Introduction to Deep Learning 2022

Lecturers: Wojtek Kowalczyk
Andrius Bernatavicius

TA’s:

Andrius Bernatavicius (coordinator)
&
5 TA’s (details will follow soon)

Course Objectives

Classical Neural Networks:

- *Basics of Machine Learning and Pattern Recognition (?)*
- Perceptron: Linear Separability, Convergence Theorem, Cover's Theorem
- Multi-layer Perceptron and its expressive power
- Algorithms for training MLPs: Stochastic Gradient Descent and its variants; Backpropagation
- Alternative activation and loss functions; Initialization, Regularization, Dropout, Batch Normalization
- Introduction to Keras, TensorFlow, GPU-computing

Course Objectives

Deep Neural Networks:

- Convolutional Networks; key architectures and applications; Transfer Learning
- Recurrent Neural Networks: Backpropagation through time, LSTM and GRU Networks
- Attention mechanism and Transformers
- Autoencoders
- Generative models: Variational Autoencoders, GAN Networks
- Graph Convolutional Networks
- *Deep Learning for Reinforcement Learning (?)*

Course Organization

Lectures:

Gorlaeus 01, Fridays, 15:15-17:00 or 17:15-19:00

https://www.student.universiteitleiden.nl/en/study--studying/study/educational-information/schedules/science/computer-science-msc?cd=computer-science-msc&cf=science#tab-2

Computer Labs (with own laptops(?)):

Wednesdays, 13:15-15:00

https://www.student.universiteitleiden.nl/en/study--studying/study/educational-information/schedules/science/computer-science-msc?cd=computer-science-msc&cf=science#tab-2

*Presence during lectures & comp. lab is **not compulsory***

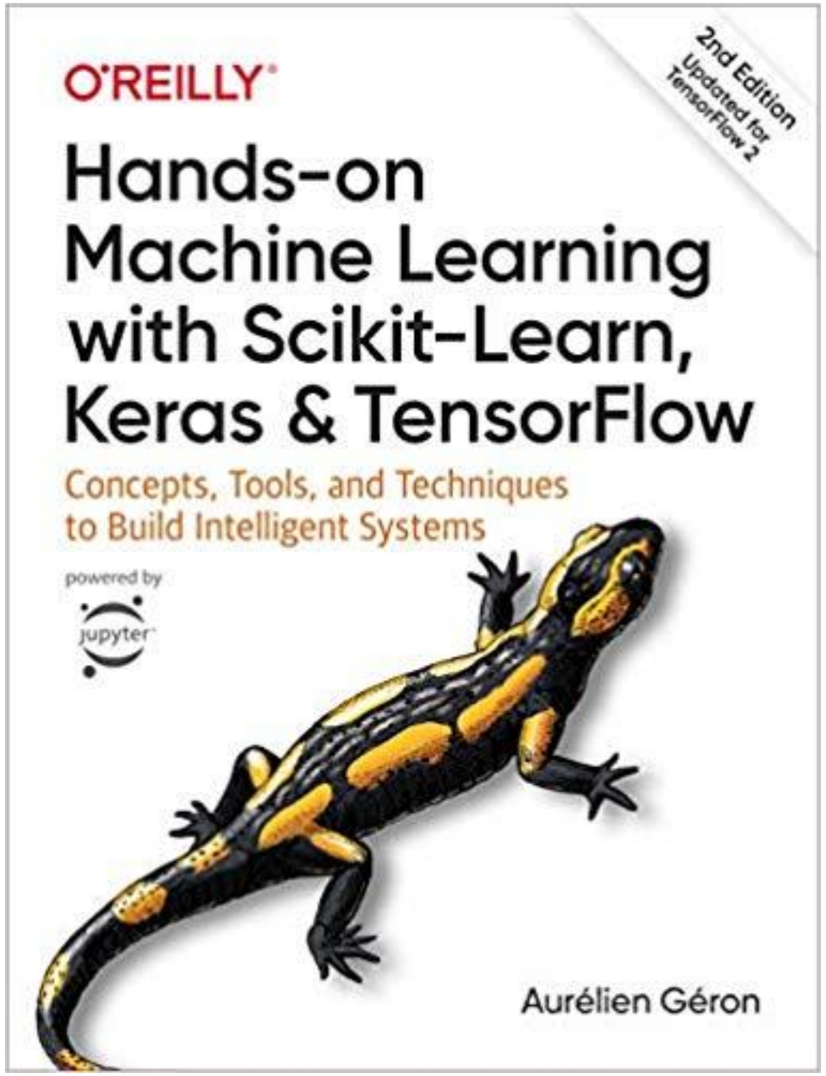
Grading

- Course workload: 6 ects (about 168 hours)

- Final grade is based on two components:
  - Exam (40% of the final grade):
    various questions/small problems related to theory and practice (most likely via the ANS system)
  - Practicals (60% of the final grade): work in 3-persons teams
    two programming & reporting assignments (30%+30%)
  - **<u>Both parts (exam+prac) must be passed </u>(grade > 5.5)!**

Textbook (practical)

3^{d} edition published soon (?)

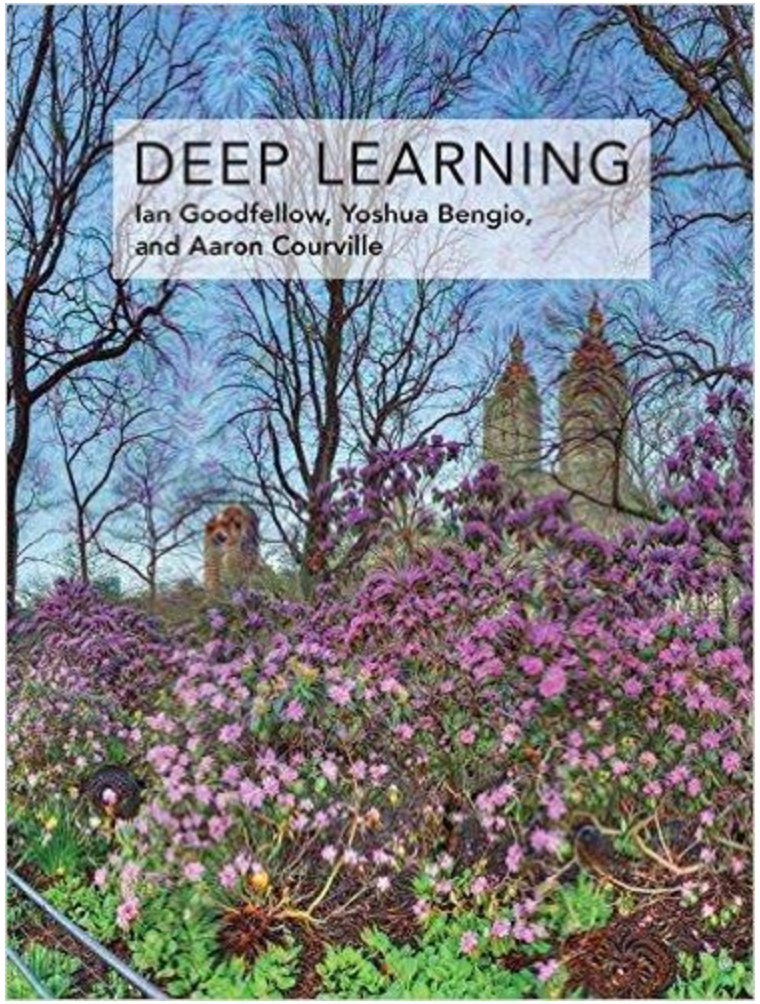

O'REILLY
2nd Edition
Updated for TensorFlow 2
Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow
Concepts, Tools, and Techniques to Build Intelligent Systems
powered by
jupyter
Aurélien Géron

Additional (for interested students)

www.deeplearningbook.org

Available on-line; print it to PDF
or find on the internet a pdf dump;

Highly theoretical …



Additional Internet Resources

http://deeplearning.net (a bit old)

Blogs, Medium/TDS, Github, Neurips.cc, …

Specific links provided during the course

A free experimental Google platform:

colab.research.google.com

Jupyter notebook server powered by CPUs, GPUs or TPUs

Neural Networks:

motivation

history

context

How could we program computers to do:

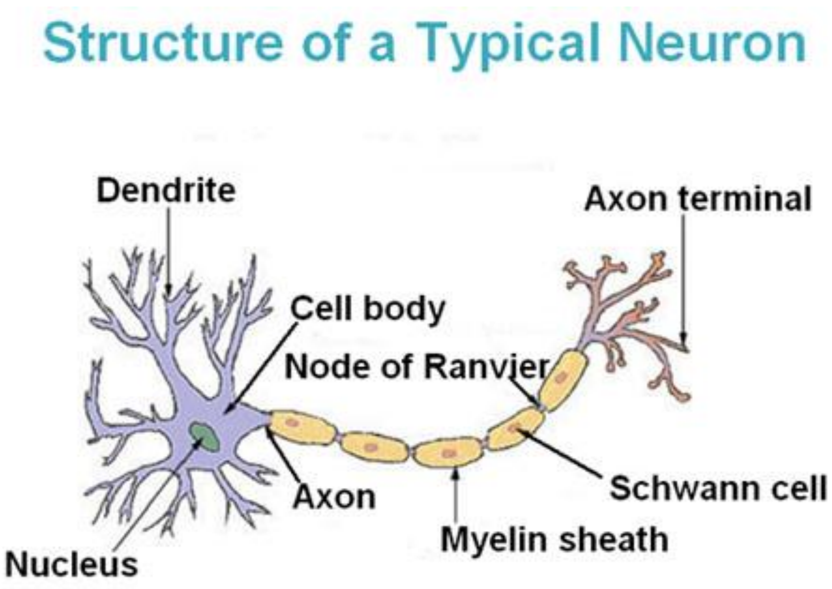
- Optical Character Recognition
- Text2Speech translation
- Speech2Text translation
- Language2Language translation
- Robot arm control
- Face recognition
- Driving a car
- Detecting fraud with credit cards
- Playing arcade games, poker, chess, go, ...

How people do it? “training data” + “self-learning”

The biological model

Human brain

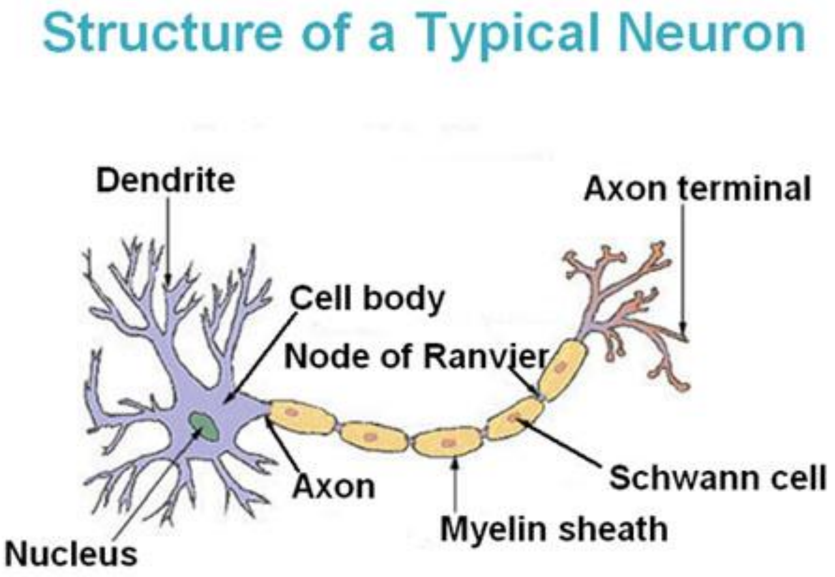
around 10^{11} neurons, and 10^{15} interconnections (synapses)



The biological model

How a neuron works ?

