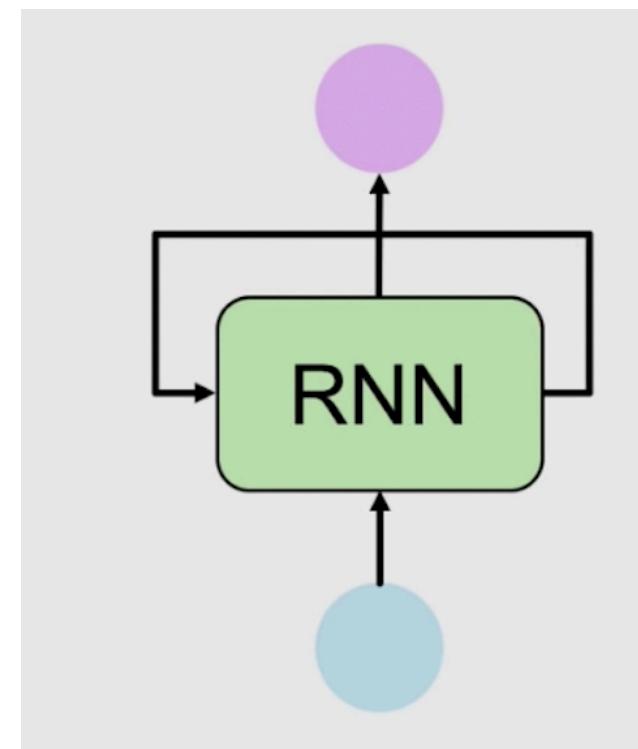


# **RECURRENT NEURAL NETWORKS (RNNs)**

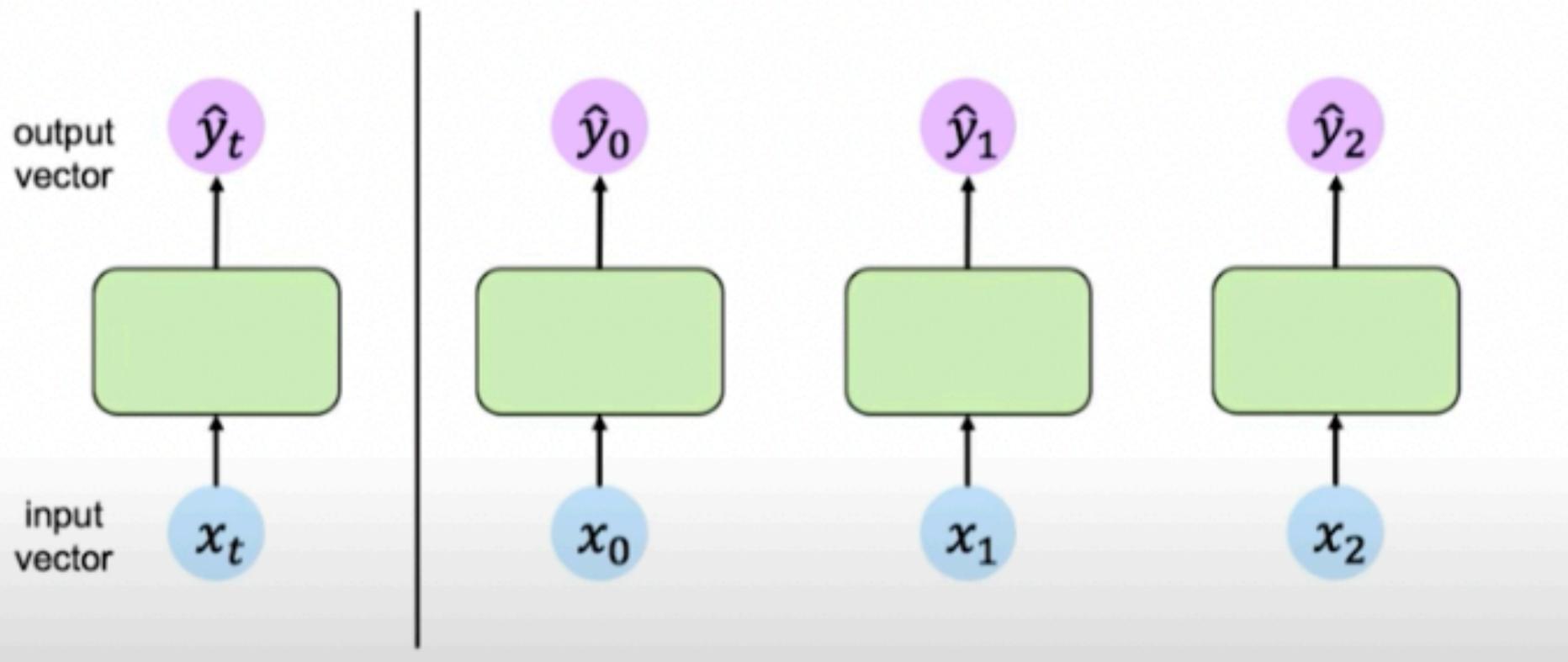
## OUR WISH LIST:

1. Handle variable-length sequences
2. Track long term dependencies
3. Maintain information about order
4. Share parameters across the sequence

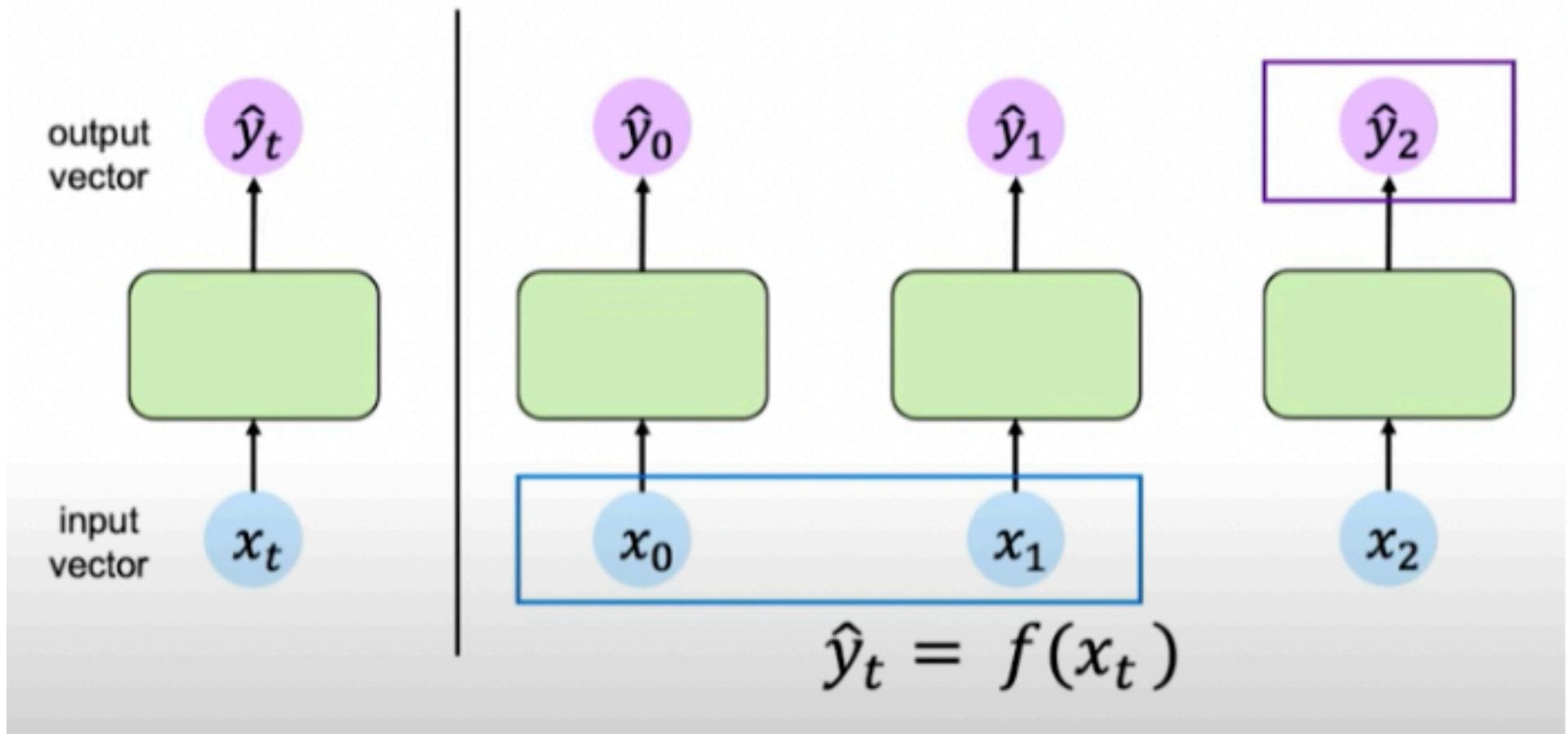
Recurrent Neural Networks  
offer a first solution to this problem



