Introduction to Deep Learning

Greg Tsagkatakis

ICS - FORTH

Machine Learning

Play Video

https://www.youtube.com/watch?v=f_uwKZIAeM0

Agenda

Lecture #1

- Introduction
- Supervised Deep Learning

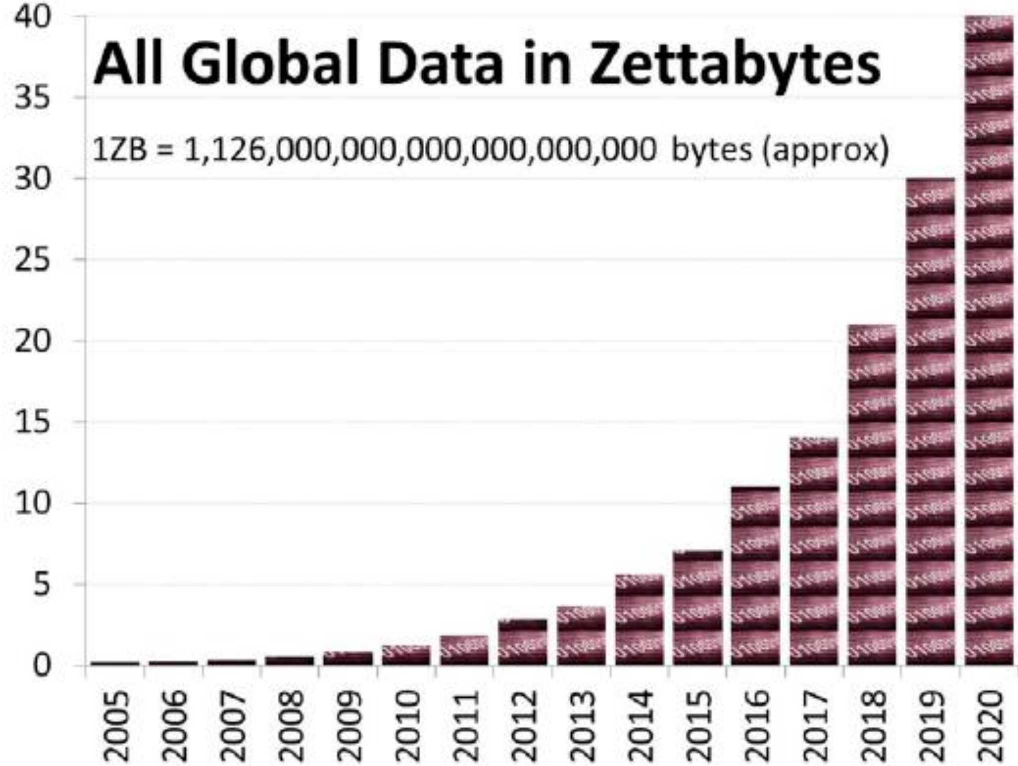
Lecture #2

- Unsupervised Deep Learning
- Deep Reinforcement learning

Big Data

The 5Vs

- **Volume**

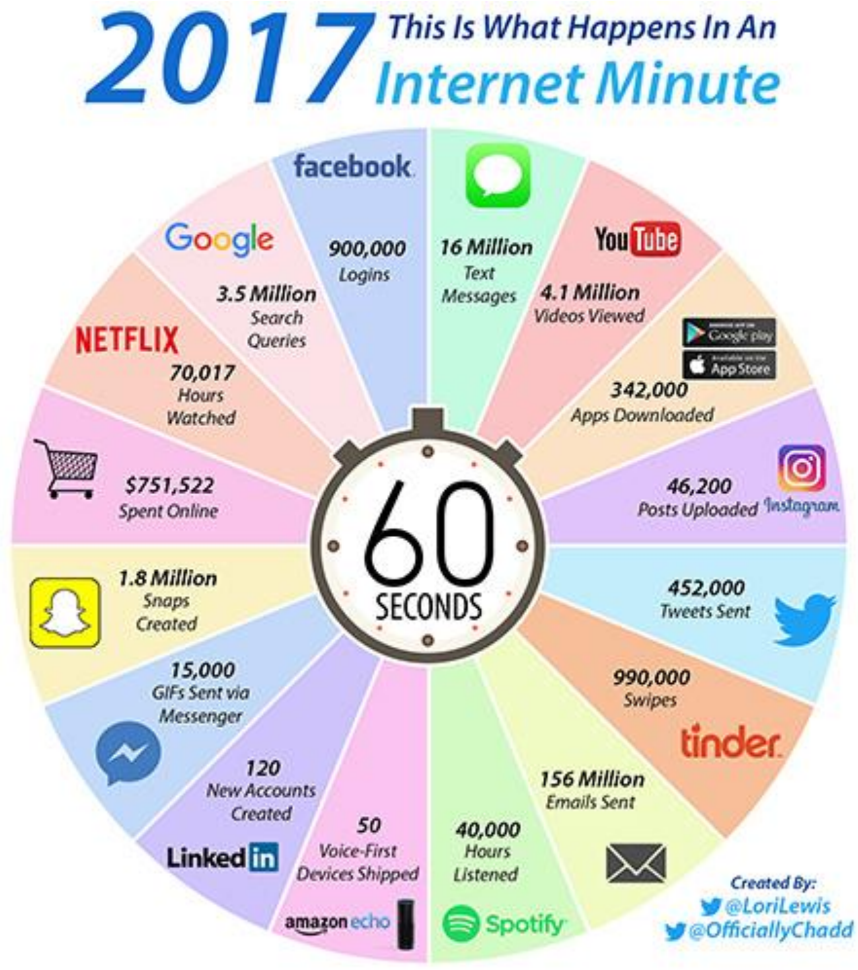


The growth in data as seen by United Nations Economic Commission for Europe.

Big Data

The 5Vs

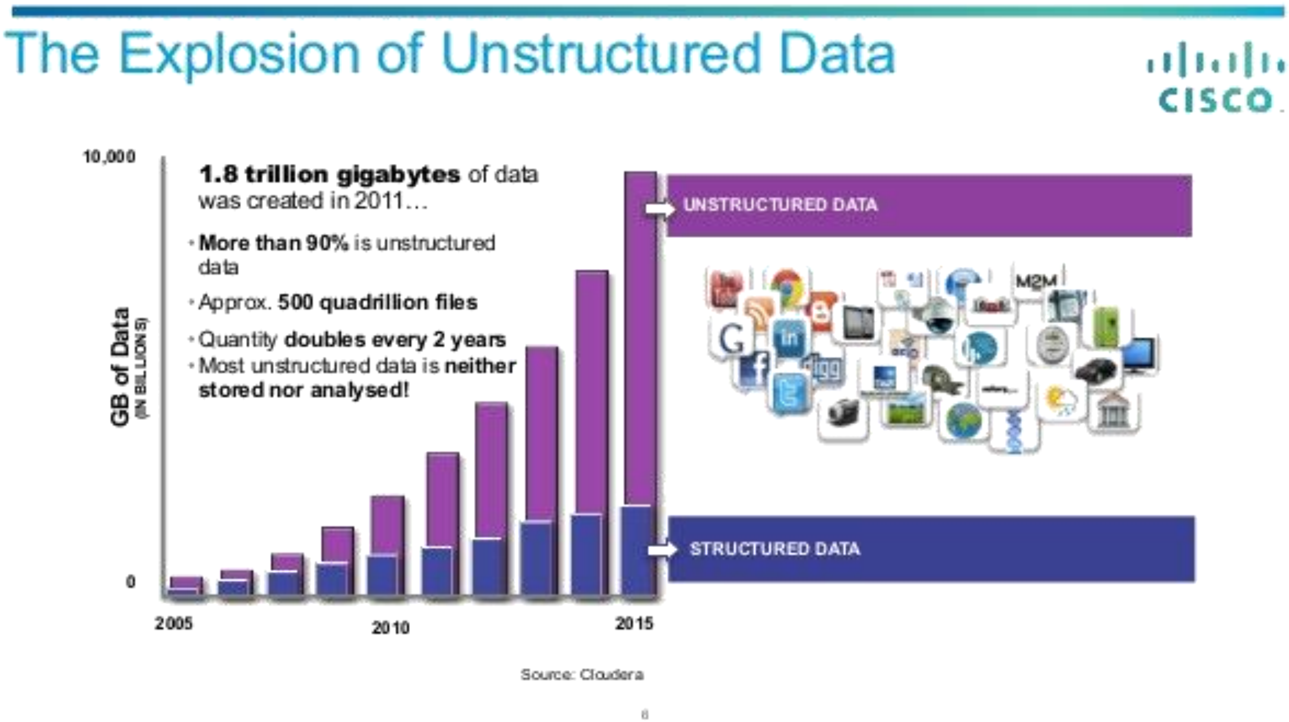
- Volume
- **Velocity**



Big Data

The 5Vs

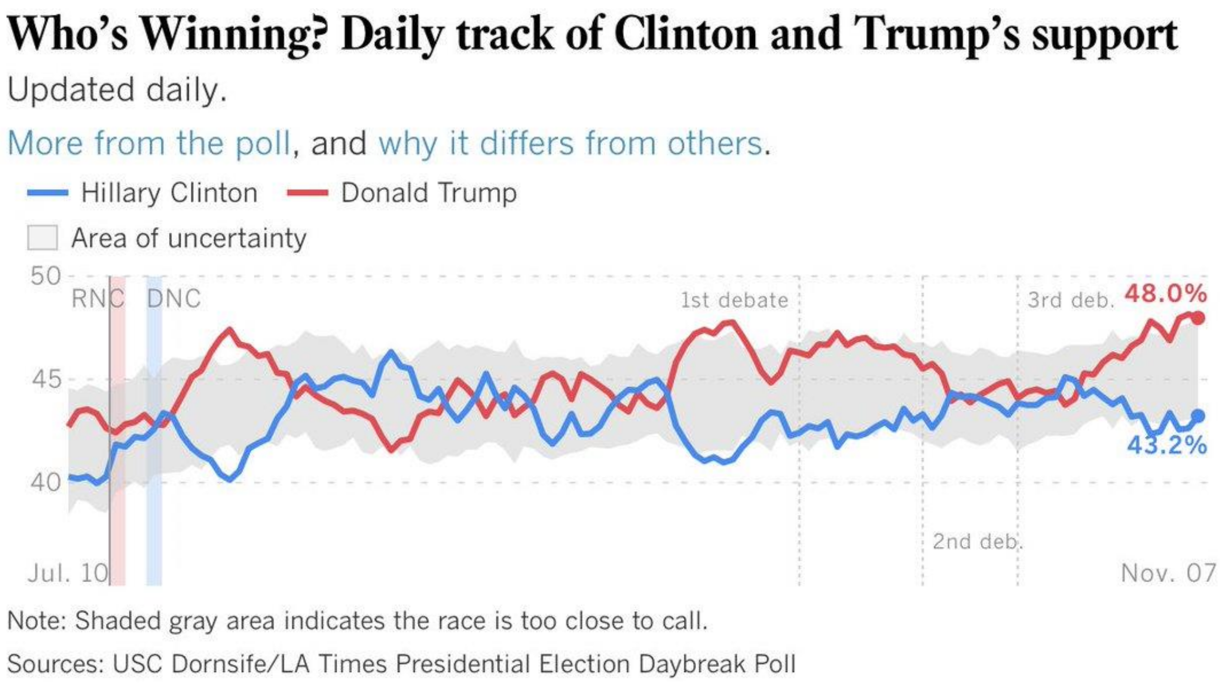
- ➢Volume
- ➢Velocity
- ➢**Variety**



Big Data

The 5Vs

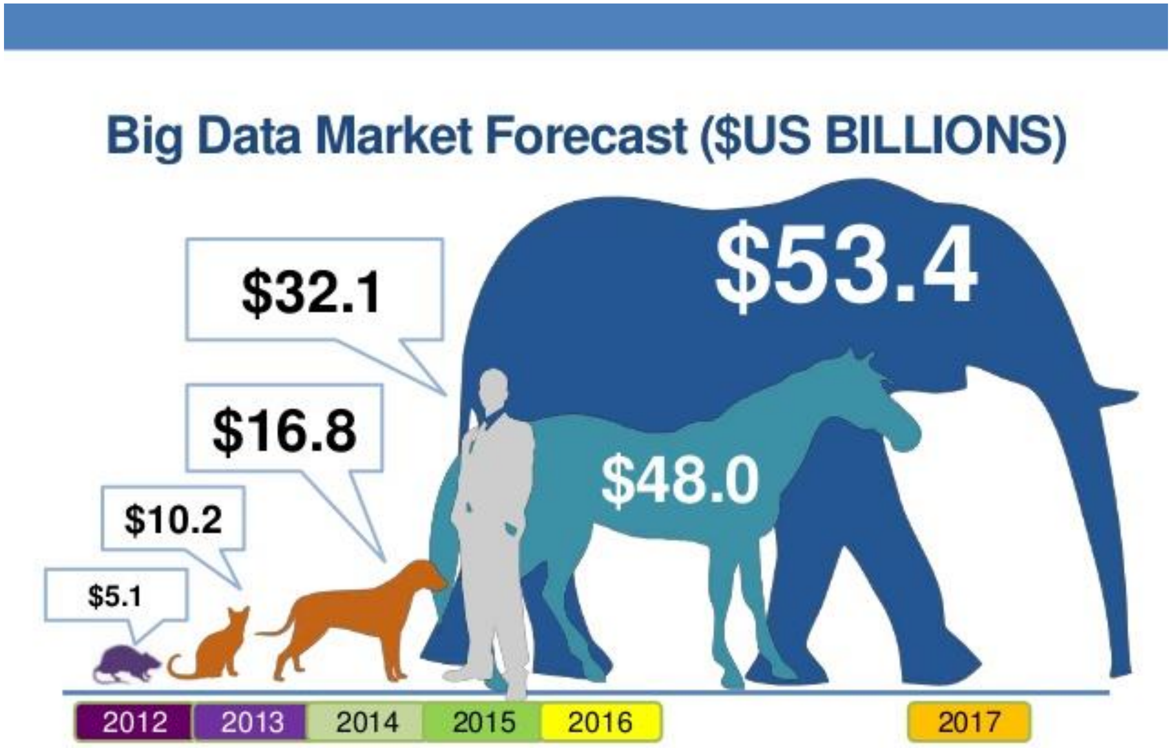
- Volume
- Velocity
- Variety
- **Veracity**



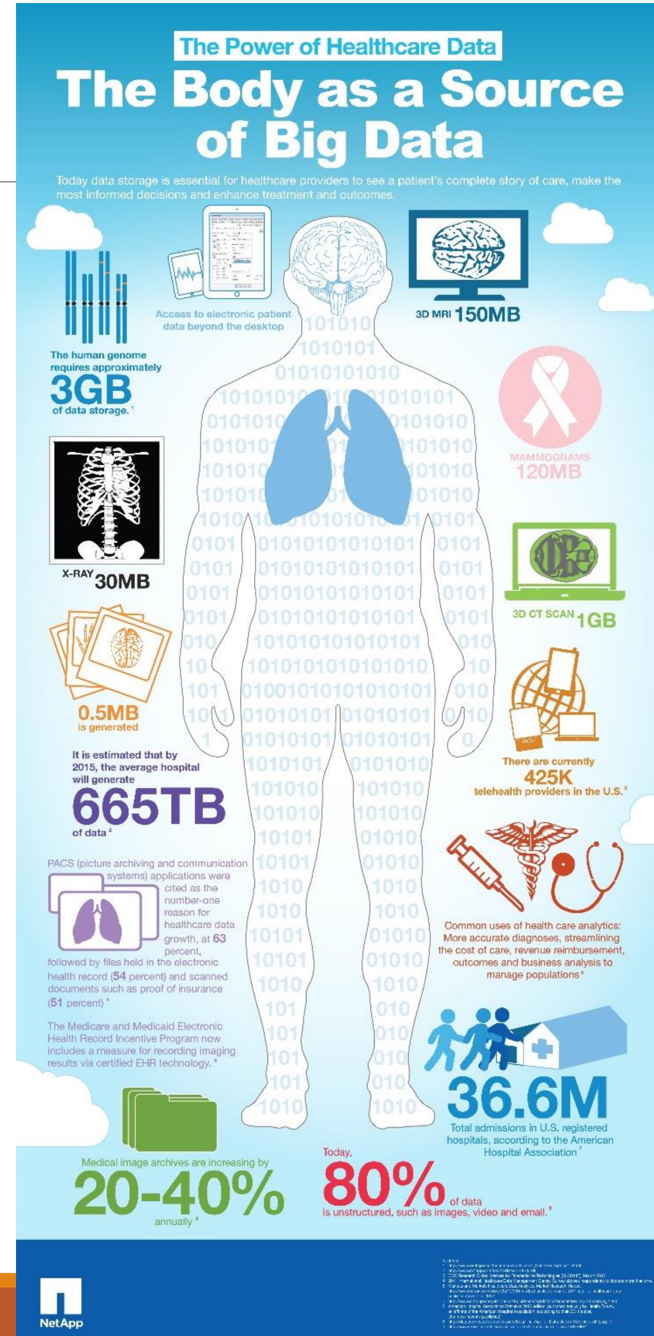
Big Data

The 5Vs

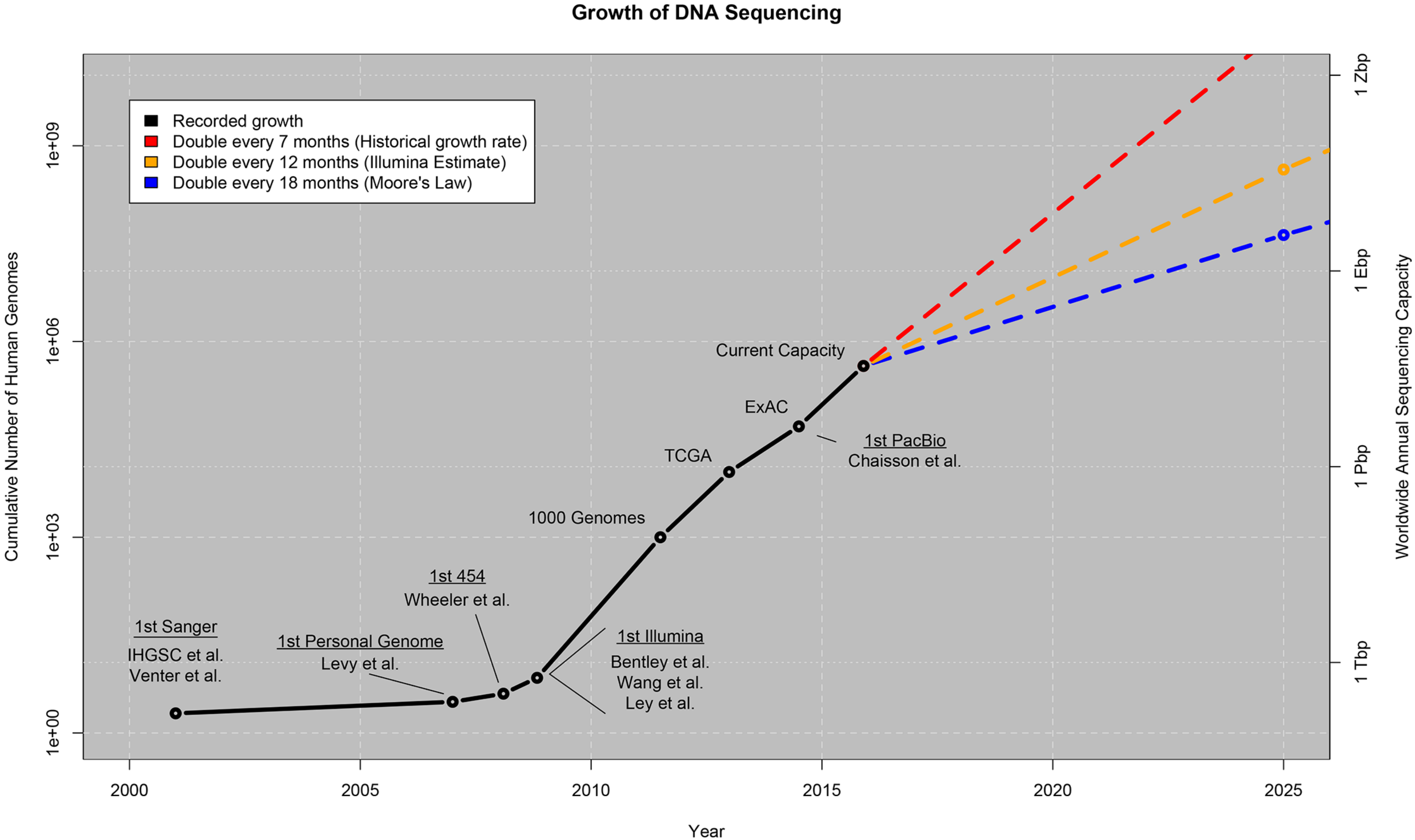
- Volume
- Velocity
- Variety
- Veracity
- **Value**



BD & Healthcare



Big Data & Genetics



Big Data & Astrophysics

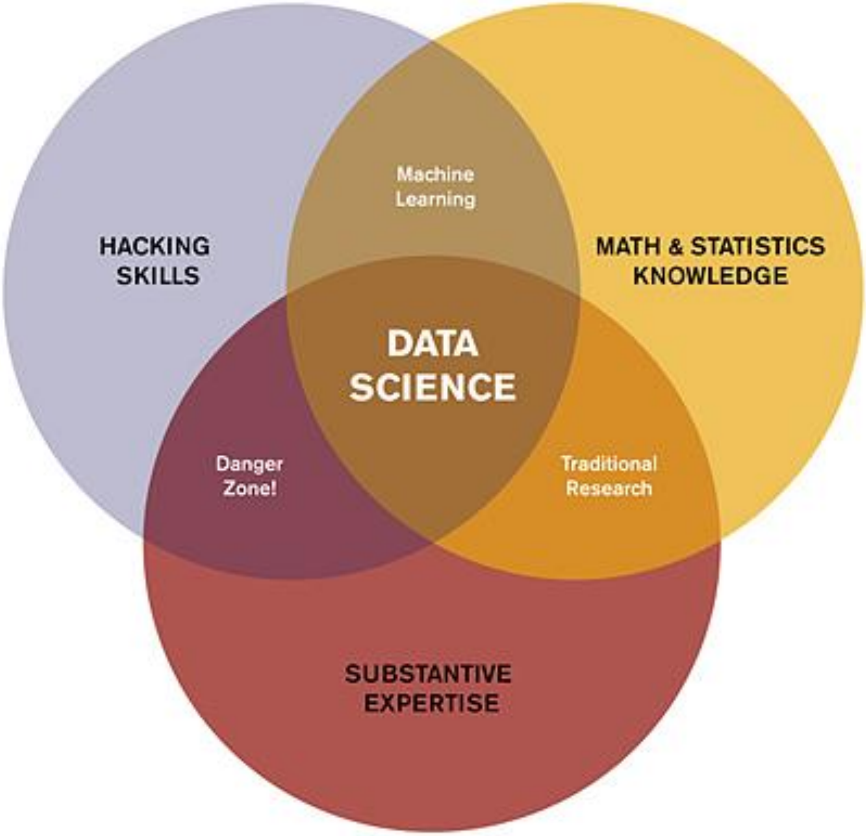
Astronomy & Astrophysics

Sky Survey Project	Volume	Velocity	Variety
Sloan Digital Sky Survey (SDSS)	50 TB	200 GB per day	Images, redshifts
Large Synoptic Survey Telescope (LSST)	~ 200 PB	10 TB per day	Images, catalogs
Square Kilometer Array (SKA)	~ 4.6 EB	150 TB per day	Images, redshifts

Astrophysics and Big Data: Challenges, Methods, and Tools. Mauro Garofalo, Alessio Botta, and Giorgio Ventre.

Handling Big Data

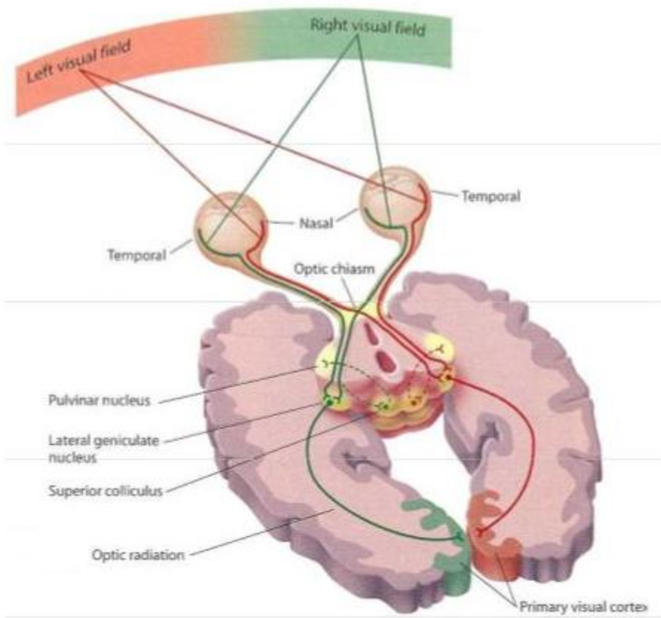
Machine Learning + Big Data -> Data science



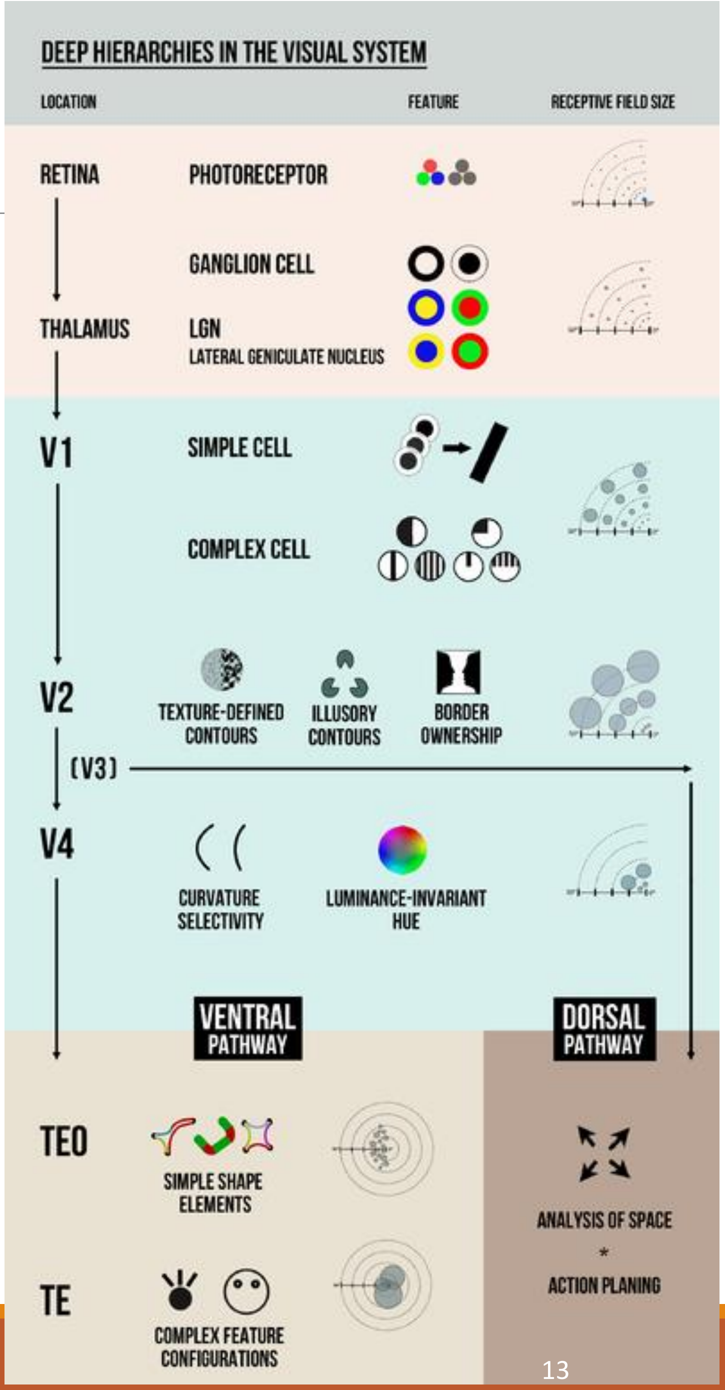
Source: https://s3.amazonaws.com/aws.drewconway.com/viz/venn_diagram/data_science.html

Big Data &the Brain

Human Visual System



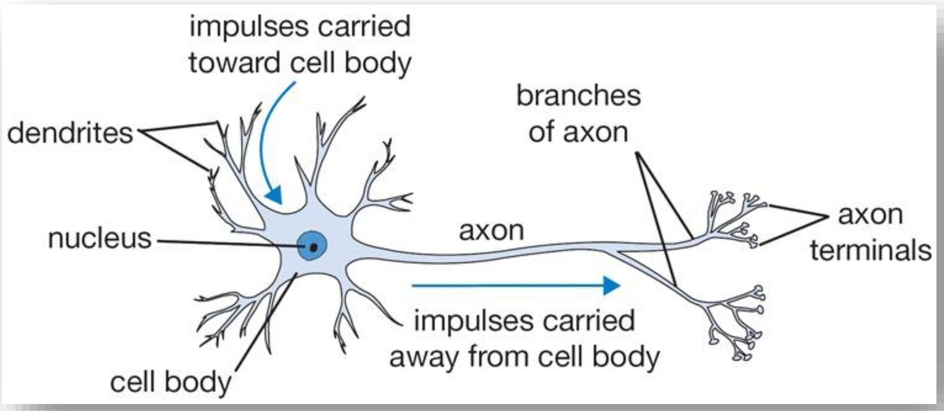
© Stephen E. Palmer, 2002



How does the Brain do it?

