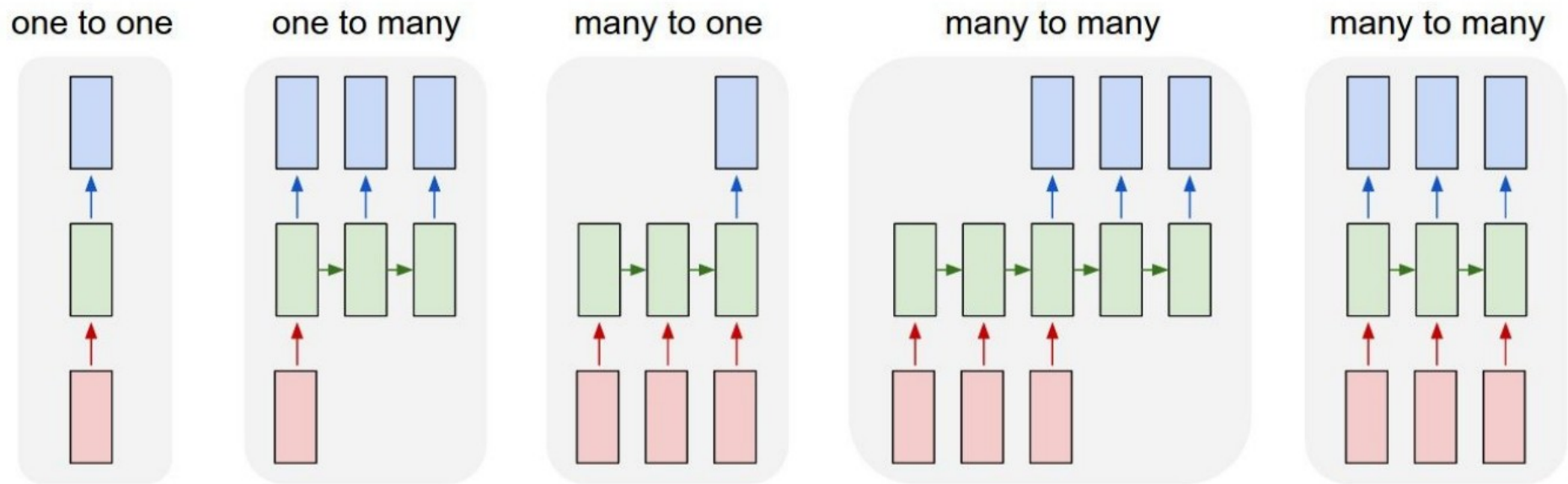


Machine Learning

- Intro to Recurrent Neural Networks -

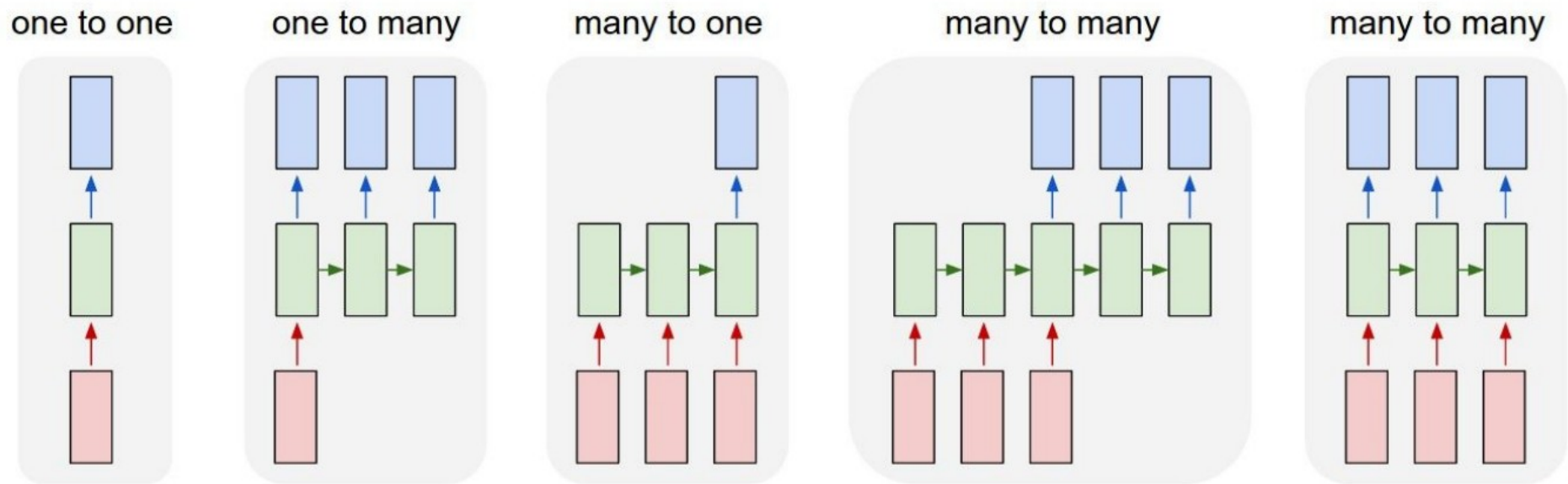
RNN Tasks

Recurrent Neural Networks: Process Sequences



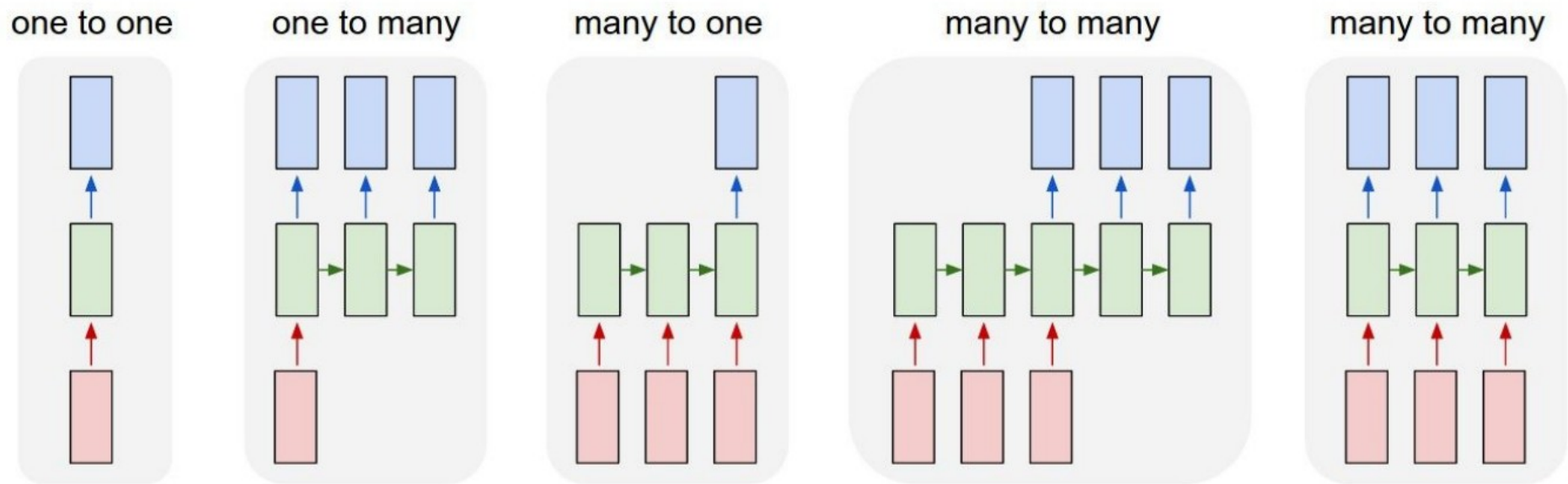
Vanilla RNNs

Recurrent Neural Networks: Process Sequences



