ENSAE ParisTech

École nationale de la statistique et de l'administration économique

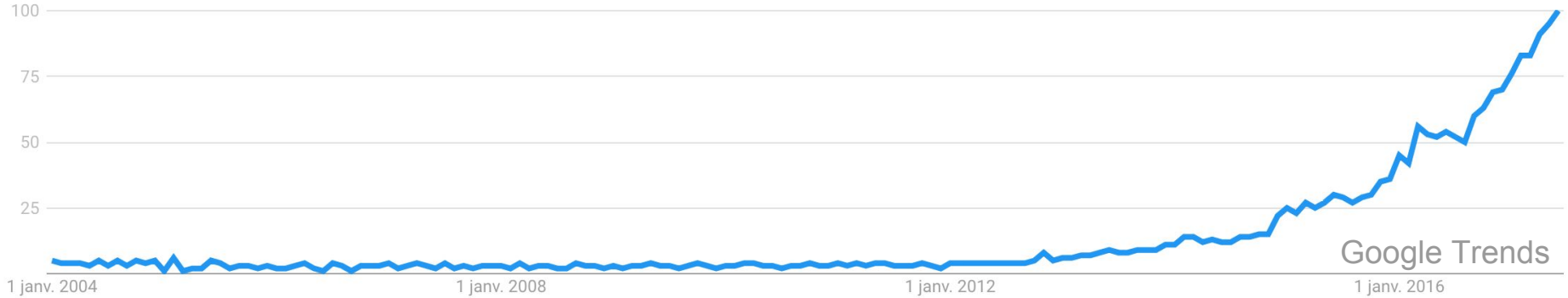
université PARIS-SACLAY

DEEP LEARNING

1.

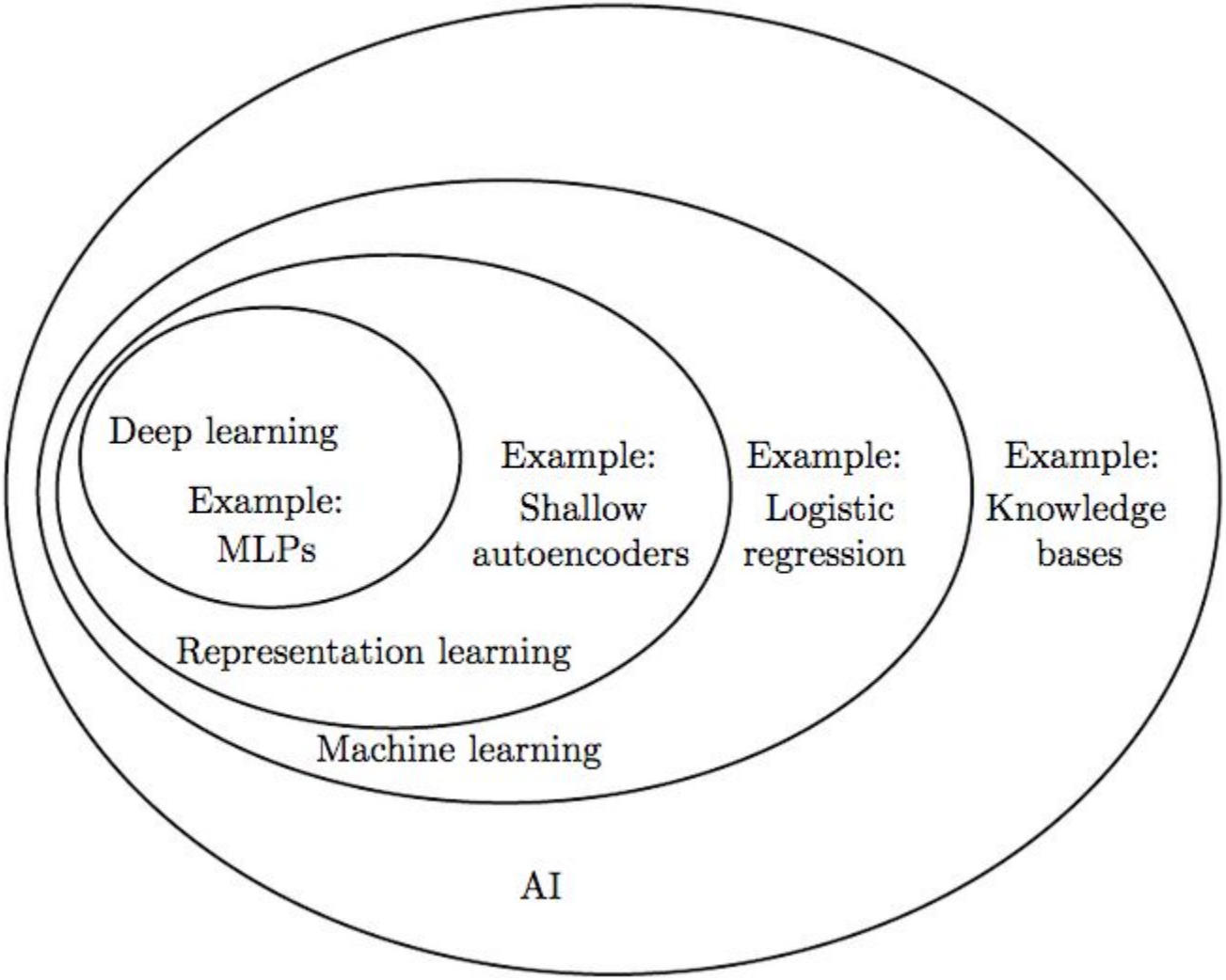
Deep Learning : introduction

Deep Learning



What Deep Learning is NOT

- NOT just a buzzword
- NOT the most efficient machine learning algorithm
- NOT General Artificial Intelligence


Deep learning
Example:
MLPs
Example:
Shallow
autoencoders
Example:
Logistic
regression
Example:
Knowledge
bases
Representation learning
Machine learning
AI

Representation Learning

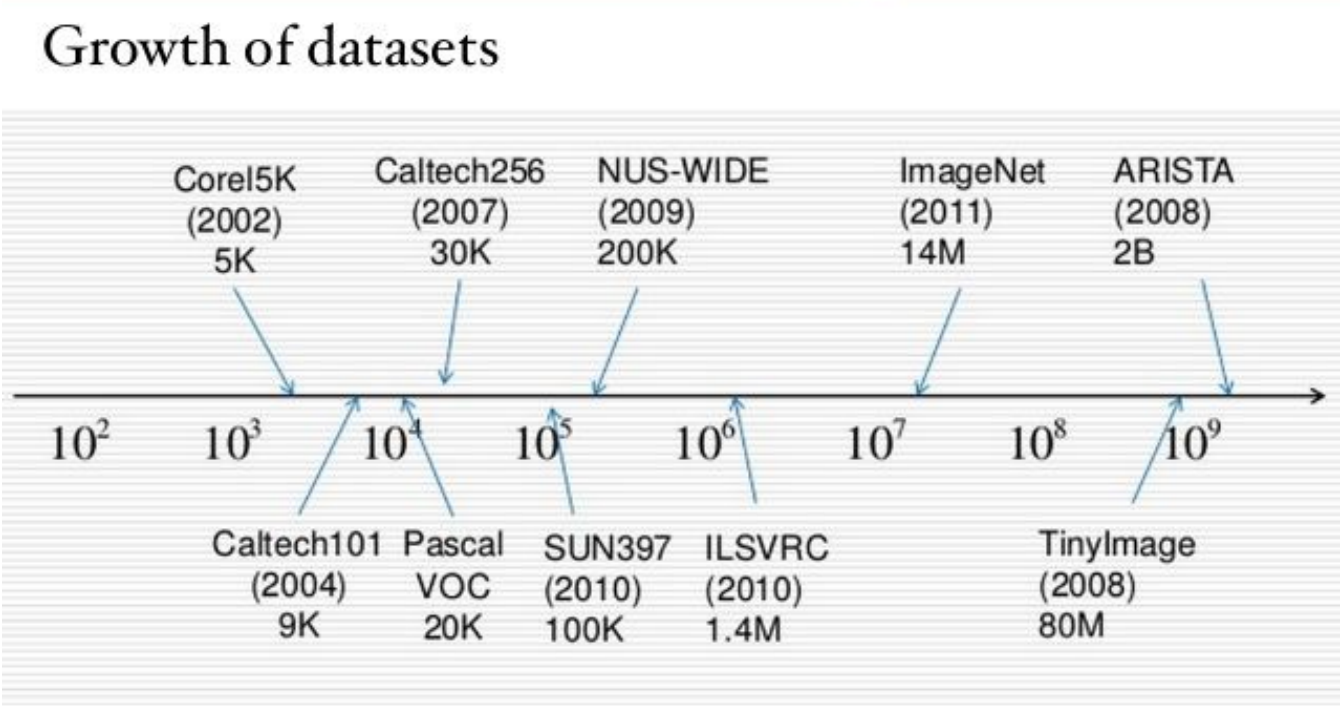
- Creating features is very time consuming

- Deep Learning can learn features from unstructured data
  - Pixels
  - Text
  - Sound

History: why now?

- More data
- Better hardware: CPU, GPU
- Better algorithms for training deep networks

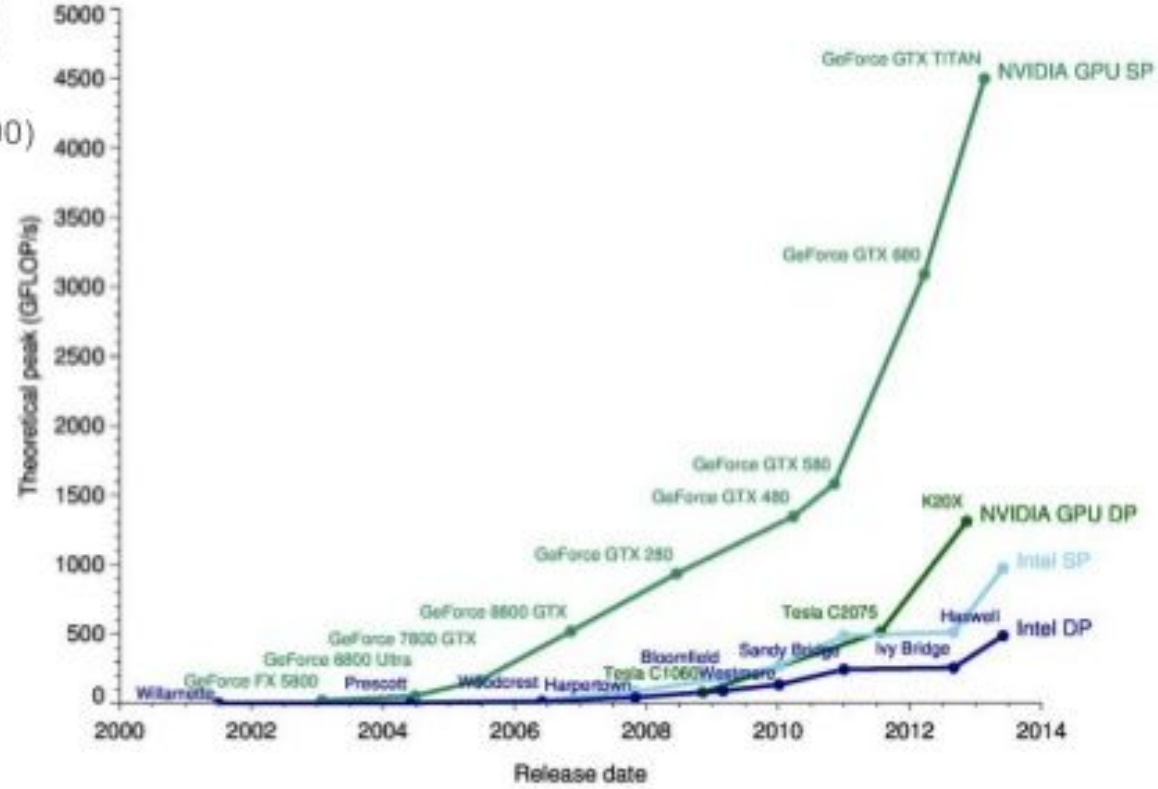
History: why now?



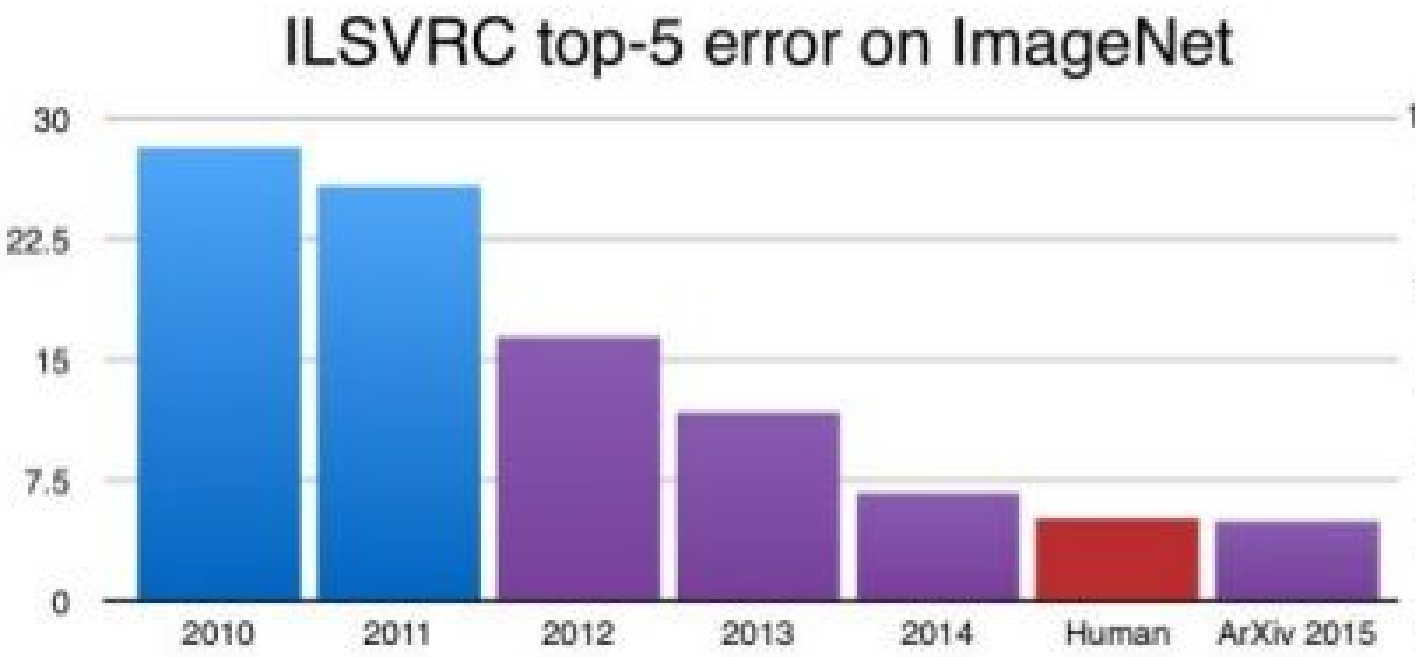
History: why now?

GPU computing power is growing

- NVIDIA DGX-1 ($129,000)
  - 170 TFLOPS (FP16)
  - 85 TFLOPS (FP32)
- NVIDIA GTX Titan X ($1000)
  - 6.1 TFLOPS (FP32)
- NVIDIA Drive PX
  - 2.3 TFLOPS
- NVIDIA Drive PX-2
  - 8.0 TFLOPS
- Intel Core i7-6700K
  - ~0.1-0.2 TFLOPS



Impact on computer vision



- Blue: Traditional Computer Vision
- Purple: Deep Learning
- Red: Human

Impact on speech recognition

Speech Recognition: Word Error Rate (WER)

"Google now has just an 8 percent error rate. Compare that to 23 percent in 2013" (2015)
http://venturebeat.com/2015/05/28/google-says-its-speech-recognition-technology-now-has-only-an-8-word-error-rate/

IBM Watson. "The performance of our new system – an 8% word error rate – is 36% better than previously reported external results." (2015)
https://developer.ibm.com/watson/blog/2015/05/26/ibm-watson-announces-breakthrough-in-conversational-speech-transcription/

Baidu. "We are able to reduce error rates of our previous end-to-end system in English by up to 43%, and can also recognize Mandarin speech with high accuracy. Creating high-performing recognizers for two very different languages, English and Mandarin, required essentially no expert knowledge of the languages" (2015)
http://arxiv.org/abs/1512.02595

Application: car driving

Actually a "Perception to Action" system. The visual perception and control system is a Deep learning architecture trained end to end to transform pixels from the cameras into steering angles. And this car uses regular color cameras, not LIDARS like the Google cars. It is watching the driver and learns.

https://www.youtube.com/watch?v=YuyT2SDcYrU

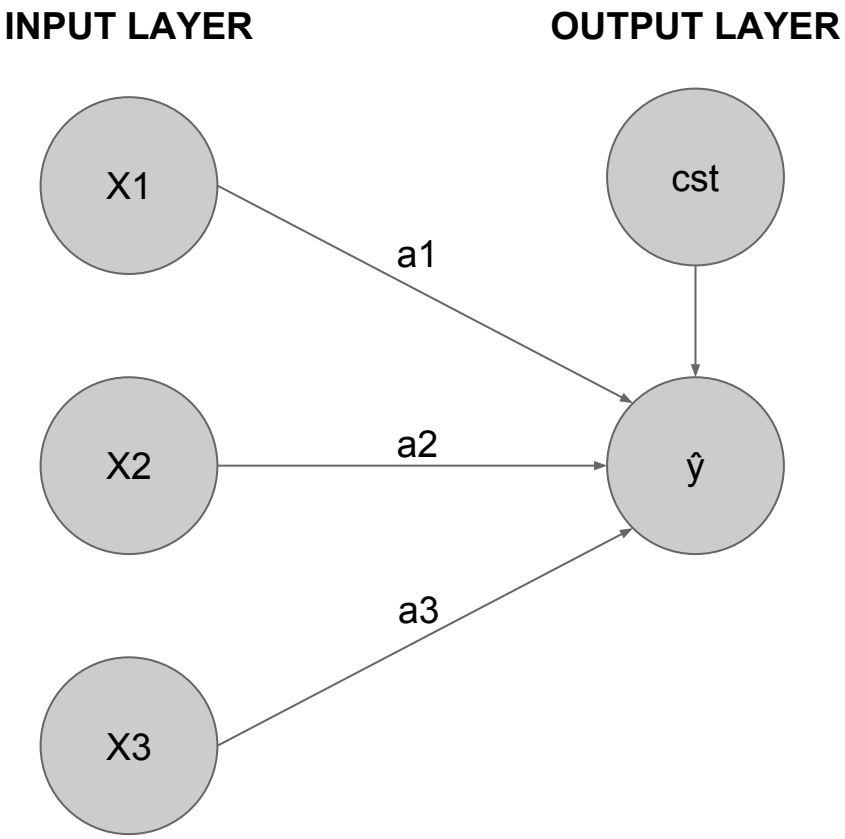
Application: Visual Question Answering

https://avisingh599.github.io/deeplearning/visual-qa/

2.

Getting started : Gradient descent

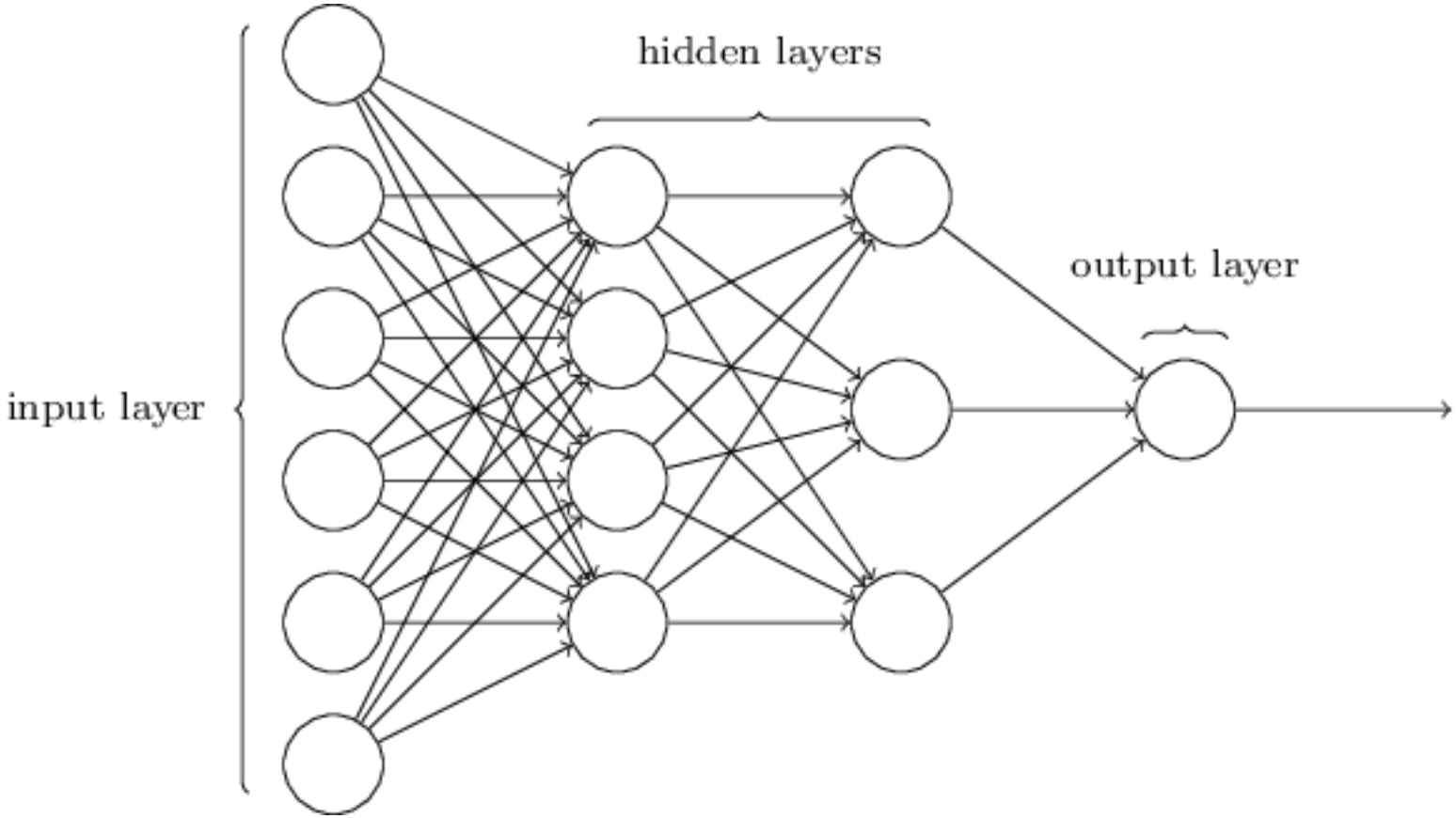
Remember the logistic regression?