10^{11} neurons

10^{14}-10^{15} synapses



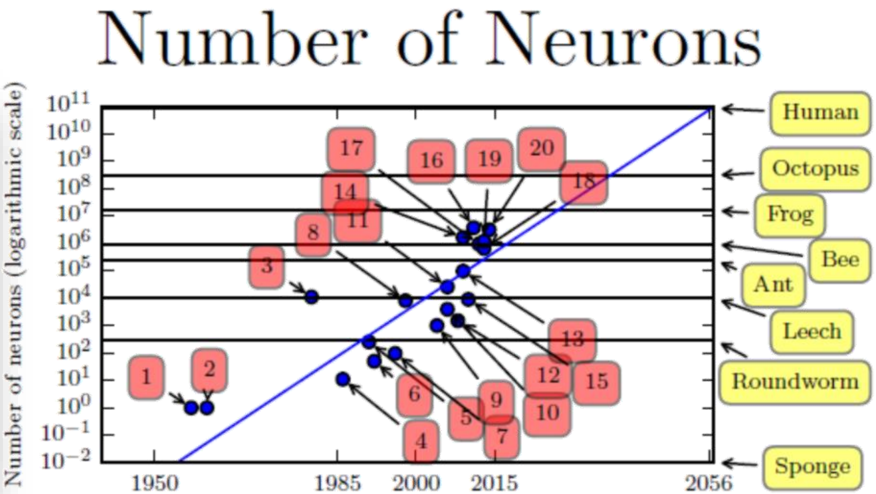
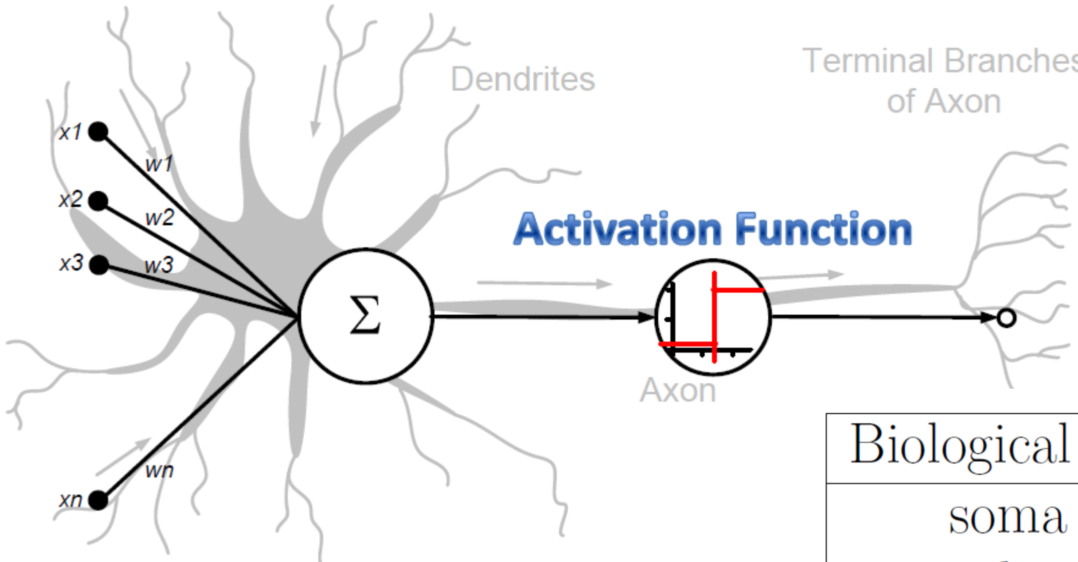


Figure 1.11

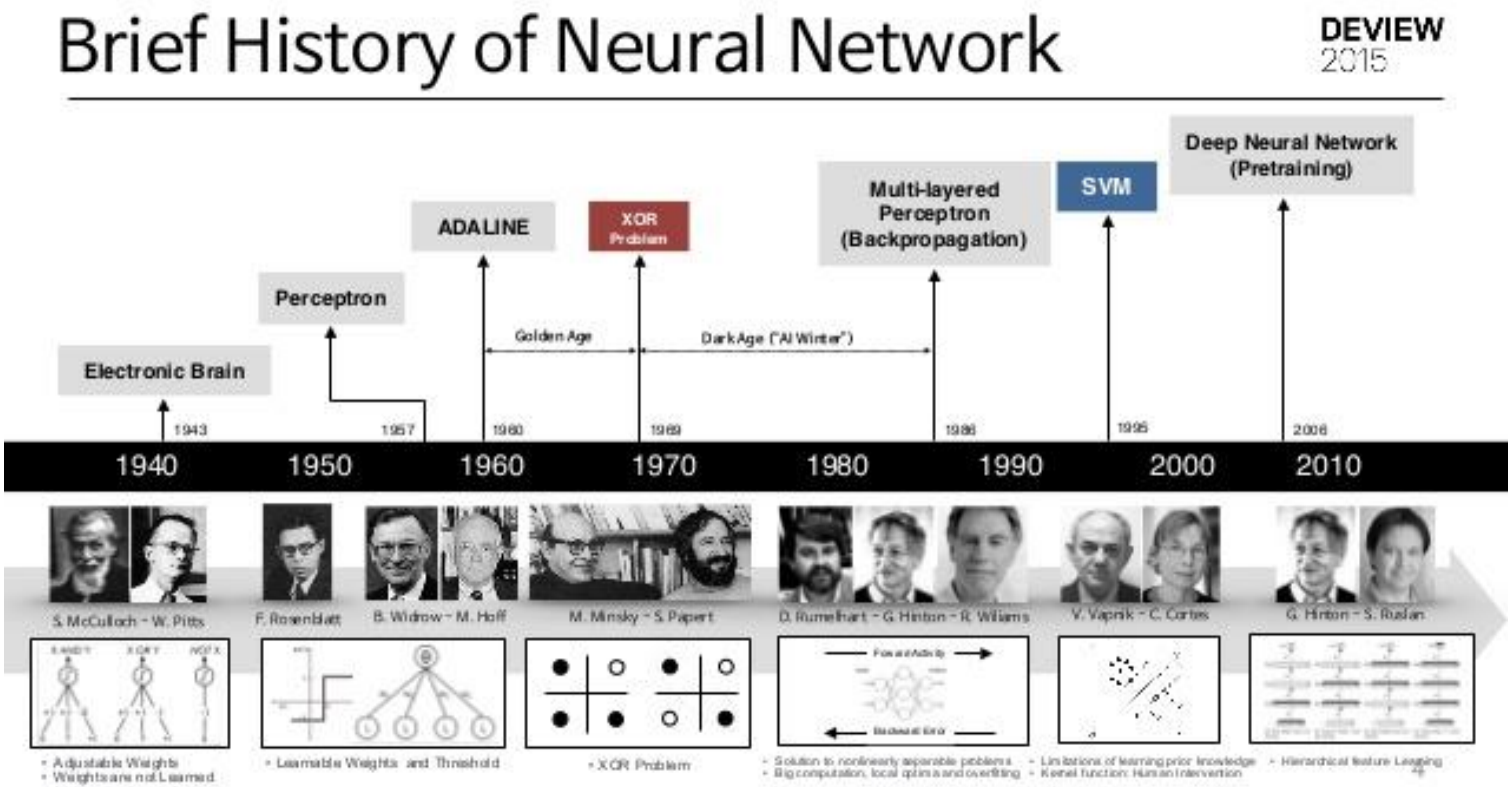
(Goodfellow 2016)

Artificial Neural Networks



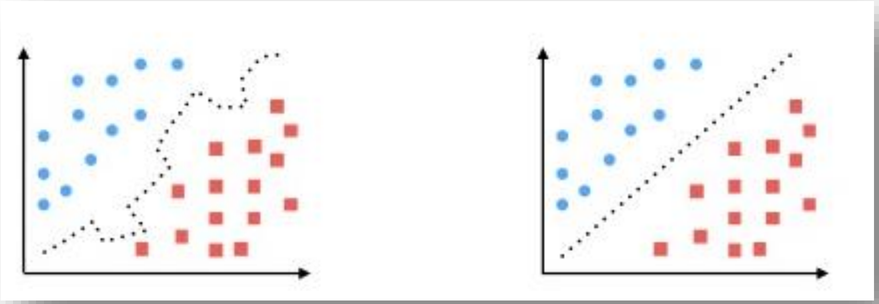
Biological NN	Artificial NN
soma	unit
axon, dendrite	connection
synapse	weight
potential	weighted sum
threshold	bias weight
signal	activation

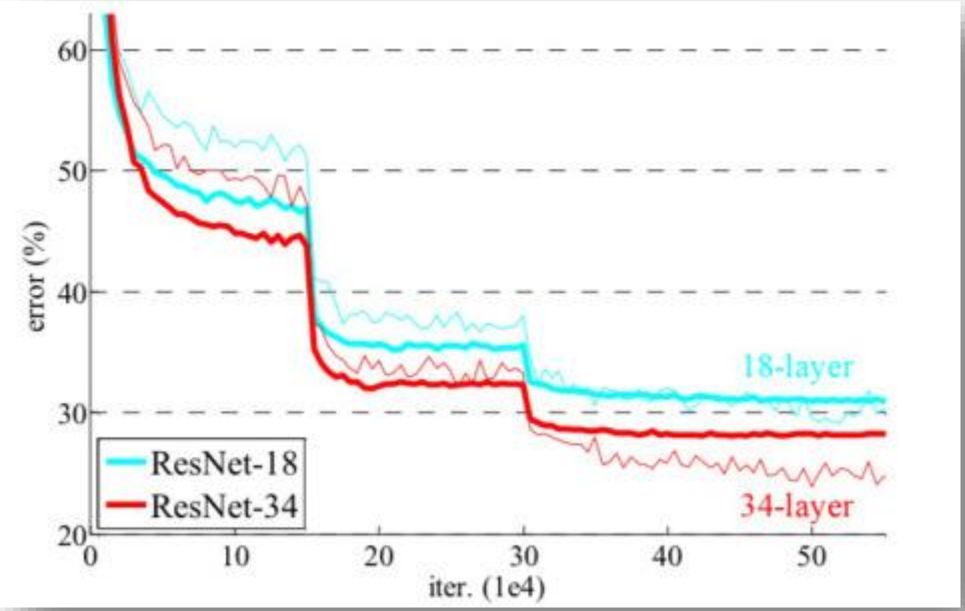
Brief history of DL



Why Today?

Lots of Data

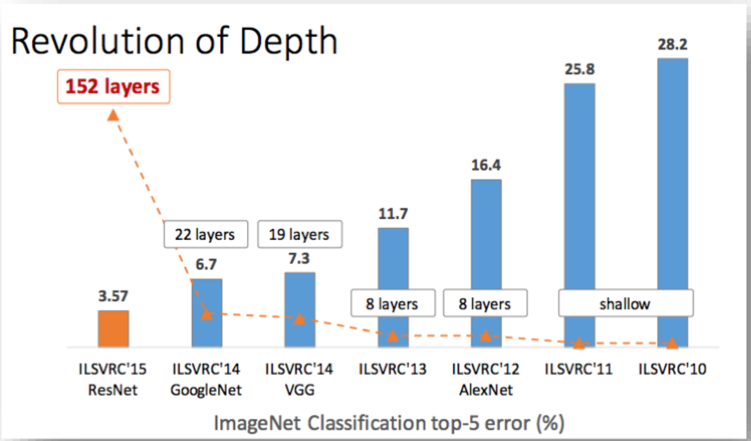


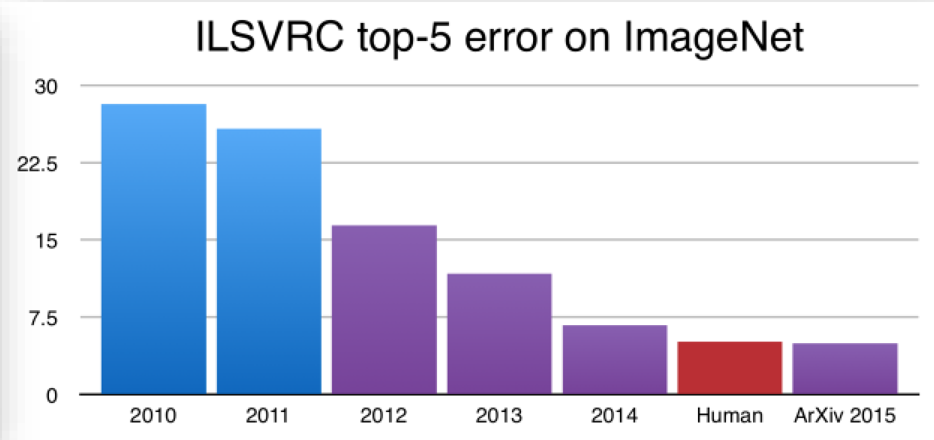


Why Today?

Lots of Data

Deeper Learning



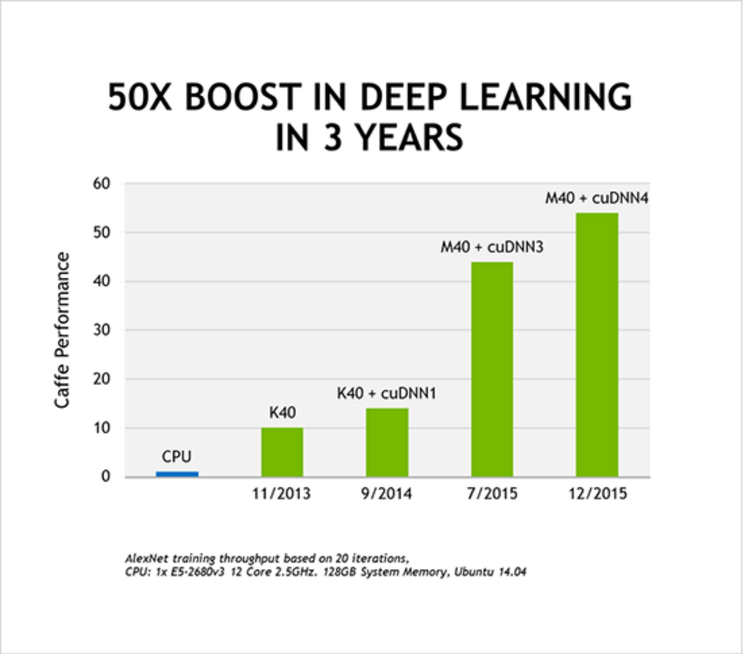


Why Today?

Lots of Data

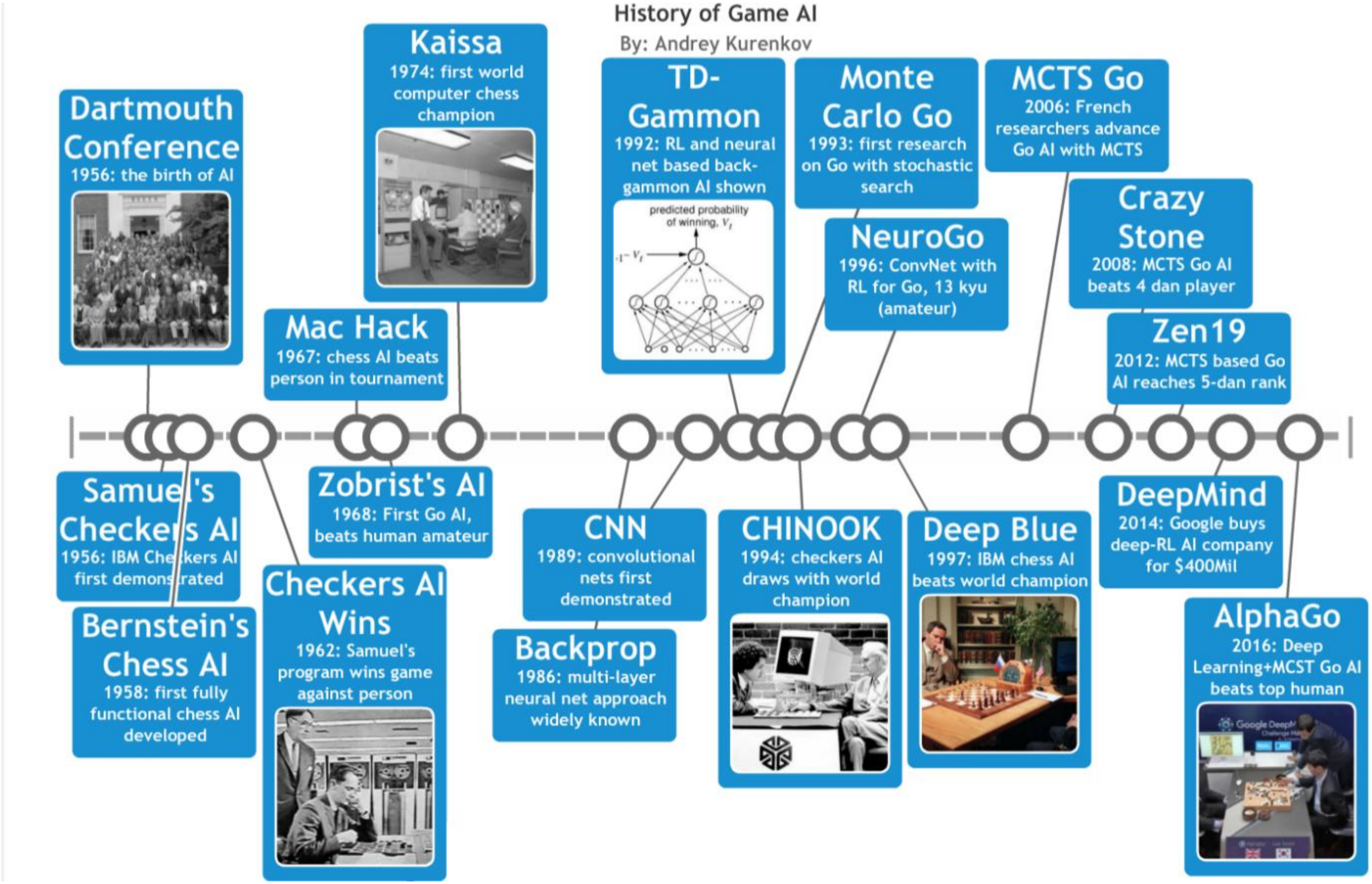
Deep Learning

More Power



https://blogs.nvidia.com/blog/2016/01/12/accelerating-ai-artificial-intelligence-gpus/
https://www.slothparadise.com/what-is-cloud-computing/

Apps: Gaming



History of Game AI
By: Andrey Kurenkov
Dartmouth Conference
1956: the birth of AI
Kaissa
1974: first world computer chess champion
Mac Hack
1967: chess AI beats person in tournament
TD-Gammon
1992: RL and neural net based back-gammon AI shown
predicted probability of winning, V_t
Monte Carlo Go
1993: first research on Go with stochastic search
NeuroGo
1996: ConvNet with RL for Go, 13 kyu (amateur)
MCTS Go
2006: French researchers advance Go AI with MCTS
Crazy Stone
2008: MCTS Go AI beats 4 dan player
Zen19
2012: MCTS based Go AI reaches 5-dan rank
Samuel's Checkers AI
1956: IBM Checkers AI first demonstrated
Bernstein's Chess AI
1958: first fully functional chess AI developed
Checkers AI Wins
1962: Samuel's program wins game against person
Zobrist's AI
1968: First Go AI, beats human amateur
CNN
1989: convolutional nets first demonstrated
Backprop
1986: multi-layer neural net approach widely known
CHINOOK
1994: checkers AI draws with world champion
Deep Blue
1997: IBM chess AI beats world champion
DeepMind
2014: Google buys deep-RL AI company for $400Mil
AlphaGo
2016: Deep Learning+MCST Go AI beats top human

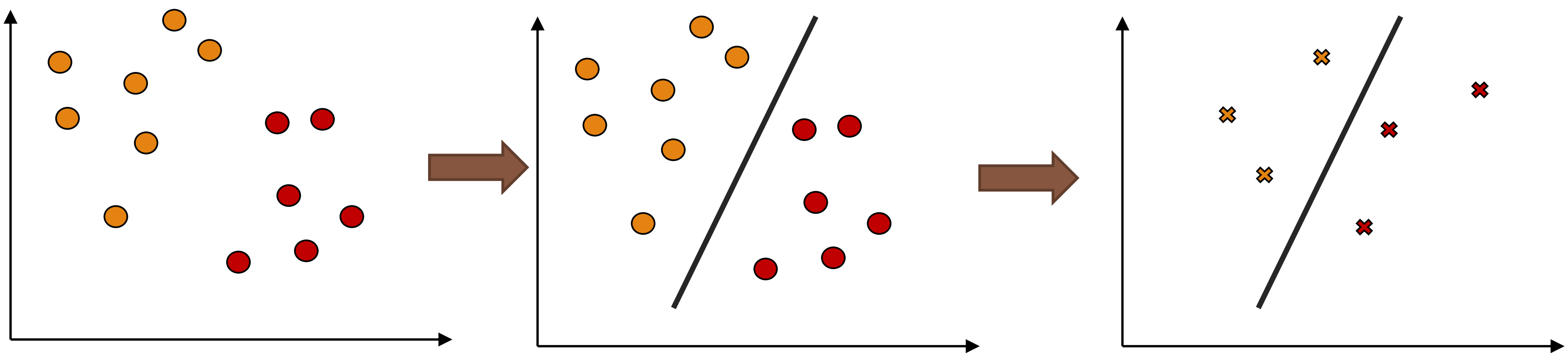
Apps: Self-driving cars

https://www.youtube.com/watch?v=VG68SKoG7vE

Intro to ML

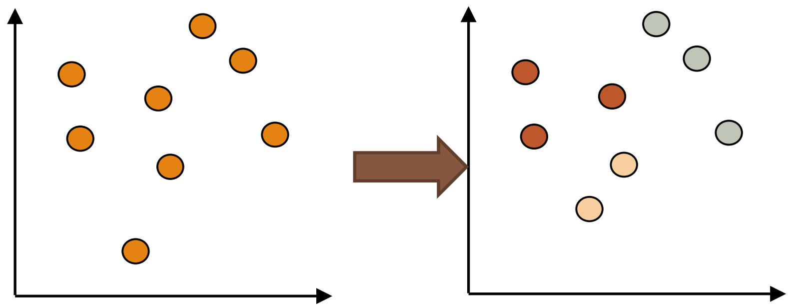
Types of Machine Learning

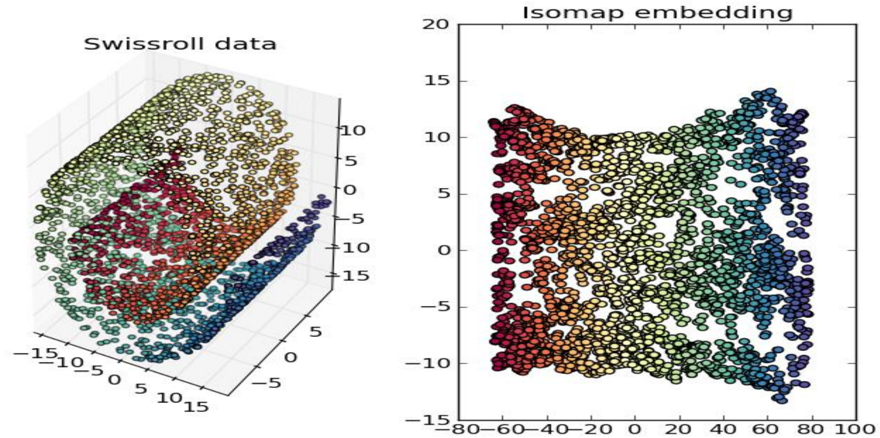
Supervised learning: present example inputs and their desired outputs (**labels**) → learn a general rule that maps inputs to outputs.



Types of Machine Learning

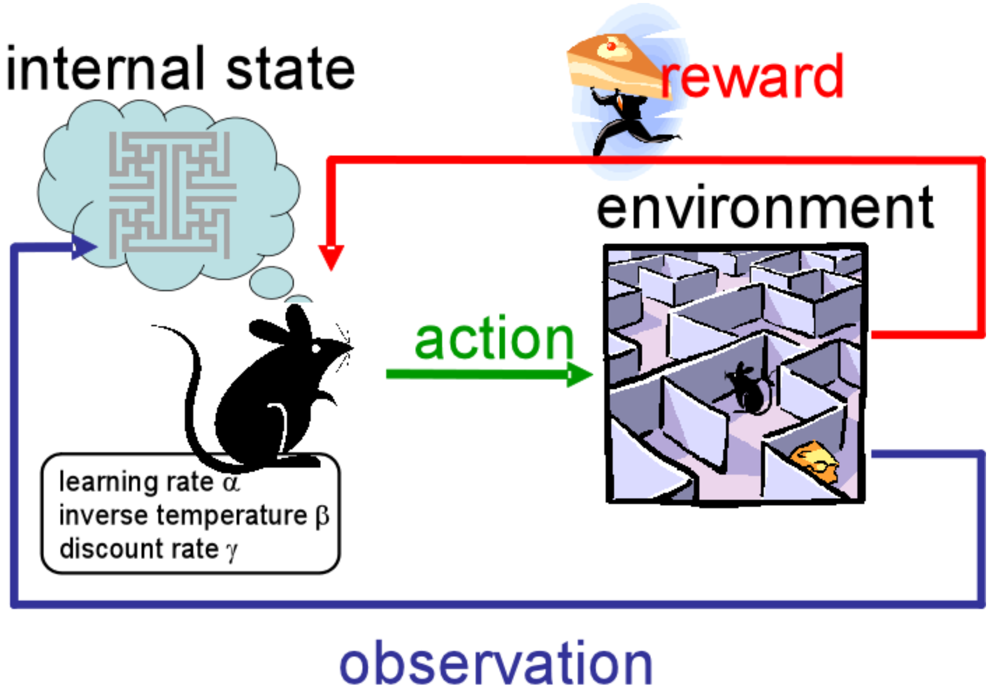
Unsupervised learning: no labels are given → find structure in input.



