ACM-BCB 2016 Tutorial

Seattle, WA, USA | October 2, 2016

Deep Learning for Bioinformatics and Health Informatics

SungRoh Yoon

Data Science Laboratory | Seoul National University

http://data.snu.ac.kr

About this tutorial

- ⋆ place, time, and date
  - Emerald II, 4–6pm on October 2, 2016 (Sunday)
- ⋆ objectives
  - to review fundamentals of machine learning
  - to provide a gentle introduction to deep learning (DL)
  - to survey recent applications of DL in biomedicine
- ⋆ anticipated difficulty level
  - easy (for data scientists) / moderate (for life scientists)
- ⋆ presenter: Prof. Sungroh Yoon, Seoul National University
  - PhD in EE (Stanford)
  - research areas: deep learning, data-driven AI, bioinformatics

Outline

Main references

⋆ books:

- *Deep Learning* by Goodfellow, Bengio and Courville ▸ Link
- *Learning from Data* by Abu-Mostofa et al. ▸ Link
- *Neural Networks and Learning Machines* (3rd edition) by Haykin

⋆ papers:

- *Learning Deep Architecture for AI* by Bengio ▸ Link
- *Supervised Sequence Labelling with RNNs* by Graves ▸ Link

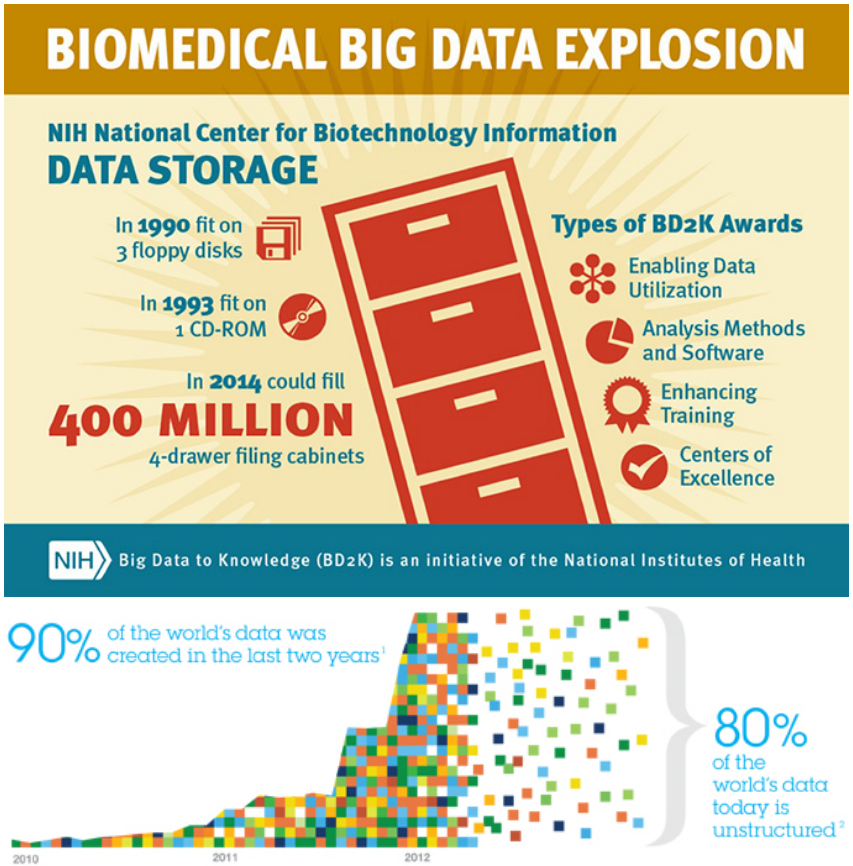
⋆ online resources:

- *Stanford CS224d: Deep Learning for Natural Language Processing* ▸ Link
- *Stanford CS231n: CNN for Visual Recognition* ▸ Link
- *Deep Learning Tutorial* at deeplearning.net ▸ Link
- *Deep Learning Tutorial* by LeCun ▸ Link
- *Online Course on Neural Nets* by Larochelle ▸ Link

Outline

Challenges

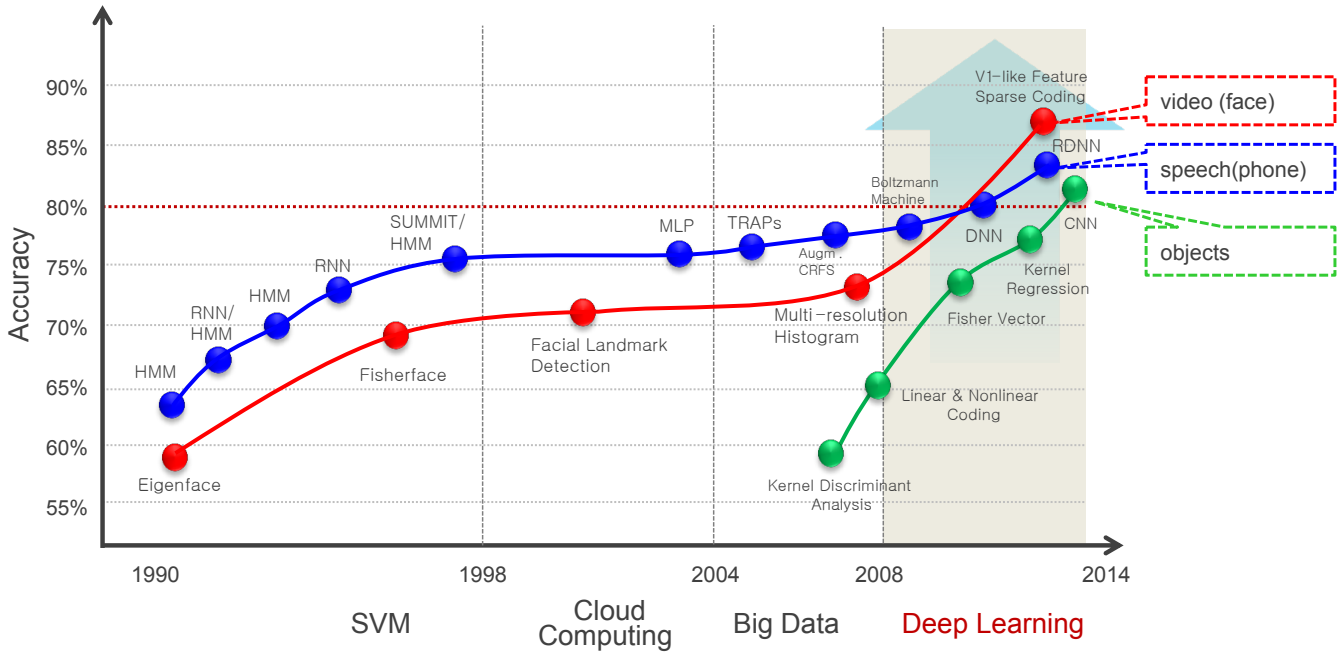
- ⋆ biomedical big data explosion
  - "..exceeds researchers' ability to capitalize.." (NIH)
- ⋆ needs for precise analysis
  - BD2K initiative ($656 mil)
  - PM initiative ($215 mil)



source: National Institutes of Health (NIH), IBM

News from other domains with big data

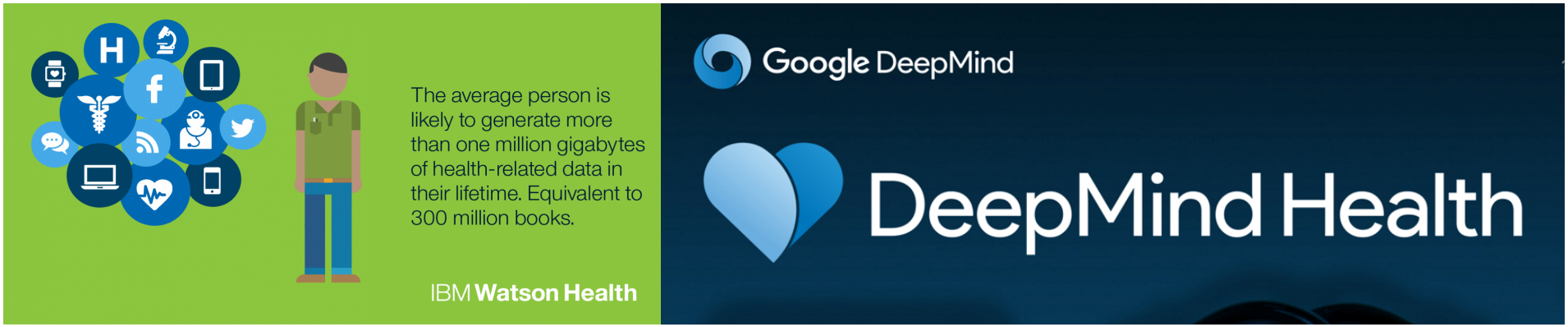
- ⋆ deep learning is achieving breakthrough performance
  - for tasks easy for human beings but difficult for computers

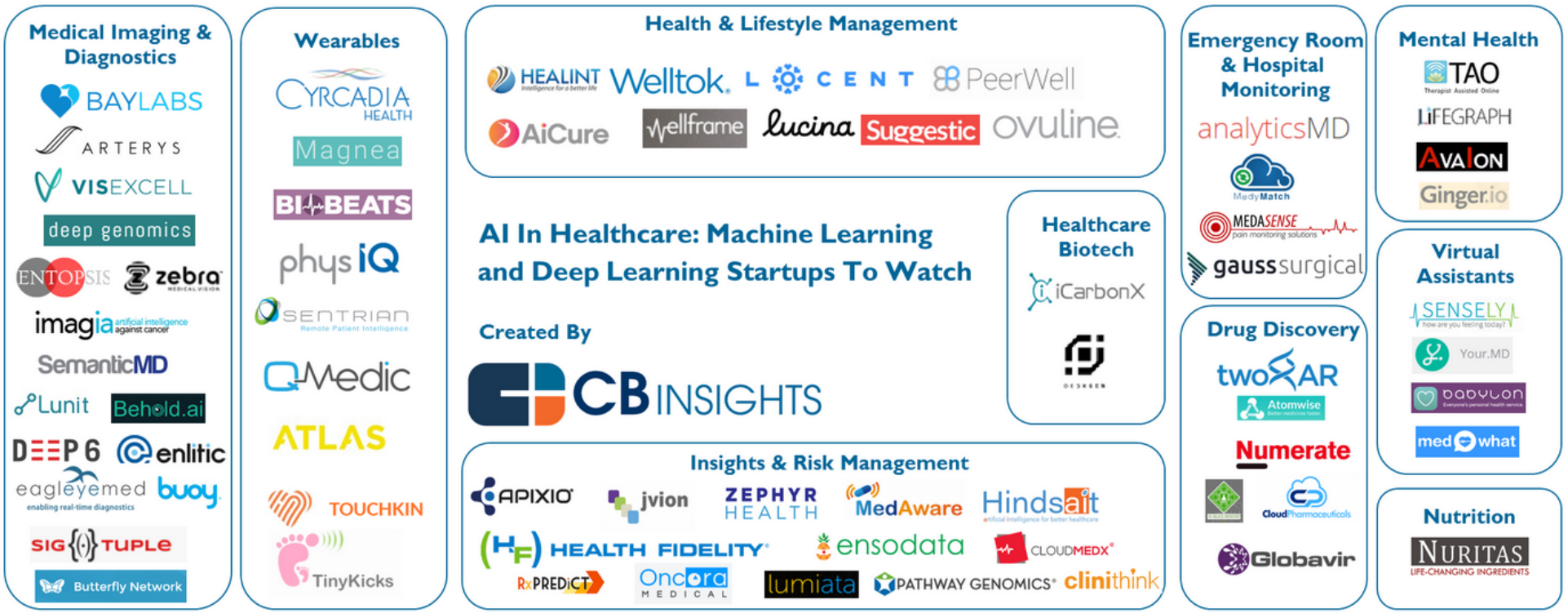


source: Samsung (2015)

- ⋆ examples: video/speech/object recognition
  - how about tasks in bio/health informatics?

AI/machine learning boom in biomedicine

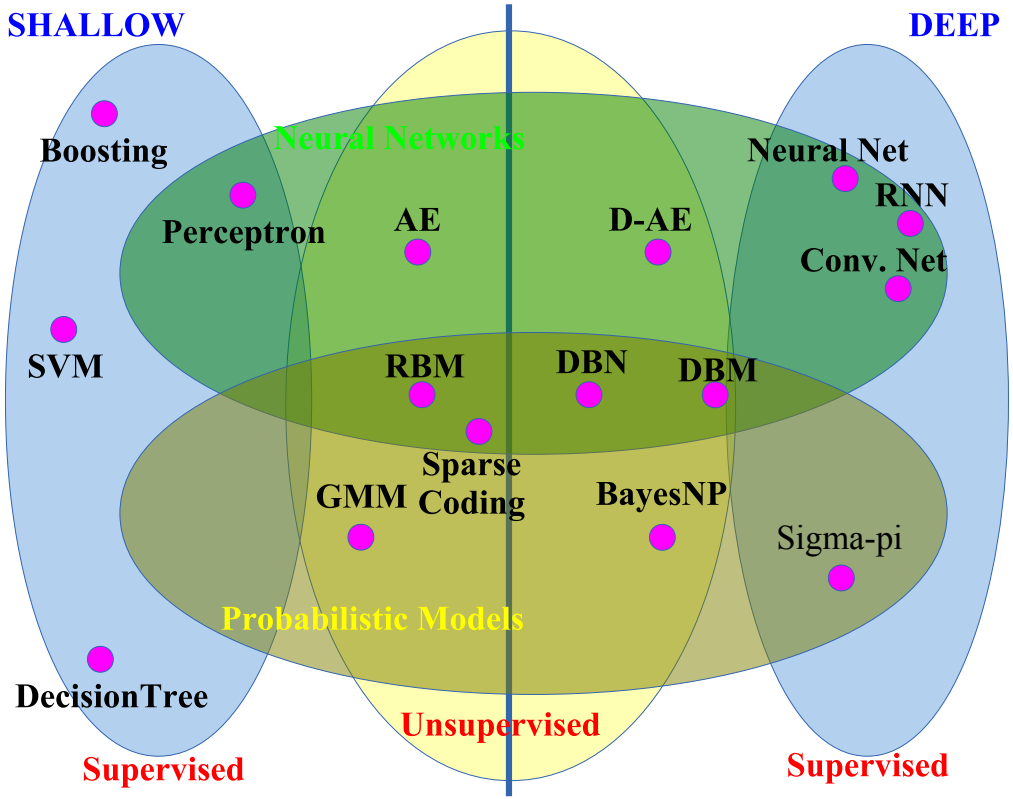




source: IBM, Google DeepMind, CBinsights

Machine learning

- ⋆ a subfield of artificial intelligence
  - "field of study that gives computers the ability to learn without being explicitly programmed" [A. Samuel, 1959]



source: LeCun & Ranzato (2013)

Why AI/ML approaches to biomedicine?

- ⋆ solving complex problems with huge amounts of data but little theory

  [Peng et al., 2010] diverse AI applications "from knowledge-based reasoning to learning and discovering novel biomedical knowledge"

- ⋆ generalization, reliability, and accuracy

  [Leung et al., 2016] ML can "infer models that are capable of generalizing to new contexts (improving the longevity and quality of life for millions of individuals, both now and in years to come")

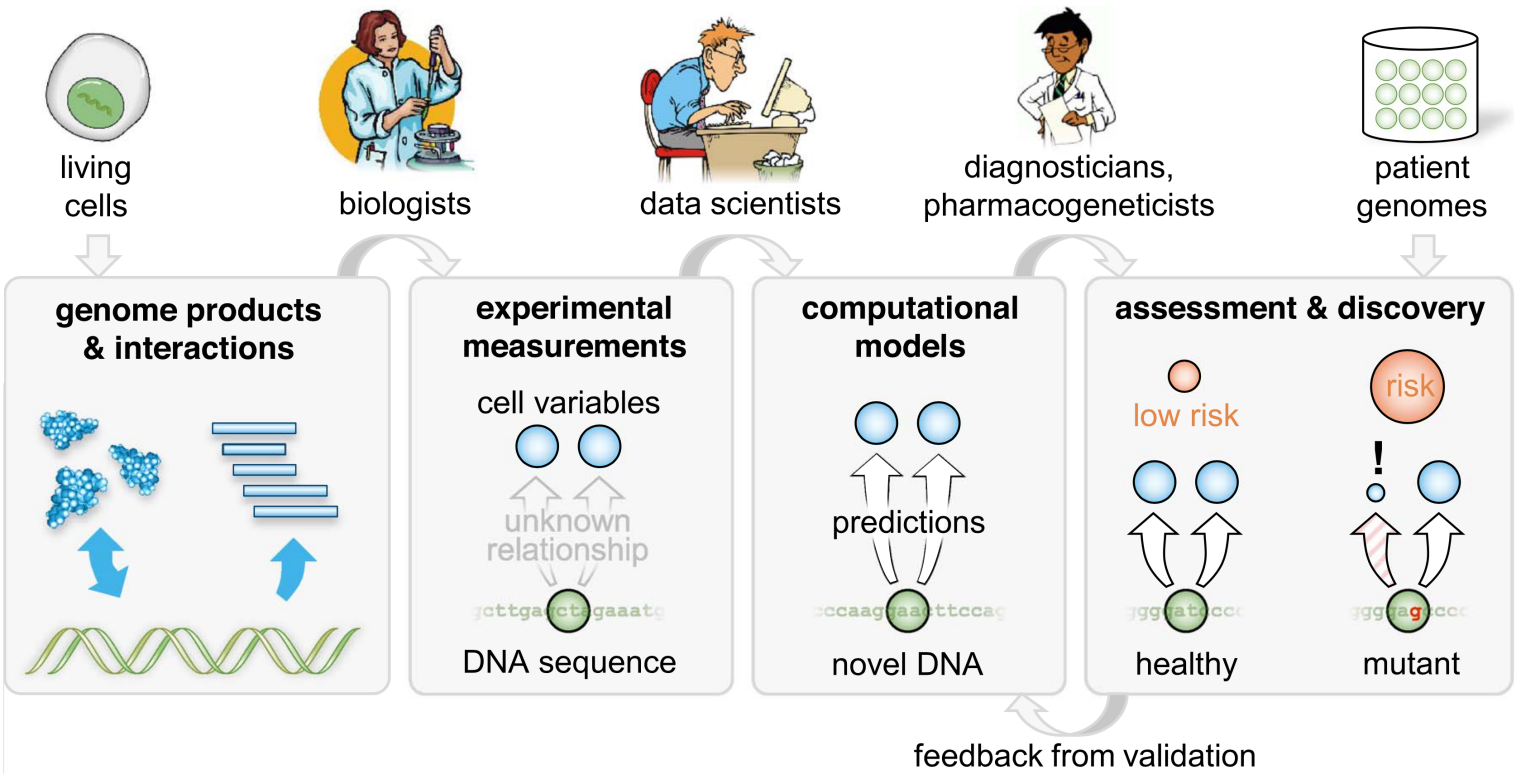
  [Coiera, 2003] AI approaches "actually have proven their reliability and accuracy on repeated occasions"

- ⋆ predictive modeling under uncertainty

  [Angermueller et al., 2016] ML can "drive predictive models without a need for strong assumptions about underlying mechanisms (which are frequently unknown or insufficiently defined)"

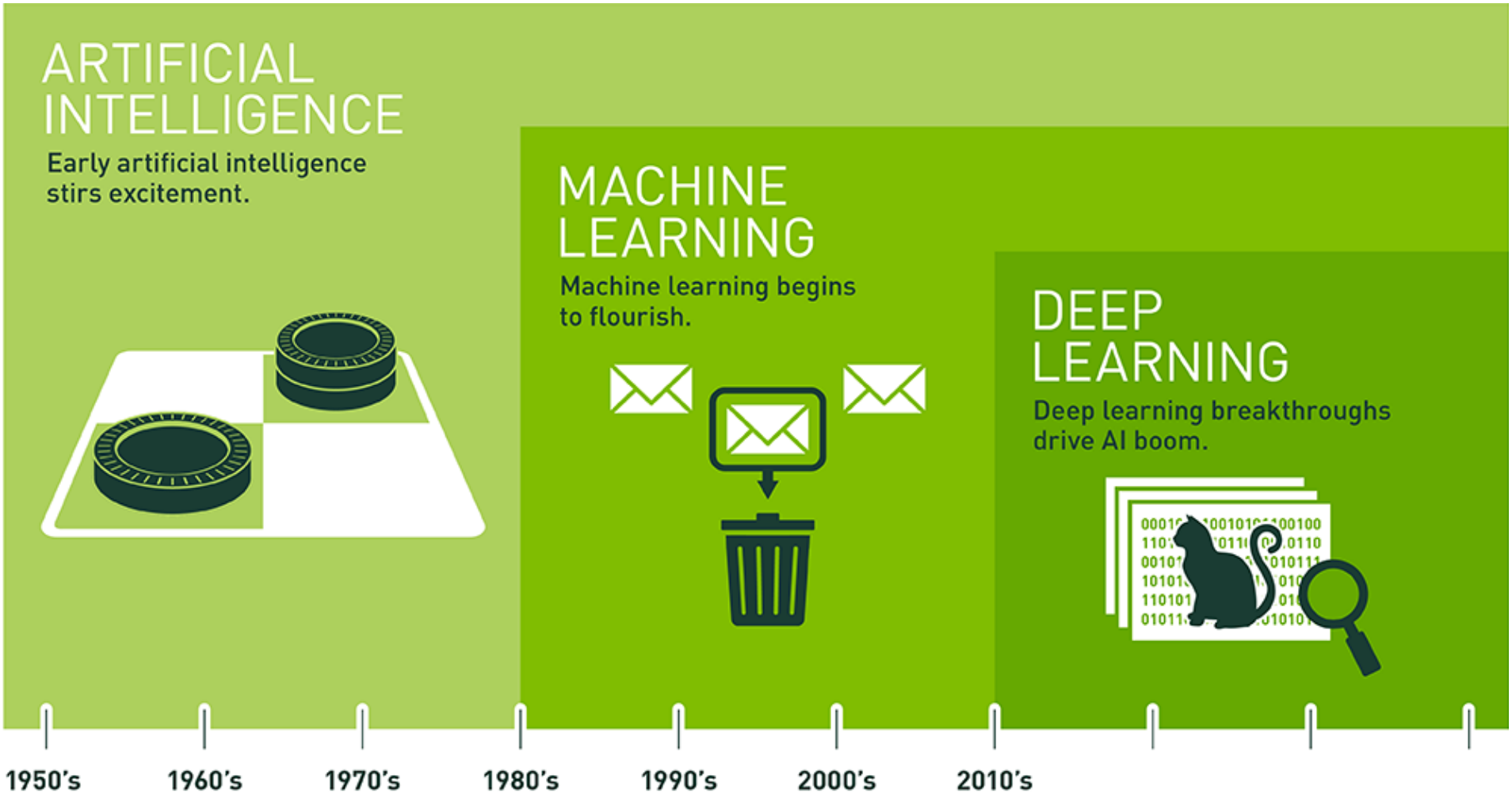
⋆ high-throughput data modeling

[Leung et al., 2016] "ML plays a central role by turning high-throughput measurements into specialized or general-purpose predictive models"



source: [Leung et al., 2016]

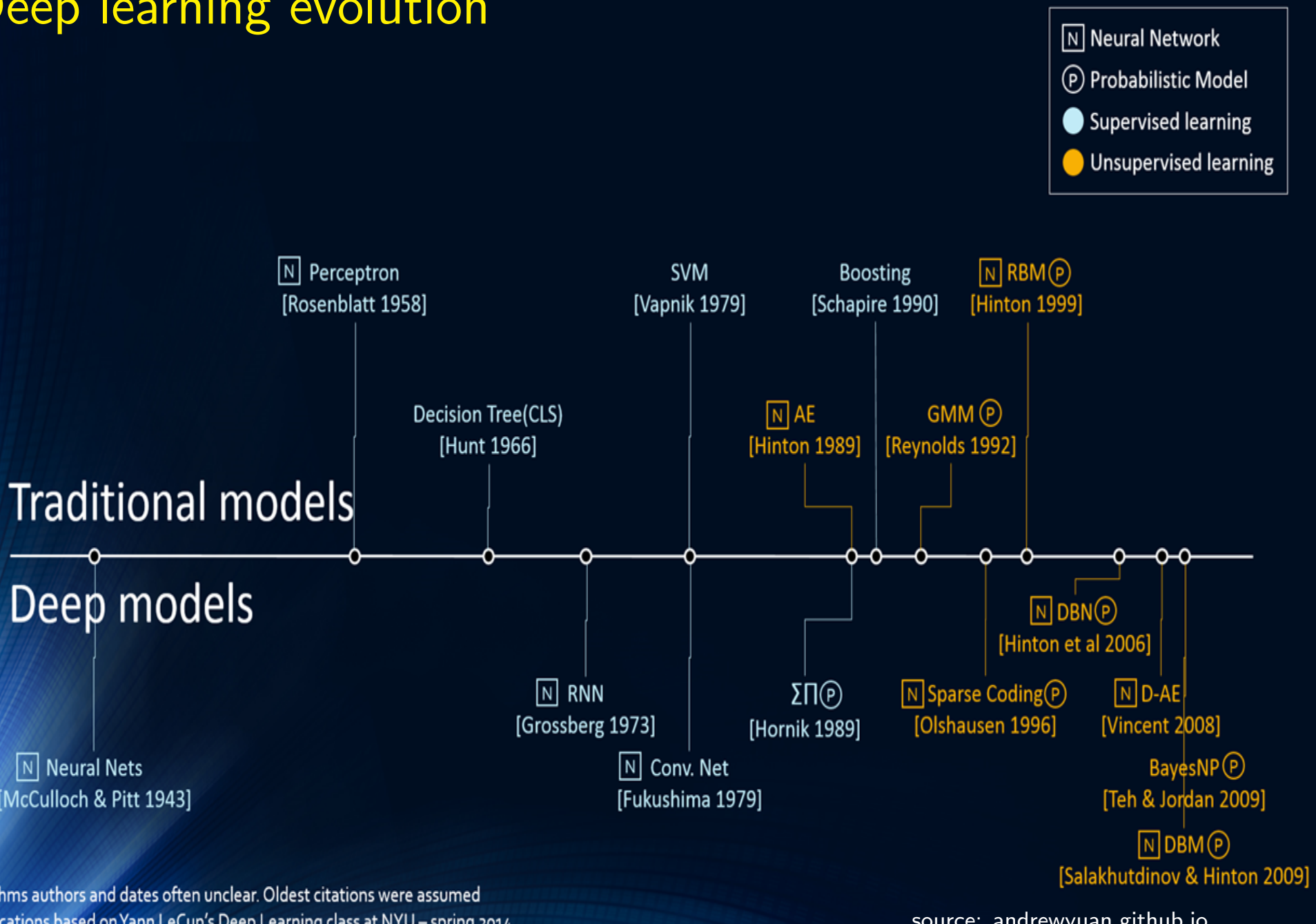
Comparison



Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

source: blogs.nvidia.com

Deep learning evolution

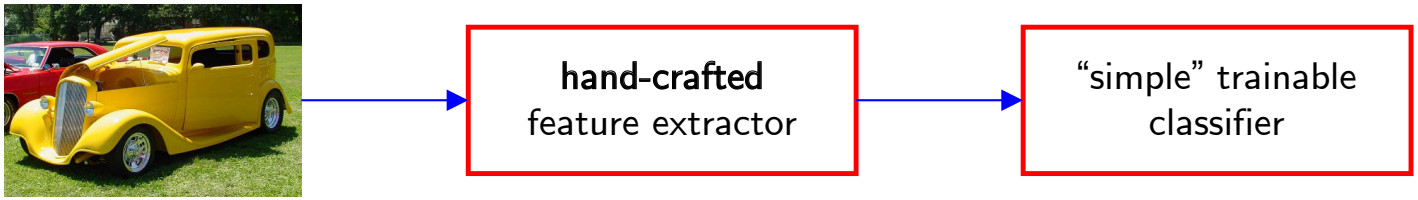


source: andrewyuan.github.io

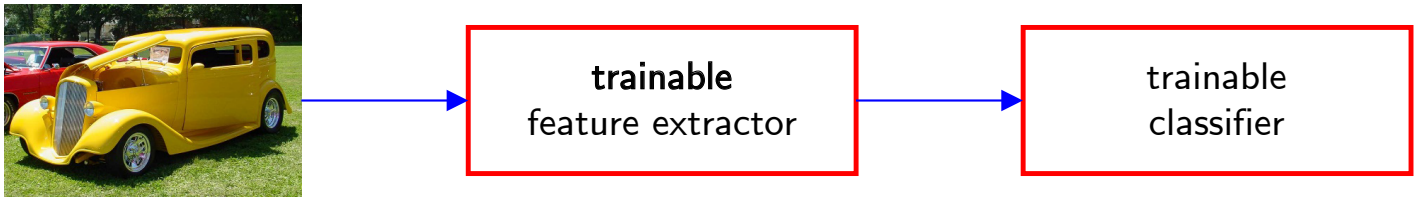
What is deep learning?

- ⋆ a "new" ML technique that has a long "history"
  - deep learning is driving the recent AI boom
- ⋆ Wikipedia
  - a set of algorithms in machine learning that attempt to model high-level abstractions in data by using model architectures composed of multiple nonlinear transforms
- ⋆ deeplearning.net
  - a new area of machine learning research, which has been introduced with the objective of moving machine learning closer to one of its original goals: artificial intelligence

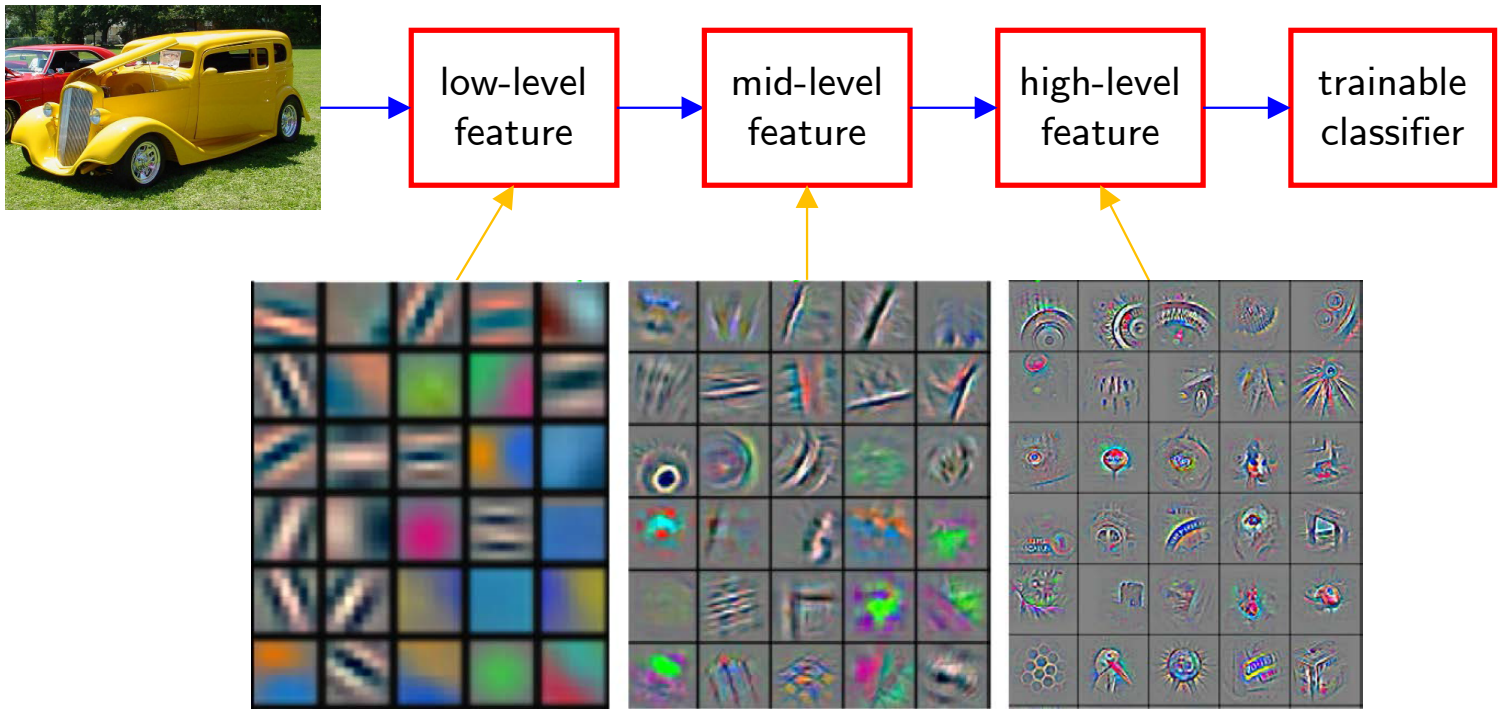
- ⋆ Y. LeCun
  - DL = learning representations/features
  - traditional model of pattern recognition (since the late 50's)
    - ▷ fixed/engineered features (or kernel) + trainable classifier



  - end-to-end learning/feature learning/deep learning
    - ▷ trainable features (or kernel) + trainable classifier

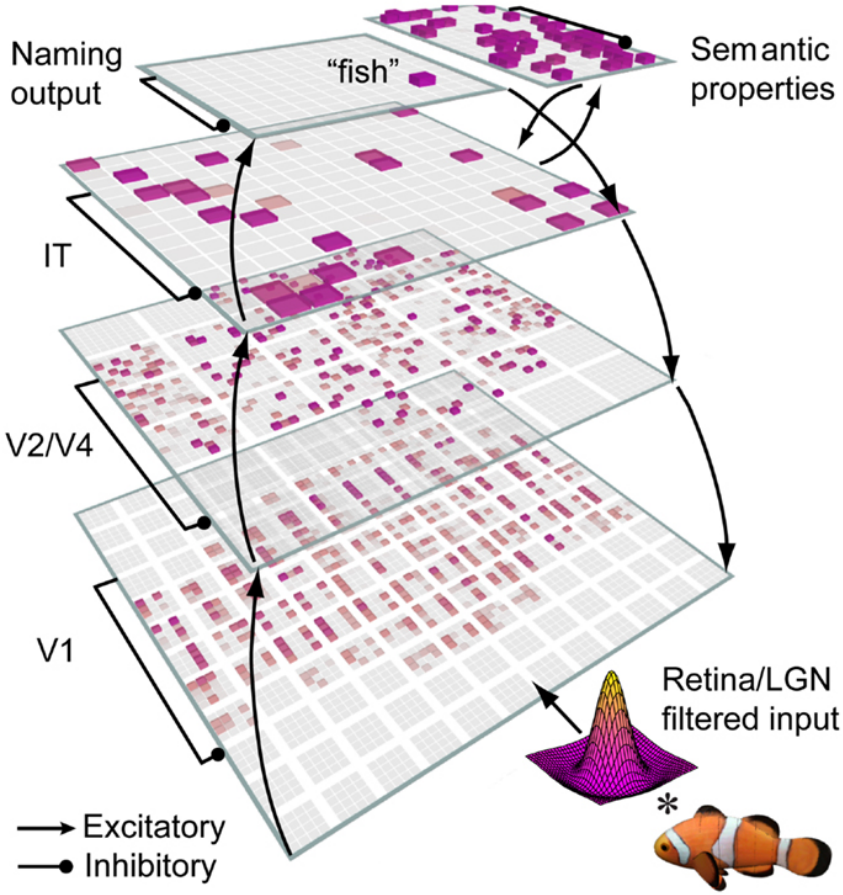


- ⋆ Y. LeCun
  - ▶ DL = learning hierarchical representations
    - ▷ deep if more than one stage of nonlinear feature transform



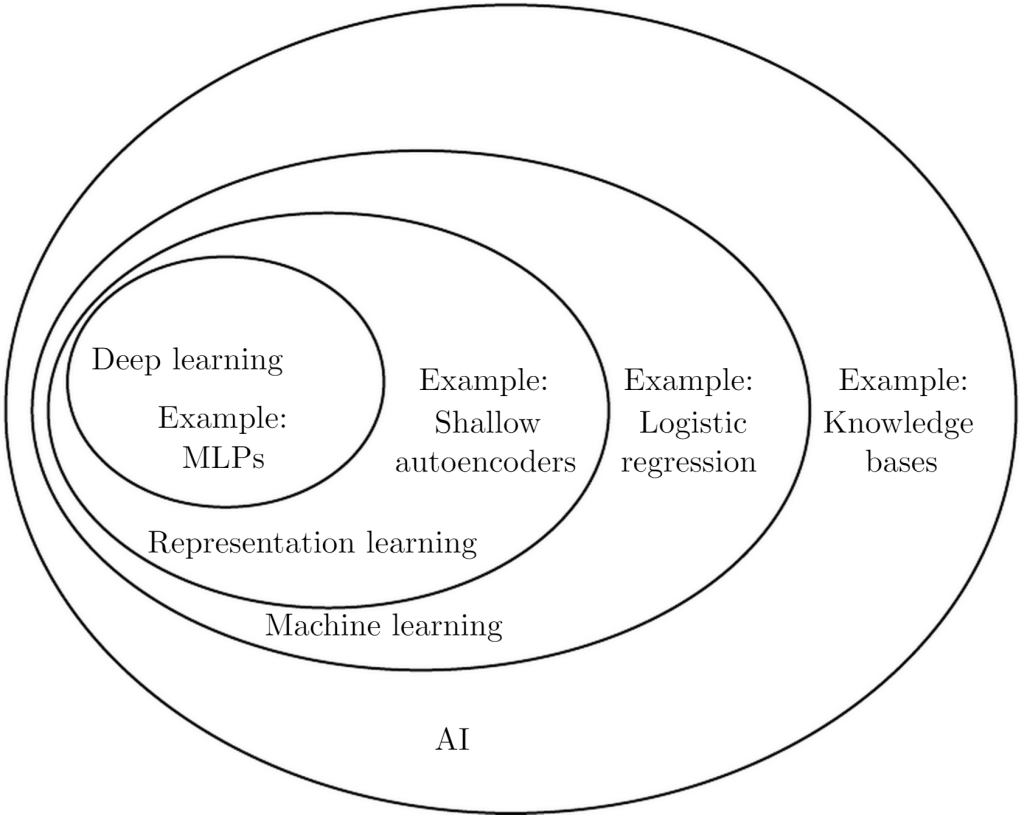
feature visualization of CNN trained on ImageNet (Zeiler and Fergus 2013)

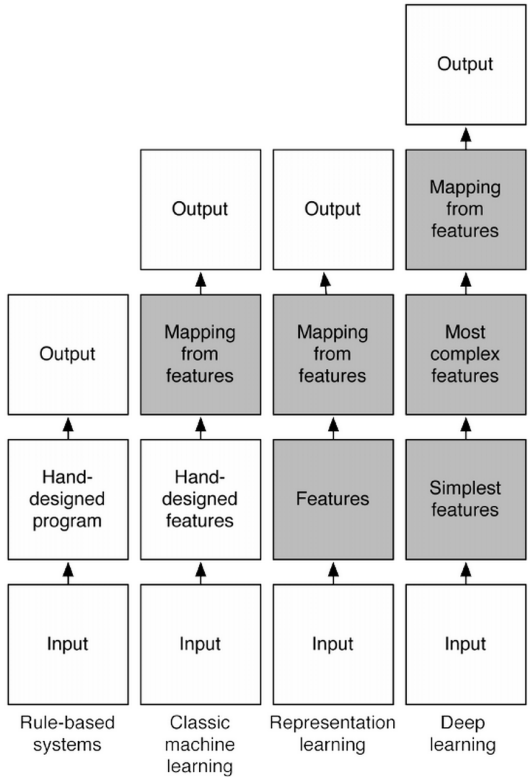
- ⋆ rationale: human visual system layers
  - higher levels receives input from lower levels



source: [O'Reilly et al., 2013]

- ⋆ [Goodfellow et al., 2016]
  - DL $\subseteq$ representation learning $\subseteq$ machine learning

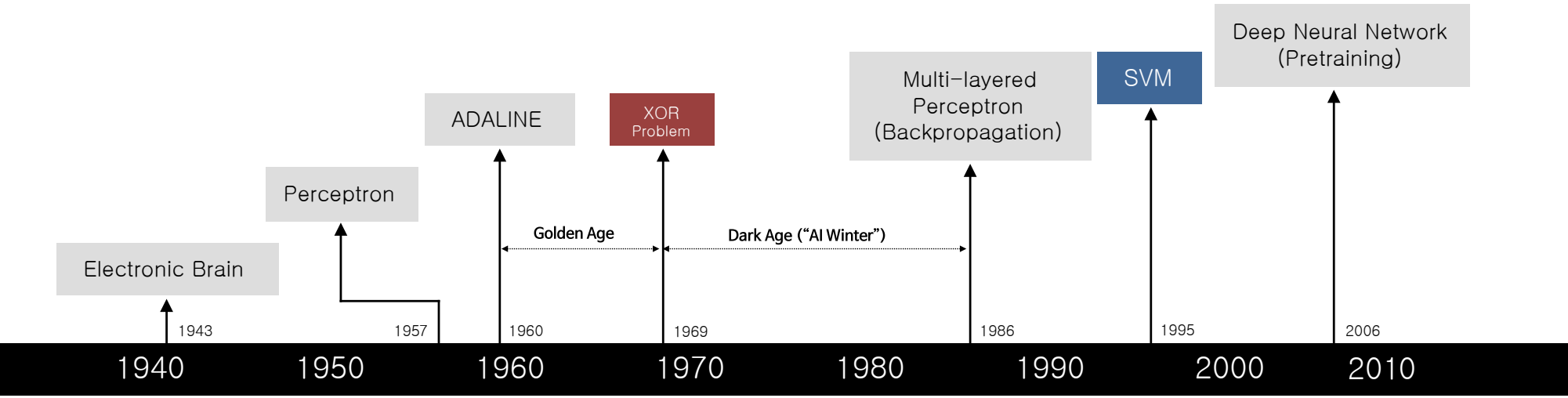


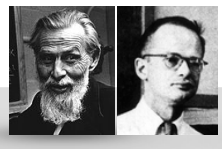


Historical trends in deep learning

- ⋆ key trends in deep learning [Goodfellow et al., 2016]
  - it has gone by many names
  - it has waxed and waned in popularity
  - it has become more useful with more training data
  - its models have grown size as infra (hw/sw) has improved
  - it has solved increasingly complicated applications with increasing accuracy over time

Related history





S. McCulloch - W. Pitts

F. Rosenblatt

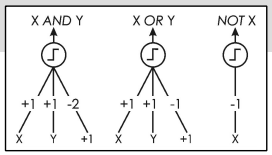
B. Widrow - M. Hoff

M. Minsky - S. Papert

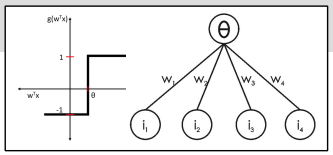
D. Rumelhart - G. Hinton - R. Wiliams

V. Vapnik - C. Cortes

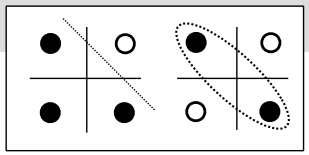
G. Hinton - S. Ruslan



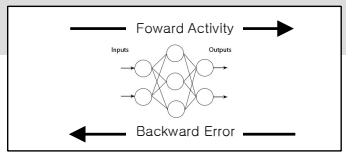
- Adjustable Weights
- Weights are not Learned



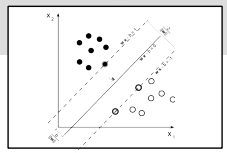
- Learnable Weights and Threshold



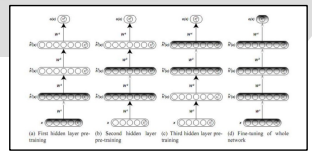
- XOR Problem



- Solution to nonlinearly separable problems
- Big computation, local optima and overfitting



- Limitations of learning prior knowledge
- Kernel function: Human Intervention



- Hierarchical feature Learning

source: vuno (2015)

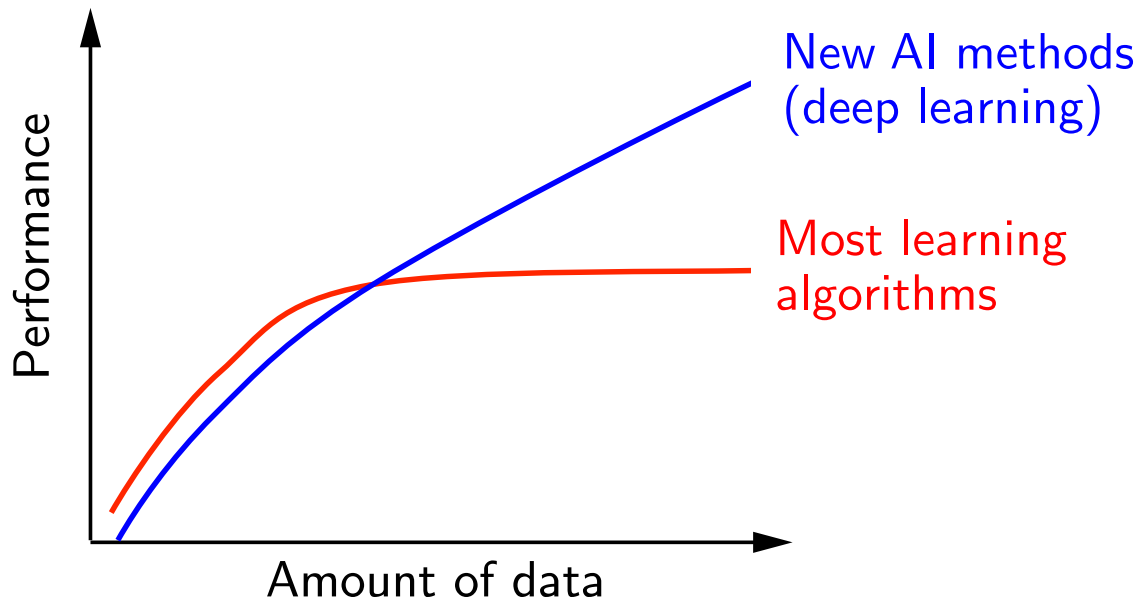
Three waves of DL development [Goodfellow et al., 2016]

1. "cybernetics": 1940's–1960's [neuroscience]
   - perceptron, δ-rule, stochastic gradient decent
   - essentially linear models $\Rightarrow$ limitations (*e.g.*, XOR)
2. "connectionism": 1980's–1990's [cognitive science]
   - simple units achieve intelligence when networked together
   - neural nets, distributed representation[1], back propagation
   - training issues $\Rightarrow$ eclipsed by kernel/graphical models
3. "deep learning": 2006–present [multidisciplinary]
   - breakthrough: greedy layer-wise pre-training (Hinton)
   - unsupervised $\rightarrow$ supervised/reinforcement $\rightarrow$ unsupervised ...

