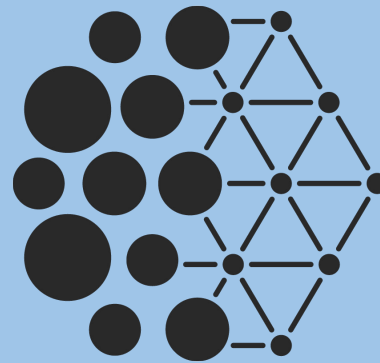


Institut
des algorithmes
d'apprentissage
de Montréal



Mila

Recurrent Neural Networks

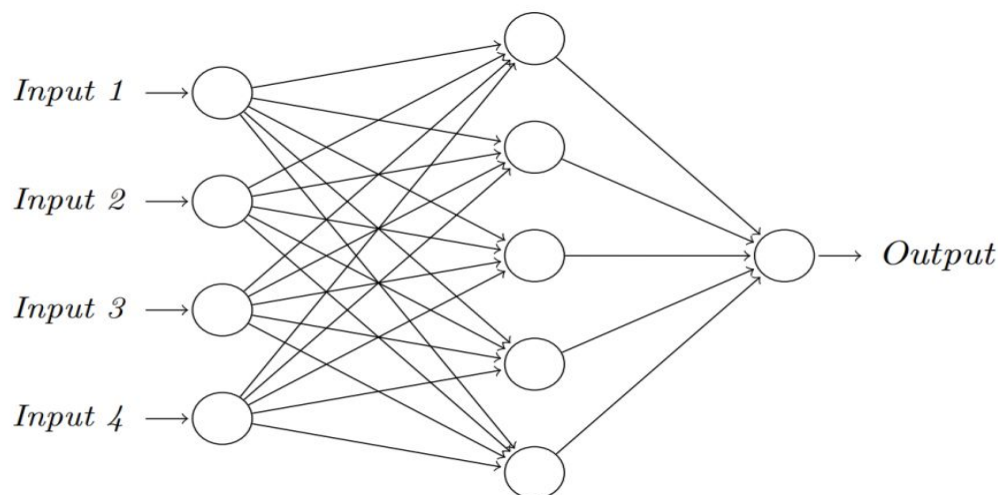
Nan Rosemary Ke

Plan

- Motivation
- Introduction to recurrent neural networks (RNNs)
- Training RNNs
- Difficulties for training
- RNN architectures
- Teacher forcing
- Different variations of RNNs

Motivation

- Seen how to train model with fixed size data

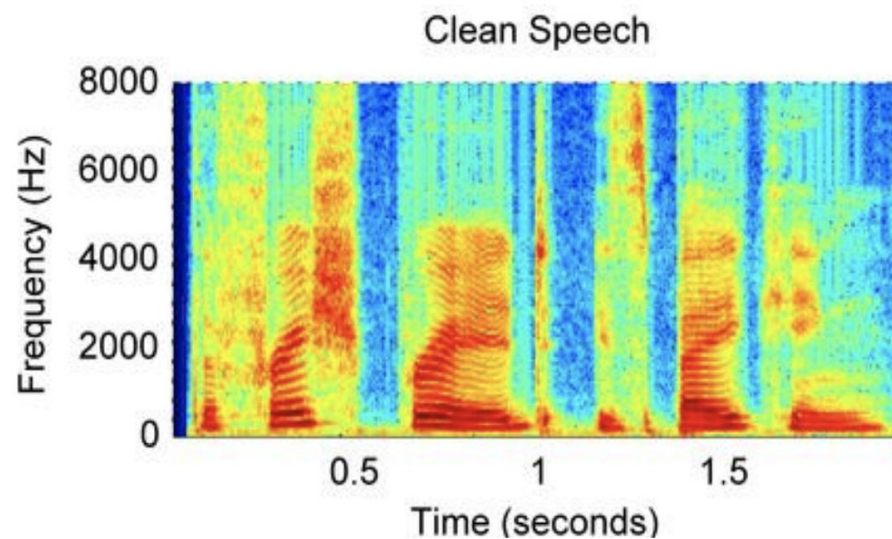
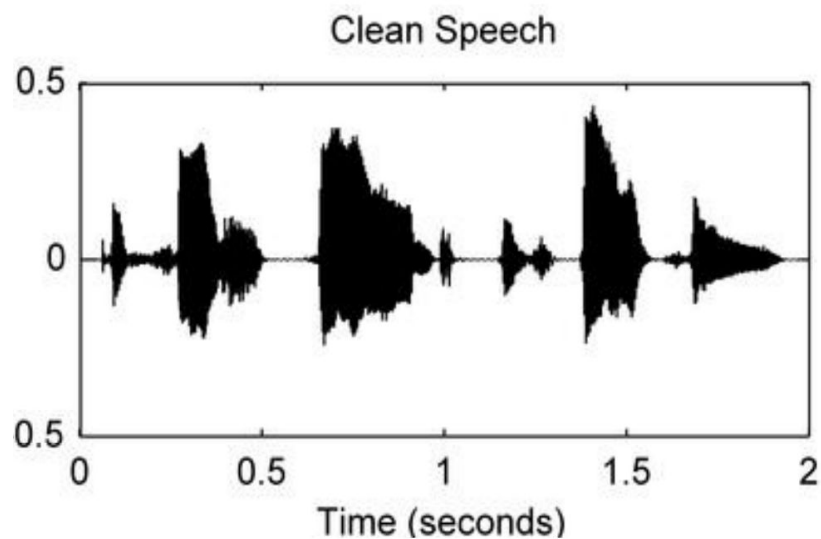


- What about sequential data?

Images from Cedar Laurent's slides

Sequential data

- Speech recognition
 - Audio to text



Images from Cedar Laurent's slides

Sequential data

- Machine translation
 - Text to text

Traduction

Français

Anglais

Arabe

Détecter la langue

↔

J'aime les réseaux de neurones performants.

43/5000

Désactiver la traduction instantanée

★

Anglais

Français

Arabe

Traduire

I like high-performance neural networks.

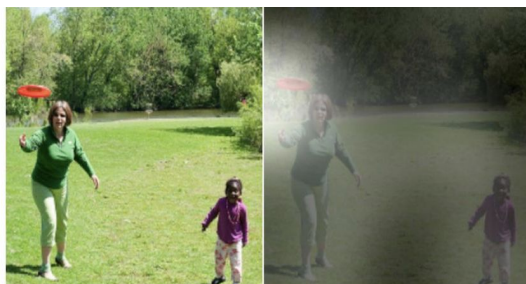
☆ 📄 🔊 🔗

✎ Suggérer une modification

Images from Cedar Laurent's slides

Sequential data

- Image captioning
 - Image to text



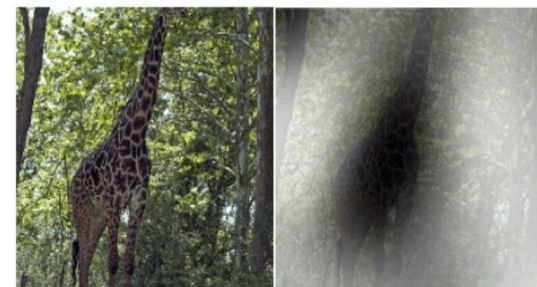
A woman is throwing a frisbee in a park.



A stop sign is on a road with a mountain in the background.



A group of people sitting on a boat in the water.



A giraffe standing in a forest with trees in the background.

Images from Cedar Laurent's slides

Sequential data

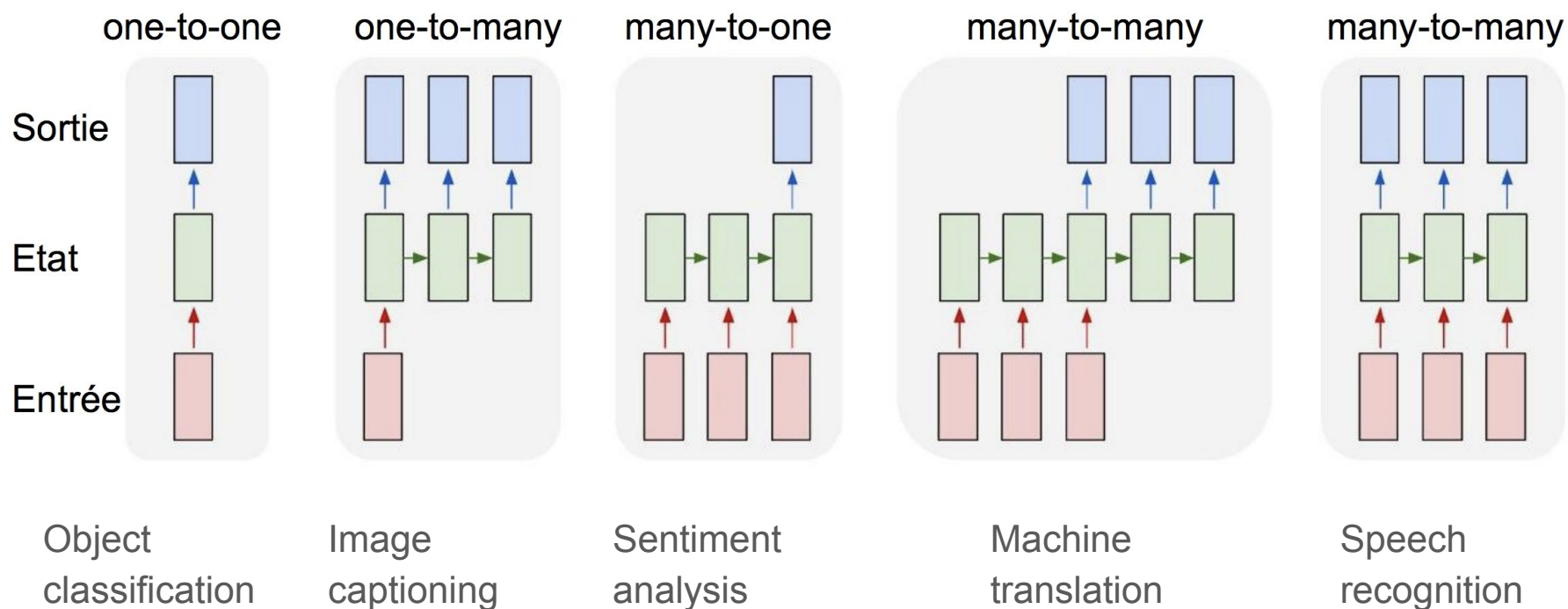
- More examples
 - Text (language modeling)
 - Video (video generation, video understanding)
 - Biological data
 - Medical imaging
 - DNA sequences

