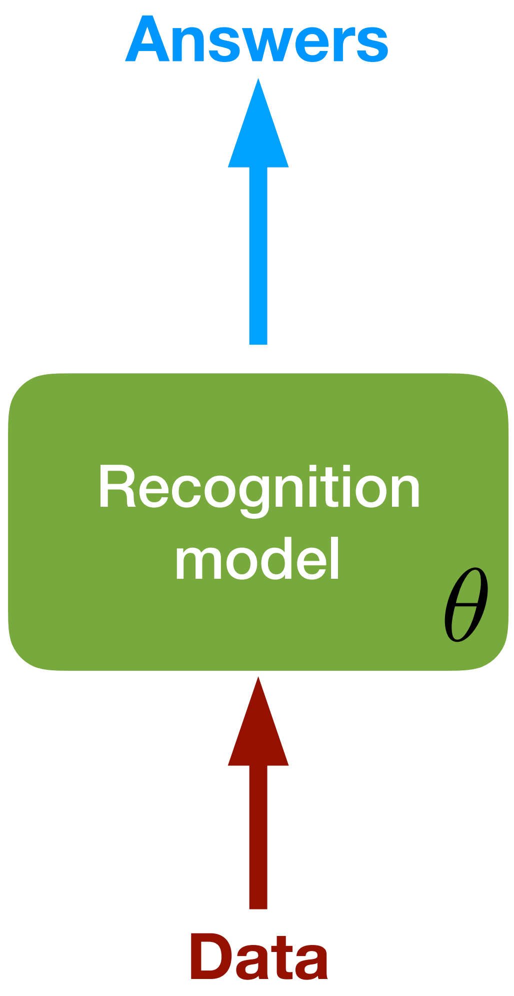
DS-GA 3001.008 Modelling time series data