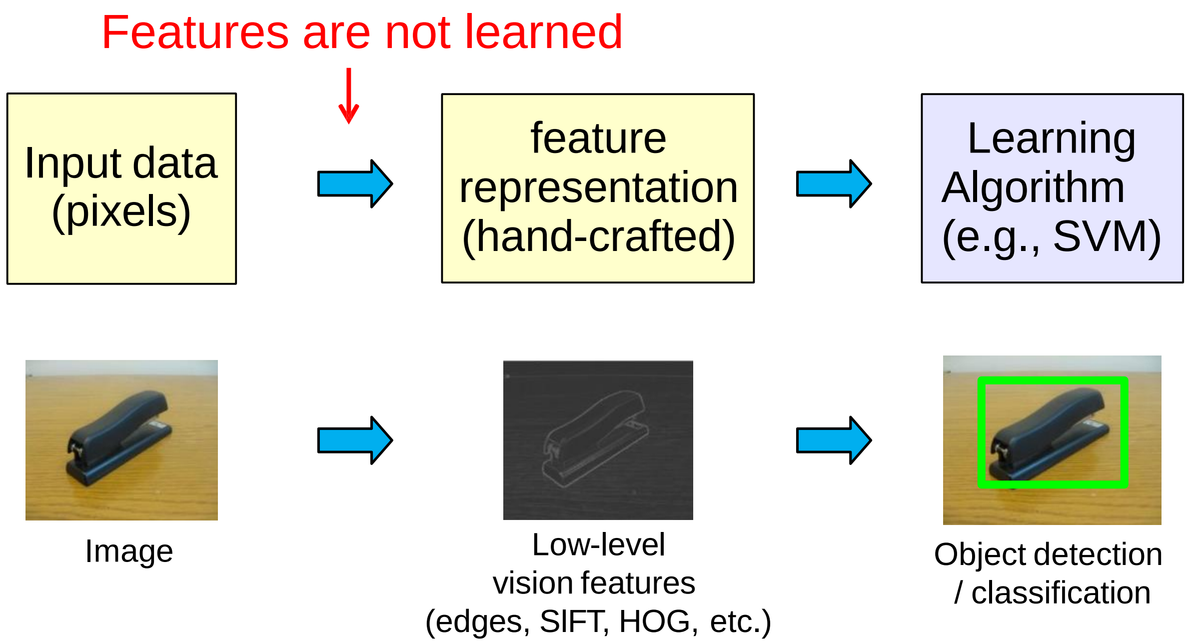


Types of Machine Learning

Reinforcement learning: system interacts with environment and must perform a certain goal without explicitly telling it whether it has come close to its goal or not.



Feature extraction in ML



Feature learning

Pixel 1

Pixel 2

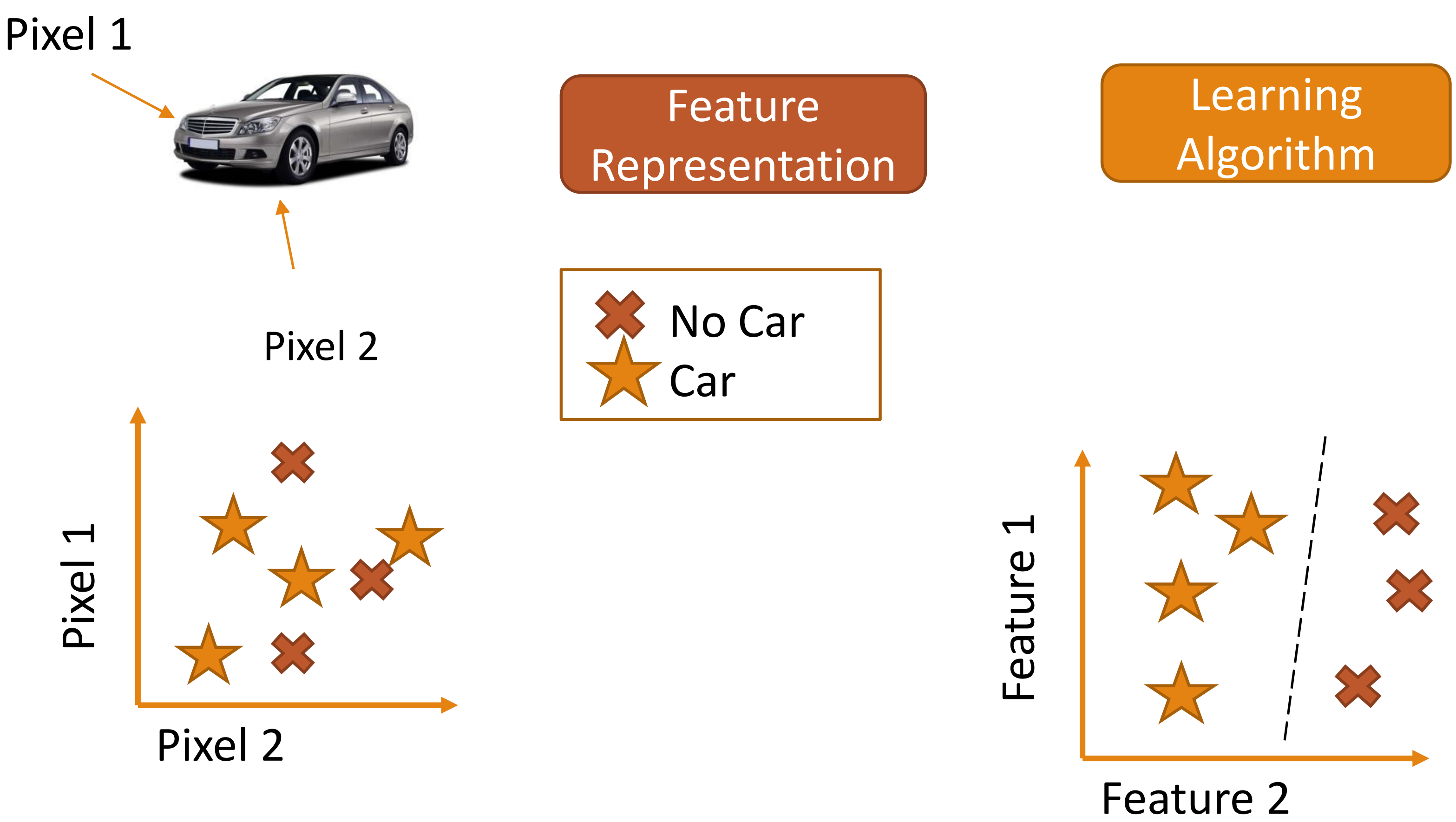
No Car
Car

Learning Algorithm

Pixel 1

Pixel 2

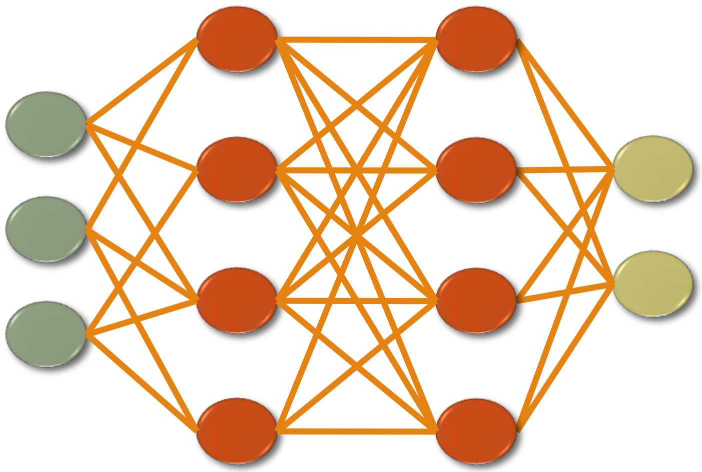
Feature learning



Fundamentals of ANN

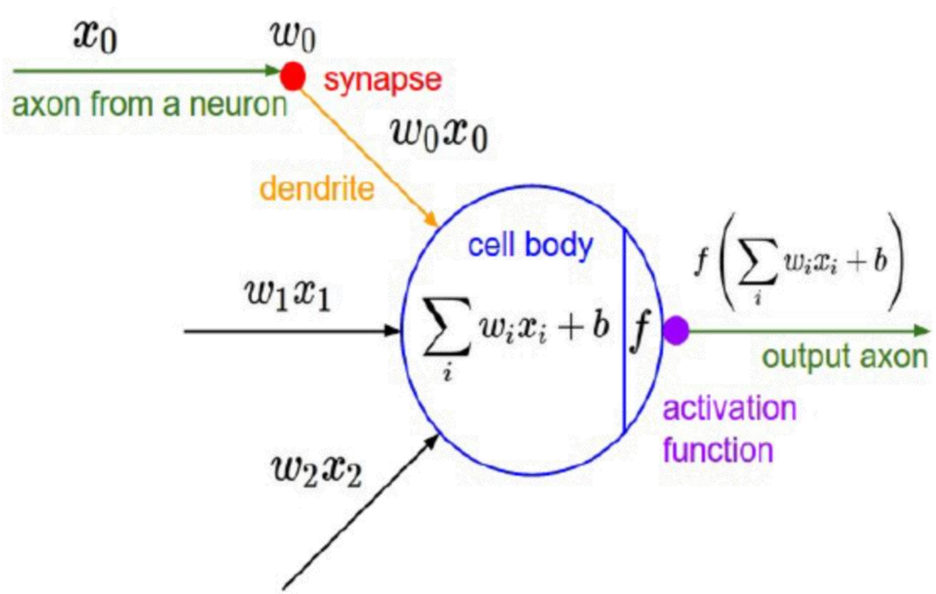
Key components of ANN

- Architecture (input/hidden/output layers)



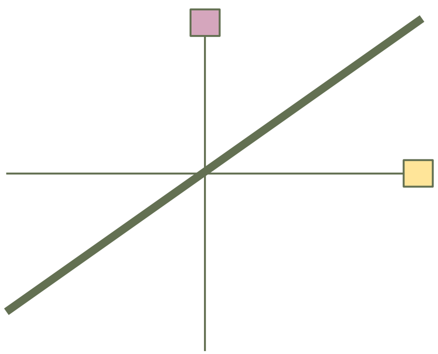
Key components of ANN

- Architecture (input/hidden/output layers)
- Weights

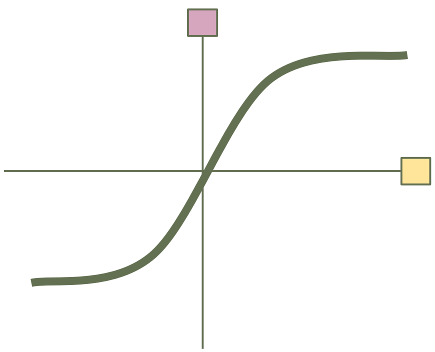


Key components of ANN

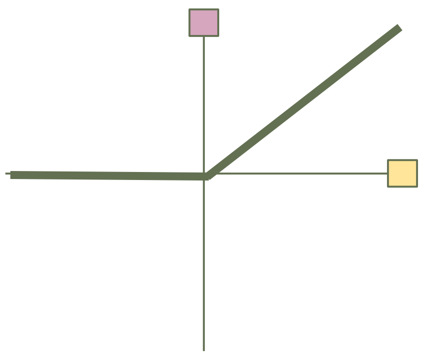
- Architecture (input/hidden/output layers)
- Weights
- Activations



LINEAR



LOGISTIC / SIGMOIDAL / TANH



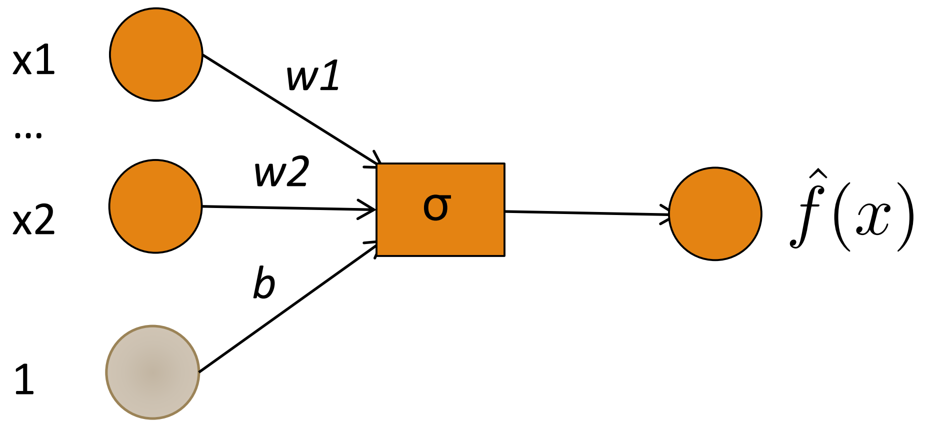
RECTIFIED LINEAR (ReLU)

Perceptron: an early attempt

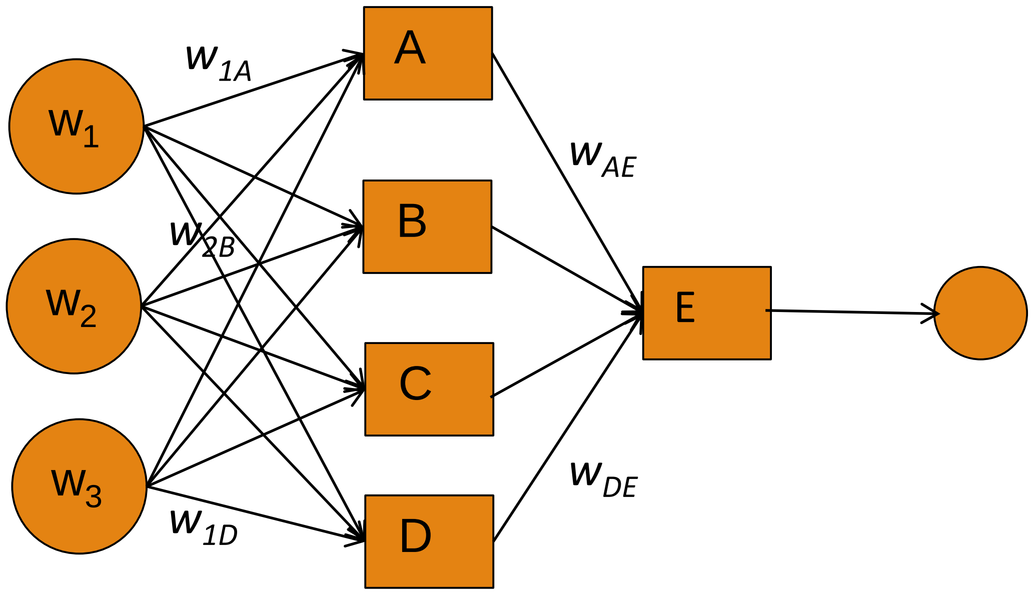
Activation function

$$\hat{f}(x) = \sigma(w \cdot x + b) \qquad \sigma(y) = \begin{cases} 1, & y > 0 \\ 0, & o/w \end{cases}$$

Need to tune w and b



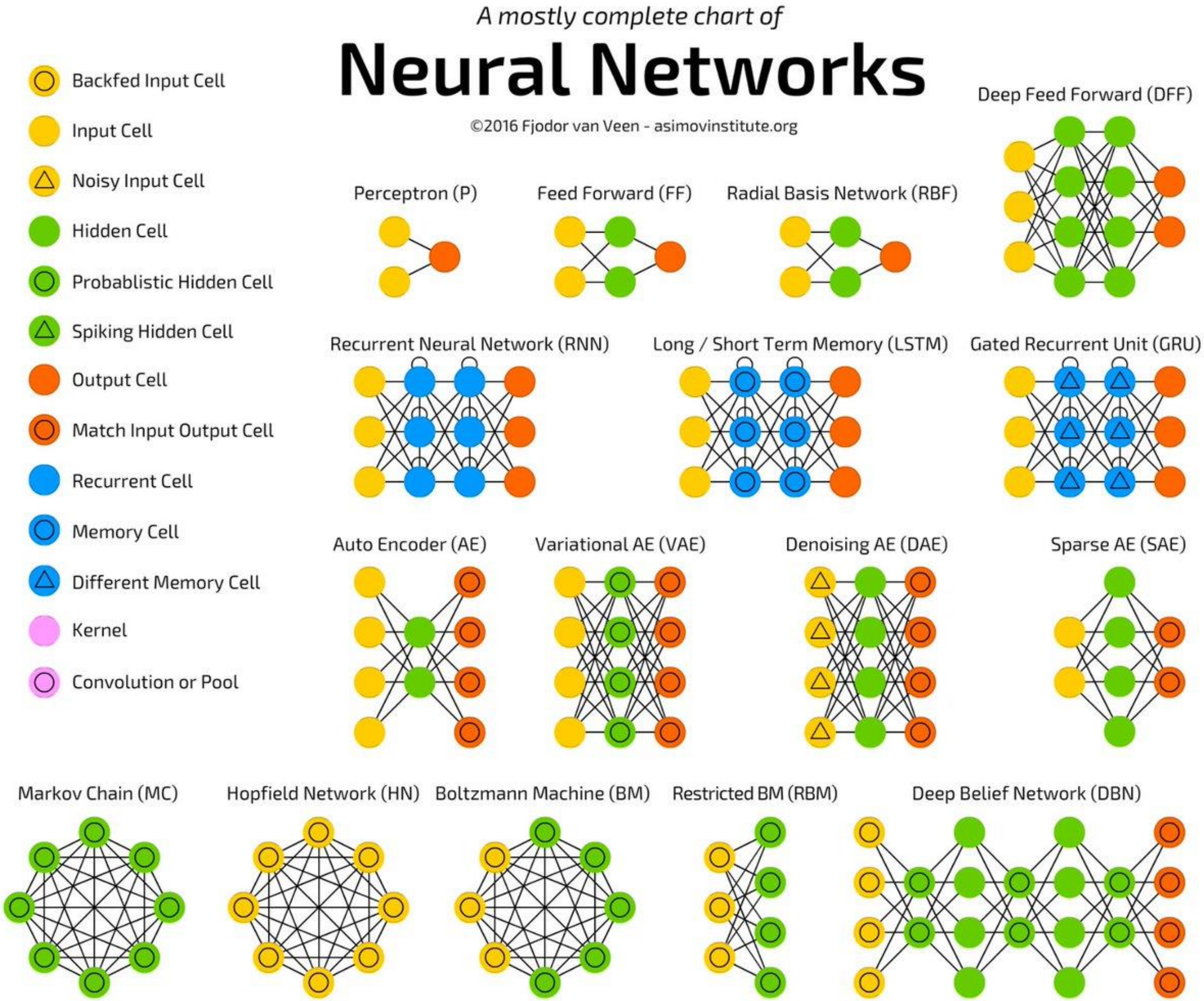
Multilayer perceptron



We just added a neuron layer!

We just introduced non-linearity!

A neuron is of the form **σ(w.x + b)** where **σ** is an *activation* function



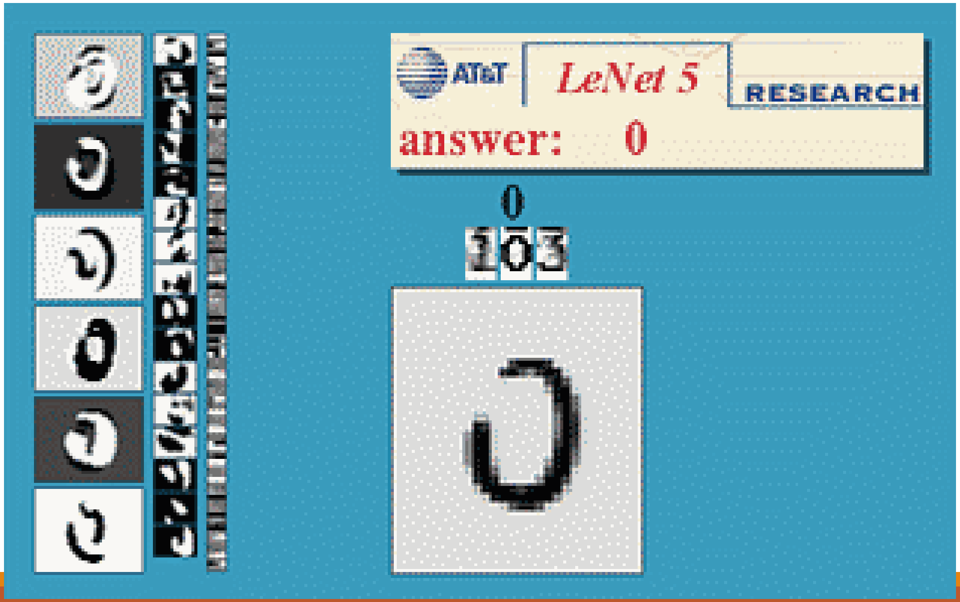
[Sasen Cain (@spectralradius)]

Training & Testing

Training: determine weights

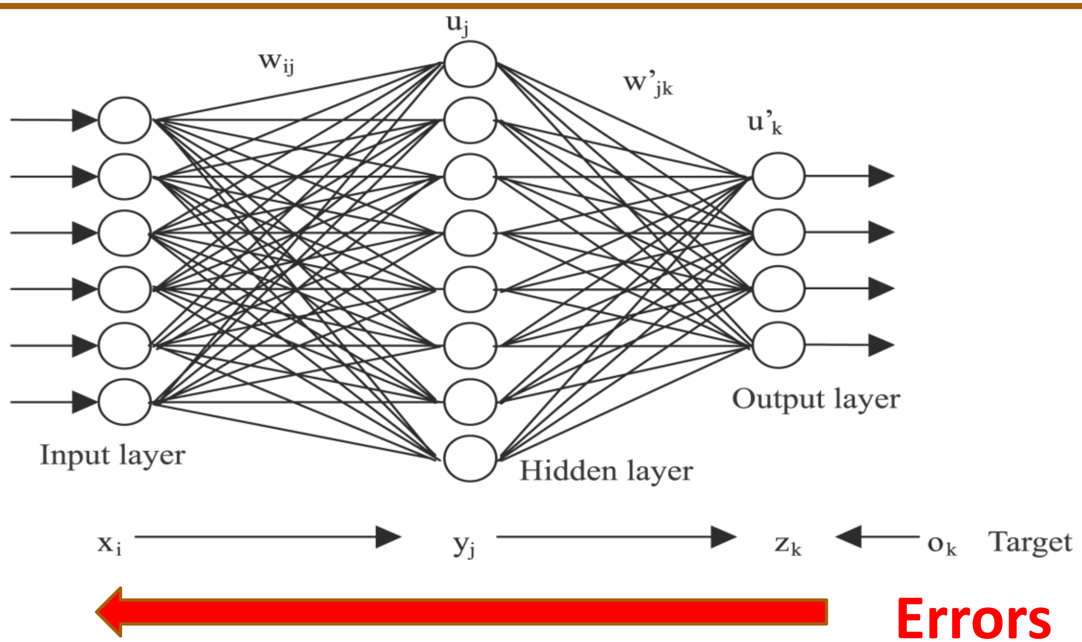
- Supervised: labeled training examples
- Unsupervised: no labels available
- Reinforcement: examples associated with rewards

Testing (Inference): apply weights to new examples



Training DNN

1. Get batch of data
2. Forward through the network -> estimate loss
3. Backpropagate error
4. Update weights based on gradient



BackPropagation

Chain Rule in Gradient Descent: Invented in 1969 by Bryson and Ho

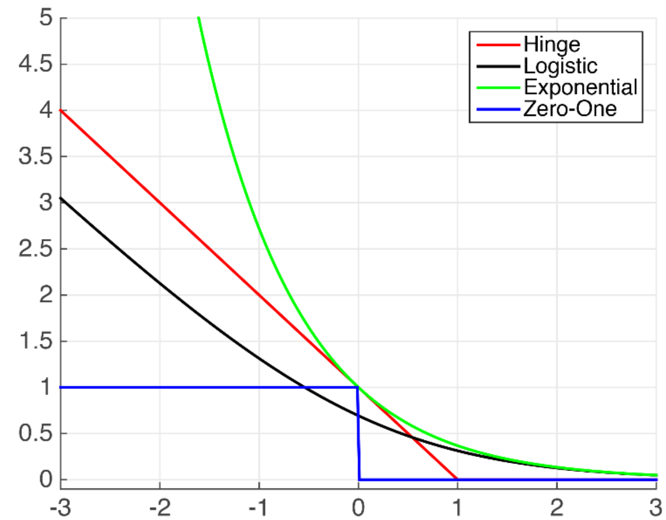
Defining a loss/cost function

Assume a function $J(x,y;\theta) = \frac{1}{2}\sum(y - f(x;\theta))^2$

$$f(x;\theta) = w^T x + b \quad , \quad \theta = \{w, b\}$$

Types of Loss function

- Hinge $J(x,y) = max\{0, 1 - xy\}$
- Exponential $J(x,y) = exp(-xy)$
- Logistic $J(x,y) = log_2(1 + exp(-xy))$



Gradient Descent

- Minimize function J w.r.t. parameters θ

New weights → $$\theta^* = \theta - n * \nabla J(y, x; \theta)$$ ← Gradient

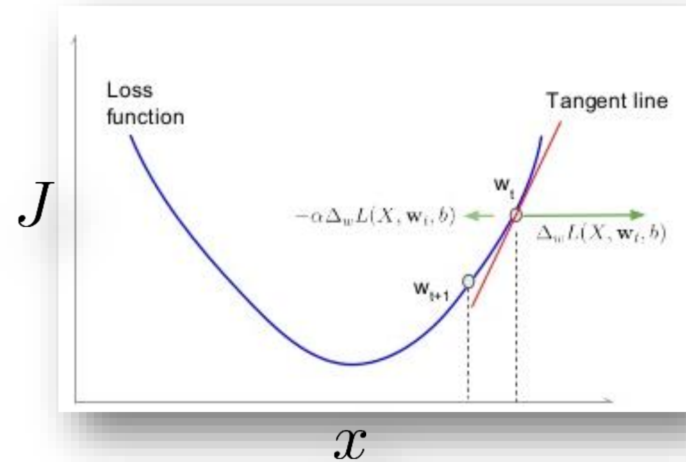
Old weights

Learning rate

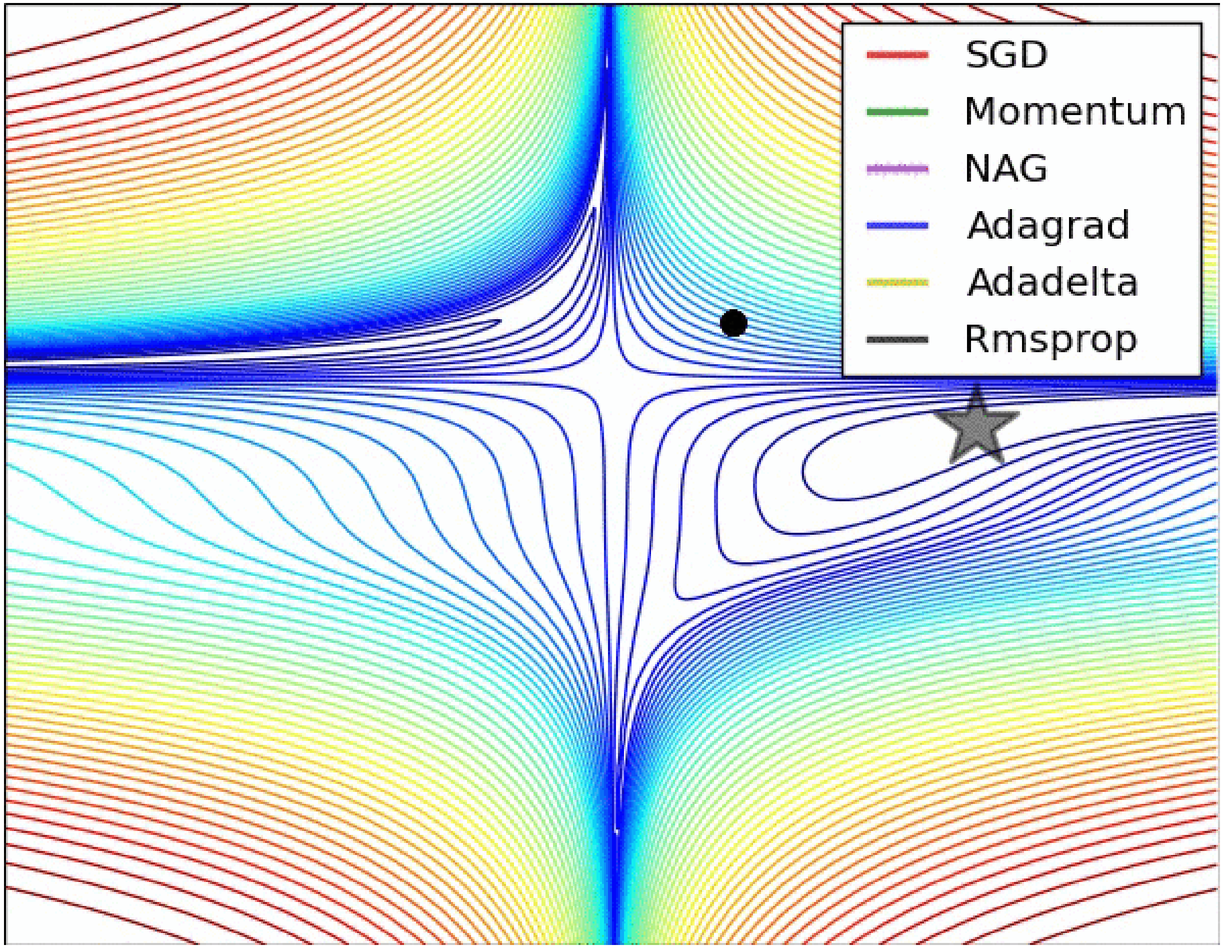
- Gradient

$$\nabla J(x) = \left(\frac{\partial J(x)}{\partial x_1}, \frac{\partial J(x)}{\partial x_2}, \ldots, \frac{\partial J(x)}{\partial x_n}\right)$$

