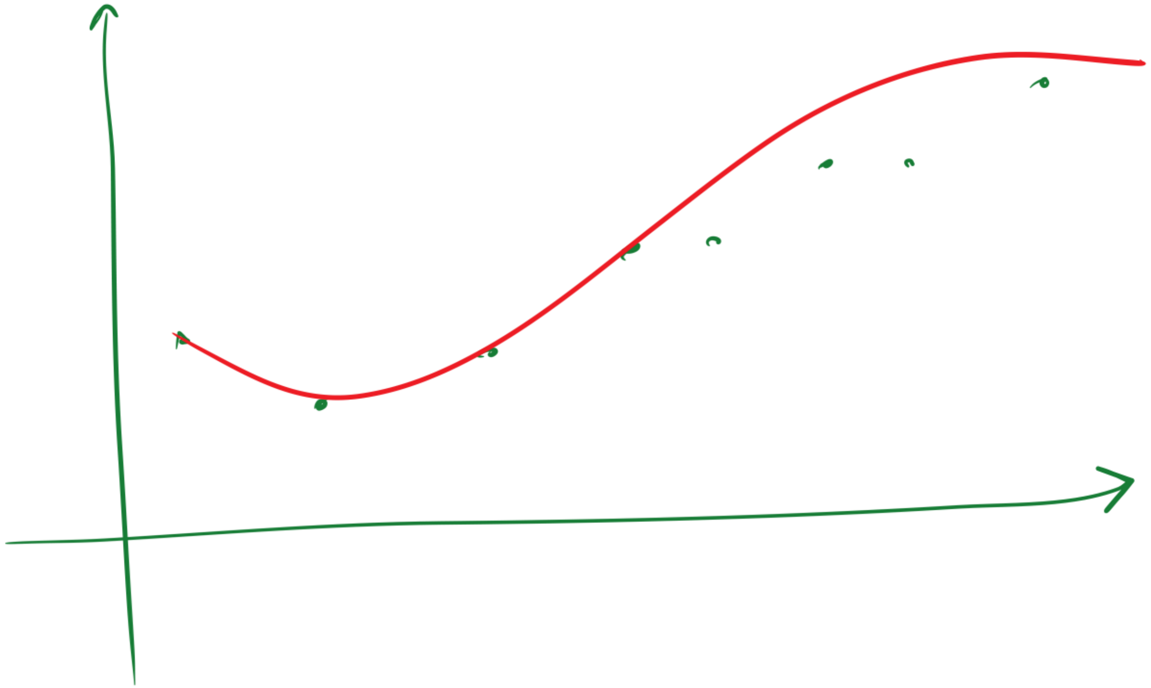
Time Series



Regression

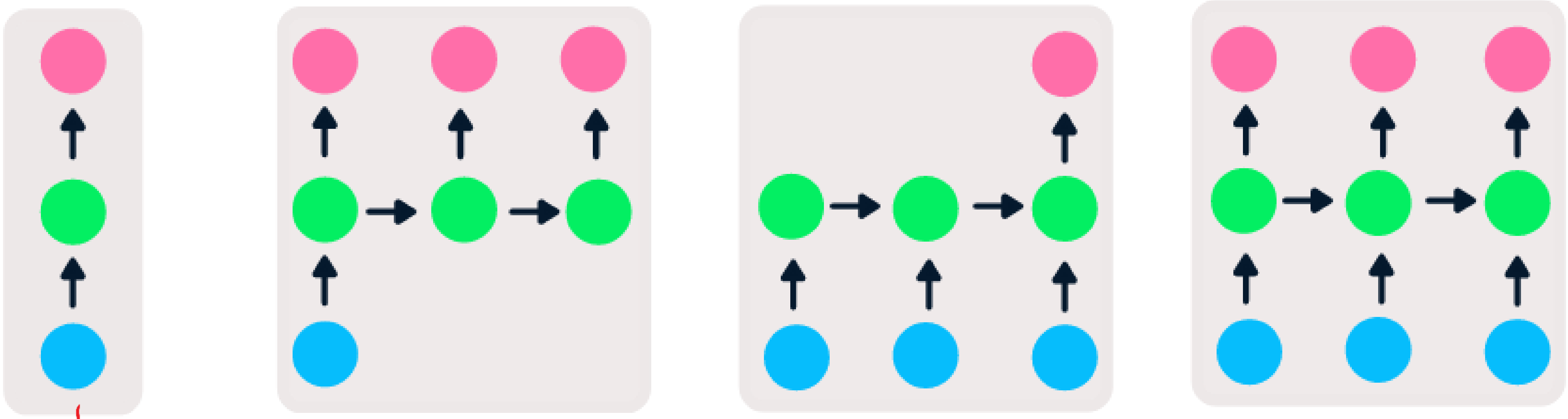
Time Series

RNN

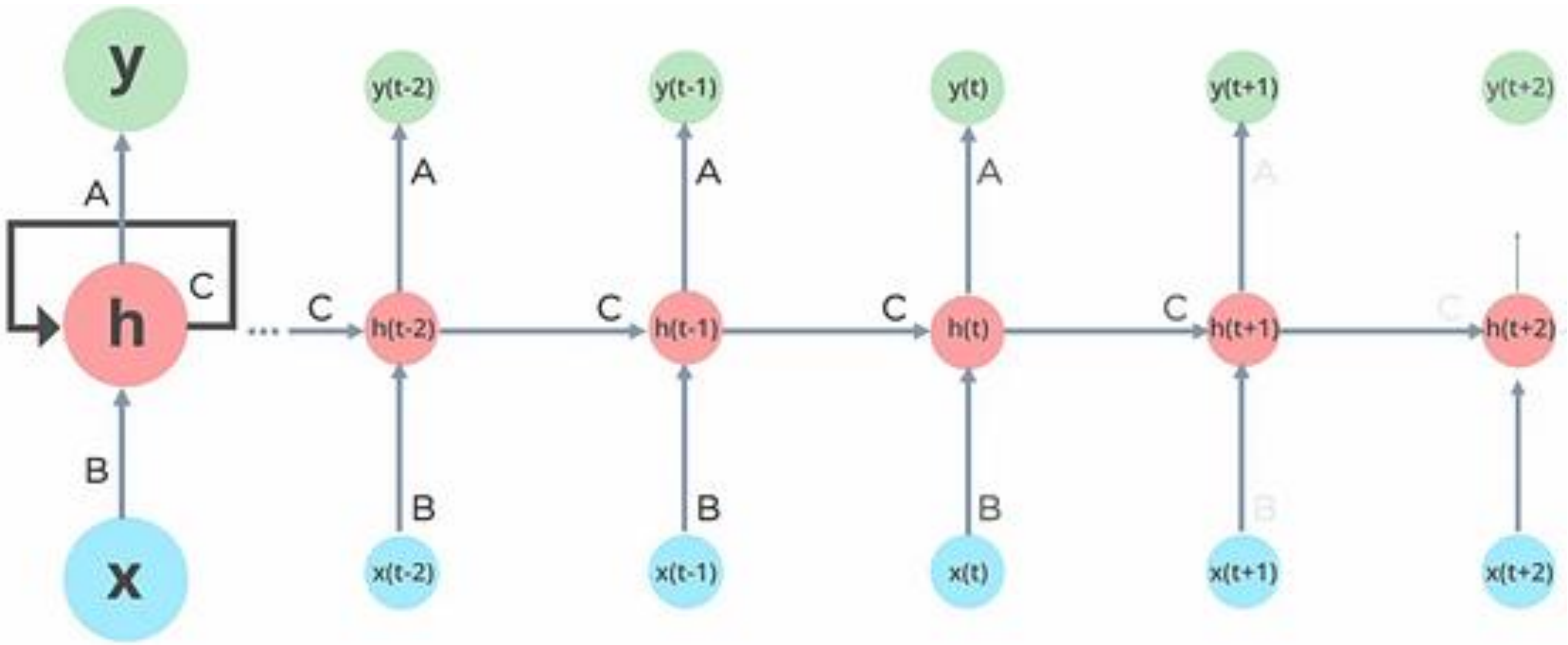
Recurrent NN 1993

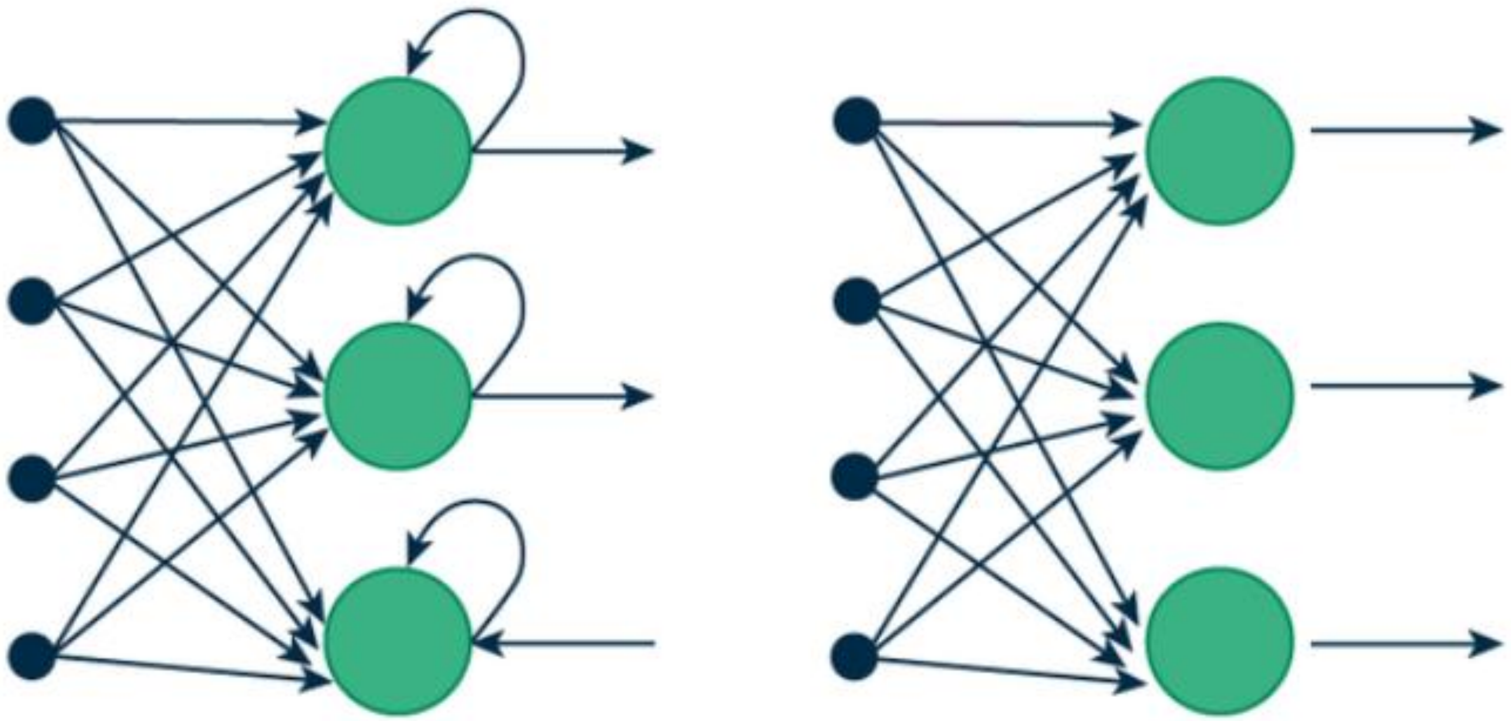
image Captioning text Classification Translate

