ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

From Transformer to Mamba: May the Princess be Saved






Quan Thanh Tho
qttho@hcmut.edu.vn

HCMC, 8/2024

Assoc. Prof. Quan Thanh Tho
Acting Dean
Faculty of Computer Science and Engineering
Ho Chi Minh City University of Technology (HCMUT)
Vietnam National University - Ho Chi Minh City
qttho@hcmut.edu.vn
http://www.cse.hcmut.edu.vn/qttho/

- BEng, HCMUT, Vietnam, 1998
- PhD, NTU, Singapore, 2006
- Research Interests: Artificial Intelligence, Natural Language Processing, intelligent systems, formal methods

Agenda

Part I : Transformer

- Sequence data and sequence models
- Seq2Seq and attention
- Transformer

Part II: Mamba

- To save the princess: The state-based approach
- SMM = RNN + Transformer?
- Mamba model

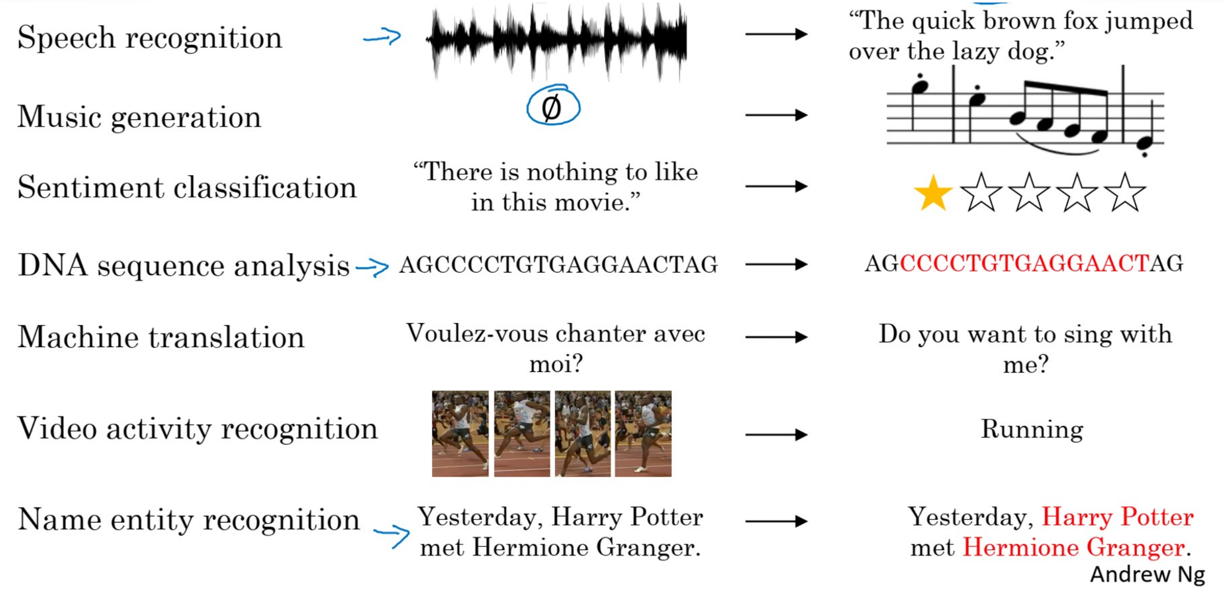
Part I: Transformer

Sequence data and sequence models

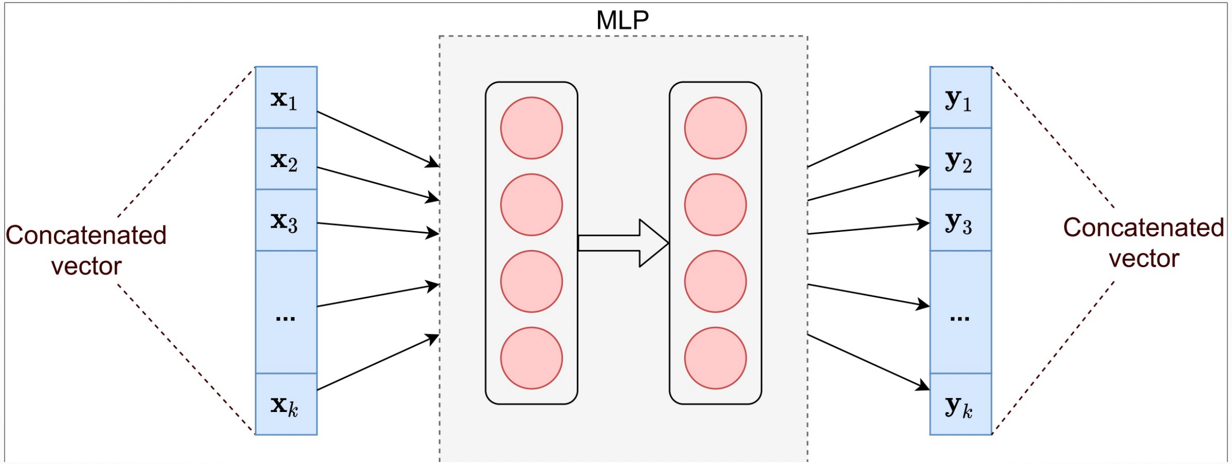
Sequence data

A series of data points whose points reliant on each other

- Length can be varied
- Positions matter

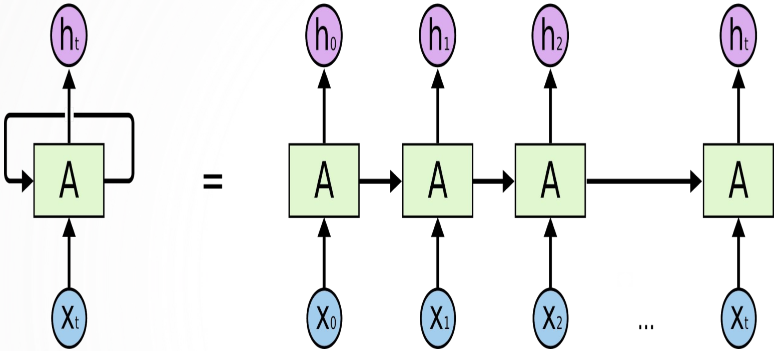


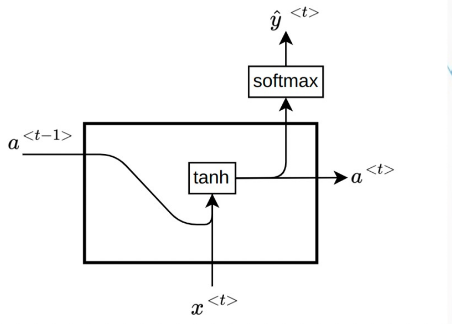
Problem of Standard Networks



- Inputs, outputs can be different lengths in different examples.
- Relations between positions are not well reflected

RNN comes as a rescue





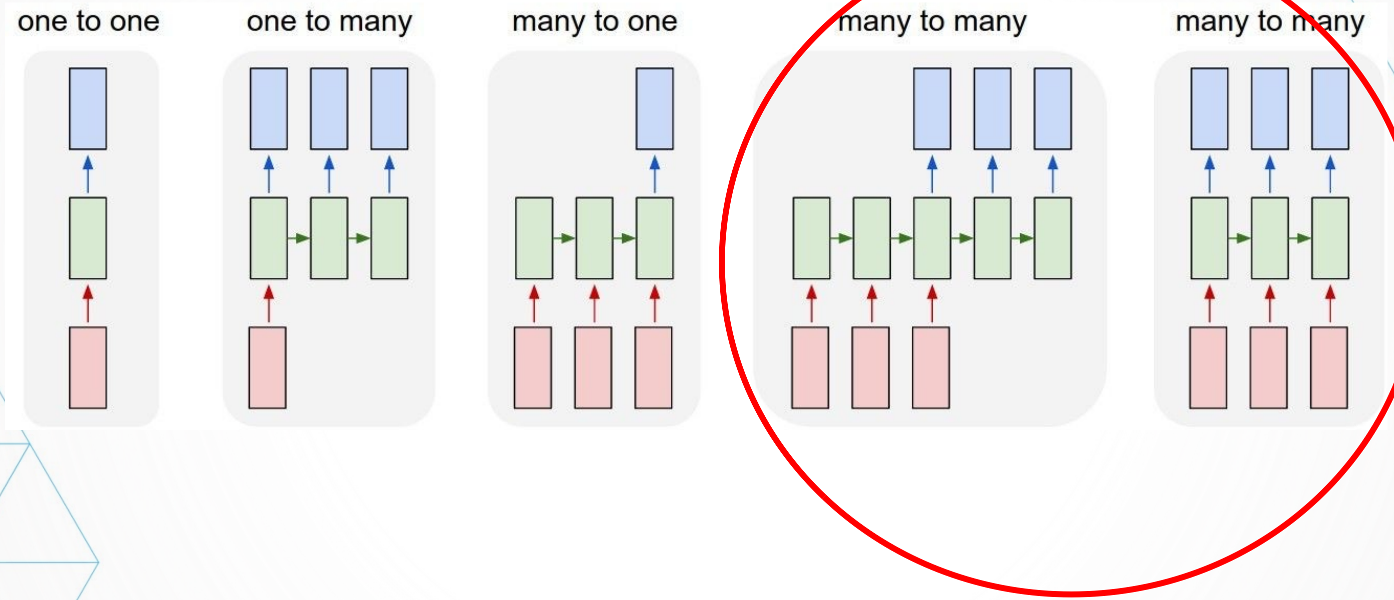
RNN: an architecture tailored for sequence data:

1. Doesn't depend on data length

2. Take advantage of past information

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In D. E. Rumelhart & J. L. Mcclelland (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition, Volume 1: Foundations* (pp. 318–362). MIT Press

RNN Revision

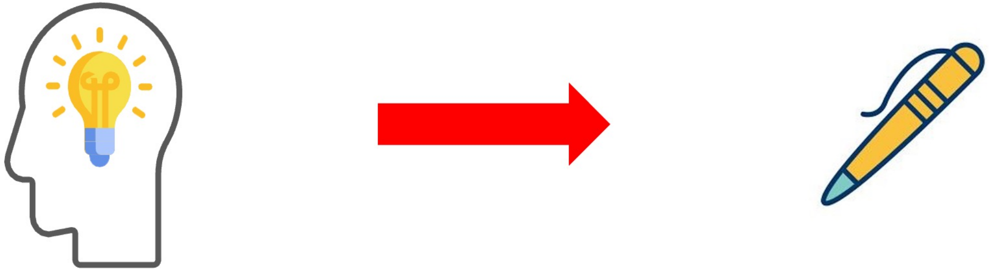


Seq2seq and Attention

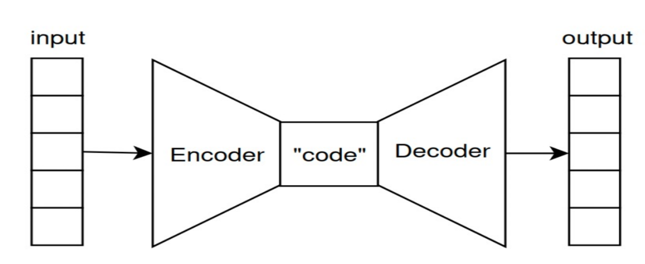
Intuition

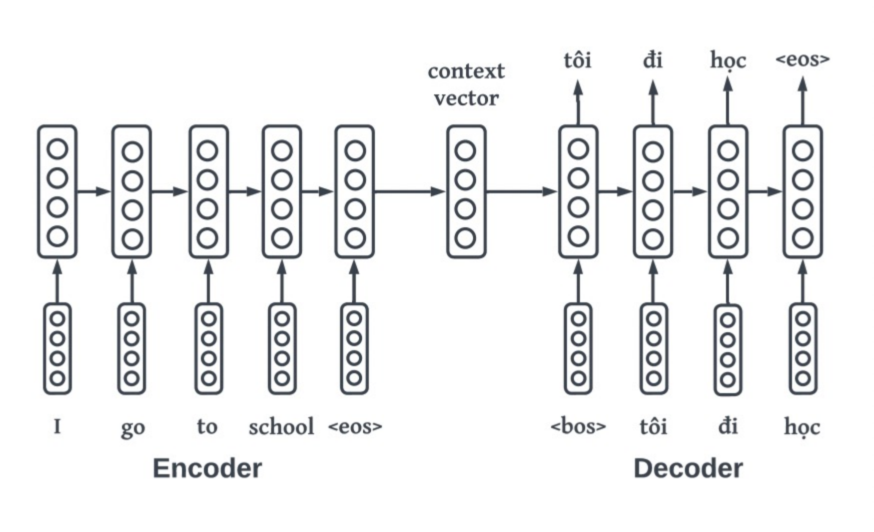
Take machine translation task as an example: human would first read some parts of the text and then start to do the translation

The cat likes to eat pizza *el gato le gusta comer pizza*



Seq2Seq architecture



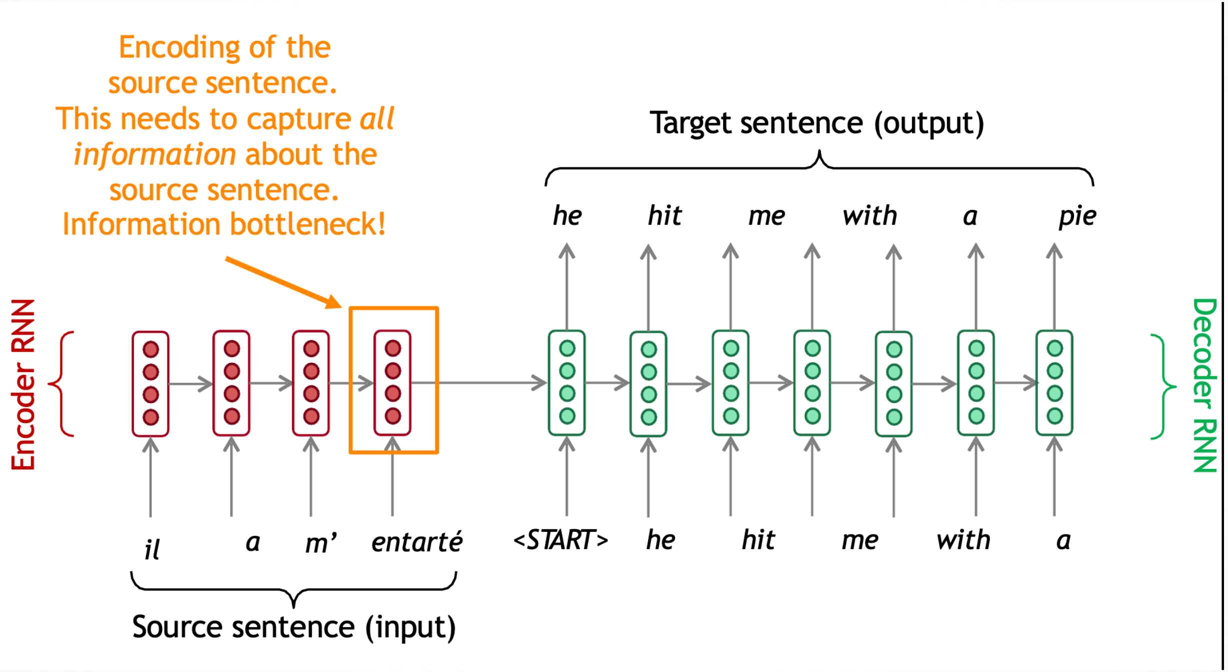


NLP researchers also employ that **idea** into designing a structure dubbed as Sequence-to-Sequence (Seq2Seq), which extends AutoEncoder architecture

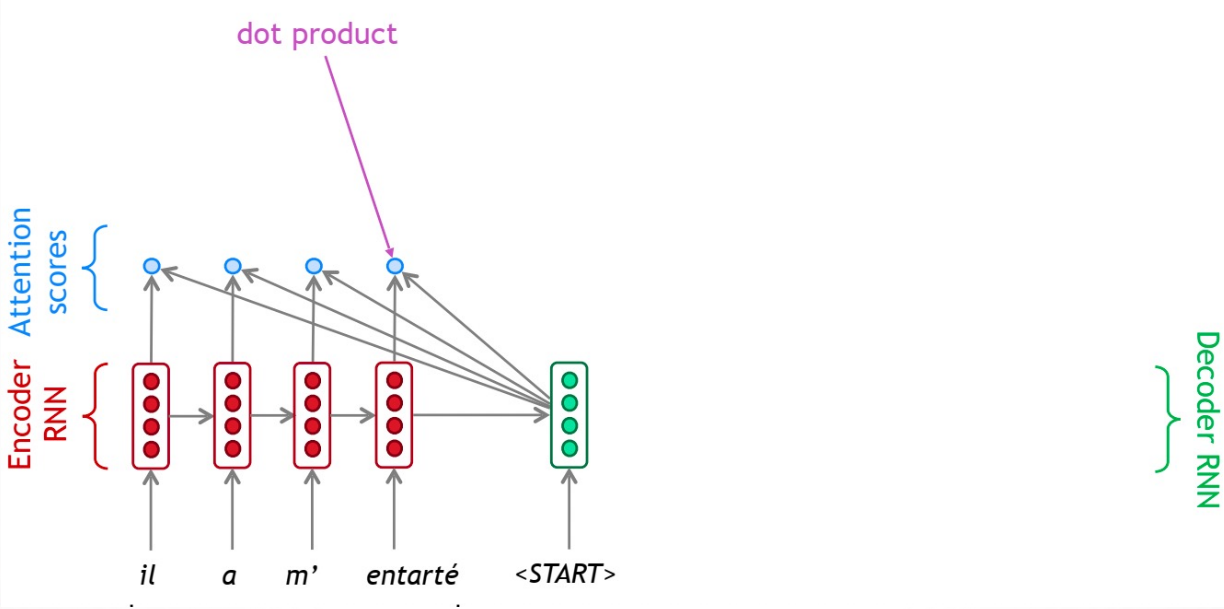
Kiros, R., Zhu, Y., Salakhutdinov, R. R., Zemel, R., Urtasun, R., Torralba, A., & Fidler, S. (2015). Skip-thought vectors. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in neural information processing systems*. Curran Associates, Inc

