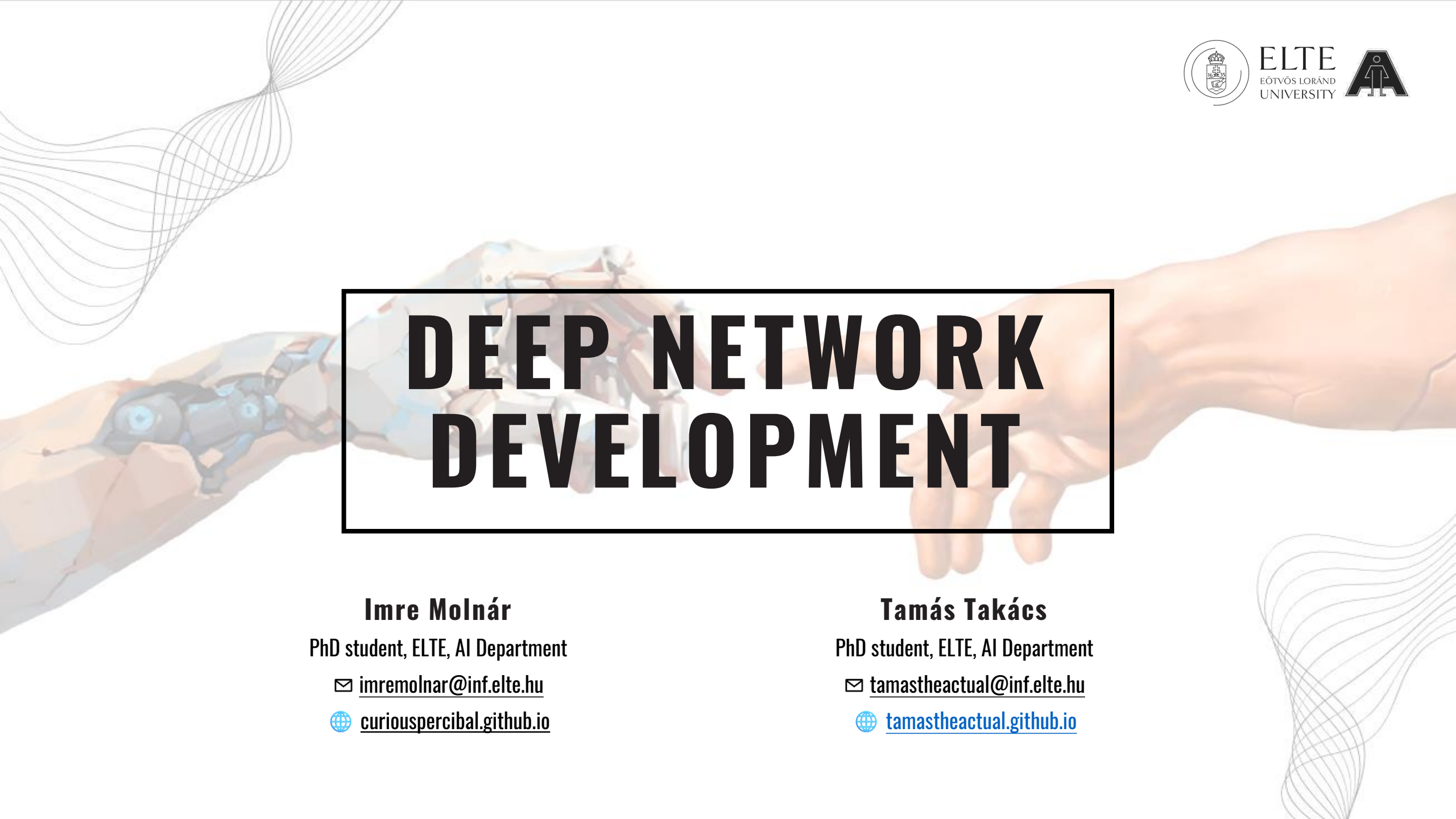


DEEP NETWORK DEVELOPMENT

Imre Molnár

PhD student, ELTE, AI Department

✉ imremolnar@inf.elte.hu

🌐 curiouspercibal.github.io

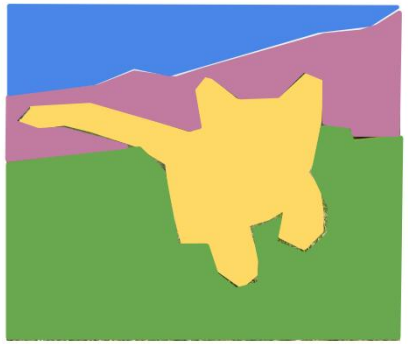
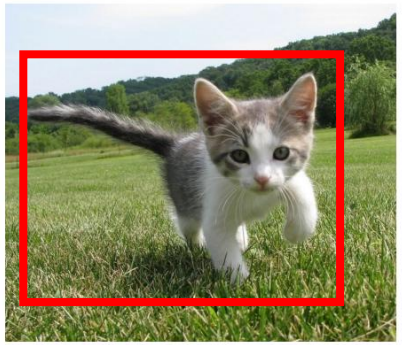
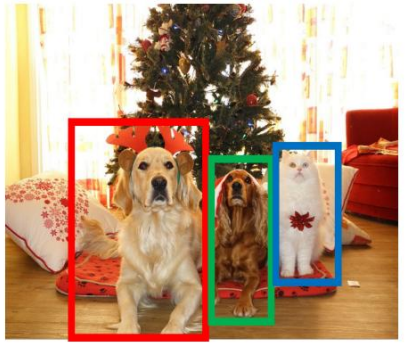
Tamás Takács

PhD student, ELTE, AI Department

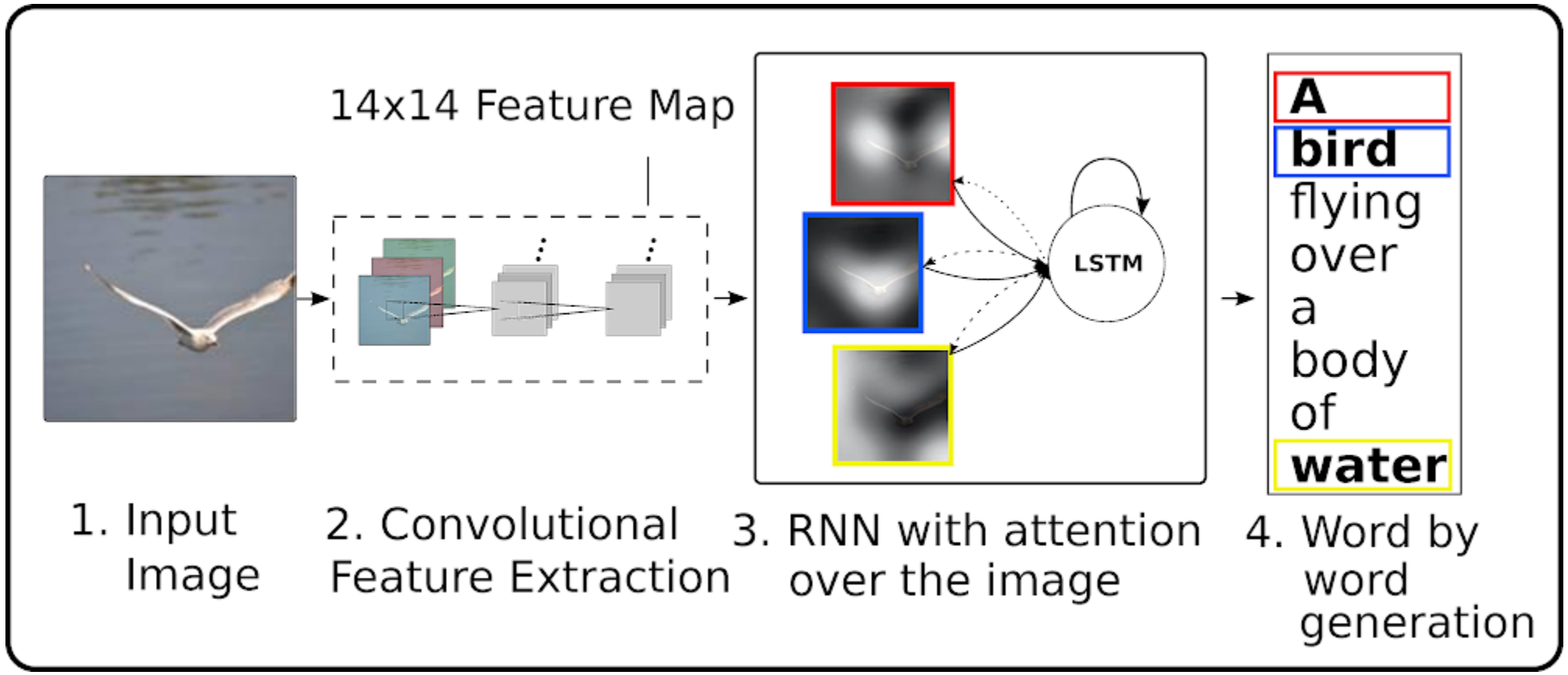
✉ tamastheactual@inf.elte.hu

🌐 tamastheactual.github.io

Supervised Learning tasks

Classification	Semantic Segmentation	Classification + Localization	Object Detection	Instance Segmentation
				
CAT	GRASS, CAT, TREE, SKY	CAT	DOG, DOG, CAT	DOG, DOG, CAT
Single Object	No objects, just pixels	Single Object	Multiple Objects	Multiple Objects
✓	✓	✓	✓	✓

Big Assignment



Deep Network Development

Lecture 7.

Natural Language Processing Basics and Recurrent Neural Networks

Budapest, 4th November 2025

1 NLP Basics

2 RNN, LSTM, GRU & Seq2Seq

3 Attention Mechanism

Sequential Data Processing

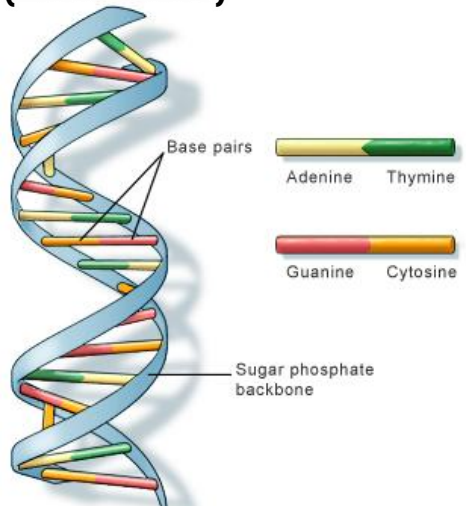
Sequential data

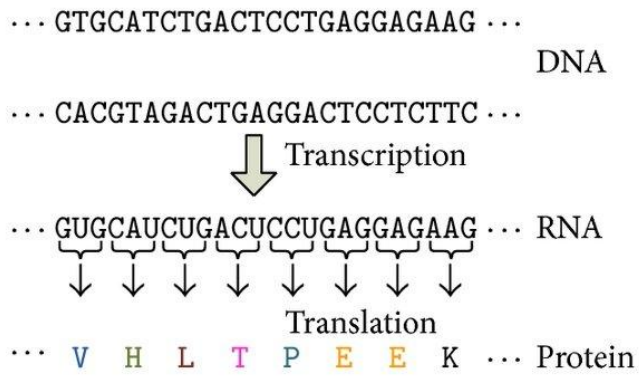
Text – sequence of words / characters

Speech – sequence of signals / acoustic features

Video – sequence of images (frames)

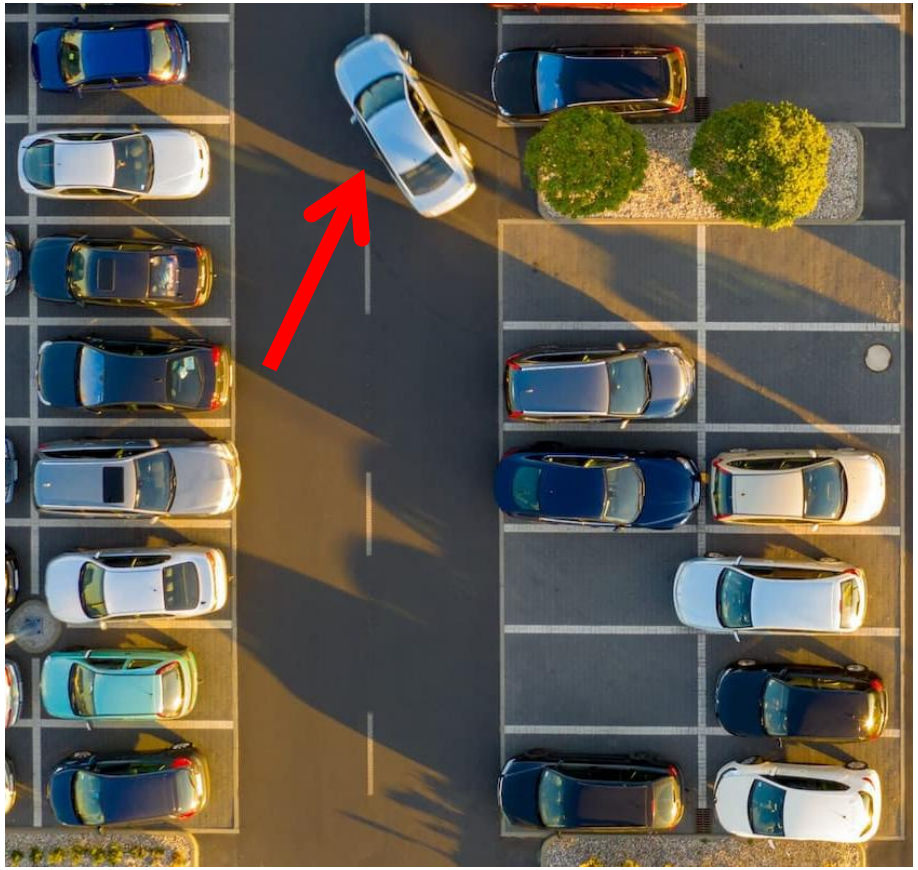
DNA – sequence of symbols (nucleotides)





Sequential Data Processing

Sequential data carry **temporal information** – Is this car parking or leaving?



Sequential Data Processing

Sequential data carry **context**

BANK

“The bank will lend us money.”

"Let's swim to the opposite bank."

Natural Language Processing Tasks

Name Entity Recognition

- A text corpus tagged with 4 different entity types
  - PER (persons), LOC (locations), ORG (organization), MISC (miscellaneous)

Index	0	1	2	3	4	5	6	7	8
Input X:	Harry	Potter	and	Hermione	Granger	invented	a	new	spell
Tags:	B-PER	I-PER	O	B-PER	I-PER	O	O	O	O
Output Y:	1	2	0	1	2	0	0	0	0

Index	0	1	2	3	4	5	6	7	8	9	10
Input X:	The	European	Commission	said	on	Thursday	it	disagreed	with	German	advice
Tags:	O	B-ORG	I-ORG	O	O	O	O	O	O	B-MISC	O
Output Y:	0	3	4	0	0	0	0	0	0	7	0

Natural Language Processing Tasks

Question Answering

- Stanford Question Answering Dataset (SQuAD)
- **Context**
- **Question** – **Answer** pairs
- The answer is a span in a given Wikipedia paragraph

Context	In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under **gravity**. The main forms of precipitation include drizzle, rain, sleet, snow, **graupel** and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals **within a cloud**. Short, intense periods of rain in scattered locations are called "showers".
Question 1	What causes precipitation to fall?
Answer 1	**gravity**
Question 2	What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?
Answer 2	**graupel**
Question 3	Where do water droplets collide with ice crystals to form precipitation?
Answer 3	**within a cloud**

Natural Language Processing Tasks

Textual Entailment

Stanford Natural Language Inference (SNLI)

Determining whether a "**hypothesis**" is true, given a "**premise**".

Five different annotator – **Gold label**

Contradiction - the hypothesis cannot be true given the premise.

Neutral - the hypothesis might be true or false; it's not entailed or contradicted.

Entailment - the hypothesis must be true given the premise.

Text	Judgements	Hypothesis
A man inspects the uniform of a figure in some East Asian country.	contradiction C C C C C	The man is sleeping
An older and younger man smiling.	neutral N N E N N	Two men are smiling and laughing at the cats playing on the floor.
A black race car starts up in front of a crowd of people.	contradiction C C C C C	A man is driving down a lonely road.
A soccer game with multiple males playing.	entailment E E E E E	Some men are playing a sport.
A smiling costumed woman is holding an umbrella.	neutral N N E C N	A happy woman in a fairy costume holds an umbrella.

Natural Language Processing Tasks

Machine Translation

- The name speaks for itself
- BUT, **input – output** pairs in **source – target** language.
- For one input there might be **multiple candidate** translations

English	French
It's cold.	C'est froid.
Stay calm.	Reste tranquille.
Stop that.	Arrêtez ça !
I will submit the assignment tonight	Je rendrai le devoir ce soir.

Other Sequence Processing Tasks

- Speech recognition

→ "The quick brown fox jumped over the lazy dog."

- DNA sequence analysis

AGCCCCTGTGAGGAACTAG → AGCCCCTGTGAGGAACTAG

- Name entity recognition

Yesterday, Harry Potter met Hermione Granger. → Yesterday, Harry Potter met Hermione Granger.

- Sentiment Analysis
- Semantic role labelling
- Conference resolution
- Etc.

Sequential Data Processing

Representing words as One hot vectors

Dataset: (X,Y)

Vocabulary: [a, aron, …, harry, …, potter, …, zulu]

Position: 1, 2, …, 4075, …., 6883, …., 10000

Index	0	1	2	3	4	5	6	7	8
Input X:	Harry	Potter	and	Hermione	Granger	invented	a	new	spell
Position:	4075	6883	2	…	…	…	1	…	…
Representation	$\begin{bmatrix}0\\0\\0\\\vdots\\1\\0\\\vdots\\0\end{bmatrix}$	$\begin{bmatrix}0\\0\\0\\\vdots\\0\\1\\\vdots\\0\end{bmatrix}$	$\begin{bmatrix}0\\1\\0\\\vdots\\0\\0\\\vdots\\0\end{bmatrix}$	…	…	…	$\begin{bmatrix}1\\0\\0\\\vdots\\\vdots\\\vdots\\\vdots\\0\end{bmatrix}$	…	…

Embeddings

Representing words as One hot vectors

Input X: My favorite sport is football.

$X^{<1>}$ $X^{<2>}$ $X^{<3>}$ $X^{<4>}$ $X^{<5>}$

Vocabulary: [favorite, football, is, my, sport]

Position: 1 2 3 4 5

Representation:

Football = [0, 1, 0, 0, 0]

Position: 1 2 3 4 5

Sport = [0, 0, 0, 0, 1]

Position: 1 2 3 4 5

Problems

- Scalability - huge vector for each word
  - If we have a dataset of several sentences, from which we form a vocabulary of 10 000 words.
  - Each word would be represented as a 10 000 long vector, having a single element set to 1. $X^{<1>}$ = [0 , 0, ..., 1, ..., 0, 0]
- There is **no relationship between words**. Each word is treated as an independent entity with no similarity to other words.

Embeddings: Word representation

Featurized representation: Word Embedding

Vocabulary size: 10 000

Vocabulary: [a, ..., apple, ..., football, ..., man, ..., orange, ..., sport, ..., woman, ..., zulu]

Position: 1 456 2078 5391 6257 7301 9853 10 000

	Man (5391)	Woman (9853)	King (4914)	Queen (7151)	Apple (456)	Orange (6257)
Gender	**-1**	**1**	**-0.95**	**0.97**	0.00	0.01
Royal	0.01	0.02	**0.93**	**0.95**	-0.01	0.00
Age	0.03	0.02	**0.7**	**0.68**	0.03	-0.02
Food	0.04	0.01	0.02	0.01	**0.95**	**0.97**
...	...	...	...	...	...	...

Embeddings: Word representation

Featurized representation: Word Embedding

Man (5391) instead of being represented a one hot encoded vector [0,0,...,1,...,0,0] would be represented as:

e_{5391}=[-1, 0.01, 0.03, 0.04, ...]

Embedding Matrix **X** one hot man = embedding man

(# features , vocab size) (vocab size, 1) (# features, 1)

	Man (5391)	Woman (9853)	King (4914)	Queen (7151)	Apple (456)	Orange (6257)
Gender	**-1**	**1**	**-0.95**	**0.97**	0.00	0.01
Royal	0.01	0.02	**0.93**	**0.95**	-0.01	0.00
Age	0.03	0.02	**0.7**	**0.68**	0.03	-0.02
Food	0.04	0.01	0.02	0.01	**0.95**	**0.97**
...	...	...	...	...	...	...

Embeddings: Word representation

Featurized representation: Word Embedding

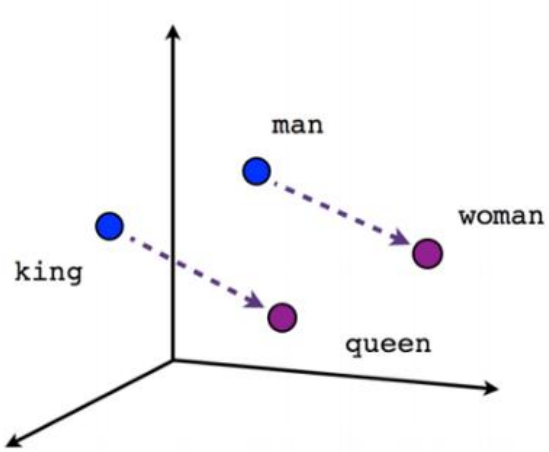
If we subtract man and woman, main difference is gender

We can compute word similarities

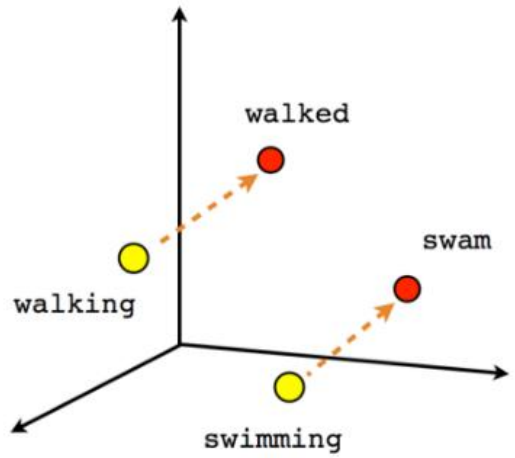
We can compute word analogies: man is to woman as king is to

https://vectors.nlpl.eu/explore/embeddings/en/calculator/

	Man (5391)	Woman (9853)	King (4914)	Queen (7151)	Apple (456)	Orange (6257)
Gender	**-1**	**1**	**-0.95**	**0.97**	0.00	0.01
Royal	0.01	0.02	**0.93**	**0.95**	-0.01	0.00
Age	0.03	0.02	**0.7**	**0.68**	0.03	-0.02
Food	0.04	0.01	0.02	0.01	**0.95**	**0.97**
…	…	…	…	…	…	…



Male-Female



Verb tense

Embeddings

How to create better embeddings?

- We learn the cooccurrences of the words (they appear in similar contexts)
- Let's create input – output pairs as follows
- Given an example sentence

"A person is playing with a cat on the couch."

- We create a list of words that occurs around the cat
- Then we use the one-hot encoding of the words

Skip-grams

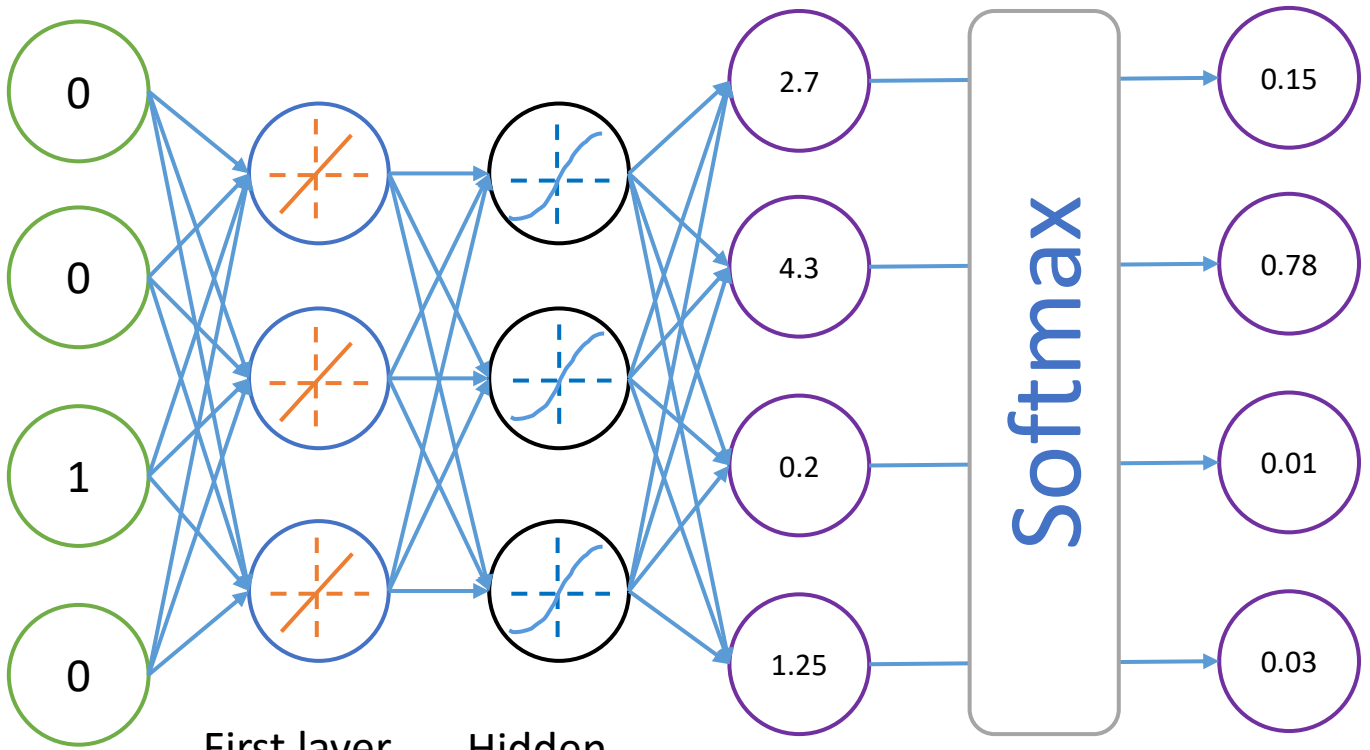
Input	Output
Cat	On
Cat	The
…	…
Cat	Couch
Cat	Person
Cat	Playing

Embeddings

How to create better embeddings?

