Tutorial on Deep Learning for Natural Language Processing
ICON-2017, Jadavpur University, Kolkata, India.

Recurrent Neural Network

Md Shad Akhtar
Research Scholar
AI-NLP-ML Group
Department of Computer Science & Engineering
Indian Institute of Technology Patna

shad.pcs15@iitp.ac.in
https://iitp.ac.in/~shad.pcs15/

Outline

- Recurrent Neural Network (RNN)
  - Training of RNNs
    - BPTT
  - Visualization of RNN through Feed-Forward Neural Network
  - Usage
  - Problems with RNNs
- Long Short Term Memory (LSTM)
- Attention Mechanism

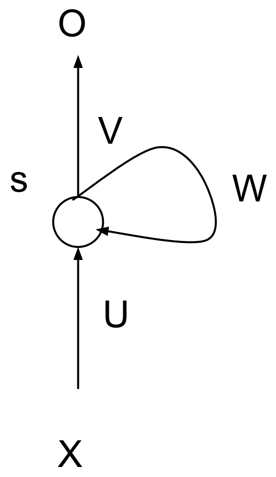
Recurrent Neural Network (RNN)

Basic definition:

A neural network with feedback connections.

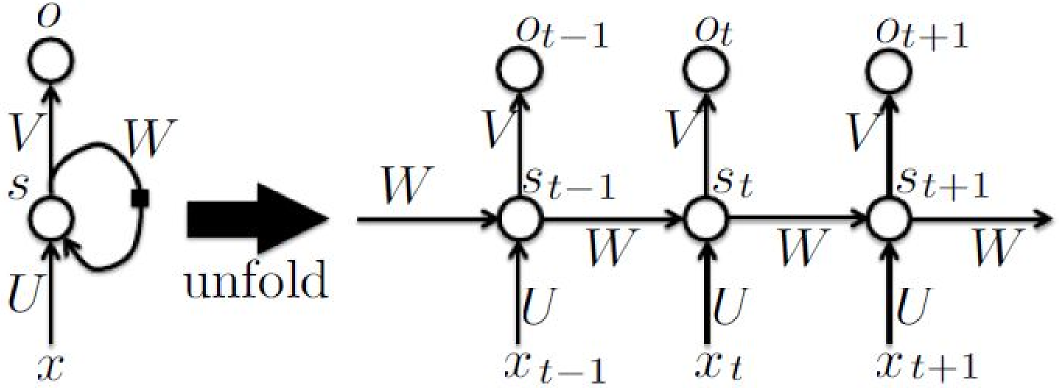
X: Input
O: Ouput
S: Hidden state

Weights: [U,V,W]
Learned during training



Recurrent Neural Network (RNN)

- Enable networks to do temporal processing
- Good at learning sequences
- Acts as memory unit



Memory

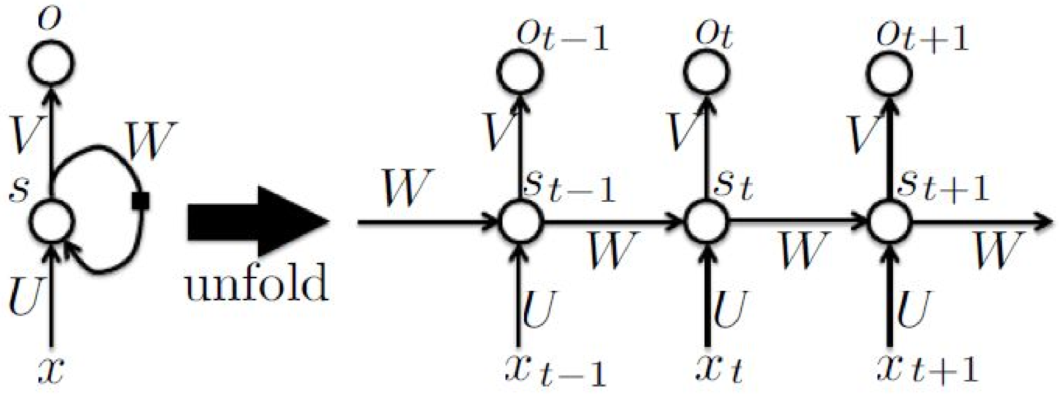
$$
\begin{aligned}
a_t &= b + W s_{t-1} + U x_t \\
s_t &= \tanh(a_t) \\
o_t &= c + V s_t \\
p_t &= \mathrm{softmax}(o_t)
\end{aligned}
$$

RNN - Example 1

Part-of-speech tagging:

- Given a sentence X, tag each word its corresponding grammatical class.

***O* = [*PRP* *VBP* *NNS*]**



***X* = [*I* *love* *mangoes*]**

RNN - Example 2

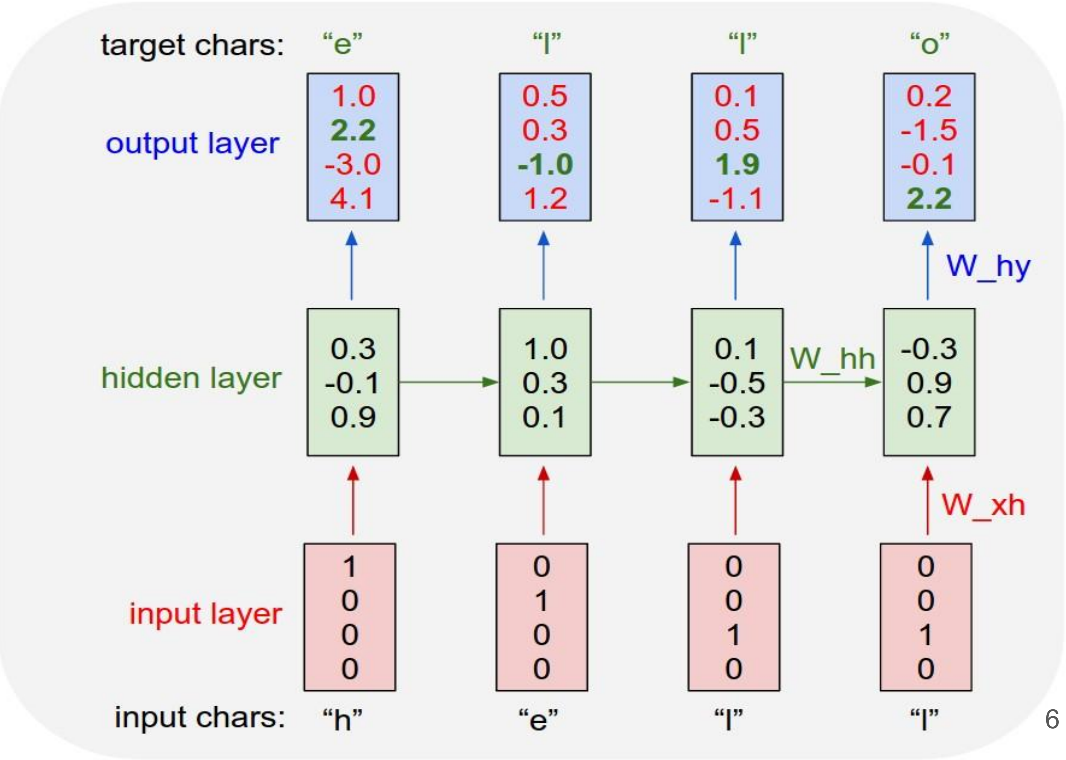
Character level language model:

- Given previous and current characters, predict the next character in the sequence.

Let

- **Vocabulary: [h,e,l,o]**

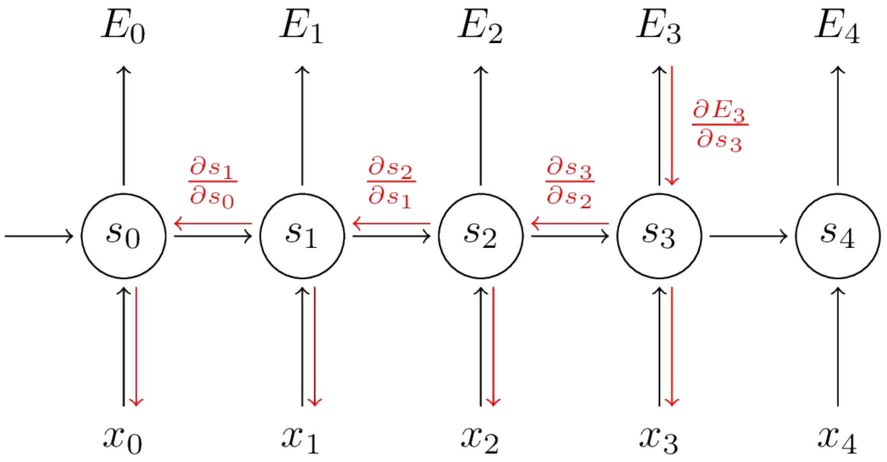
- **One-hot representations**
  - **h = [1 0 0 0]**
  - **e = [0 1 0 0]**
  - **l = [0 0 1 0]**
  - **o = [0 0 0 1]**



Training of RNNs

How to train RNNs?

- Typical FFN
  - Backpropagation algorithm
- RNNs
  - A variant of backpropagation algorithm namely **Back-Propagation Through Time (BPTT)**.

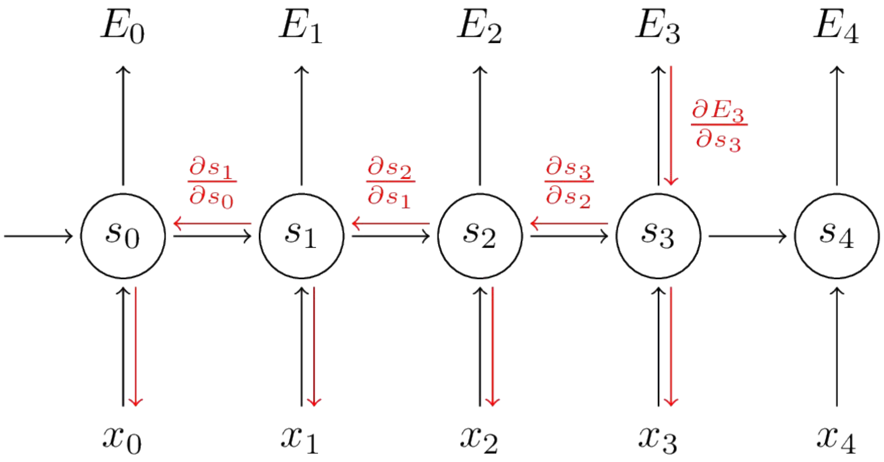


BackPropagation Through Time (BPTT)

Error for an instance = Sum of errors at each time step of the instance

Gradient of error

$$\frac{\partial E}{\partial W} = \sum_t \frac{\partial E_t}{\partial W}$$



BackPropagation Through Time (BPTT)

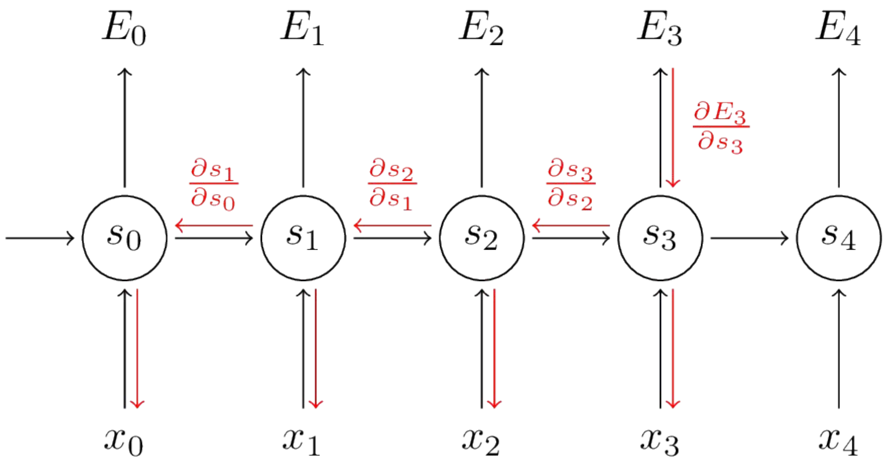
For V

$$\frac{\partial E_3}{\partial V} = \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial V}$$