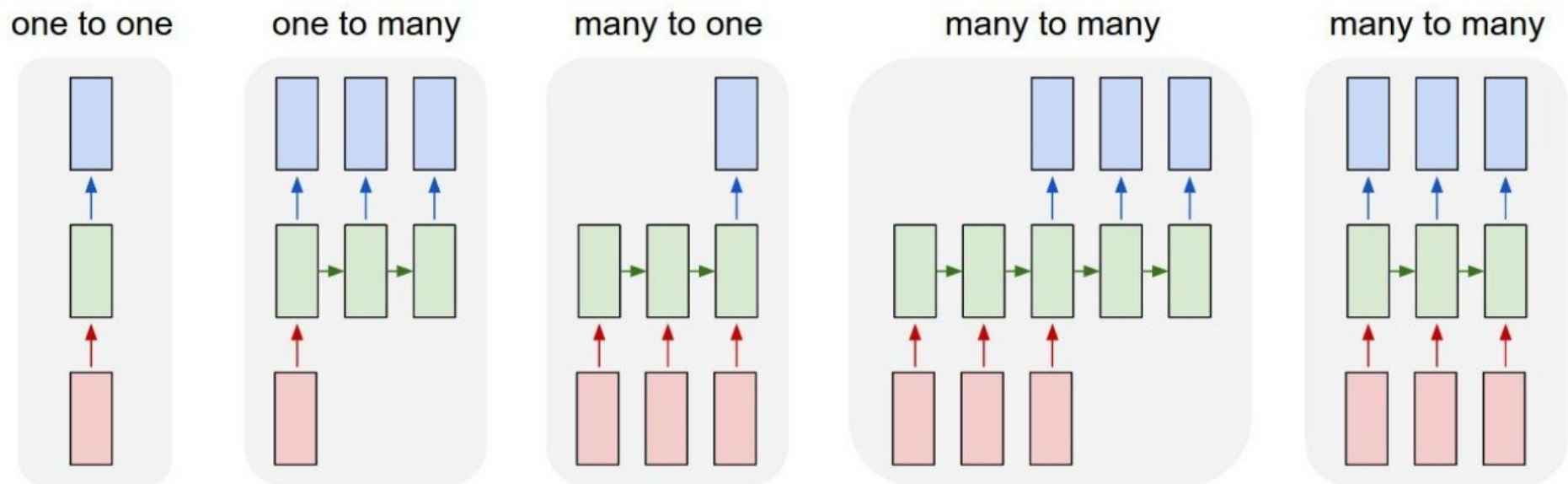
e.g. Image Captioning
Image → sequence of words

Recurrent Neural Networks: Process Sequences



e.g. Sentiment Classification
Sequence of words → sentiment

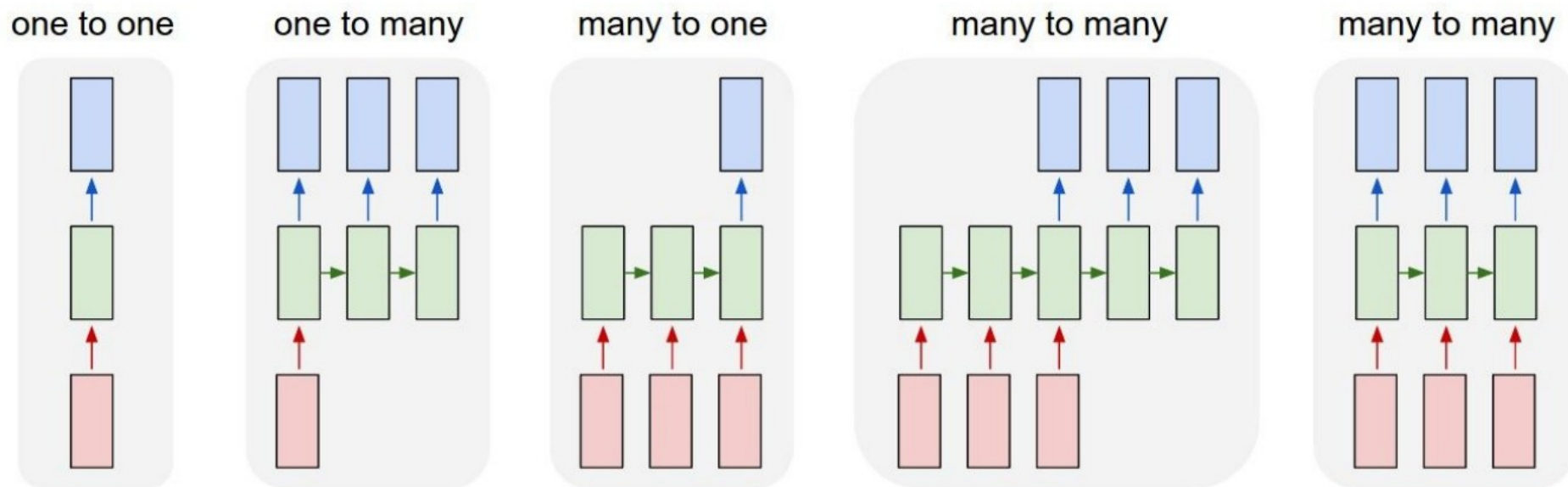
Recurrent Neural Networks: Process Sequences