Compare output with the Outputs (**Person**) one-hot encoded vector

Cat one-hot encoded vector

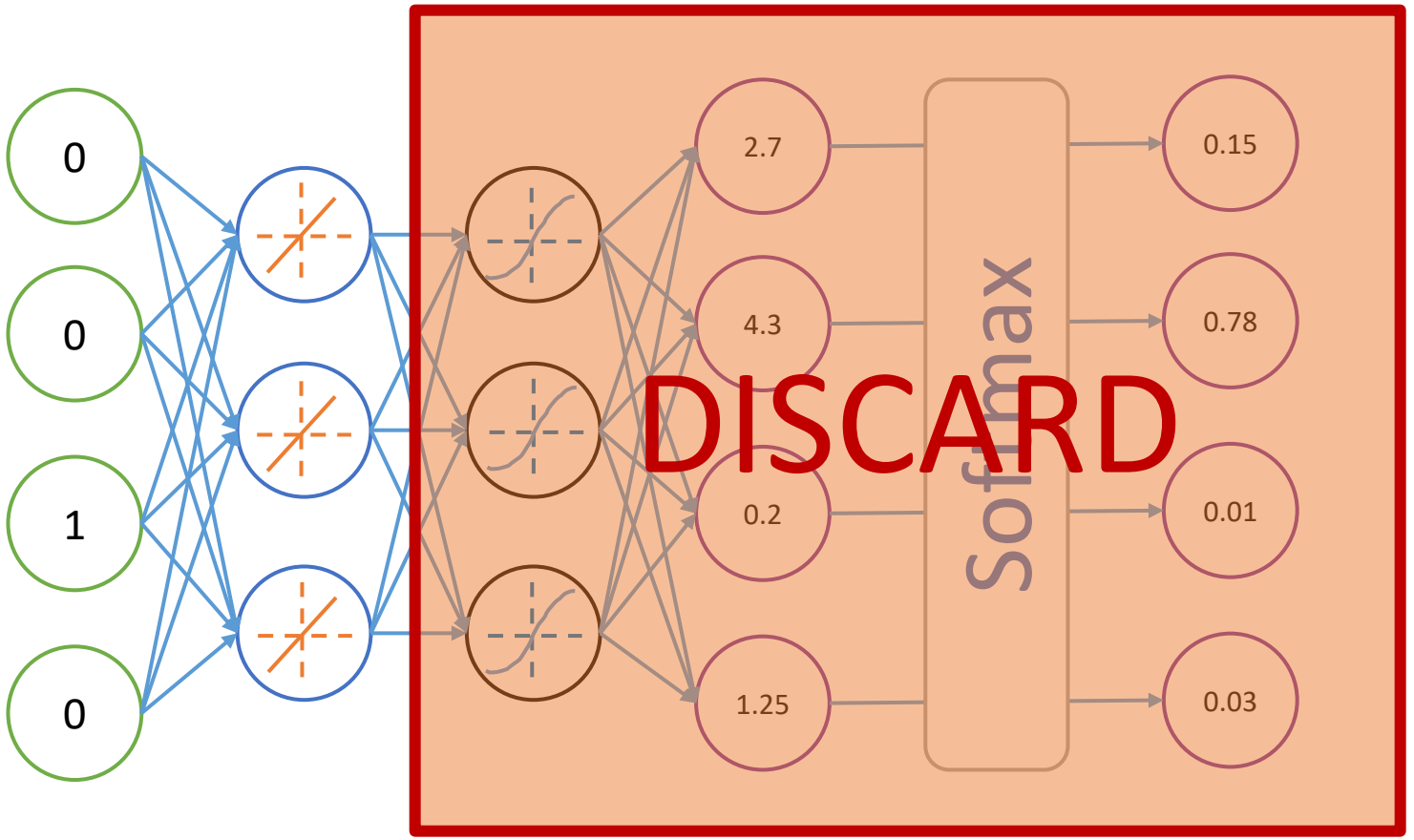


Input	Output
Cat	On
Cat	The
…	…
Cat	Couch
Cat	Person
Cat	Playing

Embeddings

How to create better embeddings?

- Keep only the first layer (thus creating a linear projection of the input vector)



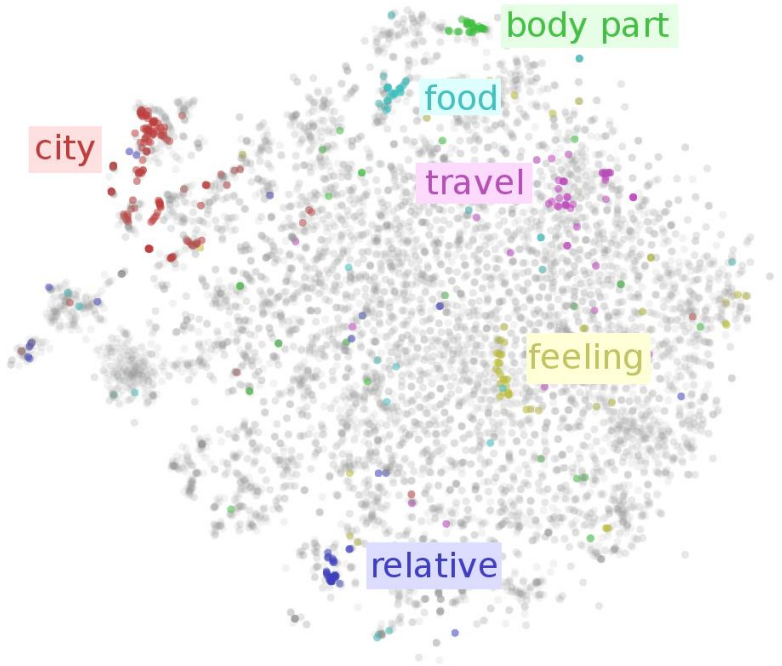
Embeddings: Word representation

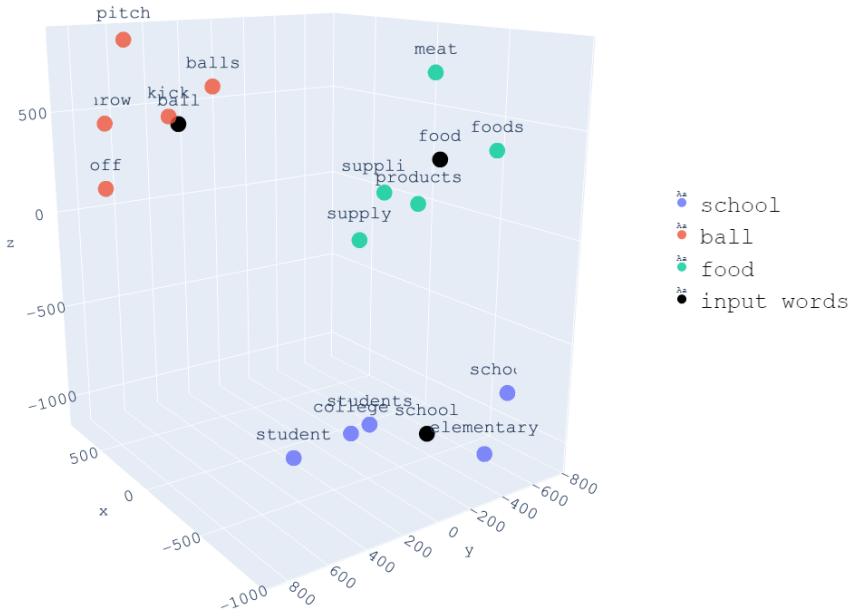
Featurized representation: Word Embedding

The word embeddings are learned with training.

Therefore, in practice, the features aren't that understandable.

We can visualize lower representations of the embeddings with techniques such as T-SNE





Embeddings: Word representation

Embeddings can be used to represent other types of data:

Speech: speaker embeddings

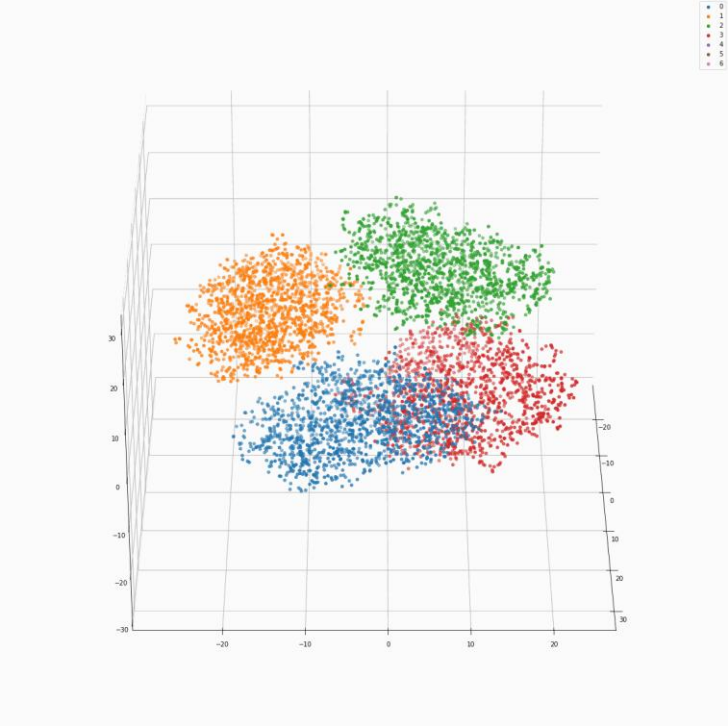
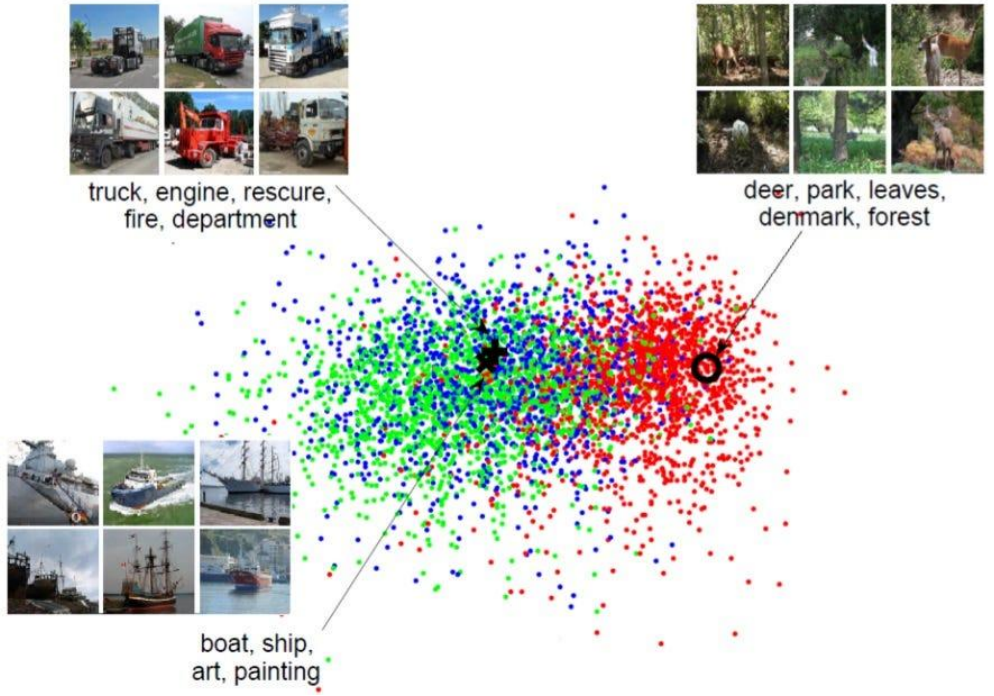


Image: image embeddings



Deep Network Development

Lecture 7.

Natural Language Processing Basics and Recurrent Neural Networks

Budapest, 4th November 2025

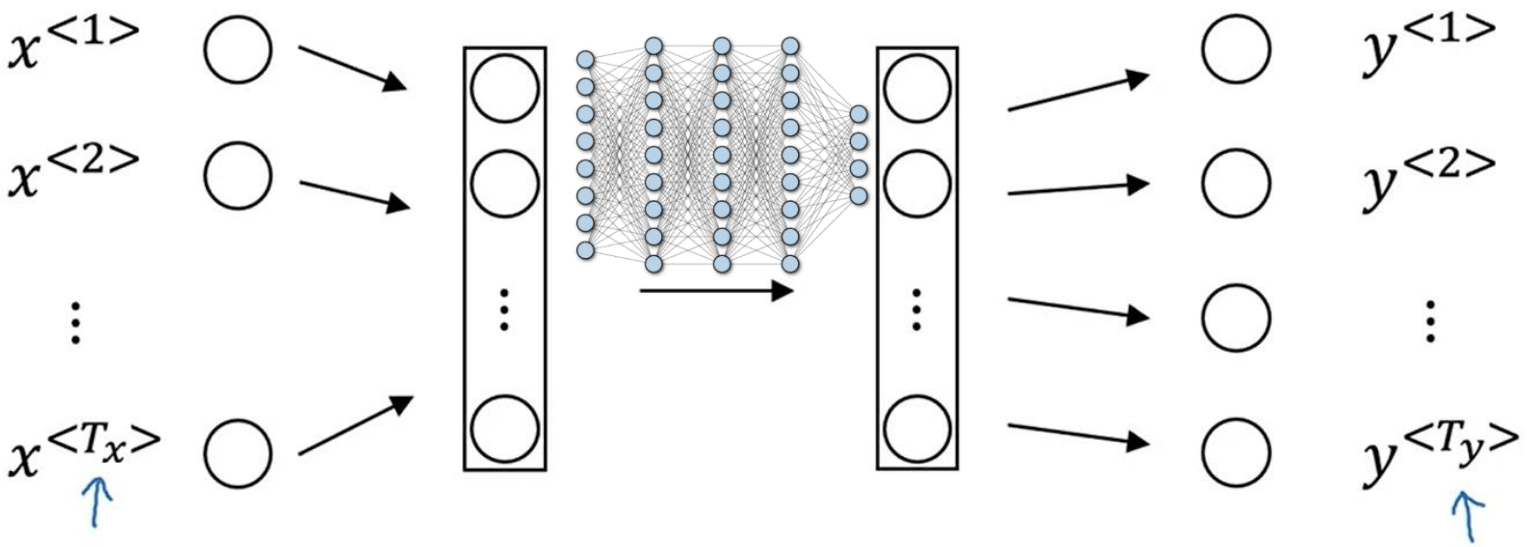
1 NLP Basics

2 RNN, LSTM, GRU & Seq2Seq

3 Attention Mechanism

Introduction to Recurrent Neural Network

Why not a standard network?



Problems:

- Inputs, outputs can be different lengths in different examples
- Doesn’t share features learned across different positions of text

Introduction to Recurrent Neural Network

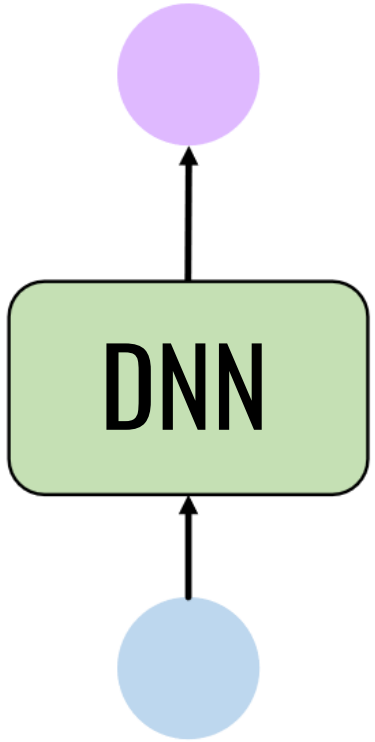
What should we consider?

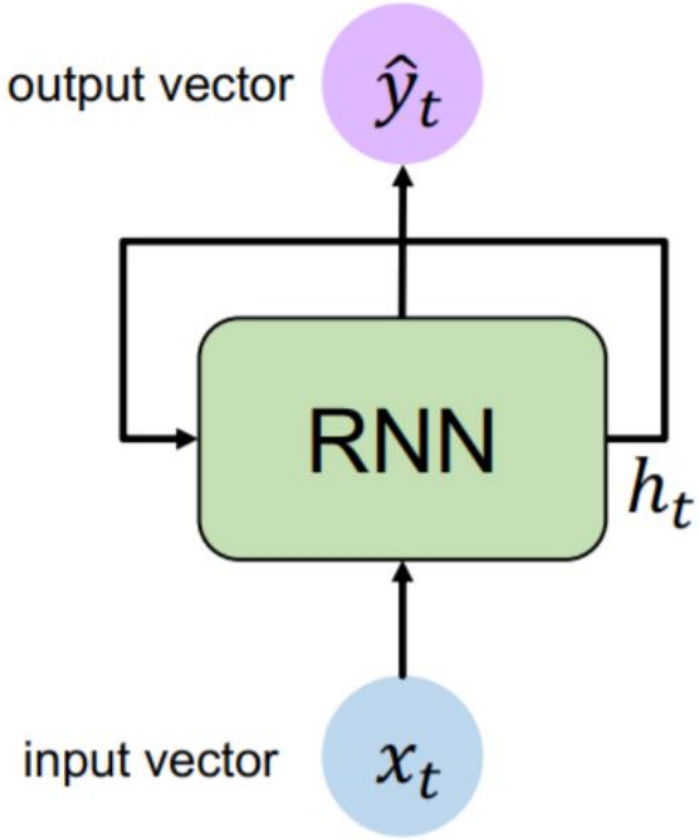
The model needs to:

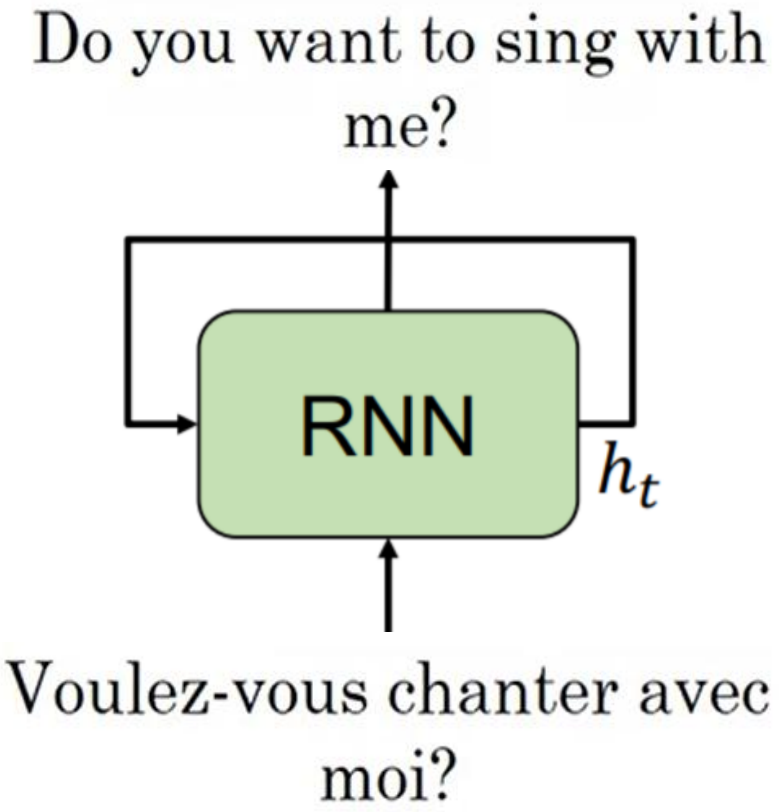
- Handle variable length sequences
- Track long term dependencies
- Maintain the order of the input
- Share parameters across sequences

Introduction to Recurrent Neural Network

What about Recurrent Neural Networks (RNNs)?

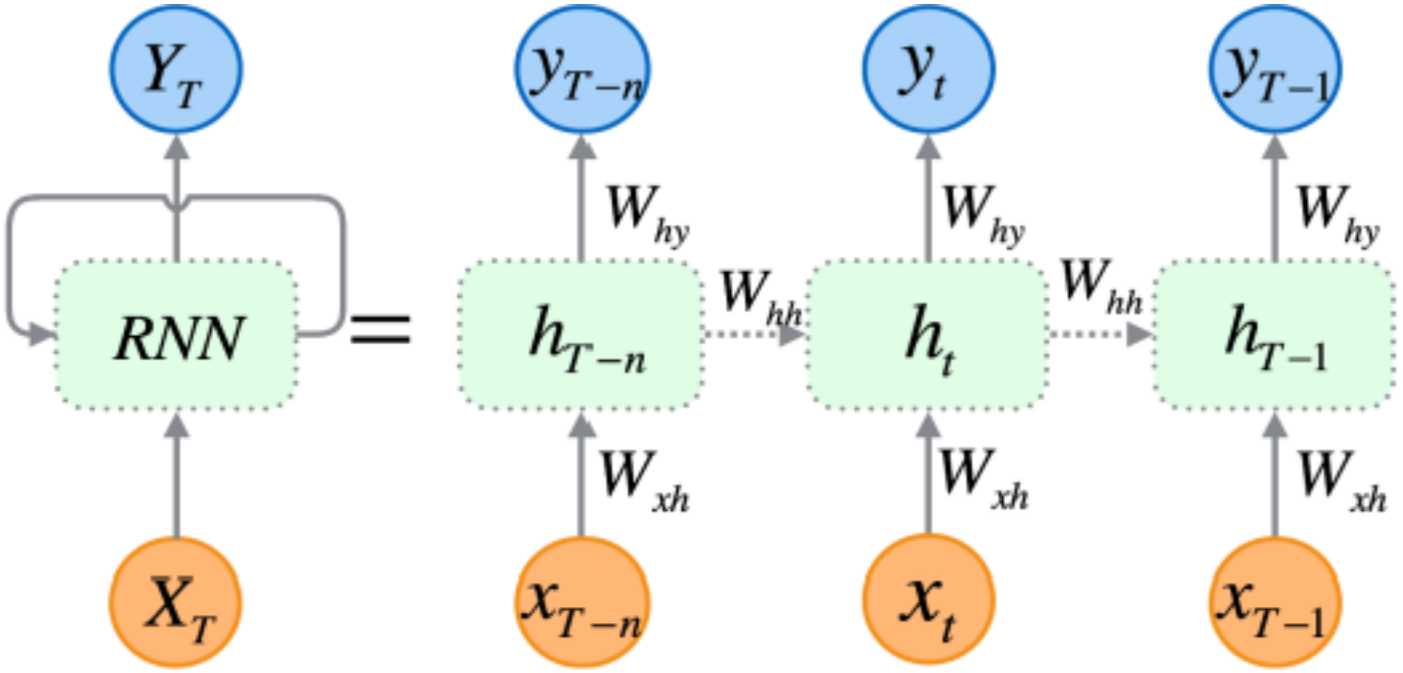






Introduction to Recurrent Neural Network

RNN (unrolled version)



Hidden state:

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

Output vector:

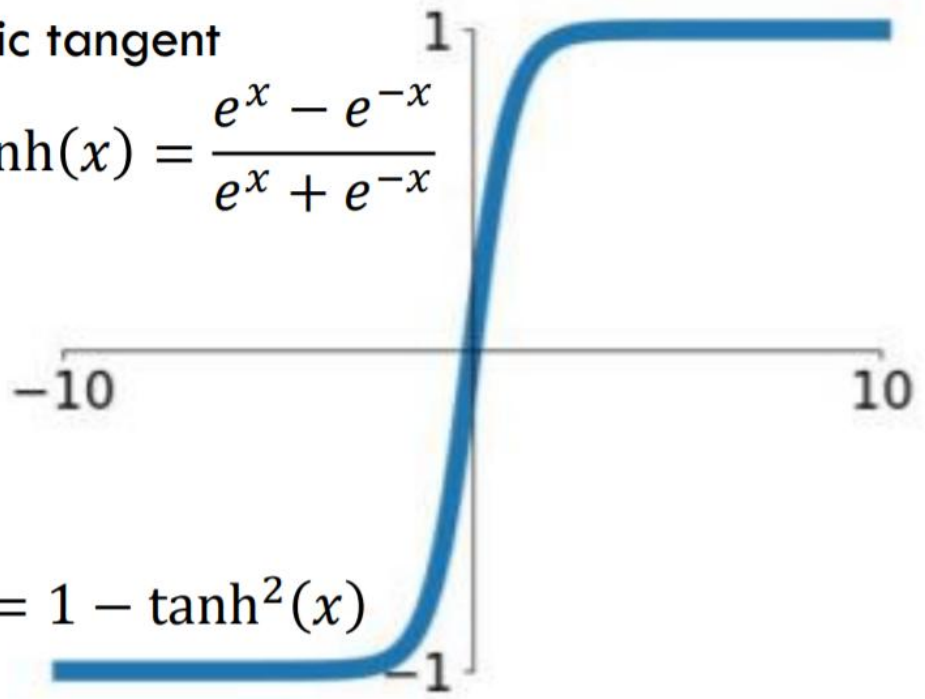
$$\hat{y}_t = tanh(W_{hy}h_t + b_y)$$

Introduction to Recurrent Neural Network

RECAP: ACTIVATION FUNCTION – TANGENT

Hyperbolic tangent

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$



$$\tanh'(x) = 1 - \tanh^2(x)$$

[LeCun et al., 1991]

Pros:

- Zero centered.
- Activations are bounded in range [-1,1].
- The gradient is stronger for tanh than sigmoid.
  Derivatives are steeper.

