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

Deep Network Development

Lecture 7.

Recurrent Neural Networks

Budapest, 28th March 2025

1 RNNs and Embeddings

2 LSTM, GRU & Seq2Seq

3 Attention Mechanism

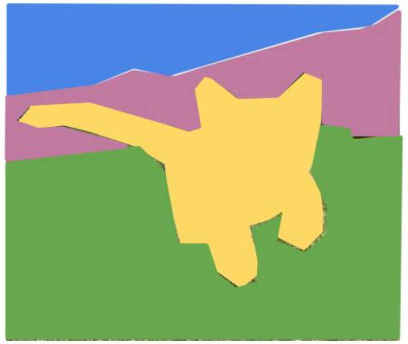
Supervised Learning tasks

Classification

CAT

Single Object

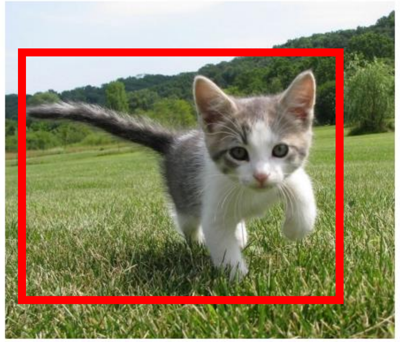
Semantic Segmentation



GRASS, CAT, TREE, SKY

No objects, just pixels

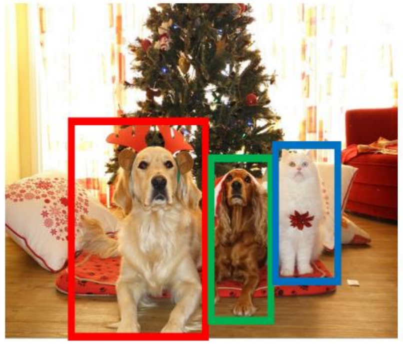
Classification + Localization



CAT

Single Object

Object Detection



DOG, DOG, CAT

Multiple Objects

Instance Segmentation

DOG, DOG, CAT

Multiple Objects

Deep Network Development

Lecture 7.

RNNs and Embeddings

Budapest, 28th March 2025

1 RNNs and Embeddings | 2 LSTM, GRU & Seq2Seq | 3 Attention Mechanism

Sequential Data Processing

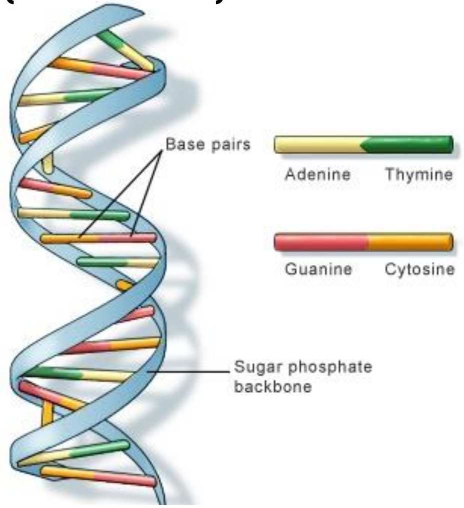
Sequential data

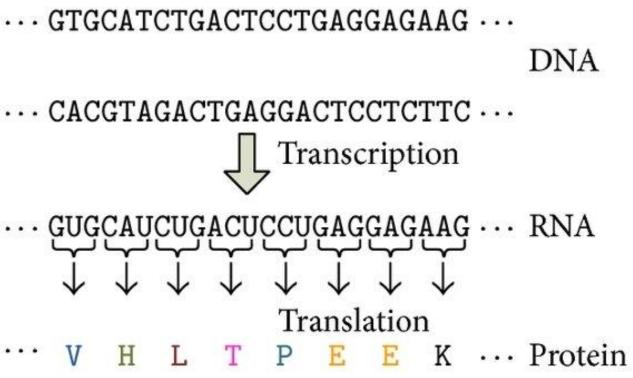
Text – sequence of words / characters

Speech – sequence of signals / acoustic features

Video – sequence of images (frames)

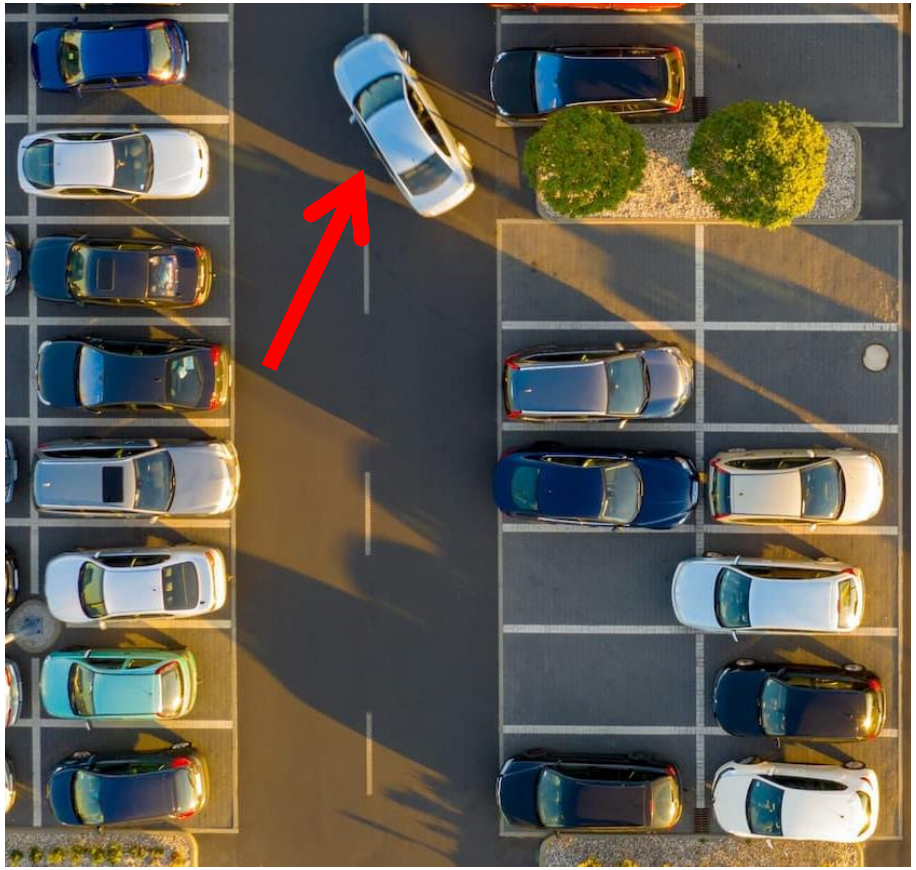
DNA – sequence of symbols (nucleotides)





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."

Sequential Data Processing

Name entity recognition

Input X: Harry Potter and Hermione Granger invented a new spell.

$X^{<1>}$ $X^{<2>}$ $X^{<3>}$ … $X^{<t>}$ … $X^{<9>}$

Output Y: 1 1 0 1 1 0 0 0 0

$Y^{<1>}$ $Y^{<2>}$ $Y^{<3>}$ … $Y^{<t>}$ … $Y^{<9>}$

Sequential Data Processing

Representing words as One hot vectors

Dataset: (X,Y)

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

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

Input X: Harry Potter and Hermione Granger invented a new spell.

$X^{<1>}$ $X^{<2>}$ $X^{<3>}$ ... $X^{<t>}$ ... $X^{<9>}$

Representation:

Harry = [0, 0, 0, ..., 1, 0, 0, ..., 0]

Position: 0 4075 10000

Potter = [0, 0, 0, ..., 0, ..., 1, 0, ..., 0]

Position: 0 6883 10000

Introduction to Recurrent Neural Network

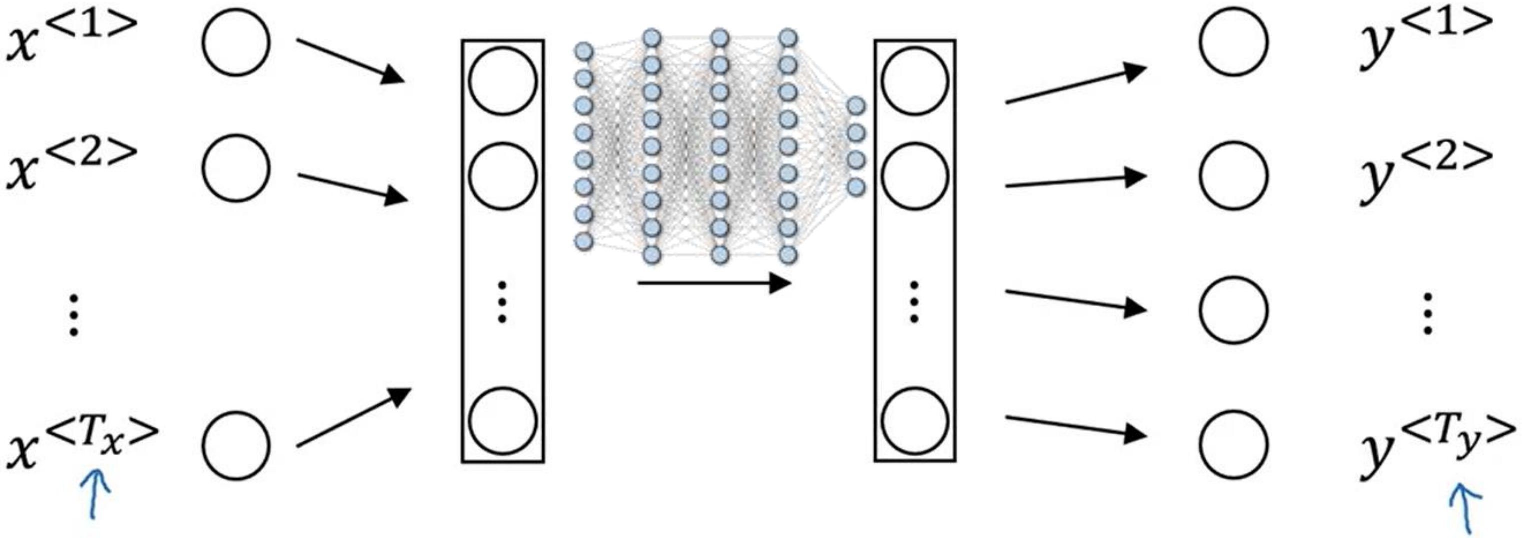
What should we consider?

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

Why not a standard network?

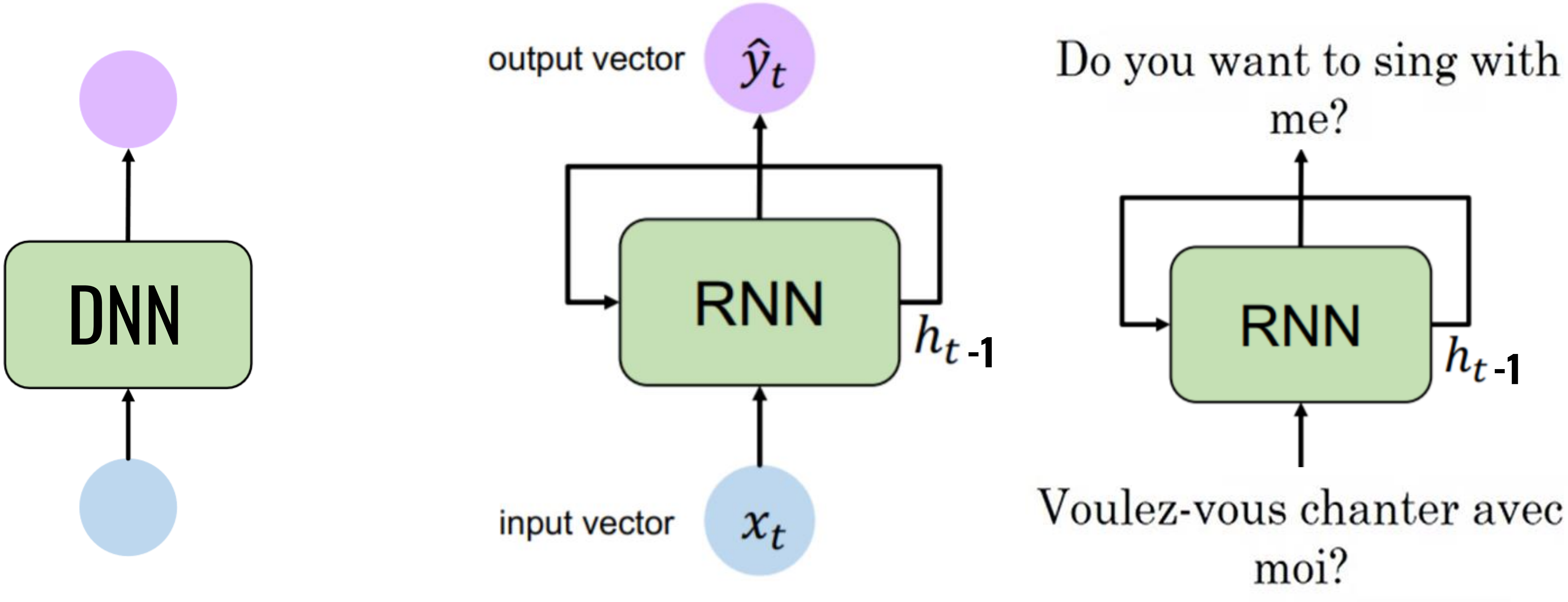


Problems:

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

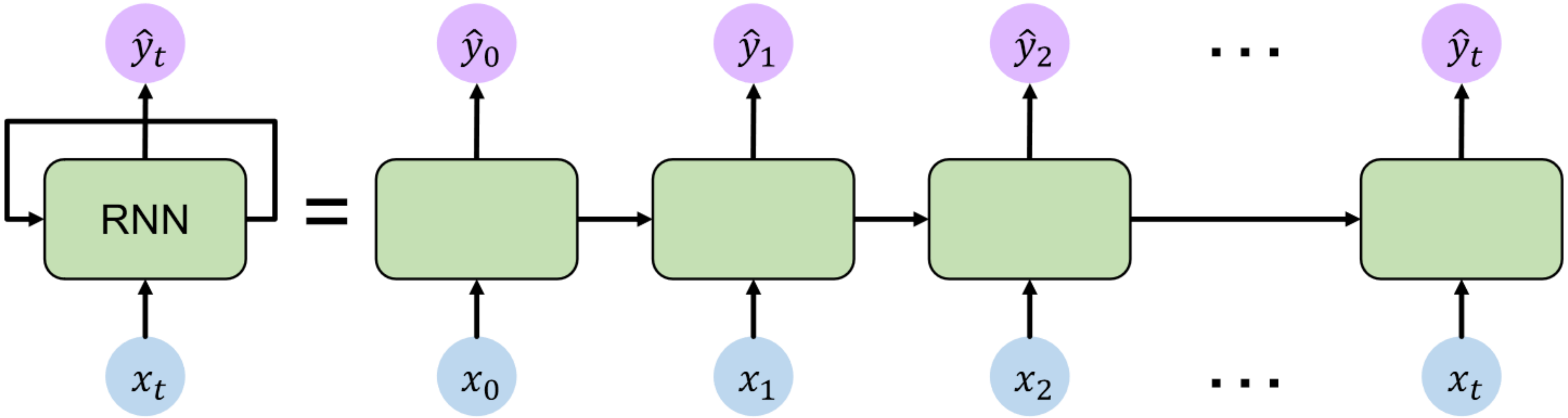
Introduction to Recurrent Neural Network

What about Recurrent Neural Networks (RNNs)?



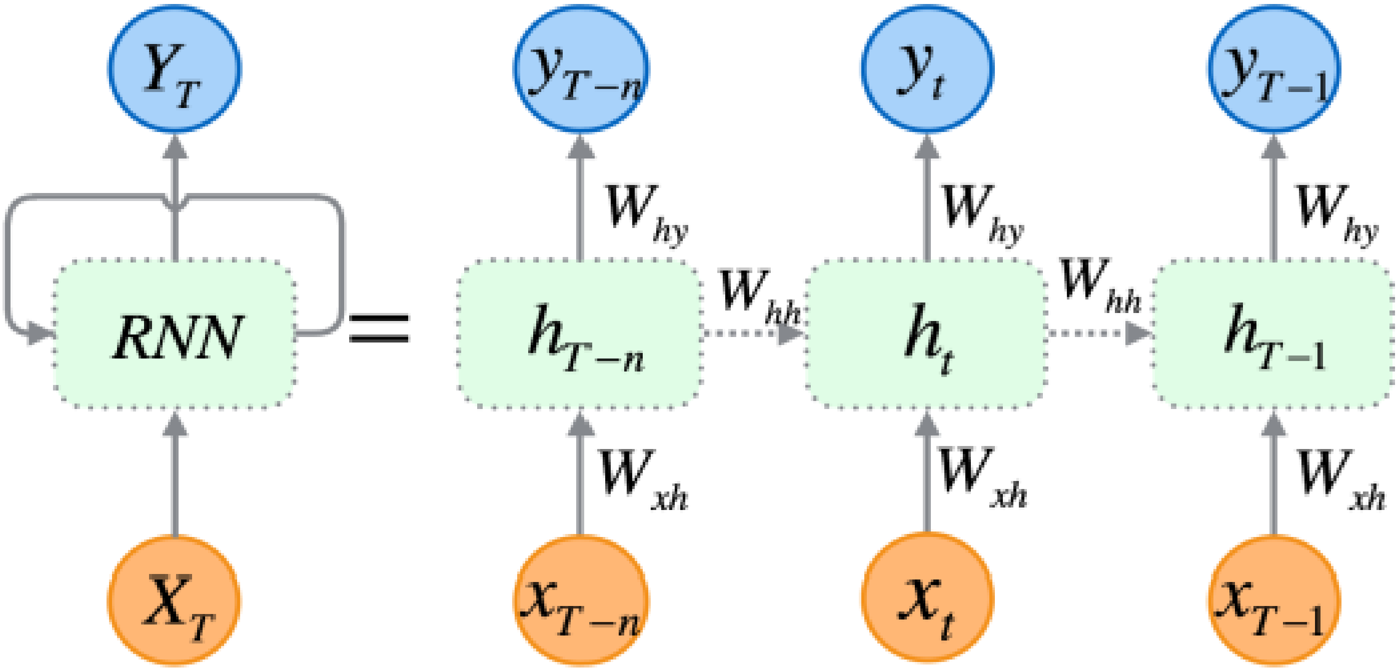
Introduction to Recurrent Neural Network

RNN (unrolled version)

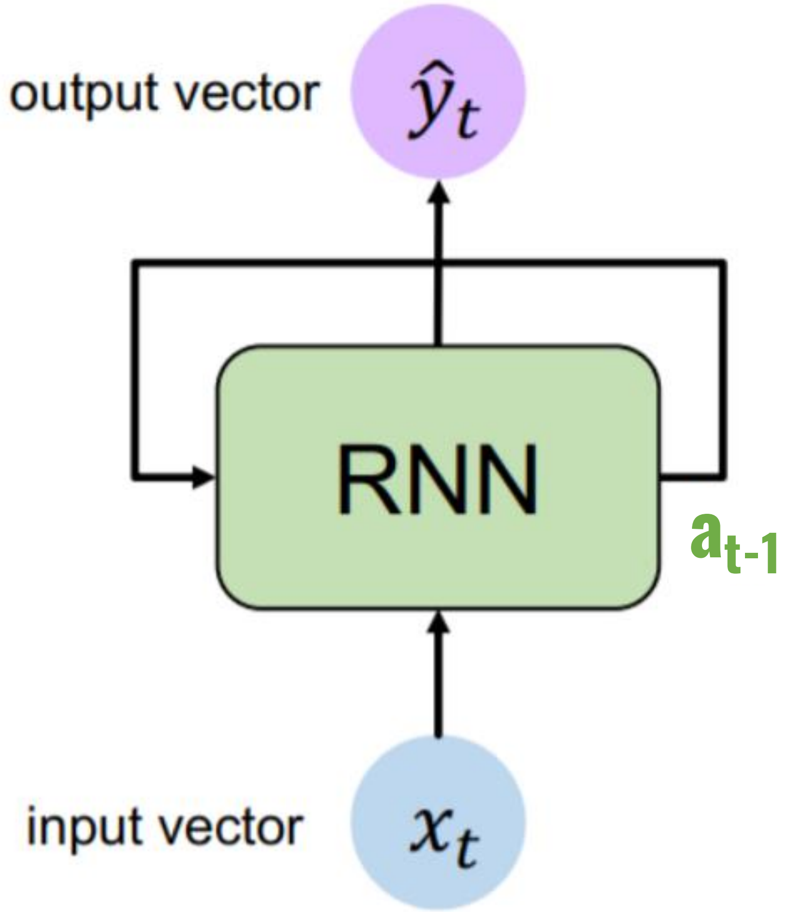


Introduction to Recurrent Neural Network

RNN (unrolled version)



Introduction to Recurrent Neural Network



Hidden state $a_t = \tanh(W_{aa}\, a_{t-1} + W_{ax}\, x_t + b_a)$

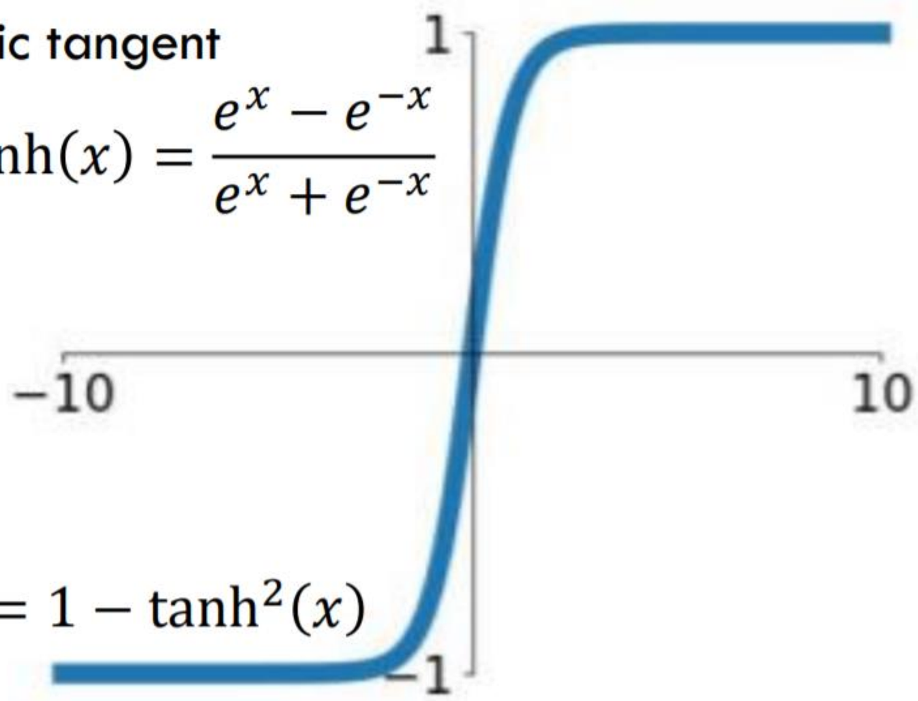
Output vector $\hat{y}_t = \tanh(W_{ya}\, a_t + b_y)$

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.

