Zurich University
of Applied Sciences

Artificial Intelligence
V08: Supervised Learning using Neural Networks

Introduction to learning agents
Artificial Neural Networks
Doing machine learning

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

Zurich University of Applied Sciences and Arts
CAI Centre for Artificial Intelligence (stdm)

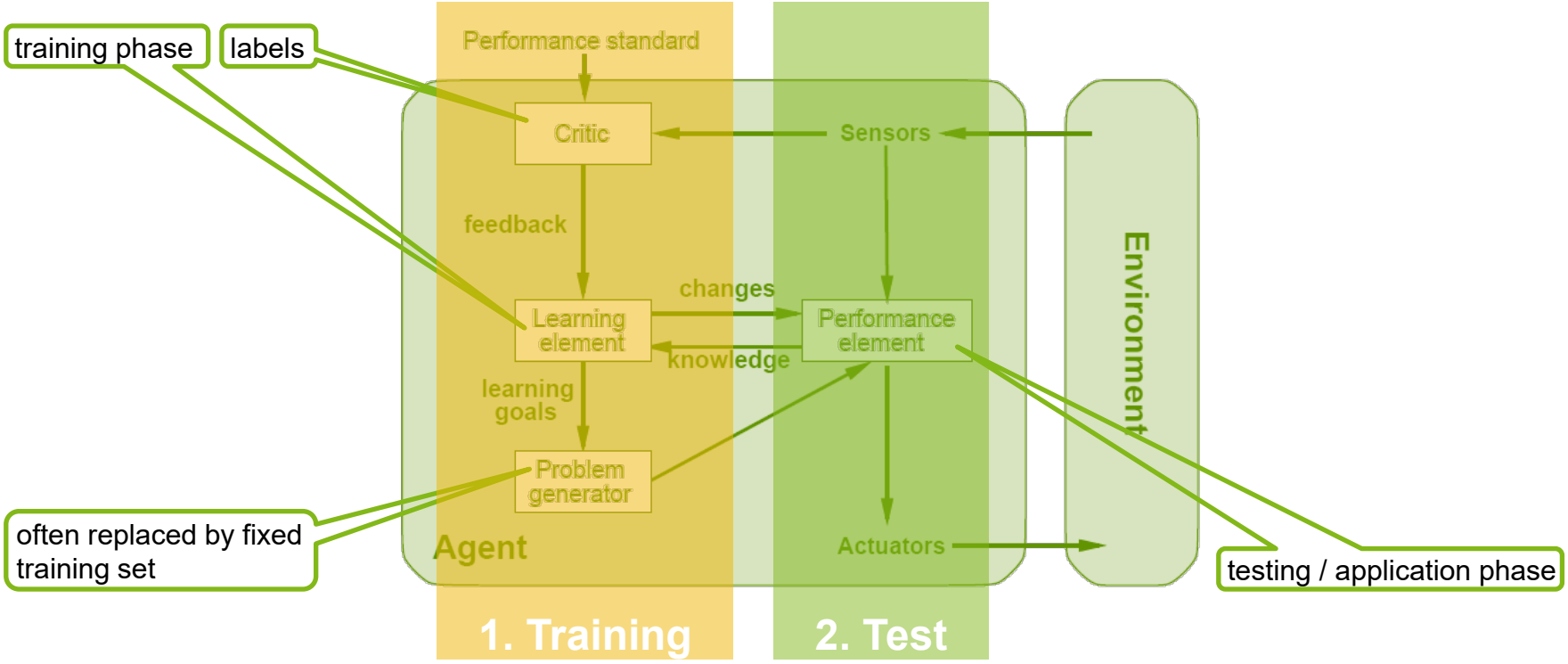
Educational objectives

- **Remember** the correct **technical terms** to explain **machine learning**
- Have a **basic understanding** of the architecture and working **of neural networks**
- **Defend** your **own view on** the existence of good **general learners**

"In which we describe agents that can improve their behavior through diligent study of past experiences and predictions about the future."

➔ Reading: AIMA, ch. 19-19.6

1. INTRODUCTION TO LEARNING AGENTS



The discipline of machine learning – mapped

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

ML

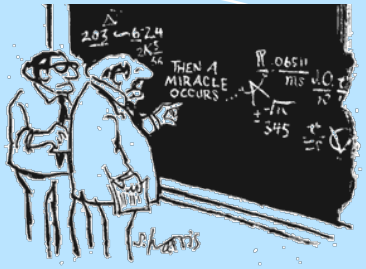
Famous: used in most **human** learning, definition of **scientific method**

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 …

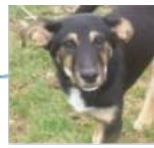
- parametric / non-parametric → fixed size / growing with data
- discriminative / generative → learn boundary / blueprint
- predictive / inferential → black box / explanatory

Supervised machine learning in a nutshell

Training data points, represented by some feature vector $\boldsymbol{x}$

This model seems neither to overfit nor underfit



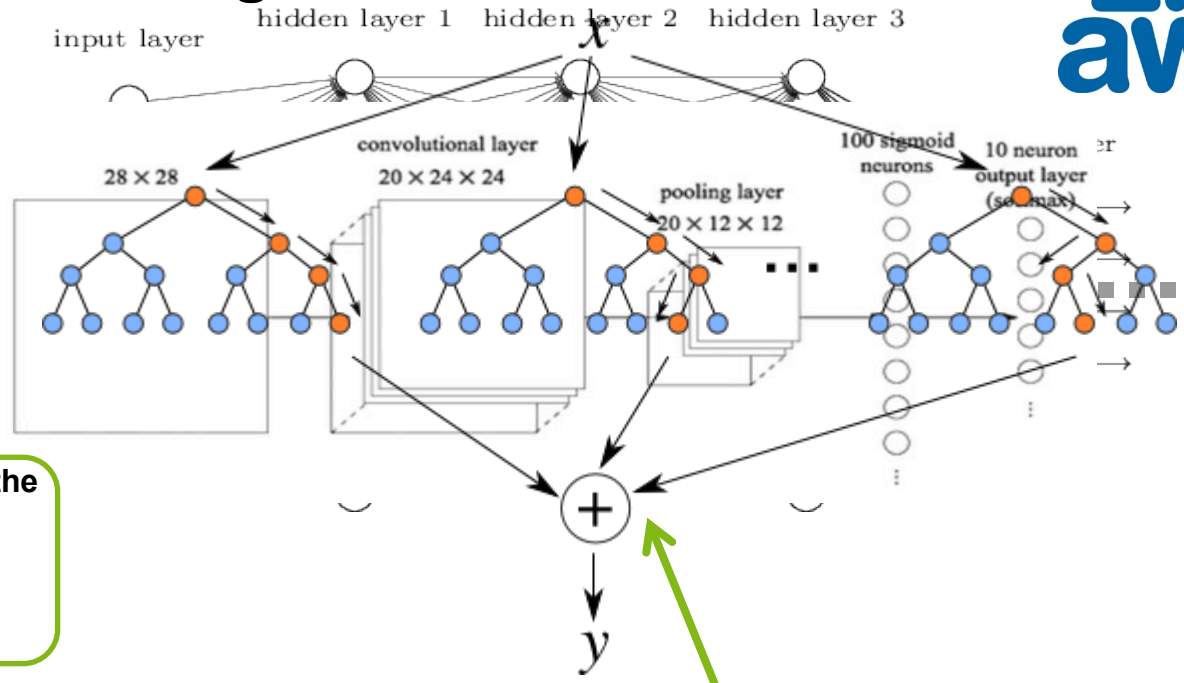

$$\arg\min_{h \in \mathcal{H}} \sum_{(\boldsymbol{x},y) \in D} \ell\big(y, h(\boldsymbol{x})\big)$$

We hope (and design) for good generalization to unseen test data

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

Learning as search through $\mathcal{H}$

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



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 θ)

Shallow vs. deep learning

Add depth (layers ➔ capability) to learn features automatically

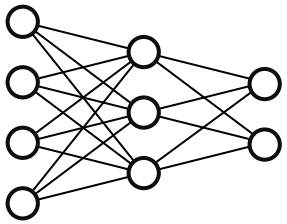
Classic computer vision

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

(0.2, 0.4, …)

(0.4, 0.3, …)

Classification (SVM, neuronal net, etc.)



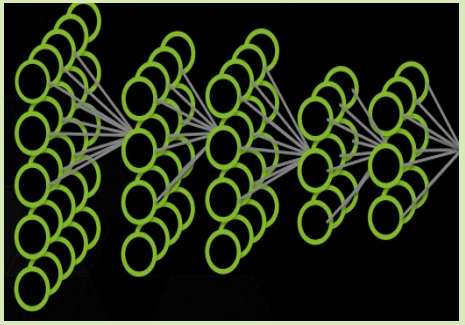
container ship

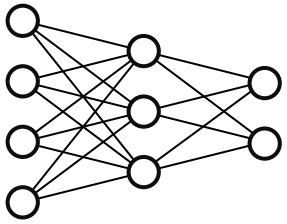
tiger

…

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

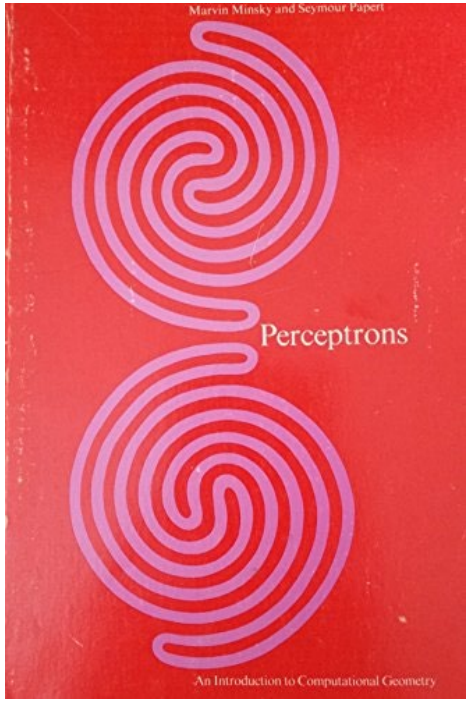
Examples of sensor data for pattern recognition tasks («Labeled faces in the wild» dataset) and tabular data («Iris» dataset)

