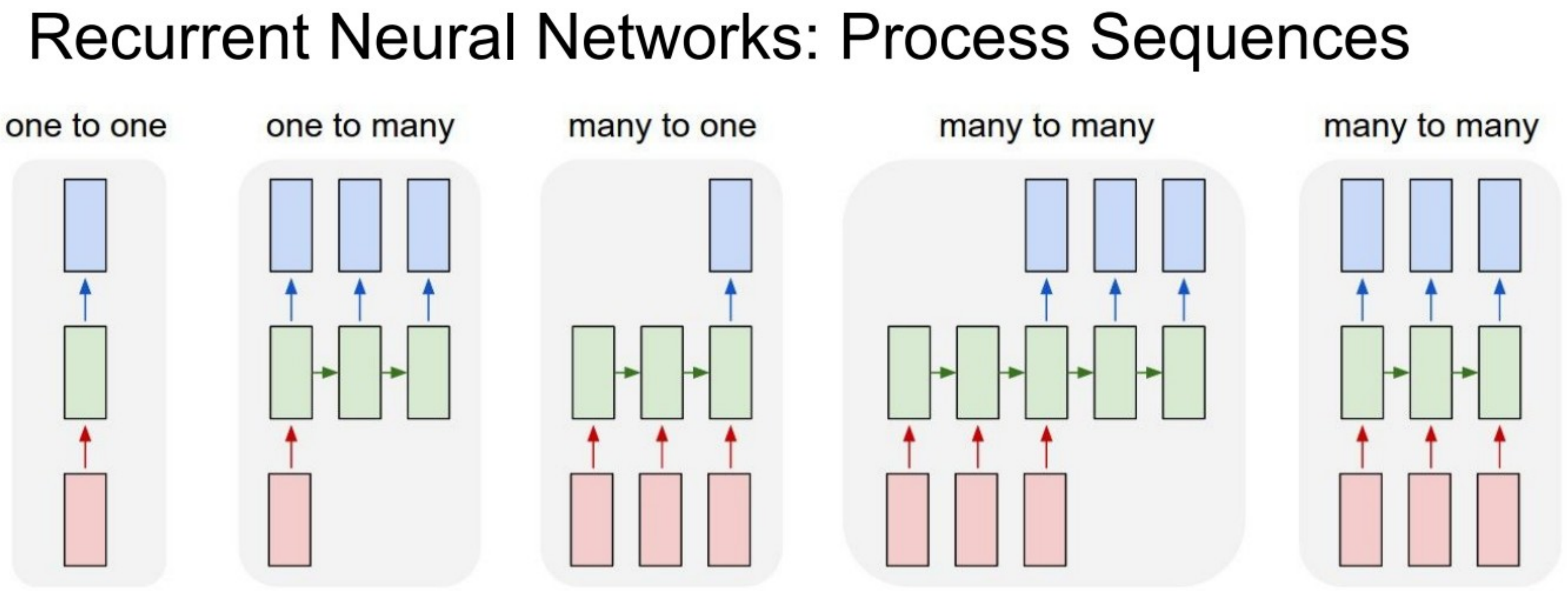
Machine Learning

- Intro to Recurrent Neural Networks -

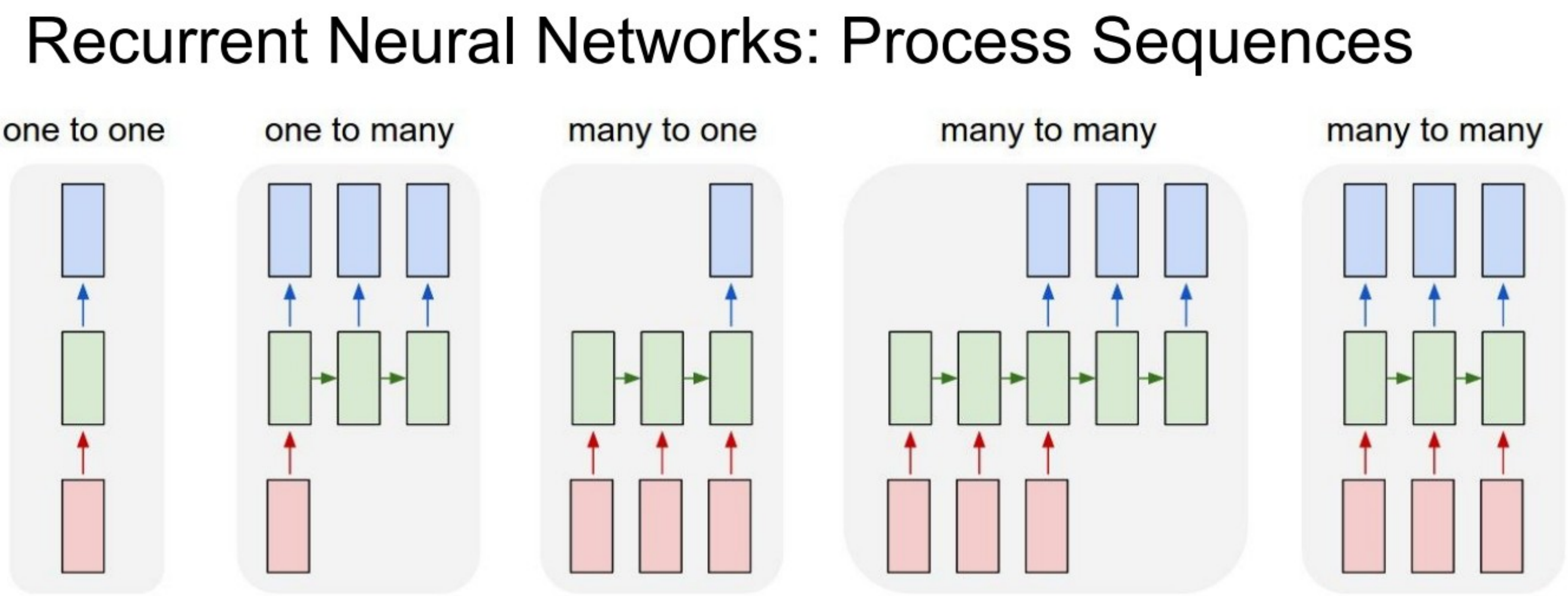
RNN Tasks

RNN Tasks



Vanilla RNNs