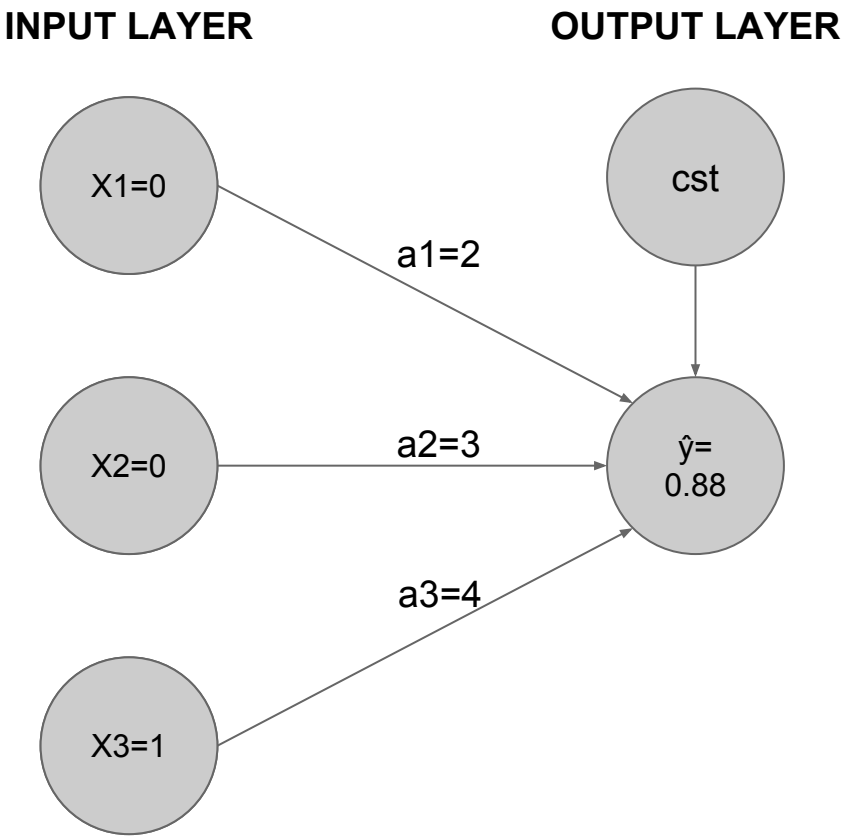
$\hat{y} = f(a1.x1 + a2.x2 + a3.x3)$

$f(z)=1/(1+\exp(-z))$

Deep Learning is a biiiiiiiiig logistic regression!



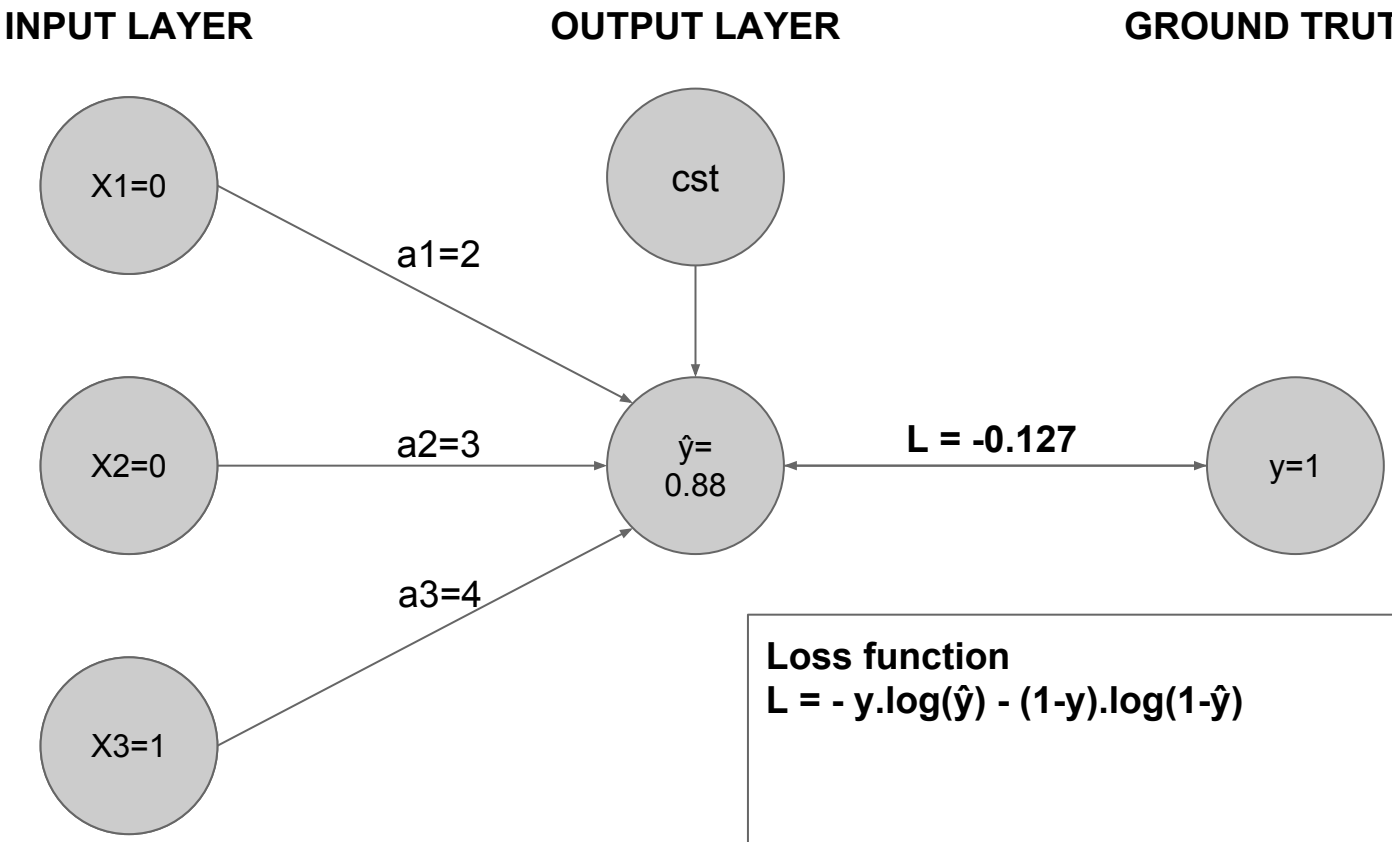
Backpropagation: intro



$\hat{y} = f(a1.x1 + a2.x2 + a3.x3)$

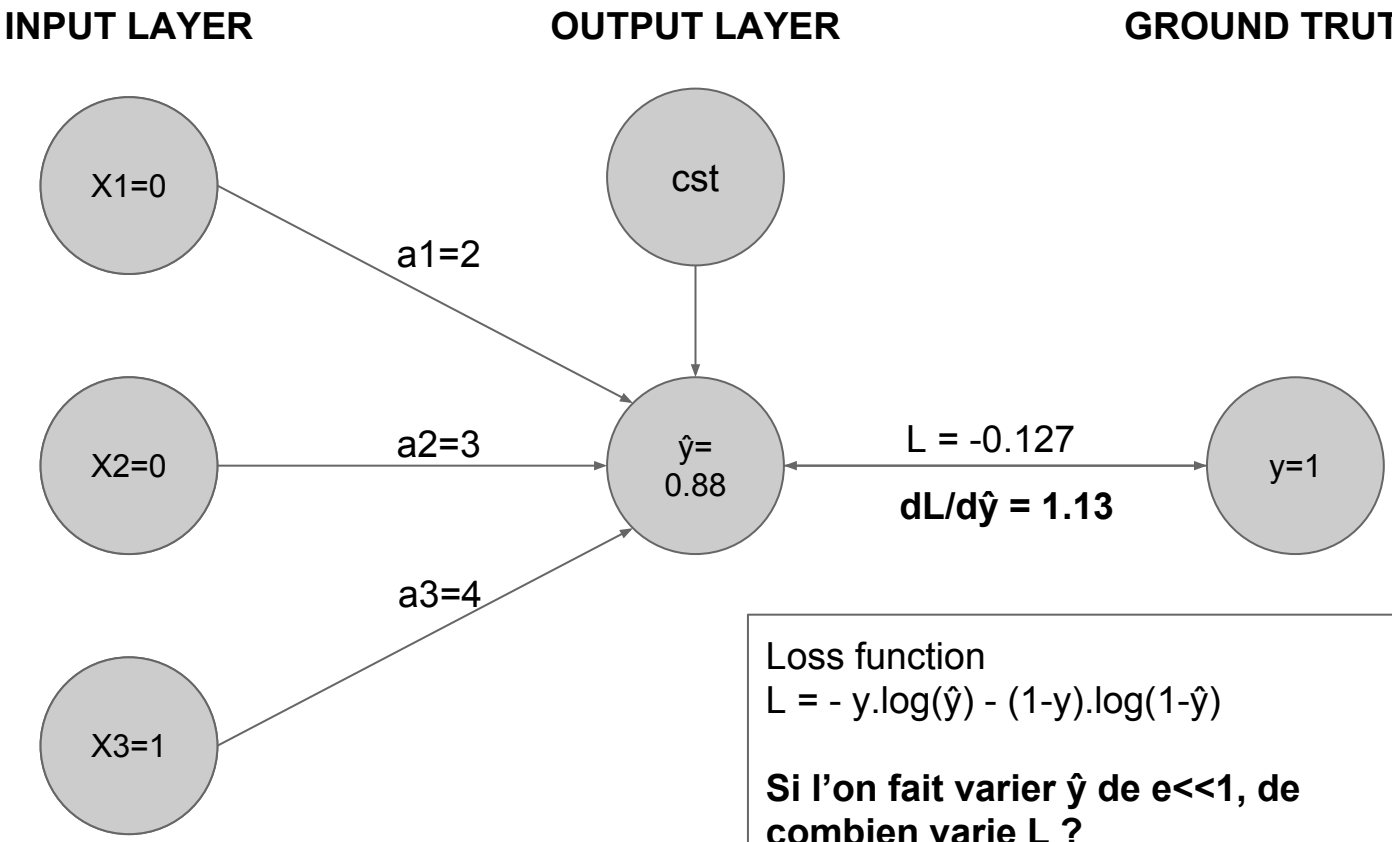
Avec $f(z)=1/(1+\exp(-z))$

Backpropagation: intro



Loss function
L = - y.log(ŷ) - (1-y).log(1-ŷ)

Backpropagation: intro

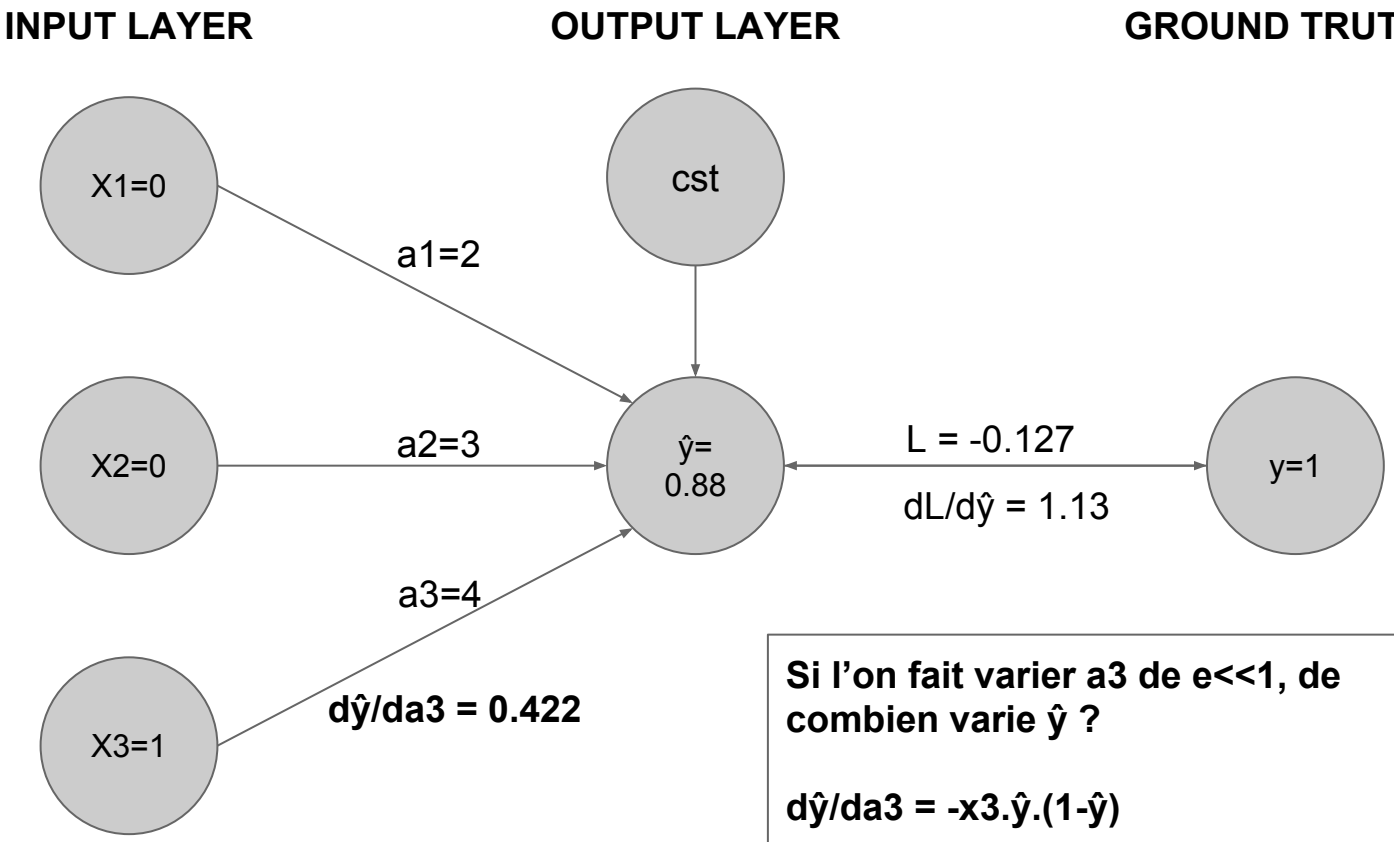


Loss function
L = - y.log(ŷ) - (1-y).log(1-ŷ)

Si l'on fait varier ŷ de e<<1, de combien varie L ?

dL/dŷ = -y/ŷ - (1-y) / (1-ŷ)

Backpropagation: intro



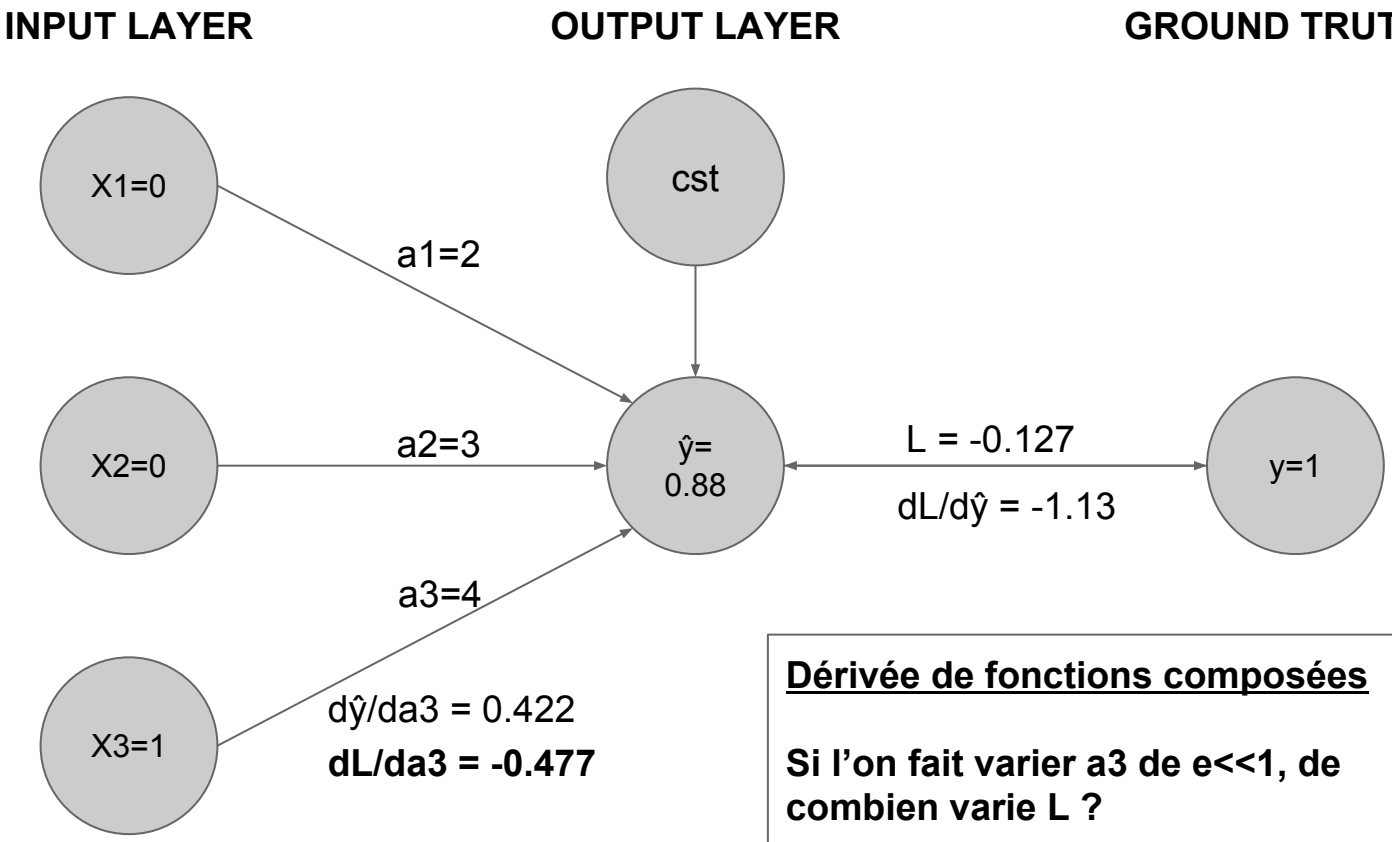
$\hat{y} = f(a1.x1 + a2.x2 + a3.x3)$

Avec $f(z)=1/(1+\exp(-z))$

Si l'on fait varier a3 de e<<1, de combien varie ŷ ?

