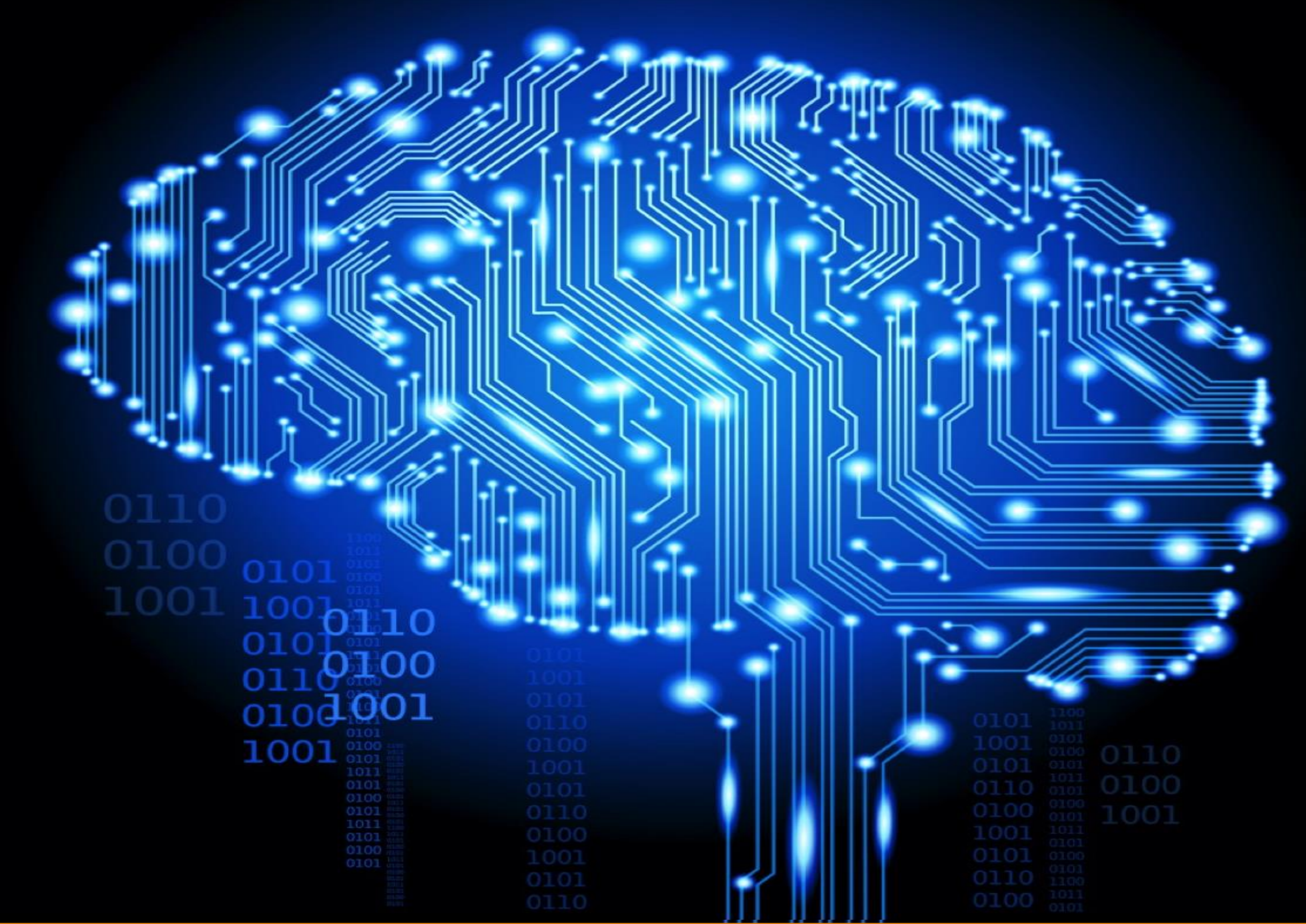


Lecture 1: Introduction to Neural Networks and Deep Learning

Deep Learning @ UvA

Prerequisites

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

Course overview

- Idea: Go in depth in theory & get hands-on practical experience
- What will you learn?
  - *How to train Deep Learning models*
  - *Neural Networks for Computer Vision*
  - *Neural Networks for Language*
  - *Unsupervised and Bayesian Deep Learning*
  - *Deep Reinforcement Learning*
- All material uploaded on the course website
- Book on Neural Networks and Deep Learning from Y. Bengio

Course Logistics

- Course: Theory (4 hours per week) + Labs (4 hours per week)
- Rooms are not always the same, check Datanose
- Final grade = 50% from lab assignments + 50% from final exam
  - Exam moved to Dec 20, 13.00-15.00, check Datanose

Practicals

- 6 lab assignments=5 practical assignments + 1 presentation
  - Equally weighed
- Practical assignments done individually
  - Python + Tensorflow
- Presentation in groups of 3
  - Pick your team & paper by Nov 30
  - Present end of December, schedule announced after Nov 30
  - 7 min per presentation, 3 min for questions, we will give you template
  - Graded: 50% presentation skills (group), 50% per student Q&A (individual)

Who we are and how to reach us

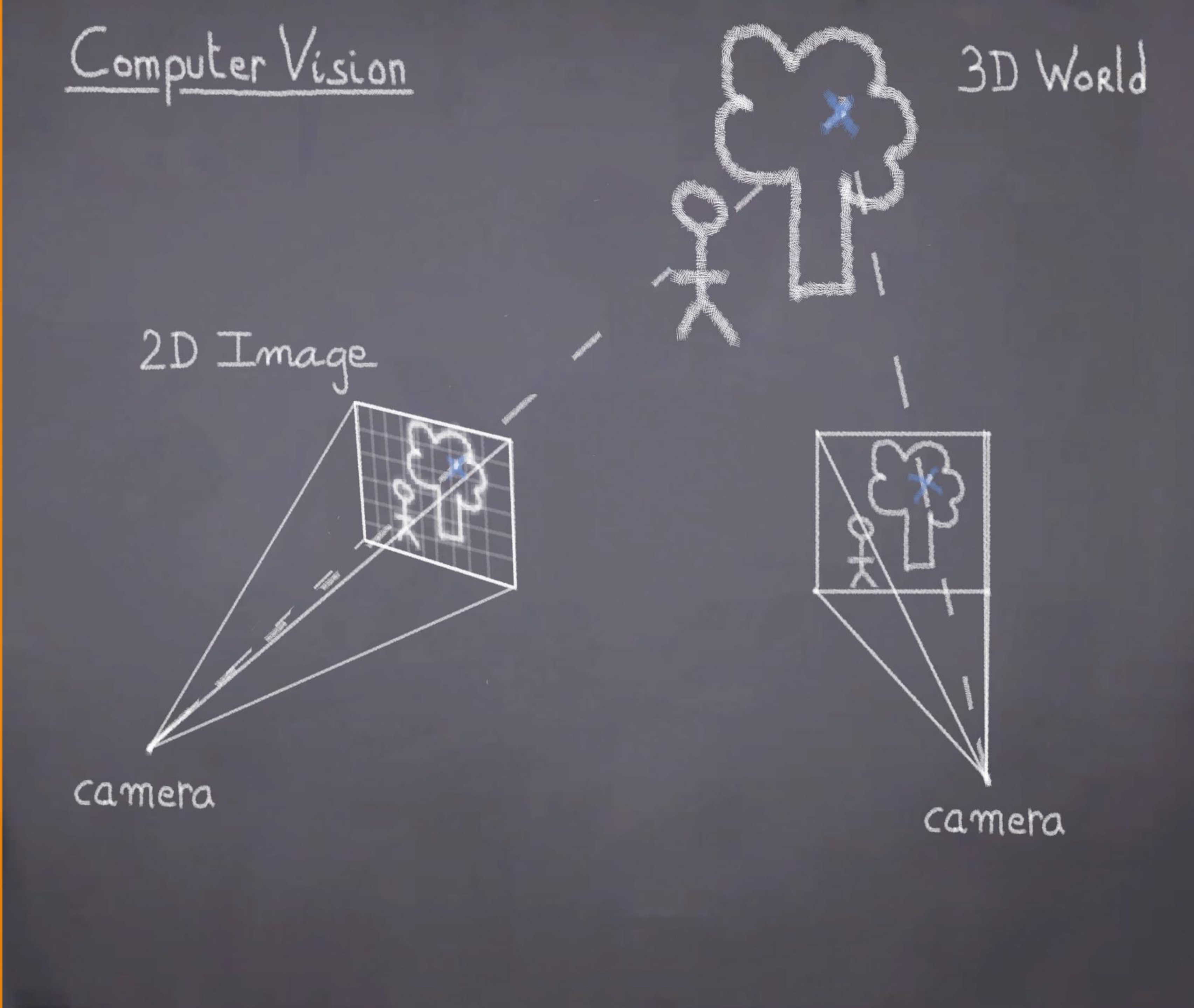
- Efstratios Gavves, TAs: Kirill Gavrilyuk, Berkay Kicanaoglu, Patrick Putzky
  - uva.deeplearning@gmail.com

- Course website: http://uvadlc.github.io
  - Lecture sides & notes, practicals
- Virtual classroom
  - Piazza: www.piazza.com/university_of_amsterdam/spring2016/uvadlc/home
  - Datanose: https://datanose.nl/#course[55904]

Lecture Overview

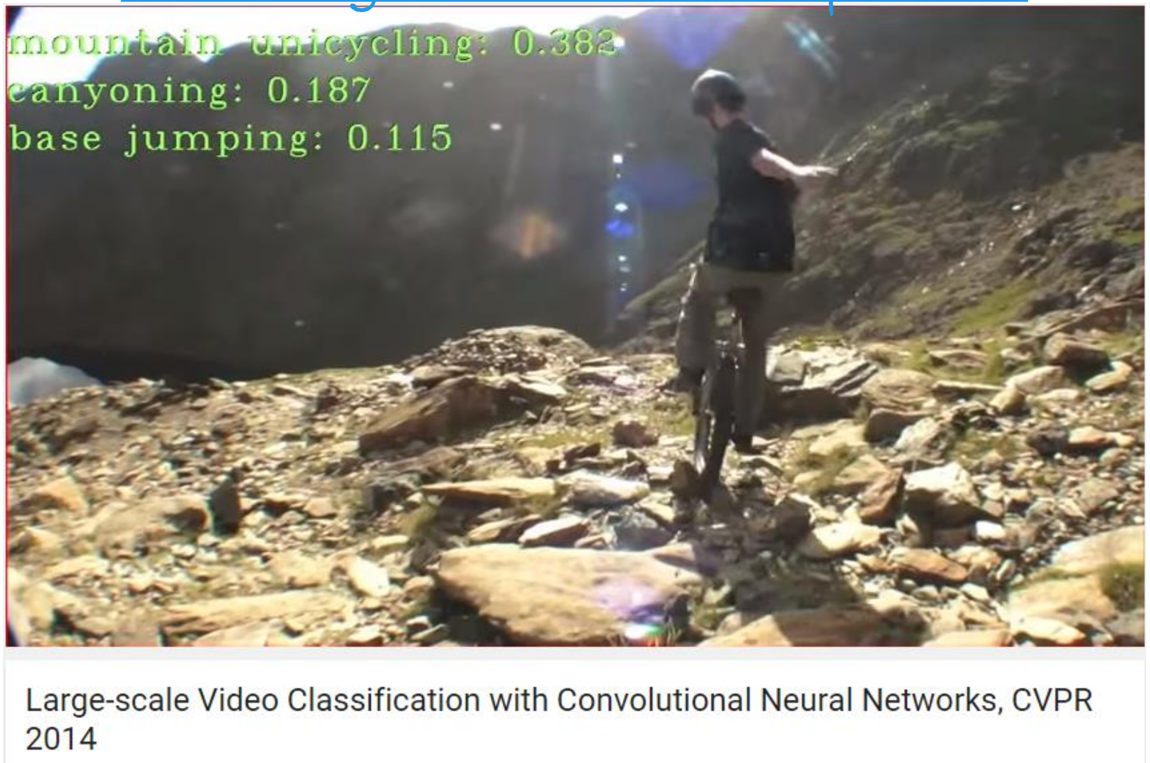
- Deep Learning in
  - Computer Vision
  - Natural Language Processing (NLP)
  - Speech
  - Robotics and AI
  - Music and the arts!
- A brief history of Neural Networks and Deep Learning
- Basics of Neural Networks