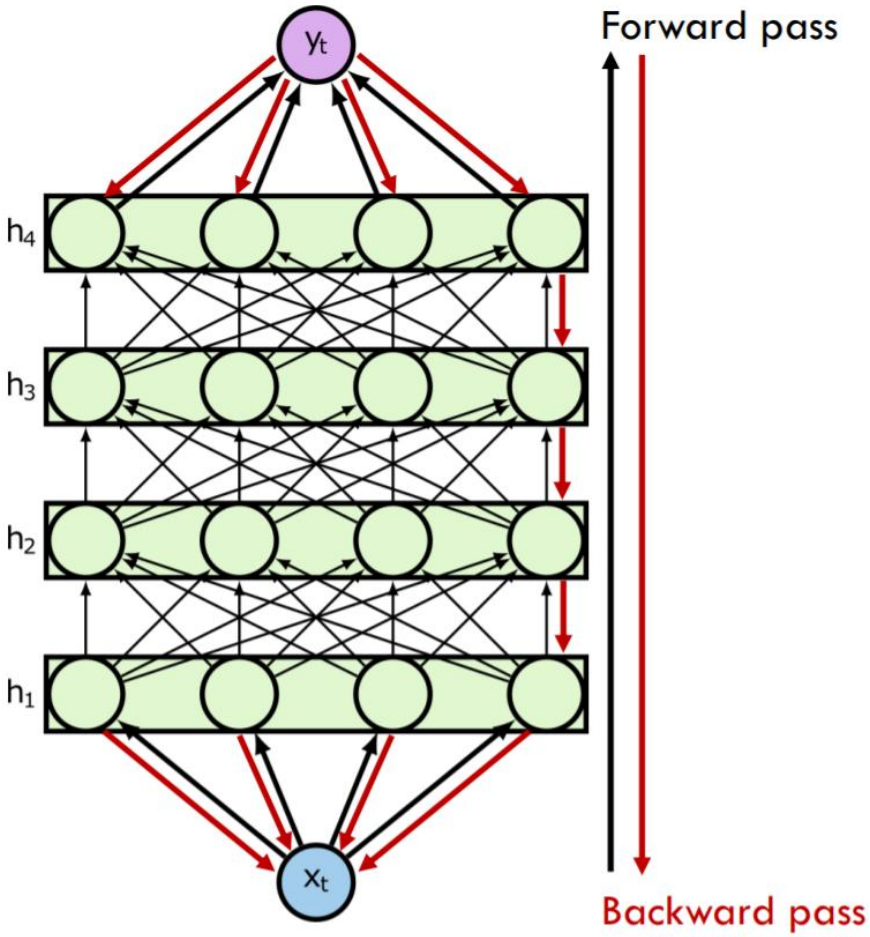
Cons:

- Y values tend to respond very less to changes in X towards either end of the function.
- Vanishing gradients.

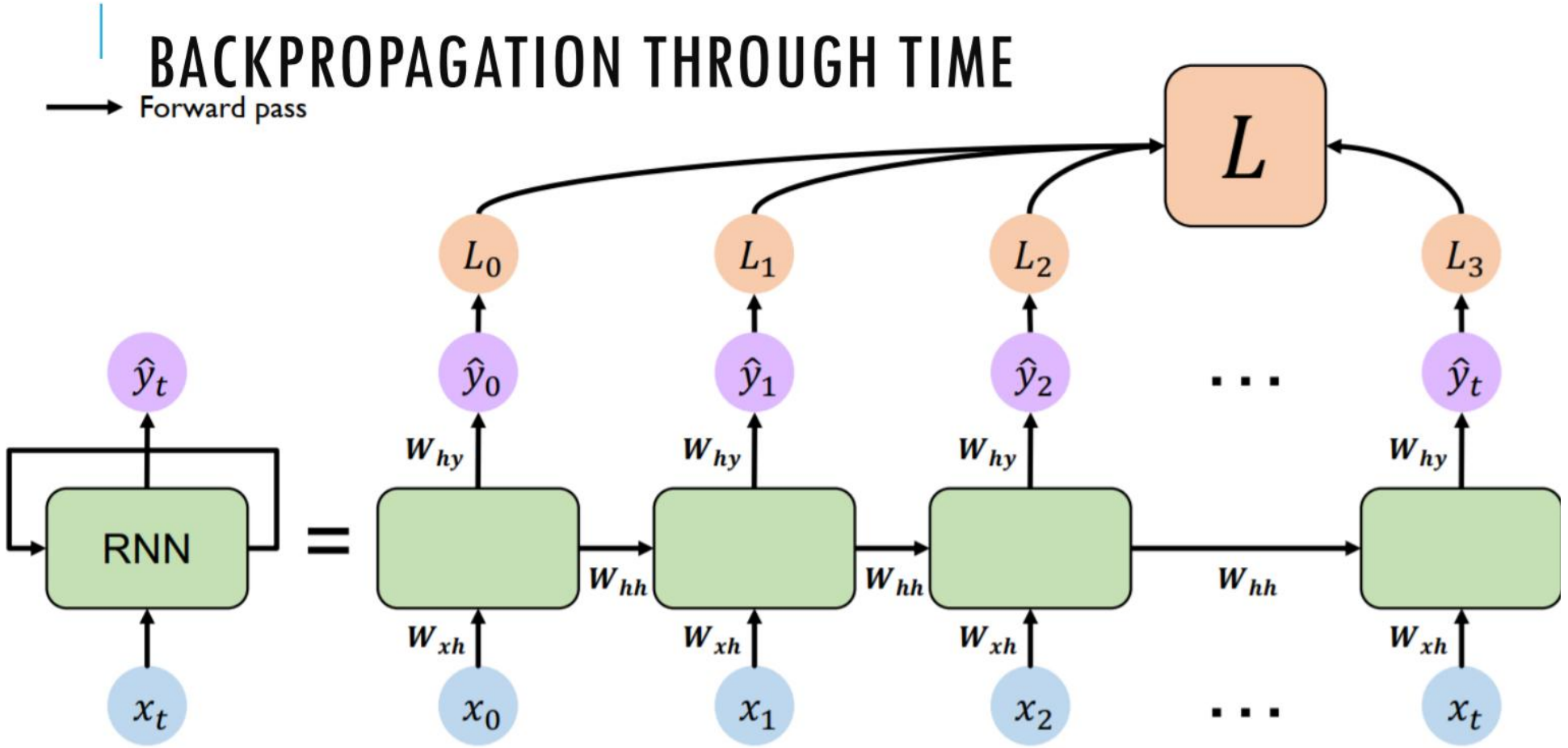
Introduction to Recurrent Neural Network

Backpropagation

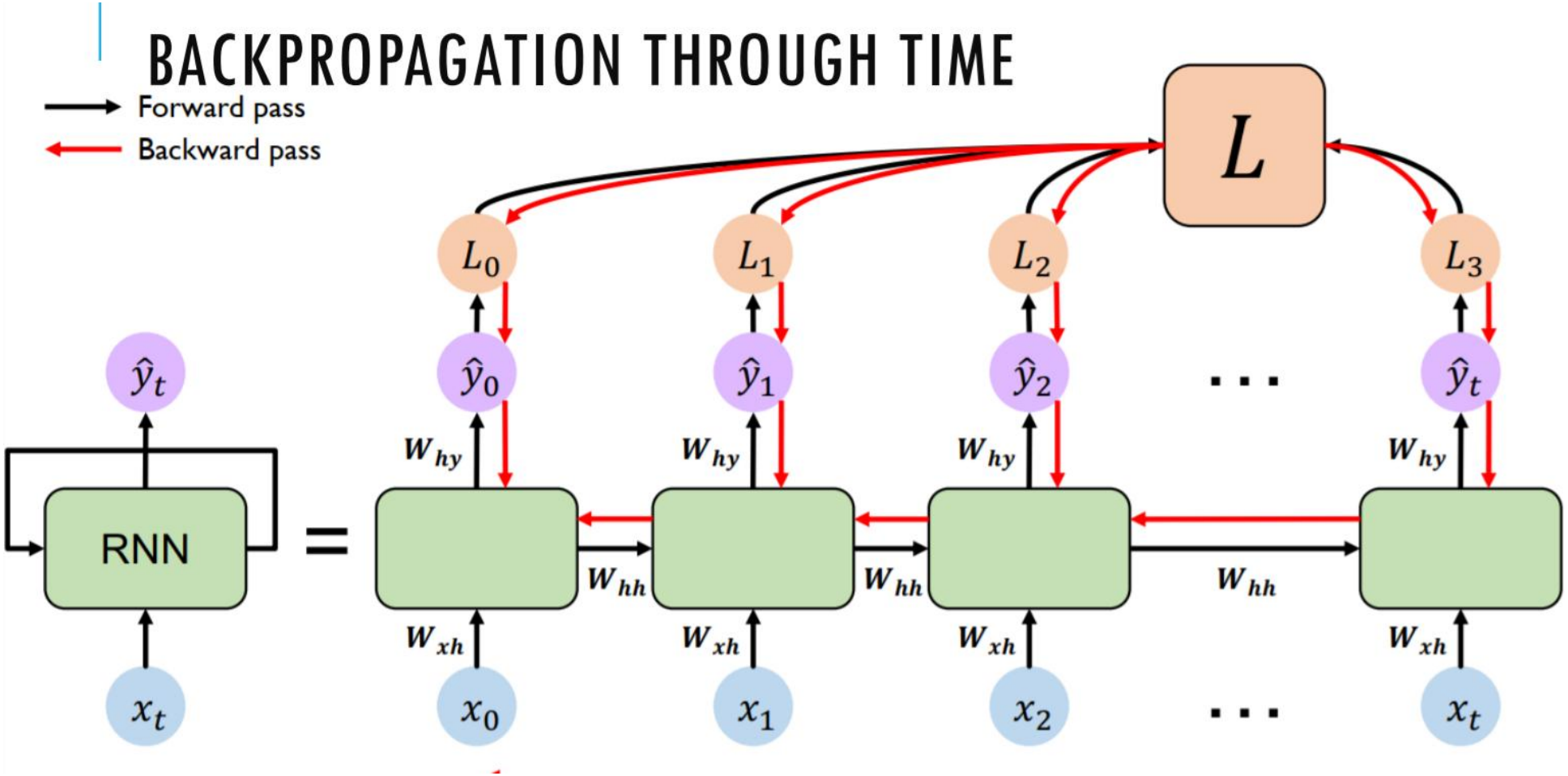
1. Calculate the forward pass
2. Determine the loss
3. Take the partial derivative (gradient) of the loss respect to each parameter
4. Shift the parameters to minimize the loss



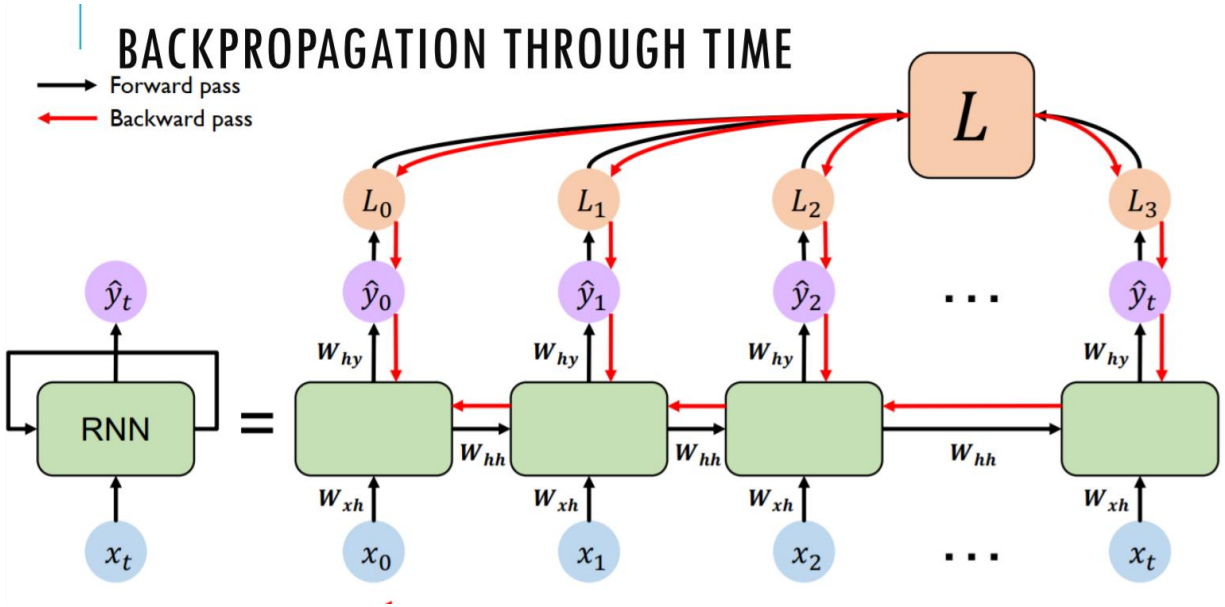
Introduction to Recurrent Neural Network



Introduction to Recurrent Neural Network



Introduction to Recurrent Neural Network



Hidden state:

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

Output vector:

$$\hat{y}_t = tanh(W_{hy}h_t + b_y)$$

Loss:

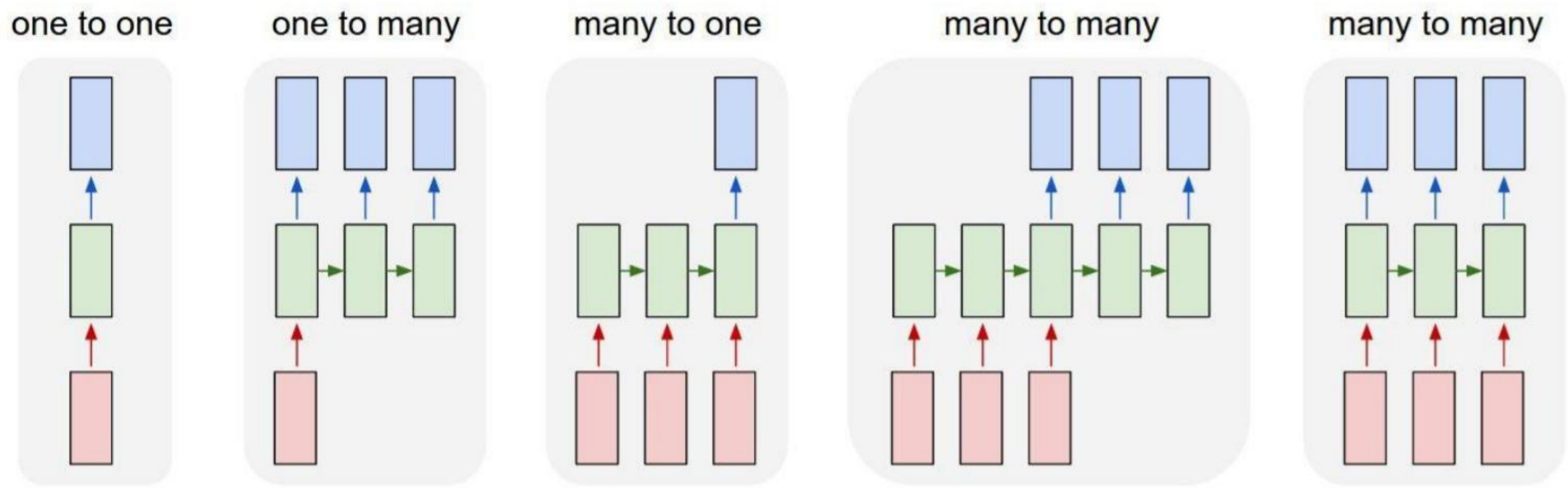
$$L(x_1, \dots, x_T, y_1, \dots, y_T, W_{xh}, W_{hh}, W_{hy}) = \frac{1}{T}\sum_{t=1}^{T} l(y_t, \hat{y}_t)$$

Gradients:

$$\frac{\partial L}{\partial W_{hh}} = \frac{1}{T}\sum_{t=1}^{T} \frac{\partial l(y_t, \hat{y}_t)}{\partial W_{hh}}$$