For W (Similarly for U)

$$\frac{\partial E_3}{\partial W} = \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial s_3}\frac{\partial s_3}{\partial W}$$

$$\frac{\partial E_3}{\partial W} = \sum_{k=0}^{3}\frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial s_3}\frac{\partial s_3}{\partial s_k}\frac{\partial s_k}{\partial W}$$



Visualization of RNN through Feed-Forward Neural Network

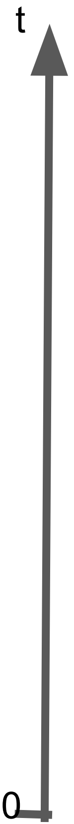
Problem, Data and Network Architecture

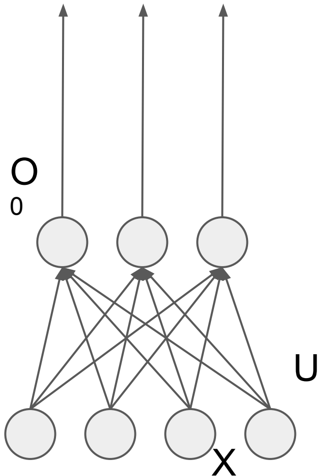
- Problem:
  - I/p sequence (X) : $X^0, X^1, \ldots, X^T$
  - O/p sequence (O) : $O^0, O^1, \ldots, O^T$

- Representation of data:
  - I/p dimension : 4
    - $X^0 \rightarrow 0\ 1\ 1\ 0$
  - O/p dimension : 3
    - $O^0 \rightarrow 0\ 0\ 1$

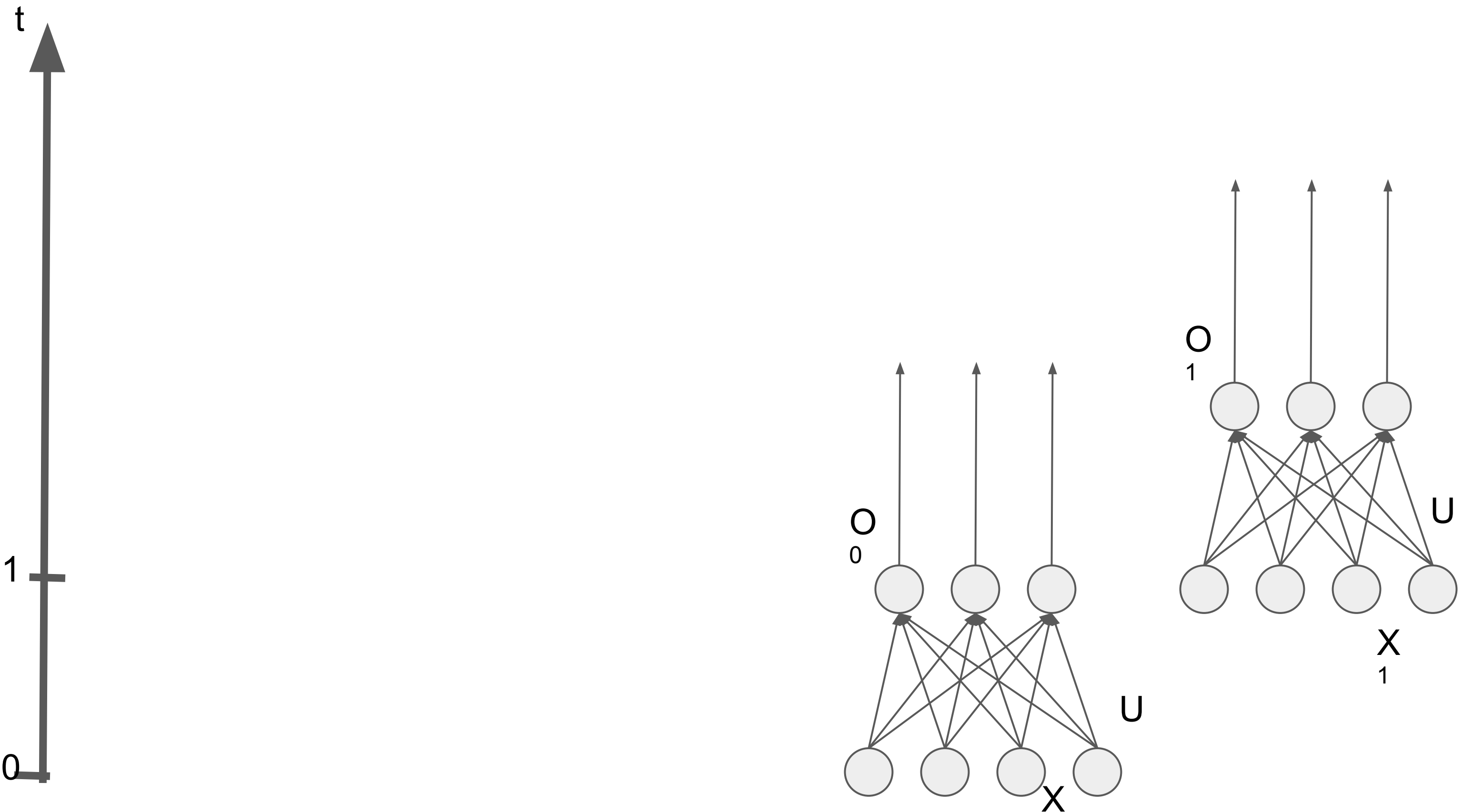
- Network Architecture
  - Number of neurons at I/p layer : 4
  - Number of neurons at O/p layer : 3
  - Do we need hidden layers?
    - If yes, number of neurons at each hidden layers

