Sequence modeling

Prototyping with Deep Learning

Learning outcomes

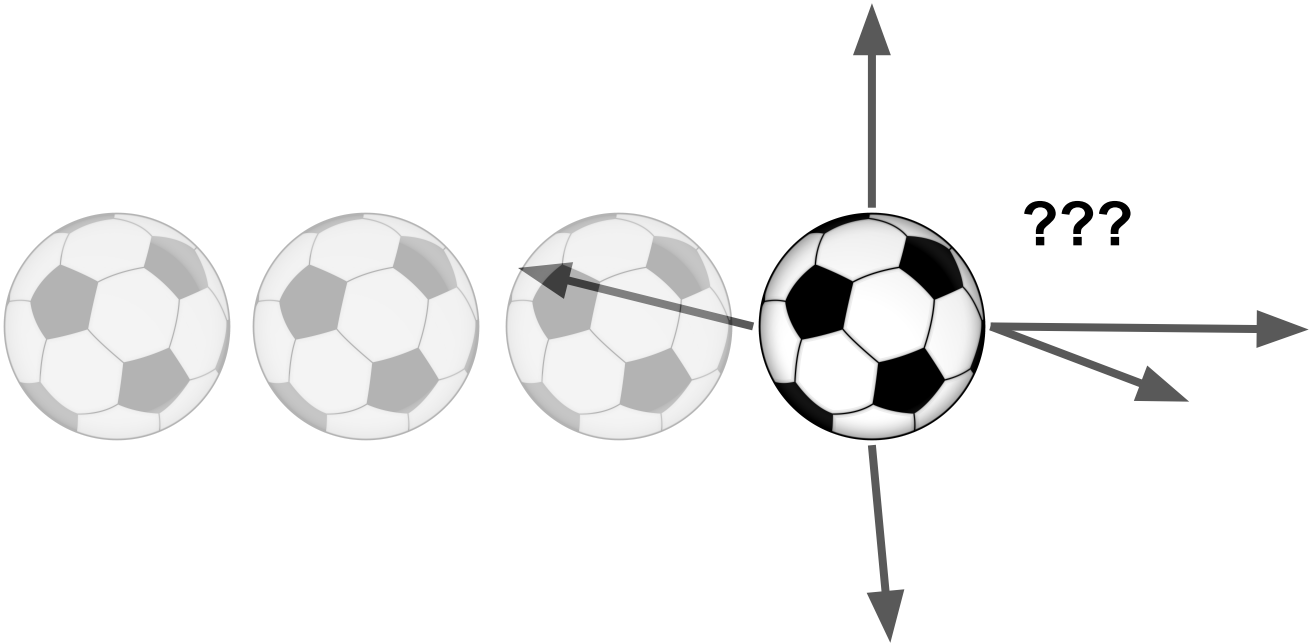
UNIVERSITÉ DU LUXEMBOURG

After this lesson you will be able to:

- Understand fundamentals of Sequence modelling
- Understand the limitations of RNNs for Sequence modeling tasks.
- Understand the concept of Self-Attention for Sequence modeling.
- Know a powerful DL architecture based on self-Attention called Transformers.
- Know popular transformer models and their applications.

Sequence Modeling

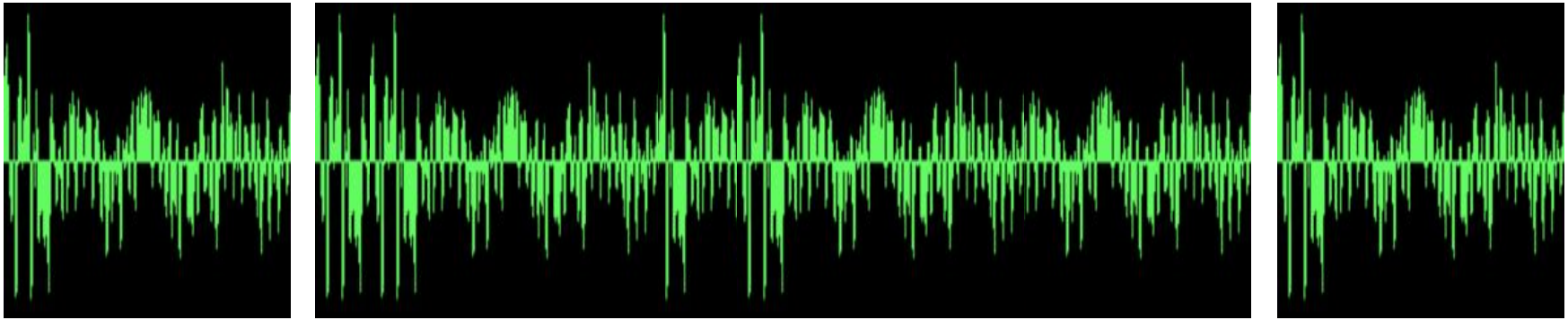
Given the current position of the ball can you predict where it will go next?



Sequence Modeling

UNIVERSITÉ DU LUXEMBOURG

Sequential data



Audio

Sequence Modeling

Sequential data

Character: P r o t o t y p i n g

Prototyping with Deep Learning

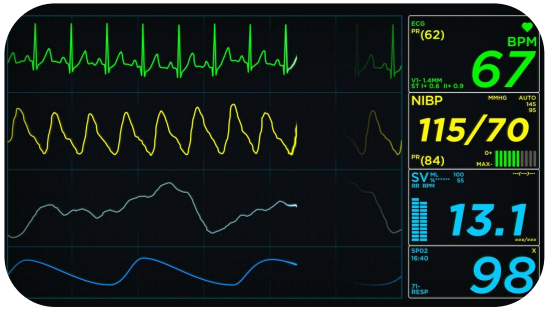
Word: with Deep Learning

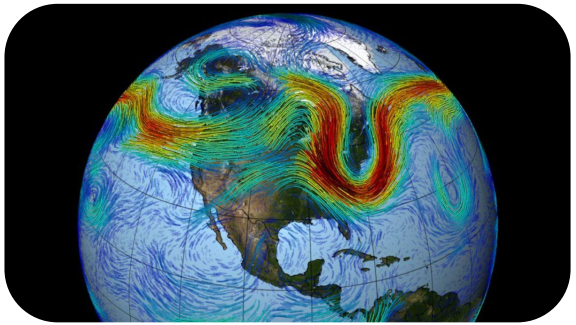
Text

Sequence Modeling

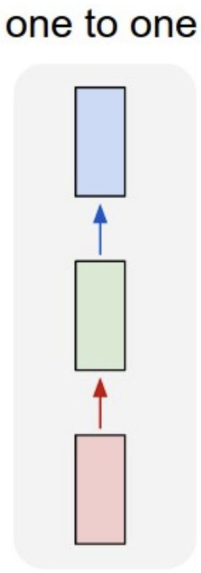
UNIVERSITÉ DU LUXEMBOURG

Sequential data





Sequence Modeling Application



Binary classification

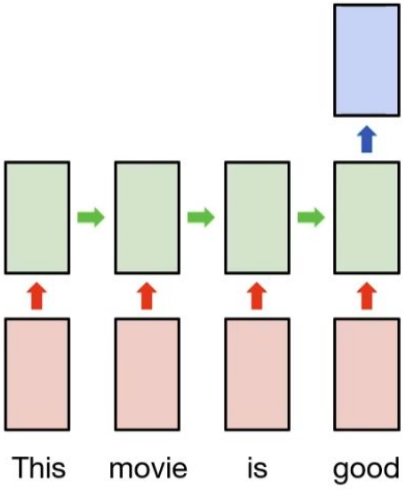
Cat or Dog

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Sequence Modeling

Example: Sentiment Classification Task

Classification : Positive or negative?



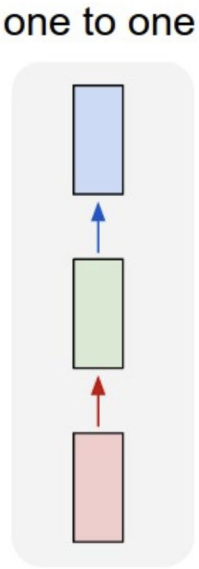
RNN many to one

Input: Sequence of words
Output: Probability of having positive sentiment

```
loss = tf.nn.softmax_cross_entropy_wiht_logits(y, predited)
```

Predicting sentiment associated with a tweet

Sequence Modeling Application



Binary classification
Cat or Dog

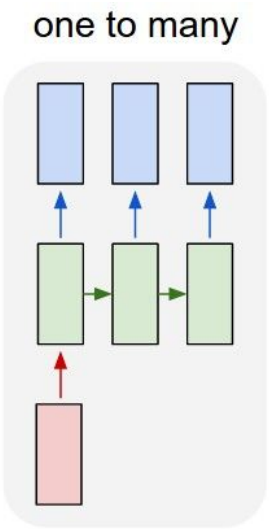
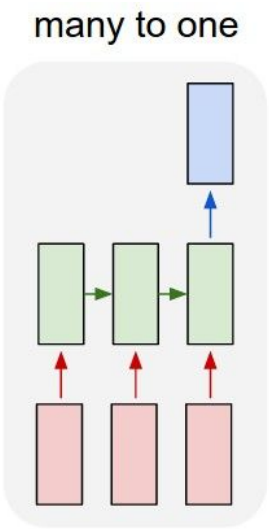
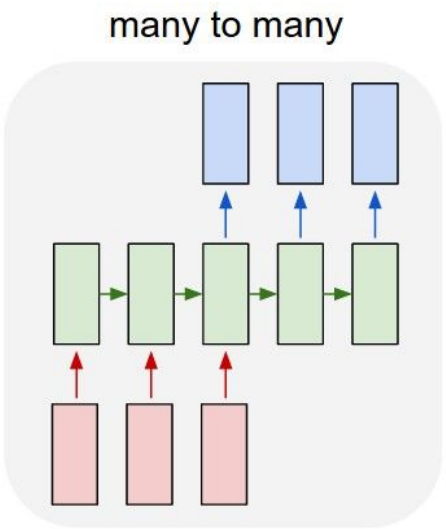


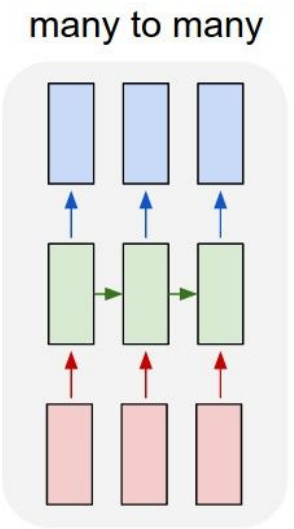
Image captioning



Sentiment classification



Machine translation

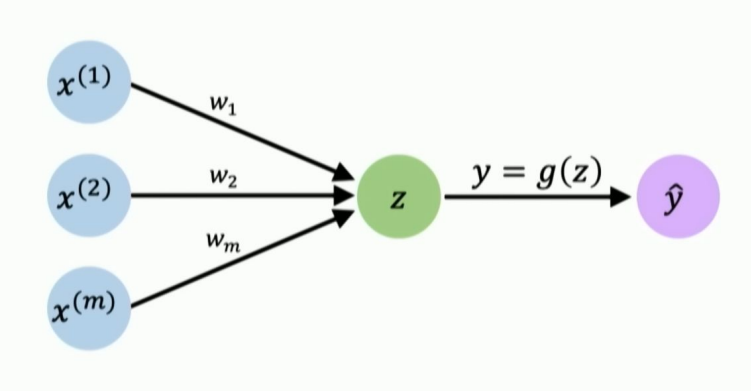


Video classification (label each frame)

