

Introduction to Deep Learning and Applications



Fabio A. González
Univ. Nacional de Colombia



Some history





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



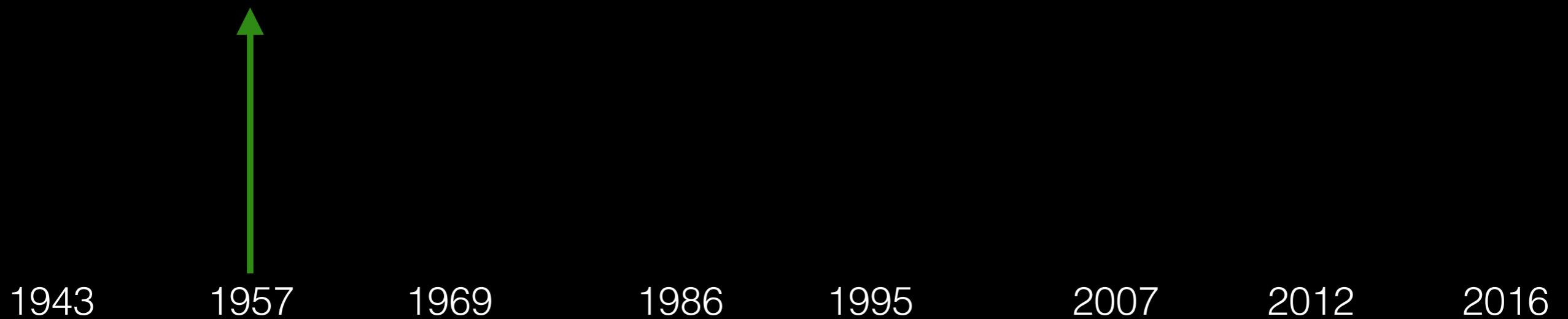
https://www.youtube.com/watch?v=cNxadbrN_al

Rosenblatt's Perceptron (1957)

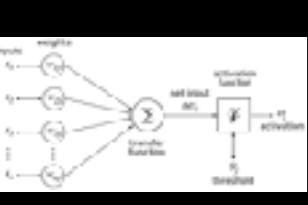
- Input: 20x20 photocells array
- Weights implemented with potentiometers
- Weight updating performed by electric motors



Neural networks time line



Neural networks time line



1943

1957

1969

1986

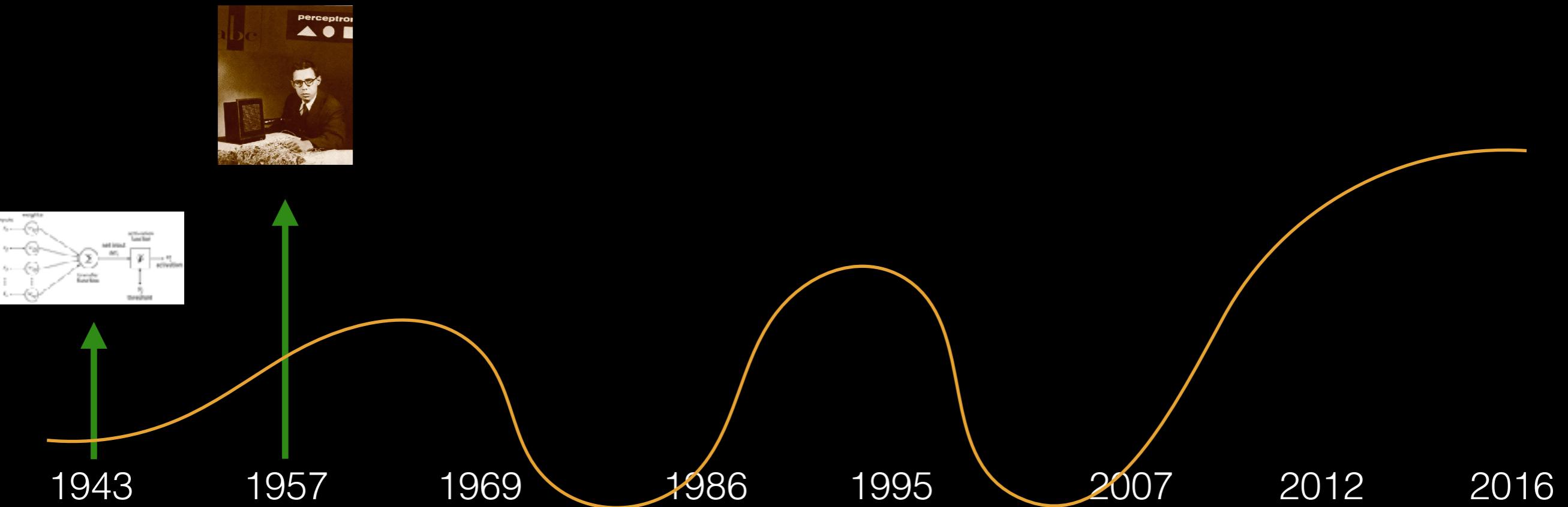
1995

2007

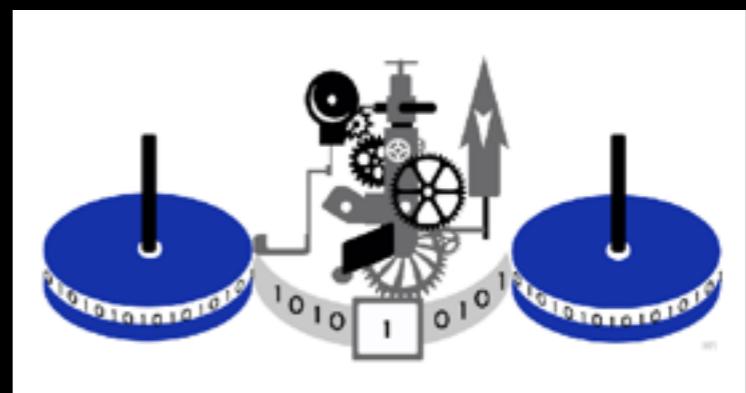
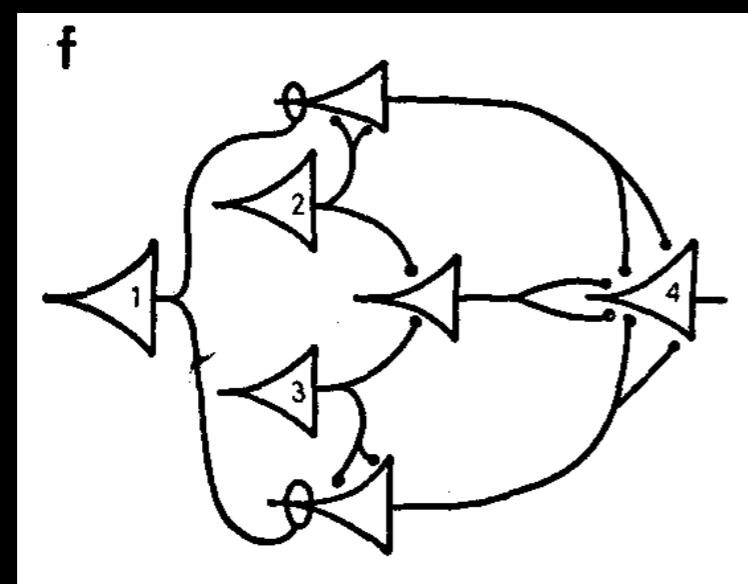
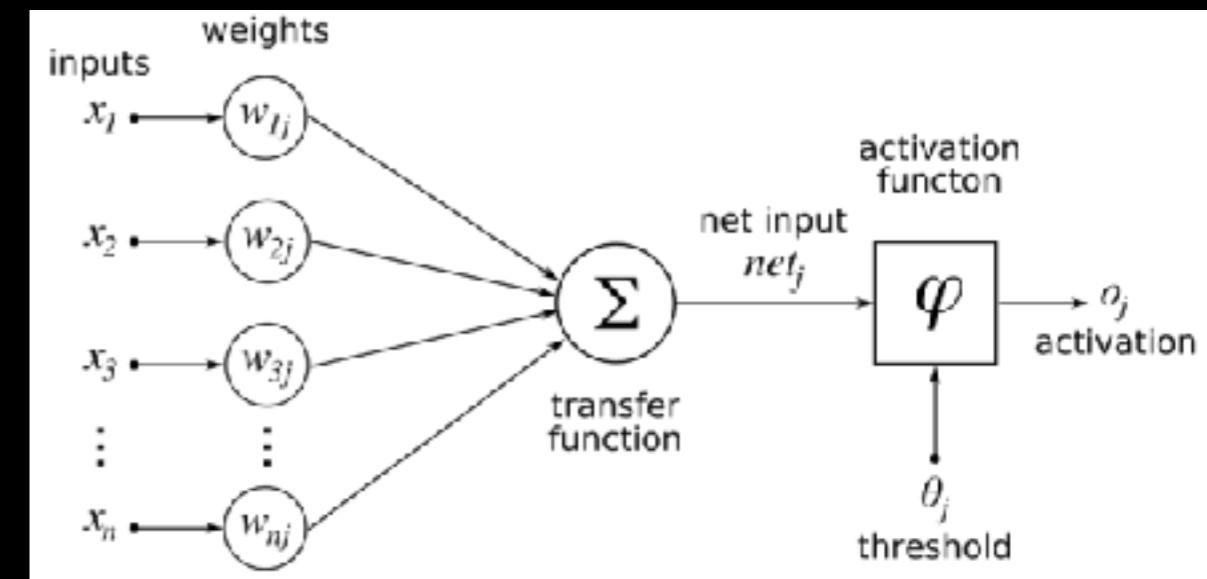
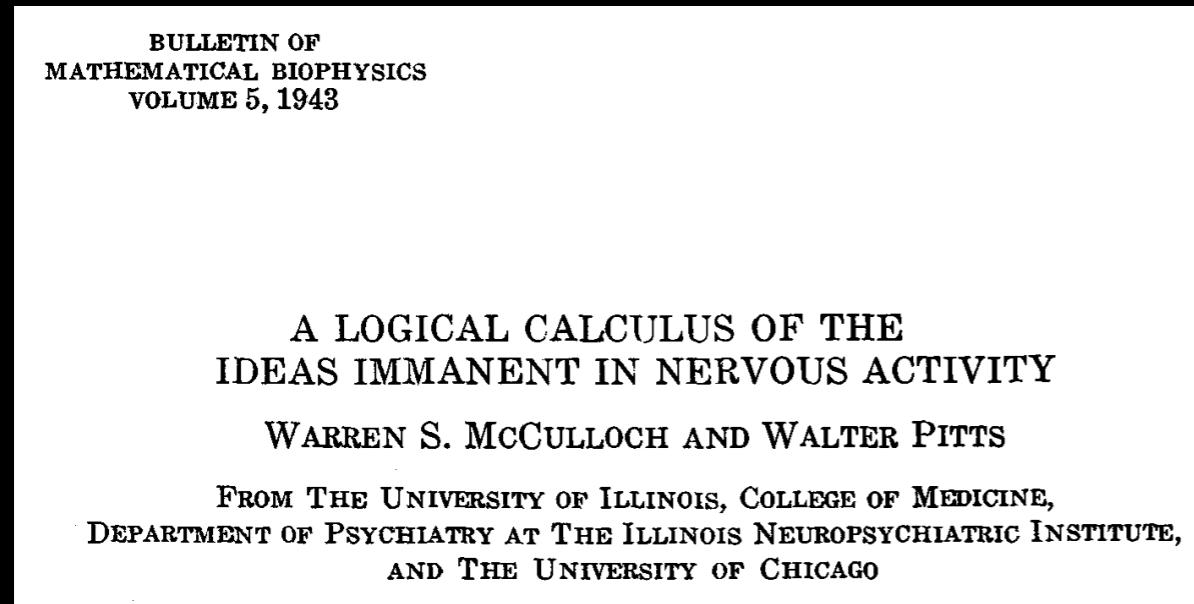
2012

2016

Neural networks time line



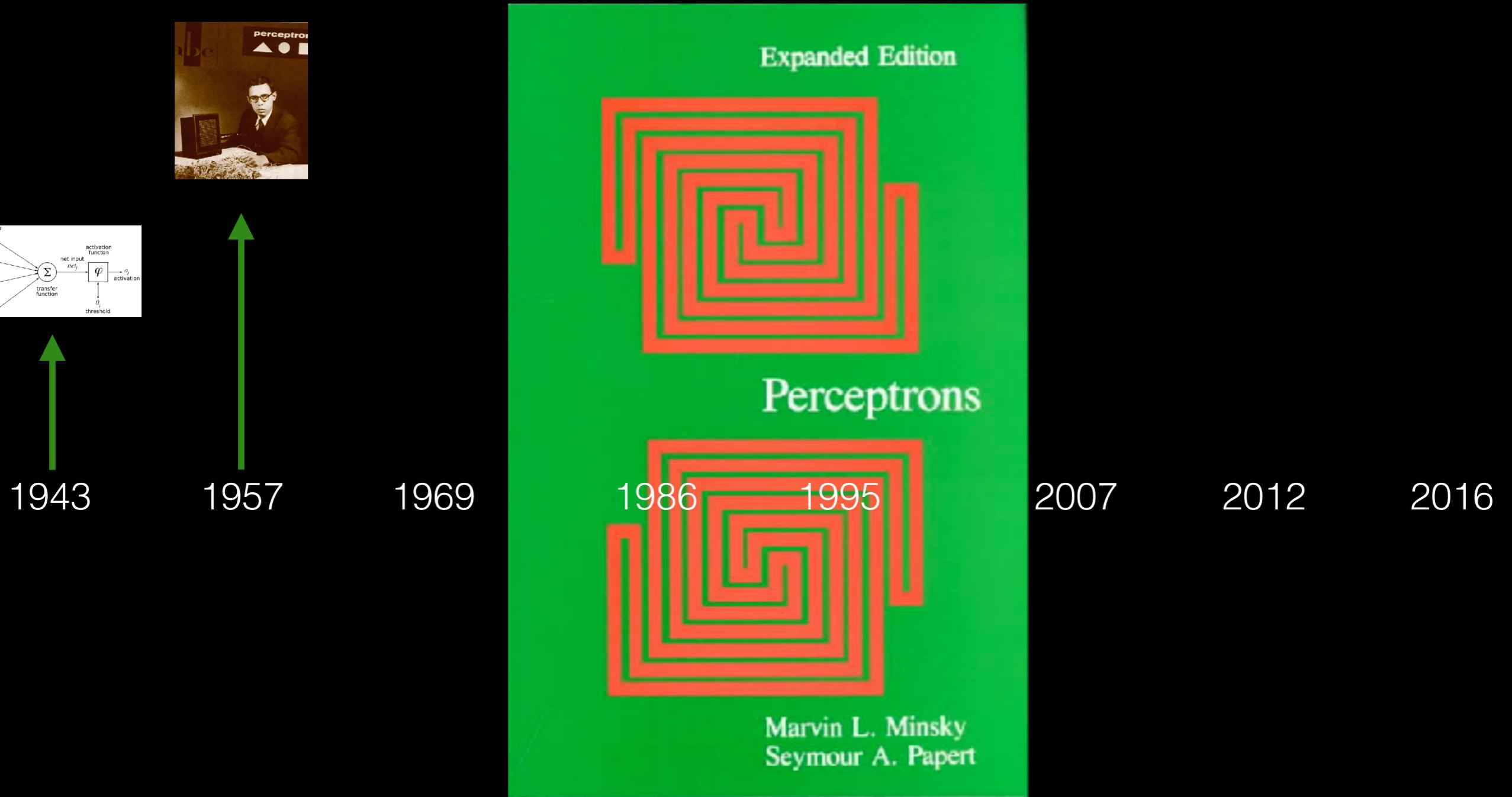
McCulloch & Pitts Artificial Neuron



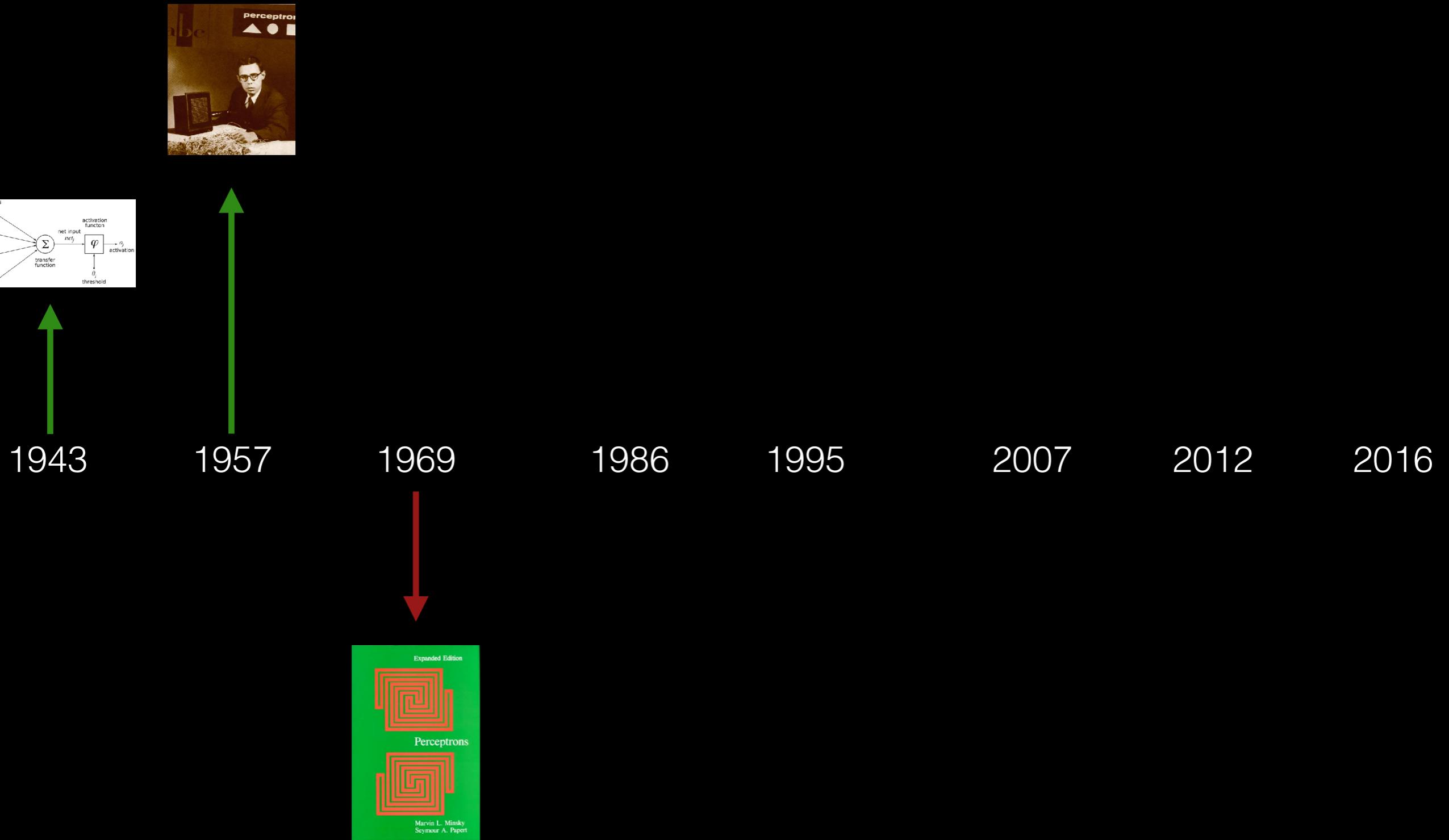
Neural networks time line



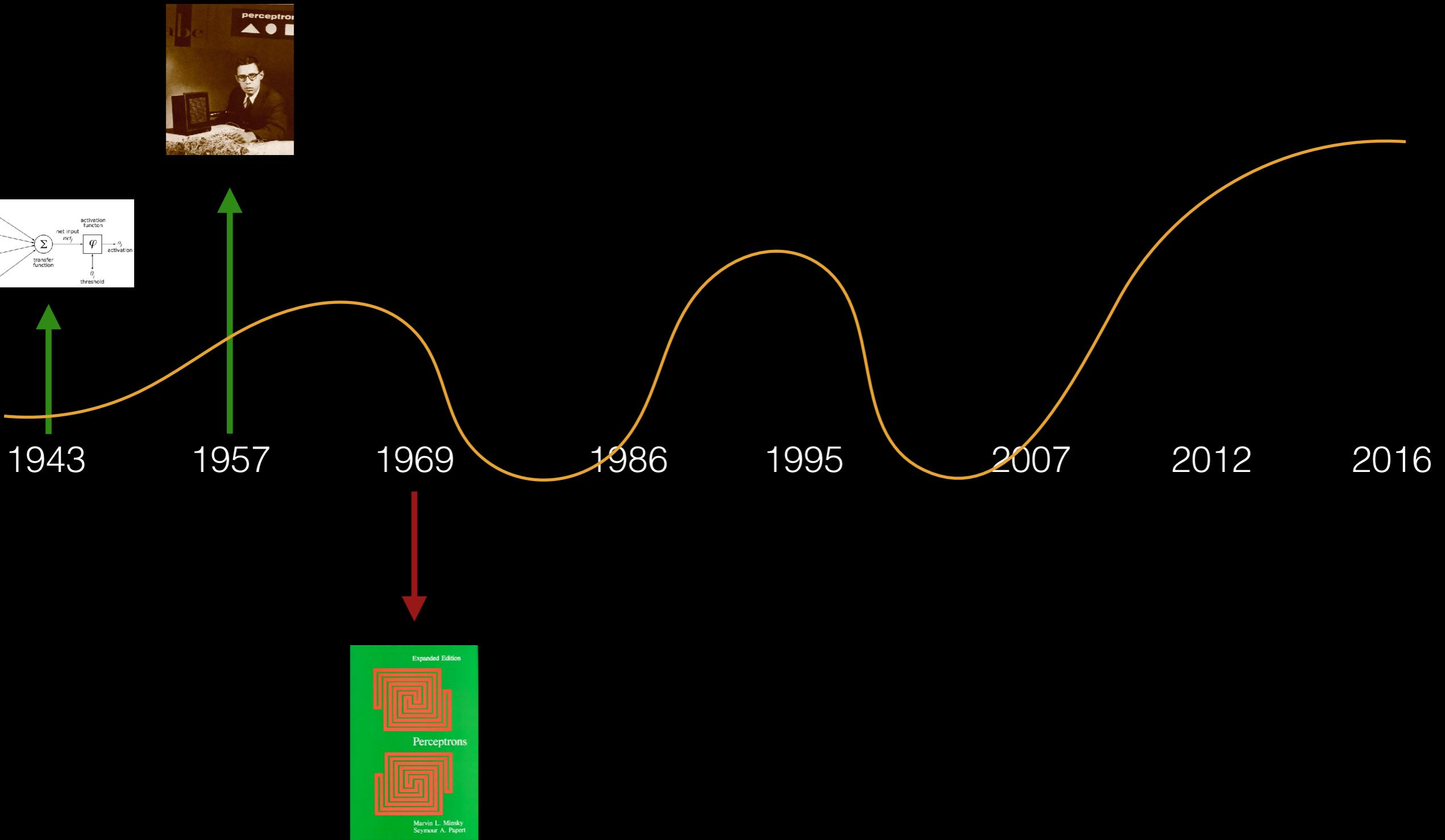
Neural networks time line



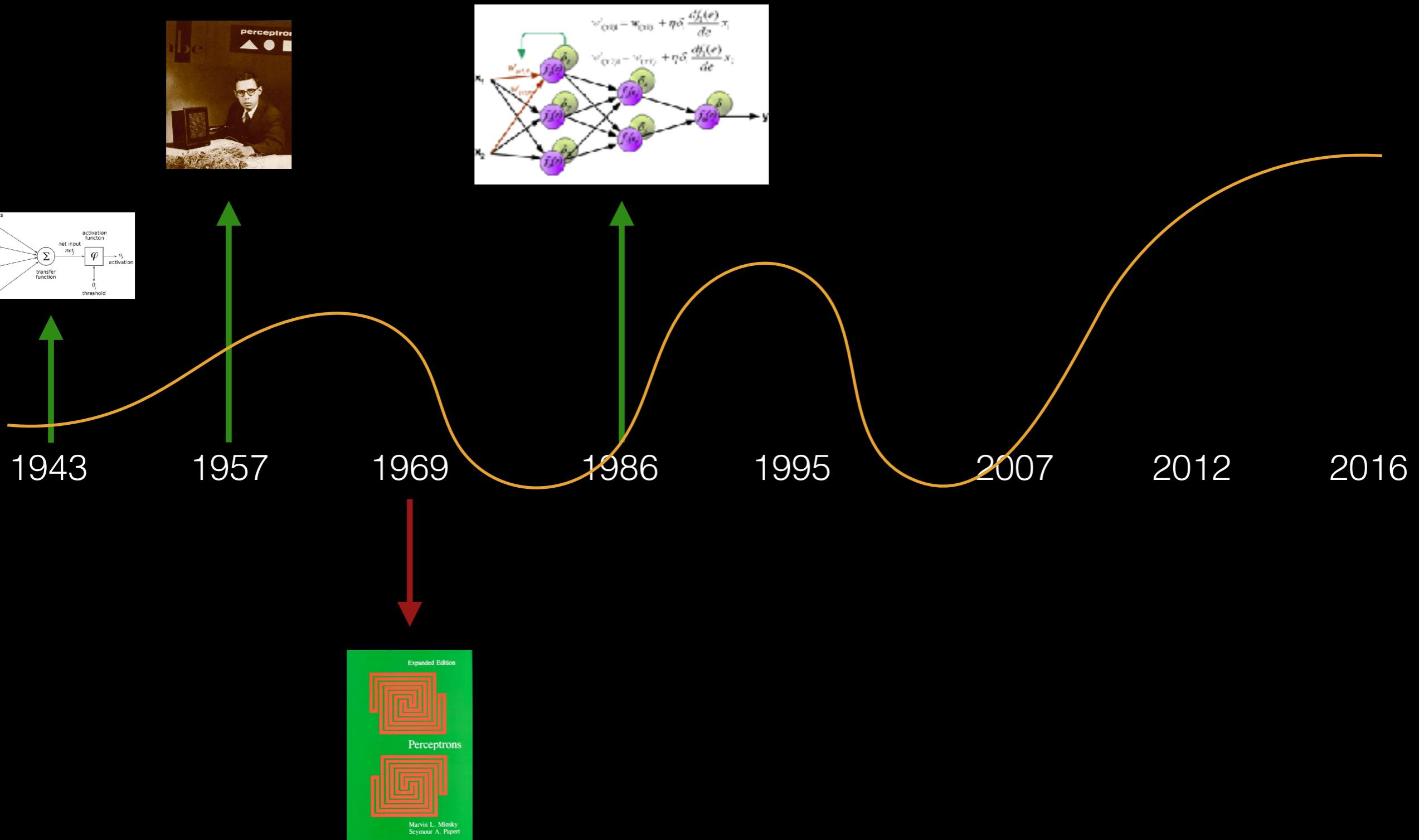
Neural networks time line



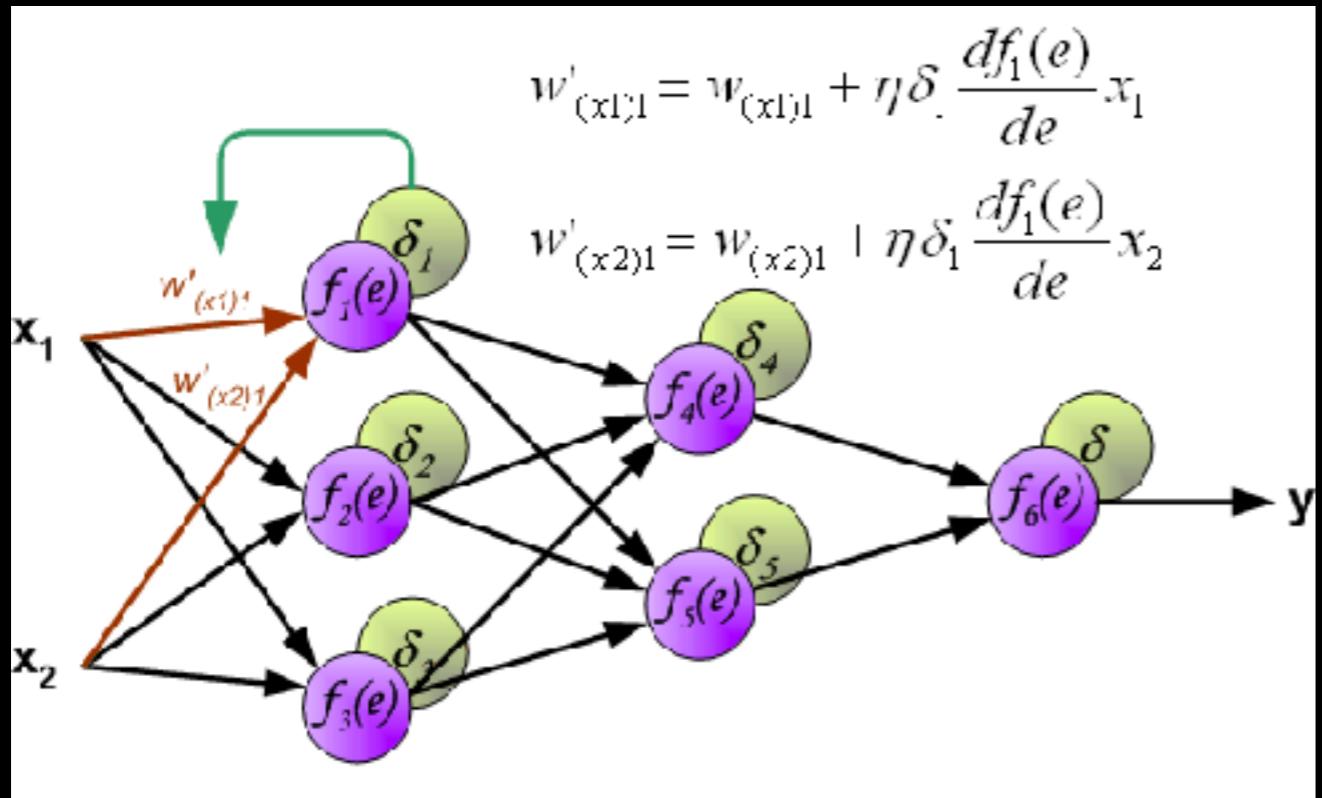
Neural networks time line



Neural networks time line



Backpropagation



Source: http://home.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html

Backpropagation

$$w'_{(x1|1)} = w_{(x1|1)} + \eta \delta_1 \frac{df_1(e)}{de} x_1$$
$$w'_{(x2|1)} = w_{(x2|1)} + \eta \delta_1 \frac{df_1(e)}{de} x_2$$

letters to nature

Nature 323, 533 - 536 (09 October 1986); doi:10.1038/323533a0

**Learning representations
by back-propagating errors**

**David E. Rumelhart*, Geoffrey E. Hinton†
& Ronald J. Williams***

* Institute for Cognitive Science, C-015, University of California,
San Diego, La Jolla, California 92093, USA
† Department of Computer Science, Carnegie-Mellon University,
Pittsburgh, Philadelphia 15213, USA

Source: http://home.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html

Backpropagation

$$w'_{(x1|1)} = w_{(x1|1)} + \eta \delta_1 \frac{df_1(e)}{de} x_1$$
$$w'_{(x2|1)} = w_{(x2|1)} + \eta \delta_1 \frac{df_1(e)}{de} x_2$$

letters to nature

Nature 323, 533 - 536 (09 October 1986); doi:10.1038/323533a0

Learning representations by back-propagating errors

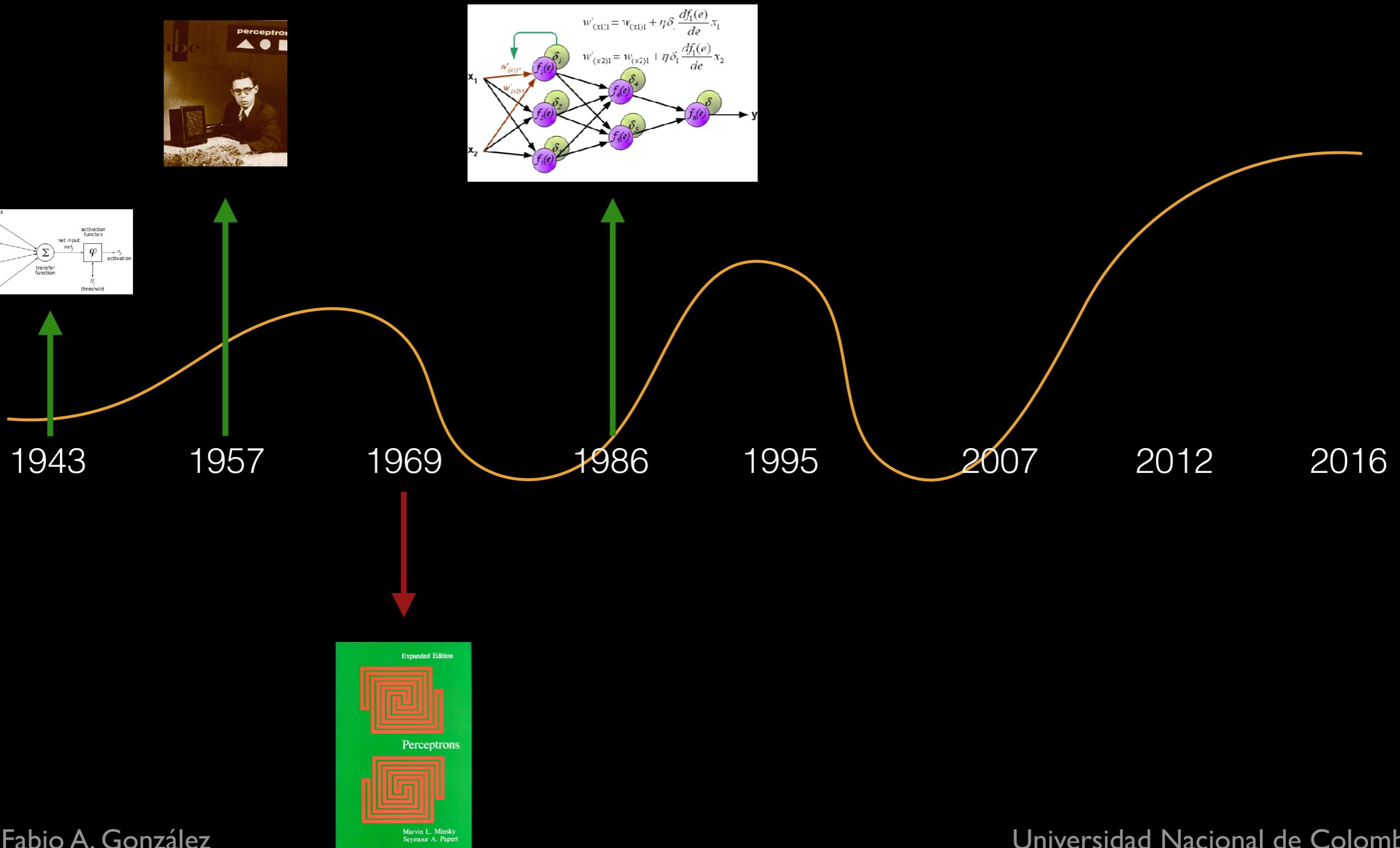
David E. Rumelhart*, **Geoffrey E. Hinton†**
& Ronald J. Williams*

* Institute for Cognitive Science, C-015, University of California
San Diego, La Jolla, California 92093, USA
† Department of Computer Science, Carnegie-Mellon University
Pittsburgh, Philadelphia 15213, USA