Deep Learning in Computer Vision



Object and activity recognition

Click to go to the video in Youtube

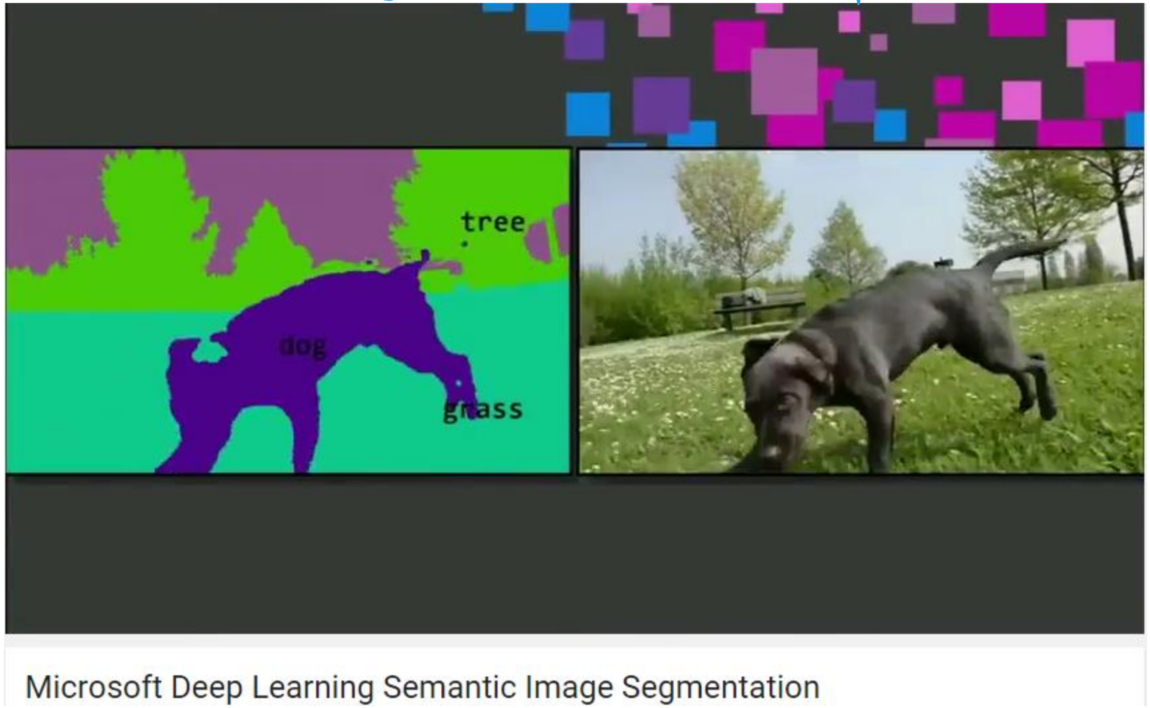


mountain unicycling: 0.382
canyoning: 0.187
base jumping: 0.115

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

Object detection and segmentation

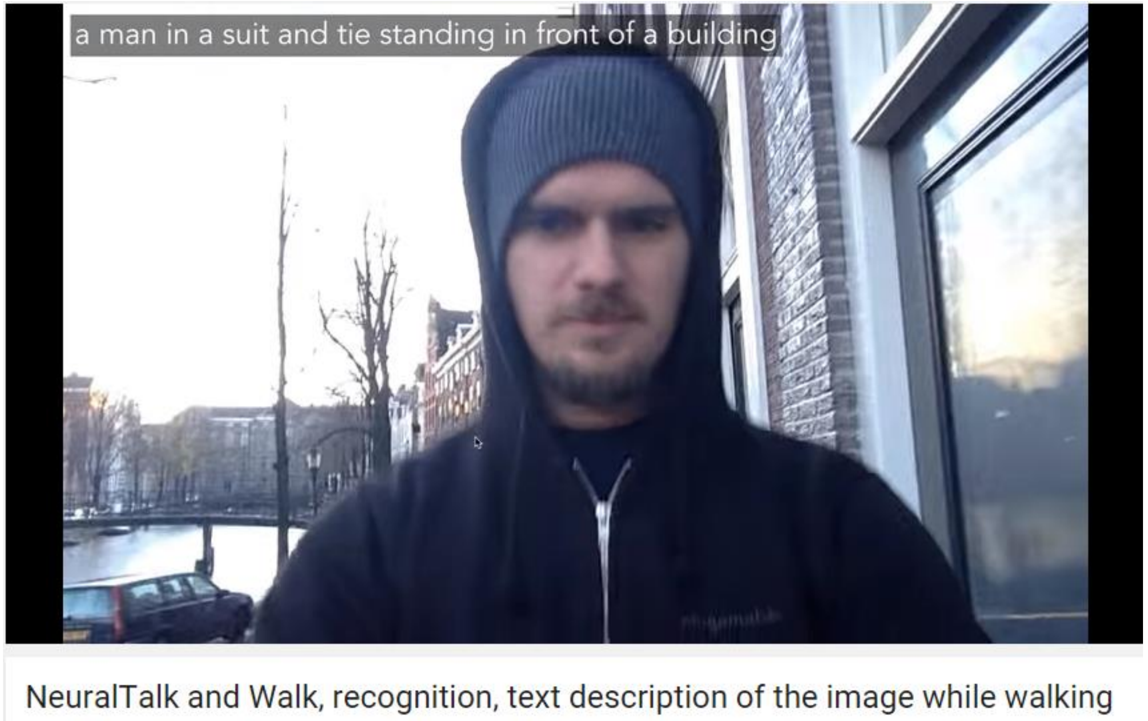
Click to go to the video in Youtube



Microsoft Deep Learning Semantic Image Segmentation

Image captioning and Q&A

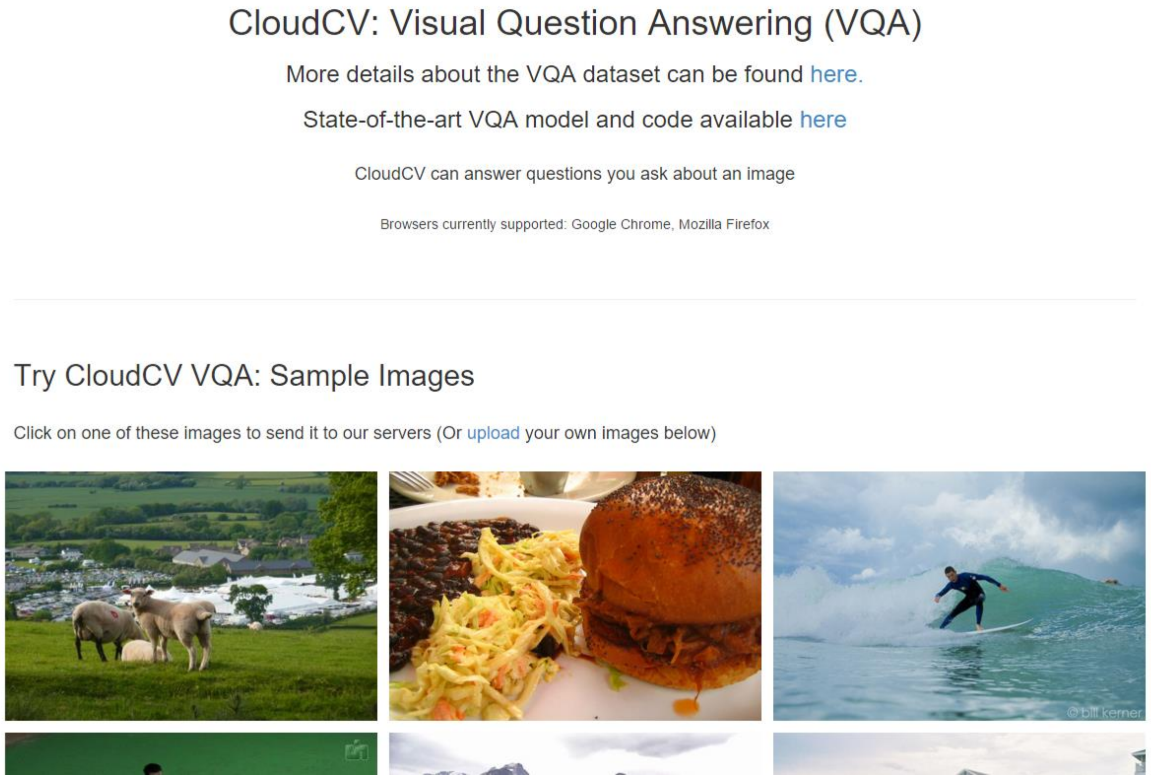
Click to go to the video in Youtube



a man in a suit and tie standing in front of a building

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

Click to go to the website



CloudCV: Visual Question Answering (VQA)

More details about the VQA dataset can be found here.

State-of-the-art VQA model and code available here

CloudCV can answer questions you ask about an image

Browsers currently supported: Google Chrome, Mozilla Firefox

Try CloudCV VQA: Sample Images

Click on one of these images to send it to our servers (Or upload your own images below)

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

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

Game AI

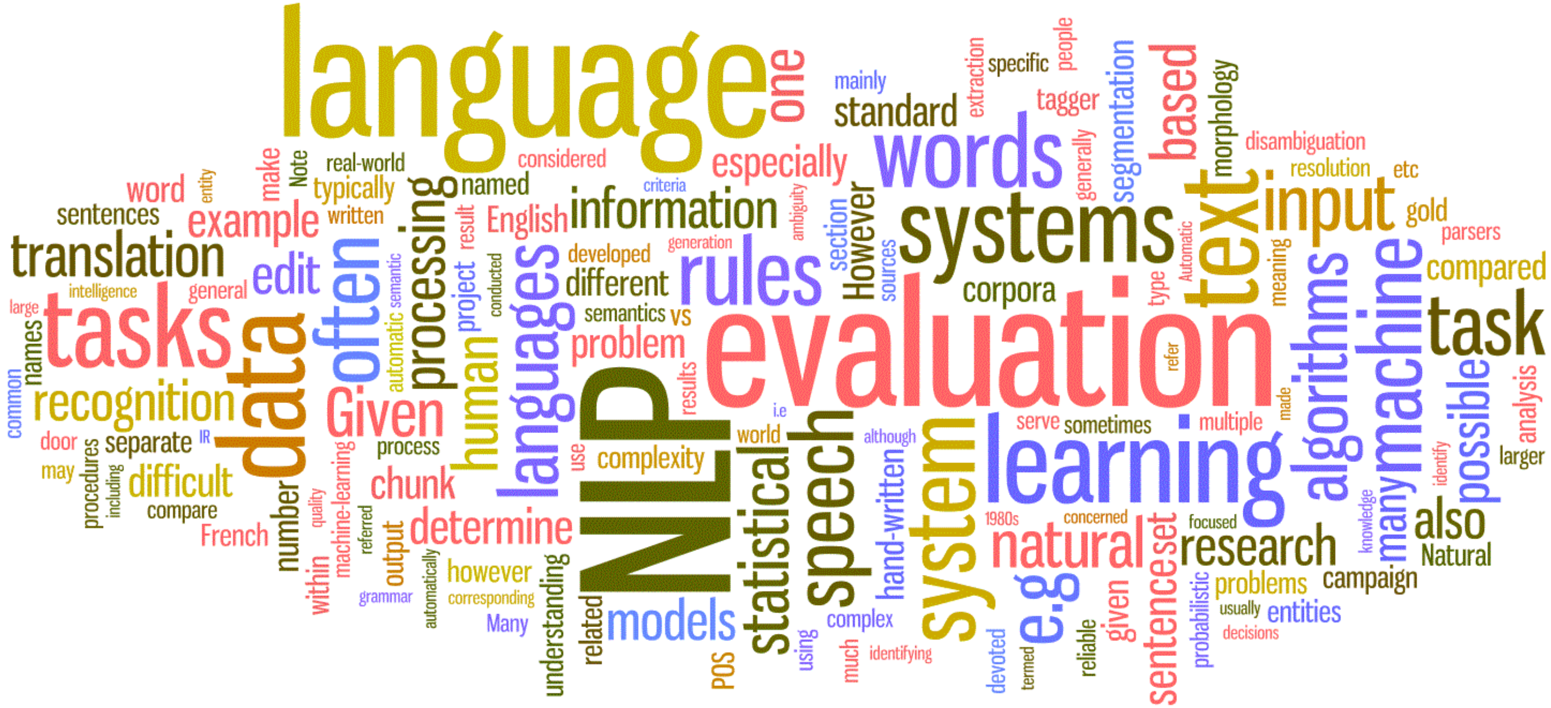
Click to go to the video in Youtube

Google DeepMind's Deep Q-learning playing Atari Breakout

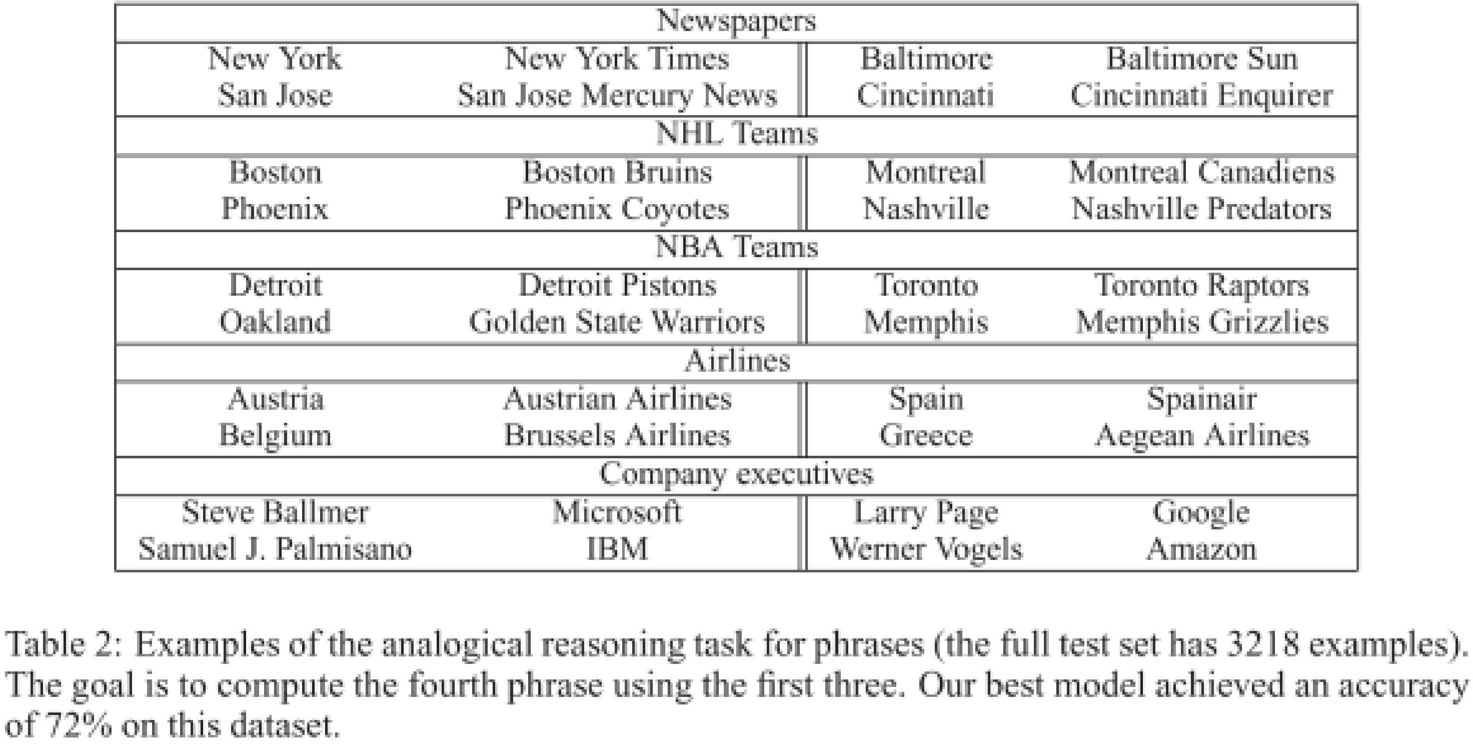
Why should we be impressed?

- 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



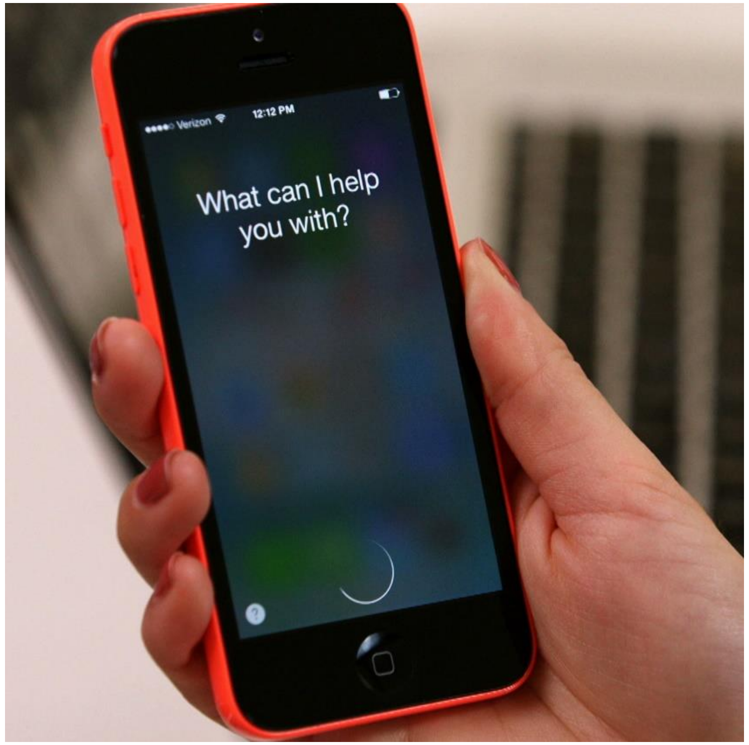
Word and sentence representations



Newspapers			
New York	New York Times	Baltimore	Baltimore Sun
San Jose	San Jose Mercury News	Cincinnati	Cincinnati Enquirer
NHL Teams			
Boston	Boston Bruins	Montreal	Montreal Canadiens
Phoenix	Phoenix Coyotes	Nashville	Nashville Predators
NBA Teams			
Detroit	Detroit Pistons	Toronto	Toronto Raptors
Oakland	Golden State Warriors	Memphis	Memphis Grizzlies
Airlines			
Austria	Austrian Airlines	Spain	Spainair
Belgium	Brussels Airlines	Greece	Aegean Airlines
Company executives			
Steve Ballmer	Microsoft	Larry Page	Google
Samuel J. Palmisano	IBM	Werner Vogels	Amazon

Table 2: Examples of the analogical reasoning task for phrases (the full test set has 3218 examples). The goal is to compute the fourth phrase using the first three. Our best model achieved an accuracy of 72% on this dataset.

Speech recognition and Machine translation

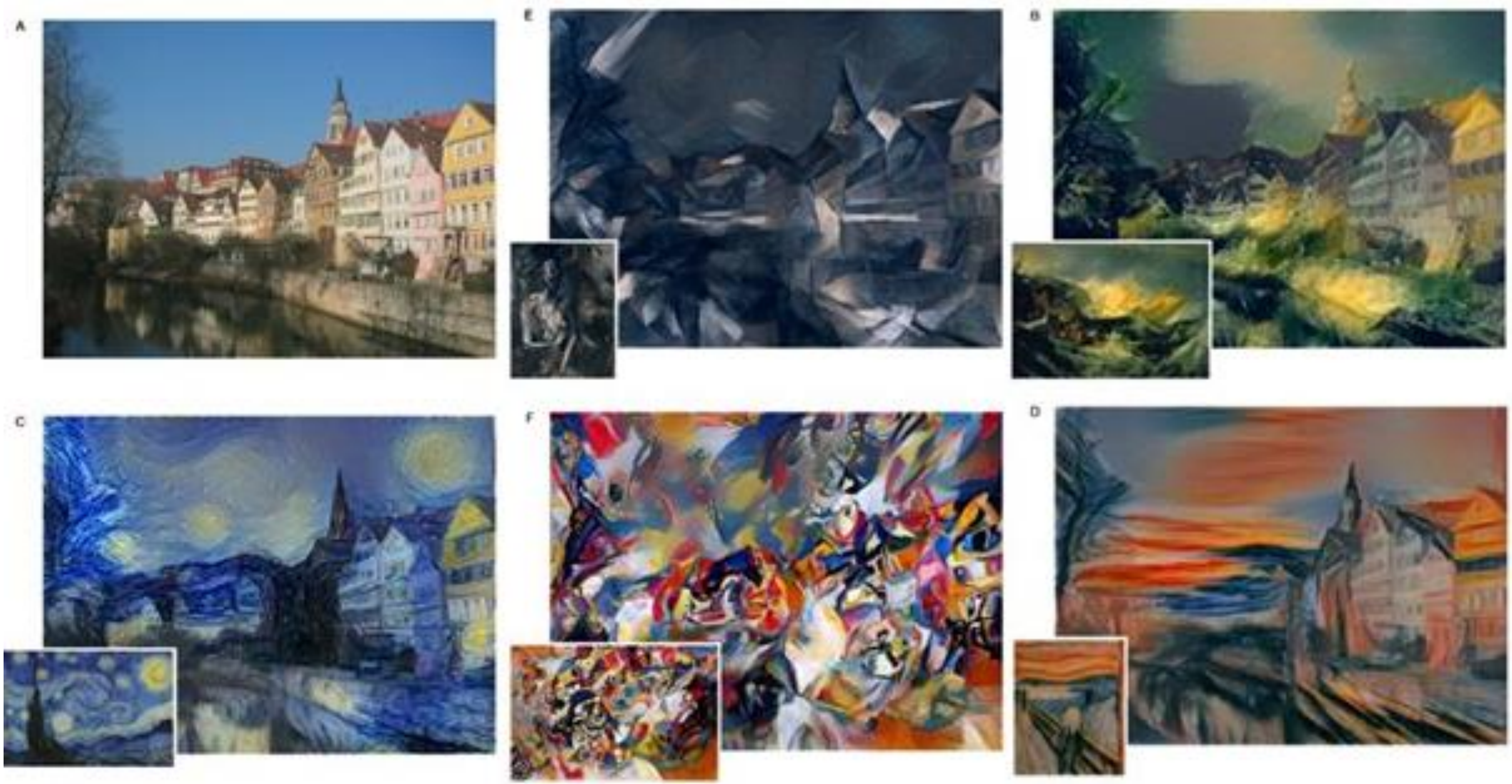


Why should we be impressed?

