Natural Language Processing with PyTorch

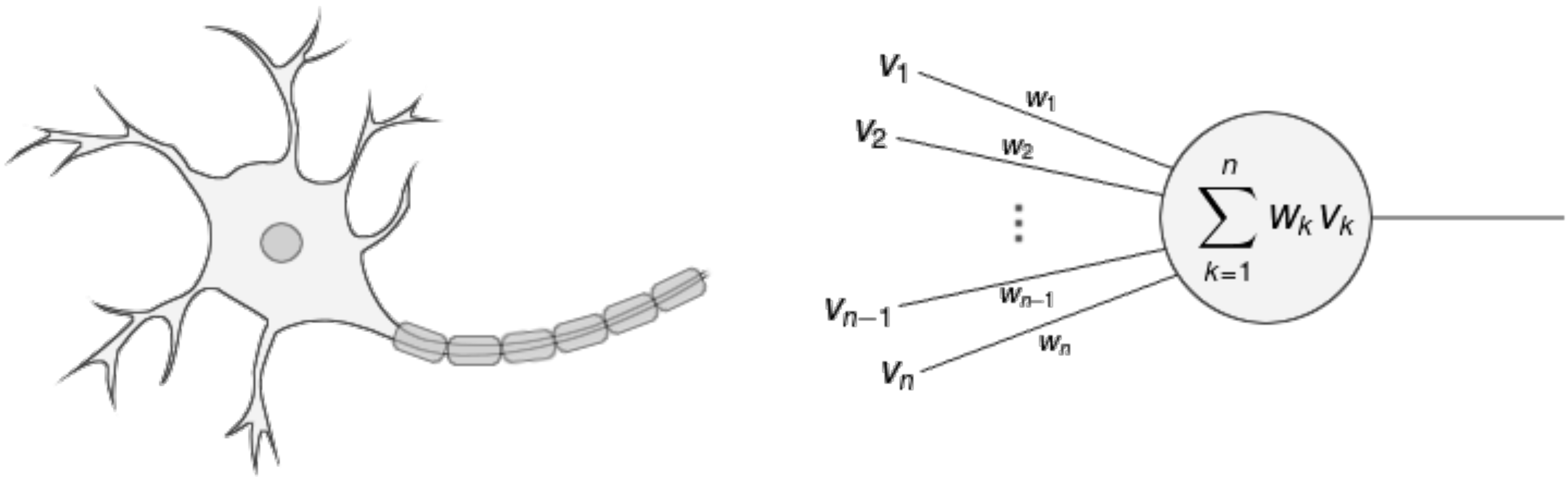
Week 3 Neural Language Processing

Week 2 Review & Warranty

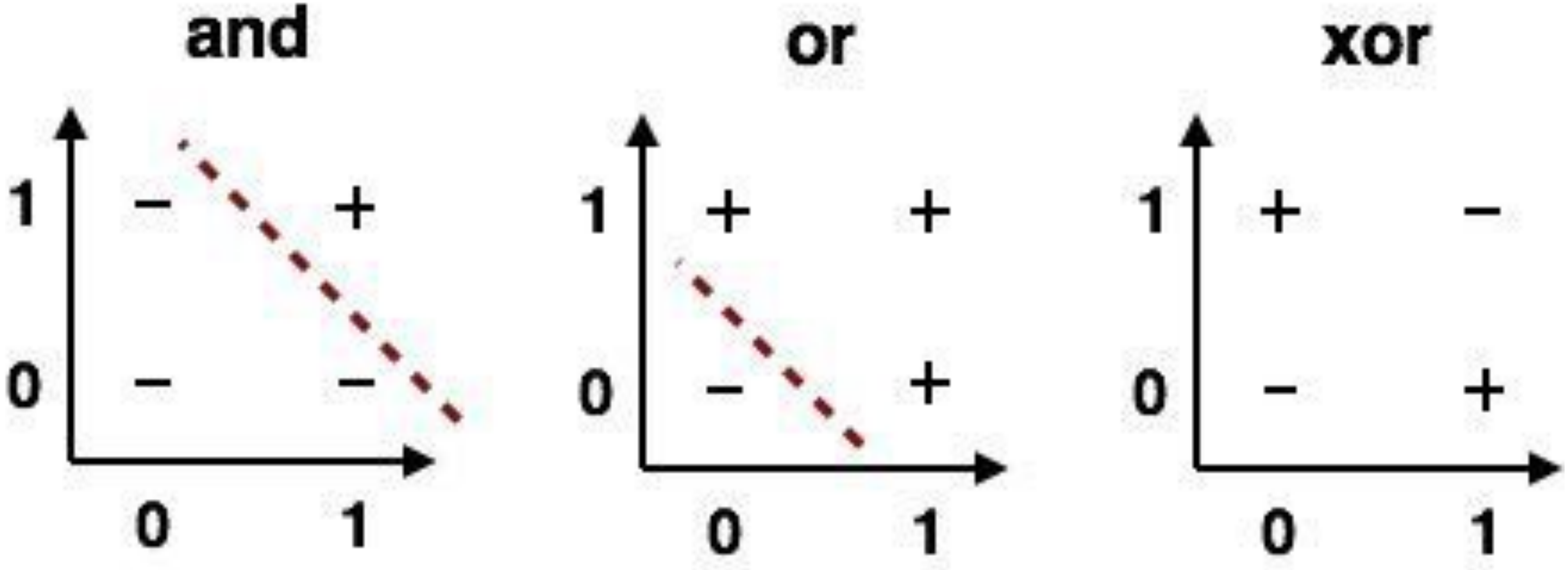
The story of CNNs