Ascertainment from kaggle.com

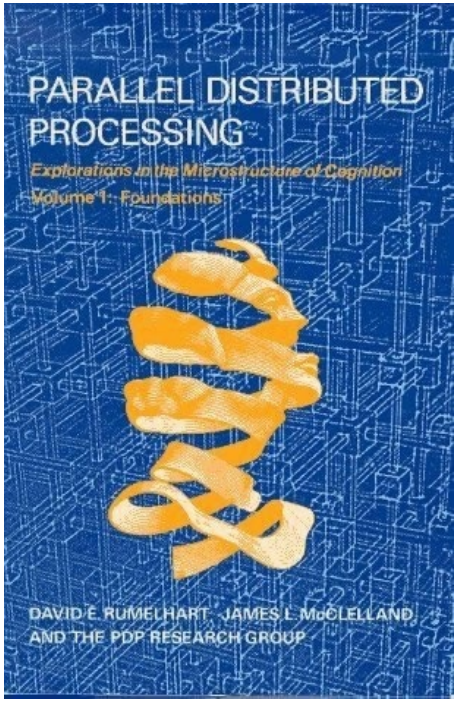
- **Tabular** data: do handcrafted feature engineering, followed by an **ensemble of decision trees**
- **Sensor** data (images, speech, …): use a *suitable* **deep neural network**

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

2. ARTIFICIAL NEURAL NETWORKS



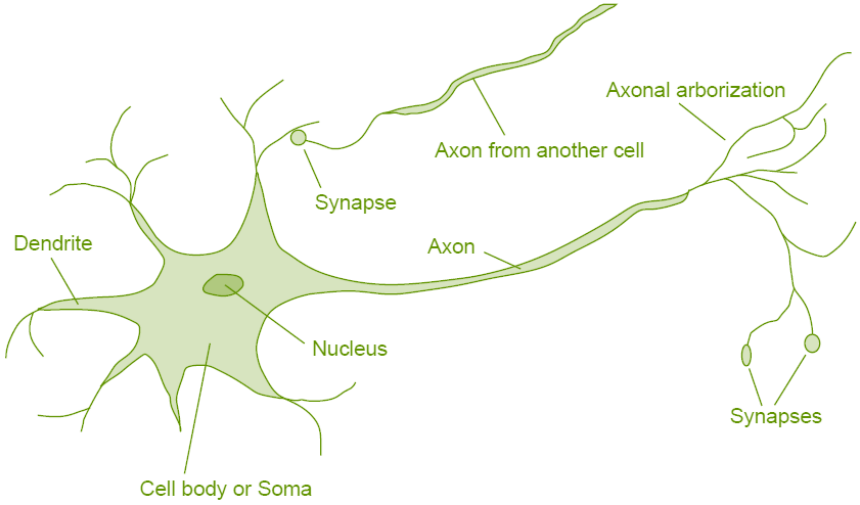
The critique of the first neural network hype:
Minsky & Papert, “Perceptrons”, MIT Press Ltd, 1969



…and its resurrection:
Rumelhart et al., “Parallel Distributed Processing, Volume 1”, MIT Press Ltd, 1987

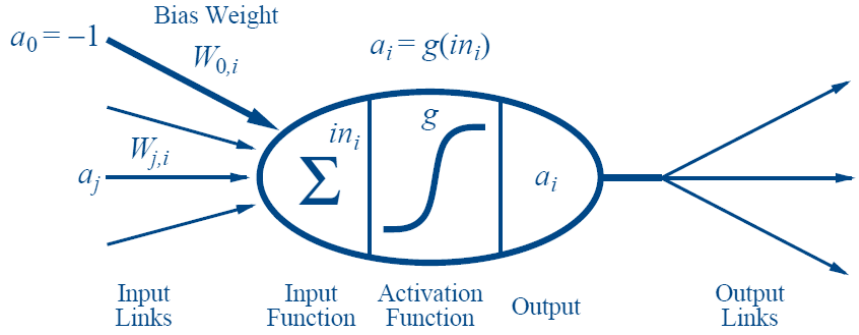
Neurons

Biological model



- 10^{11} neurons of > 20 types
- 10^{14} synapses
- 1ms – 10ms cycle time
- Signals are noisy “spike trains" of electrical potential
- Organized in layers to form a brain

McColloch-Pitts “unit” (1943)

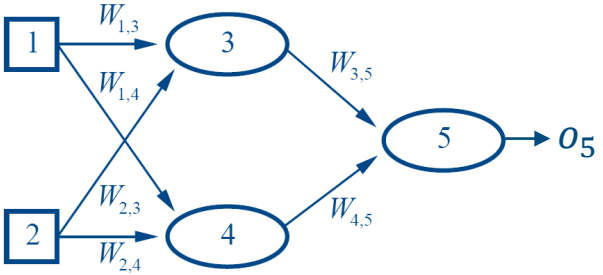


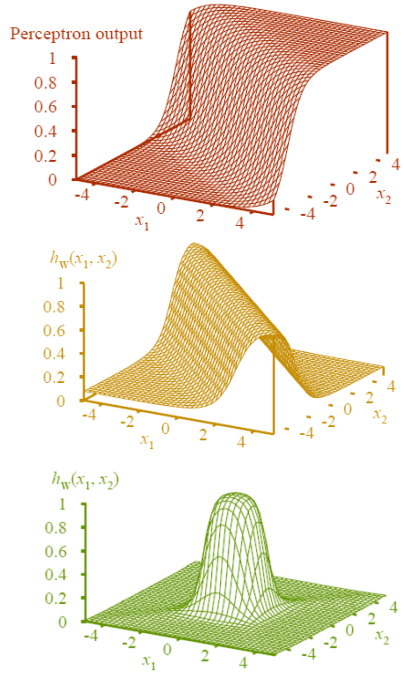
- Output is a **thresholded linear function** of the inputs: $a_i = g(in_i) = g\left(\sum_j W_{j,i} \cdot a_j\right)$
- Changing the bias weight $W_{0,i}$moves the threshold location
- A **gross oversimplification** of real neurons!
- Purpose: develop understanding of what **networks of simple units** can do

Feed-forward network example

FNN: a parameterized family of nonlinear functions $g()$

- $o_5 = g(W_{3,5} \cdot o_3 + W_{4,5} \cdot o_4)$
  $= g\left(W_{3,5} \cdot g(W_{1,3} \cdot o_1 + W_{2,3} \cdot o_2) + W_{4,5} \cdot g(W_{1,4} \cdot o_1 + W_{2,4} \cdot o_2)\right)$





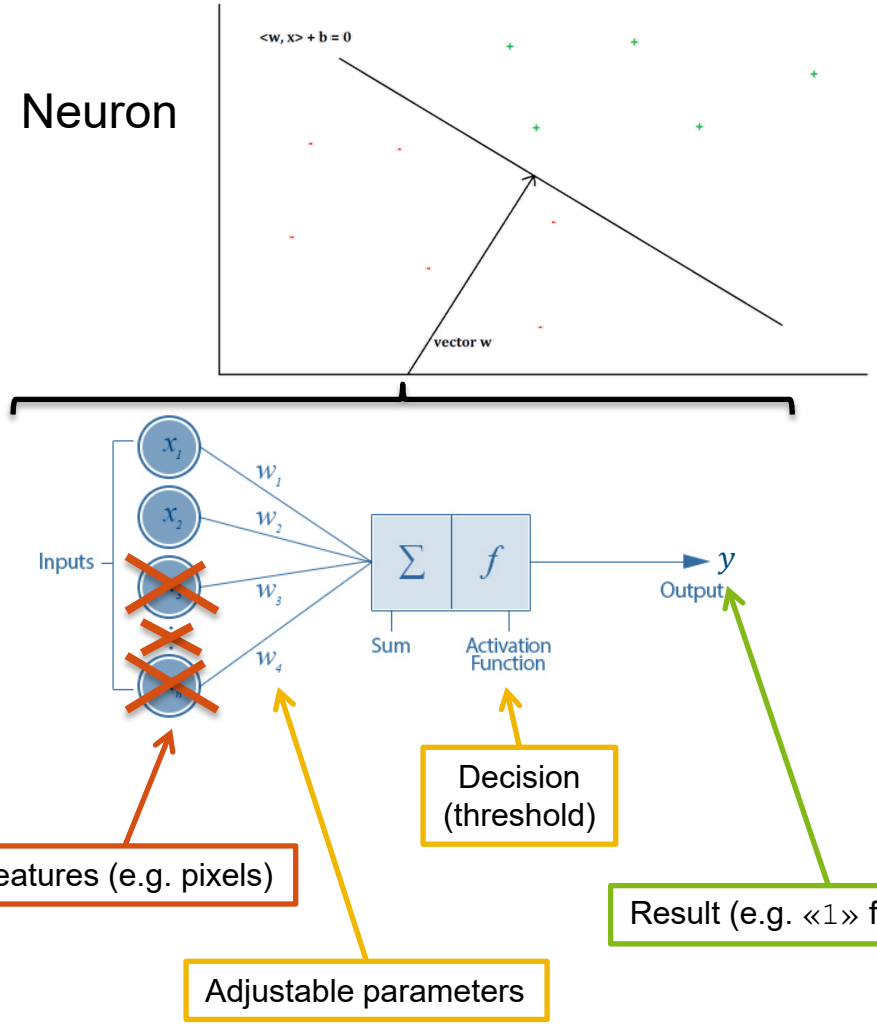
- **Adjusting weights** changes the function: **learning** works this way!
  (→ see appendix for first ideas)

Expressiveness of multilayer networks (multilayer perceptrons)

