



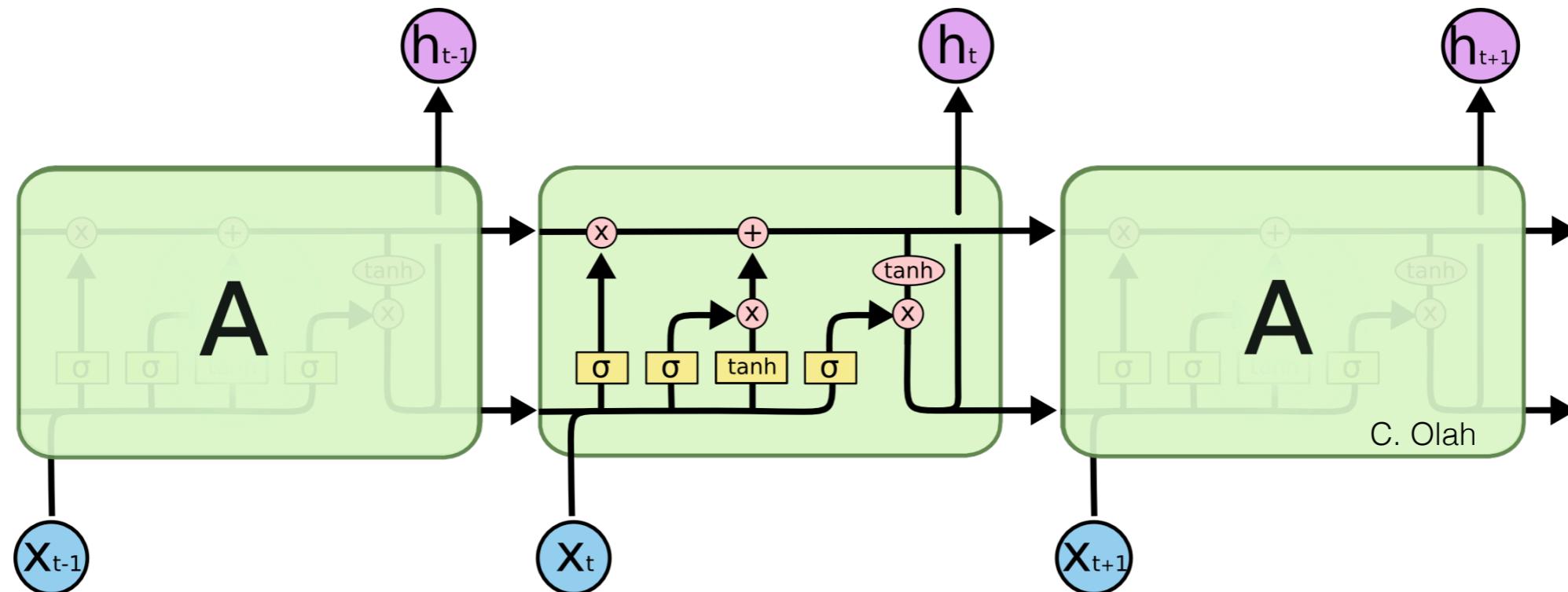
# UNIVERSIDAD DEL ROSARIO

# CENTER FOR ASTROPHYSICS

HARVARD & SMITHSONIAN

# Recurrent Neural Networks (RNNs)

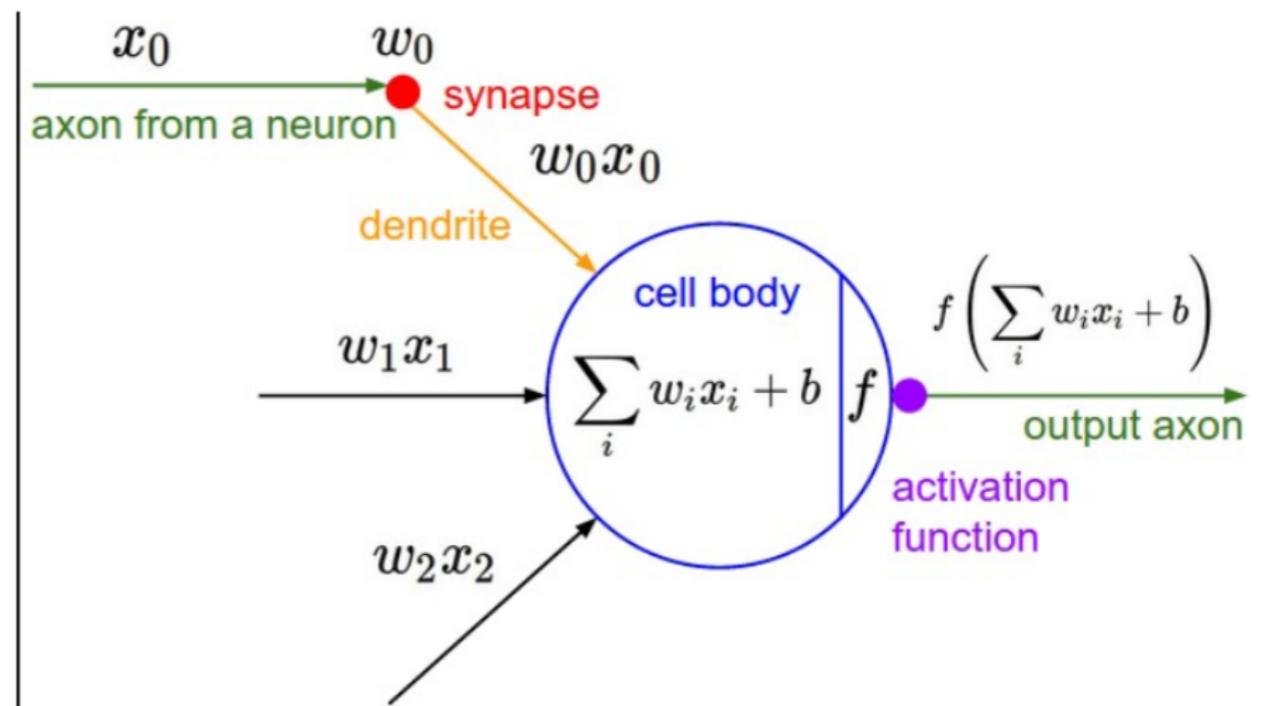
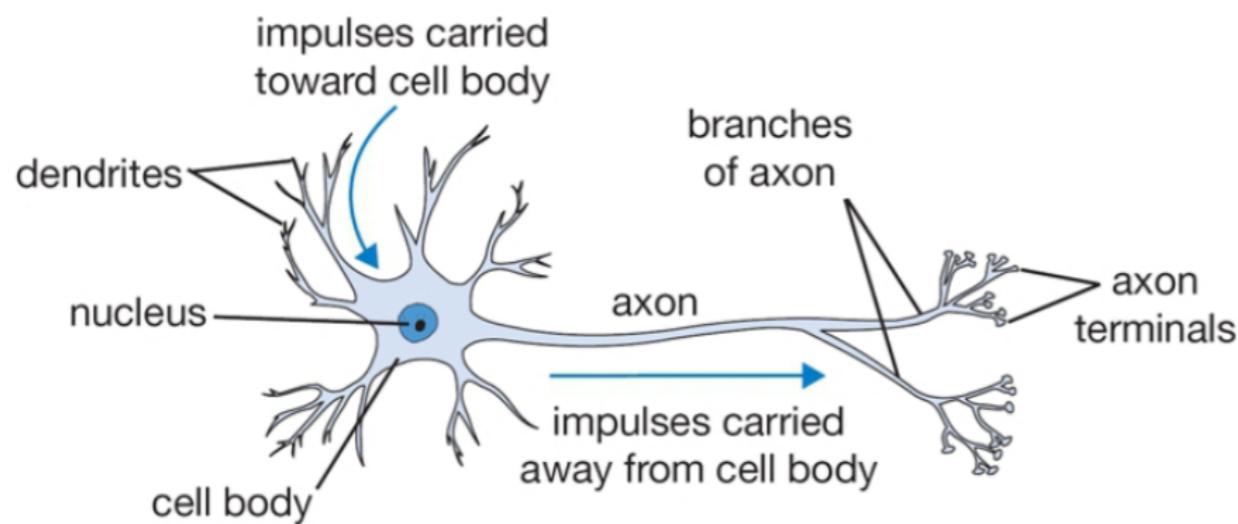
Rafael Martínez-Galarza



BigDataCo, December 12, 2020

# Feedforward Networks

# A simple perceptron



- A non-linear activation function is applied to the linear combination of the features.
- The loss (error) function that we want to minimize is minus the logarithm of the joint probability of the labels observed in the training set:

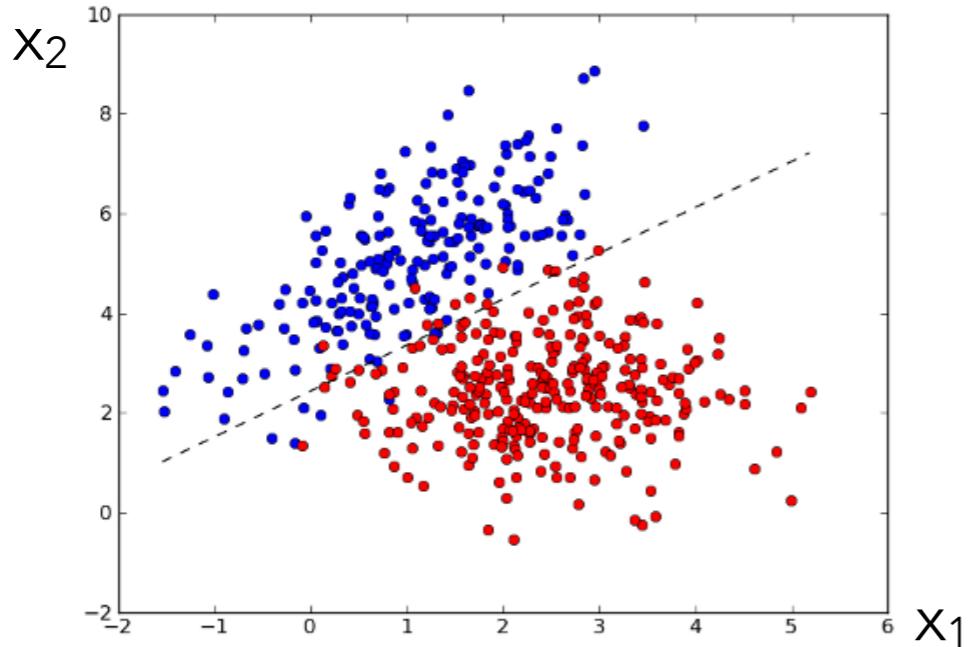
$$J(\theta) = - \sum_i (y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})))$$