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

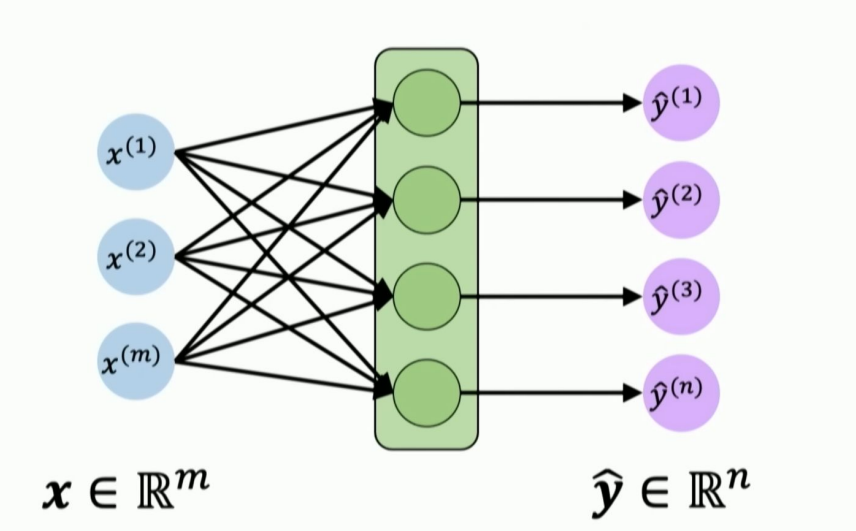
Sequence Modeling

Feed-Forward: No notion of time or temporal dependency



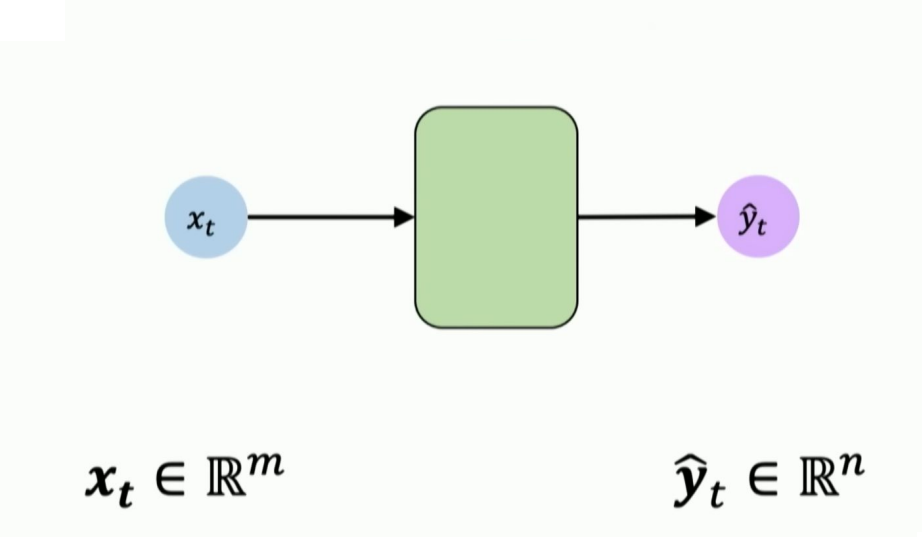
Sequence Modeling

Feed-Forward: No notion of time or temporal dependency



Sequence Modeling

Feed-Forward: No notion of time or temporal dependency

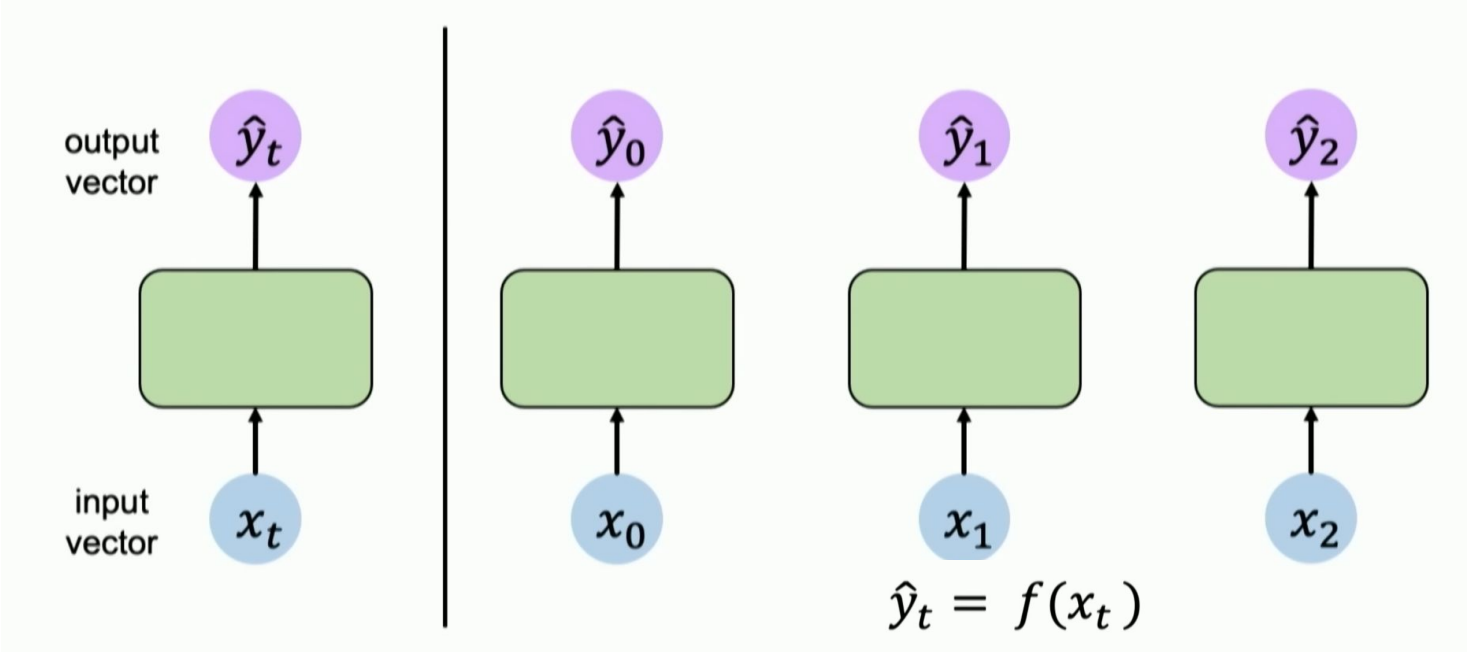


How can we extend this to model multiple time steps of a sequential data?