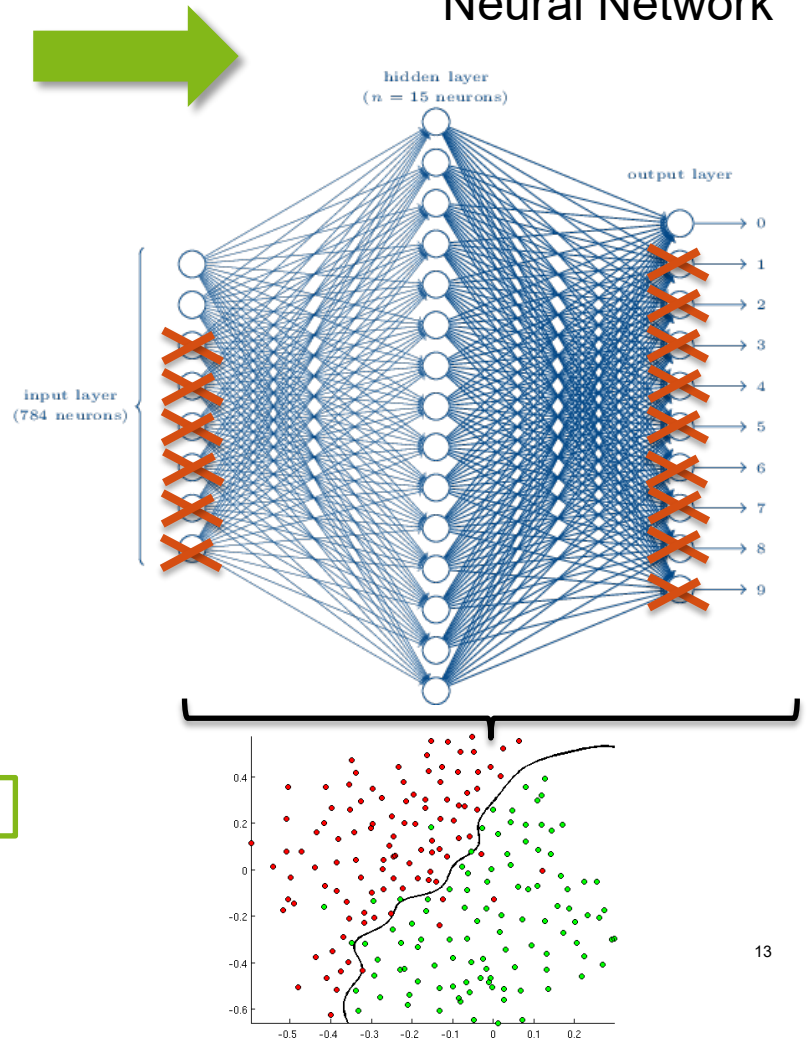
- All continuous functions w/ 2 layers, all functions w/ 3 layers
  - Combine two opposite-facing threshold functions to make a ridge
  - Combine two perpendicular ridges to make a bump
  - Add bumps of various sizes and locations to fit any surface

What is the effect of weight adjustment?

Neuron

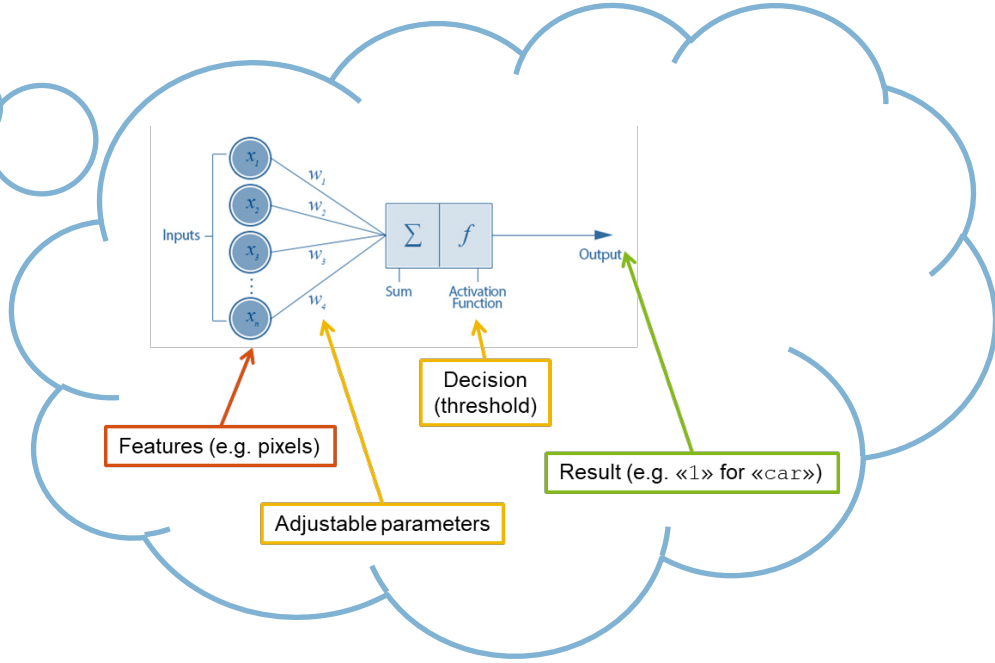


Neural Network



How are the weights adjusted?
First intuition

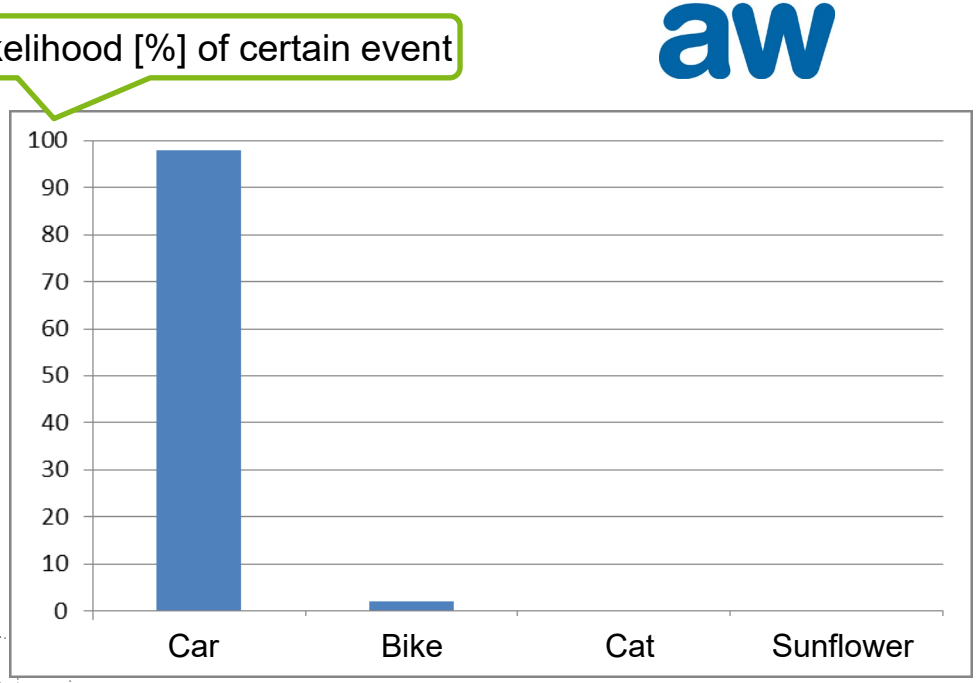
- Our example neural network: $f_W(x) = y$
  with image x, ground truth y und parameters W
  ($W = \{w_1, w_2, \ldots\}$ initialized randomly)

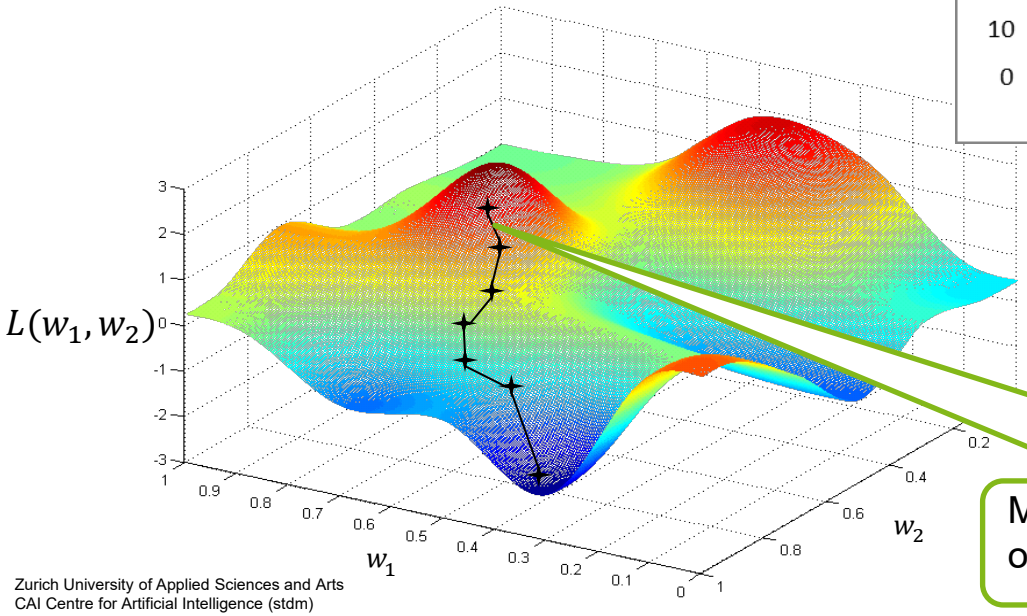


How are the weights adjusted? (contd.)

First intuition

- Our example neural network: $f_W(x) = y$
  with image x, ground truth y und parameters W
  ($W = \{w_1, w_2, \dots\}$ initialized randomly)
- Error measure: $L(W) = \frac{1}{N}\sum_{i=1}^{N}(f_W(x_i) - y_i)^2$
  Average of quadratic difference on all images (loss function)





← error «landscape»

Method: adapt weights of f in the direction of the steepest descent (downwards) of L

How are the weights adjusted?

