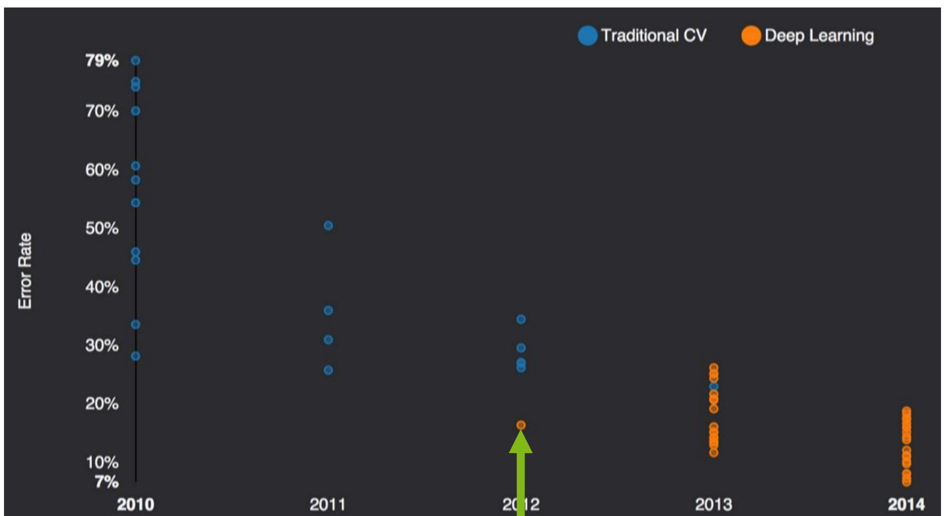

A. Krizhevsky uses a «Deep Convolutional Neural Network» (CNN) for the first time

# Why is this current hype about deep learning?

## **The ImageNet Competition** (more on deep learning → see appendix)

1000 categories
1 mio. training examples
…




A. Krizhevsky uses a «Deep Convolutional Neural Network» (CNN) for the first time

Zurich University
of Applied Sciences

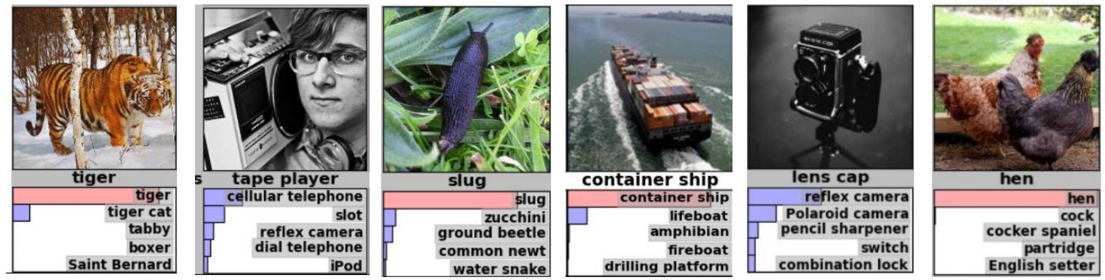
# Why is this current hype about deep learning?

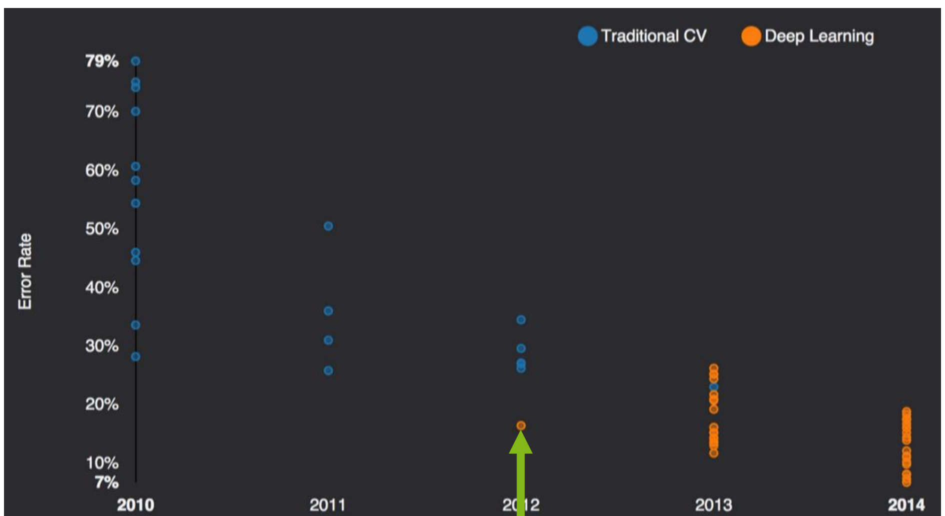
## **The ImageNet Competition** (more on deep learning → see appendix)


1000 categories

1 mio. training examples

…




A. Krizhevsky uses a «Deep Convolutional Neural Network» (CNN) for the first time

**2015: Computers learned to «see»**

4.95% Microsoft (Feb 06)
→ super-human performance (human: 5.10%)

4.80% Google (Feb 11)

4.58% Baidu (May 11)

3.57% Microsoft (Dec 10)

# 2016: A summer of breakthroughs in ML
## …enabled by deep learning

Impressive novelties within a summer’s timespan
- Game playing: beating the human Go world champion
- Audio synthesis: Synthesizing speech & music sample by sample
- Art style transfer: Redraw the content of a picture in the style of any painting
- Image synthesis: Completion of missing parts in pictures
- Text synthesis: Generation of text in specific styles (e.g., Shakespeare, $L^AT_EX$, …)
- Word vectors: Arithmetic with semantic meaning of text and images

➔ See next slides

# Google Acquires Artificial Intelligence Startup DeepMind For More Than $500M

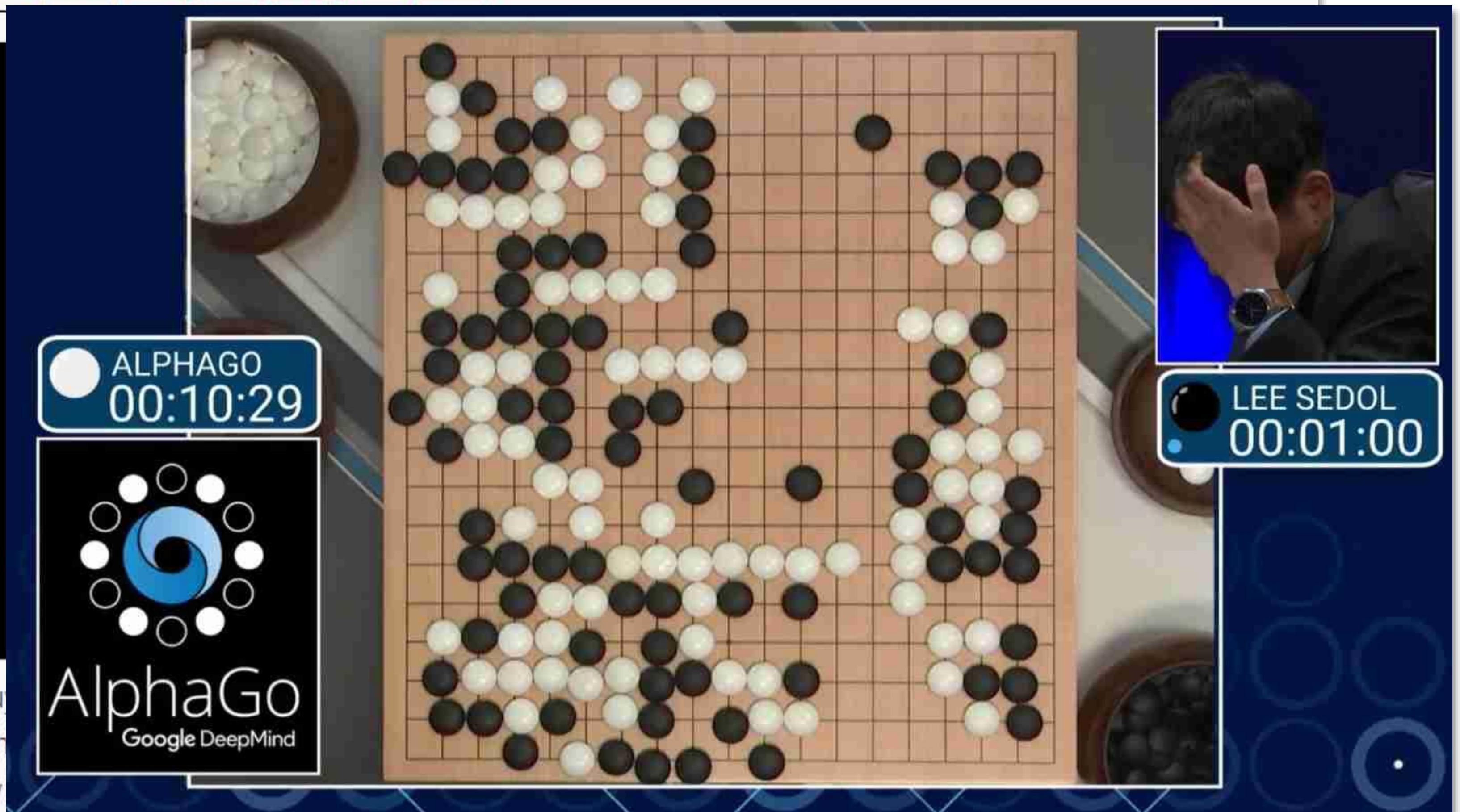
Posted Jan 26, 2014 by Catherine Shu (@catherineshu)

Google will buy London-based artificial intelligence company DeepMind. The Information reports that the acquisition price was more than $500 million, and that Facebook was also in talks to buy the startup late last year. DeepMind confirmed the acquisition to us, but couldn’t disclose deal terms.

The acquisition was originally confirmed by Google to Re/code.

# Google Acquires Artificial Intelligence Startup DeepMind For More Than $500M

Posted Jan 26, 2014 by Catherine Shu (@catherineshu)

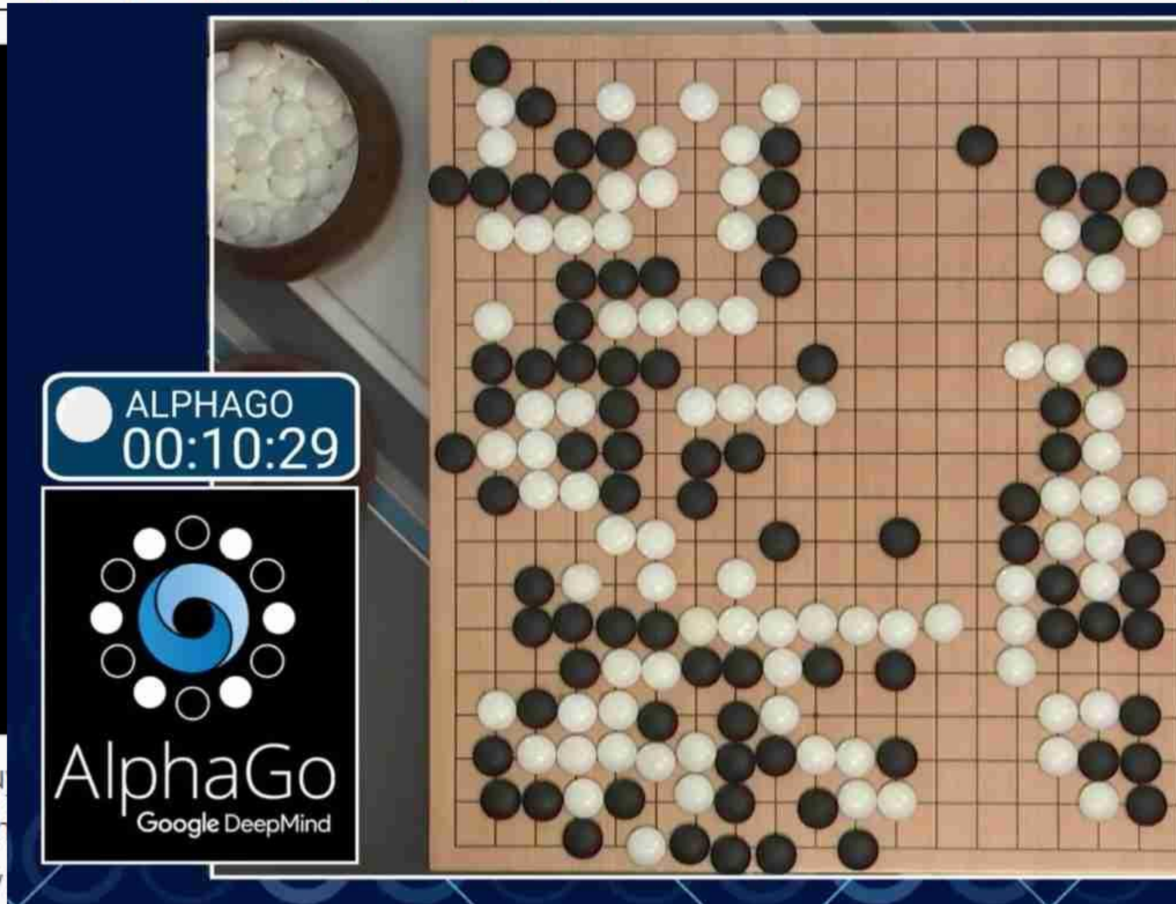



Google will bu
reports that th
in talks to buy
couldn't disclose deal terms.

