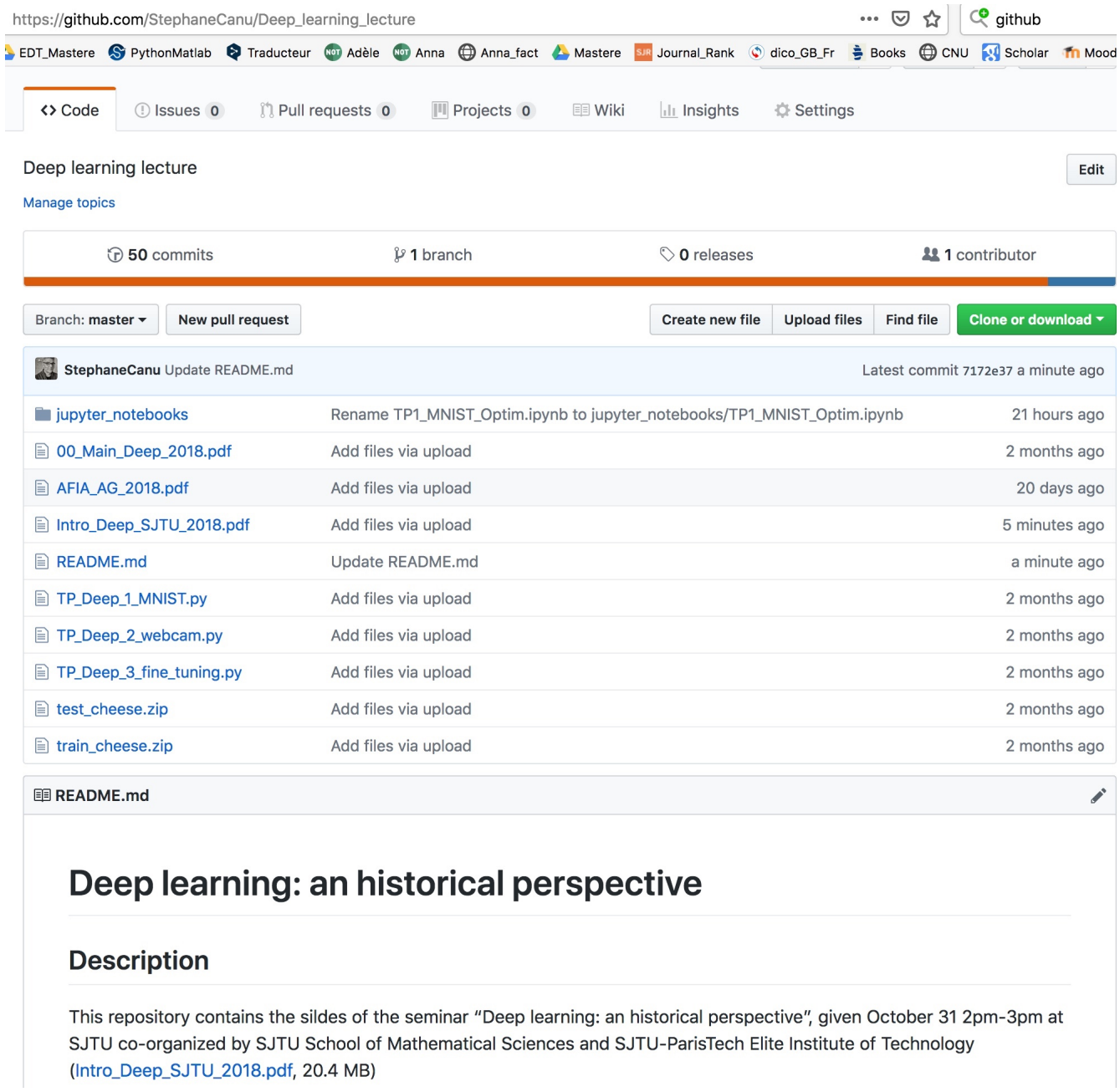
Introduction to Deep Learning

S. Canu

CRIANN

Friday, june 14

github.com/StephaneCanu/Deep_learning_lecture



https://github.com/StephaneCanu/Deep_learning_lecture

github

EDT_Mastere PythonMatlab Traducteur Adèle Anna Anna_fact Mastere Journal_Rank dico_GB_Fr Books CNU Scholar Mood

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

Deep learning lecture

Edit

Manage topics

50 commits 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download

StephaneCanu Update README.md Latest commit 7172e37 a minute ago

jupyter_notebooks	Rename TP1_MNIST_Optim.ipynb to jupyter_notebooks/TP1_MNIST_Optim.ipynb	21 hours ago
00_Main_Deep_2018.pdf	Add files via upload	2 months ago
AFIA_AG_2018.pdf	Add files via upload	20 days ago
Intro_Deep_SJTU_2018.pdf	Add files via upload	5 minutes ago
README.md	Update README.md	a minute ago
TP_Deep_1_MNIST.py	Add files via upload	2 months ago
TP_Deep_2_webcam.py	Add files via upload	2 months ago
TP_Deep_3_fine_tuning.py	Add files via upload	2 months ago
test_cheese.zip	Add files via upload	2 months ago
train_cheese.zip	Add files via upload	2 months ago

README.md

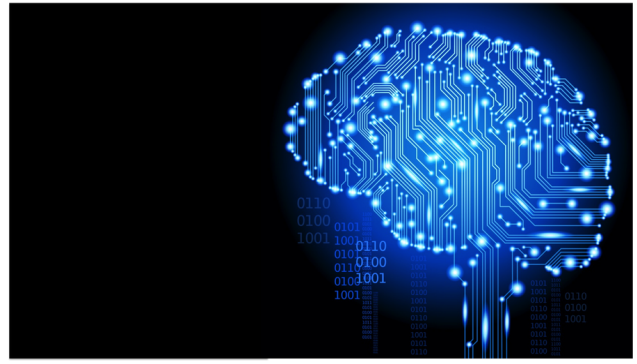
Deep learning: an historical perspective

Description

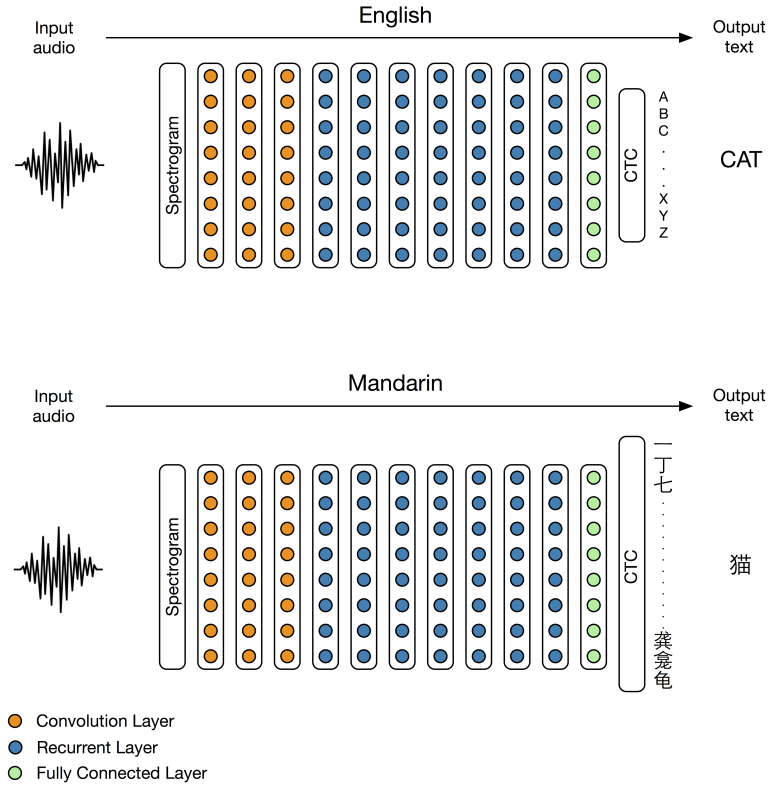
This repository contains the sildes of the seminar "Deep learning: an historical perspective", given October 31 2pm-3pm at SJTU co-organized by SJTU School of Mathematical Sciences and SJTU-ParisTech Elite Institute of Technology (Intro_Deep_SJTU_2018.pdf, 20.4 MB)

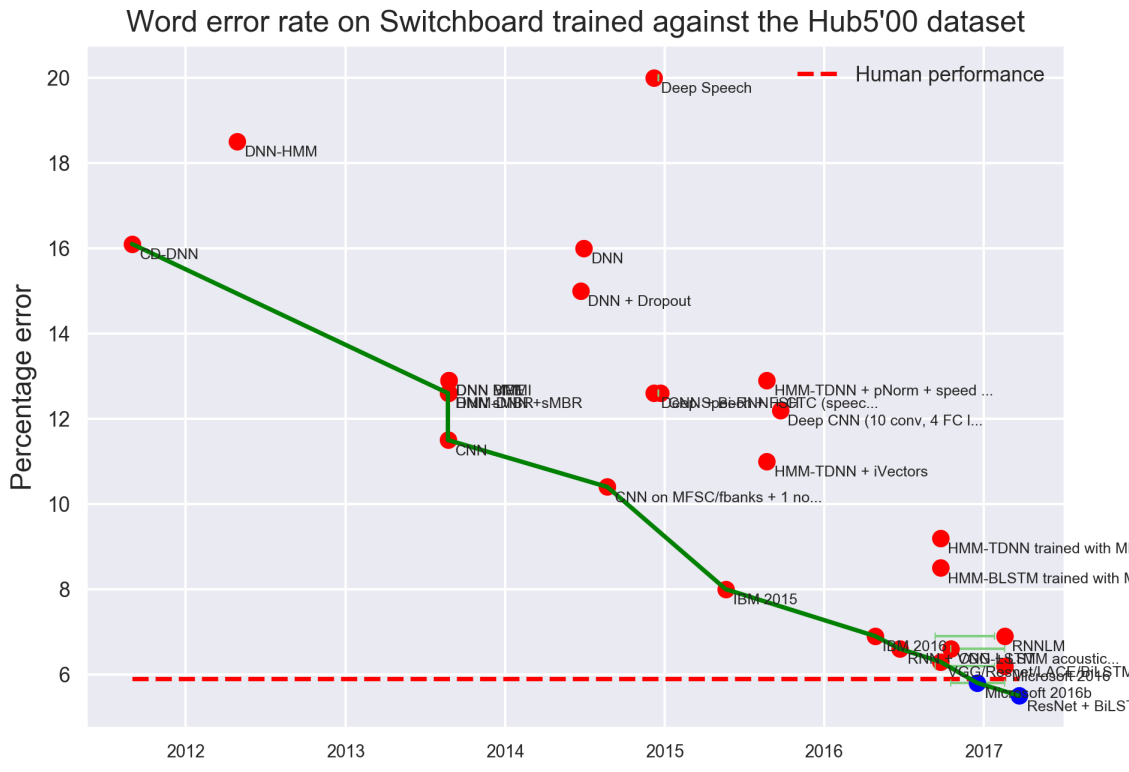
Road map

1. Why deep learning?
2. The first stage: 1890 - 1969
3. The second stage: 1985 - 1995
4. The third stage: 2006 - (2012) - 2018...
5. What's new in deep learning?
   - Big is beautiful
   - Two Hot topics: data and architecture
6. Conclusion



Deep learning for turning text into speech (and vice versa)



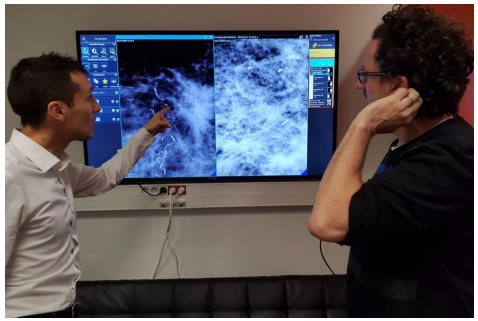


Baidu deep speech 2 (2015) and Deep voice (2017)

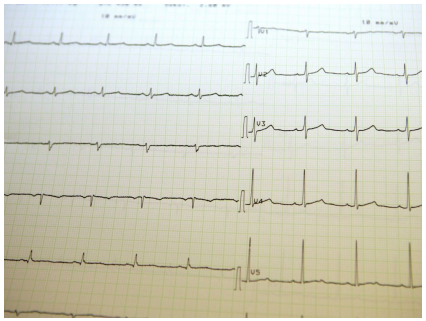
Trained on 9,400 hours of labeled audio with 11 million utterances.

Deep learning for healthcare

Skin cancer classification
130 000 training images
validation error rate : 28 % (human 34 %)



the Digital Mammography DREAM Challenge
640 000 mammographies (1209 participants)
5 % less false positive

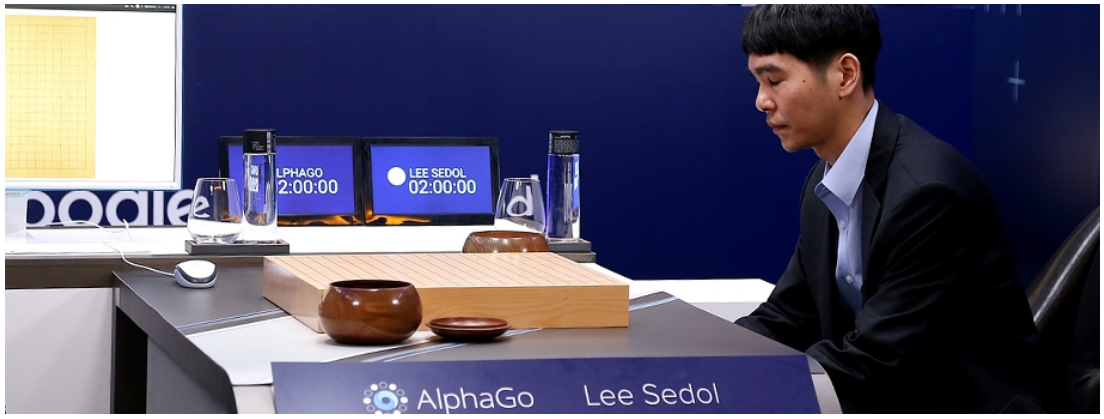


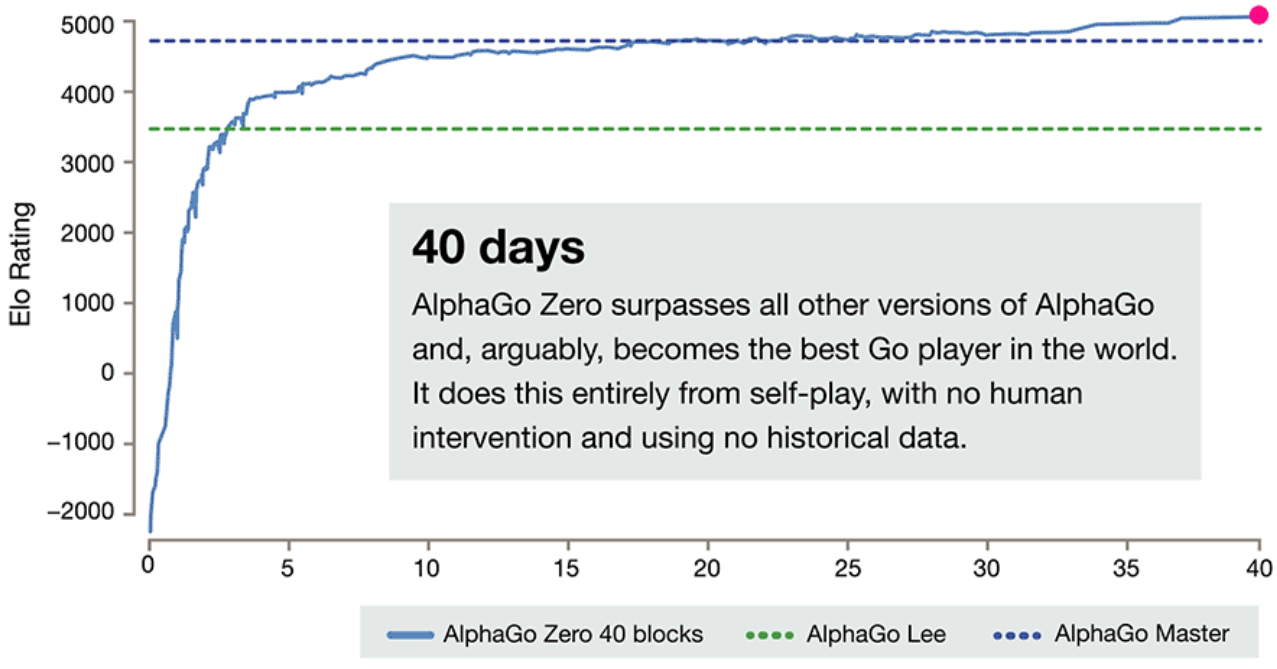
heart rate analysis
500 000 ECG
precision 92.6 % (humain 80.0 %) sensitivity 97 %

Statistical machine learning: retrieving correlations

with deep learning end-to-end architecture
"April showers bring May flowers"

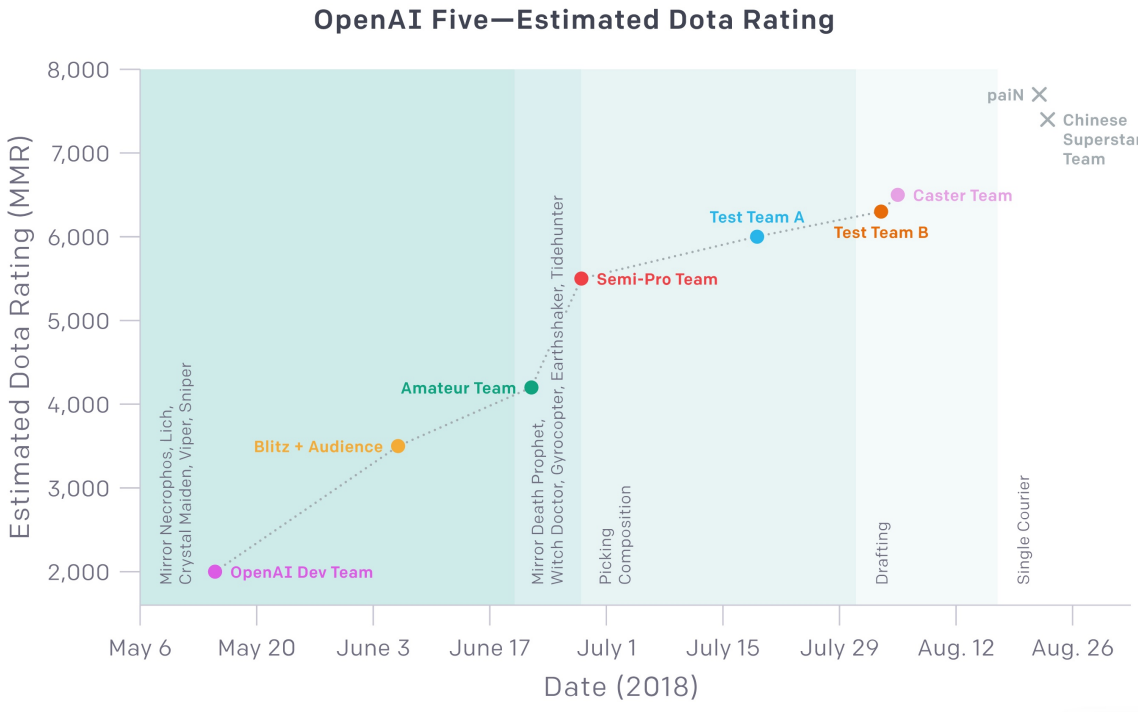
Deep learning success in playing GO





Mastering the game of Go without human knowledge D. Silver et al. Nature, 550, 2017

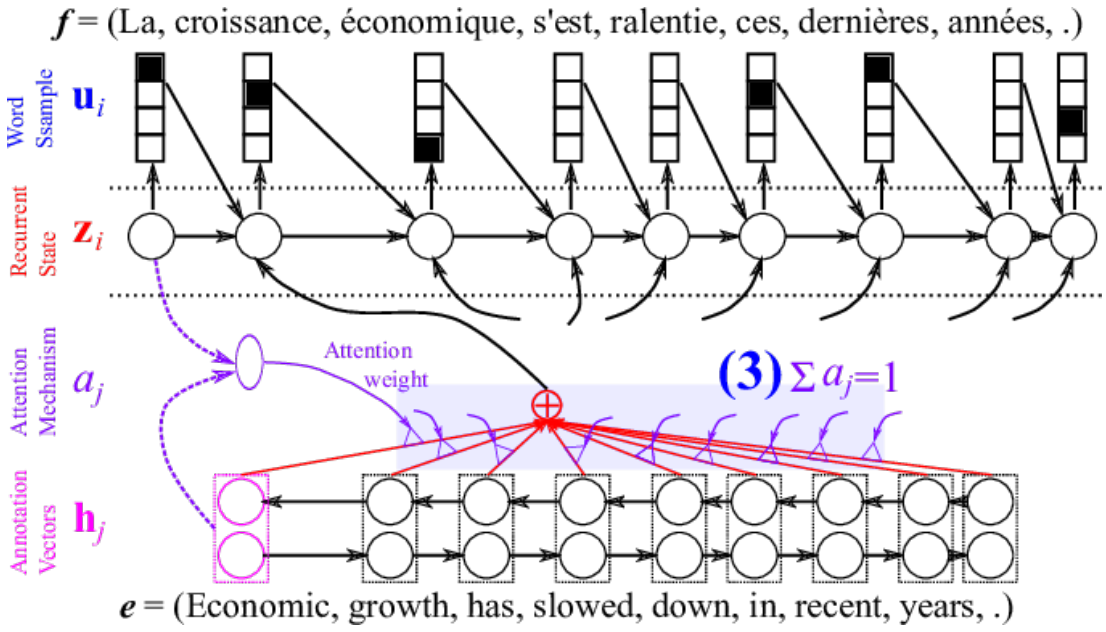
Deep learning progresses in playing Dota 2



