شبکه‌های بازگشت‌کننده

transformer

دنباله‌ها و سری‌های زمانی

RNN

1D CNN

دانشگاه شهید بهشتی
پژوهشکده‌ی فضای مجازی
پاییز ۱۴۰۱
احمد محمودی ازناوه

http://faculties.sbu.ac.ir/~a_mahmoudi/

فهرست مطالب

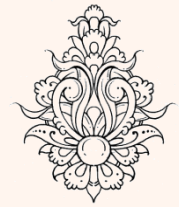
- سری‌های زمانی و دنباله‌ها
  - بازنمایی داده‌های متنی
  - شبکه‌های عصبی بازگشت‌کننده
  - شبکه‌های کانولوشنی یک بعدی
  - transformers

معرفی

- در موارد بسیاری داده‌ها به صورت یک دنباله هستند.
  - بین اجزای این دنباله‌ها وابستگی وجود دارد.
  - لازم است وابستگی بین داده‌ها در نظر گرفته شود.
    - خروجی در یک لحظه تنها به ورودی آن لحظه وابسته نیست، ورودی‌های پیشین و یا اثر آن‌ها باید در نظر گرفته شود.
  - طول دنباله‌ها متفاوت است. شیوه‌ای مطلوب است که قابلیت پردازش با دنباله با طول‌های متفاوت را داشته باشد.
  - یک الگوی خاص ممکن است در موقعیت‌های مختلفی از دنباله رخ دهد.

کاربردها

- دسته‌بندی سری‌های زمانی
  - دسته‌بندی مستندات متنی
  - تشخیص موضوع متن
  - تشخیص نویسنده
- مقایسه‌ی سرهای زمانی
  - مقایسه‌ی متون
- یادگیری دنباله-دنباله
  - ترجمه همزمان
- تجزیه و تحلیل احساسات
  - دسته‌بندی نظرات کاربران
- پیش‌بینی سری‌های زمانی
  - پیش‌بینی دما
- سیستم‌های پرسش و پاسخ

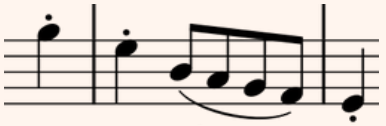


مثال

Speech recognition

→ *"The quick brown fox jumped over the lazy dog."*

Music generation

ø → 

Sentiment classification *"There is nothing to like in this movie."* → ★☆☆☆☆

Machine translation *Voulez-vous chanter avec moi?* → *Do you want to sing with me?*

Video activity recognition

→ *Running*

Name entity recognition *Yesterday, Harry Potter met Hermione Granger.* →

Yesterday, ***Harry Potter*** *met* ***Hermione Granger****.*

Adopted from Andrew Ng

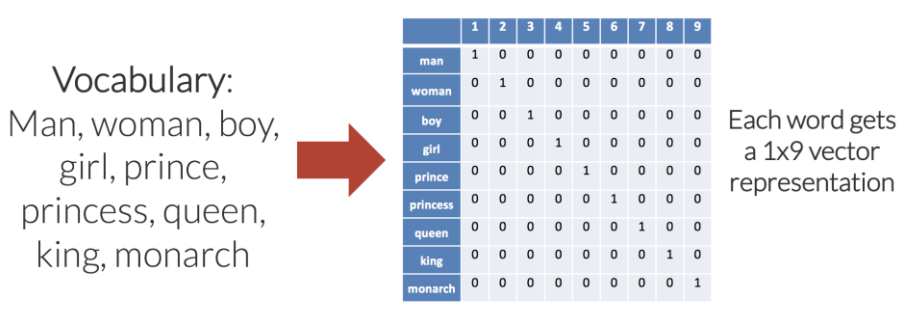
بازنمایی متن

bag of word

- کیسه‌ای از کلمات:
  - در این شیوه مجموعه‌ی کلمات متن به صورت برداری در نظر گرفته می‌شود، با توجه به این که این شیوه ترتیب دنباله متن را حفظ نمی‌کند، کارایی لازم را ندارد.

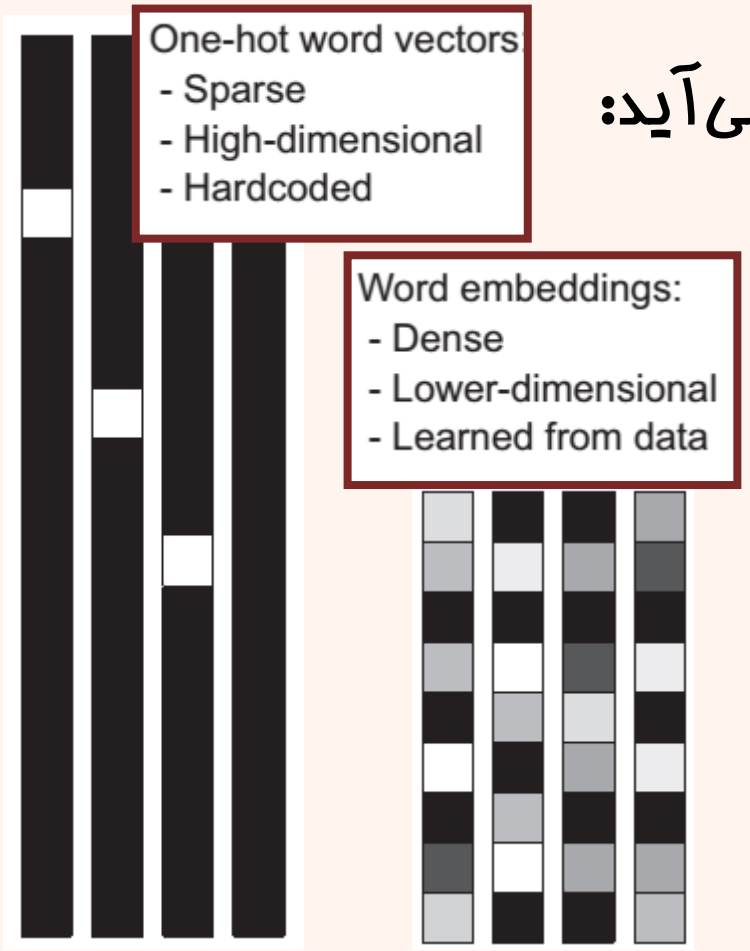
order preserving

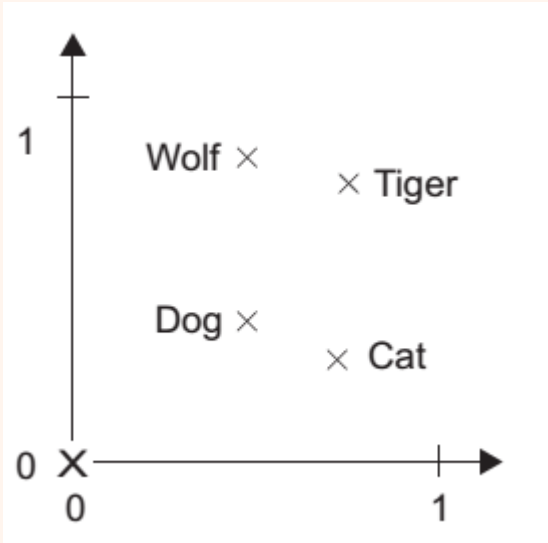
- کدگذاری one-hot:
  - در این شیوه برداری با طول تعداد کل واژگان مورد استفاده در نظر گرفته می‌شود. به ازای هر کلمه یکی از ابعاد در نظر گرفته می‌شود.
  - عیب: ابعاد بالا



