Recurrent Neural Networks II

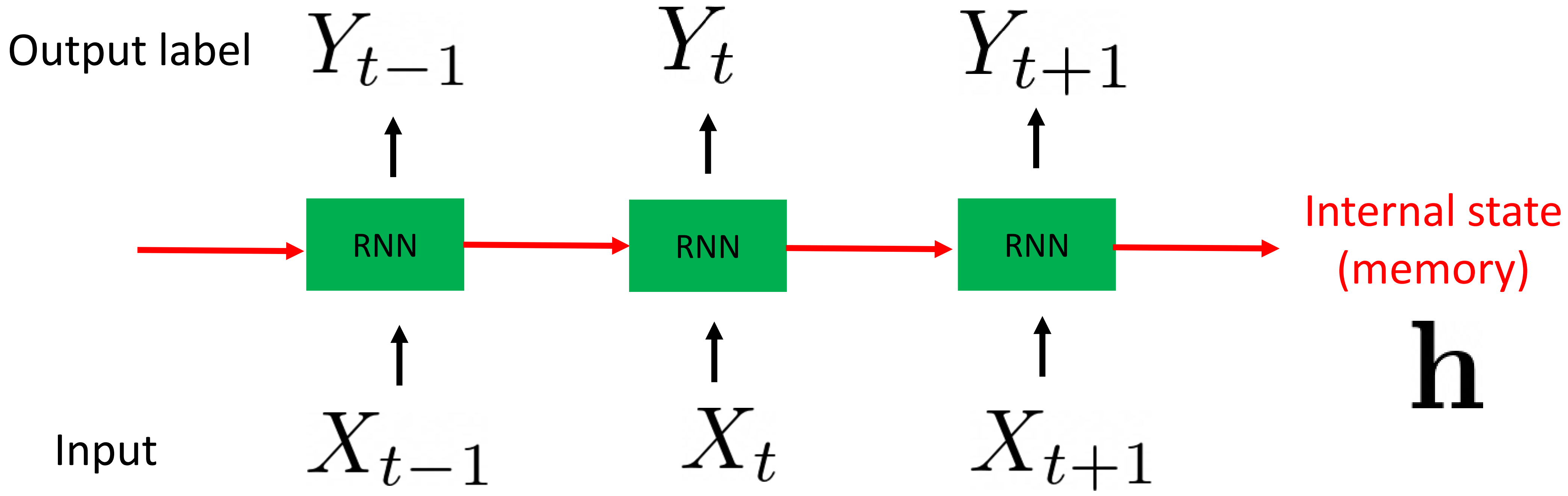
CS 4391 Introduction Computer Vision

Professor Yu Xiang

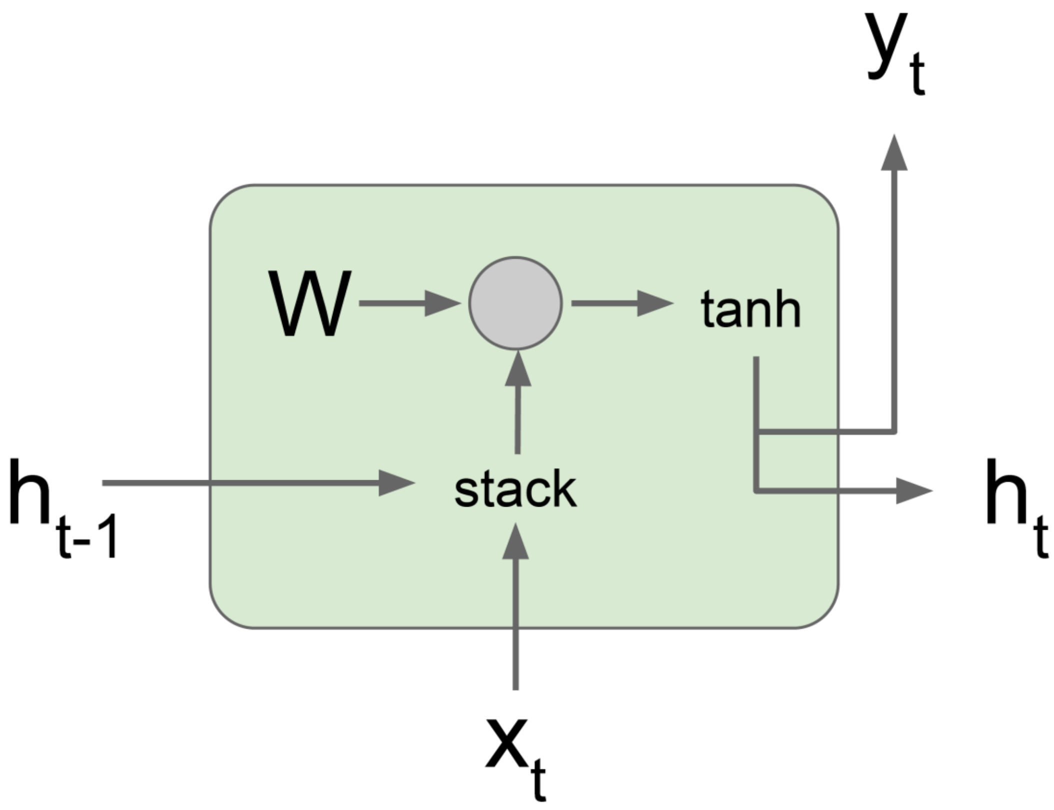
The University of Texas at Dallas

Some slides of this lecture are courtesy Stanford CS231n

Recurrent Neural Networks



Vanilla RNN



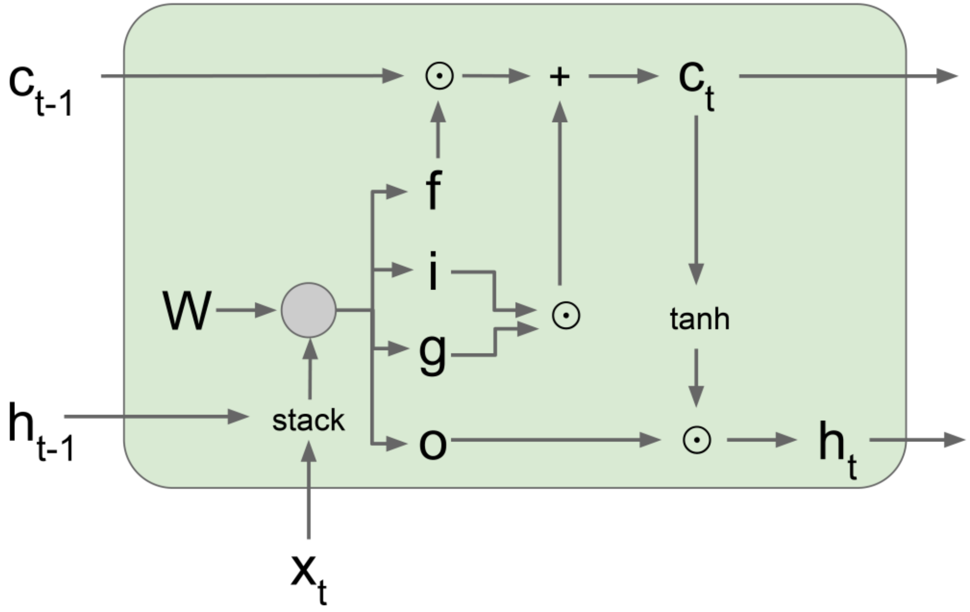
$$\mathbf{h}_t = \tanh(W_{hh}\mathbf{h}_{t-1} + W_{xh}\mathbf{x}_t)$$

$$= \tanh\left(\begin{pmatrix} W_{hh} & W_{hx} \end{pmatrix} \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

$$= \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

$$\mathbf{y}_t = W_{hy}\mathbf{h}_t$$

Long Short Term Memory (LSTM)



LSTM

$$\begin{array}{l} \text{Input gate} \\ \text{forget gate} \\ \text{output gate} \\ \text{gate gate} \end{array} \quad \begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

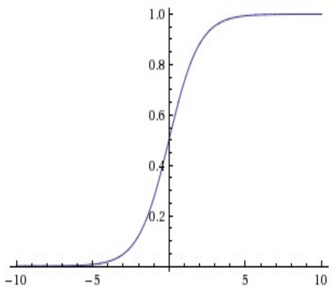
Cell $c_t = f \odot c_{t-1} + i \odot g$