- It receives input signals through its dendrites
- The input signals generate difference of electrical potential on the cell membrane
- The difference is propagated to the axon hillock
- A train of electrical impulses is generated along the axon
- The impulses on the axon generate the release in the synaptic space of some neurotransmitters



Learning by changing the topology & thickness of connections

Basic principle

Artificial neural network (ANN)= set of highly interconnected simple processing units (also called neurons)

Processing unit = simplified model of a neuron

Learning = finding the architecture of connections
and their weights

Computer versus Brain

Von Neumann machine

•One or a few high speed (ns) processors with considerable computing power

•One or a few shared high speed buses for communication

•Sequential memory access by address

•Problem-solving knowledge is separated from the computing component
•Hard to be adaptive

Human Brain

- Large number (10^{11}) of low speed processors (ms) with limited computing power
- Large number (10^{15}) of low speed connections
- Content addressable recall (CAM)
- Problem-solving knowledge resides in the connectivity of neurons
- ***Adaptation by changing the connectivity (topology & thickness of connections)***

Human Brain

- **graceful degradation**:
  its performance degrades gracefully under partial damage
- **self-learning:**
  it can learn (reorganize itself) from experience
- **self-healing:**
  partial recovery from damage is possible
- **massive parallelism:**
  performs parallel computations extremely efficiently. E.g., complex visual perception takes less than 100 ms, that is, 10 processing steps!
- **intelligence** and **self-awareness** (whatever it means)

ANN History: Roots …

Roots of Neural Networks are in:

Neurobiological studies (more than one century ago):

- How do nerves behave when stimulated by different magnitudes of electric current? Is there a minimal threshold needed for nerves to be activated? Given that no single nerve cell is long enough, how do different nerve cells communicate among each other?

Psychological studies:

- How do animals learn, forget, recognize and perform other types of tasks?

Psycho-physical experiments

- Understand how individual neurons and groups of neurons work.

ANN History: First steps

- **Pitts & McCulloch (1943)**
  - First mathematical model of biological neurons
  - All Boolean operations can be implemented by these neuron-like nodes (with different threshold and excitatory/inhibitory connections).
  - Competitor to Von Neumann model for general purpose computing device
  - Origin of automata theory.
- **Hebb (1949)**
  - Hebbian rule of learning: increase the connection strength between neurons ***i*** and ***j*** whenever both ***i*** and ***j*** are activated.
  - Or increase the connection strength between nodes ***i*** and ***j*** whenever both nodes are simultaneously ON or OFF.

ANN History: Perceptron

- **Early booming (50's – early 60's)**
  - Rosenblatt (1958)
    - Perceptron: network of threshold nodes for pattern classification. Perceptron learning rule – first learning algorithm
    - Perceptron convergence theorem:

      "everything that can be represented by a perceptron can be learned"
  - Widrow and Hoff (1960, 1962)
    - Learning rule that is based on minimization methods
  - Minsky's attempt to build a general purpose machine with Pitts/McCullock units

ANN History: Setback …

- **The setback (mid 60’s – late 70’s)**
  - Minsky and Papert publish a book "Perceptrons" (1969):
    - Single layer perceptron cannot represent (learn) simple functions such as XOR
    - Multi-layer of non-linear units may have greater power but there was no learning algorithm for such nets
    - Scaling problem: connection weights may grow infinitely
  - ***US Defense/Government stop funding research on ANN***

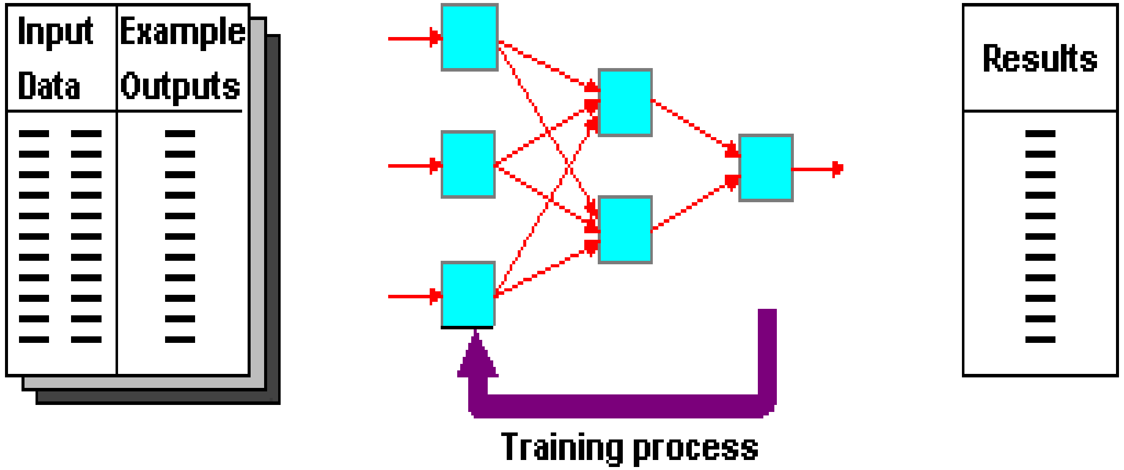
ANN History: Backpropagation

- **Renewed enthusiasm and progress (80’s – 90's)**
  - New techniques
    - Backpropagation learning for multi-layer feed forward nets (with non-linear, differentiable node functions)
    - Physics inspired models (Hopfield net, Boltzmann machine, etc.)
    - Unsupervised learning (LVQ nets, Kohonen nets)
  - Impressive applications (character recognition, speech recognition, text-to-speech transformation, process control, associative memory, etc.)

But:

- Criticism from Statisticians, Neurologists, Biologists, Ordinary Users, ...
- Lots of ad-hoc solutions, "wild creativity"
- A lot of rubbish is produced ...

Supervised Learning



- Training data: inputs & targets (= desired outputs)

- Network: a complicated function with many parameters to tune

- Training: a process of tweaking the parameters to minimize the errors of predictions (outputs should be close to targets)

$$Error = \sum_{Examples} (\text{output}_i - \text{target}_i)^2$$

- Generalization: we want the network to perform well on data that was not used during the training process!

Example: Recognizing Handwritten Digits

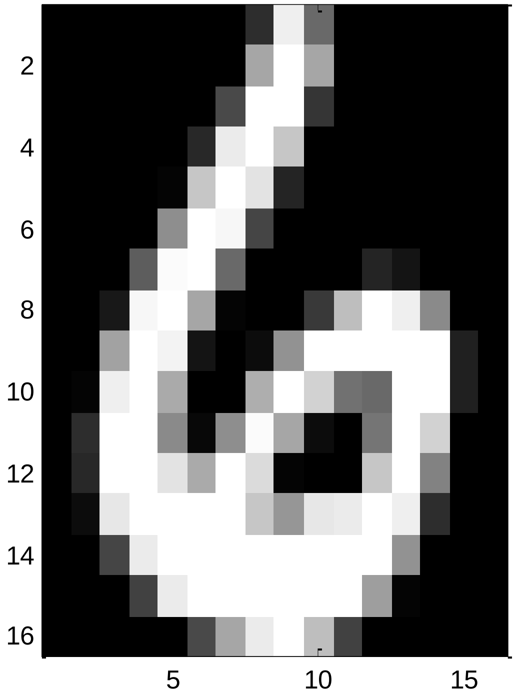
Training set: 2000 images of digits

Each image: 16x16 pixels = a vector of 256 components

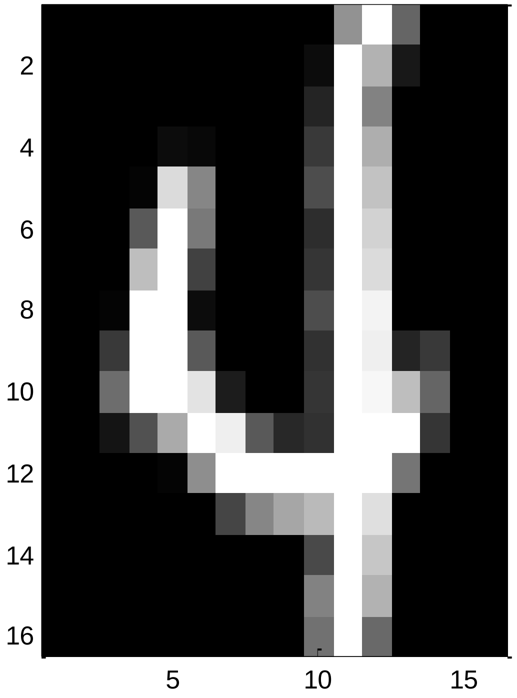
The network: 256 inputs and 10 outputs

Test set: 1000 images of digits

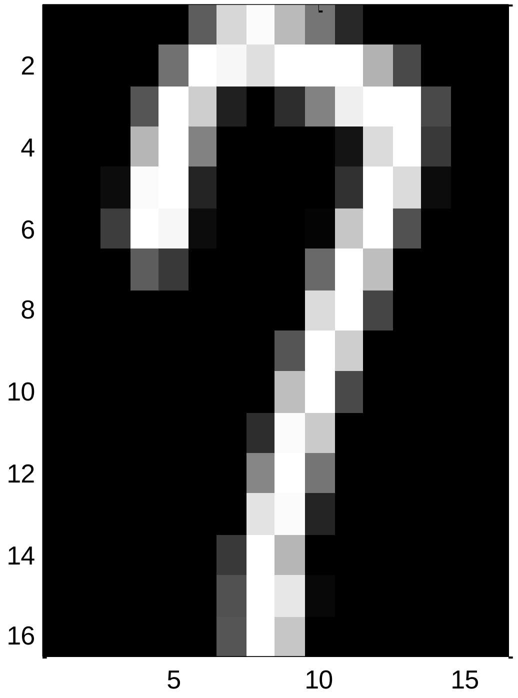
Digit: 6



Digit: 4



Digit: 7



Backpropagation algorithm

- The backpropagation algorithm searches for weight values that **minimize the total error of the network**

- It consists of the **repeated application of two passes**:

  - **Forward pass**: in this step the network is activated on one example and the **error** of each neuron of the output layer is **computed**.

  - **Backward pass**: in this step the network error is used for updating the weights. Starting at the output layer, **the error is propagated backwards through the network, layer by layer,** with help of the **generalized delta rule**. Finally, all weights are updated.

- No guarantees of convergence
  (when learning rate too big or too small)

- In case of convergence: local (or global) minimum

- In practice: try several starting configurations and learning rates.

