Martin Mundt, Dr. Iuliia Pliushch, Prof. Dr. Visvanathan Ramesh

Pattern Analysis & Machine Intelligence Praktikum: MLPR-SS21

Week 6: Recurrent Neural Networks (RNNs)

Recurrent Neural Network: Cells

- Activations of previous time-step influence next time-step
- In addition now also weights operating between A_t
- Weights shared across t

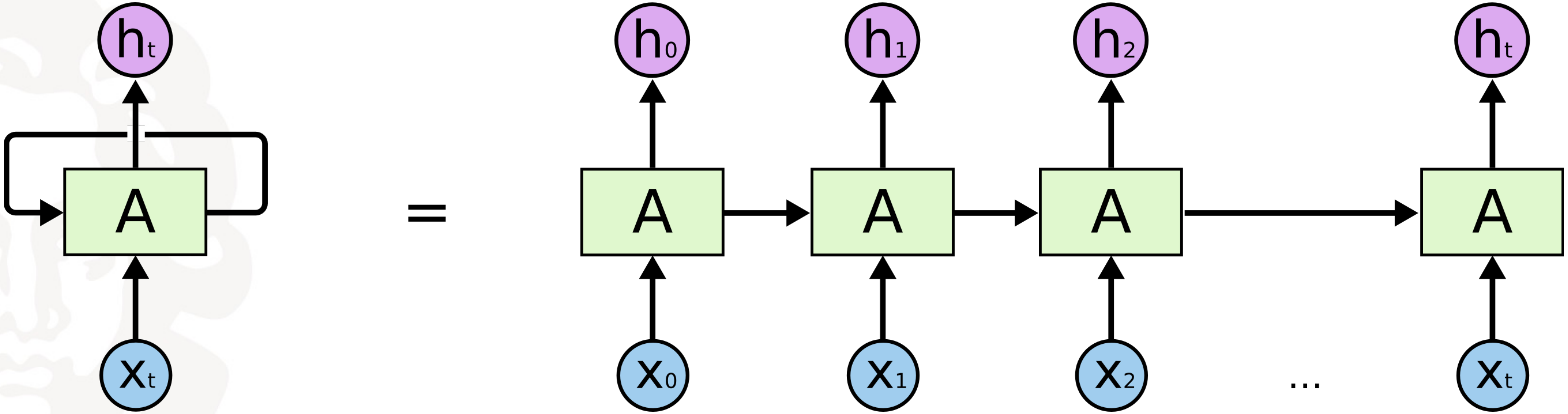


Image taken from: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Recurrent Neural Networks: Cells

- “h” is the hidden state after a tanh activation function. There can be multiple such cells stacked to a deep network.
- For the final network prediction another layer needs to be added. See next slide
- 2 sets of weights: input to hidden, hidden to hidden (& then hidden to next cell hidden/output etc.).

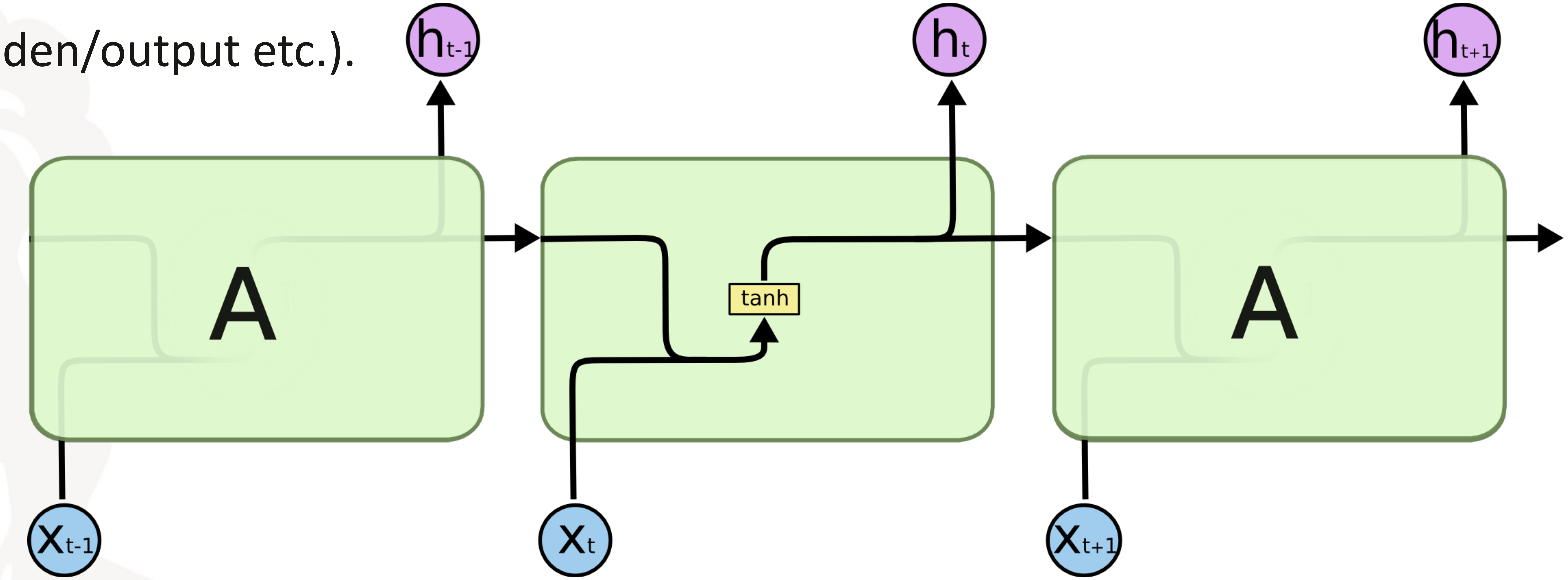


Image taken from: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Recurrent Neural Networks: Training

- Very similar to backpropagation in feedforward NNs (here a 1 hidden-layer MLP)
- Key difference: Sum gradients for W at each time step
- PyTorch is going to handle backprop for us: see reference below for full derivation