- separate LSTM for each hero
- 180 years/days of games against itself
- Proximal Policy Optimization
- 256 GPUs and 128,000 CPU
- the OpenAI Five is very much still a work in progress project

https://blog.openai.com/the-international-2018-results/

Deep learning (limited) success in NLP

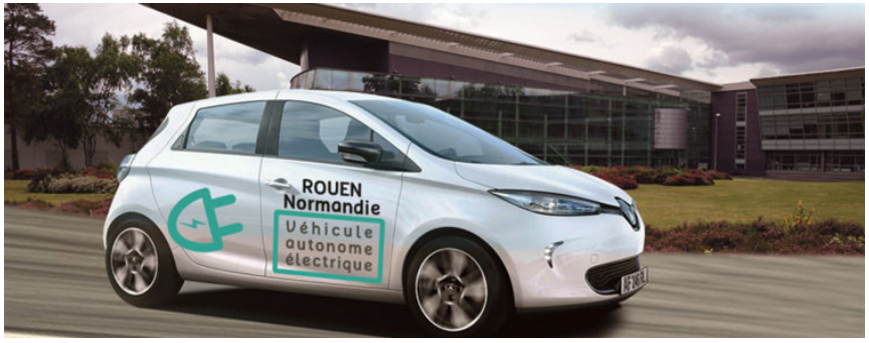


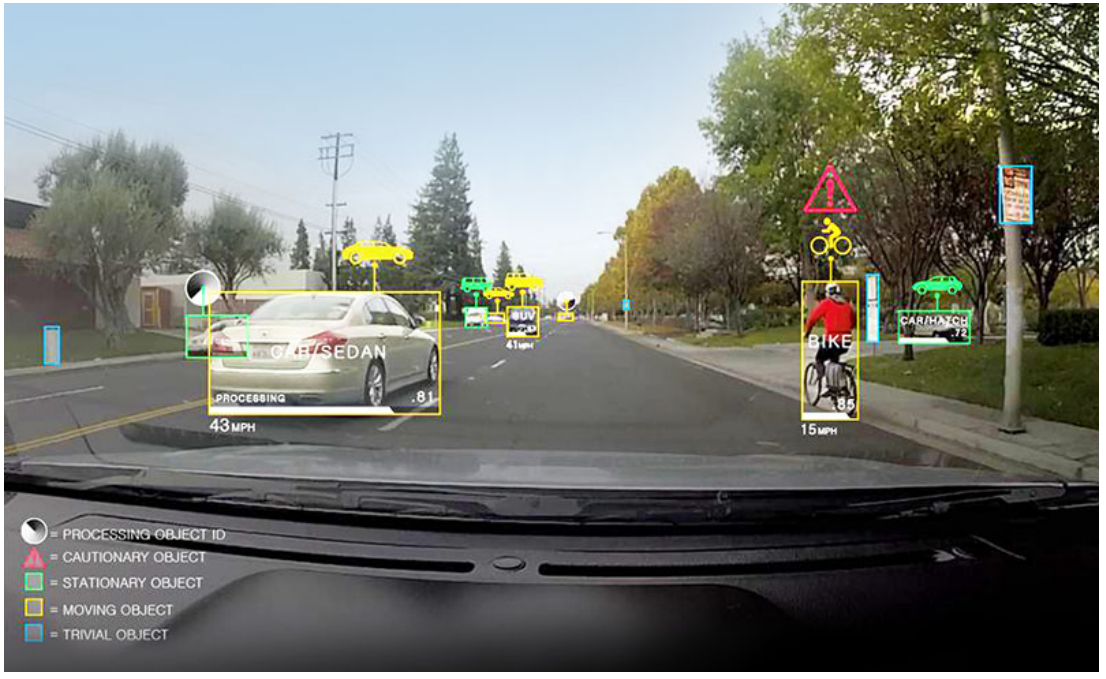
Learning to translate with 36 million sentences

- Near Human-Level Performance in Grammatical Error Correction
- Achieving Human Parity on Automatic News Translation

https://devblogs.nvidia.com/author/kcho/

Deep learning to drive: the Rouen autonomous lab





Driving Video Database = 100.000 videos – 120 million images

- When It Comes to Safety, Autonomous Cars Are Still "Teen Drivers"
- companies are developing many different levels of automation

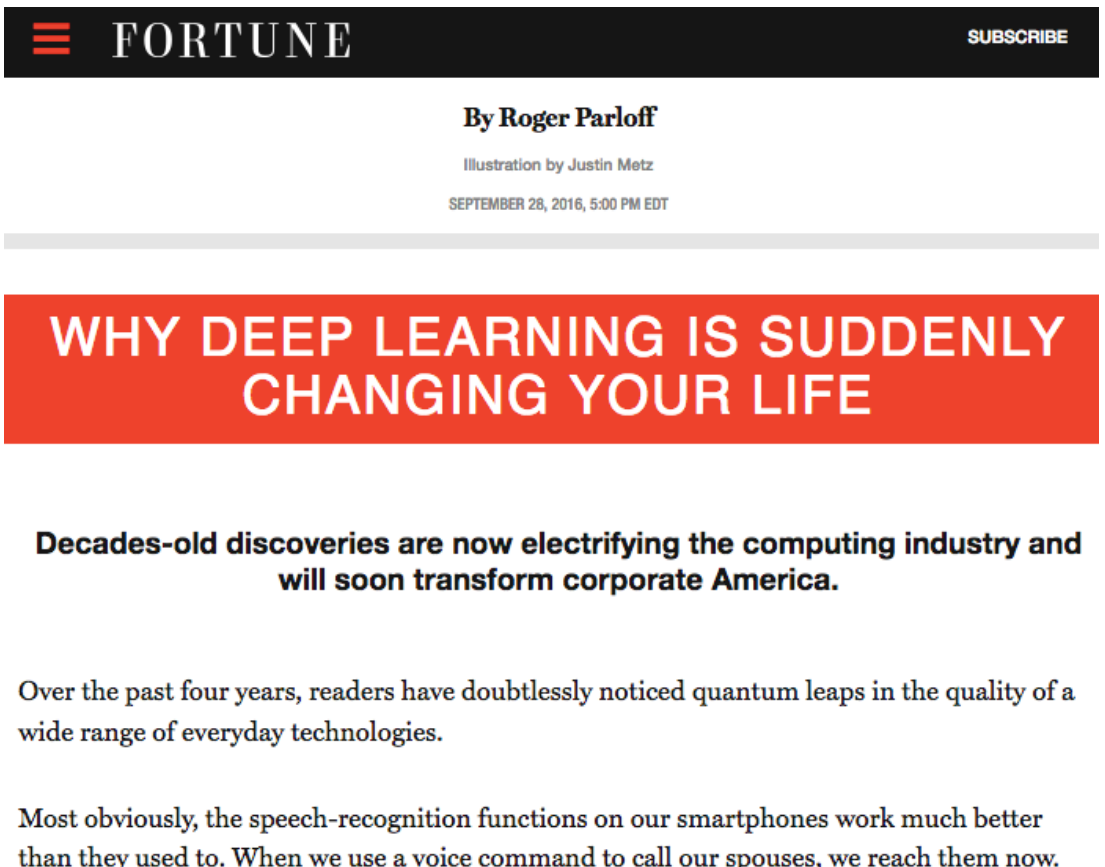
https://www.rouennormandyautonomouslab.com/?lang=en
http://bdd-data.berkeley.edu

So far, so good

- Deep learning performance breakthrough
  - Low level perception tasks: speech, image and video processing, natural language processing, games...
  - ...and specific tasks in health care, astronomy...

- It requires
  - Big data
  - Big computers
  - Specific tasks

- Yet to be solved
  - Complex games
  - Translation
  - Virtual Assistant
  - Autonomous vehicle



FORTUNE SUBSCRIBE

By Roger Parloff

Illustration by Justin Metz

SEPTEMBER 28, 2016, 5:00 PM EDT

WHY DEEP LEARNING IS SUDDENLY CHANGING YOUR LIFE

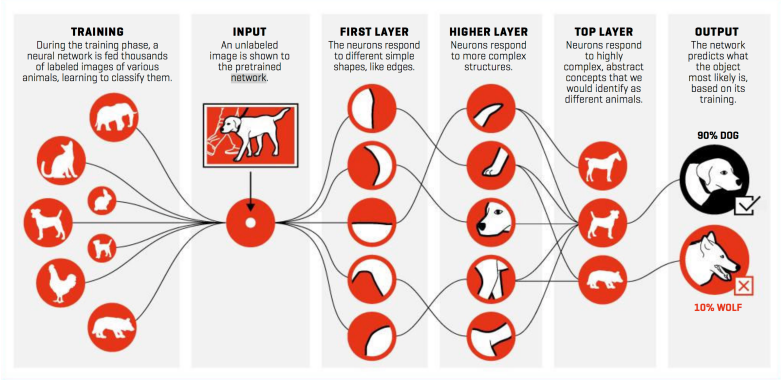
Decades-old discoveries are now electrifying the computing industry and will soon transform corporate America.

Over the past four years, readers have doubtlessly noticed quantum leaps in the quality of a wide range of everyday technologies.

Most obviously, the speech-recognition functions on our smartphones work much better than they used to. When we use a voice command to call our spouses, we reach them now.

Road map

1. Why deep learning?
2. The first stage: 1890 - 1969
3. The second stage: 1985 - 1995
4. The third stage: 2006 - (2012) - 2018...
5. What's new in deep learning?
   - Big is beautiful
   - Two Hot topics: data and architecture
6. Conclusion



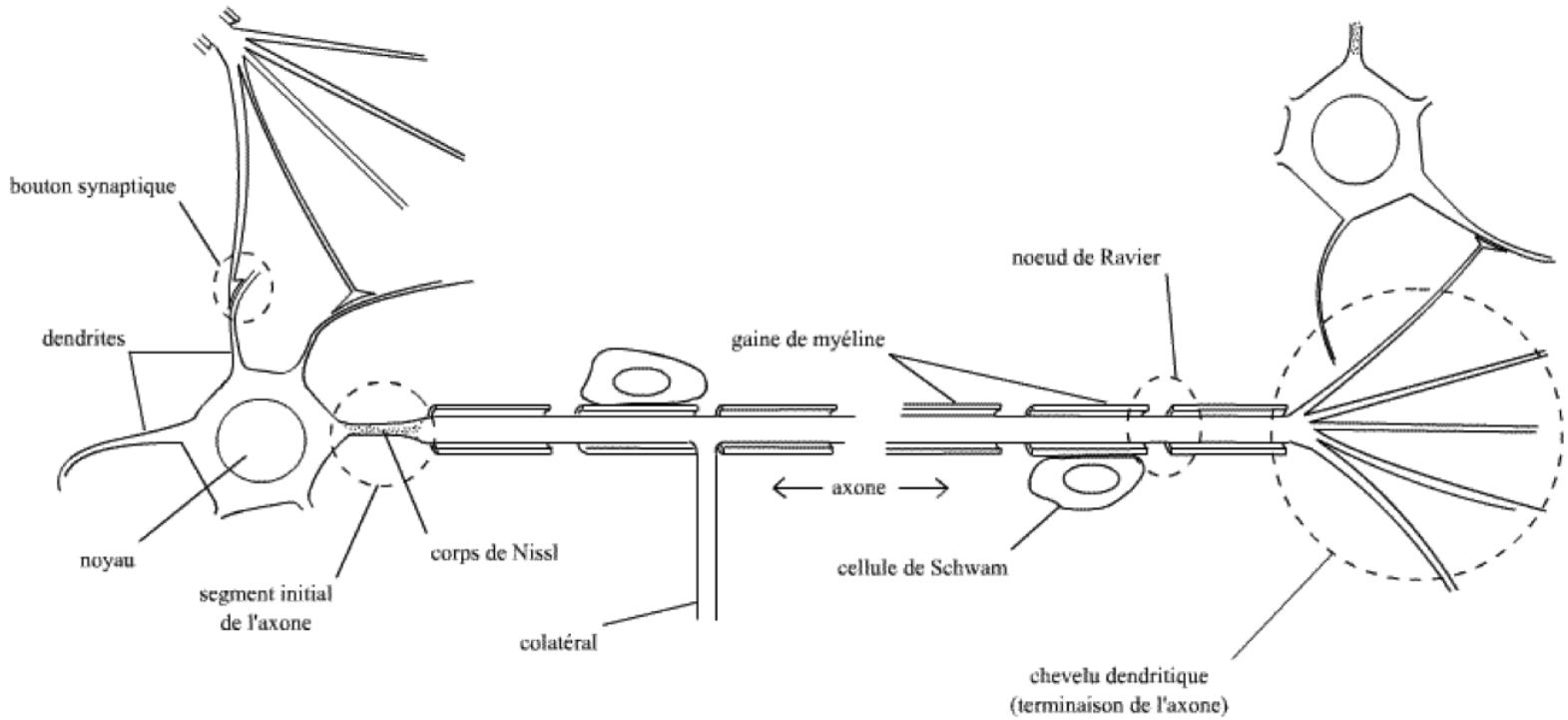
The neural networks time line

- The first stage: 1890 - 1969

~1890 Ramón y Cajal: the biological neuron
1943 McCulloch & Pitts formal neuron
1949 Hebb's rule
1958 Rosenblatt's Perceptron: learning with stochastic gradient
1969 Minsky & Papert: stop – the 1st NN winter

- The second stage: 1985 - 1995
- The third stage: 2006 - (2012) - 2018...

The biological neuron



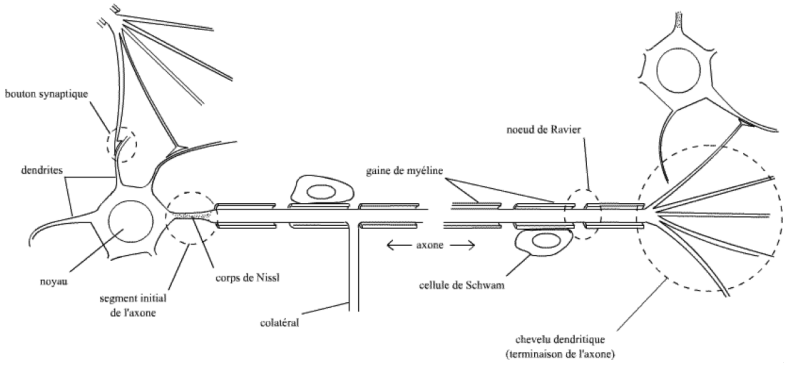
The neural networks time line

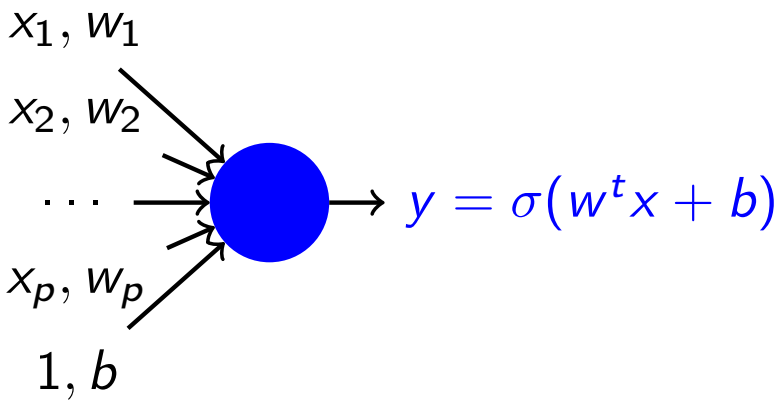
- The first stage: 1890 - 1969

~1890 Ramón y Cajal: the biological neuron
1943 McCulloch & Pitts formal neuron
1949 Hebb's rule
1958 Rosenblatt's Perceptron: learning with stochastic gradient
1969 Minsky & Papert: stop – the 1st NN winter

- The second stage: 1985 - 1995
- The third stage: 2006 - (2012) - 2018...

McCulloch & Pitts formal neuron 1943



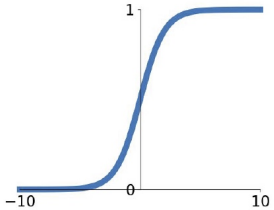


- x input $\in \mathbb{R}^p$
- w weight, b bias
- σ activation function
- y output $\in \mathbb{R}$

Activation functions

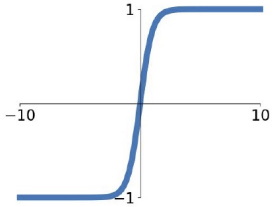
Sigmoid

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



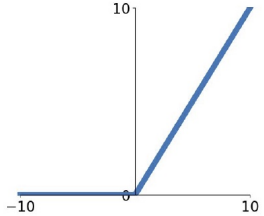
tanh

$$\tanh(x)$$



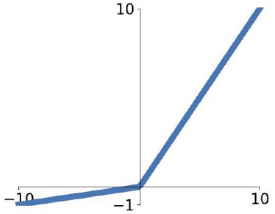
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

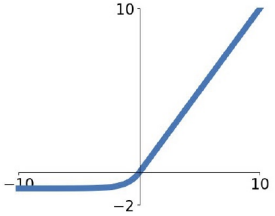


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



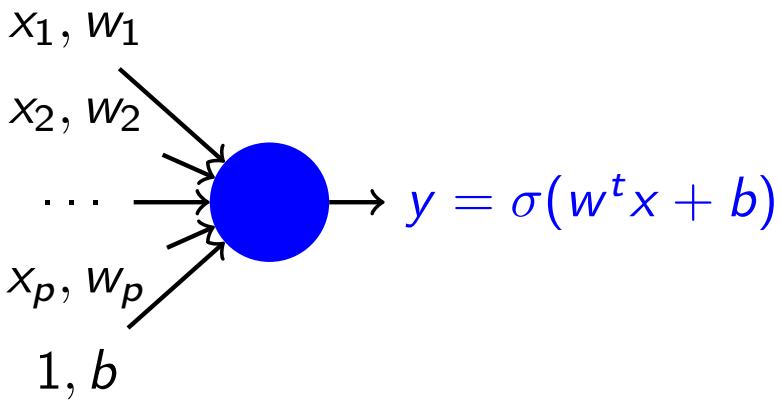
- non linear
- computationally efficient
- differentiable
- non zero

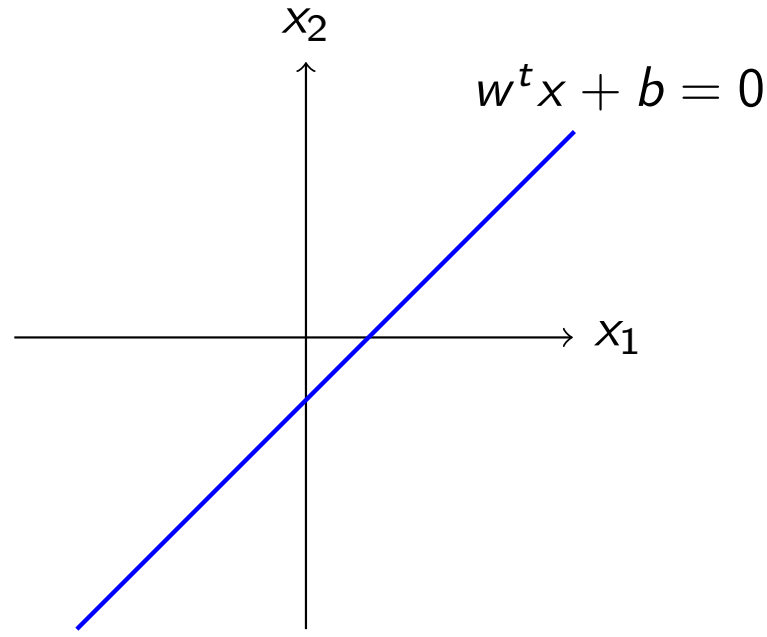
Softmax

$$\sigma_M(x) = \frac{\exp^x}{\sum_k \exp^{x_k}}$$

https://en.wikipedia.org/wiki/Activation_function

The artificial neuron as a linear threshold unit





- x input $\in \mathbb{R}^p$
- w weight, b bias
- a activation, $a = w^t x + b$
- σ activation function
- Φ transfer function
- y output $\in \mathbb{R}$

σ activation function (non linear)

$$\begin{array}{ccc} \mathbb{R} & \mapsto & \mathbb{R} \\ a & \to & y = \sigma(a) \end{array}$$

Φ transfer function

$$\begin{array}{ccc} \mathbb{R}^p & \mapsto & \mathbb{R} \\ \mathbf{x} & \to & y = \Phi(\mathbf{x}) = \sigma(w^t x + b) \end{array}$$

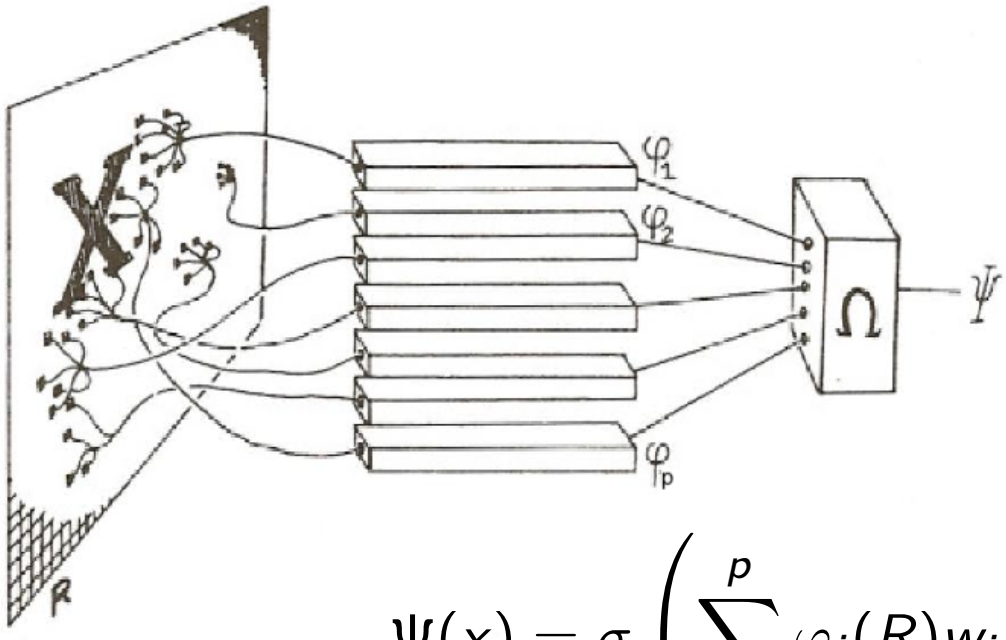
The neural networks time line

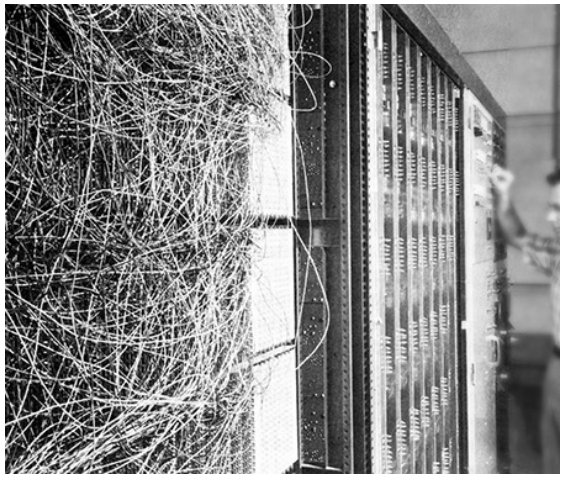
- The first stage: 1890 - 1969

~1890 Ramón y Cajal: the biological neuron
1943 McCulloch & Pitts formal neuron
1949 Hebb's rule
1958 Rosenblatt's Perceptron: learning with stochastic gradient
1969 Minsky & Papert: stop – the 1st NN winter

- The second stage: 1985 - 1995
- The third stage: 2006 - (2012) - 2018...