The story of RNNs

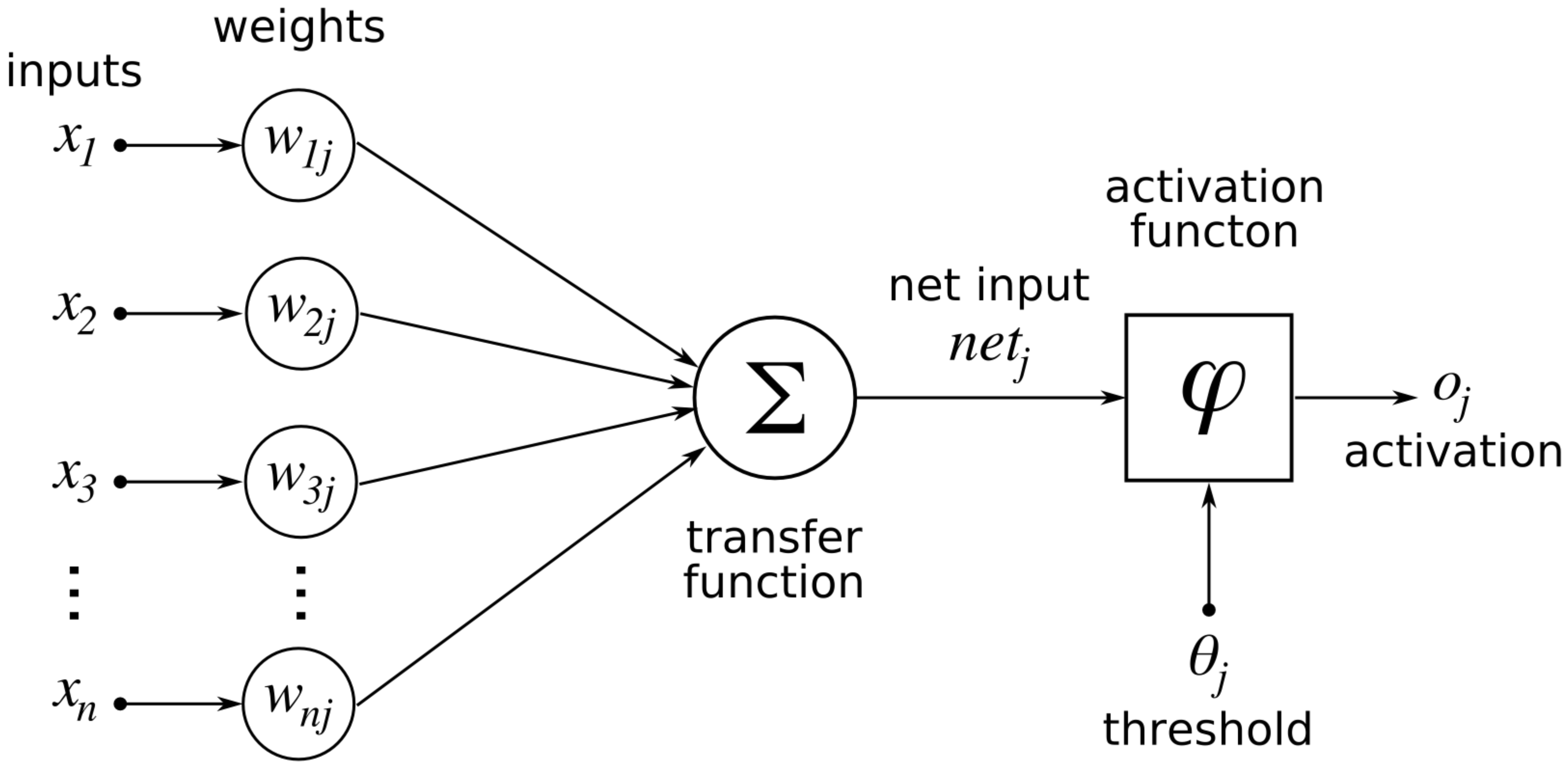
The story of residual connections



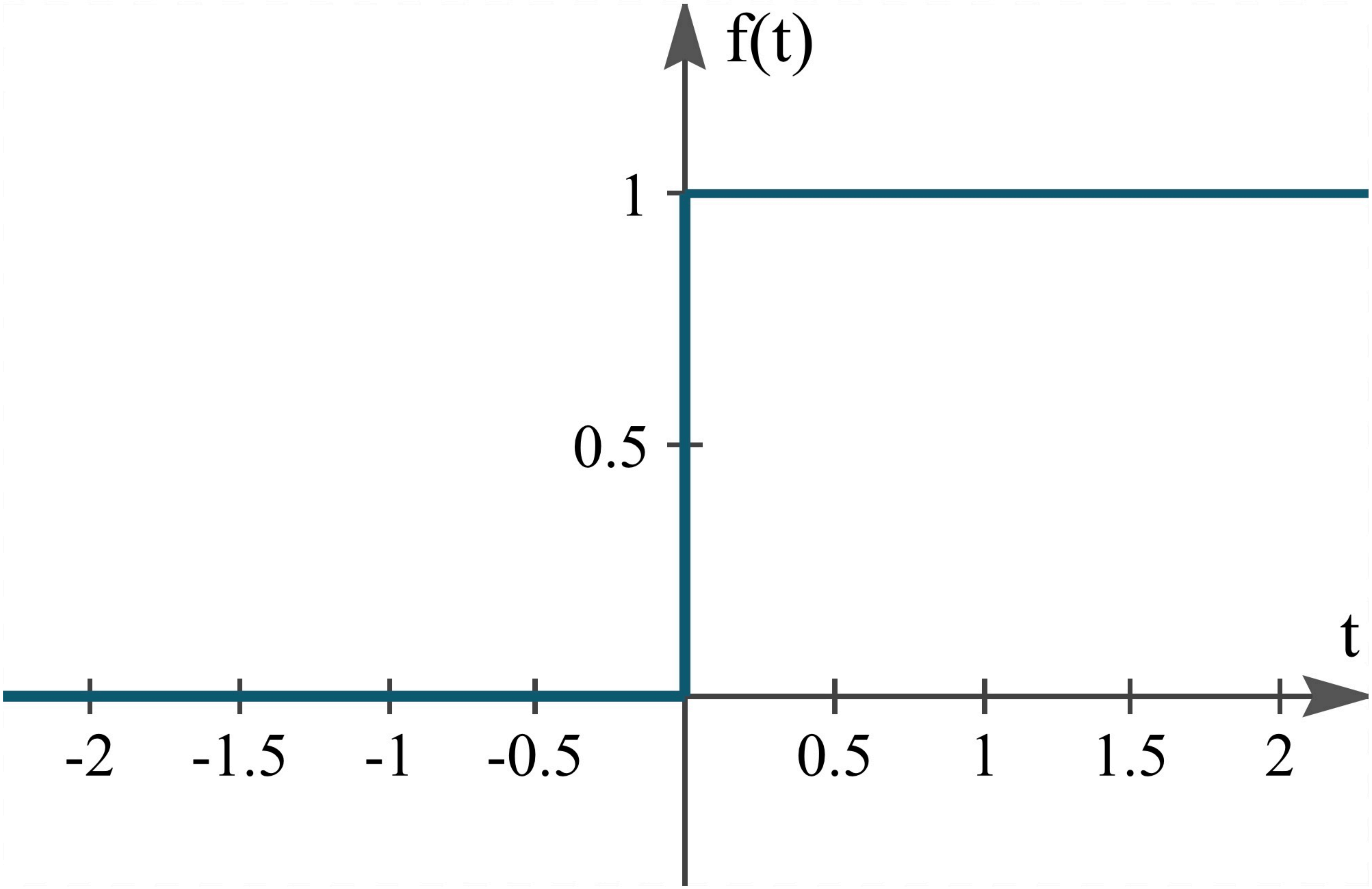
A biological neuron and its mathematical modelling