Recurrent Neural Networks

- For handling sequential data
 - Variable length input
 - Variable order
 - “I visited Paris in 2014”
 - “In 2014, I visited Paris”
- Use shared parameters across time

Sequence modeling

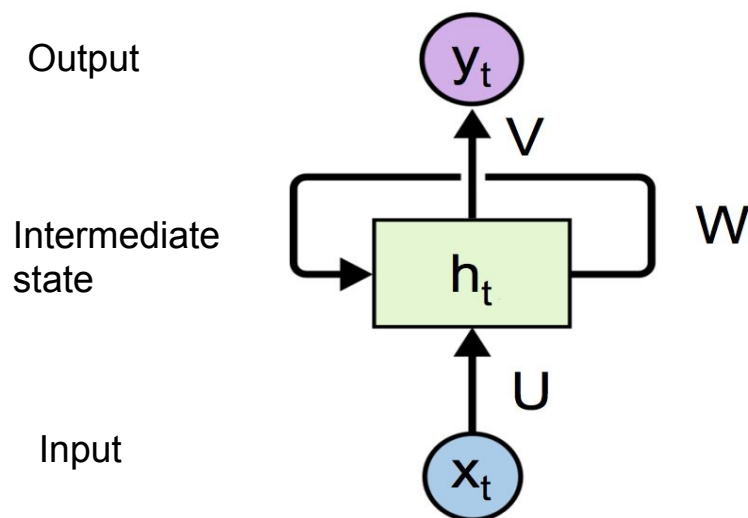
- Different applications



Images from Chris Olah's slides

Introduction to recurrent neural networks (RNNs)

- Input: x_0, x_1, \dots, x_T
- Output: $y_0, y_1, \dots, y_T,$
- Intermediate state: $h_0, h_1, \dots, h_T,$



Images from Chris Olah's slides

Recurrent neural networks

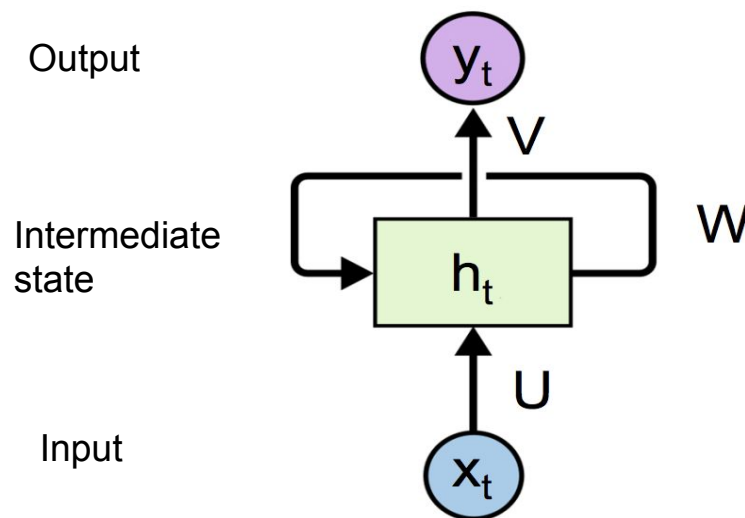
Vanilla recurrent neural networks

$$h_t = \tanh(Ux_t + Wh_{t-1})$$

$$y_t = f(Vh_t)$$

Parameters of the network

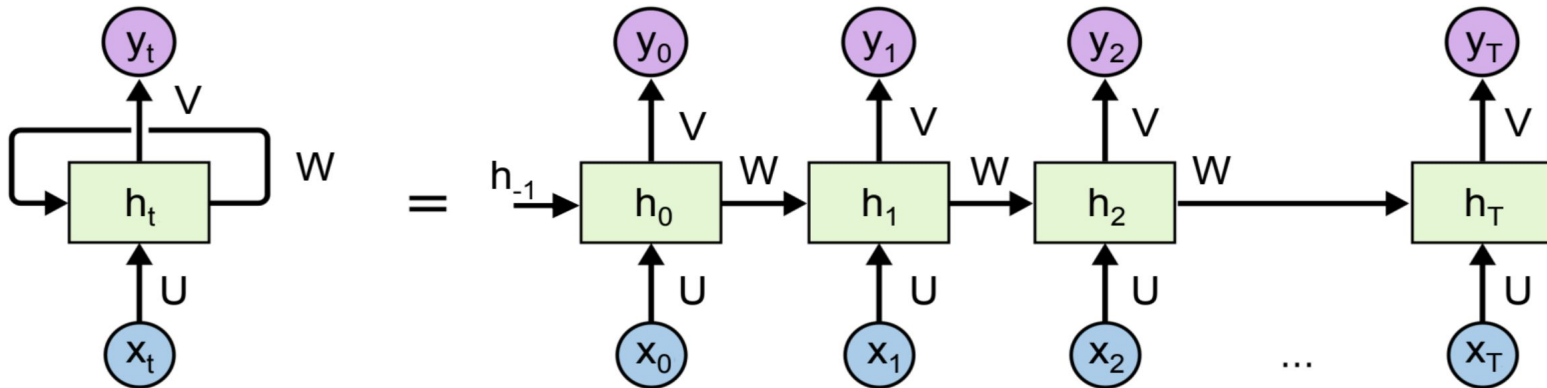
- U, W, V
- Shared across time
- W is recurrent



Images from Chris Olah's slides

Recurrent neural networks

Parameters unrolled across time



Images from Chris Olah's slides

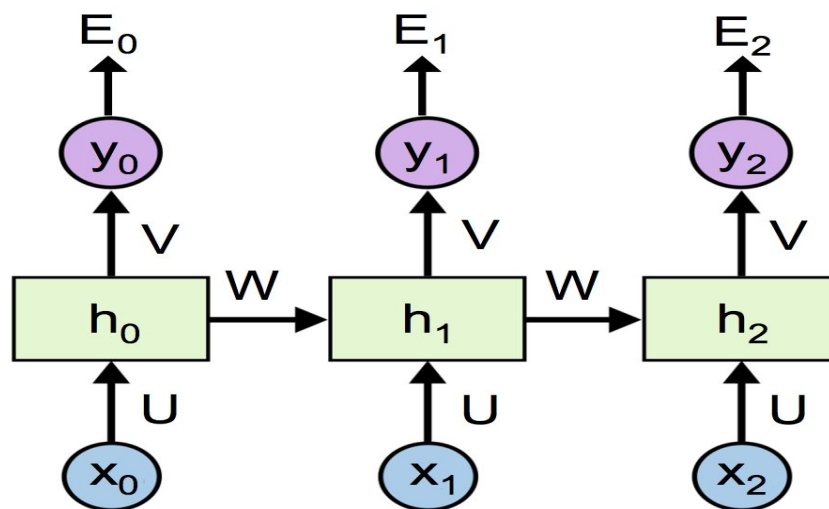
Training RNNs

- All losses

$$E = \sum_{t=0}^T E_t$$

- Normal backprop

$$\frac{\partial E}{\partial U} = \sum_{t=0}^T \frac{\partial E_t}{\partial U}$$

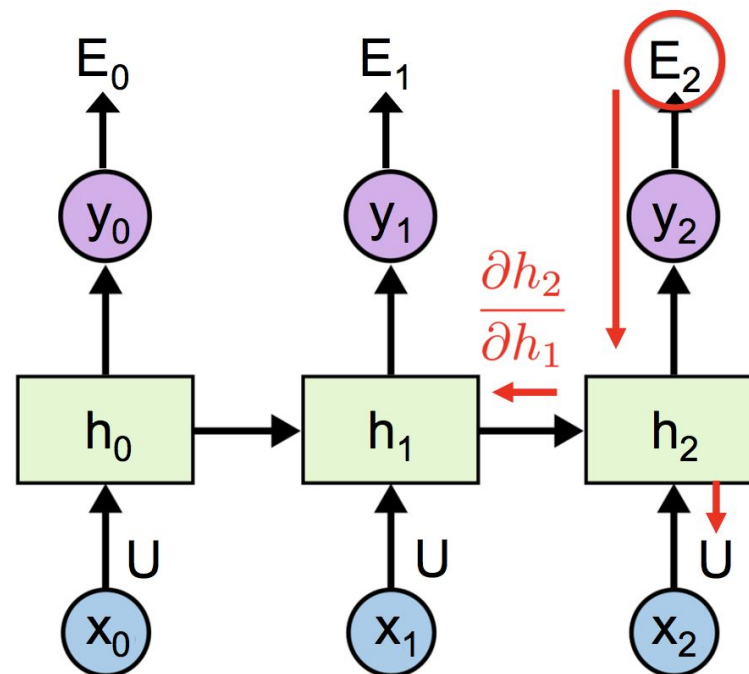


Images from Chris Olah's slides

Backpropagation through time (BPTT)

$$\frac{\partial E_2}{\partial U} = \frac{\partial E_2}{\partial h_2} \left(x_2^T + \frac{\partial h_2}{\partial h_1} (\dots \right)$$