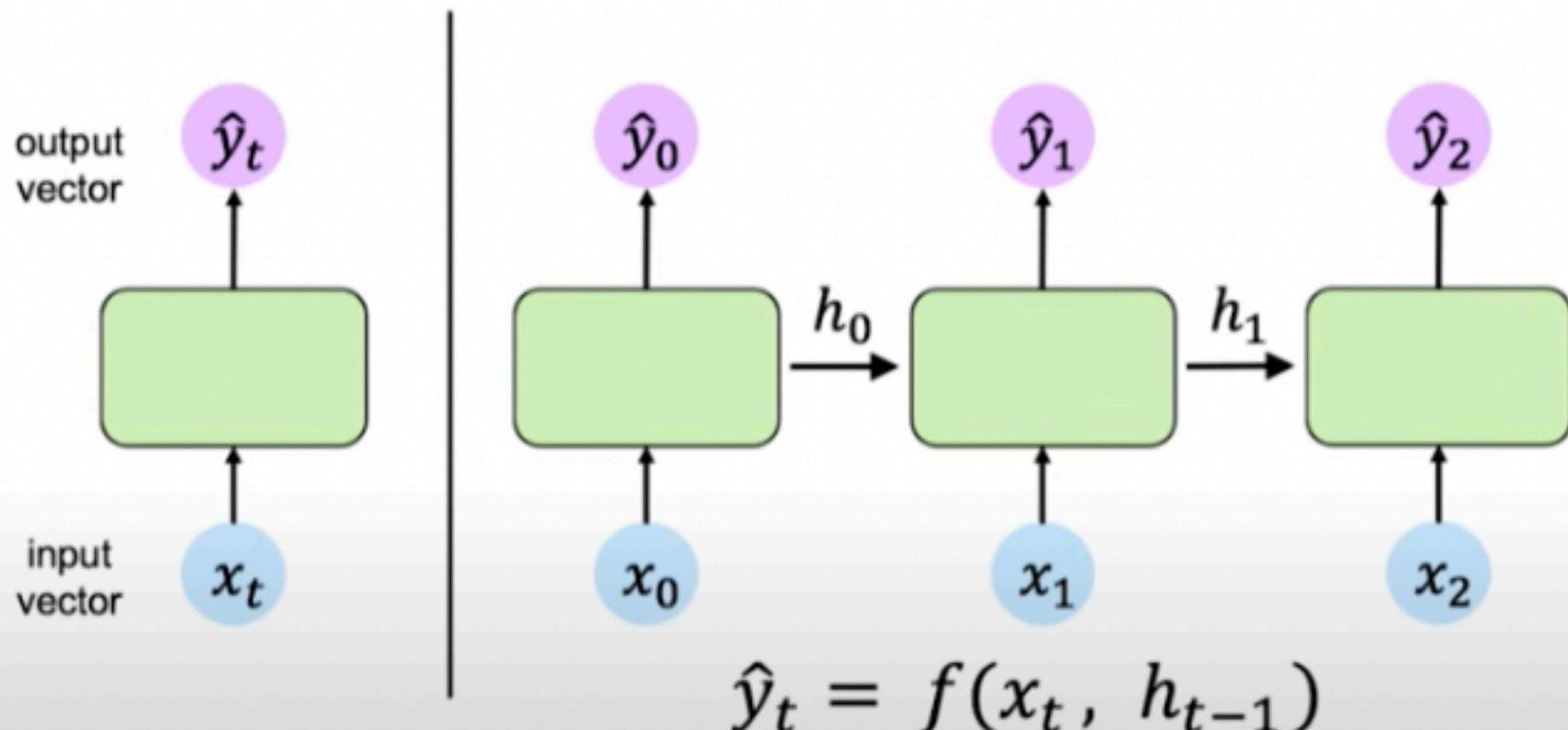
# Handling Individual Time Steps



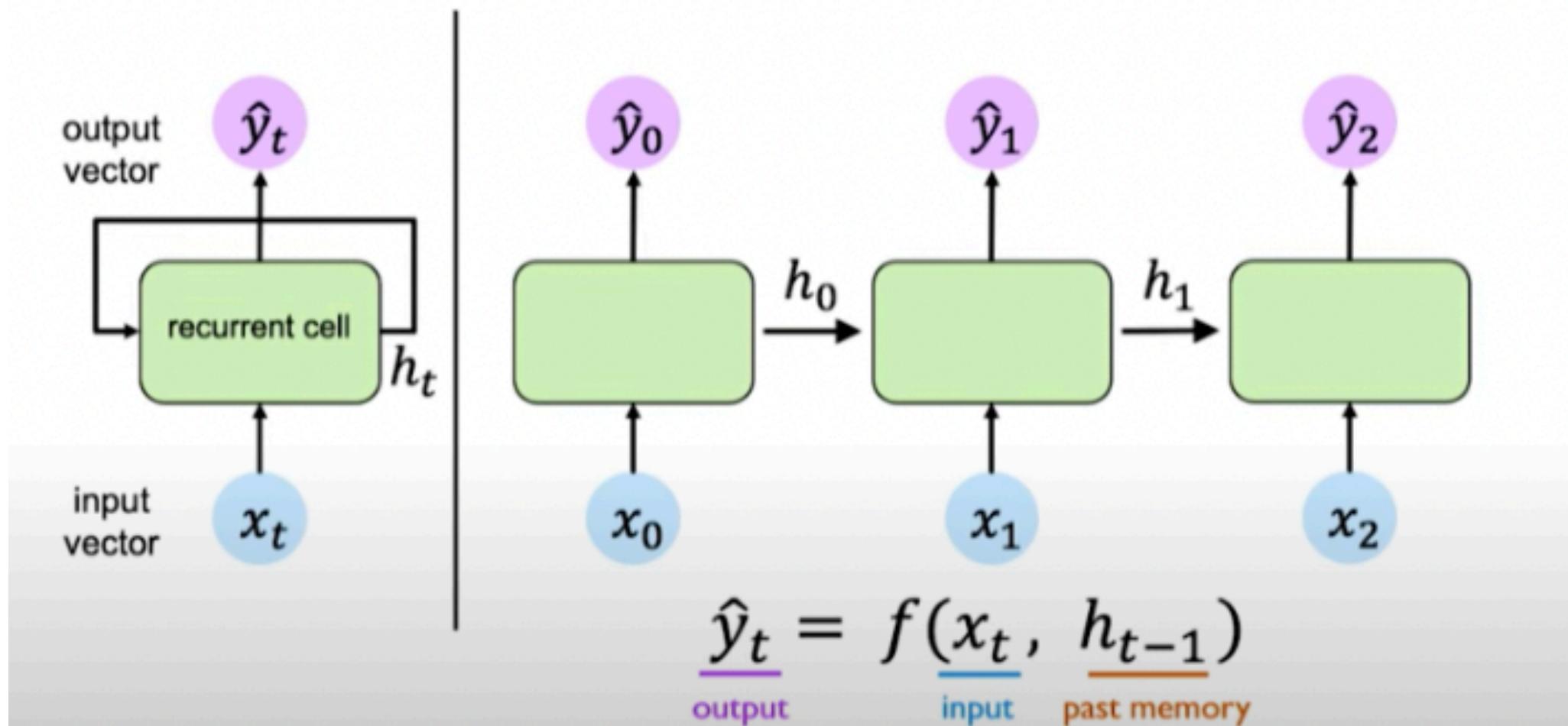
# Handling Individual Time Steps



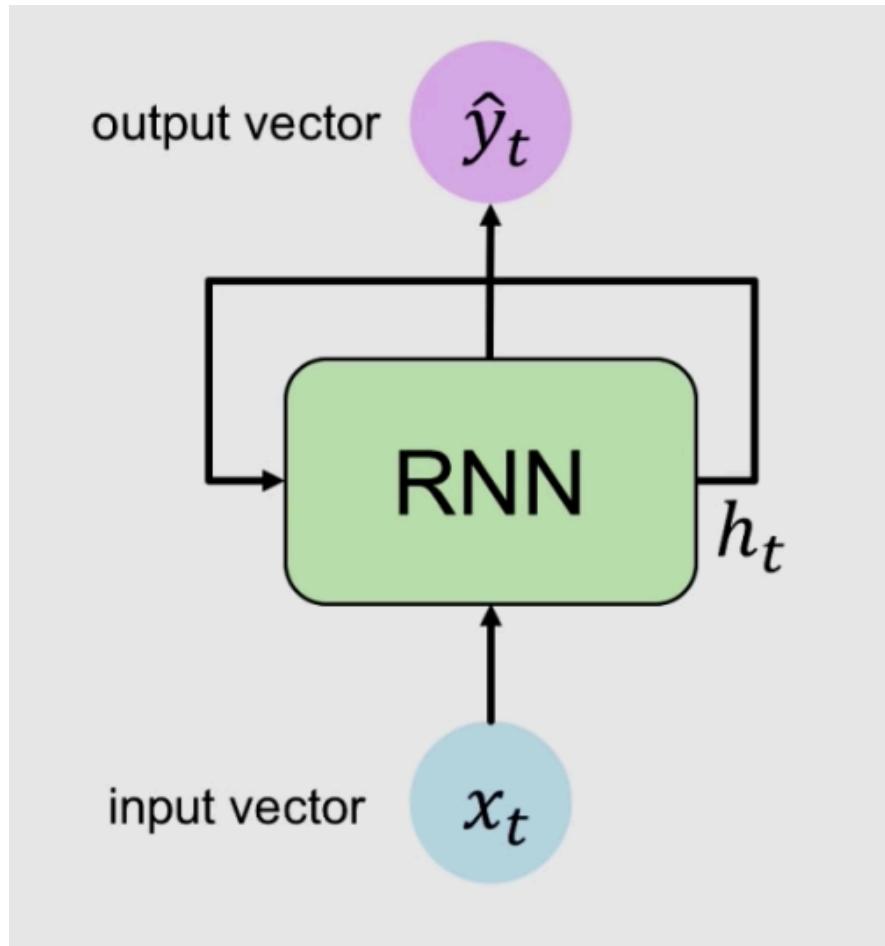
# Neurons with Recurrence



# Neurons with Recurrence



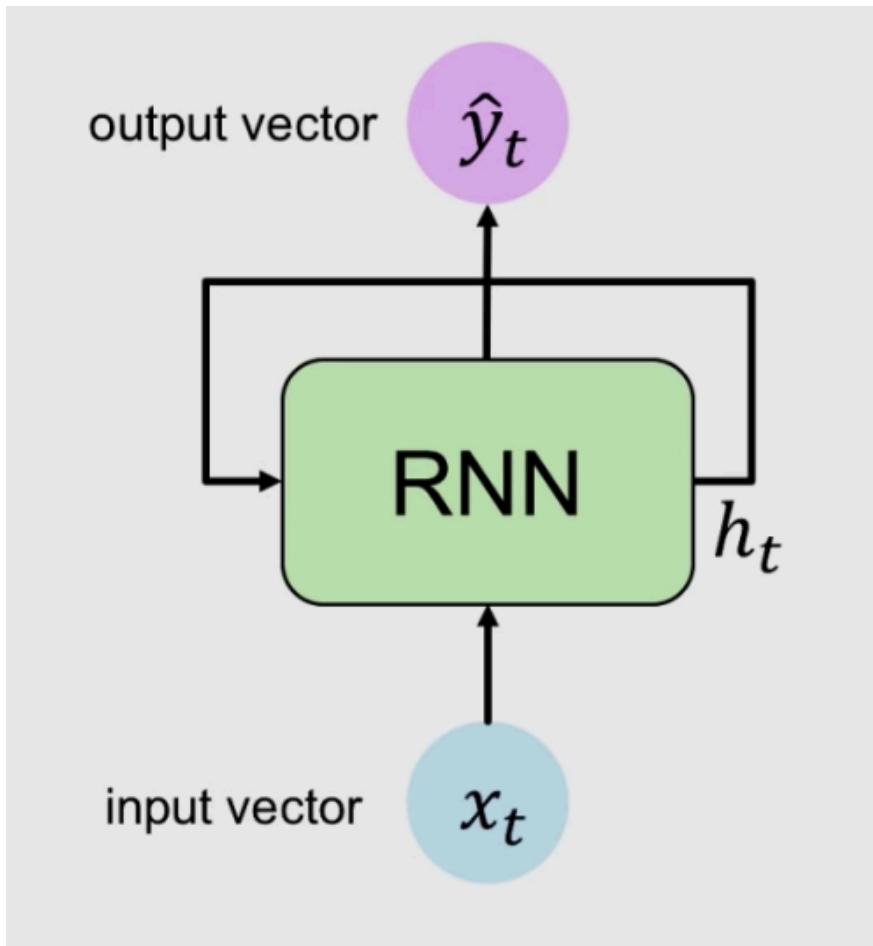
# The RNN Block



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

cell state      function parameterized by W  
old state      input vector at time step t

# The RNN Block



Output Vector

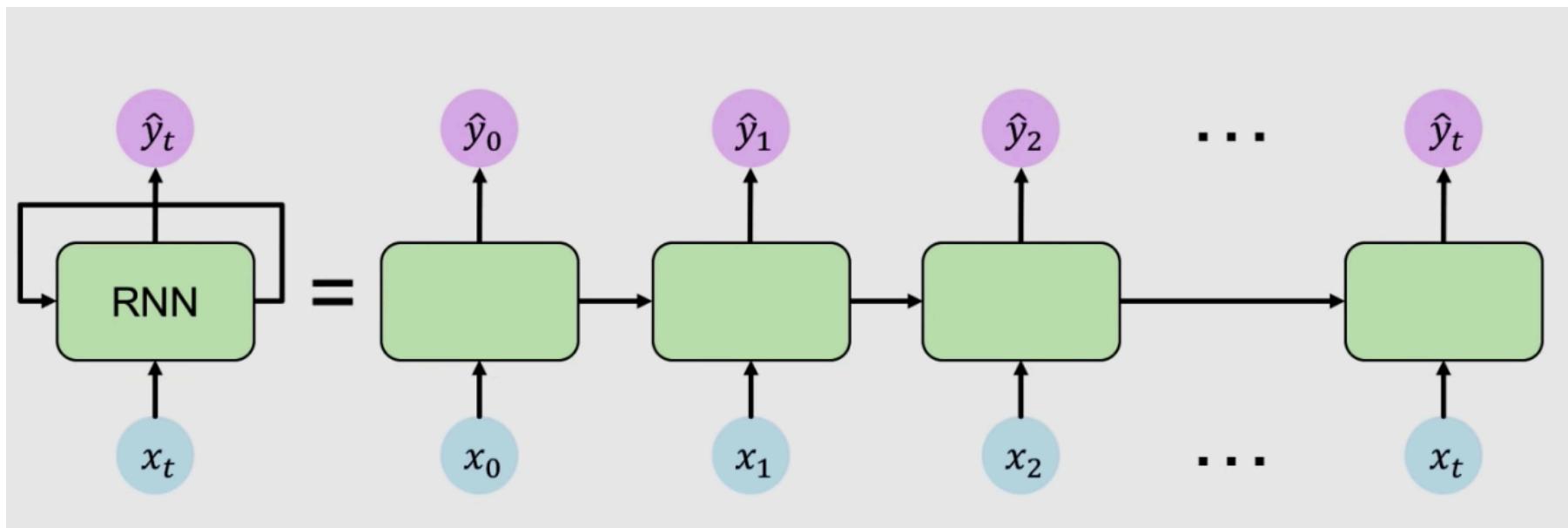
$$\hat{y}_t = \mathbf{W}_{hy}^T h_t$$

Update Hidden State

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

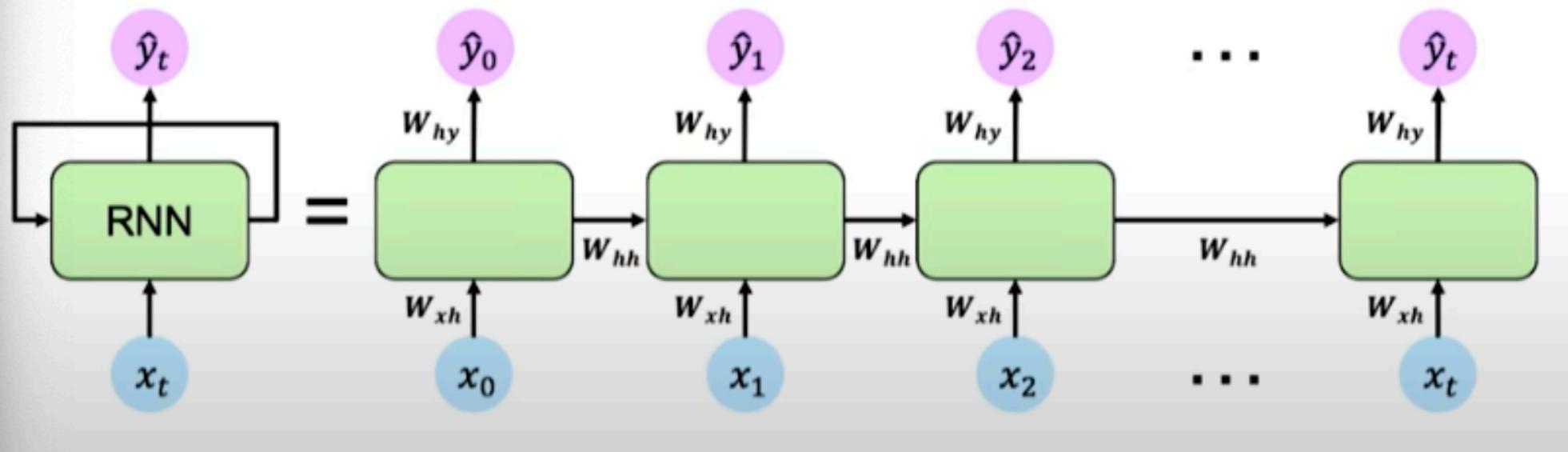
Input Vector

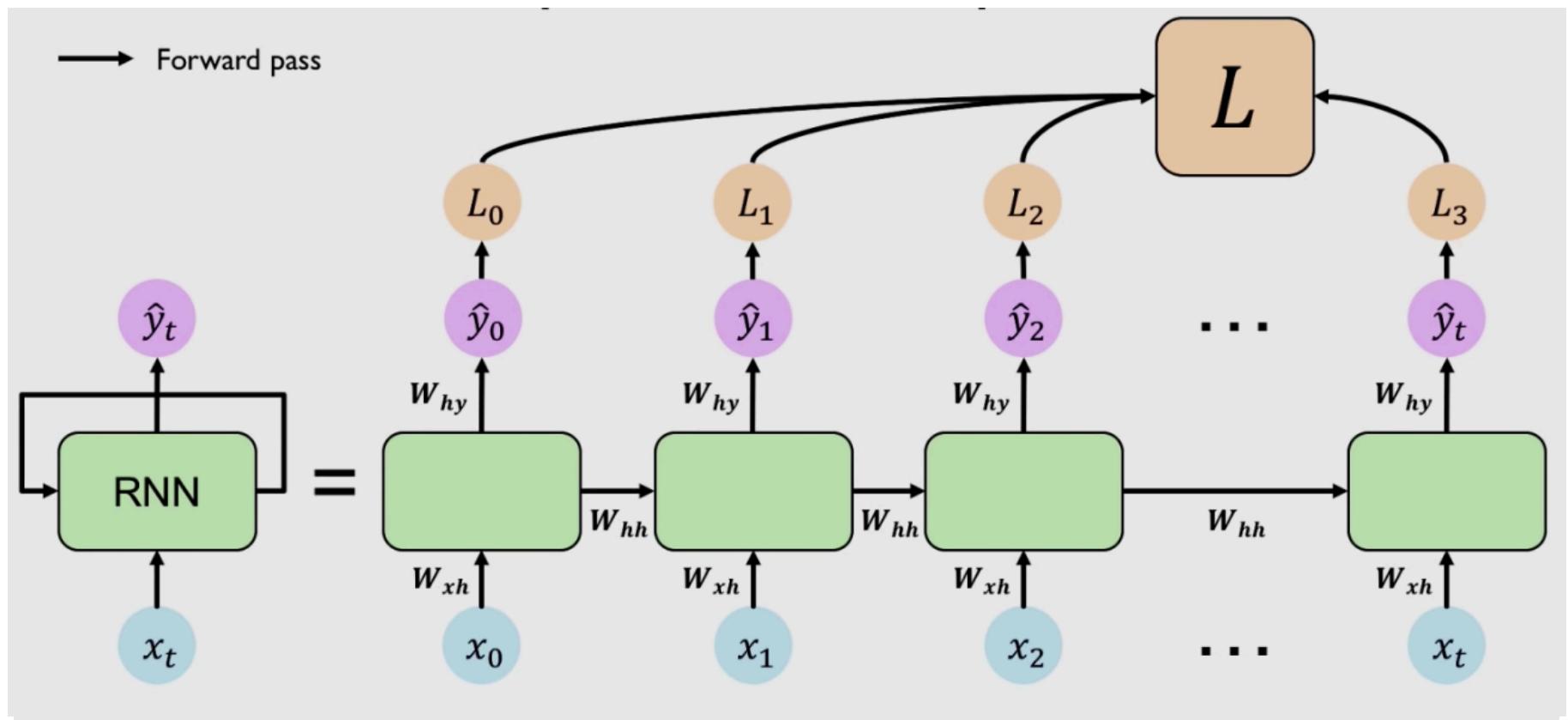
$$x_t$$



# RNNs: Computational Graph Across Time

Re-use the **same weight matrices** at every time step





# A Sequence Modeling Problem: Predict the Next Word

"This morning I took my cat for a walk."