The acquisition was originally confirmed by Google to Re/code.

# Google Acquires Artificial Intelligence Startup DeepMind For More Than $500M

Posted Jan 26, 2014 by Catherine Shu (@catherineshu)




Google will bu
reports that th
in talks to buy
couldn't disclose deal terms.

The acquisition was originally confirmed by Google to Re/code.

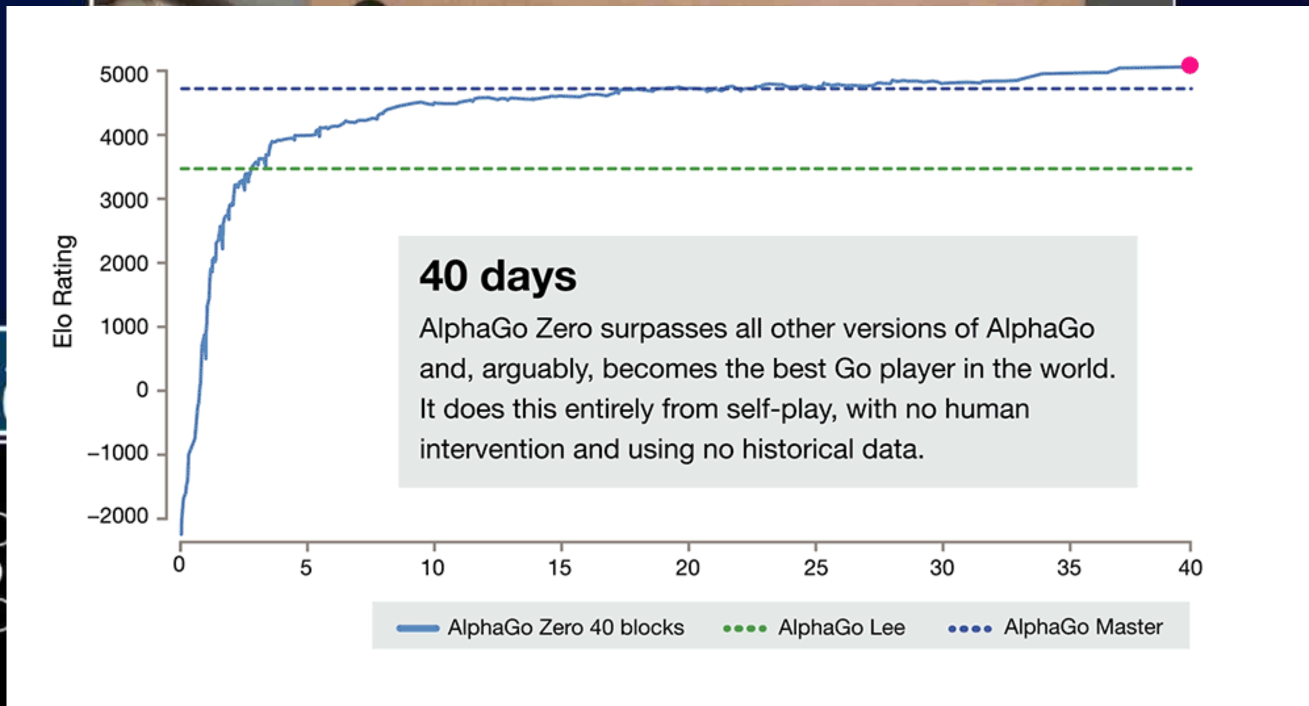

Zurich University of Applied Sciences

zh aw

Zurich University of Applied Sciences

# Google Acquires Artificial Intelligence Startup DeepMind For More Than $500M

Posted Jan 26, 2014 by Catherine Shu (@catherineshu)



Google will bu
reports that th
in talks to buy
couldn't disclose deal terms.

The acquisition was originally confirmed by Google to Re/code.

# Google's WaveNet uses neural nets to generate eerily convincing speech and music

Posted Sep 9, 2016 by Devin Coldewey

Generating speech from a piece of text is a common and important task undertaken by computers, but it's pretty rare that the result could be mistaken for ordinary speech. A new technique from researchers at Alphabet's DeepMind takes a completely different approach, producing speech and even music that sounds eerily like the real thing.

Early systems used a large library of the parts of speech (phonemes and morphemes) and a large ruleset that described all the ways letters combined to produce those sounds. The pieces were joined, or concatenated, creating functional speech synthesis that can handle most words, albeit with unconvincing cadence and tone. Later systems parameterized the generation of sound, making a library of speech fragments unnecessary. More compact — but often less effective.

WaveNet, as the system is called, takes things deeper. It simulates the sound of speech at as low a level as possible: one sample at a time. That means building the waveform from scratch — 16,000 samples per second.



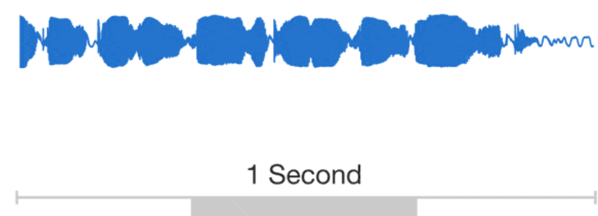
Crunchbase

Google –

Zurich University of Applied Sciences

Generated speech from text

Generated music out of creativity



# Google's WaveNet uses neural nets to generate eerily convincing speech and music

Posted Sep 9, 2016 by Devin Coldewey

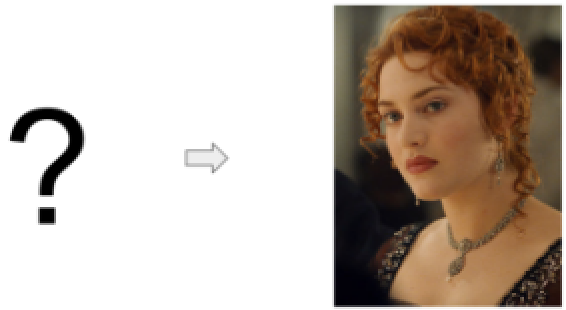
Generating speech from a piece of text
computers, but it's pretty rare that the
technique from researchers at Alphabe
producing speech and even music that

Early systems used a large library of the
large ruleset that described all the ways
pieces were joined, or concatenated, cr
most words, albeit with unconvincing ca
generation of sound, making a library o
but often less effective.

WaveNet, as the system is called, takes
low a level as possible: one sample at a
scratch — 16,000 samples per second.

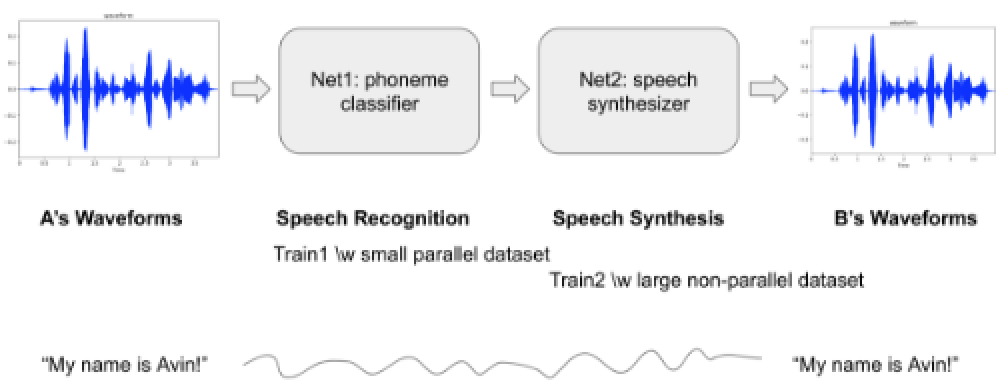
## Intro

What if you could imitate a famous celebrity's voice or sing like a famous singer? This project started with a goal to convert someone's voice to a specific target voice. So called, it's voice style transfer. We worked on this project that aims to convert someone's voice to a famous English actress Kate Winslet's voice. We implemented a deep neural networks to achieve that and more than 2 hours of audio book sentences read by Kate Winslet are used as a dataset.




## Model Architecture

This is a many-to-one voice conversion system. The main significance of this work is that we could generate a target speaker's utterances without parallel data like <source's wav, target's wav>, <wav, text> or <wav, phone>, but only waveforms of the target speaker. (To make these parallel datasets needs a lot of effort.) All we need in this project is a number of waveforms of the target speaker's utterances and only a small set of <wav, phone> pairs from a number of anonymous speakers.



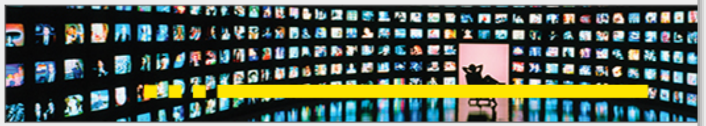

Zurich University of Applied Sciences

nerated
ech from text

nerated music
of creativity

MIT Technology Review

Topics+ Top Stories



Computing

# Algorithm Clones Van Gogh's Artistic Style and Pastes It onto Other Images, Movies

A deep neural network has learned to transfer artistic styles to other images.

by Emerging Technology from the arXiv May 10, 2016

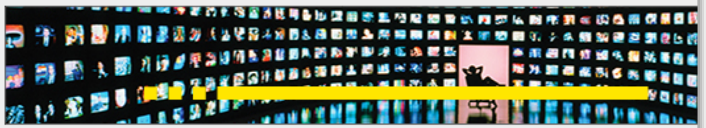
**The nature of artistic style is something of a mystery to most people. Think** of Vincent Van Gogh's *Starry Night*, Picasso's work on cubism, or Edvard Munch's *The Scream*. All have a powerful, unique style that humans recognize easily.

Zurich University of Applied Sciences

MIT Technology Review

Topics+ Top Stories

Zurich University of Applied Sciences



Computing

# Algorithm Clones Van Gogh's Artistic Style and Pastes It onto Other Images, Movies

A deep neural network has learned to transfer artistic styles to other images.

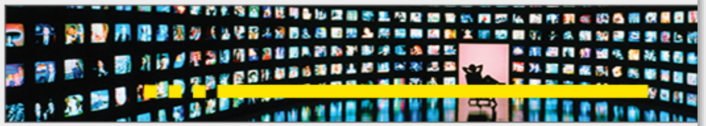
by Emerging Technology from the arXiv May 10, 2016

**The nature of artistic style is something of a mystery to most people. Think** of Vincent Van Gogh's *Starr* n, or Edvard Munch's *The Scream* e th humans recognize easily.

Zurich University
of Applied Sciences

MIT
Technology
Review

Topics+ Top Stories



Computing

# Algorithm Clones Van Gogh's Artistic Style and Pastes It onto Other Images, Movies

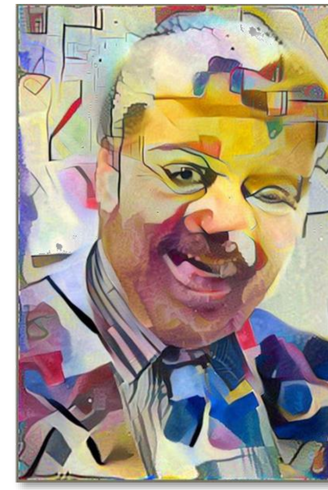
A deep neural network has learned to transfer artistic styles to other images.

by Emerging Technology from the arXiv May 10, 2016

**The nature of artistic style is something of a mystery to most people. Think** of Vincent Van Gogh's *Starr* … n, or Edvard Munch's *The Scream* … e th … humans recognize easily.