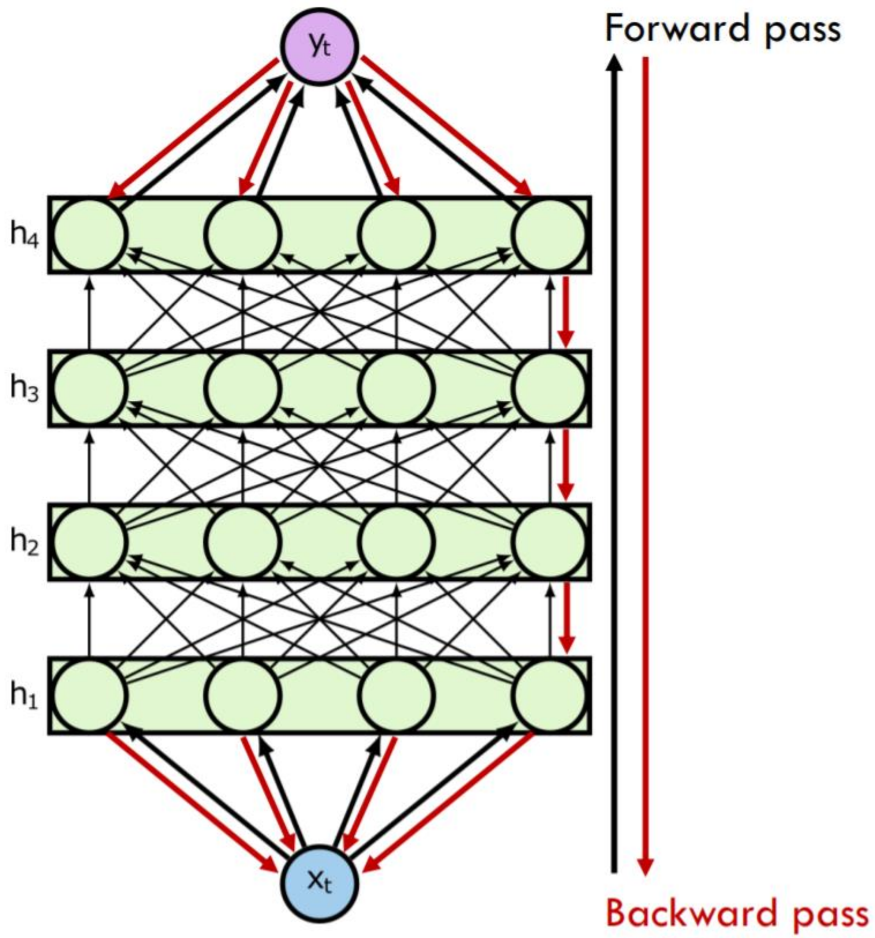
Cons:

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

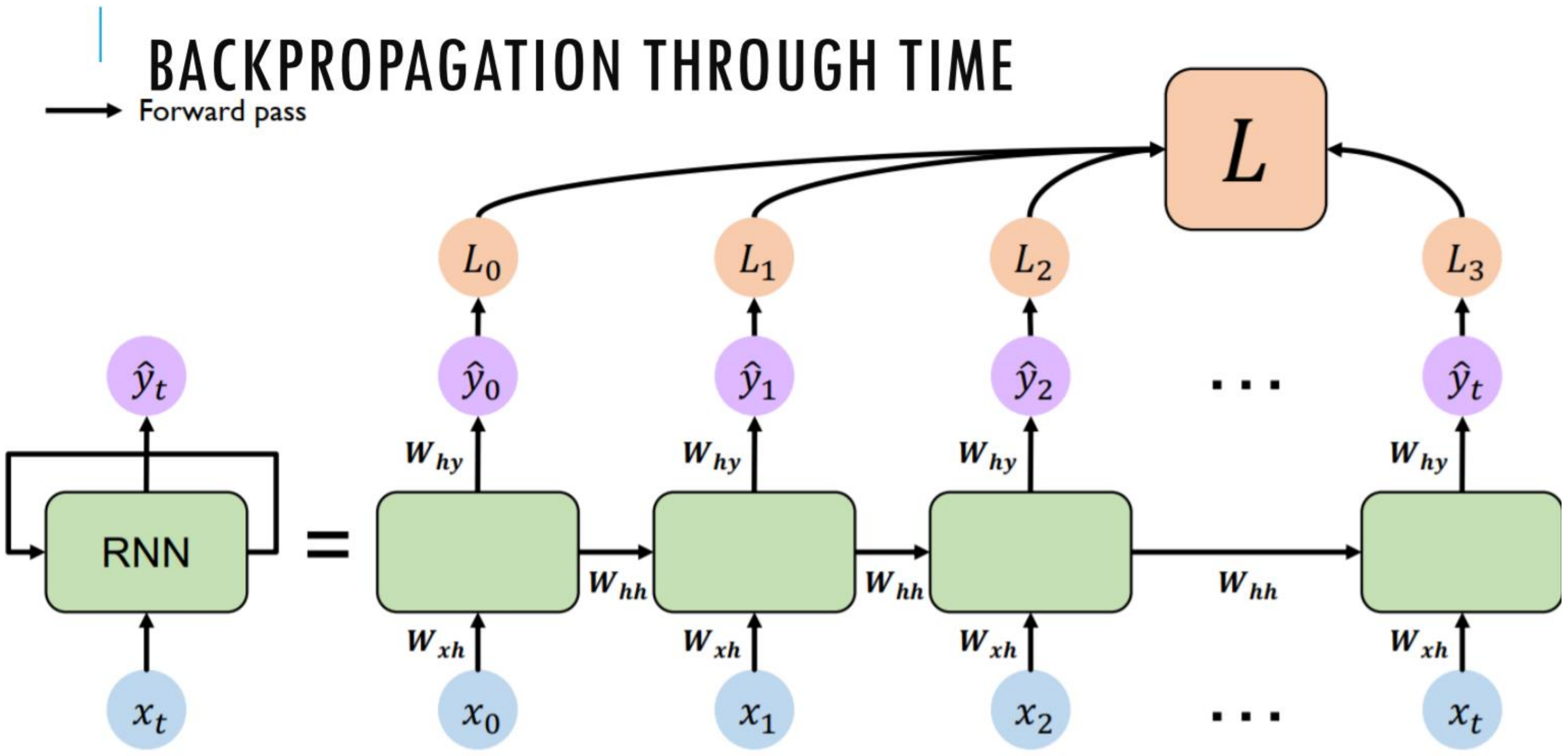
Introduction to Recurrent Neural Network

Backpropagation

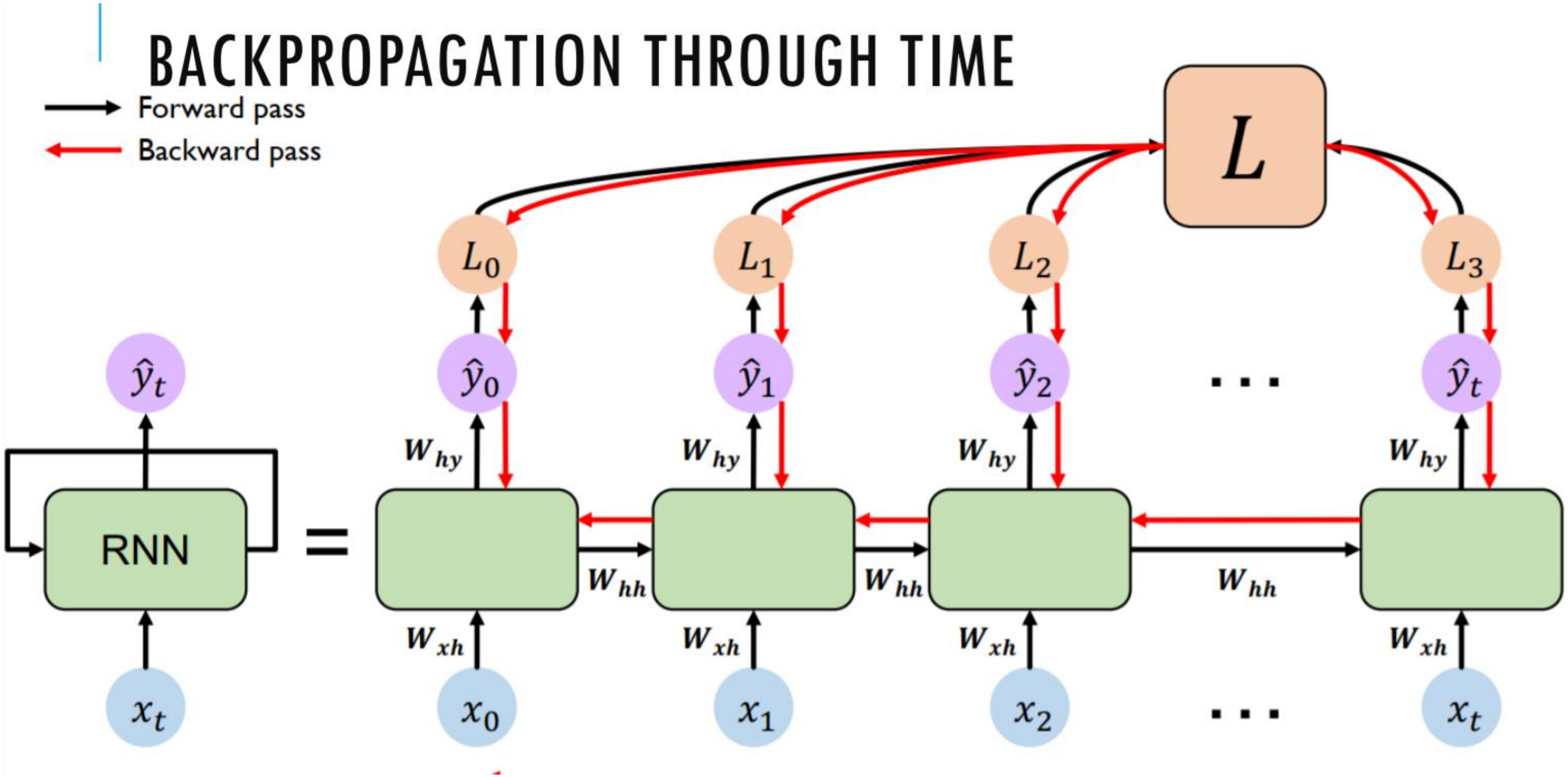
1. Calculate the forward pass
2. Determine the loss
3. Take the partial derivative (gradient) of the loss respect to each parameter
4. Shift the parameters to minimize the loss