$d\hat{y}/da3 = -x3.\hat{y}.(1-\hat{y})$

Backpropagation: intro

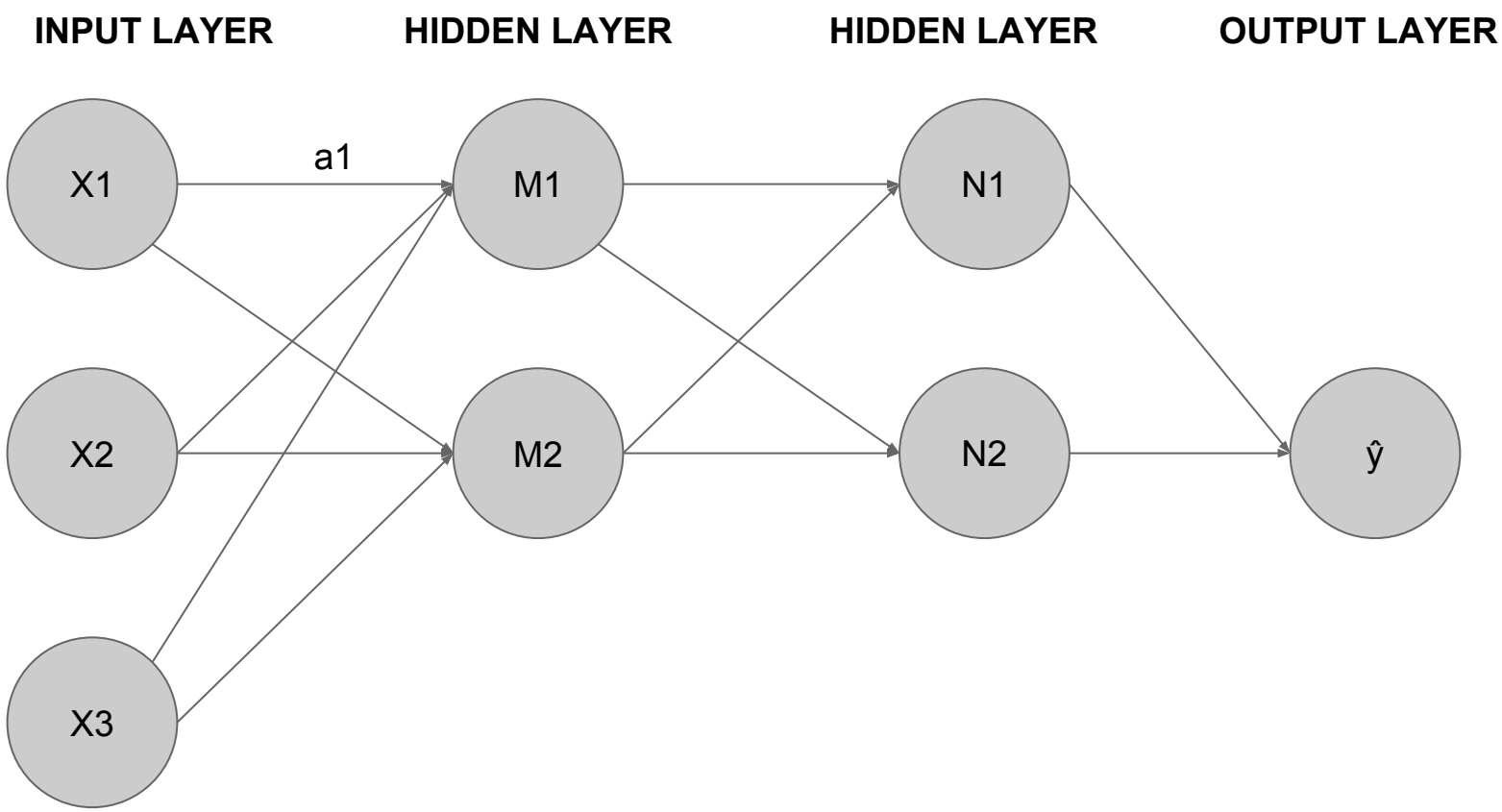


<u>Dérivée de fonctions composées</u>

Si l'on fait varier a3 de e<<1, de combien varie L ?

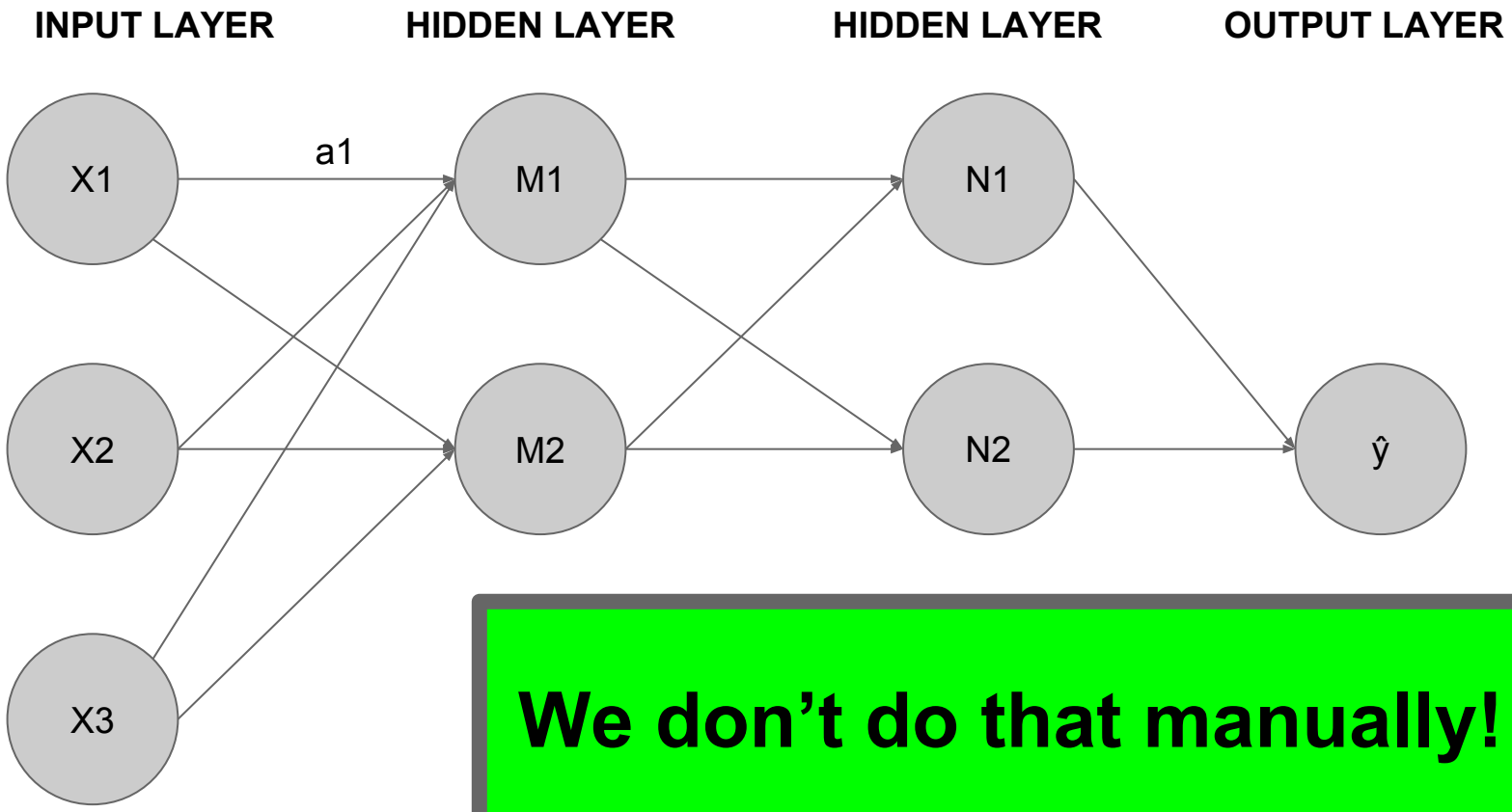
dL/da3 = da3/dŷ * dŷ/dL !

Deep Learning is a biiiiiiiiig logistic regression!



dL/a1 = dL/dŷ . (dŷ/dN1.dN1/dM1 + dŷ/dN2.dN2/dM1) . dM1/da1

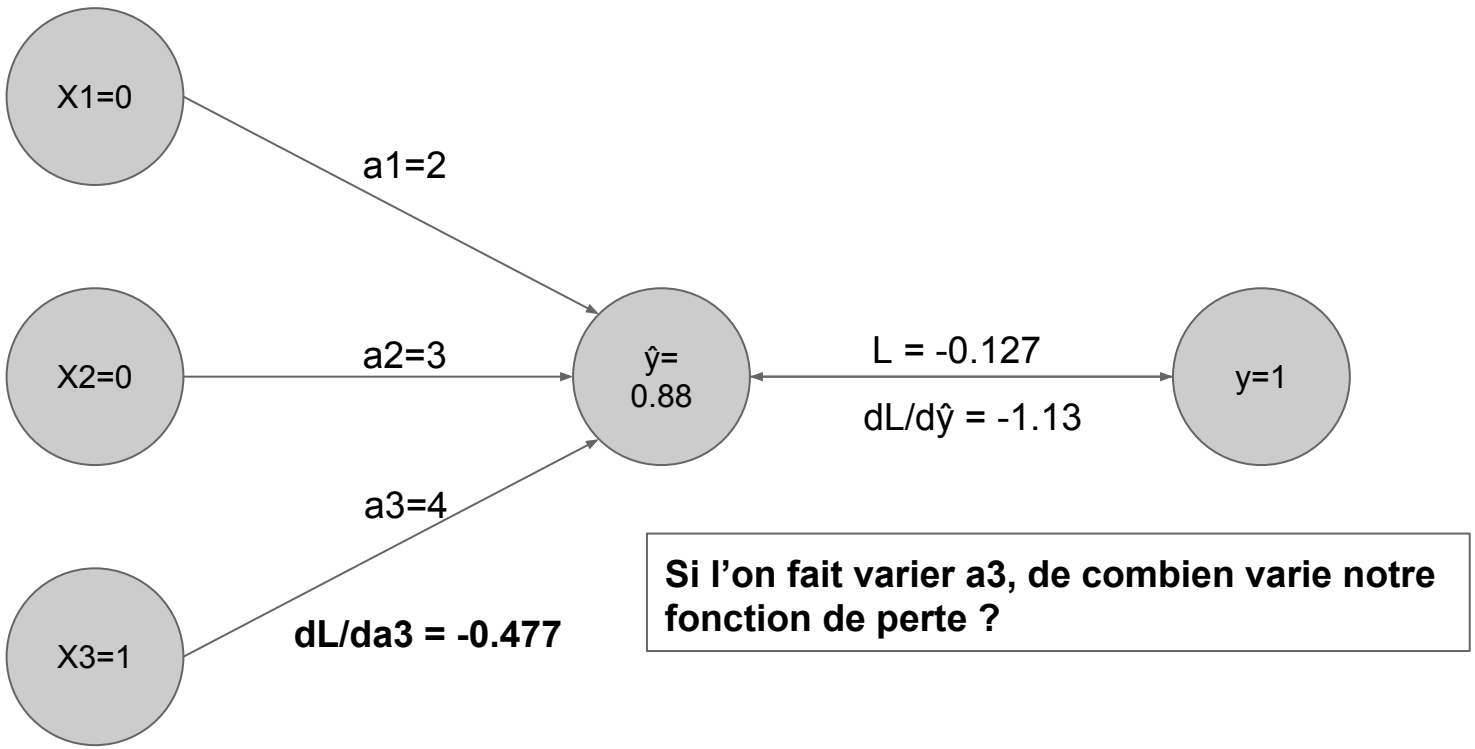
Deep Learning is a biiiiiiiiig logistic regression!



We don’t do that manually! ;)

dL/a1 = dL/dŷ . (dŷ/dN1.dN1/dM1 + dŷ/dN2.dN2/dM1) . dM1/da1

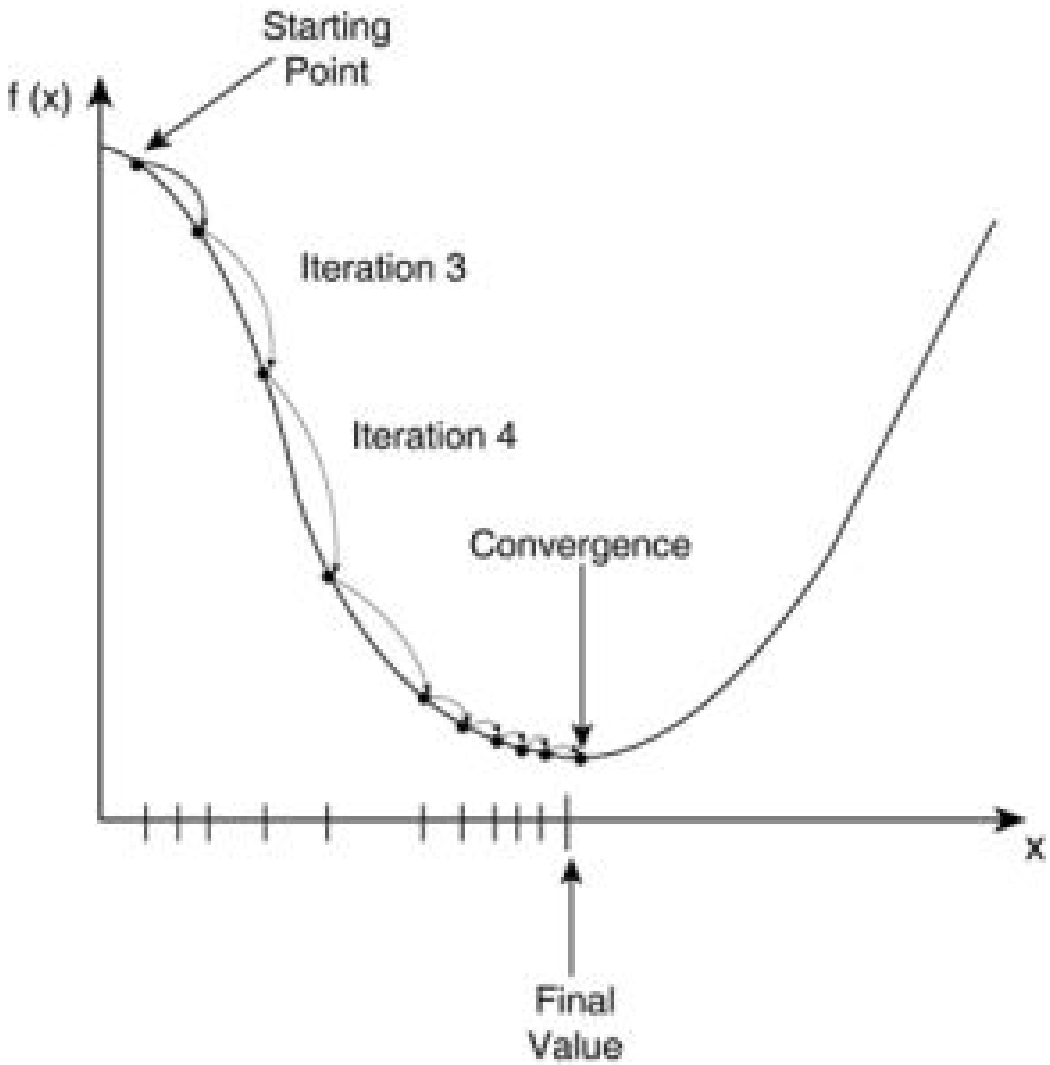
Gradient descent: main idea



- If a3 ↗□, our loss function ↘□

- Let's INCREASE a3!

Gradient descent in 1D



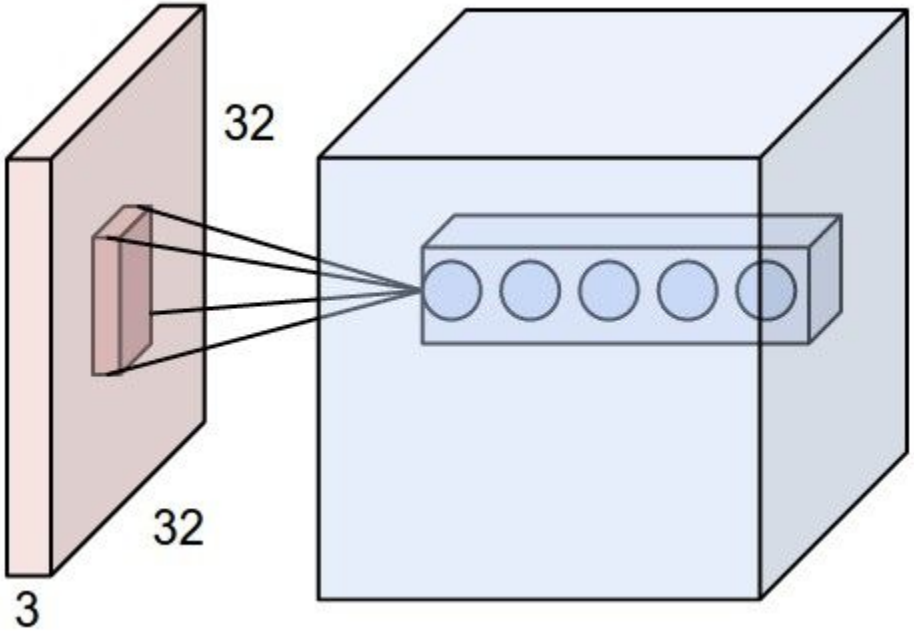
Practical Gradient Descent

- Stochastic Gradient Descent
- Learning rate α

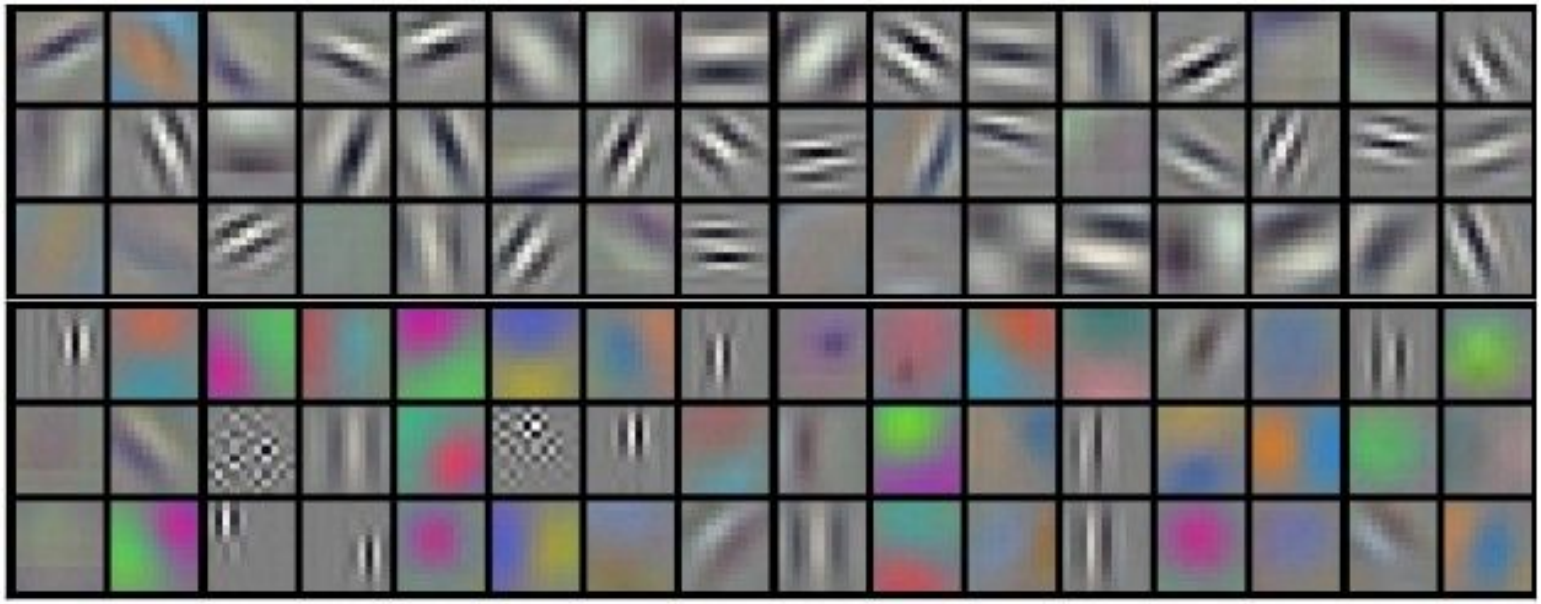
3.

