[course site]

#DLUPC

Day 1 Lecture 3

Convolutional and Recurrent Neural Models with Attention

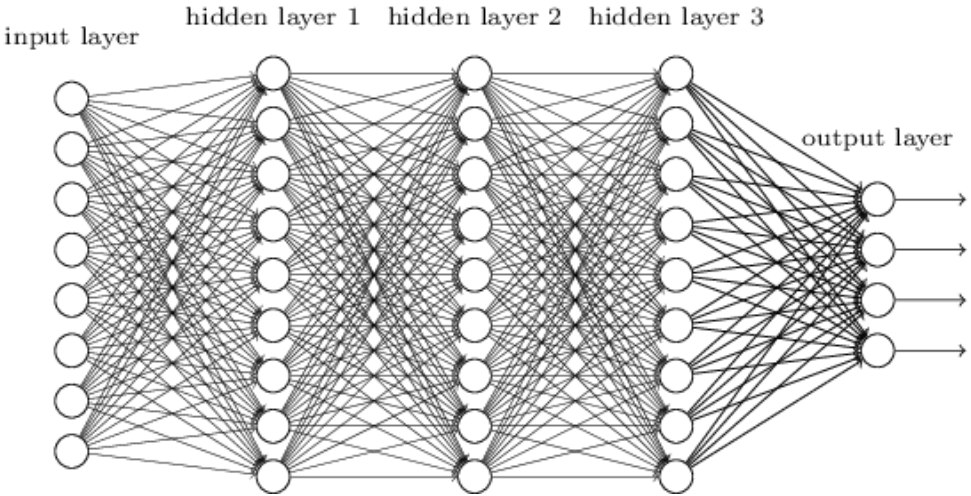
Xavier Giro-i-Nieto
xavier.giro@upc.edu

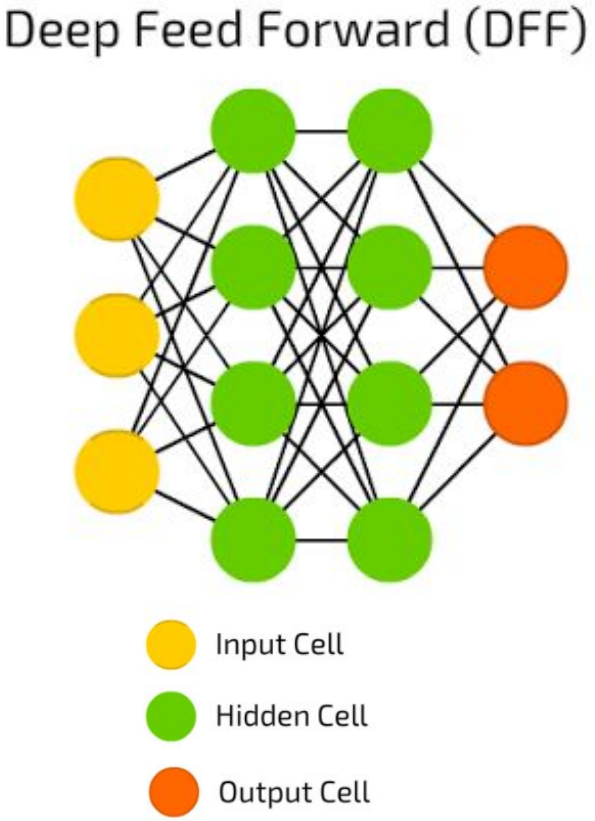
Associate Professor
Universitat Politecnica de Catalunya
Technical University of Catalonia

Outline

1. **Deep Convolutional Networks**
2. Recurrent Neural Networks
3. Attention Models

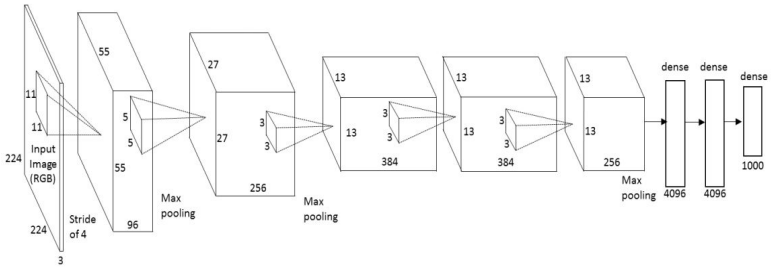
Deep Neural Network (DNN)

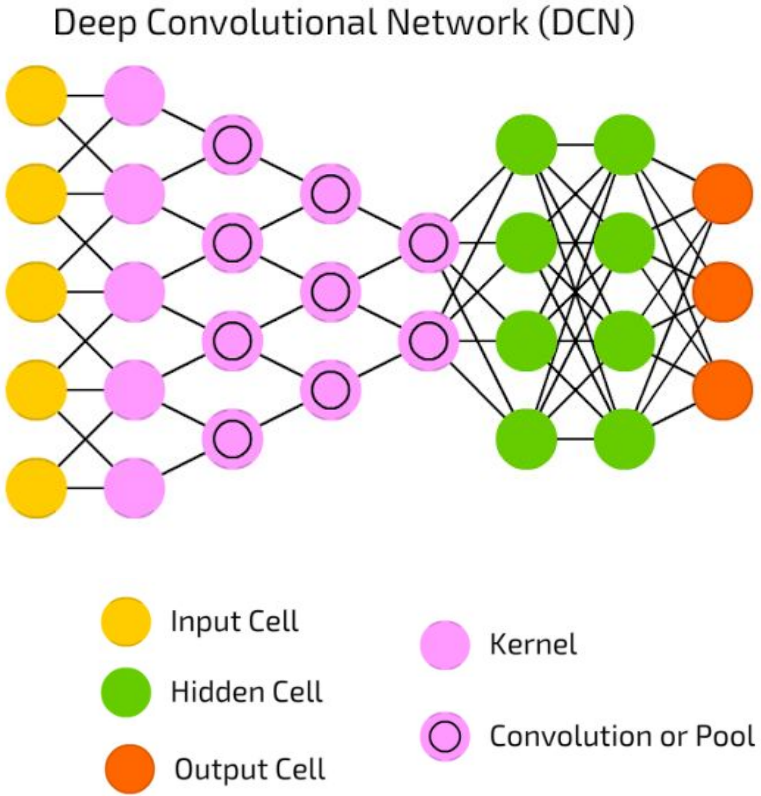




F. Van Veen, [“The Neural Network Zoo”](#) (2016)

Convolutional Neural Network (CNN)



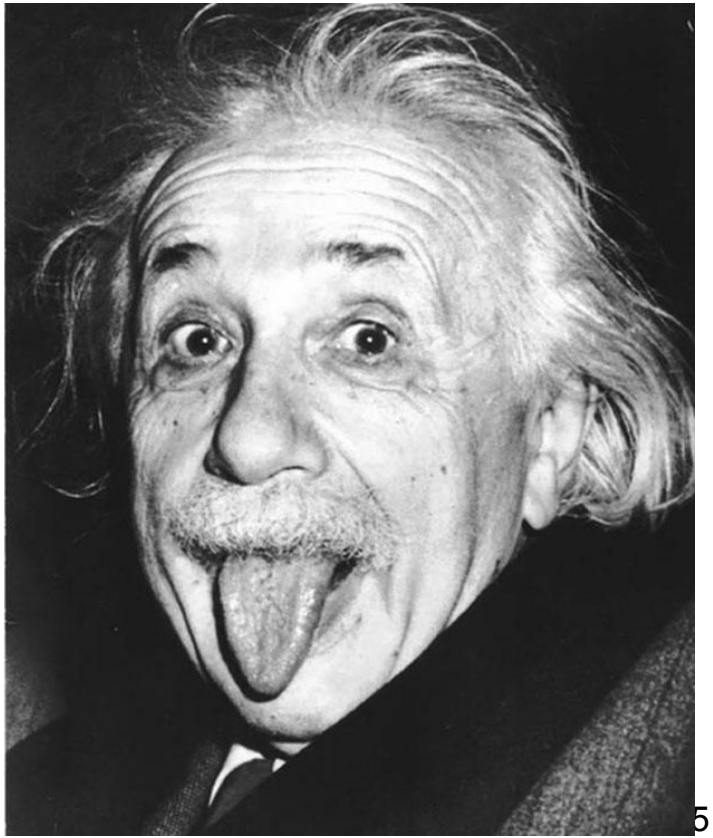


F. Van Veen, [“The Neural Network Zoo”](#) (2016)

Convolutional Neural Network (CNN)

What if the input is a 2D signal?

(images, spectrogram, but also 1D signals)



From Neurons to Convolutional Neural Networks

Fully connected network:

For a 200x200 image, we have $4x10^4$ neurons each one with $4x10^4$ inputs, that is $16x10^8$ parameters (*), only for one layer!!!

(*) biases not counted

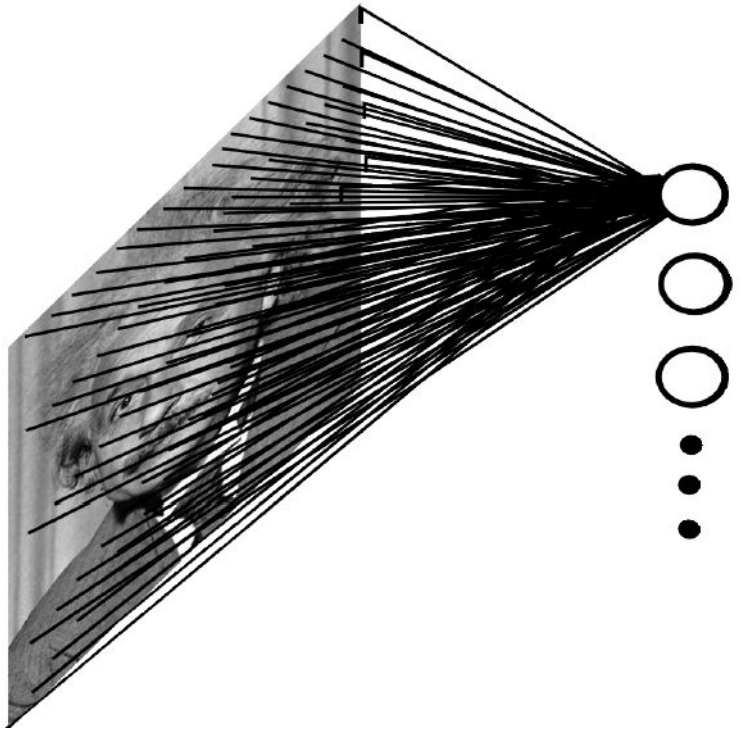
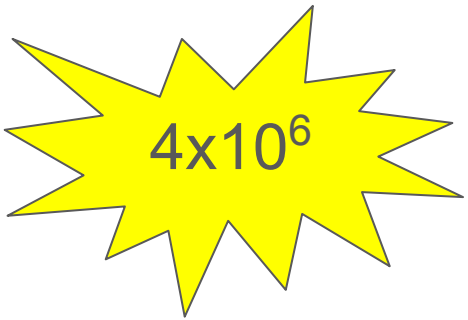


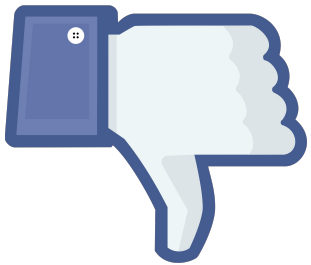
Figure Credit: Ranzatto

From Neurons to Convolutional Neural Networks

Locally connected network:

For a 200x200 image, we have $4x10^4$ neurons each one with 10x10 "**local connections**" (also called receptive field) inputs, that is $4x10^6$





What else can we do to reduce the number of parameters?

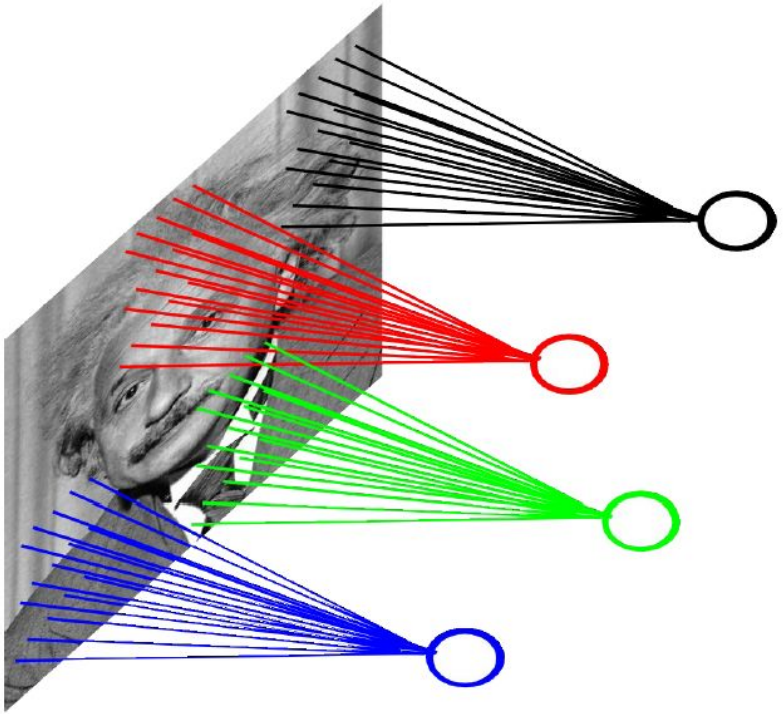


Figure Credit: Ranzatto

From Neurons to Convolutional Neural Networks

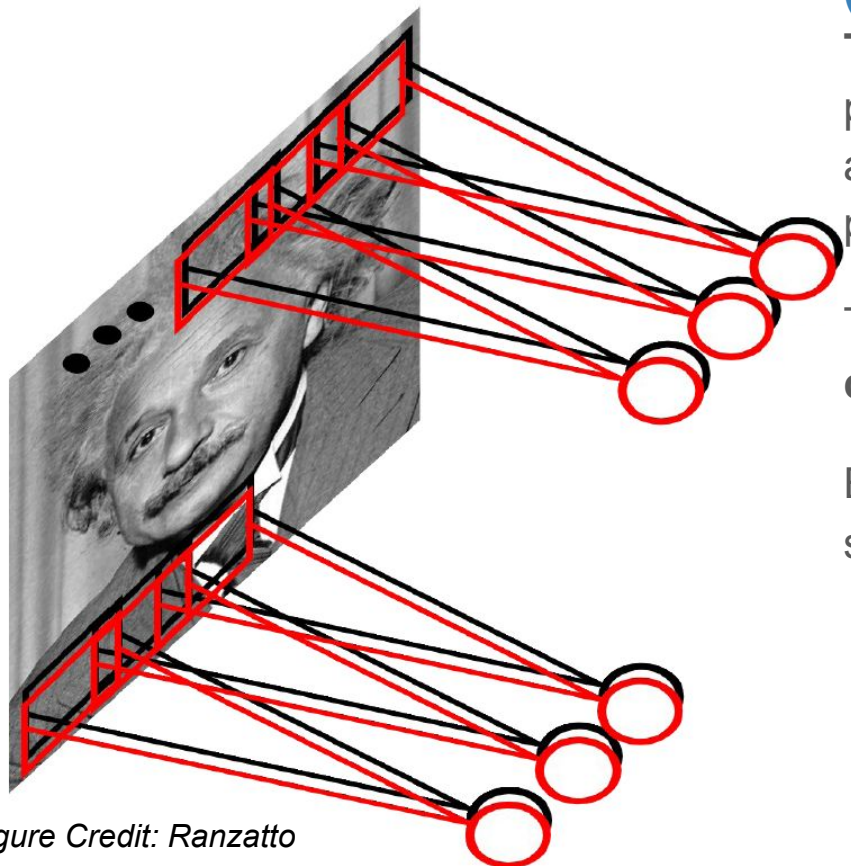


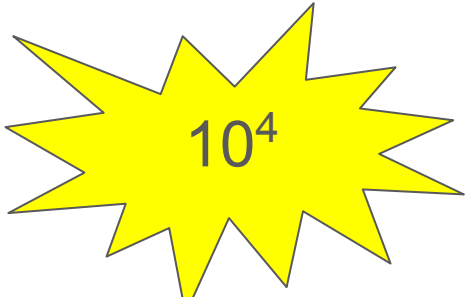
Figure Credit: Ranzatto

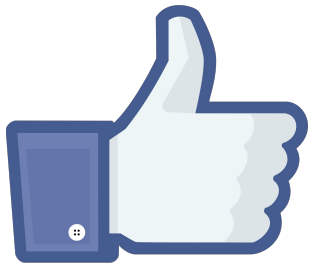
Convolutional Neural Networks (ConvNets, CNNs):

Translation invariance: we can use same parameters to capture a specific “feature” in any area of the image. We can try different sets of parameters to capture different features.

These operations are equivalent to perform **convolutions** with different filters.

