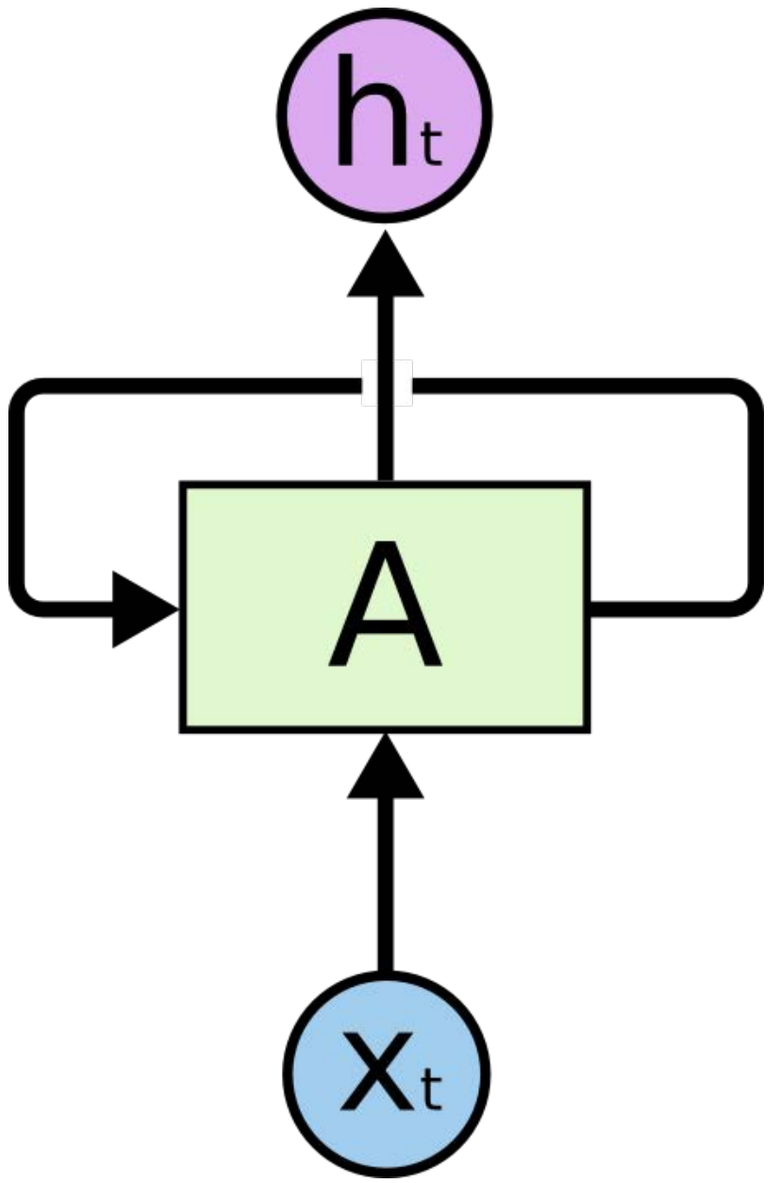
ENM 540: Data-driven modeling and probabilistic scientific computing

Lecture #9: Recurrent neural networks

Paris Perdikaris

February 13, 2018

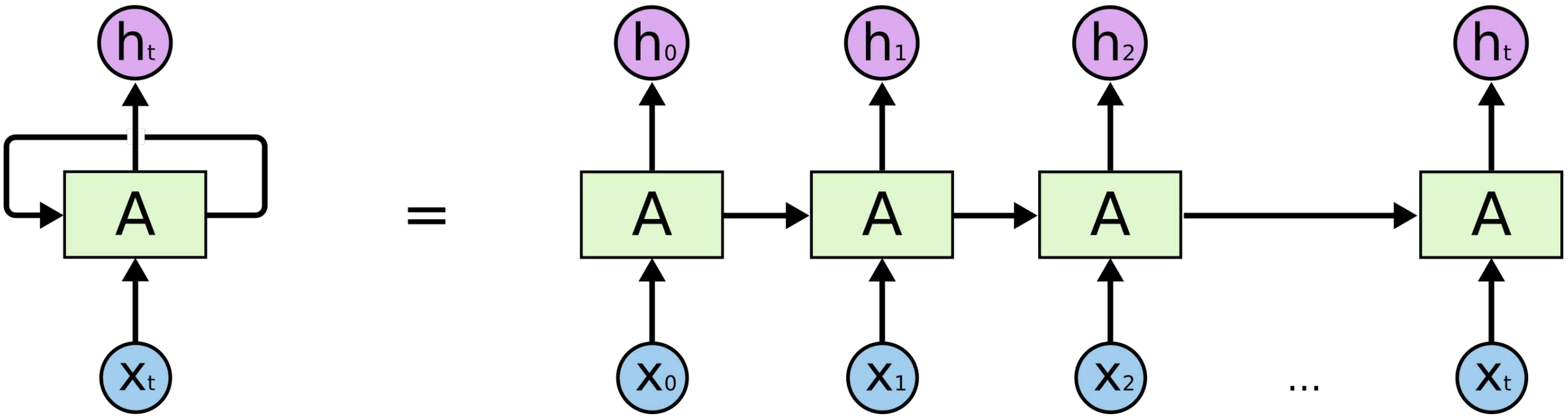
Modeling of sequence data



Recurrent Neural Networks have loops.