Image from Christopher Olah's blog

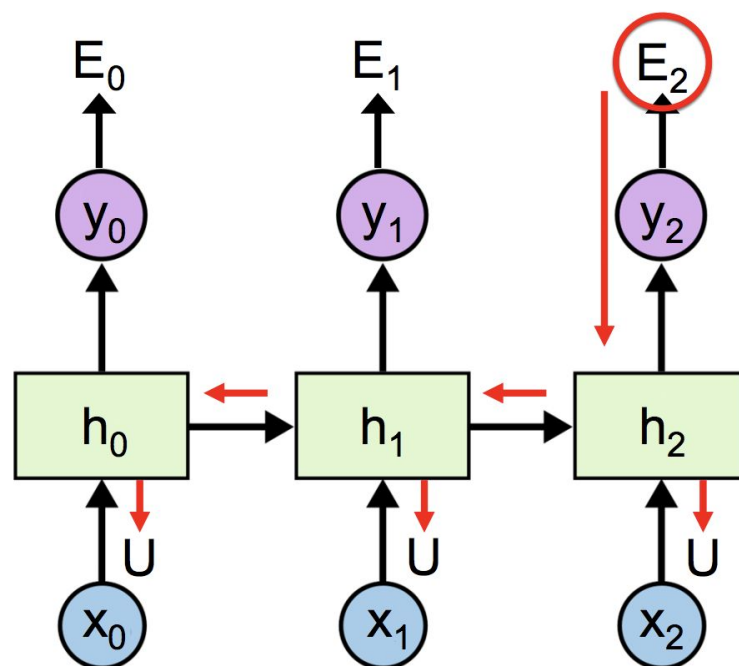


Images from Chris Olah's slides

Backpropagation through time (BPTT)

$$\frac{\partial E_2}{\partial U} = \frac{\partial E_2}{\partial h_2} \left(x_2^T + \frac{\partial h_2}{\partial h_1} \left(x_1^T + \frac{\partial h_1}{\partial h_0} x_0^T \right) \right)$$

image from Christopher Olah's blog

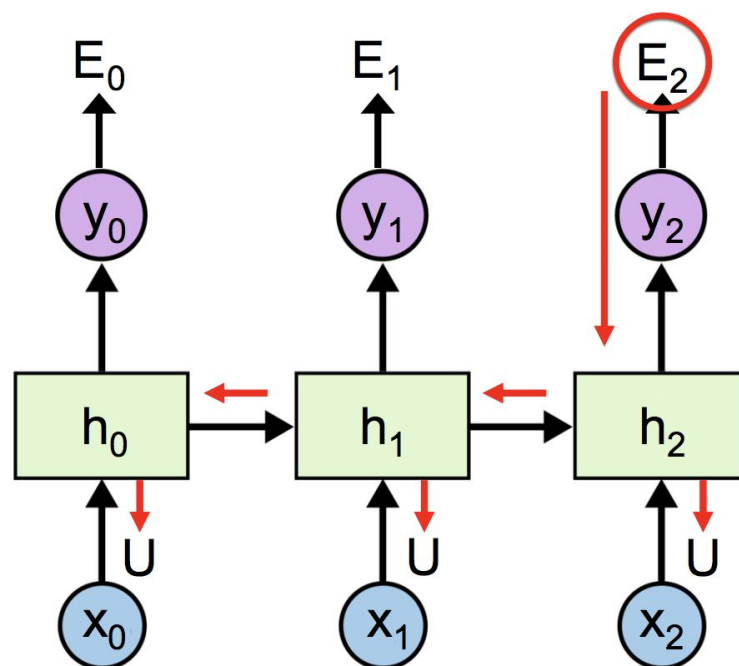


Images from Chris Olah's slides

Backpropagation through time (BPTT)

$$\frac{\partial E_2}{\partial U} = \frac{\partial E_2}{\partial h_2} \left(x_2^T + \frac{\partial h_2}{\partial h_1} \left(x_1^T + \frac{\partial h_1}{\partial h_0} x_0^T \right) \right)$$

image from Christopher Olah's blog

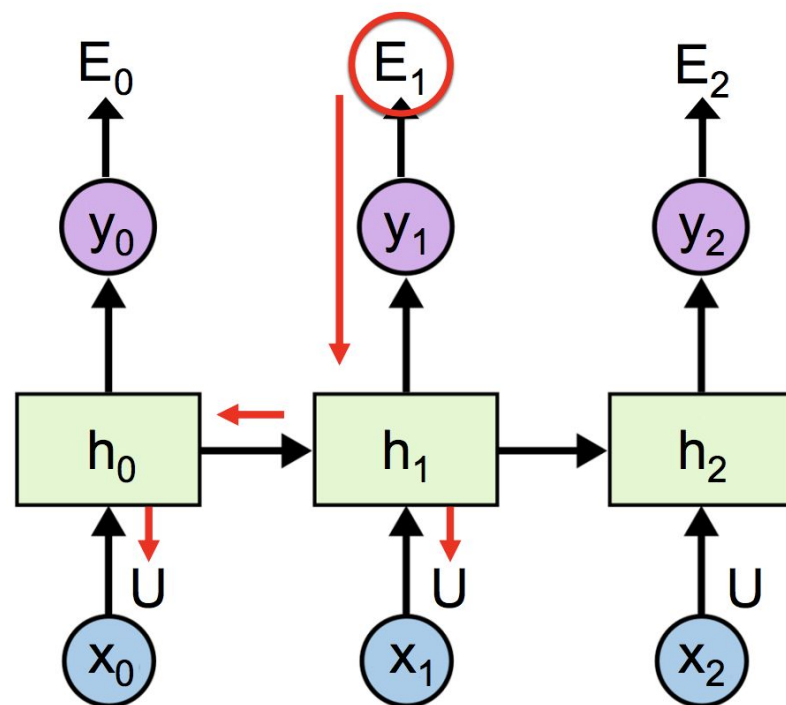


Images from Chris Olah's slides

Backpropagation through time (BPTT)

Same procedure

1. Compute dE_1/dh_1
2. Compute dh_1/dU at time 1
3. Compute dh_0/dU



Images from Chris Olah's slides

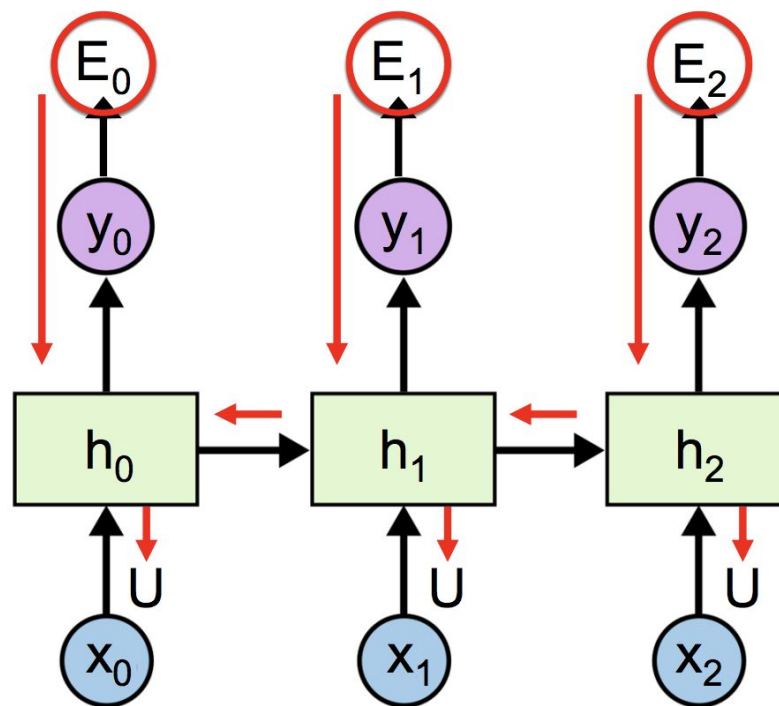
Backpropagation through time (BPTT)

All gradients are summed

$$\frac{\partial E}{\partial U} = \sum_{t=0}^T \frac{\partial E_t}{\partial U}$$

$$\frac{\partial E}{\partial W} = \sum_{t=0}^T \frac{\partial E_t}{\partial W}$$

$$\frac{\partial E}{\partial W} = \sum_{t=0}^T \frac{\partial E_t}{\partial W}$$



Images from Chris Olah's slides

RNN Training

- Trained to predict future from the past
 - h_t is a lossy summary of past
 - Depending on training criteria, h_t decides to keep certain information
 - Difficulty: if y_t depends on the distant past, so h_t has to keep information from many timesteps ago.