# Logistic regression as a classifier



1 0

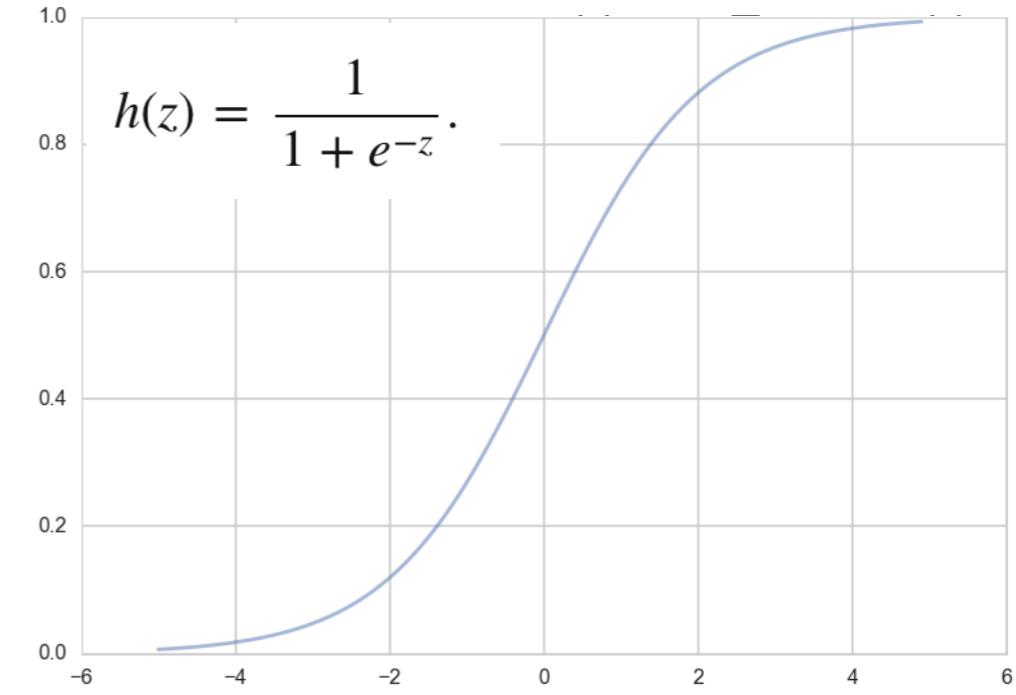
- Simple idea: find a line (surface) that separates (two) classes in the feature space.



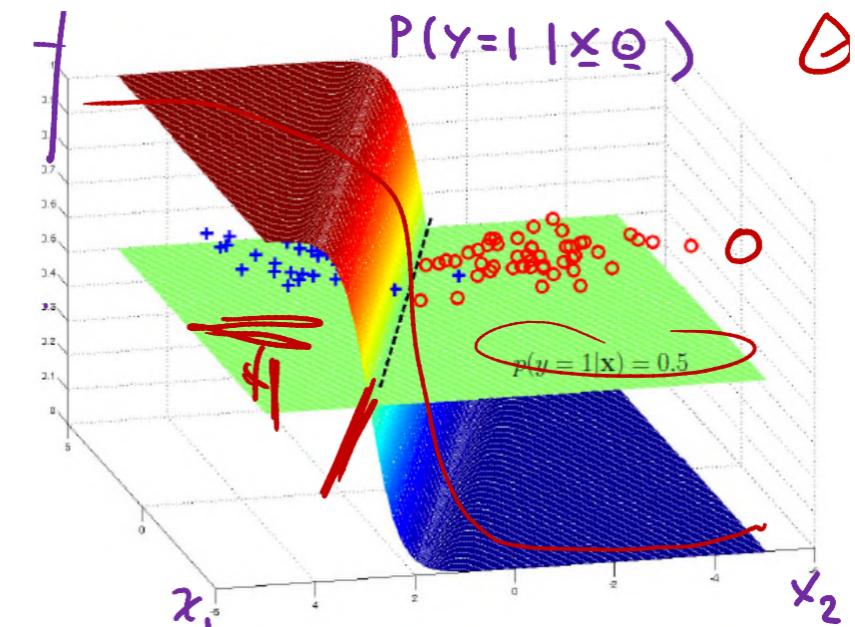
$$P(y|\mathbf{x}, \mathbf{w}) = h(\mathbf{w} \cdot \mathbf{x})^y (1 - h(\mathbf{w} \cdot \mathbf{x}))^{(1-y)}$$

$$P(y|\mathbf{x}, \mathbf{w}) = \prod_{y_i \in \mathcal{D}} h(\mathbf{w} \cdot \mathbf{x}_i)^{y_i} (1 - h(\mathbf{w} \cdot \mathbf{x}_i))^{(1-y_i)}$$

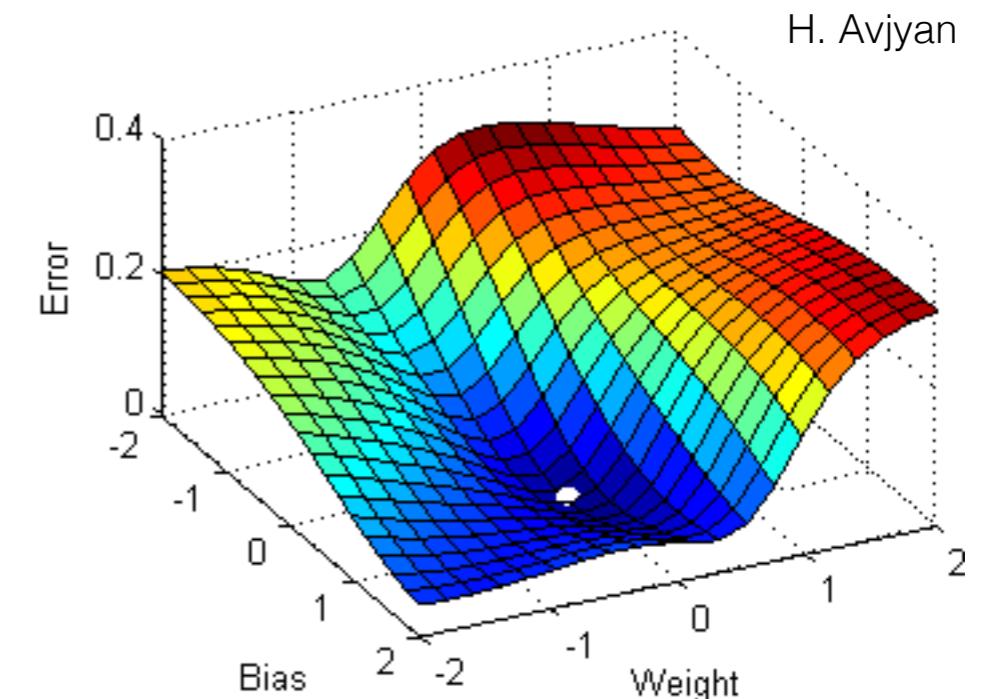
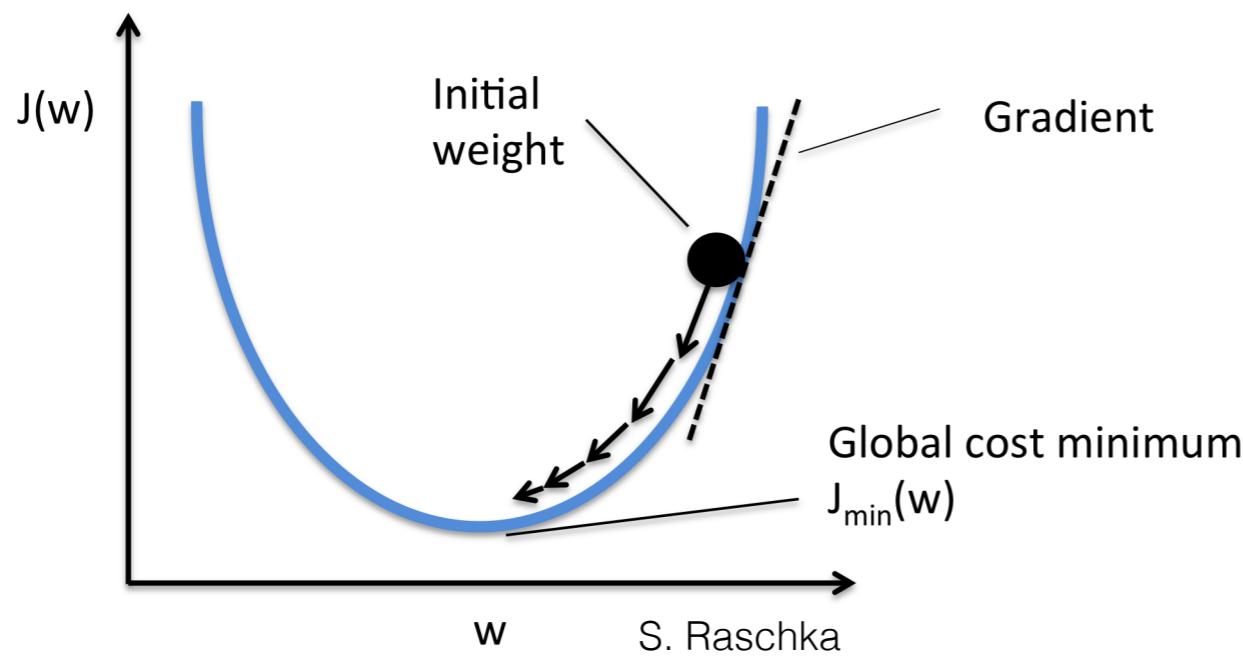
Use training set to learn the values of  $w$  that give  $P(y|\mathbf{x}, w)$  large when the object is blue and  $P(y|\mathbf{x}, w)$  small when the object is red.



- We can define  $z = \mathbf{w}^* \mathbf{x}$  to squash its value in the range [0,1] and interpret  $h(z)$  as the probability that one particular sample is blue.



# Gradient Descent



- Start with random bias and weights
- Use gradient of error function w.r.t. the bias and weights at each point to update those values.
- Iterate until certain convergence criterium is met.

$$\vec{w}_{i+1} = \vec{w}_i - \alpha \frac{\partial E(X)}{\partial \vec{w}_i}$$

$$b_{i+1} = b_i - \alpha \frac{\partial E(X)}{\partial b_i}$$

Learning rate!