L8. RNNs - continued

Instructor: Cristina Savin

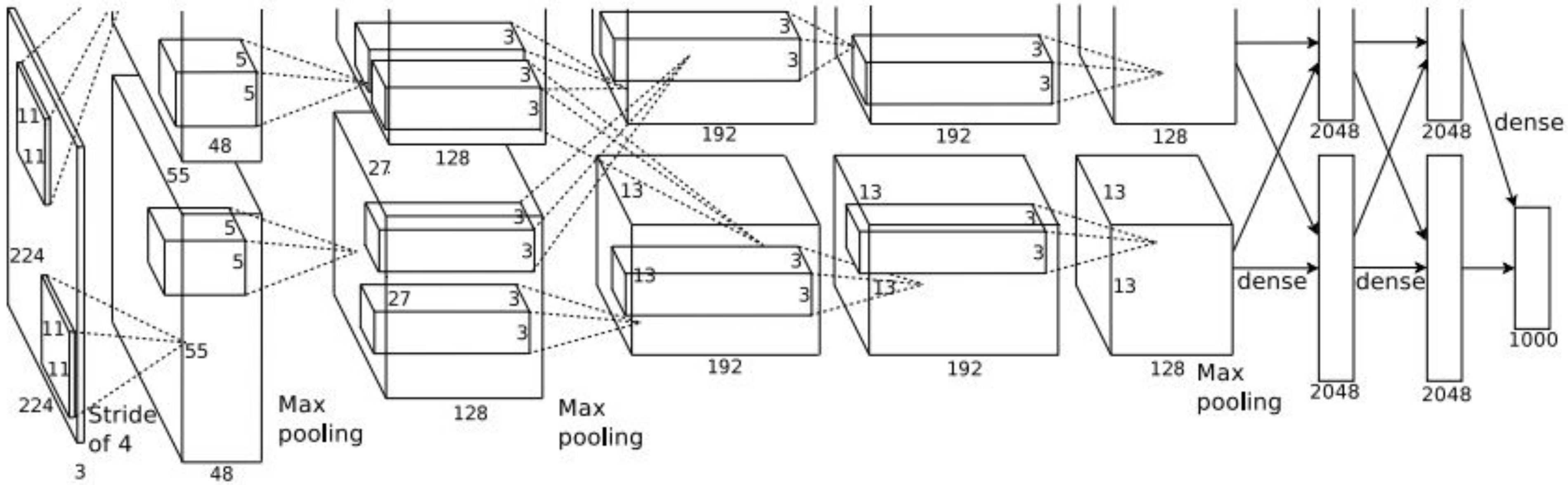
NYU, CNS & CDS

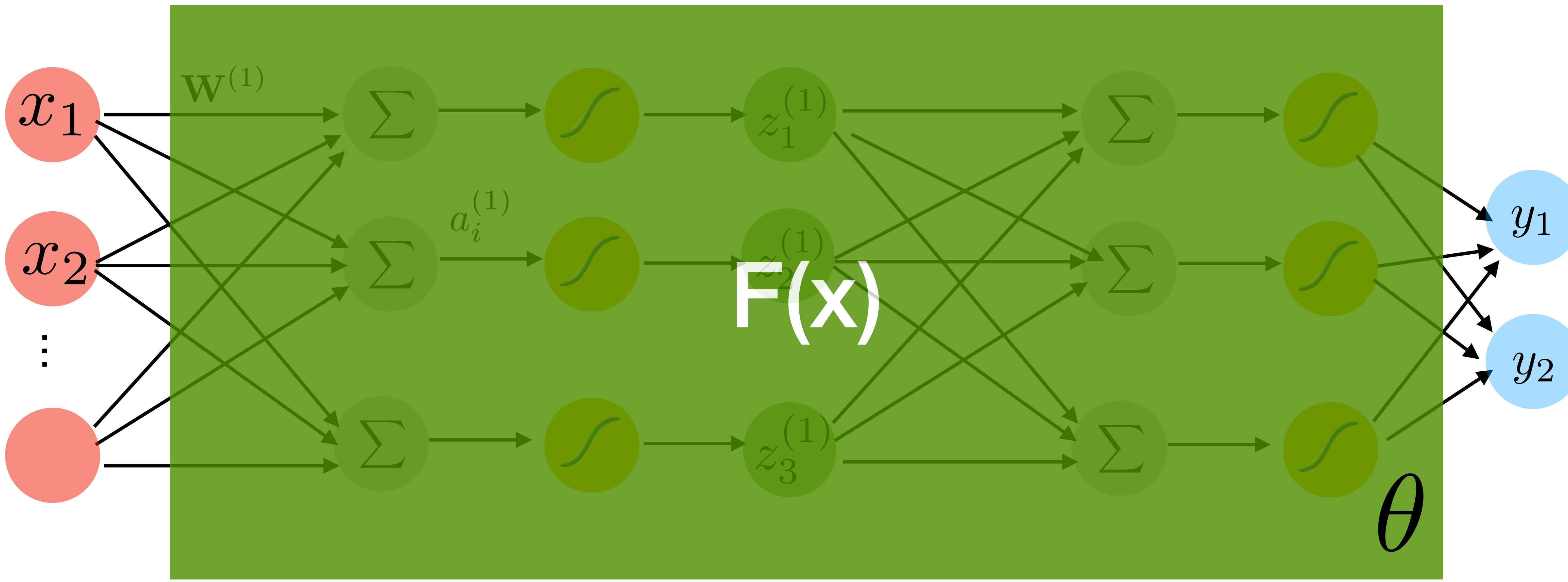
The alternative: just build a recognition model from the get go