Long term dependencies

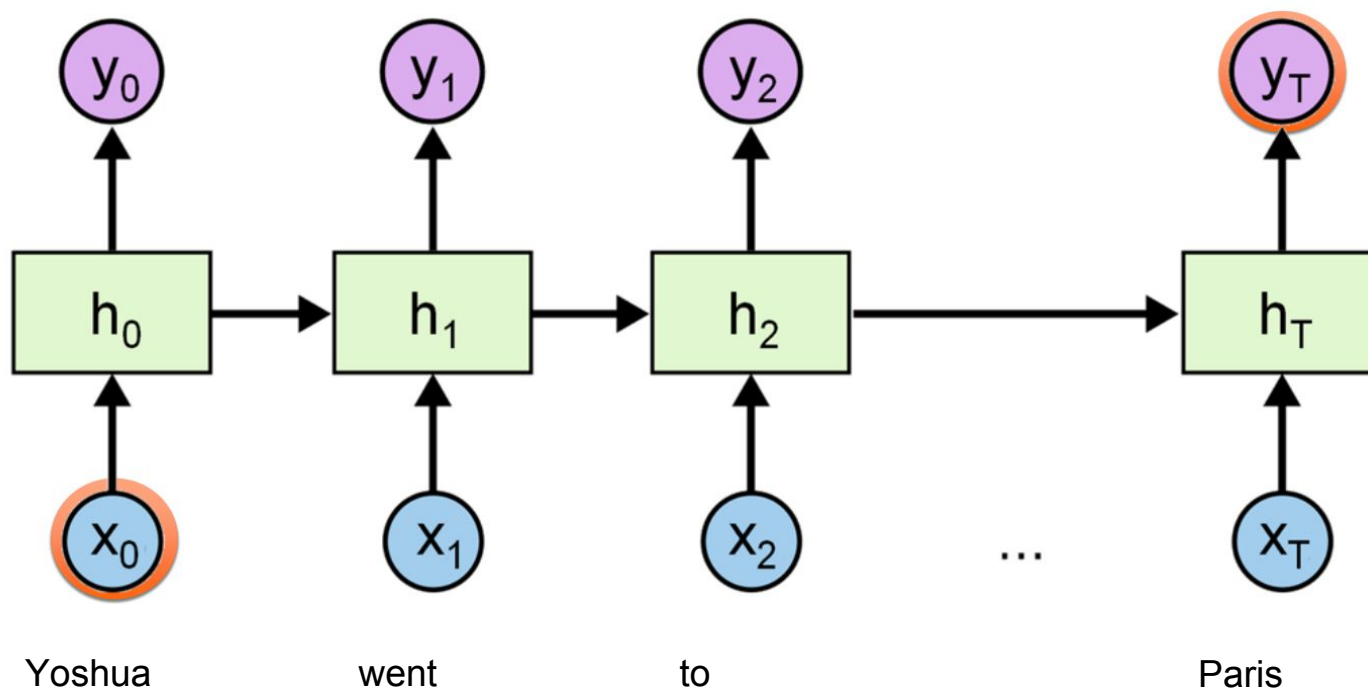
Challenges in RNN training

- Parameters are shared across time
 - # of parameters do not change with longer sequences
 - Consequence:
 - Optimization issue
 - Assumption: same param can be used for different time. Conditional probability distribution over variables are same at $t+1$ compared to t
 - Exploding/ vanishing gradients

Challenges in RNN training

Long term dependencies

Who went to Paris?

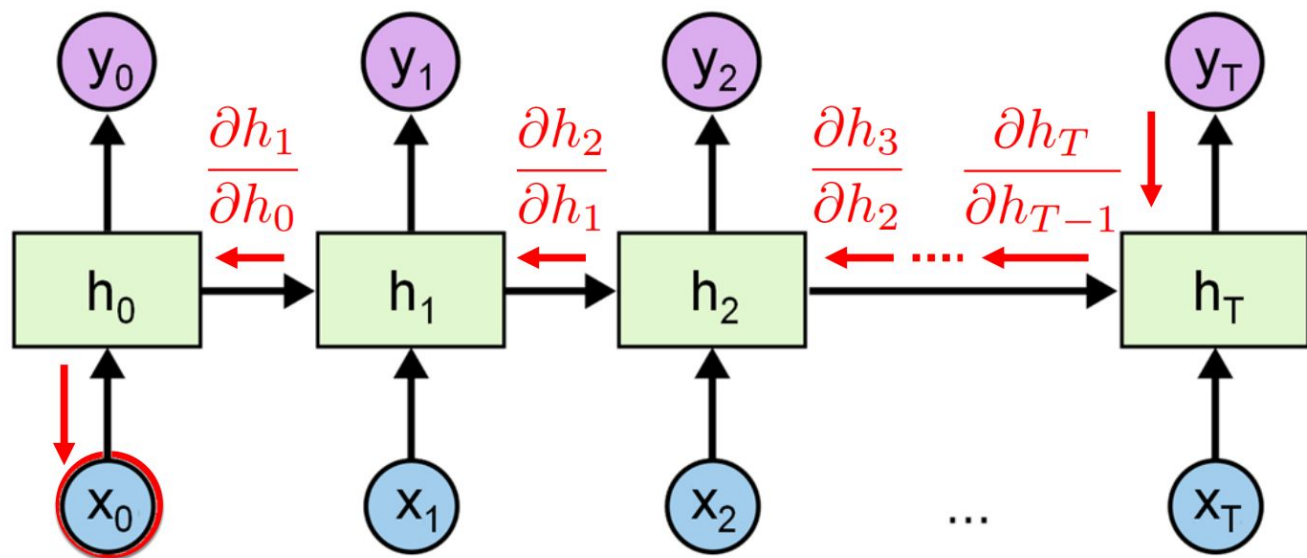


Images from Chris Olah's slides

Challenges in RNN training

Long term dependencies

- Gradients going far back in time



Images from Chris Olah's slides

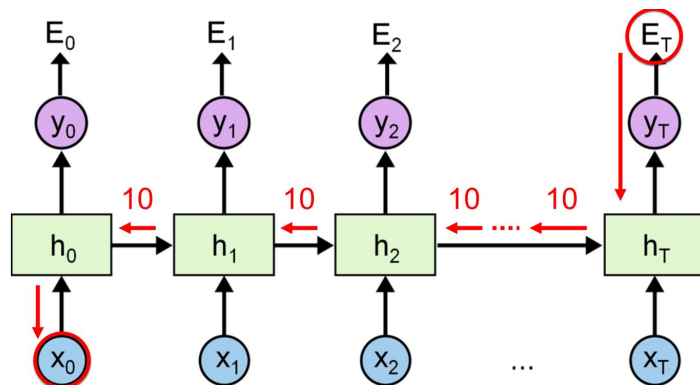
Challenges in RNN training

Long term dependencies

- Gradients going far back in time

$$\frac{\partial y_T}{\partial x_0} = \frac{\partial y_T}{\partial h_T} \frac{\partial h_T}{\partial h_{T-1}} \dots \frac{\partial h_3}{\partial h_2} \frac{\partial h_2}{\partial h_1} \frac{\partial h_1}{\partial h_0} \frac{\partial h_0}{\partial x_0}$$

- Gradients can become unstable



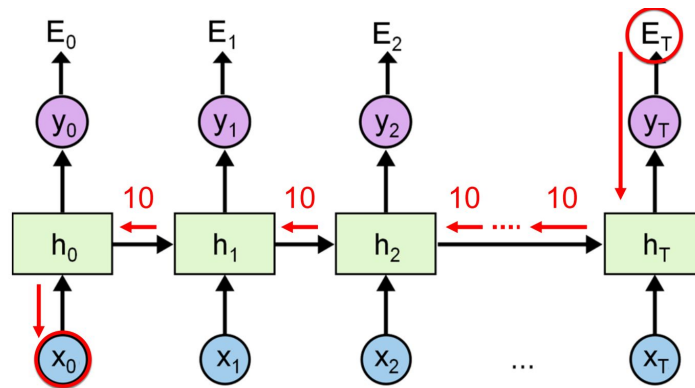
Images from Chris Olah's slides

Exploding gradients

- Gradients going far back in time

$$\frac{\partial y_T}{\partial x_0} = \frac{\partial y_T}{\partial h_T} \frac{\partial h_T}{\partial h_{T-1}} \dots \frac{\partial h_3}{\partial h_2} \frac{\partial h_2}{\partial h_1} \frac{\partial h_1}{\partial h_0} \frac{\partial h_0}{\partial x_0}$$

- Gradients magnified at each time \rightarrow exploding gradient!



Images from Chris Olah's slides

Exploding gradients

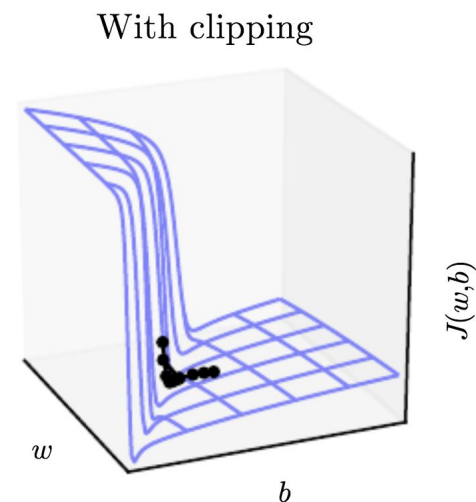
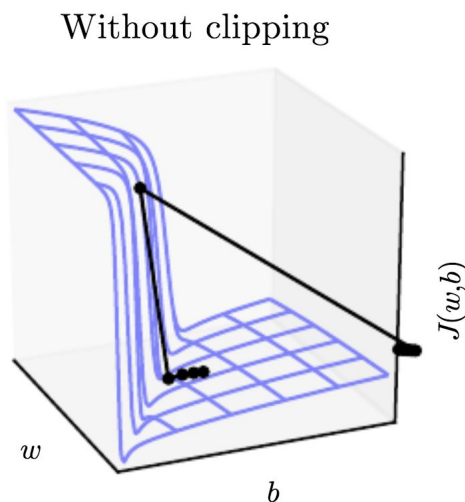
Solution:

- Gradient clipping

$$g = \frac{\partial E}{\partial W}$$

If $\|g\| \geq threshold$

then $g = \frac{threshold}{\|g\|} g$



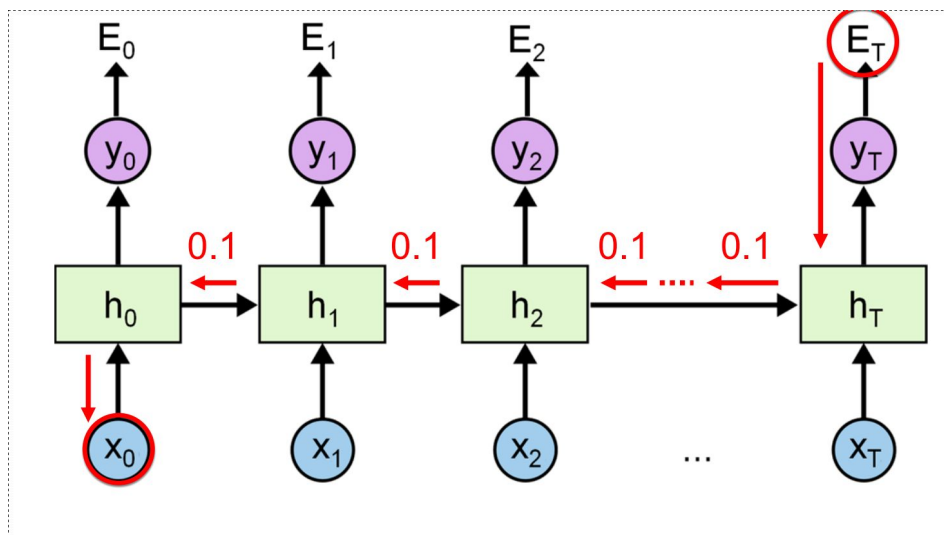
Images from the deep learning book

Vanishing gradients

- Gradients going far back in time

$$\frac{\partial y_T}{\partial x_0} = \frac{\partial y_T}{\partial h_T} \frac{\partial h_T}{\partial h_{T-1}} \dots \frac{\partial h_3}{\partial h_2} \frac{\partial h_2}{\partial h_1} \frac{\partial h_1}{\partial h_0} \frac{\partial h_0}{\partial x_0}$$

- Gradients contracts at each time \rightarrow vanishing gradient!