The formal neuron as a learning machine: fit the w





$$\Psi(x) = \sigma\left(\sum_{j=1}^{p} \varphi_j(R) w_j + b\right)$$

Rosenblatt's Perceptron, 1958 (Widrow & Hoff's Adaline, 1960)

given n pairs of input–output data $\mathbf{x}_i = \varphi_j(R_i), t_i, i = 1, n$

find w such that $\underbrace{\sigma(\mathrm{w}^t \mathbf{x}_i)}_{\text{prediction of the model}} = \underbrace{t_i}_{\text{ground truth}}$

Cost minimization (energy-based model)

Minimize a loss $\min_{\mathrm{w}\in\mathbb{R}^{p+1}} \sum_{i=1}^{n} \mathit{loss}(\mathrm{w})$ $\quad \mathit{loss}(\mathrm{w}) = \left(\sigma(\mathrm{w}^t \mathbf{x}_i) - t_i\right)^2$

Gradient descent $\mathrm{w} \leftarrow \mathrm{w} - \rho\mathbf{d}$ $\qquad \mathbf{d} = \sum_{i=1}^{n} \nabla_{\mathrm{w}} \mathit{loss}(\mathrm{w})$

Stochastic gradient $\qquad \mathbf{d} = \nabla_{\mathrm{w}} \mathit{loss}(\mathrm{w})$

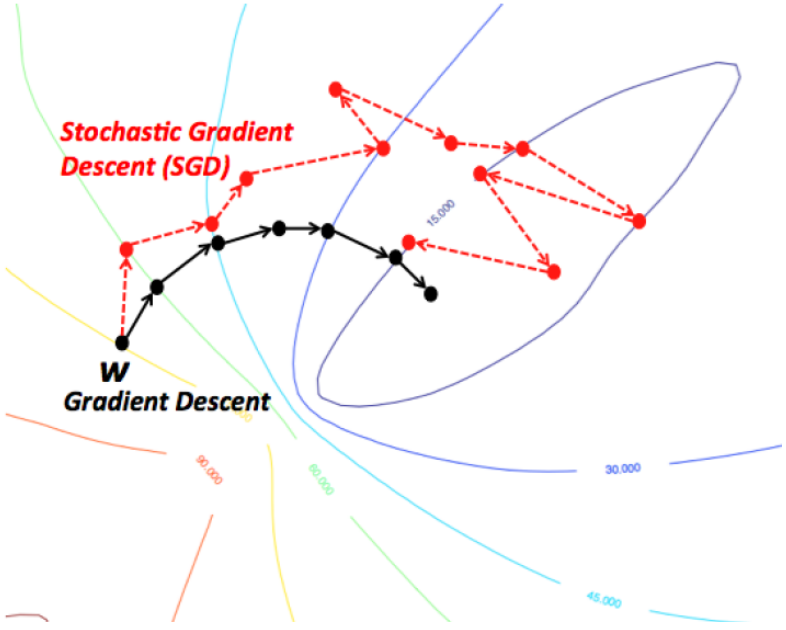
Algorithm 1 Gradient epoch

Data: w initialization, ρ stepsize
Result: w
for *i=1,n* **do**
  $\mathbf{x}_i, t_i \leftarrow$ pick a point i
  $\mathbf{d} \leftarrow d + \nabla_{\mathrm{w}} \mathit{loss}(\mathrm{w}, x_i, t_i)$
end
$\mathrm{w} \leftarrow \mathrm{w} - \rho\mathbf{d}$

Algorithm 2 Stochastic gradient

Data: w initialization, ρ stepsize
Result: w
for *i=1,n* **do**
  $\mathbf{x}_i, t_i \leftarrow$ pick a point i
  $\mathbf{d} \leftarrow \nabla_{\mathrm{w}} \mathit{loss}(\mathrm{w}, x_i, t_i)$
  $\mathrm{w} \leftarrow \mathrm{w} - \rho\mathbf{d}$
end

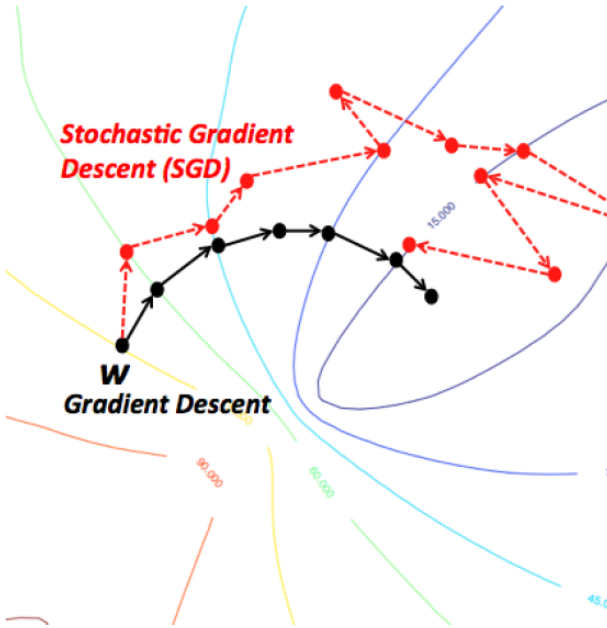
Gradient vs. stochastic gradient



- trading precision for computational efficiency
- minimizing the generalization error (early stopping)
- slower to optimize but faster to learn
- a large scale learning perspective: $\mathcal{O}(n)$

Accelerating the stochastic gradient

- stochastic average (mini batch)
  - parameters (Polyak and Juditsky, 1992)
  - gradients SAG-A, (Le Roux et al 2012)
  - variance reduction (Johnson, Zhang, 2013)
- convergence acceleration
  - Nesterov's method (1983)
  - momentum (heuristic)
- acceleration and averaging
  - (Dieuleveut, Flammarion & Bach, 2016)
- stepsize adaptation
  - RMSprop (Tieleman & Hinton, 2012)
  - Adaptive Moment Estimation – ADAM (Kingma & Ba, 2015)
  - AMSGRAD (Reddi et al, BPA ICRL 2018)



Averaging through examples (mini batch)

For mini batch of size m (typically $m \in \{32, 256\}$, up to 64k!)

$$\mathbf{d} = \sum_{i=1}^{m \ll n} \nabla_{\mathrm{w}} loss(\mathrm{w})$$

Algorithm 3 Mini batch gradient epoch

Data: w initialization, ρ stepsize
Result: w
for i=1,n **do**
  $\mathbf{x}_i, t_i \leftarrow$ pick a point i
  $\mathbf{d} \leftarrow d + \nabla_{\mathrm{w}} loss(\mathrm{w}, x_i, t_i)$
  if $n \mod m == 0$ **then**
    $\mathrm{w} \leftarrow \mathrm{w} - \rho\mathbf{d}$
    $d \leftarrow 0$
  end
end

Averaging through examples (mini batch)

For mini batch of size m (typically $m \in \{32, 256\}$, up to 64k!)

$$\mathbf{d} = \sum_{i=1}^{m \ll n} \nabla_{\mathrm{w}} loss(\mathrm{w})$$

Algorithm 4 Mini batch gradient epoch

Data: w initialization, ρ stepsize
Result: w
for i=1,n **do**
  $\mathbf{x}_i, t_i \leftarrow$ pick a point i
  $\mathbf{d} \leftarrow d + \nabla_{\mathrm{w}} loss(\mathrm{w}, x_i, t_i)$
  if $n \mod m == 0$ **then**
    $\mathrm{w} \leftarrow \mathrm{w} - \rho\mathbf{d}$
    $d \leftarrow 0$
  end
end

with momentum: $d \leftarrow \mu d$

Accelerating stochastic gradient

- convergence acceleration: use velocity $= \mathrm{w}_t - \mathrm{w}_{t-1}$
  - stochastic gradient with momentum (MOM) $\mu = .9$:

    $$d = \mu d + \nabla_w L(\mathrm{w})$$
    $$\mathrm{w} = \mathrm{w} - \rho \mathbf{d}$$

  - Nesterov accelerated gradient (NAG, 1982) $\mu = .5$:

    $$g_t = w - \rho \nabla_w L(\mathrm{w})$$
    $$\mathrm{w} = g_t - \mu(g_t - g_{t-1})$$

- stepsize adaptation
  - Adaptive stepsize gradient $\varepsilon = 10^{-7}$

    $$\rho = \frac{\rho \mathbf{0}}{\|\mathbf{d}\| + \varepsilon}$$

  - RMSprop (Tieleman and Hinton, 2012) $\alpha = 0.9$

    $$g = \alpha g + (1 - \alpha)\mathbf{d}^\top \mathbf{d}$$
    $$\rho = \frac{\rho \mathbf{0}}{\sqrt{g} + \varepsilon}$$

  - Adam (Kingma and Ba, 2015) $\alpha_1 = 0.9, \alpha_2 = 0.999$

    $$h = \alpha_1 h + (1 - \alpha)\mathbf{d}$$
    $$g = \alpha_2 g + (1 - \alpha)\mathbf{d}^\top \mathbf{d}$$
    $$\mathrm{w} = \mathrm{w} - \frac{\rho \mathbf{0}}{\sqrt{g} + \varepsilon} h$$

Optimizers available with Keras

Parameters common to all Keras optimizers

The parameters `clipnorm` and `clipvalue` can be used with all optimizers to control gradient clipping:

```
from keras import optimizers

# All parameter gradients will be clipped to
# a maximum norm of 1.
sgd = optimizers.SGD(lr=0.01, clipnorm=1.)
```

```
from keras import optimizers

# All parameter gradients will be clipped to
# a maximum value of 0.5 and
# a minimum value of -0.5.
sgd = optimizers.SGD(lr=0.01, clipvalue=0.5)
```

SGD [source]

```
keras.optimizers.SGD(lr=0.01, momentum=0.0, decay=0.0, nesterov=False)
```

Stochastic gradient descent optimizer.

Includes support for momentum, learning rate decay, and Nesterov momentum.

Arguments

- **lr**: float >= 0. Learning rate.
- **momentum**: float >= 0. Parameter that accelerates SGD in the relevant direction and dampens oscillations.
- **decay**: float >= 0. Learning rate decay over each update.
- **nesterov**: boolean. Whether to apply Nesterov momentum.

RMSprop [source]

```
keras.optimizers.RMSprop(lr=0.001, rho=0.9, epsilon=None, decay=0.0)
```

RMSProp optimizer.

loss functions and gradients

stochastic gradient descent $\quad \mathrm{w} \leftarrow \mathrm{w} - \rho \mathbf{d} \quad \mathbf{d} = \sum_{i=1}^{m} \nabla_{\mathrm{w}} \mathit{loss}(\mathrm{w})$

t target values (expected output)

y output of the perceptron

Square error: $\quad \mathit{loss}(\mathbf{y}, t) = \frac{1}{2}(y - t)^2$

$\nabla_{\mathbf{y}} \mathit{loss}(\mathrm{w}) = (y - t)$

Cross entropy: $t \in \{0, 1\} \quad \mathit{loss}(\mathrm{w}) = -t \log y - (1 - t) \log(1 - y)$

$\nabla_{\mathbf{y}} \mathit{loss}(\mathrm{w}) = \dfrac{y - t}{y(1 - y)}$

loss functions available with Keras


```
from keras import losses

model.compile(loss=losses.mean_squared_error, optimizer='sgd')
```

You can either pass the name of an existing loss function, or pass a TensorFlow/Theano symbolic function that returns a scalar for each data-point and takes the following two arguments:

- **y_true**: True labels. TensorFlow/Theano tensor.
- **y_pred**: Predictions. TensorFlow/Theano tensor of the same shape as y_true.

The actual optimized objective is the mean of the output array across all datapoints.

For a few examples of such functions, check out the losses source.

Available loss functions

mean_squared_error

```
mean_squared_error(y_true, y_pred)
```

mean_absolute_error

```
mean_absolute_error(y_true, y_pred)
```

mean_absolute_percentage_error

```
mean_absolute_percentage_error(y_true, y_pred)
```

mean_squared_logarithmic_error

```
mean_squared_logarithmic_error(y_true, y_pred)
```

squared_hinge

```
squared_hinge(y_true, y_pred)
```

hinge

```
hinge(y_true, y_pred)
```

Linear neuron and the gradient

- propagation

  input $x \in \mathbb{R}^p$

  activation $a = w^\top x + b \in \mathbb{R}$

  output $y = \sigma(a)$

  loss $J(x) = L(y, t)$

- gradient update: $w = w - \rho \nabla_w L$
  - chain rule: $\nabla_w J(x) = \dfrac{\partial L(y,t)}{\partial y} \dfrac{\partial y}{\partial a} \nabla_w a =$ back propagation

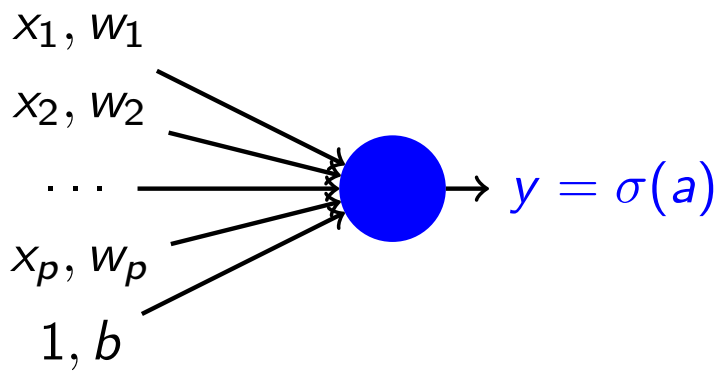
$$\frac{\partial L(y,t)}{\partial w_i} = \frac{\partial L(y,t)}{\partial y} \frac{\partial y}{\partial a} \frac{\partial a}{\partial w_i}$$

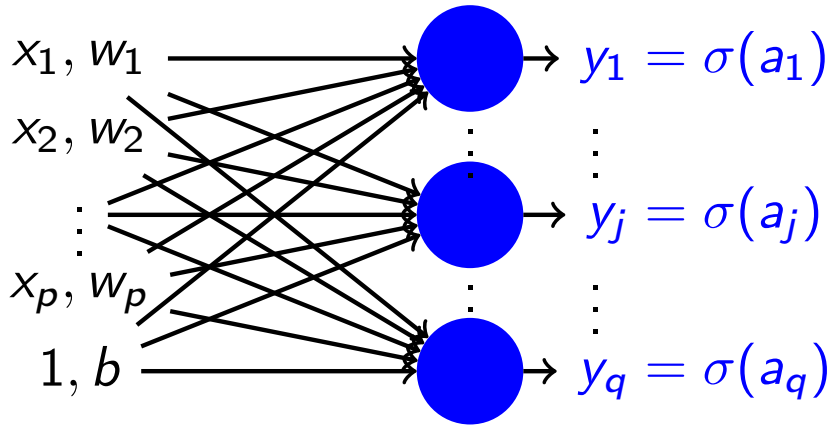
    - ★ $\partial_y L$
    - ★ $\partial_a y = \sigma'(a)$
    - ★ $\nabla_w a = x$

  - for the square loss: $w = w - \rho(y - t)\sigma'(a)x$

- gradient with respect to inputs: $\nabla_x J(x) = (y - t)\sigma'(a)w$

From one neuron to many (q)





input	$\mathbf{x} \in \mathbb{R}^p$
activation	$a = w^\top \mathbf{x} + b \in \mathbb{R}$
weights	$\mathrm{w} \in \mathbb{R}^p$
output	$y = \sigma(a)$
loss	$J(\mathbf{x}) = L(y, t)$

input	$\mathbf{x} \in \mathbb{R}^p$
activation	$\mathbf{a} = W\mathbf{x} + b \in \mathbb{R}^q$
weights	$W \in \mathbb{R}^{q \times p}$
output	$\mathbf{y} = \sigma(\mathbf{a})$ q output
loss	$J(\mathbf{x}) = L(\mathbf{y}, t)$
	$L(\mathbf{y}, t) = \sum_{j=1}^{q} L(y_j, t_j)$

Linear neurons and the gradient

- propagation

input	$x \in \mathbb{R}^p$	
activation	$a = Wx + b \in \mathbb{R}^q$	q neurons
weights	$W \in \mathbb{R}^{q \times p}$	a matrix of synaptic weights
output	$y = \sigma(a)$	q output
loss	$J(x) = L(y, t)$	$L(y, t) = \sum_{j=1}^{q} L(y_j, t_j)$

- gradient update: $W = W - \rho \nabla_w L$
  - chain rule: $\nabla_w J(x) = \sum_{j=1}^{q} \frac{\partial L(y_j, t_j)}{\partial y_j} \frac{\partial y_j}{\partial a_j} \nabla_w a_j$ = back propagation

$$\frac{\partial L(y, t)}{\partial w_{ij}} = \frac{\partial L(y_j, t_j)}{\partial y_j} \frac{\partial y_j}{\partial a_j} \frac{\partial a_j}{\partial w_{ij}}$$

  - for the square loss:
    $W = W - \rho \underbrace{\mathbf{v}\, x^\top}_{\text{outer product}}$ with $\mathbf{v} = (y - t).\sigma'(a)$

- gradient with respect to inputs : $\nabla_x J(x) = W^\top \mathbf{v}$

The neural networks time line

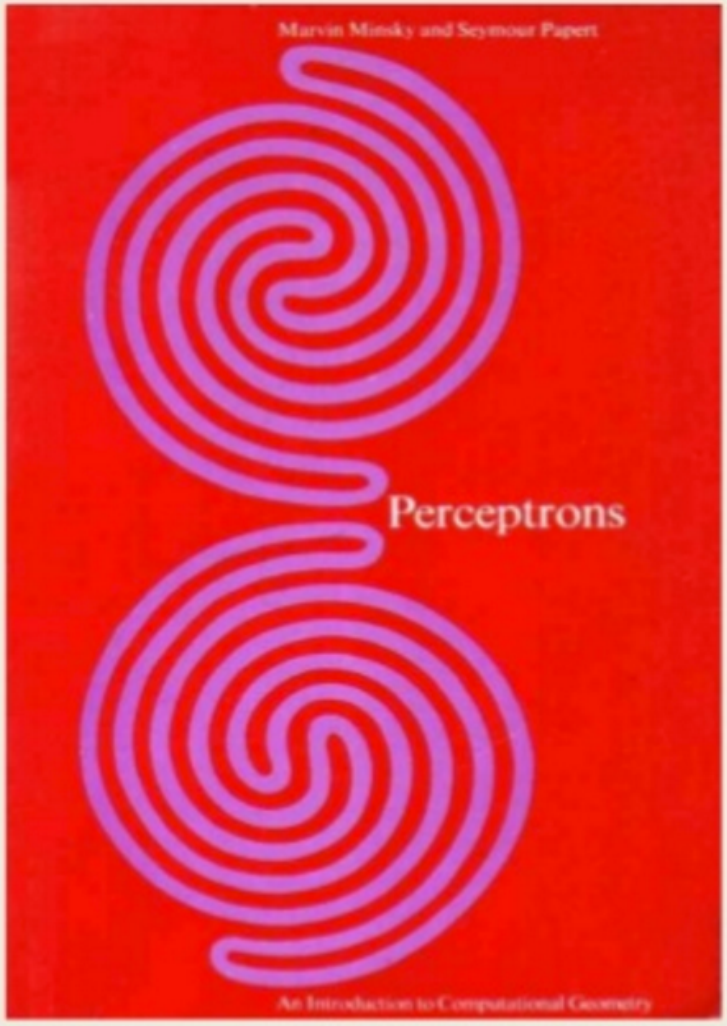
- The first stage: 1890 - 1969

∼1890 Ramón y Cajal: the biological neuron
1943 McCulloch & Pitts formal neuron
1949 Hebb's rule
1958 Rosenblatt's Perceptron: learning with stochastic gradient
1969 Minsky & Papert: stop – the 1st NN winter

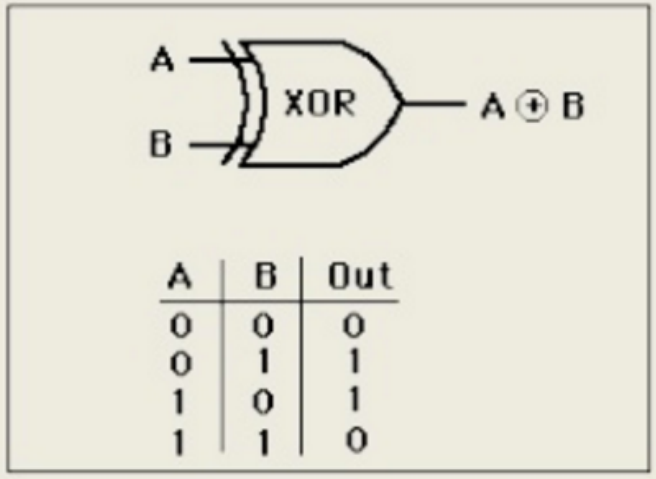
- The second stage: 1985 - 1995
- The third stage: 2006 - (2012) - 2018...

However, linear neurons are linear

1969: Perceptrons can't do XOR!



http://www.i-programmer.info/images/stories/BabBag/AI/book.jpg



A	B	Out
0	0	0
0	1	1
1	0	1
1	1	0

http://hyperphysics.phy-astr.gsu.edu/hbase/electronic/ietron/xor.gif

Minsky & Papert

https://constructingkids.files.wordpress.com/2013/05/minsky-papert-71-csolomon-x640.jpg

Limitations of the Perceptron

- Linear threshold units as Boolean gates
- Circuit theory is poorly known
- Learning deep circuits means solving the credit assignment pb
- Linearly separable problems are few
- Elementary problems need complex circuits. (parity, x-or...)
- But have simple algorithmic solutions
  $\rightarrow$programming versus learning

Abandon perceptrons and other analog computers

Develop symbolic computers and symbolic AI techniques.