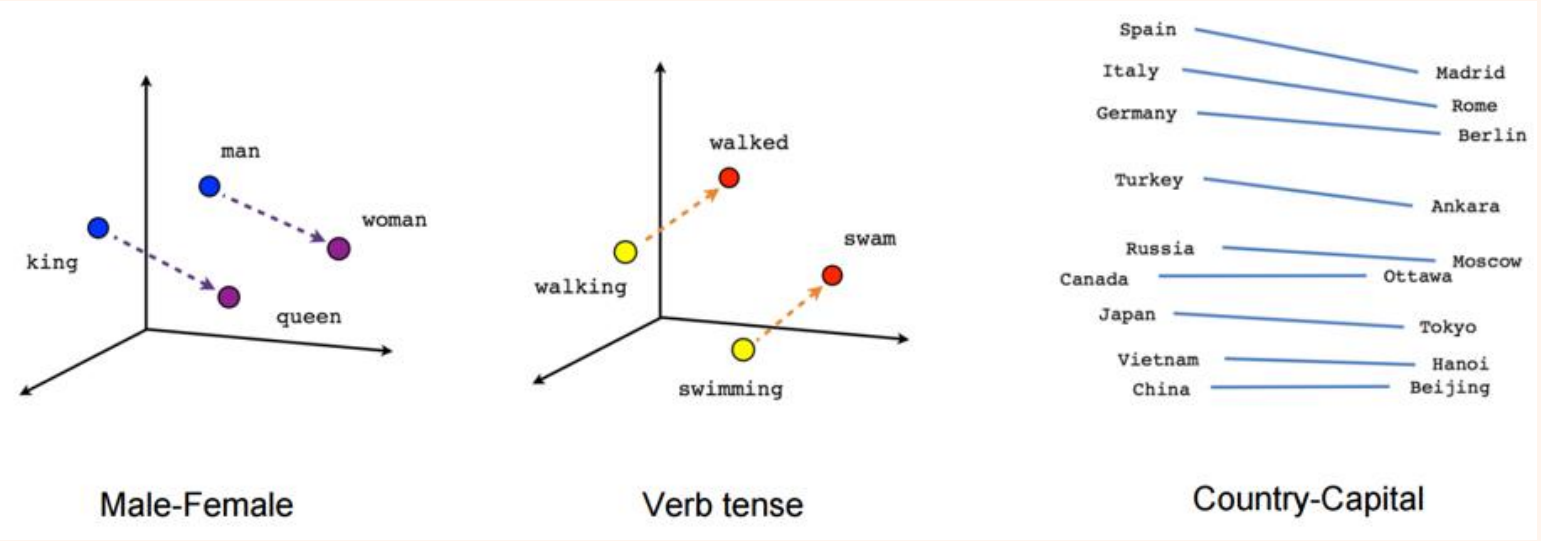
	1	2	3	4	5	6	7	8	9
man	1	0	0	0	0	0	0	0	0
woman	0	1	0	0	0	0	0	0	0
boy	0	0	1	0	0	0	0	0	0
girl	0	0	0	1	0	0	0	0	0
prince	0	0	0	0	1	0	0	0	0
princess	0	0	0	0	0	1	0	0	0
queen	0	0	0	0	0	0	1	0	0
king	0	0	0	0	0	0	0	1	0
monarch	0	0	0	0	0	0	0	0	1

- در مقابل، به هر کلمه یک بردار dense اختصاص داد.
- این بردار به دو شیوه به دست می‌آید:
  - آموزش در کنار کاربرد اصلی
  - استفاده از مدل‌های قبلی





مثال



vector[Queen] ≈ vector[King] - vector[Man] + vector[Woman]

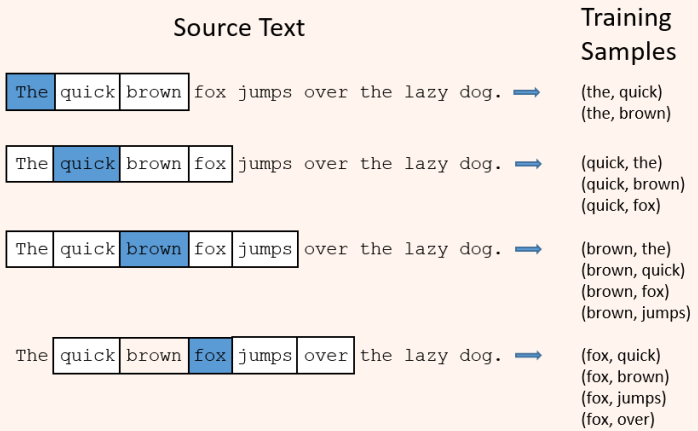
vector[Paris] ≈ vector[France] - vector[Italy] + vector[Rome]

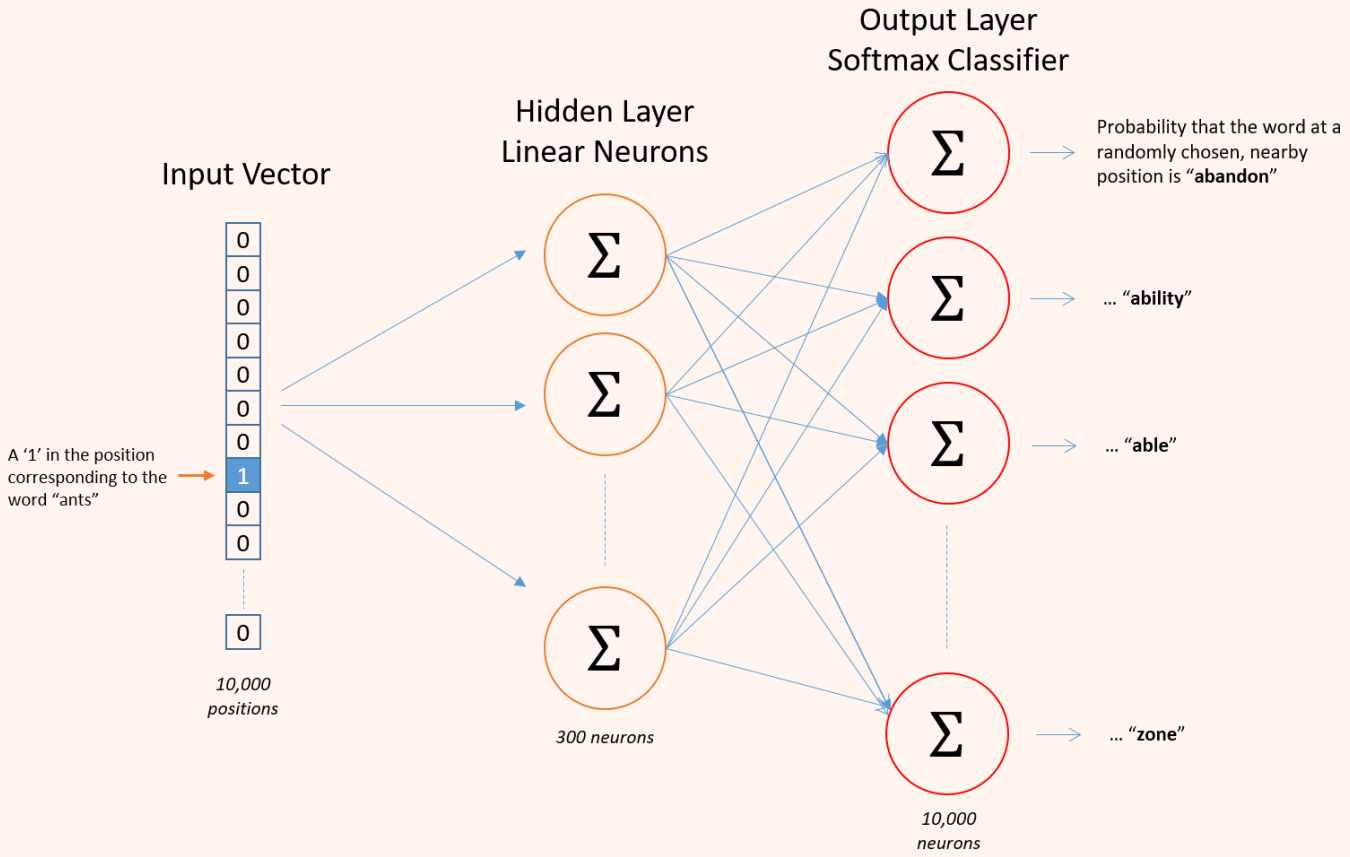
This can be interpreted as “France is to Paris as Italy is to Rome”.