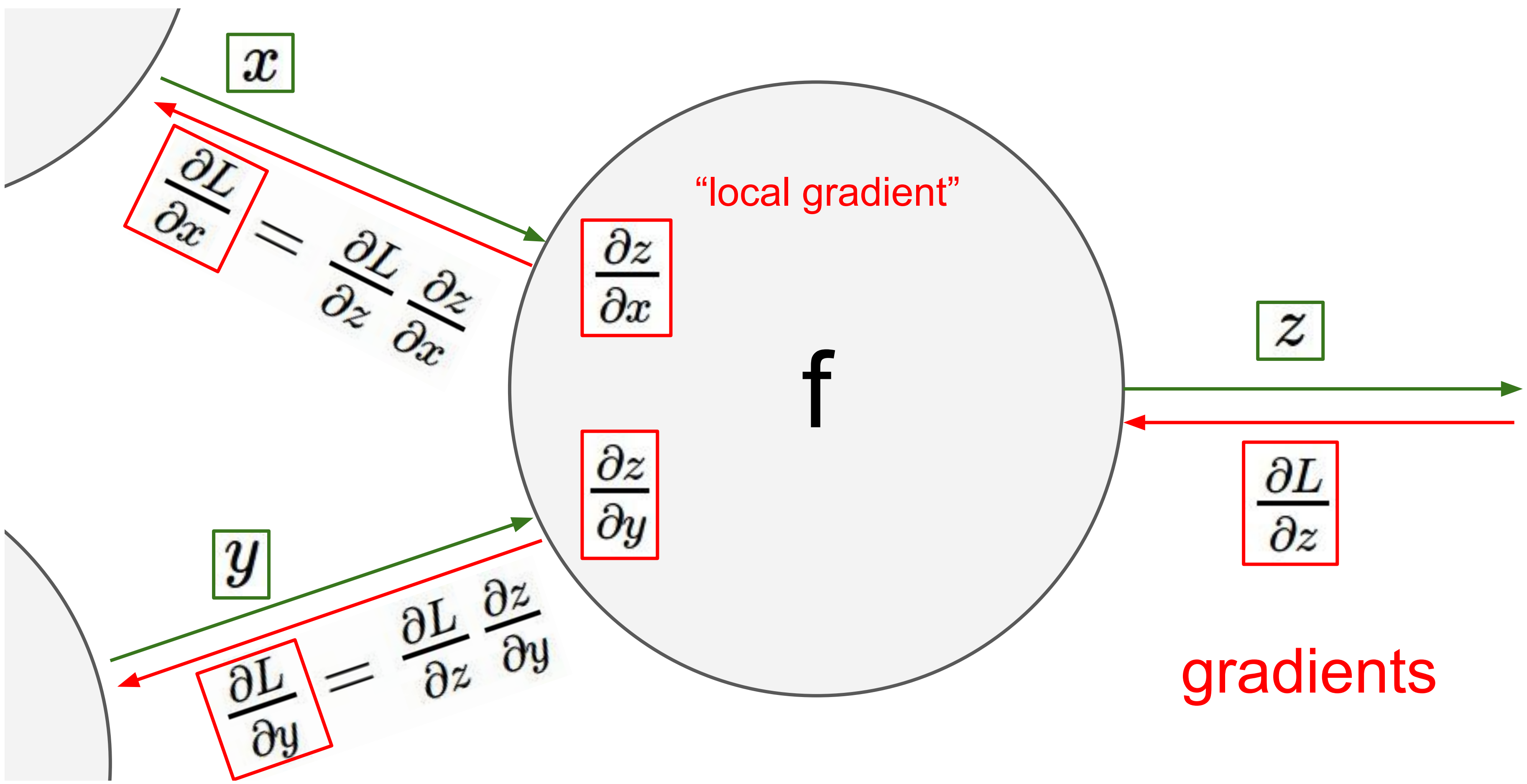
Neural networks: flexible input-out map

Backpropagation: how to train such models





A cascade of linear-nonlinear steps-> a differentiable map



http://cs231n.github.io/optimization-2/

Vanilla RNNs math

We can process a sequence of vectors **x** by
applying a **recurrence formula** at every time step:

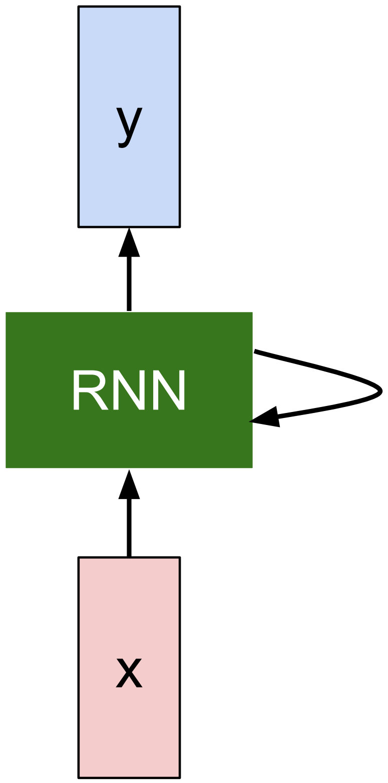
$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