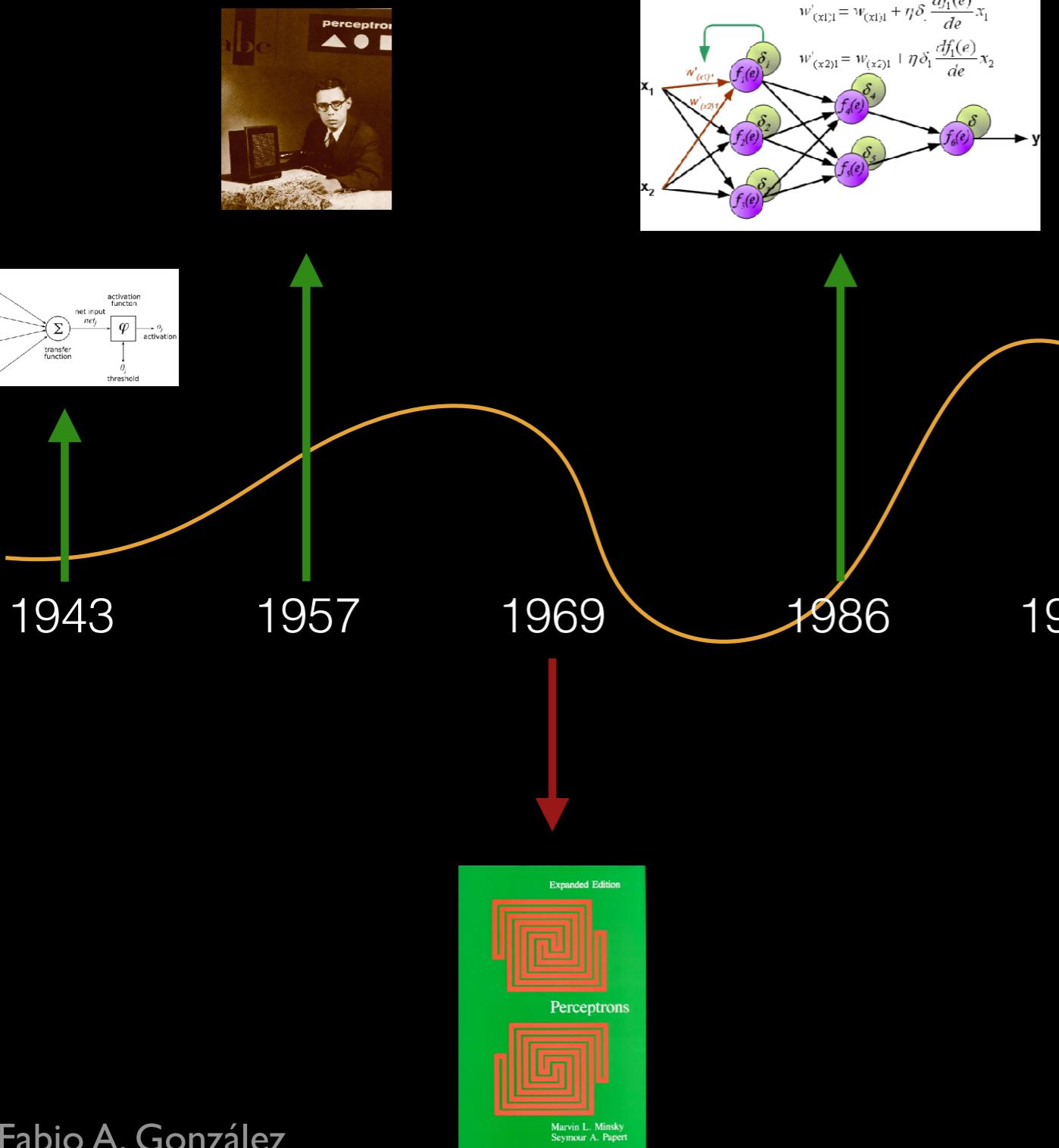


Source: http://home.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html

Neural networks time line



Neural networks



Statistics for
Engineering and
Information Science

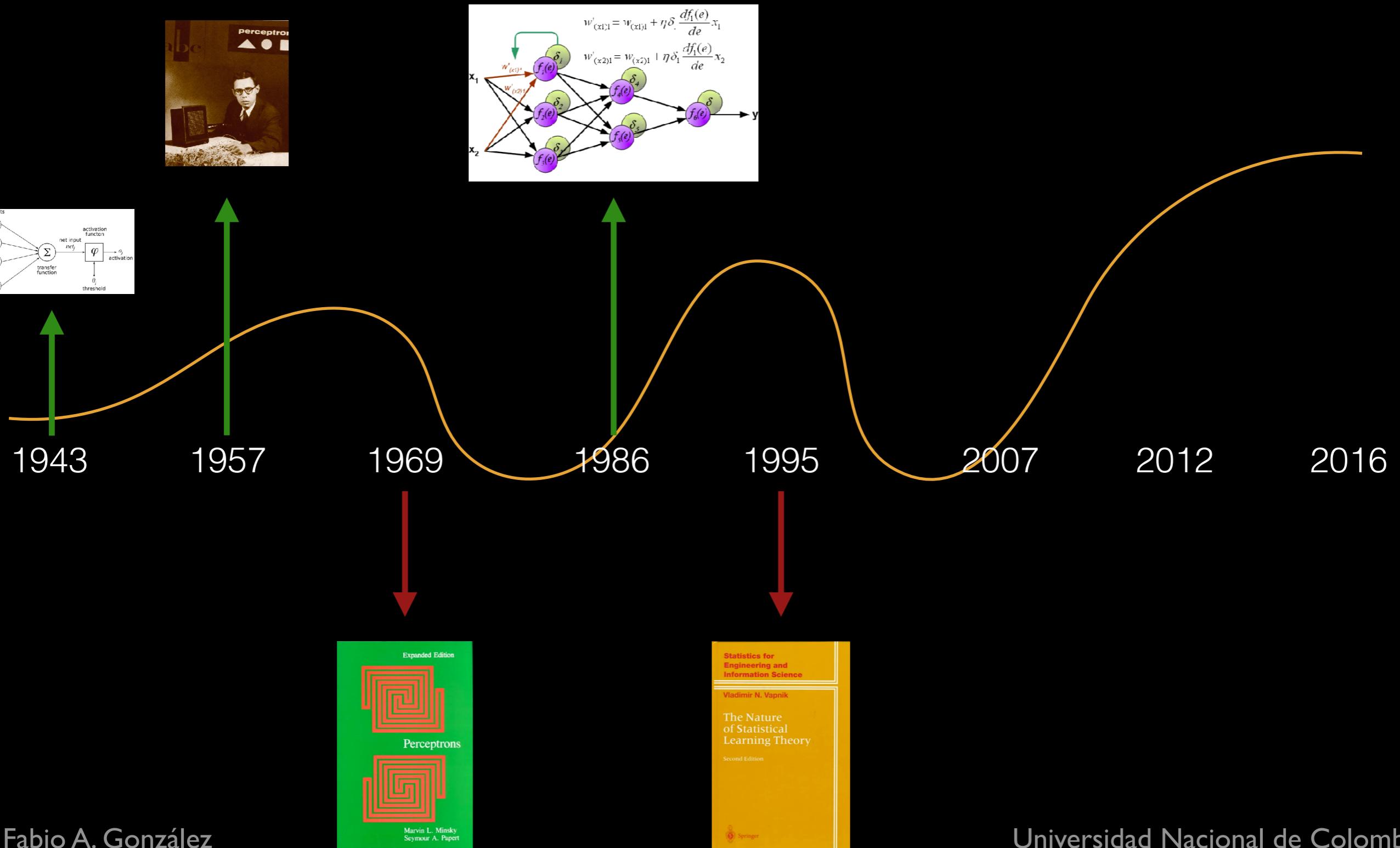
Vladimir N. Vapnik

The Nature
of Statistical
Learning Theory

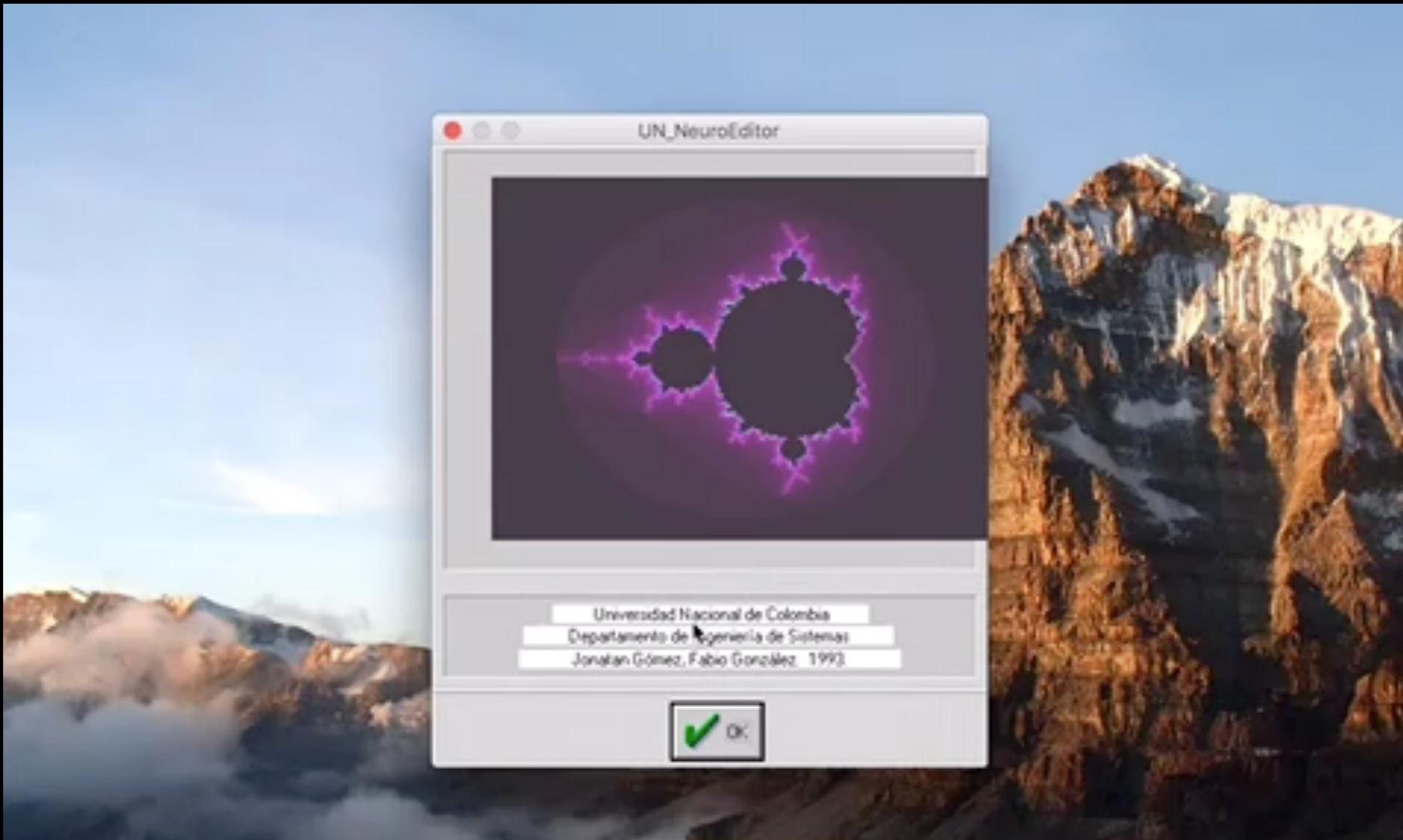
Second Edition



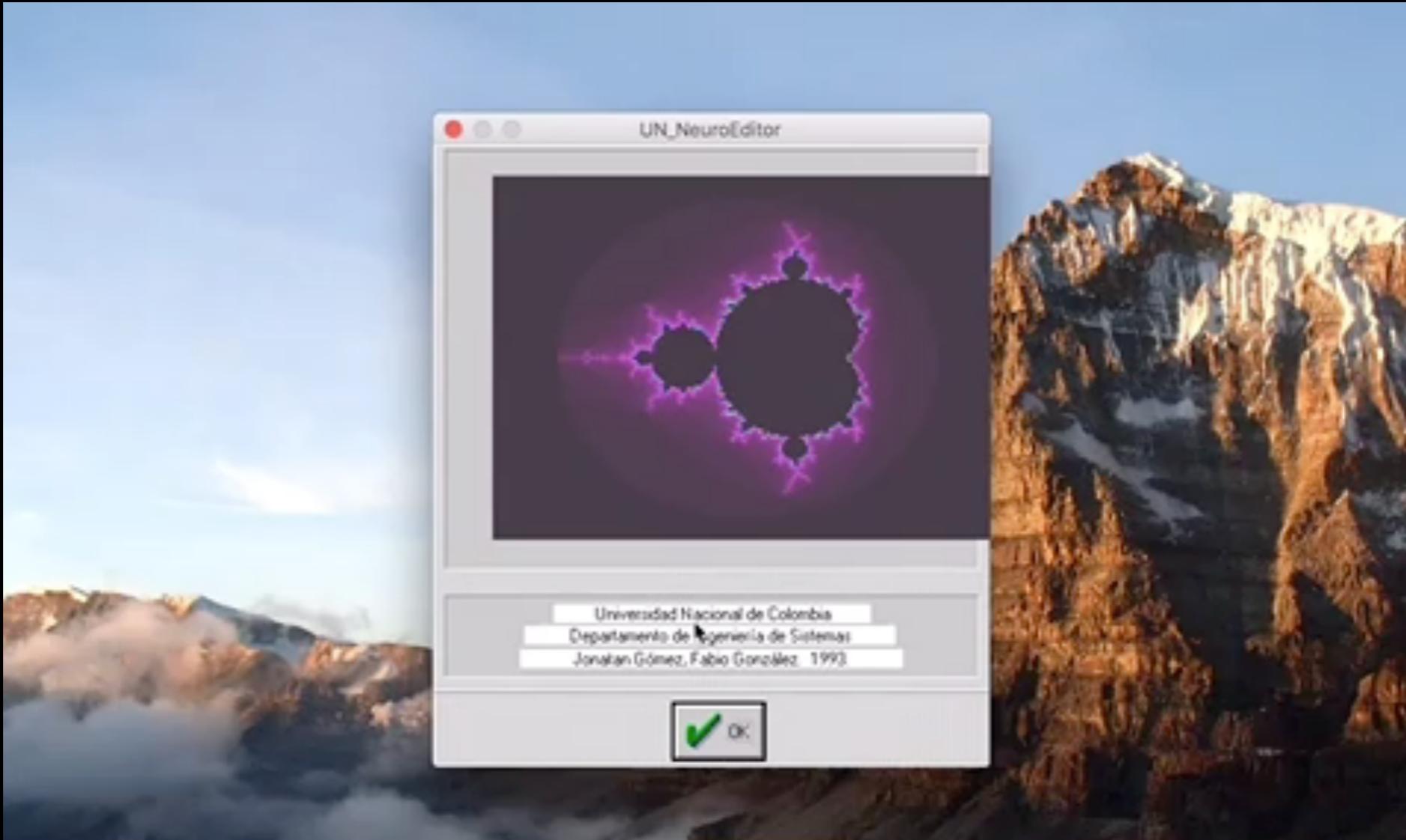
Neural networks time line



My own history with NN (circa 1993)

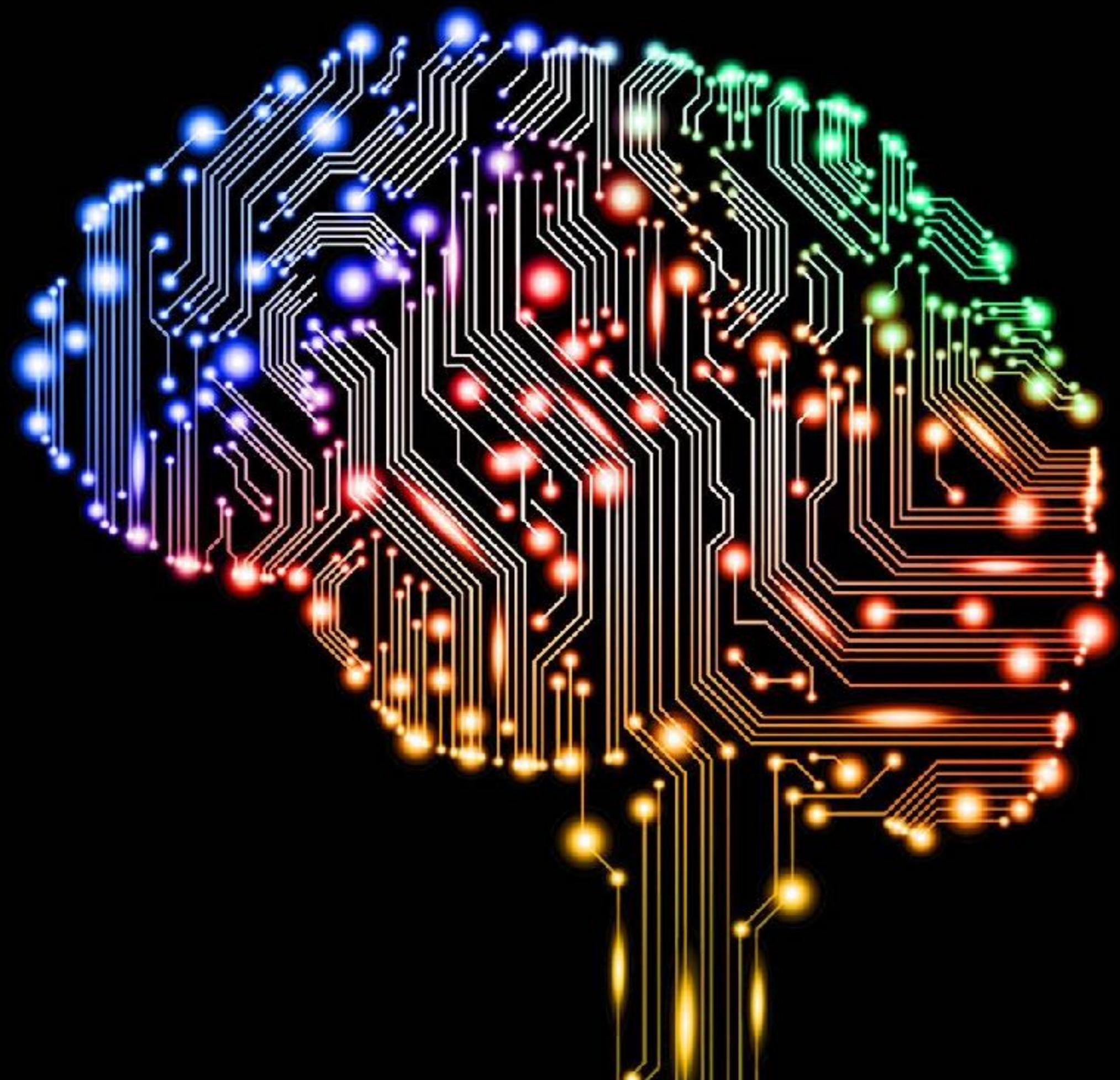


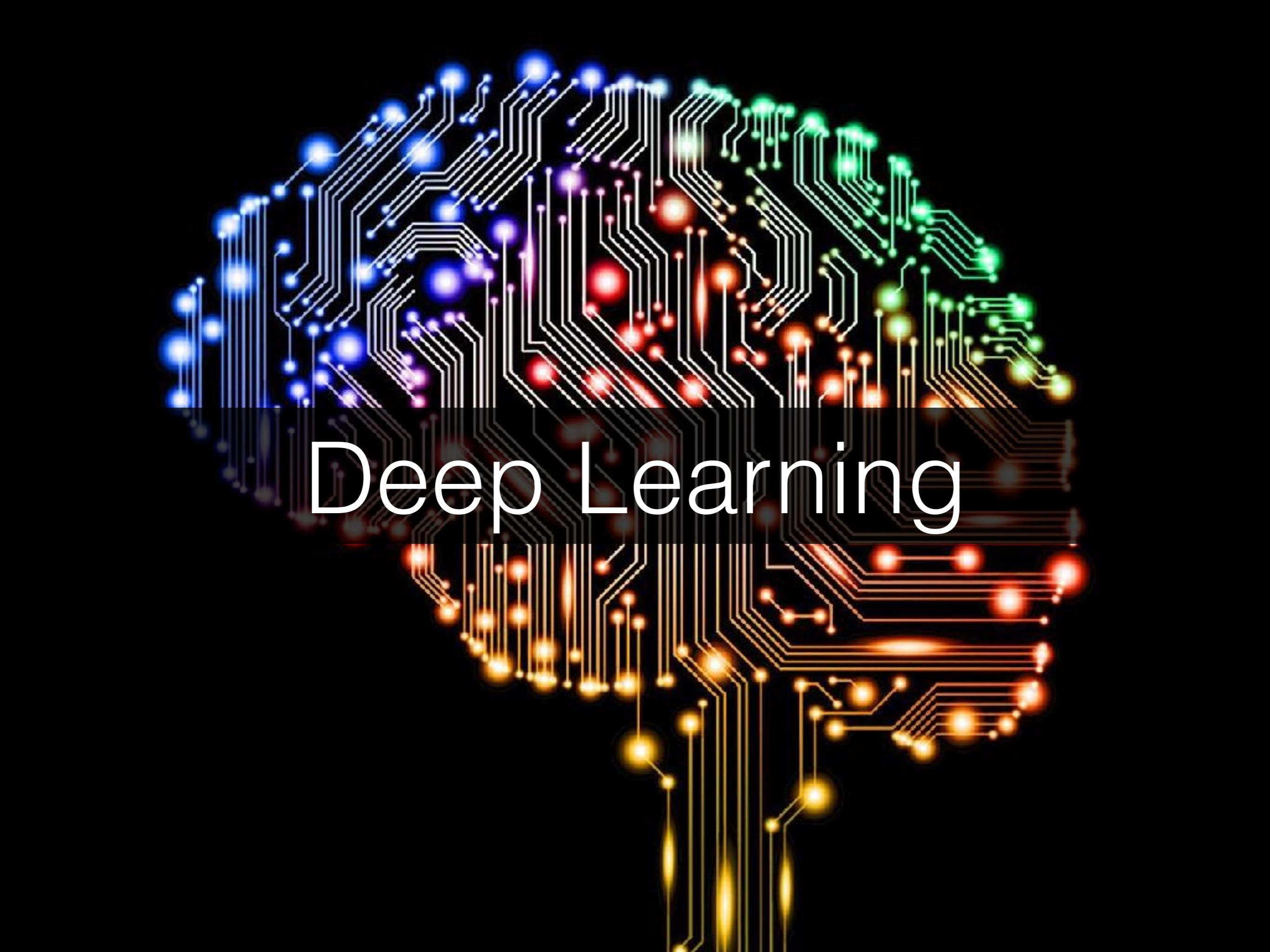
My own history with NN (circa 1993)



NN Demo

<https://playground.tensorflow.org/>





Deep Learning

Deep learning boom



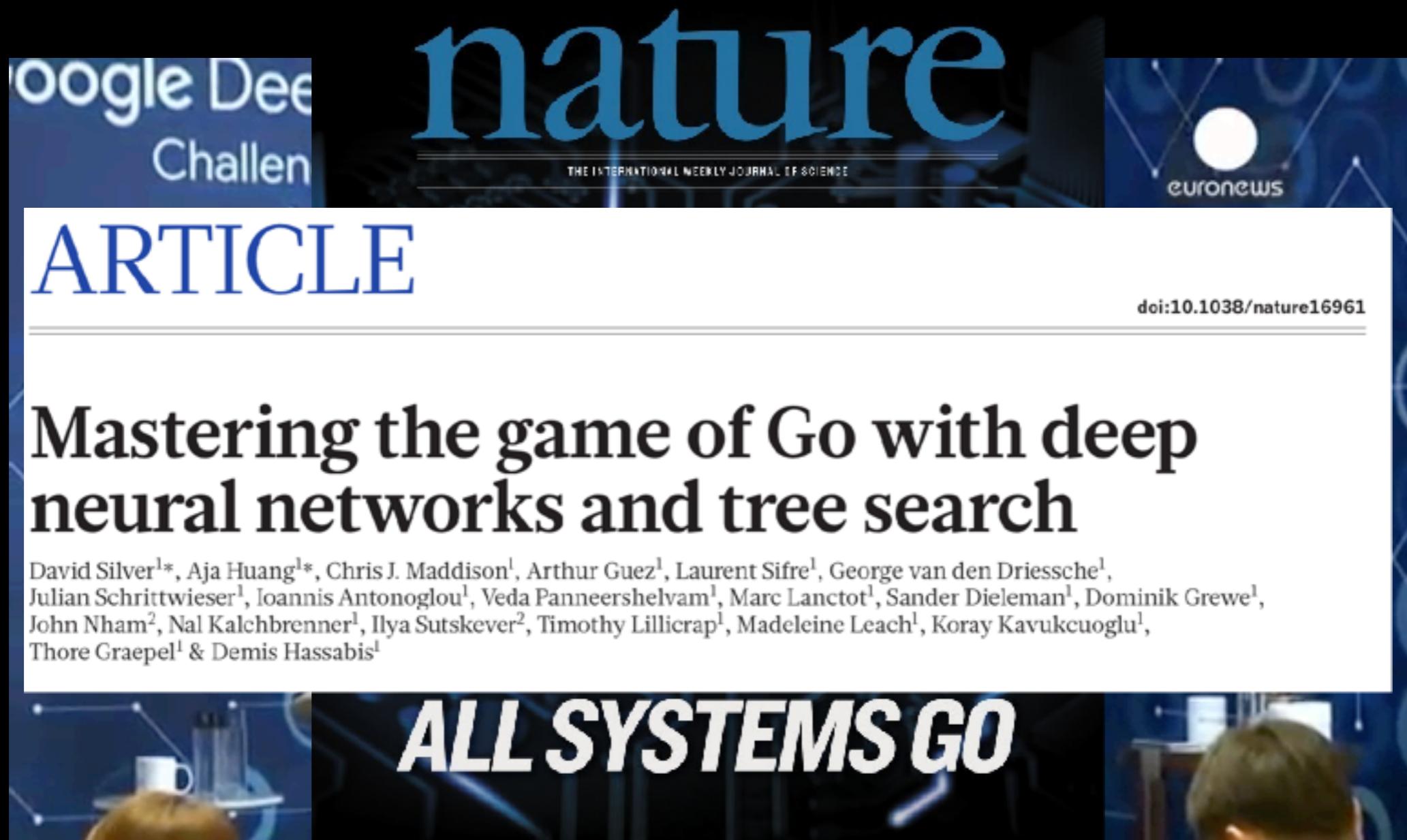
Deep learning boom



Deep learning boom



Deep learning boom



Deep learning boom

Deep learning boom

The screenshot shows a news article from Popular Mechanics. At the top left, there is a green bar with the text "3 Comments" and a speech bubble icon, followed by "2928" and a share icon. To the right of this is a dark gray sidebar with a circular graphic containing a stylized car wheel and a central gear-like element, with blue and green light rays emanating from it. The main content area has a dark gray background. At the top left of this area, there is a small white square icon with a vertical line and the word "DRIVING" in white capital letters. Below this is the article title: "Here's How Deep Learning Will Accelerate Self-Driving Cars" in large, bold, white font. Underneath the title, the text "By Danny Shapiro on February 24, 2015" is written in a smaller, white, sans-serif font.

3 Comments 2928

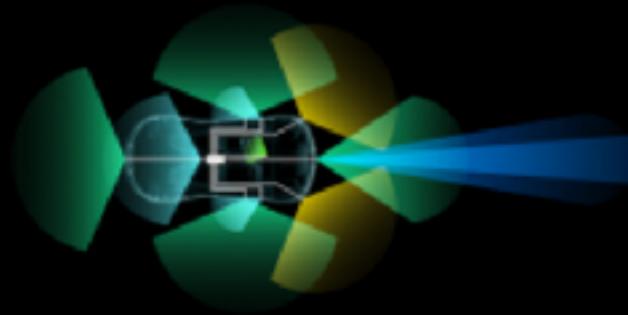
DRIVING

Here's How Deep Learning Will Accelerate Self-Driving Cars

By Danny Shapiro on February 24, 2015

Deep learning boom

3 Comments  2928 



DRIVING

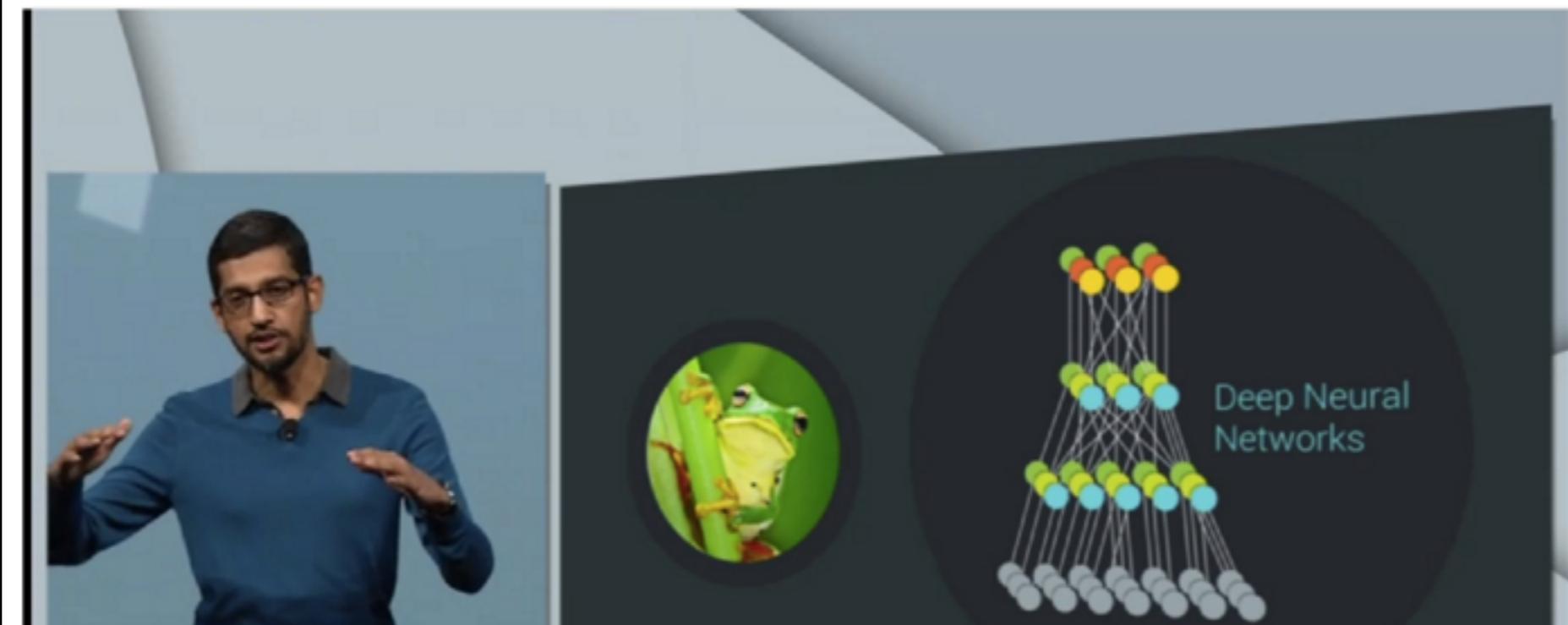
Here's How Deep Learning Will Accelerate Self-Driving Cars

By [Danny Shapiro](#) on February 24, 2015

Google says its speech recognition technology now has only an 8% word error rate

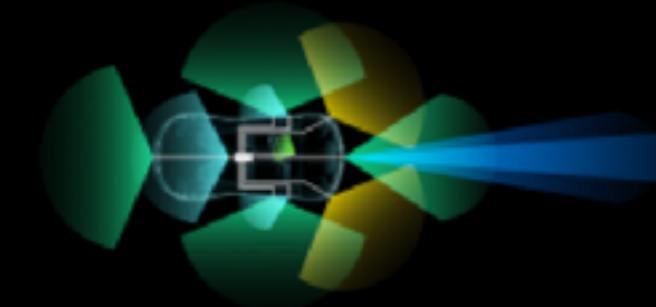
JORDAN NOVET MAY 28, 2015 10:40 AM

TAGS: [ARTIFICIAL INTELLIGENCE](#), [DEEP LEARNING](#), [GOOGLE](#), [GOOGLE I/O 2015](#), [SUNDAR PICHAI](#)



Deep learning boom

3 Comments  2928 



DRIVING

Here's How Deep Learning Will Accelerate Self-Driving Cars

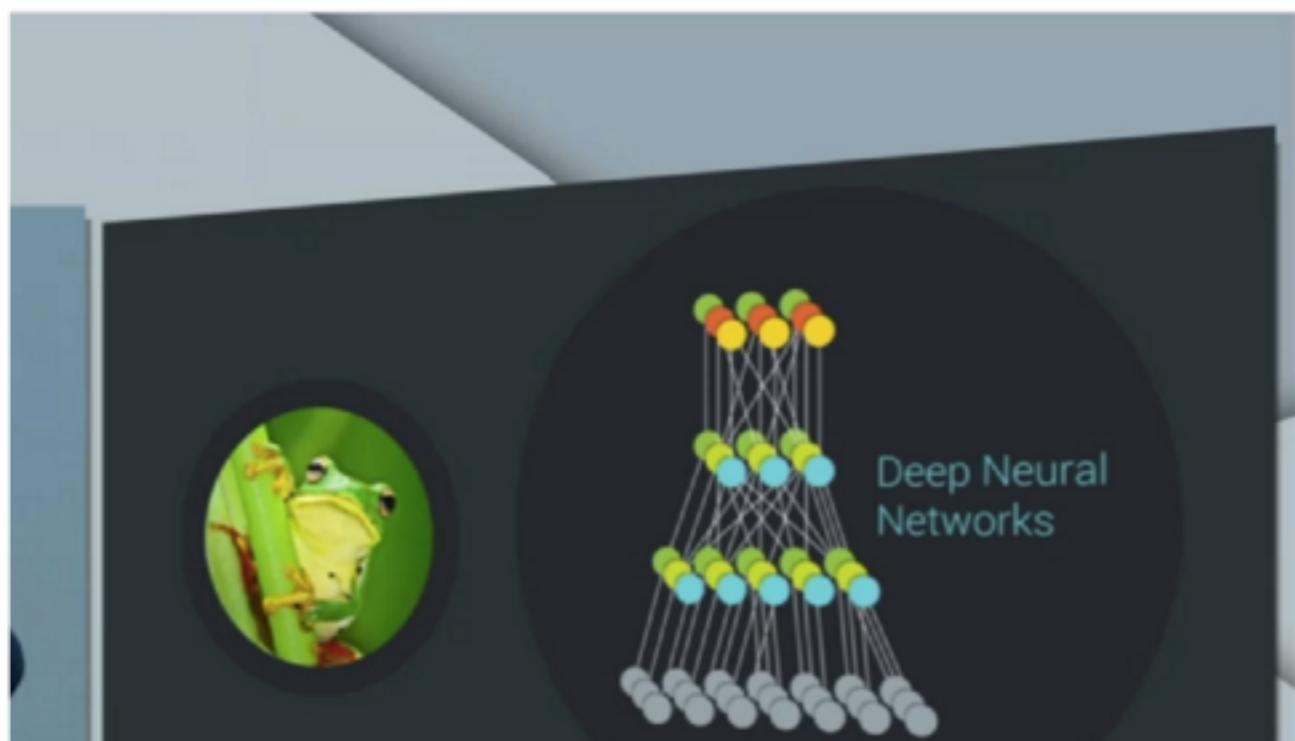
By [Danny Shapiro](#) on February 24, 2015

ROBERT MCMILLAN BUSINESS 03.13.13 6:30 AM

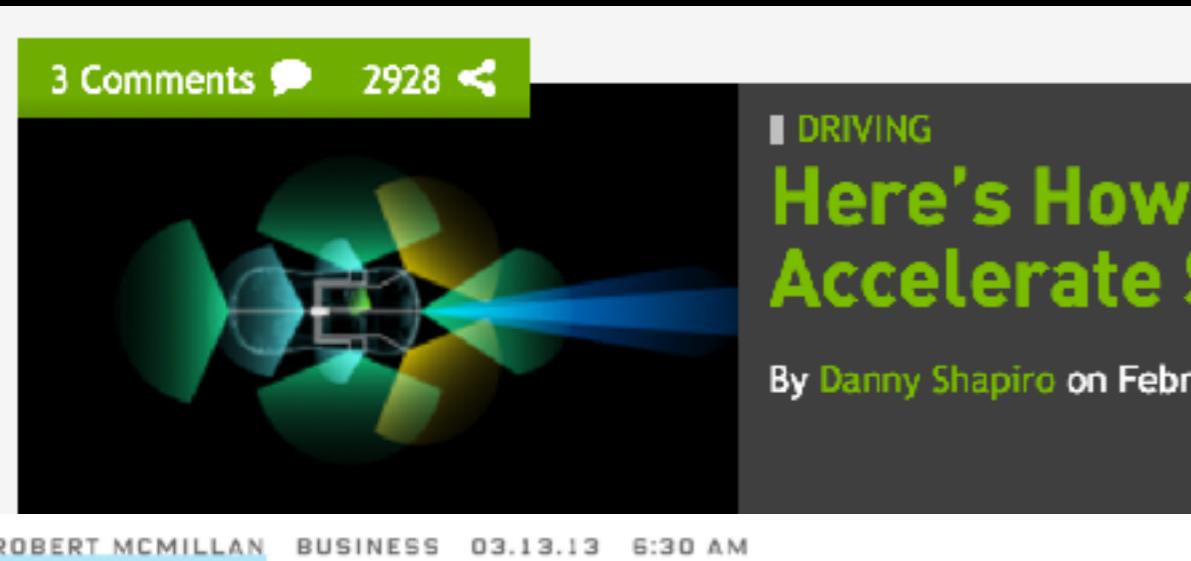
GOOGLE HIRES BRAINS THAT HELPED SUPERCHARGE MACHINE LEARNING

speech recognition technology now word error rate

ING, GOOGLE, GOOGLE I/O 2015, SUNDAR PICHAI

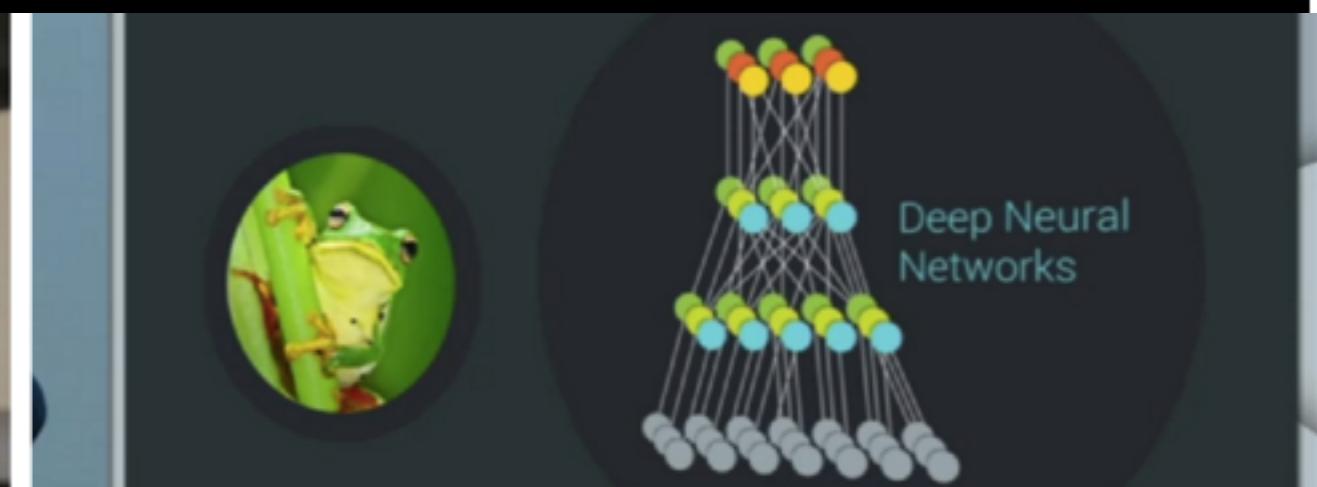
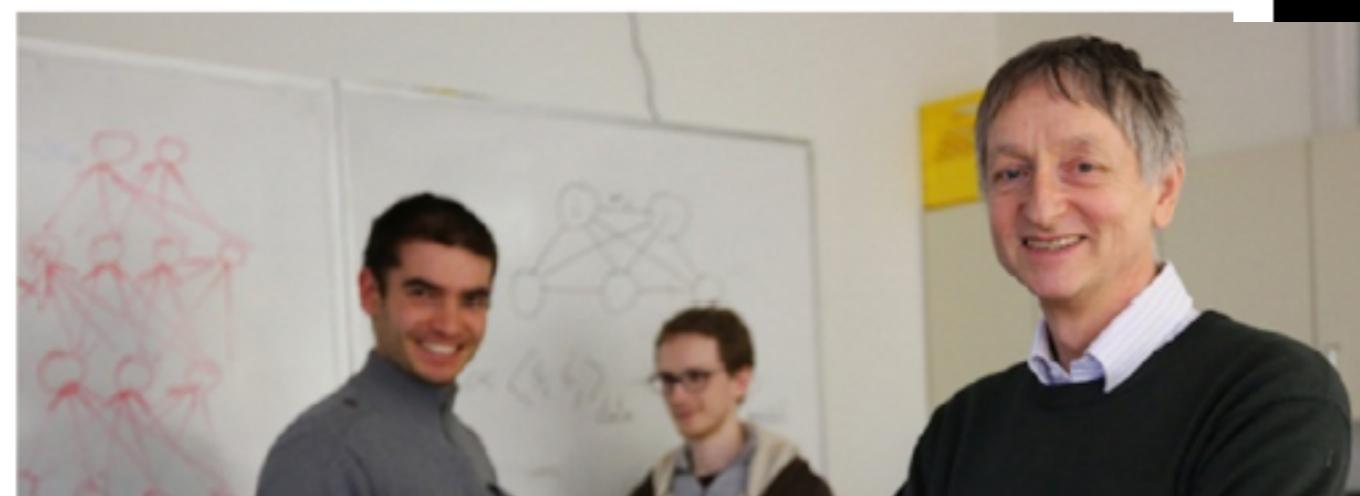


Deep learning boom

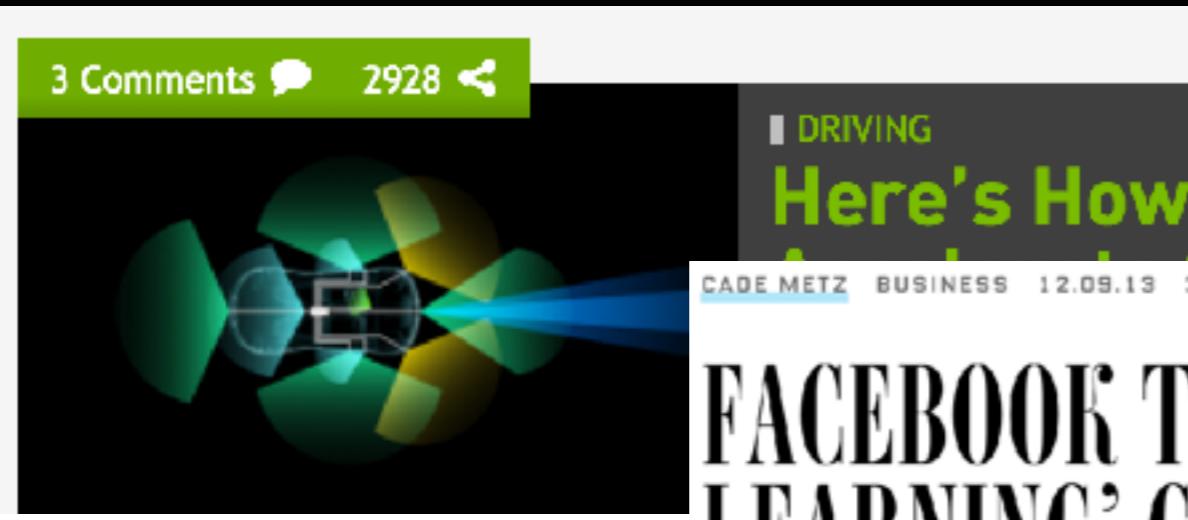


GOOGLE HIRES BRAINS THAT HELPED SUPERCHARGE MACHINE LEARNING

Chinese Search Giant Baidu Hires Man Behind the “Google Brain”



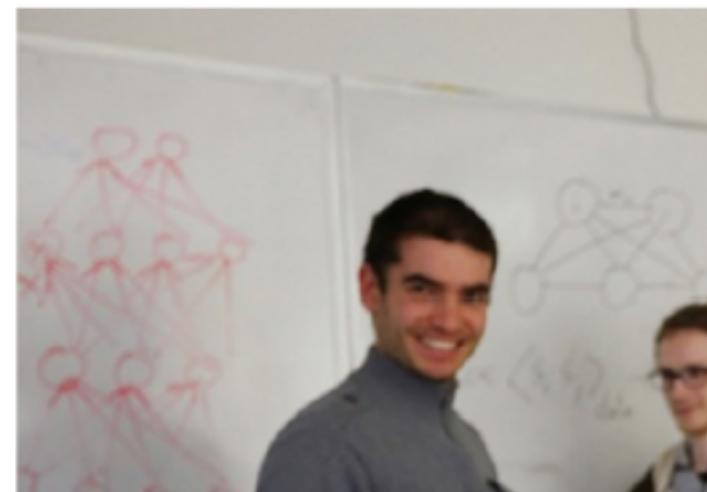
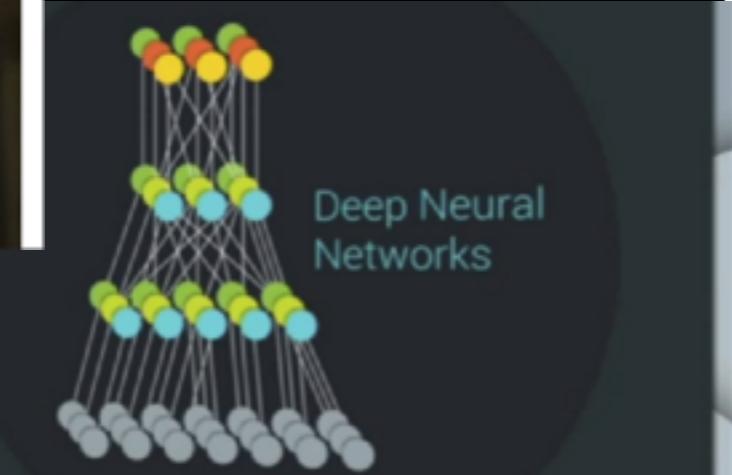
Deep learning boom