+

=



+

=

MIT
Techn
Revie

Computing

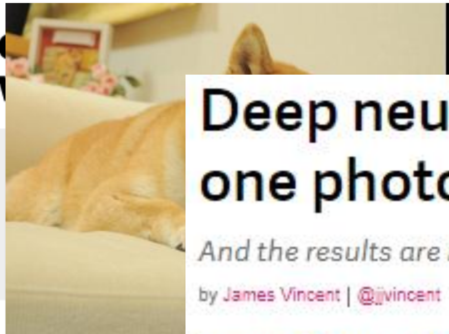
# Algorithm Clones Van Gogh's Artistic Style and Pastes It onto Other Images, Movies

A deep neural network has learned to transfer artistic styles to other images.

by Emerging Technology from the arXiv   May 10, 2016

**The nature of artistic style is something of a mystery to most people. Think** of Vincent Van Gogh's *Starr*… …n, or Edvard Munch's *The Scream*… …e th… humans recognize easily.

MIT
Techn
Revie



Zurich University
of Applied Sciences

zh
aw

Computing

# Algorith
# Artistic
# Other In

A deep neural n
other images.

by Emerging Tech

**The nature of arti**
of Vincent Van G
Edvard Munch's
humans recogni

f

# Deep neural networks can now transfer the style of one photo onto another

*And the results are impressive*

by James Vincent | @jjvincent | Mar 30, 2017, 1:53pm EDT

SHARE TWEET LINKEDIN



Original photo Reference photo Result

You've probably heard of an AI technique known as "style transfer" — or, if you haven't heard of it, you've seen it. The process uses neural networks to apply the look and feel of one image to another, and appears in apps like Prisma and Facebook. These style transfers, however, are stylistic, not photorealistic. They look good because they look like they've been painted. Now a group of researchers from Cornell University and Adobe have augmented

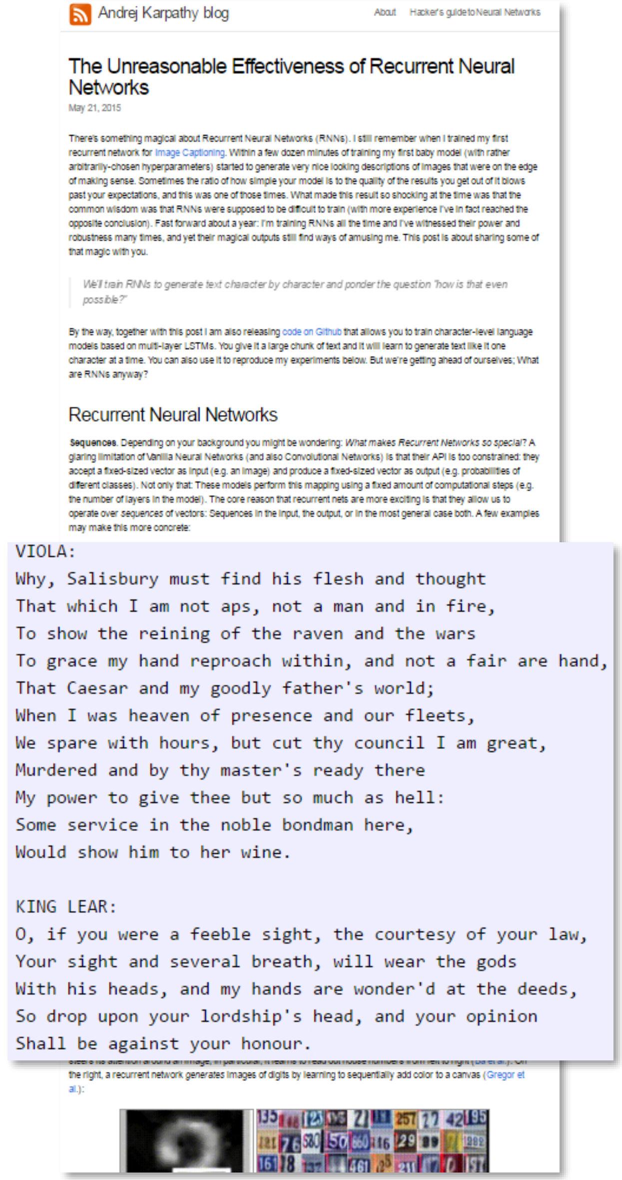
NOW TRENDING

Zurich University
of Applied Sciences

# …and the list could be continued

Zurich University
of Applied Sciences

# …and the list could be continued

Zurich University
of Applied Sciences

# …and the list could be continued

Brandon Amos   About   Blog

## Image Completion with Deep Learning in TensorFlow

*August 9, 2016*



### Introduction

Content-aware fill is a p
completion and inpaintin
do content-aware fill, im
"Semantic Image Inpaint
shows how to use deep l
some deeper portions fo
section can be skipped i
from images of faces. I h
completion.tensorflow.

We'll approach image co

1. We'll first interpret
2. This interpretation
3. Then we'll find the

Andrej Karpathy blog   About   Hacker's guide to Neural Networks

## The Unreasonable Effectiveness of Recurrent Neural Networks

May 21, 2015

There's something magical about Recurrent Neural Networks (RNNs). I still remember when I trained my first recurrent network for Image Captioning. Within a few dozen minutes of training my first baby model (with rather arbitrarily-chosen hyperparameters) started to generate very nice looking descriptions of images that were on the edge of making sense. Sometimes the ratio of how simple your model is to the quality of the results you get out of it blows past your expectations, and this was one of those times. What made this result so shocking at the time was that the common wisdom was that RNNs were supposed to be difficult to train (with more experience I've in fact reached the opposite conclusion). Fast forward about a year: I'm training RNNs all the time and I've witnessed their power and robustness many times, and yet their magical outputs still find ways of amusing me. This post is about sharing some of that magic with you.

> *We'll train RNNs to generate text character by character and ponder the question "how is that even possible?"*

By the way, together with this post I am also releasing code on Github that allows you to train character-level language models based on multi-layer LSTMs. You give it a large chunk of text and it will learn to generate text like it one character at a time. You can also use it to reproduce my experiments below. But we're getting ahead of ourselves; What are RNNs anyway?

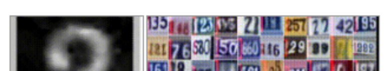
### Recurrent Neural Networks

**Sequences.** Depending on your background you might be wondering: *What makes Recurrent Networks so special?* A glaring limitation of Vanilla Neural Networks (and also Convolutional Networks) is that their API is too constrained: they accept a fixed-sized vector as input (e.g. an image) and produce a fixed-sized vector as output (e.g. probabilities of different classes). Not only that: These models perform this mapping using a fixed amount of computational steps (e.g. the number of layers in the model). The core reason that recurrent nets are more exciting is that they allow us to operate over sequences of vectors: Sequences in the input, the output, or in the most general case both. A few examples may make this more concrete:

```
VIOLA:
Why, Salisbury must find his flesh and thought
That which I am not aps, not a man and in fire,
To show the reining of the raven and the wars
To grace my hand reproach within, and not a fair are hand,
That Caesar and my goodly father's world;
When I was heaven of presence and our fleets,
We spare with hours, but cut thy council I am great,
Murdered and by thy master's ready there
My power to give thee but so much as hell:
Some service in the noble bondman here,
Would show him to her wine.

KING LEAR:
O, if you were a feeble sight, the courtesy of your law,
Your sight and several breath, will wear the gods
With his heads, and my hands are wonder'd at the deeds,
So drop upon your lordship's head, and your opinion
Shall be against your honour.
```

the right, a recurrent network generates images of digits by learning to sequentially add color to a canvas (Gregor et al.):



## the morning paper

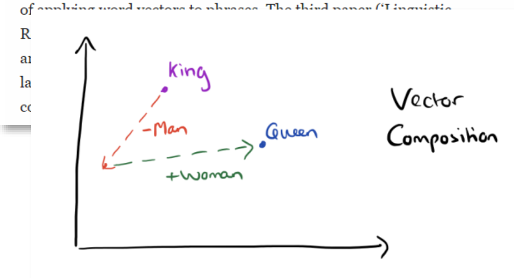
### The amazing power of word vectors

APRIL 21, 2016

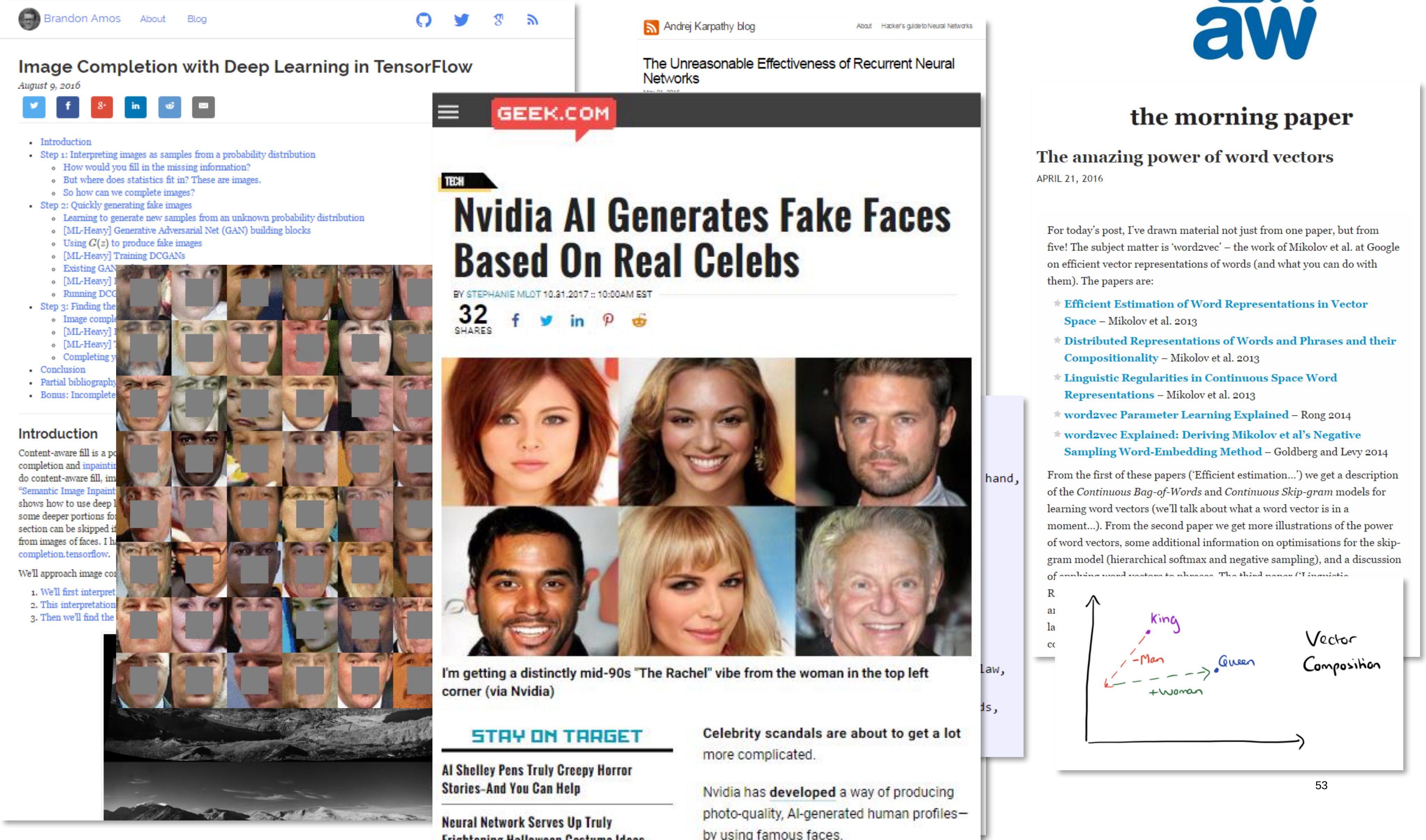
For today's post, I've drawn material not just from one paper, but from five! The subject matter is 'word2vec' – the work of Mikolov et al. at Google on efficient vector representations of words (and what you can do with them). The papers are:

- **Efficient Estimation of Word Representations in Vector Space** – Mikolov et al. 2013
- **Distributed Representations of Words and Phrases and their Compositionality** – Mikolov et al. 2013
- **Linguistic Regularities in Continuous Space Word Representations** – Mikolov et al. 2013
- **word2vec Parameter Learning Explained** – Rong 2014
- **word2vec Explained: Deriving Mikolov et al's Negative Sampling Word-Embedding Method** – Goldberg and Levy 2014

From the first of these papers ('Efficient estimation…') we get a description of the *Continuous Bag-of-Words* and *Continuous Skip-gram* models for learning word vectors (we'll talk about what a word vector is in a moment…). From the second paper we get more illustrations of the power of word vectors, some additional information on optimisations for the skip-gram model (hierarchical softmax and negative sampling), and a discussion