One to One | One to Many | Many to One | Many to Many

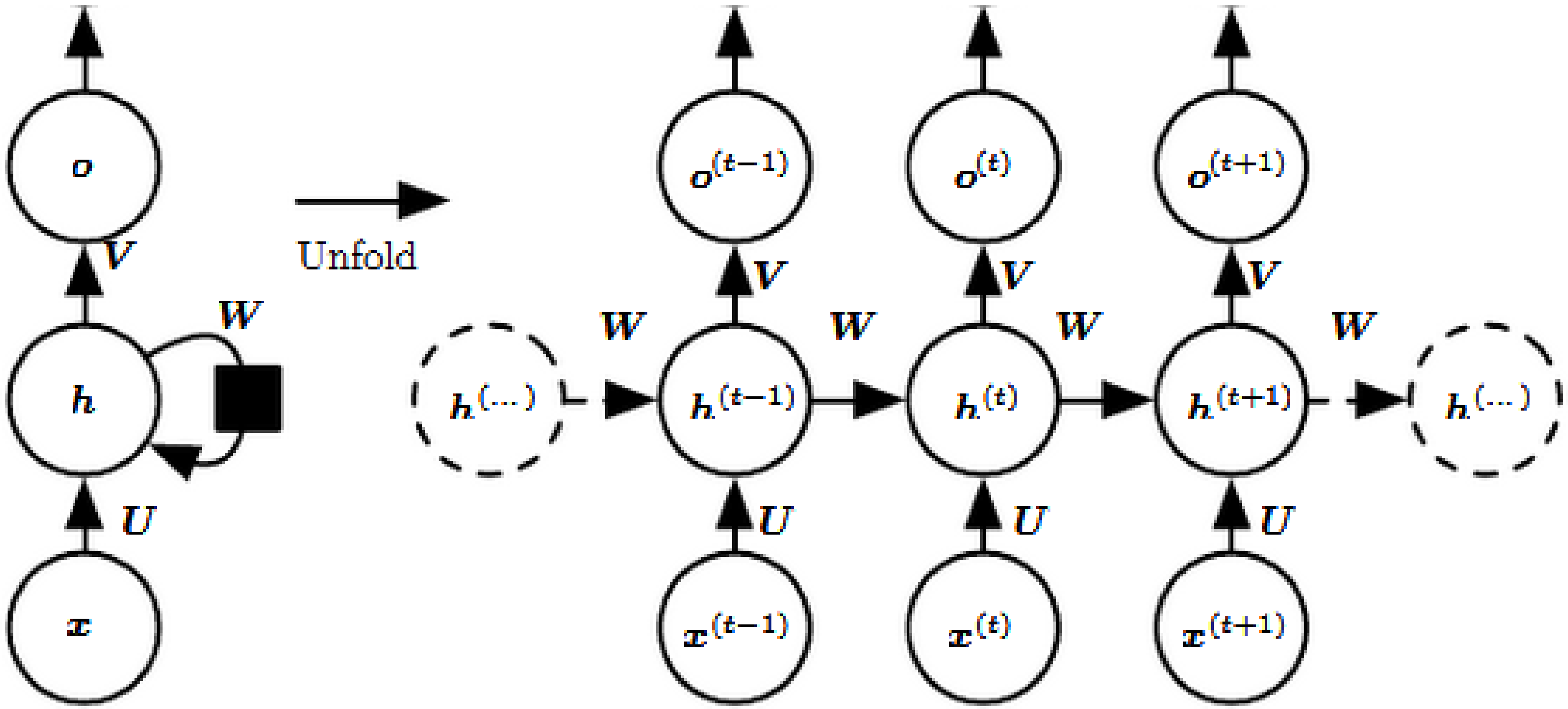


RNN



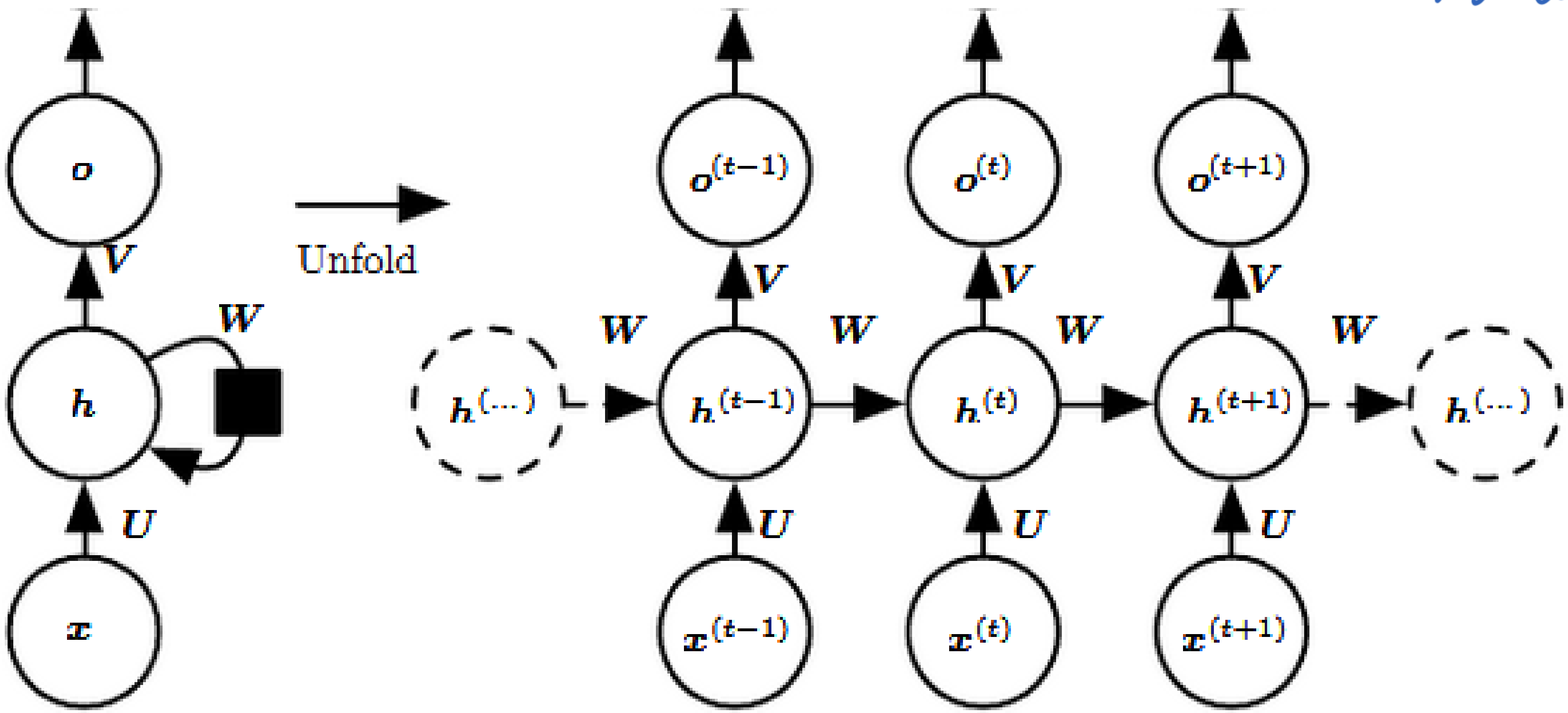


(a) Recurrent Neural Network (b) Feed-Forward Neural Network



o
V
W
h
U
x
Unfold
$o^{(t-1)}$
$o^{(t)}$
$o^{(t+1)}$
V
V
V
W
W
W
W
$h^{(\dots)}$
$h^{(t-1)}$
$h^{(t)}$
$h^{(t+1)}$
$h^{(\dots)}$
U
U
U
$x^{(t-1)}$
$x^{(t)}$
$x^{(t+1)}$

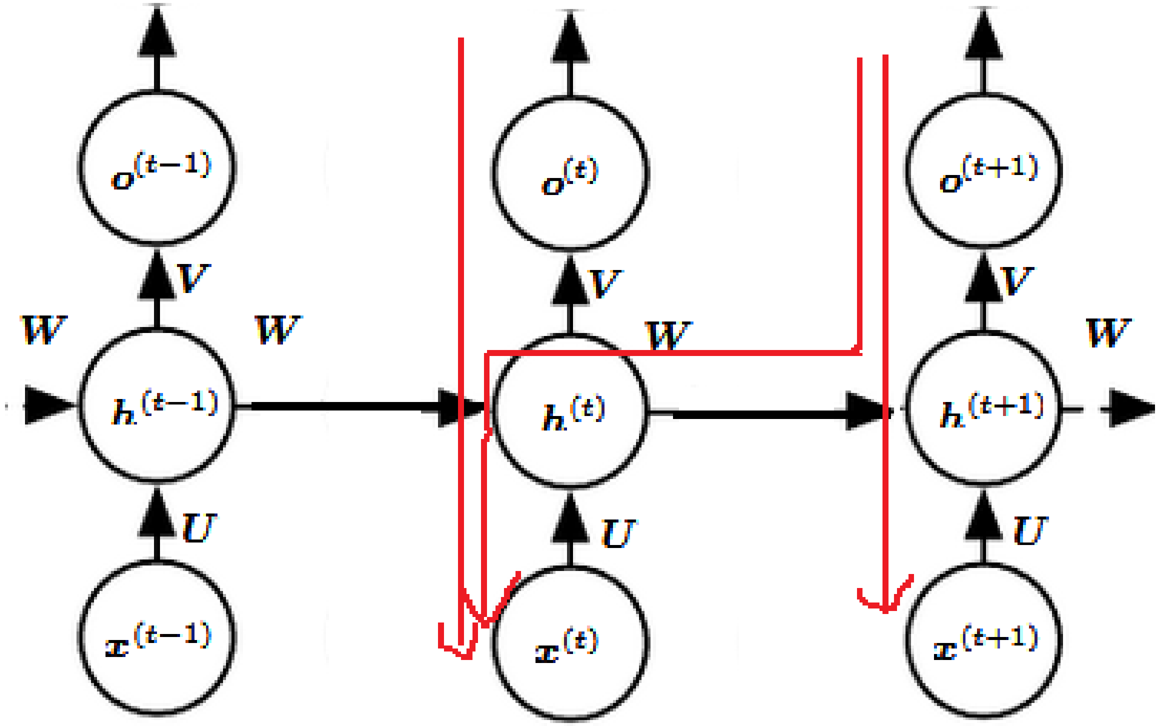
forward



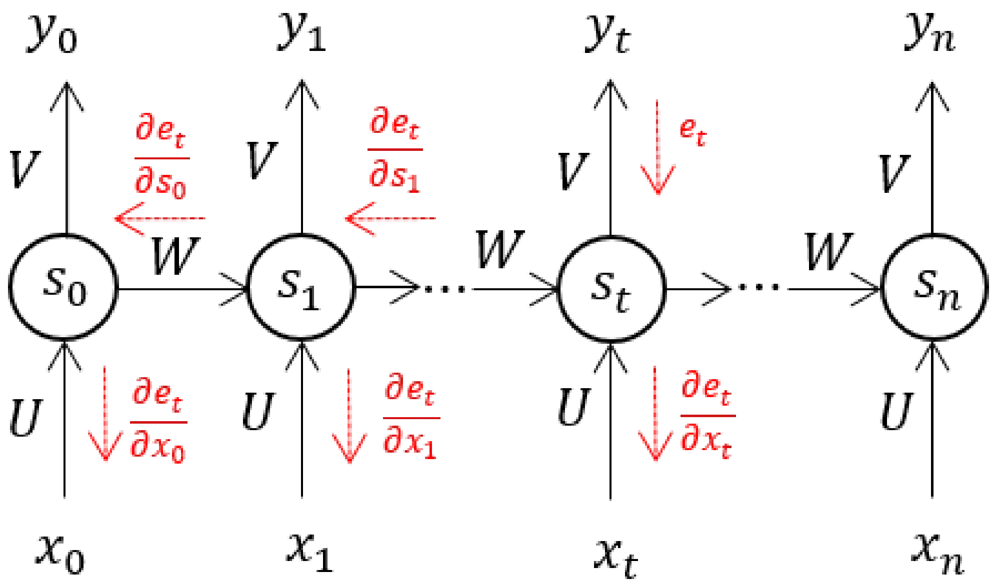
$$
\begin{aligned}
\boldsymbol{a}^{(t)} &= \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)} \\
\boldsymbol{h}^{(t)} &= \tanh(\boldsymbol{a}^{(t)}) \\
\boldsymbol{o}^{(t)} &= \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)} \\
\hat{\boldsymbol{y}}^{(t)} &= \mathrm{softmax}(\boldsymbol{o}^{(t)})
\end{aligned}
$$

$${}_{\mathrm{units}}n + ({}_{\mathrm{units}}n \times {}_{\mathrm{units}}n) + ({}_{\mathrm{units}}n \times {}_{\mathrm{input}}n)$$

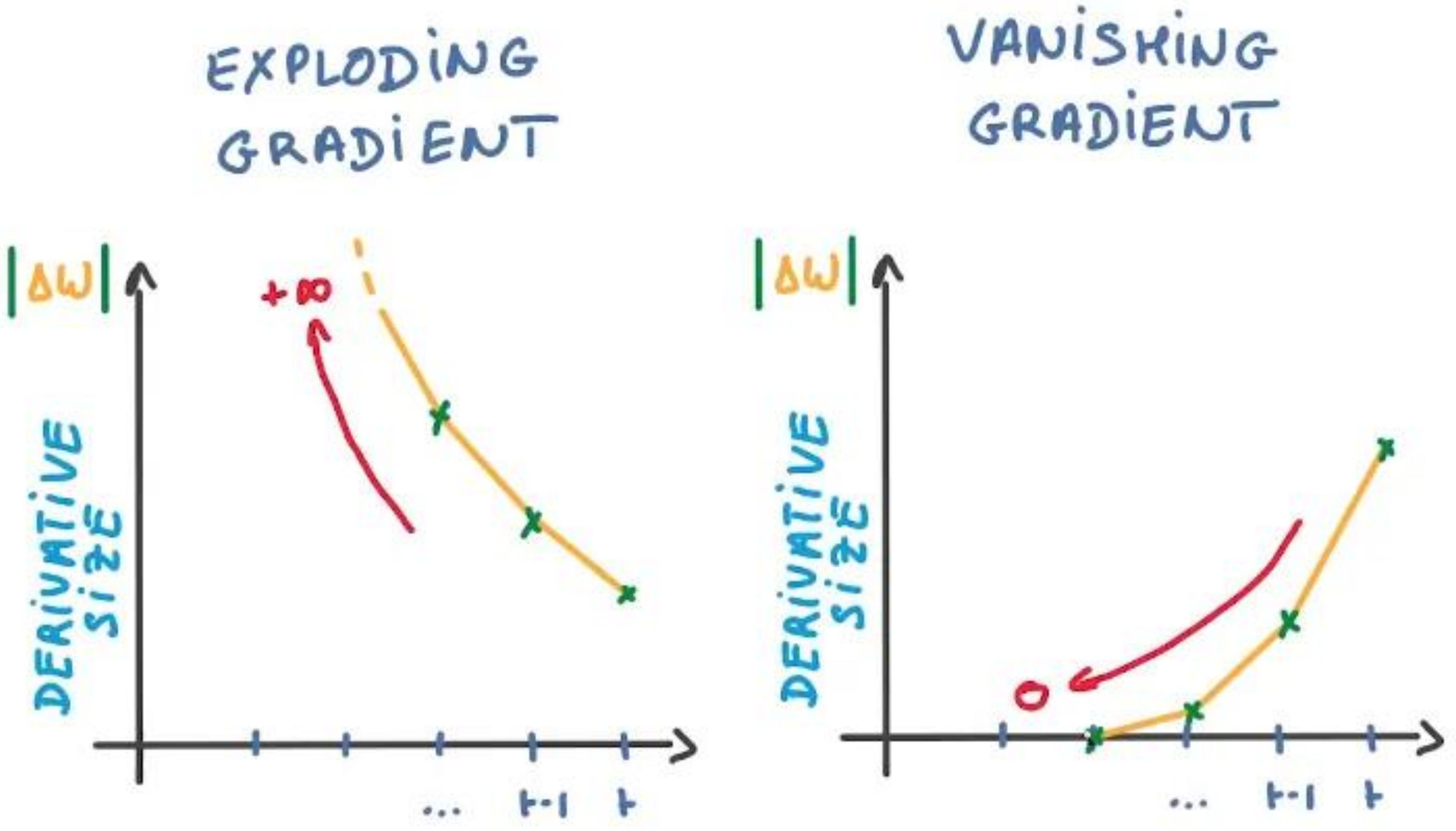
Train (Back propagation)



$$
\begin{aligned}
\nabla_{c} L &= \sum_t \left(\frac{\partial o^{(t)}}{\partial c}\right)^{\top} \nabla_{o^{(t)}} L = \sum_t \nabla_{o^{(t)}} L \\
\nabla_{b} L &= \sum_t \left(\frac{\partial h^{(t)}}{\partial b^{(t)}}\right)^{\top} \nabla_{h^{(t)}} L = \sum_t \operatorname{diag}\left(1-\left(h^{(t)}\right)^2\right) \nabla_{h^{(t)}} L \\
\nabla_{V} L &= \sum_t \sum_i \left(\frac{\partial L}{\partial o_i^{(t)}}\right) \nabla_{V^{(t)}} o_i^{(t)} = \sum_t \left(\nabla_{o^{(t)}} L\right) h^{(t)^{\top}} \\
\nabla_{W} L &= \sum_t \sum_i \left(\frac{\partial L}{\partial h_i^{(t)}}\right) \nabla_{W^{(t)}} h_i^{(t)} = \sum_t \operatorname{diag}\left(1-\left(h^{(t)}\right)^2\right) \left(\nabla_{h^{(t)}} L\right) h^{(t-1)^{\top}} \\
\nabla_{U} L &= \sum_t \sum_i \left(\frac{\partial L}{\partial h_i^{(t)}}\right) \nabla_{U^{(t)}} h_i^{(t)} = \sum_t \operatorname{diag}\left(1-\left(h^{(t)}\right)^2\right) \left(\nabla_{h^{(t)}} L\right) x^{(t)^{\top}}
\end{aligned}
$$



y_0
y_1
y_t
y_n
V
V
V
V
$\frac{\partial e_t}{\partial s_0}$
$\frac{\partial e_t}{\partial s_1}$
e_t
W
W
W
s_0
s_1
s_t
s_n
U
U
U
U
$\frac{\partial e_t}{\partial x_0}$
$\frac{\partial e_t}{\partial x_1}$
$\frac{\partial e_t}{\partial x_t}$
x_0
x_1
x_t
x_n