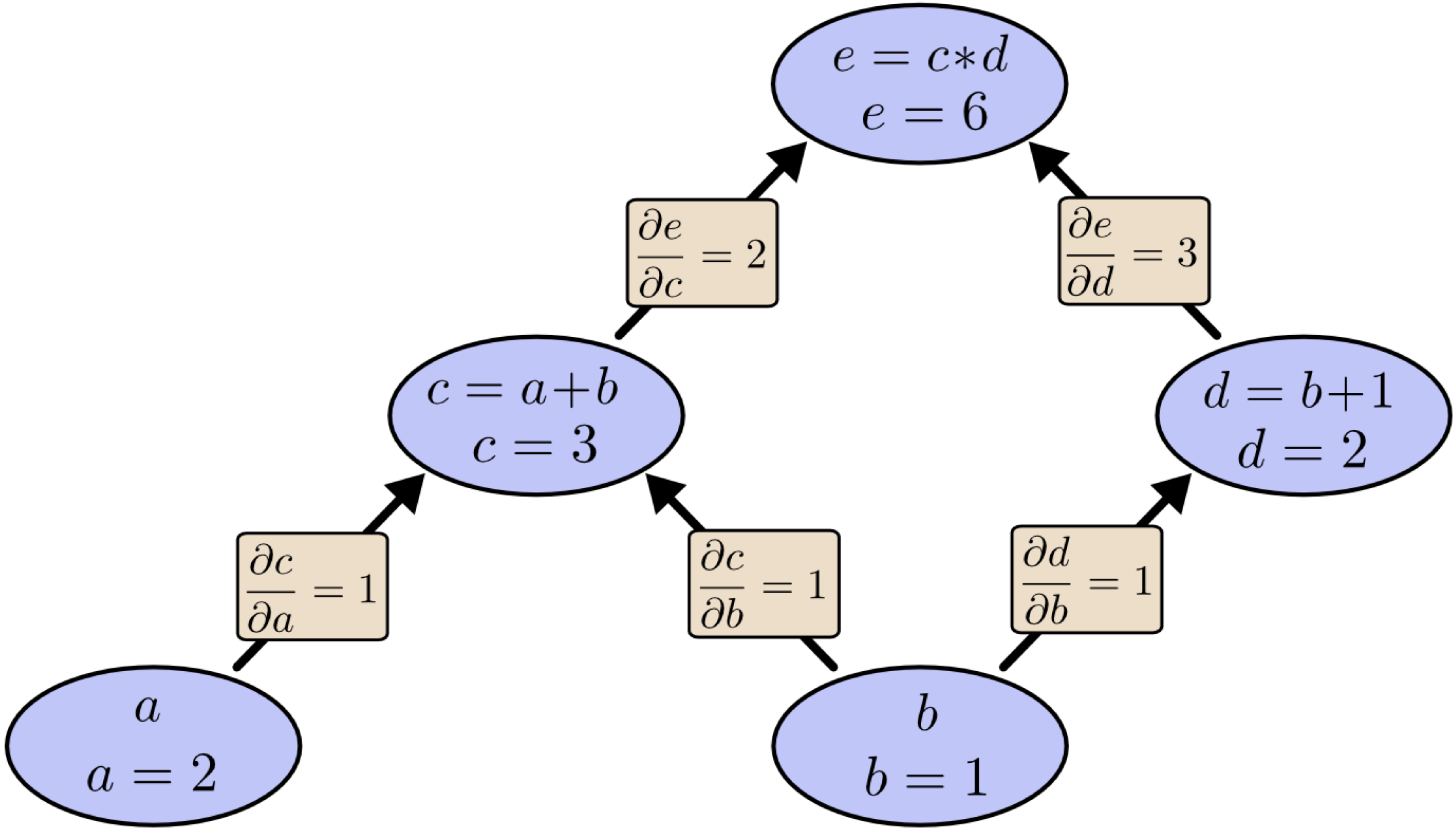
The XOR problem



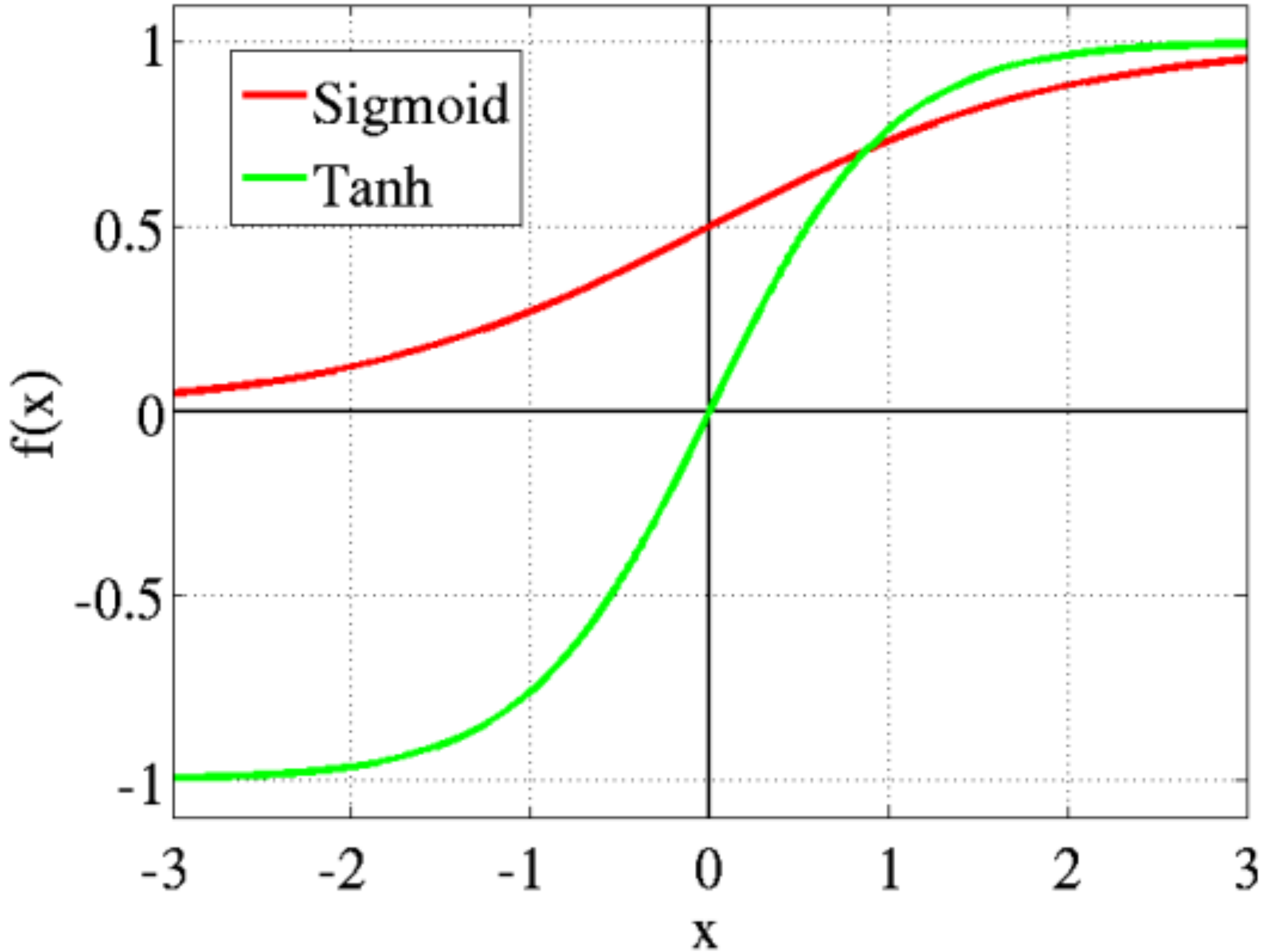
A perceptron with an activation function