Vanishing gradients

- Gradients going far back in time

$$\frac{\partial y_T}{\partial x_0} = \frac{\partial y_T}{\partial h_T} \frac{\partial h_T}{\partial h_{T-1}} \dots \frac{\partial h_3}{\partial h_2} \frac{\partial h_2}{\partial h_1} \frac{\partial h_1}{\partial h_0} \frac{\partial h_0}{\partial x_0}$$

- Parameters are stable
 - No explosion
- Can not learn long-term dependencies
 - No easy solution
 - Make architecture changes

Solutions to long-term dependencies

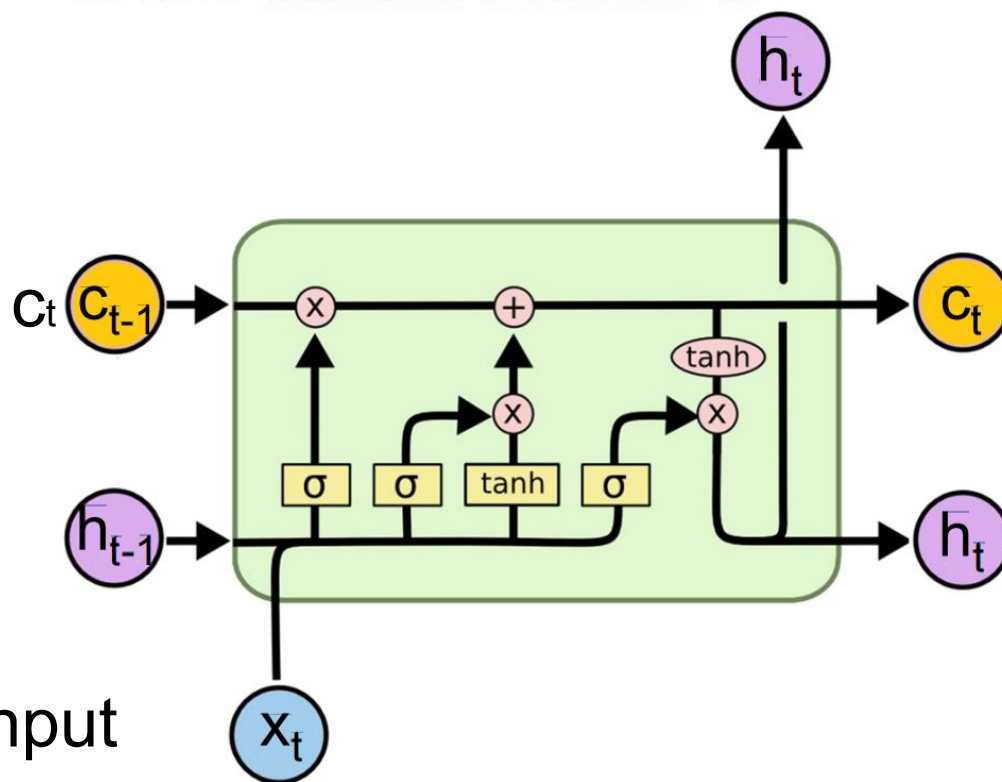
- Gated recurrent neural networks
 - Self-loop for gradients to flow for many steps
 - Model can learn what to forget
 - Long-short term memory (LSTM)
 - Gated recurrent neural networks (GRU)

LSTM

Memory cell

Internal state

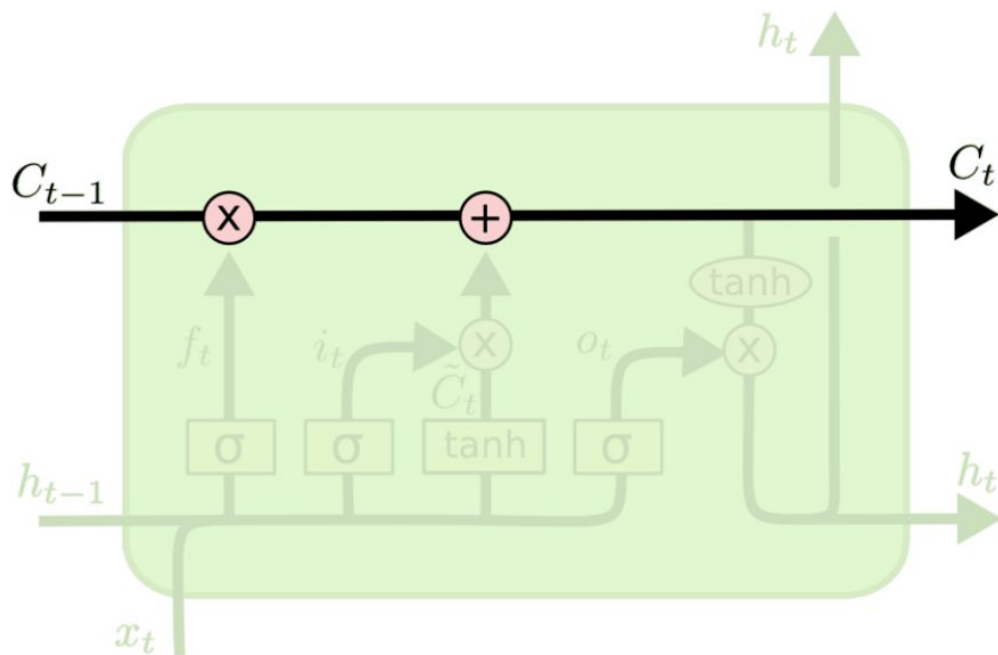
Sequence input



Images from Chris Olah's slides

Long short term memory (LSTM)

Memory cell is critical



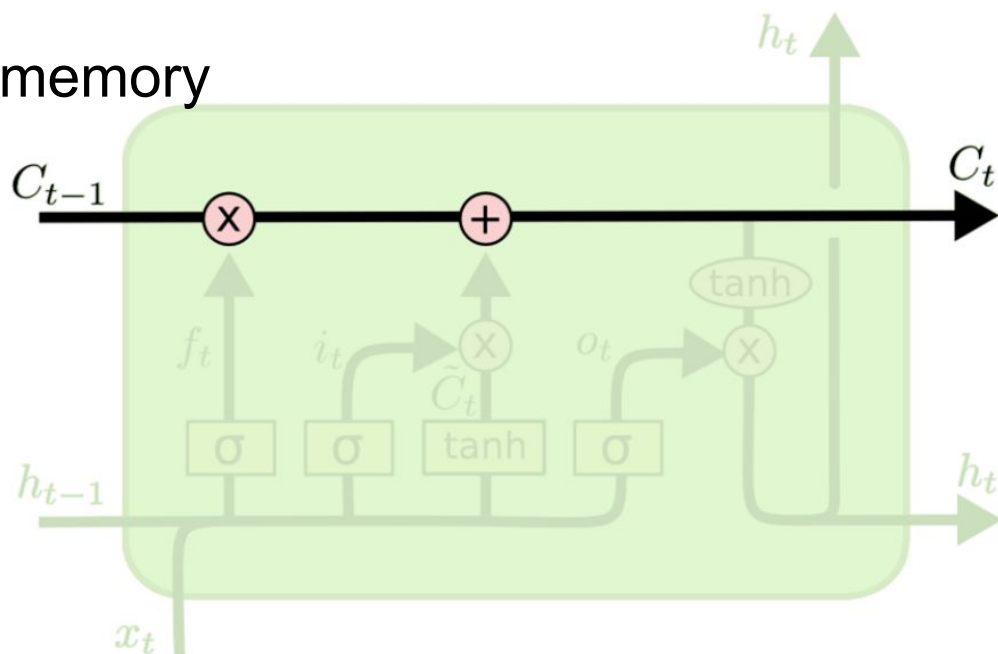
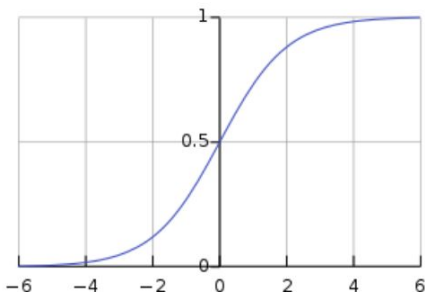
Images from Chris Olah's slides