EXPLODING GRADIENT
VANISHING GRADIENT
|ΔW|
DERIVATIVE SIZE
+∞
0
...
t-1
t

This is a classroom

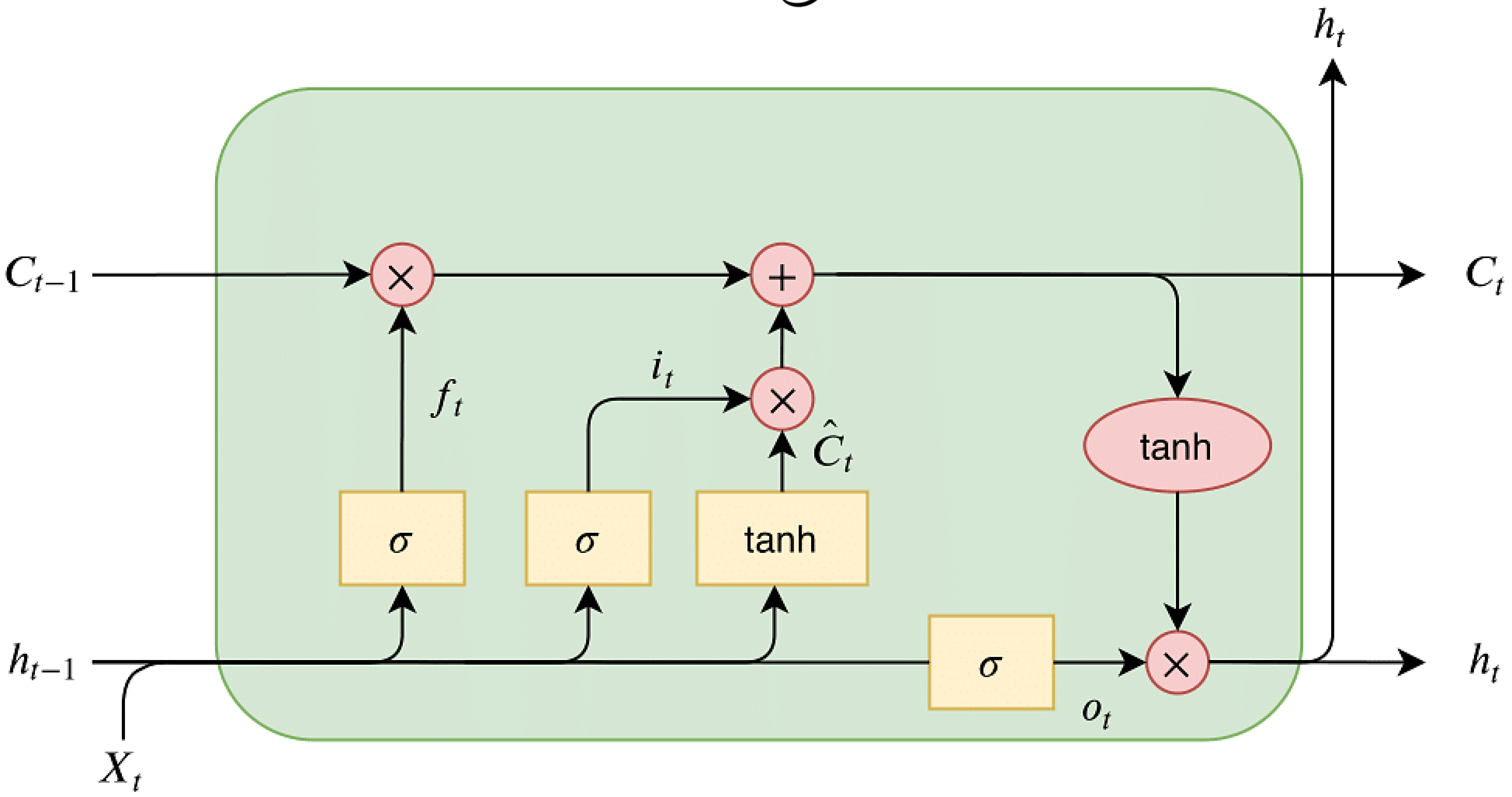
T h i → s

h i s → ' '