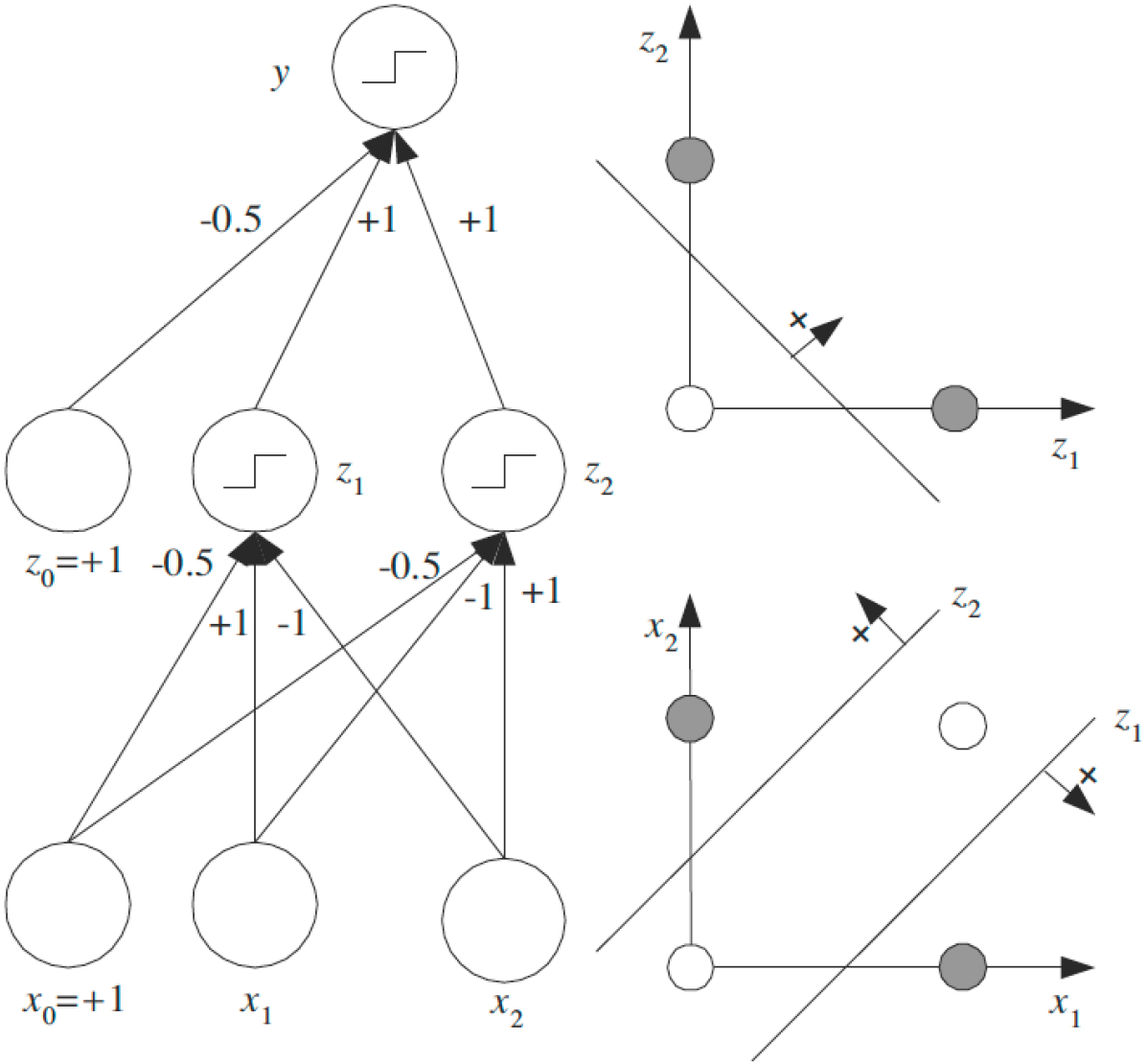
The neural networks time line

- The first stage: 1890 - 1969
- The second stage: 1985 - 1995

1985 Rumelhart, Hinton & Williams; Le Cun: go - backpropagation
1989 Universal Approximation Cybenko-Hornik-Funahashi Theorem
1989 Y. Le Cun's convolutional neural networks
1995 Recurrent neural networks, LSTM
1995 SVM
2004 Caltech 101: the 2nd NN winter

- The third stage: 2006 - (2012) - 2018...

Non linearity combining linear neurons: the Xor case



Alpaydın, Introduction to Machine Learning, 2010

Neural networks

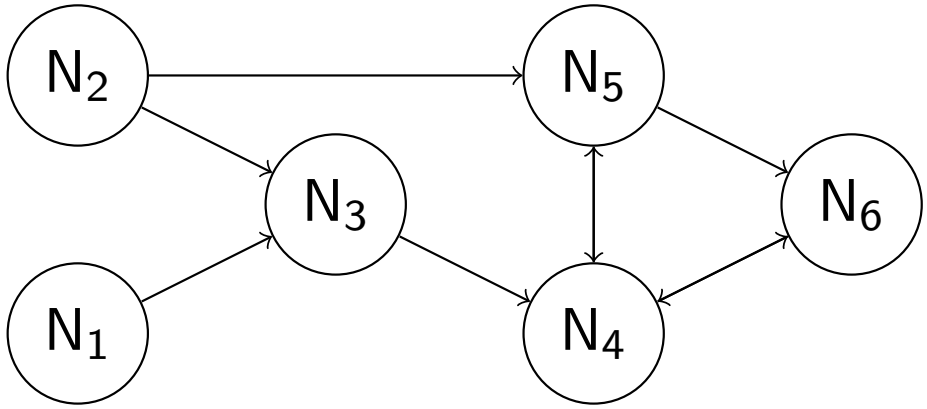
Definition: Neural network

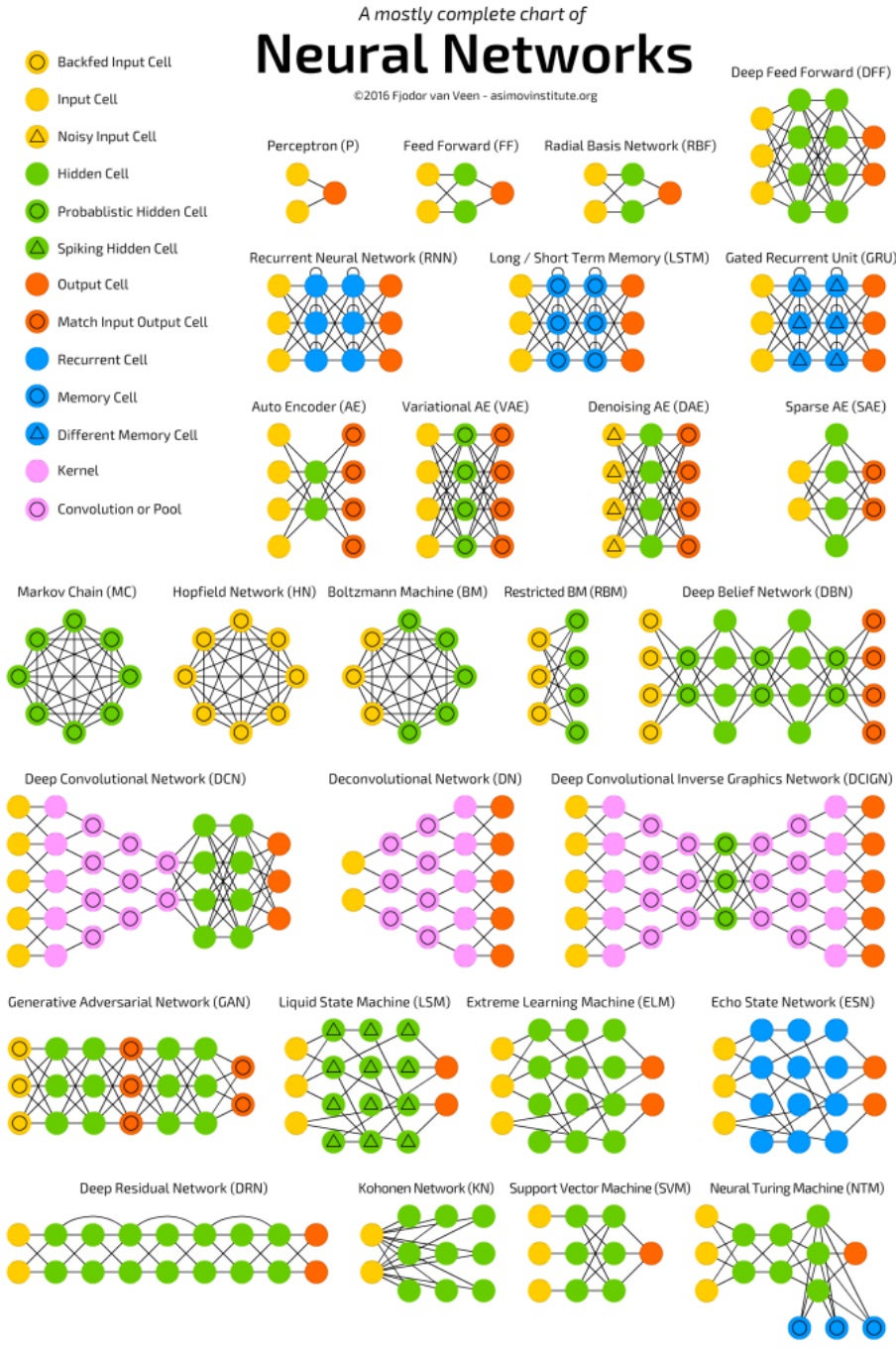
A neural network is an oriented graph of formal neurons

When two neurons are connected (linked by an oriented edge of the graph), the output of the head neuron is used as an input by the tail neuron. It can be seen as a weighted directed graph.

3 different neurons are considered:

- input neurons (connected with the input)
- output neurons
- hidden neurons





The Asimov Institute: `http://www.asimovinstitute.org/neural-network-zoo/`

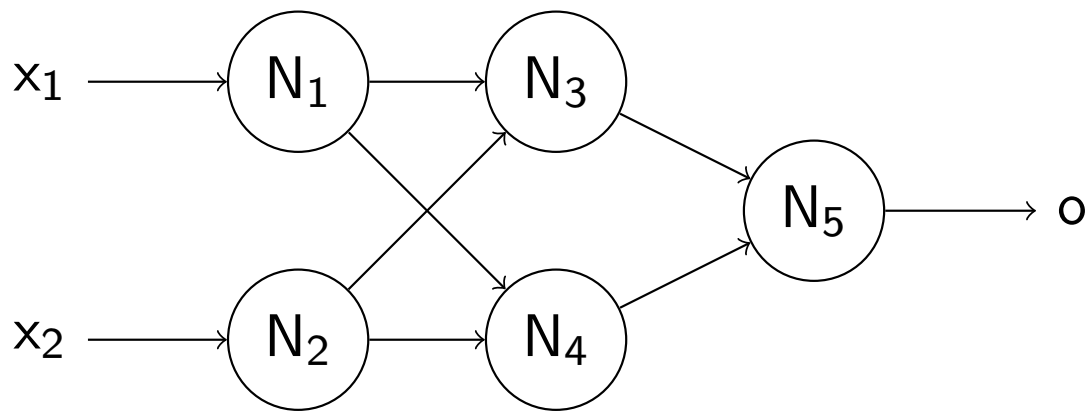
The Multilayer peceptron (MLP)

Definition: Multilayer peceptron

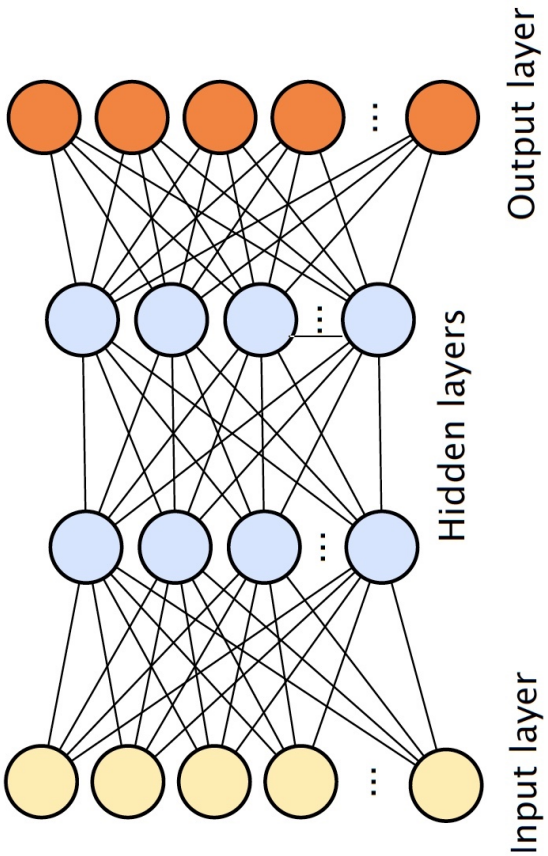
A Multilayer peceptron is an acyclic neural network,

where the neurons are structued in successive layers, begining by an input layer and finishing with an output layer.

Example: The X-or neural network is a MLP with a single hidden unit with 2 hidden neurons.



MLP training with back propagation (and SGD)



$$y = \sigma(W_3 h^{(2)}) \qquad \nabla_{W_3} J = (y - y_a)\sigma'(W_3 h^{(2)})h^{(2)}$$

$$\uparrow \qquad\qquad \downarrow$$

$$h^{(2)} = \sigma(W_2 h^{(1)}) \qquad \nabla_{W_2} J = \nabla_{h^{(2)}} J \sigma'(W_2 h^{(1)})h^{(1)}$$

$$\uparrow \qquad\qquad \downarrow$$

$$h^{(1)} = \sigma(W_1 \mathbf{x}) \qquad \nabla_{W_1} J = \nabla_{h^{(1)}} J \sigma'(W_1 x)x^\top$$

$$\uparrow$$

$$\mathbf{x}$$

$$y = \sigma\Big(W_3 \sigma\big(W_2 \sigma(W_1 \mathbf{x})\big)\Big)$$

backpropagation = chain rule (autodiff)

Used to learn internal representation W_1, W_2, W_3

Back propagation is differential learning

Yann LeCun

5 janvier ·

OK, Deep Learning has outlived its usefulness as a buzz-phrase.
Deep Learning est mort. Vive Differentiable Programming!

Numpy

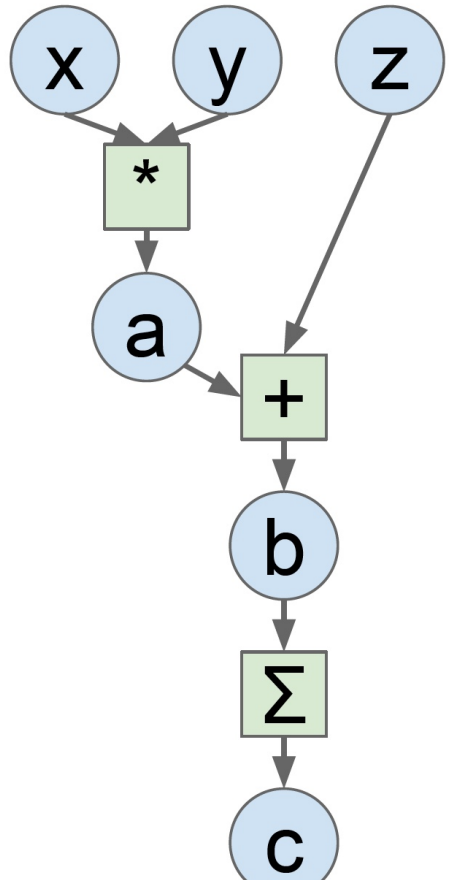
```
import numpy as np
np.random.seed(0)

N, D = 3, 4

x = np.random.randn(N, D)
y = np.random.randn(N, D)
z = np.random.randn(N, D)

a = x * y
b = a + z
c = np.sum(b)

grad_c = 1.0
grad_b = grad_c * np.ones((N, D))
grad_a = grad_b.copy()
grad_z = grad_b.copy()
grad_x = grad_a * y
```



The neural networks time line

- The first stage: 1890 - 1969
- The second stage: 1985 - 1995

  1985 Rumelhart, Hinton & Williams; Le Cun: go - backpropagation
  1989 Universal Approximation Cybenko-Hornik-Funahashi Theorem
  1989 Y. Le Cun's convolutional neural networks
  1995 Recurrent neural networks, LSTM
  1995 SVM
  2004 Caltech 101: the 2nd NN winter

- The third stage: 2006 - (2012) - 2018...

Two theoretical results about MLP

Universal approximation theorem for one hidden layer MLP

- given any $\varepsilon > 0$
- for any continuous function f on compact subsets of $\mathbb{R}^p$
- for any admissible activation function σ (not a polynomial)
- there exists h, $W_1 \in \mathbb{R}^{p \times h}$, $b \in \mathbb{R}^h$, $c \in \mathbb{R}$ and $w_2 \in \mathbb{R}^h$ such that

$$\|f(x) - w_2\sigma(W_1 x + b) + c\|_\infty \leq \varepsilon$$

Approximation theory of the MLP model in neural networks, A Pinkus - Acta Numerica, 1999

SVM, Boosting and Random Forest also are unversal approximators

Why two hidden layers can be better than one?

There exists a function on $\mathbb{R}^p$, expressible by a small two hidden layer MLP, which cannot be approximated by any two hidden layer MLP, to more than a certain constant accuracy, unless its width is exponential in the p.

The power of depth for feedforward neural networks, R. Eldan and O. Shamir, 2015.

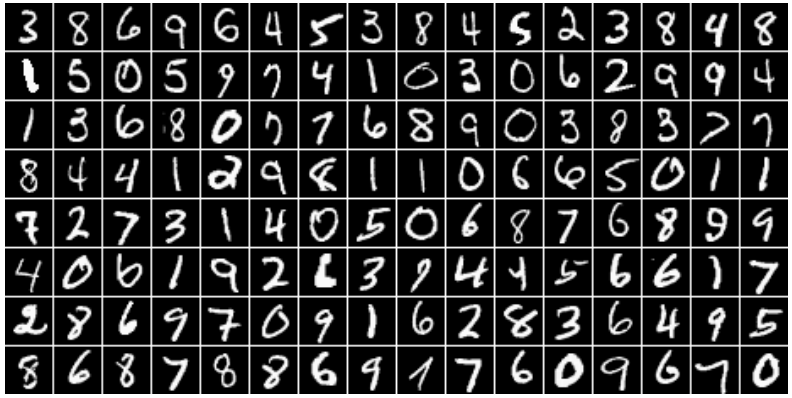
The neural networks time line

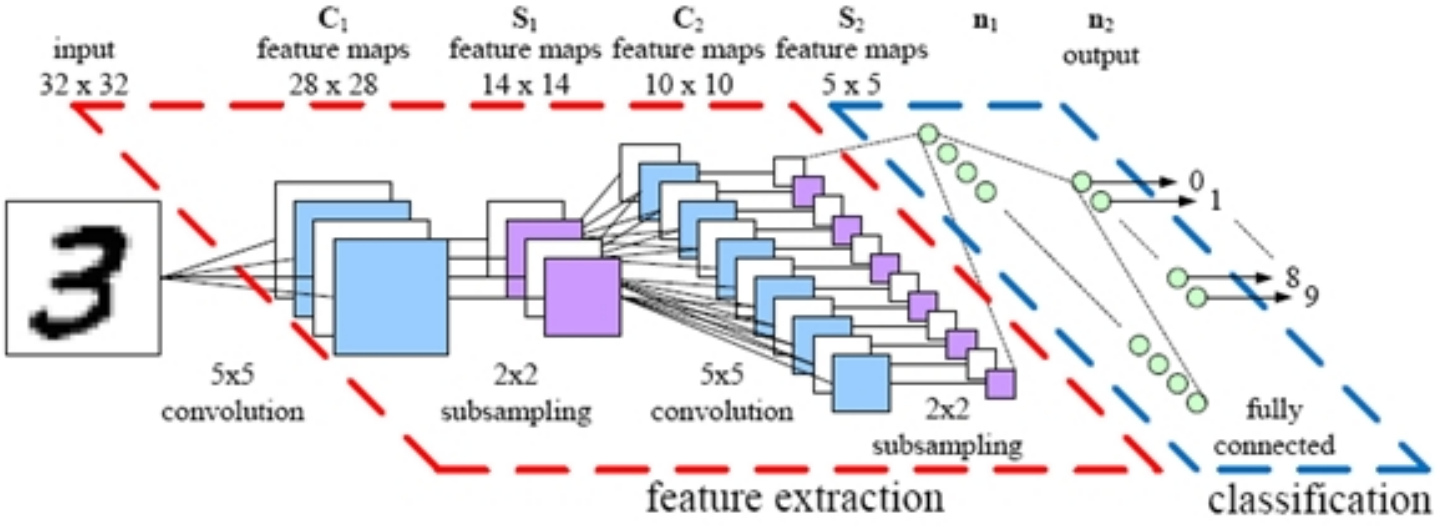
- The first stage: 1890 - 1969
- The second stage: 1985 - 1995

  1985 Rumelhart, Hinton & Williams; Le Cun: go - backpropagation
  1989 Universal Approximation Cybenko-Hornik-Funahashi Theorem
  1989 Y. Le Cun's convolutional neural networks
  1995 Recurrent neural networks, LSTM
  1995 SVM
  2004 Caltech 101: the 2nd NN winter

- The third stage: 2006 - (2012) - 2018...