new state

old state

input vector at
some time step

some function
with parameters W



$$h_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t)$$

$$y_t = W_{hy} h_t$$

RNN architectures for time series

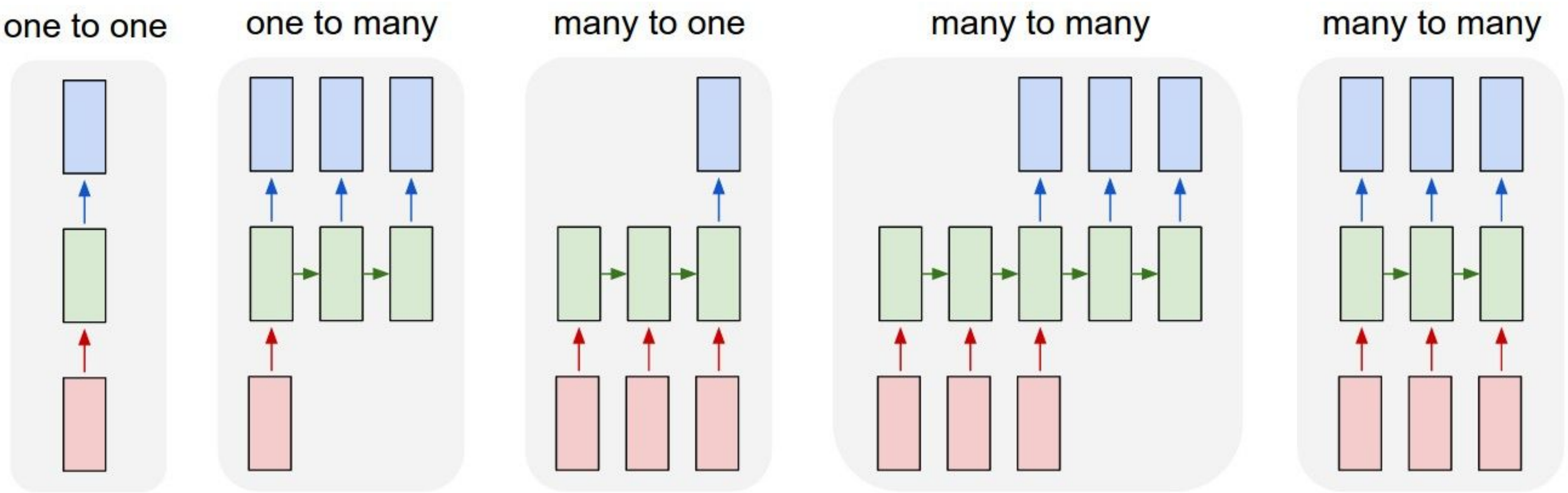


image-> caption

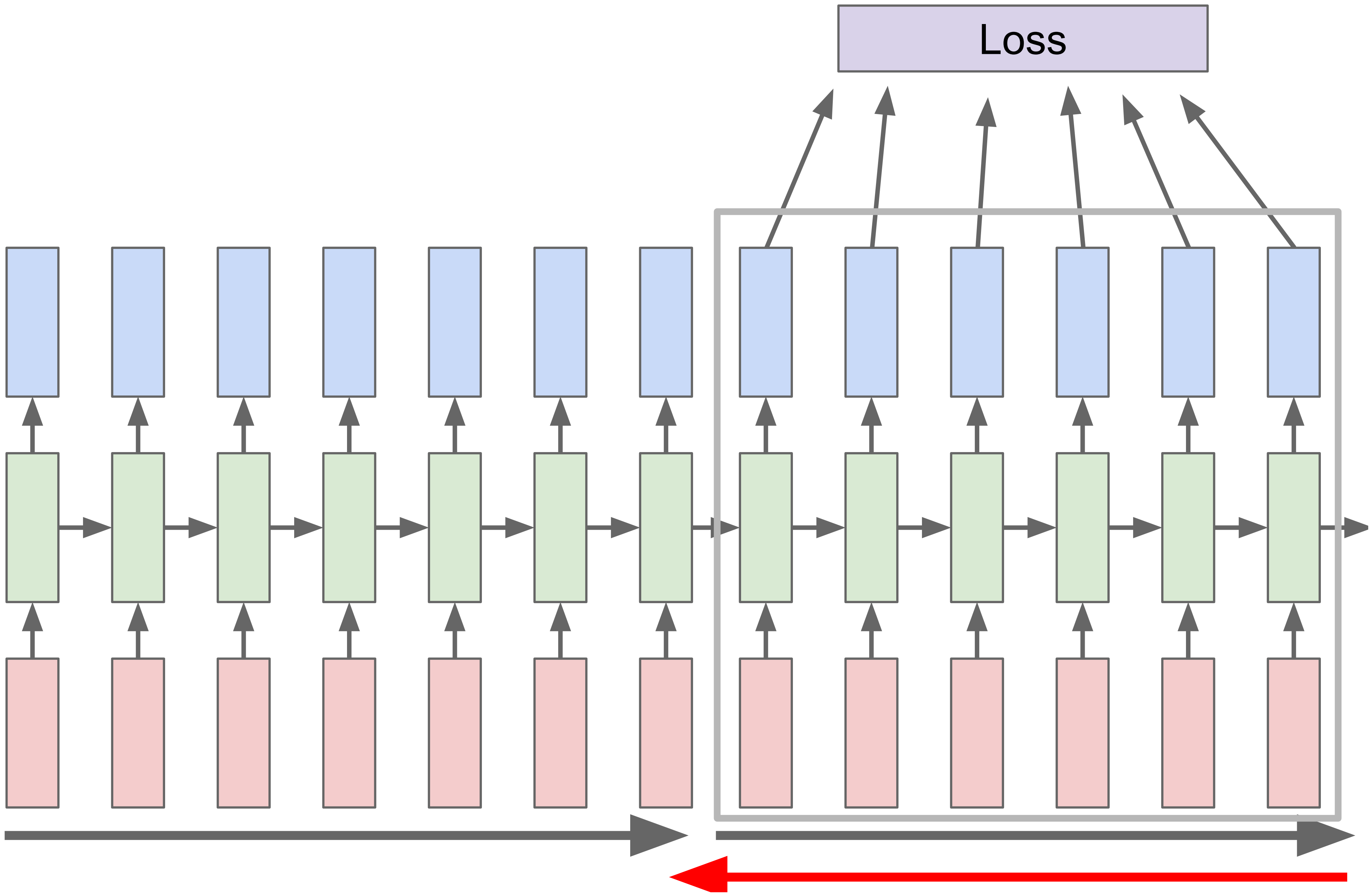
video-> label

sentence-> sentence