MIT
Technology
Review

FACEBOOK TAPS 'DEEP
LEARNING' GIANT FOR NEW AI
LAB

Search Giant
Man Behind
Brain"



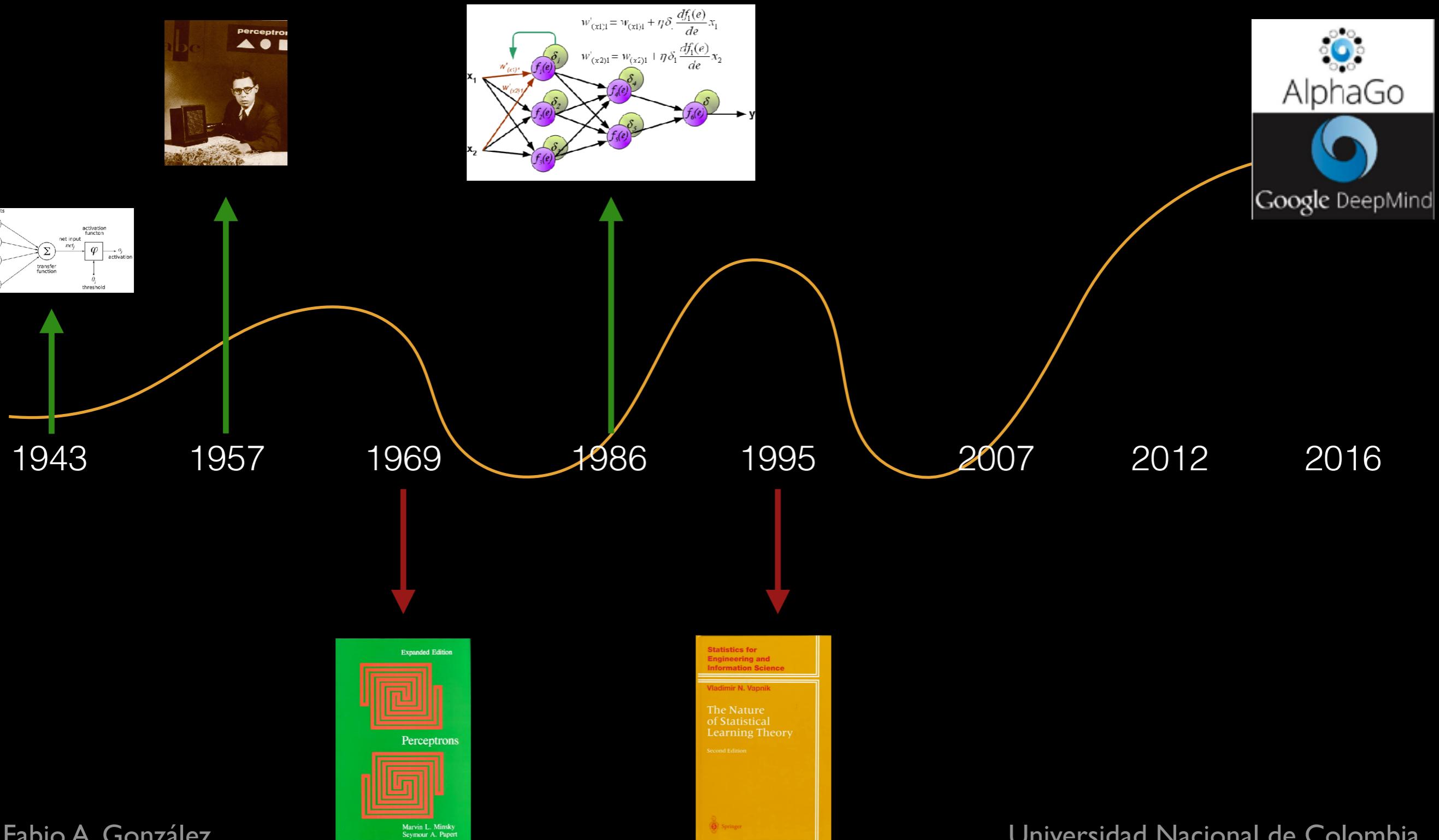
Google assistant calling to make a reservation



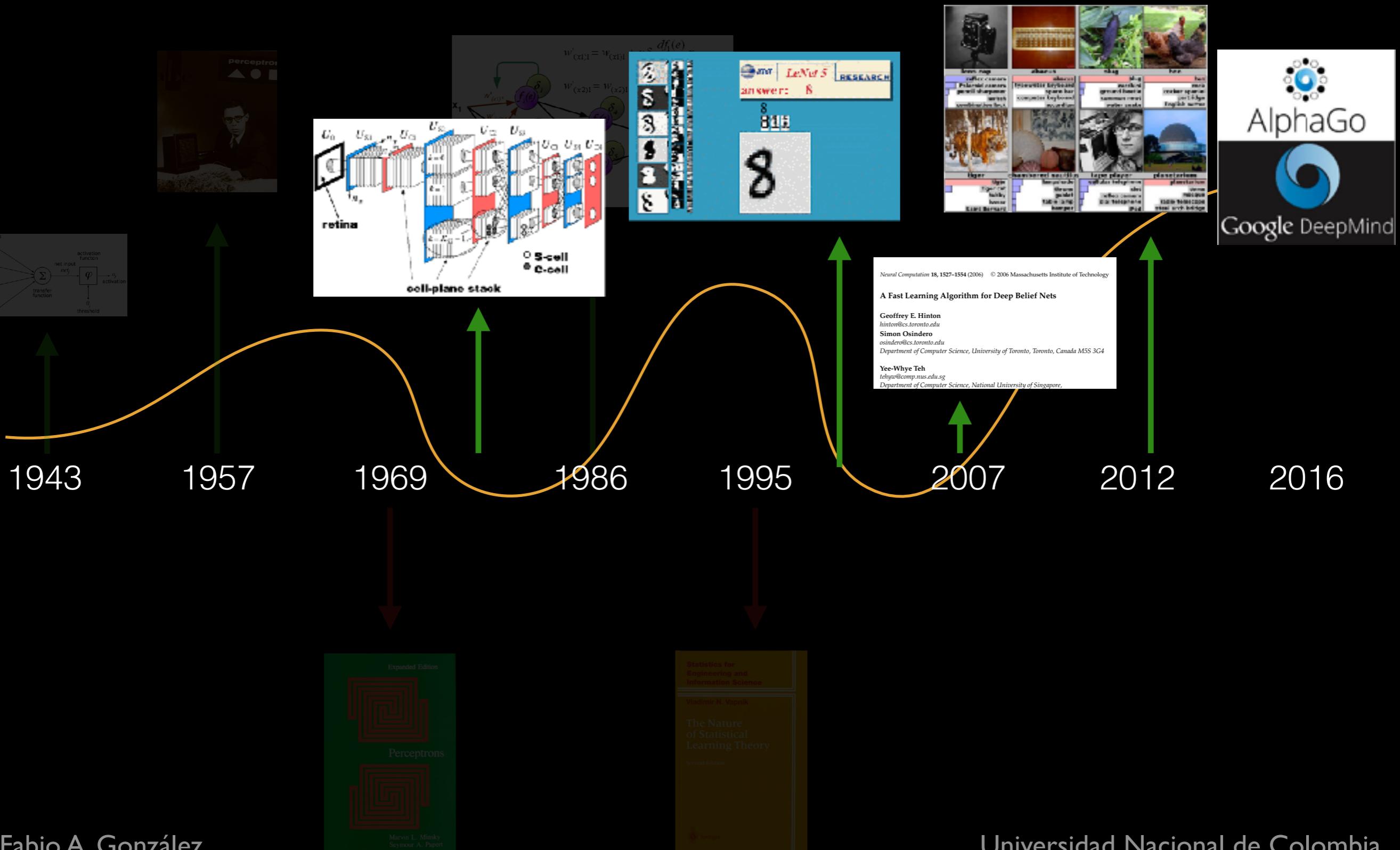
Google assistant calling to make a reservation



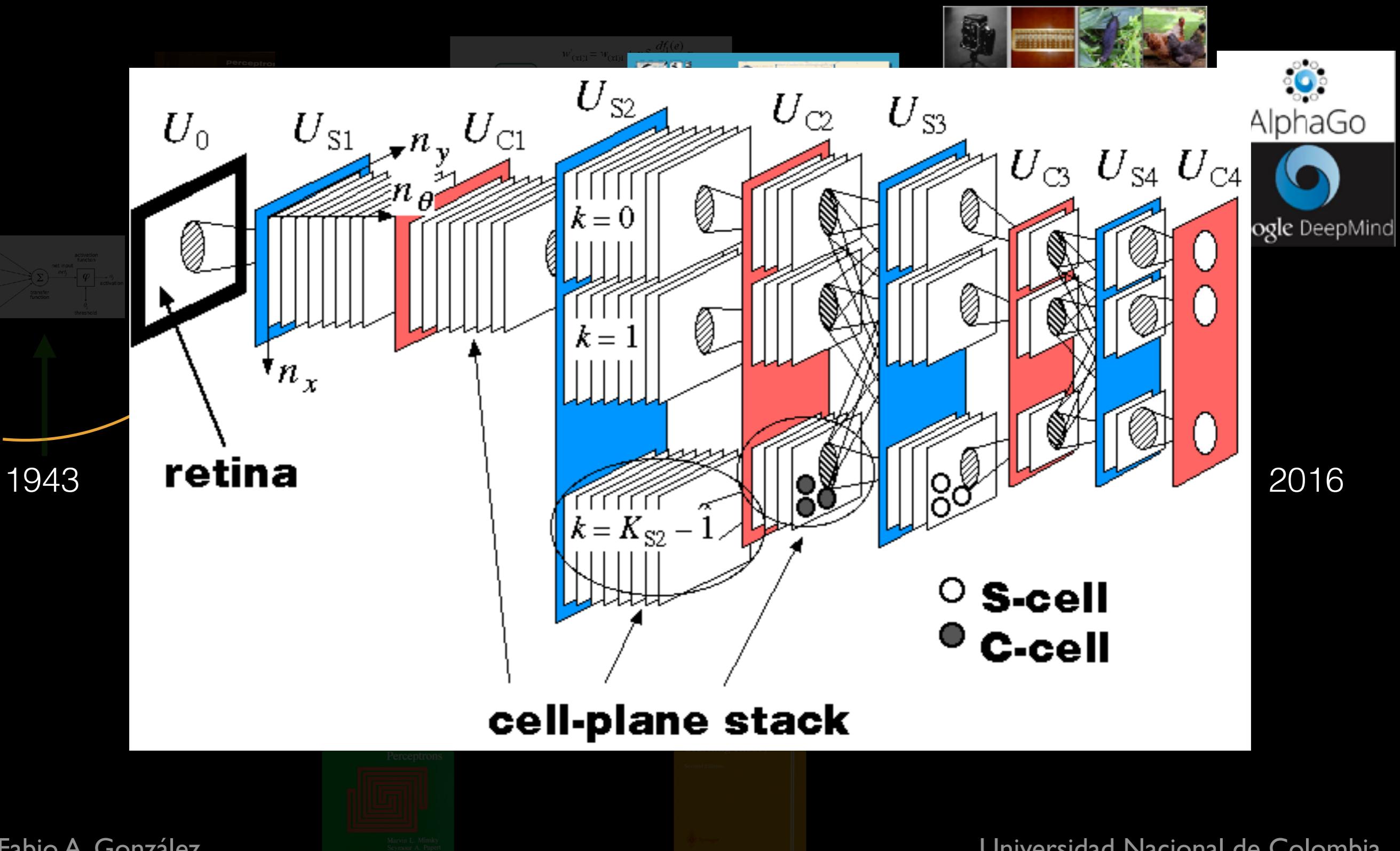
Deep learning time line



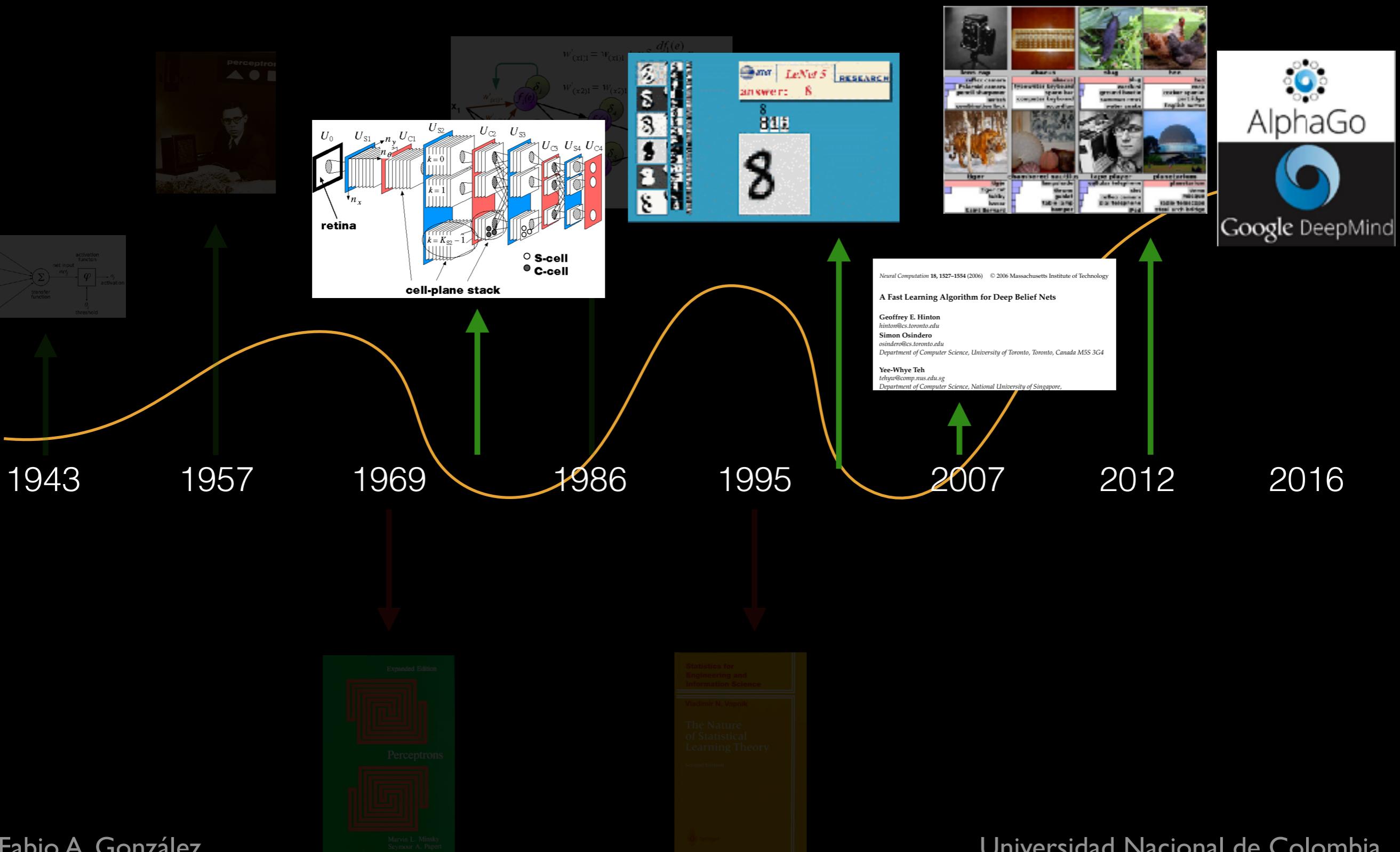
Deep learning time line



Deep learning time line



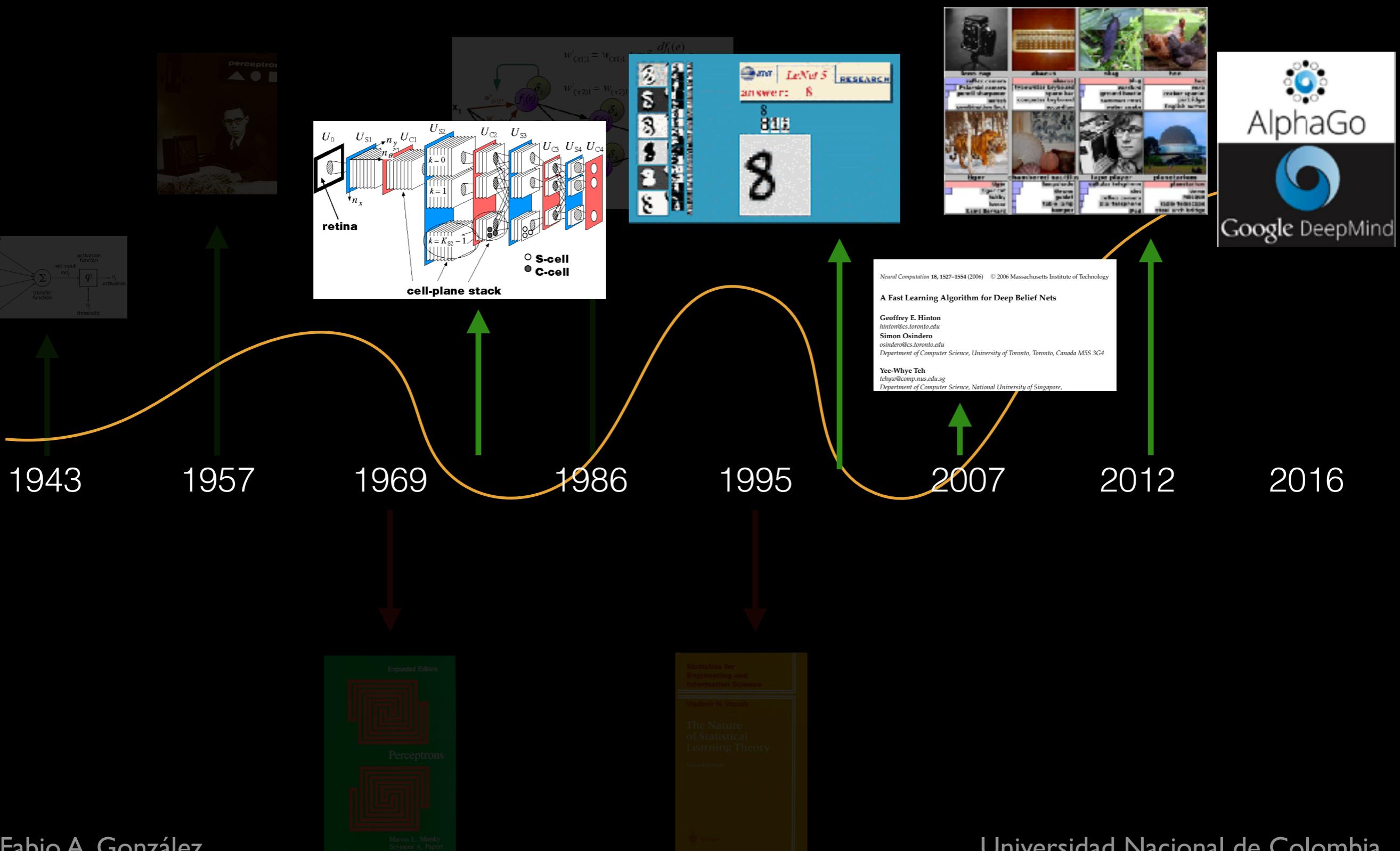
Deep learning time line



Deep learning time line



Deep learning time line



Deep Learning is Born

Neural Computation **18**, 1527–1554 (2006) © 2006 Massachusetts Institute of Technology

A Fast Learning Algorithm for Deep Belief Nets

Geoffrey E. Hinton

hinton@cs.toronto.edu

Simon Osindero

osindero@cs.toronto.edu

Department of Computer Science, University of Toronto, Toronto, Canada M5S 3G4

Yee-Whye Teh

tehyw@comp.nus.edu.sg

Department of Computer Science, National University of Singapore,

Deep Learning is Born

Neural Computation 18, 1527–1554 (2006)

© 2006 Massachusetts Institute of Technology

A Fast Learning Algorithm for Sparse Coding and Image Recognition¹⁵²⁸

Geoffrey E. Hinton

hinton@cs.toronto.edu

Simon Osindero

osindero@cs.toronto.edu

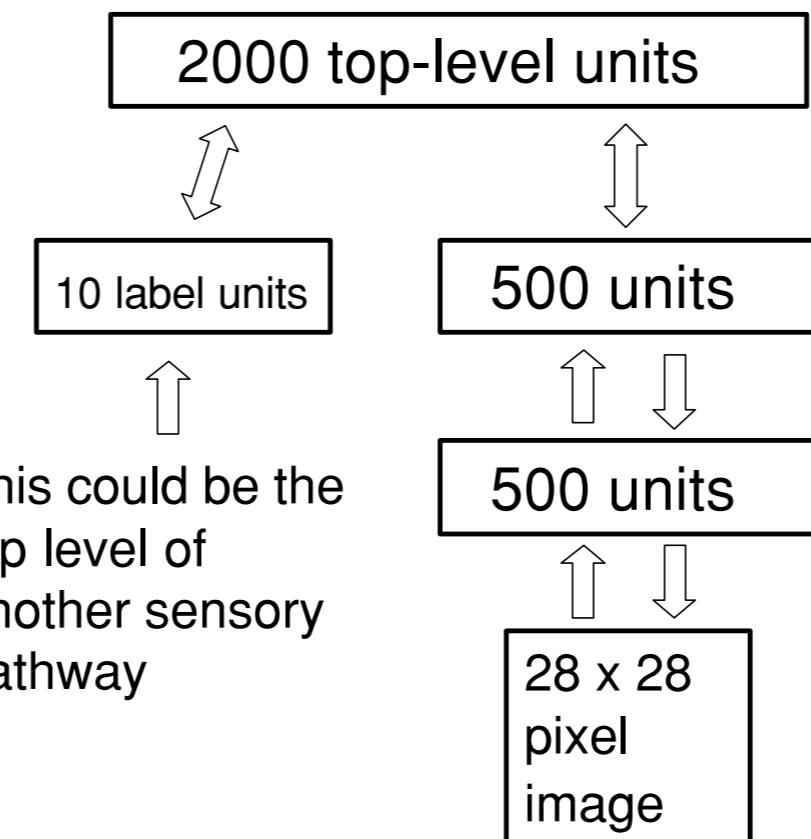
Department of Computer Science
University of Toronto

Yee-Whye Teh

tehyw@comp.nus.edu.sg

Department of Computer Science
National University of Singapore

G. Hinton, S. Osindero, and Y.-W. Teh



Deep Learning is Born

Neural Computation 18, 1527–1554 (2006) © 2006 Massachusetts Institute of Technology

A Fast Learning Algorithm for Sparse Coding Using Correlation-Based Hebbian Learning

1528

G. Hinton, S. Osindero, and Y.-W. Teh

Geoffrey E. Hinton

hinton@cs.toronto.edu

Simon Osindero

osindero@cs.toronto.edu

Department of Computer Science
University of Toronto

Yee-Whye Teh

tehyw@comp.nus.edu.sg

Department of Computer Science
National University of Singapore

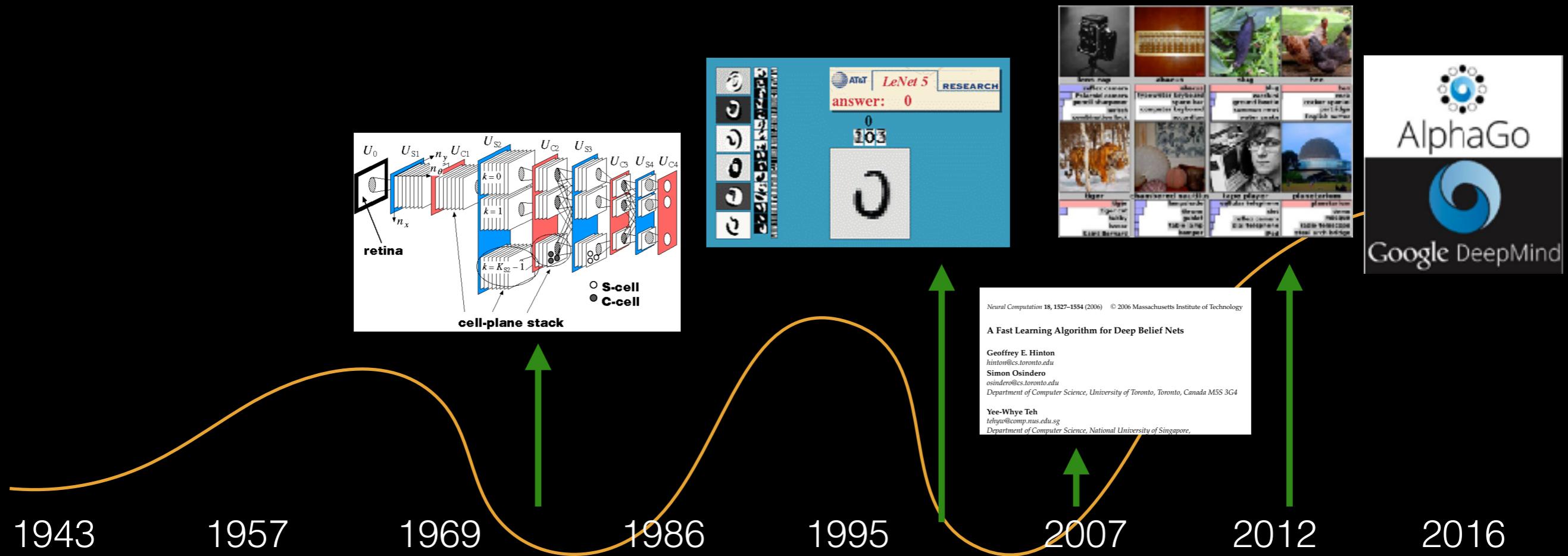
2000 top-level units

10 labels

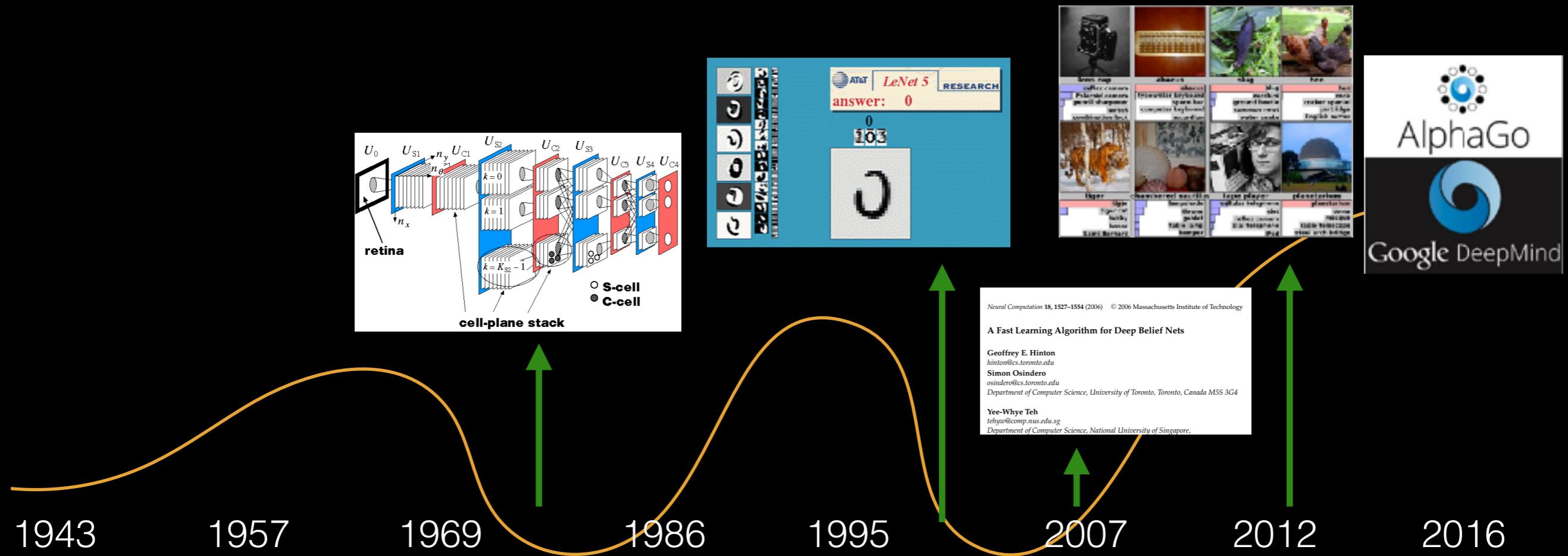
This could be the
top level of
another set
of pathways



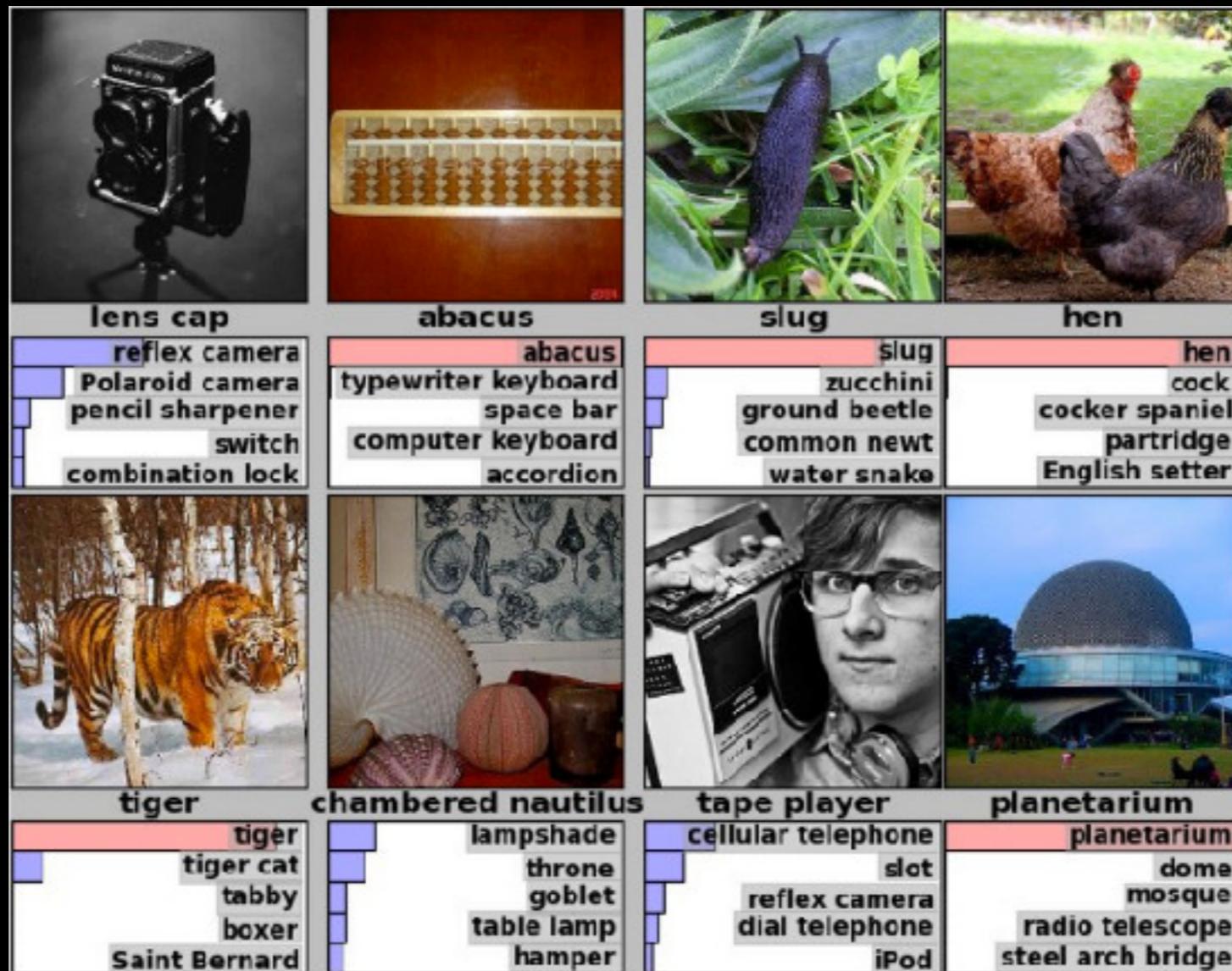
Deep learning time line



Deep learning time line



Deep learning model won ILSVRC 2012 challenge



Deep learning model won ILSVRC 2012 challenge

- ILSVRC 2012 (ImageNet
Large Scale Visual
Recognition)

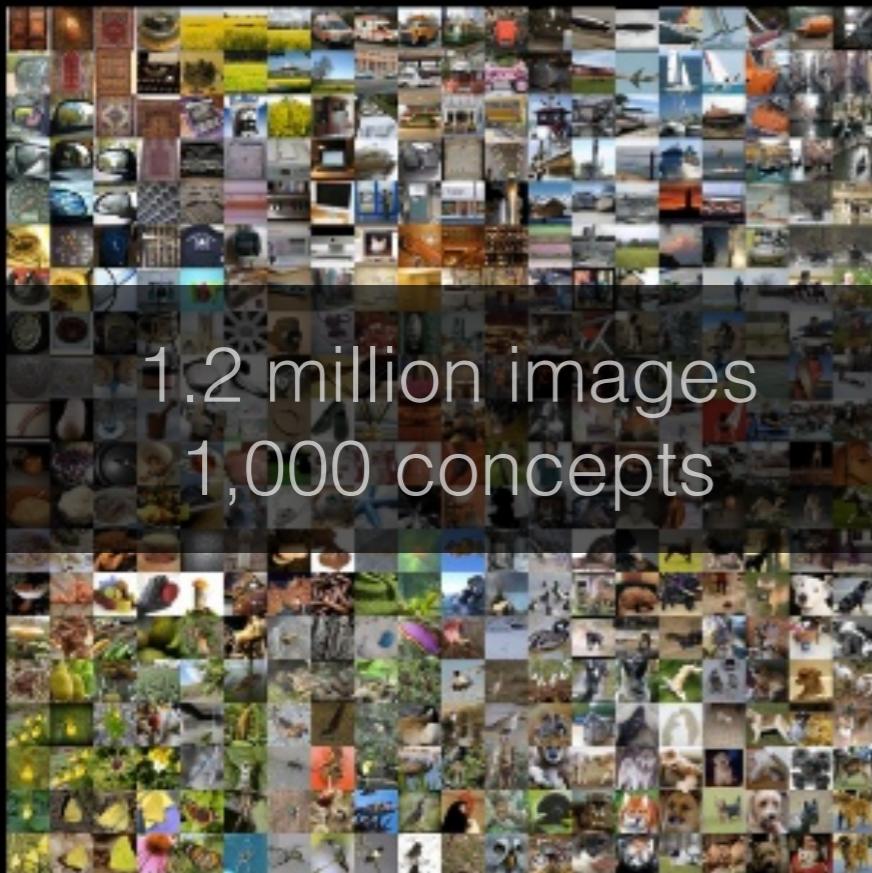
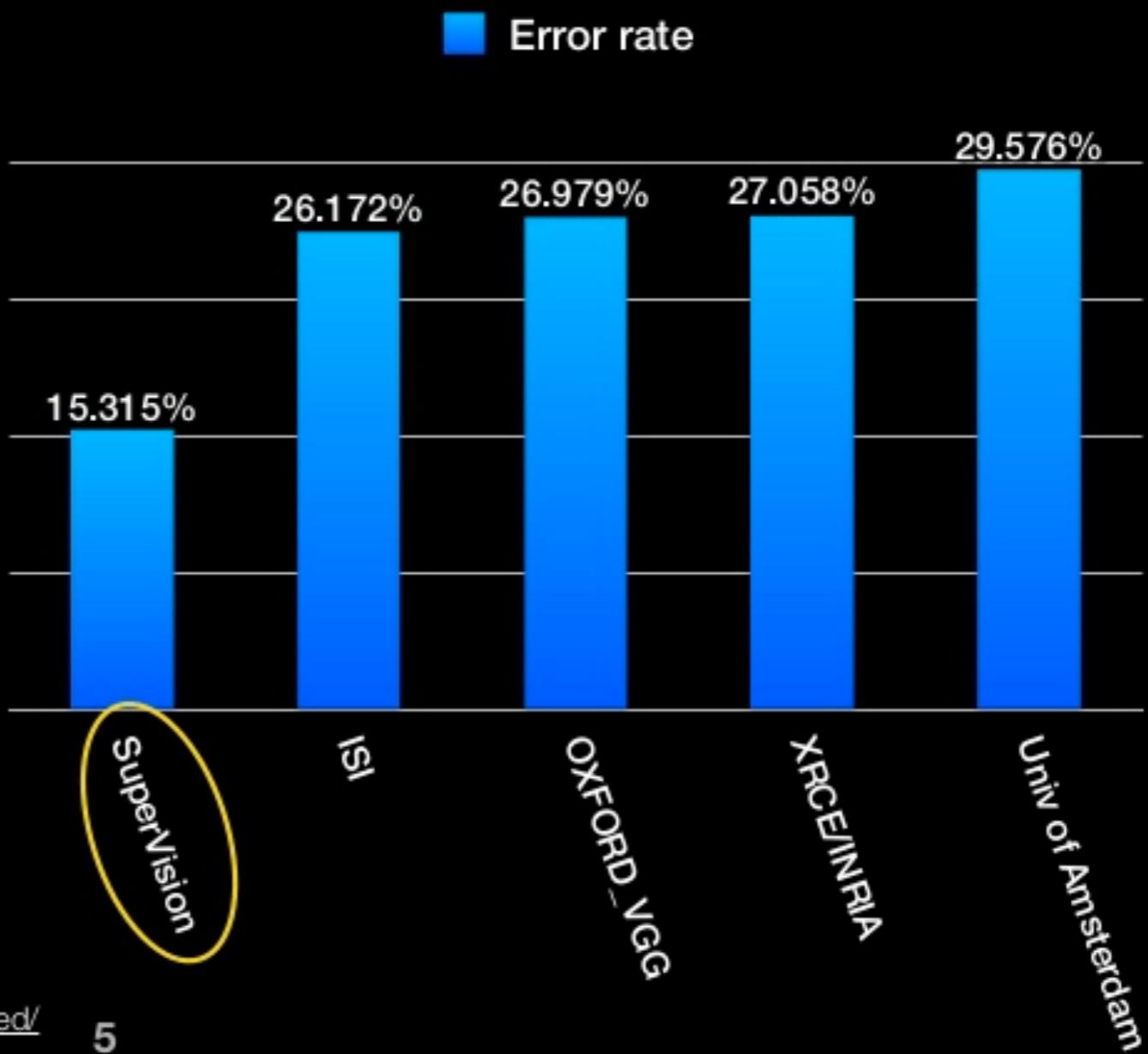