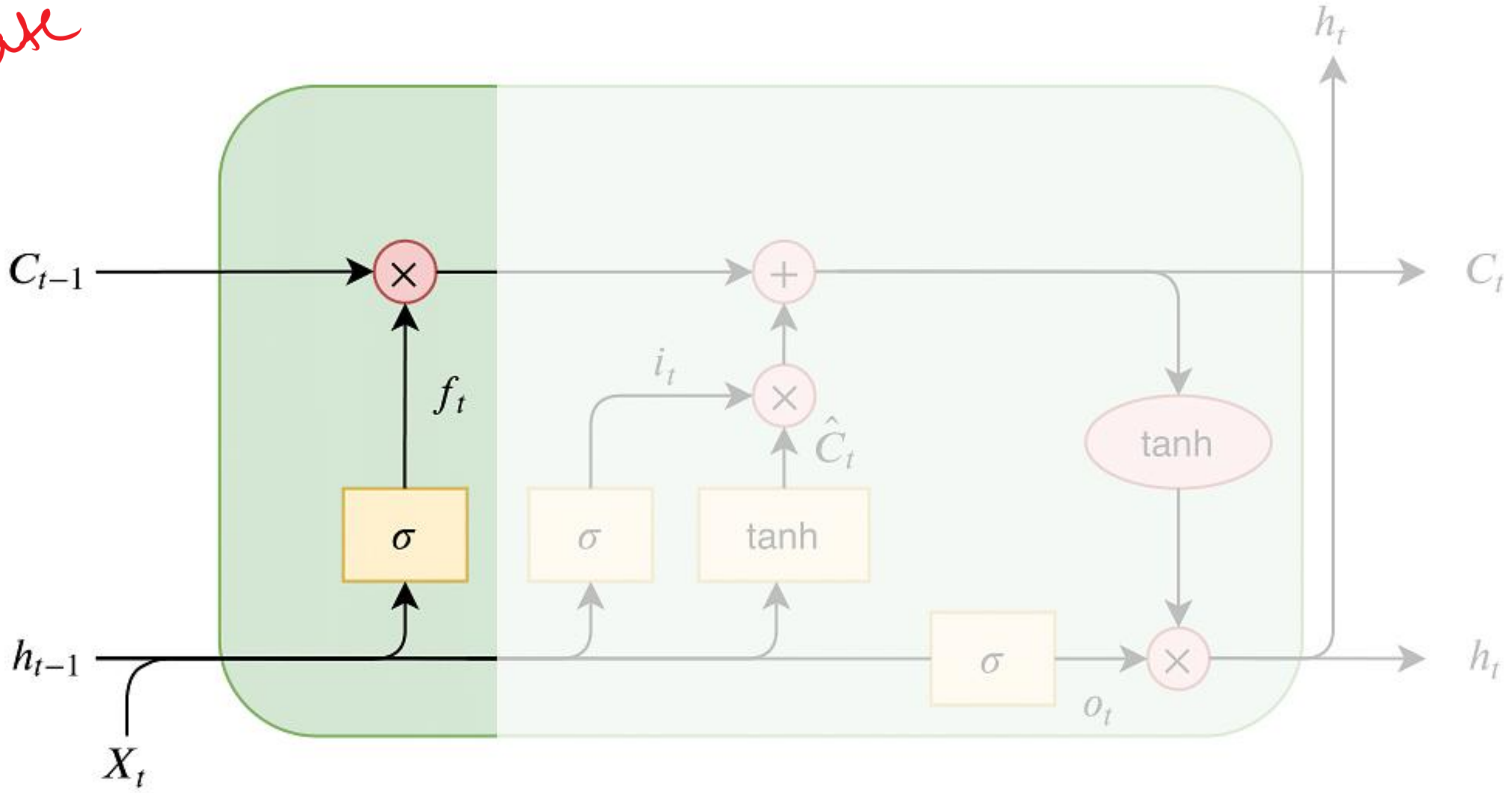
i s → i

1997

LSTM long short term memory

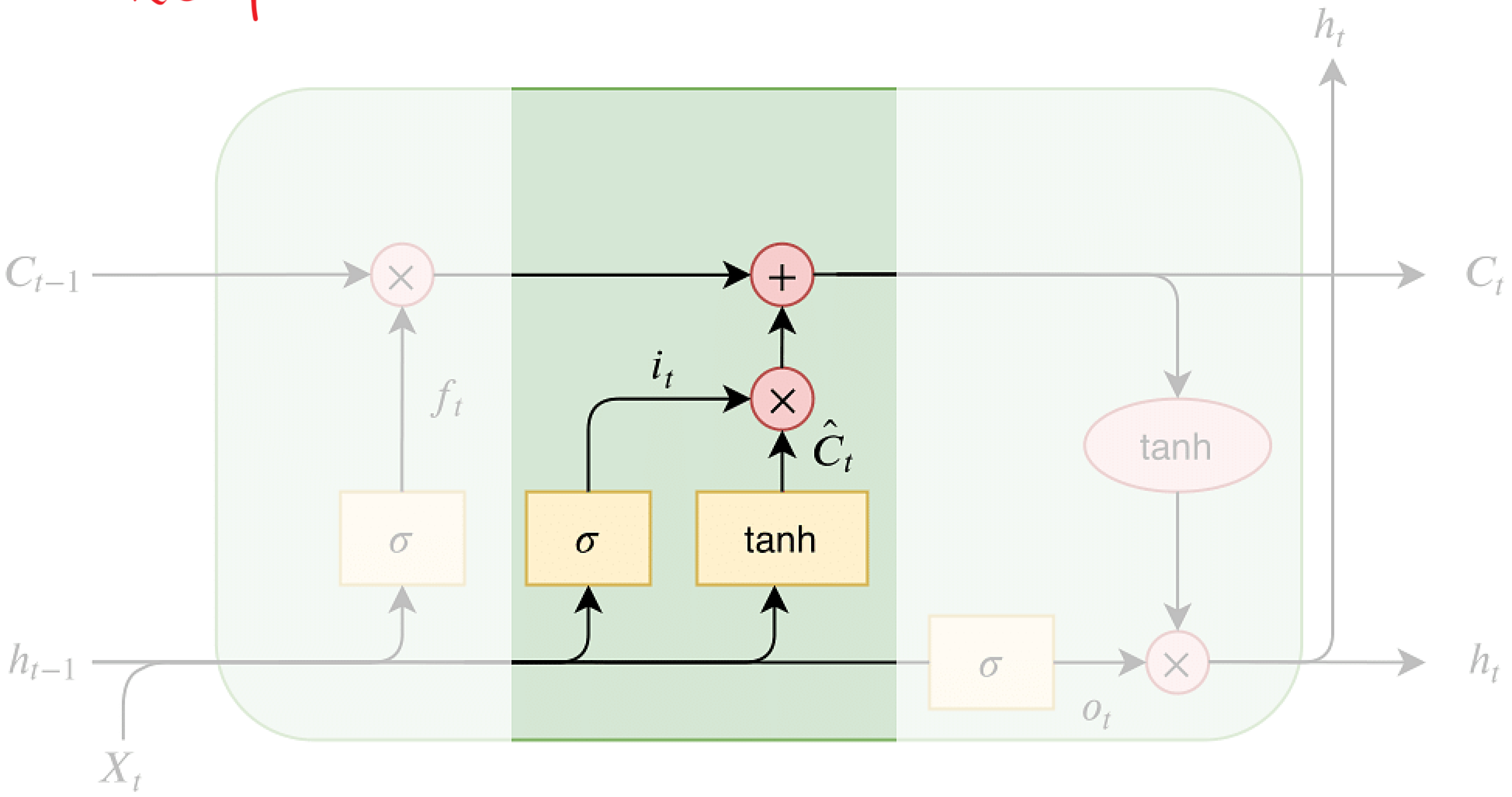


forget gate

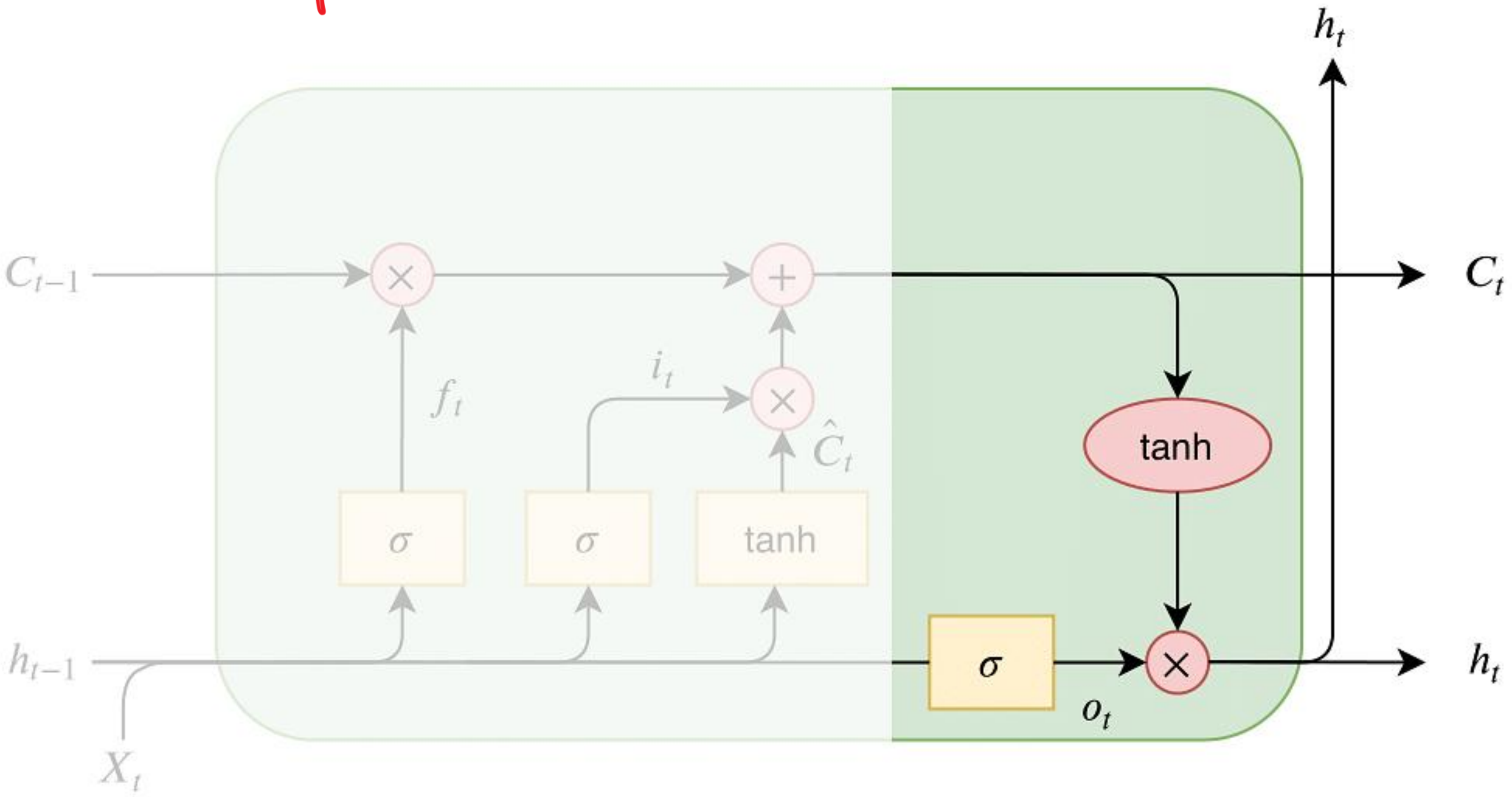


update

keep

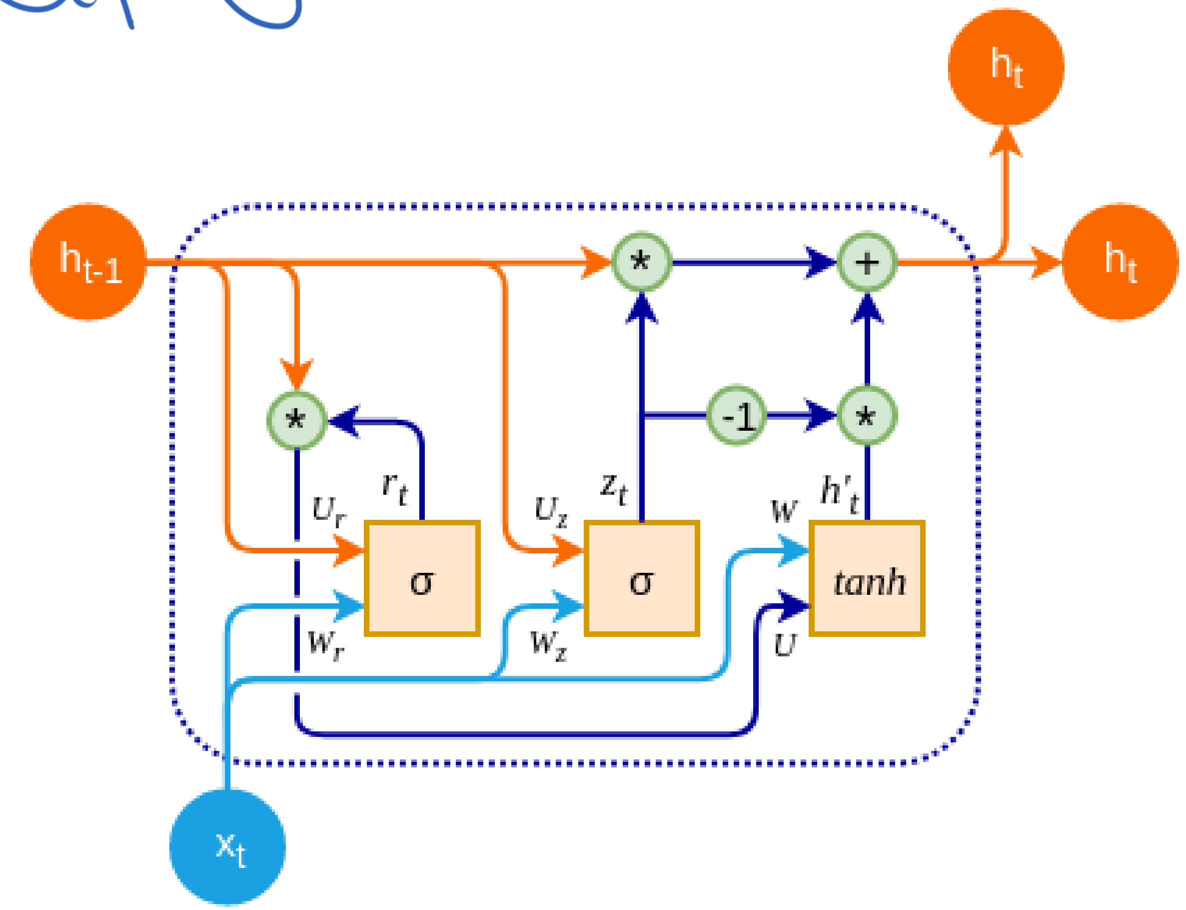


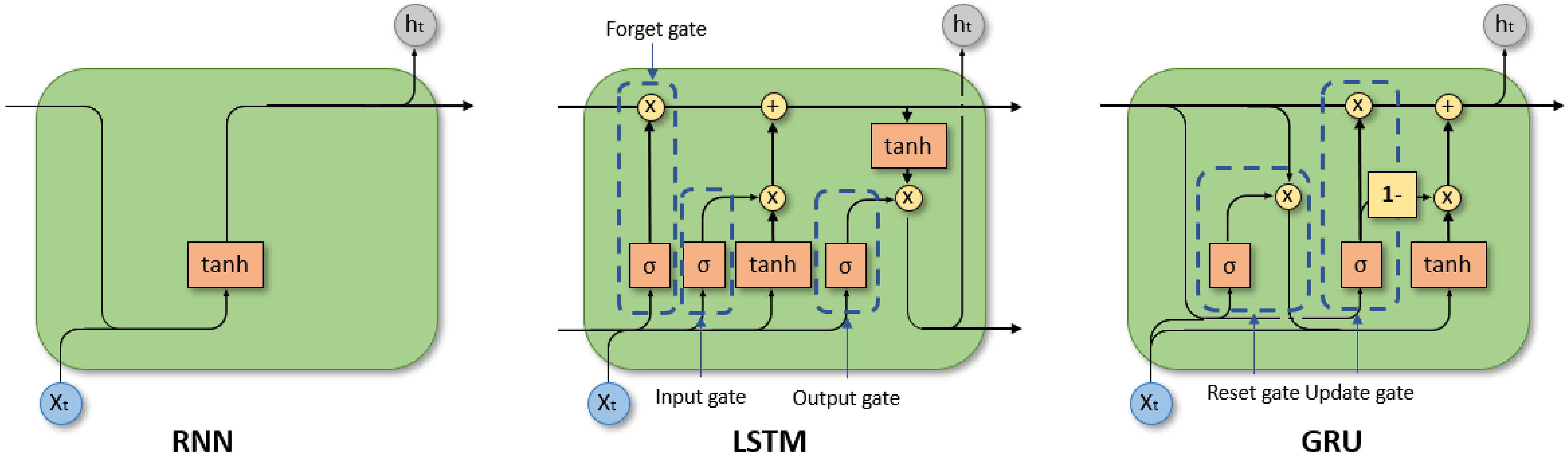
Output

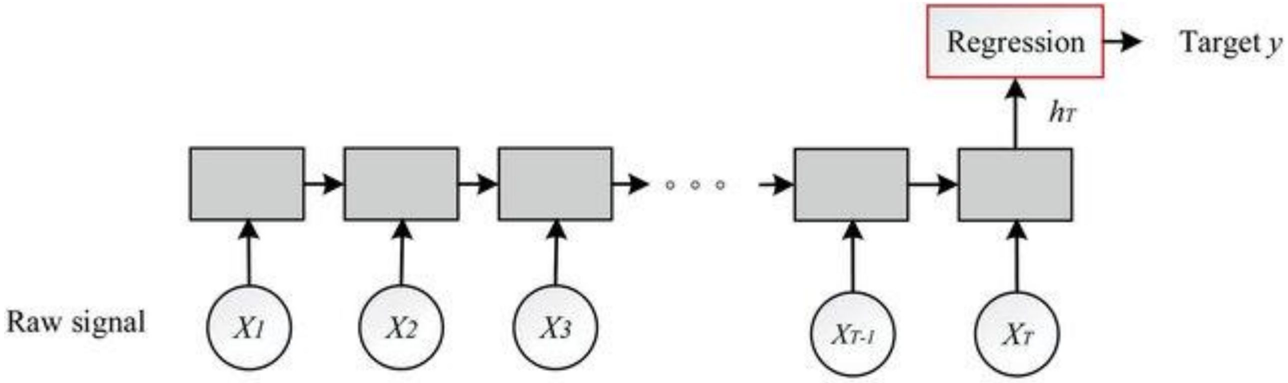


2014

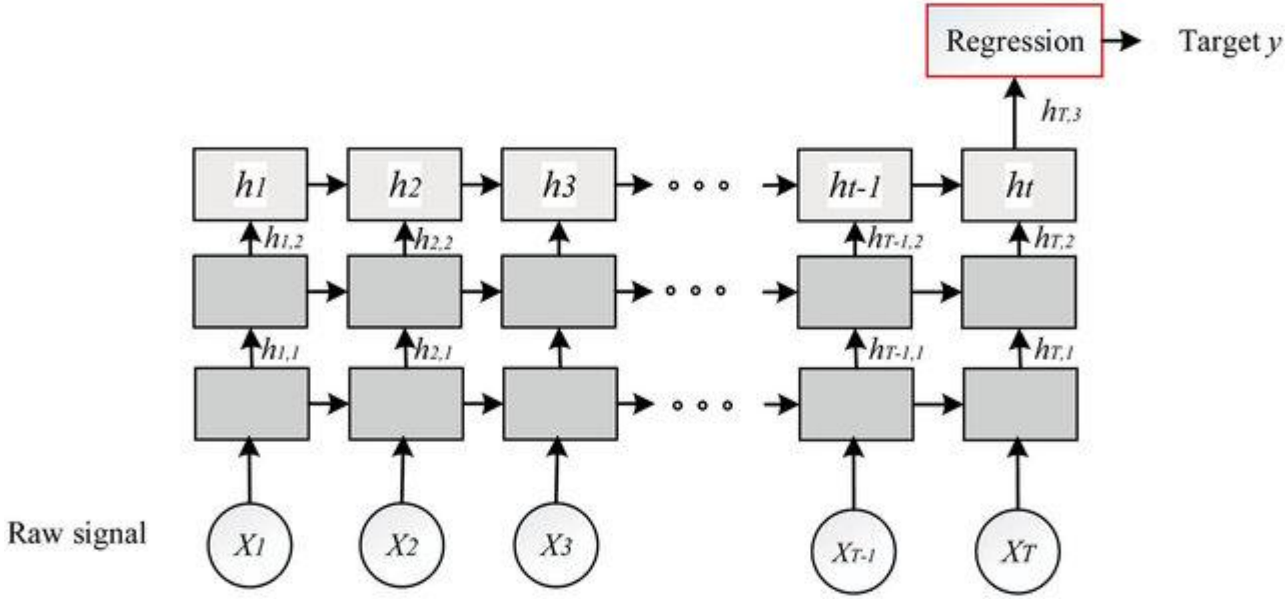
GRU



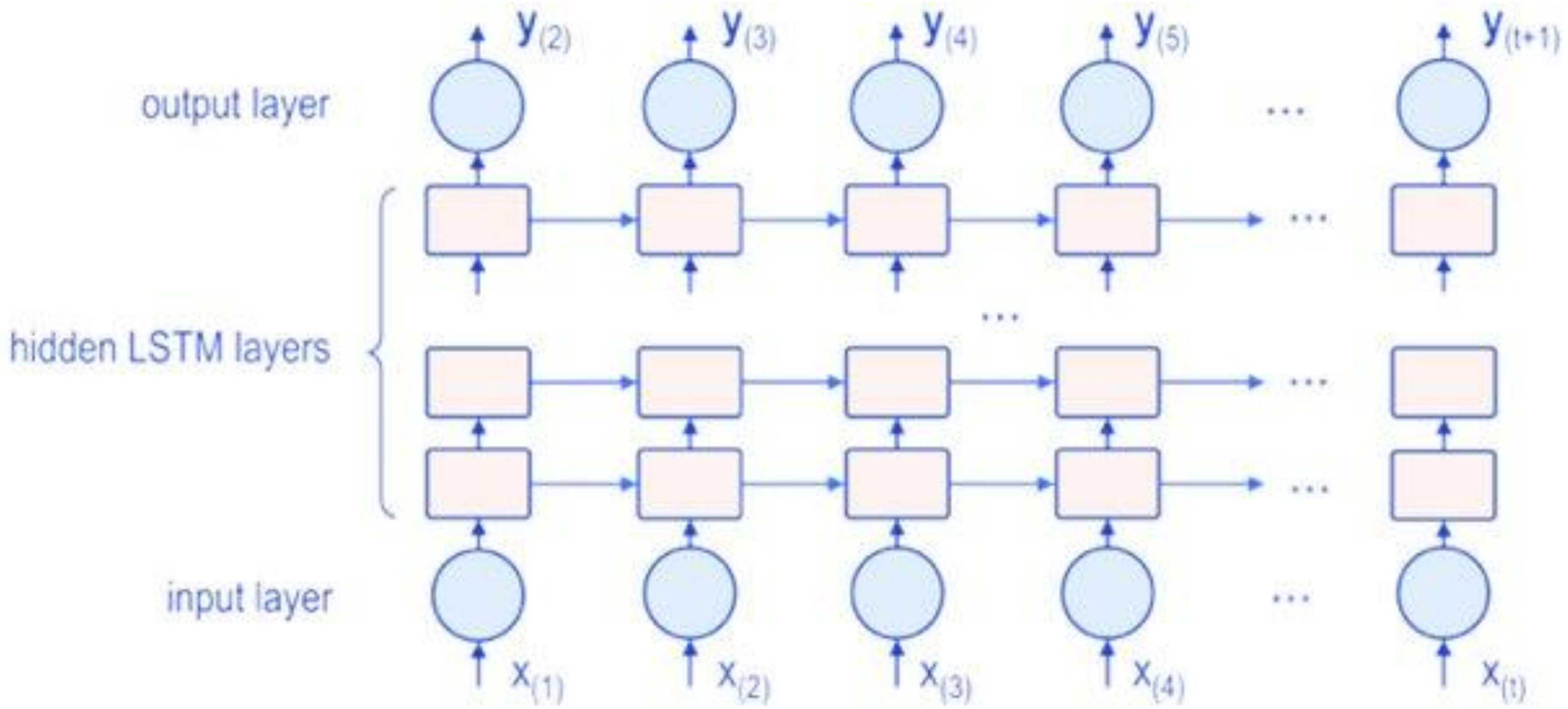

ht
tanh
Xt
RNN
Forget gate
ht
x
+
tanh
x
x
σ
σ
tanh
σ
Xt
Input gate
Output gate
LSTM
ht
x
+
1-
x
x
σ
σ
tanh
Xt
Reset gate
Update gate
GRU