LSTM- forget gates

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

Decides what to forget in memory

Sigmoid function



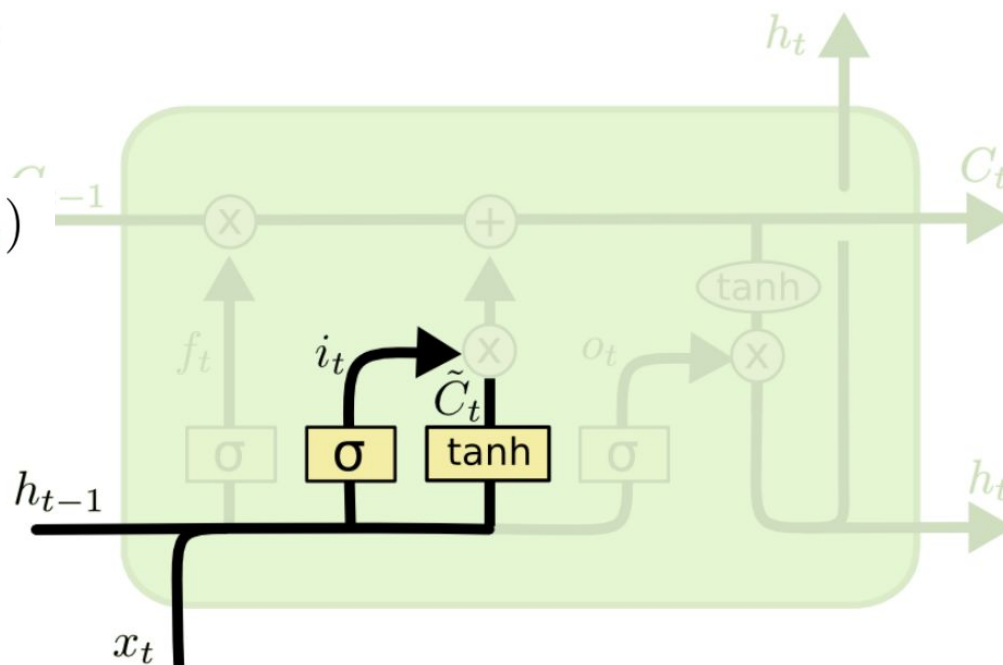
Images from Chris Olah's slides

LSTM- input gates

Controls how much/ what in memory cell

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

$$g_t = \tanh(U_g x_t + W_g h_{t-1} + b_g)$$



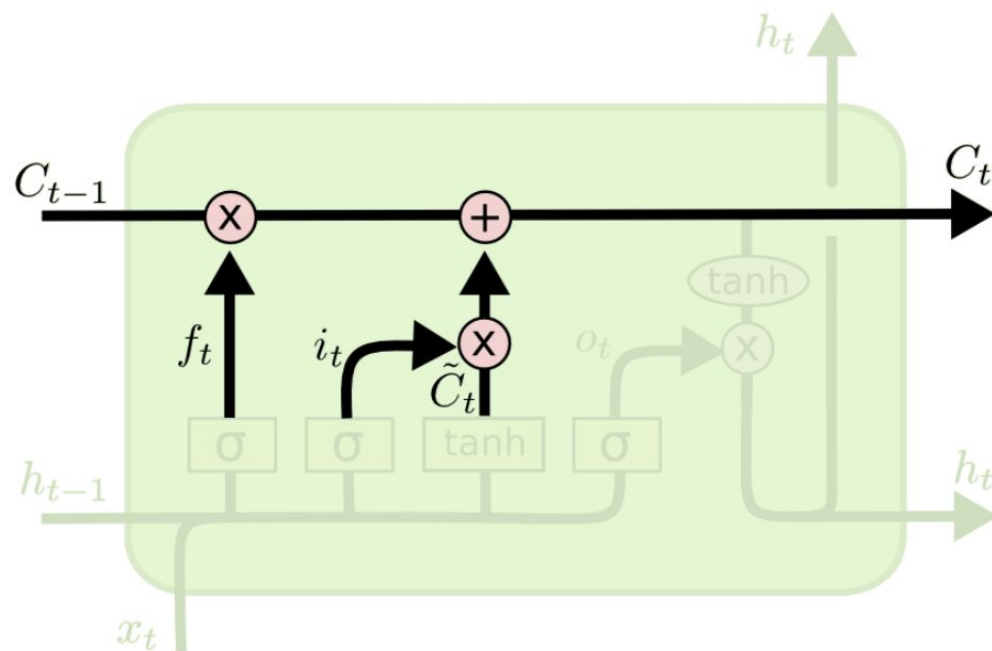
Images from Chris Olah's slides

LSTM- input gates

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

\odot Element-wise multiply

Input gates allow to add to cell
Forget gates allows to forget



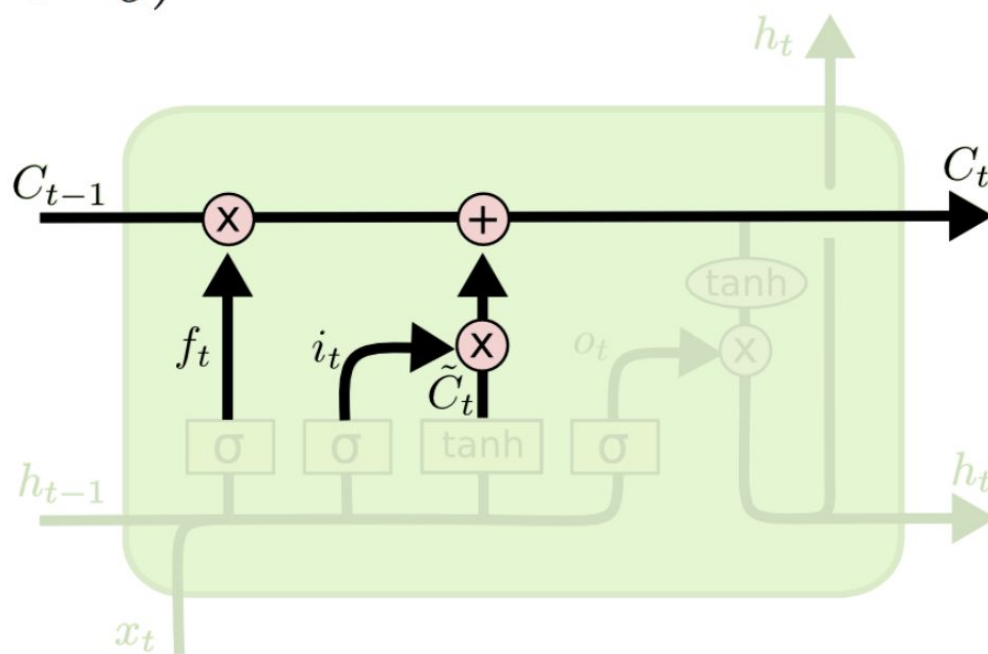
Images from Chris Olah's slides

LSTM - output

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

Controls what goes out of memory cell

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



Images from Chris Olah's slides

LSTM

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

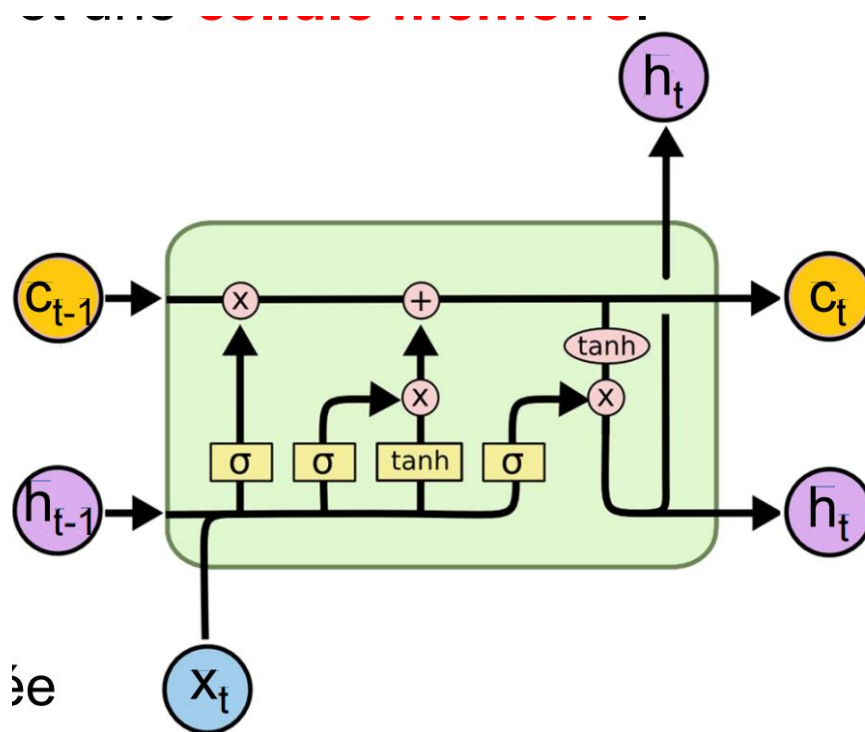
$$f_t = \sigma(U_f x_t + W_f h_{t-1} + b_f)$$

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

$$g_t = \tanh(U_g x_t + W_g h_{t-1} + b_g)$$

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

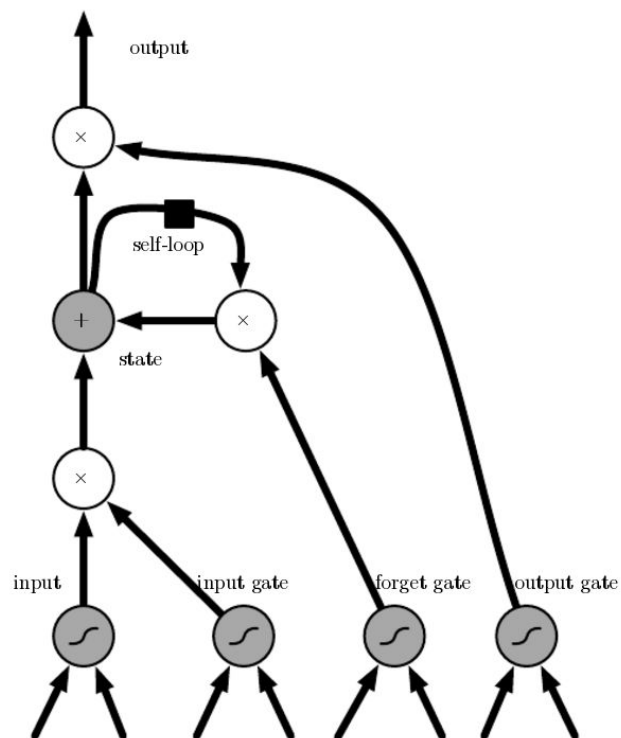
$$h_t = o_t \odot \tanh(c_t)$$



Images from Chris Olah's slides

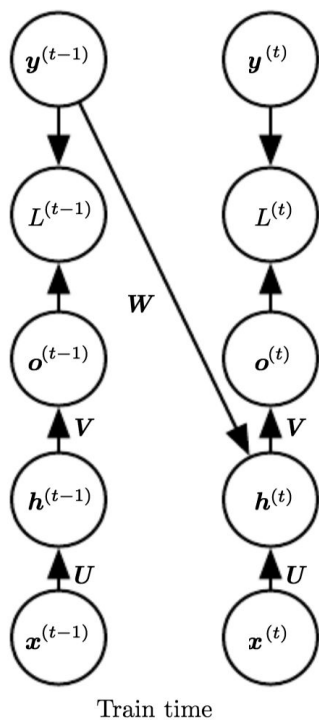
LSTM

- Long-short term memories (LSTM)

