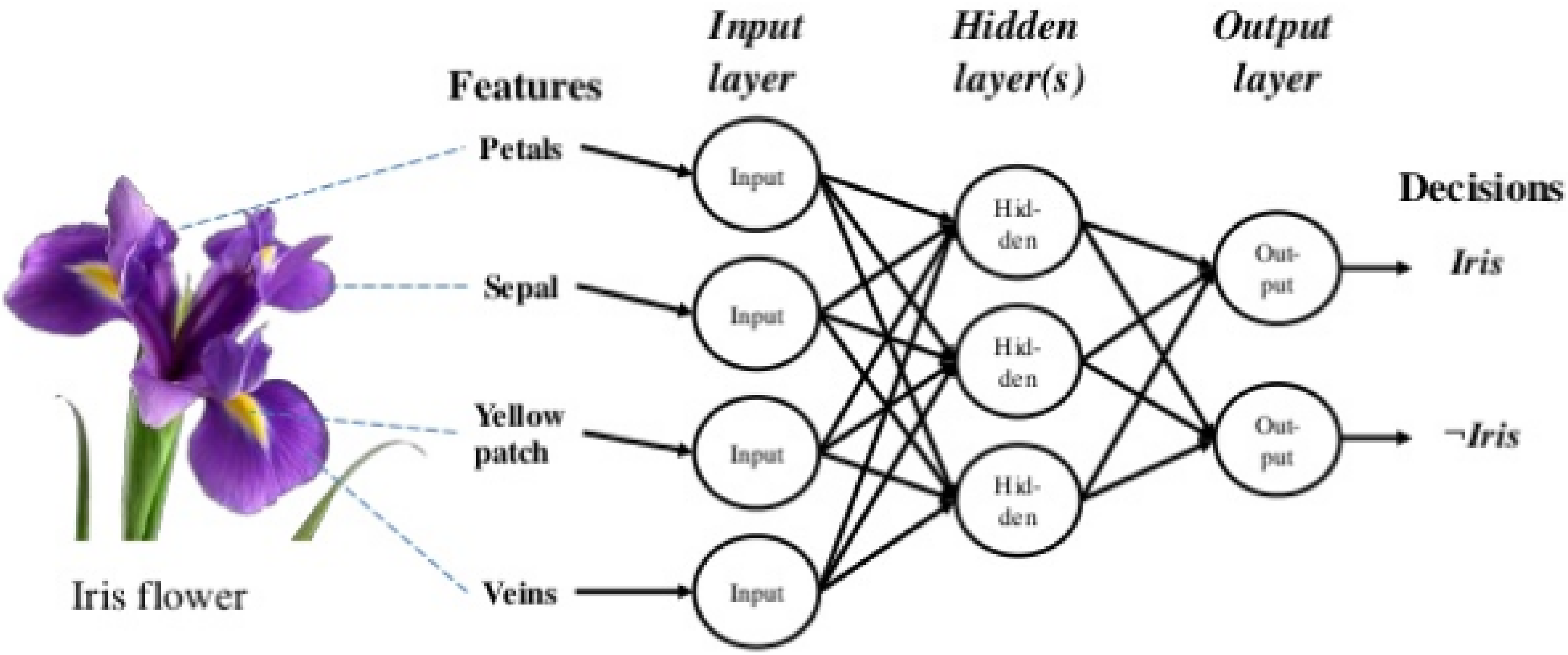
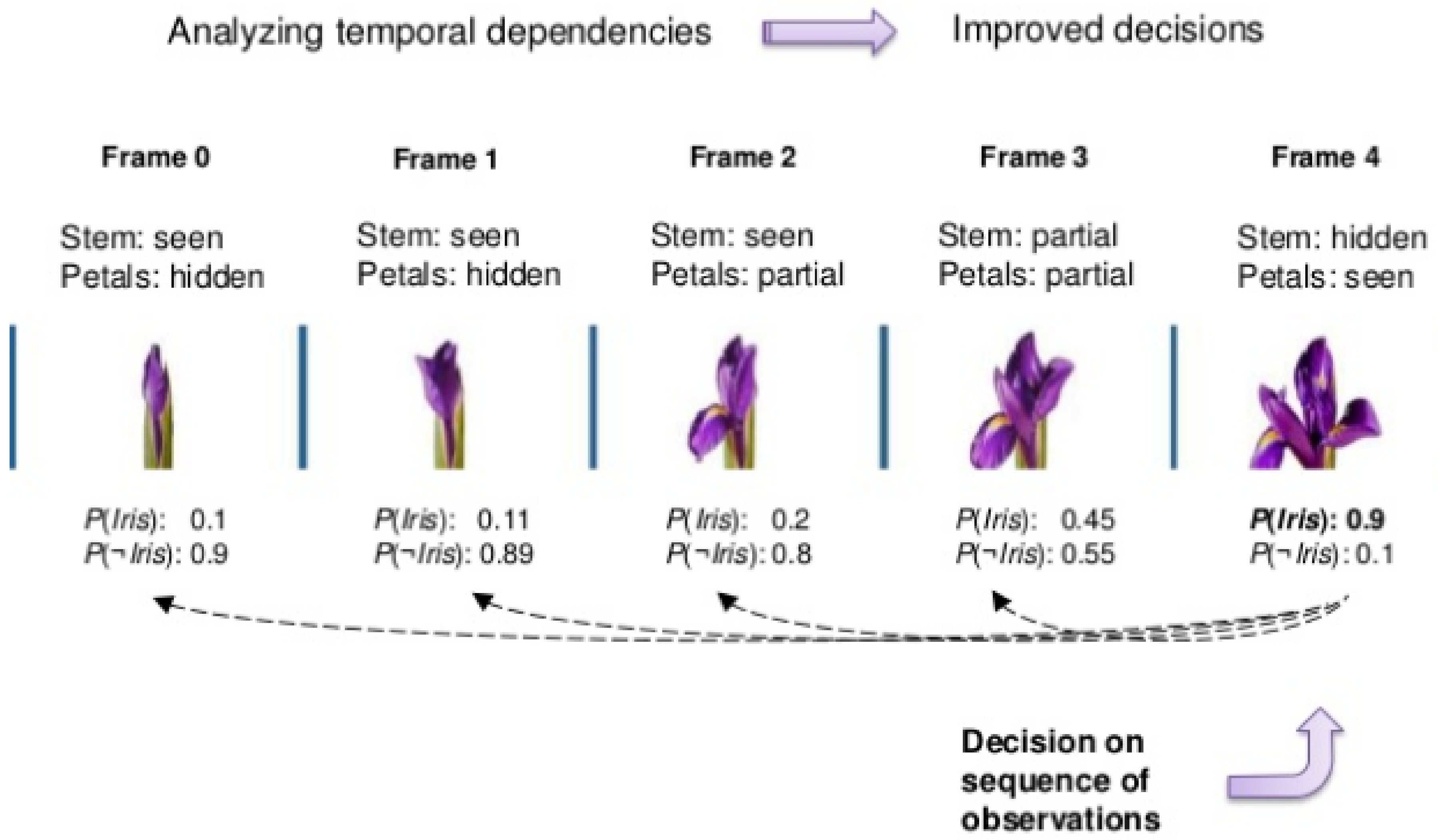
Motivation using RNN