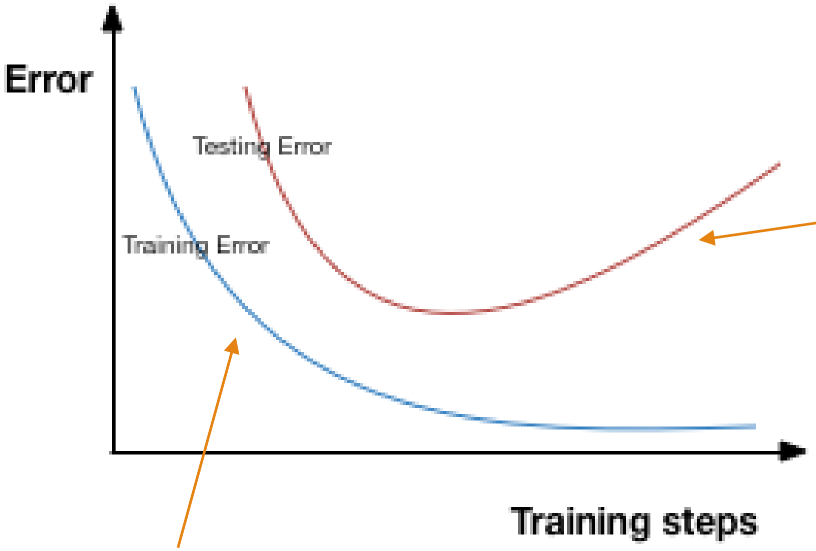
- Chain rule



Visualization



Training Characteristics



Over-fitting

Under-fitting

References

Stephens, Zachary D., et al. "Big data: astronomical or genomical?." *PLoS biology* 13.7 (2015): e1002195.

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." *Nature* 521.7553 (2015): 436-444.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

Kietzmann, Tim Christian, Patrick McClure, and Nikolaus Kriegeskorte. "Deep Neural Networks In Computational Neuroscience." *bioRxiv* (2017): 133504.

Introduction to Deep Learning

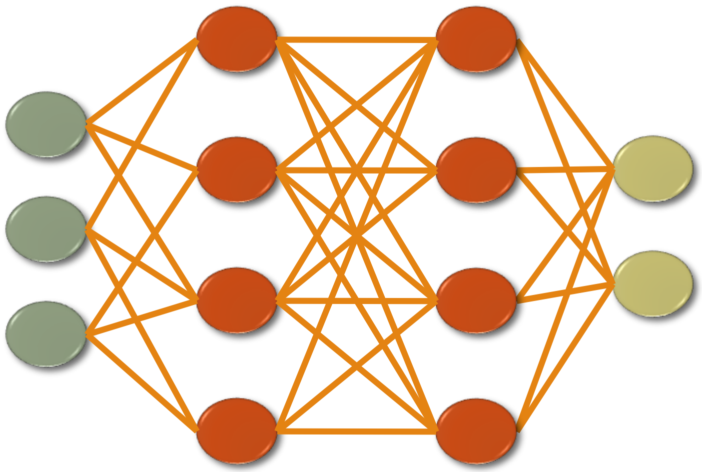
Greg Tsagkatakis

ICS - FORTH

Fundamentals of ANN

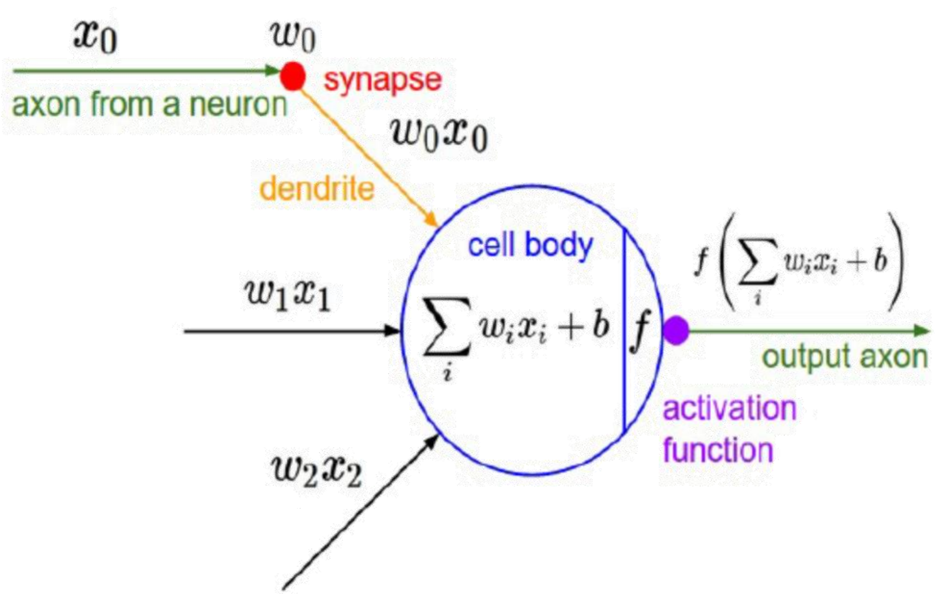
Key components of ANN

- Architecture (input/hidden/output layers)



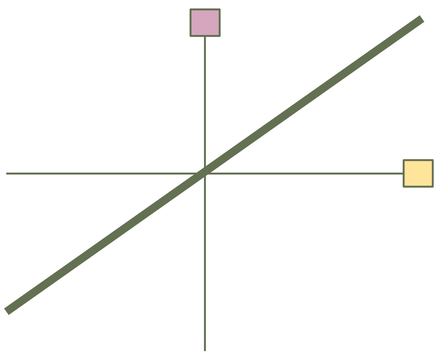
Key components of ANN

- Architecture (input/hidden/output layers)
- Weights

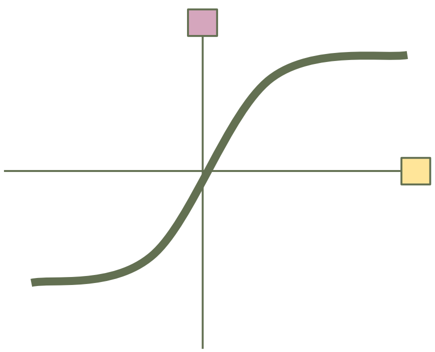


Key components of ANN

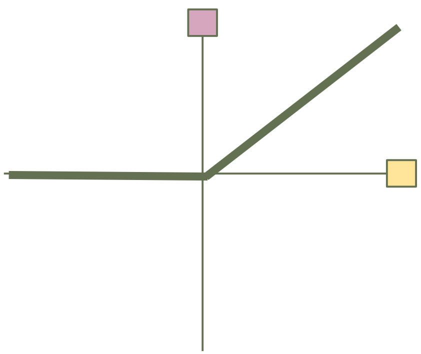
- Architecture (input/hidden/output layers)
- Weights
- Activations



LINEAR



LOGISTIC / SIGMOIDAL / TANH



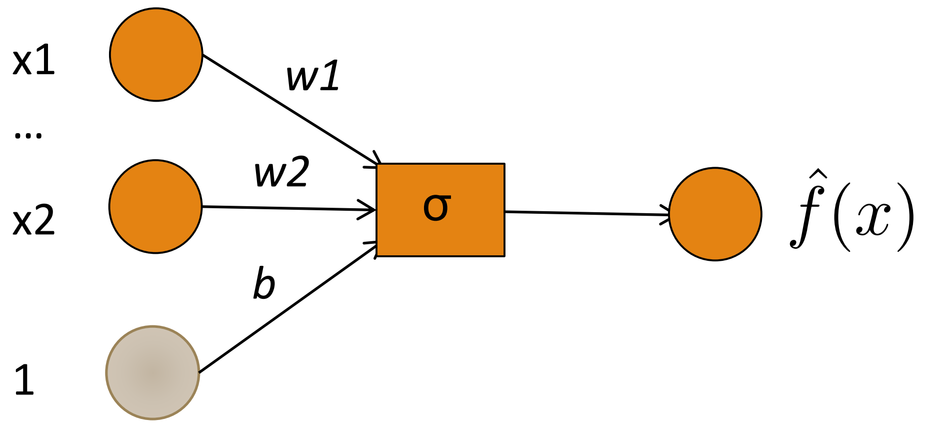
RECTIFIED LINEAR (ReLU)

Perceptron: an early attempt

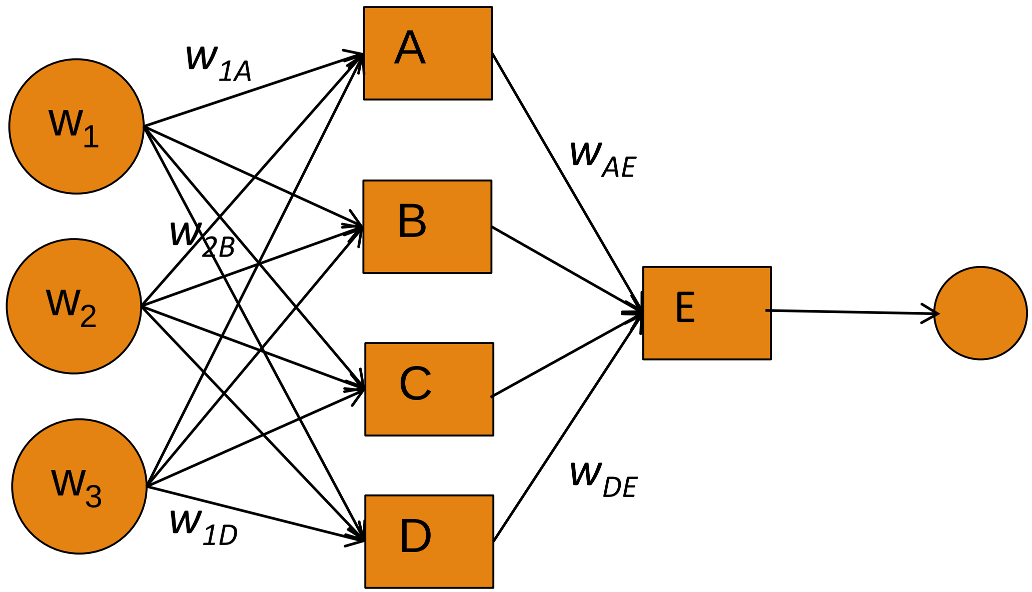
Activation function

$$\hat{f}(x) = \sigma(w \cdot x + b) \qquad \sigma(y) = \begin{cases} 1, & y > 0 \\ 0, & o/w \end{cases}$$

Need to tune w and b



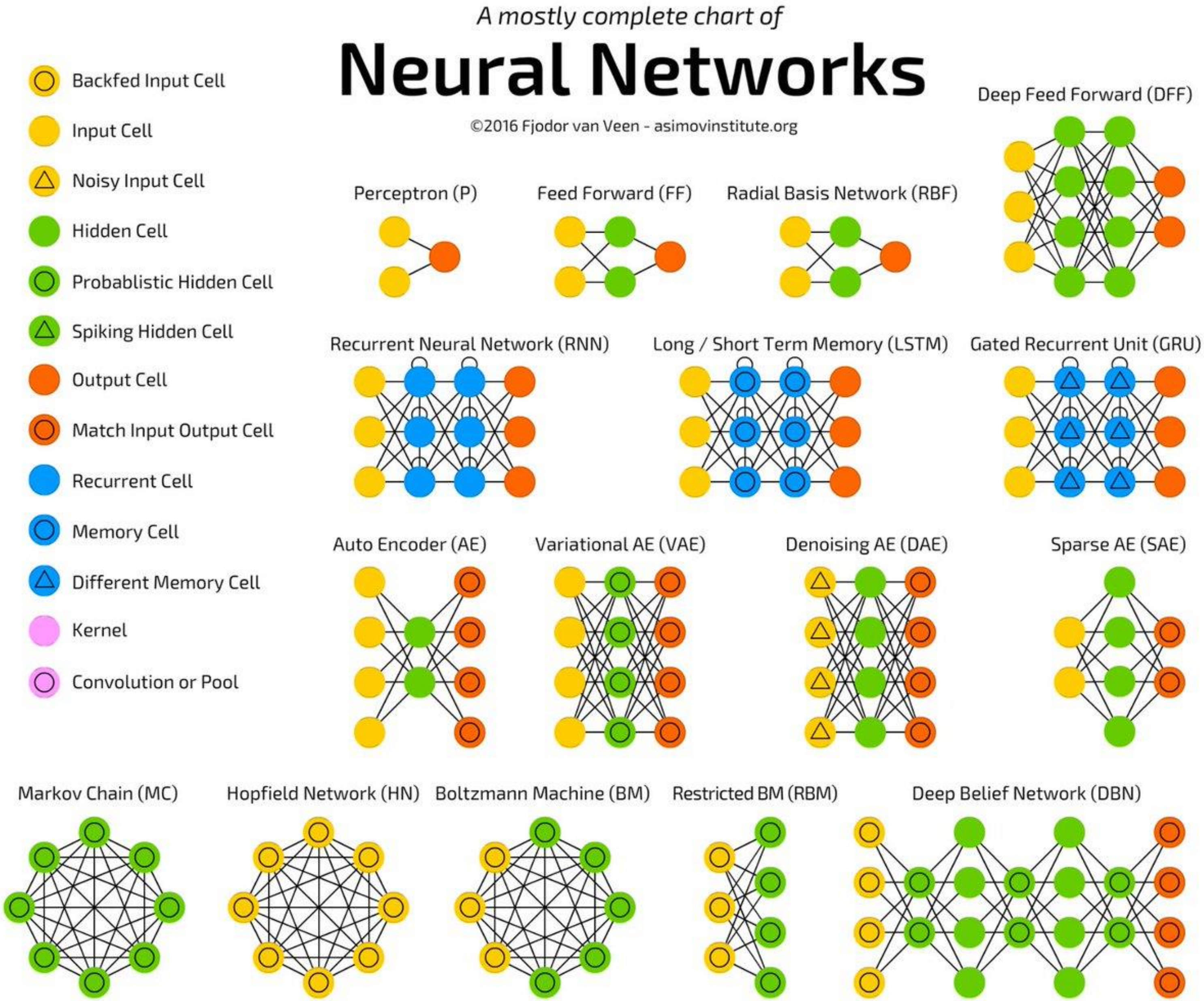
Multilayer perceptron



We just added a neuron layer!

We just introduced non-linearity!

A neuron is of the form **σ(w.x + b)** where **σ** is an *activation* function



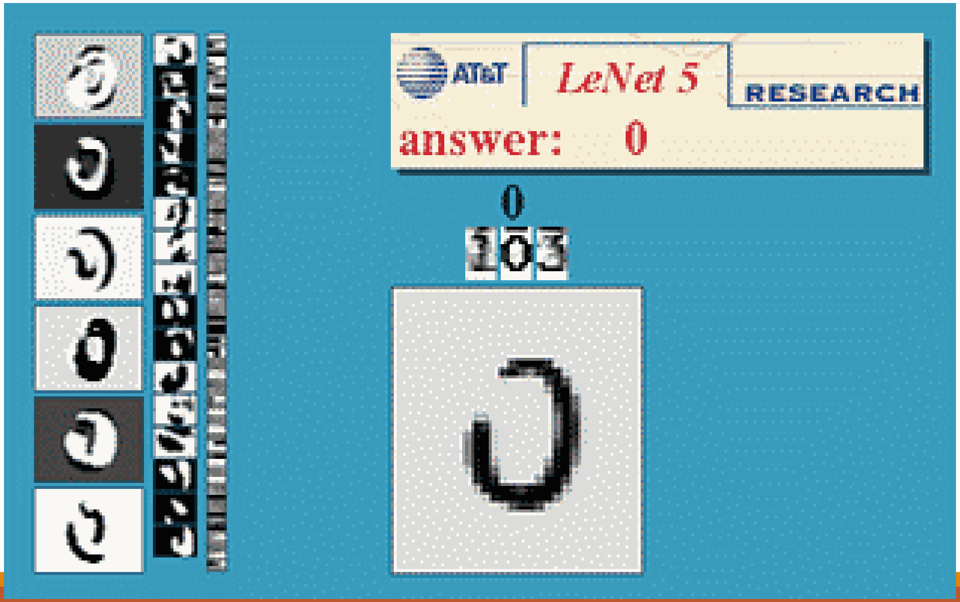
[Sasen Cain (@spectralradius)](#)

Training & Testing

Training: determine weights

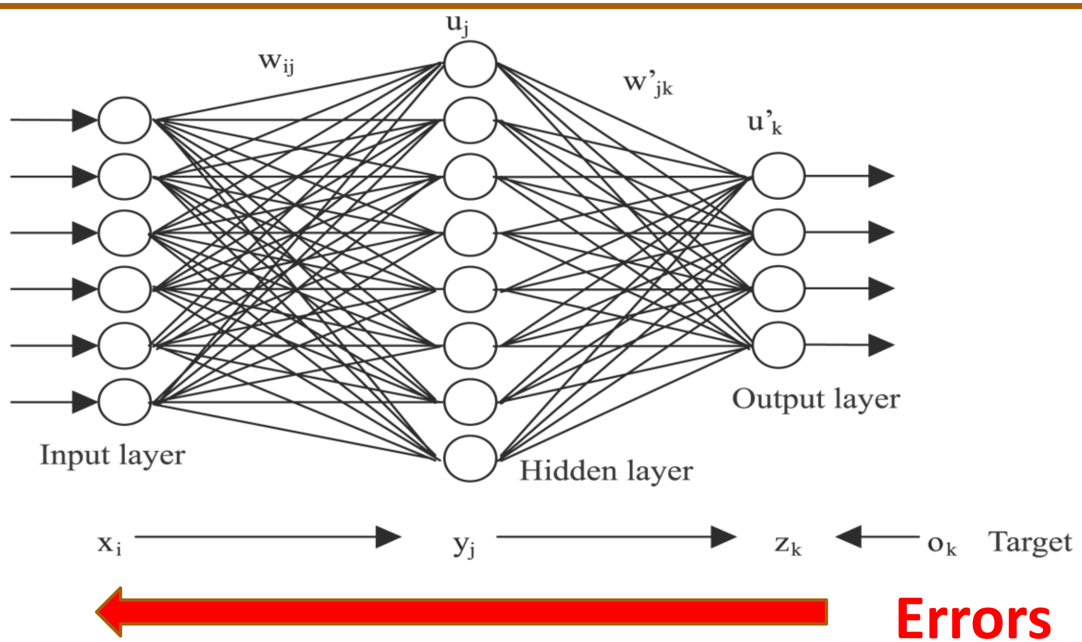
- Supervised: labeled training examples
- Unsupervised: no labels available
- Reinforcement: examples associated with rewards

Testing (Inference): apply weights to new examples



Training DNN

1. Get batch of data
2. Forward through the network -> estimate loss
3. Backpropagate error
4. Update weights based on gradient



BackPropagation

Chain Rule in Gradient Descent: Invented in 1969 by Bryson and Ho

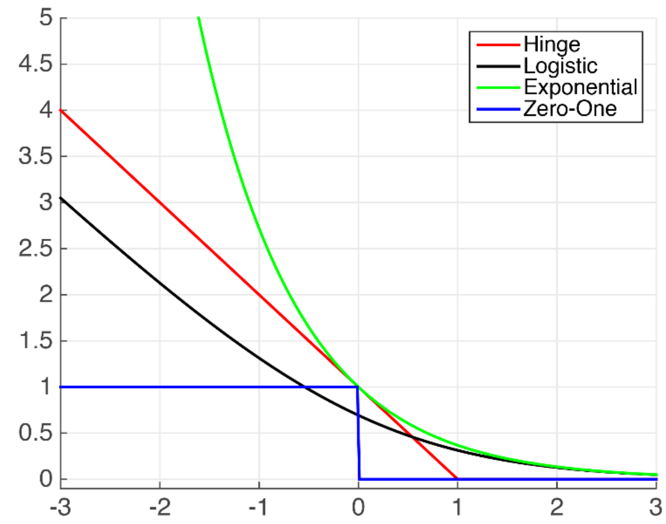
Defining a loss/cost function

Assume a function $$J(x, y; \theta) = \frac{1}{2}\sum(y - f(x;\theta))^2$$

$$f(x;\theta) = w^T x + b \quad , \quad \theta = \{w, b\}$$

Types of Loss function

- Hinge $J(x, y) = max\{0, 1 - xy\}$
- Exponential $J(x, y) = exp(-xy)$
- Logistic $J(x, y) = log_2(1 + exp(-xy))$



Gradient Descent

➢ Minimize function J w.r.t. parameters θ

New weights $$\theta^* = \theta - n * \nabla J(y, x; \theta)$$ Gradient

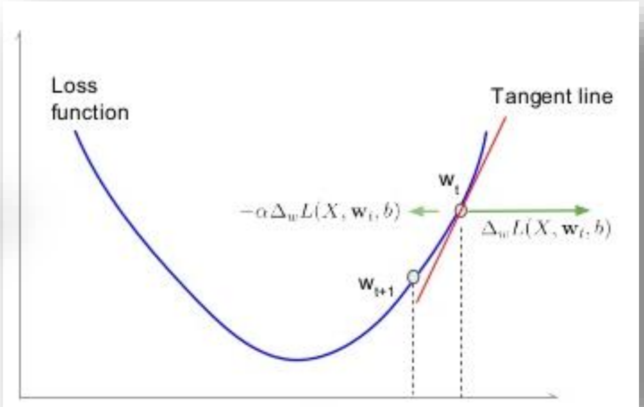
Old weights   Learning rate

- Gradient

$$\nabla J(x) = \left(\frac{\partial J(x)}{\partial x_1}, \frac{\partial J(x)}{\partial x_2}, \ldots, \frac{\partial J(x)}{\partial x_n}\right)$$

- Chain rule

J



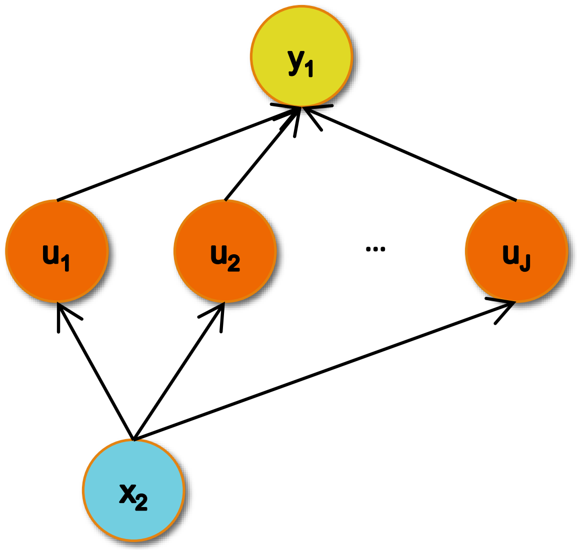
x

BackProp

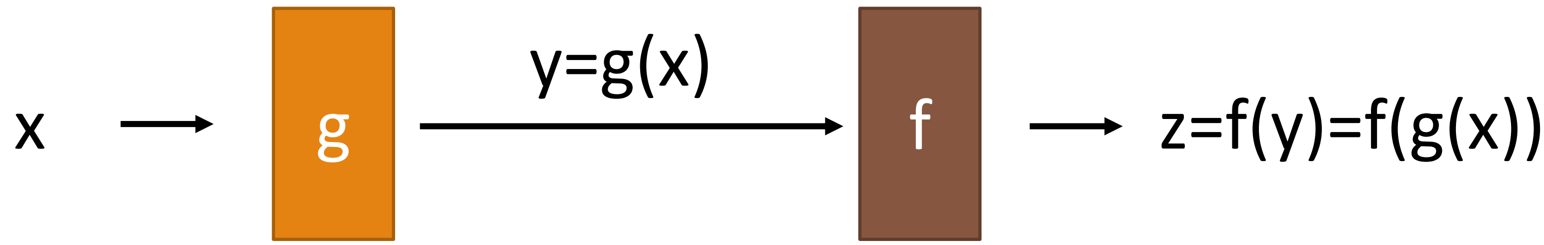
Given: $\boldsymbol{y} = g(\boldsymbol{u})$ and $\boldsymbol{u} = h(\boldsymbol{x})$.