- 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

Imitating famous painters

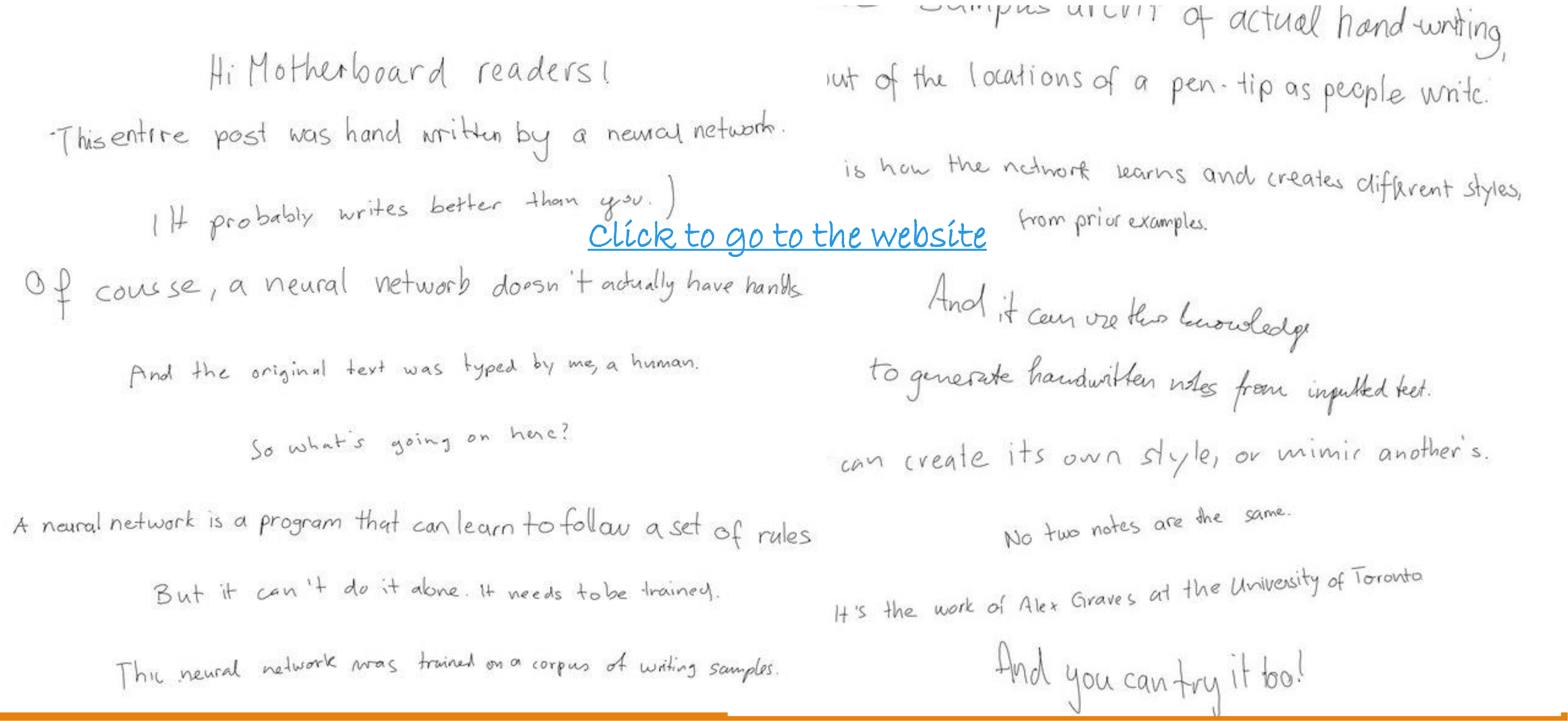


Or dreaming …

Click to go to the video in Youtube

Journey Through the Layers of the Mind

Handwriting



Hi Motherboard readers!

This entire post was hand written by a neural network.

(It probably writes better than you.)

Of course, a neural network doesn't actually have hands.

And the original text was typed by me, a human.

So what's going on here?

A neural network is a program that can learn to follow a set of rules

But it can't do it alone. It needs to be trained.

This neural network was trained on a corpus of writing samples.

samples drawn of actual hand-writing,

ut of the locations of a pen-tip as people write.

is how the network learns and creates different styles,

from prior examples.

And it can use this knowledge

to generate handwritten notes from inputted text.

can create its own style, or mimic another's.

No two notes are the same.

It's the work of Alex Graves at the University of Toronto

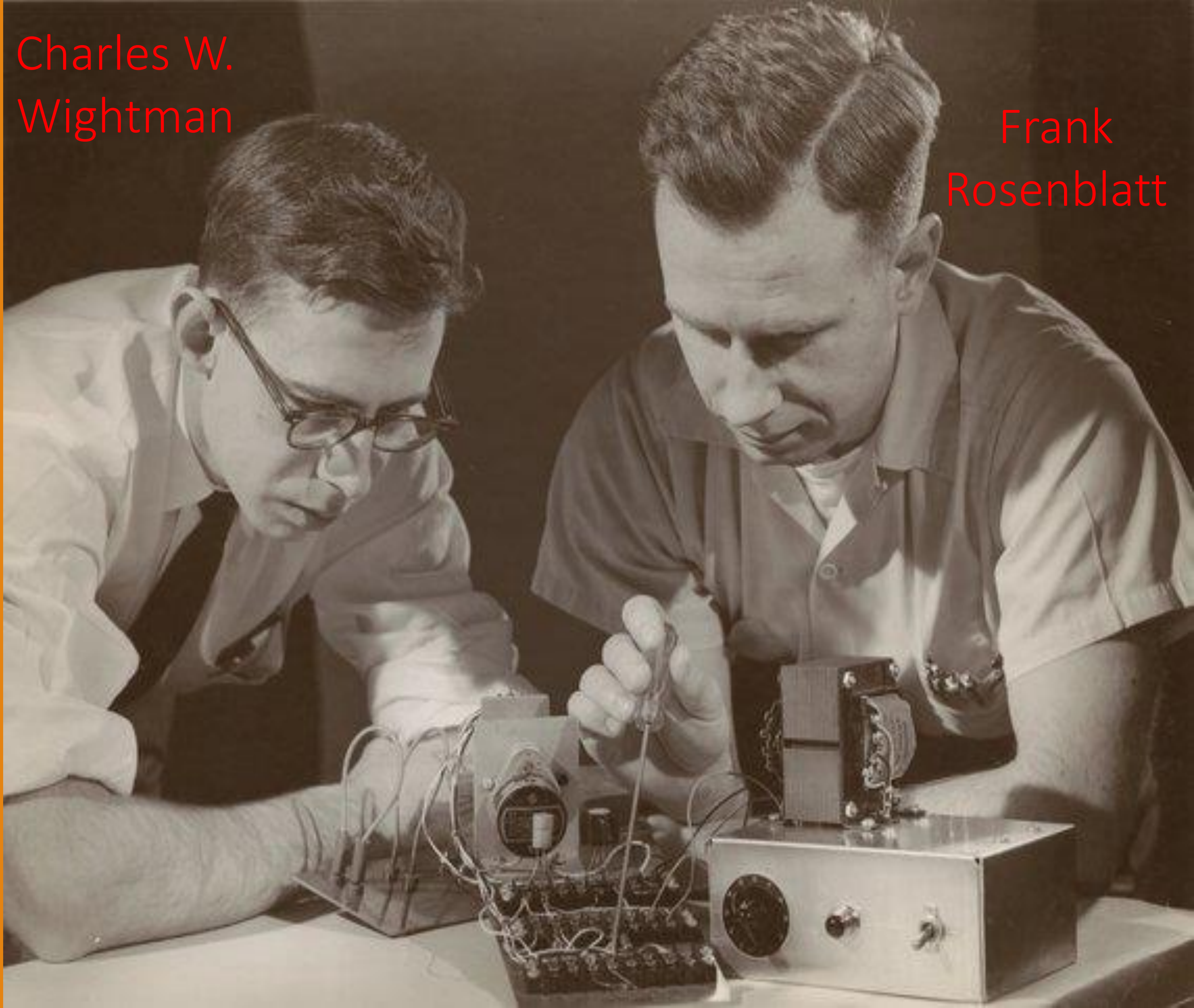
And you can try it too!

Click to go to the website

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?

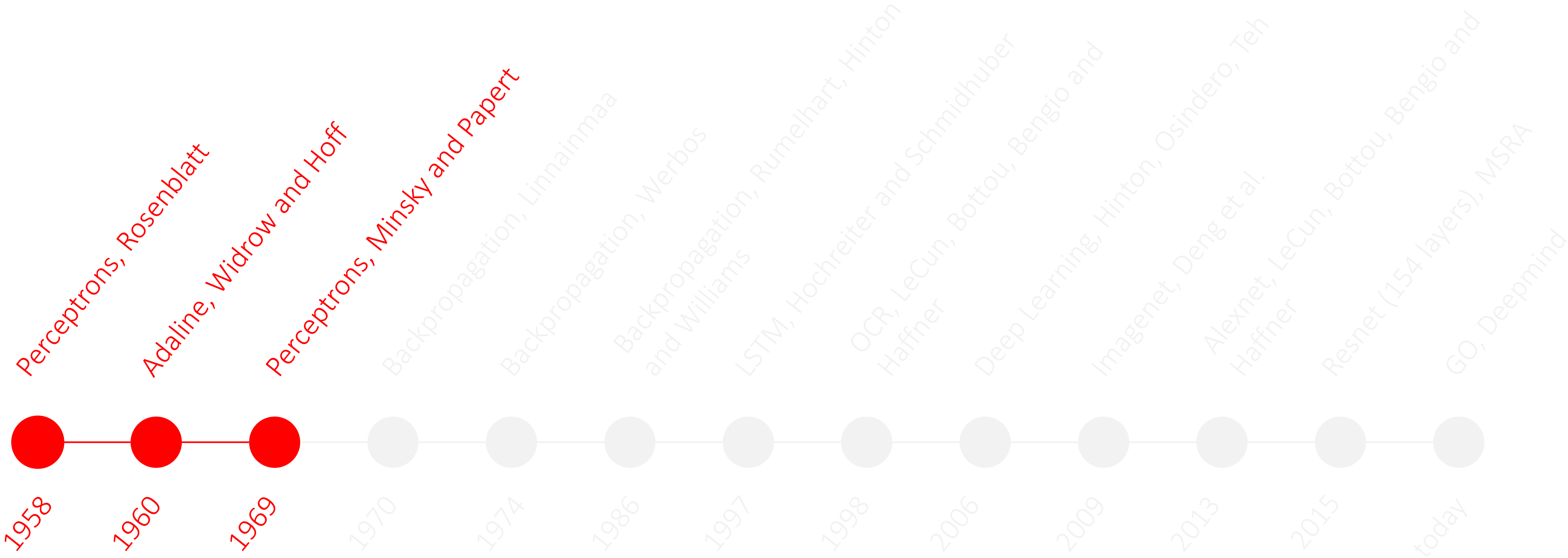
A brief history of Neural Networks & Deep Learning



Charles W. Wightman

Frank Rosenblatt

First appearance (roughly)



Perceptrons, Rosenblatt

1958

Adaline, Widrow and Hoff

1960

Perceptrons, Minsky and Papert

1969

Backpropagation, Linnainmaa

1970

Backpropagation, Werbos

1974

Backpropagation, Rumelhart, Hinton and Williams

1986

LSTM, Hochreiter and Schmidhuber

1997

OCR, LeCun, Bottou, Bengio and Haffner

1998

Deep Learning, Hinton, Osindero, Teh

2006

Imagenet, Deng et al.

2009

Alexnet, LeCun, Bottou, Bengio and Haffner

2013

Resnet (154 layers), MSRA

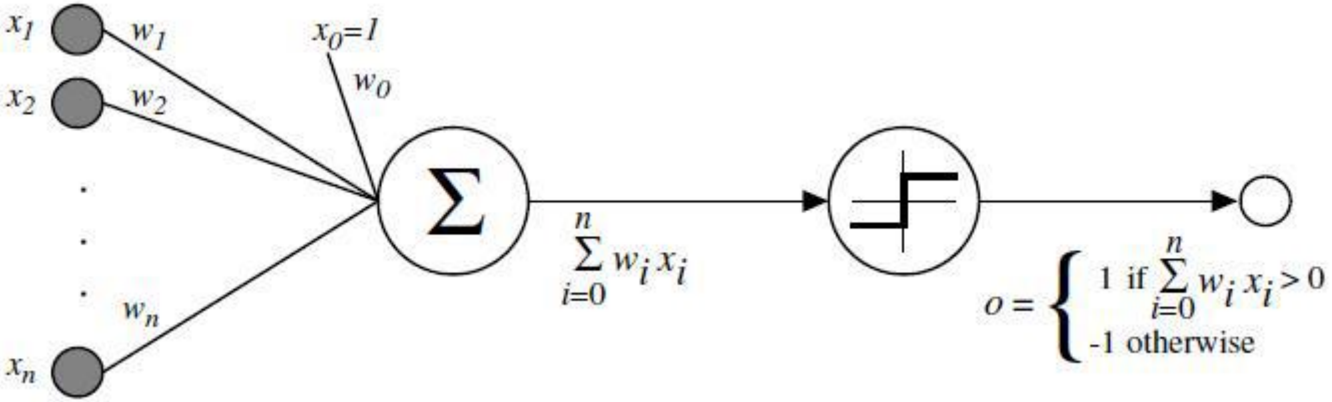
2015

GO, Deepmind

today

Perceptrons

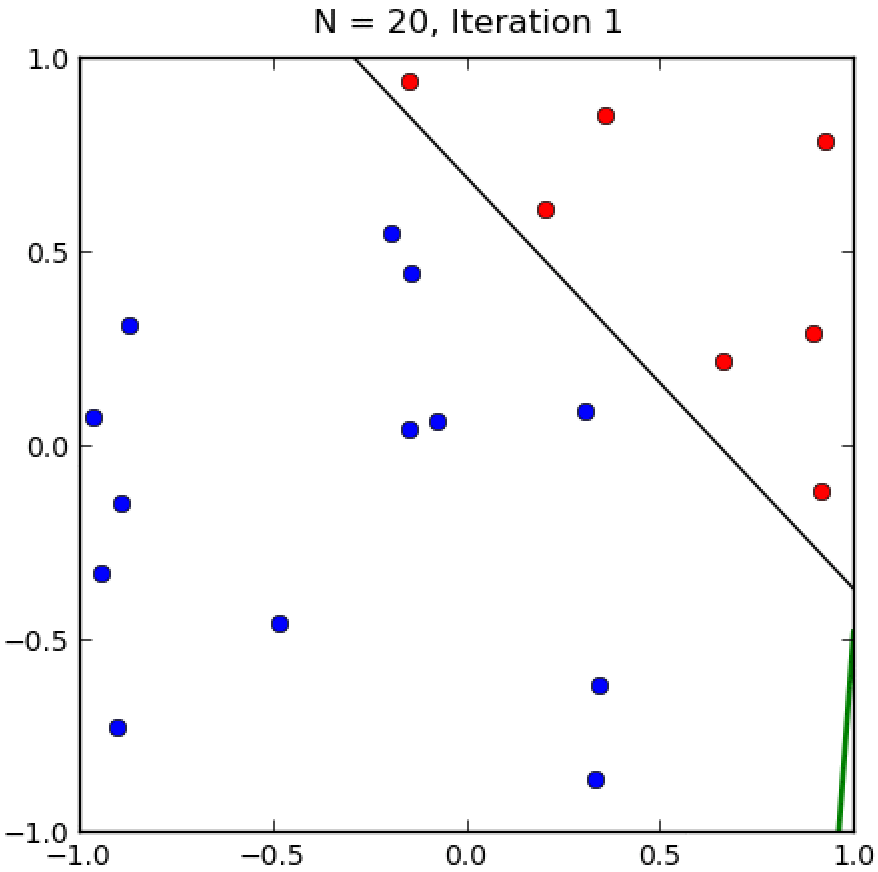
- Rosenblatt proposed a machine for binary classifications
- Main idea
  - One weight w_i per input x_i
  - Multiply weights with respective inputs and add bias x_0 =+1
  - If result larger than threshold return 1, otherwise 0



x_1
x_2
x_n
w_1
w_2
w_n
$x_0=1$
w_0
Σ
$\sum_{i=0}^{n} w_i x_i$
$o = \begin{cases} 1 \text{ if } \sum_{i=0}^{n} w_i x_i > 0 \\ -1 \text{ otherwise} \end{cases}$