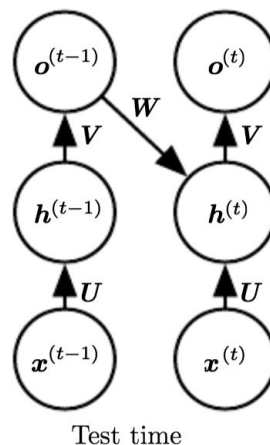


Teacher forcing

- Network uses its output at previous timestep as input



Teacher forcing



Free-running

Images from the “Deep Learning book”

Teacher forcing/ Free-running

- Derived from maximum likelihood criteria

$$\begin{aligned} \log p(y_2, y_1 | x_2, x_1) \\ = \log p(y_2 | y_1, x_2, x_1) + \log p(y_1 | x_2, x_1) \end{aligned}$$

- Pass in ground truth y_1 at time step 2
- Used for generative tasks
 - Language modeling

Issues with teacher forcing

- Difference between train and test
 - During test time, model makes small mistakes
 - Errors accumulate, models has never seen mistakes
 - Low error during training, high during test

Solutions for teacher forcing

No easy solutions

- Scheduled sampling
 - Fed in either free-running or ground truth input during training
- Professor forcing
 - Use a discriminator to match distributions during teacher forcing and free-running

RNNs conditioned on context

For example

- Conditioned language generation
 - Image captioning

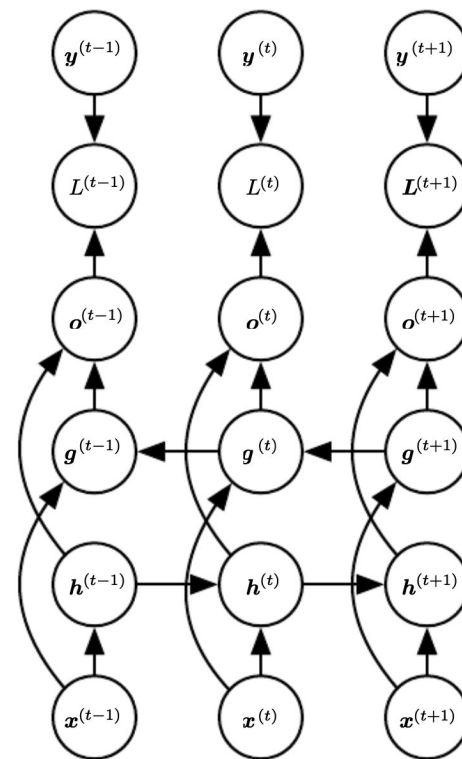
How to pass extra input:

- As an extra input at every step
- As an initial hidden state h_0
- both

Bidirectional RNN

So far, all outputs are conditioned on the past, what about outputs that are conditioned both on the past and the future?

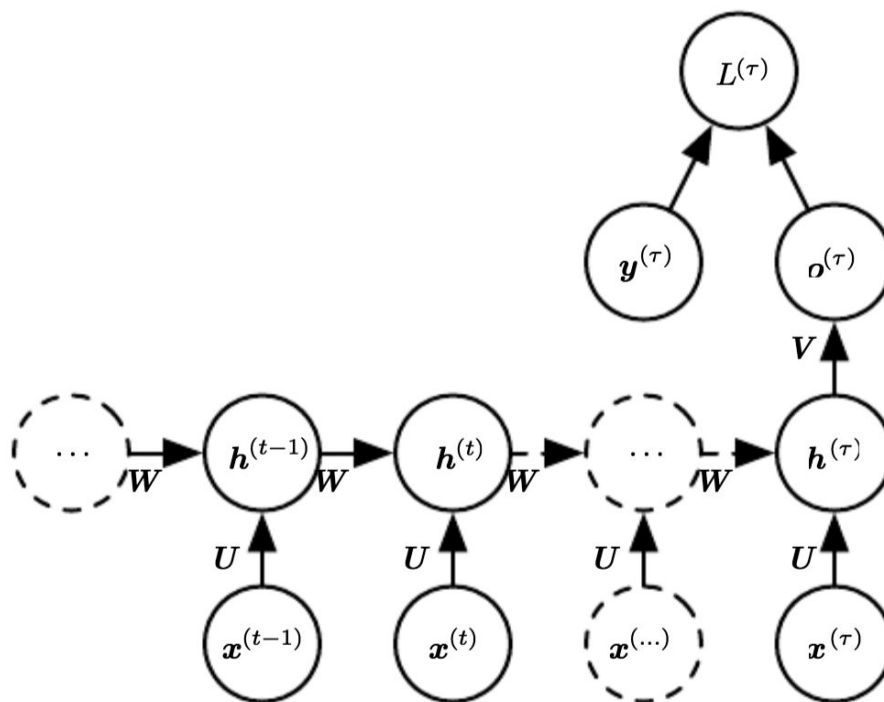
- Speech recognition
 - Depends on both past and future



RNN with single output

Image classification

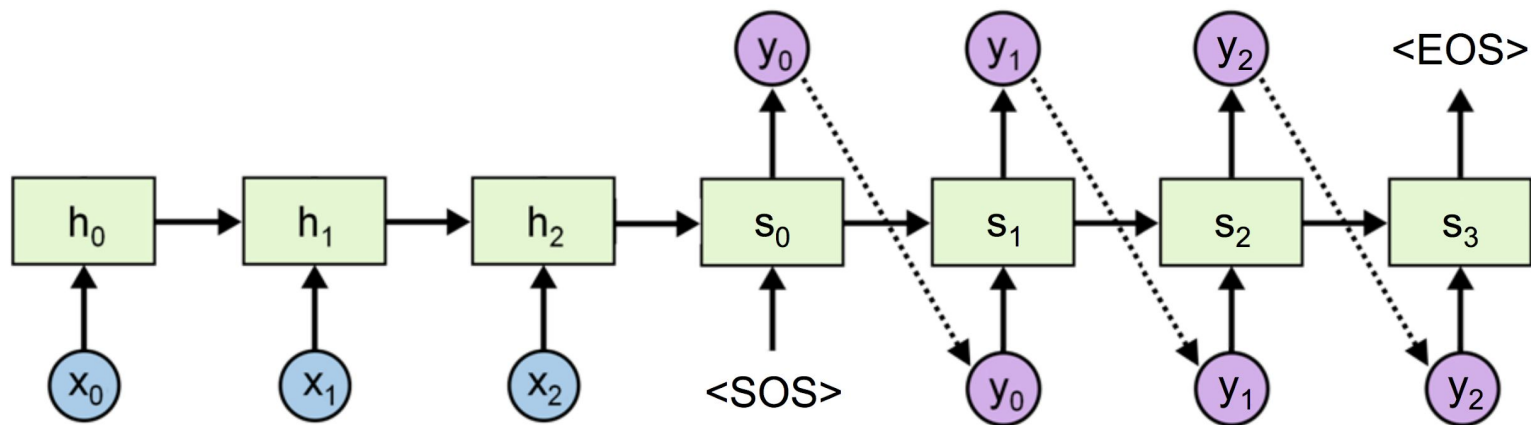
- MNIST recognition
 - Mapping all input
To single output



Encoder - Decoder RNN

Sequence to sequence

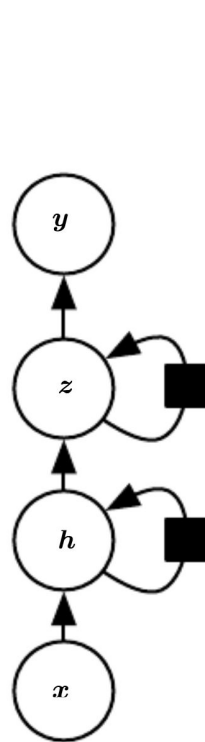
- Encoder
 - maps inputs to a single vector
- Decoder
 - a conditional RNN



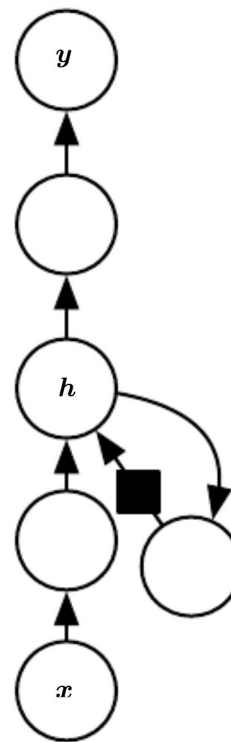
Images from Chris Olah's slides

Deep recurrent networks

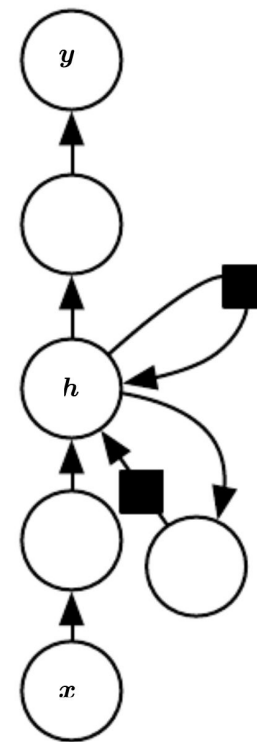
- Does depth help?
 - Does help
 - Harder to optimize
- How to make it deep?
 -



(a)