Adopted frome Dr. Reda Bouadjenek slides

Skip-Gram Model

unsupervised feature learning

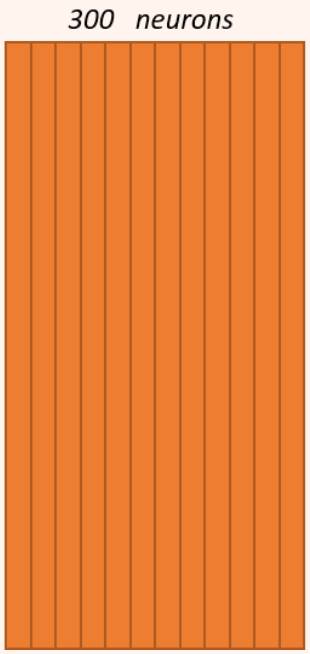




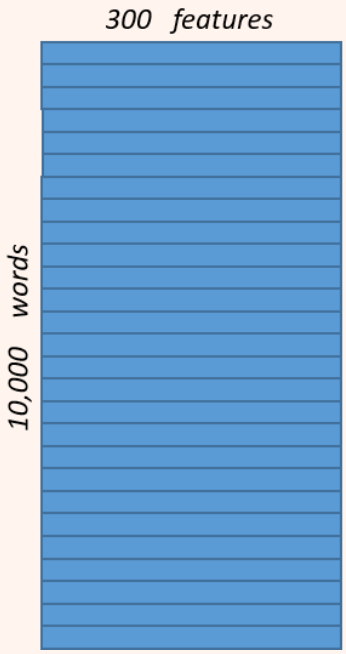
http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

Skip-Gram Model

Hidden Layer Weight Matrix



Word Vector Lookup Table!



$$[0 \quad 0 \quad 0 \quad 1 \quad 0] \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = [10 \quad 12 \quad 19]$$

```
from keras.layers import Embedding
embedding_layer = Embedding(1000, 64)
```

Word index ⟶ Embedding layer ⟶ Corresponding word vector

(samples, sequence_length)

(samples, sequence_length, embedding_dimensionality).

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

"Efficient Estimation of Word Representations in Vector Space" Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean, Arxiv 2013

مثال IMDB

```
from keras.models import Sequential
from keras.layers import Flatten, Dense
model = Sequential()
model.add(Embedding(10000, 8, 20))
model.add(Flatten())
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='rmsprop',
loss='binary_crossentropy', metrics=['acc'])
model.summary()
history = model.fit(x_train, y_train,
                    epochs=10,
                    batch_size=32,
                    validation_split=0.2)
```

val_acc: 0.7464

Layer (type)	Output Shape	Param #
embedding_3 (Embedding)	(None, 20, 8)	80000
flatten_1 (Flatten)	(None, 160)	0
dense_1 (Dense)	(None, 1)	161

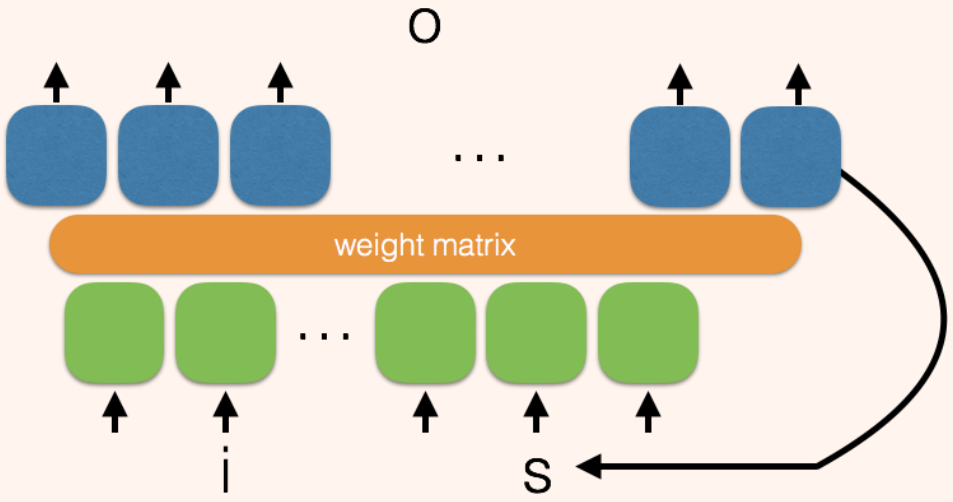
پردازش دنباله با طول متغیر

عدم وابستگی به موقعیت

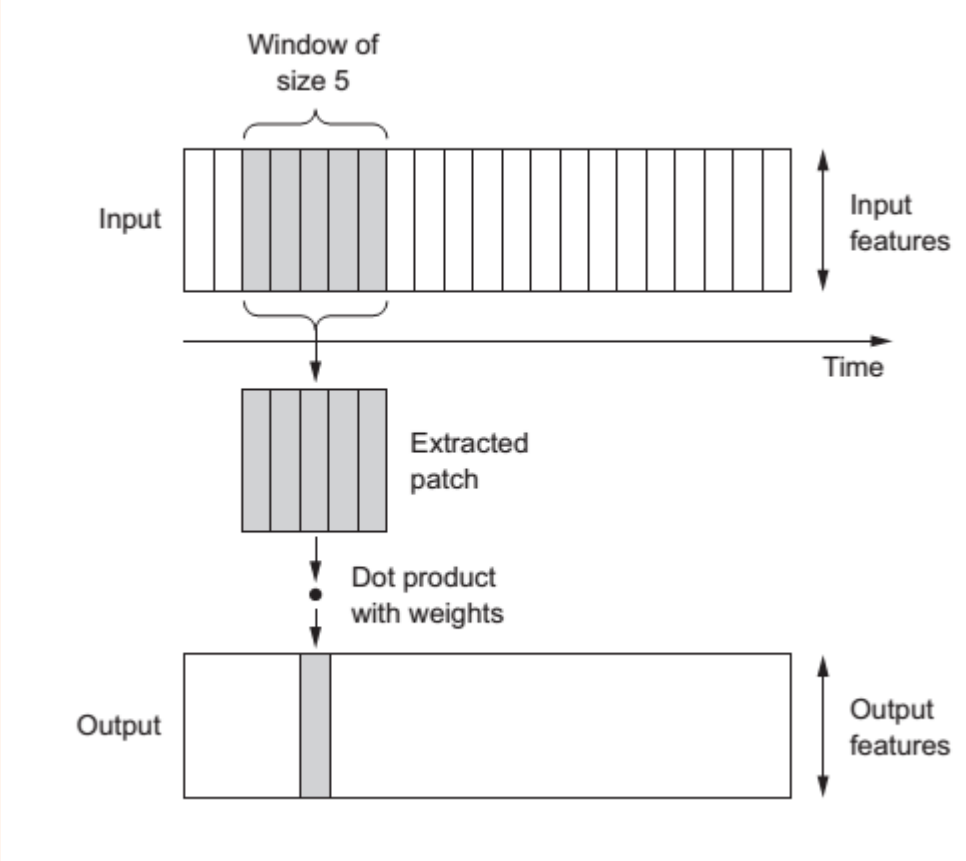
اشتراک پارامترها

شبکه‌های کانولوشنی یک بعدی

بازخورد(فیدبک)