Image source: <http://cs.stanford.edu/people/karpathy/cnnembed/>



Deep learning model won ILSVRC 2012 challenge

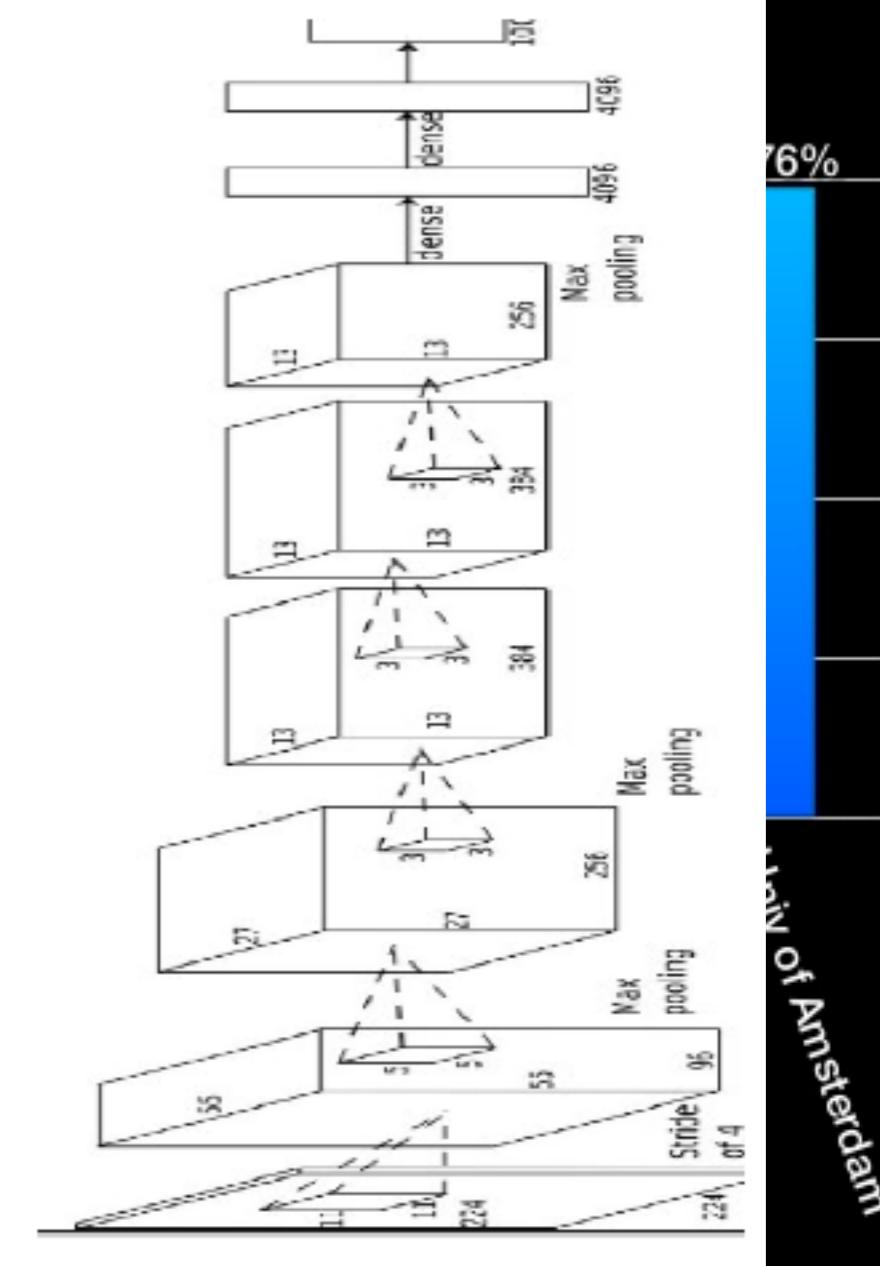
- ILSVRC Large Recognition



Image source

■ Won the 2012 ImageNet LSVRC. 60 Million parameters, 832M MAC ops

4M	FULL CONNECT	4Mflop
16M	FULL 4096/ReLU	16M
37M	FULL 4096/ReLU	37M
	MAX POOLING	
442K	CONV 3x3/ReLU 256fm	74M
1.3M	CONV 3x3ReLU 384fm	224M
884K	CONV 3x3/ReLU 384fm	149M
	MAX POOLING 2x2sub	
307K	LOCAL CONTRAST NORM	
	CONV 11x11/ReLU 256fm	223M
	MAX POOL 2x2sub	
35K	LOCAL CONTRAST NORM	
	CONV 11x11/ReLU 96fm	105M



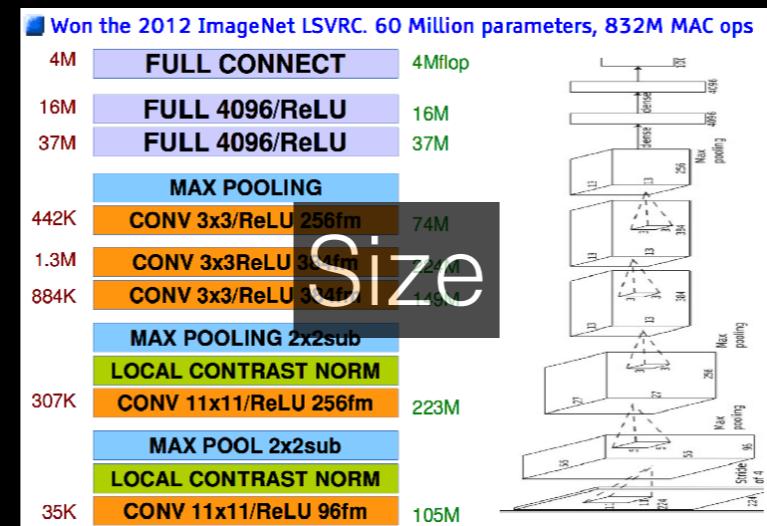
Deep learning recipe

Data

Algorithms

Feature
learning

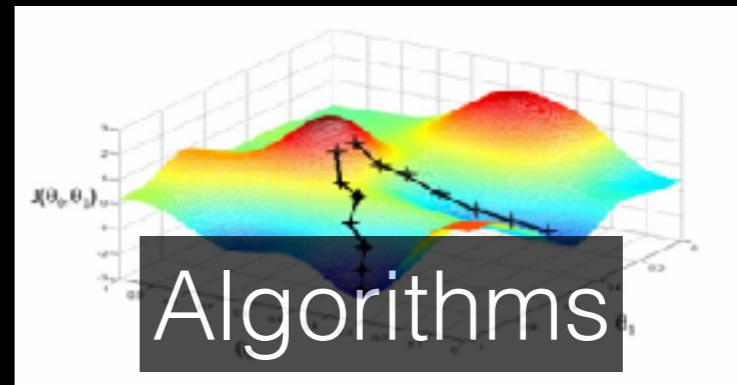
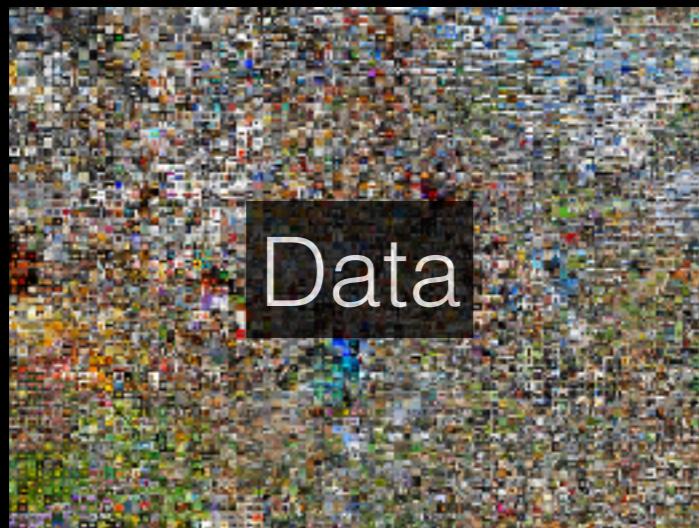
Size



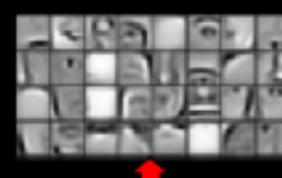
Tricks

HPC

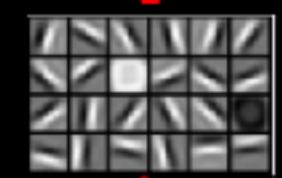
Deep learning recipe



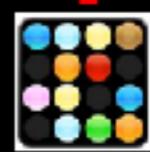
object models



object parts
(combination
of edges)



Feature
learning
edges

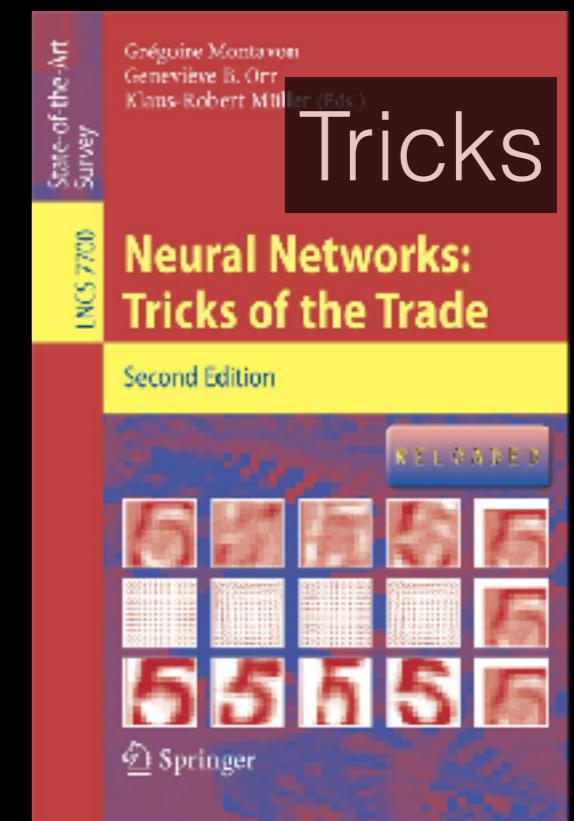
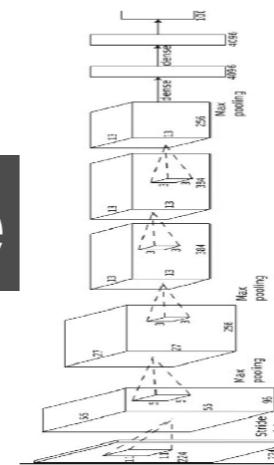


pixels

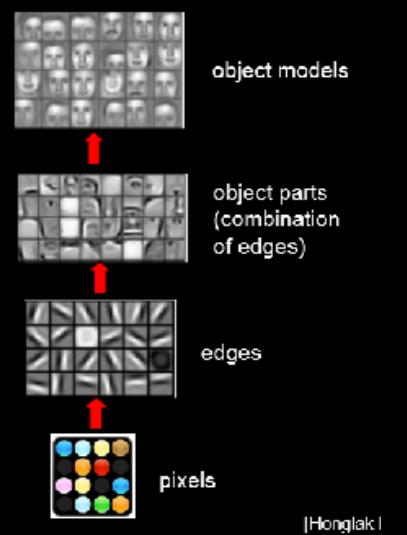
(Honglak)

Won the 2012 ImageNet LSVRC. 60 Million parameters, 832M MAC ops		
4M	FULL CONNECT	4Mflop
16M	FULL 4096/ReLU	16M
37M	FULL 4096/ReLU	37M
	MAX POOLING	
442K	CONV 3x3/ReLU 256fm	74M
1.3M	CONV 3x3ReLU 384fm	124M
884K	CONV 3x3ReLU 384fm	149M
	MAX POOLING 2x2sub	
	LOCAL CONTRAST NORM	
307K	CONV 11x11/ReLU 256fm	223M
	MAX POOL 2x2sub	
	LOCAL CONTRAST NORM	
35K	CONV 11x11/ReLU 96fm	105M

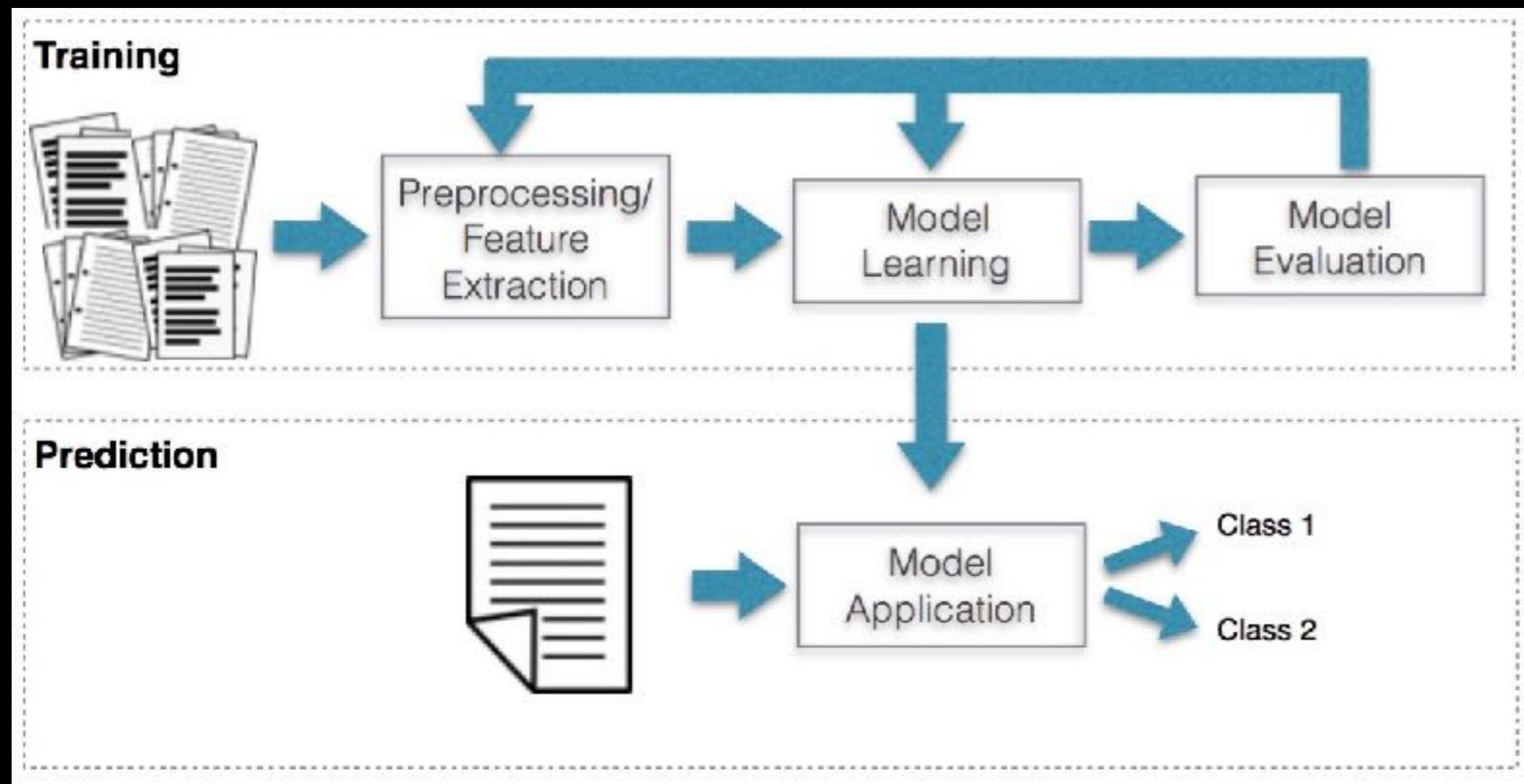
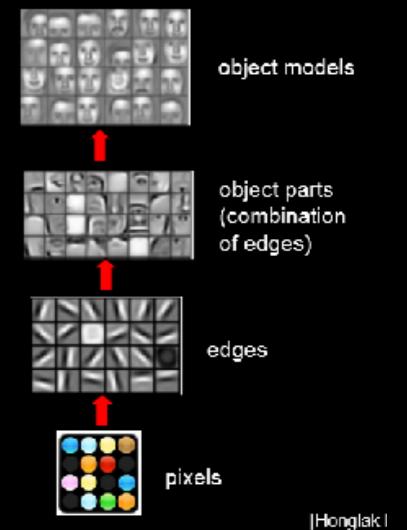
Size



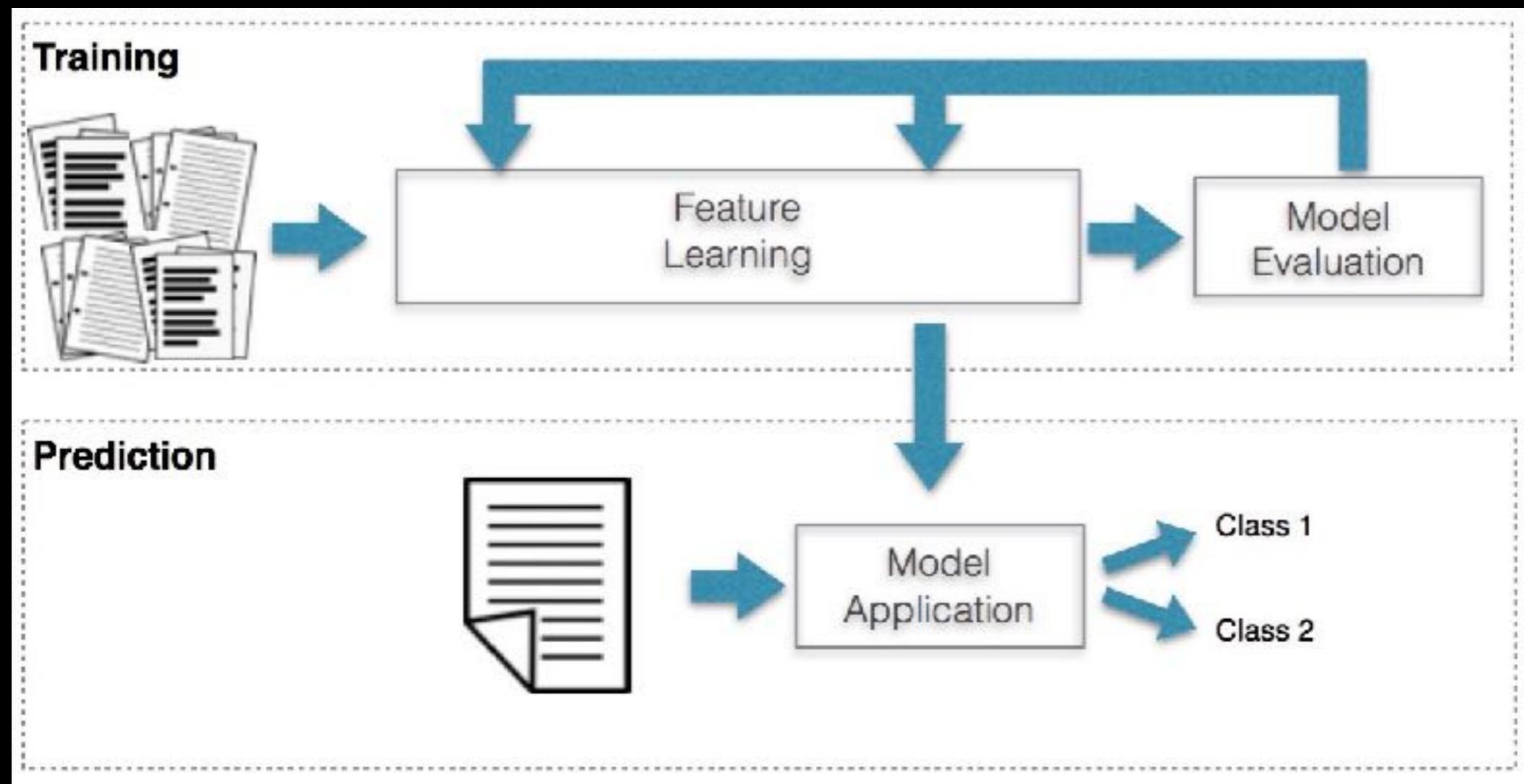
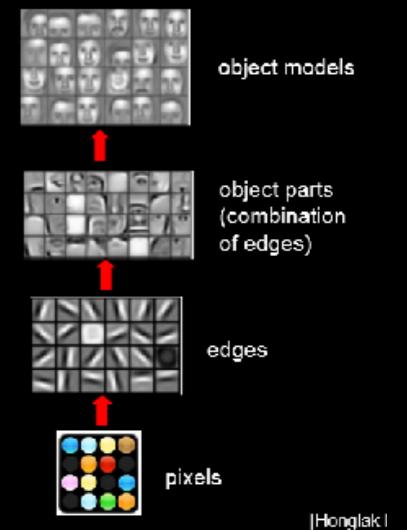
Feature learning



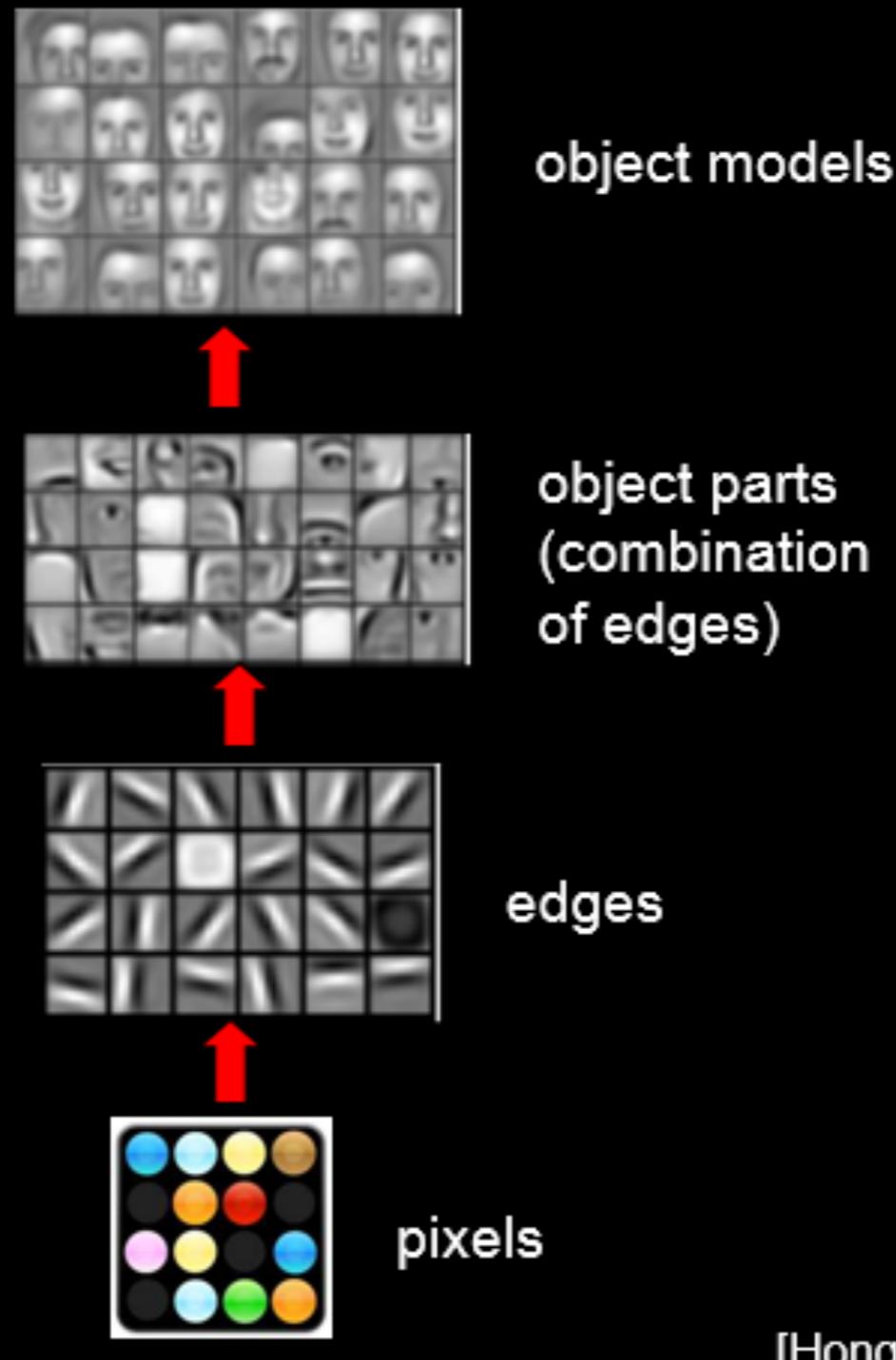
Feature learning



Feature learning

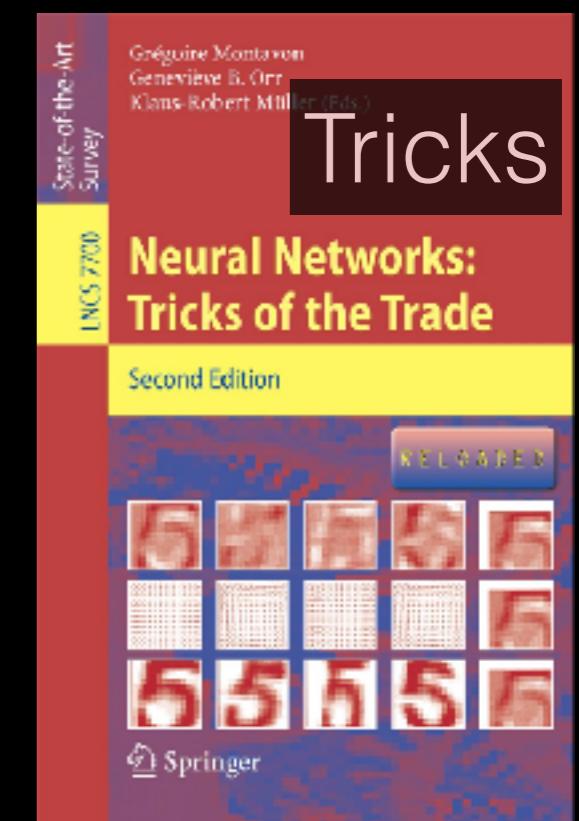
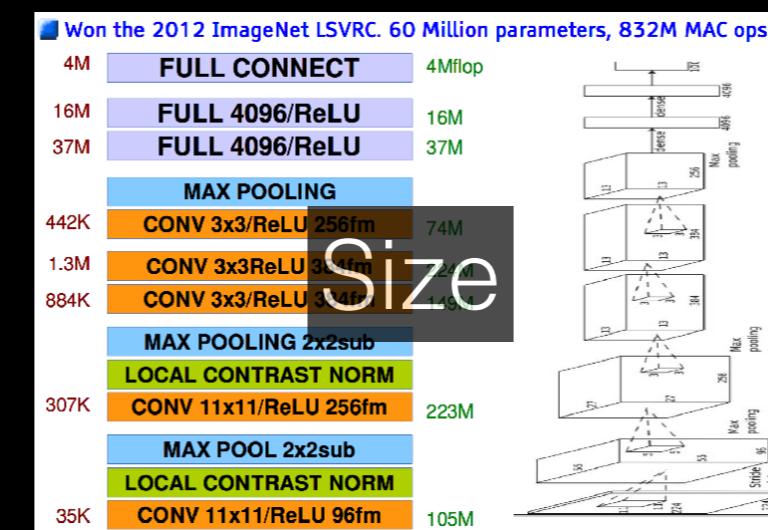
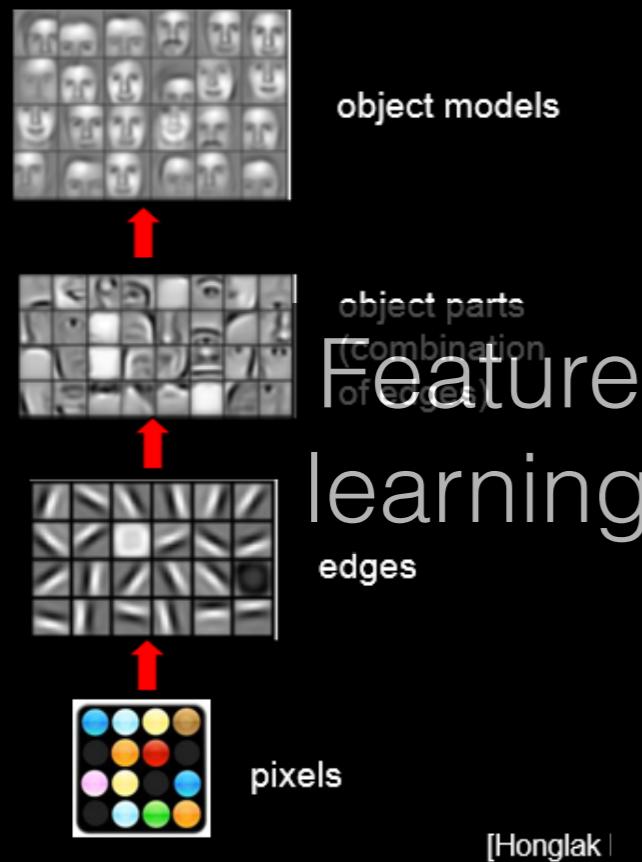
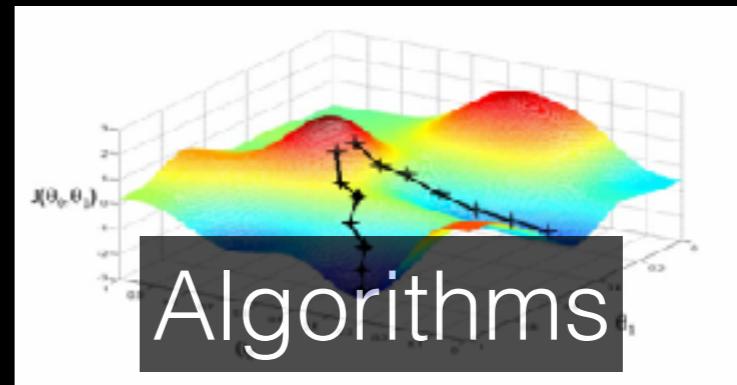
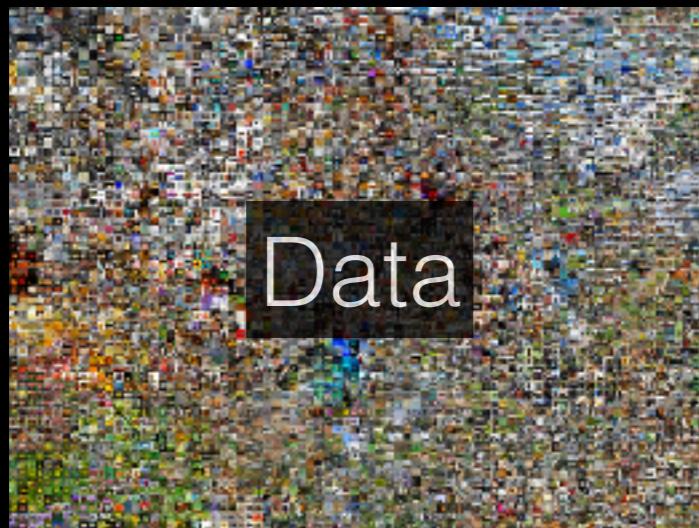


Feature learning

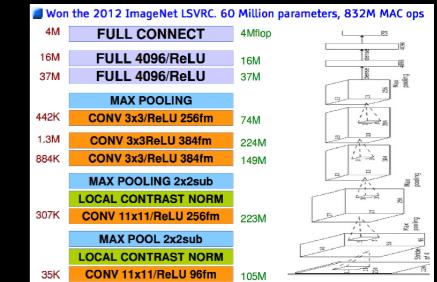


[Honglak |

Deep learning recipe

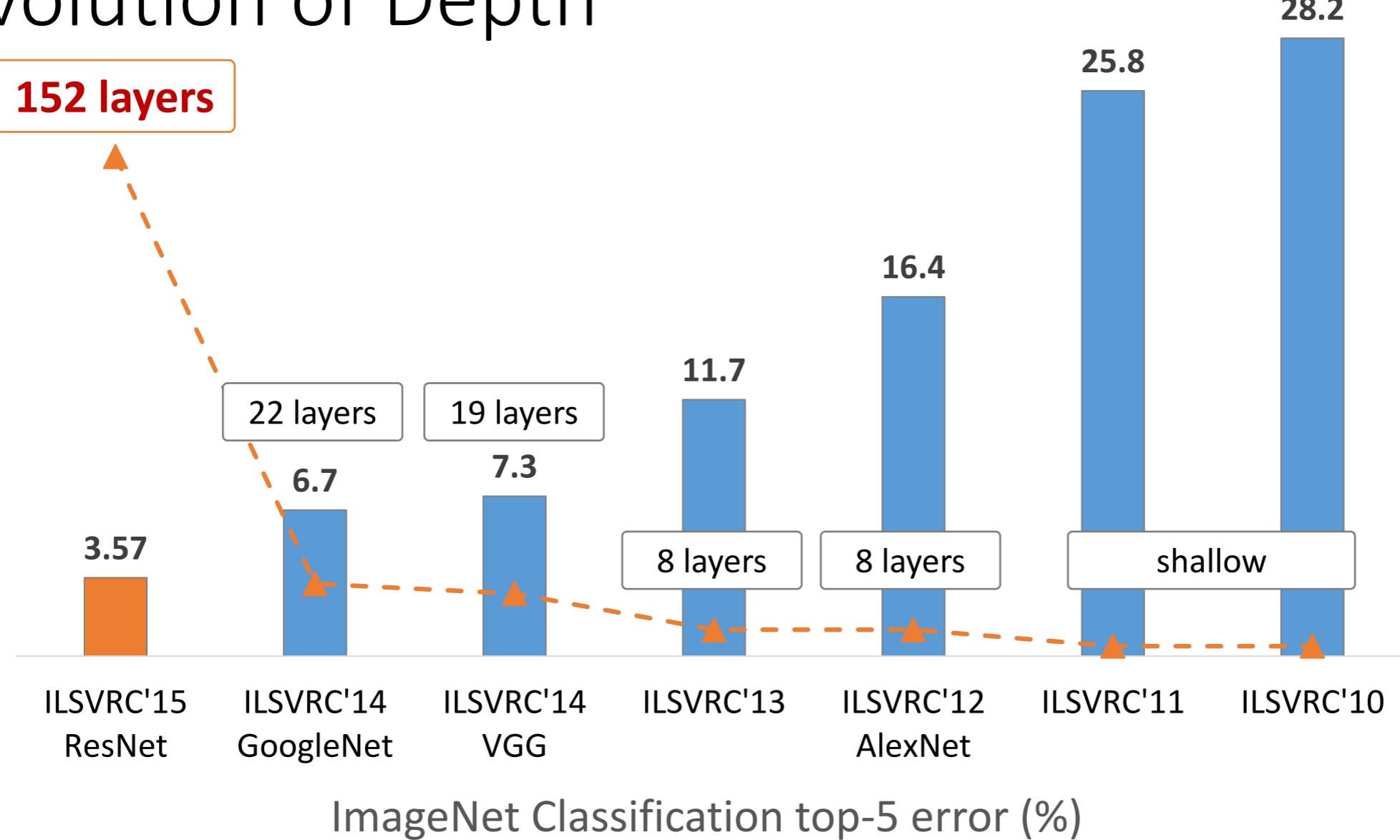


Deep → Bigger

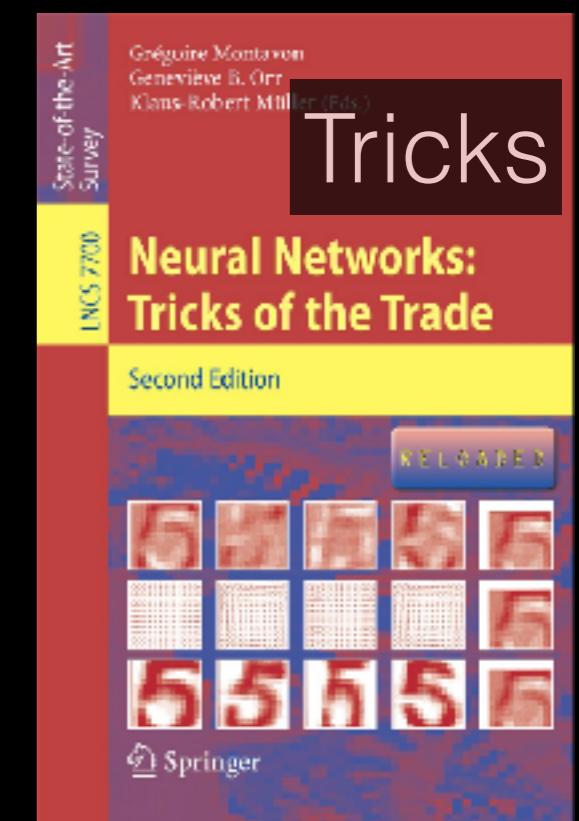
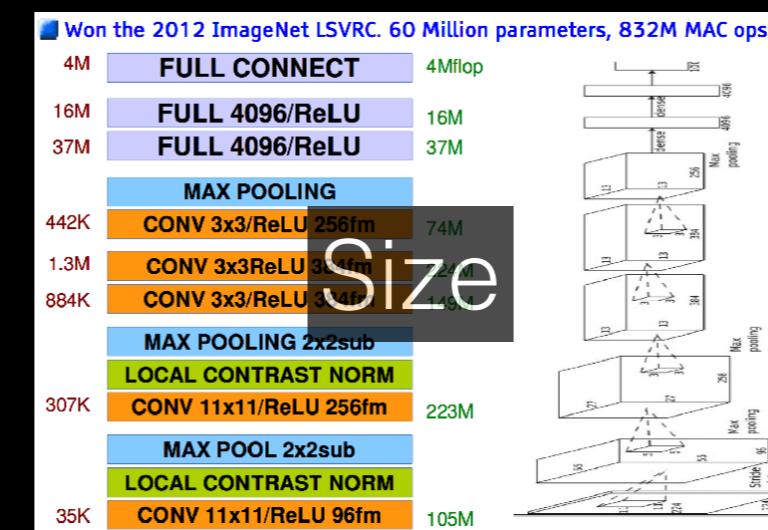
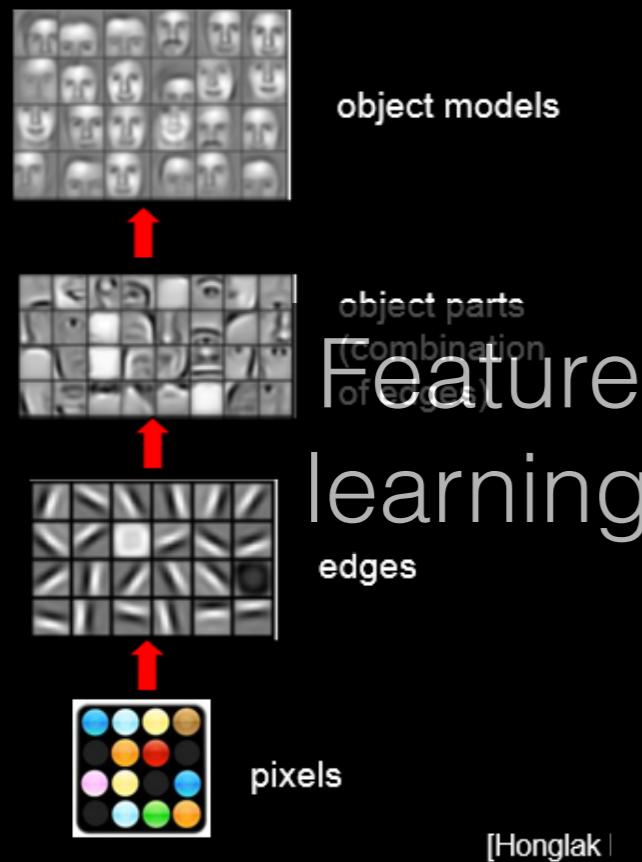
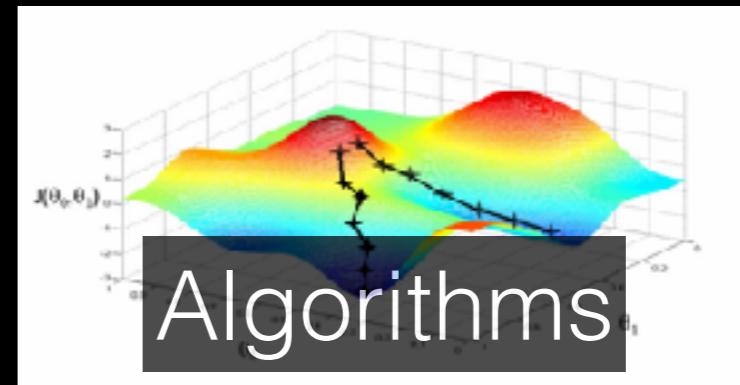
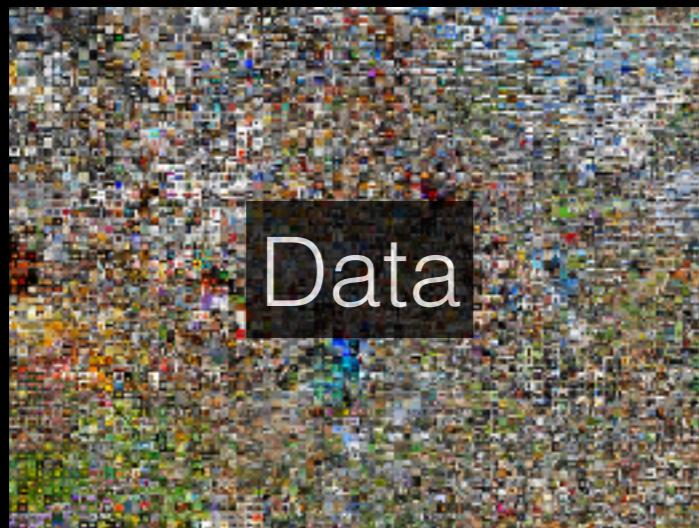