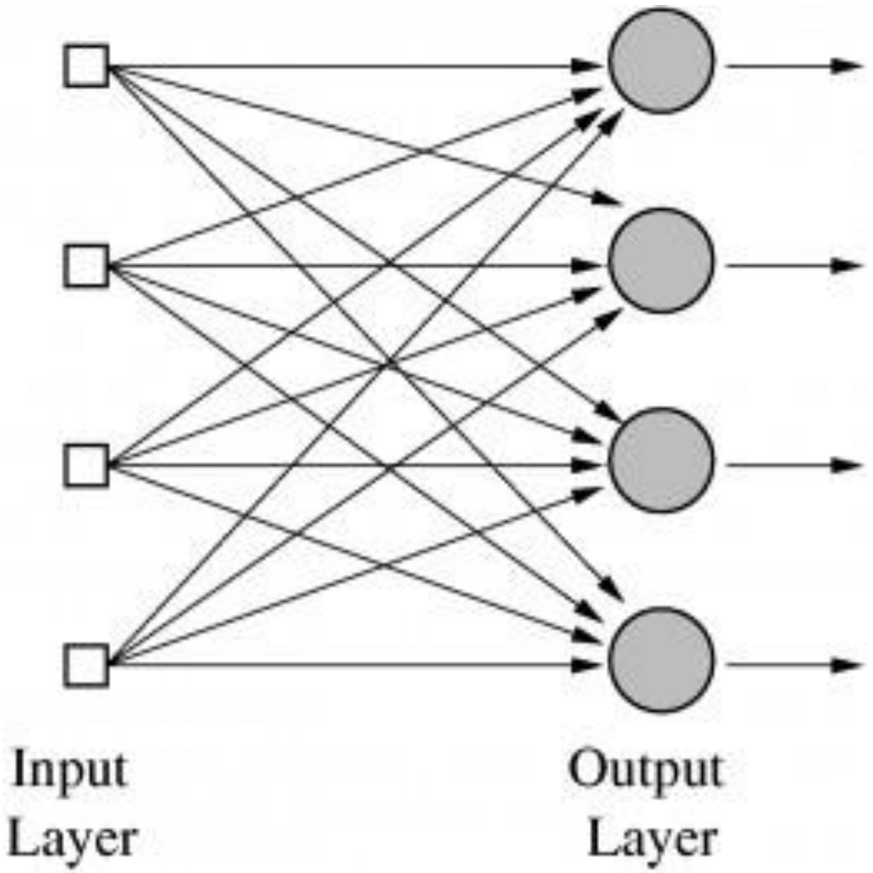
Training a perceptron

- 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 $\widehat{y_i} = 0$ and $y_i = 1$, increase weights
    - If the output was $\widehat{y_i} = 1$ 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
- 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)

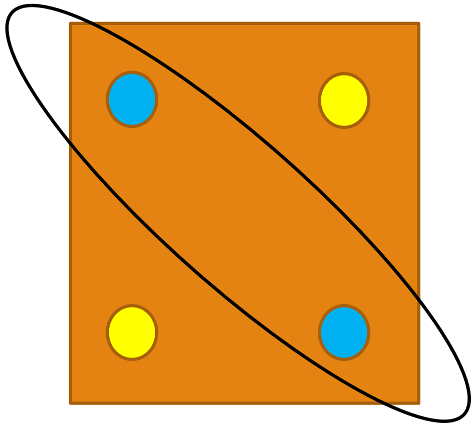


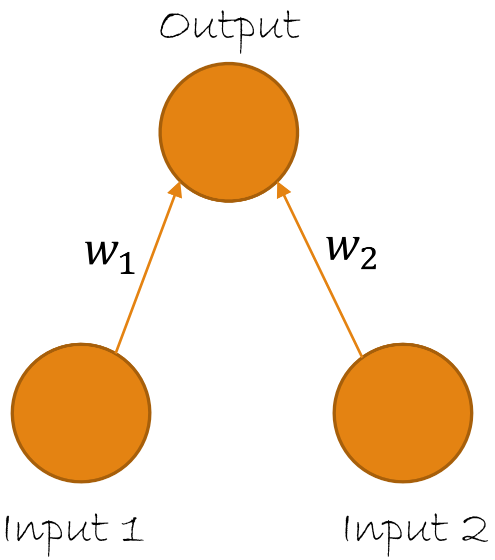
XOR & Multi-layer Perceptrons

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

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

Inconsistent!!

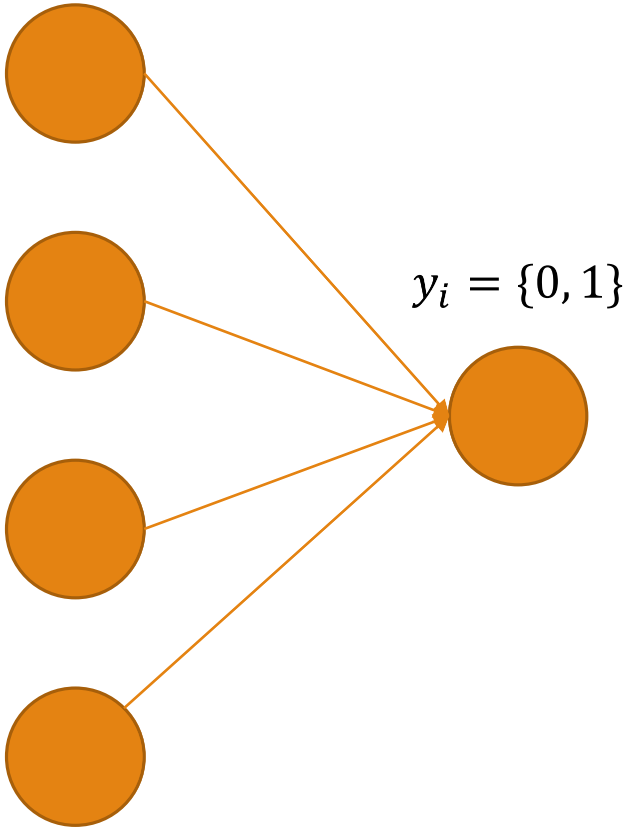




Input 1	Input 2	Output
1	1	0
1	0	1
0	1	1
0	0	0

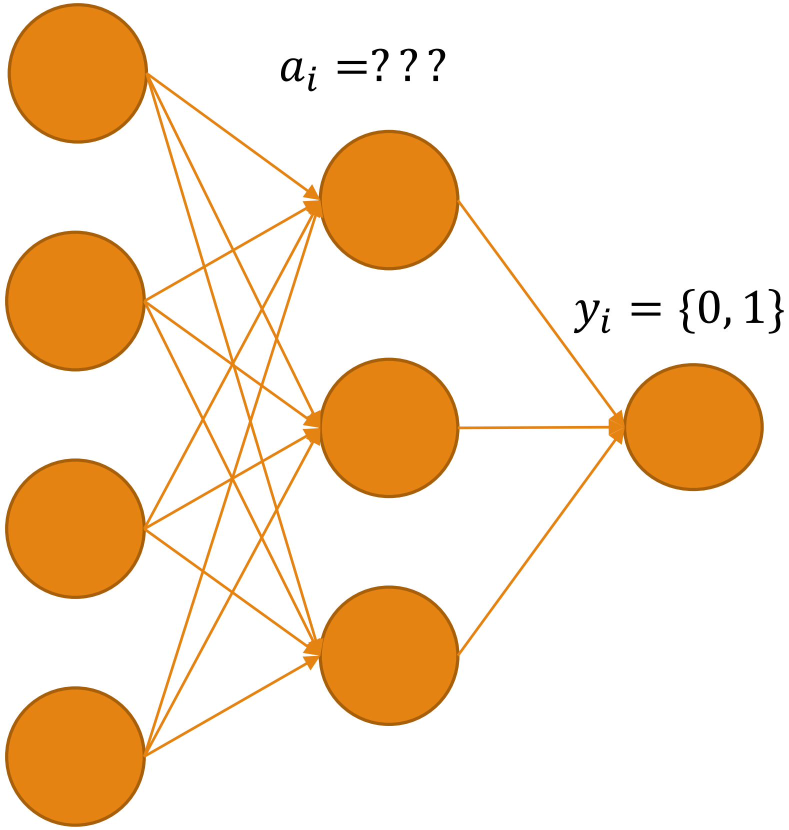
Minsky & Multi-layer perceptrons

- Interestingly, Minksy **never said** XOR cannot be solved by neural networks
  - Only that XOR cannot be solved with <u>1 layer</u> 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

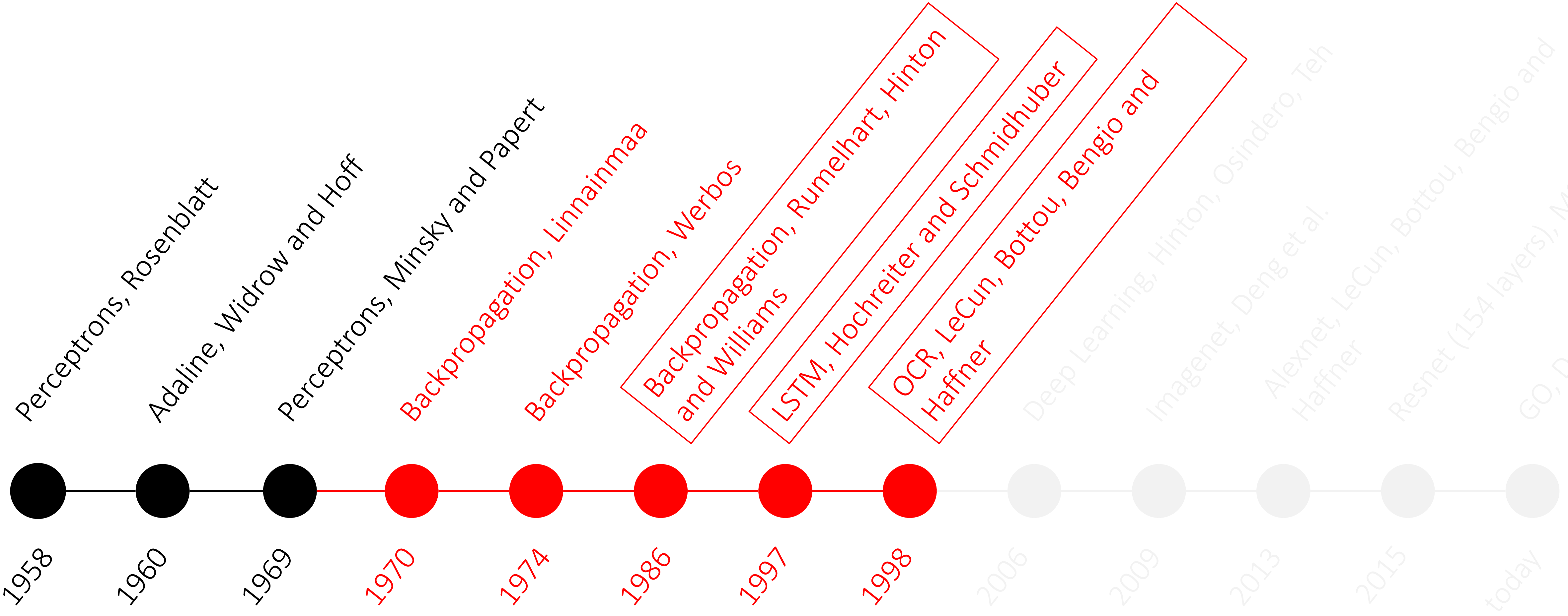


Minsky & Multi-layer perceptrons

- Interestingly, Minksy **never said** XOR cannot be solved by neural networks
  - Only that XOR cannot be solved with <u>**1 layer**</u> 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



Perceptrons, Rosenblatt

1958

Adaline, Widrow and Hoff

1960

Perceptrons, Minsky and Papert

1969

Backpropagation, Linnainmaa

1970

Backpropagation, Werbos

1974

Backpropagation, Rumelhart, Hinton and Williams

1986

LSTM, Hochreiter and Schmidhuber

1997

OCR, LeCun, Bottou, Bengio and Haffner

1998

Deep Learning, Hinton, Osindero, Teh

2006

Imagenet, Deng et al.

2009

Alexnet, LeCun, Bottou, Bengio and Haffner

2013

Resnet (154 layers), MSRA

2015

GO, Deepmind

today

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

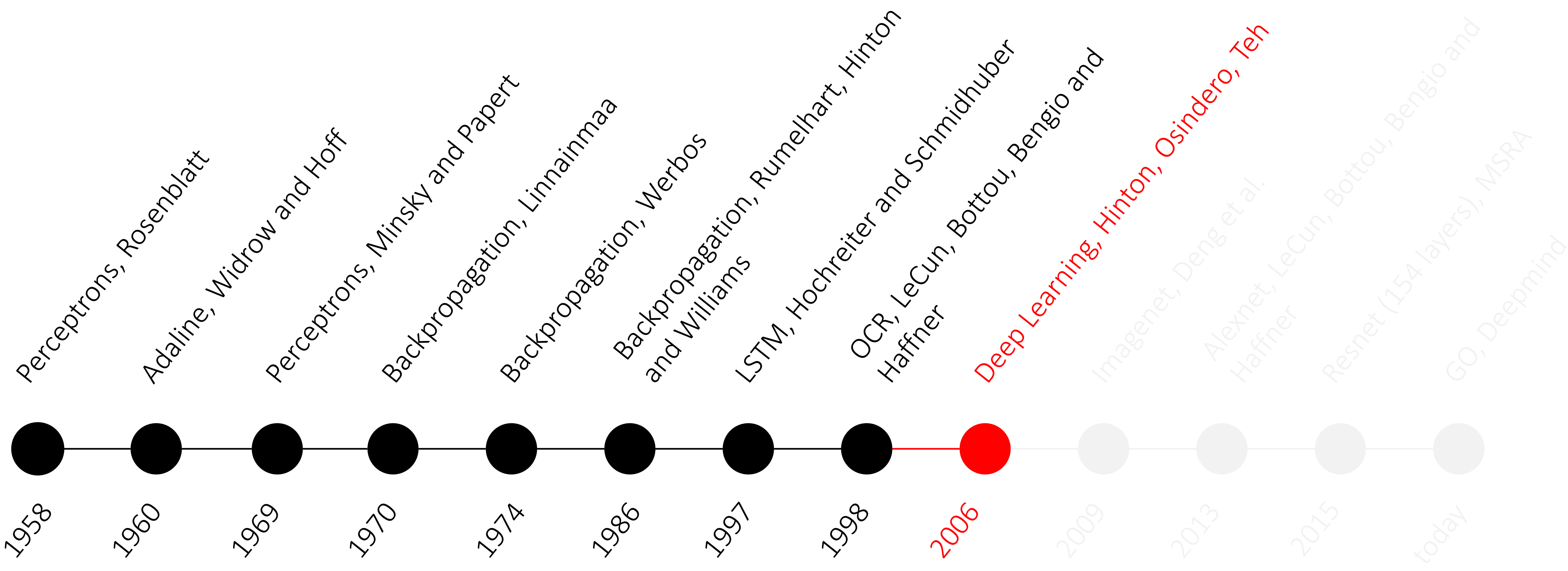
Recurrent networks

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

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
  - Similar accuracies in the same tasks
  - Neural networks could not improve beyond a few layers
  - Kernel Machines included much fewer heuristics & nice proofs on generalization
- As a result, once again the AI community turns away from Neural Networks

The thaw of the “AI winter”