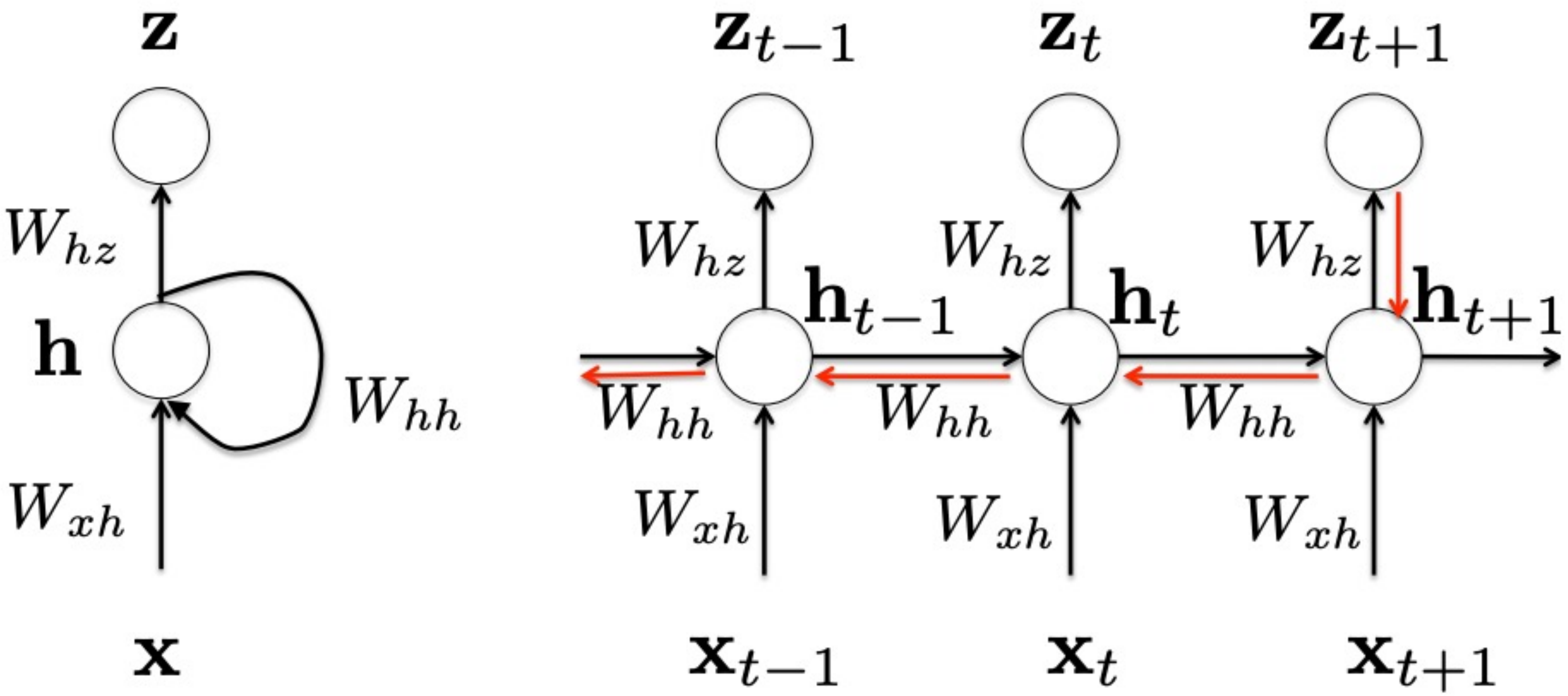


Image taken from "A Gentle Tutorial of Recurrent Neural Network with Error Backpropagation"
https://arxiv.org/pdf/1610.02583.pdf

Recurrent Neural Networks: Tasks

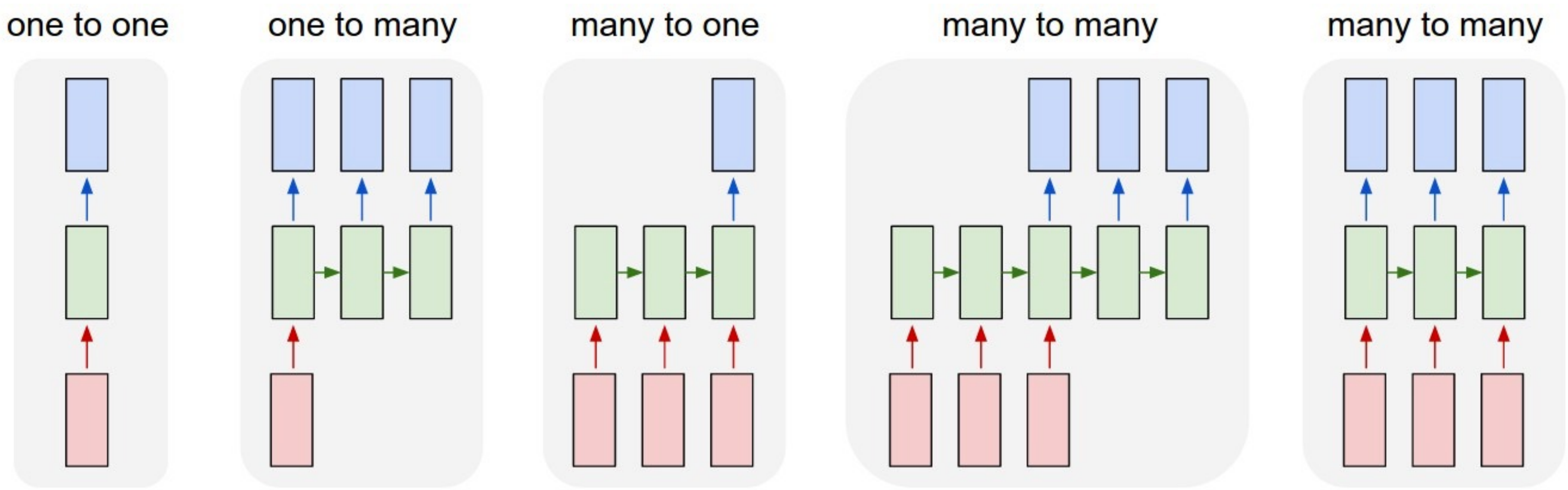


Image taken from: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Examples:

1. One to one: image classification
2. One to many: image captioning
3. Many to one: sentiment analysis
4. Many to many: language translation
5. Many to many: video classification

Recurrent Neural Networks: Long Range Dependencies

”The **cat** just had plenty of delicious food and **is** now full”.

”The **cats** just had plenty of delicious food and **are** now full.”