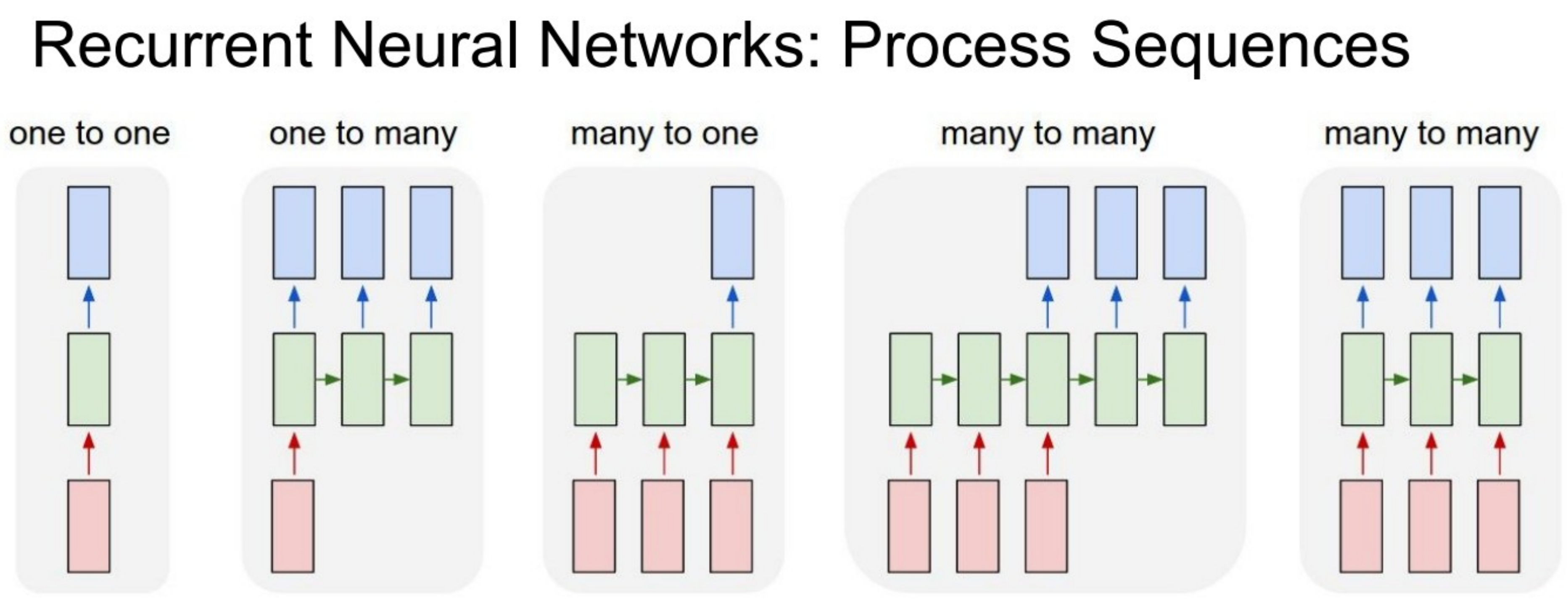
Source: CS231n Lecture 10



e.g. Image Captioning
Image → sequence of words

Source: CS231n Lecture 10

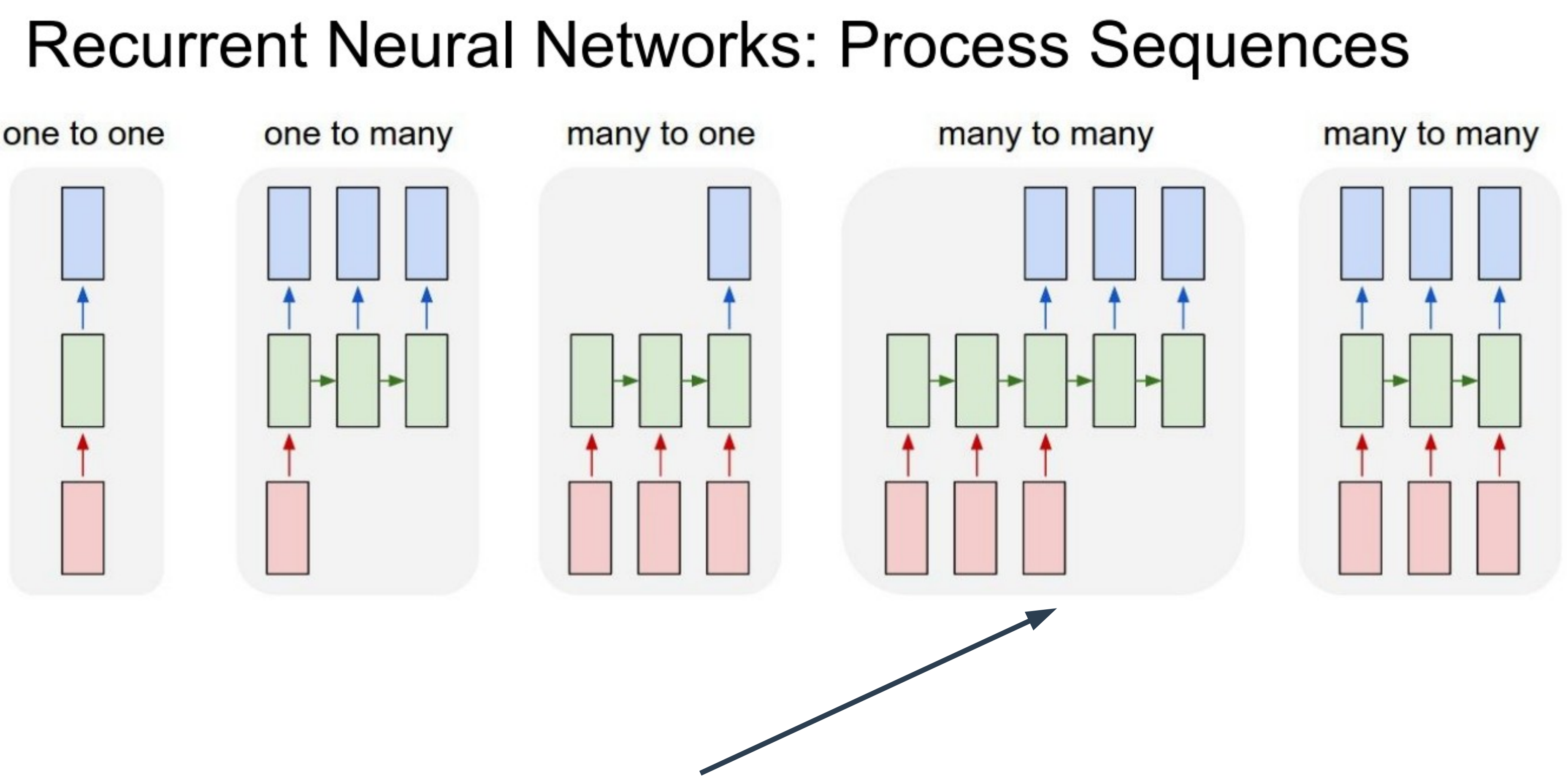