e.g. Translation

Sequence of words → sequence of words

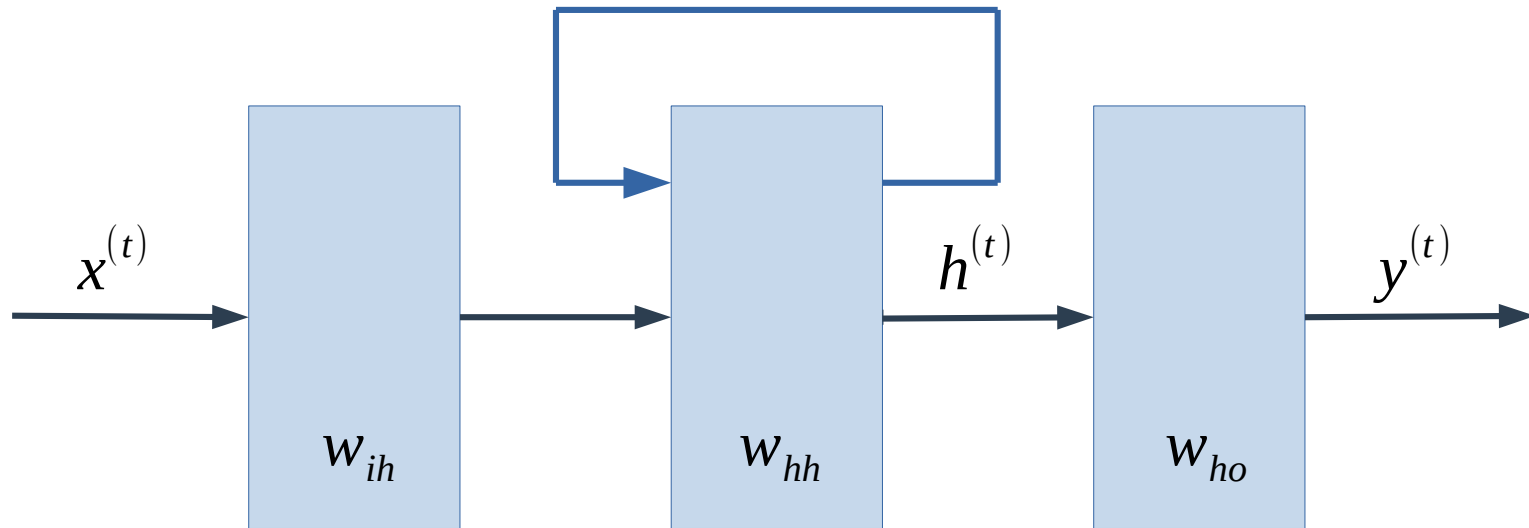
Recurrent Neural Networks: Process Sequences