Ex: With 100 different filters (or feature extractors) of size 10x10, the number of parameters is 10^4





From Neurons to Convolutional Neural Networks

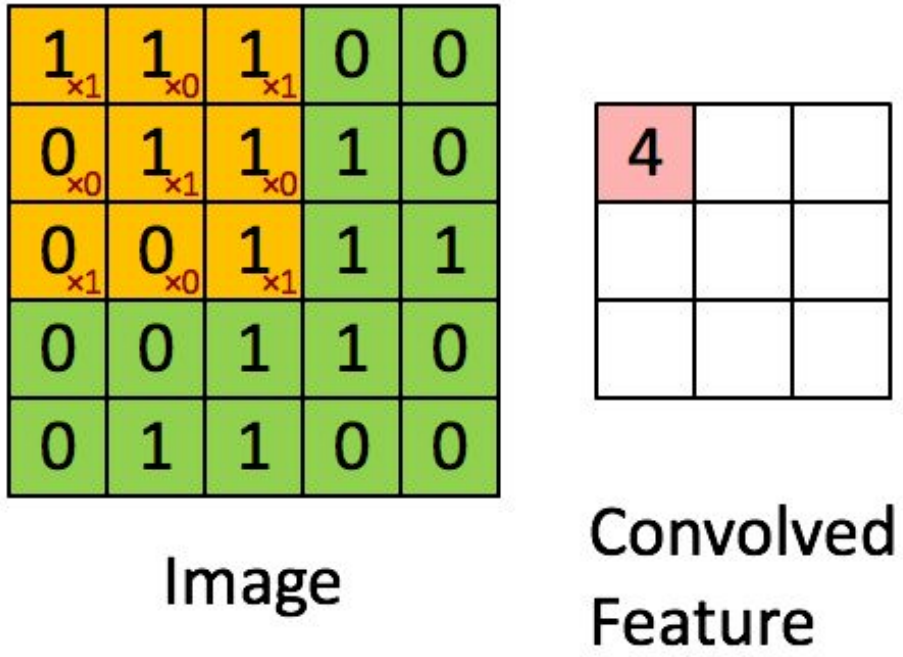


Image credit: Deep learning Tutorial (Stanford University)

From Neurons to Convolutional Neural Networks

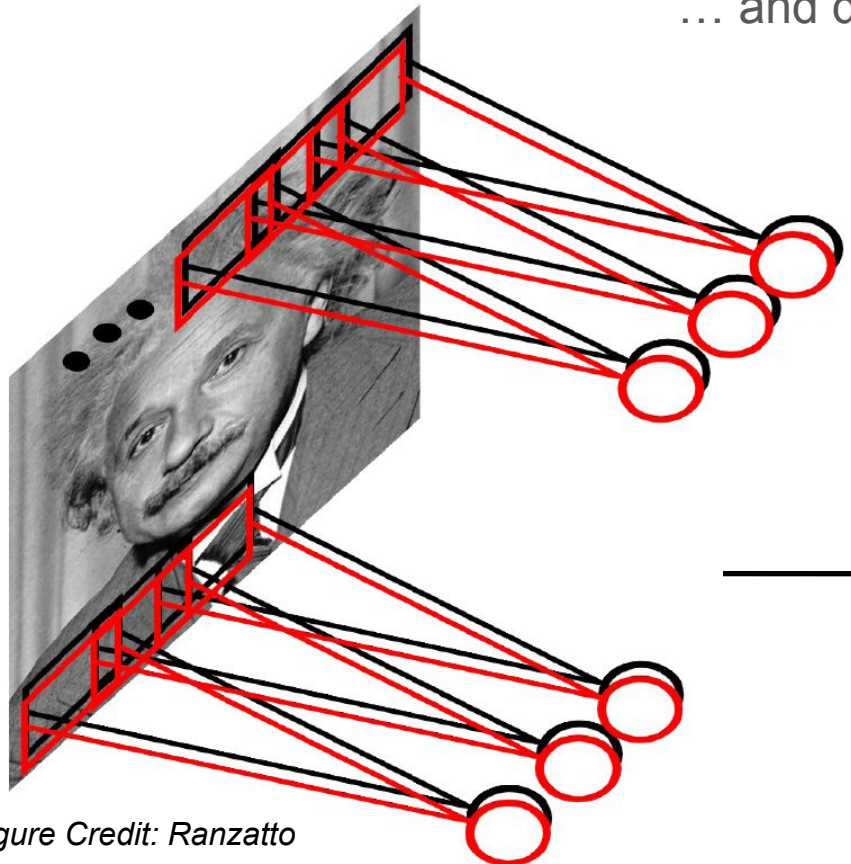
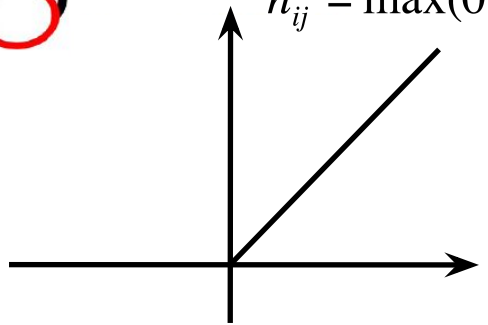


Figure Credit: Ranzatto

… and don't forget the non-linear activation function!

$$a_{ij} = \sum_{k,l} w_{kl} x_{k-i,l-j} + b$$

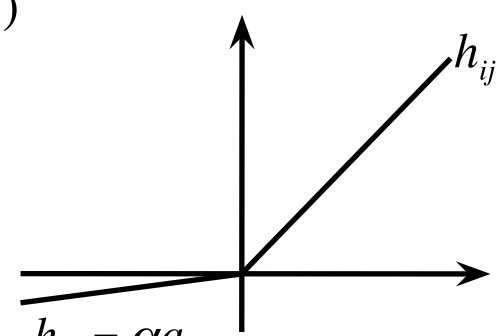
$$h_{ij} = \max(0, a_{ij})$$



ReLu

$$h_{ij} = \alpha a_{ij}$$

$$h_{ij} = a_{ij}$$



PReLu

From Neurons to Convolutional Neural Networks

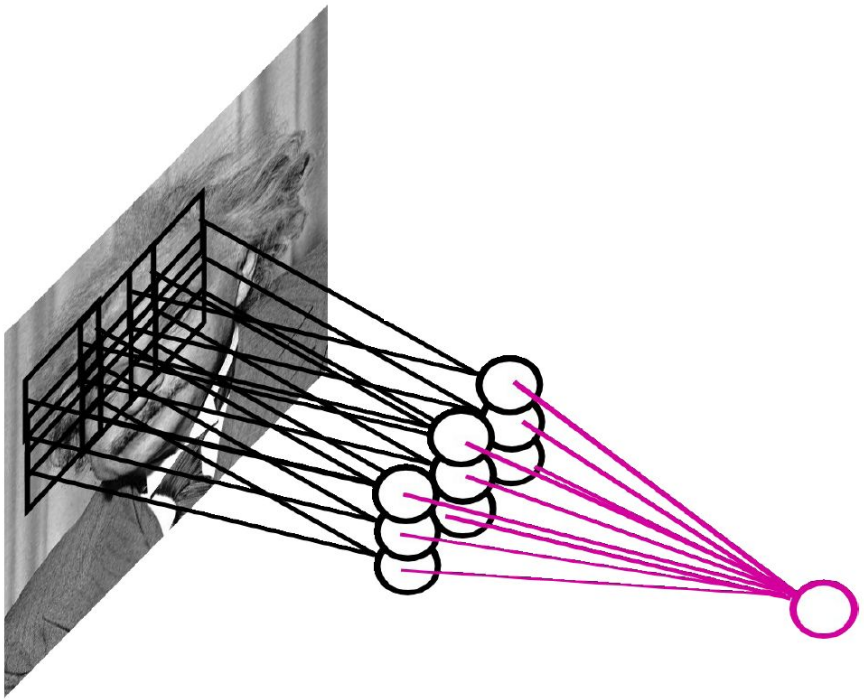


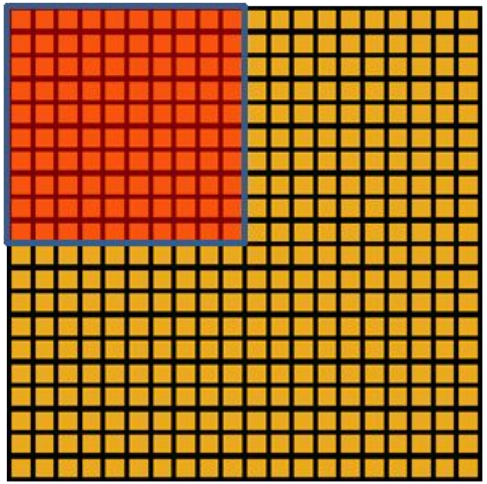
Figure Credit: Ranzatto

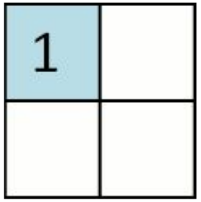
Most ConvNets also include use **Pooling layers** (or subsampling) to reduce dimensionality and provide invariance to small local changes.

Pooling options:

- **Max**
- Average
- Stochastic pooling

From Neurons to Convolutional Neural Networks





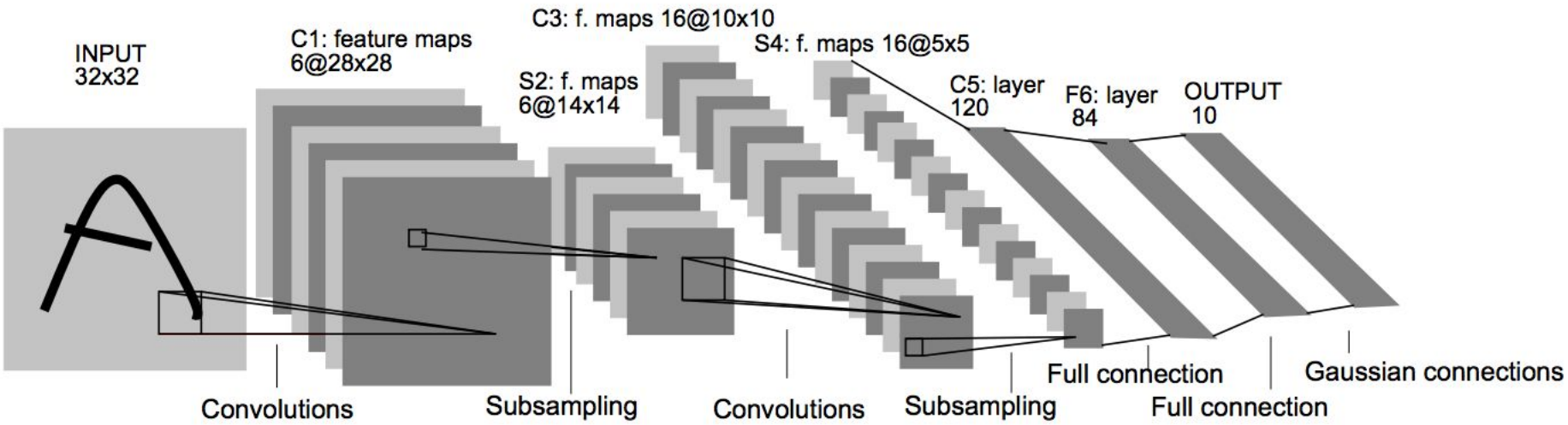
Convolved feature

Pooled feature

Image credit: Deep learning Tutorial (Stanford University)

Convolutional Neural Network (CNN)

LeNet-5



LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278-2324.

Convolutional Neural Network (CNN)

Yann LeCun
t'ha retuitat el tuit.

Mostra-ho

Yann LeCun

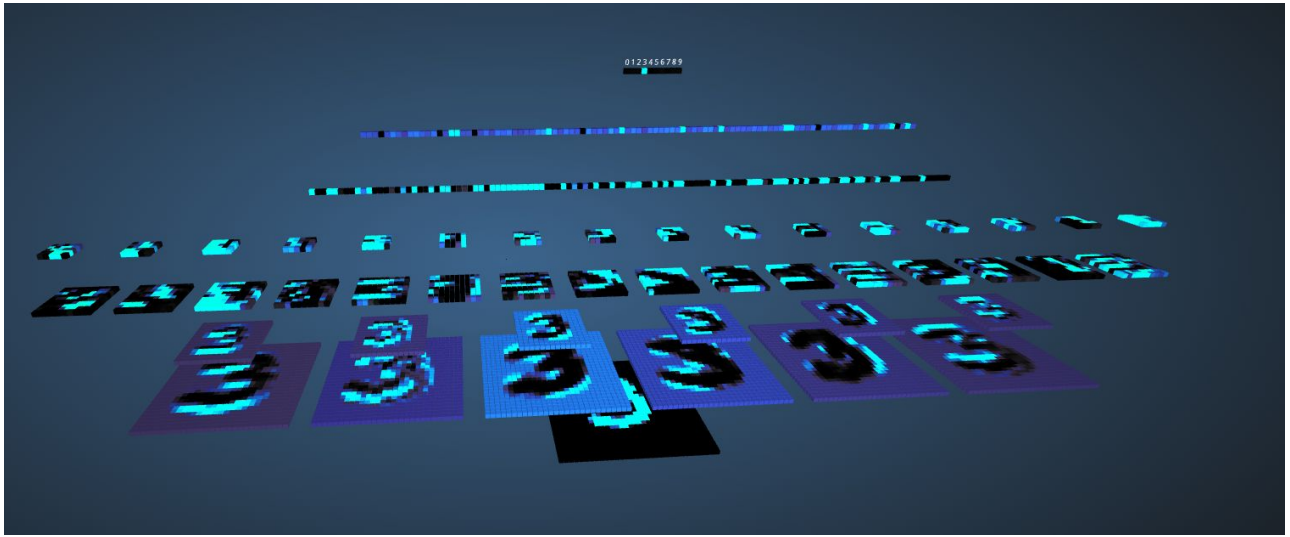
NYU | COURANT INSTITUTE OF MATHEMATICAL SCIENCES

Facebook AI Research

You can also check @boredyannlecun on Twitter...

Convolutional Neural Network (CNN)

Demo: 3D Visualization of a Convolutional Neural Network



Harley, Adam W. "An Interactive Node-Link Visualization of Convolutional Neural Networks." In Advances in Visual Computing, pp. 867-877. Springer International Publishing, 2015.

Convolutional Neural Network (CNN)

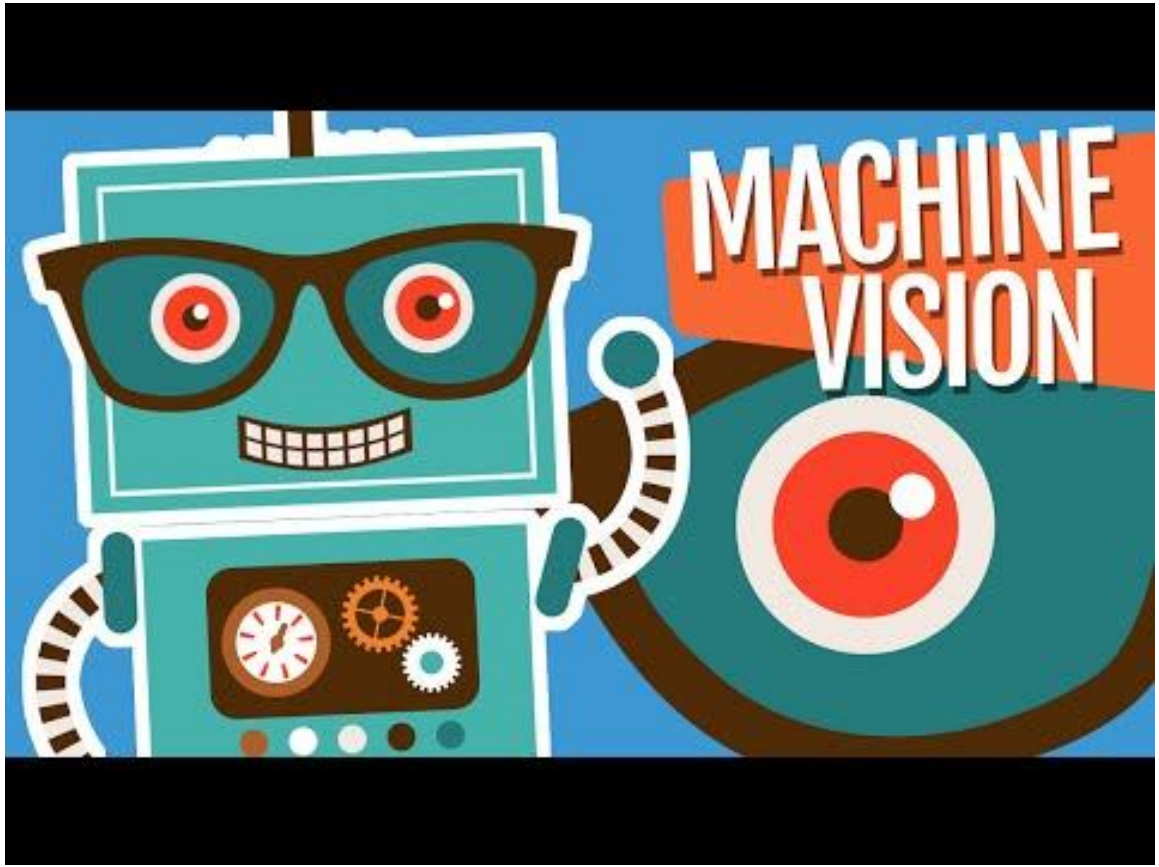
Demo: [Classify MNIST digits with a Convolutional Neural Network]

ConvNetJS

Deep Learning in your browser

“ConvNetJS is a Javascript library for training Deep Learning models (mainly Neural Networks) entirely in your browser. Open a tab and you're training. No software requirements, no compilers, no installations, no GPUs, no sweat.”