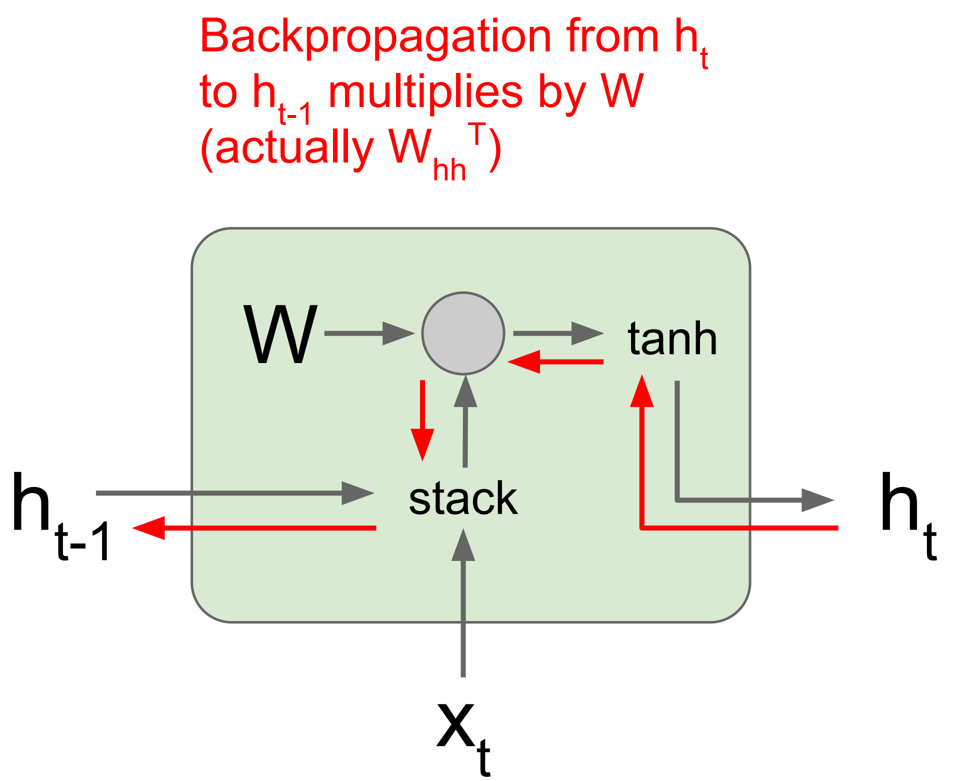
video-> video

text-> text

Truncated backprop through time



Life is harder in the RNN world



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

K steps back: W^k

Exponential decay or explosion of gradients

If gradients are exploding: clip to a max value

Fancy regularization: unitary matrices

Smarter architectures!