Neural network training ideas

→ see also https://stdm.github.io/downloads/papers/ADS_2019_DeepLearning.pdf

Trained by gradient descent (complete network is differentiable)

- Forward pass: calculation of loss function L for a mini batch of training examples
- Backward pass: calculation of $\frac{\partial L}{\partial W_{l,i}}$ for each weight $W_{l,i}$ on overall loss
  - Efficiently computable by layer-wise application of chain rule (backpropagation algorithm)

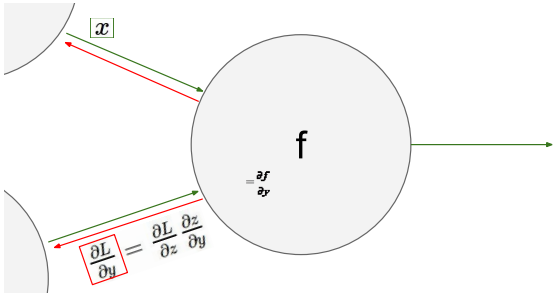


Illustration:
http://cs231n.stanford.edu/slides/winter1516_lecture4.pdf

Many details to be considered for training to work in practice

- Weight initialization: choose random initial weights according to the magnitude of the inputs
- Gradient flow: secure sufficient gradient magnitude for fast training convergence via batchnorm
- Learning rate: choose adaptive learning rates, e.g. using the ADADELTA optimizer
- Batch composition: care for sufficient randomness in the presentation order
- Regularization: use dropout to overcome the problem of more parameters then input data

Convolutional Neural Networks (CNNs)

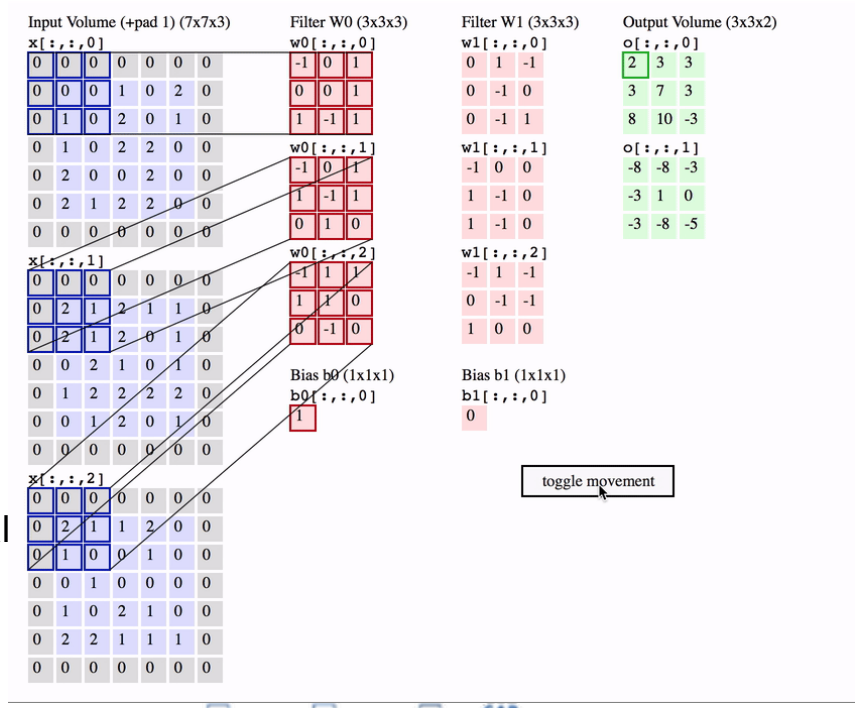
Intuition: cp. https://medium.freecodecamp.org/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050

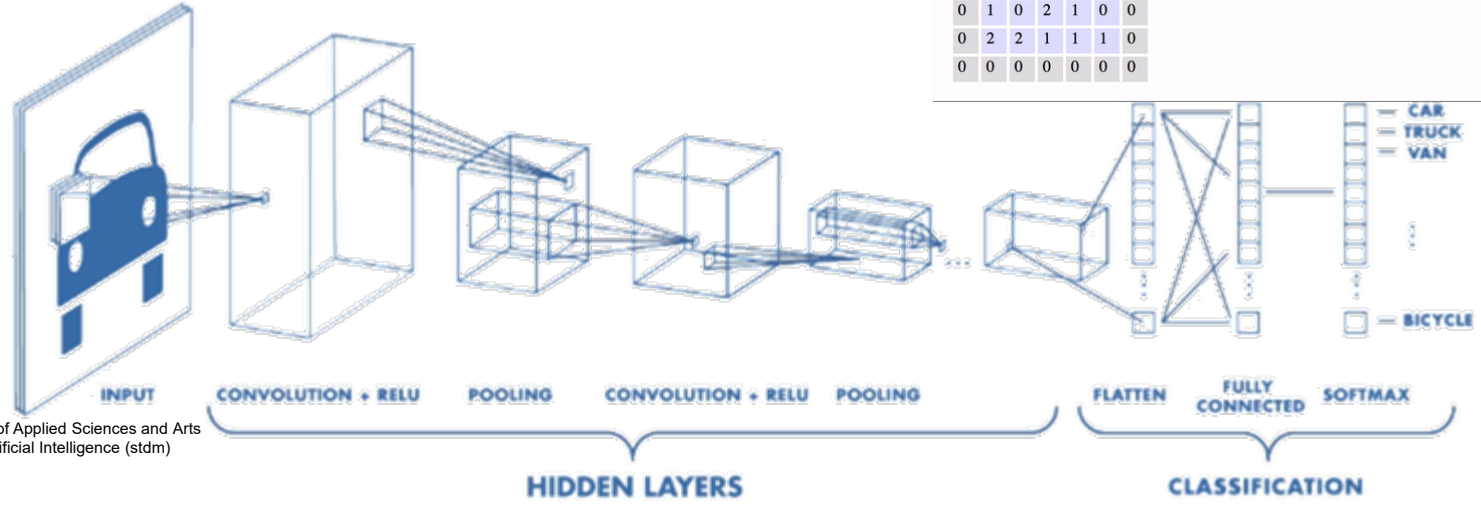
Goal: fewer free parameters → eases learning

Idea: exploit 2D-correlated local structure in (image) input data
→ inspired by mammal visual cortex

Principle

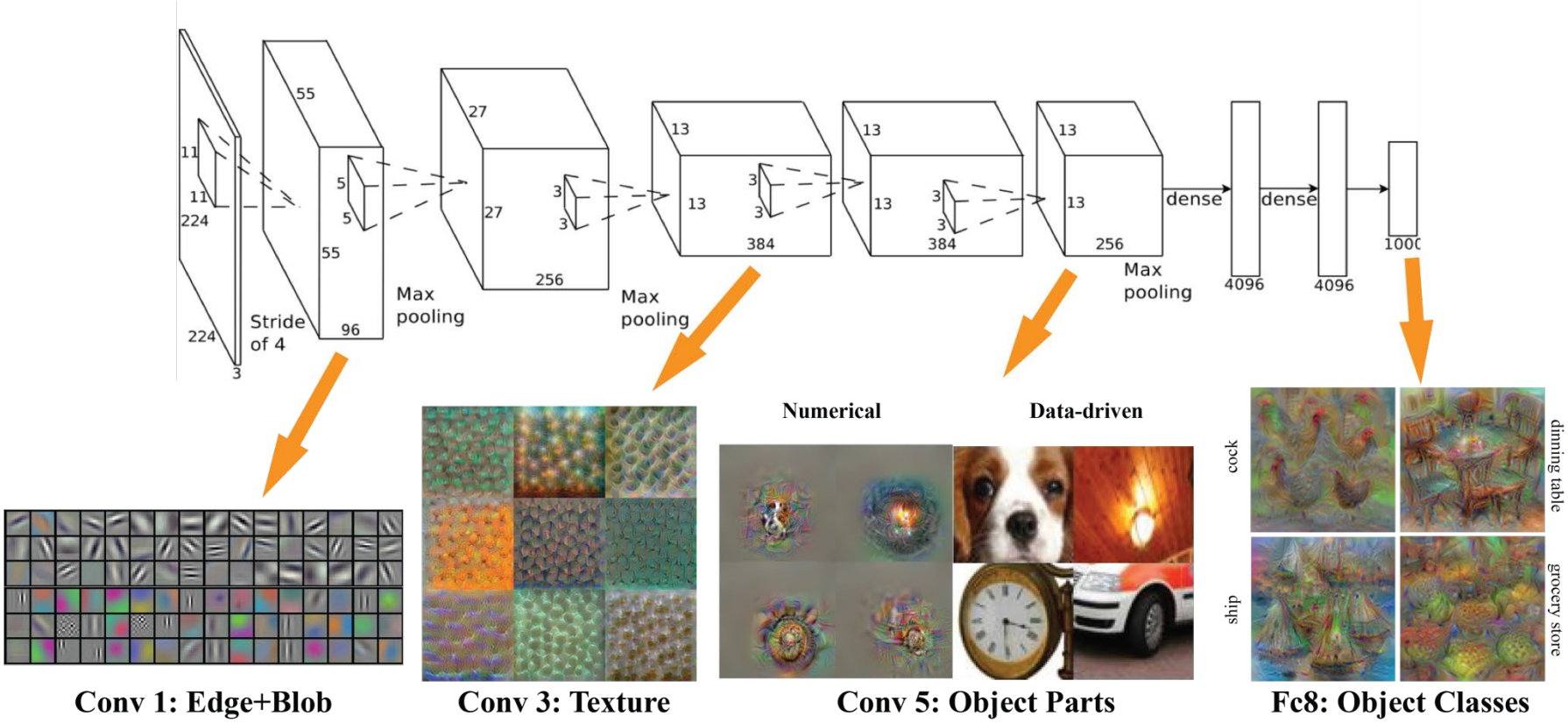
- A "**filter**" moves over every input pixel and calculates a feature that **describes** the **pixel's local context**
  → map result to same spatial location
  → filter weights (i.e., feature meaning) is trainable
- Have **several such** "filters" to encode different features
- After each filtering layer, **sub-sample** result to reduce spatial resolution and increase "field of vision"





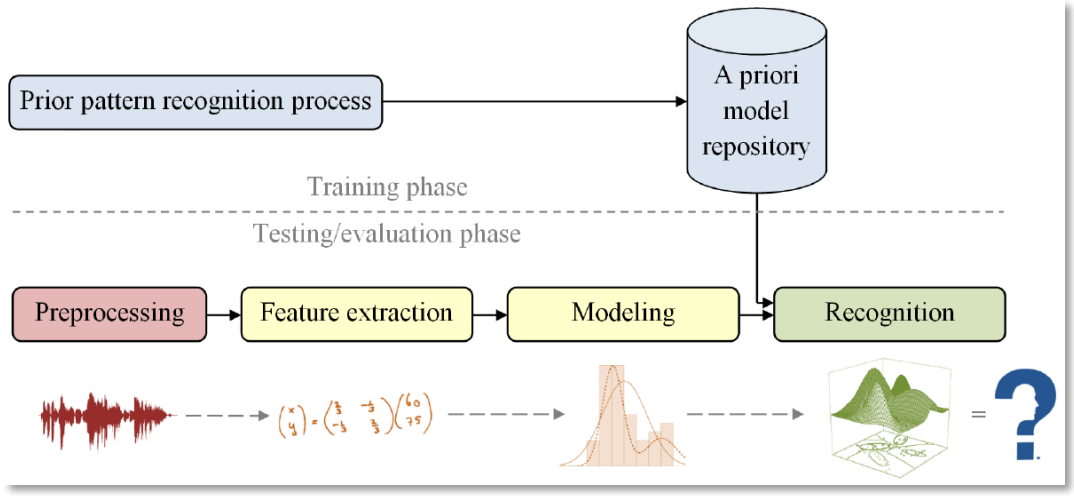
What does a convnet «see»?

