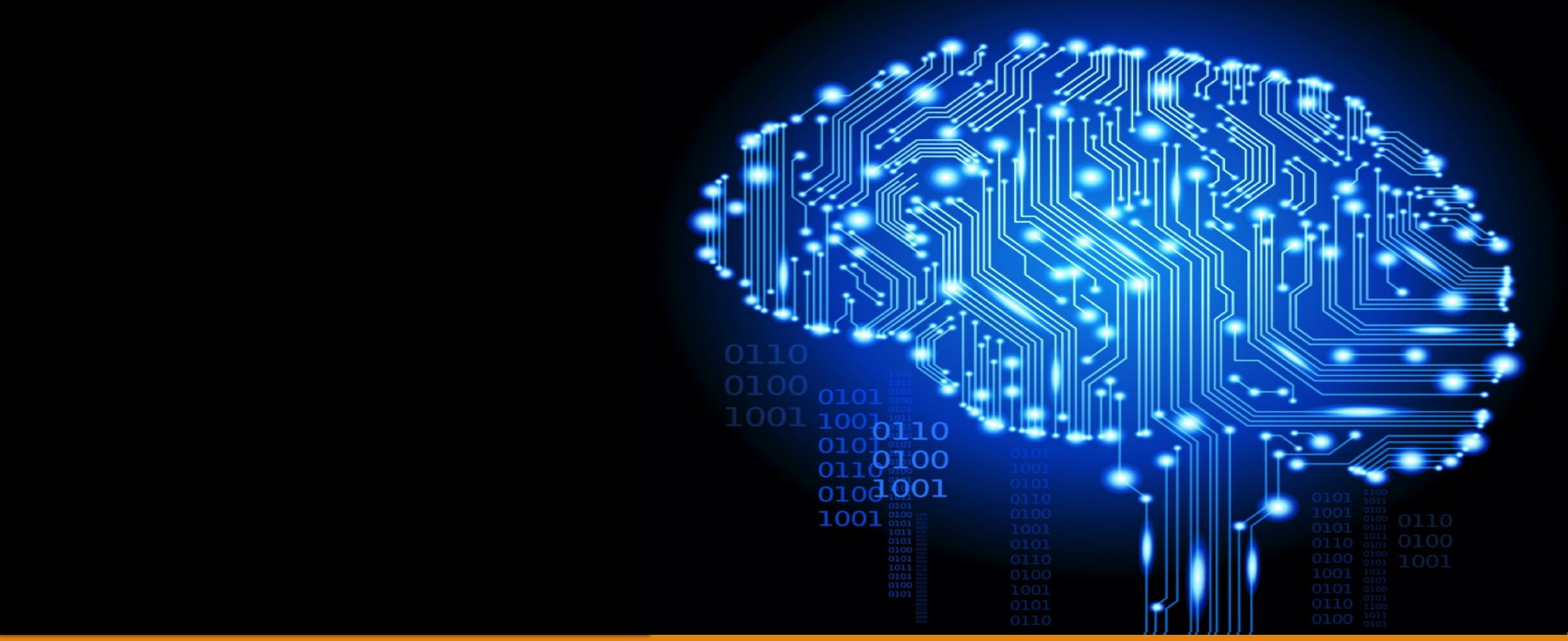


Lecture 1: Introduction to Neural Networks and Deep Learning

Deep Learning @ UvA

Course Logistics

- Course: Theory (4 hours per week) + Labs (4 hours per week)
  - Assignments in Torch
  - Free to choose your own framework (with some rules)
- The times, days and rooms are **different** each week
  - Detailed schedule: https://datanose.nl/#course[46217]
- Final grade = 50% from lab assignments + 50% from final exam
  - 5 lab assignments = (50 + 100 + 100 + 100 +100)=450 points in total
  - The assignments should be done individually
  - + Bonus questions to recover points
  - 7 free late days in total for all assignments. Use them as you please. After that lose 10% per extra late day

Prerequisites

- Calculus, Linear Algebra
  - Gradients
  - Matrix operations
- Probability and Statistics
- Advanced programming
- Time and patience

Course overview

- Course designed to go in depth in theory and get hands-on practical experience
- Organized lectures and practicals in 5 thematic groups
  - *Theme 1: How to learn with a neural network?*
  - *Theme 2: How to classify pixels?*
  - *Theme 3: How to synthesize words?*
  - *Theme 4: Unsupervised and Bayesian Deep Learning*
  - *Theme 5: The bigger picture*
- Slides, lecture notes and papers
  - All posted on the course website
- Book on Neural Networks and Deep Learning from Y. Bengio

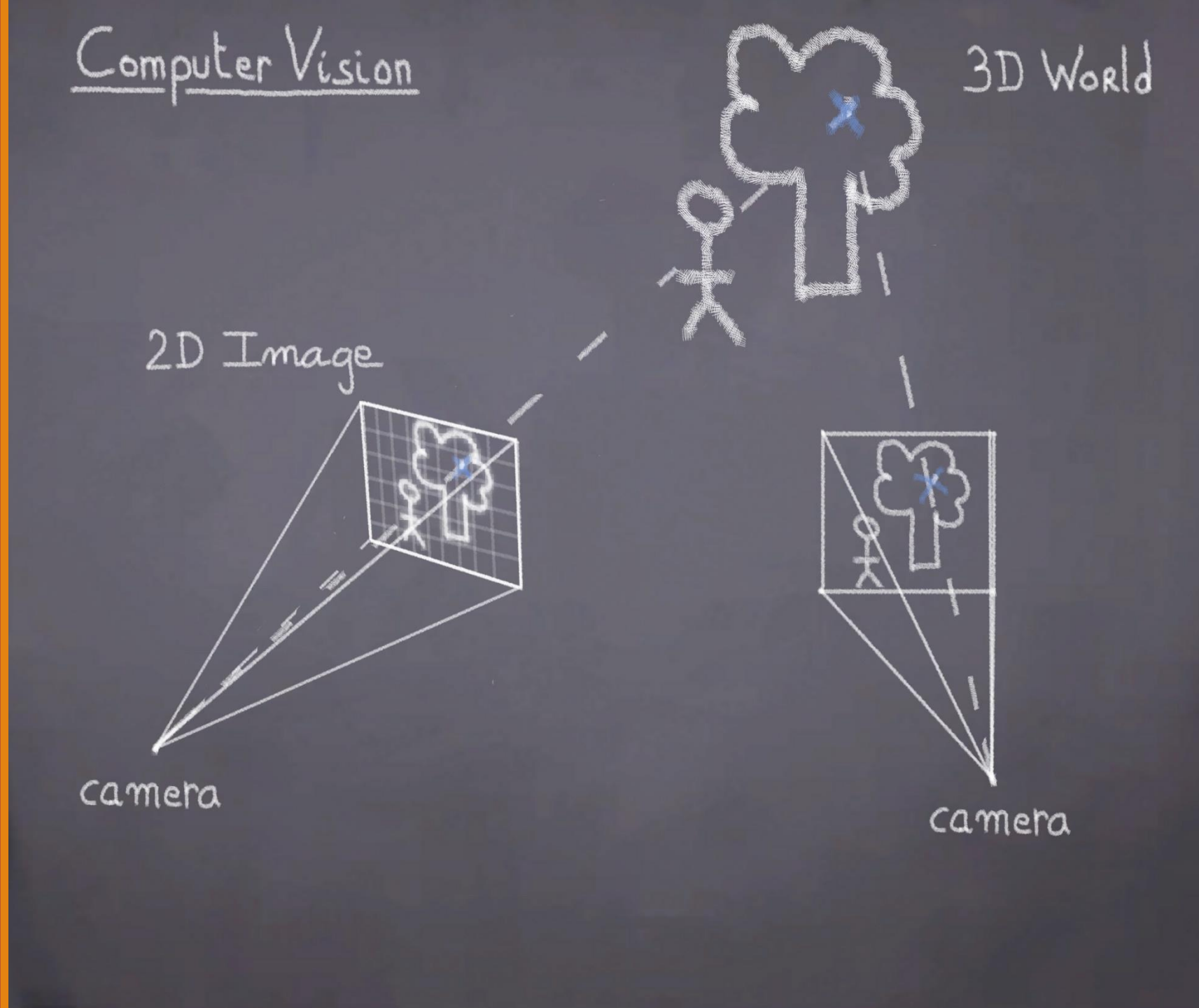
Lecture Overview

- Deep Learning in
  - Computer Vision
  - Natural Language Processing (NLP)
  - Speech
  - Robotics and AI
  - Music and the arts!
- A brief history of Neural Networks and Deep Learning
- Basics of Neural Networks

Who we are and how to reach us

- Taught by Efstratios Gavves, Max Welling, Patrick Putzky (Teaching Assistant)
- Course website
  - http://uvadlc.github.io
  - Lectures, lecture notes, practicals, extra info regarding the course content
- Classroom website
  - www.piazza.com/university_of_amsterdam/spring2016/uvadlc/home
  - Participate in a virtual classroom, ask questions to us and your fellow students, get in contact with others, check grades etc.
- Email us at
  - uva.deeplearning@gmail.com

Deep Learning in Computer Vision



UVA DEEP LEARNING COURSE
EFSTRATIOS GAVVES

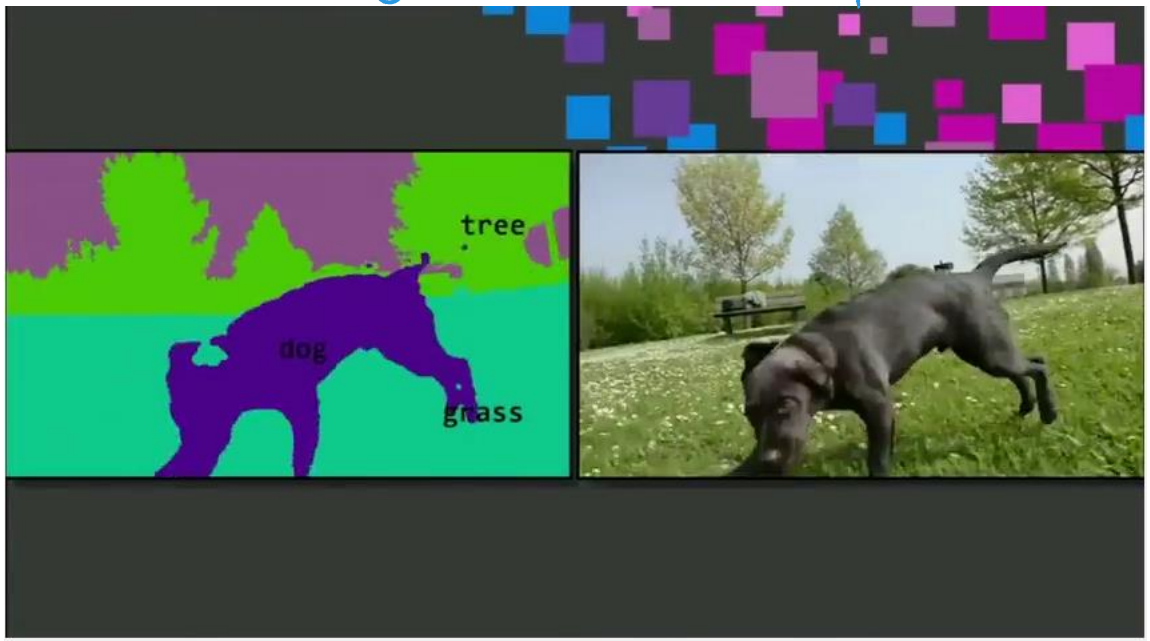
Object and activity recognition

Click to go to the video in Youtube

Large-scale Video Classification with Convolutional Neural Networks, CVPR 2014

Object detection and segmentation

Click to go to the video in Youtube



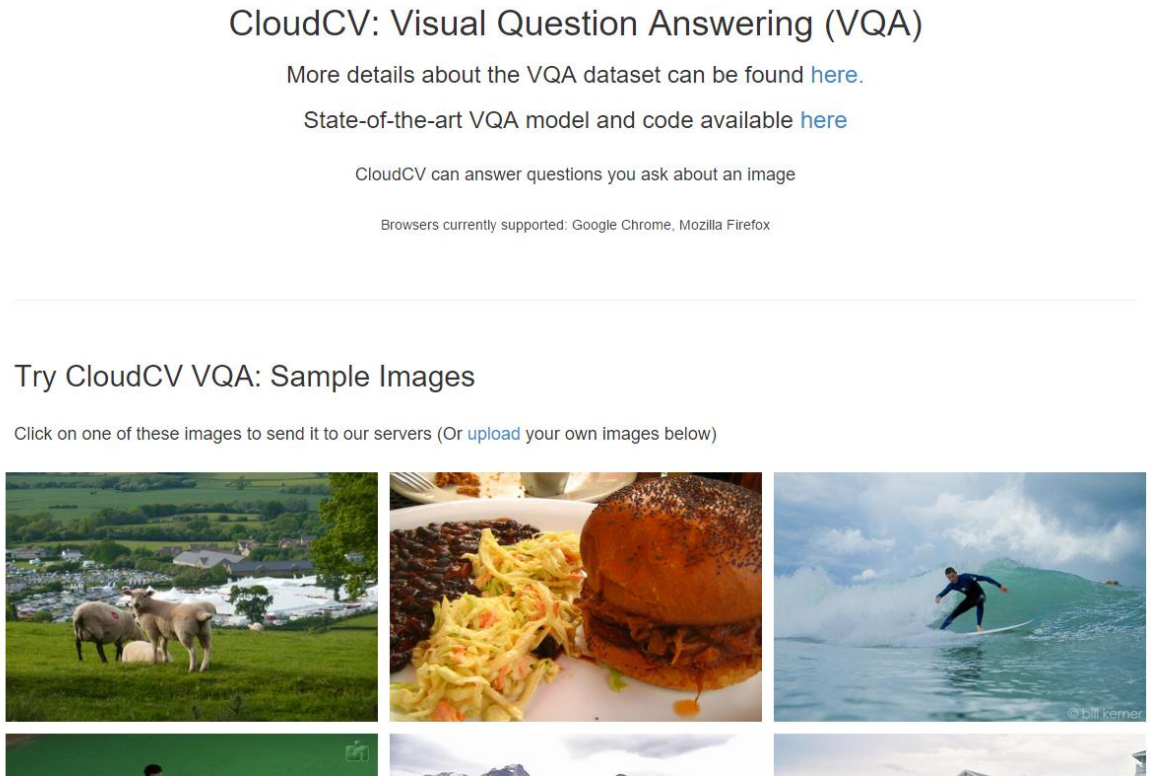
Microsoft Deep Learning Semantic Image Segmentation

Image captioning and Q&A

Click to go to the video in Youtube

NeuralTalk and Walk, recognition, text description of the image while walking

Click to go to the website



Why should we be impressed?

- Vision is ultra challenging!
  - For a small 256x256 resolution and for 256 pixel values
  - a total $2^{524,288}$ of possible images
  - In comparison there are about 10^{24}stars in the universe
- Visual object variations
  - Different viewpoints, scales, deformations, occlusions
- Semantic object variations
  - Intra-class variation
  - Inter-class overlaps

Deep Learning in Robotics

UVA DEEP LEARNING COURSE EFSTRATIOS GAVVES
INTRODUCTION ON DEEP LEARNING AND NEURAL NETWORKS - PAGE 12

Self-driving cars

Click to go to the video in Youtube

Self Driving Cars HD

Drones and robots

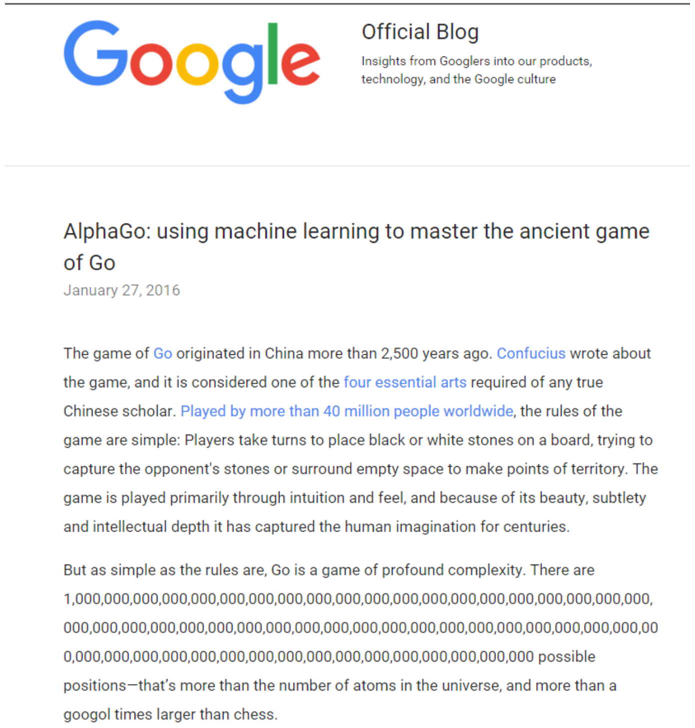
Click to go to the video in Youtube

Deep Sensorimotor Learning

Click to go to the video in Youtube

Stanford Autonomous Helicopter - Airshow #1

Game AI


Official Blog

Insights from Googlers into our products, technology, and the Google culture

AlphaGo: using machine learning to master the ancient game of Go

January 27, 2016

The game of Go originated in China more than 2,500 years ago. Confucius wrote about the game, and it is considered one of the four essential arts required of any true Chinese scholar. Played by more than 40 million people worldwide, the rules of the game are simple: Players take turns to place black or white stones on a board, trying to capture the opponent's stones or surround empty space to make points of territory. The game is played primarily through intuition and feel, and because of its beauty, subtlety and intellectual depth it has captured the human imagination for centuries.

But as simple as the rules are, Go is a game of profound complexity. There are 1,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000 possible positions—that's more than the number of atoms in the universe, and more than a googol times larger than chess.

[Click to go to the video in Youtube]

Google DeepMind's Deep Q-learning playing Atari Breakout

UVA DEEP LEARNING COURSE - EFSTRATIOS GAVVES & MAX WELLING
INTRODUCTION ON DEEP LEARNING AND NEURAL NETWORKS - PAGE 15

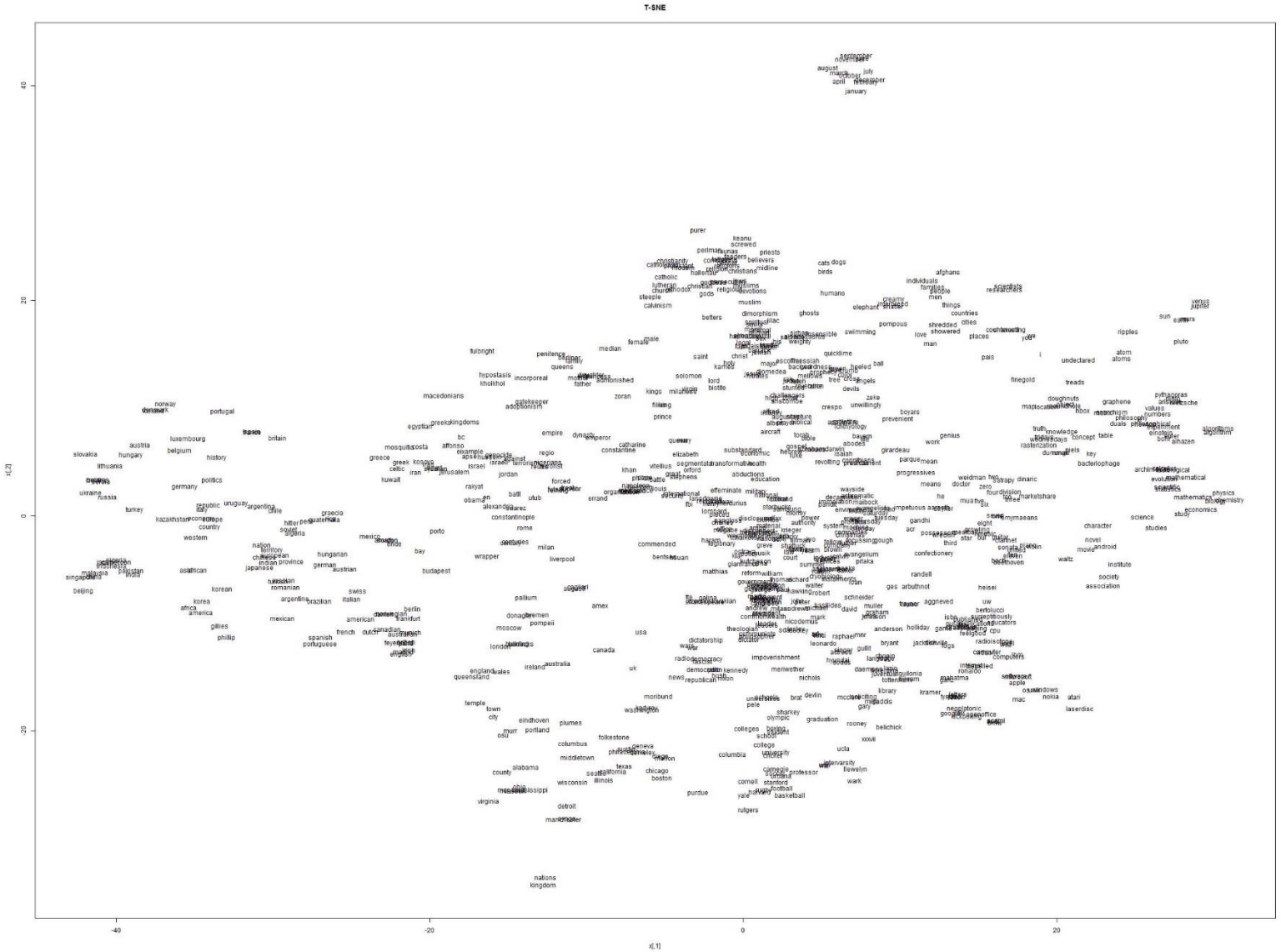
Why should we be impressed?

- Typically robotics are considered in controlled environments
  - Laboratory settings
  - Predictable positions
  - Standardized tasks (like in factory robots)
- What about real life situations?
  - Environments constantly change
  - New tasks need to be learnt without guidance
  - Unexpected factors must be dealt with
- Game AI
  - At least $10^{10^{48}}$ possible GO games. Where do we even start?

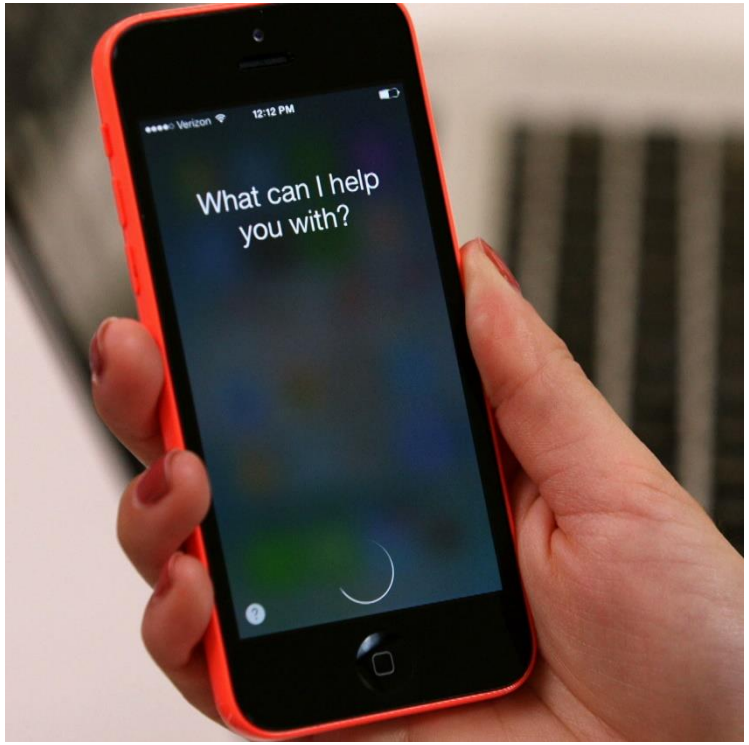
Deep Learning in NLP and Speech

UVA DEEP LEARNING COURSE
EFSTRATIOS GAVVES
INTRODUCTION ON DEEP LEARNING AND NEURAL NETWORKS - PAGE 17

Word and sentence representations



Speech recognition and Machine translation



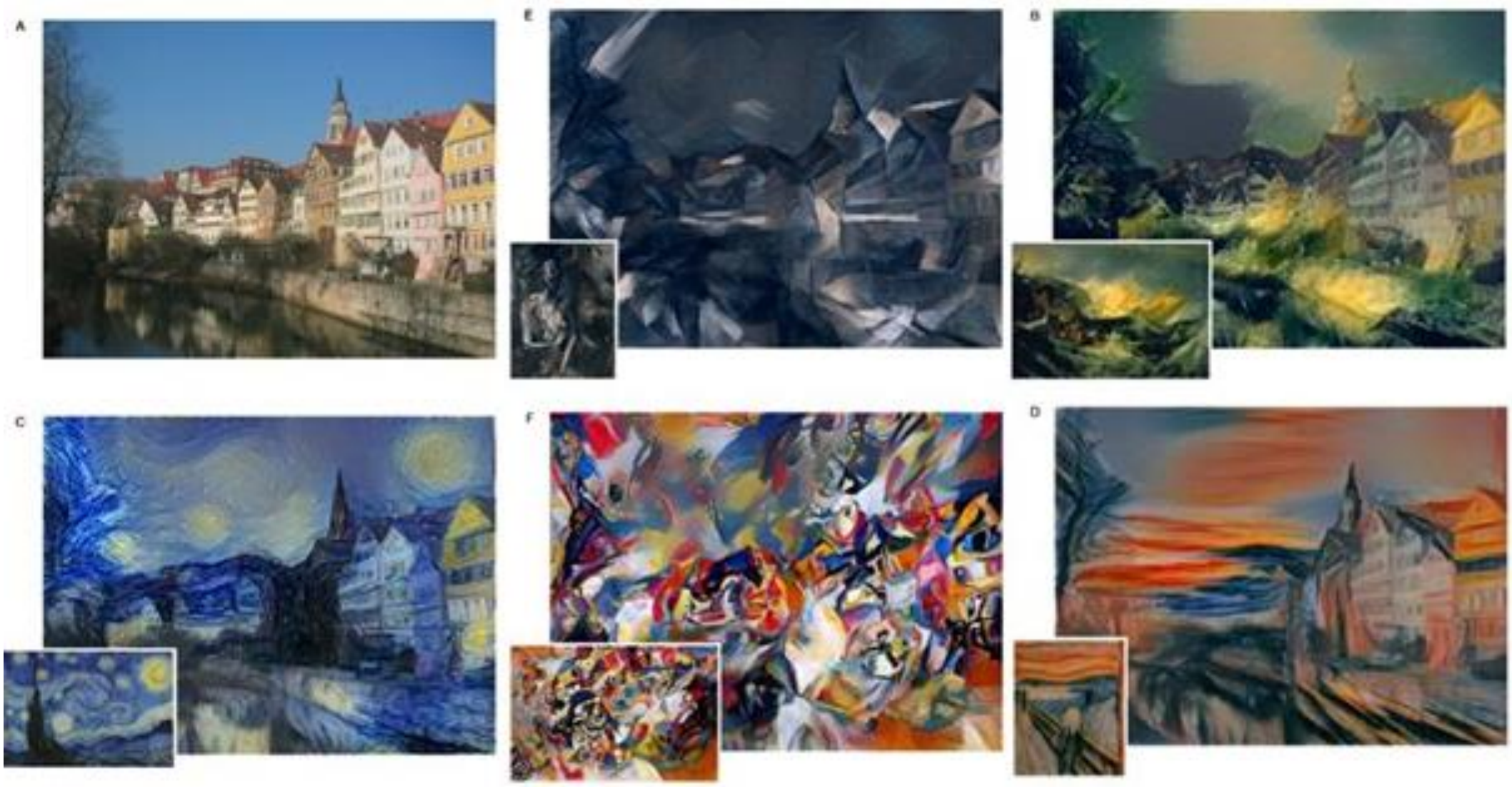
Why should we be impressed?