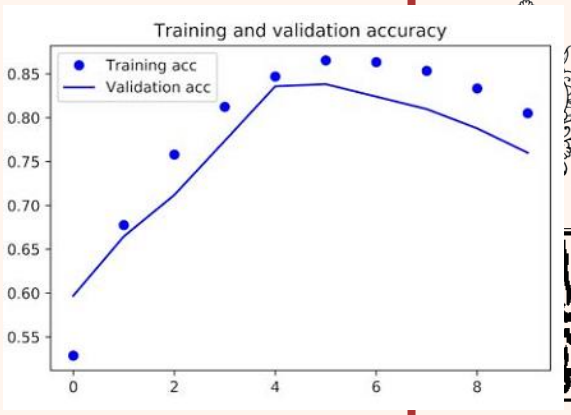


شبکه‌های کانولوشنی یک بعدی



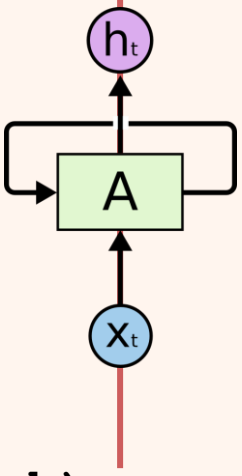
کانولوشن می‌تواند الگوهای محلی را تشخیص دهد

```
from keras.models import Sequential
from keras import layers
from keras.optimizers import RMSprop
model = Sequential()
model.add(layers.Embedding(max_features, 128,
input_length=max_len))
model.add(layers.Conv1D(32, 7, activation='relu'))
model.add(layers.MaxPooling1D(5))
model.add(layers.Conv1D(32, 7, activation='relu'))
model.add(layers.GlobalMaxPooling1D())
model.add(layers.Dense(1))
model.summary()
model.compile(optimizer=RMSprop(lr=1e-4),
loss='binary_crossentropy',
metrics=['acc'])
history = model.fit(x_train, y_train,
epochs=10,
batch_size=128,
validation_split=0.2)
```



شبکه‌های بازگشت‌کننده

- شبکه‌های عصبی feedforward هر ورودی را بدون در نظر گرفتن سایر ورودی‌ها(به صورت مستقل) پردازش می‌کند.
  - حافظه ندارند.
  - باید کل ورودی یک‌باره اعمال شود.
  - شبکه‌های بازگشت‌کننده برای برطرف کردن این نقیصه مطرح شده‌اند:

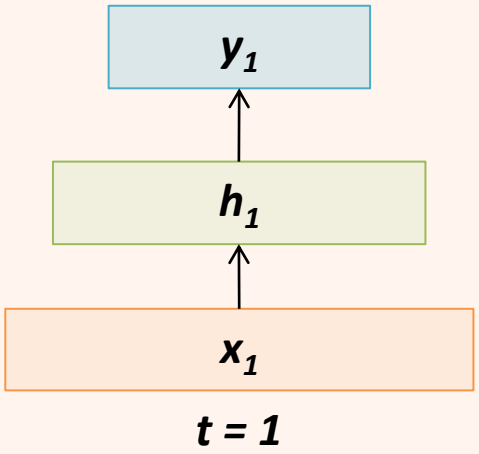



```
state_t = 0
for input_t in input_sequence:
output_t = activation(dot(W, input_t) + dot(U, state_t) + b)
state_t = output_t
```


http://colah.github.io/posts/2015-08-Understanding-LSTMs/

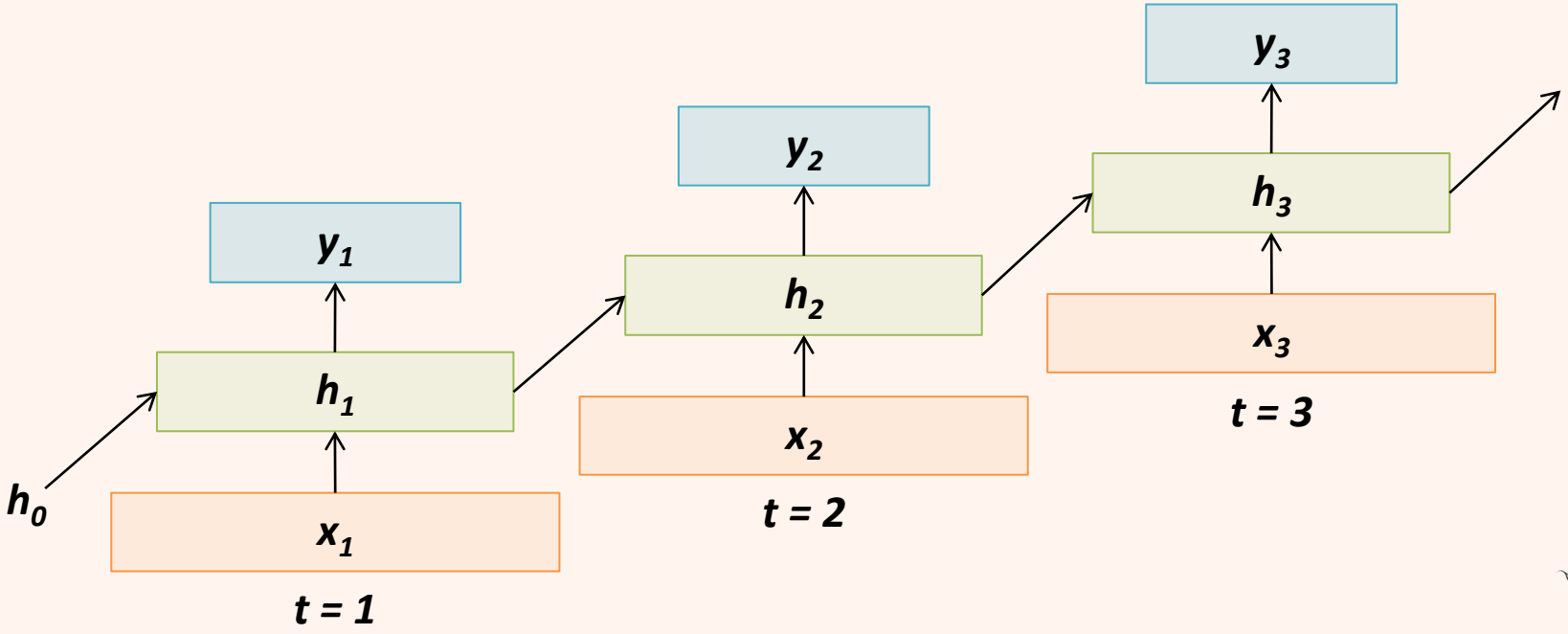
شبکه‌های عصبی feedforward

- در شبکه‌های عصبی feedforward قابلیت پردازش دنباله‌ها با طول‌های متفاوت وجود ندارد.
  - هر ورودی بدون در نظر گرفتن سایر ورودی‌ها(به صورت مستقل) پردازش می‌شود.
  - ترتیب ورودی‌ها را در نظر نمی‌گیرد.