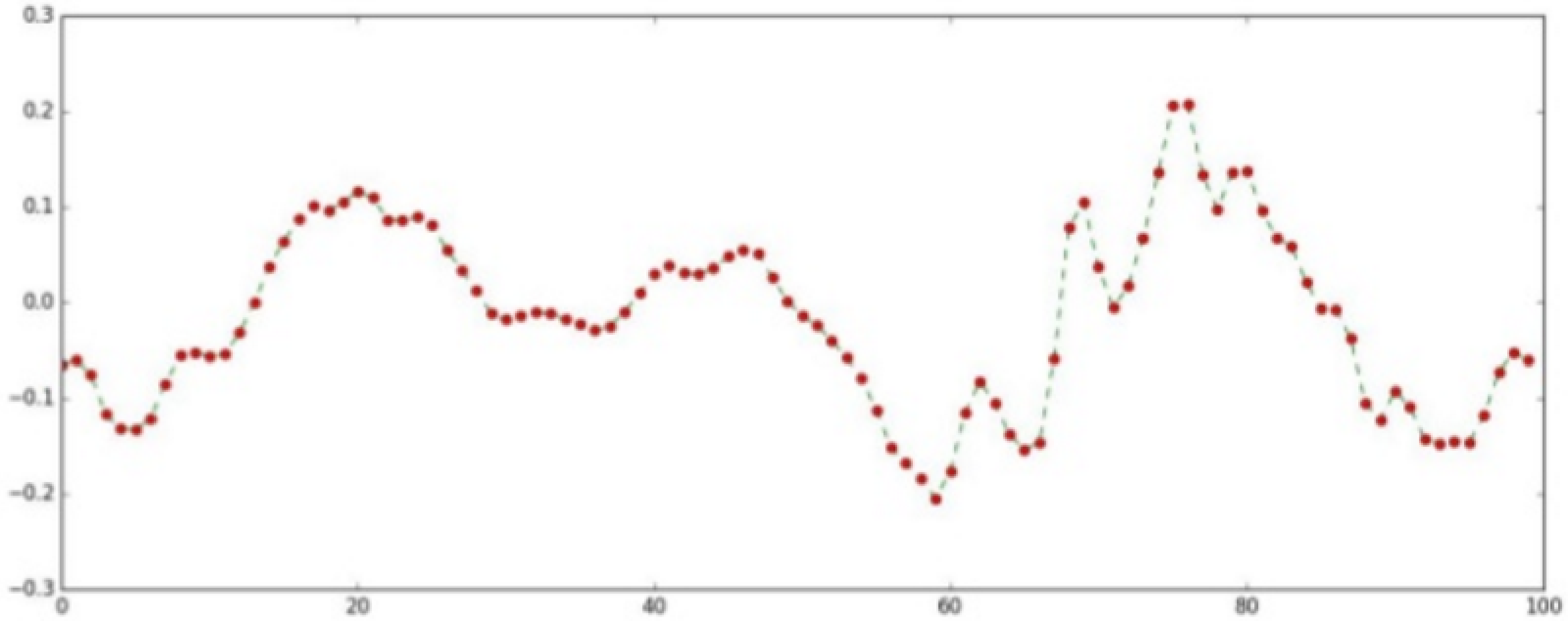


Motivation using RNN



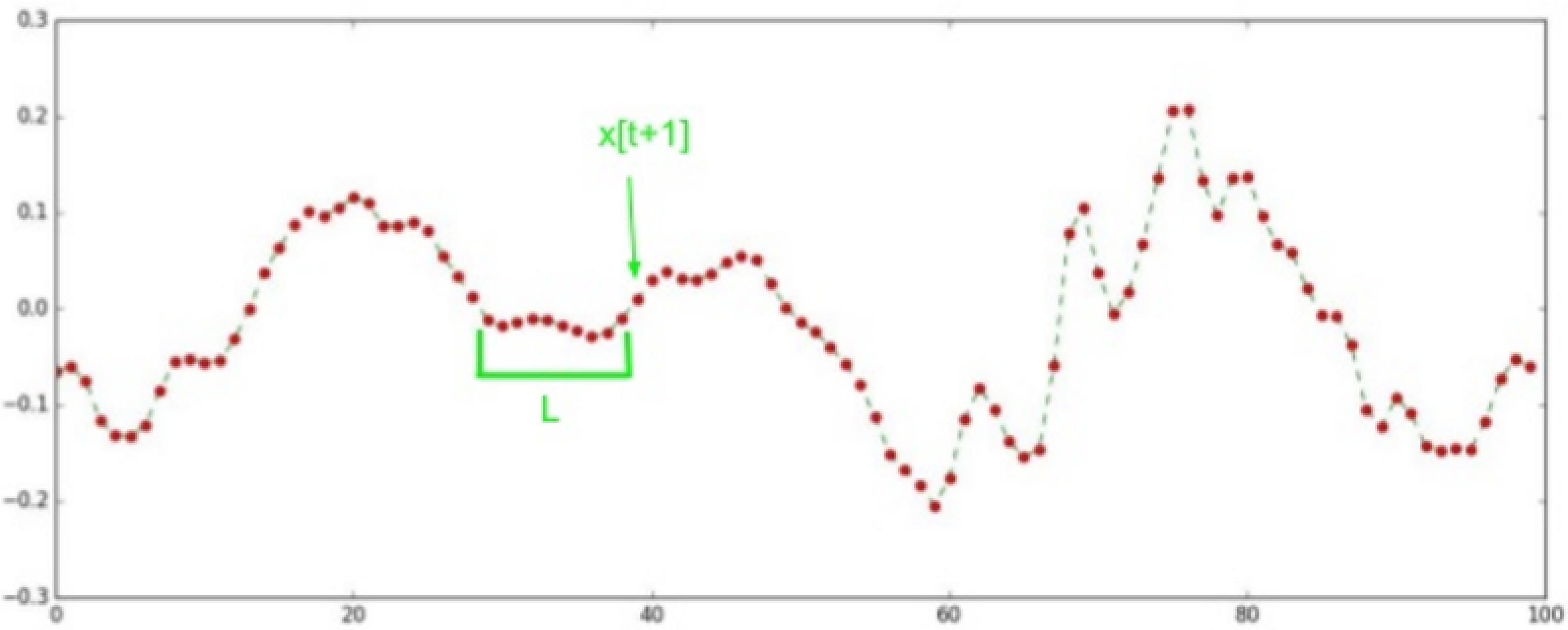
From FF to RNN

If we have a sequence of samples...



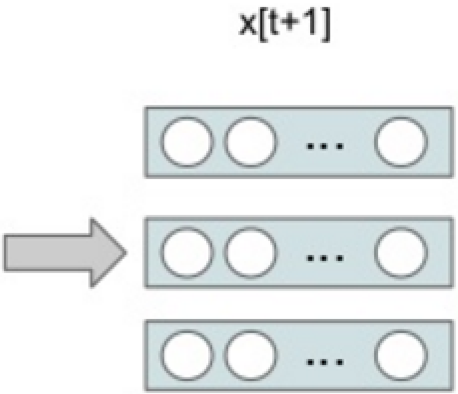
predict sample x[t+1] knowing previous values {x[t], x[t-1], x[t-2], …, x[t-τ]}