Hidden state $h_t = o \odot \tanh(c_t)$

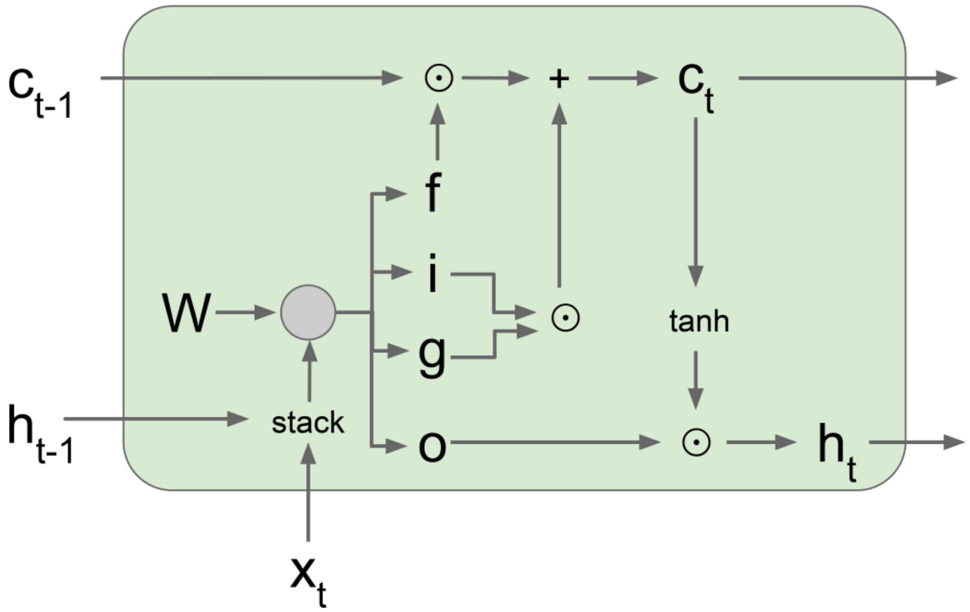
Store Cell and hidden states

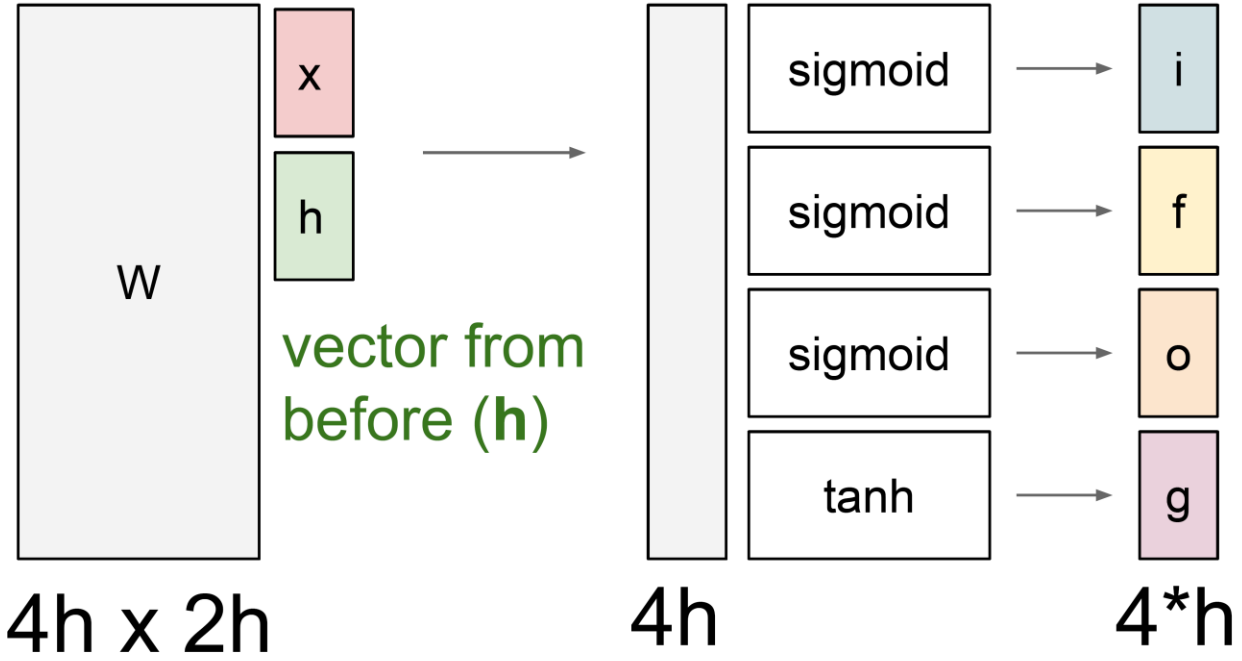
Sigmoid

$$\sigma(x) = 1/(1 + e^{-x})$$



Long Short Term Memory (LSTM)





$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

- **g**: Gate gate, how much to write to cell
- **i**: Input gate, whether to write to cell
- **f**: Forget gate, whether to erase cell