Introduction to Recurrent Neural Network

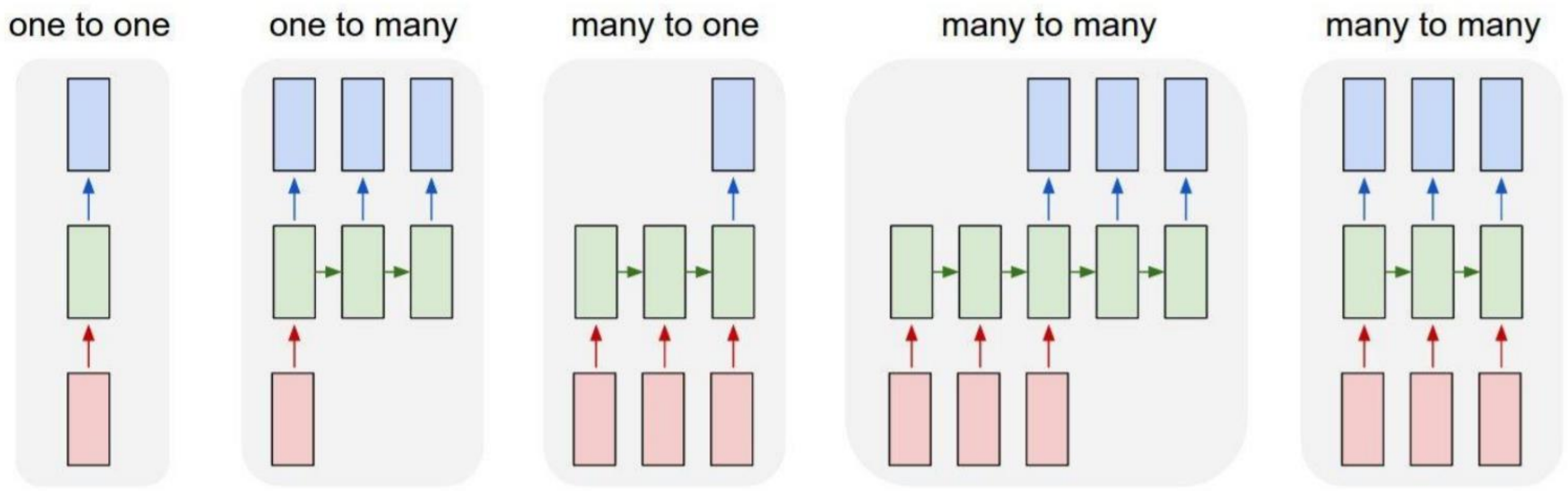


Introduction to Recurrent Neural Network



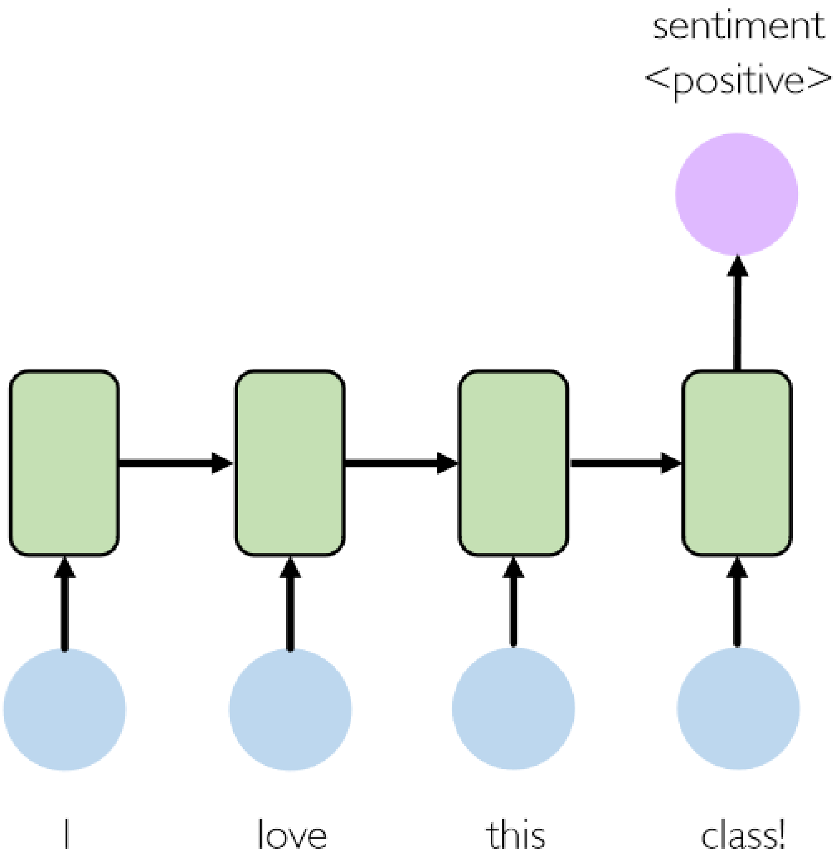
Tasks using RNNs

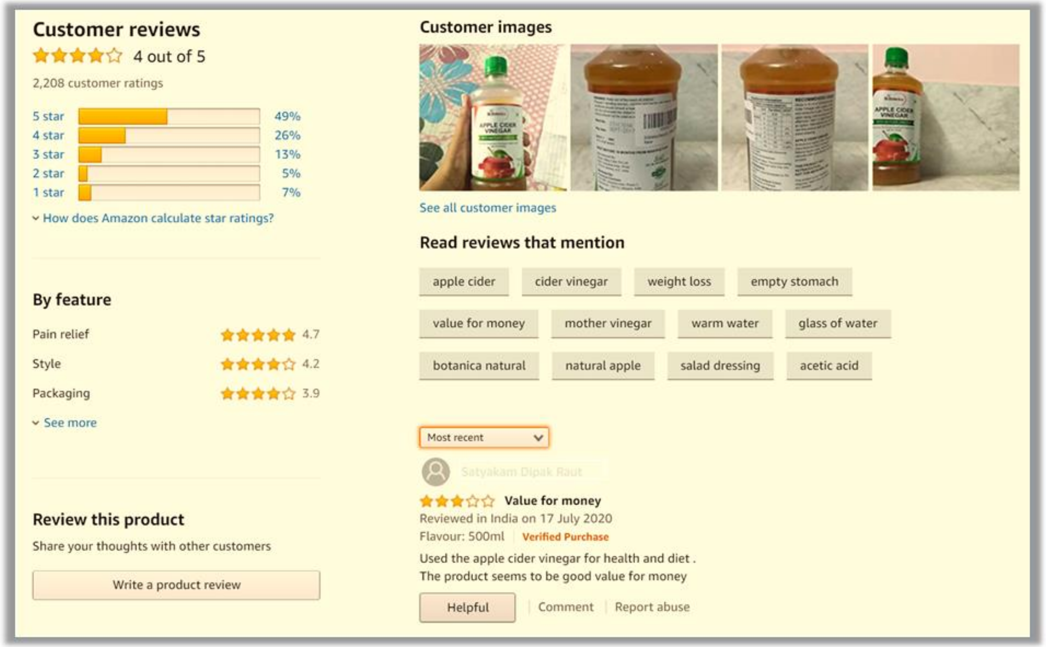
RNNS COME IN MANY FORMS



Tasks using RNNs

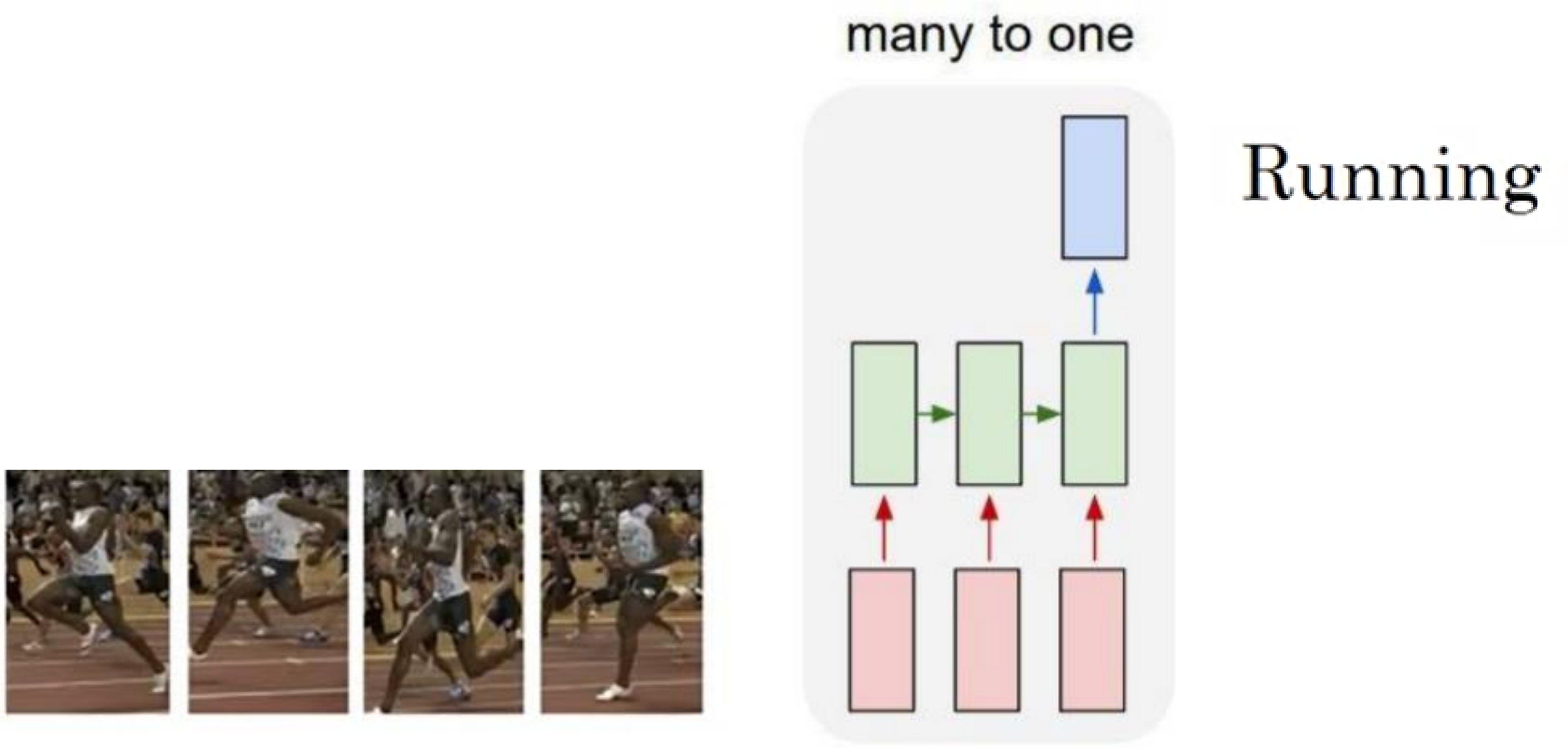
RNN Many to One – Sentiment Analysis





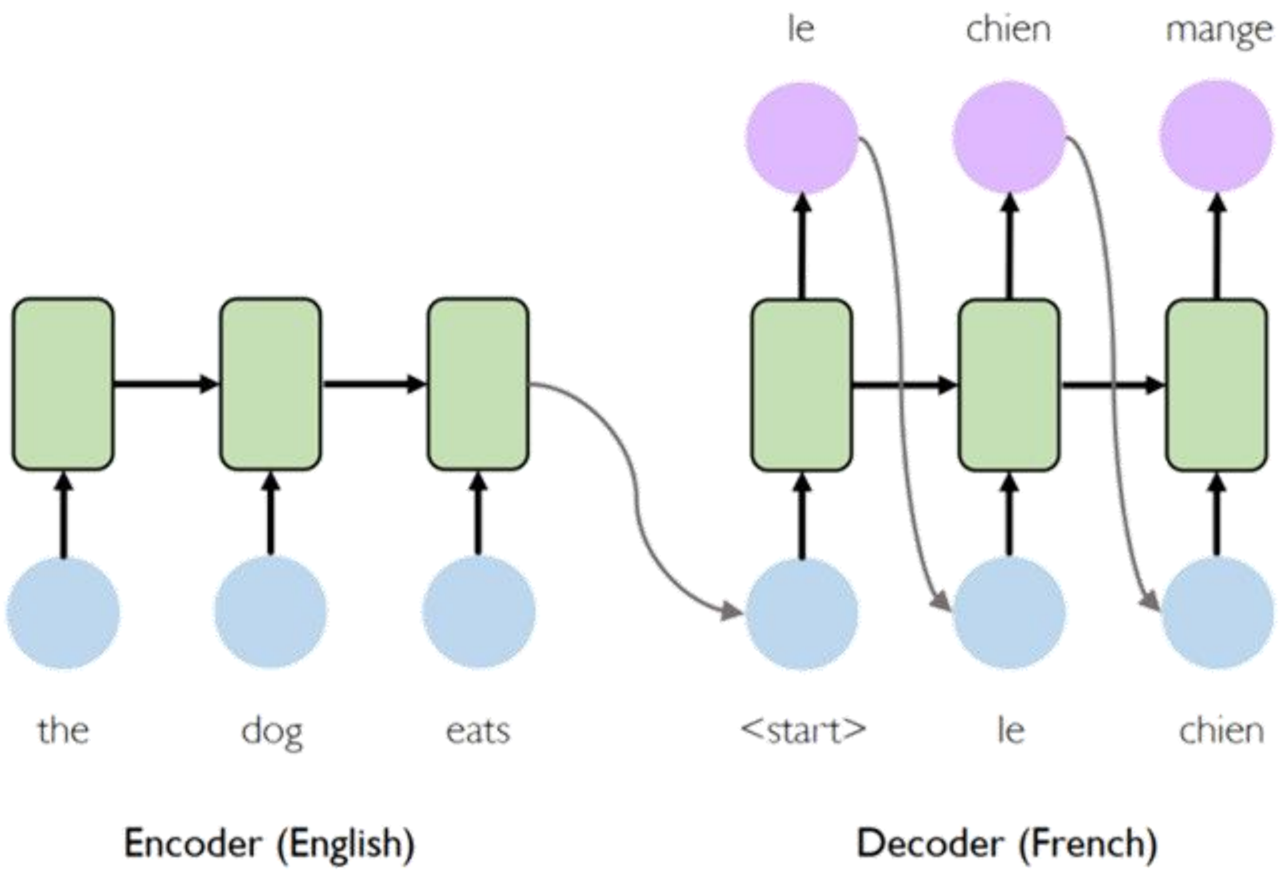
Tasks using RNNs

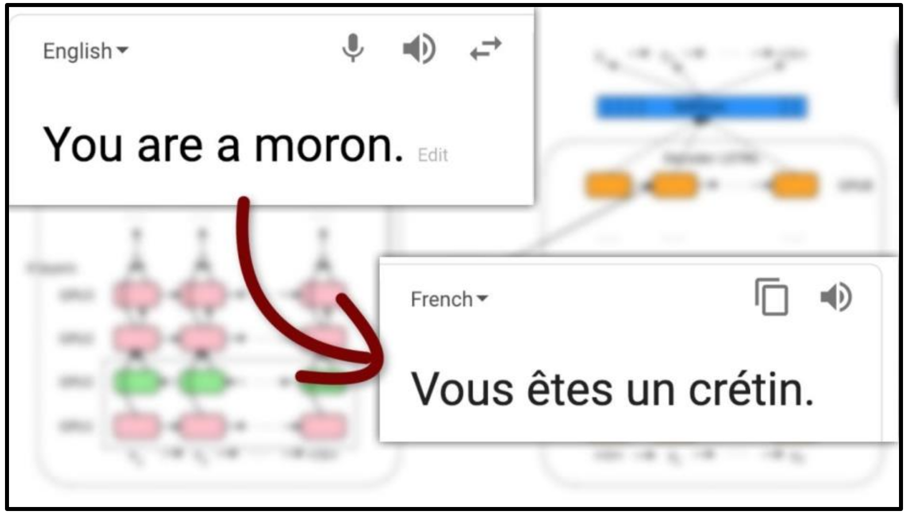
RNN Many to One – Video Activity Recognition



Tasks using RNNs

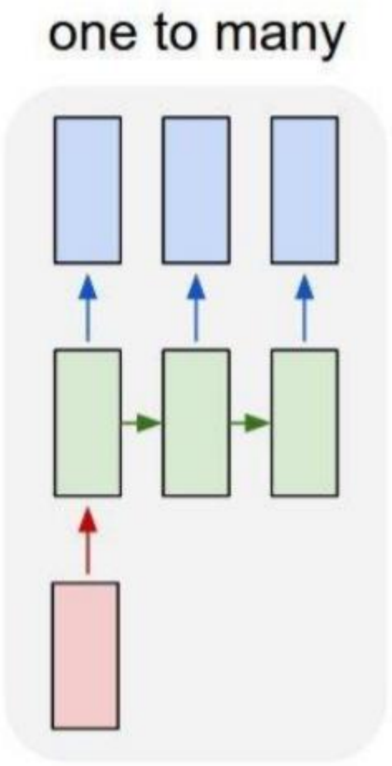
RNN Many to Many – Machine Translation

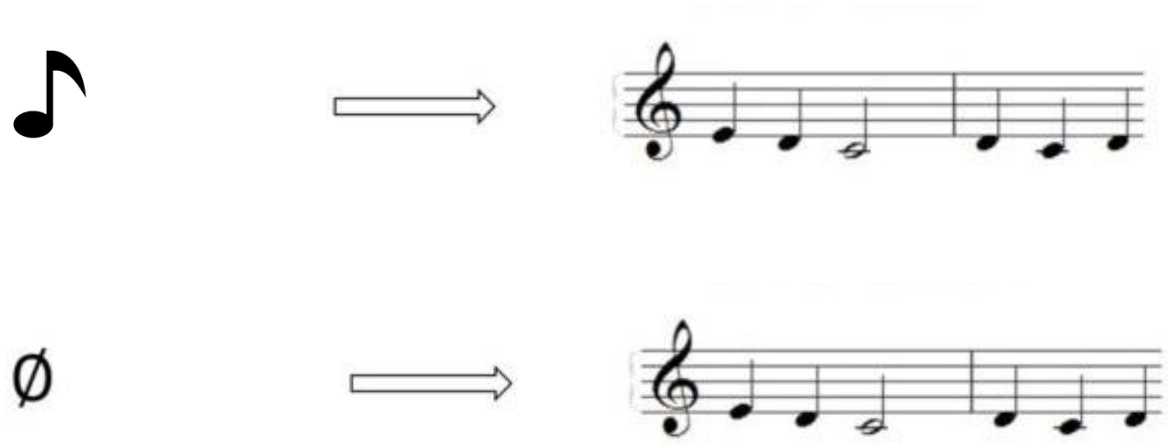




Tasks using RNNs

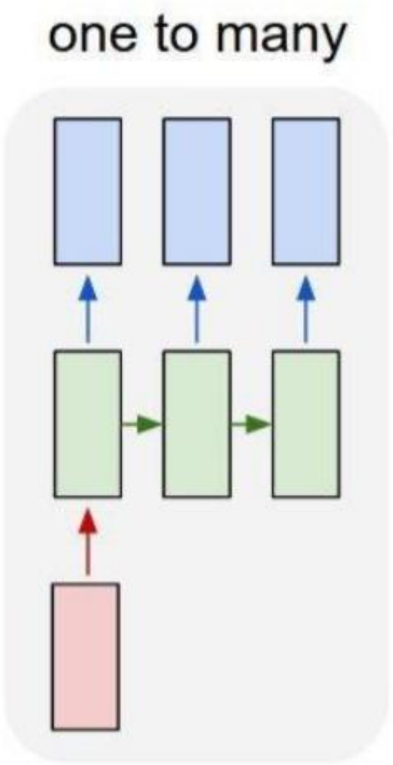
RNN One to Many – Music Generation

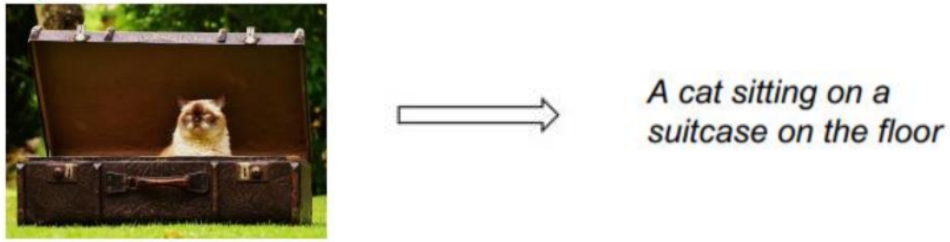


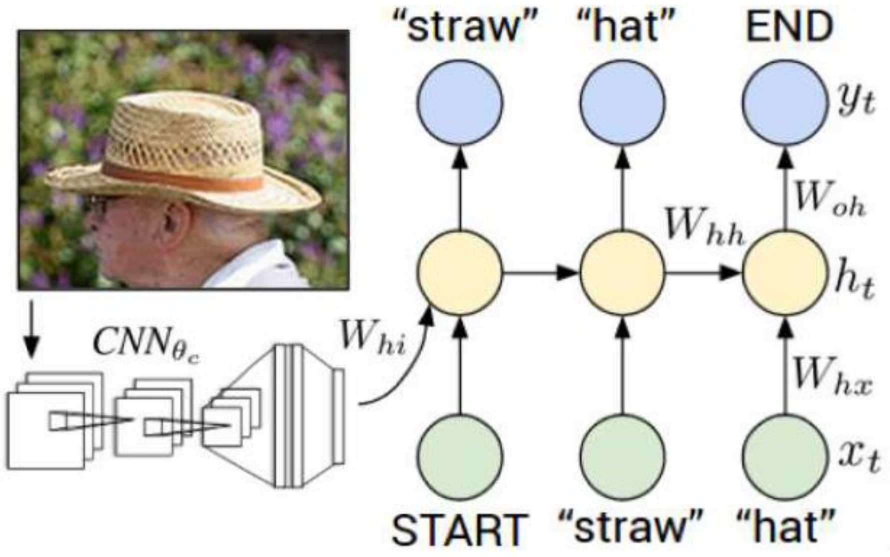


Tasks using RNNs

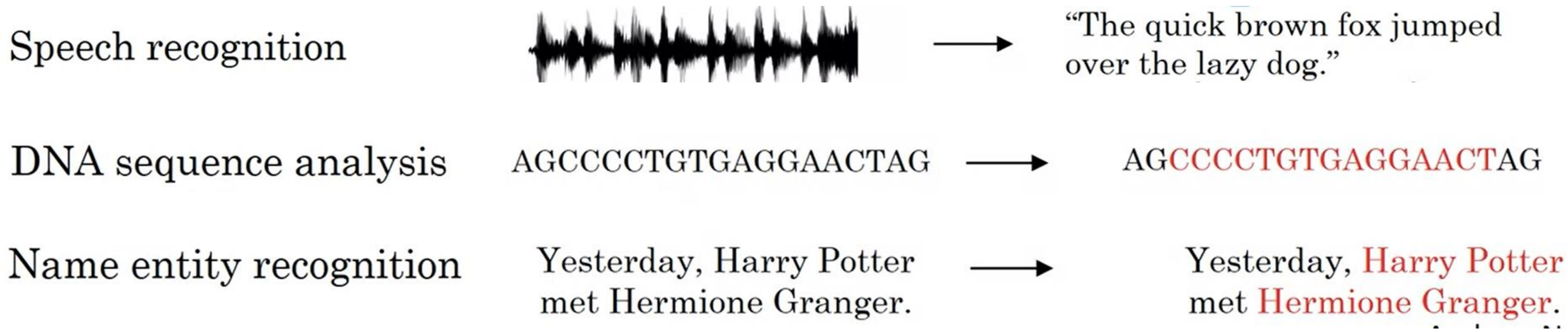
RNN One to Many – Image Caption







Tasks using RNNs



…

Time series

Q&A

And more….

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

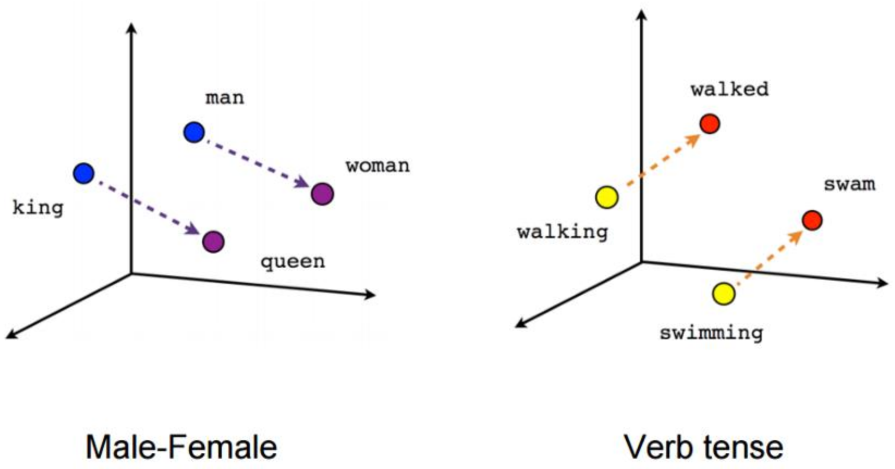
Featurized representation: Word Embedding

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**
…	…	…	…	…	…	…

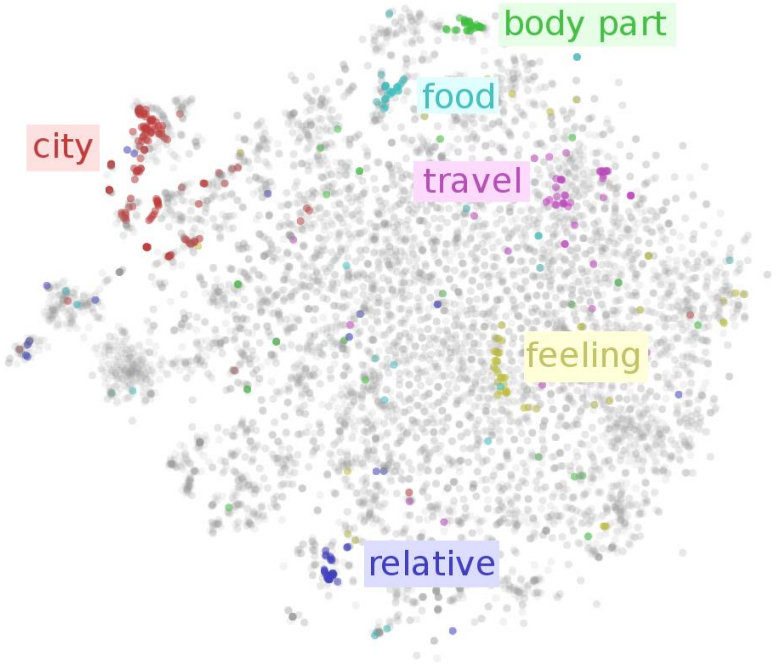
Embeddings: Word representation

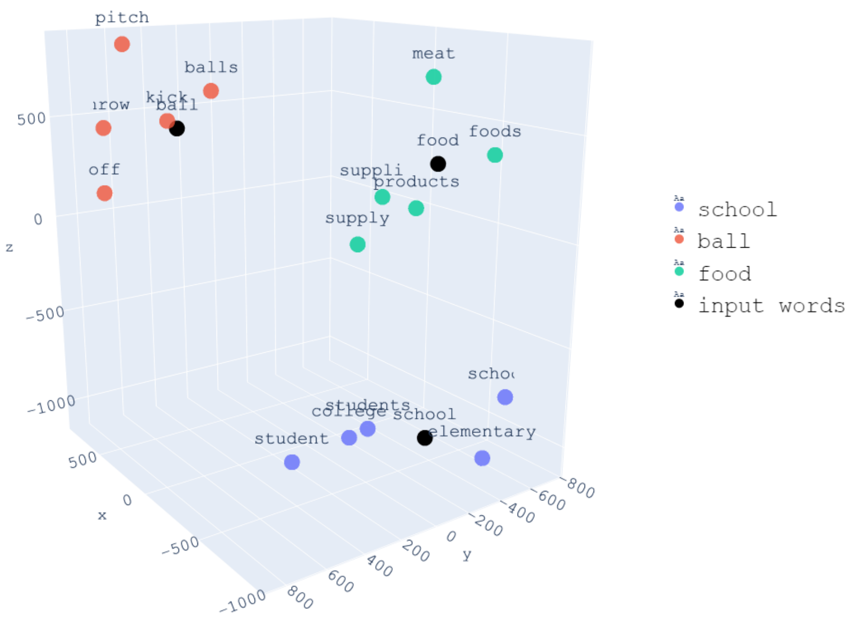
Featurized representation: Word Embedding

The word embeddings are learned with training.

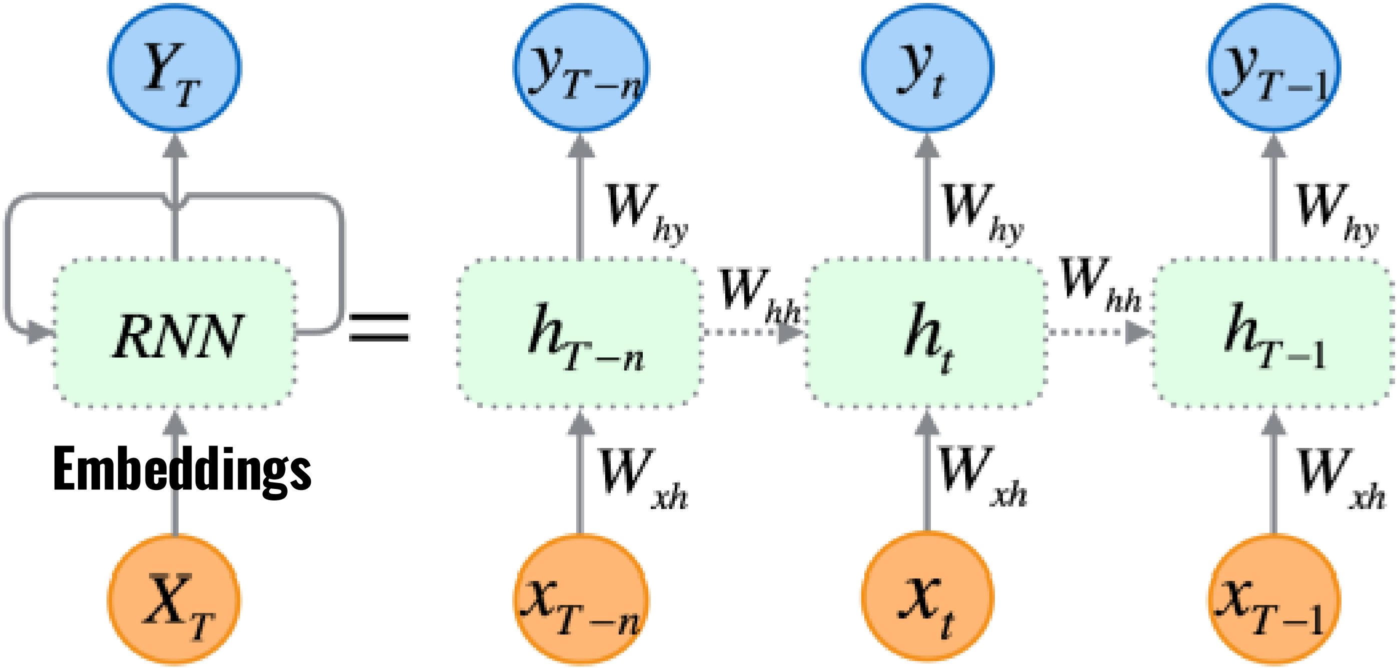
Therefore, in practice, the features aren't that understandable.

