# Social Media & Text Analysis

## lecture 9 - Deep Learning for NLP

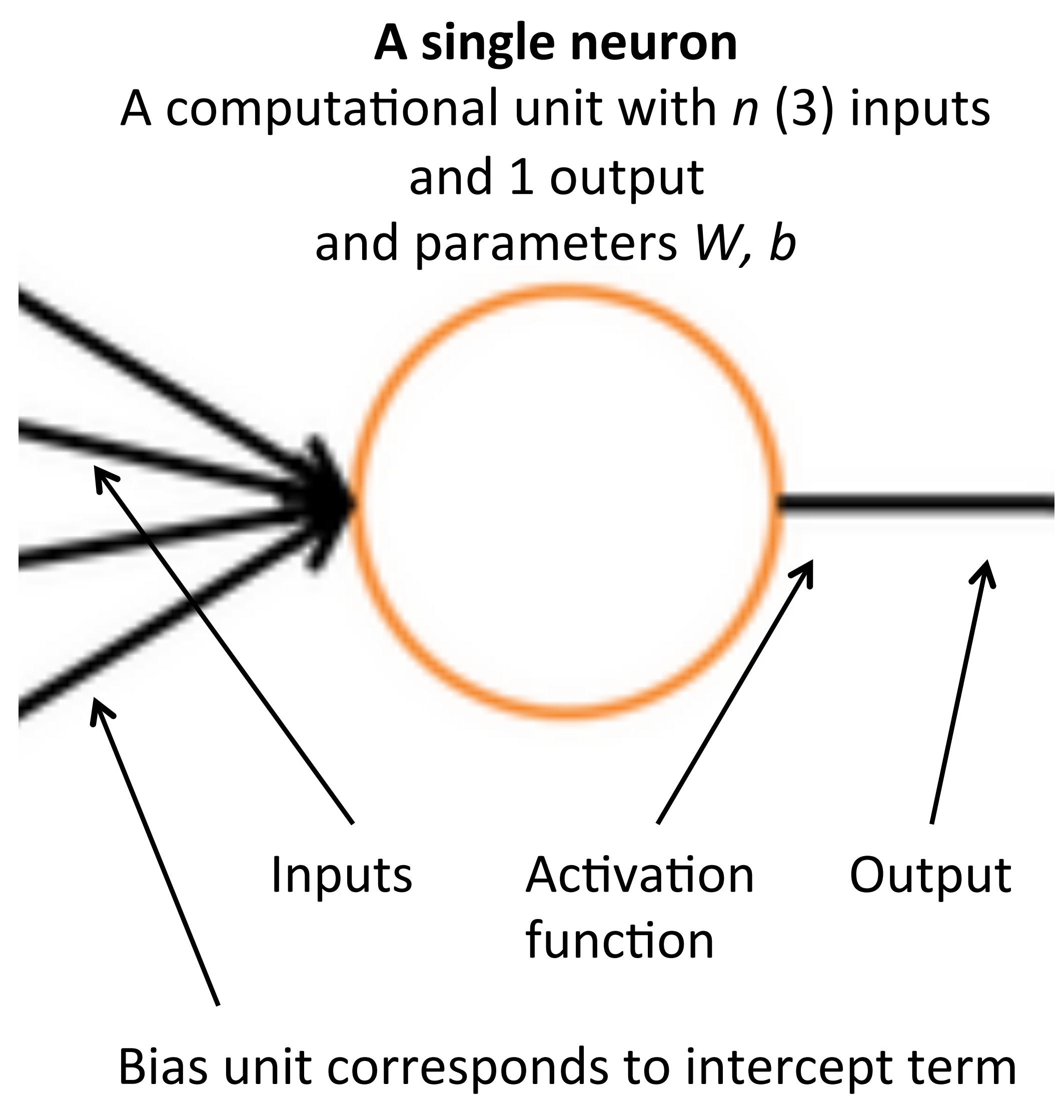
Follow @cocoweixu

**CSE 5539-0010 Ohio State University**
**Instructor: Wei Xu**
**Website: [socialmedia-class.org](socialmedia-class.org)**

some slides are adapted from Richard Socher, Greg Durret, Chris Dyer, Dan Jurafsky, Chris Manning

# A Neuron

- If you know Logistic Regression, then you already understand a basic neural network neuron!



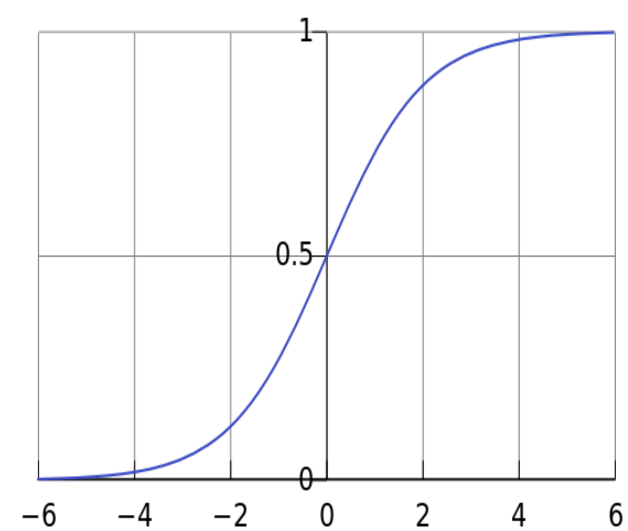
# A Neuron

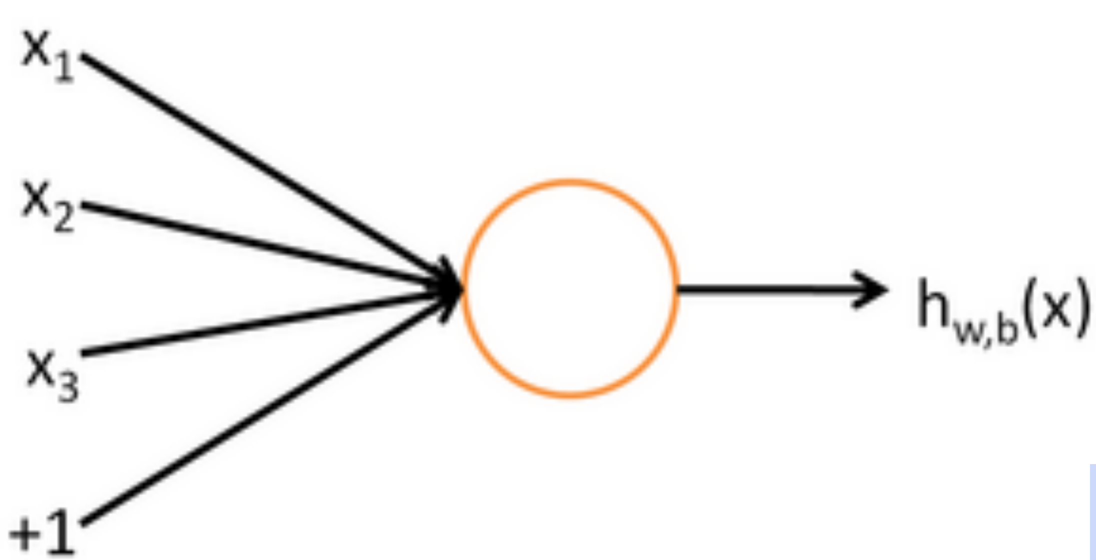
is essentially a binary logistic regression unit

$$h_{w,b}(x) = f(w^{\mathsf{T}}x + b)$$

*b:* We can have an "always on" feature, which gives a class prior, or separate it out, as a bias term