The step function

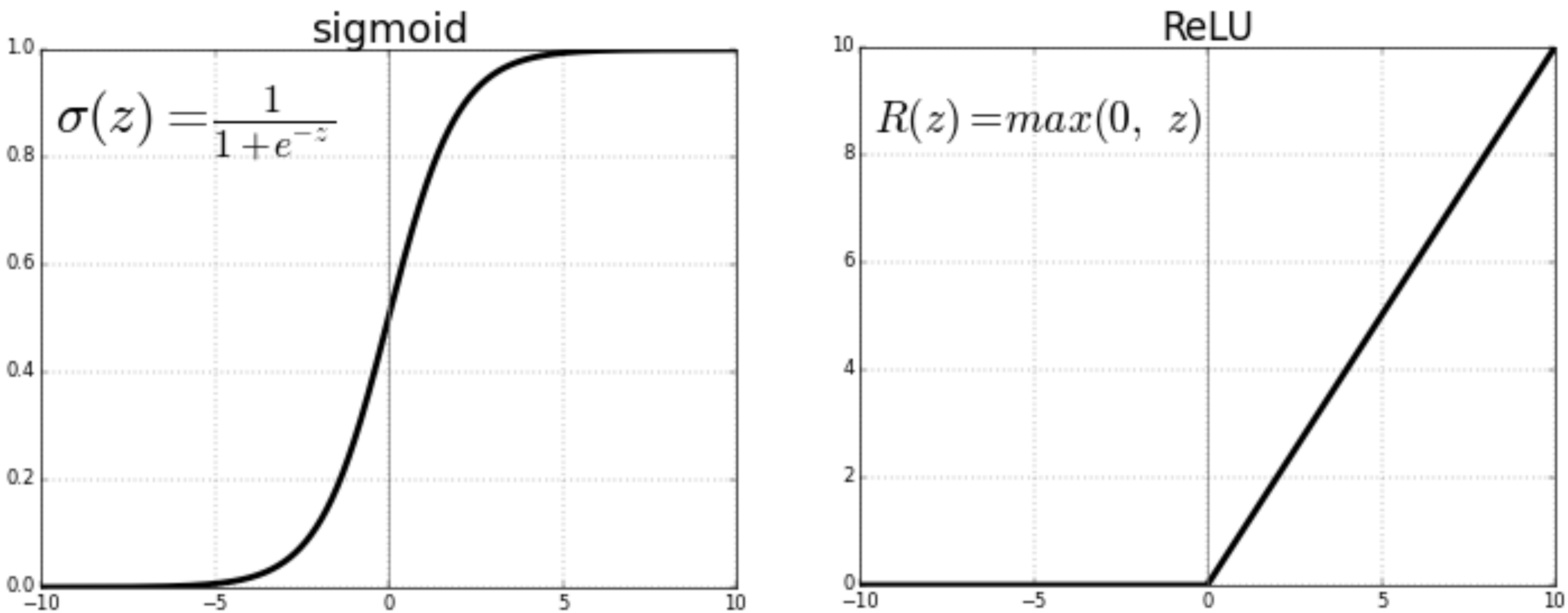


Backpropagation

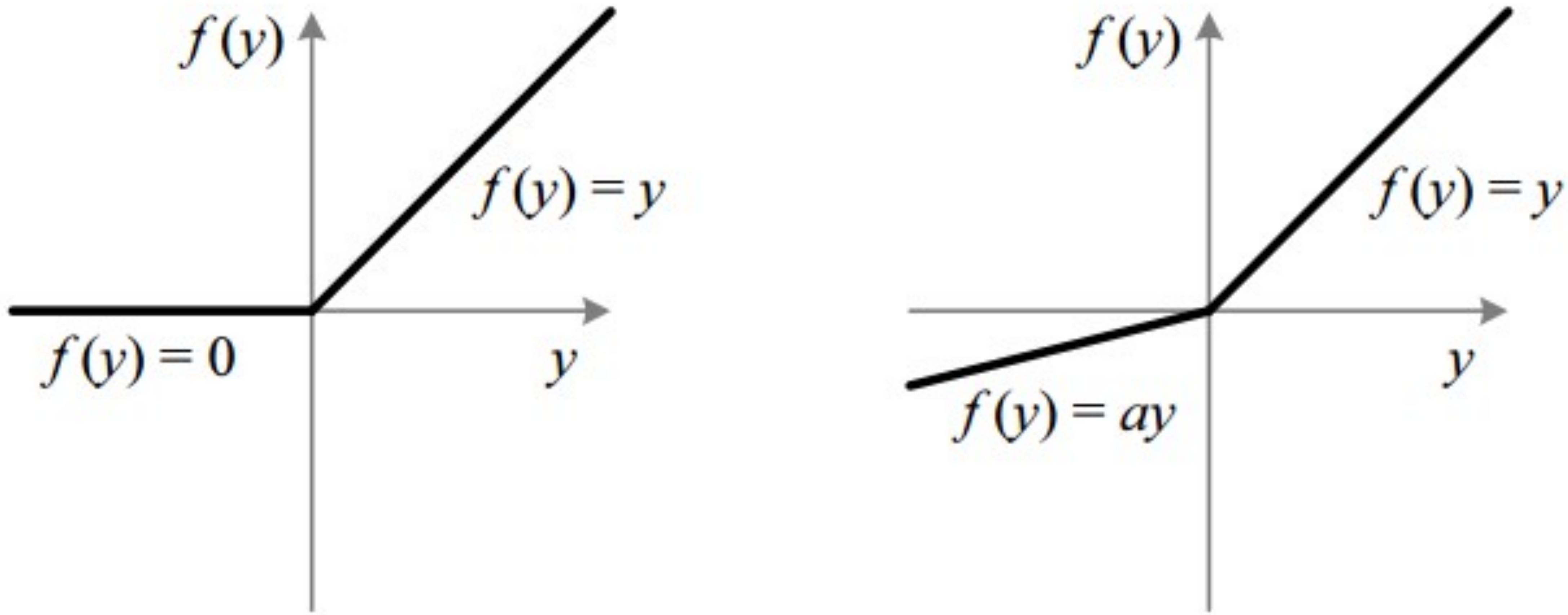


Sigmoid and Tanh

https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6



Sigmoid and ReLU



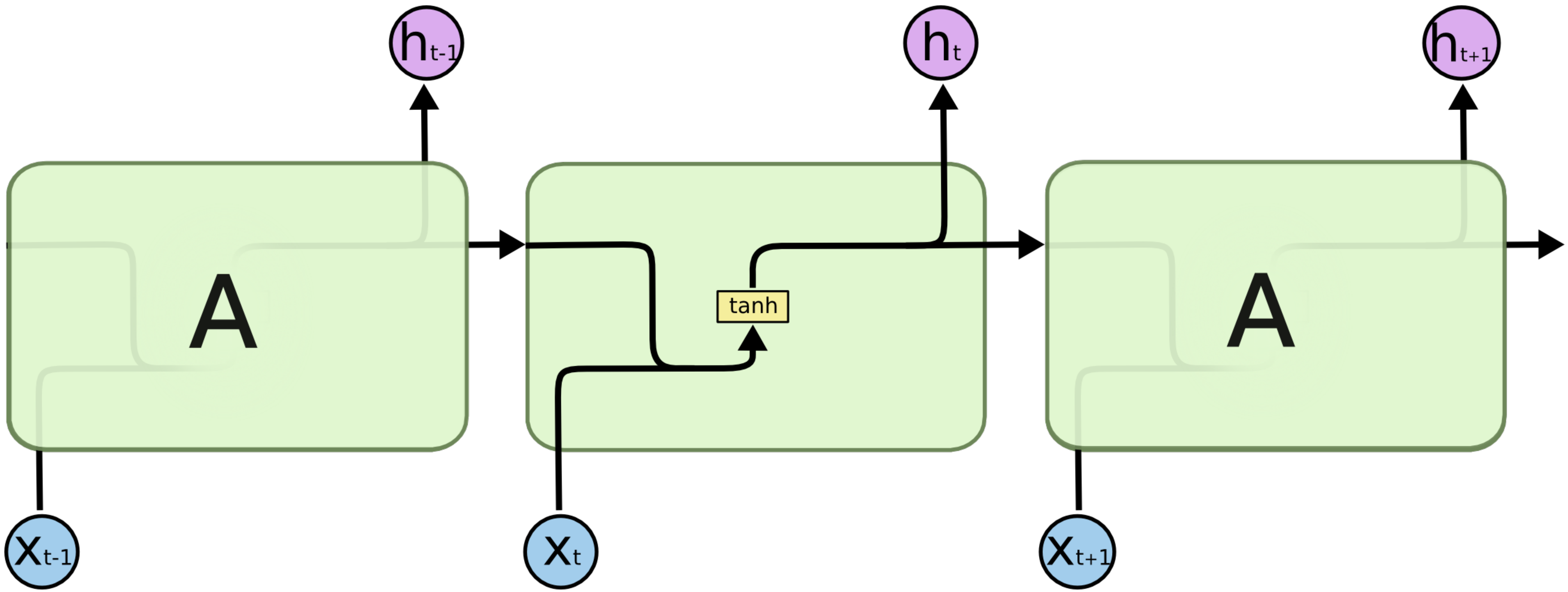
ReLU and Leaky ReLU

The story of CNNs

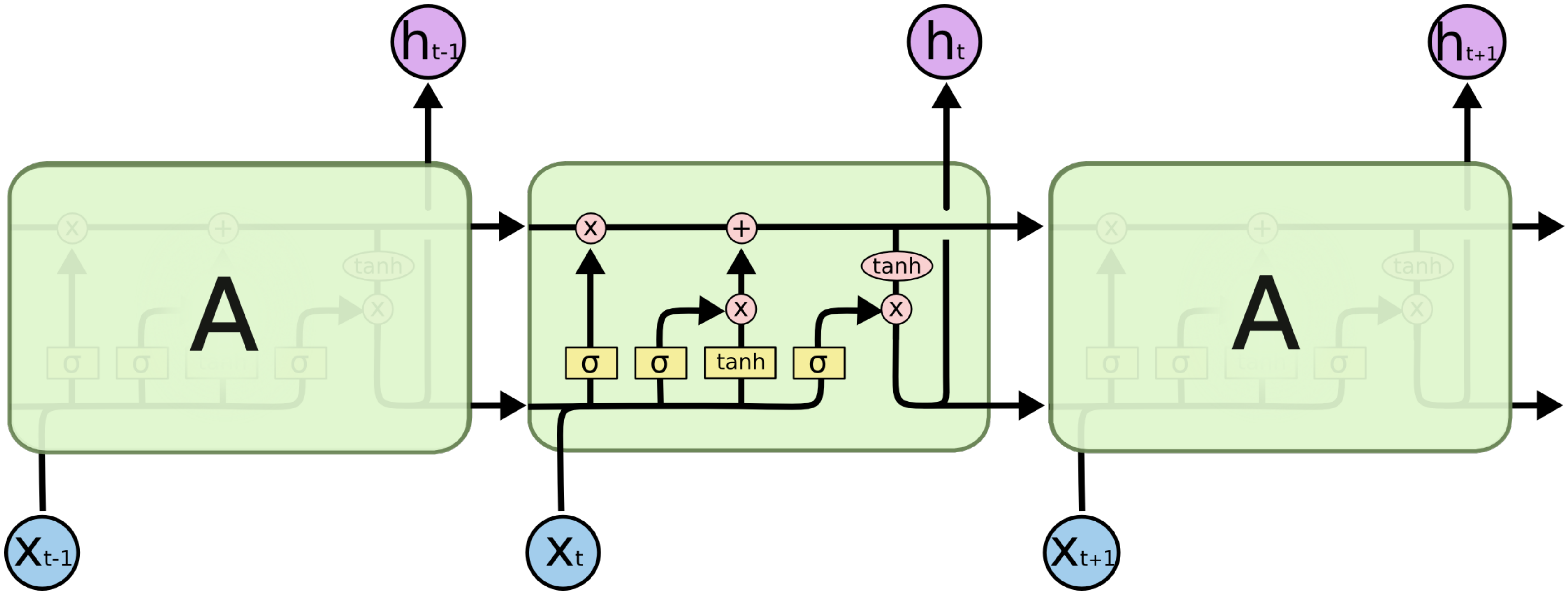
- **Perceptrons**!
- A single perceptron cannot solve **the XOR problem**; and for perceptrons to go multi-layer, some form of **nonlinearity** is needed