Chain Rule:

$$\frac{dy_i}{dx_k} = \sum_{j=1}^{J} \frac{dy_i}{du_j} \frac{du_j}{dx_k}, \quad \forall i, k$$



BackProp



Chain rule:

- **Single variable**

$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}.$$

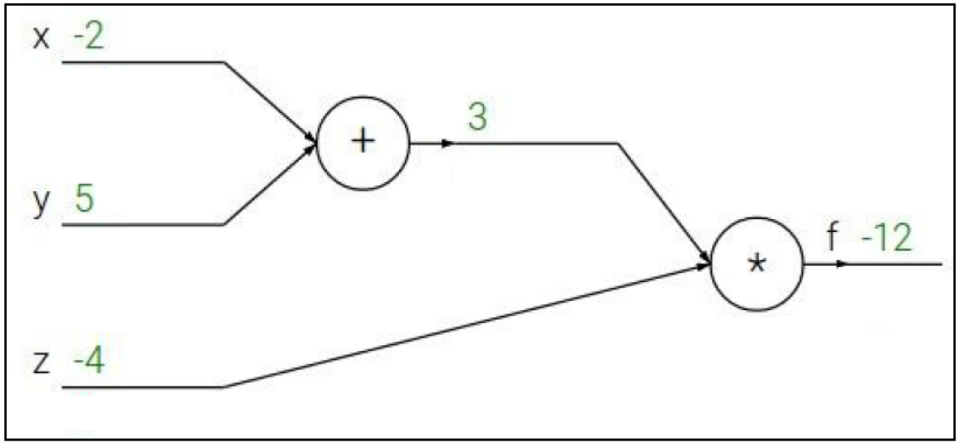
- **Multiple variables**

$$\frac{\partial z}{\partial x_i} = \sum_j \frac{\partial z}{\partial y_j}\frac{\partial y_j}{\partial x_i}.$$

Backpropagation: a simple example

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4



Backpropagation: a simple example

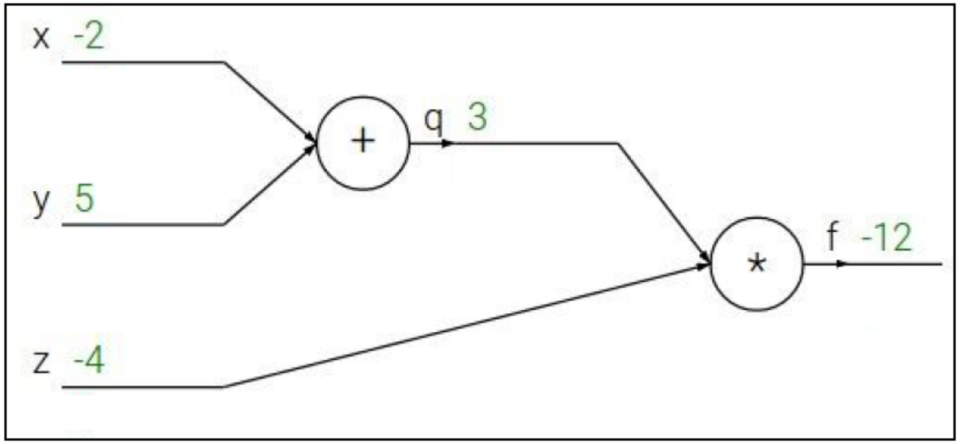
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



Backpropagation: a simple example

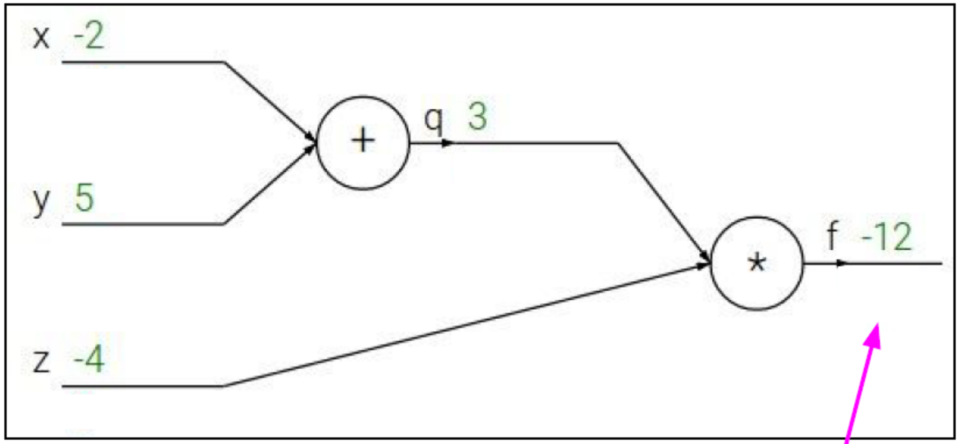
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\frac{\partial f}{\partial f}$$

Backpropagation: a simple example

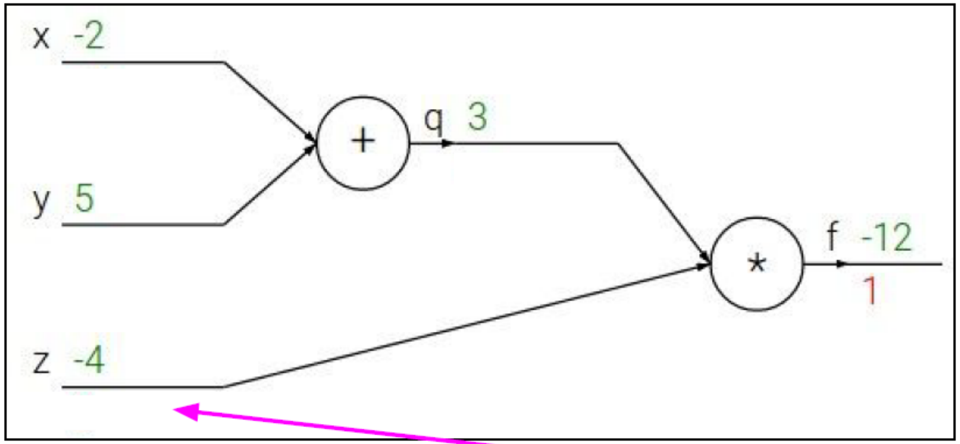
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\frac{\partial f}{\partial z}$$

Backpropagation: a simple example

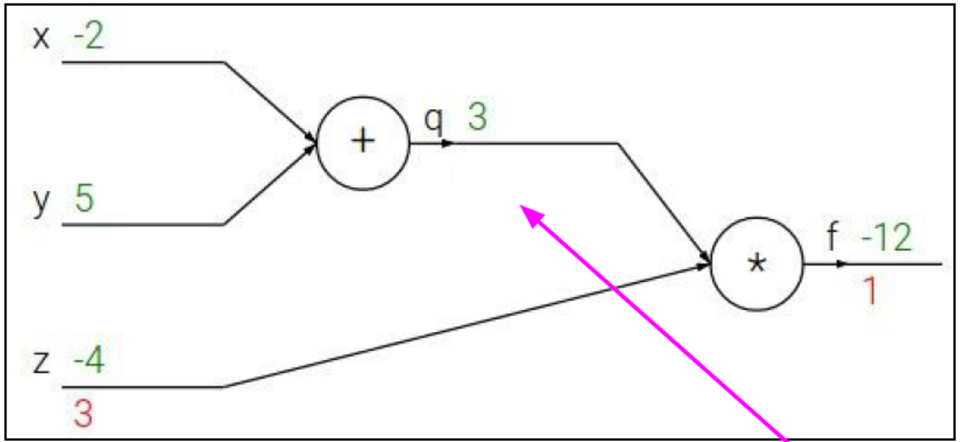
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\frac{\partial f}{\partial q}$$

Backpropagation: a simple example

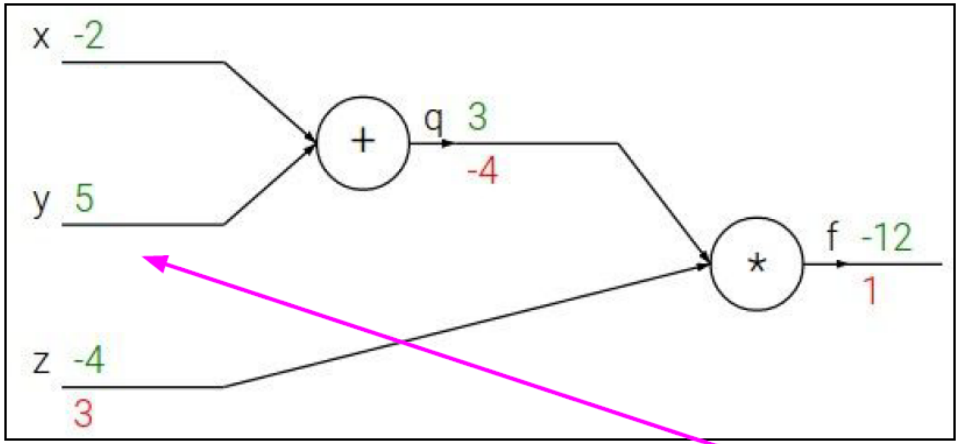
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\frac{\partial f}{\partial y}$$

Backpropagation: a simple example

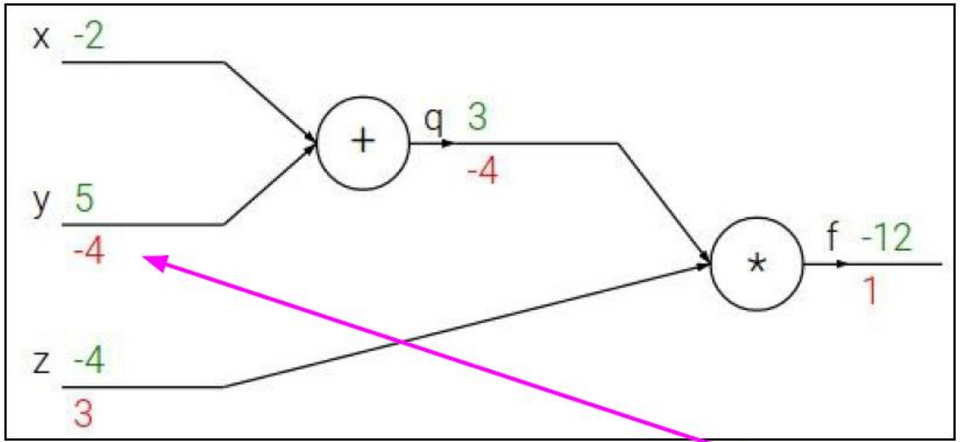
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

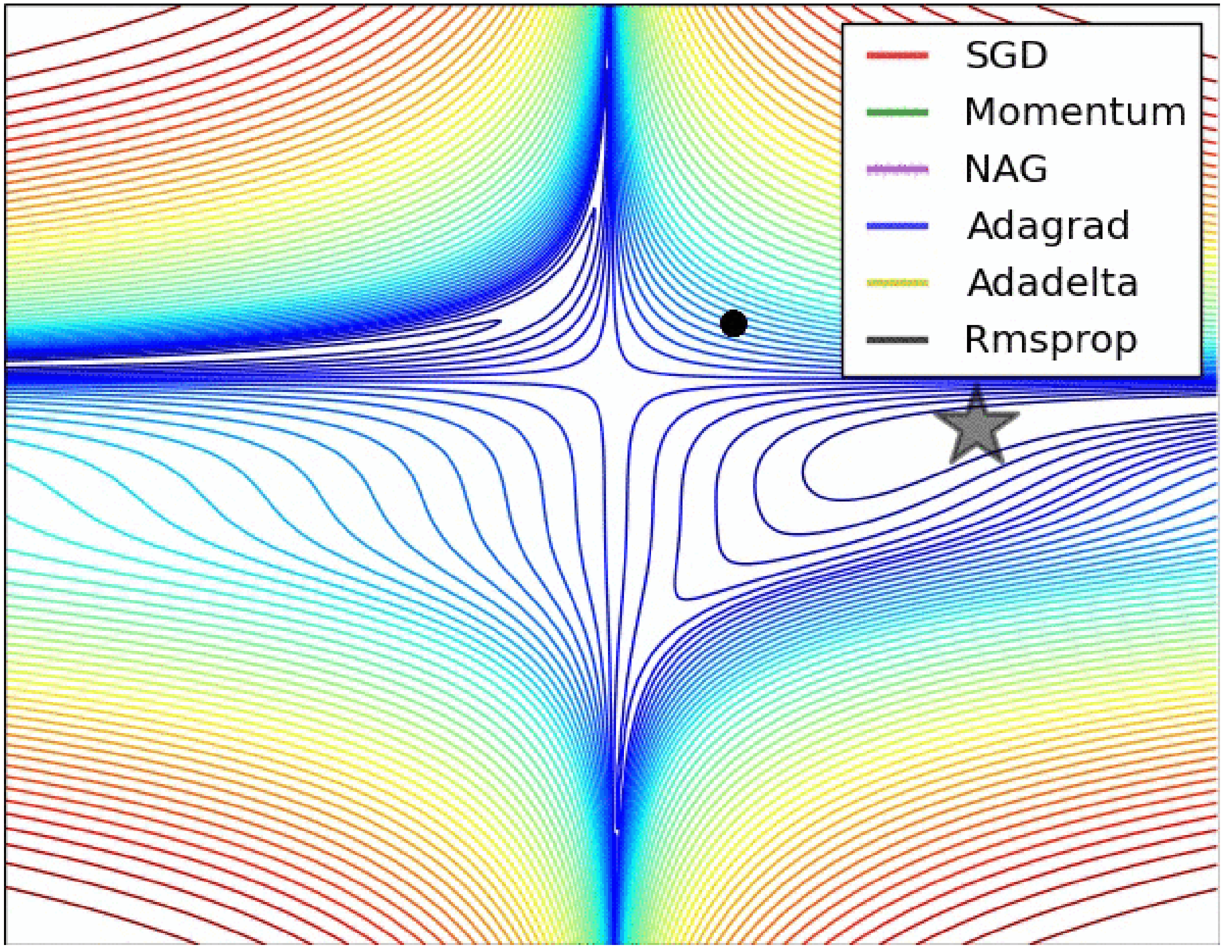


$$\frac{\partial f}{\partial y}$$

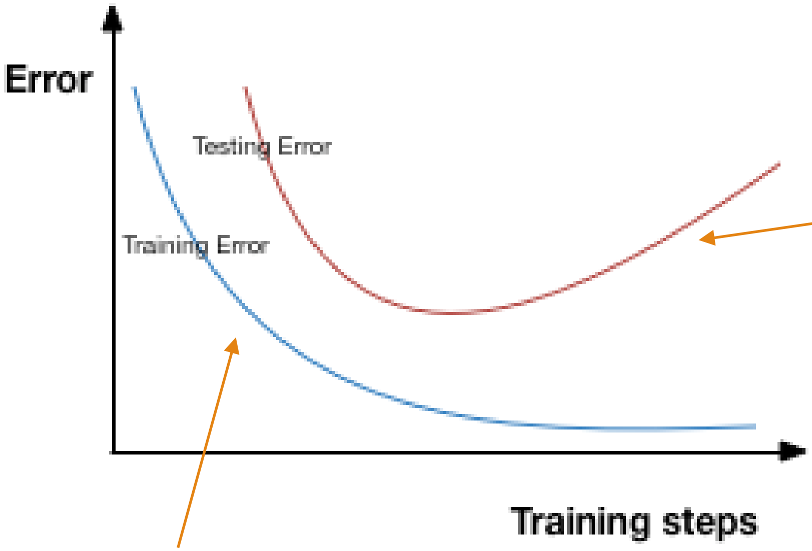
Chain rule:

$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial y}$$

Visualization



Training Characteristics



Over-fitting

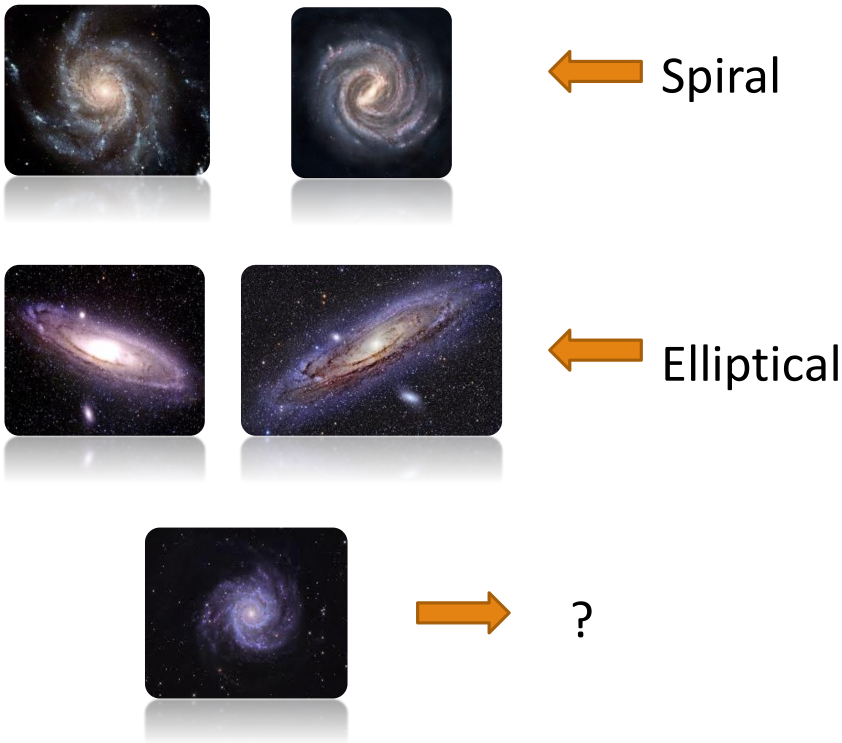
Under-fitting

Supervised Learning

Supervised Learning

Data
Labels

Model
Prediction



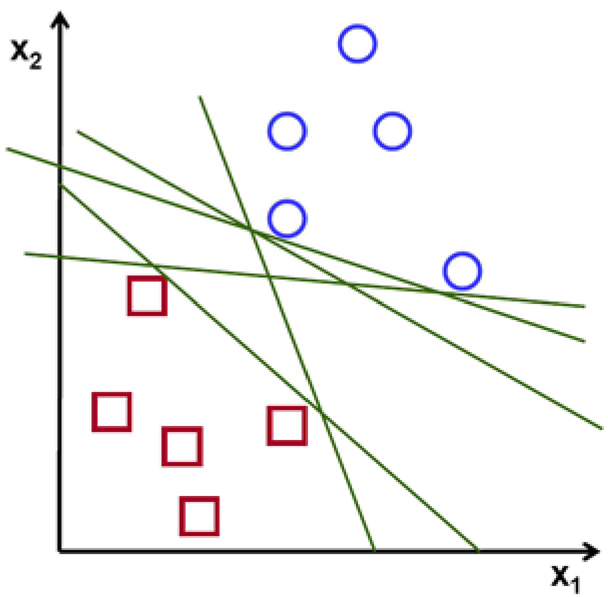
Exploiting prior knowledge

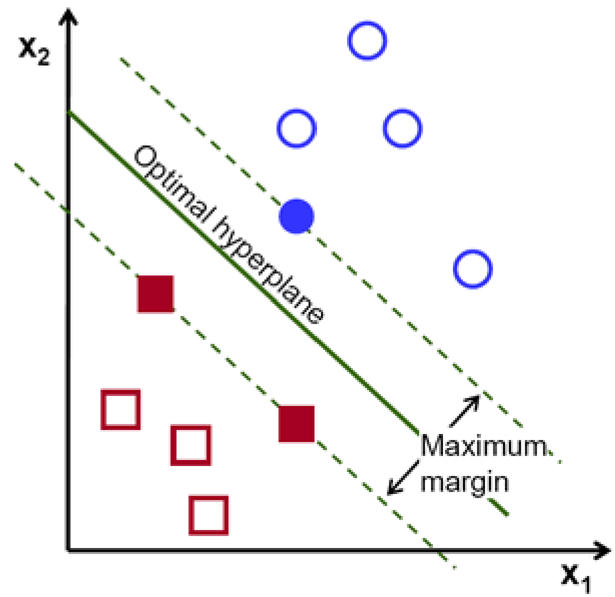
- Expert users
- Crowdsourcing
- Other instruments

State-of-the-art (before Deep Learning)

Support Vector Machines

- Binary classification

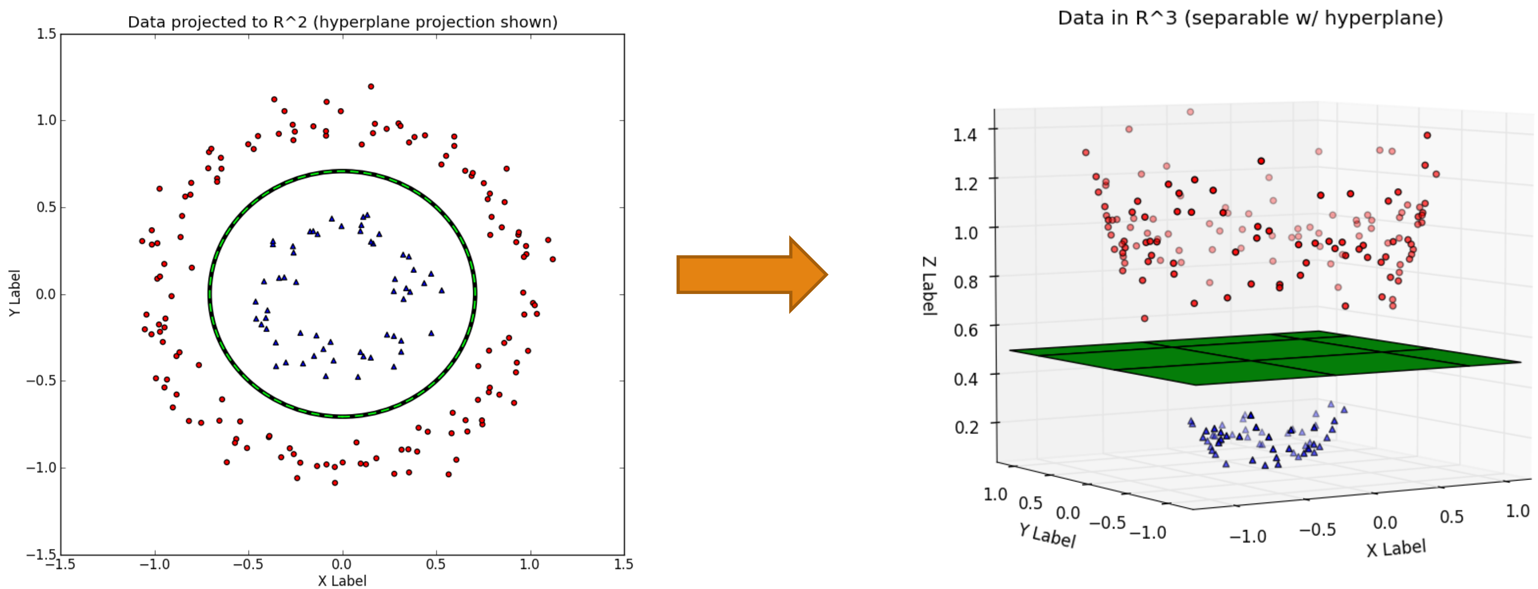




State-of-the-art (before Deep Learning)

Support Vector Machines

- Binary classification
- Kernels <-> non-linearities



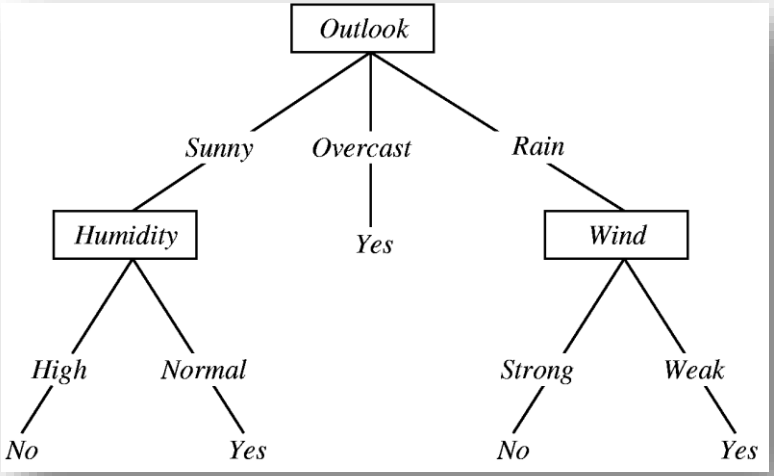
State-of-the-art (before Deep Learning)

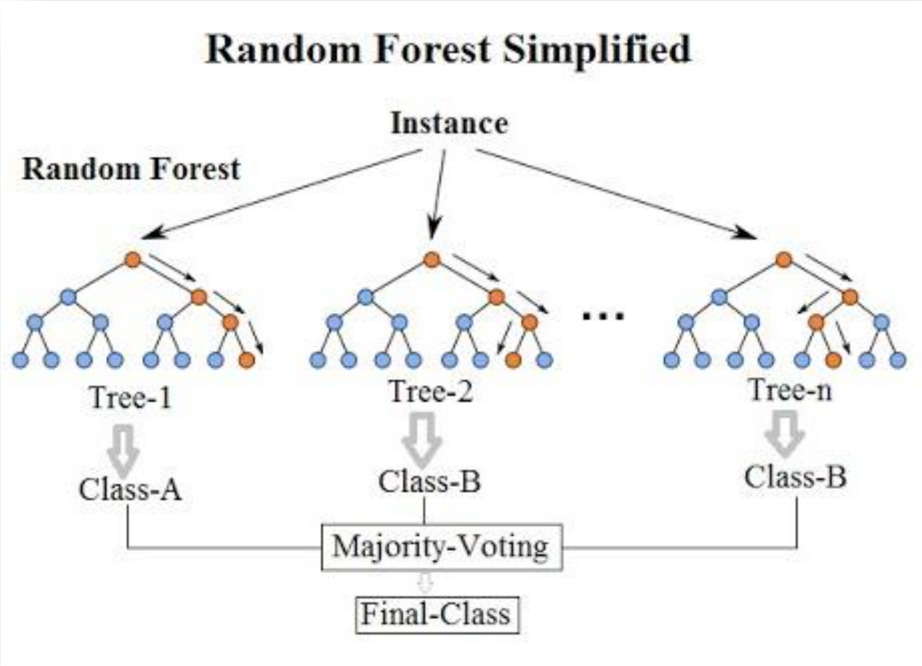
Support Vector Machines

- Binary classification
- Kernels <-> non-linearities

Random Forests

- Multi-class classification





State-of-the-art (before Deep Learning)

Support Vector Machines

- Binary classification
- Kernels <-> non-linearities

Random Forests

- Multi-class classification

Markov Chains/Fields

- Temporal data

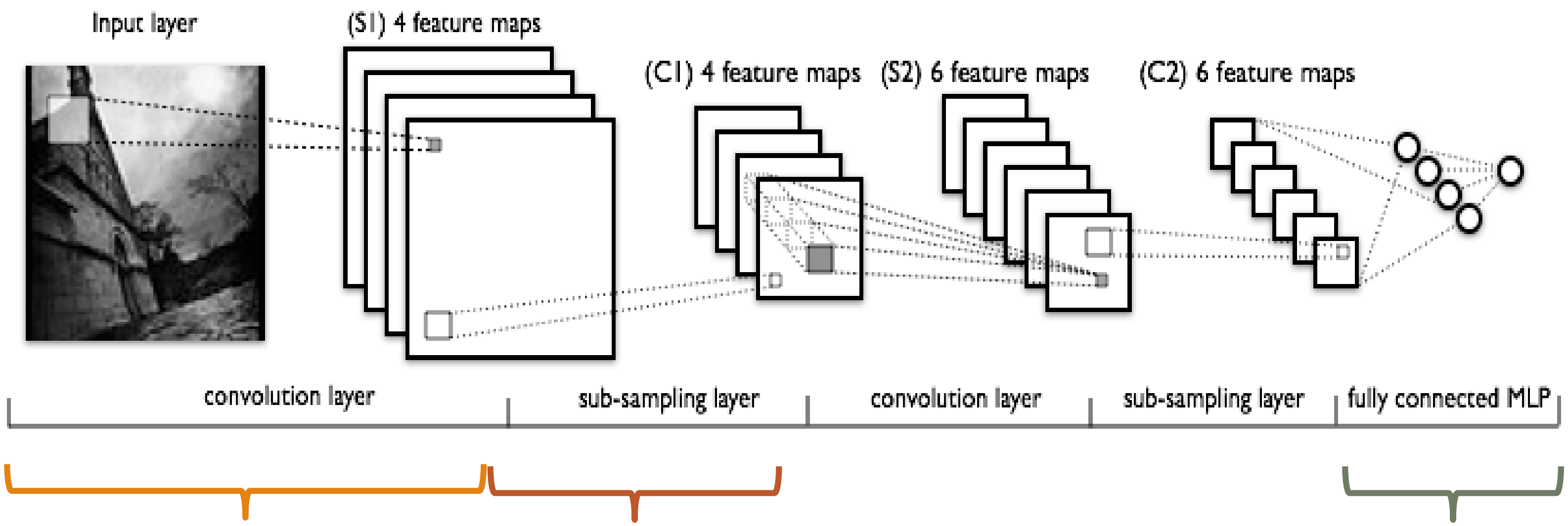
State-of-the-art (since 2015)