OCR: MNIST database (LeCun, 1989)





use convolution layers

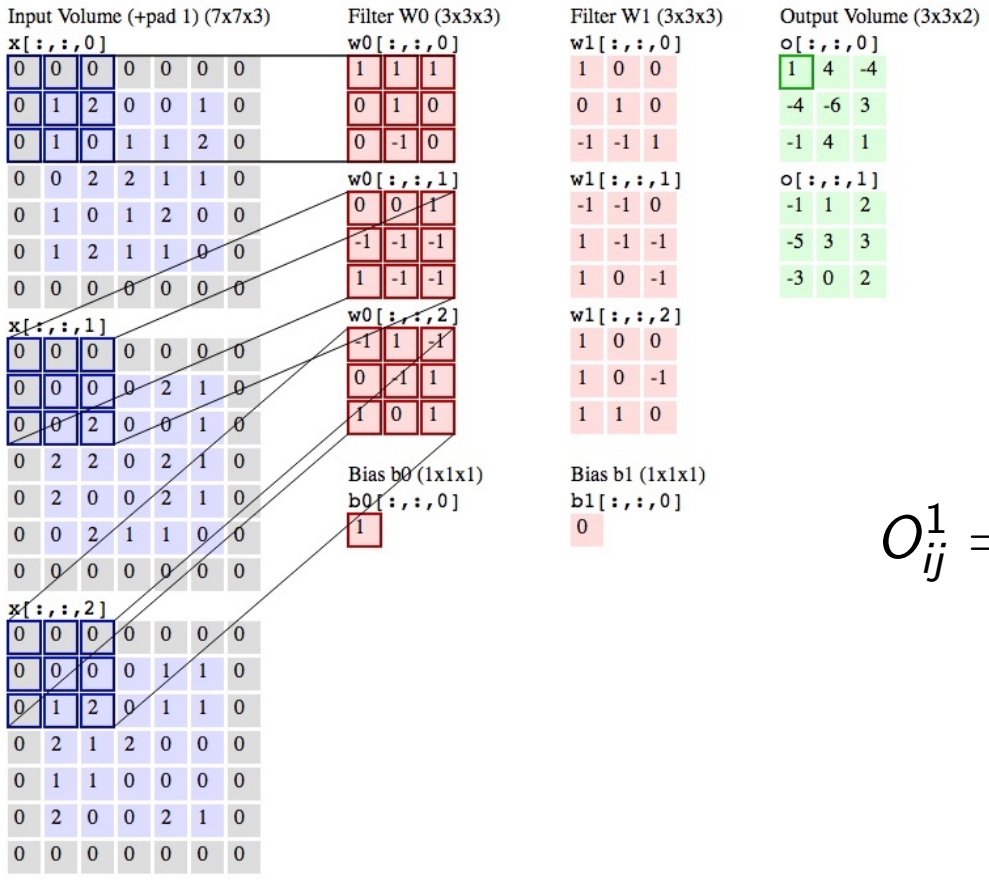
Y. LeCun,et al. Gradient-Based Learning Applied to Document Recognition, 1998

A convolution layer in a neural network

R

G

B



$$O^1_{ij} = \sum_{k,\ell,m=1}^{3} w^1_{k,\ell,m} I_{2i+k,2j+\ell,m} + b_1$$

Convolution reduces the number of parameters

http://cs231n.github.io/convolutional-networks/

CNN results on MNIST in 1998 (historical debate)

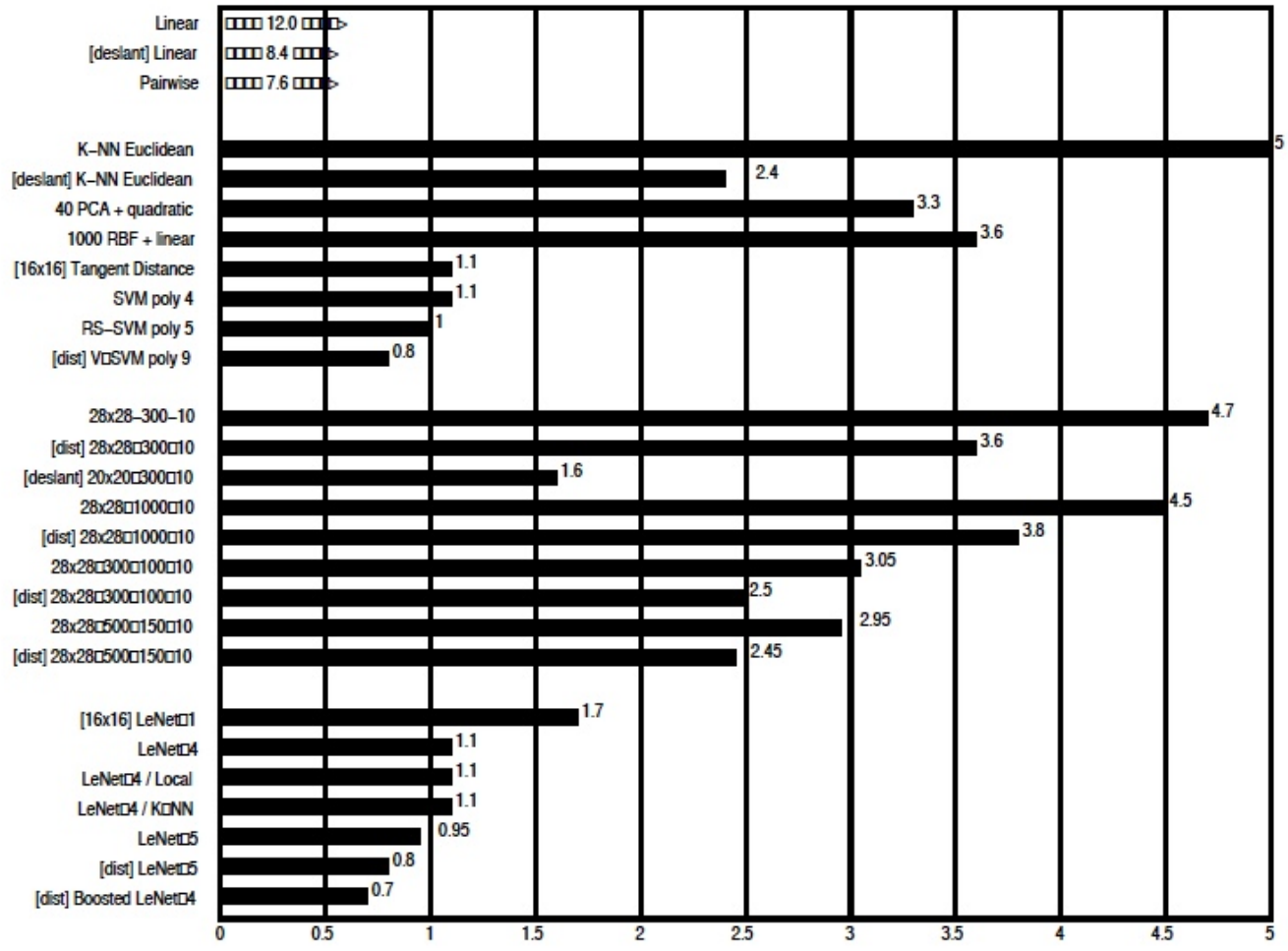


Fig. 9. Error rate on the test set (%) for various classification methods. [deslant] indicates that the classifier was trained and tested on the deslanted version of the database. [dist] indicates that the training set was augmented with artificially distorted examples. [16x16] indicates that the system used the 16x16 pixel images. The uncertainty in the quoted error rates is about 0.1%.

http://yann.lecun.com/exdb/mnist/

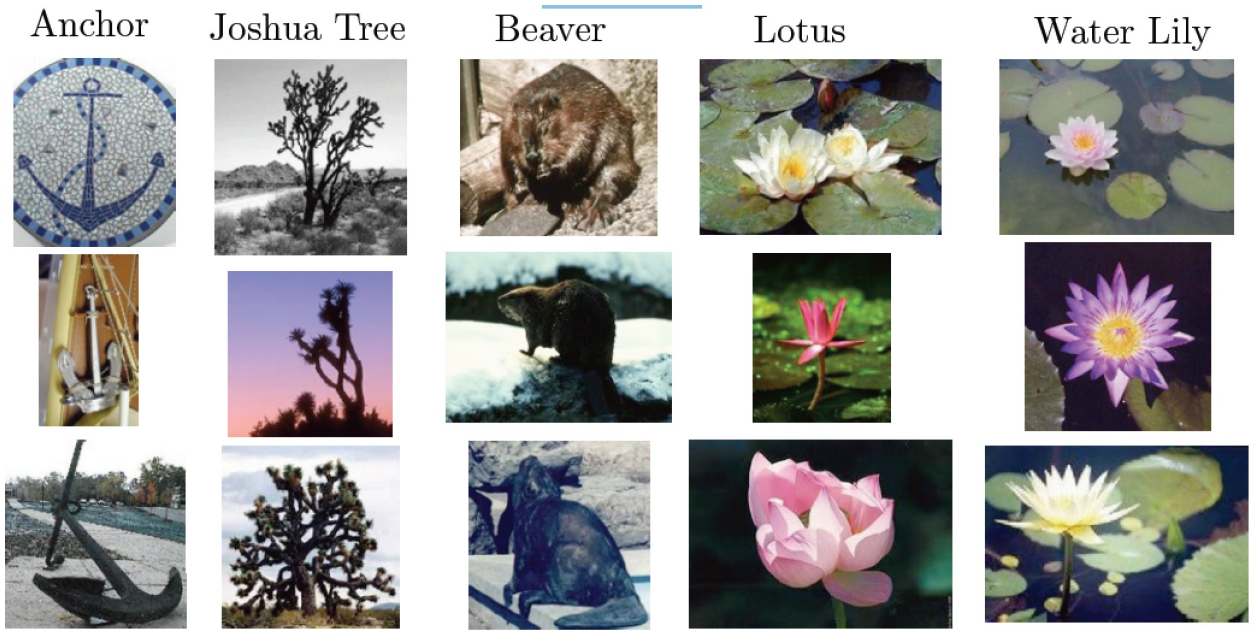
The neural networks time line

- The first stage: 1890 - 1969
- The second stage: 1985 - 1995

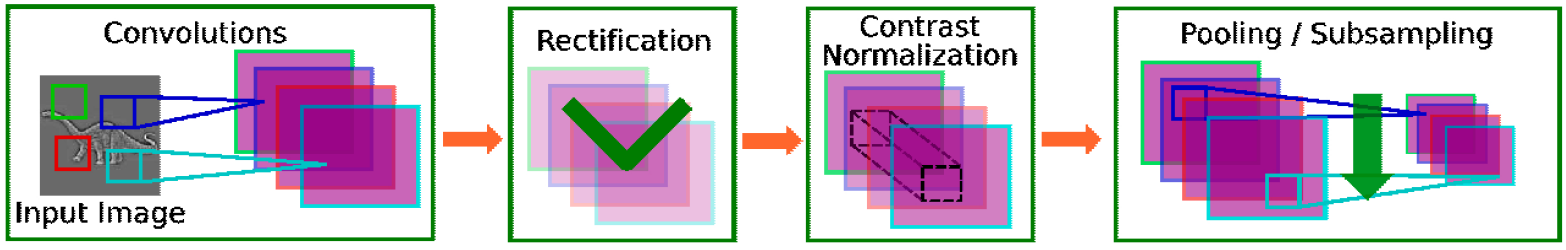
1985 Rumelhart, Hinton & Williams; Le Cun: go - backpropagation
1989 Universal Approximation Cybenko-Hornik-Funahashi Theorem
1989 Y. Le Cun's convolutional neural networks
1995 Recurrent neural networks, LSTM
1995 SVM
2004 Caltech 101: the NN winter

- The third stage: 2006 - (2012) - 2018...

The caltech 101 database (2004)



- 101 classes,
- 30 training images per category
- ...and the winner is NOT a deep network
  - dataset is too small



use convolution + Rectification + Normalization + Pooling

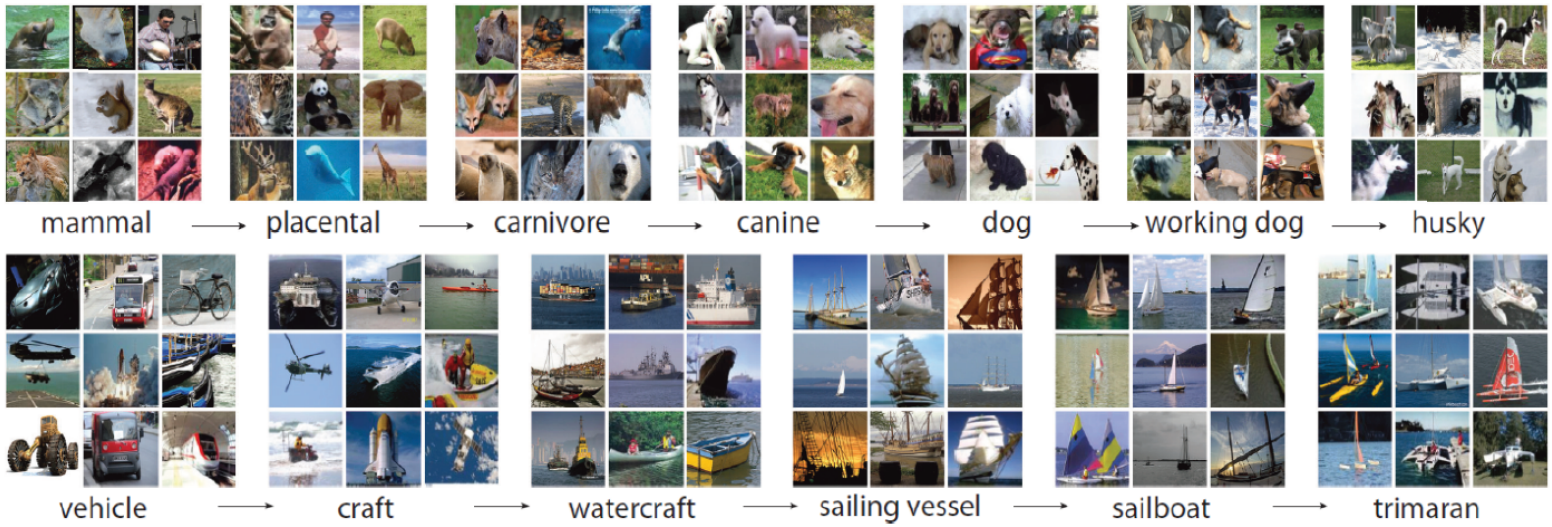
in What is the Best Multi-Stage Architecture for Object Recognition? Jarrett et al, 2009

The neural networks time line

- The first stage: 1890 - 1969
- The second stage: 1985 - 1995
- The third stage: 2006 - (2012) - 2018...

2006 Deep learning: Bengio's, Hinton's RBM, Y LeCun's proposals
2010 Andrew Ng's GPU for Deep GPU
2011 Deep frameworks, tools (theano, torch, cuda-convnet...)
2012 ImageNet – AlexNet
2013 M. Zuckerberg at NIPS the deep fashion
2014 Representation learning fine tuning
2015 Deep learning in the industry: speech, traduction, image...
2016 Goodfellow's generative adversarial networks (GAN)
2017 Reinforcement learning: Deep win's GO
2018 Automatic design, adversarial defense, green learning, theory...

The ImageNet database (Deng et al., 2012)



ImageNet = 15 million high-resolution images of 22,000 categories.
Large-Scale Visual Recognition Challenge (a subset of ImageNet)

- 1000 categories.
- 1.2 million training images,
- 50,000 validation images,
- 150,000 testing images.

www.image-net.org/challenges/LSVRC/2012/

A new fashion in image processing

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

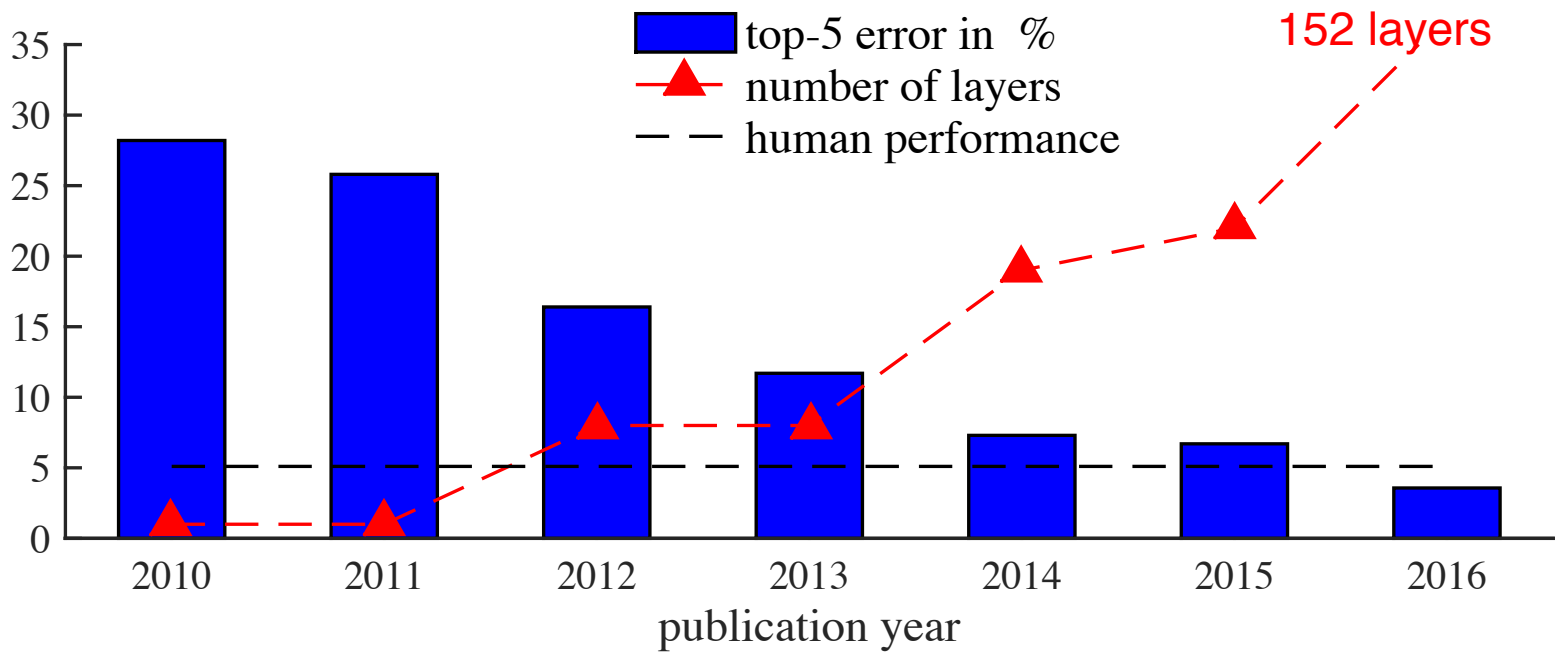
2013 Teams	%error
Clarifai (NYU spinoff)	11.7
NUS (singapore)	12.9
Zeiler-Fergus (NYU)	13.5
A. Howard	13.5
OverFeat (NYU)	14.1
UvA (Amsterdam)	14.2
Adobe	15.2
VGG (Oxford)	15.2
VGG (Oxford)	23.0

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

shallow approaches

deep learning

ImageNet results



2012 Alex Net

2013 ZFNet

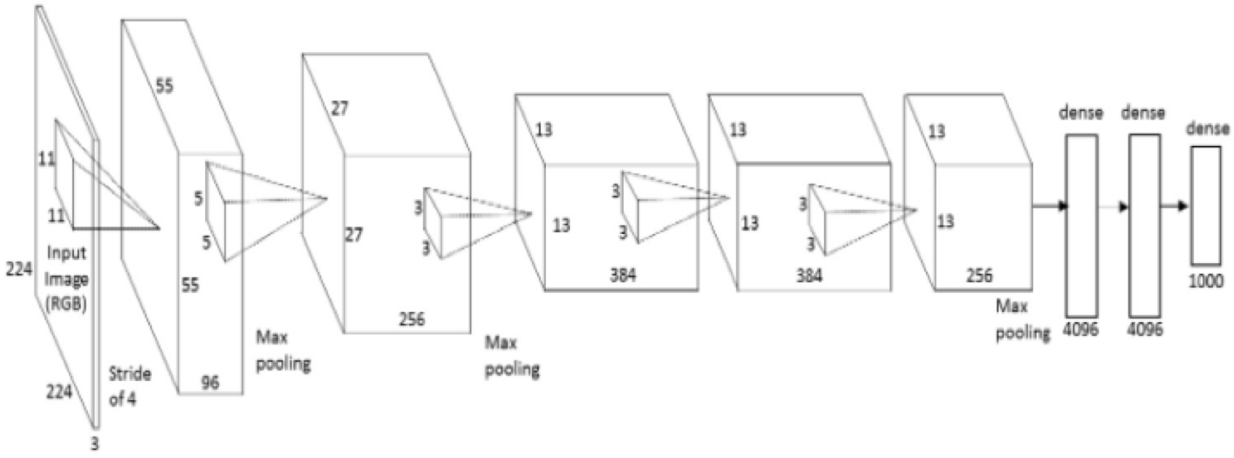
2014 VGG

2015 GoogLeNet / Inception

2016 Residual Network

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

Deep architecture for ImageNet (15%)

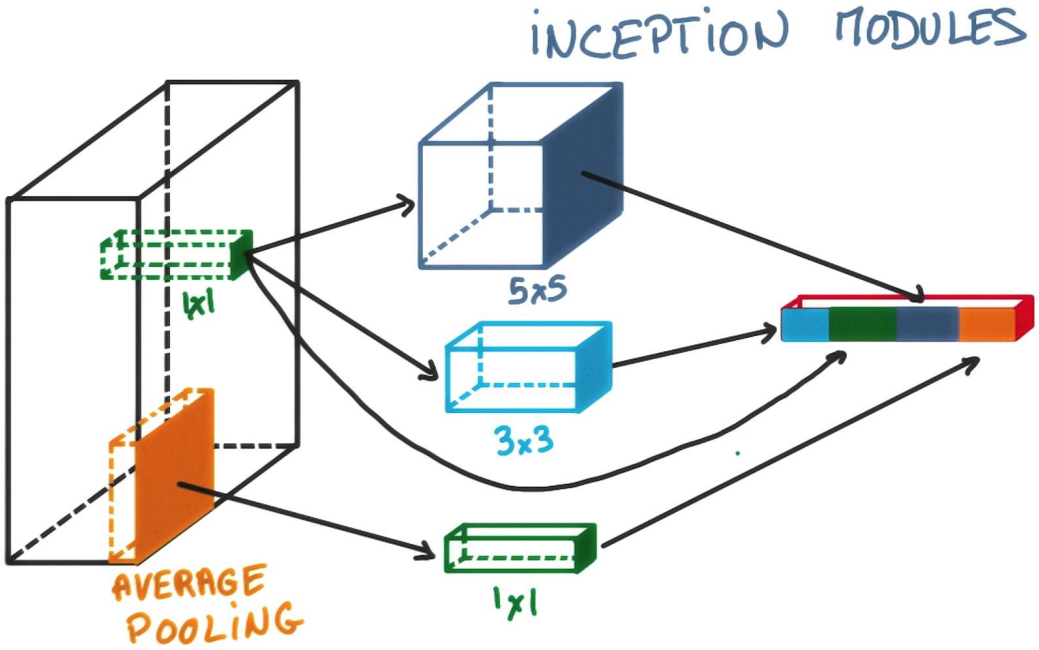


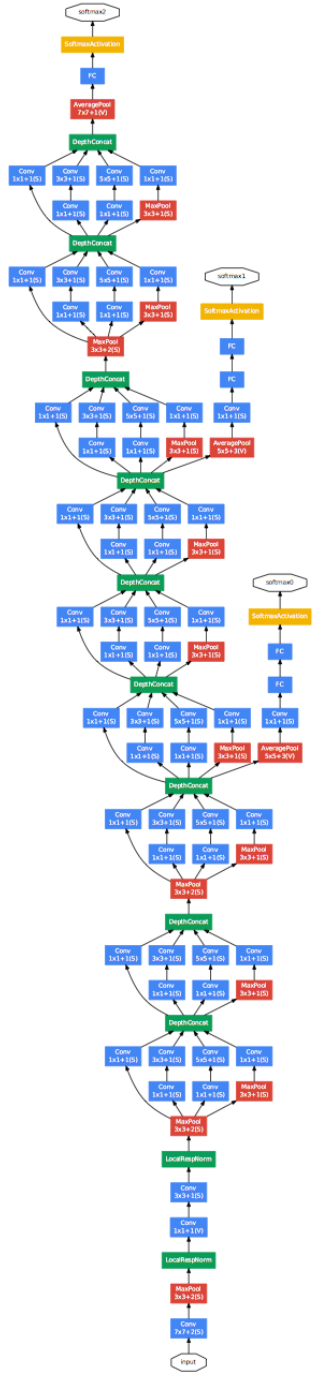
The *AlexNet* architecture [Krizhevsky, Sutskever, Hinton, 2012]