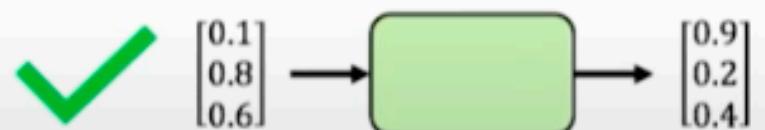
given these words

predict the  
next word

## Representing Language to a Neural Network



*Neural networks cannot interpret words*

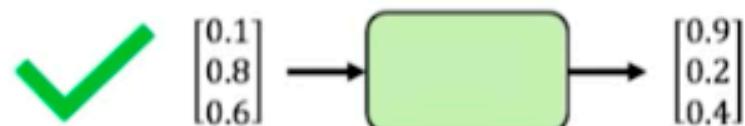


*Neural networks require numerical inputs*

# Encoding Language for a Neural Network



*Neural networks cannot interpret words*



*Neural networks require numerical inputs*

**Embedding:** transform indexes into a vector of fixed size.

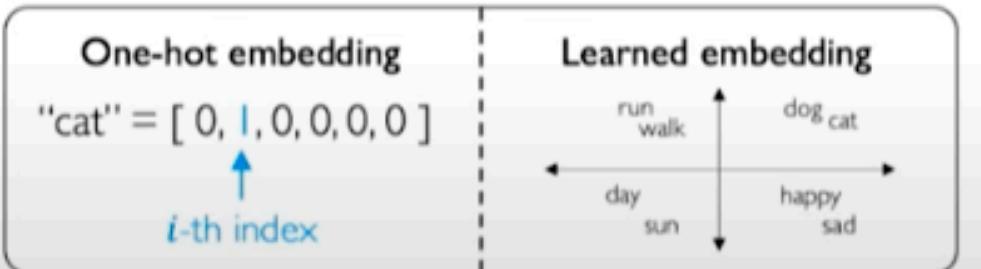
|      |      |      |
|------|------|------|
| this | cat  | for  |
| my   | took | I    |
| a    | I    | walk |

morning

**1. Vocabulary:**  
Corpus of words

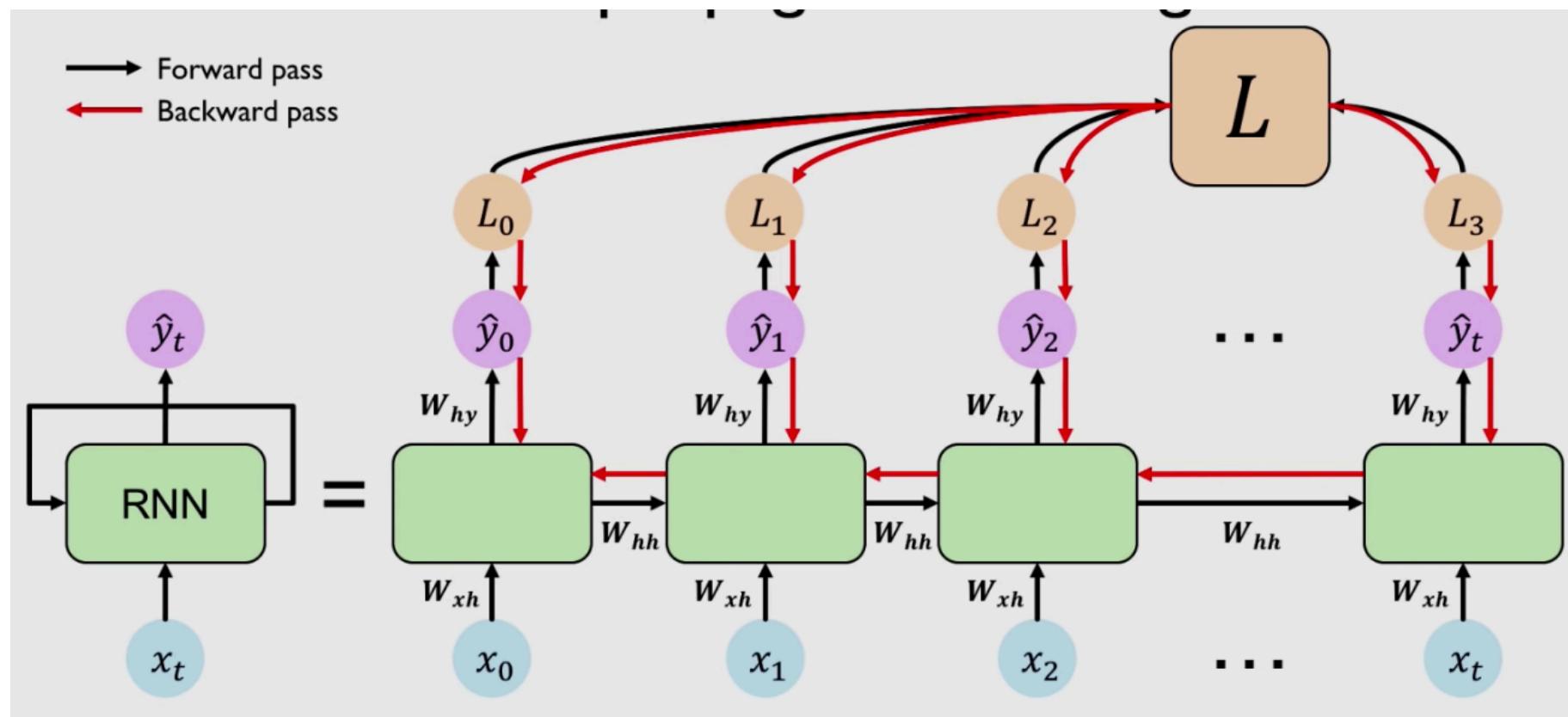
|      |   |     |
|------|---|-----|
| a    | → | 1   |
| cat  | → | 2   |
| ...  | → | ... |
| walk | → | N   |

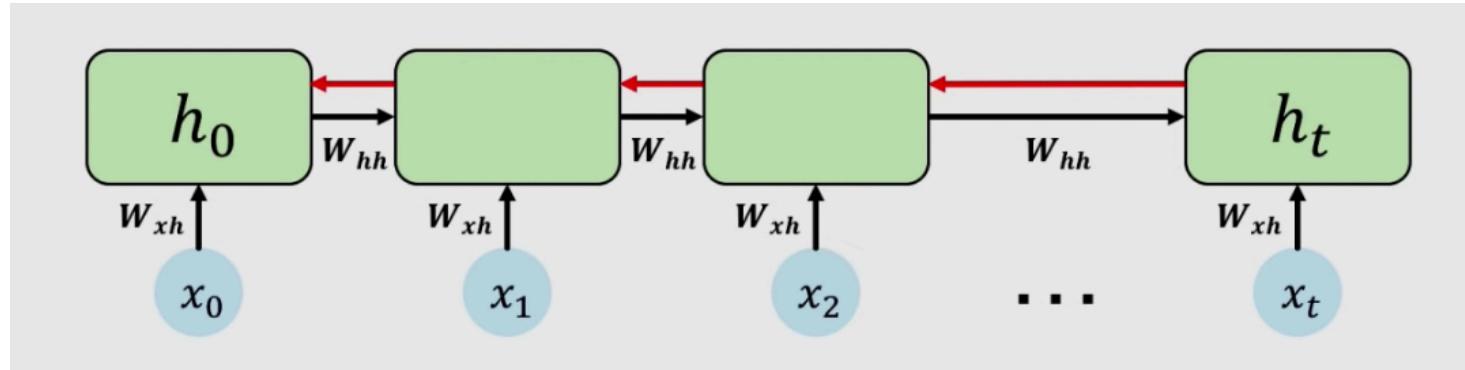
**2. Indexing:**  
Word to index



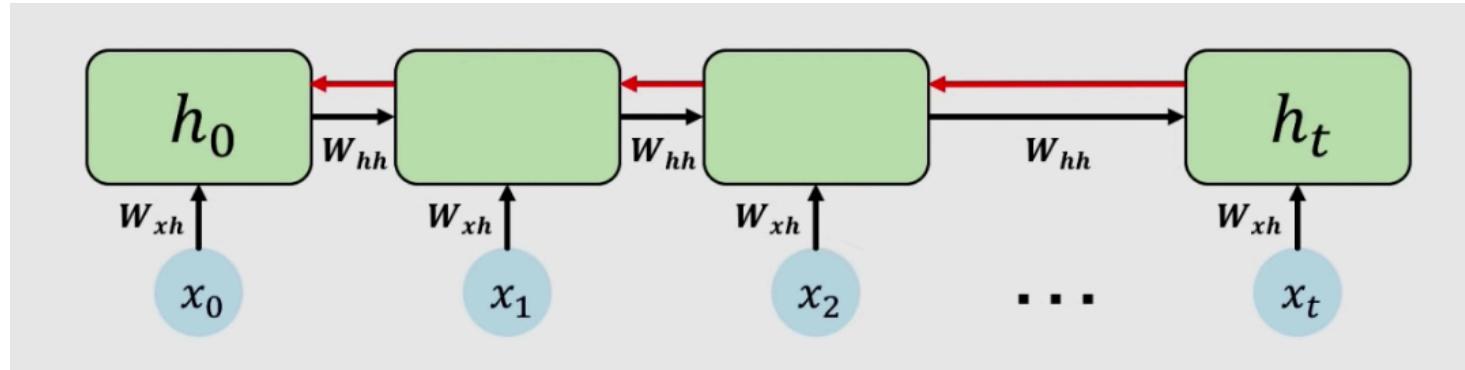
**3. Embedding:**  
Index to fixed-sized vector

# BACKPROPAGATION THROUGH TIME (BPTT)





BPTT implies a large amount of weight multiplications: if we want to take into account long term memory, networks become quickly very deep and so are subjected to vanishing and exploding gradients problems (see previous lecture)

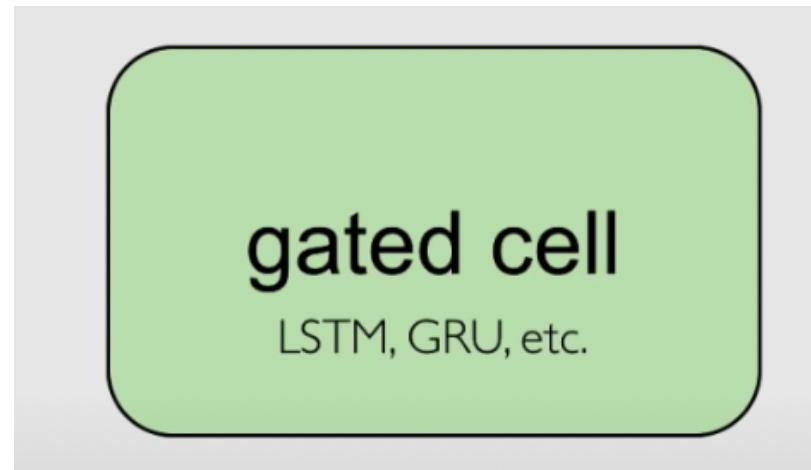


BPTT implies a large amount of weight multiplications: if we want to take into account long term memory, networks become quickly very deep and so are subjected to vanishing and exploding gradients problems (see previous lecture)

ONE CAN USE THE SAME TRICKS THAT WE DISCUSSED  
FOR DEEP CNNs (WEIGHT INITIALISATION, RELU  
ACTIVATION FUNCTION ...)

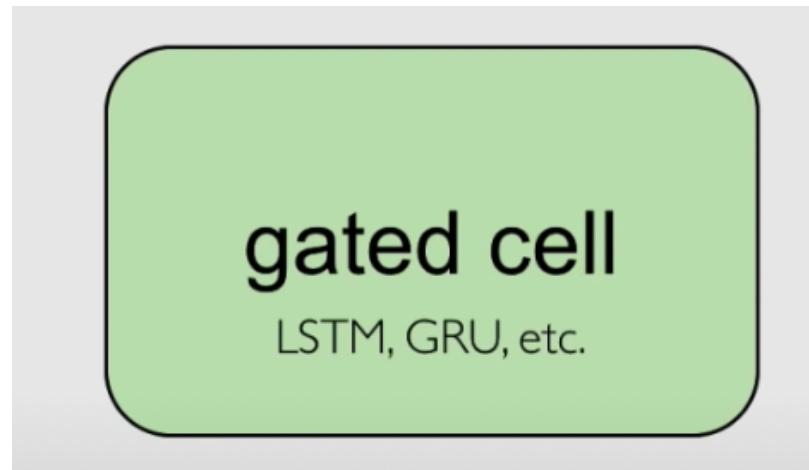
## **ANOTHER SPECIFIC SOLUTION: GATED CELLS**

Use a more complex recurrent unit with gates to control what information is passed through...



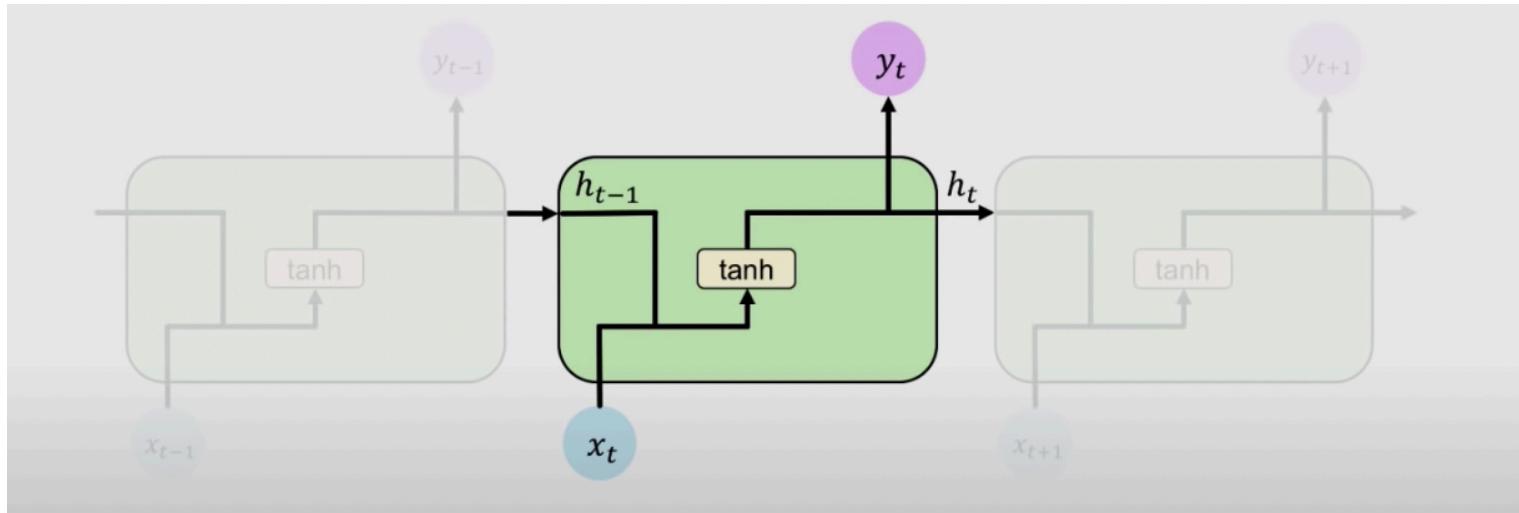
## ANOTHER SPECIFIC SOLUTION: GATED CELLS

Use a more complex recurrent unit with gates to control what information is passed through...