Deep Learning (DL)

Convolutional Neural Networks (CNN) <-> Images

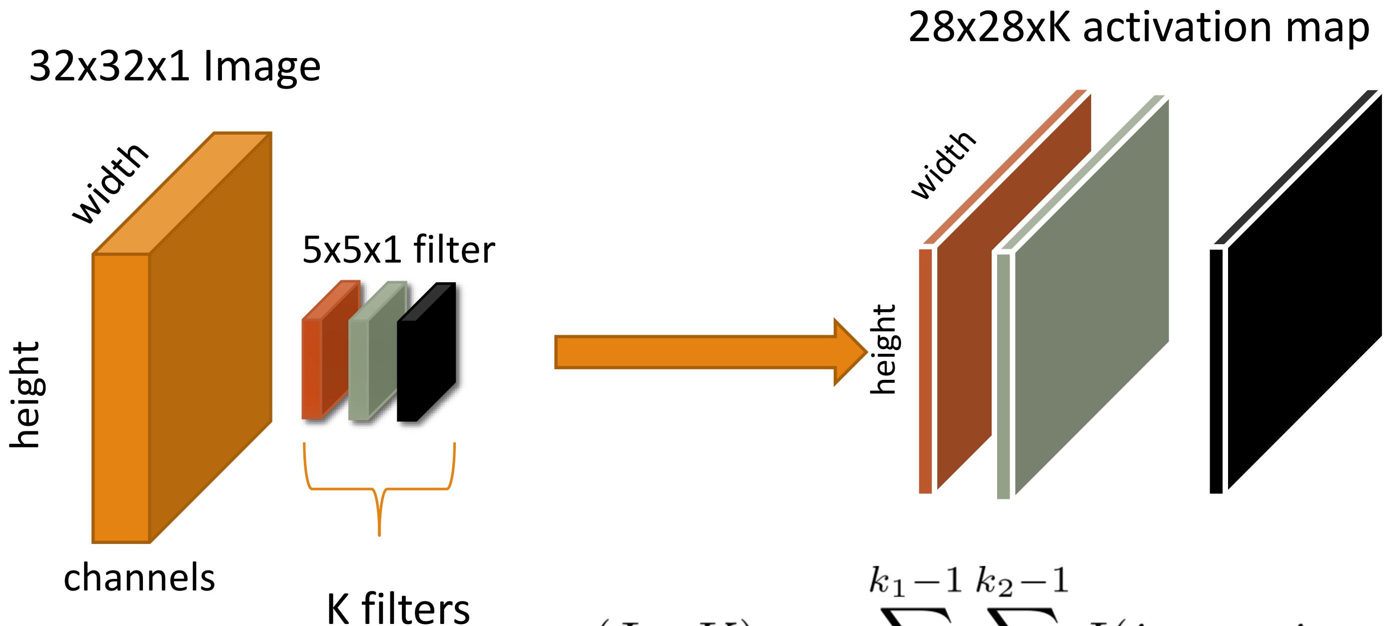
Recurrent Neural Networks (RNN) <-> Audio

Convolutional Neural Networks



(Convolution + Subsampling) + () … + Fully Connected

Convolutional Layers



$$
\begin{aligned}
(I * K)_{ij} &= \sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} I(i-m, j-n)K(m,n) \\
&= \sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} I(i+m, j+n)K(-m,-n)
\end{aligned}
$$

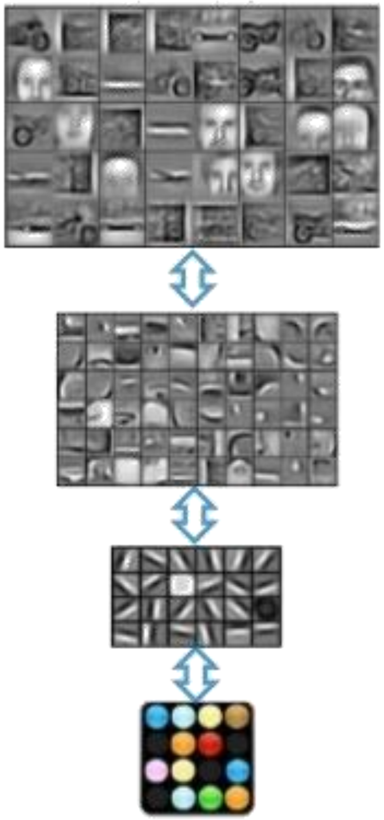
Convolutional Layers

Characteristics

- Hierarchical features
- Location invariance

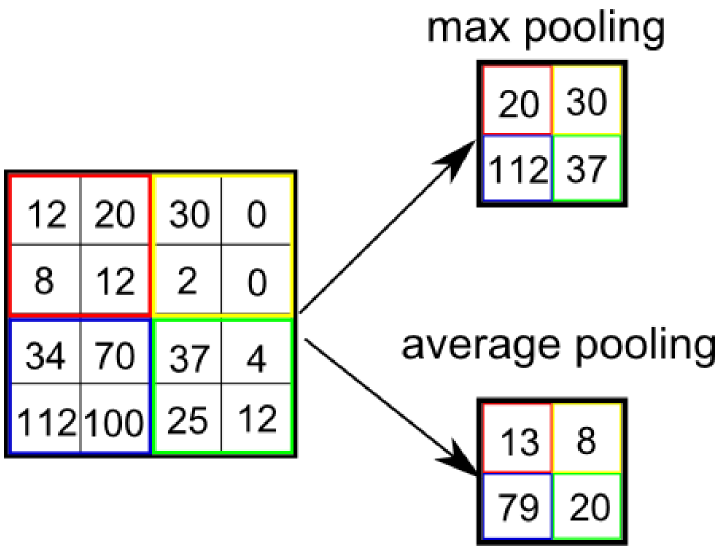
Parameters

- Number of filters (32,64…)
- Filter size (3x3, 5x5)
- Stride (1)
- Padding (2,4)



“Machine Learning and AI for Brain Simulations” – Andrew Ng Talk, UCLA, 2012

Subsampling (pooling) Layers



<-> downsampling

➢ Scale invariance

Parameters

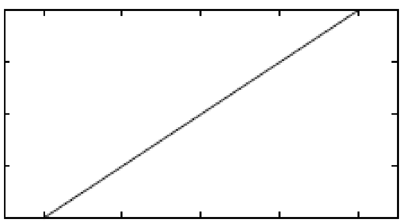
- Type
- Filter Size
- Stride

Activation Layer

Introduction of non-linearity

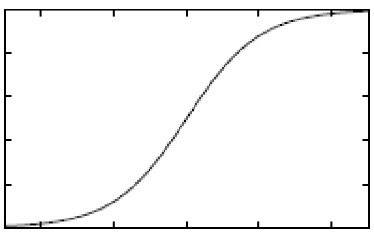
- Brain: thresholding -> spike trains

Identity (Linear)



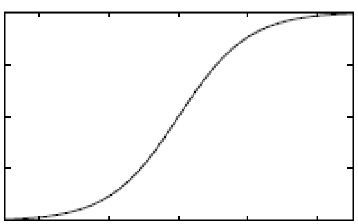
$$identity(x) = x$$

Sigmoid



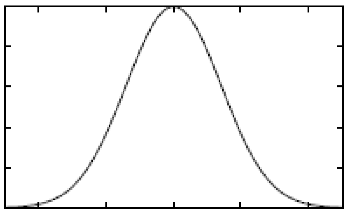
$$sigmoid(x) = \frac{1}{1 + e^{-x}}$$

Tanh (Hypertangent)



$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Gaussian



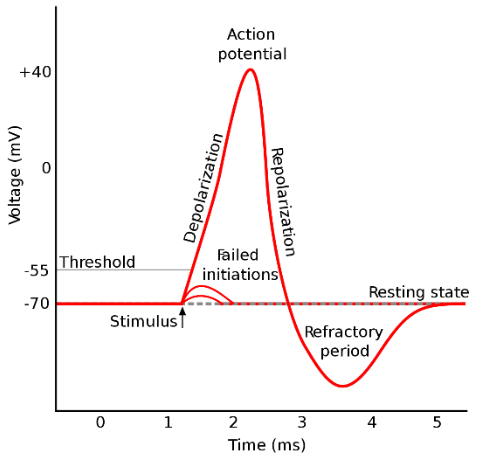
$$gaussian(x) = e^{-x^2/\sigma^2}$$

Activation Layer

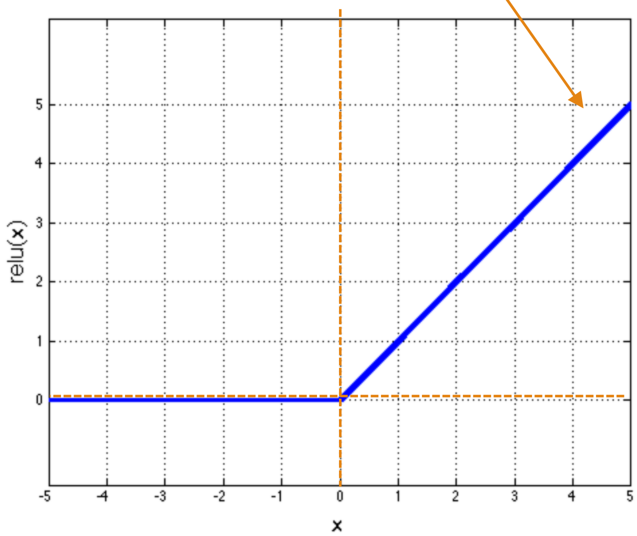
ReLU: x=max(0,x)

- ✓ Simplifies backprop
- ✓ Makes learning faster
- ✓ Avoids saturation issues
- ✓ ~ non-negativity constraint

(Note: The brain)



No saturated gradients

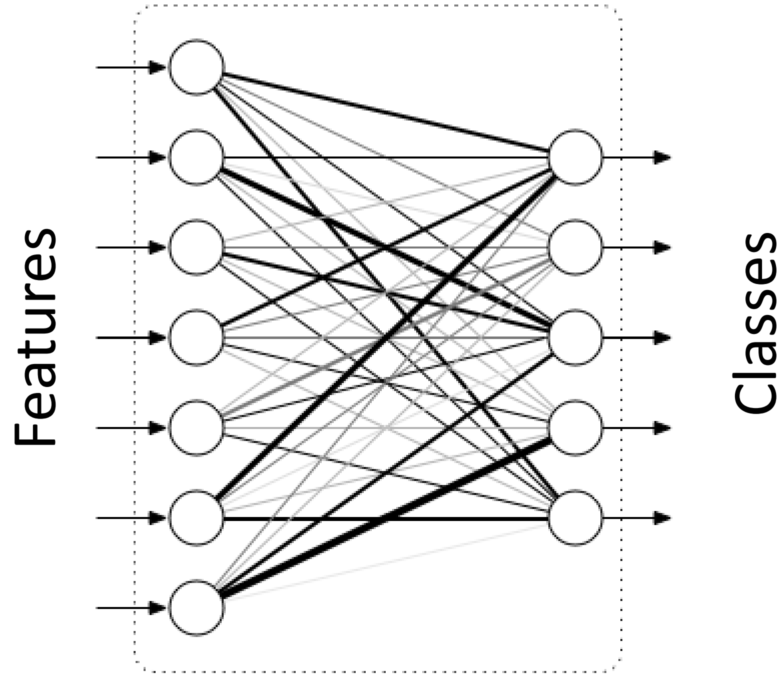


Fully Connected Layers

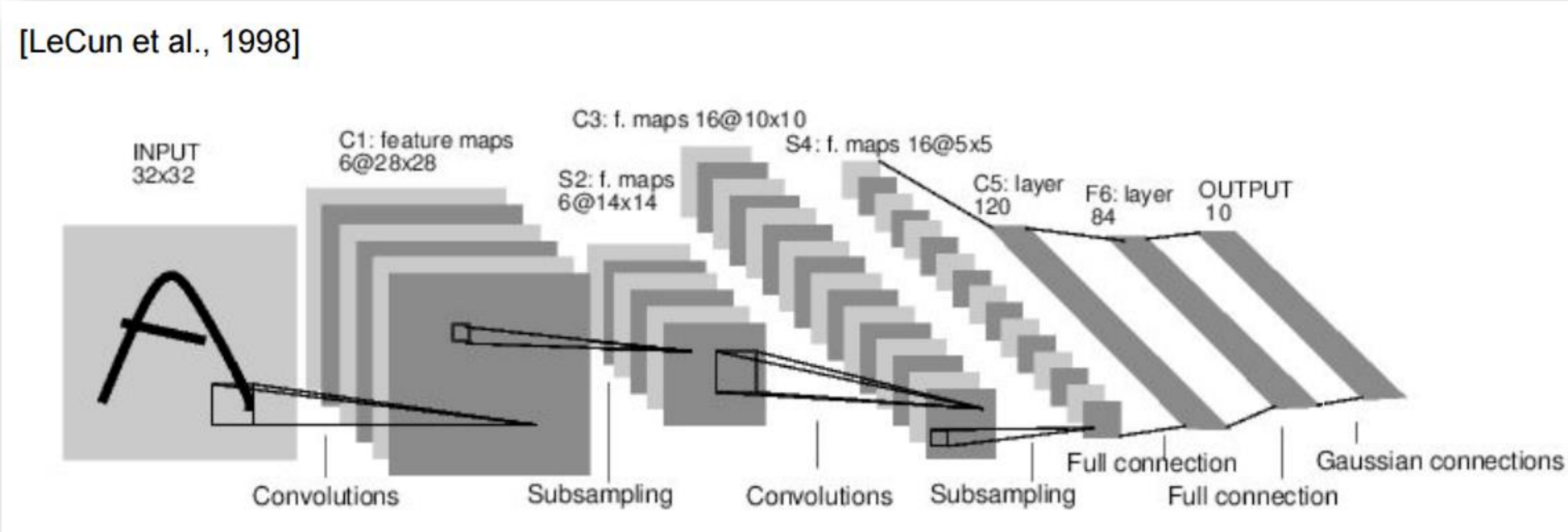
Full connections to all activations in previous layer

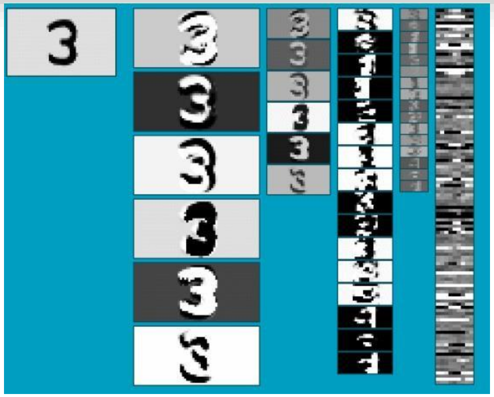
Typically at the end

Can be replaced by conv

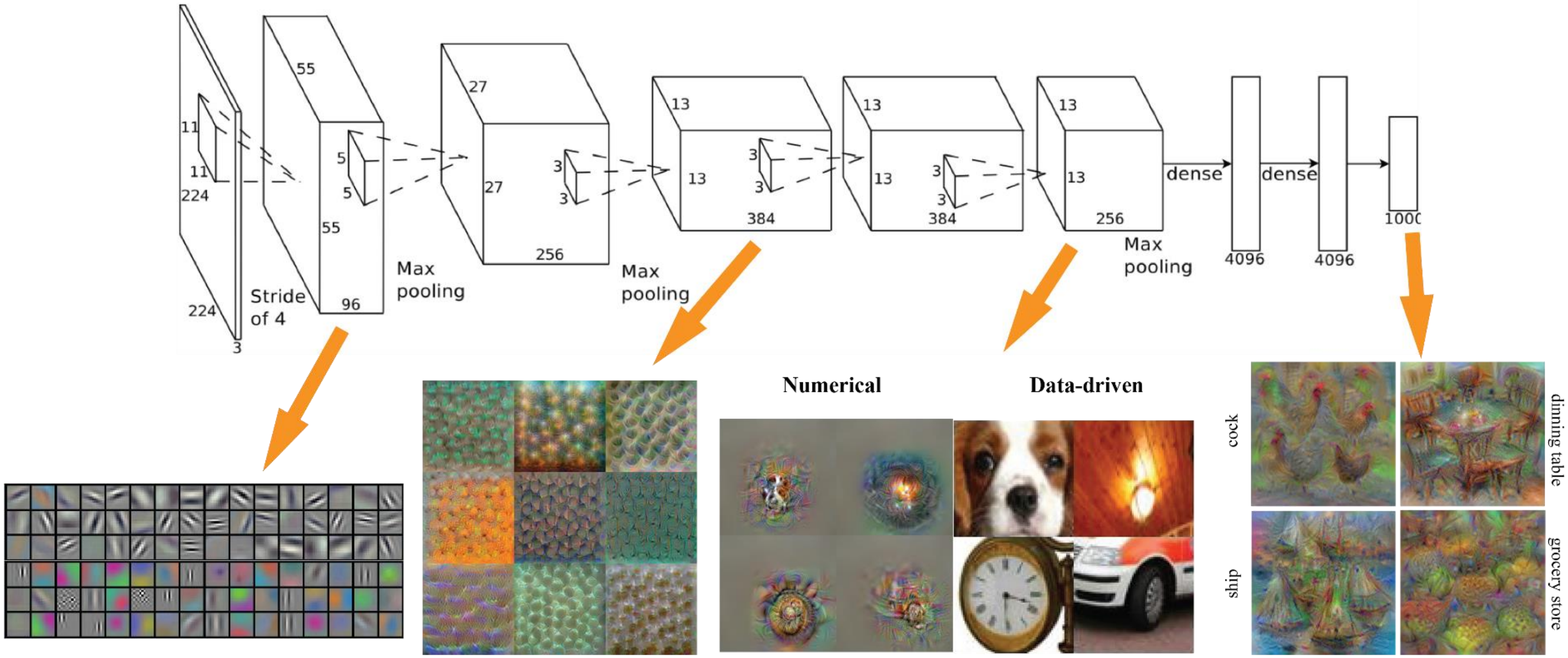


LeNet [1998]


[LeCun et al., 1998]
INPUT
32x32
C1: feature maps
6@28x28
S2: f. maps
6@14x14
C3: f. maps 16@10x10
S4: f. maps 16@5x5
C5: layer
120
F6: layer
84
OUTPUT
10
Convolutions
Subsampling
Convolutions
Subsampling
Full connection
Full connection
Gaussian connections



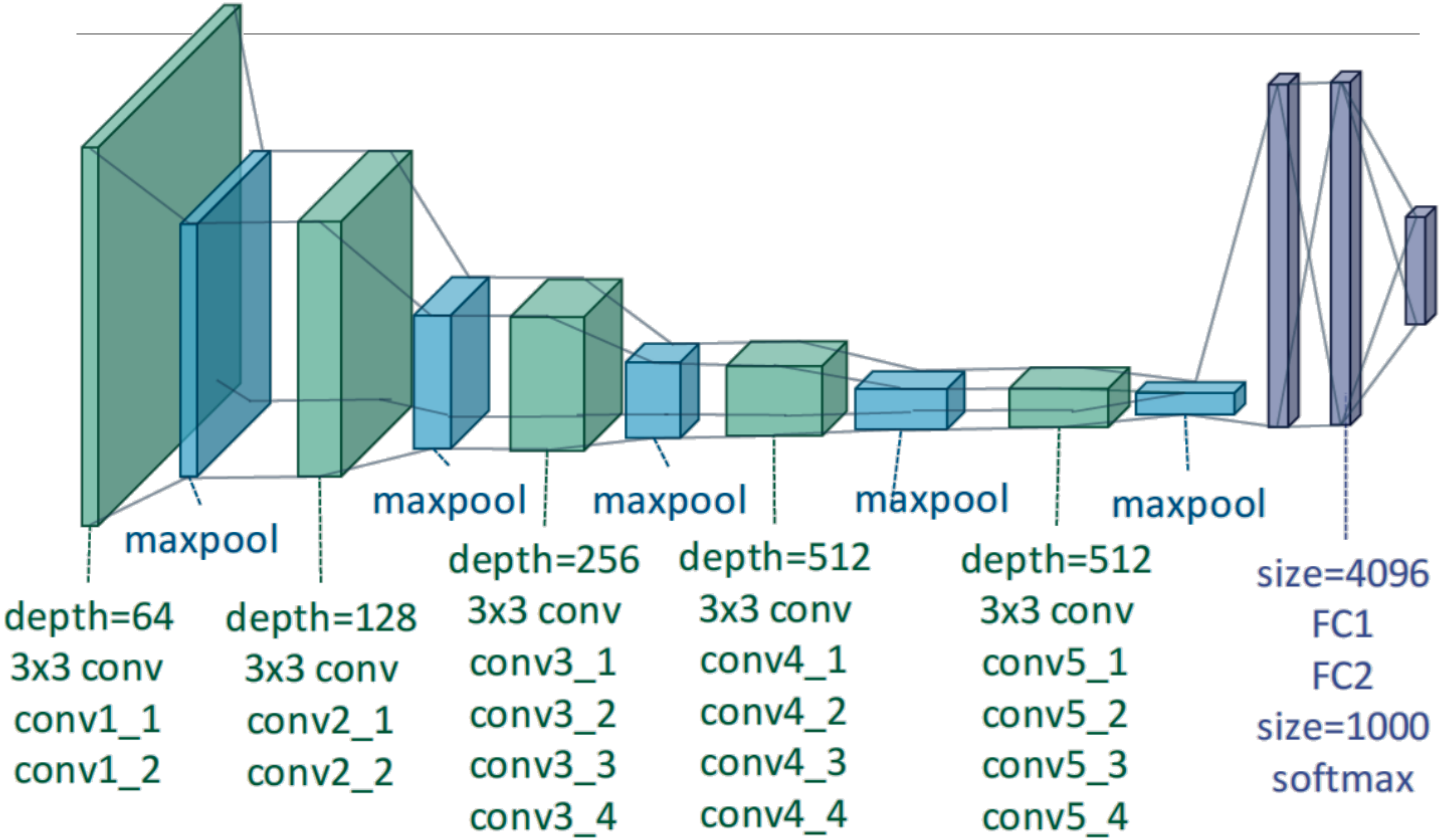
AlexNet [2012]



Conv 1: Edge+Blob
Conv 3: Texture
Conv 5: Object Parts
Fc8: Object Classes

Alex Krizhevsky, Ilya Sutskever and Geoff Hinton, ImageNet ILSVRC challenge in 2012
http://vision03.csail.mit.edu/cnn_art/data/single_layer.png

VGGnet [2014]



K. Simonyan, A. Zisserman Very Deep Convolutional Networks for Large-Scale Image Recognition, arXiv technical report, 2014

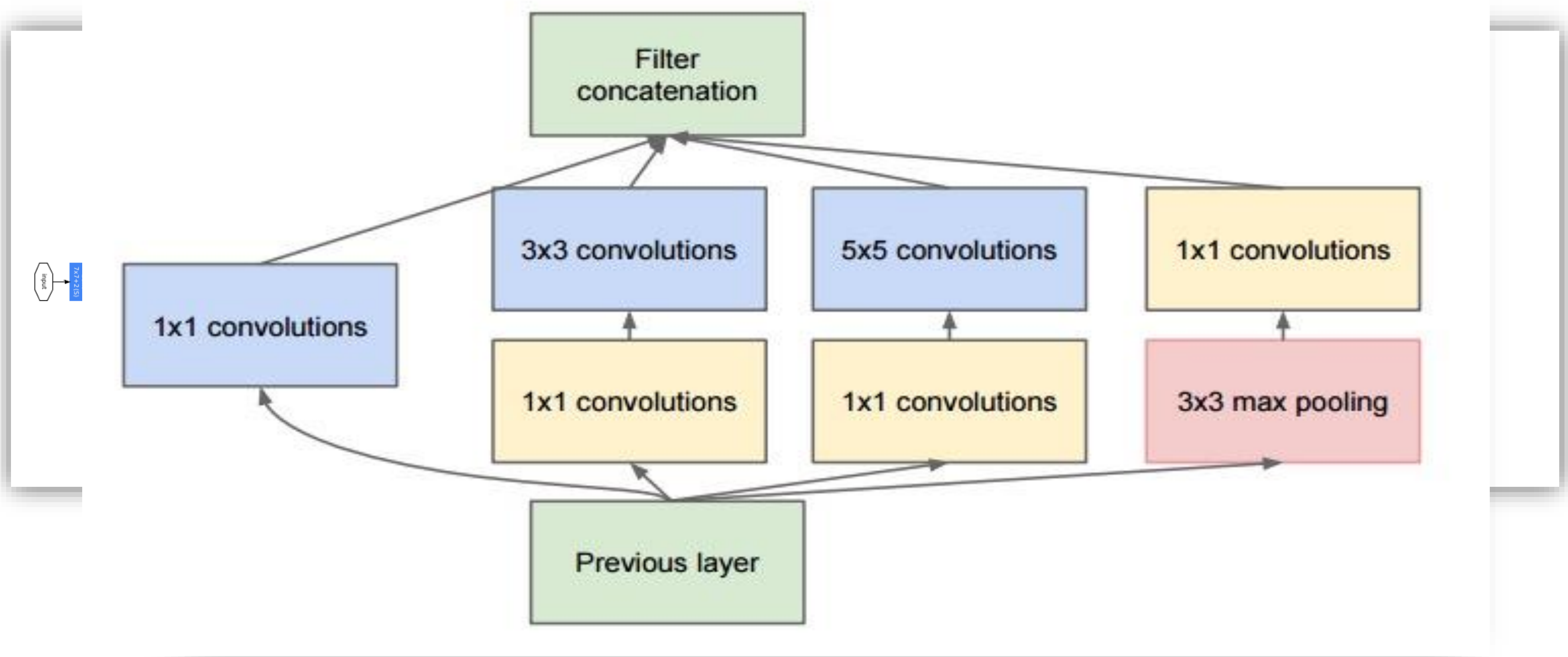
VGGnet

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

D: VGG16
E: VGG19
All filters are 3x3

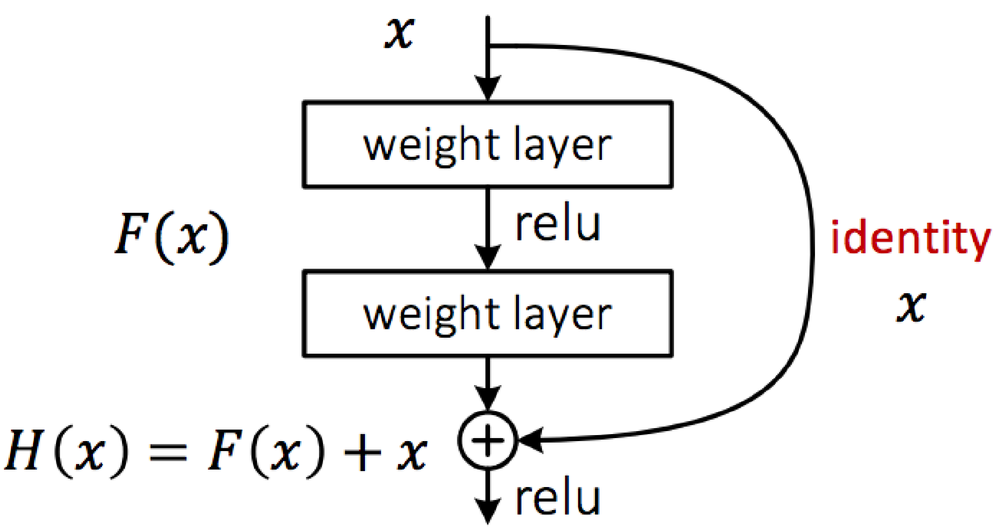
More layers
smaller filters

Inception (GoogLeNet, 2014)