Convolutional Neural Network (CNN)



Deep Learning TV, [“Convolutional Neural Networks - Ep. 8”]

Outline

1. Deep Convolutional Networks
2. **Recurrent Neural Networks**
3. Attention Models

Acknowledgments

Santiago Pascual

The importance of context

- Recall the 5th digit of your phone number
- Sing your favourite song beginning at third sentence
- Recall 10th character of the alphabet

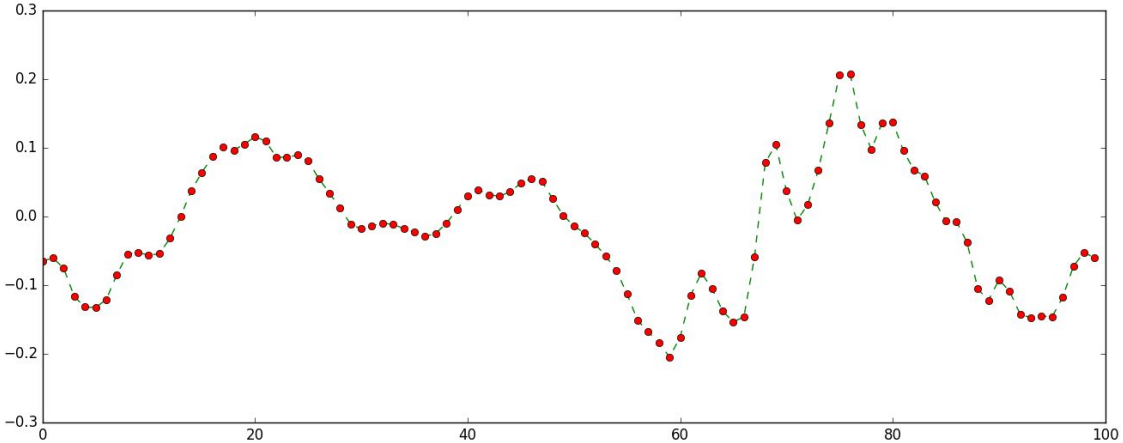
Probably you went straight from the beginning of the stream in each case…

because in sequences order matters!

Idea: retain the information preserving the importance of order

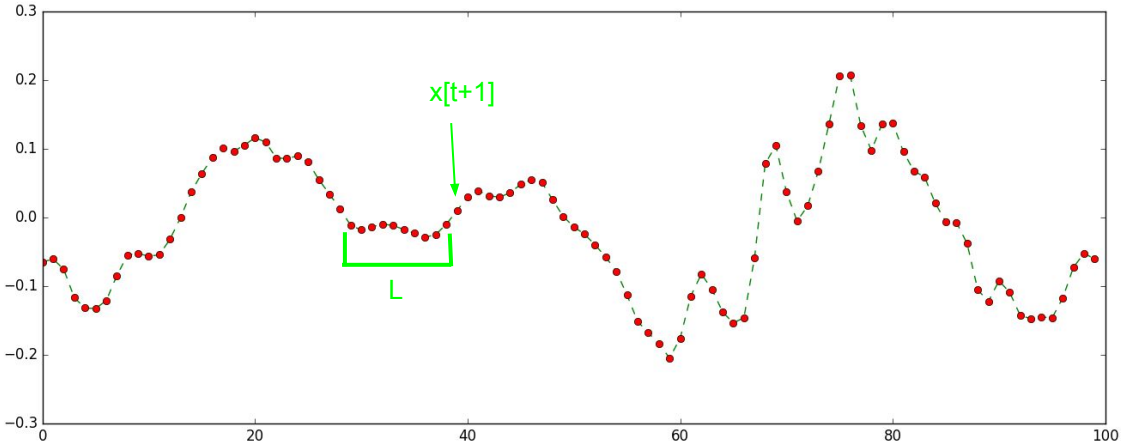
Where is the Memory?

If we have a sequence of samples...



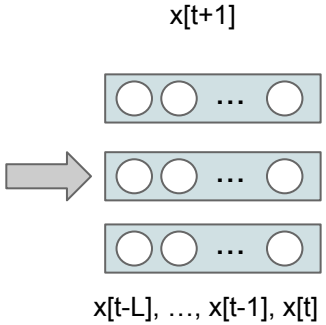
predict sample x[t+1] knowing previous values {x[t], x[t-1], x[t-2], …, x[t-τ]}

Where is the Memory?

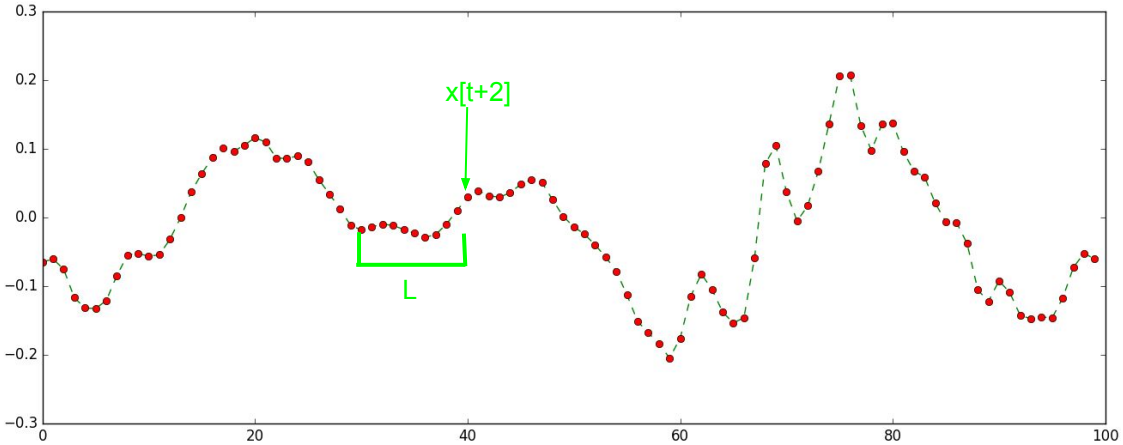


Feed Forward approach:

- static window of size L
- slide the window time-step wise

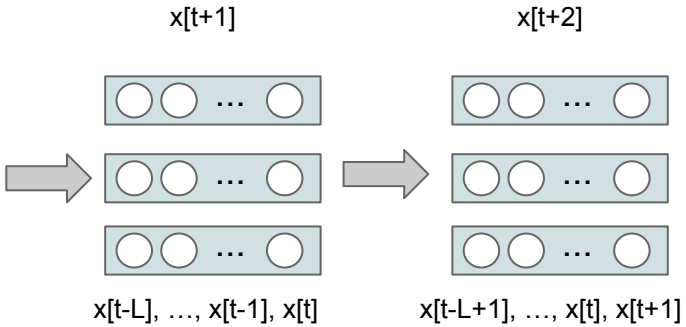


Where is the Memory?

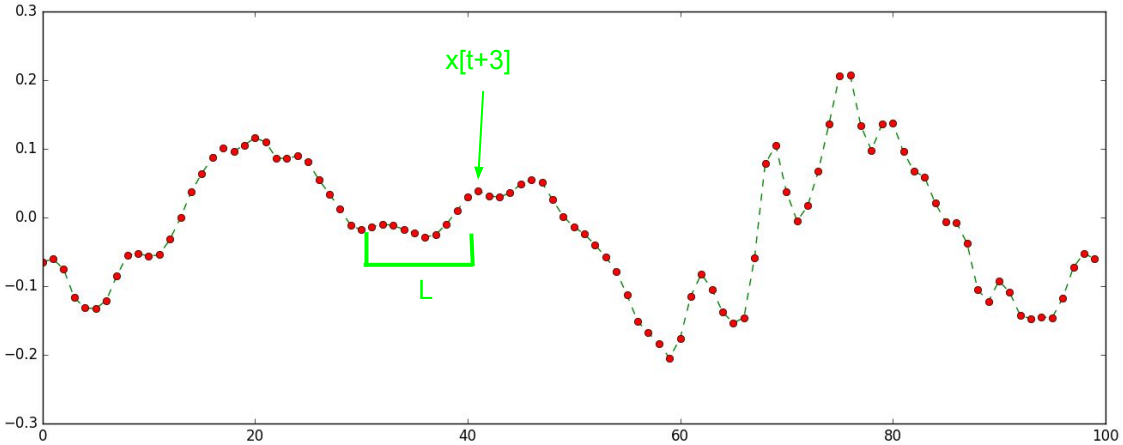


Feed Forward approach:

- static window of size L
- slide the window time-step wise

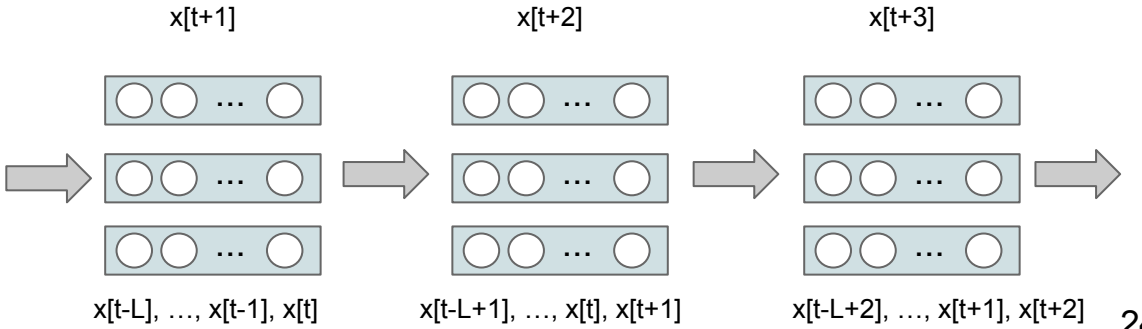


Where is the Memory?



Feed Forward approach:

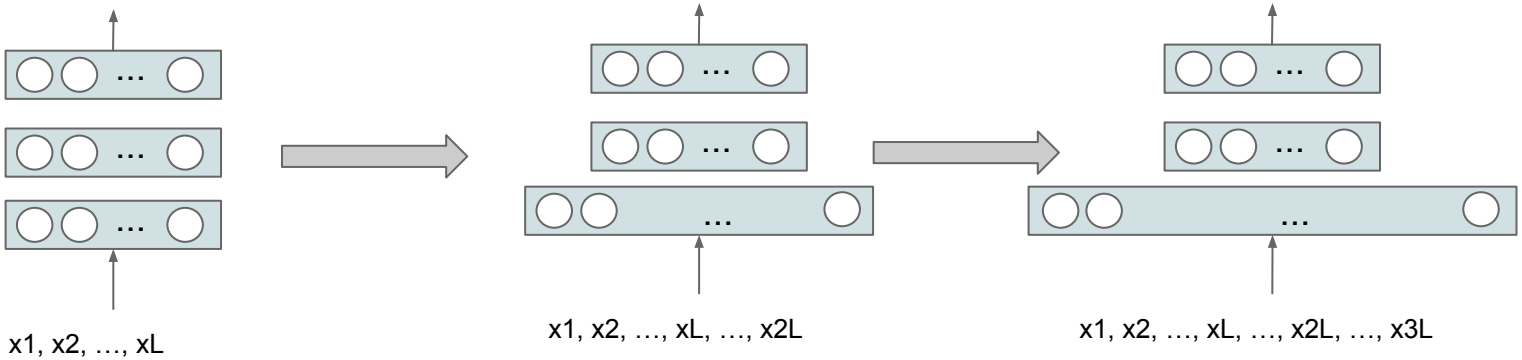
- static window of size L
- slide the window time-step wise



Where is the Memory?

Problems for the feed forward + static window approach:

- What's the matter increasing L? → Fast growth of num of parameters!
- Decisions are independent between time-steps!
  - The network doesn't care about what happened at previous time-step, only present window matters → doesn't look good
- Cumbersome padding when there are not enough samples to fill L size
  - Can't work with variable sequence lengths

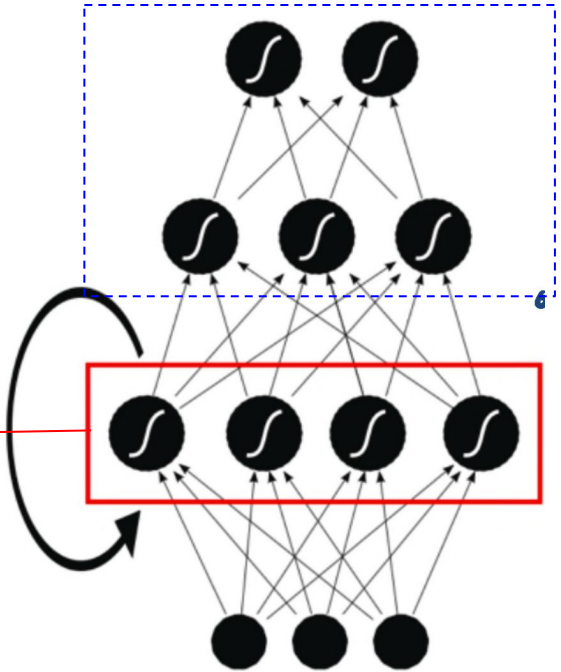


Recurrent Neural Network

Solution: Build specific connections capturing the temporal evolution → **Shared weights in time**

- Give volatile memory to the network

$$h_t = f(W \cdot x_t + U \cdot h_{t-1} + b)$$

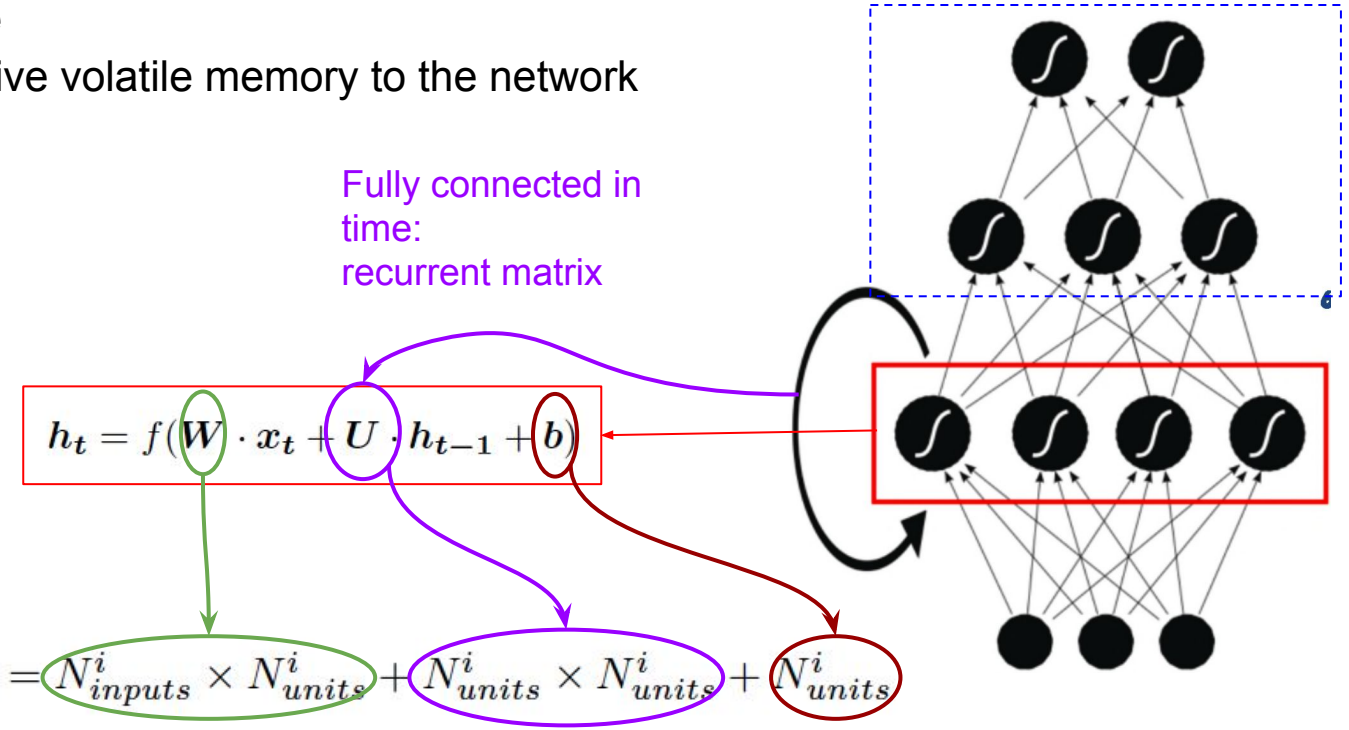


Feed-Forward
Fully-Connected

Recurrent Neural Network

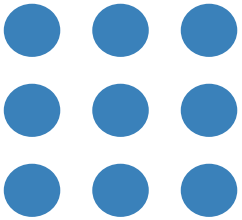
Solution: Build specific connections capturing the temporal evolution → **Shared weights in time**