- NLP is an extremely complex task
  - synonymy (“chair”, “stool” or “beautiful”, “handsome”)
  - ambiguity (“I made her duck”, “Cut to the chase”)
- NLP is very high dimensional
  - assuming 150K english words, we need to learn 150K classifiers
  - with quite sparse data for most of them
- Beating NLP feels the closest to achieving true AI
  - although true AI probably goes beyond NLP, Vision, Robotics, … alone

Deep Learning in the arts

UVA DEEP LEARNING COURSE EFSTRATIOS GAVVES
INTRODUCTION ON DEEP LEARNING AND NEURAL NETWORKS - PAGE 21

Imitating famous painters

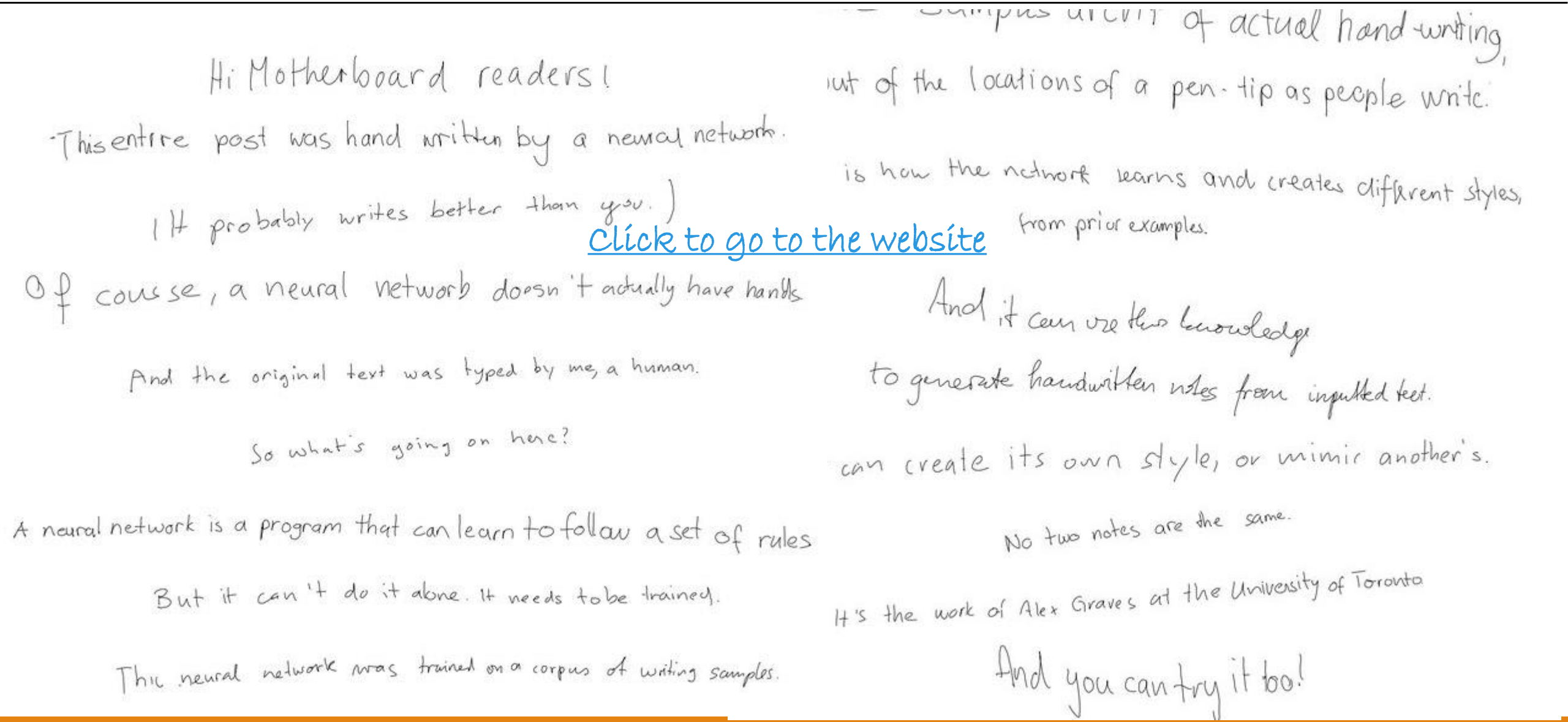


Or dreaming …

Click to go to the video in Youtube

Journey Through the Layers of the Mind

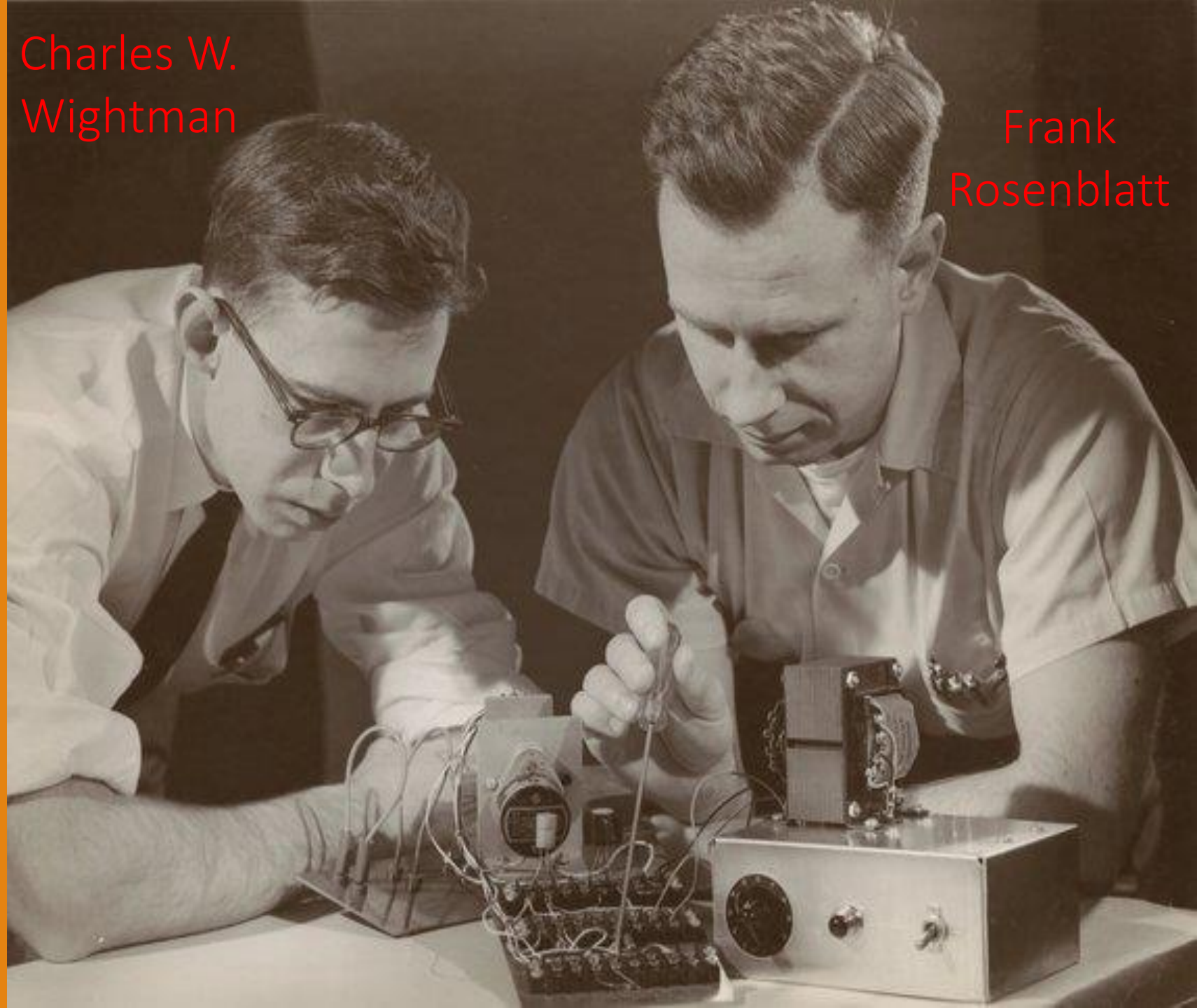
Handwriting



Why should we be impressed?

- Music, painting, etc. are tasks that are uniquely human
  - Difficult to model
  - Even more difficult to evaluate (if not impossible)
- If machines can generate novel pieces even remotely resembling art, they must have understood something about “beauty”, “harmony”, etc.
- Have they really learned to generate new art, however?
  - Or do they just fool us with their tricks?

A brief history of Neural Networks & Deep Learning

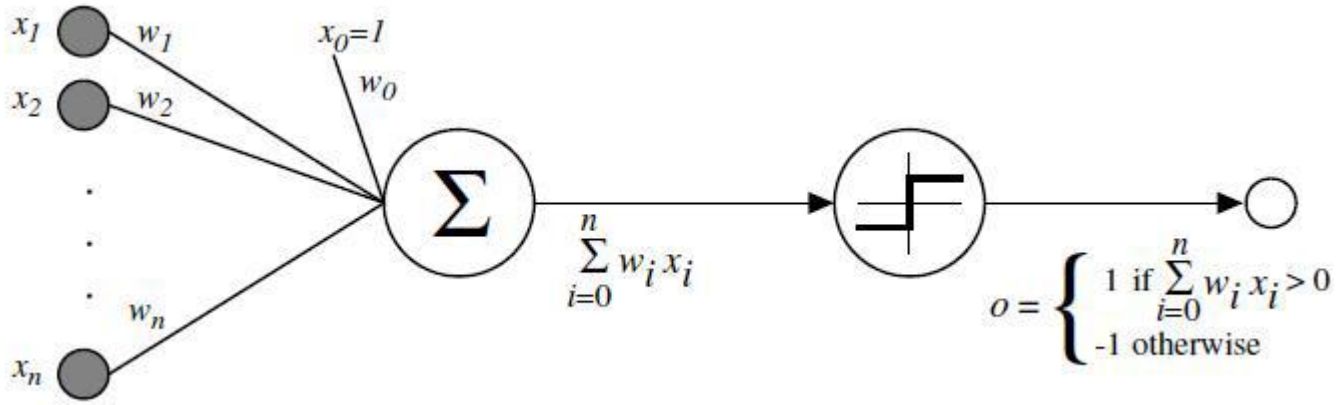


UVA DEEP LEARNING COURSE EFSTRATIOS GAVVES
INTRODUCTION ON DEEP LEARNING AND NEURAL NETWORKS - PAGE 26

First appearance (roughly)

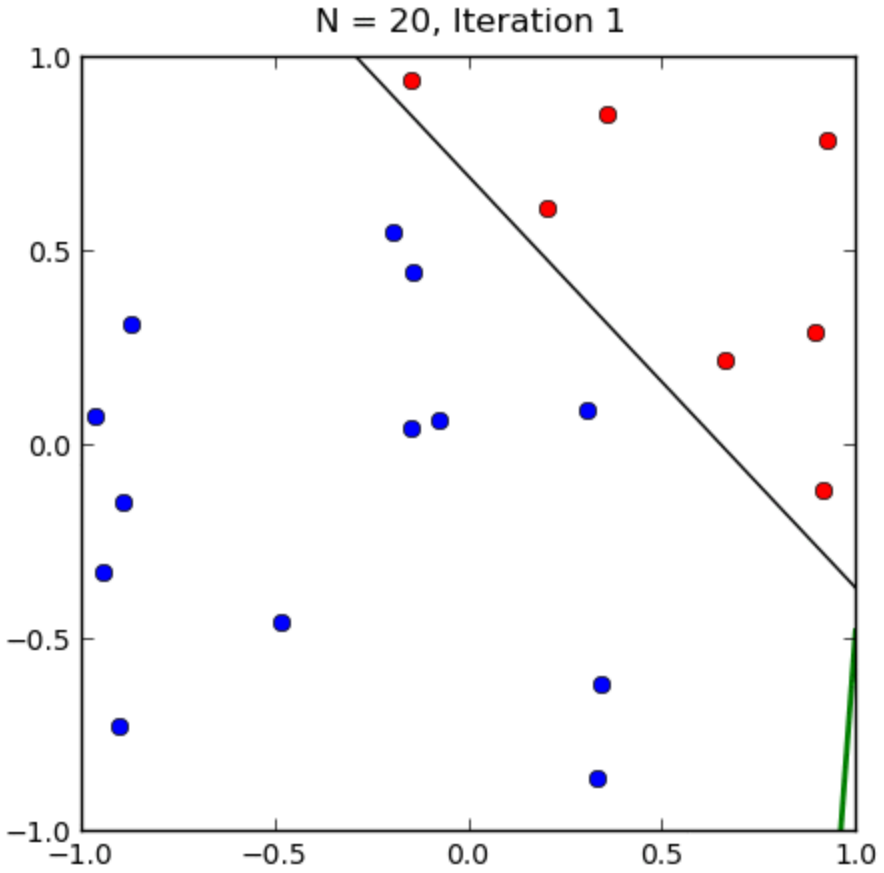
Perceptrons

- Rosenblatt proposed a machine for binary classifications
- Main idea
  - One weight w_i per input x_i
  - Multiply weights with respective inputs and add bias x_0 =+1
  - If result larger than threshold return 1, otherwise 0



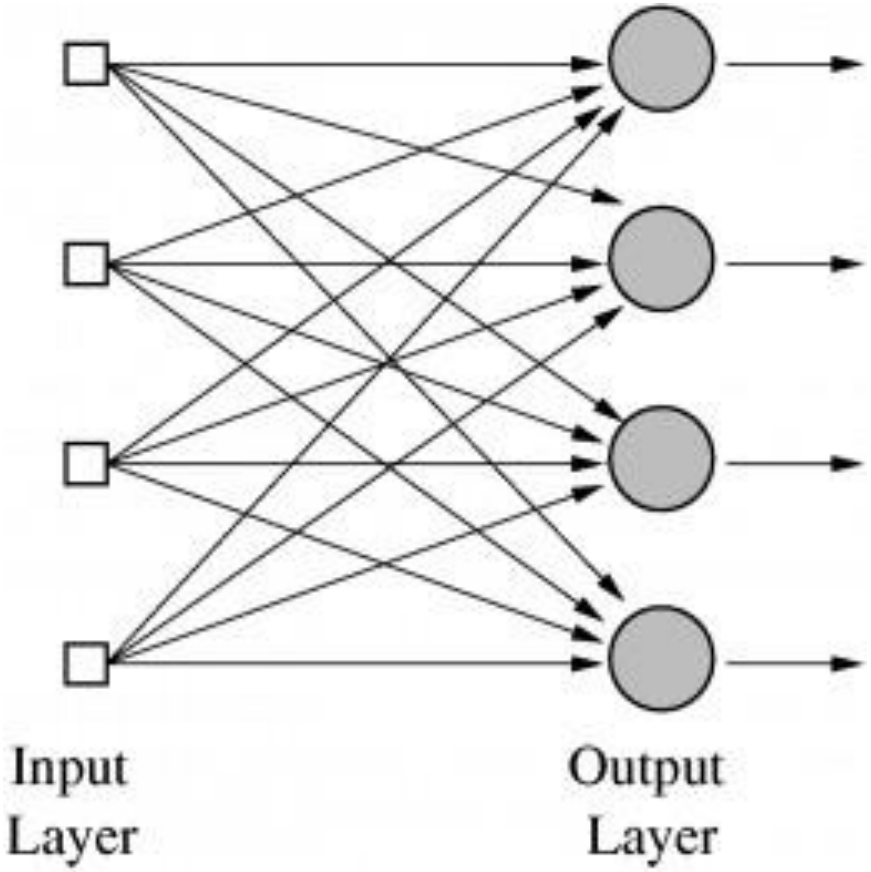
Training a perceptron

- Rosenblatt's innovation was mainly the learning algorithm for perceptrons
- Learning algorithm
  - Initialize weights randomly
  - Take one sample x_i and predict y_i
  - For erroneous predictions update weights
    - If the output was $\hat{y}_i = 0$ and $y_i = 1$, increase weights
    - If the output was $\hat{y}_i = 1$ and $y_i = 0$, decrease weights
  - Repeat until no errors are made



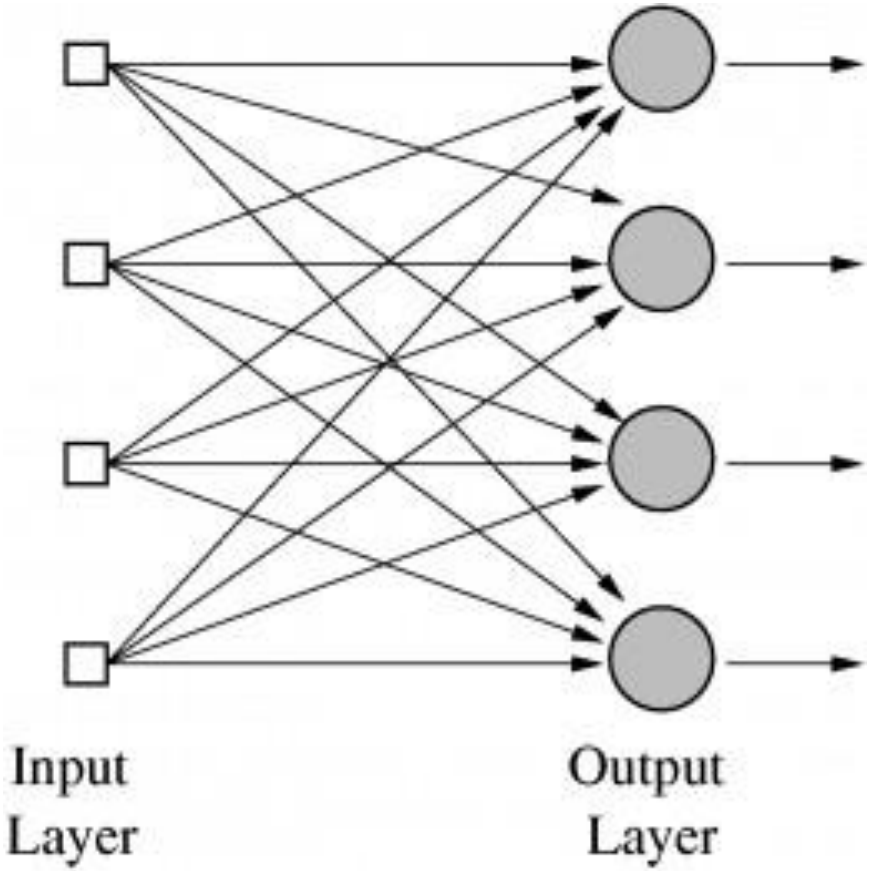
From a perceptron to a neural network

- One perceptron = one decision
- What about multiple decisions?
  - E.g. digit classification



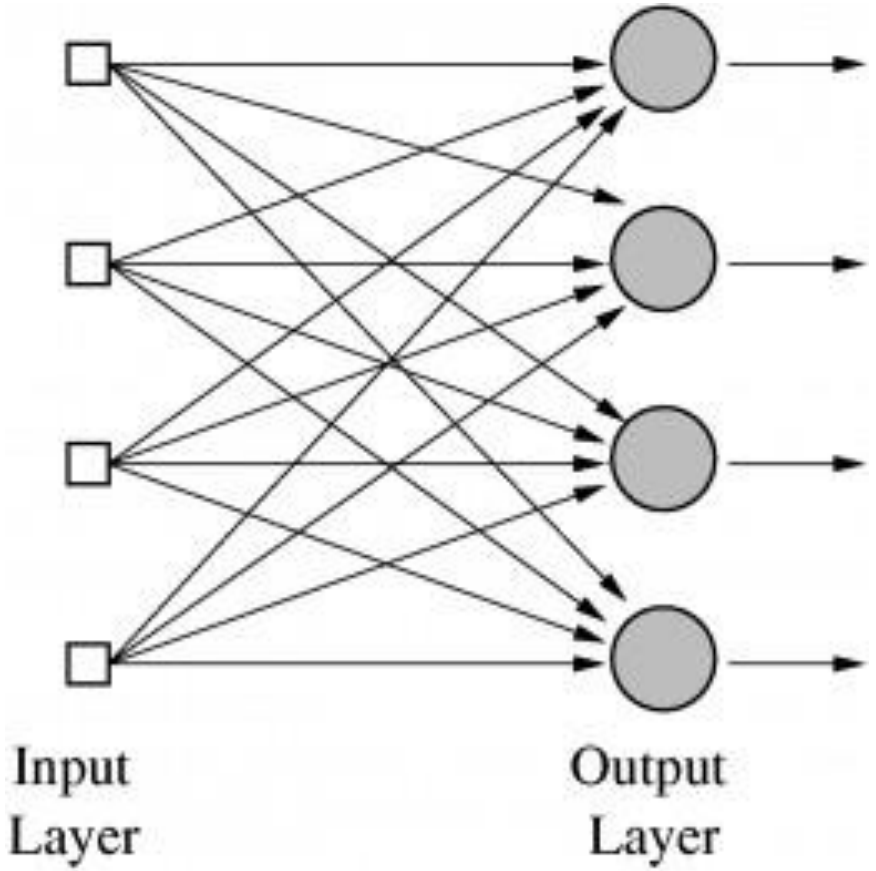
From a perceptron to a neural network

- One perceptron = one decision
- What about multiple decisions?
  - E.g. digit classification
- Stack as many outputs as the possible outcomes into a layer
  - Neural network



From a perceptron to a neural network

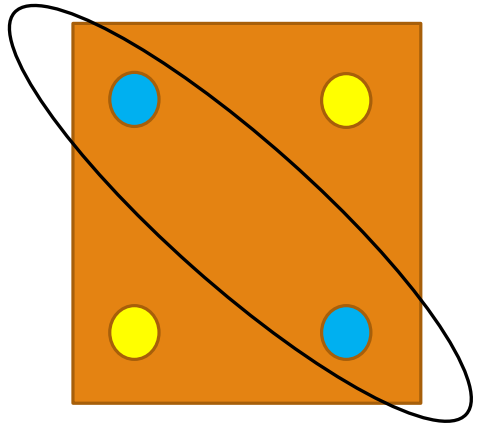
- One perceptron = one decision
- What about multiple decisions?
  - E.g. digit classification
- Stack as many outputs as the possible outcomes into a layer
  - Neural network
- Use one layer as input to the next layer
  - Multi-layer perceptron (MLP)

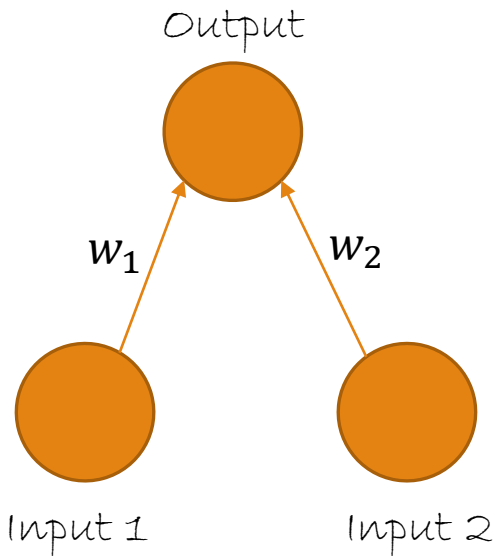


XOR & Multi-layer Perceptrons

- However, the exclusive or (XOR) cannot be solved by perceptrons
  - [Minsky and Papert, "Perceptrons", 1969]

    - $0\ w_1 + 0w_2 < \theta \rightarrow 0 < \theta$
    - $0\ w_1 + 1w_2 > \theta \rightarrow w_2 > \theta$
    - $1\ w_1 + 0w_2 > \theta \rightarrow w_1 > \theta$
    - $1\ w_1 + 1w_2 < \theta \rightarrow w_1 + w_2 < \theta$