- **o**: Output gate, how much to reveal cell

Long Short Term Memory (LSTM)

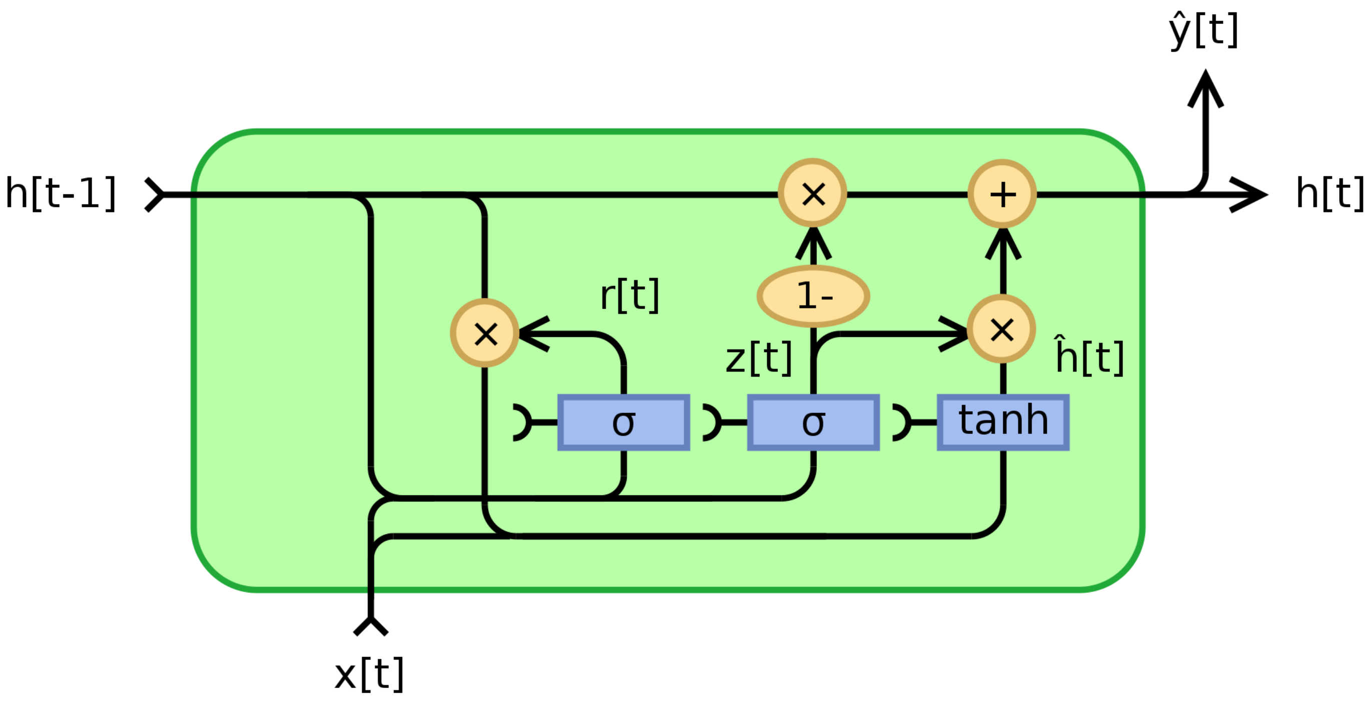
- Make the RNN easier to preserve information over many steps
  - E.g., f = 1 and i = 0
  - This is difficult for vanilla RNN
- LSTM does not guarantee that there is no vanishing or exploding gradient
- It provides an easier way to learn long-distance dependencies