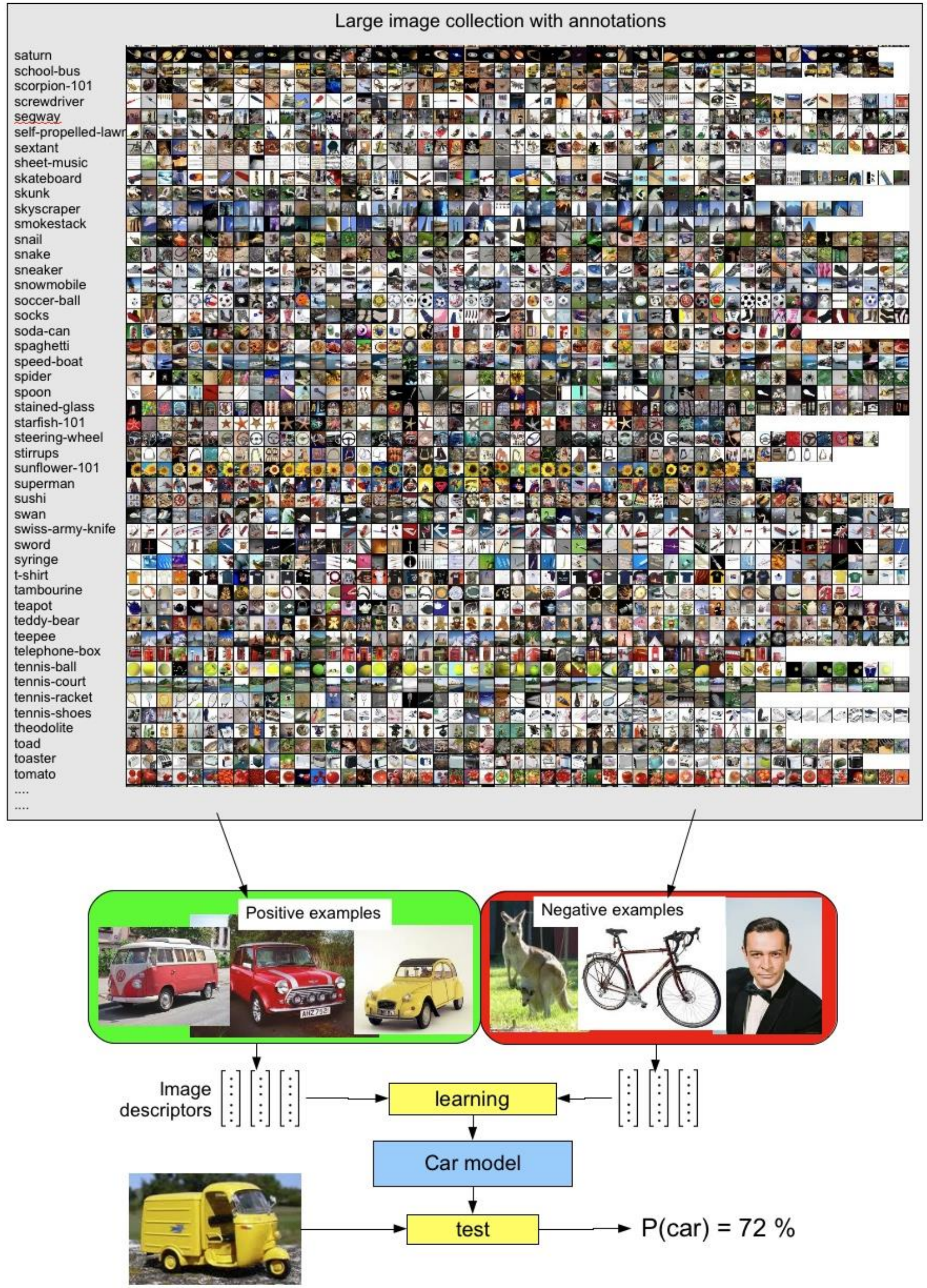


# …and the list could be continued



# Inductive supervised learning

Assumption
- A model fit to *enough* training examples…
- …will **generalize** *well* to unseen test data




Source: http://lear.inrialpes.fr/job/postdoc-large-scale-classif-11-img/attribs_patchwork.jpg

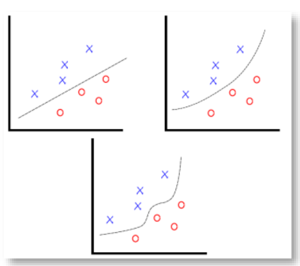
# Inductive supervised learning

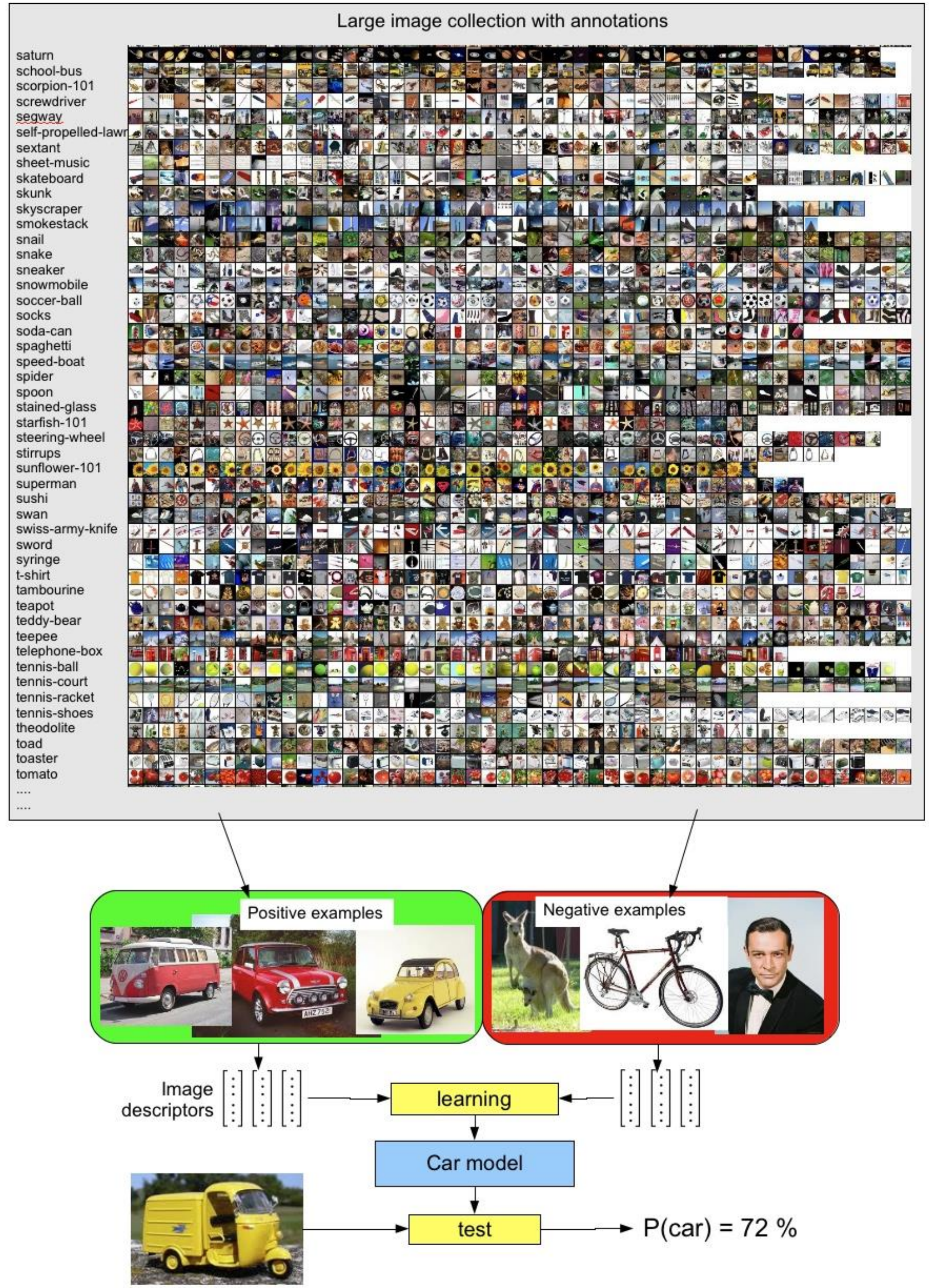
## Assumption

- A model fit to *enough* training examples…
- …will **generalize** *well* to unseen test data

## Method

- **Search for parameters** of a given class of functions…
- …such that every training input (e.g. an image) is mapped to the correct output label (e.g. «`car`»)

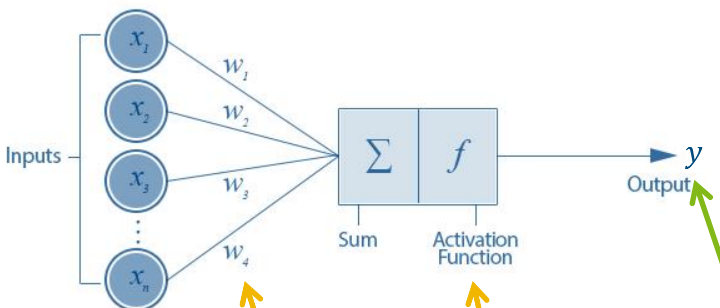





Source: http://lear.inrialpes.fr/job/postdoc-large-scale-classif-11-img/attribs_patchwork.jpg

# What is the effect of parameter search?
## What is the effect of more capable function classes?

Neuron




Features (e.g. pixels)

Adjustable parameters

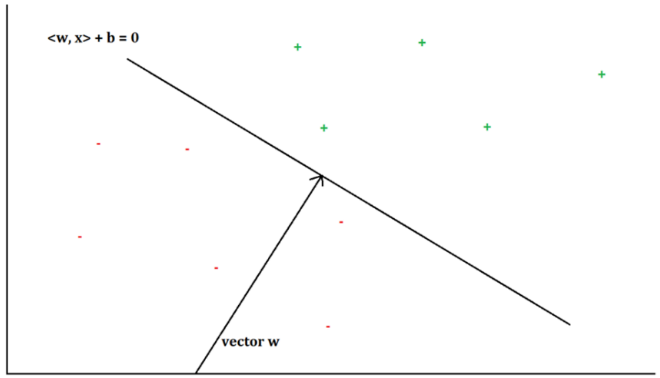
Decision (threshold)

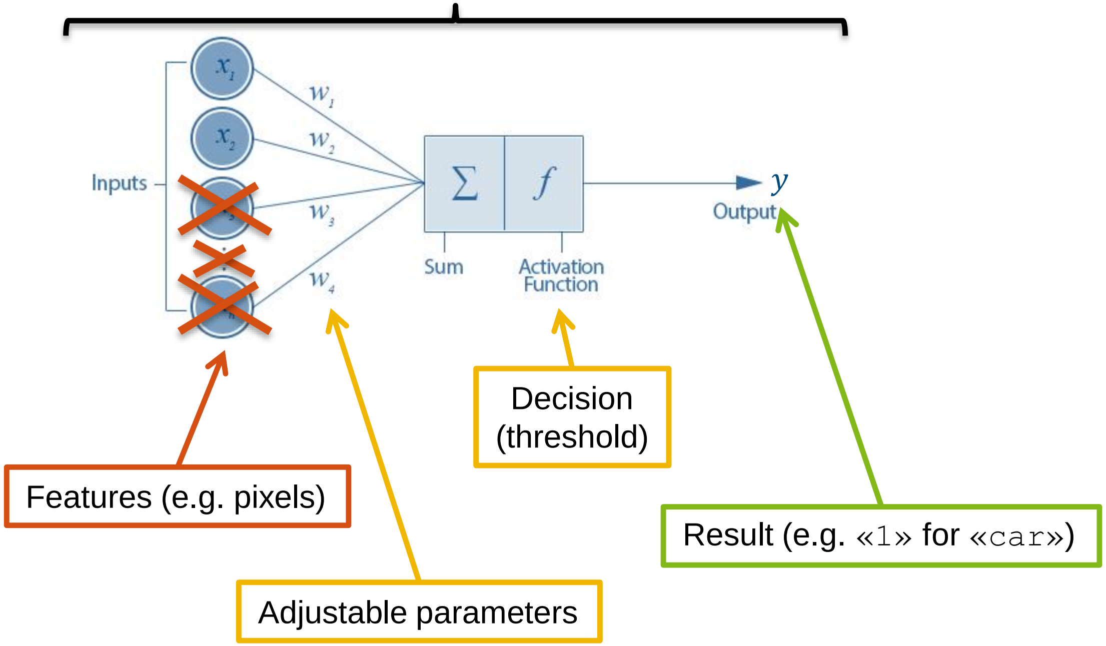
Result (e.g. «`1`» for «`car`»)