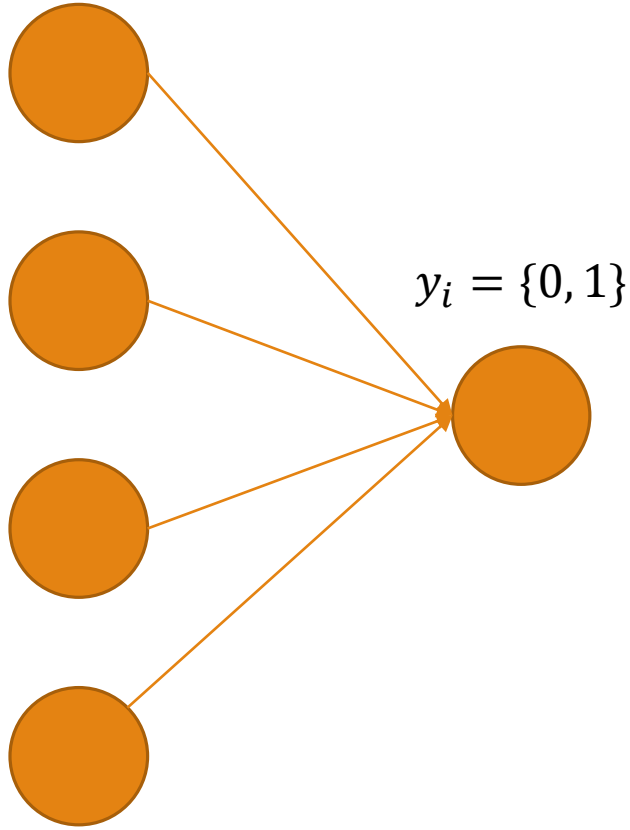




Input 1	Input 2	Output
1	1	0
1	0	1
0	1	1
0	0	0

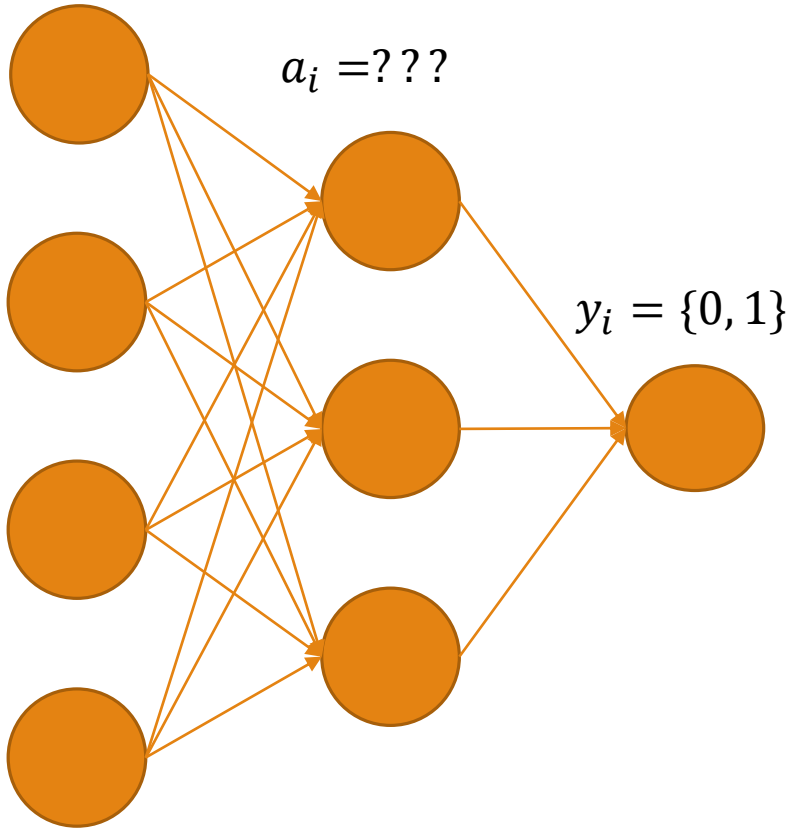
Minsky & Multi-layer perceptrons

- Interestingly, Minksy **never said** XOR cannot be solved by neural networks
  - Only that XOR cannot be solved with **1 layer** perceptrons
- Multi-layer perceptrons can solve XOR
  - 9 years earlier Minsky built such a multi-layer perceptron

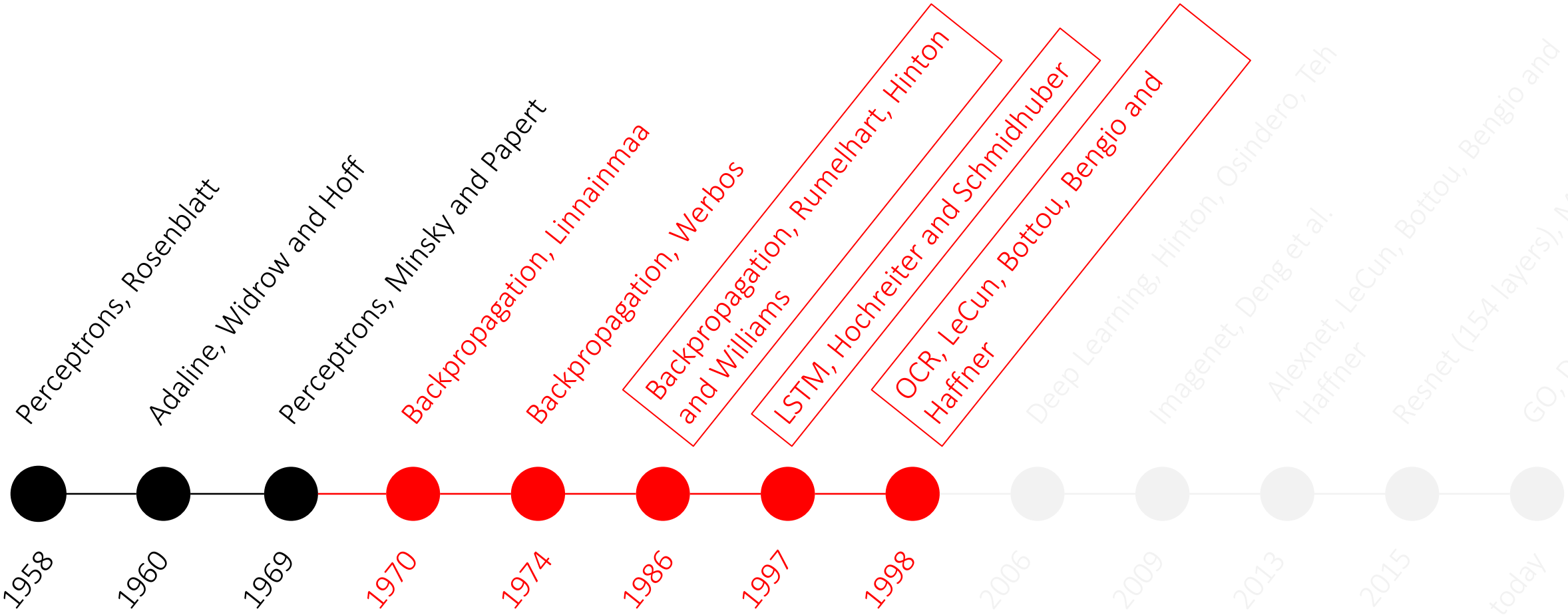


Minsky & Multi-layer perceptrons

- Interestingly, Minksy **never said** XOR cannot be solved by neural networks
  - Only that XOR cannot be solved with **1 layer** perceptrons
- Multi-layer perceptrons can solve XOR
  - 9 years earlier Minsky built such a multi-layer perceptron
- However, how to train a multi-layer perceptron?
- Rosenblatt's algorithm not applicable, as it expects to know the desired target
  - For hidden layers we cannot know the desired target



The “AI winter” despite notable successes



The first “AI winter”

- What everybody thought: “If a perceptron cannot even solve XOR, why bother?
  - Also, the exaggeration did not help (walking, talking robots were promised in the 60s)
- As results were never delivered, further funding was slushed, neural networks were damned and AI in general got discredited
- “The **AI winter is coming**”

The first “AI winter”

- What everybody thought: “If a perceptron cannot even solve XOR, why bother?
  - Also, the exaggeration did not help (walking, talking robots were promised in the 60s)
- As results were never delivered, further funding was slushed, neural networks were damned and AI in general got discredited
- “The **AI winter is coming**”
- Still, a few people persisted
- Significant discoveries were made, that laid down the road for today’s achievements

Backpropagation

- Learning multi-layer perceptrons now possible
  - XOR and more complicated functions can be solved
- Efficient algorithm
  - Process hundreds of example without a sweat
  - Allowed for complicated neural network architectures
- Backpropagation still is the backbone of neural network training today
- Digit recognition in cheques (OCR) solved before the 2000

Recurrent networks

- Traditional networks are “too plain”
  - Static Input → Processing → Static Output
- What about dynamic input
  - Temporal data, Language, Sequences

Recurrent networks

- Traditional networks are “too plain”
  - Static Input → Processing → Static Output
- What about dynamic input
  - Temporal data, Language, Sequences

What about inputs that appear in sequences, such as text? Could a neural network handle such modalities?

Recurrent networks

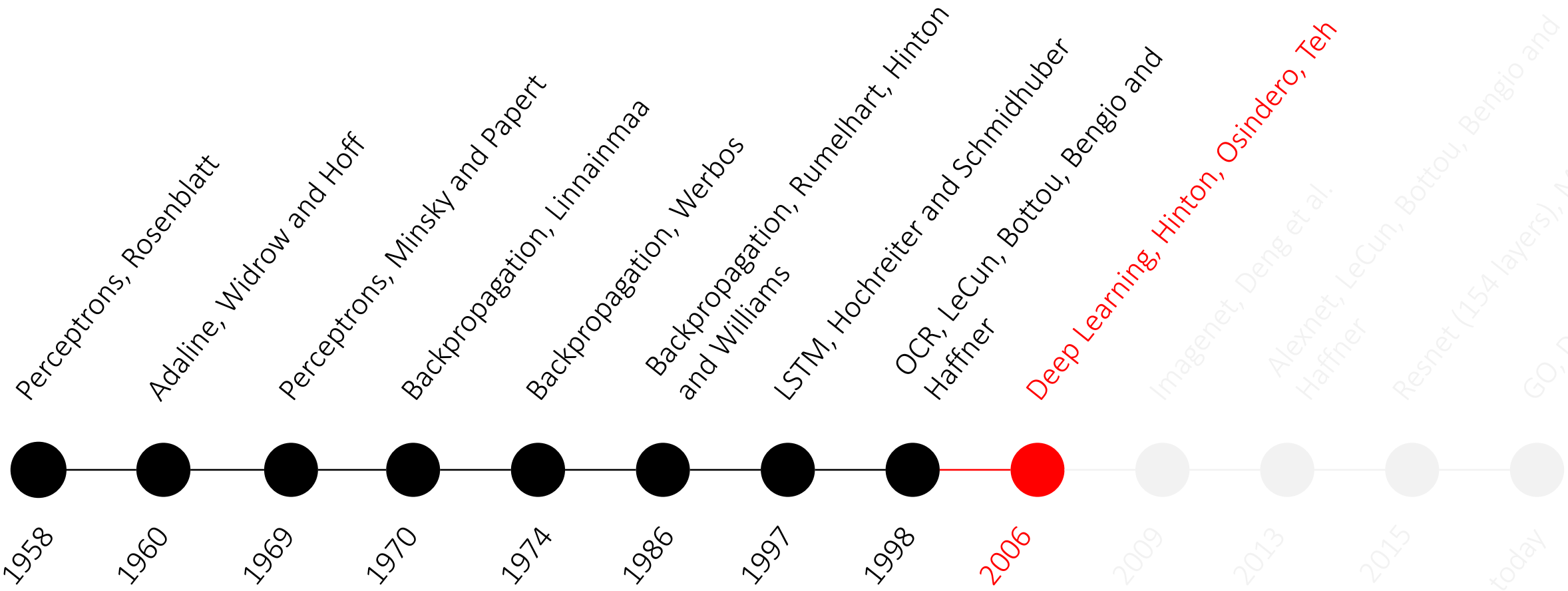
- Traditional networks are “too plain”
  - Static Input → Processing → Static Output
- What about dynamic input
  - Temporal data, Language, Sequences
- Memory is needed to “remember” state changes
  - Recurrent feedback connections
- What kind of memory
  - Long, Short?
  - Both! Long-short term memory networks (LSTM), Schmidhuber 1997

What about inputs that appear in sequences, such as text? Could a neural network handle such modalities?

The second “AI winter”

- Until 1998 some nice algorithms and methods were proposed
  - Backpropagation
  - Recurrent Long-Short Term Memory Networks
  - OCR with Convolutional Neural Networks
- However, at the same time Kernel Machines (SVM etc.) suddenly become very popular
  - They had similar accuracies in the same tasks, as neural networks could not improve beyond a few layers
  - Kernel Machines included much fewer heuristics driven with nice proofs on generalization
- As a result, once again the AI community turns away from Neural Networks

The thaw of the “AI winter”



Neural Network and Deep Learning problems

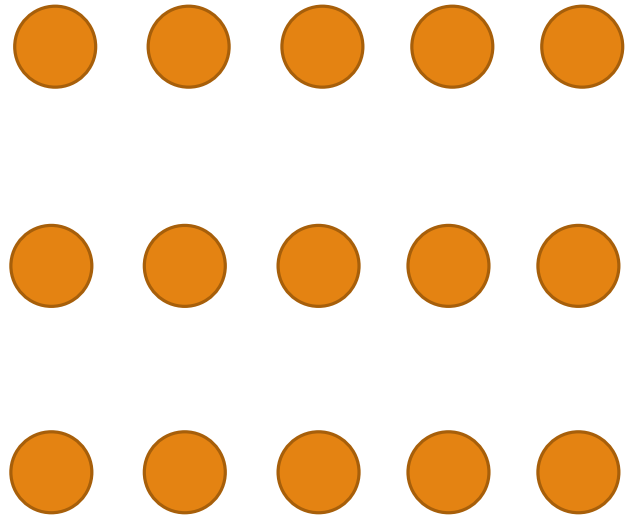
- Lack of processing power
  - No GPUs at the time
- Lack of data
  - No big, annotated datasets at the time
- Overfitting
  - Because of the above, models could not generalize all that well
- Vanishing gradient
  - While learning with NN, you need to multiply several numbers $a_1 \cdot a_2 \cdot \cdots \cdot a_n$.
  - If all are equal to 0.1, for $n = 10$ the result is 0.0000000001, smaller than our sensitivity threshold

Despite Backpropagation …

- Experimentally, training multi-layer perceptrons was not that useful
  - Accuracy didn't improve with more layers
- The inevitable question
  - Are 1-2 hidden layers the best neural networks can do?
  - Or is it that the learning algorithm is not really mature yet

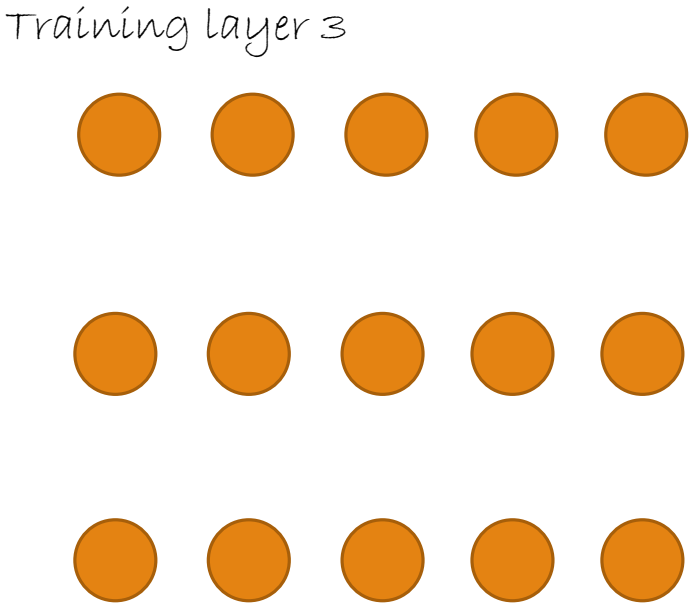
Deep Learning arrives

- Layer-by-layer training
  - The training of each layer individually is an easier undertaking
- Training multi-layered neural networks became easier
- Per-layer trained parameters initialize further training using contrastive divergence

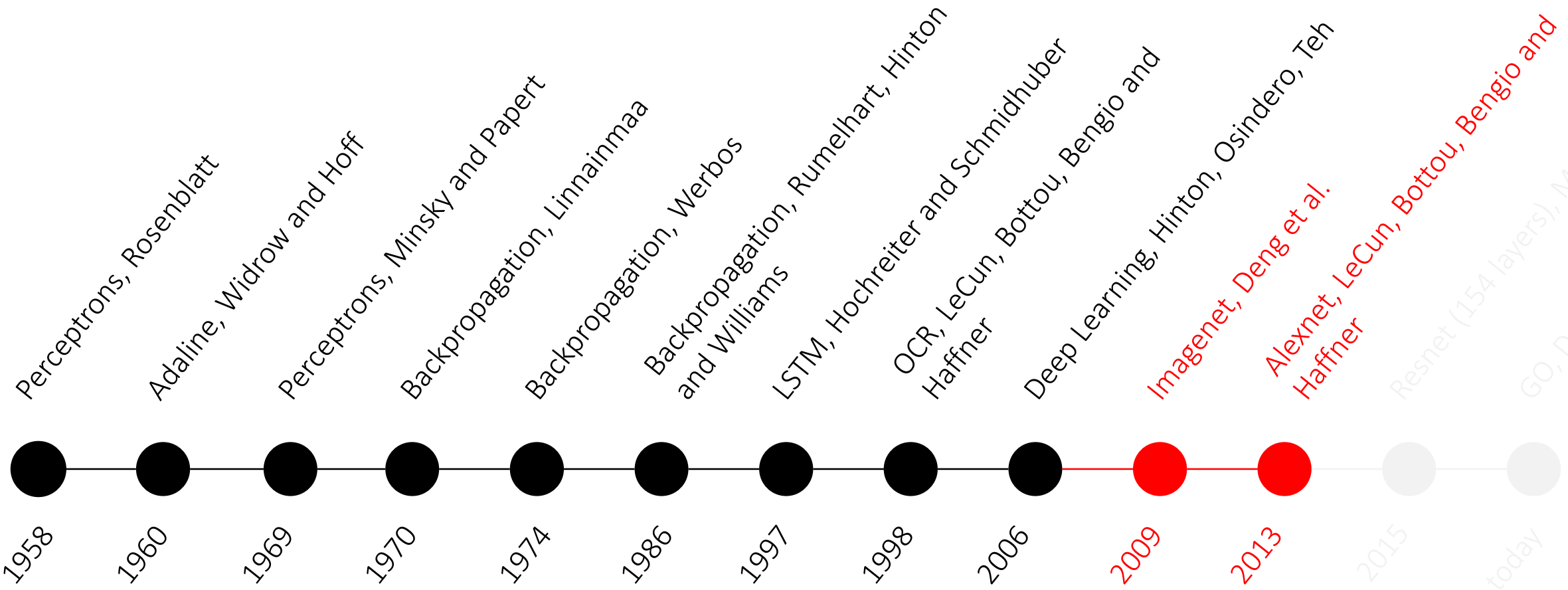


Deep Learning arrives

- Layer-by-layer training
  - The training of each layer individually is an easier undertaking
- Training multi-layered neural networks became easier
- Per-layer trained parameters initialize further training using contrastive divergence



Deep Learning Renaissance



More data, more …

- In 2009 the Imagenet dataset was published [Deng et al., 2009]
  - Collected images for each term of Wordnet (100,000 classes)
  - Tree of concepts organized hierarchically
    - “ambulance”, “Dalmatian dog”, “Egyptian cat”, …
  - About 16 million images annotated by humans
- Imagenet Large Scale Visual Recognition Challenge (ILSVRC)
  - 1 million images
  - 1,000 classes
  - Top-5 and top-1 error measured

Alexnet

- In 2013 Krizhevsky, Sutskever and Hinton re-implemented [Krizhevsky2013] a convolutional neural network [LeCun1998]
  - Trained on Imagenet, Two GPUs were used for the implementation
- Additional tricks
  - Rectified Linear Units (ReLU) instead of sigmoid or tanh
  - Dropout
  - Data augmentation
- In the 2013 Imagenet Workshop a legendary turmoil
  - Blasted competitors by an impressive 16% top-5 error, Second best around 26%
  - Most didn't even think of NN as remotely competitive
- At the same time similar results in the speech recognition community
  - One of G. Hinton students collaboration with Microsoft Research, improving state-of-the-art by an impressive amount after years of incremental improvements [Hinton2012]

Alexnet architecture

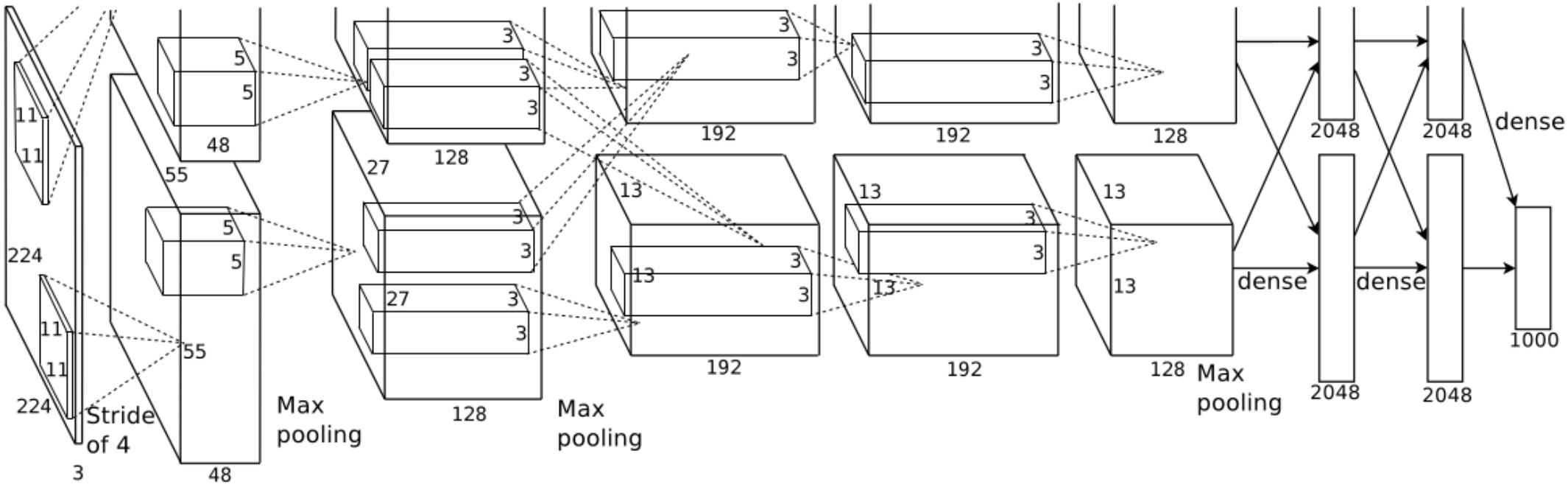
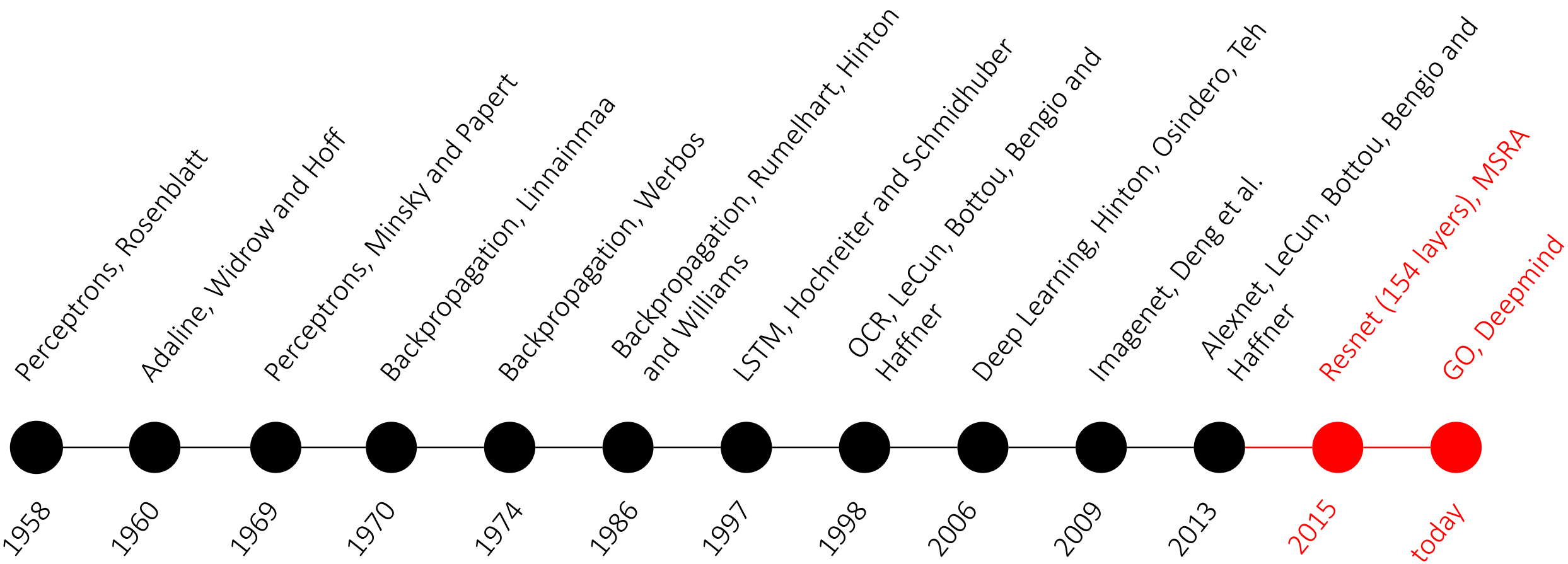


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

Deep Learning Golden Era



The today

- Deep Learning is almost everywhere
  - Object classification
  - Object detection, segmentation, pose estimation
  - Image captioning, question answering
  - Machine translation
  - Speech recognition
  - Robotics
- Some strongholds
  - Action classification, action detection
  - Object retrieval
  - Object tracking

The ILSVC Challenge over the last three years

CNN based, non-CNN based

2012 Teams	%error
Supervision (Toronto)	15.3
ISI (Tokyo)	26.1
VGG (Oxford)	26.9
XRCE/INRIA	27.0
UvA (Amsterdam)	29.6
INRIA/LEAR	33.4

2013 Teams	%error
Clarifai (NYU spinoff)	11.7
NUS (singapore)	12.9
Zeiler-Fergus (NYU)	13.5
A. Howard	13.5
OverFeat (NYU)	14.1
UvA (Amsterdam)	14.2
Adobe	15.2
VGG (Oxford)	15.2
VGG (Oxford)	23.0

2014 Teams	%error
GoogLeNet	6.6
VGG (Oxford)	7.3
MSRA	8.0
A. Howard	8.1
DeeperVision	9.5
NUS-BST	9.7
TTIC-ECP	10.2
XYZ	11.2
UvA	12.1