e.g. Video classification
on frame level

RNN Model

Vanilla RNN Model

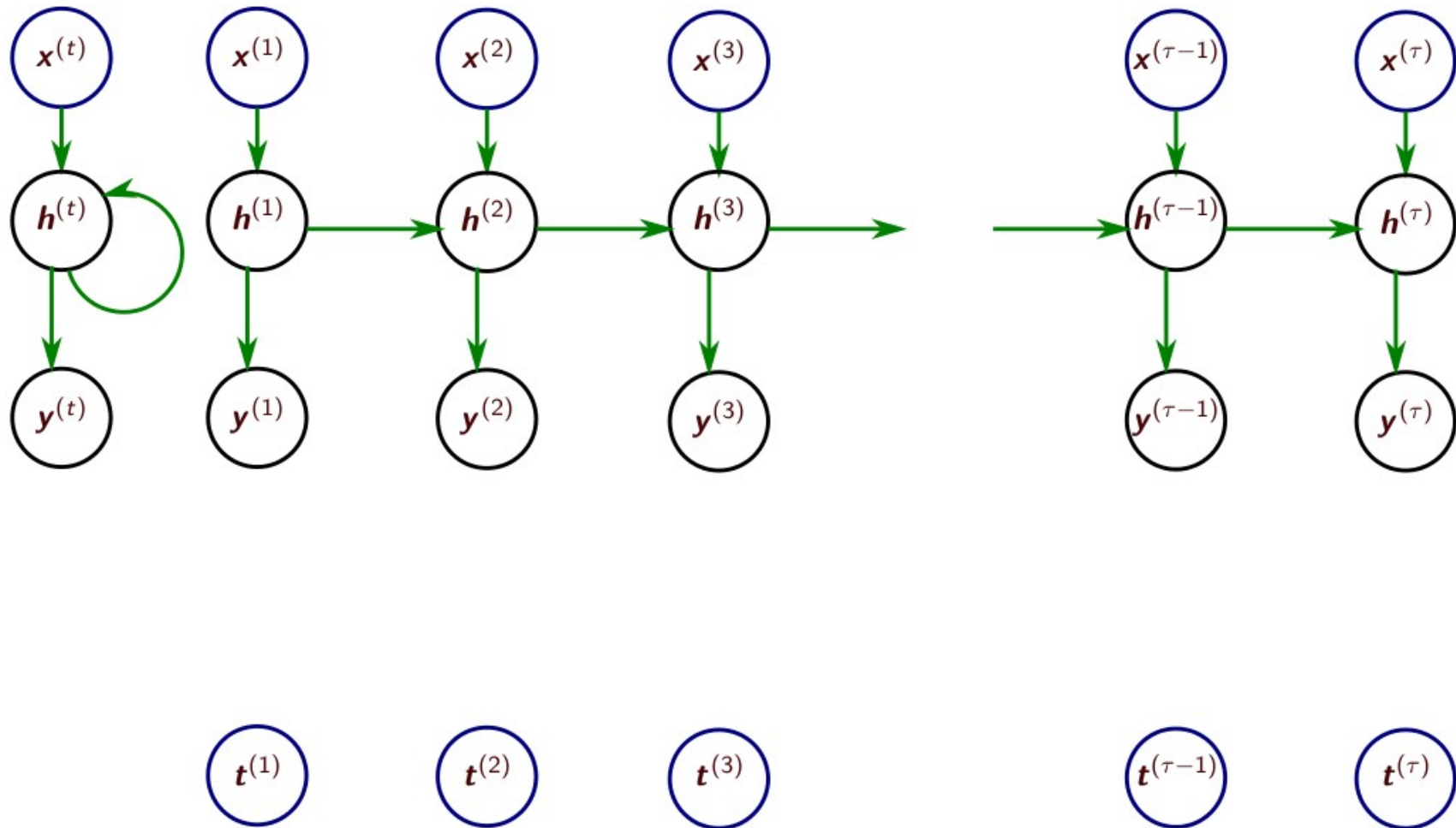


- Current state depends on *current inputs* and *previous state*
- RNNs can yield outputs at each time step

$$h^{(t)} = f_{w_{hh}}(h^{(t-1)}, f_{w_{ih}}(x^{(t)}))$$

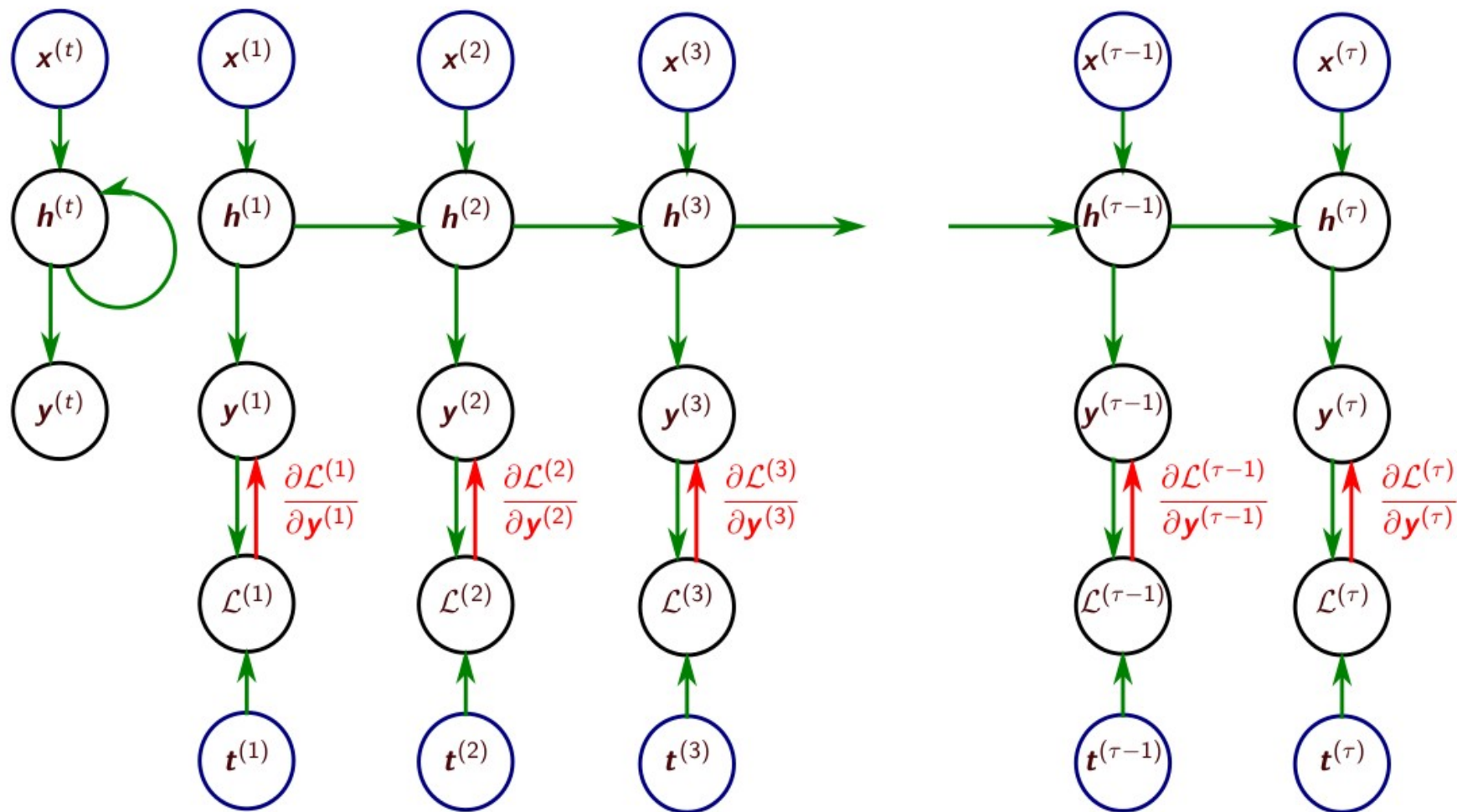
$$y^{(t)} = f_{w_{ho}}(h^{(t)}), \forall t \in \{1 \dots \tau\}$$