Regression
Target y
h_T
Raw signal
X_1
X_2
X_3
X_{T-1}
X_T

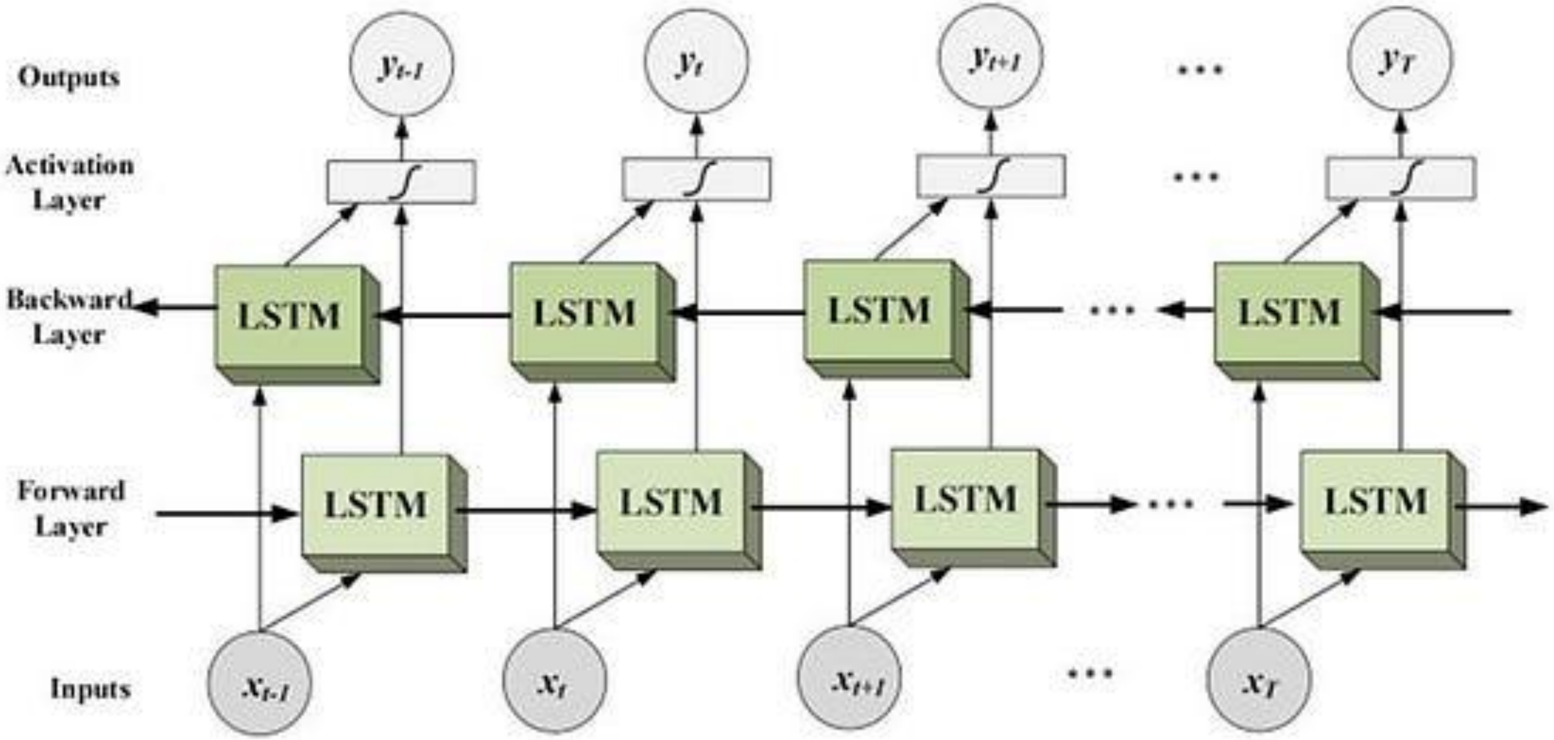
(a)


Regression
Target y
$h_{T,3}$
h_1
h_2
h_3
h_{t-1}
h_t
$h_{1,2}$
$h_{2,2}$
$h_{T-1,2}$
$h_{T,2}$
$h_{1,1}$
$h_{2,1}$
$h_{T-1,1}$
$h_{T,1}$
Raw signal
X_1
X_2
X_3
X_{T-1}
X_T

(b)


y(2)
y(3)
y(4)
y(5)
y(t+1)
output layer
...
hidden LSTM layers
input layer
x(1)
x(2)
x(3)
x(4)
x(t)