- Give volatile memory to the network



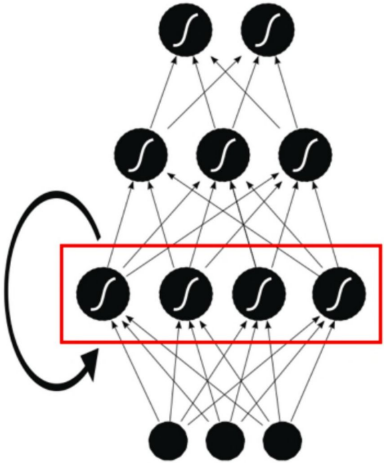
Recurrent Neural Network

Front View

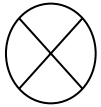


Rotation 90º

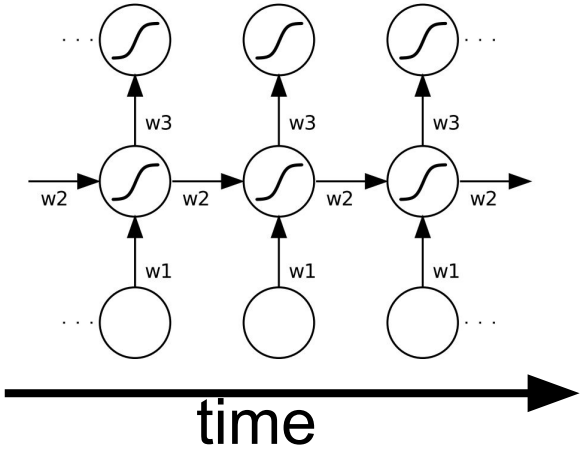
Side View



time



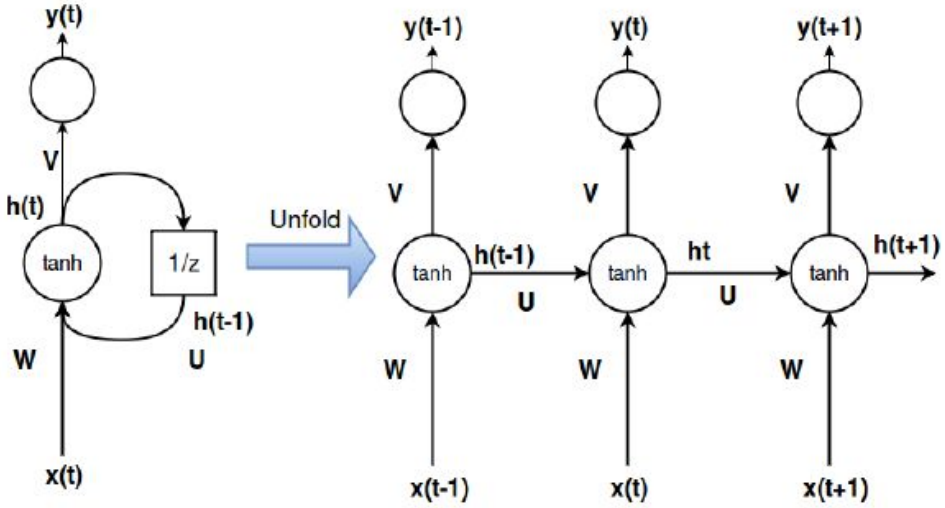
Unfold (Rotation 90º)



Recurrent Neural Network

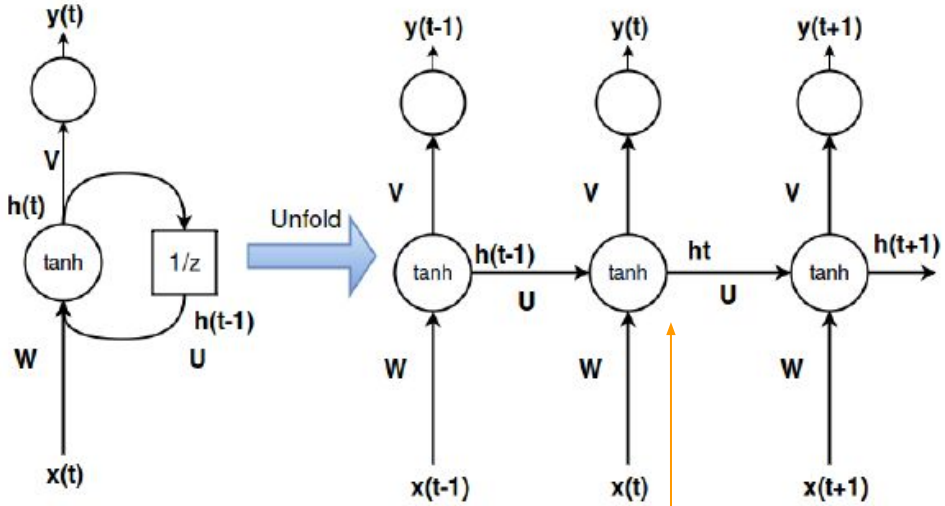
Hence we have two data flows: **Forward in neural layers + time** propagation

BEWARE: We have extra depth now! Every time-step is an extra level of depth (as a deeper stack of layers in a feed-forward fashion!)



Recurrent Neural Network

Hence we have two data flows: **Forward in layers + time** propagation



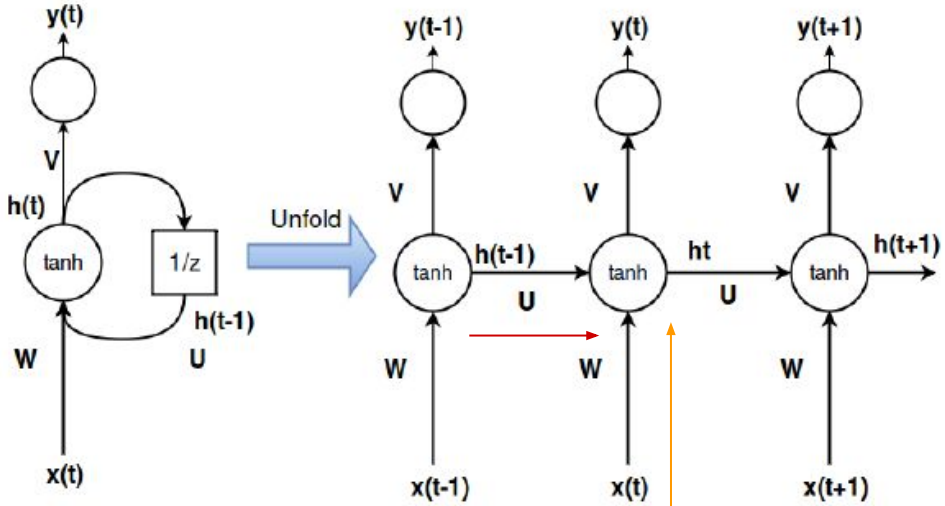
$$\mathbf{h}_t = g(\mathbf{W} \cdot \mathbf{x}_t + \mathbf{U} \cdot \mathbf{h}_{t-1} + \mathbf{b}_h)$$

$$\mathbf{x}_t \in \mathbb{R}^n$$

Recurrent Neural Network

Hence we have two data flows: **Forward in layers + time** propagation

- Last time-step includes the context of our decisions recursively



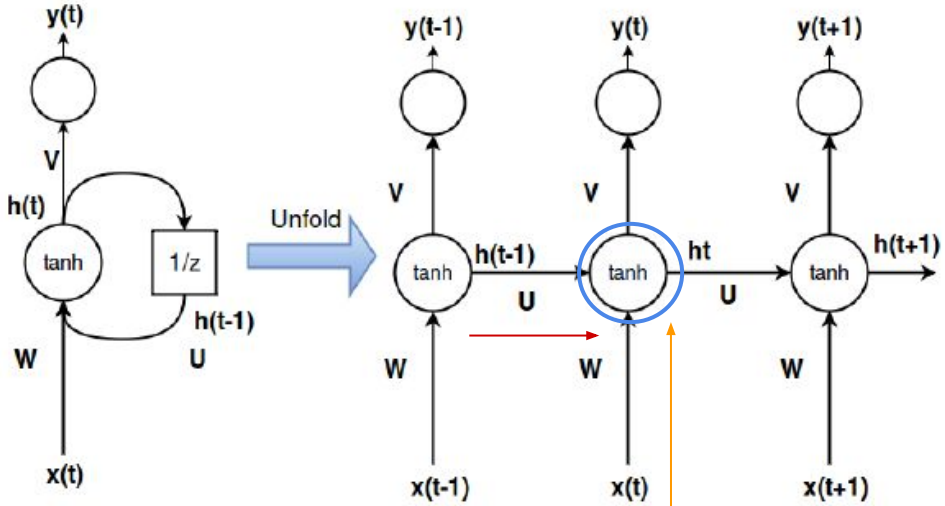
$$\mathbf{h}_t = g(\mathbf{W} \cdot \mathbf{x}_t + \mathbf{U} \cdot \mathbf{h}_{t-1} + \mathbf{b}_h)$$

$$\mathbf{x}_t \in \mathbb{R}^n$$

Recurrent Neural Network

Hence we have two data flows: **Forward in layers + time** propagation

- Last time-step includes the context of our decisions recursively



$$\mathbf{h}_t = g(\mathbf{W} \cdot \mathbf{x}_t + \mathbf{U} \cdot \mathbf{h}_{t-1} + \mathbf{b}_h)$$

Recurrent Neural Network

Back Propagation Through Time (BPTT): The training method has to take into account the time operations:

Total error at the output is the sum of errors at each time-step

$$E(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{t=1}^{T} E_t(\mathbf{y}_t, \hat{\mathbf{y}}_t)$$

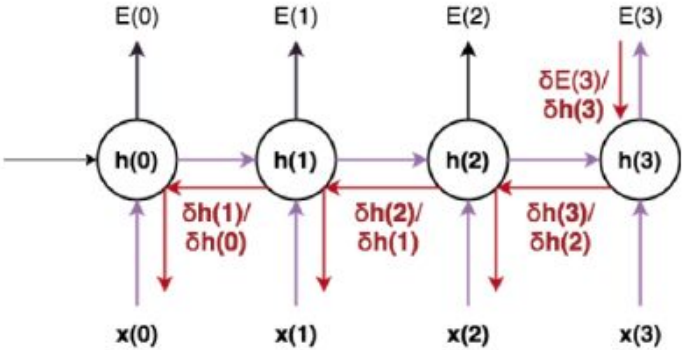
Total gradient is the sum of gradients at each time-step

$$\frac{\partial E}{\partial \mathbf{W}} = \sum_{t=0}^{T-1} \frac{\partial E_t}{\partial \mathbf{W}}$$

T: **max amount of time-steps to do back-prop**. In Keras this is specified when defining the "input shape" to the RNN layer, by means of:
(batch size, sequence length (T), input_dim)

Input shape

3D tensor with shape `(nb_samples, timesteps, input_dim)`.



Example back-prop in time with 3 time-steps

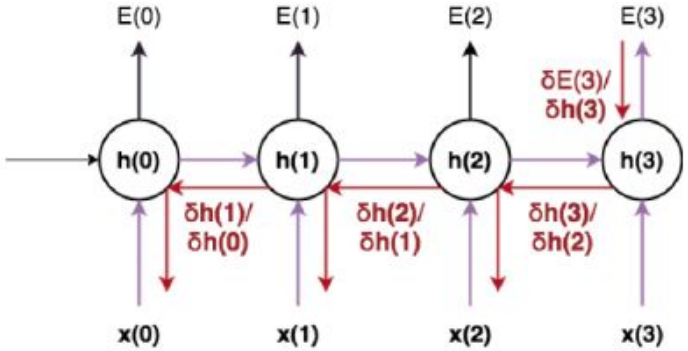
Recurrent Neural Network

Main problems:

- **Long-term memory** (remembering quite far time-steps) **vanishes quickly** because of the recursive operation with **U**

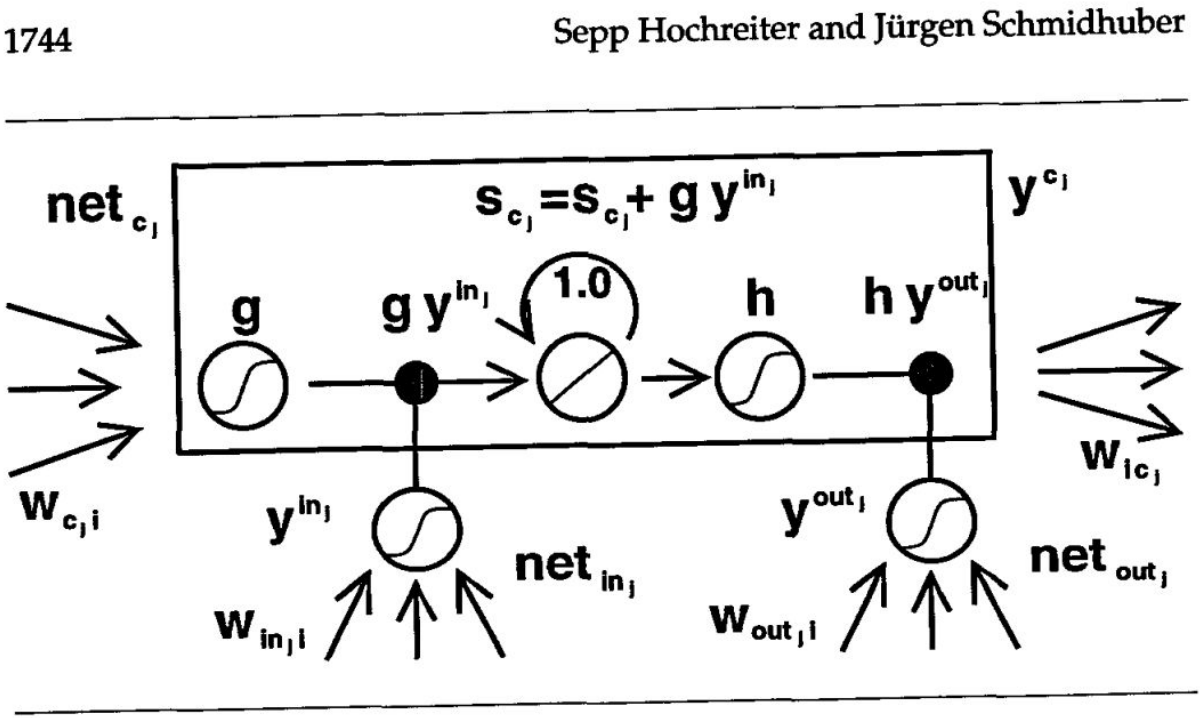
$$\mathbf{h}_t = g(\mathbf{W} \cdot \mathbf{x}_t + \mathbf{U} \cdot g(\cdots g(\mathbf{W} \cdot \mathbf{x}_{t-T} + \mathbf{U} \cdot \mathbf{h}_{t-T} + \mathbf{b}_h) \cdots) + \mathbf{b}_h)$$

- **During training gradients explode/vanish easily because of depth-in-time** → Exploding/Vanishing gradients!