Unfolding RNN in time



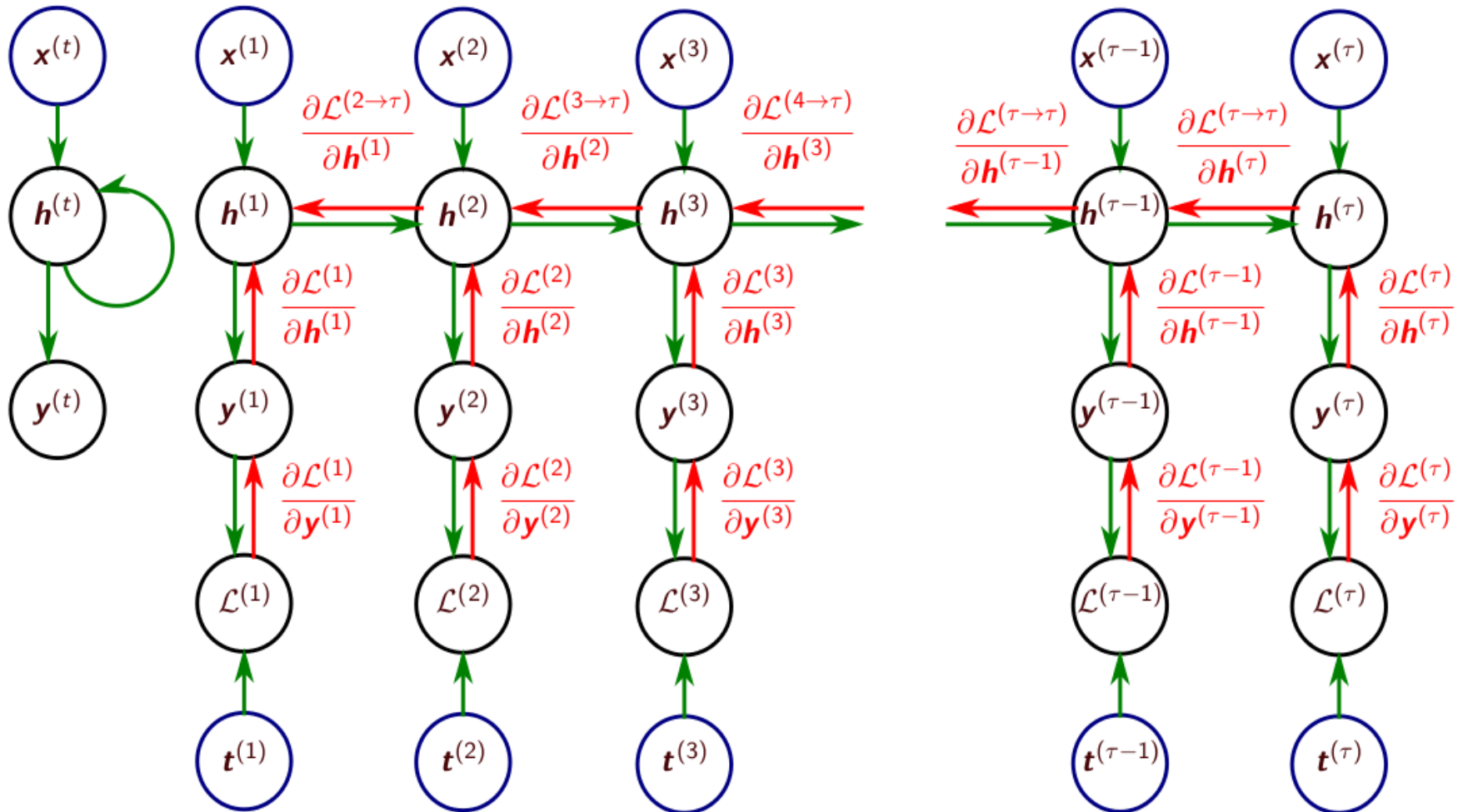
Source: NN Lectures, Tudor
Berariu, 2016