Network @ t = 0

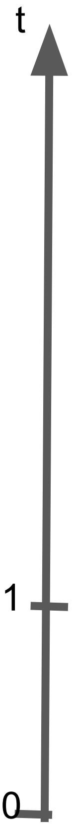


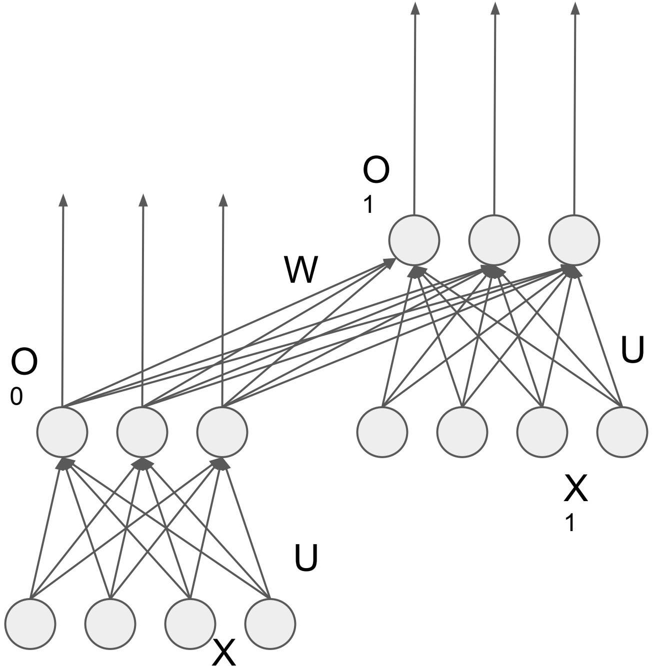


Network @ t = 1

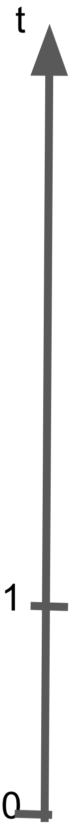


Network @ t = 1

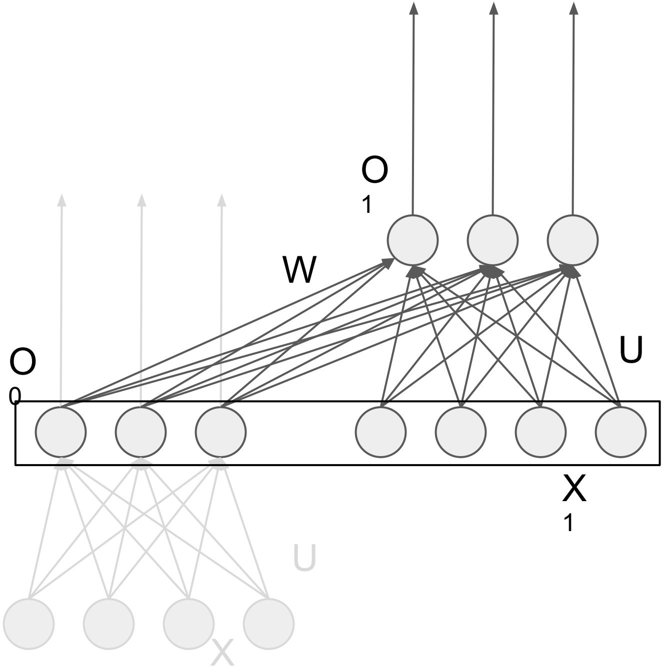




Network @ t = 1

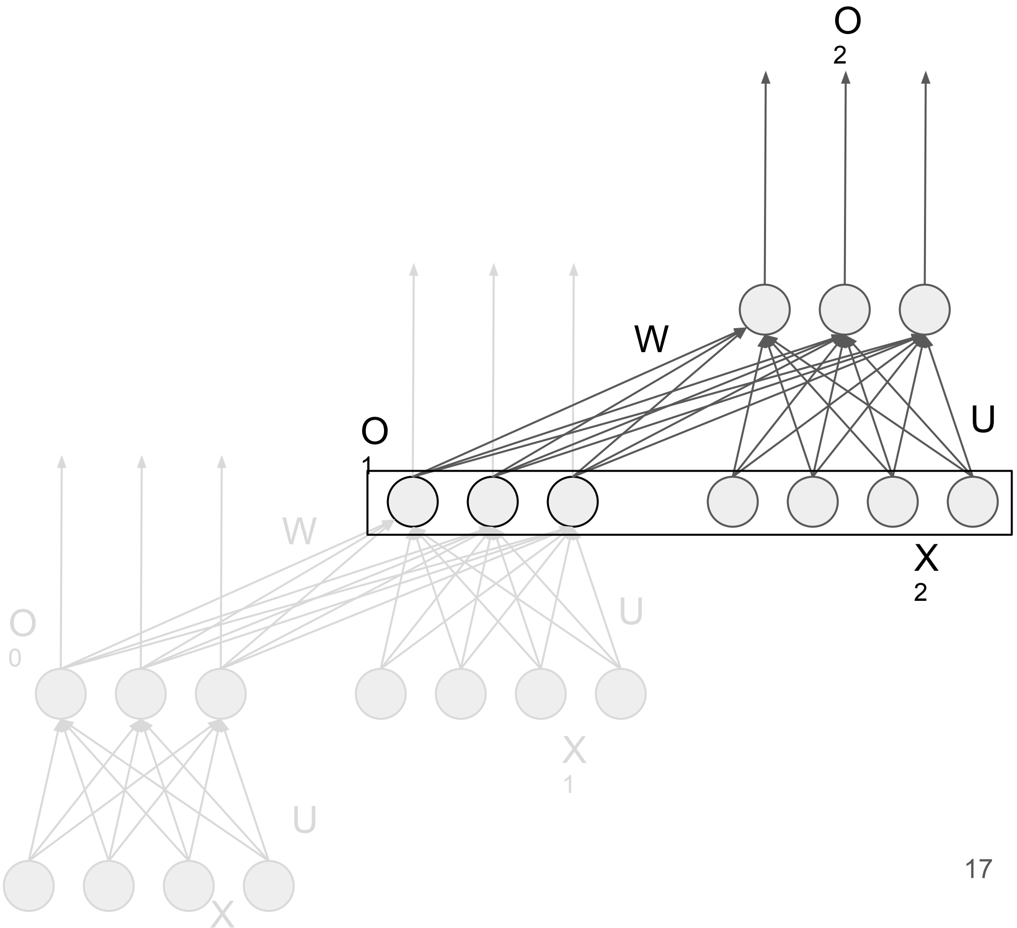


$$O^1 = f(W.O^0 + U.X^1)$$
$$= f([W, U] \,.\, [O^0, x^1])$$

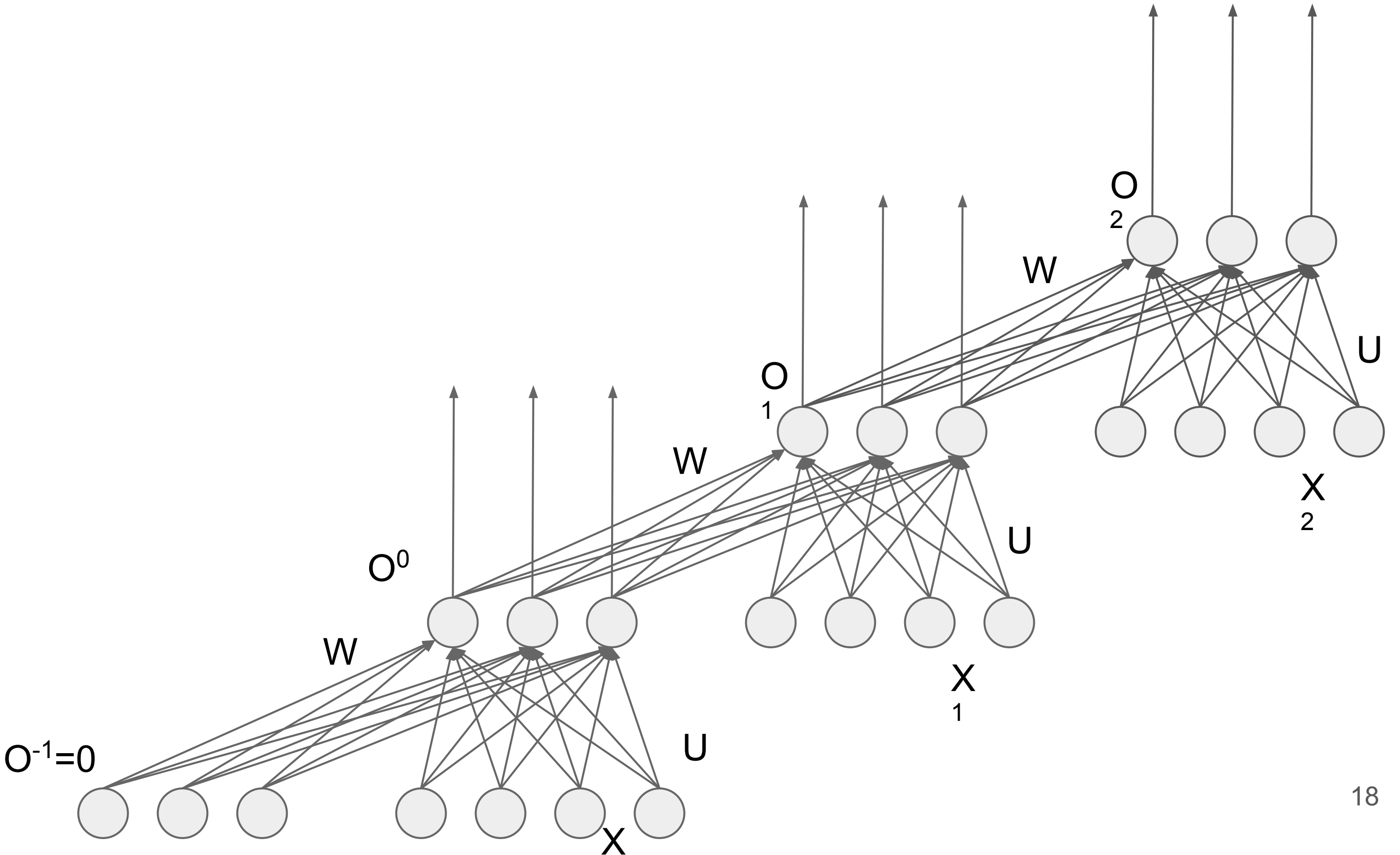


Network @ t = 2

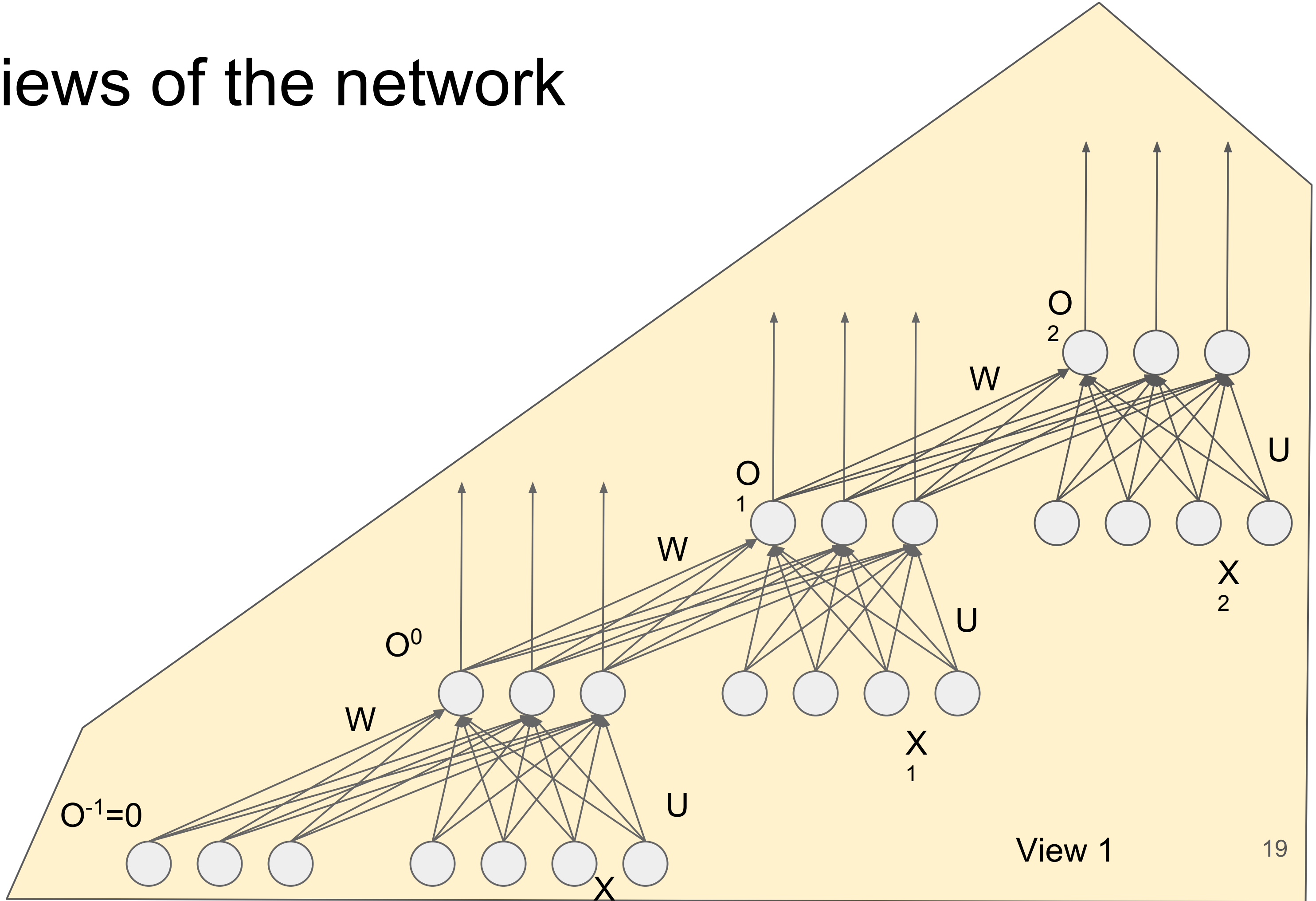
$$
\begin{aligned}
O^2 &= f(W.O^1 + U.X^2) \\
&= f([W, U] \,.\, [O^1, x^2])
\end{aligned}
$$