Example back-prop in time with 3 time-steps

Long Short-Term Memory (LSTM)



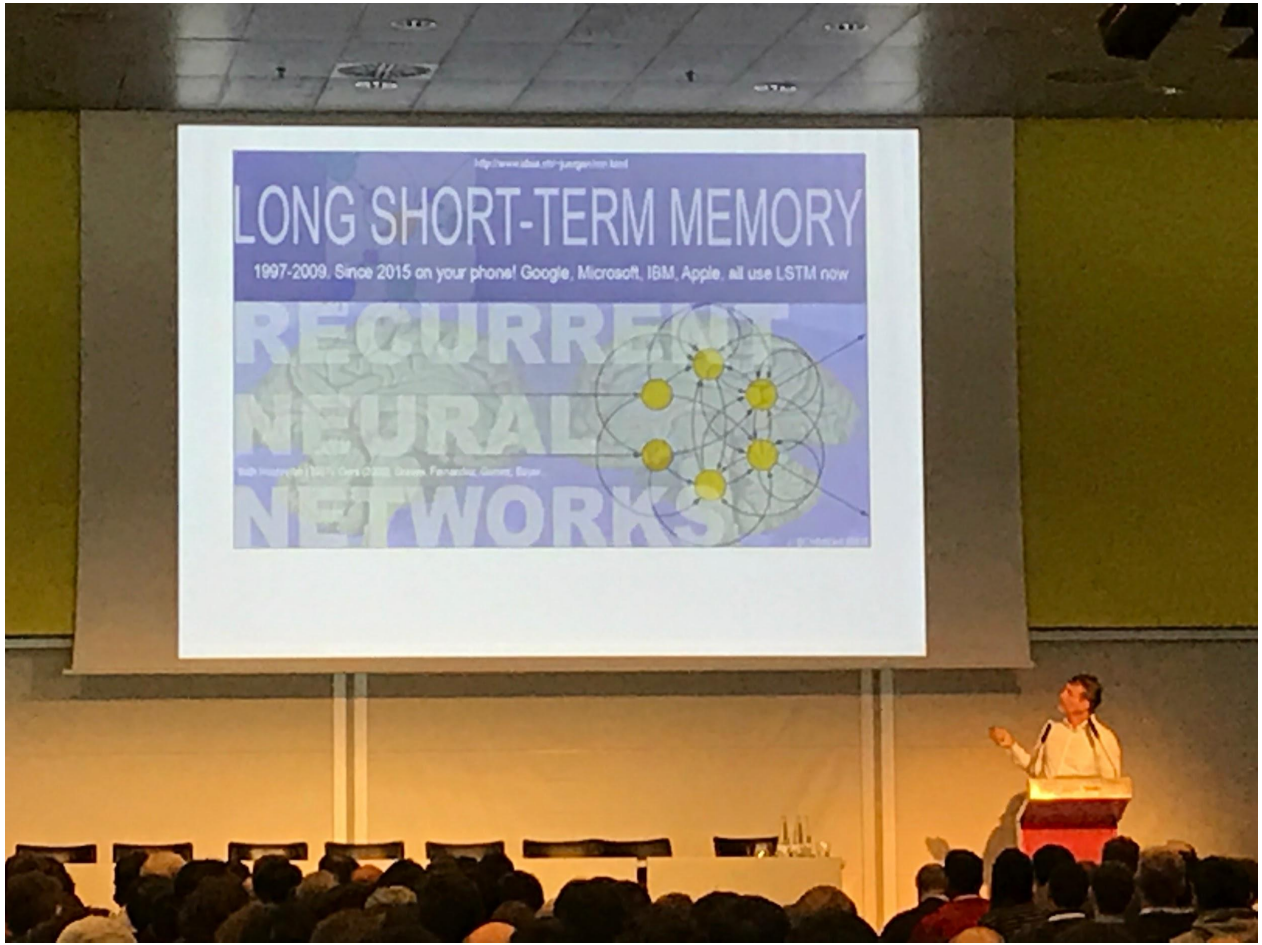
Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9, no. 8 (1997): 1735-1780.

Long Short-Term Memory (LSTM)

The New York Times, “When A.I. Matures, It May Call Jürgen Schmidhuber ‘Dad’” (November 2016)

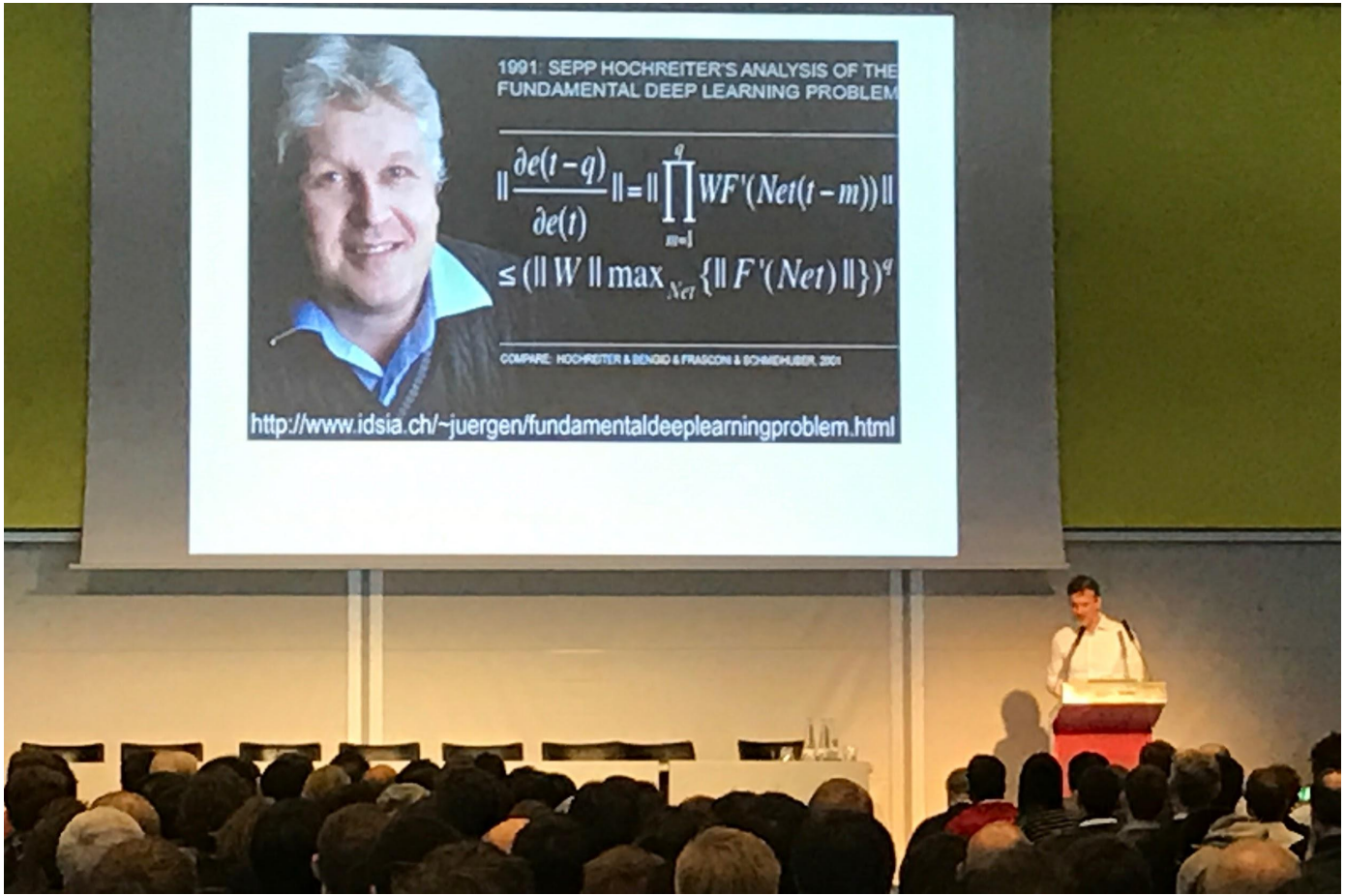
USI/SUPSI
Istituto Dalle Molle di studi sull’intelligenza artificiale
IDSIA

Long Short-Term Memory (LSTM)



Jürgen Schmidhuber @ NIPS 2016 Barcelona

Long Short-Term Memory (LSTM)



Jürgen Schmidhuber @ NIPS 2016 Barcelona

Long Short-Term Memory (LSTM)

Jürgen Schmidhuber @ NIPS 2016 Barcelona

Gating method

Solutions:

1. Change the way in which past information is kept → create the notion of **cell state:** a memory unit that keeps long-term information in a safer way by protecting it from recursive operations
2. Make every RNN unit able to **forget whatever may not be useful anymore** by clearing that info from the cell state (optimized clearing mechanism)
3. Make every RNN unit able to decide whether **the current time-step information matters or not**, to accept or discard (optimized reading mechanism)
4. Make every RNN unit able to **output the decisions whenever it is ready to do so** (optimized output mechanism)

Long Short-Term Memory (LSTM)

Three **gates** are governed by *sigmoid* units (btw [0,1]) define the control of in & out information with a product..

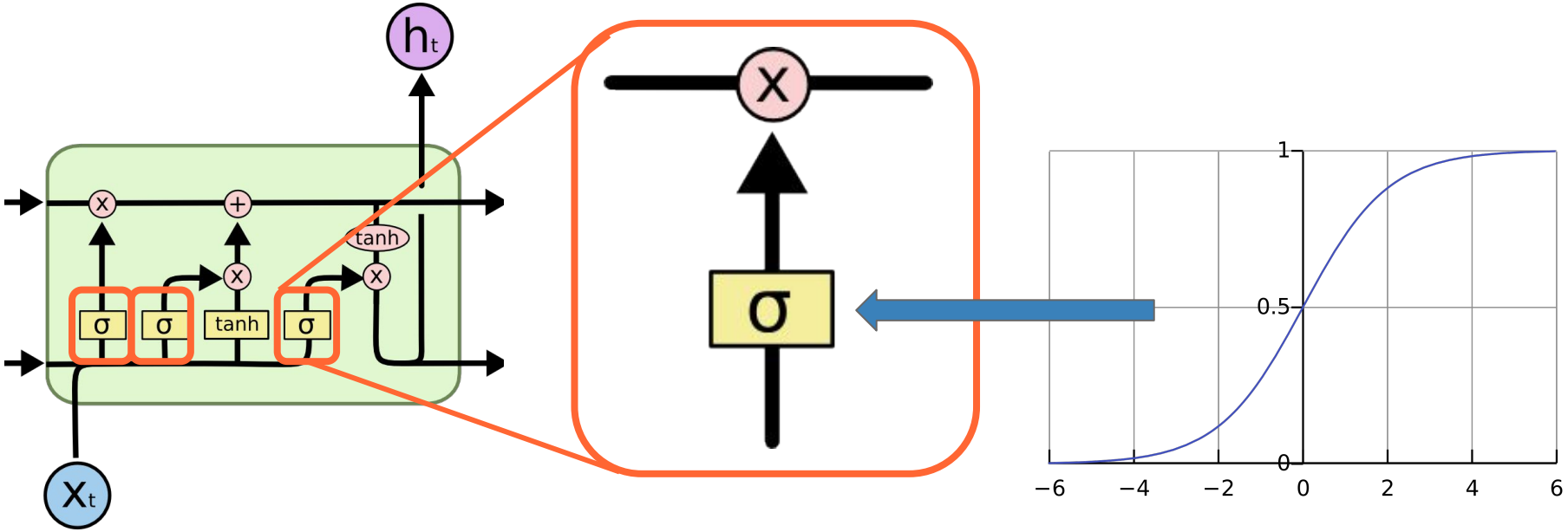
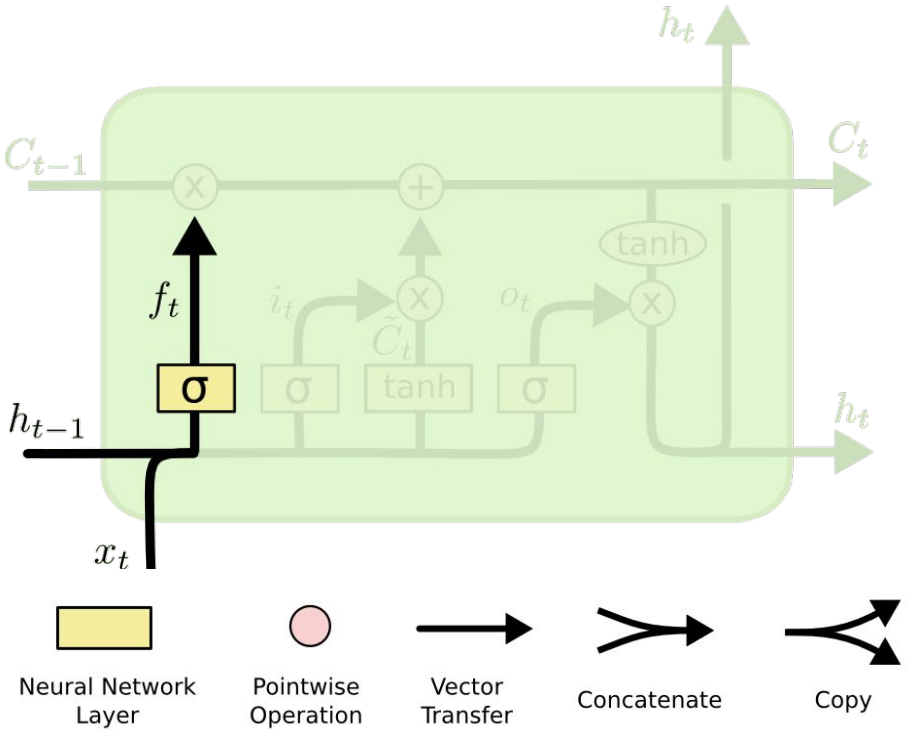


Figure: Cristopher Olah, "Understanding LSTM Networks" (2015)

Long Short-Term Memory (LSTM)

Make every RNN unit able to **forget whatever may not be useful anymore** by clearing that info from the cell state (optimized clearing mechanism)



Forget Gate:

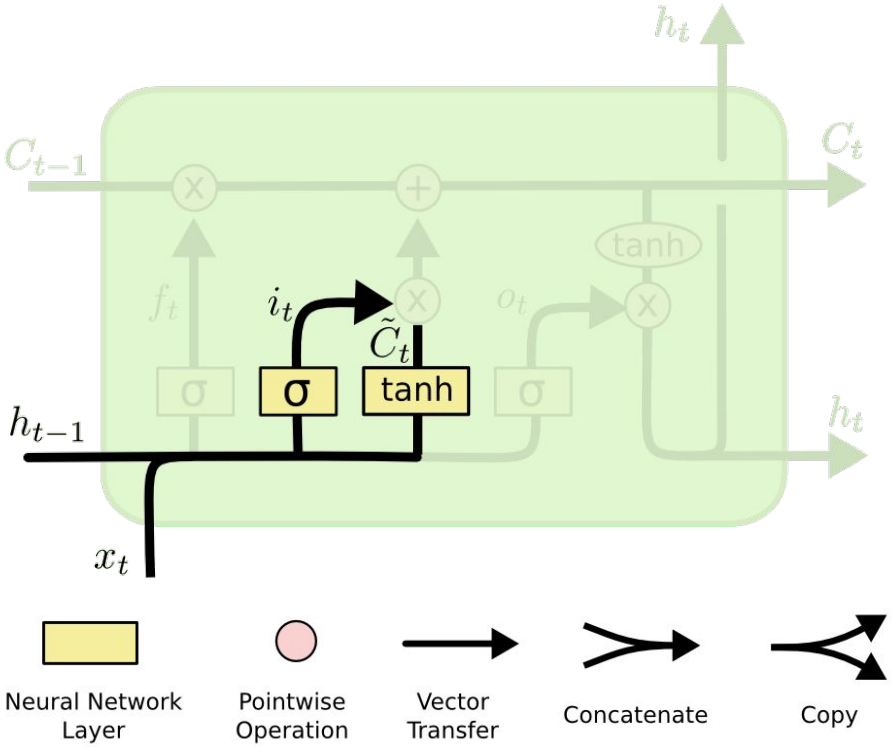
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$

Concatenate

Figure: Cristopher Olah, "Understanding LSTM Networks" (2015) / Slide: Alberto Montes

Long Short-Term Memory (LSTM)

Make every RNN unit able to decide whether **the current time-step information matters or not**, to accept or discard (optimized reading mechanism)



Input Gate Layer

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

New contribution to cell state

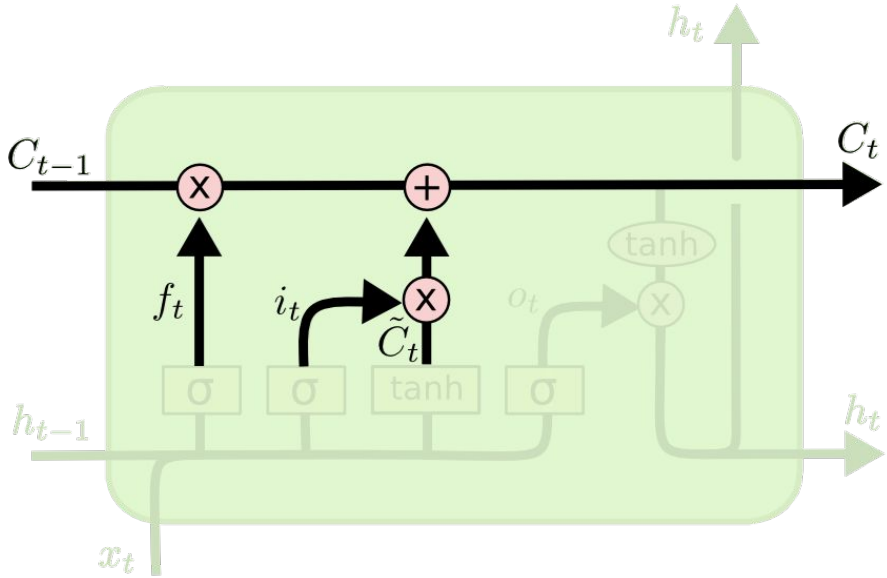
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Classic neuron

Figure: Cristopher Olah, "Understanding LSTM Networks" (2015) / Slide: Alberto Montes