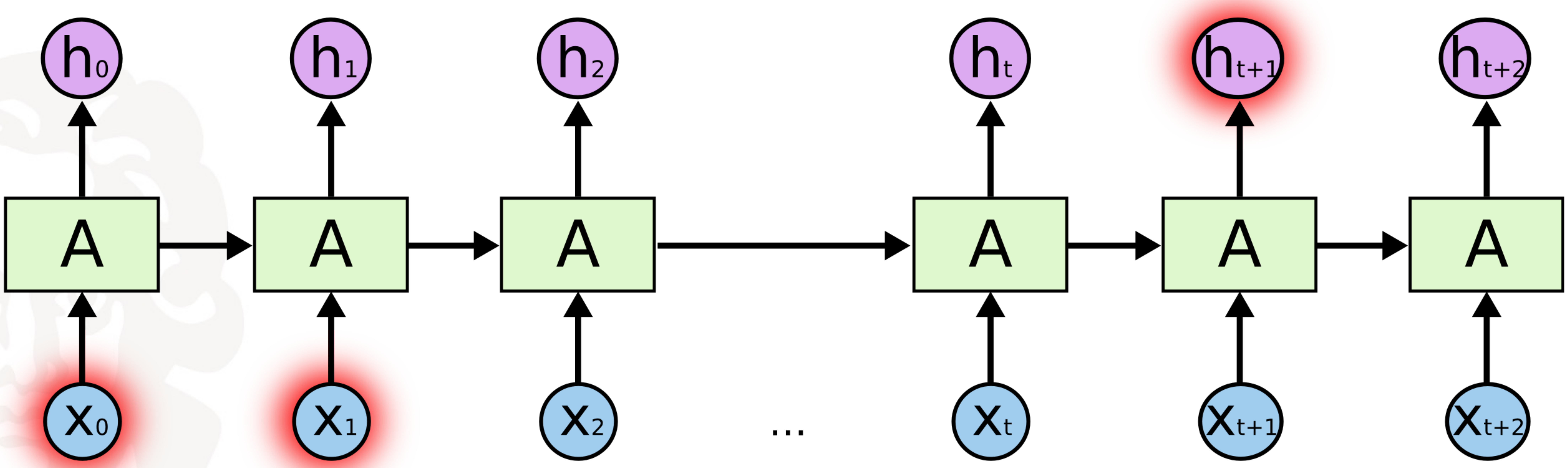


Image taken from: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short Term Memory (LSTM)

- This part follows http://colah.github.io/posts/2015-08-Understanding-LSTMs/ because the illustrations simply cannot be done any better
- Instead of one layer, have four with different responsibilities:
- Cell state and three gates to interact with: forget gate, input gate, output gate

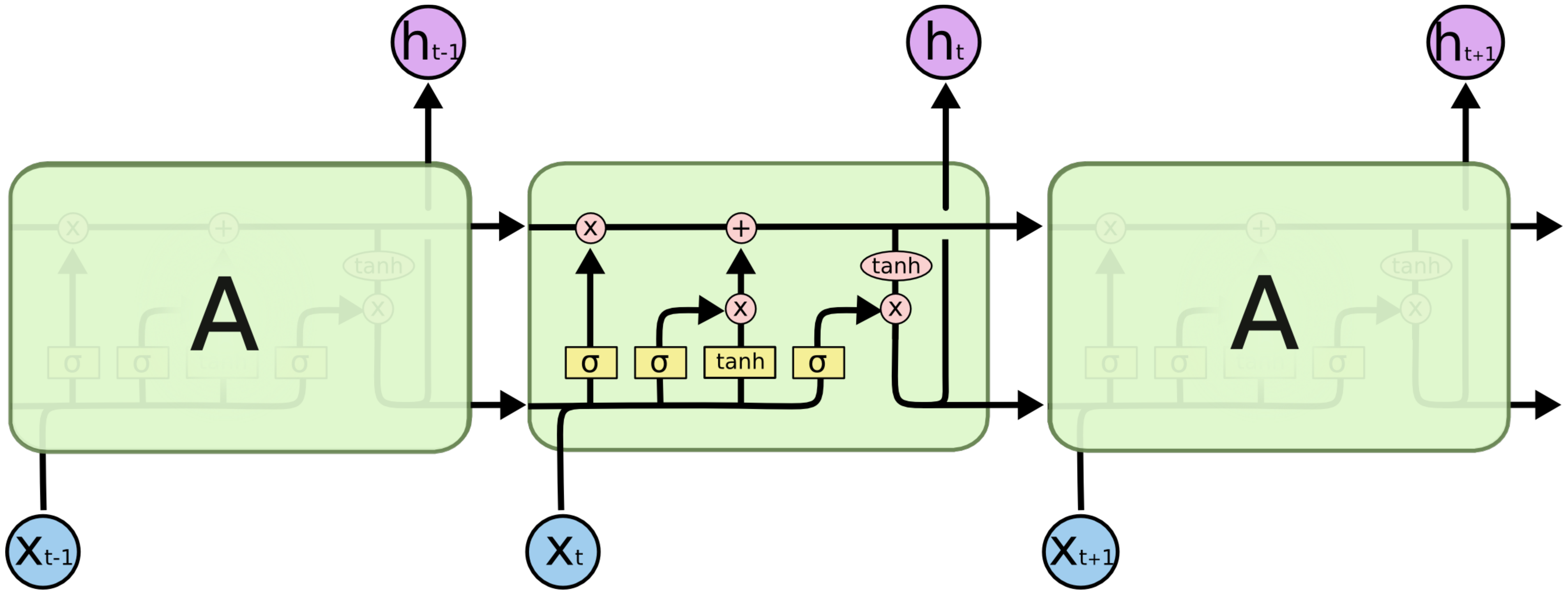


Image taken from: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LTSM: Cell state

- Lets information flow through the entire chain
- Only minor linear interactions
- Information will get controlled through additional gating