Sequence Modeling

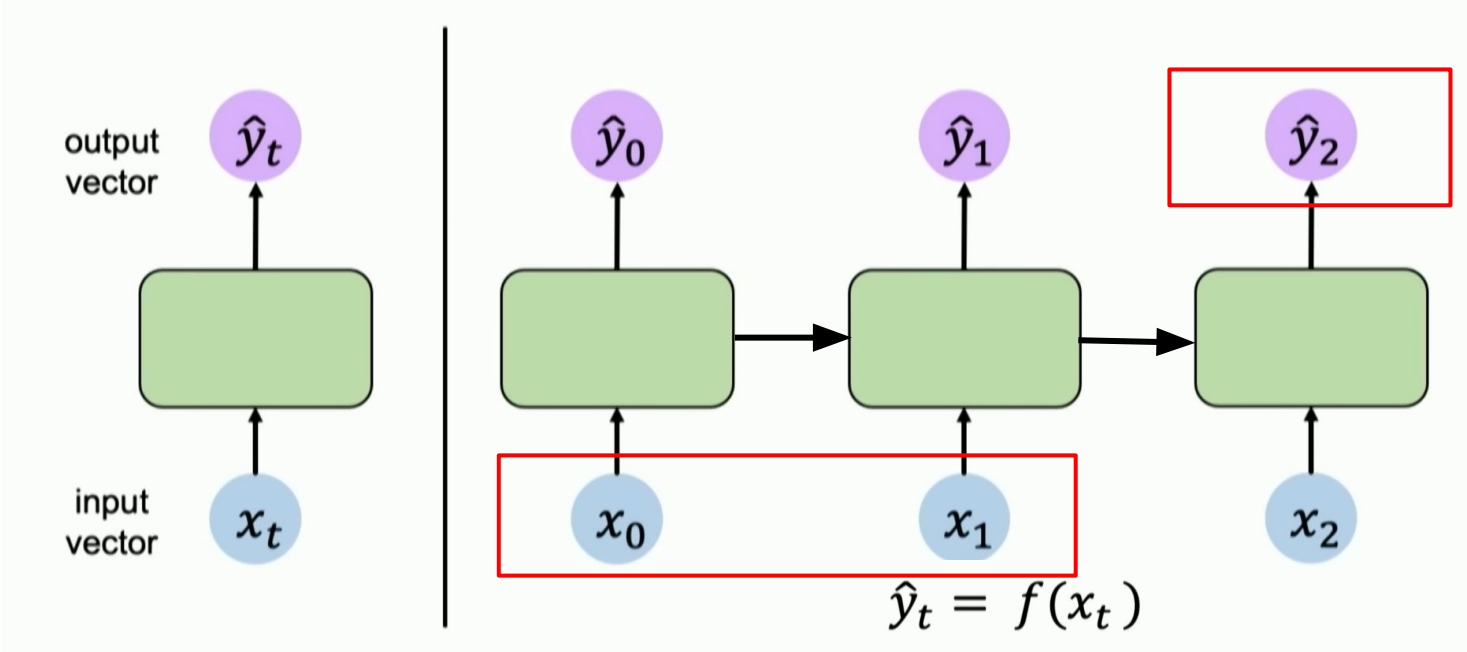
UNIVERSITÉ DU LUXEMBOURG

Handling individual time steps



Sequence Modeling

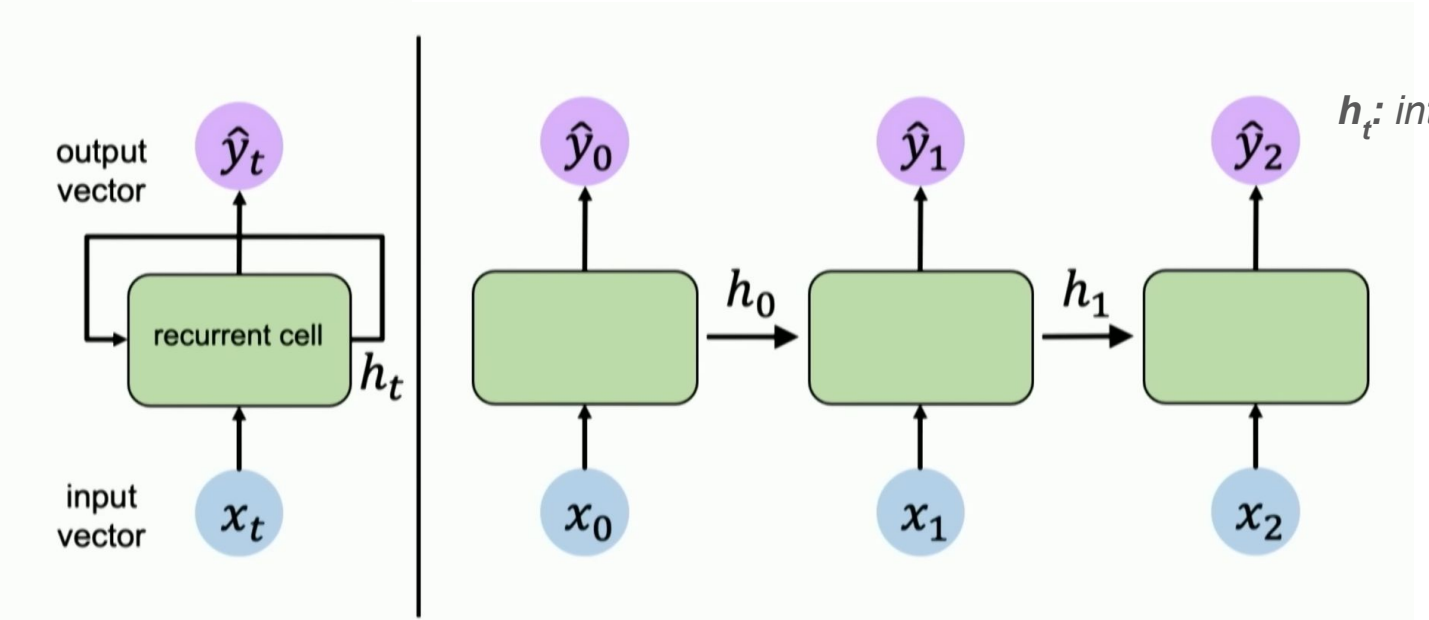
Handling individual time steps



Recurrence relation

Sequence Modeling

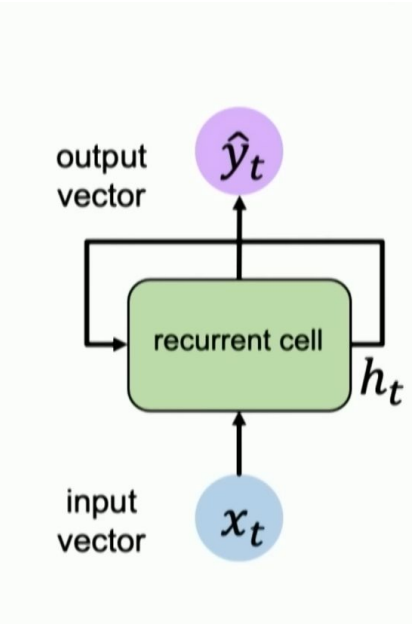
Handling individual time steps



h_t: internal state/ memory term

$$\underbrace{\hat{y}_t}_{\text{output}} = f(\underbrace{x_t}_{\text{input}}, \underbrace{h_{t-1}}_{\text{past memory}})$$

Sequence Modeling: RNNs



Apply recurrence relation at every time step to process the sequence:

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

Cell State

function with weights W

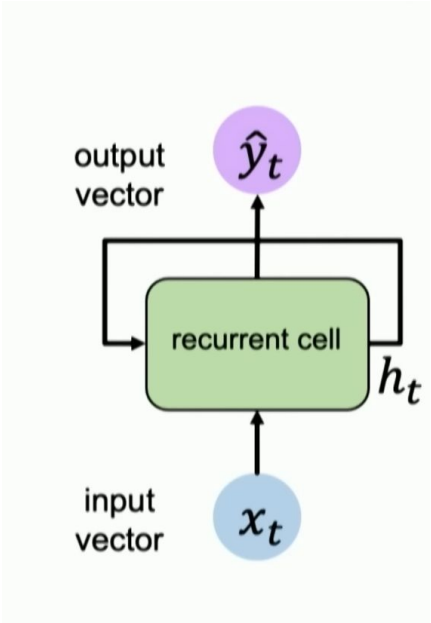
input

Old State

- The same function and set of parameters are used at every time step

RNNs have internal state h_t that is updated at each time step as a sequence is processed

Sequence Modeling: RNNs



Output Vector

$$\hat{y}_t = \boldsymbol{W}_{\boldsymbol{hy}}^{\boldsymbol{T}} h_t$$

Update Hidden State

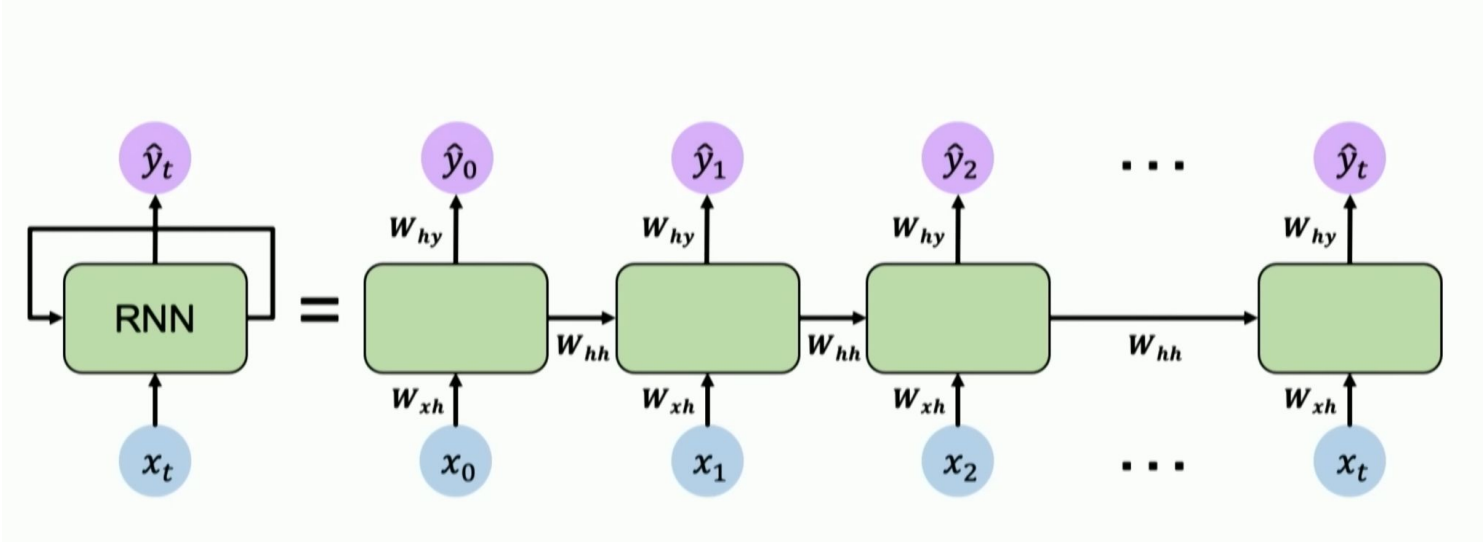
$$h_t = \tanh(\boldsymbol{W}_{\boldsymbol{hh}}^{\boldsymbol{T}} h_{t-1} + \boldsymbol{W}_{\boldsymbol{xh}}^{\boldsymbol{T}} x_t)$$