$$f(z) = \frac{1}{1+e^{-z}}$$



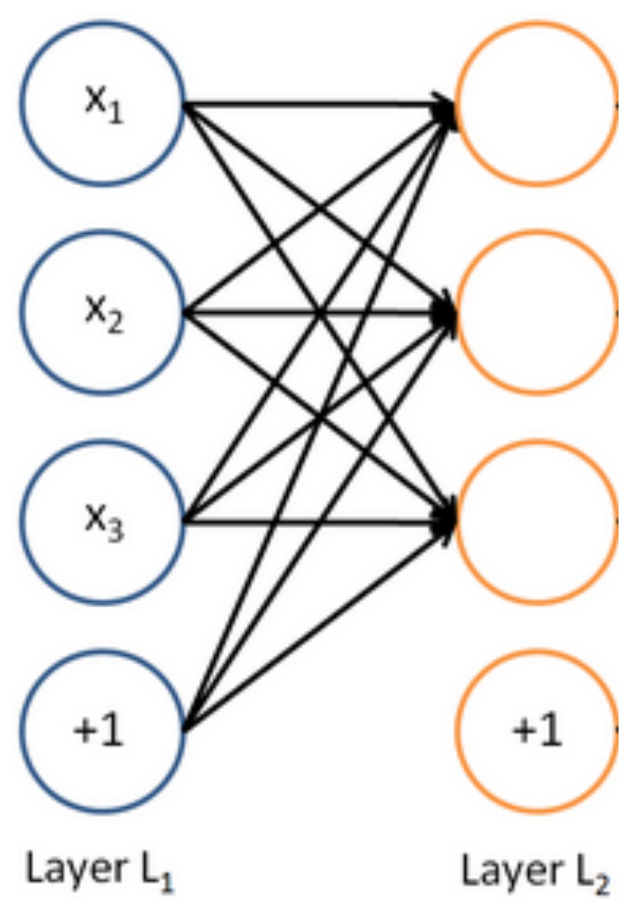




*w*, *b* are the parameters of this neuron
i.e., this logistic regression model

# A Neural Network

= running several logistic regressions at the same time

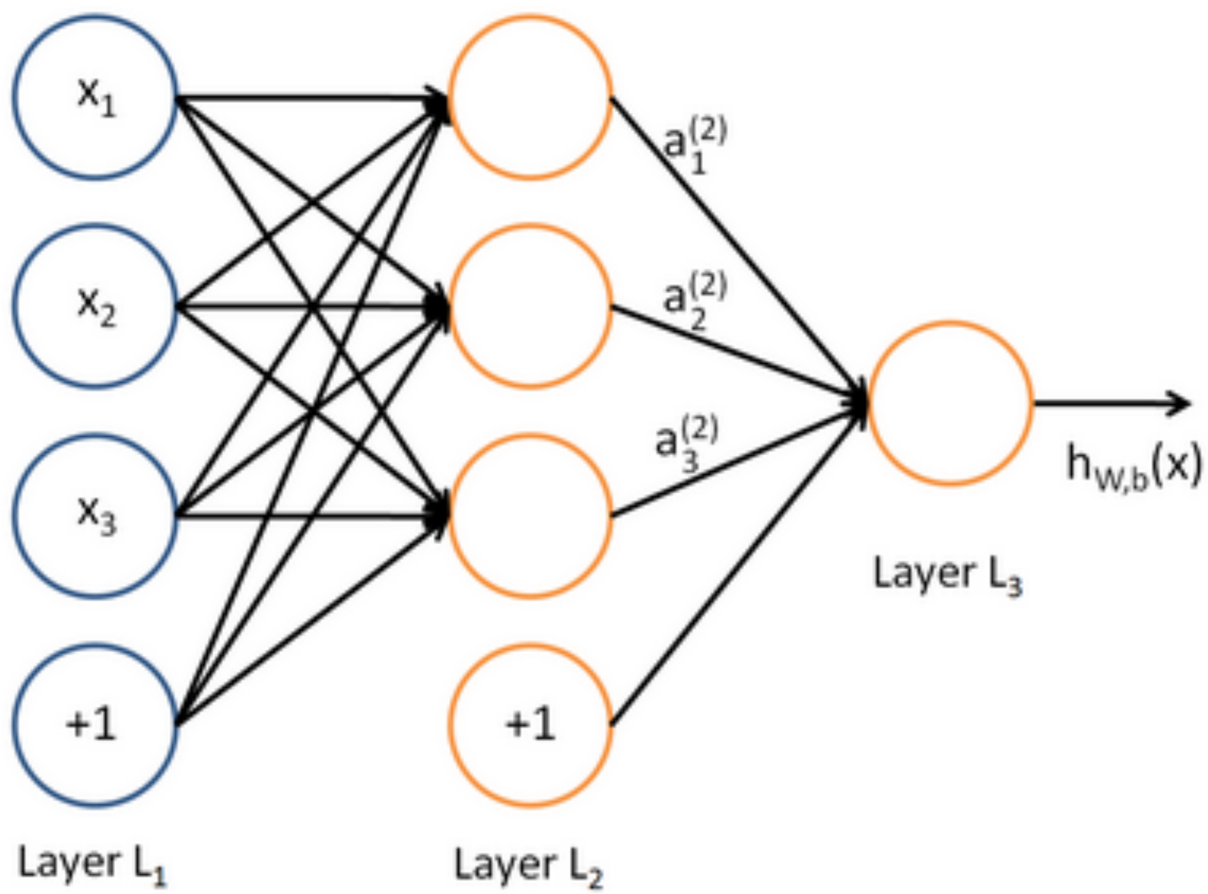
If we feed a vector of inputs through a bunch of logistic regression functions, then we get a vector of outputs …



# A Neural Network

= running several logistic regressions at the same time

… which we can feed into another logistic regression function

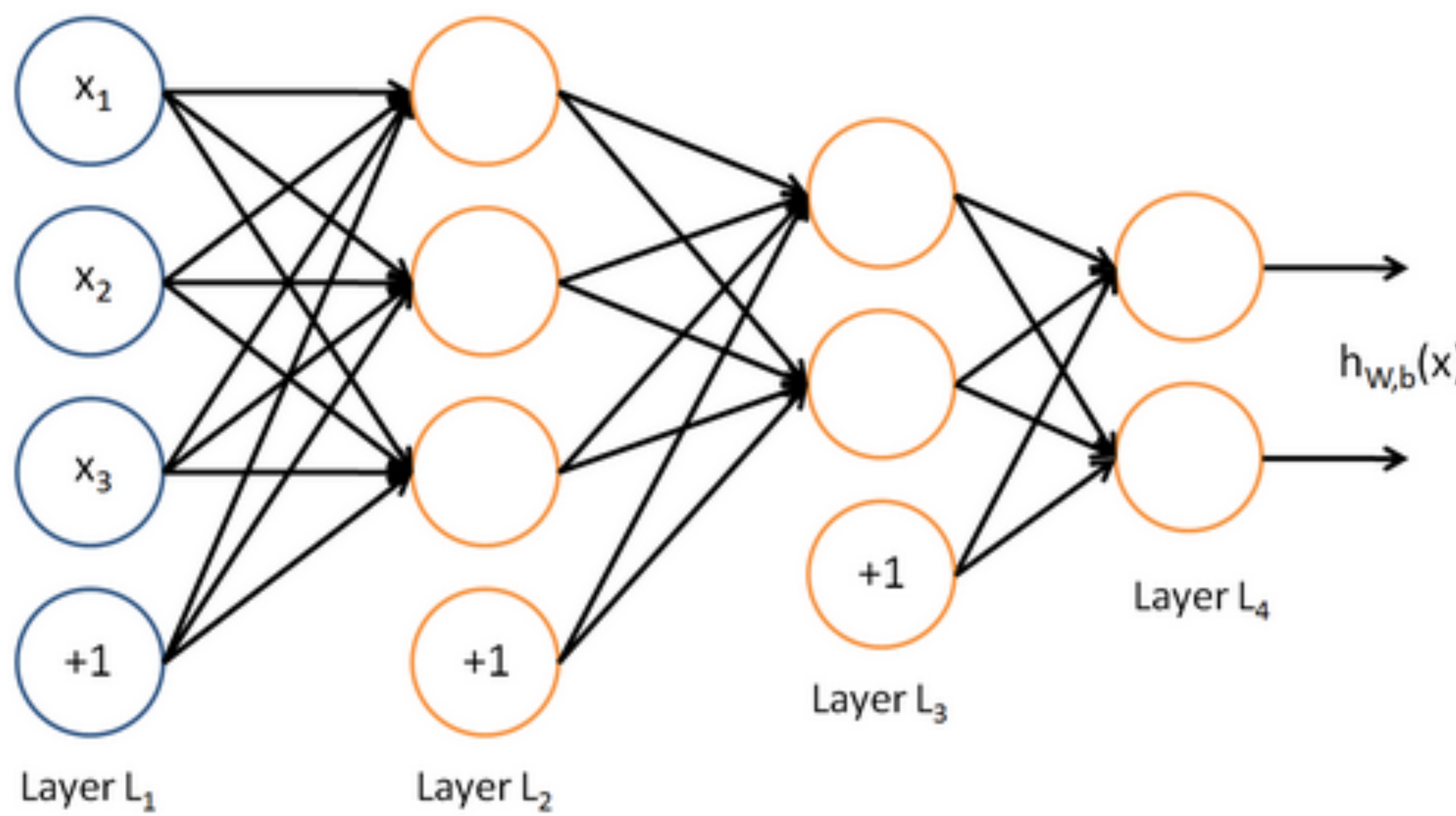



*It is the loss function that will direct what the intermediate hidden variables should be, so as to do a good job at predicting the targets for the next layer, etc.*

# A Neural Network

= running several logistic regressions at the same time

Before we know it, we have a multilayer neural network….