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

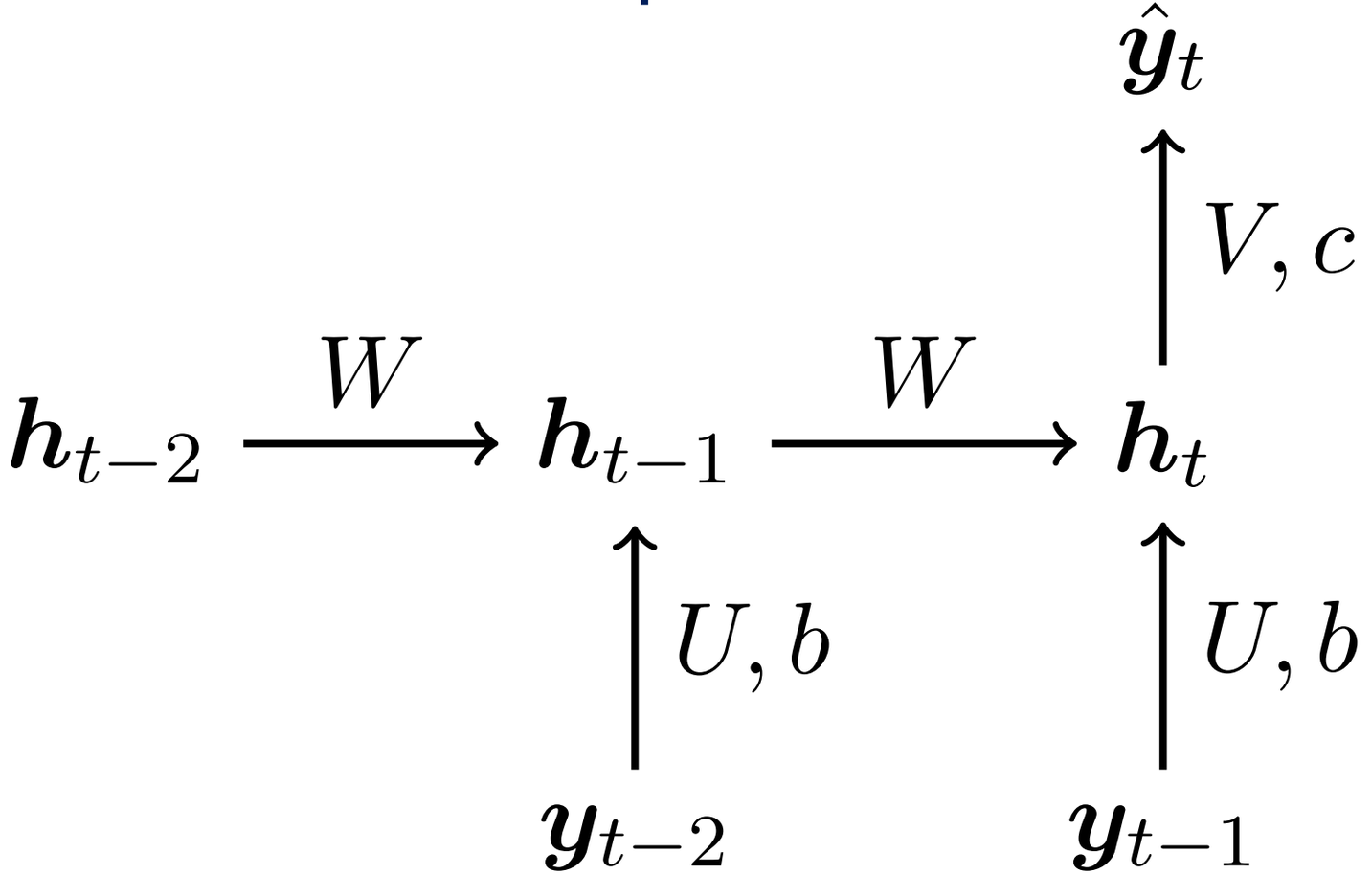
Modeling of sequence data



An unrolled recurrent neural network.

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

A simple RNN



$$\hat{\boldsymbol{y}}_t = f(\boldsymbol{y}_{t-1}, \boldsymbol{y}_{t-2})$$

$$\hat{\boldsymbol{y}}_t = \boldsymbol{h}_t V + c,\ \boldsymbol{h}_t = \tanh\left(\boldsymbol{h}_{t-1} W + \boldsymbol{y}_{t-1} U + b\right)$$

$$\boldsymbol{h}_{t-1} = \tanh\left(\boldsymbol{h}_{t-2} W + \boldsymbol{y}_{t-2} U + b\right)$$