From FF to RNN

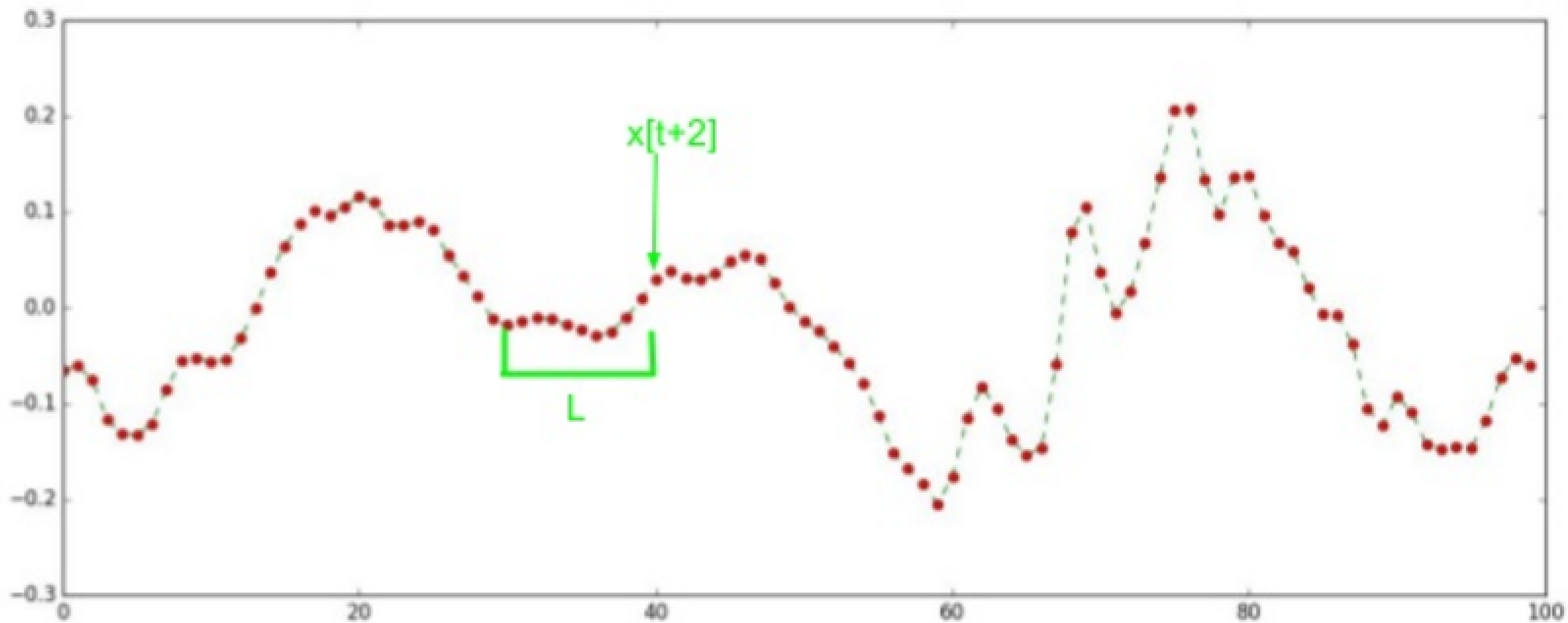


Feed Forward approach:

- static window of size L
- slide the window time-step wise

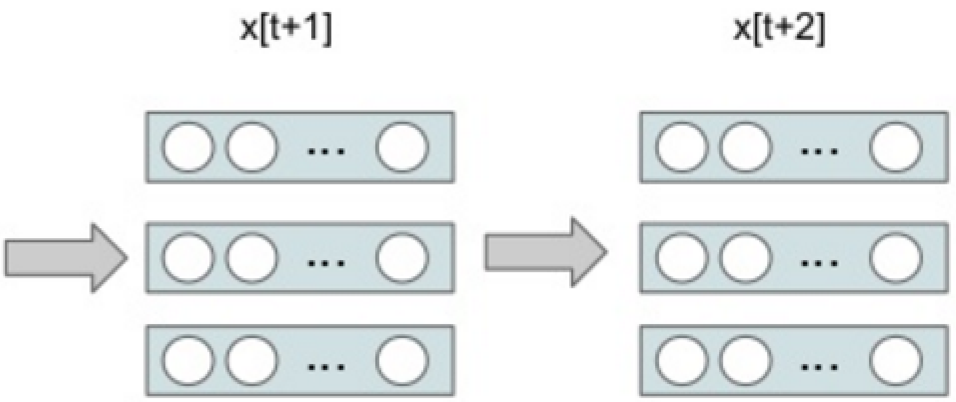


From FF to RNN

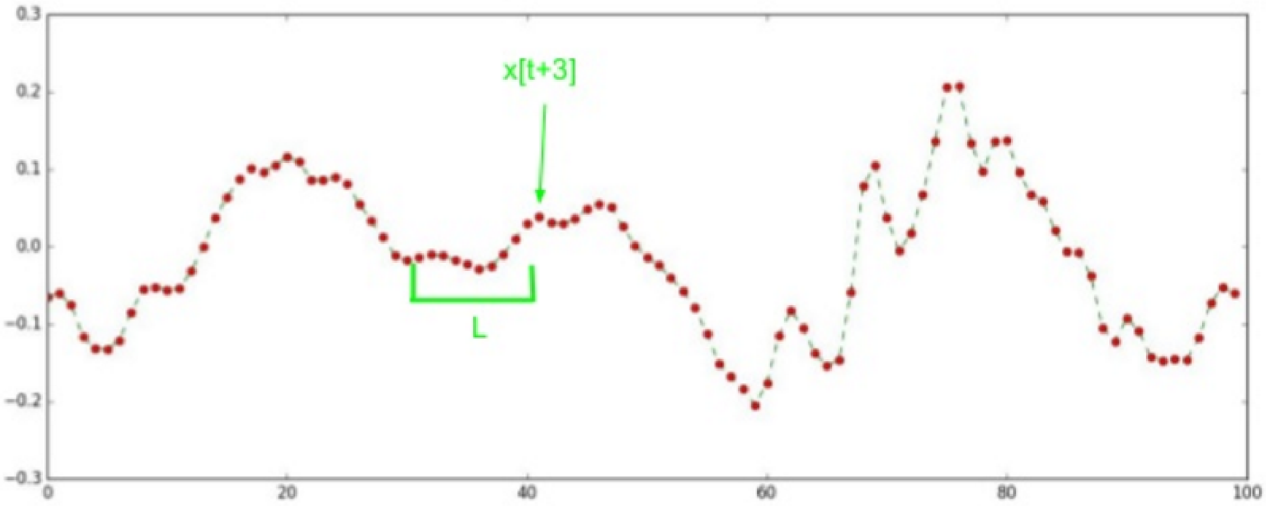


Feed Forward approach:

- static window of size L
- slide the window time-step wise

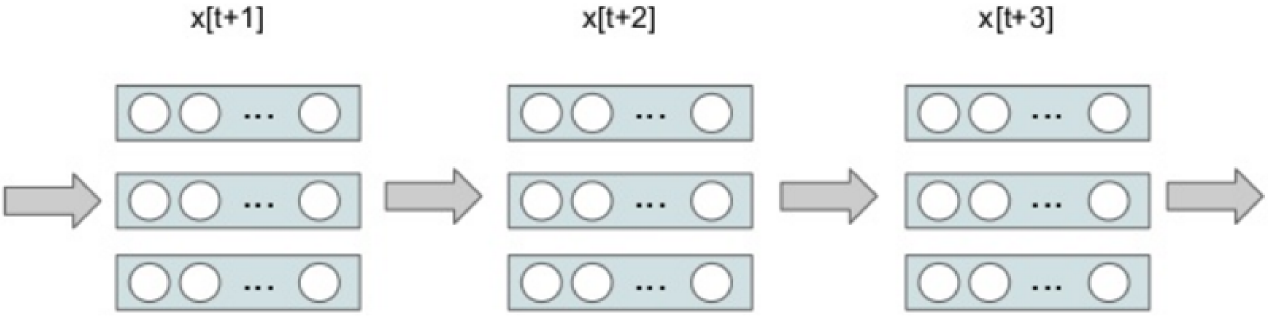


From FF to RNN



Feed Forward approach:

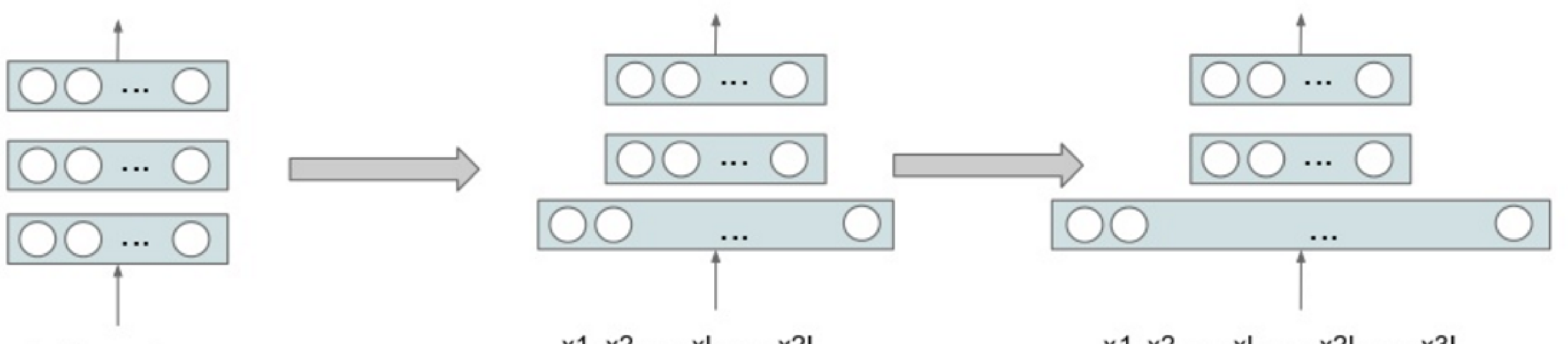
- static window of size L
- slide the window time-step wise



From FF to RNN

Problems for the feed forward + static window approach:

- What's the matter increasing L? → Fast growth of num of parameters!
- Decisions are independent between time-steps!
    - The network doesn't care about what happened at previous time-step, only present window matters → doesn't look good
- Cumbersome padding when there are not enough samples to fill L size
    - Can't work with variable sequence lengths

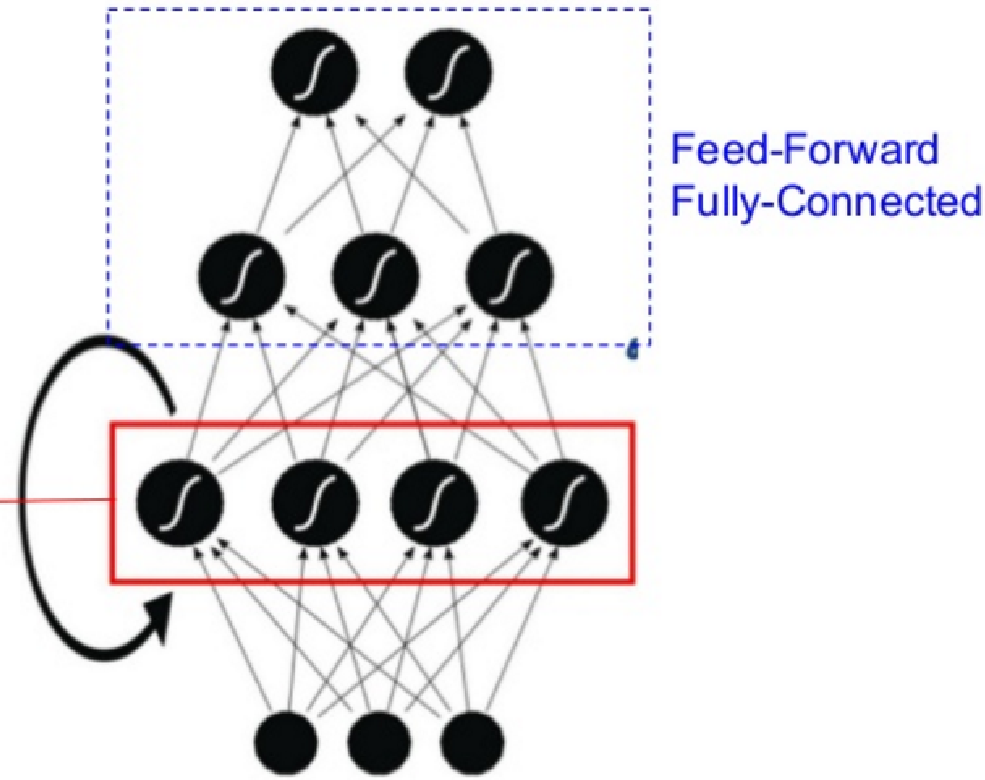


From FF to RNN

Solution: Build specific connections capturing the temporal evolution → **Shared weights in time**