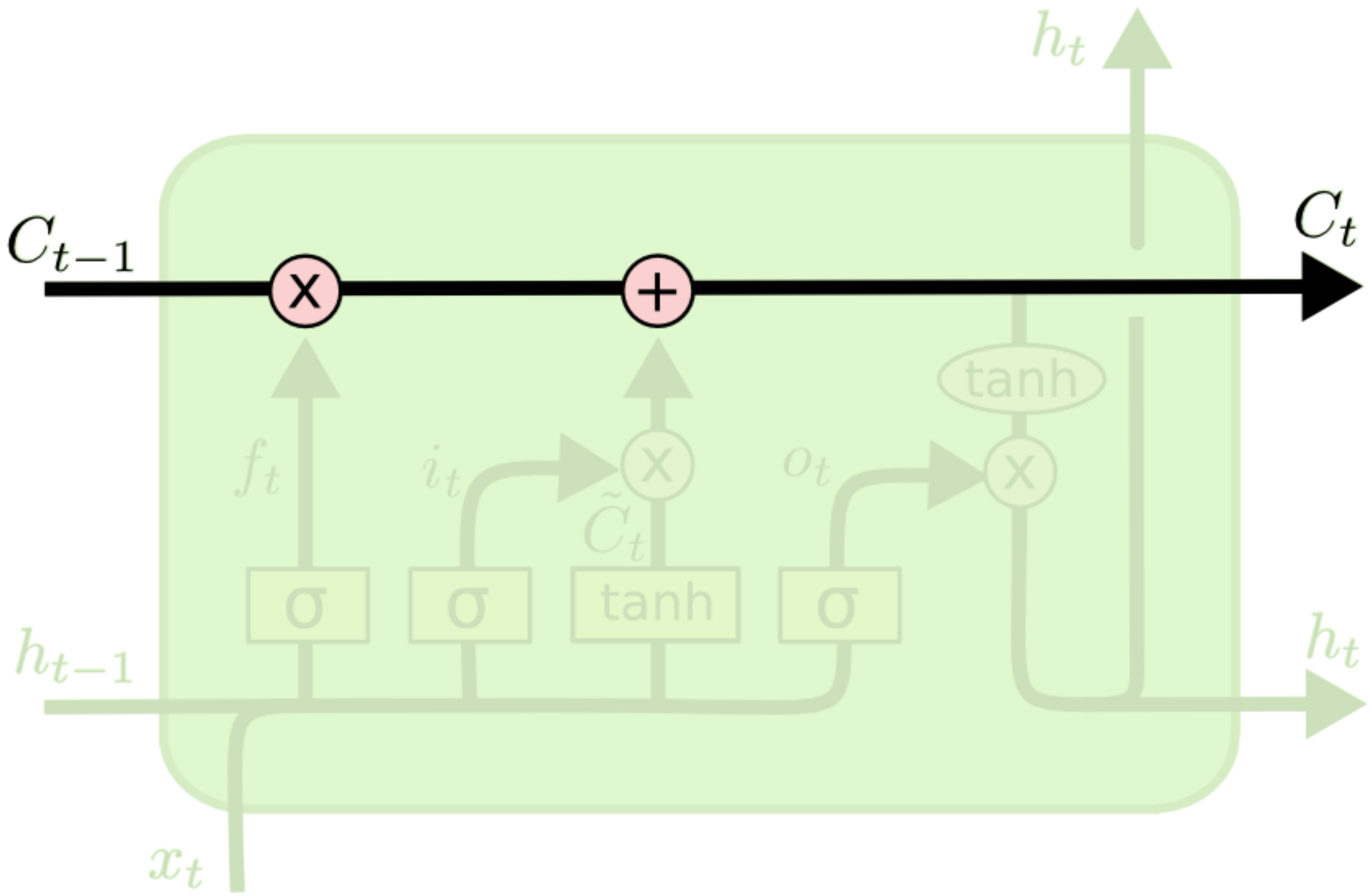
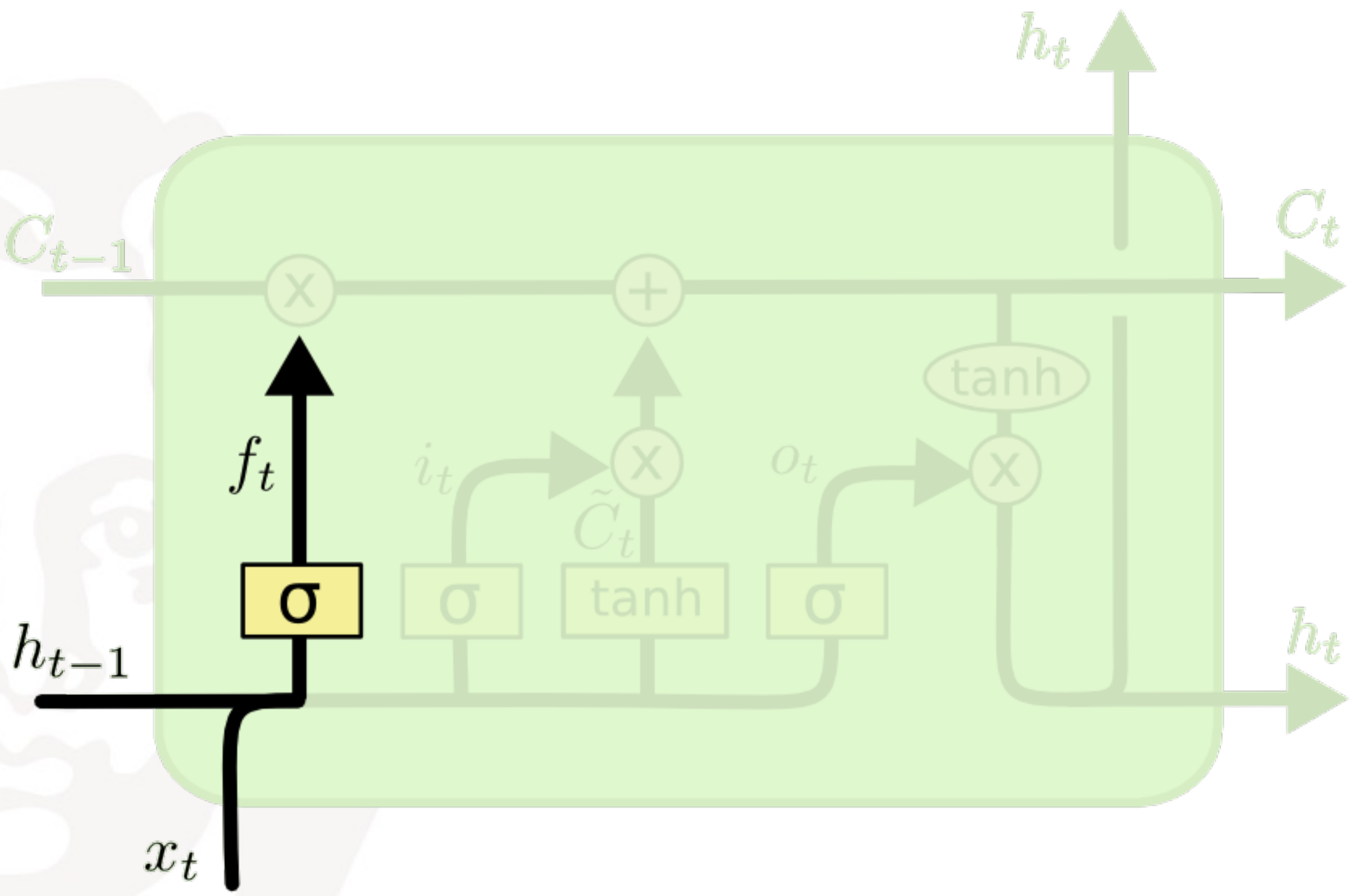