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27

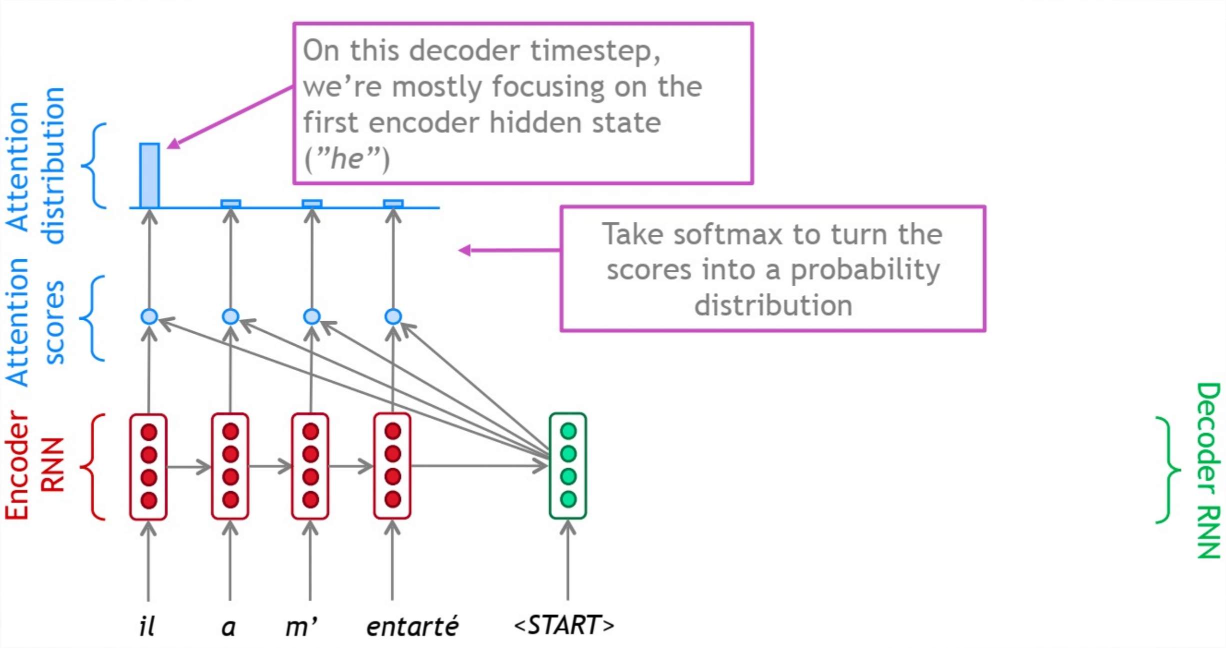
Seq2Seq: The bottle neck problem



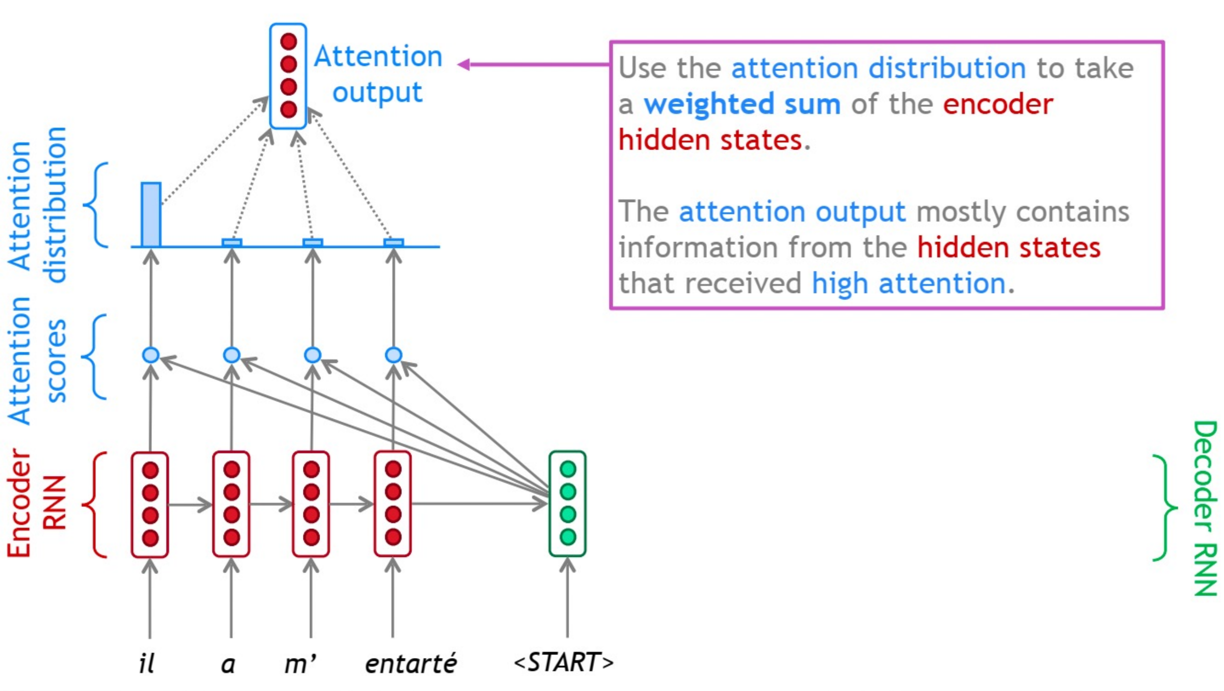
Seq2Seq with attention



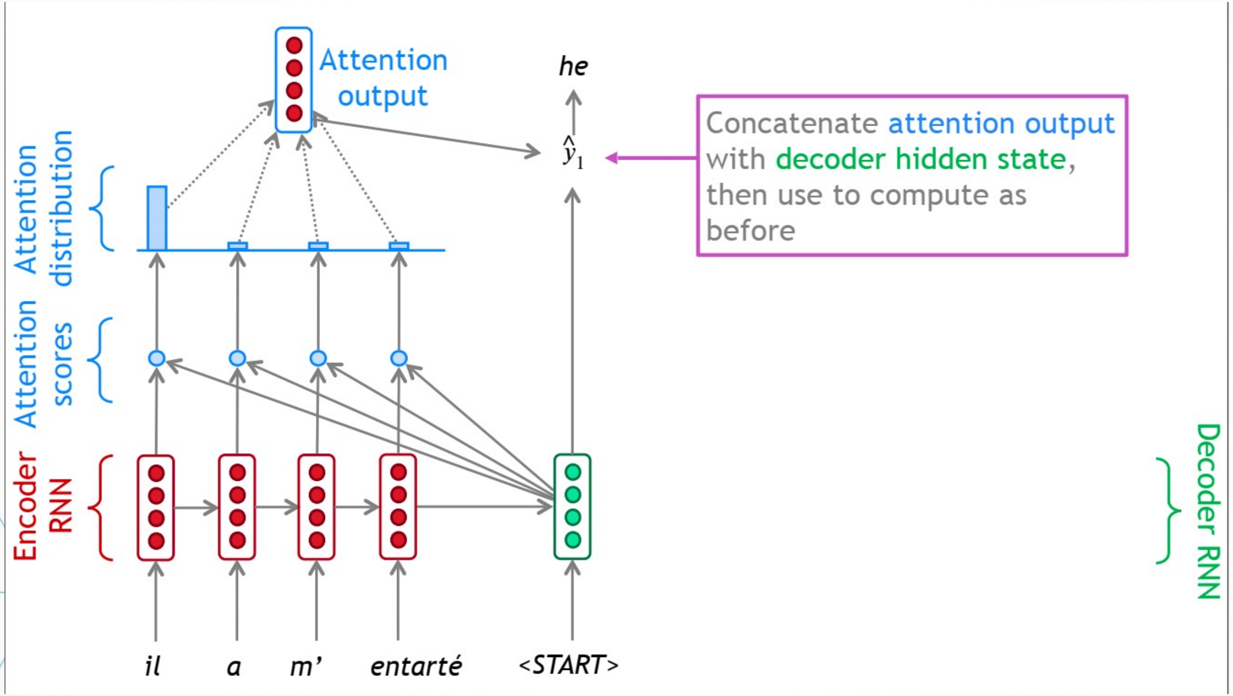
Seq2Seq with attention



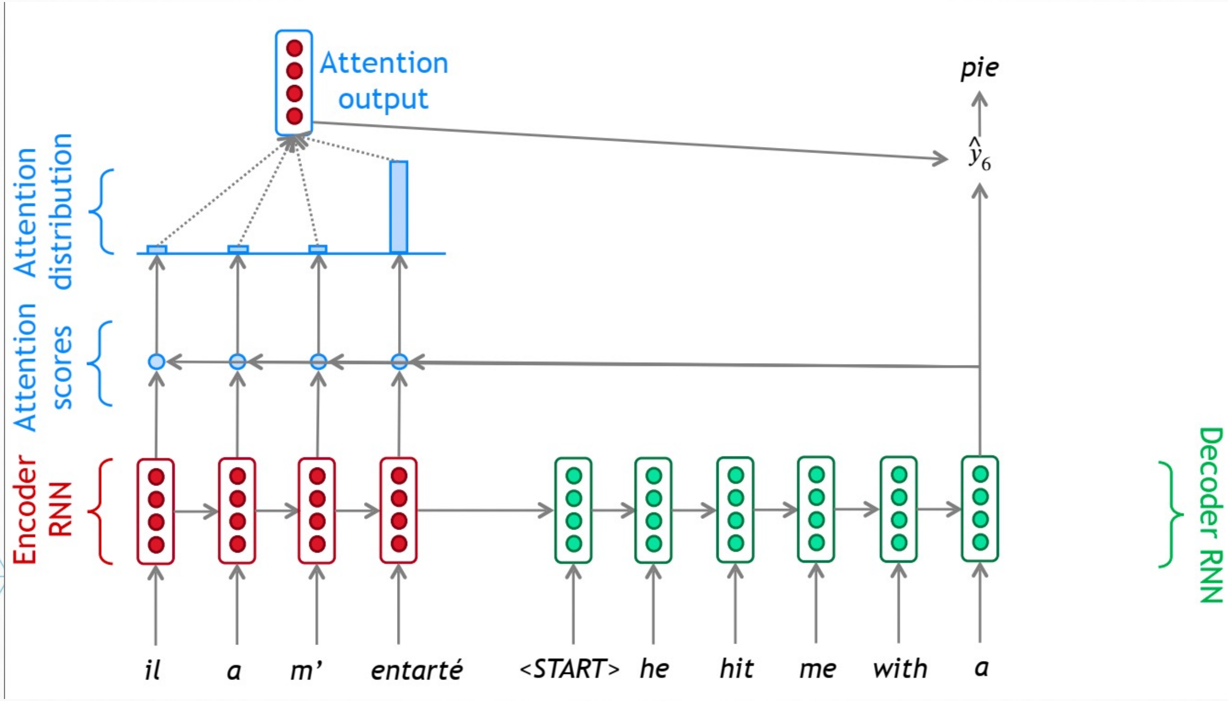
Seq2Seq with attention



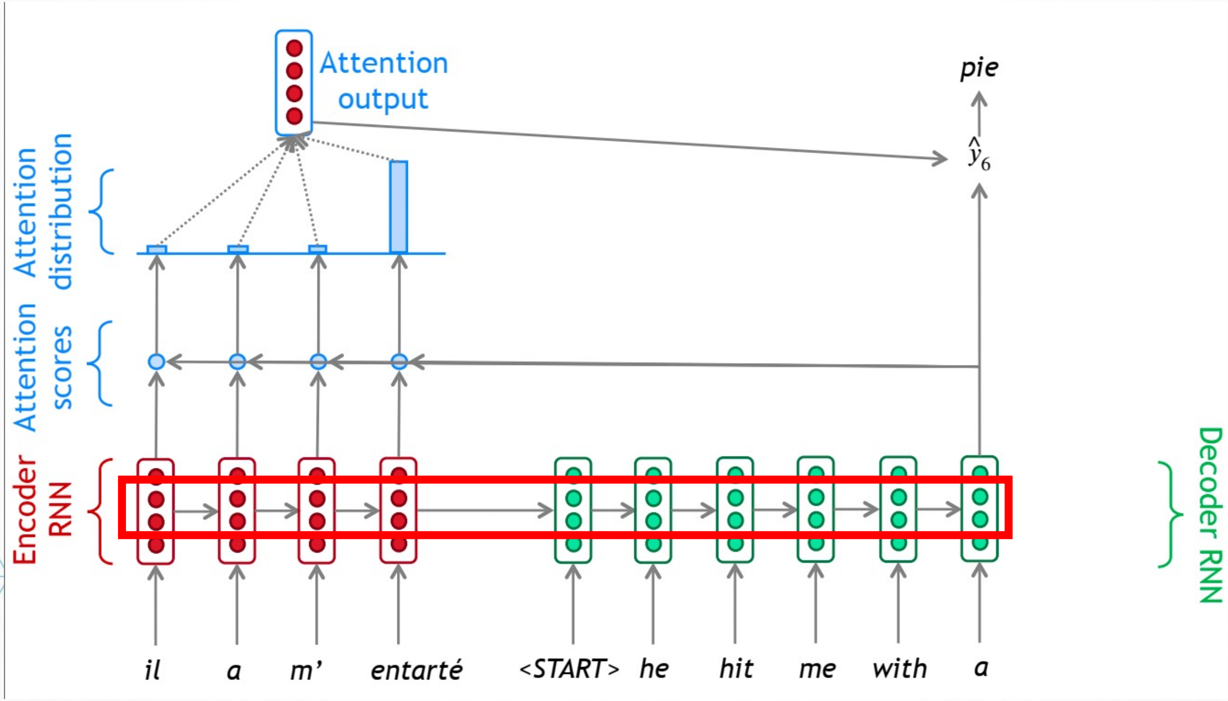
Seq2Seq with attention



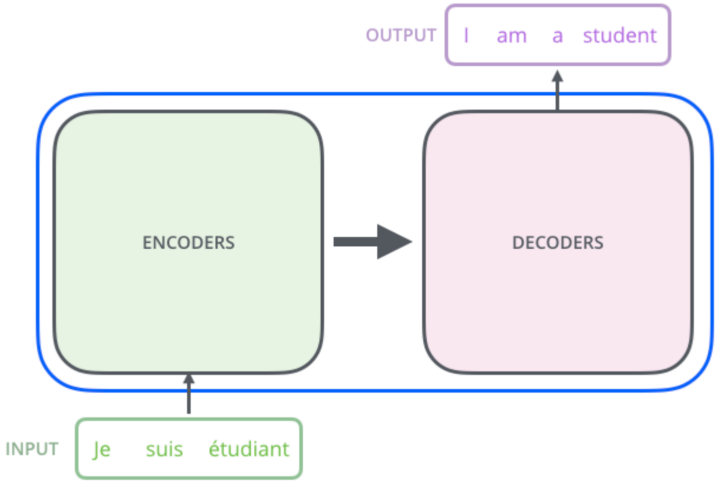
Seq2Seq with attention



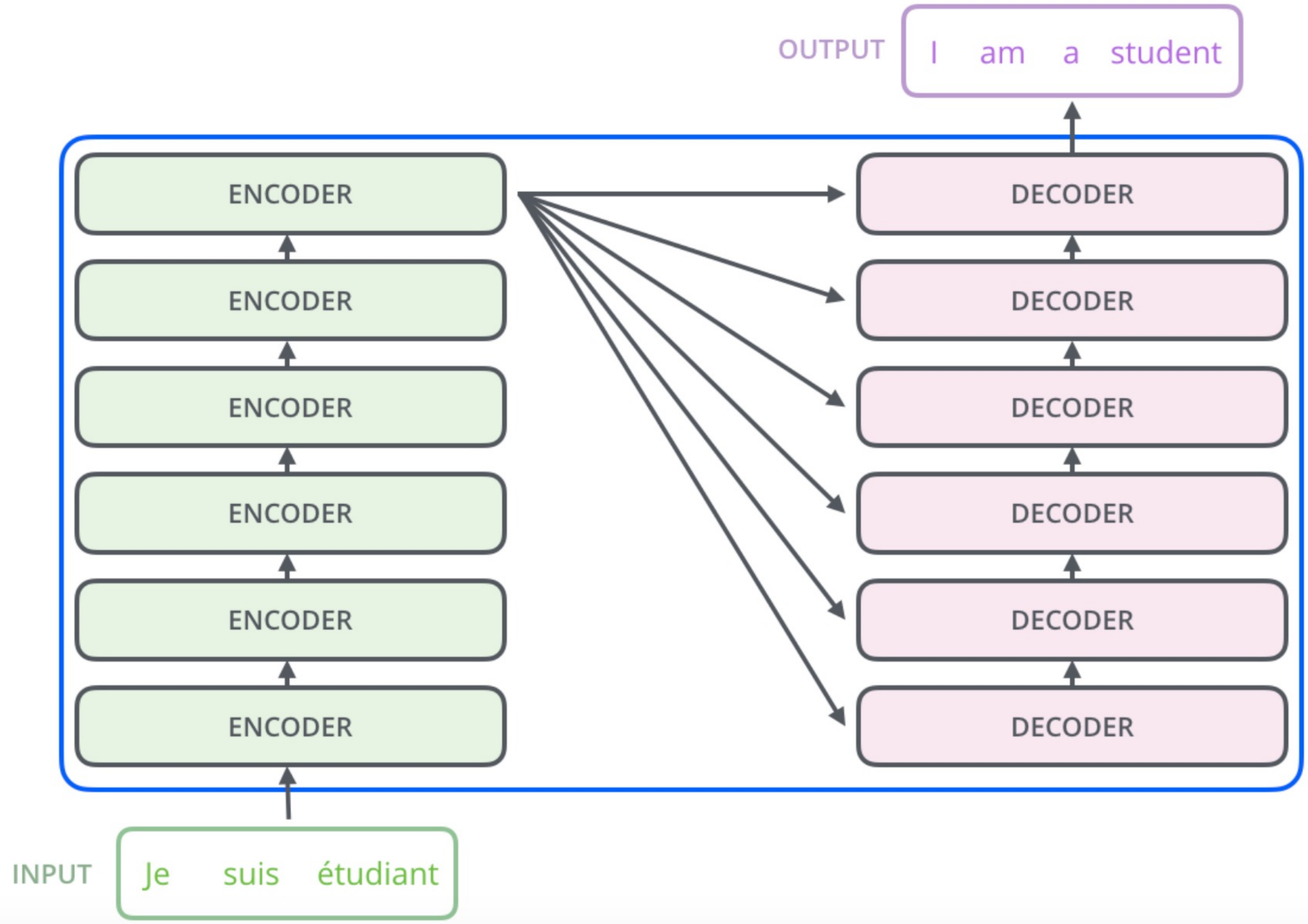
Seq2Seq with another bottleneck



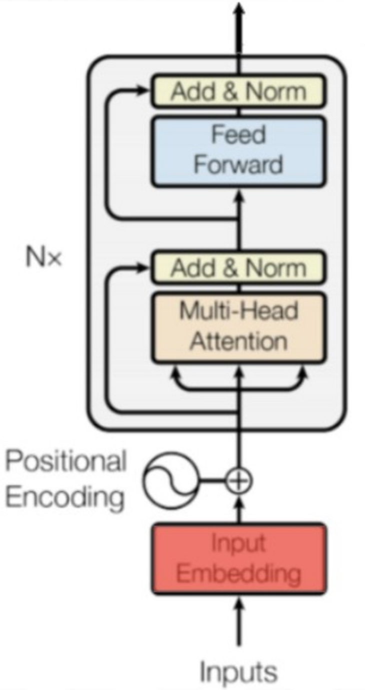

Transformer

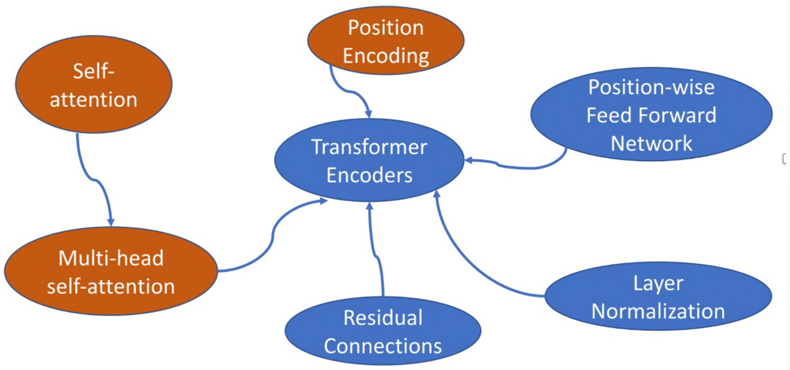


Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in neural information processing systems*. Curran Associates, Inc

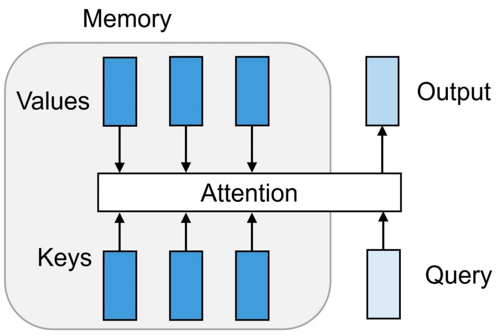

OUTPUT
I am a student
ENCODER
ENCODER
ENCODER
ENCODER
ENCODER
ENCODER
DECODER
DECODER
DECODER
DECODER
DECODER
DECODER
INPUT
Je suis étudiant