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





Embeddings: Word representation



Embeddings: Word representation

Embeddings can be used to represent other types of data:

Speech: speaker embeddings

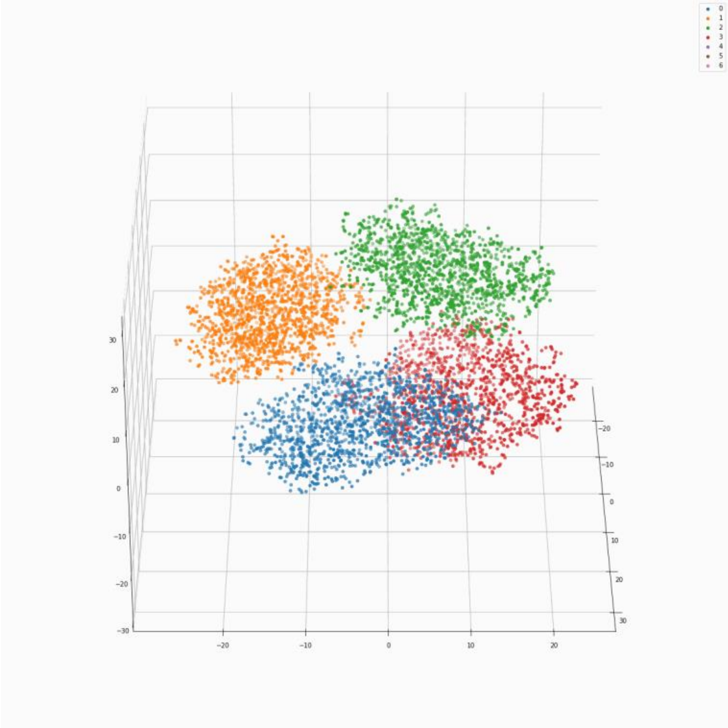
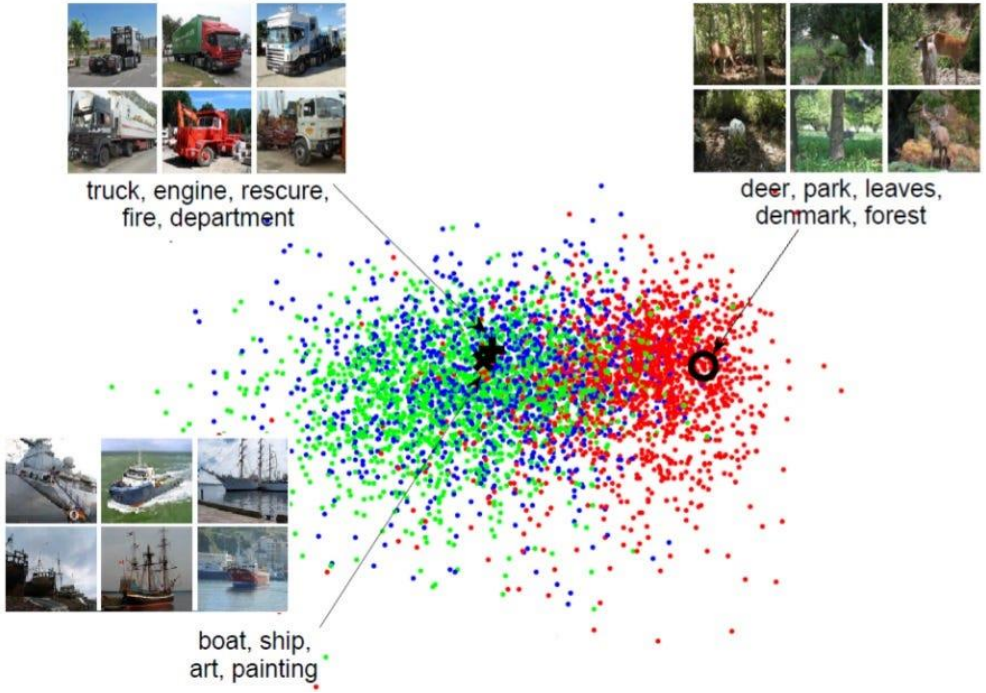
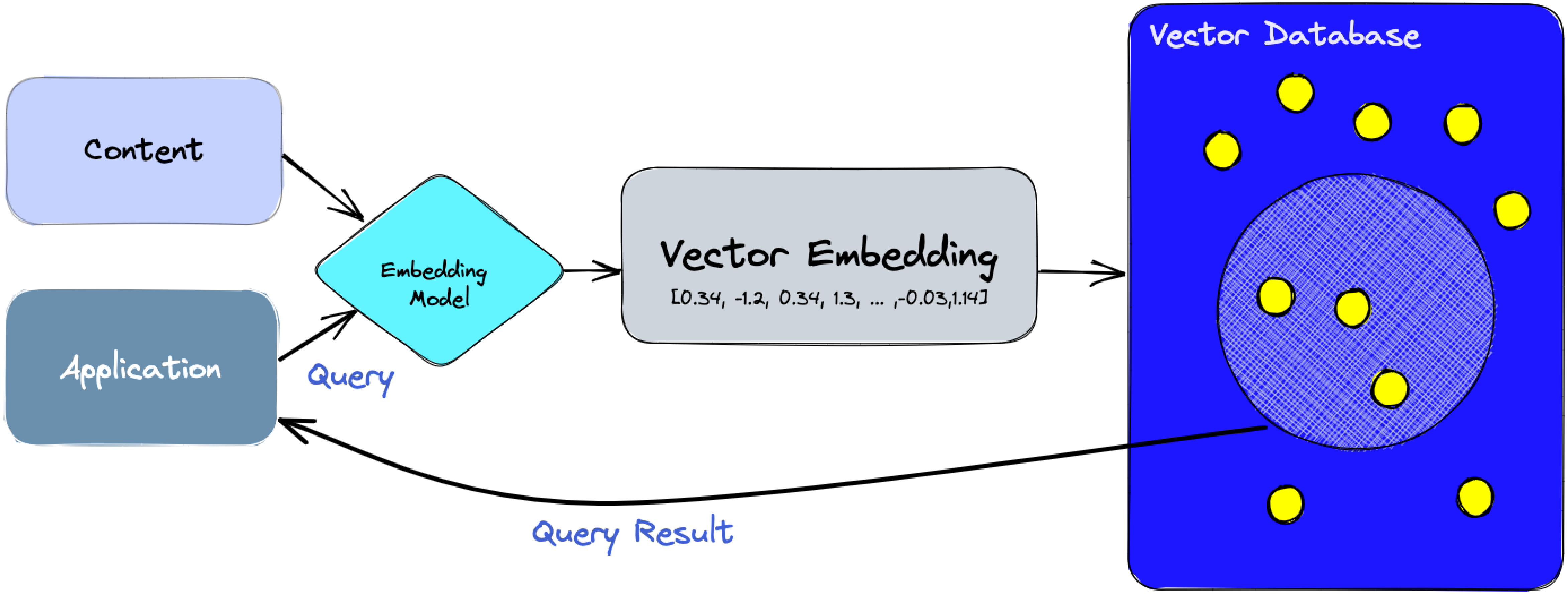


Image: image embeddings



Embeddings: Word representation



How does Google Translate work? https://www.pinecone.io/learn/vector-database/

Deep Network Development

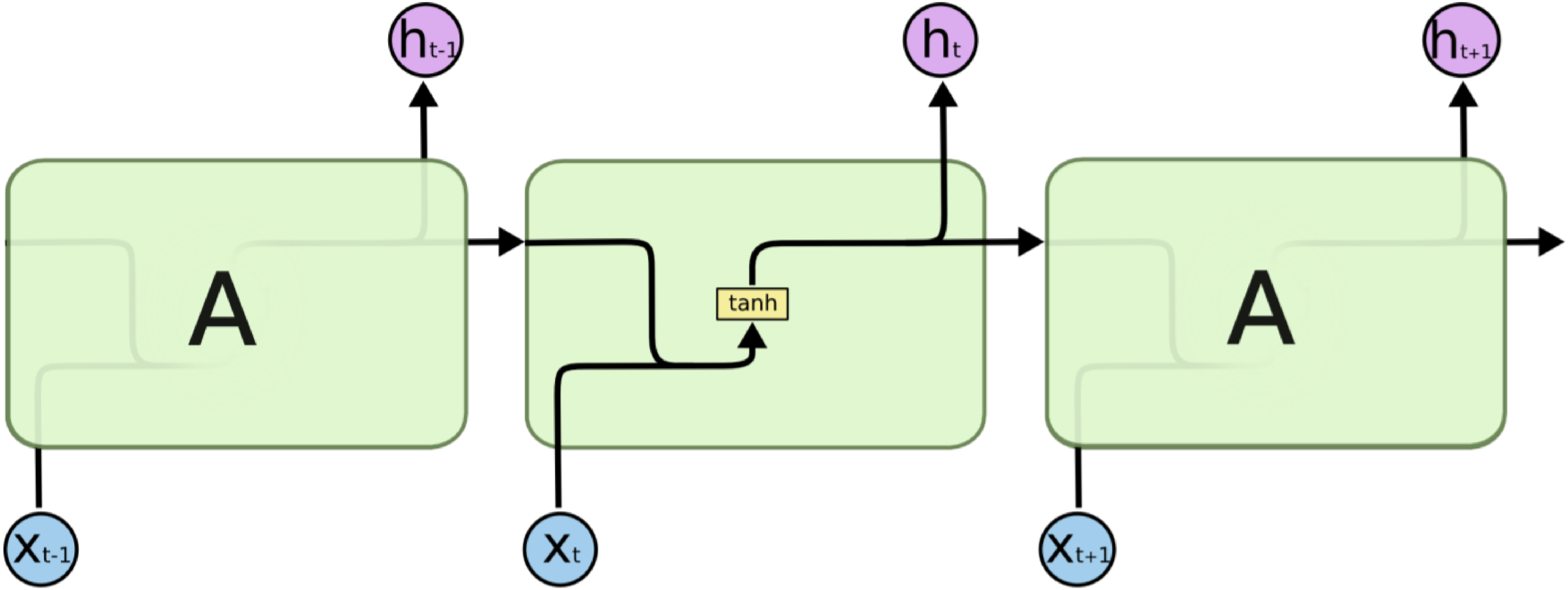
Lecture 7.

LSTM, GRU & Seq2Seq

Budapest, 28th March 2025

1 RNNs and Embeddings | **2 LSTM, GRU & Seq2Seq** | 3 Attention Mechanism

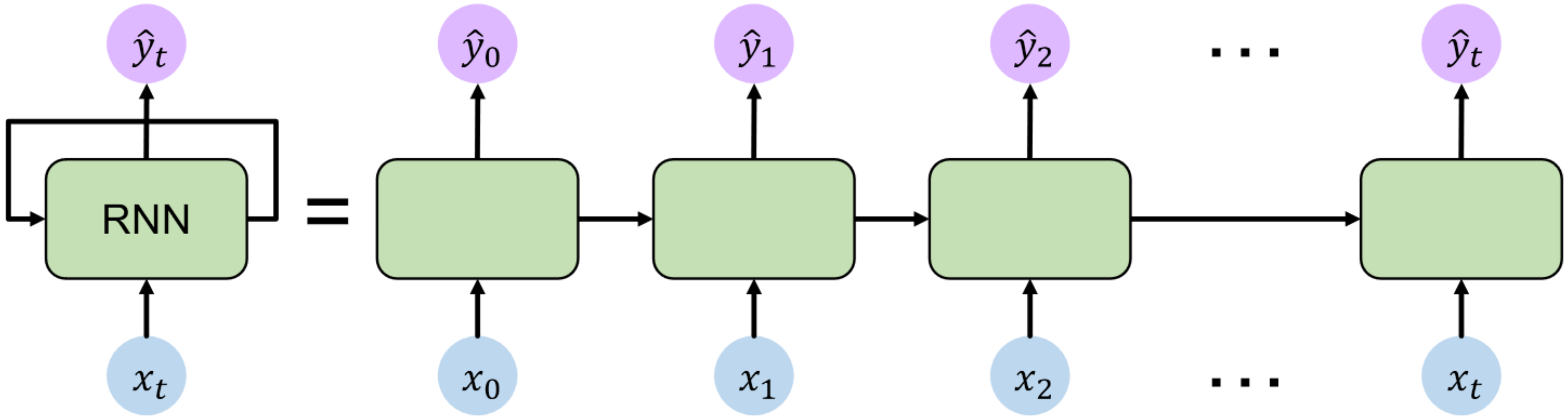
Vanilla RNNs



Vanilla RNNs

Problems with vanilla RNN

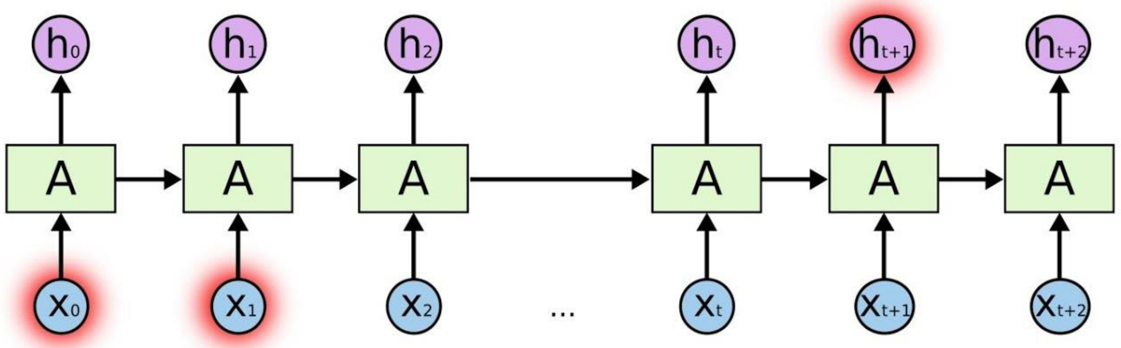
- Vanishing gradients
- Short term dependency



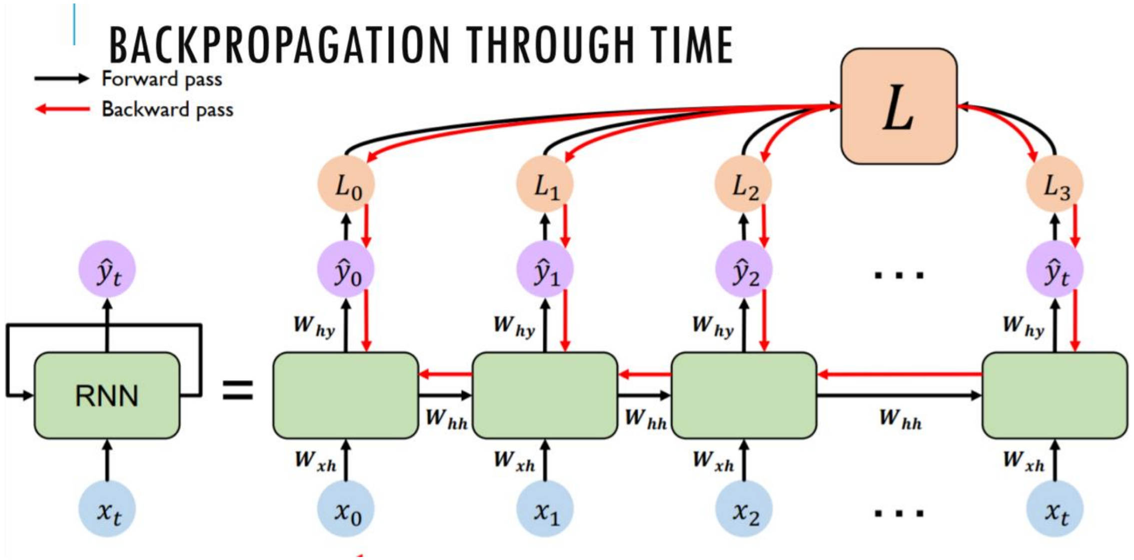
Vanilla RNNs

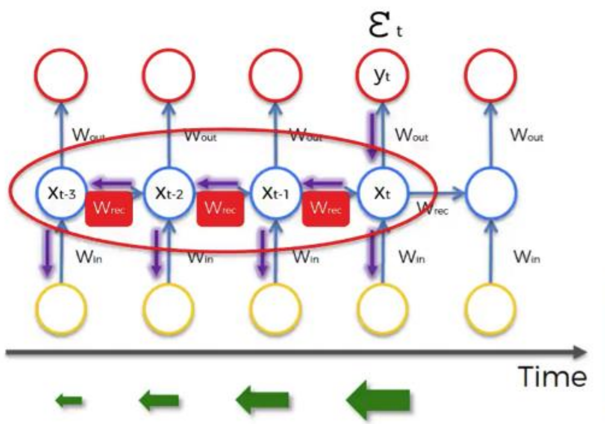
Problems with vanilla RNN

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



I grew up in France…..............................I speak fluent ....





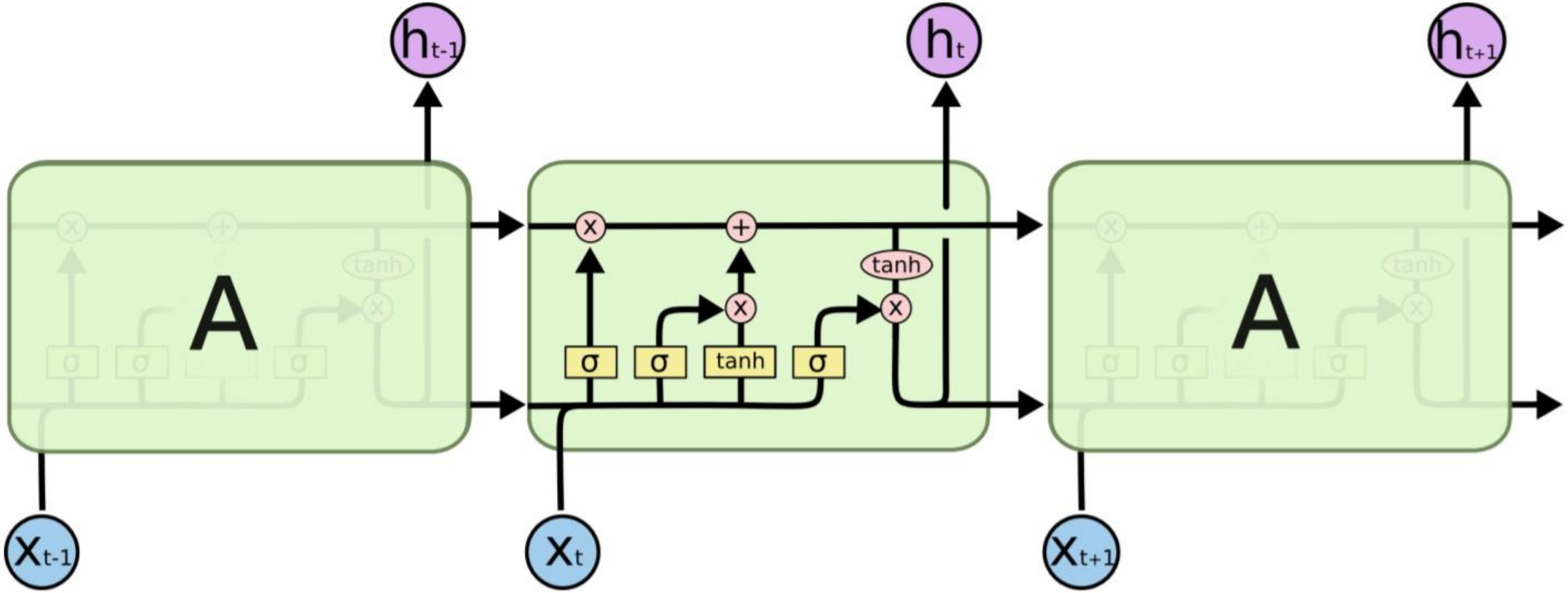
$$\frac{\partial \mathcal{E}}{\partial \theta} = \sum_{1 \le t \le T} \frac{\partial \mathcal{E}_t}{\partial \theta} \quad (3)$$

$$\frac{\partial \mathcal{E}_t}{\partial \theta} = \sum_{1 \le k \le t} \left(\frac{\partial \mathcal{E}_t}{\partial \mathbf{x}_t} \frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_k} \frac{\partial^+ \mathbf{x}_k}{\partial \theta} \right) \quad (4)$$

$$\frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_k} = \prod_{t \ge i > k} \frac{\partial \mathbf{x}_i}{\partial \mathbf{x}_{i-1}} = \prod_{t \ge i > k} \mathbf{W}_{rec}^T diag(\sigma'(\mathbf{x}_{i-1})) \quad (5)$$

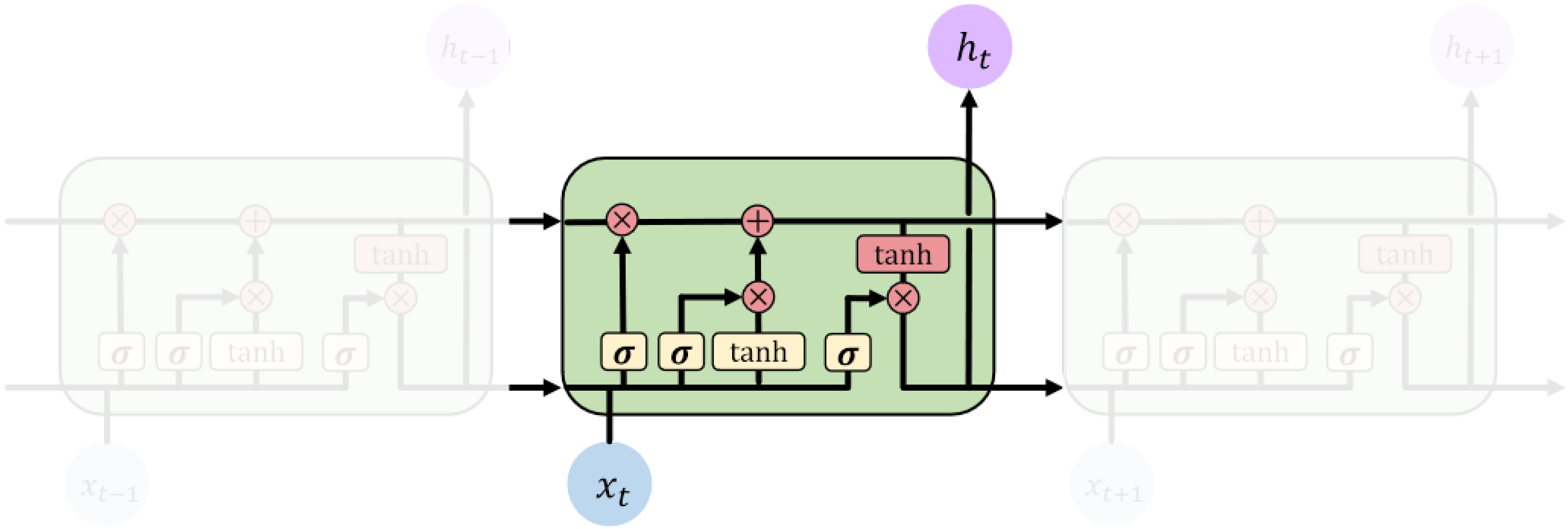
Long Short-Term Memory (LSTM)

LSTM – a more complex network



Long Short-Term Memory (LSTM)

LSTM – no vanishing gradients



Long Short-Term Memory (LSTM)

LSTM – a more complex network

Activations

- a = h
- $a_t = h_t$

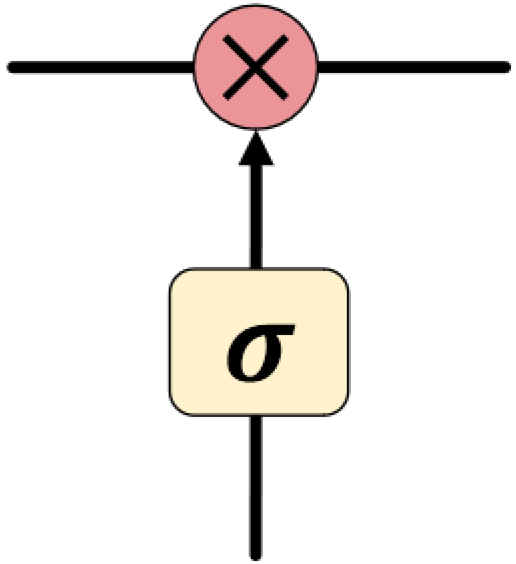
Cell state (memory)