Biderectional

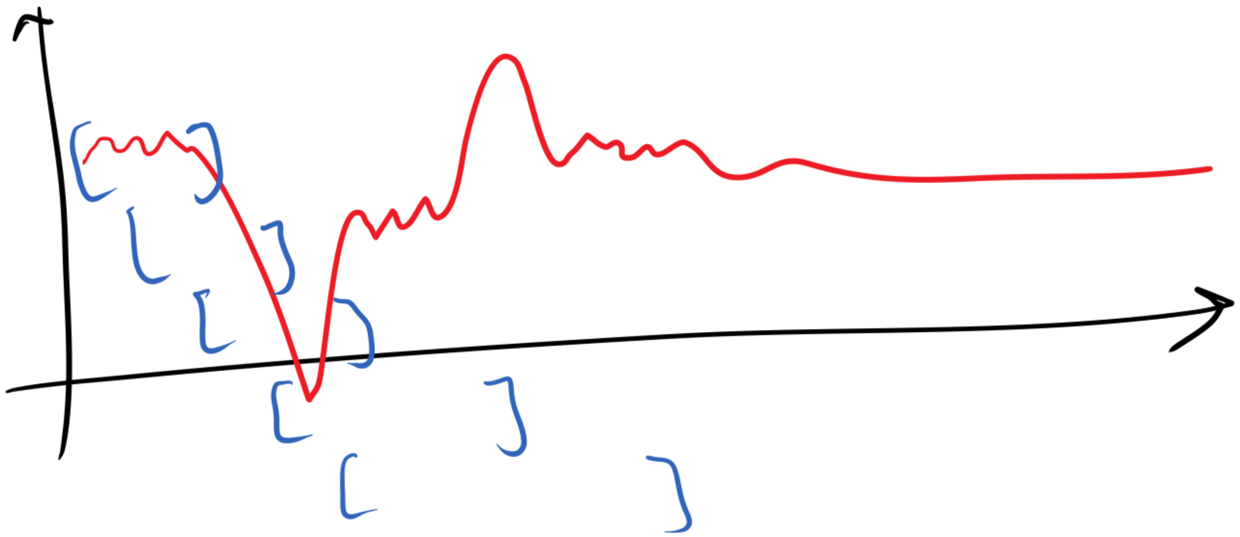


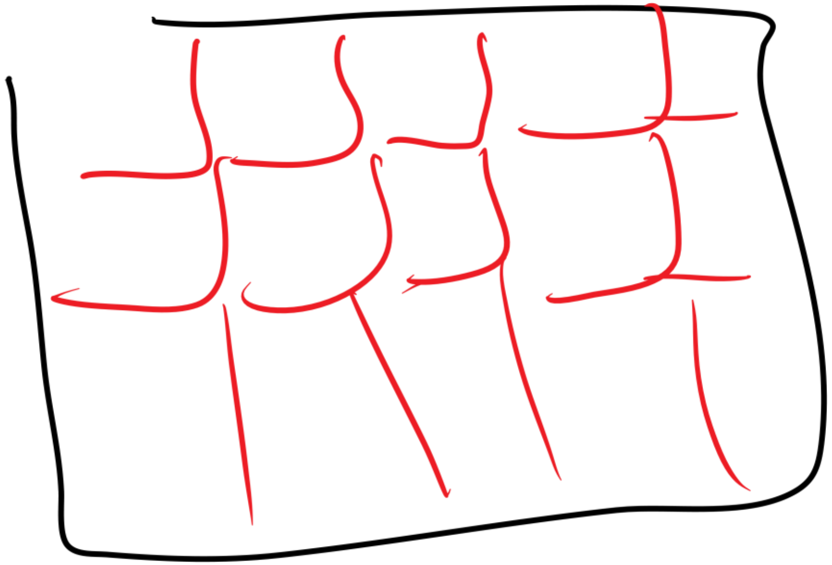
CNN + RNN

LSTM

کپشن زدن

Tracking



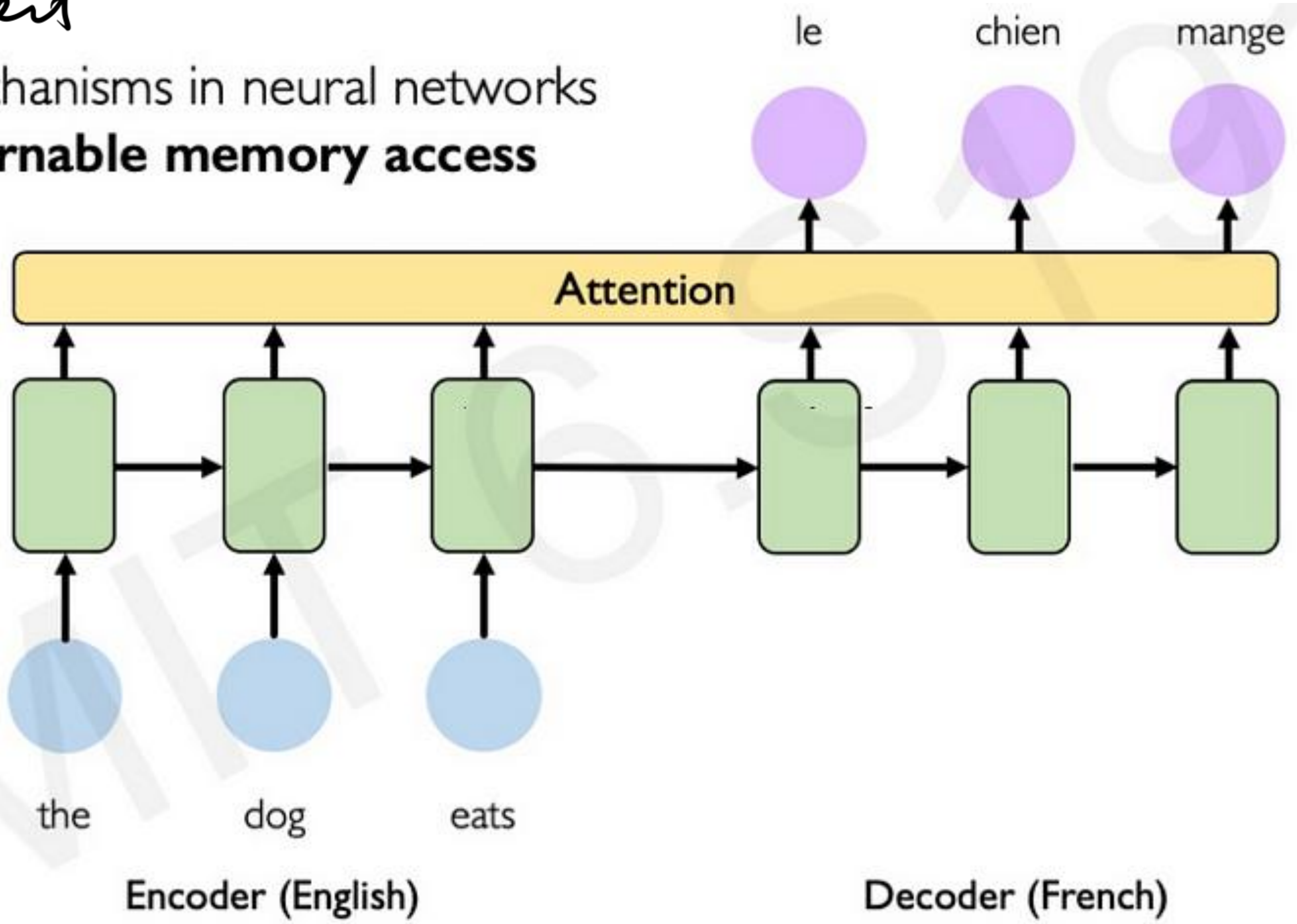


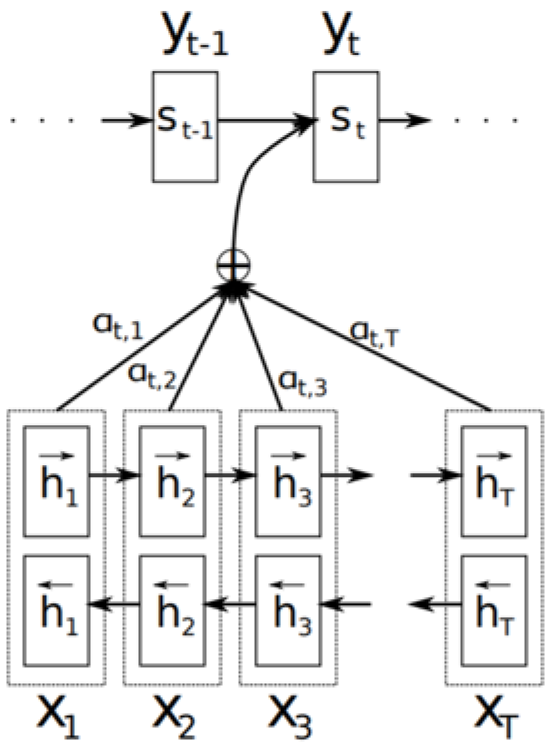
CNN + LSTM

من یک دانشجو هستم

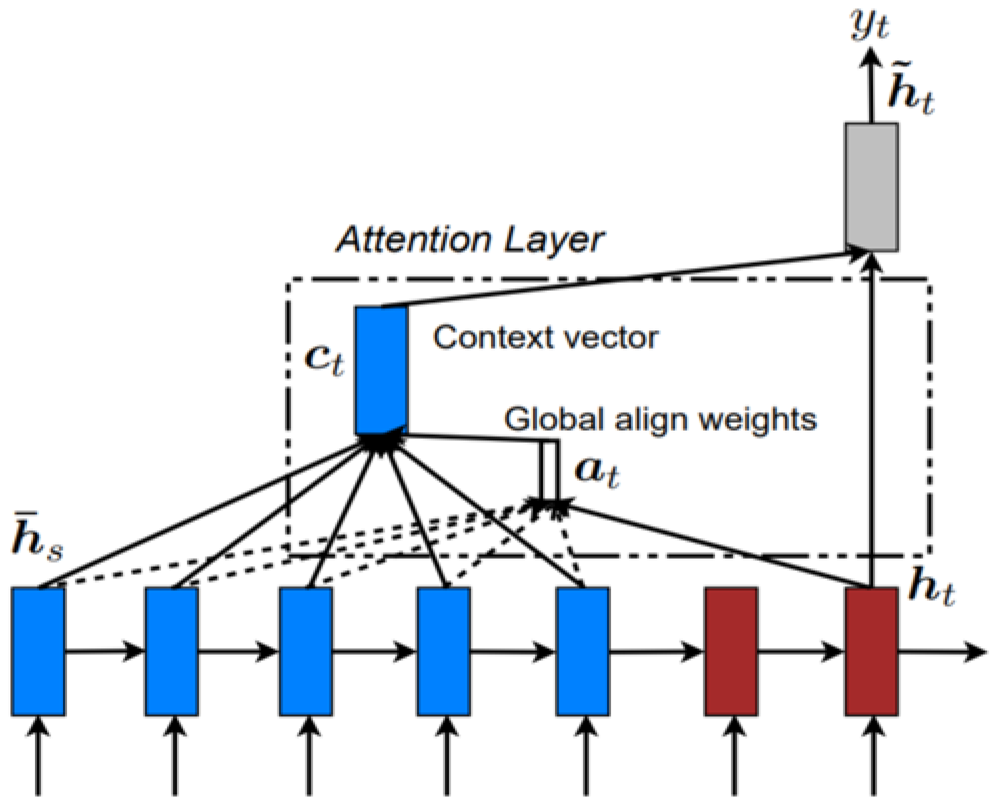
I am a student

Attention mechanisms in neural networks provide **learnable memory access**

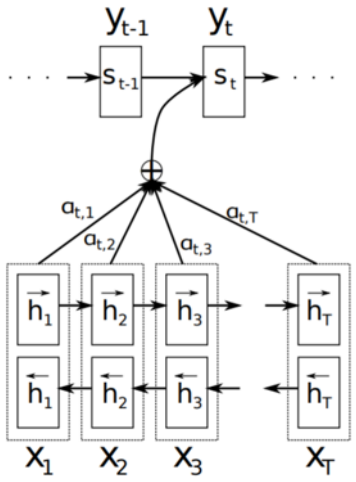




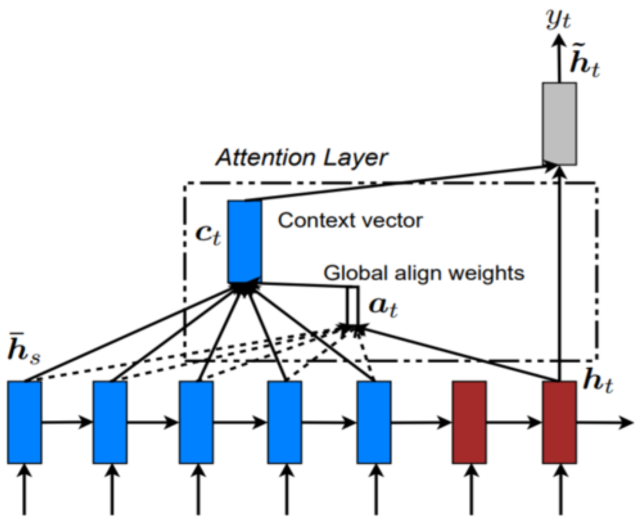
Bahdanau attention mechanism



Luong attention mechanism



Bahdanau attention mechanism



Luong attention mechanism

$$\alpha_{ts} = \frac{\exp\left(\operatorname{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)\right)}{\sum_{s'=1}^{S} \exp\left(\operatorname{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_{s'})\right)} \qquad \text{[Attention weights]}$$

$$\boldsymbol{c}_t = \sum_s \alpha_{ts} \bar{\boldsymbol{h}}_s \qquad \text{[Context vector]}$$

$$\boldsymbol{a}_t = f(\boldsymbol{c}_t, \boldsymbol{h}_t) = \tanh(\boldsymbol{W_c}[\boldsymbol{c}_t; \boldsymbol{h}_t]) \qquad \text{[Attention vector]}$$

$$\operatorname{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \boldsymbol{W} \bar{\boldsymbol{h}}_s & \text{[Luong's multiplicative style]} \\ \boldsymbol{v}_a^\top \tanh\left(\boldsymbol{W_1} \boldsymbol{h}_t + \boldsymbol{W_2} \bar{\boldsymbol{h}}_s\right) & \text{[Bahdanau's additive style]} \end{cases}$$

Attention Is All You Need

Ashish Vaswani*
Google Brain
avaswani@google.com

Noam Shazeer*
Google Brain
noam@google.com

Niki Parmar*
Google Research
nikip@google.com

Jakob Uszkoreit*
Google Research
usz@google.com

Llion Jones*
Google Research
llion@google.com

Aidan N. Gomez* †
University of Toronto
aidan@cs.toronto.edu

Łukasz Kaiser*
Google Brain
lukaszkaiser@google.com

Illia Polosukhin* ‡
illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions

Encoder
Decoder

ViT

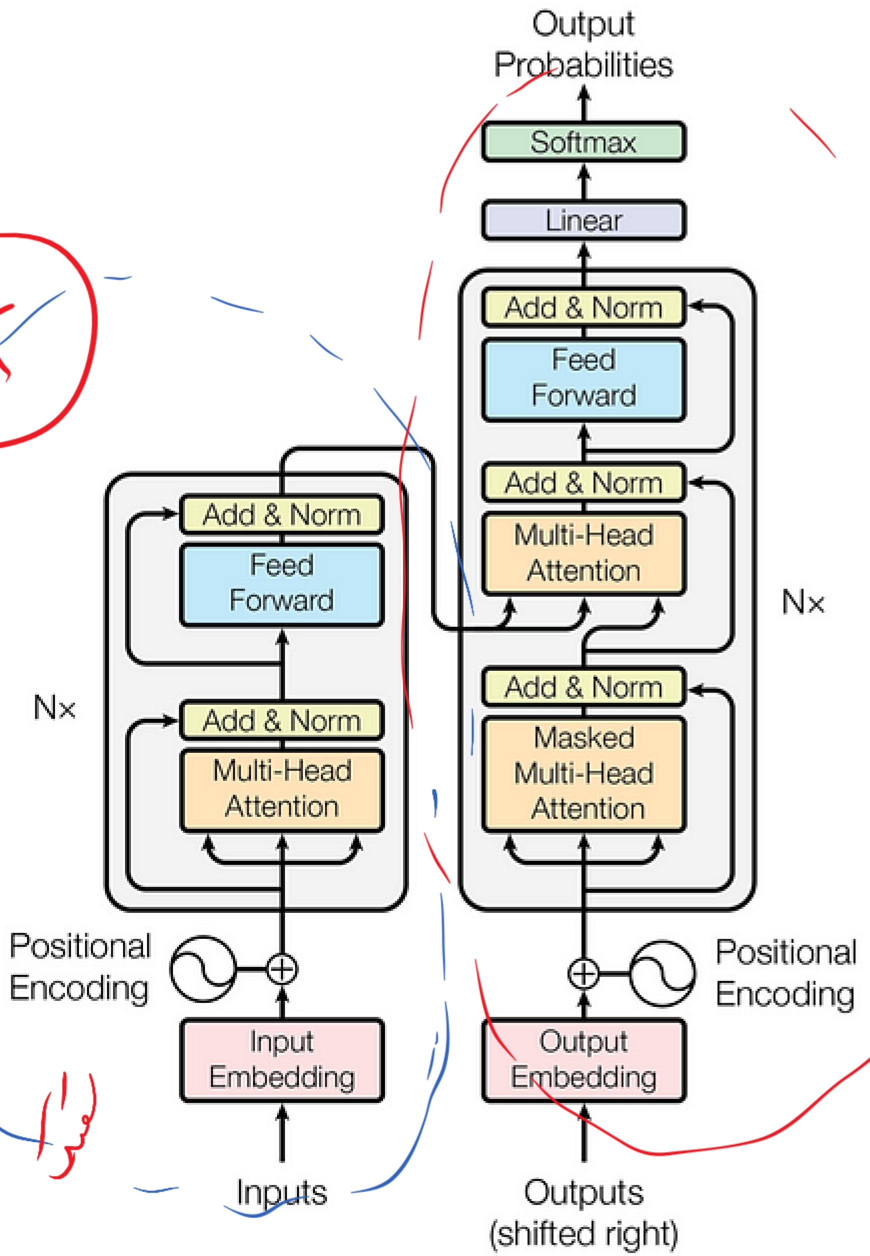
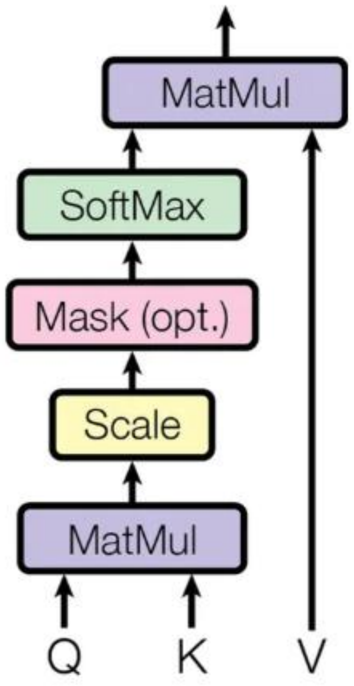
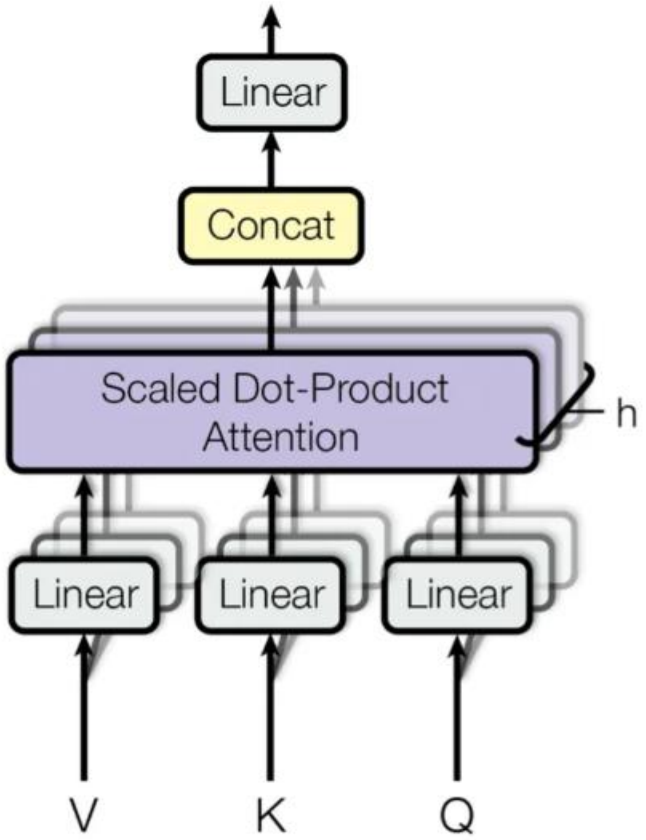


Figure 1: The Transformer - model architecture.