Input Vector

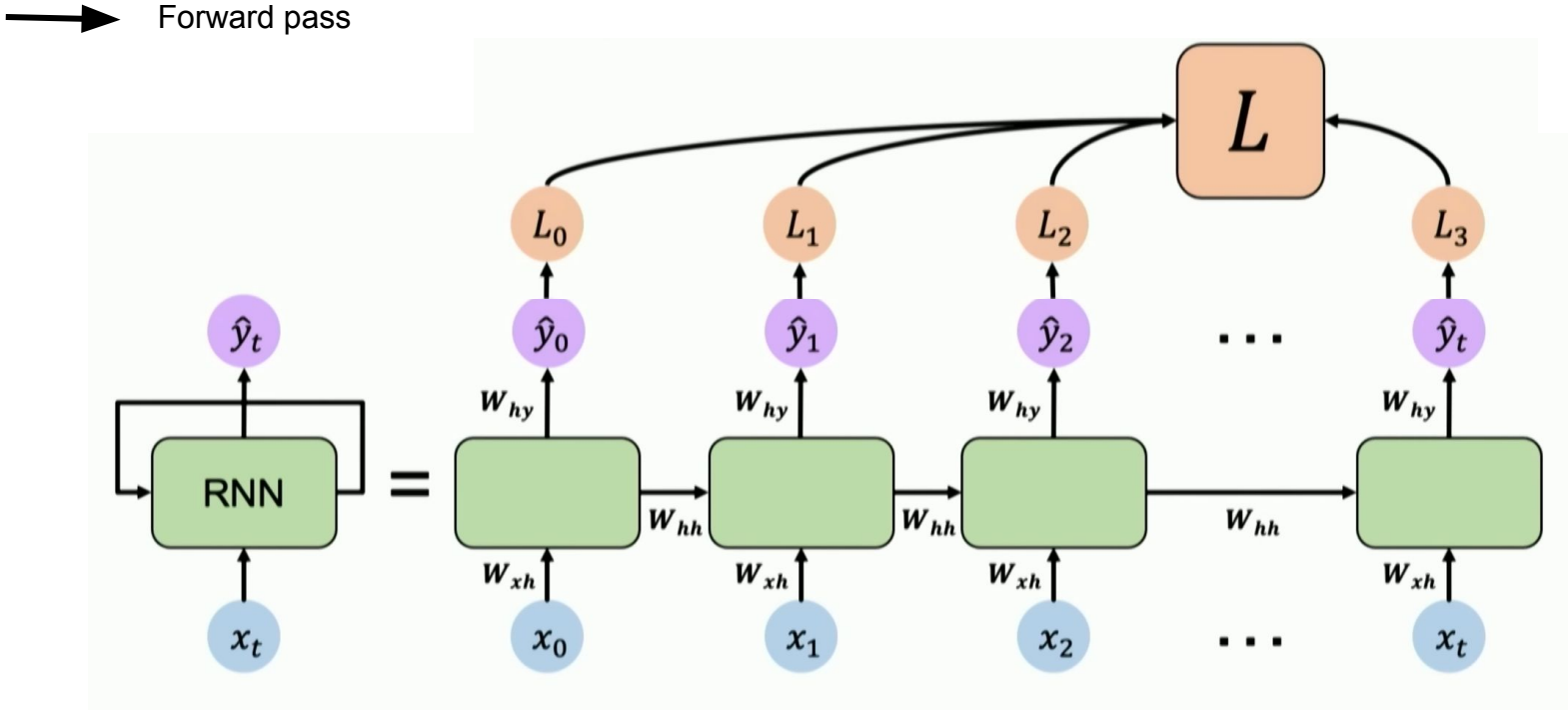
$$x_t$$

Sequence Modeling: RNNs

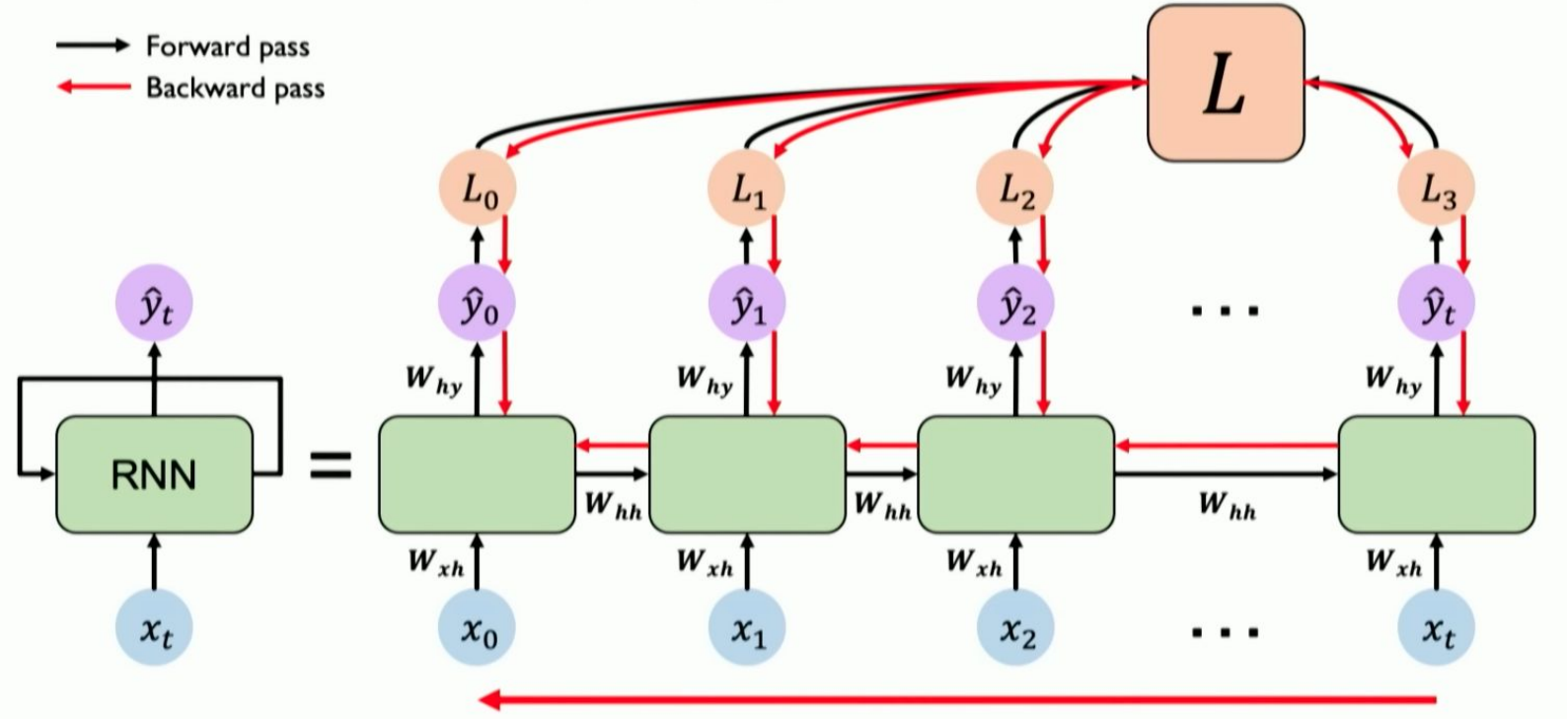
Re-using the same weight matrices at every time step



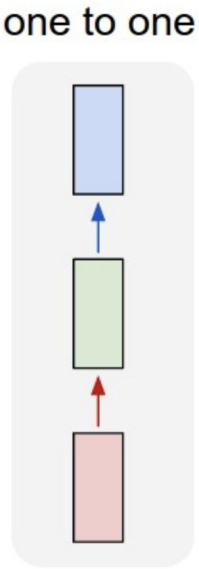
Sequence Modeling: RNNs



Sequence Modeling: RNNs



Sequence Modeling Application



Binary classification
Cat or Dog

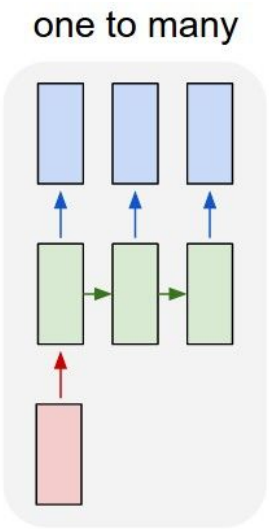
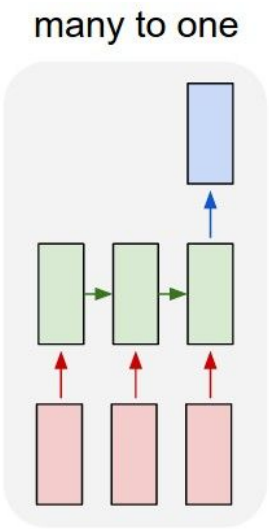
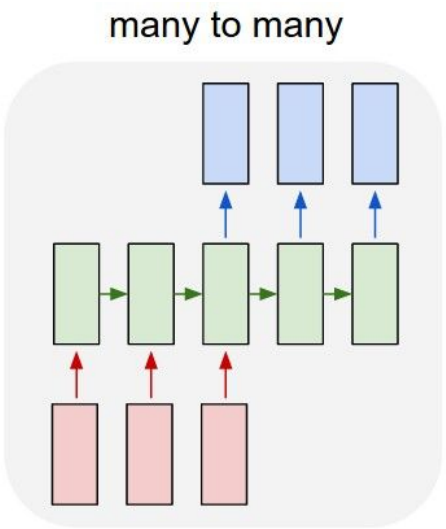


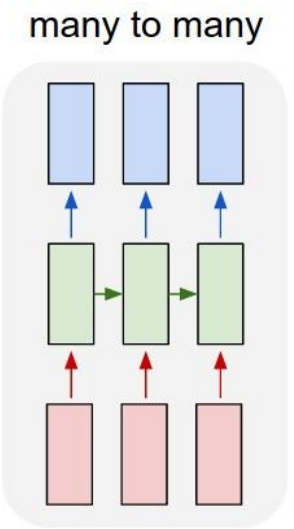
Image captioning



Sentiment classification



Machine translation



Video classification (label each frame)

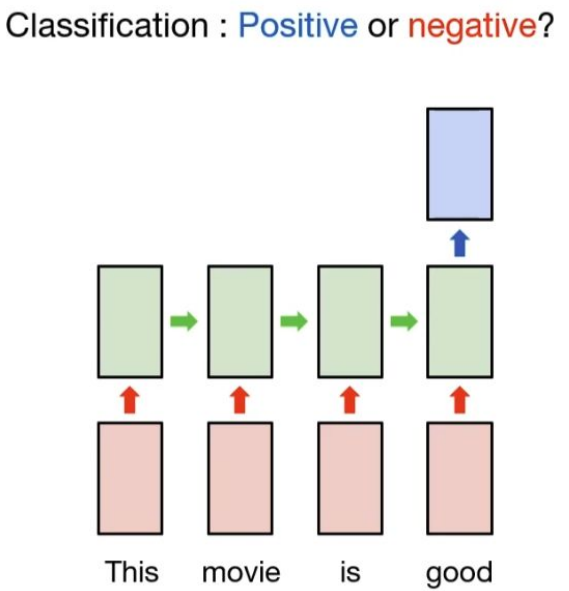
http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Sequence Modeling: Design Criteria

To model sequence we need to:

- Handle **Variable-length** sequences
- Handle **long-term** dependencies
- Maintain information about **Order**

RNN Limitations

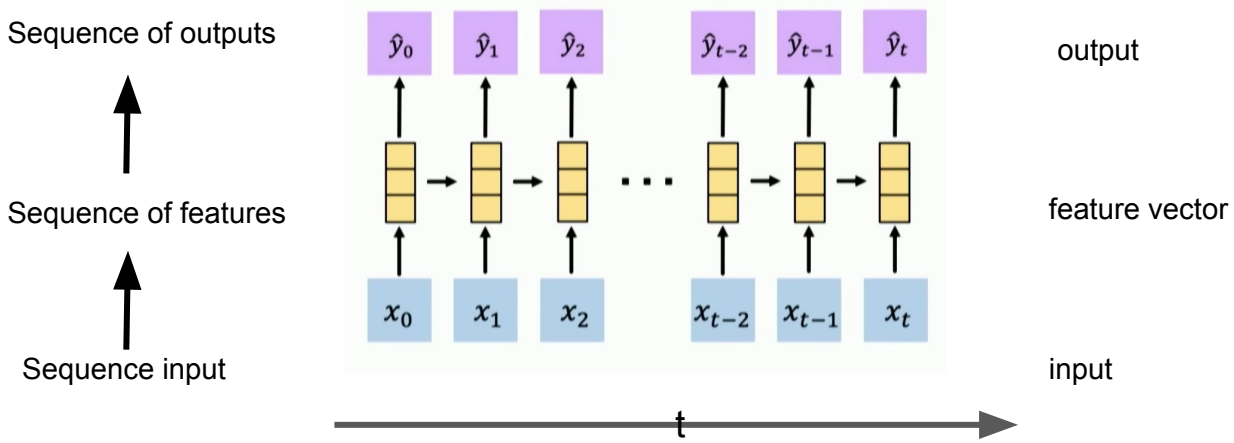


Limitations of Recurrent Models

- **Encoding bottleneck:** Information can be lost
- **Slow, No parallelization:** Not efficient training
- **No long memory:** Don't scale well for *very* long sequences