A hierarchy of progressively complex features, visualized



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

3. DOING MACHINE LEARNING



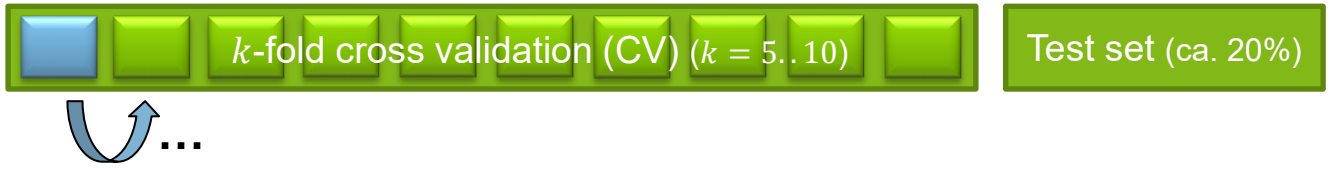
Performance measurement

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



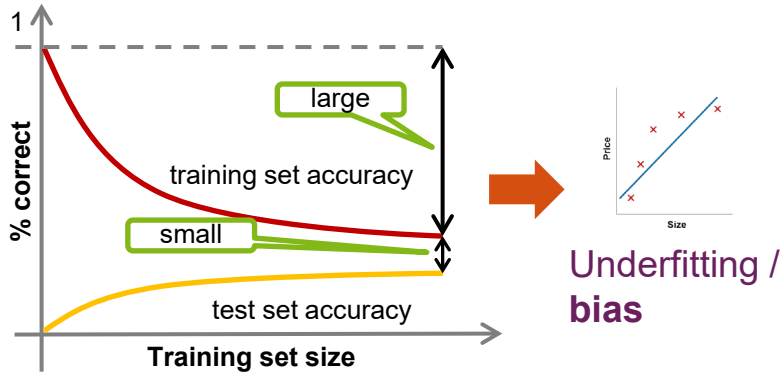
3. Report performance using recognized figures of merit
   - E.g. accuracy (or test set error) if all errors are equally costly: $\text{accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$
   - E.g. recall/precision if false alarms and misses differ in cost: $\text{recall} = \frac{TP}{TP+FN}$, $\text{precision} =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

classification → ↓ label	1	0
1	true positive (TP, "hit")	false negative (FN, "miss")
0	false positive (FP, "false alarm")	true negative (TN)

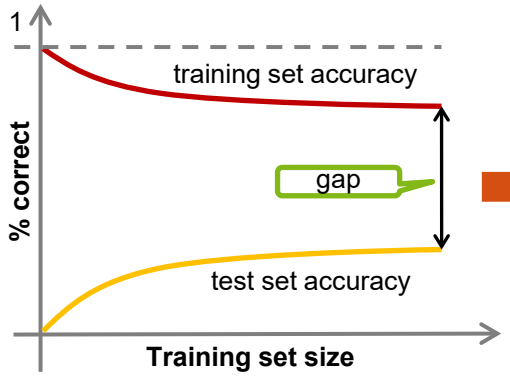
Debugging machine learning models

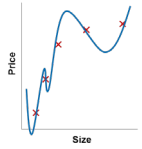
Learning curve: **%correct on train & test** set **as** a **function of training** set size

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



Underfitting /
bias





Overfitting /
variance

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)

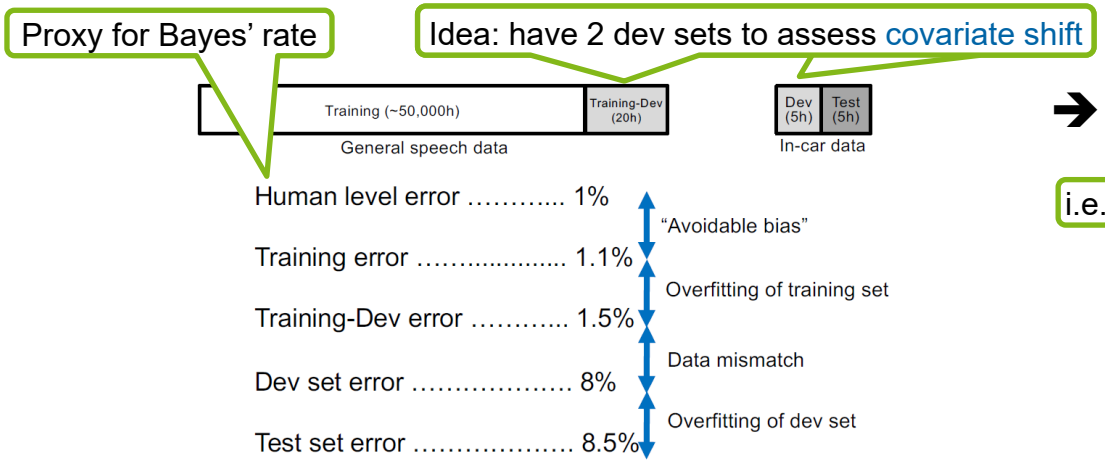
Regularization: Any method that **limits** the **expressiveness of** the **hypothesis space** by adding constraints to learning; e.g., pruning decision trees.

Bias & variance in the deep learning era

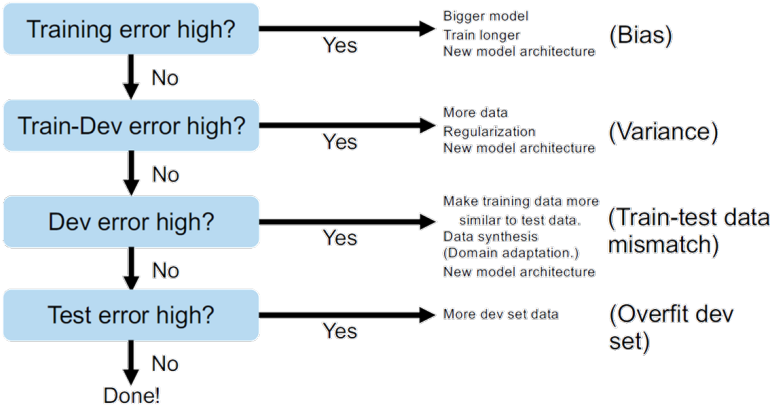
A. Ng‘s tutorial „Nuts and bolts of building AI…“ @ NIPS 2016

Today‘s situation

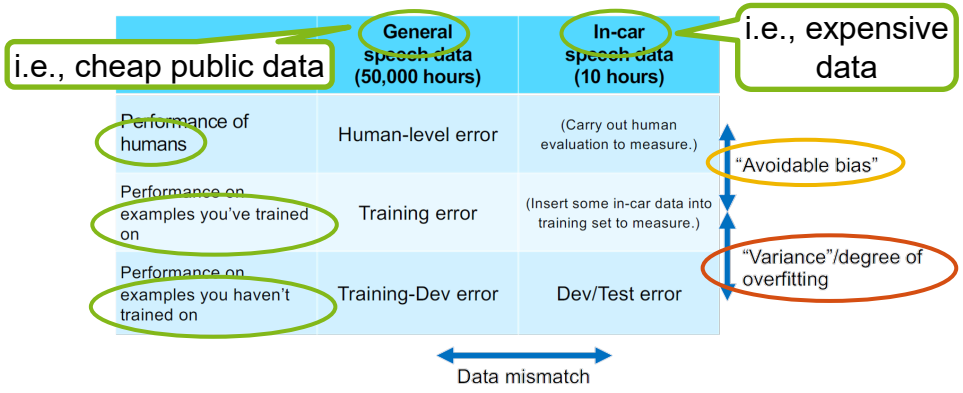
- We usually combine data sets from different distributions → **make validation and test data similar to obtain insight**
- Example in-car speech recognition: Spread special in-car data to dev (validation) and test set!



➔ New recipe for machine learning



➔ General human/bias/variance analysis

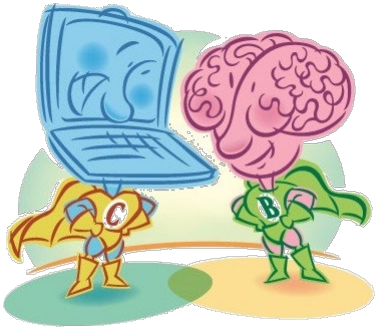


Where's the intelligence?
Man vs. machine

- 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) and reasoning
  → current avant-garde deep learning research explores this route (→ see e.g., GANs in V11)



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**
- **Artificial neural networks** are inspired by biology, but **no “brain in a jar”**
  - **Popular** models because of feature-learning capabilities
  - Work especially well for pattern recognition
  - Difficult to interpret and maintain → give rise to new fields of **XAI** and **MLOps**
- 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)

Zurich University
of Applied Sciences

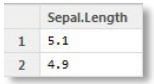
APPENDIX

Decision Trees
Deep Learning Success Story

Zurich University
of Applied Sciences

DECISION TREES

Attribute-based representations of data

Valid for all kinds of data ( , )

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

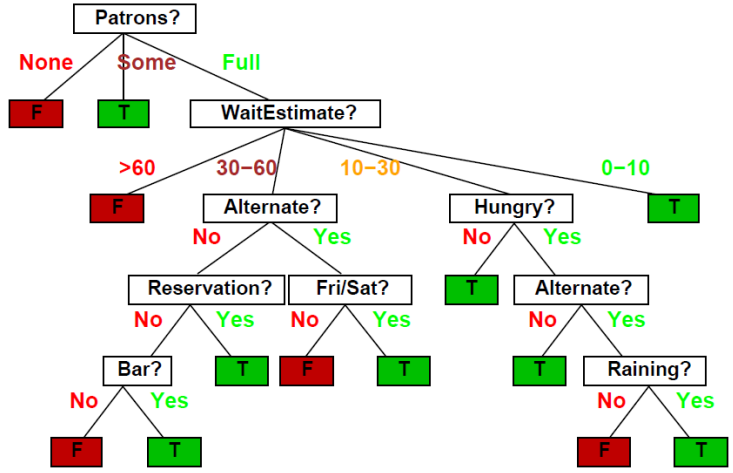
Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	*T*	*F*	*F*	*T*	*Some*	*$$$*	*F*	*T*	*French*	*0–10*	*T*
X_2	*T*	*F*	*F*	*T*	*Full*	*$*	*F*	*F*	*Thai*	*30–60*	*F*
X_3	*F*	*T*	*F*	*F*	*Some*	*$*	*F*	*F*	*Burger*	*0–10*	*T*
X_4	*T*	*F*	*T*	*T*	*Full*	*$*	*F*	*F*	*Thai*	*10–30*	*T*
X_5	*T*	*F*	*T*	*F*	*Full*	*$$$*	*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*