Inside an Encoder Block

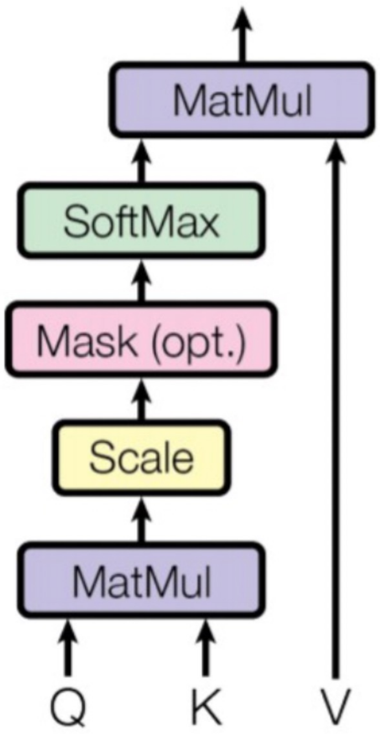




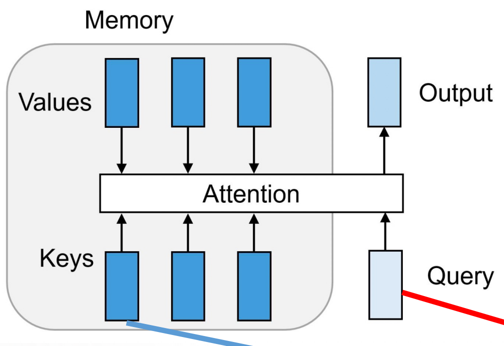
Scaled Dot Product Attention



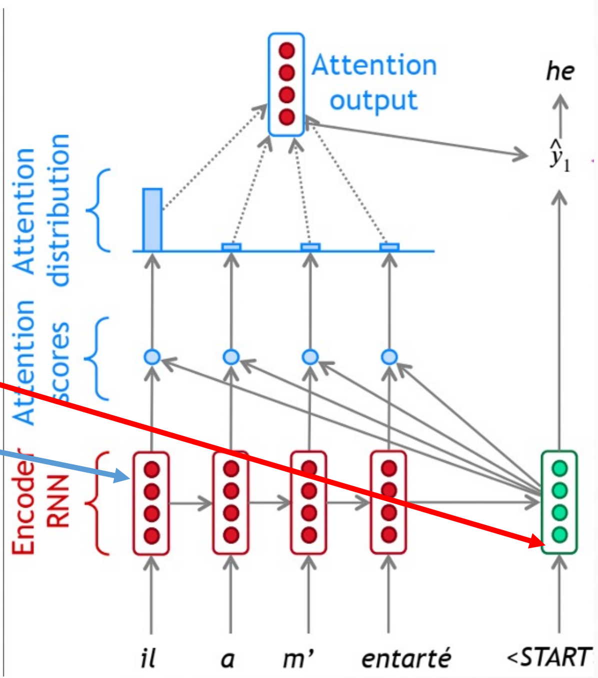
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



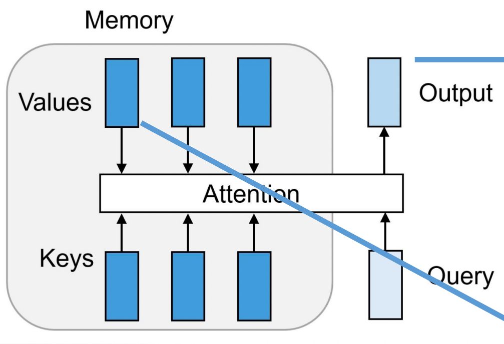
Scaled Dot Product Attention



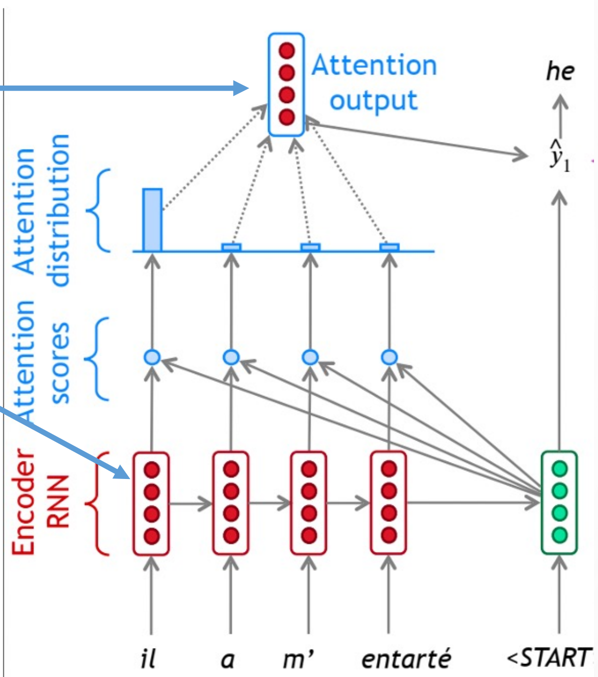
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



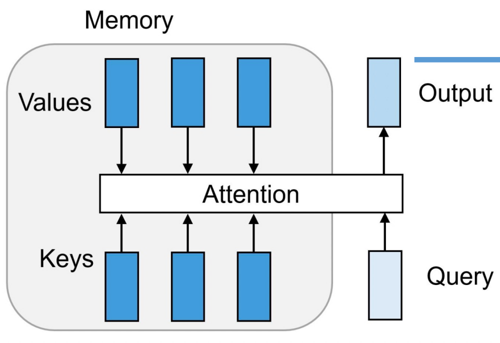
Scaled Dot Product Attention

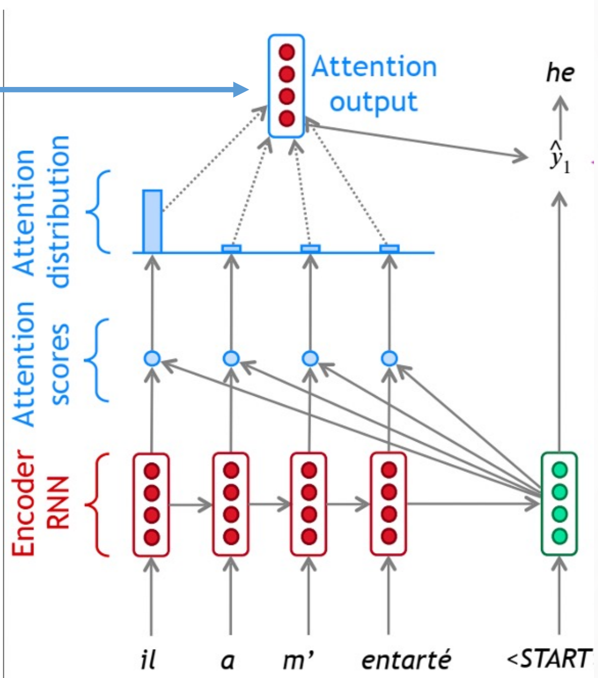


$$\mathrm{Attention}(Q,K,V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



Scaled Dot Product Attention



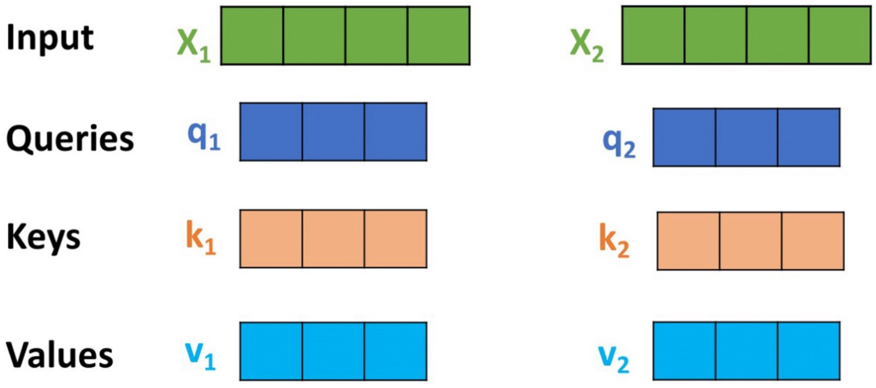


RNN-based Seq2Seq:

- Keys and Values are the same
- Queries are provided from encoder

Self-Attention in Transformer

- Attention maps a query and a set of key-value pairs to an output
  - query, keys, and output are all vectors



Use matrices W^Q, W^K and W^V to project input into query, key and value vectors

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

d_k is the dimension of key vectors

Self-Attention

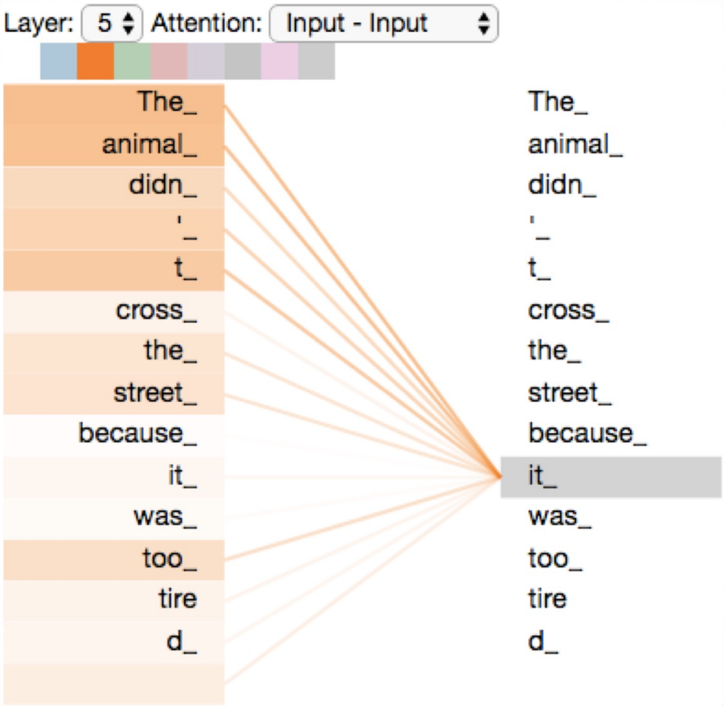
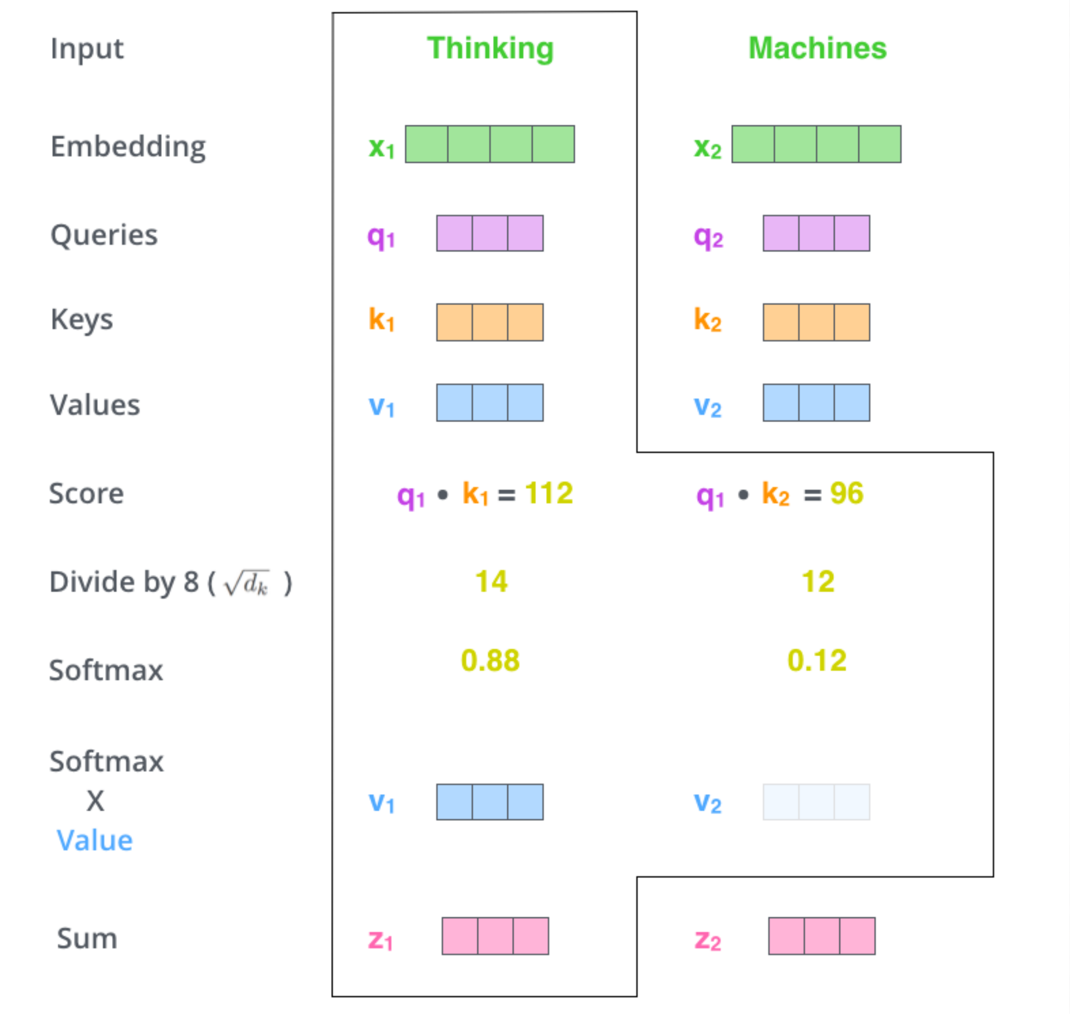
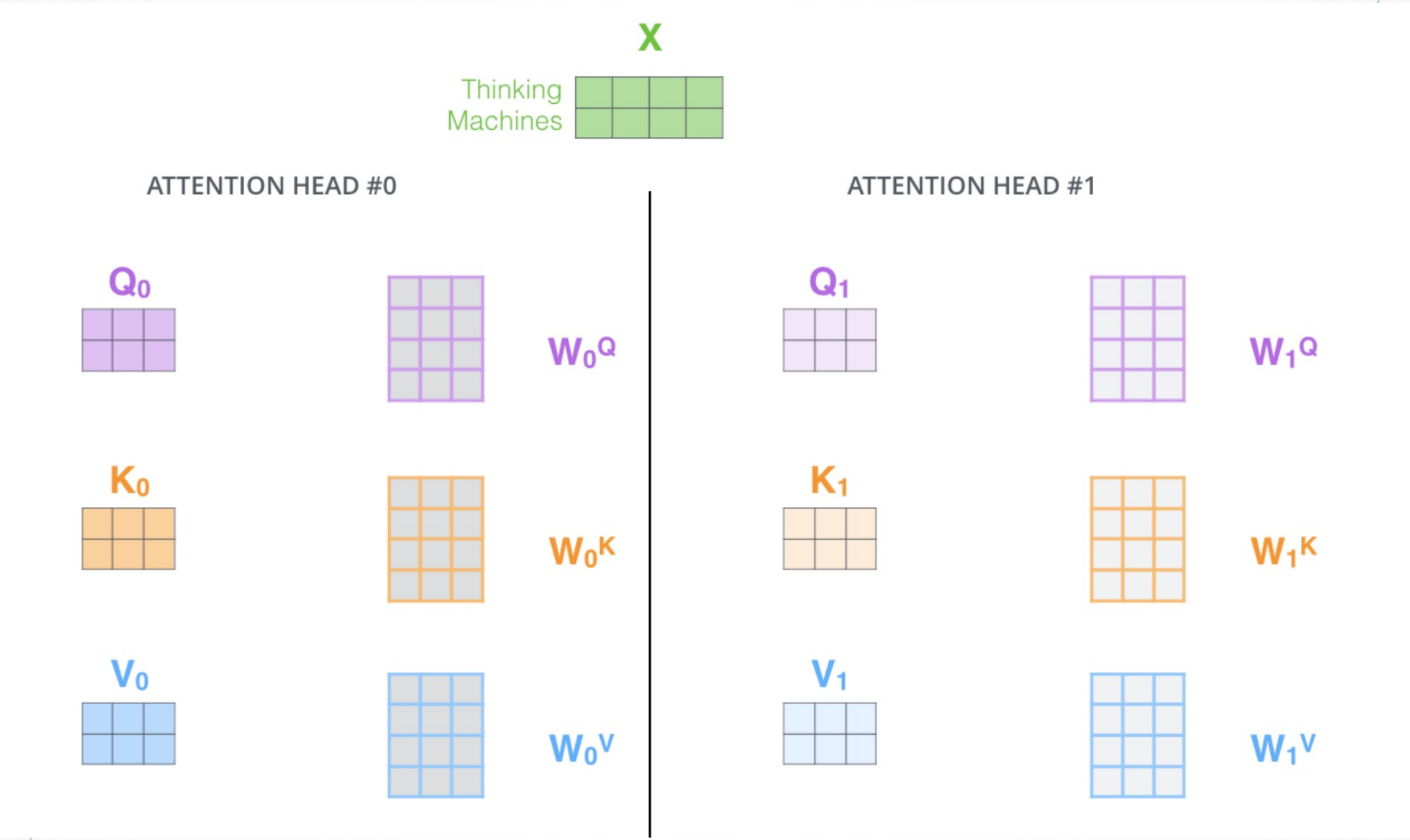


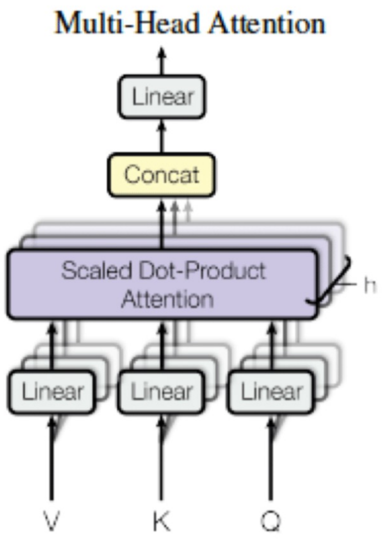
Image source: https://jalammar.github.io/illustrated-transformer/

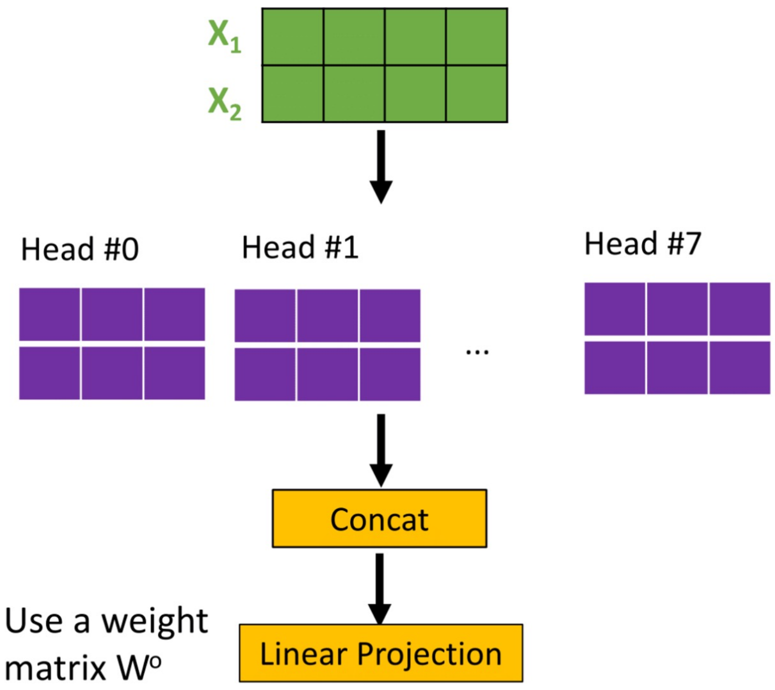

Input
Thinking
Machines
Embedding
x1
x2
Queries
q1
q2
Keys
k1
k2
Values
v1
v2
Score
q1 • k1 = 112
q1 • k2 = 96
Divide by 8 ($\sqrt{d_k}$)
14
12
Softmax
0.88
0.12
Softmax
X
Value
v1
v2
Sum
z1
z2