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

Gated Recurrent Unit (GRU)



https://en.wikipedia.org/wiki/Gated_recurrent_unit

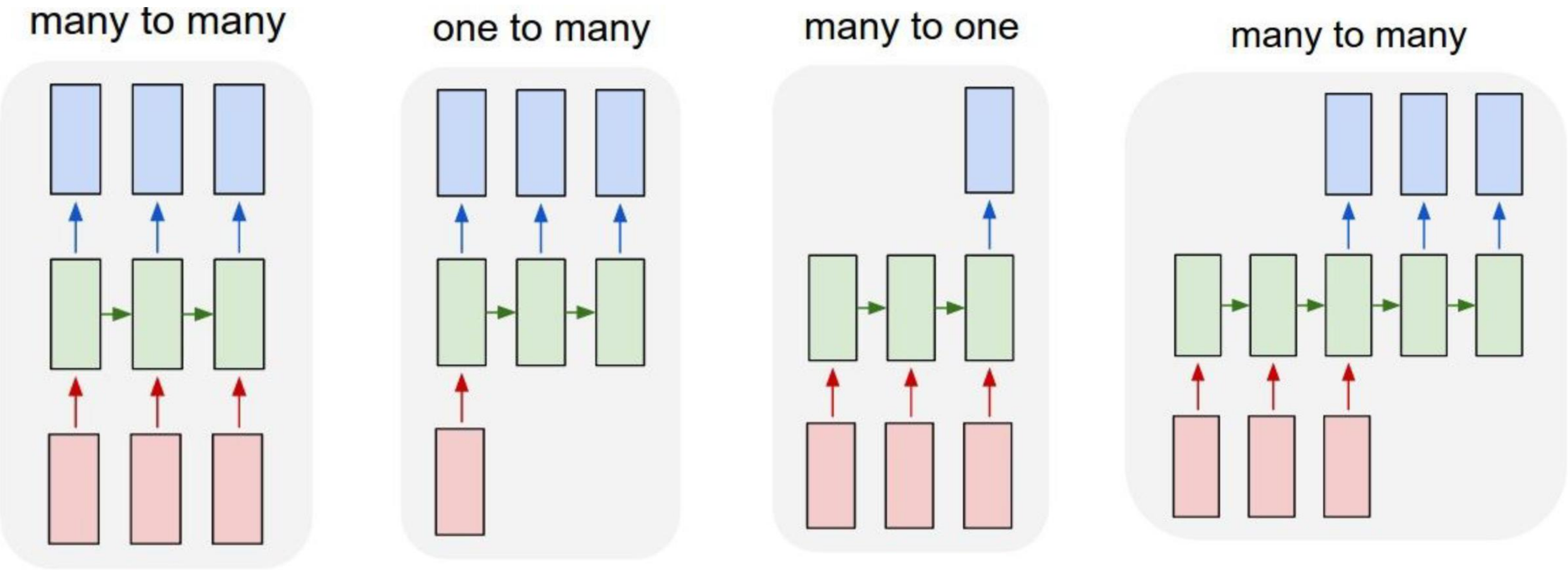
$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z)$$
$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r)$$
$$\hat{h}_t = \phi_h(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h)$$
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h}_t$$

- x_t: input vector
- h_t: output vector
- $\hat{h}_t$: candidate activation vector
- z_t: update gate vector
- r_t: reset gate vector
- W, U and b: parameter matrices and vector

GRUs vs. LSTMs

- Both have a forget gate
- GRU has fewer parameters, no output gate
- GRUs have similar performance compared to LSTMs, have shown better performance on certain datasets