- The **step function** as a nonlinear activation function!
- For **backpropagation**, a **differentiable** activation function is needed
- **Sigmoid** and **Tanh** functions!

The story of CNNs

- Sigmoid and tanh activations suffer from **gradient saturation** and **gradient vanishing**
- **ReLU**!
- ReLU suffers from **dying ReLUs**.
- **Leaky ReLU**!



RNN

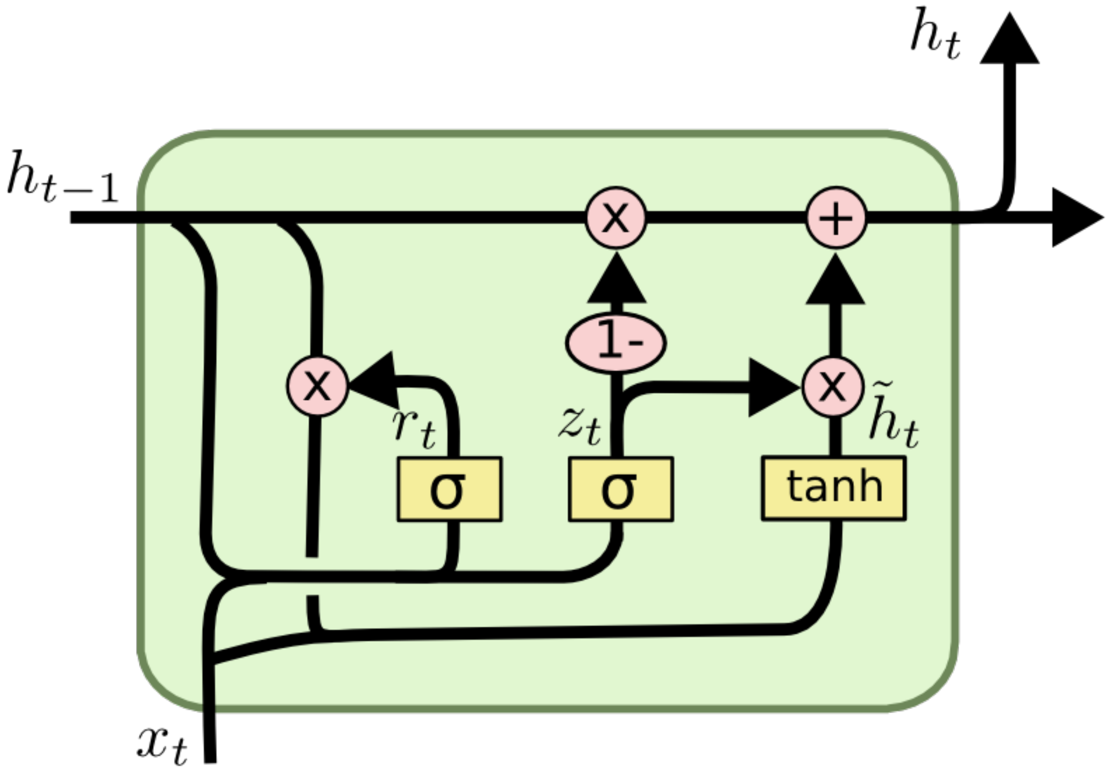


LSTM

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

$$\mu_t = \alpha \cdot \mu_{t-1} + (1 - \alpha) \cdot v_t$$

Leaky units, or Exponential Moving Average



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

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

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

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

GRU

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

$$\nabla[\tanh(\mathbf{X}) \otimes \sigma(\mathbf{X})] = \tanh'(\mathbf{X})\nabla\mathbf{X} \otimes \sigma(\mathbf{X}) + \sigma'(\mathbf{X})\nabla\mathbf{X} \otimes \tanh(\mathbf{X}).$$