Images

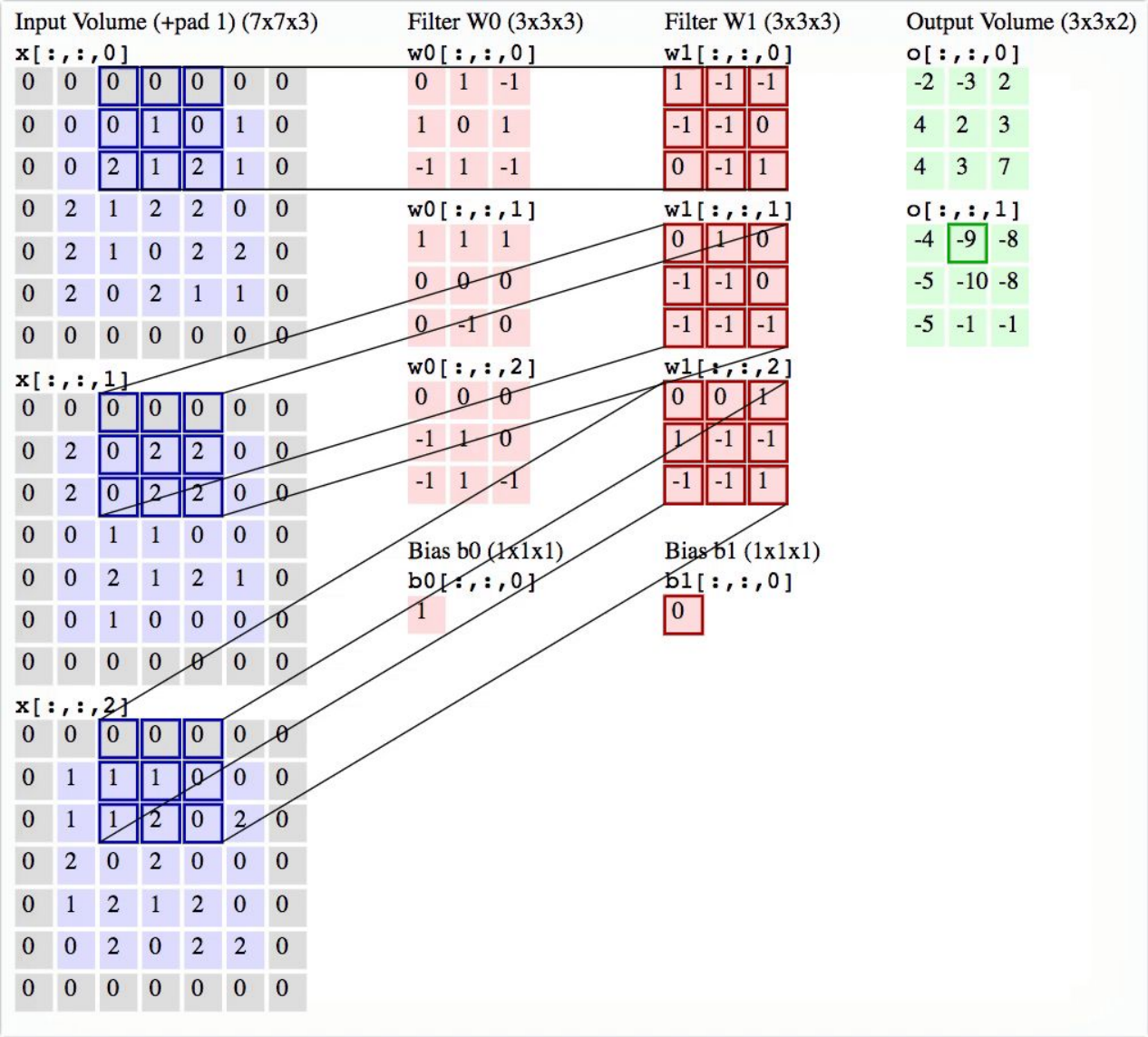
2D convolution



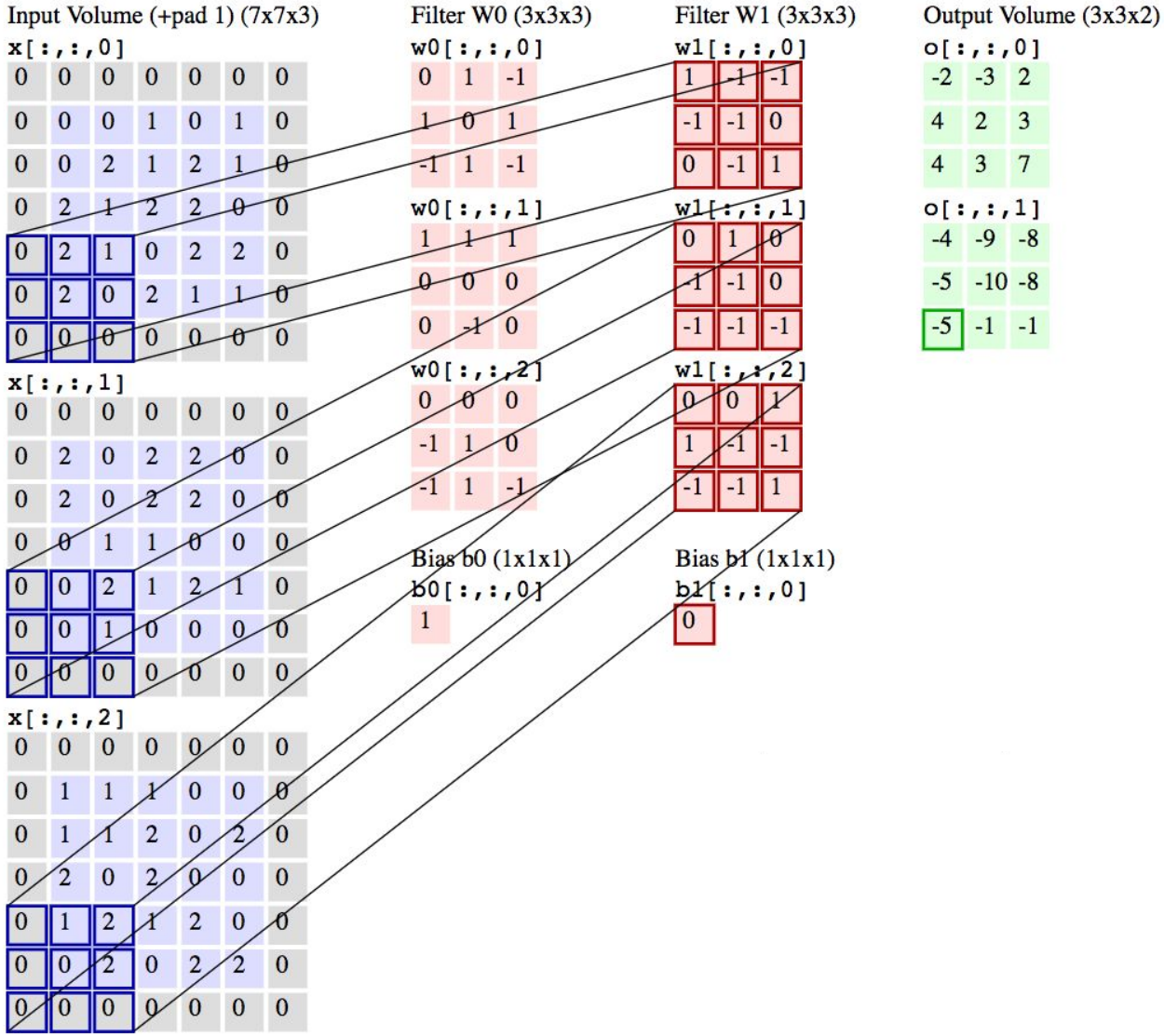
Example of filters



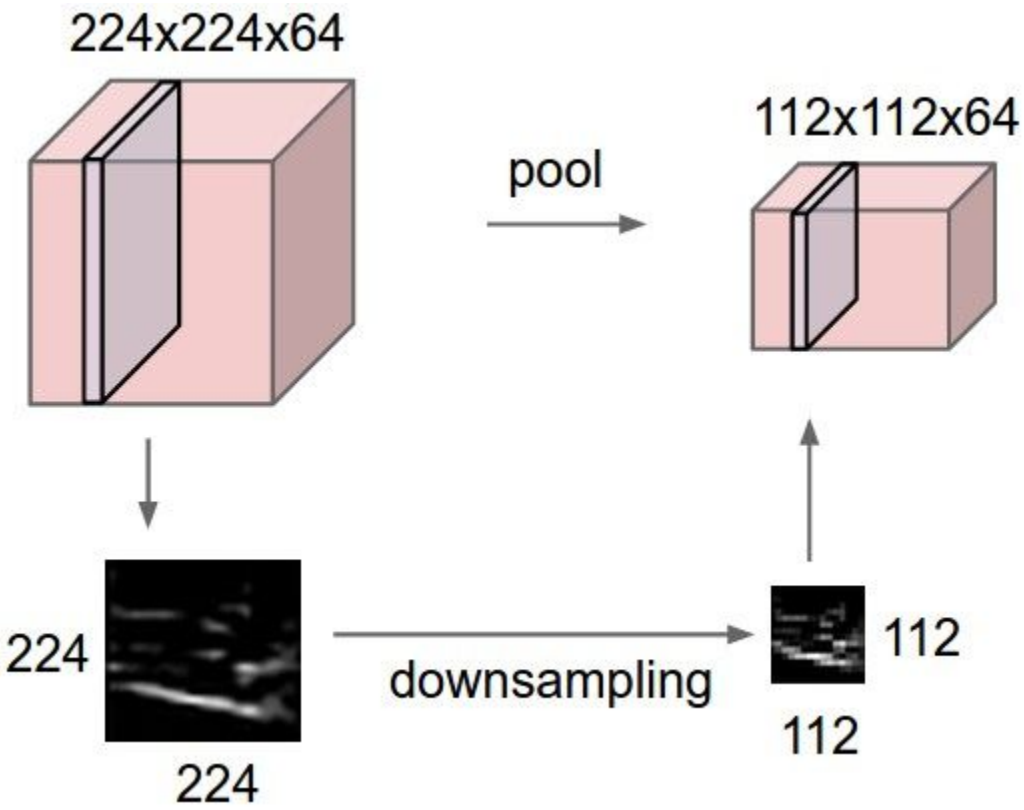
Animated example



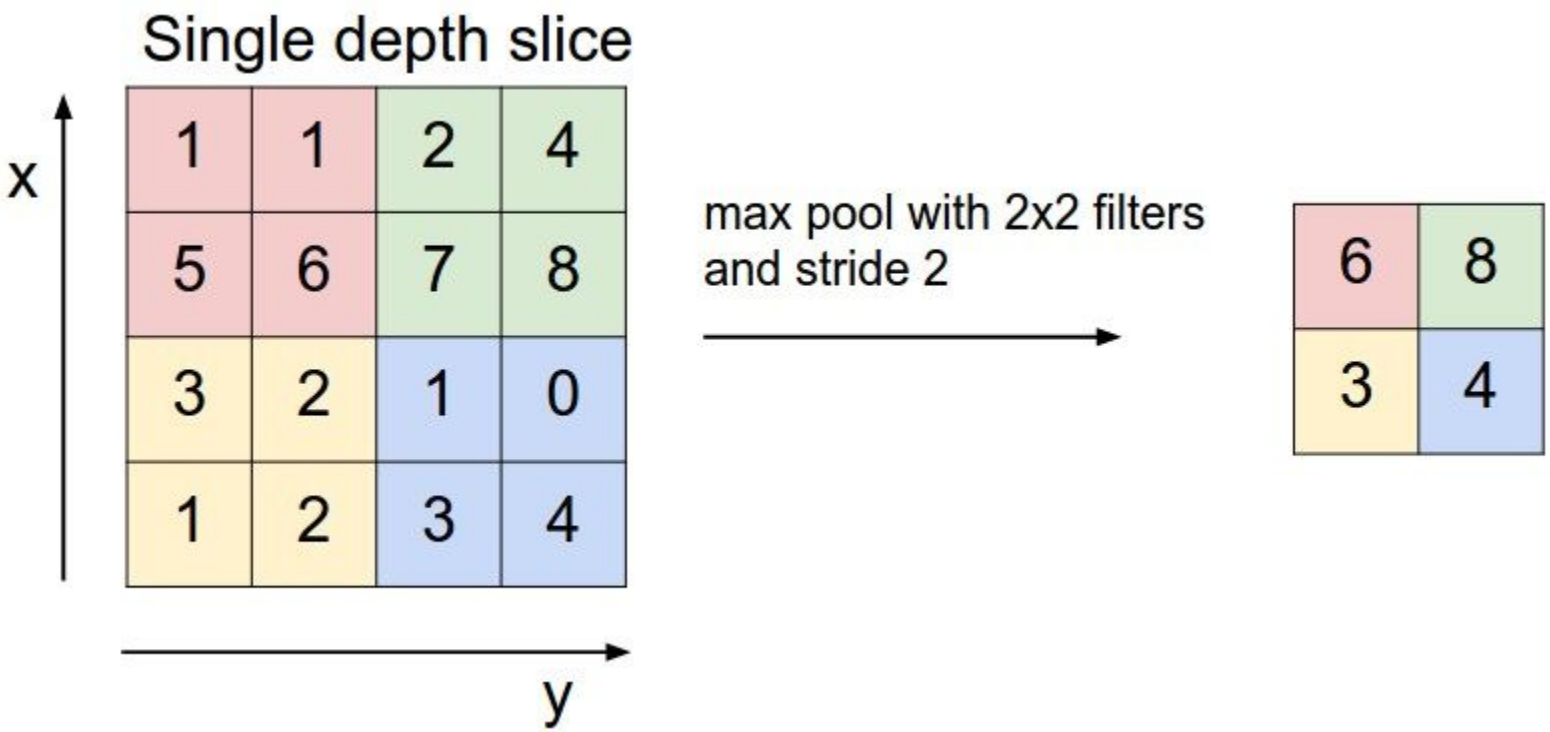
Animated example



Pooling



Max pooling: example

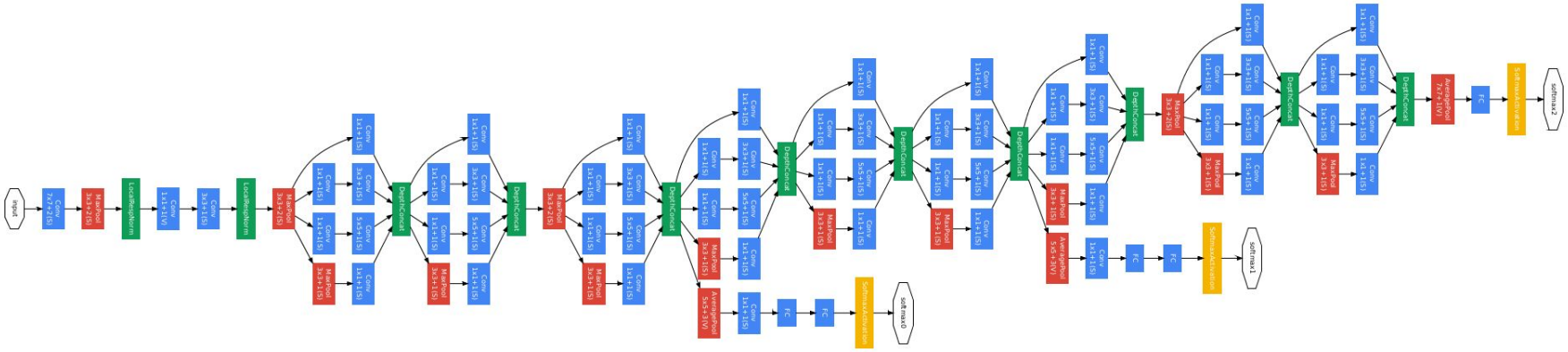


Example: VGG 16

```
INPUT: [224x224x3]        memory:  224*224*3=150K   weights: 0
CONV3-64: [224x224x64]  memory:  224*224*64=3.2M   weights: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64]  memory:  224*224*64=3.2M   weights: (3*3*64)*64 = 36,864
POOL2: [112x112x64]  memory:  112*112*64=800K   weights: 0

CONV3-128: [112x112x128]  memory:  112*112*128=1.6M   weights: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128]  memory:  112*112*128=1.6M   weights: (3*3*128)*128 = 147,456
POOL2: [56x56x128]  memory:  56*56*128=400K   weights: 0

CONV3-256: [56x56x256]  memory:  56*56*256=800K   weights: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256]  memory:  56*56*256=800K   weights: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256]  memory:  56*56*256=800K   weights: (3*3*256)*256 = 589,824
POOL2: [28x28x256]  memory:  28*28*256=200K   weights: 0

CONV3-512: [28x28x512]  memory:  28*28*512=400K   weights: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512]  memory:  28*28*512=400K   weights: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512]  memory:  28*28*512=400K   weights: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512]  memory:  14*14*512=100K   weights: 0

CONV3-512: [14x14x512]  memory:  14*14*512=100K   weights: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512]  memory:  14*14*512=100K   weights: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512]  memory:  14*14*512=100K   weights: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512]  memory:  7*7*512=25K  weights: 0

FC: [1x1x4096]  memory:  4096  weights: 7*7*512*4096 = 102,760,448
FC: [1x1x4096]  memory:  4096  weights: 4096*4096 = 16,777,216
FC: [1x1x1000]  memory:  1000 weights: 4096*1000 = 4,096,000
```

TOTAL memory: 24M * 4 bytes ~= 93MB / image (only forward! ~*2 for bwd)
TOTAL params: 138M parameters

Example: GoogleNet



Convolution
Pooling
Softmax
Other

Residual networks

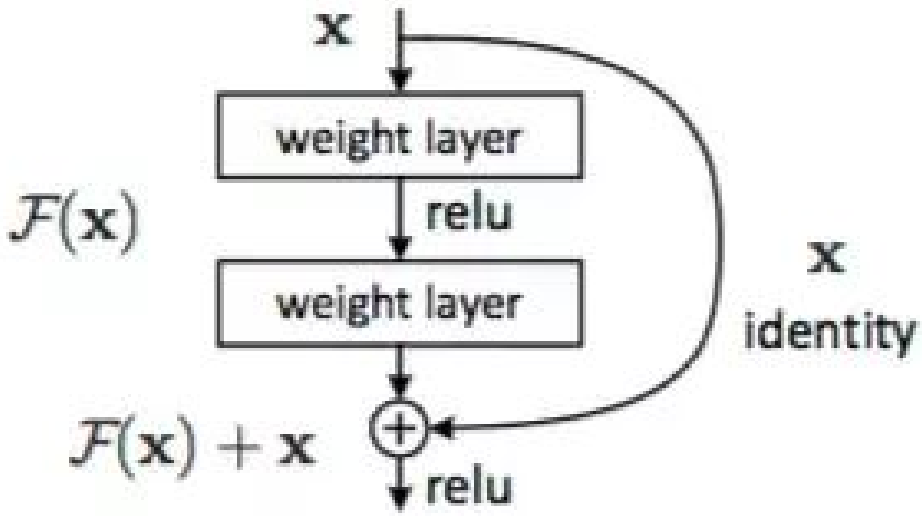


Figure 2. Residual learning: a building block.

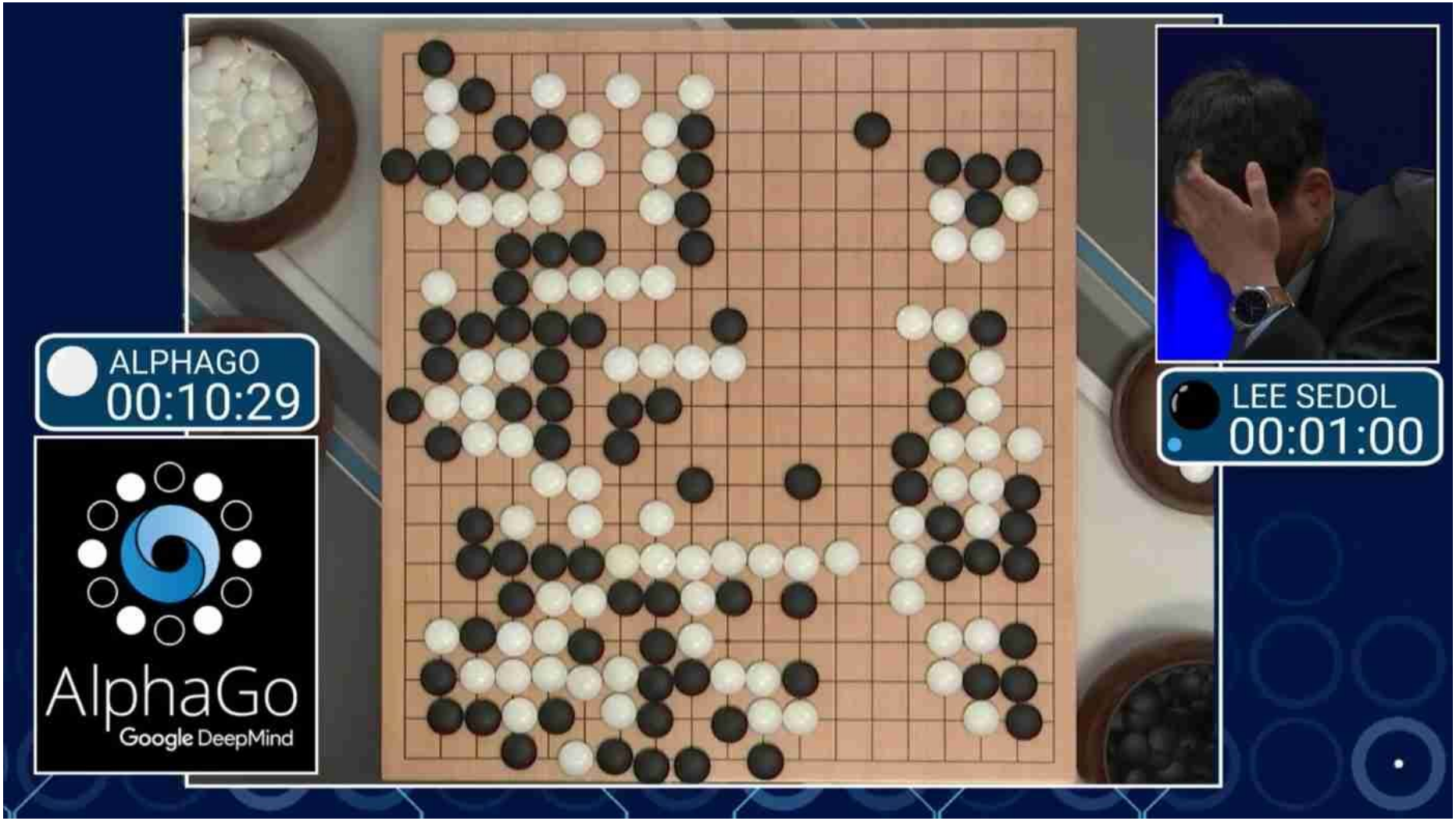
Batch Normalisation

- Main idea: make the output of the layers more stable

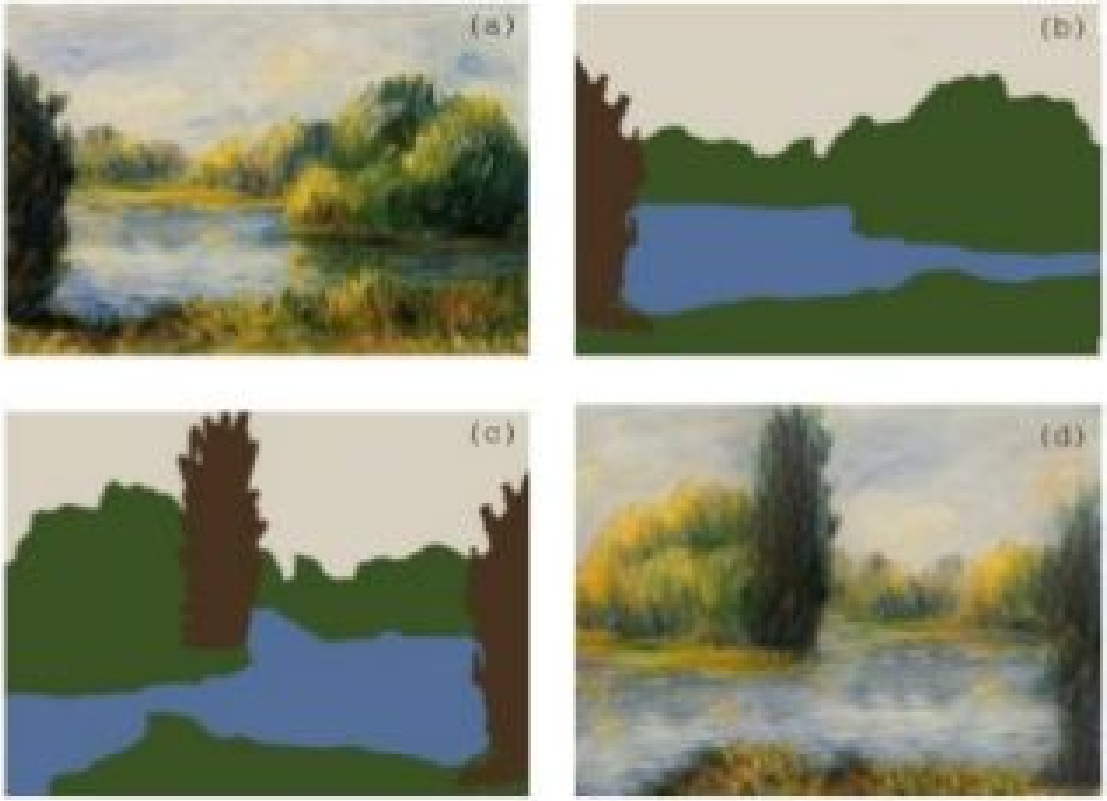
- Wy:
  - Limit Internal Covariate shift
  - Limit vanishing gradient

- Very good regularizer of CNN

Application: AlphaGo



Application: drawing paintings



(a) Original painting by Renoir, (b) semantic annotations, (c) desired layout, (d) generated output.