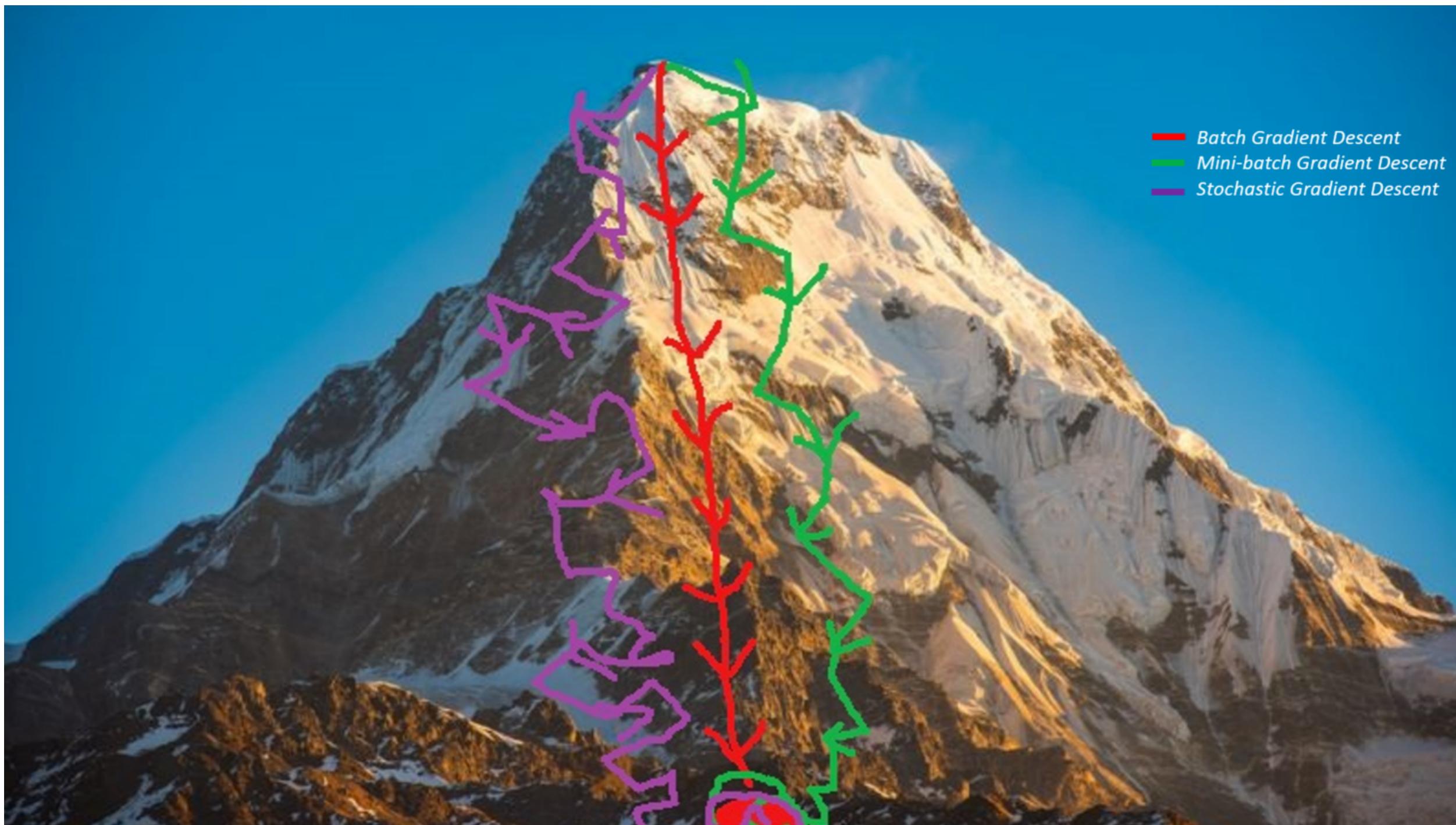
# Parenthesis: Stochastic GD

- Remember that the error function is a sum over all examples in the training set:

$$J(\theta) = - \sum_i (y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})))$$

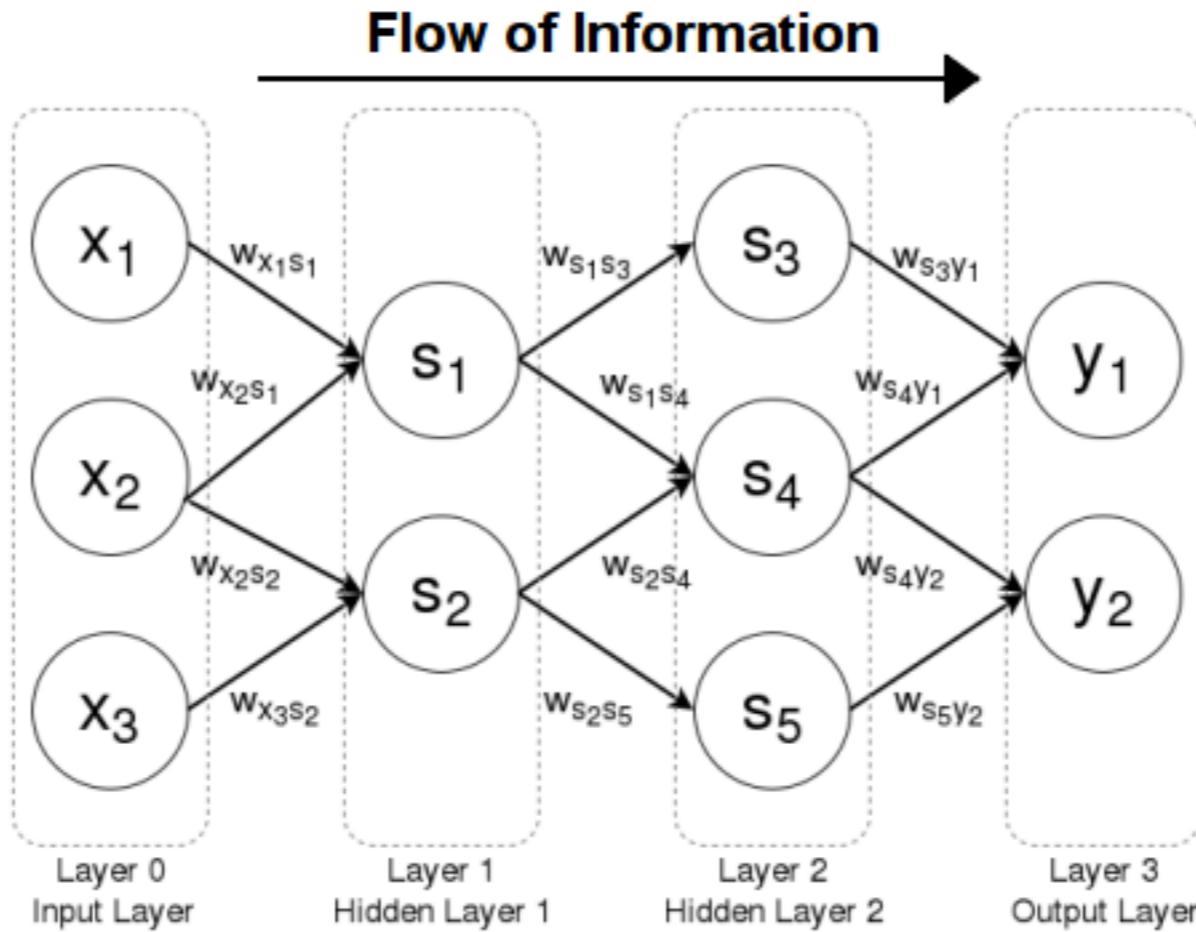
- For a single update of weights, I need to go over the entire data set. Not efficient.
- Solution 1: approximate the gradient using a single random example. Not too accurate.
- Solution 2: Perform an update using a mini-batch of examples.

# Parenthesis: Stochastic GD



I. Dabbura

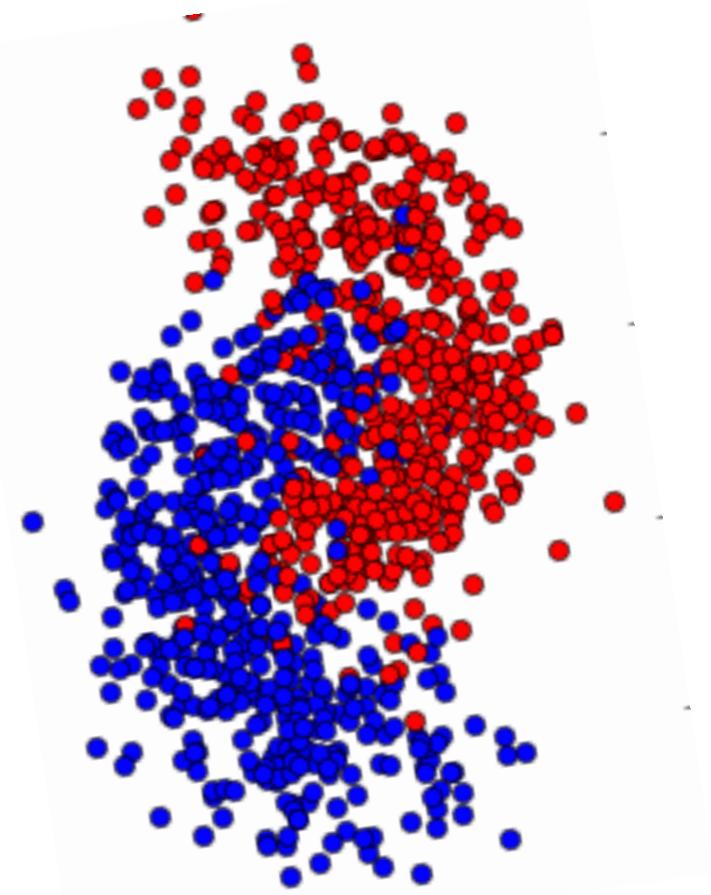
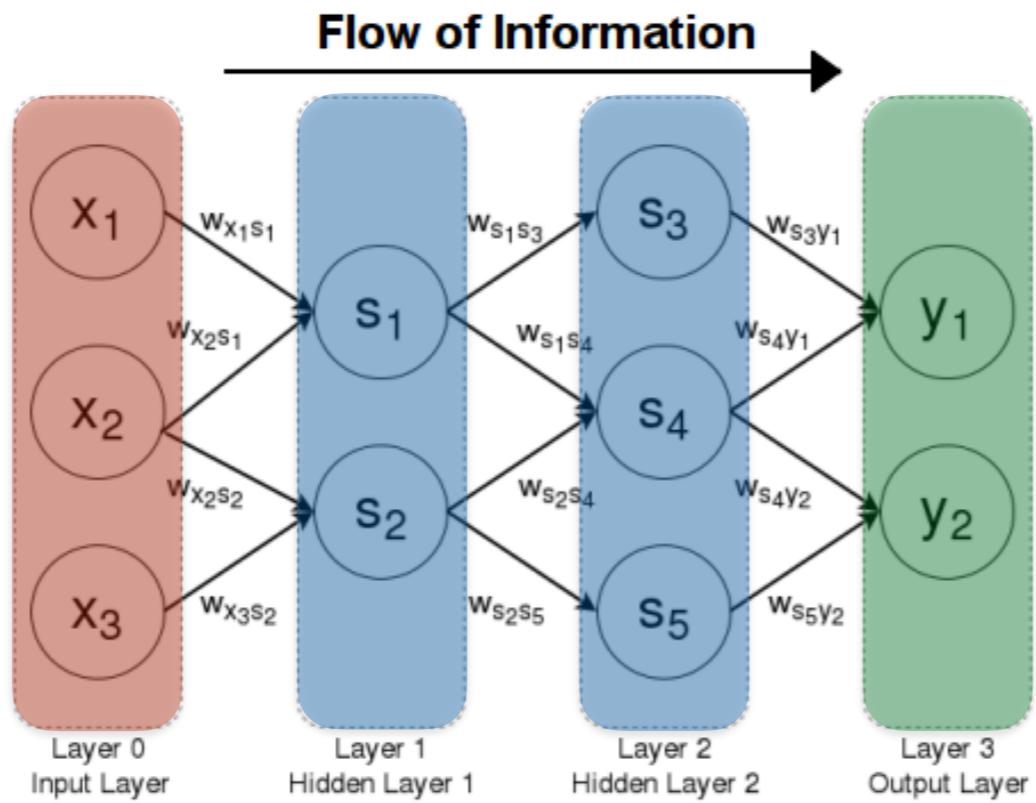
# More in general



Feedforward: connections do not form a cycle

Computes a function  $f$  on fixed size input  $x$  such that  $f(x) \sim y$  for training pairs  $(x, y)$

# Multi-layer perceptron



- ANN composed on many perceptrons.
- Capable of learning non-linearly separable functions.
- Good for regression and classification in supervised learning.
- The MLP is organized in layers: **an input layer**, **some hidden layers**, and an **output layer**.
- Regression: single neuron in the output layer
- Classification: more than one neuron in output layer
- The MLP above is said to be **FULLY CONNECTED** (all neurons in a given layer are connected to all neurons in the next)