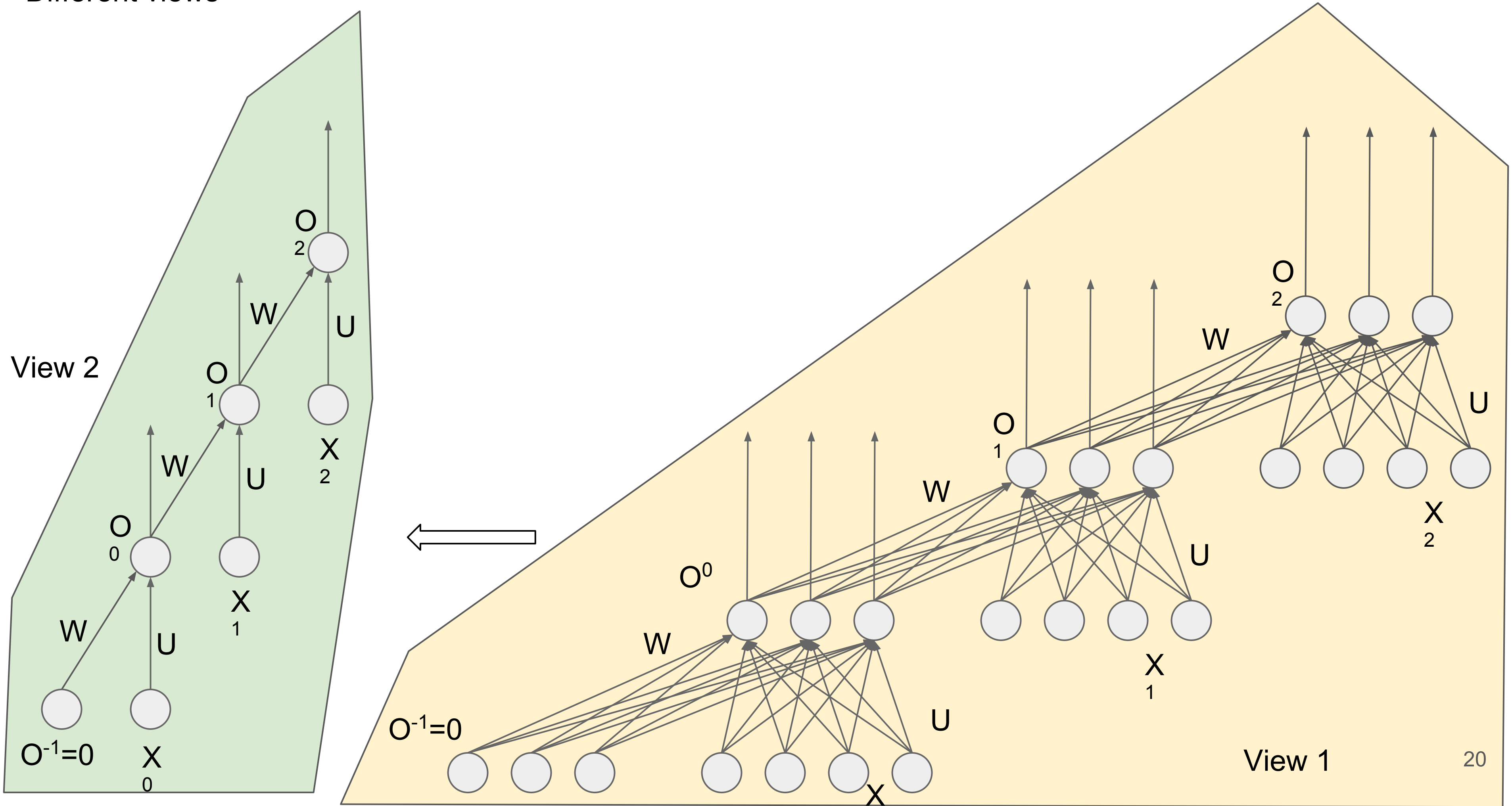
Complete Network



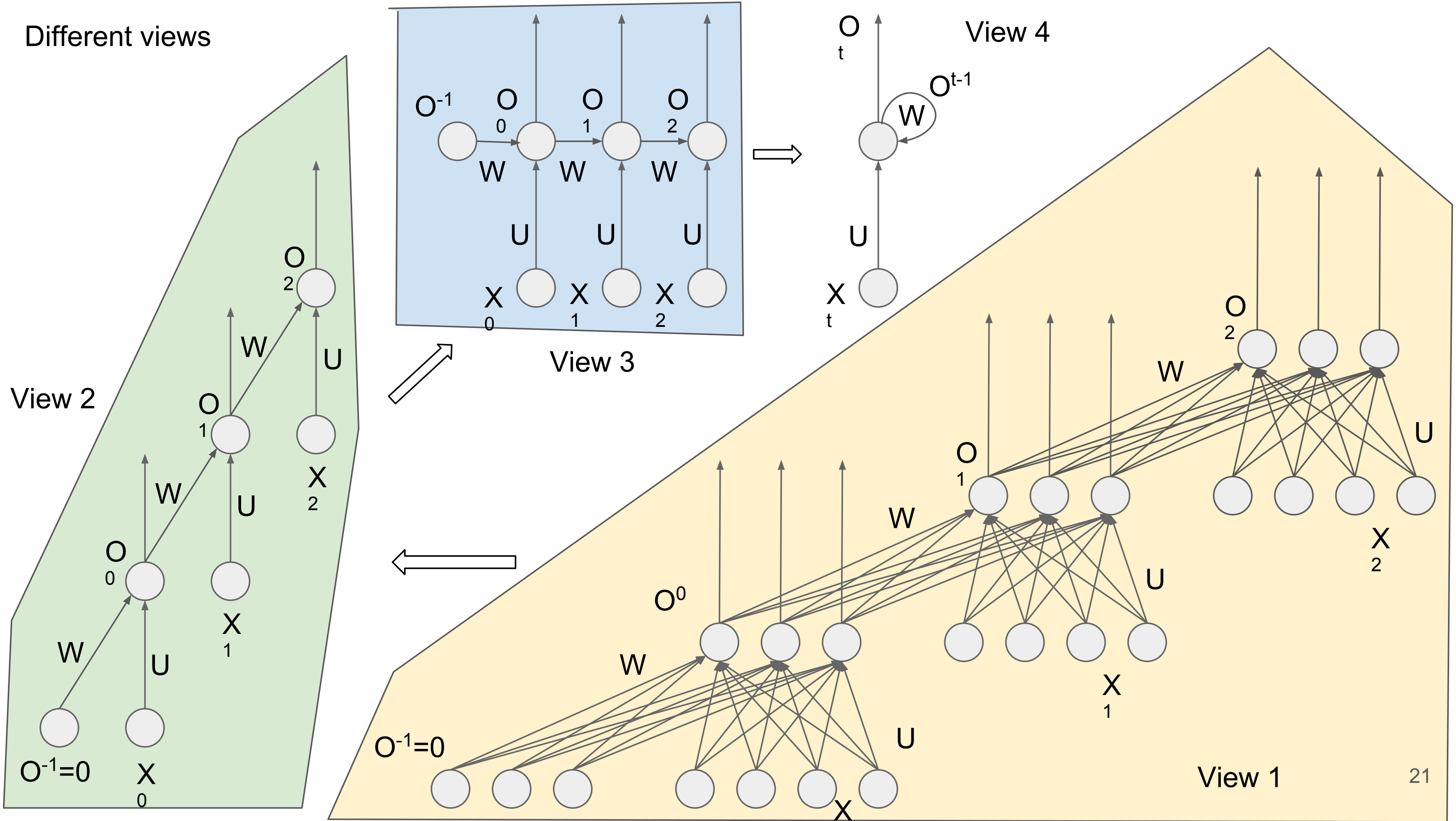
Different views of the network



Different views



Different views



When to use RNNs

Usage

- Depends on the problems that we aim to solve.
- Typically good for sequence processings.
- Some sort of memorization is required.

Bit reverse problem

- Problem definition:
  - **Problem 1:** Reverse a binary digit.
    - 0 → 1 and 1 → 0
  - **Problem 2:** Reverse a sequence of binary digits.
    - 0 1 0 1 0 0 1 → 1 0 1 0 1 1 0
    - Sequence: Fixed or Variable length
  - **Problem 3:** Reverse a sequence of bits over time.
    - 0 1 0 1 0 0 1 → 1 0 1 0 1 1 0
  - **Problem 4:** Reverse a bit if the current i/p and previous o/p are same.

Input sequence	1	1	0	0	1	0	0	0	1	1
Output sequence	1	0	1	0	1	0	1	0	1	0

Data

Let

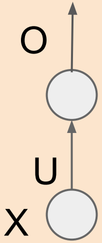
- **Problem 1**
  - I/p dimension: **1 bit** O/p dimension: **1 bit**
- **Problem 2**
  - Fixed
    - I/p dimension: **10 bit** O/p dimension: **10 bit**
  - Variable: Pad each sequence upto max sequence length: **10**
    - Padding value: **-1**
    - I/p dimension: **10 bit** O/p dimension: **10 bit**
- **Problem 3 & 4**
  - Dimension of each element of I/p (X) : **1 bit**
  - Dimension of each element of O/p (O) : **1 bit**
  - Sequence length : **10**

No. of I/p neurons = I/p dimension
No. of O/p neurons = O/p dimension

Network Architecture

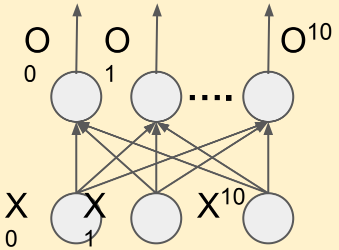
Problem 1:

- I/p neurons = 1
- O/p neurons = 1



Problem 2: Fixed & Variable

- I/p neurons = 10
- O/p neurons = 10

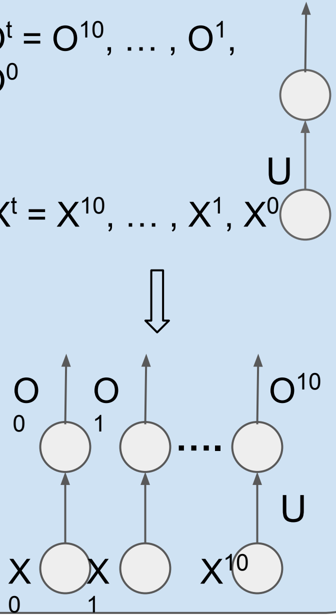


Problem 3:

- I/p neurons = 1
- O/p neurons = 1
- Seq len = 10

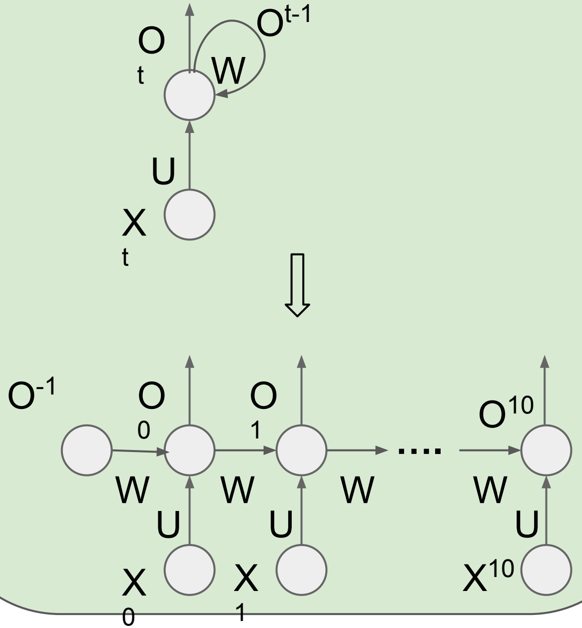
$O^t = O^{10}, \dots , O^1, O^0$

$X^t = X^{10}, \dots , X^1, X^0$



Problem 4:

- I/p neurons = 1
- O/p neurons = 1
- Seq len = 10



Different configurations of RNNs

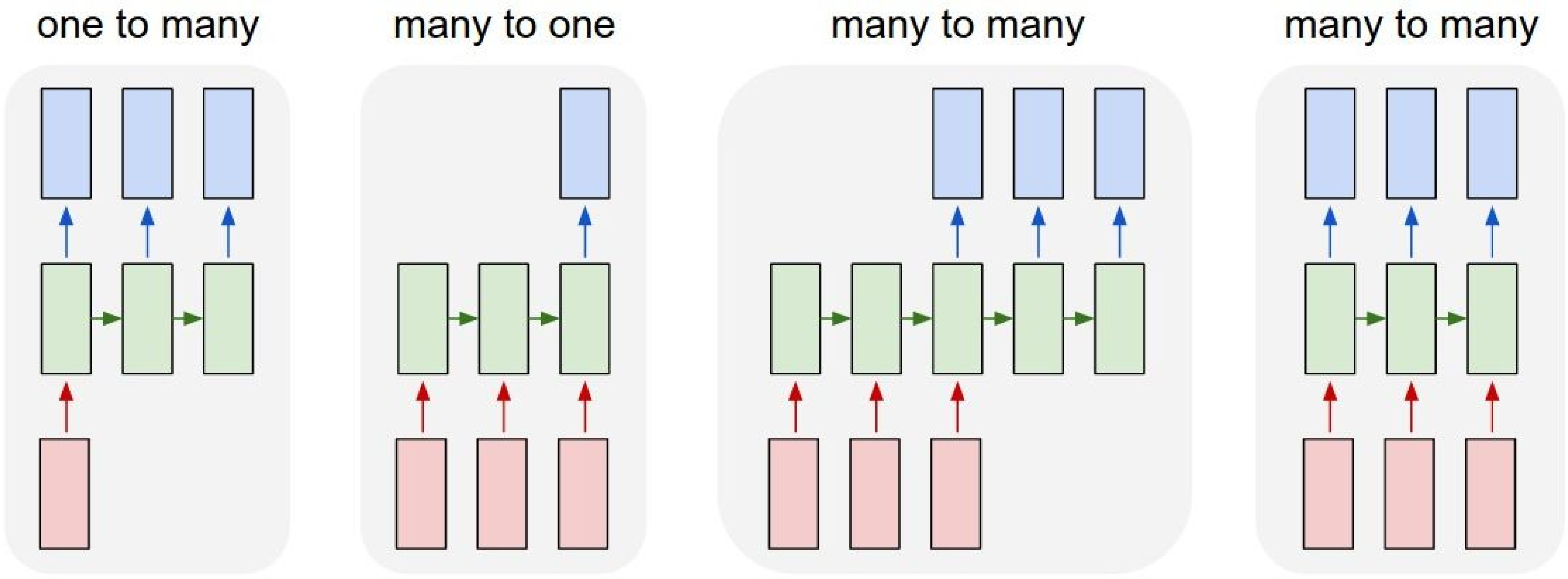
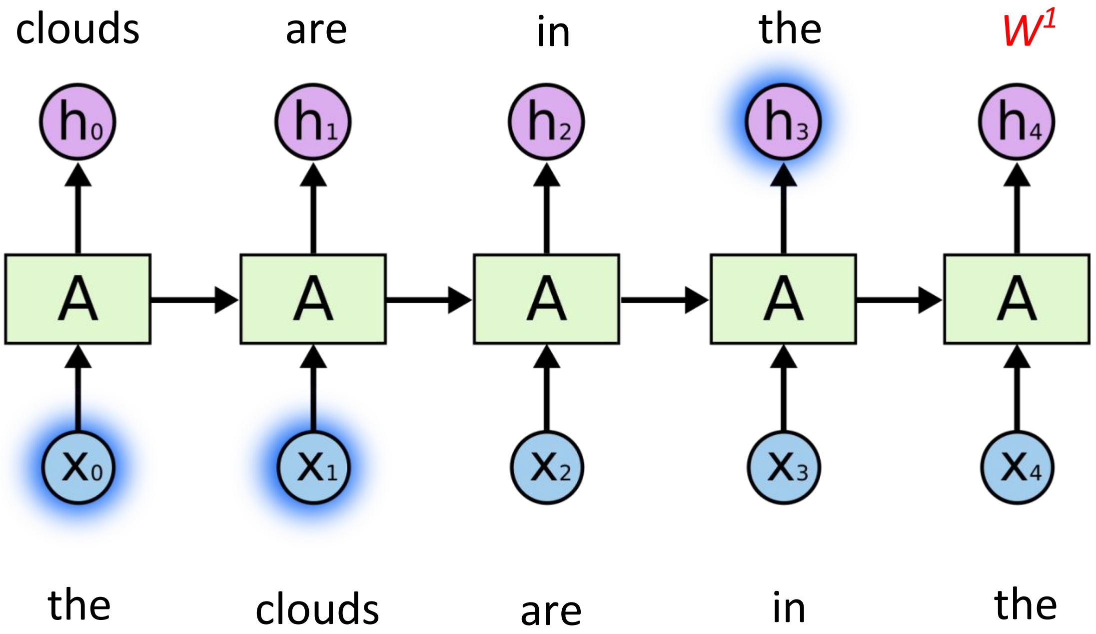