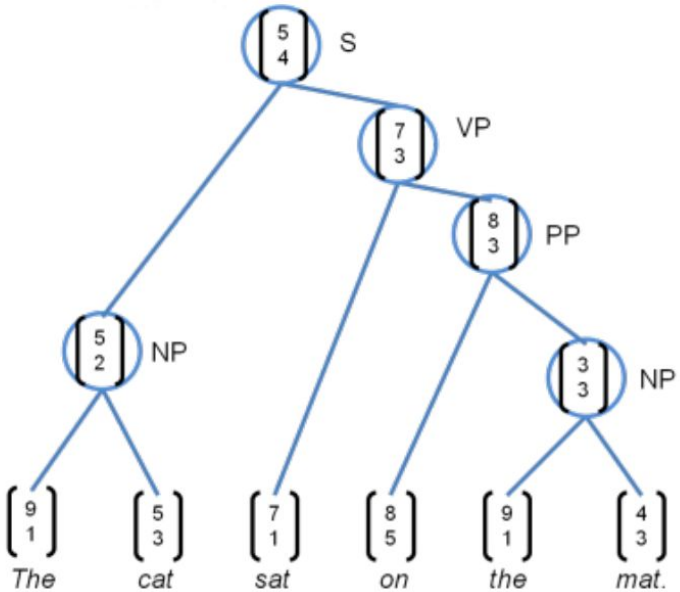
http://arxiv.org/abs/1603.01768 Semantic Style Transfer and Turning Two-Bit Doodles into Fine Artworks

4.

Text

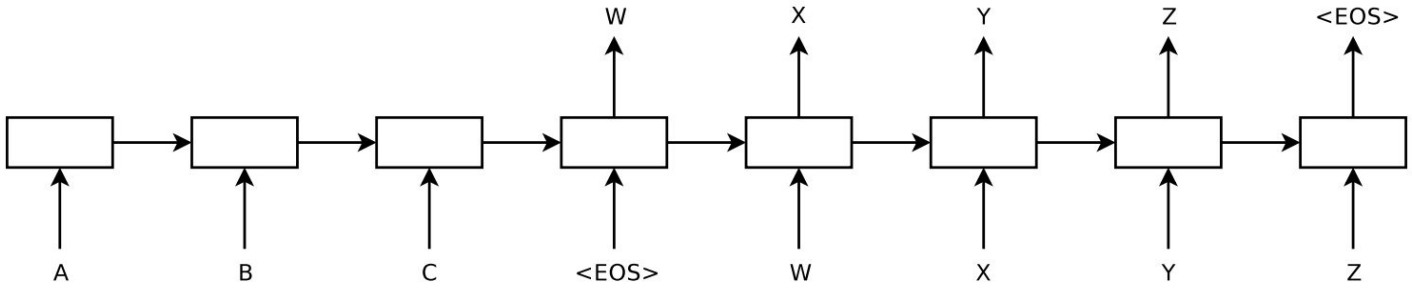
Text : recurrent

- DL:
  - Every word and every phrase is a vector
  - a neural network combines two vectors into one vector
  - Socher et al. 2011



Machine Translation

- Source sentence mapped to vector, then output sentence generated.



- Sequence to Sequence Learning with Neural Networks by Sutskever et al. 2014; Luong et al. 2016

- About to replace very complex hand engineered architectures

One Hot Encoding

The vast majority of rule-based **and** statistical NLP work regards words as atomic symbols: hotel, conference, walk

In vector space terms, this is a vector with one 1 and a lot of zeroes

[0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]

Dimensionality: 20K (speech) – 50K (PTB) – 500K (big vocab) – 13M (Google 1T)

We call this a "one-hot" representation. Its problem:

motel [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0] AND
hotel [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0] = 0

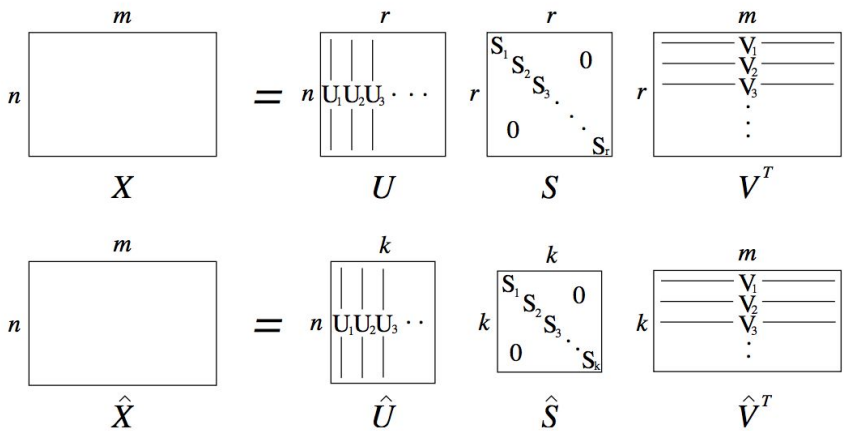
Co-occurrence matrix

- I like deep learning.
- I like NLP.
- I enjoy flying.

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

Word vectors

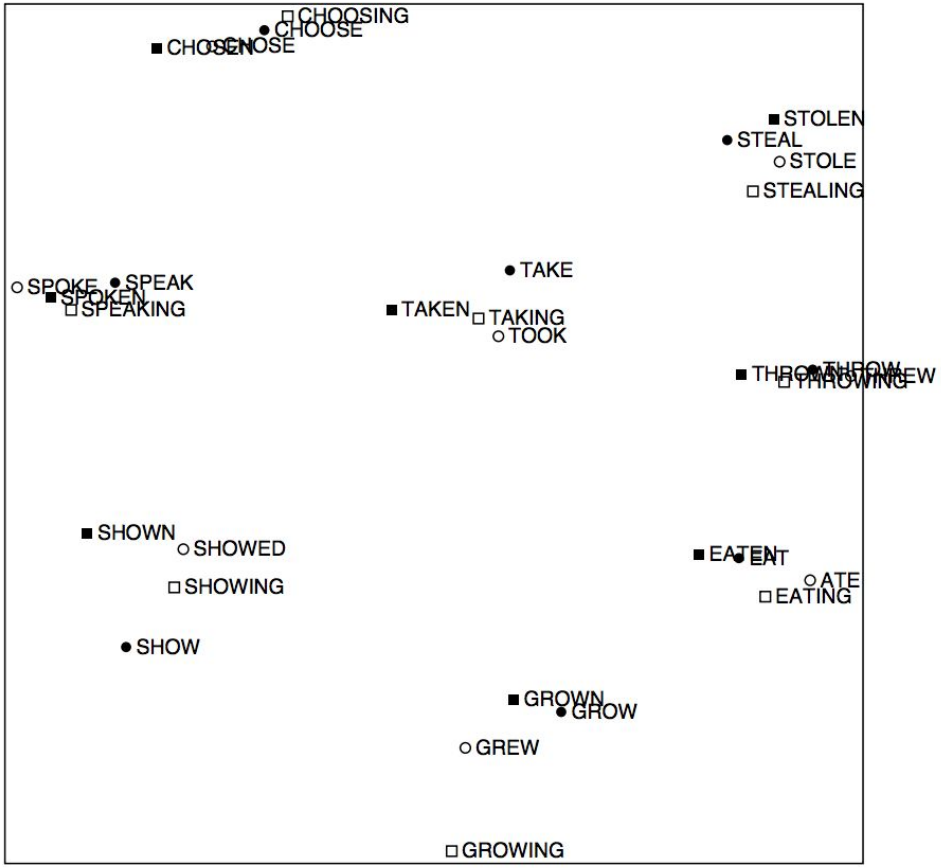
Singular Value Decomposition of cooccurrence matrix *X*.



$\hat{X}$ is the best rank *k* approximation to *X* , in terms of least squares.

- Idea: store "most" of the important information in a fixed, small number of dimensions: a dense vector
- Usually around 25 – 1000 dimensions

Word vectors similarity



An Improved Model of Semantic Similarity Based on Lexical Co-Occurrence
Rohde et al. 2005

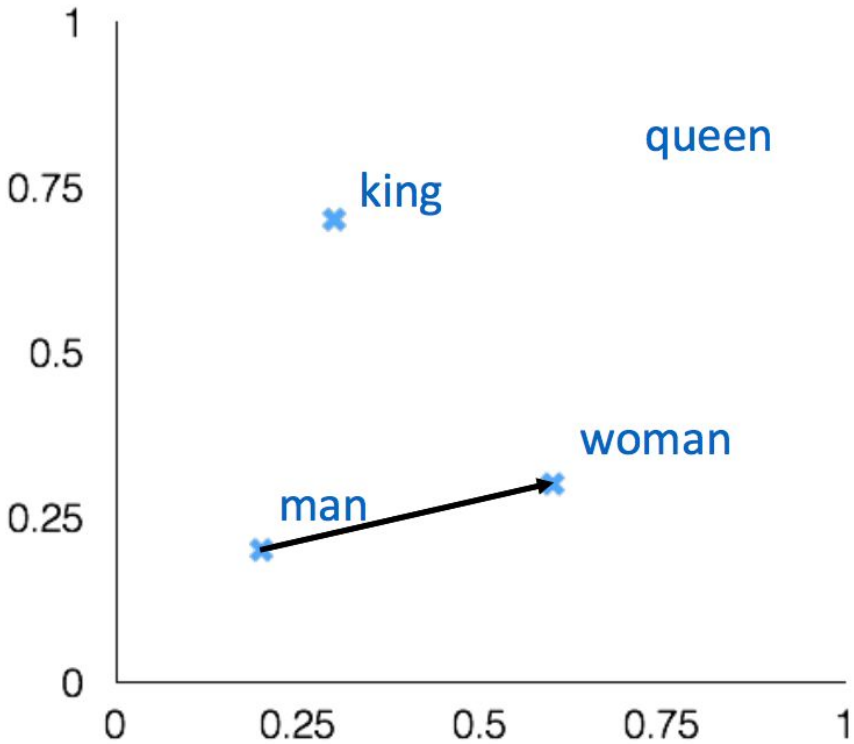
Word vectors arithmetic

a:b :: c:?

$$d = \arg\max_x \frac{(w_b - w_a + w_c)^T w_x}{||w_b - w_a + w_c||}$$

man:woman :: king:?

+	king	[0.30 0.70]
-	man	[0.20 0.20]
+	woman	[0.60 0.30]
	queen	[0.70 0.80]



Word vectors in practice

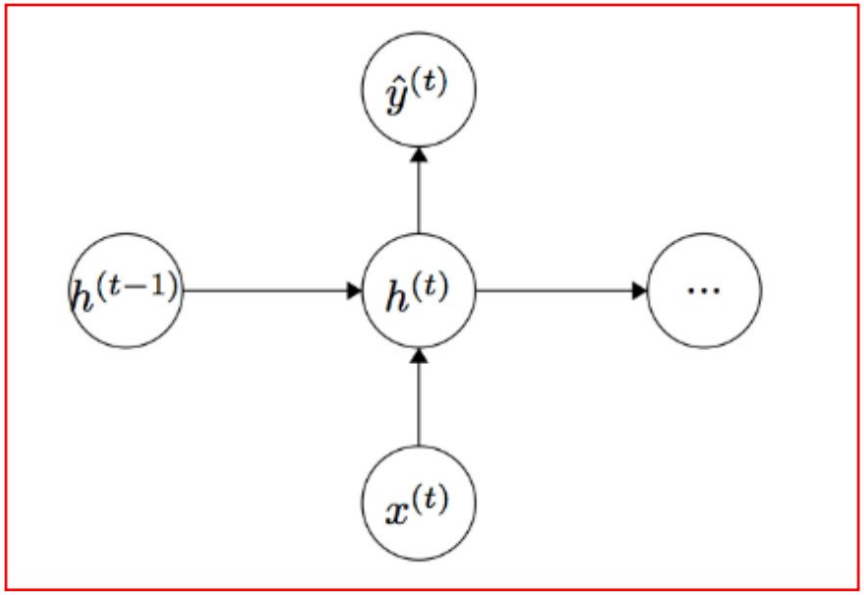
- Use pretrained Glove or Word2Vec
- If your dataset is large, it may be better to retrain the vectors

RNN : basics

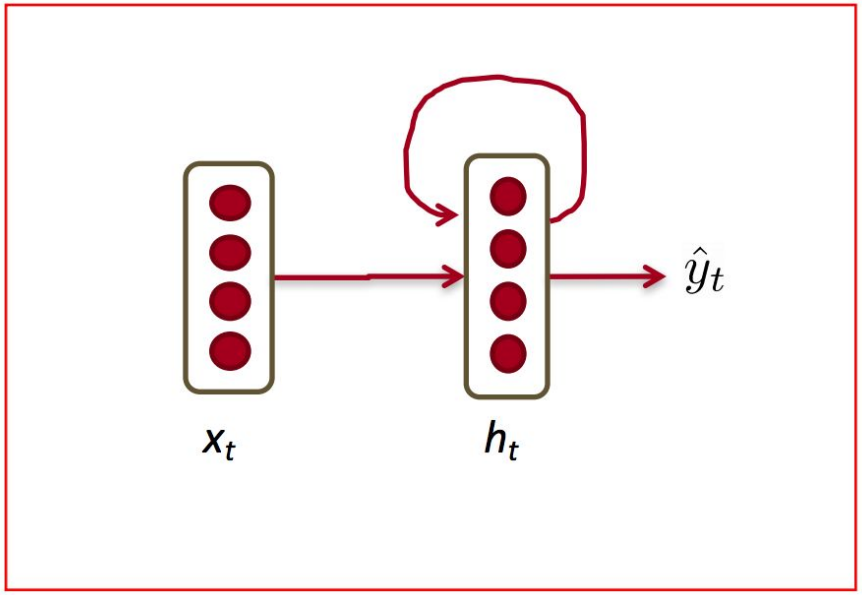
Given list of word **vectors:** $x_1, \ldots, x_{t-1}, x_t, x_{t+1}, \ldots, x_T$

At a single time step:

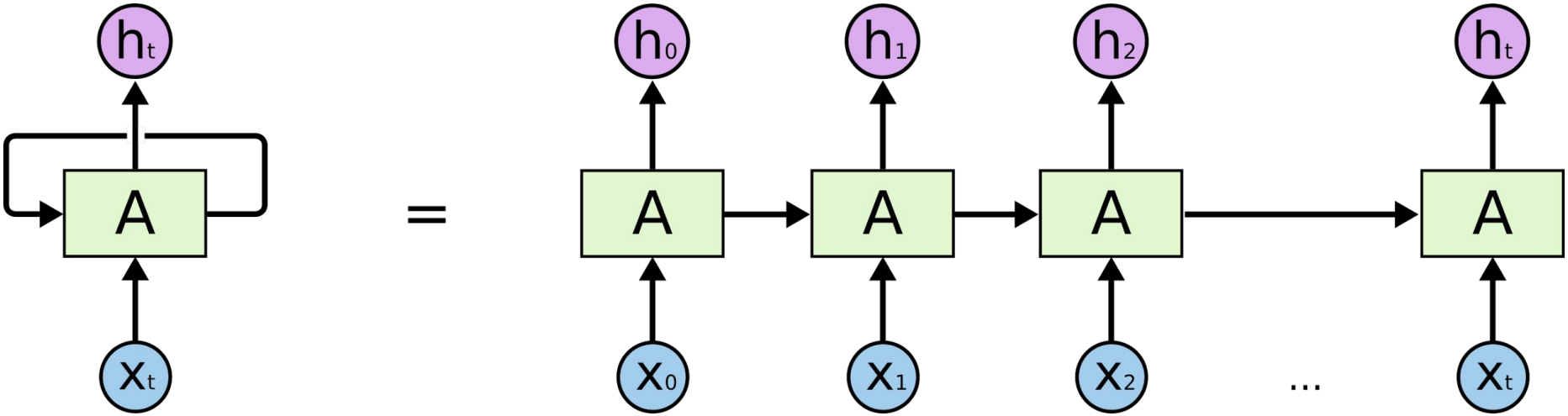
$$
\begin{aligned}
h_t &= \sigma\left(W^{(hh)} h_{t-1} + W^{(hx)} x_{[t]}\right) \\
\hat{y}_t &= \text{softmax}\left(W^{(S)} h_t\right) \\
\hat{P}(x_{t+1} = v_j \mid x_t, \ldots, x_1) &= \hat{y}_{t,j}
\end{aligned}
$$