# Convolutional Neural Networks

# Finding structure in images



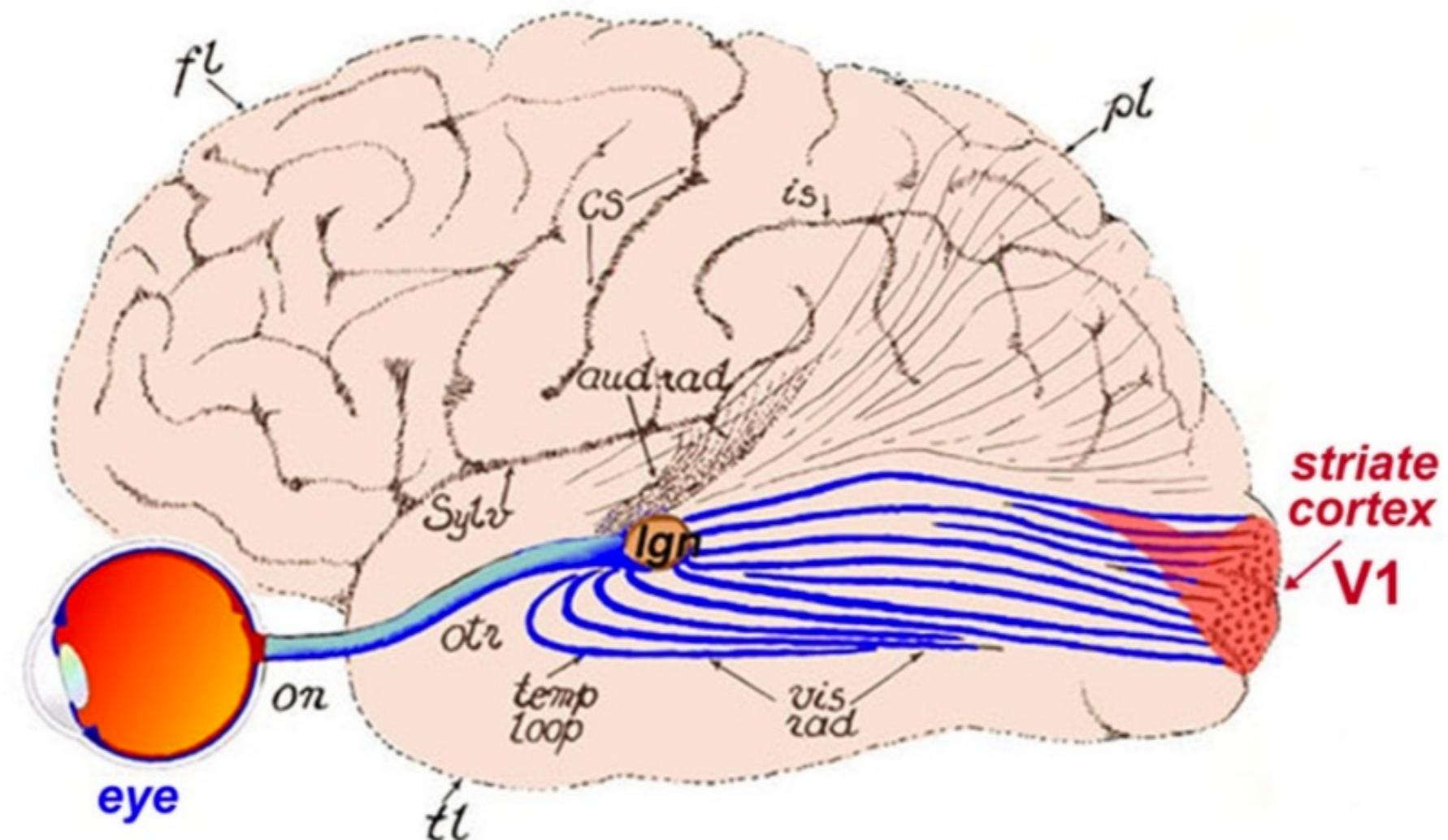
Nearby pixels are more strongly related than distant ones.

Objects are built up out of smaller parts.

A fully connected network would be very inefficient at learning patterns here.

Instead, we can learn from nature.

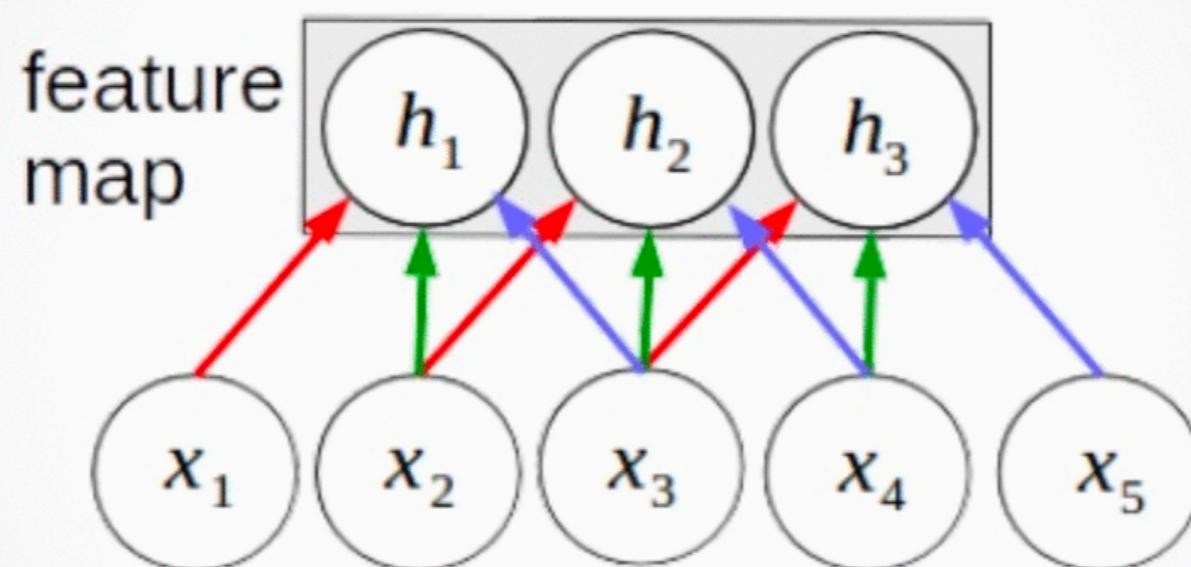
# Vision in nature



Cell in the visual cortex are not sensitive to the entire visual field  
First neurons connected to the retina specialize in detecting edges  
Neurons in further layers identify other types of patterns