Figures taken from Y. LeCun's CVPR 2015 plenary talk

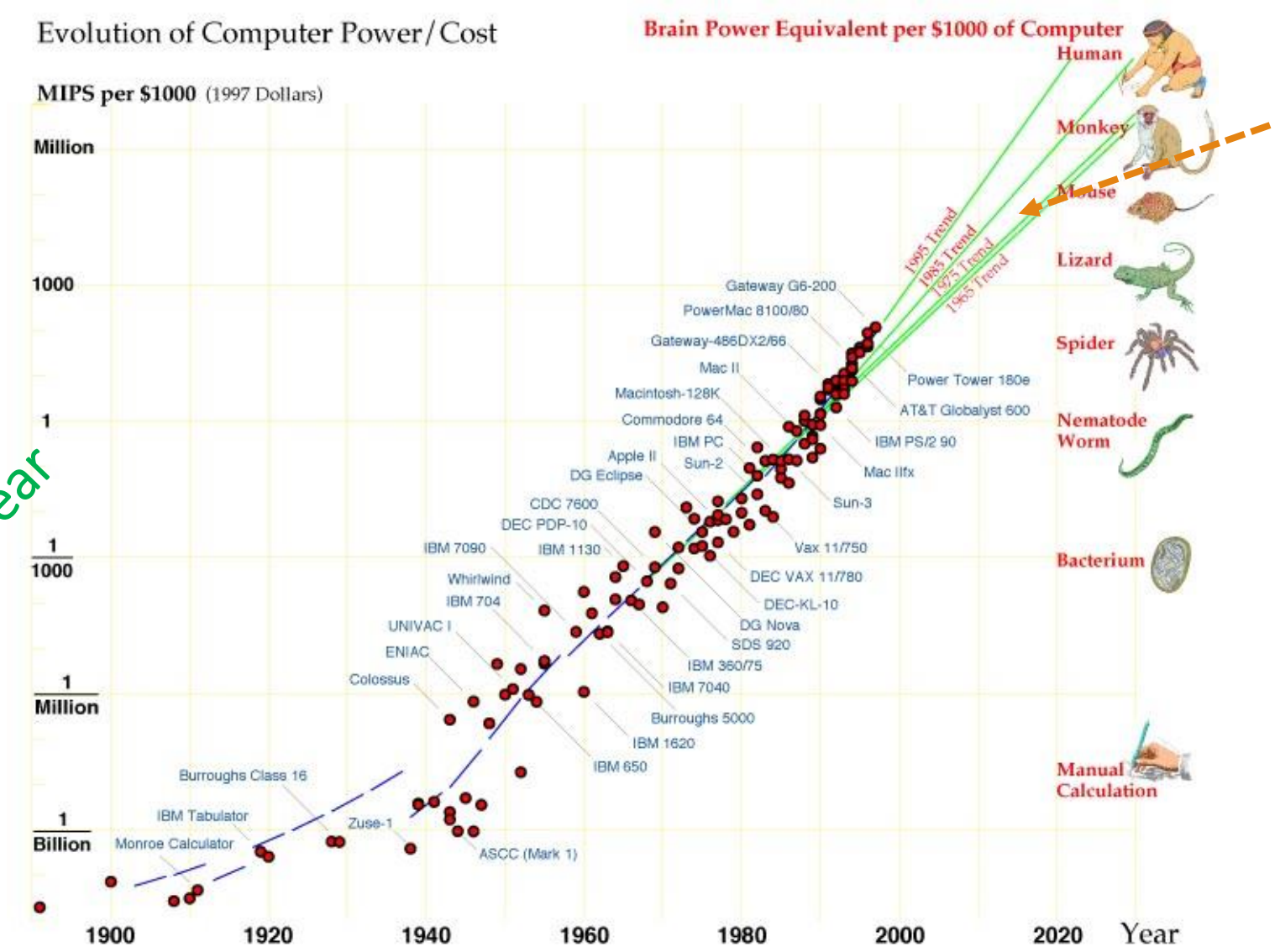
Latest 2015 ILSVC Challenge

- Microsoft Research Asia won the competition with a legendary 150-layered network
- Almost superhuman accuracy: only 3.5% error
- In comparison last year's GoogLeNet had 22 layers

Google Deepmind’s AlphaGO

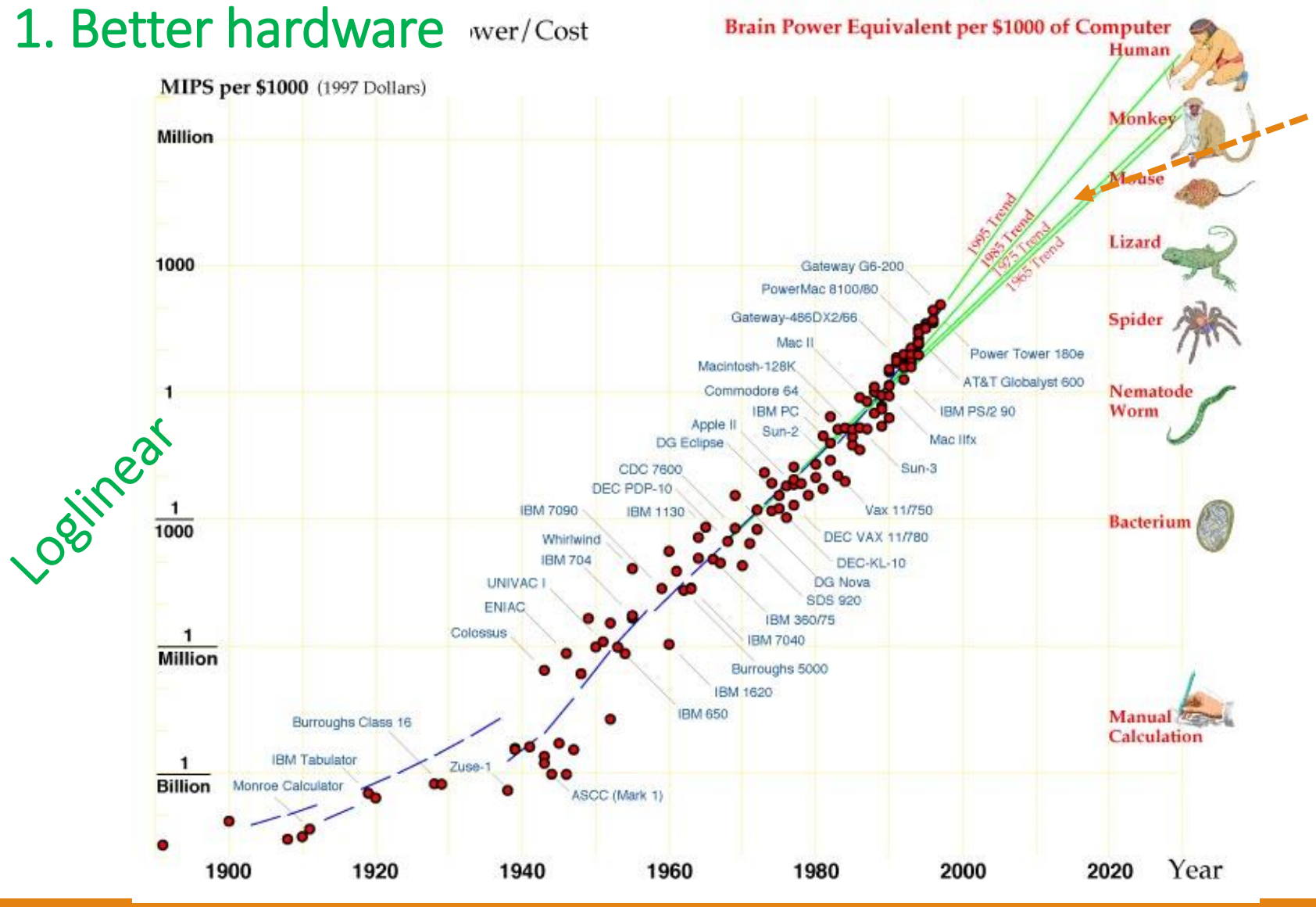
- Last week Google Deepmind beat the European champion in GO
- GO is the last stronghold where humans still beat the computers
  - 1,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000, 000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,00 0,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000 possible positions
  - Still, doubts are casted whether the game has been “mastered”

So, why now?


Evolution of Computer Power/Cost
MIPS per $1000 (1997 Dollars)
Million
1000
1
1/1000
1/Million
1/Billion
1900
1920
1940
1960
1980
2000
2020
Year
Brain Power Equivalent per $1000 of Computer
Human
Monkey
Mouse
Lizard
Spider
Nematode Worm
Bacterium
Manual Calculation
1995 Trend
1985 Trend
1975 Trend
1965 Trend
Gateway G6-200
PowerMac 8100/80
Gateway-486DX2/66
Mac II
Macintosh-128K
Commodore 64
IBM PC
Apple II
DG Eclipse
Sun-2
CDC 7600
DEC PDP-10
IBM 7090
IBM 1130
Whirlwind
IBM 704
UNIVAC I
ENIAC
Colossus
Burroughs Class 16
IBM Tabulator
Monroe Calculator
Zuse-1
ASCC (Mark 1)
IBM 650
IBM 1620
Burroughs 5000
IBM 7040
IBM 360/75
SDS 920
DG Nova
DEC-KL-10
DEC VAX 11/780
Vax 11/750
Sun-3
Mac IIfx
IBM PS/2 90
AT&T Globalyst 600
Power Tower 180e
Loglinear

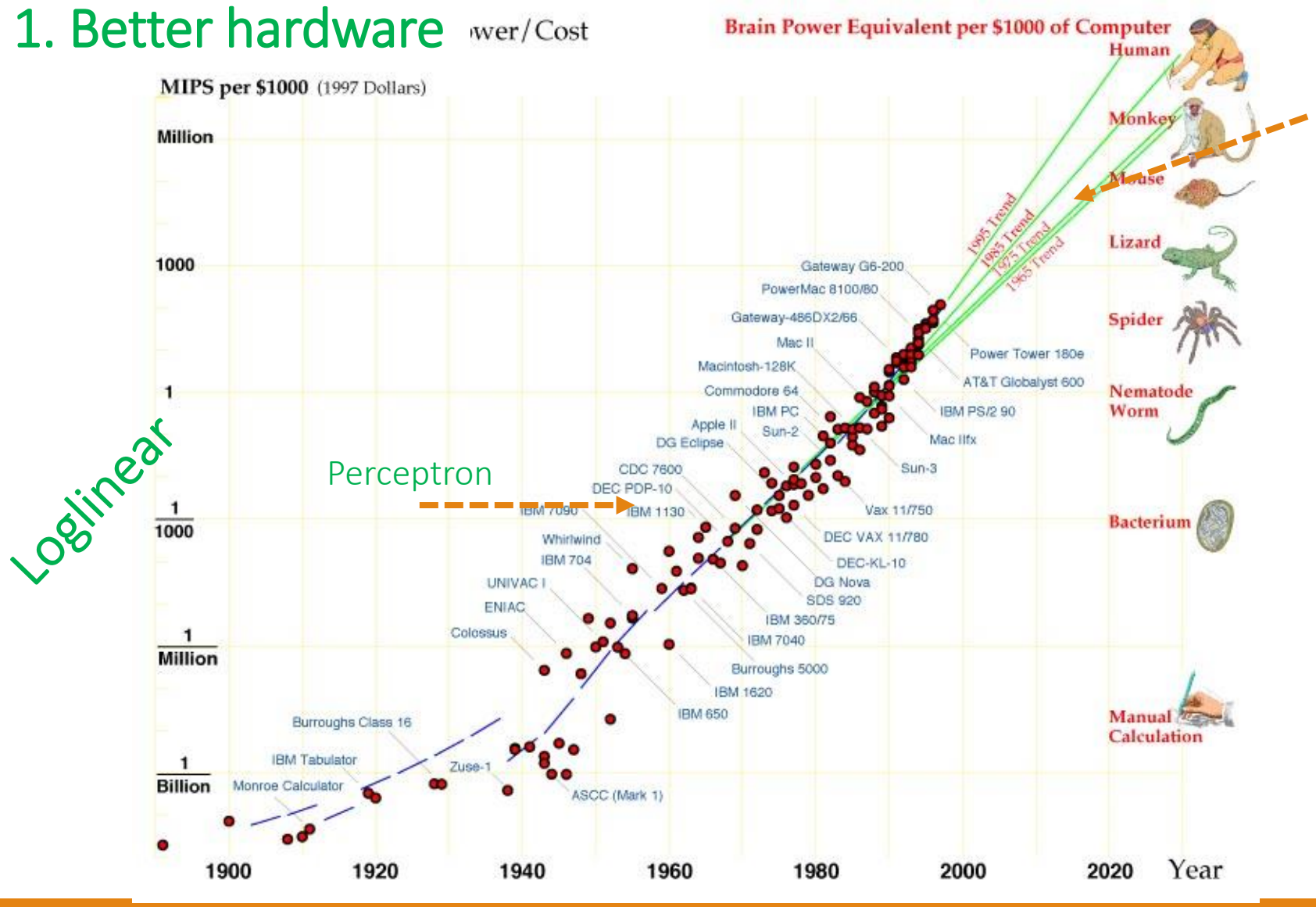
So, why now?

1. Better hardware



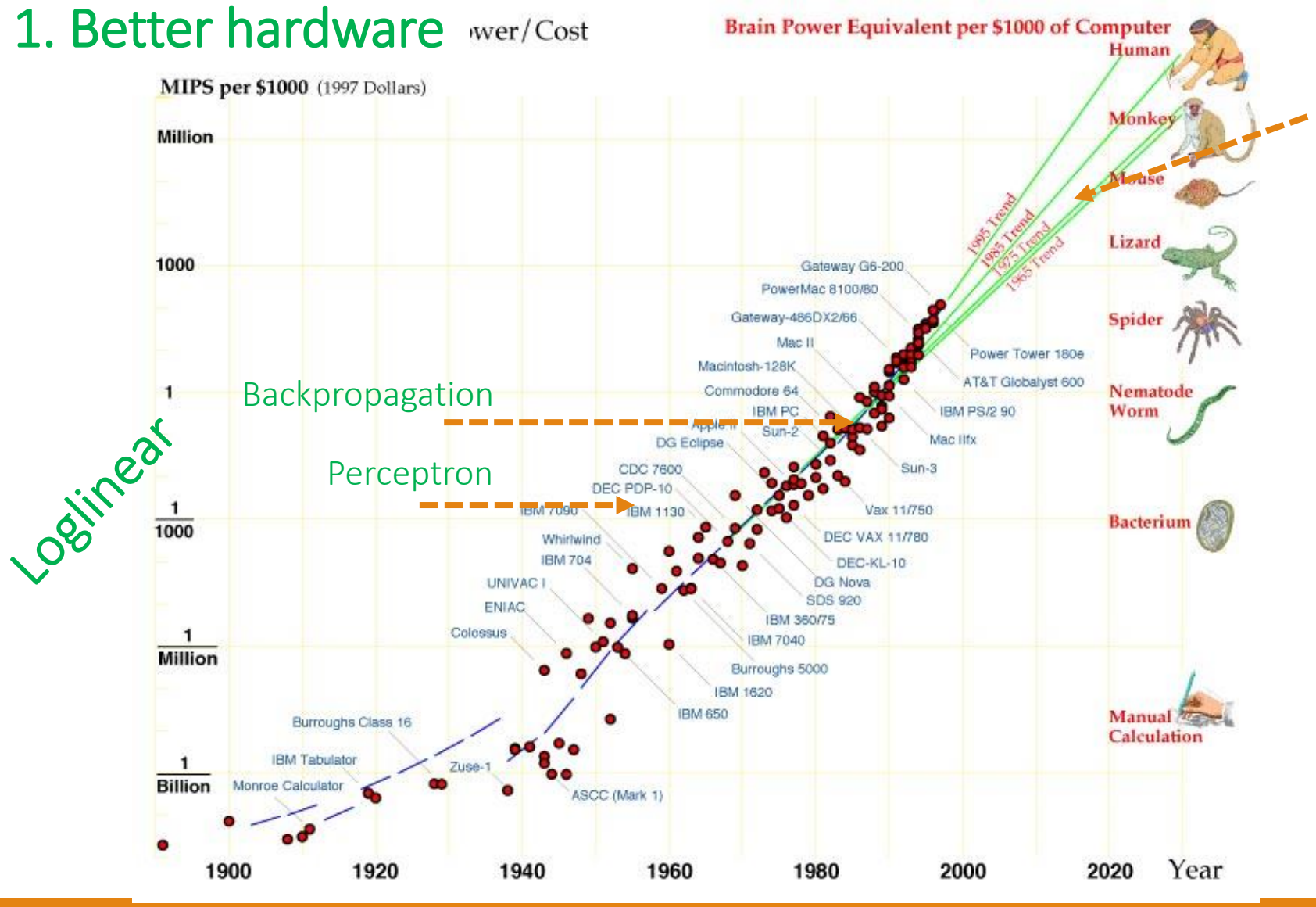
So, why now?

1. Better hardware



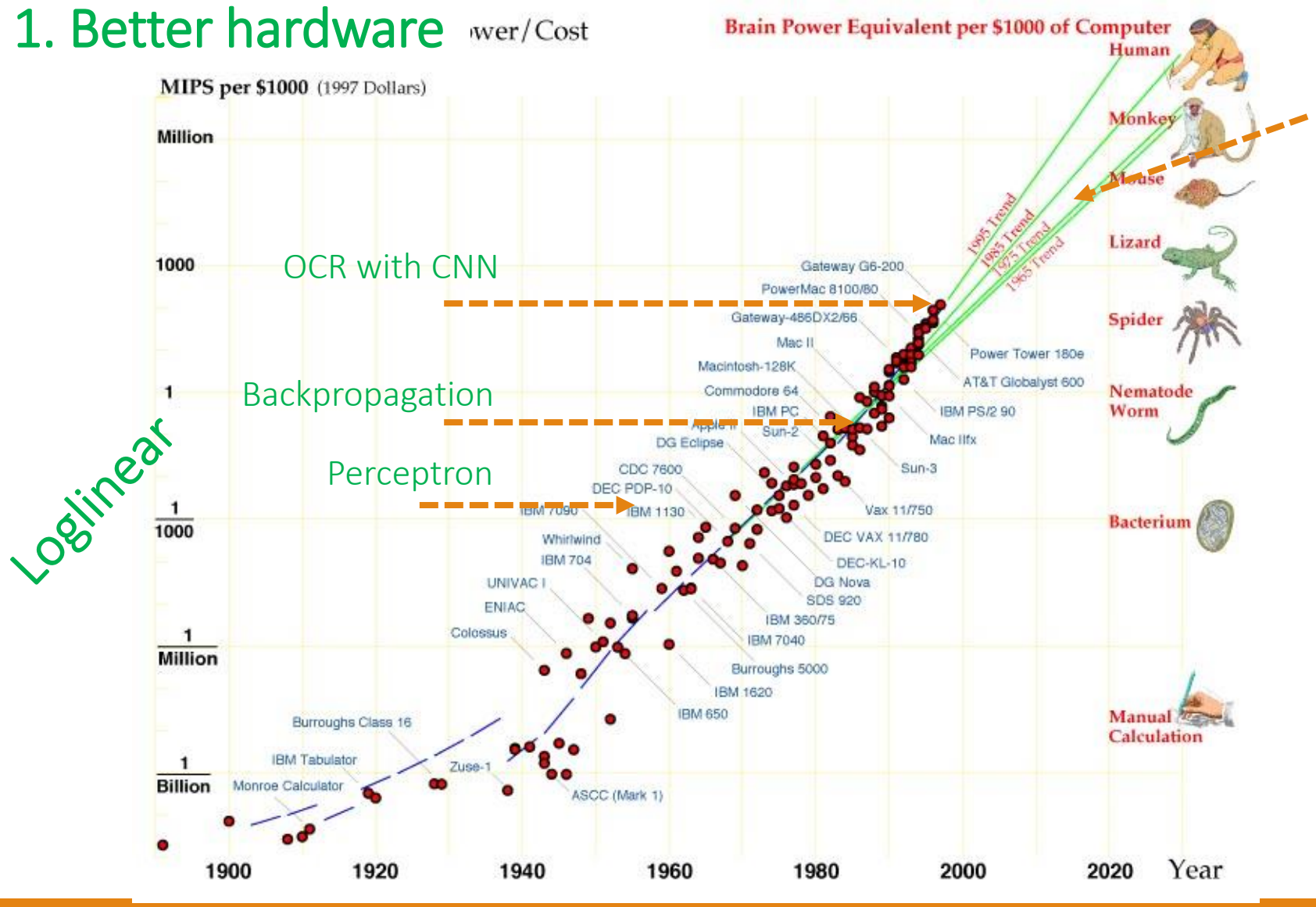
So, why now?

1. Better hardware



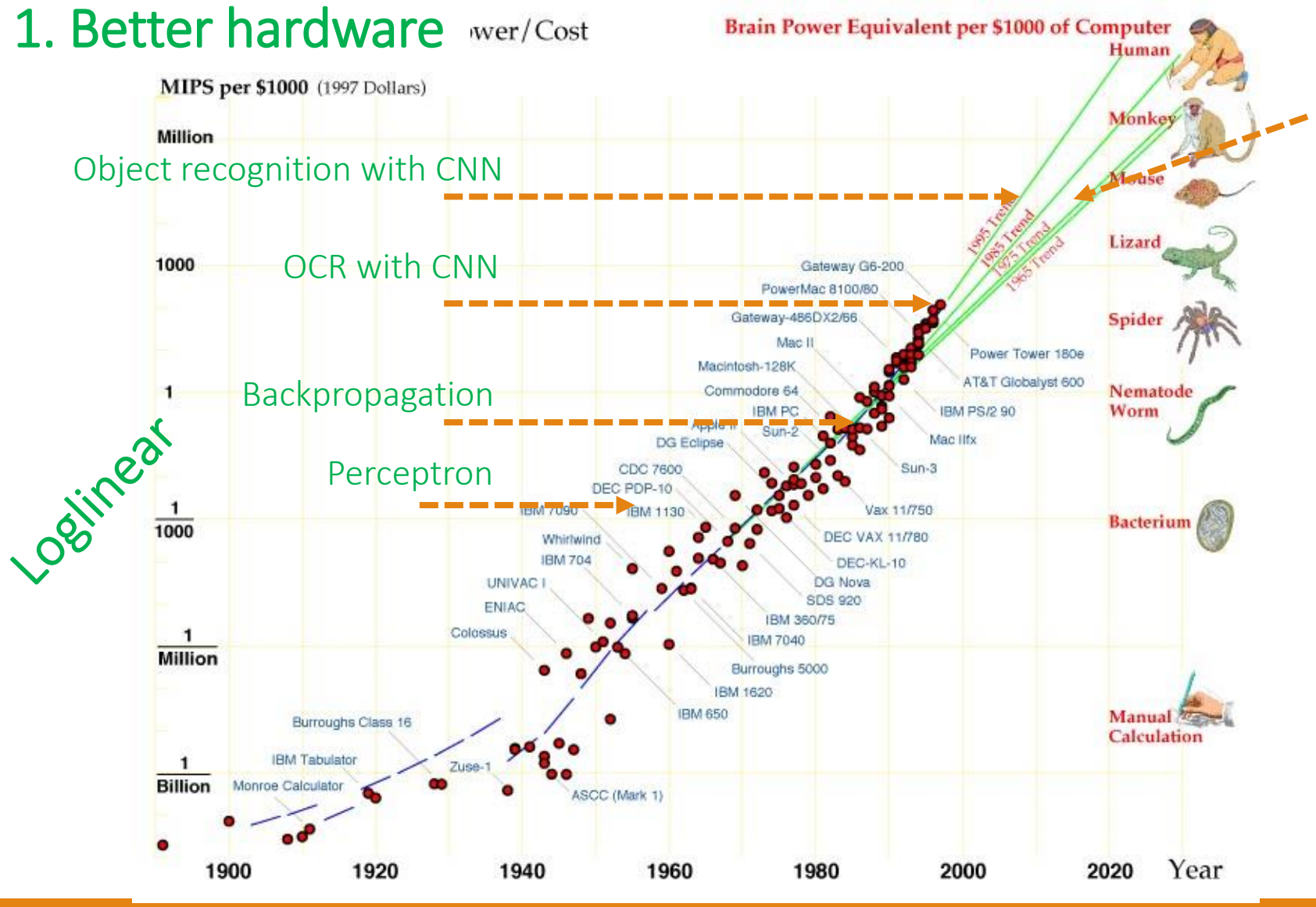
So, why now?

1. Better hardware



So, why now?

1. Better hardware



Object recognition with CNN

OCR with CNN

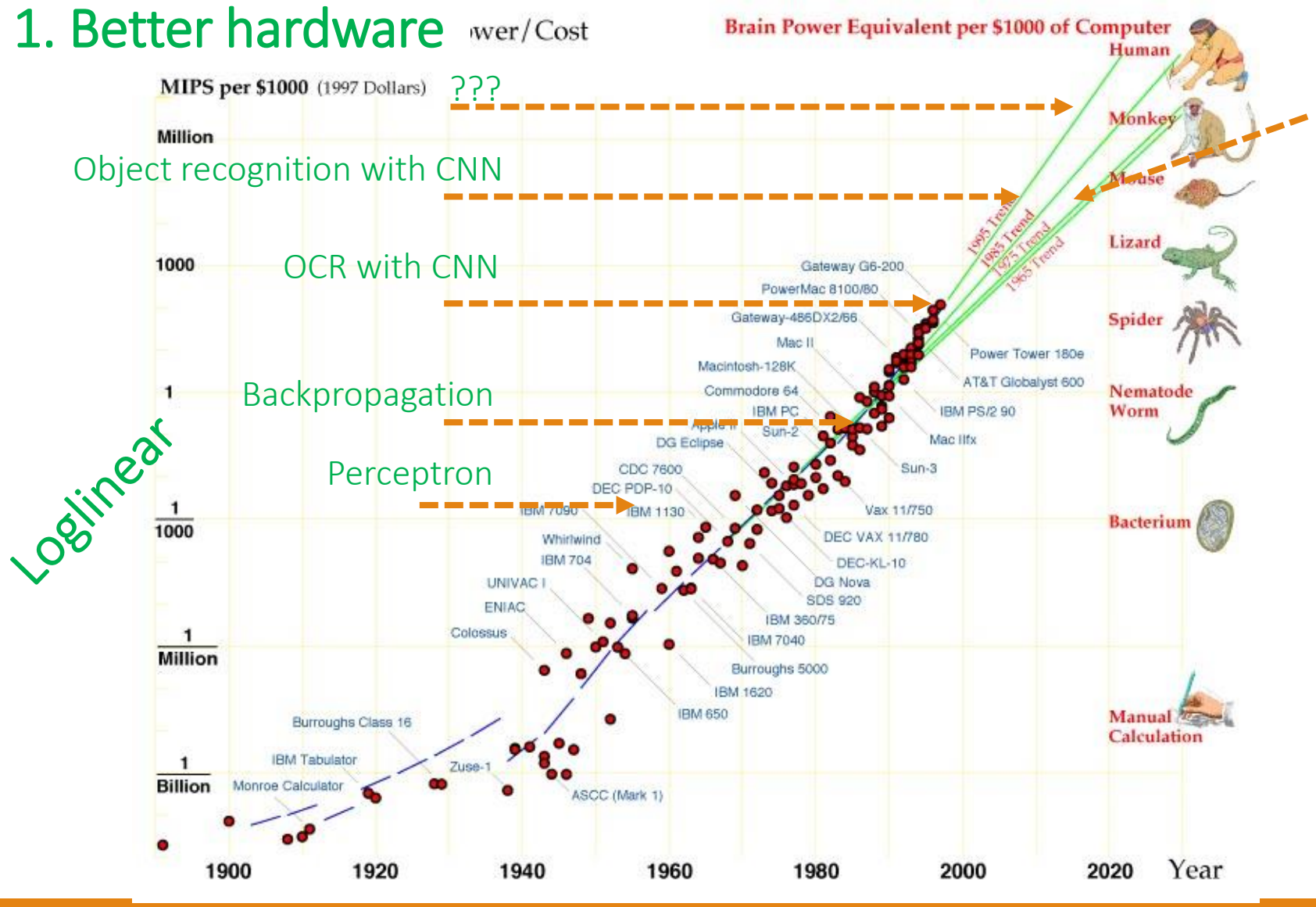
Backpropagation

Perceptron

Loglinear

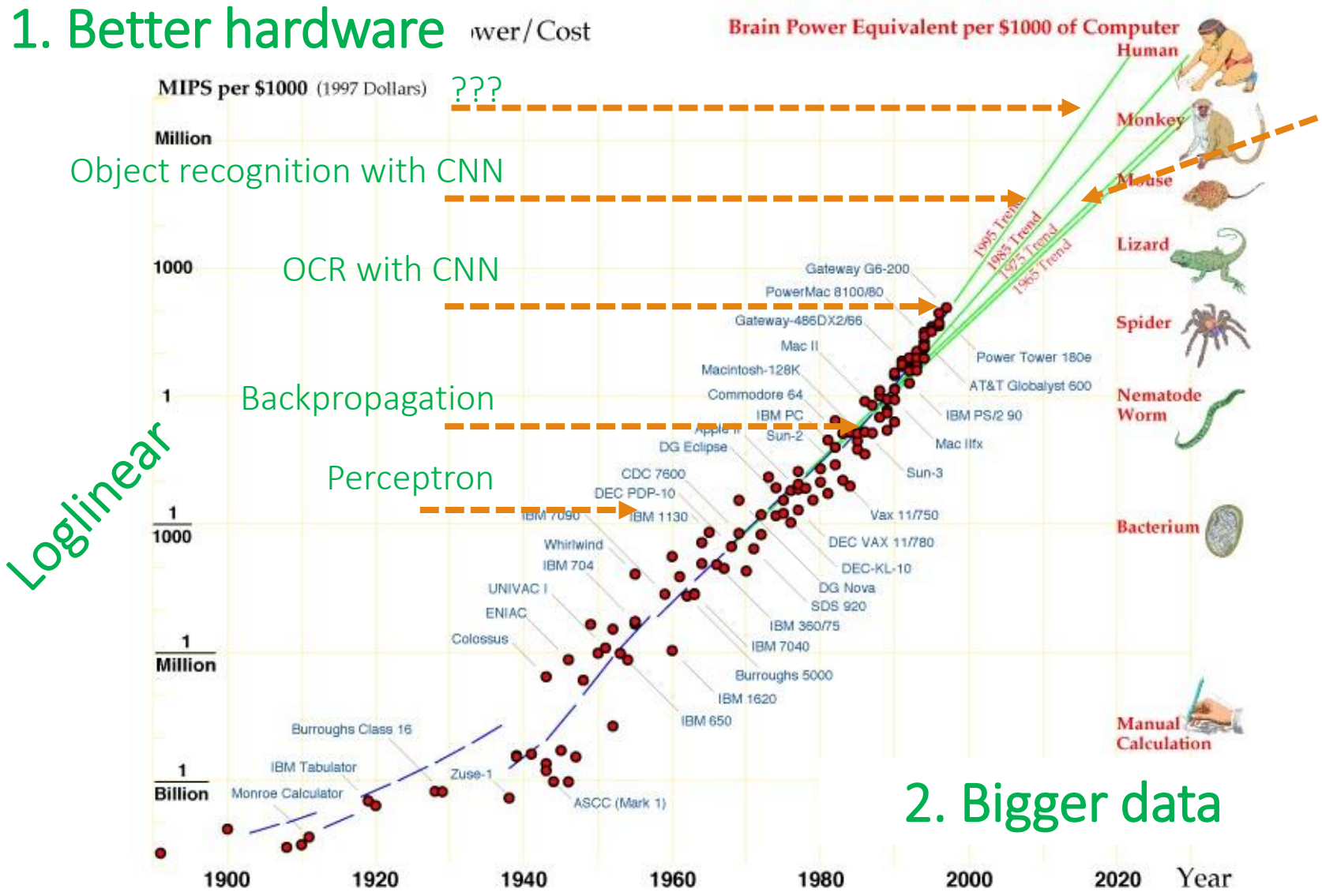
So, why now?