$$\boldsymbol{h}_{t-2} = \mathbf{0}$$

A simple RNN

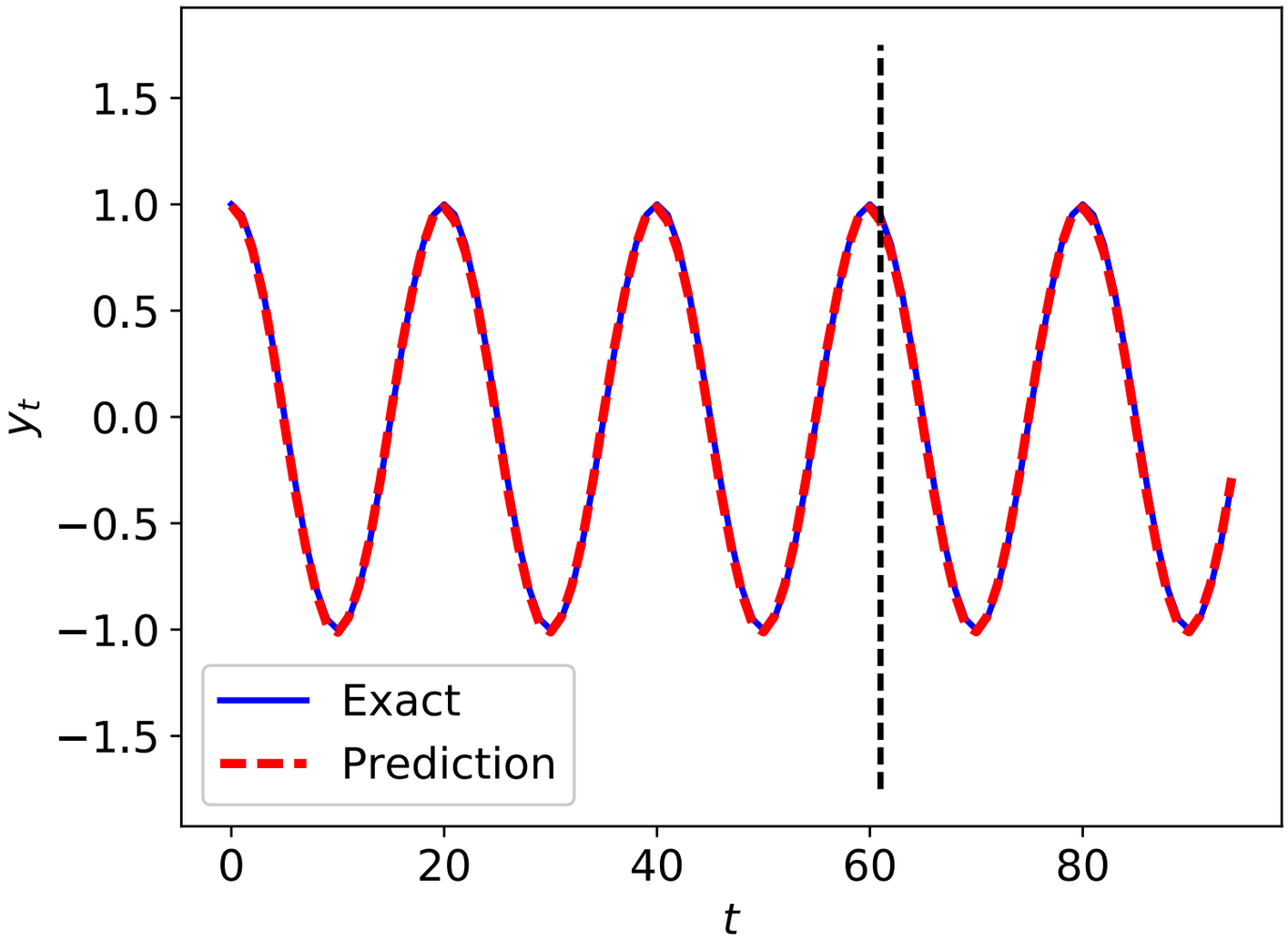
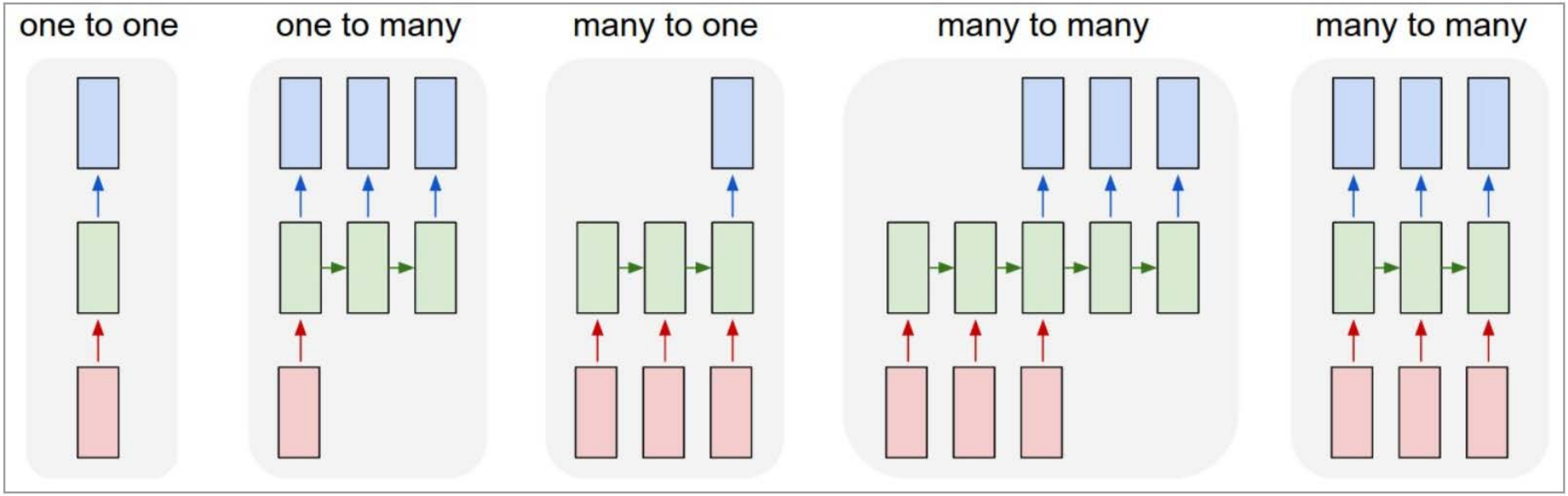


FIG. 3.1. *Recurrent neural network predicting the dynamics of a simple sine wave.*