[1] each input to a system should be represented by many features, and each feature should be involved in the representation of many possible inputs

Deep learning is in its prime time

- ⋆ three key driving forces
  1. big data
  2. power of parallel/distributed computing
  3. sophisticated algorithms



source: Andrew Ng

How much data?

a rough rule of thumb as of 2016 [Goodfellow et al., 2016]:

- ⋆ 5000 labeled examples per category
  - to achieve acceptable performance by supervised deep learning
- ⋆ at least 10 million labeled examples
  - to match or exceed human performance
- ⋆ active research areas
  - pre-training and/or transfer learning
  - un/semi-supervised learning to use unlabeled data

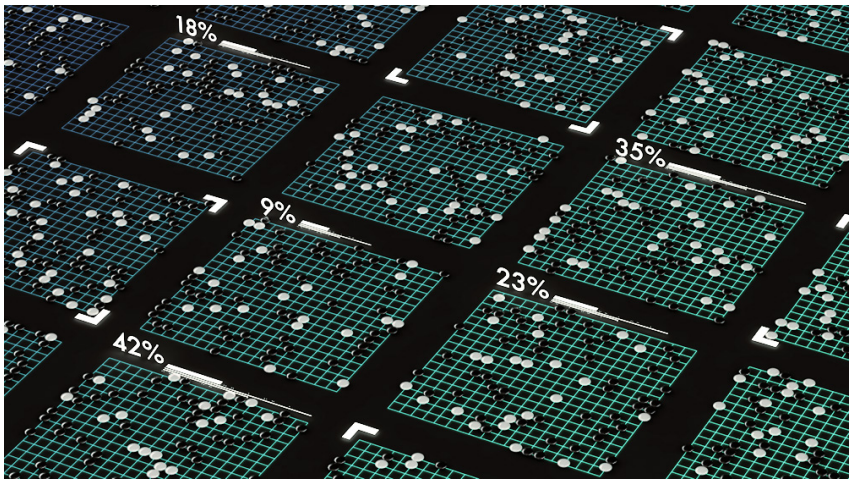
GPU-ACCELERATED DEEP LEARNING FRAMEWORKS

	CAFFE	TORCH	THEANO	CUDA-CONVNET2	KALDI
Domain	Deep Learning Framework	Scientific Computing Framework	Math Expression Compiler	Deep Learning Application	Speech Recognition Toolkit
cuDNN	R2	R2	R2	--	--
Multi-GPU	In Progress	Partial	Partial	✓	✓ (nnet2)
Multi-CPU	✗	✗	✗	✗	✓ (nnet2)
License	BSD-2	GPL	BSD	Apache 2.0	Apache 2.0
Interface(s)	Text-based definition files, Python, MATLAB	Python, Lua, MATLAB	Python	C++	C++, Shell scripts
Embedded (TK1)	✓	✓	✗	✗	✗

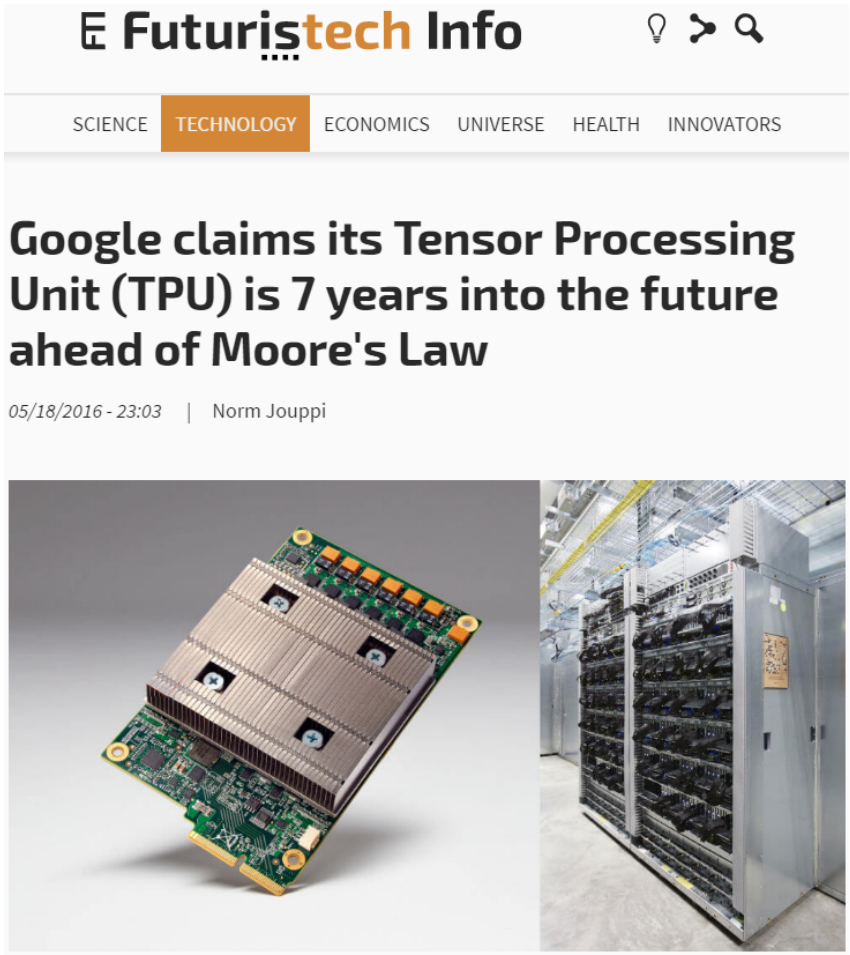
http://developer.nvidia.com/deeplearning

Human vs machine

- ⋆ AlphaGo (DeepMind)

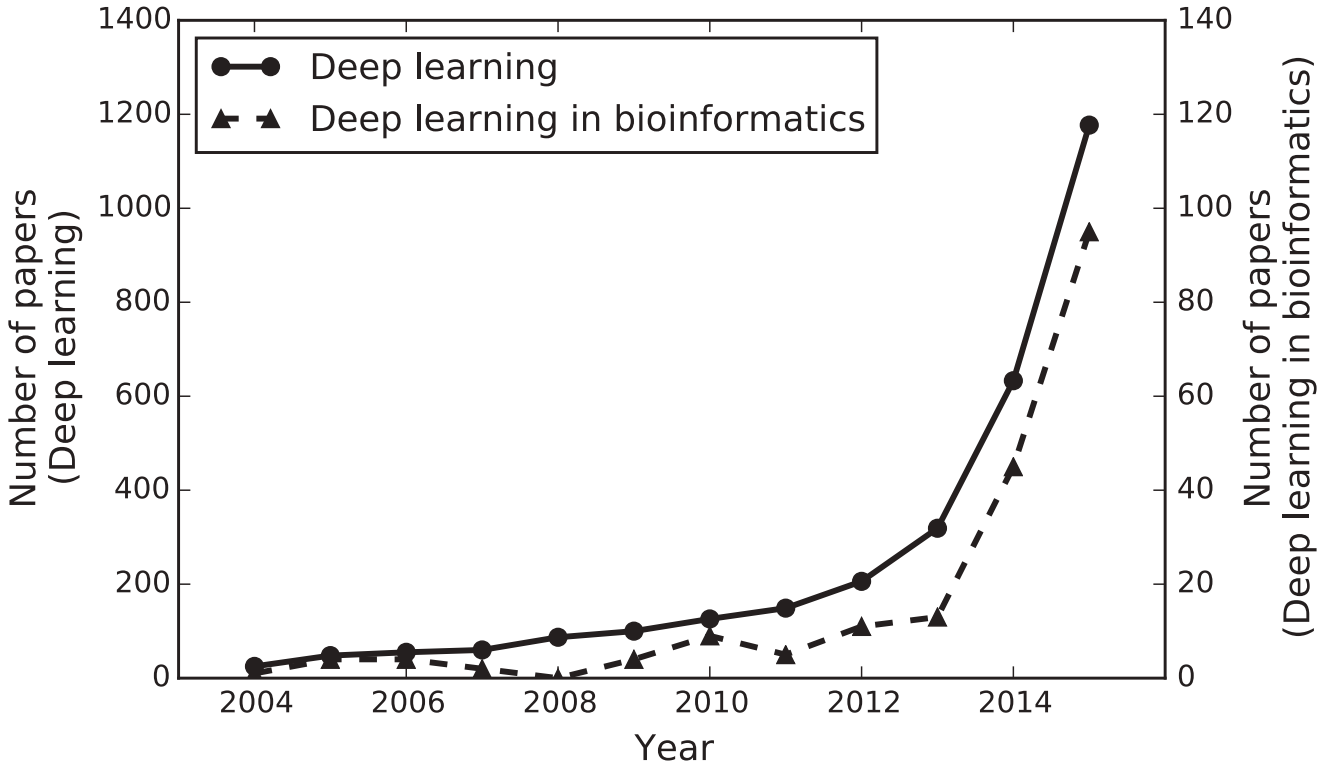


- ⋆ TPU (Google)



Why deep learning in bioinformatics?

- ⋆ many people do it ☺
  - "`Reviewer 1:  ..you should consider using DL..`"



source: [Min et al., 2016]

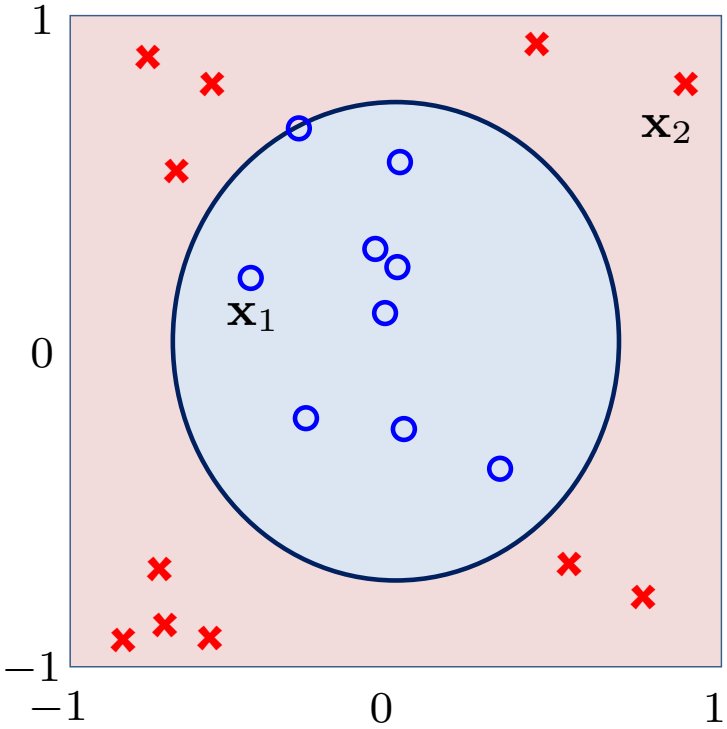
- ⋆ a **better** toolbox
  - better classification & regression by DL
  - DL replaces the "back-end" in conventional ML pipeline
- ⋆ a **new** toolbox
  - the representation learning capability of DL

  $\Rightarrow$ can relieve the burden of laborious feature engineering
  - DL complements the "front-end"
- ⋆ a natural **integrative** framework
  - biomedical big data: heterogeneous, multi-modal
  - DL provides multi-modal/transfer learning capabilities

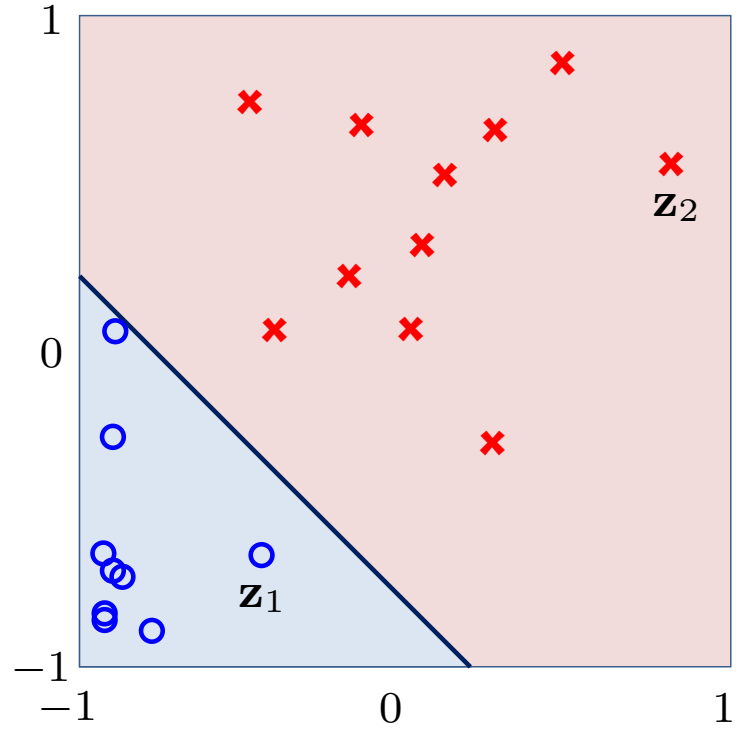
Importance of feature representation

⋆ original (not linearly separable) vs transformed data (separable)

source: [Abu-Mostafa et al., 2012]



(a) data in $\mathcal{X}$-space



(b) transformed data in $\mathcal{Z}$-space

$$\mathbf{x} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix}$$

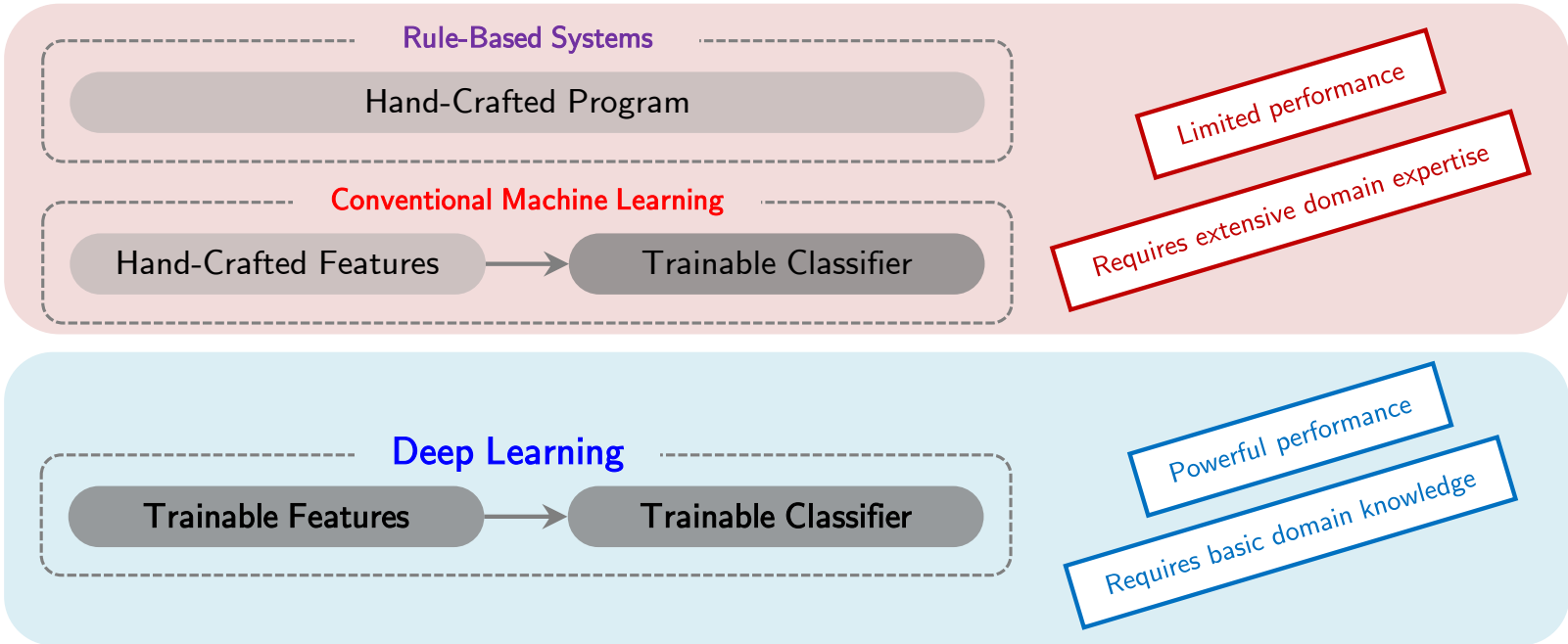
$$\mathbf{z} = \Phi(\mathbf{x}) = \begin{bmatrix} 1 \\ x_1^2 \\ x_2^2 \end{bmatrix}$$

Perspectives on deep learning in biomedicine

- ⋆ state-of-the-art performance [LeCun et al., 2015a]
  - deep learning has beaten other machine learning techniques in various biomedical applications
- ⋆ great promise for bio/health informatics [Park and Kellis, 2015]
  - capture multiple levels of information/abstraction within cells
  - summarize into lower-dimensional representations
- ⋆ the potential of DL is clear in bio [Angermueller et al., 2016]
  - better exploit the availability of large/high-dimensional data
  - discover high-level features
  - increase performance over traditional models
  - enhance interpretability and understanding of systems

Summary

- ⋆ evergrowing challenges in biomedicine
  - biomedical big data explosion + needs for precise analysis
- ⋆ recent solutions: AI & machine learning, especially deep learning
  - deep learning: data-driven representation learning
  - great promises in omics/imaging/signal/EHR and more



Outline

The learning problem

- ⋆ machine learning is useful when [Abu-Mostafa et al., 2012]
  - we have data, and a pattern exists
  - but we can hardly pin it down mathematically
- ⋆ learning model: hypothesis set + learning algorithm
  - ex) logistic regression, SVM, HMM, DNN
- ⋆ types of machine learning

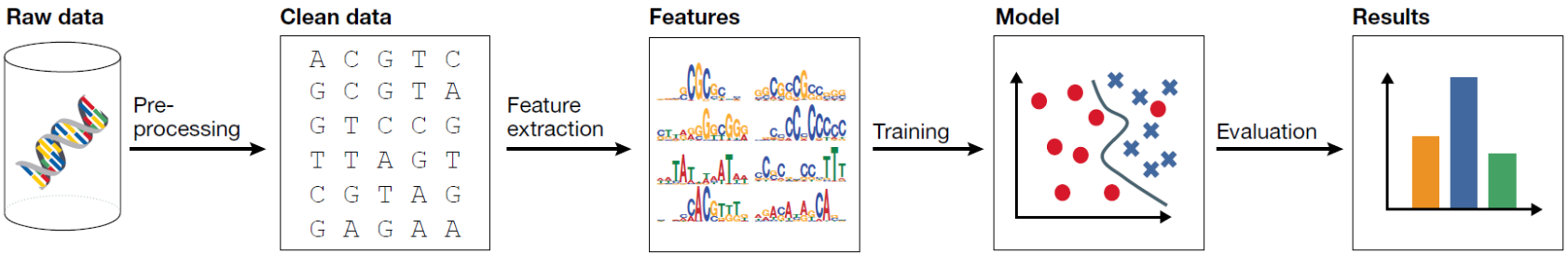
  supervised: (input, correct output/label)

  unsupervised: (input, no label)

  reinforcement: (input, some output, grade for this output)
- ⋆ supervised learning: the most studied type
  - unknown target function $y = f(\mathbf{x})$
  - known training data set $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$
  - learning algorithm picks $h \approx f$ from hypothesis set $\mathcal{H}$

Classical machine learning workflow

- ⋆ multiple key steps

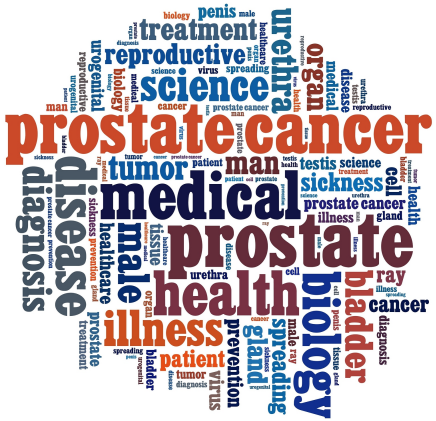


source: [Angermueller et al., 2016]

- preprocessing
- feature engineering
- model selection & learning
- evaluation

Running example: prostate cancer diagnosis

- ⋆ prostate cancer
  - ▶ more than 200,000 US cases per year
- ⋆ goal: data-driven diagnosis
  - ▶ patient information $\longrightarrow$
- ⋆ prostate cancer or not?



feature	value
age	67
gender	male
digital rectal exam	positive
transitional volume	20 ml
PSA level	720 ng/ml
prostate volume	100 ml
...	...

Components of learning: formalization

- ⋆ let $\mathcal{X} = \mathbb{R}^d$ be the input space
  - $\mathbb{R}^d$: the d-dimensional Euclidean space
  - input vector $\mathbf{x} \in \mathcal{X}$: $\mathbf{x} = (x_1, x_2, \ldots, x_d)$
- ⋆ let $\mathcal{Y} = \{+1, -1\}$ be the output space
  - denotes a binary (yes/no) decision
- ⋆ in our diagnosis example
  - coordinates of input $\mathbf{x}$: age, PSA level, prostate volume, ...
  - binary output y: cancer positive or negative

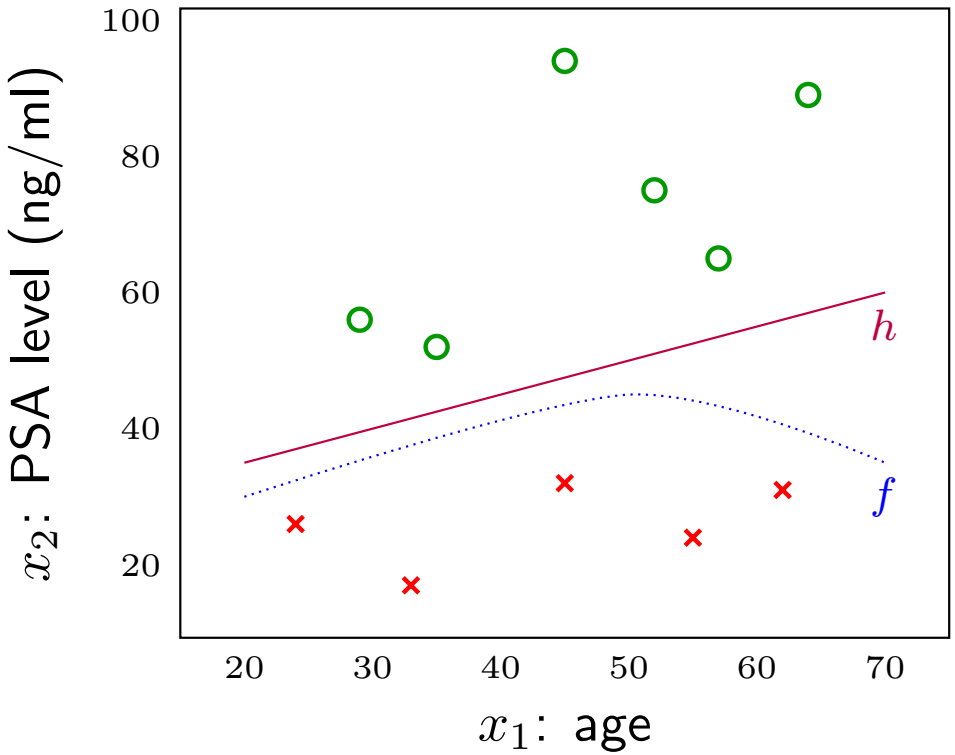
component	symbol	cancer diagnosis metaphor
input	$\mathbf{x}$	patient information
output	y	positive or negative
target function	$f : \mathcal{X} \to \mathcal{Y}$	ideal diagnosis formula
data	$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$	historical records
hypothesis	$h : \mathcal{X} \to \mathcal{Y}$	formula to be used

- f: unknown target function
- $\mathcal{X}$: input space (set of all possible inputs $\mathbf{x}$)
- $\mathcal{Y}$: output space (set of all possible outputs)
- N: the number of input-output examples (*i.e.*, training examples)
- $\mathcal{D} \triangleq \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$: data set where $y_n = f(\mathbf{x}_n)$
- $\mathcal{H}$: a set of hypotheses from which h is taken ($h \in \mathcal{H}$)
- $\mathrm{e}(\hat{y}, y)$: point-wise error ($\hat{y} = h(\mathbf{x})$)
- $\mathrm{E}_{\mathrm{in}}(h)$: in-sample (training) error with hypothesis h used

Example

- $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ where x_1: age and x_2: PSA level in ng/ml
- $N = 11$, $d = 2$, $\mathcal{X} = \mathbb{R}^2$, and $\mathcal{Y} = \{\text{positive}, \text{negative}\}$
- data set $\mathcal{D}$:

n	x_1	x_2	y
1	29	56	positive
2	64	89	positive
3	33	17	negative
4	45	94	positive
5	24	26	negative
6	55	24	negative
7	35	52	positive
8	57	65	positive
9	45	32	negative
10	52	75	positive
11	62	31	negative



Making a decision

- ⋆ to make a decision
  - weighted coordinates are combined to form a 'cancer score'
  - the resulting score is then compared to a threshold value

- ⋆ in our cancer diagnosis example
  - for input $\mathbf{x} = (x_1, \ldots, x_d)$, 'attributes of a patient':

$$\text{cancer positive if } \sum_{i=1}^{d} w_i x_i > \text{threshold}$$

$$\text{cancer negative if } \sum_{i=1}^{d} w_i x_i < \text{threshold}$$

The 'perceptron'

⋆ this linear formula $h \in \mathcal{H}$ can be written more compactly as

$$h(\mathbf{x}) = \text{sign}\left(\left(\sum_{i=1}^{d} w_i x_i\right) - \text{threshold}\right) \quad (1)$$

$$= \text{sign}\left(\left(\sum_{i=1}^{d} w_i x_i\right) + b\right) \quad (2)$$

where b is called the *bias* and $\text{sign}(s) = \begin{cases} +1 & \text{if } s > 0 \\ -1 & \text{if } s < 0 \end{cases}$

⋆ this model of $\mathcal{H}$ is called the *perceptron*

- ⋆ different values for the parameters w_1, w_2, b
  - ▶ correspond to different lines $w_1 x_1 + w_2 x_2 + b = 0$
- ⋆ for simplification
  - ▶ treat the bias b as a weight $w_0 \equiv b$
  - ▶ introduce an artificial coordinate $x_0 \equiv 1$
- ⋆ with this convention, $\mathbf{w}^{\mathrm{T}}\mathbf{x} = \sum_{i=0}^{d} w_i x_i$
  - ▶ this gives the perceptron in the vector form:

$$\boxed{h(\mathbf{x}) = \mathrm{sign}(\mathbf{w}^{\mathrm{T}}\mathbf{x})} \tag{3}$$

Three linear models

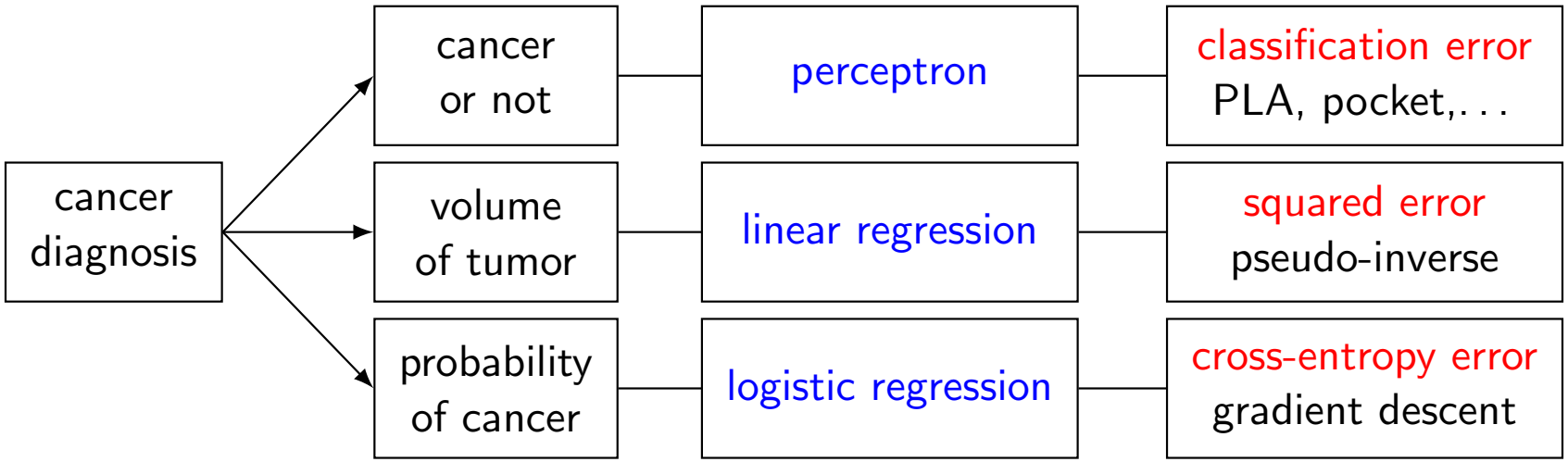


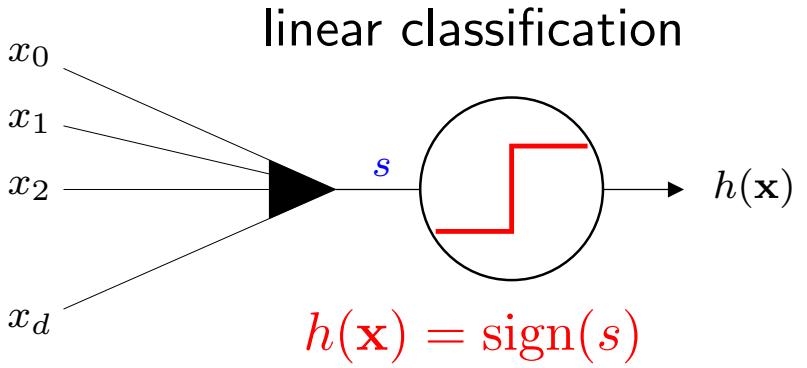
Figure 1 : comparison of the three linear models considered

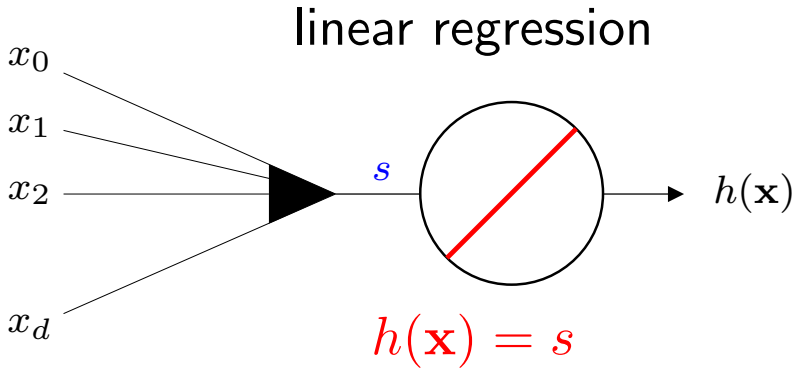
⋆ have their respective goals, error measures, and algorithms

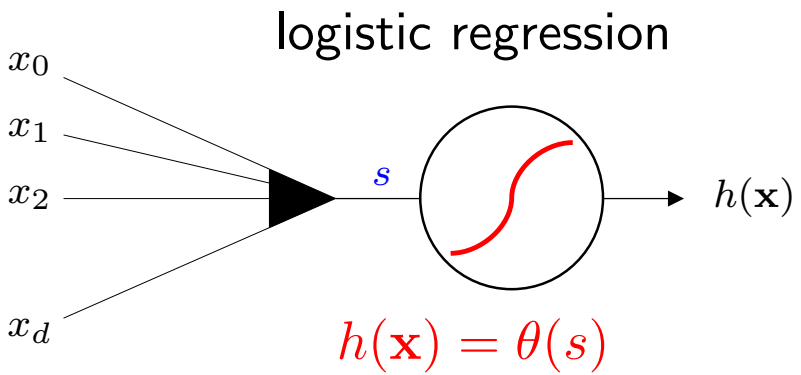
- nonetheless, share similar sets of linear hypotheses

- three linear models based on "signal" s:

$$s = \sum_{i=0}^{d} w_i x_i$$







source: [Abu-Mostafa et al., 2012]

Comparison

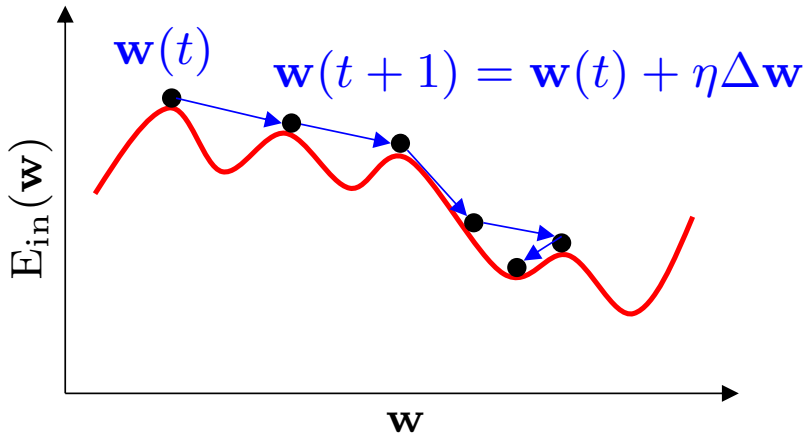
	linear classification	linear regression	logistic regression
$\mathcal{Y}$	$\{-1, +1\}$	$\mathbb{R}$	$\{-1, +1\}$
$\hat{y} = h(\mathbf{x})$	$\text{sign}(\mathbf{w}^{\text{T}}\mathbf{x})$	$\mathbf{w}^{\text{T}}\mathbf{x}$	$\theta^{\star}(\mathbf{w}^{\text{T}}\mathbf{x})$
e$(\hat{y}, y)$	0-1 loss $[\![\hat{y} \neq y]\!]$	squared error $(\hat{y} - y)^2$	cross-entropy error $[\![y{=}+1]\!] \ln \frac{1}{\hat{y}}$ $+[\![y{=}-1]\!] \ln \frac{1}{1-\hat{y}}$
$\text{E}_{\text{in}}(h)$	$\frac{1}{N}\sum_{n=1}^{N}[\![h(\mathbf{x}_n){\neq}y_n]\!]$	$\frac{1}{N}\sum_{n=1}^{N}(h(\mathbf{x}_n){-}y_n)^2$	$\frac{1}{N}\sum_{n=1}^{N}\ln\left(1{+}e^{-y_n\mathbf{w}^{\text{T}}\mathbf{x}_n}\right)$
opt.	combinatorial optimization (NP-hard)	set $\nabla \text{E}_{\text{in}}(\mathbf{w}) = 0$ (closed-form solution exists)	set $\nabla \text{E}_{\text{in}}(\mathbf{w}) = 0$ iterative optimization (*e.g.*, gradient descent)

$\star$ logistic sigmoid $\theta(s) = 1/(1 + e^{-s})$

Method of gradient descent

⋆ gradient descent: iteratively solves unconstrained optimization

- general technique for minimizing twice-differentiable functions
- batch gradient descent: $\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla \mathrm{E}_{\mathrm{in}}(\mathbf{w})$ [over all data]
- stochastic gradient descent: $\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla \mathrm{e}_n(\mathbf{w})$ [pointwise]
- minibatch gradient descent, batch normalization