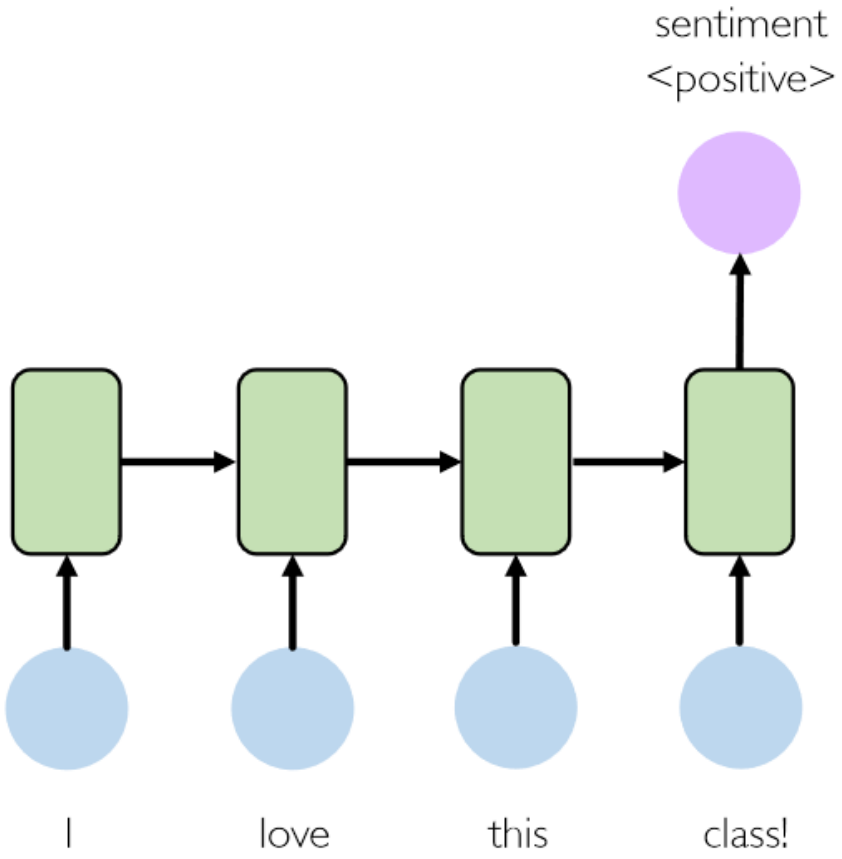
Tasks using RNNs

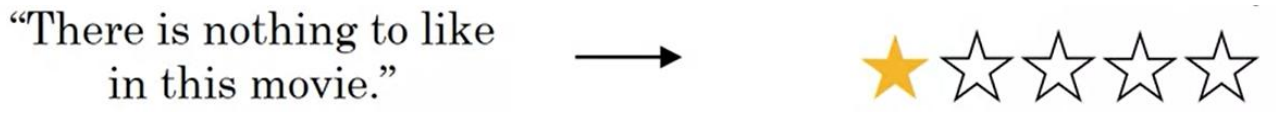
RNNS COME IN MANY FORMS

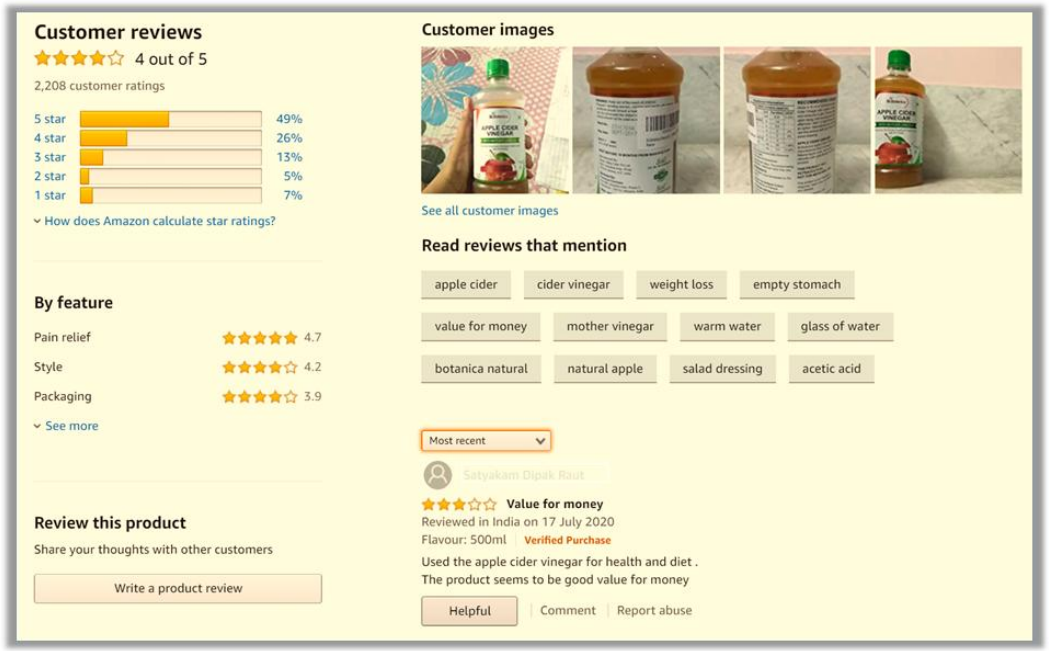


Tasks using RNNs

RNN Many to One – Sentiment Analysis

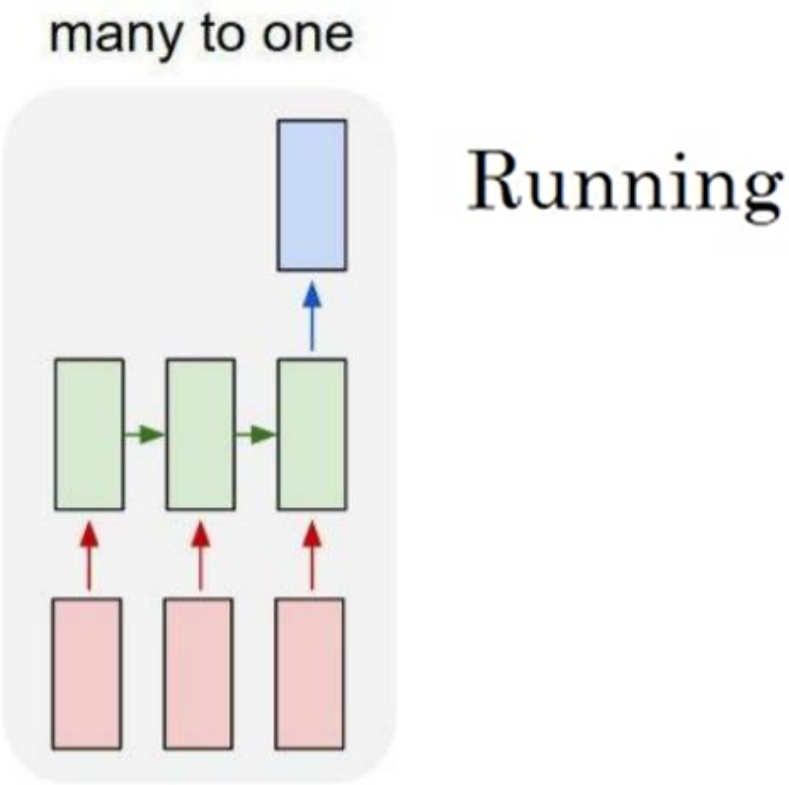






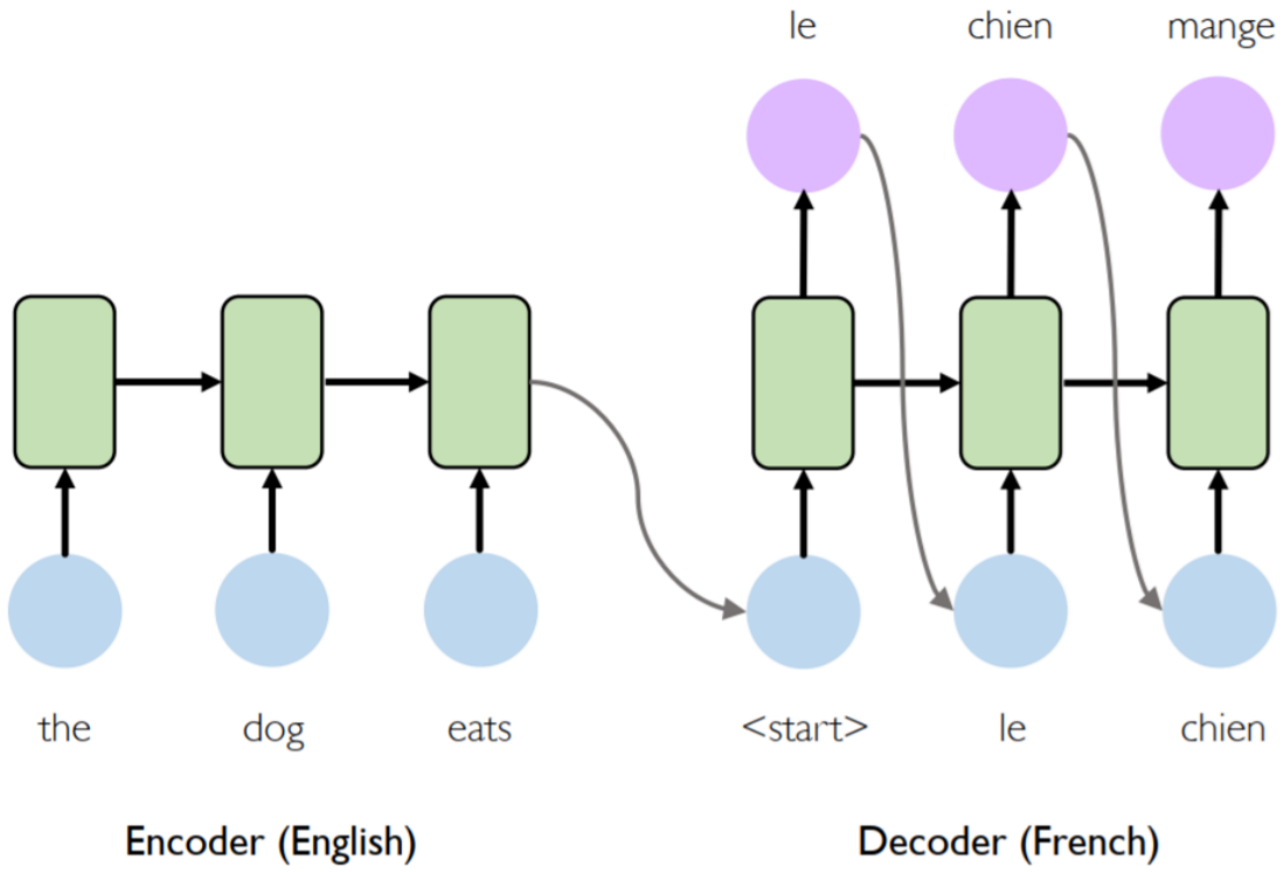
Tasks using RNNs

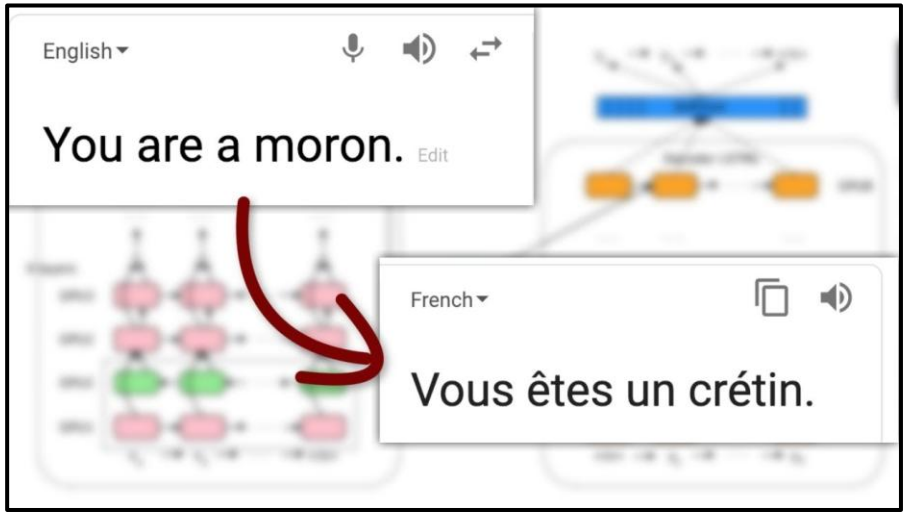
RNN Many to One – Video Activity Recognition



Tasks using RNNs

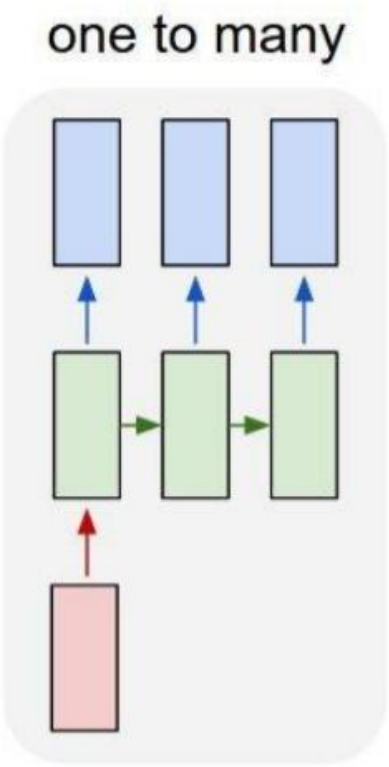
RNN Many to Many – Machine Translation

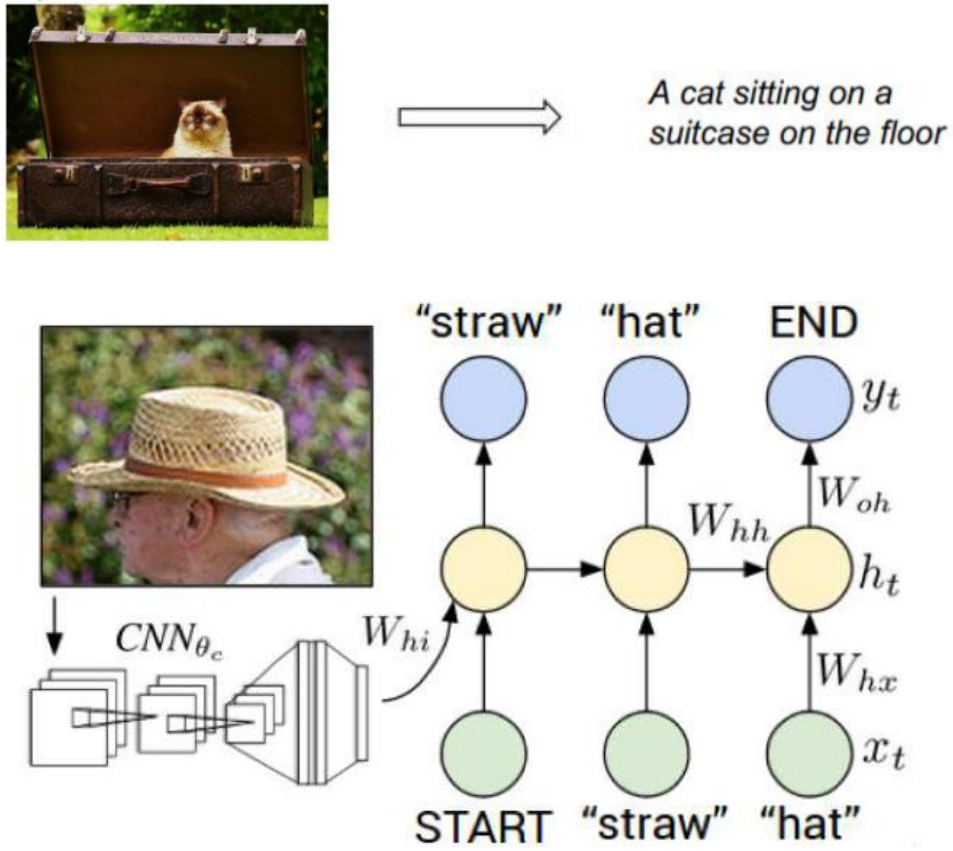




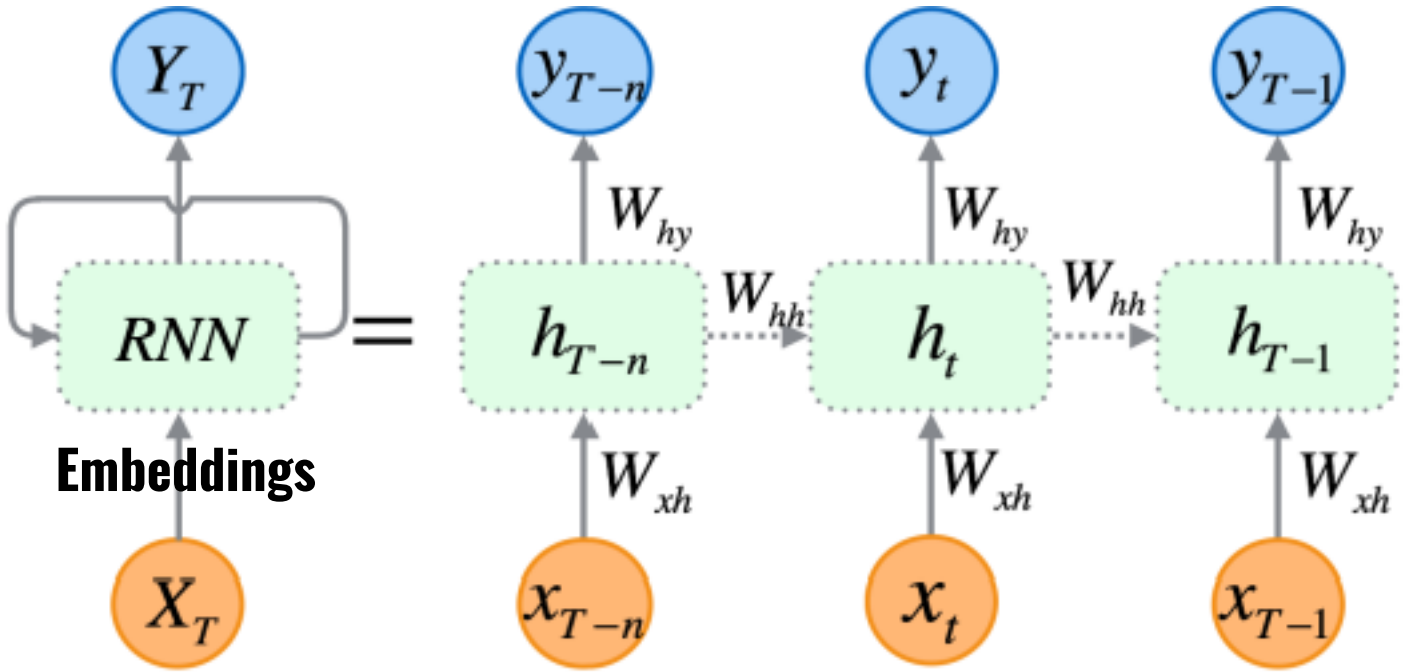
Tasks using RNNs

RNN One to Many – Image Caption

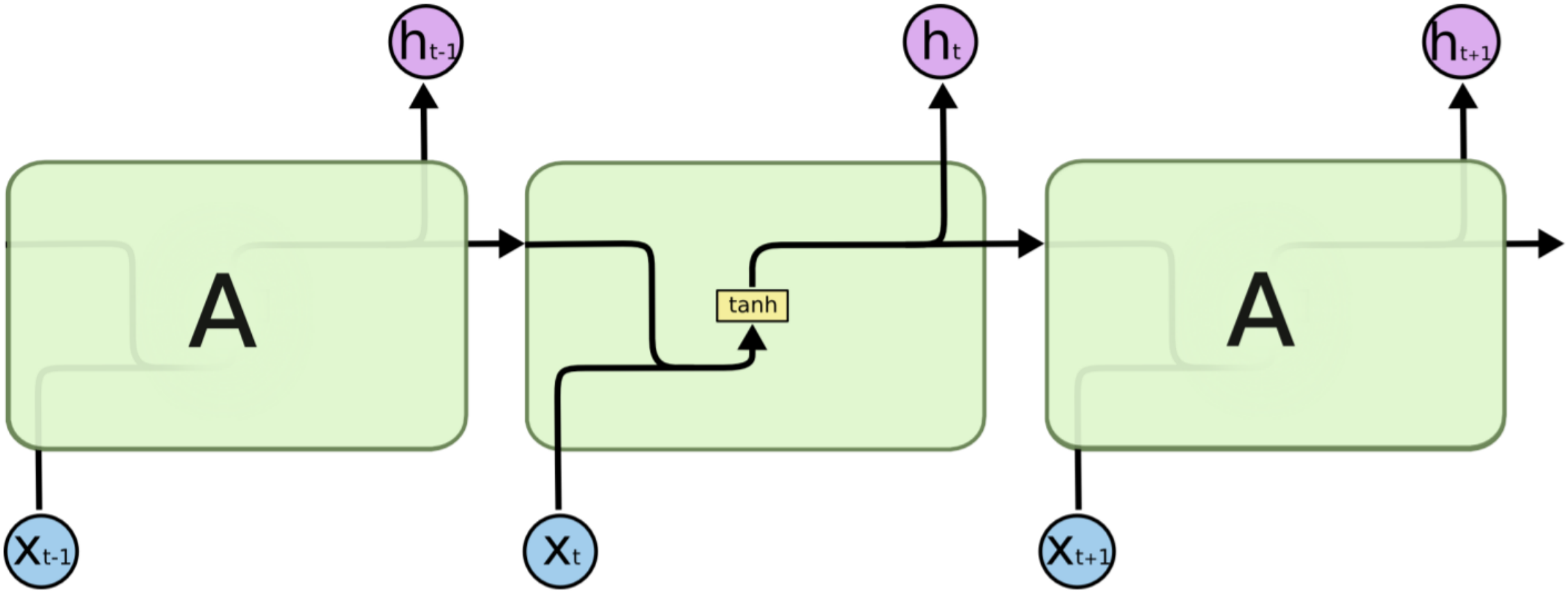




Embeddings: Word representation



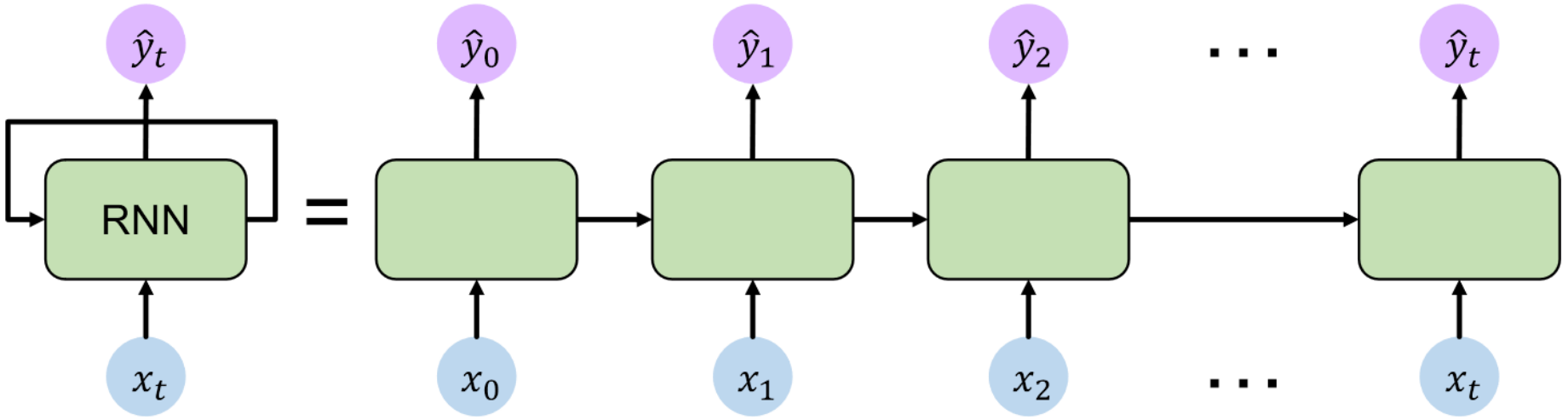
Vanilla RNNs



Vanilla RNNs

Problems with vanilla RNN

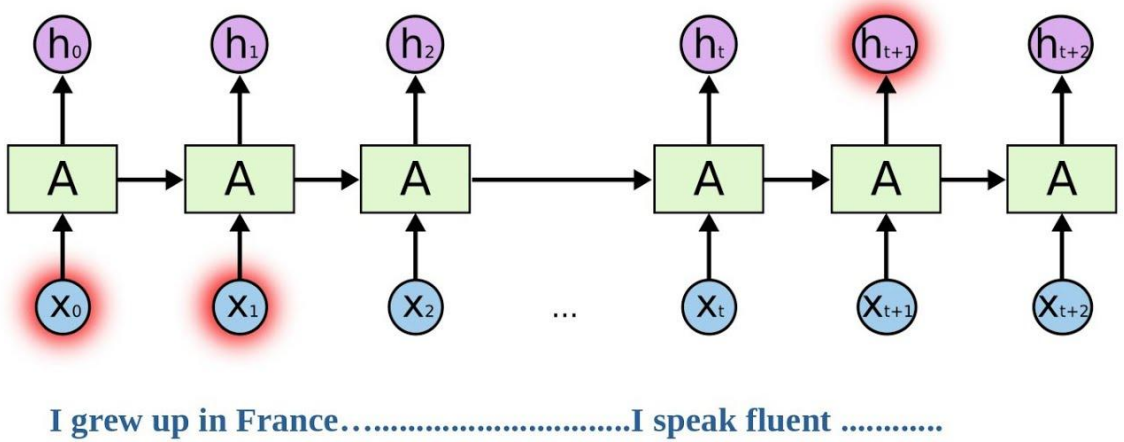
- Vanishing gradients
- Short term dependency

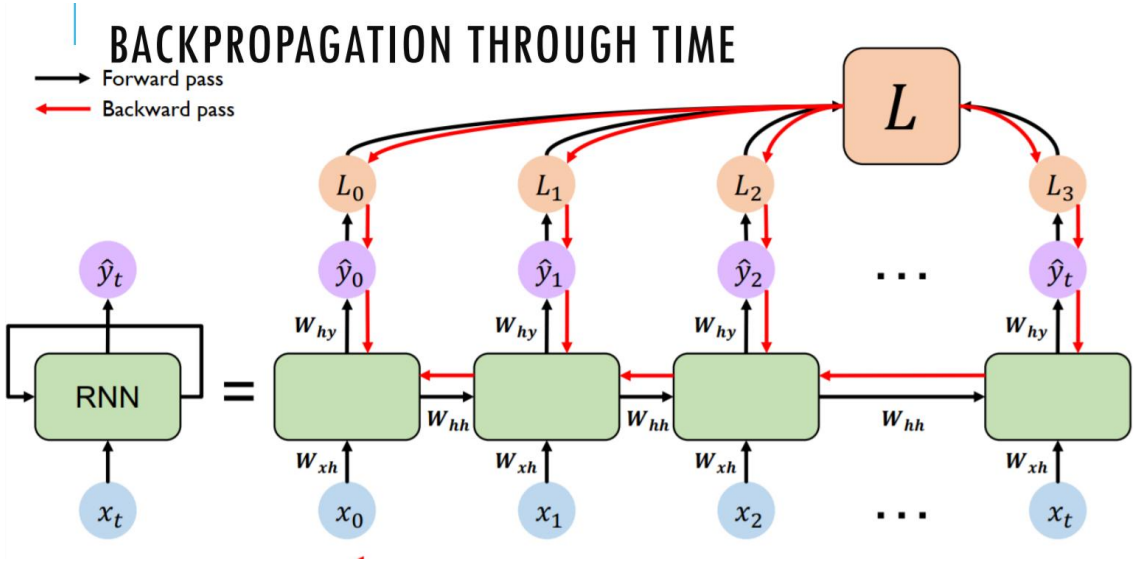


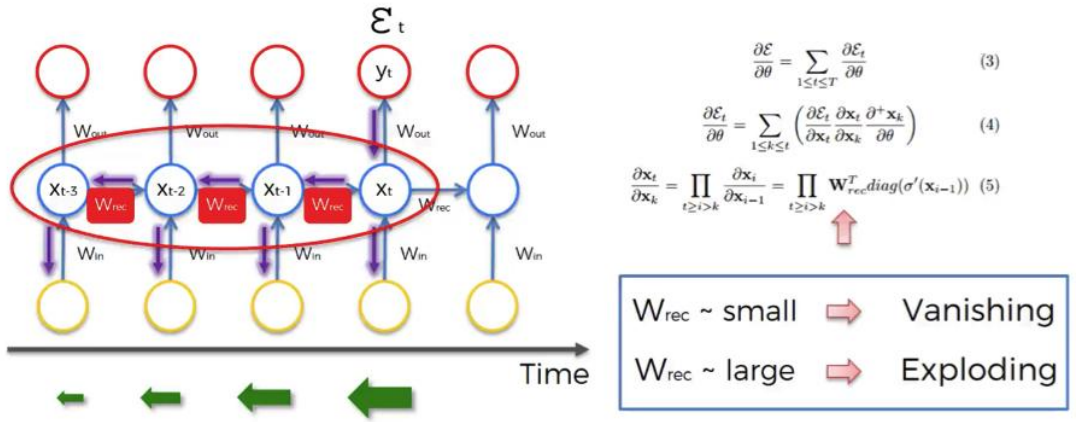
Vanilla RNNs

Problems with vanilla RNN

- It is like a very deep neural network
- Vanishing gradients
- Short term dependency





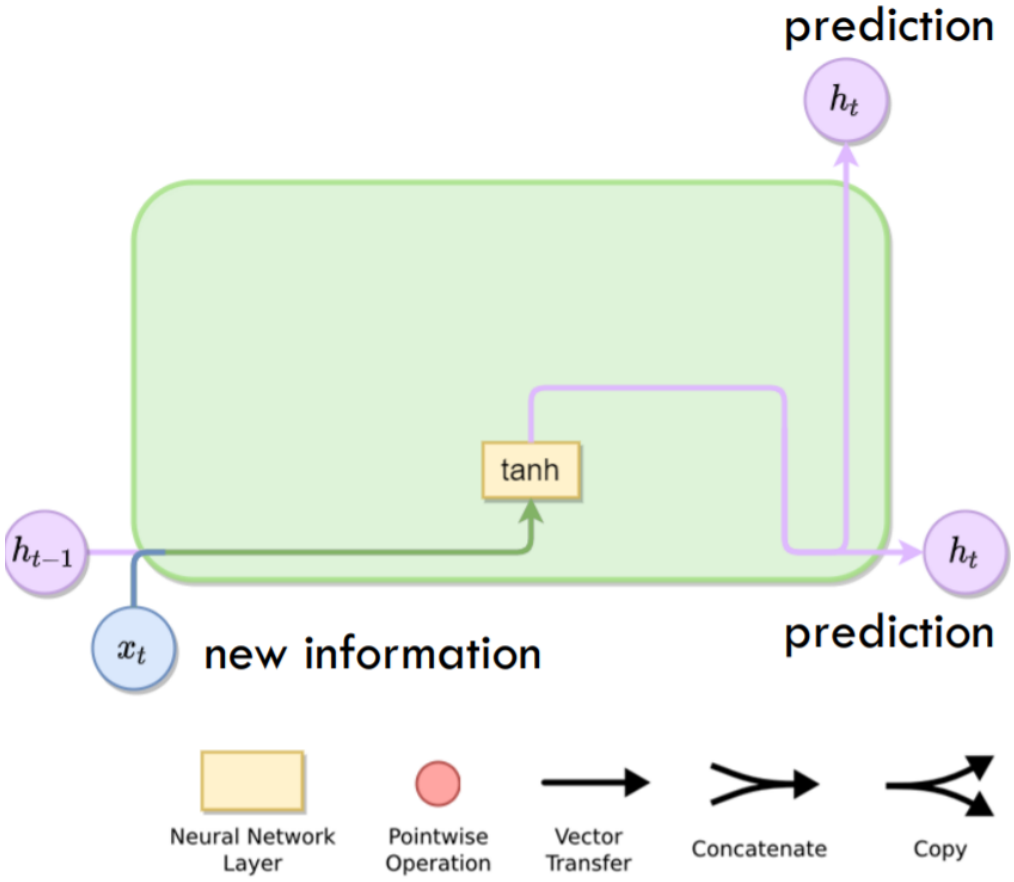


Long Short-Term Memory (LSTM)

Starting from a Vanilla RNN

Hidden state:

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



Long Short-Term Memory (LSTM)