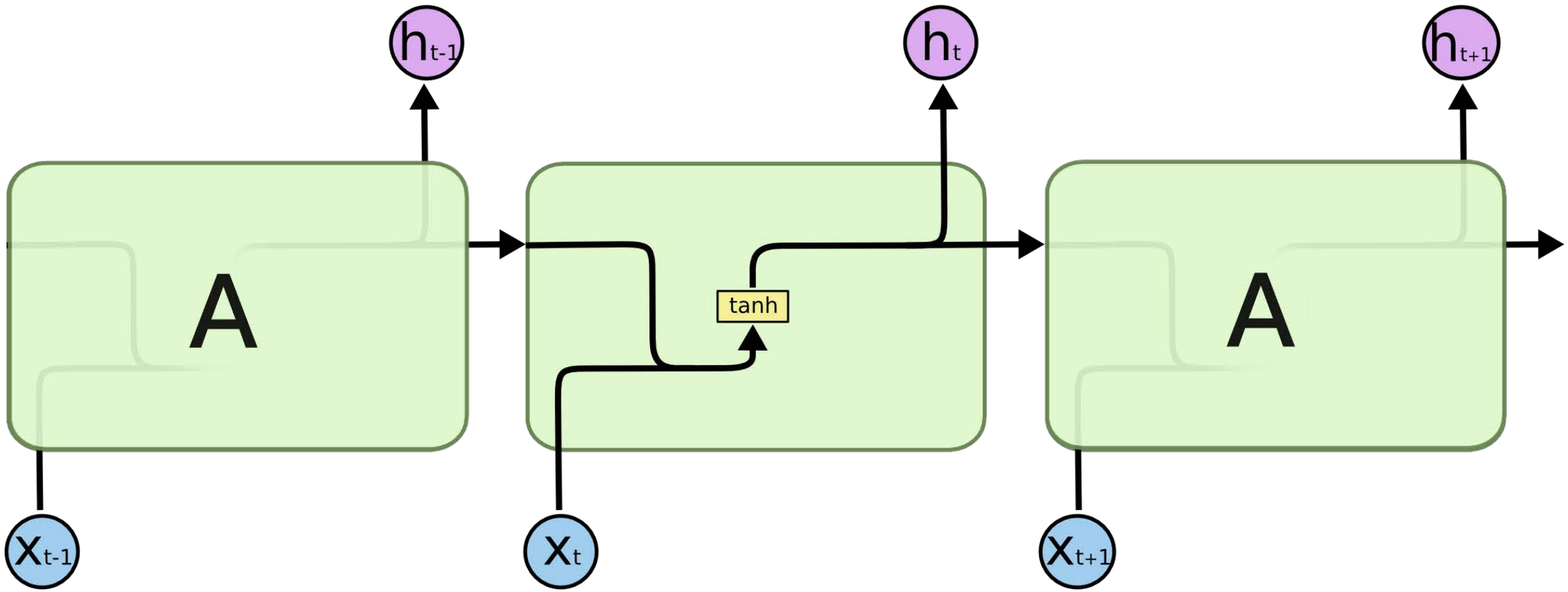
Modeling of sequence data



Each rectangle is a vector and arrows represent functions (e.g. matrix multiply). Input vectors are in red, output vectors are in blue and green vectors hold the RNN's state (more on this soon). From left to right: **(1)** Vanilla mode of processing without RNN, from fixed-sized input to fixed-sized output (e.g. image classification). **(2)** Sequence output (e.g. image captioning takes an image and outputs a sentence of words). **(3)** Sequence input (e.g. sentiment analysis where a given sentence is classified as expressing positive or negative sentiment). **(4)** Sequence input and sequence output (e.g. Machine Translation: an RNN reads a sentence in English and then outputs a sentence in French). **(5)** Synced sequence input and output (e.g. video classification where we wish to label each frame of the video). Notice that in every case are no pre-specified constraints on the lengths sequences because the recurrent transformation (green) is fixed and can be applied as many times as we like.

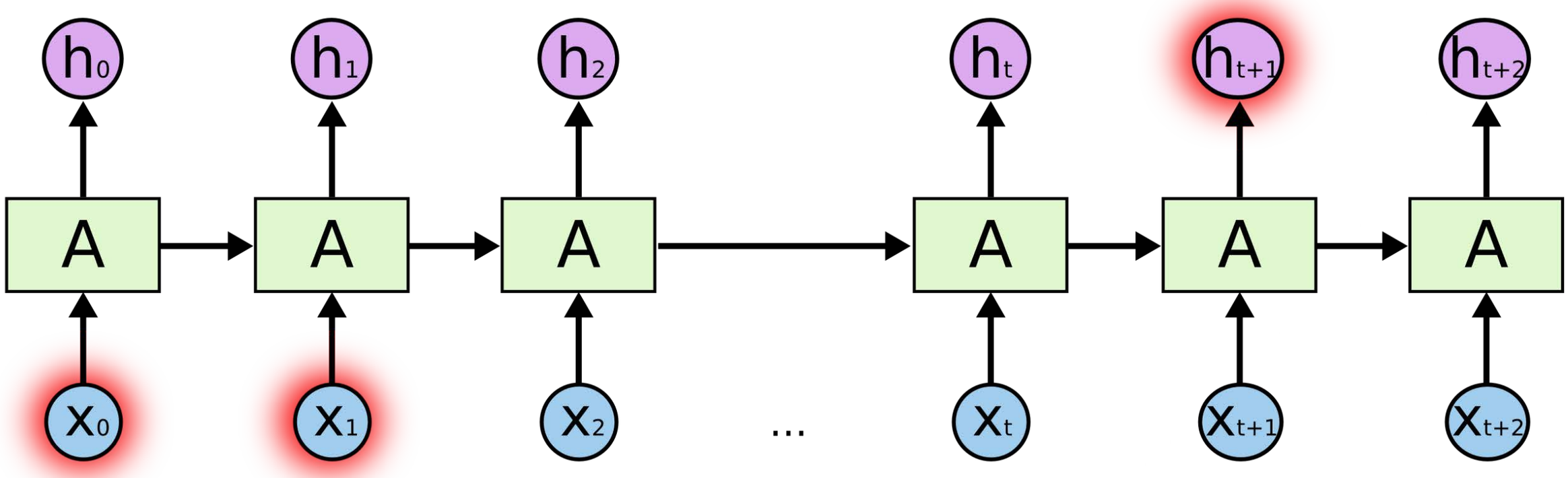
http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Modeling of sequence data