RNN Tasks



e.g. Sentiment Classification
Sequence of words → sentiment

Source: CS231n Lecture 10

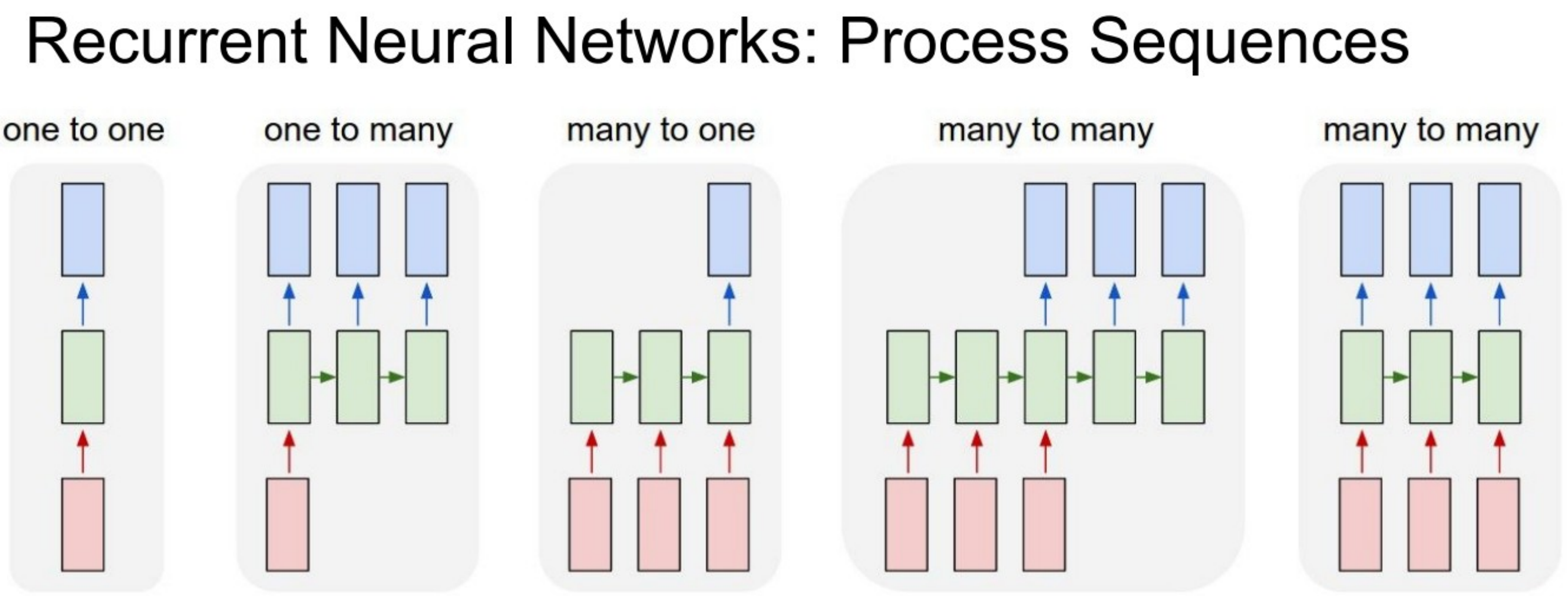
RNN Tasks



e.g. Translation