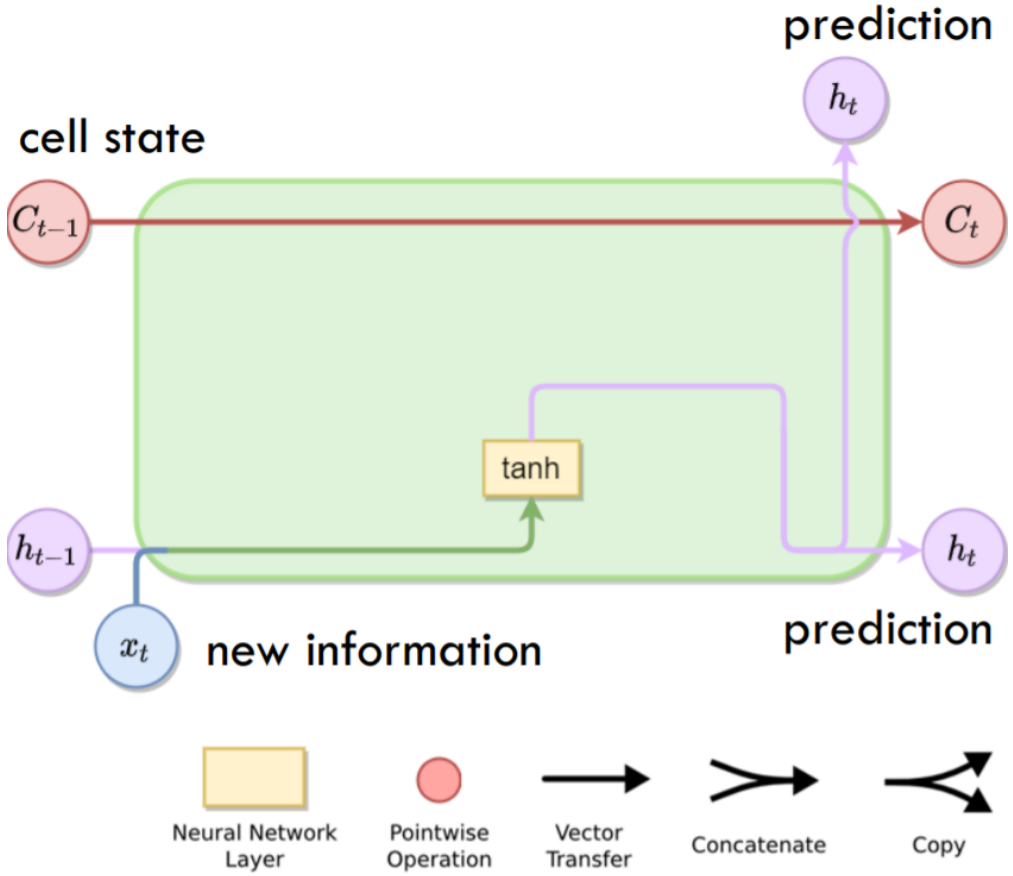
Add cell state to remember what happened many timesteps before

Hidden state:

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

Cell state:

$$C_t = C_{t-1}$$



Long Short-Term Memory (LSTM)

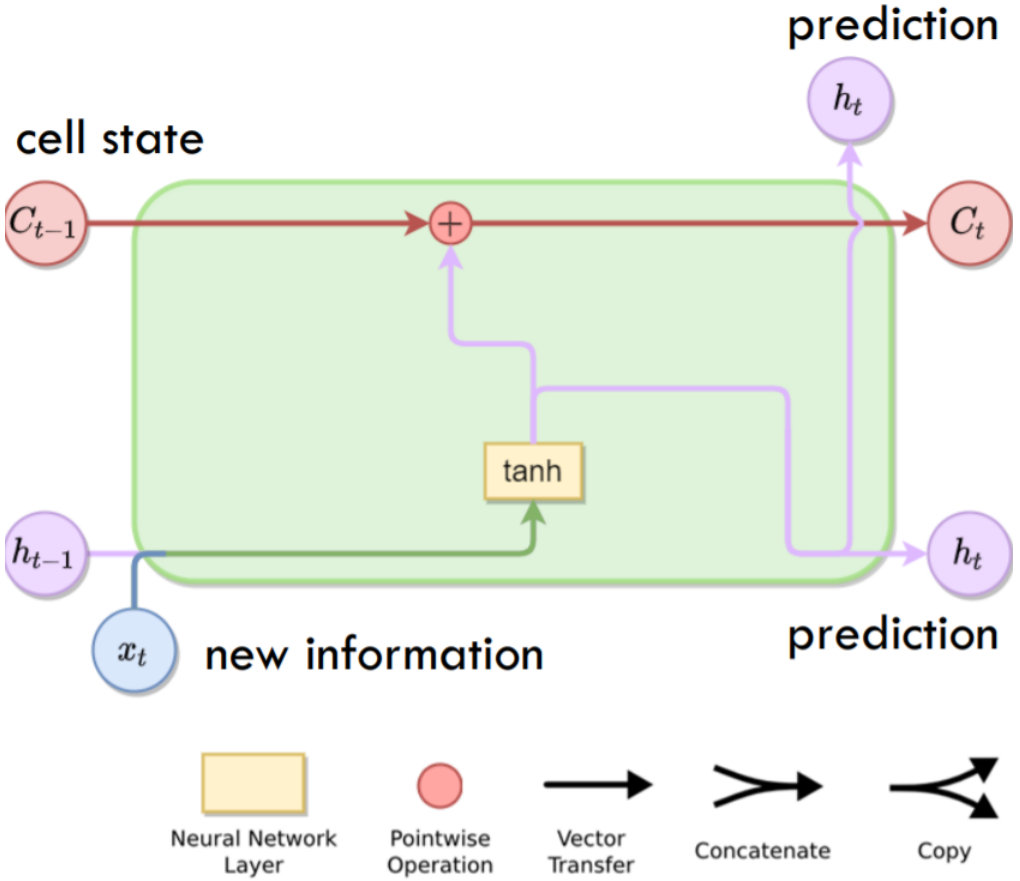
Update cell state based on the new timestep

Hidden state:

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

Cell state:

$$C_t = C_{t-1} + h_t$$



Long Short-Term Memory (LSTM)

Memory: to remember what happened many timesteps before

Forgetting: Based on what we are seeing now, we decide what we want to forget/remember

A copy of the predictions is saved for the next timestep

- Some of them are forgotten, some of them are remembered
- Then added back to the prediction

Hidden state:

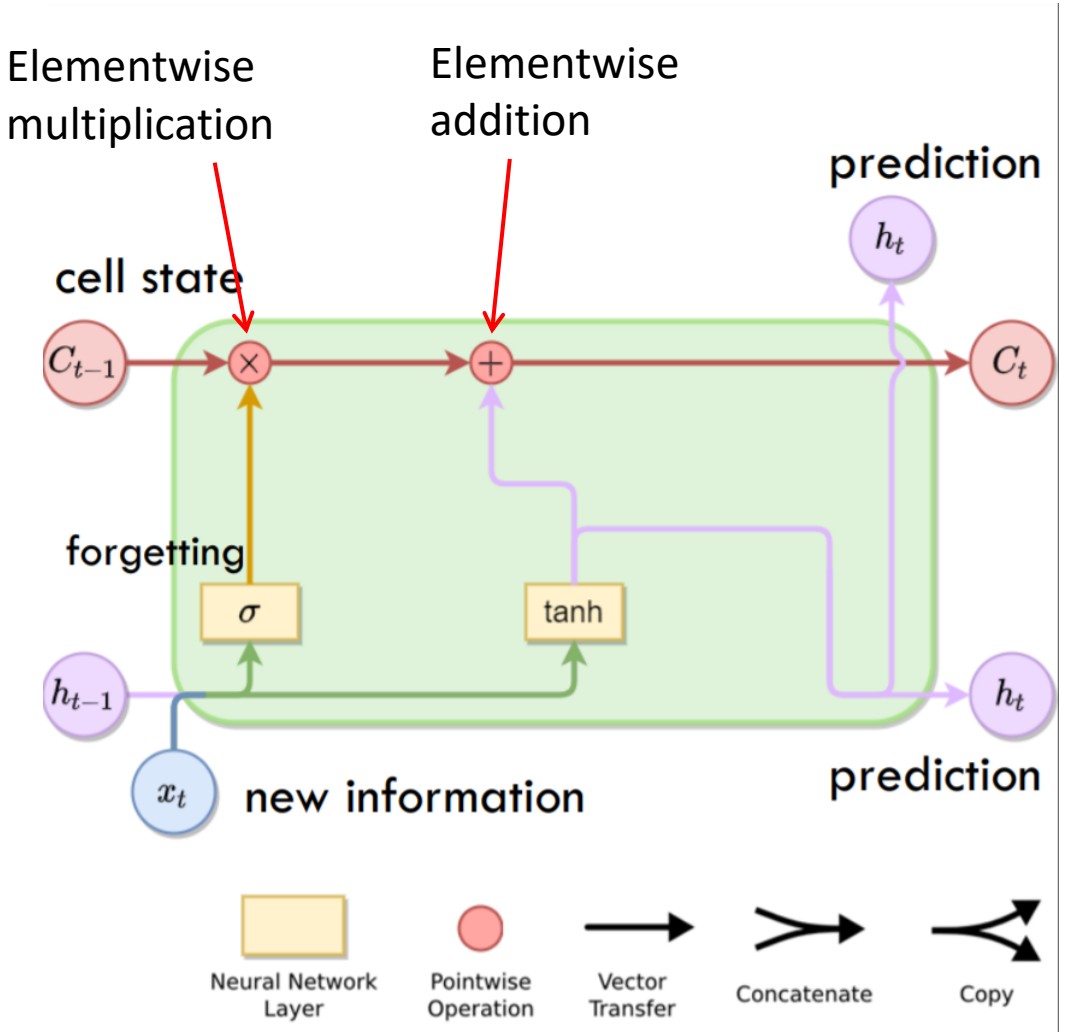
$$h_t = tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$$

Forget gate:

$$f_t = \sigma(W_{hf}h_{t-1} + W_{xf}x_t + b_f)$$

Cell state:

$$C_t = f_t * C_{t-1} + h_t$$

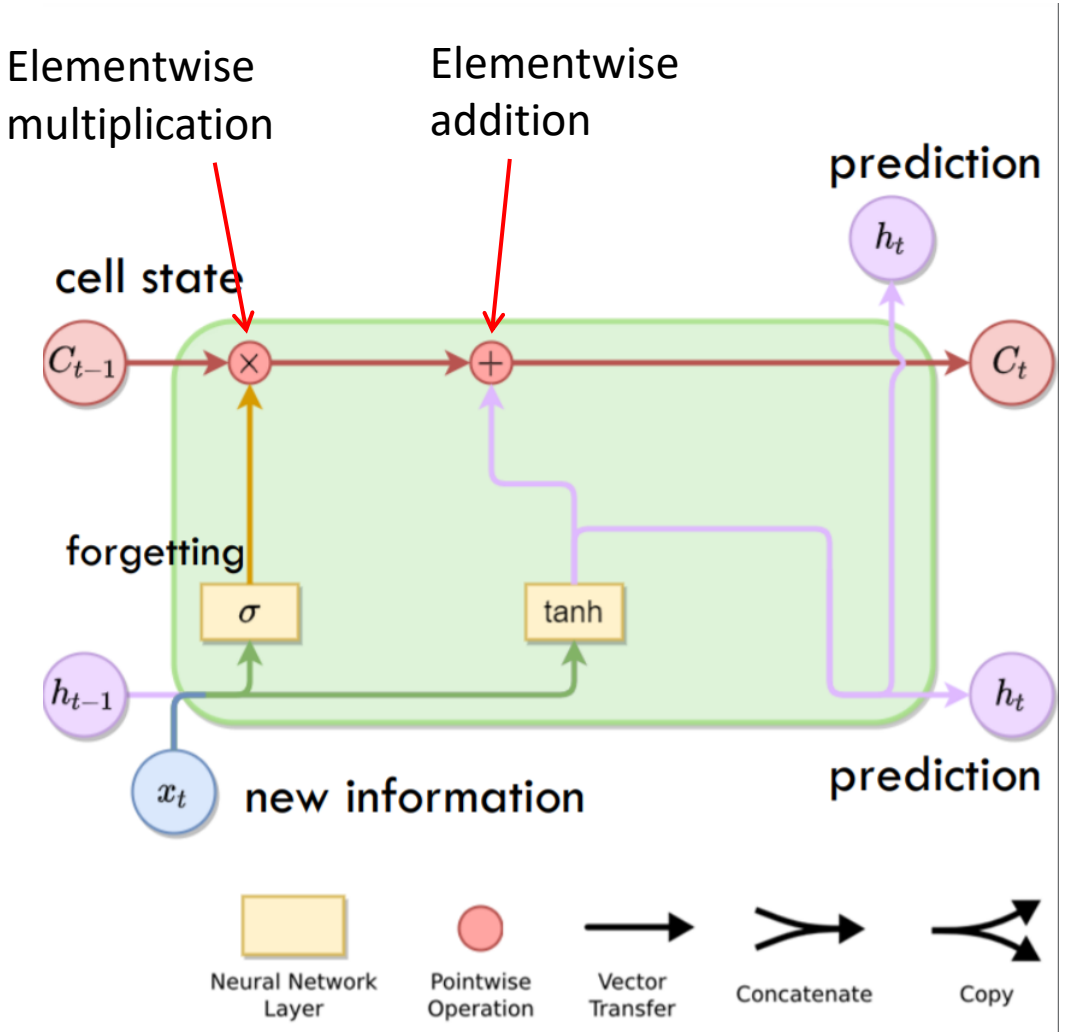


Long Short-Term Memory (LSTM)

Memory: to remember what happened many timesteps before

Forgetting: Based on what we are seeing now, we decide what we want to forget/remember

$$\begin{bmatrix}0.8\\0.8\\0.8\end{bmatrix} \times= \begin{bmatrix}1.0\\0.5\\0.0\end{bmatrix} = \begin{bmatrix}0.8 \cdot 1.0\\0.8 \cdot 0.5\\0.8 \cdot 0.0\end{bmatrix} = \begin{bmatrix}0.8\\0.4\\0.0\end{bmatrix}$$



Long Short-Term Memory (LSTM)

- Now we have a new state based on the previous and current time step
- **The hidden state is not affected by the cell state**
- We ignore what is not immediately relevant – **ignore gate**
- After addition we need **to rescale the cell state** – tanh

Forget gate:

$$f_t = \sigma(W_{hf} h_{t-1} + W_{xf} x_t + b_f)$$

Ignore gate:

$$i_t = \sigma(W_{hi} h_{t-1} + W_{xi} x_t + b_i)$$

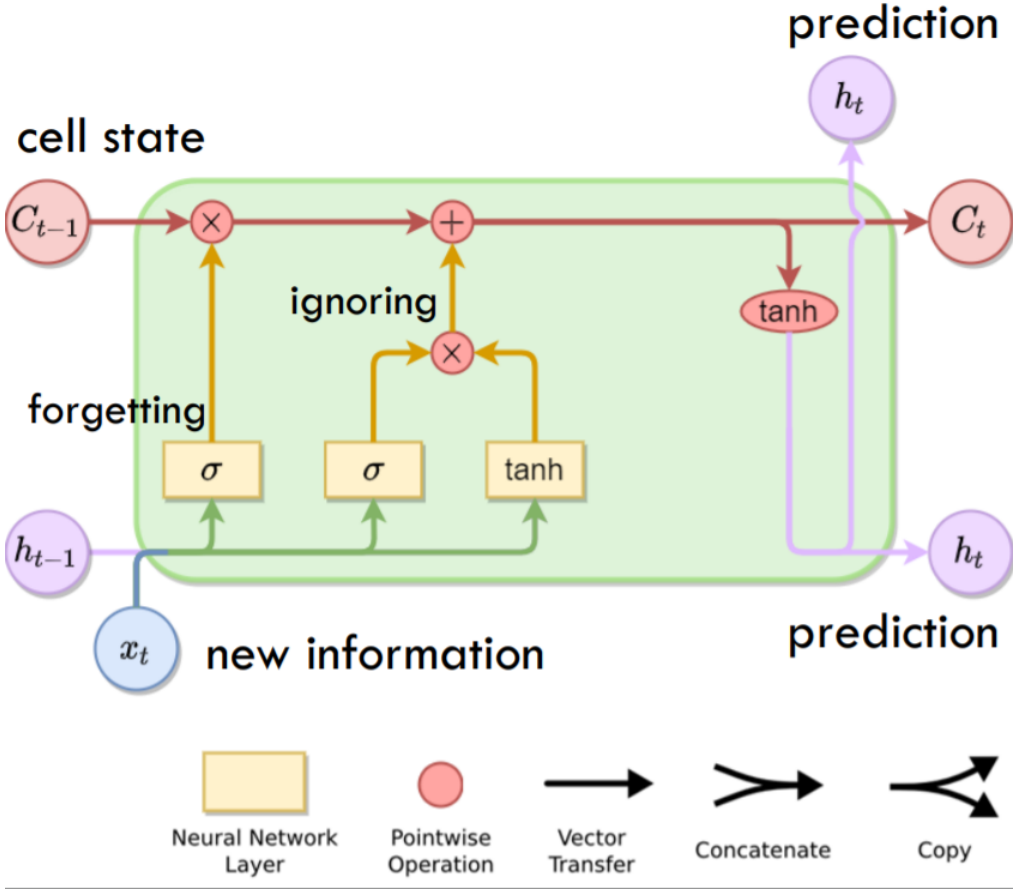
Cell state:

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

$$C_t = f_t * C_{t-1} + i_t * \widetilde{C_t}$$

Hidden state:

$$h_t = tanh(C_t)$$



Long Short-Term Memory (LSTM)

- We may not want to show everything after we combined our prediction with our memory
- Introduce a filter to keep our memories inside and let our prediction out – **selection/output gate**

Forget gate:

$$f_t = \sigma(W_{hf}h_{t-1} + W_{xf}x_t + b_f)$$

Ignore gate:

$$i_t = \sigma(W_{hi}h_{t-1} + W_{xi}x_t + b_i)$$

Output gate:

$$o_t = \sigma(W_{ho}h_{t-1} + W_{xo} + b_o)$$

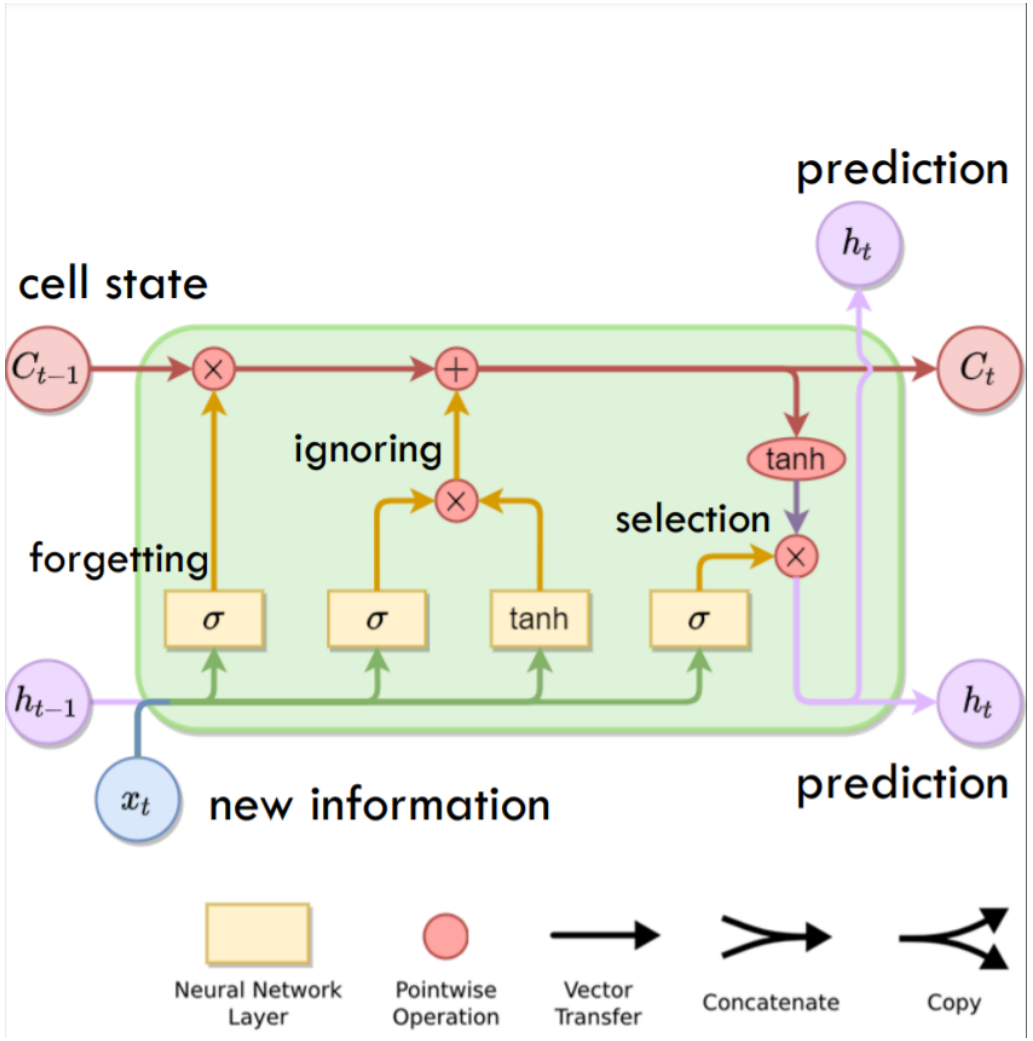
Cell state:

$$\widetilde{C_t} = tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h) \quad \text{(Candidate cell state)}$$

$$C_t = f_t * C_{t-1} + i_t * \widetilde{C_t}$$

Hidden state:

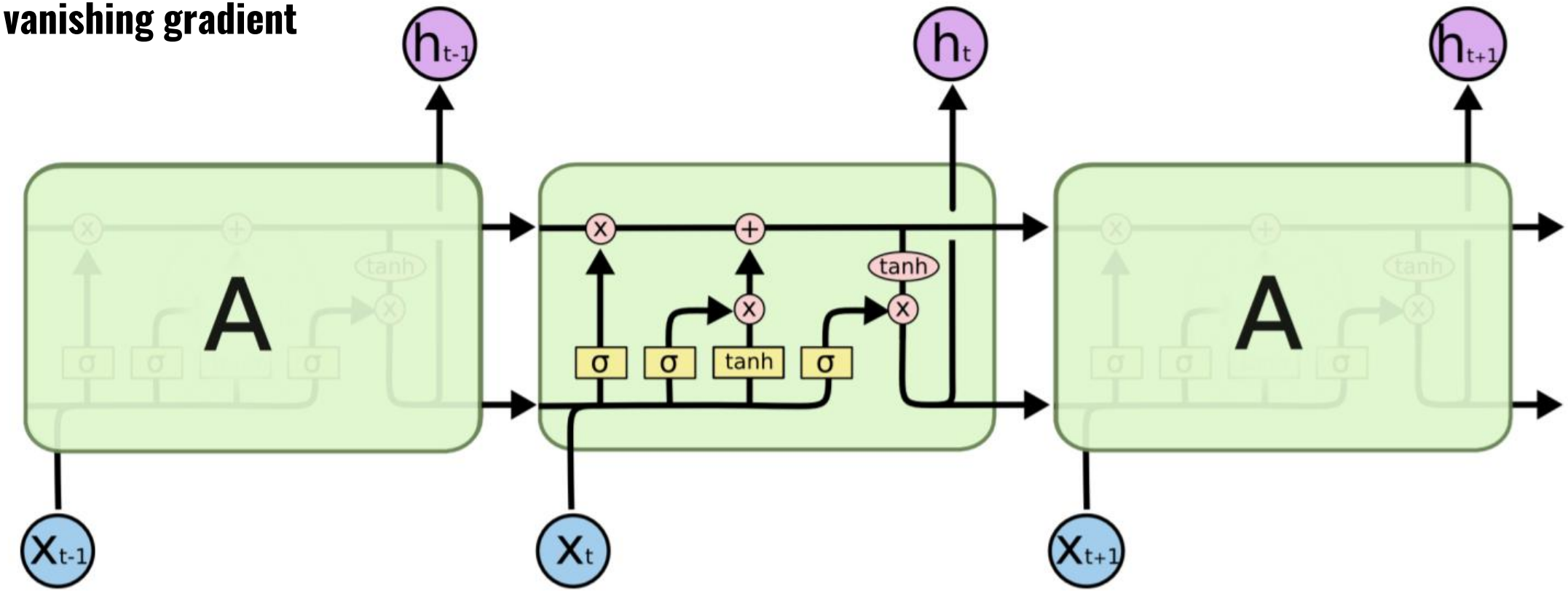
$$h_t = o_t * tanh(C_t)$$



Long Short-Term Memory (LSTM)

LSTM

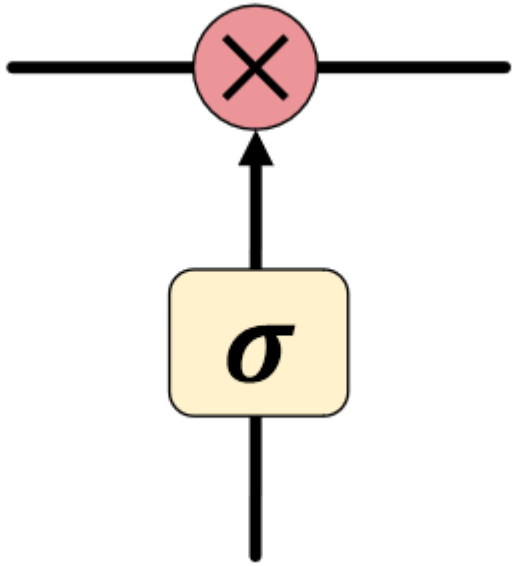
- a more complex network
- No vanishing gradient