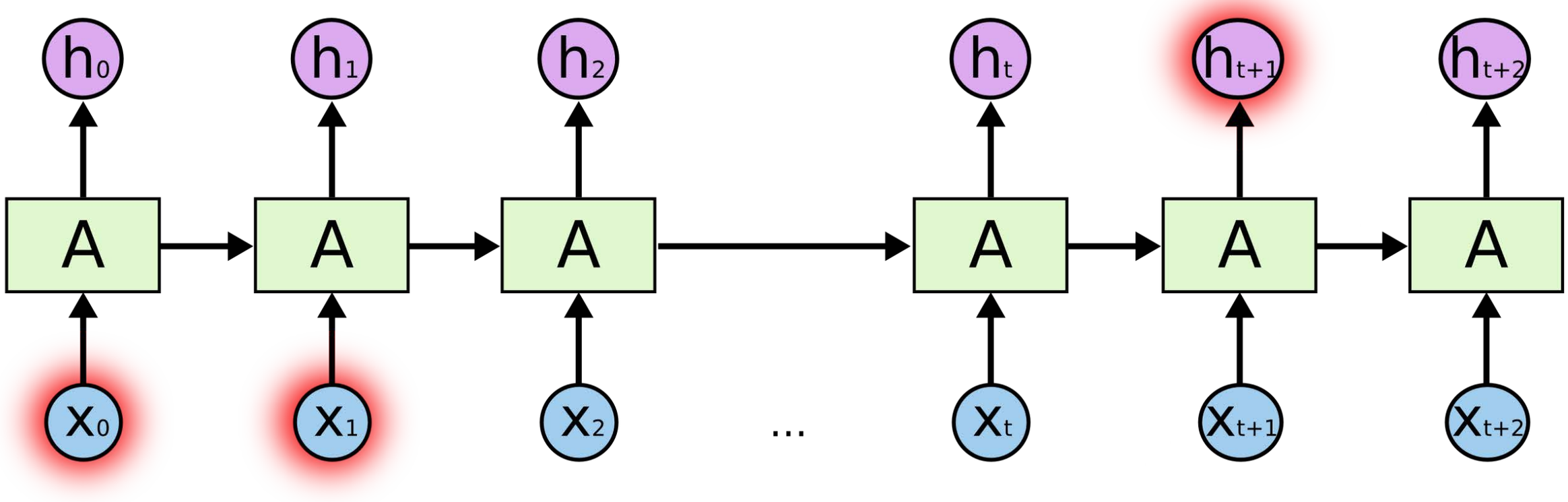


The repeating module in a standard RNN contains a single layer.

Modeling long dependencies



The Problem of Long-Term Dependencies



Vanishing and/or exploding gradients

Why gradients explode or vanish?

$$\mathbf{h}^{(t)} = f(\mathbf{x}^{(t)}, \mathbf{h}^{(t-1)})$$

$$\mathbf{h}^{(4)} = f(f(f(\mathbf{h}^{(1)}, \mathbf{x}^{(2)}), \mathbf{x}^{(3)}), \mathbf{x}^{(4)})$$

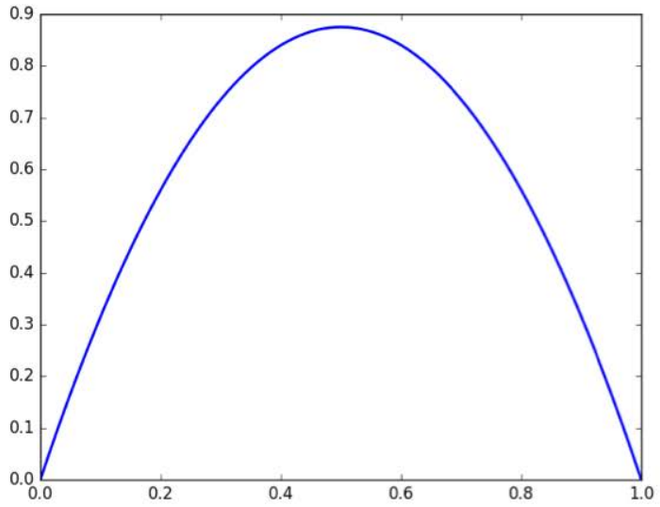
This looks a bit like repeatedly applying the function f. Therefore, we can gain some intuition for how RNNs behave by studying **iterated functions**, i.e. functions which we iterate many times.

Pascanu, R., Mikolov, T., & Bengio, Y. (2013, February). On the difficulty of training recurrent neural networks. In International Conference on Machine Learning (pp. 1310-1318).

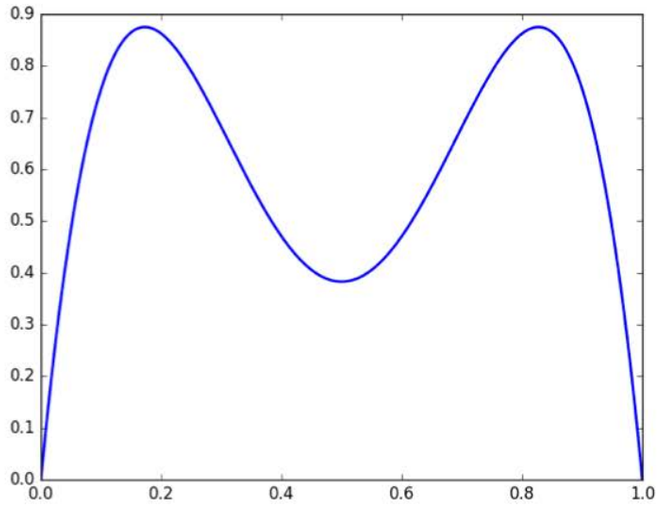
http://www.cs.toronto.edu/~rgrosse/courses/csc321_2017/readings/L15%20Exploding%20and%20Vanishing%20Gradients.pdf

Why gradients explode or vanish?

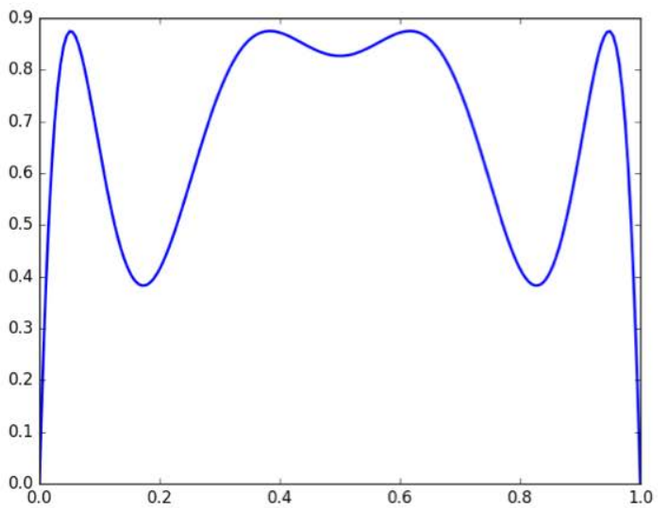
Let's consider an example: $f(x) = 3.5\,x\,(1-x)$