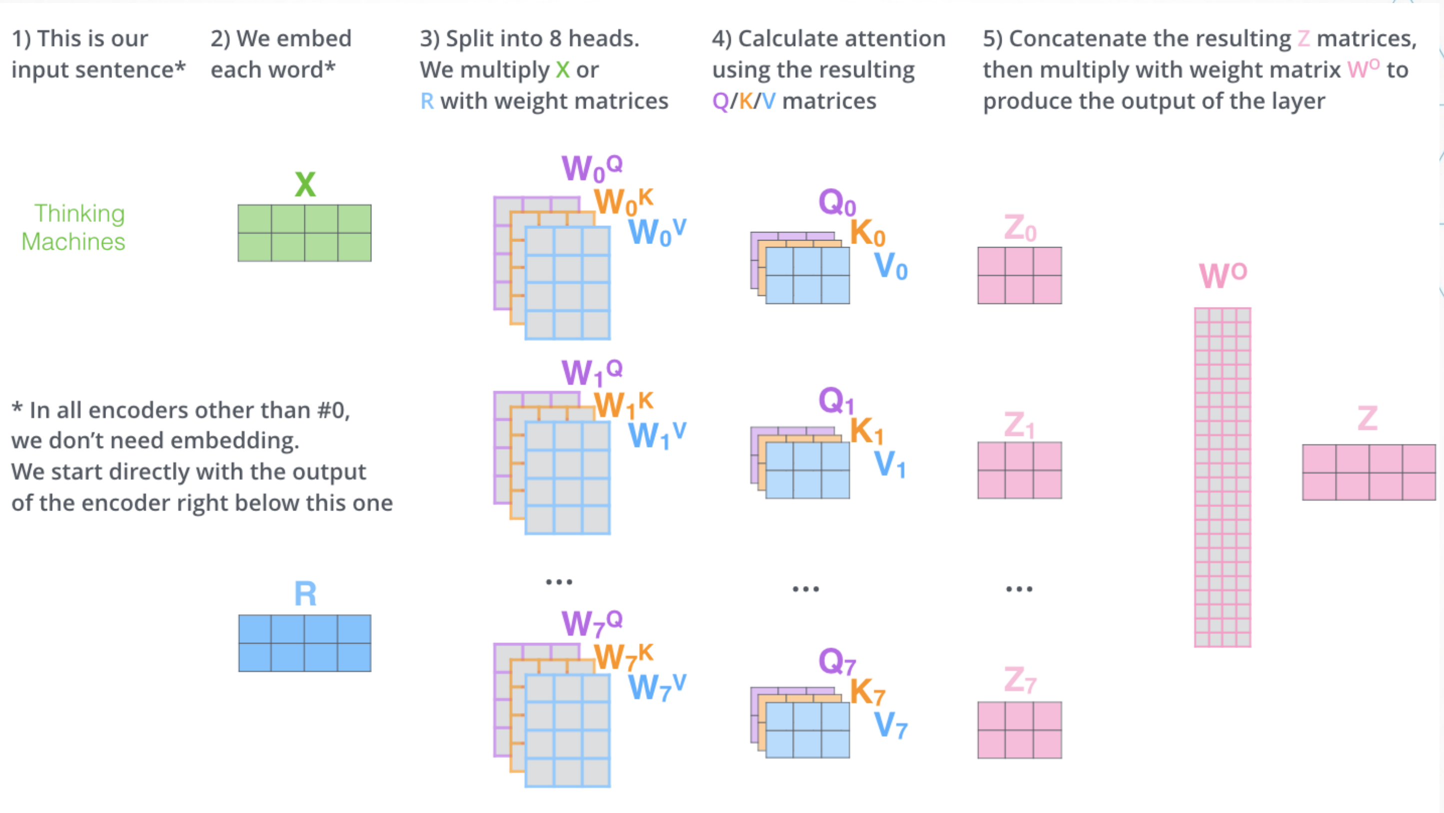


X
Thinking
Machines
ATTENTION HEAD #0
ATTENTION HEAD #1
Q_0
W_0^Q
K_0
W_0^K
V_0
W_0^V
Q_1
W_1^Q
K_1
W_1^K
V_1
W_1^V

Multi-Head Attention







1) This is our input sentence*
2) We embed each word*
3) Split into 8 heads. We multiply X or R with weight matrices
4) Calculate attention using the resulting Q/K/V matrices
5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer
Thinking Machines
X
R
* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one
W_0^Q W_0^K W_0^V
W_1^Q W_1^K W_1^V
...
W_7^Q W_7^K W_7^V
Q_0 K_0 V_0
Q_1 K_1 V_1
...
Q_7 K_7 V_7
Z_0
Z_1
...
Z_7
W^O
Z

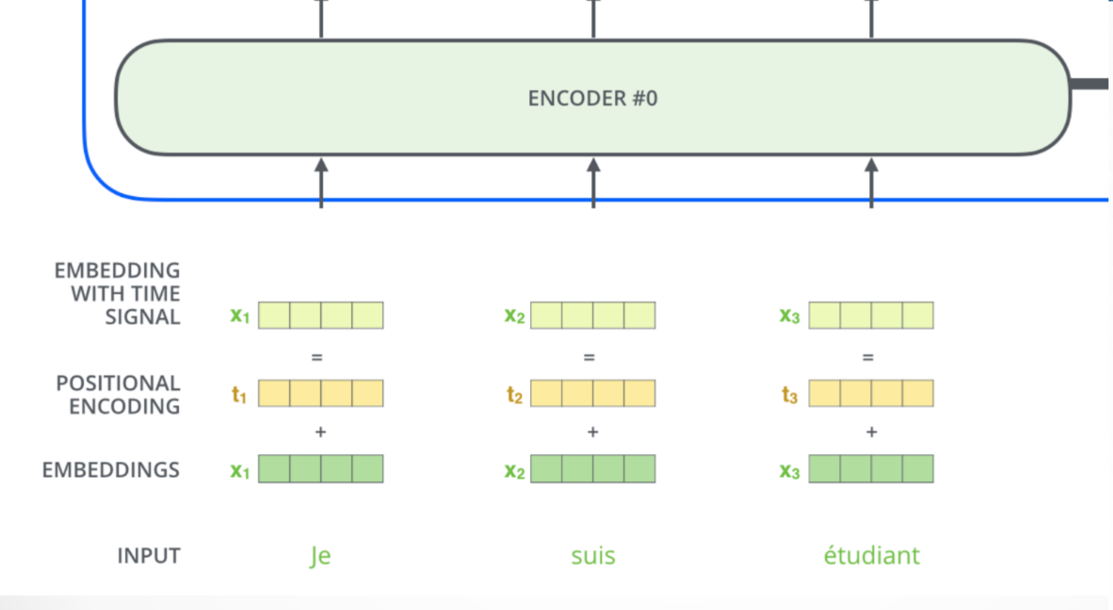
Position Encoding

- Position Encoding is used to make use of the order of the sequence
  - Since the model contains no recurrence and no convolution
- In Vawasni et al., 2017, authors used sine and cosine functions of different frequencies

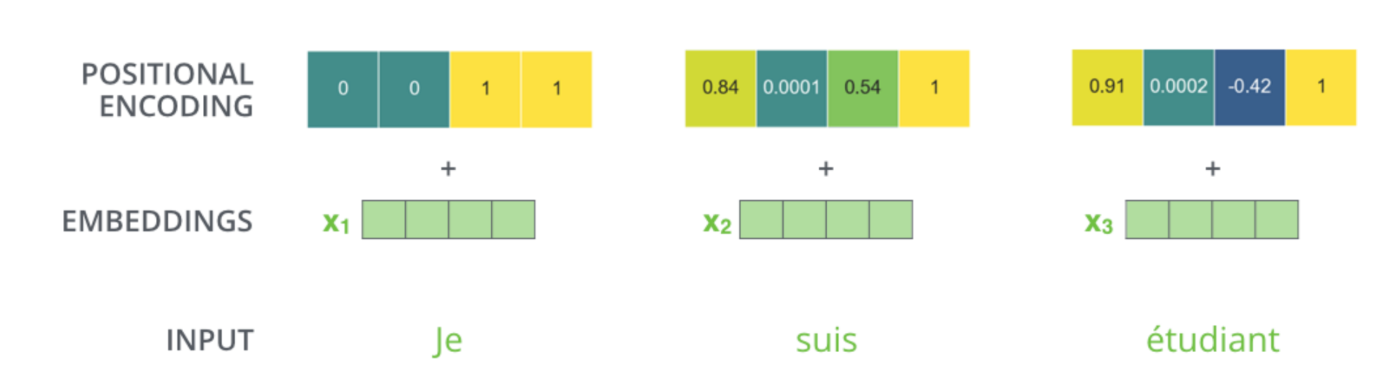
$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right)$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right)$$

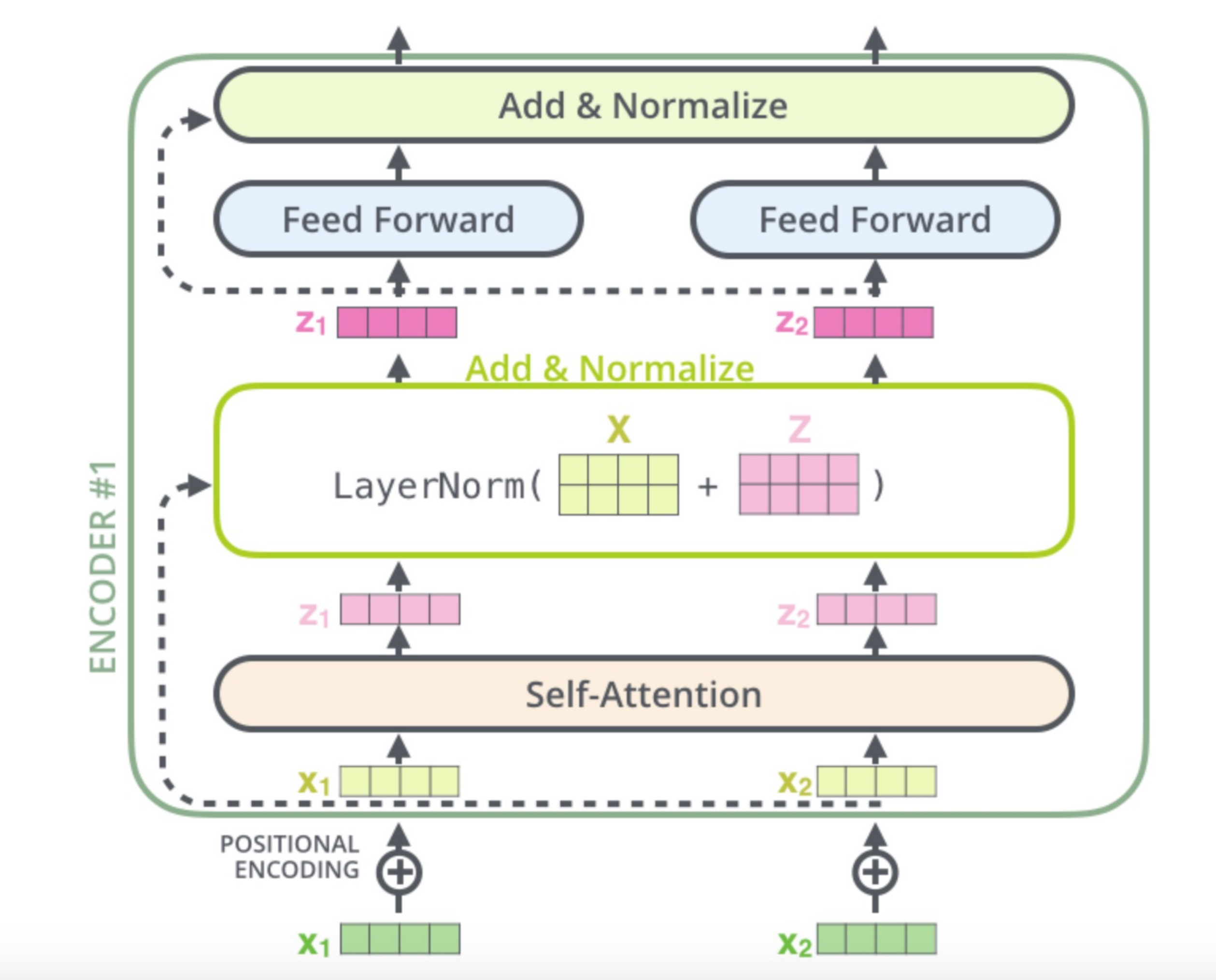
  - *pos* is the position and i is the dimension



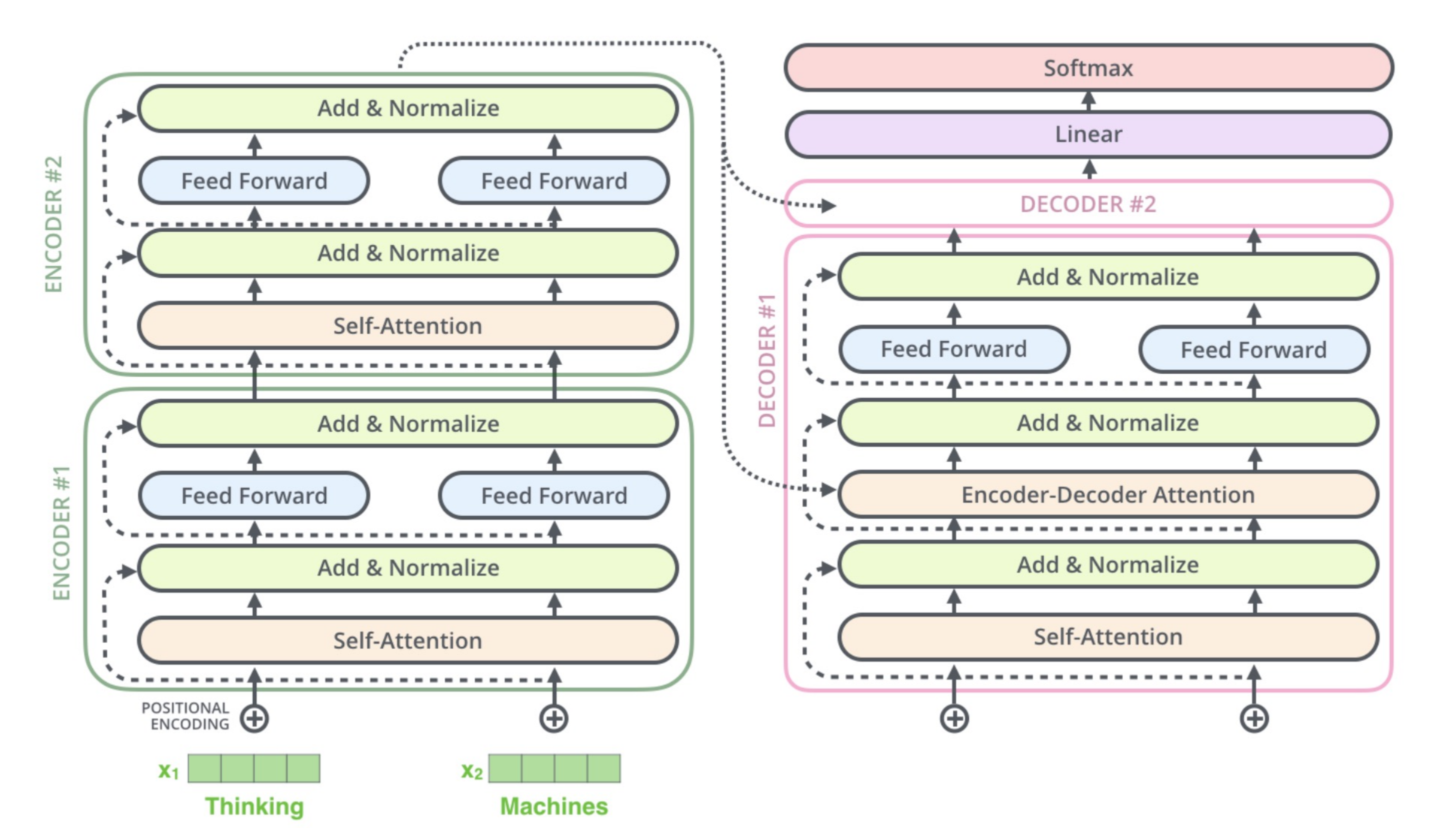
ENCODER #0
EMBEDDING WITH TIME SIGNAL
x1
x2
x3
=
POSITIONAL ENCODING
t1
t2
t3
+
EMBEDDINGS
x1
x2
x3
INPUT
Je
suis
étudiant