Unfolding RNN in time



Source: NN Lectures, Tudor Berariu, 2016

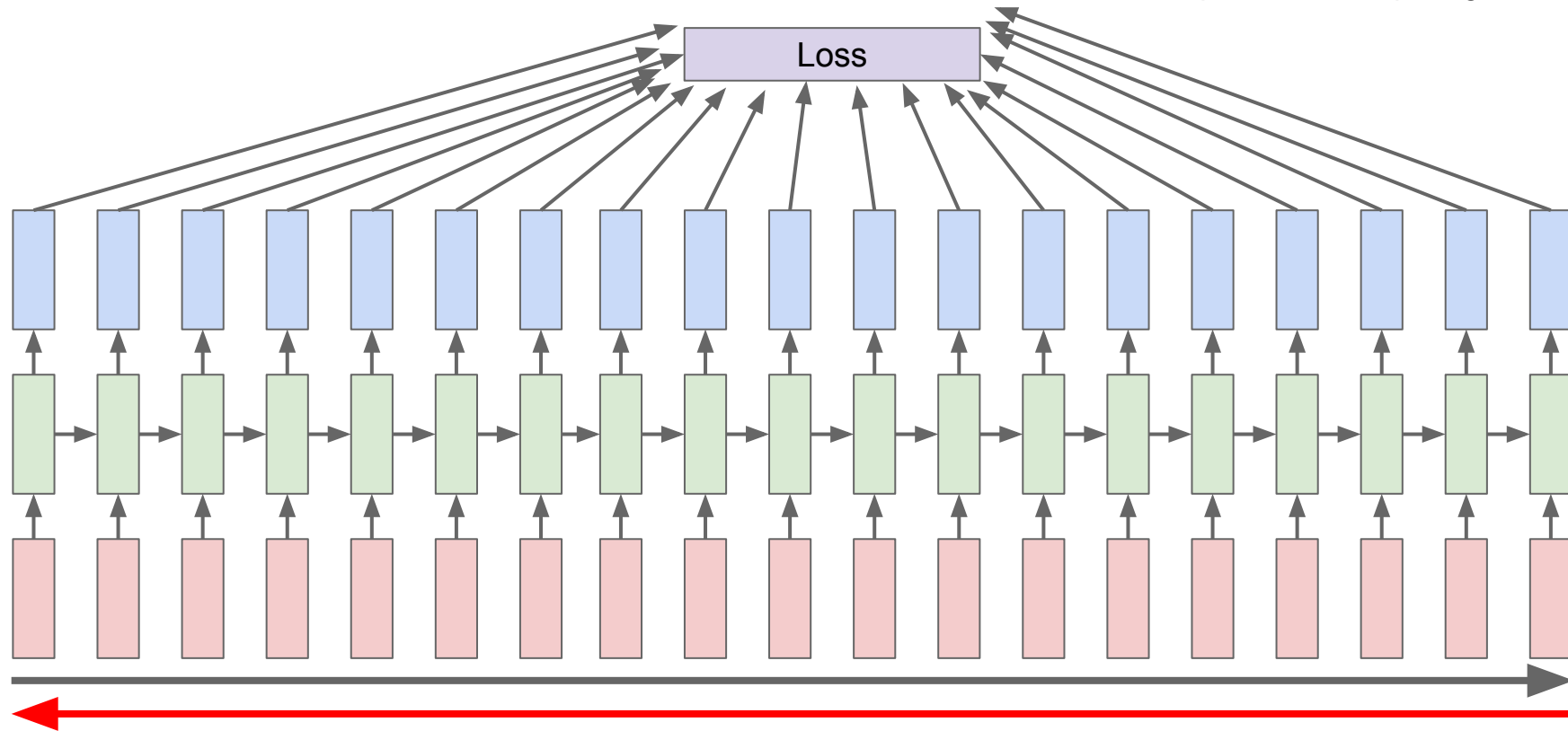
Unfolding RNN in time



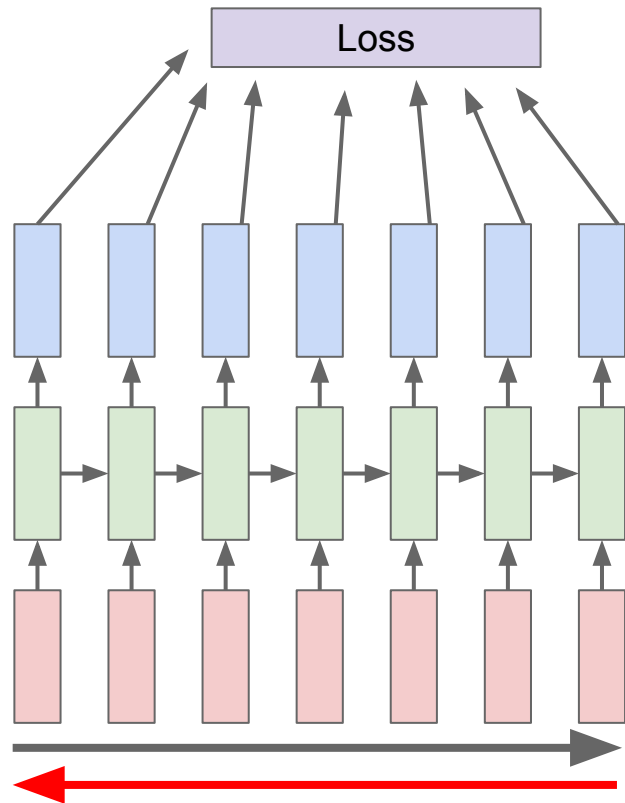
Source: NN Lectures, Tudor Berariu, 2016

Backpropagation through time

Forward through entire sequence to compute loss, then backward through entire sequence to compute gradient