Sequence of words → sequence of words

Source: CS231n Lecture 10

RNN Tasks

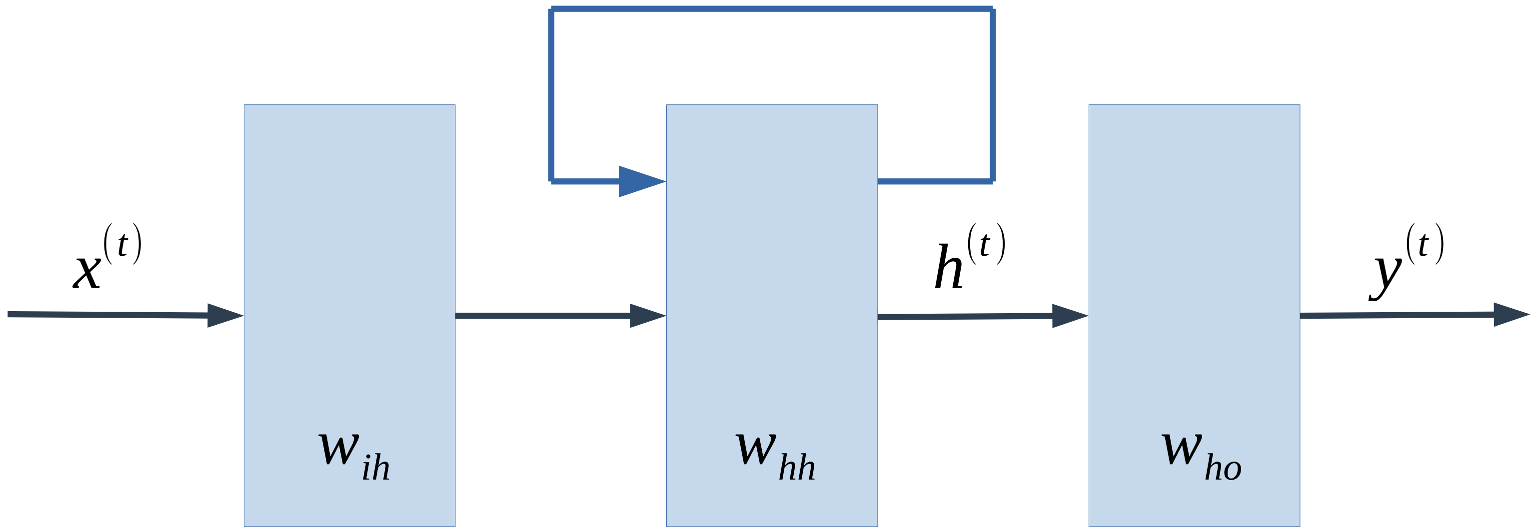


e.g. Video classification
on frame level

Source: CS231n Lecture 10