# $f$: Activation Function

We have

$$a_1 = f(W_{11}x_1 + W_{12}x_2 + W_{13}x_3 + b_1)$$
$$a_2 = f(W_{21}x_1 + W_{22}x_2 + W_{23}x_3 + b_2)$$

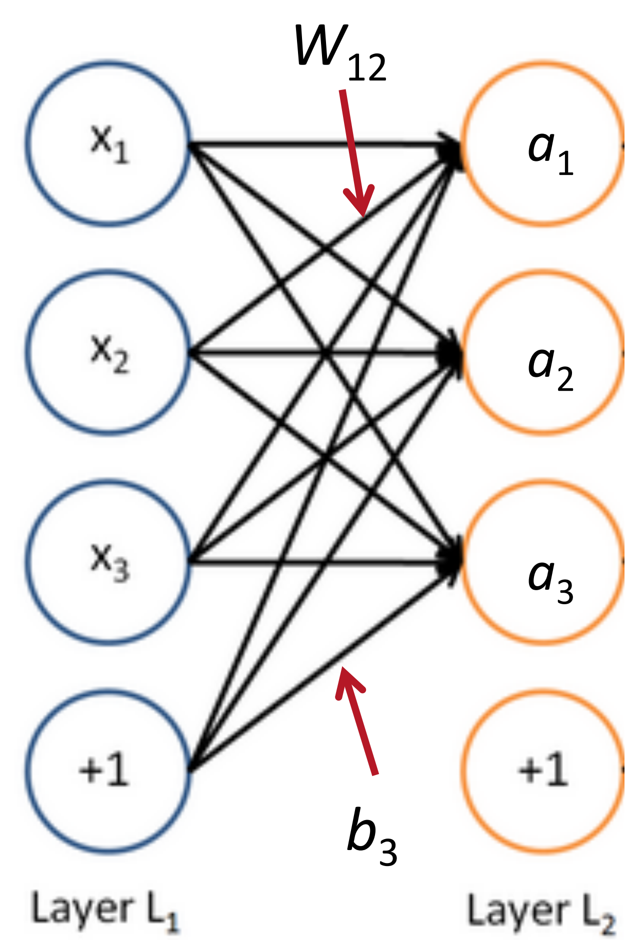
etc.

In matrix notation

$$z = Wx + b$$
$$a = f(z)$$

where $f$ is applied element-wise:

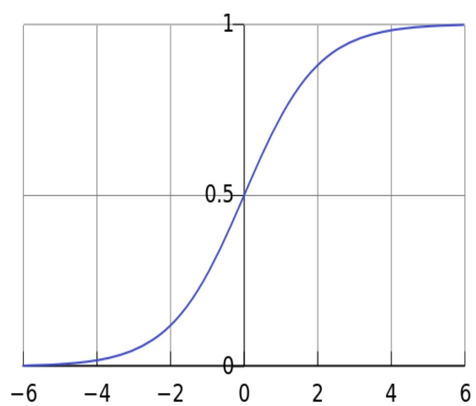
$$f([z_1, z_2, z_3]) = [f(z_1), f(z_2), f(z_3)]$$



# Activation Function

logistic ("sigmoid")

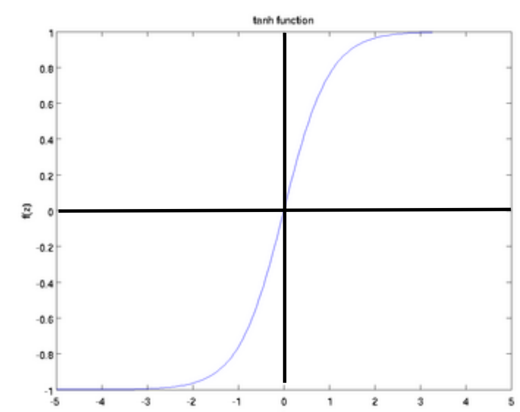
$$f(z) = \frac{1}{1+\exp(-z)}.$$




$$f'(z) = f(z)(1 - f(z))$$

tanh

$$f(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}},$$




$$f'(z) = 1 - f(z)^2$$

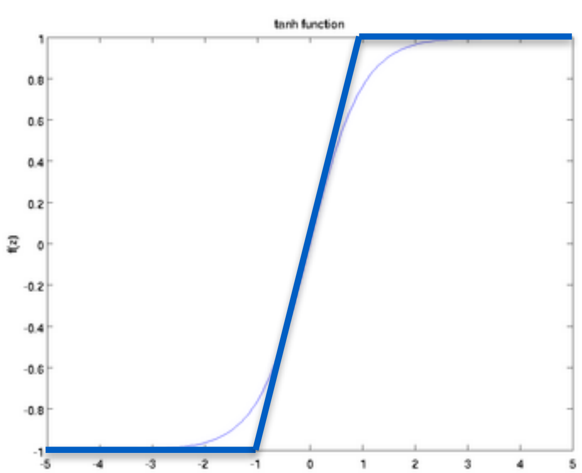
tanh is just a rescaled and shifted sigmoid

$$\tanh(z) = 2\text{logistic}(2z) - 1$$

# Activation Function

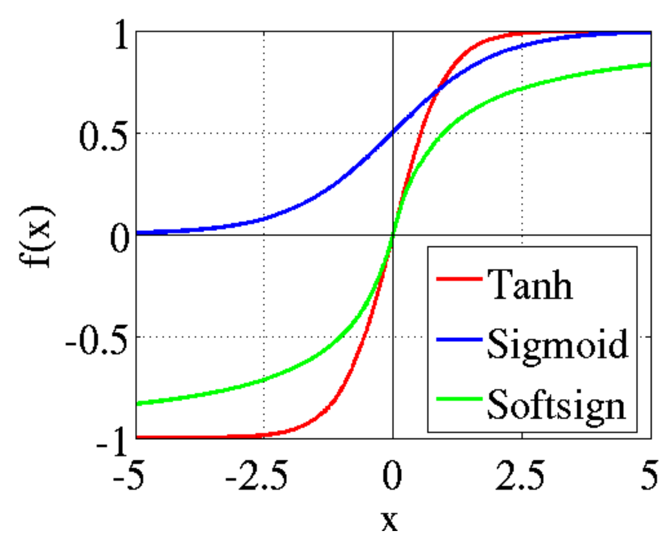
hard tanh

$$\mathrm{HardTanh}(x) = \begin{cases} -1 & \text{if } x < -1 \\ x & \text{if } -1 <= x <= 1 \\ 1 & \text{if } x > 1 \end{cases}$$