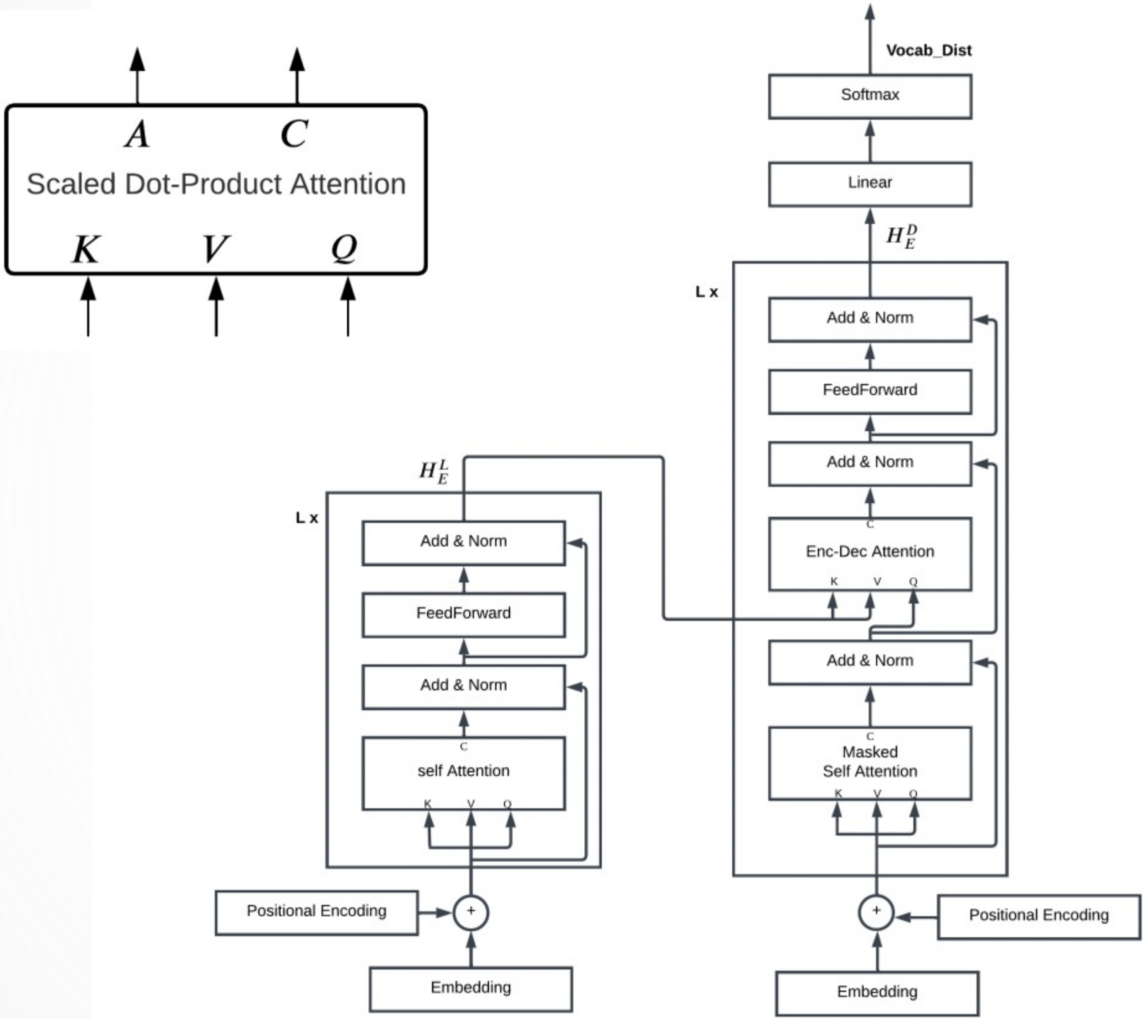


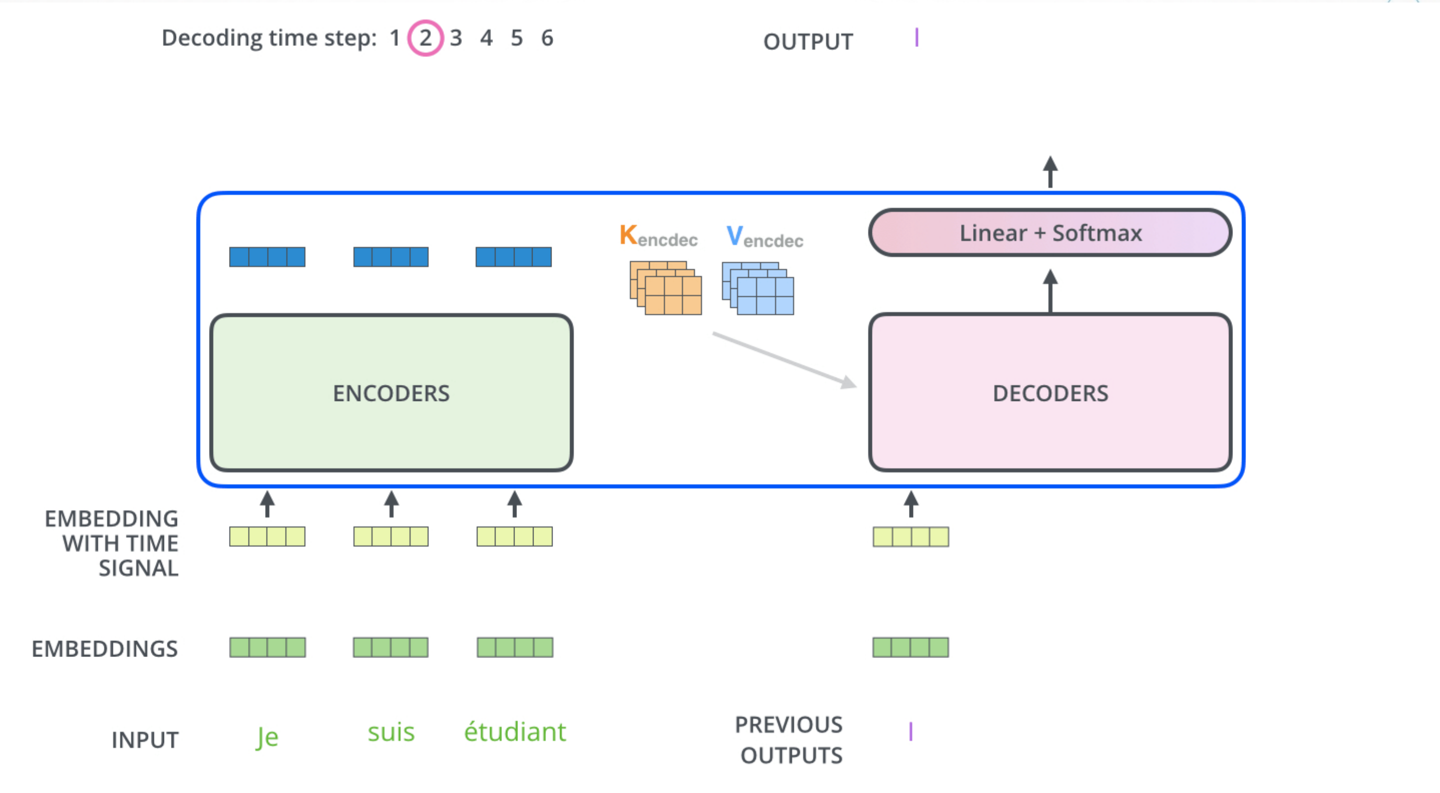
POSITIONAL ENCODING
0
0
1
1
0.84
0.0001
0.54
1
0.91
0.0002
-0.42
1
+
EMBEDDINGS
x1
x2
x3
INPUT
Je
suis
étudiant



Add & Normalize
Feed Forward
Feed Forward
z1
z2
Add & Normalize
X
Z
LayerNorm(+)
ENCODER #1
z1
z2
Self-Attention
x1
x2
POSITIONAL
ENCODING
x1
x2


ENCODER #2
Add & Normalize
Feed Forward
Feed Forward
Add & Normalize
Self-Attention
ENCODER #1
Add & Normalize
Feed Forward
Feed Forward
Add & Normalize
Self-Attention
POSITIONAL ENCODING
x_1
Thinking
x_2
Machines
Softmax
Linear
DECODER #2
DECODER #1
Add & Normalize
Feed Forward
Feed Forward
Add & Normalize
Encoder-Decoder Attention
Add & Normalize
Self-Attention

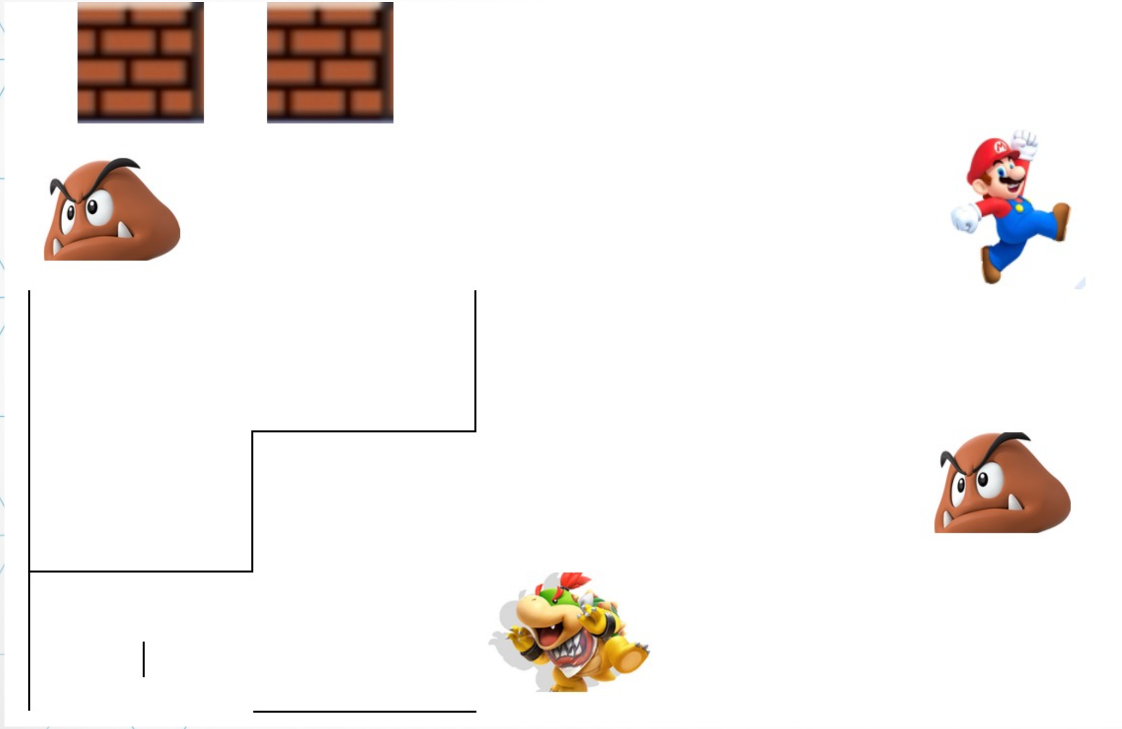

A
C
Scaled Dot-Product Attention
K
V
Q
Vocab_Dist
Softmax
Linear
H_E^D
L x
Add & Norm
FeedForward
Add & Norm
Enc-Dec Attention
Add & Norm
Masked Self Attention
H_E^L
L x
Add & Norm
FeedForward
Add & Norm
self Attention
Positional Encoding
Embedding
Positional Encoding
Embedding


Decoding time step: 1 2 3 4 5 6
OUTPUT
I
Kencdec
Vencdec
Linear + Softmax
ENCODERS
DECODERS
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant
PREVIOUS OUTPUTS
I

Part II: Mamba

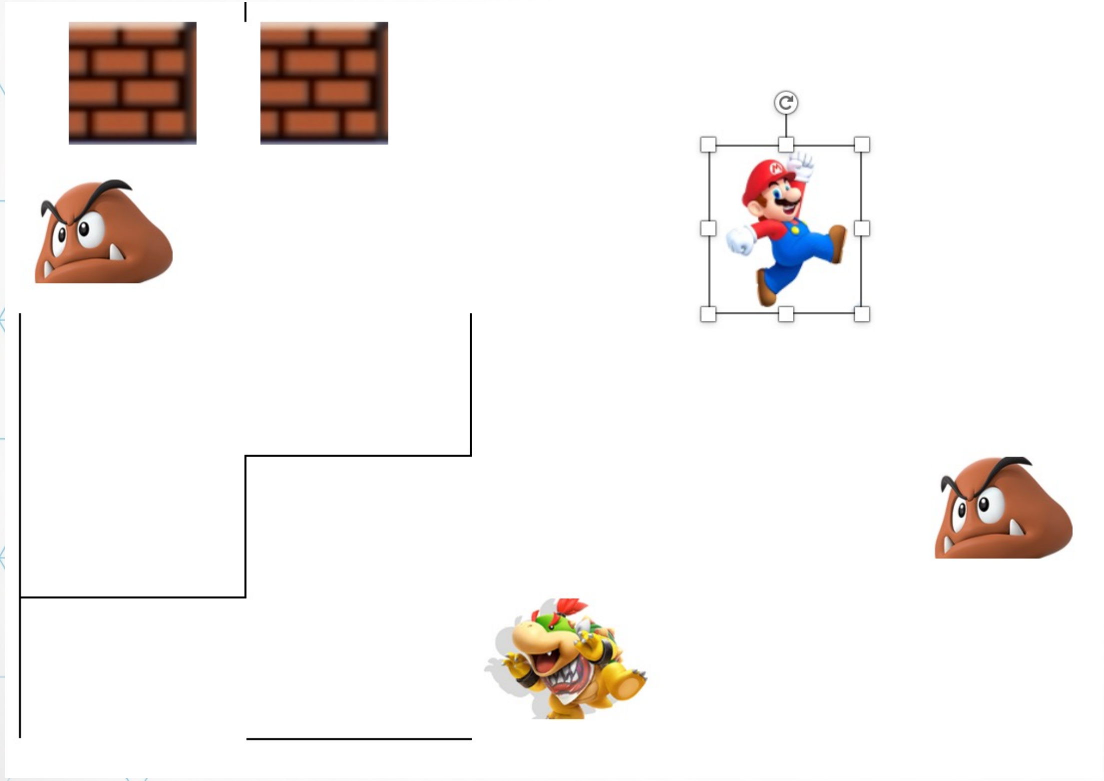
To save the princess: a state-based approach

Where is the princess?



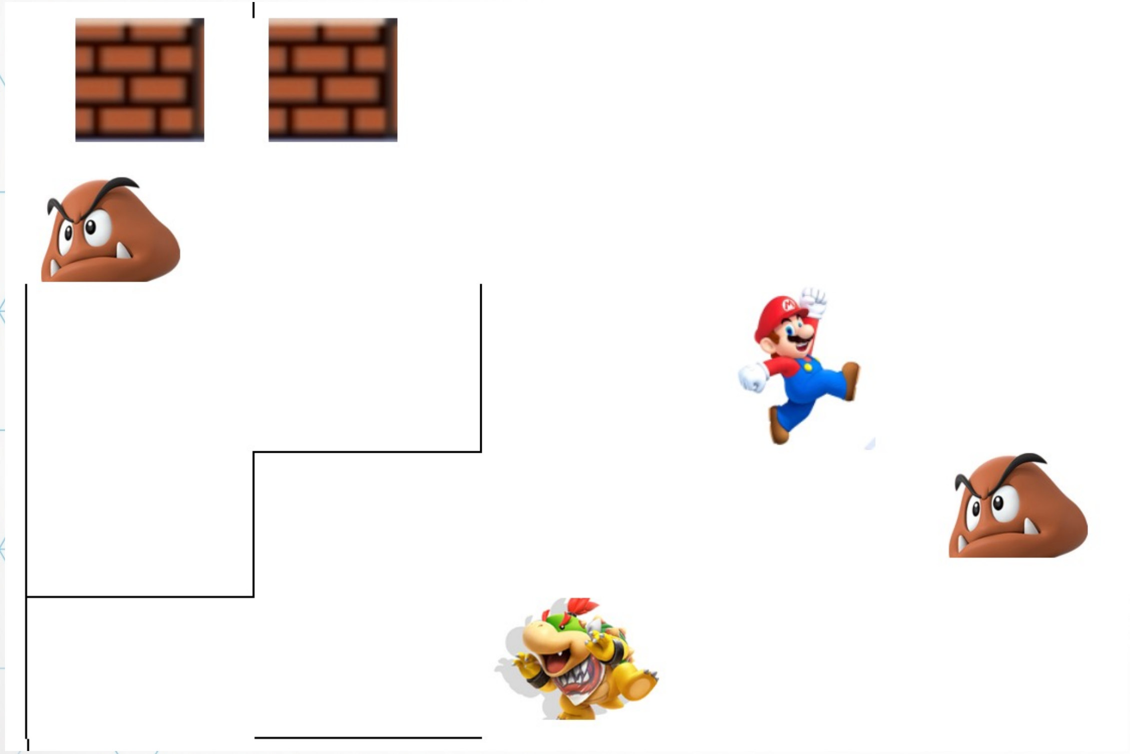
Input Sequence

X_1=L



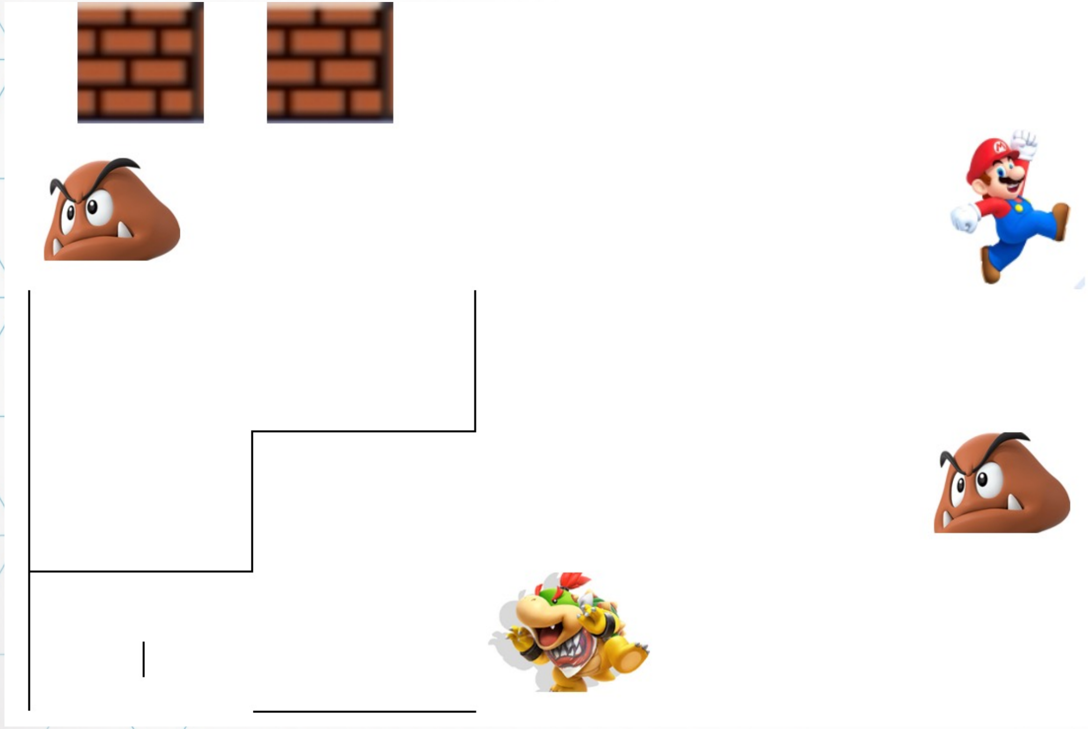
Input Sequence

X_1=L
X_2=D



Where is the princess?