# What is the effect of parameter search?

## What is the effect of more capable function classes?

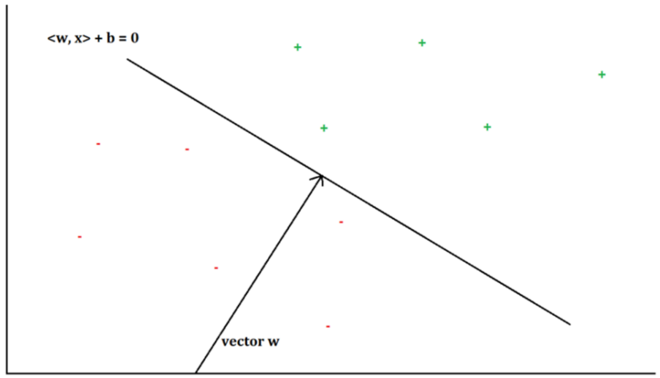
Neuron

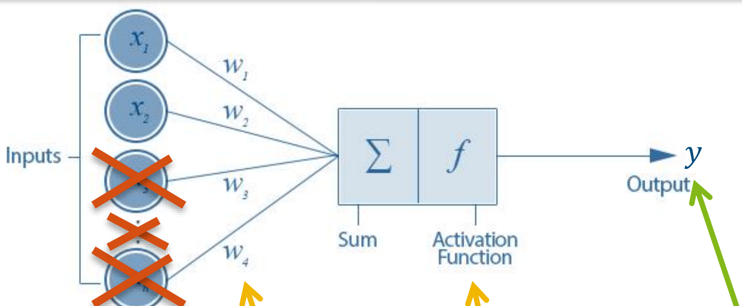


# What is the effect of parameter search?
## What is the effect of more capable function classes?

Neuron



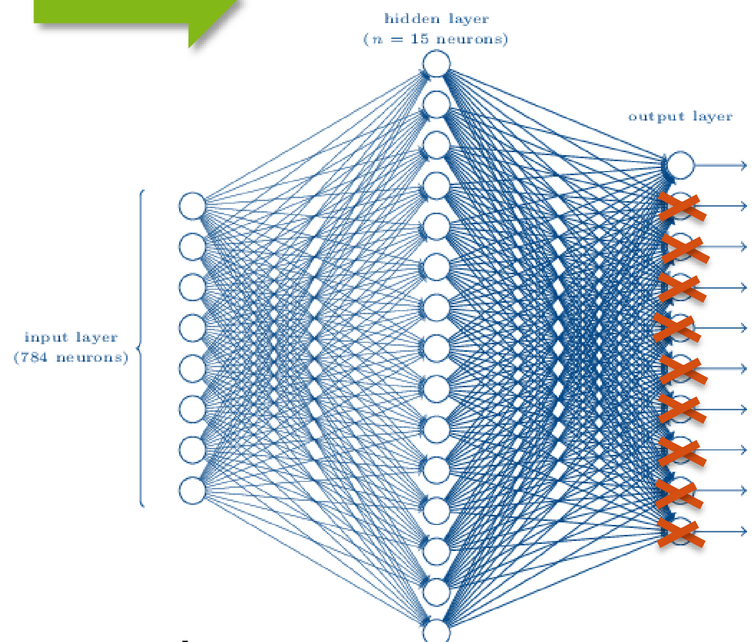

Features (e.g. pixels)

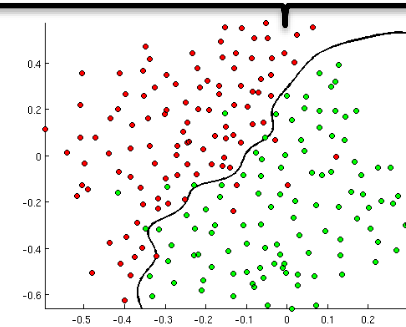
Adjustable parameters

Decision (threshold)

Result (e.g. «1» for «car»)

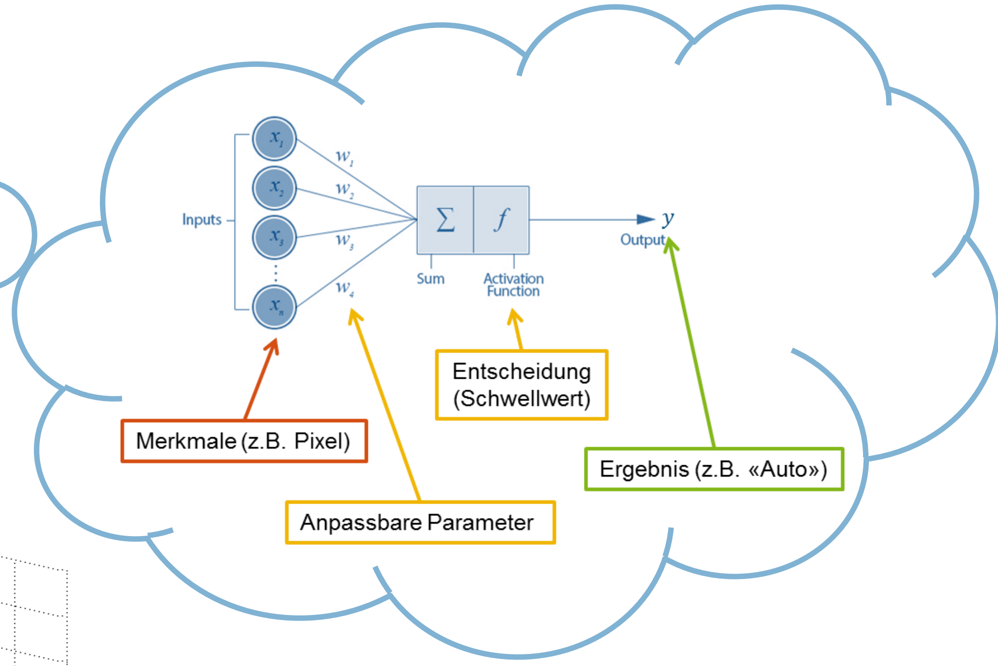
Neural Network



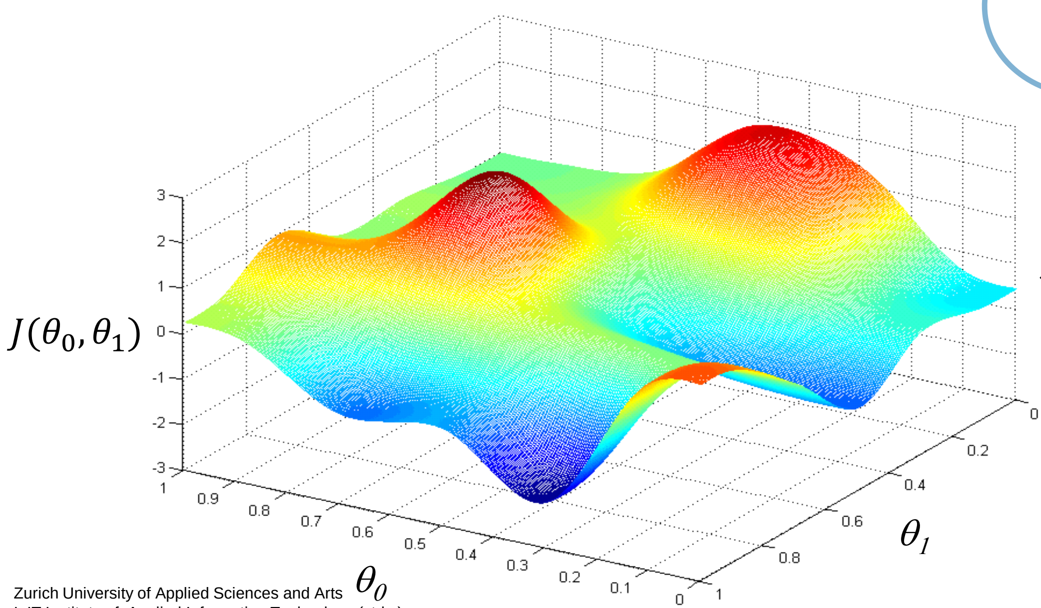



# How are the parameters found?

- Definition of the neural net: $f_{\vec{\theta}}(x) = y$
  with image $x$, true result $y$ and all parameters $\vec{\theta}$
  ($\vec{\theta} = \{w_1, w_2\}$ chosen randomly at start)



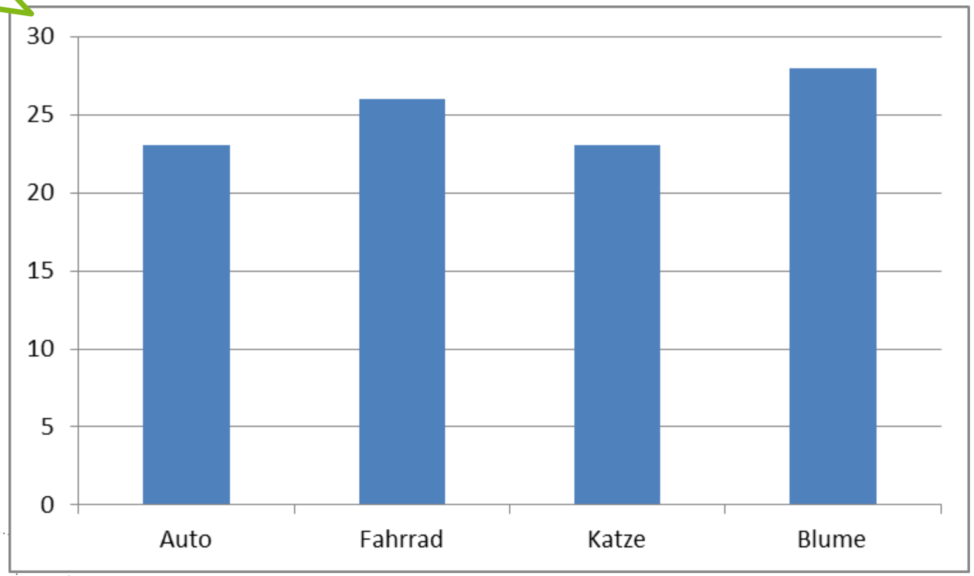

- Error measure: $J\left(\vec{\theta}\right) = \frac{1}{N}\sum_{i=1}^{N}\left(f_{\vec{\theta}}(x_i) - y_i\right)^2$
  Mean squared error




← Error landscape

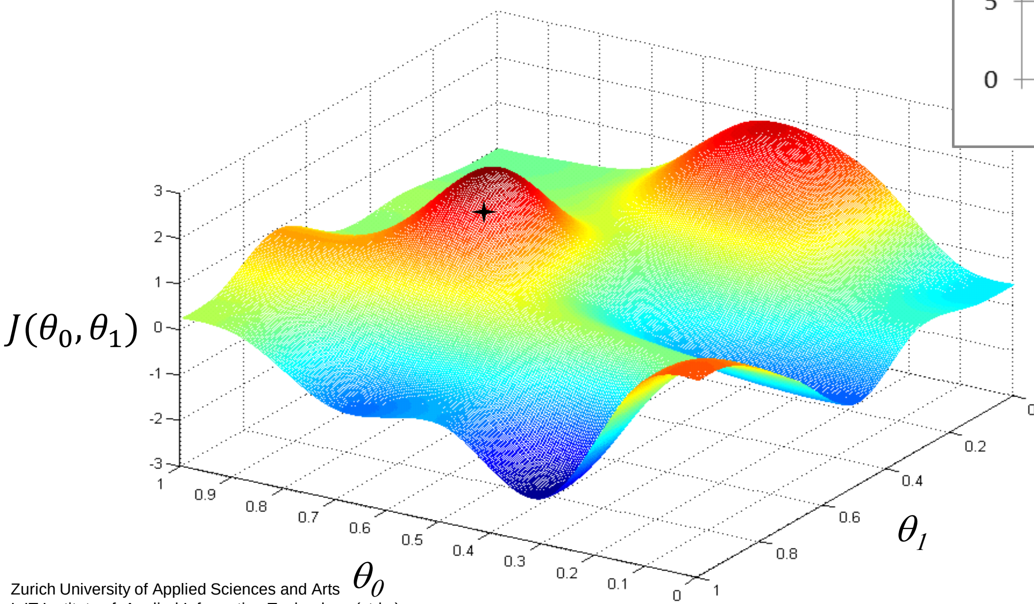
# How are the parameters found?