Three examples of MLP

NetTalk: a network that reads aloud texts

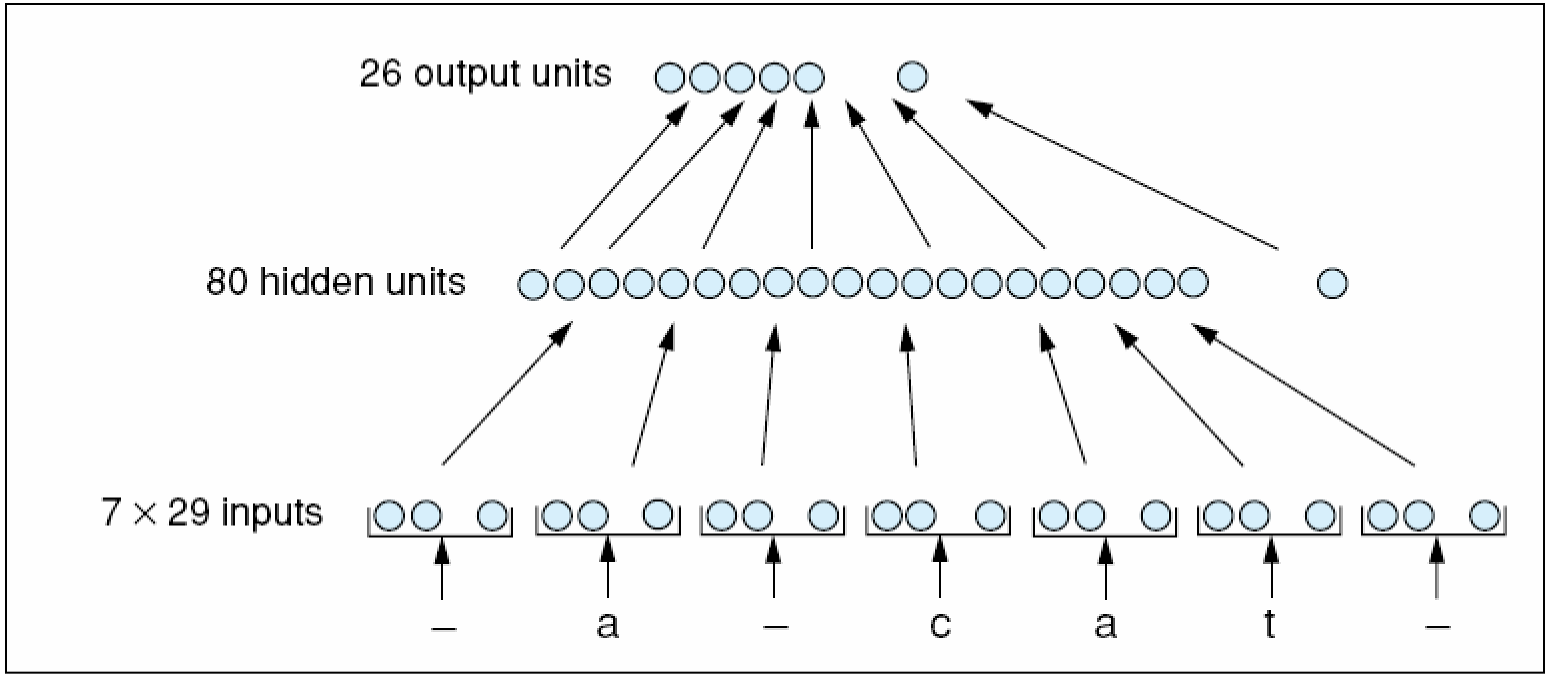
ALVINN: a Neural network that drives a car

Falcon: a real-time system for detecting fraud with credit card transactions

NETtalk (Sejnowski & Rosenberg, 1987)

- The task is to learn to pronounce English text from examples (text-to-speech)
- Training data: a list of <phrase, phonetic representation>
- Input: 7 consecutive characters from written text presented in a moving window that scans text
- Output: phoneme code giving the pronunciation of the letter at the center of the input window
- Network topology: 7x29 binary inputs (26 chars + punctuation marks), 80 hidden units and 26 output units (phoneme code). Sigmoid units in hidden and output layer

NETtalk



NETtalk (contd.)

- Training protocol: 95% accuracy on training set after 50 epochs of training by full gradient descent.
  78% accuracy on a test set

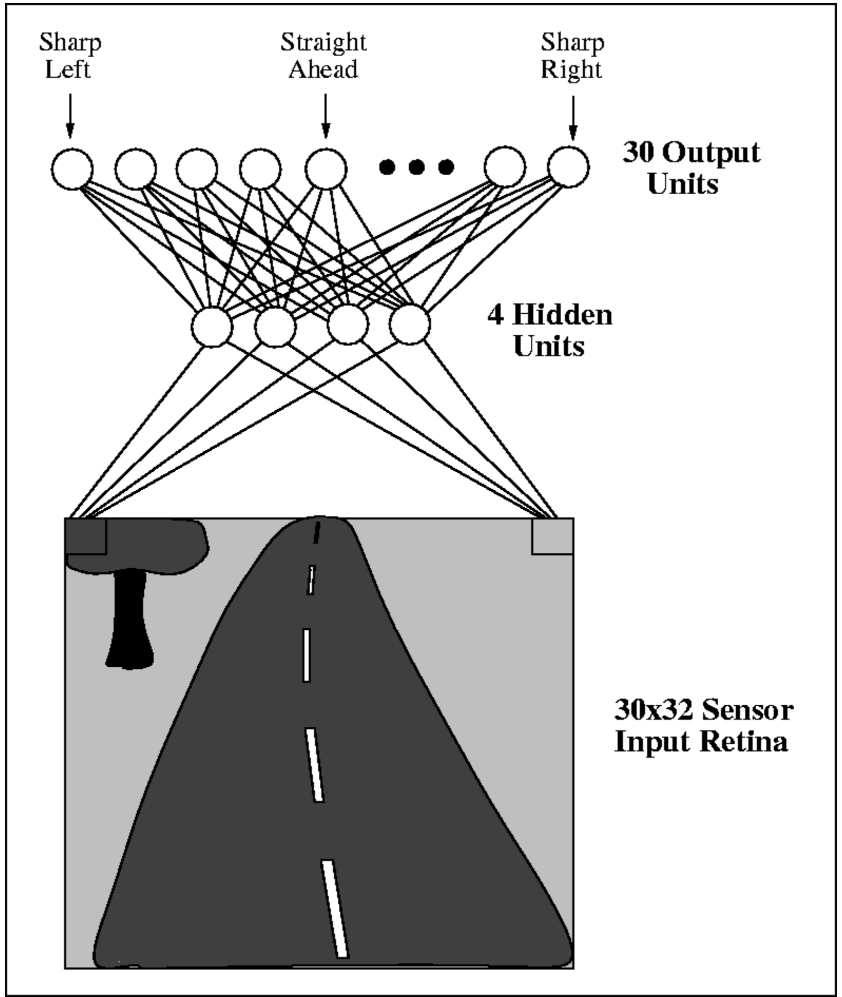
- DEC-talk: a rule-based system crafted by experts
  (a decade of efforts by many linguists)

- Functionality/Accuracy almost the same

- Try: http://cnl.salk.edu/Media/nettalk.mp3

ALVINN (1989)

D.A. Pomerleau, Autonomous Land Vehicle In a Neural Network (NIPS '89)



FALCON: A Fraud Detection System

http://www.fico.com/en/Products/DMApps/Pages/FICO-Falcon-Fraud-Manager.aspx

Right now, leading institutions around the world trust Fair Isaac's Falcon™ Fraud Manager to protect more than 450 million active credit and debit cards. Why? They know they'll receive regular technological advances which will allow them to stay a step ahead of new and emerging fraud types. Protecting 65% of the world's credit card transactions, Falcon detects fraud with pinpoint accuracy via **proven neural network models** and other proprietary predictive technologies. Debit, credit, oil and retail card issuers in numerous marketplaces rely on Falcon to detect and stop fraudulent transactions - and combat identity theft - in real time.

Why only “shallow networks” ?

- In practice, "classical" multi-layer networks work fine only for a very small number of hidden layers (typically 1 or 2) - this is (*was*) an empirical fact ...

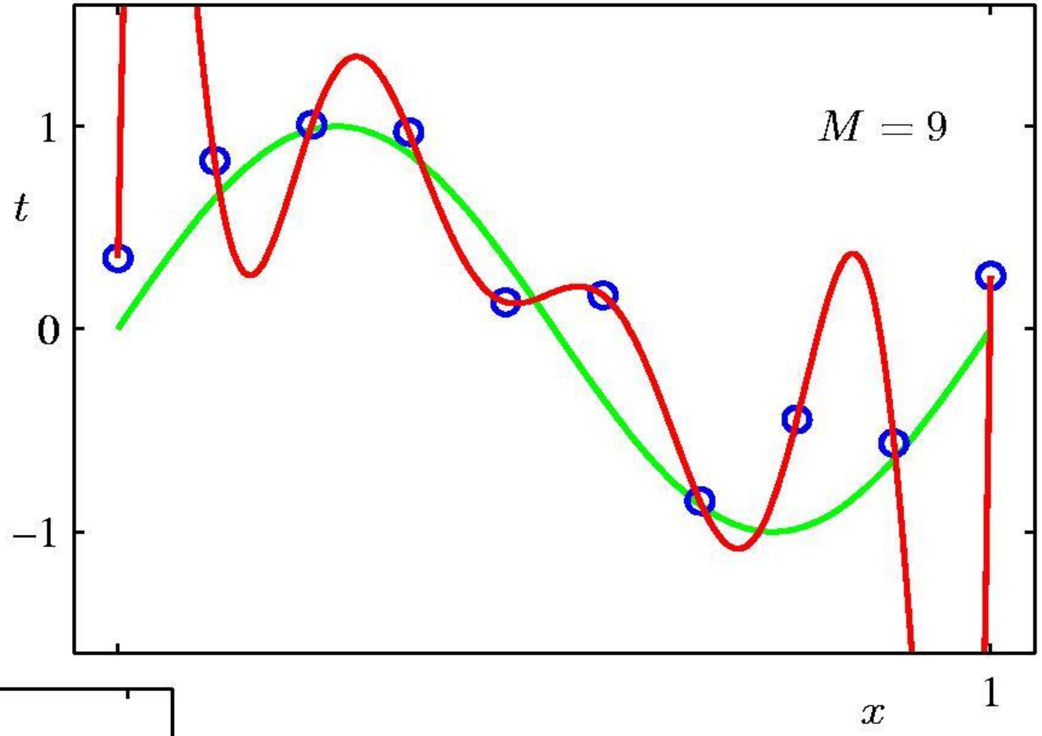
- Adding layers is harmful because:
  - the increased number of weights quickly leads to data overfitting (lack of generalization)

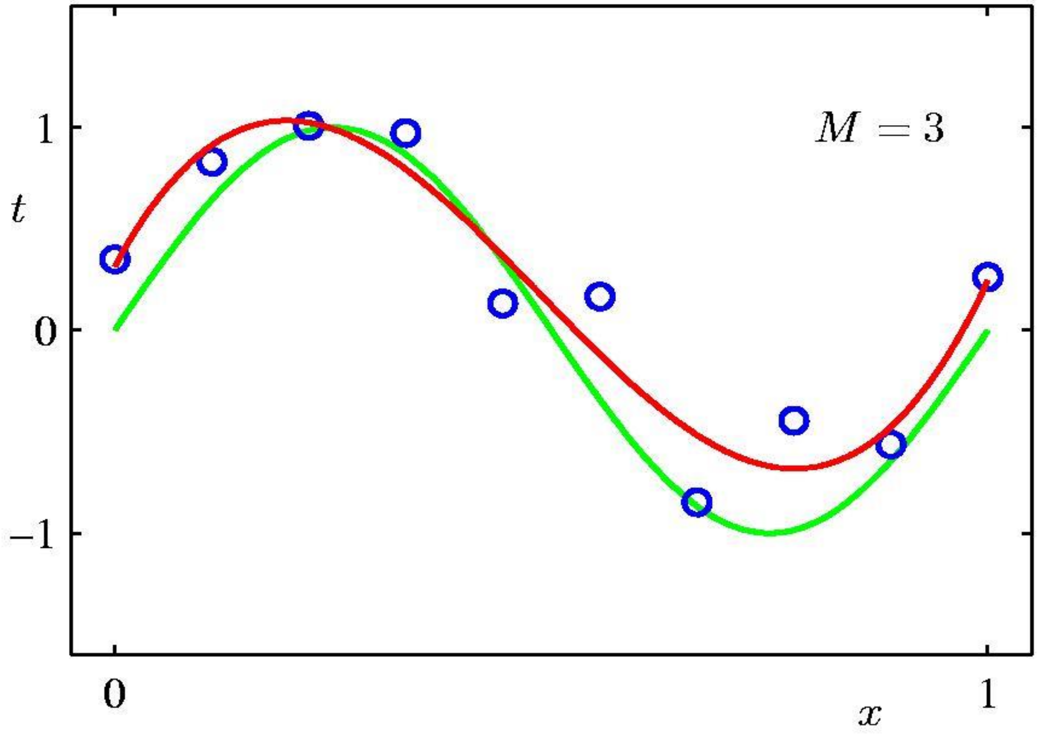
  - huge number of (bad) local minima trap the gradient descent algorithm

  - vanishing or exploding gradients (the update rule involves products of many numbers) cause additional problems

Overfitting:

which polynomial better approximates the green line?