←→

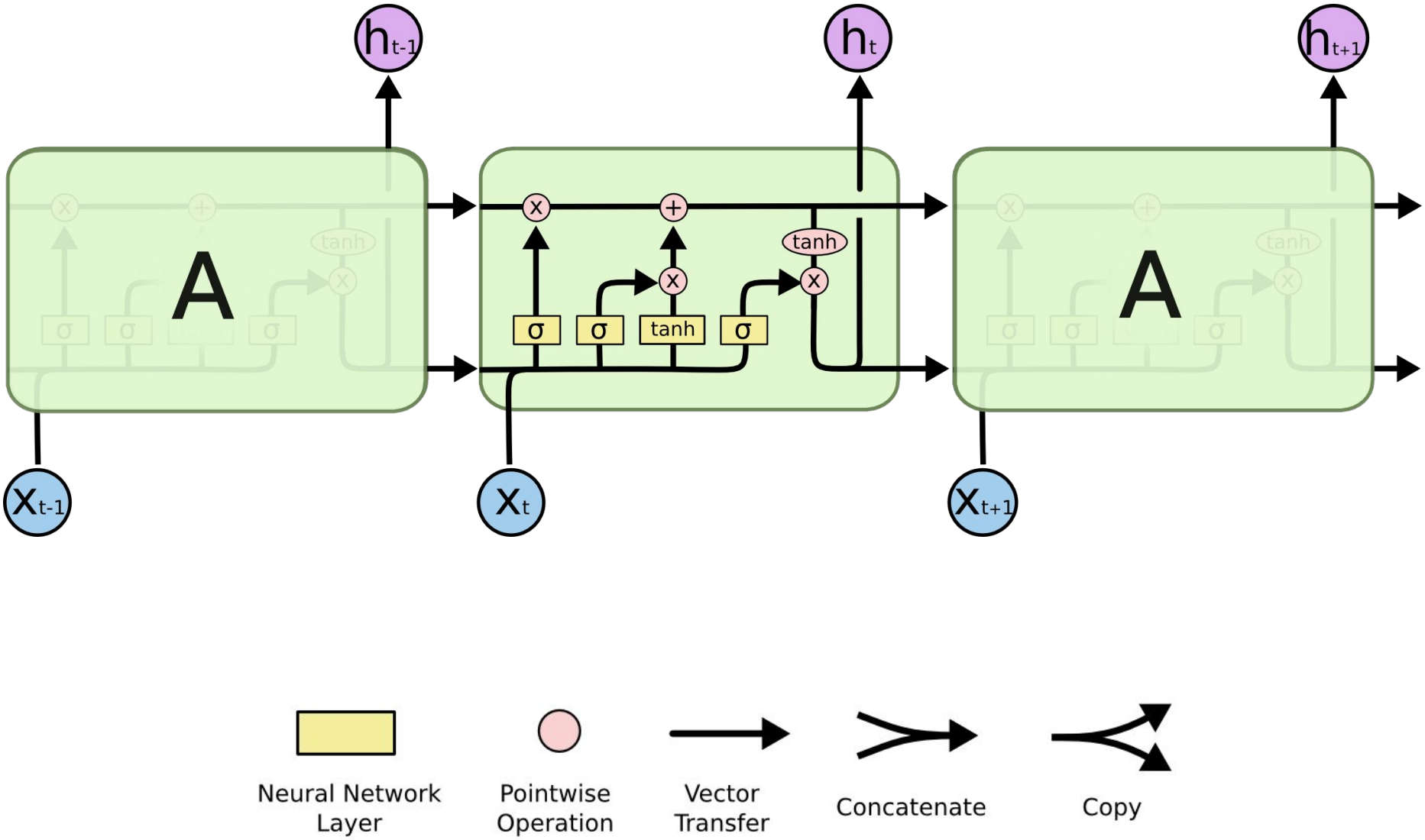


RNN : loop unrolling

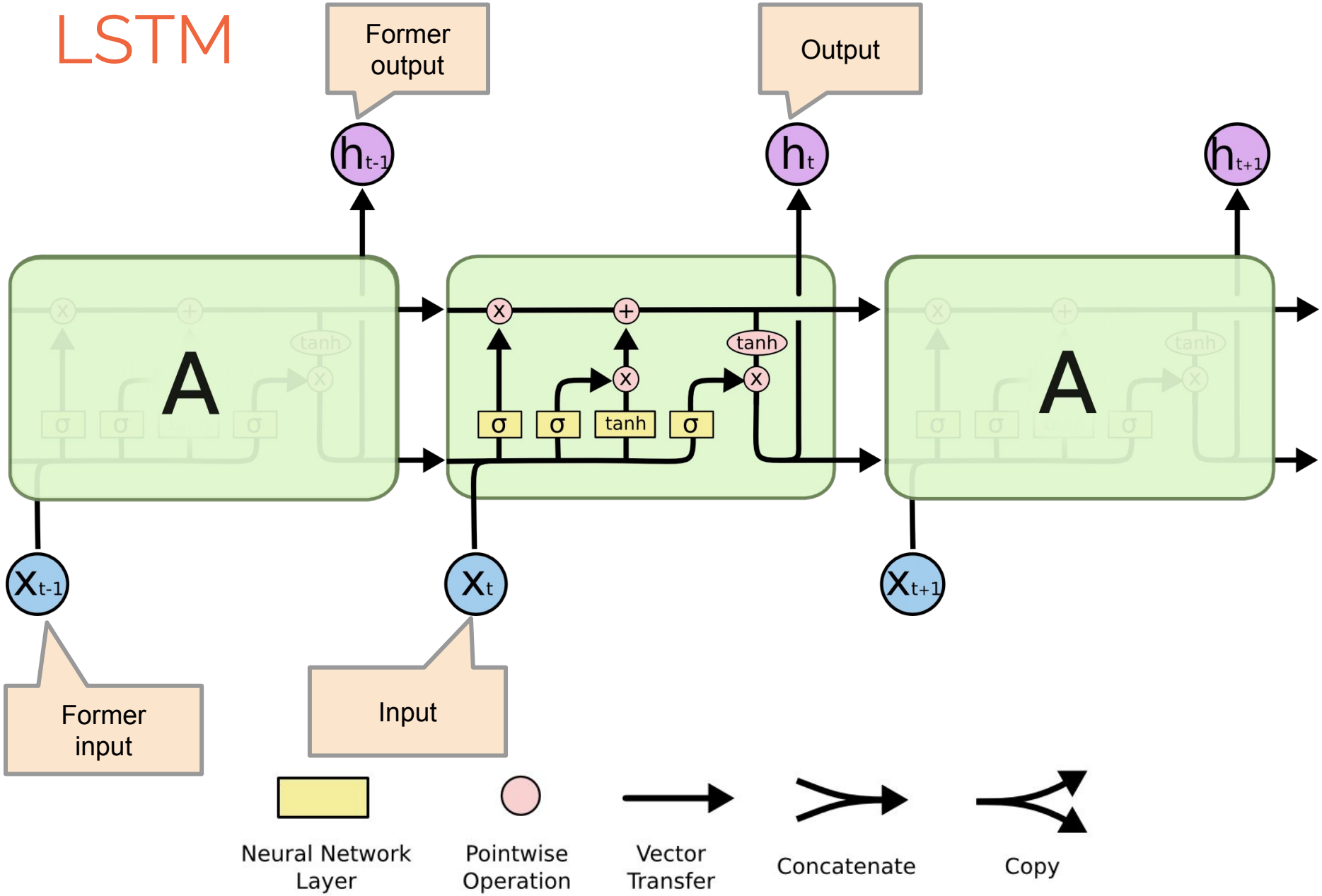


RNN : limit

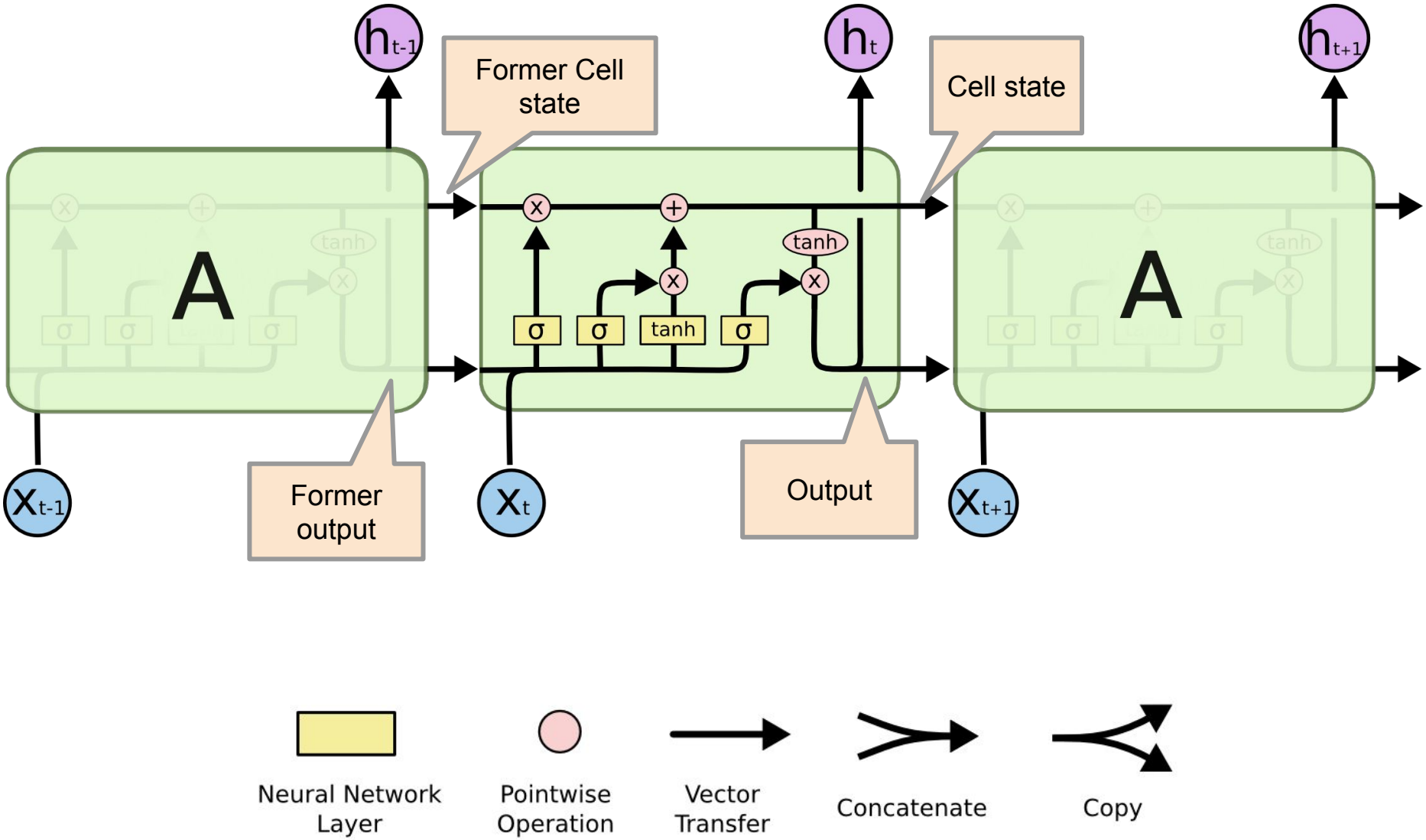
LSTM


h_{t-1}
h_t
h_{t+1}
A
A
tanh
σ
σ
tanh
σ
X_{t-1}
X_t
X_{t+1}
Neural Network Layer
Pointwise Operation
Vector Transfer
Concatenate
Copy

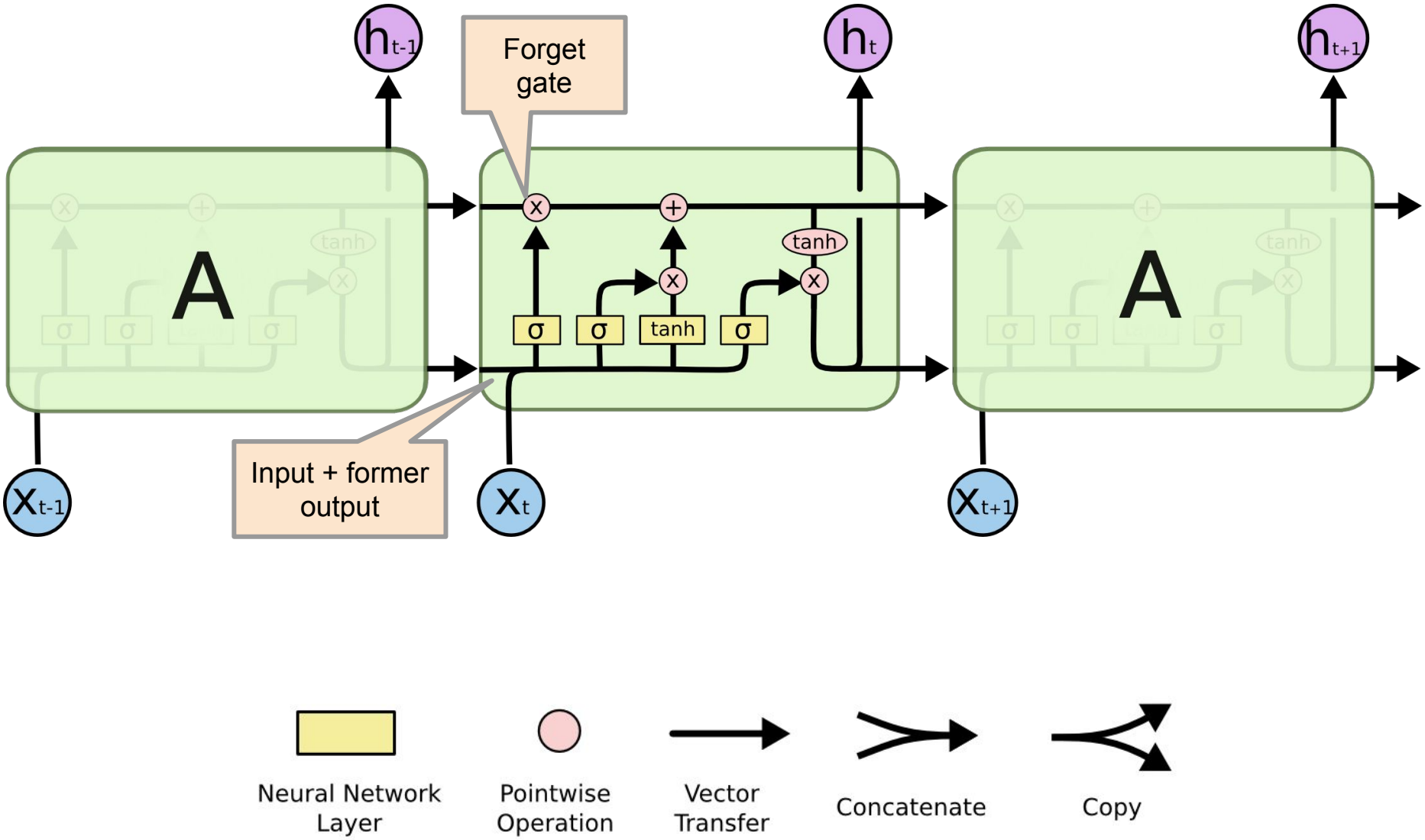
LSTM


Former output
Output
h_{t-1}
h_t
h_{t+1}
A
A
σ
σ
tanh
σ
tanh
X_{t-1}
X_t
X_{t+1}
Former input
Input
Neural Network Layer
Pointwise Operation
Vector Transfer
Concatenate
Copy

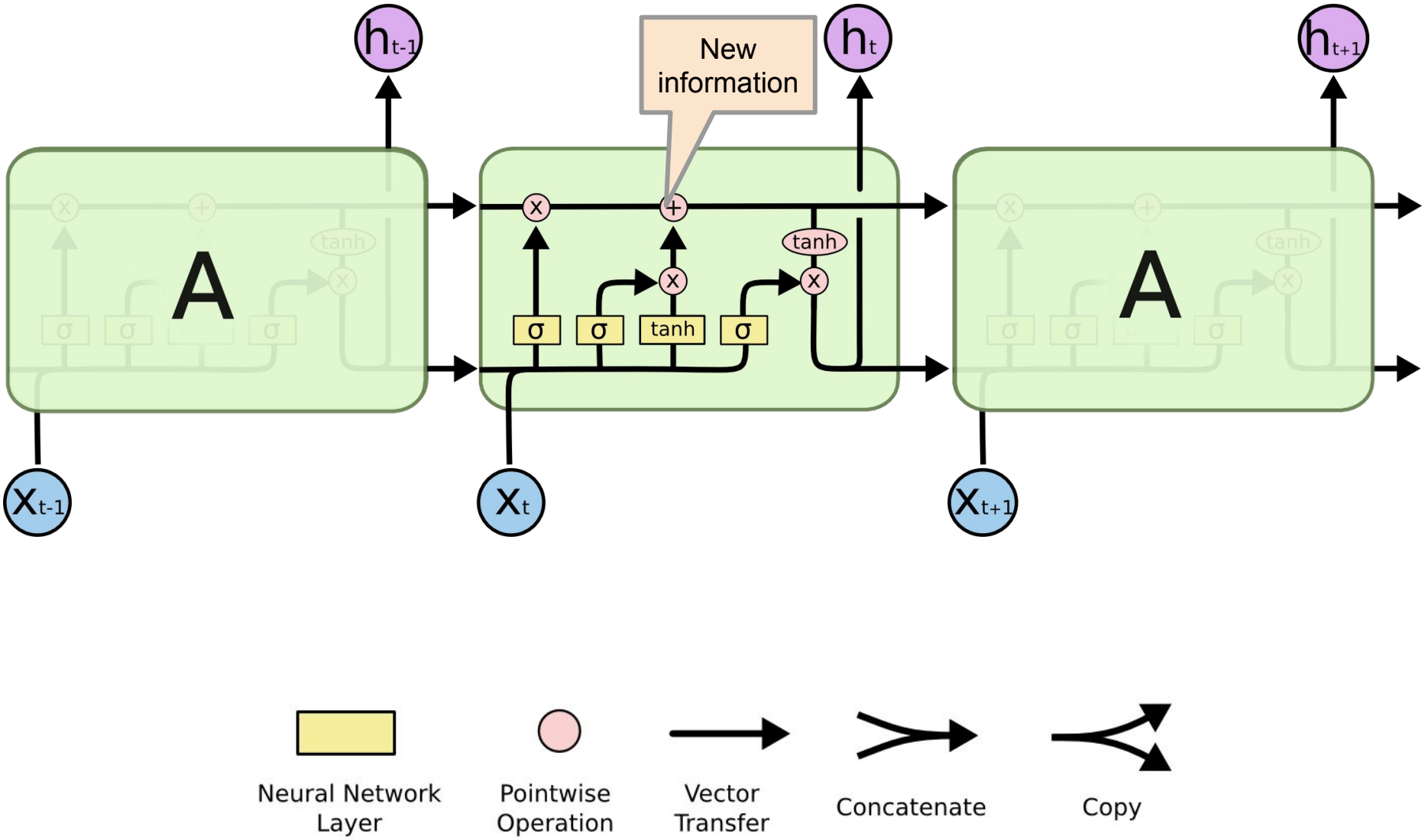
LSTM


h_{t-1}
Former Cell state
h_t
Cell state
h_{t+1}
A
A
σ
σ
tanh
σ
tanh
X_{t-1}
Former output
X_t
Output
X_{t+1}
Neural Network Layer
Pointwise Operation
Vector Transfer
Concatenate
Copy

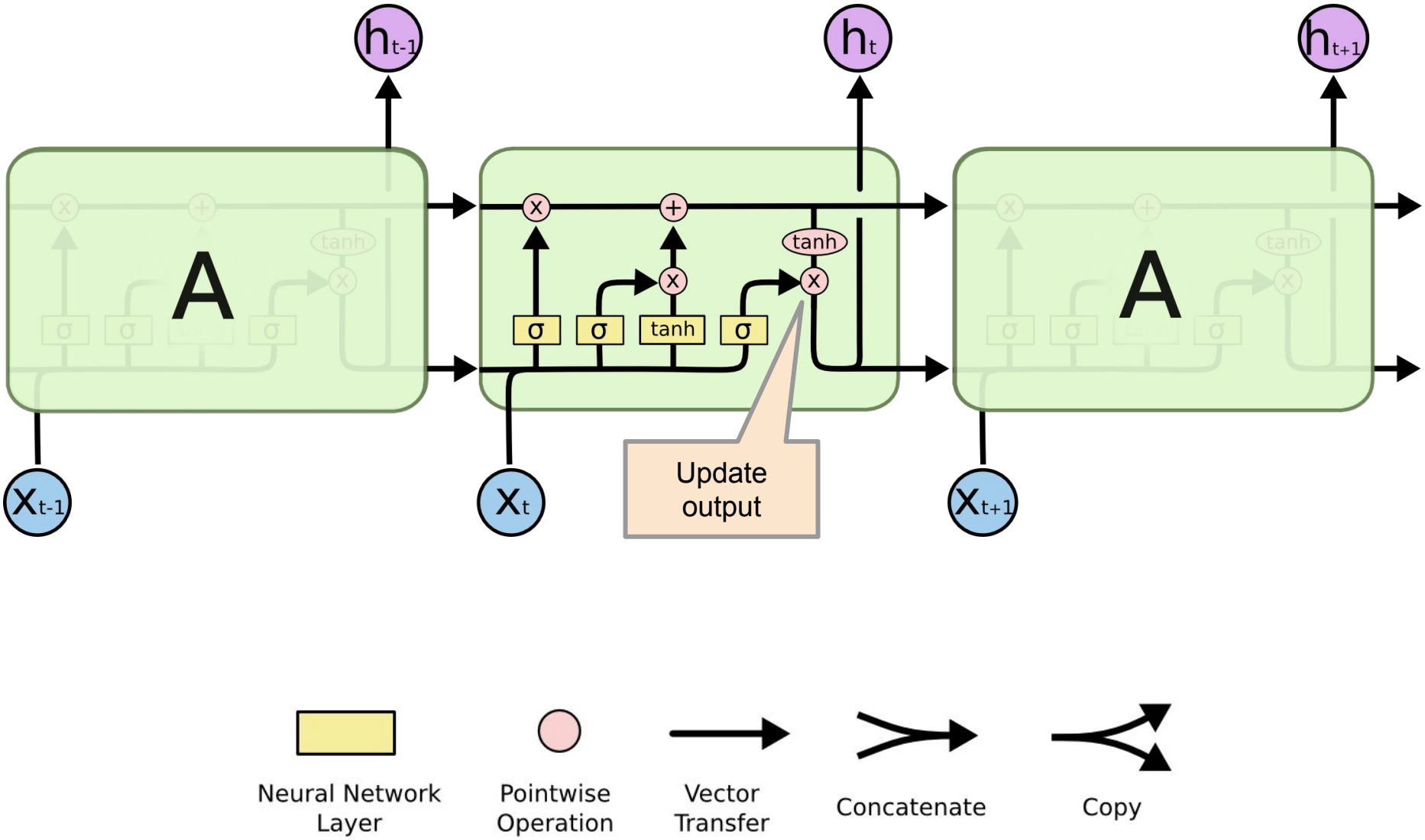
LSTM


h_{t-1}
Forget gate
h_t
h_{t+1}
A
tanh
σ
σ
tanh
σ
A
Input + former output
X_{t-1}
X_t
X_{t+1}
Neural Network Layer
Pointwise Operation
Vector Transfer
Concatenate
Copy