Recurrent Neural Networks



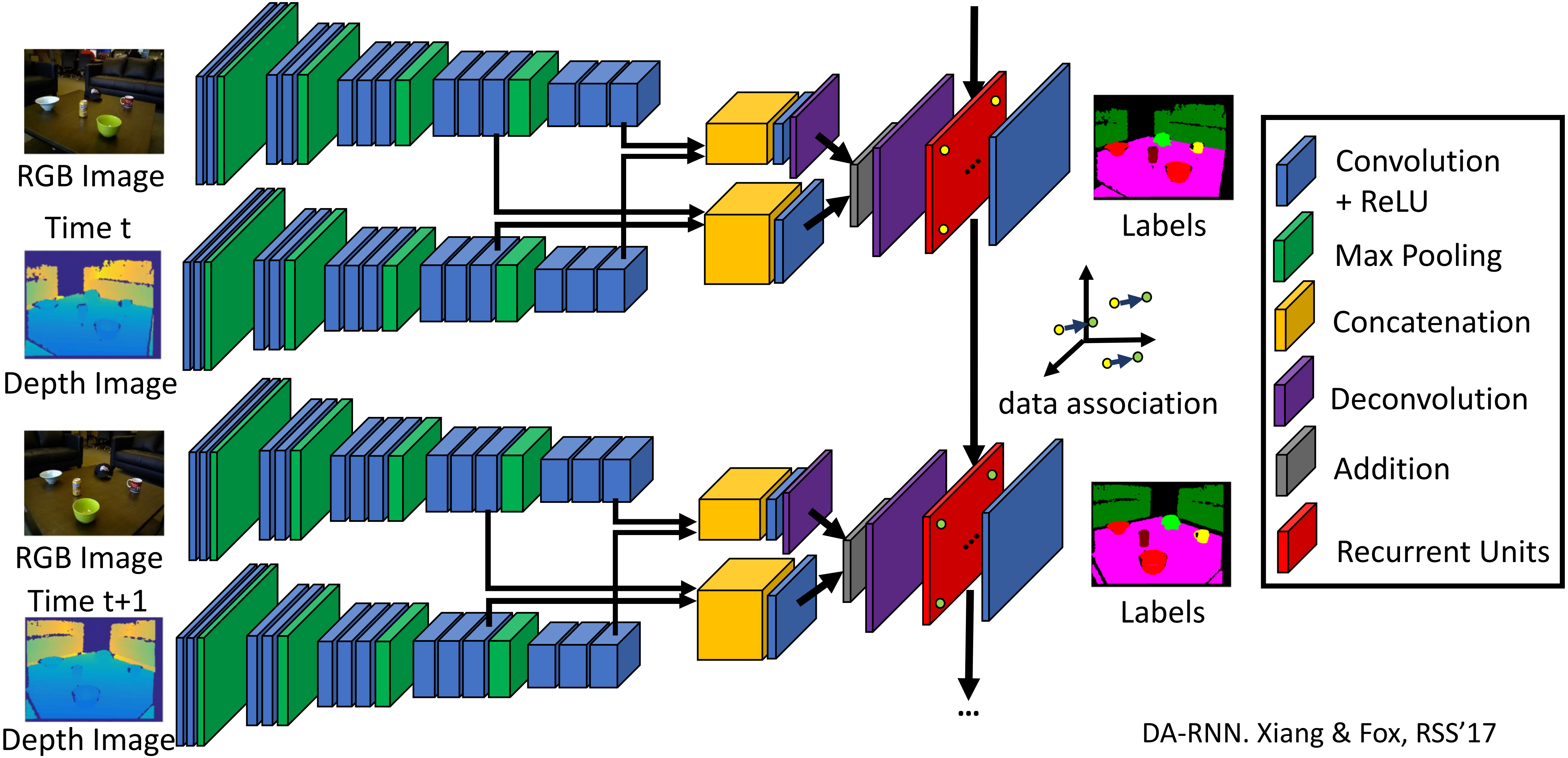
E.g., action recognition on video frames

E.g., image captioning, image -> sequences of words

E.g., action prediction, sequences of frames -> action class

E.g., Video Captioning Sequence of video frames -> caption

Recurrent Units on CNN Features



DA-RNN. Xiang & Fox, RSS'17

Machine Translation

- Translate a phrase from one language to anther
  - E.g., English phrase to French phrase