- C
- C~

Long Short-Term Memory (LSTM)

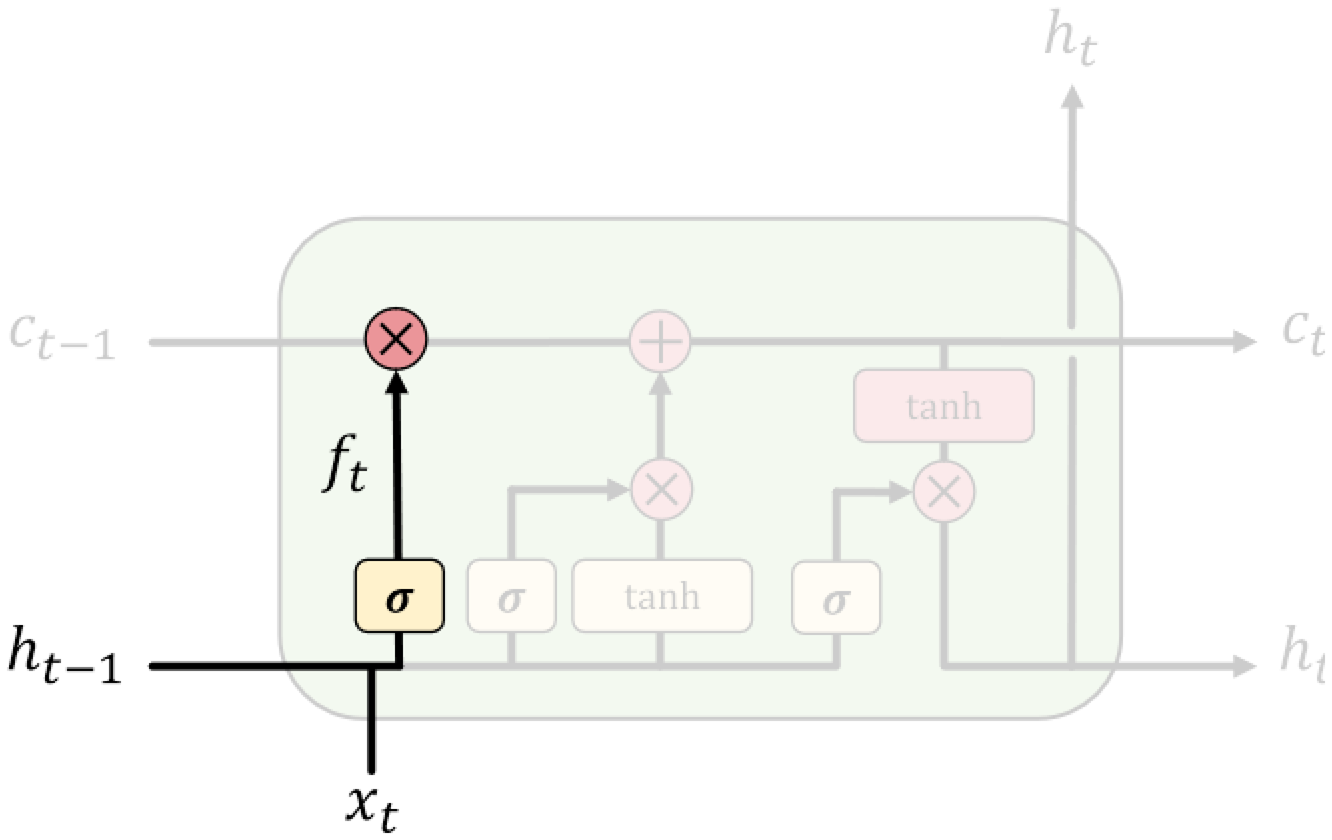
LSTM - Gate

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



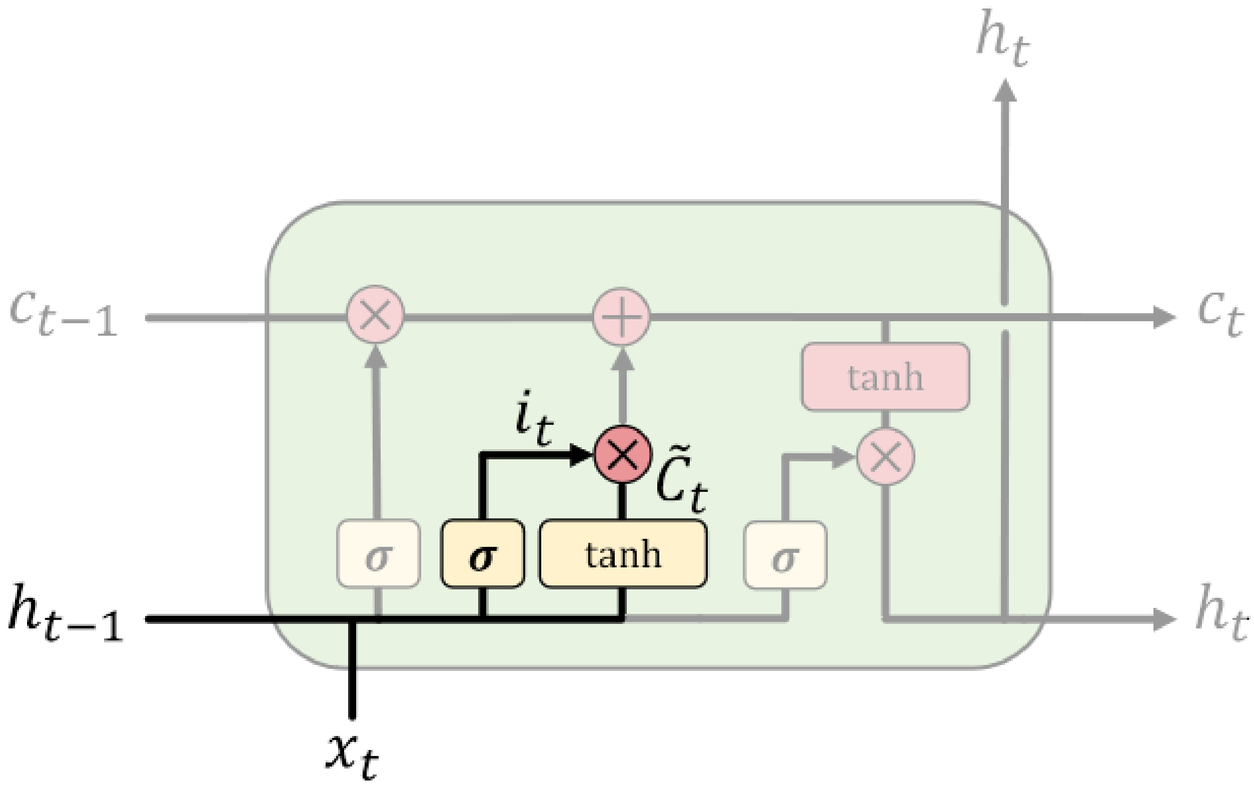
Long Short-Term Memory (LSTM)

LSTM – Forget Gate



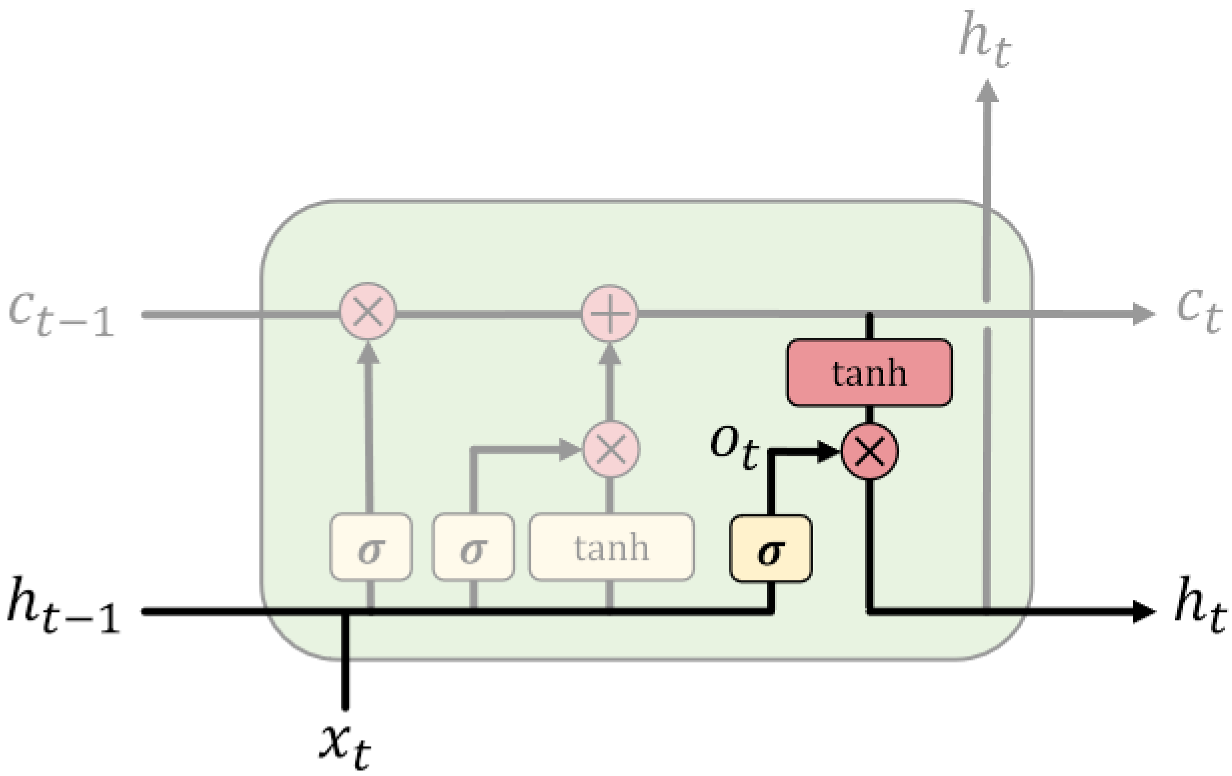
Long Short-Term Memory (LSTM)

LSTM – Input / Ignore Gate



Long Short-Term Memory (LSTM)

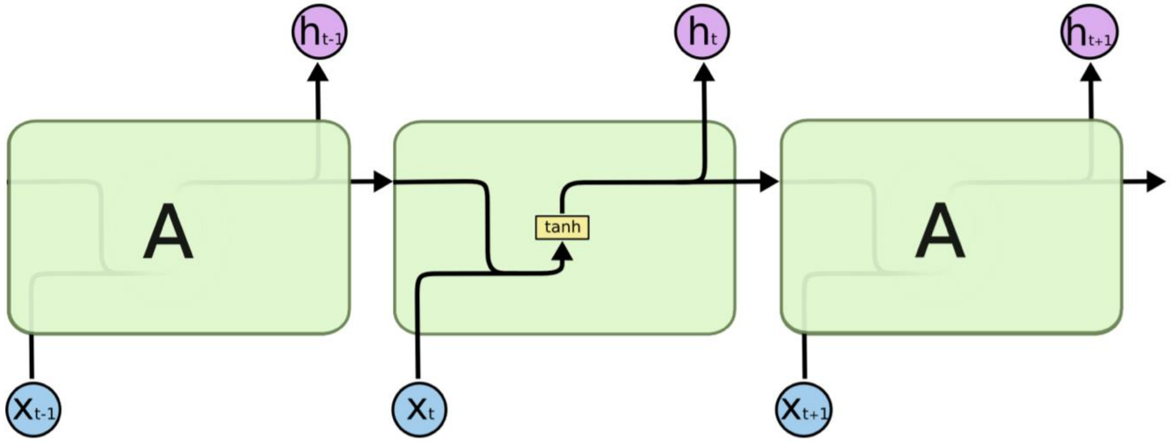
LSTM – Output Gate



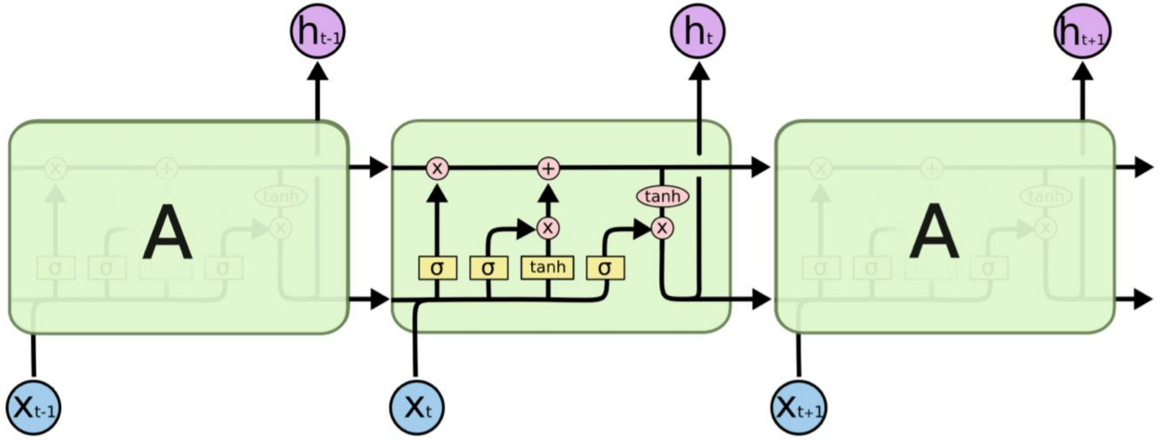
Long Short-Term Memory (LSTM)

LSTM – a more complex network

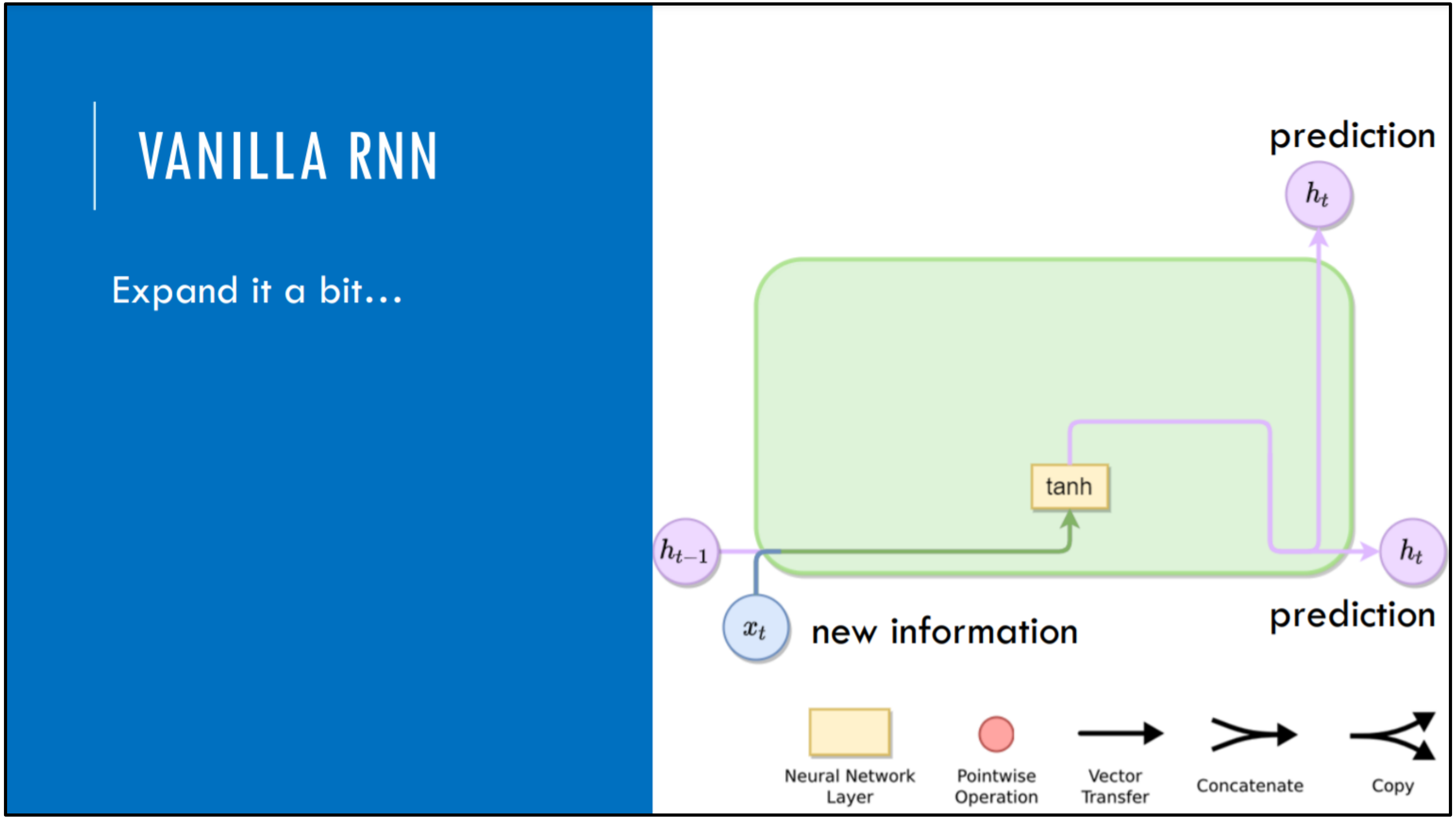
Vanilla RNN:



LSTM:



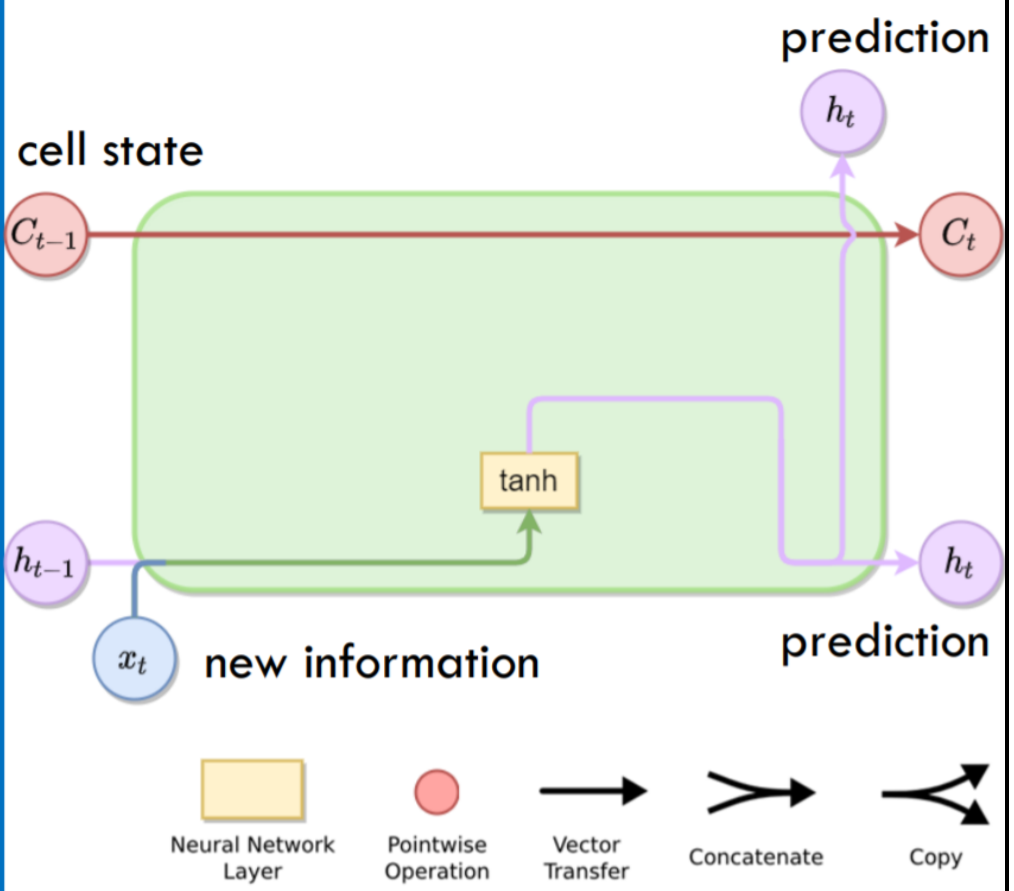
Long Short-Term Memory (LSTM)



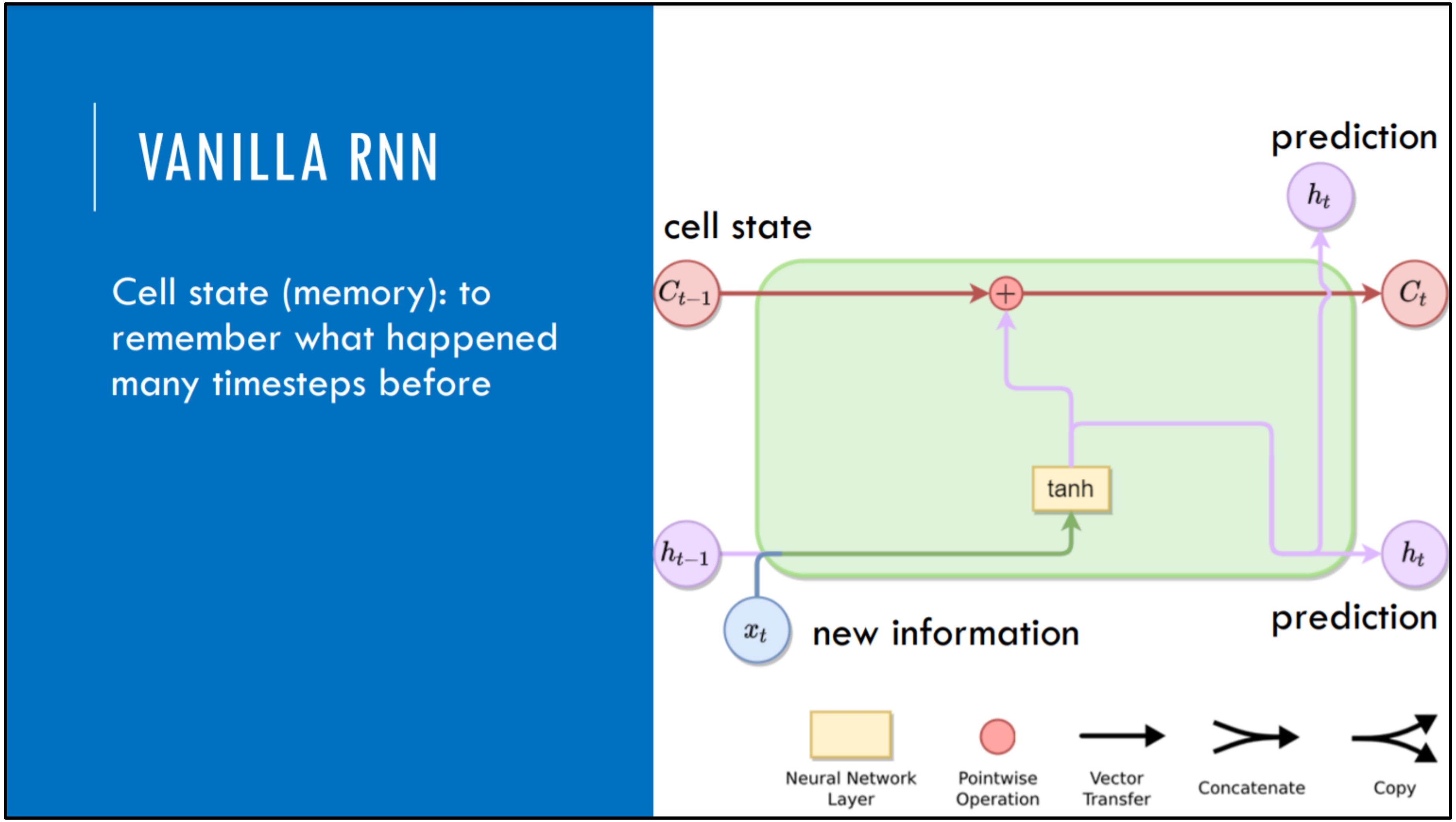
Long Short-Term Memory (LSTM)