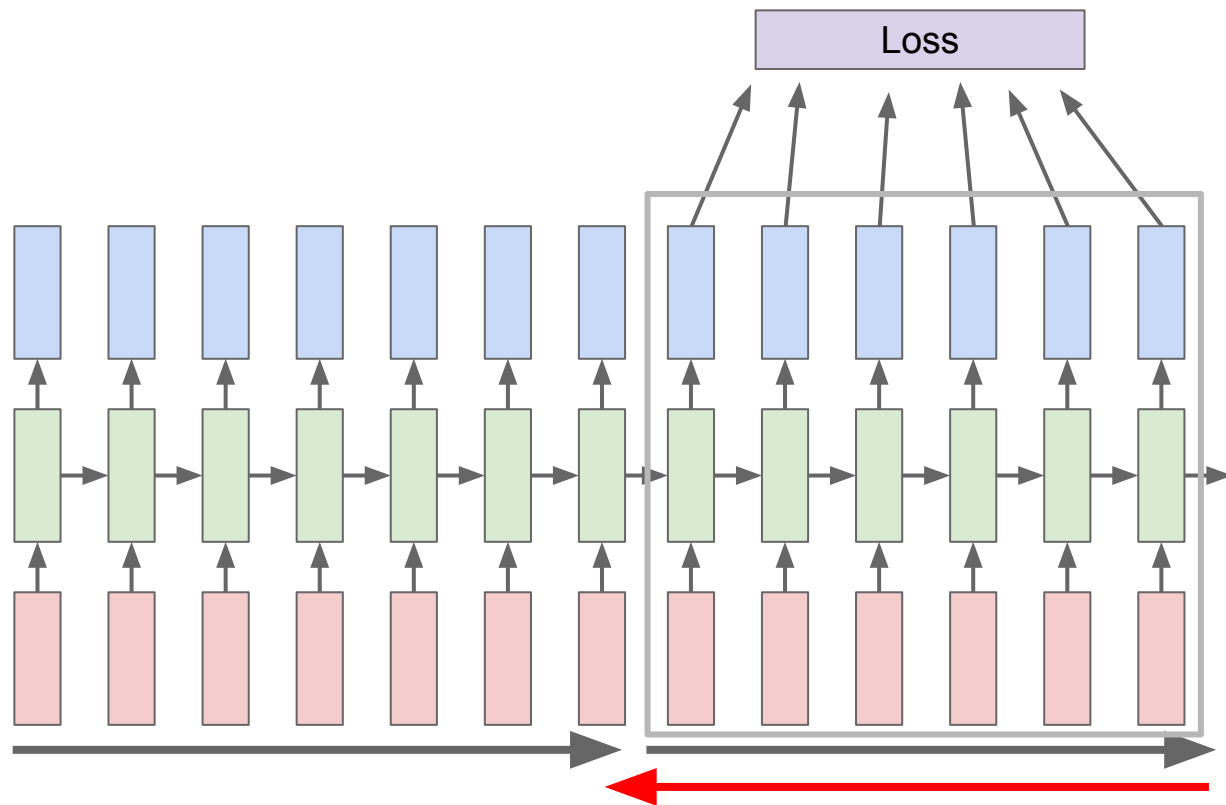


Truncated Backpropagation through time



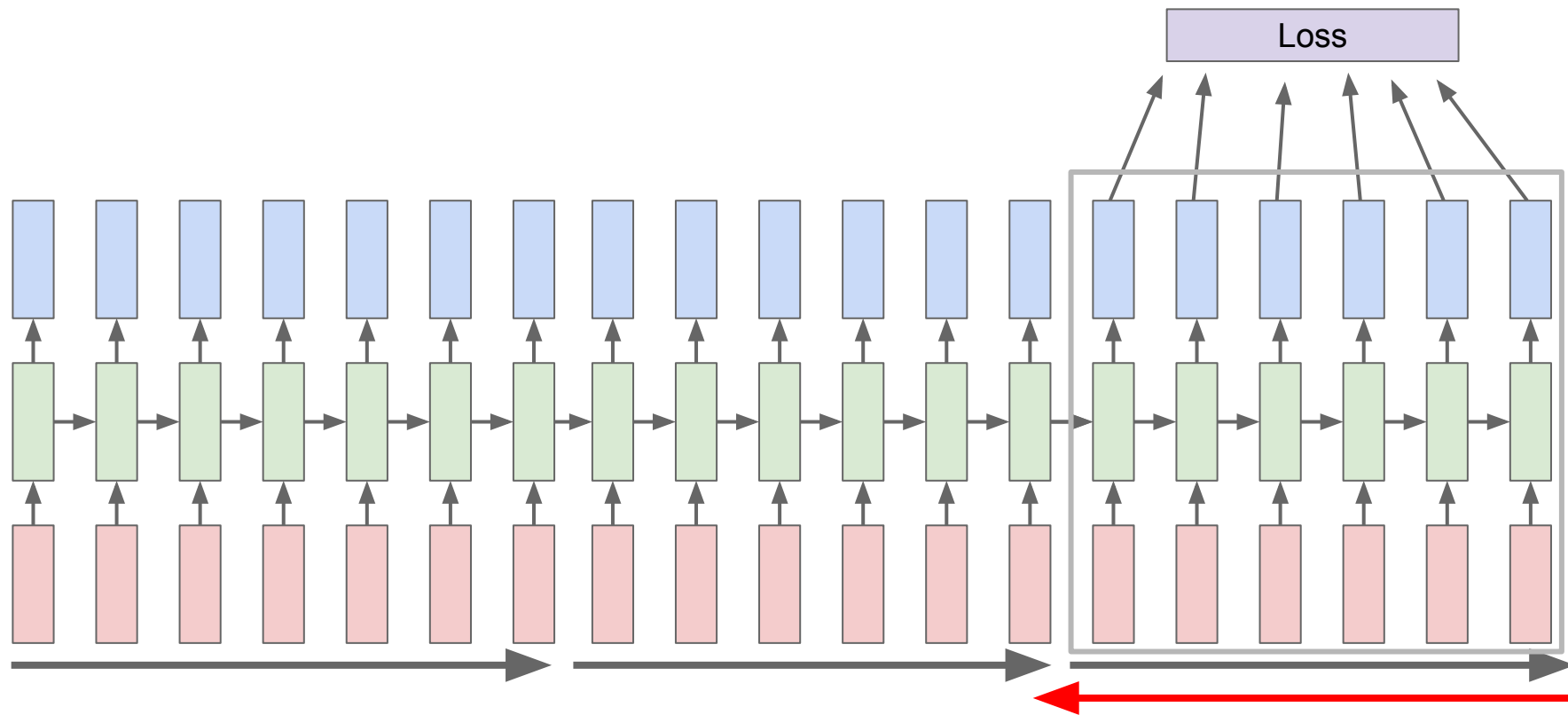
Run forward and backward through chunks of the sequence instead of whole sequence

Truncated Backpropagation through time



Carry hidden states forward in time forever, but only backpropagate for some smaller number of steps