Vanilla RNN

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

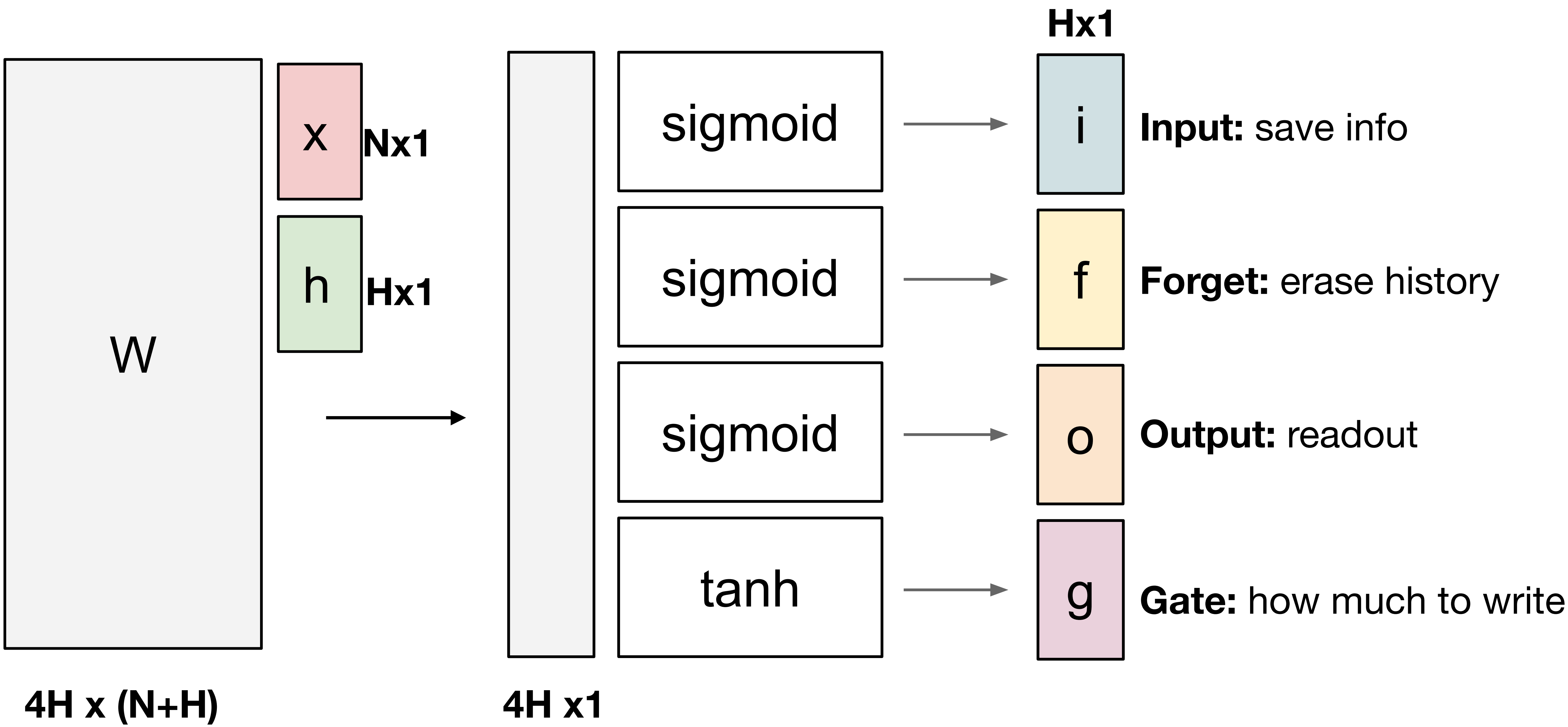
LSTM

$$\boxed{\begin{gathered} \begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \\ c_t = f \odot c_{t-1} + i \odot g \\ h_t = o \odot \tanh(c_t) \end{gathered}}$$