VANILLA RNN

Cell state (memory): to remember what happened many timesteps before



Long Short-Term Memory (LSTM)



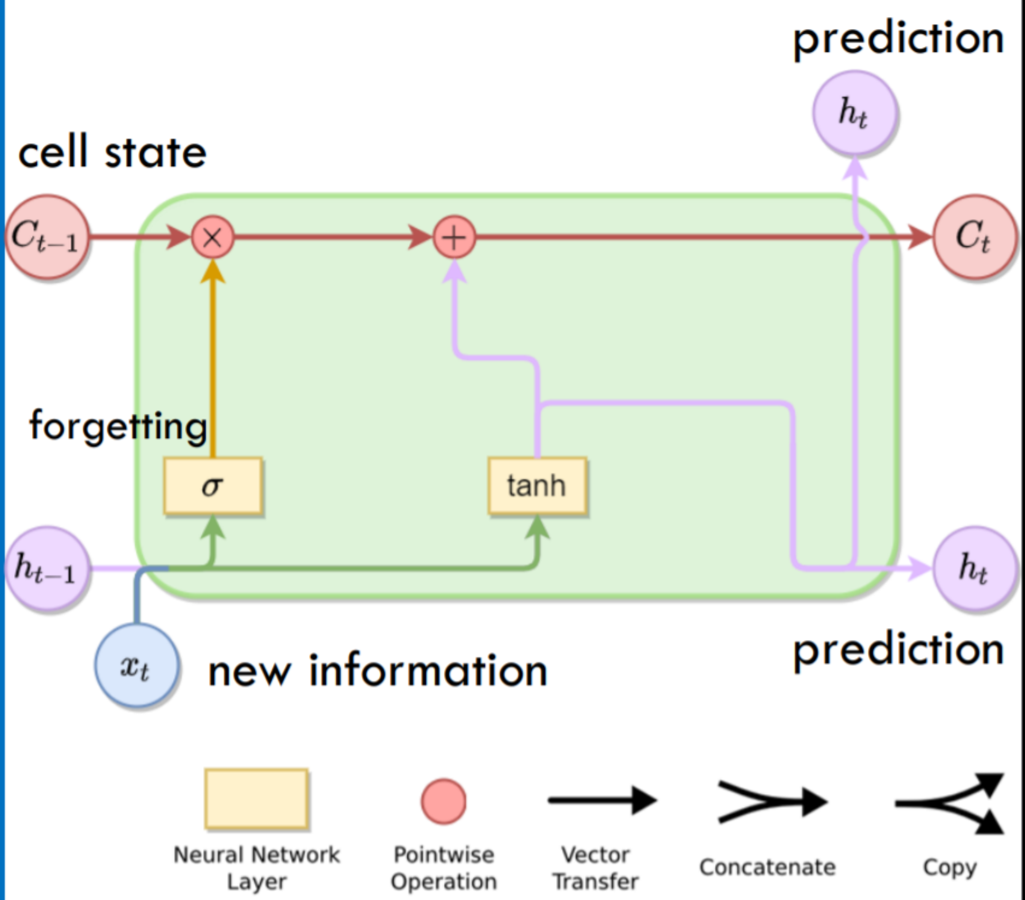
Long Short-Term Memory (LSTM)

FORGET GATE

Memory: want to remember what happened many timesteps before

Forgetting: based on what we are seeing now it decided 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

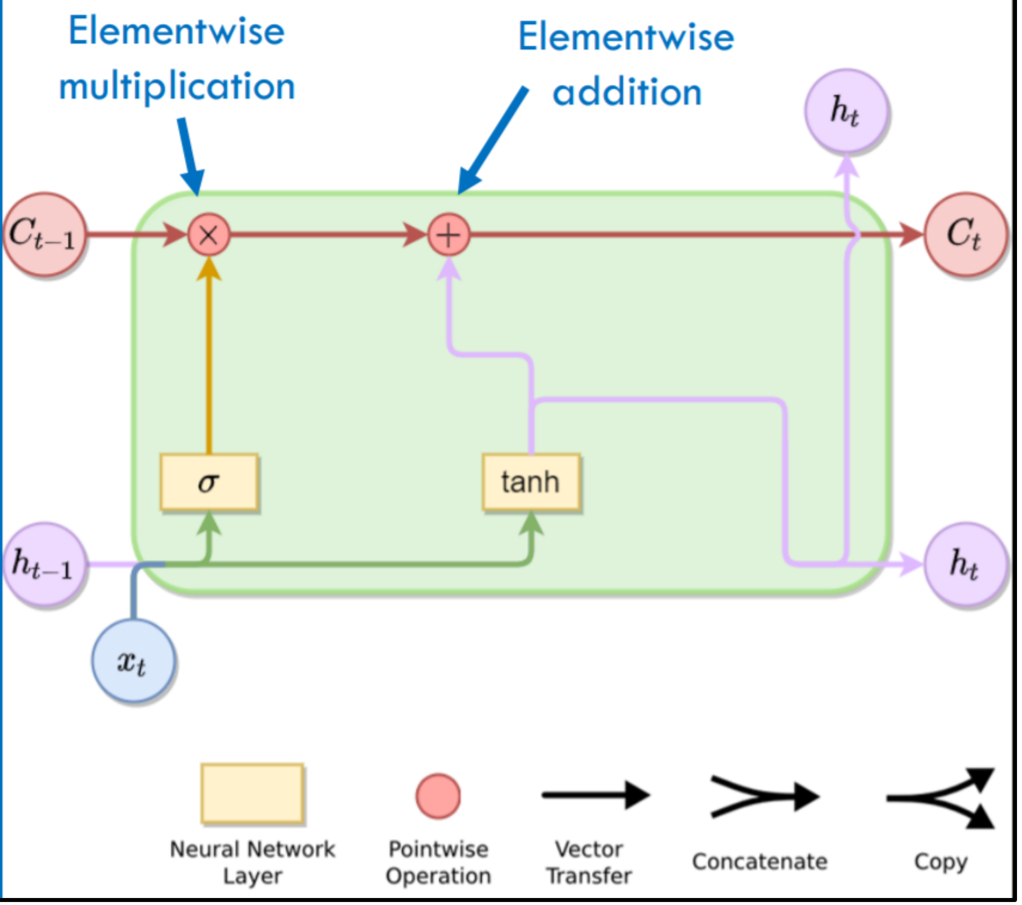


Long Short-Term Memory (LSTM)

GATING

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

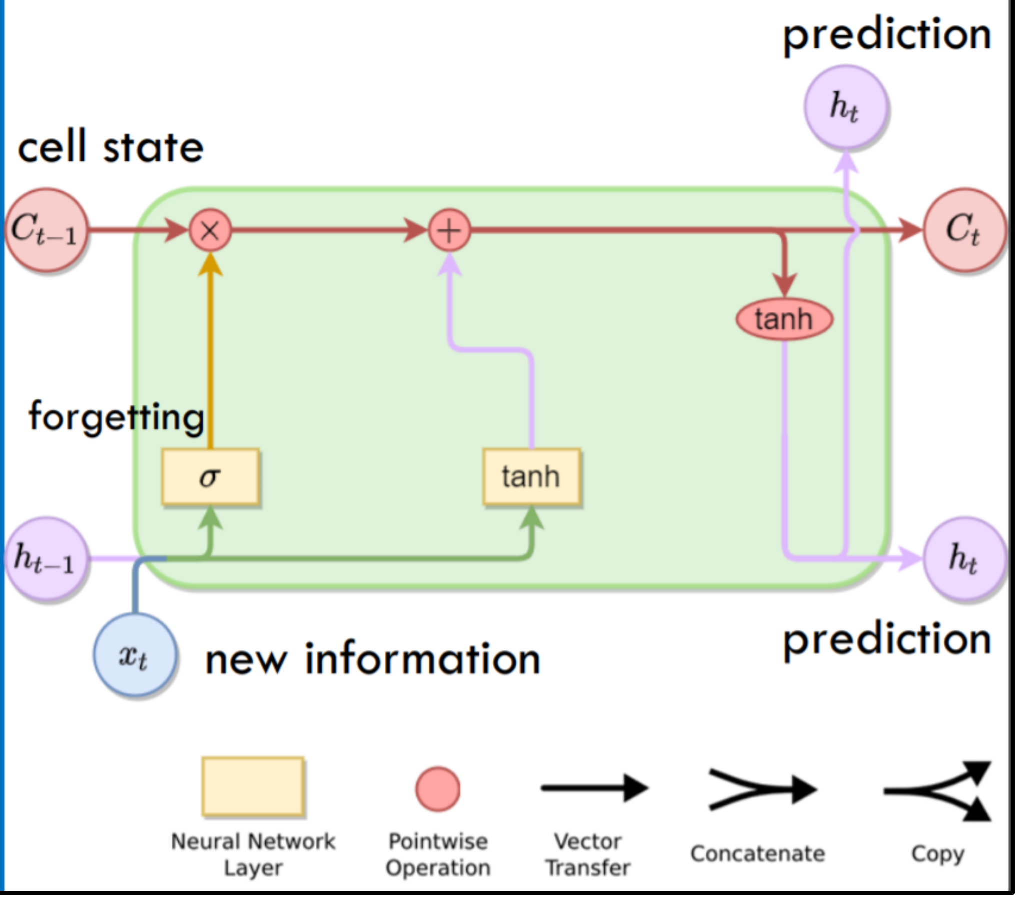
Let's us control what is passing through and what is blocked



Long Short-Term Memory (LSTM)

USE CELL STATE

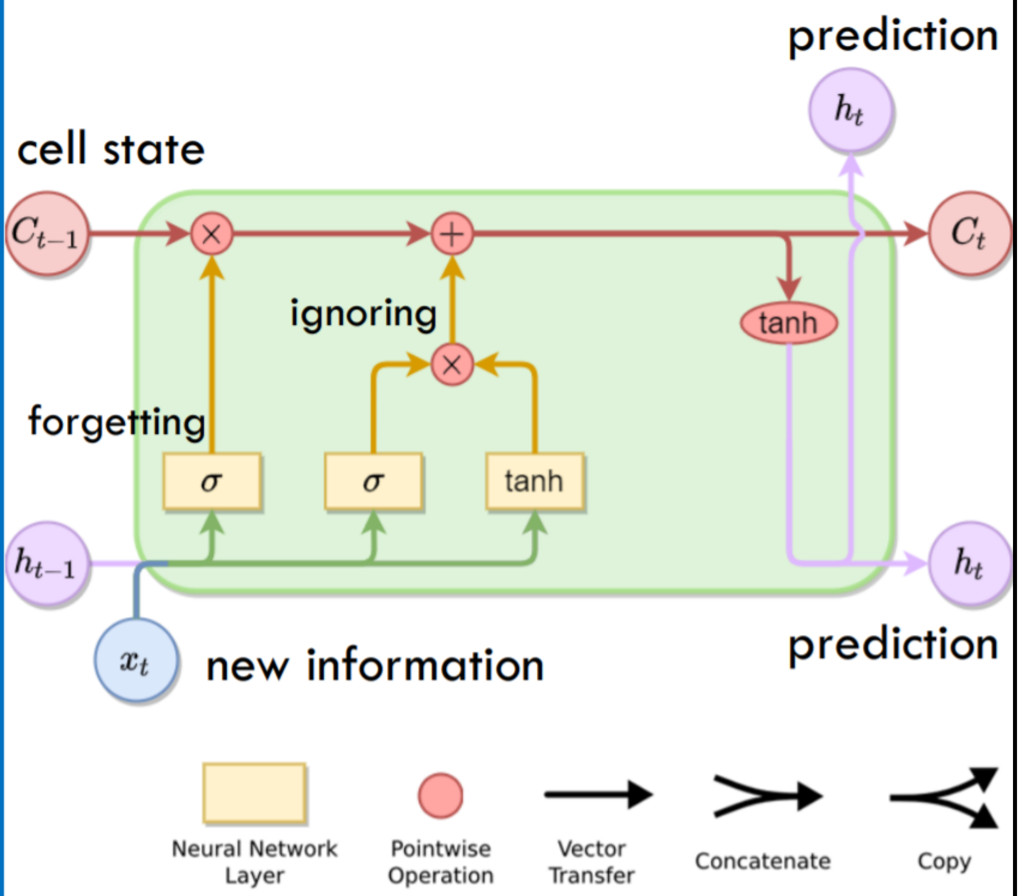
tanh: after addition we need to rescale it



Long Short-Term Memory (LSTM)

IGNORE GATE

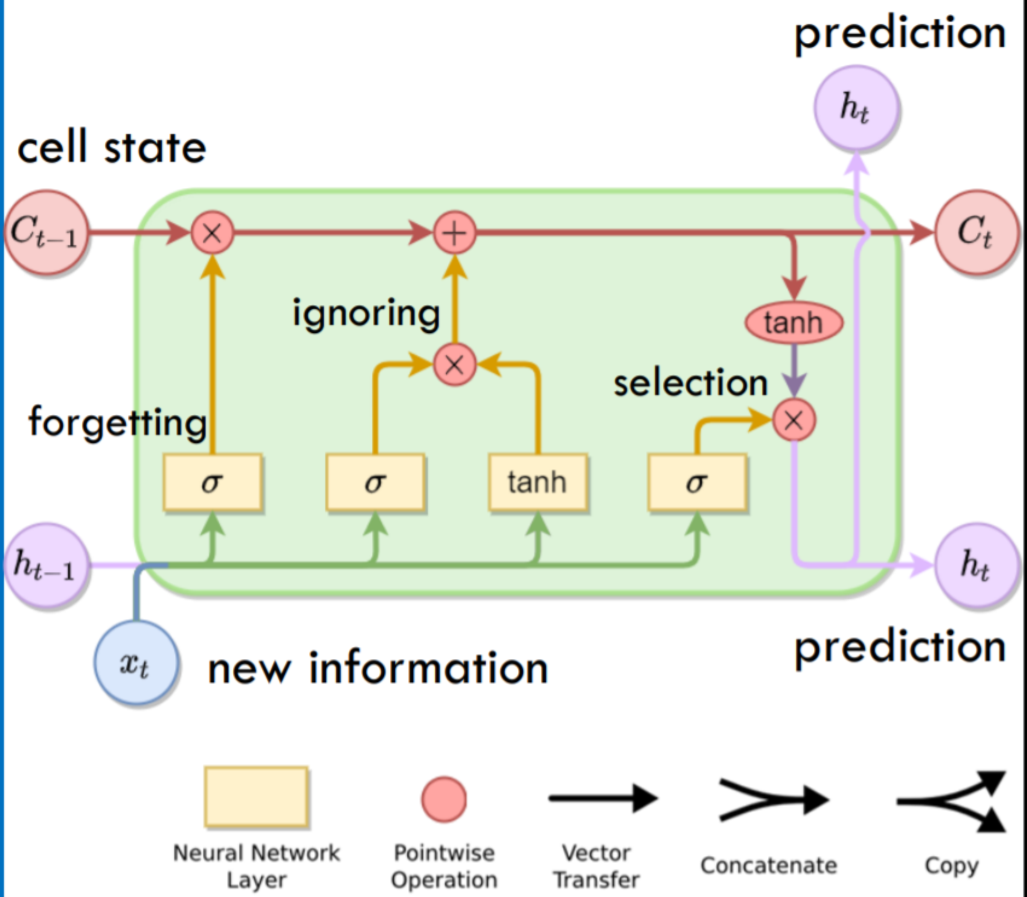
Ignoring: ignore things that aren't immediately relevant, so they don't cloud the predictions in memory going forward



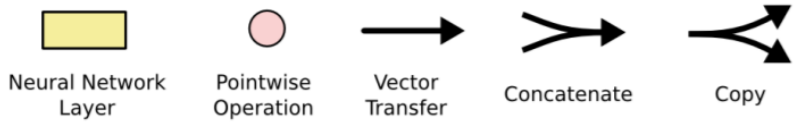
Long Short-Term Memory (LSTM)

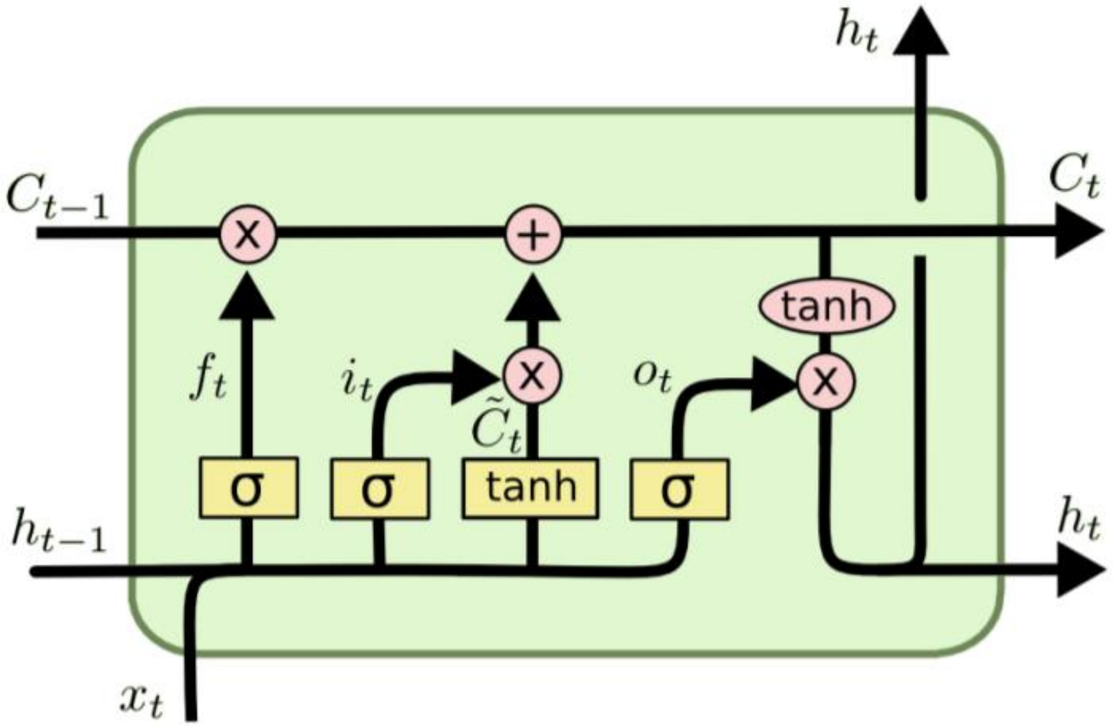
OUTPUT GATE

Selection: 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 predictions out



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:

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

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

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

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

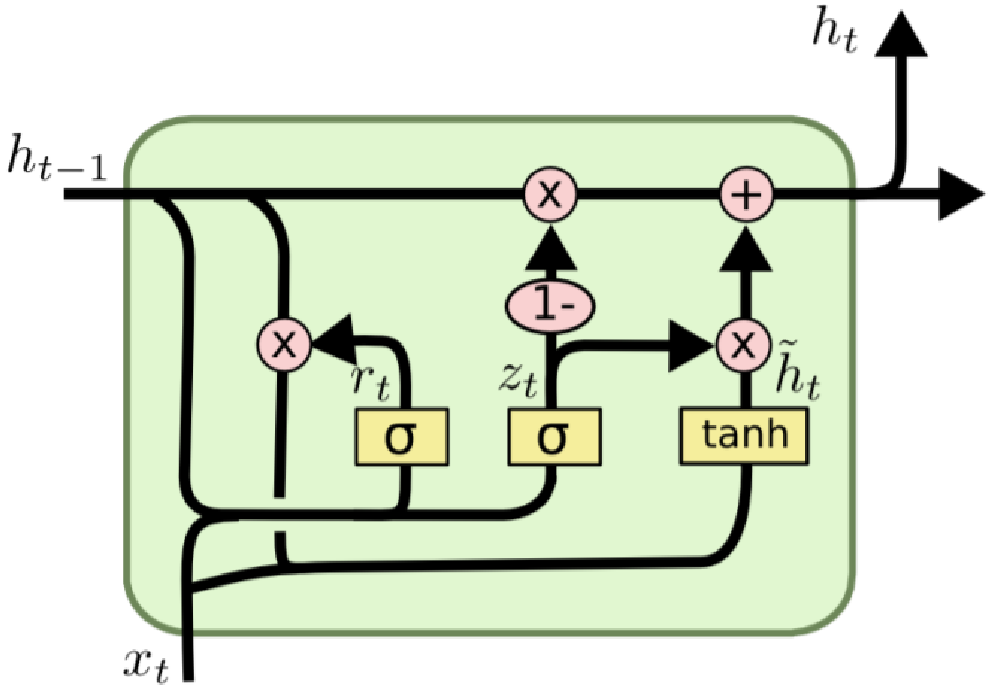
Outputs:

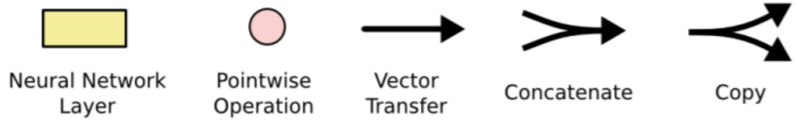
$$C_t = f_t \circ C_{t-1} + i_t \circ \tilde{C}_t$$

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

where ° 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.