Probability [%] for a specific outcome




- Definition of the neural net: $f_{\vec{\theta}}(x) = y$
  with image $x$, true result $y$ and all parameters $\vec{\theta}$
  ($\vec{\theta} = \{w_1, w_2\}$ chosen randomly at start)

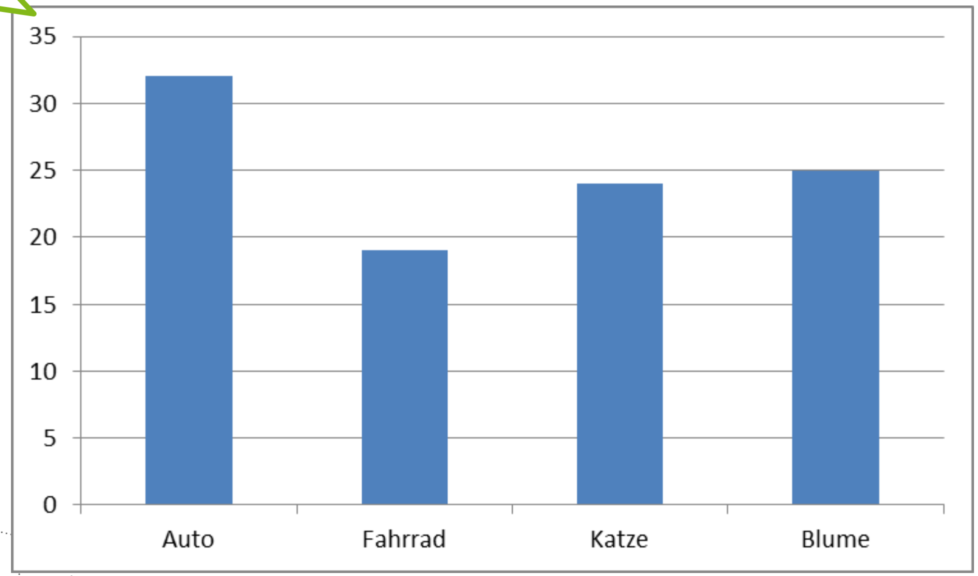
- Error measure: $J\left(\vec{\theta}\right) = \frac{1}{N}\sum_{i=1}^{N}\left(f_{\vec{\theta}}(x_i) - y_i\right)^2$
  Mean squared error




← Error landscape

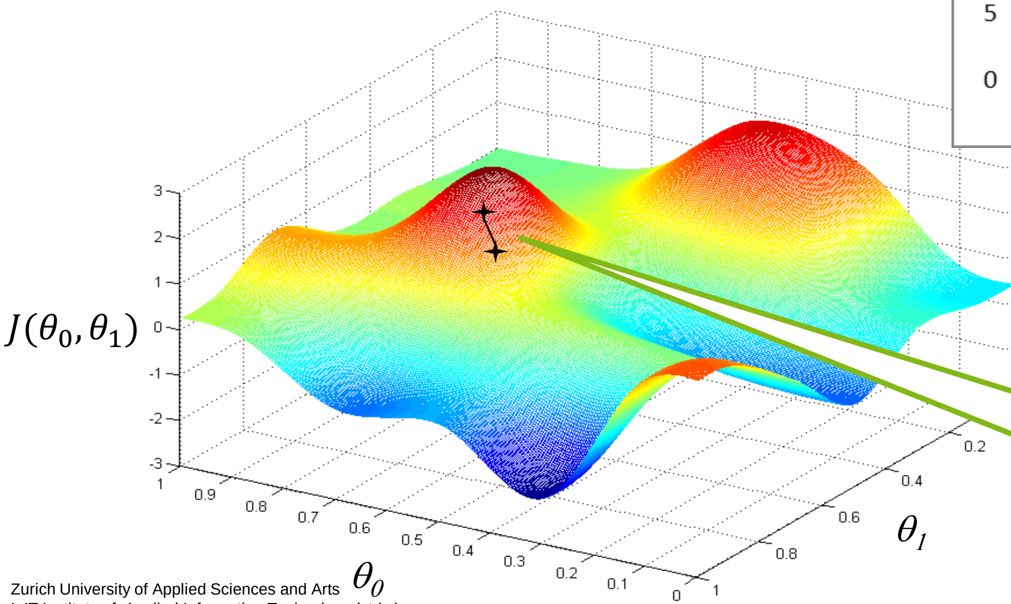
# How are the parameters found?

Probability [%] for a specific outcome




- Definition of the neural net: $f_{\vec{\theta}}(x) = y$
  with image $x$, true result $y$ and all parameters $\vec{\theta}$
  ($\vec{\theta} = \{w_1, w_2\}$ chosen randomly at start)

- Error measure: $J\left(\vec{\theta}\right) = \frac{1}{N}\sum_{i=1}^{N}\left(f_{\vec{\theta}}(x_i) - y_i\right)^2$
  Mean squared error

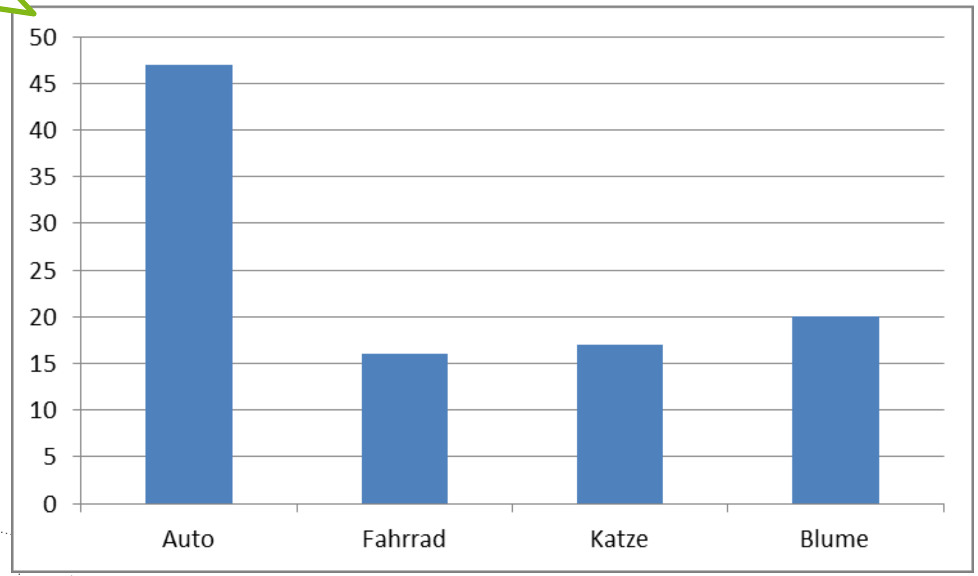



🡐 Error landscape

Method: Adaptation of weights of $f$ in the direction of the steepest gradient (descending) of $J$

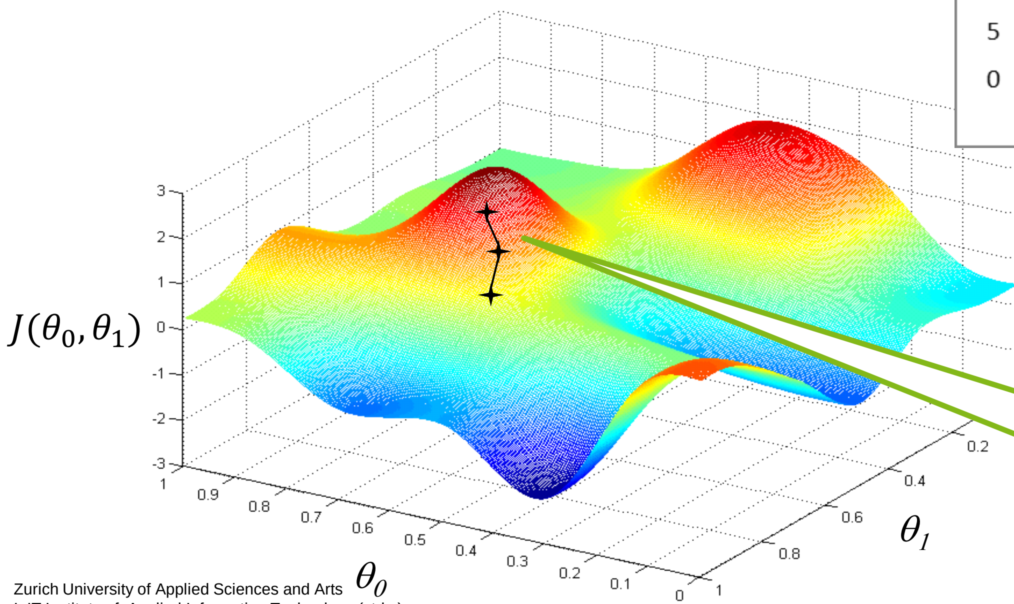
# How are the parameters found?

Probability [%] for a specific outcome




- Definition of the neural net: $f_{\vec{\theta}}(x) = y$
  with image $x$, true result $y$ and all parameters $\vec{\theta}$
  ($\vec{\theta} = \{w_1, w_2\}$ chosen randomly at start)

- Error measure: $J\left(\vec{\theta}\right) = \frac{1}{N}\sum_{i=1}^{N}\left(f_{\vec{\theta}}(x_i) - y_i\right)^2$
  Mean squared error

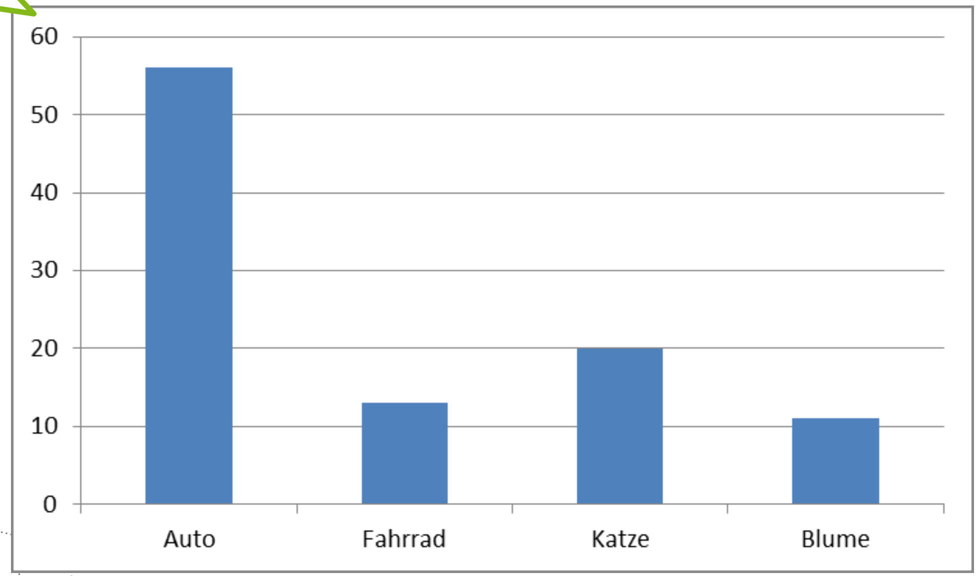



← Error landscape

Method: Adaptation of weights of $f$ in the direction of the steepest gradient (descending) of $J$

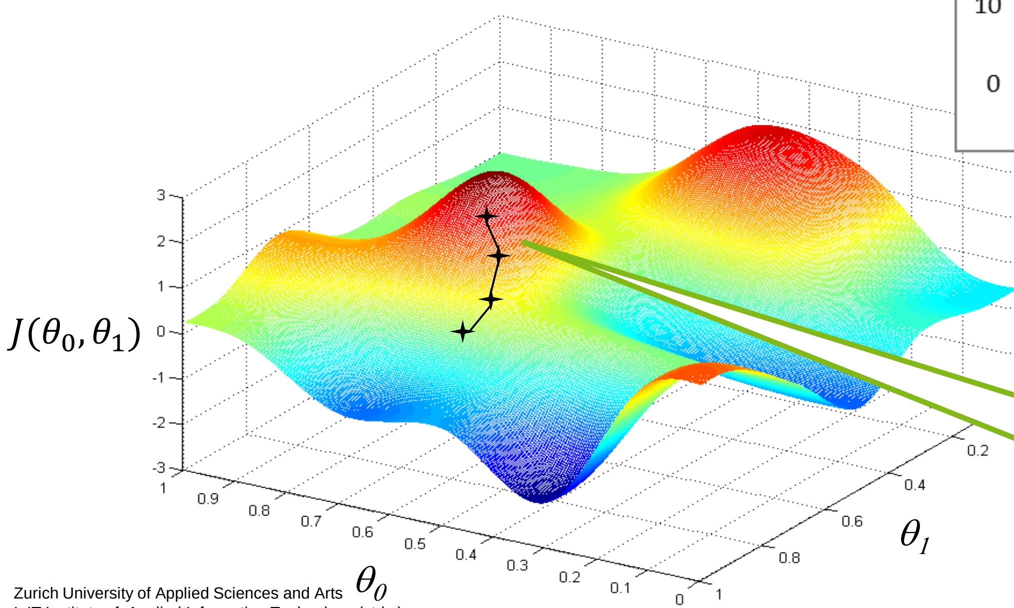
# How are the parameters found?