RNN Limitations

RNNs: recurrence to model sequence dependencies



RNN Limitations

- Encoding bottleneck:
- Slow, No parallelization
- No long memory

Desired Capabilities

- Continuous stream
- Parallelization
- Long memory

Transformers

Google

Bidirectional Encoder Representation from Transformers

OpenAI

Generative Pre-Training

CHAT GPT

GPT-4

Attention Is All You Need

Self-Attention

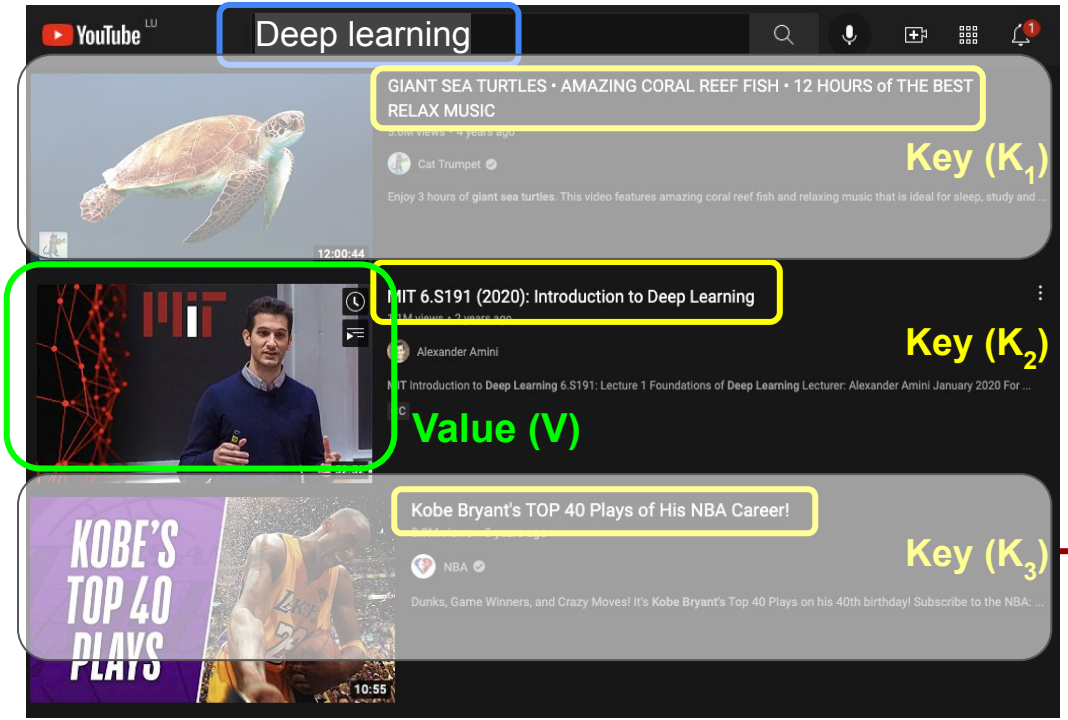
Self-Attention: Attending to the most important part of an input

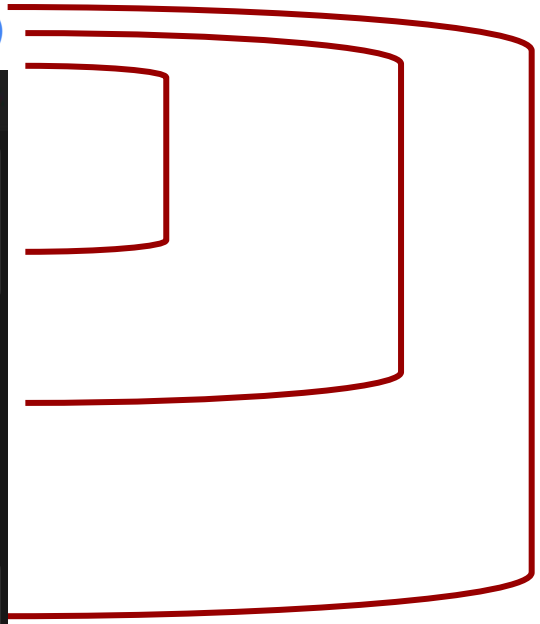
Similar to a search problem

1. Identify which parts to attend to
2. Extract the features with high attention

Self-Attention

Example: Search





1. Compute attention mask: How similar is each key to the desired query?

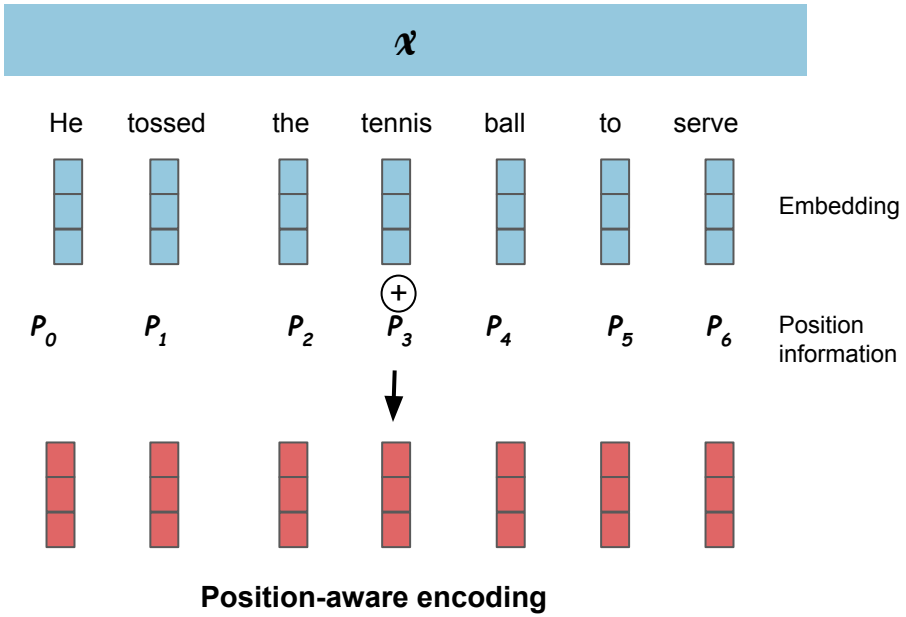
2. Extract values based on attention: Return the values with highest attention

Self-Attention

Learning Self-Attention with Neural Networks

Goal: identify and attend to most important features in input

1. Encode **position** information



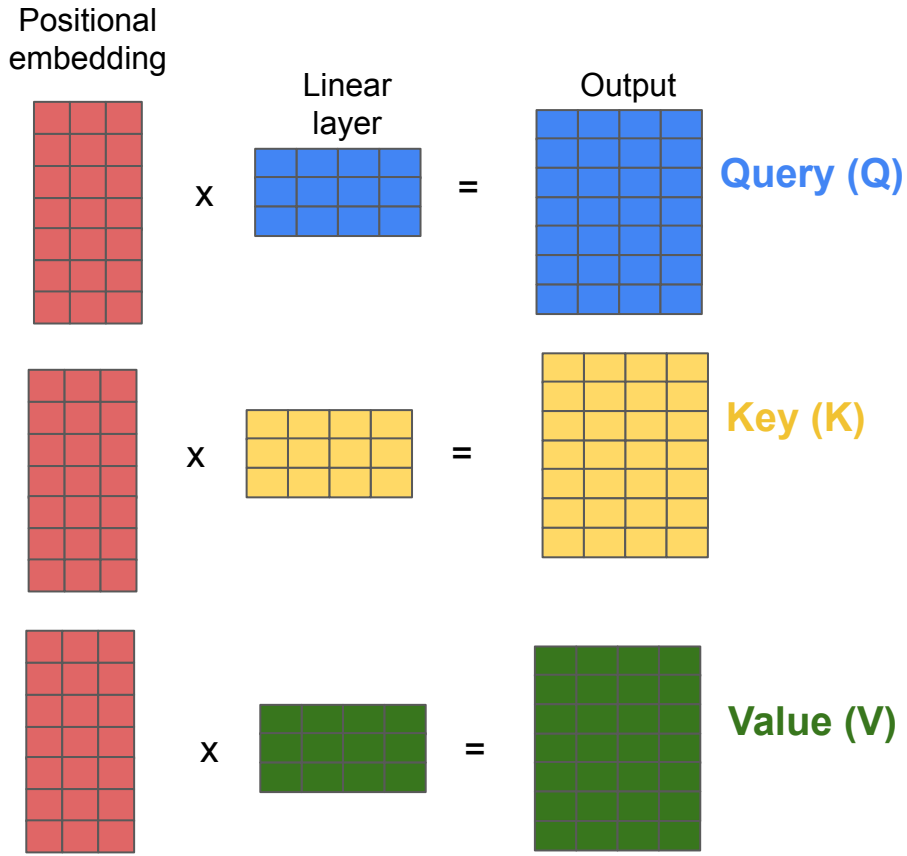
Data is fed in *all at once*! Need to encode position information to understand **order**

Self-Attention

Learning Self-Attention with Neural Networks

Goal: identify and attend to most important features in input

1. Enode **position** information
2. Extract **query, key, value** for search



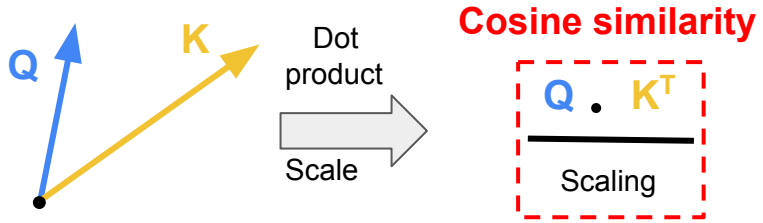
Self-Attention

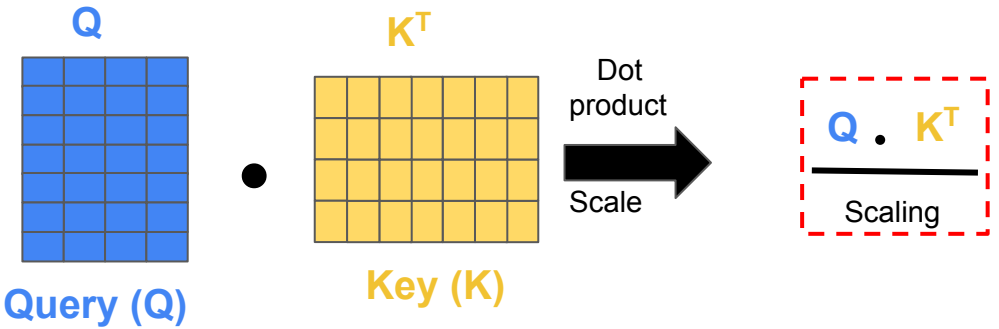
Learning Self-Attention with Neural Networks

Goal: identify and attend to most important features in input

1. Enode **position** information
2. Extract **query, key, value** for search
3. Compute **attention weighting**

Attention score: compute pairwise similarity between each **Query (Q)** and **Key (K)**





Self-Attention

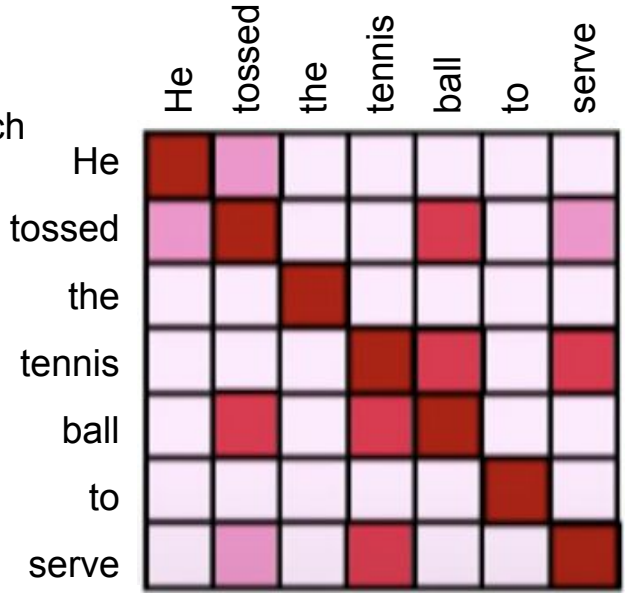
UNIVERSITÉ DU LUXEMBOURG

Learning Self-Attention with Neural Networks

Goal: identify and attend to most important features in input

Attention weighting: captures where in the input to attend to!
How similar is the key to the query?