1958 Perceptrons, Rosenblatt

1960 Adaline, Widrow and Hoff

1969 Perceptrons, Minsky and Papert

1970 Backpropagation, Linnainmaa

1974 Backpropagation, Werbos

1986 Backpropagation, Rumelhart, Hinton and Williams

1997 LSTM, Hochreiter and Schmidhuber

1998 OCR, LeCun, Bottou, Bengio and Haffner

2006 Deep Learning, Hinton, Osindero, Teh

2009 Imagenet, Deng et al.

2013 Alexnet, LeCun, Bottou, Bengio and Haffner

2015 Resnet (154 layers), MSRA

today GO, Deepmind

Neural Network and Deep Learning problems

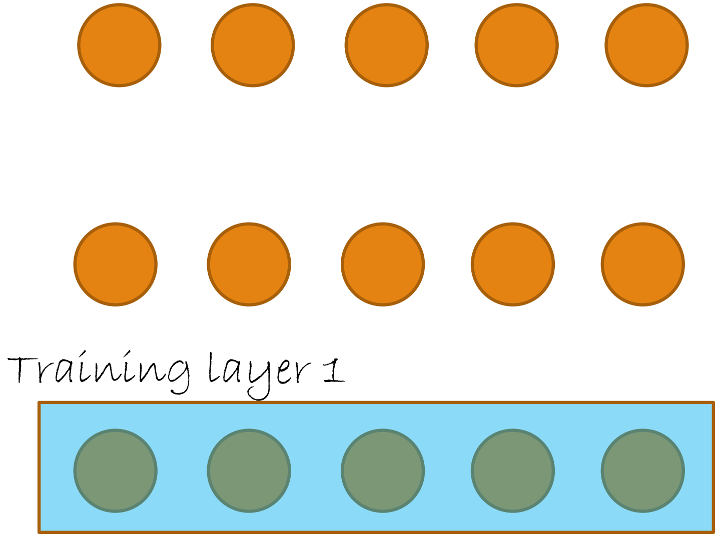
- Lack of processing power
  - No GPUs at the time
- Lack of data
  - No big, annotated datasets at the time
- Overfitting
  - Because of the above, models could not generalize all that well
- Vanishing gradient
  - While learning with NN, you need to multiply several numbers $a_1 \cdot a_2 \cdot \dots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

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

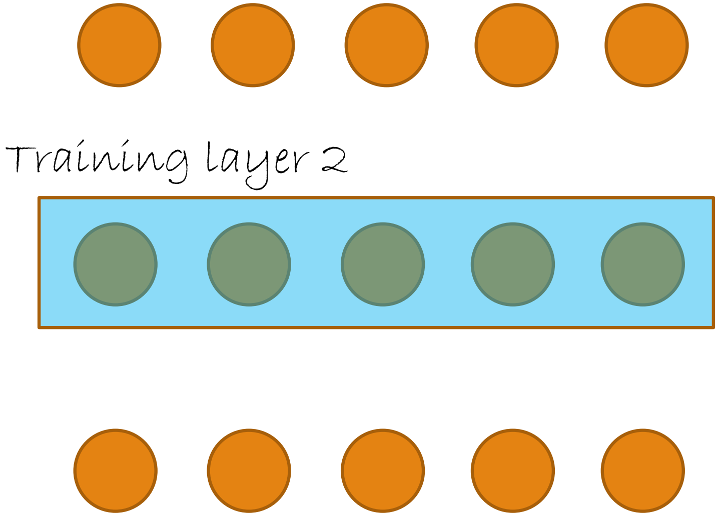
Deep Learning arrives

- 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



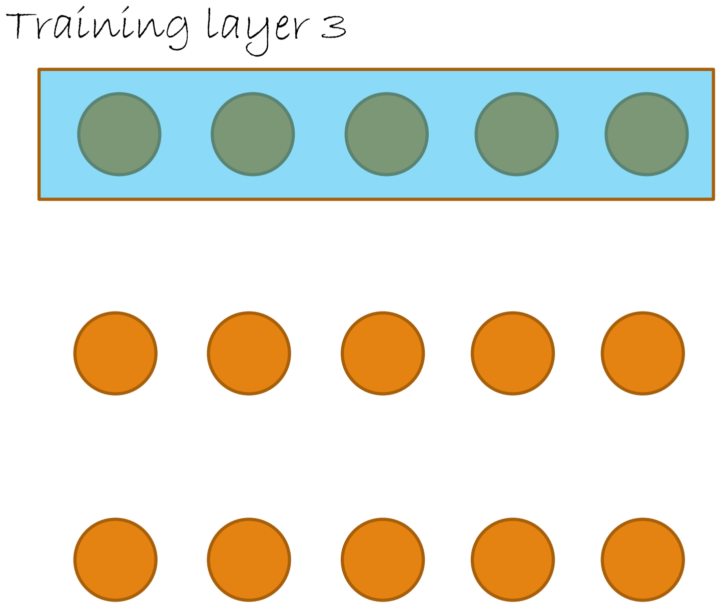
Deep Learning arrives

- 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

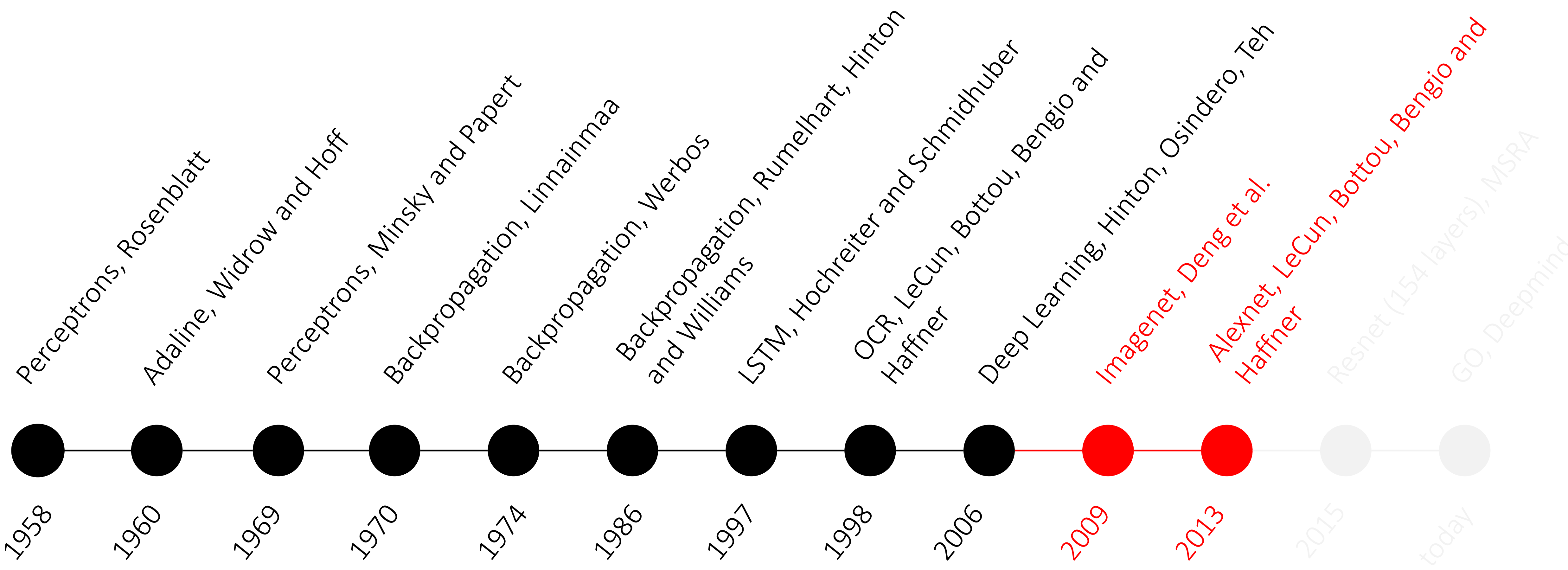


Deep Learning arrives

- 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



1958 Perceptrons, Rosenblatt

1960 Adaline, Widrow and Hoff

1969 Perceptrons, Minsky and Papert

1970 Backpropagation, Linnainmaa

1974 Backpropagation, Werbos

1986 Backpropagation, Rumelhart, Hinton and Williams

1997 LSTM, Hochreiter and Schmidhuber

1998 OCR, LeCun, Bottou, Bengio and Haffner

2006 Deep Learning, Hinton, Osindero, Teh

2009 Imagenet, Deng et al.

2013 Alexnet, LeCun, Bottou, Bengio and Haffner

2015 Resnet (154 layers), MSRA

today GO, Deepmind

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
- Further theoretical improvements
  - 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

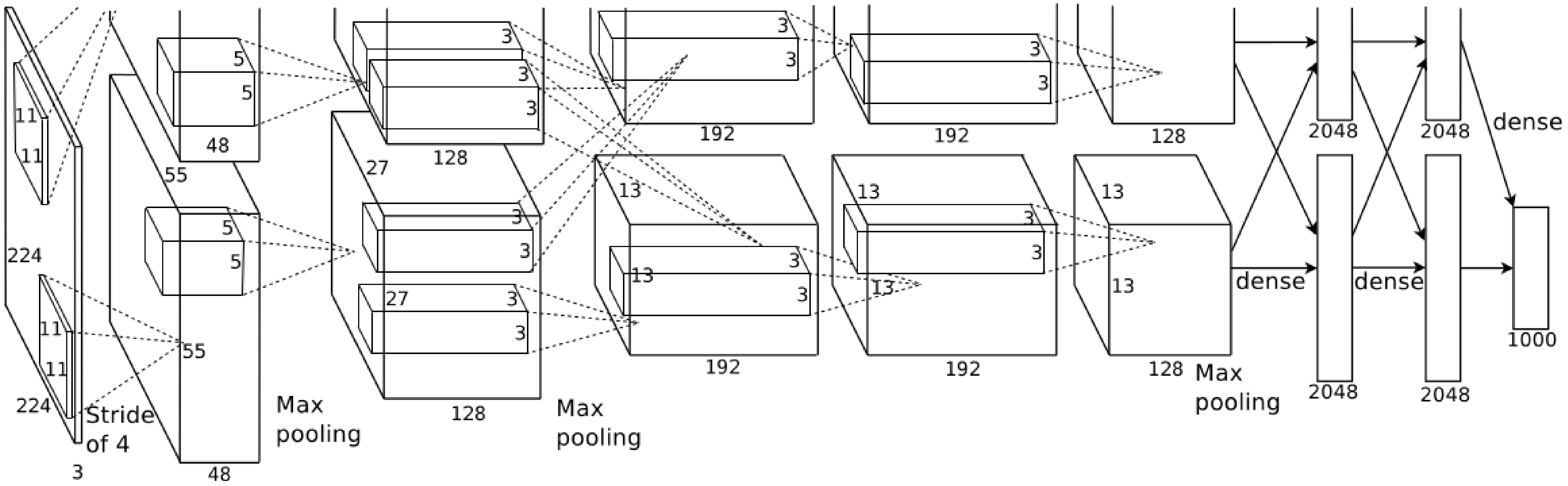
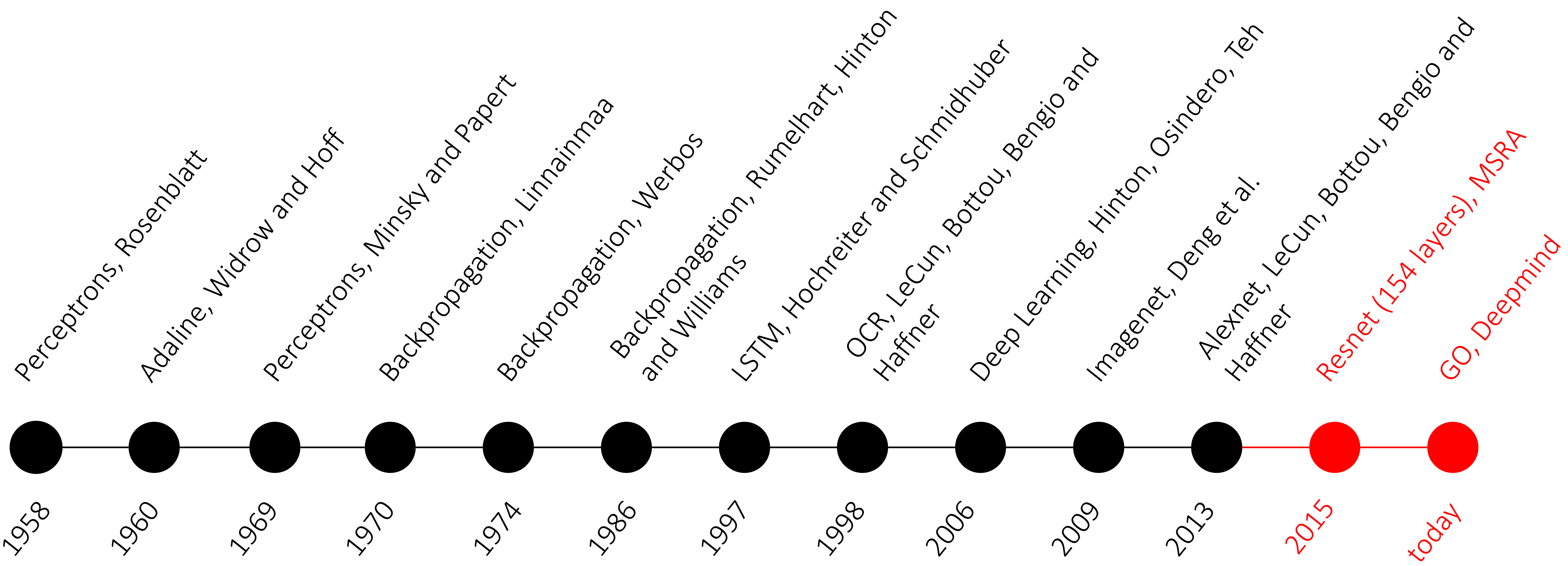


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input 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



1958 Perceptrons, Rosenblatt

1960 Adaline, Widrow and Hoff

1969 Perceptrons, Minsky and Papert

1970 Backpropagation, Linnainmaa

1974 Backpropagation, Werbos

1986 Backpropagation, Rumelhart, Hinton and Williams

1997 LSTM, Hochreiter and Schmidhuber

1998 OCR, LeCun, Bottou, Bengio and Haffner

2006 Deep Learning, Hinton, Osindero, Teh

2009 Imagenet, Deng et al.

2013 Alexnet, LeCun, Bottou, Bengio and Haffner

2015 Resnet (154 layers), MSRA

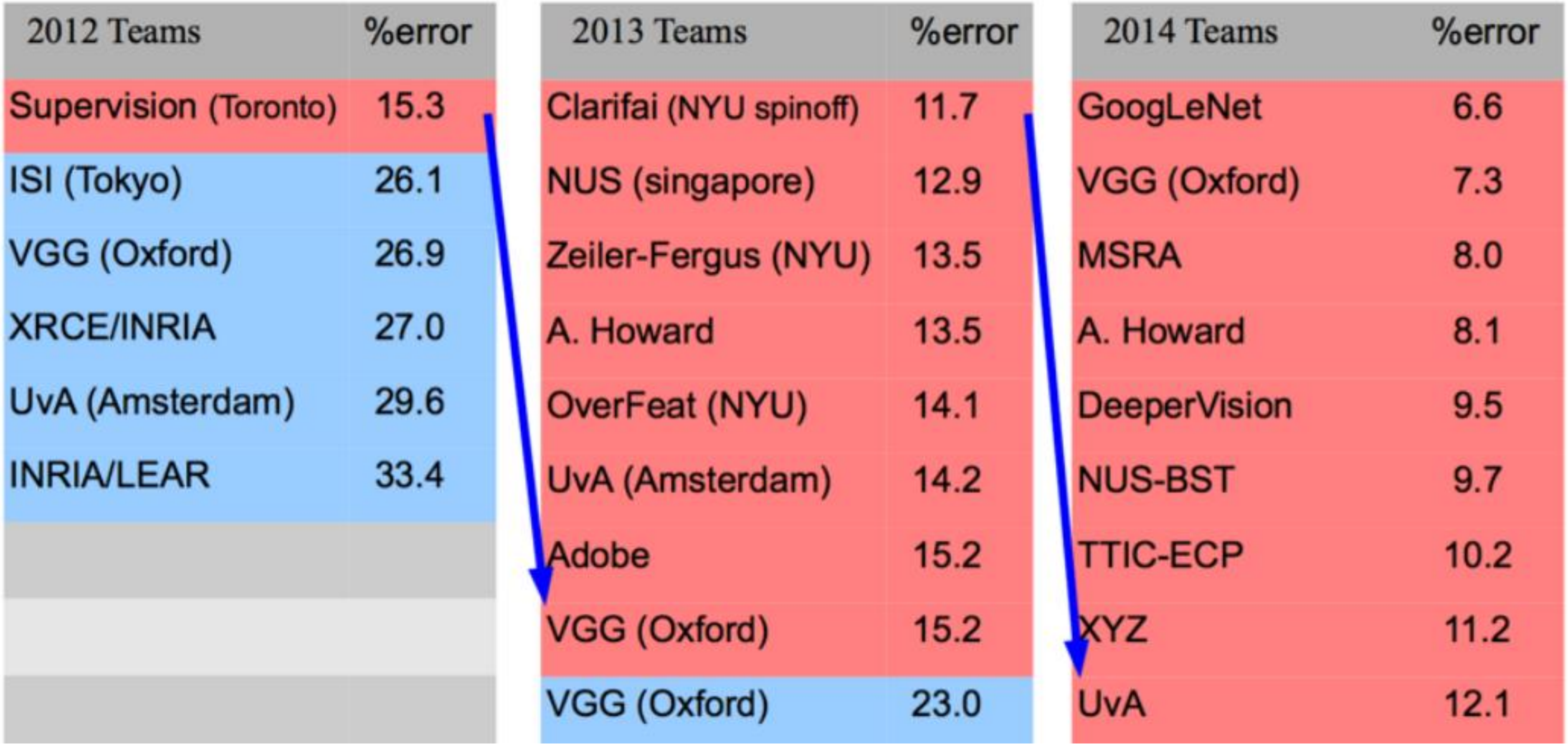
today GO, Deepmind

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

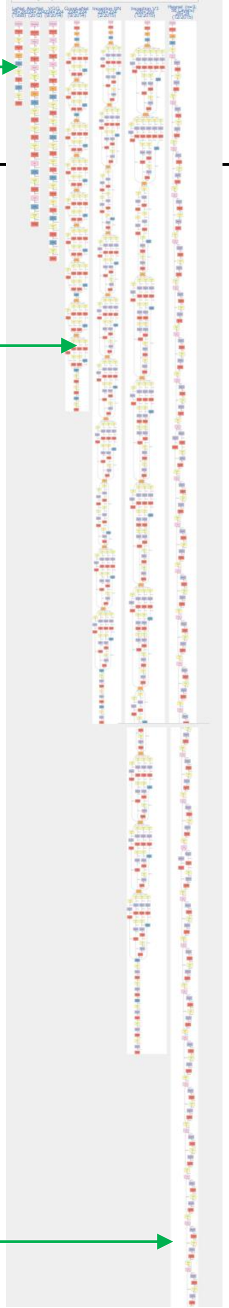
2013 Teams	%error
Clarifai (NYU spinoff)	11.7
NUS (singapore)	12.9
Zeiler-Fergus (NYU)	13.5
A. Howard	13.5
OverFeat (NYU)	14.1
UvA (Amsterdam)	14.2
Adobe	15.2
VGG (Oxford)	15.2
VGG (Oxford)	23.0