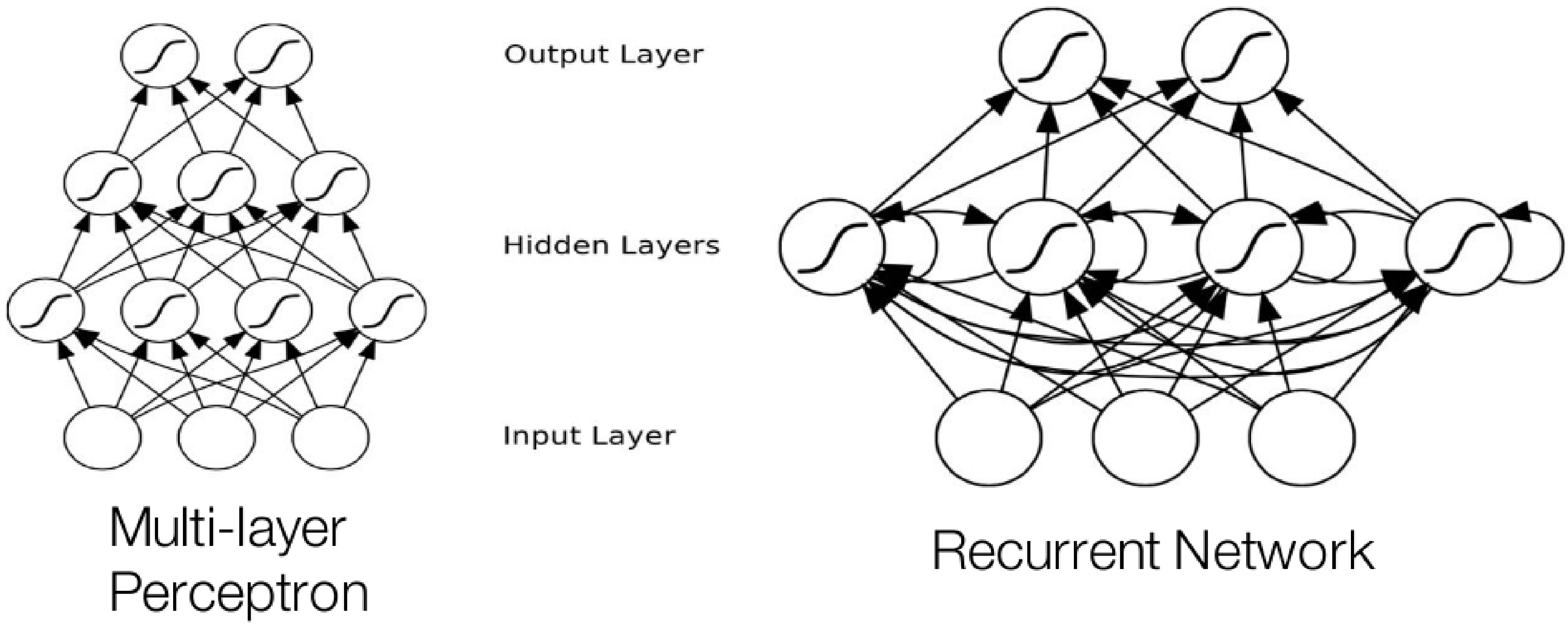
- Give volatile memory to the network

$$\boxed{h_t = f(W \cdot x_t + U \cdot h_{t-1} + b)}$$

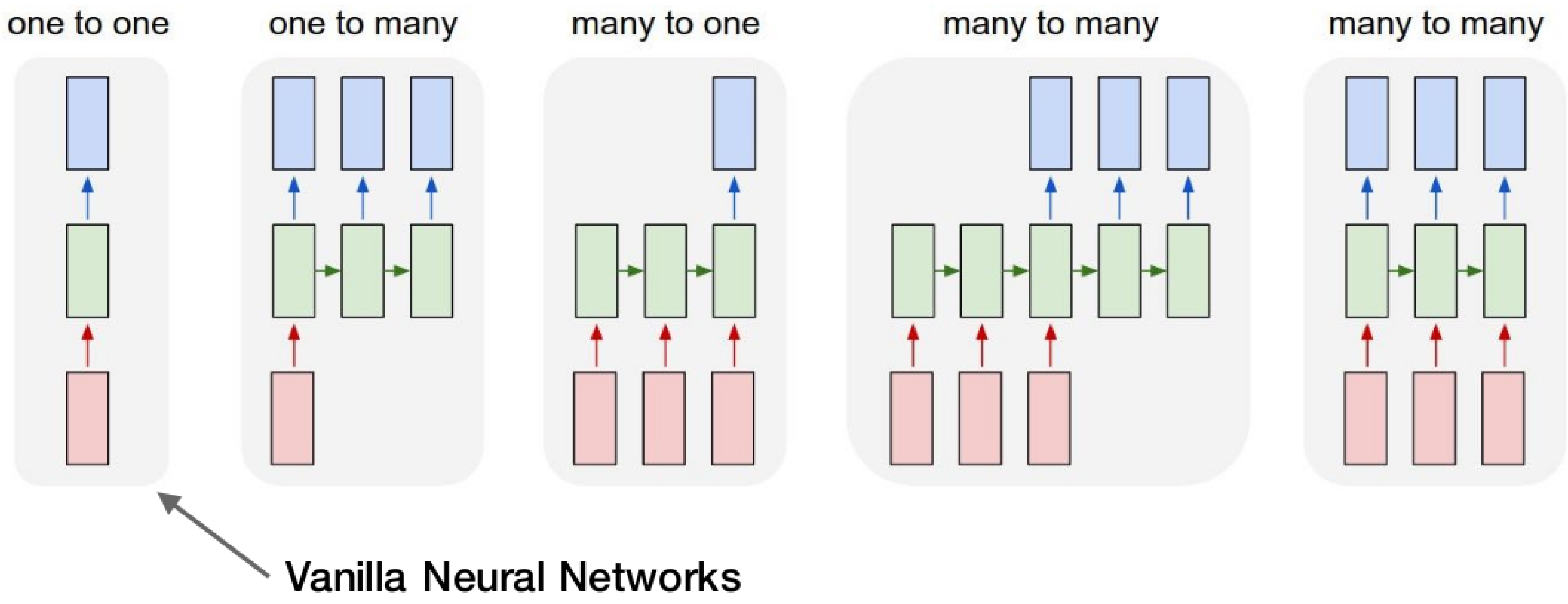


RNN

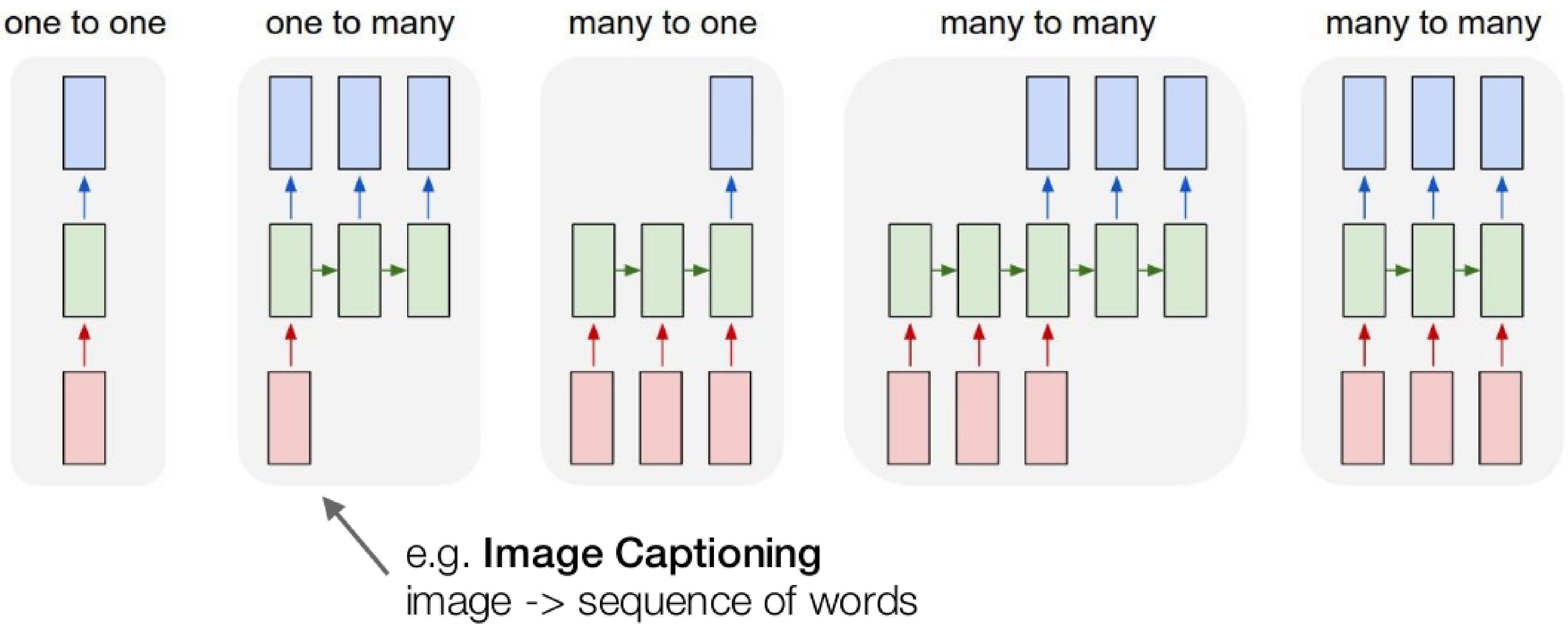
- An MLP can only map from input to output vectors, whereas an RNN can, in principle, map from the entire history of previous inputs to each output.