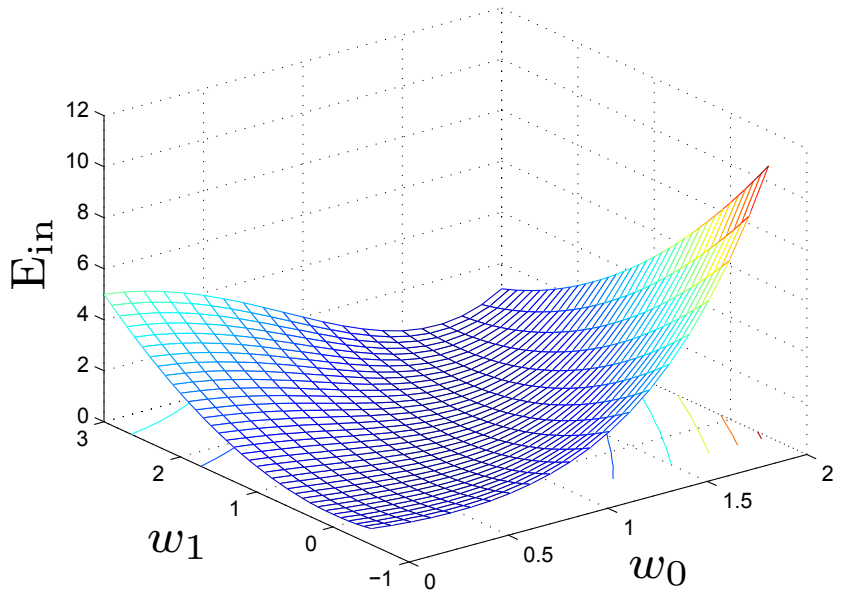


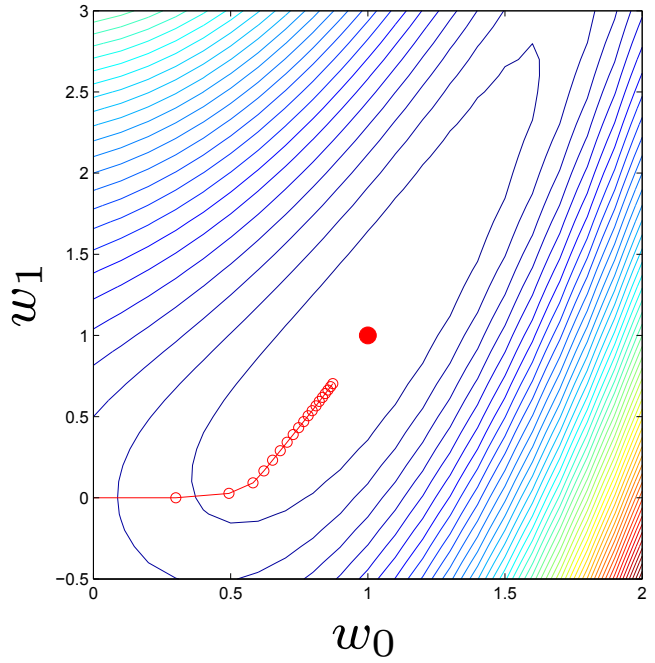
$$\Delta \mathbf{w} = -\nabla \mathrm{E}_{\mathrm{in}}(\mathbf{w}(t))$$

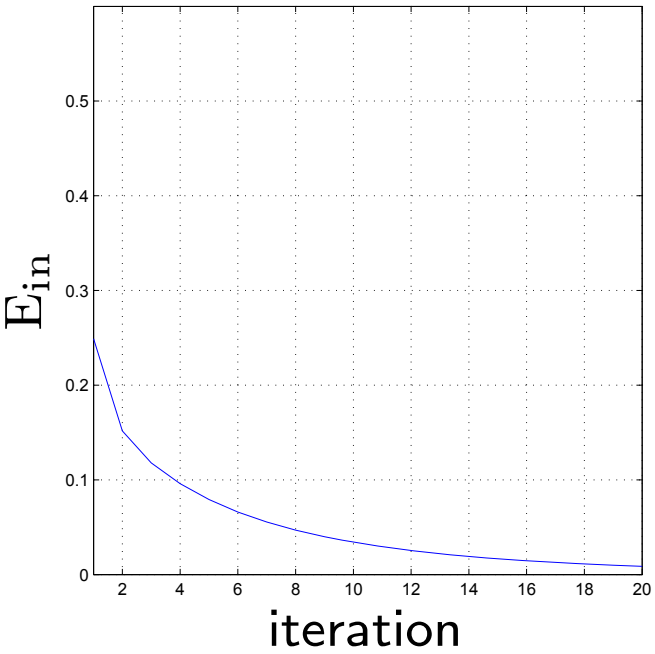
$\eta =$ learning rate

Example

- ⋆ assume a simple function for $\mathrm{E_{in}}$
  - global minimum 0 at (1,1)

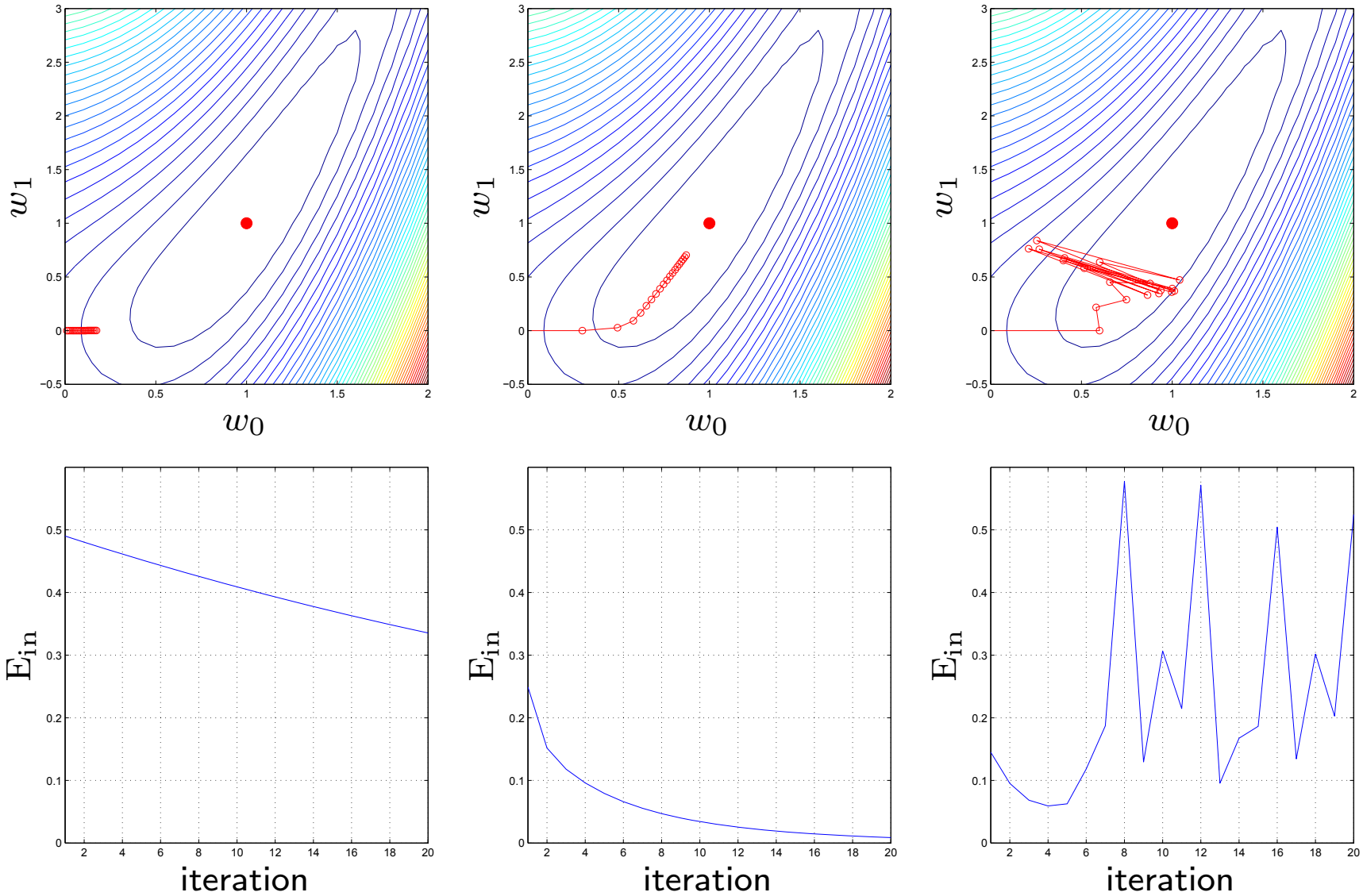






- ⋆ gradient descent starting at (0,0)
  - # iterations (steps) $= 20$
  - step size (learning rate) $\eta = 0.3$

$\eta = 0.01$ $\eta = 0.3$ $\eta = 0.6$

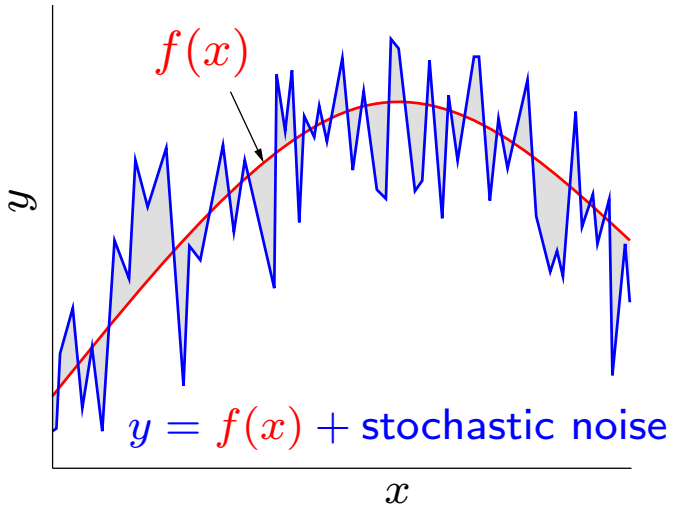


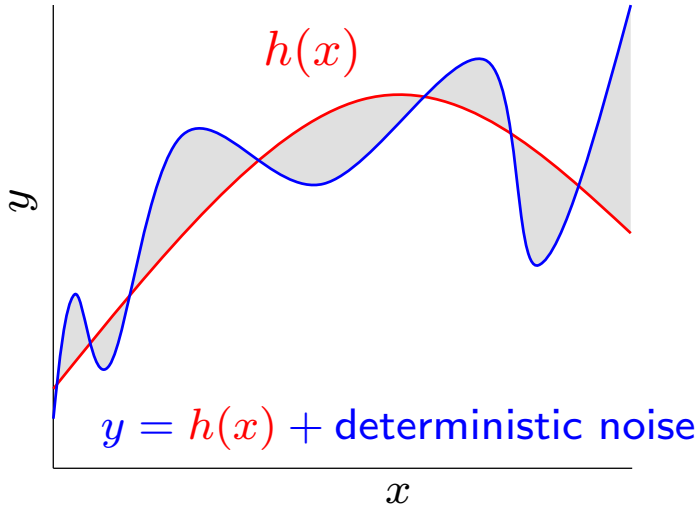
Feasibility of learning [Abu-Mostafa et al., 2012]

- ⋆ learning *unknown* target f: only possible in a probabilistic sense
  - we need hypothesis $h \approx f$, which means $\mathrm{E}_{\mathrm{out}}(h) \approx 0$
  - E_{in}: in-sample (training) error, $\mathrm{E}_{\mathrm{out}}$: out-of-sample (test) error
- ⋆ feasibility of learning is split into two questions
  1. can we make sure that $\mathrm{E}_{\mathrm{out}}(h) \approx \mathrm{E}_{\mathrm{in}}(h)$?
  2. can we make $\mathrm{E}_{\mathrm{in}}(h) \approx 0$?
- ⋆ answering Q1: theoretical
  - Hoeffding's inequality: $\mathbb{P}[|\mathrm{E}_{\mathrm{in}}(h) - \mathrm{E}_{\mathrm{out}}(h)| > \epsilon] \leq 2Me^{-2\epsilon^2 N}$
  - generalization theory (*e.g.*, VC analysis and bias-variance analysis)
- ⋆ answering Q2: practical
  - various learning paradigms will help us achieve that

Noise: part of y we cannot model

- ⋆ stochastic/deterministic noise hurts learning by leading to overfitting





Overfitting

⋆ overfitting: common phenomenon in learning [Abu-Mostafa et al., 2012]

- meaning: fitting data more than is warranted
- behavior: selecting lower E_{in} gives higher $\mathrm{E}_{\mathrm{out}}$
- consequence: bad generalization
- cause: deterministic/stochastic noise (part of y we cannot model)

$$y = \underbrace{h(\mathbf{x}) + \text{deterministic noise}}_{f(\mathbf{x})} + \text{stochastic noise}$$

$$\mathbb{E}_{\mathcal{D}}[\mathrm{E}_{\mathrm{out}}] = \underbrace{\text{var}}_{\substack{\uparrow \\ \text{indirect impact of noise}}} + \underbrace{\text{bias}}_{\substack{\uparrow \\ \text{deterministic noise}}} + \underbrace{\sigma^2}_{\substack{\uparrow \\ \text{stochastic noise}}}$$

number of data points	↑	overfitting	↓
deterministic noise	↑	overfitting	↑
stochastic noise	↑	overfitting	↑

⋆ cure for overfitting: regularization and validation

Regularization

⋆ regularization: constraining a model not to fit noise (a must in practice)

- effective when stochastic/deterministic noise is present

⋆ ways to constrain a model (toward simpler/smoother direction)

1. hard-order constraint: use a low-order model
2. soft-order constraint: constrain model parameters
3. augmented error: use a better proxy for $\mathrm{E}_{\mathrm{out}}$ than E_{in}

$$\mathrm{E}_{\mathrm{aug}}(h, \lambda, \Omega) = \mathrm{E}_{\mathrm{in}}(h) + \underbrace{\overbrace{\frac{\lambda}{N}}^{\text{regularization parameter}} \overbrace{\Omega(h)}^{\text{regularizer}}}_{\text{overfit penalty}} \tag{4}$$

⋆ choosing regularizer $\Omega(h)$: heuristic (popular: l_1 lasso, l_2 ridge, dropout)

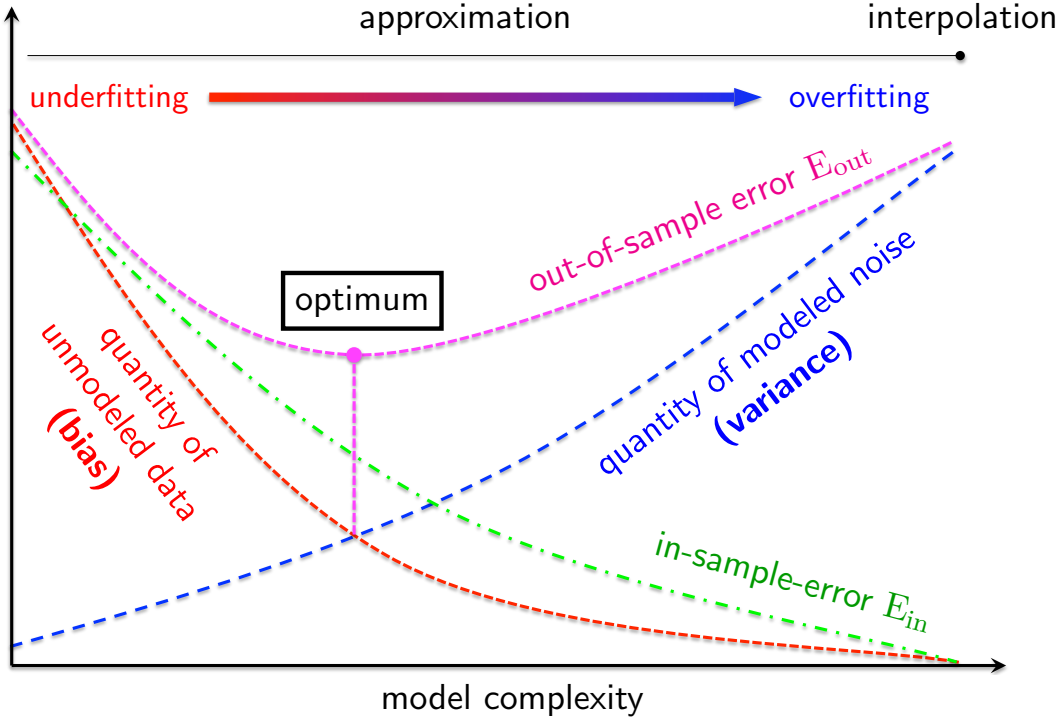
⋆ selecting regularization parameter λ: by validation ($\lambda = 0$ for wrong Ω)

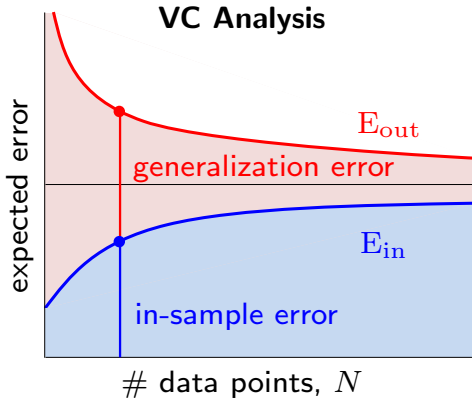
Bias-variance tradeoff

⋆ true in all types of machine learning [Geman et al., 1992]

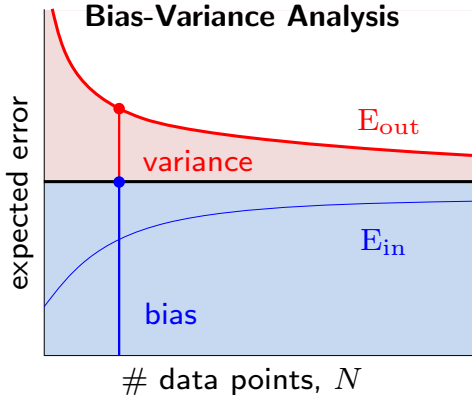
- approximation - generalization tradeoff

$\underbrace{\text{approximation}}_{E_{in}\approx 0,\ \text{bias}}$ - $\underbrace{\text{generalization}}_{E_{out}\approx E_{in},\ \text{var}}$ tradeoff





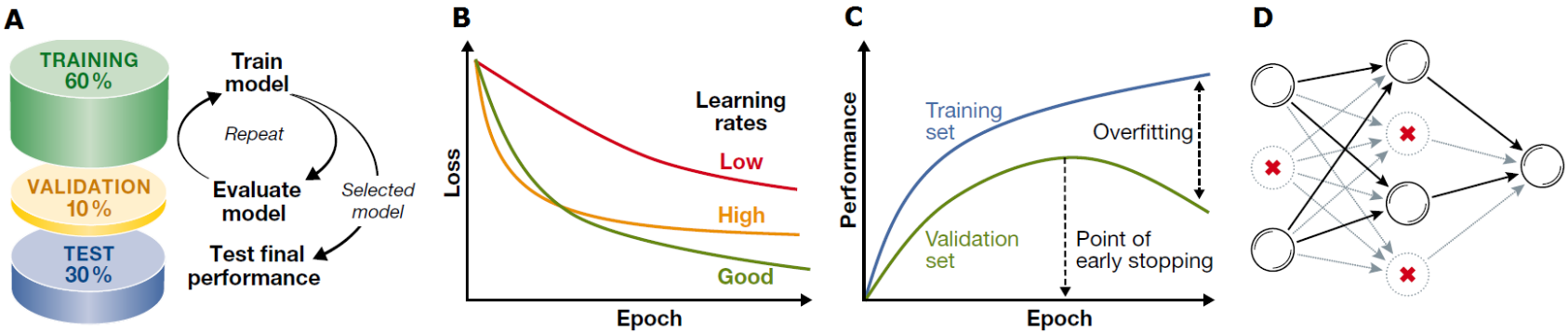
$$E_{out} \leq E_{in} + \Omega$$



$$E_{out} = \text{bias} + \text{var}$$

source: [Abu-Mostafa et al., 2012]

Model selection and validation



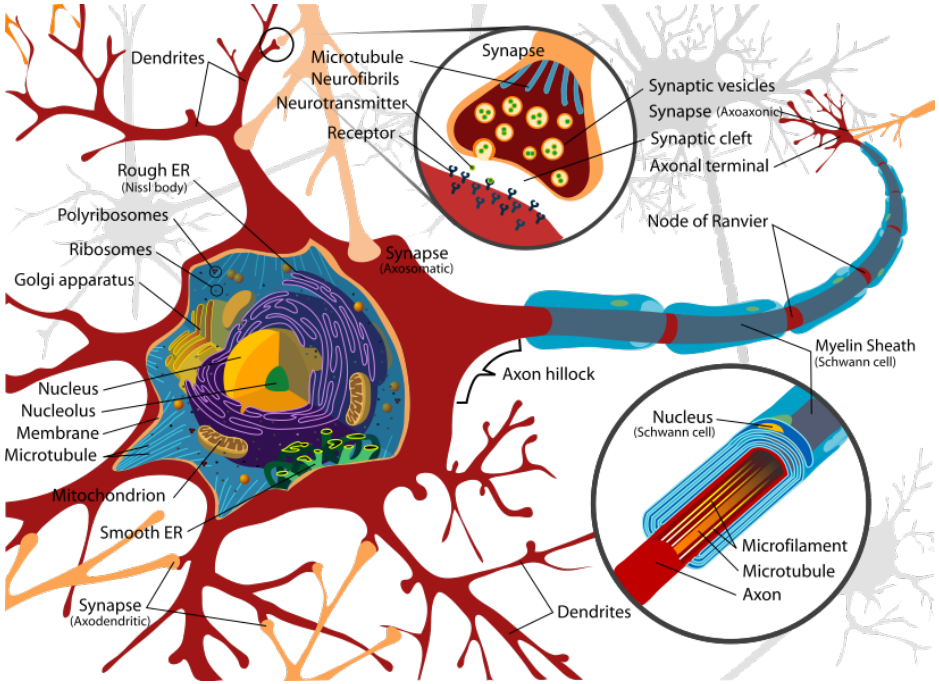
source: [Angermueller et al., 2016]

- training-validation-testing (A)
- effect of learning rate on training (B)
- overfitting (C): decreasing E_{in} gives increasing $\mathrm{E}_{\mathrm{out}}$
- dropout regularization (D)

Outline

Neuron

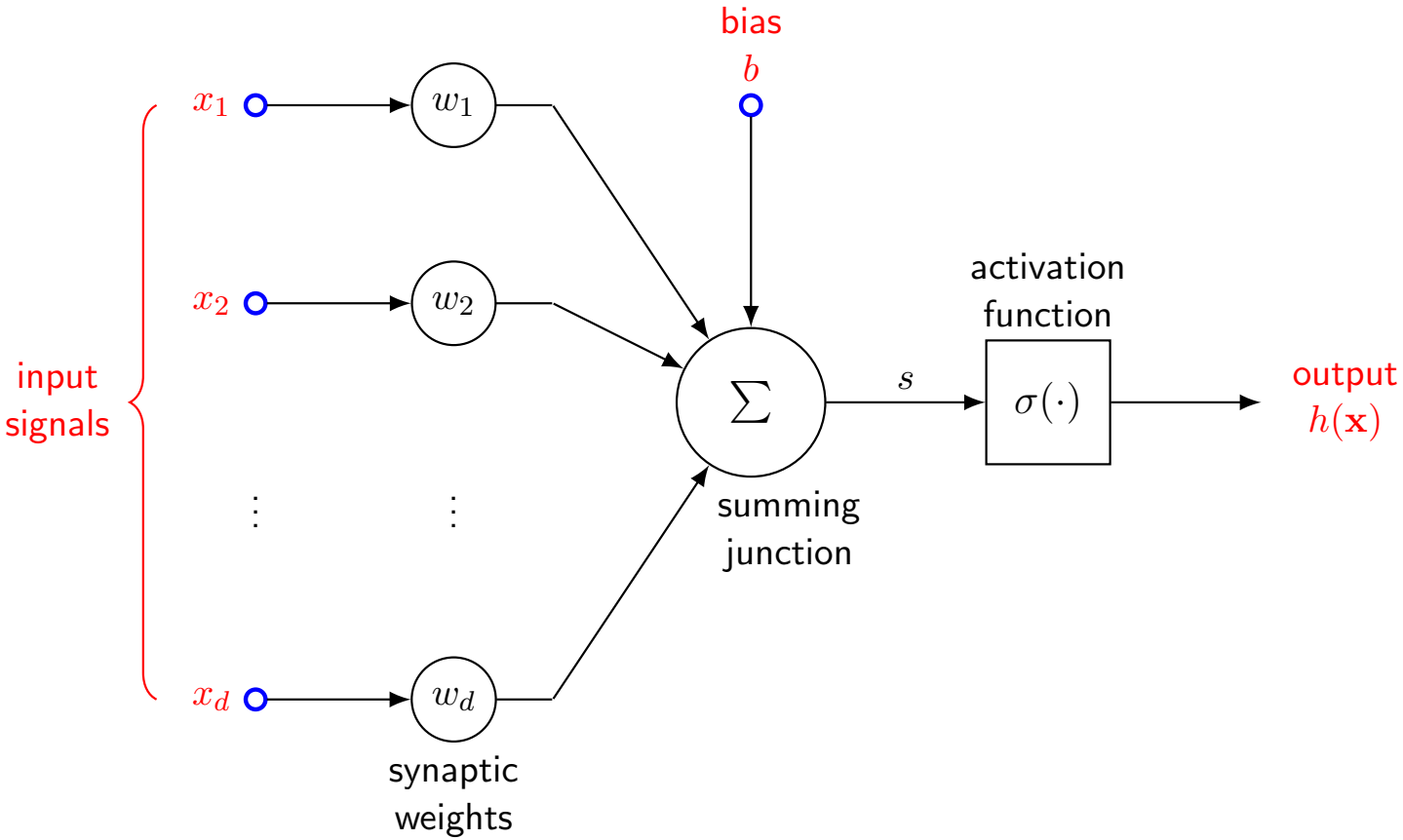
- ⋆ electrically excitable cell that processes and transmits information through electrical and chemical signals



source: http://en.wikipedia.org/wiki/Neuron

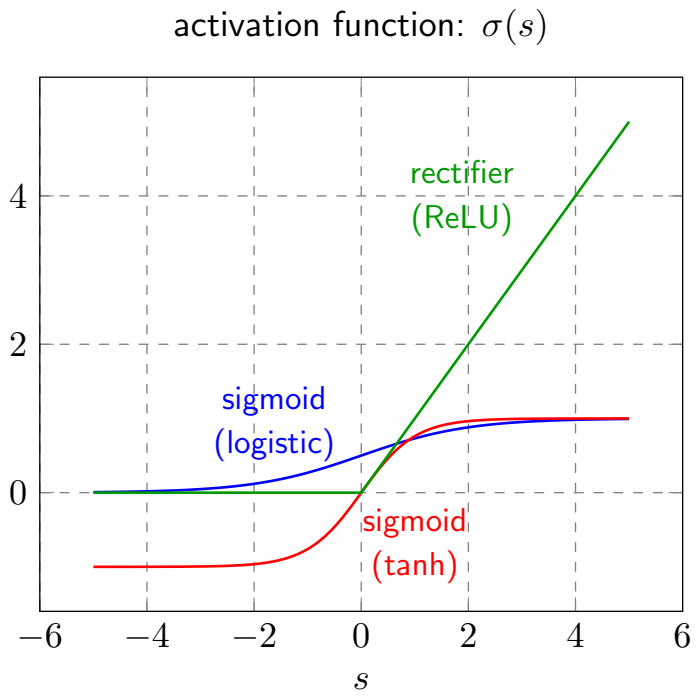
- ⋆ three basic elements of neural modeling
  1. synapses (with weights)
  2. adder (input vector $\rightarrow$ scalar)
  3. activation function (possibly nonlinear)

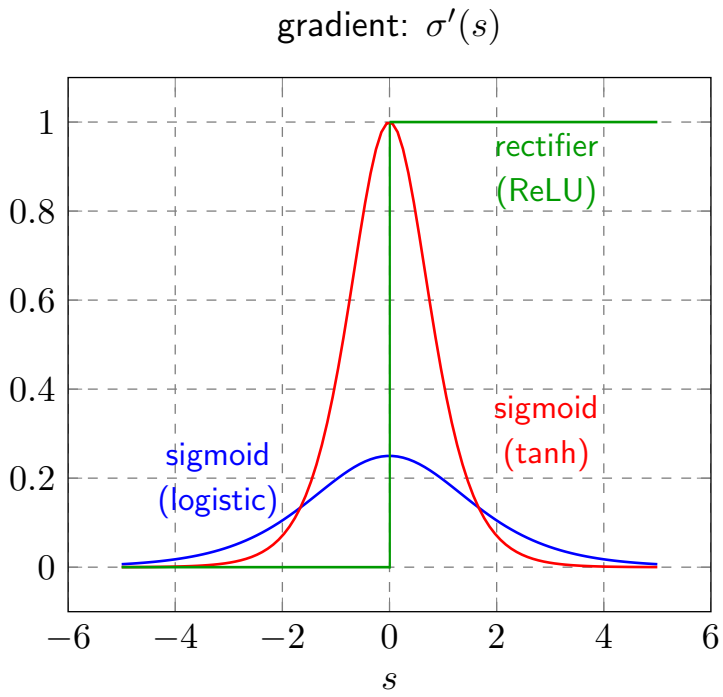
⋆ computational model of a neuron:



source: [Haykin, 2009]

Activation function



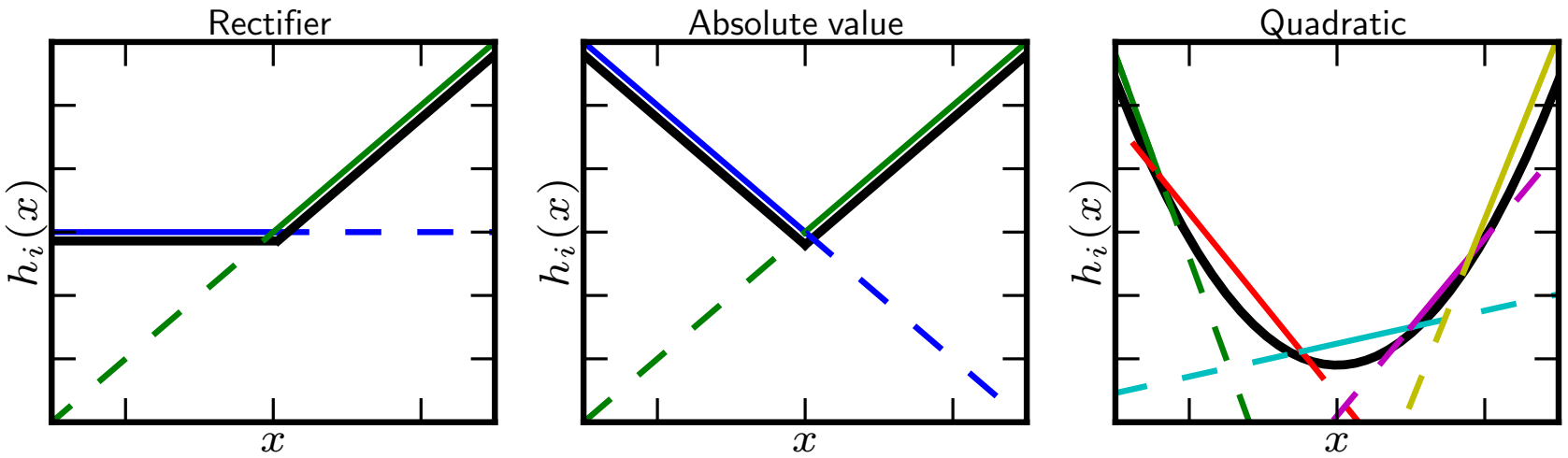


⋆ comparison

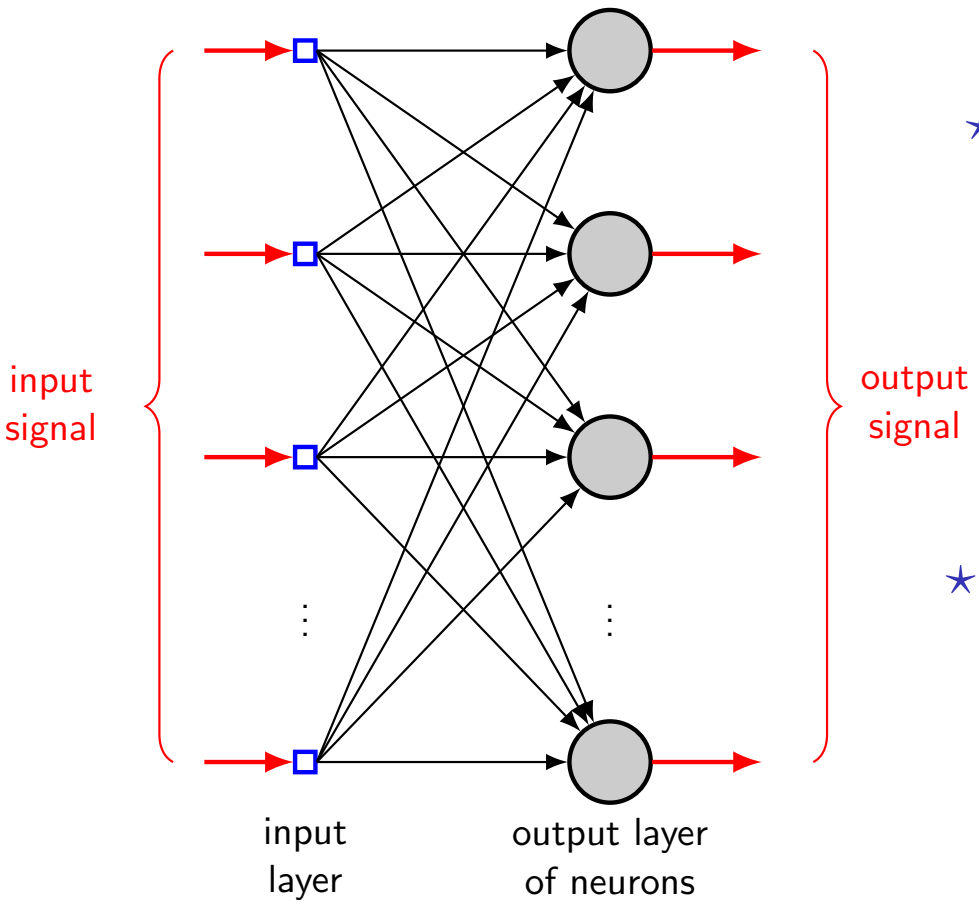
- logistic: suffers from the vanishing gradient problem
- tanh: better than logistic but the problem exists
- rectifier (ReLU): very popular in deep nets

Maxout

- ⋆ generalization of ReLU [Goodfellow et al., 2013]
  - output is the max value from k models from given input $\mathbf{x}$
  - can be used as a universal approximator



Single-layer feedforward network

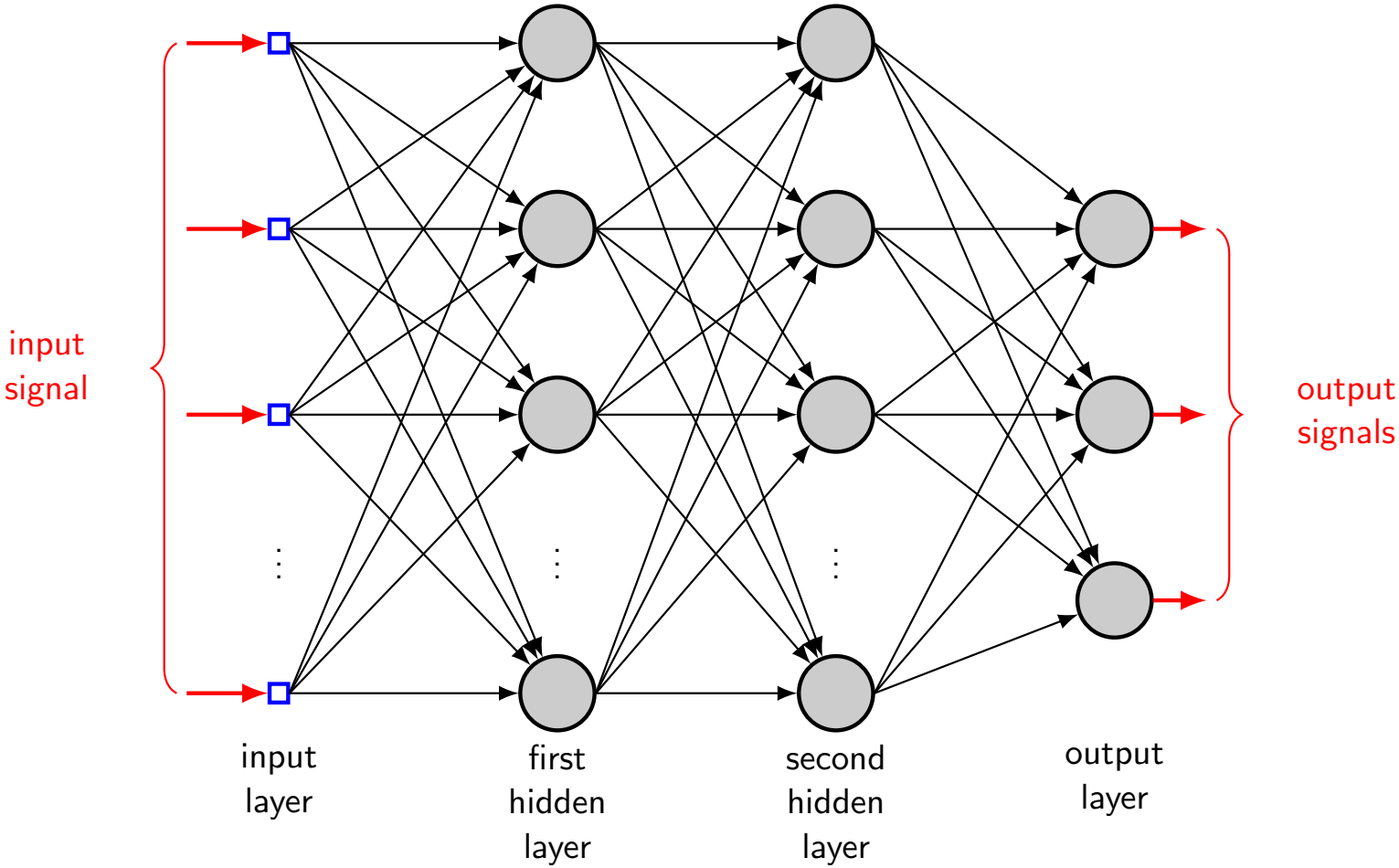


- ⋆ "layered"
  - input layer (of sources)
  - output layer (of neurons)
- ⋆ "feedforward"
  - (function) signal direction: input $\rightarrow$ output
  - not vice versa

source: [Haykin, 2009]

Multilayer feedforward network (multilayer perceptron)

- ⋆ one or more *hidden layers* of neurons



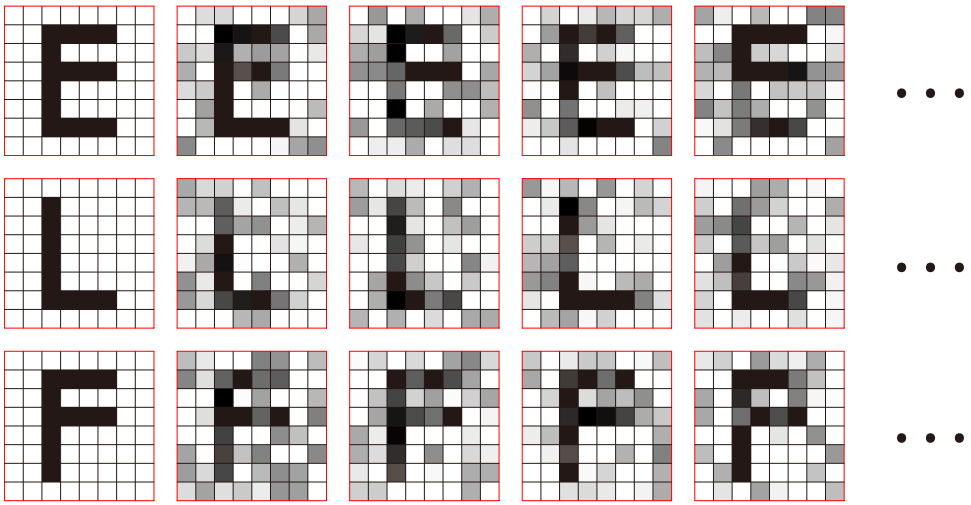
source: [Haykin, 2009]

Function of hidden neurons

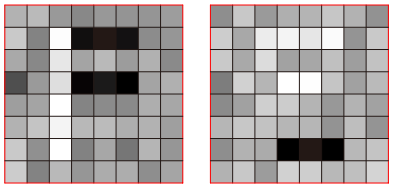
- ⋆ play critical role in operation of multilayer perceptron
  - ▶ each layer corresponds to "distributed representation"
- ⋆ hidden neurons act as feature detectors
  - ▶ as learning goes on, they gradually "discover" salient features characterizing training data
  - ▶ they do so by performing nonlinear transformation on input data into new space called *feature space*