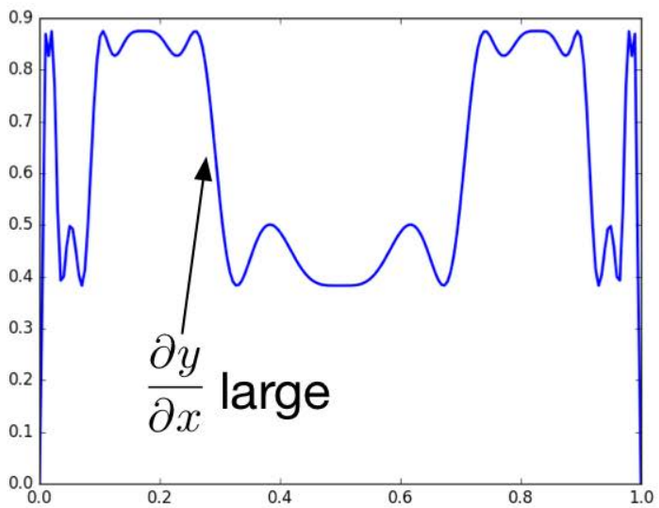
$y = f(x)$



$y = f(f(x))$



$y = f(f(f(x)))$



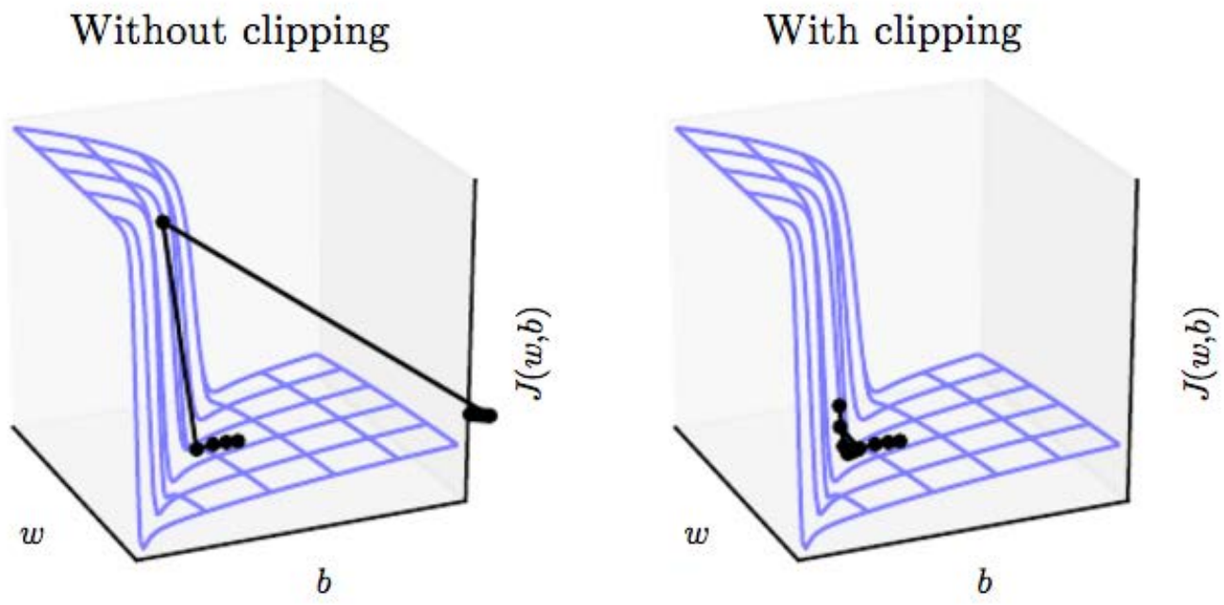
$y = \underbrace{f \circ \cdots \circ f}_{6 \text{ times}}(x)$

http://www.cs.toronto.edu/~rgrosse/courses/csc321_2017/readings/L15%20Exploding%20and%20Vanishing%20Gradients.pdf

Possible remedies

$$\textbf{Gradient Clipping} \quad \mathbf{g} \leftarrow \frac{\eta \mathbf{g}}{\|\mathbf{g}\|}$$

This biases the training procedure, since the resulting values won't actually be the gradient of the cost function. However, this bias can be worth it if it keeps things stable. The following figure shows an example with a cliff and a narrow valley; if you happen to land on the face of the cliff, you take a huge step which propels you outside the good region. With gradient clipping, you can stay within the valley.



— Goodfellow et al., *Deep Learning*

http://www.cs.toronto.edu/~rgrosse/courses/csc321_2017/readings/L15%20Exploding%20and%20Vanishing%20Gradients.pdf