Probability [%] for a specific outcome




- Definition of the neural net: $f_{\vec{\theta}}(x) = y$
  with image $x$, true result $y$ and all parameters $\vec{\theta}$
  ($\vec{\theta} = \{w_1, w_2\}$ chosen randomly at start)

- Error measure: $J\left(\vec{\theta}\right) = \frac{1}{N} \sum_{i=1}^{N} \left(f_{\vec{\theta}}(x_i) - y_i\right)^2$
  Mean squared error




← Error landscape

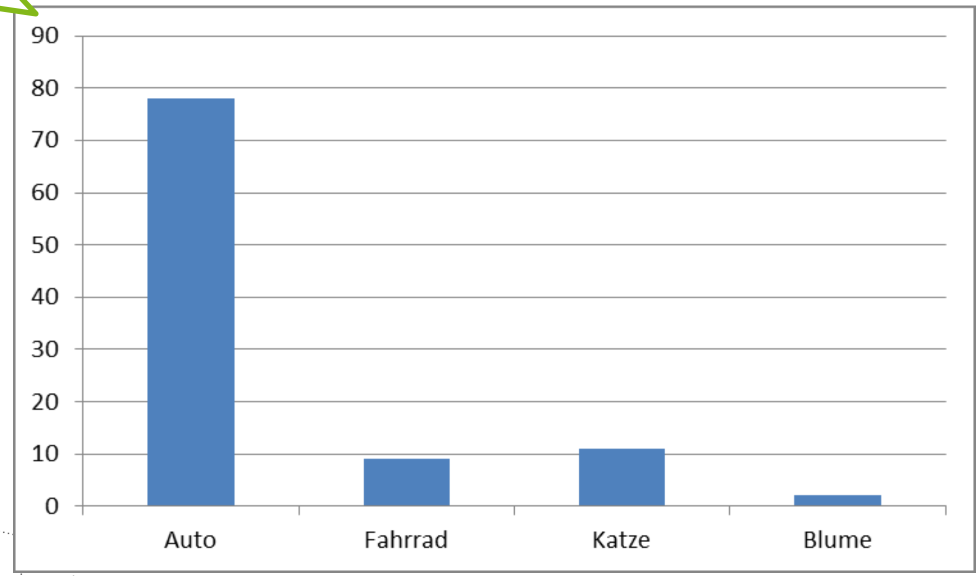
Method: Adaptation of weights of $f$ in the direction of the steepest gradient (descending) of $J$

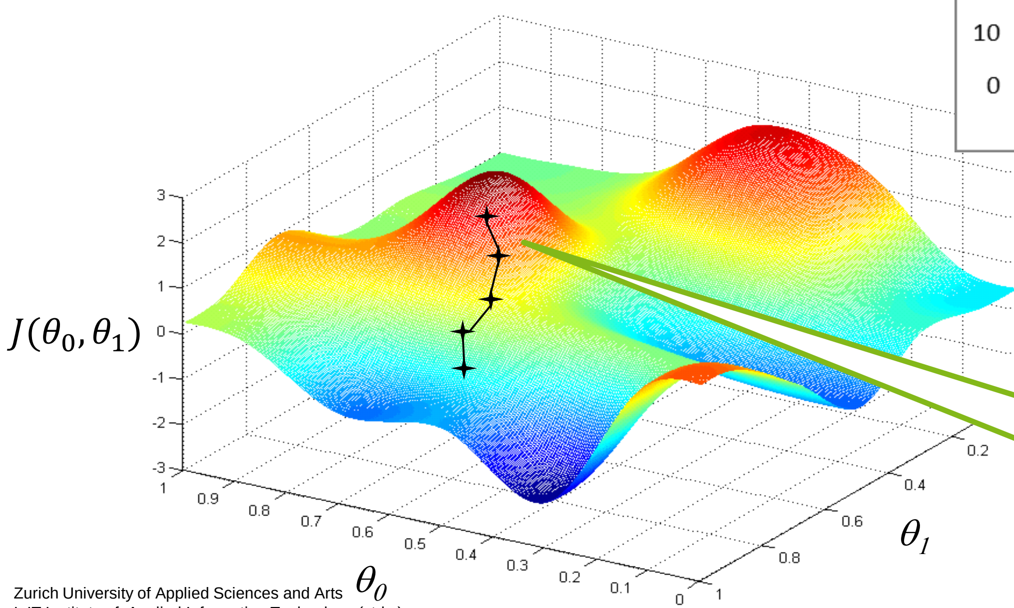
# How are the parameters found?

- Definition of the neural net: $f_{\vec{\theta}}(x) = y$
  with image $x$, true result $y$ and all parameters $\vec{\theta}$
  ($\vec{\theta} = \{w_1, w_2\}$ chosen randomly at start)

- Error measure: $J\left(\vec{\theta}\right) = \frac{1}{N}\sum_{i=1}^{N}\left(f_{\vec{\theta}}(x_i) - y_i\right)^2$
  Mean squared error

Probability [%] for a specific outcome



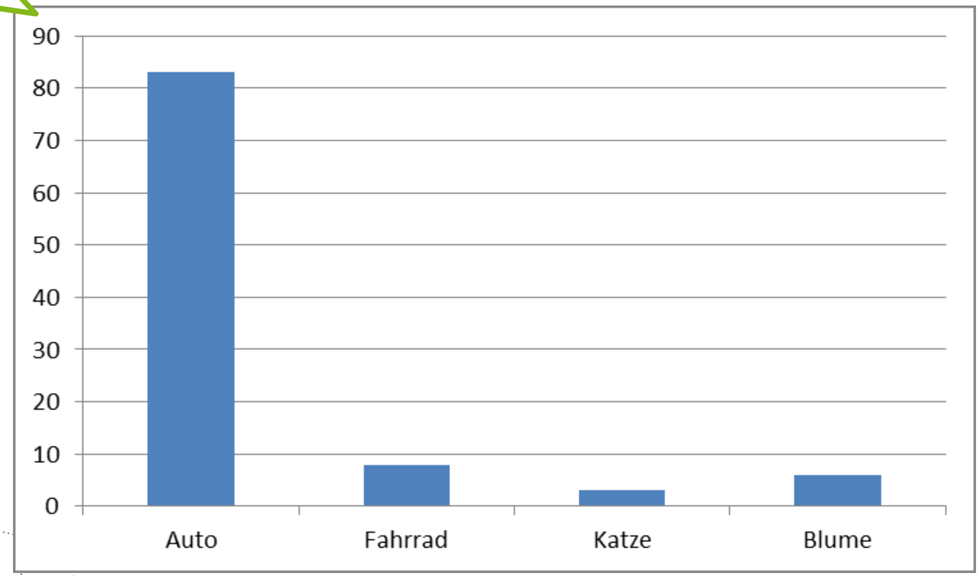




← Error landscape

Method: Adaptation of weights of $f$ in the direction of the steepest gradient (descending) of $J$

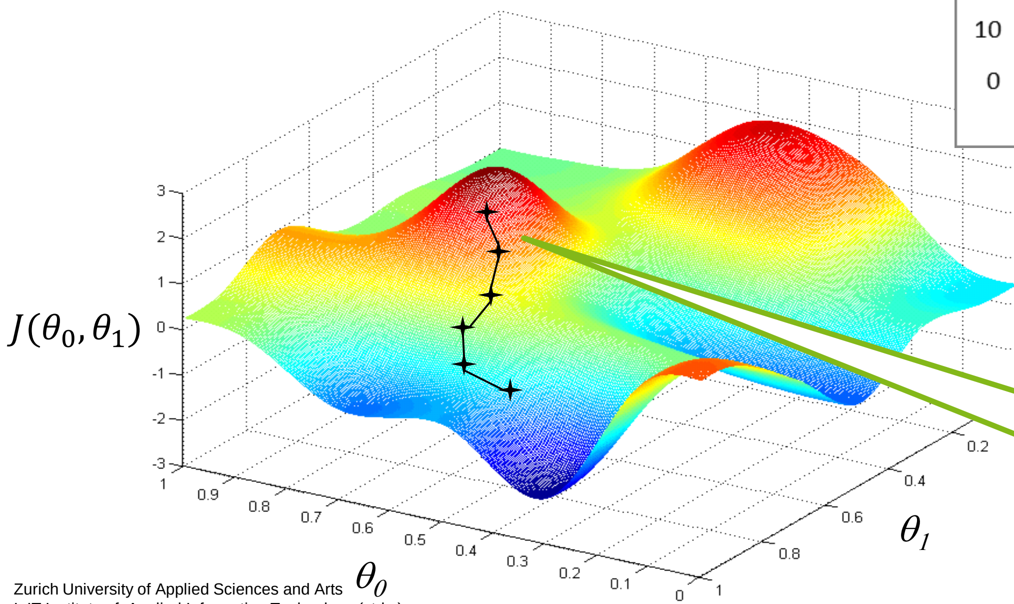
# How are the parameters found?

Probability [%] for a specific outcome




- Definition of the neural net: $f_{\vec{\theta}}(x) = y$
  with image $x$, true result $y$ and all parameters $\vec{\theta}$
  ($\vec{\theta} = \{w_1, w_2\}$ chosen randomly at start)

- Error measure: $J\left(\vec{\theta}\right) = \frac{1}{N}\sum_{i=1}^{N}\left(f_{\vec{\theta}}(x_i) - y_i\right)^2$
  Mean squared error

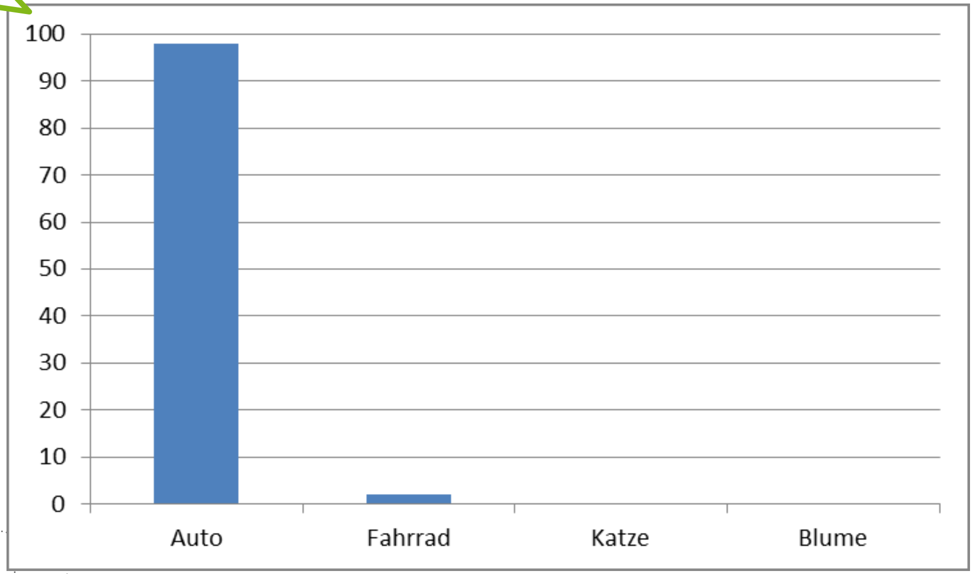



🡐 Error landscape

Method: Adaptation of weights of $f$ in the direction of the steepest gradient (descending) of $J$

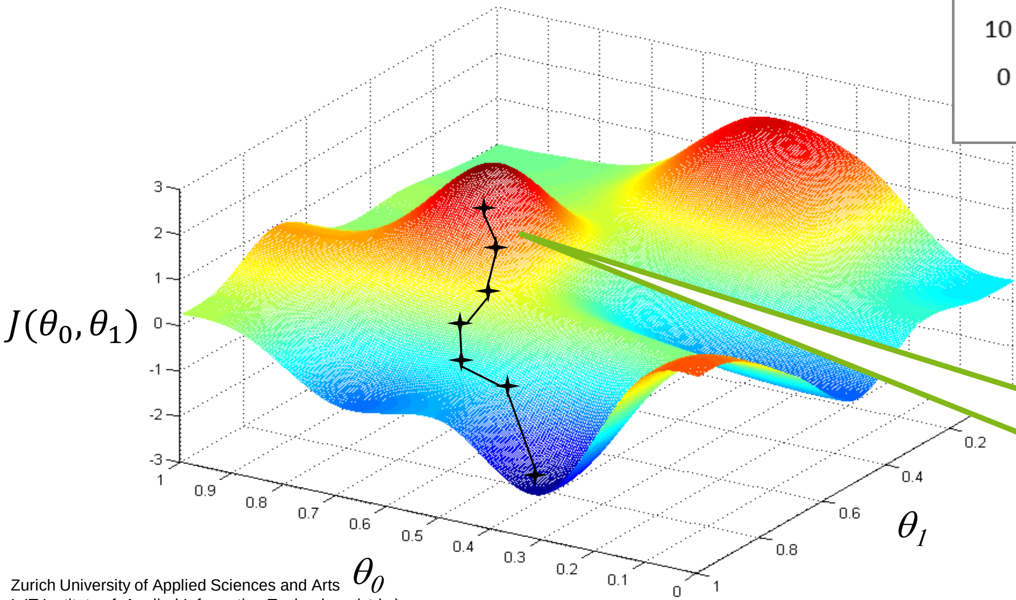
# How are the parameters found?