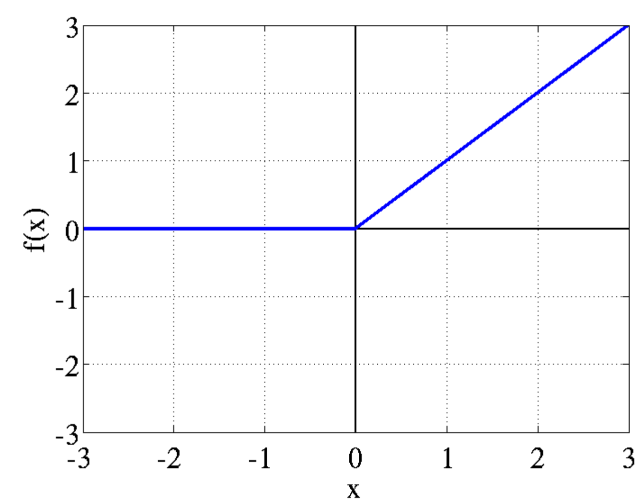
soft sign

$$\mathrm{softsign}(z) = \frac{a}{1+|a|}$$




**rectified linear (ReLu)**

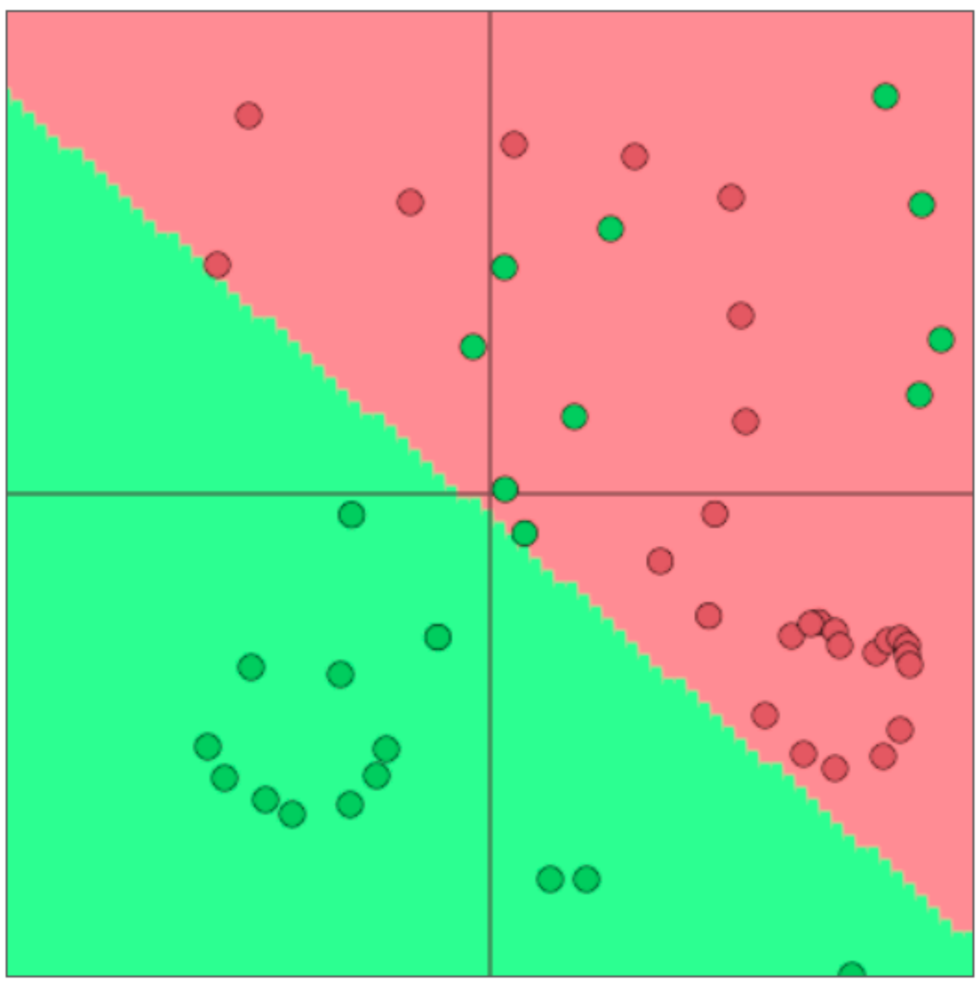
$$\mathrm{rect}(z) = \max(z, 0)$$




- hard tanh similar but computationally cheaper than tanh and saturates hard.
- Glorot and Bengio, *AISTATS* 2011 discuss softsign and rectifier

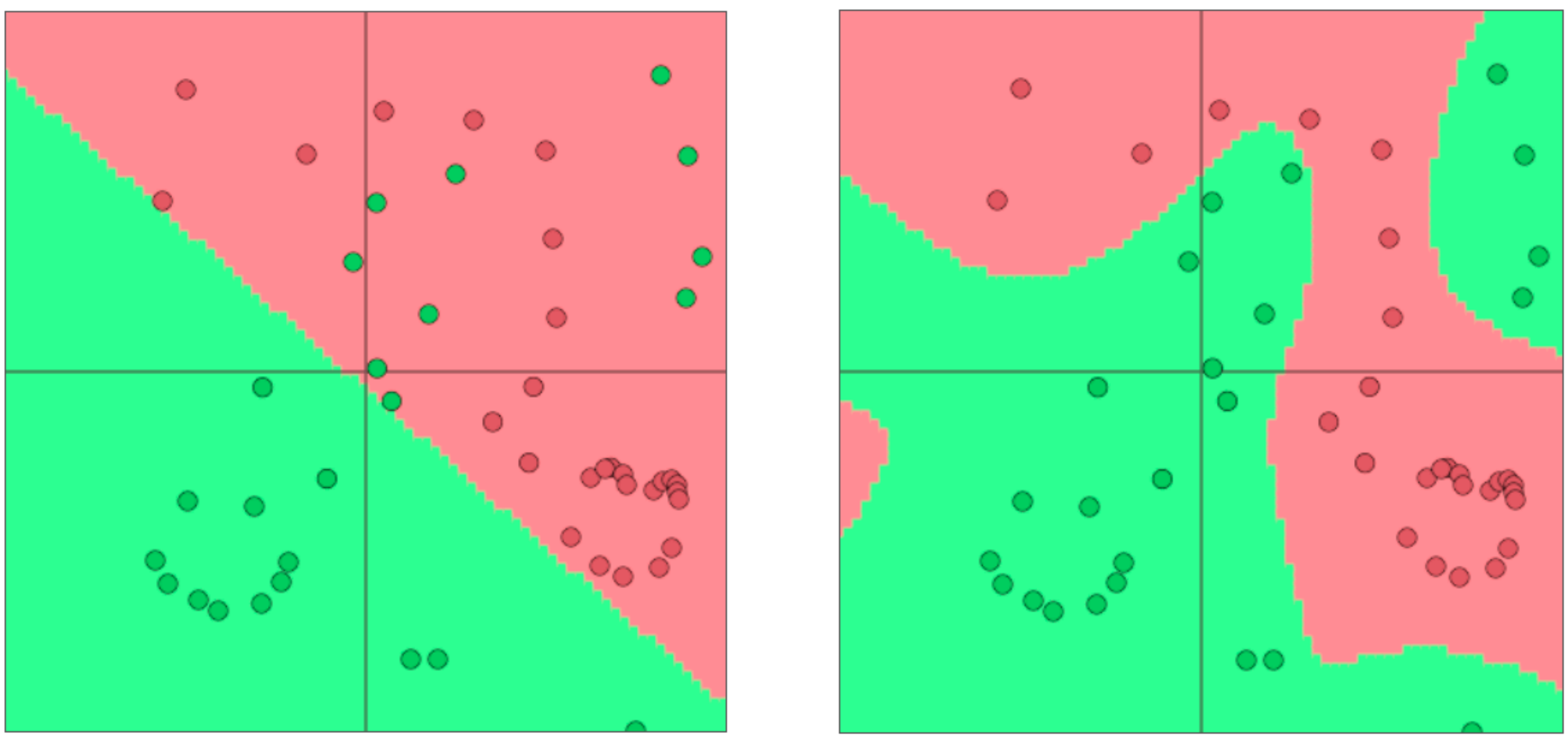
# Non-linearity

- Logistic (Softmax) Regression only gives linear decision boundaries

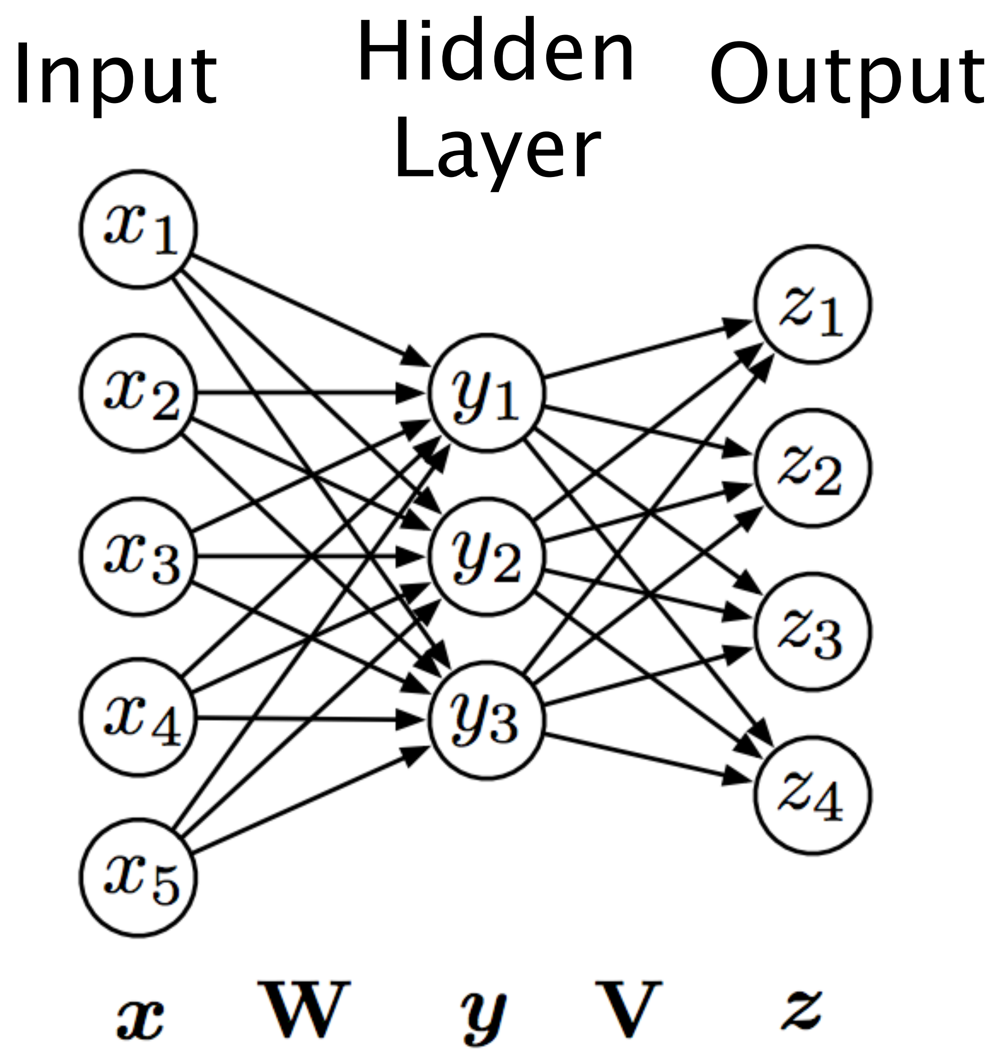


# Non-linearity

- Neural networks can learn much more complex functions and nonlinear decision boundaries!