Possible remedies

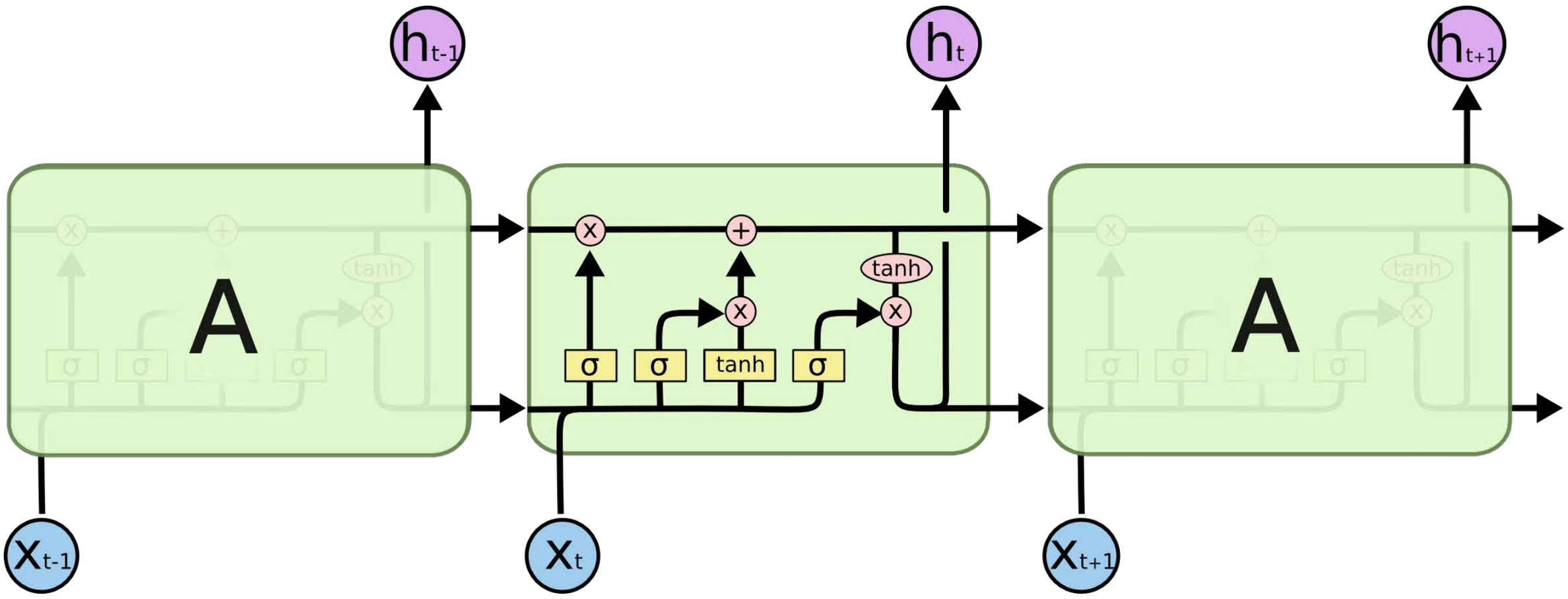
Identity Initialization

In general, iterated functions can have complex and chaotic behavior. But there's one particular function you can iterate as many times as you like: the identity function $f(x) = x$. If your network computes the identity function, the gradient computation will be perfectly stable, since the Jacobian is simply the identity matrix. Of course, the identity function isn't a very interesting thing to compute, but it still suggests we can keep things stable by encouraging the computations to stay close to the identity function.

The **identity RNN** architecture[2] is a kind of RNN where the activation functions are all ReLU, and the recurrent weights are initialized to the identity matrix. The ReLU activation function clips the activations to be nonnegative, but for nonnegative activations, it's equivalent to the identity function. This simple initialization trick achieved some neat results; for instance, it was able to classify MNIST digits which were fed to the network one pixel at a time, as a length-784 sequence.

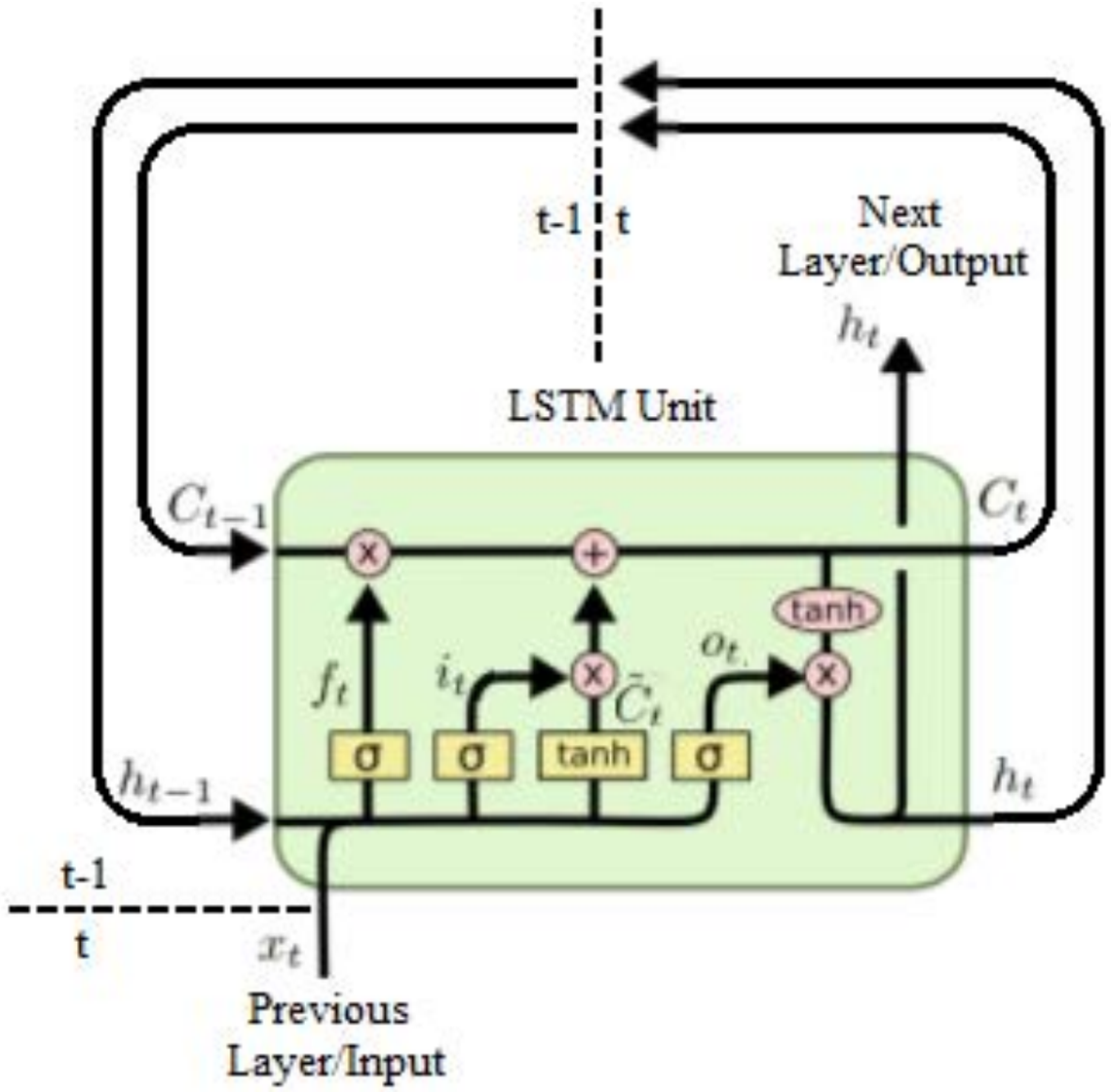
http://www.cs.toronto.edu/~rgrosse/courses/csc321_2017/readings/L15%20Exploding%20and%20Vanishing%20Gradients.pdf