The more parameters the higher the risk of overfitting !

Why do we need many layers?

- In theory, one hidden layer is sufficient to model any function with an arbitrary accuracy; however, the number of required nodes and weights grows exponentially fast

- The deeper the network the less nodes are required to model "complicated" functions

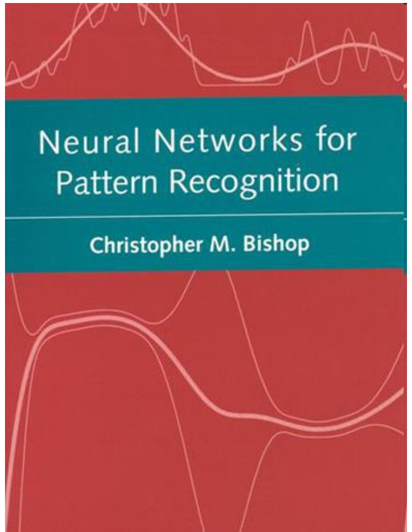
- Consecutive layers "learn" features of the training patterns; from simplest (lower layers) to more complicated (top layers)

- Visual cortex consists of about 10 layers

An end of hype?

Dominance of new techniques (90's - 2005):

- Support Vector Machines (Vapnik)
- Kernel Methods (Vapnik, Scholkopf, ...)
- Ensemble methods (Breiman, Hasti, Tibshirani, Friedman,...)
- Random Forests
- ***Neural Networks lose their prominent role in the fields of Pattern Recognition and Machine Learning***
- ***since 1996 no new textbooks on Neural Networks !***



Deep Learning Revolution

~ 2006 : G. Hinton's group: new ideas, inventions, applications

- Restricted Boltzmann Machine as a building block for DNN
- Contrastive Divergence algorithm
- Combine supervised with unsupervised learning
- Deep Belief Networks, Stacked Auto-Encoders

2010 : AlexNet (ImageNet Challenge): ***start of the DL-revolution!***

2013: DeepMind "solves" the Atari 2600 games: -> Reinforcement Learning

2016-: AlphaGo, AlphaGoZero, AlphaZero (Go, Chess, …?)

Many spectacular applications beating existing approaches

1989 (!) : LeCun et al: A Convolutional Network for OCR (AT&T) !???

2010-… : deeper networks, complicated architectures, more applications, …

Enabling Factors

- Availability of “Big Data”: the more data we have the better we can train a network

- Powerful Hardware (GPU’s): speedup of the training process by 100-1000 times reduces the training time from years to hours

- New algorithms and architectures: leaving the MLP standard behind …

- ***Many spectacular successes!***

Key Deep Learning Architectures

- Convolutional Networks: when adding layers enforce hidden nodes to learn "local features" - that reduces the number of parameters

- Recurrent Networks, Transformers: networks for modeling sequential data (text, speech, video, 3D-live brain scans, …)

- Autoencoders: networks that learn intrinsic features of the data

- VAEs, GANs: generative models of the data

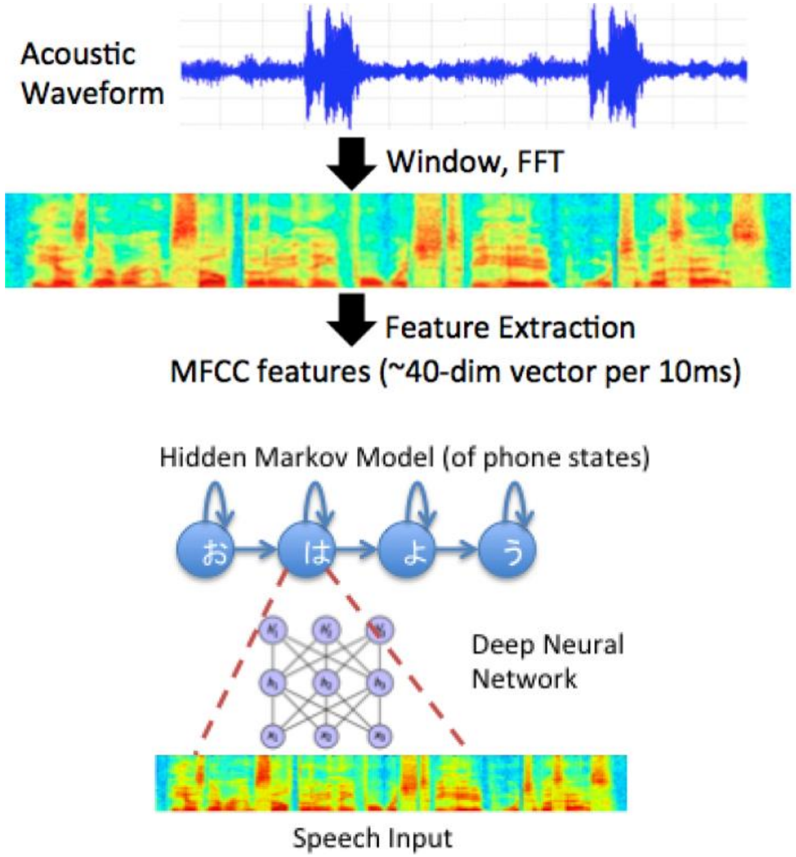
- Graph Convolutional Networks: networks for learning from graphs

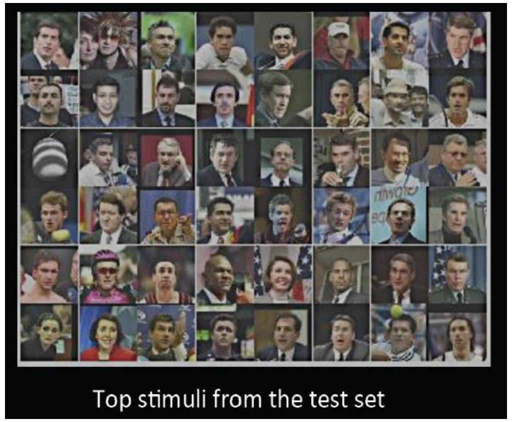
Autonomous Land Vehicle

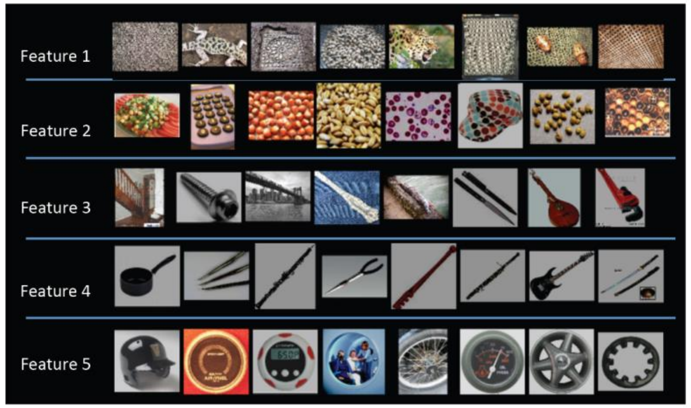
(DARPA's GrandChallenge 2005 contest)

Other success stories

http://cl.naist.jp/~kevinduh/a/deep2014/140121-ResearchSeminar.pdf







Convolution Networks

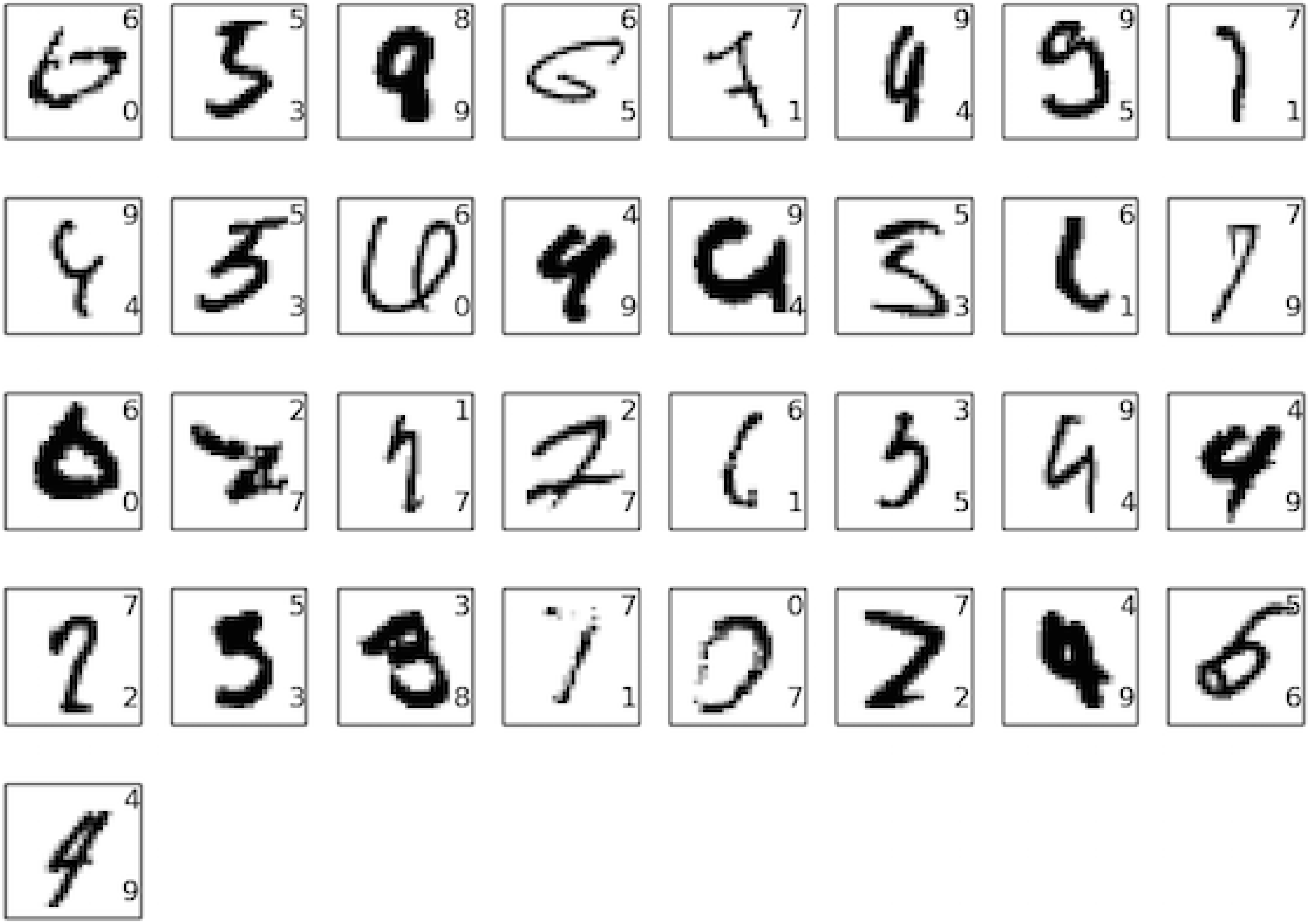
training:
60.000 images

testing:
10.000 images

each image:
32x32 pixels

accuracy: **99.7%**
(on the test set)