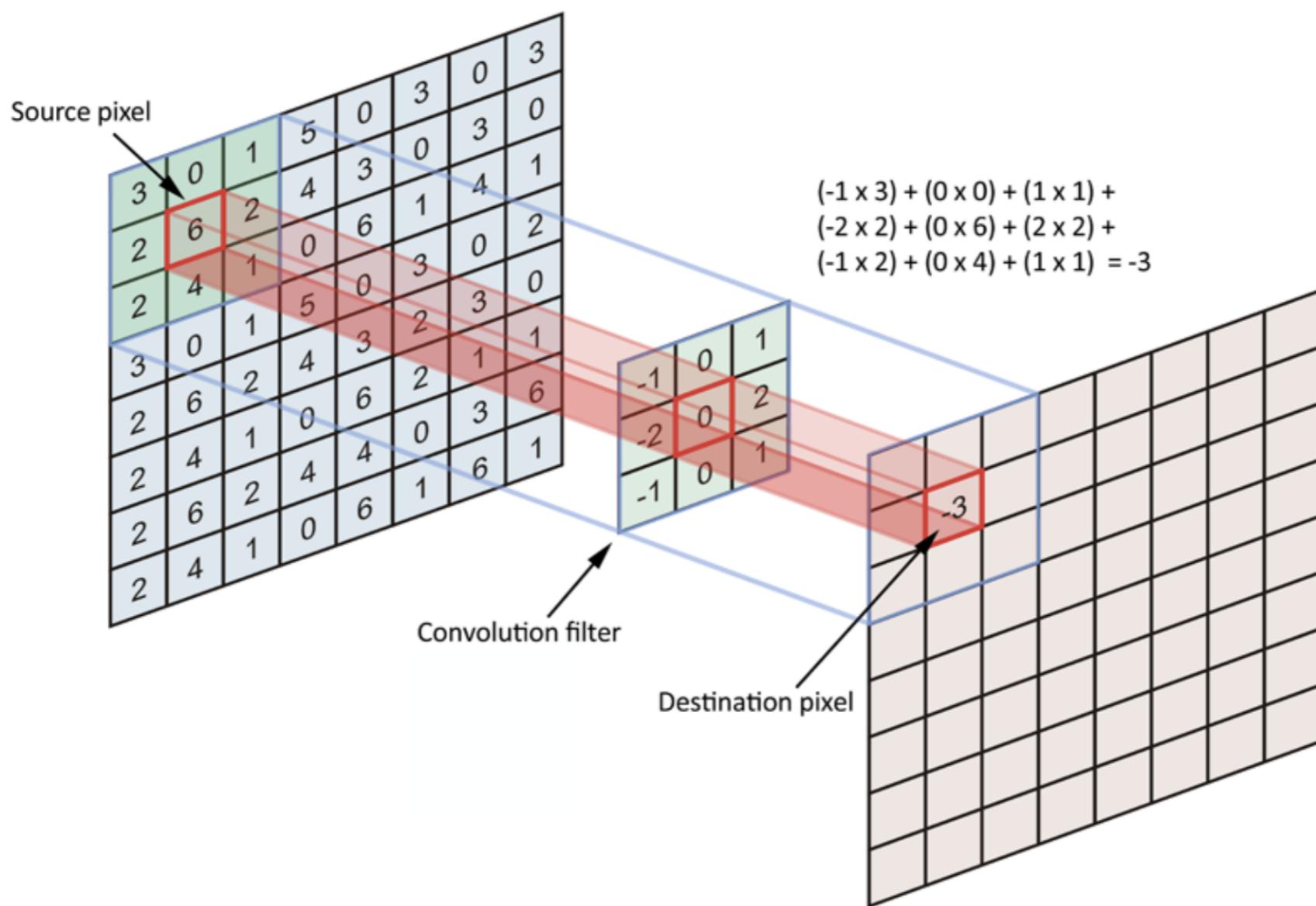
Not fully connected

# 1D Convolution



$$h_i = f((W * x)_i)$$

# “Convolution” operation



# Feature extraction

*Edge detection*


$$* \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} =$$


Kernel

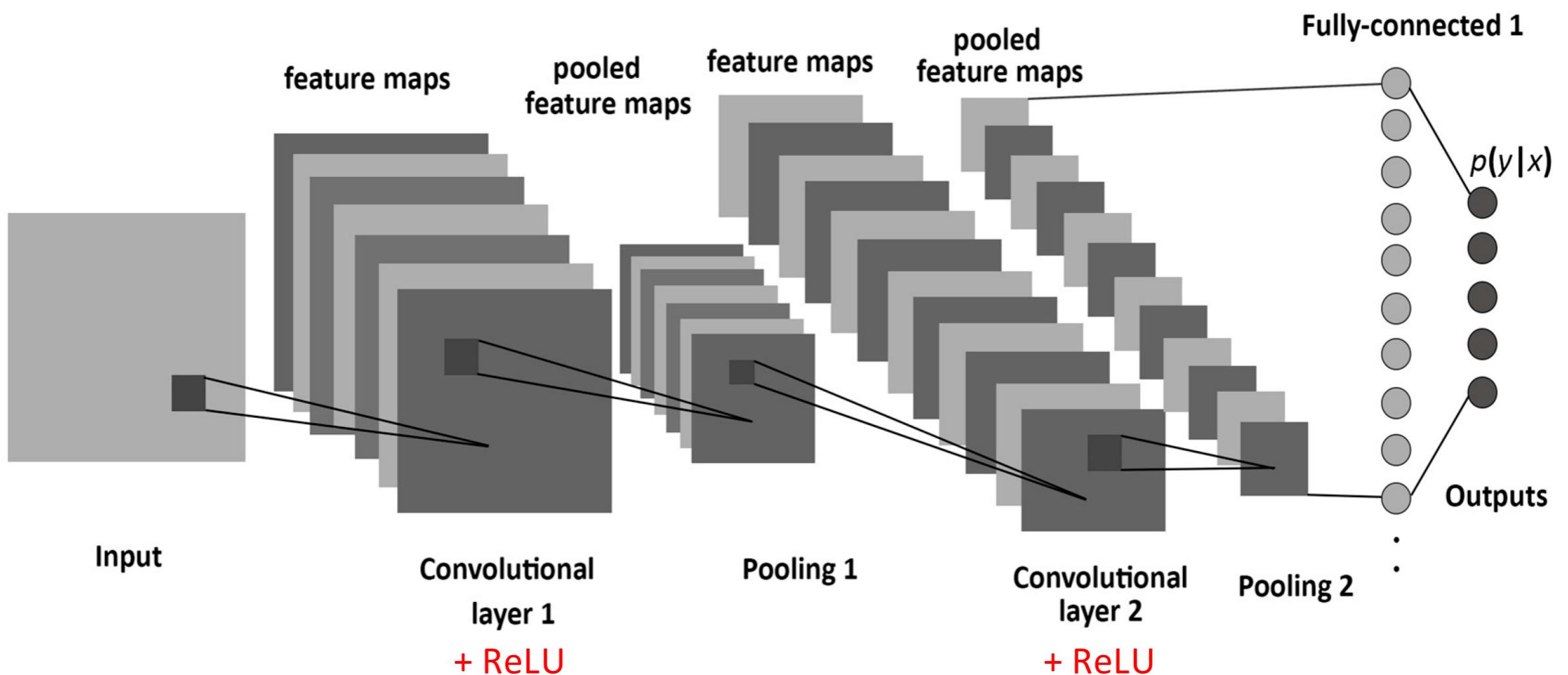
A diagram illustrating edge detection. On the left is a photograph of a deer's head. In the center is a convolution operation: the input image is multiplied by a 3x3 kernel matrix. An arrow points from the text "Kernel" to the matrix. The result of the multiplication is an output image where edges are highlighted in white against a dark background.

*Sharpen*

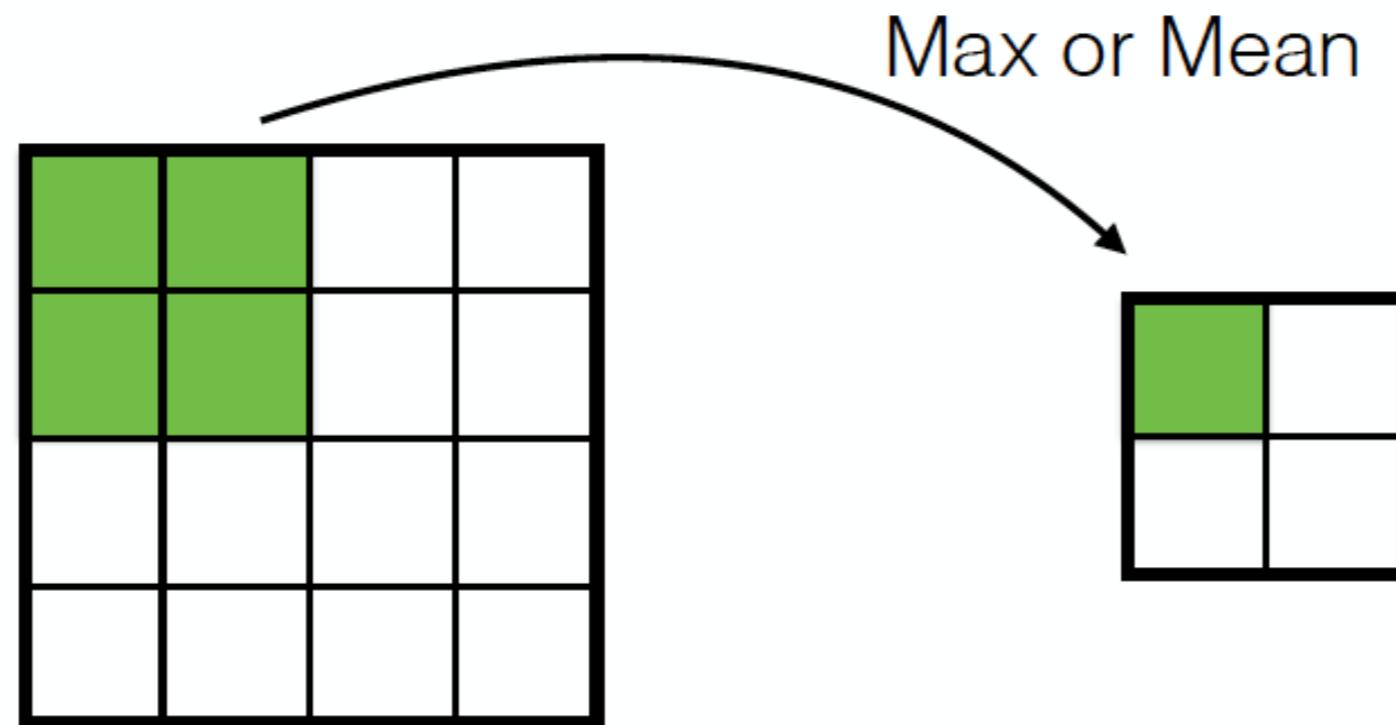

$$* \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} =$$


A diagram illustrating sharpening. On the left is a photograph of a deer's head. In the center is a convolution operation: the input image is multiplied by a 3x3 kernel matrix. The result is an output image that appears slightly more defined than the original.

# A convolutional network

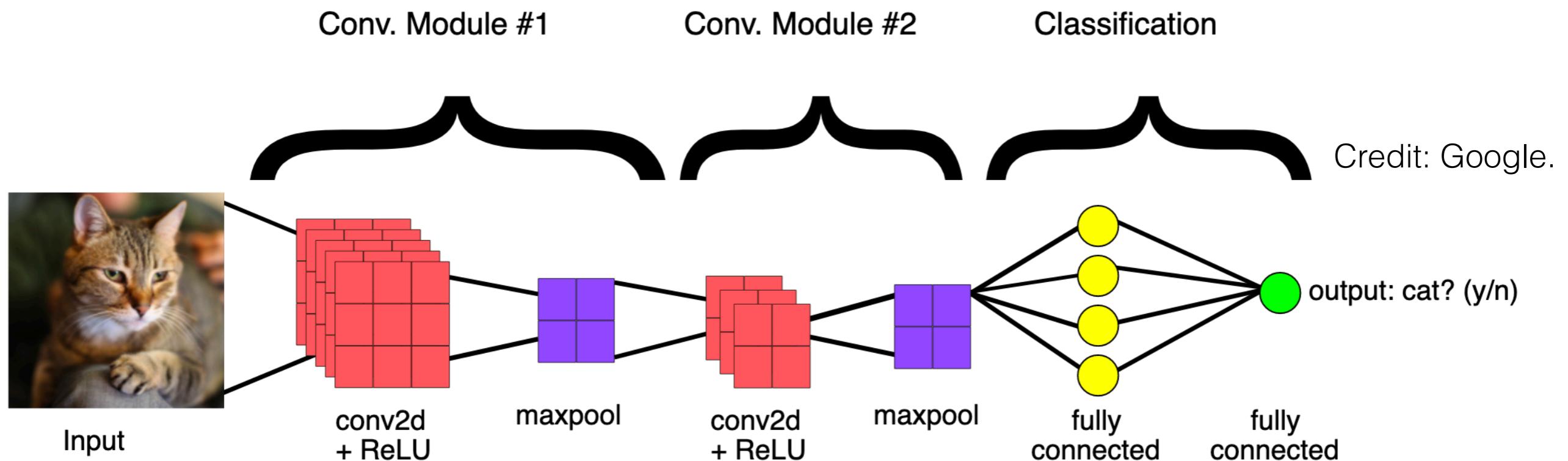


# Pooling



- Reduced input size without loosing capability to generalize.
- Invariant to small local transitions.

# Fully connected NNs and CNNs work well for classification and regression



But most interesting things, i.e., language, weather, are not static, but sequential and time-dependent.

# Recurrent Neural Networks

# The importance of persistence

I grew up in France, I speak fluent... French

# The importance of persistence

PANDARUS:

Alas, I think he shall be come approached and the day  
When little strain would be attain'd into being never fed,  
And who is but a chain and subjects of his death,  
I should not sleep.

Second Senator:

They are away this miseries, produced upon my soul,  
Breaking and strongly should be buried, when I perish  
The earth and thoughts of many states.

DUKE VINCENTIO:

Well, your wit is in the care of side and that.

Second Lord:

They would be ruled after this chamber, and  
my fair nues begun out of the fact, to be conveyed,  
Whose noble souls I'll have the heart of the wars.

Clown:

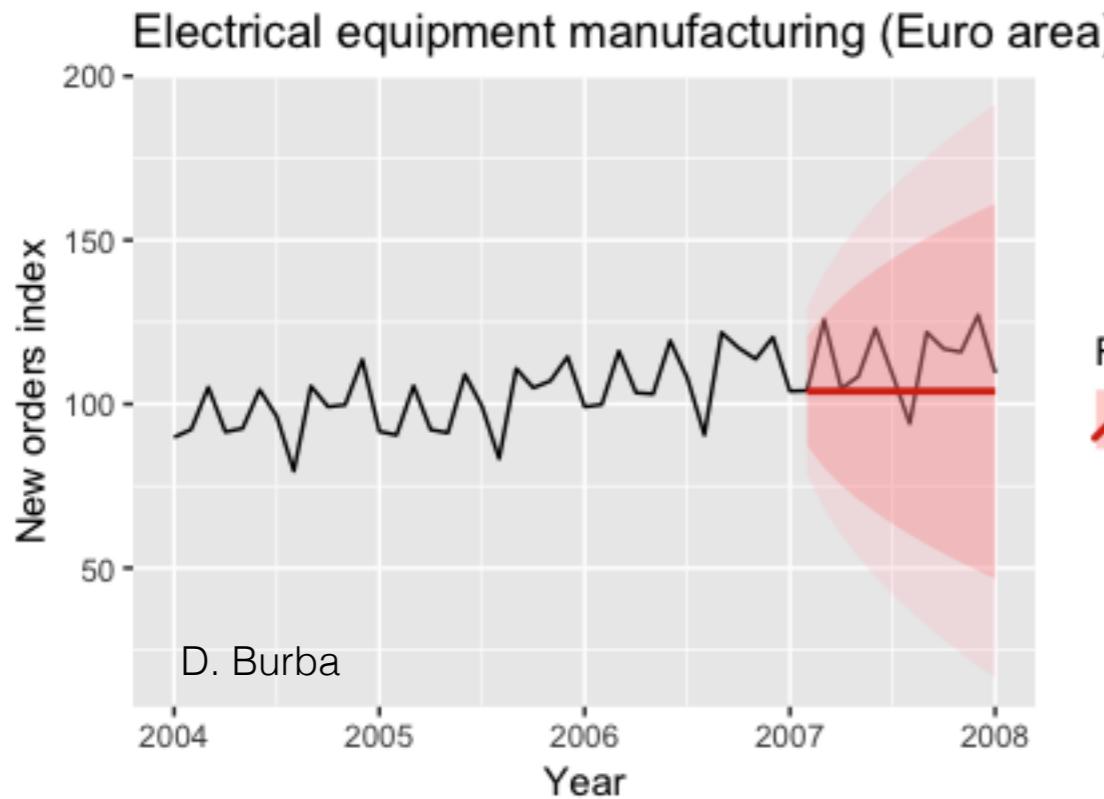
Come, sir, I will make did behold your worship.