$\boldsymbol{x}$ $\boldsymbol{y}$

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

Decision tree representation of hypotheses

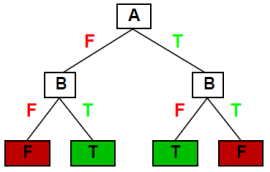
Example: Stuart Russell's *"true"* tree to **decide whether to wait** in a restaurant



Expressiveness

- Decision trees can **express any function** of the input attributes
  E.g. for Boolean functions: truth table row ➔ path to leaf

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



- **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

Hypothesis spaces

Even a constrained hypothesis space is large

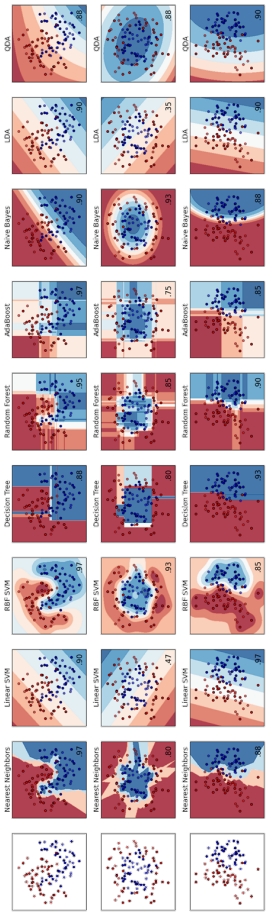
- **How many distinct** decision **trees** with n Boolean attributes?
  - $=$ number of Boolean functions
  - $=$ number of distinct truth tables with 2^n rows $= 2^{2^n}$
  - Example: **6** Boolean **attributes** ➔ $18'446'744'073'709'551'616$ possible trees

- 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

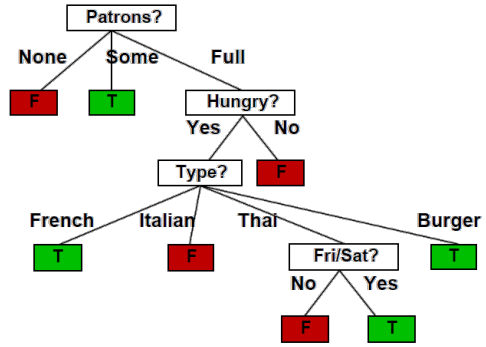
- …increase chance that **target** function can be **expressed** ☺
- …increases **number** of hypotheses **consistent** w/ training set ➔ **may get worse** predictions ☹