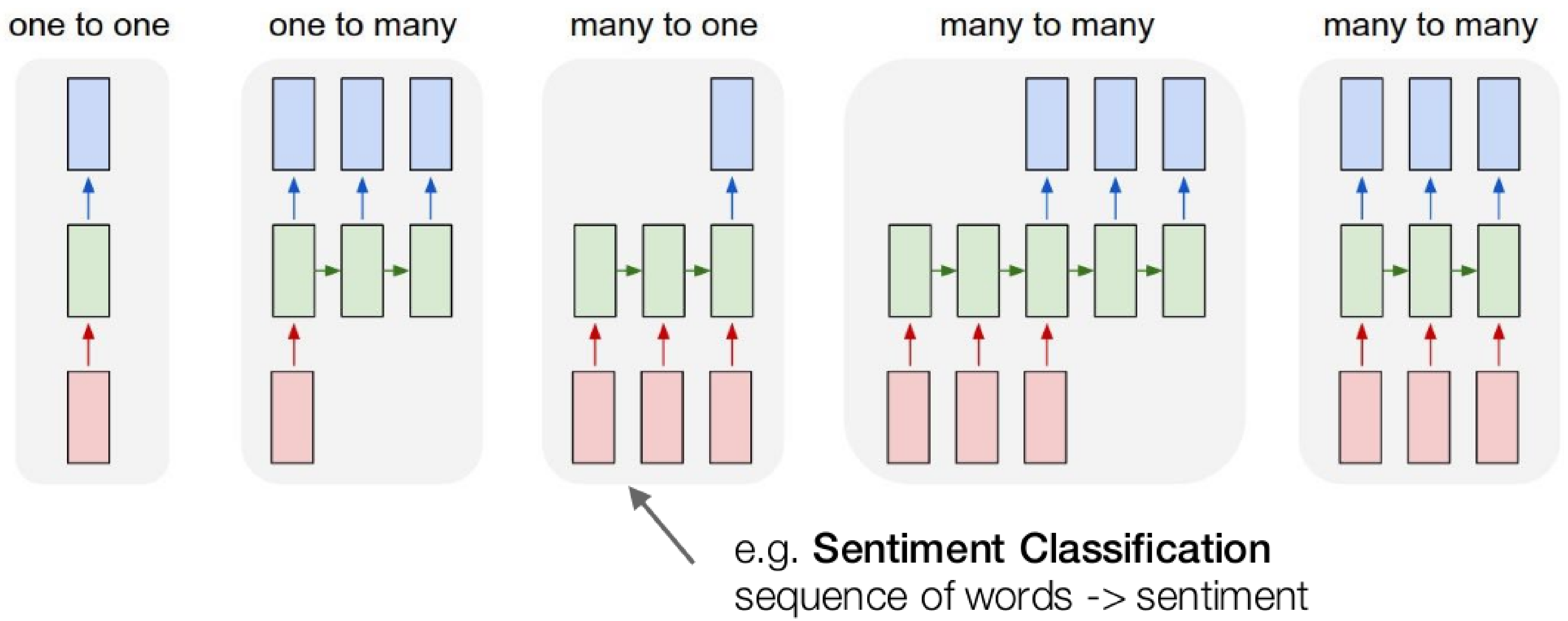
RNN offer a lot of flexibility



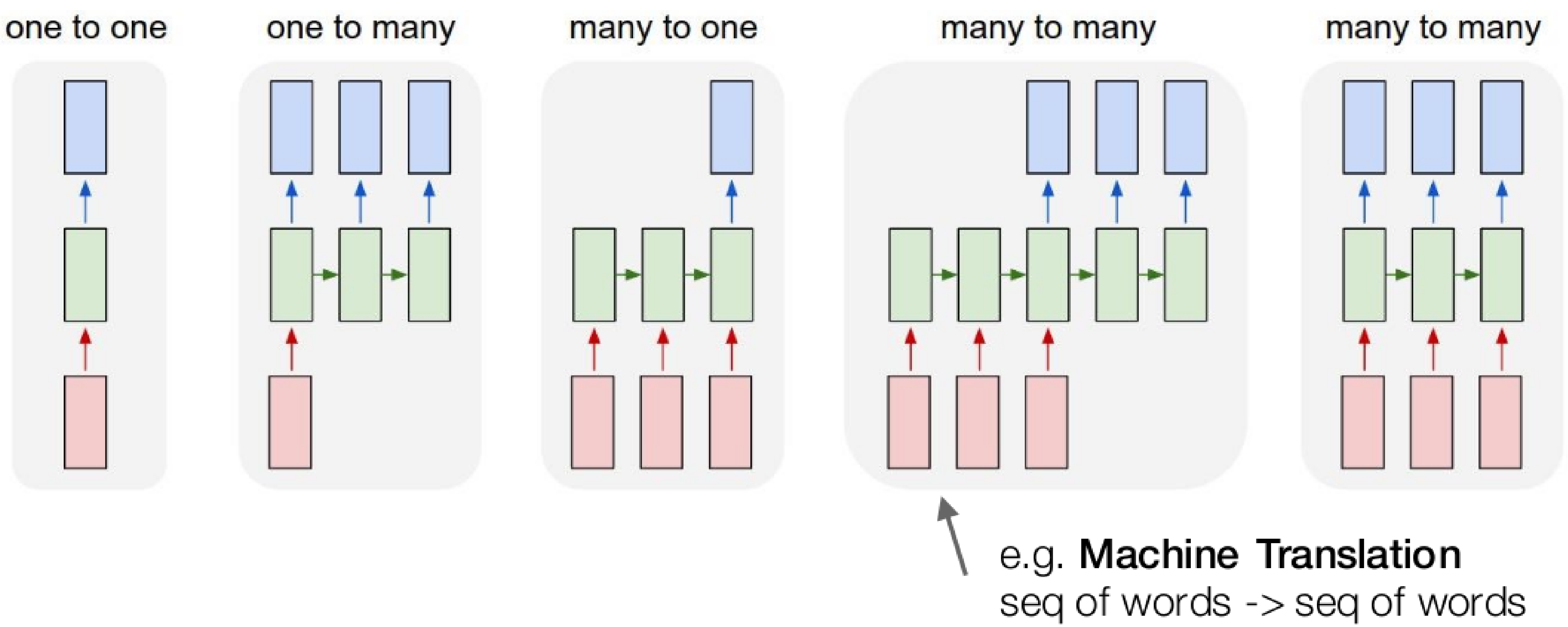
RNN offer a lot of flexibility



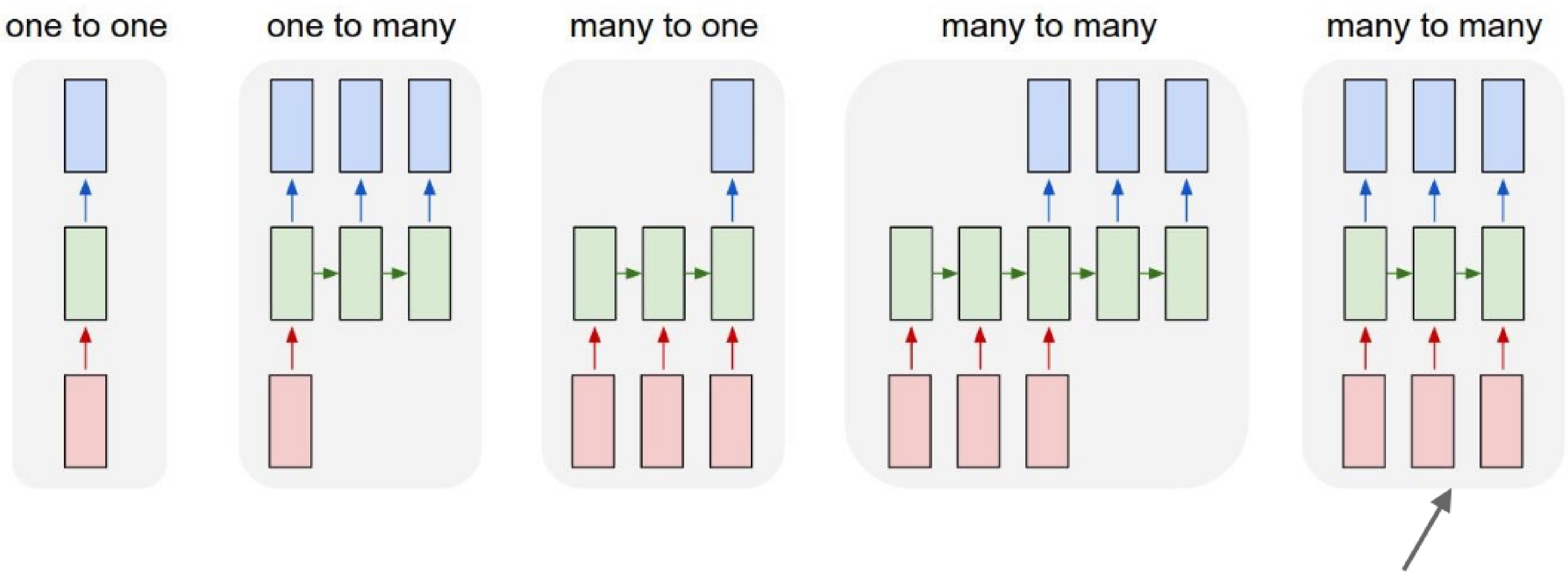
RNN offer a lot of flexibility



RNN offer a lot of flexibility



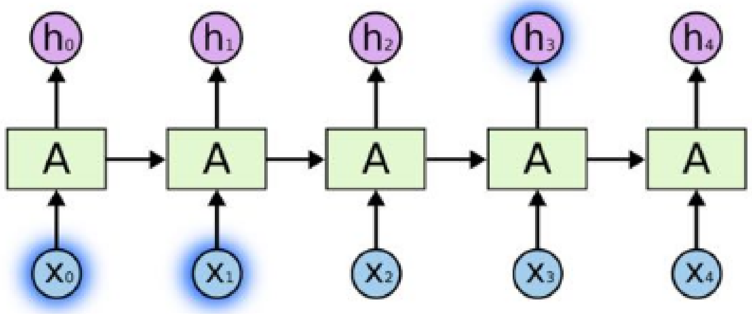
RNN offer a lot of flexibility



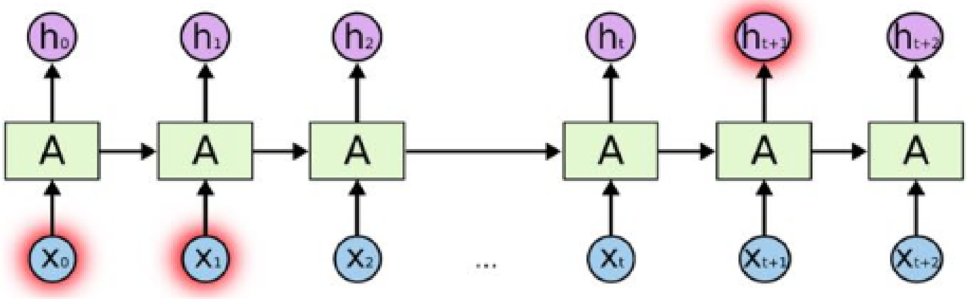
e.g. **Video classification on frame level**

The problem of long-term dependencies

- (Vanilla) RNNs connect previous information to present task:
- - enough for predicting the next word for "the clouds are in the *sky*"

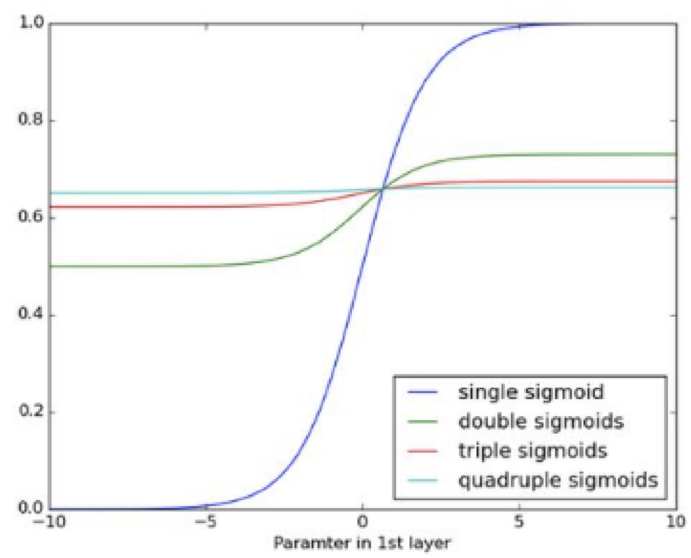