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

2015 ILSVRC Challenge

- Microsoft Research Asia won the competition with a legendary 150-layered network
- Almost superhuman accuracy: 3.5% error
  - In 2016 <3% error
- In comparison in 2014 GoogLeNet had 22 layers



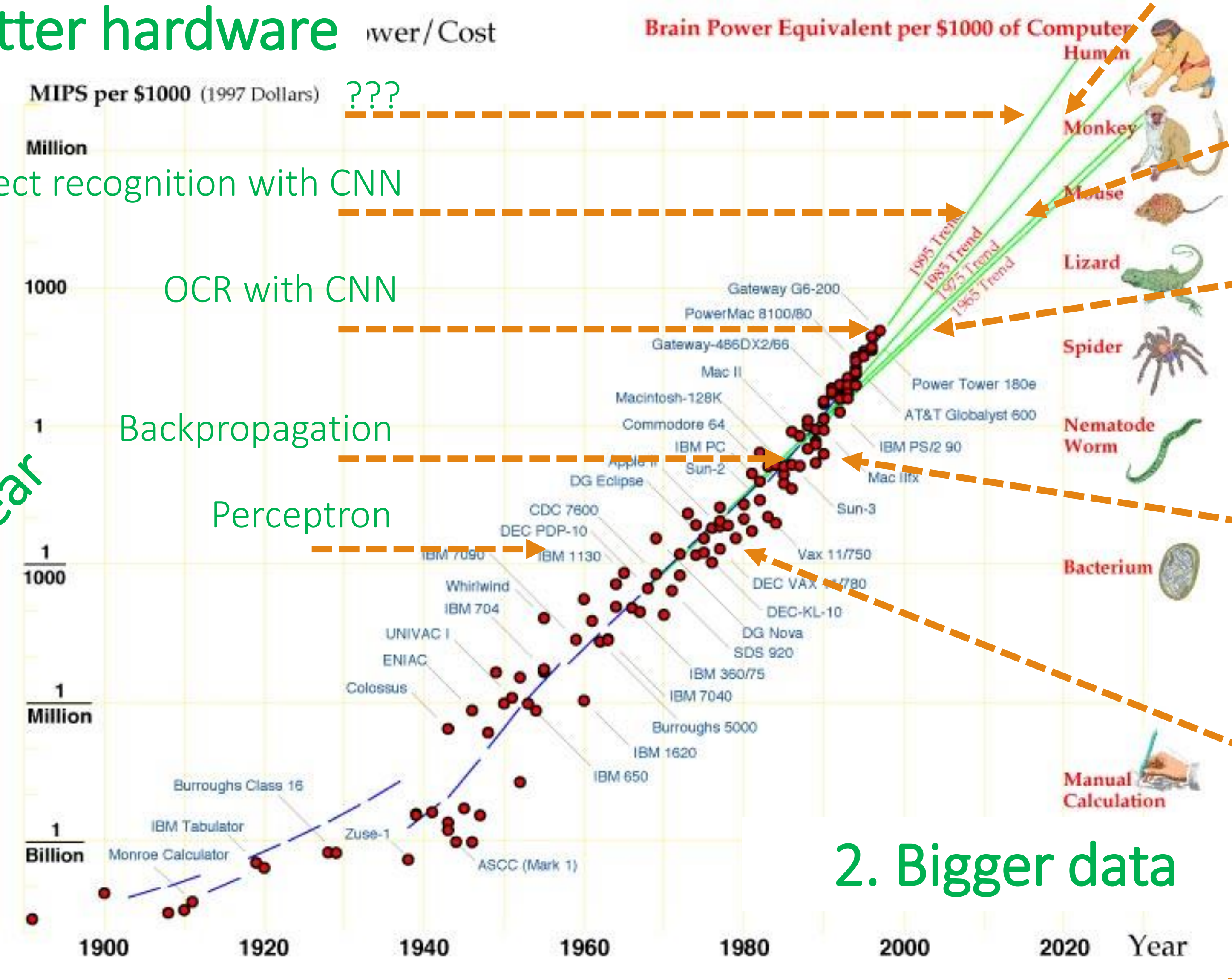
Alexnet, 2012

2014

2015

So, why now?

1. Better hardware



Loglinear

???

Object recognition with CNN

OCR with CNN

Backpropagation

Perceptron

2. Bigger data

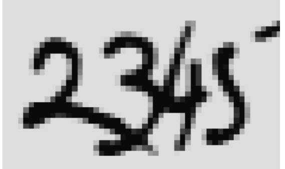
Datasets of everything (captions, question-answering, …), reinforcement learning, ???

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

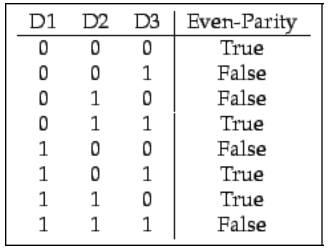
Results:

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

Bank cheques

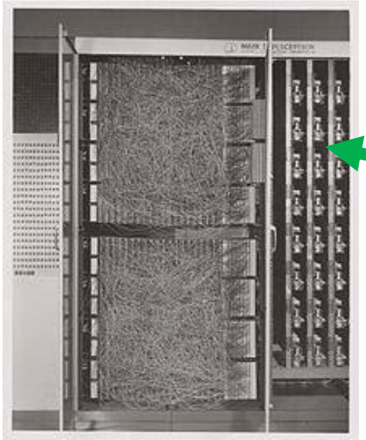


Parity, negation problems



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

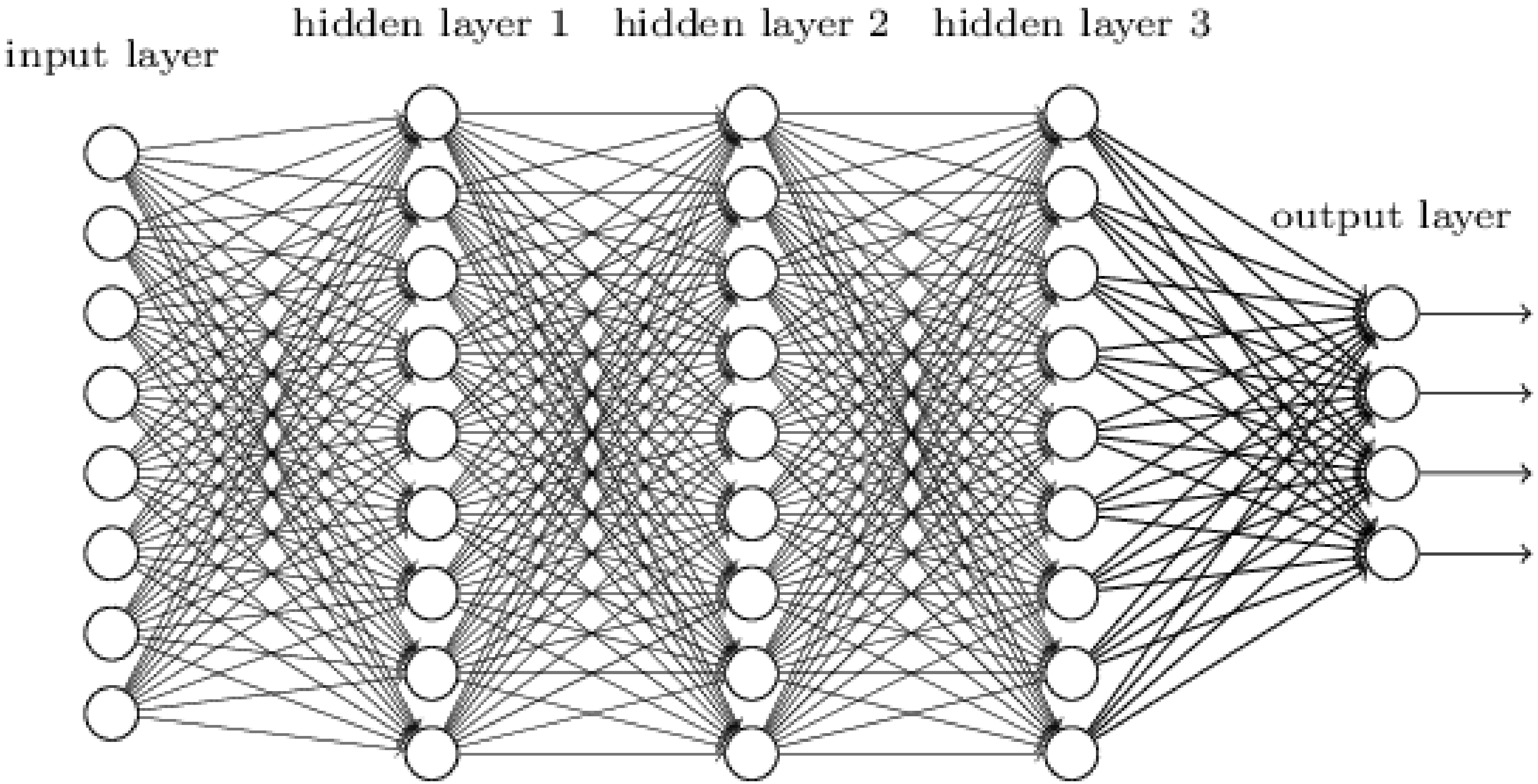


Potentiometers implement perceptron weights

So, why now? (2)

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

Deep Learning: The *What* and *Why*



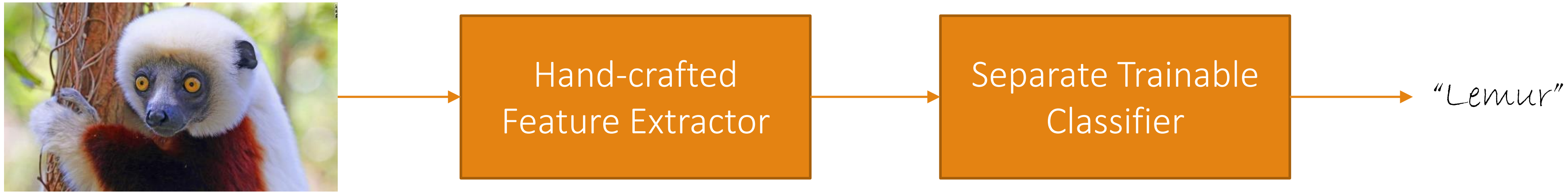
Long story short

- 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.
- $a_L(x; \theta_{1,\ldots,\mathrm{L}}) = h_L\,(h_{L-1}(\ldots h_1(x, \theta_1), \theta_{L-1}),\ \theta_L)$
  - 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

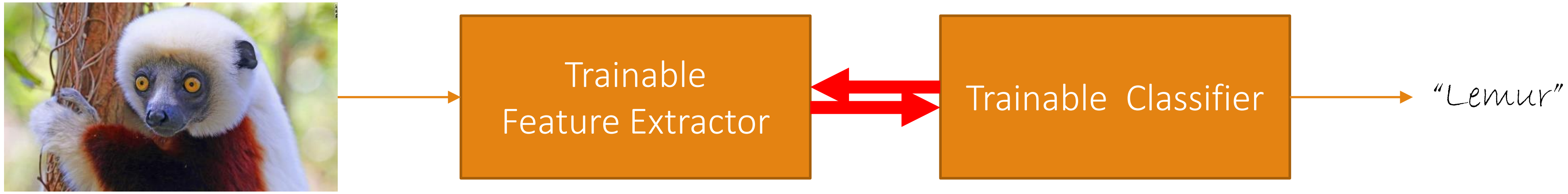
$$\theta^* \leftarrow \arg\min_\theta \sum_{(x,y)\subseteq(X,Y)} \ell(y, a_L(x; \theta_{1,\ldots,\mathrm{L}}))$$

Learning Representations & Features

- Traditional pattern recognition

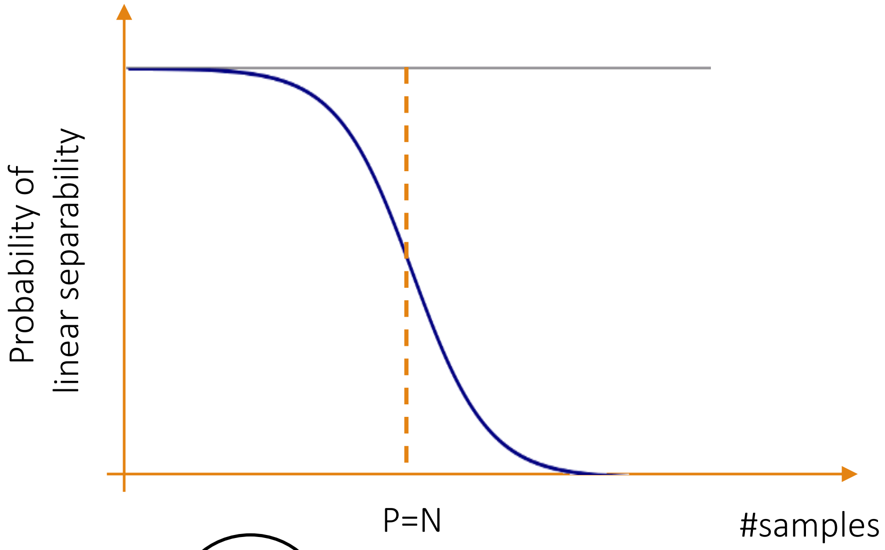


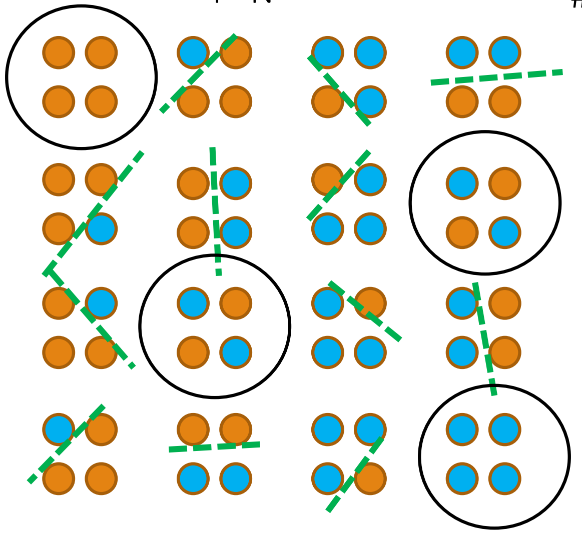
- End-to-end learning → Features are also learned from data



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
- 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-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?
  - Non-linear kernels, e.g., polynomial, RBF, etc
  - Explicit design of features (SIFT, HOG)?