RNN Model

Vanilla RNN Model

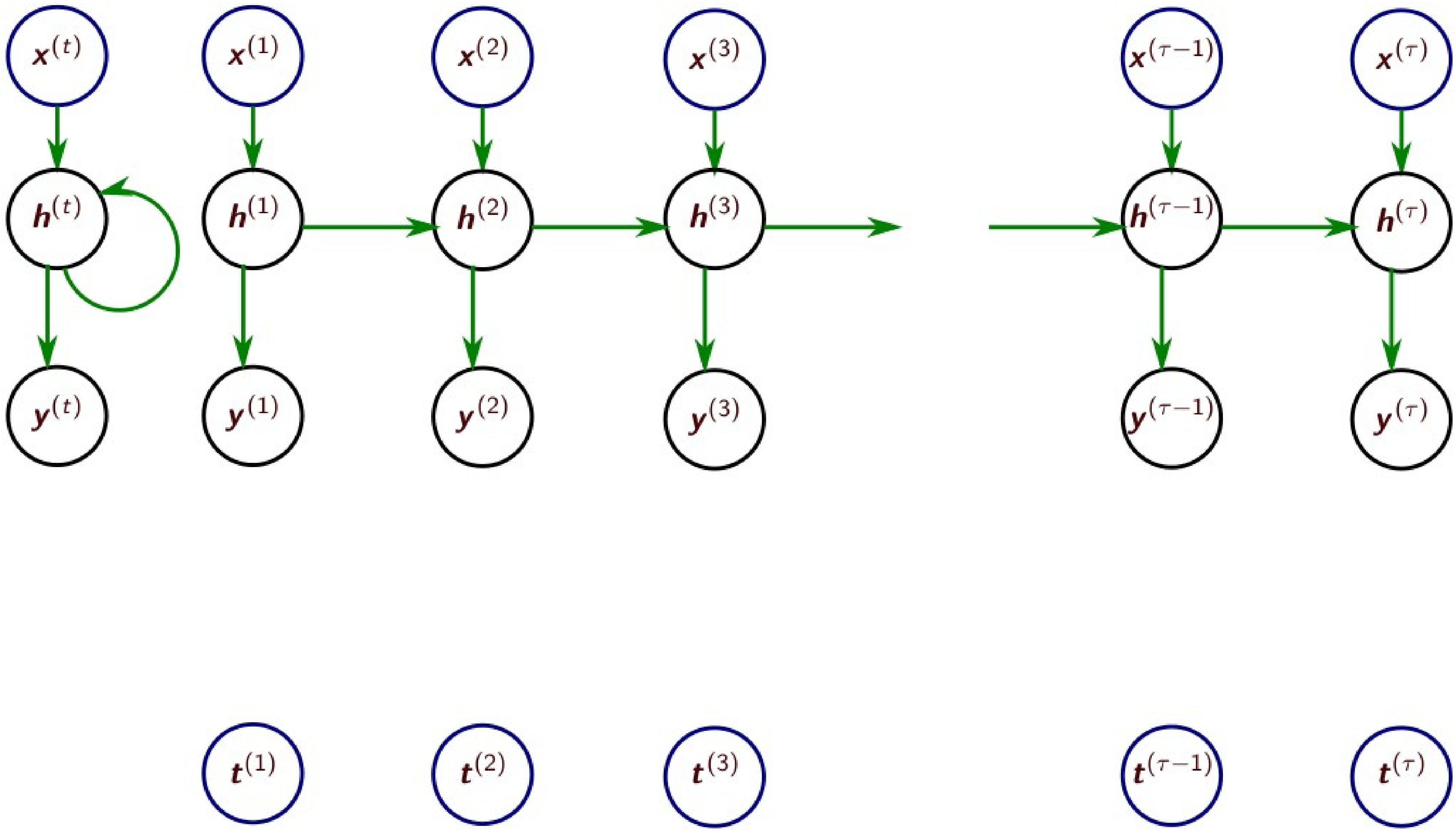


- Current state depends on *current inputs* and *previous state*
- RNNs can yield outputs at each time step