1. Better hardware



So, why now?

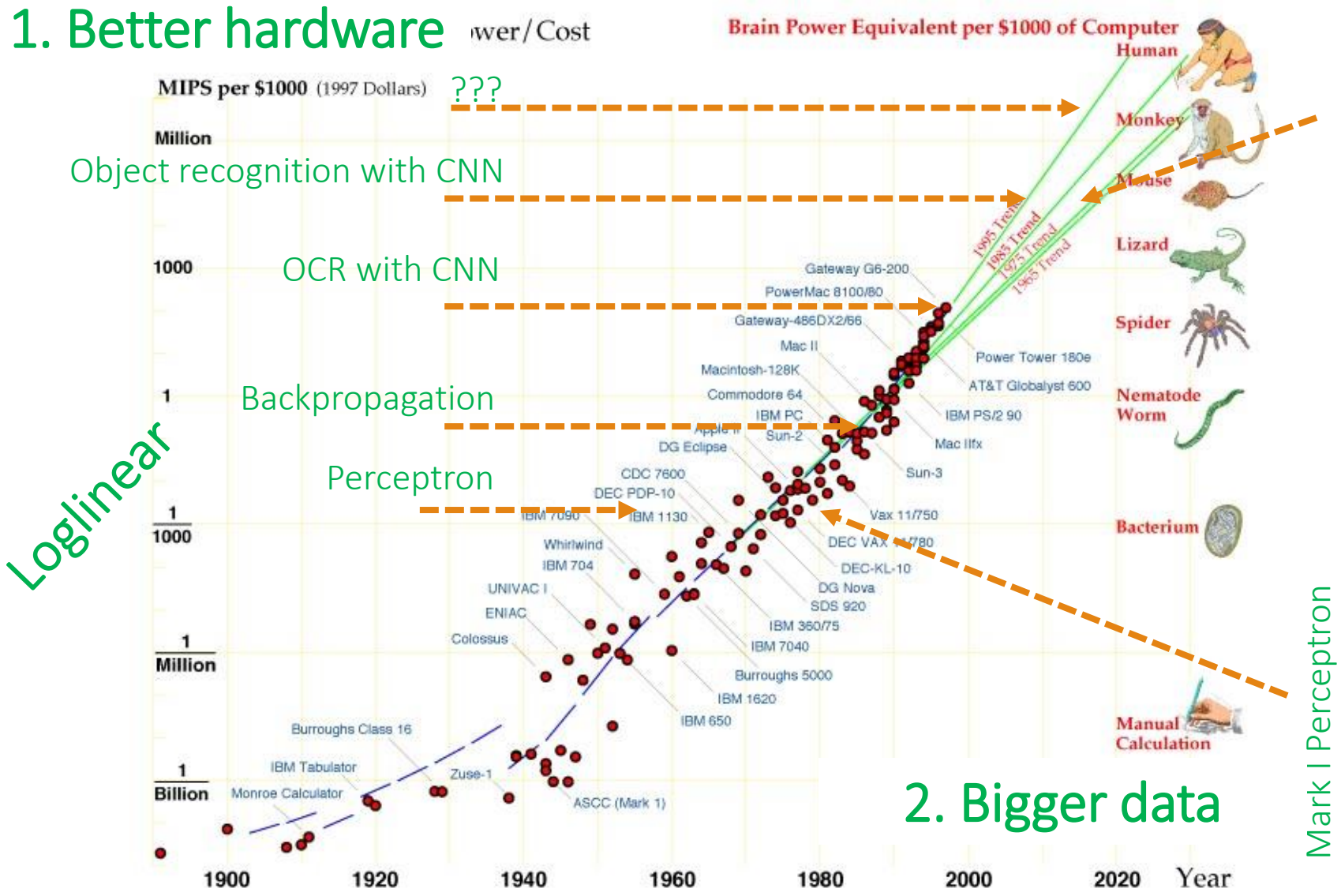
1. Better hardware



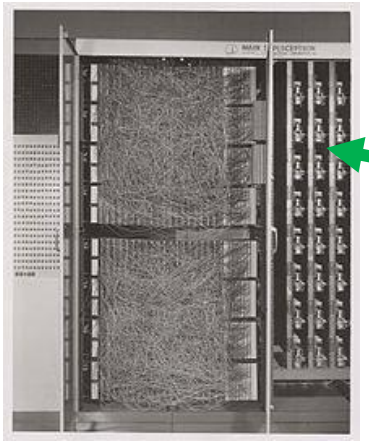
2. Bigger data

So, why now?

1. Better hardware



2. Bigger data

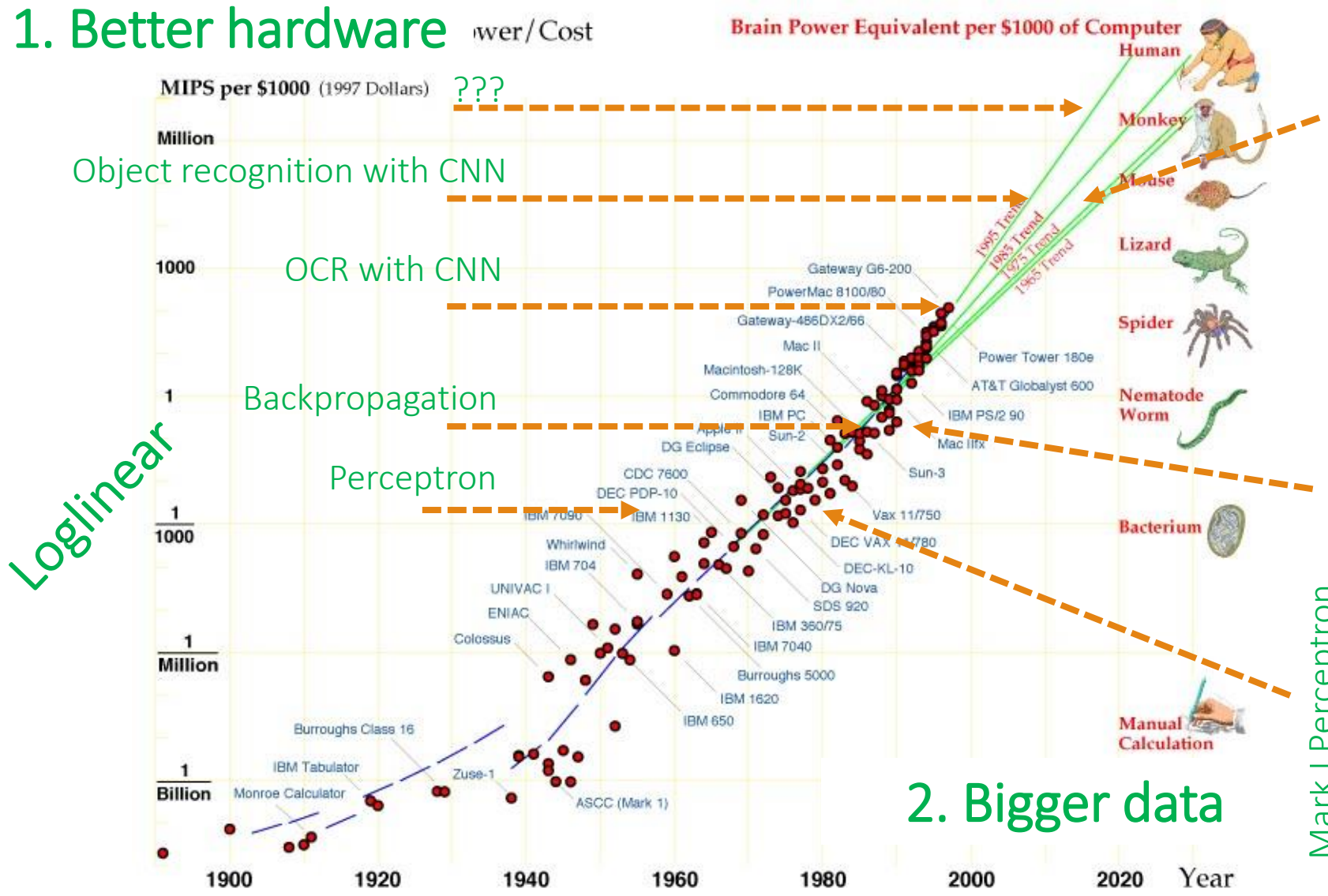


Mark I Perceptron

Potentiometers implement perceptron weights

So, why now?

1. Better hardware



???

Object recognition with CNN

OCR with CNN

Backpropagation

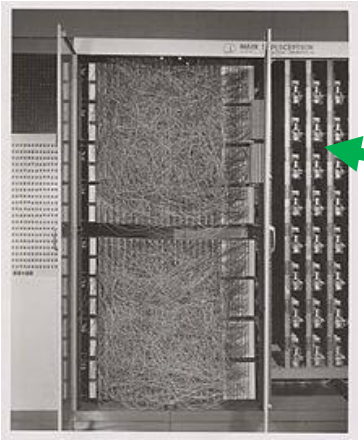
Perceptron

Loglinear

Parity, negation problems

D1	D2	D3	Even-Parity
0	0	0	True
0	0	1	False
0	1	0	False
0	1	1	True
1	0	0	False
1	0	1	True
1	1	0	True
1	1	1	False

Mark I Perceptron

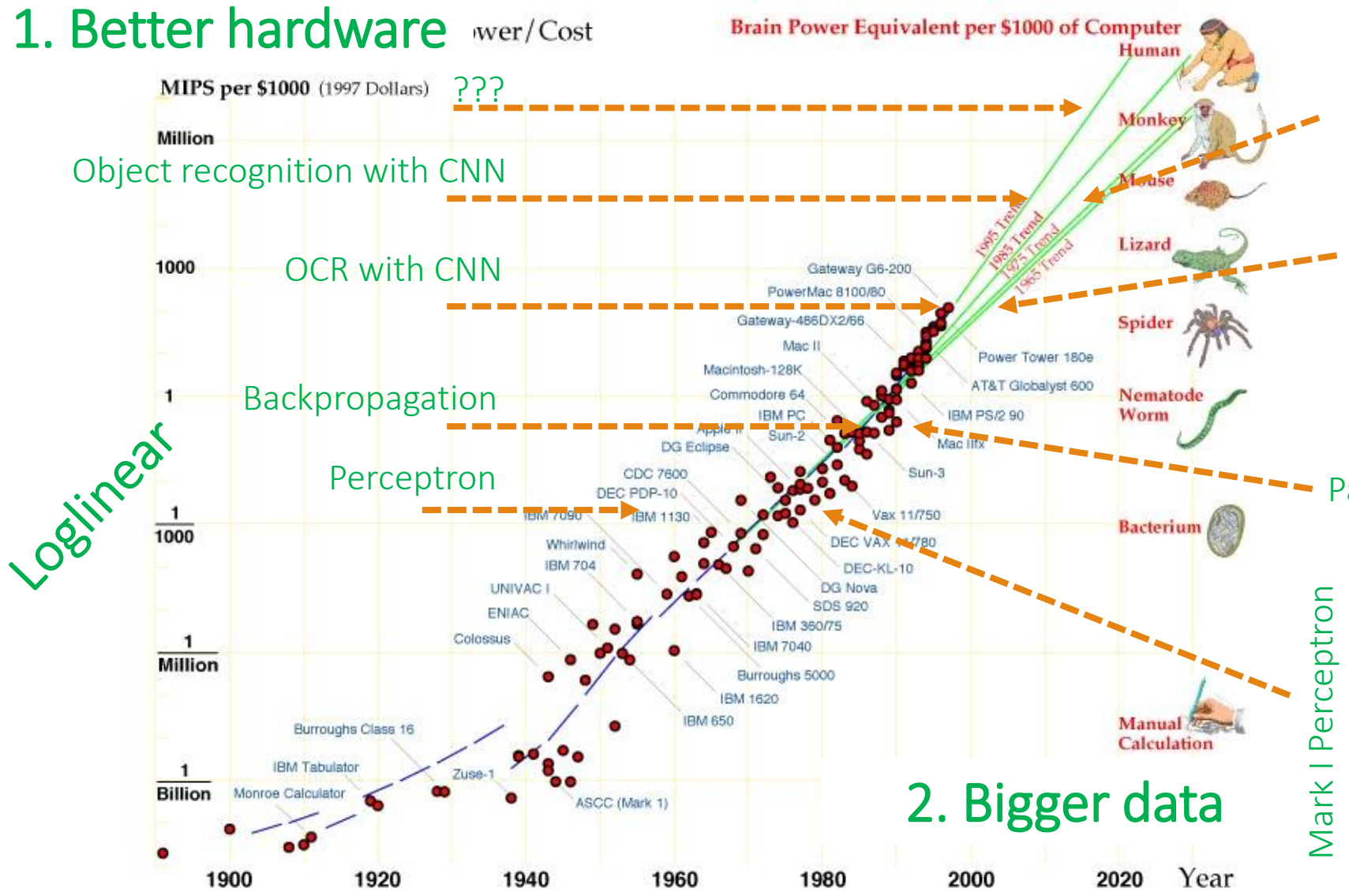


Potentiometers implement perceptron weights

2. Bigger data

So, why now?

1. Better hardware



??? Object recognition with CNN

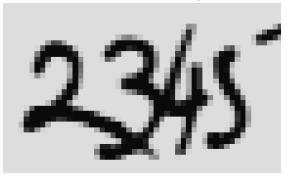
OCR with CNN

Backpropagation

Perceptron

Loglinear

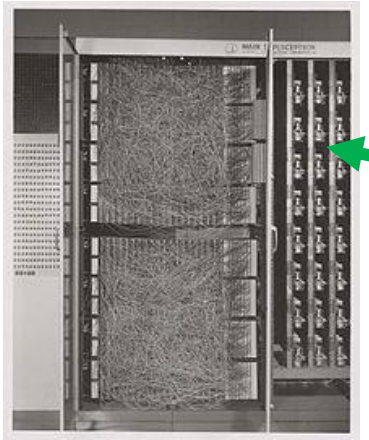
Bank cheques



Parity, negation problems

D1	D2	D3	Even-Parity
0	0	0	True
0	0	1	False
0	1	0	False
0	1	1	True
1	0	0	False
1	0	1	True
1	1	0	True
1	1	1	False

Mark I Perceptron

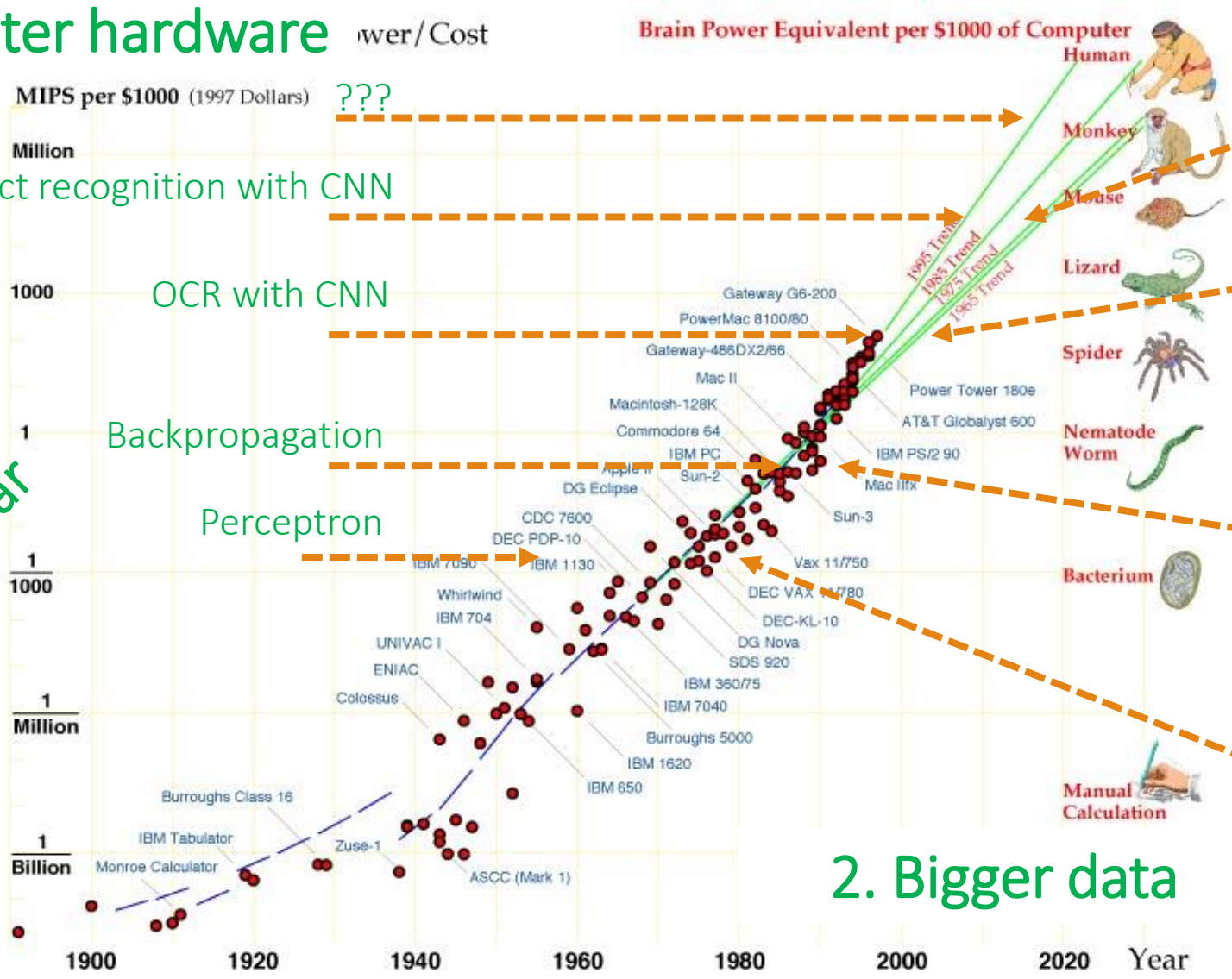


Potentiometers implement perceptron weights

2. Bigger data

So, why now?

1. Better hardware



Loglinear

??? Object recognition with CNN

OCR with CNN

Backpropagation

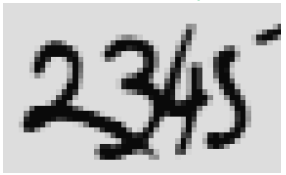
Perceptron

Imagenet: 1,000 classes from real images, 1,000,000 images

Results:

- Persian cat: 0.35211
- Egyptian cat: 0.23635
- hamster: 0.20282
- tiger cat: 0.05896
- lynx: 0.05759

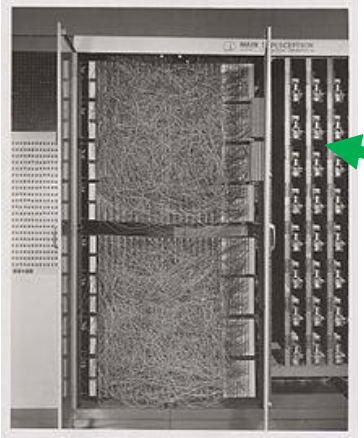
Bank cheques



Parity, negation problems

D1	D2	D3	Even-Parity
0	0	0	True
0	0	1	False
0	1	0	False
0	1	1	True
1	0	0	False
1	0	1	True
1	1	0	True
1	1	1	False

Mark I Perceptron

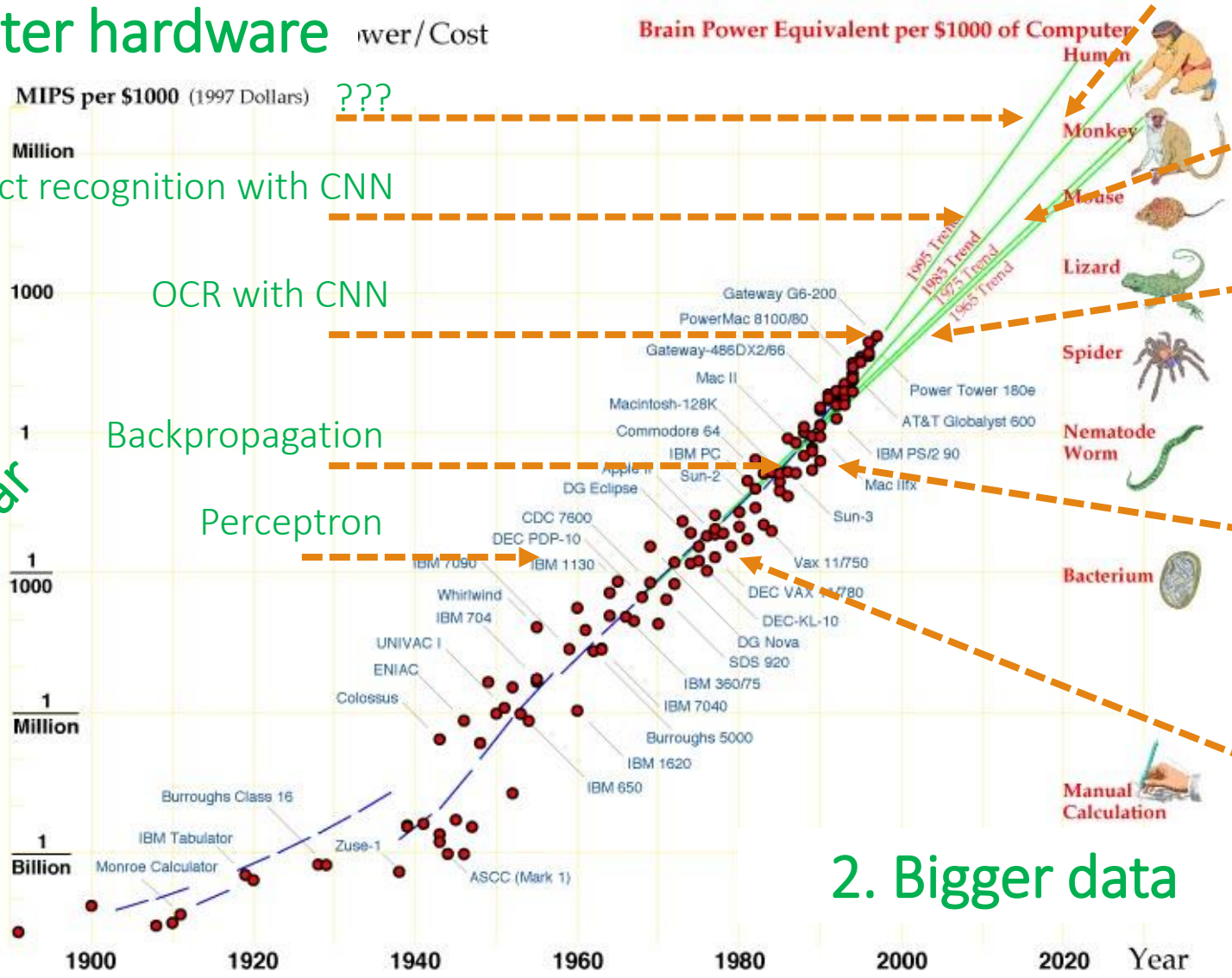


Potentiometers implement perceptron weights

2. Bigger data

So, why now?

1. Better hardware



Loglinear

???

Object recognition with CNN

OCR with CNN

Backpropagation

Perceptron

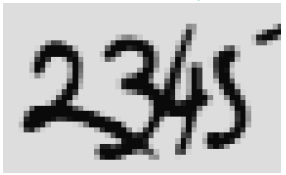
Datasets of everything (captions, question-answering, …), reinforced learning, ???