Hochschreiter & Schmidhuber, 1997

$\odot$ Element-wise multiplication

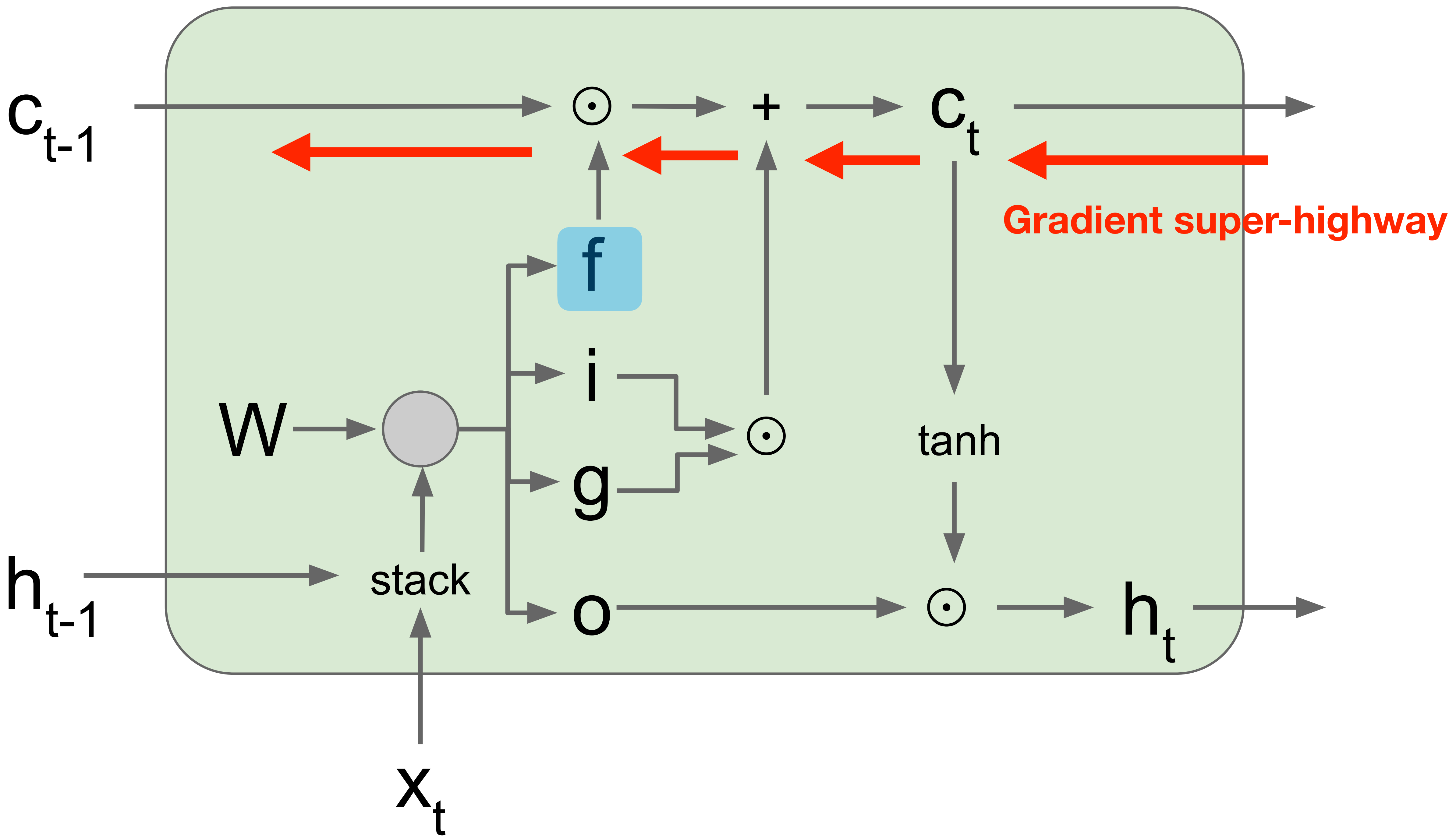
LSTM details



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

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

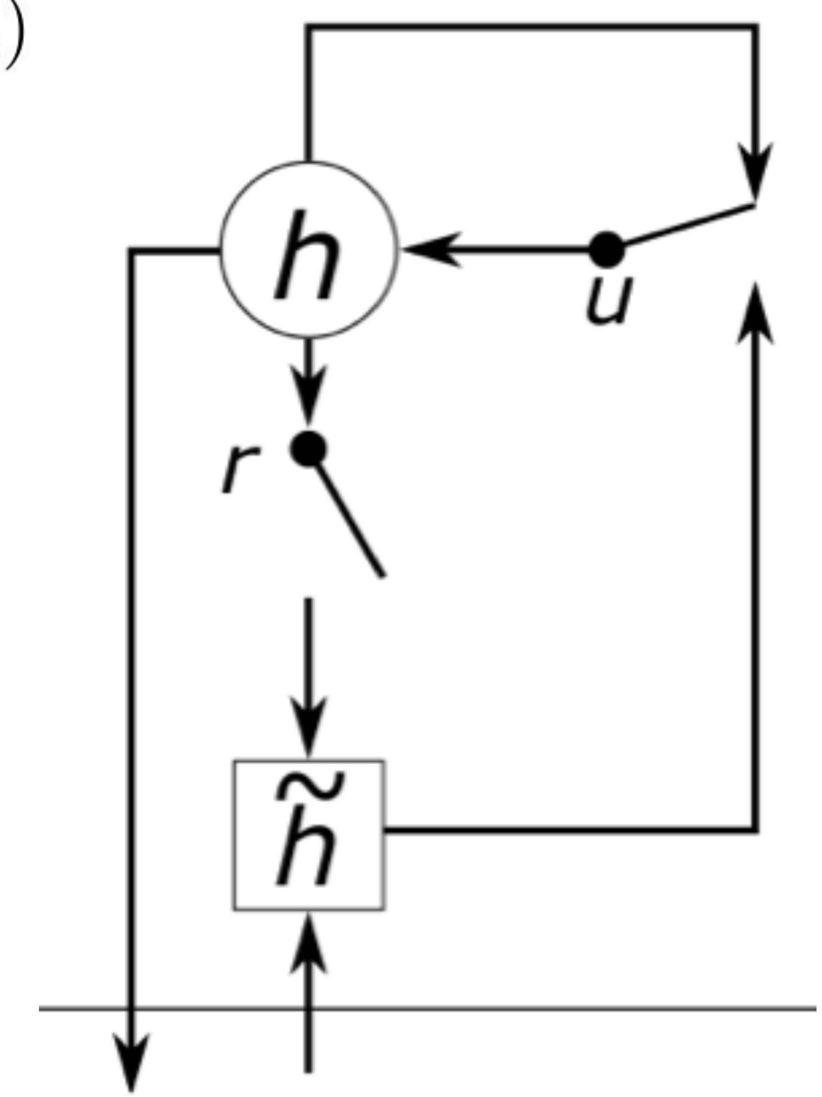
Gradient for c multiplied (element wise) with f, which varies over time and bounded so much more numerically stable