Google Translation

13 words

15 words

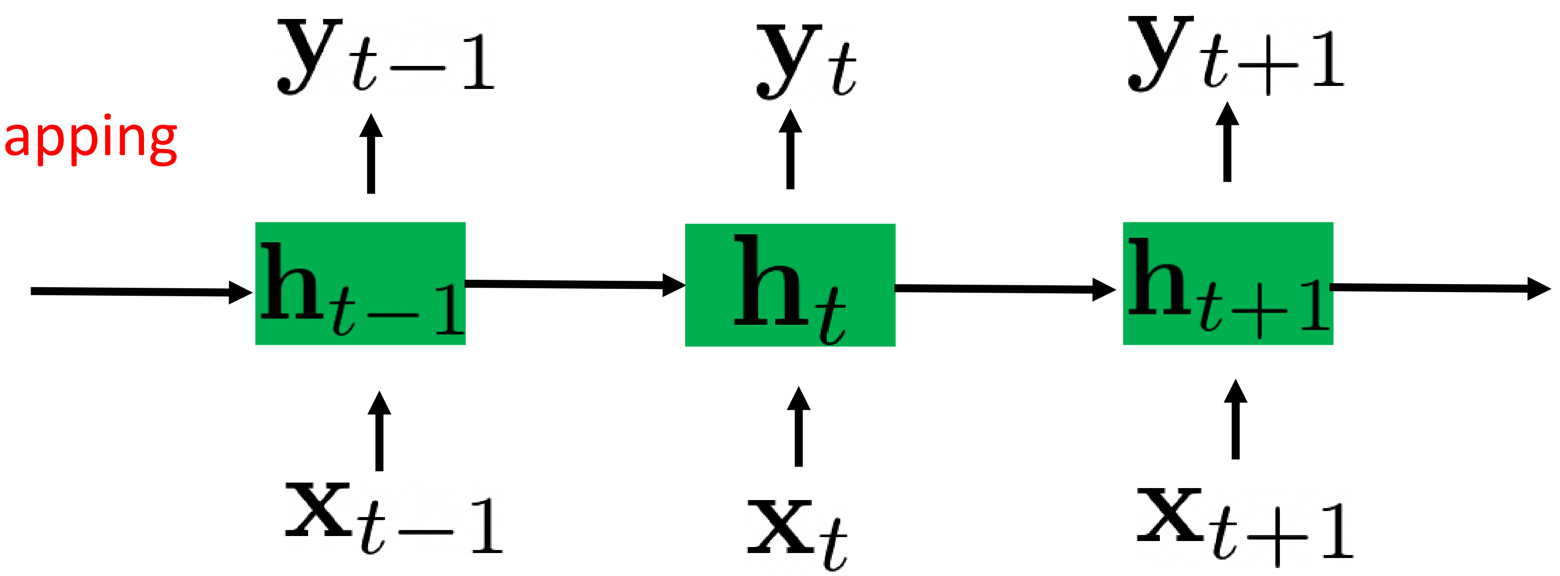
Machine Translation

- Input $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T)$
- Output $\mathbf{y} = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{T'})$