Example

(a) sample training patterns



(b) learned input-to-hidden weights



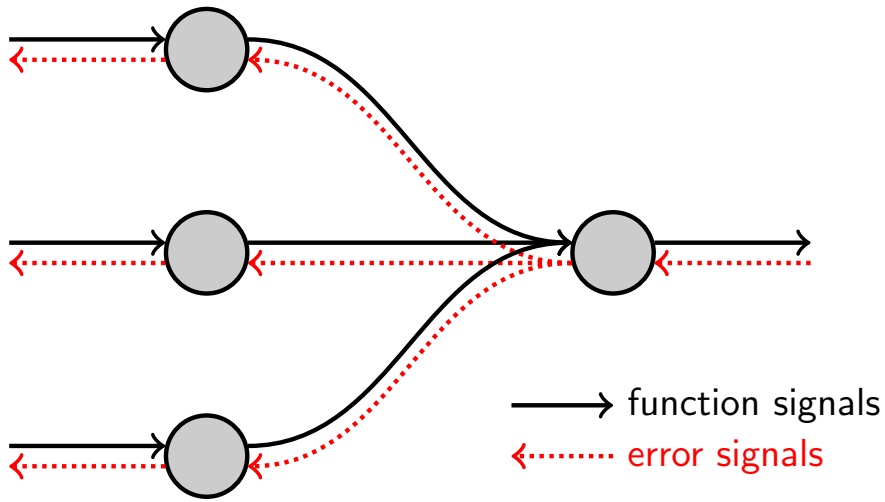
source: [Duda et al., 2012]

- ⋆ 64-2-3 network for classifying 3 characters
  - 64-dim inputs
  - 2 hidden units
  - 3 output units
- ⋆ learned i-to-h weights
  - describe feature groupings useful for classification

Types of signals in neural networks

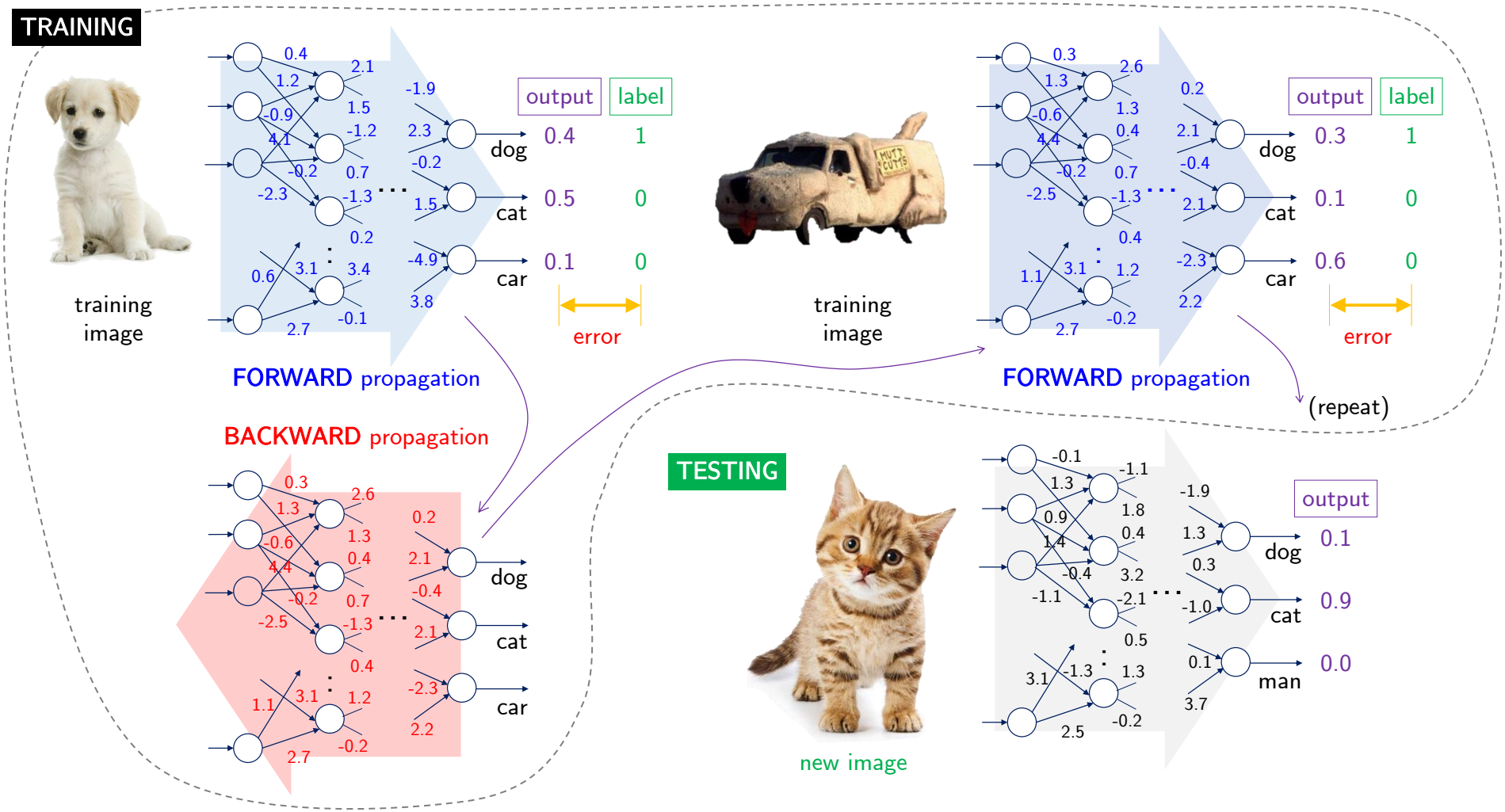
1. function signals: forward propagation
   - input signal comes in at input
   - propagates forward through network, and
   - emerges at output

2. error signals: back(ward) propagation
   - originates at an output neuron, and
   - propagates backward



source: [Haykin, 2009]

Training by forward/backward propagation



Training neural networks

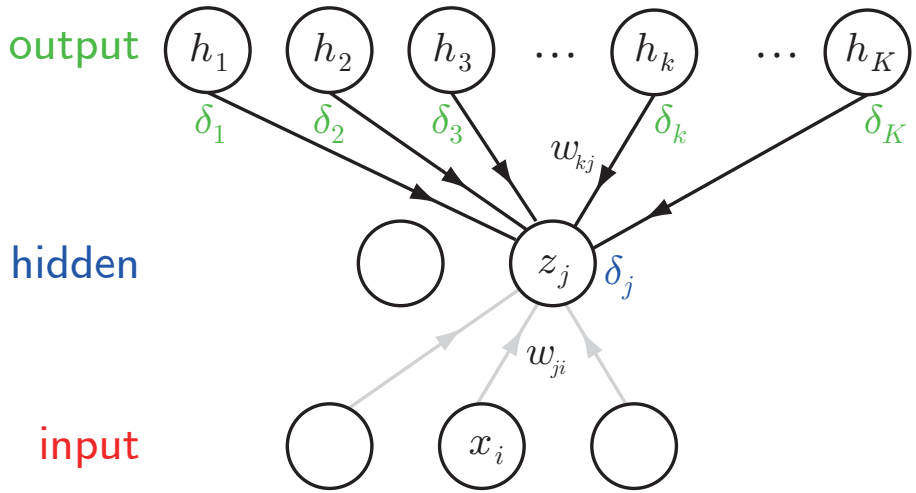
⋆ forward propagation: send input $\mathbf{x}$ via hidden $\mathbf{z}$ to output $\mathbf{h}$

- activation function:

$$\sigma(s) = 1/(1+e^{-s})$$

- hidden layer:

$$z_j(\mathbf{x}) = \sigma\left(\sum_i w_{ji} x_i\right)$$



- output layer:

$$h_k(\mathbf{x}) = \sigma\left(\sum_j w_{kj} z_j\right) = \sigma\left(\sum_j w_{kj}\ \sigma\left(\sum_i w_{ji} x_i\right)\right)$$

⋆ back propagation: enables us to train hidden weights

- input-to-hidden weights w_{ji}

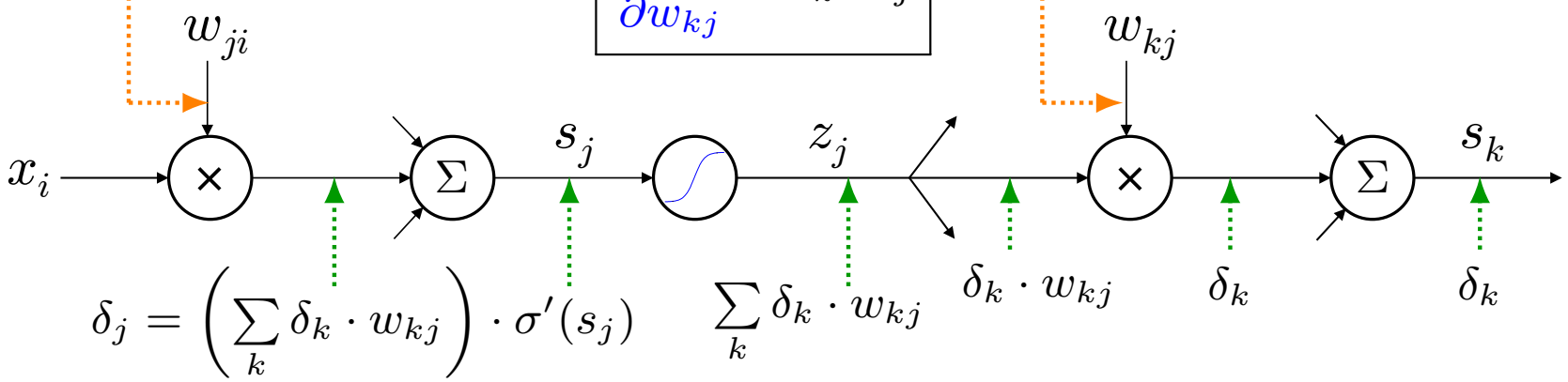
$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial w_{ji}} &= \delta_j \cdot x_i \\ \delta_j &= \sigma'(s_j) \cdot \sum_k w_{kj}\delta_k \\ \Delta w_{ji} &= -\eta \cdot \delta_j \cdot x_i\end{aligned}$$

- hidden-to-output weights w_{kj}

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial w_{kj}} &= \delta_k \cdot z_j \\ \delta_k &= \sigma'(s_k) \cdot e_k \\ \Delta w_{kj} &= -\eta \cdot \delta_k \cdot z_j\end{aligned}$$

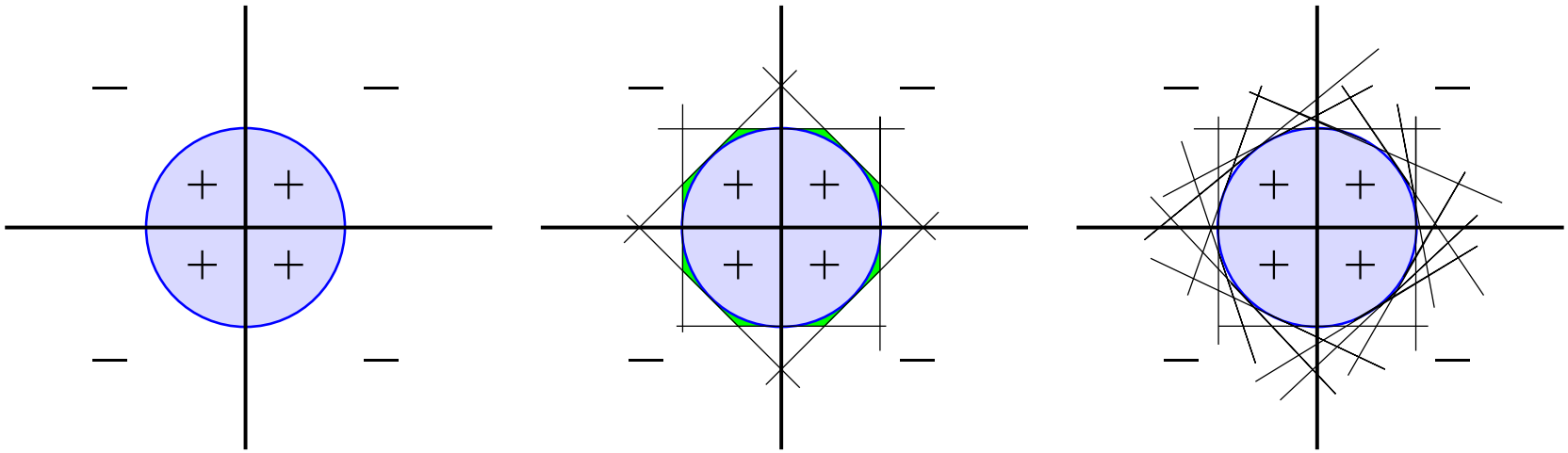
$$\left(\sum_k \delta_k \cdot w_{kj}\right) \cdot \sigma'(s_j) \cdot x_i \implies \boxed{\frac{\partial \mathcal{E}}{\partial w_{ji}} = \delta_j \cdot x_i = \left(\sum_k \delta_k \cdot w_{kj}\right) \cdot \sigma'(s_j) \cdot x_i}$$

$$\boxed{\frac{\partial \mathcal{E}}{\partial w_{kj}} = \delta_k \cdot z_j}$$



Neural networks: powerful model

- ⋆ universal approximators
  - GMM: universal *density* estimator (given enough Gaussians)
  - ANN: universal *function* estimator (given enough neurons)



target | 8 perceptrons | 16 perceptrons

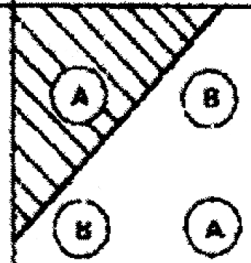
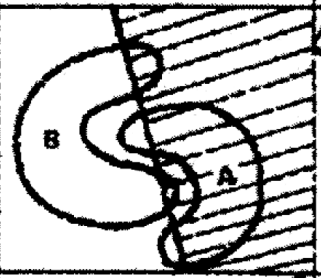
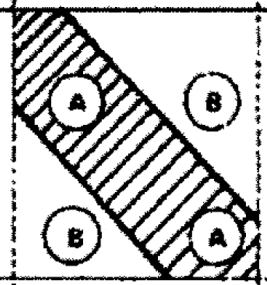
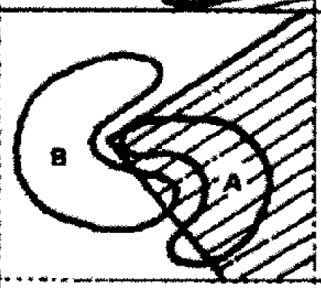
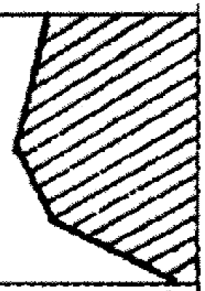
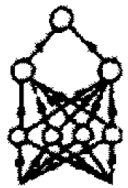
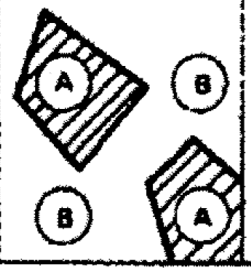
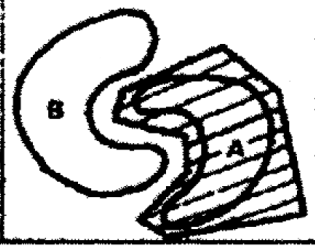
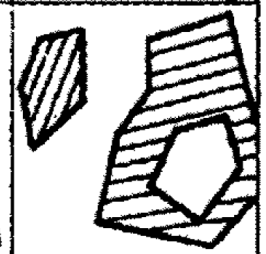
source: [Abu-Mostafa et al., 2012]

- ⋆ types of decision regions that can be formed
  - more layers → more "intelligent"

NO hidden layer

ONE hidden layer

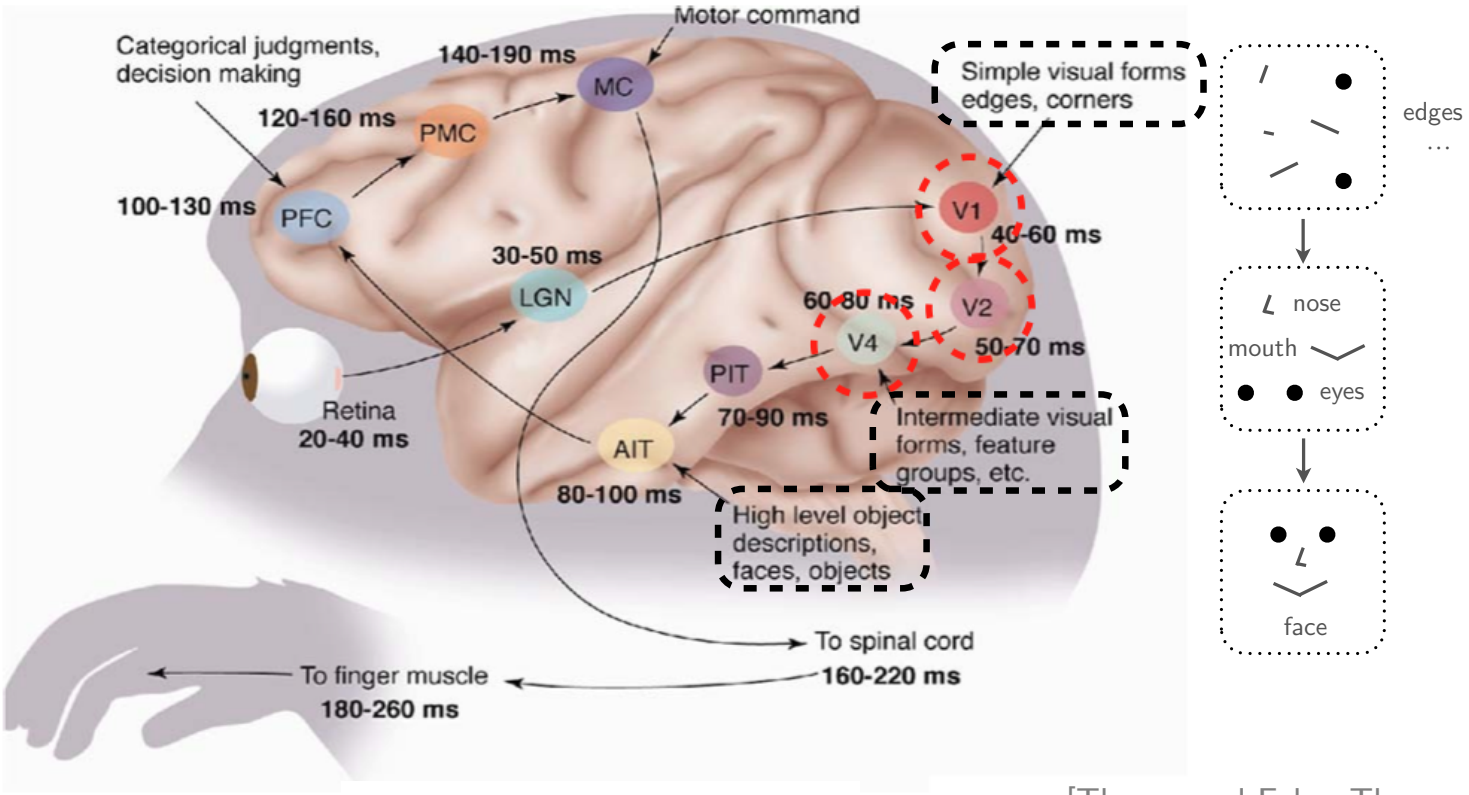
TWO hidden layers

STRUCTURE	TYPES OF DECISION REGIONS	EXCLUSIVE OR PROBLEM	CLASSES WITH MESHED REGIONS	MOST GENERAL REGION SHAPES
SINGLE LAYER 	HALF PLANE BOUNDED BY HYPERPLANE			
TWO LAYER 	CONVEX OPEN OR CLOSED REGIONS			
THREE-LAYER 	ARBITRARY (Complexity Limited By Number of Nodes)			

source: [Lippmann, 1987]

Inspiration from visual cortex

- ⋆ human brain: at least 5–10 layers for visual processing
  - ▶ hierarchical model needed for human-level intelligence



source: [Thorpe and Fabre-Thorpe, 2001]

Theoretical justification & challenge

- ⋆ DL can represent certain functions (exponentially) more compactly
- ⋆ example: Boolean functions
  - a sort of feed-forward network (hidden units are logic gates)
  - any Boolean function can be represented by a two-layer circuit with an exponential number of hidden units
  - the same function can be represented by a polynomial number of hidden units if we can adapt the number of layers
- ⋆ Huston, we have a problem!
  - training becomes significantly harder with more layers

Outline

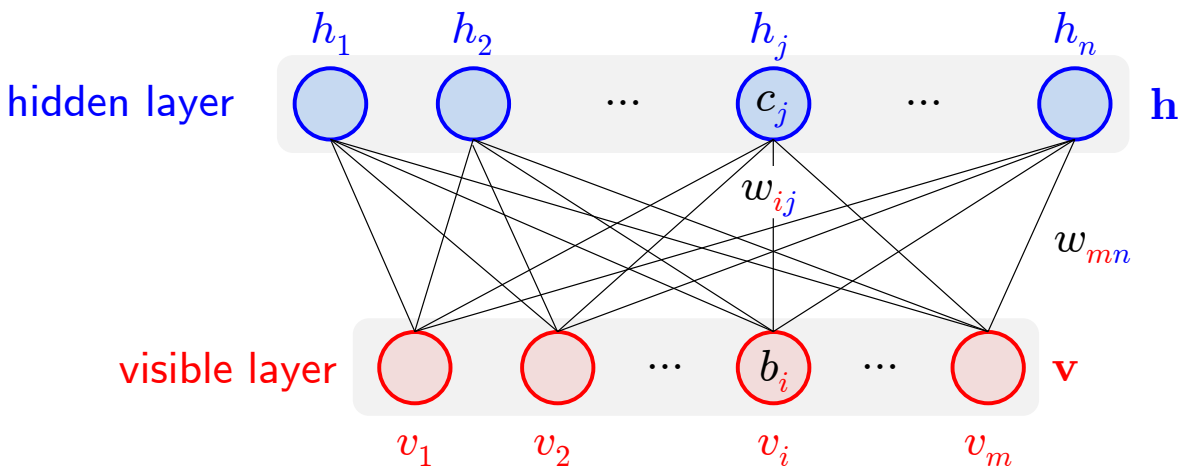
Unsupervised learning

- ⋆ aims at learning important aspects of unknown *target* distribution
  - ▶ based on samples from this target distribution
- ⋆ only use inputs for learning
  - ▶ automatically extract meaningful features for your data
  - ▶ leverage the availability of unlabeled data
- ⋆ examples of neural networks for unsupervised learning
  - ▶ (restricted) Boltzmann machines
  - ▶ autoencoders

Restricted Boltzmann machine (RBM)

- ⋆ a generative stochastic neural network
  - can learn a probability distribution over input data (parameterizes it with the Boltzmann distribution)
  - a popular building block of deep belief network (DBN) and other deep neural networks

- ⋆ representation: bipartite undirected graph ⇒ training feasible



⋆ energy-based model

- low energy $\Leftrightarrow$ high probability

$$p(\mathbf{v}) \propto \sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}$$

- training $=$ reshaping E so that desired $\mathbf{v}$ have high $p(\mathbf{v})$

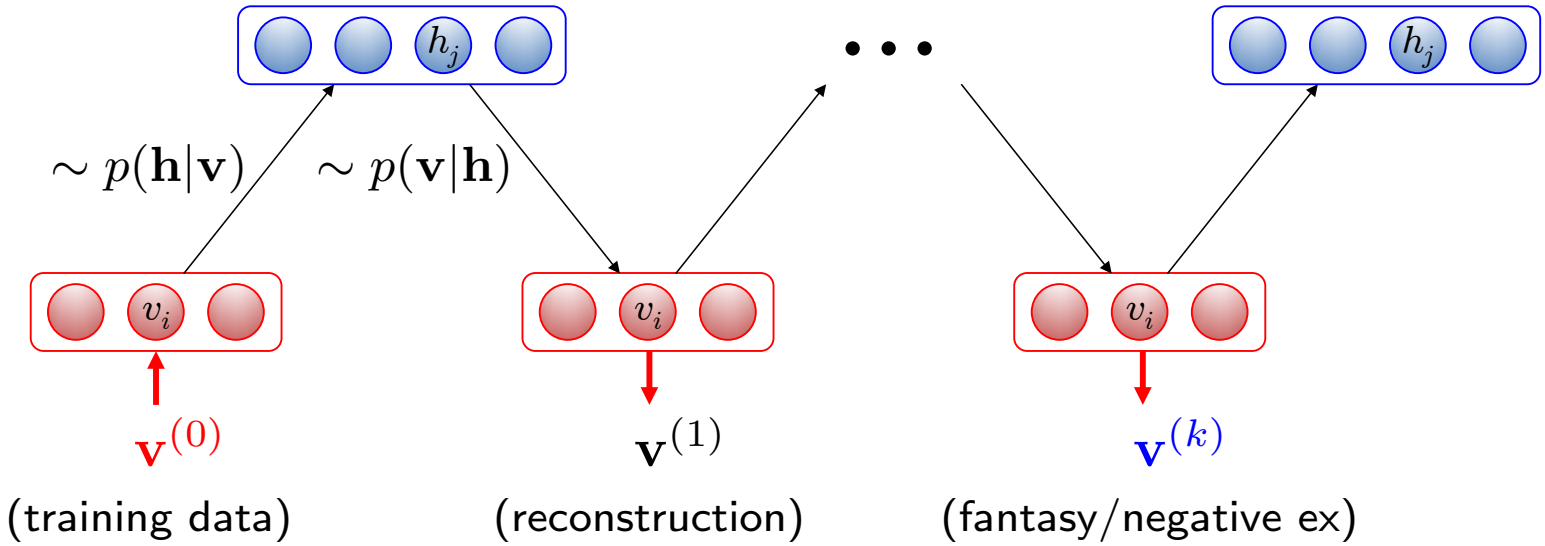
⋆ energy function in RBM

$$E(\mathbf{v},\mathbf{h}) = -\sum_{i=1}^{m}\sum_{j=1}^{n} w_{ij} v_i h_j - \sum_{i=1}^{m} b_i v_i - \sum_{j=1}^{n} c_j h_j$$

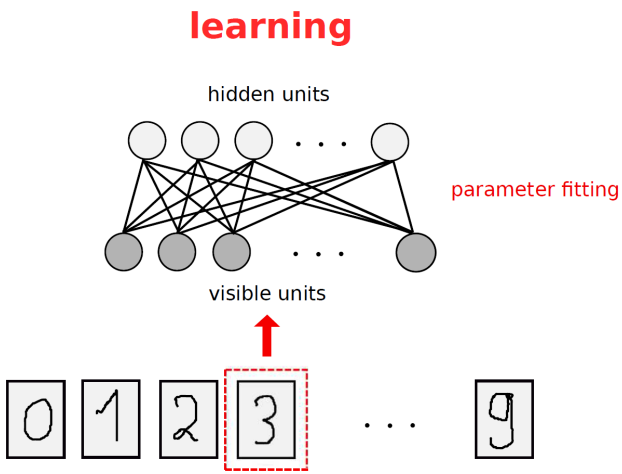
- ⋆ training objective: maximize data likelihood $\sum \log p(\mathbf{v})$
  - use stochastic gradient decent for maximization
  - contrastive divergence (CD-k): "approximate SGD"

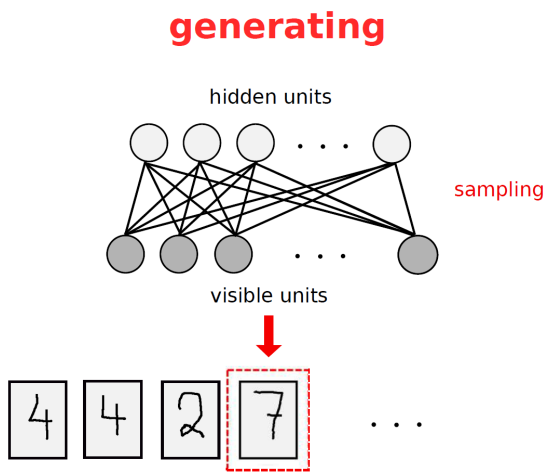
$$\frac{\partial \log p(\mathbf{v})}{\partial w_{ij}} = \underbrace{\mathbb{E}_{\mathbf{h}}\left[\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial w_{ij}}\right]}_{\text{positive phase (easy)}} - \underbrace{\mathbb{E}_{\tilde{\mathbf{v}},\mathbf{h}}\left[\frac{\partial E(\tilde{\mathbf{v}},\mathbf{h})}{\partial w_{ij}}\right]}_{\text{negative phase (hard)}}$$

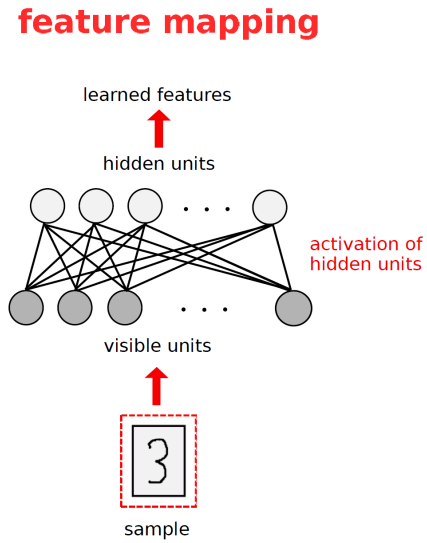
$$= \underbrace{v_i}_{\text{from data}} p(h_j|\mathbf{v}) - \underbrace{v_i^k}_{\text{by Gibbs sampling}} p(h_j|\mathbf{v}^k)$$

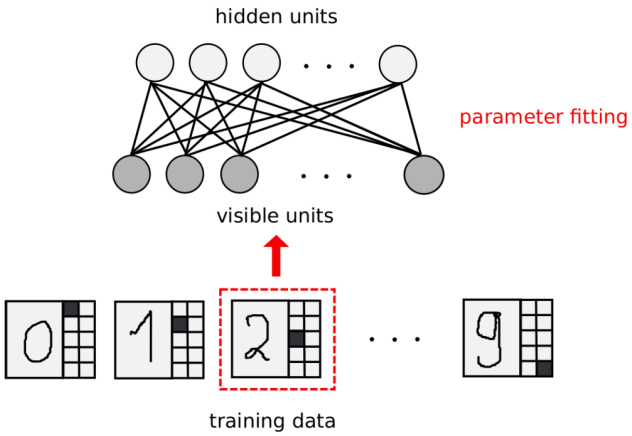


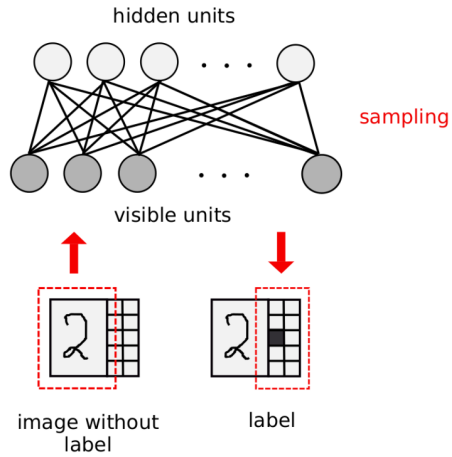
Applications of RBM









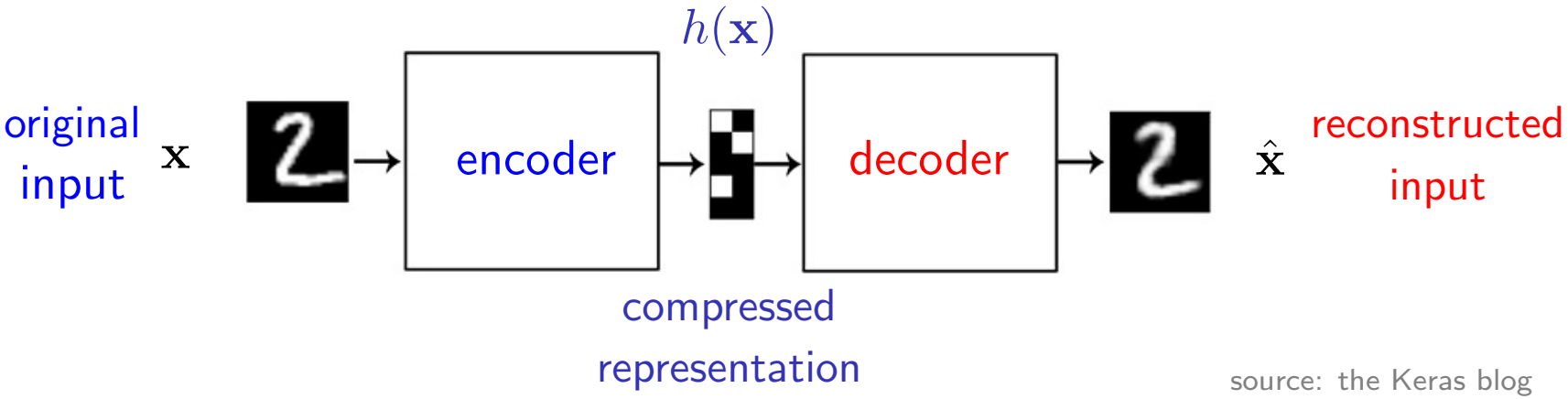


source: [Fischer and Igel, 2014]

Autoencoder

⋆ feed-forward NN trained to reproduce its input at the output

- training objective: minimize recon error (*e.g.*, $\min ||\hat{\mathbf{x}} - \mathbf{x}||$)



source: the Keras blog

⋆ the hidden layer: compressed representation of input

- data-specific, lossy, learned automatically from examples
- undercomplete or overcomplte

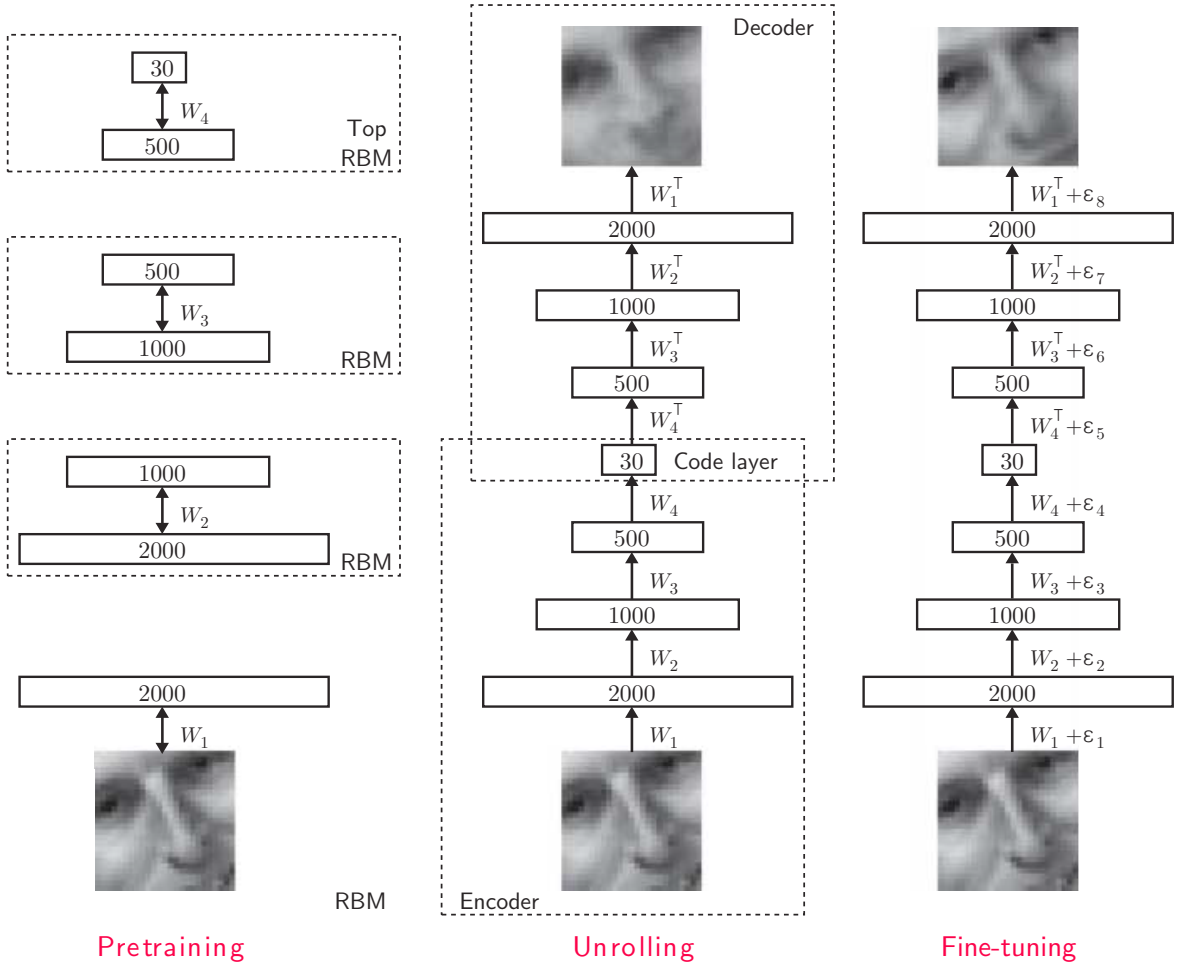
⋆ two main applications: data denoising and dimensionality reduction

Variations

- ⋆ denoising autoencoder [Vincent et al., 2008]
  - randomly corrupts input (*i.e.*, sets to 0)
  - discovers more robust features, avoiding simple identity
- ⋆ sparse autoencoder [Ng, 2011]
  - puts sparsity on (overcomplete) hidden units during training
  - can learn sparse representation of inputs
- ⋆ variational autoencoder (VAE) [Kingma and Welling, 2013]
  - inherits AE architecture (thus called autoencoder)
  - but makes strong assumptions on hidden variables (use the "variational Bayesian" method to learn latent representations)
  - a generative model (for handwritten digits, faces, and others)

Deep autoencoder [Hinton and Salakhutdinov, 2006]

- ⋆ pre-training can be used to initialize a deep autoencoder
  - ▶ better reconstruction than a single-layer network (*e.g.*, PCA)

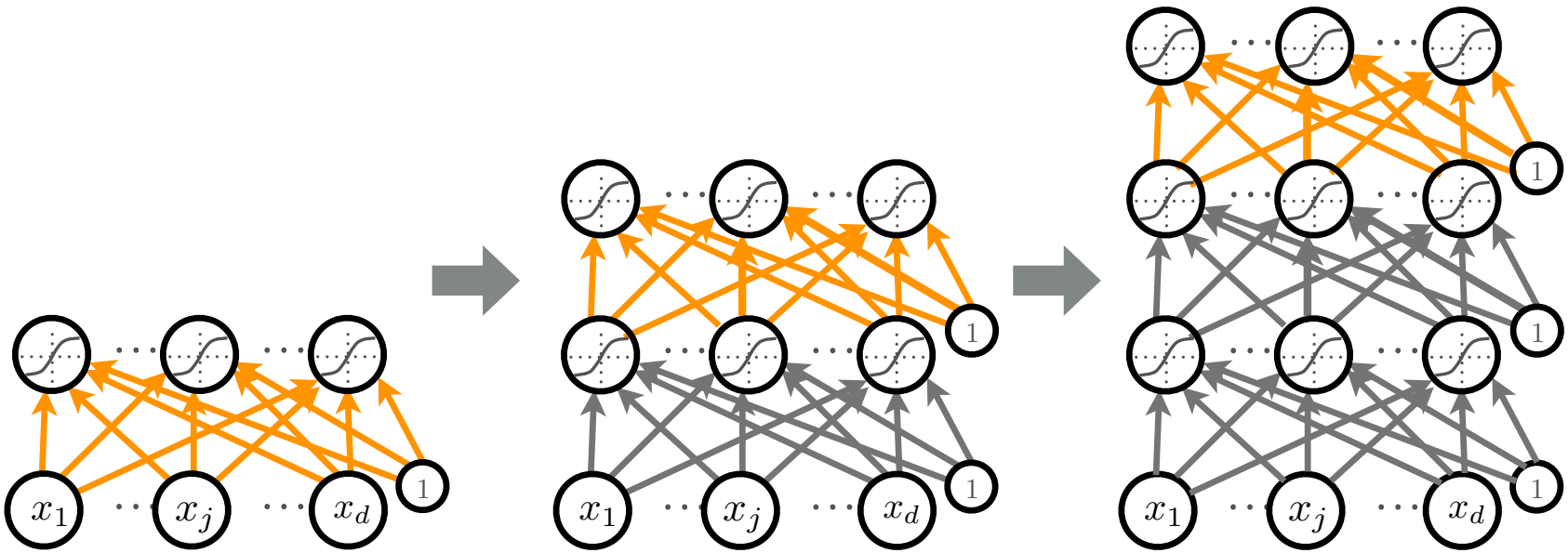


Types of training protocols [LeCun]

- ⋆ purely supervised
  - initialize parameters randomly
  - train in supervised mode (typically with SGD + backprop)
  - used in most practical systems for speech/image recognition
- ⋆ unsupervised, layerwise + supervised classifier on top
  - train each layer unsupervised (one after the other)
  - train a supervised classifier on top (the other layers: fixed)
  - useful when very few labeled examples are available
- ⋆ unsupervised, layerwise + global supervised fine-tuning
  - train each layer unsupervised (one after the other)
  - add a classifier layer and retrain the whole thing supervised
  - useful when the labeled training set is poor
- ⋆ unsupervised pre-training often uses regularized auto-encoders