Good features

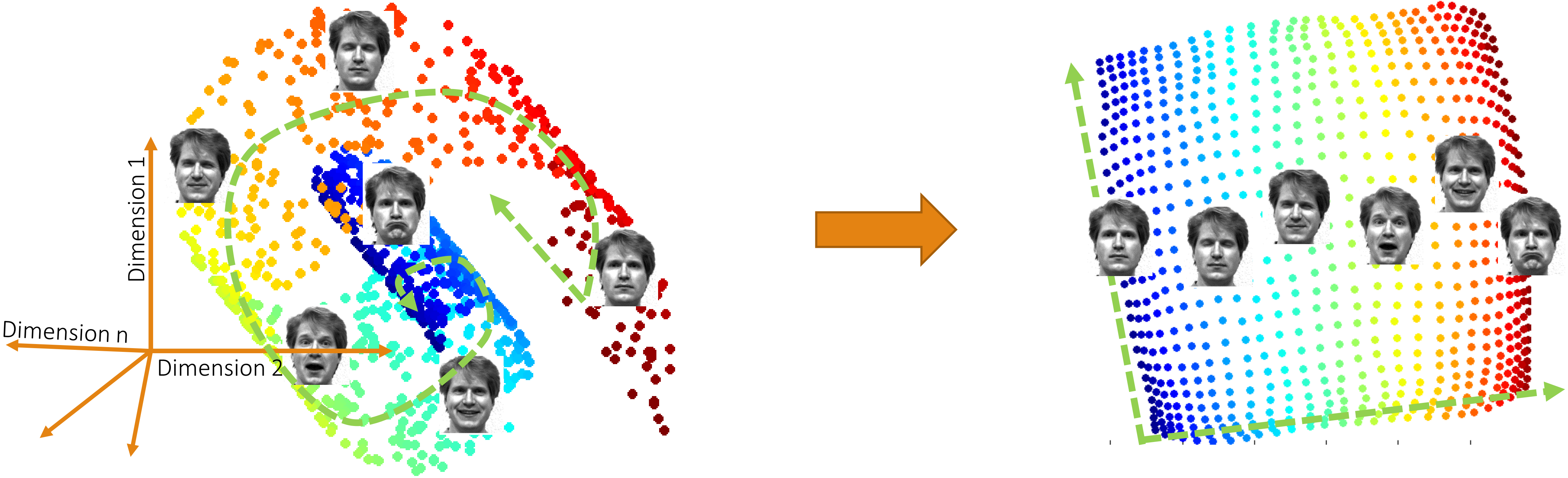
- Invariant
  - But not too invariant
- Repeatable
  - But not bursty
- Discriminative
  - But not too class-specific
- Robust
  - But sensitive enough

How to get good features?

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

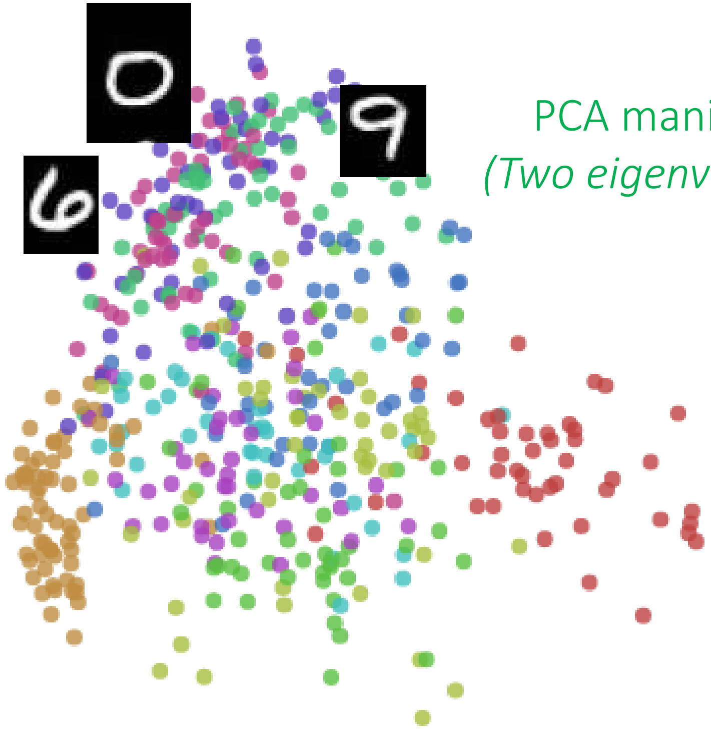
Feature manifold example

- Raw data live in huge dimensionalities
- 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?

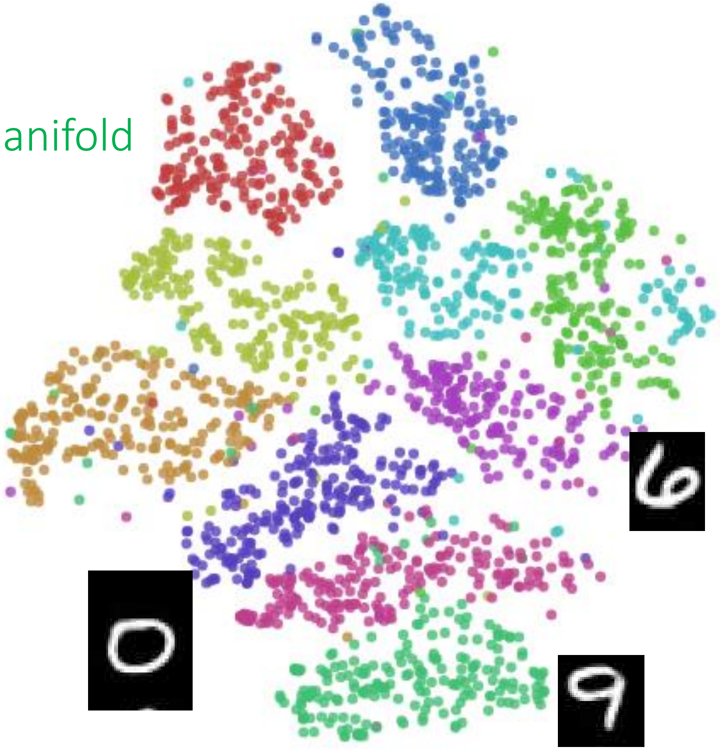


The digits manifolds

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



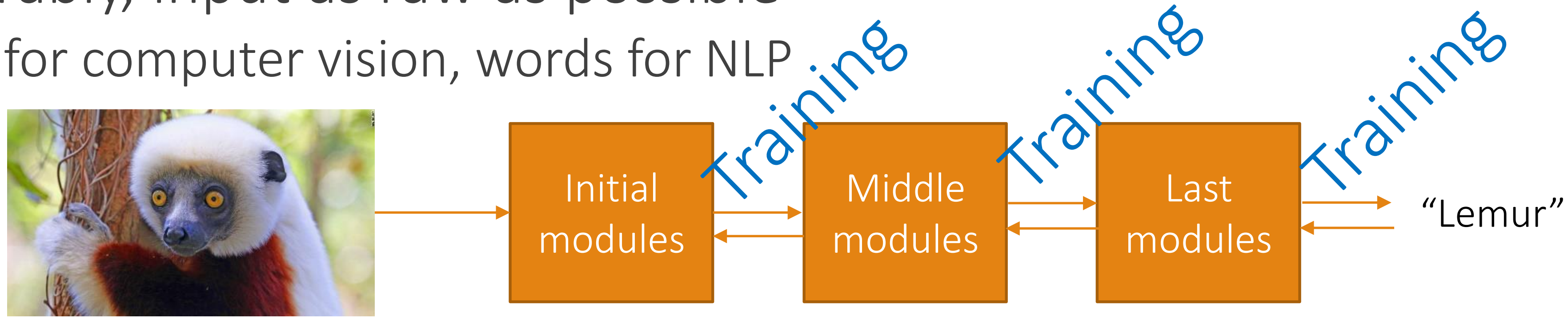
PCA manifold
(Two eigenvectors)



t-SNE manifold

End-to-end learning of feature hierarchies

- A pipeline of successive modules
- 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
- 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 designing features now spent for designing architectures

vs.

Types of learning

- Supervised learning
  - (Convolutional) neural networks

Is this a dog or a cat?