# Non-linearity




$$\boldsymbol{y} = g(\mathbf{W}\boldsymbol{x} + \boldsymbol{b})$$

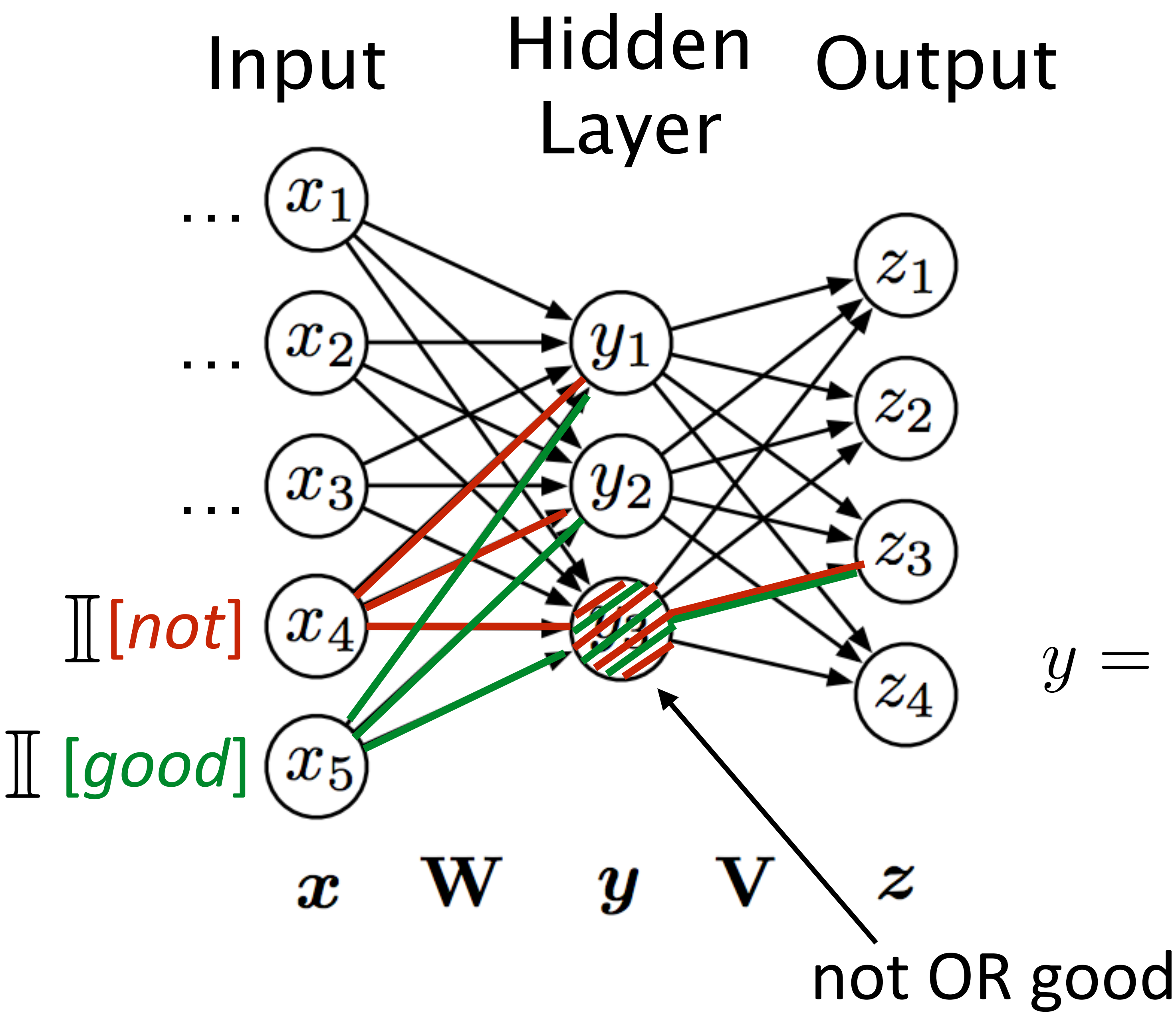
$$\mathbf{z} = g(\mathbf{V}\underbrace{g(\mathbf{W}\mathbf{x} + \mathbf{b})}_{\text{output of first layer}} + \mathbf{c})$$

With no nonlinearity:

$$\mathbf{z} = \mathbf{V}\mathbf{W}\mathbf{x} + \mathbf{V}\mathbf{b} + \mathbf{c}$$

Equivalent to $\mathbf{z} = \mathbf{U}\mathbf{x} + \mathbf{d}$

# Non-linearity




Nodes in the hidden layer can learn interactions or conjunctions of features

$$y = -2x_1 - x_2 + 2\tanh(x_1 + x_2)$$

# What about Word2vec (Skip-gram and CBOW)?

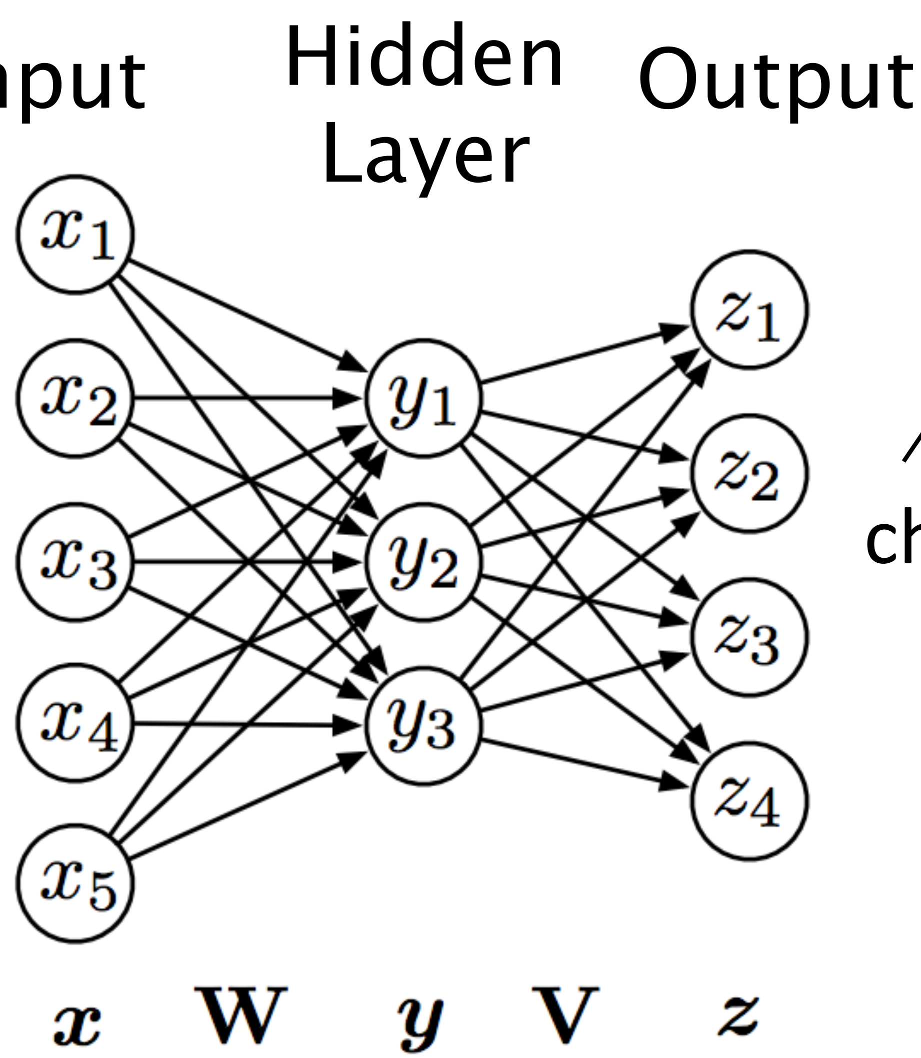
# So, what about Word2vec (Skip-gram and CBOW)?

It is not deep learning — but “shallow” neural networks.

It is — in fact — a log-linear model (softmax regression).

So, it is faster over larger dataset yielding better embeddings.