VIOLA:

I'll drink it.

A. Karpathy

# The power of prediction



- Forecasting is of extreme importance for finance, weather prediction, etc.
- In Machine Learning, only RNNs can do this efficiently
- Forecasting has value even in the case when it fails: it tends to fail more with truly anomalous time series

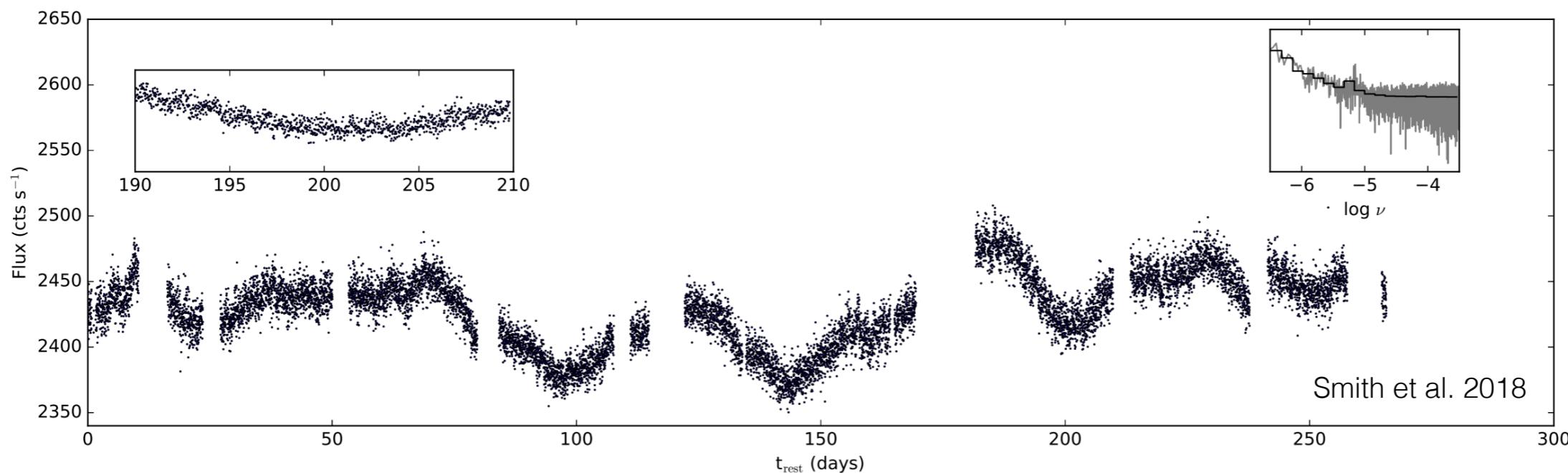
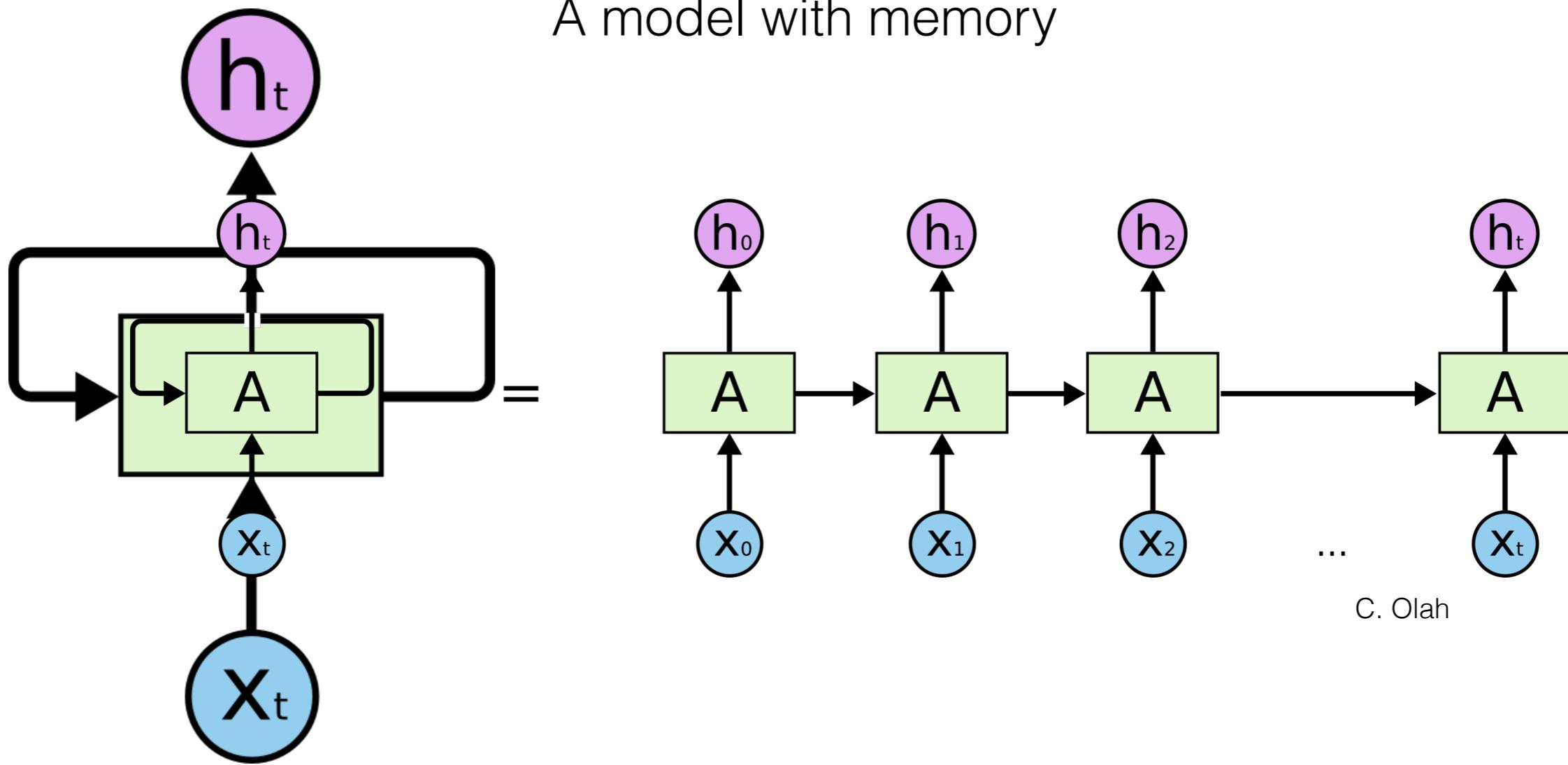


FIG. 1.— Light curve of spectroscopic AGN KIC 11614932, an object with stellar contamination within the extraction aperture. An excerpt of the light curve (left) and the power spectrum of the full light curve with the periodic signal visible (right) are shown as insets.