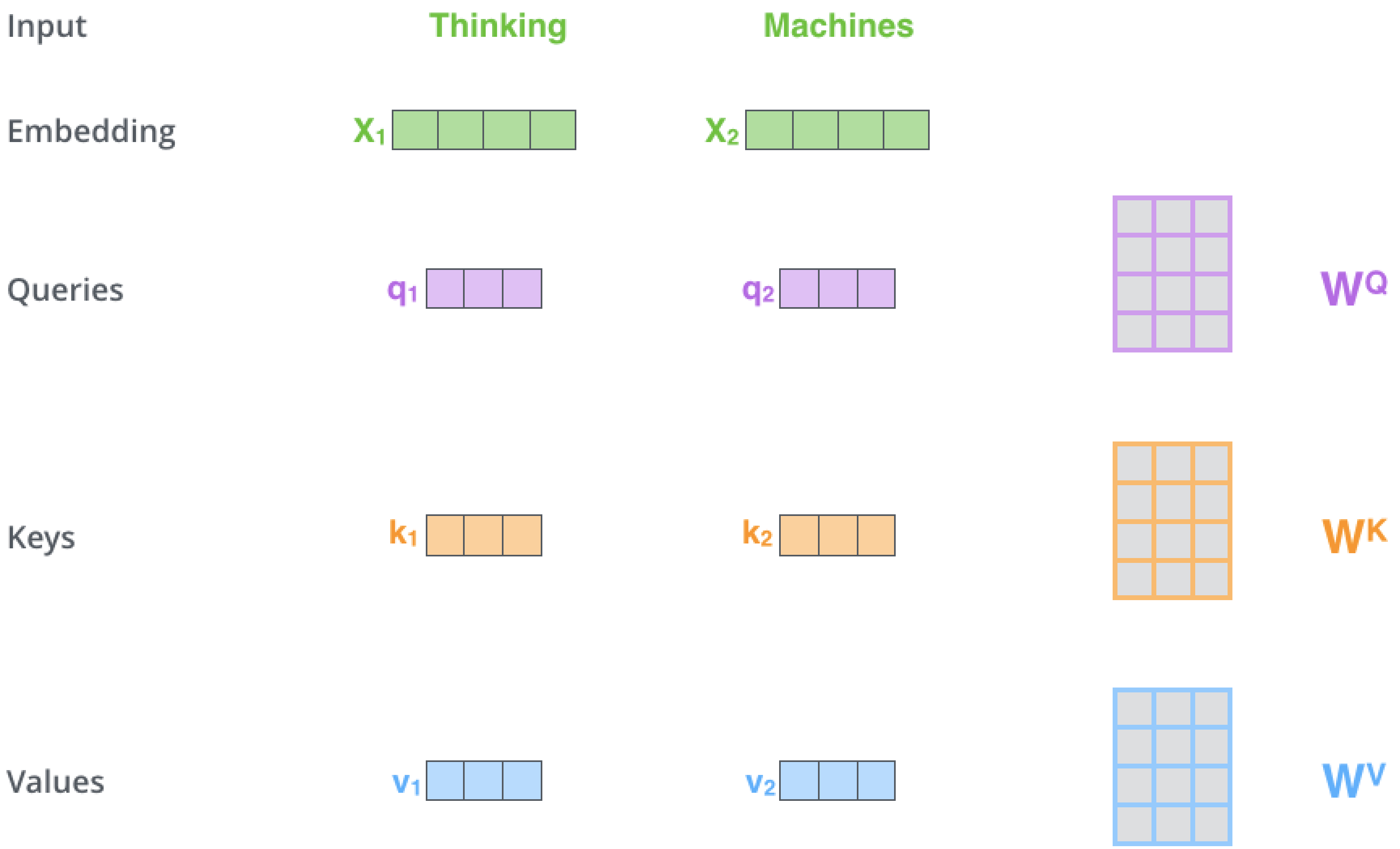
Scaled Dot-Product Attention

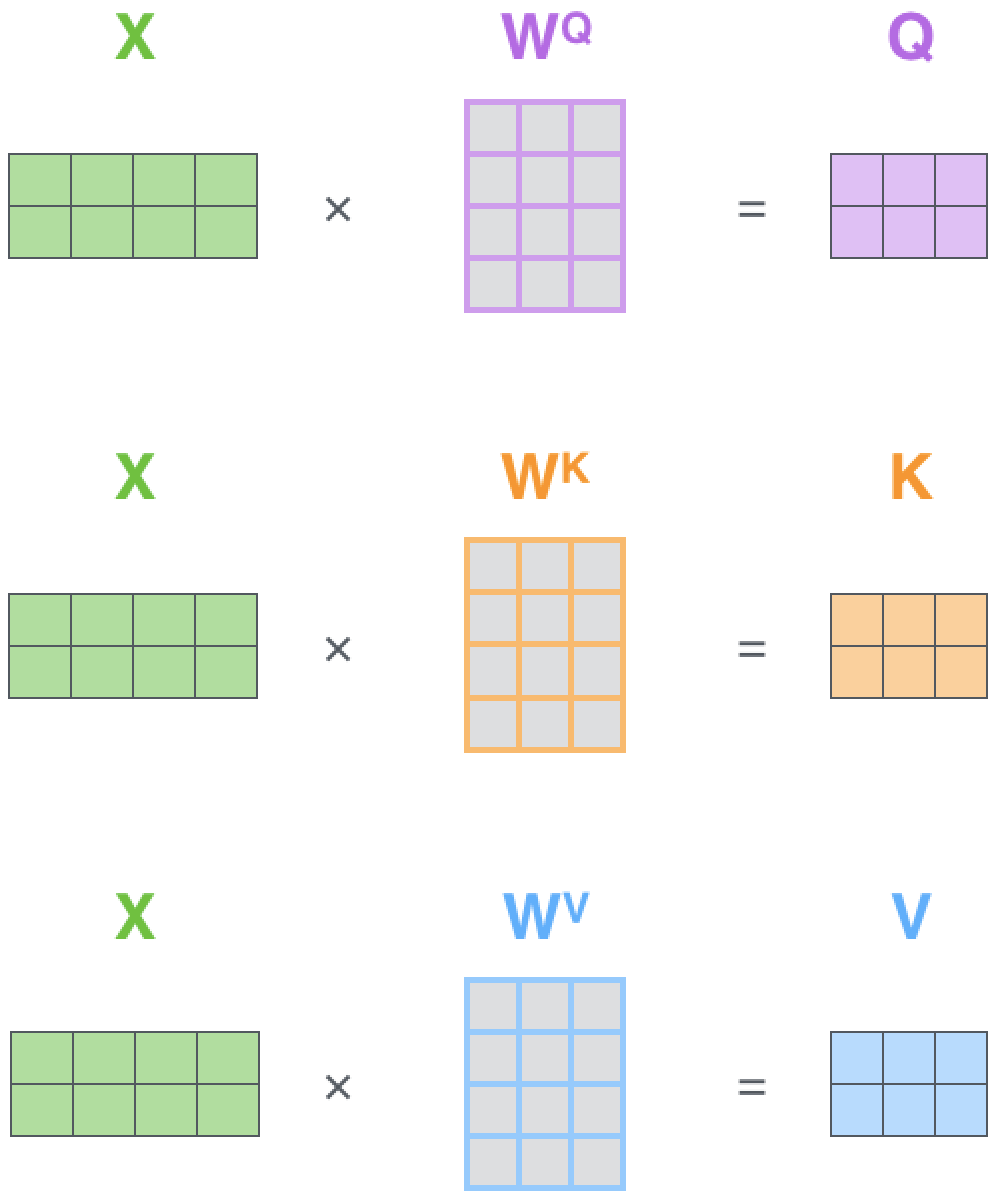


Multi-Head Attention



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

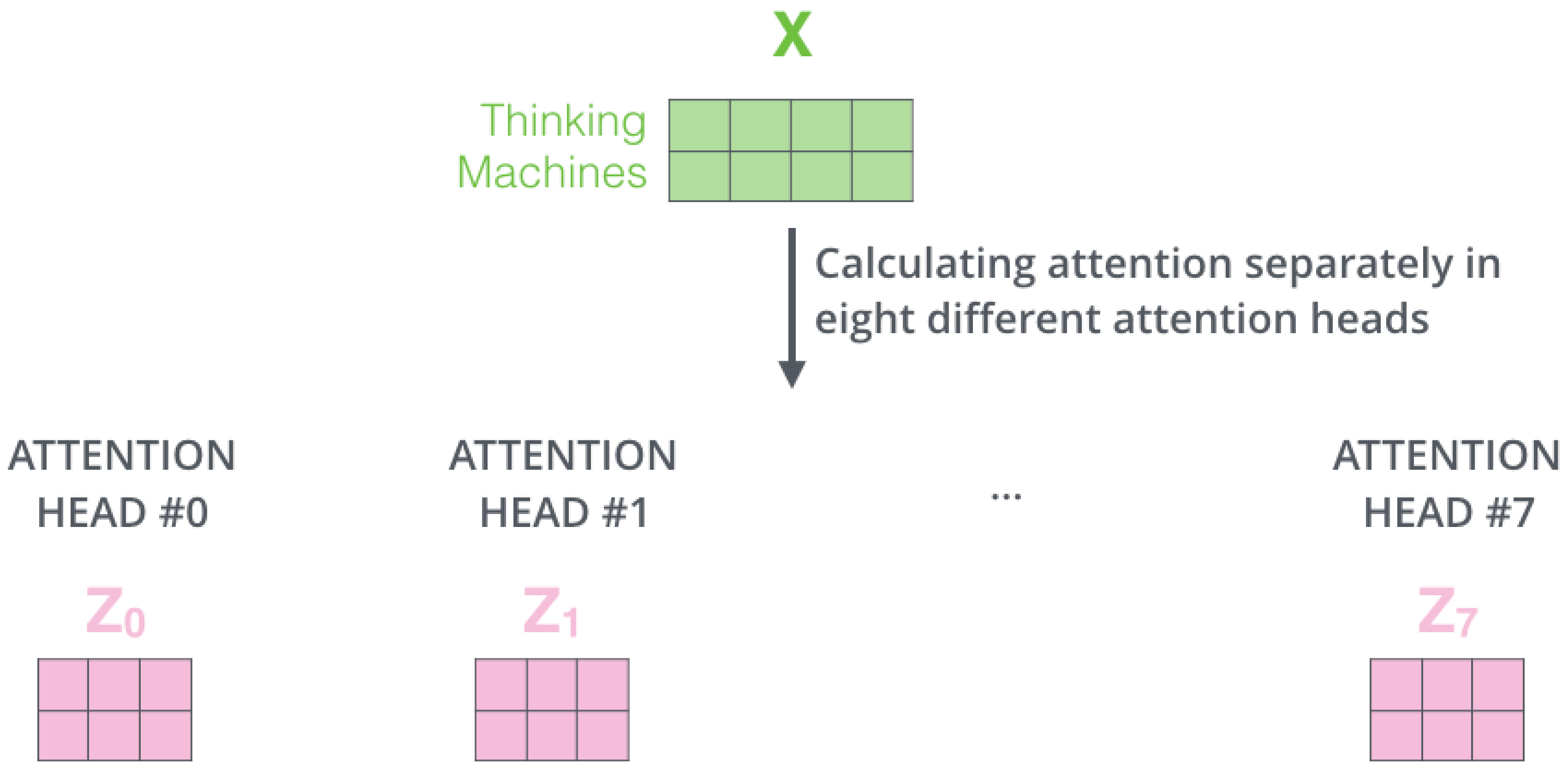

Input
Thinking
Machines
Embedding
X1
X2
Queries
q1
q2
WQ
Keys
k1
k2
WK
Values
v1
v2
WV

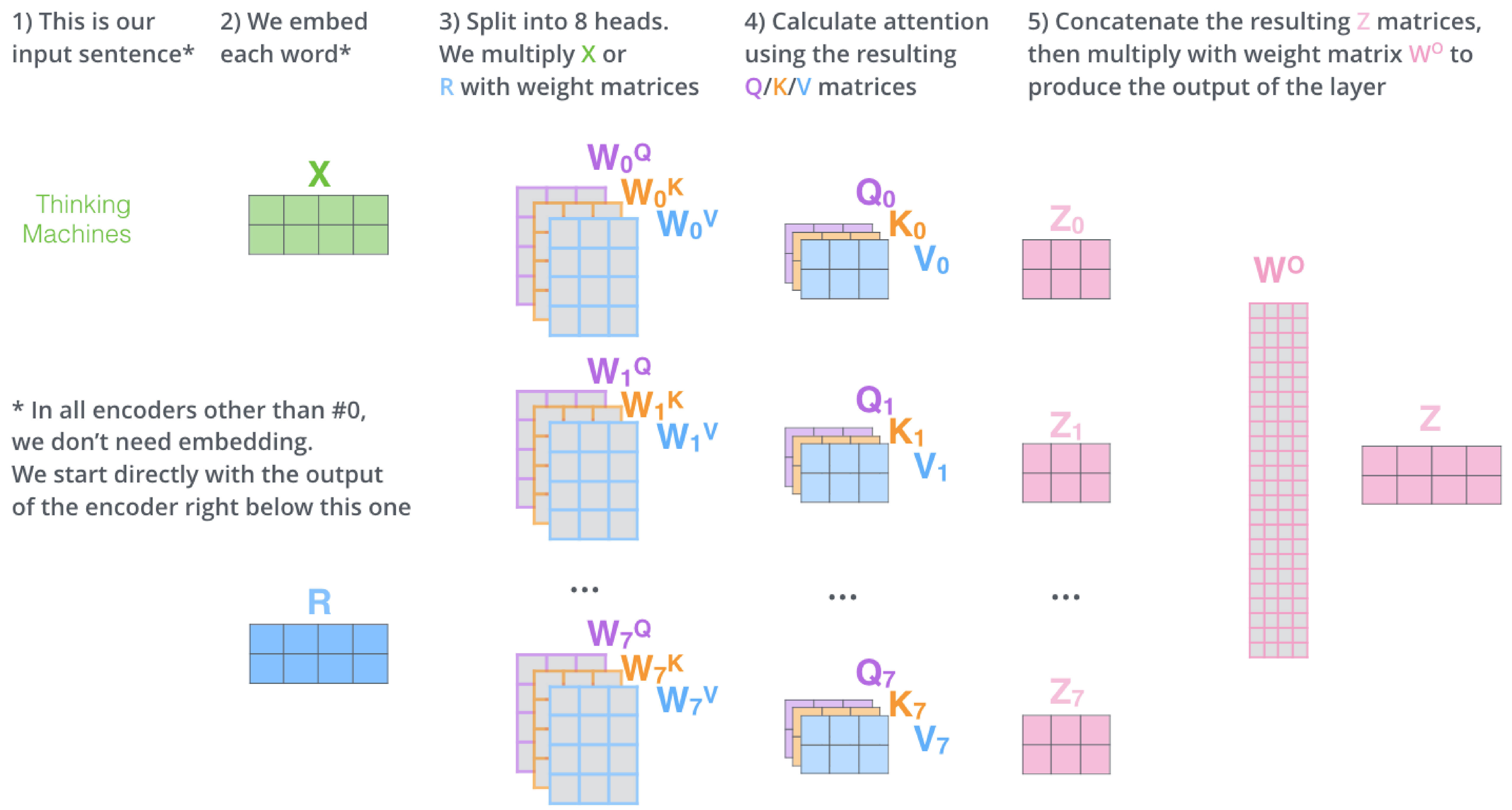

X
W^Q
Q
×
=
X
W^K
K
×
=
X
W^V
V
×
=

$$\cos\theta : \frac{A \cdot B}{\|A\|\|B\|}$$

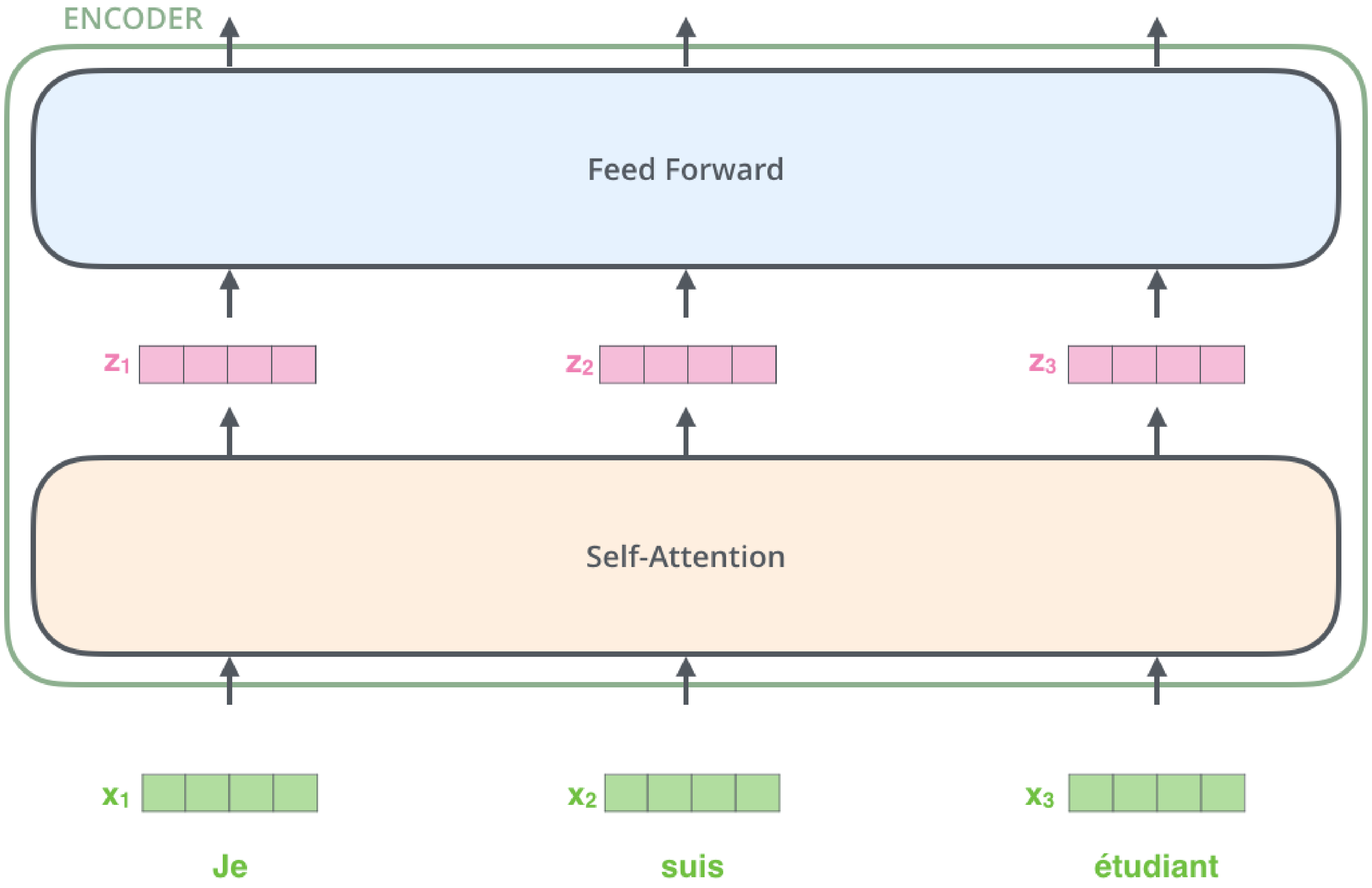
$$\text{softmax}\left(\frac{Q \times K^{T}}{\sqrt{d_k}}\right) V$$