X_0= N/A s(0) = (5,4)

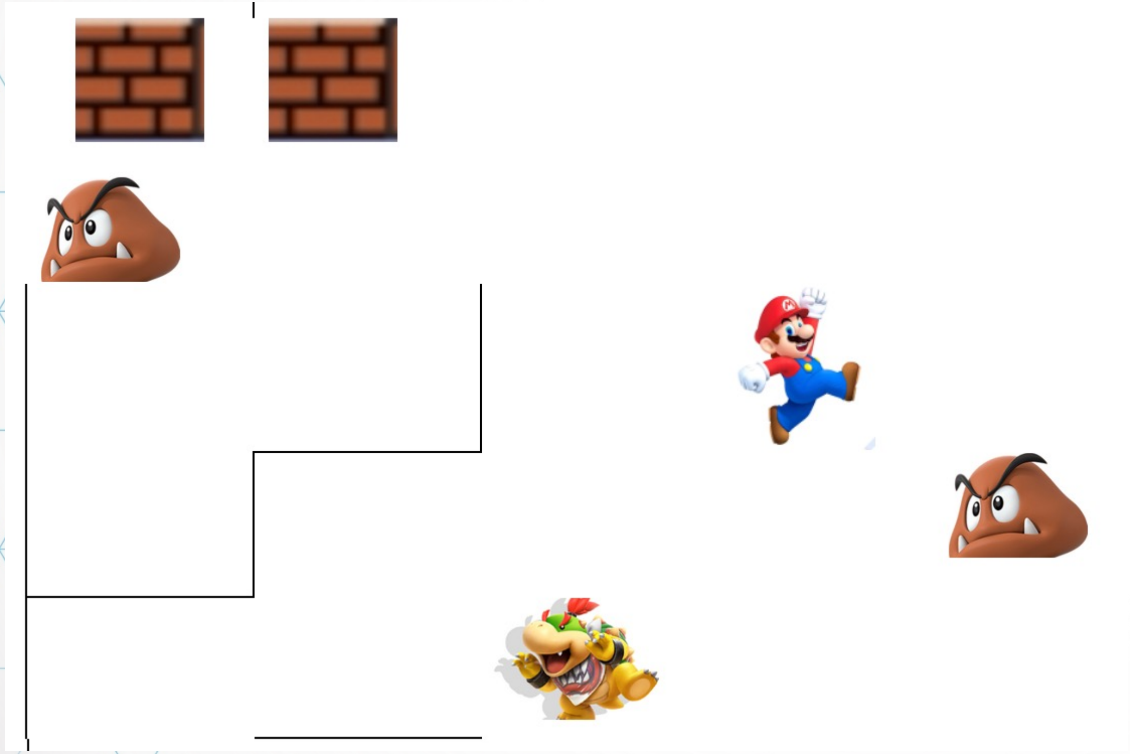


State Transition

$X_0 = \_ \quad s(0) = (5,4)$

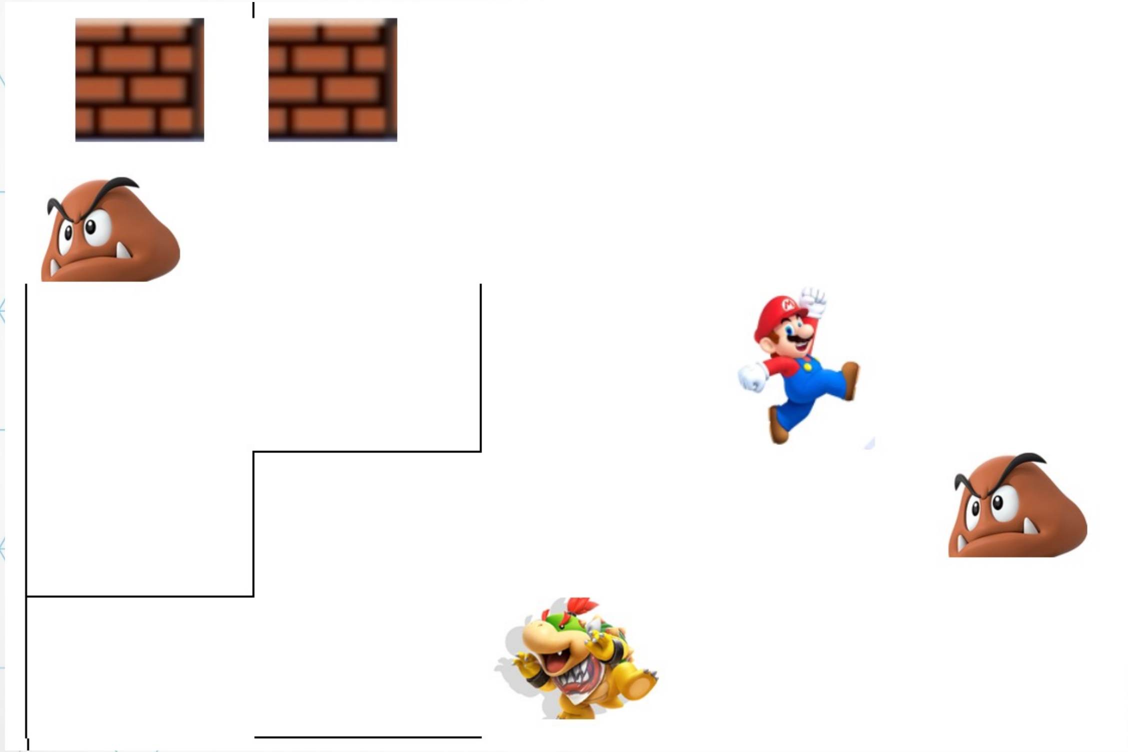
$X_1 = L \quad s(1) = (4,4)$

$X_2 = D \quad s(2) = (4,3)$



State Representation

$X_0 = \_ \;\; h(0) = (5,4,7)$
$X_1 = L \;\; h(1) = (4,4,6)$
$X_2 = D \;\; h(2) = (4,3,5)$



Next move?

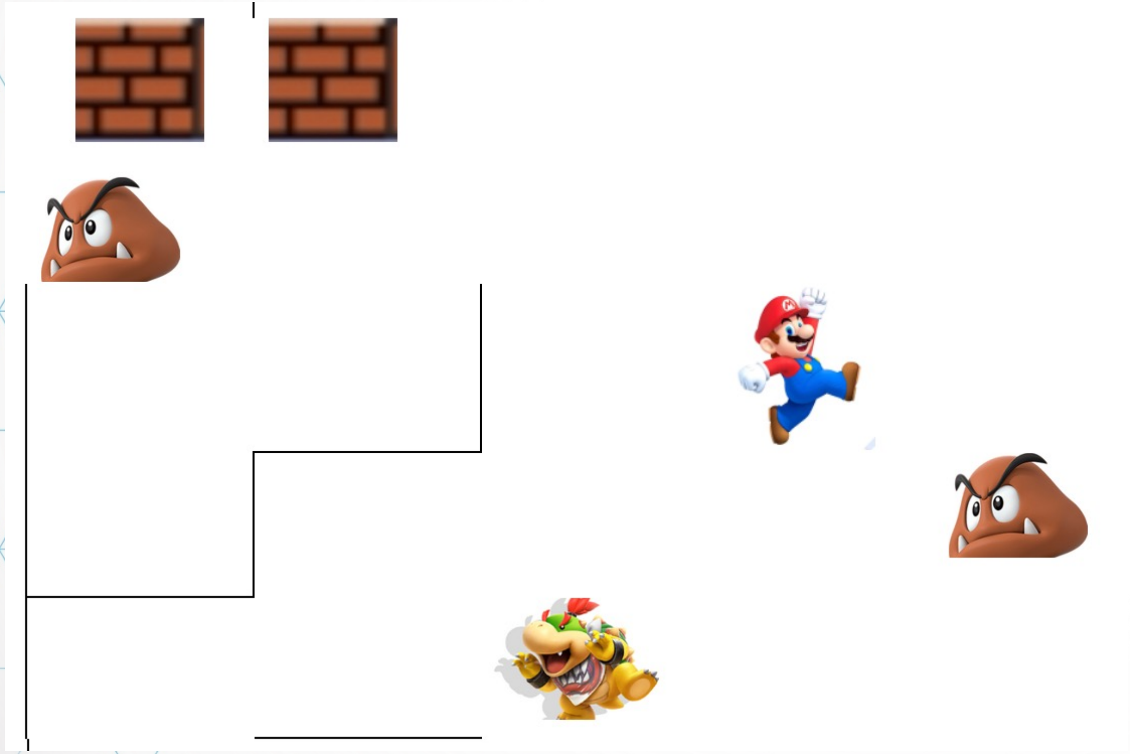
$X_0 = \_$ $h(0) = (5,4,7)$

$X_1 = L$ $h(1) = (4,4,6)$

$X_2 = D$ $h(2) = (4,3,5)$

$h(3) = ?$

$Y(\sim X3) = ?$

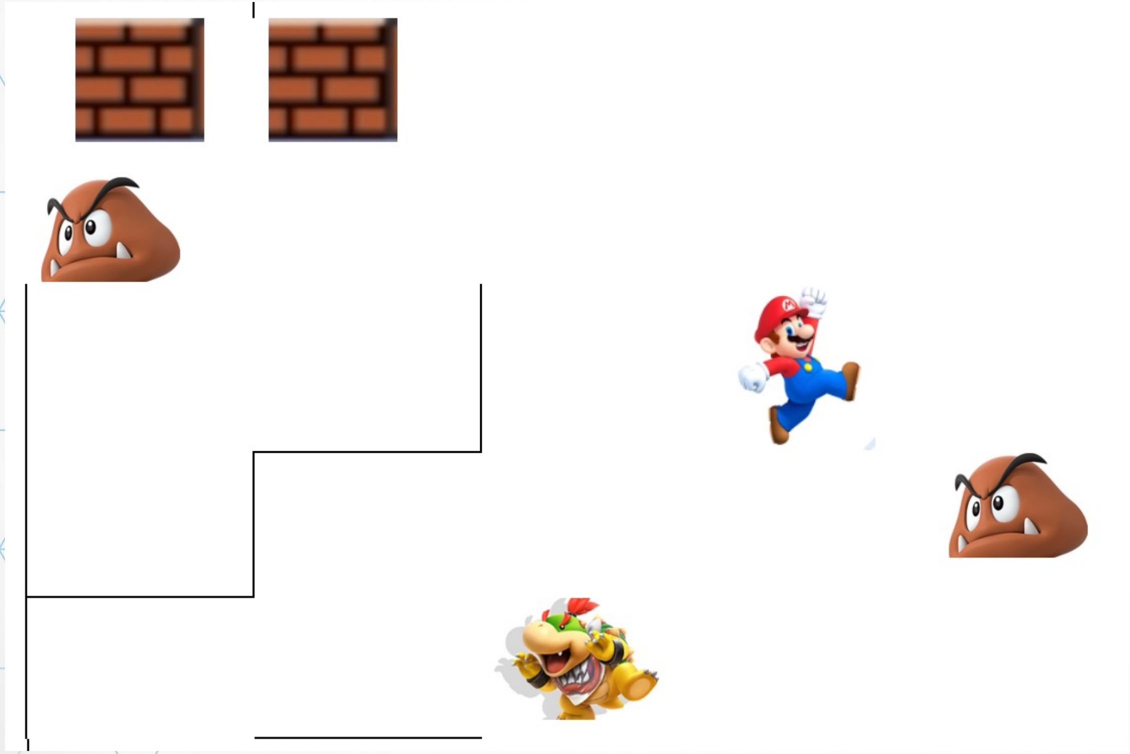


Next move from the current state

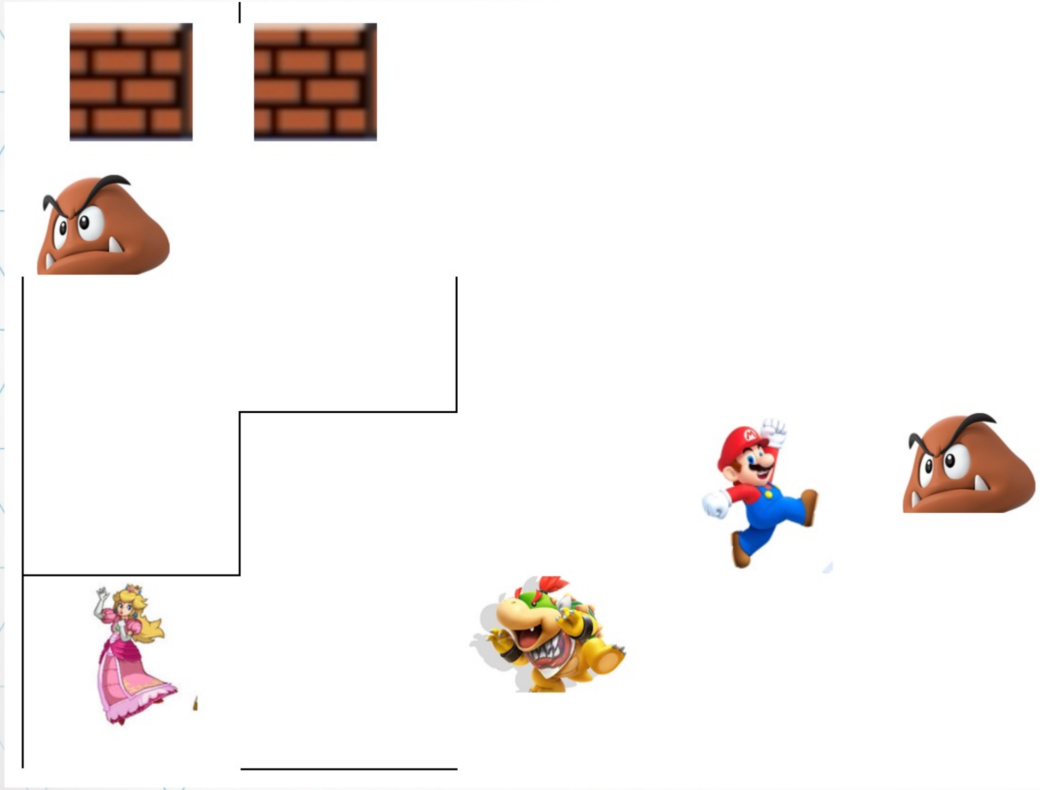
X_2=D h(2) = (4,3,5)

h(3) = ?

Y (~X3) = ?



Let's save the princess



X_2=D h(2) = (4,3,5)

h(3) = (4,3,4)

Y (~X3) = D

...

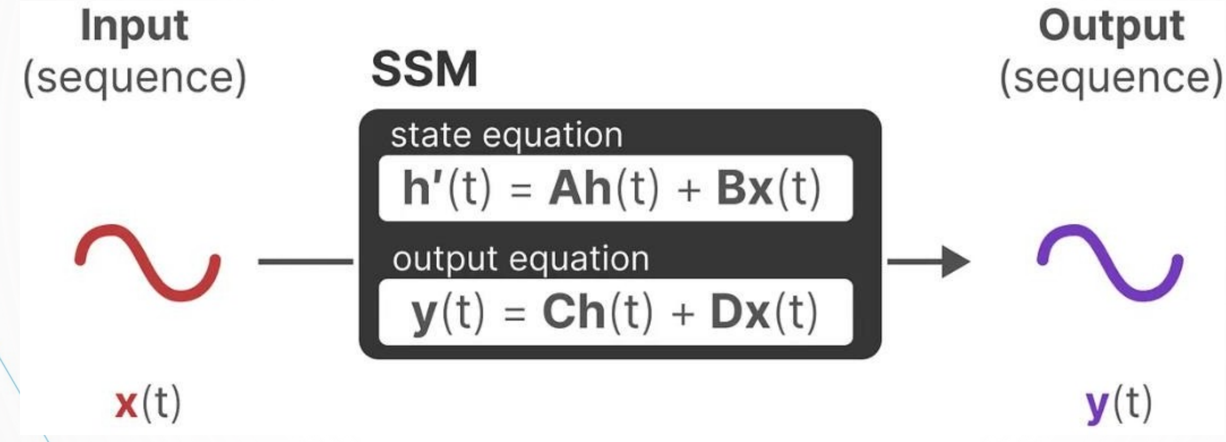
h_final = (1,1,0)

Introduction: State space model

SSMs are models used to describe these state representations and make predictions of what their next state could be depending on some input

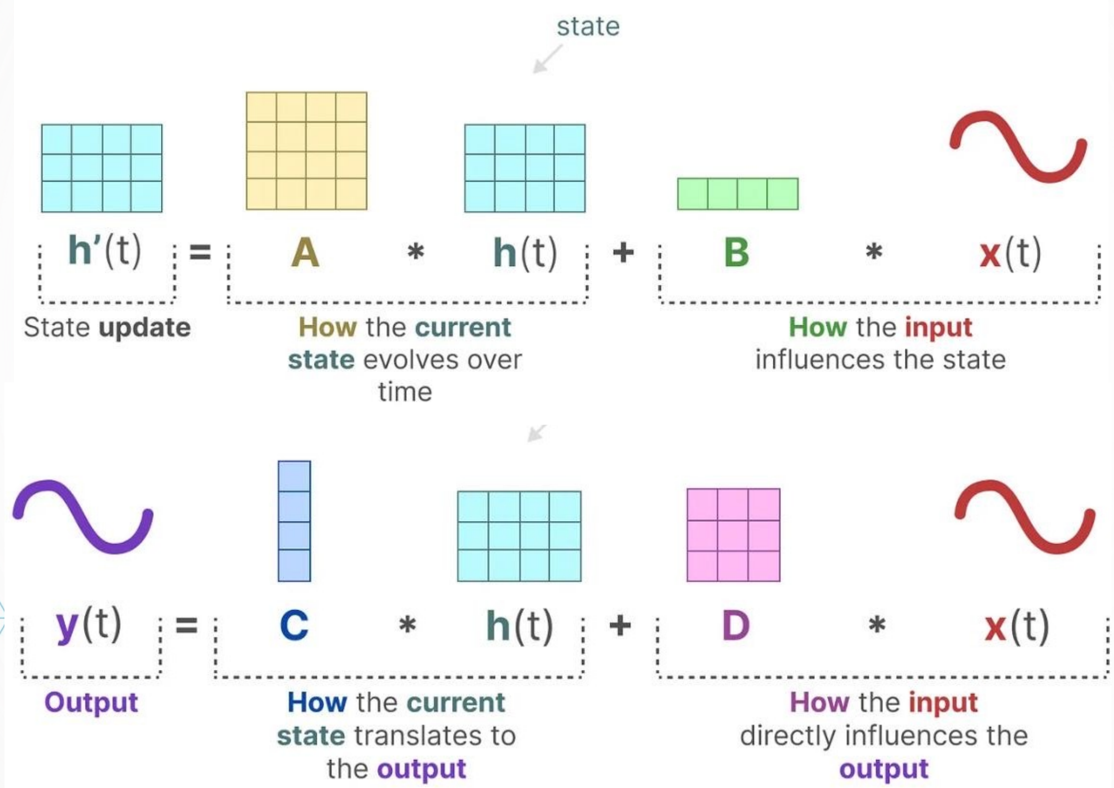
Introduction: State space model

Assumption: Dynamic system can be predicted from its state at time t by two equations



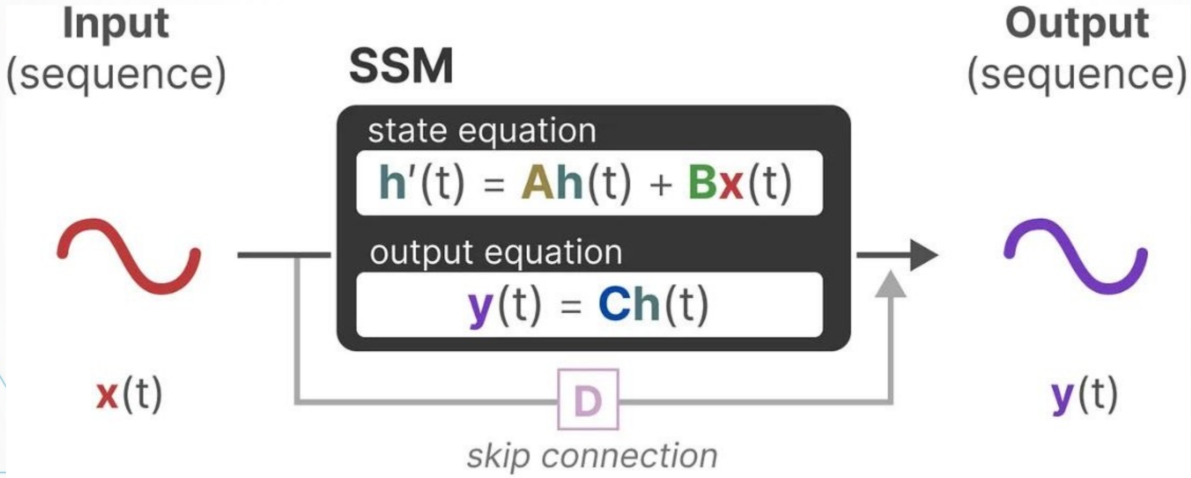
=> **Goal**: Find h(t)

Introduction: State space model



Introduction: State space model

Ignore the skip connection in the SSM

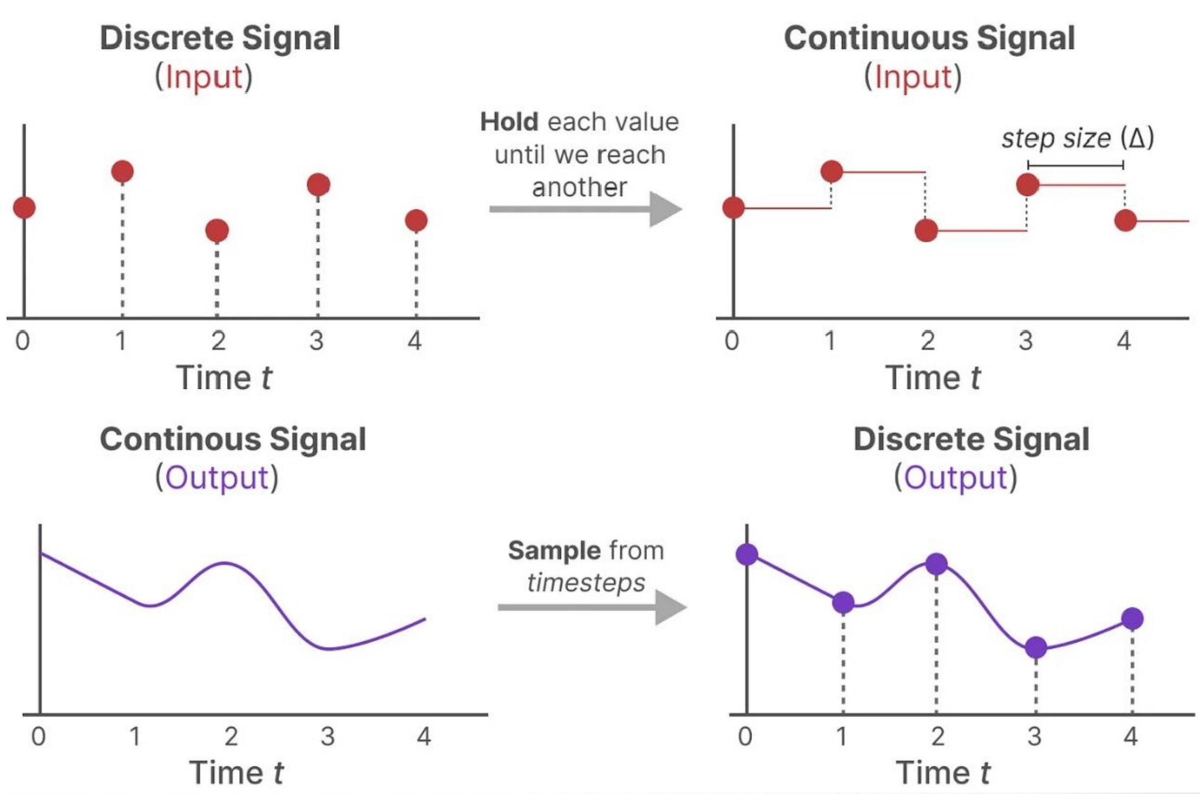


=> Only work for continuous time representation?