Image taken from: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTMs: Forget gate

- The forget gate multiplies a Sigmoid output (0 to 1) with the current cell state to allow information to pass through fully or fully forget it in the two extremes

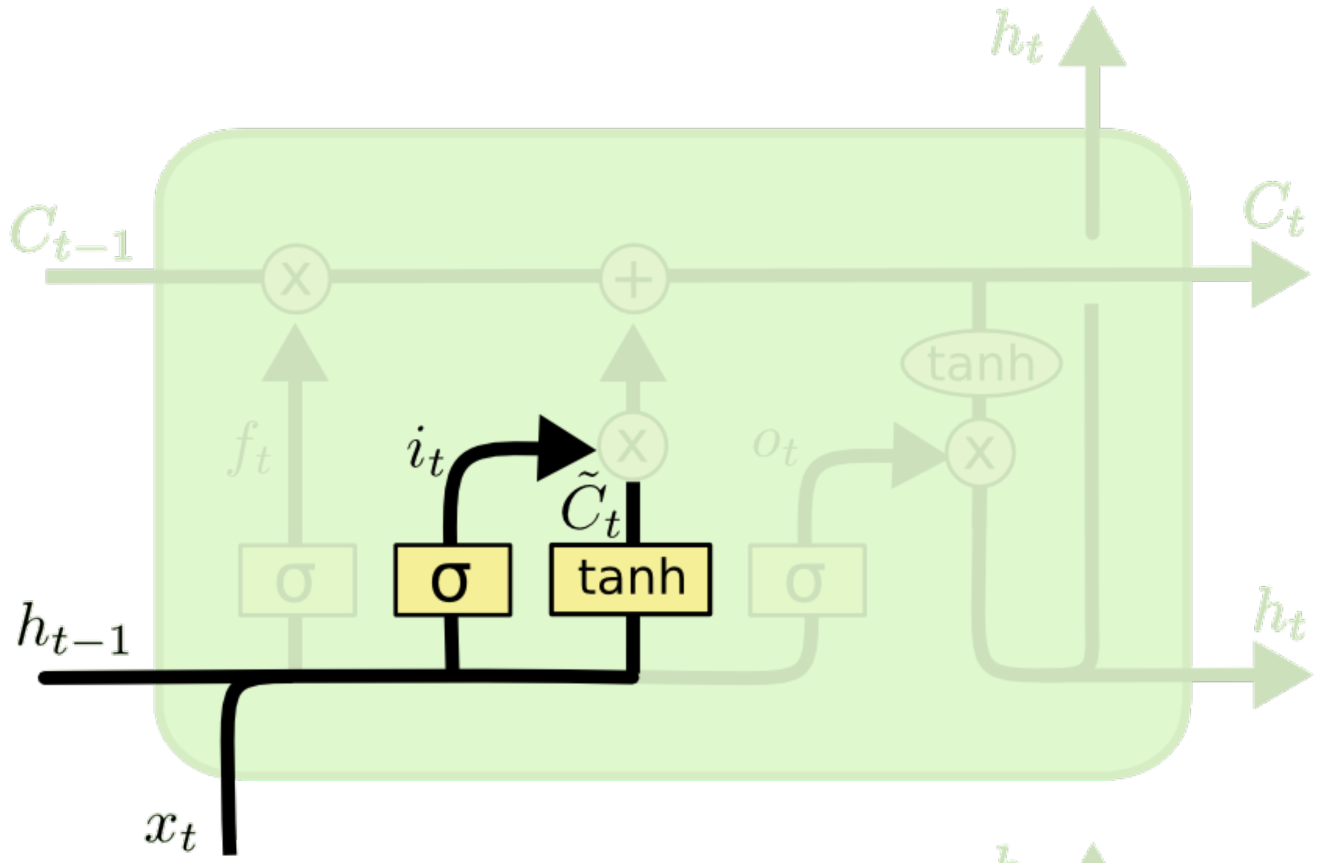


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

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

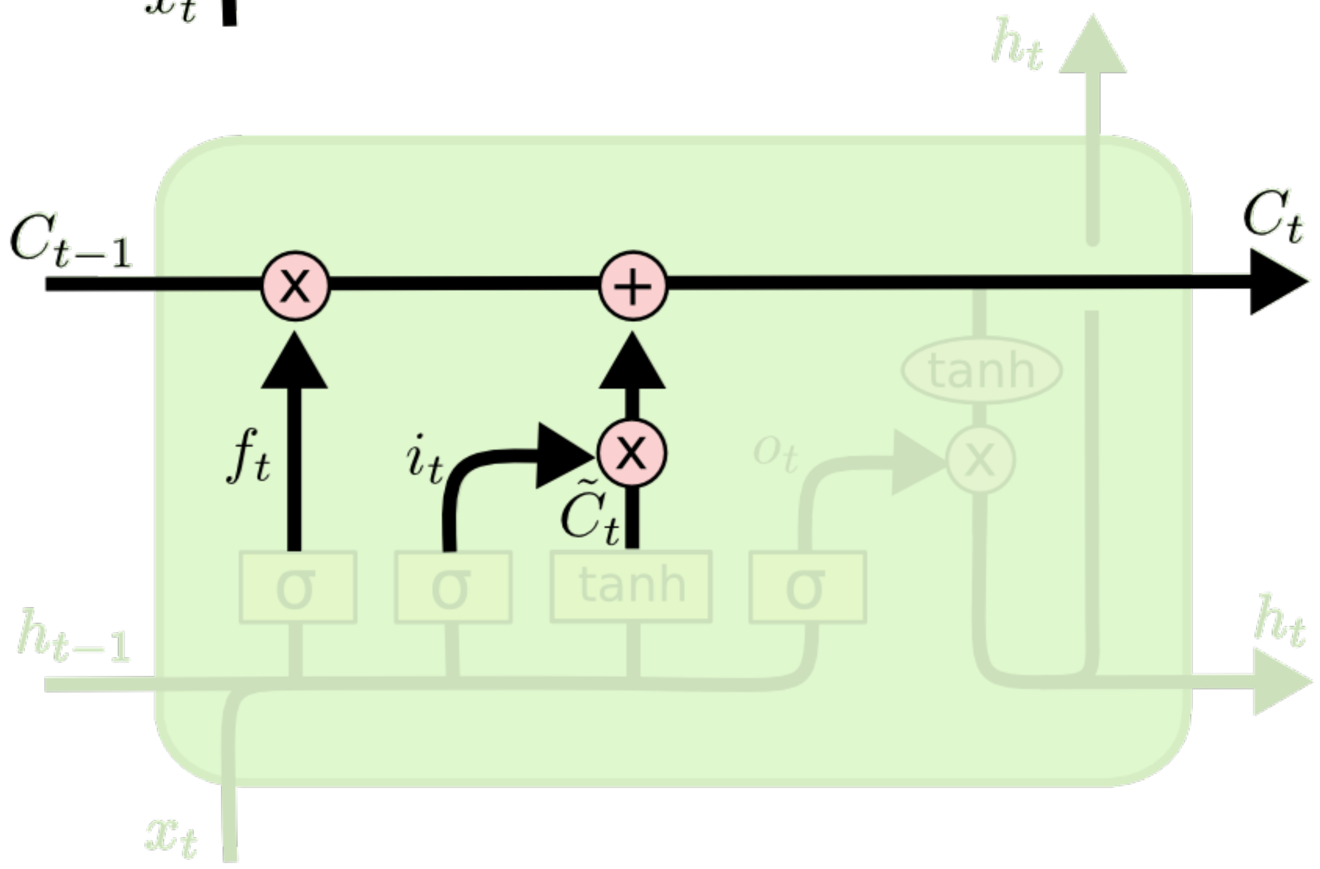
LSTMs: Input gate