Convolution + Rectification (ReLU) + Normalization + Pooling

- 60 million parameters
- using 2 GPU – 6 days
- regularization
  - data augmentation
  - dropout
  - weight decay

From 15% to 7%: Inceptionism

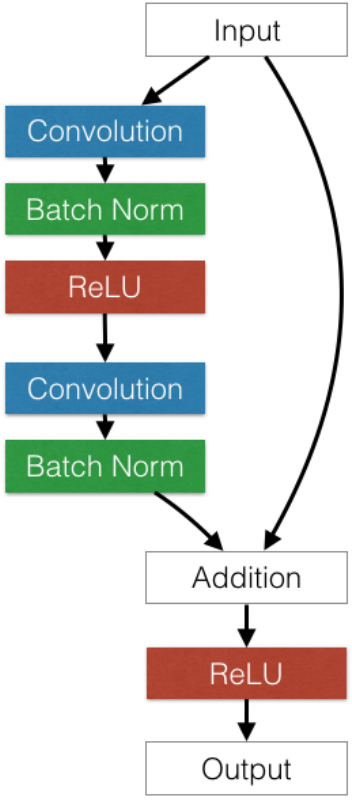


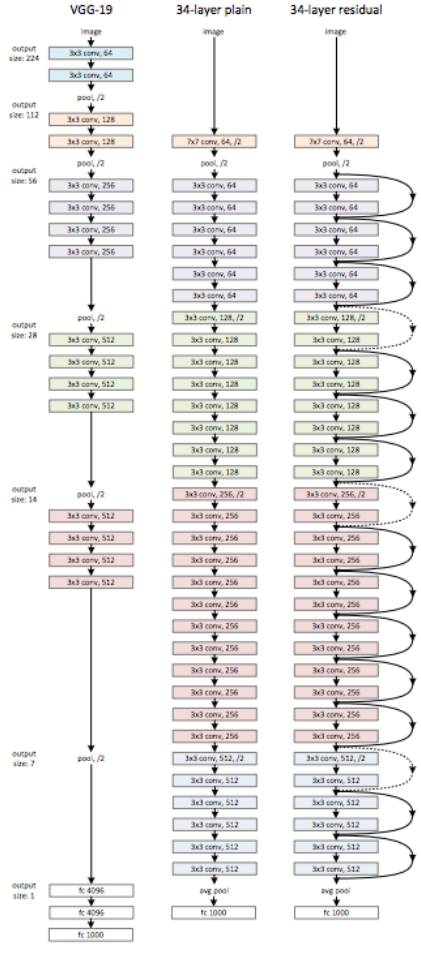


Network in a network (deep learning lecture at Udacity)

Christian Szegedy et. al. Going deeper with convolutions. CVPR 2015.

From 7% to 3%: Residual Nets





Beating the gradient vanishing effect

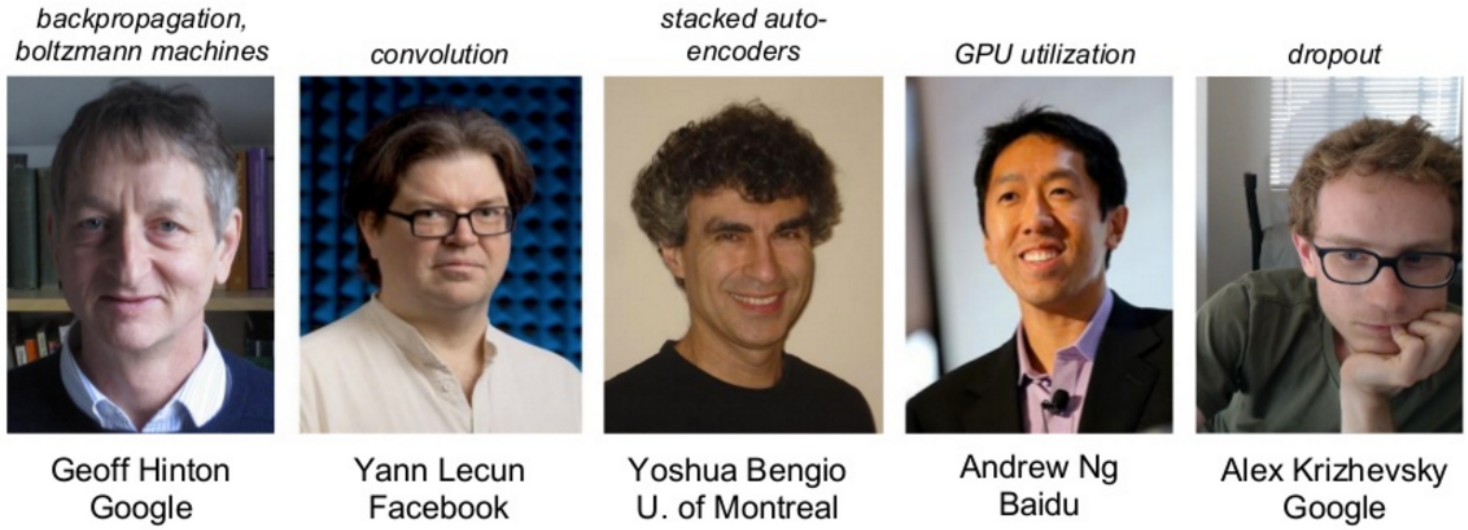
K. He *et al*, 2016

The neural networks time line

- The first stage: 1890 - 1969
- The second stage: 1985 - 1995
- The third stage: 2006 - (2012) - 2018...

2006 Deep learning: Bengio's, Hinton's RBM, Y LeCun's proposals
2010 Andrew Ng's GPU for Deep GPU
2011 Deep frameworks, tools (theano, torch...)
2012 ImageNet – AlexNet
2013 M. Zuckerberg at NIPS: the deep fashion
2014 Representation learning fine tuning
2015 Deep learning in the industry: speech, traduction, image...
2016 Goodfellow's generative adversarial networks (GAN)
2017 Reinforcement learning: Deep win's GO
2018 Automatic design, adversarial defense, green learning, theory...

Deep learning, AI and the industry



- data science, artificial intelligence and deep learning
- the GAFAM – BATX vision
  - they got the infrastructure (hard+software)
  - they got the data
  - deep learning bridges the gap between applications and ML

In 2016, Google Chief Executive Officer (CEO) Sundar Pichai said,

Machine learning [a subfield of AI] is a core, transformative way by which we're rethinking how we're doing everything.

So far so good

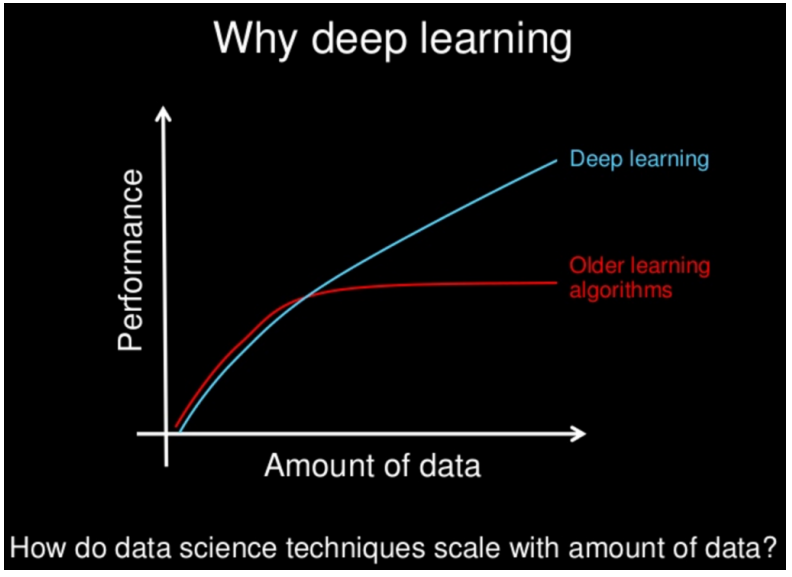
- from the formal neuron to deep learning
  - one neuron is a linear perceptron
  - many layered neurons are non linear multilayer perceptrons
  - deep networks is a new name for multilayer perceptrons
- deep learning breakthrough starts with ImageNet
  - better than human performances
  - on many perception tasks
- deep learning could transform almost any industry
  - the AI revolution

Neural networks+backpropagation exist since 1985

$\rightarrow$ what's new?

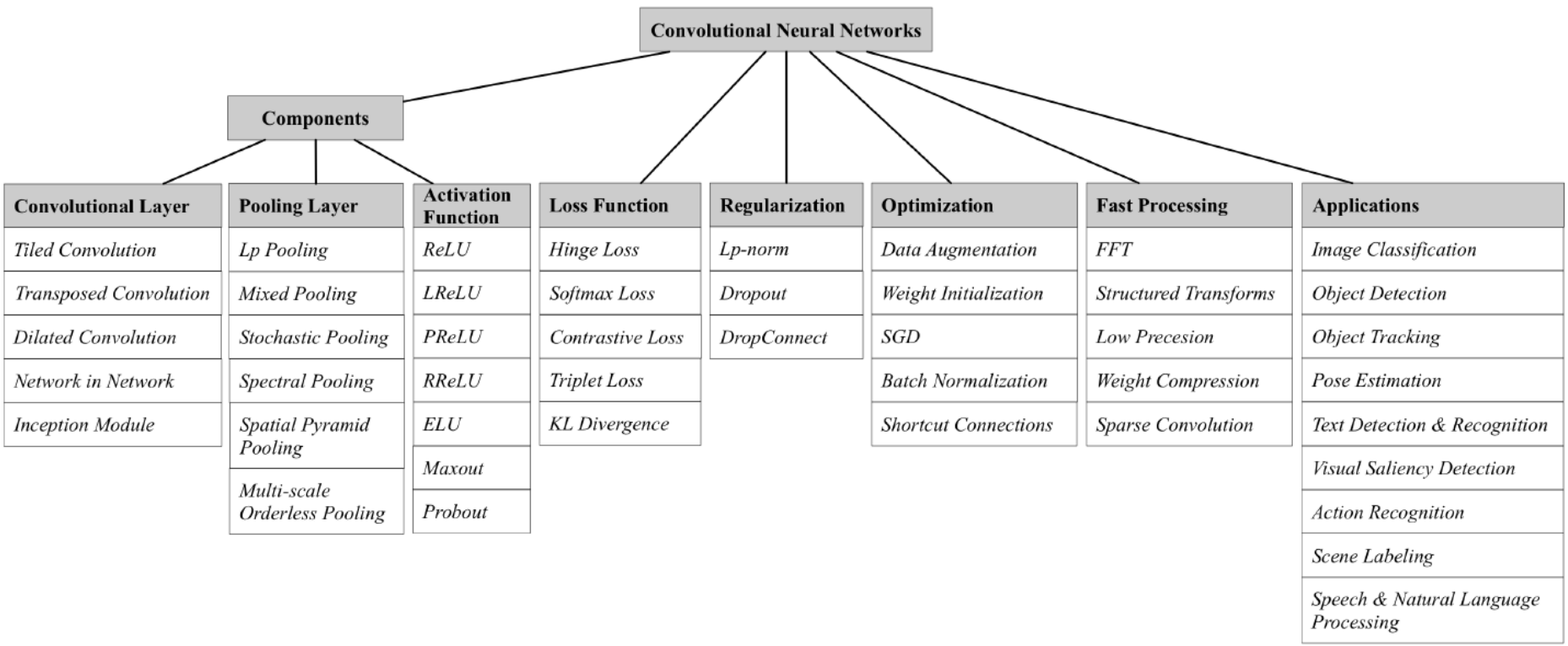
Road map

1. Why deep learning?
2. The first stage: 1890 - 1969
3. The second stage: 1985 - 1995
4. The third stage: 2006 - (2012) - 2018...
5. What's new in deep learning?
   - Big is beautiful
   - Two Hot topics: data and architecture
6. Conclusion



What's new with deep learning

- a lot of data (big data)
- big computing resources (hardware & software),
- big model (deep vs. shallow)
  - → new architectures
  - → new learning tricks



from Recent advances in convolutional neural networks Gu et al. Pattern Recognition, 2017

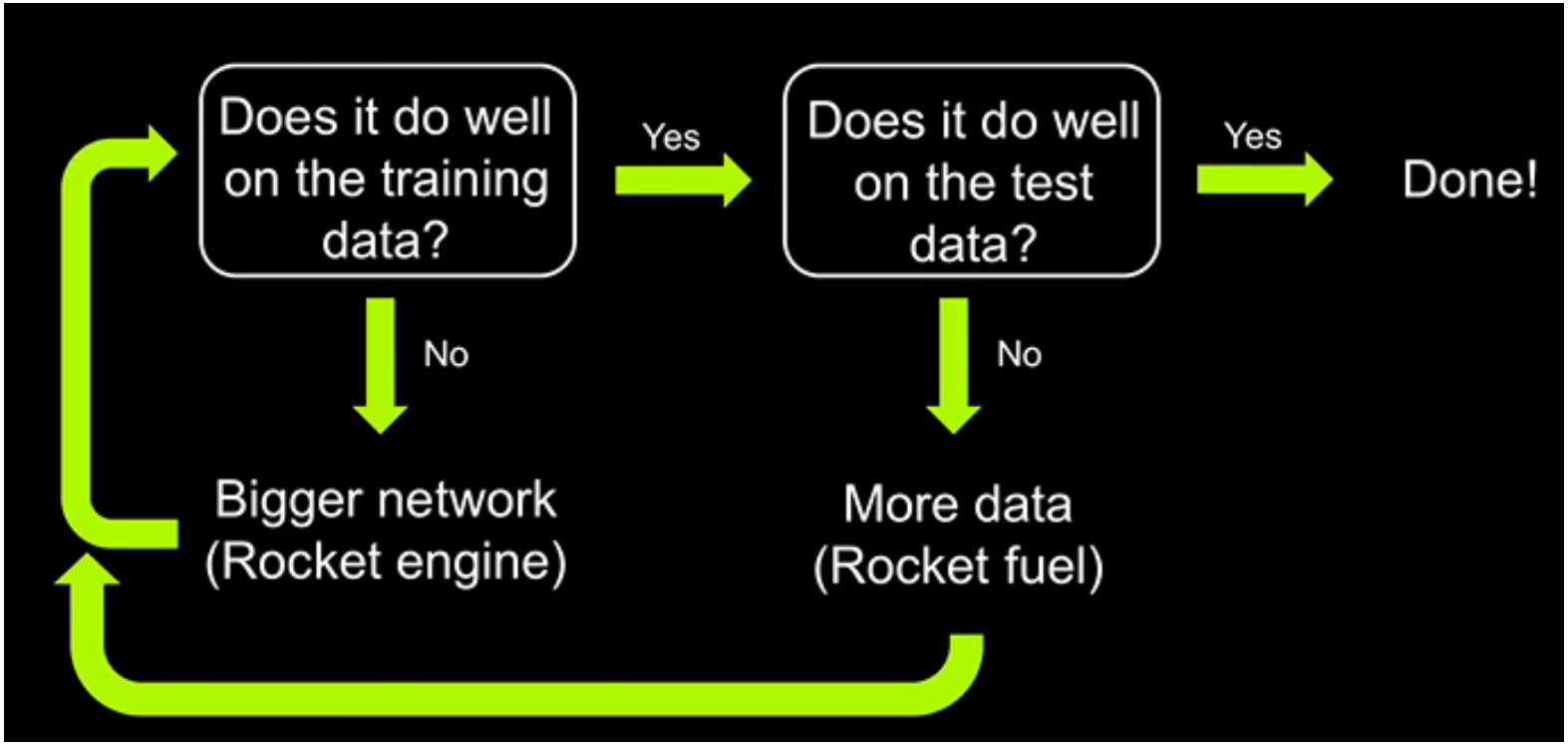
Big data: a lot of available training data

- ImageNet: 1,200,000x256x256x3 (about 200GB) block of pixels
- MS COCO for supervised learning
  - Multiple objects per image
  - More than 300,000 images
  - More than 2 Million instances
  - 80 object categories
  - 5 captions per image
- YFCC100M for unsupervised learning
- Google Open Images, 9 million URLs to images annotated over 6000 categories
- Visual genome: data + knowledge `http://visualgenome.org/`

Andrew Ng basic recipe for machine learning

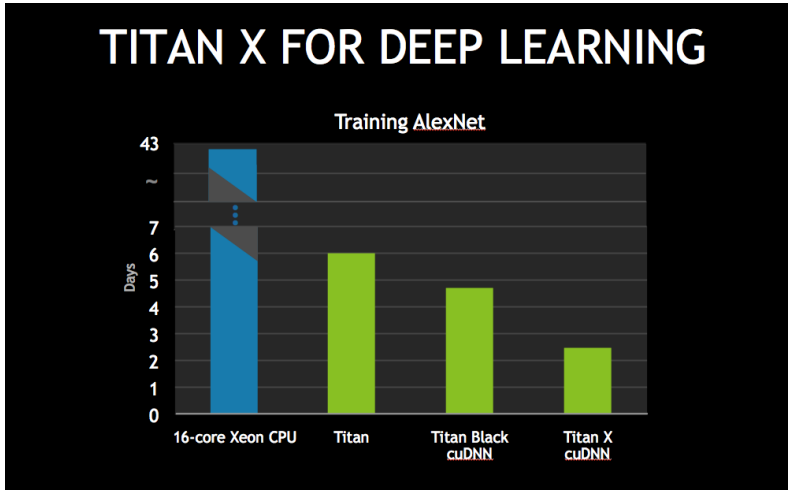
Why is Deep Learning taking off?

fuel = data engine = model (a deep network) .



Andrew Ng GTC 2015 Keynote, GPU Technology Nvidia

Big computers: GPU needed



Now 2 hours with Nvidia DGX-1, and enough Memory

Table 1 : Training time and top-1 1-crop validation accuracy with ImageNet/ResNet-50

	Batch Size	Processor	DL Library	Time	Accuracy
He et al. **[7]**	256	Tesla P100 x8	Caffe	29 hours	75.3%
Goyal et al. **[1]**	8K	Tesla P100 x256	Caffe2	1 hour	76.3%
Smith et al. **[4]**	8K→16K	full TPU Pod	TensorFlow	30 mins	76.1%
Akiba et al. **[5]**	32K	Tesla P100 x1024	Chainer	15 mins	74.9%
Jia et al. **[6]**	64K	Tesla P40 x2048	TensorFlow	6.6 mins	75.8%
This work	**34K→68K**	**Tesla V100 x2176**	**NNL**	**224 secs**	**75.03%**

ImageNet/ResNet-50 Training in 224 Seconds, 2018

big software: deep learning frameworks

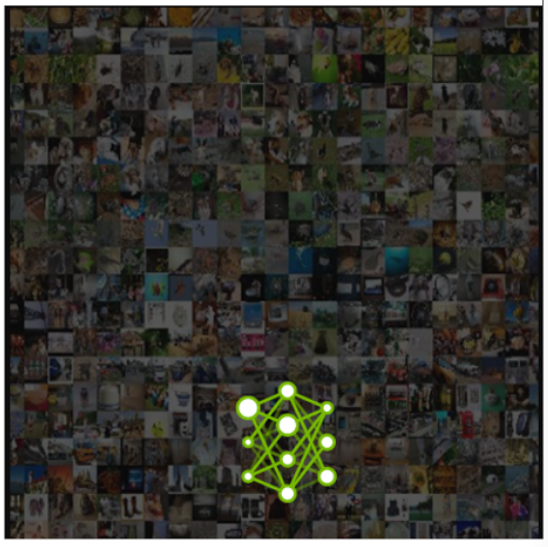
	Languages	Tutorials and training materials	CNN modeling capability	RNN modeling capability	Architecture: easy-to-use and modular front end	Speed	Multiple GPU support	Keras compatible
Theano	Python, C++	++	++	++	+	++	+	+
Tensor-Flow	Python	+++	+++	++	+++	++	++	+
Torch	Lua, Python (new)	+	+++	++	++	+++	++	
Caffe	C++	+	++		+	+	+	
MXNet	R, Python, Julia, Scala	++	++	+	++	++	+++	
Neon	Python	+	++	+	+	++	+	
CNTK	C++	+	+	+++	+	++	+	

Tensorflow (Google) is the most popular with Keras. Pytorch is a chalenger.