Image Captioning | **Sentiment Analysis** | **Machine Translation** | **Language modelling**

Problems with RNNs

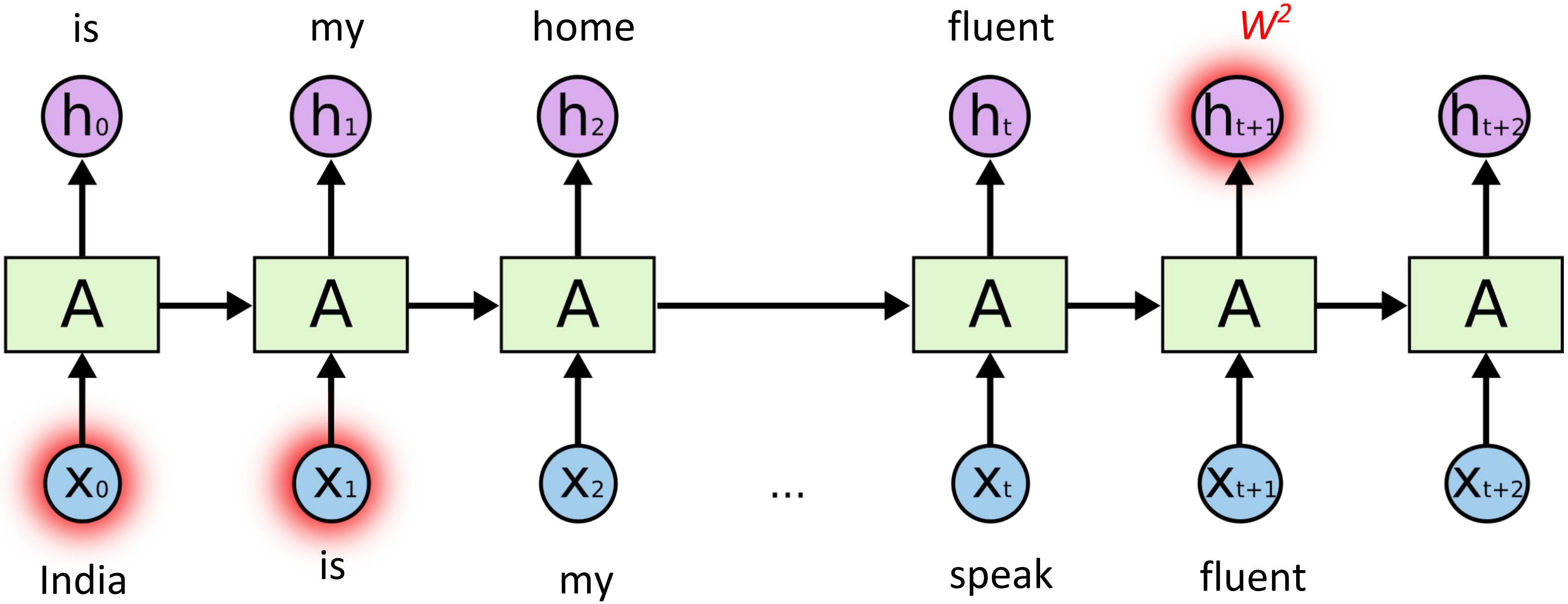
Language modelling: Example - 1

- “the clouds are in the *sky*”



Language modelling: Example - 2

- "India is my home country. I can speak fluent *Hindi*."



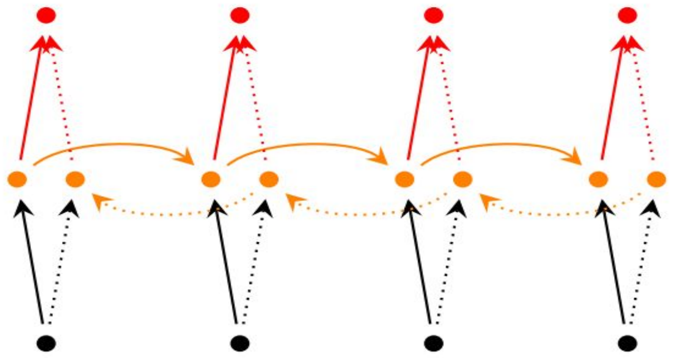
Vanishing/Exploding gradients

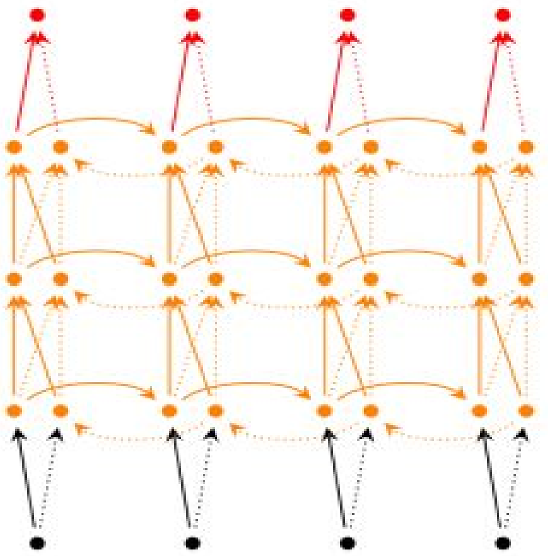
- Cue word for the prediction
  - Example 1: ***sky*** → ***clouds*** [3 units apart]
  - Example 2: ***hindi*** → ***India*** [9 units apart]

- As the sequence length increases, it becomes hard for RNNs to learn "long-term dependencies."
  - **Vanishing gradients:** If weights are small, gradient shrinks exponentially. Network stops learning.
  - **Exploding gradients:** If weights are large, gradient grows exponentially. Weights fluctuate and become unstable.

RNN extensions

- Bi-directional RNN
- Deep (Bi-directional) RNN





Long Short Term Memory (LSTM)

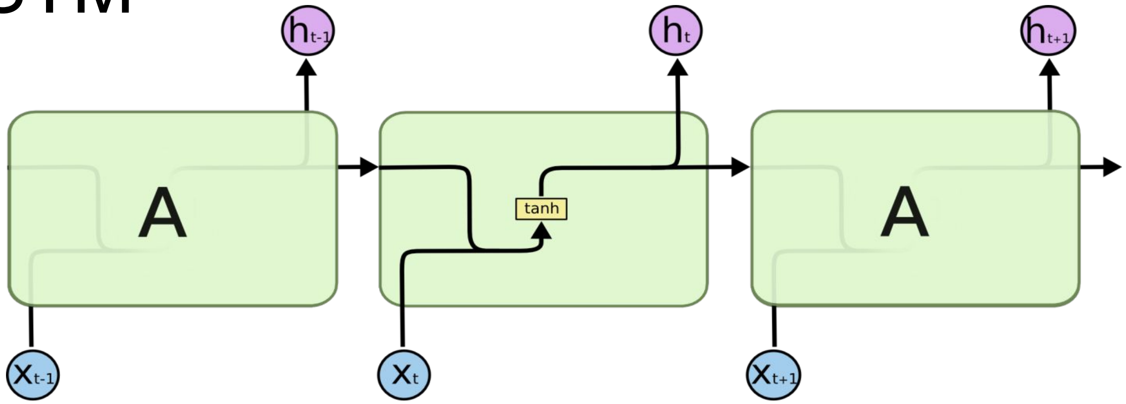
Hochreiter & Schmidhuber (1997)

LSTM

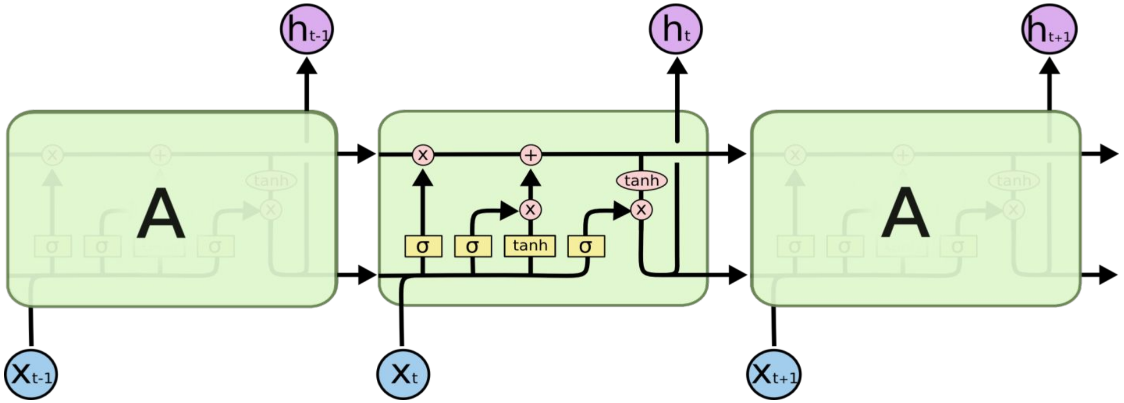
- A variant of simple RNN (Vanilla RNN)
- Capable of learning long dependencies.
- Regulates information flow from recurrent units.

Vanilla RNN vs LSTM

Vanilla RNN cell



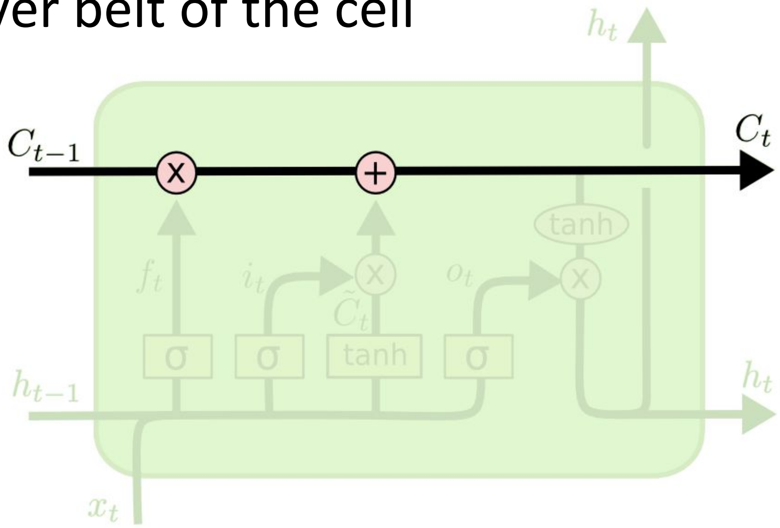
LSTM cell



LSTM cell

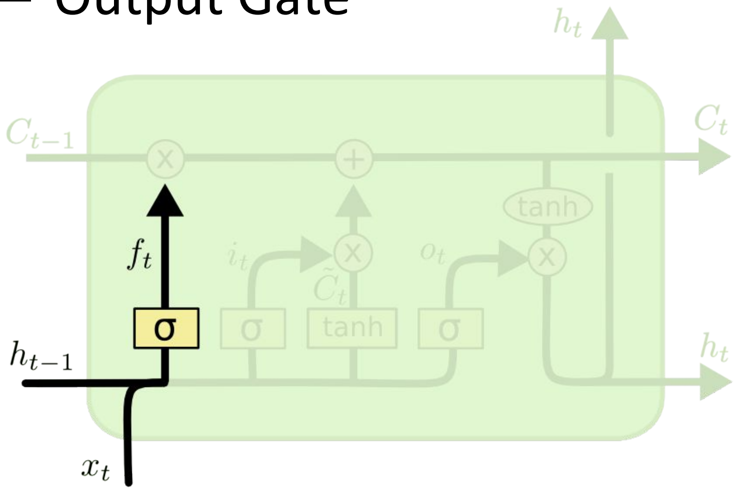
- LSTM removes or adds information to the cell state, carefully regulated by structures called gates.