Imagenet: 1,000 classes from real images, 1,000,000 images

Results:

- Persian cat: 0.35211
- Egyptian cat: 0.23635
- hamster: 0.20282
- tiger cat: 0.05896
- lynx: 0.05759

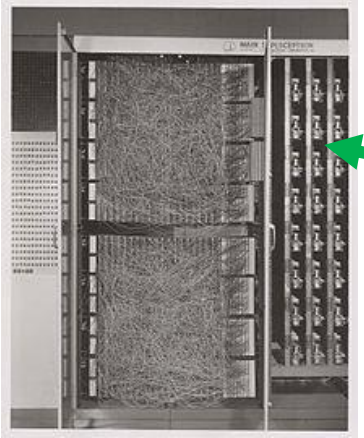
Bank cheques



Parity, negation problems

D1	D2	D3	Even-Parity
0	0	0	True
0	0	1	False
0	1	0	False
0	1	1	True
1	0	0	False
1	0	1	True
1	1	0	True
1	1	1	False

Mark I Perceptron



Potentiometers implement perceptron weights

2. Bigger data

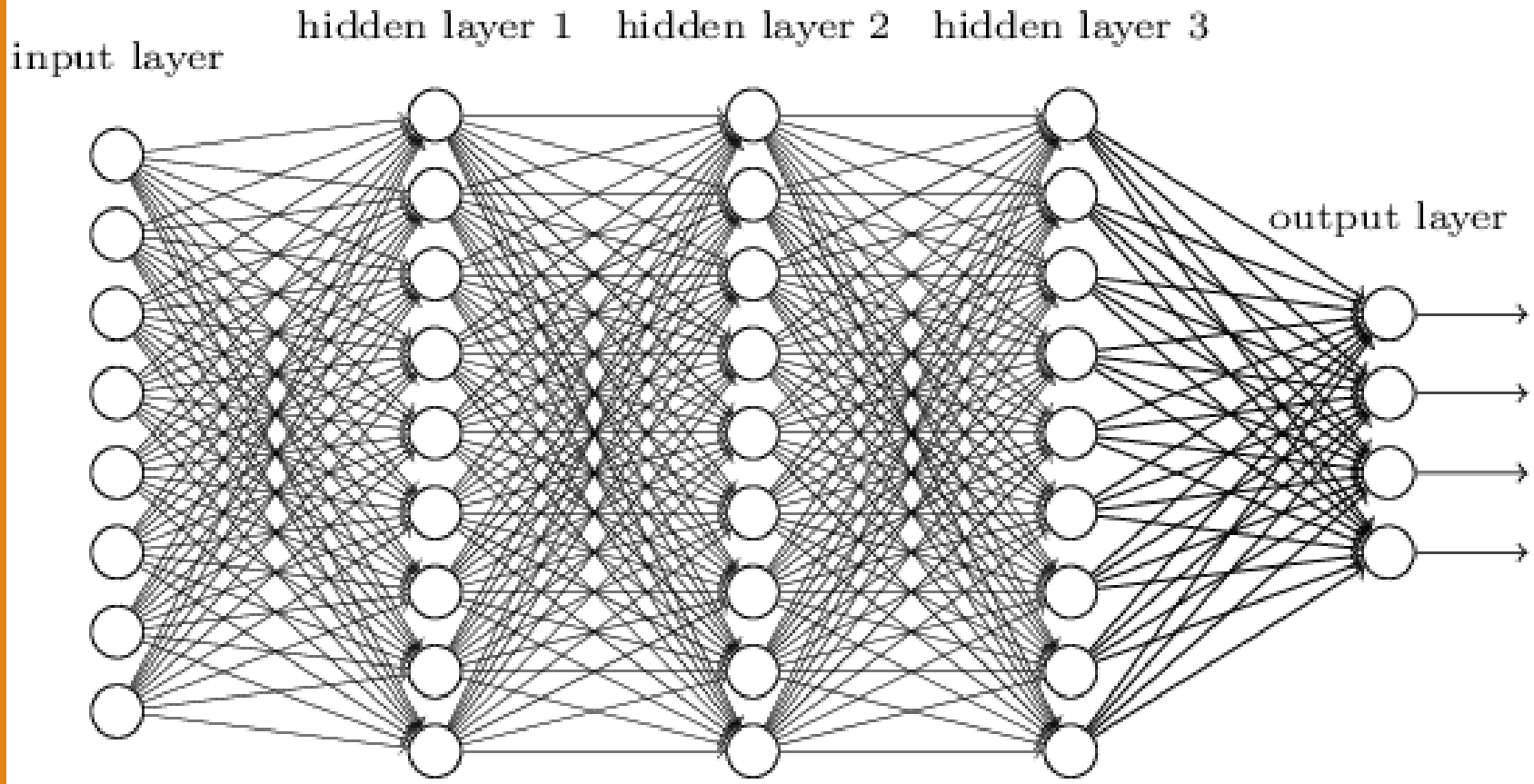
So, why now? (2)

1. Better hardware
2. Bigger data

So, why now? (2)

1. Better hardware
2. Bigger data
3. Better regularization methods, such as dropout
4. Better optimization protocols, such as batch normalization

Deep Learning: The *What* and *Why*



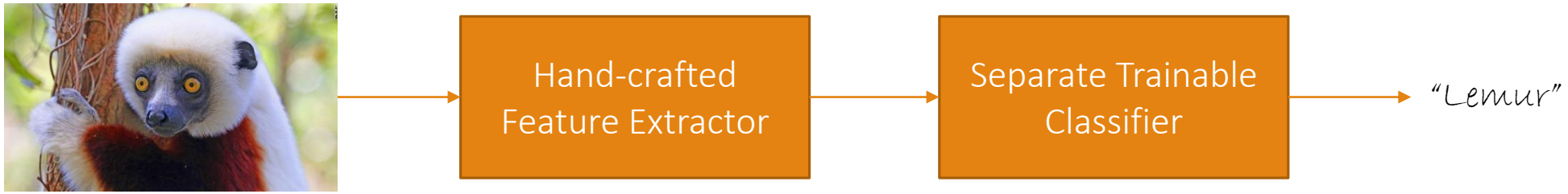
Long story short

- A family of parametric, non-linear and hierarchical representation learning functions, which are massively optimized with stochastic gradient descent to encode domain knowledge, i.e. domain invariances, stationarity.
- $a_L\left(x; \theta_{1,\ldots,\mathrm{L}}\right) = h_L\left(h_{L-1}(\ldots h_1(x, \theta_1), \theta_{L-1}),\ \theta_L\right)$
  - x:input, θ_l: parameters for layer l, $a_l = h_l(x, \theta_l)$: (non-)linear function
- Given training corpus $\{X, Y\}$ find optimal parameters

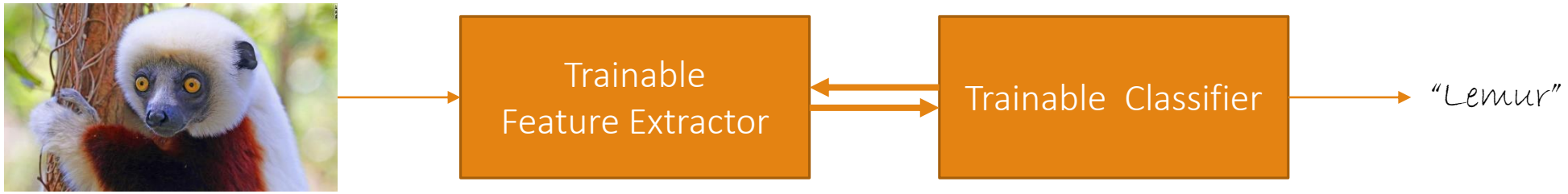
$$\theta^* \leftarrow \arg\min_\theta \sum_{(x,y)\subseteq(X,Y)} \ell\left(y, a_L\left(x; \theta_{1,\ldots,\mathrm{L}}\right)\right)$$

Learning Representations & Features

- Traditional pattern recognition

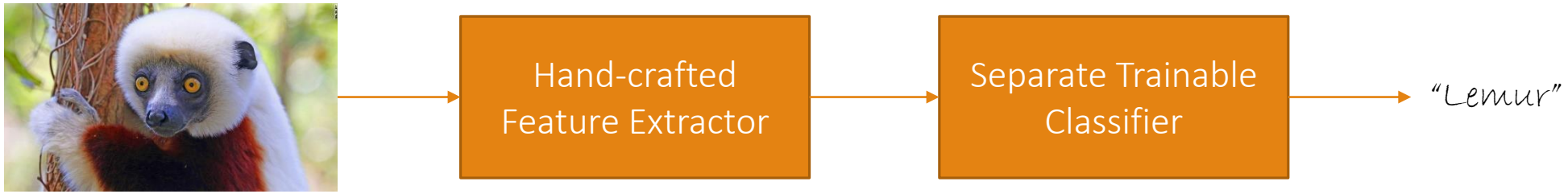


- End-to-end learning → Features are also learned from data

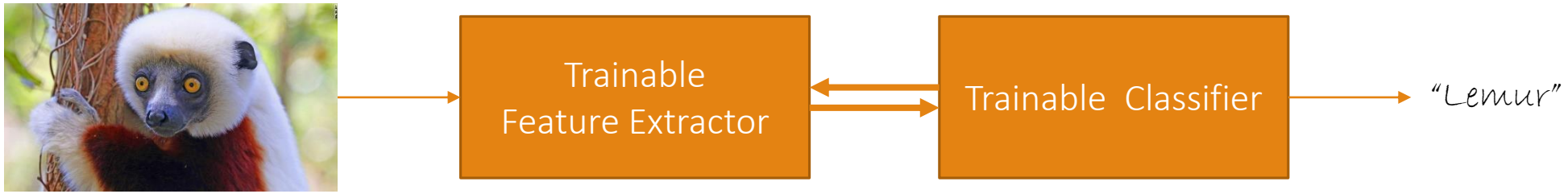


Learning Representations & Features

- Traditional pattern recognition



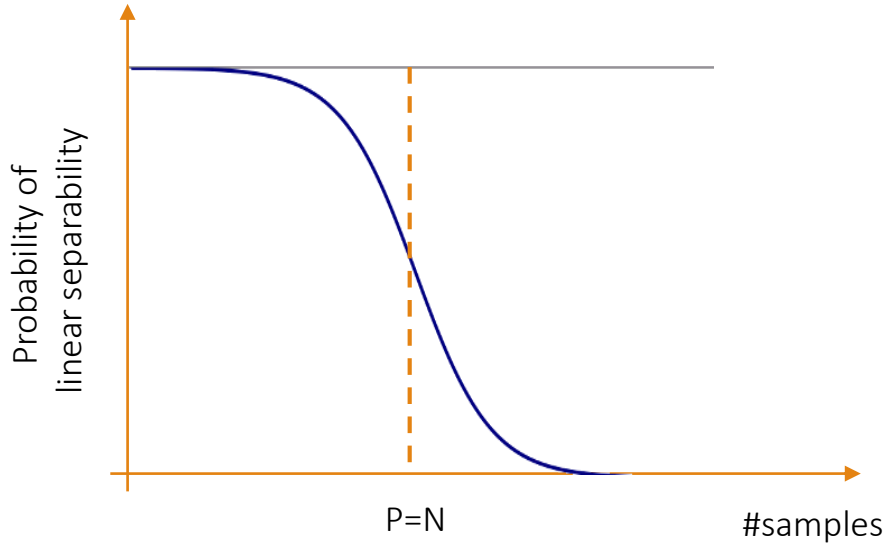
- End-to-end learning → Features are also learned from data



Non-separability of linear machines

$$X = \{x_1, x_2, \dots, x_n\} \in \mathcal{R}^d$$

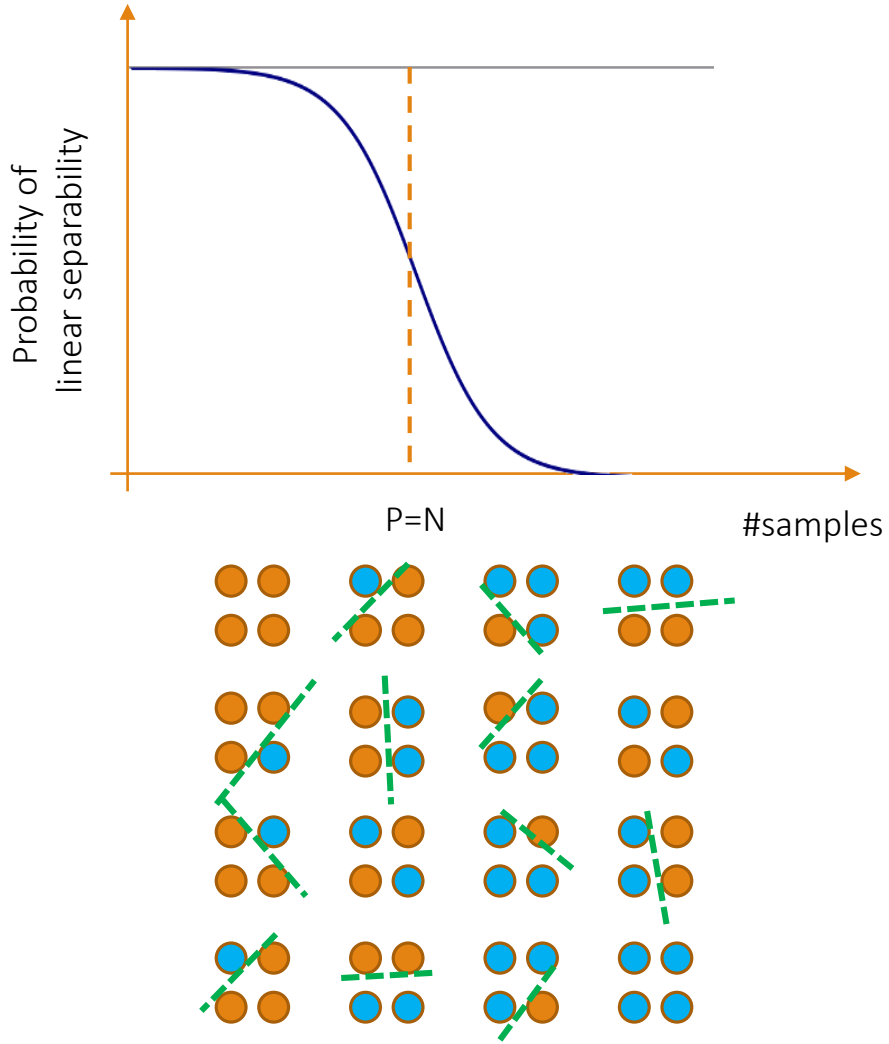
- Given the n points there are in total 2^n dichotomies
- are linearly separable
- With $n > d$ the probability X is linearly separable converges to 0 very fast
- The chances that a dichotomy is linearly separable is very small



Non-separability of linear machines

$$X = \{x_1, x_2, \dots, x_n\} \in \mathcal{R}^d$$

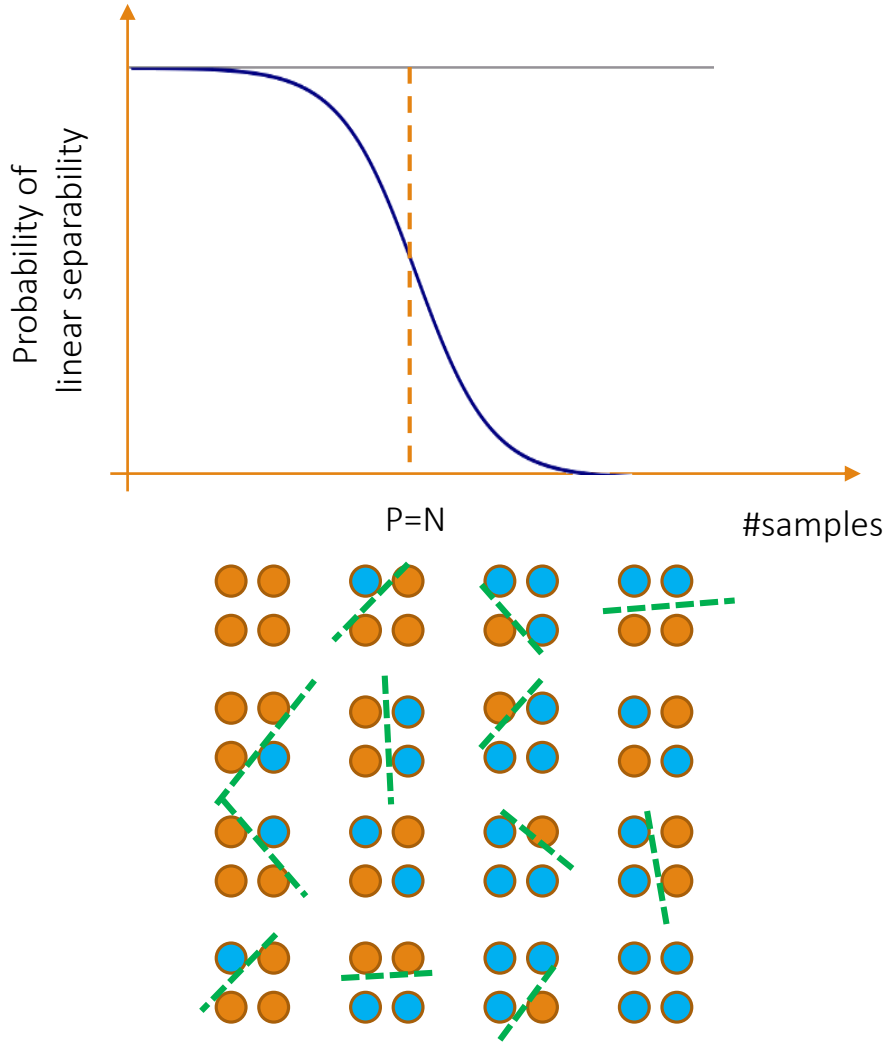
- Given the n points there are in total 2^n dichotomies
- Only about d are linearly separable
- With $n > d$ the probability X is linearly separable converges to 0 very fast
- The chances that a dichotomy is linearly separable is very small



Non-separability of linear machines

$tsaf\ yrev\ 0\ ot\ segrevnoc\ elbarapes\ ylraenil\ siXX\ \ ytilibaborp\ eht\ ddX = \{x_1, x_2, \dots, x_n\} \in \mathcal{R}^d$

- Given the n points there are in total 2^n dichotomies
- Only about d are linearly separable
- The chances that a dichotomy is linearly separable is very small
- The chances that a dichotomy is linearly separable is very small



Non-linearizing linear machines

- Most data distributions and tasks are non-linear
- A linear assumption is often convenient, but not necessarily truthful
- **Problem:** How to get non-linear machines without too much effort?

Non-linearizing linear machines

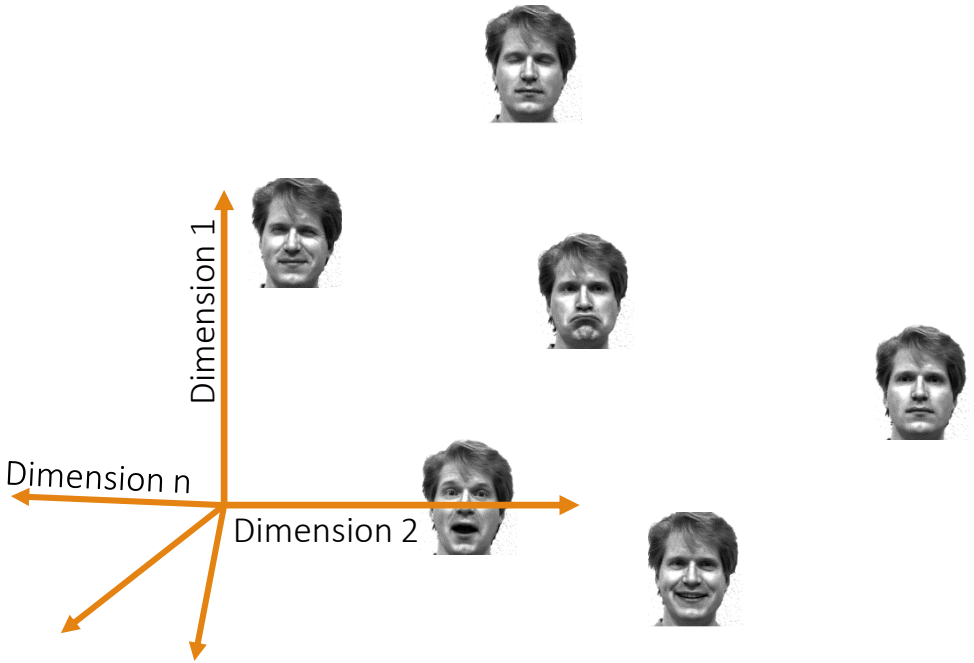
- Most data distributions and tasks are non-linear
- A linear assumption is often convenient, but not necessarily truthful
- **Problem:** How to get non-linear machines without too much effort?
- **Solution:** Make features non-linear
- What is a good non-linear feature?
  - Polynomials and cross products of the input
  - Explicit design of features (SIFT, HOG)?

Good features

- High-dimensional data lie in lower dimensional manifolds
  - **Goal:** discover these lower dimensional manifolds
  - These manifolds are most probably highly non-linear
- **Hypothesis (1):** By computing the coordinates of the input (e.g. face image) to this non-linear manifold data become separable
- **Hypothesis (2):** Semantically similar things lie closer together than semantically dissimilar things

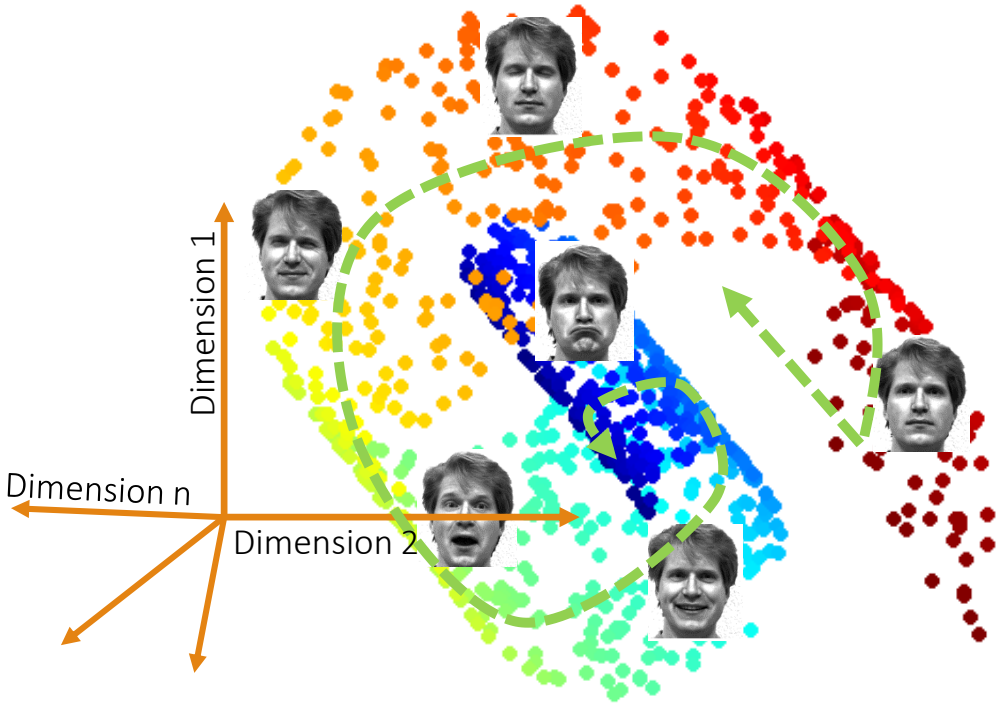
Feature manifold example

- Raw data live in huge dimensionalities



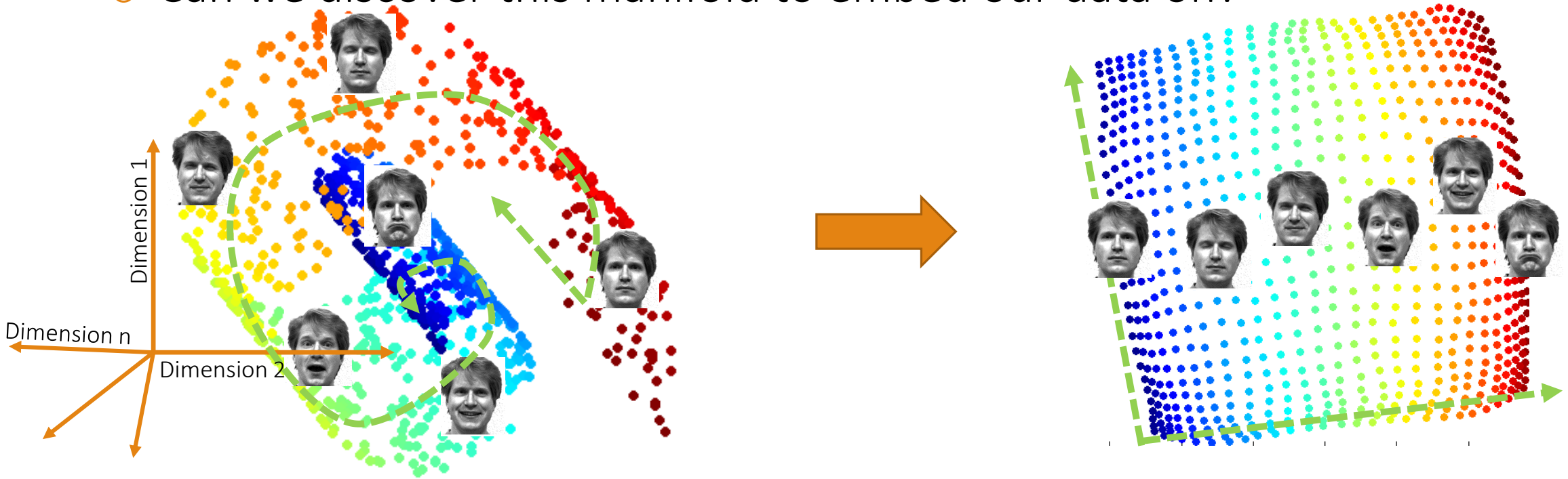
Feature manifold example

- Raw data live in huge dimensionalities
- Semantically meaningful raw data prefer lower dimensional manifolds
  - Which still live in the same huge dimensionalities



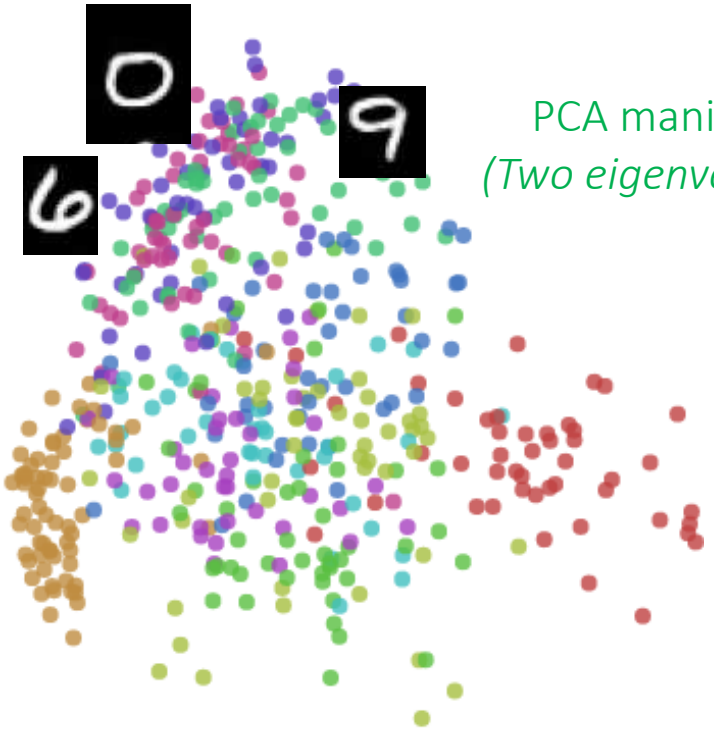
Feature manifold example

- Raw data live in huge dimensionalities
- Semantically meaningful raw data prefer lower dimensional manifolds
  - Which still live in the same huge dimensionalities
- Can we discover this manifold to embed our data on?