# Learning Neural Networks




change in output w.r.t. hidden

$$\frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx}$$

change in hidden w.r.t. input

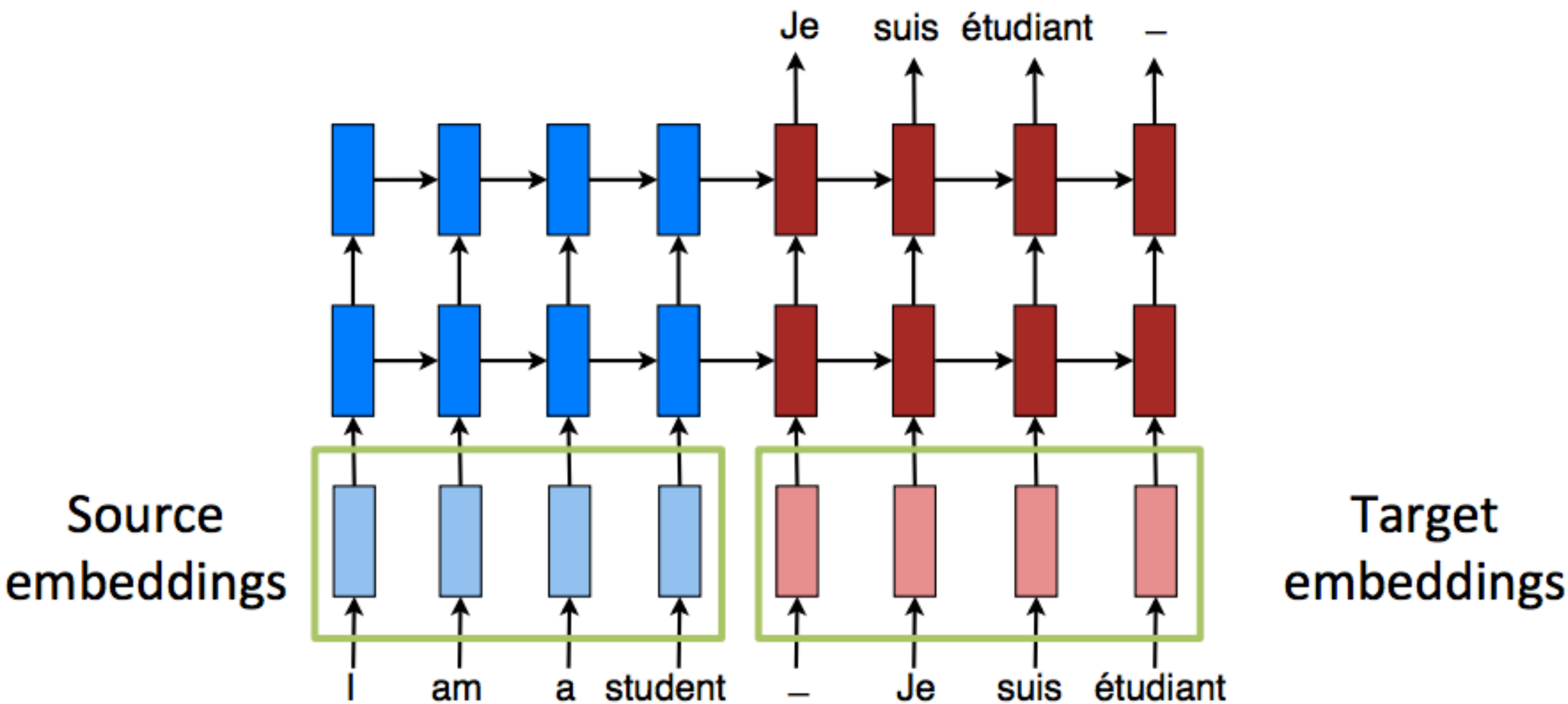
change in output w.r.t. input

Computing these looks like running this network in reverse (backpropagation)

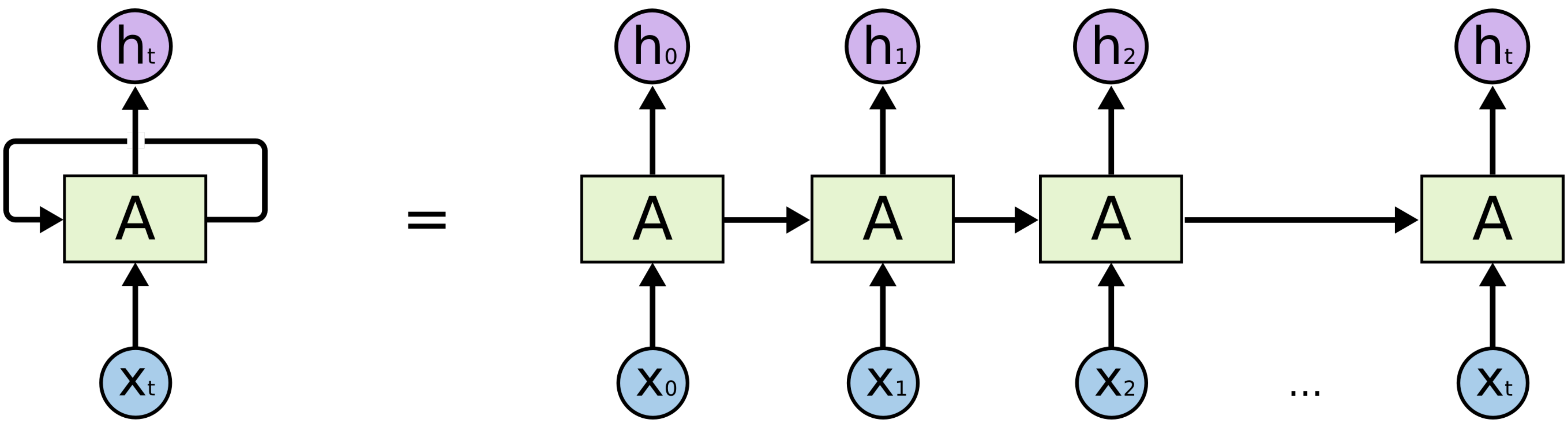
# Strategy for Successful NNs

- Select network structure appropriate for problem
  - Structure: Single words, fixed windows, sentence based,  document level; bag of words, recursive vs. recurrent, CNN, …
  - Nonlinearity
- Check for implementation bugs with gradient checks
- Parameter initialization
- Optimization tricks
- Check if the model is powerful enough to overfit
  - If not, change model structure or make model “larger”
  - If you can overfit: regularize