Long Short-Term Memory (LSTM)

Make every RNN unit able to decide whether **the current time-step information matters or not**, to accept or discard (optimized reading mechanism)



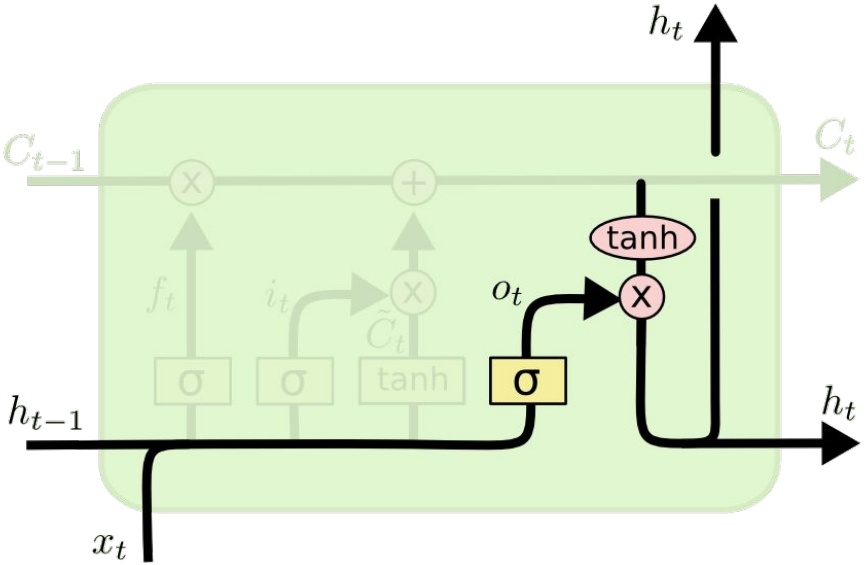
Forget + Input Gates = Update Cell State (memory):

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

Figure: Cristopher Olah, "Understanding LSTM Networks" (2015) / Slide: Alberto Montes

Long Short-Term Memory (LSTM)

Make every RNN unit able to **output the decisions whenever it is ready to do so** (optimized output mechanism)



Output Gate Layer

$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

Output to next layer & timestep

$$h_t = o_t * \tanh\left(C_t\right)$$

Figure: Cristopher Olah, "Understanding LSTM Networks" (2015) / Slide: Alberto Montes

Long Short-Term Memory (LSTM)

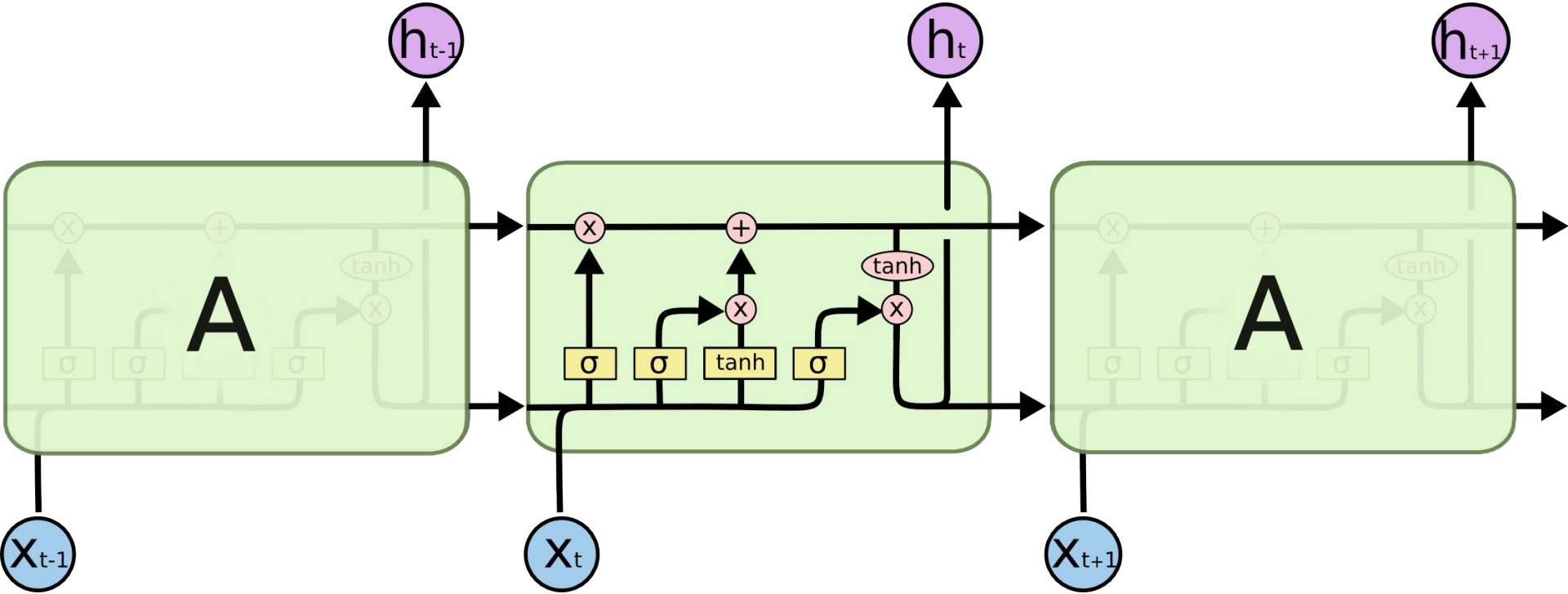
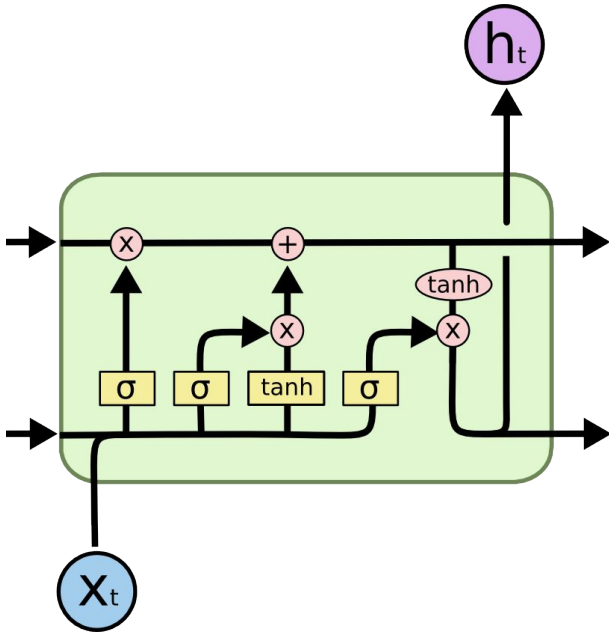


Figure: Cristopher Olah, "Understanding LSTM Networks" (2015) / Slide: Alberto Montes

Long Short Term Memory (LSTM) cell



Compared to a non-gated RNN, an LSTM has four times more parameters because of the additional neurons that govern the gates:

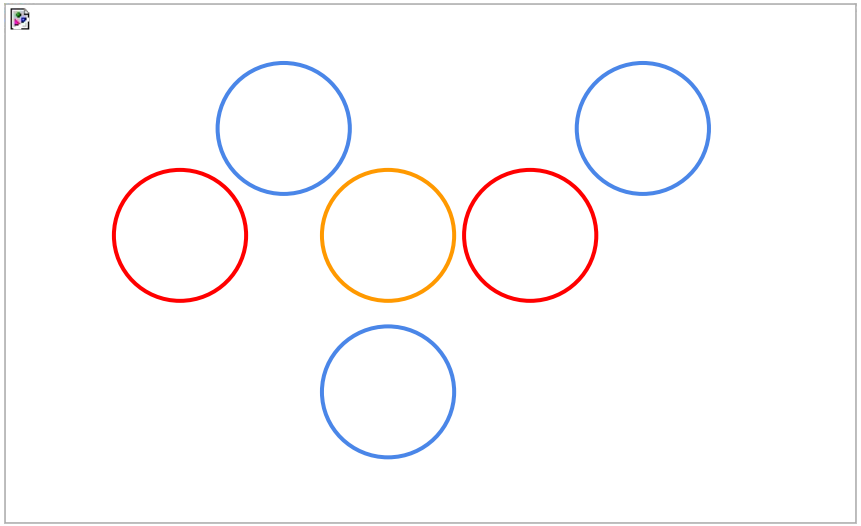
$$N^i_{params} = 4 \times (N^i_{inputs} \times N^i_{units} + N^i_{units} \times N^i_{units} + N^i_{units})$$

3 gates + input activation

Long Short Term Memory (LSTM) cell

Updating an LSTM cell requires 6 computations:

1. Gates
2. Activation units
3. Cell state



$$\mathbf{i}_t = \sigma(\mathbf{W}_i\mathbf{x}_t + \mathbf{U}_i\mathbf{h}_{t-1} + \mathbf{b}_i)$$

$$\hat{\mathbf{C}}_t = \tanh(\mathbf{W}_c\mathbf{x}_t + \mathbf{U}_c\mathbf{h}_{t-1} + \mathbf{b}_c)$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_f\mathbf{x}_t + \mathbf{U}_f\mathbf{h}_{t-1} + \mathbf{b}_f)$$

$$\mathbf{C}_t = \mathbf{i}_t \odot \hat{\mathbf{C}}_t + \mathbf{f}_t \odot \mathbf{C}_{t-1}$$

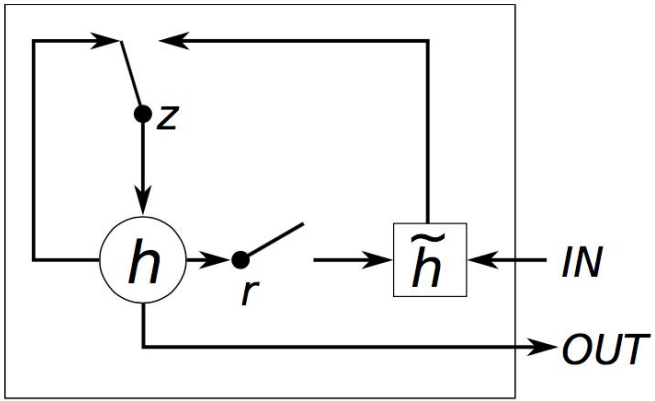
$$\mathbf{o}_t = \sigma(\mathbf{W}_o\mathbf{x}_t + \mathbf{U}_o\mathbf{h}_{t-1} + \mathbf{b}_o)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{C}_t)$$

Computation Flow

Gated Recurrent Unit (GRU)

GRU obtain a similar performance as LSTM with one gate less.



$$u_i = \sigma\left(W^{(u)}x_i + U^{(u)}h_{i-1} + b^{(u)}\right) \quad (1)$$

$$r_i = \sigma\left(W^{(r)}x_i + U^{(r)}h_{i-1} + b^{(r)}\right) \quad (2)$$

$$\tilde{h}_i = \tanh\left(Wx_i + r_i \circ Uh_{i-1} + b^{(h)}\right) \quad (3)$$

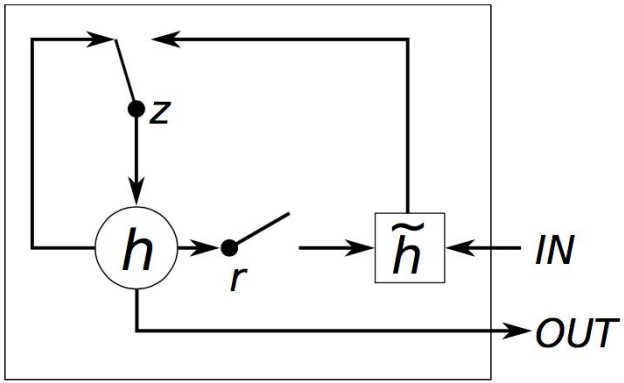
$$h_i = u_i \circ \tilde{h}_i + (1 - u_i) \circ h_{i-1} \quad (4)$$

$$N^i_{params} = 3 \times (N^i_{inputs} \times N^i_{units} + N^i_{units} \times N^i_{units} + N^i_{units})$$

Cho, Kyunghyun, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." AMNLP 2014.

Gated Recurrent Unit (GRU)

GRU obtain a similar performance as LSTM with one gate less.



$$u_i = \sigma\left(W^{(u)}x_i + U^{(u)}h_{i-1} + b^{(u)}\right) \quad (1)$$

$$r_i = \sigma\left(W^{(r)}x_i + U^{(r)}h_{i-1} + b^{(r)}\right) \quad (2)$$

$$\tilde{h}_i = \tanh\left(Wx_i + r_i \circ Uh_{i-1} + b^{(h)}\right) \quad (3)$$

$$h_i = u_i \circ \tilde{h}_i + (1 - u_i) \circ h_{i-1} \quad (4)$$

$$N^i_{params} = 3 \times (N^i_{inputs} \times N^i_{units} + N^i_{units} \times N^i_{units} + N^i_{units})$$

Cho, Kyunghyun, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." AMNLP 2014.

Recurrent Neural Network (RNN)

Deep Learning TV, [“Recurrent Neural Networks - Ep. 9”]

Outline

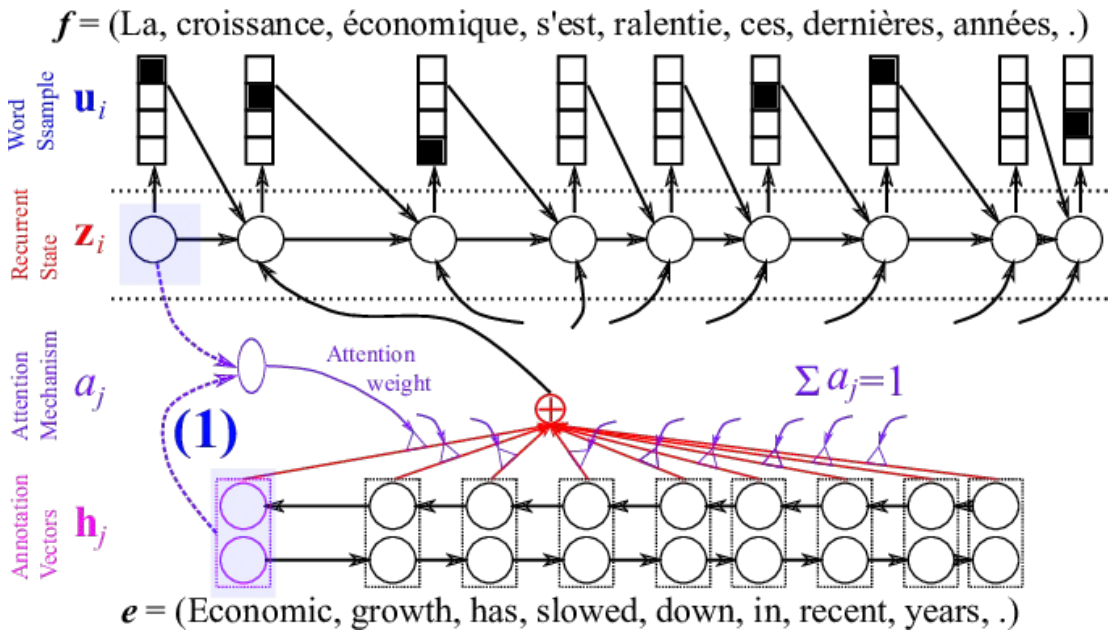
1. Deep Convolutional Networks
2. Recurrent Neural Networks
3. **Attention Models**

Attention Models

Attend to different parts of the input to optimize a certain output

Attention Mechanism

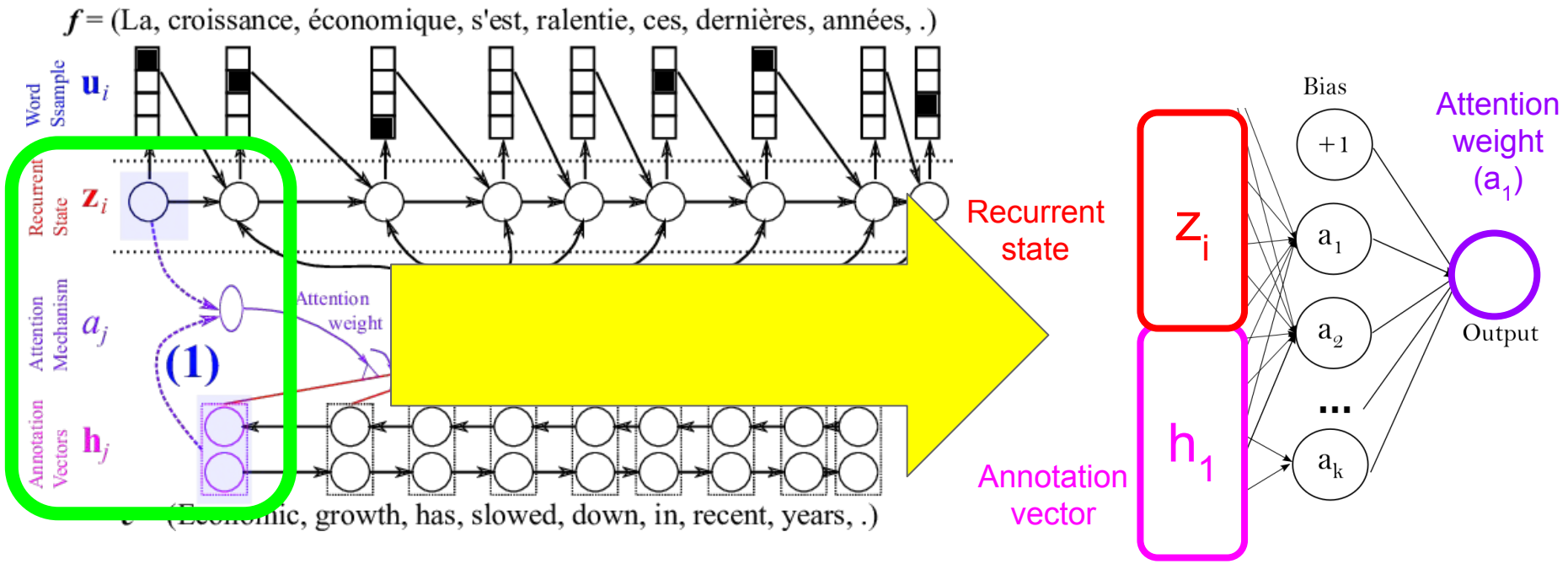
The vector to be fed to the RNN at each timestep is a weighted sum of all the annotation vectors.