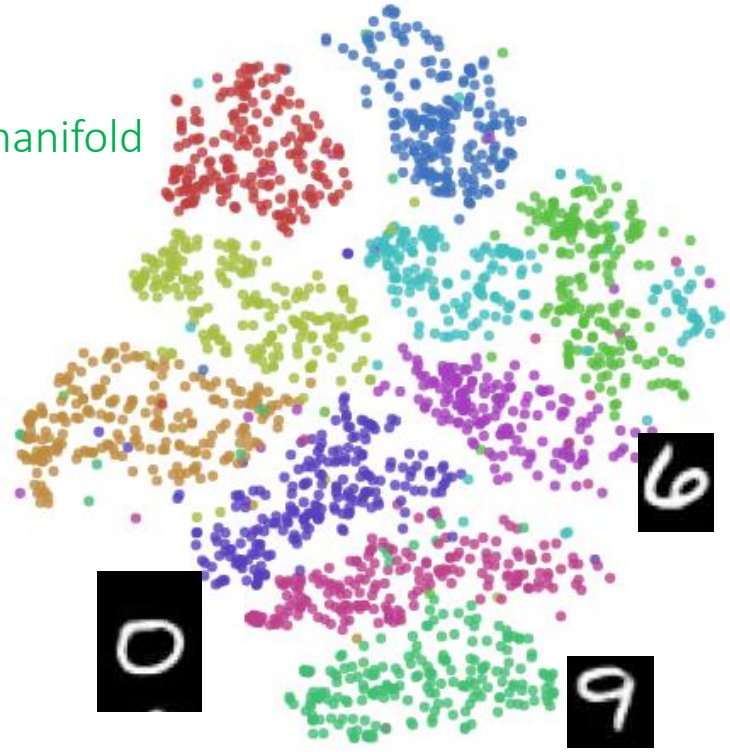
The digits manifolds

- There are good features and bad features, good manifold representations and bad manifold representations
- 28 pixels x 28 pixels = 784 dimensions



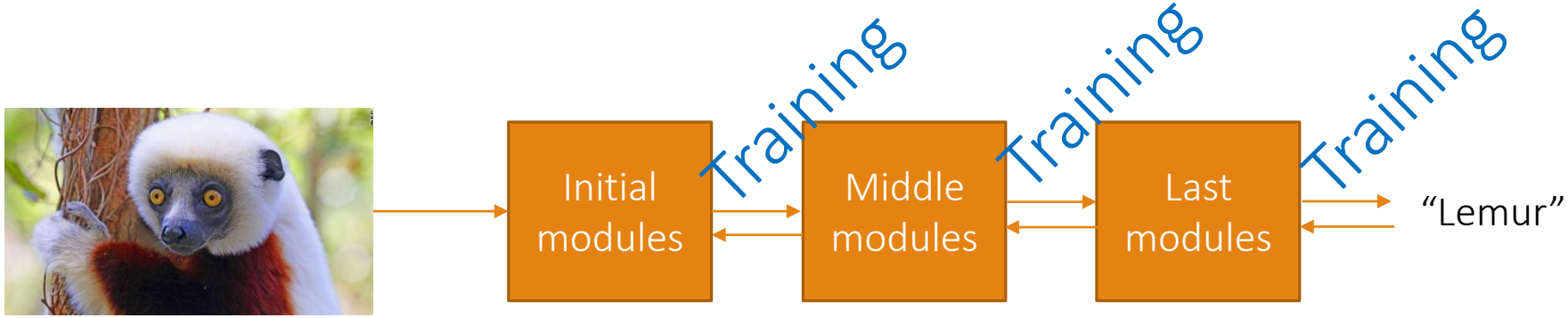
PCA manifold
(Two eigenvectors)

t-SNE manifold



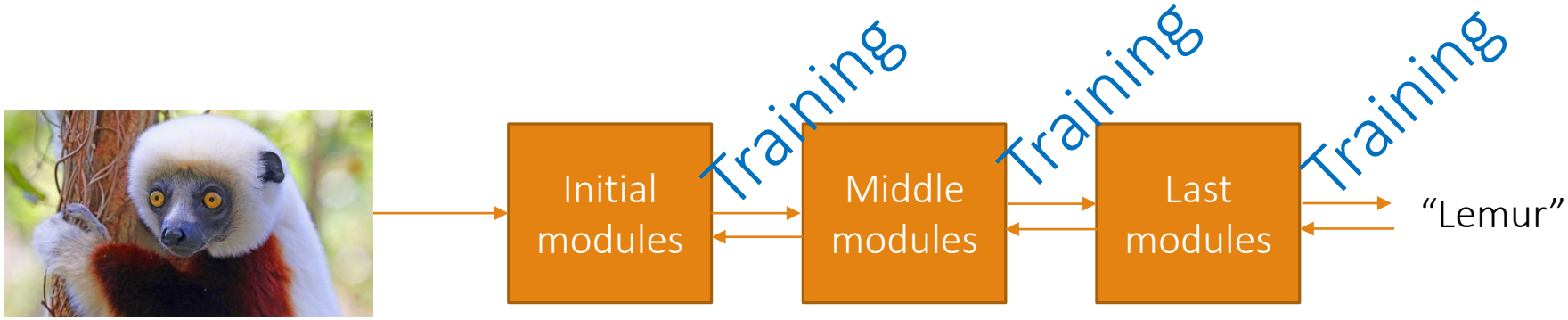
End-to-end learning of feature hierarchies

- A pipeline of successive modules



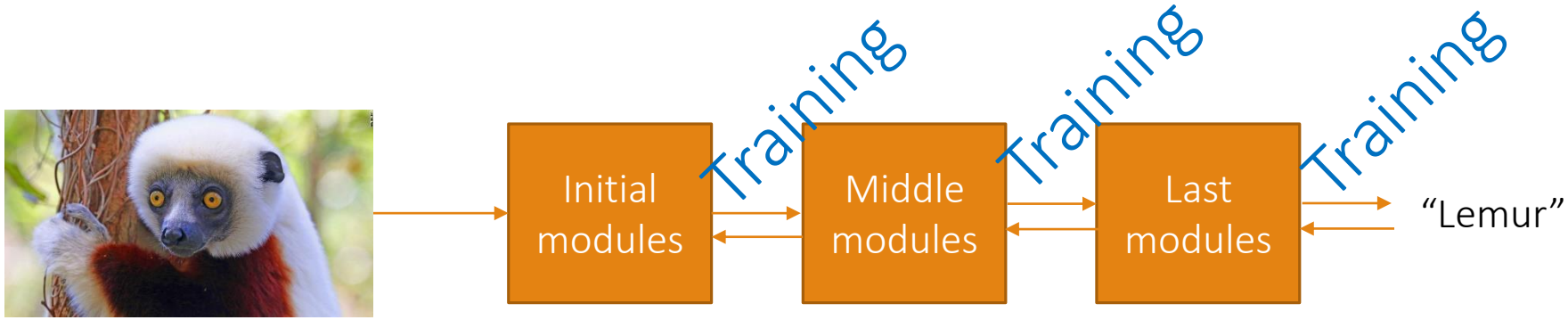
End-to-end learning of feature hierarchies

- A pipeline of successive modules
- Each module's output is the input for the next module



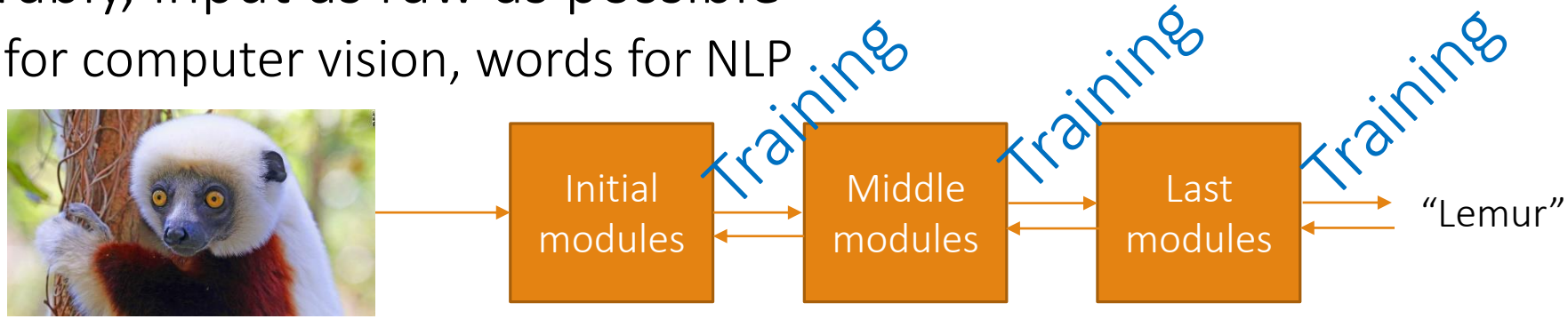
End-to-end learning of feature hierarchies

- A pipeline of successive modules
- Each module's output is the input for the next module
- Modules produce features of higher and higher abstractions
  - Initial modules capture low-level features (e.g. edges or corners)
  - Middle modules capture mid-level features (e.g. circles, squares, textures)
  - Last modules capture high level, class specific features (e.g. face detector)



End-to-end learning of feature hierarchies

- A pipeline of successive modules
- Each module's output is the input for the next module
- Modules produce features of higher and higher abstractions
  - Initial modules capture low-level features (e.g. edges or corners)
  - Middle modules capture mid-level features (e.g. circles, squares, textures)
  - Last modules capture high level, class specific features (e.g. face detector)
- Preferably, input as raw as possible
  - Pixels for computer vision, words for NLP



Why learn the features?

- Manually designed features
  - Often take a lot of time to come up with and implement
  - Often take a lot of time to validate
  - Often they are incomplete, as one cannot know if they are optimal for the task

vs.

Why learn the features?

- Manually designed features
  - Often take a lot of time to come up with and implement
  - Often take a lot of time to validate
  - Often they are incomplete, as one cannot know if they are optimal for the task
- Learned features
  - Are easy to adapt
  - Very compact and specific to the task at hand
  - Given a basic architecture in mind, it is relatively easy and fast to optimize

vs.

Why learn the features?

- Manually designed features
  - Often take a lot of time to come up with and implement
  - Often take a lot of time to validate
  - Often they are incomplete, as one cannot know if they are optimal for the task
- Learned features
  - Are easy to adapt
  - Very compact and specific to the task at hand
  - Given a basic architecture in mind, it is relatively easy and fast to optimize
- Time spent for <u>designing features</u> now spent for <u>designing architectures</u>

vs.

Types of learning

- Supervised learning
  - (Convolutional) neural networks

Is this a dog or a cat?

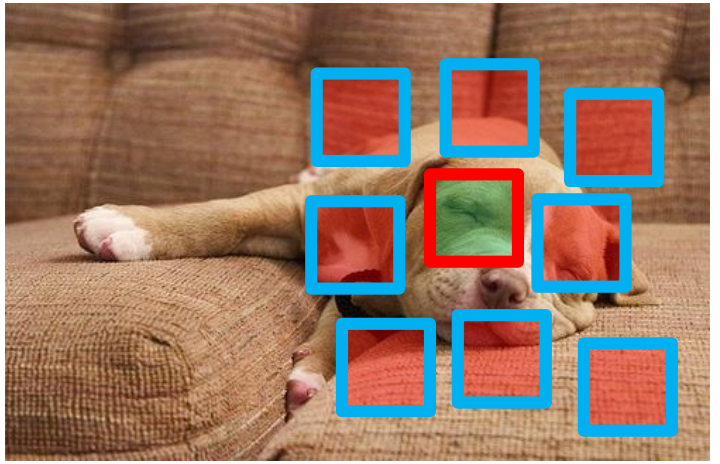
Types of learning

- Supervised learning
  - (Convolutional) neural networks
- Unsupervised learning
  - Autoencoders, layer-by-layer training

Reconstruct this image

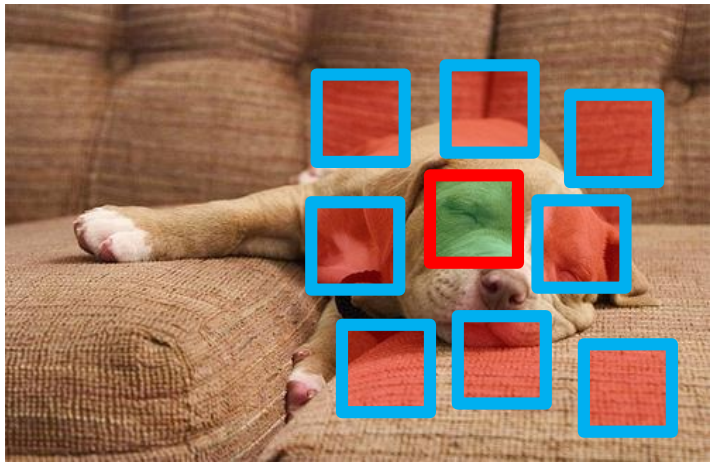
Types of learning

- Supervised learning
  - (Convolutional) neural networks
- Unsupervised learning
  - Autoencoders, layer-by-layer training
- Self-supervised learning
  - A mix of supervised and unsupervised learning



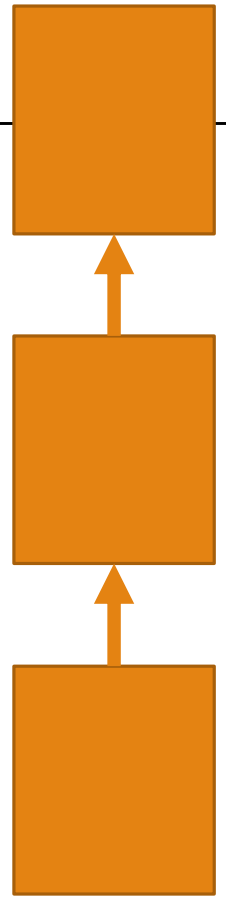
Types of learning

- Supervised learning
  - (Convolutional) neural networks
- Unsupervised learning
  - Autoencoders, layer-by-layer training
- Self-supervised learning
  - A mix of supervised and unsupervised learning
- Reinforcement learning
  - Learn from noisy, delayed rewards from your environment
  - Perform actions in your environment, so as to make decisions what data to collect



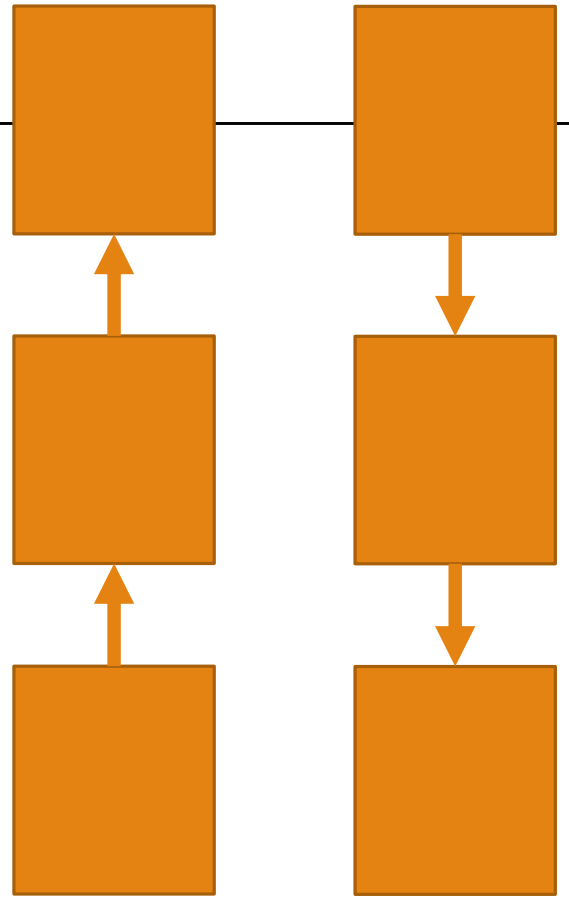
Deep architectures

- Feedforward
  - (Convolutional) neural networks



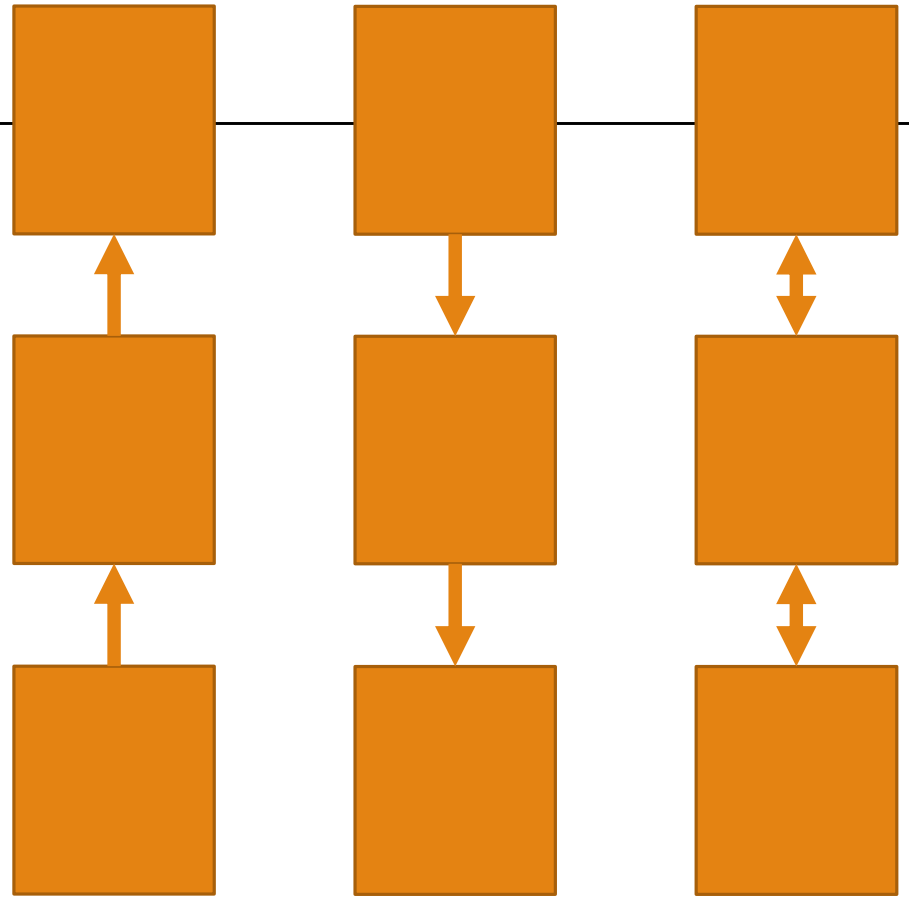
Deep architectures

- Feedforward
  - (Convolutional) neural networks
- Feedback
  - Deconvolutional networks



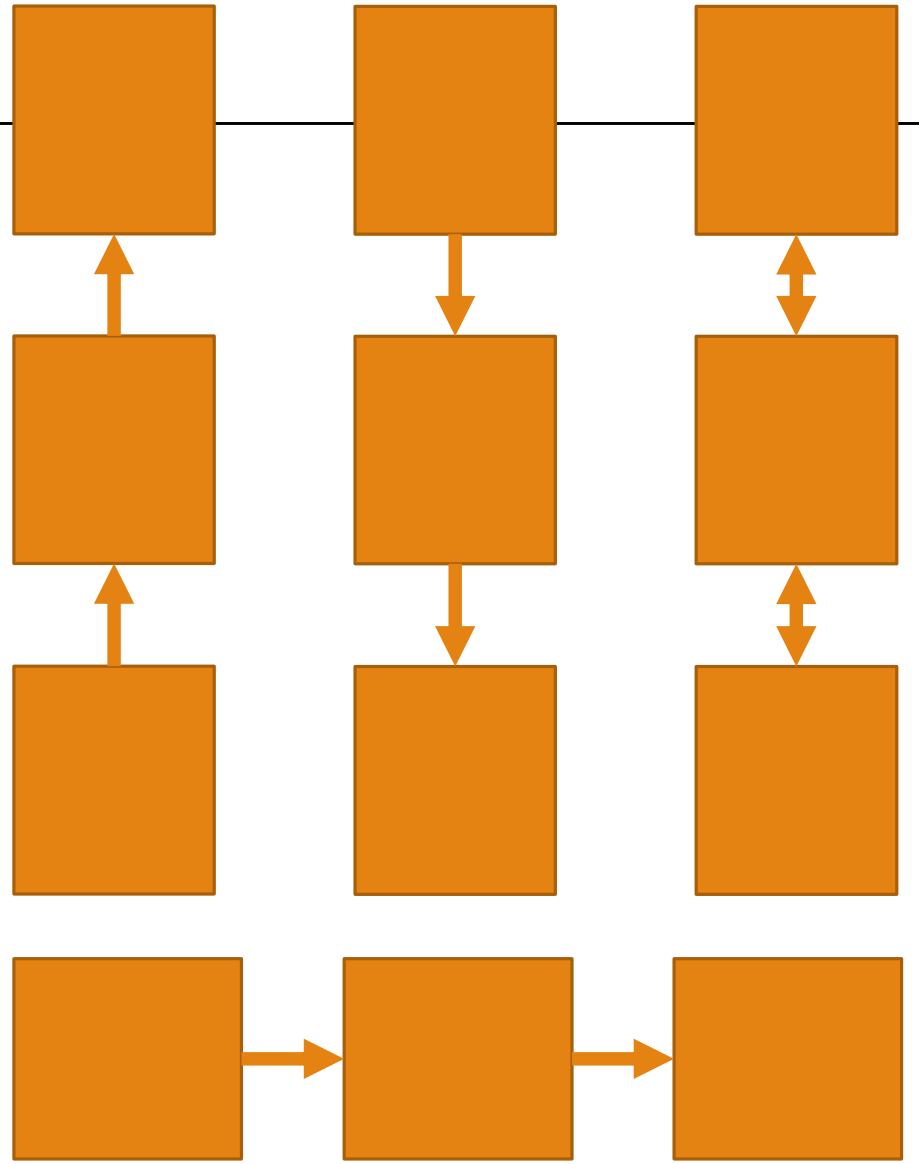
Deep architectures

- Feedforward
  - (Convolutional) neural networks
- Feedback
  - Deconvolutional networks
- Bi-directional
  - Deep Boltzmann Machines, stacked autoencoders



Deep architectures

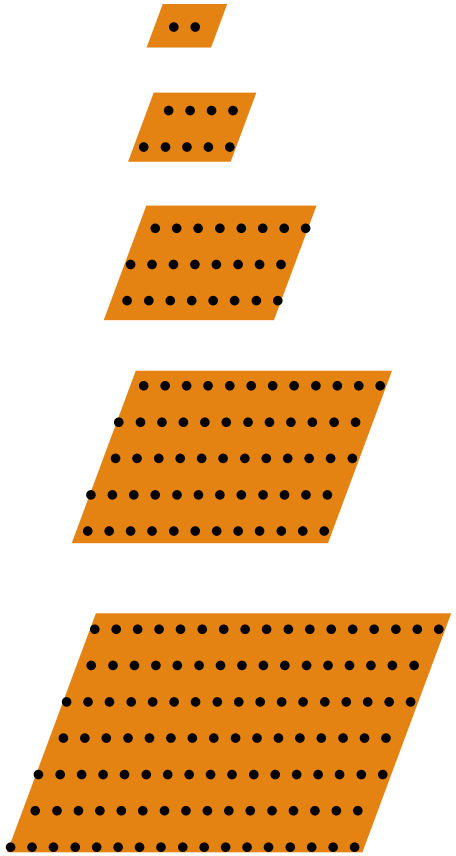
- Feedforward
  - (Convolutional) neural networks
- Feedback
  - Deconvolutional networks
- Bi-directional
  - Deep Boltzmann Machines, stacked autoencoders
- Sequence based
  - RNNs, LSTMs



Convolutional networks in a nutshell

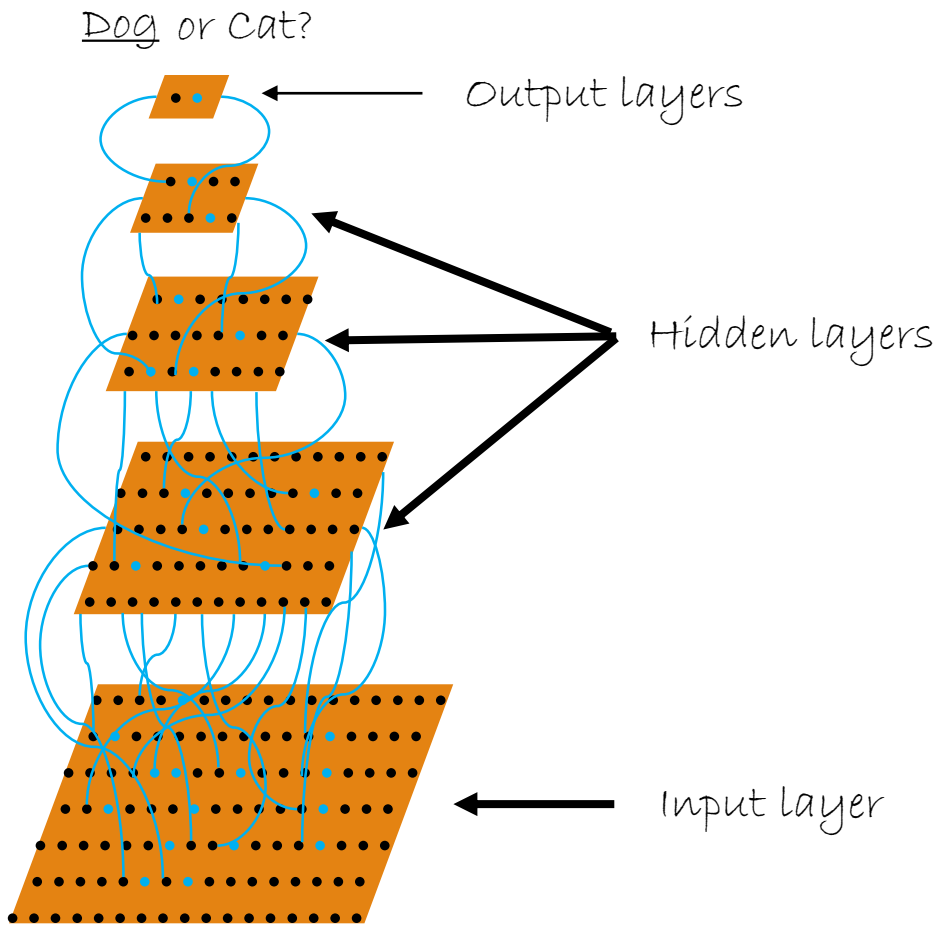
Is this a dog or a cat?