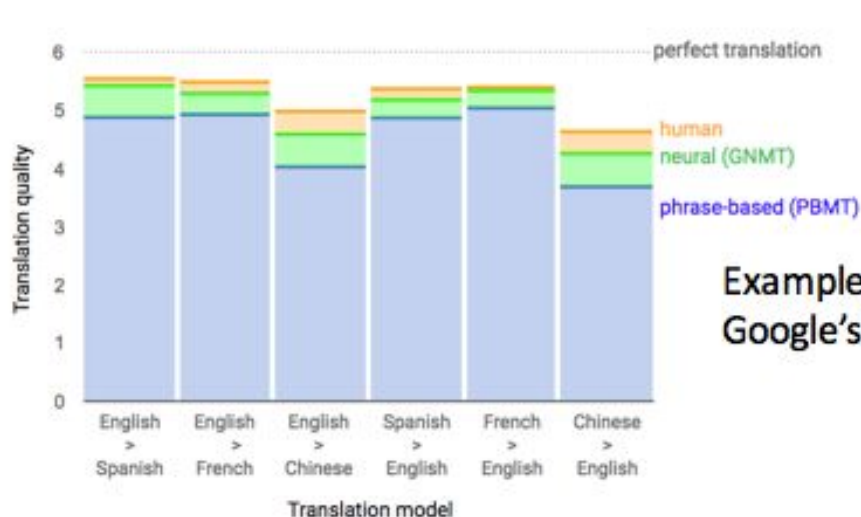


(b)



(c)

Comparing Google's Neural Machine Translation (GNMT) with previous Phrase-Based method (PBMT)



Input sentence:

李克強此行將啟動中加總理年度對話機制，與加拿大總理杜魯多舉行兩國總理首次年度對話。

Translation (PBMT):

Li Keqiang premier added this line to start the annual dialogue mechanism with the Canadian Prime Minister Trudeau two prime ministers held its first annual session.

Translation (GNMT):

Li Keqiang will start the annual dialogue mechanism with Prime Minister Trudeau of Canada and hold the first annual dialogue between the two premiers.

Translation (human):

Li Keqiang will initiate the annual dialogue mechanism between premiers of China and Canada during this visit, and hold the first annual dialogue with Premier Trudeau of Canada.

The New York Times

SCIENCE

Researchers Announce Advance in Image-Recognition Software

By JOHN MARKOFF NOV. 17, 2014

MOUNTAIN VIEW, Calif. — Two groups of scientists, working independently, have created artificial intelligence software capable of **recognizing and describing the content of photographs and videos** with far greater accuracy than ever before, sometimes even mimicking human levels of understanding.

Captioned by Human and by Google's Experimental Program



Human: "A young hockey player playing in the ice rink."
Computer model: "Two hockey players are fighting over the puck."

2 of 6



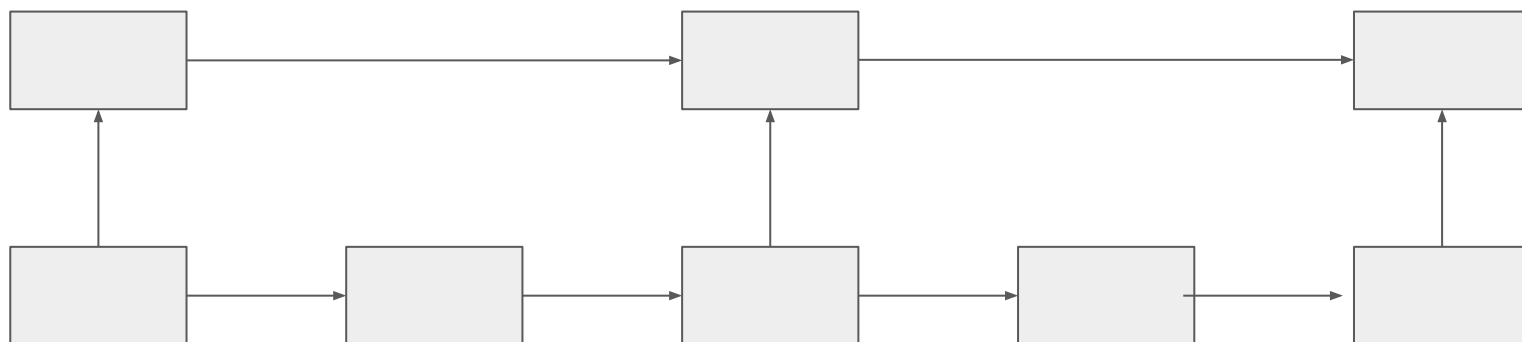
Skip connections through time

Helps with exploding and vanishing gradients across time

- Gradients can propagate over longer spans through skip connections

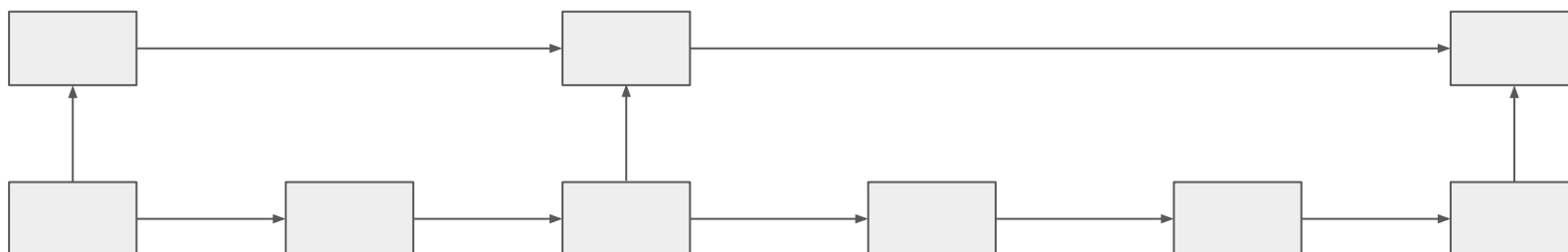
Learning hierarchical representation in RNNs?

- Updates higher layers less frequently.
- Clockwork RNN (Bengio et. al, 94, Schmidhuber et. al, 14)
 - Updates at fixed time steps
- Gradients can propagate over longer spans through slow time-scale paths



Learning hierarchical representation in RNNs?

- Previous work: Hierarchical Multiscale Recurrent Neural Network (Chung et. al, 16)
 - Learns when to update.
- Hierarchical Encoder Network (our method)
 - Learns when to update conditionally.



Neural networks with attention

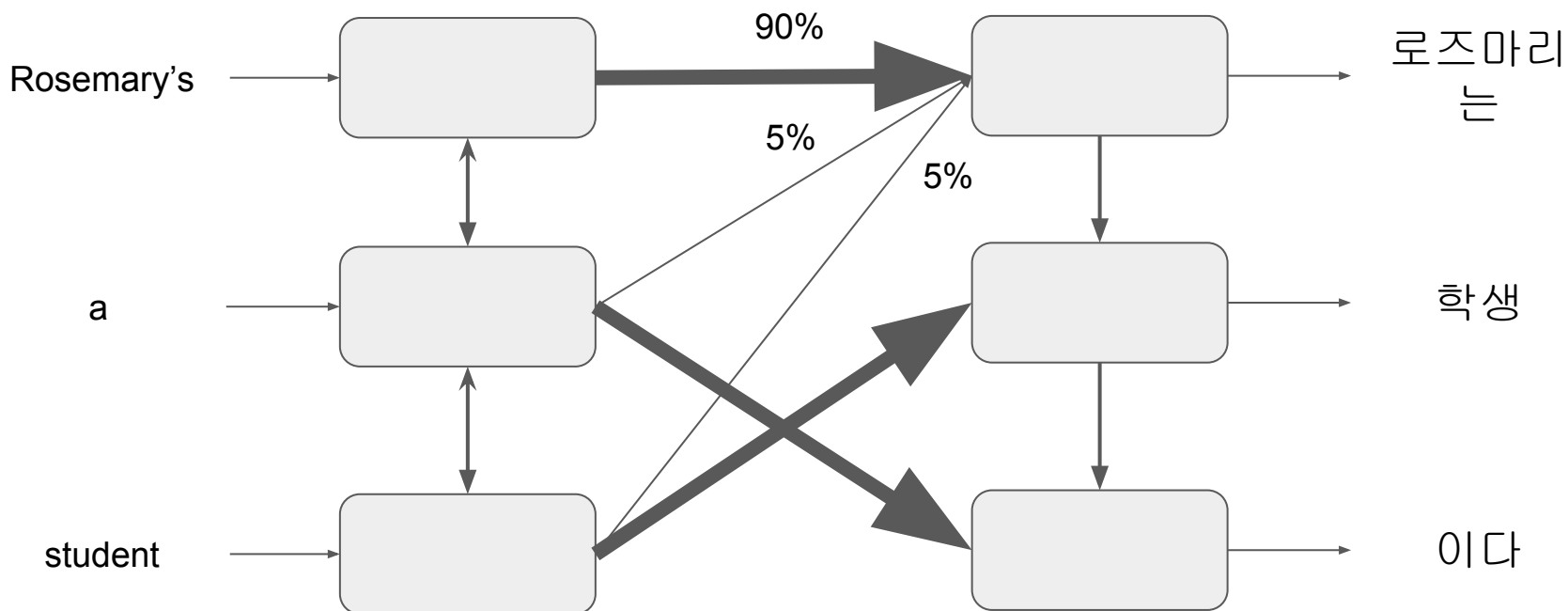
- Encoder-decoder
 - Last hidden state contains all information for inputs
 - Bottleneck state
 - How to make it better?

- Attention

- $$a_j = \frac{e^{A(z_i, h_j)}}{\sum_{j'} e^{A(z_i, h_{j'})}}$$

- $$r = \sum_j a_j h_j$$

RNNs with attention



Open research questions

- How to efficiently assign credit in RNNs
- How to regularize RNN for better longer term planning
- How to build models that are better adjusted to the difference between training (teacher forcing) and test (free-running) mode?