Other architectures

GRU (Cho et al, 2013)

$$r_t = \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r)$$
$$z_t = \sigma(W_{xz}x_t + W_{hz}h_{t-1} + b_z)$$
$$\tilde{h}_t = \tanh(W_{xh}x_t + W_{hh}(r_t \odot h_{t-1}) + b_h)$$
$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t$$



Other architectures

MUT1:

$$
\begin{aligned}
z &= \mathrm{sigm}(W_{\mathrm{xz}} x_t + b_{\mathrm{z}}) \\
r &= \mathrm{sigm}(W_{\mathrm{xr}} x_t + W_{\mathrm{hr}} h_t + b_{\mathrm{r}}) \\
h_{t+1} &= \tanh(W_{\mathrm{hh}}(r \odot h_t) + \tanh(x_t) + b_{\mathrm{h}}) \odot z \\
&+ \; h_t \odot (1 - z)
\end{aligned}
$$

MUT2:

$$
\begin{aligned}
z &= \mathrm{sigm}(W_{\mathrm{xz}} x_t + W_{\mathrm{hz}} h_t + b_{\mathrm{z}}) \\
r &= \mathrm{sigm}(x_t + W_{\mathrm{hr}} h_t + b_{\mathrm{r}}) \\
h_{t+1} &= \tanh(W_{\mathrm{hh}}(r \odot h_t) + W_{xh} x_t + b_{\mathrm{h}}) \odot z \\
&+ \; h_t \odot (1 - z)
\end{aligned}
$$

MUT3:

$$
\begin{aligned}
z &= \mathrm{sigm}(W_{\mathrm{xz}} x_t + W_{\mathrm{hz}} \tanh(h_t) + b_{\mathrm{z}}) \\
r &= \mathrm{sigm}(W_{\mathrm{xr}} x_t + W_{\mathrm{hr}} h_t + b_{\mathrm{r}}) \\
h_{t+1} &= \tanh(W_{\mathrm{hh}}(r \odot h_t) + W_{xh} x_t + b_{\mathrm{h}}) \odot z \\
&+ \; h_t \odot (1 - z)
\end{aligned}
$$

(Jozefowicz et al., 2015)

Link to other approaches

Deep Kalman Filter *(Krishnan et al, 2016)*

Transition distribution

$$z_t | z_{t-1} \sim \mathcal{N}(F_\theta^\mu(z_{t-1}), F_\theta^\Sigma(z_{t-1}))$$

Emission distribution

$$x_t | z_t \sim \mathcal{N}(G_\theta^\mu(z_t), G_\theta^\Sigma(z_t))$$

Approximate posterior distribution

$$z_t | x_{1:T} \sim \mathcal{N}(R_\theta^\mu(x_{1:T}), R_\theta^\Sigma(x_{1:T}))$$

Feedforward neural networks

Bidirectional recurrent networks

$$\mathcal{F}(q, \theta) = \int q_\theta(z) \log p_\theta(x|z) \mathrm{d}z + \int q_\theta(z) \log \frac{p_\theta(z)}{q_\theta(z)} \mathrm{d}z$$

Directly maximize the free energy using GD and backpropagation

Summary

RNNs provide a **flexible** way for modeling temporal dependencies at scale.

Neural networks (deep learning) enable us to handle
- High-dimensional observations (often 100s-1000s)
- High-dimensional latent variables (often 100s-1000s)
- Large-scale data (often millions or more)

- Neural networks (deep learning) enable us to capture
  - Nonlinear dynamics
  - Long-term dependencies

- Neural networks (deep learning) enable us to work with
  - Intractable probabilistic models

- Neural networks enable us to transfer knowledge across problems
  - Parameter sharing across multiple datasets
  - One neural network with a task indicator input

Summary

Data hungry, and not necessarily easy to train

LSTMs and GRUs used in practice

Good at capturing long-term dependencies but not high frequency structure

Ongoing research: better architectures, theoretical understanding.

Greff et al. *LSTM: A Search Space Odyssey,* 2015.

Goodfellow, Courville, Bengio. *Deep Learning,* 2016.

Goldberg. *Neural Network Methods for Natural Language Processing,* 2017.

K Cho lecture notes: https://github.com/nyu-dl/NLP_DL_Lecture_Note