# Neural Machine Translation

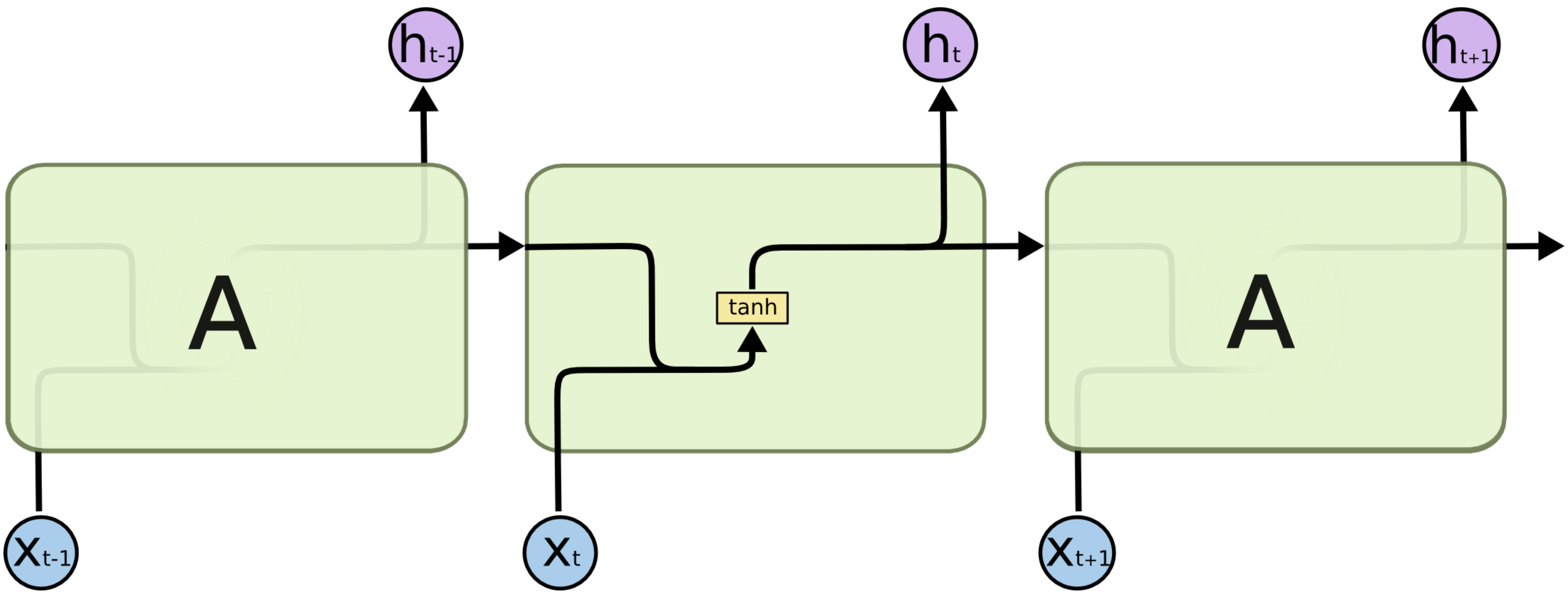


# Recurrent Neural Network (RNN)




Source: Colah's Blog

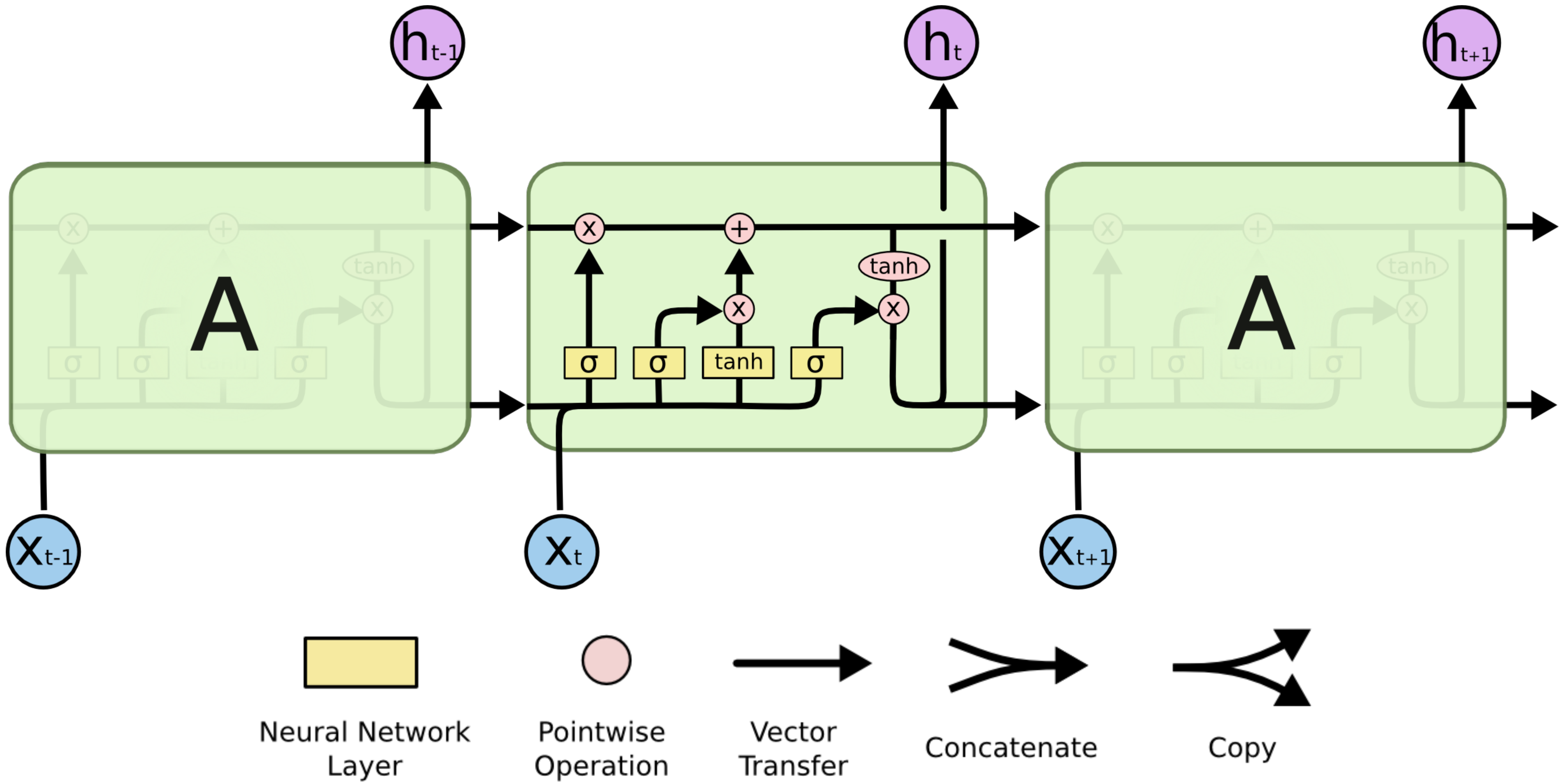
# Recurrent Neural Network (RNN)




Source: Colah's Blog

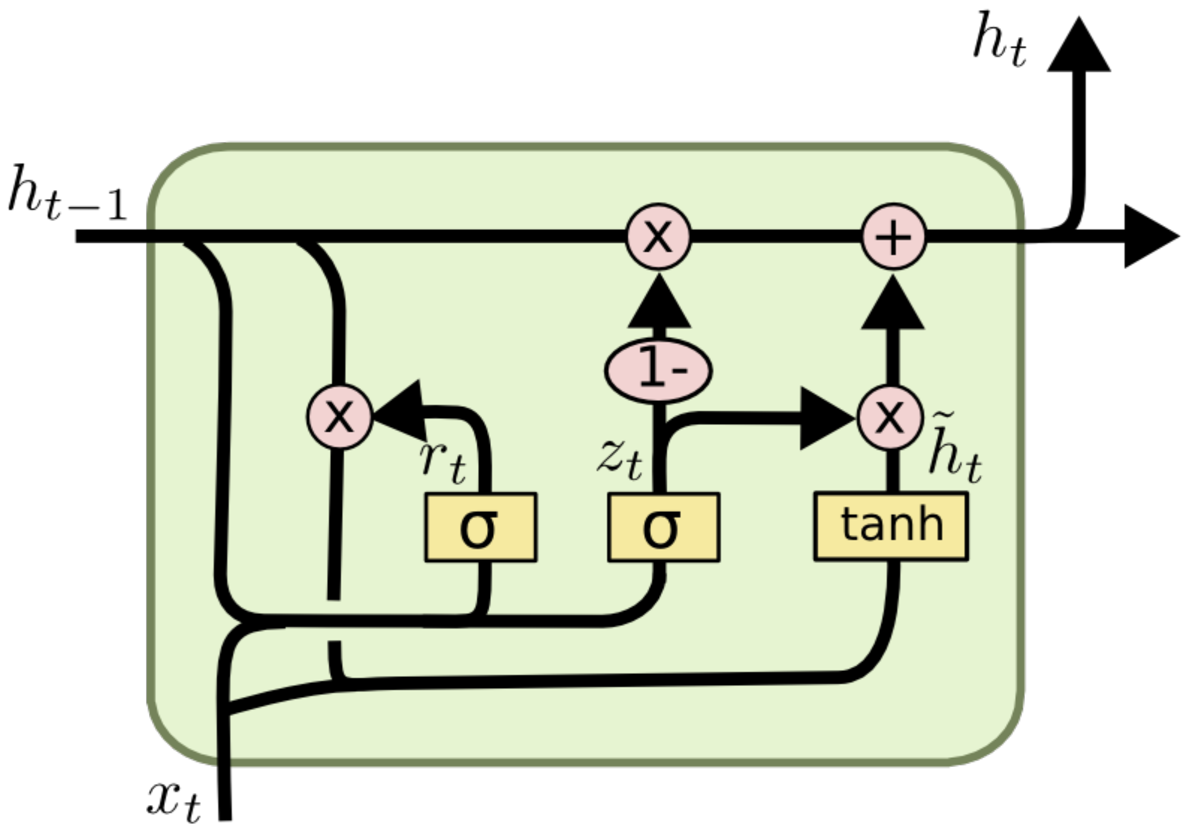
# Long Short-Term Memory Networks (LSTM)

(Hochreiter & Schmidhuber, 1997)