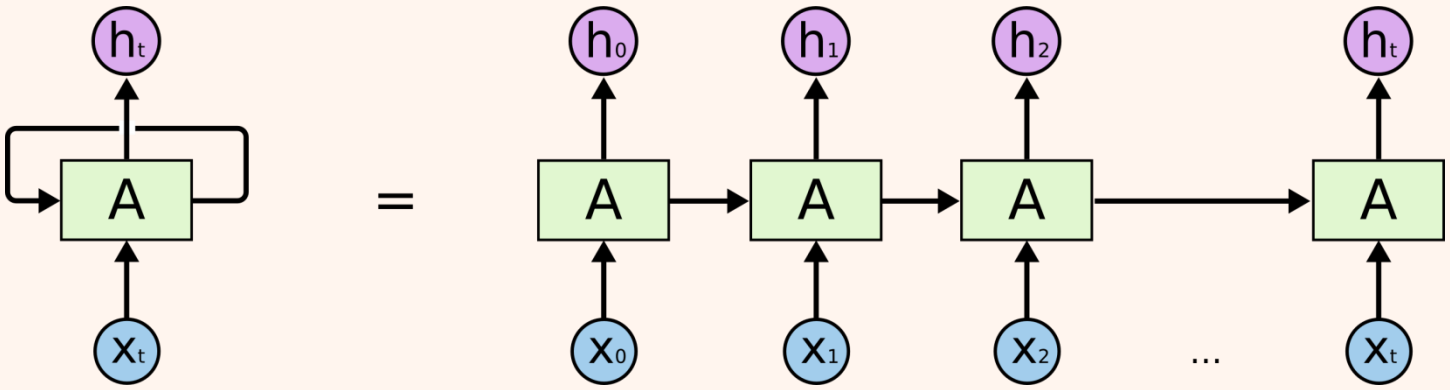
```
state_t = 0                          <-- The state at t
for input_t in input_sequence:       <-- Iterates over sequence elements
    output_t = f(input_t, state_t)
    state_t = output_t               <-- The previous output becomes the state for the next iteration.
```

- چنان‌چه با دنباله‌ای از داده مواجه باشیم:



- لازم است ورودی‌های پیشین(اثر آن‌ها به عنوان خروجی آن لحظه/حالت سیستم) در خروجی زمان فعلی در نظر گرفته شود.

An unrolled recurrent neural network



این شبکه‌های برای پردازش سری‌های زمانی و دنباله‌ها بسیار مناسب هستند

speech recognition, language modeling, translation, image captioning

- انتخاب خروجی:
  - دنباله‌ی همه خروجی‌ها
  - آخرین خروجی

sentiment analysis

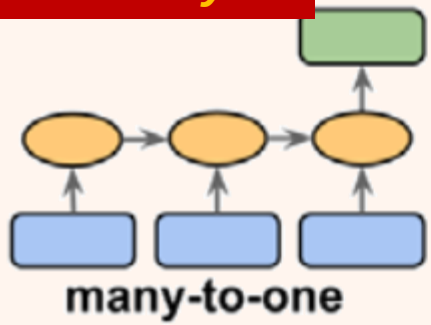
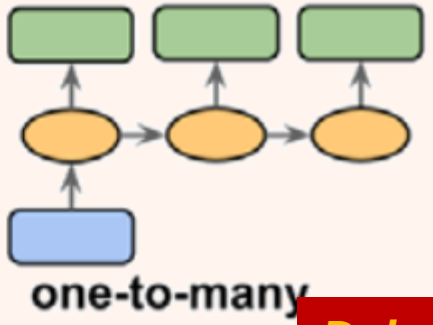
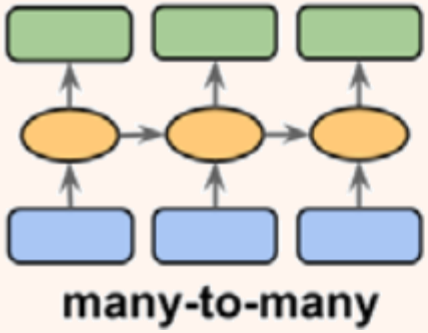
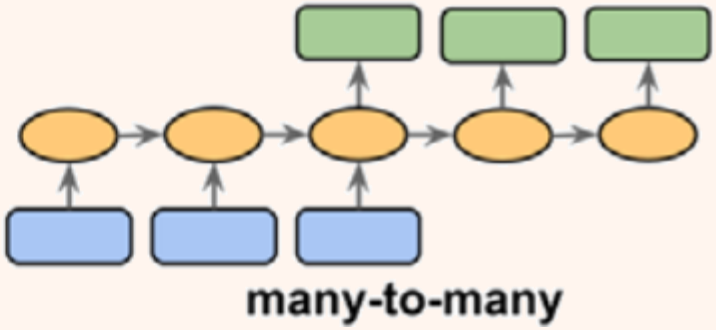


Image captioning



Delayed: translating





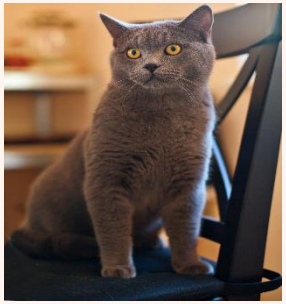
Synchronized: video classification

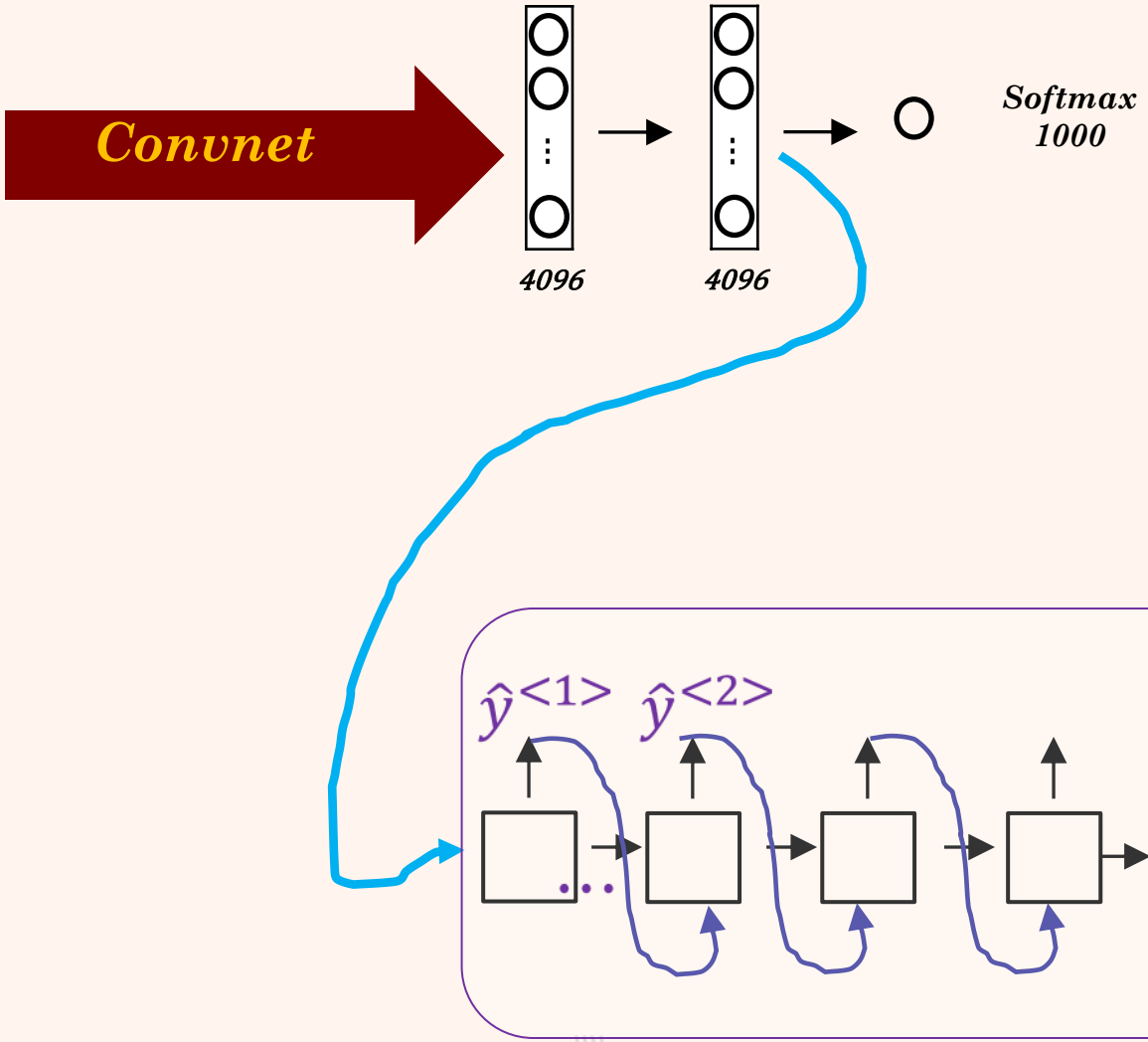
Raschka, S., and V. Mirjalili. Python Machine Learning: Machine Learning and Deep Learning with Python, Scikit-Learn, and Tensorflow 2, 3rd Edition. Packt Publishing, 2019.