Unsupervised pre-training

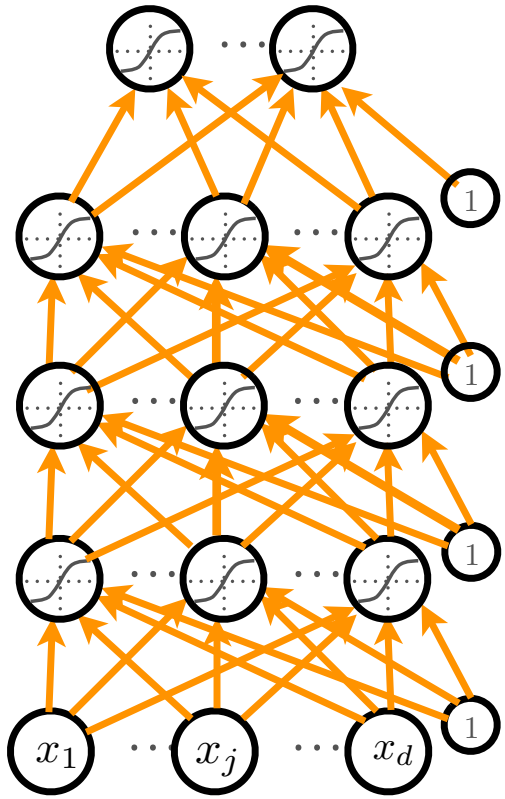
- ⋆ we use a greedy, layer-wise procedure
  - train one layer at a time (from first to last) with unsupervised criterion
  - fix the parameters of the previous hidden layers
  - previous layers viewed as feature extraction



source: Larochelle

Fine-tuning

- ⋆ once all layers are pre-trained
  - add output layer
  - train the whole net using supervised learning
- ⋆ supervised learning is performed
  - as in a regular feed-forward net
  - forward prop, backprop, and update
- ⋆ we call this last phase *fine-tuning*
  - all parameters are "tuned" for the supervised leaning task at hand
  - representation is adjusted to be more discriminative



source: Larochelle

Outline

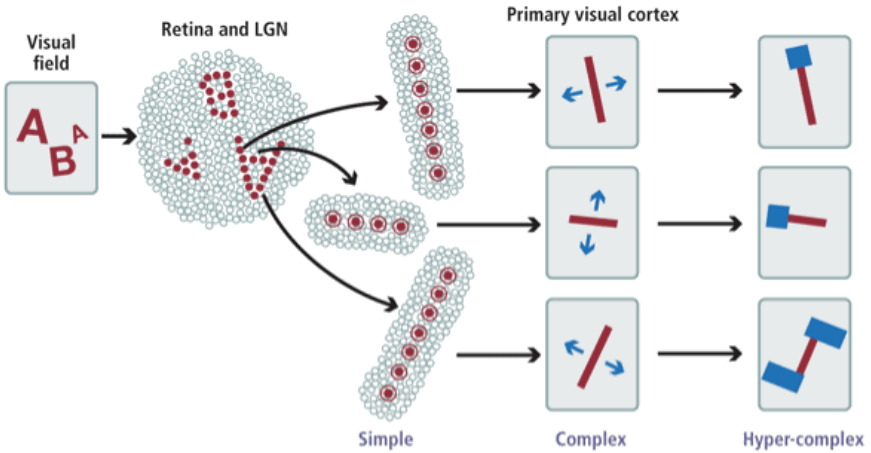
Convolutional neural networks (CNNs)

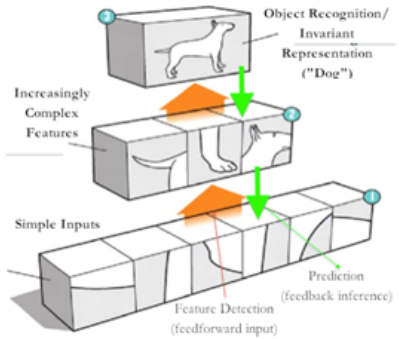
- ⋆ **simply neural networks that use convolution in their layers**
  - in place of general matrix multiplication
- ⋆ employ a mathematical operation called *convolution*
  - convolution: a special kind of linear operation
  - in ML, cross-correlation is often used instead
- ⋆ tremendously successful in practical applications
- ⋆ especially for data with a known, grid-like topology
  - time series: 1D grid of samples taken at regular time intervals
  - image: 2D grid of pixels

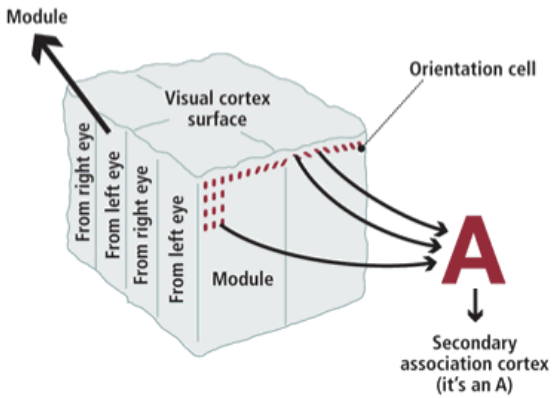
⋆ an example of neuroscientific principles influencing deep learning

ex) human vision system

- simple cells → complex cells → hyper-complex cells





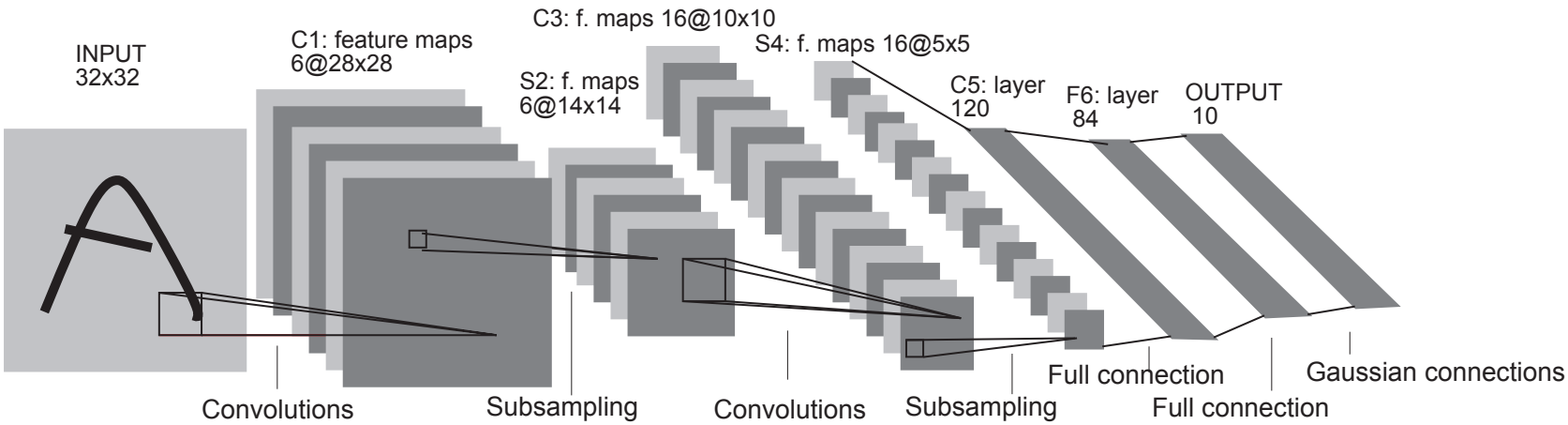


source: Univ. of Colorado, www.eidolonspeak.com

⋆ CNN leverages the following ideas [Goodfellow et al., 2016]

1. sparse interactions (aka local connectivity)
2. parameter sharing
3. equivariant[2] & invariant[3] representations

$\Rightarrow$ deliver efficiency & robustness



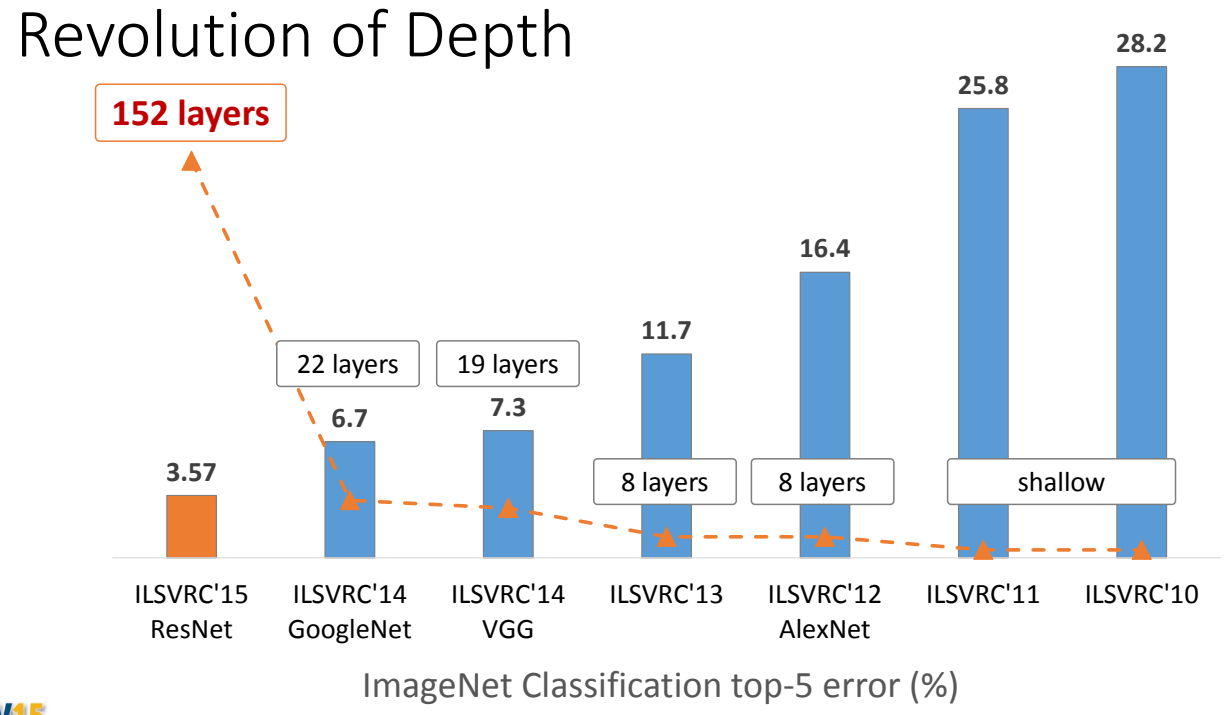
source: [LeCun et al., 2015b]

[2] represent(transform(x)) = transform(represent(x))
[3] represent(transform(x)) = represent(x)

Recent example: deep residual learning

- ⋆ winner of ILSVRC 2015 competitions: ResNet [He et al., 2015]
  - ▶ "ultra-deep" (152 layers)

Microsoft Research



ICCV15 International Conference on Computer Vision

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". arXiv 2015.

Definition: convolution

- ⋆ for a function $x(t)$ and a weighting function w
  - a new function s provides a smoothed estimate of x

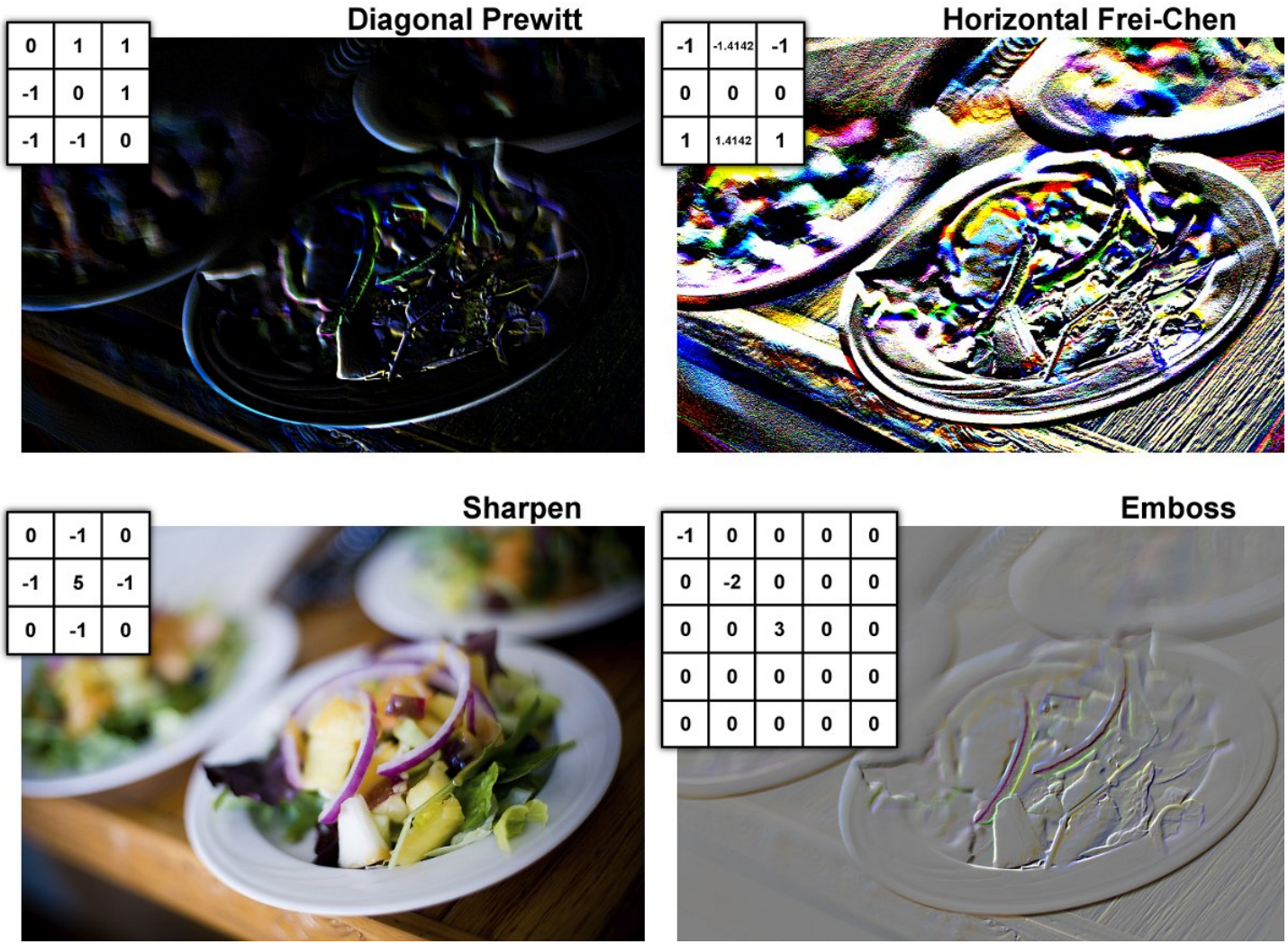
$$s(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a) \tag{5}$$

$$\triangleq (x * w)(t) \tag{6}$$

  - this operation is called *convolution* (symbol: $*$)
  - multi-dimensional convolution is similarly defined

- ⋆ in CNN terminology
  - the first argument (function x): *input*
  - the second argument (function w): *kernel* or *filter*
  - the output (function s): *feature map*

⋆ kernel (= *filter* or *convolution matrix*) examples



source: www.gimpbible.com

Cross-correlation

- ⋆ 2D convolution:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n)$$
$$= \sum_m \sum_n I(i-m, i-n)K(m,n) \tag{7}$$

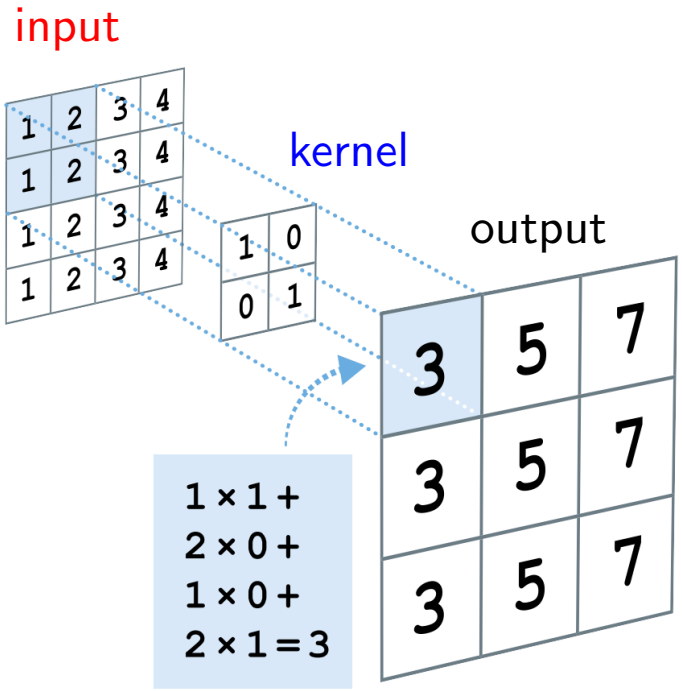
- ⋆ cross-correlation: same as convolution but without kernel flipping

$$S(i,j) = (I * K)(i,j)$$
$$= \sum_m \sum_n I(i+m, j+n)K(m,n) \tag{8}$$

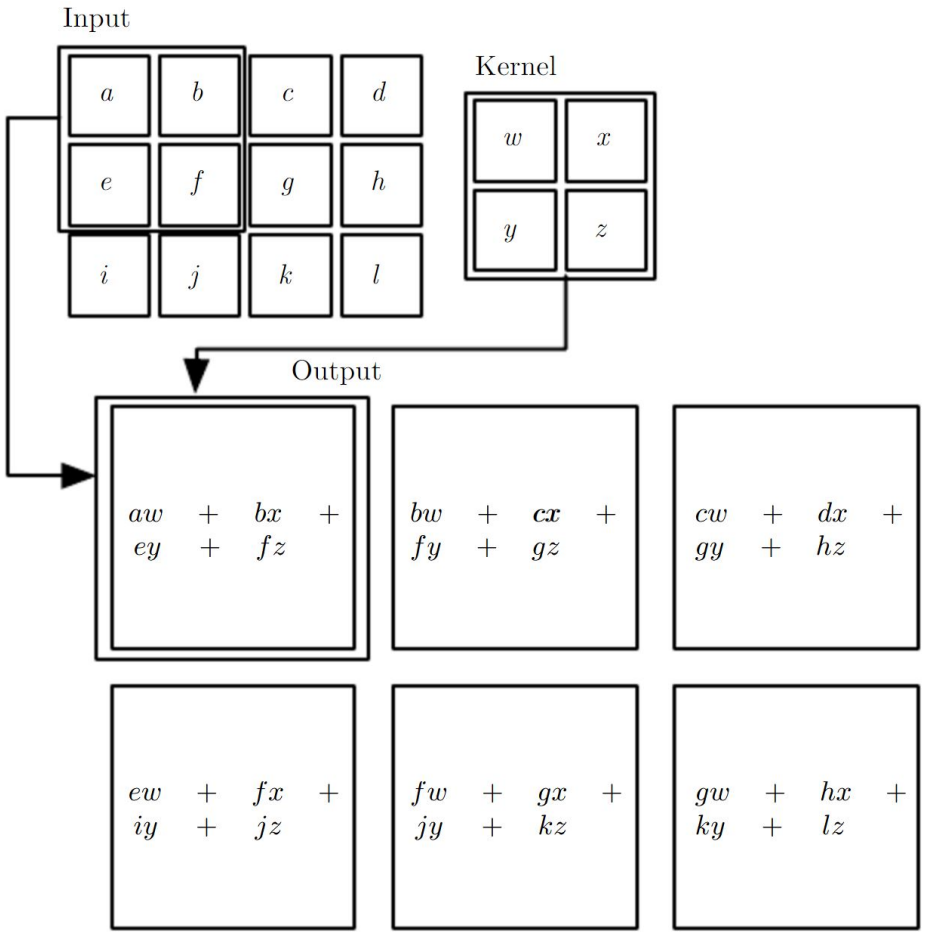
- ⋆ many NN libraries implement cross-correlation
  - ▶ but call it convolution (without kernel flipping)

Example

⋆ convolution applied to a 2D tensor (without kernel flipping)



source: [Angermueller et al., 2016]

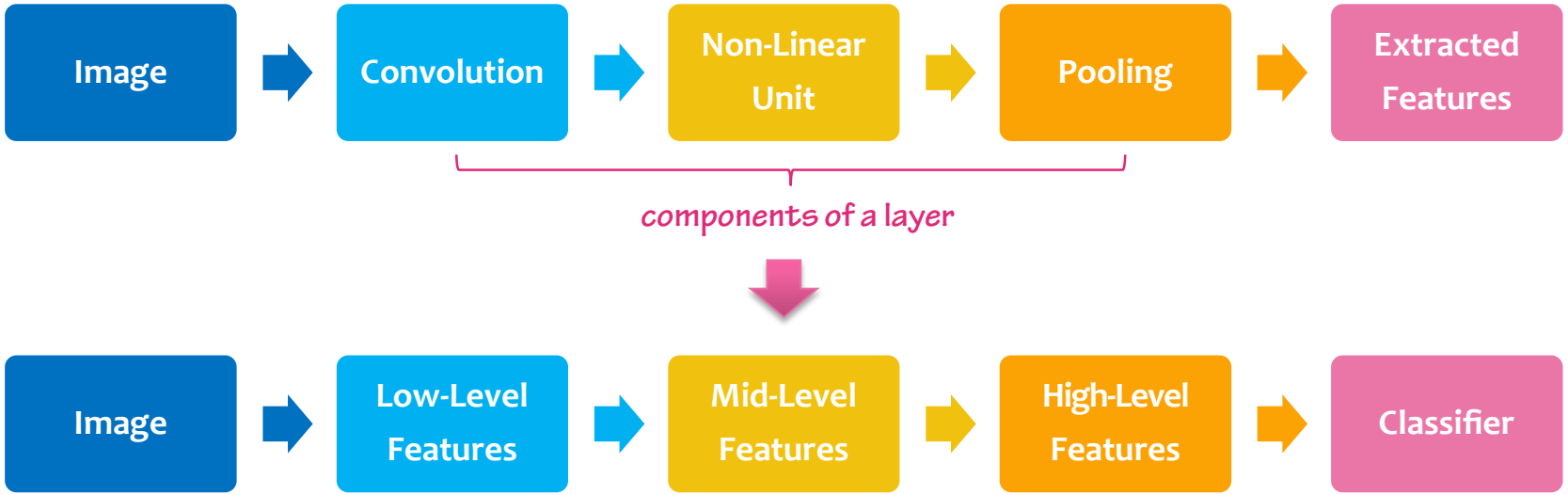


source: [Goodfellow et al., 2016]

Big picture of CNN architecture

⋆ best architectures consistently used multiple stages of

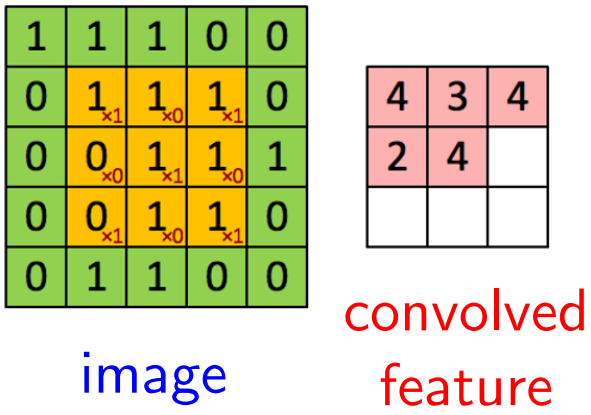
- convolution → non-linearity → pooling

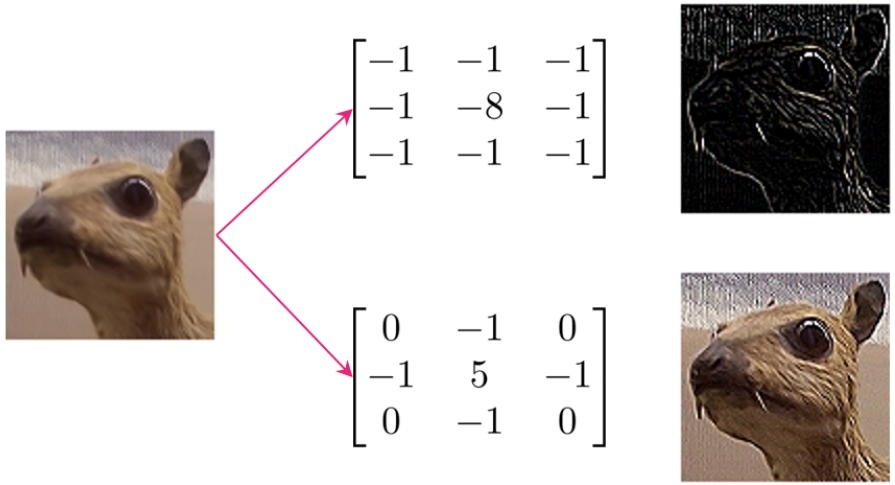


- convolution: feature extraction, filtering, ...
- non-linearity: sparsification, saturation, lateral inhibition, ...
- pooling: aggegation over space or feature type, ...

Convolutional stage

⋆ input image is convolved with a set of filters, giving feature maps





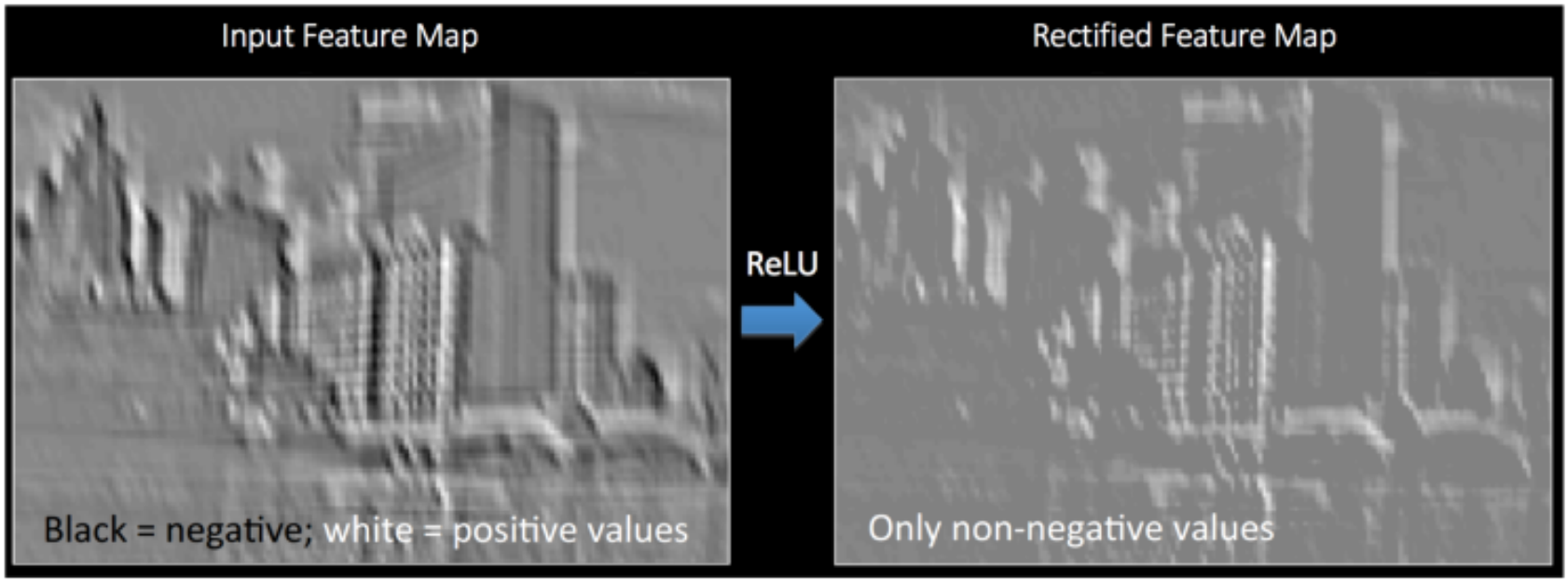
source: www.wikipedia.org

⋆ filter (aka kernel or convolution matrix)

- different filters extract different features (*e.g.*, edges)
- filter weights: trained (by backprop) or pre-determined
- weight sharing & local connectivity

Detector (nonlinear) stage

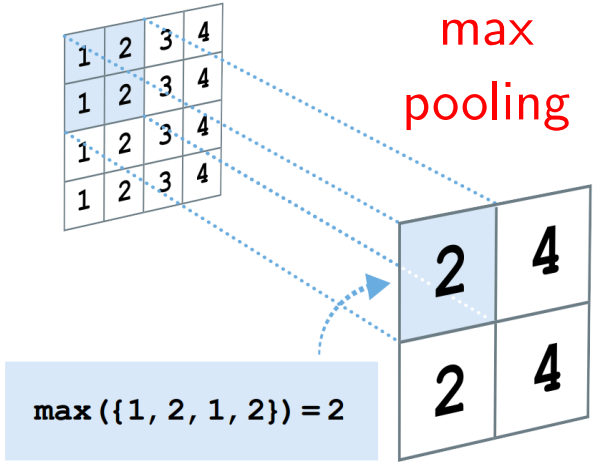
- ⋆ nonlinear activation
  - increases the nonlinear properties of the decision function and of the overall network
  - most popular: ReLU (rectifier linear units)

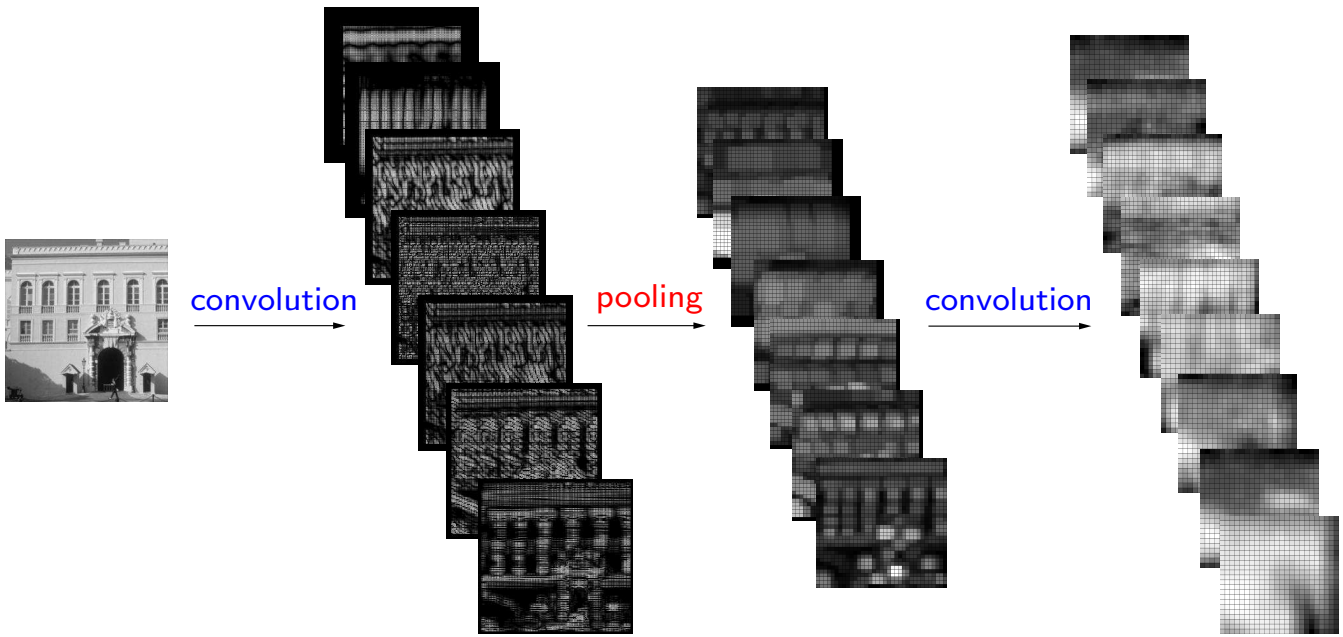


source: [Rob Fergus, 2015]

Pooling/subsampling stage

- ⋆ nonlinear down-sampling
  - aggregates statistics of local features
  - reduced variance: provides invariance to local transformations

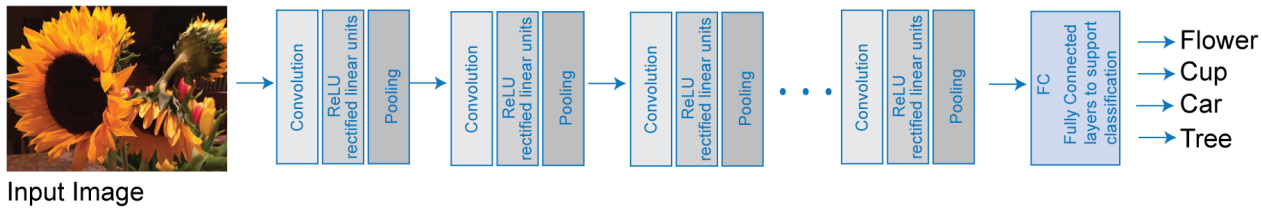


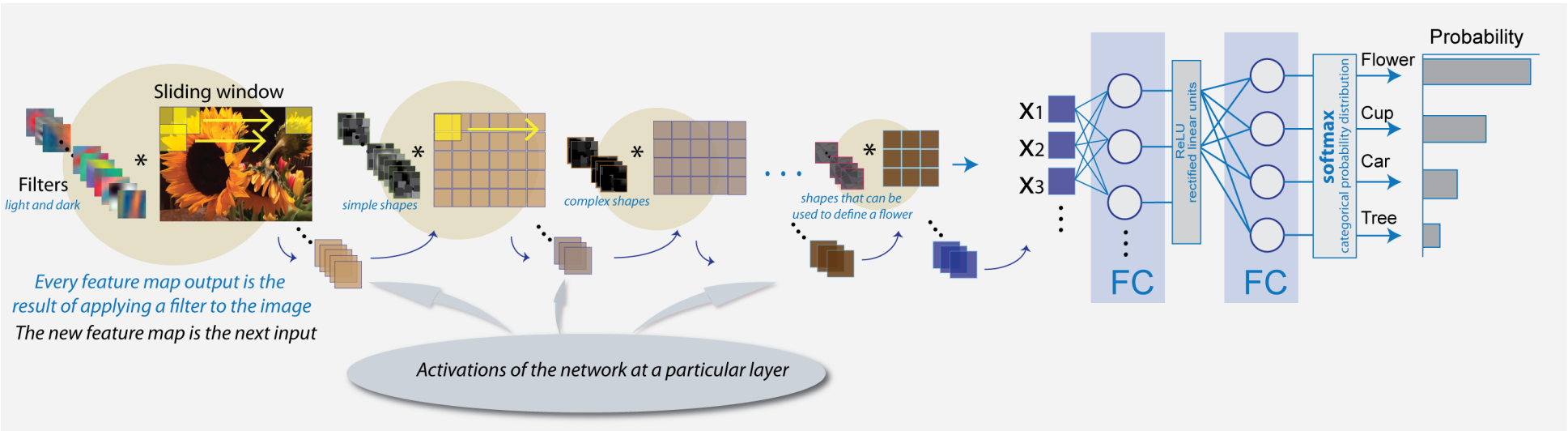


source: [Angermueller et al., 2016, Thériault et al., 2013]

Fully connected (FC) layer

- ⋆ for high-level reasoning such as classification and regression
  - ▶ often with dropout regularization & softmax output

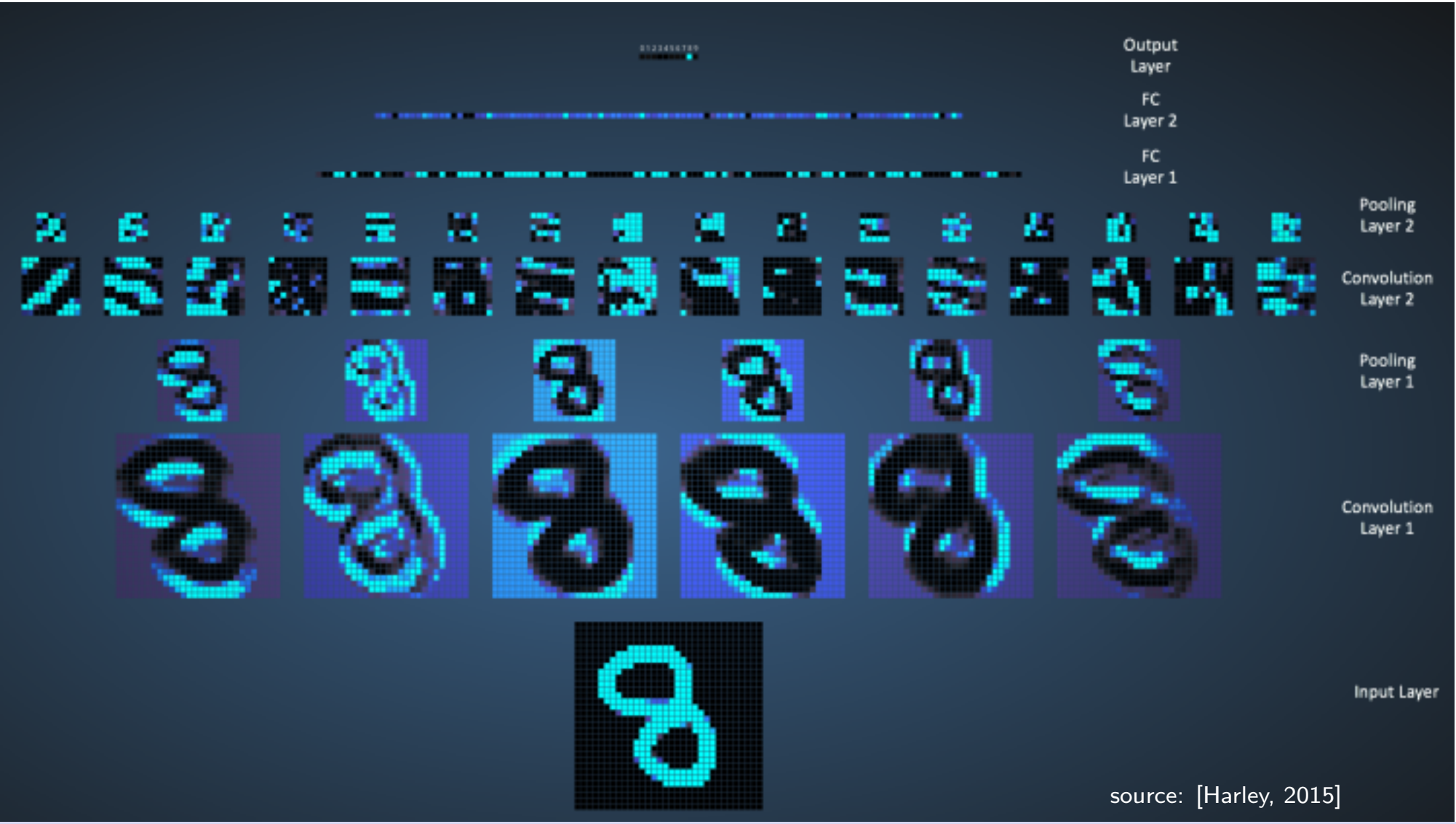




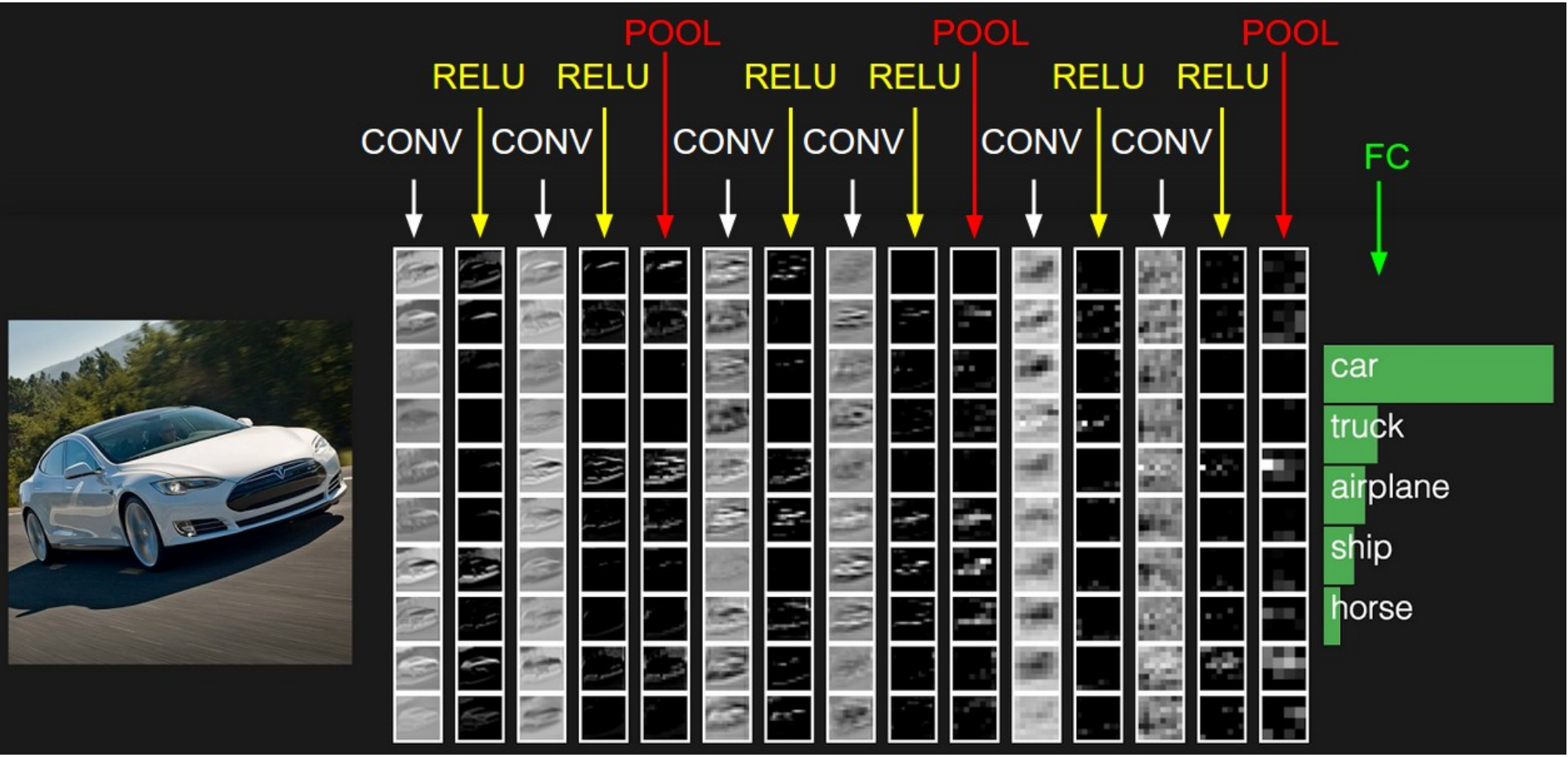
source: www.mathworks.com

Putting it all together

- ⋆ ex: digit classification (ReLU layer not shown)