All 33 misclassified digits

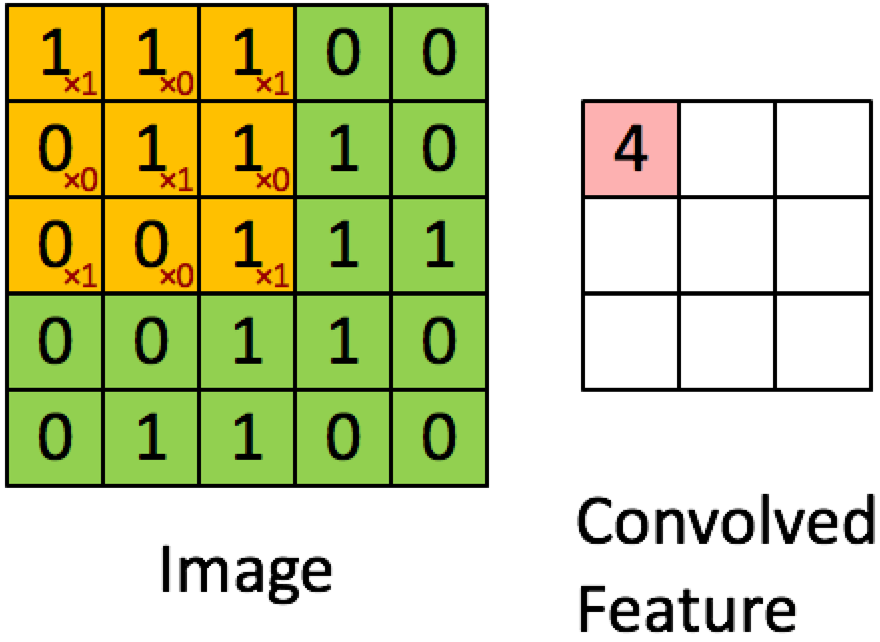


A Convolutional Filter

We want to find locations in the that look like a 3x3 cross.
We take the matrix **F** (called **a convolutional filter**) and "convolve it" with all possible locations in the image.
We will get another (smaller) matrix with "degrees of overlap":

$$F=\begin{matrix}1 & 0 & 1\\ 0 & 1 & 0\\ 1 & 0 & 1\end{matrix}$$

"multiply and add"



1 ×1	1 ×0	1 ×1	0	0
0 ×0	1 ×1	1 ×0	1	0
0 ×1	0 ×0	1 ×1	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

Convolved Feature

Filters as feature detectors

A filter: a “feature detector” – returns high values when the corresponding patch is similar to the filter matrix

Think about all pixels being **-1** (black) or **+1** (white) and filter parameters also restricted to **-1** and **1**

Example: what features are “detected” by:

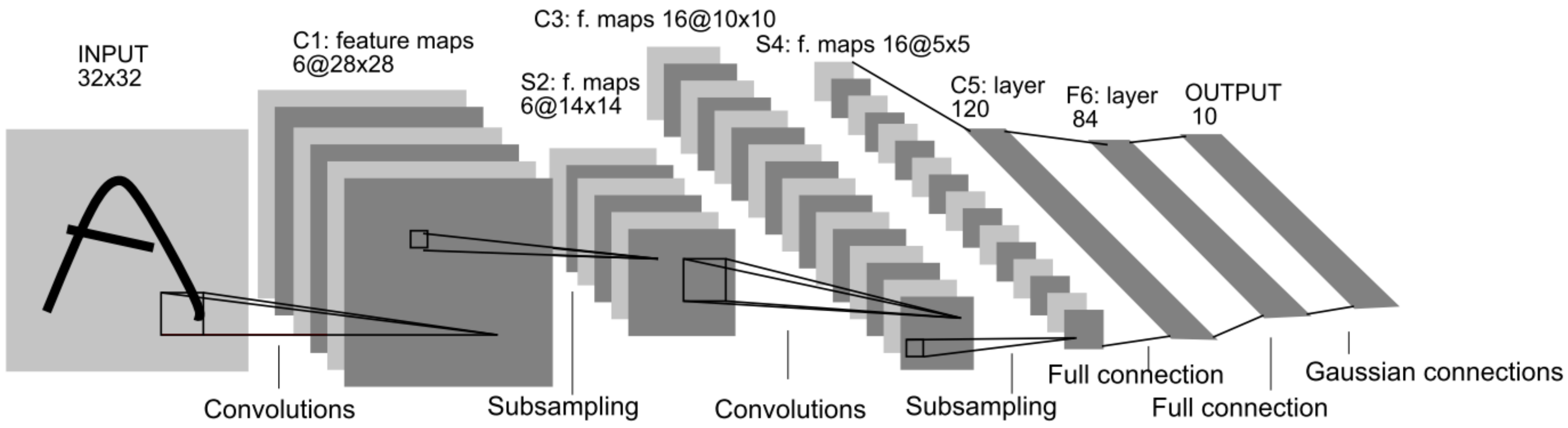
+	-	-
-	+	-
-	-	+

+	-	+
-	+	-
+	-	+

-	+	-
-	+	-
-	+	-

-	-	-
-	+	-
-	-	-

LeNet for digit recognition

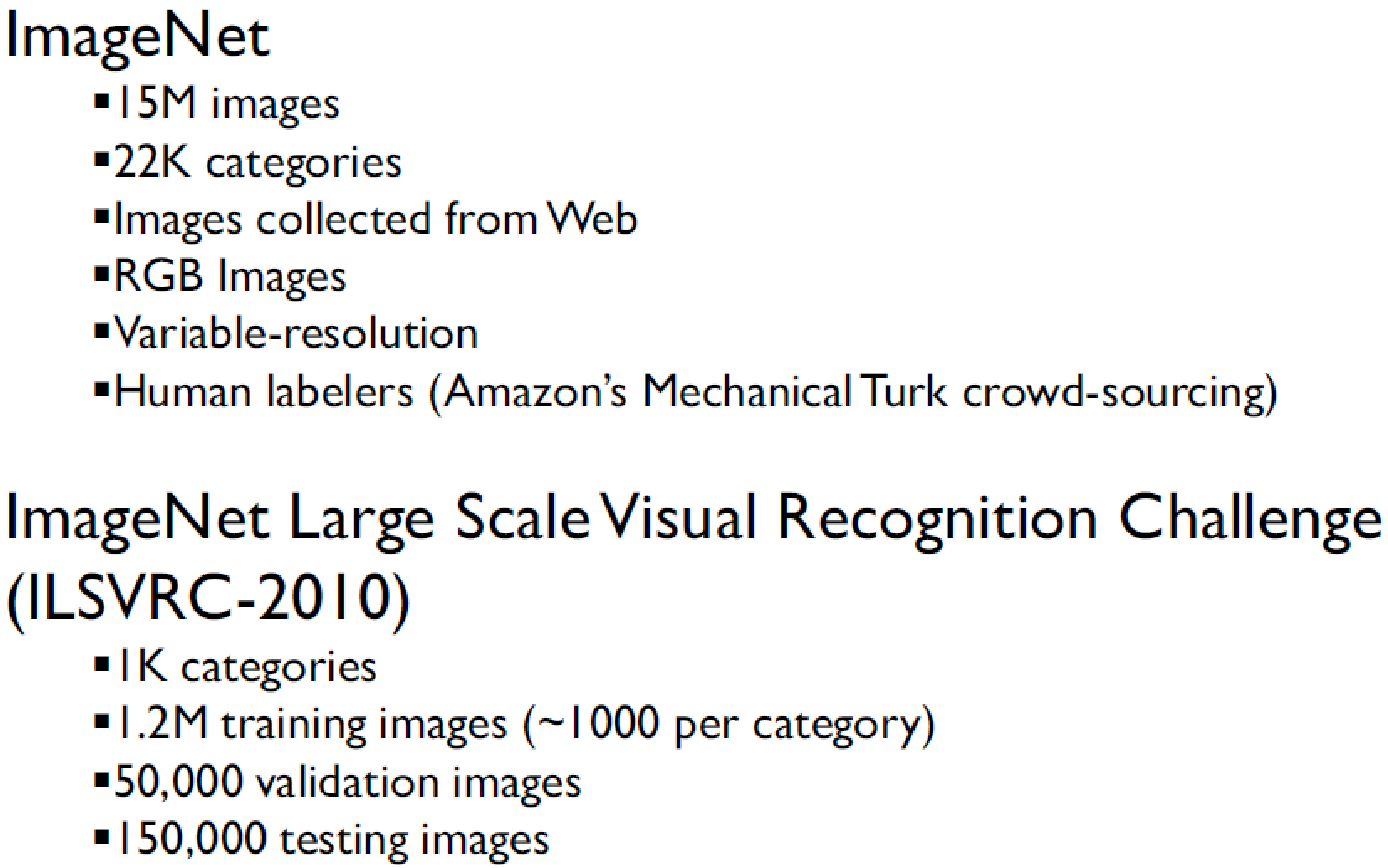


- C1: 6 convolutional filters of size 5x5
- S2: lowers the resolution by factor 2
- C3: 16 filters of size 5x5
- …
- The whole network has:
  - 1256 nodes
  - 64.660 connections
  - **9.760 trainable parameters (and not millions!)**
  - **trained with the Backpropagation algorithm!**

ImageNet

- 15M images
- 22K categories
- Images collected from Web
- Human labelers (Amazon’s Mechanical Turk crowd-sourcing)
- ImageNet Large Scale Visual Recognition Challenge (ILSVRC-2010)
  - 1K categories
  - 1.2M training images (~1000 per category)
  - 50,000 validation images
  - 150,000 testing images
- RGB images
- Variable-resolution, but this architecture scales them to 256x256 size

From LeNet to ImageNet (2010/2012)



ImageNet

- 15M images
- 22K categories
- Images collected from Web
- RGB Images
- Variable-resolution
- Human labelers (Amazon's Mechanical Turk crowd-sourcing)