Dog or Cat?

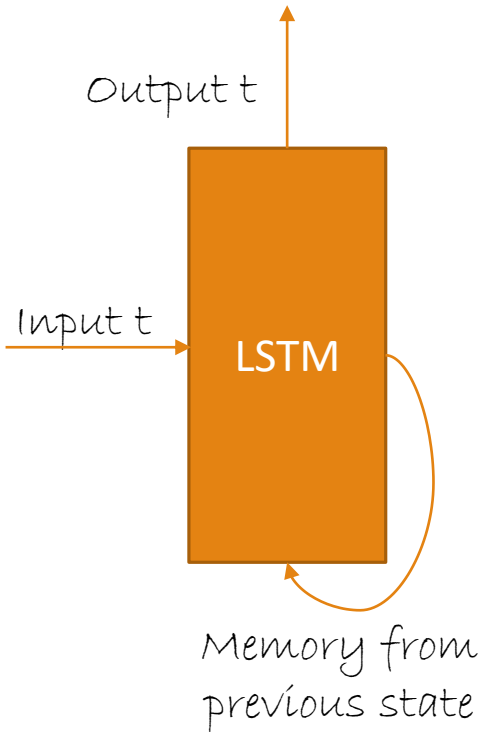


Convolutional networks in a nutshell

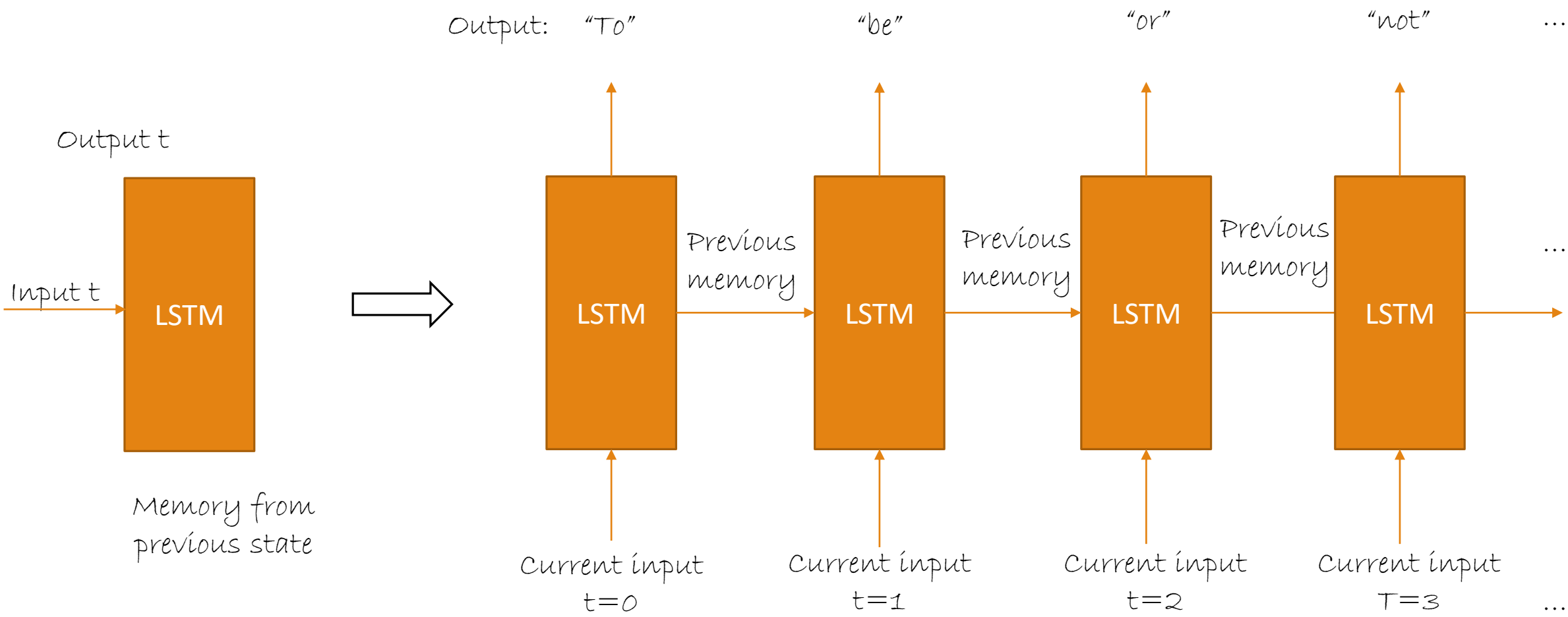
Is this a dog or a cat?



Recurrent networks in a nutshell

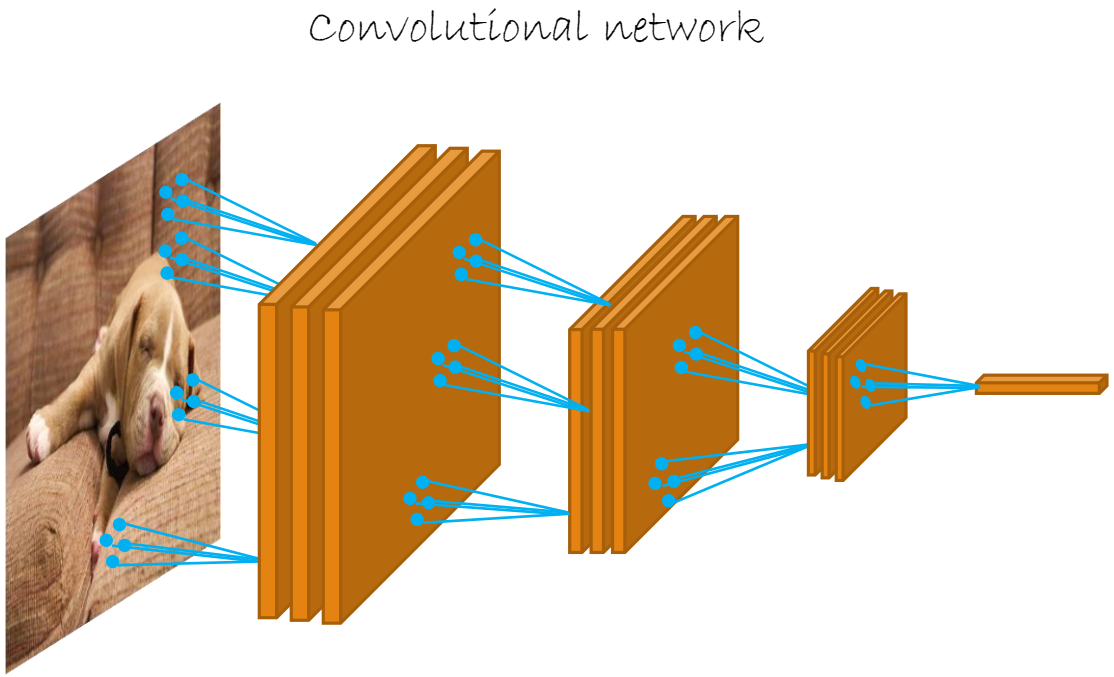


Recurrent networks in a nutshell

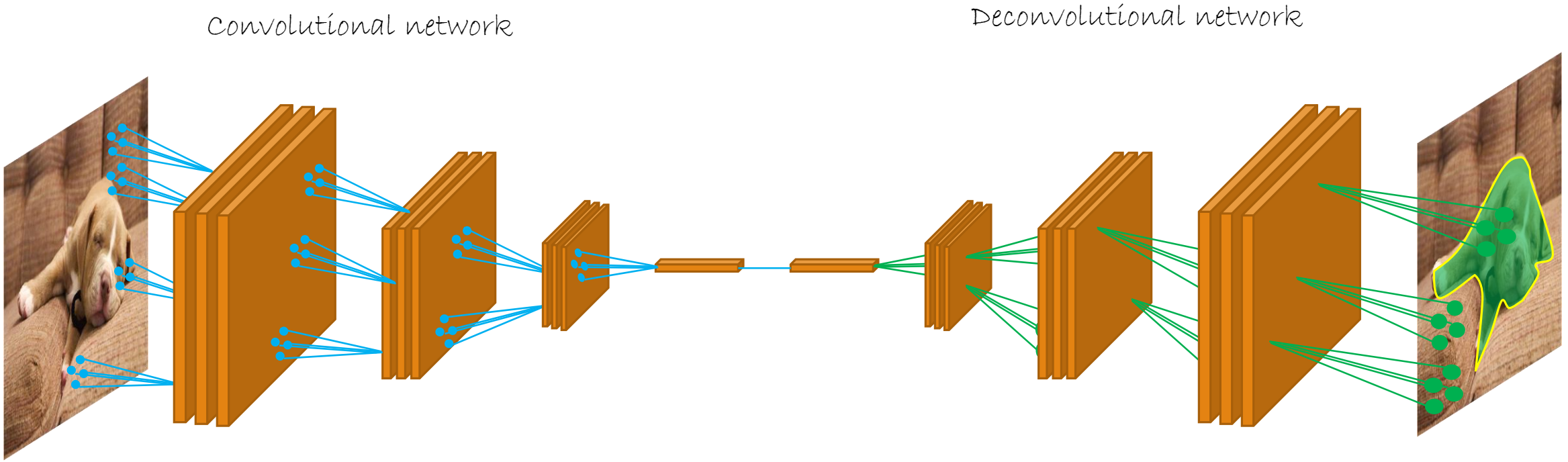


Deconvolutional networks

Deconvolutional networks

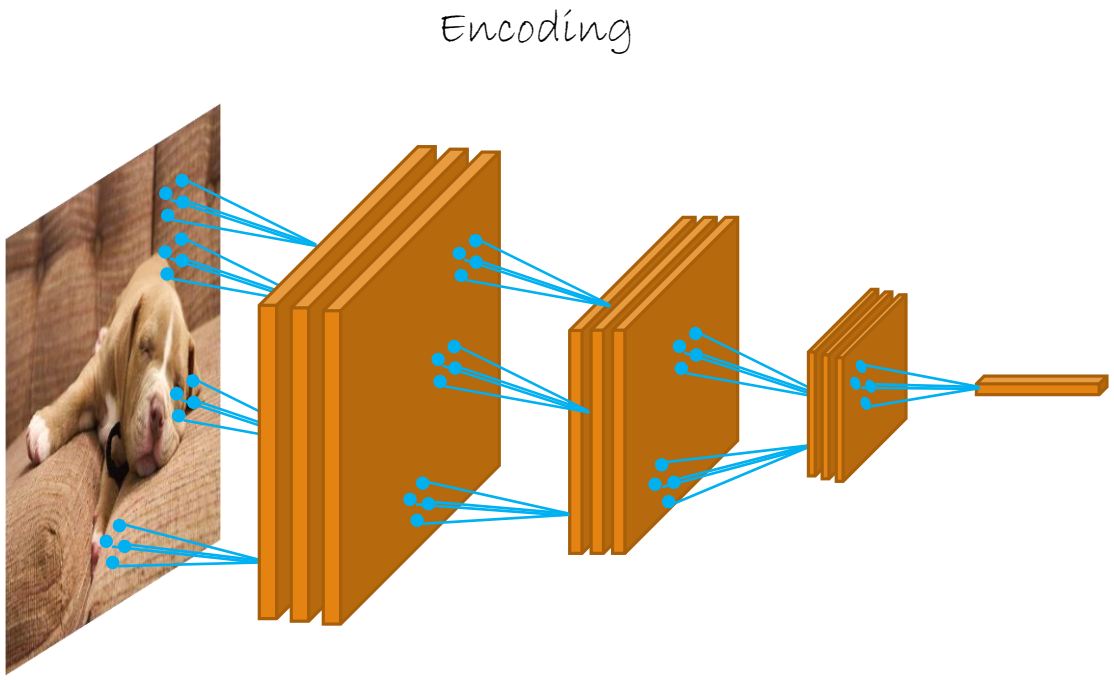


Deconvolutional networks

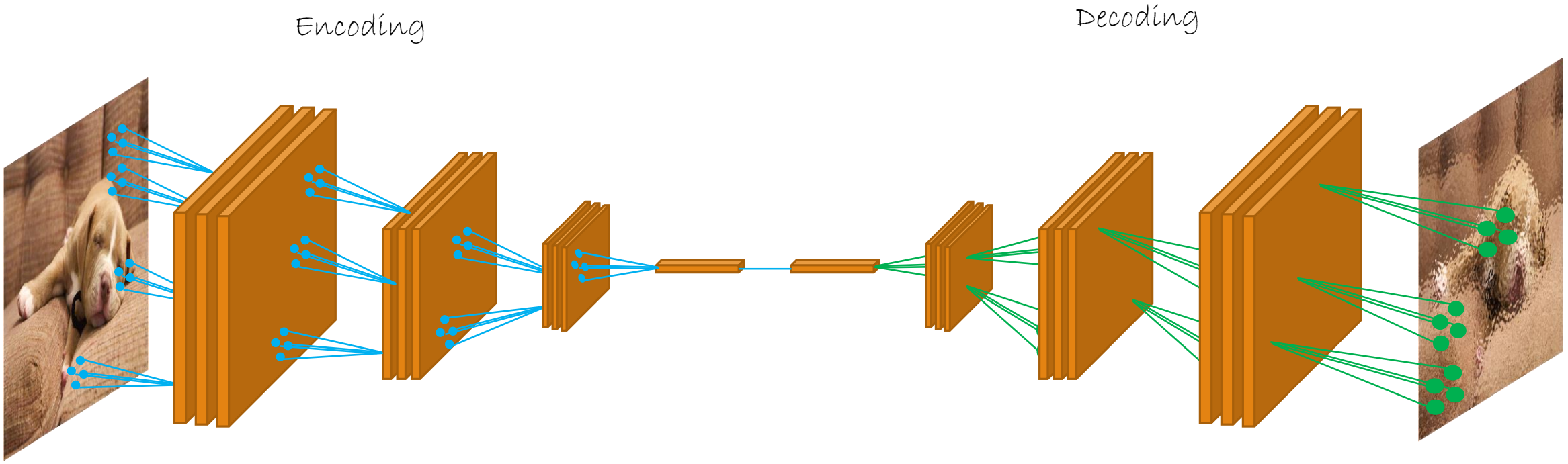


Autoencoders in a nutshell

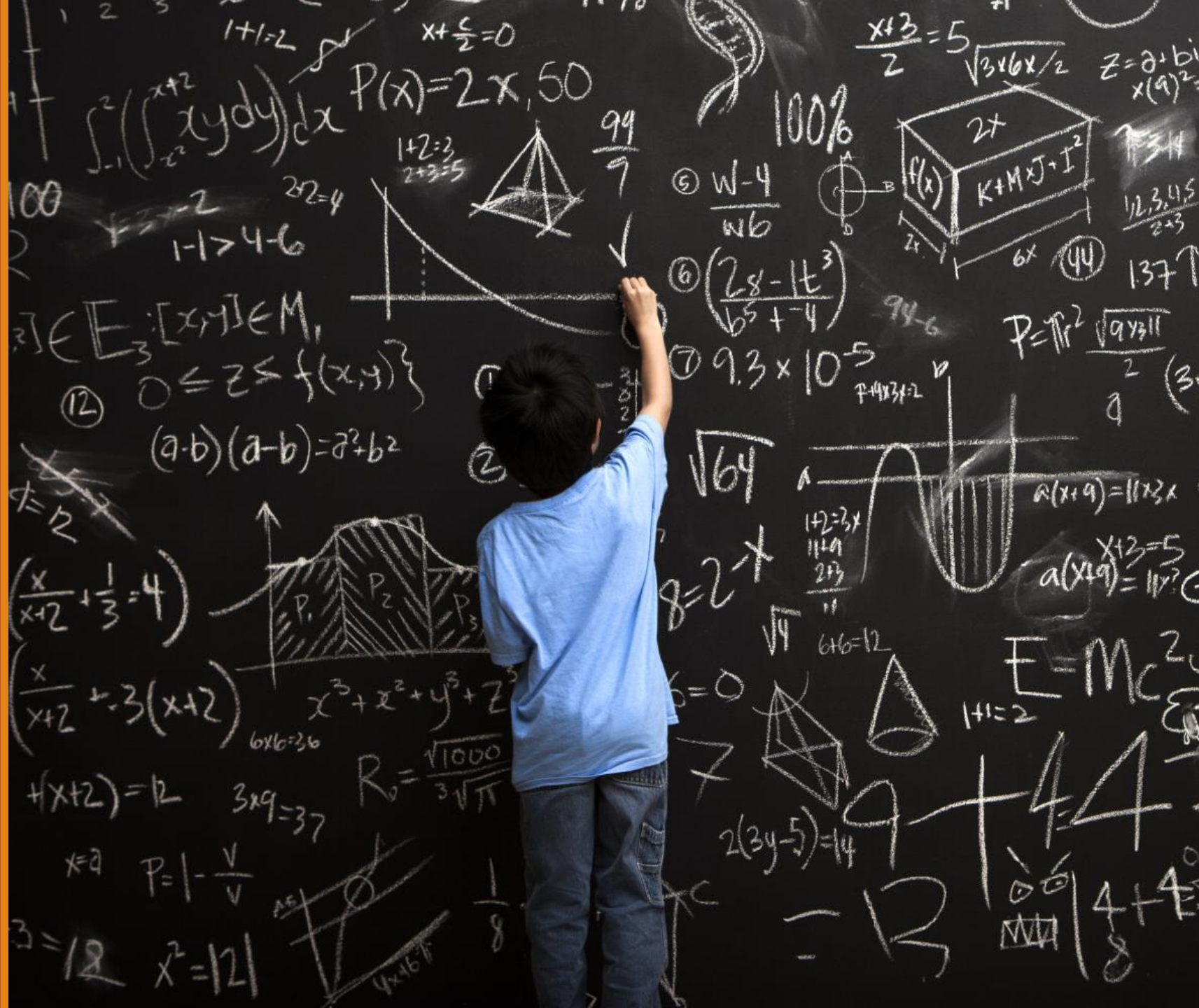
Autoencoders in a nutshell



Autoencoders in a nutshell



Theory of Deep Learning



Deep Learning Approximation Theory

- Deep Networks are universal approximators

Theorem *Let $\rho()$ be a bounded, non-constant continuous function. Let I_m denote the m-dimensional hypercube, and $C(I_m)$ denote the space of continuous functions on and $\epsilon > 0$, there exists $N>0$ and $v_i, w_i, b_i, i = 1, \dots, N$ such that $F(x) = \sum_{i \leq N} v_i \rho(w_i^T x + b_i)$ satisfies $\sup_{x \in I_m} |f(x) - F(x)| < \epsilon$.*

- Even a single hidden layer can approximate any function and represent any locally linear boundary.
- Of course, the theorem does not say what is the architecture or how to find the optimal parameters

Deep Learning Estimation Theory

Theorem *[Barron 92'] The mean integrated square error between the estimated network is bounded by* $O\left(\frac{C_f^2}{N}\right) + O\left(\frac{Nm}{K}\log K\right)$, *where* K *is the number of training points,* N *is the number of neurons,* m *is the input dimension, and* C_f *measures the global smoothness of* f.

- Combines approximation and estimation error
- Does not explain the relation between online or stochastic optimization and batch normalization
- Does not relate the generalization error with the network architecture

VC dimensions

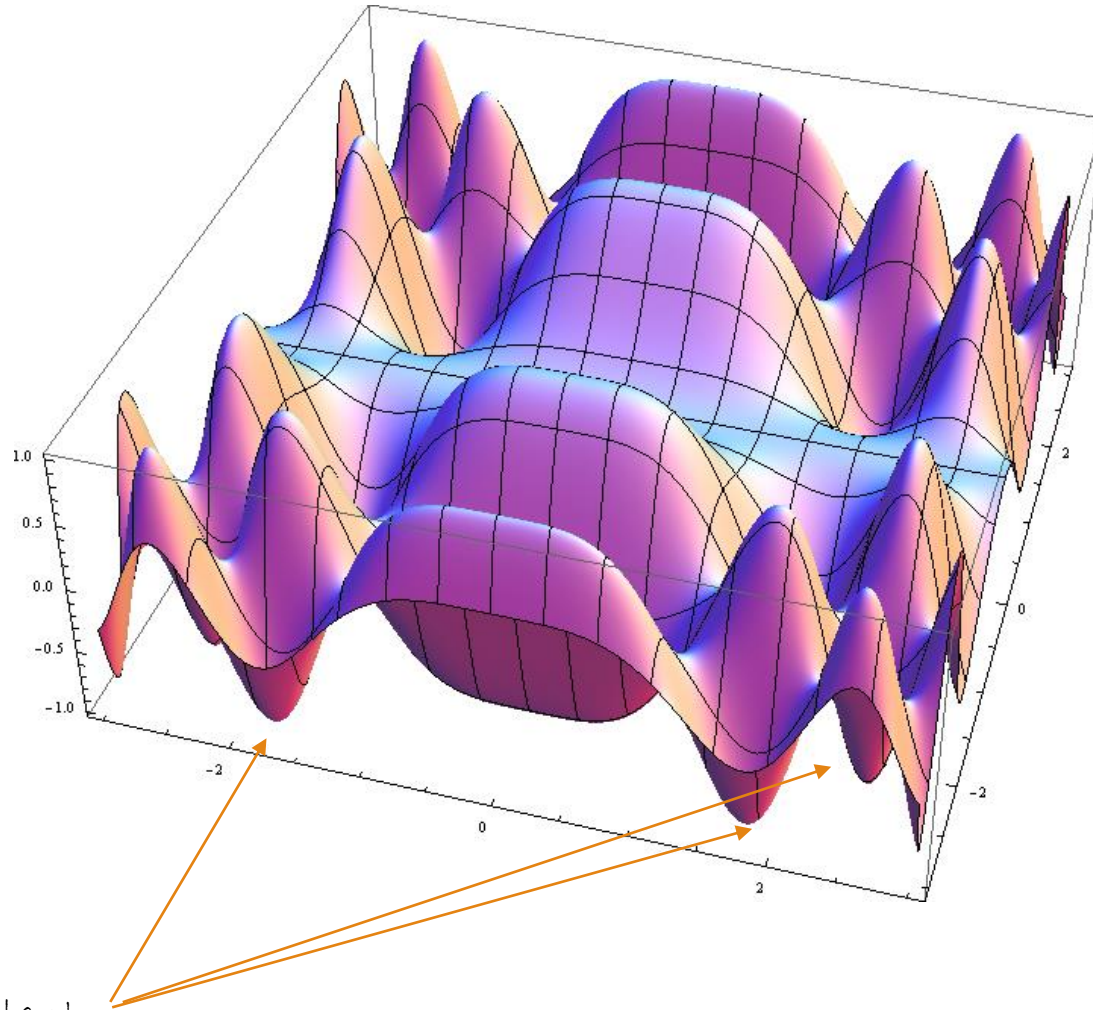
- **VC dimension.** The capacity of the model is measured by the number of configurations that it can shatter
  - [P. Bartlett et al., “Vapnik-Chernovenkis Dimension of Neural Nets”]
- If the capacity of the network is measured by the number of pieces in a piecewise linear approximation, the capacity increases with depth
  - [Montufar, Pascanu et al. 2014]
- Upper bound on the empirical risk of deep networks
- Do not explain why deep networks generalize that well
- Bounds are often not tight enough in practice, although this is the same for many other models too

So, why deep and not shallow?

- Although with two-layer (shallow) network, we can approximate all possible functions
  - Given the network layers are wide enough
- Deep architectures tend to be more efficient
  - Or otherwise, the network capacity given number of parameters is larger
- Also, deep and narrow architectures tend to generalize better than shallow and wide architectures

(Non)-convexity

- Highly non-convex
  - neural networks are stable and accurate enough though
  - So, is this more of a real problem or an interesting observation we should explain?
- Most local optima lie close to the global optima and hence all lead to equivalent solutions
- Whether you have one set of parameters or the other matters little in practice
- Often ensembles of models are to be preferred anyways
- Many assumptions made to obtain the result
- You cannot know if your local optimum is near the global optimum



Roughly equivalent

More (unanswered) theoretical questions

- Theory of unsupervised learning equivalent to statistical learning theory? How to do best unsupervised learning?
- Several intractable deep network losses
- Deep structured outputs
- Combining external static knowledge (e.g. Wikipedia knowledge) with stochastic methods like neural nets
- And many more …

First lab assignment

Deep Learning Framework

- Torch
  - www.torch.ch
- One of the top Deep Learning frameworks
  - Started from Y. LeCun's lab
- Currently used heavily by Google DeepMind and Facebook AI Research Lab (FAIR)
- Do not take it lightly, the language can be a real pickle in the beginning
  - With iTorch (from Jupyter) you can make very nice visualizations

Content & Goal

- Learn how to perform standard mathematical operations
- Learn how to visualize data, results, etc.
- Learn how to use optimization packages
- Learn to derive gradients and use them to optimize a function
- Implement a perceptron on toy data
- Deliver by Feb 10th, 23:59

Summary

- A brief history of neural networks and deep learning
- What is deep learning and why is it happening now?
- What types of deep learning exist?
- Demos and tasks where deep learning is currently the preferred choice of models

UVA DEEP LEARNING COURSE EFSTRATIOS GAVVES
INTRODUCTION ON NEURAL NETWORKS AND DEEP LEARNING - PAGE 122

Next lecture

- Learn how to describe neural networks as a pipeline of layers and modules
- Learn how to build your own modules
- Learn how to optimize modular neural networks efficiently in theory and in practice

UVA DEEP LEARNING COURSE EFSTRATIOS GAVVES
INTRODUCTION ON NEURAL NETWORKS AND DEEP LEARNING - PAGE 123

For enthusiastic students

- A more comprehensive review of NN history
- A 'Brief' History of Neural Nets and Deep Learning, Part 1, 2, 3, 4
- Deep Learning in a Nutshell: History and Training
- The Brain vs Deep Learning