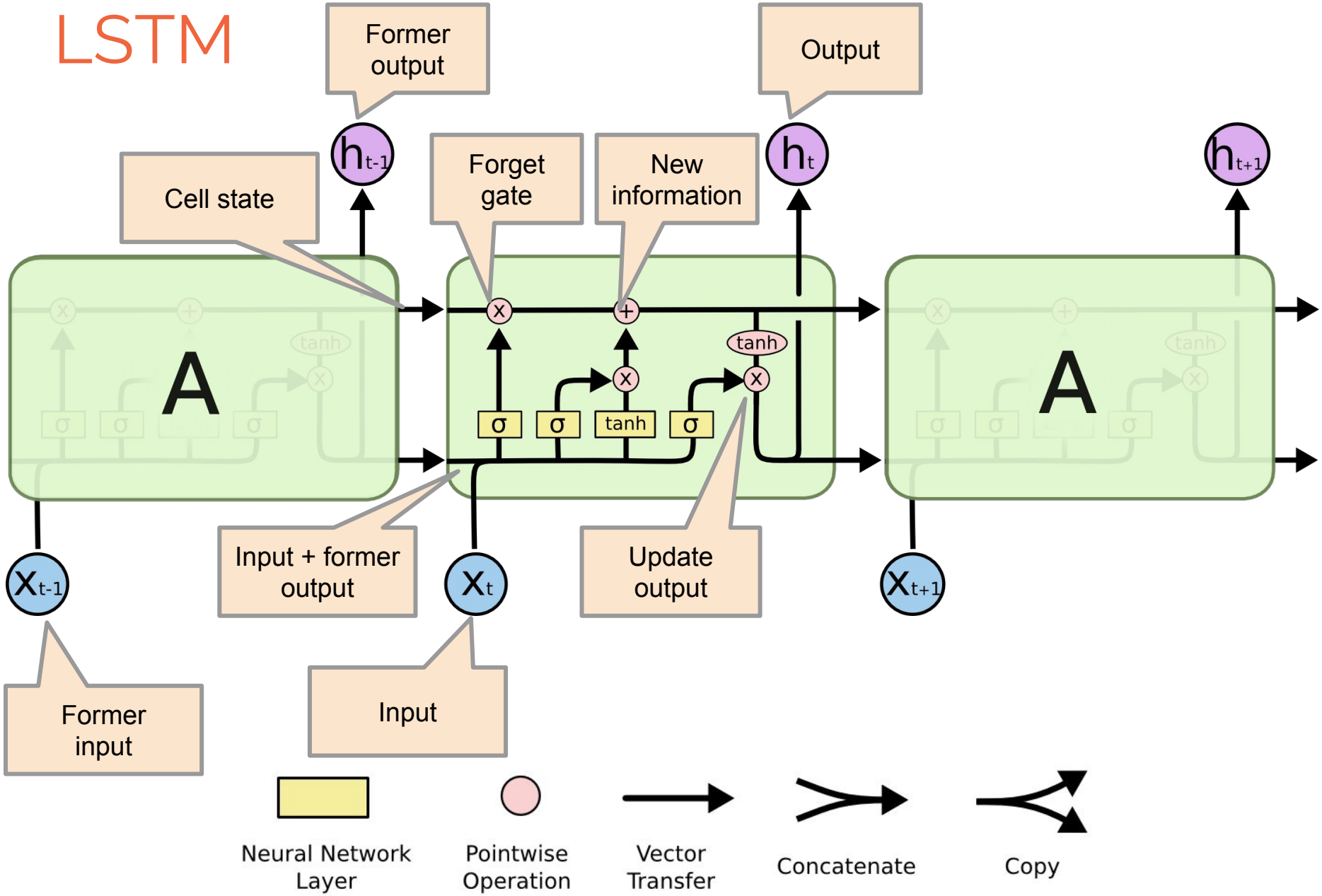
LSTM


h_{t-1}
New information
h_t
h_{t+1}
A
A
tanh
σ
σ
tanh
σ
X_{t-1}
X_t
X_{t+1}
Neural Network Layer
Pointwise Operation
Vector Transfer
Concatenate
Copy

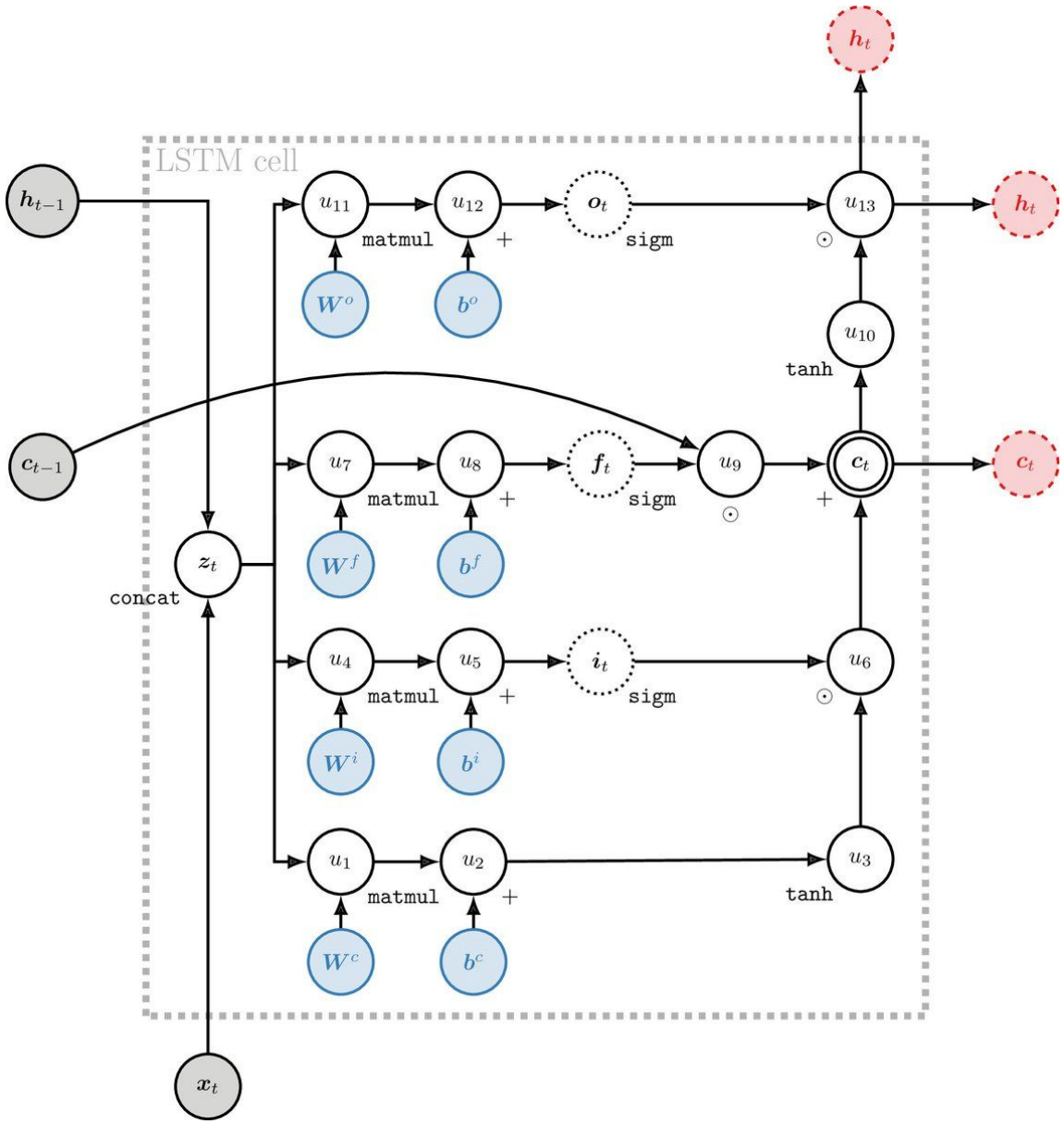
LSTM


h_{t-1}
h_t
h_{t+1}
A
A
tanh
σ
σ
tanh
σ
Update output
X_{t-1}
X_t
X_{t+1}
Neural Network Layer
Pointwise Operation
Vector Transfer
Concatenate
Copy

LSTM


Former output
Output
Cell state
Forget gate
New information
h_{t-1}
h_t
h_{t+1}
A
A
tanh
σ
σ
tanh
σ
Input + former output
Update output
X_{t-1}
X_t
X_{t+1}
Former input
Input
Neural Network Layer
Pointwise Operation
Vector Transfer
Concatenate
Copy

LSTM


LSTM cell
h_{t-1}
c_{t-1}
x_t
z_t
concat
u_{11}
u_{12}
o_t
u_{13}
matmul
+
sigm
W^o
b^o
u_{10}
tanh
u_7
u_8
f_t
u_9
c_t
W^f
b^f
u_4
u_5
i_t
u_6
W^i
b^i
u_1
u_2
u_3
W^c
b^c
h_t
c_t

LSTM in practice

- Vanishing Gradient is still an issue
  - Clipping the gradients
  - Initialization + RELU

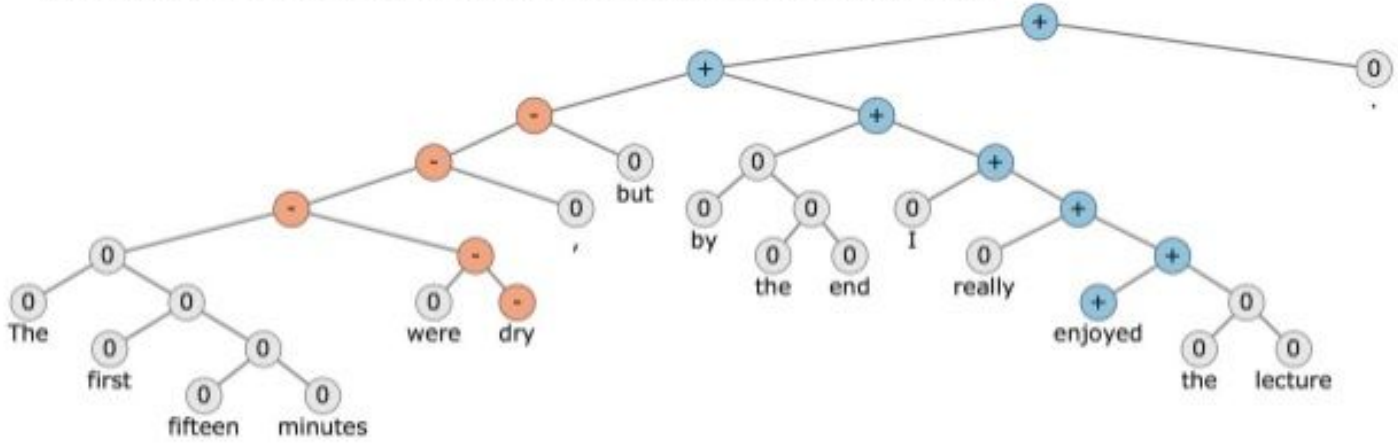
- Bidirectional RNN

- Many variant of LSTM

Application: Sentiment Analysis

Can capture complex cases where bag-of-words models fail.

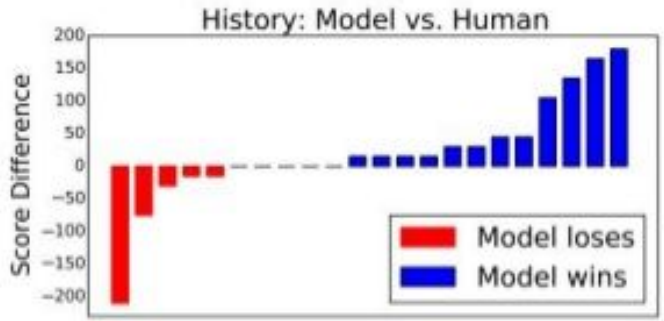
"This movie was actually neither that funny, nor super witty."

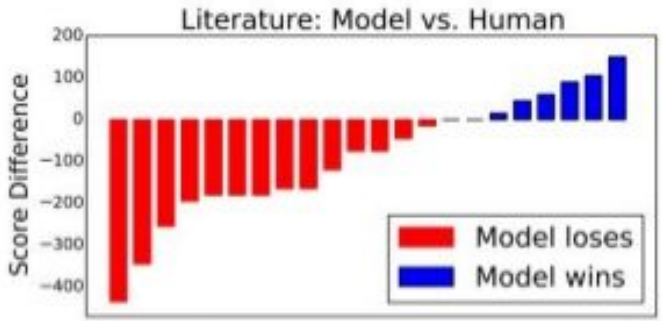


http://nlp.stanford.edu/sentiment/

Application: Question Answering

QUESTION:
He left unfinished a novel whose title character forges his father's signature to get out of school and avoids the draft by feigning desire to join. A more famous work by this author tells of the rise and fall of the composer Adrian Leverkühn. Another of his novels features the jesuit Naptha and his opponent Settembrini, while his most famous work depicts the aging writer Gustav von Aschenbach. Name this German author of The Magic Mountain and Death in Venice.
ANSWER: Thomas Mann





A Neural Network for Factoid Question Answering over Paragraphs, https://cs.umd.edu/~miyyer/qblearn/

Application: Image captioning

Human: "A group of men playing Frisbee in the park."
Computer model: "A group of young people playing a game of Frisbee."

5.
Generative Adversarial Networks

GAN and the variations that are now being proposed is the most interesting idea in the last 10 years in ML, in my opinion.

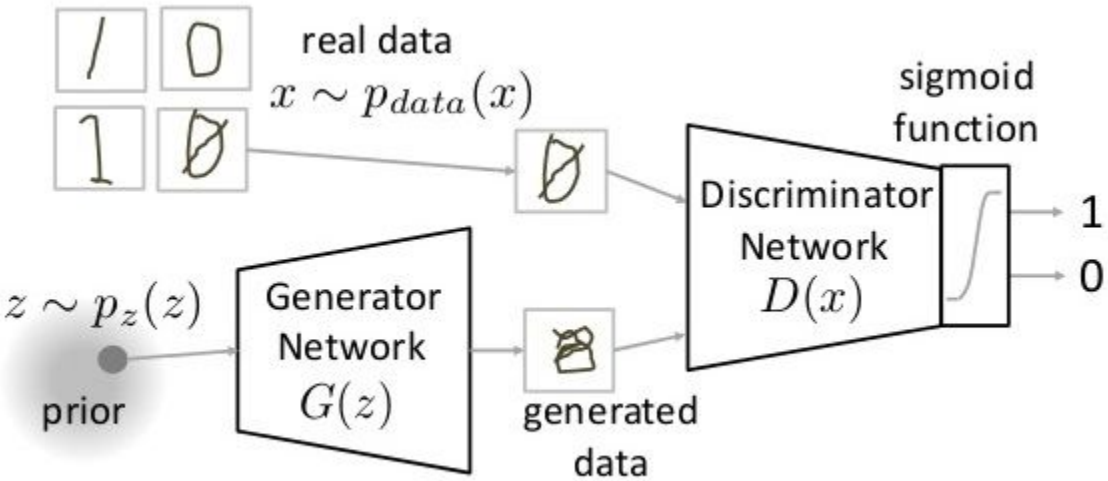
Yann Lecun

Generative models

- A generative model is a model for generating all values for a phenomenon
- Those that can be observed in the world
- And "target" variables that can only be computed from those observed.

Generative models: gaussian mixtures

Generative adversarial models



Generative adversarial models

$$\min_G \max_D V(D,G)$$

$$V(D,G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z))]$$

p_{data} (x) -> the distribution of real data

X -> sample from pdata(x)

P(z) -> distribution of generator

Z -> sample from p(z)

G(z) -> Generator Network

D(x) -> Discriminator Network

Why GAN are original

- **General procedure for creating a density estimator**
- **No pre-defined loss function**
- **Create parametrization of complicated real data manifold**

Application: Super-resolution

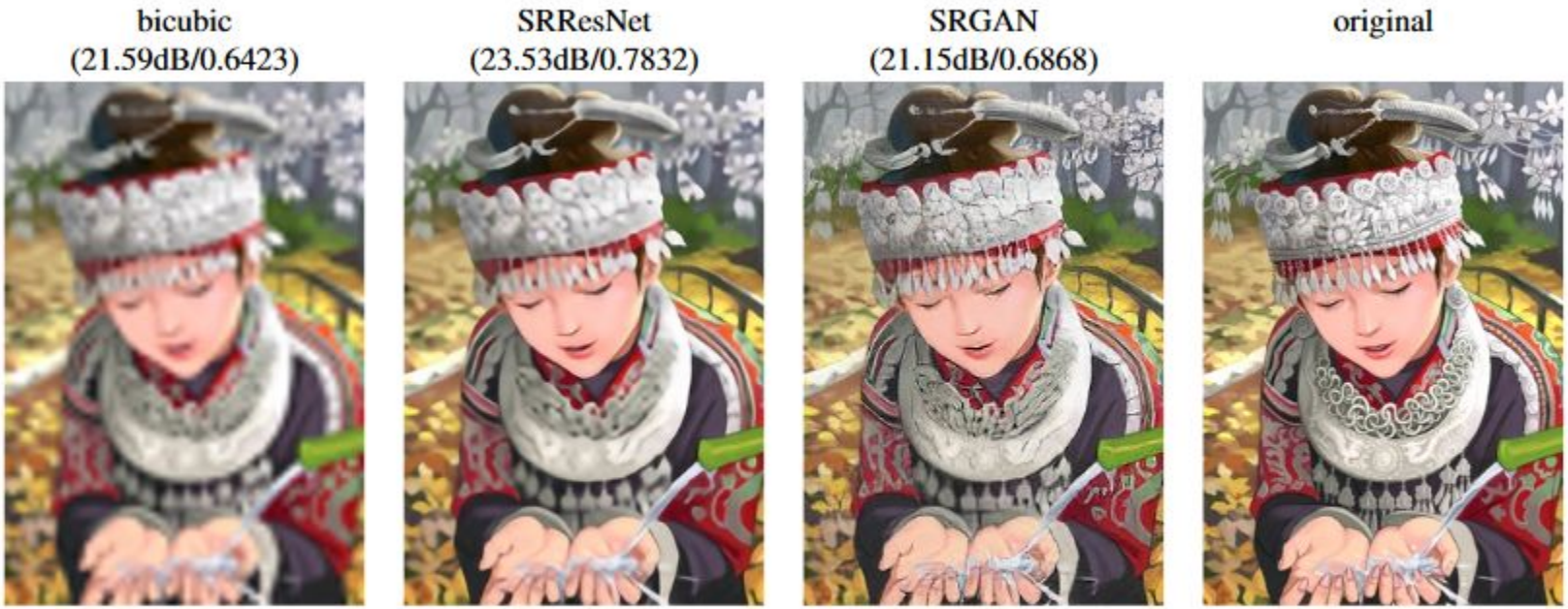
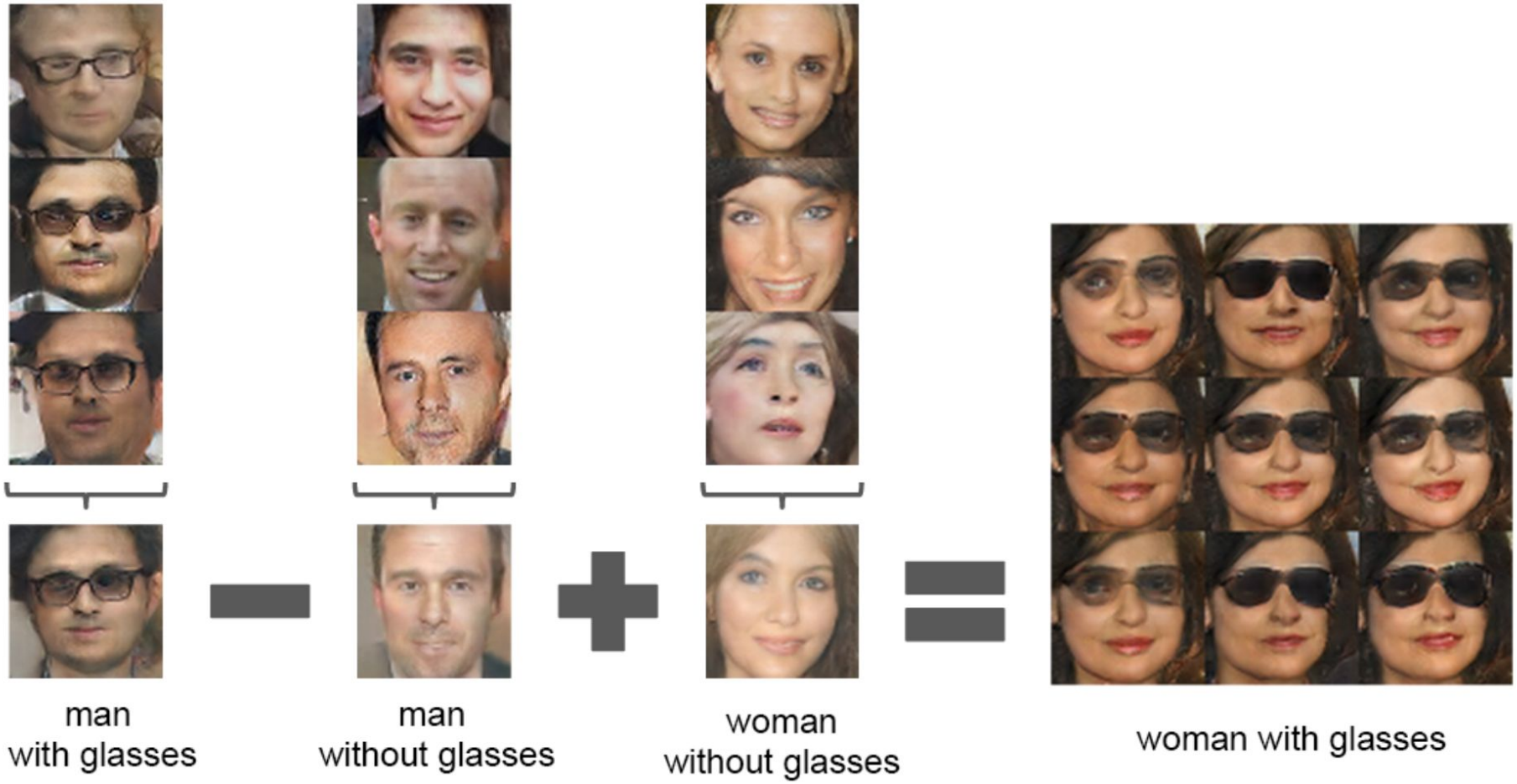


Figure 2: From left to right: bicubic interpolation, deep residual network optimized for MSE, deep residual generative adversarial network optimized for a loss more sensitive to human perception, original HR image. Corresponding PSNR and SSIM are shown in brackets. [4× upscaling]

Application: Face arithmetic



Application: Generating pokemons

6.