⋆ ex: CIFAR[4]-10 classification

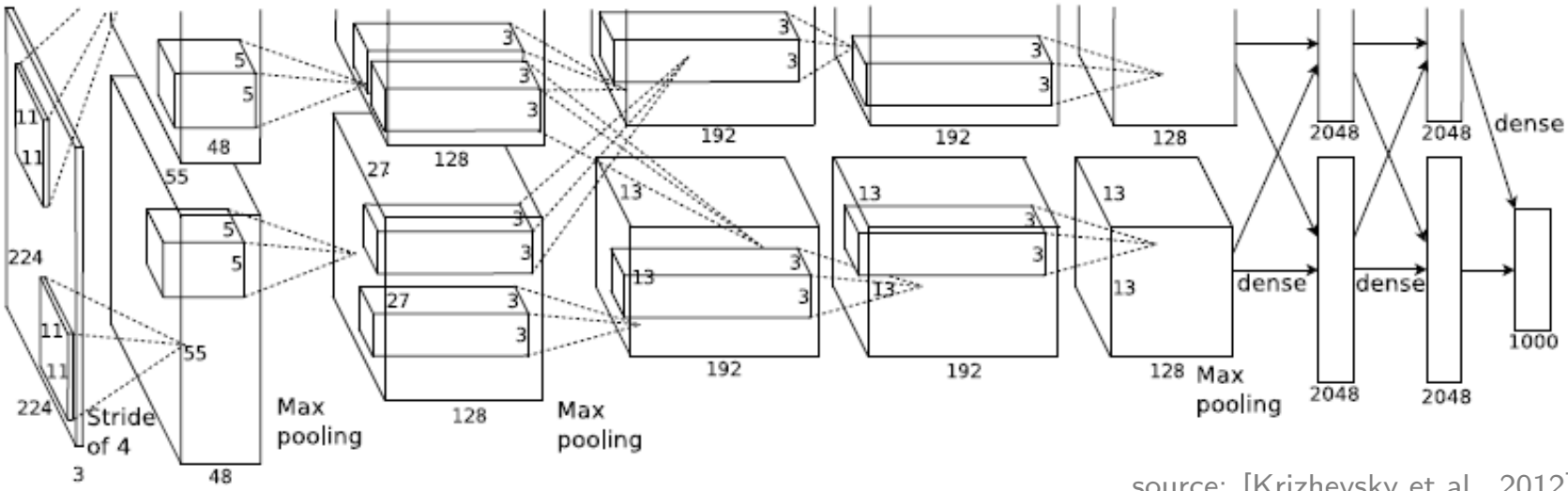


source: Stanford CS231n

[4]Canadian Institute for Advanced Research

More real-world CNN examples

- ⋆ LeNet (LeCun): the first successful applications of CNN
  - successfully used to read zip codes/digits
- ⋆ AlexNet (Krizhevsky, Sutskever and Hinton): ILSVRC[5] 2012 winner
  - the first work that popularized CNN in computer vision
  - had a similar architecture basic as LeNet, but was deeper, bigger

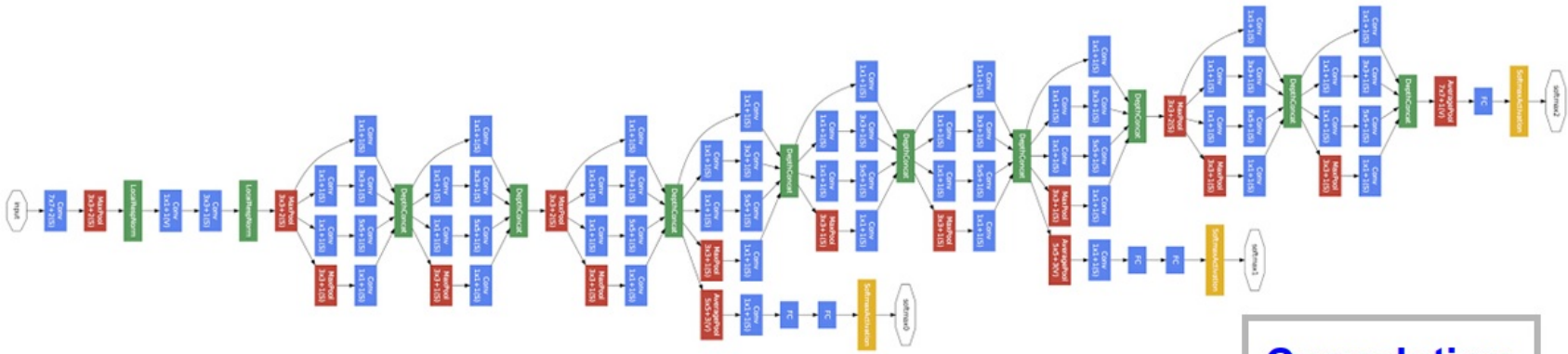


source: [Krizhevsky et al., 2012]

[5] ImageNet Large Scale Visual Recognition Challenge ▸ Link

- ZF Net (Zeiler and Fergus): ILSVRC 2013 winner
  - an improvement on AlexNet by tweaking the architecture
  - in particular by expanding the size of middle convolutional layers

- GoogLeNet (Szegedy et al.): ILSVRC 2014 winner
  - has an Inception Module that dramatically reduced the number of parameters in the network (4M, compared to AlexNet with 60M)



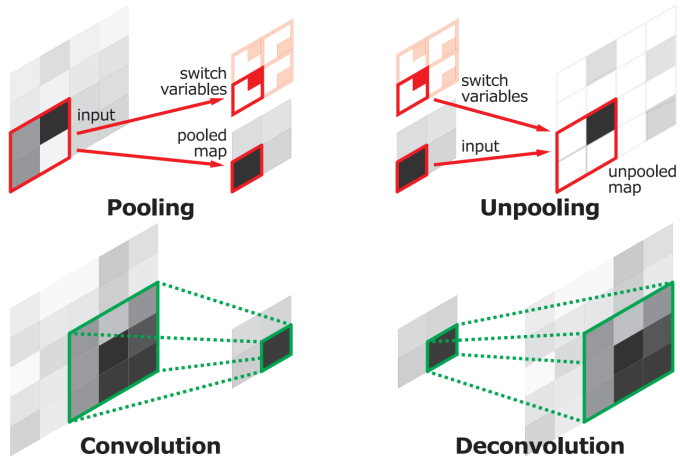
source: Google

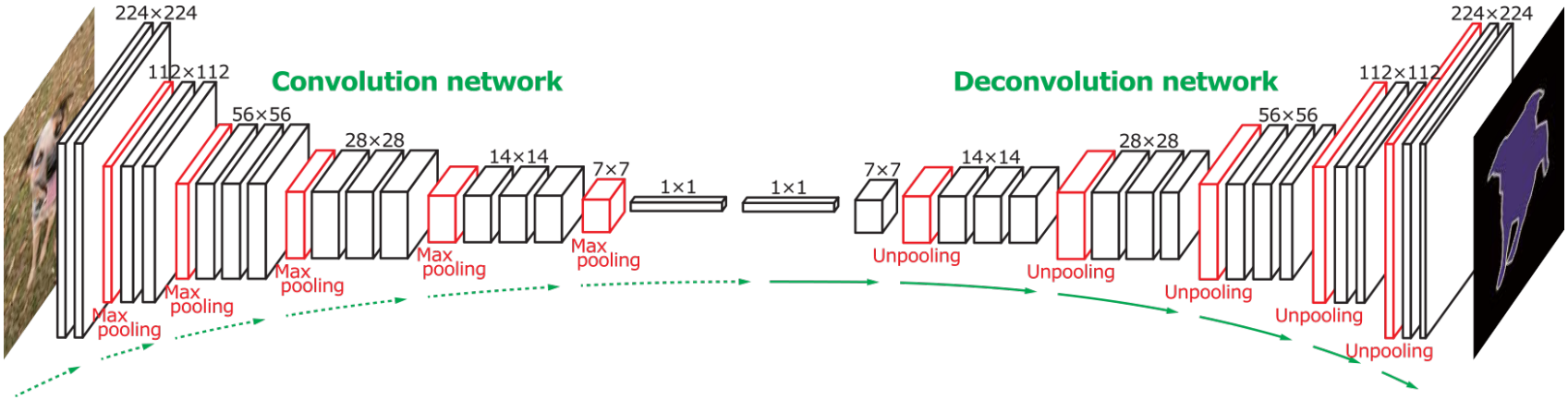
- ⋆ VGGNet (Simonyan and Zisserman): runner-up in ILSVRC 2014
  - features an extremely homogeneous architecture that only performs 3x3 convolutions and 2x2 pooling from the beginning to the end
  - VGGNet features outperform those of GoogLeNet in transfer learning tasks
  - currently the most preferred choice in the community when extracting CNN features from images
  - a pre-trained model is available for plug and play use in Caffe
  - drawbacks: more expensive to evaluate and uses a lot more memory and parameters (140M)

- ⋆ deep residual network (ResNet): ILSVRC & COCO 2015 winner
  - He, Zhang, Ren & Sun (Microsoft Research Asia)
  - 1st places in all 5 main tracks

Additional CNN architectures

- ⋆ deconvolution networks [Zeiler et al., 2010, Noh et al., 2015]
  - permits unsupervised construction of hierarchical image representations
  - useful for denoising, object recognition, and semantic segmentation

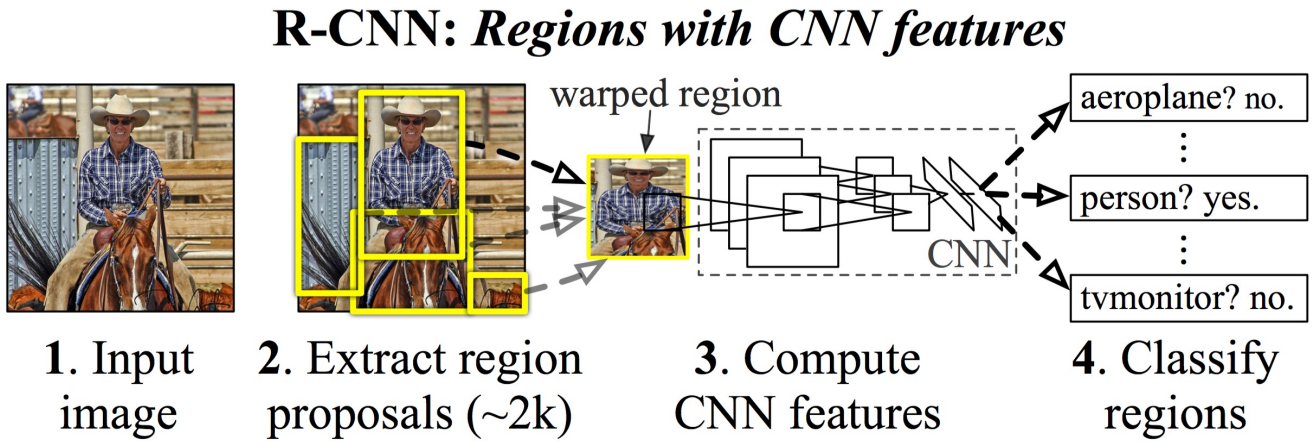




source: [Noh et al., 2015]

⋆ region-based CNN (R-CNN) [Girshick et al., 2014]

- uses CNN as feature extractor for object detection
- object detection: region proposals (slow) + CNN features



source: [Girshick et al., 2014]

⋆ upgrades from R-CNN

- Fast R-CNN (joint learning of steps 3 and 4) [Girshick, 2015]
- Faster R-CNN (use of region proposal networks) [Ren et al., 2015]

CNN summary

- ⋆ convolution (neural) networks (CNNs)
  - neural networks with convolution operations
  - specialized for grid topology (*e.g.*, images and time series)
  - tremendous commercial success (intense interest by industry)
- ⋆ fundamental principles & properties
  - sparse interactions, parameter sharing $\Rightarrow$ efficiency
  - equivariance (convolution), invariance (pooling)
- ⋆ typical CNN architecture & training
  - CNN layer(s): convolution, detector, and pooling stages
  - fully connected layer(s) near output
  - training by backpropagation

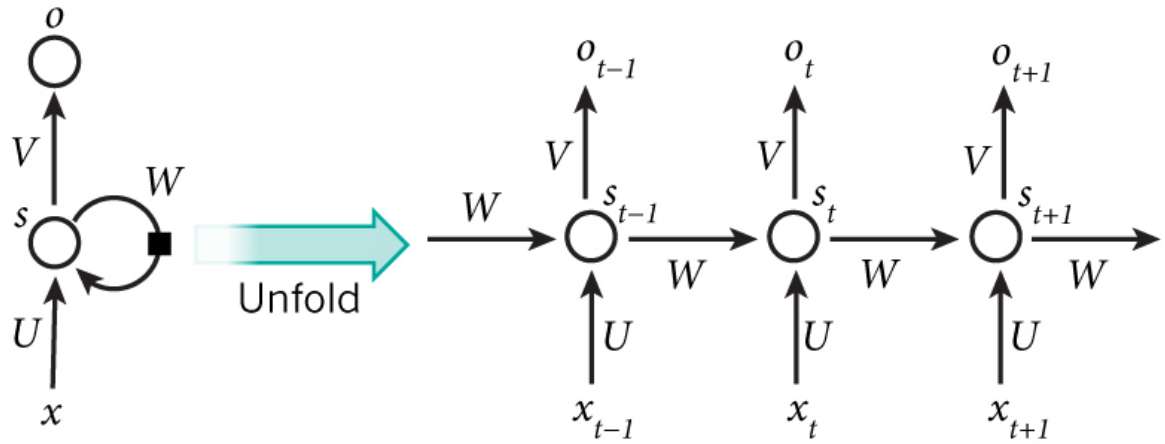
Outline

Deep learning RNNaissance

source: www.youtube.com

Recurrent neural networks (RNNs)

⋆ recurrence (Markov connection) between hidden units



source: [LeCun et al., 2015a]

⋆ RNNs process an input sequence one element at a time

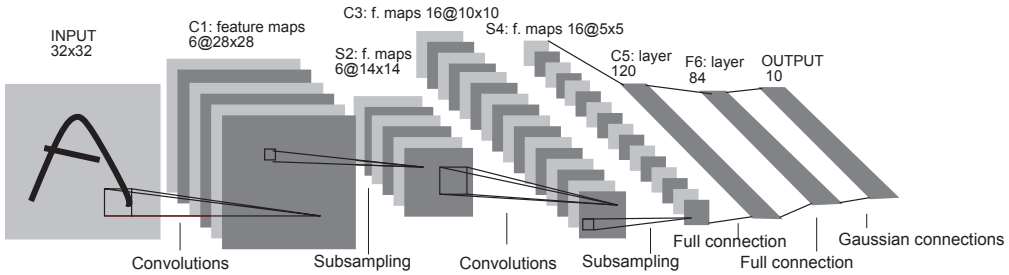
- maintain a 'state vector' in their hidden units
- the vector implicitly contains the history of the sequence

⋆ RNNs are better for tasks that involve sequential inputs

- such as speech, language, and genome

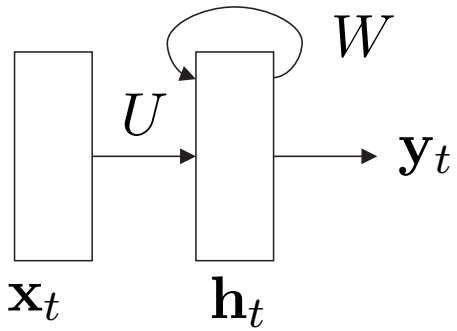
Parameter sharing in neural networks

- ⋆ convolutional neural networks
  - parameter sharing gives robustness and efficiency



source: [LeCun et al., 2015b]

- ⋆ recurrent neural networks
  - the same weights are used at different time steps
  - this allows much better generalization properties



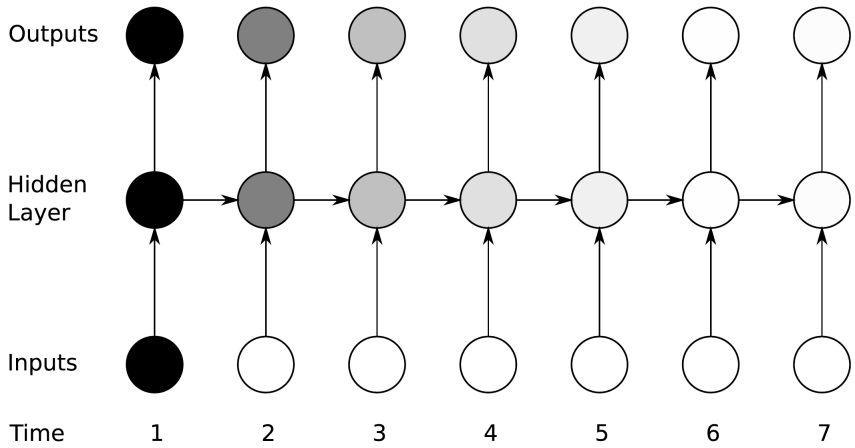
A brief history of RNN

- ⋆ RNNs are very powerful dynamic systems but
  - training them has proved to be problematic
- ⋆ 1980's: gradients-based fundamental methods
  - backpropagation through time (BPTT)
  - real-time recurrent learning (RTRL)
- ⋆ 1990's: proposals to augment RNNs with memory
  - long short-term memory (LSTM) networks [Hochreiter and Schmidhuber, 1997]
- ⋆ more recent approaches
  - gated recurrent units (GRU) [Cho et al., 2014a]
  - neural Turing machines [Graves et al., 2014]
  - memory networks [Weston et al., 2014]

Challenges in RNN training

- ⋆ backpropagated gradients either grow or shrink at each time step
  - ▶ so over many time steps they typically explode or vanish
- ⋆ the vanishing gradient problem
  - ▶ when the weights in a weight matrix are small (*i.e.*, the leading eigenvalue of the weight matrix is smaller than 1.0)
  - ▶ learning either becomes very slow or stops working altogether
  - ▶ learning long-term dependencies in the data becomes hard
- ⋆ the exploding gradient problem
  - ▶ when the weights in the matrix are large (*i.e.*, the leading eigenvalue of the weight matrix is larger than 1.0)
  - ▶ large gradient signals can cause learning to diverge

- ⋆ vanishing grad. makes learning long-term dependency problematic
  - the error information quickly disappears during backprop and cannot inform most previous steps of the error



source: [Graves, 2012]

- ⋆ NLP example: predicting the next word
  - (a) `Jane walked into the room. John walked in too. Jane said hi to _____.`
  - (b) `Jane walked into the room. John walked in too. It was late in the day, and everyone was walking home after a long day at work. Jane said hi to _____.`

source: Stanford CS224d

A solution: use of explicit memory

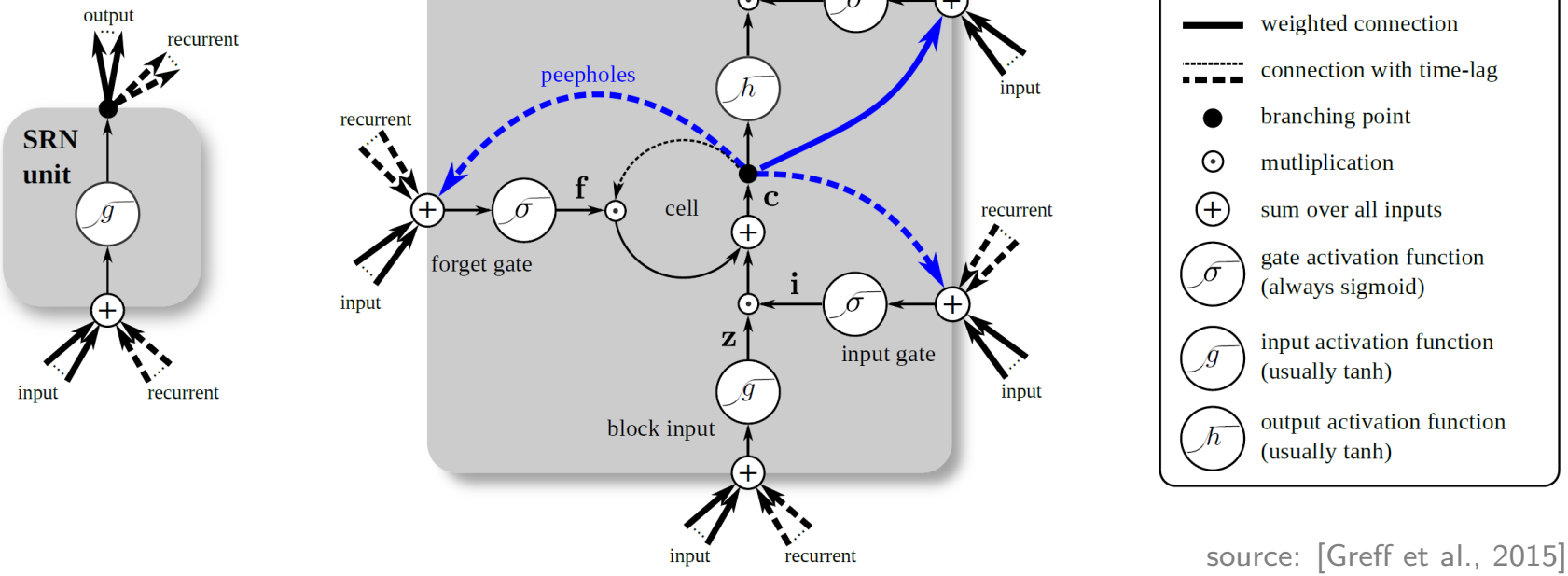
- ⋆ RNNs have been found to perform better with
  - more complex ("gated") activation units:
  - LSTM (long short-term memory)
  - GRU (gated recurrent unit)

- ⋆ LSTM networks: the first proposal of this kind
  - more effective than conventional RNNs
  - especially when they have several layers for each time step
  - *en vogue* default model for sequence modeling

- ⋆ the main difference of GRU with LSTM
  - in GRU, a single gating unit simultaneously controls the forgetting factor and the decision to update the state unit

Long short-term memory (LSTM) networks

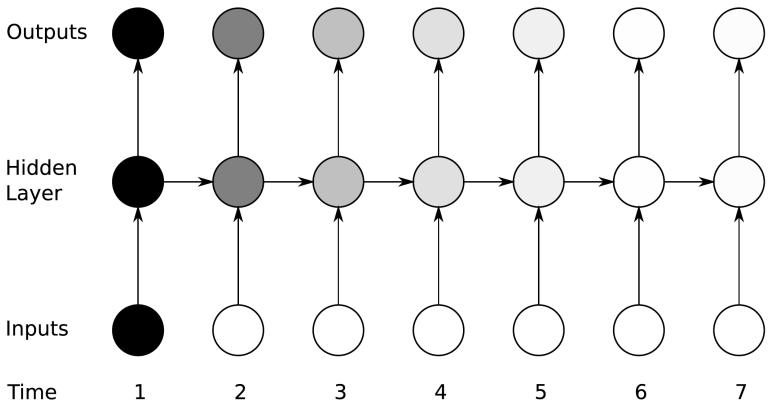
- ⋆ use special hidden units called the *memory cells*
  - memory cells can keep information intact unless inputs make them forget it or overwrite it with new input
  - cell can decide to output this information or just store it



source: [Greff et al., 2015]

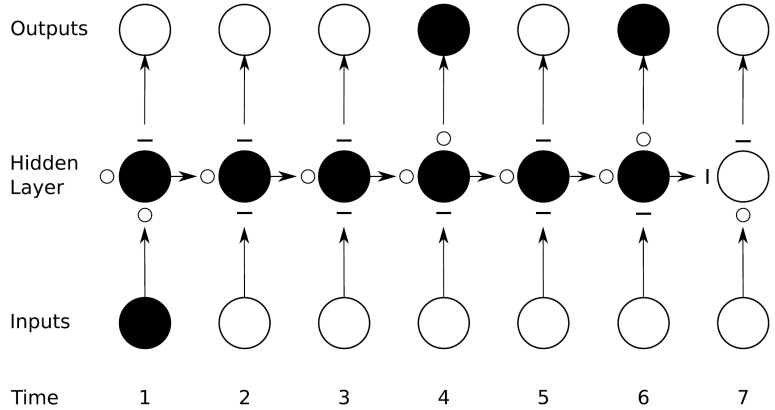
Comparison

⋆ the vanishing gradient problem for RNNs



The shading of the nodes in the unfolded network indicates their sensitivity to the inputs at time one (the darker the shade, the greater the sensitivity). The sensitivity decays over time as new inputs overwrite the activations of the hidden layer, and the network forgets the first inputs.

⋆ preservation of gradient information by LSTM

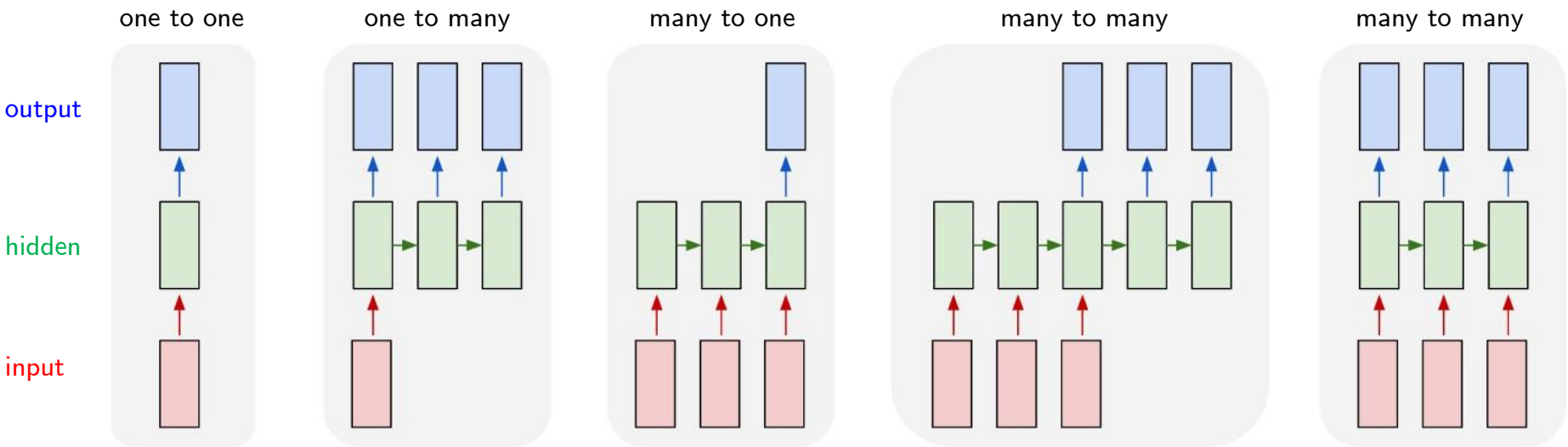


The black nodes are maximally sensitive and the white nodes are entirely insensitive. The state of the input, forget, and output gates are displayed below, to the left and above the hidden layer respectively. For simplicity, all gates are either entirely open (0) or closed (−). The memory cell 'remembers' the first input as long as the forget gate is open and the input gate is closed. The sensitivity of the output layer can be switched on and off by the output gate without affecting the cell.

source: [Graves, 2012]

RNN: a powerful sequence modeling tool

⋆ various sequencing modeling tasks [Goodfellow et al., 2016]



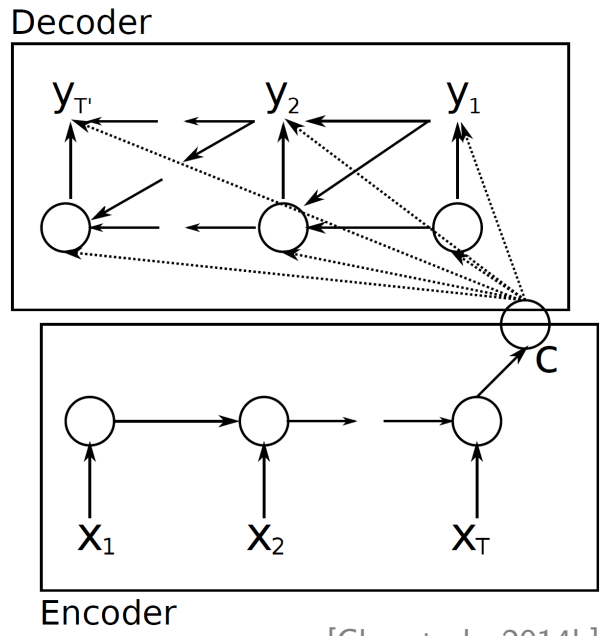
source: [Karpathy, 2015]

Encoder-decoder architecture

- ⋆ the architecture consists of two RNNs
  - ▶ that act as an encoder and a decoder pair [Cho et al., 2014b]

- ⋆ the encoder maps
  - ▶ a variable-length source seq to a fixed-length vector

- ⋆ the decoder maps
  - ▶ the vector-representation back to a variable-length target seq

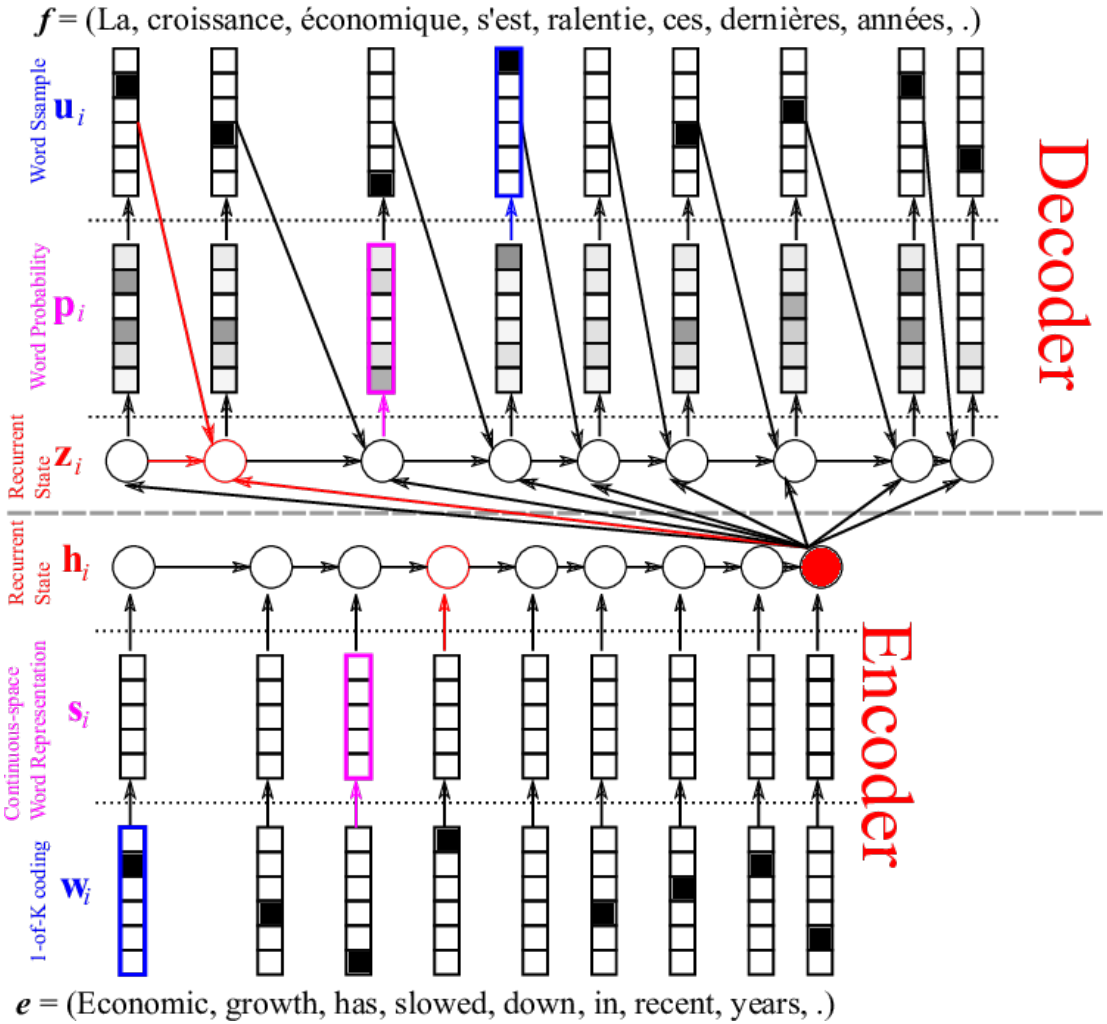


source: [Cho et al., 2014b]

- ⋆ the two networks are trained jointly to maximize
  - ▶ the conditional prob of the target seq given a source seq

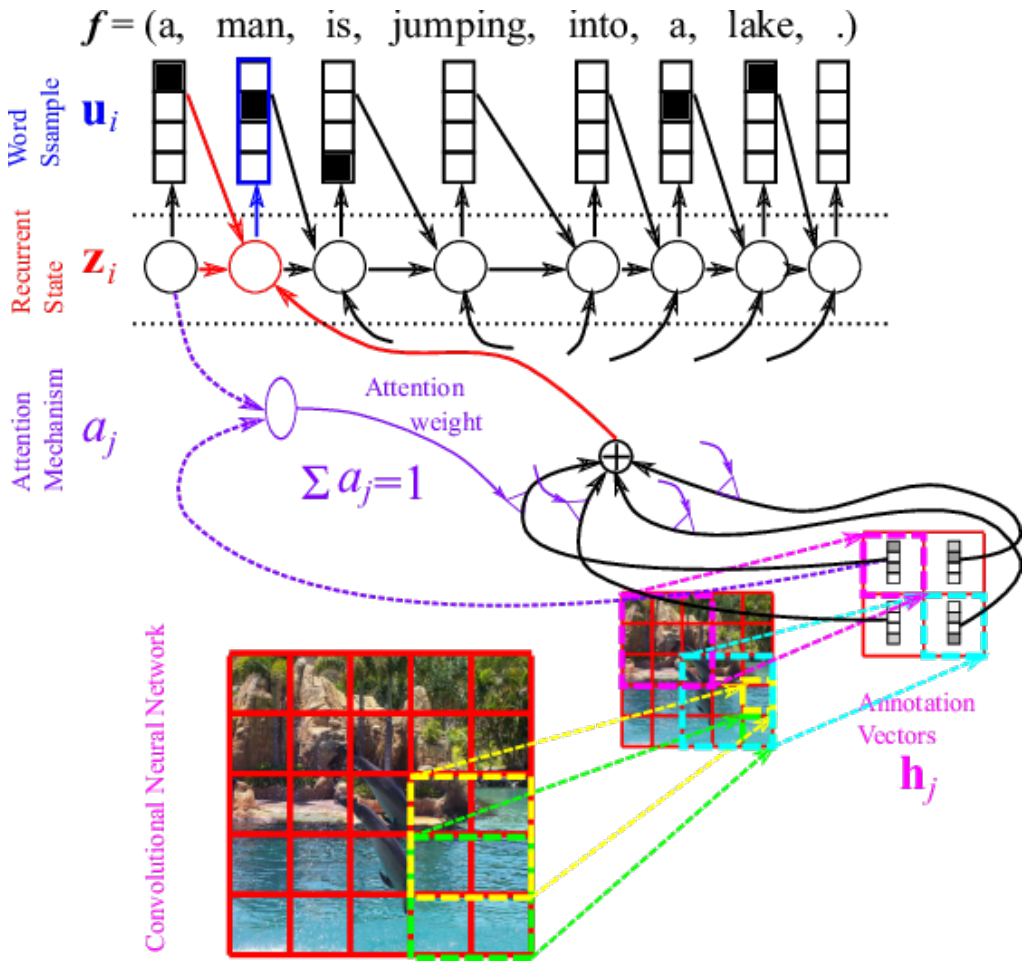
Examples

⋆ neural machine translation [Bahdanau et al., 2014, Cho et al., 2014a]



source: devblogs.nvidia.com

⋆ caption generation with attention mechanism



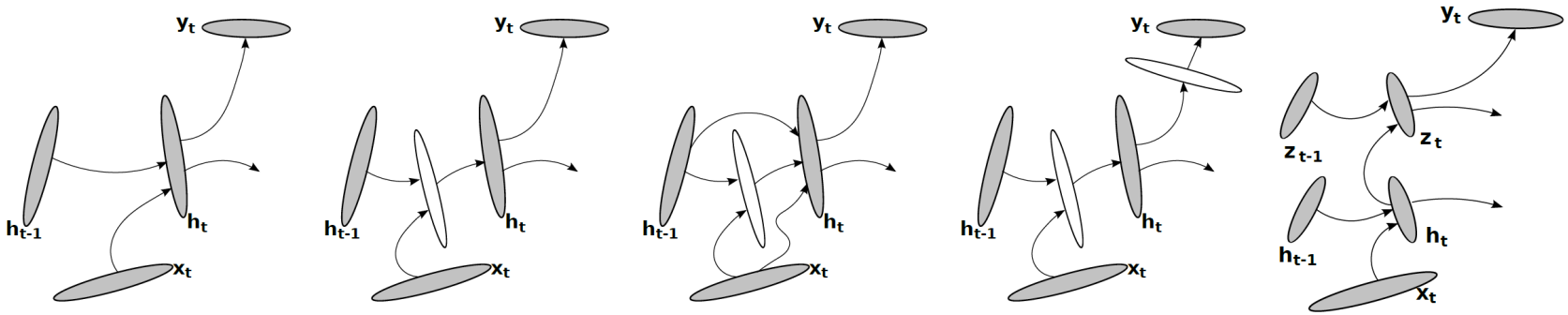
source: devblogs.nvidia.com

⋆ encoder
- CNN features

⋆ decoder
- LSTM network

Constructing deep RNN architectures

- ⋆ [Pascanu et al., 2013]



(a) RNN (b) DT-RNN (b*) DT(S)-RNN (c) DOT-RNN (d) Stacked RNN

Figure 2: Illustrations of four different recurrent neural networks (RNN). (a) A conventional RNN. (b) Deep Transition (DT) RNN. (b*) DT-RNN with shortcut connections (c) Deep Transition, Deep Output (DOT) RNN. (d) Stacked RNN

source: [Pascanu et al., 2013]

RNN Summary

- ⋆ recurrent neural networks (RNNs)
  - well suited for tasks that involves sequential inputs
  - in theory: can capture long-term dependencies
  - in practice: vanishing/exploding gradient problems
- ⋆ approaches to alleviating training challenges
  - constant error flow, augmentation with memory

  e.g., LSTM, GRU, neural Turing machines, memory networks
- ⋆ a variety of architectures (and their combinations)
  - stacked/deep, bidirectional, multi-dimensional, recursive, hierarchical
- ⋆ known applications of RNNs (many more coming soon)
  - speech/handwriting recognition, acoustic/music modeling
  - bioinformatics, machine translation, caption generation, Q&A systems

Outline

Deep learning in bionformatics

- ⋆ surveys
  - "Deep learning in bioinformatics" [Min et al., 2016]
  - "Deep learning for computational biology" [Angermueller et al., 2016]
  - "Applications of deep learning in biomedicine" [Mamoshina et al., 2016]
  - "Deep learning for regulatory genomics" [Park and Kellis, 2015]
  - "Deep learning for population genetic inference" [Sheehan and Song, 2016]
  - "Deep learning in medical imaging" [Greenspan et al., 2016]
  - "A perspective on deep imaging" [Wang, 2016]
  - "Deep learning for neuroimaging: a validation study" [Plis et al., 2014]

⋆ other relevant resources

- "TensorFlow: biology's gateway to deep learning?" [Rampasek and Goldenberg, 2016]
- "Deep learning meets genome biology" ▸ Link
- "Awesome DeepBio" ▸ Link
- "deeplearning-biology" ▸ Link
- "Deep learning for computational biology/chemistry" ▸ Link

Categorization of existing work

1. omics
2. biomedical imaging
3. biomedical signal processing
4. health informatics

Table 2. Categorization of deep learning applied research in bioinformatics

	Omics		Biomedical imaging		Biomedical signal processing	
	Research topics	Reference	Research topics	Reference	Research topics	Reference
Deep neural networks	Protein structure Gene expression regulation Protein classification Anomaly classification	[84–87] [93–98] [108] [111]	Anomaly classification Segmentation Recognition Brain decoding	[122–124] [133] [142, 143] [149, 150]	Brain decoding Anomaly classification	[158–163] [171–175]
Convolutional neural networks	Gene expression regulation	[99–104]	Anomaly classification Segmentation Recognition	[125–132] [134–140] [144–147]	Brain decoding Anomaly classification	[164–167] [176]
Recurrent neural networks	Protein structure Gene expression regulation Protein classification	[88–90] [105–107] [109, 110]			Brain decoding Anomaly classification	[168] [177, 178]
Emergent architectures	Protein structure	[91, 92]	Segmentation	[141]	Brain decoding	[169, 170]