Gated Tanh Units

$$\nabla[\mathbf{X} \otimes \sigma(\mathbf{X})] = \nabla \mathbf{X} \otimes \sigma(\mathbf{X}) + \mathbf{X} \otimes \sigma'(\mathbf{X}) \nabla \mathbf{X}$$

Gated Linear Units

The story of RNNs

- **RNN**s for sequential, variable-length data!
- As a cell shares weight for all timesteps, its easy for **gradient vanishing and exploding** to happen
- **LSTM**s with gradient super highway!
- LSTMs are unnecessarily complex
- **GRU**s for decreased time resolution!

The story

- But sigmoid and tanh keeps causing **gradient saturation**, and ReLU cannot be used because of gradient exploding
- **Gated Linear Units**!

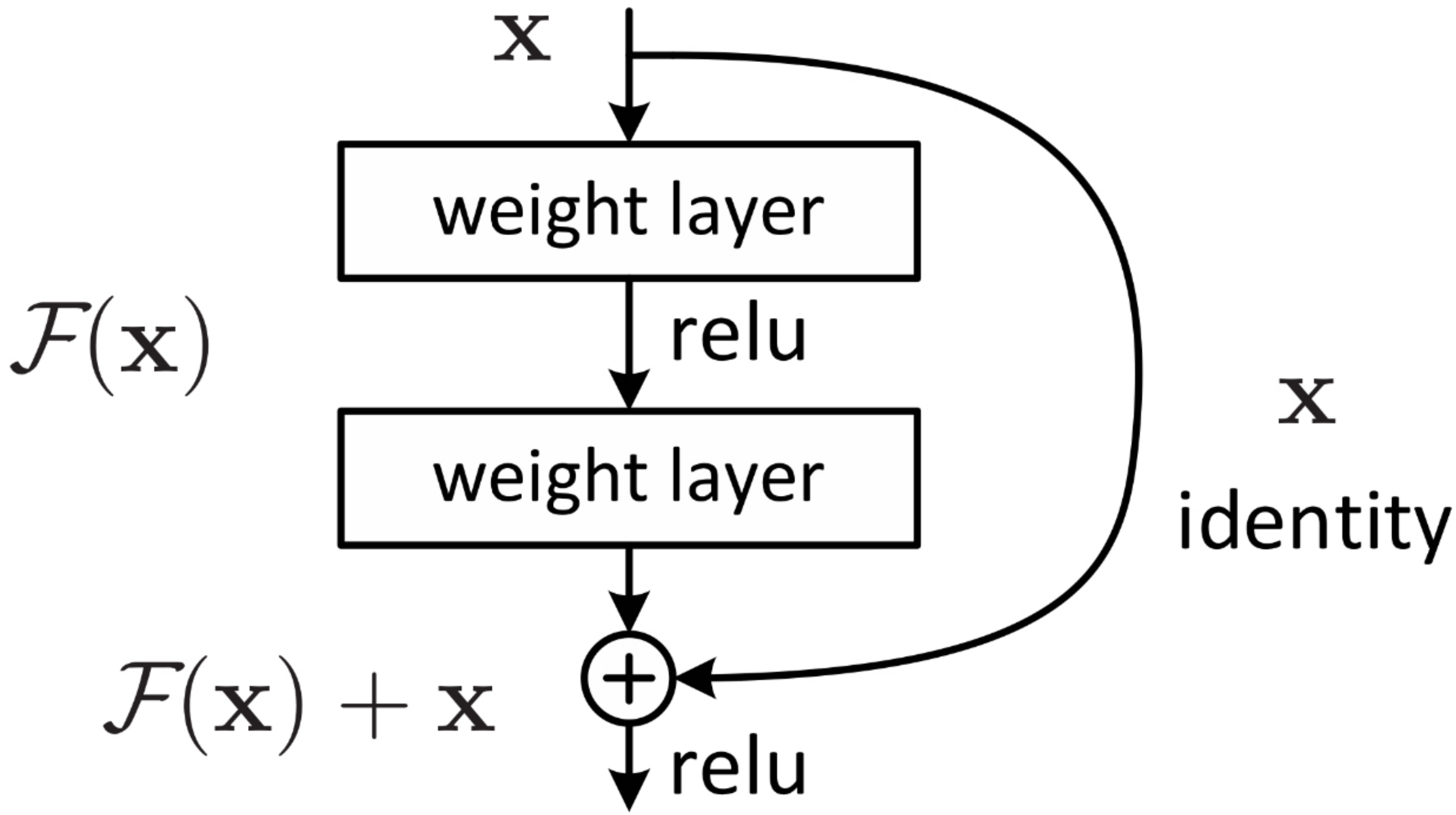
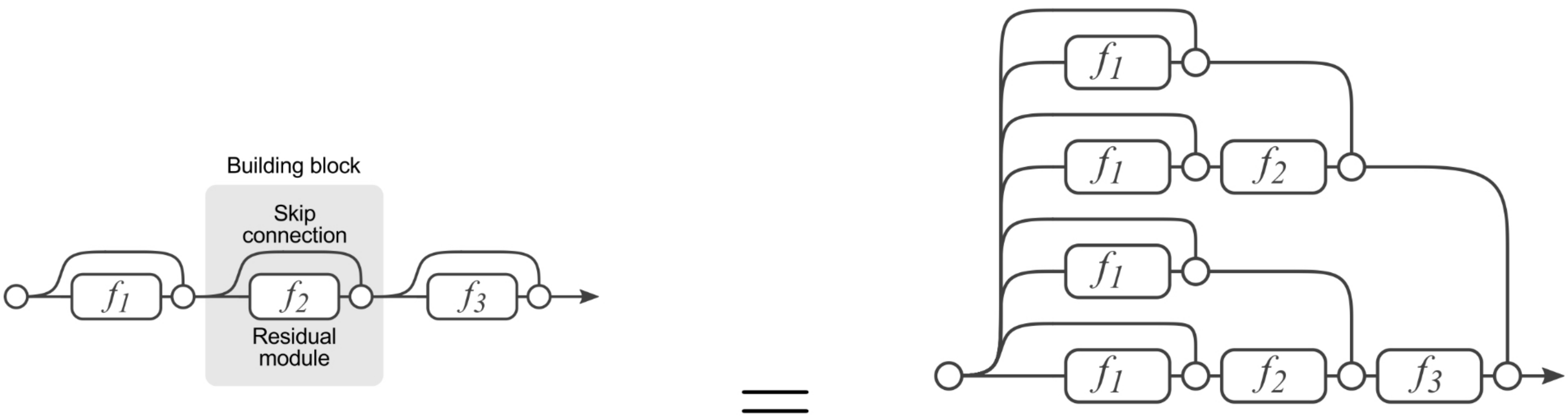