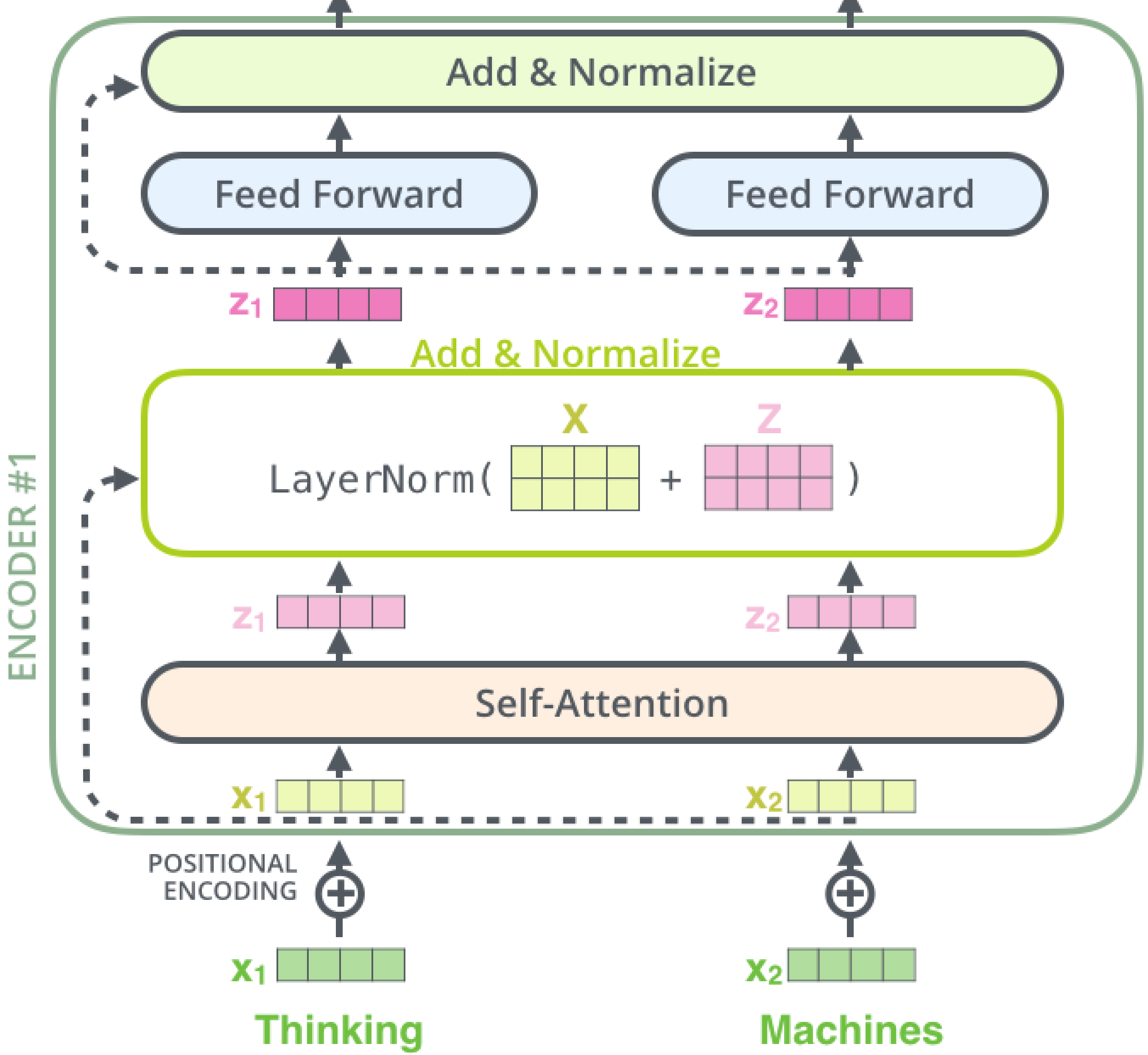
$$= Z$$

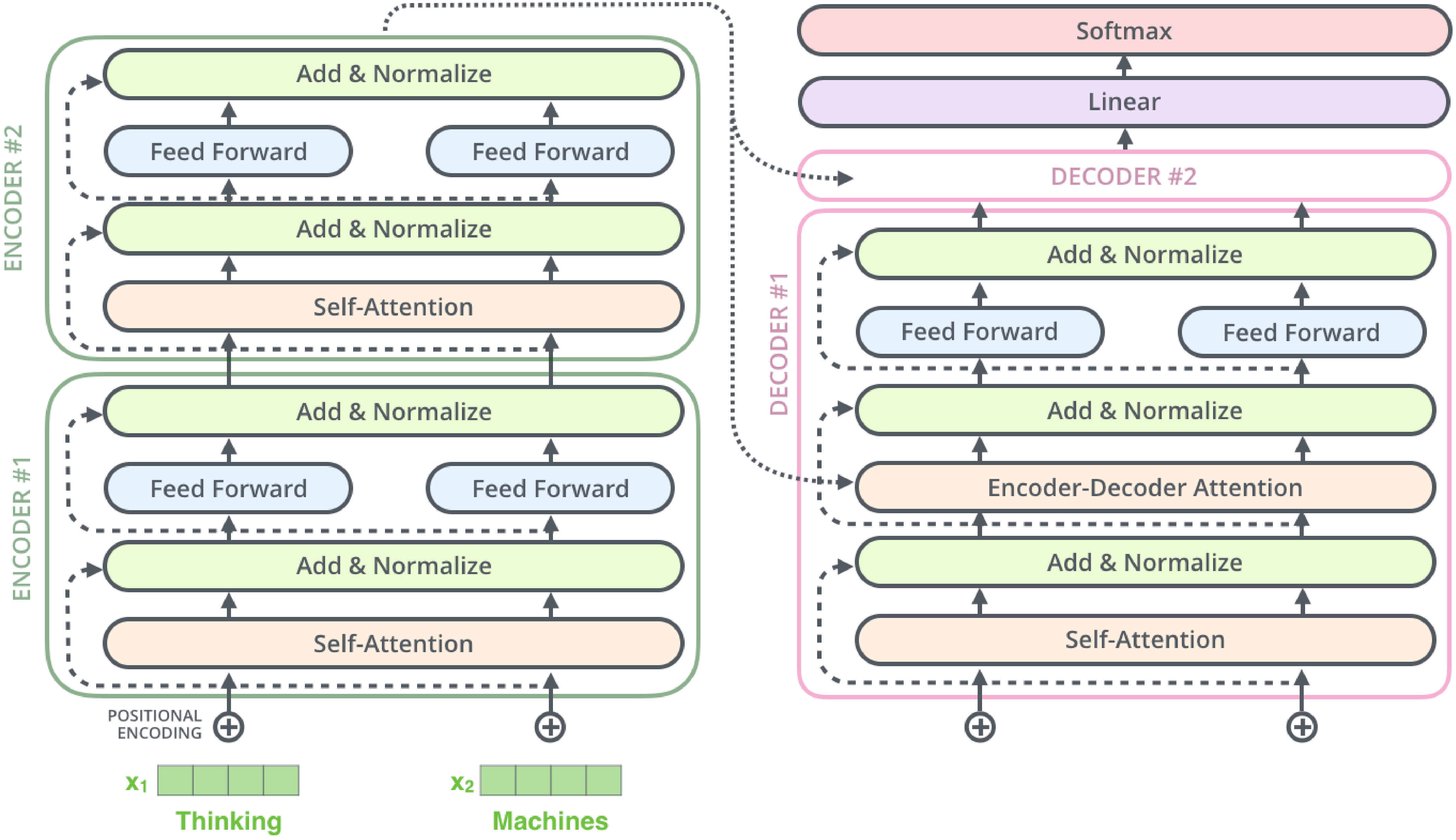

X
Thinking
Machines
Calculating attention separately in
eight different attention heads
ATTENTION
HEAD #0
ATTENTION
HEAD #1
...
ATTENTION
HEAD #7
Z_0
Z_1
Z_7



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
* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one
R
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


ENCODER
Feed Forward
z1
z2
z3
Self-Attention
x1
x2
x3
Je
suis
étudiant

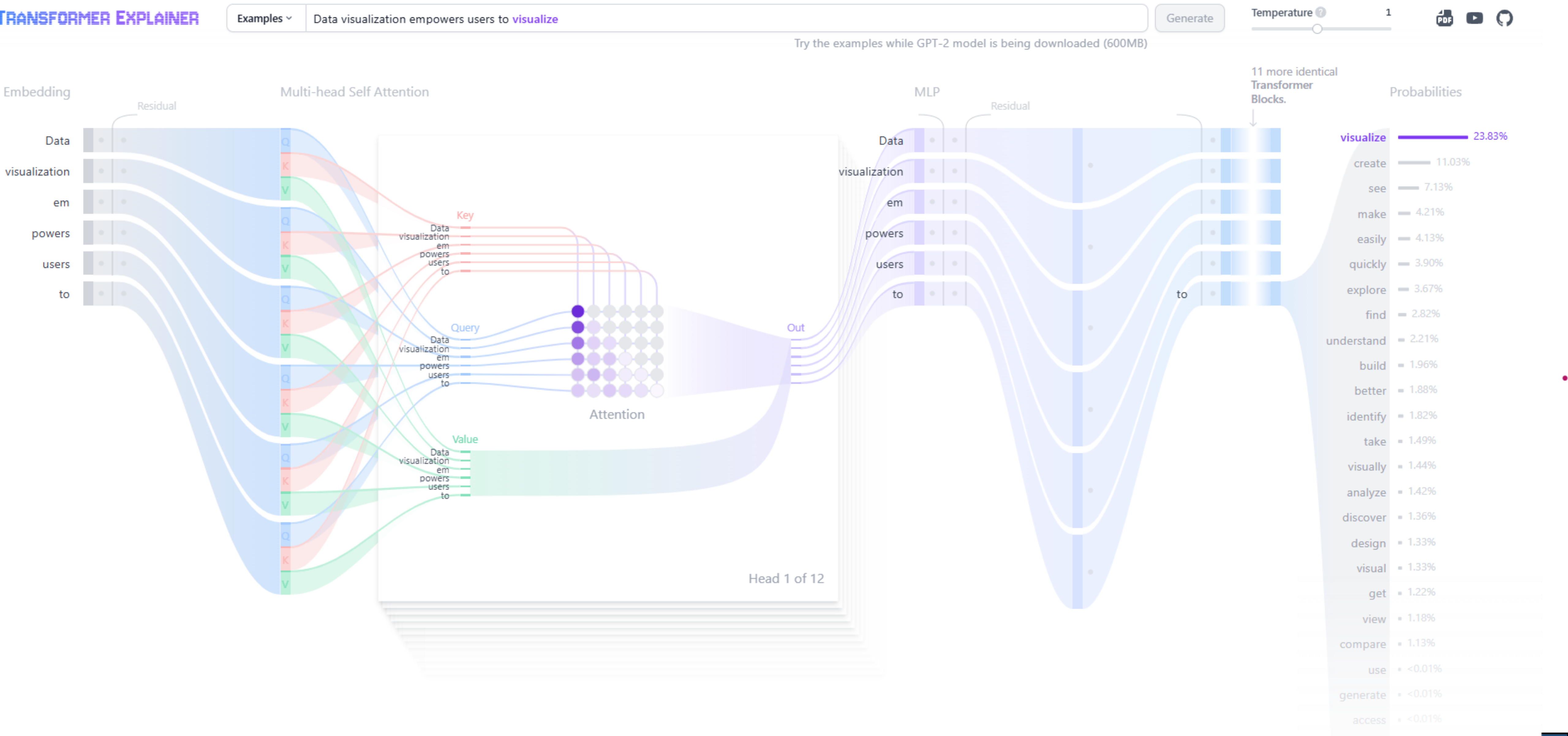

Add & Normalize
Feed Forward
Feed Forward
z1
z2
Add & Normalize
X
Z
LayerNorm(+)
z1
z2
Self-Attention
x1
x2
ENCODER #1
POSITIONAL
ENCODING
x1
x2
Thinking
Machines


Softmax
Linear
DECODER #2
DECODER #1
Add & Normalize
Feed Forward
Feed Forward
Add & Normalize
Encoder-Decoder Attention
Add & Normalize
Self-Attention
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
x1
Thinking
x2
Machines

$$PE(pos, 2i) = \sin(pos/10000^{2i/d})$$

$$PE(pos, 2i+1) = \cos(pos/10000^{2i/d})$$

https://poloclub.github.io/transformer-explainer/



https://poloclub.github.io/transformer-explainer/

Vision Transformer ViT