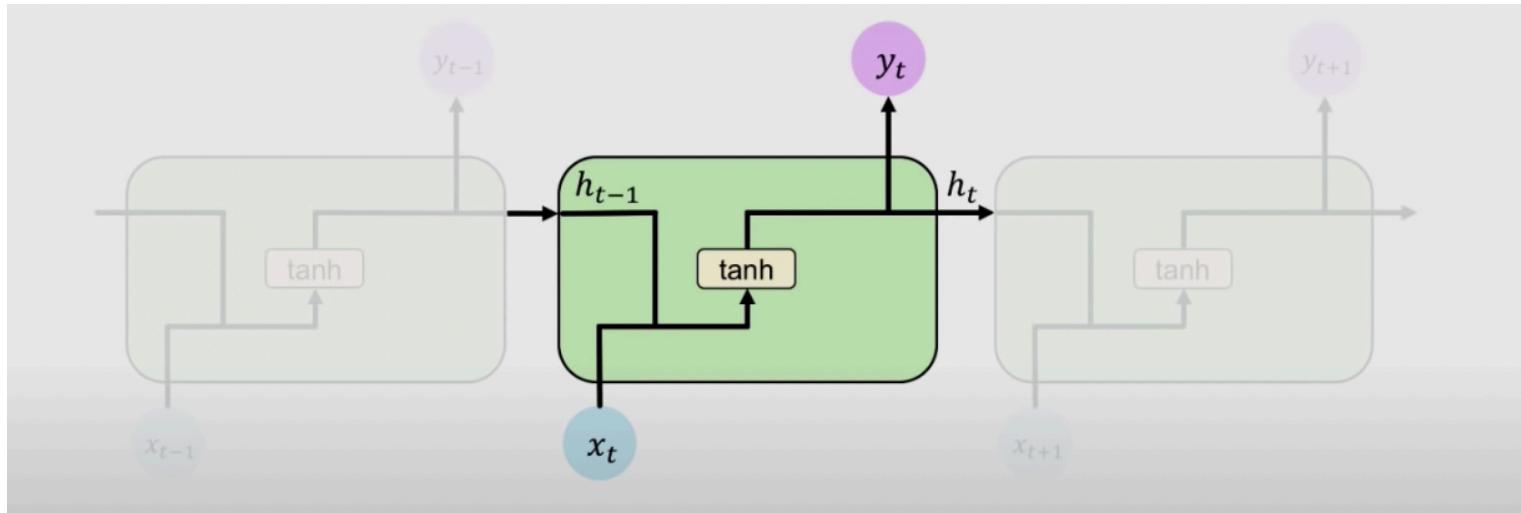


Long Short Term Memory (LSTMs) networks are an example of  
this

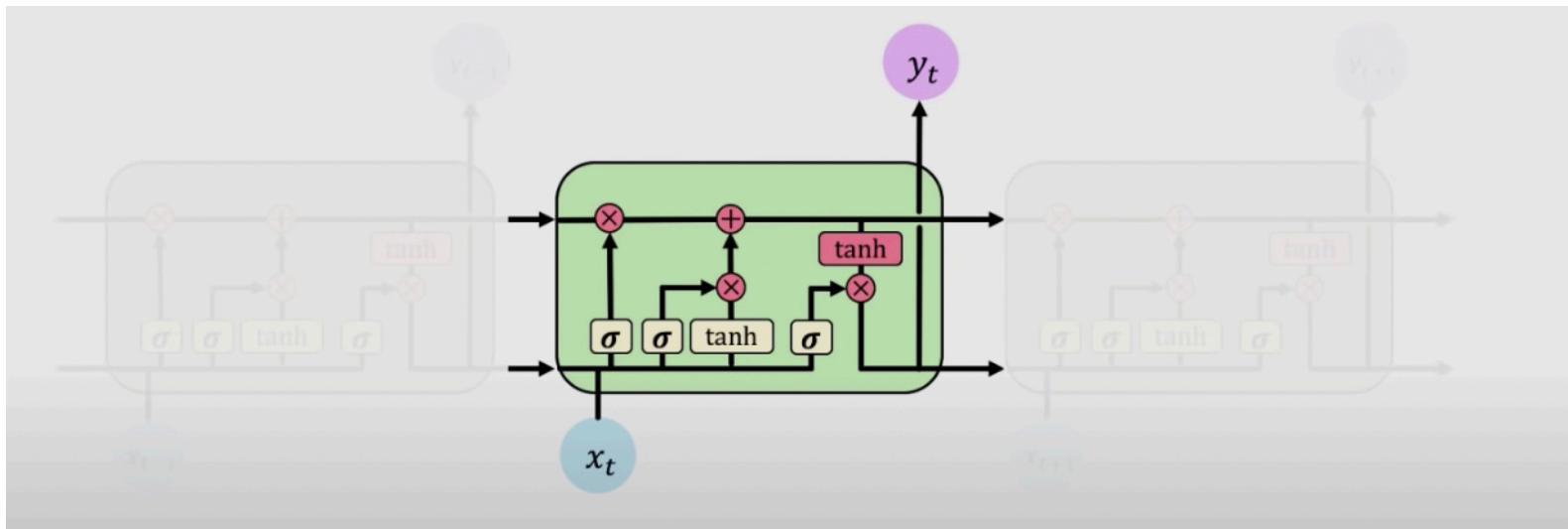
# Standard RNN



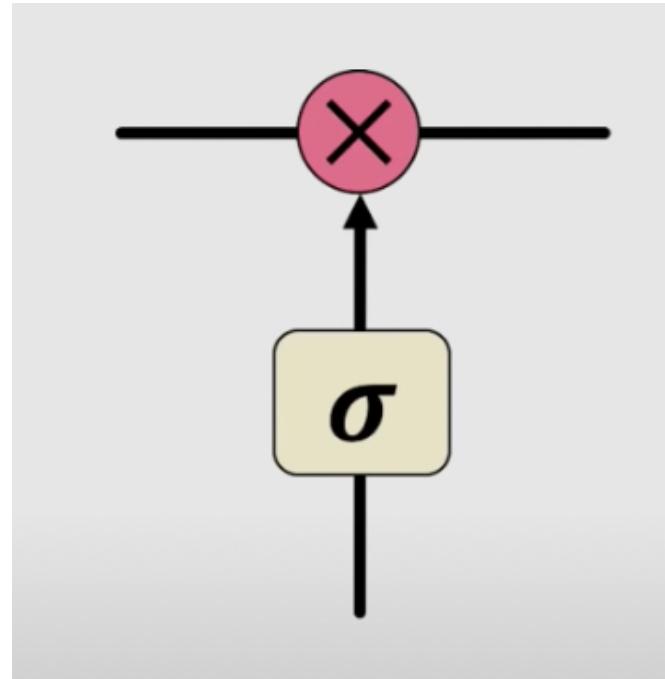
# Standard RNN



# LSTM unit

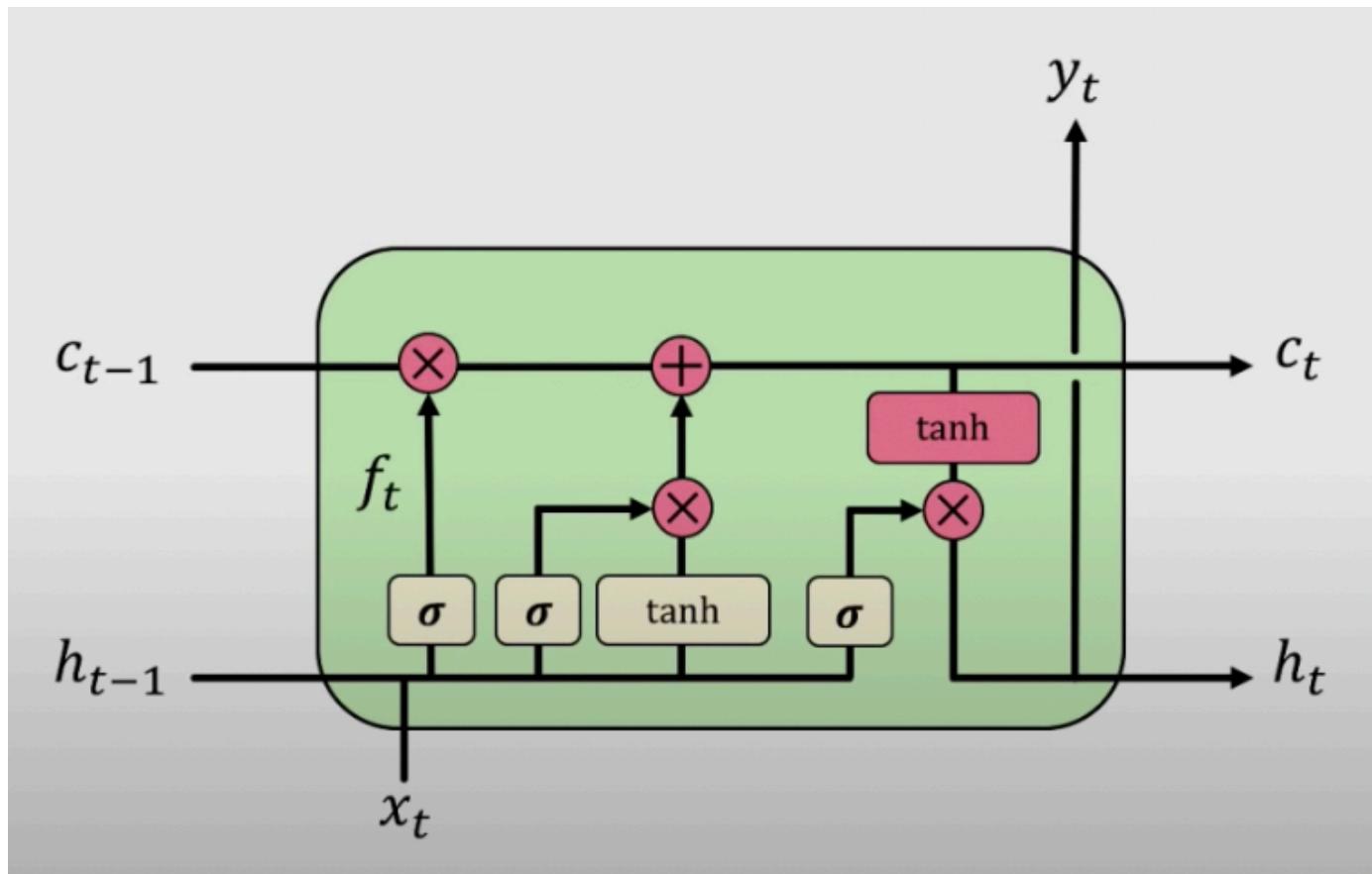


The key building block are the so-called gates



INFORMATION IS PASSED OR NOT WITH A  
COMBINATION OF A SIGMOID NN AND A  
POINTWISE MULTIPLICATION

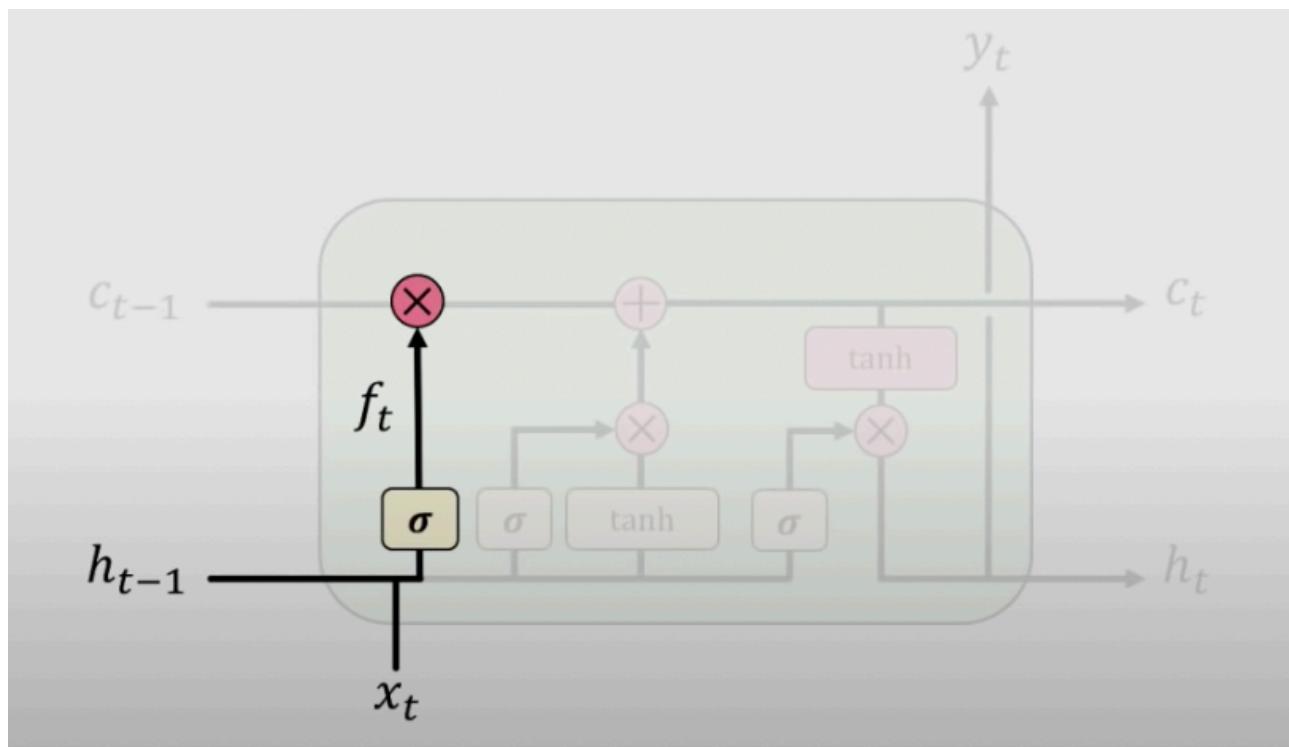
# Long Short Term Memory (LSTMs)



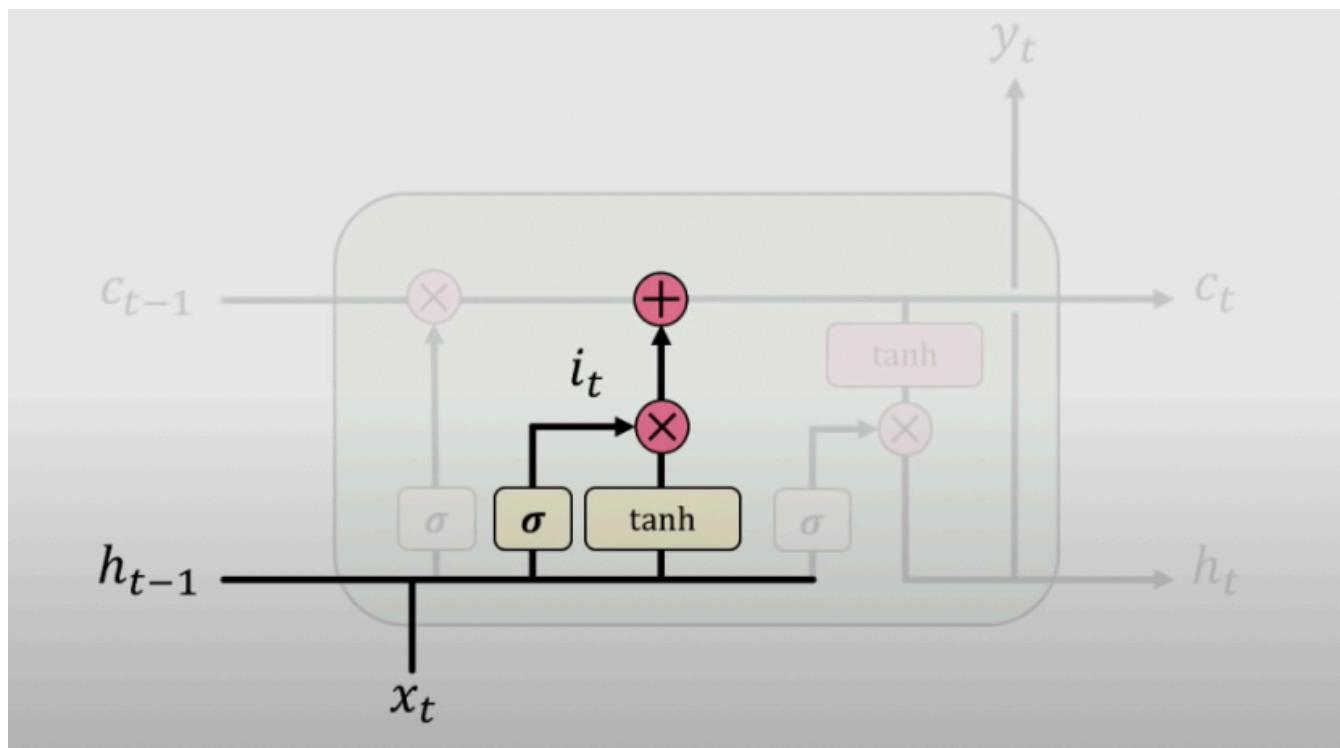
4 main steps:

**1-Forget 2-Store 3-Update 4-Output**

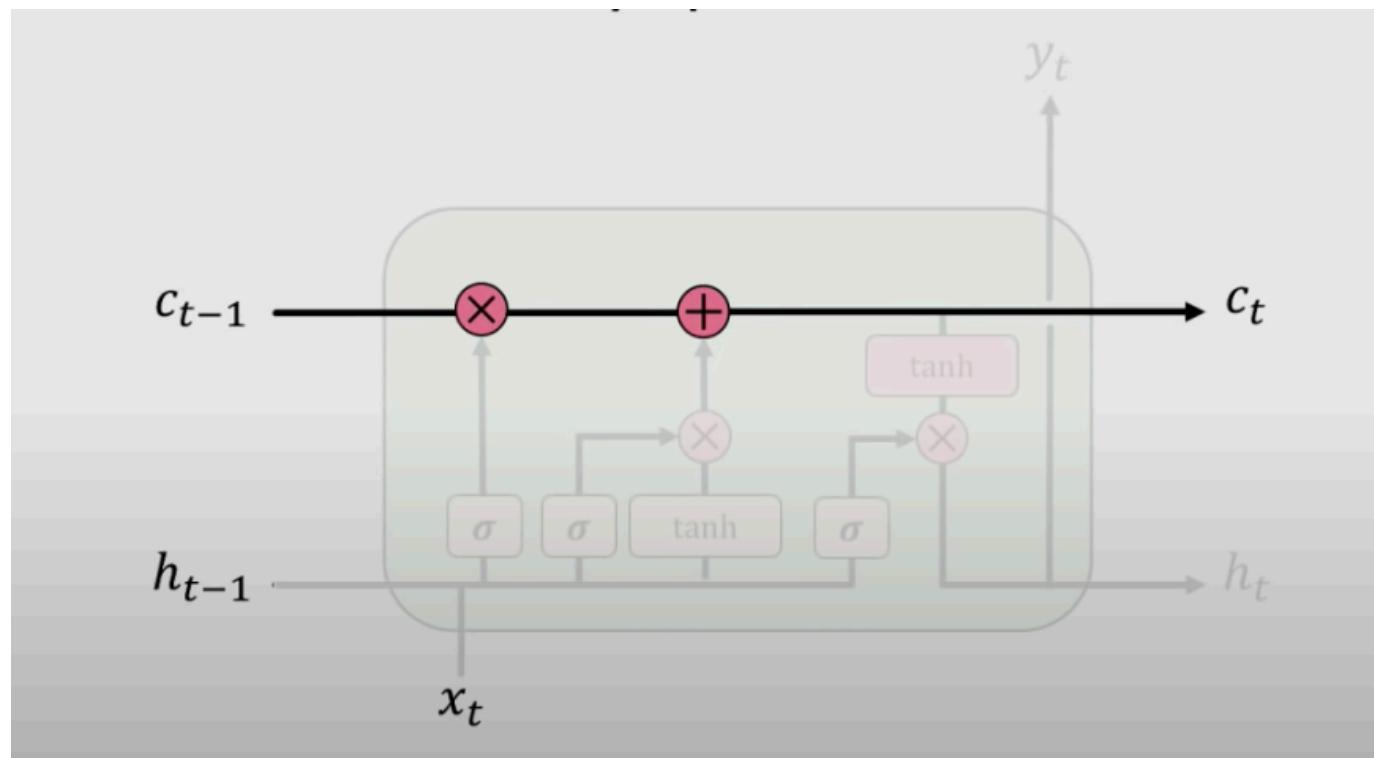
# STEP 1: FORGET



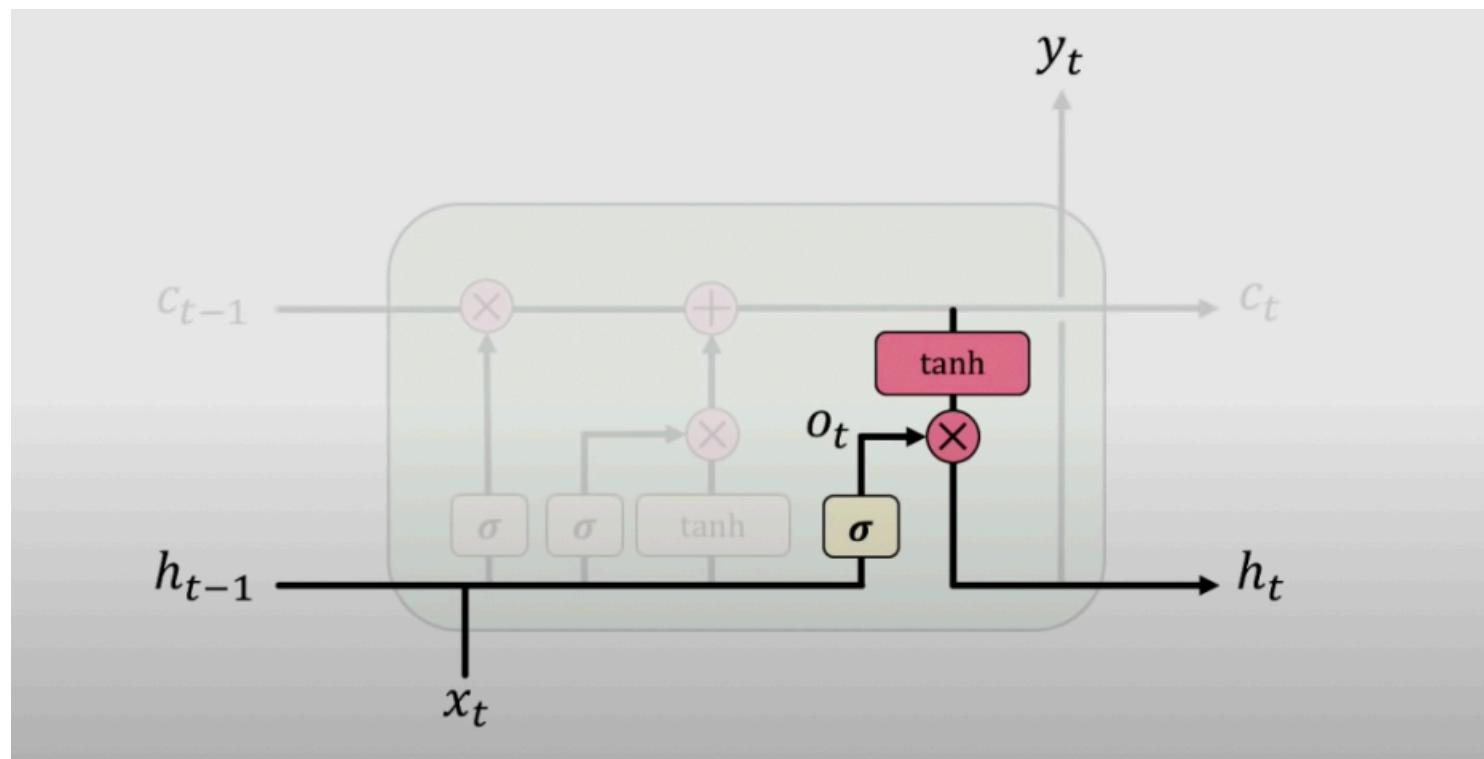
## **STEP 2: STORE**



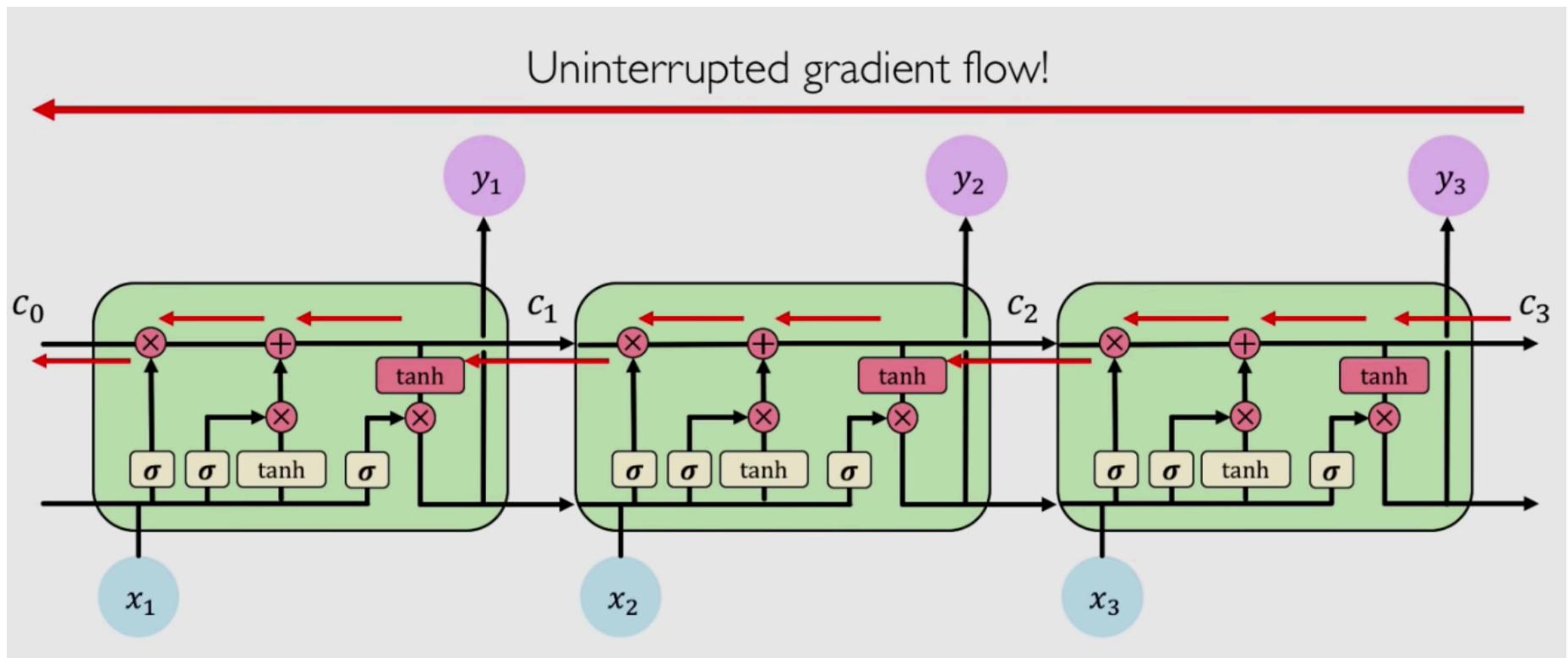
## STEP 3: UPDATE



## STEP 4: OUTPUT



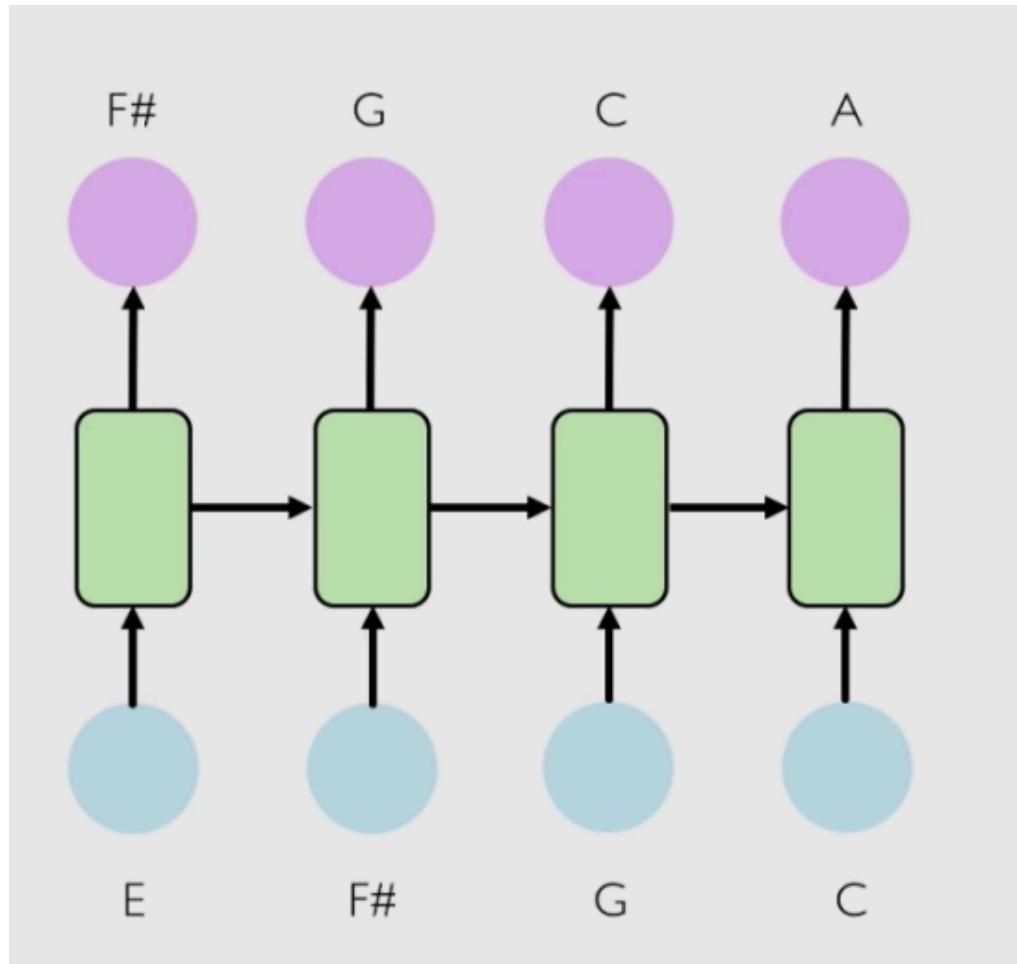
# LSTMs help mitigating the vanishing gradient problem



## **LSTMs KEY CONCEPTS**

- 1. Maintain a separate cell state from what is outputted**
- 2. Use gates to control the flow of information**
  - Forget gates get rid of irrelevant information
  - Store relevant information from current input
  - Selectively update cell state
  - Output gate returns a filtered version of the cell state
- 3. Backpropagation through time with uninterrupted gradient flow**