SSMs Discretization

With text input, need a discrete model (discrete -> discrete)

Zero order hold:
- Input: Hold value until meet another
- Output: Sample discrete values

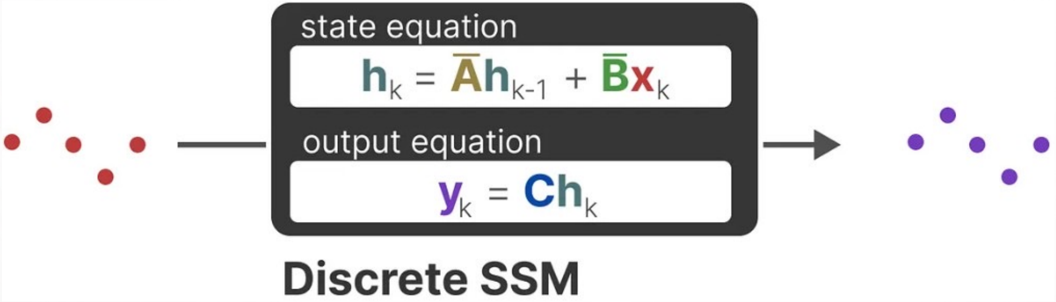


SSMs Discretization

Discrete SSM:

$$\bar{\mathbf{A}} = \exp(\Delta \mathbf{A})$$

$$\bar{\mathbf{B}} = (\Delta \mathbf{A})^{-1}(\exp(\Delta \mathbf{A}) - I) \cdot \Delta \mathbf{B}$$



SSM = RNN + Transformer?

Introduction

Problem with Transformers

Pros:

- Can route information densely within a context window
- Parallelizable training

Cons:

- Slow inference: scale quadratically with sequence length
- Can not model anything outside the context window

Introduction

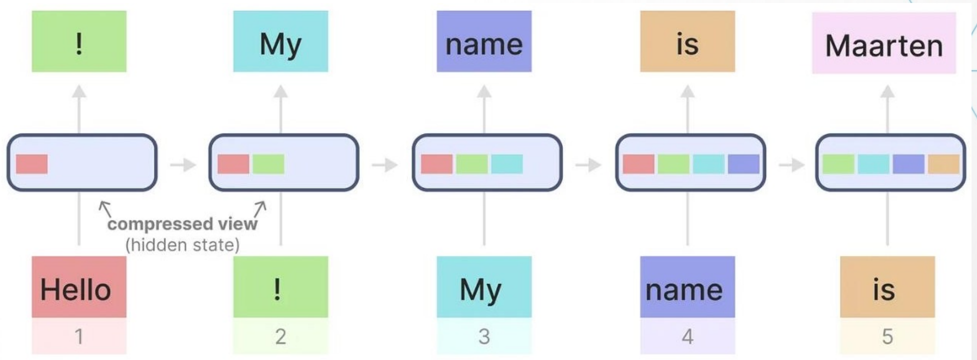
Is RNN a solution?

Pros:

- Inference is fast (linear) + can have large context length (in theory)

Cons:

- Tend to forget information over time
- Training can't be done in parallel



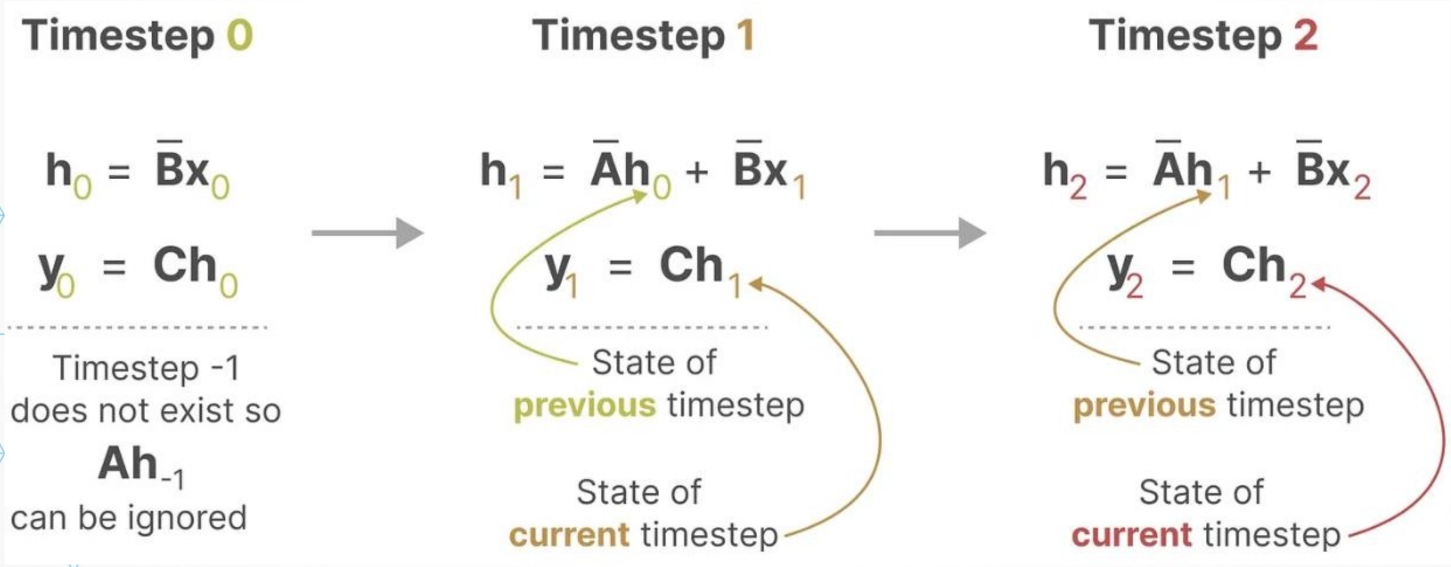
Introduction

Can we leverage both pros of the two architectures?

	Training	Inference
Transformers	**Fast!** (parallelizable)	**Slow...** (scales **quadratically** with sequence length)
RNNs	**Slow...** (not parallelizable)	**Fast!** (scales **linearly** with sequence length)

The two views of SSMs

SSM: The **RNN view** => **Fast inference**



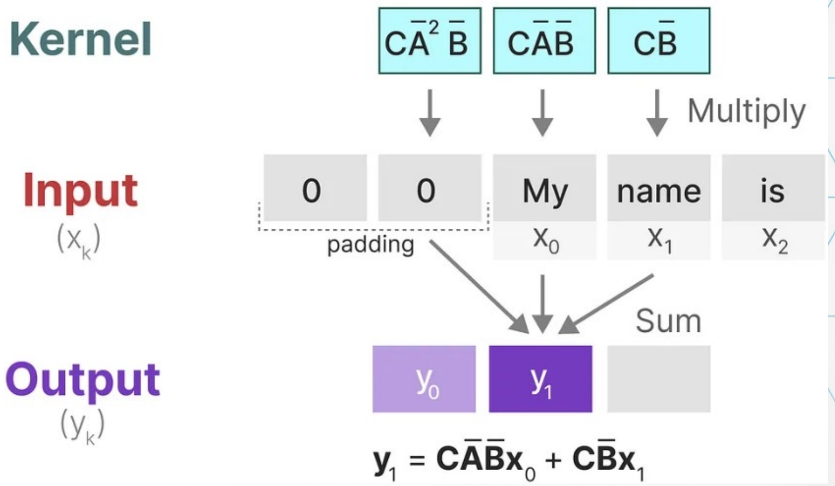
The two views of SSMs

SSM: The CNN view => Parallelizable training

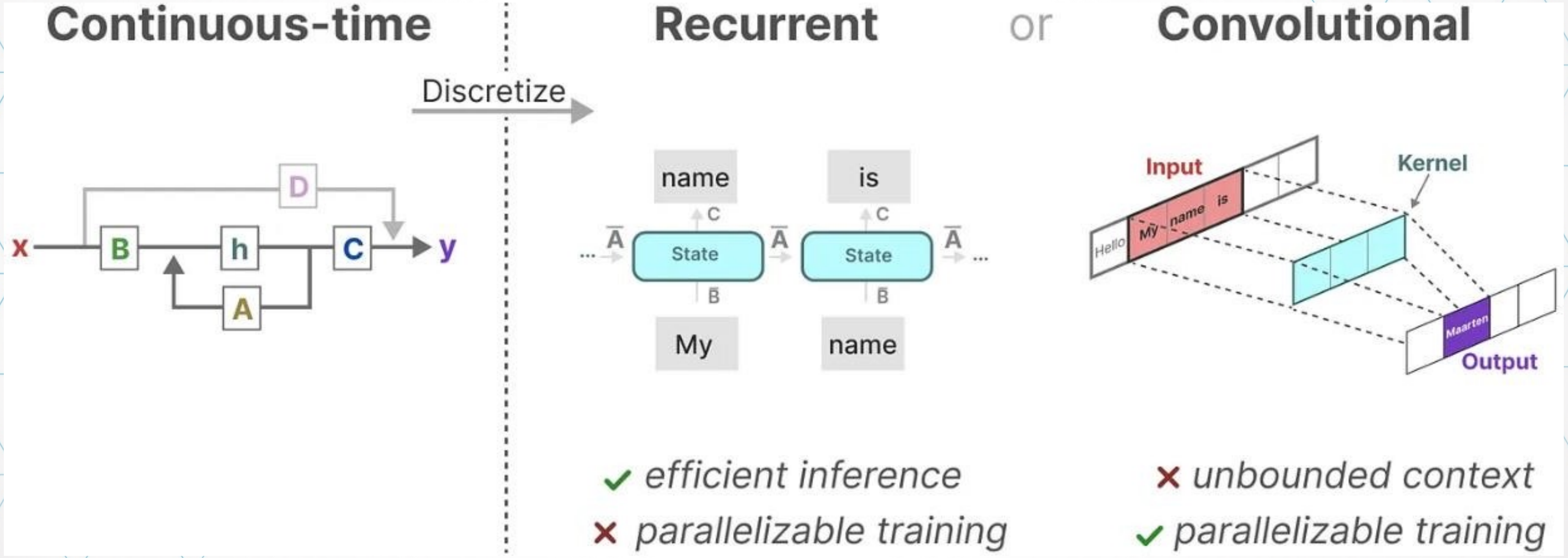
$$h_0 = \bar{B}x_0$$
$$y_0 = Ch_0 = C\bar{B}x_0$$

$$h_1 = \bar{A}h_0 + \bar{B}x_1 = \bar{A}\bar{B}x_0 + \bar{B}x_1$$
$$y_1 = Ch_1 = C(\bar{A}\bar{B}x_0 + \bar{B}x_1) = C\bar{A}\bar{B}x_0 + C\bar{B}x_1$$

$$h_2 = \bar{A}h_1 + \bar{B}x_2 = \bar{A}(\bar{A}\bar{B}x_0 + \bar{B}x_1) + \bar{B}x_2 = \bar{A}^2\bar{B}x_0 + \bar{A}\bar{B}x_1 + \bar{B}x_2$$
$$y_2 = Ch_2 = C(\bar{A}^2\bar{B}x_0 + \bar{A}\bar{B}x_1 + \bar{B}x_2) = C\bar{A}^2\bar{B}x_0 + C\bar{A}\bar{B}x_1 + C\bar{B}x_2$$



The two views of SSMs

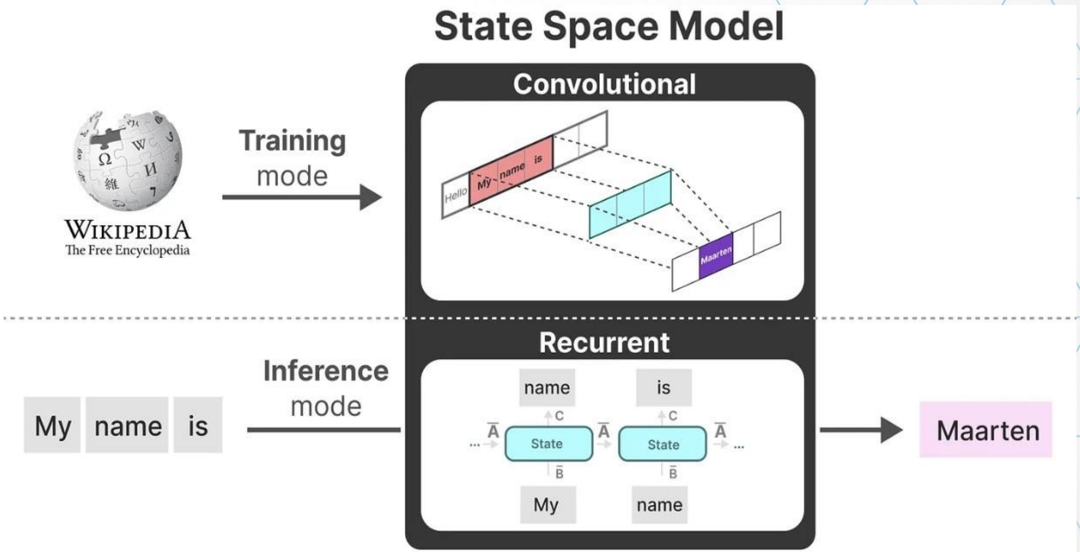


Introduction: State space model

Important property of CNN and RNN representations: **Linear time invariant**

All state space model params are **fixed for all time steps**

=> Static representation, not **content-aware**



HiPPO

$$h_t = \overline{A}h_{t-1} + \overline{B}x_t$$
$$y_t = Ch_t$$

The matrix A: captures information from previous state

- Need to be designed **carefully**

Random initialized A => poor performance

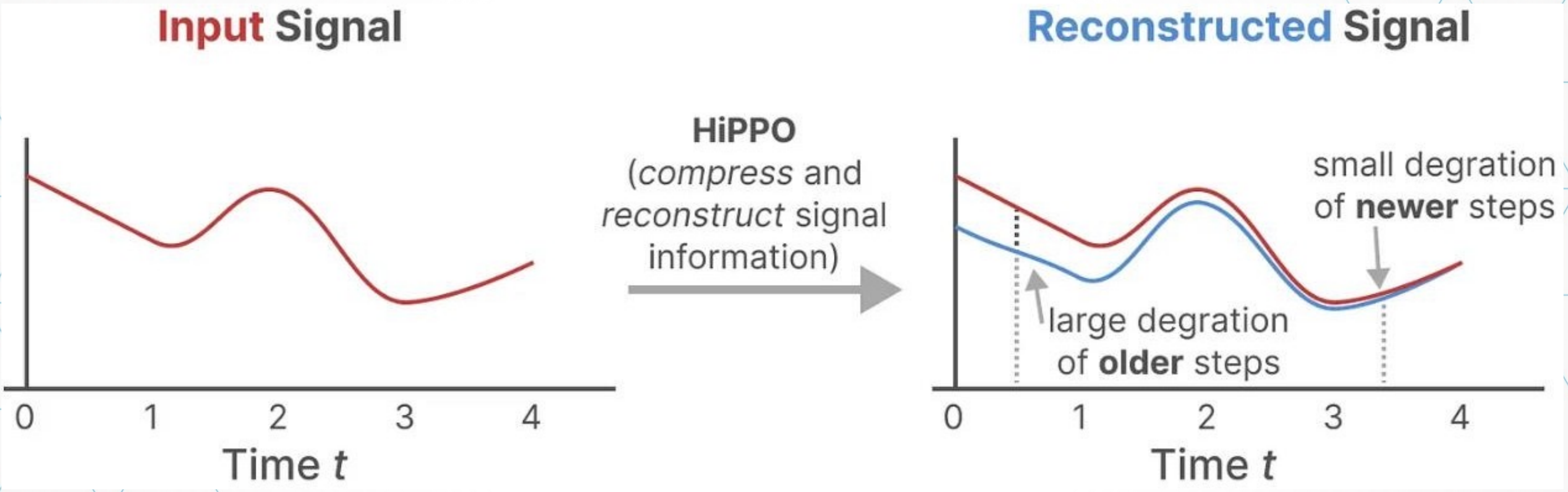
Use **HiPPO matrix**

$$\boldsymbol{A}_{nk} = -\begin{cases} (2n+1)^{1/2}(2k+1)^{1/2} & \text{if } n > k \\ n+1 & \text{if } n = k \\ 0 & \text{if } n < k \end{cases}.$$