Image captioning

$y^{<1>}$ $y^{<2>}$ $y^{<3>}$ $y^{<4>}$ $y^{<5>}$ $y^{<6>}$

A cat sitting on a chair





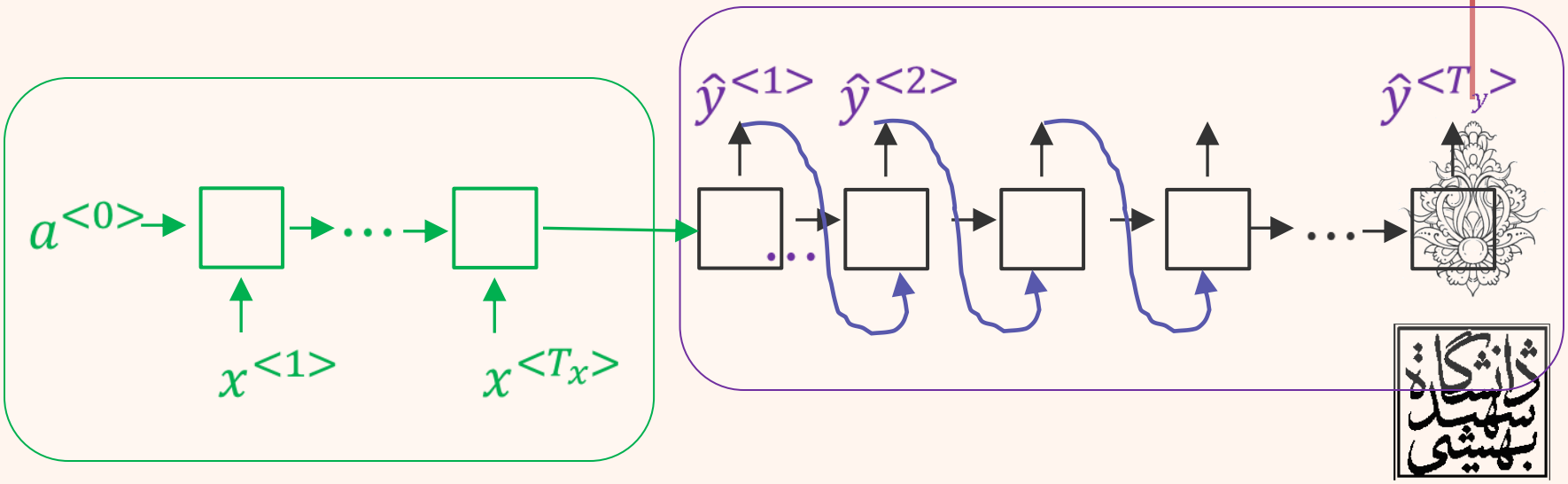
دنباله-دنباله

$x^{<1>}$ $x^{<2>}$ $x^{<3>}$ $x^{<4>}$ $x^{<5>}$

Jane visite l'Afrique en septembre

⟶ ***Jane is visiting Africa in September.***

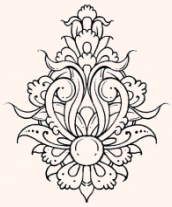
$y^{<1>}$ $y^{<2>}$ $y^{<3>}$ $y^{<4>}$ $y^{<5>}$ $y^{<6>}$



شبکه‌های بازگشت‌کننده

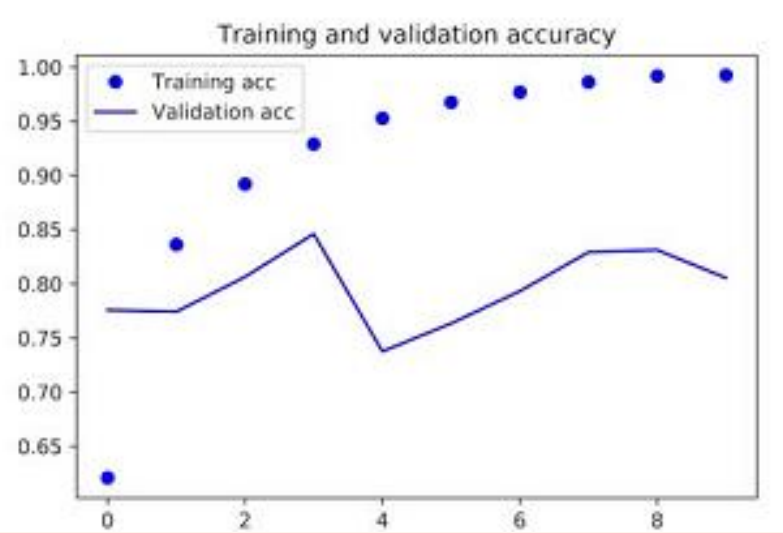
```
from keras.models import Sequential
from keras.layers import Embedding, SimpleRNN

model = Sequential()
model.add(Embedding(10000, 32))
model.add(SimpleRNN(10))
model.summary()
```