Microsoft
Research

Revolution of Depth

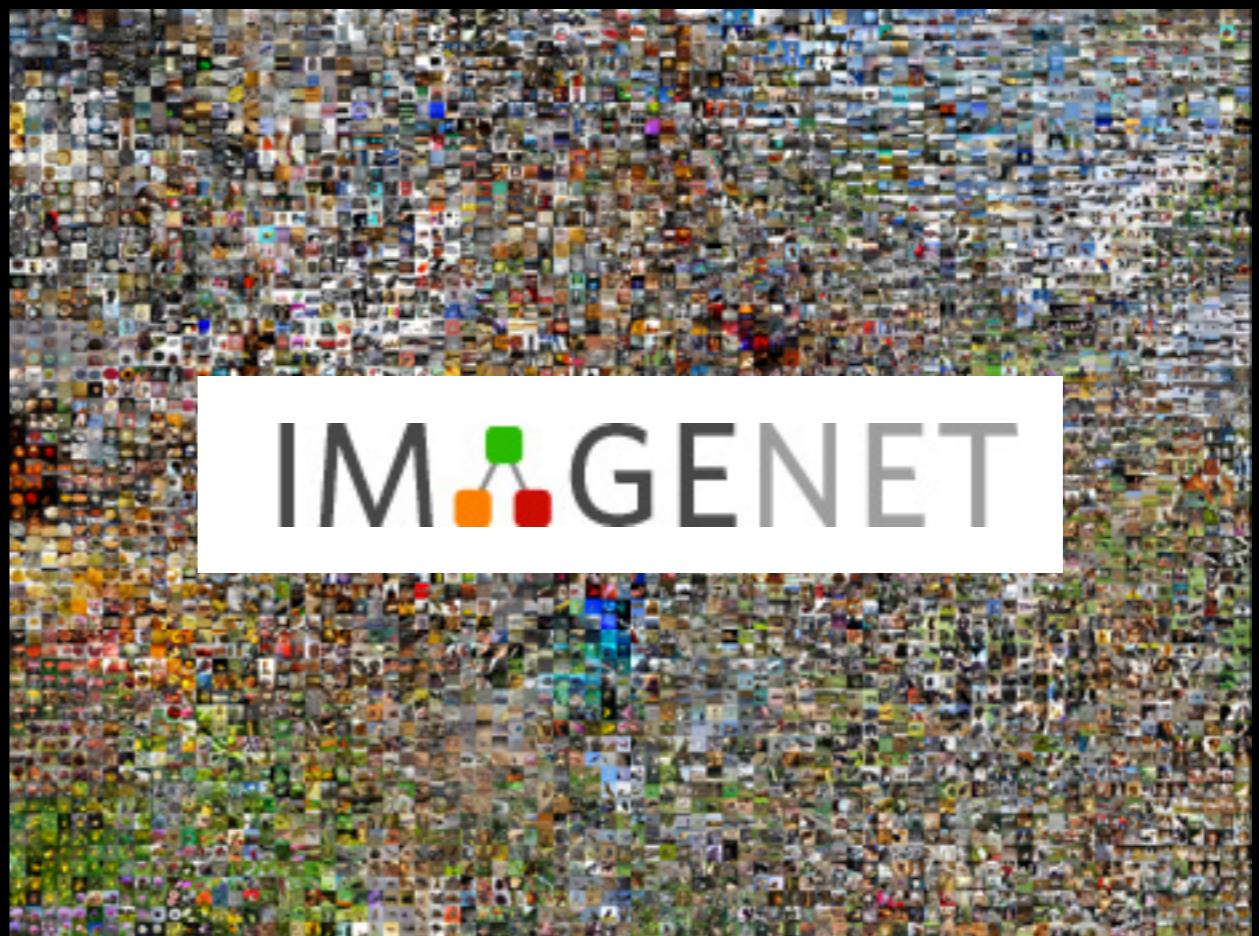


Deep learning recipe

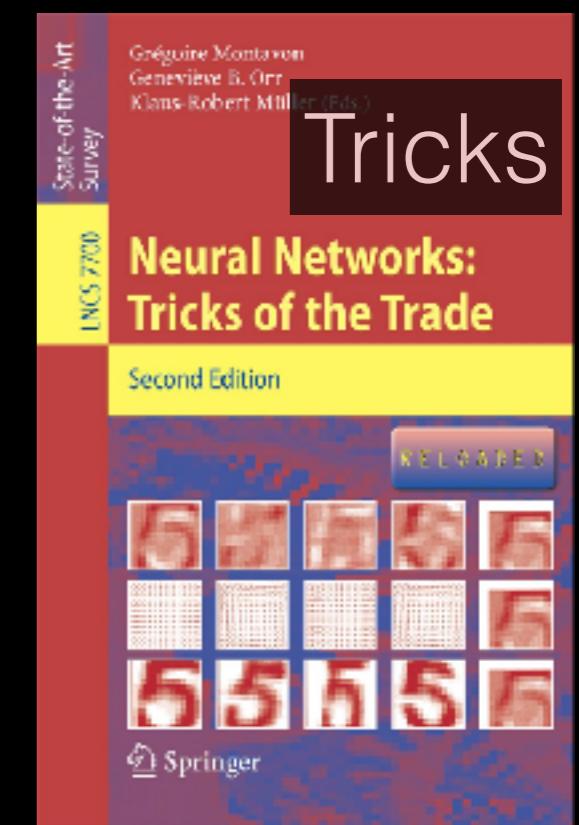
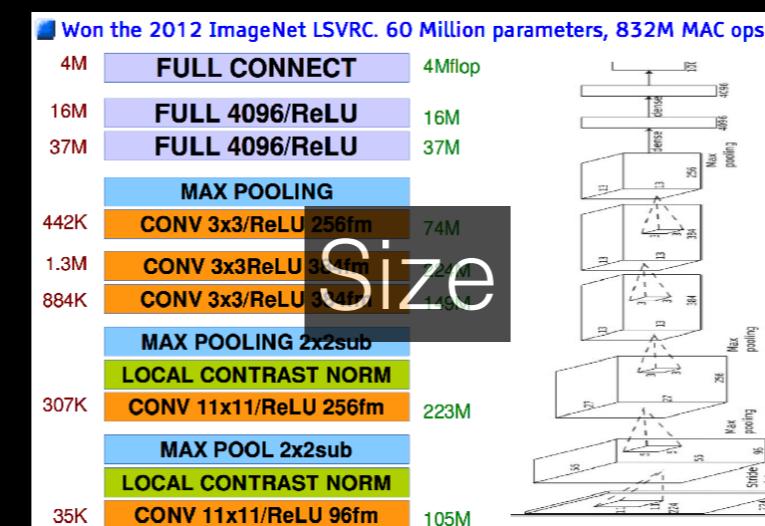
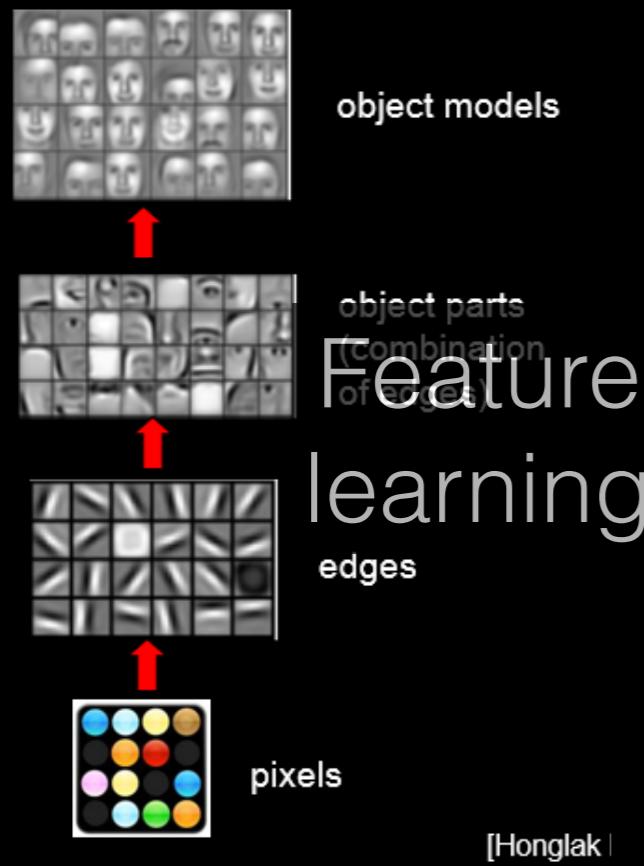
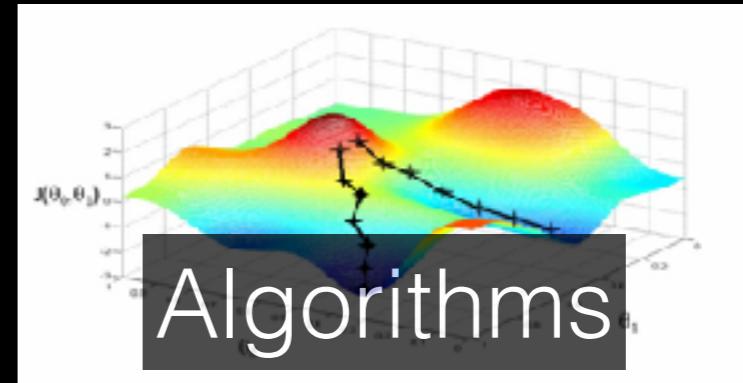
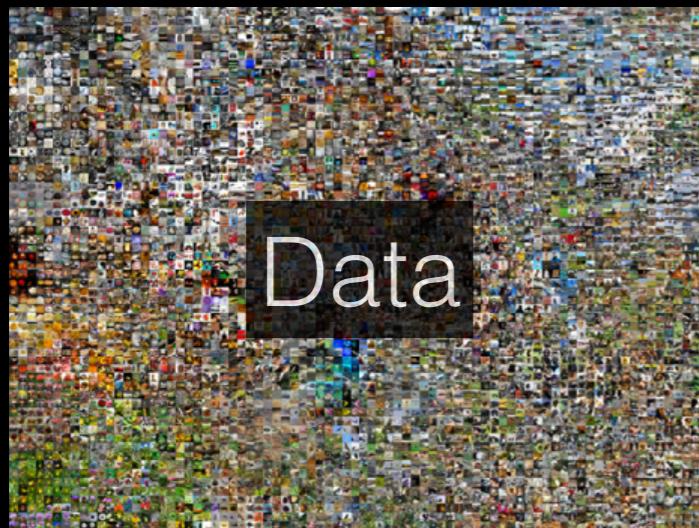


Data...

- Images annotated with WordNet concepts
- Concepts: 21,841
- Images: 14,197,122
- Bounding box annotations: 1,034,908
- Crowdsourcing



Deep learning recipe



HPC



Tesla K40

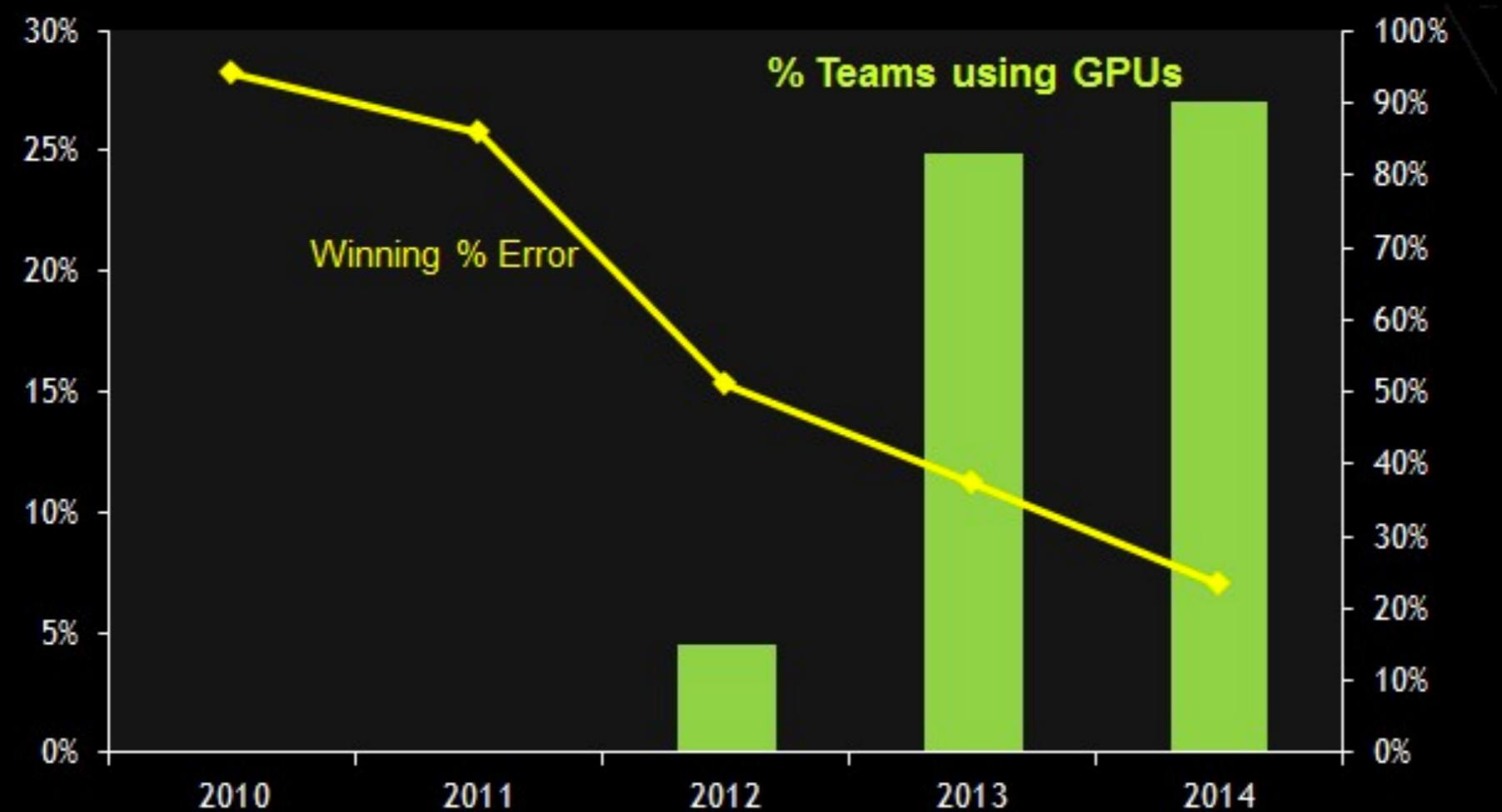
Larger datasets
Faster acceleration
Dynamic Power Scaling
Same power

Expands GPU Opportunity &
Lead over Competition

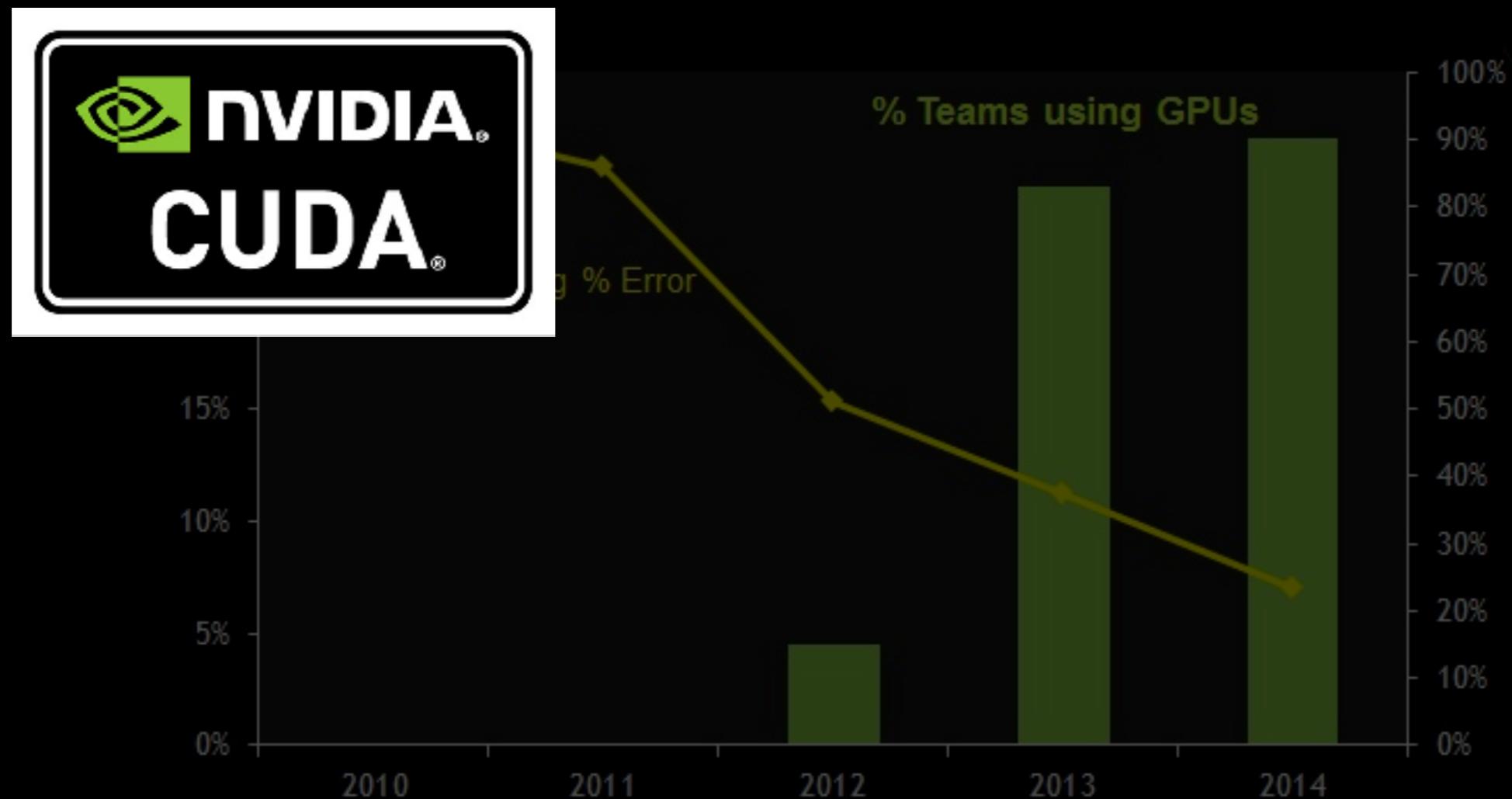
K40 specs are preliminary

	K20X	K40 (Atlas)
GPU(s)	1x GK110	1x GK180
Peak Single Precision	3.95 TF	>4.0 TF
Peak SGEMM	2.90 TF	
Peak Double Precision	1.32 TF	>1.4 TF
Peak DGEMM	1.22 TF	
Memory size	6 GB	12 GB
Memory BW (ECC off)	250 GB/s	288 GB/s
Workload for Boost Clocks	-	AMBER, ANSYS
PCIe Gen	Gen 2	Gen 3
# CUDA Cores	2688	2880
Total Board Power	235W	235W (245W SXM)
Form factor	PCIe Passive, SXM	PCIe Passive, Active & TTP, SXM

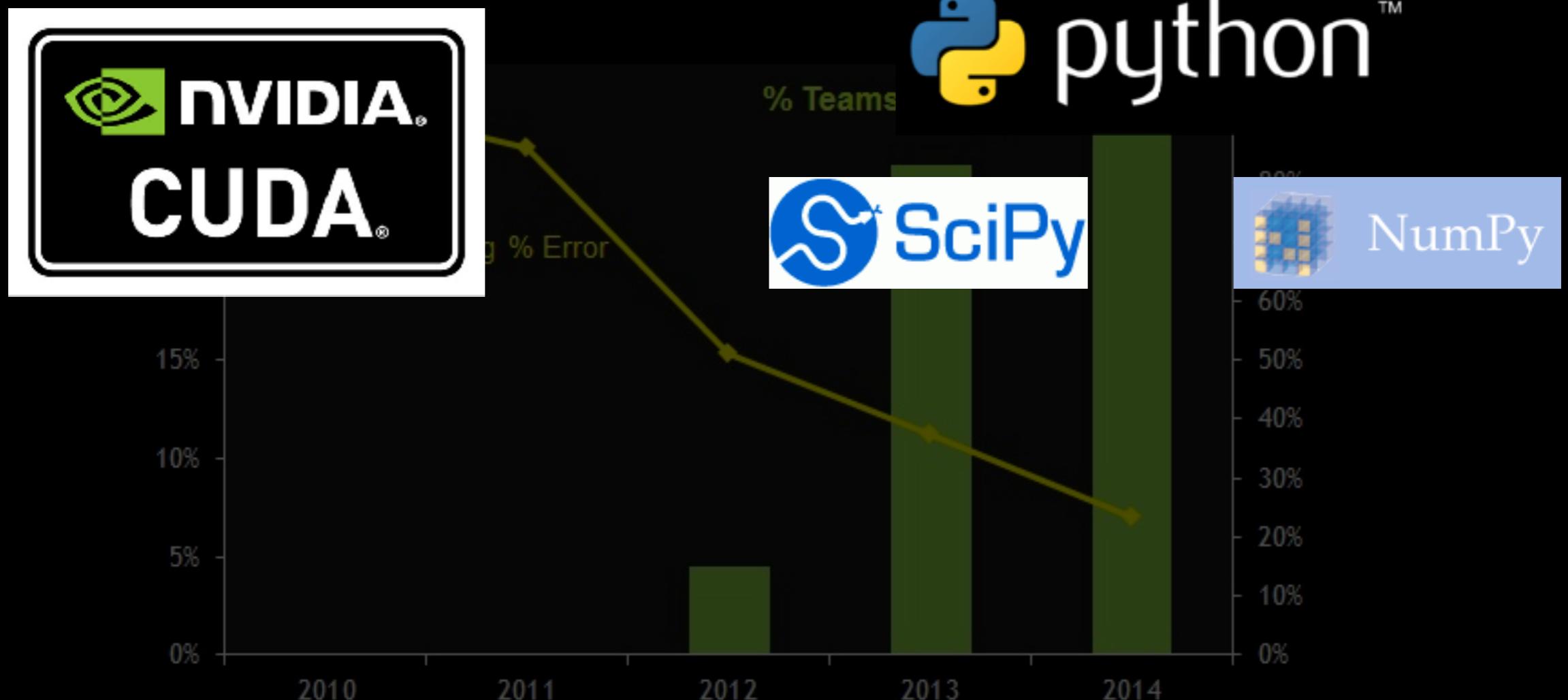
HPC



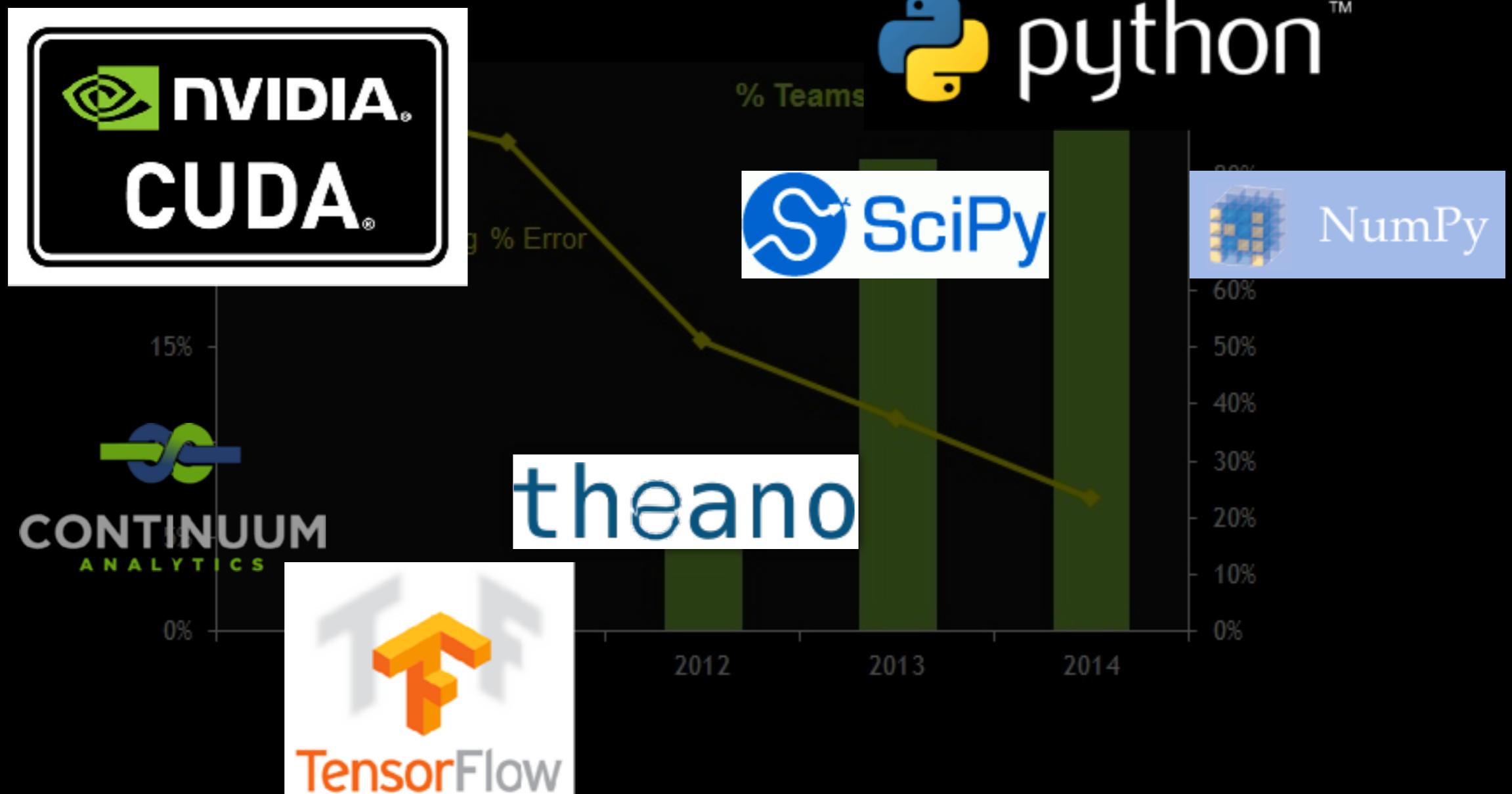
HPC



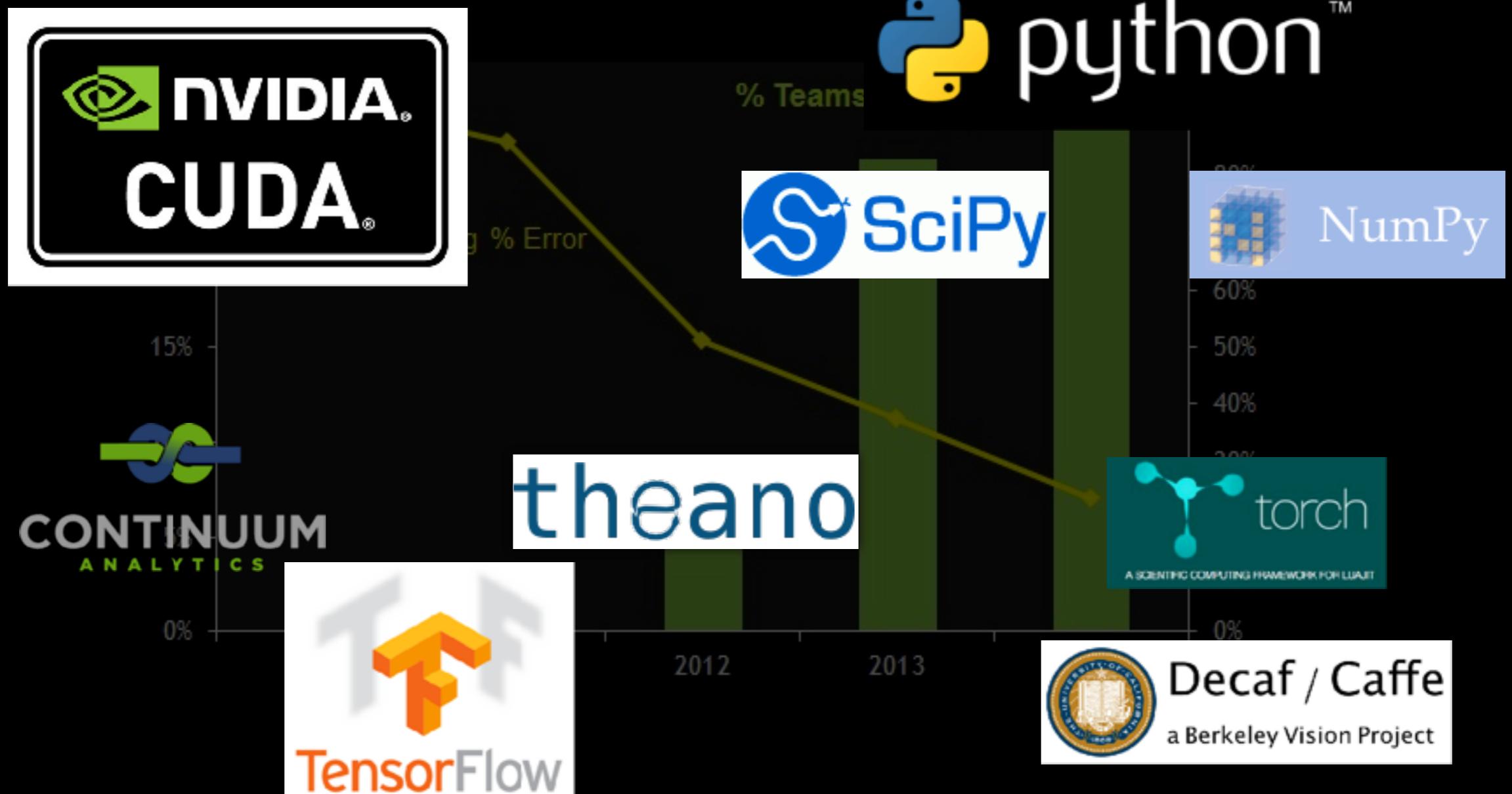
HPC



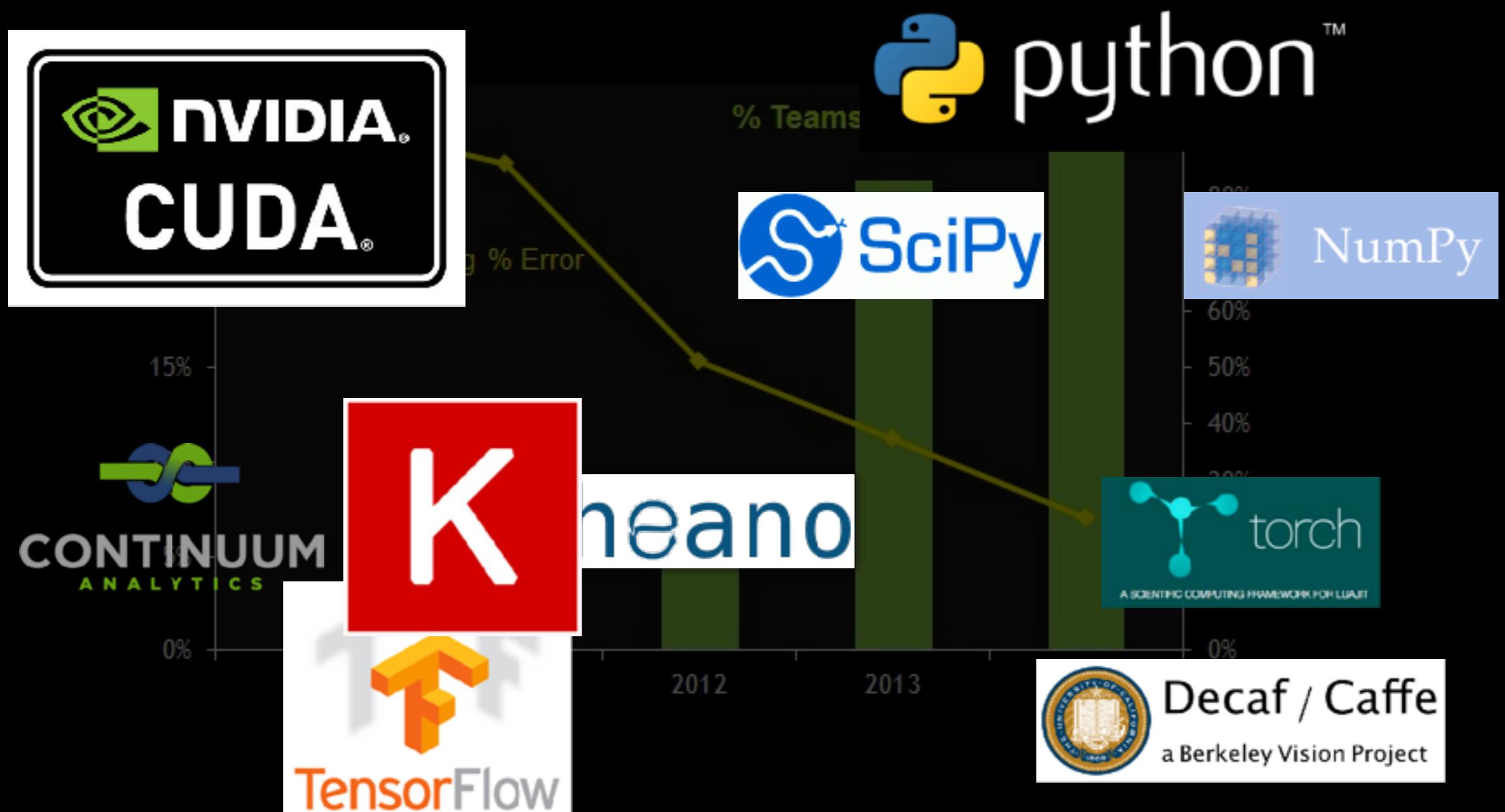
HPC



HPC



HPC



HPC



HPC



```
int main( void ) {
    CPUBitmap bitmap( DIM, DIM );
    unsigned char      *dev_bitmap;

    HANDLE_ERROR( cudaMalloc( (void**)&dev_bitmap,
                           bitmap.image_size() ) );

    dim3      grid(DIM,DIM);
    kernel<<<grid,1>>>( dev_bitmap );

    HANDLE_ERROR( cudaMemcpy( bitmap.get_ptr() ,
                           dev_bitmap,
                           bitmap.image_size(),
                           cudaMemcpyDeviceToHost ) );
    bitmap.display_and_exit();

    cudaFree( dev_bitmap );
}
```



HPC



```
# Declare Theano symbolic variables
x = T.matrix("x")
y = T.vector("y")
w = theano.shared(rng.randn(feats).astype(theano.config.floatX), name="w")
b = theano.shared(numpy.asarray(0., dtype=theano.config.floatX), name="b")
x.tag.test_value = D[0]
y.tag.test_value = D[1]

# Construct Theano expression graph
p_1 = 1 / (1 + T.exp(-T.dot(x, w)-b)) # Probability of having a one
prediction = p_1 > 0.5 # The prediction that is done: 0 or 1
xent = -y*T.log(p_1) - (1-y)*T.log(1-p_1) # Cross-entropy
cost = xent.mean() + 0.01*(w**2).sum() # The cost to optimize
gw,gb = T.grad(cost, [w,b])

# Compile expressions to functions
train = theano.function(
    inputs=[x,y],
    outputs=[prediction, xent],
    updates=[(w, w-0.01*gw), (b, b-0.01*gb)],
    name = "train")
predict = theano.function(inputs=[x], outputs=prediction,
    name = "predict")
```

theano



HPC



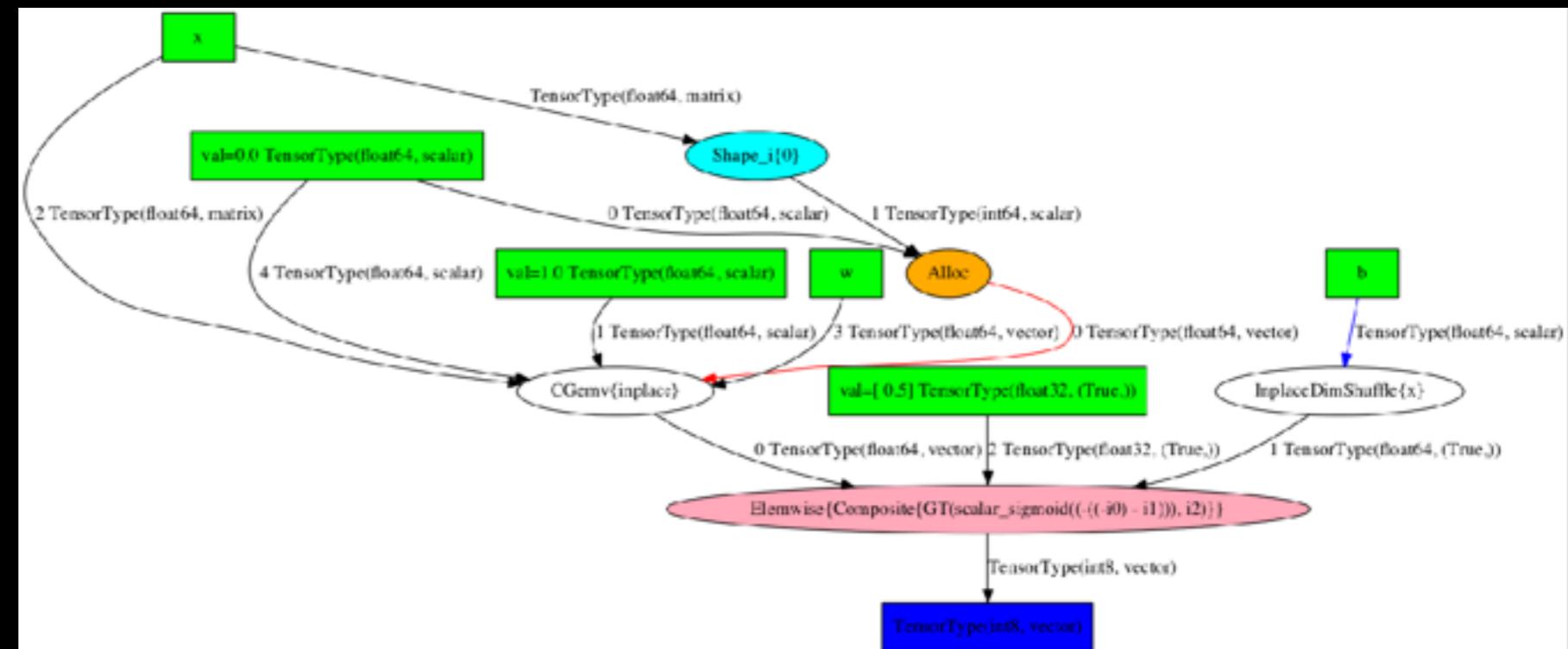
theano



HPC



theano



HPC



theano



```
# Parameters
learning_rate = 0.01
training_epochs = 25
batch_size = 100
display_step = 1

# tf Graph Input
x = tf.placeholder(tf.float32, [None, 784]) # mnist data image of shape 28*28=784
y = tf.placeholder(tf.float32, [None, 10]) # 0-9 digits recognition => 10 classes

# Set model weights
W = tf.Variable(tf.zeros([784, 10]))
b = tf.Variable(tf.zeros([10]))

# Construct model
pred = tf.nn.softmax(tf.matmul(x, W) + b) # Softmax

# Minimize error using cross entropy
cost = tf.reduce_mean(-tf.reduce_sum(y*tf.log(pred), reduction_indices=1))
# Gradient Descent
optimizer = tf.train.GradientDescentOptimizer(learning_rate).minimize(cost)

# Initializing the variables
init = tf.initialize_all_variables()
```