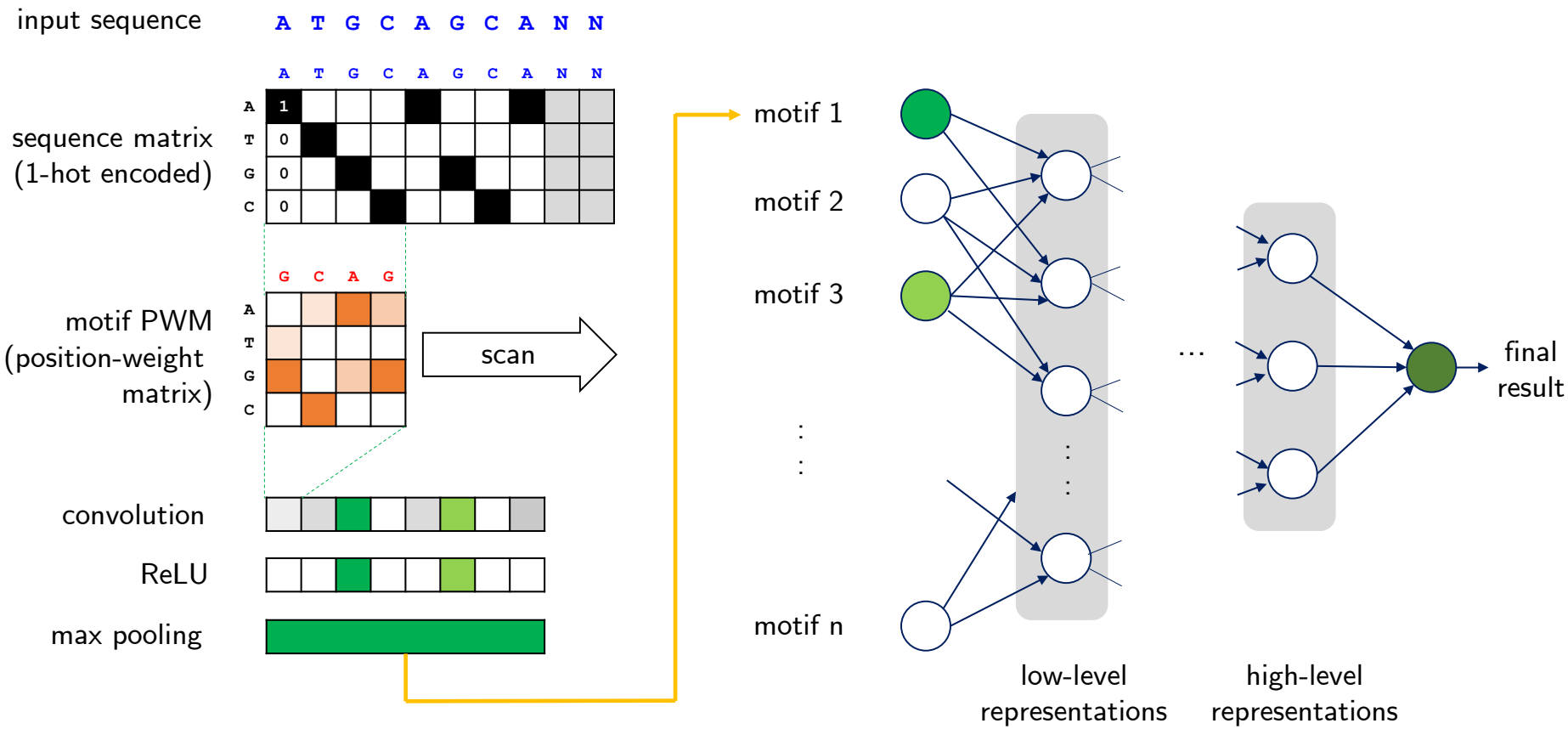
source: [Min et al., 2016]

Outline

Common principles

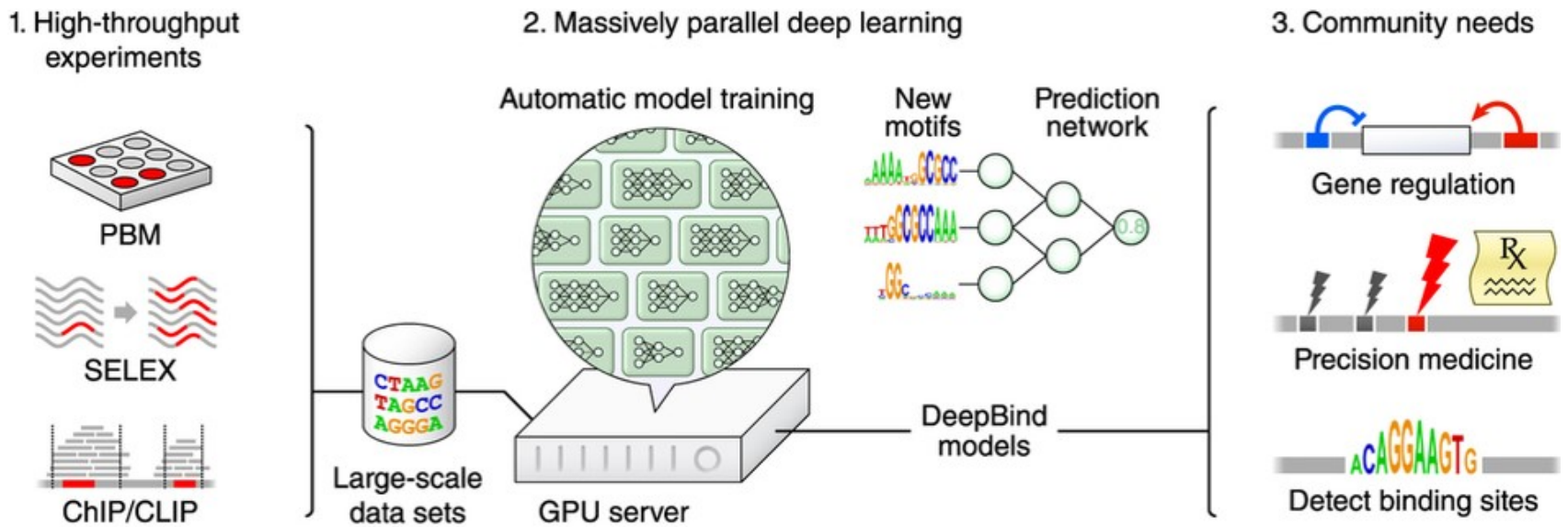
⋆ [Alipanahi et al., 2015, Kelley et al., 2016, Zhou and Troyanskaya, 2015]

- CNN (detection) + DNN (representation & classification)

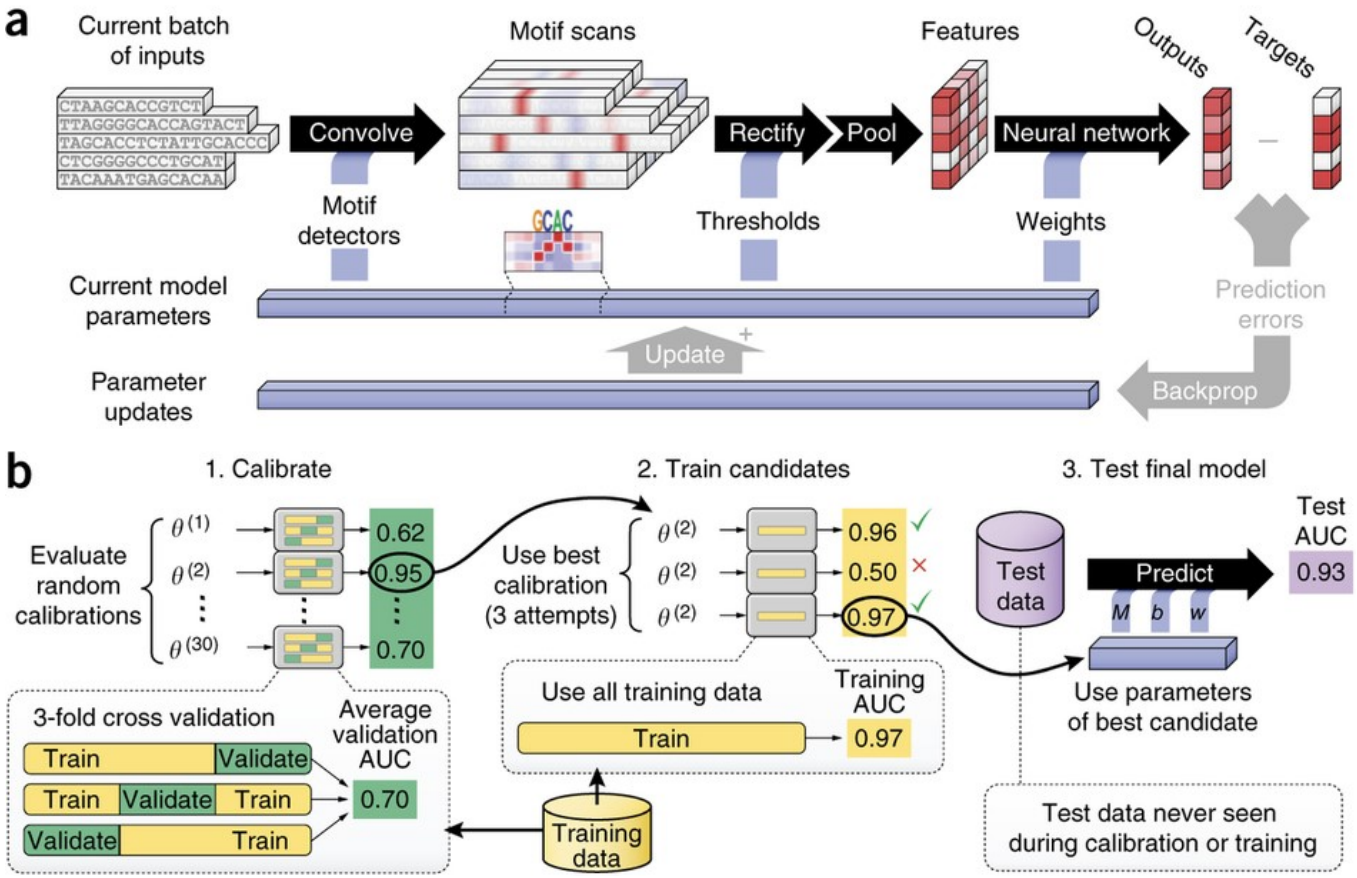


Case #1: DeepBind [Alipanahi et al., 2015]

- ⋆ DNA-/RNA-binding protein prediction
  - can predict sequence specificities of binding proteins
  - was trained on 12 terabases of sequence data from public databases
  - found 927 models for 538 distinct TFs/194 distinct RBPs

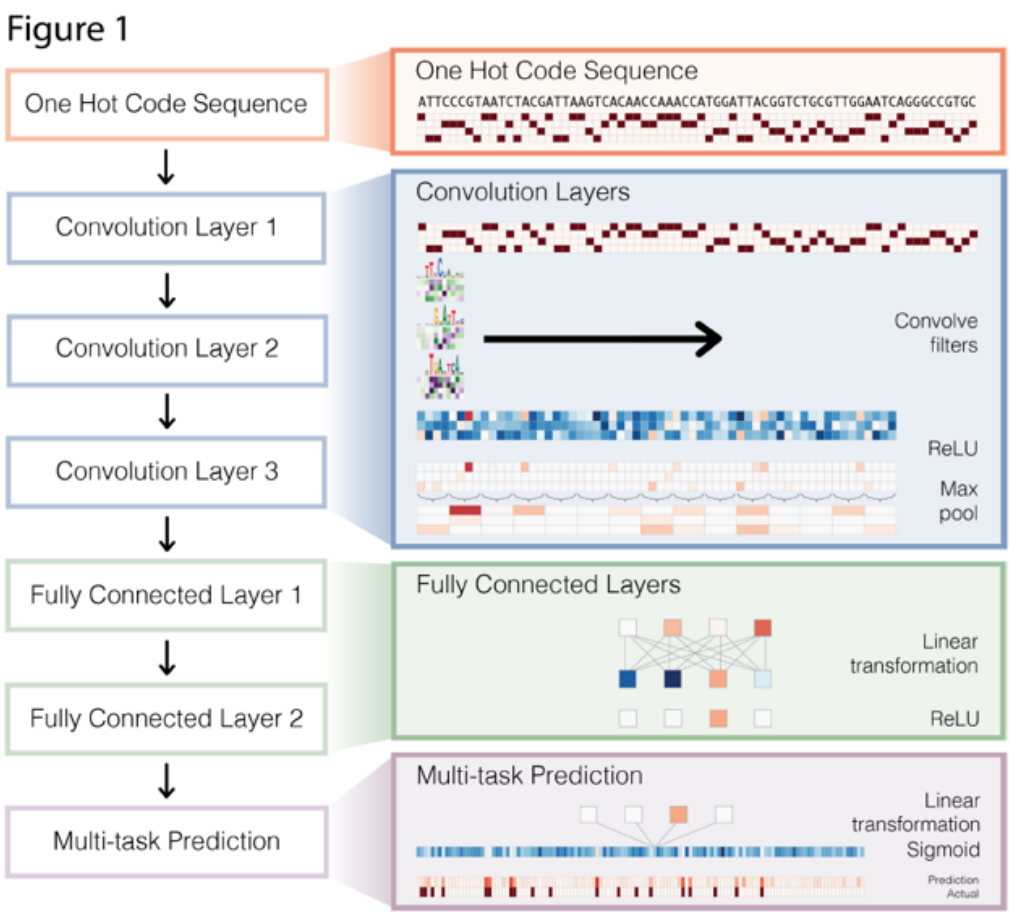


- ⋆ consists of two modules: convolution and prediction module
  - convolution filters of CNN are used as a motif detector
  - prediction module (DNN) synthesizes local features



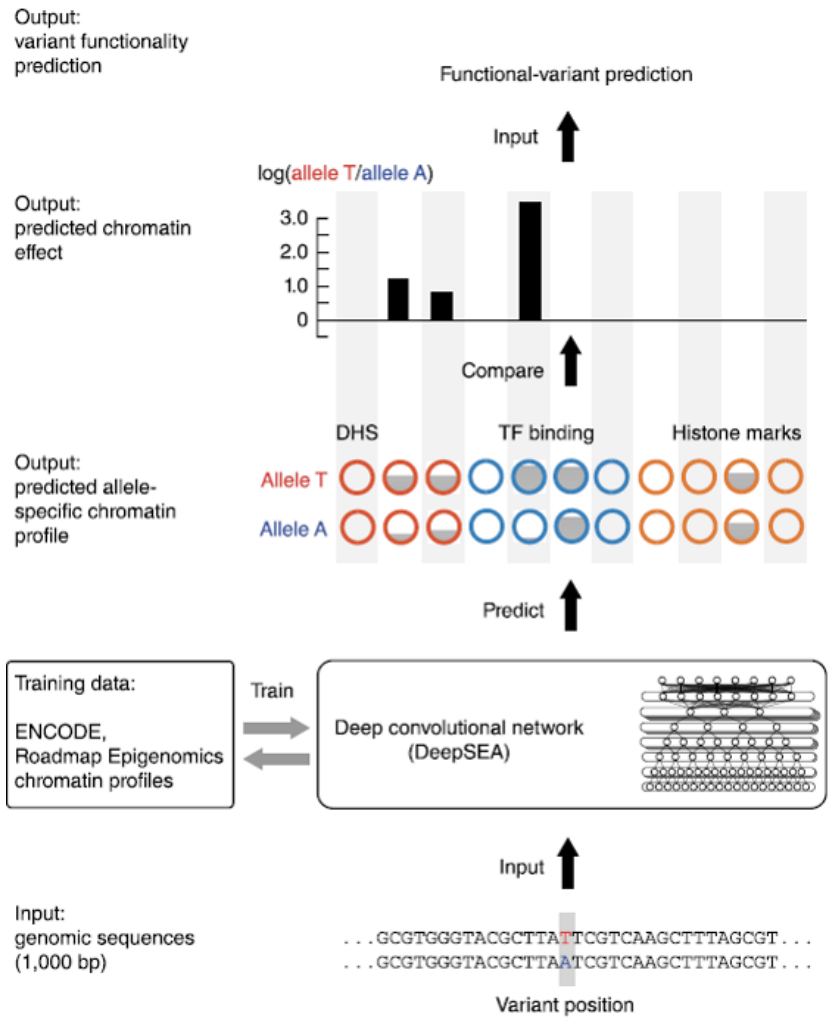
Case #2: Basset [Kelley et al., 2016]

- ⋆ regulatory code learning
  - predicts the change in accessibility between two variant alleles
  - learns chromatin accessibility code and annotates every mutation
  - has multiple CNN and fully connected layers
  - achieves +15% increase in AUC over alternatives



Case #3: DeepSEA [Zhou and Troyanskaya, 2015]

- ⋆ prediction of functional effects of noncoding variants
  - can predict the chromatin effects of sequence alterations (with single nucleotide sensitivity) and prioritize functional noncoding variants
  - exploits the fact that the sequence surrounding a variant position determines the regulatory properties
  - uses CNN to directly learn regulatory sequence code from chromatin-profiling data (TF binding, DNase I sensitivity, and histone-mark profiles)



Case #4: [Zhang et al., 2016]

- ⋆ RNA-binding protein (RBP) binding site prediction
  - predict binding sites based on RNA structure information
  - use multimodal (primary, secondary, tertiary structures) DBN

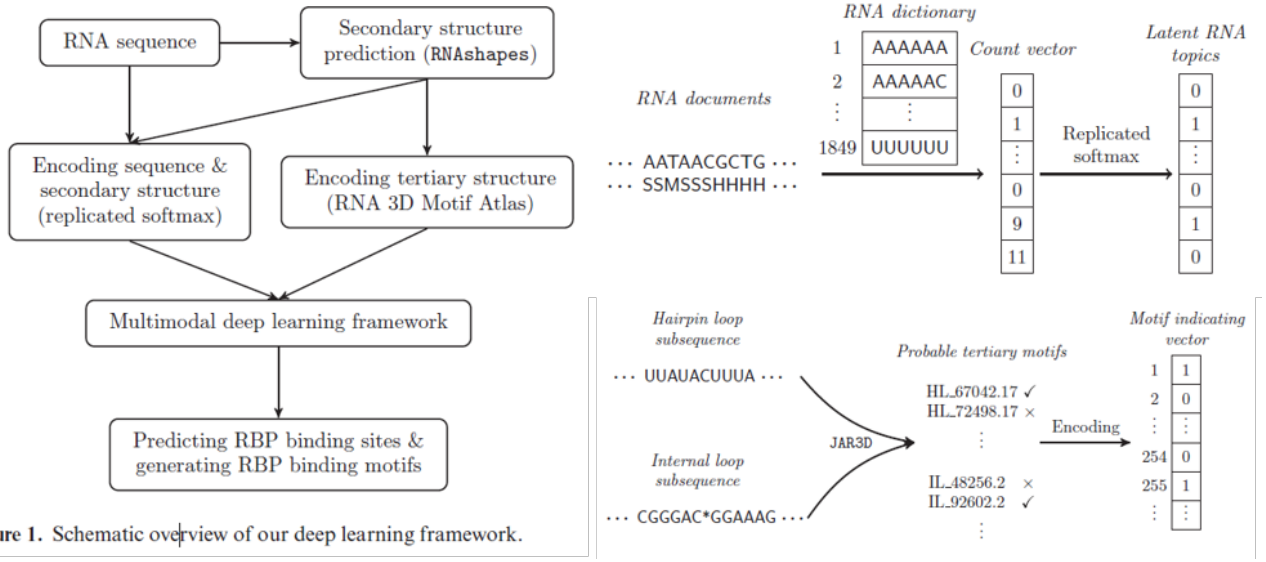
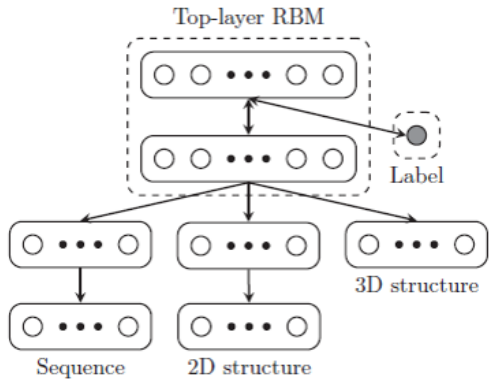


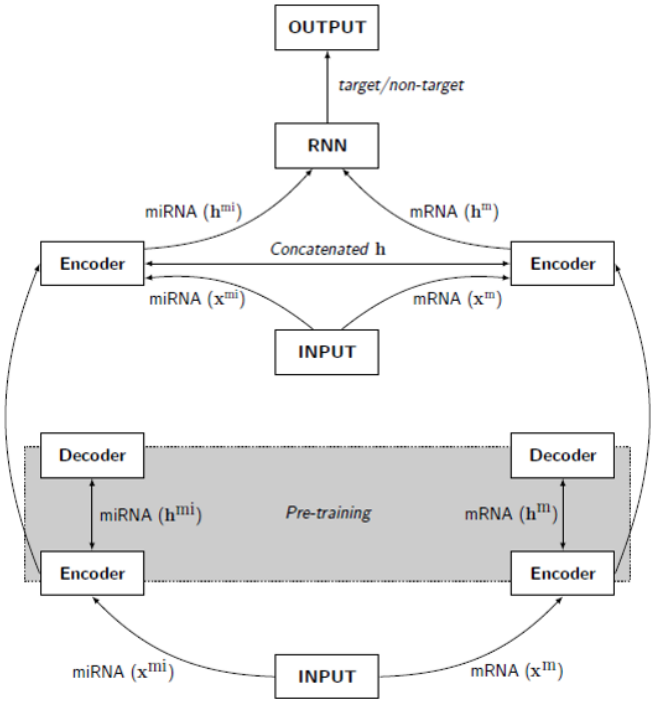
Figure 1. Schematic overview of our deep learning framework.

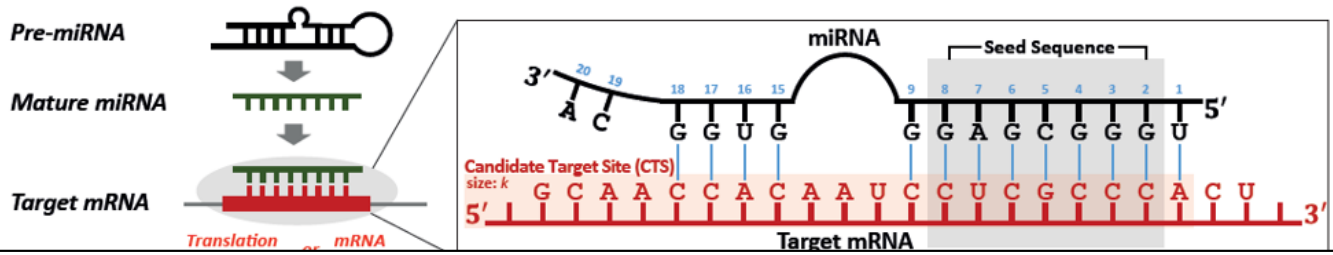


B The multimodal DBN used in our framework

Case #5: deepTarget [Lee et al., 2016]

- ⋆ microRNA target prediction
  - an end-to-end framework
  - autoencoders: unsupervised feature learning
  - stacked RNNs: interaction learning
  - gives +25% increase in F-measure over alternatives



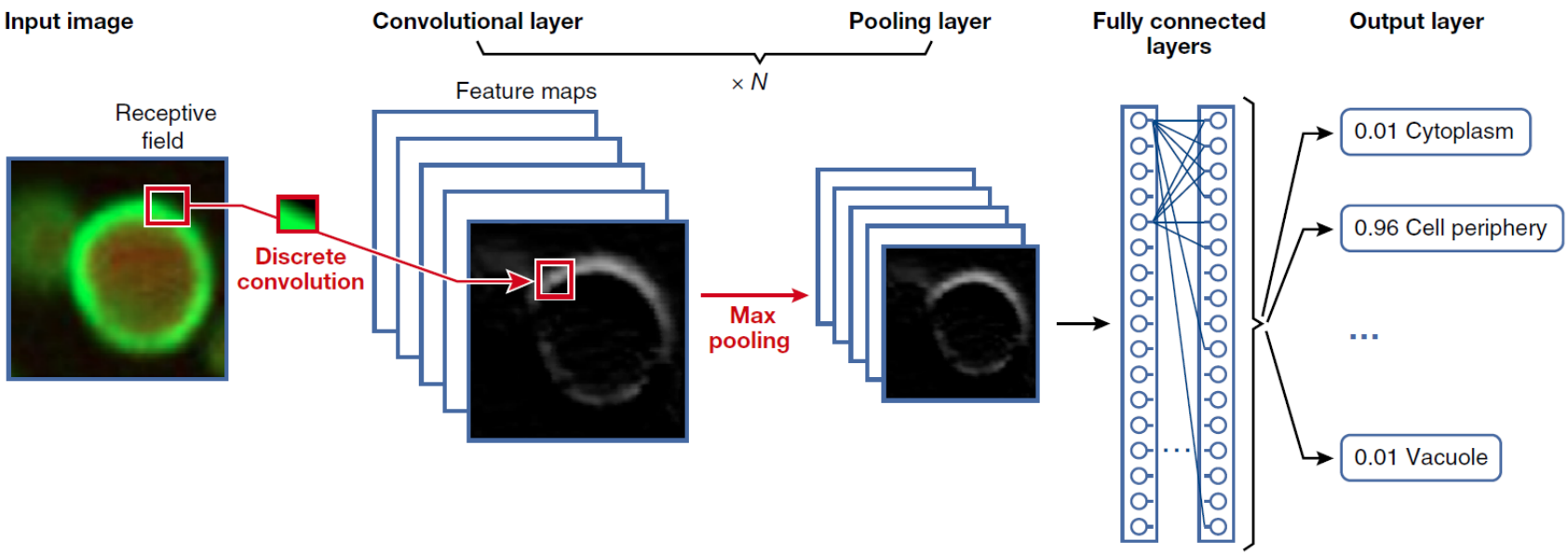


more details: paper 111 (Session 6A, Seattle I, 2pm on 10/5 Wed)

Outline

Basic architecture

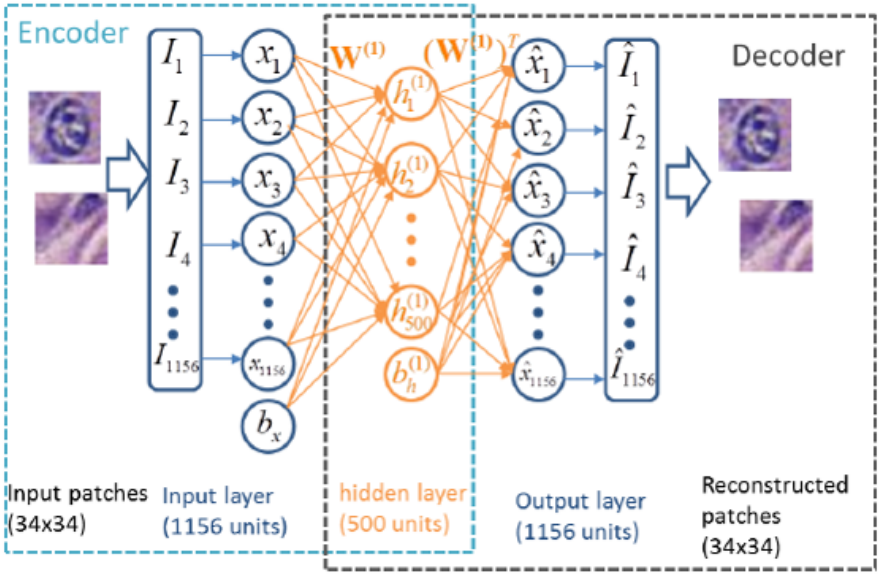
- ⋆ hierarchical representation by successive application of
  - convolution (pattern matching/detection) and
  - pooling (aggregation)
- ⋆ fully connected layer for classification

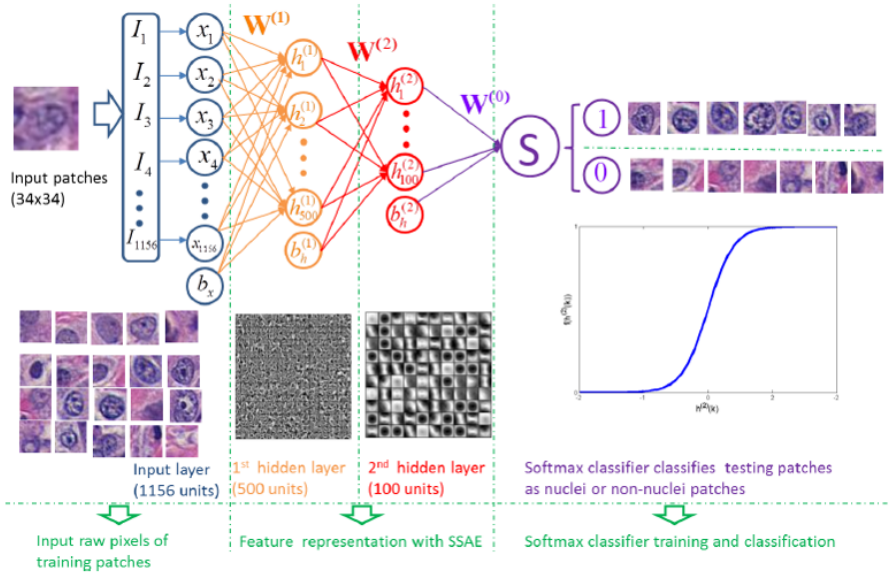


source: [Angermueller et al., 2016]

Case #6: [Xu et al., 2016]

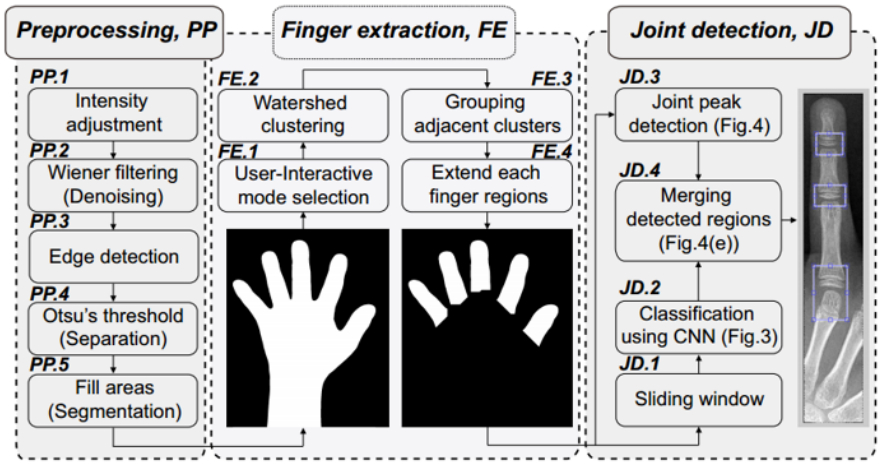
- ⋆ nuclei classification for breast cancer detection
  - feature extraction by stacked sparse autoencoder (SSAE)
  - SSAE enables unsupervised feature learning from raw images

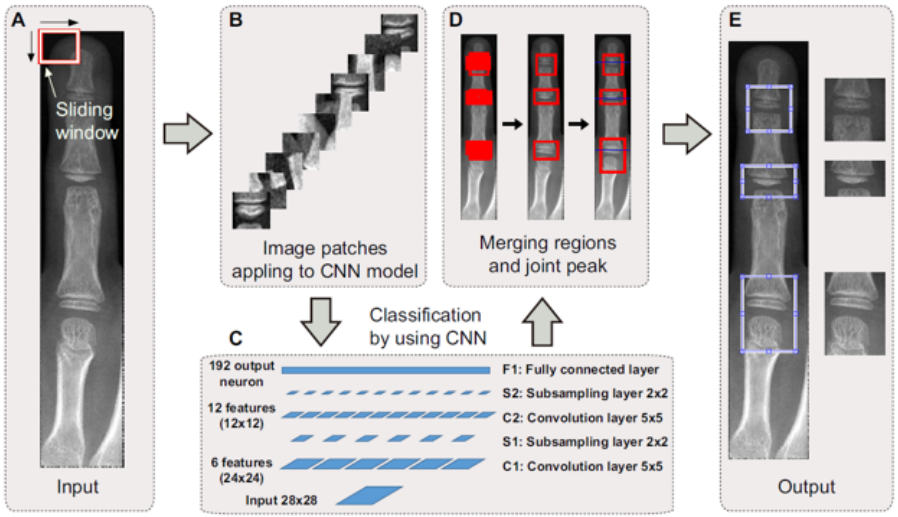




Case #7: FinterNet [Lee et al., 2015]

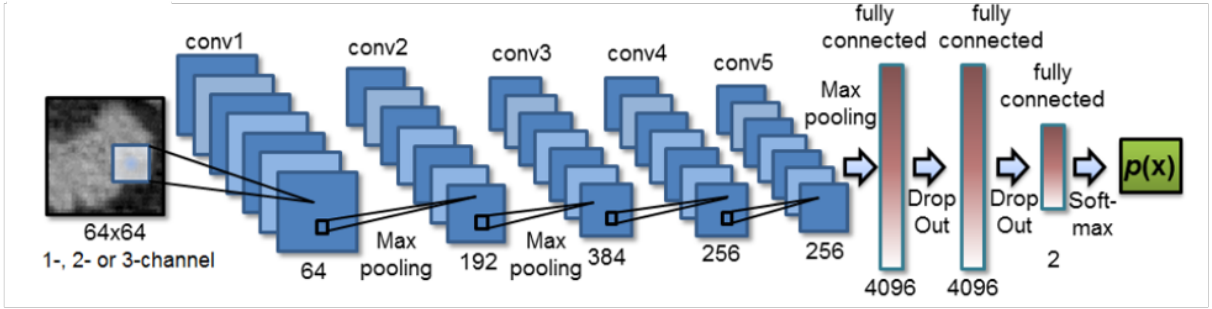
- ⋆ high-throughput finger joint detection
  - feature representation using domain-knowledge
  - additional feature learning and classification by CNN
  - achieved 98.02% accuracy for 130 datasets

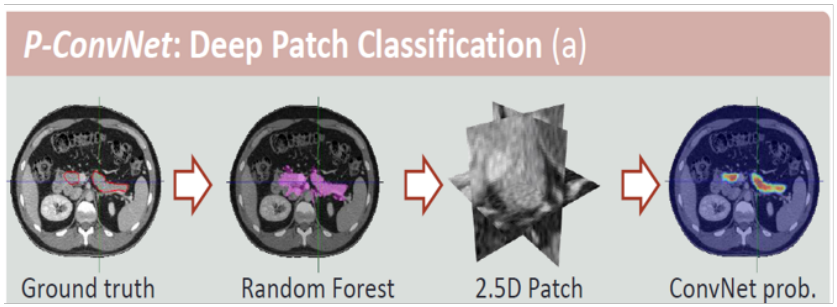


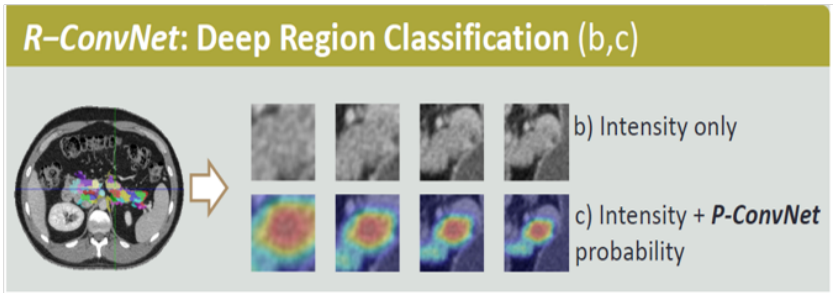


Case #8: DeepOrgan [Roth et al., 2015]

- ⋆ pancreas segmentation
  - CNNs hierarchically classify image patches and regions
  - P-ConvNet for patch classification
  - R-ConvNet for region classification

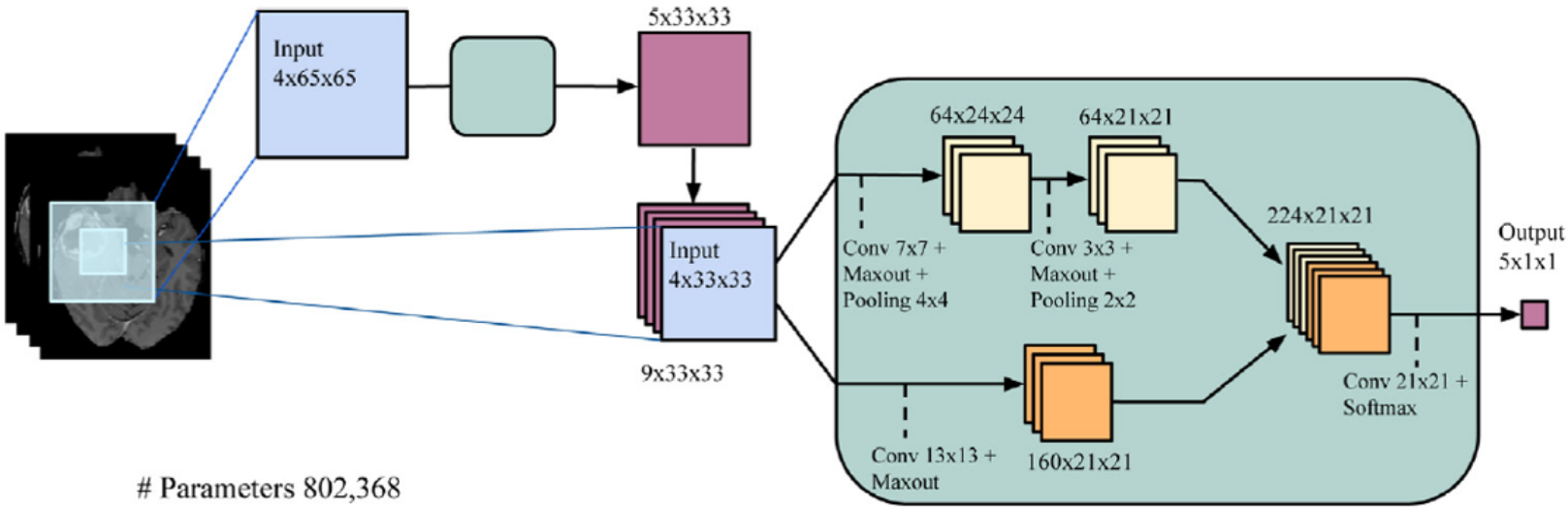






Case #9: [Havaei et al., 2016]

- ⋆ brain tumor segmentation from MR images
  - *two-pathway* CNN: considers local/global info together
  - *cascaded* CNN: checks vicinity of a window
  - validated with real patient data from 2013 brain tumor segmentation challenge (BRATS2013)



Case #10: [Plis et al., 2014]

- ⋆ analysis of structural and functional brain imaging data
  - generative modeling using RBM and DBN
  - learn physiologically important representations and detect latent regions in neuroimaging data

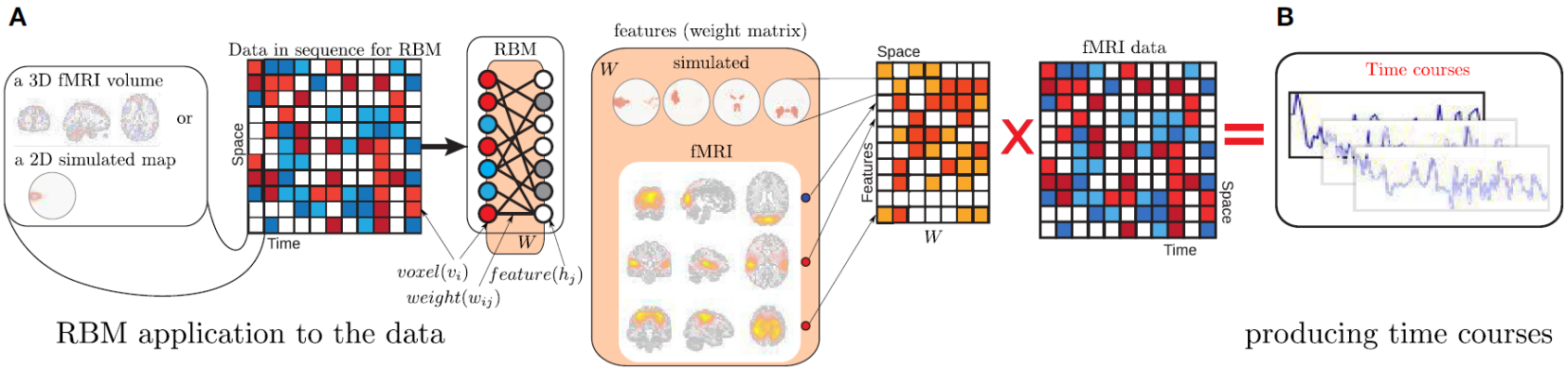


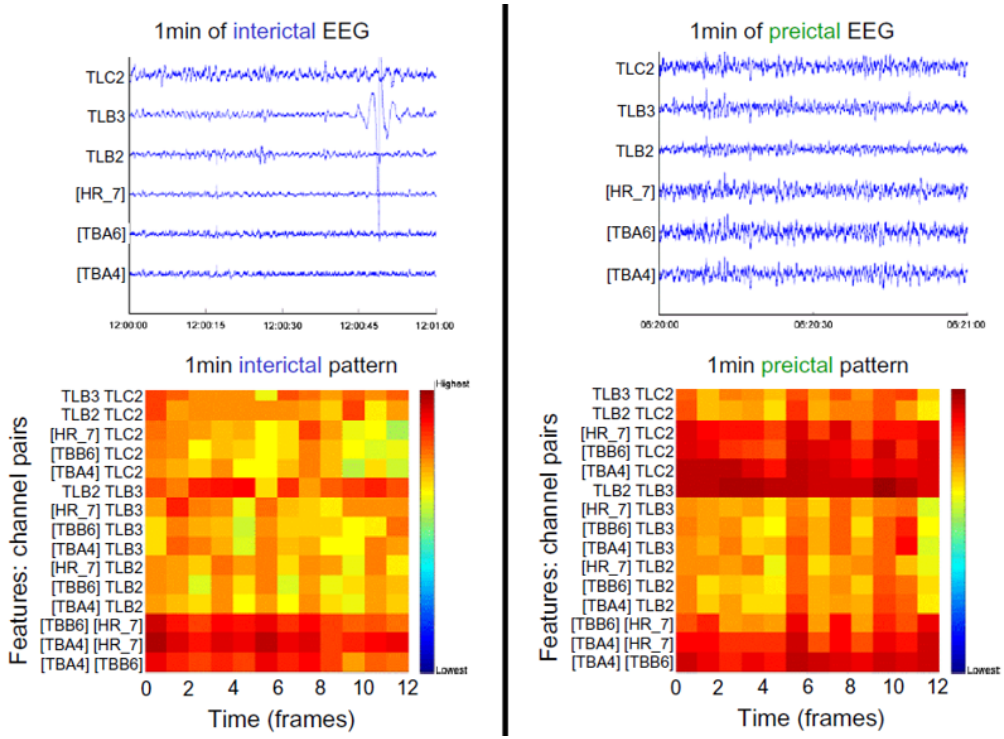
FIGURE 1 | (A) An illustration of how an RBM is applied to the data as well as a graphical representation of RBM's structure. **(B)** Demonstrates the way we produce time courses from the data and learned features, which is simply a projection of data into the feature space. Note, for time course computation in Section 2 we do not apply the hidden units' non-linearity after this projection, while in Section 3 the complete feed-forward processing is realized.