Long Short-Term Memory (LSTM)

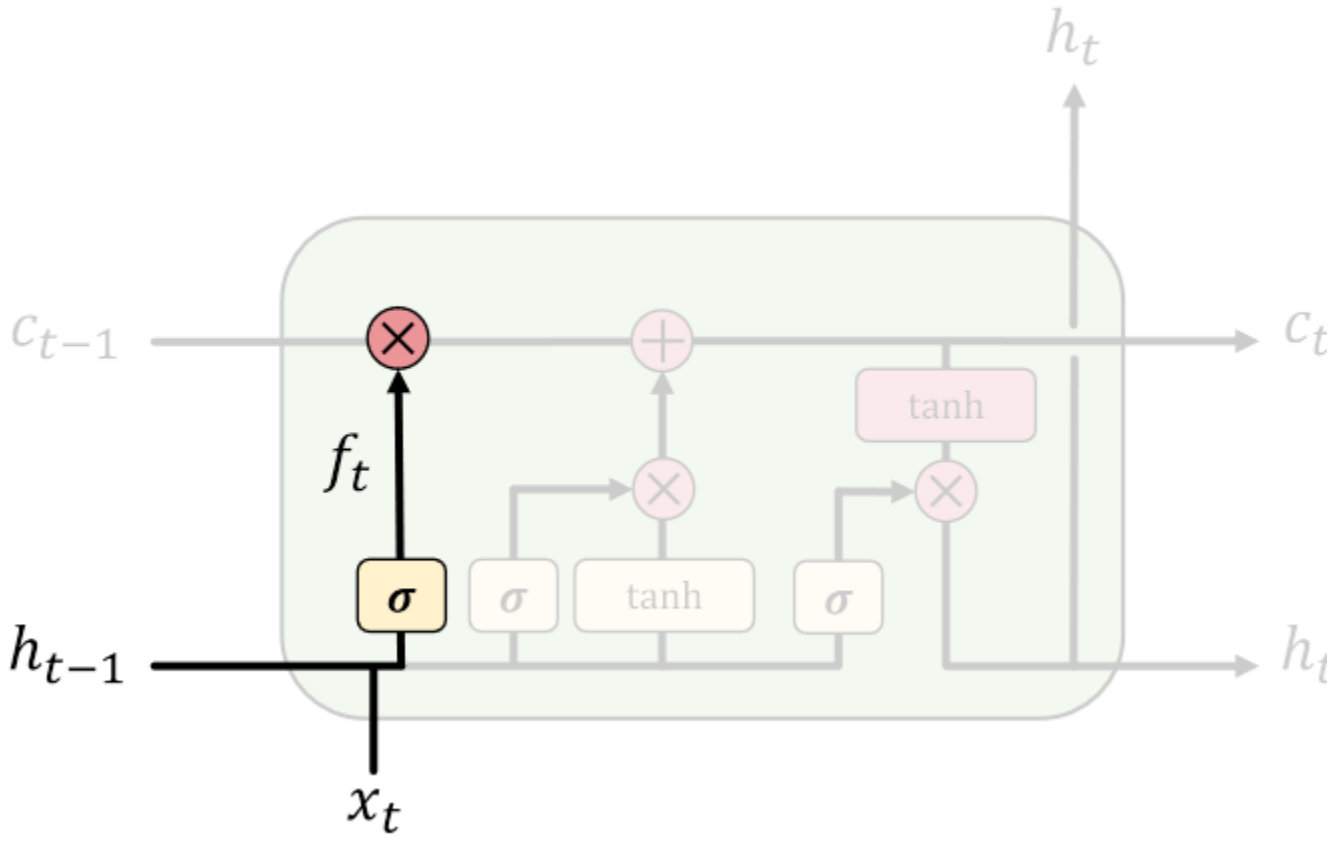
LSTM - Gate

- Sigmoid puts output between 0 and 1
- Output of sigmoid controls which info goes through the gate



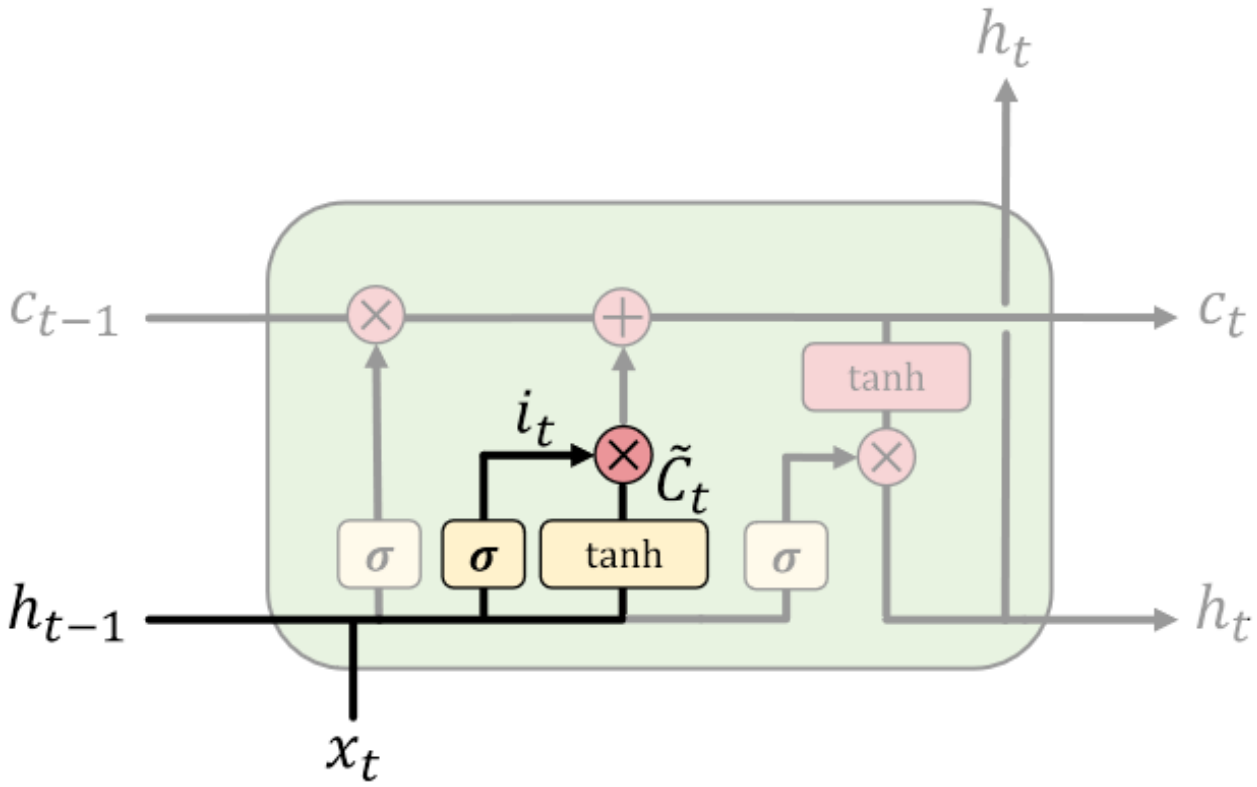
Long Short-Term Memory (LSTM)

LSTM – Forget Gate



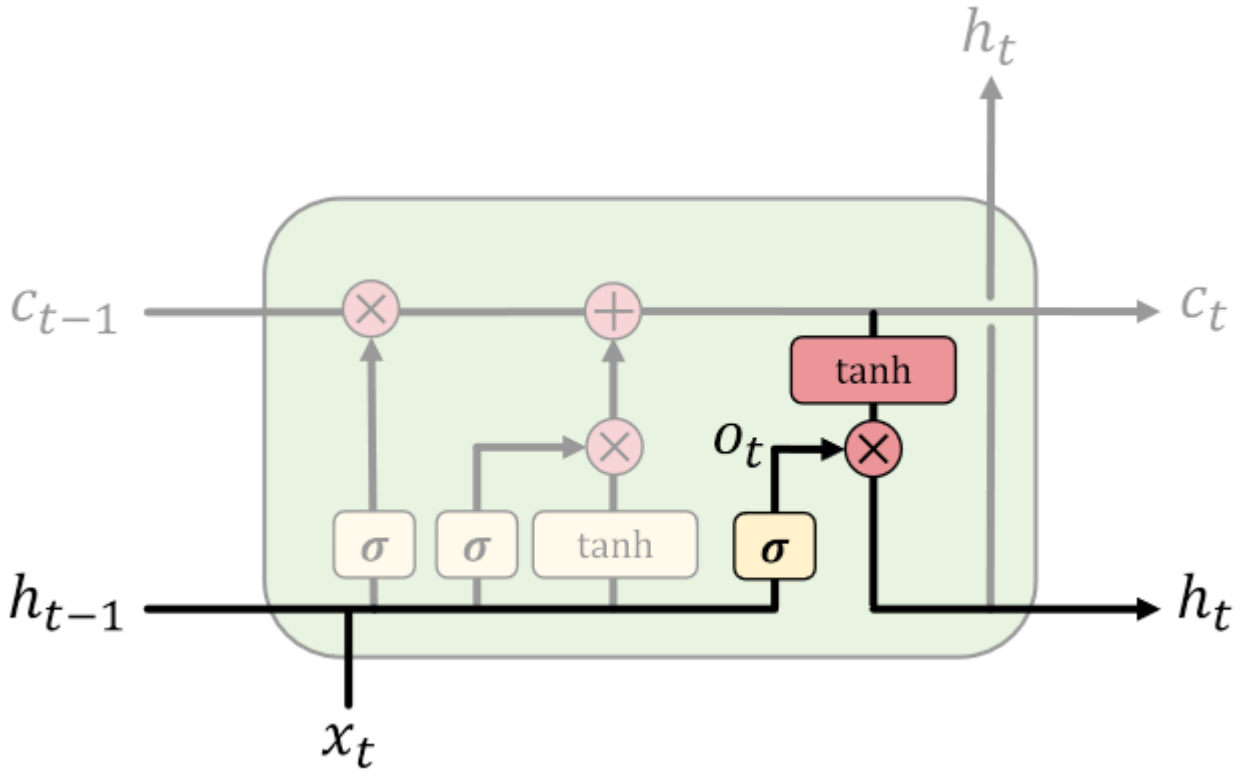
Long Short-Term Memory (LSTM)

LSTM – Input / Ignore Gate



Long Short-Term Memory (LSTM)

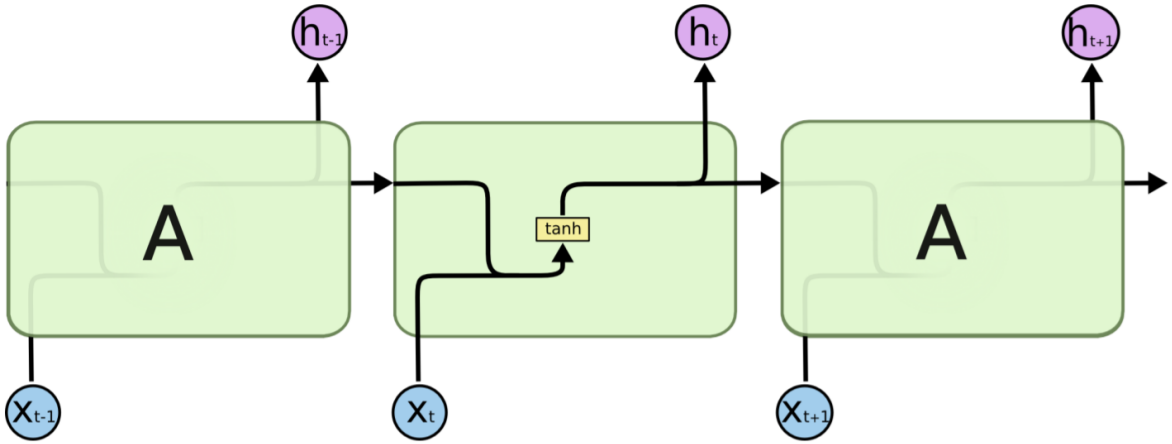
LSTM – Output Gate



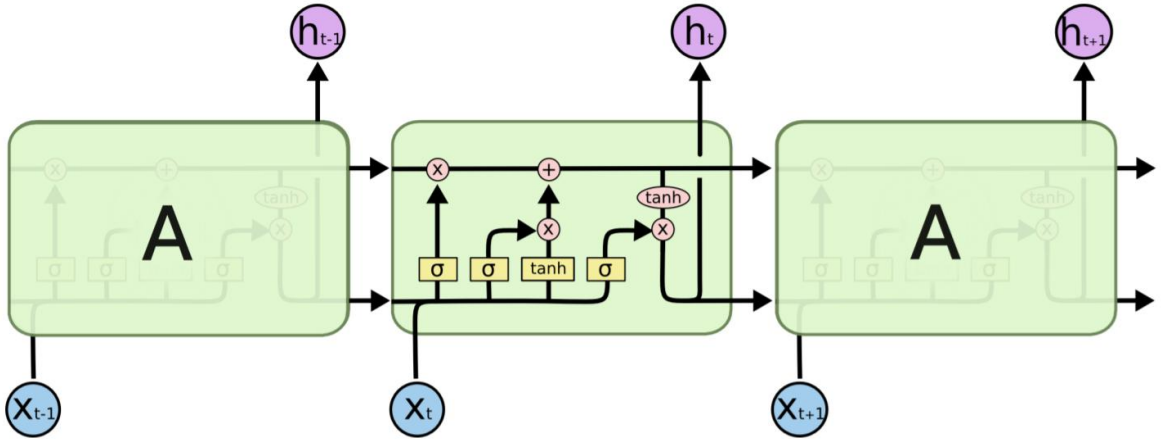
Long Short-Term Memory (LSTM)

LSTM – a more complex network

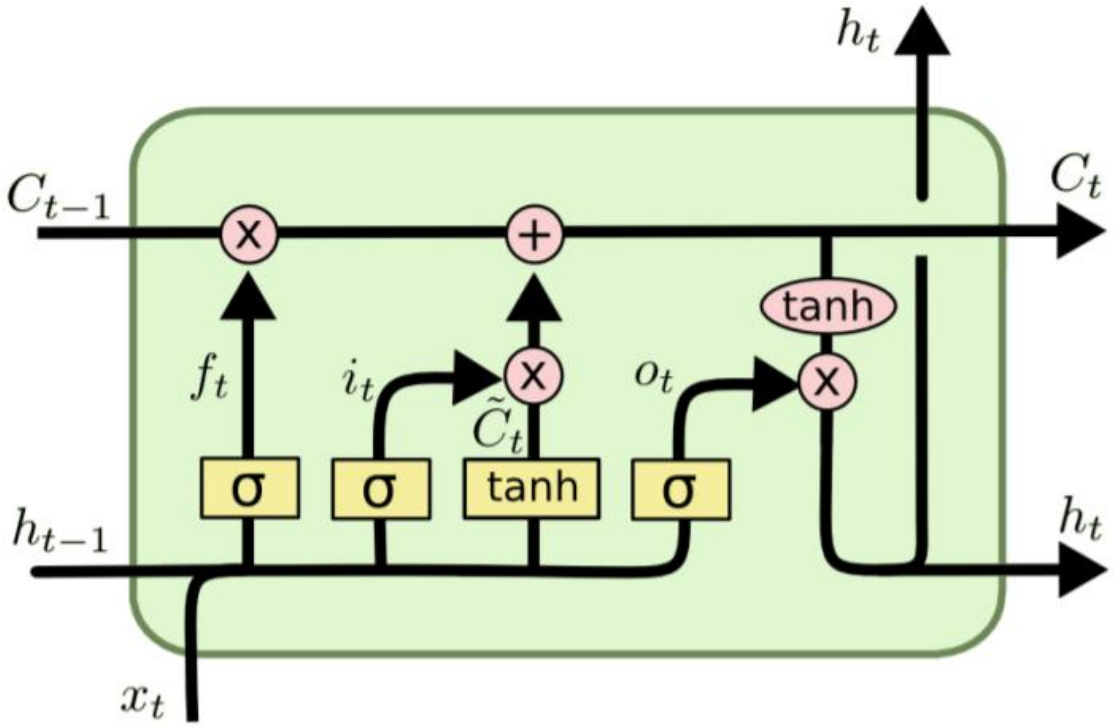
Vanilla RNN:



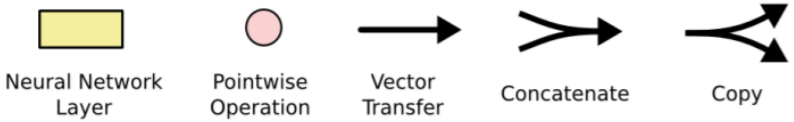
LSTM:



Long Short-Term Memory (LSTM)



f – forget gate: Whether to erase cell
i – ignore gate: Whether to write to cell
c̃ – "vanilla RNN": How much to write to cell
o – output gate: How much to reveal cell



Gates:

$$i_t = \sigma(W_{hi}h_{t-1} + W_{xi}x_t + b_i)$$

$$f_t = \sigma(W_{hf}h_{t-1} + W_{xf}x_t + b_f)$$

$$o_t = \sigma(W_{ho}h_{t-1} + W_{xo}x_t + b_o)$$

$$\tilde{C}_t = \tanh(W_{hc}h_{t-1} + W_{xc}x_t + b_c)$$

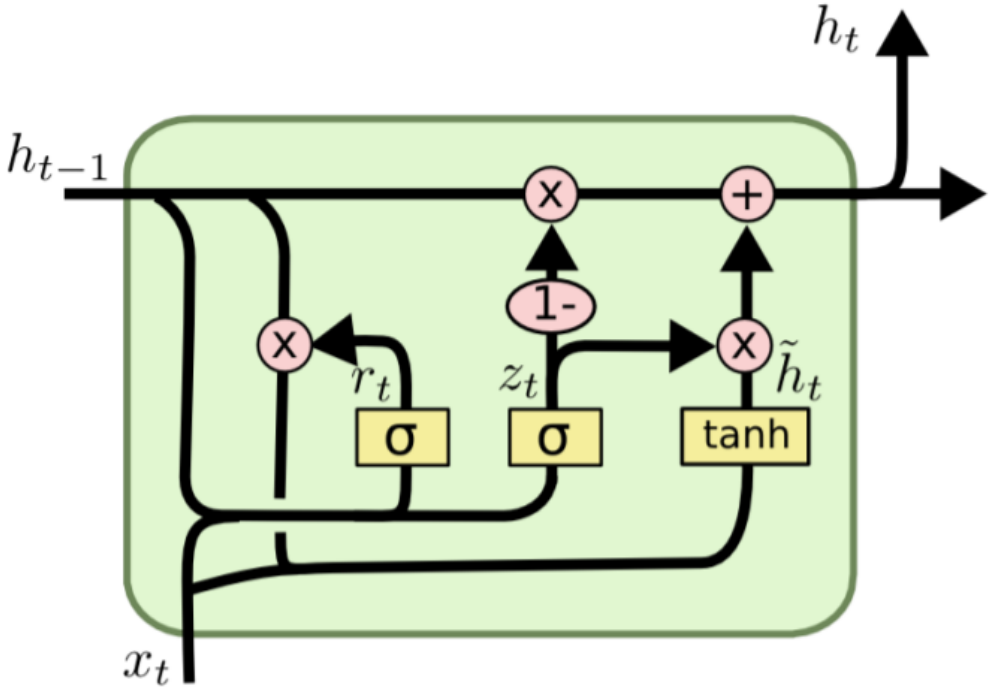
Outputs:

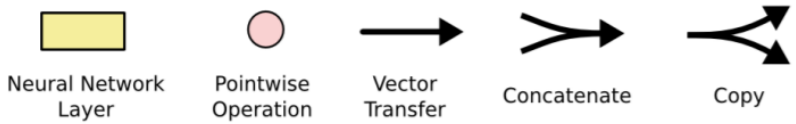
$$C_t = f_t \circ C_{t-1} + i_t \circ \tilde{C}_t$$

$$h_t = o_t \circ \tanh(\tilde{C}_t)$$

where $\circ$ is element-wise multiplication operation

Gated Recurrent Unit (GRU)





$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

LSTM vs GRU

The main differences between GRU and LSTM are:

- **Number of gates:** GRU has two gates - an update gate and a reset gate, whereas LSTM has three gates - input, forget, and output gates.
- **Memory cell:** Unlike LSTM, GRU doesn't have a separate memory cell. It combines the hidden state and memory cell into a single hidden state, simplifying the structure.

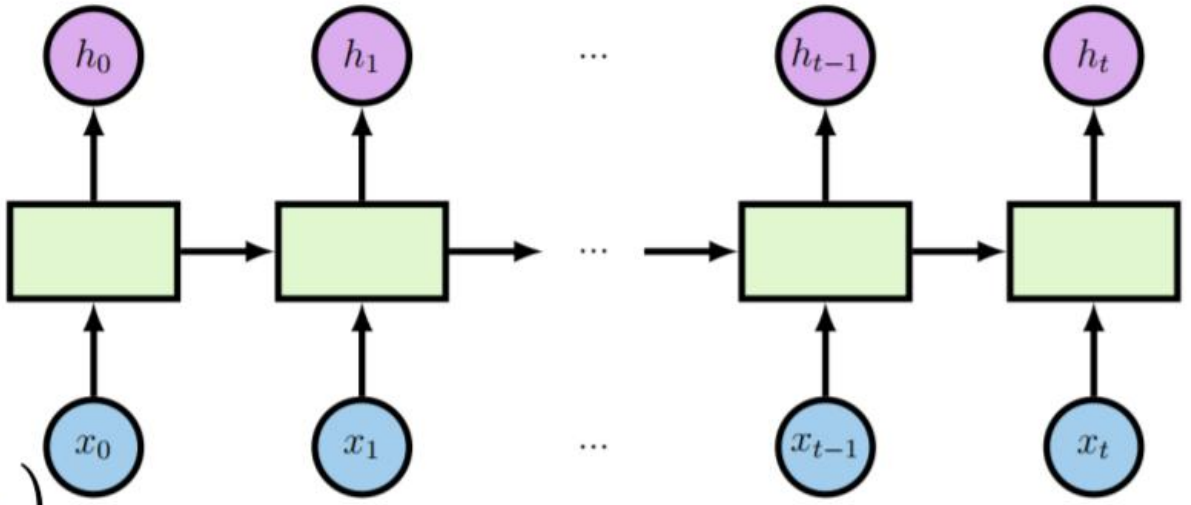
Recurrent Layer

Have an internal hidden state

- Updated every time and fed back to the model, when a new input is read
- Used as a past contextual information.

Suitable for learning handling long-term dependencies

- E.g. time series and sequences



A recurrent layer:

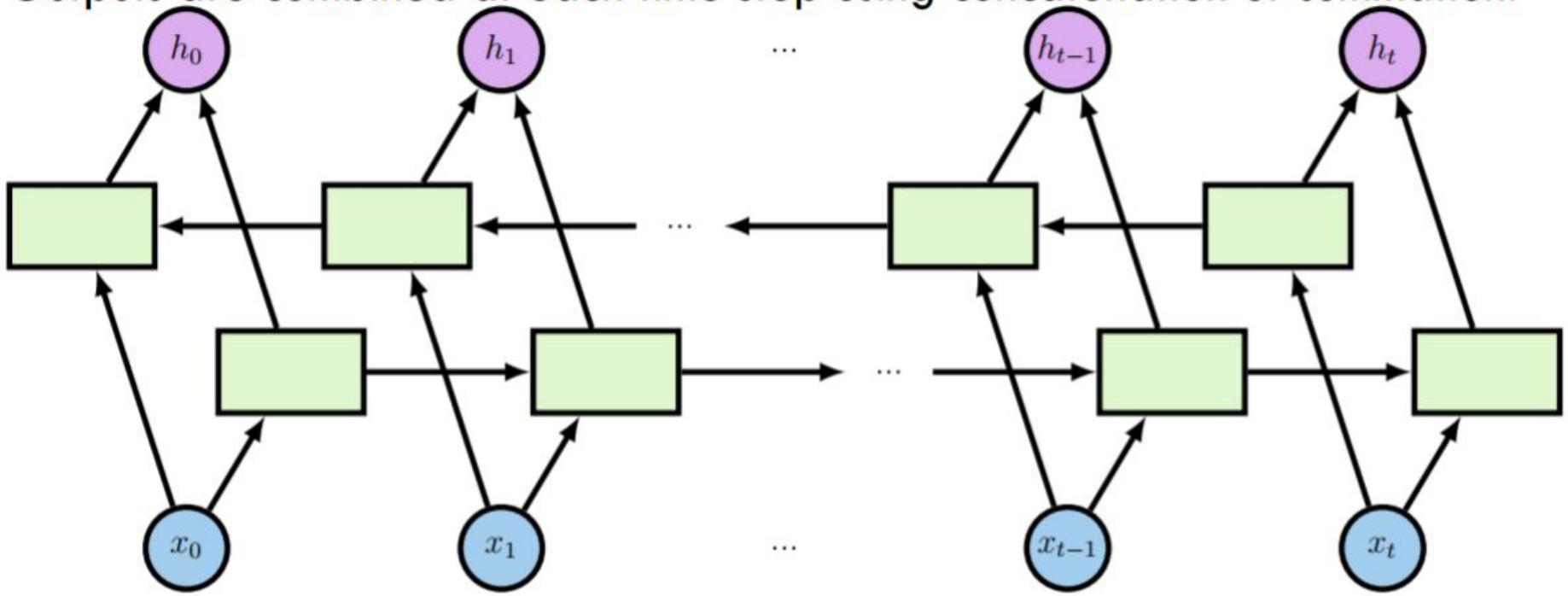
$$h_{d,t}^{(n)} = g_d\left(W_d^h\, h_{d,t-1}^{(n)} + W_d^x\, h_{d-1,t}^{(n)} + b_d\right)$$

- Where $t = 1, \dots, T_n$; $\theta_d = \{W_d^h, W_d^x, b_d\}$

Bidirectional RNN

Idea: use 2 independent recurrent models together.