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

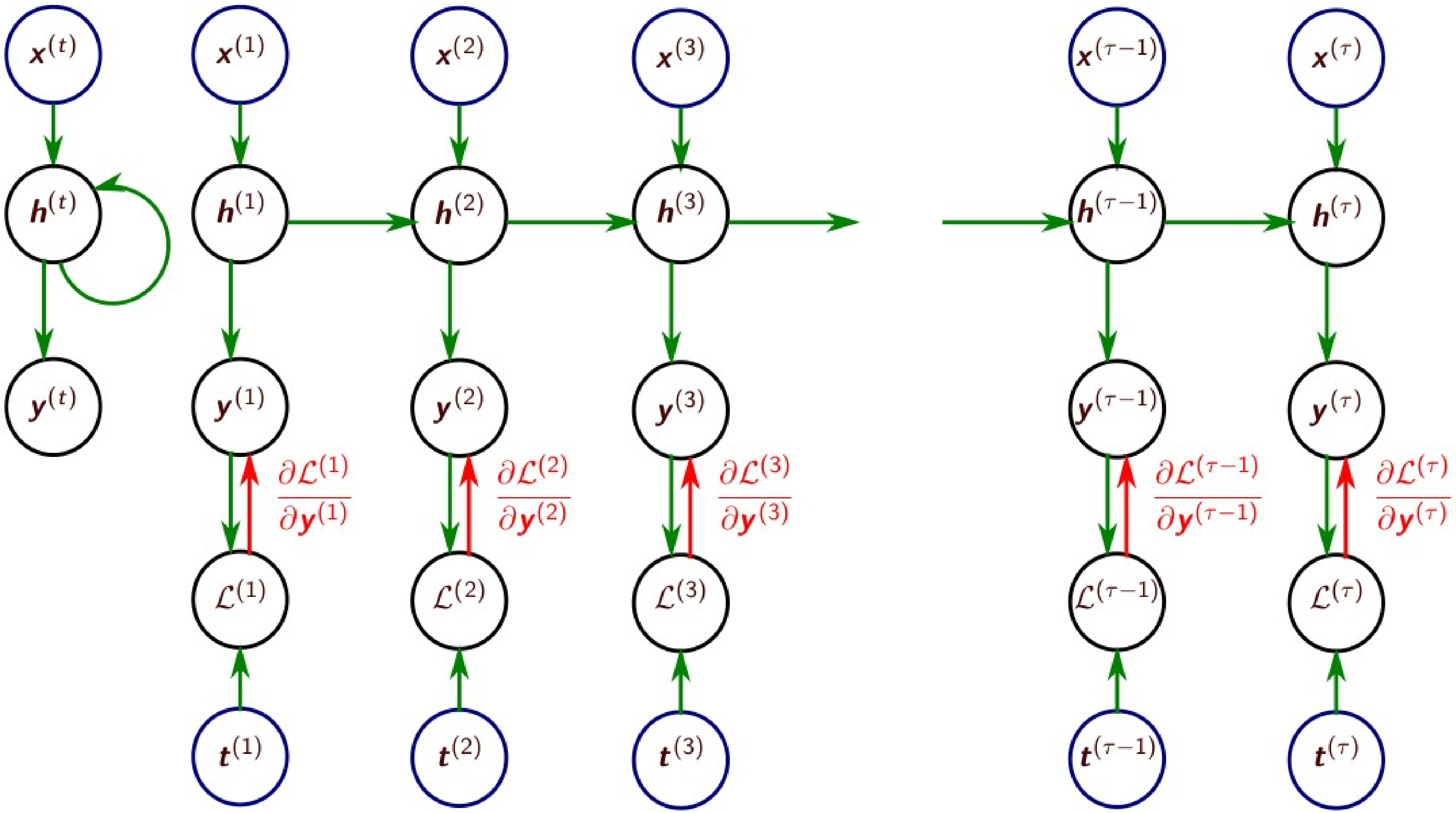
$$y^{(t)} = f_{w_{ho}}\left(h^{(t)}\right), \forall t \in \{1 \ldots \tau\}$$