- The input gate multiplies a Sigmoid output (0-1) with the new information to decide what new infomation to add to the cell state & update it



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

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

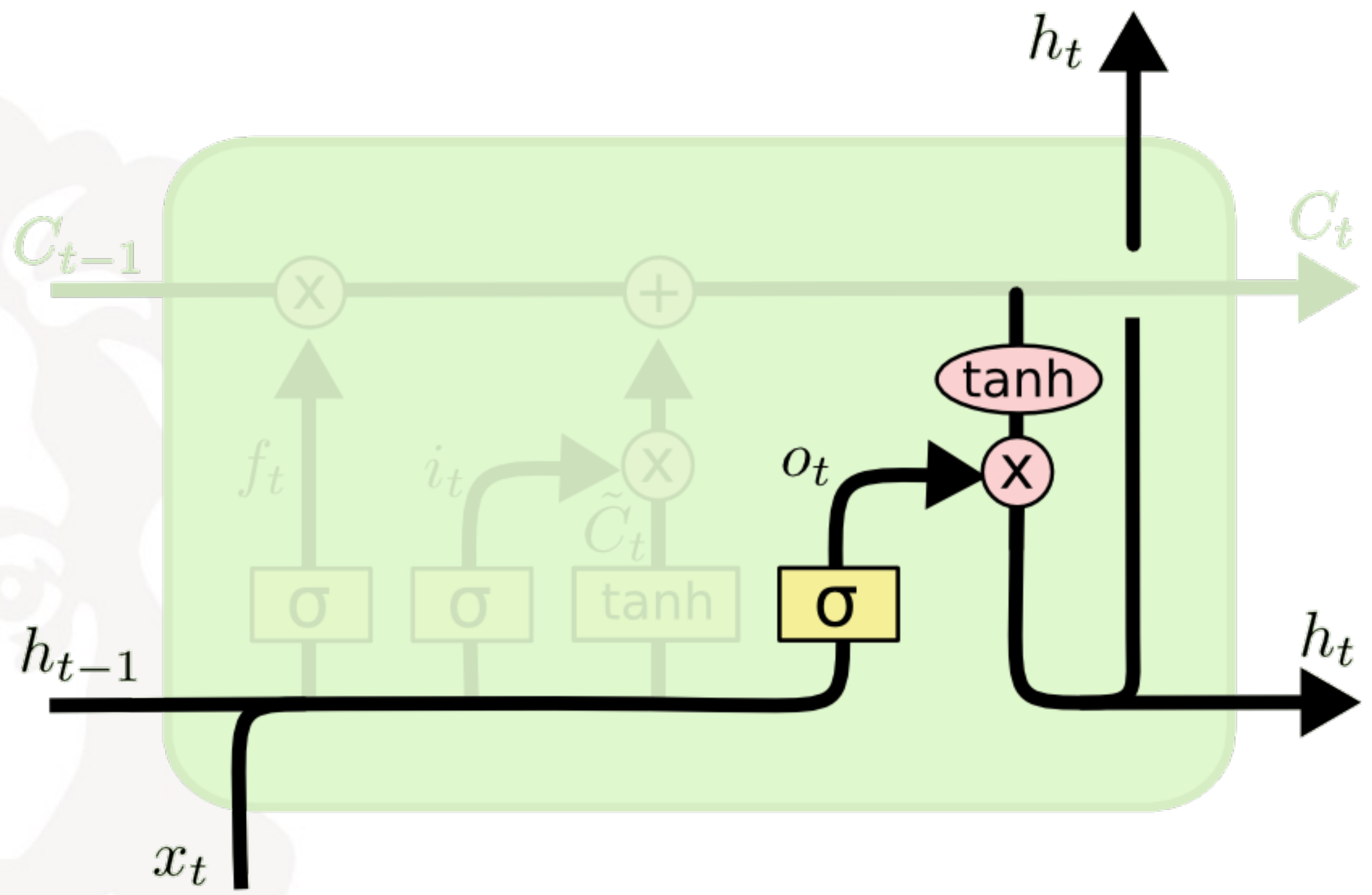


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

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

LSTMs: Output gate

- The output gate takes the updated cell state and the input and decides what to give as output