# How do RNNs work?

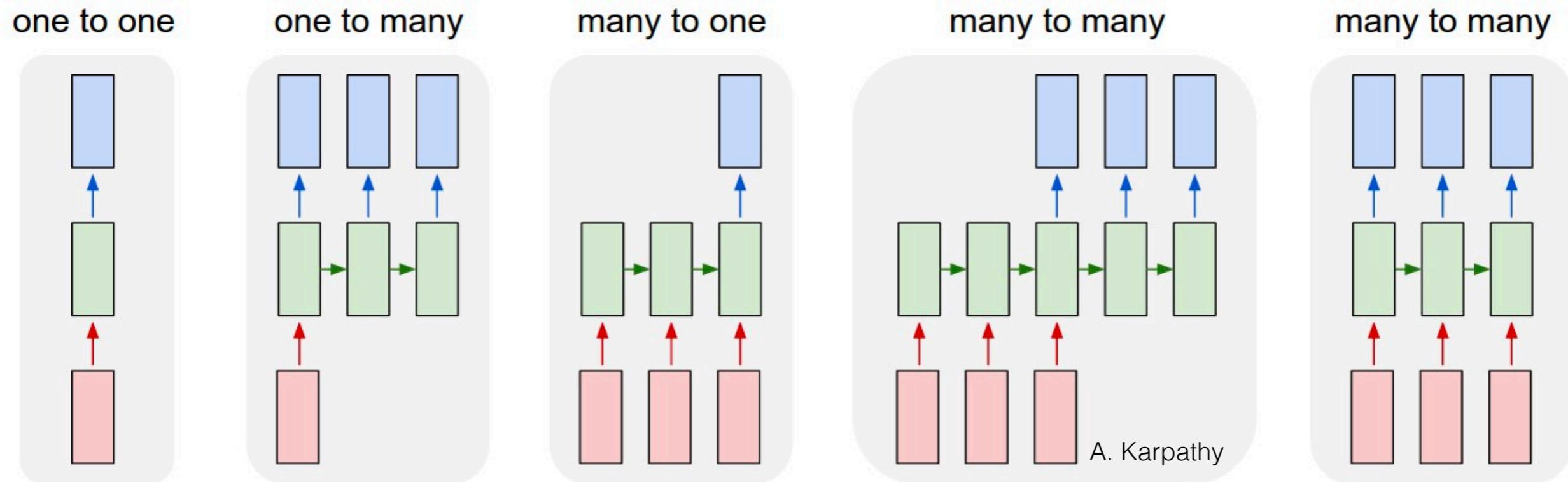
A model with memory



C. Olah

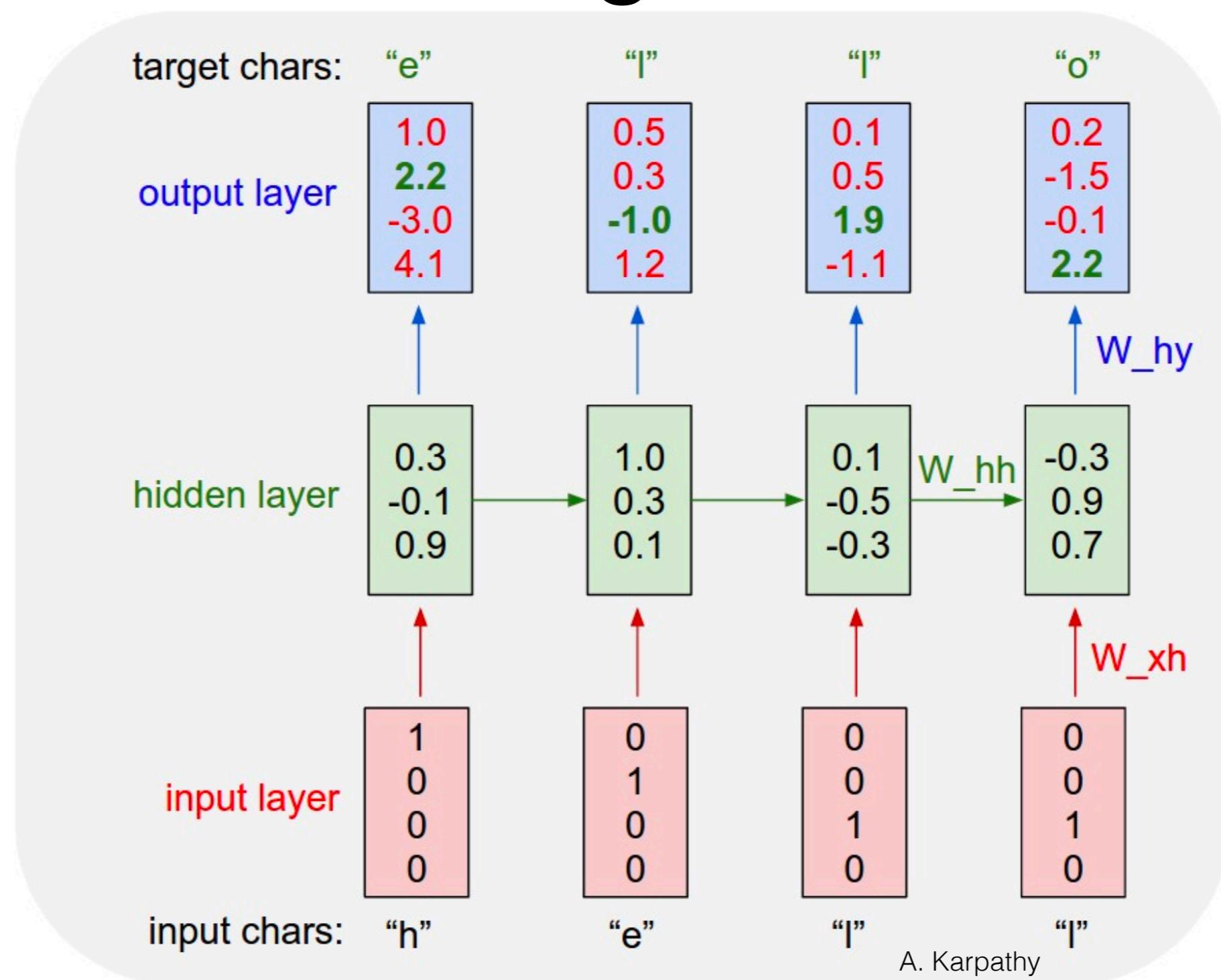
- The input is a sequence “ $\mathbf{x}$ ” (for example, over time). The output is also a sequence,  $\mathbf{h}$  (formally  $\mathbf{y} = \mathbf{W}_{hy}^* \mathbf{h}$ )
- Hidden neural units “A” have an internal state  $h_t$  that gets updated based on the input “ $x_t$ ”, and on the **previous internal state**.
- Therefore, the content of the output  $h_t$  depends not only on the current input  $x_t$ , but on **the entire history of previous inputs**.

# RNNs operate on sequences of vectors!



- RNNs are the most general form of ANNs.
- One to one: standard forward fed NN (image classification)
- One to many: Input single vector, output sequence (image captioning)
- Many to one: Input sequence, output vector (sentiment analysis)
- Many to Many: both input and output are sequences: language translation, image sequence classification

# Training RNNs



- The three matrices ( $W_{xh}$ ,  $W_{hh}$ , and  $W_{hy}$ ) are initialized randomly. Then weights are adjusted (using gradient descent) to obtain desired output.

# Training RNNs

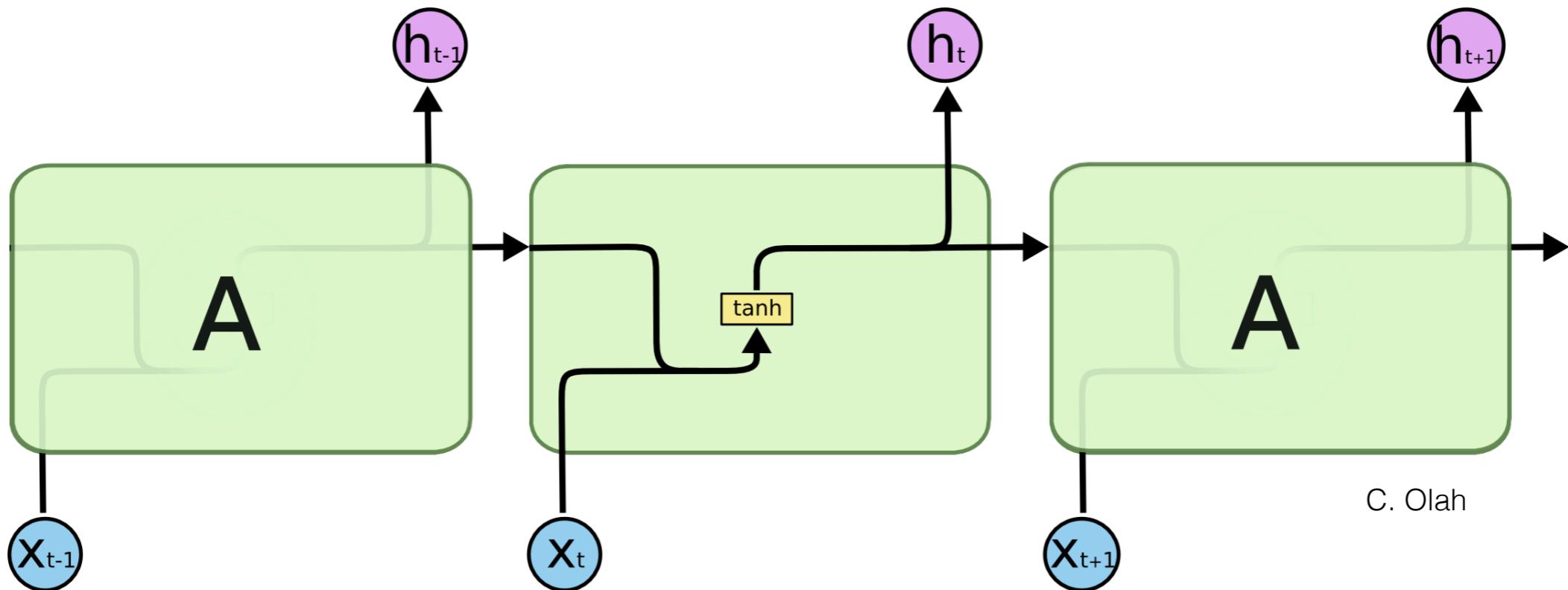
Muchos años después, frente al pelotón de

fusilamiento, el coronel Aureliano Buendía había de

recordar aquella tarde remota en que su padre lo

llevó a conocer el hielo.

# The math behind a RNN



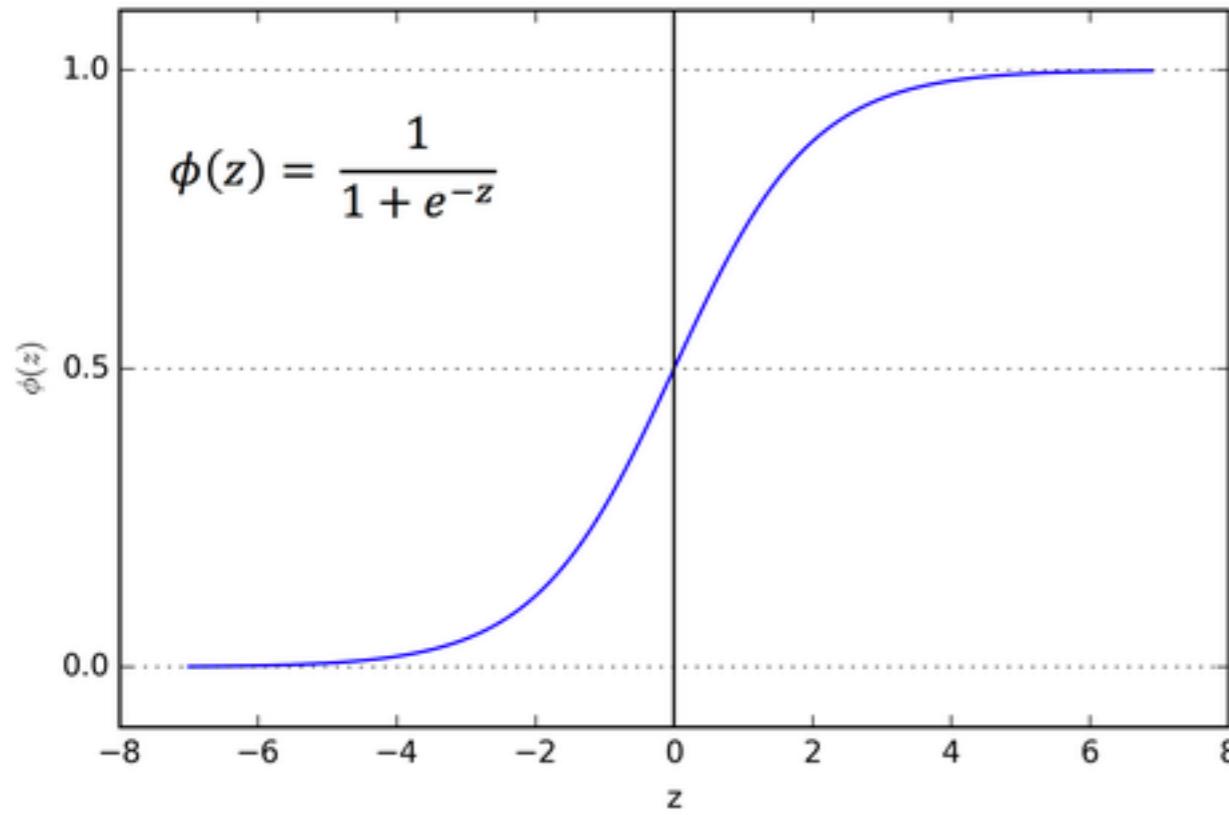
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

Code:

```
class RNN:  
    # ...  
    def step(self, x):  
        # update the hidden state  
        self.h = np.tanh(np.dot(self.W_hh, self.h) + np.dot(self.W_xh, x))  
        # compute the output vector  
        y = np.dot(self.W_hy, self.h)  
        return y
```