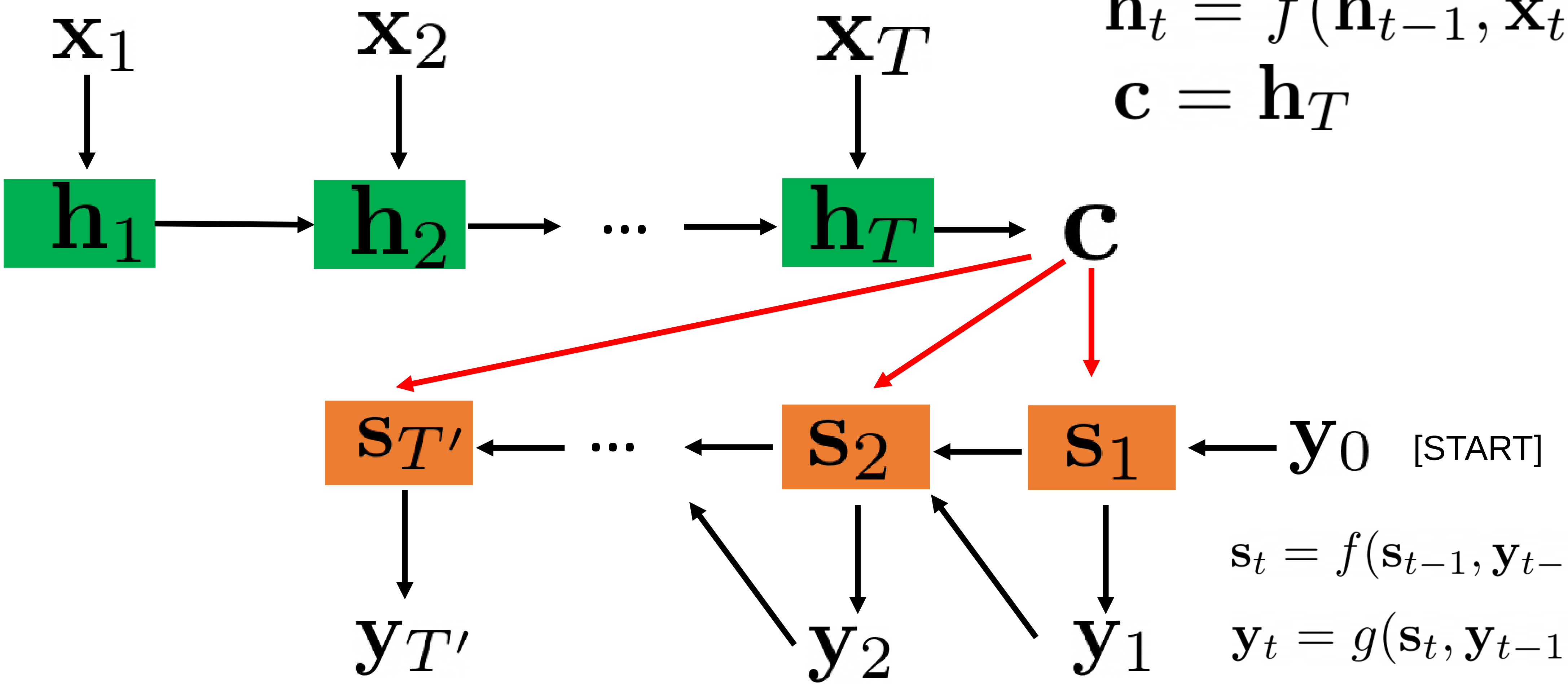
$$T \neq T'$$

Not one to one mapping

RNN



RNN Encoder-Decoder



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

$$\mathbf{c} = \mathbf{h}_T$$

$$\mathbf{s}_t = f(\mathbf{s}_{t-1}, \mathbf{y}_{t-1}, \mathbf{c})$$

$$\mathbf{y}_t = g(\mathbf{s}_t, \mathbf{y}_{t-1}, \mathbf{c})$$

Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. Cho et al., EMNLP’14

RNN Encoder-Decoder

- Encoder $\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t) \quad \mathbf{c} = \mathbf{h}_T$
- Decoder $\mathbf{s}_t = f(\mathbf{s}_{t-1}, \mathbf{y}_{t-1}, \mathbf{c}) \quad \mathbf{y}_t = g(\mathbf{s}_t, \mathbf{y}_{t-1}, \mathbf{c})$

- Pros
  - Can deal with different input size and output size

- Cons
  - The fixed length embedding $\mathbf{c}$ cannot handle long sentence well (long-distance dependencies)

Limitations of RNNs

- The sequential computation of hidden states precludes parallelization within training examples

$$\longrightarrow \mathbf{h}_{t-1} \longrightarrow \mathbf{h}_t \longrightarrow \mathbf{h}_{t+1} \longrightarrow$$

- Cannot handle long sequences well
  - Truncated back-propagation due to memory limits
  - Difficult to capture dependencies in long distances

Summary

- RNNs can be used for sequential data to capture dependencies in time
- LSTMs and GRUs are better then vanilla RNNs
- It is difficult to capture long-term dependencies in RNNs
- Use transformers (in future lectures)

Further Reading

- Stanford CS231n, lecture 10, Recurrent Neural Networks http://cs231n.stanford.edu/
- Long Short Term Memory https://www.researchgate.net/publication/13853244_Long_Short-term_Memory
- Gated Recurrent Units https://arxiv.org/pdf/1412.3555.pdf