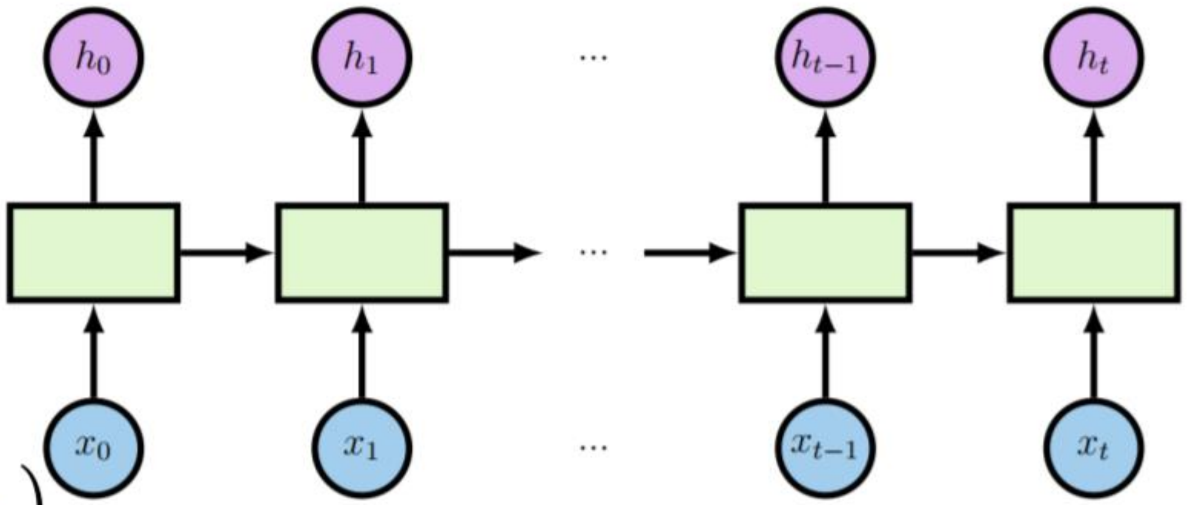
Recurrent Layer

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:

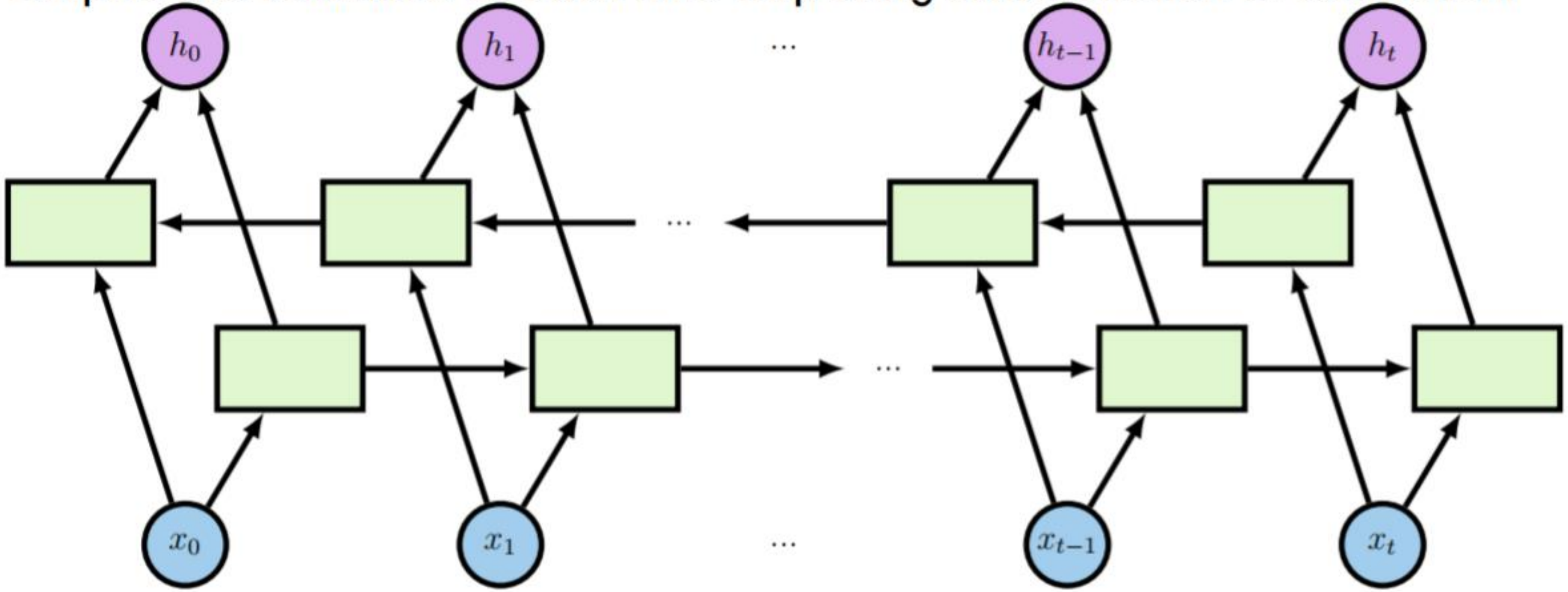
$$h_{d,t}^{(n)} = g_d\left(W_d^h\, h_{d,t-1}^{(n)} + W_d^x\, h_{d-1,t}^{(n)} + b_d\right)$$

- 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

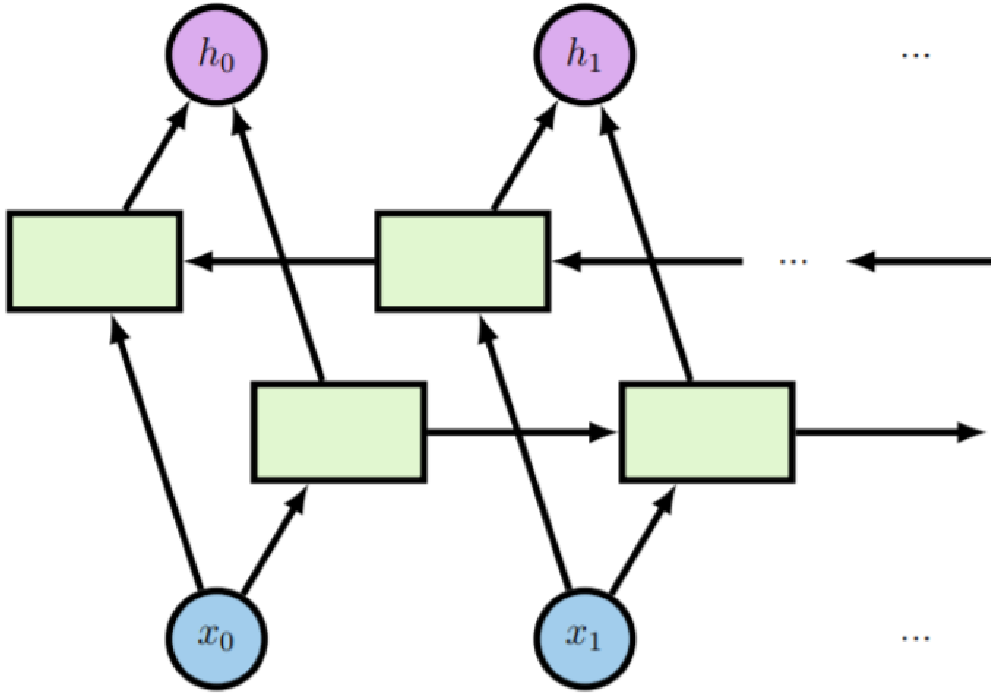
BIDIRECTIONAL RNN (BRNN)

A BRNN layer:

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

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

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

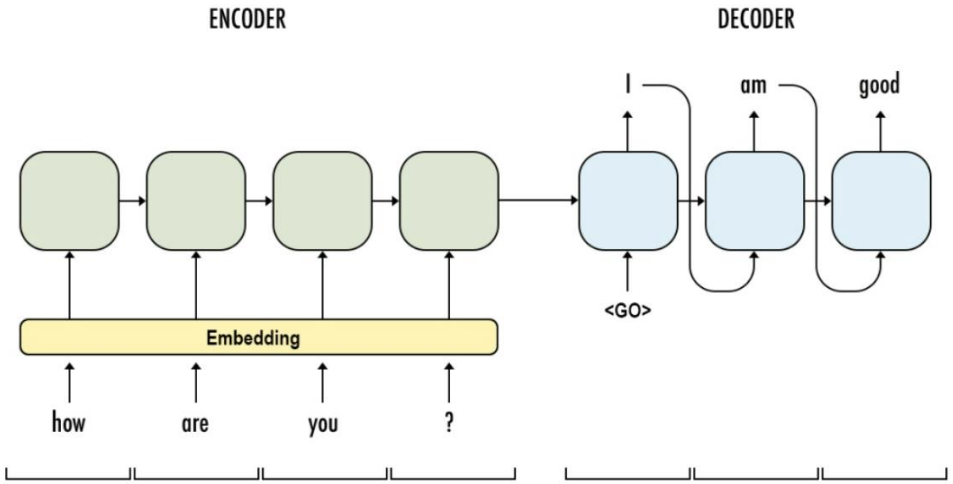


Where

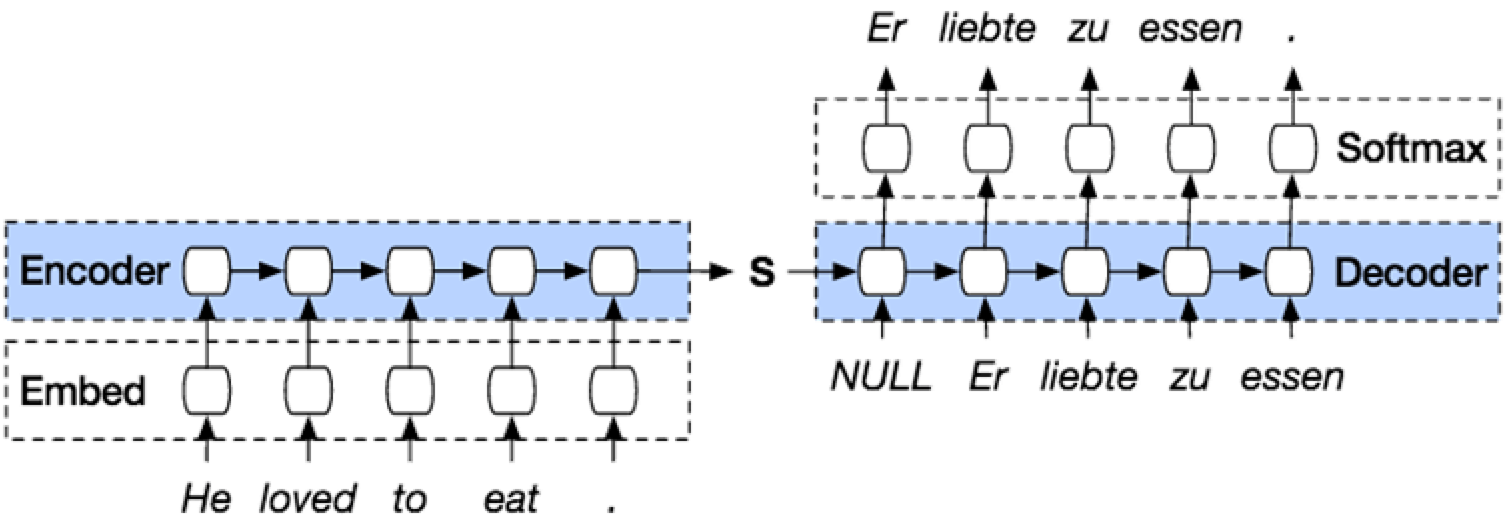
- "$\longrightarrow$" means normal time order, while "$\longleftarrow$" is associated with reverse time order
- $t = 1, \ldots, T_n$;
- $\boldsymbol{\theta}_d = \left\{\overrightarrow{\boldsymbol{W}^h_d}, \overrightarrow{\boldsymbol{W}^x_d}, \overrightarrow{\boldsymbol{b}_d}, \overleftarrow{\boldsymbol{W}^h_d},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

Encoder – Decoder (Seq2Seq)

Question : Answer
(sequence) : (sequence)

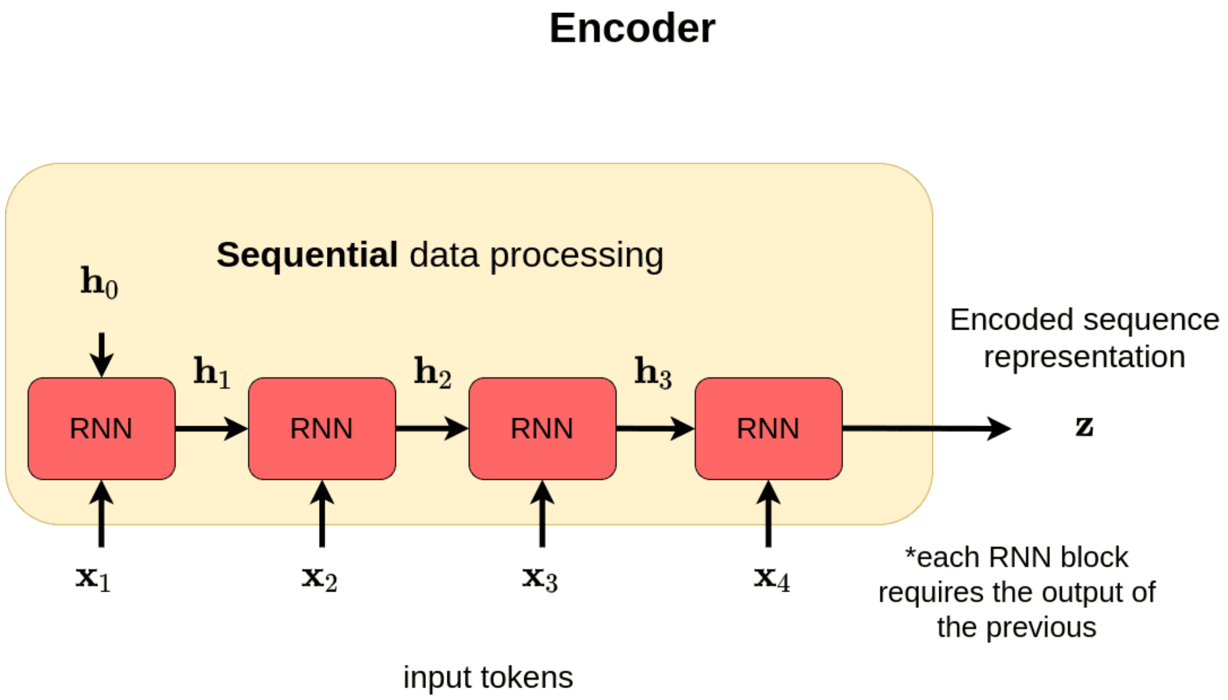


Machine Translation
English : German
(sequence) : (sequence)



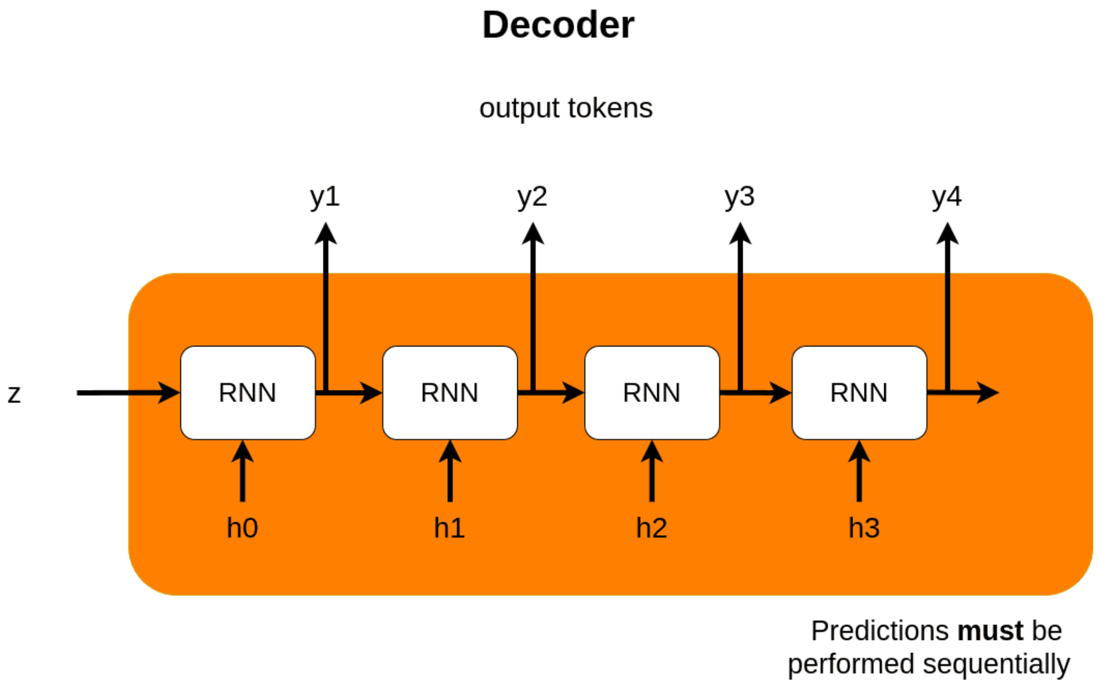
Encoder – Decoder (Seq2Seq)

- 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)

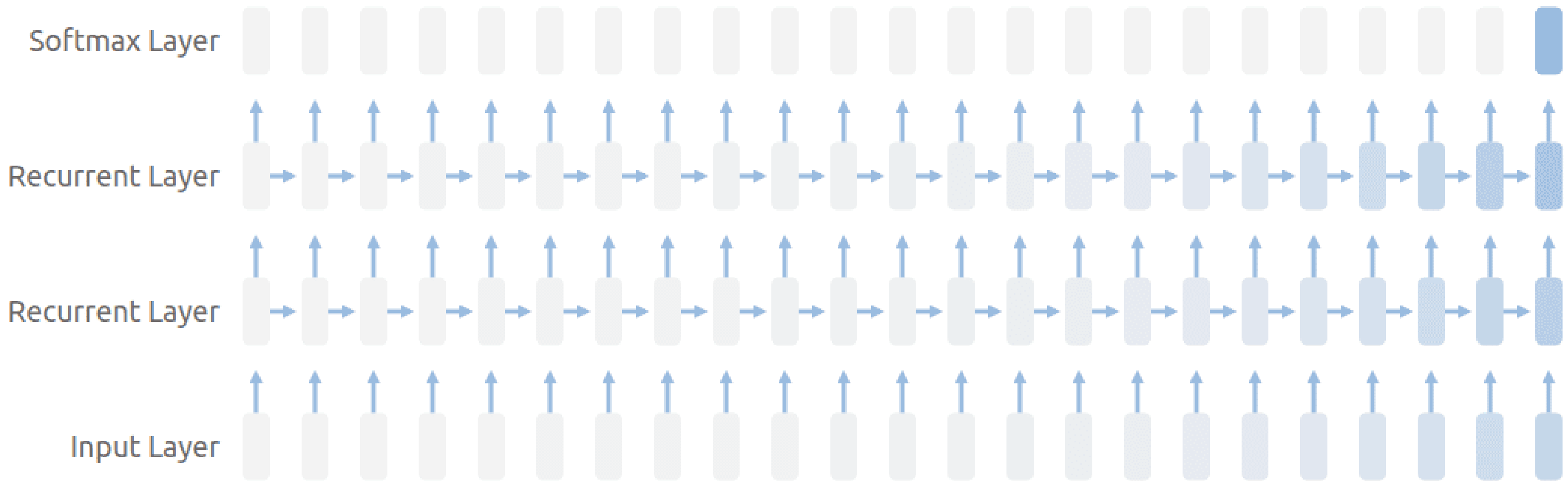
- On the other hand, the decoder receives the context vector z and generates the output sequence. The most common application of Seq2seq is language translation. We can think of the input sequence as the representation of a sentence in English and the output as the same sentence in French.