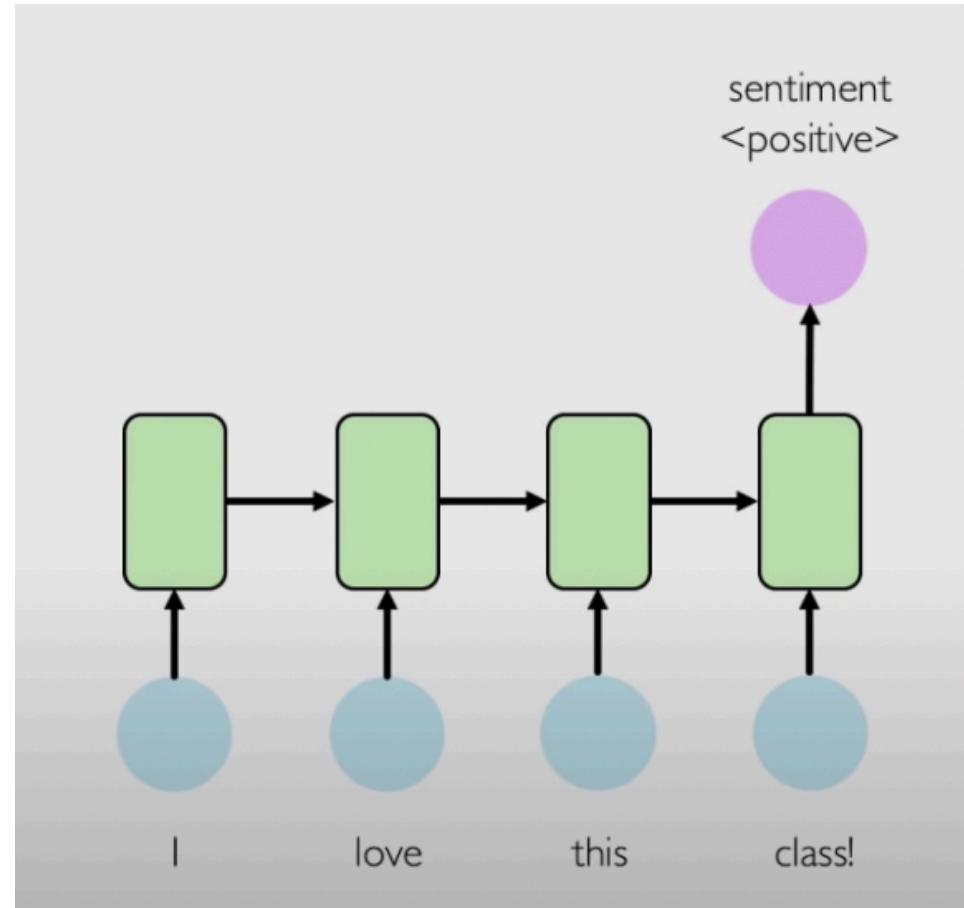
# SOME APPLICATIONS (OUTSIDE ASTRONOMY)



Automatic music  
generation

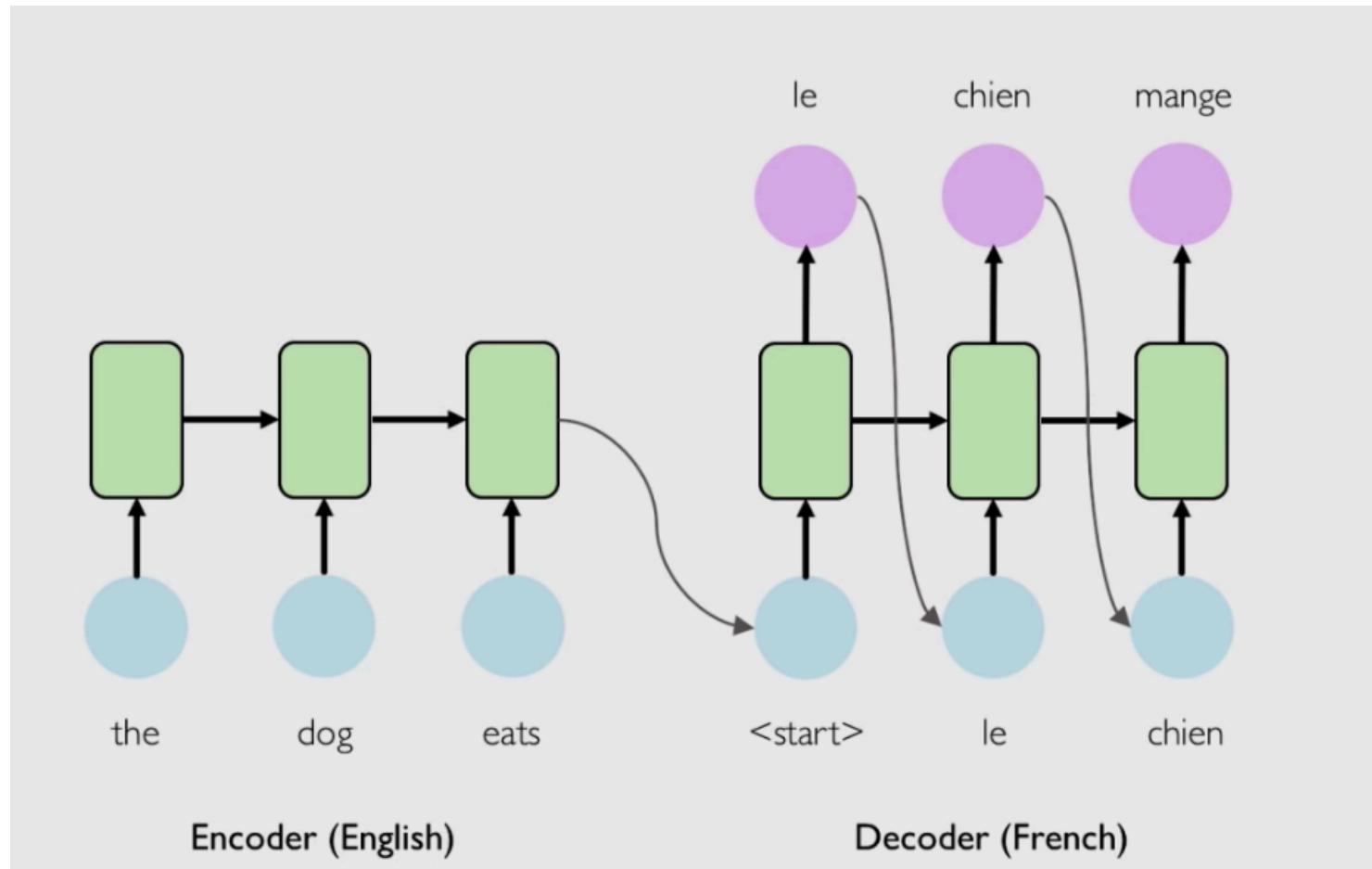


# SOME APPLICATIONS (OUTSIDE ASTRONOMY)



SENTIMENT  
CLASSIFICATION

# MACHINE TRANSLATION



# Intuition Behind Self-Attention

Attending to the most important parts of an input.



# Intuition Behind Self-Attention

Attending to the most important parts of an input.



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

# Intuition Behind Self-Attention

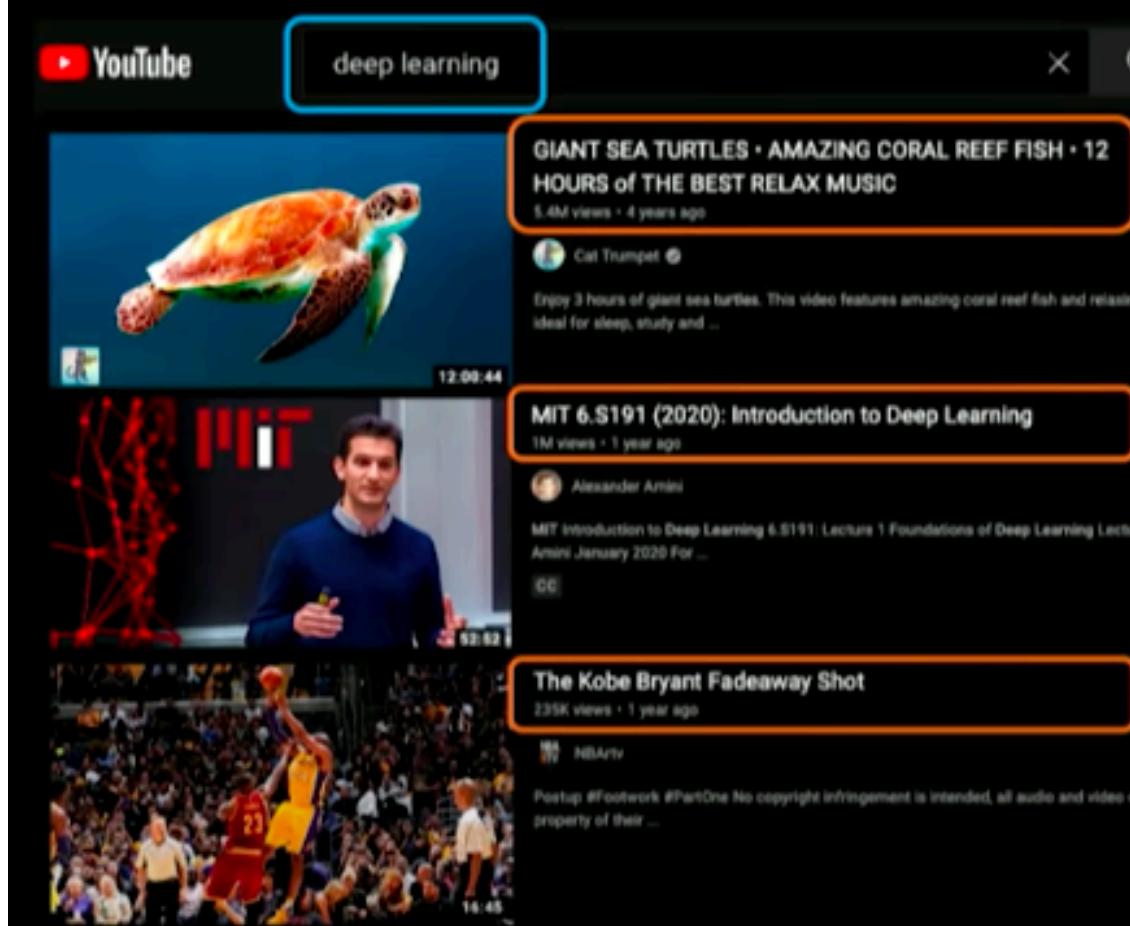
Attending to the most important parts of an input.



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

Similar to a  
search problem!

# Understanding Attention with Search



Query (Q)

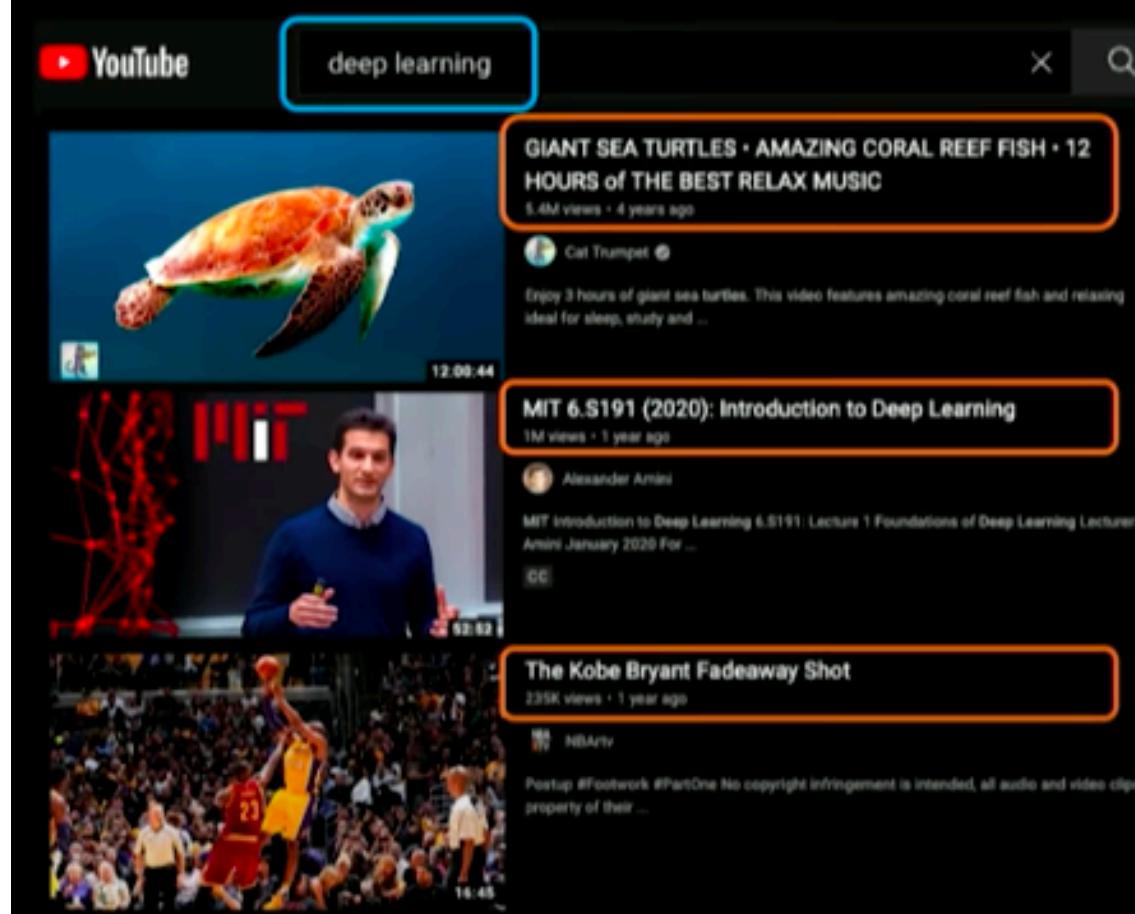
Key ( $K_1$ )

Key ( $K_2$ )

Key ( $K_3$ )

I. **Compute attention mask:** how similar is each key to the desired query?

# Understanding Attention with Search



Query (Q)

Key ( $K_1$ )

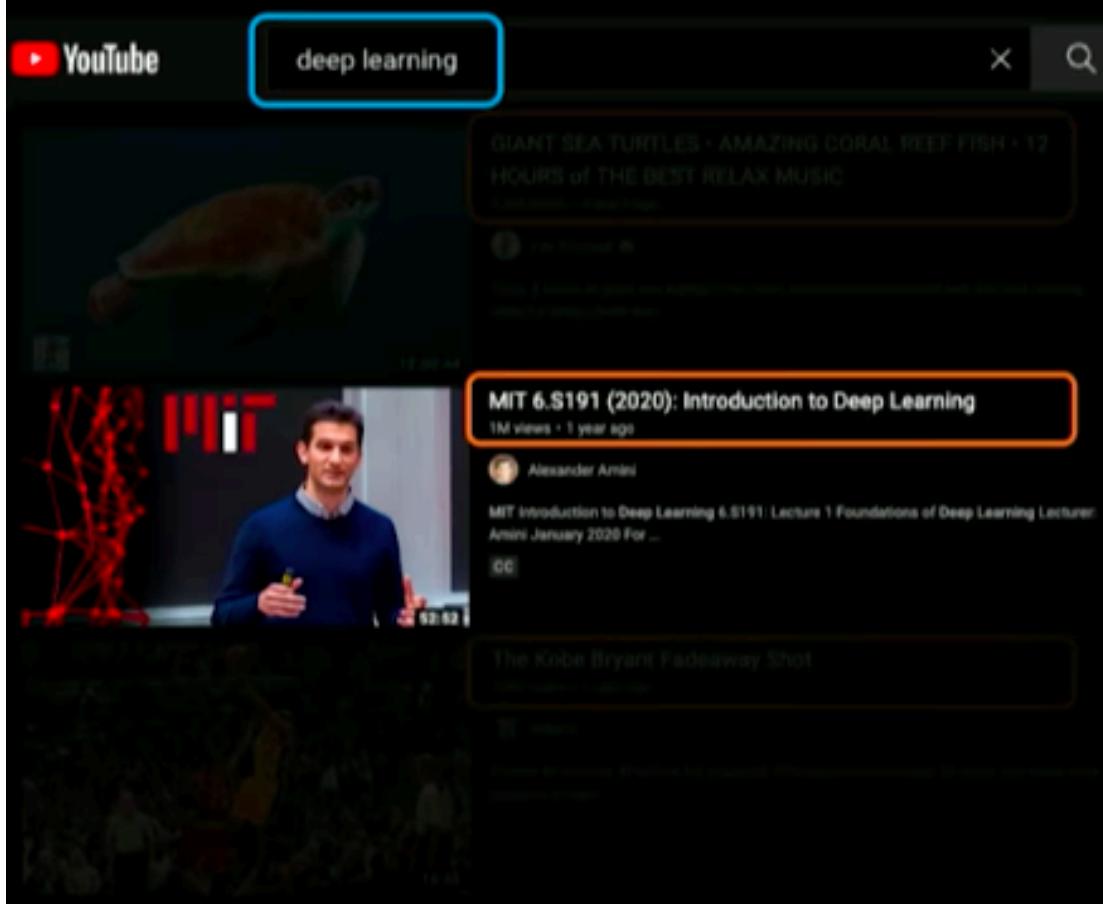
How similar is the key to the query?