Figure 2. Residual learning: a building block.

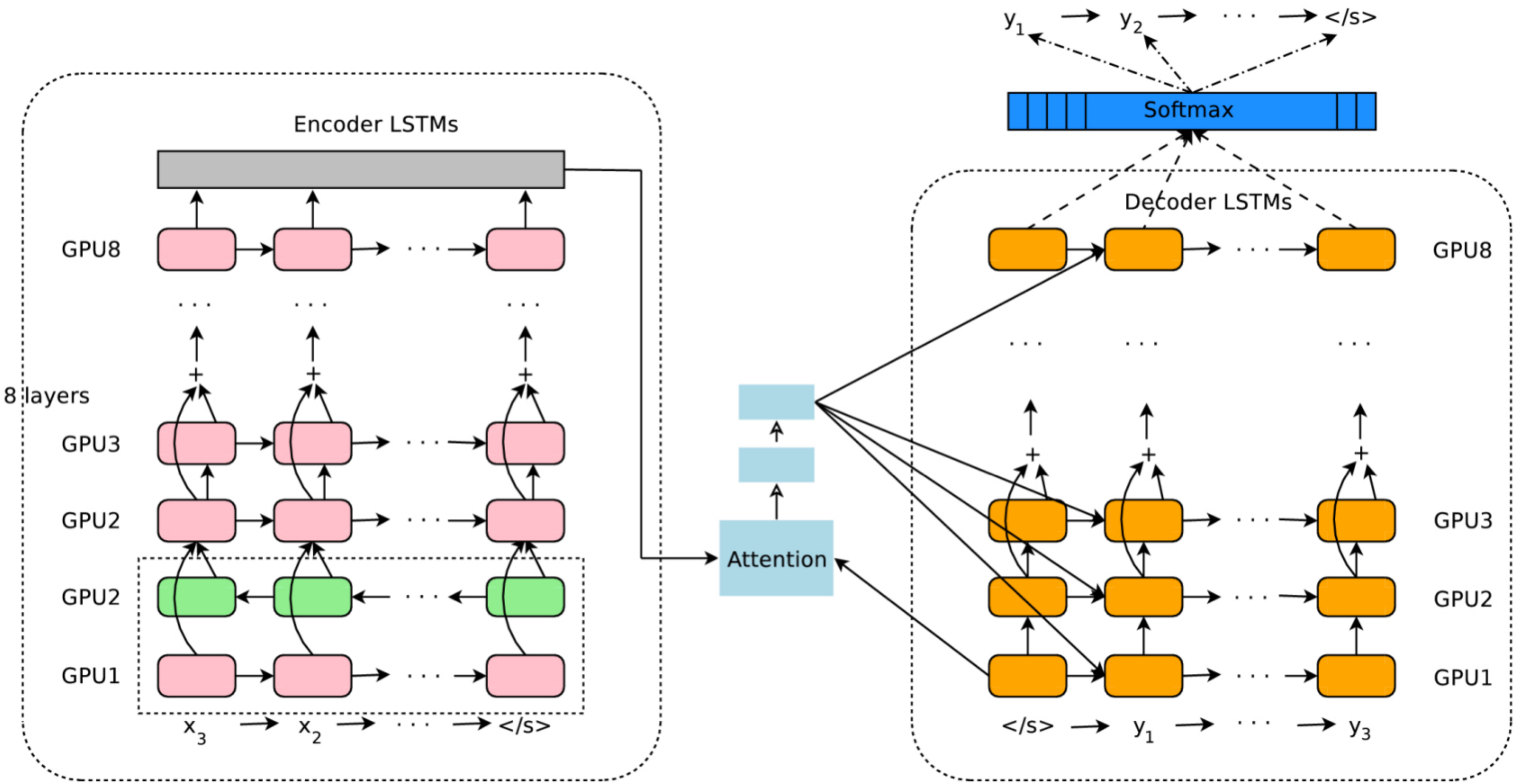
Residual connections in CNNs



(a) Conventional 3-block residual network

(b) Unraveled view of (a)

Residual networks allowing stochastic layer depths



Stack-wise residual connections in RNNs

Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation (2016)

In **zoneout**, the values of the hidden state and memory cell randomly either maintain their previous value or are updated as usual. This introduces stochastic identity connections between subsequent time steps:

$$c_t = d_t^c \odot c_{t-1} + (1 - d_t^c) \odot \left(f_t \odot c_{t-1} + i_t \odot g_t\right)$$
$$h_t = d_t^h \odot h_{t-1} + (1 - d_t^h) \odot \left(o_t \odot \tanh\left(f_t \odot c_{t-1} + i_t \odot g_t\right)\right)$$

Time-wise residual connections in RNNs (Zoneout)

Zonouts: Regularizing RNNs by Randomly Preserving Hidden Activations (2017)

The story of residual connections

- Deep neural networks are difficult to train
- **ResNet**!
- Residual connections in RNNs?
  - Stack-wise residual connections: Google NMT (2016)
  - Time-wise residual connections: Zoneout (2017)