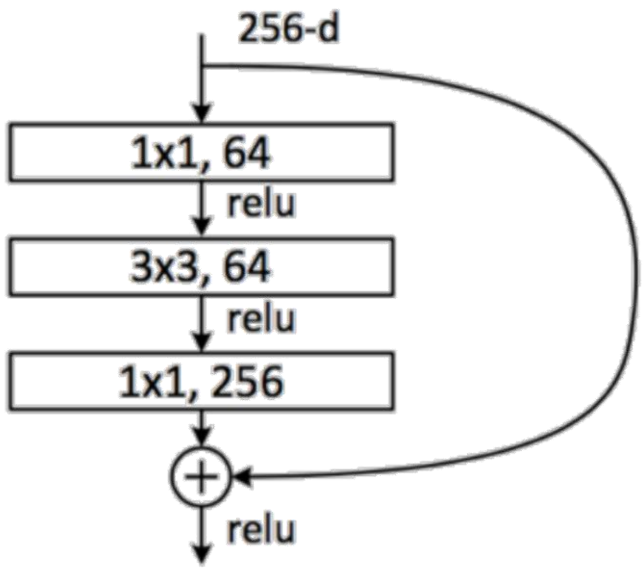


Inception module with dimensionality reduction

Residuals



x
weight layer
relu
$F(x)$
weight layer
identity
x
$H(x) = F(x) + x$
relu

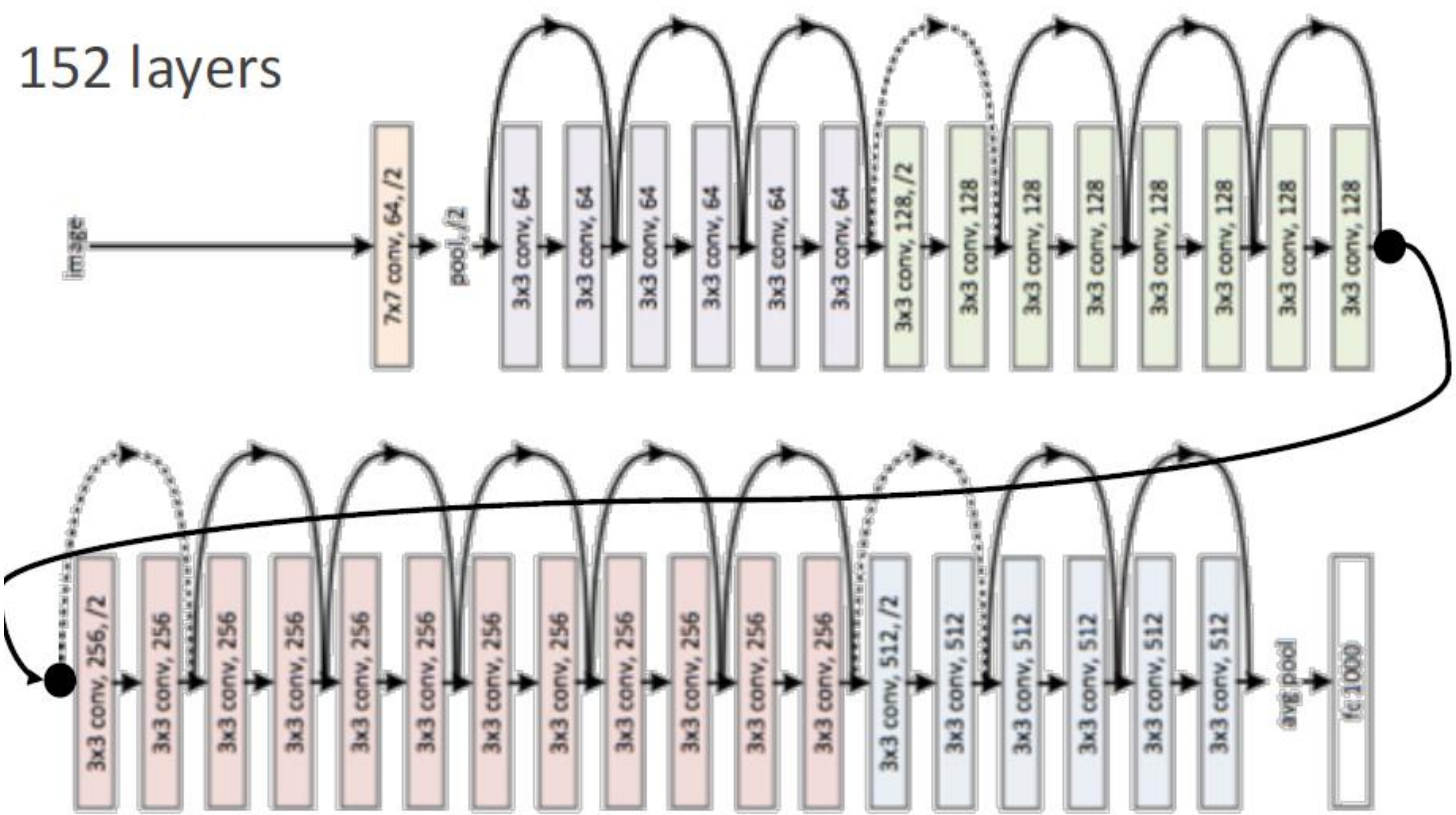


256-d
1x1, 64
relu
3x3, 64
relu
1x1, 256
relu

ResNet, 2015

Residual Networks

152 layers



He, Kaiming, et al. "Deep residual learning for image recognition." *IEEE CVPR*. 2016.

Training protocols

Fully Supervised

- Random initialization of weights
- Train in supervised mode (example + label)

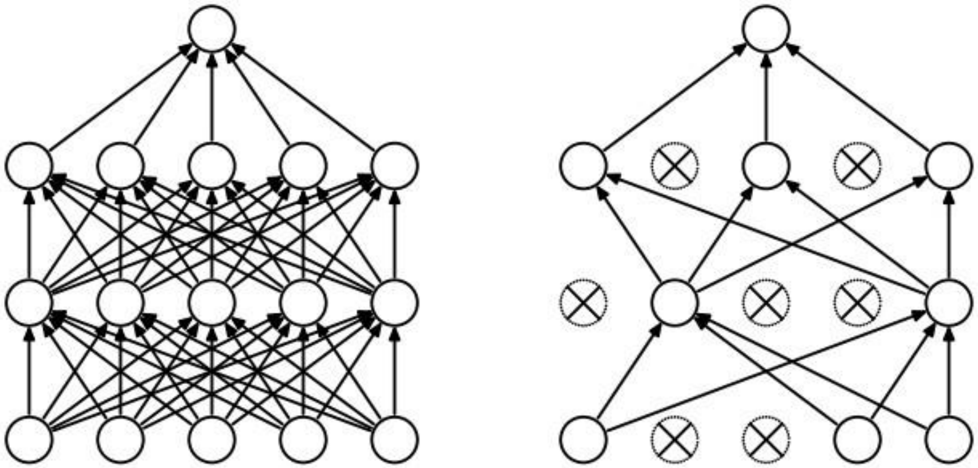
Unsupervised pre-training + standard classifier

- Train each layer unsupervised
- Train a supervised classifier (SVM) on top

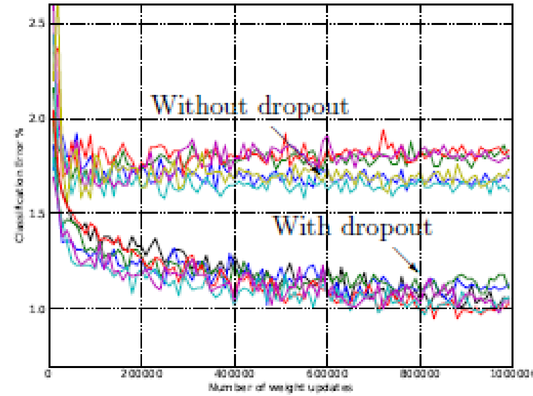
Unsupervised pre-training + supervised fine-tuning

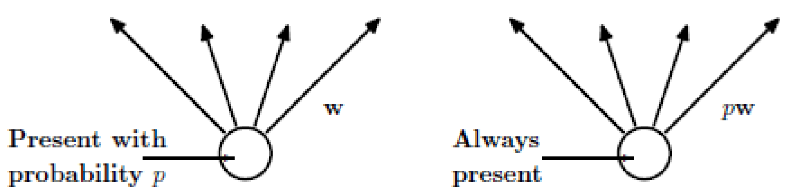
- Train each layer unsupervised
- Add a supervised layer

Dropout



(a) Standard Neural Net (b) After applying dropout.





(a) At training time (b) At test time

Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." *Journal of machine learning research*15.1 (2014): 1929-1958.

Batch Normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1 \ldots m}\}$;
Parameters to be learned: γ, β

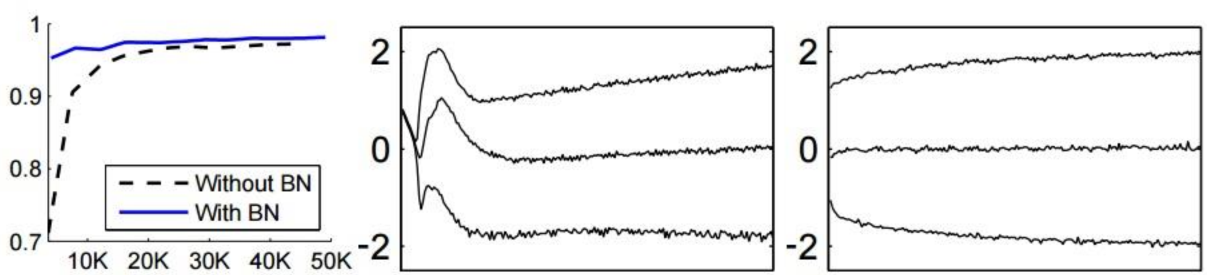
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

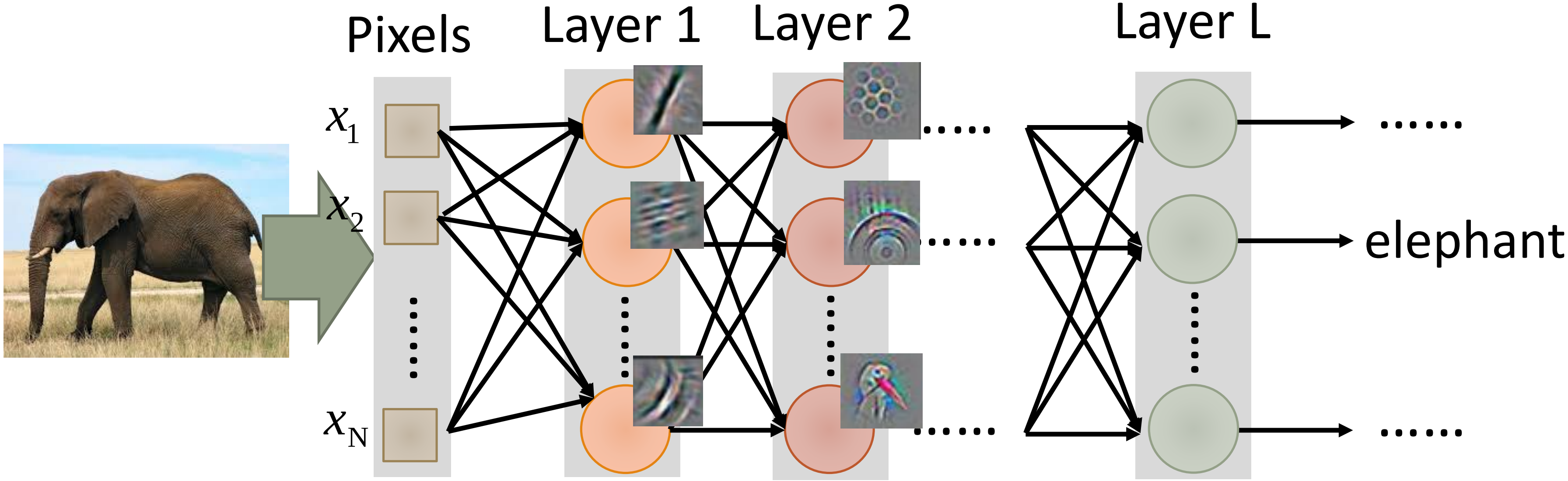
$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$



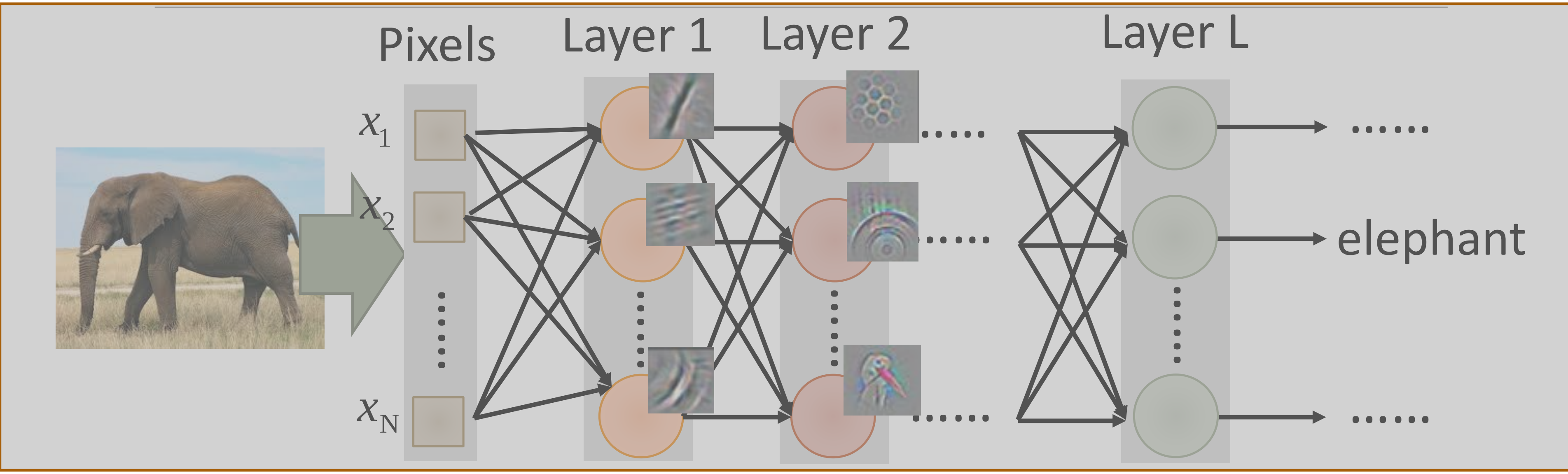
(a) (b) Without BN (c) With BN

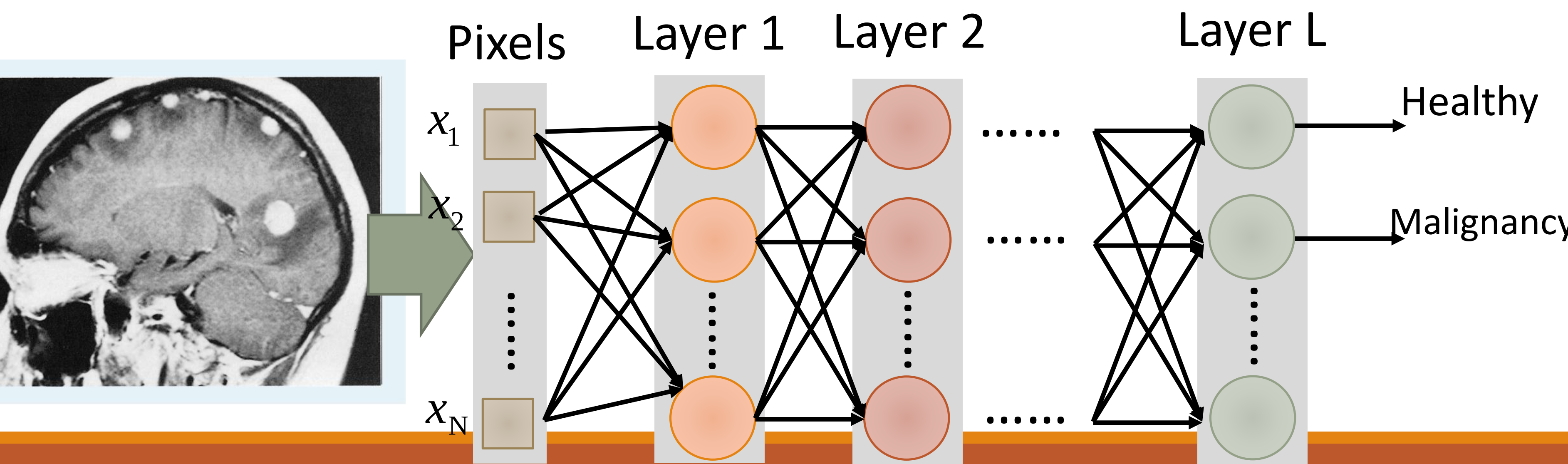
Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift [Ioffe and Szegedy 2015]

Transfer Learning

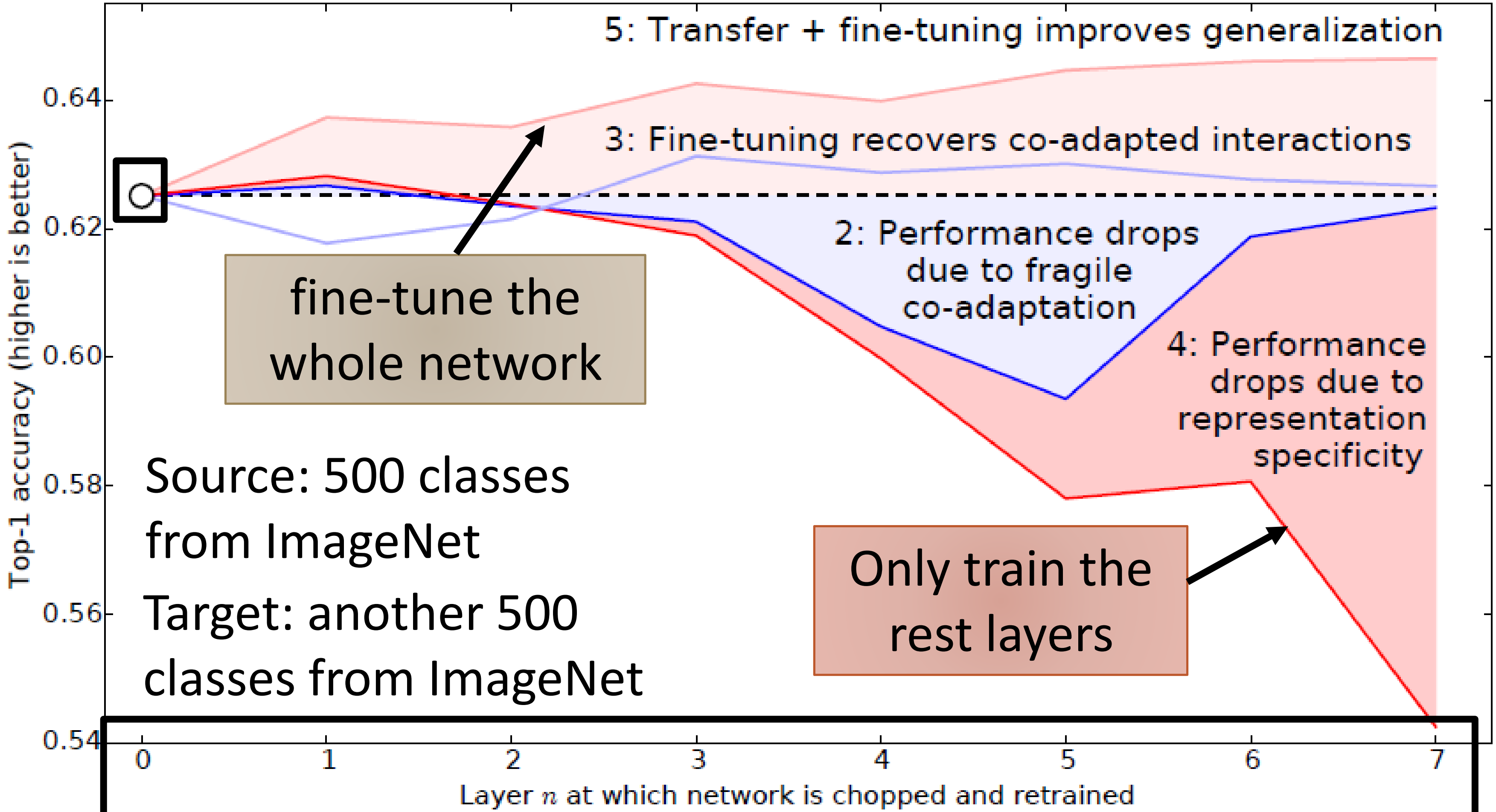


Transfer Learning





Layer Transfer - Image



fine-tune the whole network

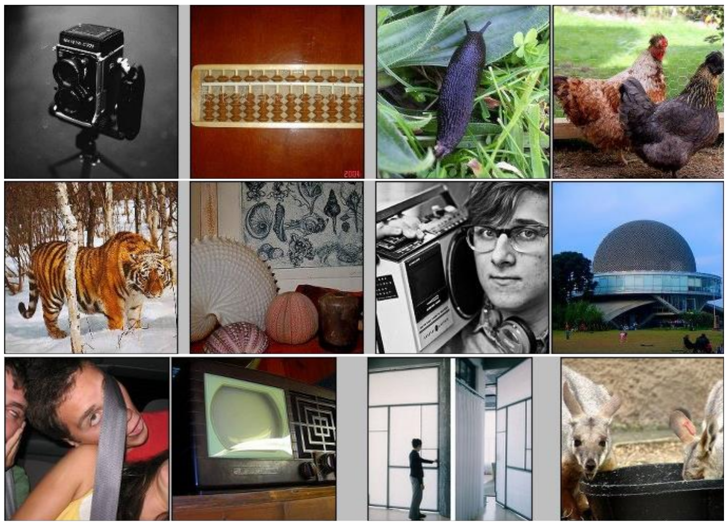
Only train the rest layers

Source: 500 classes from ImageNet

Target: another 500 classes from ImageNet

Jason Yosinski, Jeff Clune, Yoshua Bengio, Hod Lipson, "How transferable are features in deep neural networks?", NIPS, 2014

ImageNET



- ~14 million labeled images, 20k classes
- Images gathered from Internet
- Human labels via Amazon MTurk
- ImageNet Large-Scale Visual Recognition Challenge (ILSVRC):
  1.2 million training images, 1000 classes

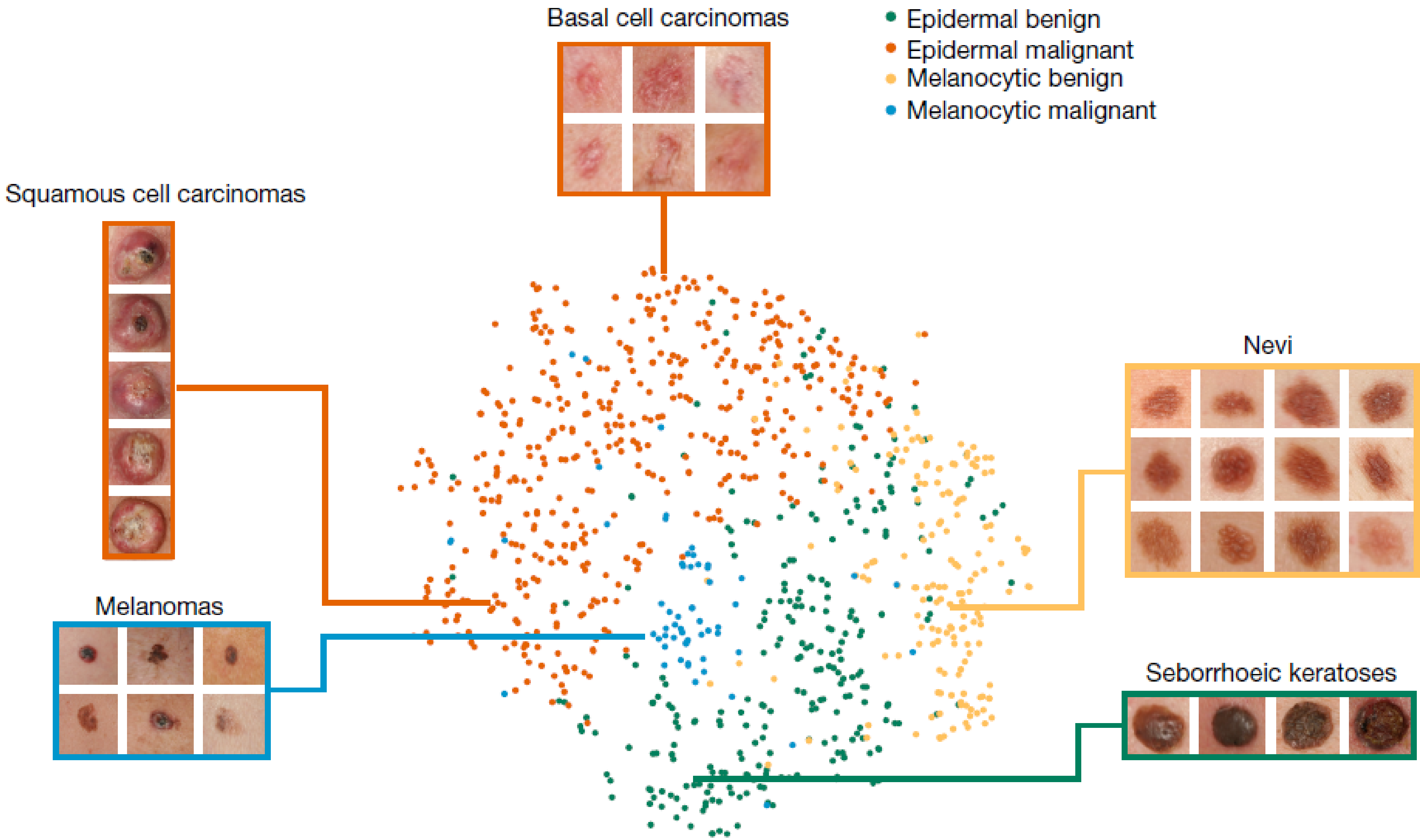
www.image-net.org/challenges/LSVRC/

Summary: ILSVRC 2012-2015

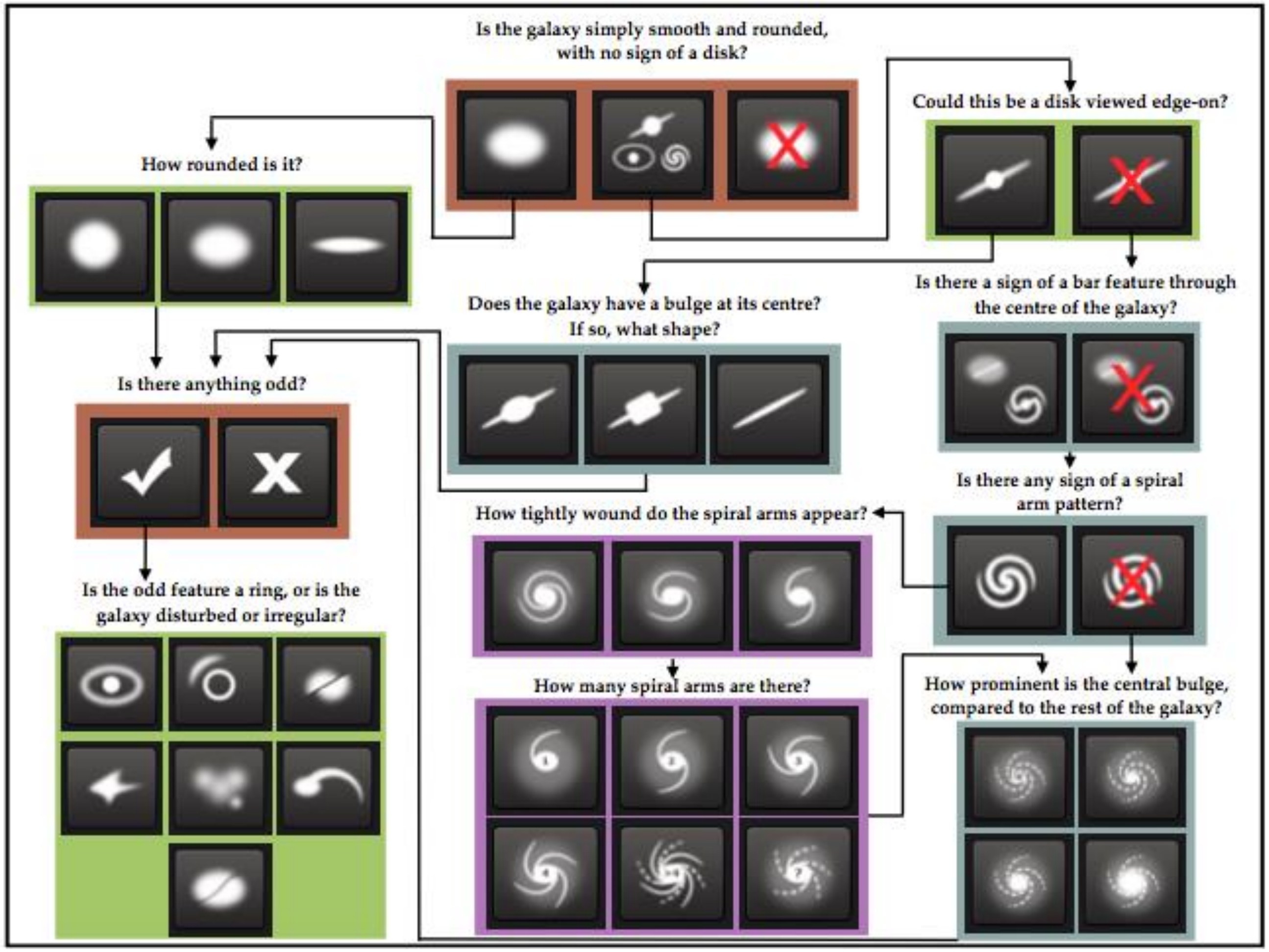
Team	Year	Place	Error (top-5)	External data
(AlexNet, 7 layers)	2012	-	16.4%	no
SuperVision	2012	1st	15.3%	ImageNet 22k
Clarifai – NYU (7 layers)	2013	-	11.7%	no
Clarifai	2013	1st	11.2%	ImageNet 22k
VGG – Oxford (16 layers)	2014	2nd	7.32%	no
GoogLeNet (19 layers)	2014	1st	6.67%	no
ResNet (152 layers)	2015	1st	3.57%	
Human expert*			5.1%	