Practical Deep learning

We need help!

- It’s not possible to calculate the derivative of big networks 😶‍🌫️
- Software can do that for us
- Many possible frameworks: Tensorflow, Keras, Pytorch, Caffee, ...

Example of a 2 layers perceptron in Pytorch

```
class Net(nn.Module):
       def __init__(self):
                super(Net, self).__init__()
                self.hidden = nn.Linear(X.shape[1], n_hidden)
                self.hidden2 = nn.Linear(n_hidden, n_hidden)
                self.out   = nn.Linear(n_hidden, 1)

       def forward(self, x):
                x = F.relu(self.hidden(x))
                x = F.tanh(self.hidden2(x))
                x = self.out(x)
                return x
```

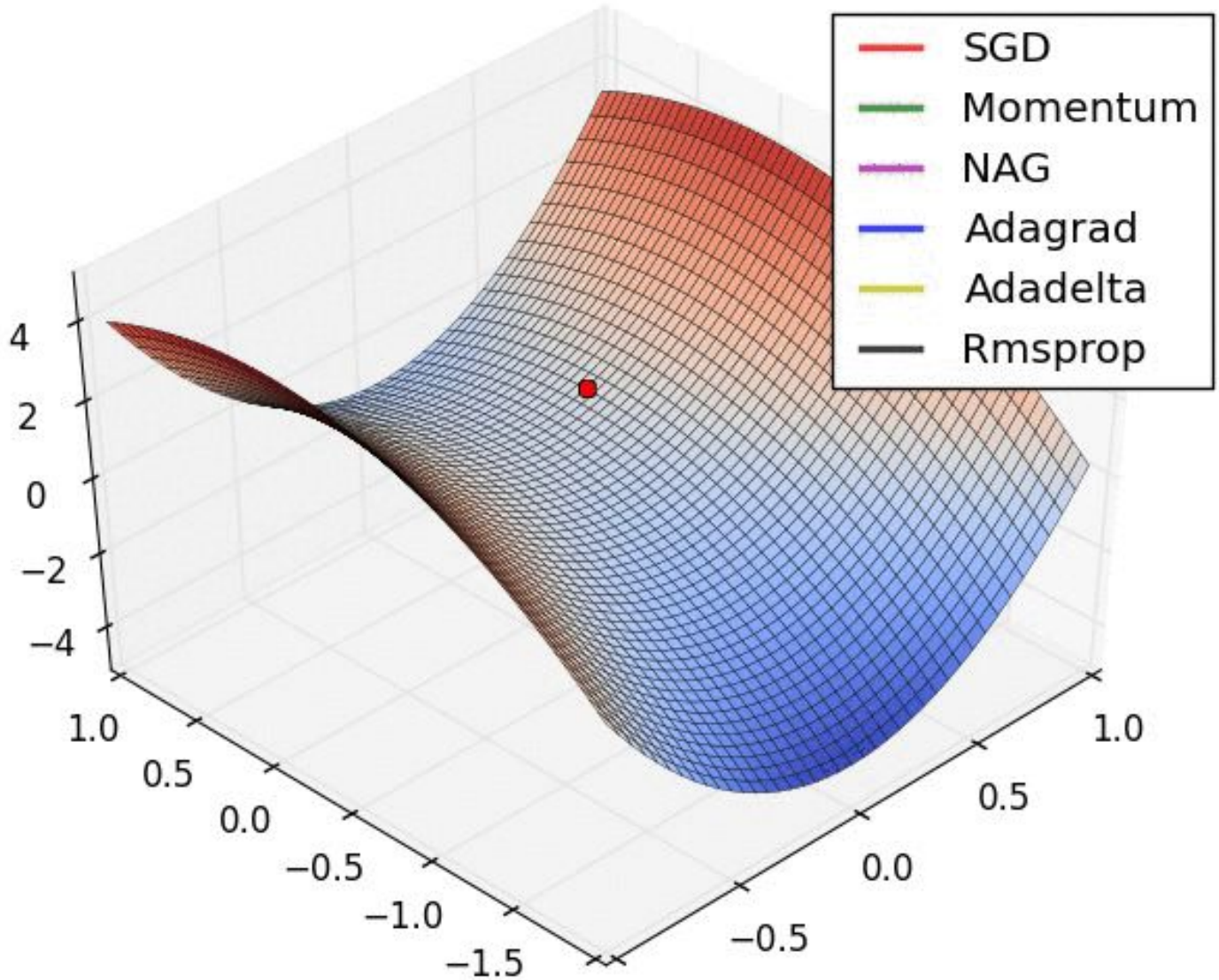
Transfer learning

- You may just have a few examples
- You may not have enough time or hardware
- SOLUTION: reuse an existing model!

Gradient Descent: Adam

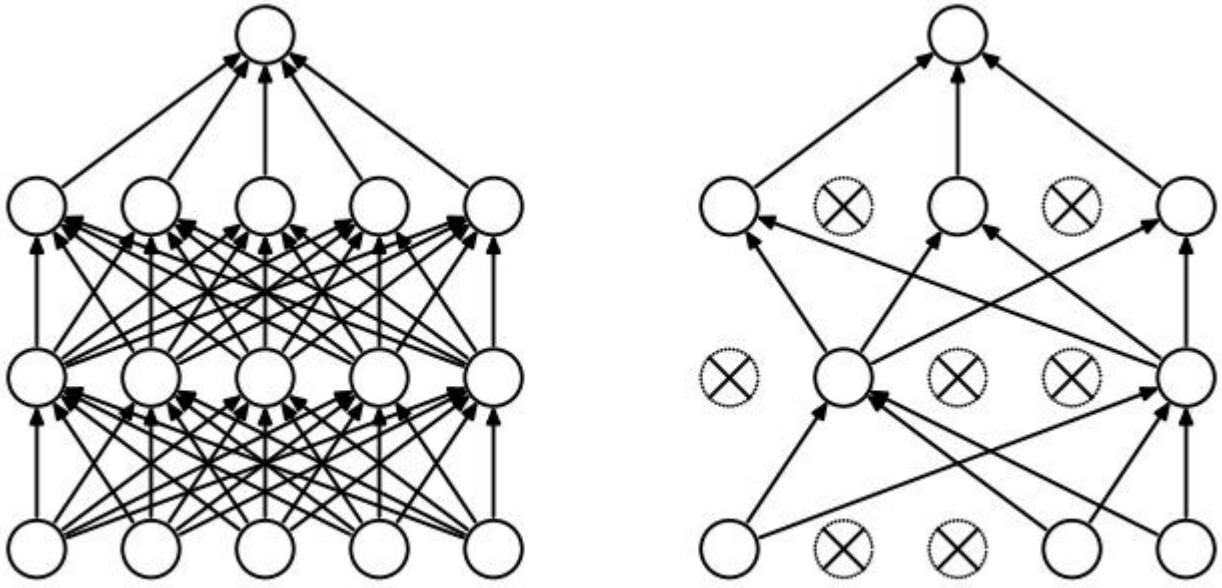
- Main problem of gradient descent: how to choose the learning rate?
- Idea: adaptative, per parameter, learning rate
- In practice: <u>silver bullet</u>
- May give a slightly worse model than SGD, but without any parameter tweaking

Gradient Descent: Adam



Dropout

- Prevent co-adaptation of neurons during training
- Very efficient with dense layers



(a) Standard Neural Net (b) After applying dropout.

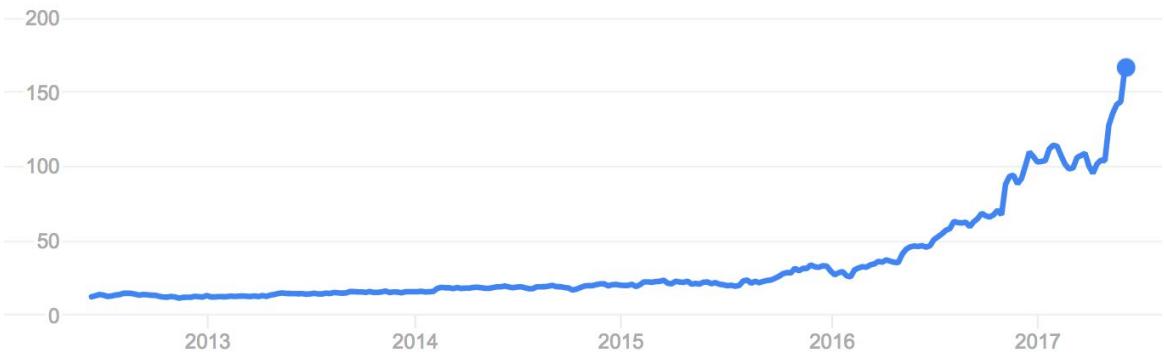
“

Trial & error is still king

CPU or GPU

- CPU are inefficient for training
  - Only for inference and transfer learning

- NVIDIA only



Ouverture	164,74	Capitalis.	105,59 Md
+Haut	168,50	Cours/bén.	56
+Bas	164,15	Rend. div.	0,34%

GPU: which one?

- Buying: professional cards are NOT needed

	TFlops	Memory (Go)	Price	Comment
GTX 1070	5.7	8	399$	Best flops/$
GTX 1080 ti	10.6	11	700$	Reference
Titan XP	11.3	12	1200$	Most powerful, “Professional”

- Cloud computing : more flexible, more expensive, less powerful

amazon web services™

Azure

Paperspace

Google Cloud Platform

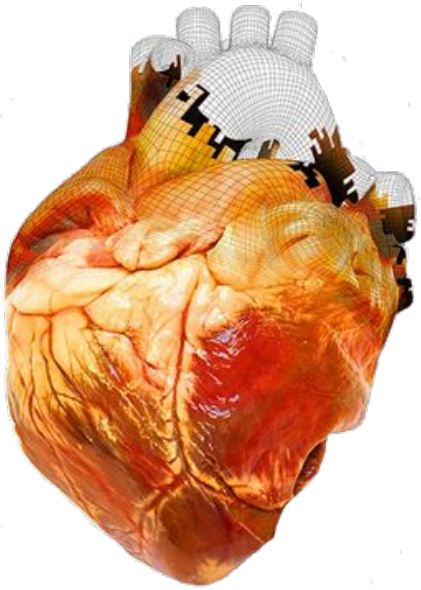
Papers, algorithms: so many papers

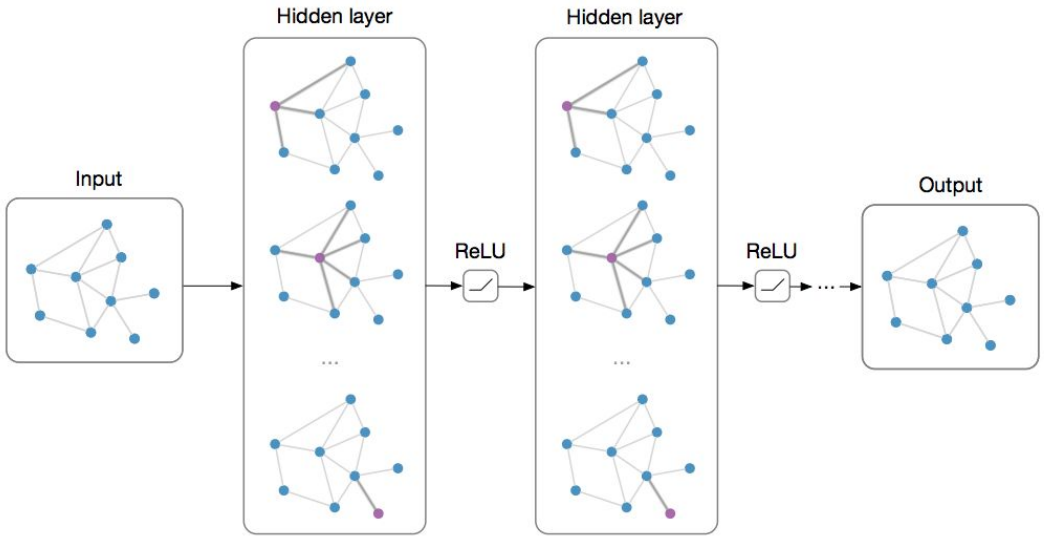
arXiv.org

7.

Futur of deep learning

Tomorrow: Data
Harder, Bigger, Diverse





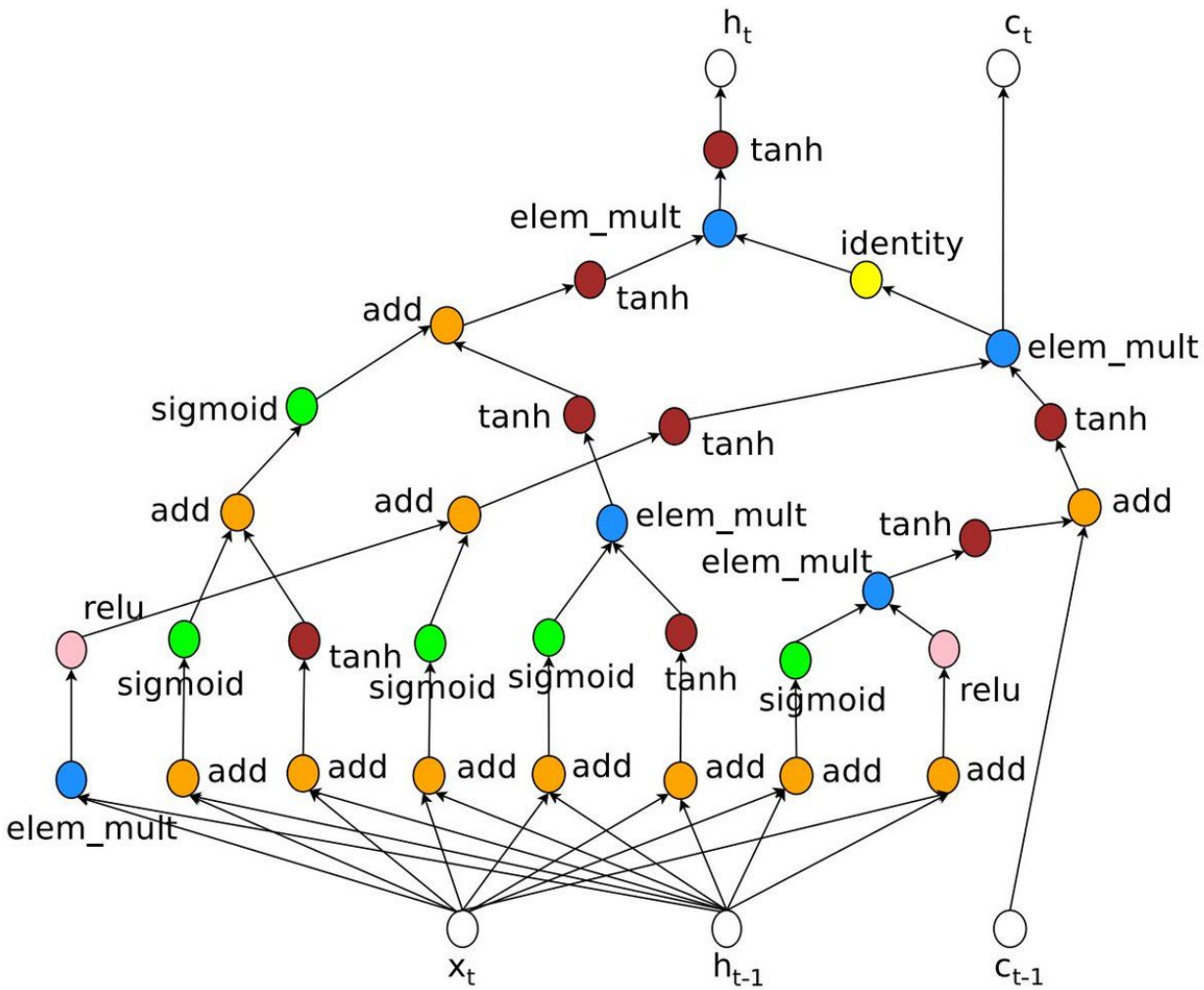
Tomorrow: Specialized hardware

Tensor Processing Unit (TPU)

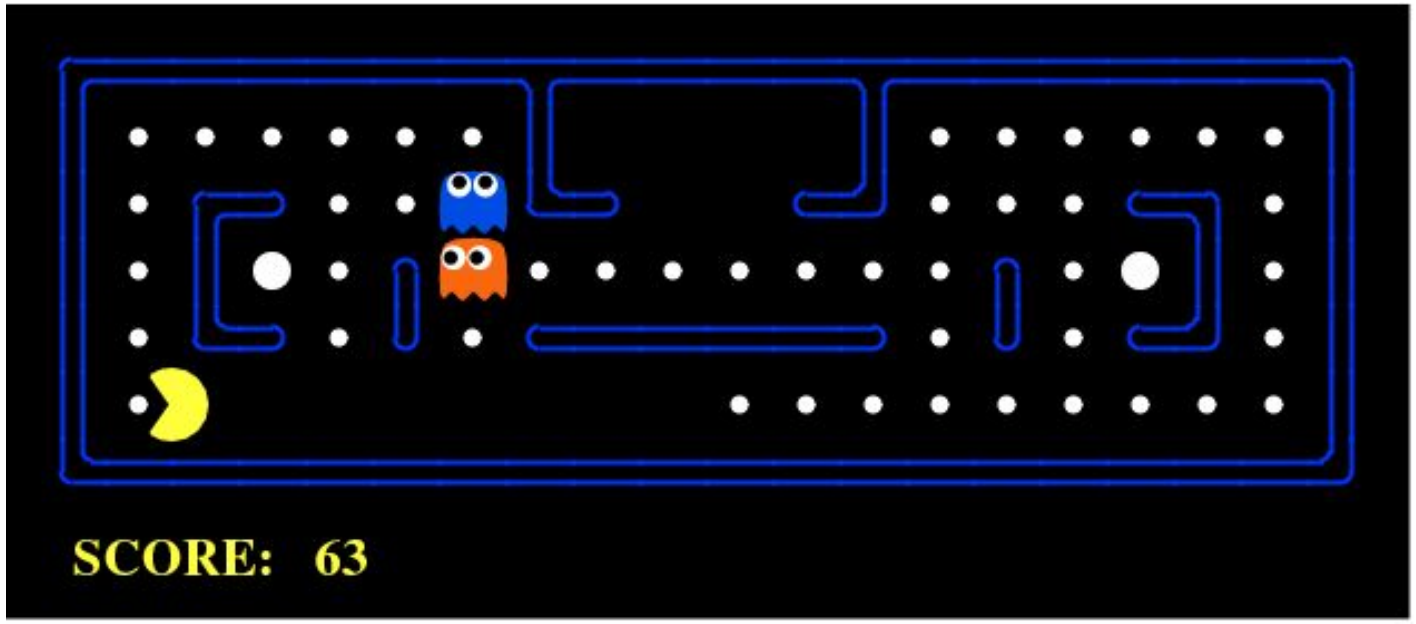
- **30-80x** TOPS/watt vs. 2015 CPUs and GPUs.
- 8 GiB DRAM.
- 8-bit fixed point.
- 256x256 MAC unit.
- Support for data reordering, matrix multiply, activation, pooling, and normalization.

Figure 3. TPU Printed Circuit Board. It can be inserted in the slot for an SATA disk in a server, but the card uses PCIe Gen3 x16.

Tomorrow: AutoML



Tomorrow: Reinforcement Learning



Ressources

- **Deep Learning Book**
- **Stanford CS231n: Convolutional Neural Networks for Visual Recognition**
- **Stanford CS224d: Deep Learning for Natural Language Processing**
- **Fast.AI : Making neural nets uncool again**
- **Papers!**

Thanks !

Questions?