1. Enode **position** information
2. Extract **query, key, value** for search
3. Compute **attention weighting**



Attention weighting

$$softmax\left(\frac{Q \cdot K^T}{\text{Scaling}}\right)$$

Where in the input to self-attend

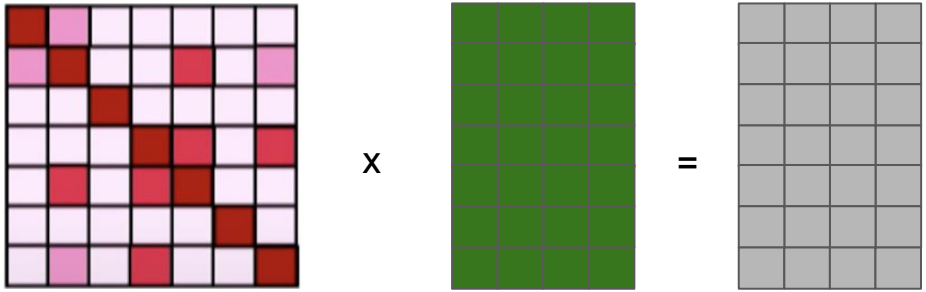
Self-Attention

Learning Self-Attention with Neural Networks

Goal: identify and attend to most important features in input

1. Enode **position** information
2. Extract **query, key, value** for search
3. Compute **attention weighting**
4. Extract **features with high attention**

Last step: self-attend to extract features



$$softmax\left(\frac{Q \cdot K^T}{\text{Scaling}}\right) \cdot V = A(Q, K, V)$$

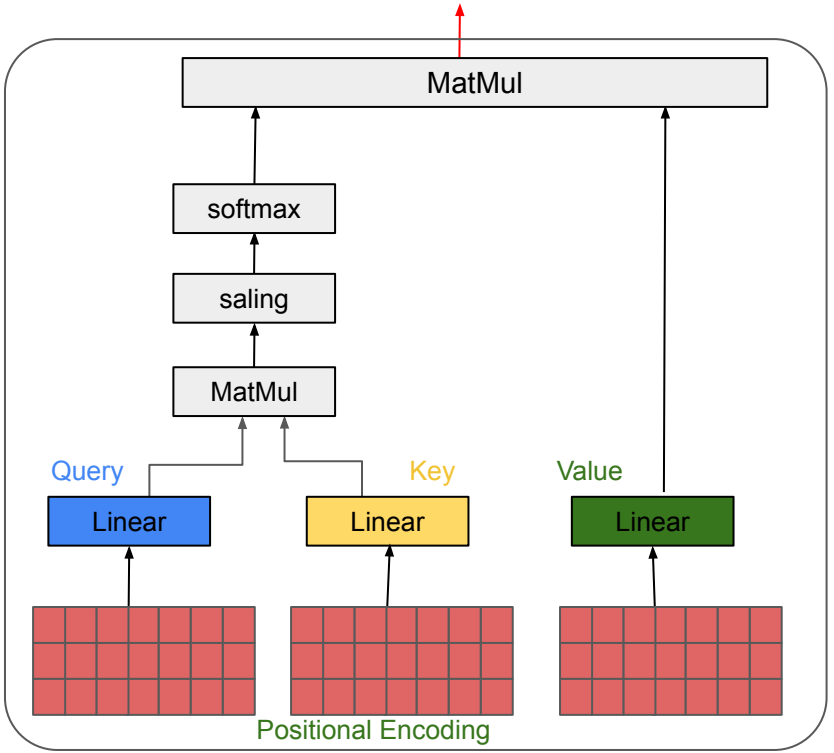
Self-Attention

Goal: identify and attend to most important features in input

1. Enode **position** information
2. Extract **query, key, value** for search
3. Compute **attention weighting**
4. Extract **features with high attention**

$$softmax\left(\frac{Q \cdot K^T}{\text{Scaling}}\right) \cdot V$$

Self-Attention head that can plug into larger network
Each head attends to different parts of the input



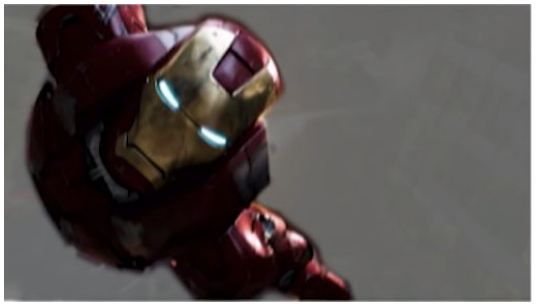
Applying Multiple Self-Attention Heads

Attention weighting

x

Value

=



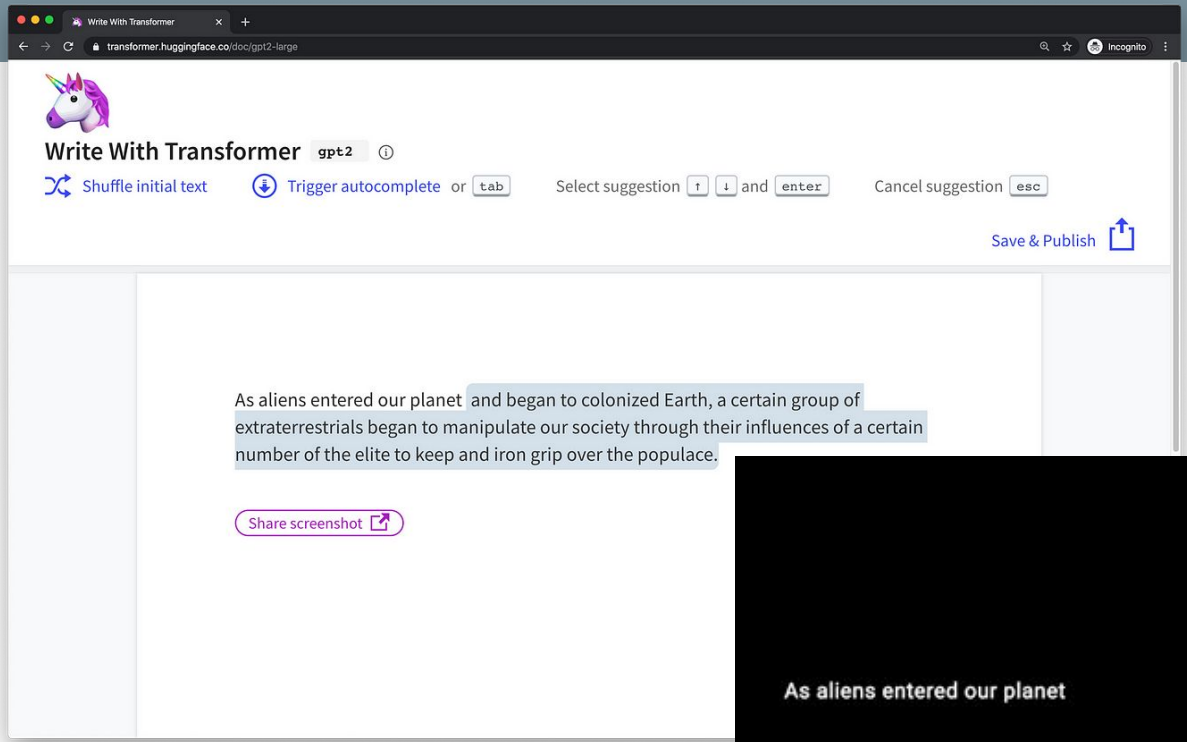
Output

Output of attention head 1

Output of attention head 2

Output of attention head 3

Self-Attention

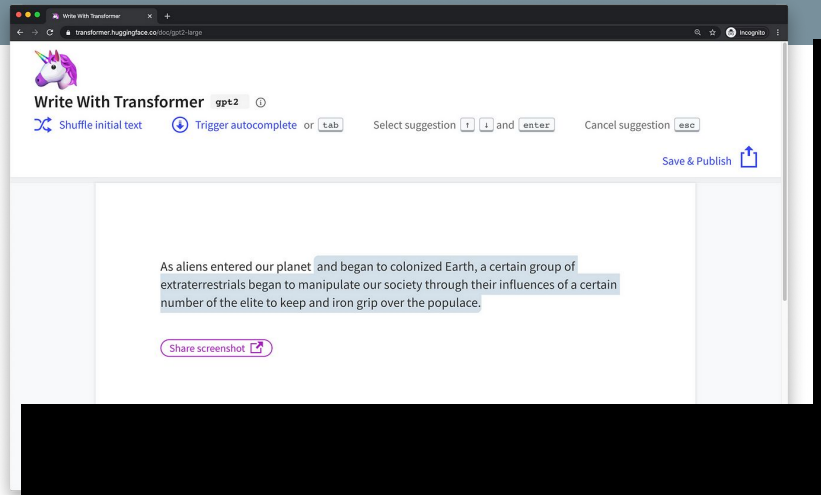


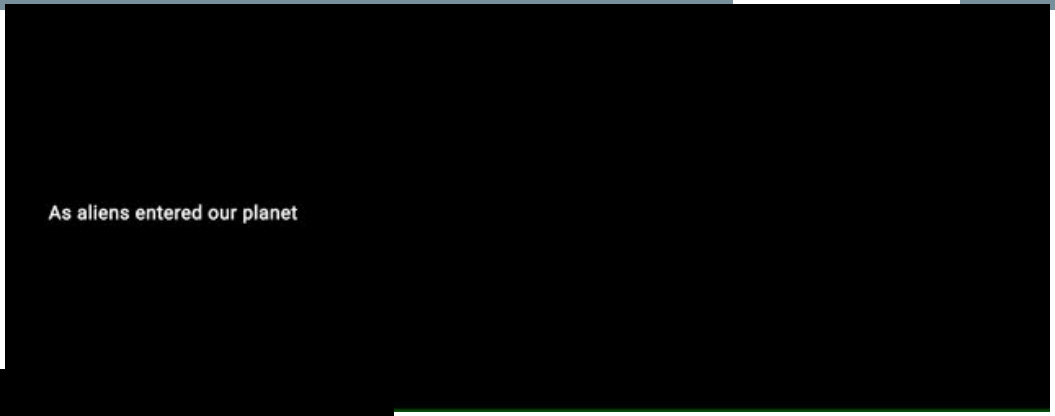
Our input: *"As Aliens entered our planet".*