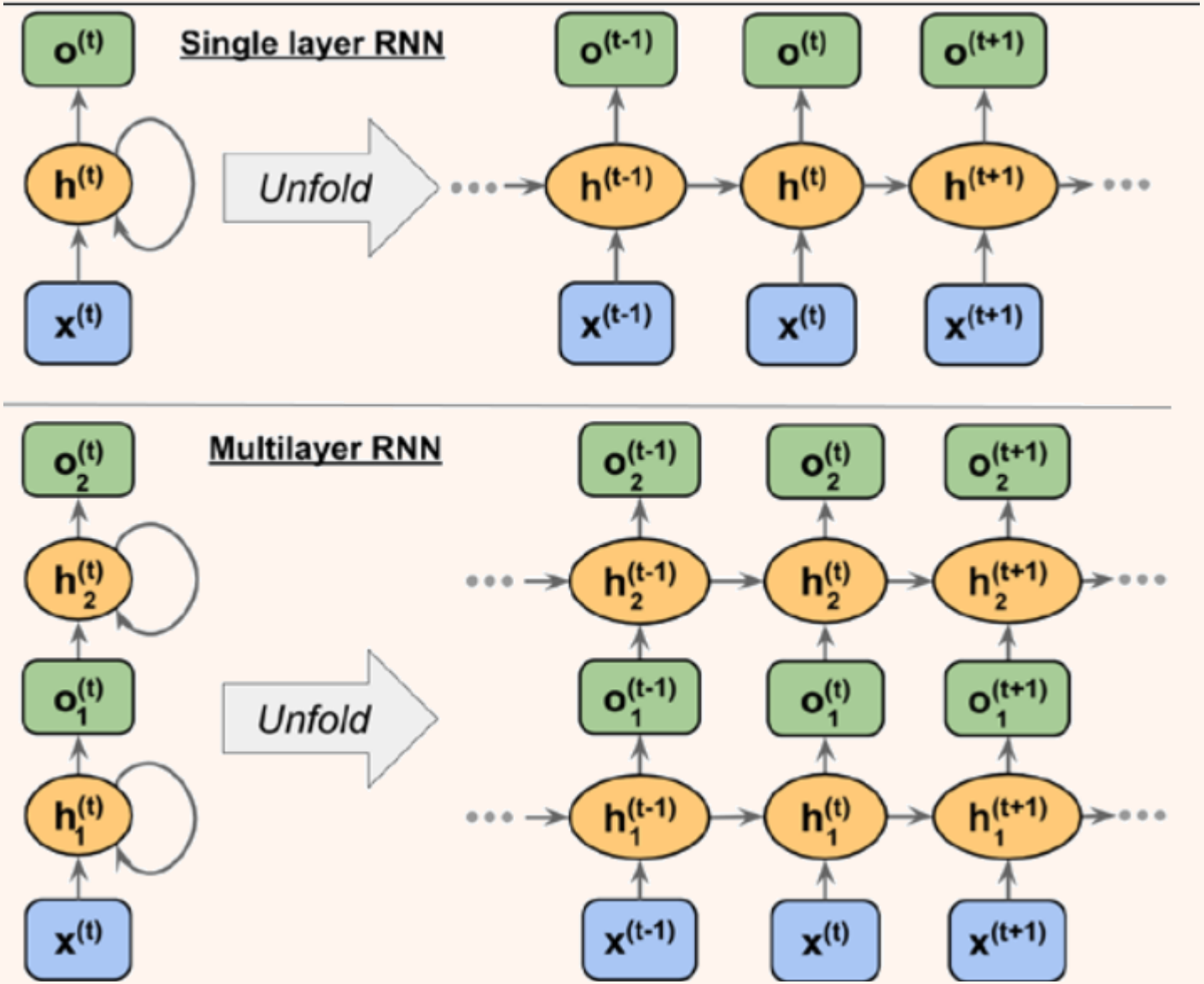
```
_________________________________________________________________
Layer (type) Output Shape Param #
=================================================================
embedding_4 (Embedding) (None, None, 32) 320000
_________________________________________________________________
simple_rnn_6 (SimpleRNN) (None, 10) 430
=================================================================
Total params: 320,430 Trainable params: 320,430 Non-trainable
params: 0
```



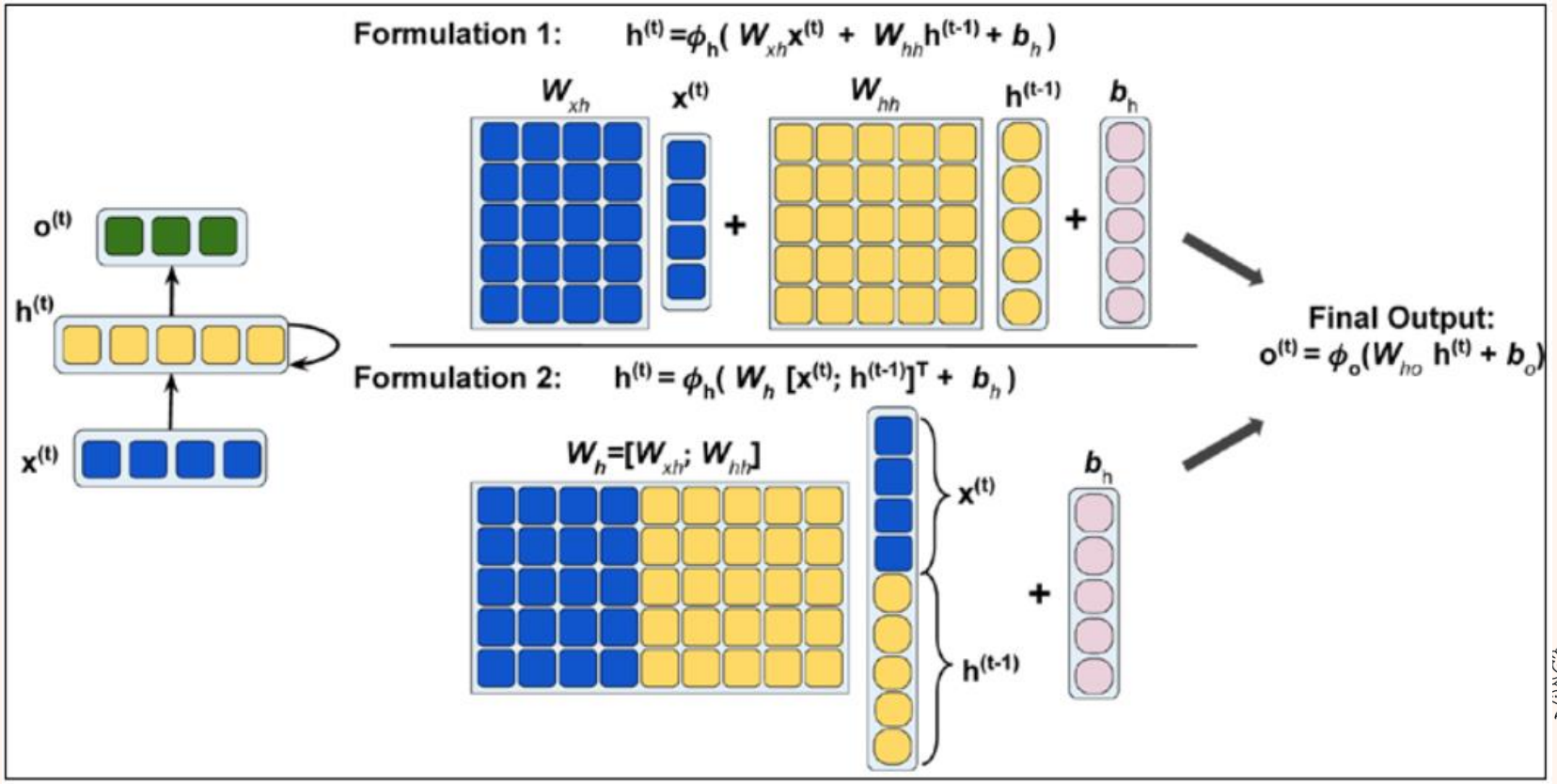
مثال IMDB

```
from keras.layers import Dense

model = Sequential()
model.add(Embedding(10000, 32, 5000))
model.add(SimpleRNN(32))
model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer='rmsprop',
loss='binary_crossentropy', metrics=['acc'])
history = model.fit(input_train, y_train,
                    epochs=10,
                    batch_size=128,
                    validation_split=0.2)
```