Outline

Case #11: [Mirowski et al., 2009]

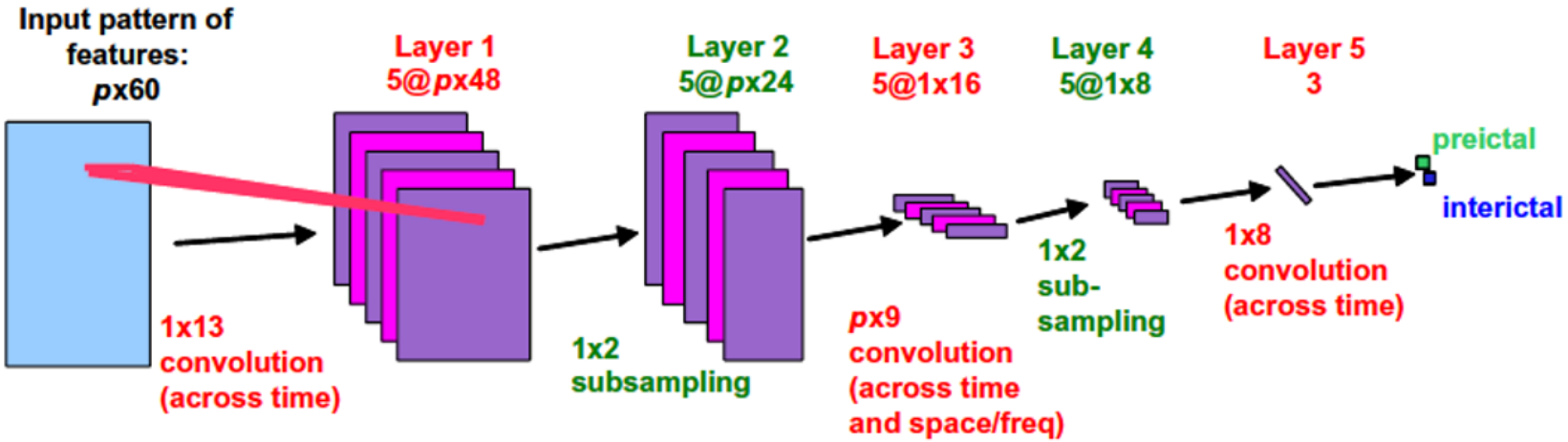
⋆ seizure prediction by electroencephalogram (EEG) classification

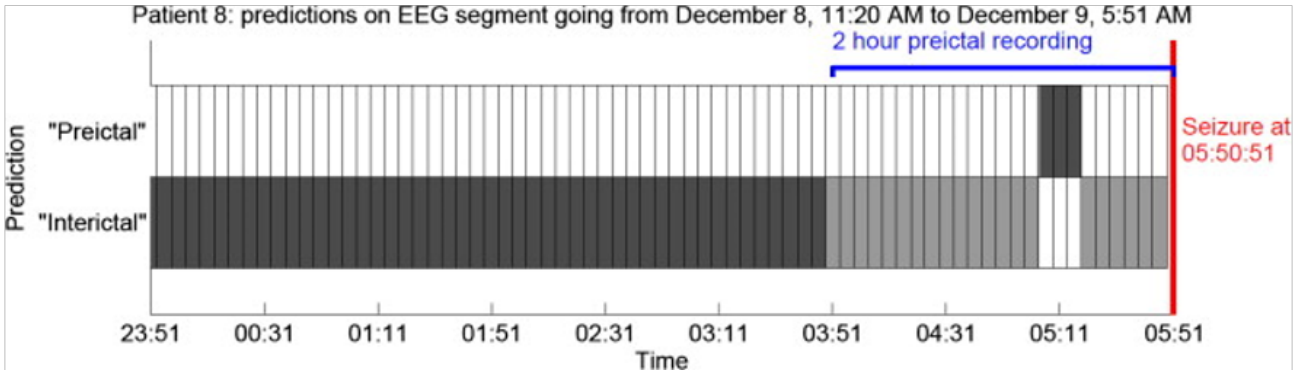
- features are extracted on user-defined time window
- spatio-temporal features are used for CNN



⋆ overall architecture: LeNet5 (five-layer CNN)

- spatial-/time-/frequency-domain features are aggregated

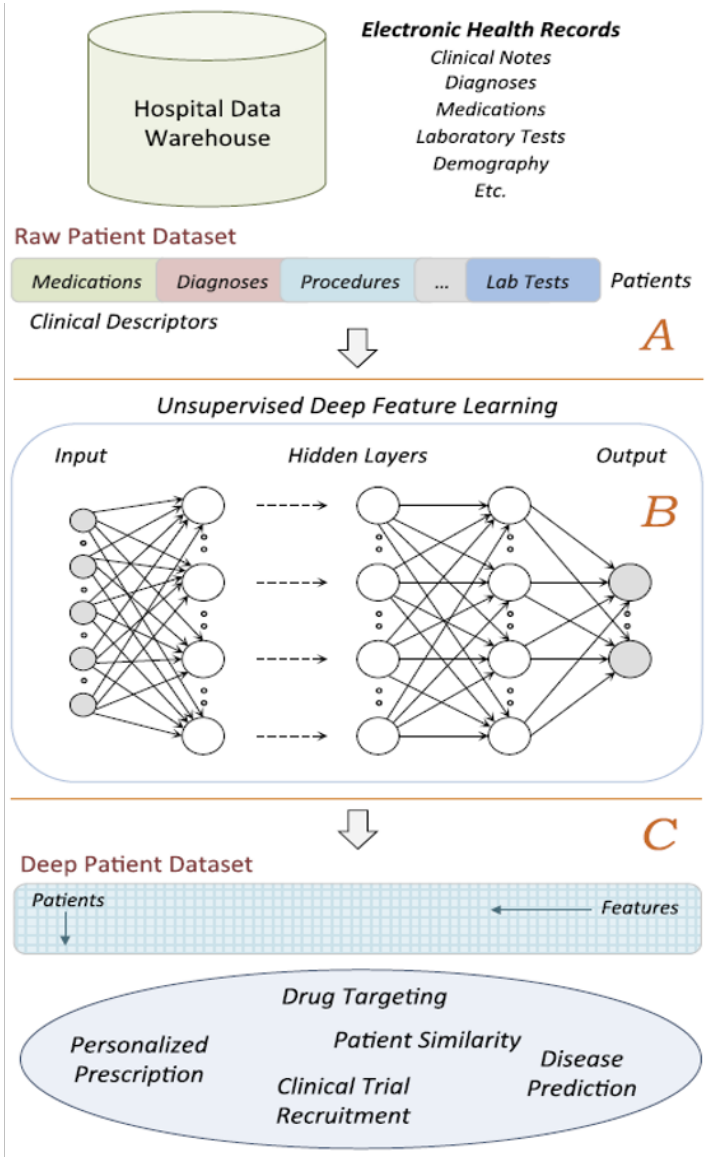




Case #12: DeepPatient [Miotto et al., 2016]

- ⋆ EHR data mining
  - unsupervised feature learning (stacked denoising AE)
  - 704,587 patients w/ 41,072 clinical descriptors
  - classification by random forests

Time Interval = 1 year (76,214 patients)			
	Area under the ROC curve		
Disease	RawFeat	PCA	DeepPatient
Diabetes mellitus with complications	0.794	0.861	**0.907**
Cancer of rectum and anus	0.863	0.821	**0.887**
Cancer of liver and intrahepatic bile duct	0.830	0.867	**0.886**
Regional enteritis and ulcerative colitis	0.814	0.843	**0.870**
Congestive heart failure (non-hypertensive)	0.808	0.808	**0.865**
Attention-deficit and disruptive behavior disorders	0.730	0.797	**0.863**
Cancer of prostate	0.692	0.820	**0.859**
Schizophrenia	0.791	0.788	**0.853**
Multiple myeloma	0.783	0.739	**0.849**
Acute myocardial infarction	0.771	0.775	**0.847**



More papers on deep learning in biomedicine

Aliper, A., Plis, S., Artemov, A., Ulloa, A., Mamoshina, P., and Zhavoronkov, A. (2016). Deep learning applications for predicting pharmacological properties of drugs and drug repurposing using transcriptomic data. *Molecular pharmaceutics*.

Asgari, E. and Mofrad, M. R. (2015). Continuous distributed representation of biological sequences for deep proteomics and genomics. *PloS one*, 10(11):e0141287.

Chen, L., Cai, C., Chen, V., and Lu, X. (2015). Trans-species learning of cellular signaling systems with bimodal deep belief networks. *Bioinformatics*, 31(18):3008–3015.

Chen, L., Cai, C., Chen, V., and Lu, X. (2016a). Learning a hierarchical representation of the yeast transcriptomic machinery using an autoencoder model. *BMC bioinformatics*, 17(1):97.

Chen, Y., Li, Y., Narayan, R., Subramanian, A., and Xie, X. (2016b). Gene expression inference with deep learning. *Bioinformatics*, page btw074.

Di Lena, P., Nagata, K., and Baldi, P. (2012). Deep architectures for protein contact map prediction. *Bioinformatics*, 28(19):2449–2457.

Eickholt, J. and Cheng, J. (2012). Predicting protein residue–residue contacts using deep networks and boosting. *Bioinformatics*, 28(23):3066–3072.

Eickholt, J. and Cheng, J. (2013). DNdisorder: predicting protein disorder using boosting and deep networks. *BMC bioinformatics*, 14(1):1.

Gawehn, E., Hiss, J. A., and Schneider, G. (2016). Deep learning in drug discovery. *Molecular Informatics*, 35(1):3–14.

Kraus, O. Z., Ba, J. L., and Frey, B. J. (2016). Classifying and segmenting microscopy images with deep multiple instance learning. *Bioinformatics*, 32(12):i52–i59.

Leung, M. K., Xiong, H. Y., Lee, L. J., and Frey, B. J. (2014). Deep learning of the tissue-regulated splicing code. *Bioinformatics*, 30(12):i121–i129.

Li, Y., Chen, C.-Y., and Wasserman, W. W. (2015). Deep feature selection: Theory and application to identify enhancers and promoters. In *International Conference on Research in Computational Molecular Biology*, pages 205–217. Springer.

Liang, M., Li, Z., Chen, T., and Zeng, J. (2015). Integrative data analysis of multi-platform cancer data with a multimodal deep learning approach. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 12(4):928–937.

Liu, F., Ren, C., Li, H., Zhou, P., Bo, X., and Shu, W. (2015). De novo identification of replication-timing domains in the human genome by deep learning. *Bioinformatics*, page btv643.

Park, Y. and Kellis, M. (2015). Deep learning for regulatory genomics. *Nat. Biotechnol*, 33:825–826.

Putin, E., Mamoshina, P., Aliper, A., Korzinkin, M., Moskalev, A., Kolosov, A., Ostrovskiy, A., Cantor, C., Vijg, J., and Zhavoronkov, A. (2016). Deep biomarkers of human aging: Application of deep neural networks to biomarker development. *Aging*, 8(5):1–021.

Quang, D., Chen, Y., and Xie, X. (2014). DANN: a deep learning approach for annotating the pathogenicity of genetic variants. *Bioinformatics*, page btu703.

Quang, D. and Xie, X. (2016). DanQ: a hybrid convolutional and recurrent deep neural network for quantifying the function of dna sequences. *Nucleic acids research*, page gkw226.

Tan, J., Ung, M., Cheng, C., and Greene, C. S. (2014). Unsupervised feature construction and knowledge extraction from genome-wide assays of breast cancer with denoising autoencoders. In *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, volume 20, pages 132–143. NIH Public Access.

Tripathi, R., Patel, S., Kumari, V., Chakraborty, P., and Varadwaj, P. K. (2016). DeepLNC, a long non-coding rna prediction tool using deep neural network. *Network Modeling Analysis in Health Informatics and Bioinformatics*, 5(1):1–14.

Wang, C., Liu, J., Luo, F., Tan, Y., Deng, Z., and Hu, Q.-N. (2014). Pairwise input neural network for target-ligand interaction prediction. In *Bioinformatics and Biomedicine (BIBM), 2014 IEEE International Conference on*, pages 67–70. IEEE.

Xie, W., Noble, J. A., and Zisserman, A. (2016). Microscopy cell counting and detection with fully convolutional regression networks. *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization*, pages 1–10.

Xiong, H. Y., Alipanahi, B., Lee, L. J., Bretschneider, H., Merico, D., Yuen, R. K., Hua, Y., Gueroussov, S., Najafabadi, H. S., Hughes, T. R., et al. (2015). The human splicing code reveals new insights into the genetic determinants of disease. *Science*, 347(6218):1254806.

Zeng, H., Edwards, M. D., Liu, G., and Gifford, D. K. (2016). Convolutional neural network architectures for predicting dna–protein binding. *Bioinformatics*, 32(12):i121–i127.

Zeng, T., Li, R., Mukkamala, R., Ye, J., and Ji, S. (2015). Deep convolutional neural networks for annotating gene expression patterns in the mouse brain. *BMC bioinformatics*, 16(1):1.

Outline

Architecture selection and hyper-parameter tuning

- ⋆ wanted: deep learning experts

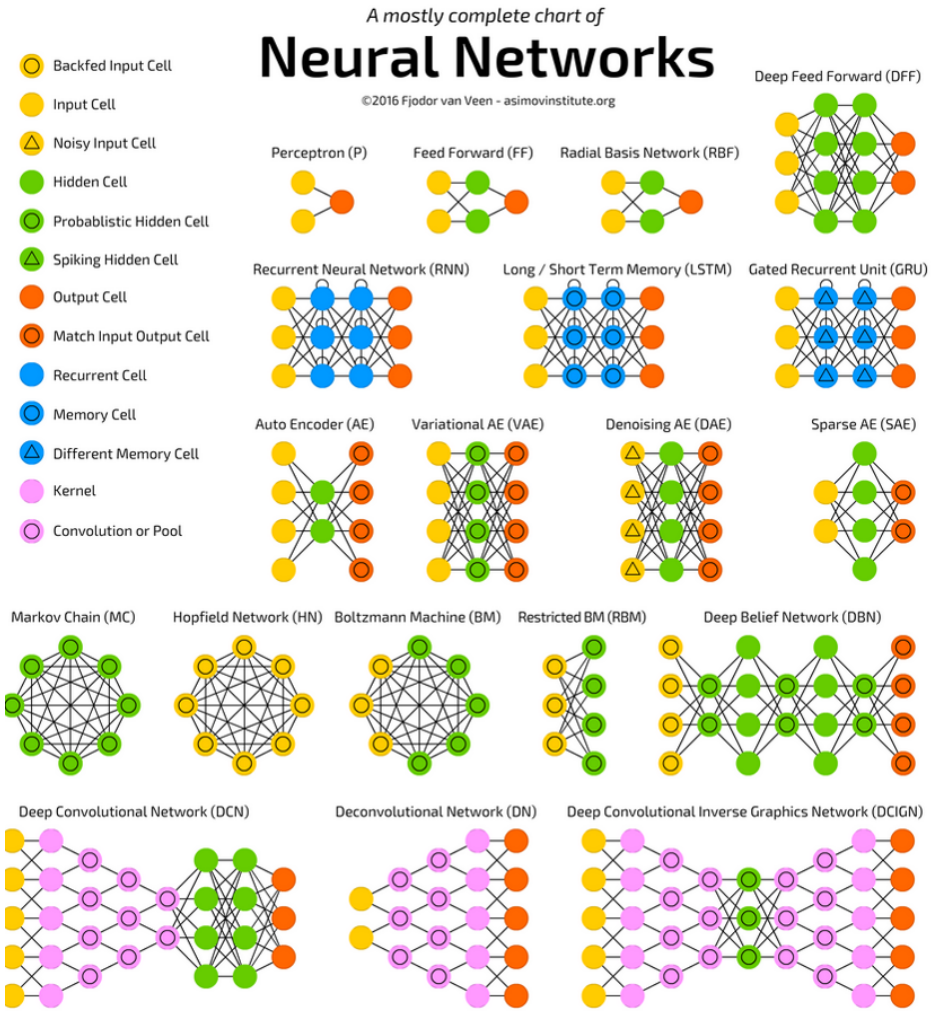


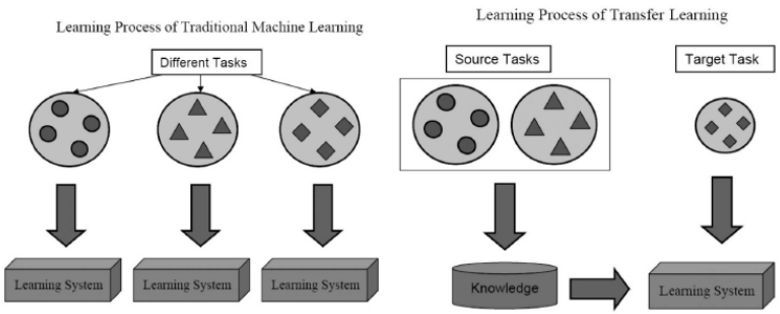
Table 2. Central parameters of a neural network and recommended settings.

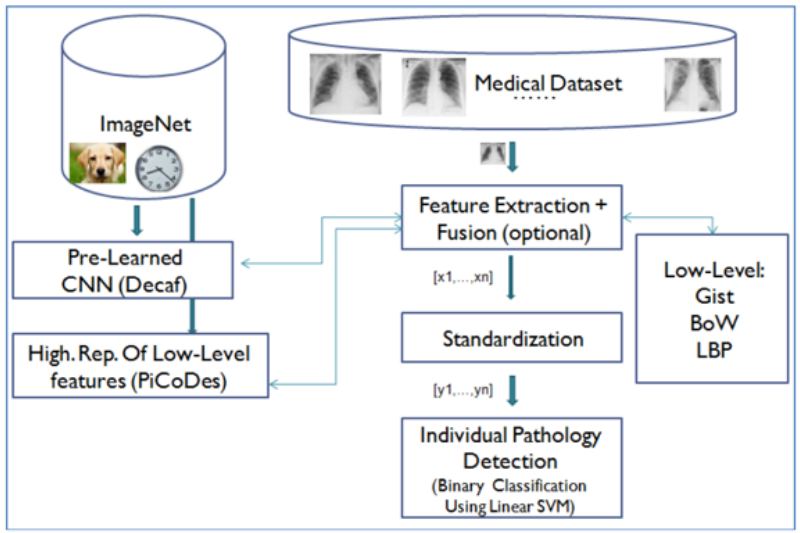
Name	Range	Default value
Learning rate	0.1, 0.01, 0.001, 0.0001	0.01
Batch size	64, 128, 256	128
Momentum rate	0.8, 0.9, 0.95	0.9
Weight initialization	Normal, Uniform, Glorot uniform	Glorot uniform
Per-parameter adaptive learning rate methods	RMSprop, Adagrad, Adadelta, Adam	Adam
Batch normalization	Yes, no	Yes
Learning rate decay	None, linear, exponential	Linear (rate 0.5)
Activation function	Sigmoid, Tanh, ReLU, Softmax	ReLU
Dropout rate	0.1, 0.25, 0.5, 0.75	0.5
L1, L2 regularization	0, 0.01, 0.001	

source: Asimov Institute, [Angermueller et al., 2016]

Data preparation and unsupervised learning

- ⋆ data preprocessing
  - handling missing/imbalanced data
  - deep learning vs state-of-the-art feature extractor
- ⋆ how to take advantage of unlabeled data
  - unsupervised/semisupervised learning
  - transfer learning

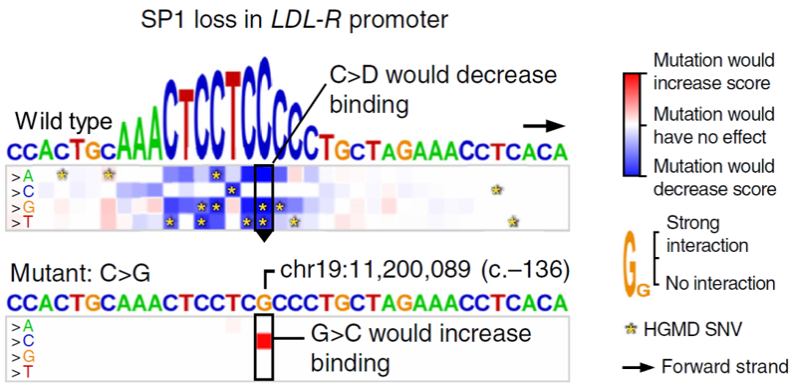


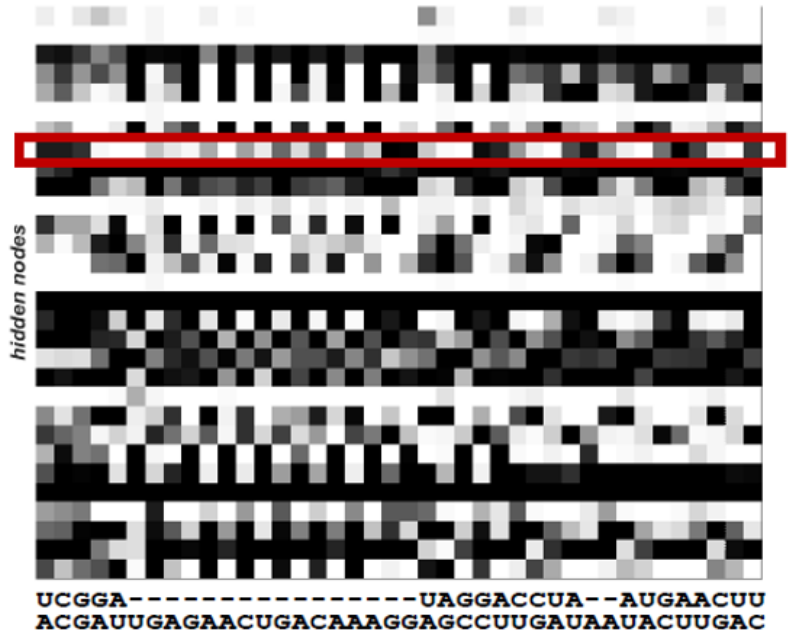


source: [Pan and Yang, 2010, Bar et al., 2015]

Interpretation

- ⋆ how to turn a black-box into a white-box?
- ⋆ interpretation is critical in biomedicine
  - not just good results but explainable results
  - visualization of deep network internals
  - attention mechanism



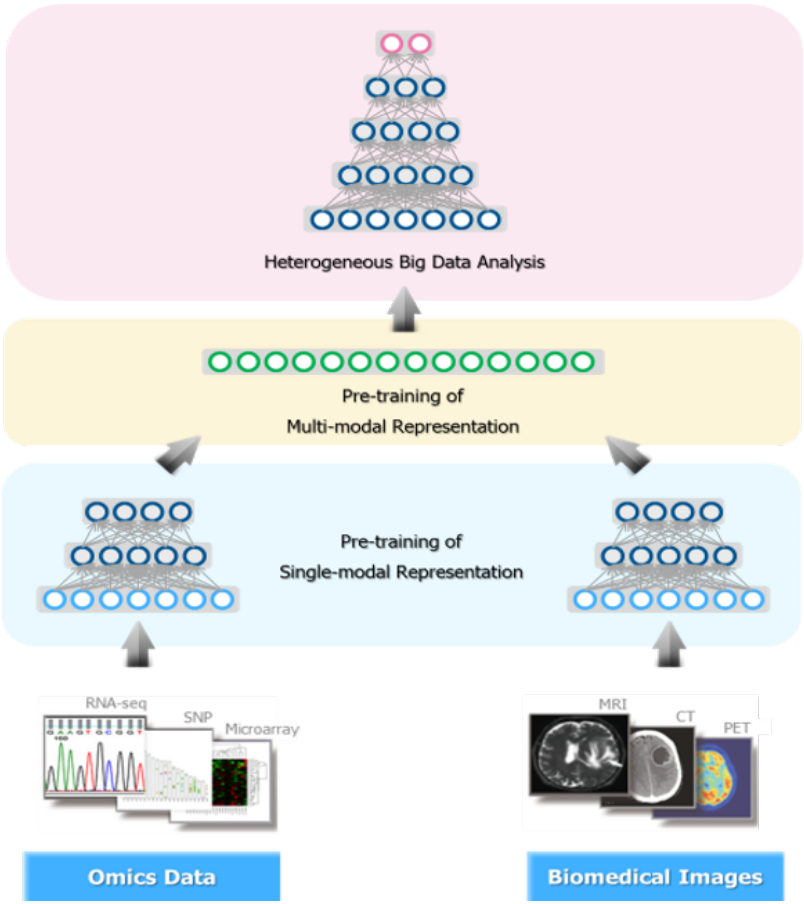


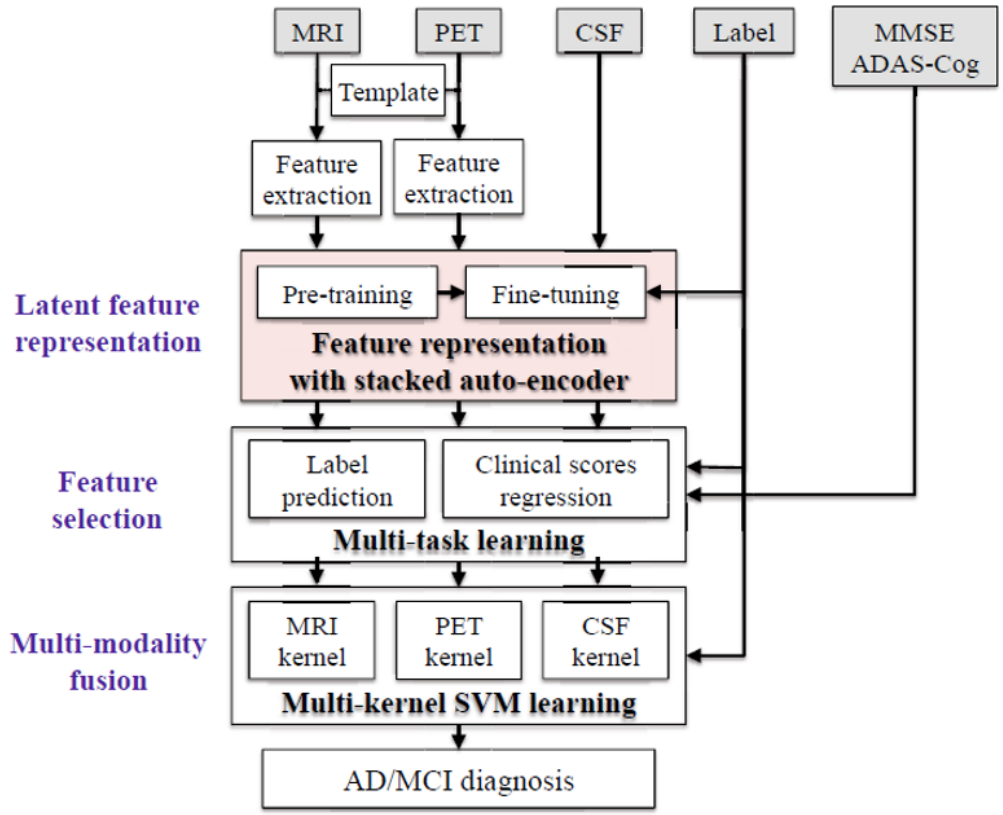
source: [Alipanahi et al., 2015, Lee et al., 2016]

Multimodality and heterogeneity of data

⋆ exploiting information from multiple sources would be beneficial

e.g. omics, images, signals, and EMRs of a patient





source: [Suk and Shen, 2013]

Deep learning libraries and frameworks

- ⋆ various tools exist (mostly in python, C++)
  - notable: Theano (Keras), Caffe, TensorFlow

Table 3. Comparison of deep learning libraries

	Core	Speed for batch* (ms)	Multi-GPU	Distributed	Strengths [56,57]
Caffe	C++	651.6	O	X	Pre-trained models supported
Neon	Python	386.8	O	X	Speed
TensorFlow	C++	962.0	O	O	Heterogeneous distributed computing
Theano	Python	733.5	X	X	Ease of use with higher-level wrappers
Torch	Lua	506.6	O	X	Functional extensionality

Notes: Speed for batch* is based on the averaged processing times for AlexNet [33] with batch size of 256 on a single GPU [57]; Caffe, Neon, Theano, Torch was utilized with cuDNN v.3 while TensorFlow was utilized with cuDNN v.2.

source: [Min et al., 2016]

Table 1. Overview of existing deep learning frameworks, comparing four widely used software solutions.

	Caffe	Theano	Torch7	TensorFlow
Core language	C++	Python, C++	LuaJIT	C++
Interfaces	Python, Matlab	Python	C	Python
Wrappers		Lasagne, Keras, sklearn-theano		Keras, Pretty Tensor, Scikit Flow
Programming paradigm	Imperative	Declarative	Imperative	Declarative
Well suited for	CNNs, Reusing existing models, Computer vision	Custom models, RNNs	Custom models, CNNs, Reusing existing models	Custom models, Parallelization, RNNs

source: [Angermueller et al., 2016]

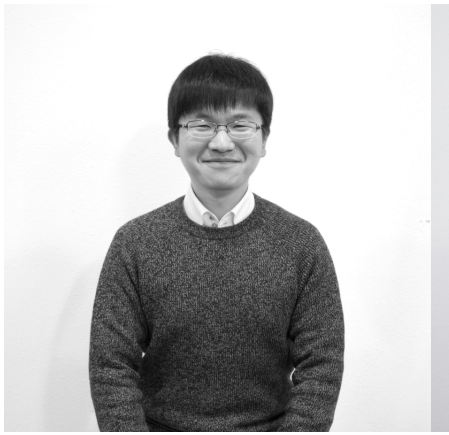
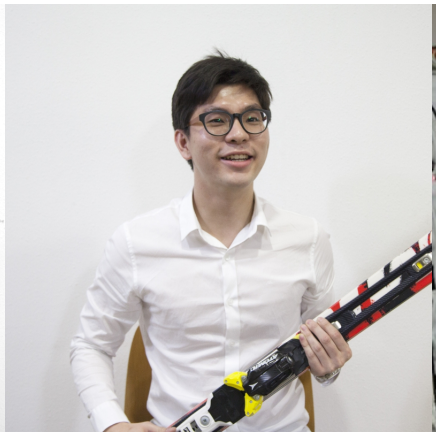
Outline

Summary

- ⋆ deep learning: neural networks with multiple hidden layers
  - attempts to model high-level abstractions to mimic humans
  - backed by big data, parallel processing & sophisticated algorithms
  - RBM, AE, DBN, CNN, RNN and many other developments
- ⋆ key elements to train deep neural networks
  - pre-training, fine-tuning, optimization, regularization, parallelization
- ⋆ various applications in bioinformatics and health informatics
  - CNN dominates for now (1-d for sequences; 2-d for imaging)
  - RNN and other architectures are emerging for new applications
  - representation learning: key to human-level intelligence
- ⋆ challenges and opportunities
  - interpretation, multimodality, data handling, unsupervised learning

Acknowledgements

⋆ Byunghan Lee, Seonwoo Min, and my students and postdoc @SNU


⋆ funding sponsors

Enjoy your ACM-BCB 2016!

for more information: sryoon@snu.ac.kr

References I

Abu-Mostafa, Y. S., Magdon-Ismail, M., and Lin, H.-T. (2012). *Learning from data*, volume 4. AMLBook Singapore.

Alipanahi, B., Delong, A., Weirauch, M. T., and Frey, B. J. (2015). Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning. *Nature biotechnology*.

Angermueller, C., Pärnamaa, T., Parts, L., and Stegle, O. (2016). Deep learning for computational biology. *Molecular Systems Biology*, 12(7):878.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Bar, Y., Diamant, I., Wolf, L., and Greenspan, H. (2015). Deep learning with non-medical training used for chest pathology identification. In *SPIE Medical Imaging*, page 94140V. International Society for Optics and Photonics.

Cho, K., Van Merriënboer, B., Bahdanau, D., and Bengio, Y. (2014a). On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014b). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Coiera, E. (2003). Clinical decision support systems. *Guide to health informatics*, 2(1).

Duda, R. O., Hart, P. E., and Stork, D. G. (2012). *Pattern Classification*. John Wiley & Sons.

Fischer, A. and Igel, C. (2014). Training restricted Boltzmann machines: An introduction. *Pattern Recognition*, 47(1):25–39.

Geman, S., Bienenstock, E., and Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural computation*, 4(1):1–58.

Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448.

References II

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). Deep learning. Book in preparation for MIT Press.

Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A. C., and Bengio, Y. (2013). Maxout networks. *ICML (3)*, 28:1319–1327.

Graves, A. (2012). *Supervised Sequence Labelling with Recurrent Neural Networks*. Springer.

Graves, A., Wayne, G., and Danihelka, I. (2014). Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Greenspan, H., van Ginneken, B., and Summers, R. M. (2016). Guest editorial deep learning in medical imaging: Overview and future promise of an exciting new technique. *IEEE Transactions on Medical Imaging*, 35(5):1153–1159.

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., and Schmidhuber, J. (2015). Lstm: A search space odyssey. *arXiv preprint arXiv:1503.04069*.

Harley, A. W. (2015). An interactive node-link visualization of convolutional neural networks. In *International Symposium on Visual Computing*, pages 867–877. Springer.

Havaei, M., Davy, A., Warde-Farley, D., Biard, A., Courville, A., Bengio, Y., Pal, C., Jodoin, P.-M., and Larochelle, H. (2016). Brain tumor segmentation with deep neural networks. *Medical Image Analysis*.

Haykin, S. S. (2009). *Neural Networks and Learning Machines*. Pearson Upper Saddle River, NJ, USA:, 3 edition.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*.

Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural computation*, 9(8):1735–1780.

References III

Kelley, D. R., Snoek, J., and Rinn, J. L. (2016). Basset: Learning the regulatory code of the accessible genome with deep convolutional neural networks. *Genome Research*.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

LeCun, Y., Bengio, Y., and Hinton, G. (2015a). Deep learning. *Nature*, 521(7553):436–444.

LeCun, Y. et al. (2015b). Lenet-5, convolutional neural networks. *URL: http://yann. lecun. com/exdb/lenet*.

Lee, B., Baek, J., Park, S., and Yoon, S. (2016). deepTarget: End-to-end learning framework for microRNA target prediction using deep recurrent neural networks. *ACM-BCB (arXiv preprint arXiv:1603.09123)*.

Lee, S., Choi, M., Choi, H.-s., Park, M. S., and Yoon, S. (2015). FingerNet: Deep learning-based robust finger joint detection from radiographs. In *IEEE Biomedical Circuits and Systems Conference (BioCAS)*. IEEE.

Leung, M. K., Delong, A., Alipanahi, B., and Frey, B. J. (2016). Machine learning in genomic medicine: A review of computational problems and data sets. *Proceedings of the IEEE*, 104(1):176–197.

Lippmann, R. (1987). An introduction to computing with neural nets. *IEEE ASSP magazine*, 4(2):4–22.

Mamoshina, P., Vieira, A., Putin, E., and Zhavoronkov, A. (2016). Applications of deep learning in biomedicine. *Molecular pharmaceutics*, 13(5):1445–1454.

Min, S., Lee, B., and Yoon, S. (2016). Deep learning in bioinformatics. *Briefings in Bionformatics (arXiv preprint arXiv:1603.06430)*.

Miotto, R., Li, L., Kidd, B. A., and Dudley, J. T. (2016). Deep patient: An unsupervised representation to predict the future of patients from the electronic health records. *Scientific Reports*, 6.

Mirowski, P., Madhavan, D., LeCun, Y., and Kuzniecky, R. (2009). Classification of patterns of EEG synchronization for seizure prediction. *Clinical neurophysiology*, 120(11):1927–1940.

References IV

Ng, A. (2011). Sparse autoencoder. *CS294A Lecture notes*, pages 1–19.

Noh, H., Hong, S., and Han, B. (2015). Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1520–1528.

O'Reilly, R. C., Wyatte, D., Herd, S., Mingus, B., and Jilk, D. J. (2013). Recurrent processing during object recognition. *Frontiers in psychology*, 4:124.

Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359.

Park, Y. and Kellis, M. (2015). Deep learning for regulatory genomics. *Nataure Biotechnololgy*, 33:825–826.

Pascanu, R., Gulcehre, C., Cho, K., and Bengio, Y. (2013). How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026*.

Peng, Y., Zhang, Y., and Wang, L. (2010). Artificial intelligence in biomedical engineering and informatics: an introduction and review. *Artificial intelligence in medicine*, 48(2):71–73.

Plis, S. M., Hjelm, D. R., Salakhutdinov, R., Allen, E. A., Bockholt, H. J., Long, J. D., Johnson, H. J., Paulsen, J. S., Turner, J. A., and Calhoun, V. D. (2014). Deep learning for neuroimaging: a validation study. *Frontiers in Neuroscience*, 8.

Rampasek, L. and Goldenberg, A. (2016). TensorFlow: Biology's gateway to deep learning? *Cell Systems*, 2(1):12–14.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99.

Roth, H. R., Lu, L., Farag, A., Shin, H.-C., Liu, J., Turkbey, E. B., and Summers, R. M. (2015). DeepOrgan: Multi-level deep convolutional networks for automated pancreas segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 556–564.

References V

Sheehan, S. and Song, Y. S. (2016). Deep learning for population genetic inference. *PLoS Comput Biol*, 12(3):e1004845.

Suk, H.-I. and Shen, D. (2013). Deep learning-based feature representation for ad/mci classification. In *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 583–590.

Thériault, C., Thome, N., and Cord, M. (2013). Extended coding and pooling in the HMAX model. *IEEE Transactions on Image Processing*, 22(2):764–777.

Thorpe, S. J. and Fabre-Thorpe, M. (2001). Seeking categories in the brain. *Science*, 291(5502):260–263.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM.

Wang, G. (2016). A perspective on deep imaging. *arXiv preprint arXiv:1609.04375*.

Weston, J., Chopra, S., and Bordes, A. (2014). Memory networks. *arXiv preprint arXiv:1410.3916*.

Xu, J., Xiang, L., Liu, Q., Gilmore, H., Wu, J., Tang, J., and Madabhushi, A. (2016). Stacked sparse autoencoder (SSAE) for nuclei detection on breast cancer histopathology images. *IEEE Transactions on Medical Imaging*, 35(1):119–130.

Zeiler, M. D., Krishnan, D., Taylor, G. W., and Fergus, R. (2010). Deconvolutional networks. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2528–2535. IEEE.

Zhang, S., Zhou, J., Hu, H., Gong, H., Chen, L., Cheng, C., and Zeng, J. (2016). A deep learning framework for modeling structural features of RNA-binding protein targets. *Nucleic Acids Research*, 44(4):e32–e32.

Zhou, J. and Troyanskaya, O. G. (2015). Predicting effects of noncoding variants with deep learning-based sequence model. *Nature Methods*, 12(10):931–934.

last compiled at 14:37:19 on 2016/10/02