Unfolding RNN in time



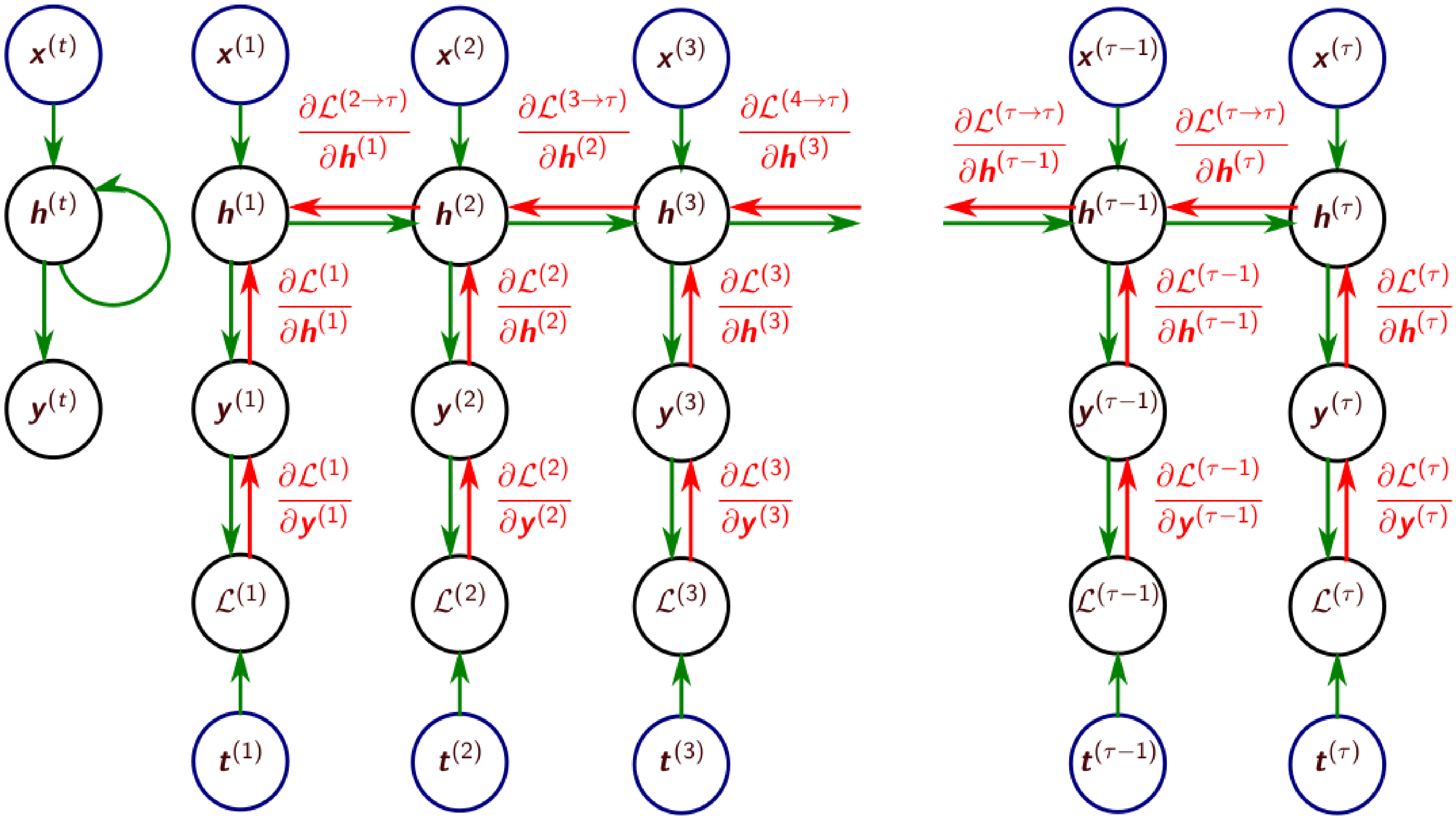
Source: NN Lectures, Tudor Berariu, 2016

Unfolding RNN in time



Source: NN Lectures, Tudor Berariu, 2016

Unfolding RNN in time



Source: NN Lectures, Tudor Berariu, 2016

Truncated BPTT

- Used in practice
- Summary of the algorithm:
  - Present a sequence of k1 timesteps of input and output pairs to the network.
  - Unroll the network then calculate and accumulate errors across k2 timesteps.
  - Roll-up the network and update weights.
  - Repeat