- Cell state: Conveyer belt of the cell



LSTM gates

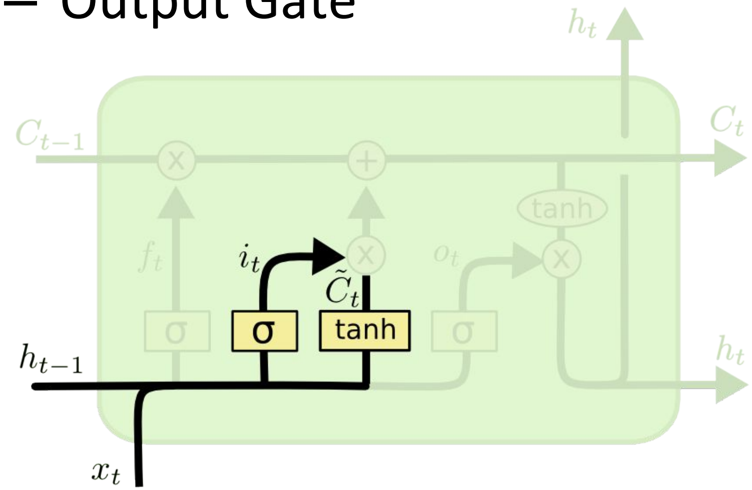
- Each LSTM unit comprises of three gates.
  - Forget Gate: Amount of memory it should forget.
  - Input Gate
  - Output Gate



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

LSTM gates

- Each LSTM unit comprises of three gates.
  - Forget Gate
  - Input Gate: Amount of new information it should memorize.
  - Output Gate

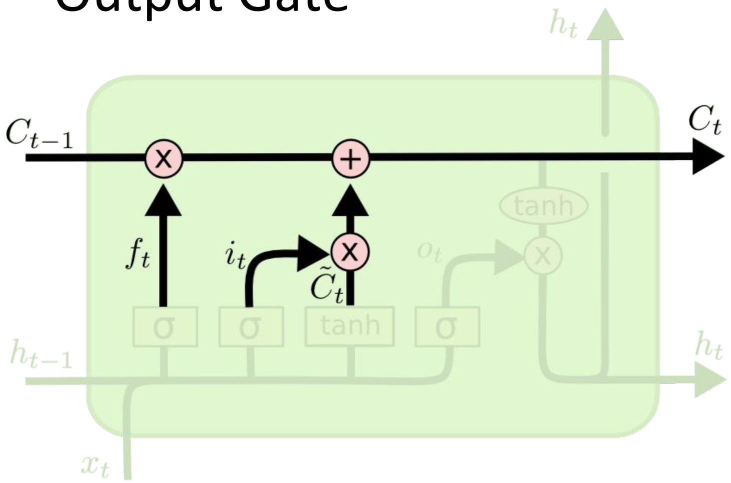


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

LSTM gates

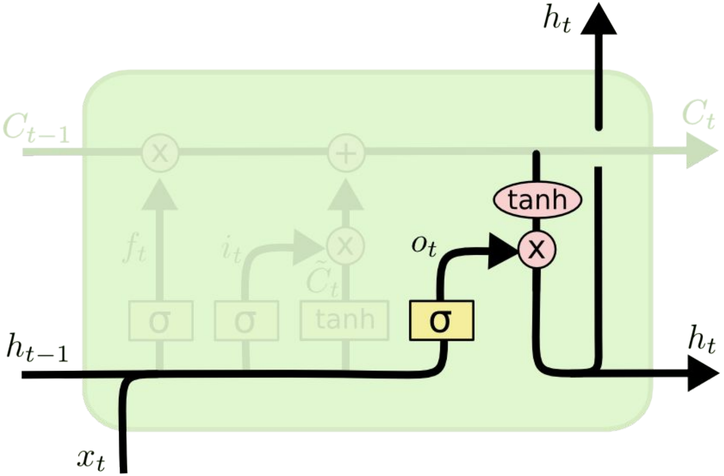
- Each LSTM unit comprises of three gates.
  - Forget Gate: Amount of memory it should forget.
  - Input Gate: Amount of new information it should memorize.
  - Output Gate



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

LSTM gates

- Each LSTM unit comprises of three gates.
  - Forget Gate
  - Input Gate
  - Output Gate: Amount of information it should pass to next unit.



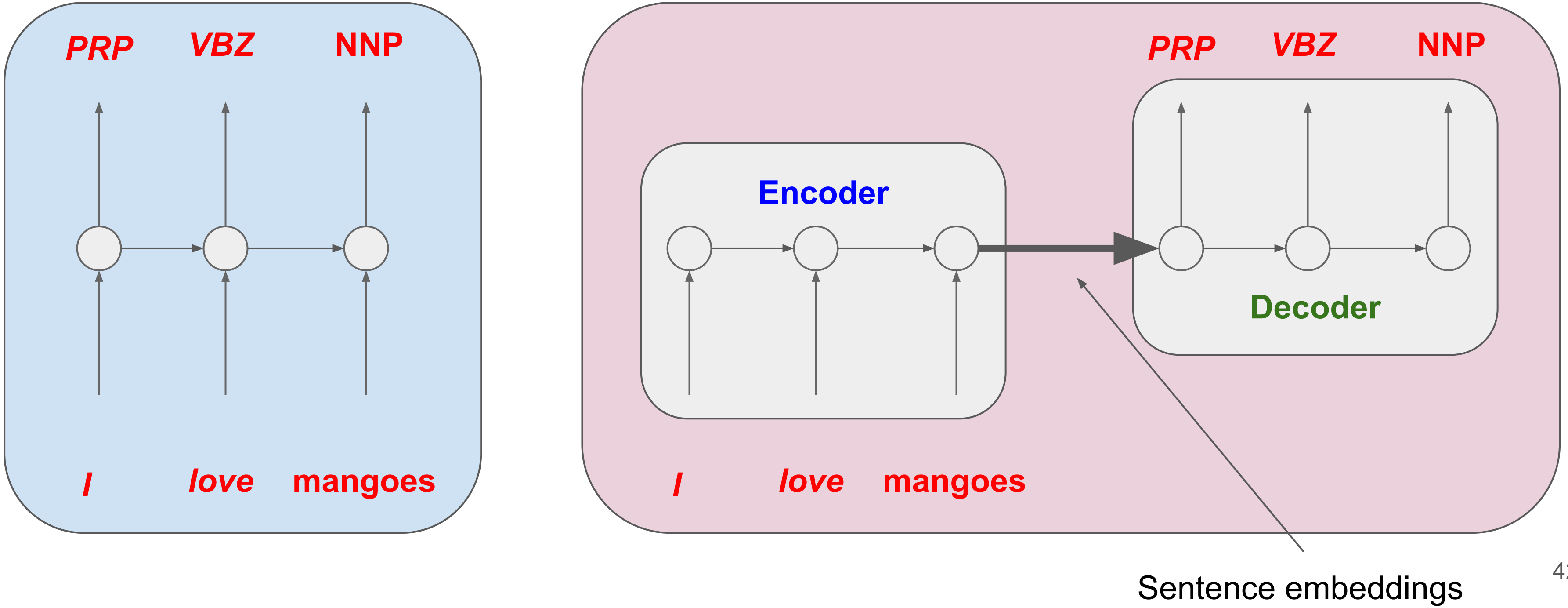
$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Sequence to sequence transformation
with
Attention Mechanism

Sequence labeling v/s Sequence transformation

- PoS Tagging

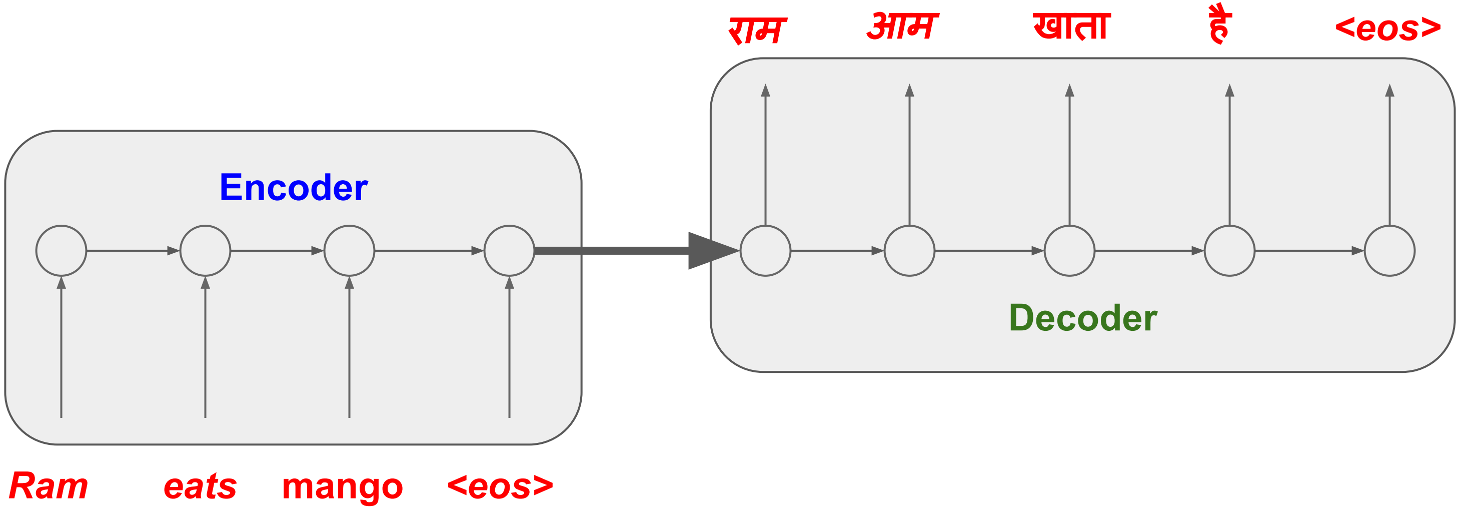


Why sequence transformation is required?

- For many application length of I/p and O/p are not necessarly same. E.g. Machine Translation, Summarization, Question Answering etc.
- For many application length of O/p is not known.
- Non-monotone mapping: Reordering of words.
- Applications like PoS tagging, Named Entity Recognition does not require these capabilities.

Encode-Decode paradigm

- English-Hindi Machine Translation
  - Source sentence: 3 words
  - Target sentecen: 4 words
  - Second word of the source sentence maps to 3rd & 4th words of the target sentence.
  - Third word of the source sentence maps to 2nd word of the target sentence



Problems with Encode-Decode paradigm

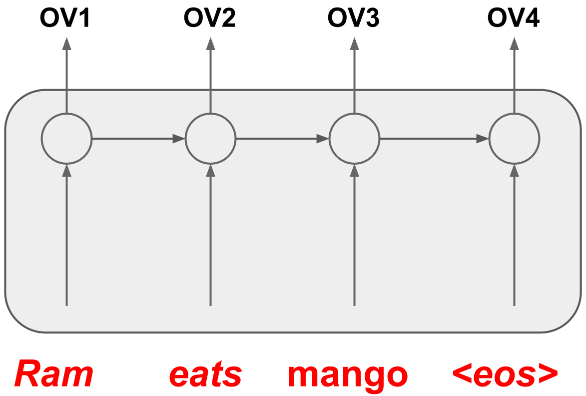
- Encoding transforms the entire sentence into a single vector.
- Decoding process uses this sentence representation for predicting the output.
    - Quality of prediction depends upon the quality of sentence embeddings.
- After few time steps decoding process may not properly use the sentence representation due to long-term dependancy.
- To imporve the quality of predictions we can
    - Improve the quality of sentence embeddings **‘OR’**
    - Present the source sentence represenation for prediction at each time step. **‘OR’**
    - Present the RELEVANT source sentence represenation for prediction at each time step.

Solutions

- To imporve the quality of predictions we can
    - Improve the quality of sentence embeddings **‘OR’**
    - Present the source sentence represenation for prediction at each time step. **‘OR’**
    - Present the RELEVANT source sentence represenation for prediction at each time step.
        - *Encode* - *Attend* - *Decode* (Attention mechanism)

Attention Mechanism

- Represent the source sentence by the set of **output vectors** from the encoder.
- Each **output vector** (OV) at time *t* is a contexual representation of the input at time *t*.