Long Short-Term Memory networks

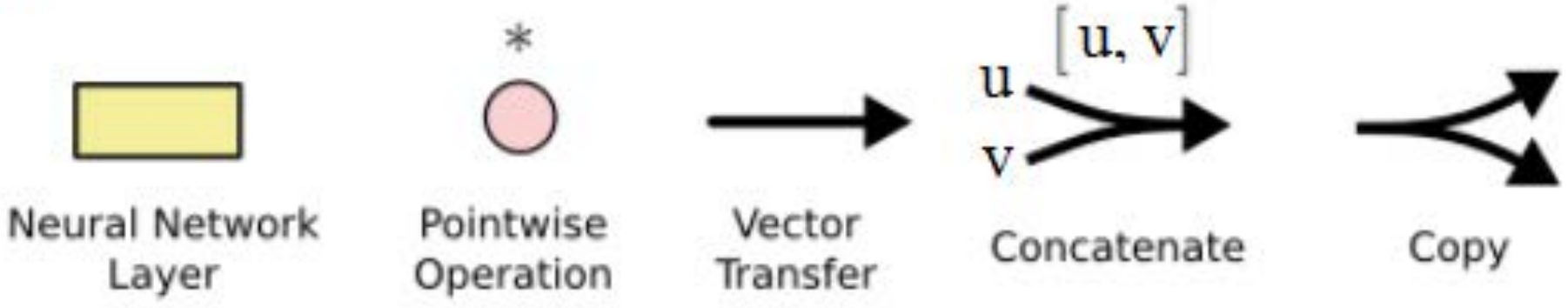


The repeating module in an LSTM contains four interacting layers.

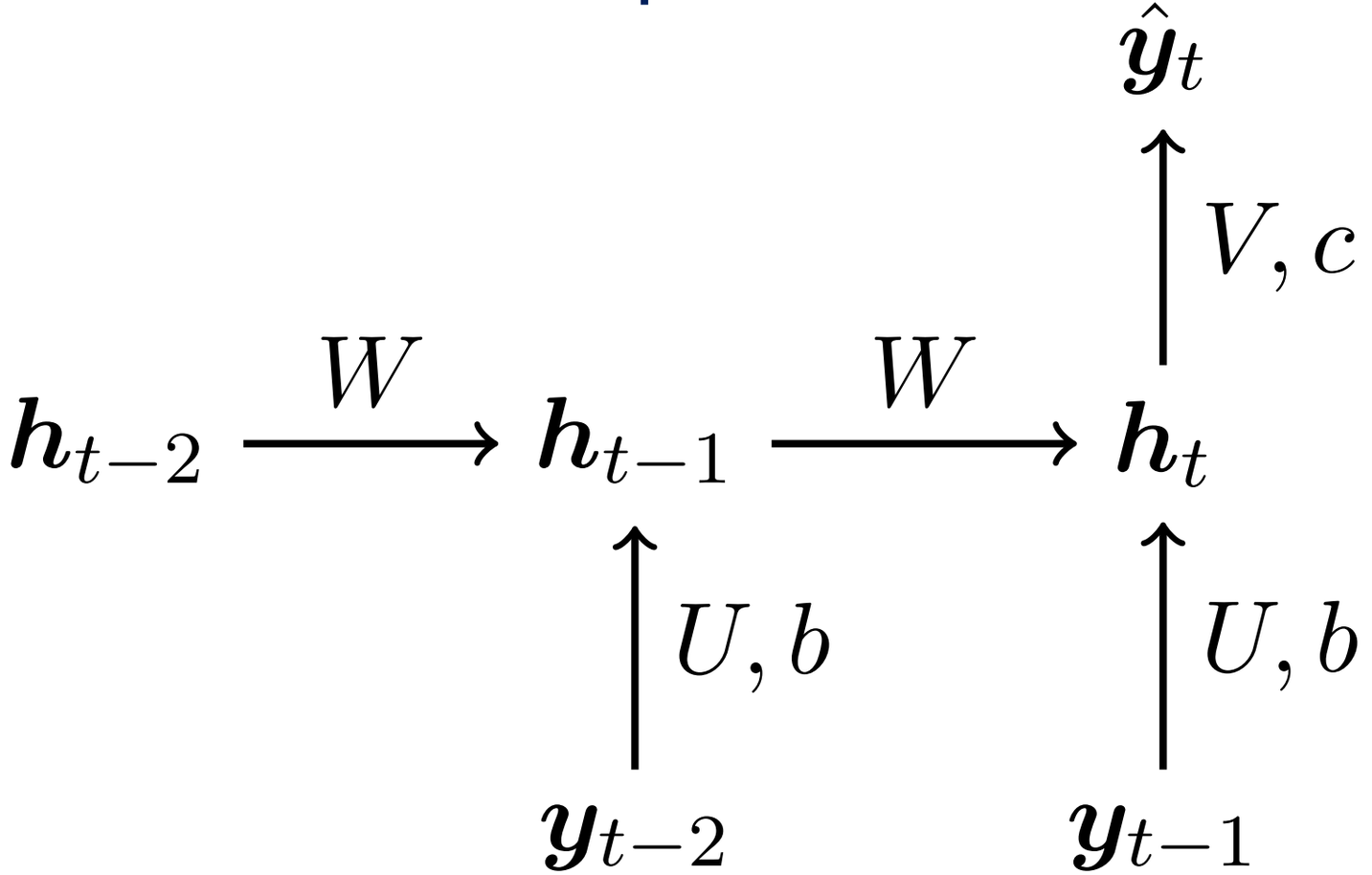
Long Short-Term Memory networks



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$
$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$



A simple RNN



$$\hat{\boldsymbol{y}}_t = f(\boldsymbol{y}_{t-1}, \boldsymbol{y}_{t-2})$$

$$\hat{\boldsymbol{y}}_t = \boldsymbol{h}_t V + c,\ \boldsymbol{h}_t = \tanh\left(\boldsymbol{h}_{t-1} W + \boldsymbol{y}_{t-1} U + b\right)$$

$$\boldsymbol{h}_{t-1} = \tanh\left(\boldsymbol{h}_{t-2} W + \boldsymbol{y}_{t-2} U + b\right)$$

$$\boldsymbol{h}_{t-2} = \mathbf{0}$$