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, 28th March 2025

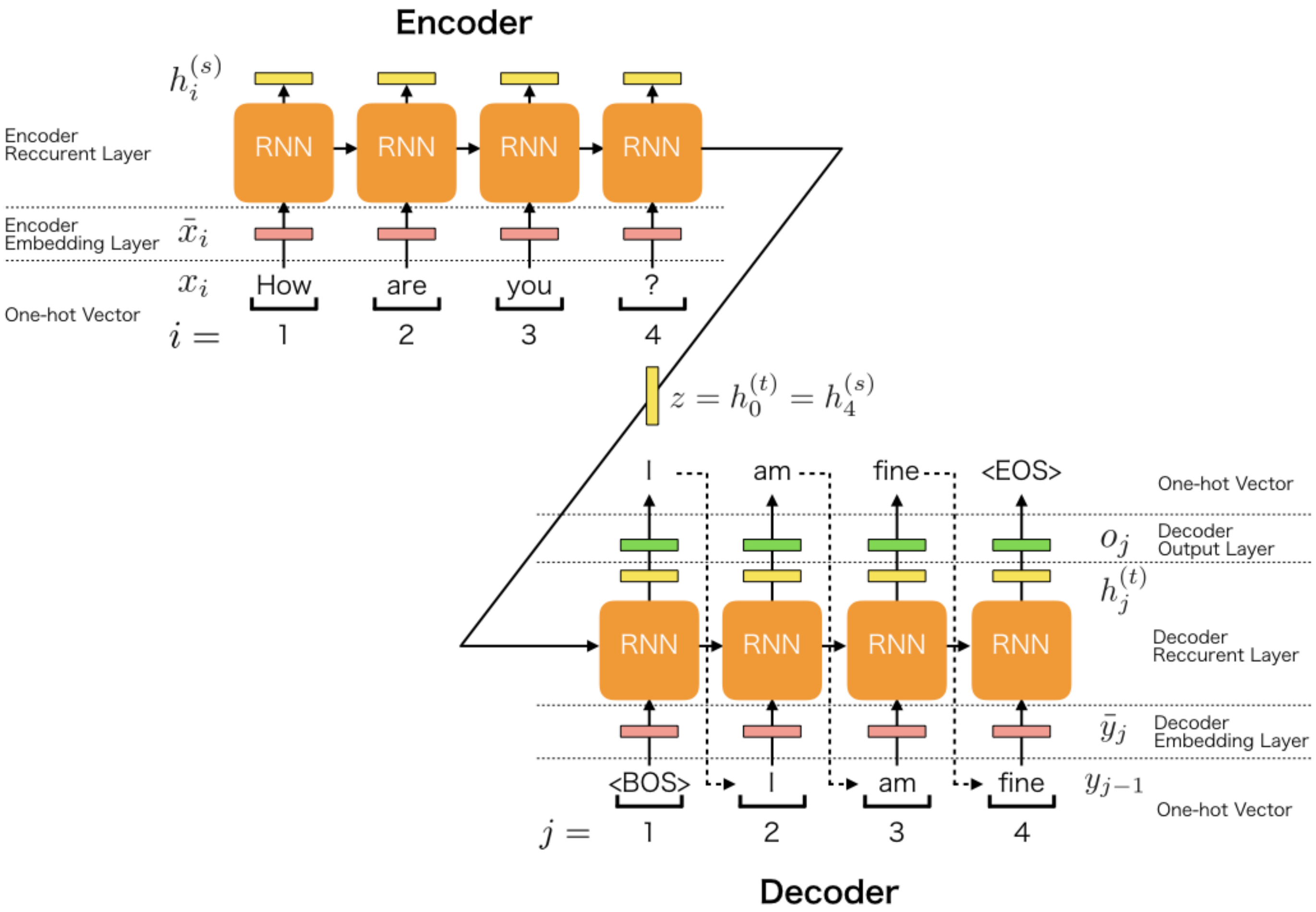
1 RNNs and Embeddings

2 LSTM, GRU & Seq2Seq

3 Attention Mechanism

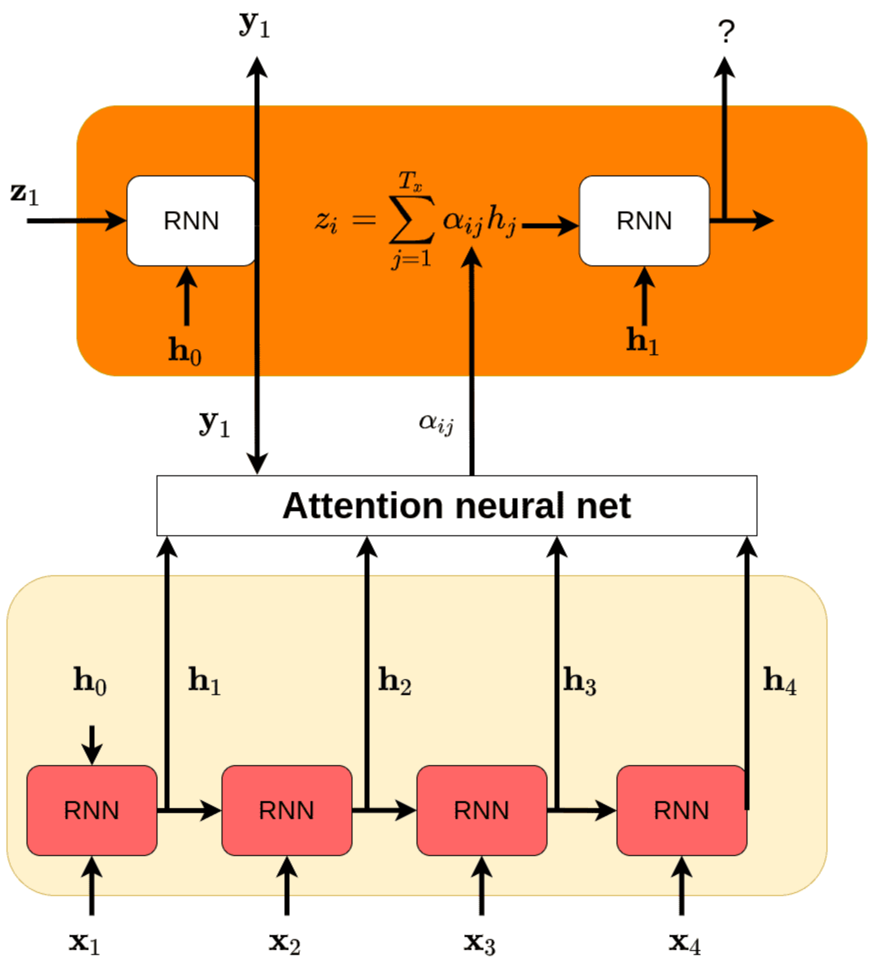
Introduction to Attention Mechanism

- 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?



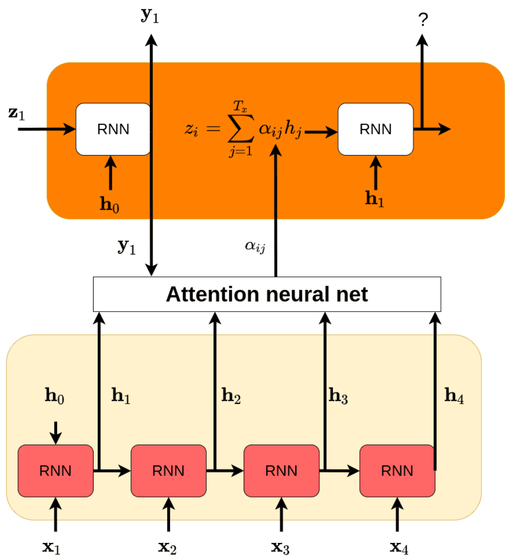
Introduction to Attention Mechanism

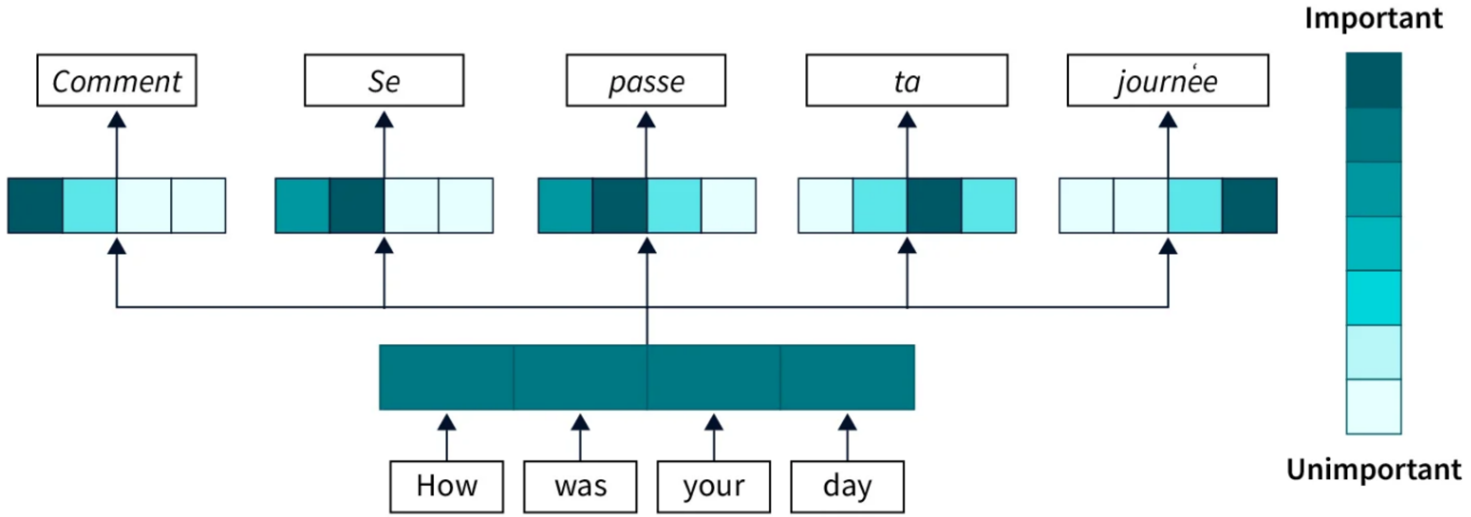
- Attention mechanism helps creating a better context vector
- It learns which information from the encoder is relevant for the decoder



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 core idea is that the context vector **z** 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 $h = h_1, h_2, \ldots, h_n$
    - Given the previous state in the decoder y_{i-1}
    - We define an attention network that gives the attention scores for the current state of the decoder

$$\mathbf{e}_i = \text{attention}_{\text{net}}\left(y_{i-1}, \mathbf{h}\right) \in R^n$$

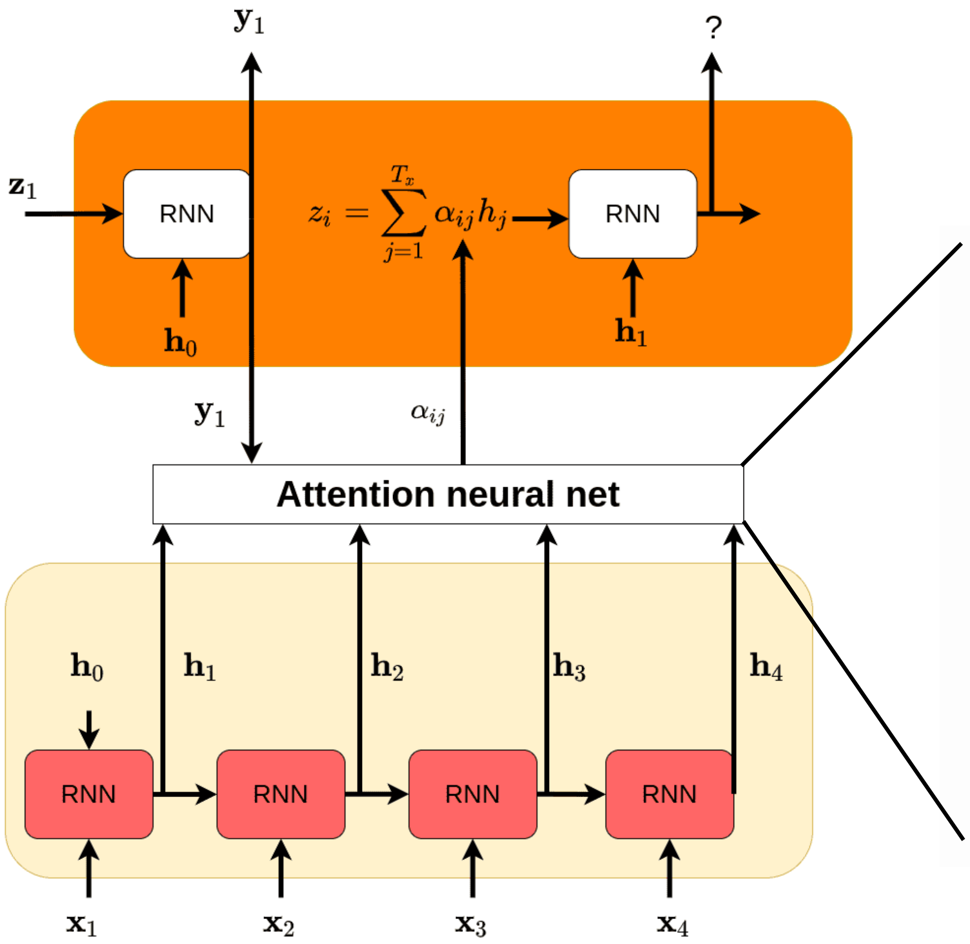
    - We convert this scores into probabilities

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

    - Finally, we get our new context vector z:

$$z_i = \sum_{j=1}^{T} \alpha_{ij} \mathbf{h}_j$$

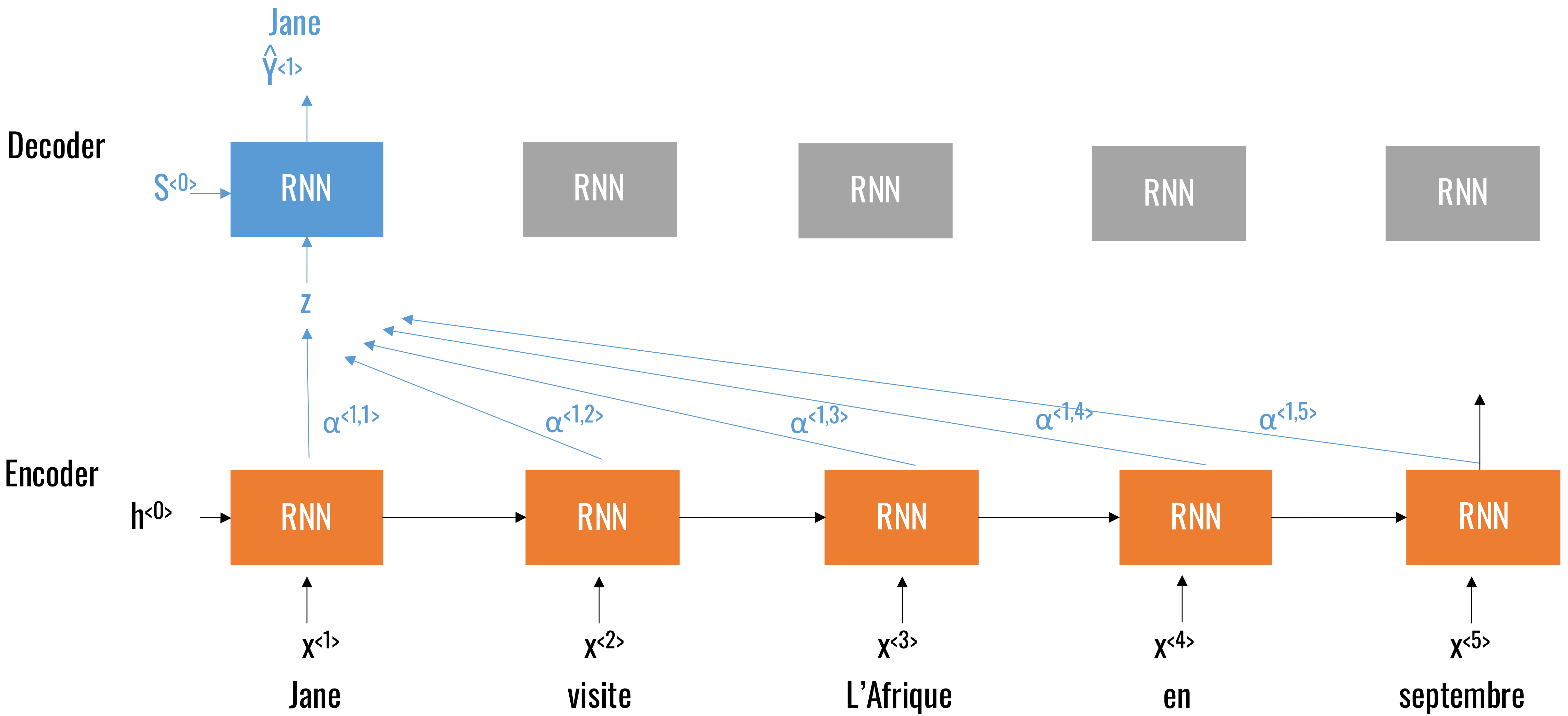
Attention Mechanism



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



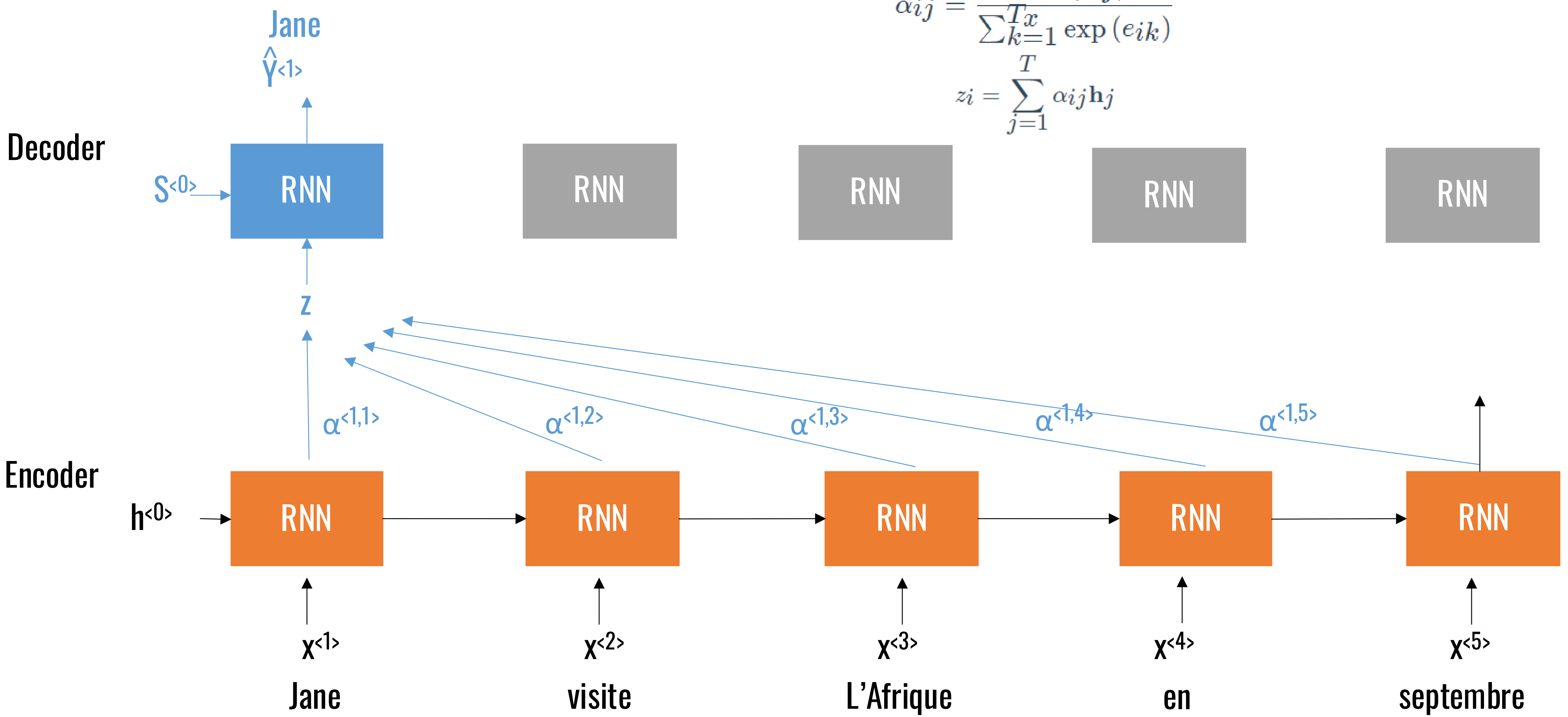
Attention Mechanism

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

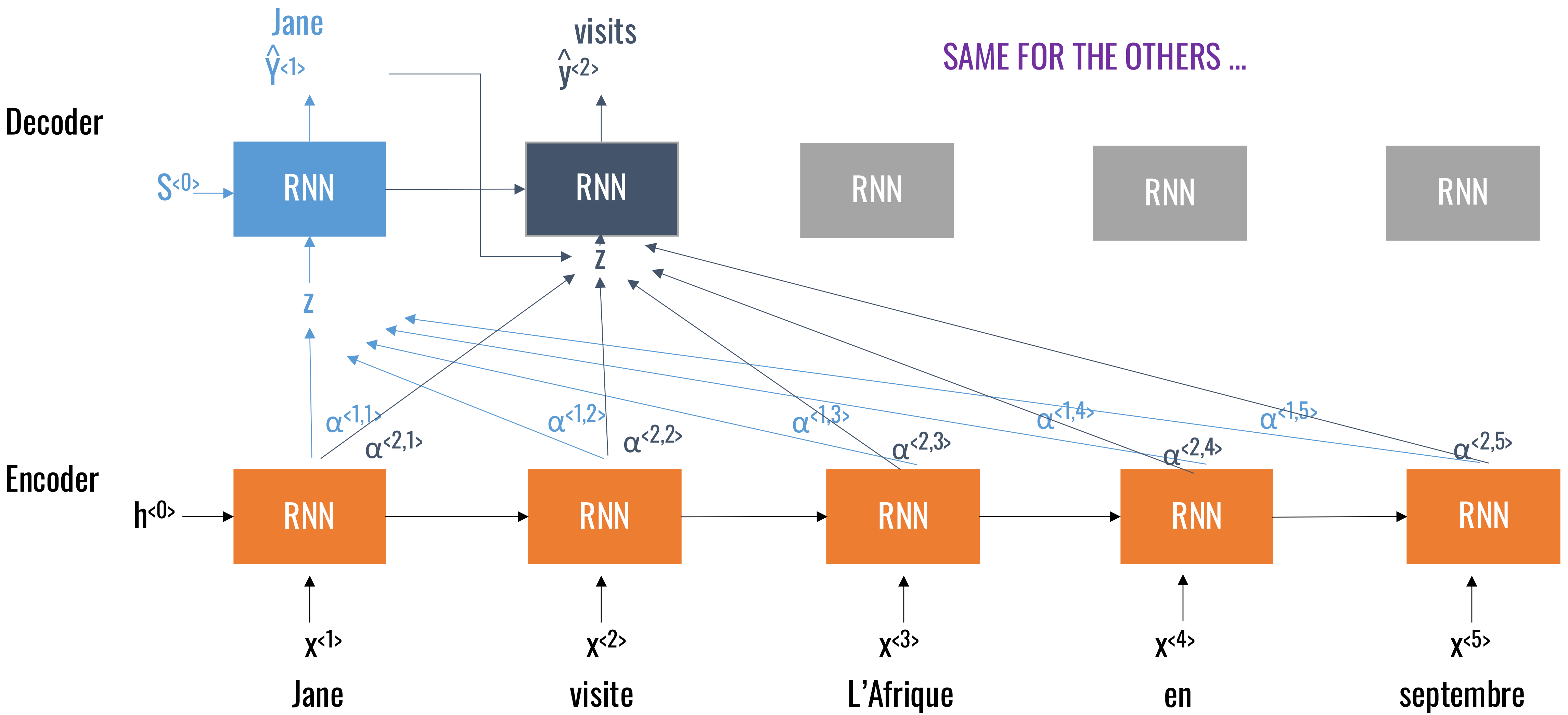
$$\mathbf{e}_i = \text{attention}_{\text{net}}\left(y_{i-1}, \mathbf{h}\right) \in R^n$$

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

$$z_i = \sum_{j=1}^{T} \alpha_{ij} \mathbf{h}_j$$



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!