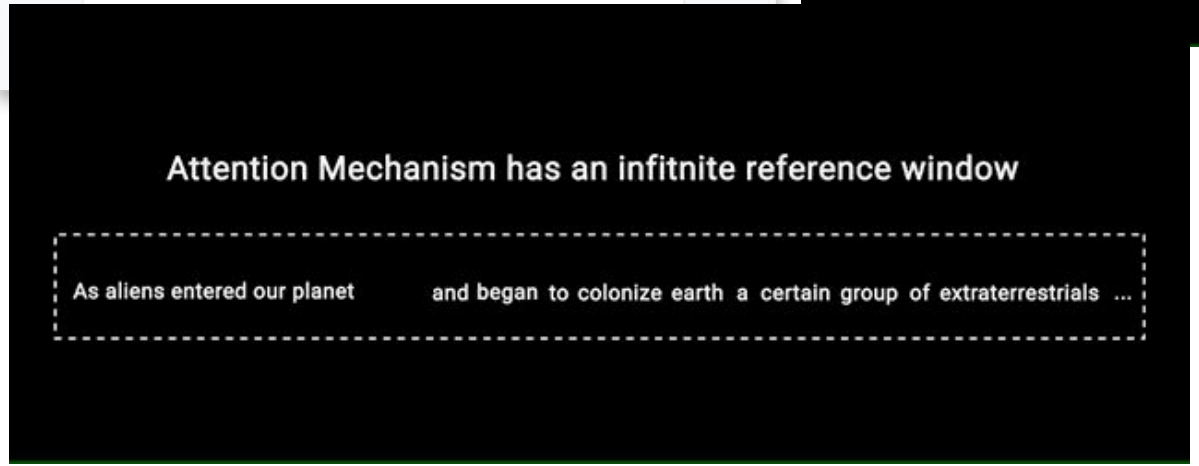
The ability to reference/ attend to parts of the input relevant to the generated word.

https://transformer.huggingface.co/

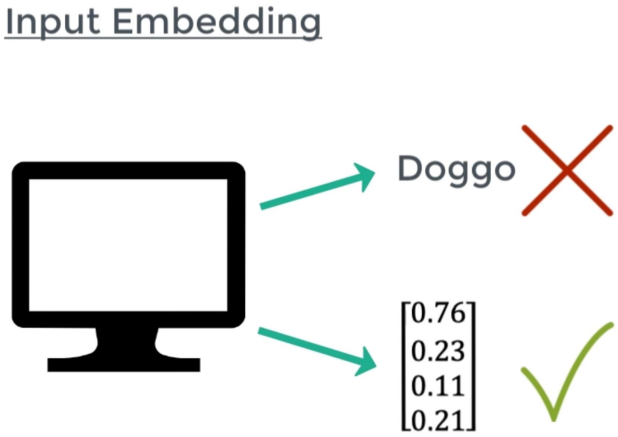
Self-Attention

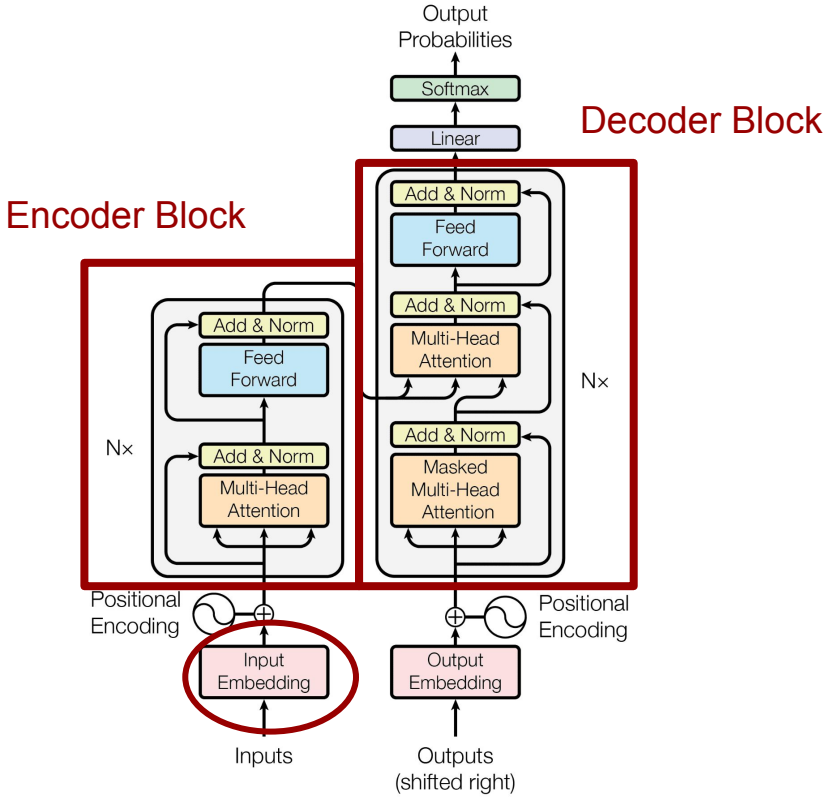




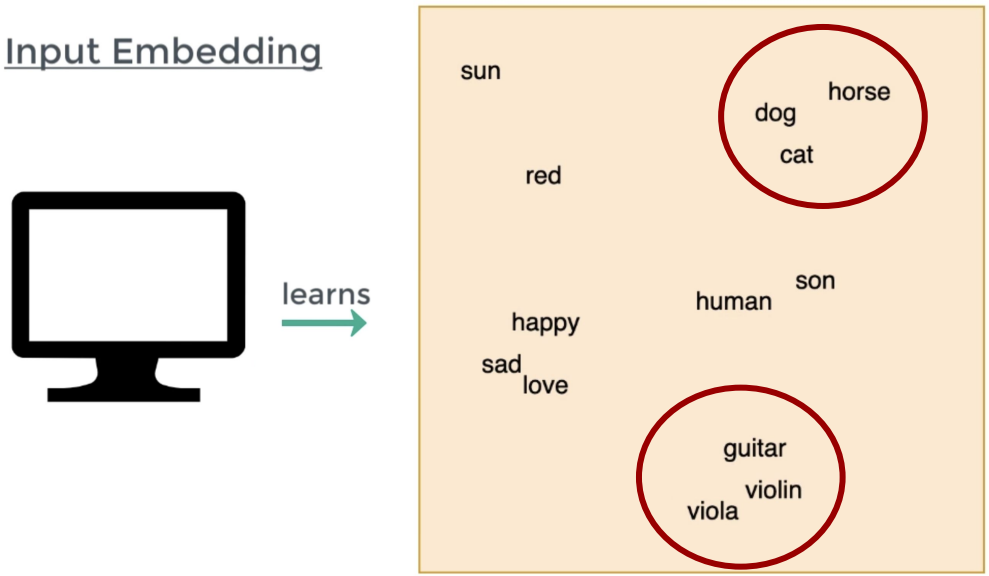


Transformers components





Transformers components



Embedding Space

Transformers components

dog → learns →

Embedding Space

→ $\begin{bmatrix} 0.37 \\ 0.99 \\ 0.01 \\ 0.08 \end{bmatrix}$

AJ's dog is a cutie

AJ looks like a dog

The same word may have different meaning in different sentences

Transformers components

Positional Encoder :vector that gives context based on position of word in sentence

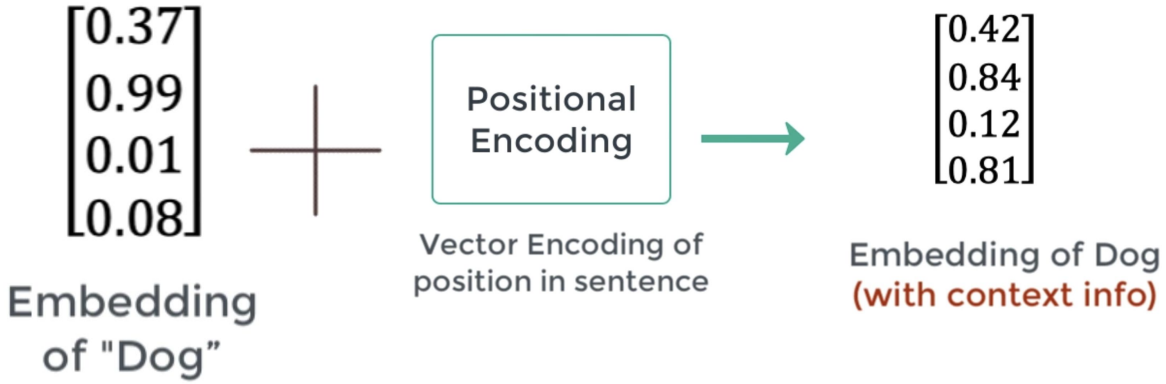
AJ's dog is a cutie → Position 2

AJ looks like a dog → Position 5

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}})$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}})$$

Transformers components



$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}})$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}})$$

Transformers components

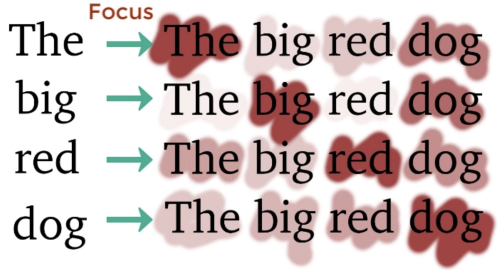
Attention Vectors

$[0.71 \quad 0.04 \quad 0.07 \quad 0.18]^T$

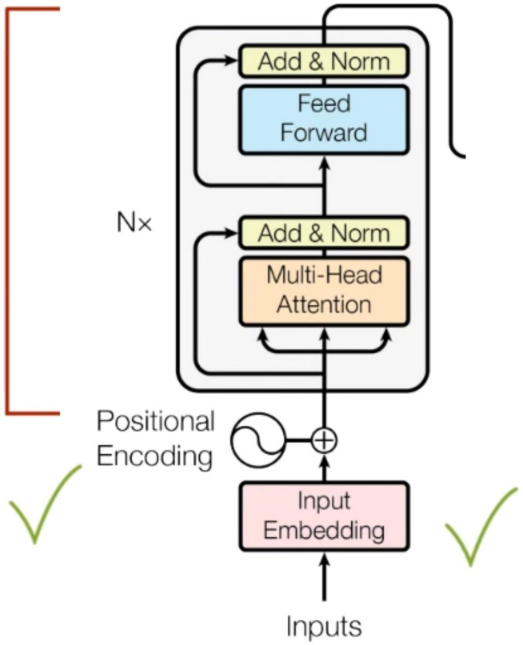
$[0.01 \quad 0.84 \quad 0.02 \quad 0.13]^T$

$[0.09 \quad 0.05 \quad 0.62 \quad 0.24]^T$

$[0.03 \quad 0.03 \quad 0.03 \quad 0.91]^T$



Encoder Block



Transformers components

UNIVERSITÉ DU LUXEMBOURG

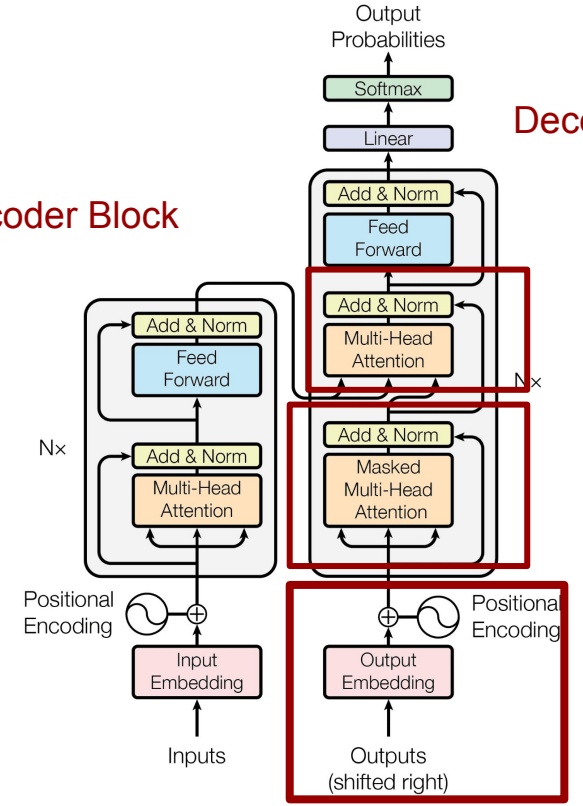
Attention Vectors

$$[0.71 \quad 0.04 \quad 0.07 \quad 0.18]^T$$
$$[0.01 \quad 0.84 \quad 0.02 \quad 0.13]^T$$
$$[0.09 \quad 0.05 \quad 0.62 \quad 0.24]^T$$
$$[0.03 \quad 0.03 \quad 0.03 \quad 0.91]^T$$