Kyunghyun Cho, [“Introduction to Neural Machine Translation with GPUs”](#) (2015)

Attention Mechanism

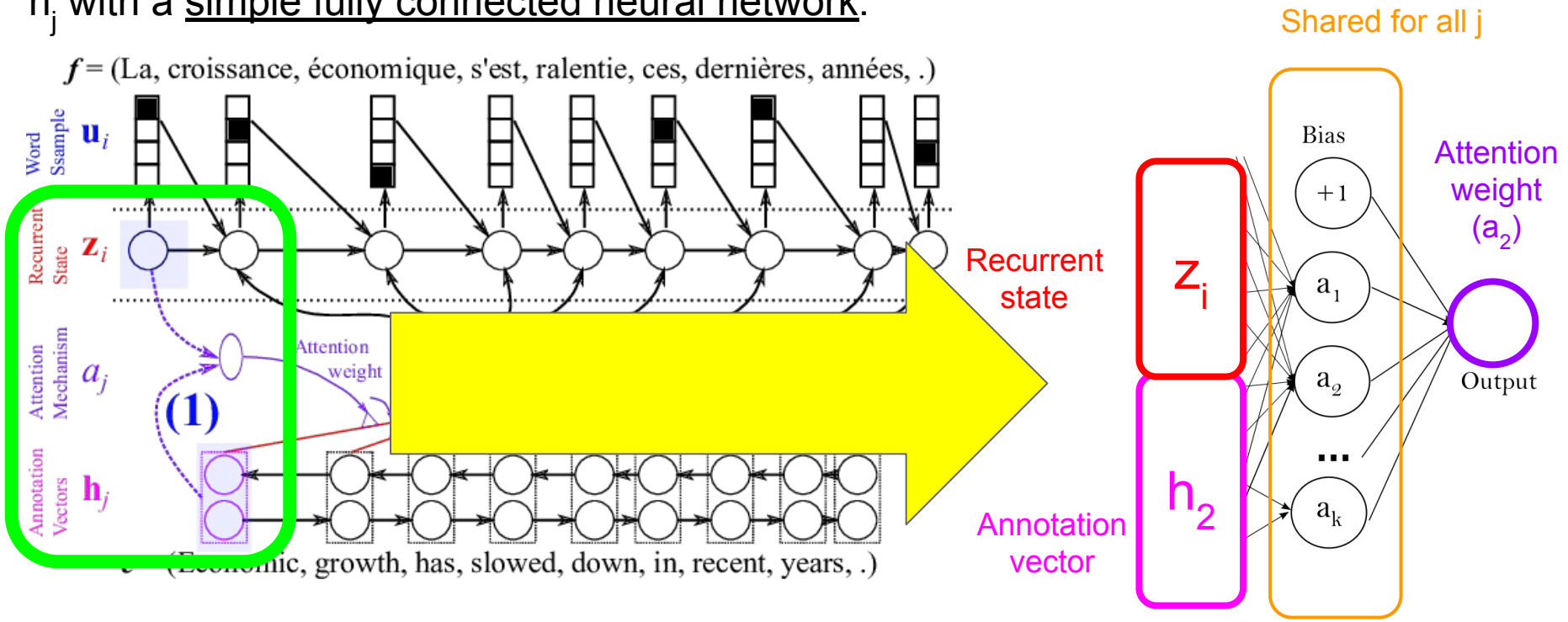
An attention weight (scalar) is predicted at each time-step for each annotation vector h_j with a simple fully connected neural network.



Kyunghyun Cho, "Introduction to Neural Machine Translation with GPUs" (2015)

Attention Mechanism

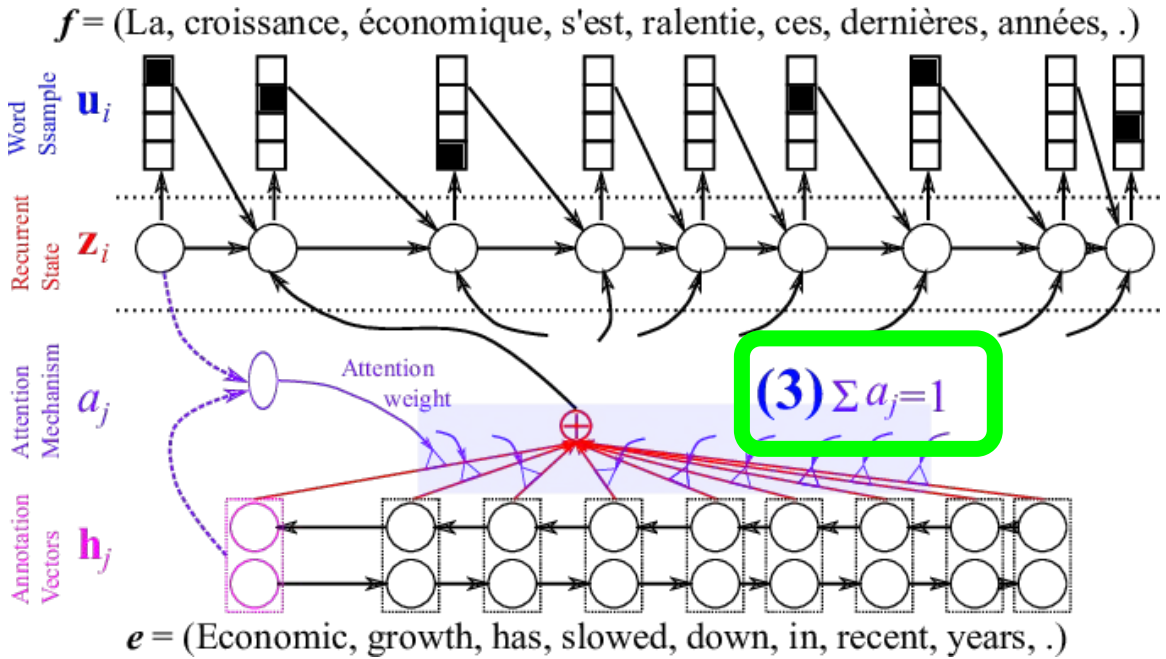
An attention weight (scalar) is predicted at each time-step for each annotation vector h_j with a simple fully connected neural network.



Kyunghyun Cho, "Introduction to Neural Machine Translation with GPUs" (2015)

Attention Mechanism

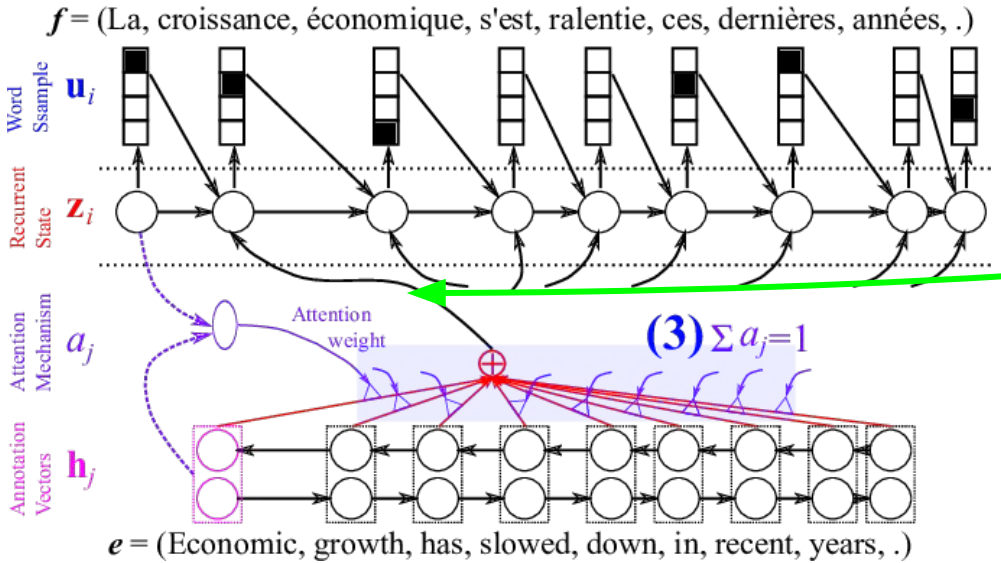
Once a relevance score (weight) is estimated for each word, they are normalized with a softmax function so they sum up to 1.



Kyunghyun Cho, "Introduction to Neural Machine Translation with GPUs" (2015)

Attention Mechanism

Finally, a context-aware representation c_{i+1} for the output word at timestep i can be defined as:

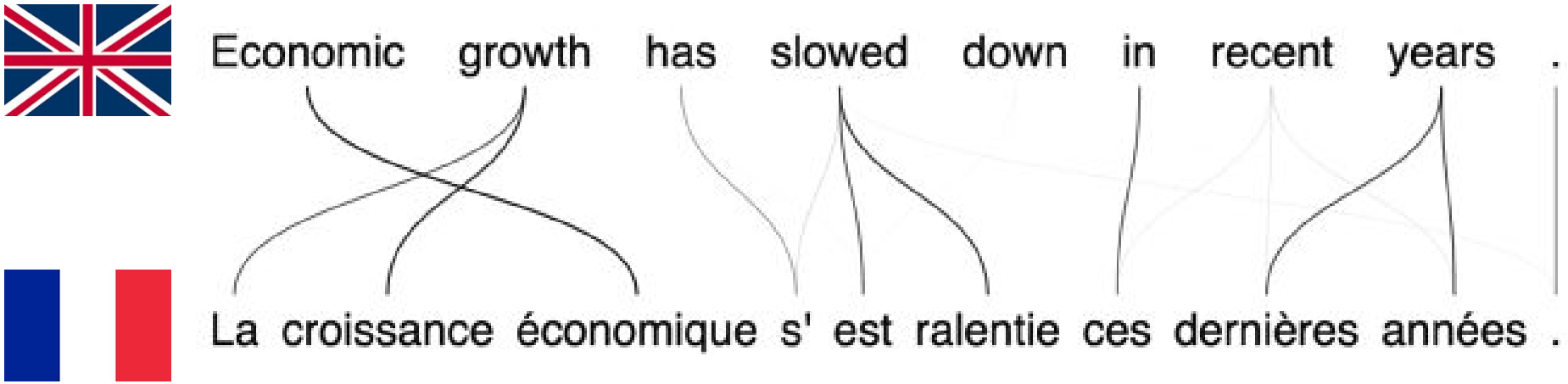


$$c_{i+1} = \sum_{j=1}^{T} \alpha_j h_j$$

Kyunghyun Cho, "Introduction to Neural Machine Translation with GPUs" (2015)

Attention Mechanism

The model automatically finds the correspondence structure between two languages (alignment).



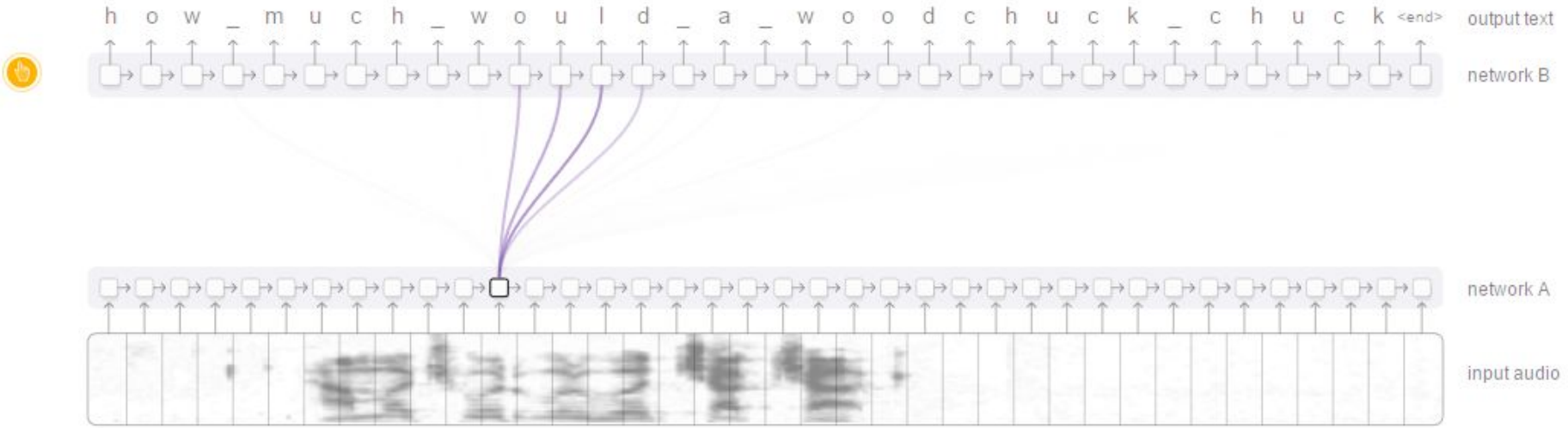
(Edge thicknesses represent the attention weights found by the attention model)

Kyunghyun Cho, "Introduction to Neural Machine Translation with GPUs" (2015)

Attention Models

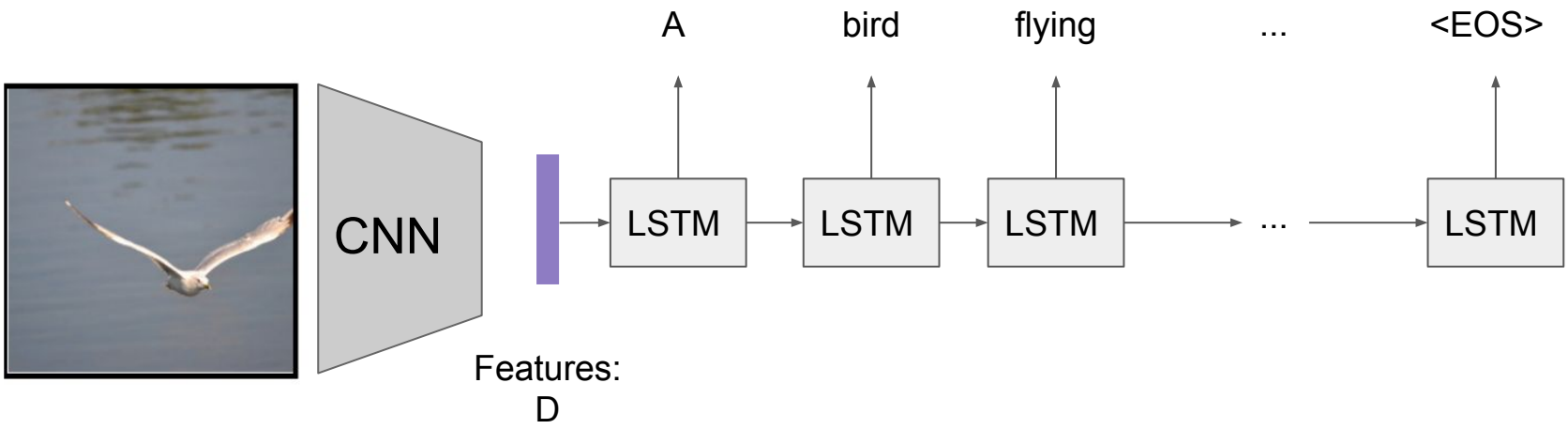
Attend to different parts of the input to optimize a certain output