Key ( $K_2$ )

Key ( $K_3$ )

I. **Compute attention mask:** how similar is each key to the desired query?

# Understanding Attention with Search



Query (Q)

Key ( $K_1$ )

Key ( $K_2$ )

Key ( $K_3$ )

How similar is the key to the query?

I. **Compute attention mask:** how similar is each key to the desired query?

# Understanding Attention with Search

YouTube deep learning X Query (Q)

GIANT SEA TURTLES • AMAZING CORAL REEF FISH • 12 HOURS of THE BEST RELAX MUSIC

MIT 6.S191 (2020): Introduction to Deep Learning

Alexander Amini

MIT Introduction to Deep Learning 6.S191: Lecture 1 Foundations of Deep Learning Lecture Amini January 2020 For ...

CC

MIT 6.S191 (2020): Introduction to Deep Learning

Key ( $K_1$ )

Key ( $K_2$ )

Value (V)

Key ( $K_3$ )

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

# Learning Self-Attention with Neural Networks

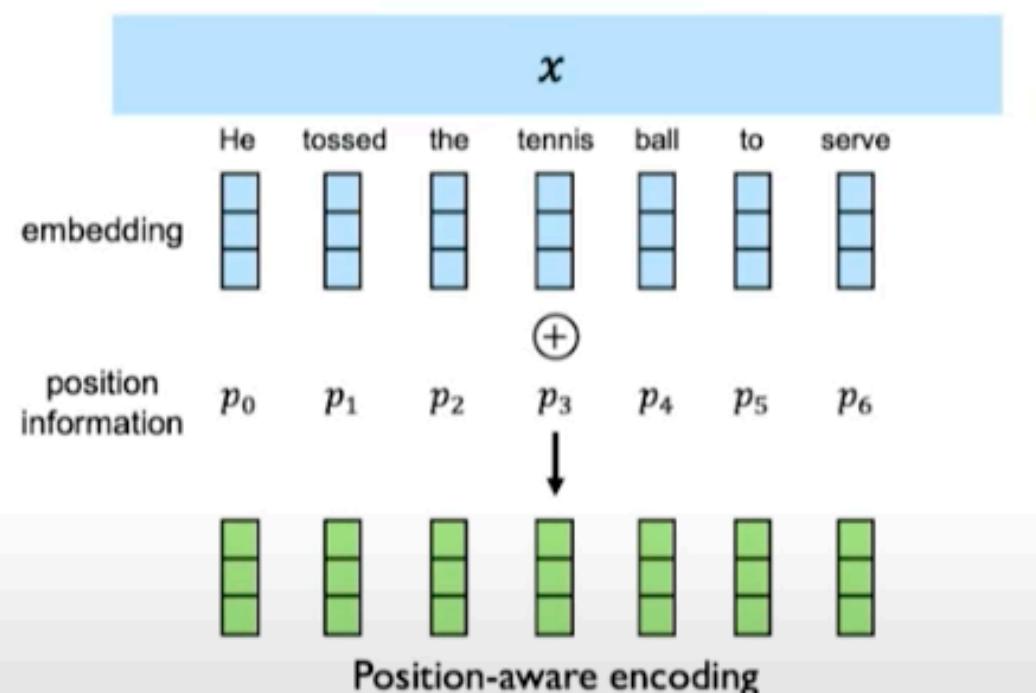
Goal: identify and attend to most important features in input.

I. Encode position information

A. Learn query, key, value functions

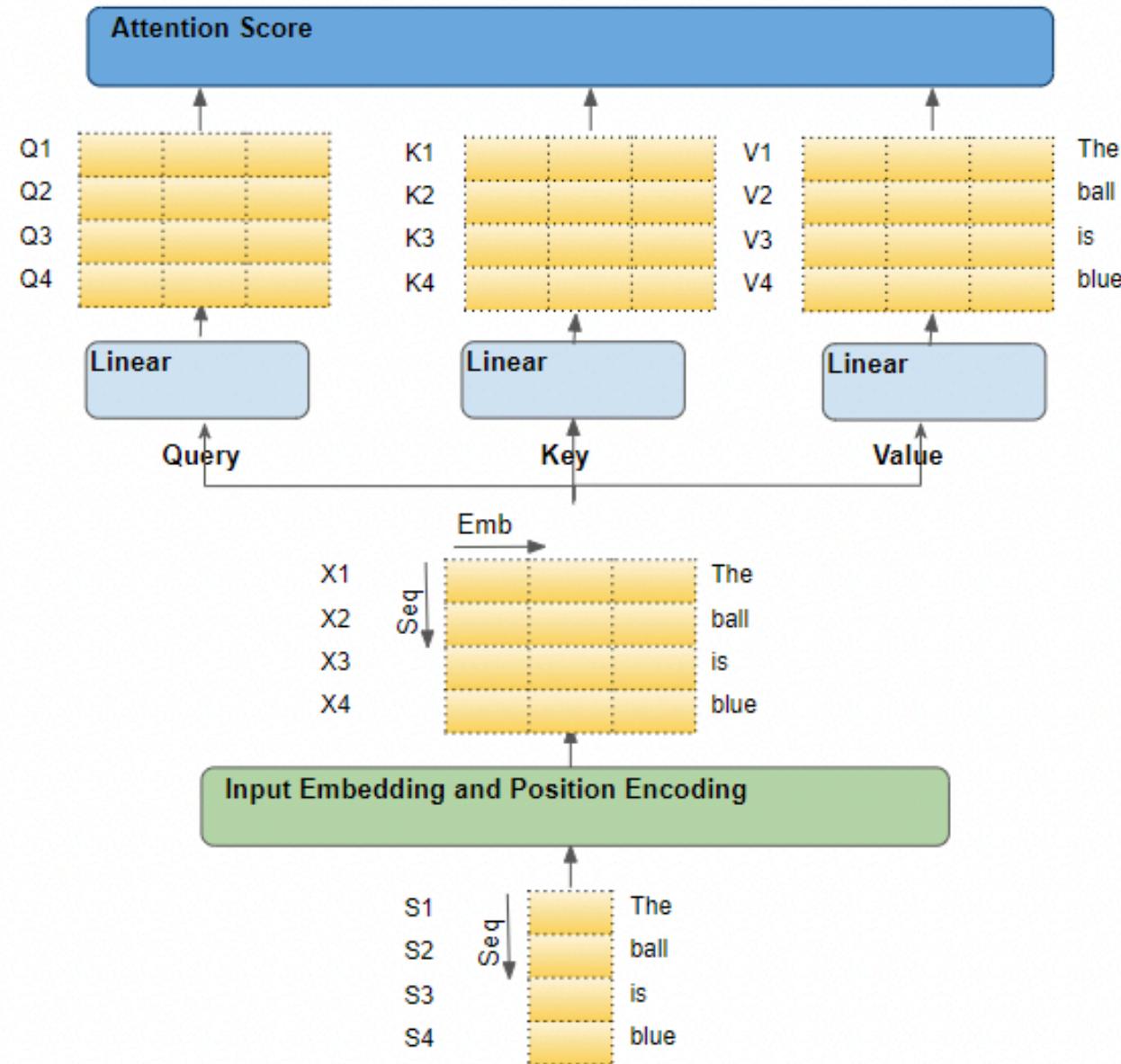
B. Compute attention weighting

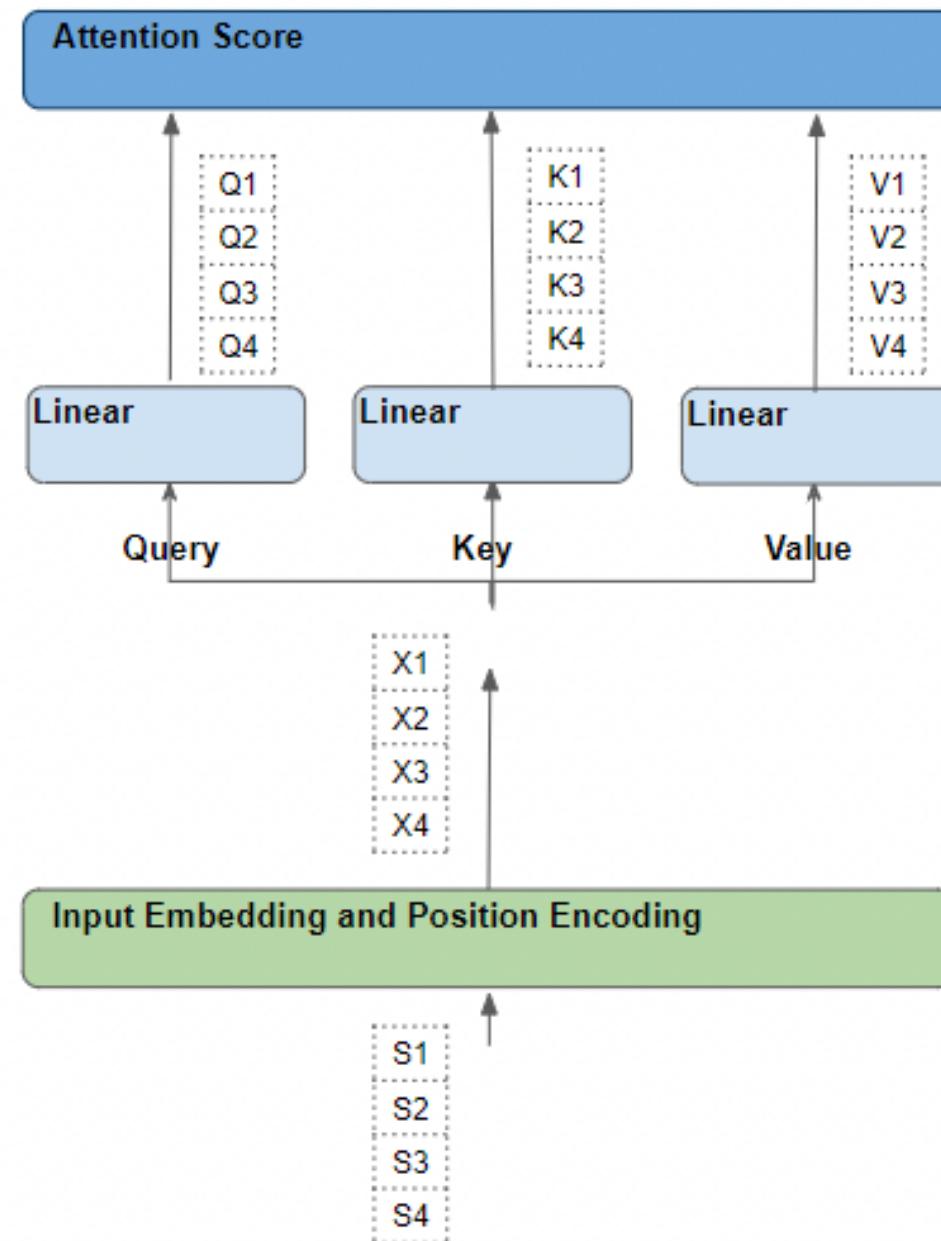
C. Attend features with high attention

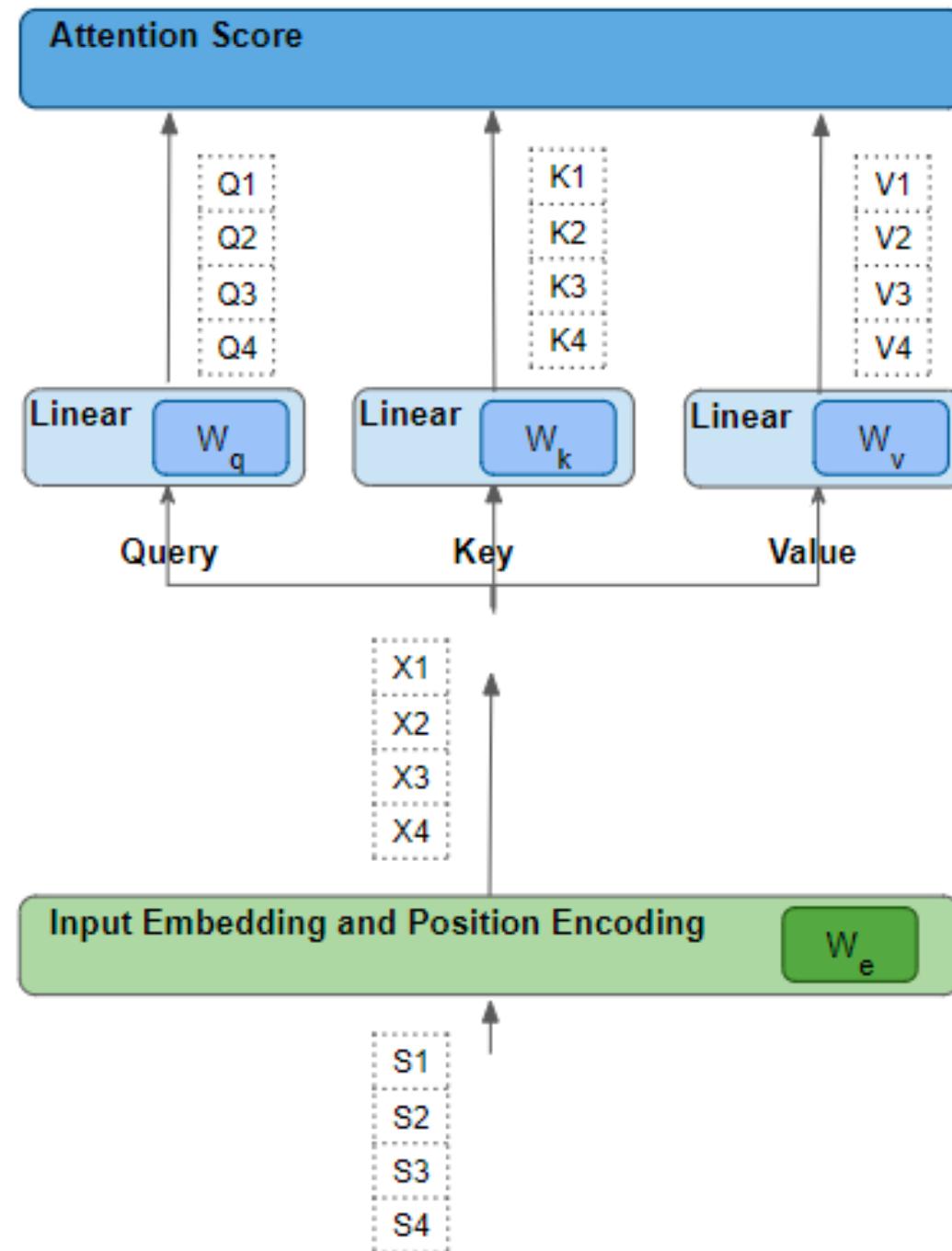


Data is fed in all at once! Need to encode position information to understand order.