http://www.kdnuggets.com/2017/03/getting-started-deep-learning.html

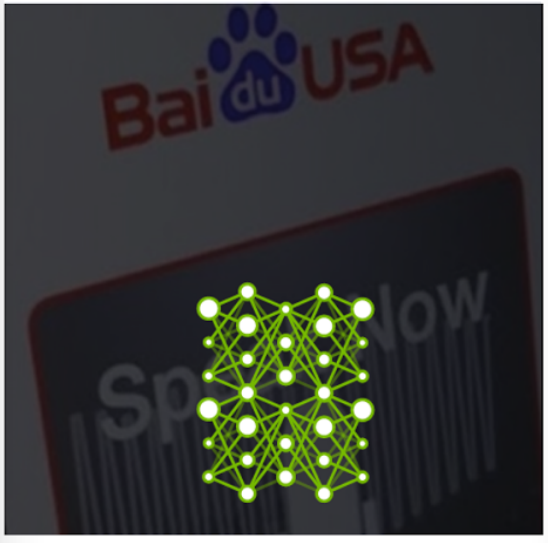
Big architectures

7 ExaFLOPS
60 Million Parameters



2015 - Microsoft ResNet
Superhuman Image Recognition

20 ExaFLOPS
300 Million Parameters



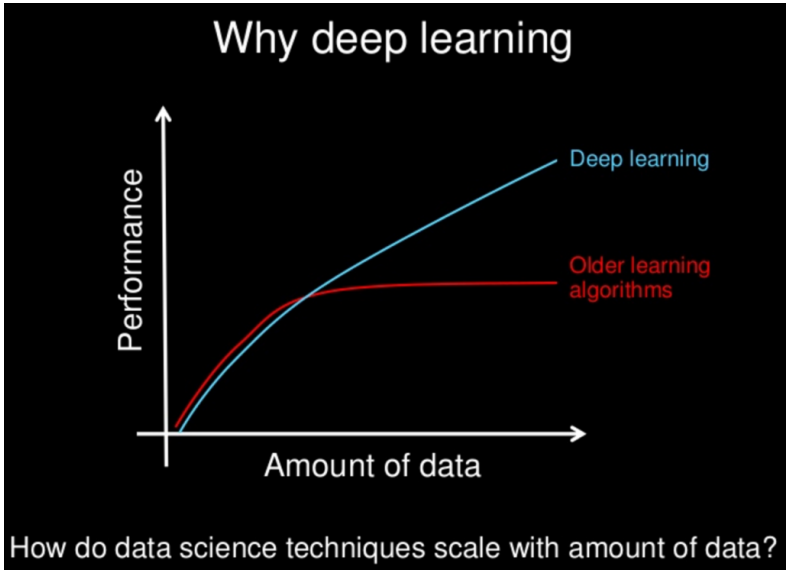
2016 - Baidu Deep Speech 2
Superhuman Voice Recognition

100 ExaFLOPS
8700 Million Parameters

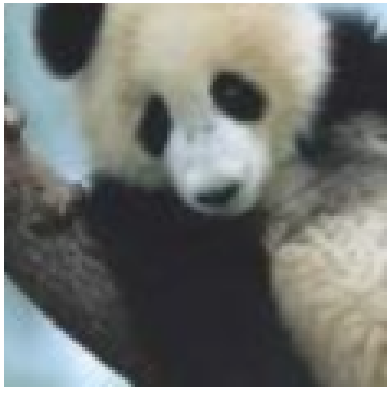
2017 - Google Neural Machine Translation
Near Human Language Translation

Road map

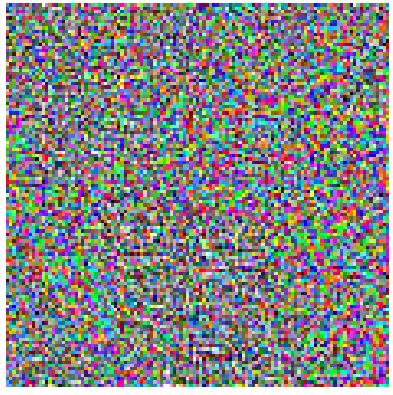
1. Why deep learning?
2. The first stage: 1890 - 1969
3. The second stage: 1985 - 1995
4. The third stage: 2006 - (2012) - 2018...
5. What's new in deep learning?
   - Big is beautiful
   - Two Hot topics: data and architecture
6. Conclusion



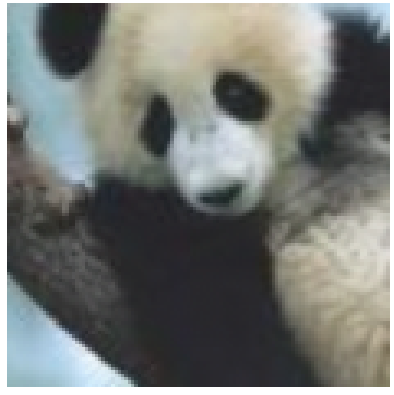
Deep neural networks are easily fooled (1/2)



$+ .007 \times$



$=$



$\boldsymbol{x}$

"panda"
57.7% confidence

$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"nematode"
8.2% confidence

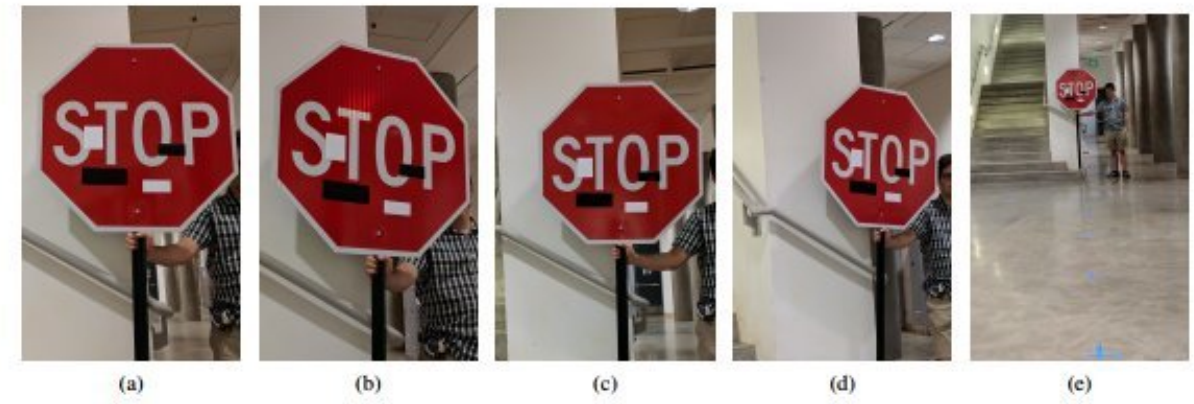
$\boldsymbol{x} + \epsilon \text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"gibbon"
99.3 % confidence

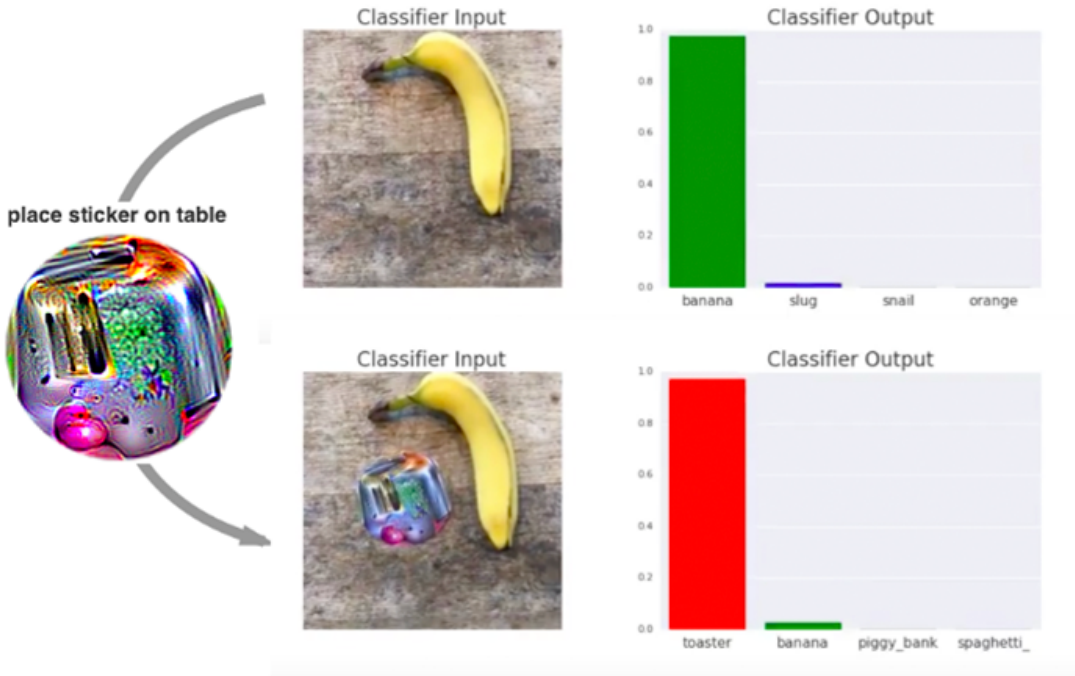
Explaining and Harnessing Adversarial Examples, Ian J. Goodfellow, Jonathon Shlens, Christian Szegedy, 2015

https://arxiv.org/abs/1412.6572

Adversarial examples (2/2)

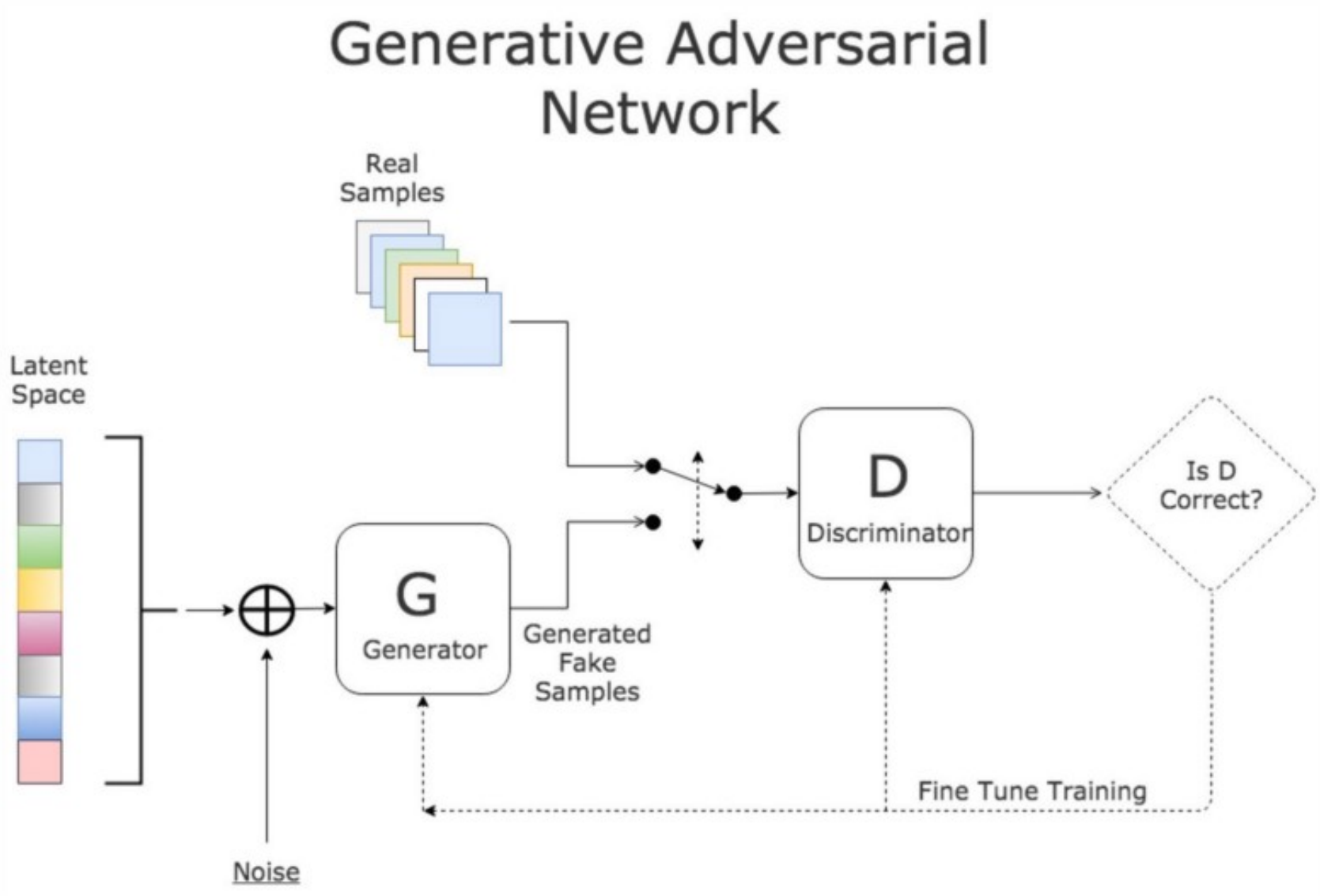


Adversarial Examples for Evaluating Reading Comprehension Systems, Robin Jia, Percy Liang, 2017



Adversarial Patch Tom B. Brown, Dandelion Mané, Aurko Roy, Martin Abadi, Justin Gilmer, 2017

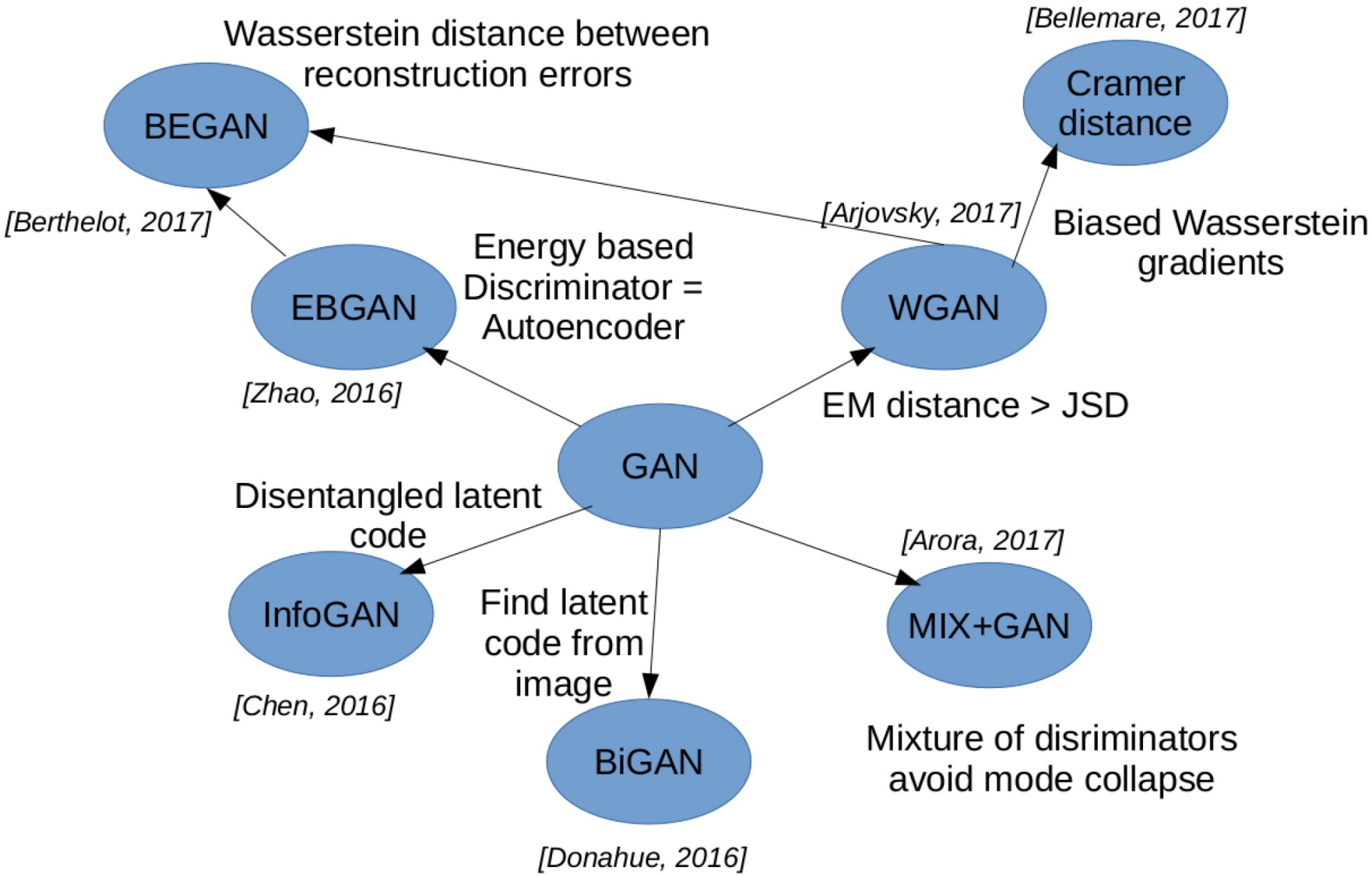
Generative models



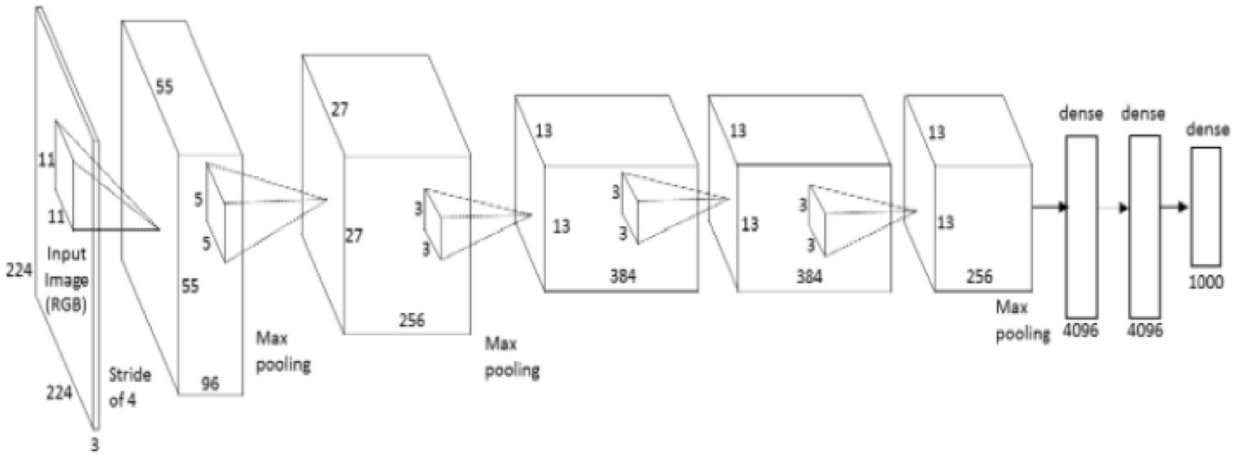
https://www.datasciencecentral.com/profiles/blogs/generative-adversarial-networks-gans-engine-and-applications

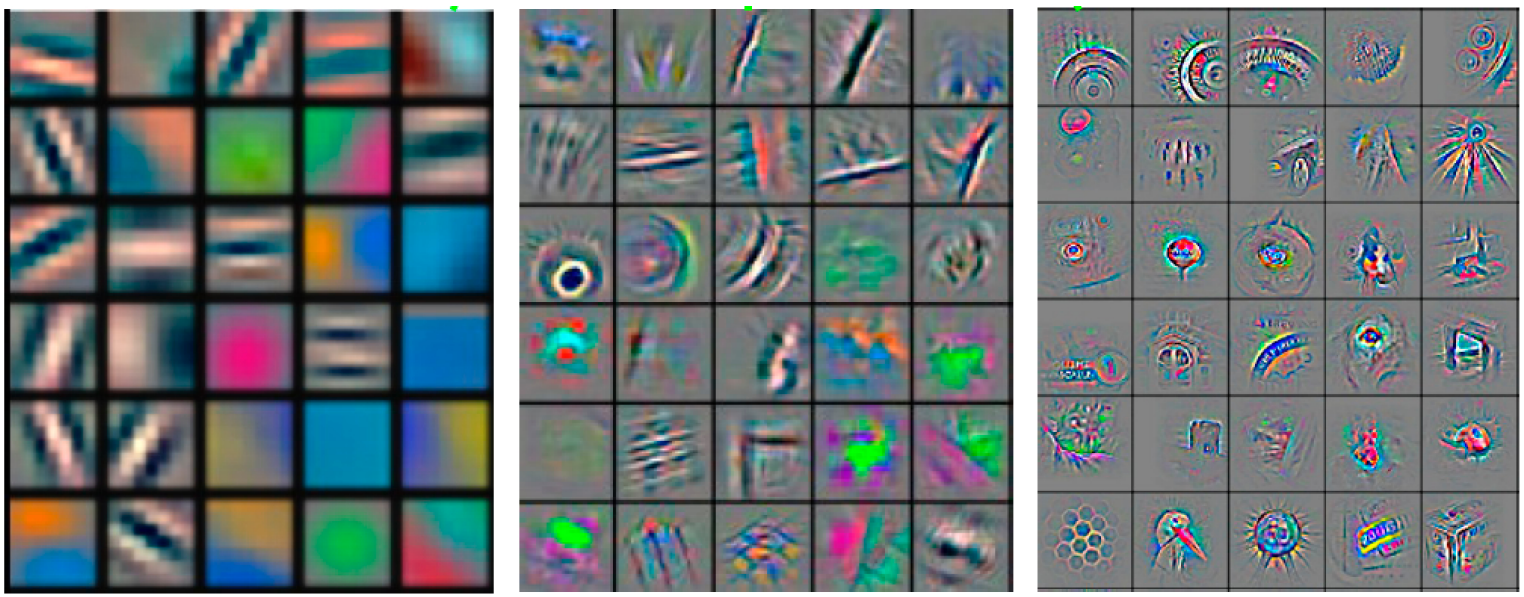
Defense-GAN: Protecting classifiers against adversarial attacks using generative models, 2018

Other Generative architectures



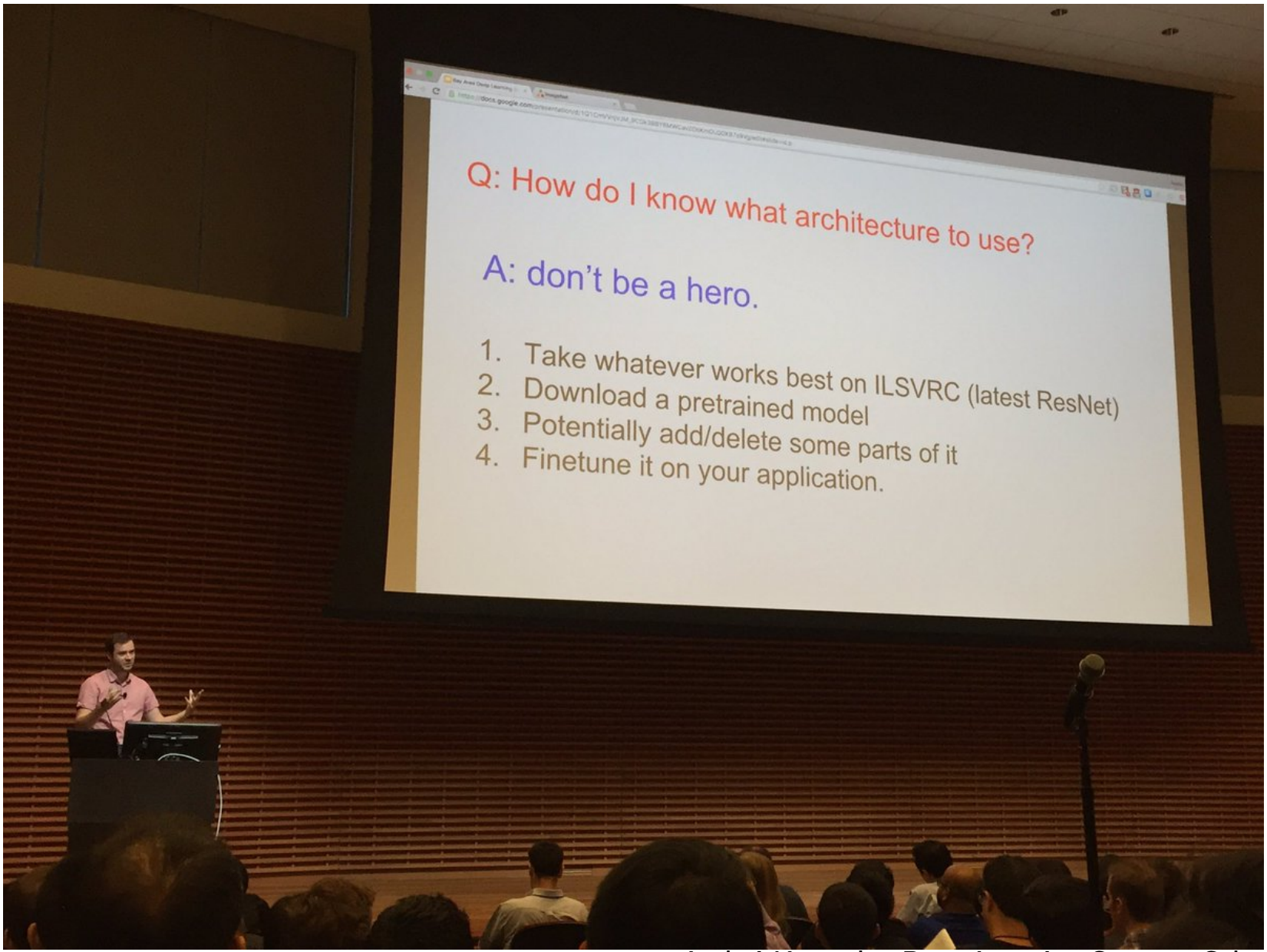
AlexNet works because of learning internal representation





Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

How to start with deep learning?