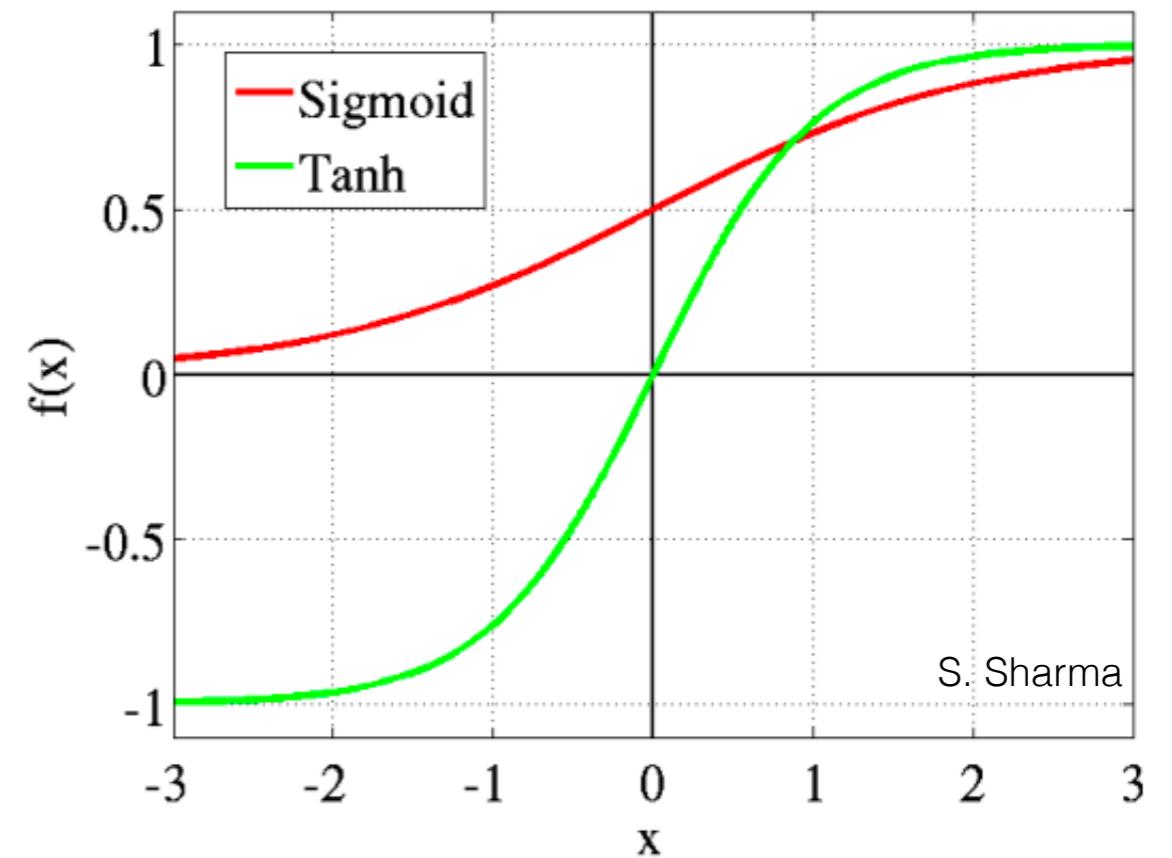
A. Karpathy

# Activation functions



Sigmoid:

- Differentiable
- Squeezes input in range [0,1]
- Easily interpretable as probability (classification)

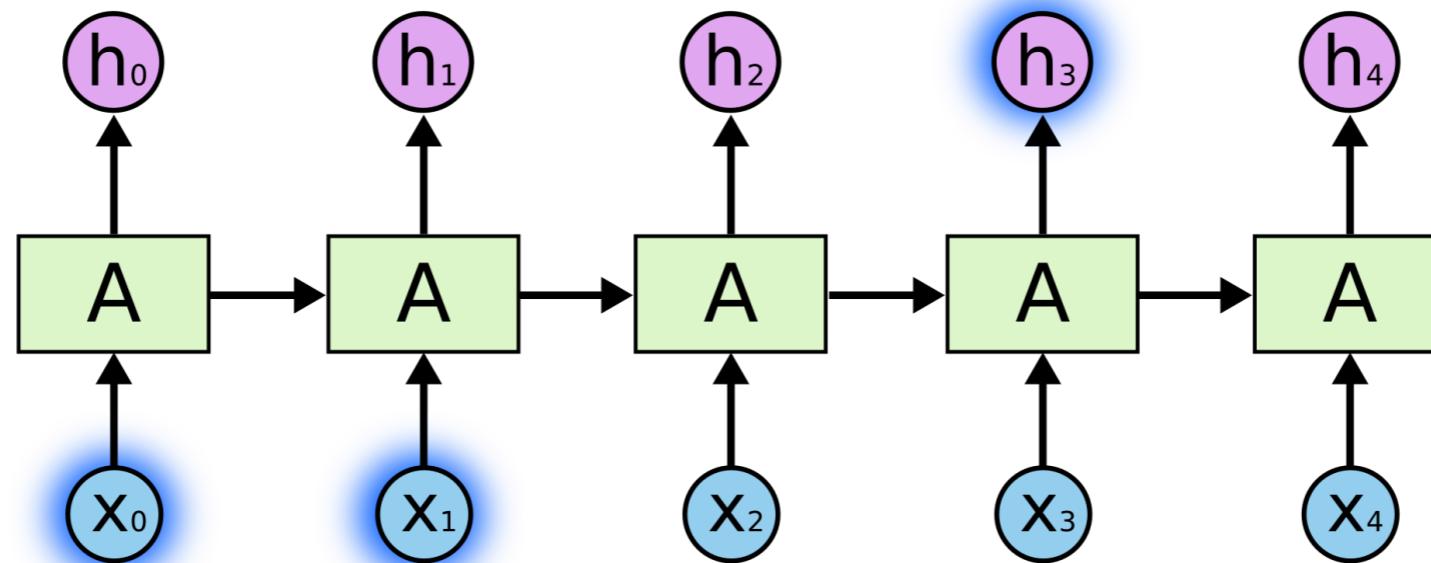


tanh:

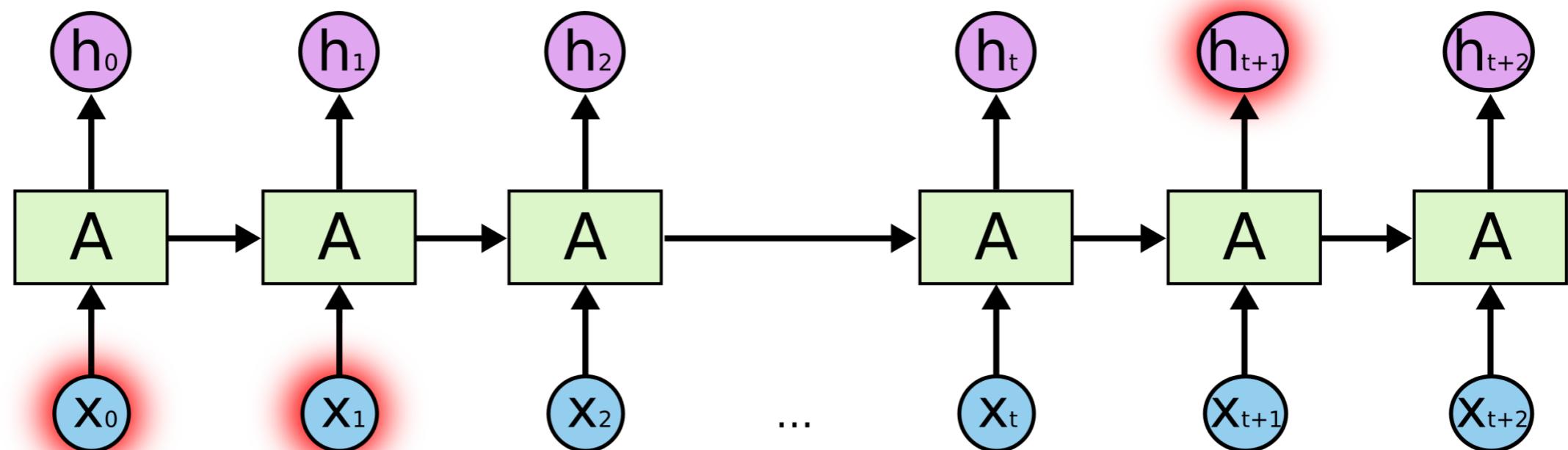
- Differentiable
- Squeezes input in range [-1,1]
- Negative and neutral values are taken into account

# Short-term vs. long-term memory

Las nubes están en el... cielo

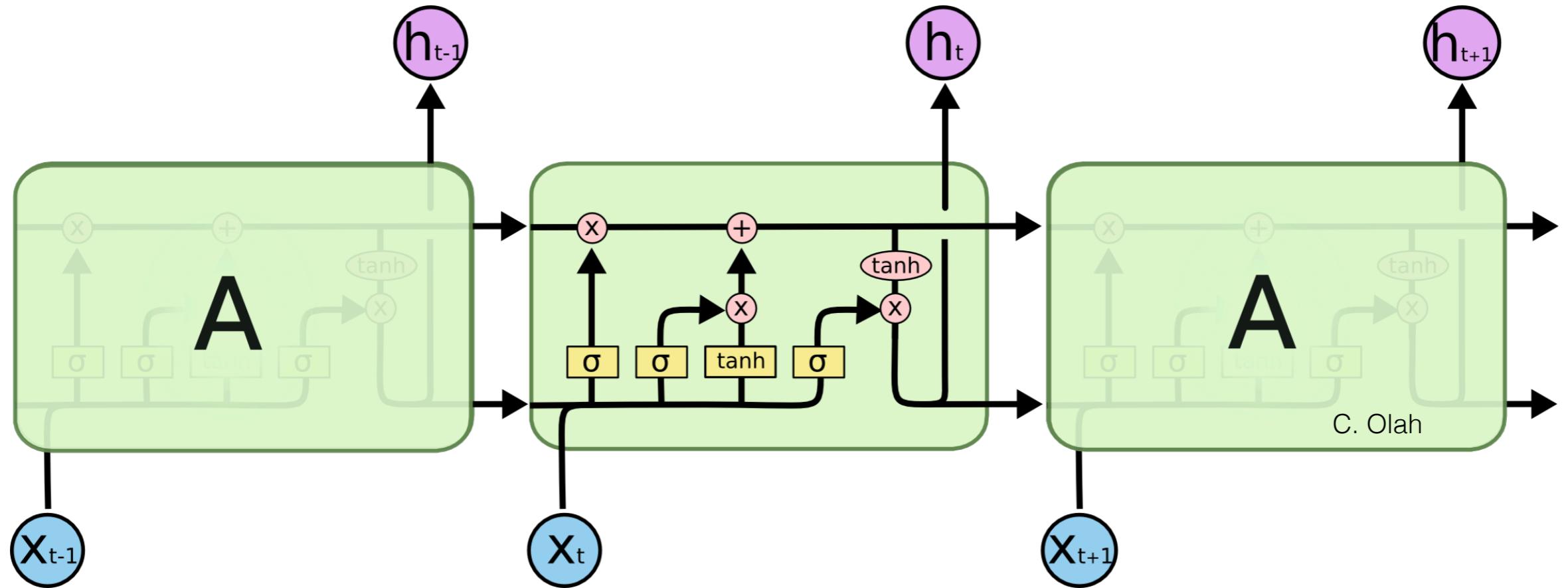


Yo crecí en Colombia, hablo perfecto... español



C. Olah

# Long Short Term Memory (LSTM) Networks



- The concept remains the same: output depends in previous hidden state, and on current input
- However, a new path is added (horizontal line at the top). This is the cell state.
- Using neural operations with the cell state, we can forget old information, or add new one, according to the input we are seeing.

Code fun!

[https://www.tensorflow.org/tutorials/  
structured data/time series](https://www.tensorflow.org/tutorials/structured_data/time_series)