# The Attention Module







|    |    |    |    |    |      |      |      |      |
|----|----|----|----|----|------|------|------|------|
| Q1 |    |    |    |    | Q1K1 | Q1K2 | Q1K3 | Q1K4 |
| Q2 |    |    |    |    | Q2K1 | Q2K2 | Q2K3 | Q2K4 |
| Q3 | K1 | K2 | K3 | K4 | Q3K1 | Q3K2 | Q3K3 | Q3K4 |
| Q4 |    |    |    |    | Q4K1 | Q4K2 | Q4K3 | Q4K4 |

|    |    |    |    |    |      |      |      |      |
|----|----|----|----|----|------|------|------|------|
| Q1 |    |    |    |    | Q1K1 | Q1K2 | Q1K3 | Q1K4 |
| Q2 |    |    |    |    | Q2K1 | Q2K2 | Q2K3 | Q2K4 |
| Q3 | K1 | K2 | K3 | K4 | Q3K1 | Q3K2 | Q3K3 | Q3K4 |
| Q4 |    |    |    |    | Q4K1 | Q4K2 | Q4K3 | Q4K4 |

|      |      |      |      |
|------|------|------|------|
| Q1K1 | Q1K2 | Q1K3 | Q1K4 |
| Q2K1 | Q2K2 | Q2K3 | Q2K4 |
| Q3K1 | Q3K2 | Q3K3 | Q3K4 |
| Q4K1 | Q4K2 | Q4K3 | Q4K4 |

|    |
|----|
| V1 |
| V2 |
| V3 |
| V4 |

=

|                                   |
|-----------------------------------|
| Q1K1V1 + Q1K2V2 + Q1K3V3 + Q1K4V4 |
| Q2K1V1 + Q2K2V2 + Q2K3V3 + Q2K4V4 |
| Q3K1V1 + Q3K2V2 + Q3K3V3 + Q3K4V4 |
| Q4K1V1 + Q4K2V2 + Q4K3V3 + Q4K4V4 |

=

|    |
|----|
| Z1 |
| Z2 |
| Z3 |
| Z4 |

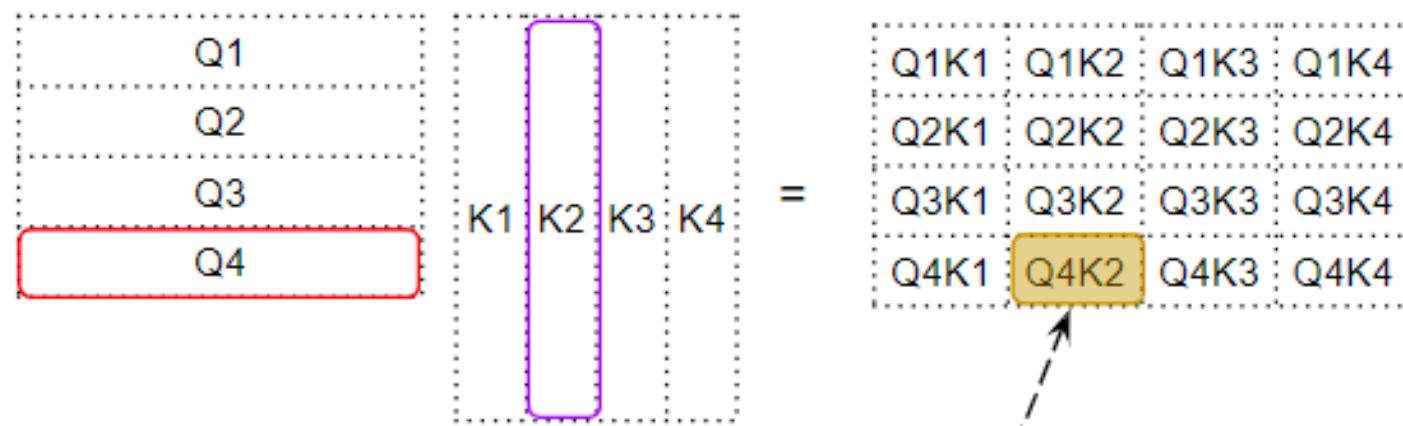
*Fourth word Score*

*Fourth Query word \* first Key word*

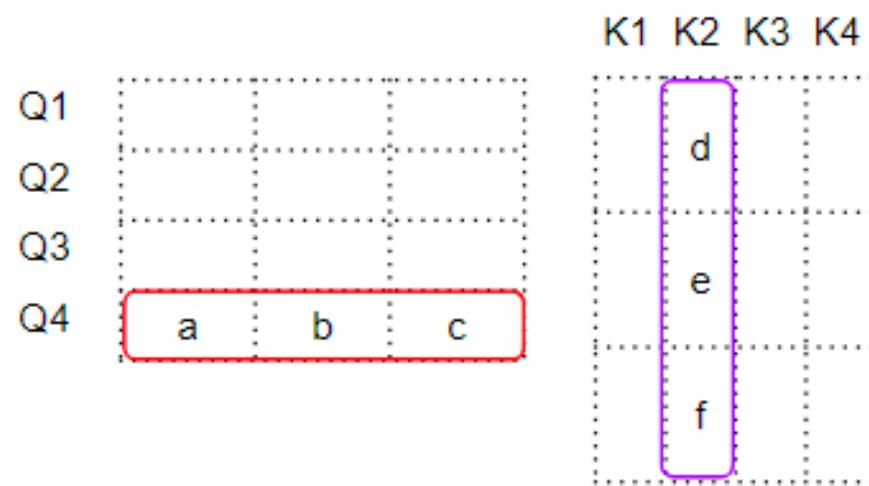
$$Z_4 = (Q_4 K_1) V_1 + (Q_4 K_2) V_2 + (Q_4 K_3) V_3 + (Q_4 K_4) V_4$$

*Fourth Query word \* second Key word*

$$Z = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



$$Q4K2 = a*d + b*e + c*f$$



# Learning Self-Attention with Neural Networks

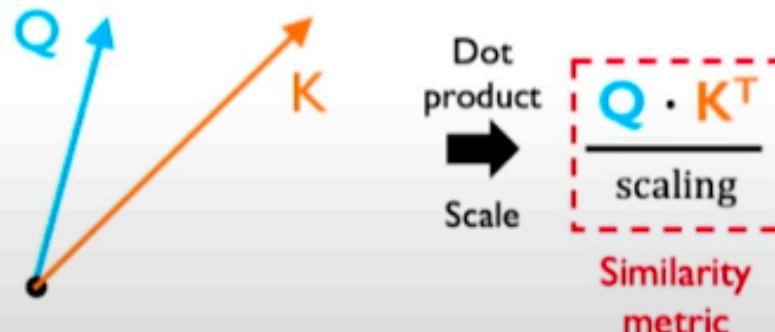
Goal: identify and attend to most important features in input.

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

Identify features with high attention

**Attention score:** compute pairwise similarity between each **query** and **key**

How to compute similarity between two sets of features?



Also known as the "cosine similarity"

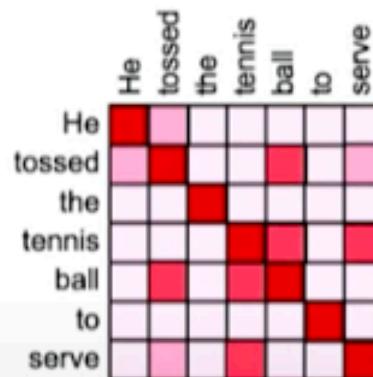
# Learning Self-Attention with Neural Networks

Goal: identify and attend to most important features in input.

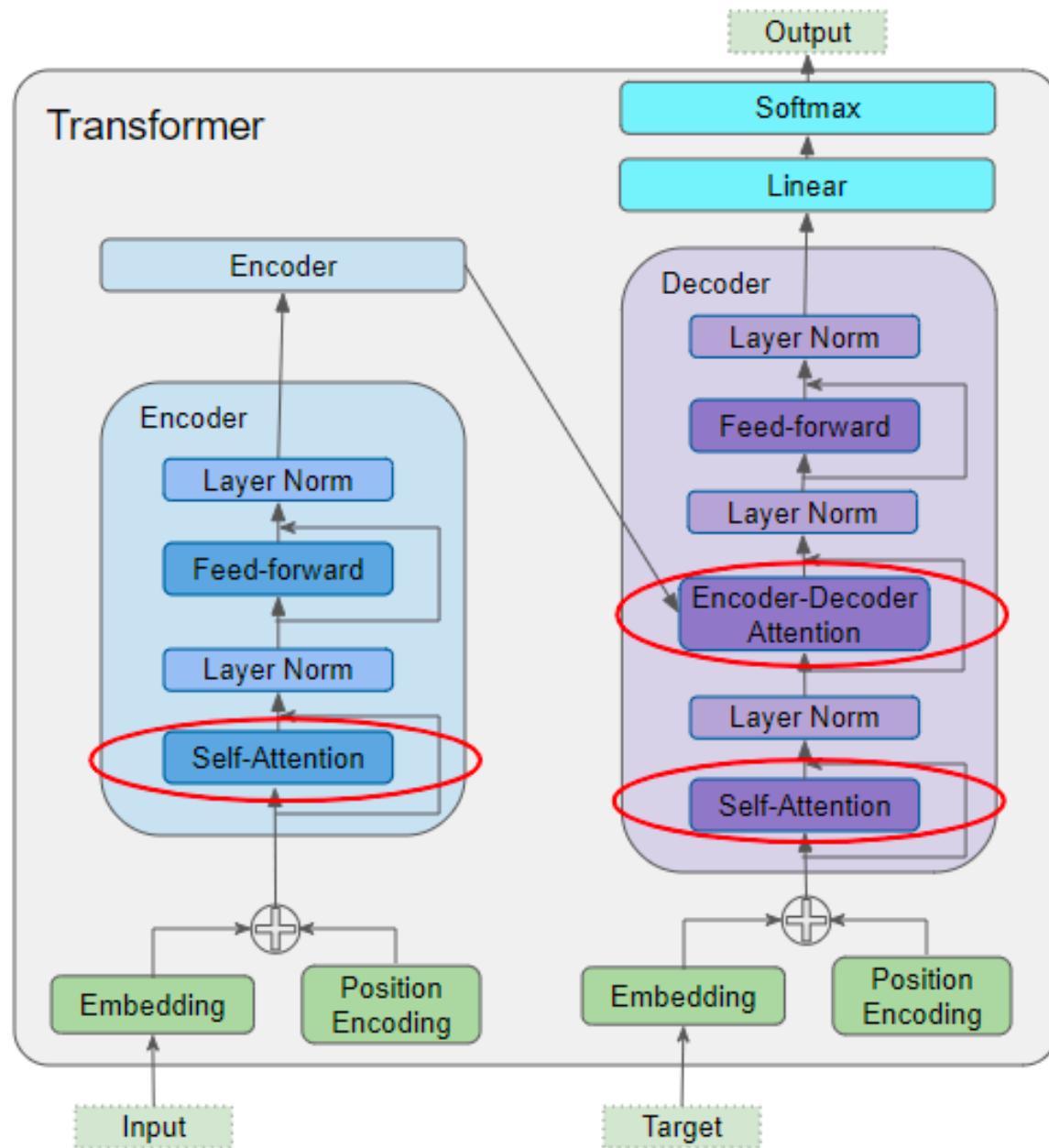
1. Encode **position** information
2. Extract **query, key, value** for search
3. Compute **attention weighting**

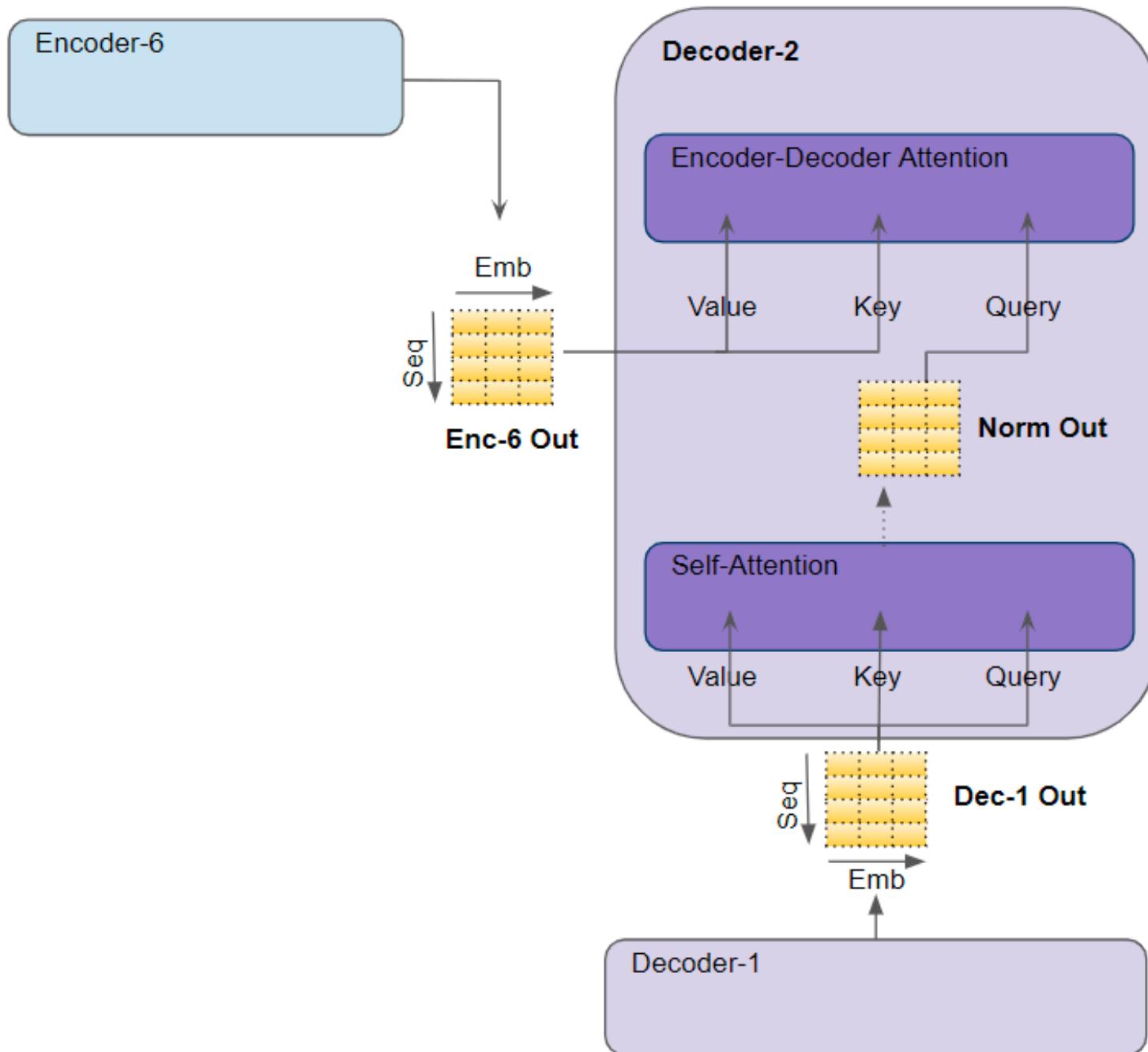
1. Identify features with high attention

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



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



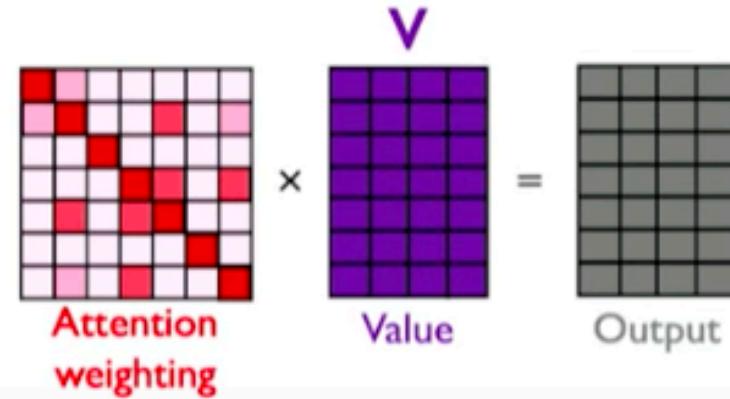


# Learning Self-Attention with Neural Networks

Goal: identify and attend to most important features in input.

1. Encode 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



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