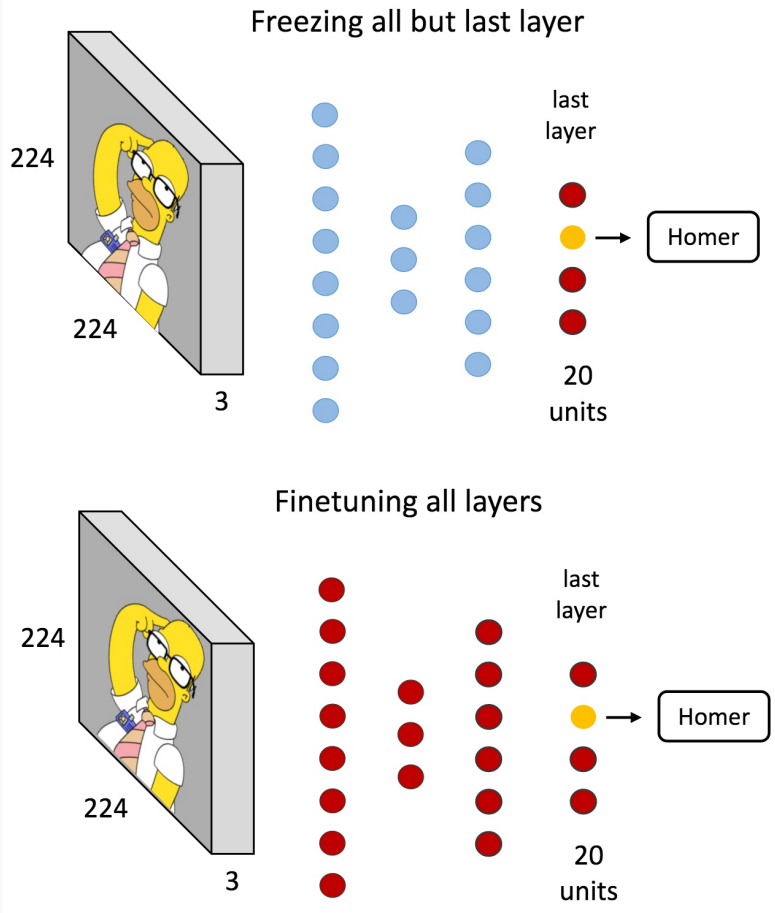


Andrej Karpathy, Deep Learning Summer School 2016

The art of using pre-trained models

- Transfer learning:
  1. download a pre-trained deep architecture (e.g. AlexNet for image processing)
  2. propagate new data through the network without its last(s) layer(s)
  3. use the output of the network as new feature

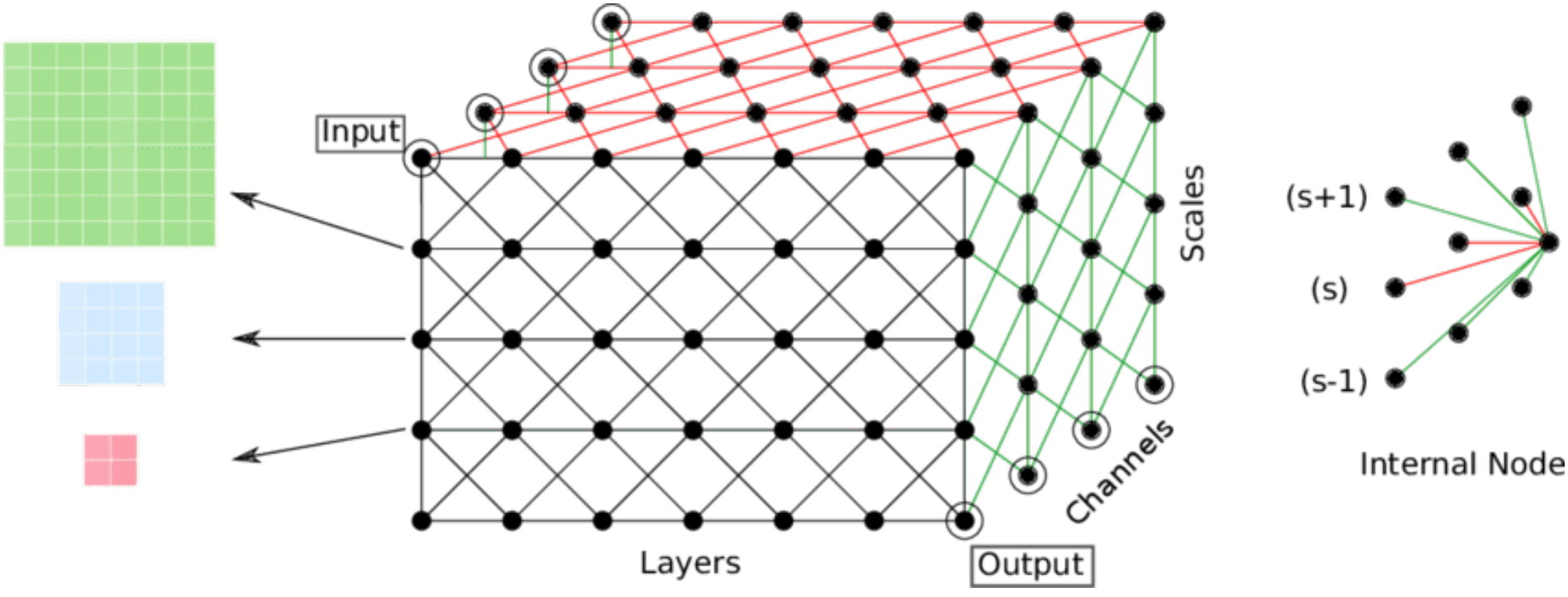
- Fine tuning
  1. download a pre-trained deep architecture (AlexNet)
  2. adapt the output layer to your problem
  3. train the deep architecture with your data using the pre-trained model as a starting point



https://miguelgfierro.com/blog/2017/a-gentle-introduction-to-transfer-learning-for-image-classification/

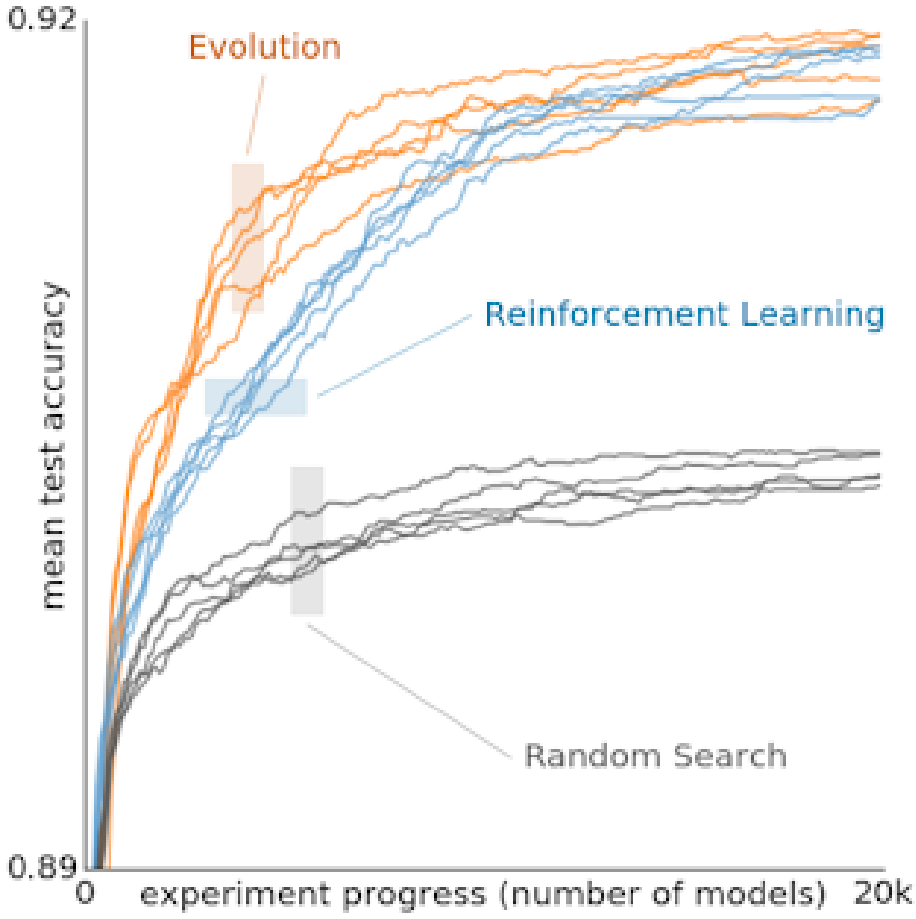
Convolutional Neural Fabrics

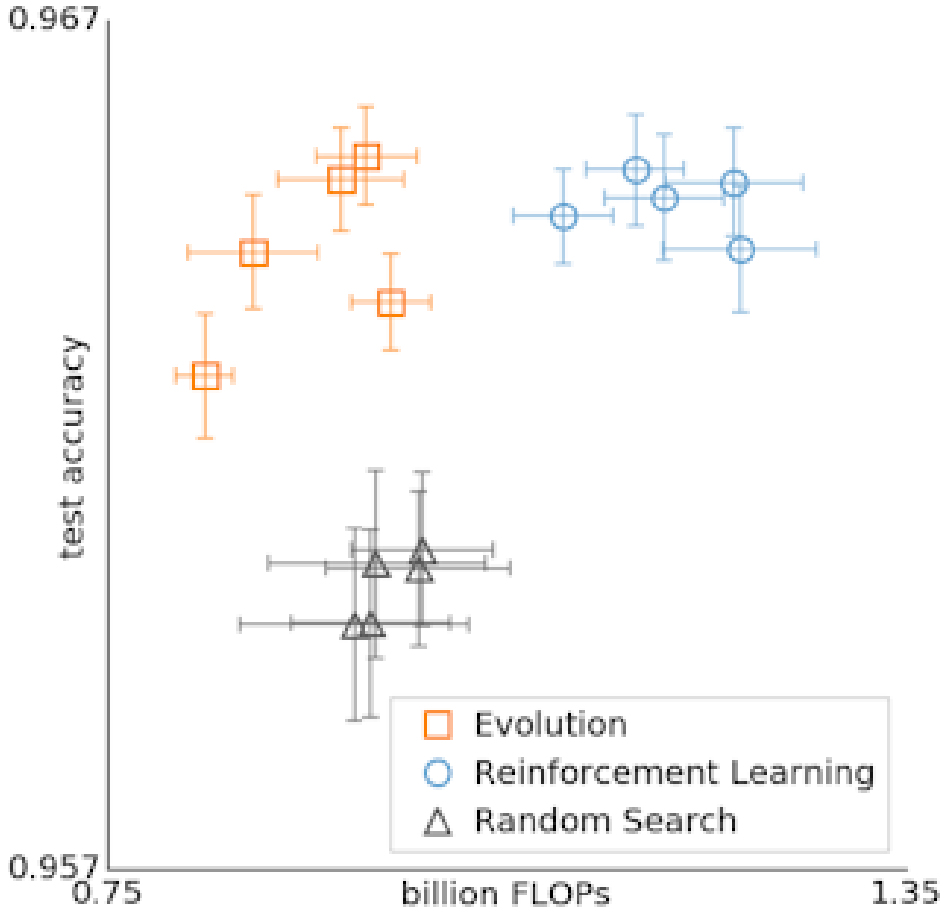
- problem: how to find the most relevant architecture
- todays solution: try and test
- A new solution: learn the architecture



Convolutional Neural Fabrics, Saxena and Verbeek, NIPS 2016

Neural Architecture Search



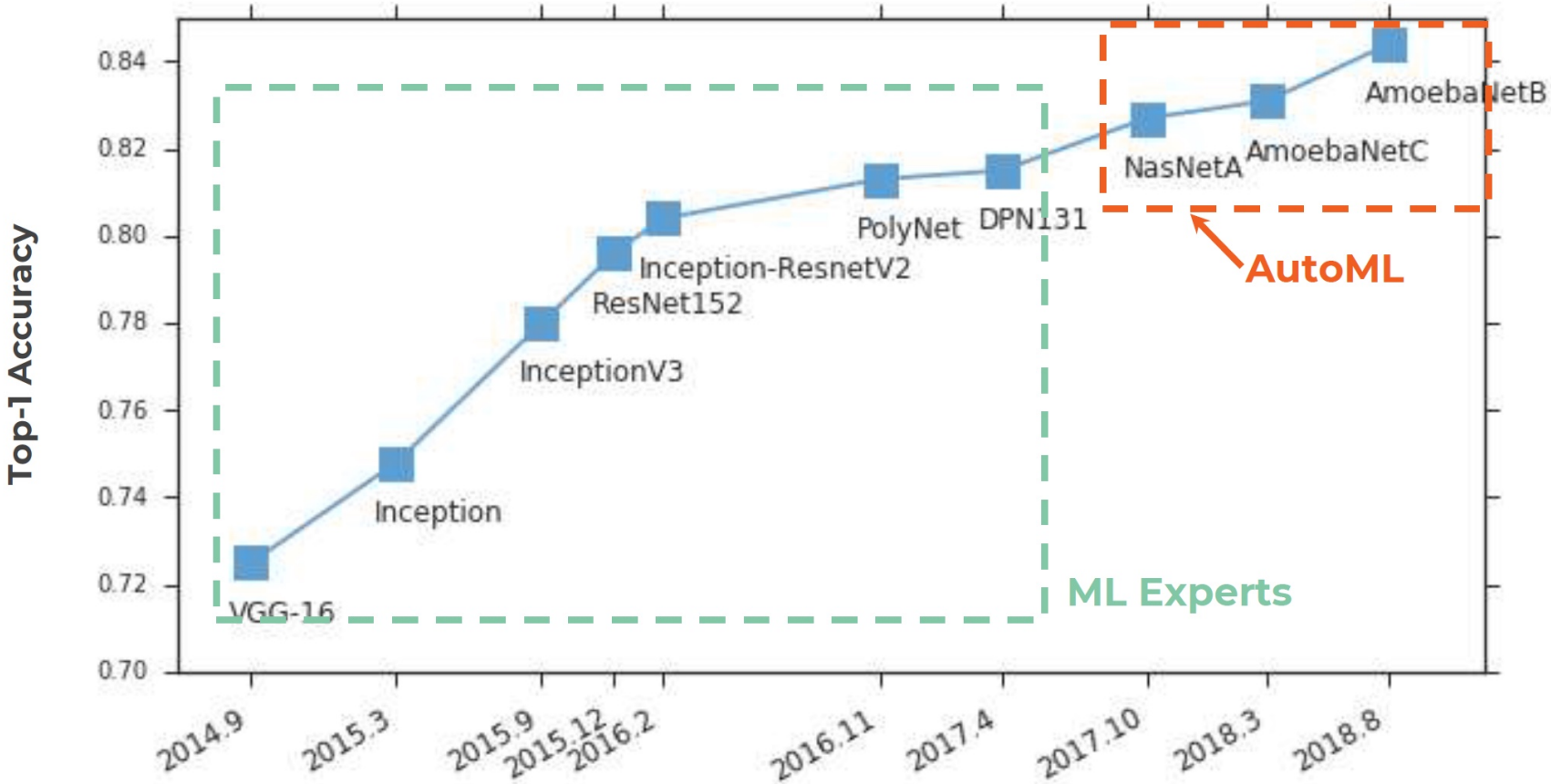


Regularized Evolution for Image Classifier Architecture Search, E. Real et al, 2018

https://chinagdg.org/2018/03/using-evolutionary-automl-to-discover-neural-network-architectures/

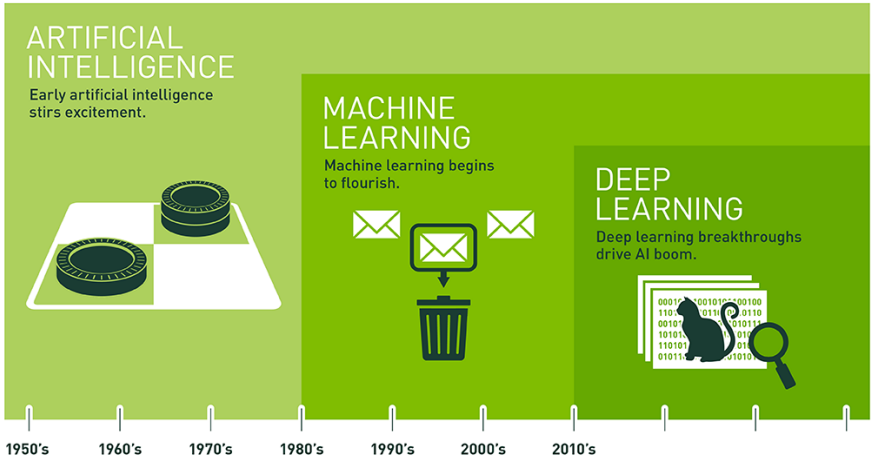
Experts vs AutoML

ImageNet



Road map

1. Why deep learning?
2. The first stage: 1890 - 1969
3. The second stage: 1985 - 1995
4. The third stage: 2006 - (2012) - 2018...
5. What's new in deep learning?
   - Big is beautiful
   - Two Hot topics: data and archit
6. Conclusion



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

The deep learning time line

- The first stage: 1890 - 1969
  - learning is optimization with stochastic gradient (to scale)
- The second stage: 1985 - 1995
  - NN are universal approximator differentiable graphs (that scales)
- The third stage: 2006 - (2012) - 2018...
  - scale with big data+computers+architecture (deep)
- Open issues
  - provide guaranties: adversarial examples and representation learning
  - architecture design (autoML)
  - theory needed
  - do more with less: green learning
  - the future of deep learning depends on trust

Open issues in *Deep learning : A critical apaisal*

Deep Learning: A Critical Appraisal

Gary Marcus

(Submitted on 2 Jan 2018)

Although deep learning has historical roots going back decades, neither the term "deep learning" nor the approach was popular just over five years ago, when the field was reignited by papers such as Krizhevsky, Sutskever and Hinton's now classic (2012) deep network model of Imagenet. What has the field discovered in the five subsequent years? Against a background of considerable progress in areas such as speech recognition, image recognition, and game playing, and considerable enthusiasm in the popular press, I present ten concerns for deep learning, and suggest that deep learning must be supplemented by other techniques if we are to reach artificial general intelligence.

Comments: 1 figure
Subjects: **Artificial Intelligence (cs.AI)**; Machine Learning (cs.LG); Machine Learning (stat.ML)
MSC classes: 97R40
ACM classes: I.2.0; I.2.6
Cite as: arXiv:1801.00631 [cs.AI]
(or arXiv:1801.00631v1 [cs.AI] for this version)

For most problems where deep learning has enabled transformationally better solutions (vision, speech), we've entered diminishing returns territory in 2016-2017.

Francois Chollet, Google, author of Keras neural network library Dec. 2017

10 Limits on the scope of deep learning

- It is data hungry
- specialized training (no learning)
- it has no natural way to deal with hierarchical structure
- it has struggled with open-ended inference
- it is not sufficiently transparent
- not well integrated with prior knowledge (NLP)
- cannot distinguish causation from correlation
- assume stationarity
- can be fooled (it is not robust)
- is difficult to engineer with

10 Limits on the scope of deep learning

GREEN LEARNNING

- It is data hungry
- specialized training (no learning)
- it has no natural way to deal with hierarchical structure
- it has struggled with open-ended inference

THEORY NEEDED

- it is not sufficiently transparent

- not well integrated with prior knowledge (NLP)
- cannot distinguish causation from correlation
- assume stationarity

ADVERSARIAL TRANING

- can be fooled (it is not robust)

- is difficult to engineer with

AUTO ML

To go further

- books
  - I. Goodfellow, Y. Bengio & A. Courville, *Deep Learning*, MIT Press book, 2016 `http://www.deeplearningbook.org/`
  - Gitbook `leonardoaraujosantos.gitbooks.io/artificial-inteligence/`
- conferences
  - NIPS, ICLR, xCML, AIStats,
- Journals
  - JMLR, Machine Learning, Foundations and Trends in Machine Learning, machine learning survey `http://www.mlsurveys.com/`
- lectures
  - Deep Learning: Course by Yann LeCun at Collège de France in 2016 `college-de-france.fr/site/en-yann-lecun/inaugural-lecture-2016-02-04-18h00.htm`
  - Convolutional Neural Networks for Visual Recognition (Stanford)
  - deep mind (`https://deepmind.com/blog/`)
  - CS 229: Machine Learning at stanford Andrew Ng
- Blogs
  - Andrej Karpathy blog (`http://karpathy.github.io/`)
  - `http://deeplearning.net/blog/`
  - `https://computervisionblog.wordpress.com/category/computer-vision/`