HPC



theano

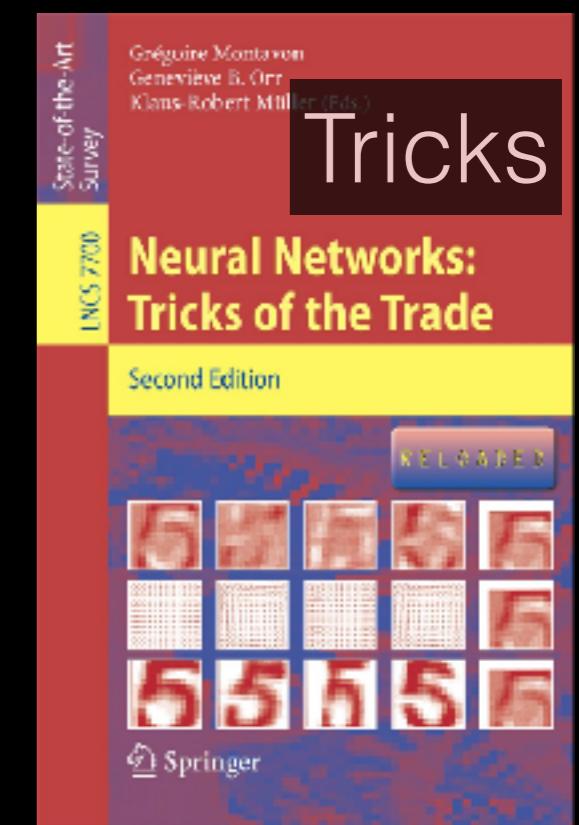
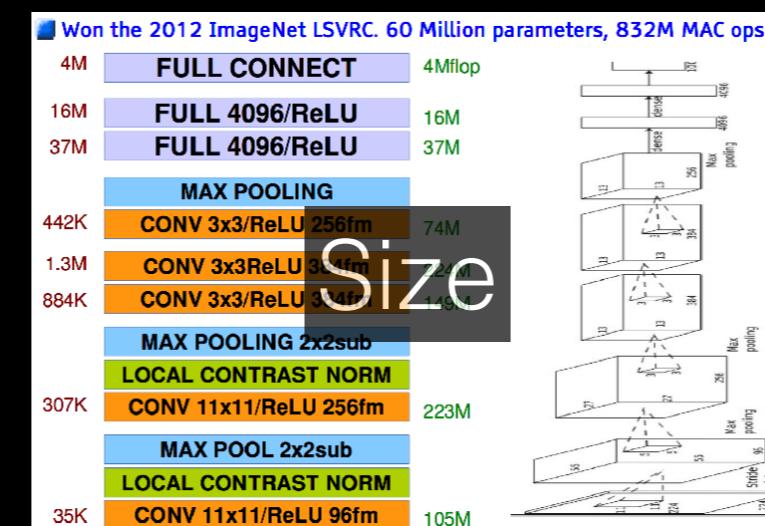
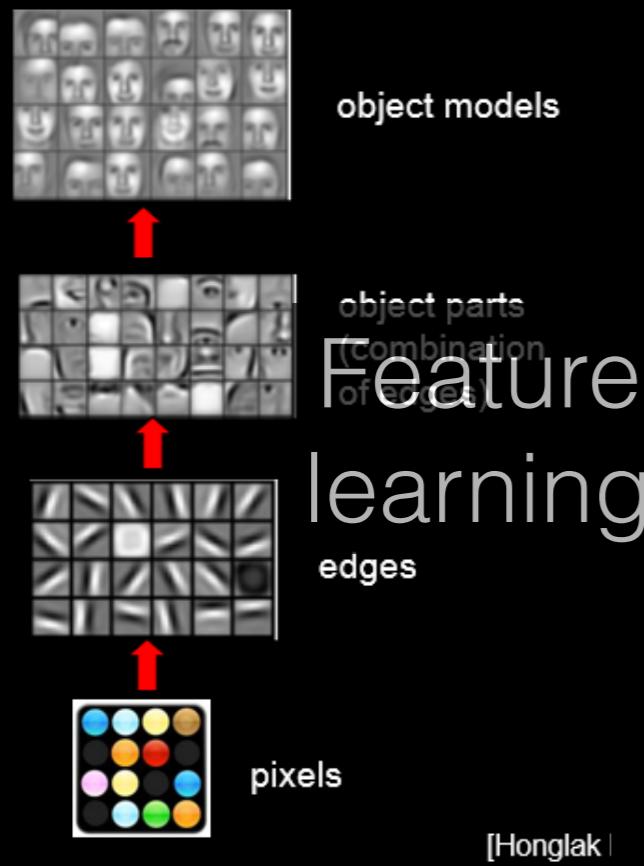
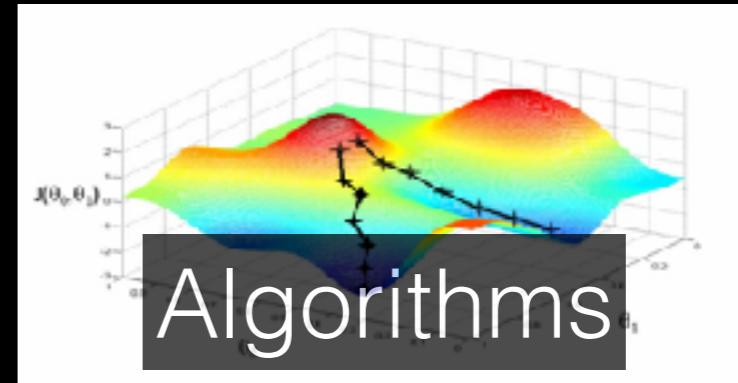
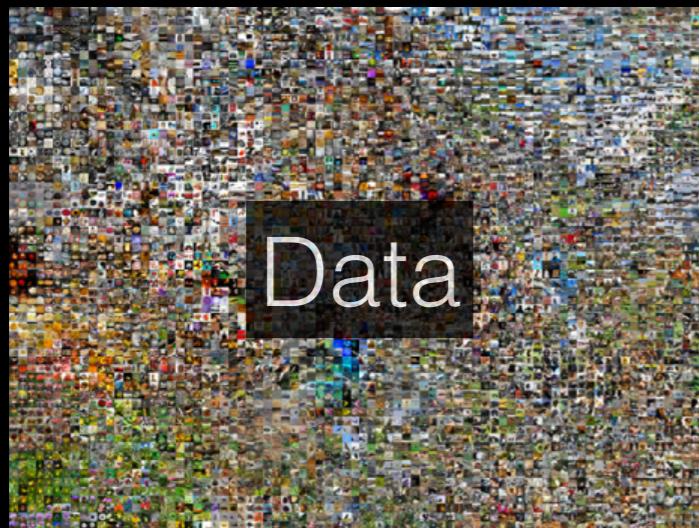
```
train_y_ohe = one_hot_encode_object_array(train_y)
test_y_ohe = one_hot_encode_object_array(test_y)

model = Sequential()
model.add(Dense(16, input_shape=(4,)))
model.add(Activation('sigmoid'))
model.add(Dense(3))
model.add(Activation('softmax'))
model.compile(loss='categorical_crossentropy', metrics=['accuracy'], optimizer='adam')

# Actual modelling
model.fit(train_X, train_y_ohe, verbose=0, batch_size=1)
score, accuracy = model.evaluate(test_X, test_y_ohe, batch_size=16, verbose=0)
print("Test score: %.2f" % score)
print("Test accuracy: %.2f" % accuracy)
```

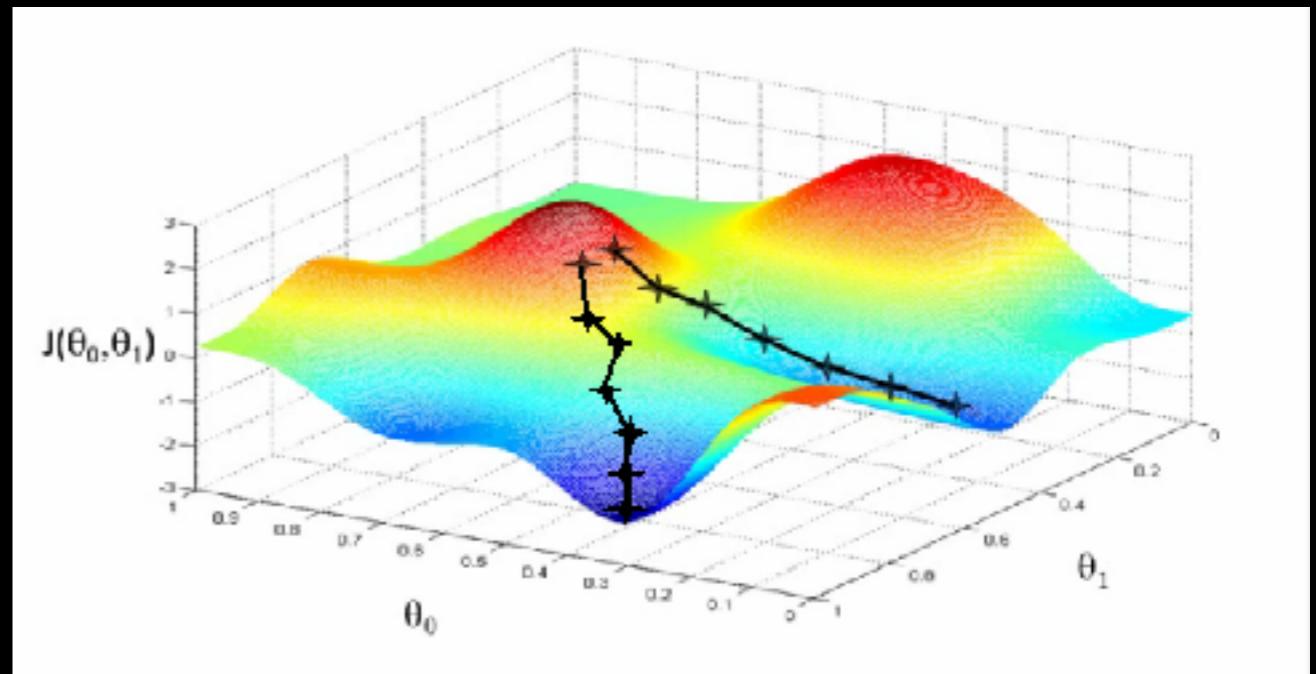


Deep learning recipe

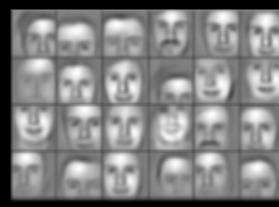
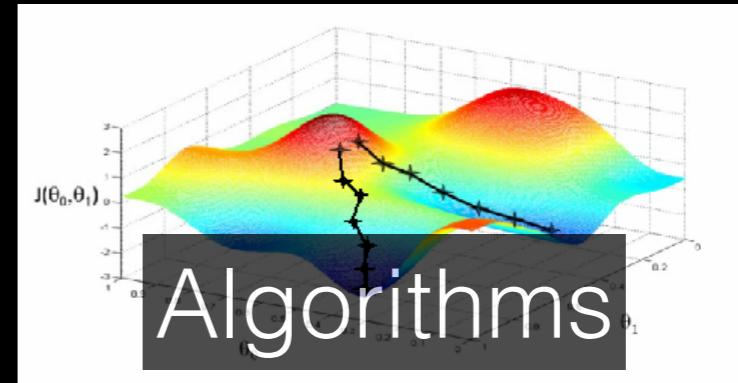
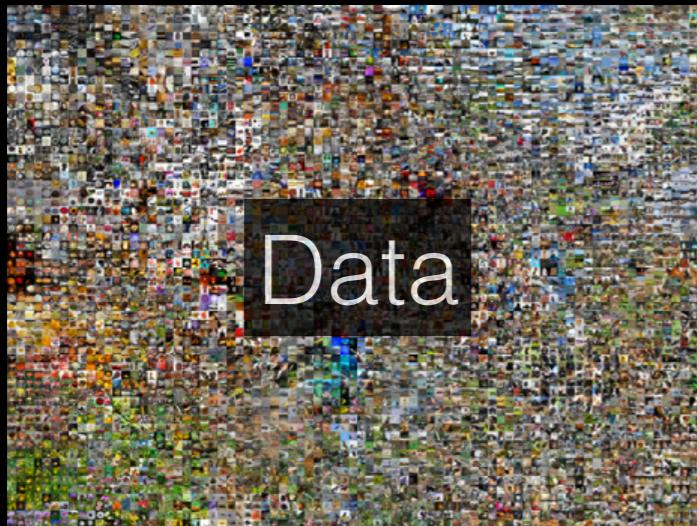


Algorithms

- Backpropagation
- Backpropagation through time
- Online learning (stochastic gradient descent)
- Softmax (hierarchical)



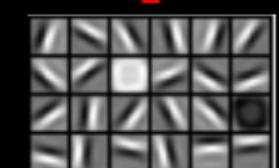
Deep learning recipe



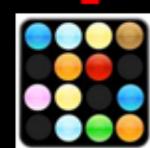
object models



object parts
(combination
of edges)



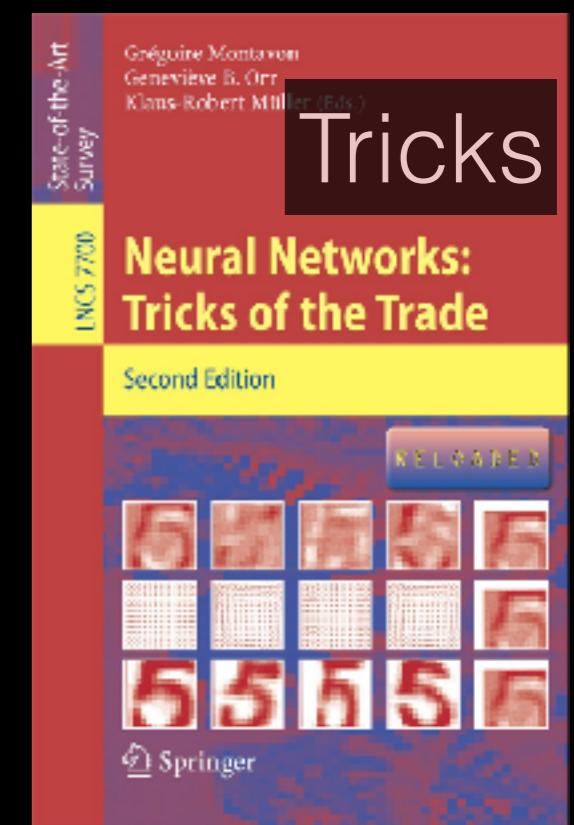
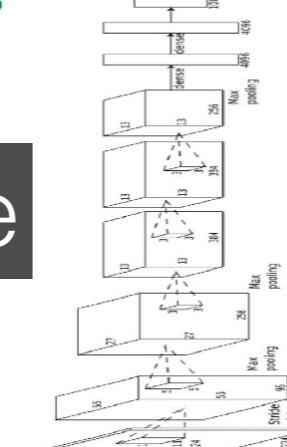
Feature
learning
edges



pixels

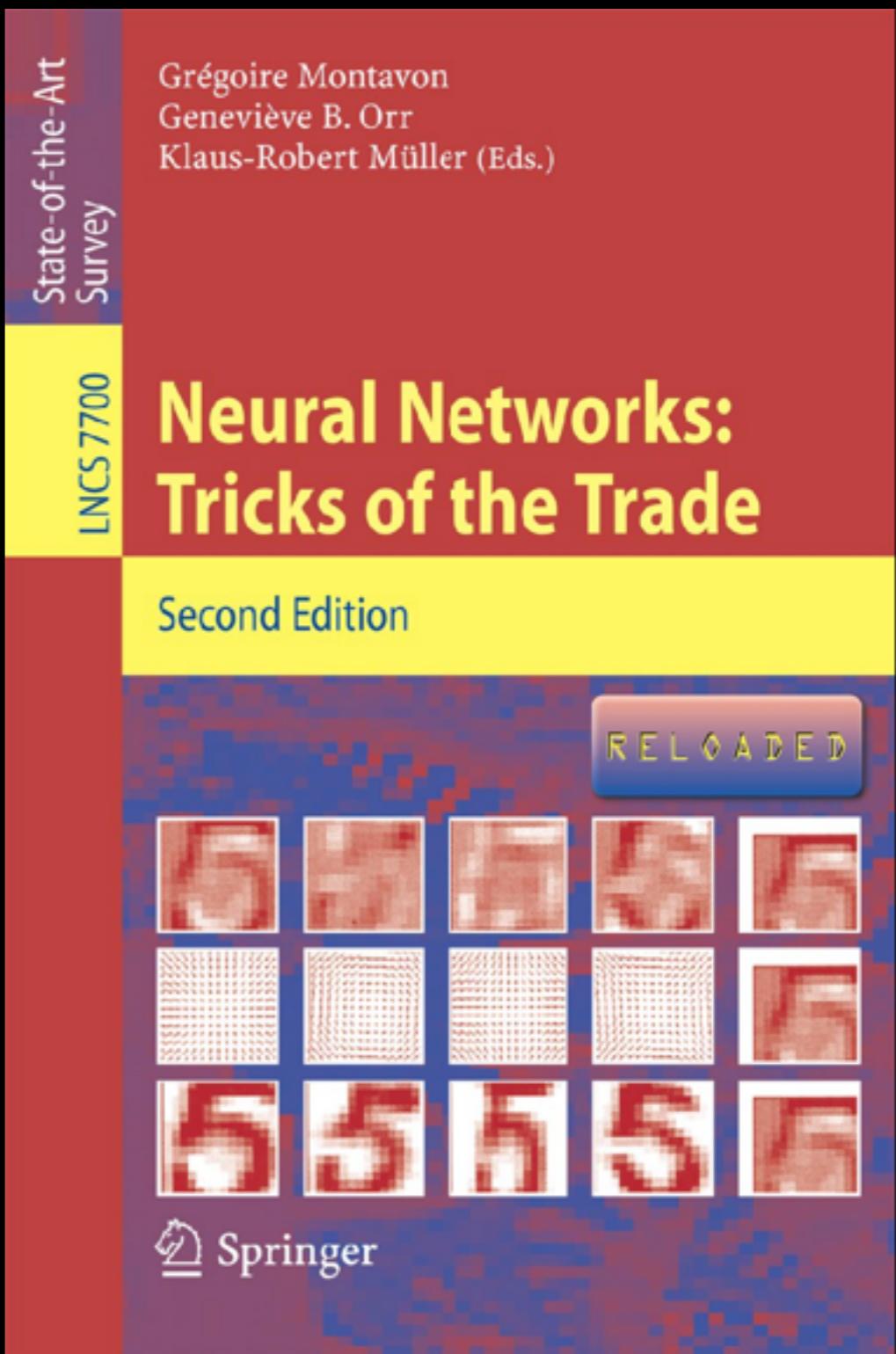
[Honglak]

Won the 2012 ImageNet LSVRC. 60 Million parameters, 832M MAC ops		
4M	FULL CONNECT	4Mflop
16M	FULL 4096/ReLU	16M
37M	FULL 4096/ReLU	37M
	MAX POOLING	
442K	CONV 3x3/ReLU 256fm	74M
1.3M	CONV 3x3/ReLU 384fm	24M
884K	CONV 3x3/ReLU 384fm	149M
	MAX POOLING 2x2sub	
	LOCAL CONTRAST NORM	
307K	CONV 11x11/ReLU 256fm	223M
	MAX POOL 2x2sub	
	LOCAL CONTRAST NORM	
35K	CONV 11x11/ReLU 96fm	105M

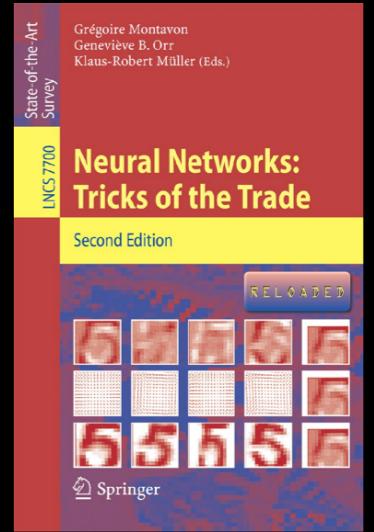


Tricks

- DL is mainly an engineering problem
- DL networks are hard to train
- Several tricks product of years of experience



Tricks



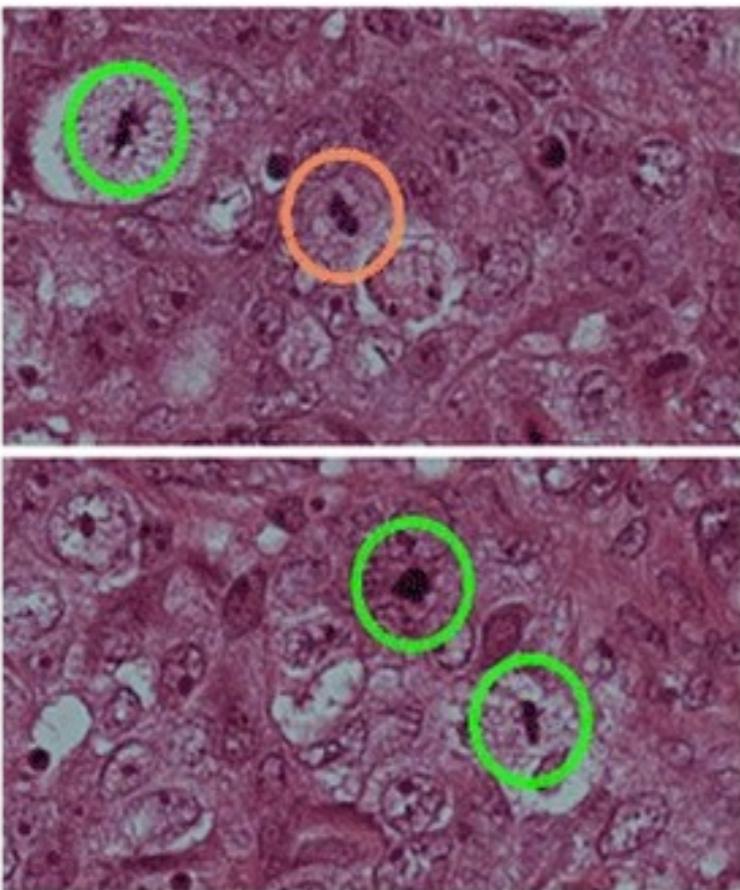
- DL is mainly an engineering problem
- DL networks are hard to train
- Several tricks product of years of experience
 - Layer-wise training
 - RELU units
 - Dropout
 - Adaptive learning rates
 - Initialization
 - Preprocessing
 - Gradient norm clipping

Applications

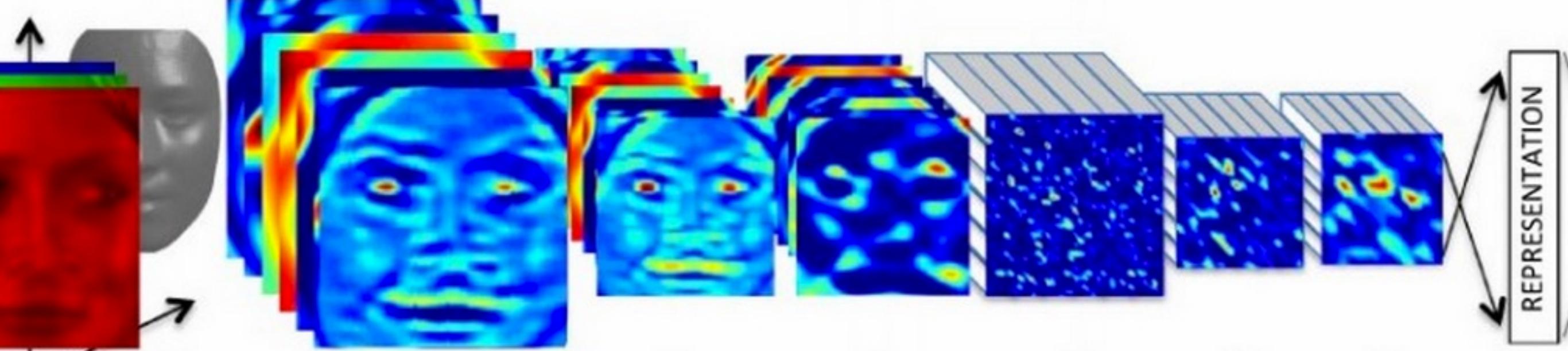
- Computer vision:
 - Image: annotation, detection, segmentation, captioning
 - Video: object tracking, action recognition, segmentation
- Speech recognition and synthesis
- Text: language modeling, word/text representation, text classification, translation
- Biomedical image analysis

Autoencoder Demo

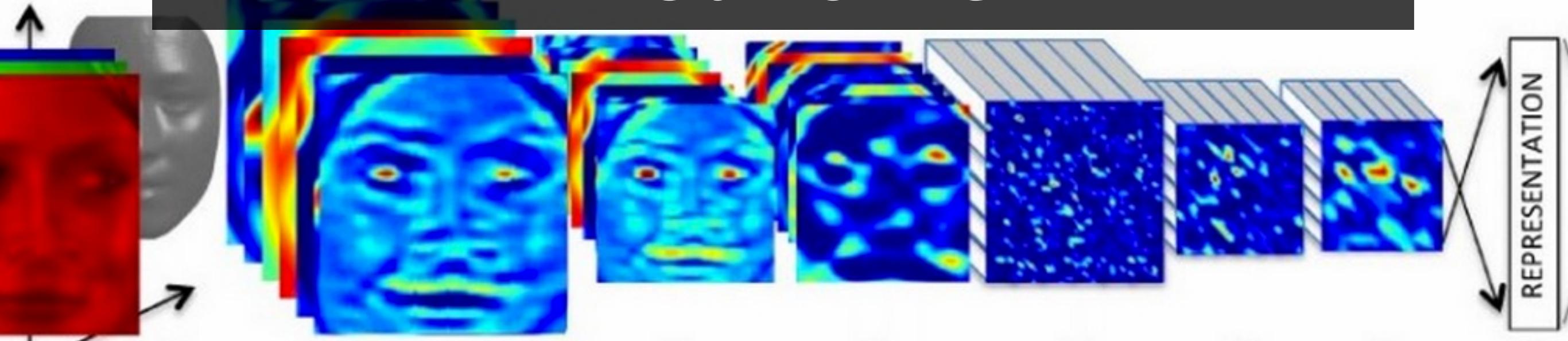
[https://cs.stanford.edu/people/karpathy/convnetjs/demo/
autoencoder.html](https://cs.stanford.edu/people/karpathy/convnetjs/demo/autoencoder.html)



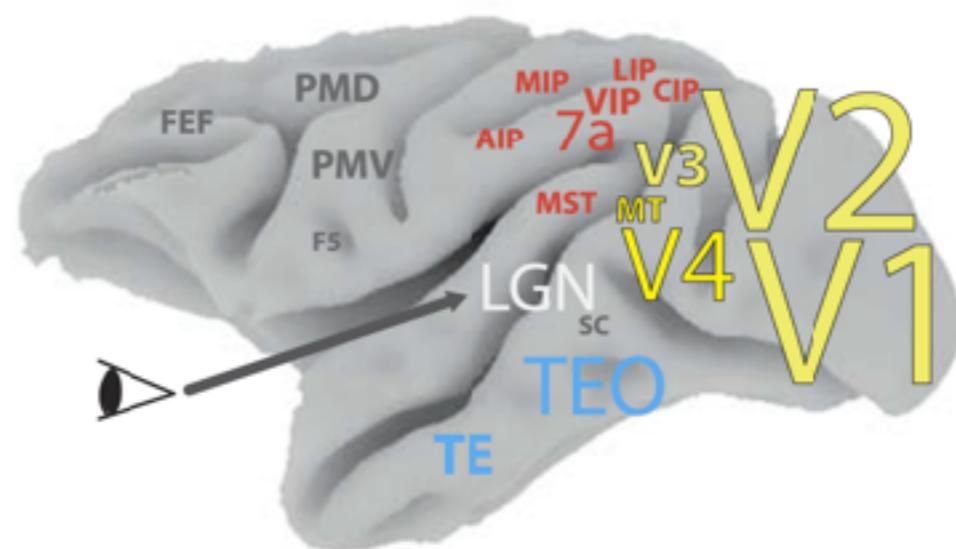
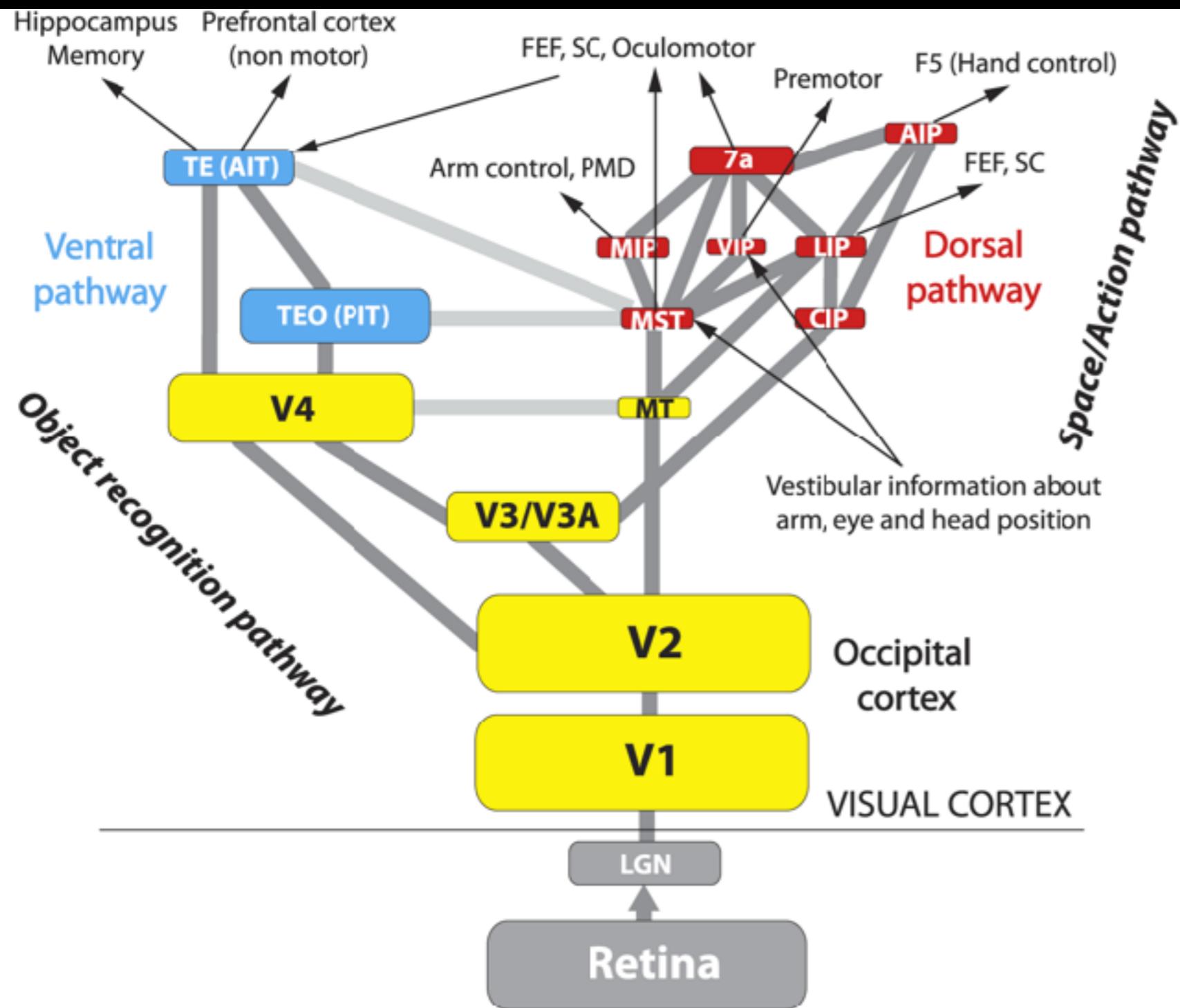
咱攢暫獎賊隨葬
擇州澤城怎增櫈曾
旅摘竚宅宋債塞瞻
湛綻樟章勳譚張哲
囉罩冰摩乃遮折哲
針傾枕疚詣震振鎮
鄭征艺枝支咷蜘蛛知
止趾只旨紙志擎擲



Convolutional Neural Networks

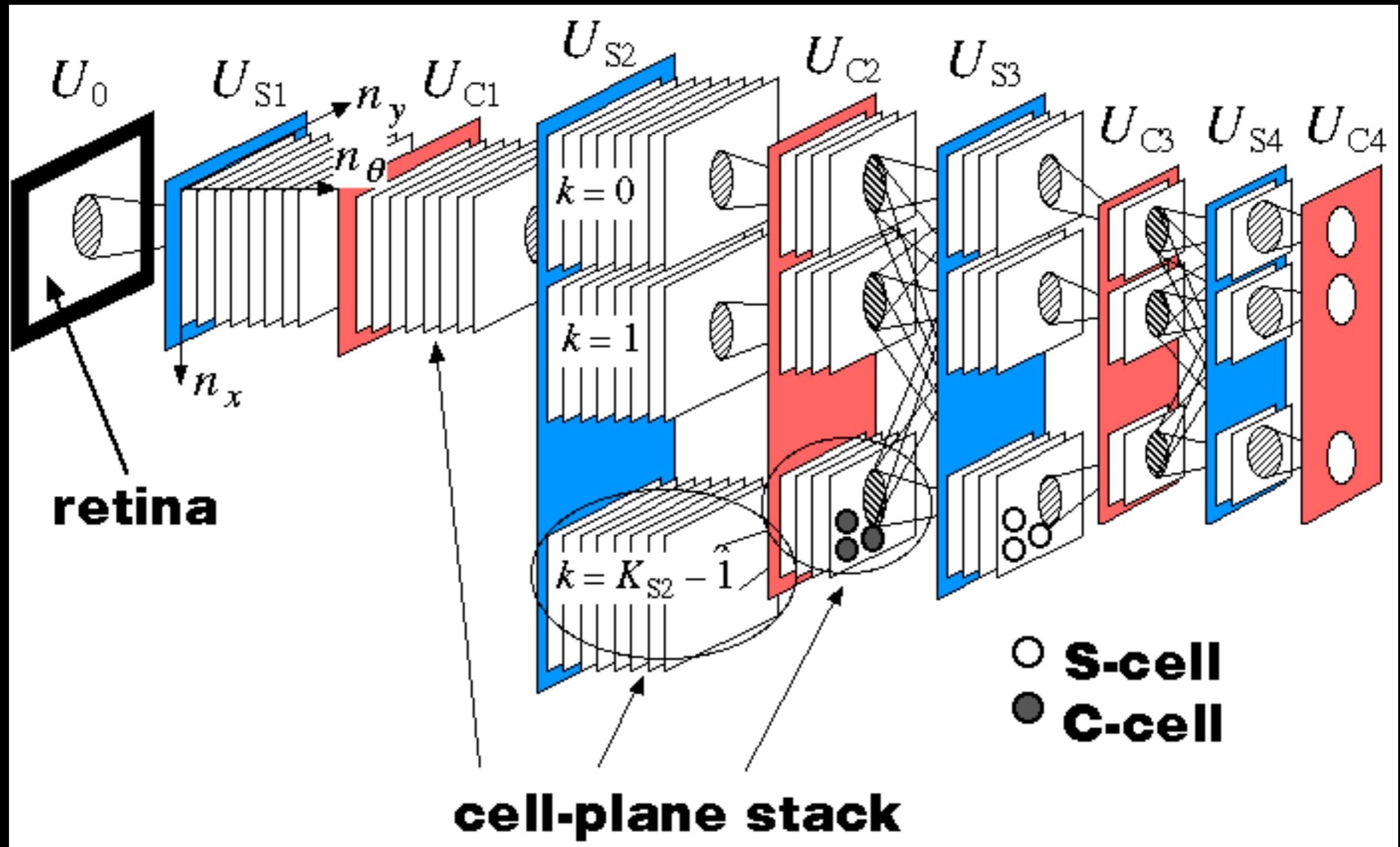


Visual Cortex



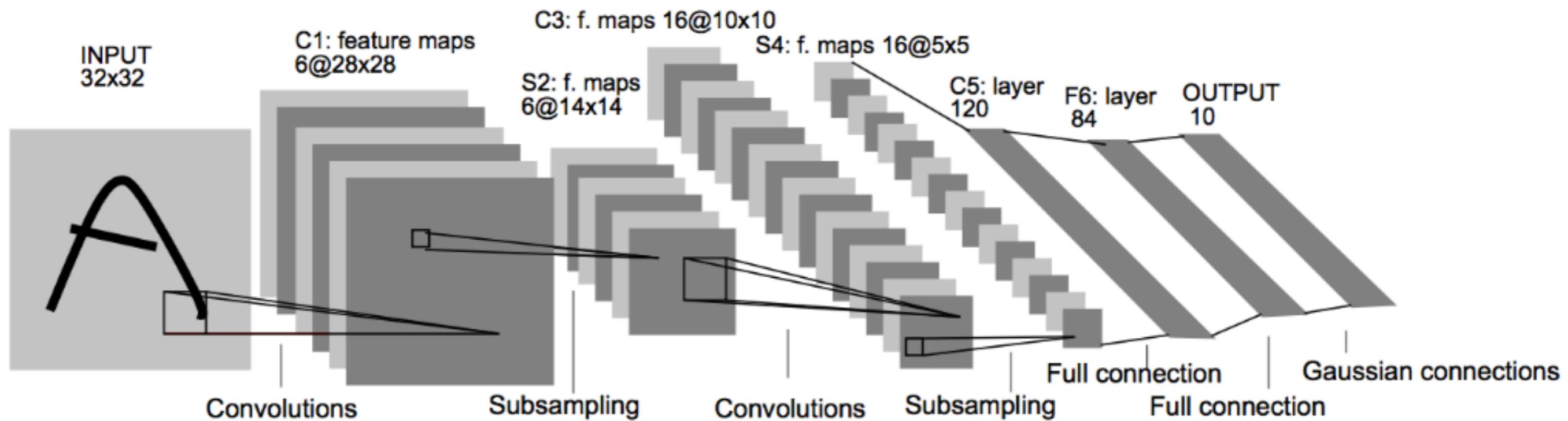
Neocognitron

(Fukushima, 1980)

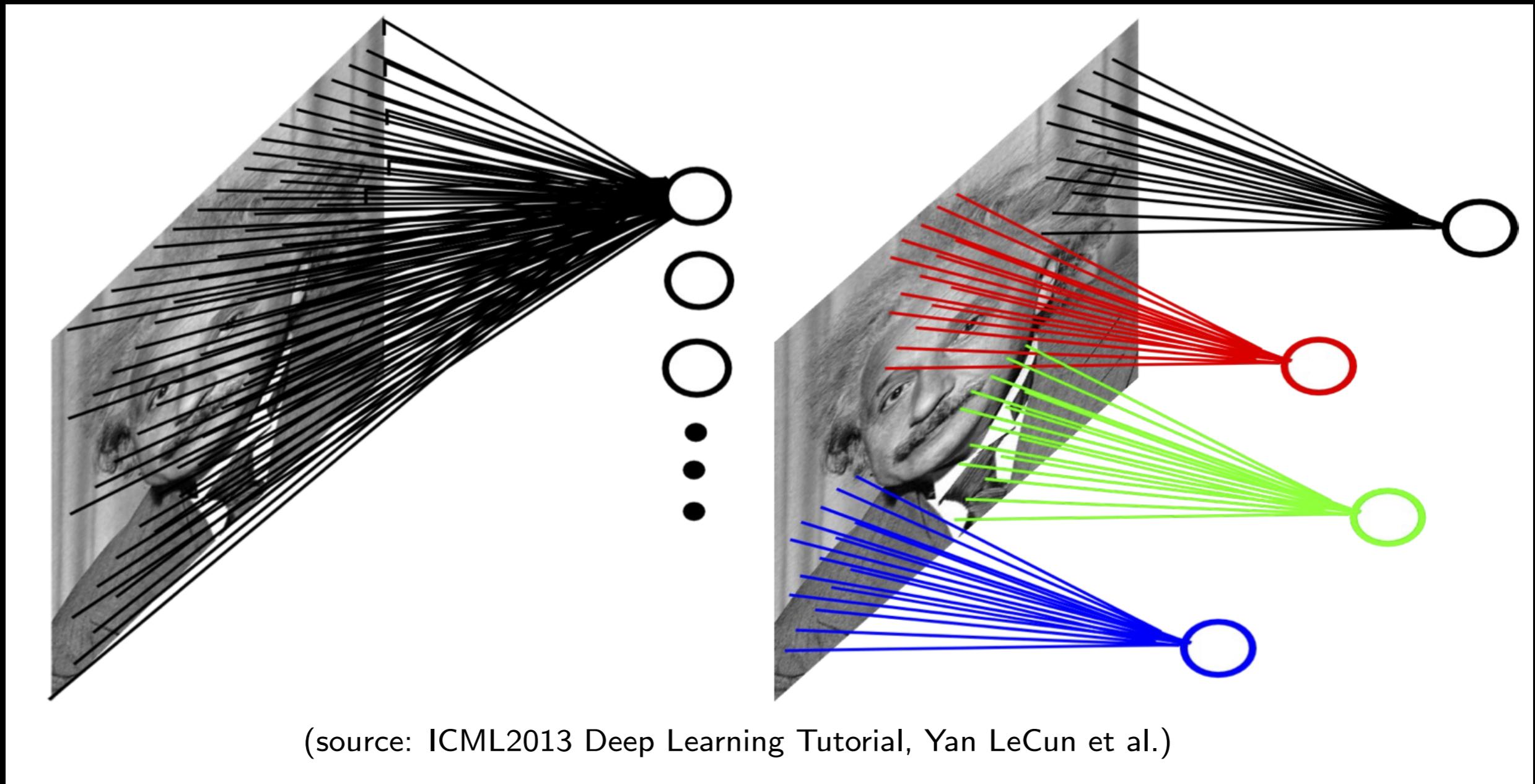


LeNet

(LeCun, 1998)