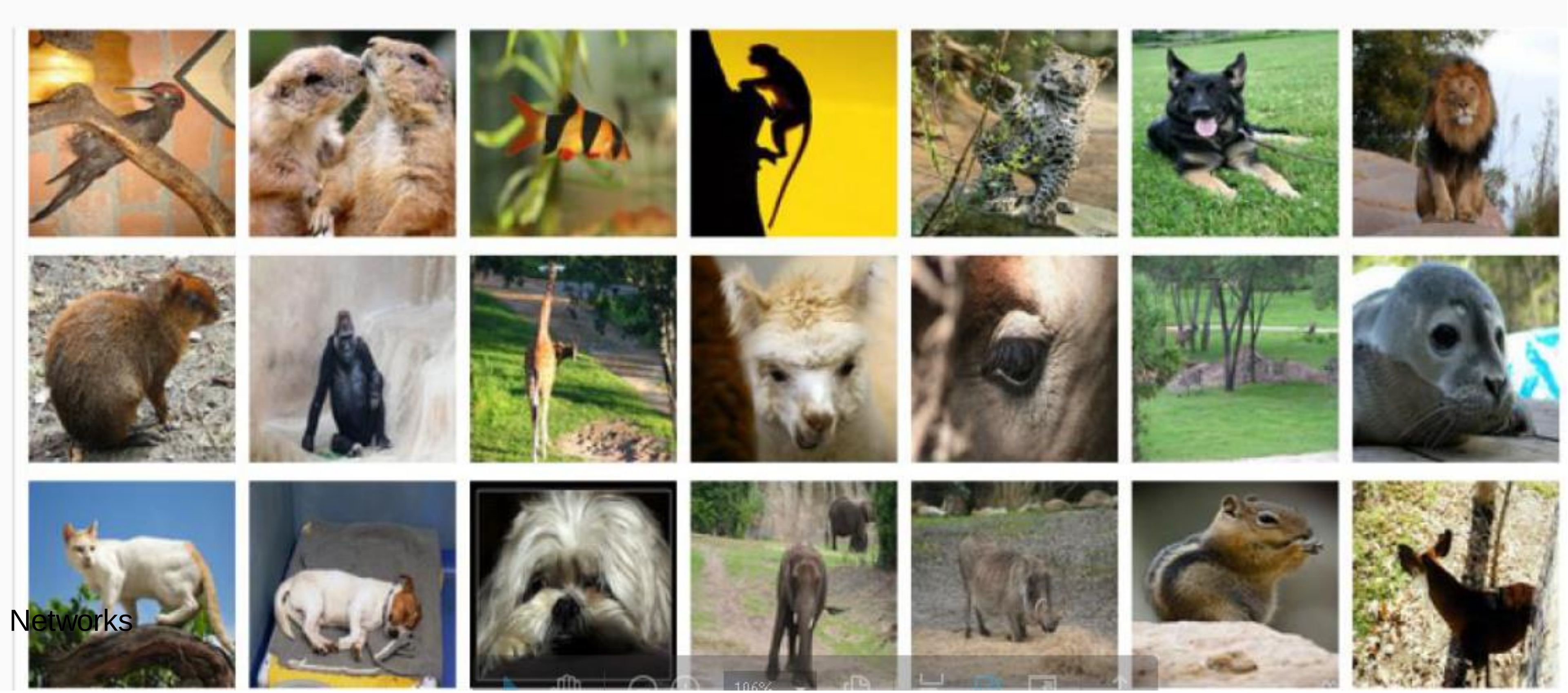
ImageNet Large Scale Visual Recognition Challenge (ILSVRC-2010)

- 1K categories
- 1.2M training images (~1000 per category)
- 50,000 validation images
- 150,000 testing images

ImageNet

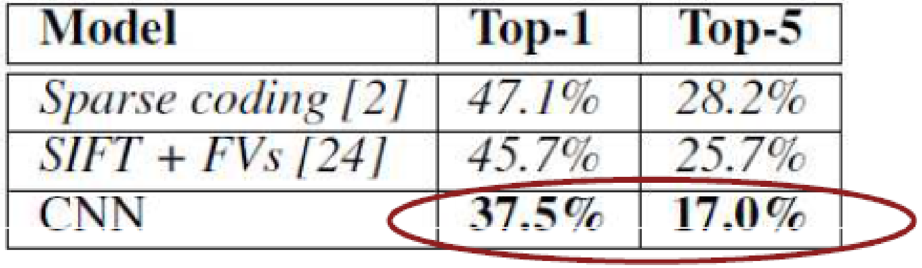
Classification goals:

- ❑ Make 1 guess about the label (Top-1 error)
- ❑ make 5 guesses about the label (Top-5 error)



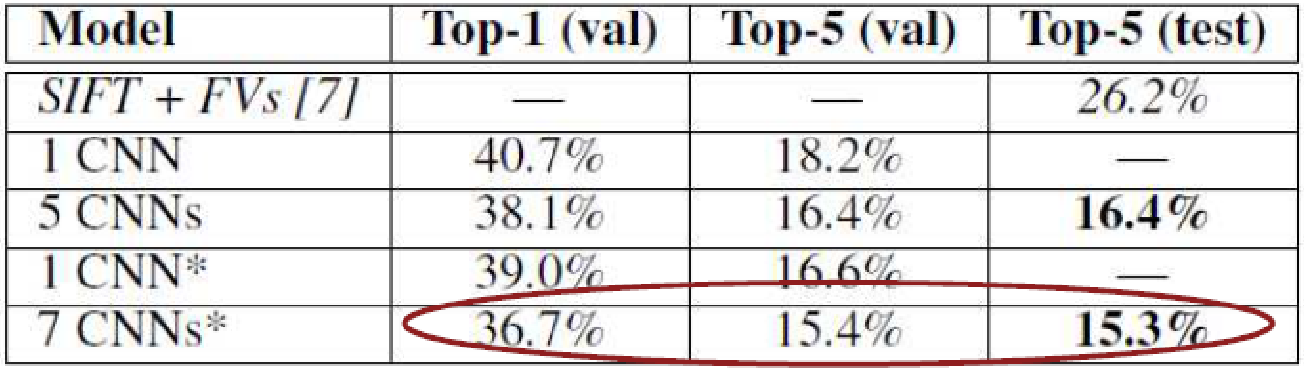
ImageNet

- ILSVRC-2010 test set



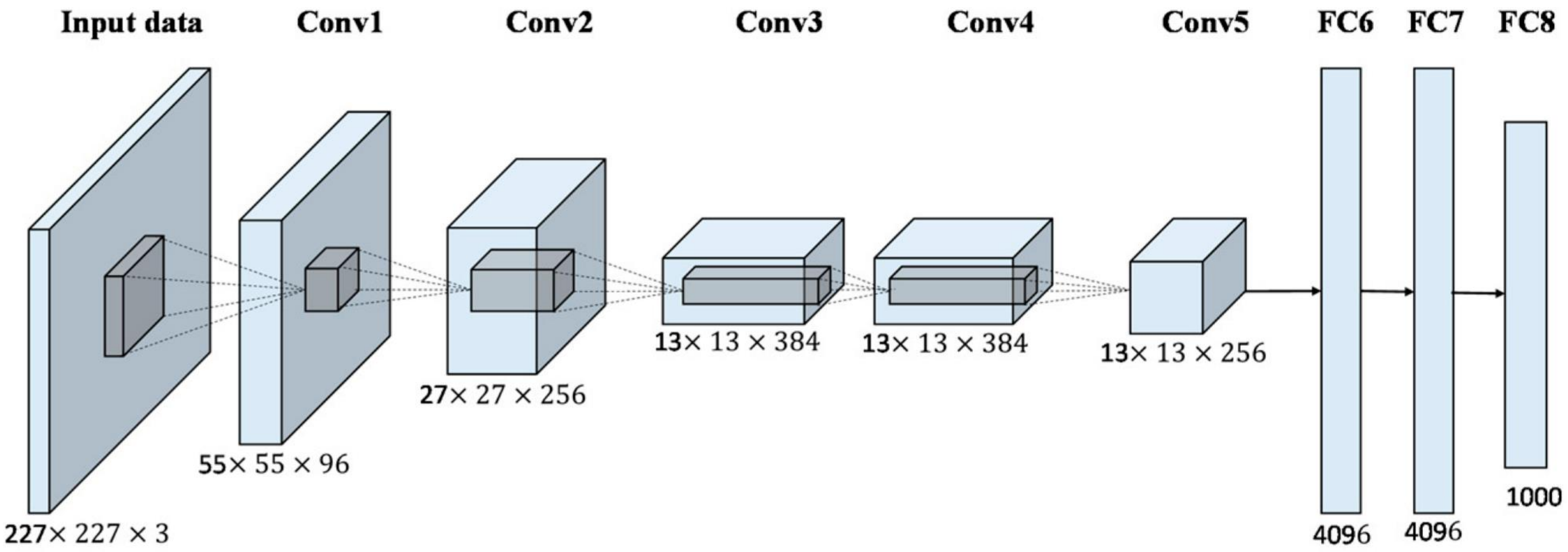
Model	Top-1	Top-5
Sparse coding [2]	*47.1%*	*28.2%*
SIFT + FVs [24]	*45.7%*	*25.7%*
CNN	**37.5%**	**17.0%**

- ILSVRC-2012 test set



Model	Top-1 (val)	Top-5 (val)	Top-5 (test)
SIFT + FVs [7]	—	—	*26.2%*
1 CNN	40.7%	18.2%	—
5 CNNs	38.1%	16.4%	**16.4%**
1 CNN*	39.0%	16.6%	—
7 CNNs*	36.7%	15.4%	**15.3%**

AlexNet simplified

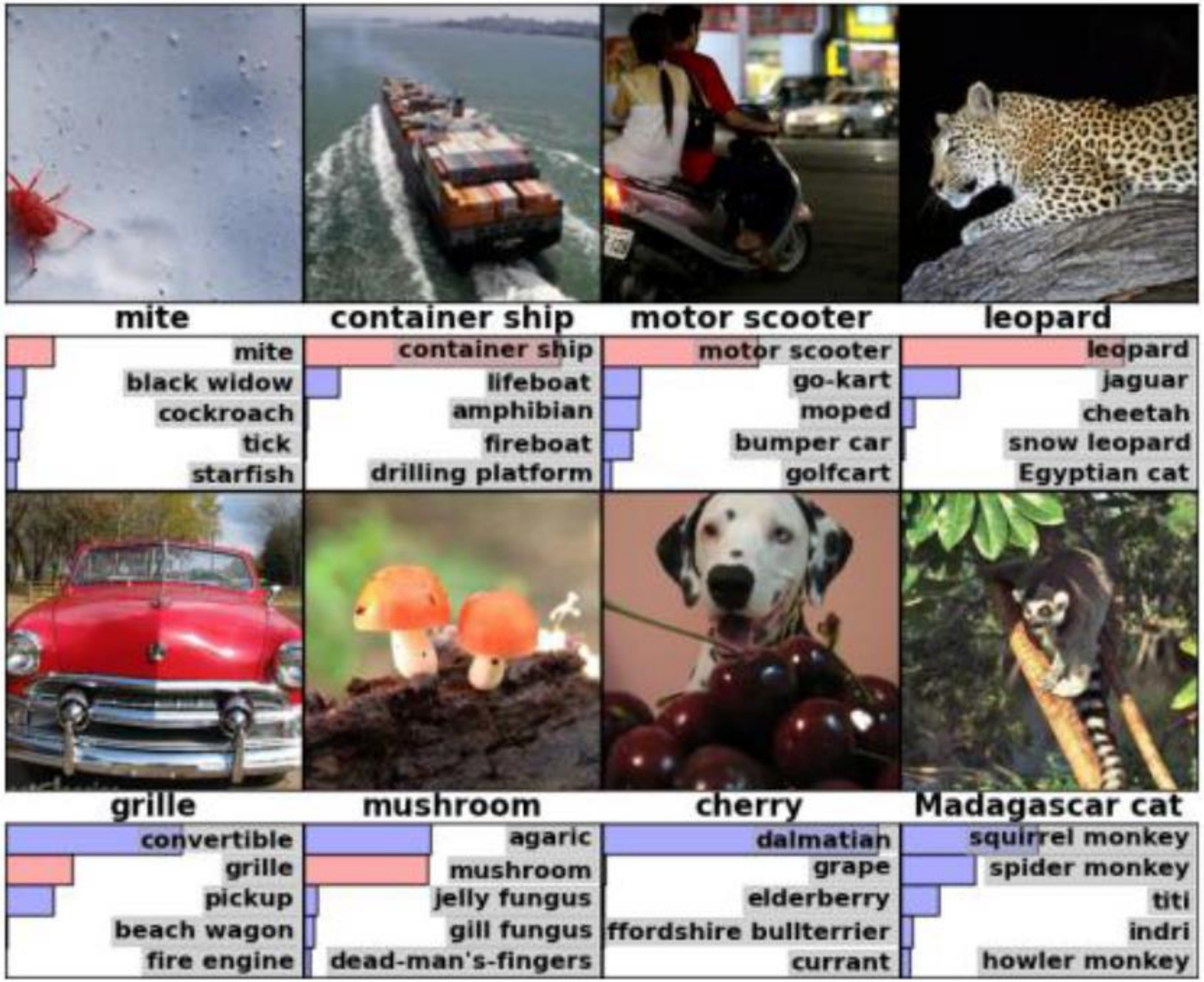


- not shown: max-pooling and local contrast normalization applied between convolutional layers