Source: Colah's Blog

# Long Short-Term Memory Networks (LSTM)




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

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

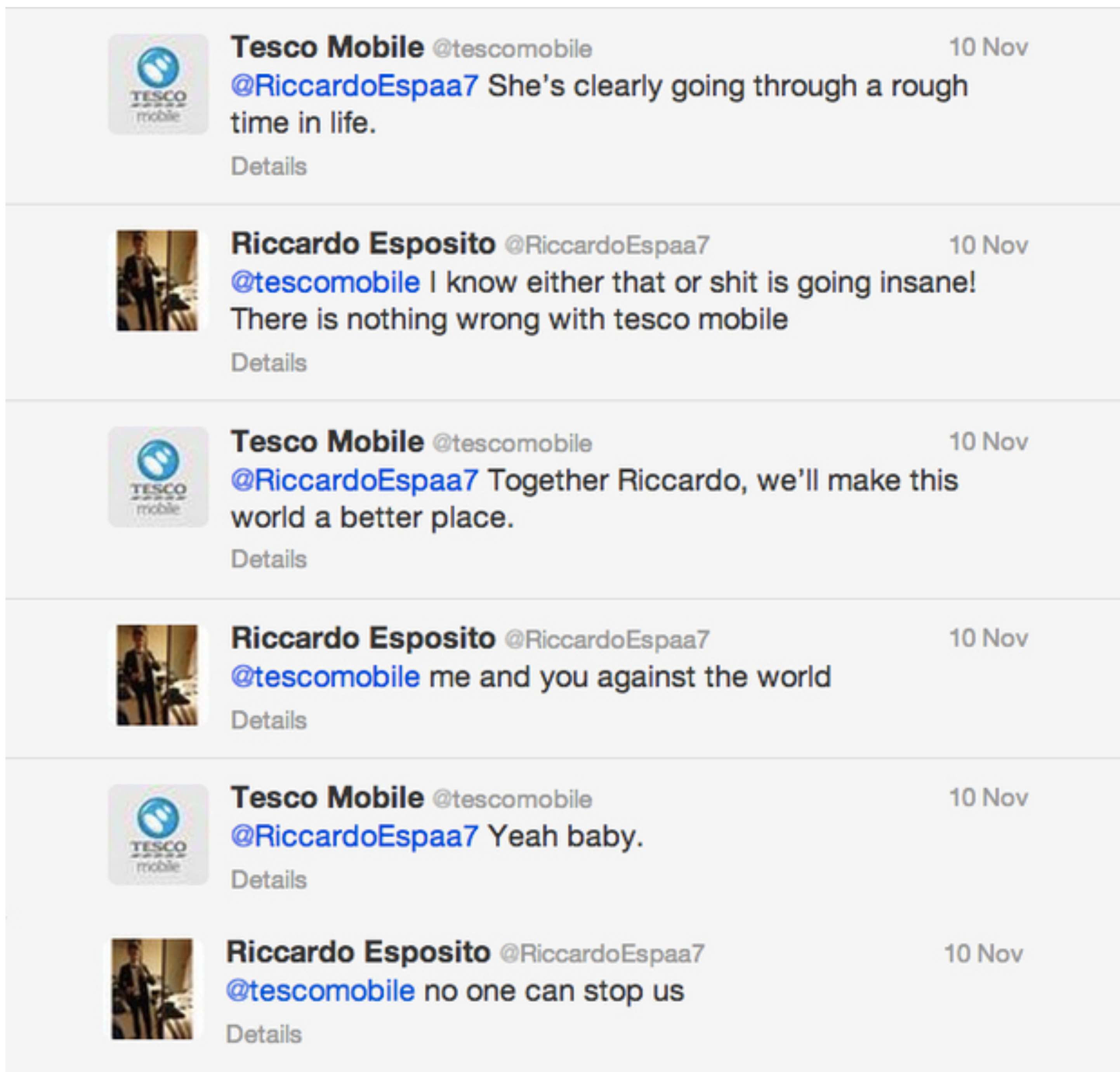
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

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

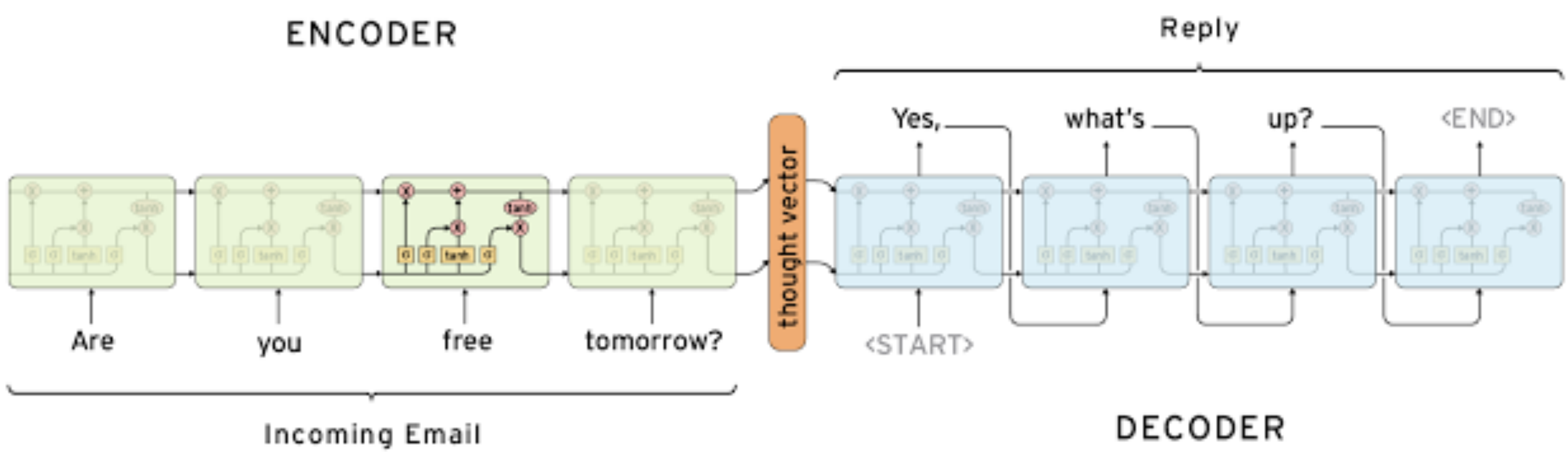



Source: Colah's Blog

# Twitter Conversation Data



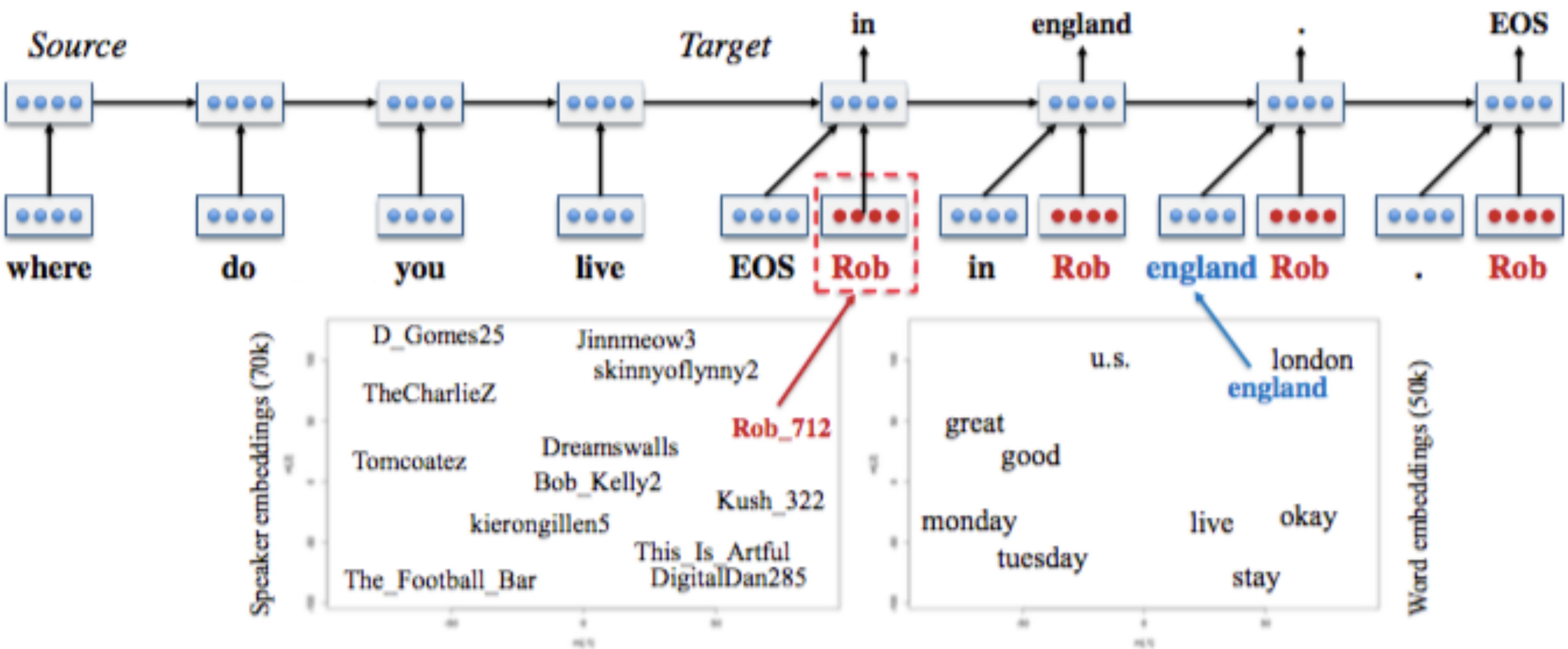
# Neural Conversation




Source: Google's blog

# Neural Conversation

modeling speakers






Source: A Persona-Based Neural Conversation Model, Li et al. (ACL 2016)

# Neural Network Toolkits

★ **PyTorch**: http://pytorch.org/
  - Facebook AI Research and many others
- **Tensorflow**: https://www.tensorflow.org/
  - By Google, actively maintained, bindings for many languages
- **DyNet**: https://github.com/clab/dynet
  - CMU and other individual researchers, dynamic structures that change for every training instance
- **Caffe**: http://caffe.berkeleyvision.org/
  - UC Berkeley, for vision
- **Theano**: http://deeplearning.net/software/theano
  - University of Montreal, less and less maintained

Follow @cocoweixu

**Instructor: Wei Xu**
**www.cis.upenn.edu/~xwe/**

**Course Website: socialmedia-class.org**