- - may not be enough when more context is needed
- "I grew up in France… I speak fluent *French*."



The problem of vanishing gradients

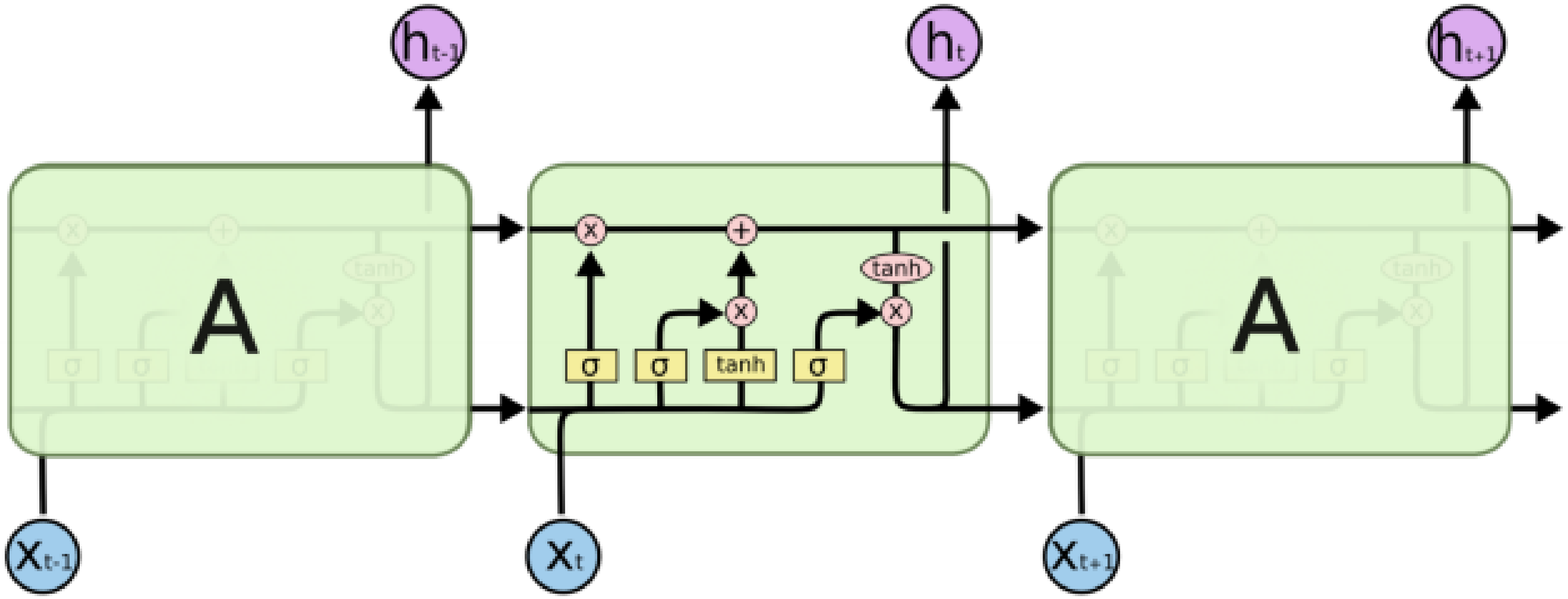
- In a traditional recurrent neural network, during the gradient backpropagation phase, the gradient signal can end up being multiplied a large number of times
- If the gradients are large
  - Exploding gradients, learning diverges
  - **Solution: Clip the gradients to a certain max value.**
- If the gradients are small
  - Vanishing gradients, learning very slow or stops
  - **Solution: introducing memory via LSTM, GRU, etc.**