AlexNet: additional “tricks”

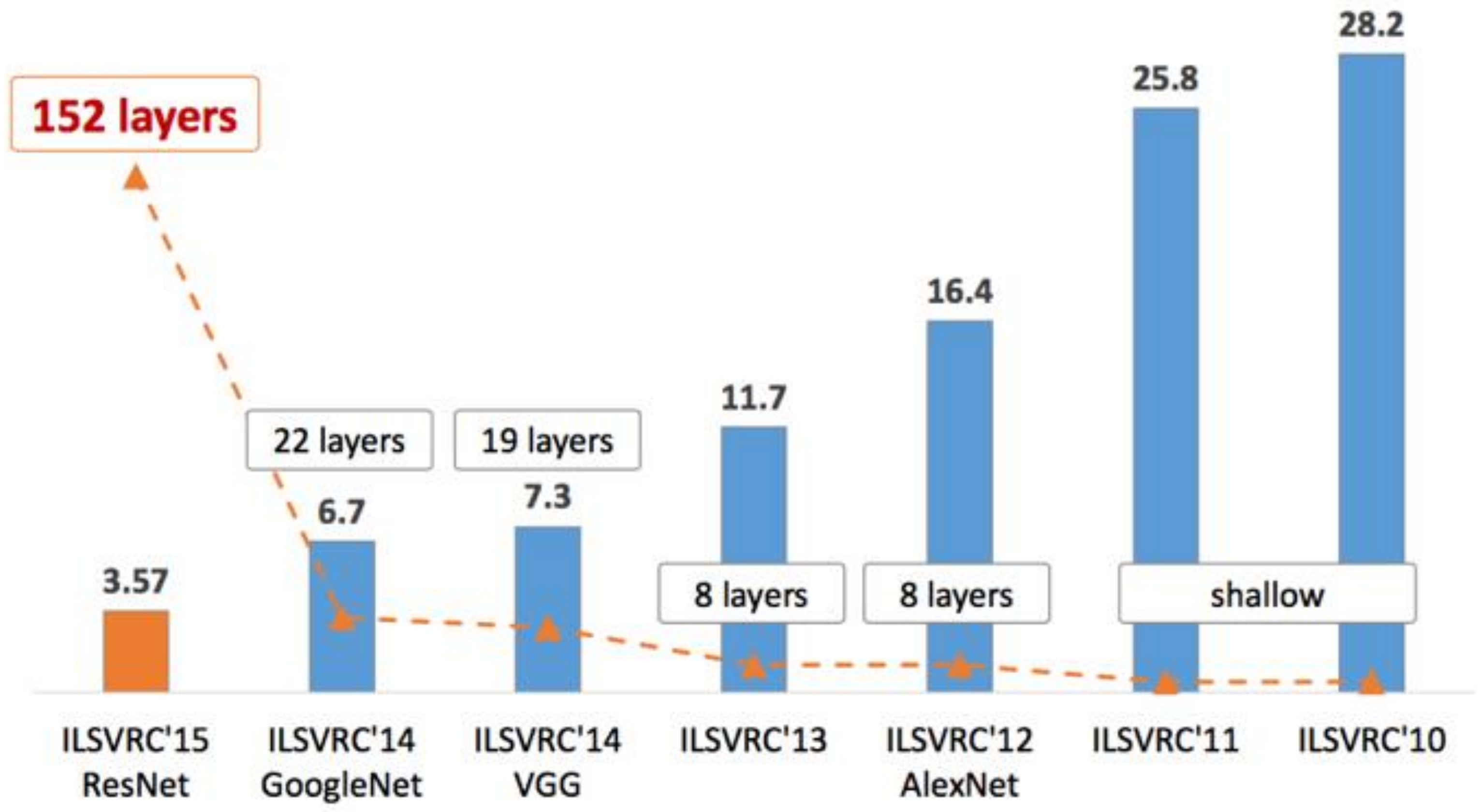
- Computations distributed over 2GPUs
- Local Contrast Normalization
- ReLU (Rectified Linear Unit) instead of sigmoid activation functions
- L2 weight normalization: punish big weights
- Data Augmentation: increase the number of training records by applying some modifications: shifts, contrasts, ...
- Dropout: when training, in every iteration disable 50% nodes (disabling weights doesn’t work!)

Results



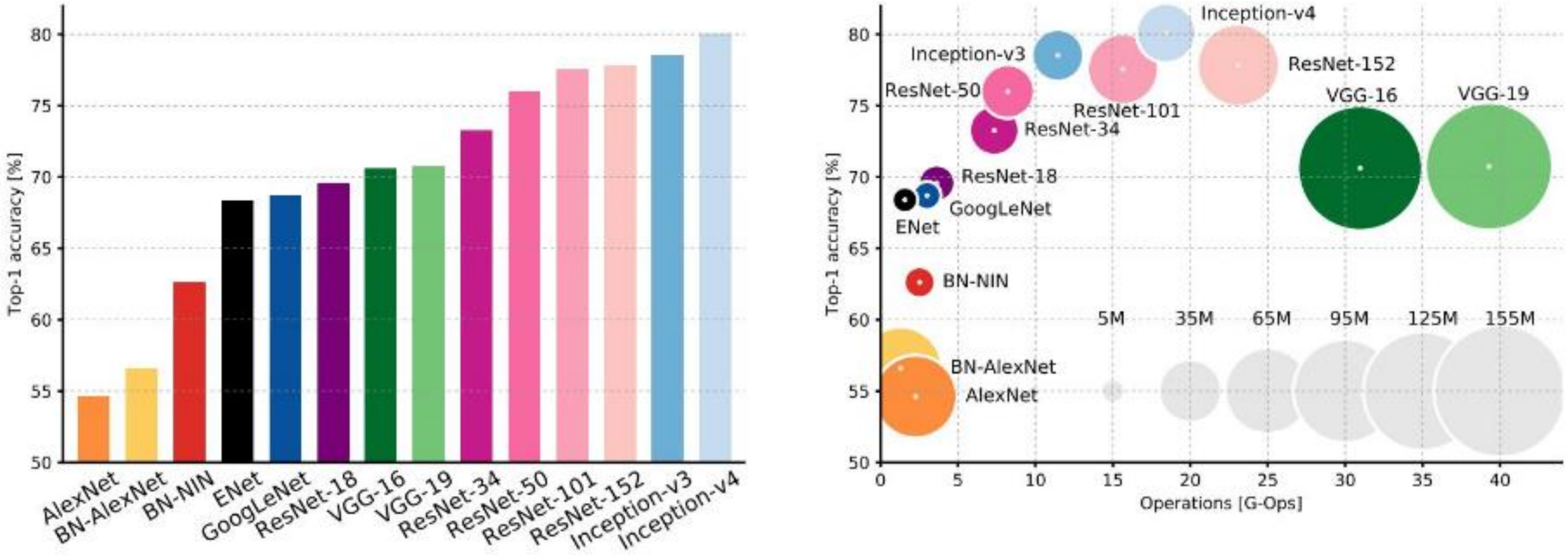
mite
mite
black widow
cockroach
tick
starfish
container ship
container ship
lifeboat
amphibian
fireboat
drilling platform
motor scooter
motor scooter
go-kart
moped
bumper car
golfcart
leopard
leopard
jaguar
cheetah
snow leopard
Egyptian cat
grille
convertible
grille
pickup
beach wagon
fire engine
mushroom
agaric
mushroom
jelly fungus
gill fungus
dead-man's-fingers
cherry
dalmatian
grape
elderberry
ffordshire bullterrier
currant
Madagascar cat
squirrel monkey
spider monkey
titi
indri
howler monkey

CNNs: progress



https://medium.com/@siddharthdas_32104/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5

CNNs: progress



An Analysis of Deep Neural Network Models for Practical Applications, 2017.

https://medium.com/@siddharthdas_32104/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5

Residual Networks

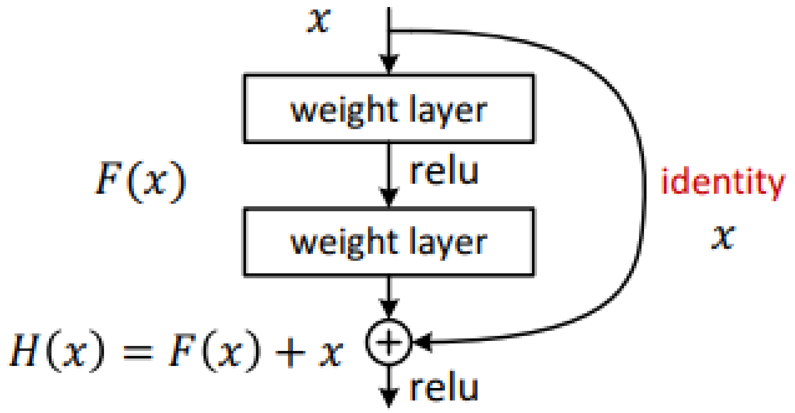
(He, Zhang, Ren, Sun, 2015)

- Key idea: it’s easier to learn “the modification of the original image than the modified image”

- Implementation of the key idea: add *identity shortcuts* between 2 (or more) layers

- Huge jump in accuracy

- 1000 layers is possible!