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 label then return the label
    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; for numerical features, use e.g. binning
            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
```

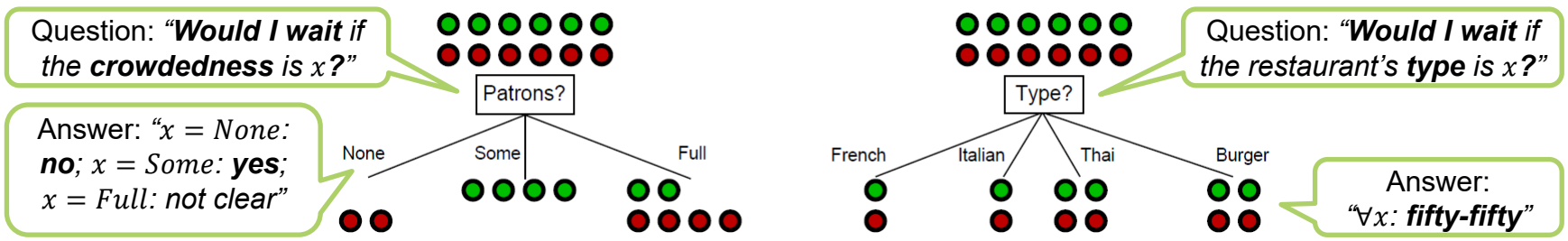
- **`PluralityValue(examples)` selects** the **most common output** among examples
- **`Importance(attribute, examples)`** selects the **most important attribute**
- On ties, both functions choose randomly

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

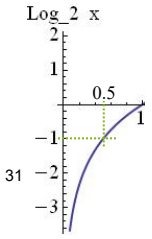
Recap: Information theory

- 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 $< P_1, \dots, P_n >$): $H(\langle P_1, \dots, P_n \rangle) = -\sum_{i=1}^{n} P_i \log_2 P_i$

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

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

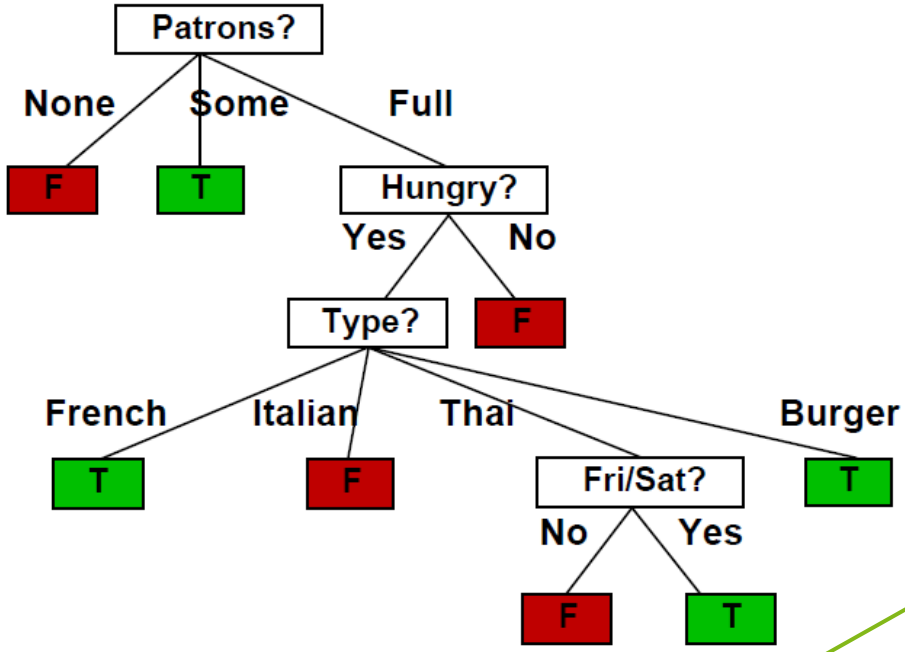
Entropy of branch i, weighted by branch's size

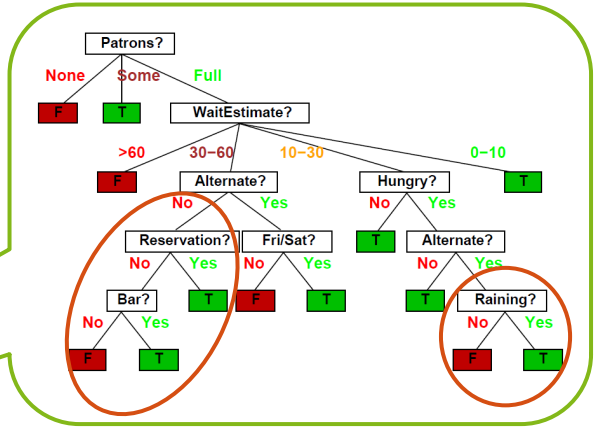
- For $Patrons$ this is $0.459\ bits$, for $Type$ this is (still) $1\ bit$
  ➔ **Choose** the attribute that **minimizes** the **remaining information need**ed, …
  ➔ i.e., maximizes information gain: $Gain(A) = H\left(\left\langle \frac{p}{p+n}, \frac{n}{p+n} \right\rangle\right) - Remainder(A)$

Entropy of original problem

Entropy remaining after splitting on A

The learned decision tree
Based on our 12 examples

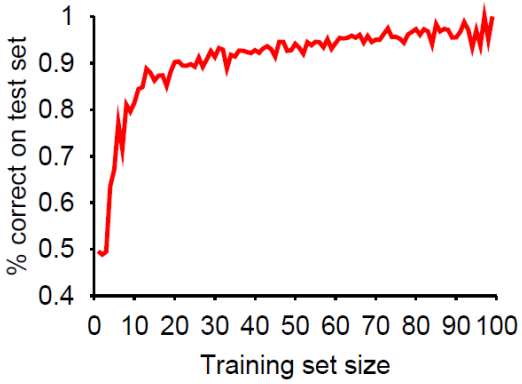


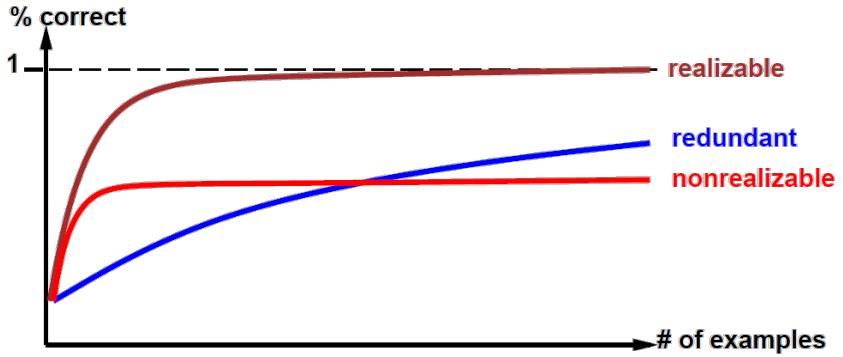


- Substantially simpler than *"true"* tree
  → E.g., $Reservation$ and $Raining$ are not needed (perfect classification possible without)
- A more complex hypothesis isn't justified by the small amount of data
  → But **what makes one tree better than another?**

Learning curves
Diagnosing learning problems

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





Accuracy shown in learning curve depends on

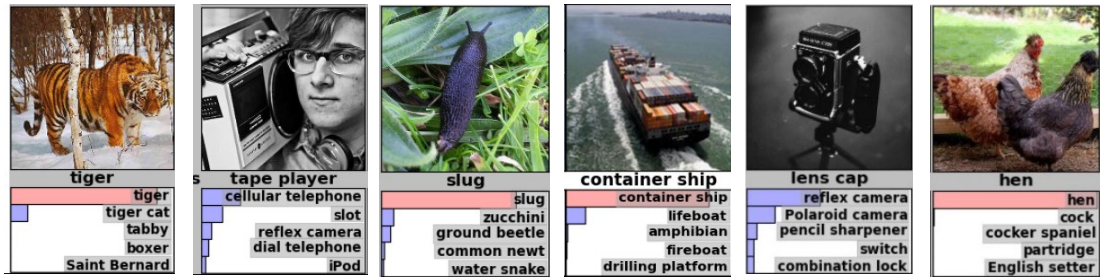
- Realizability (target function expressible in chosen hypothesis space?)
  - **Non**-realizability can be due to **missing attributes**
  - or **restricted hypothesis class** (e.g., a thresholded linear function might be overly simplistic)
- **Redundant** features
  (e.g., loads of irrelevant attributes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

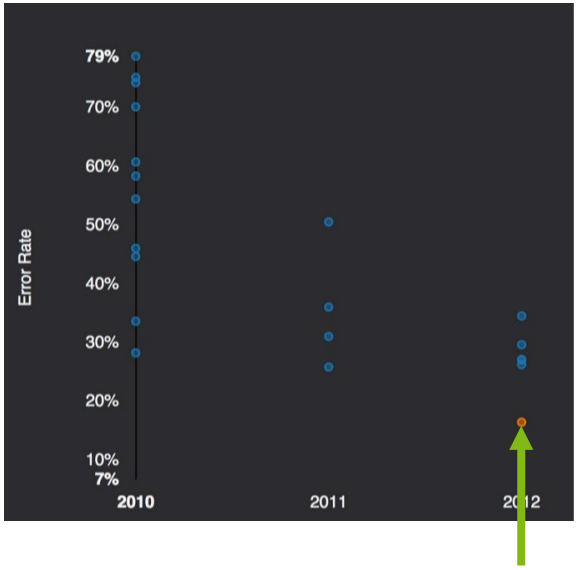
Zurich University
of Applied Sciences

DEEP LEARNING SUCCESS STORY

Why is this current hype about deep learning?
The ImageNet Competition

1000 categories
1 mio. training examples
…





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)

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

Zurich University
of Applied Sciences

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^ATT_EX, …)
- Word vectors: Arithmetic with semantic meaning of text and images

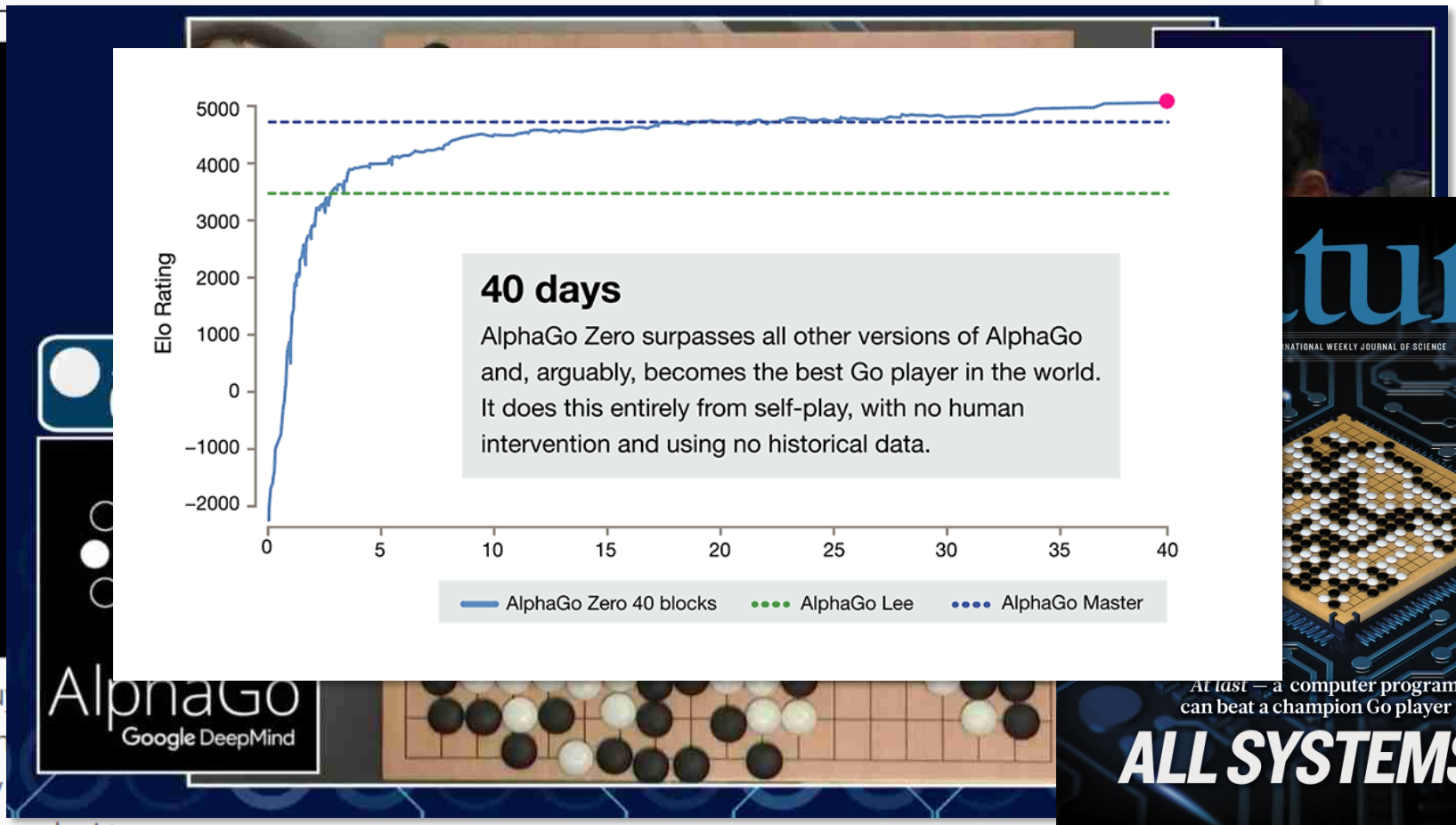
➔ See next slides

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

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