- Input is fed in the proper time order to the first one, and in reverse time order to the second one.
- Outputs are combined at each time step using concatenation or summation.



Bidirectional RNN

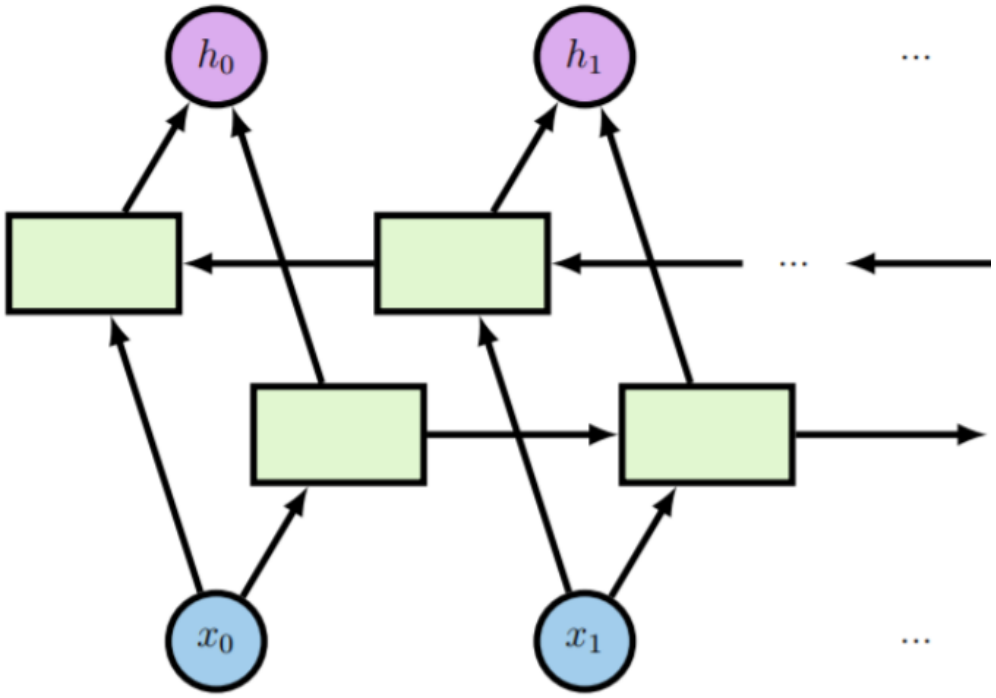
BIDIRECTIONAL RNN (BRNN)

A BRNN layer:

$$\overrightarrow{\boldsymbol{h}_{d,t}^{(n)}} = \overrightarrow{g_d}\left(\overrightarrow{\boldsymbol{W}_d^h}\overrightarrow{\boldsymbol{h}_{d,t-1}^{(n)}} + \overrightarrow{\boldsymbol{W}_d^x}\overrightarrow{\boldsymbol{h}_{d-1,t}^{(n)}} + \overrightarrow{\boldsymbol{b}_d}\right)$$

$$\overleftarrow{\boldsymbol{h}_{d,t}^{(n)}} = \overleftarrow{g_d}\left(\overleftarrow{\boldsymbol{W}_d^h}\overleftarrow{\boldsymbol{h}_{d,t+1}^{(n)}} + \overleftarrow{\boldsymbol{W}_d^x}\overleftarrow{\boldsymbol{h}_{d-1,t}^{(n)}} + \overleftarrow{\boldsymbol{b}_d}\right)$$

$$\boldsymbol{h}_{d,t}^{(n)} = g_d\left(\overrightarrow{\boldsymbol{W}_d}\overrightarrow{\boldsymbol{h}_{d,t}^{(n)}} + \overleftarrow{\boldsymbol{W}_d}\overleftarrow{\boldsymbol{h}_{d,t}^{(n)}} + \boldsymbol{b}_d\right)$$

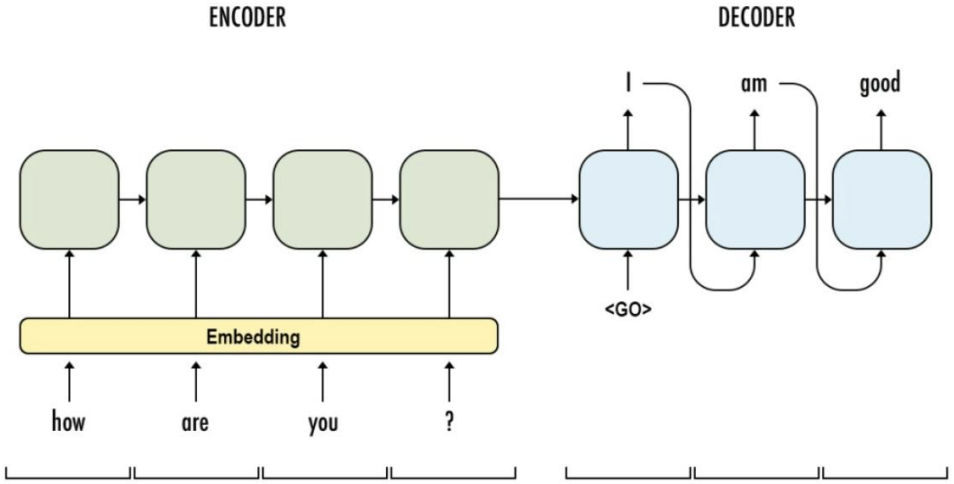


Where

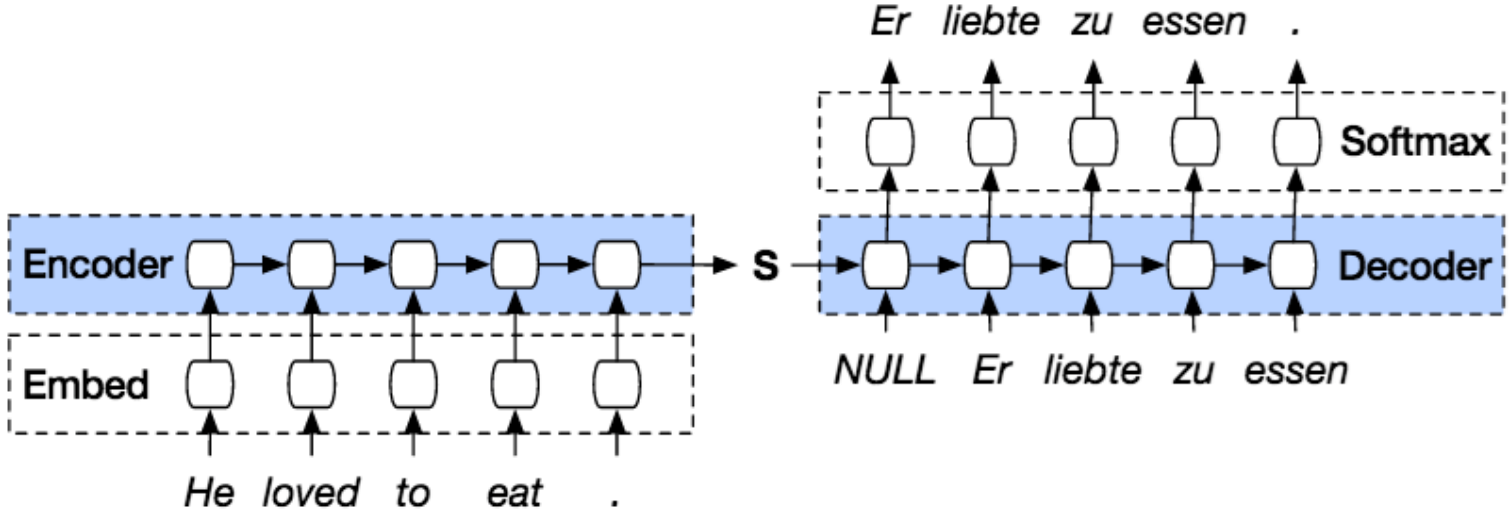
- "→" means normal time order, while "←" is associated with reverse time order
- $t = 1, \ldots, T_n$;
- $\boldsymbol{\theta}_d = \left\{\overrightarrow{\boldsymbol{W}_d^h}, \overrightarrow{\boldsymbol{W}_d^x}, \overrightarrow{\boldsymbol{b}_d}, \overleftarrow{\boldsymbol{W}_d^h}, \overleftarrow{\boldsymbol{W}_d^x}, \overleftarrow{\boldsymbol{b}_d}, \overrightarrow{\boldsymbol{W}_d}, \overleftarrow{\boldsymbol{W}_d}, \boldsymbol{b}_d\right\}$

Encoder – Decoder (Seq2Seq)

Question : Answer
(sequence) : (sequence)

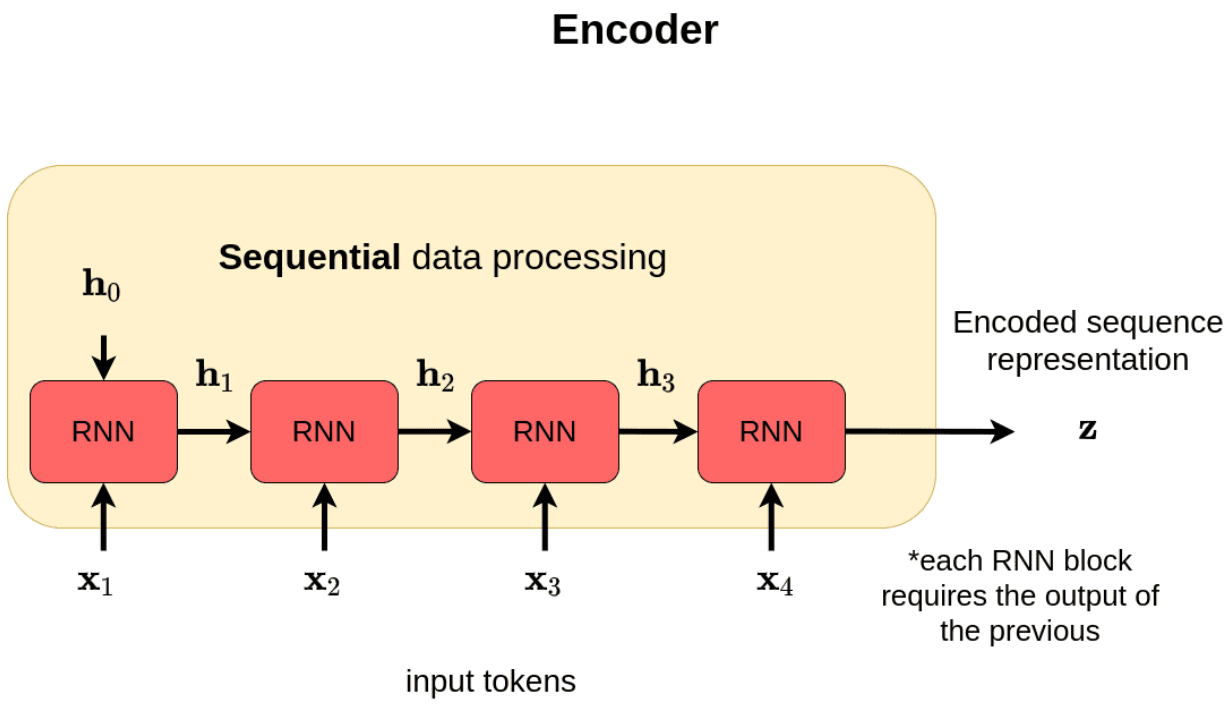


Machine Translation
English : German
(sequence) : (sequence)



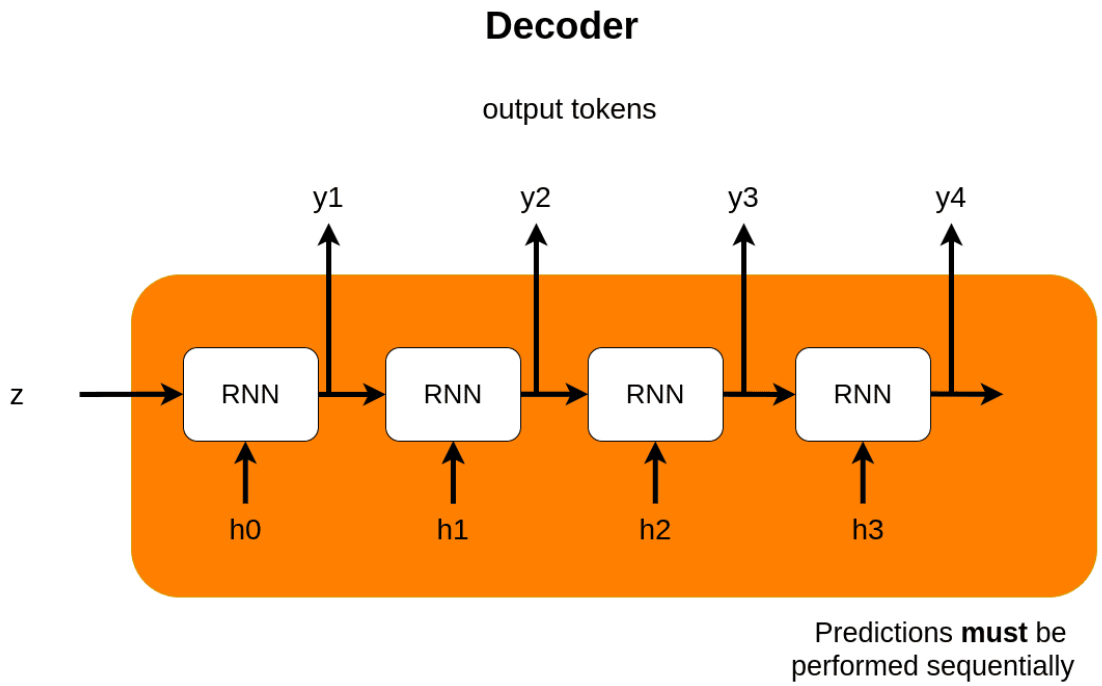
Encoder – Decoder (Seq2Seq)

- The encoder and decoder are nothing more than stacked RNN layers, such as LSTM's. The encoder processes the input and produces one compact representation, called z, from all the input timesteps. It can be regarded as a compressed format of the input.



Encoder – Decoder (Seq2Seq)

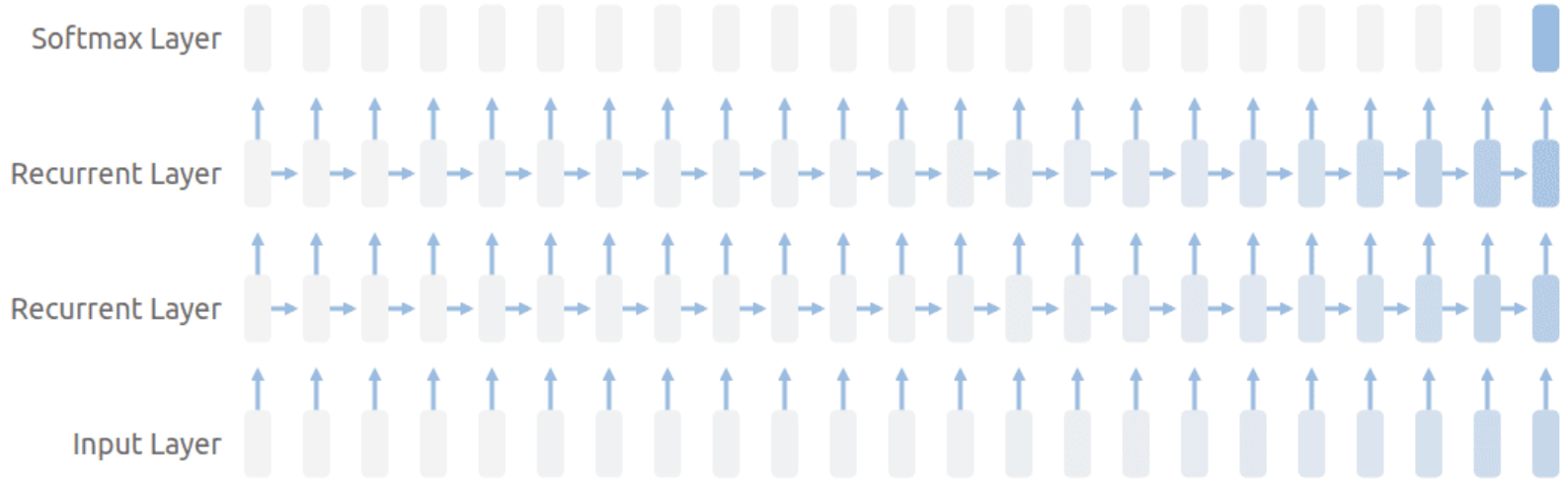
- On the other hand, the decoder receives the context vector z and generates the output sequence. The most common application of Seq2seq is language translation. We can think of the input sequence as the representation of a sentence in English and the output as the same sentence in French.



Encoder – Decoder (Seq2Seq)

Problems with the Seq2Seq

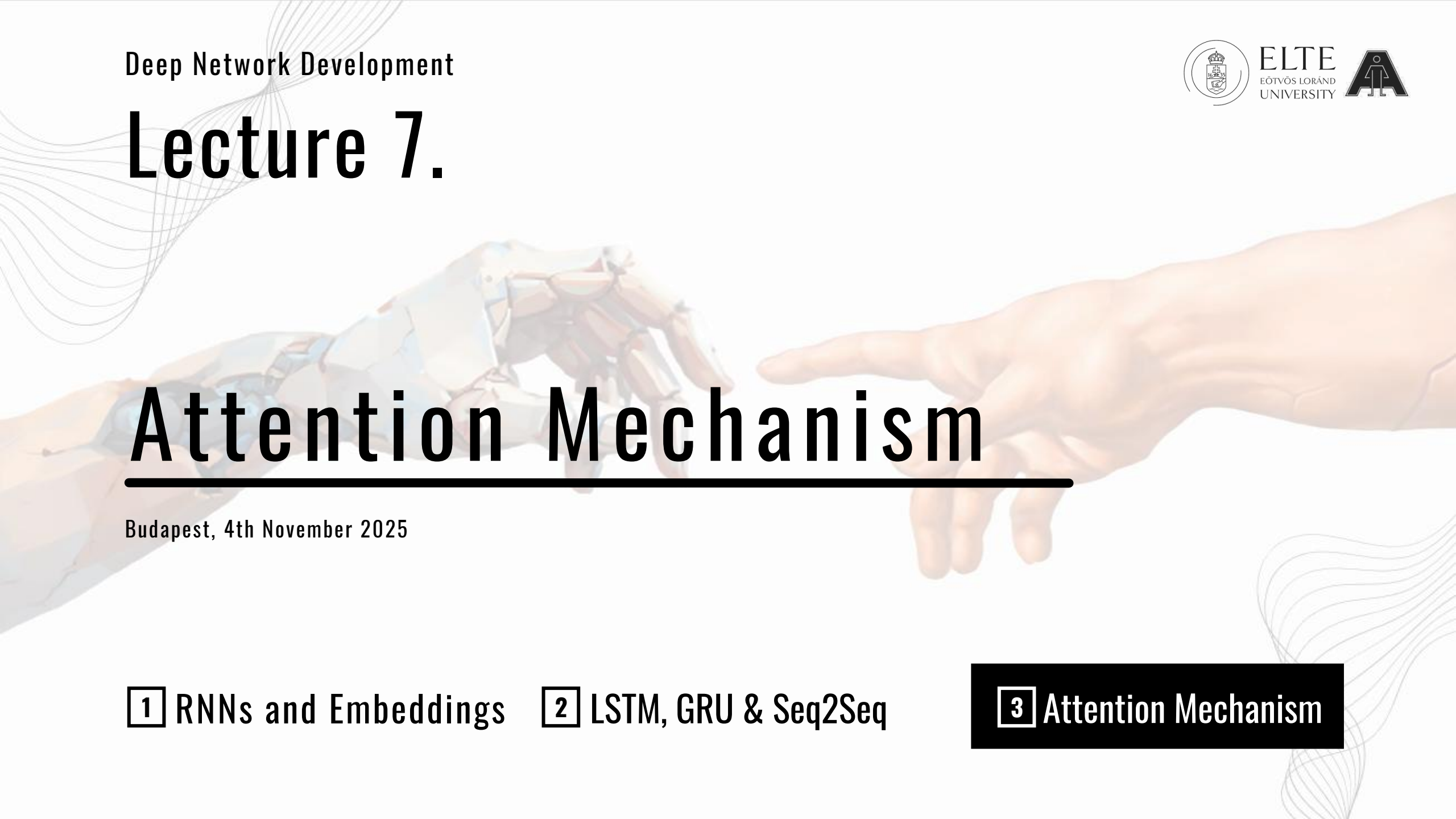
- The intermediate representation **z** cannot encode information from all the input timesteps. This is commonly known as the **bottleneck problem**. The vector z needs to capture all the information about the source sentence.
- In practice, how far we can see in the past (the so-called reference window) is finite. RNN's tend to **forget information** from timesteps that are far behind.



Vanishing Gradient: where the contribution from the earlier steps becomes insignificant in the gradient for the vanilla RNN unit.

Deep Network Development

Lecture 7.



Attention Mechanism

Budapest, 4th November 2025

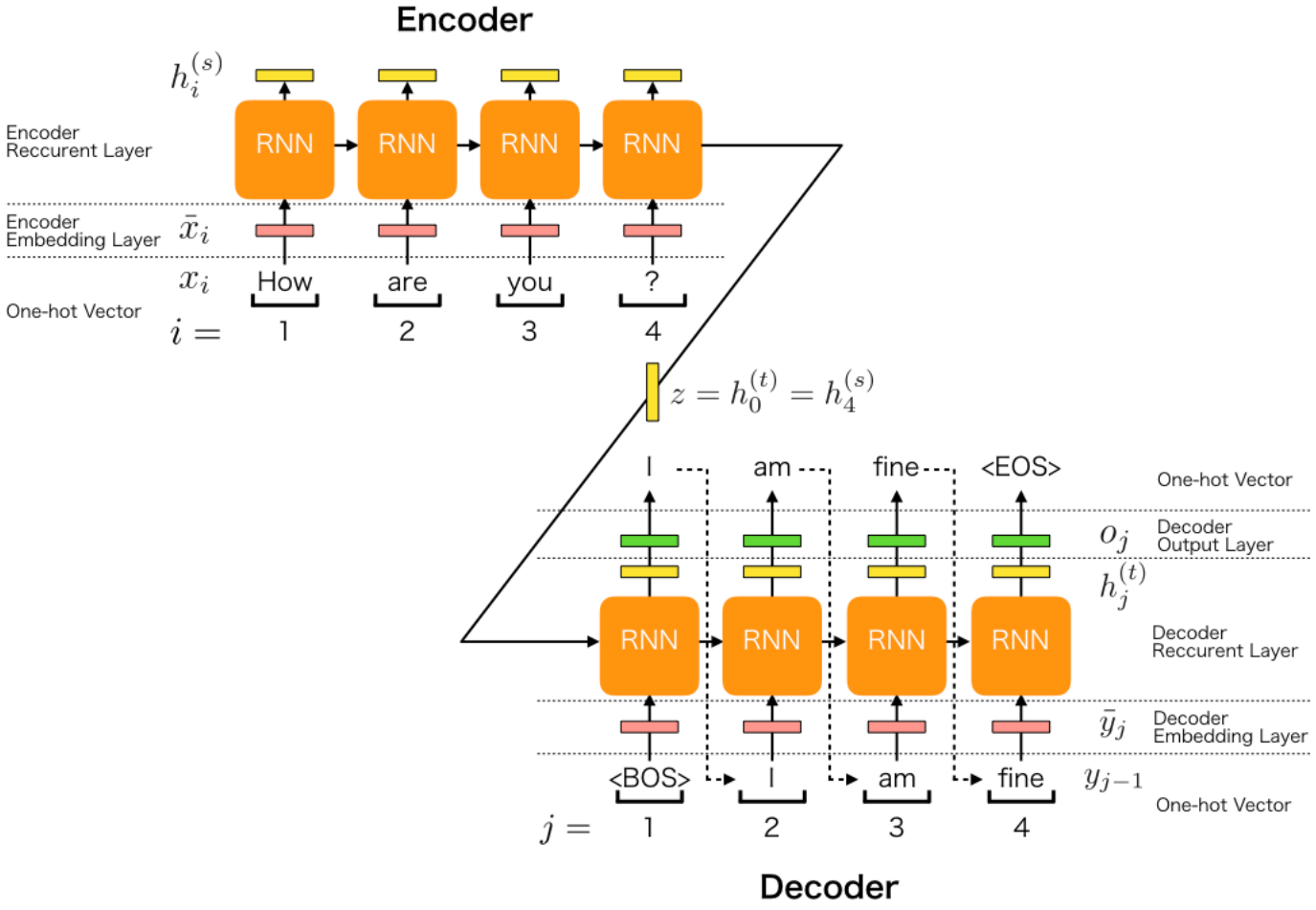
1 RNNs and Embeddings

2 LSTM, GRU & Seq2Seq

3 Attention Mechanism

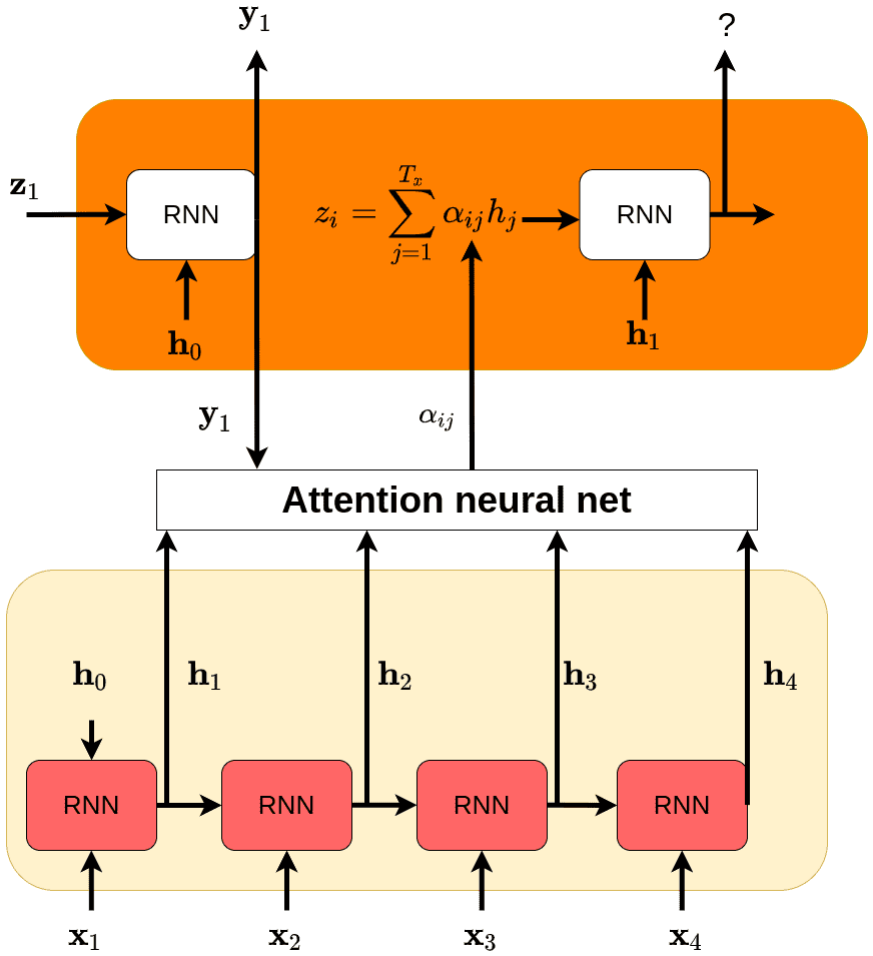
Introduction to Attention Mechanism

- All the information from the encoder is represented in the z vector (context)
- However, as seen previously, information from earlier timestamps is not preserved
- Can we create a better context vector?