Input: Audio features; Output: Text



Chan et al. Listen, Attend and Spell. ICASSP 2016
Source: distill.pub

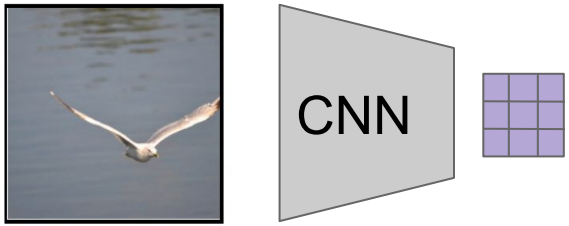
LSTM Decoder for Image Captioning



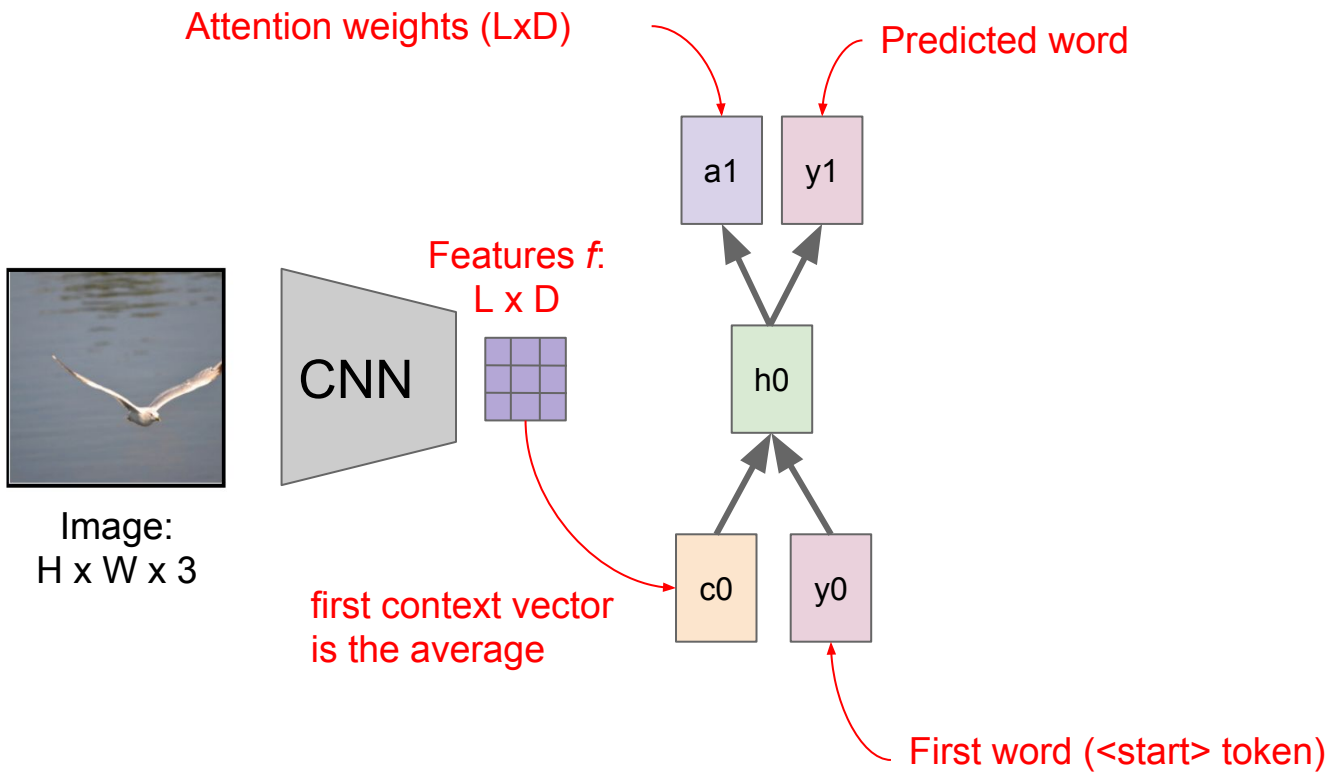
Limitation: All output predictions are based on the **final and static** output of the encoder

Vinyals et al. Show and tell: A neural image caption generator. CVPR 2015

Attention for Image Captioning



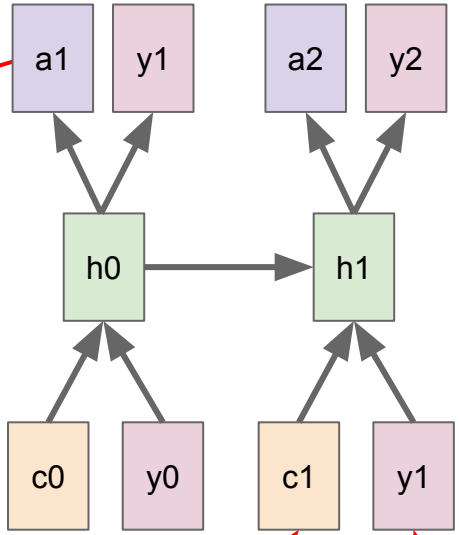
Attention for Image Captioning



Attention for Image Captioning

Image:
H x W x 3

CNN



$$c_i = \sum_{j=1}^{LD} f_j a_{ij}$$

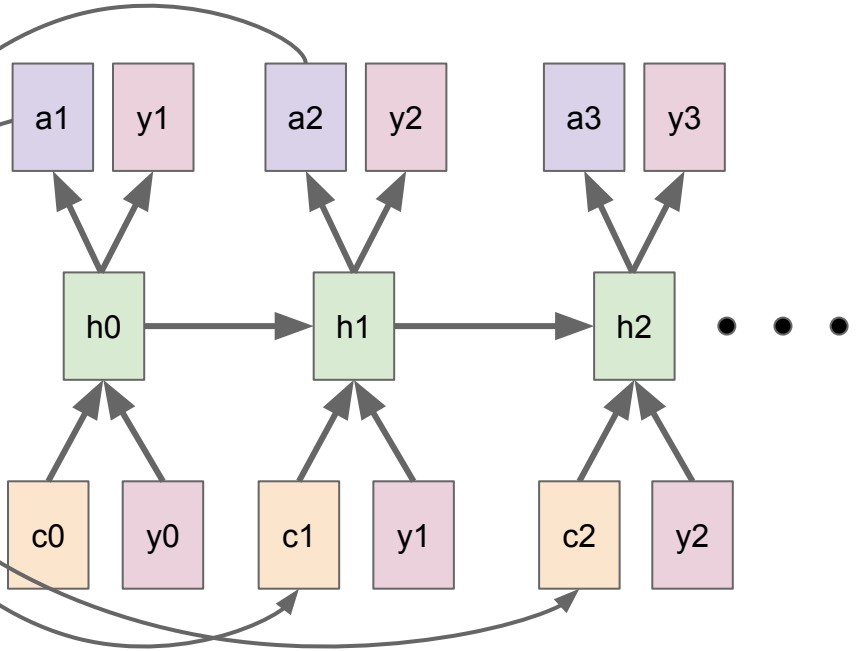
Visual features weighted with attention give the next context vector

Predicted word in previous timestep

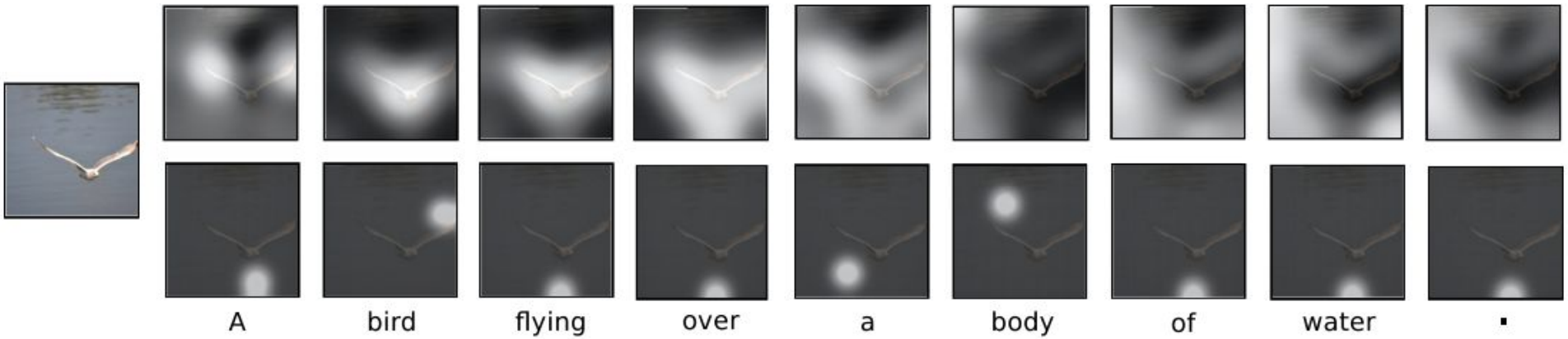
Attention for Image Captioning

Image:
H x W x 3

CNN



Attention for Image Captioning



Xu et al. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. ICML 2015

Attention for Image Captioning

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

A little girl sitting on a bed with a teddy bear.

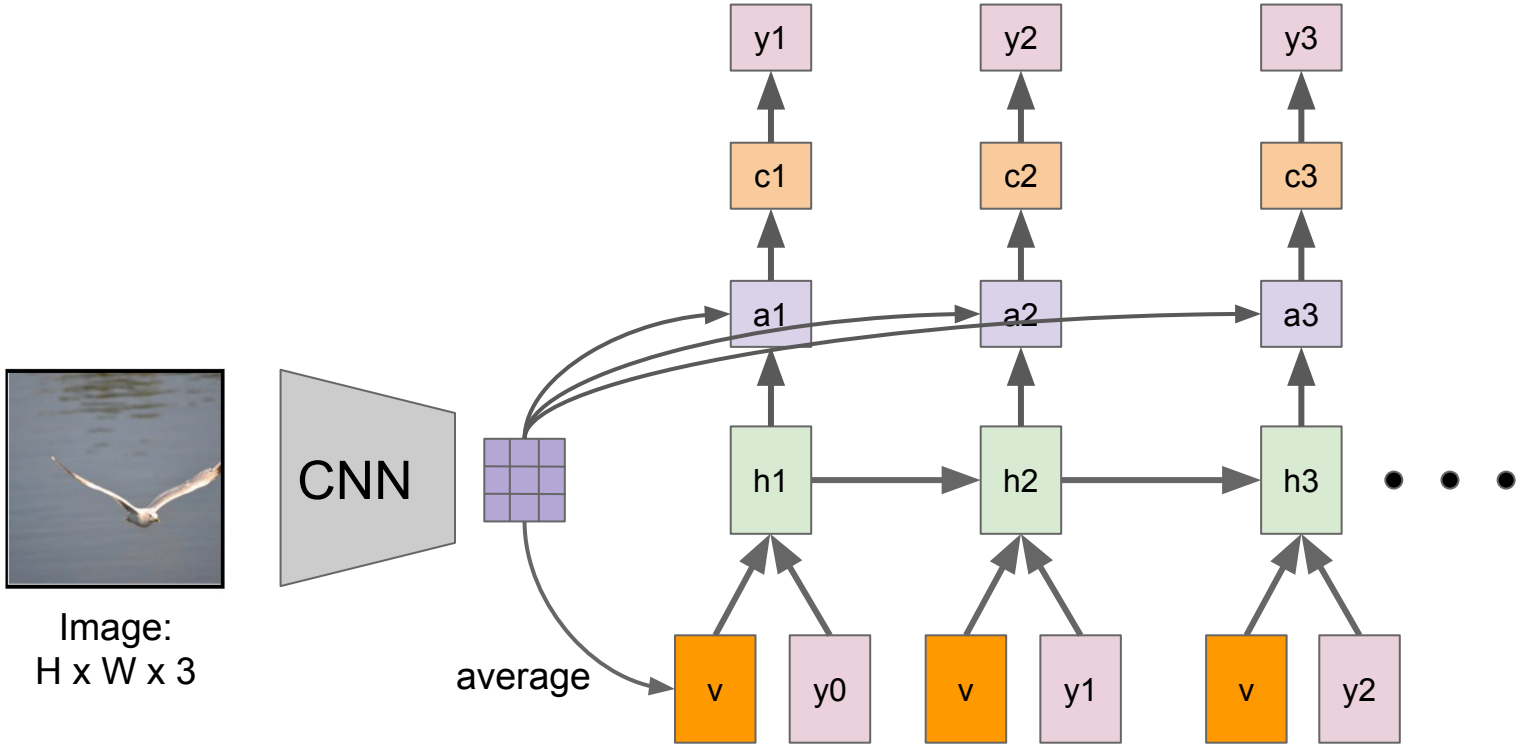
A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

Xu et al. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. ICML 2015

Attention for Image Captioning

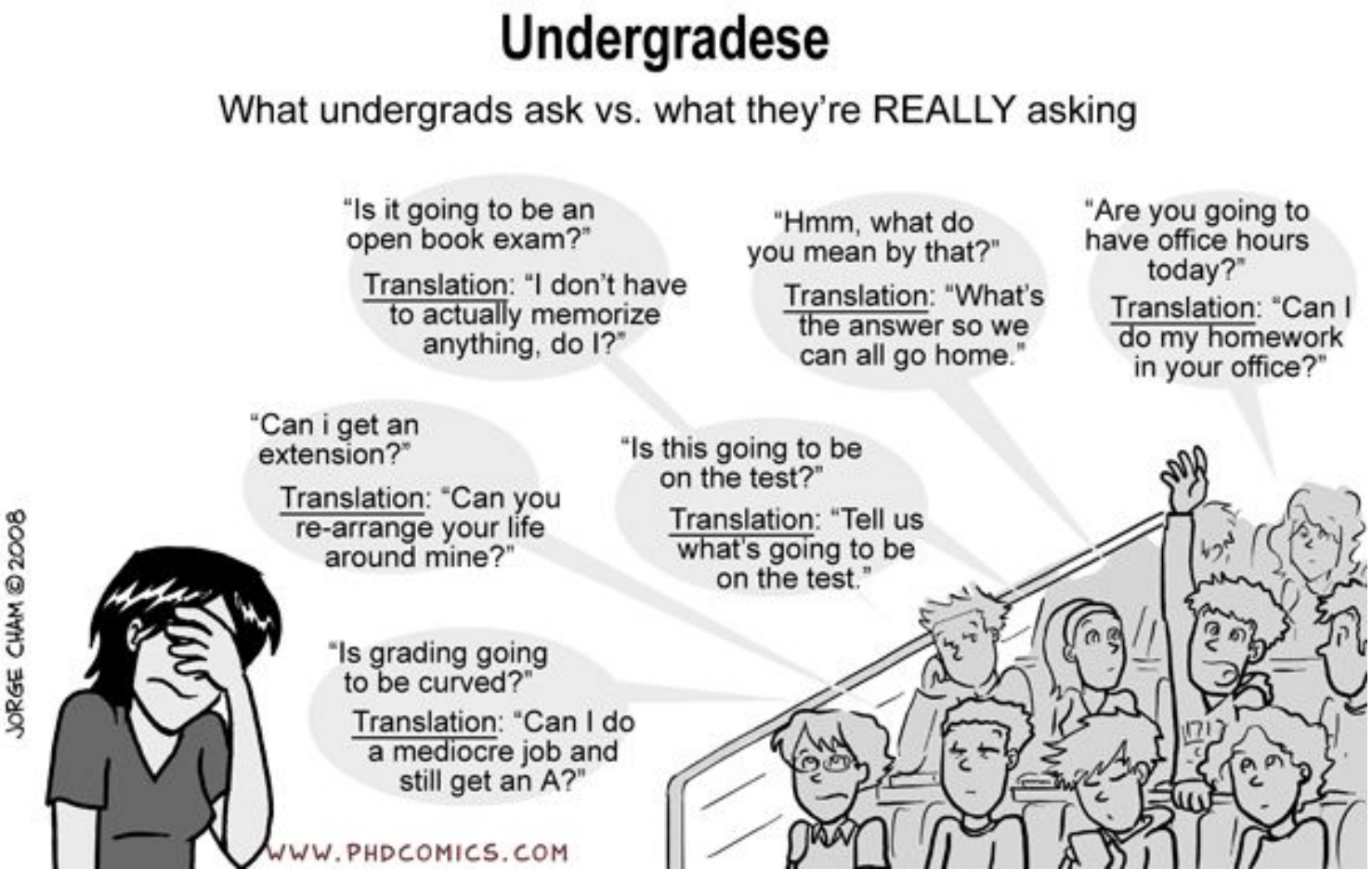
Side-note: attention can be computed with previous or **current** hidden state



Outline

1. Deep Convolutional Networks
2. Recurrent Neural Networks
3. Attention Models

Questions ?


Undergradese
What undergrads ask vs. what they're REALLY asking
"Is it going to be an open book exam?"
Translation: "I don't have to actually memorize anything, do I?"
"Hmm, what do you mean by that?"
Translation: "What's the answer so we can all go home."
"Are you going to have office hours today?"
Translation: "Can I do my homework in your office?"
"Can i get an extension?"
Translation: "Can you re-arrange your life around mine?"
"Is this going to be on the test?"
Translation: "Tell us what's going to be on the test."
"Is grading going to be curved?"
Translation: "Can I do a mediocre job and still get an A?"
JORGE CHAM © 2008
WWW.PHDCOMICS.COM