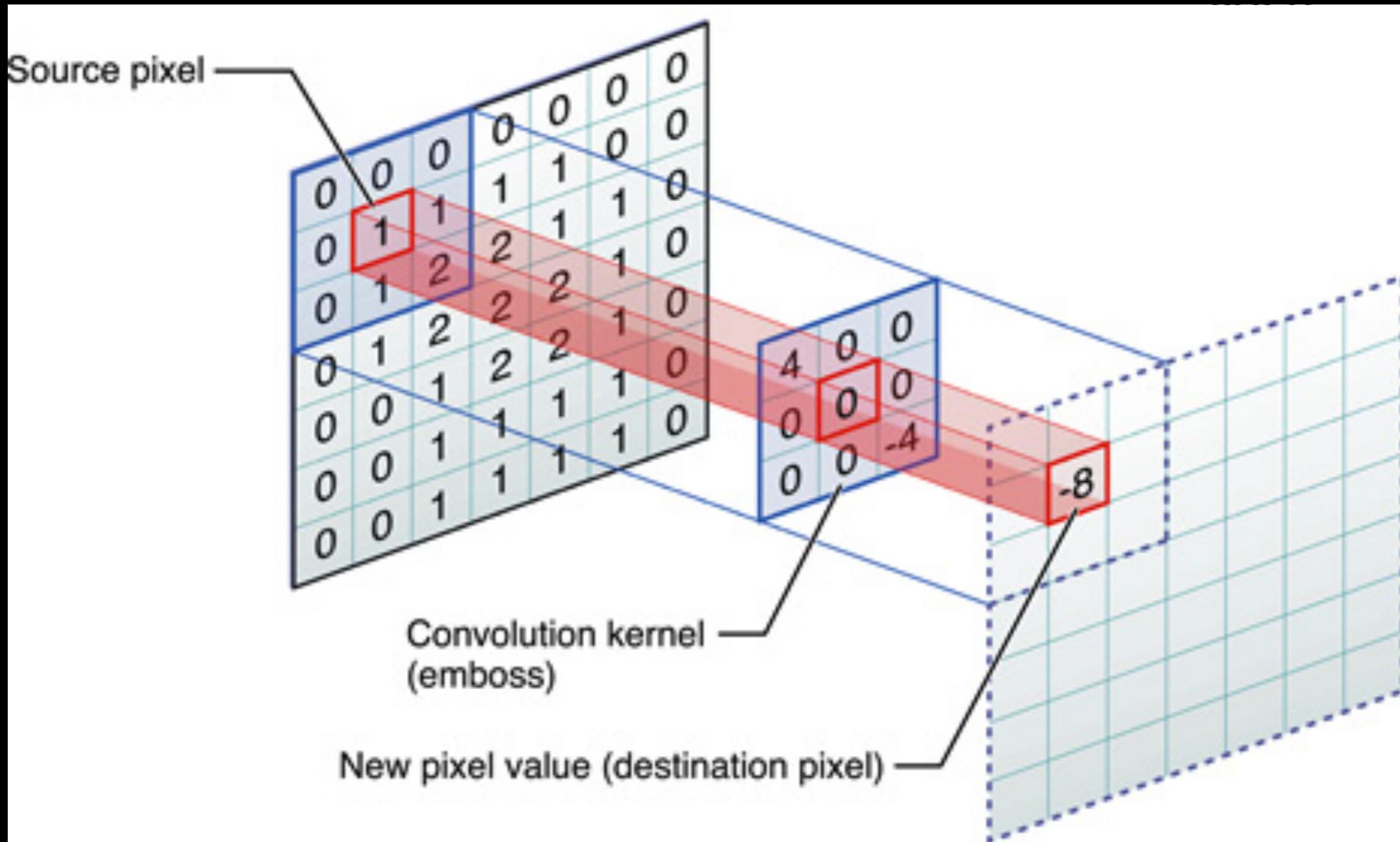


Convolution



(sources: ICML2013 Deep Learning Tutorial, Yan LeCun et al.
[Feature extraction using convolution](#), Stanford Deep Learning Wiki)

Convolution



(sources: ICML2013 Deep Learning Tutorial, Yan LeCun et al.
Feature extraction using convolution, Stanford Deep Learning Wiki)

Convolution

1 <small>x1</small>	1 <small>x0</small>	1 <small>x1</small>	0	0
0 <small>x0</small>	1 <small>x1</small>	1 <small>x0</small>	1	0
0 <small>x1</small>	0 <small>x0</small>	1 <small>x1</small>	1	1
0	0	1	1	0
0	1	1	0	0

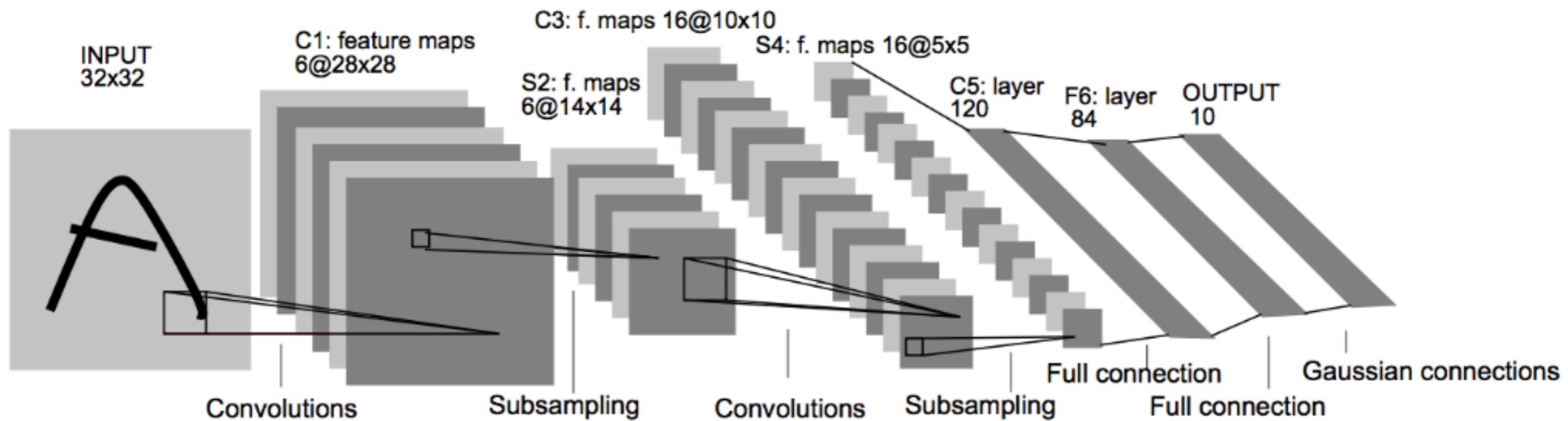
Image

4		

Convolved
Feature

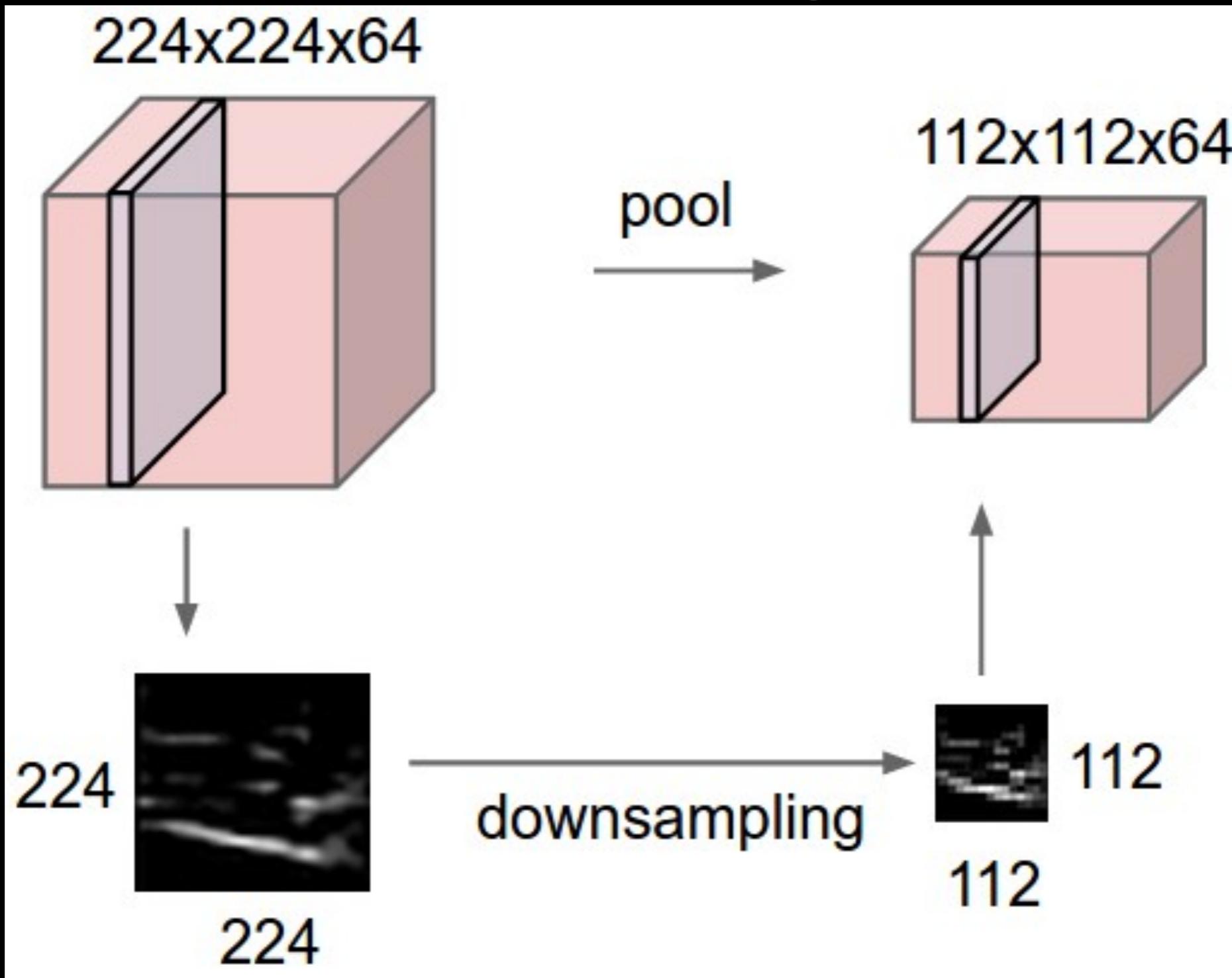
(sources: ICML2013 Deep Learning Tutorial, Yan LeCun et al.
[Feature extraction using convolution](#), Stanford Deep Learning Wiki)

Convolution



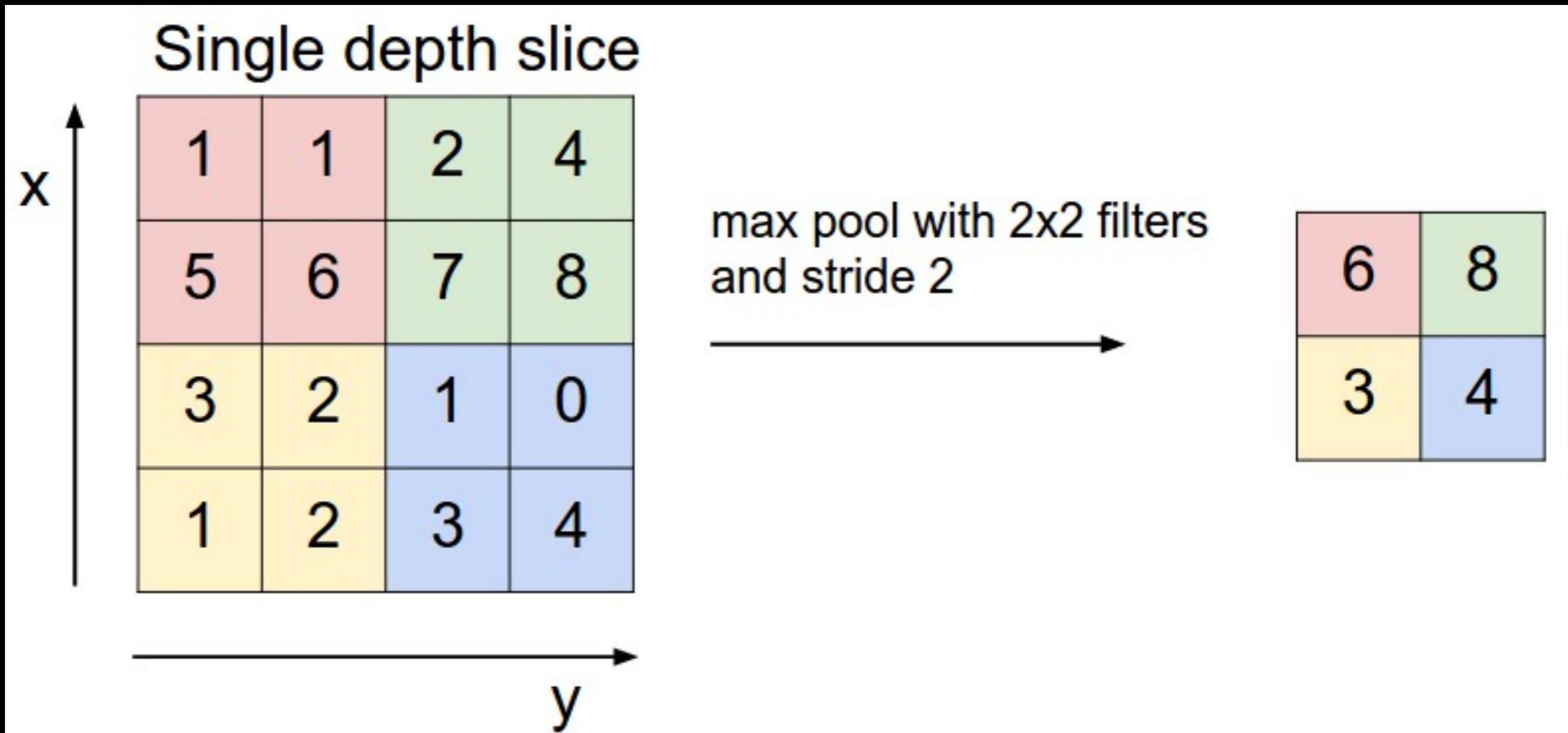
(sources: ICML2013 Deep Learning Tutorial, Yan LeCun et al.
[Feature extraction using convolution](#), Stanford Deep Learning Wiki)

Pooling



(source: Karpathy, [CS231n Convolutional Neural Networks for Visual Recognition](#))

Pooling



(source: Karpathy, [CS231n Convolutional Neural Networks for Visual Recognition](#))

CNN Demo

<https://cs.stanford.edu/people/karpathy/convnetjs/demo/mnist.html>

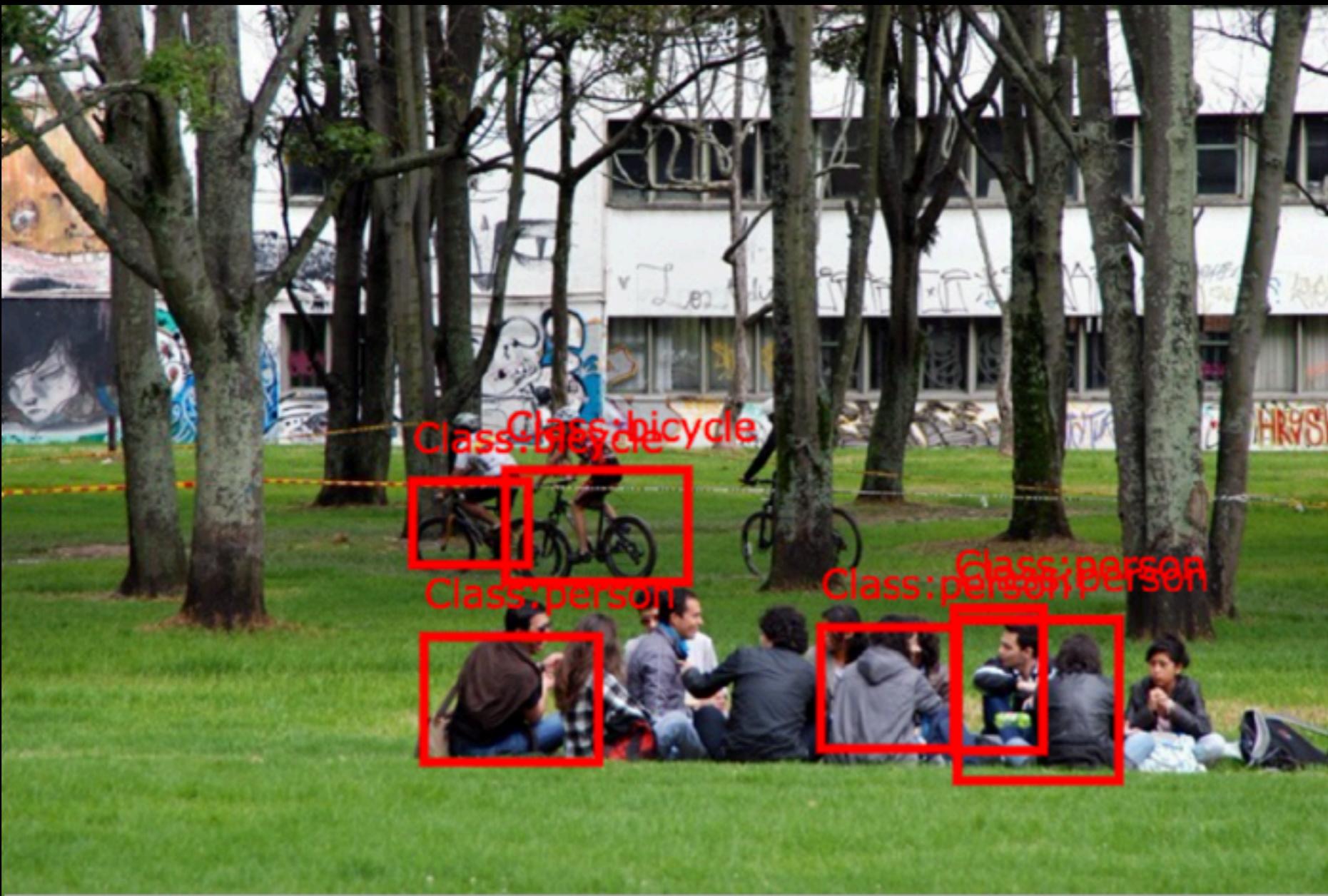
<https://cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html>

Object Detection



<http://silverpond.com.au/object-detector>

Object Detection



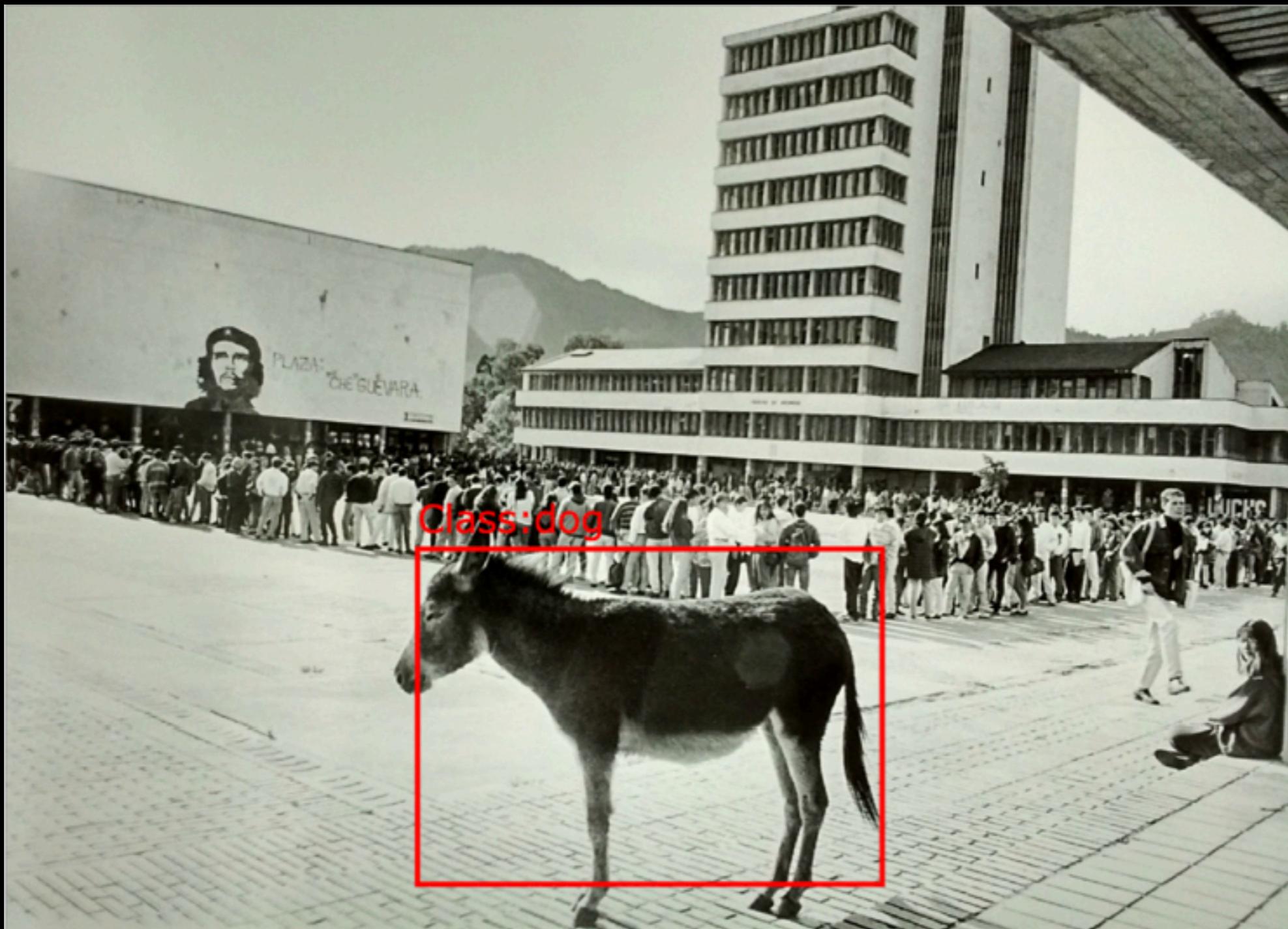
<http://silverpond.com.au/object-detector>

Object Detection



<http://silverpond.com.au/object-detector>

Object Detection



<http://silverpond.com.au/object-detector>

Image Captioning

I think it's a group of people sitting at a park.



Image Captioning

I think it's a horse standing in front of a building and he seems
😔.



Style Transfer



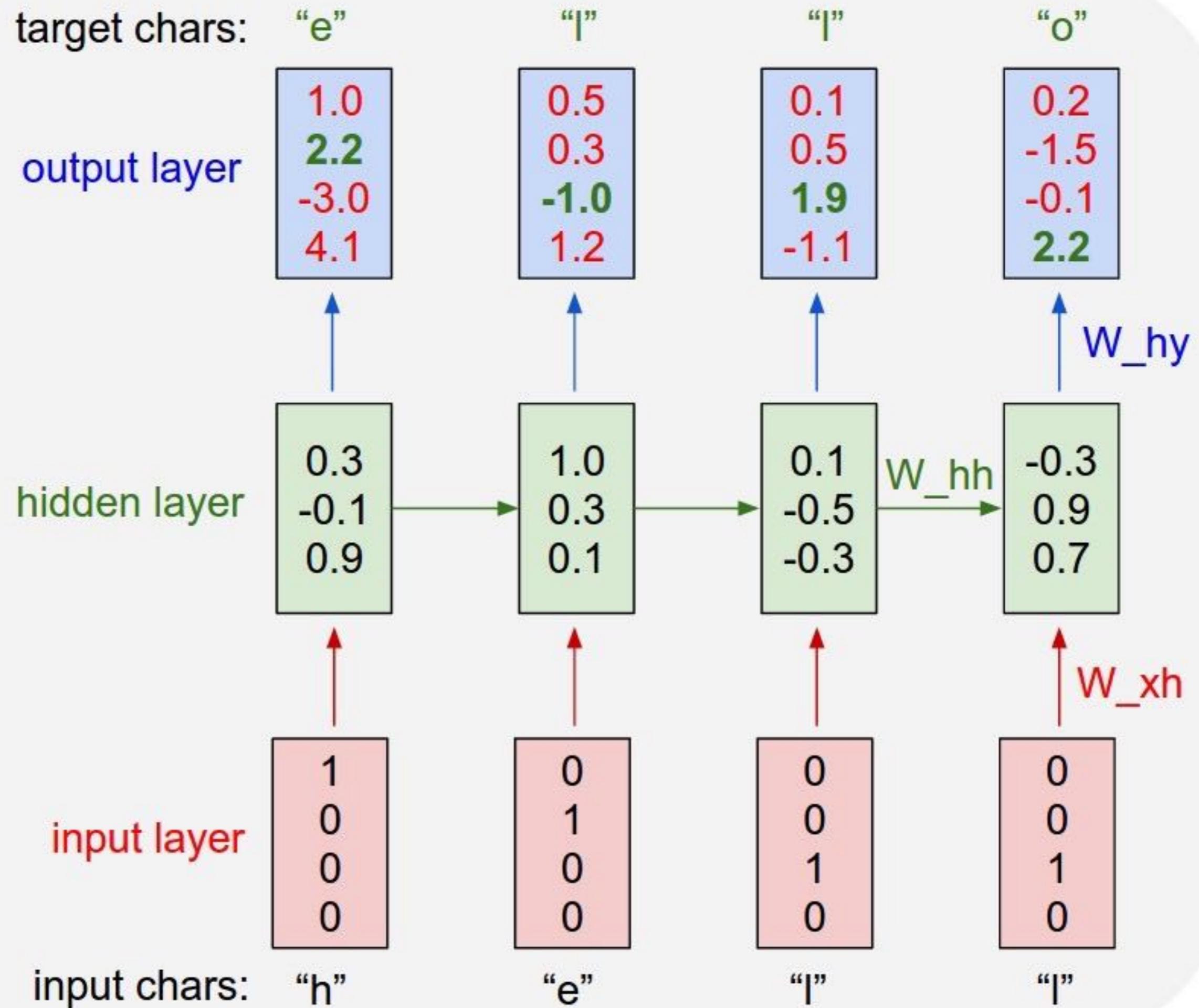
Style Transfer

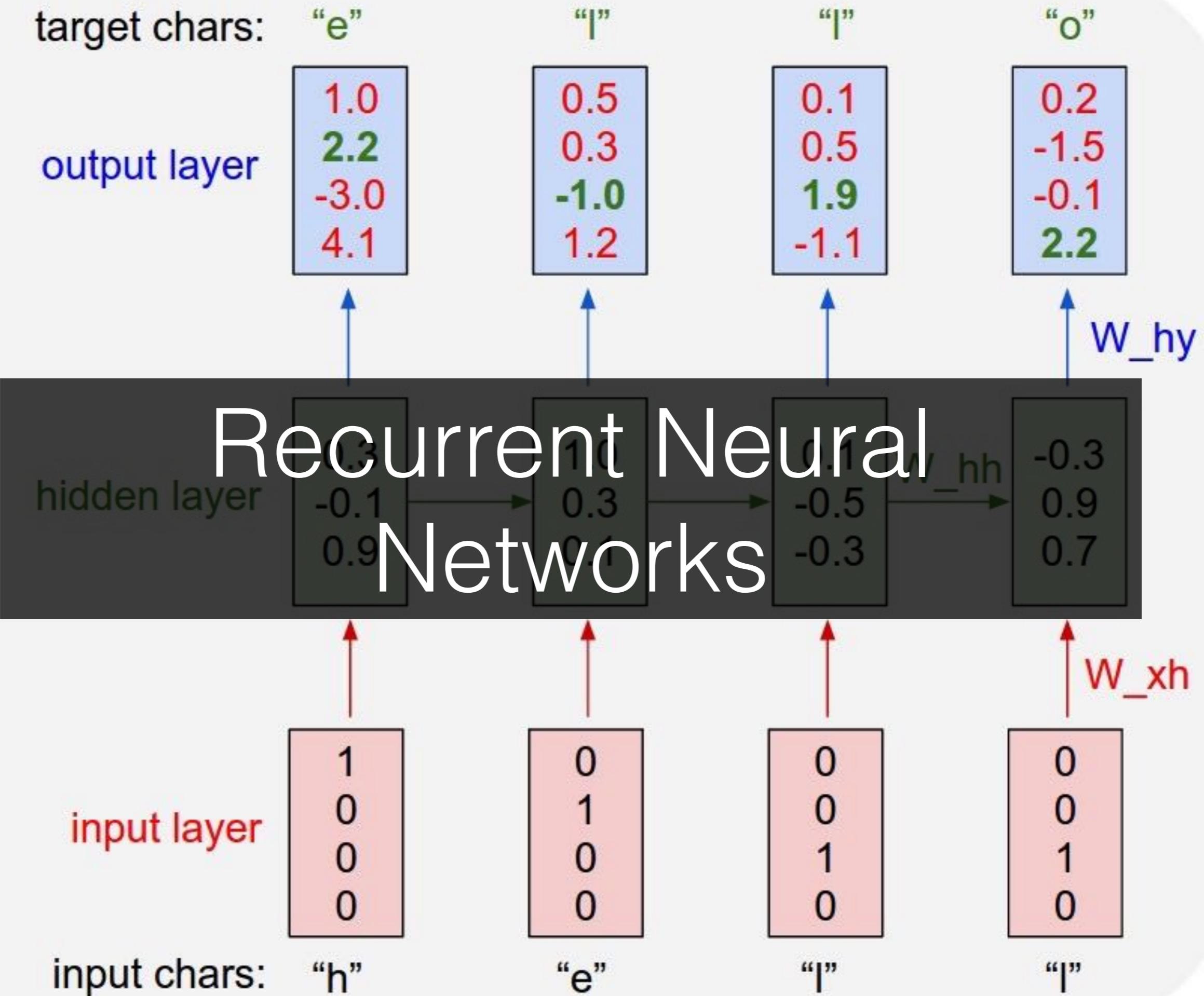


Style Transfer



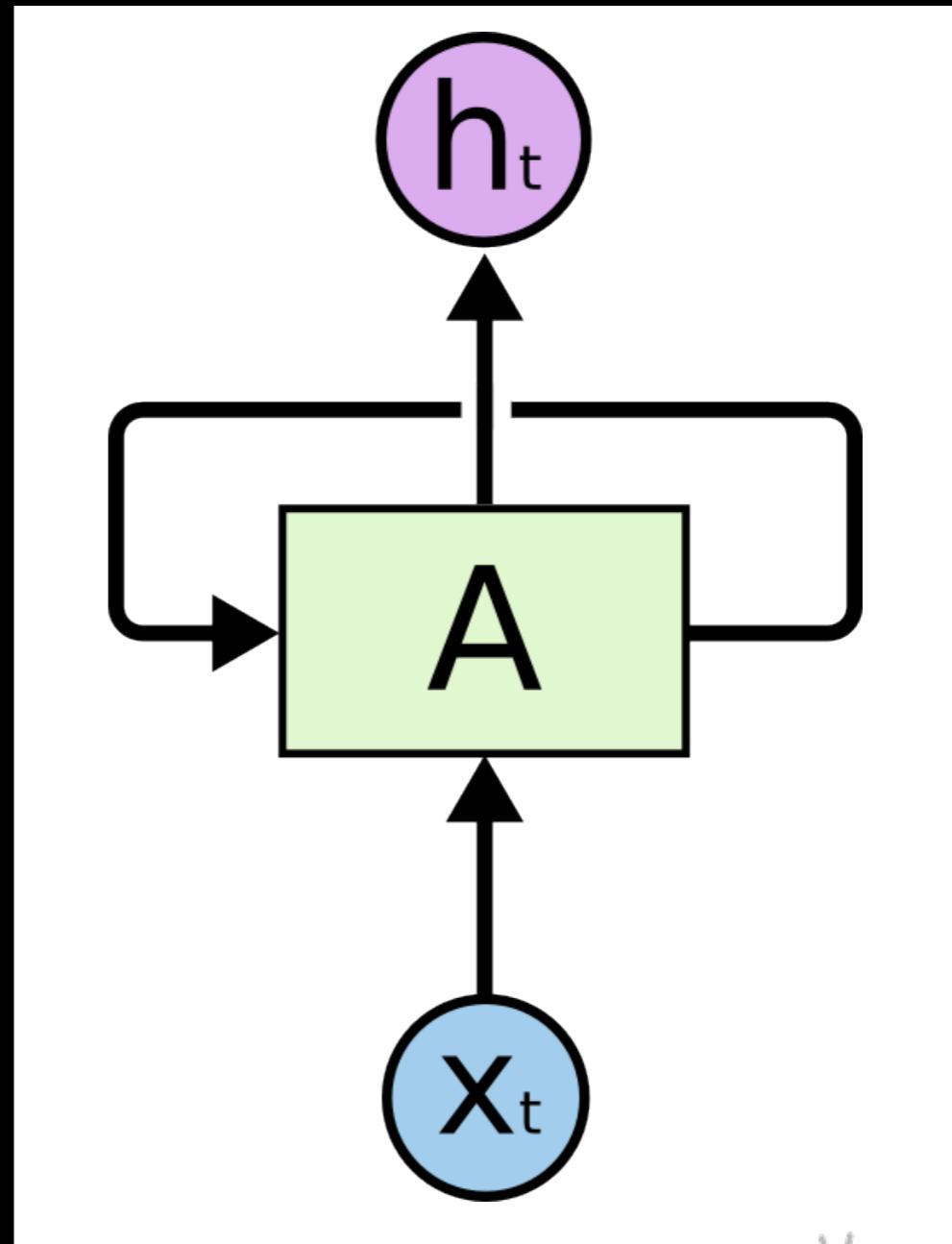
Deep-Style.io



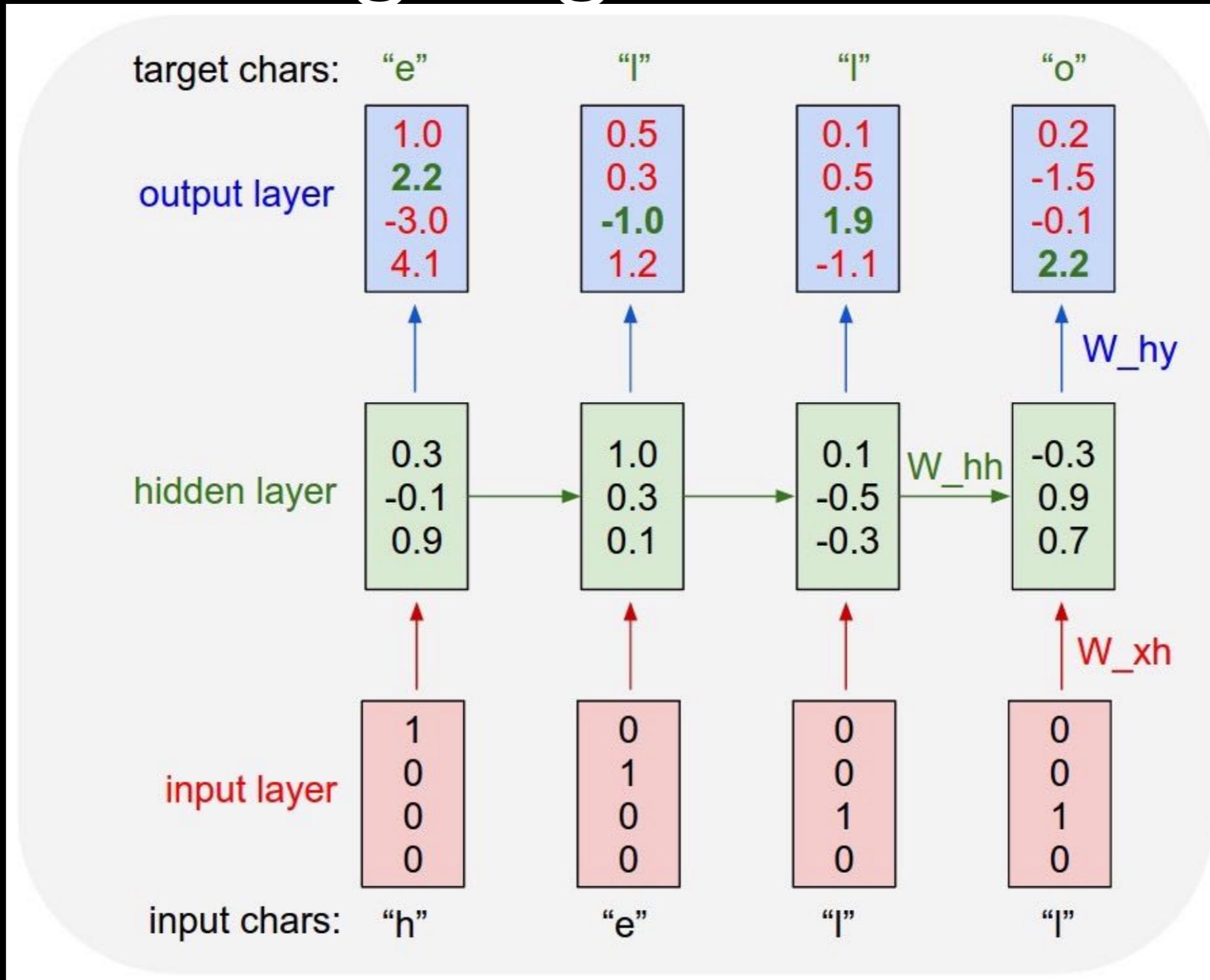


Recurrent NN

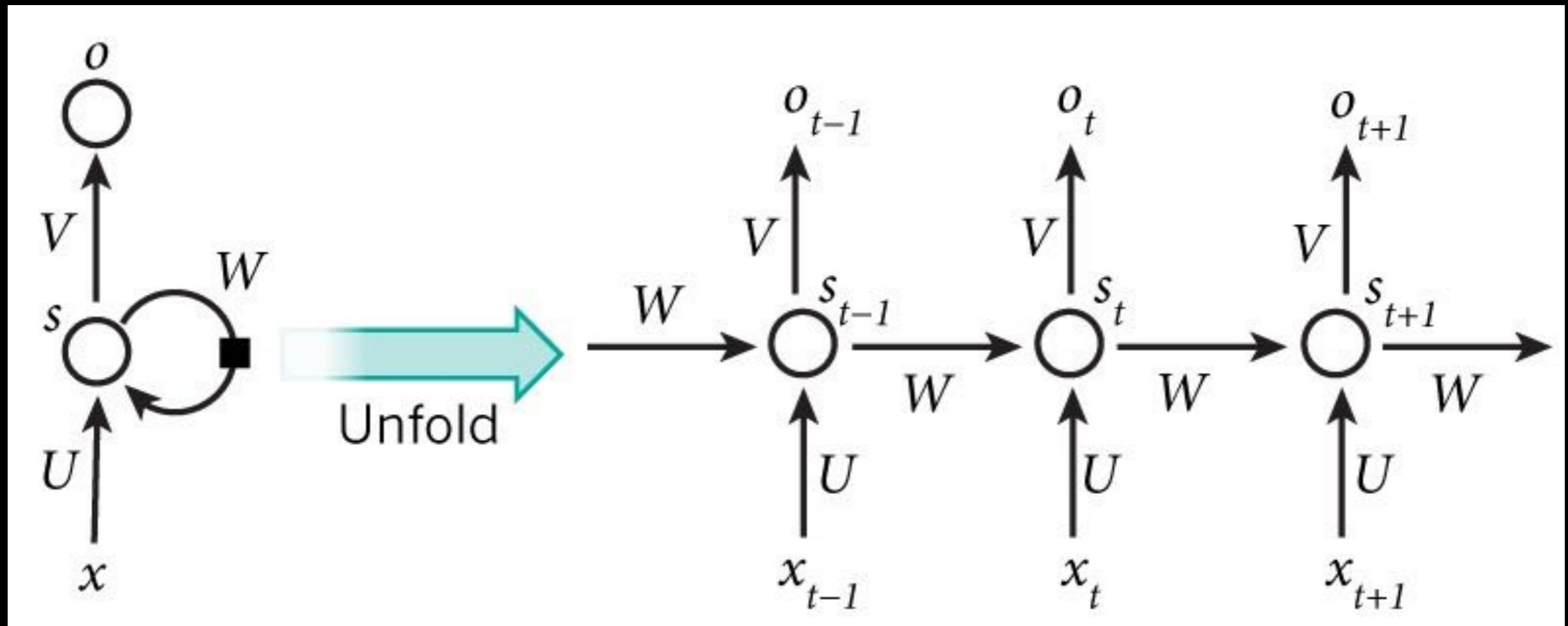
- Neural networks with memory
- Feed-forward NN: output exclusively depends on the current input
- Recurrent NN: output depends on current and previous states
- This is accomplished through lateral/backward connections which carry information while processing a sequence of inputs