http://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet/

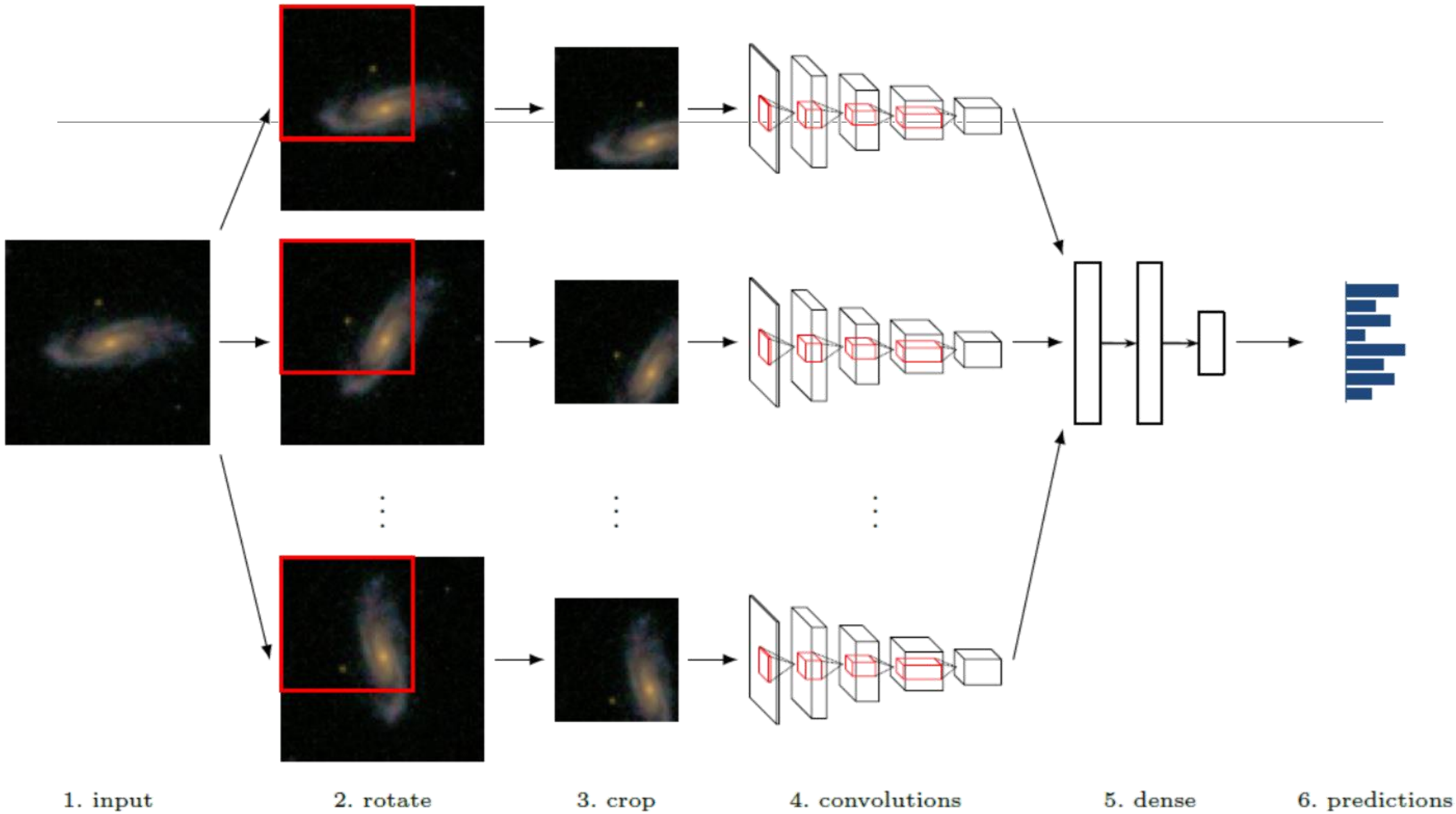
Skin cancer detection



The Galaxy zoo challenge

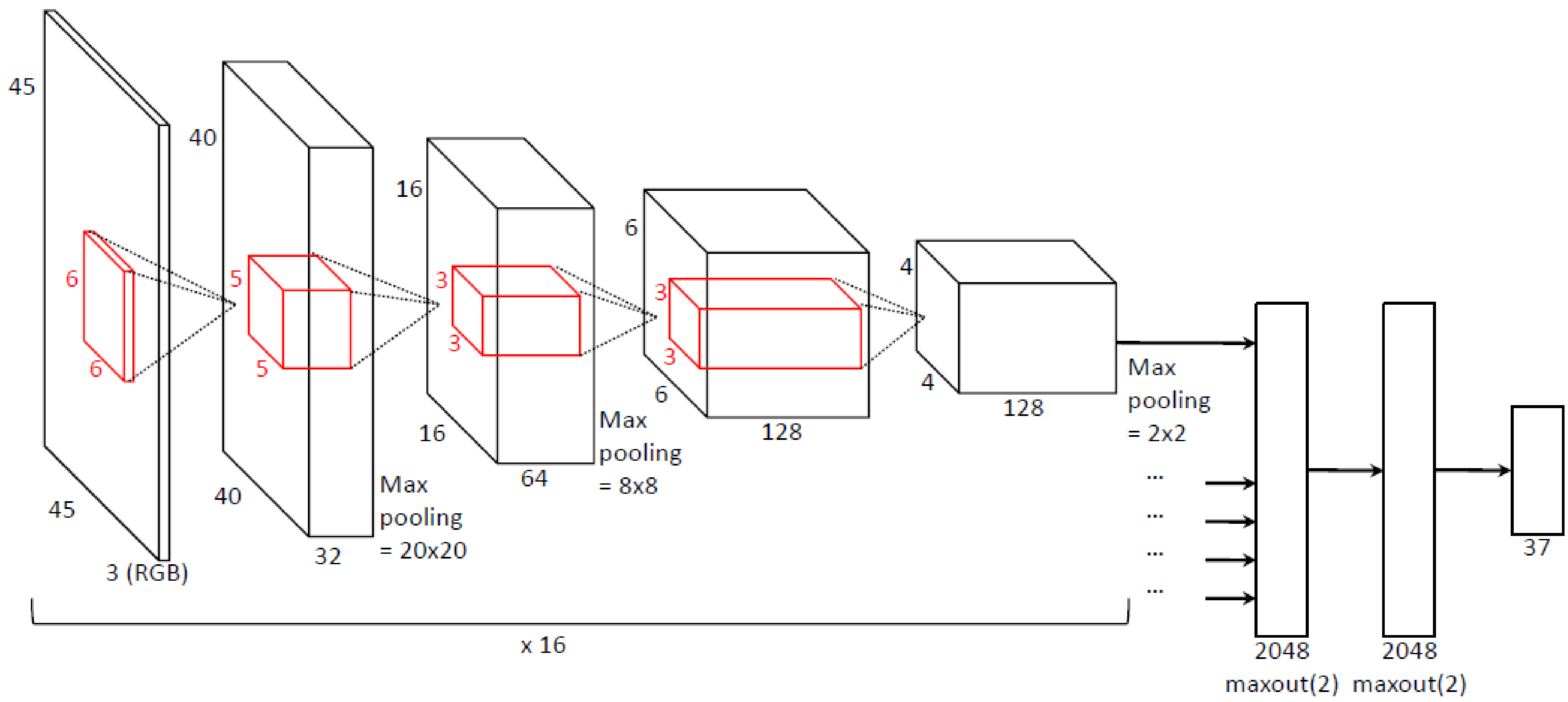


Online crowdsourcing project where users describe the morphology of galaxies based on color images 1 million galaxies imaged by the Sloan Digital Sky Survey (2007)



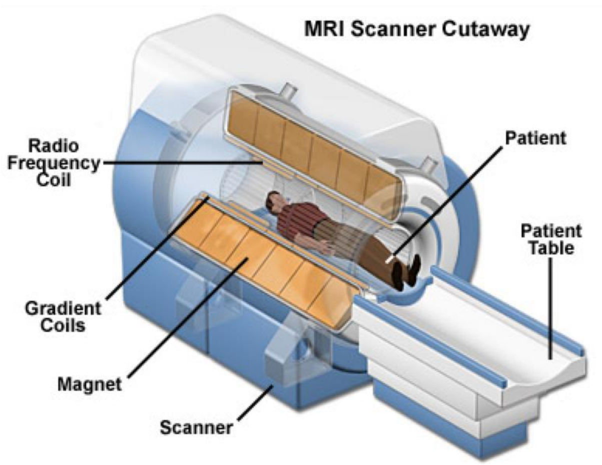
Dieleman, S., Kyle W. W., and Joni D.. "Rotation-invariant convolutional neural networks for galaxy morphology prediction." Monthly notices of the royal astronomical society, 2015

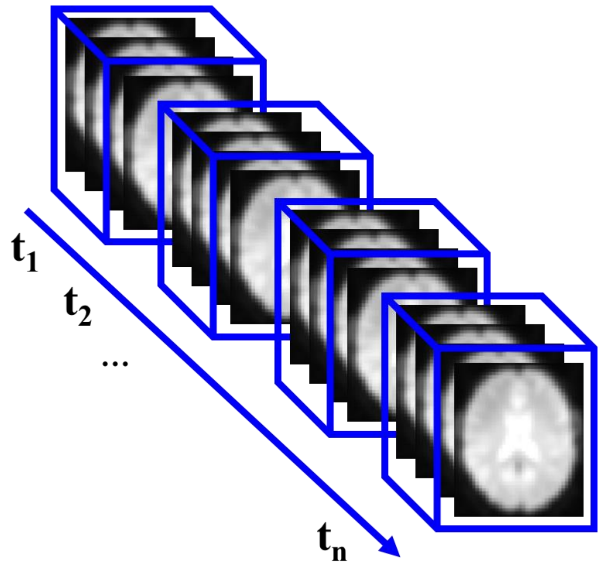
Component

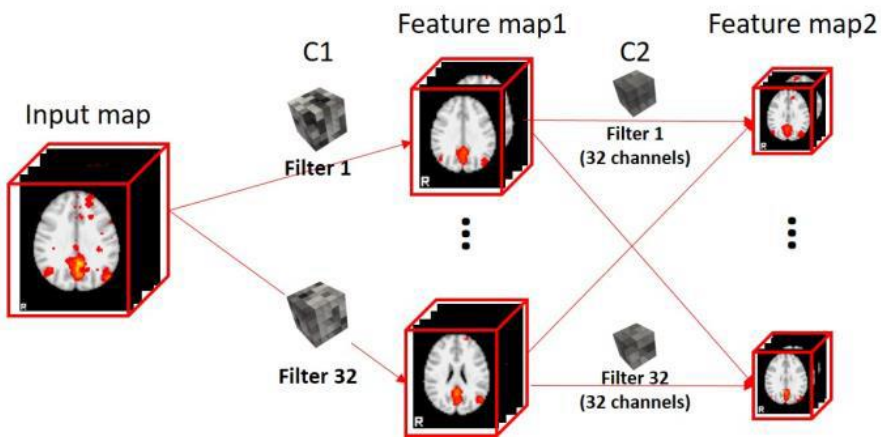


45
45
3 (RGB)
6
6
40
40
32
5
5
Max pooling = 20x20
16
16
64
3
3
Max pooling = 8x8
6
6
128
3
3
4
4
128
Max pooling = 2x2
...
...
...
...
x 16
2048
maxout(2)
2048
maxout(2)
37

CNN & FMRI


MRI Scanner Cutaway
Radio Frequency Coil
Patient
Patient Table
Gradient Coils
Magnet
Scanner


t_1
t_2
...
t_n


Input map
C1
Filter 1
Filter 32
Feature map1
C2
Filter 1
(32 channels)
Filter 32
(32 channels)
Feature map2

Demos

https://www.clarifai.com/demo

Different types of mapping

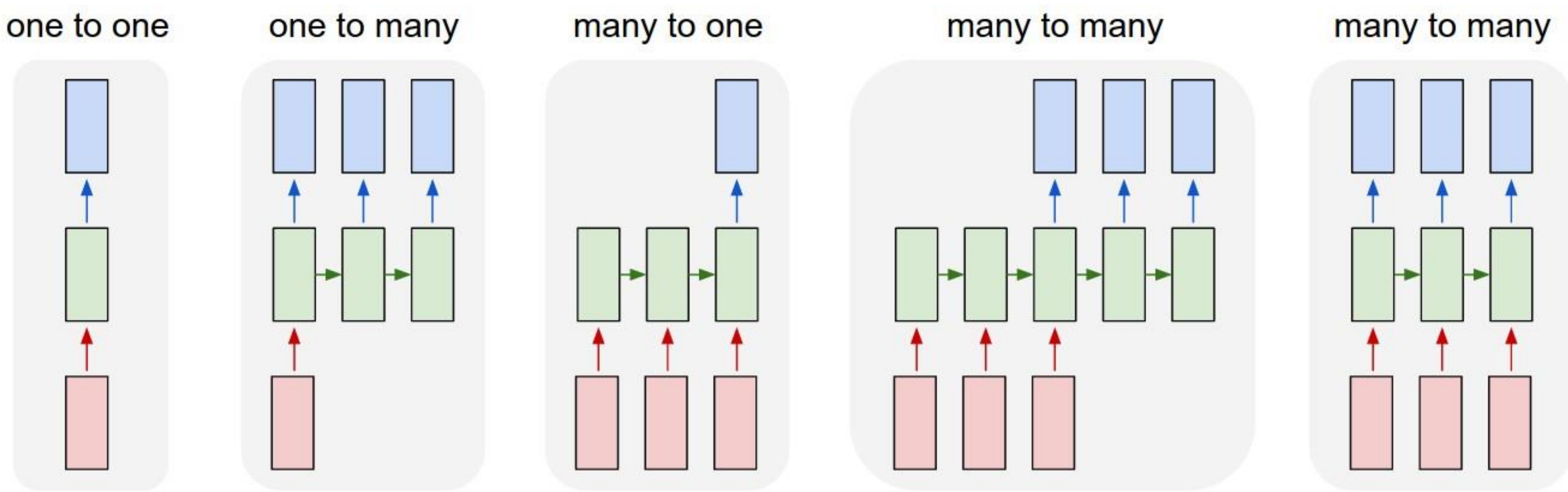


Image classification

Image captioning

Sentiment analysis

Machine translation

Synced sequence(vide
classification)

Recurrent Neural Networks

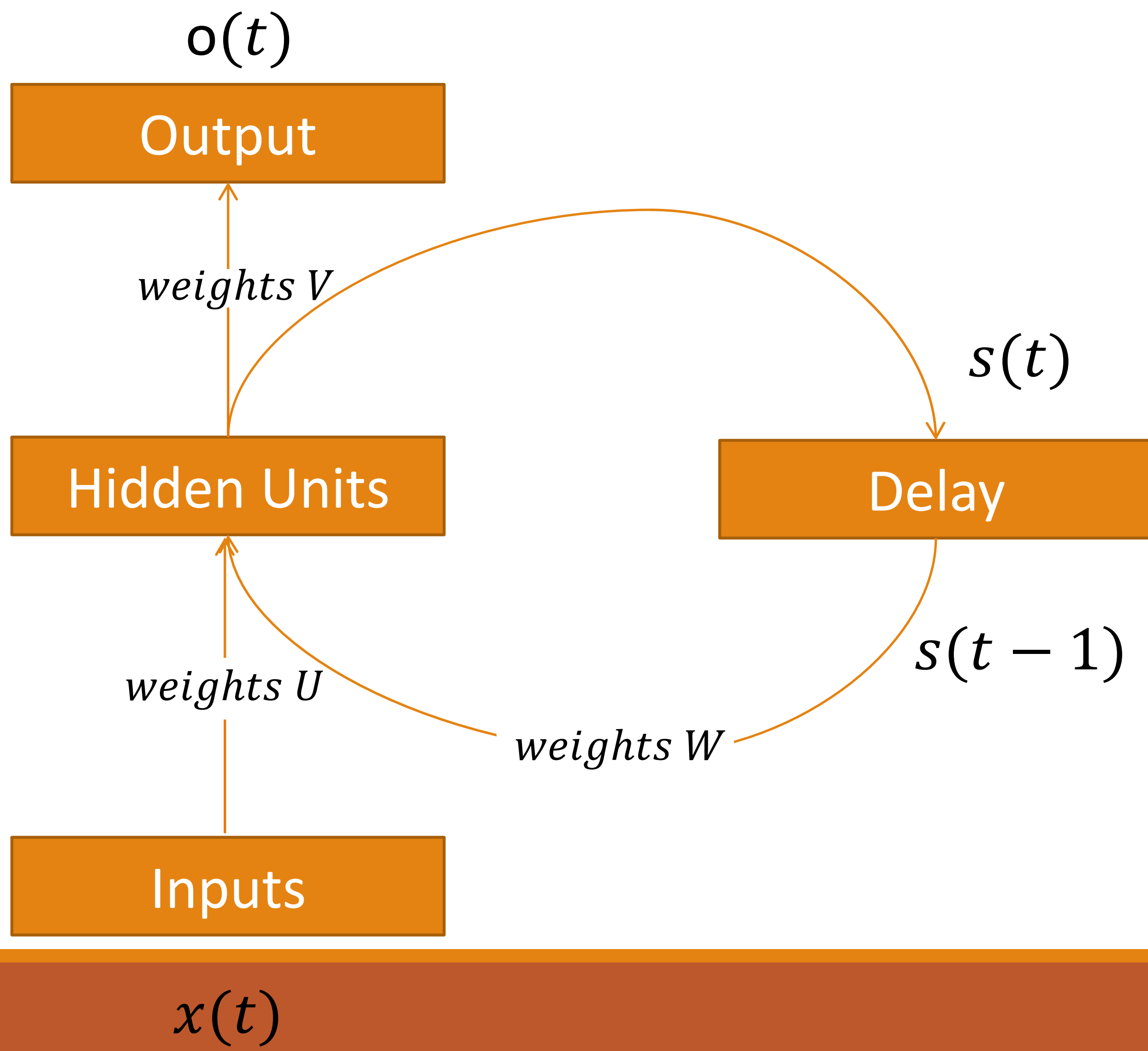
Motivation

- Feed forward networks accept a fixed-sized vector as input and produce a fixed-sized vector as output
- fixed amount of computational steps
- recurrent nets allow us to operate over *sequences* of vectors

Use cases

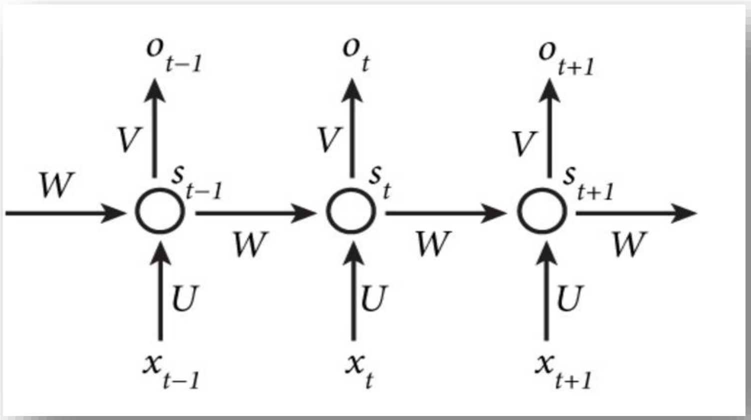
- Video
- Audio
- Text

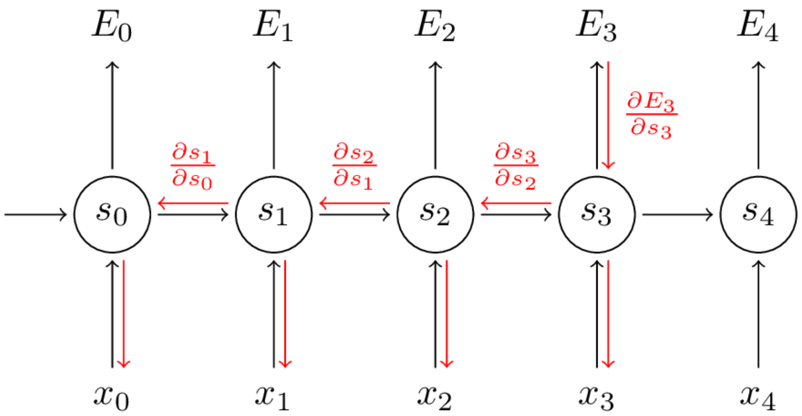
RNN Architecture



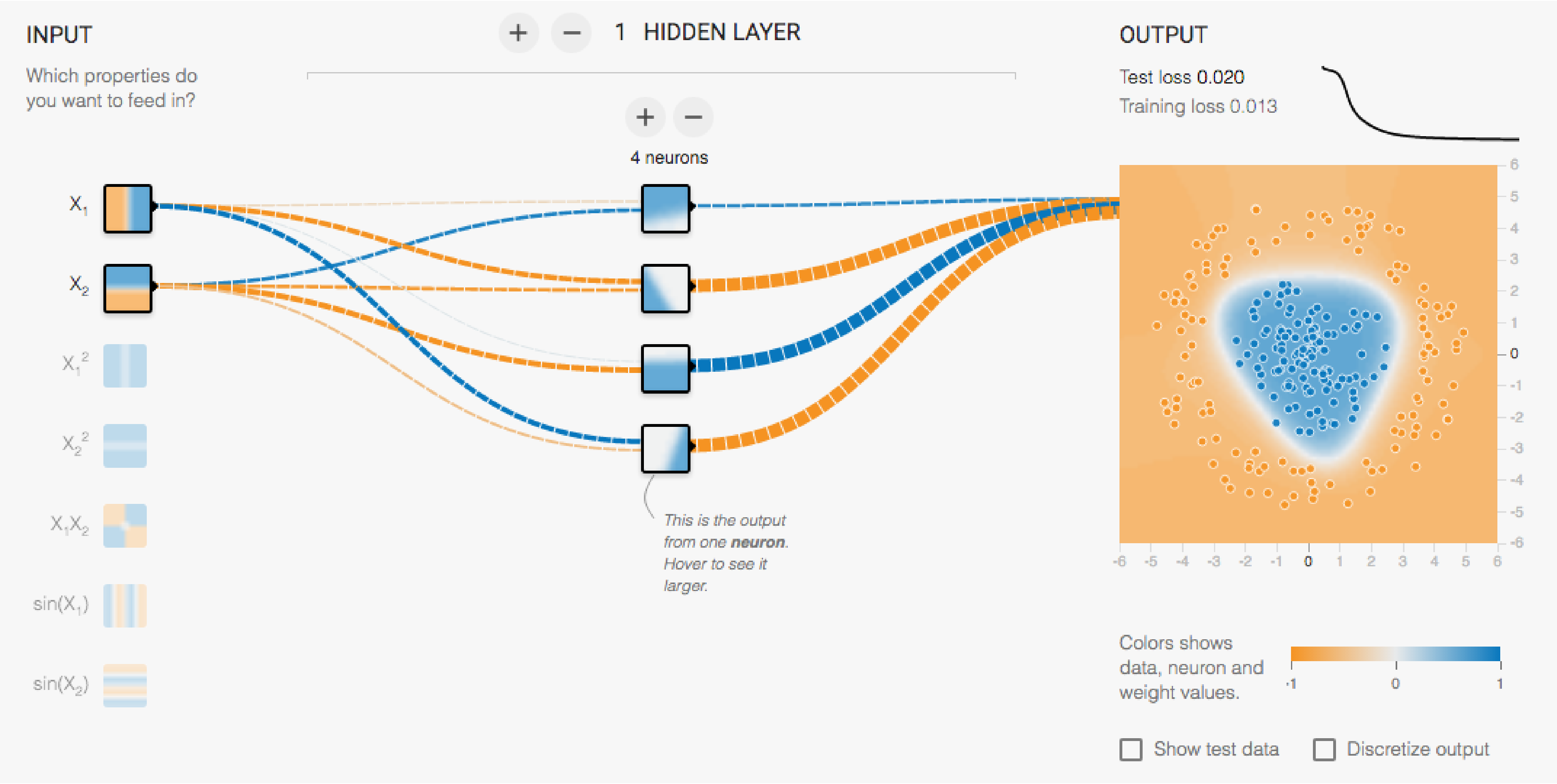
Unfolding RNNs

- Each node represents a layer of network units at a single time step.
- The same weights are reused at every time step.





Multi-Layer Network Demo



http://playground.tensorflow.org/