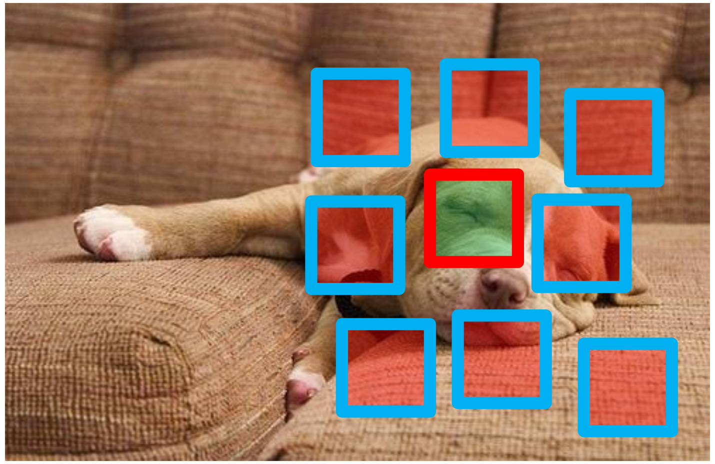
Types of learning

- 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

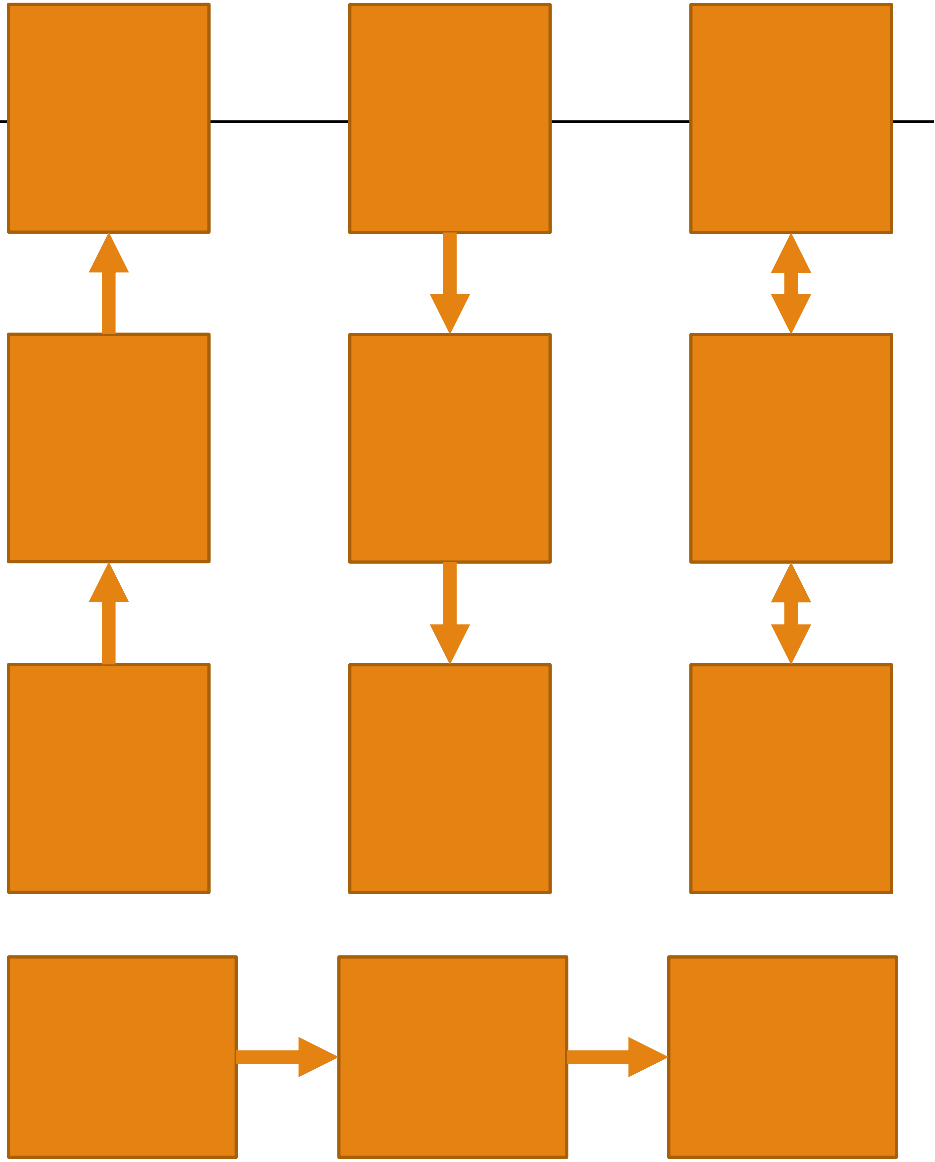


Types of learning

- 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

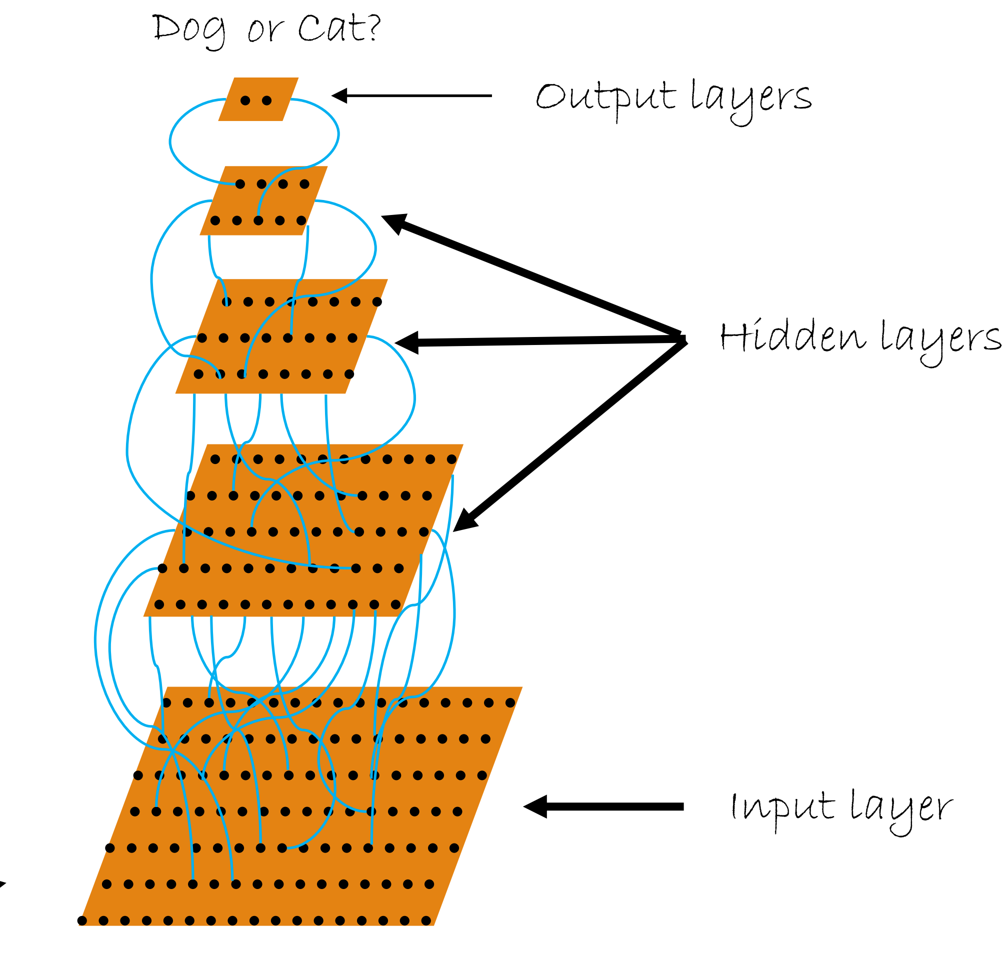
Deep architectures

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

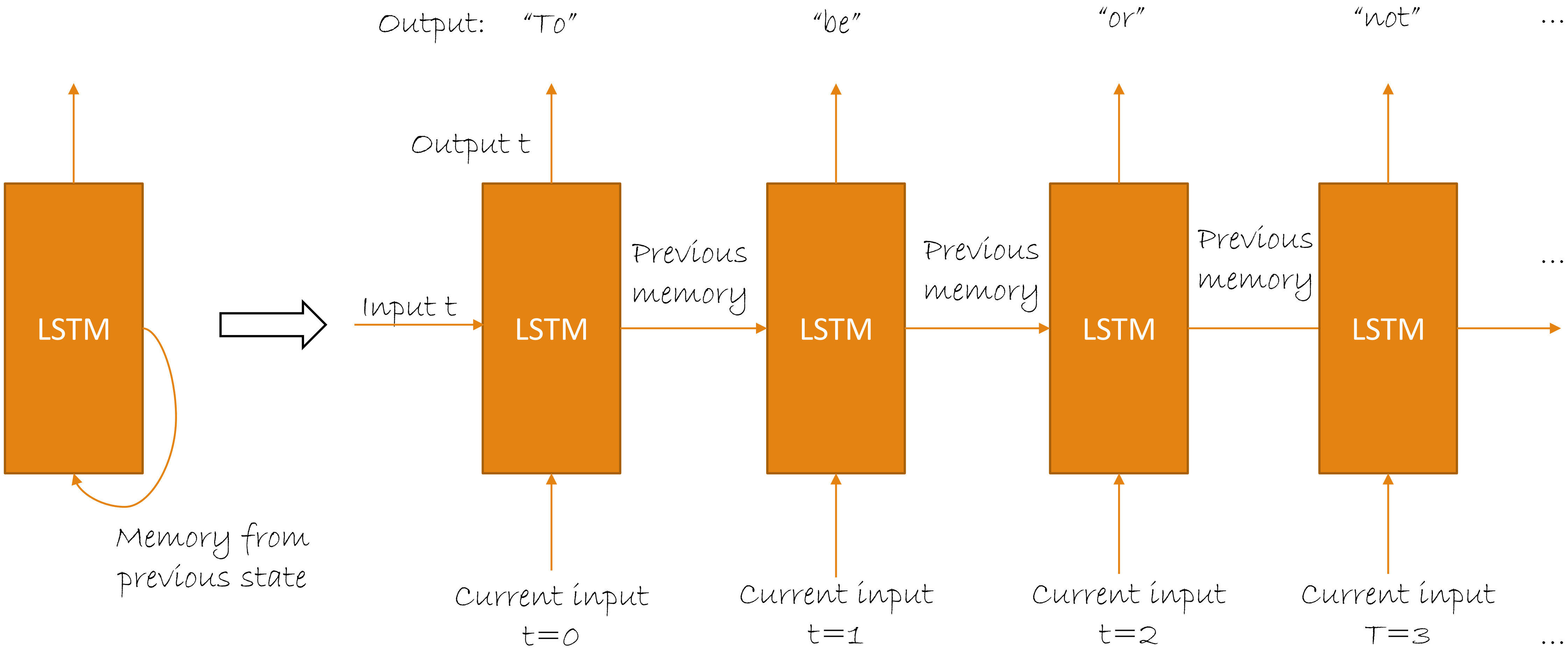


Convolutional networks in a nutshell

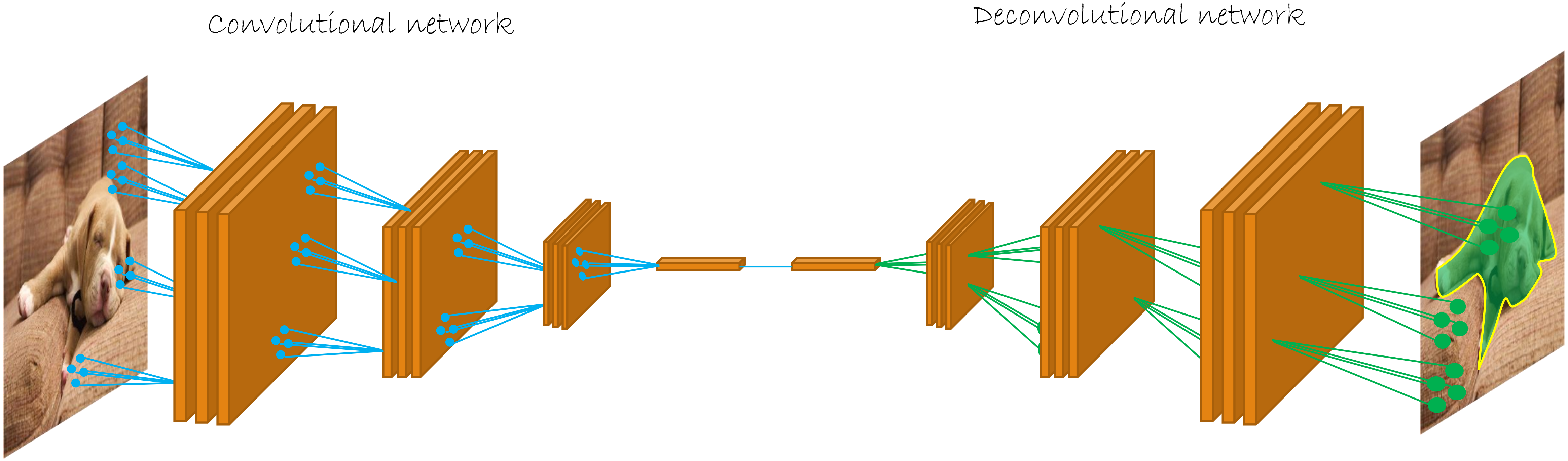
Is this a dog or a cat?



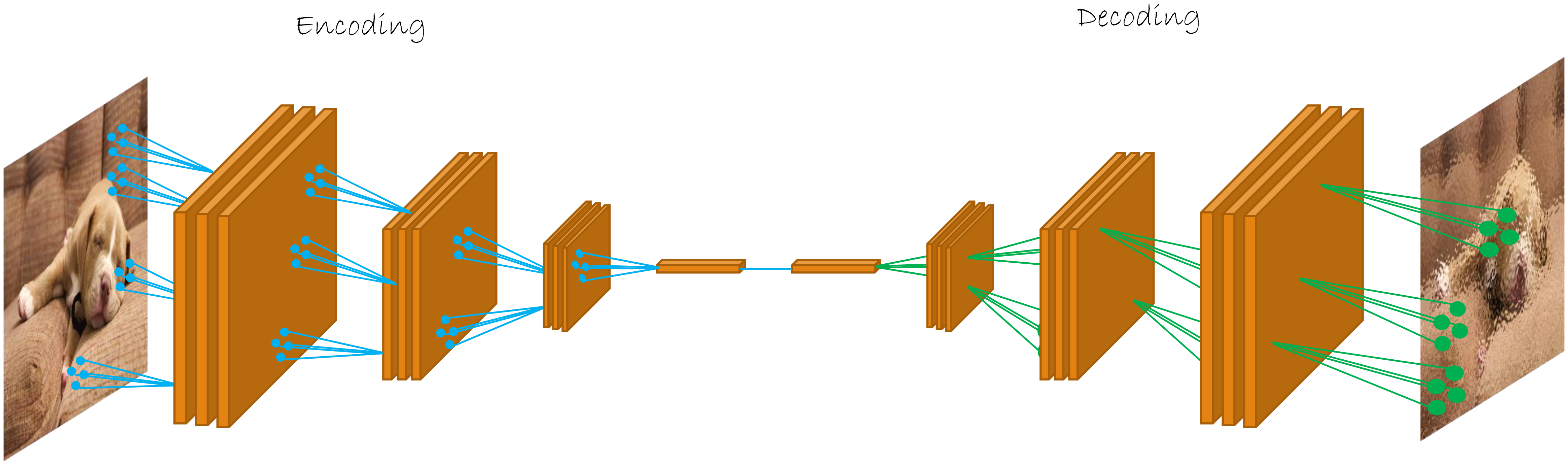
Recurrent networks in a nutshell



Deconvolutional networks



Autoencoders in a nutshell



Philosophy of the course

The bad news ☹

- We only have 2 months = 14 lectures
- Lots of material to cover
- Hence, lots of assignments that you have to implement
  - Basic neural networks, learning Tensorflow, learning to program on a server, advanced optimization techniques, convolutional neural networks, recurrent neural networks, unsupervised learning
- This course is hard

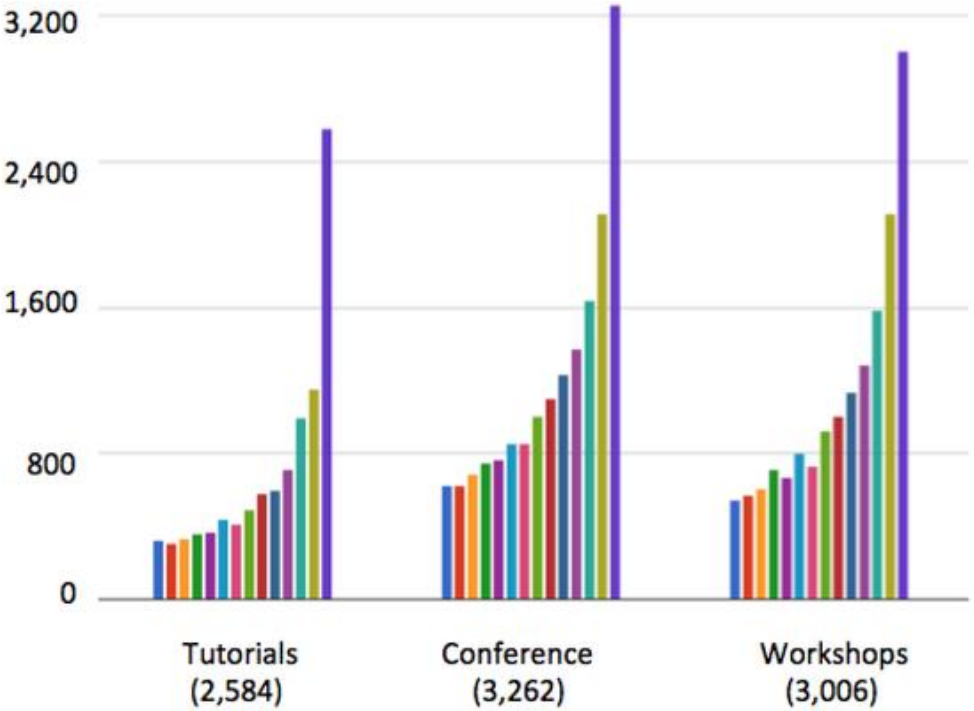
The good news ☺

- We are here to help
  - Kirill, Berkay and Patrick have done some excellent work and we are all ready here to help you with the course
- We have agreed with SURF SARA to give you access to the Dutch Supercomputer Cartesius with a bunch of (very) expensive GPUs
  - You should have no problem with resources
  - You get to know how it is to do real programming on a server
- You'll get to know some of the hottest stuff in AI today

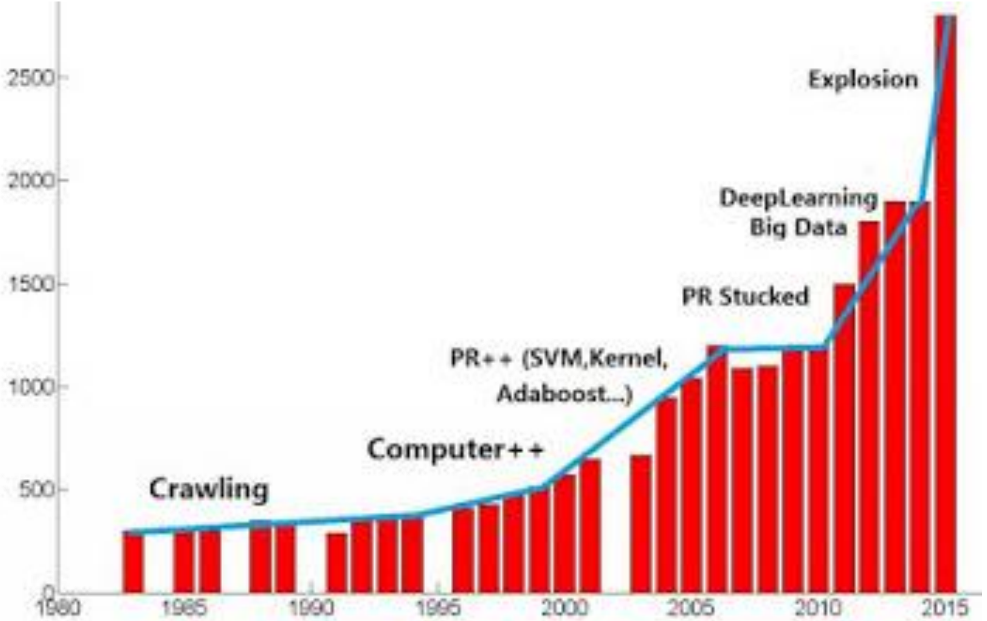
The good news ☺

- You'll get to know some of the hottest stuff in AI today
  - in academia

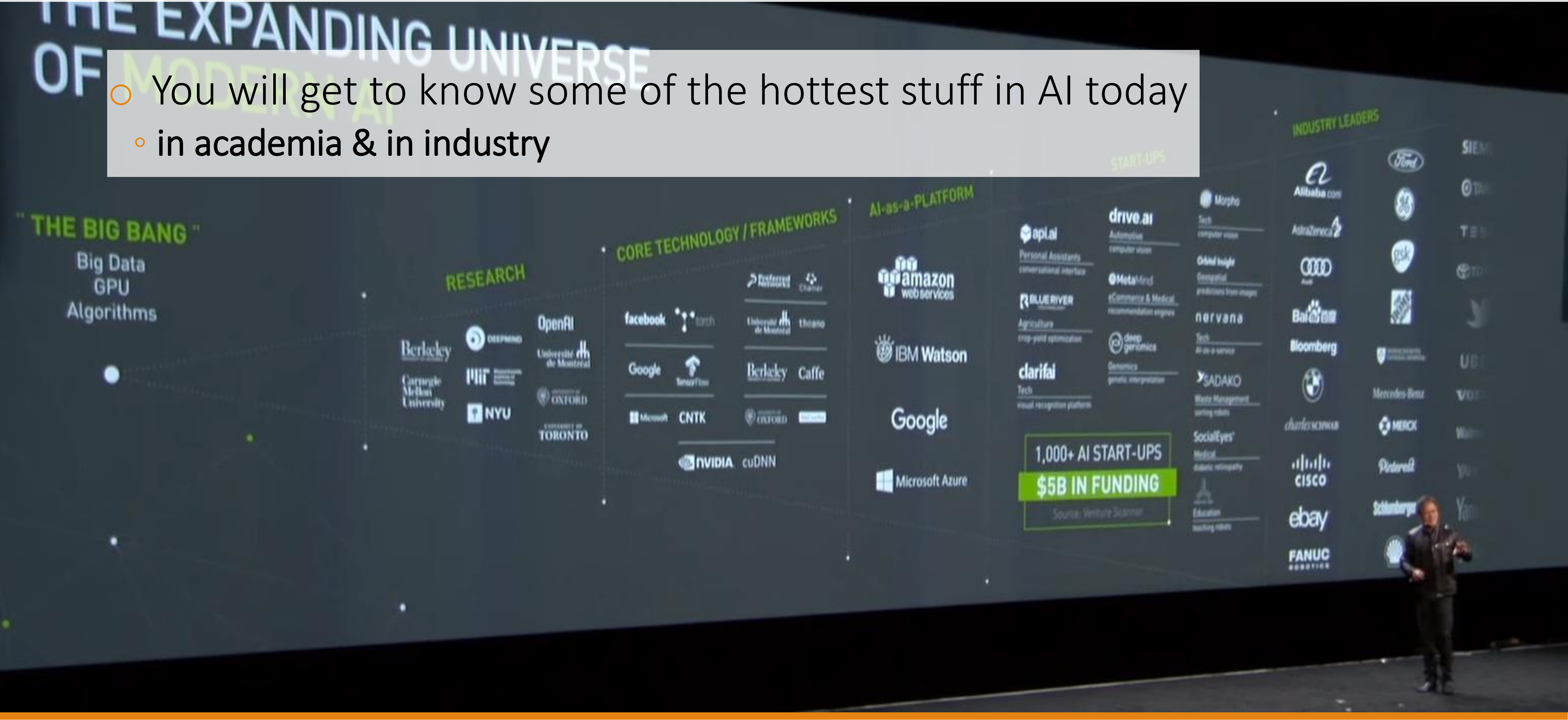
NIPS



CVPR



The good news ☺



- ○ You will get to know some of the hottest stuff in AI today
  - ◦ in academia & in industry

The even better news ☺☺☺

- In the end of the course we will give a few MSc Thesis Projects in collaboration with Qualcomm/QUVA Lab
- Students will become interns in the QUVA lab and get paid during thesis
- Requirements
  - Work hard enough and be motivated
  - Have top performance in the class
  - And interested in working with us
- Come and find me after the course finishes

Code of conduct

- We encourage you to help each other
  - 3 students with **highest participation** in Q&A in Piazza get **+1 grade**
  - Your grade depends on what you do, not what others do
- We encourage you to actively participate, give feedback etc
  - It's only the first real run of the course after all
- However, we do not tolerate **blind** copy
  - Not from each other
  - Not from the internet
  - We have (deep) ways to check that

First lab assignment

Deep Learning Framework

- Tensorflow [https://www.tensorflow.org/]
- Relies on Python
- Very good documentation
- Lots of code
  - You can get inspired but not copy, we have ways to check that

Content & Goal

- 1 hour presentation from SURF SARA on how to use their facilities
- Multi-layer perceptrons
- Solve a neural network in pen and paper
- Basic hyper-parameter tuning
- Your first neural network classifier

Some practical information

- We organized 2 sessions for you, so that you can all comfortably follow the presentation from SURF SARA
- 11.00-13.00
  - SP B0.201
  - Names: A-M
- 13.00-15.00
  - SP D1.115
  - Names N-Z

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

Reading material & references

- http://www.deeplearningbook.org/
  - Chapter 1: Introduction, p.1-p.28

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
- And some theory