Reinforcement Learning & Deep Mind

2013: DeepMind publishes a paper:
DNQ: Playing Atari with Deep Reinforcement Learning

“zero-knowledge”:

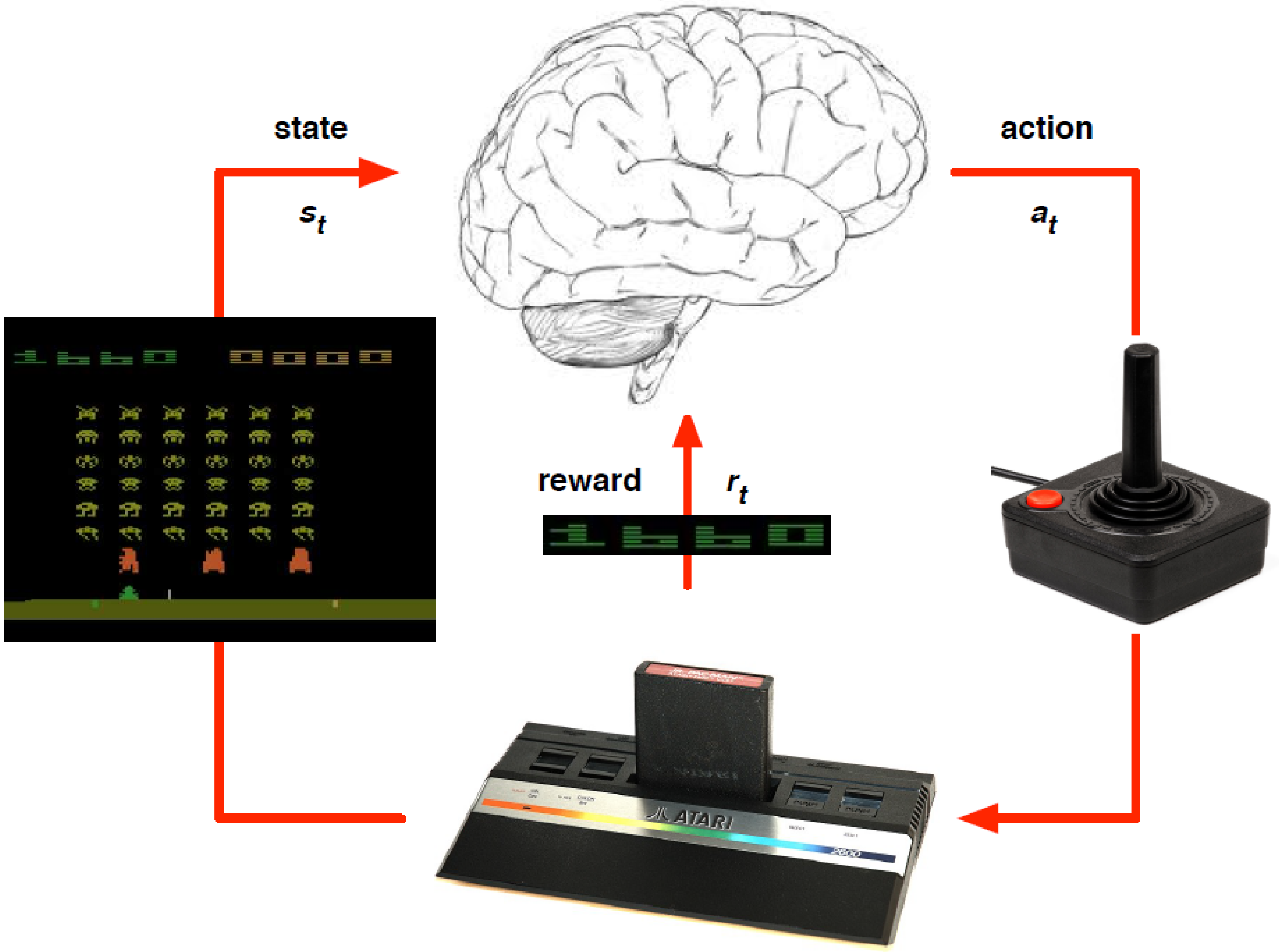
- a deep neural network can learn to **play any Atari 2600 games** using as input screen dumps and as output joystick positions

- training process: **a few months of CPU-time; 2 days of GPU-time**

2016: DeepMind/Google develops another network
AlphaGo network beats the world GO champion 6:0

2017: DeepMind develops a network that learns “from scratch” to play chess and GO beating anything that exists! AlphaGo-Zero

Reinforcement Learning in Atari



state

s_t

action

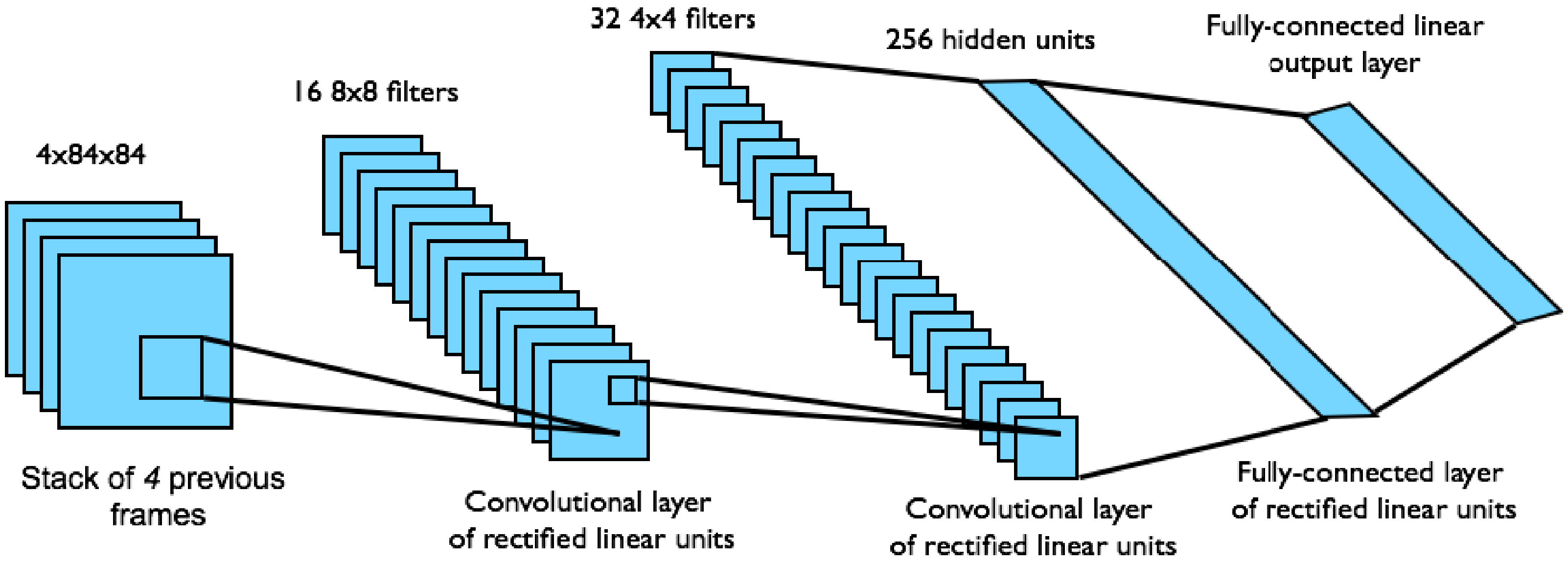
a_t

reward

r_t

DQN in Atari

- End-to-end learning of values $Q(s, a)$ from pixels s
- Input state s is stack of raw pixels from last 4 frames
- Output is $Q(s, a)$ for 18 joystick/button positions
- Reward is change in score for that step


4x84x84
16 8x8 filters
32 4x4 filters
256 hidden units
Fully-connected linear output layer
Stack of 4 previous frames
Convolutional layer of rectified linear units
Convolutional layer of rectified linear units
Fully-connected layer of rectified linear units

Network architecture and hyperparameters fixed across all games

[Mnih et al.]

DQN for Atari 2600 Games

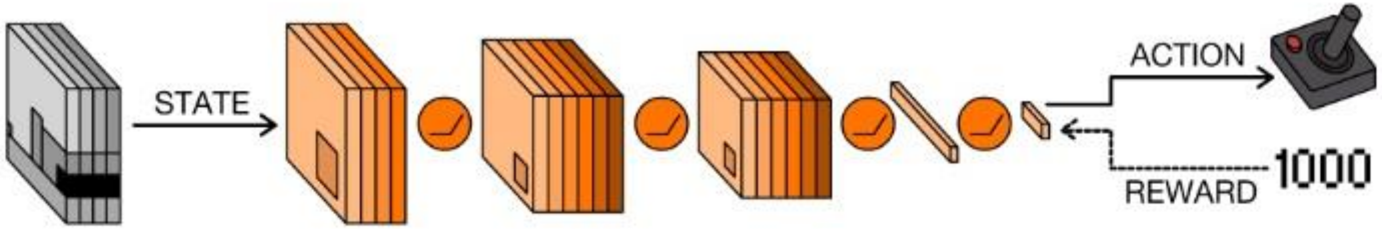
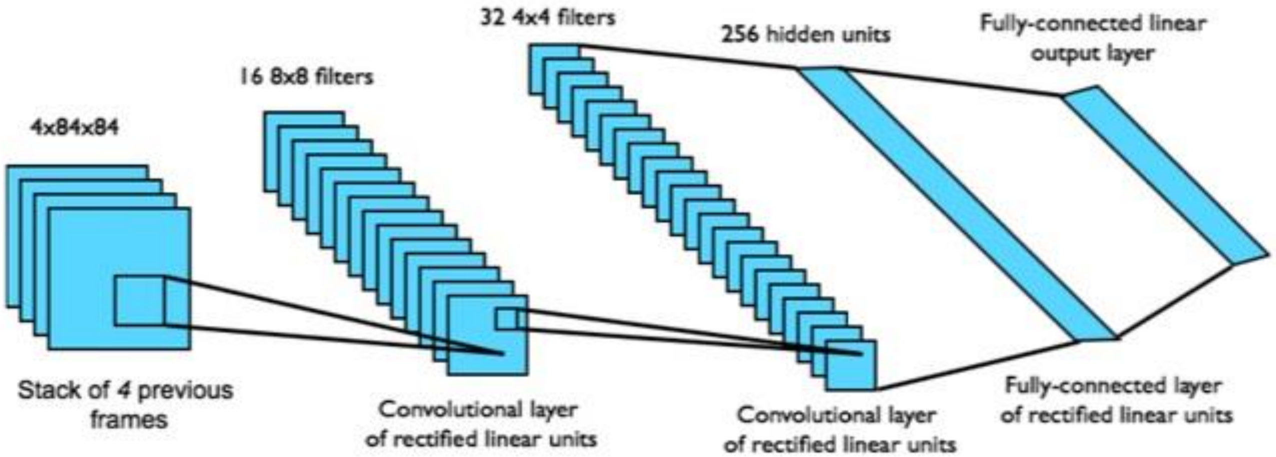
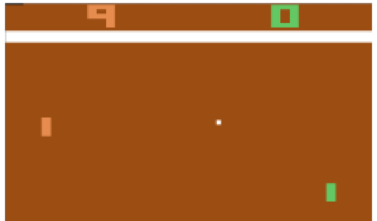
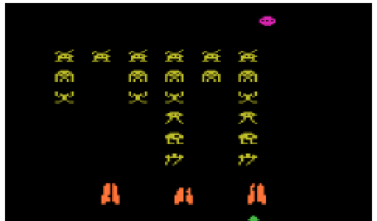
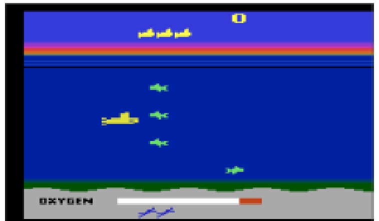


Fig. 4. The DQN [71]. The network takes the state—a stack of greyscale frames from the video game—and processes it with convolutional and fully connected layers, with ReLU nonlinearities in between each layer. At the final layer, the network outputs a discrete action, which corresponds to one of the possible control inputs for the game. Given the current state and chosen action, the game returns a new score. The DQN uses the reward—the difference between the new score and the previous one—to learn from its decision. More precisely, the reward is used to update its estimate of Q, and the error between its previous estimate and its new estimate is backpropagated through the network.


4x84x84
16 8x8 filters
32 4x4 filters
256 hidden units
Fully-connected linear output layer
Stack of 4 previous frames
Convolutional layer of rectified linear units
Convolutional layer of rectified linear units
Fully-connected layer of rectified linear units



AlphaGo, AlphaGo Zero, Alpha Zero

- October 2015: first version of AlphaGo beats European Go Champion Fan Hui 5:0
- March 2016: AlphaGo beats World Go Champion, Lee Sedol AlphaGo was pre-trained on a huge database of historical games, and then improved by playing against itself; a few TPUs
- 19 October 2017: AlphaGo Zero, trained solely on self-played games surpasses AlphaGo, beating it 100:0, hardware costs estimated on $25 million
- December 5, 2017: Alpha Zero announced: a generic architecture learning to play Go, Chess, Shogi in hours (4 hours to learn Chess on a superhuman level); hardware: 5000 TPUs

What NEXT?