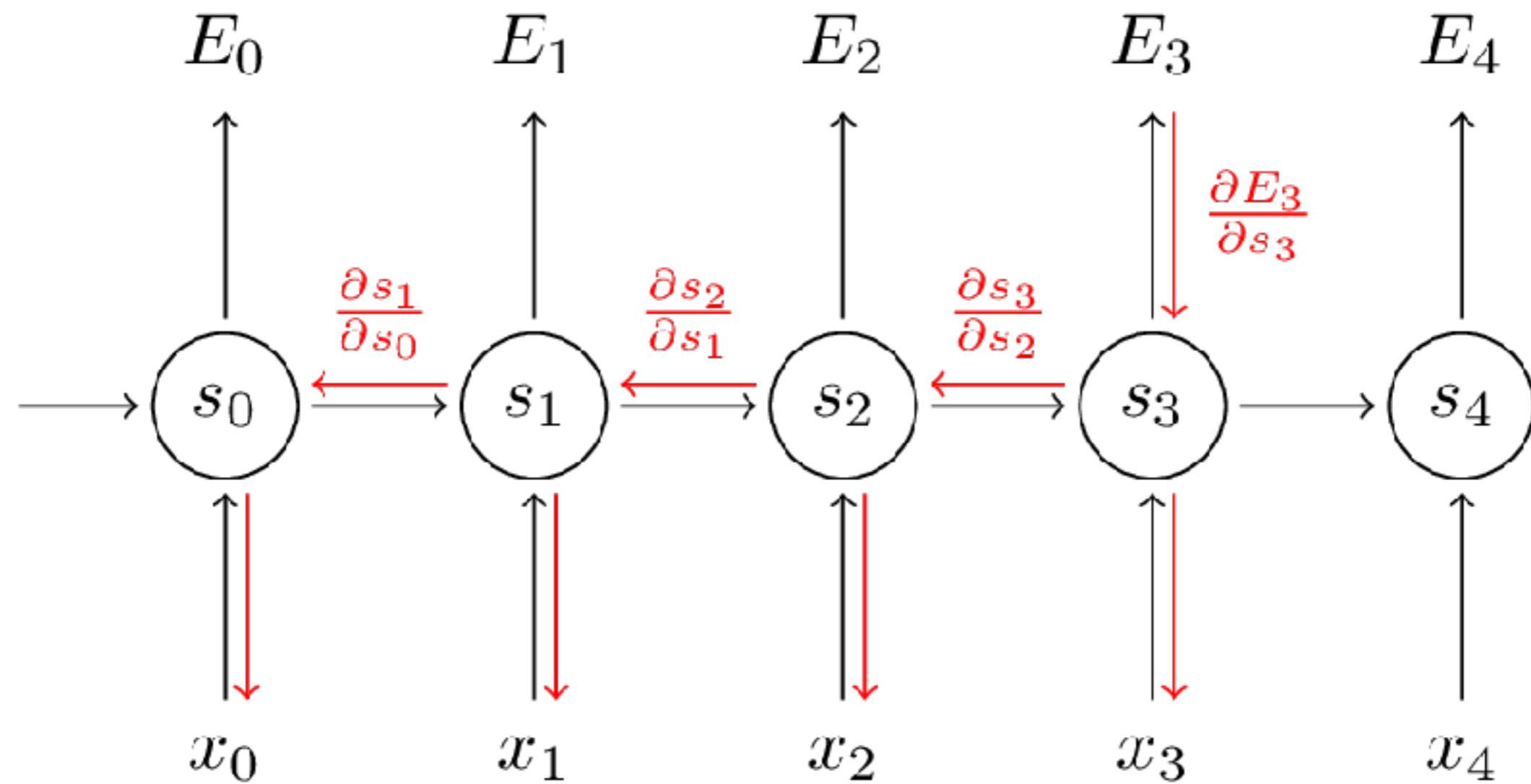
Character-level language model



Network unrolling

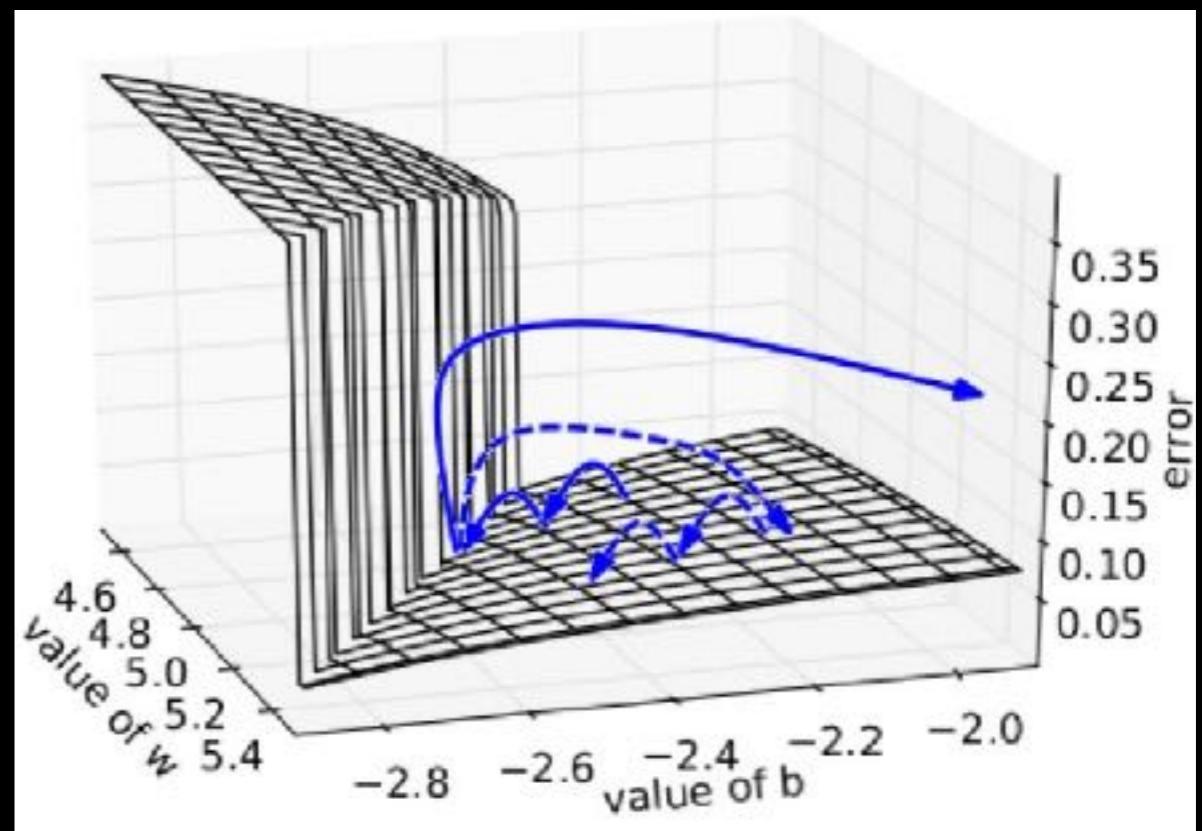


Backpropagation through time (BPTT)

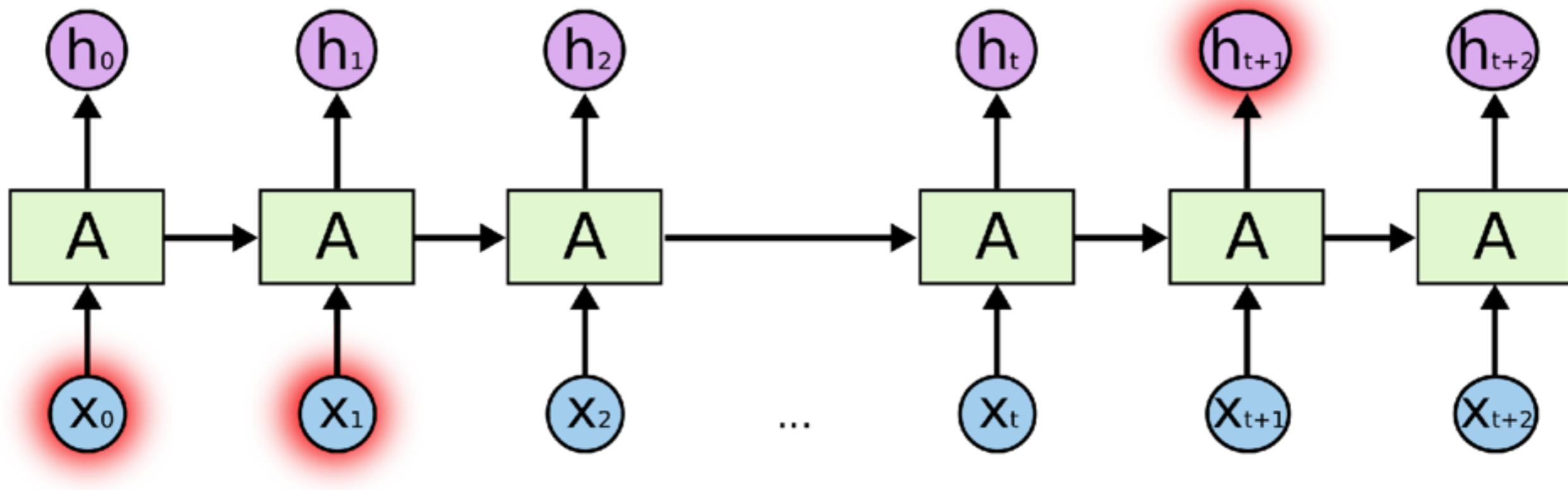


BPTT is hard

- The *vanishing* and the *exploding* gradient problem
- Gradients could vanish (or explode) when propagated several steps back
- This makes difficult to learn long-term dependencies.
- Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. *On the difficulty of training Recurrent Neural Networks*. Proc. of ICML, abs/1211.5063.



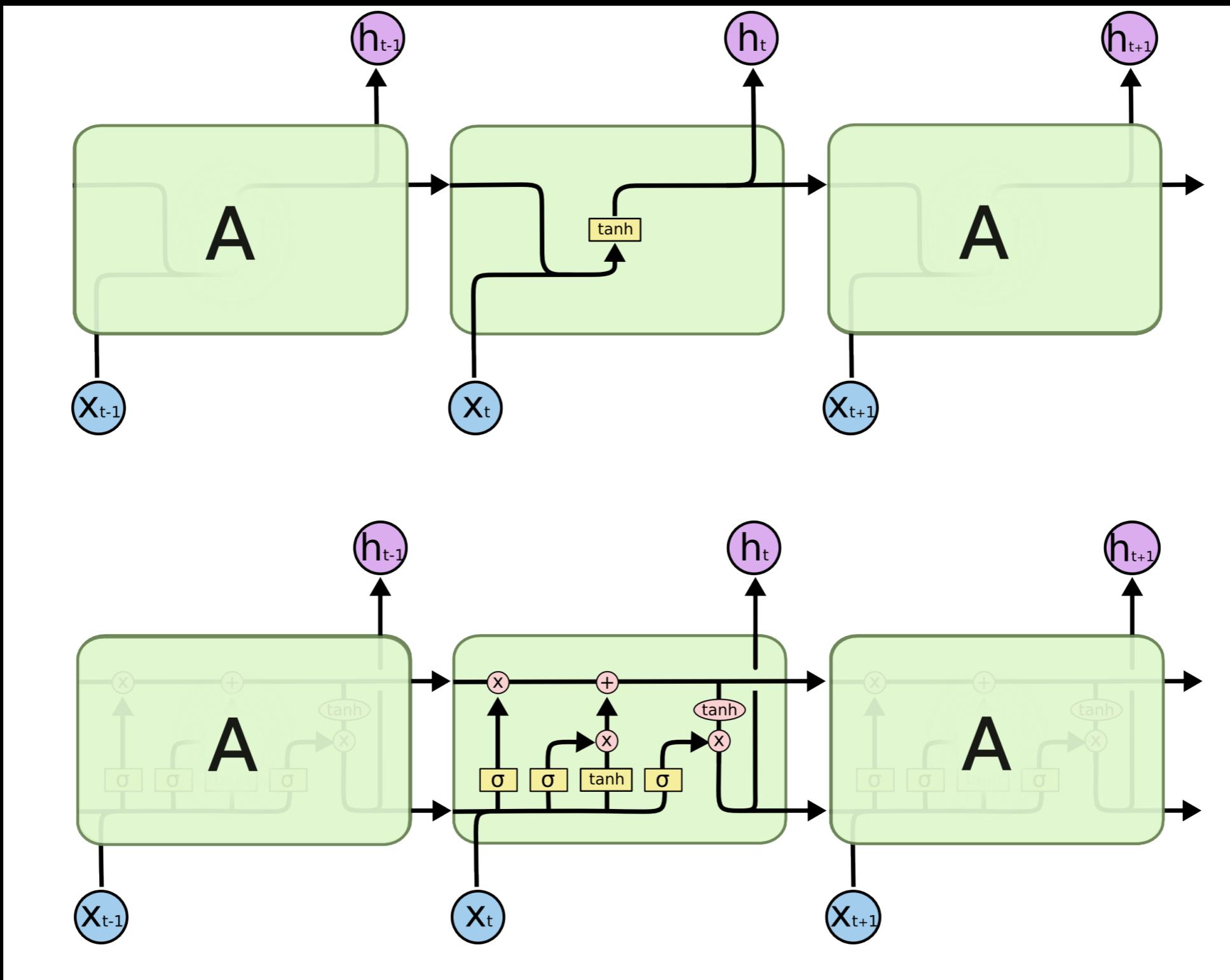
Long term dependencies



Long short-term memory (LSTM)

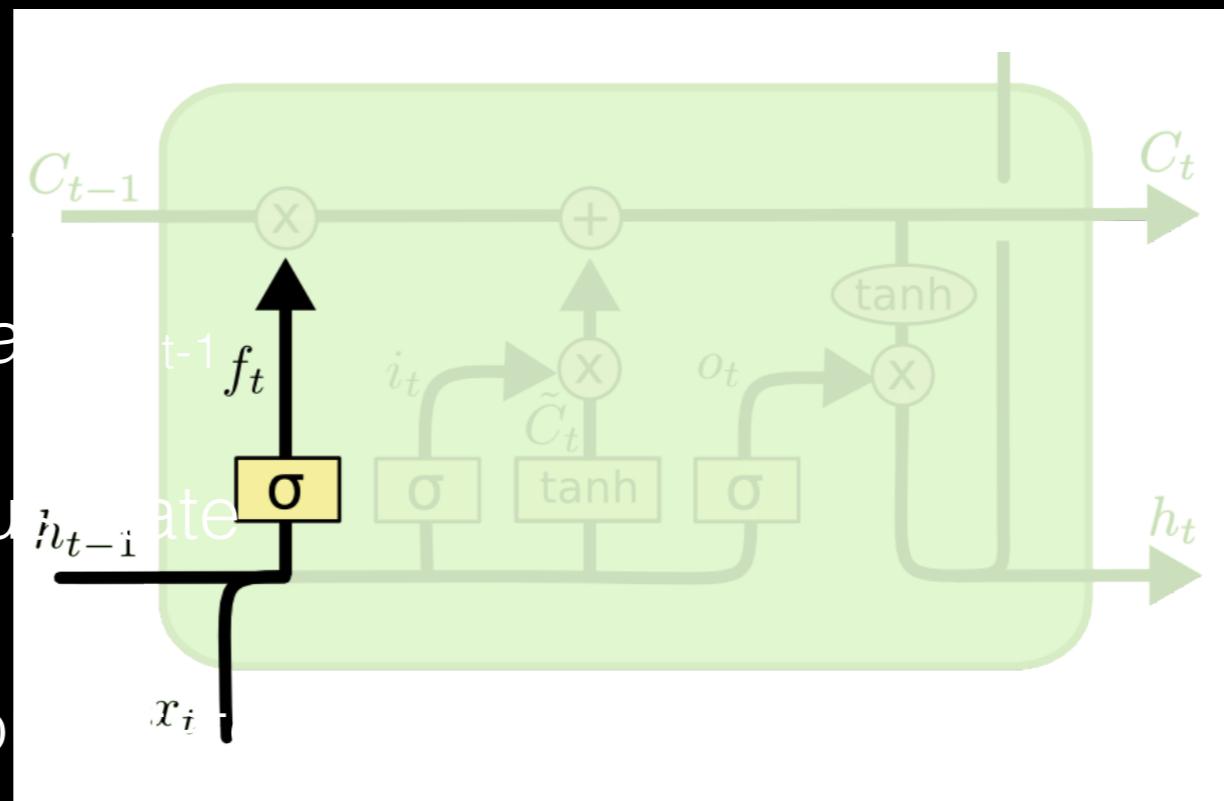
- LSTM networks solve the problem of long-term dependency problem.
- They use gates that allow to keep memory through long sequences and be updated only when required.
- Hochreiter, Sepp, and Jürgen Schmidhuber. "*Long short-term memory.*" Neural computation 9, no. 8 (1997): 1735-1780.

Conventional RNN vs LSTM



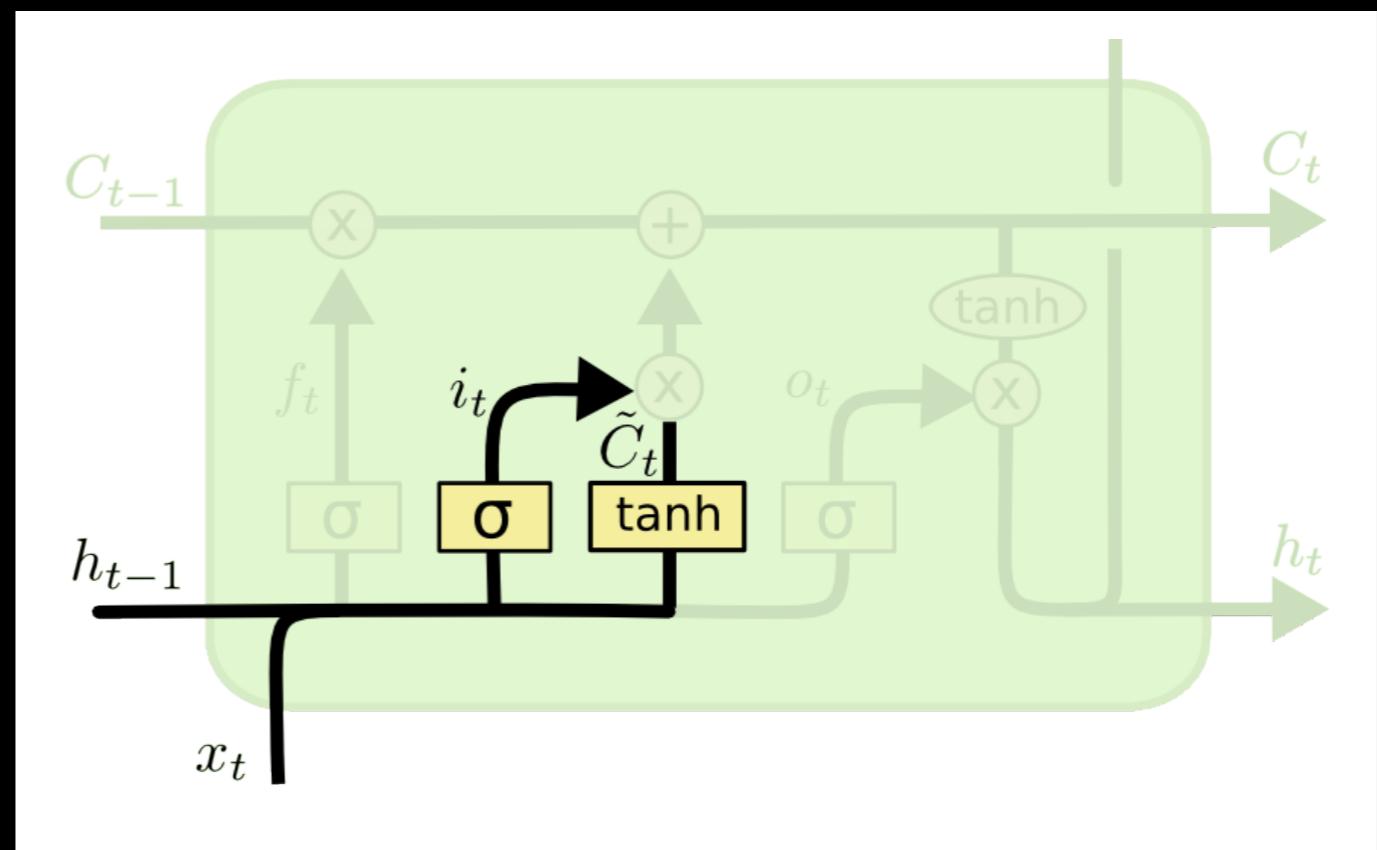
Forget gate

- Controls the flow of previous internal state
- $f_t=1 \Rightarrow$ keep previous state
- $f_t=0 \Rightarrow$ forget previous state

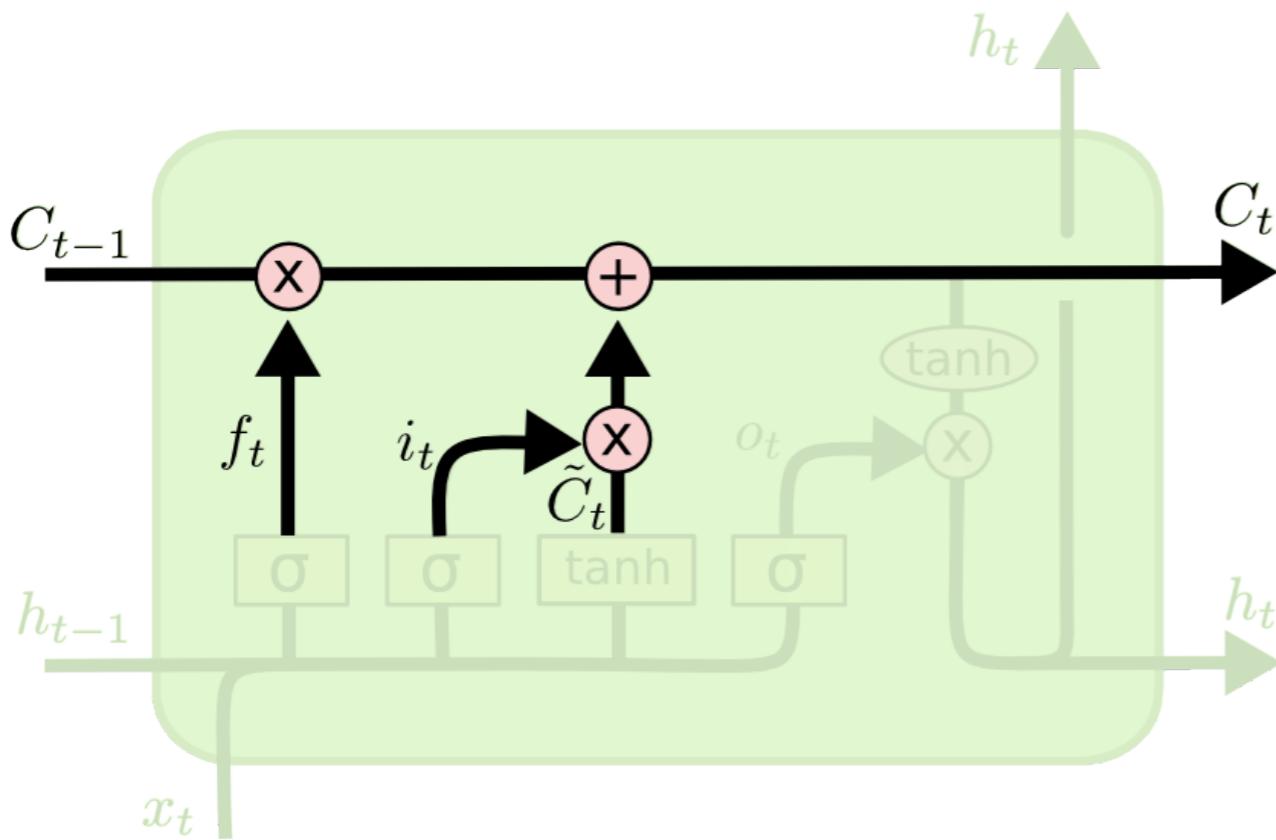


Input gate

- Controls the flow of the input state x_t
- $i_t=1 \Rightarrow$ take input into account
- $i_t=0 \Rightarrow$ ignore input

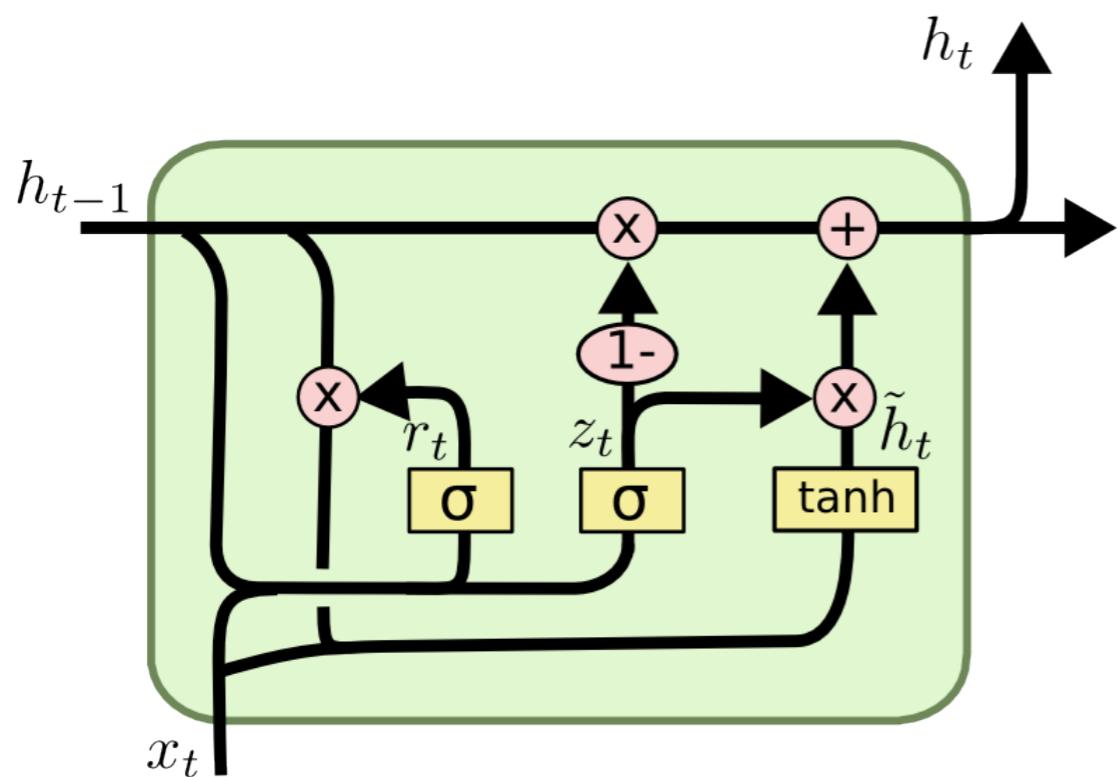


Current state calculation



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Gated recurrent units



$$z_t = \sigma (W_z \cdot [h_{t-1}, x_t])$$

$$r_t = \sigma (W_r \cdot [h_{t-1}, x_t])$$

$$\tilde{h}_t = \tanh (W \cdot [r_t * h_{t-1}, x_t])$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). ***Learning phrase representations using rnn encoder-decoder for statistical machine translation***. arXiv preprint arXiv:1406.1078.

Playing With LSTMs for Language Modeling

Interactive Demo

The Unreasonable Effectiveness of Recurrent Neural Networks

- Famous blog entry from Andrej Karpathy (UofS)
- Character-level language models based on multi-layer LSTMs.
- Data:
 - Shakspare plays
 - Wikipedia
 - LaTeX
 - Linux source code

Algebraic geometry book in LaTeX

Proof. Omitted. □

Lemma 0.1. Let \mathcal{C} be a set of the construction.

Let \mathcal{C} be a gerber covering. Let \mathcal{F} be a quasi-coherent sheaves of \mathcal{O} -modules. We have to show that

$$\mathcal{O}_{\mathcal{O}_X} = \mathcal{O}_X(\mathcal{L})$$

Proof. This is an algebraic space with the composition of sheaves \mathcal{F} on $X_{\text{étale}}$ we have

$$\mathcal{O}_X(\mathcal{F}) = \{\text{morph}_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}$$

where \mathcal{G} defines an isomorphism $\mathcal{F} \rightarrow \mathcal{F}$ of \mathcal{O} -modules. □

Lemma 0.2. This is an integer \mathcal{Z} is injective.

Proof. See Spaces, Lemma ??.

Lemma 0.3. Let S be a scheme. Let X be a scheme and X is an affine open covering. Let $\mathcal{U} \subset \mathcal{X}$ be a canonical and locally of finite type. Let X be a scheme. Let X be a scheme which is equal to the formal complex.

The following to the construction of the lemma follows.

Let X be a scheme. Let X be a scheme covering. Let

$$b : X \rightarrow Y' \rightarrow Y \rightarrow Y \rightarrow Y' \times_X Y \rightarrow X.$$

be a morphism of algebraic spaces over S and Y .

Proof. Let X be a nonzero scheme of X . Let X be an algebraic space. Let \mathcal{F} be a quasi-coherent sheaf of \mathcal{O}_X -modules. The following are equivalent

- (1) \mathcal{F} is an algebraic space over S .
- (2) If X is an affine open covering.

Consider a common structure on X and X the functor $\mathcal{O}_X(U)$ which is locally of finite type. □

This since $\mathcal{F} \in \mathcal{F}$ and $x \in \mathcal{G}$ the diagram

$$\begin{array}{ccccc}
 S & \xrightarrow{\quad} & & & \\
 \downarrow & & & & \\
 \xi & \longrightarrow & \mathcal{O}_{X'} & & \\
 \text{gor}_s & & \uparrow & \searrow & \\
 & & & & \\
 & & =a' & \longrightarrow & X \\
 & & \uparrow & & \downarrow \\
 & & =a' & \longrightarrow & \text{Mors}_{\text{etale}} \\
 & & & & d(\mathcal{O}_{X_{\text{étale}}}, \mathcal{G}) \\
 \text{Spec}(K_F) & & & &
 \end{array}$$

is a limit. Then \mathcal{G} is a finite type and assume S is a flat and \mathcal{F} and \mathcal{G} is a finite type f_* . This is of finite type diagrams, and

- the composition of \mathcal{G} is a regular sequence,
- $\mathcal{O}_{X'}$ is a sheaf of rings.

Proof. We have see that $X = \text{Spec}(R)$ and \mathcal{J} is a finite type representable by algebraic space. The property \mathcal{F} is a finite morphism of algebraic stacks. Then the cohomology of X is an open neighbourhood of U . □

Proof. This is clear that \mathcal{G} is a finite presentation, see Lemmas ??.

A reduced above we conclude that U is an open covering of C . The functor \mathcal{F} is a “field”

$$\mathcal{O}_{X,x} \longrightarrow \mathcal{F}_x \dashv \mathcal{I}(\mathcal{O}_{X_{\text{étale}}}) \longrightarrow \mathcal{O}_{X_t}^{-1} \mathcal{O}_{X_t}(\mathcal{O}_{X_t}^W)$$

is an isomorphism of covering of \mathcal{O}_{X_t} . If \mathcal{F} is the unique element of \mathcal{F} such that X is an isomorphism.

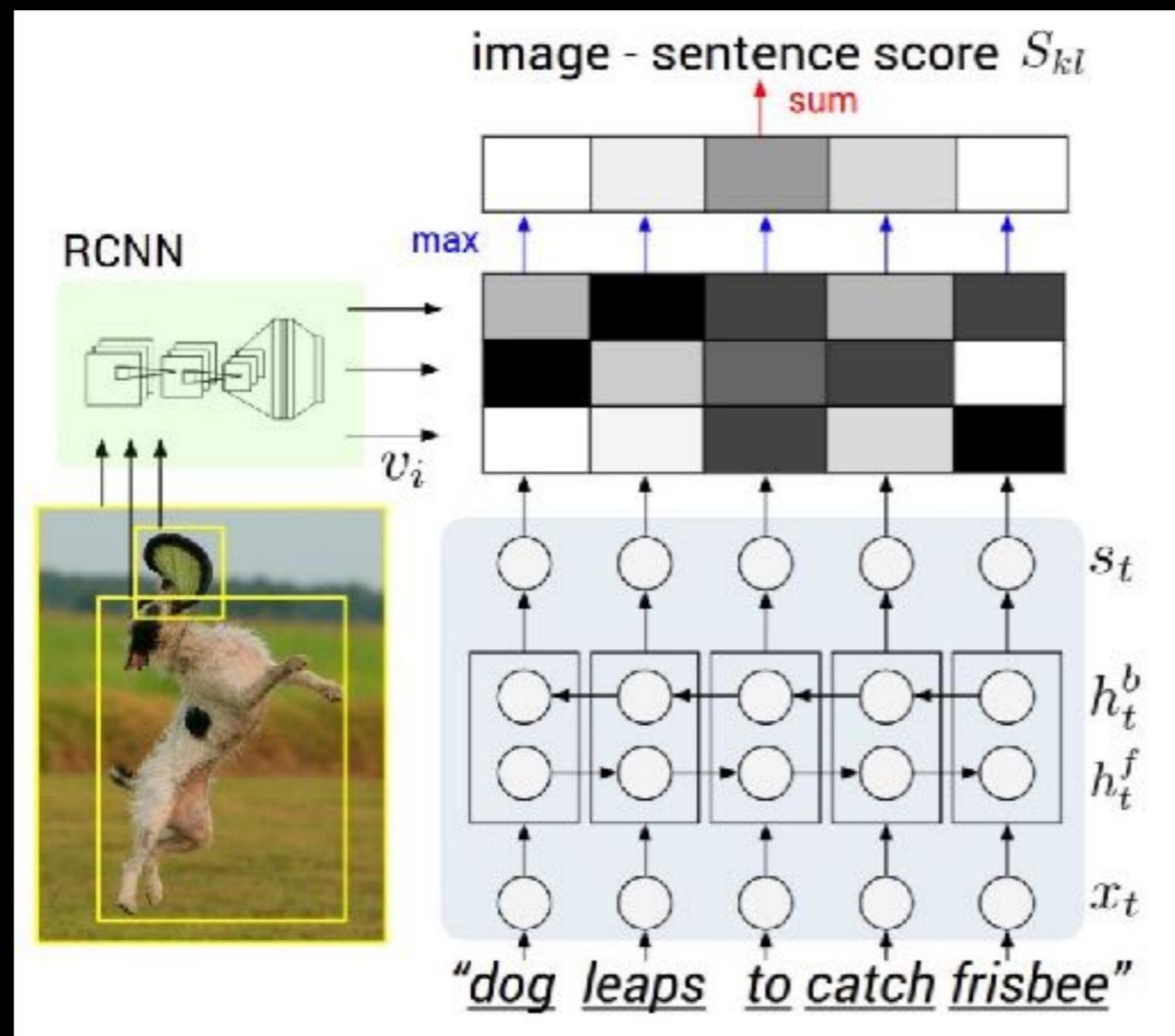
The property \mathcal{F} is a disjoint union of Proposition ?? and we can filtered set of presentations of a scheme \mathcal{O}_X -algebra with \mathcal{F} are opens of finite type over S . If \mathcal{F} is a scheme theoretic image points. □

If \mathcal{F} is a finite direct sum \mathcal{O}_{X_A} is a closed immersion, see Lemma ??, This is a sequence of \mathcal{F} is a similar morphism.

Linux source code

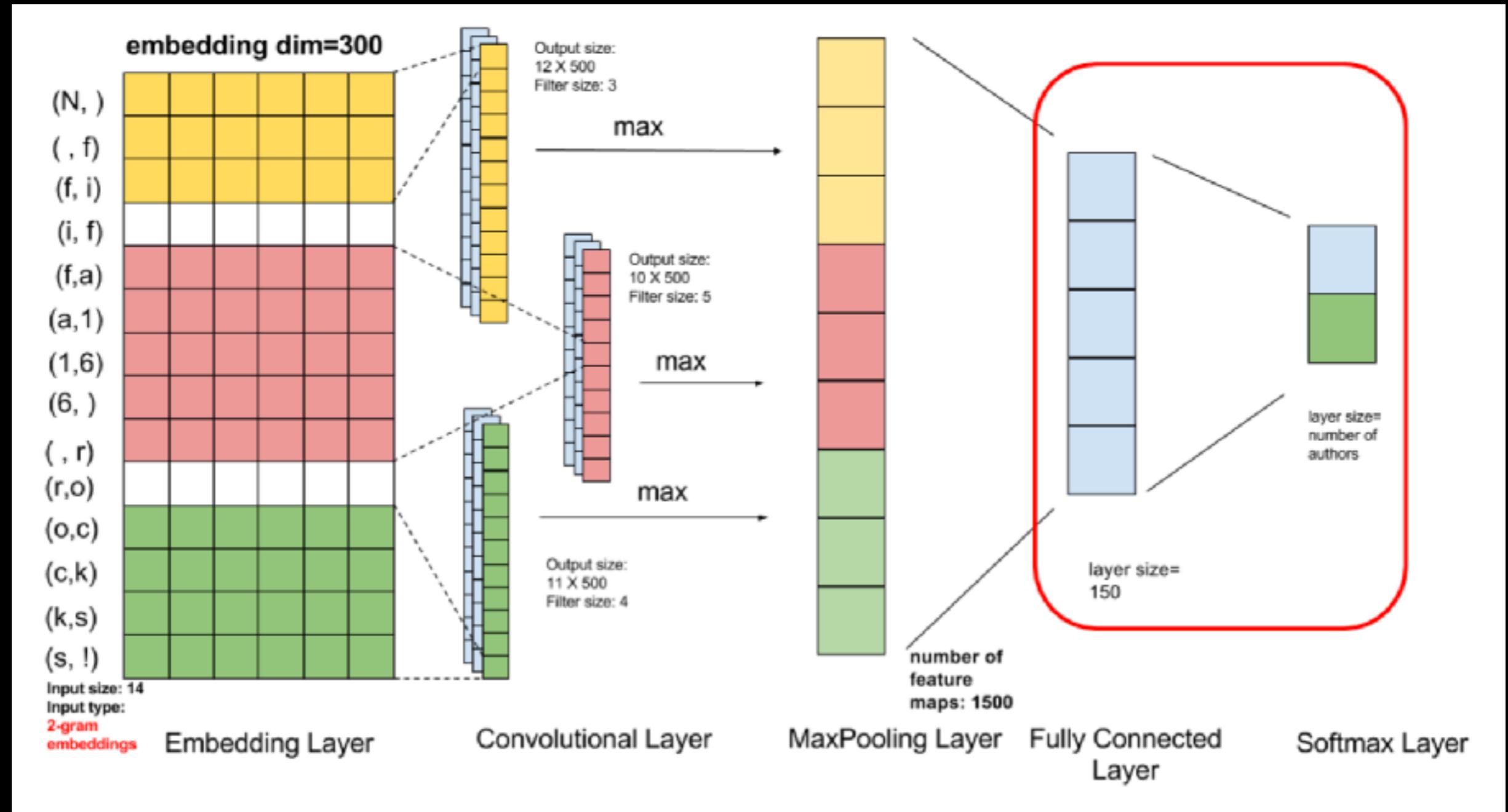
```
/*
 * Increment the size file of the new incorrect UI_FILTER group information
 * of the size generatively.
 */
static int indicate_policy(void)
{
    int error;
    if (fd == MARN_EPT) {
        /*
         * The kernel blank will coeld it to userspace.
         */
        if (ss->segment < mem_total)
            unblock_graph_and_set_blocked();
        else
            ret = 1;
        goto bail;
    }
    segaddr = in_SB(in.addr);
    selector = seg / 16;
    setup_works = true;
    for (i = 0; i < blocks; i++) {
        seq = buf[i++];
        bpf = bd->bd.next + i * search;
        if (fd) {
            current = blocked;
```

Multimodal models



Karpathy, Andrej, and Li Fei-Fei. "**Deep visual-semantic alignments for generating image descriptions.**" CVPR2015. arXiv preprint arXiv:1412.2306 (2014).

CNN for text



Additional Resources on RNN

- Christopher Olah, Understanding LSTM Networks, [<http://colah.github.io/posts/2015-08-Understanding-LSTMs/>]
- Denny Britz, Recurrent Neural Networks Tutorial, [<http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/>]
- Andrej Karpathy, The Unreasonable Effectiveness of Recurrent Neural Networks, [<http://karpathy.github.io/2015/05/21/rnn-effectiveness/>]
- Jürgen Schmidhuber, Recurrent Neural Networks, [<http://people.idsia.ch/~juergen/rnn.html>]

THE END



Gracias!

fagonzalezo@unal.edu.co

<http://mindlaboratory.org>

mind
LAB

machine learning
perception and discovery
discovery and perception