Truncated Backpropagation through time



Truncated BPTT

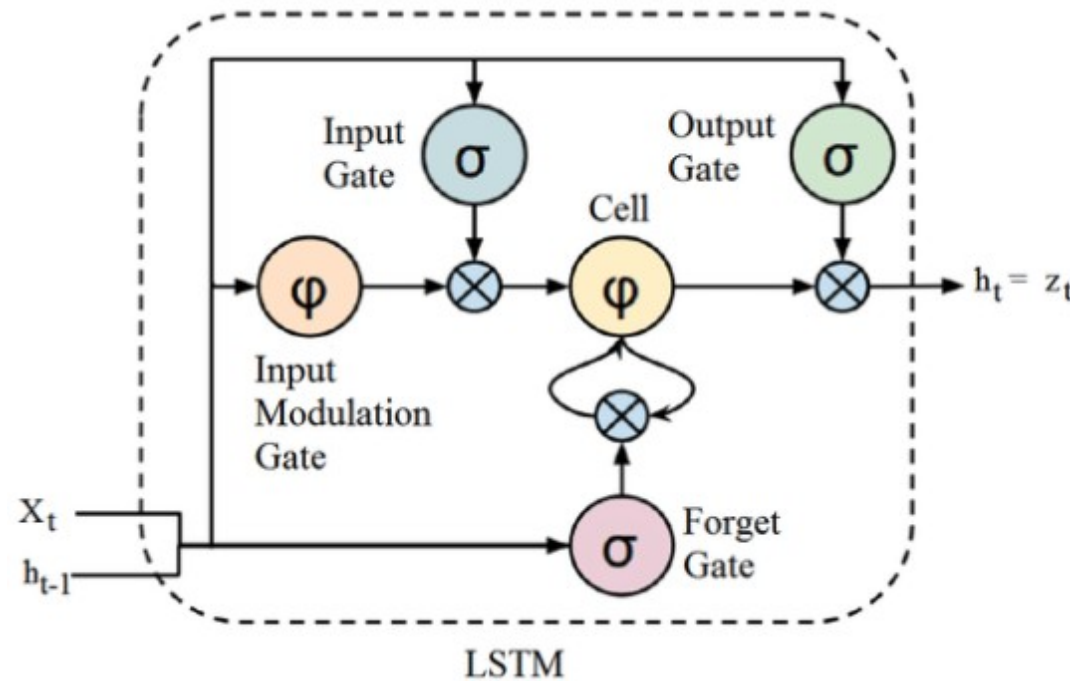
- Used in practice
- Summary of the algorithm:
 - Present a sequence of k_1 timesteps of input and output pairs to the network.
 - Unroll the network then calculate and accumulate errors across k_2 timesteps.
 - Roll-up the network and update weights.
 - Repeat

Teacher Forcing and Warm-start

- When training a RNN to generate a sequence, often, the predictions (outputs $y^{(t)}$) of a RNN cell are used as the input of the cell at the next timestamp
- **Teacher Forcing**: at training time, use the ***targets*** of the sequence, instead of RNN predictions, as inputs to the next step
- **Warm-start**: when using an RNN to predict a next value conditioned on *previous* predictions, it is sometimes necessary to give the RNN some *context* (known ground truth elements) before letting it predict on its own

LSTM

LSTM Cell



Img source:
<https://medium.com/@kangeengine/>

- Input Gate (***i*** in $(0, 1)$ – sigmoid) – scales input to cell (write)
- Output Gate (***o*** in $(0, 1)$ – sigmoid) – scales output from cell (read)
- Forget Gate (***f*** in $(0, 1)$ – sigmoid) – scales old cell values (reset mem)

LSTM Cell - Equations

$$i_t = \sigma(\theta_{xi} x^{(t)} + \theta_{hi} h^{(t-1)} + b_i)$$

$$f_t = \sigma(\theta_{xf} x^{(t)} + \theta_{hf} h^{(t-1)} + b_f)$$

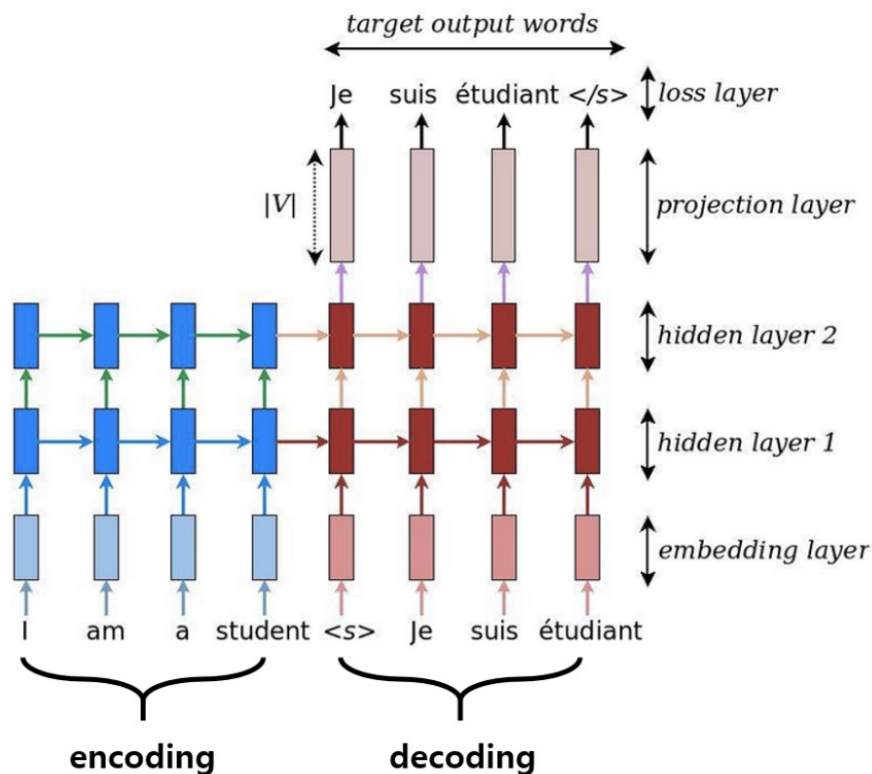
$$o_t = \sigma(\theta_{xo} x^{(t)} + \theta_{ho} h^{(t-1)} + b_o)$$

$$g_t = \tanh(\theta_{xg} x^{(t)} + \theta_{hg} h^{(t-1)} + b_g)$$

$$c_t = f_t \odot c_{(t-1)} + i_t \odot g_t$$

$$h_t = o_t \odot \tanh(c_t), \text{ where } \odot \text{ is elementwise multiplication}$$

LSTMs in practice



- Sutskever et al, Sequence to Sequence Learning with Neural Networks, NIPS 2014
 - Models are huge :-)
 - 4 layers, 1000 LSTM cells per layer
 - Input vocabulary of 160k
 - Output vocabulary of 80k
 - 1000 dimensional word embeddings