Probability [%] for a specific outcome




- Definition of the neural net: $f_{\vec{\theta}}(x) = y$
  with image $x$, true result $y$ and all parameters $\vec{\theta}$
  ($\vec{\theta} = \{w_1, w_2\}$ chosen randomly at start)

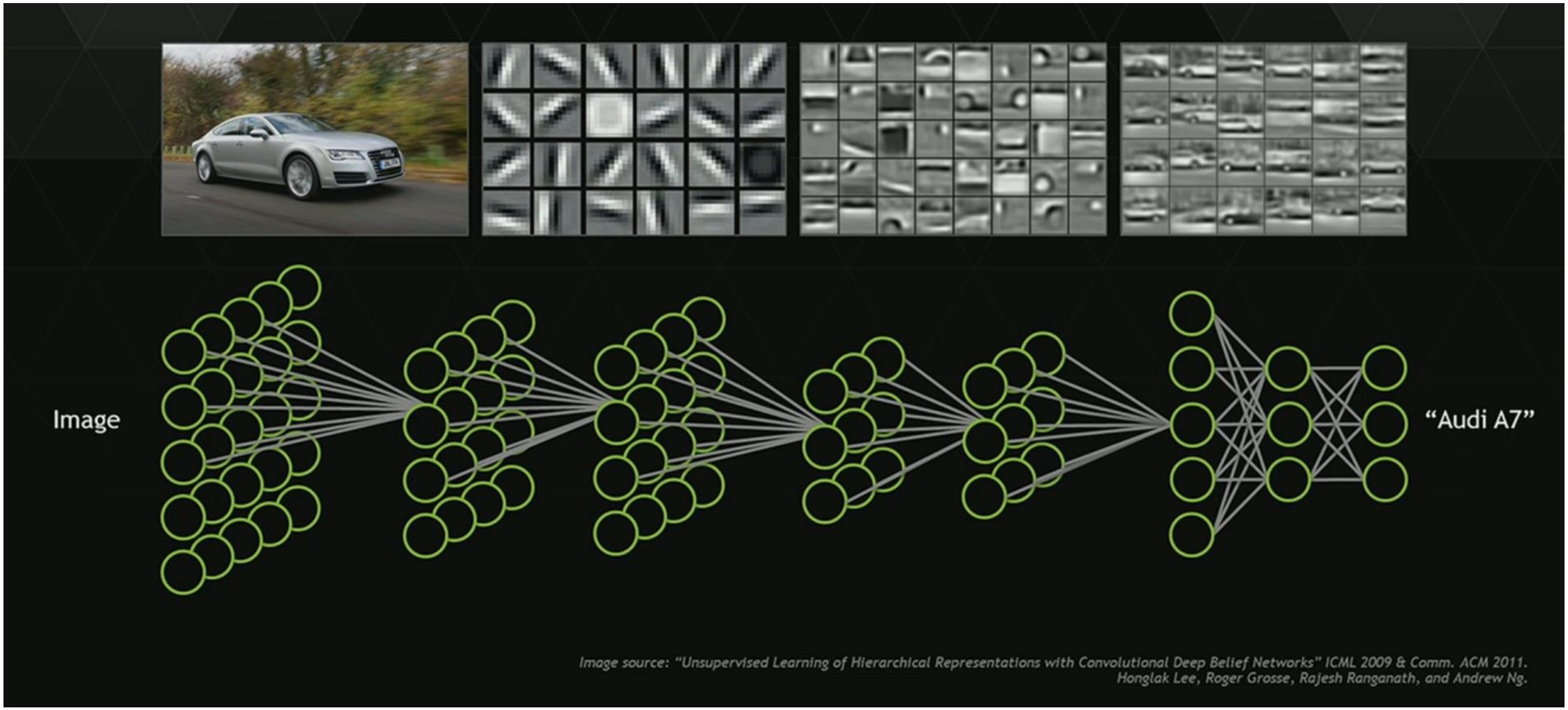
- Error measure: $J\left(\vec{\theta}\right) = \frac{1}{N}\sum_{i=1}^{N}\left(f_{\vec{\theta}}(x_i) - y_i\right)^2$
  Mean squared error




← Error landscape

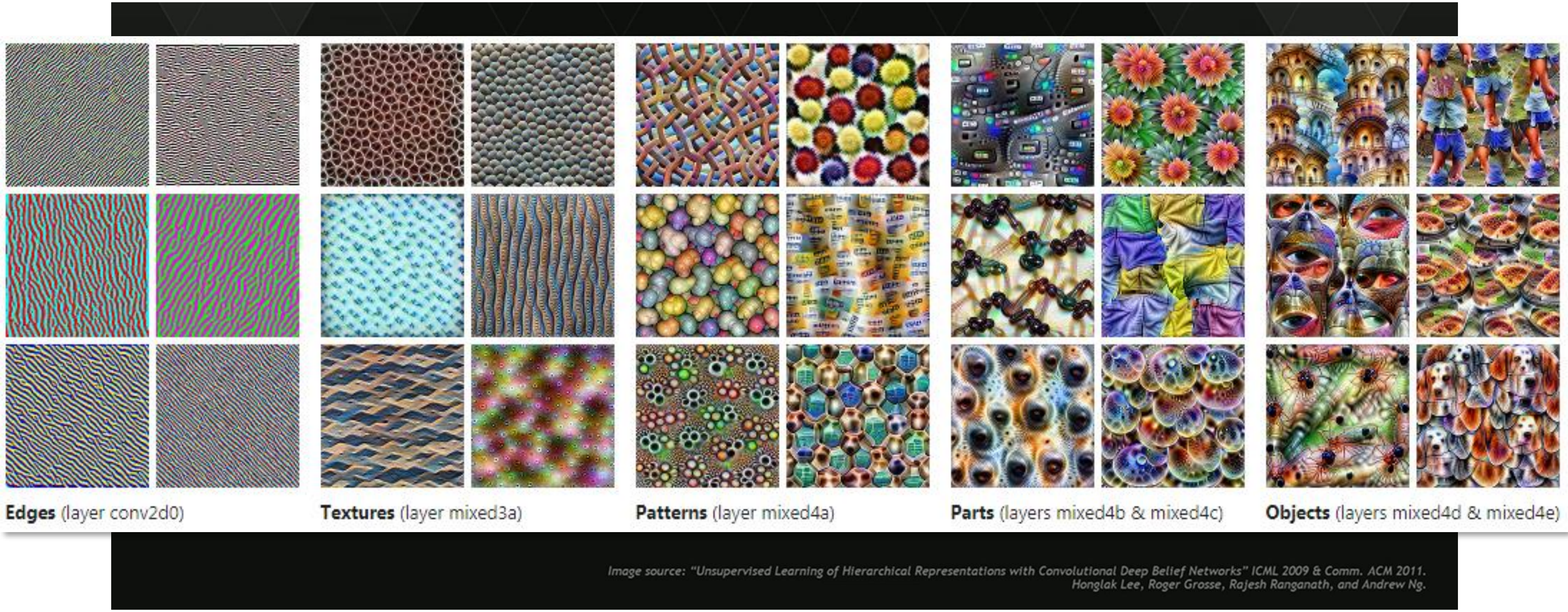
Method: Adaptation of weights of $f$ in the direction of the steepest gradient (descending) of $J$

# What does a neural network «see»?
## A hierarchy of progressively complex features



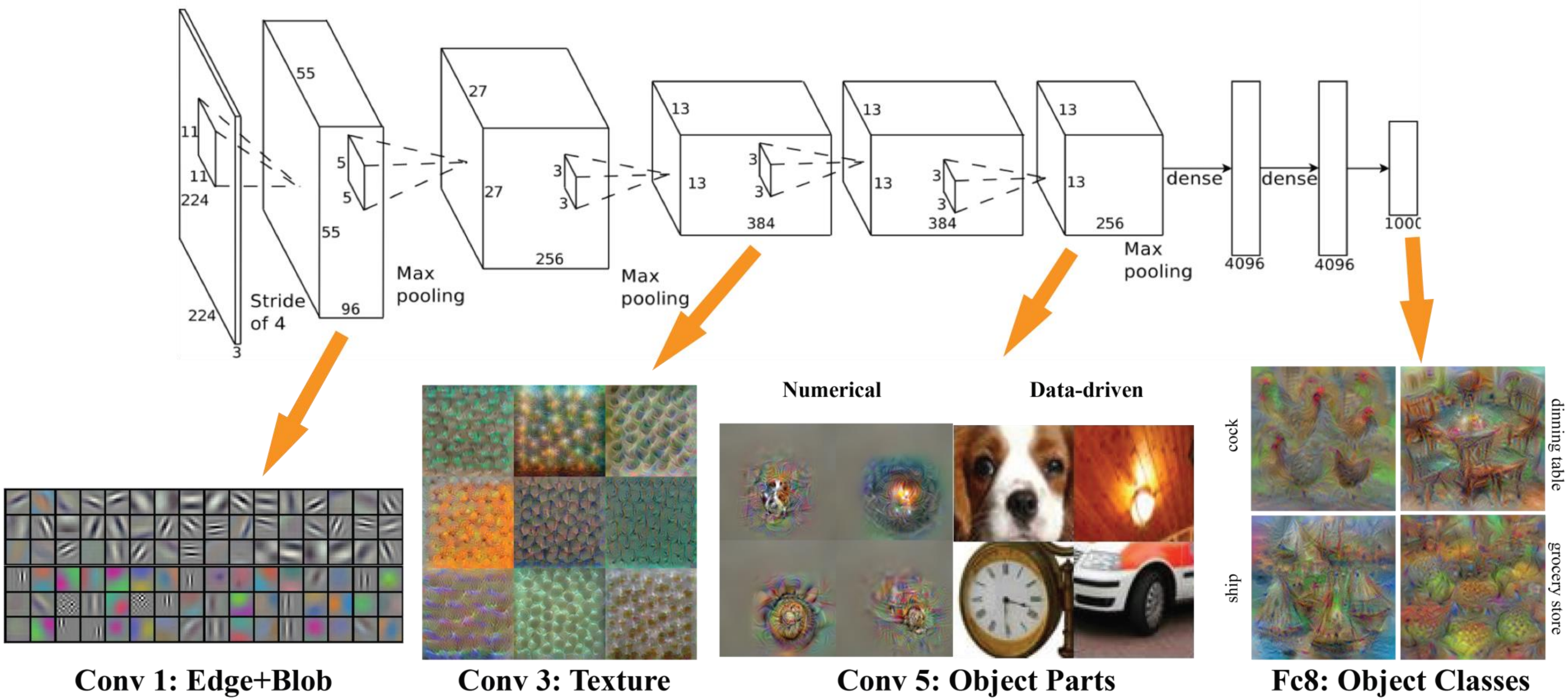

Sources: https://www.pinterest.com/explore/artificial-neural-network/
Olah, et al., "Feature Visualization", Distill, 2017, https://distill.pub/2017/feature-visualization/.

# What does a neural network «see»?
## A hierarchy of progressively complex features




Sources: https://www.pinterest.com/explore/artificial-neural-network/
Olah, et al., "Feature Visualization", Distill, 2017, https://distill.pub/2017/feature-visualization/.

# What does a neural network «see»?
## A hierarchy of progressively complex features, visualized




Source: http://vision03.csail.mit.edu/cnn_art/data/single_layer.png