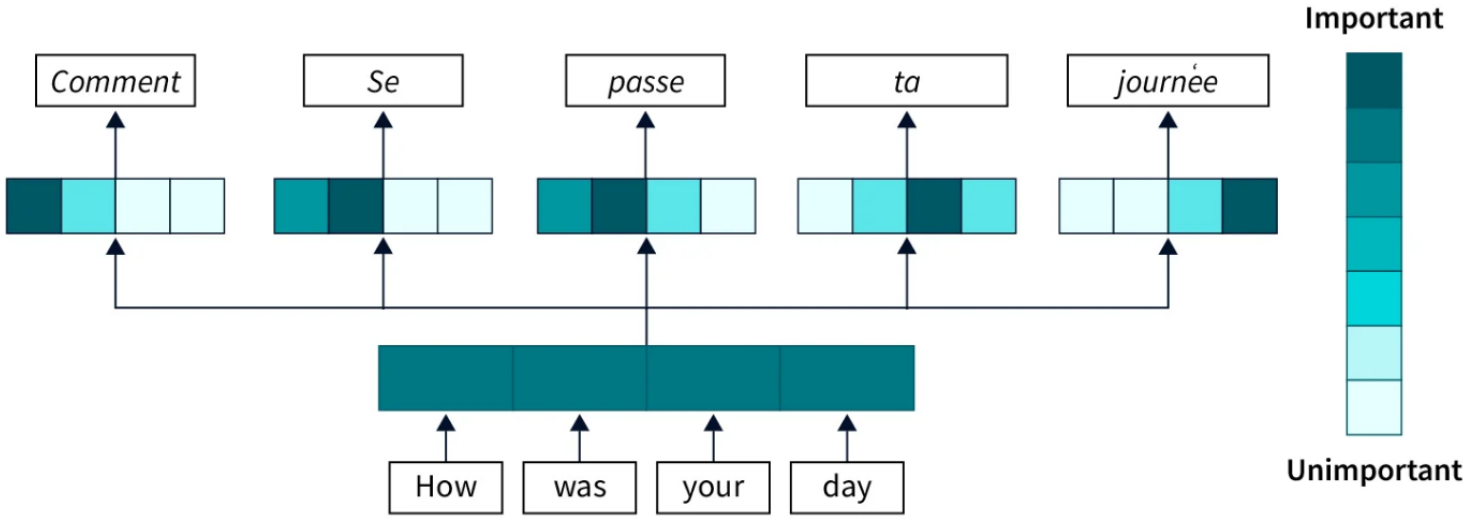
Introduction to Attention Mechanism

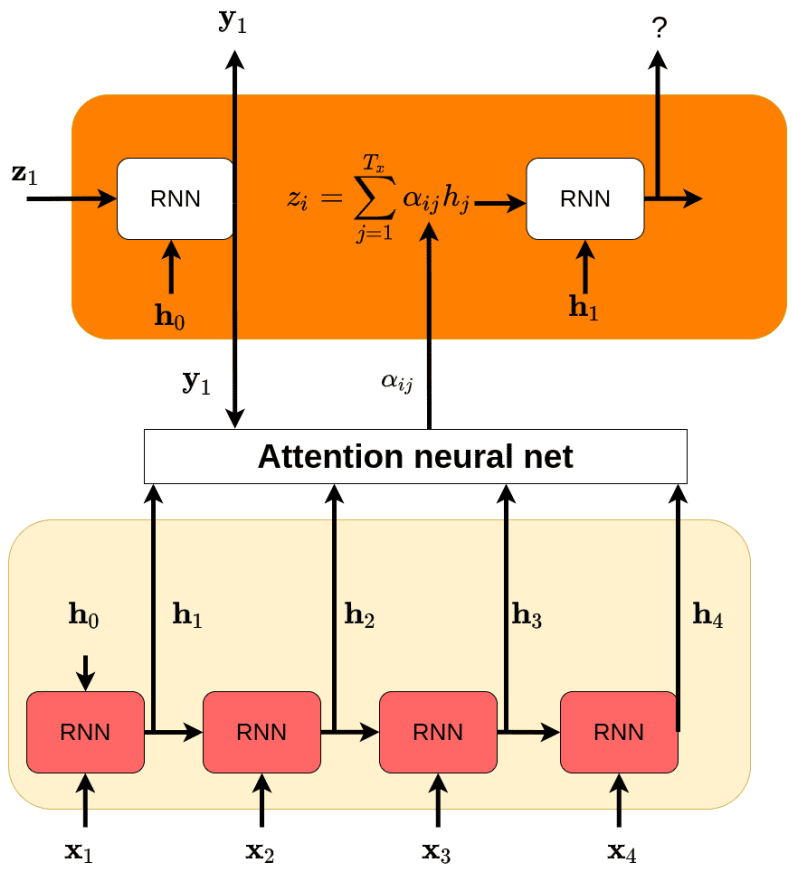
- Attention mechanism helps creating a better context vector
- It learns which information from the encoder is relevant for the decoder



Introduction to Attention Mechanism

- Attention mechanism helps creating a better context vector
- It learns which information from the encoder is relevant for the decoder





Introduction to Attention Mechanism

- The context vector **c** should have access to all parts of the input sequence instead of just the last one.
- In other words, we need to form a **direct connection** with each timestamp. We can look at all the different words at the same time and learn to "pay attention" to the correct ones depending on the task at hand.
- In the encoder-decoder:
  - Given the hidden states of the encoder at each time step h = $h_1, h_2, ..., h_n$
  - Given the previous state in the decoder y_{i-1}
  - We define an attention network that gives the attention scores for the current state (s) of the decoder

$$e_{ij} = \text{attention\_net}(s_{i-1}, h_j)$$

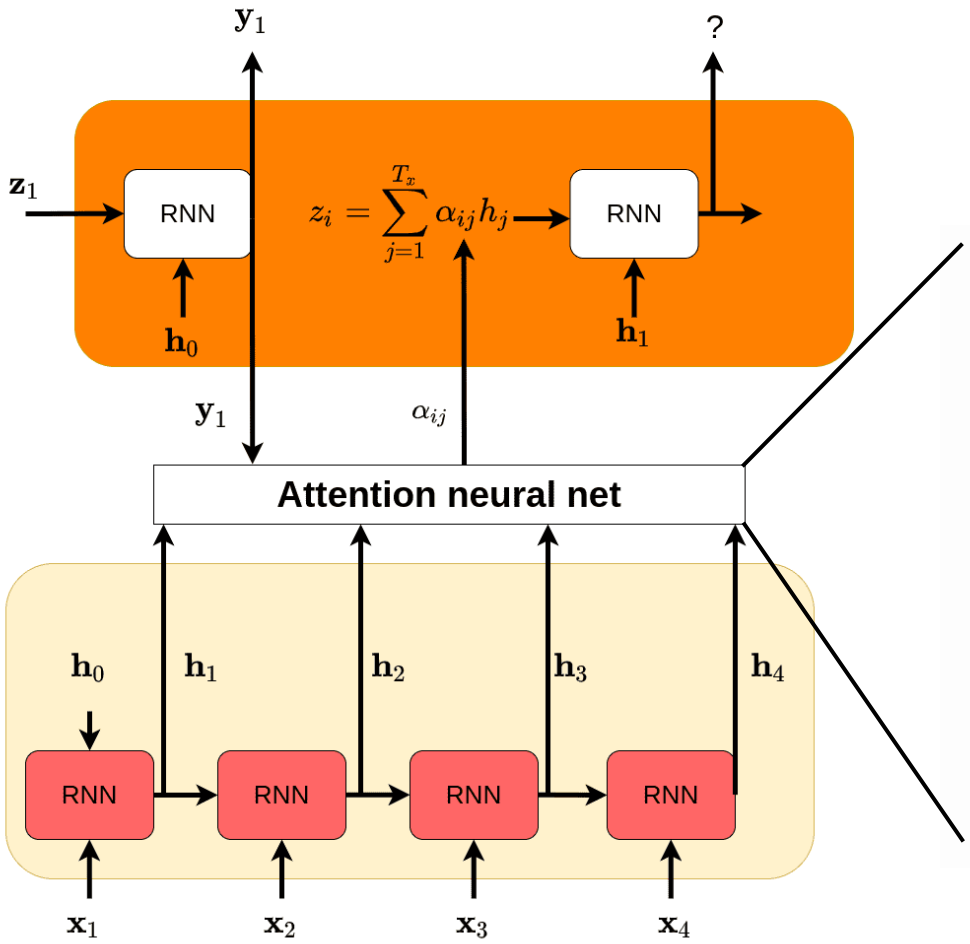
  - We convert this scores into probabilities

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$$

  - Finally, we get our new context vector c:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

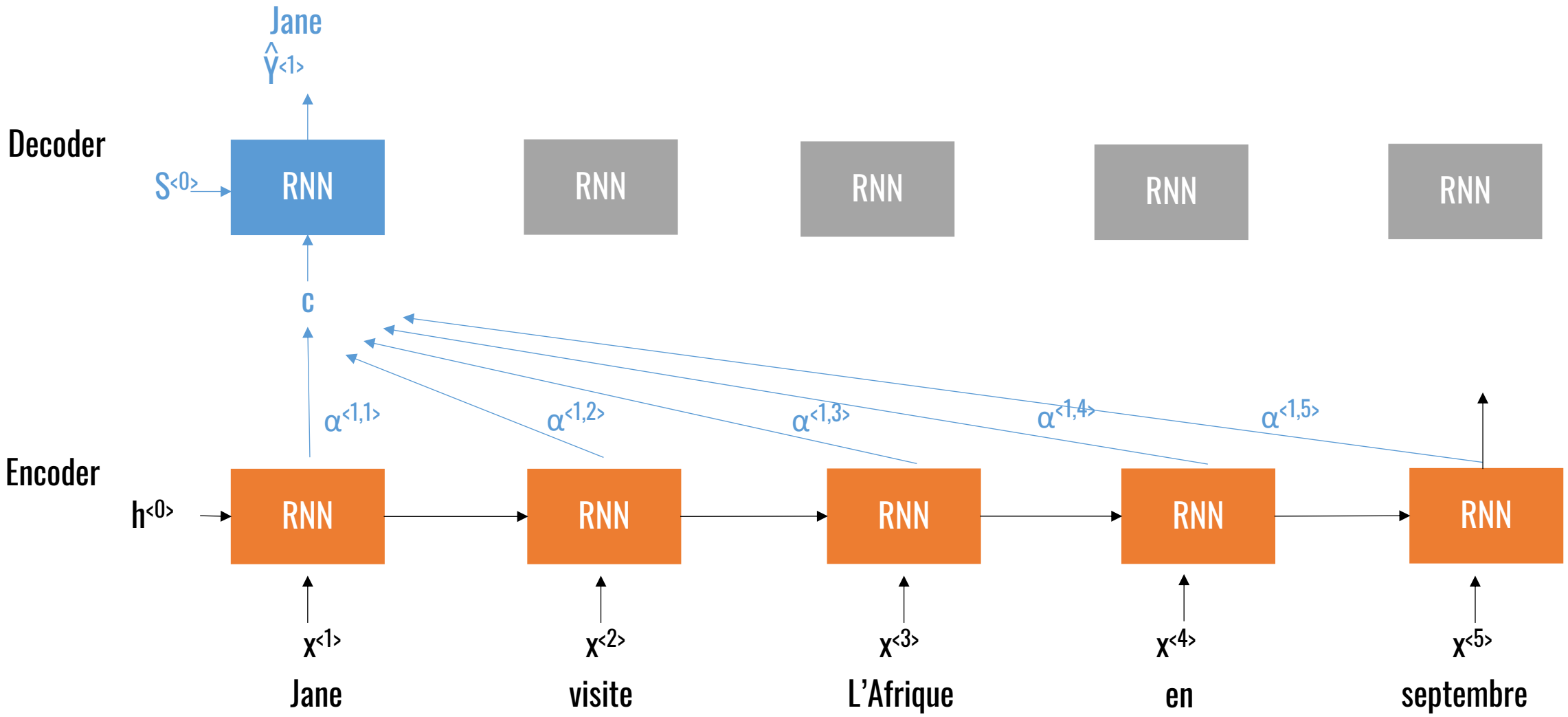
Attention Mechanism



Several ways to calculate the scores

Name	Alignment score function	Citation
Content-base attention	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \text{cosine}[\boldsymbol{s}_t, \boldsymbol{h}_i]$	Graves2014
Additive(*)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\boldsymbol{s}_t; \boldsymbol{h}_i])$	Bahdanau2015
Location-Base	$\alpha_{t,i} = \text{softmax}(\mathbf{W}_a \boldsymbol{s}_t)$ Note: This simplifies the softmax alignment to only depend on the target position.	Luong2015
General	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \mathbf{W}_a \boldsymbol{h}_i$ where $\mathbf{W}_a$ is a trainable weight matrix in the attention layer.	Luong2015
Dot-Product	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \boldsymbol{h}_i$	Luong2015
Scaled Dot-Product(^)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \frac{\boldsymbol{s}_t^\top \boldsymbol{h}_i}{\sqrt{n}}$ Note: very similar to the dot-product attention except for a scaling factor; where n is the dimension of the source hidden state.	Vaswani2017

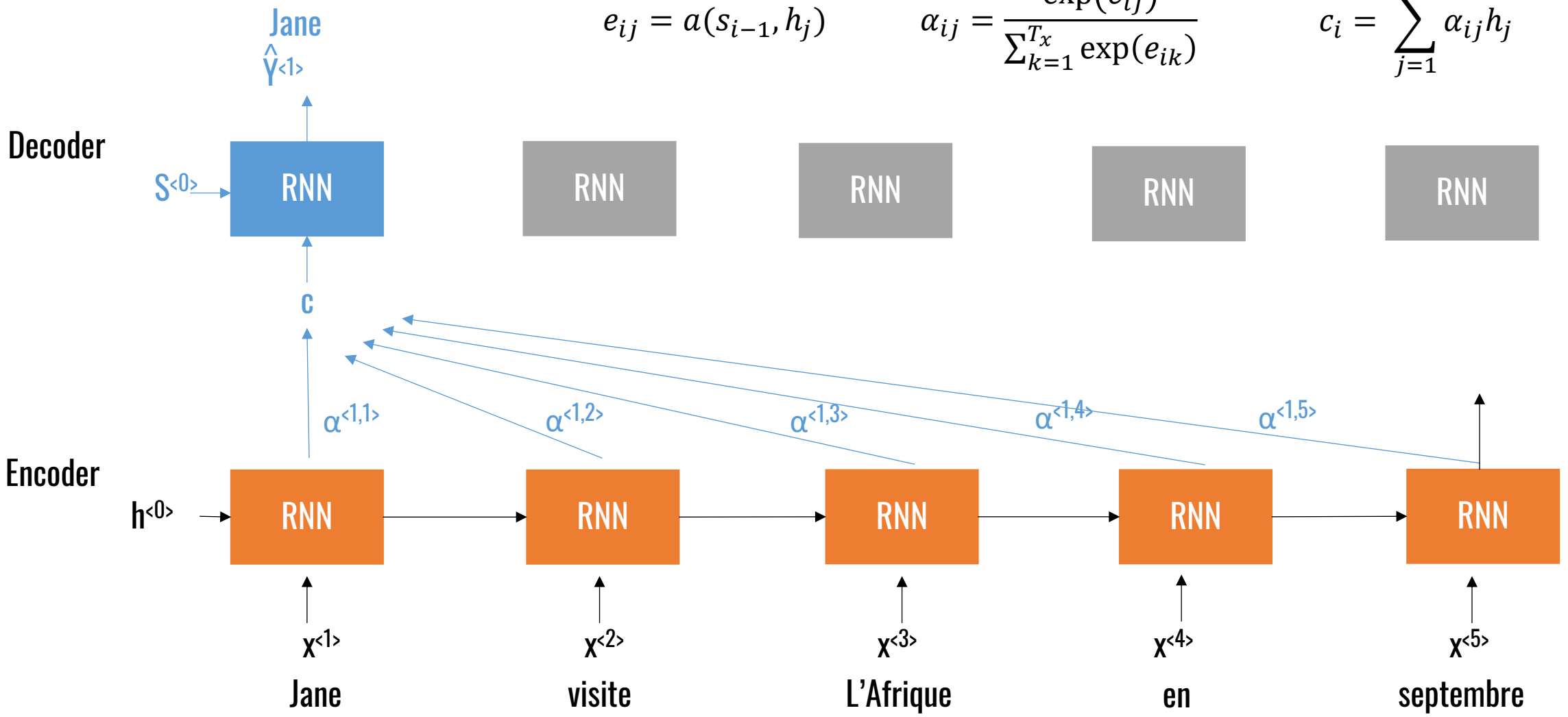
Attention Mechanism



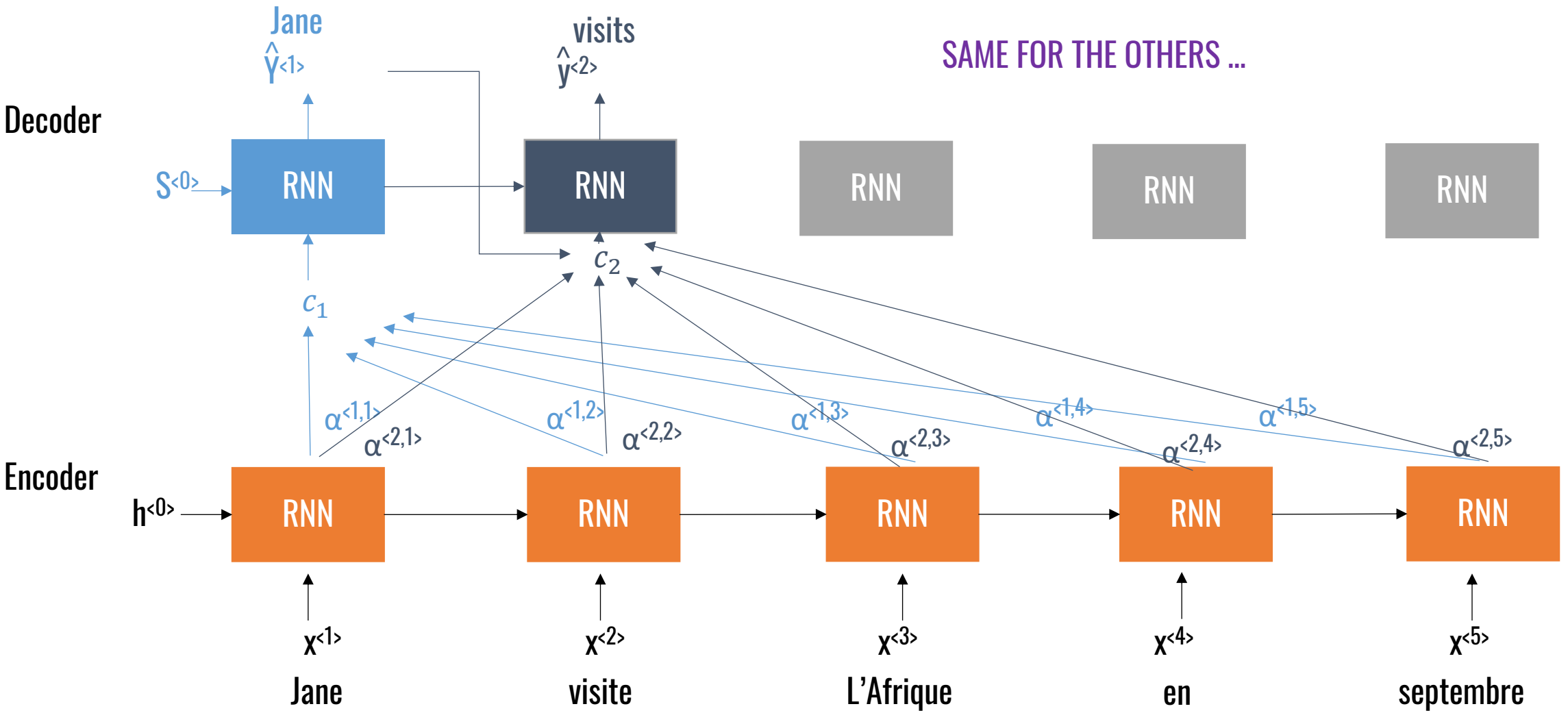
$\alpha^{<t,t'>}$ = amount of "attention" $\hat{y}^{<t>}$ should pay to $h^{<t'>}$

Attention Mechanism

$$e_{ij} = a(s_{i-1}, h_j) \qquad \alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} \qquad c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$



Attention Mechanism



Next Lecture

We will continue next lecture with Attention and Transformers…

Summary

- **Sequential data** is important because it carries temporal information and context

- **Recurrent Neural Networks:**
  - Sequence based models
  - Handle variable length sequences
  - Track dependencies
  - Maintain the order of the input
  - Share parameters across sequences
- RNNs have limitations: **vanishing gradients and short memory**

- Other architectures like **LSTM and GRU** improve the limitations of RNNs
  - Include gates to control the flow of information

- **Seq2Seq** models are encoder-decoder based architectures
  - The context vector from the encoder is limited
- **Attention mechanism provides a better context by allowing the network to pay attention to every part of the input** / have access to all hidden states
  - It computes a score / weight that tells the relevance of each part

Resources

Books:

- Courville, Goodfellow, Bengio: Deep Learning
  Freely available: https://www.deeplearningbook.org/
- Zhang, Aston and Lipton, Zachary C. and Li, Mu and Smola, Alexander J.: Dive into Deep Learning
  Freely available: https://d2l.ai/

Courses:

- Deep Learning specialization by Andrew NG
- https://www.coursera.org/specializations/deep-learning

Further Links + Resources

- Beam search: https://towardsdatascience.com/foundations-of-nlp-explained-visually-beam-search-how-it-works-1586b9849a24
- BLEU score: https://cloud.google.com/translate/automl/docs/evaluate#bleu
- https://theaisummer.com/attention/
- Coursera Deep Learning Specialization

That’s all for today!