HiPPO Matrix

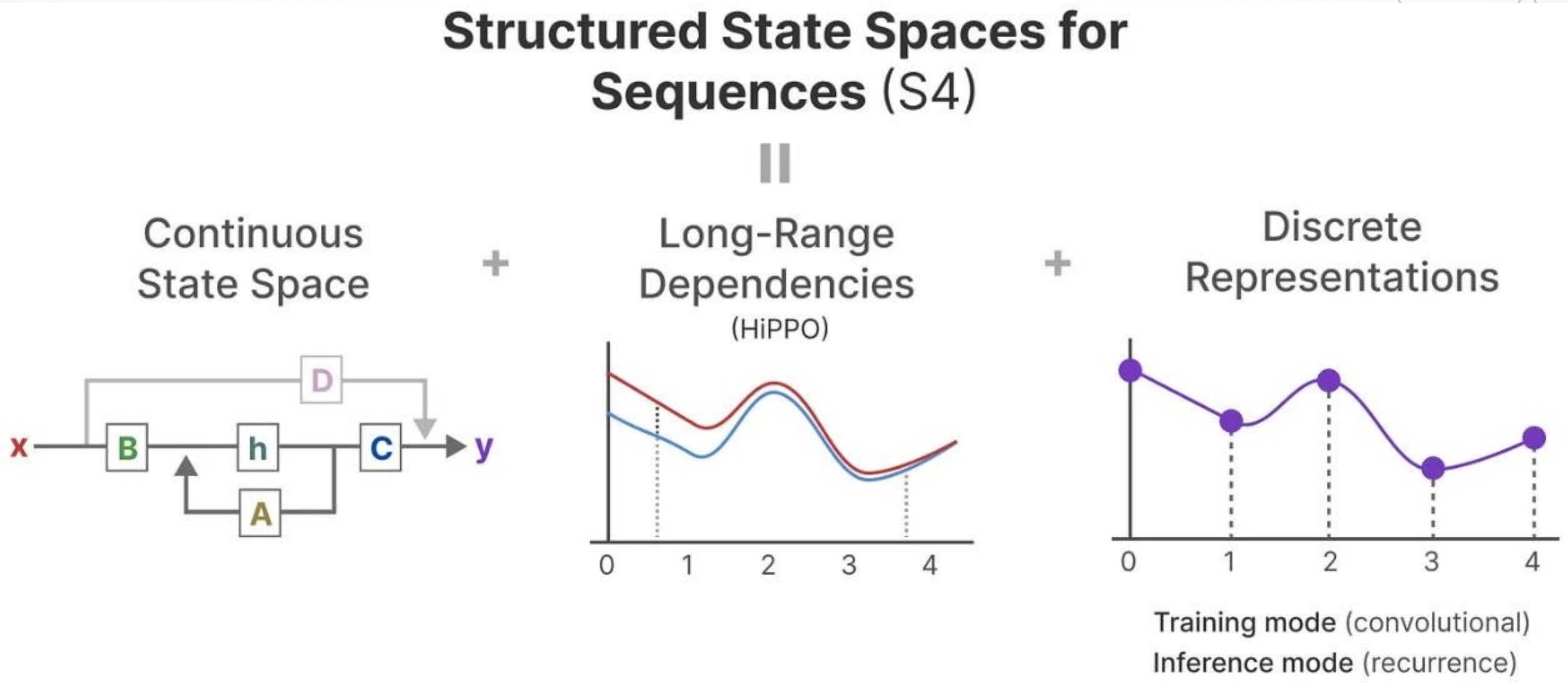
$$
\begin{array}{c|c}
\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ 1 & 3 & 3 & 0 \\ 1 & 3 & 5 & 4 \end{bmatrix} & k \\
n &
\end{array}
$$

HiPPO



Input Signal
0
1
2
3
4
Time t
HiPPO
(compress and reconstruct signal information)
Reconstructed Signal
small degration of newer steps
large degration of older steps
0
1
2
3
4
Time t

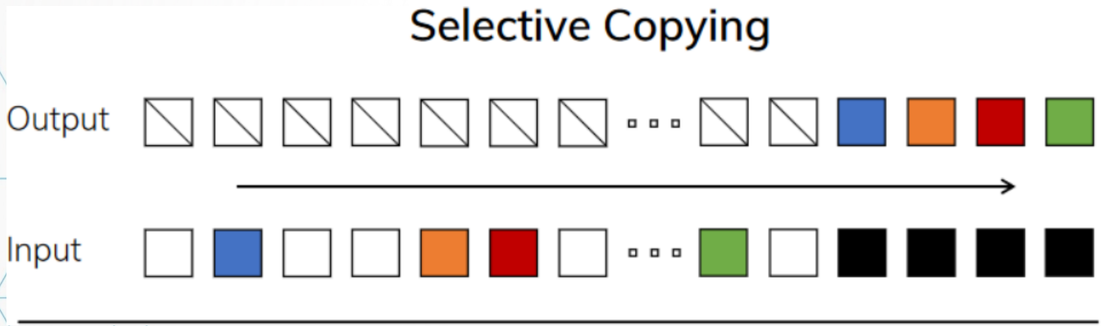
Introduction: State space model



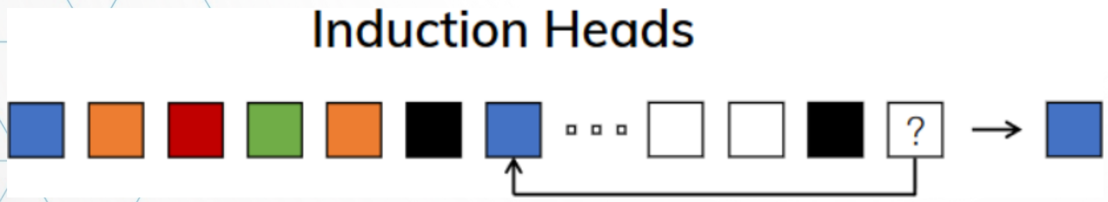
The Mamba model

Mamba model: motivation

S4 still performs **poorly on certain tasks**



given a comment on Twitter, rewrite the comment by removing all the bad words



Few shot prompting:
“1+1 = 2 then 2+2 = ?“

Mamba model: motivation

Why?

Time Invariant: all params A, B, C, D are **fixed for every tokens** it generates

=> **Can not choose** which information it need to focus

But it could be done easily by Transformer: can attend to all previous tokens when generating current one

Mamba model: Selective SSMs

- Incoporate the **sequence length** and **batch size** of the input by a **MLP projection**

Algorithm 1 SSM (S4)

Input: $x : (\mathtt{B}, \mathtt{L}, \mathtt{D})$
Output: $y : (\mathtt{B}, \mathtt{L}, \mathtt{D})$

$\boldsymbol{A} : (\mathtt{D}, \mathtt{N}) \leftarrow \mathsf{Parameter}$
▷ Represents structured $N \times N$ matrix
$\boldsymbol{B} : (\mathtt{D}, \mathtt{N}) \leftarrow \mathsf{Parameter}$
$\boldsymbol{C} : (\mathtt{D}, \mathtt{N}) \leftarrow \mathsf{Parameter}$
$\Delta : (\mathtt{D}) \leftarrow \tau_\Delta(\mathsf{Parameter})$
$\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}} : (\mathtt{D}, \mathtt{N}) \leftarrow \mathsf{discretize}(\Delta, \boldsymbol{A}, \boldsymbol{B})$
$y \leftarrow \mathsf{SSM}(\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}}, \boldsymbol{C})(x)$
▷ Time-invariant: recurrence or convolution
return y

Algorithm 2 SSM + Selection (S6)

Input: $x : (\mathtt{B}, \mathtt{L}, \mathtt{D})$
Output: $y : (\mathtt{B}, \mathtt{L}, \mathtt{D})$

$\boldsymbol{A} : (\mathtt{D}, \mathtt{N}) \leftarrow \mathsf{Parameter}$
▷ Represents structured $N \times N$ matrix
$\boldsymbol{B} : (\mathtt{B}, \mathtt{L}, \mathtt{N}) \leftarrow s_B(x)$
$\boldsymbol{C} : (\mathtt{B}, \mathtt{L}, \mathtt{N}) \leftarrow s_C(x)$
$\Delta : (\mathtt{B}, \mathtt{L}, \mathtt{D}) \leftarrow \tau_\Delta(\mathsf{Parameter} + s_\Delta(x))$
$\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}} : (\mathtt{B}, \mathtt{L}, \mathtt{D}, \mathtt{N}) \leftarrow \mathsf{discretize}(\Delta, \boldsymbol{A}, \boldsymbol{B})$
$y \leftarrow \mathsf{SSM}(\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}}, \boldsymbol{C})(x)$
▷ Time-varying: recurrence (*scan*) only
return y

$$s_B(x) = \mathsf{Linear}_N(x),\ s_C(x) = \mathsf{Linear}_N(x),\ s_\Delta(x) = \mathsf{Broadcast}_D(\mathsf{Linear}_1(x)), \quad \tau_\Delta = \mathsf{softplus}$$

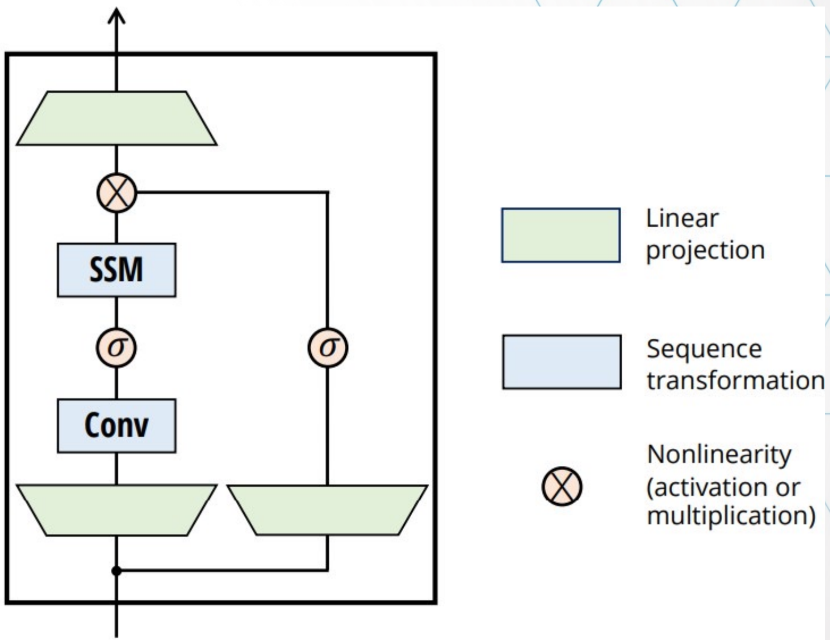
Mamba model: Selective SSMs

- **Connection**: Gating mechanism of RNNs is an instance of selection mechanism in Mamba

When $N = 1, A = -1, B = 1, s = \text{Linear}(x)$,
$t = \text{softplus}$

$$g_t = \sigma(\text{Linear}(x_t))$$
$$h_t = (1 - g_t)h_{t-1} + g_t x_t$$

⇒ When $g_t \to 0$: filter noise,
⇒ When g_t is large: reset the state



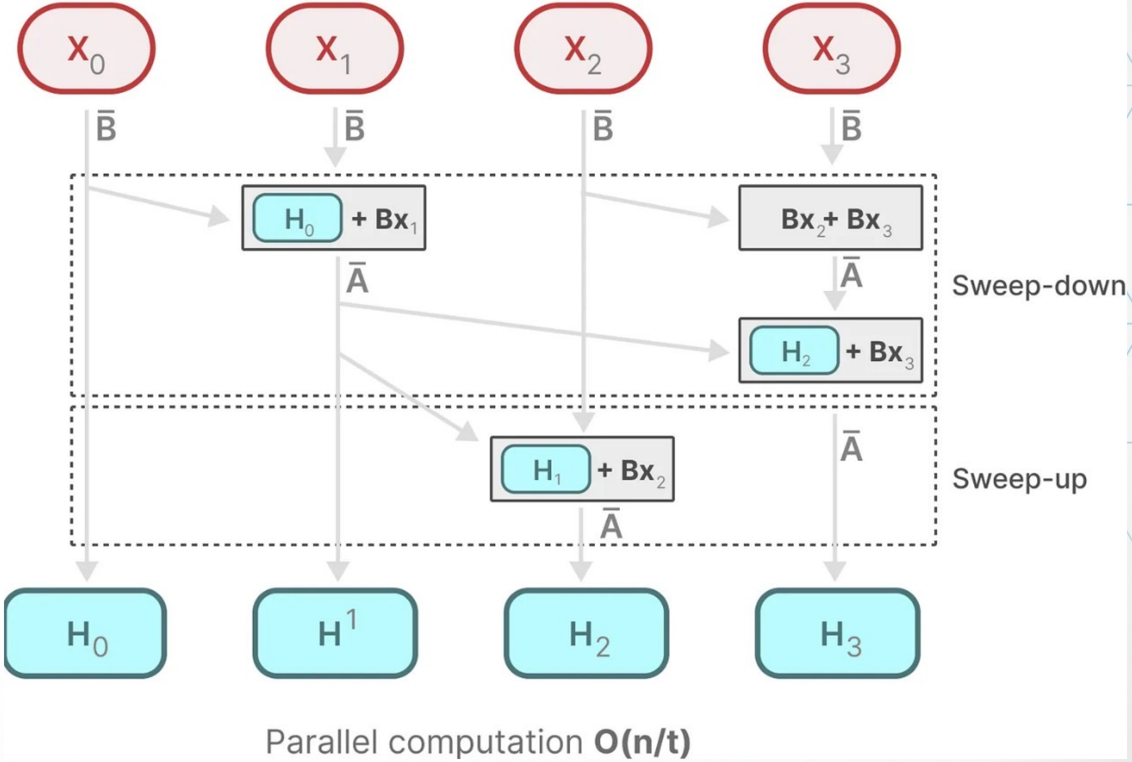
Mamba

Mamba model: Selective SSMs

But now we can't use parallel convolution?

=> Improve with parallel scan algorithm

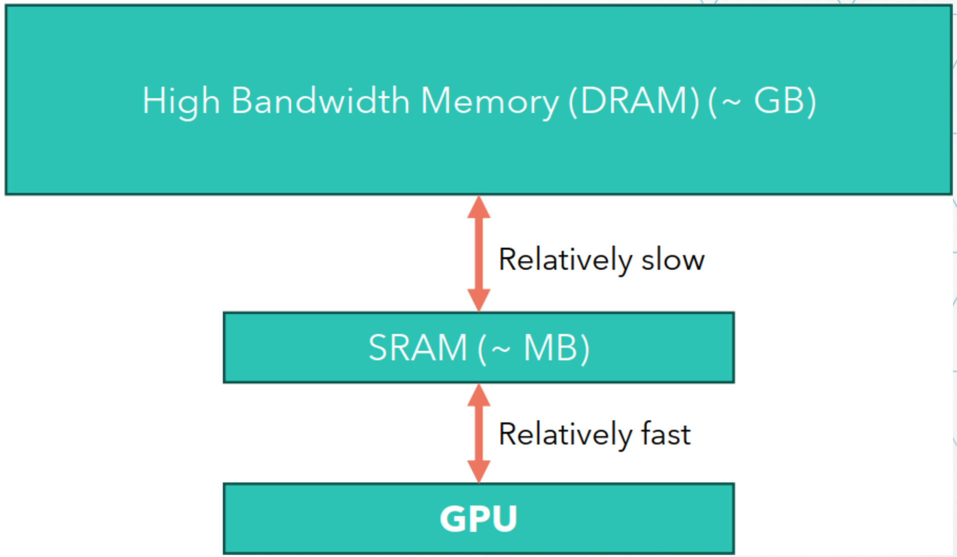
Time complexity: $O(N/T)$



Mamba model: Kernel fusion

Normal:
Load tensor from HBM to SRAM
-> compute -> save back to HBM
for each operations

- Cost time for copying operations
- Fuse CUDA kernels -> one custom CUDA kernel without copying intermediate results



Mamba model: Recomputation

Normal training: Need to cache the outputs of forward steps to perform backpropagation

$\Rightarrow$ Need to save to HBM and copy back -> **Slow**

$\Rightarrow$ Just **recompute** them during backpropagation!

Mamba model: Performance

Model	Arch.	Layer	Acc.
S4	No gate	S4	18.3
-	No gate	S6	**97.0**
H3	H3	S4	57.0
Hyena	H3	Hyena	30.1
-	H3	S6	**99.7**
-	Mamba	S4	56.4
-	Mamba	Hyena	28.4
Mamba	Mamba	S6	**99.8**

Table 1: (**Selective Copying**.) Accuracy for combinations of architectures and inner sequence layers.

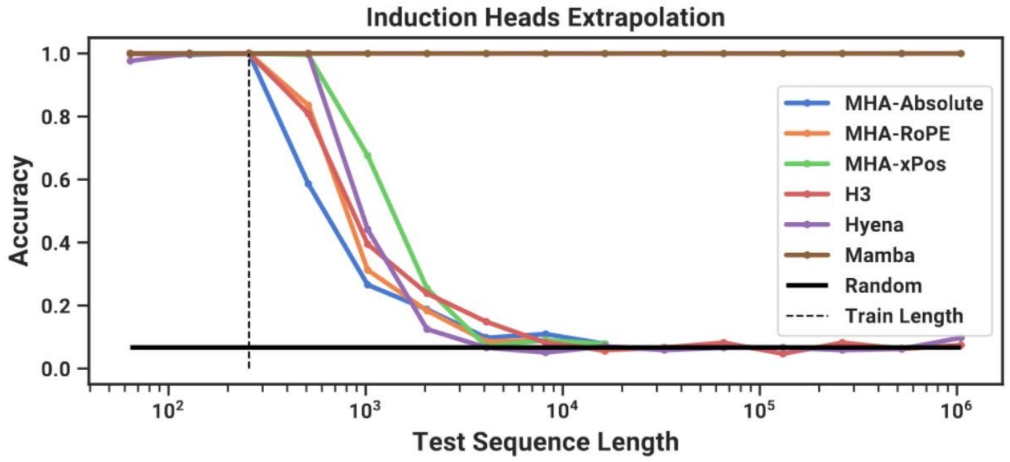


Table 2: (**Induction Heads**.) Models are trained on sequence length $2^8 = 256$, and tested on increasing sequence lengths of $2^6 = 64$ up to $2^{20} = 1048576$. Full numbers in Table 11.

Mamba model: Performance

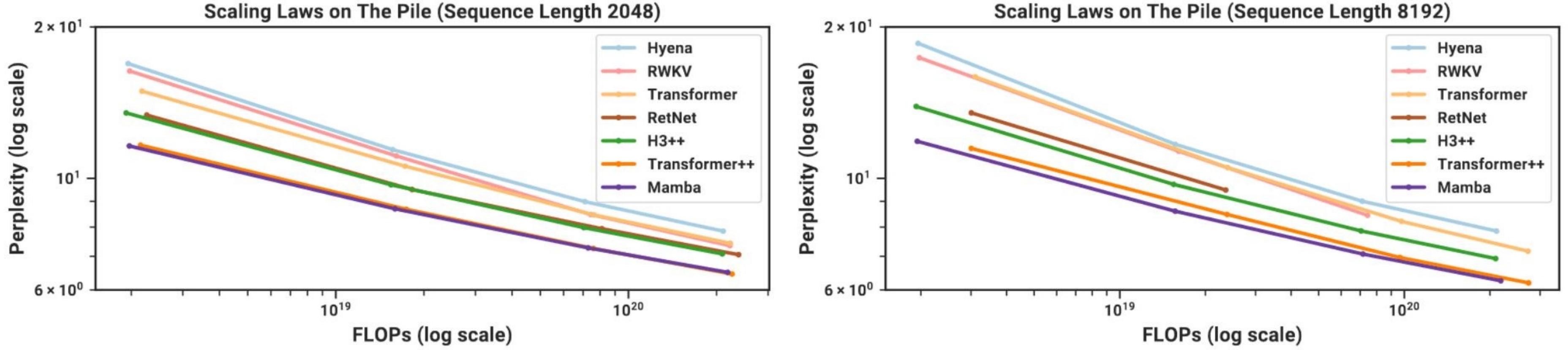


Figure 4: (**Scaling Laws**.) Models of size ≈ 125M to ≈ 1.3B parameters, trained on the Pile. Mamba scales better than all other attention-free models and is the first to match the performance of a very strong "Transformer++" recipe that has now become standard, particularly as the sequence length grows.

Thank you for your attention!

https://www.ura.hcmut.edu.vn/