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

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

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

LSTMs: Overview & Training

- Training like RNN, backpropagation through time, but with more parameters
- PyTorch has a convenient LSTM cell implementation “nn.LSTM()“ that we will use

$$\mathbf{f}_t = \sigma(W_{xf}\mathbf{x}_t + W_{hf}\mathbf{h}_{t-1} + b_f)$$
$$\mathbf{i}_t = \sigma(W_{xi}\mathbf{x}_t + W_{hi}\mathbf{h}_{t-1} + b_i)$$
$$\mathbf{g}_t = tanh(W_{xc}\mathbf{x}_t + W_{hc}\mathbf{h}_{t-1} + b_c)$$
$$\mathbf{c}_t = \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \mathbf{g}_t$$
$$\mathbf{o}_t = \sigma(W_{xo}\mathbf{x}_t + W_{ho}\mathbf{h}_{t-1} + b_o)$$
$$\mathbf{h}_t = \mathbf{o}_t \circ tanh(\mathbf{c}_t), \;\; z_t = softmax(W_{hz}\mathbf{h}_t + b_z)$$

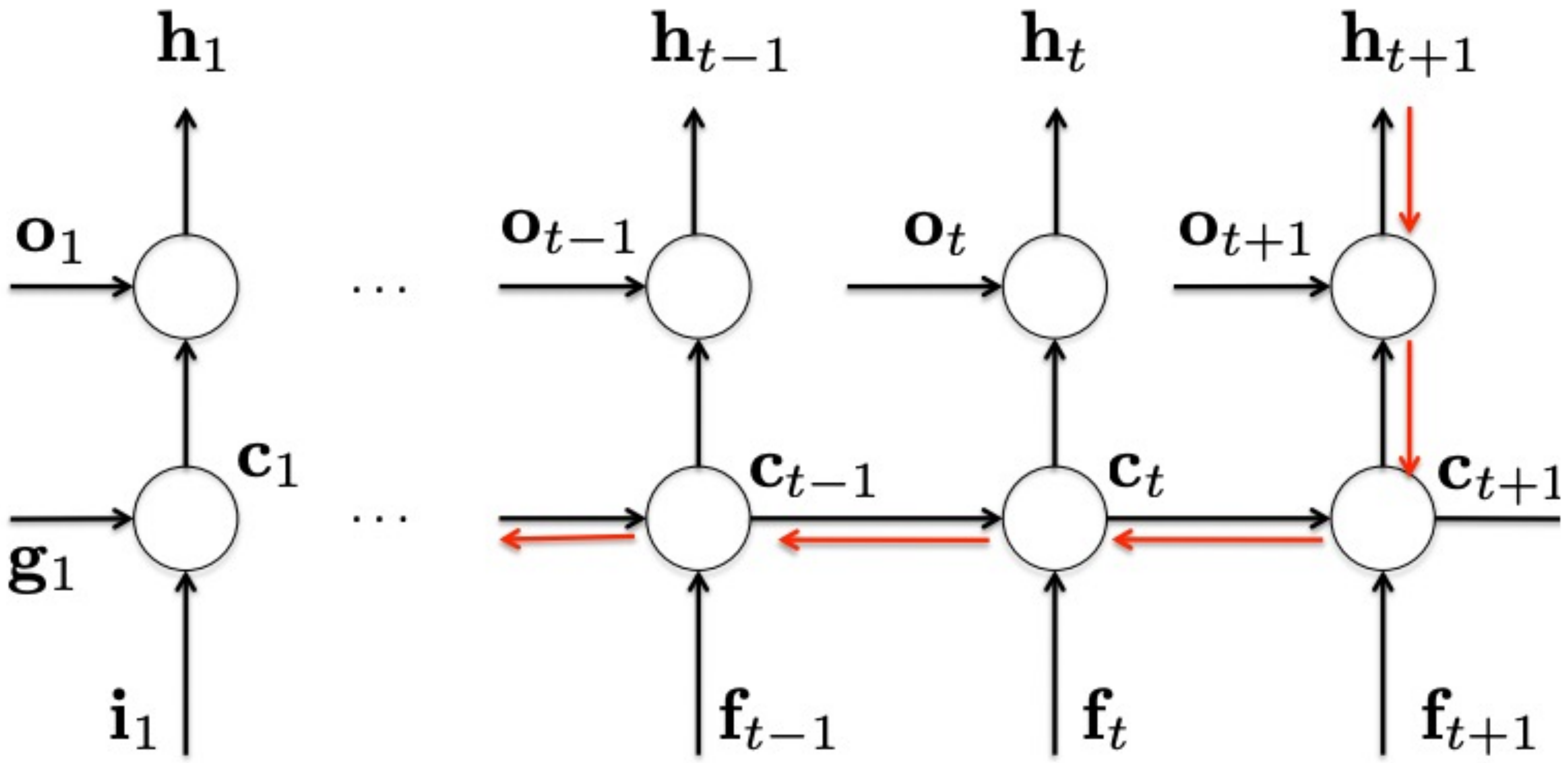


Image taken from “A Gentle Tutorial of Recurrent Neural Network with Error Backpropagation“
https://arxiv.org/pdf/1610.02583.pdf

RNNs: Are long range dependencies all we need? Bidirectional RNNs

He said: ”**Teddy** Roosevelt was a great president”

He said: ”**Teddy** bears are on sale!”