Zurich University of Applied Sciences

zh aw



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
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 c
generation of sound, making a library o
but often less effective.

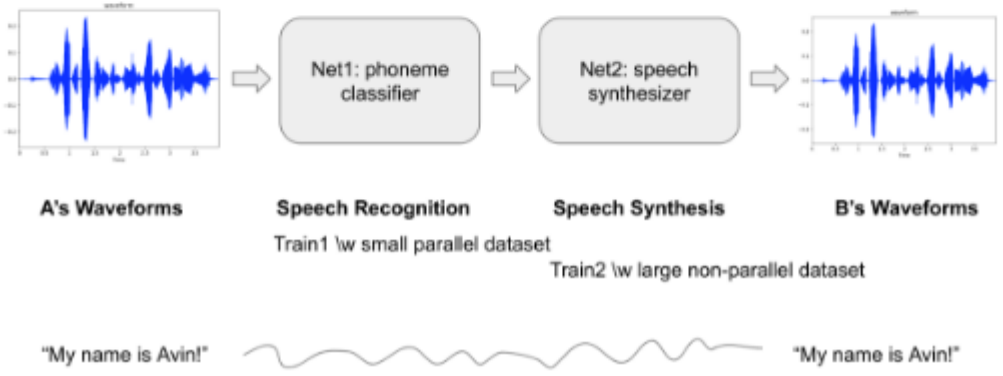
WaveNet, as the system is called, takes
low a level as possible: one sample at a
scratch — 16,000 samples per second.

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.



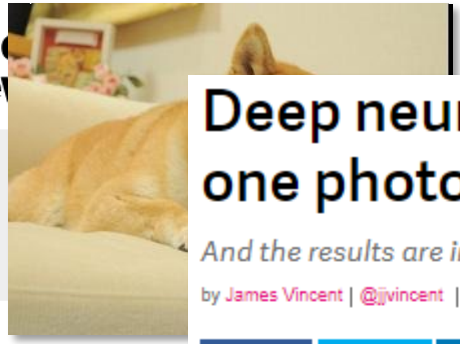
Zurich University of Applied Sciences

zh aw

nerated
ech from text

nerated music
of creativity

1 Second



MIT Techn Revie

Zurich University of Applied Sciences

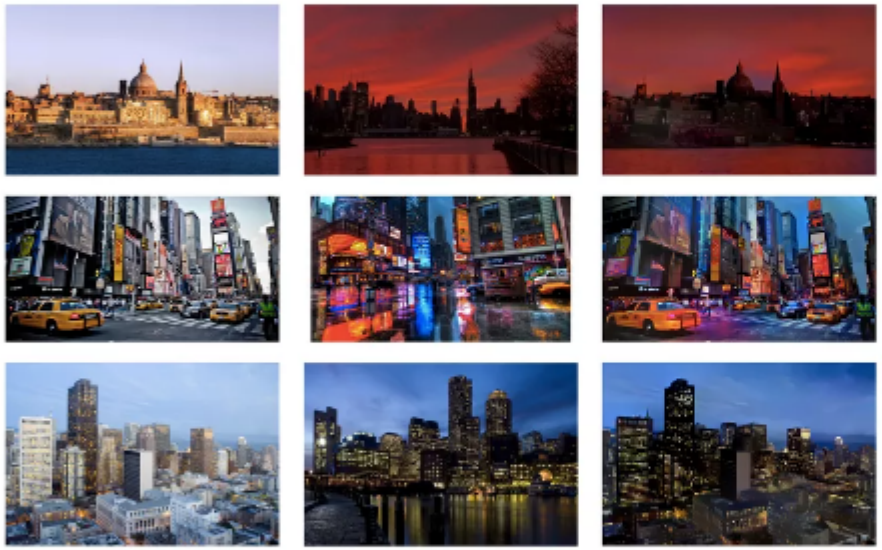
zh aw

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

Ad closed by Google

Report this ad

AdChoices

NOW TRENDING

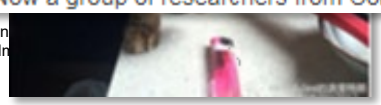
Computing

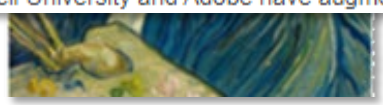
Algorith Artistic Other In

A deep neural n other images.

by Emerging Tech

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





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

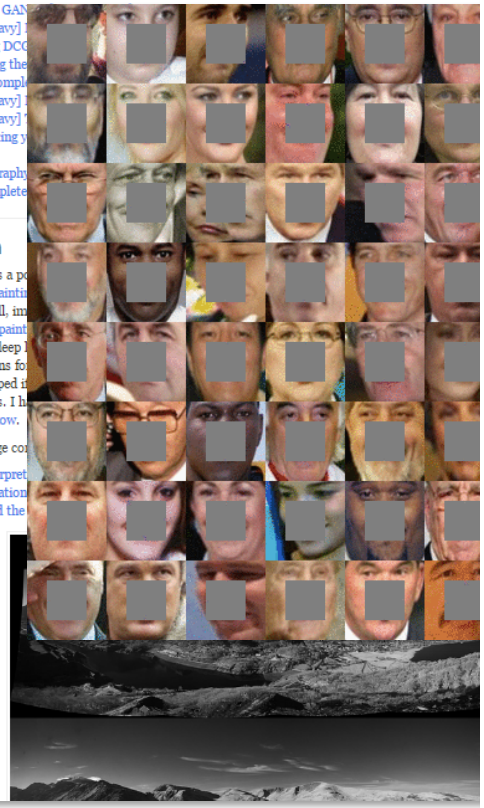
- Introduction
- Step 1: Interpreting images as samples from a probability distribution
  - How would you fill in the missing information?
  - But where does statistics fit in? These are images.
  - So how can we complete images?
- Step 2: Quickly generating fake images
  - Learning to generate new samples from an unknown probability distribution
  - [ML-Heavy] Generative Adversarial Net (GAN) building blocks
  - Using $G(z)$ to produce fake images
  - [ML-Heavy] Training DCGANs
  - Existing GAN
  - [ML-Heavy]
  - Running DC
- Step 3: Finding the
  - Image compl
  - [ML-Heavy]
  - [ML-Heavy]
  - Completing y
- Conclusion
- Partial bibliography
- Bonus: Incomplete

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

GEEK.COM

TECH

Nvidia AI Generates Fake Faces Based On Real Celebs

BY STEPHANIE MLOT 10.31.2017 :: 10:00AM EST

32 SHARES

I'm getting a distinctly mid-90s "The Rachel" vibe from the woman in the top left corner (via Nvidia)

STAY ON TARGET

AI Shelley Pens Truly Creepy Horror Stories–And You Can Help

Neural Network Serves Up Truly Frightening Halloween Costume Ideas

Celebrity scandals are about to get a lot more complicated.

Nvidia has **developed** a way of producing photo-quality, AI-generated human profiles—by using famous faces.

hand,

law,

ds,

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

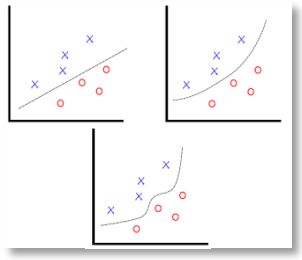
Inductive supervised learning

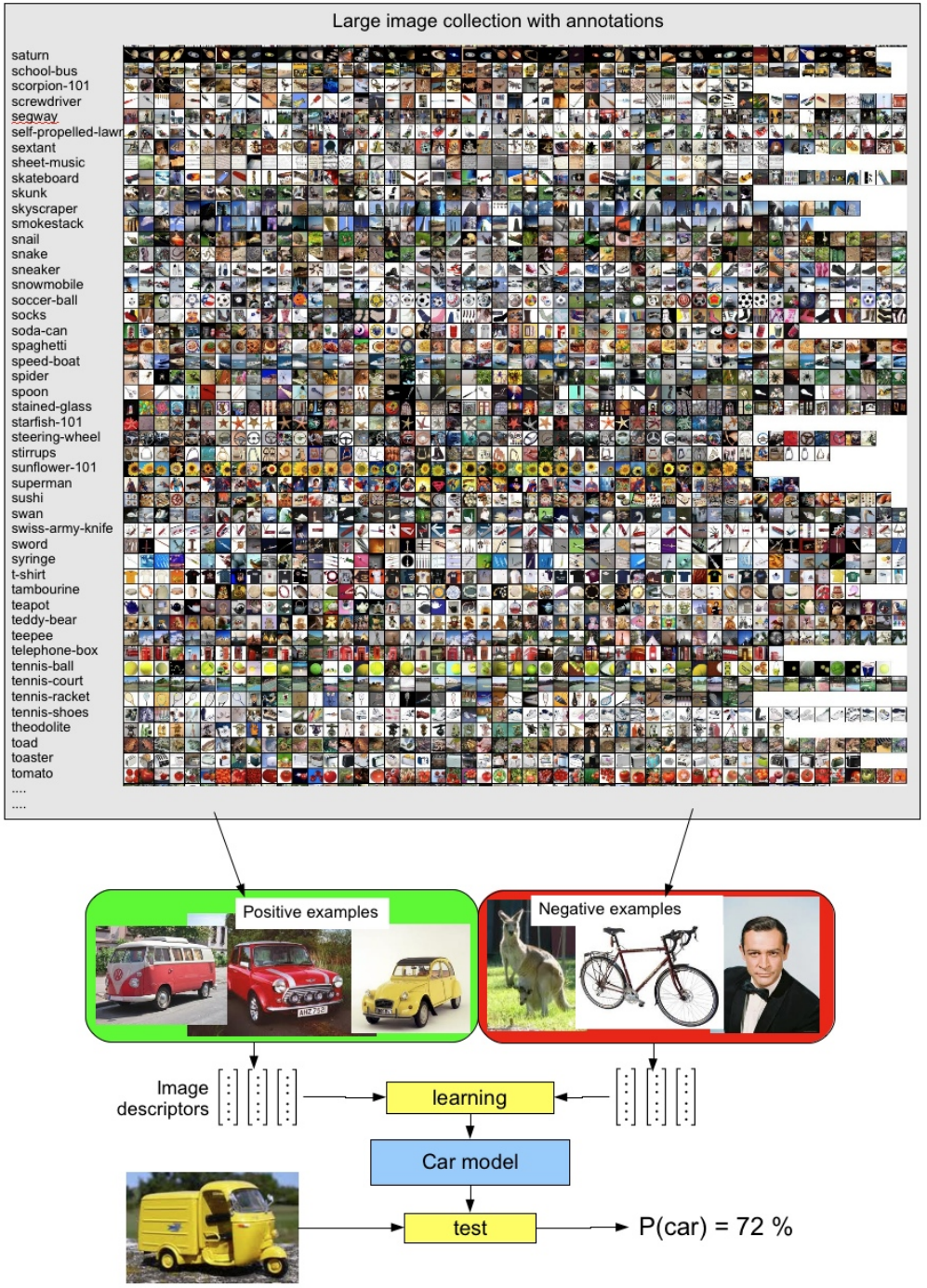
Assumption

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

Method

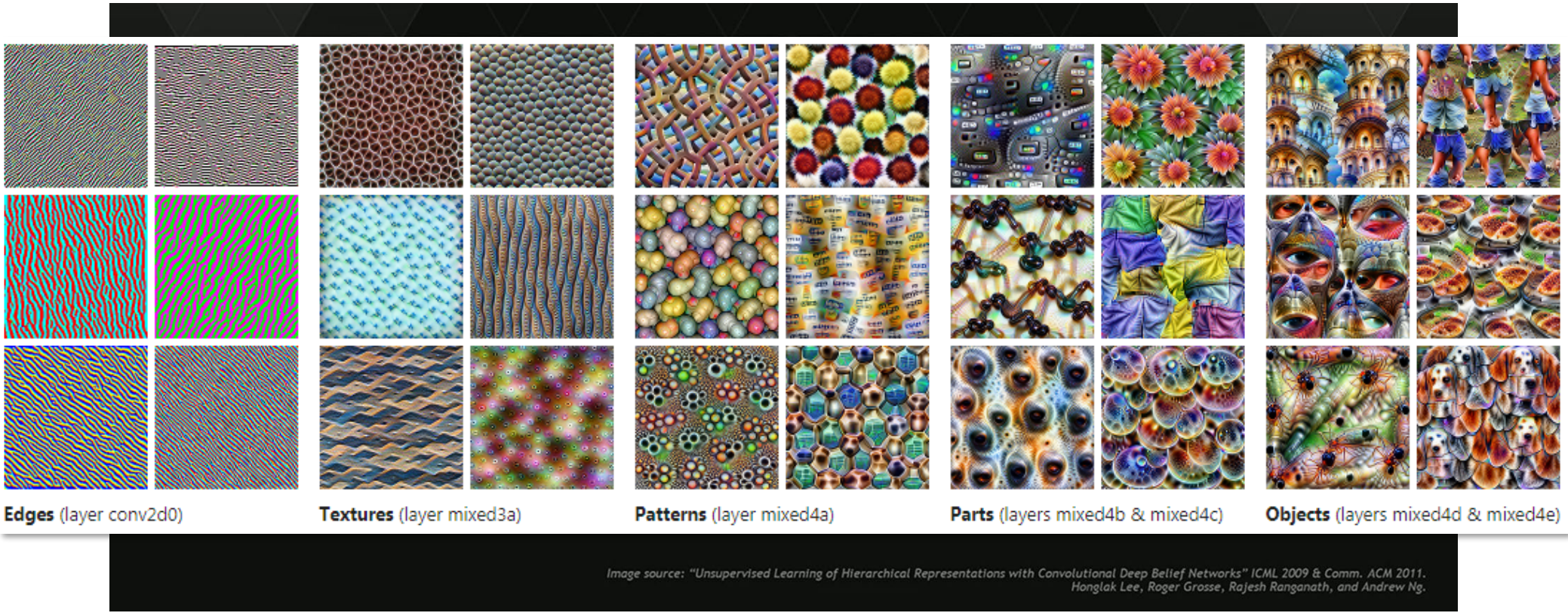
- **Search for parameters** of a given class of functions…
- …such that every training input (e.g. an image) is mapped to the correct output label (e.g. «`car`»)





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

What 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/.