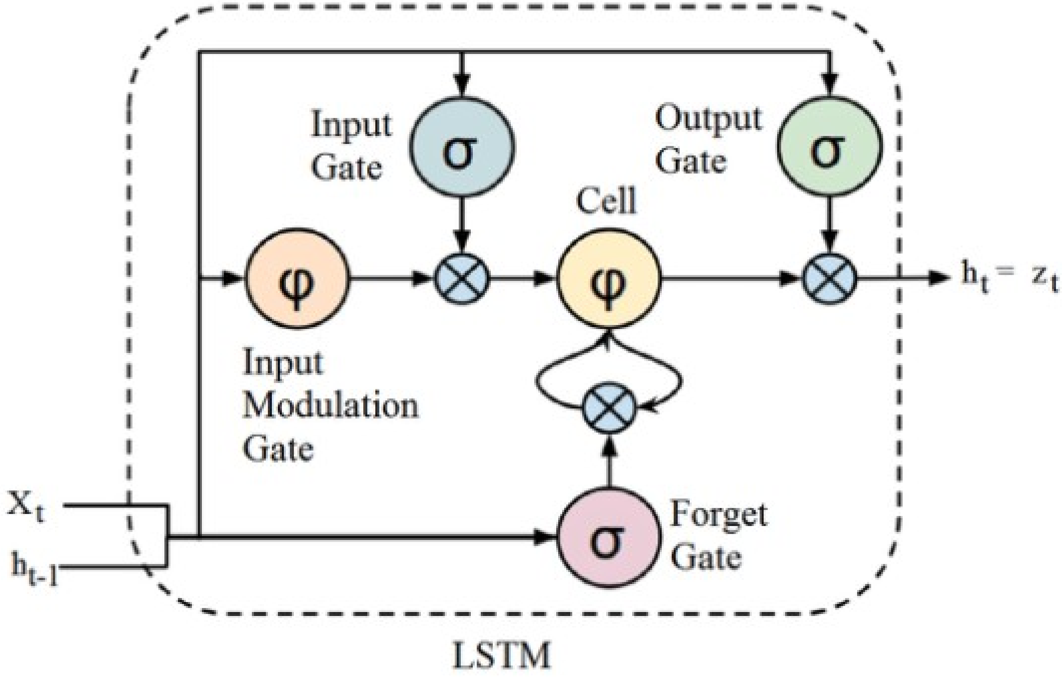
Teacher Forcing and Warm-start

- 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/@kangeugine/

- 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

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

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

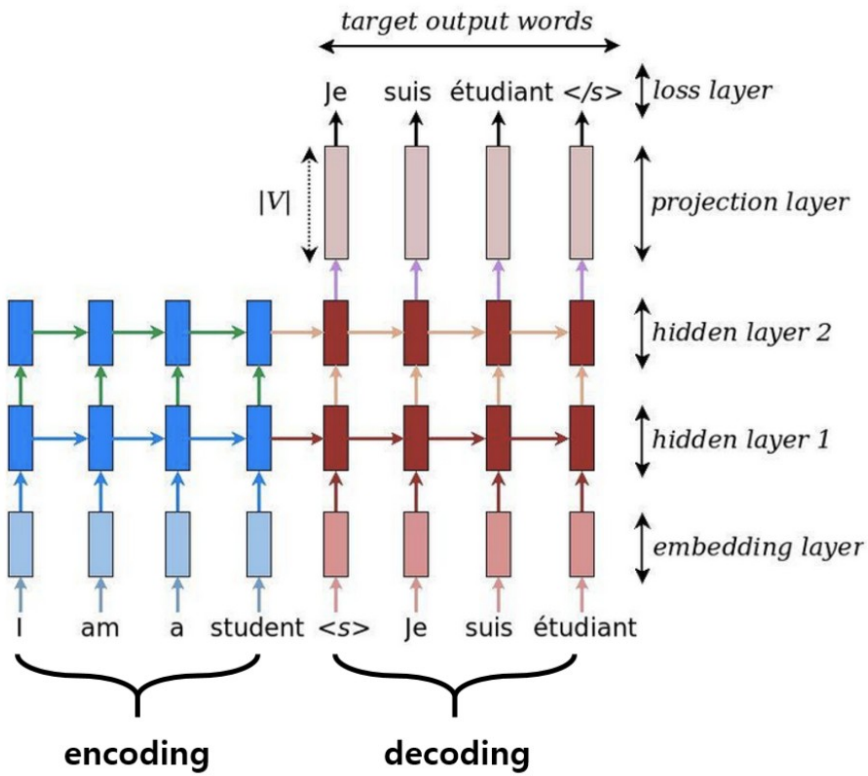
$$o_t = \sigma\left(\theta_{xo} x^{(t)} + \theta_{ho} h^{(t-1)} + b_o\right)$$

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

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

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

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