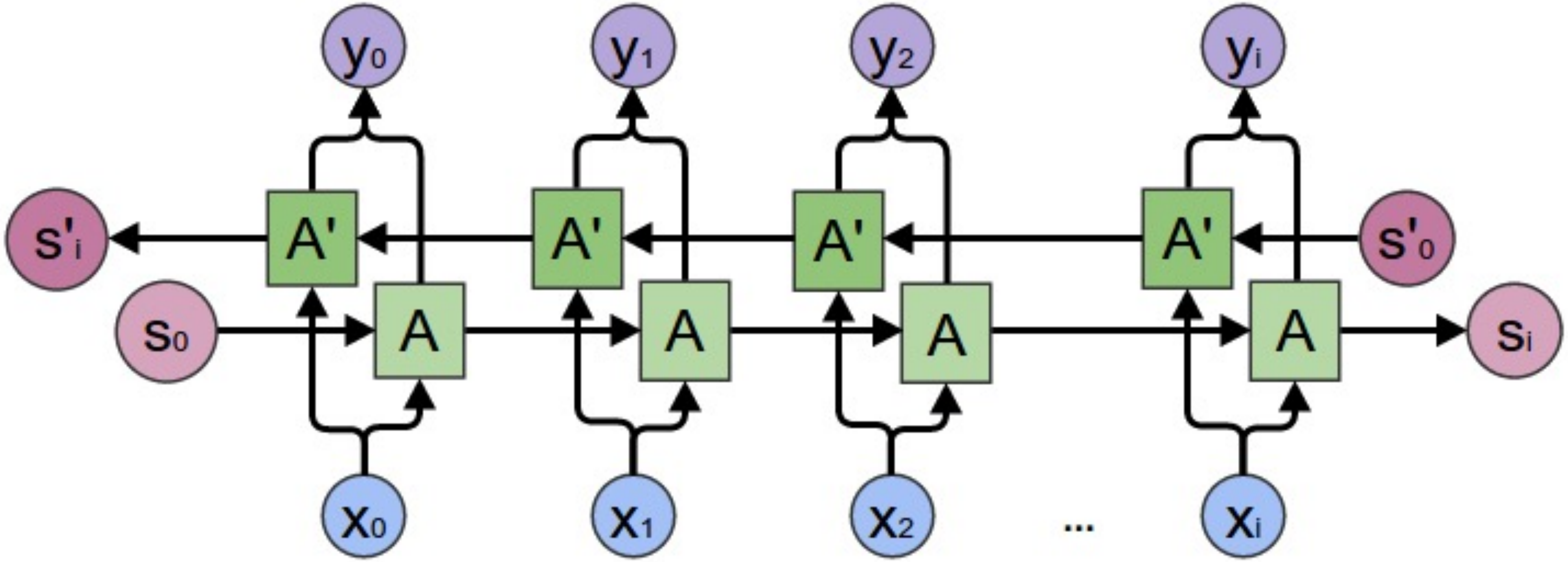


Image taken from http://colah.github.io/posts/2015-09-NN-Types-FP/

Our practice: Shakespeare poetry generation.
Character level RNN & LSTM

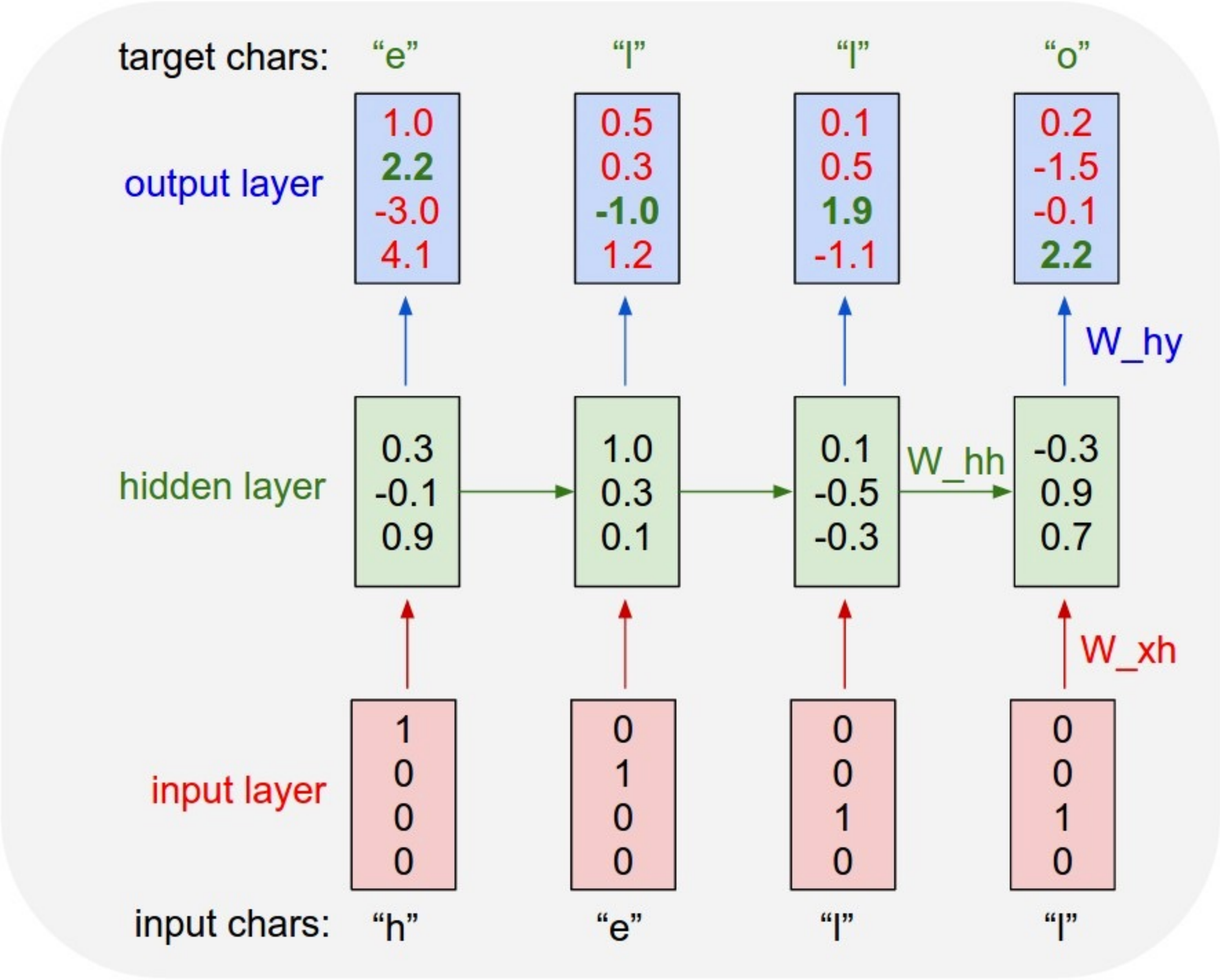


Image taken from: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Sequence Models: Tasks & Literature

- Andrew Ng’s deep learning specialization course 5: https://www.coursera.org/learn/nlp-sequence-models
- Chris Olah‘s blog: http://colah.github.io/posts/2015-08-Understanding-LSTMs/
- Andrej Karpathy’s blog: http://karpathy.github.io/2015/05/21/rnn-effectiveness/
- Stanford‘s CS224d by Richard Socher, Deep Learning for Natural Language Processing: https://cs224d.stanford.edu/