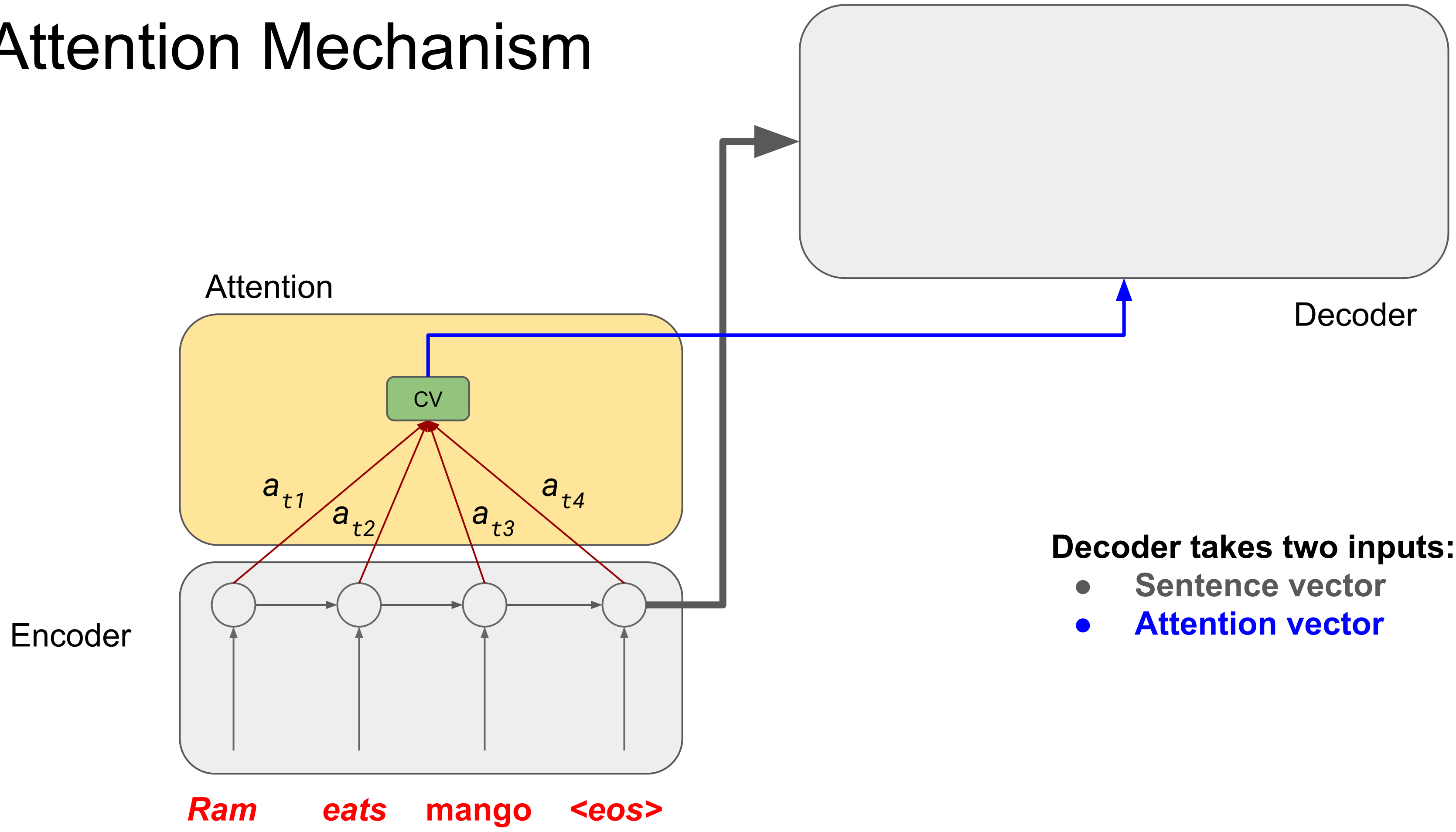


Attention Mechanism

- Each of these output vectors (OVs) may not be equally relevant during decoding process at time *t*.
- Weighted average of the output vectors can resolve the relevancy.
  - Assign more weights to an output vector that needs more ***attention*** during decoding at time *t*.
- The weighted average ***context vector (CV)*** will be the input to decoder along with the sentence representation.
  - $CV_i = \sum a_{ij} \cdot OV_j$

where $a_{ij} = weight\ of\ the\ j^{th}\ OV$

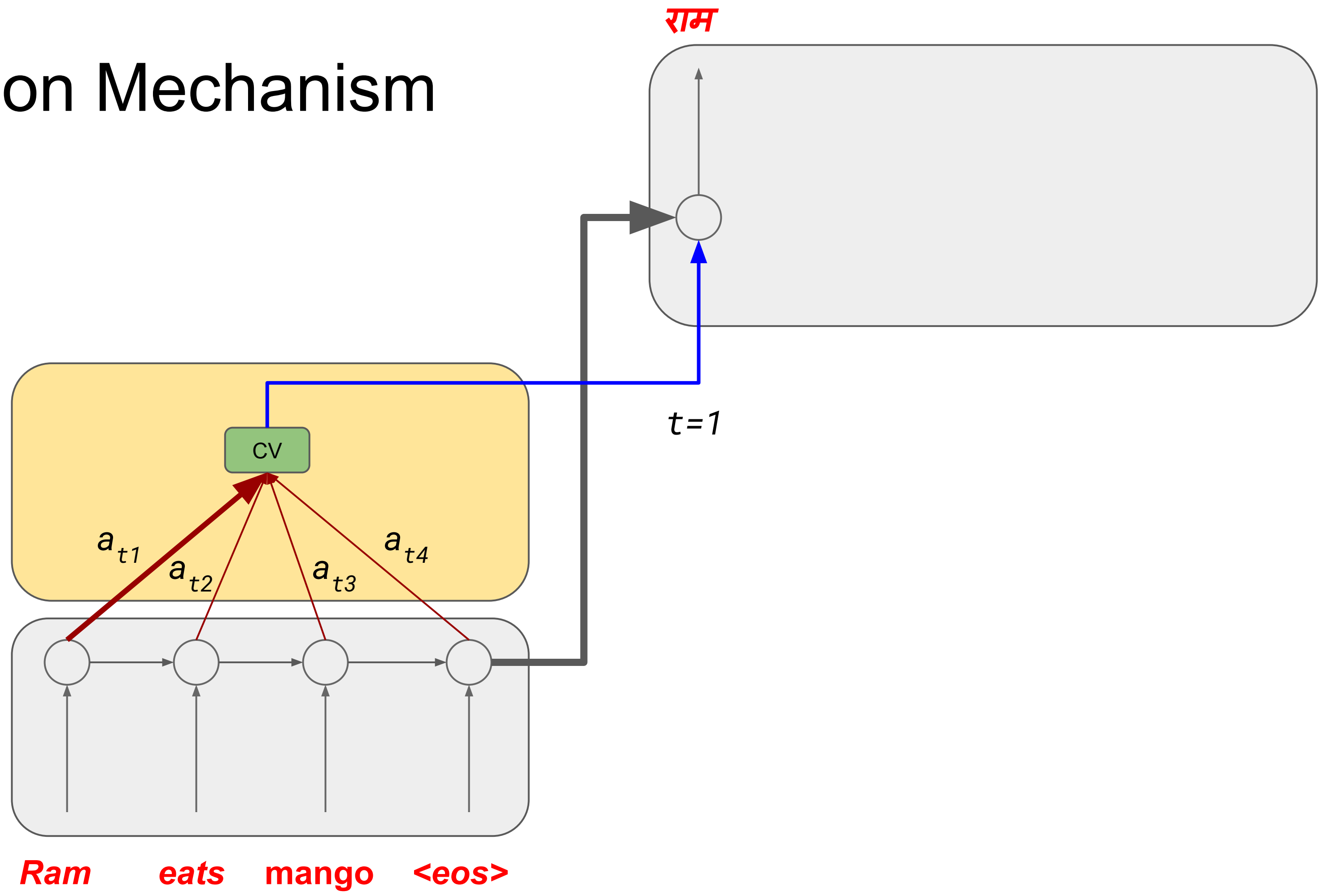
Attention Mechanism



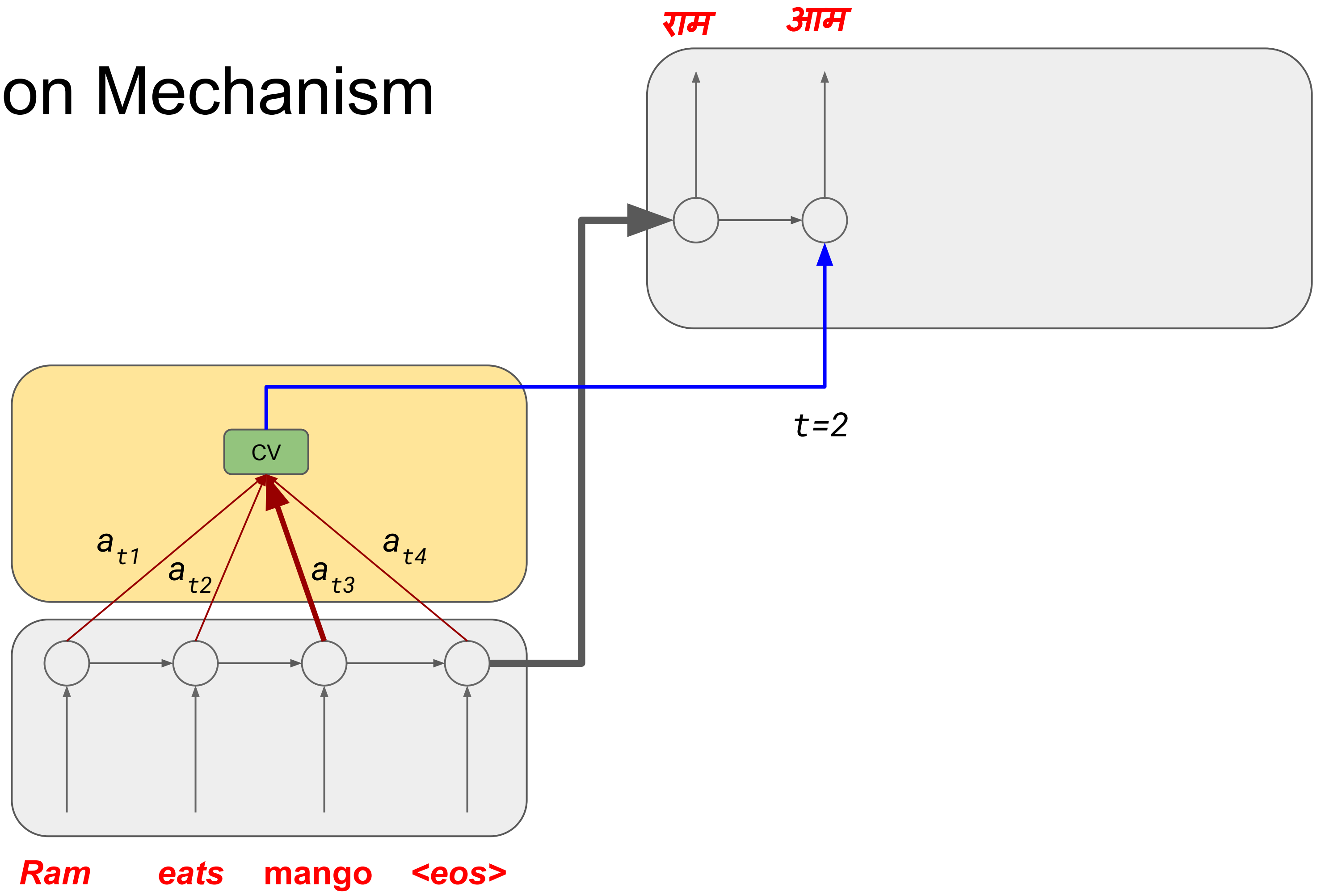
Decoder takes two inputs:

- Sentence vector
- Attention vector

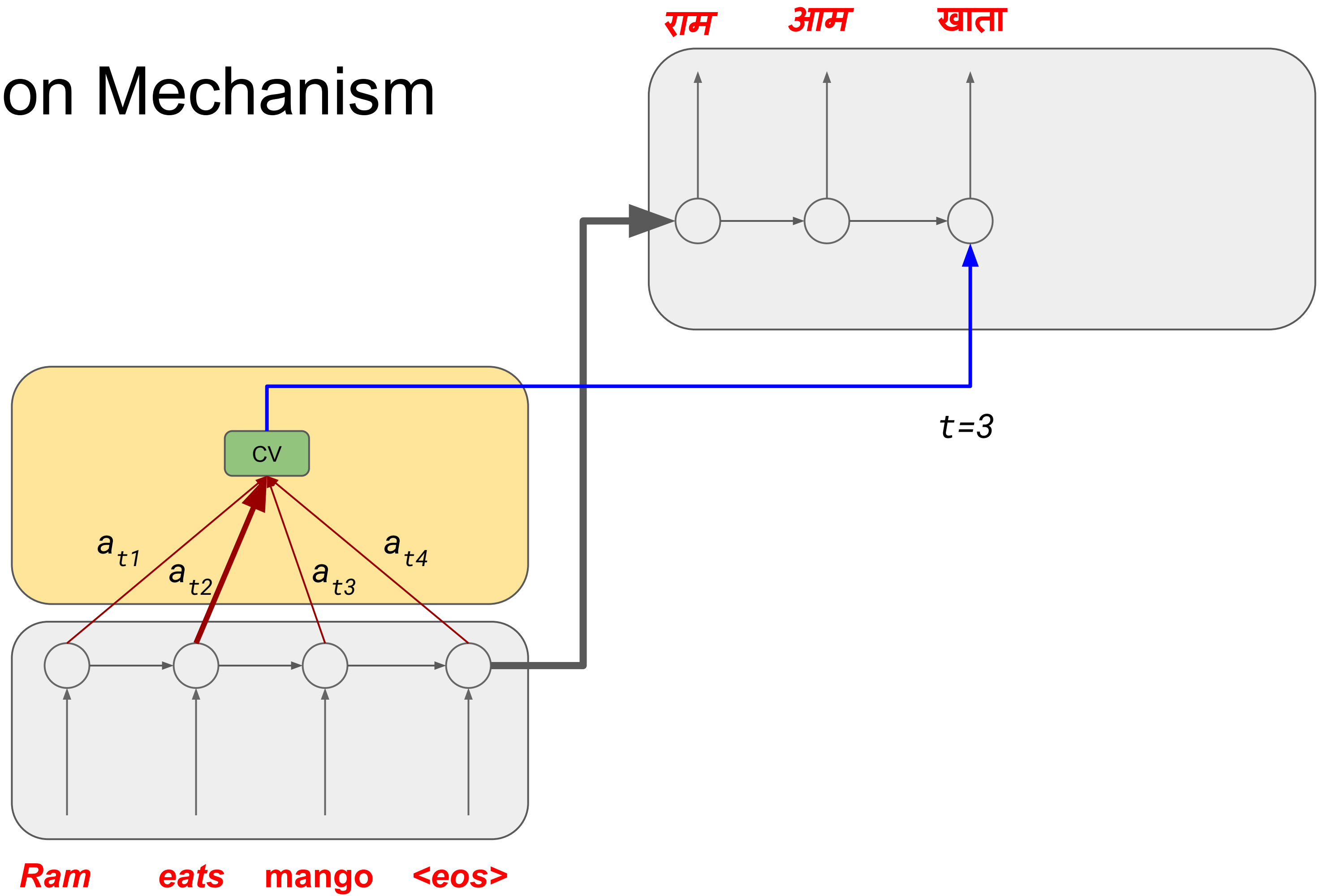
Attention Mechanism



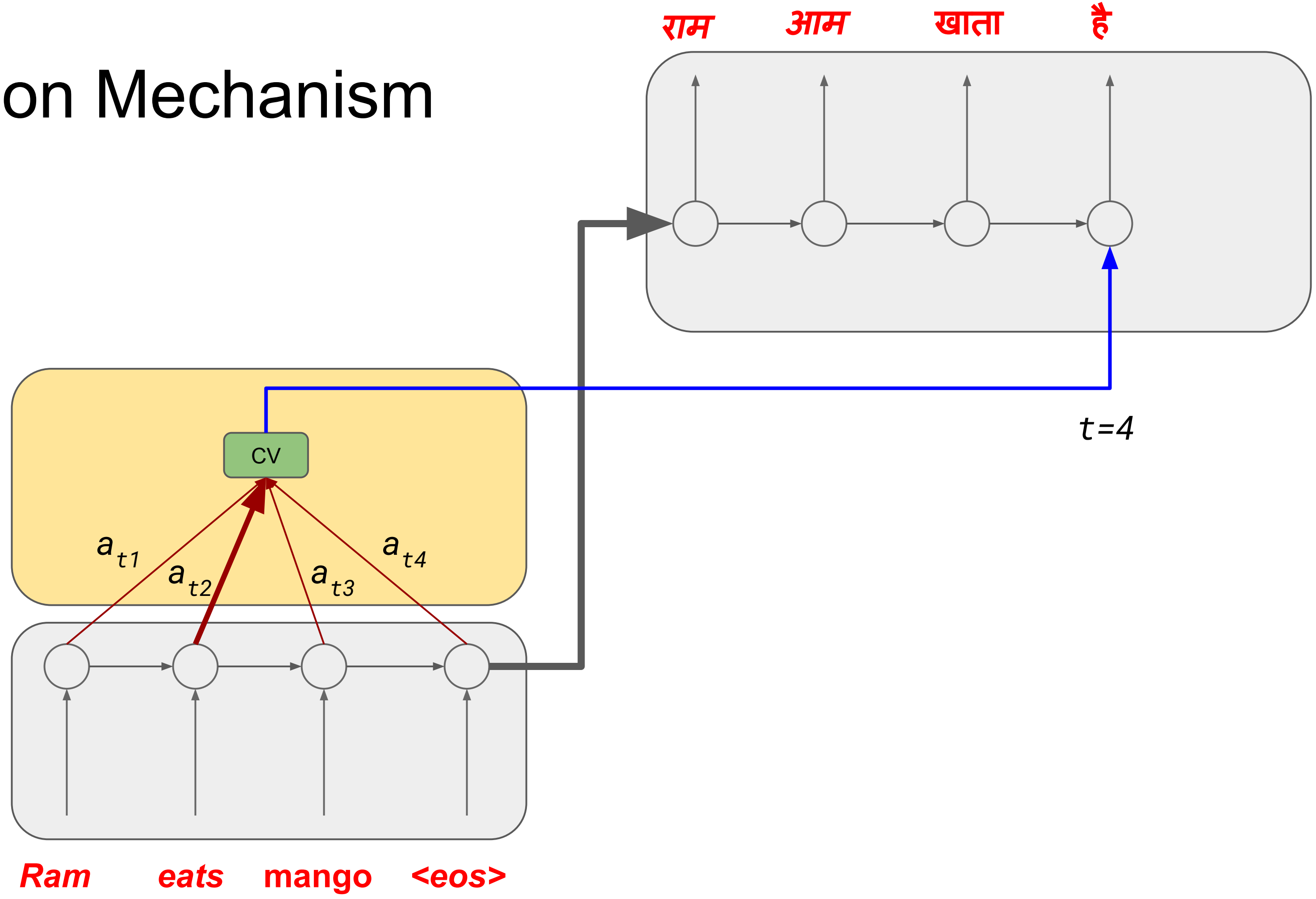
Attention Mechanism



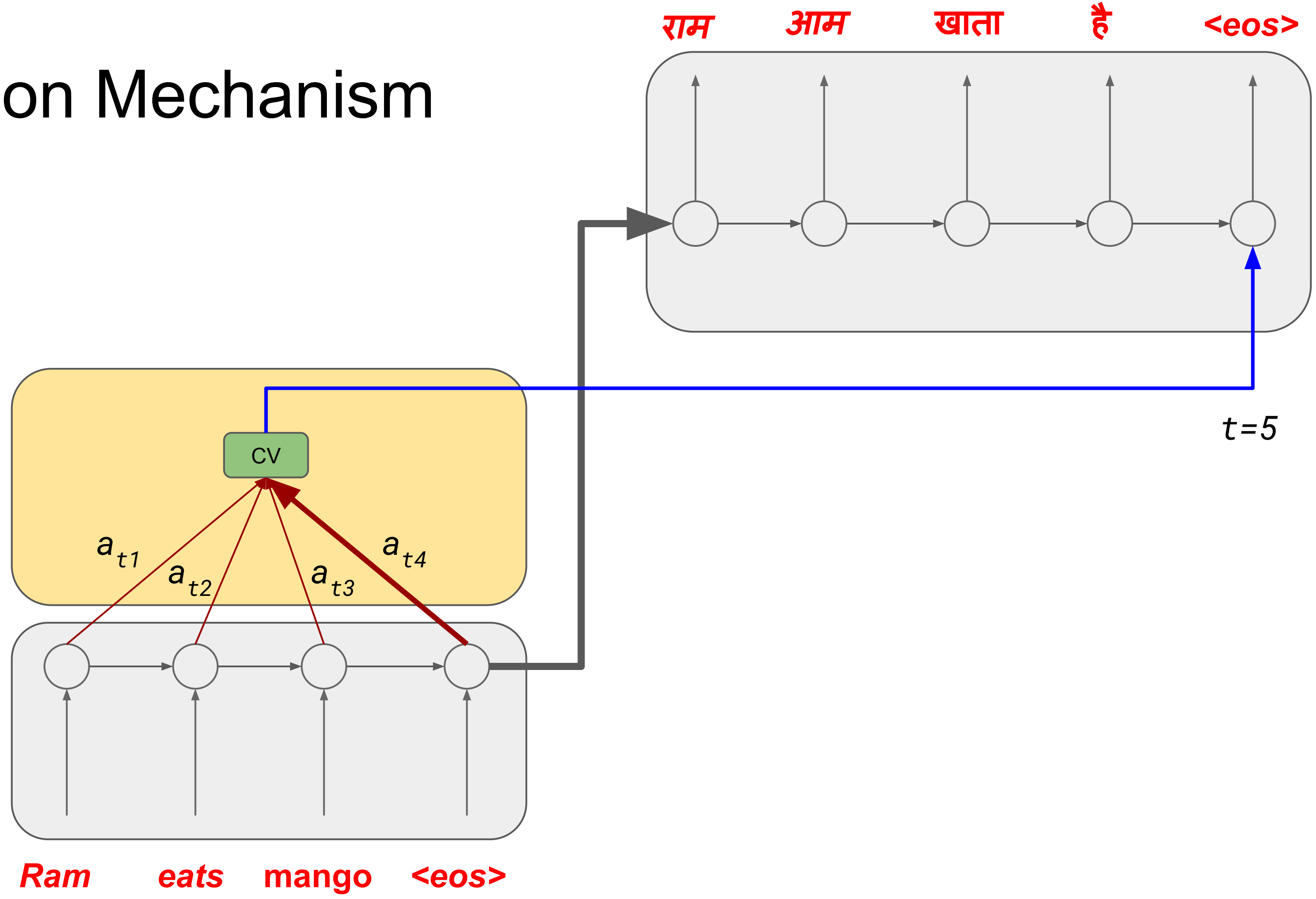
Attention Mechanism



Attention Mechanism



Attention Mechanism



Few good reads..

- Denny Britz; Recurrent Neural Networks Tutorial, Part 1-4
  http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

- Andrej Karpathy; The Unreasonable Effectiveness of Recurrent Neural Networks
  http://karpathy.github.io/2015/05/21/rnn-effectiveness/

- Chris Olah; Understanding LSTM Networks
  http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Thank You!

AI-NLP-ML Group, Department of CSE, IIT Patna (http://www.iitp.ac.in/~ai-nlp-ml/)

Research Supervisors:

- Prof. Pushpak Bhattacharyya
- Dr. Asif Ekbal
- Dr. Sriparna Saha