Long Short Term Memory (LSTM) [Hochreiter & Schmidhuber (1997)]

- Learning long-term dependencies is difficult with simple RNNs, unstable training due to vanishing gradients problem [Hochreiter, 1991]
- Limited capability (5-10 time steps) to model long-term dependencies
- LSTM RNN architecture designed to address these problems [Hochreiter and Schmidhuber, 1997]
- LSTM memory block: memory cell storing temporal state of network and 3 multiplicative units (gates) controlling the flow of information

Long Short Term Memory (LSTM) [Hochreiter & Schmidhuber (1997)]



Long Short Term Memory (LSTM) [Hochreiter & Schmidhuber (1997)]

Key difference to RNN: a memory cell c which is controlled by three gates:

- input gate i:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{7}$$

- output gate o:

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{8}$$

- forget gate f:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{9}$$

- update of memory cell and hidden state:

$$\widetilde{C_t} = tanh(W_c x_t + U_c h_{t-1} + b_C) \tag{10}$$

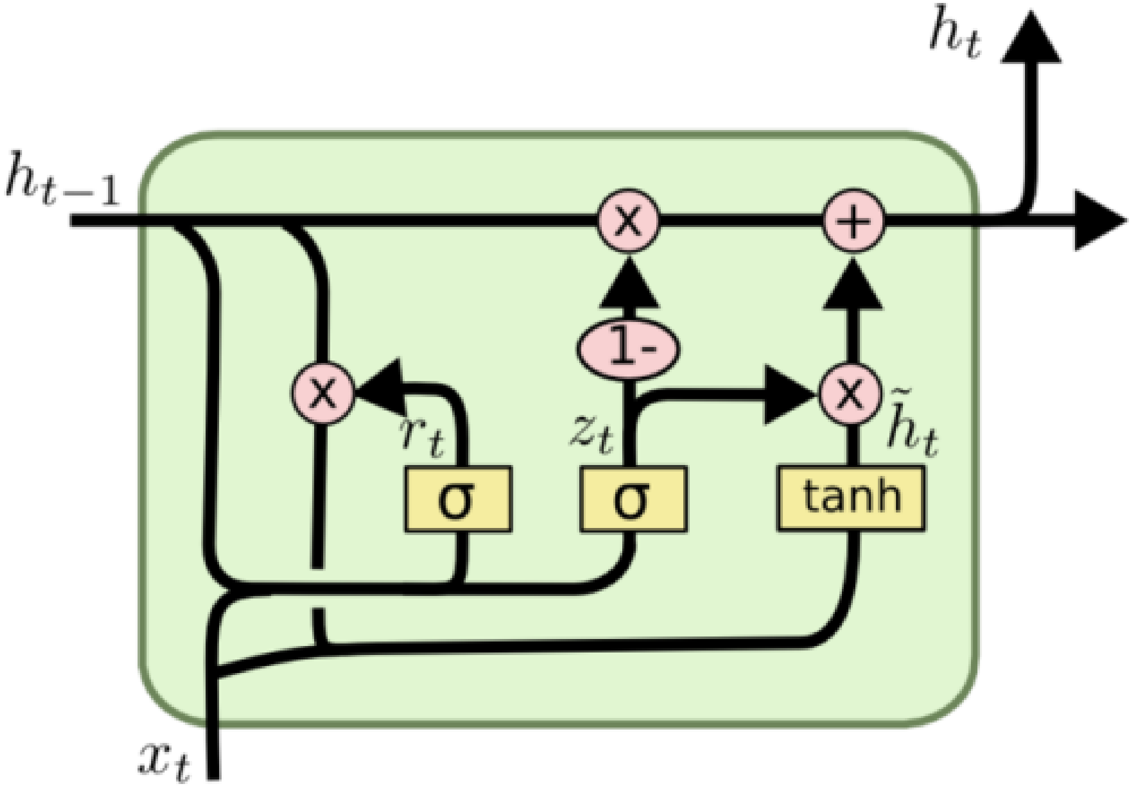
$$C_t = f_t \otimes C_{t-1} + i_t \otimes \widetilde{C_t} \tag{11}$$

$$h_t = o_t \otimes tanh(C_t) \tag{12}$$

Long Short Term Memory (LSTM) [Hochreiter & Schmidhuber (1997)]

- LSTM performs better than RNN for learning context-free and context-sensitive languages [Gers and Schmidhuber, 2001]
- Bidirectional LSTM for phonetic labeling of acoustic frames on the TIMIT [Graves and Schmidhuber, 2005]
- Online and offline handwriting recognition with bidirectional LSTM better than HMM-based system [Graves et al., 2009]
- Deep LSTM - stack of multiple LSTM layers - combined with CTC and RNN transducer predicting phone sequences gets state of the art results on TIMIT [Graves et al., 2013]

Gated Recurrent Unit



Gated Recurrent Unit

Instead of using three gates, only two gates is employed:

- update gate z:

$$z_t = \sigma(W_z x_t + U_z h_{t-1}) \tag{13}$$

- reset gate r:

$$r_t = \sigma(W_r x_t + U_r h_{t-1}) \tag{14}$$

- state update:

$$\widetilde{h_t} = tanh(W_c x_t + U(r_t \otimes h_{t-1})) \tag{15}$$

$$h_t = (1 - z_t) \otimes h_{t-1} + z_t \otimes \widetilde{h_t} \tag{16}$$

Multivariate Forecasting

Beijing Air Quality Dataset

- No: row number
- year: year of data in this row
- month: month of data in this row
- day: day of data in this row
- hour: hour of data in this row
- pm2.5: PM2.5 concentration
- DEWP: Dew Point
- TEMP: Temperature
- PRES: Pressure
- cbwd: Combined wind direction
- Iws: Cumulated wind speed
- Is: Cumulated hours of snow
- Ir: Cumulated hours of rain

Multivariate Forecasting