Focus

The → The big red dog
big → The big red dog
red → The big red dog
dog → The big red dog

Encoder Block



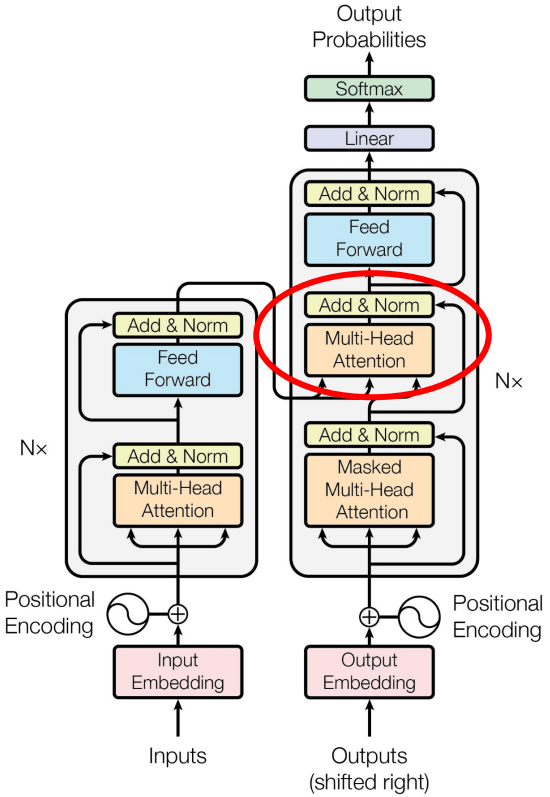
Decoder Block

$$\begin{bmatrix} 1 & 0.1 & 0.05 & 0.16 \\ 0 & 0.9 & 0.40 & 0.09 \\ 0 & 0 & 0.55 & 0.15 \\ 0 & 0 & 0 & 0.66 \end{bmatrix}$$

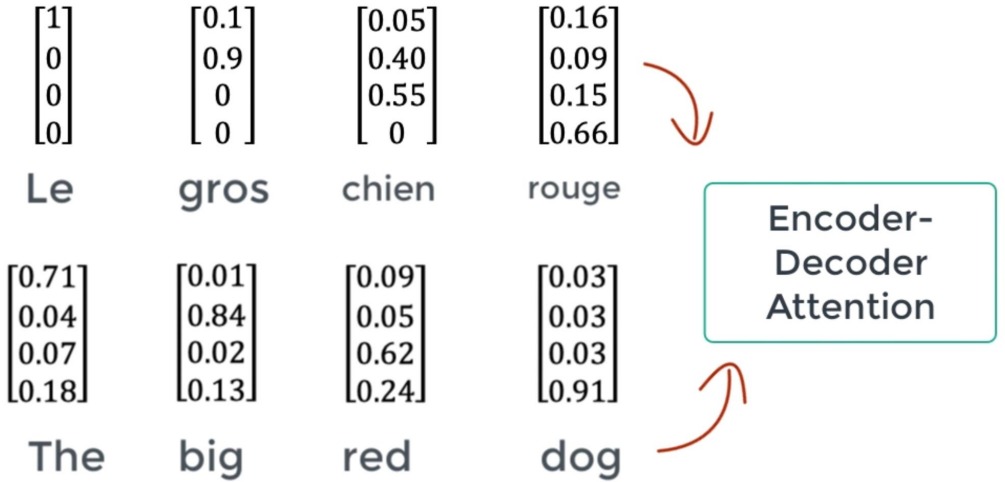
Self Attention

Le → Le gros chien rouge
gros → Le gros chien rouge
chien → Le gros chien rouge
rouge → Le gros chien rouge

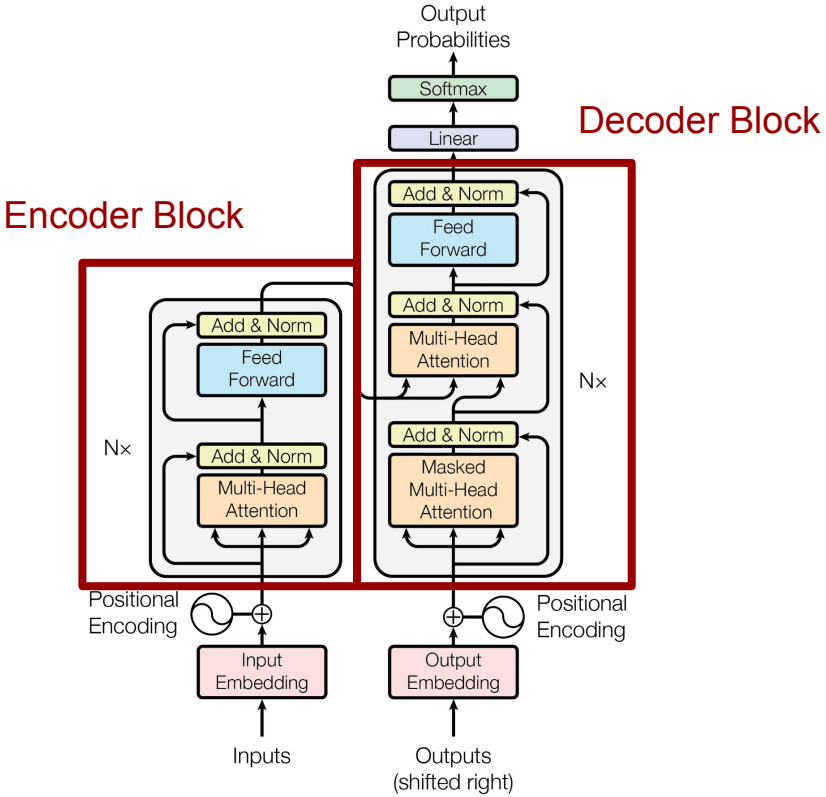
Transformers components



Encapsulates **English-French** interaction



Transformers components



Transformers applications

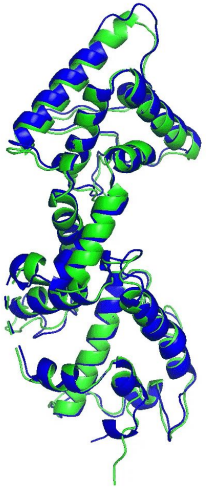
Beyond
Language processing:
Text-to-image

"An armchair in the shape of avocado"

BERT, GPT-3 DALL.E

Biological Sequence

AlphaFold2

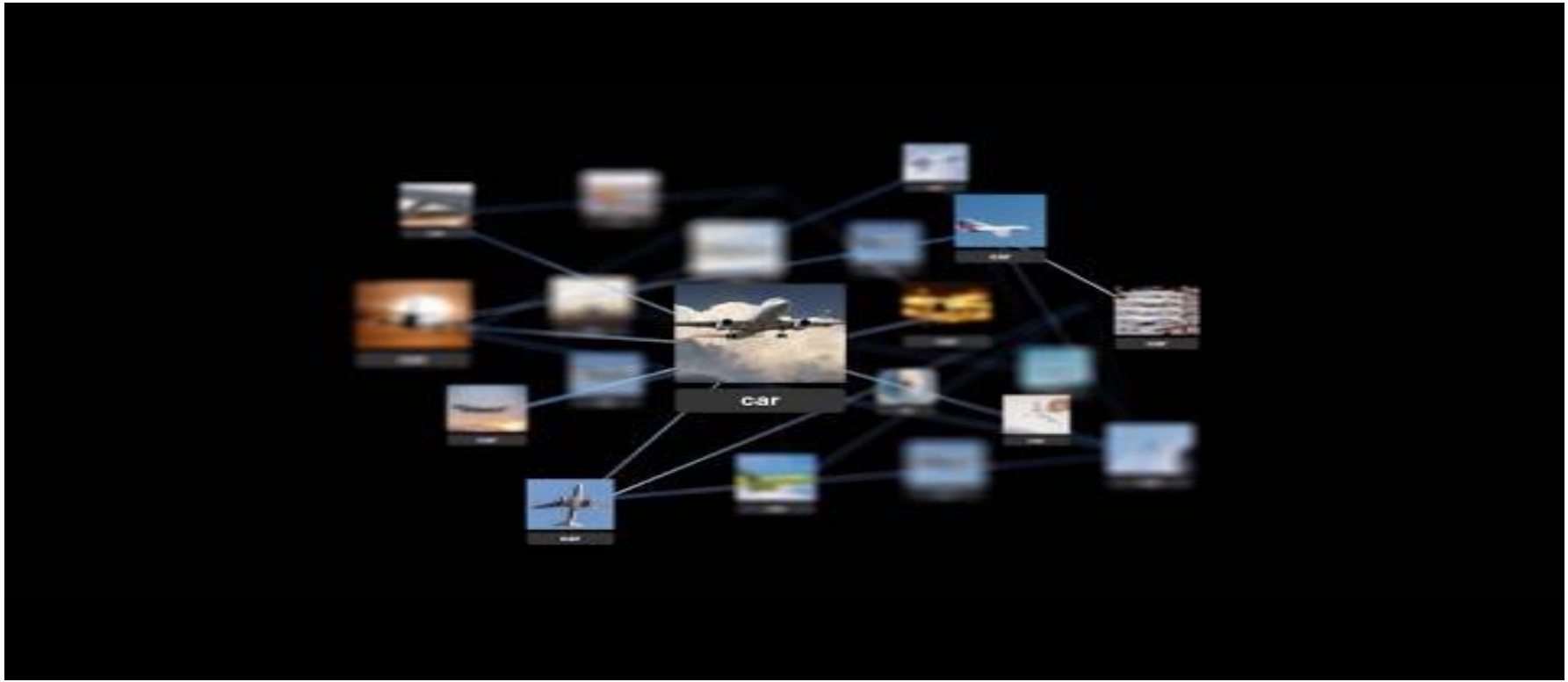


Protein structure
predictions

Image segmentation

Vision Transformers

Sequence Modeling



Sequence Modeling

UNIVERSITÉ DU LUXEMBOURG

LSTM is dead. Long Live Transformers!

- **Multi-Headed Attention:** fully parallelizable
- **Positional encoding:** preserve sequential information

🤗 Hugging Face's Transformers

- Both PyTorch and Tensorflow implementations
- Pre-trained models
- Easy to fine-tune

Further resources:

- The Illustrated Transformer
- The Illustrated Retrieval Transformer