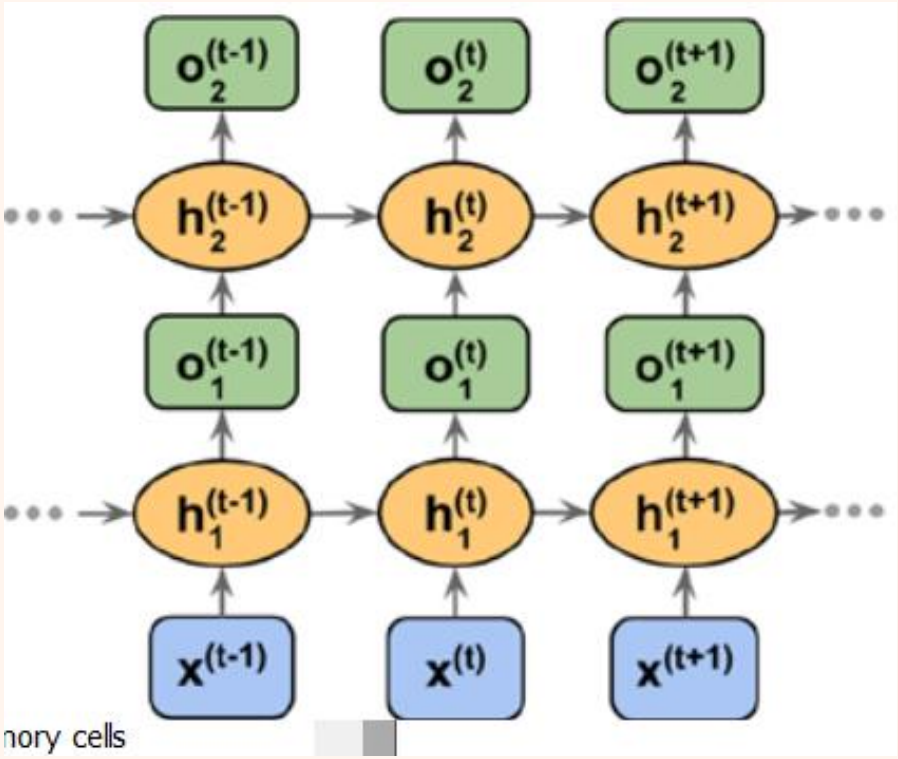


***Raschka, S., and V. Mirjalili. Python Machine Learning: Machine Learning and Deep Learning with Python, Scikit-Learn, and Tensorflow 2, 3rd Edition*. Packt Publishing, 2019.**



Raschka, S., and V. Mirjalili. Python Machine Learning: Machine Learning and Deep Learning with Python, Scikit-Learn, and Tensorflow 2, 3rd Edition. Packt Publishing, 2019.

Backpropagation through time



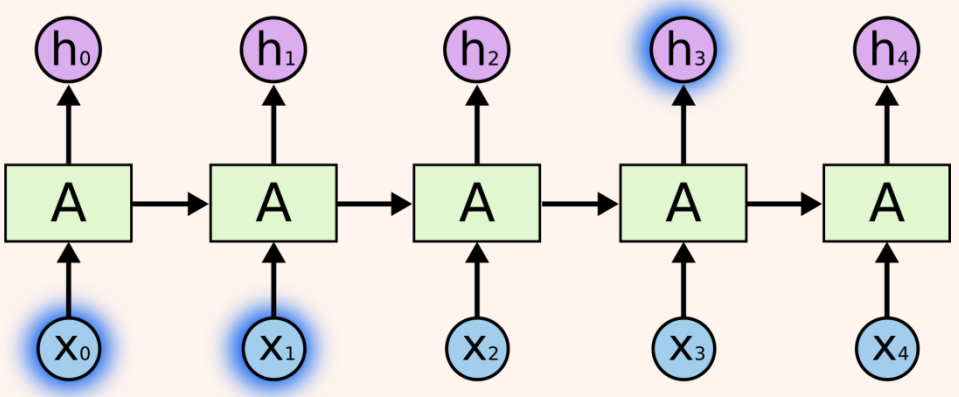
Truncated Backpropagation through time

- در این شیوه بازگشت در زمان و اصلاح وزن‌ها به صورت محدود تا بازه‌ای خاص انجام می‌شود

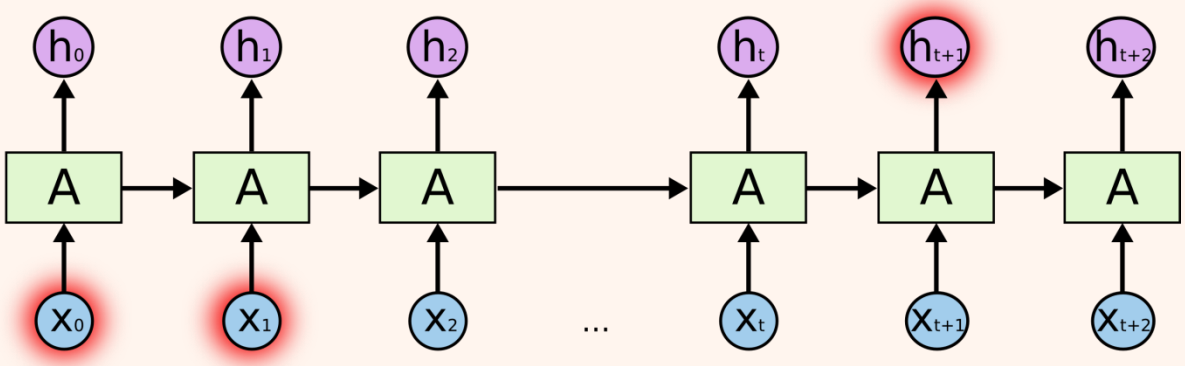
Raschka, S., and V. Mirjalili. Python Machine Learning: Machine Learning and Deep Learning with Python, Scikit-Learn, and Tensorflow 2, 3rd Edition. Packt Publishing, 2019.

وابستگی‌های Long-term

“the clouds are in the *sky*,”



“I grew up in France… I speak fluent *French*.”

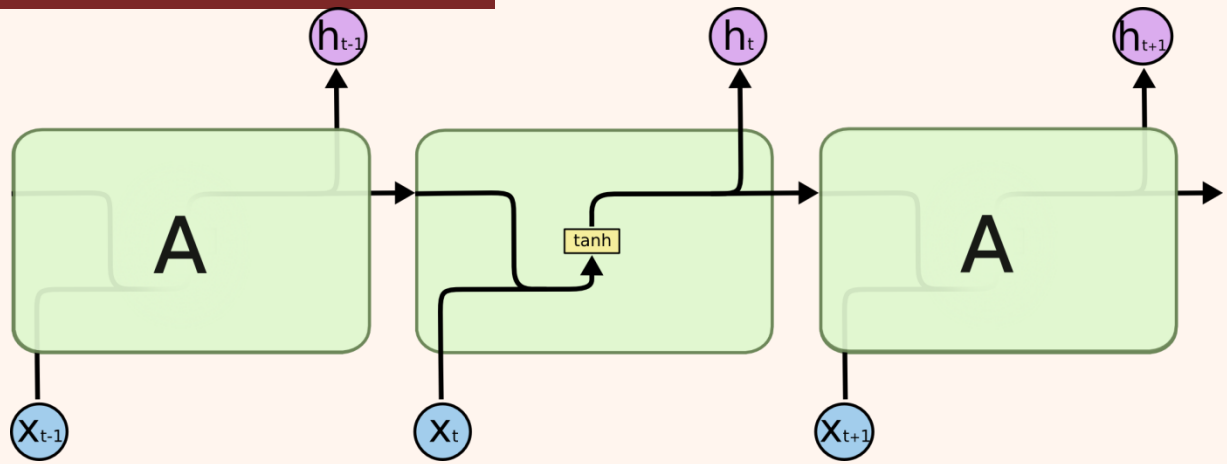


In theory, RNNs are absolutely capable of handling such “long-term dependencies.” A human could carefully pick parameters for them to solve toy problems of this form. Sadly, in practice, RNNs don’t seem to be able to learn them.

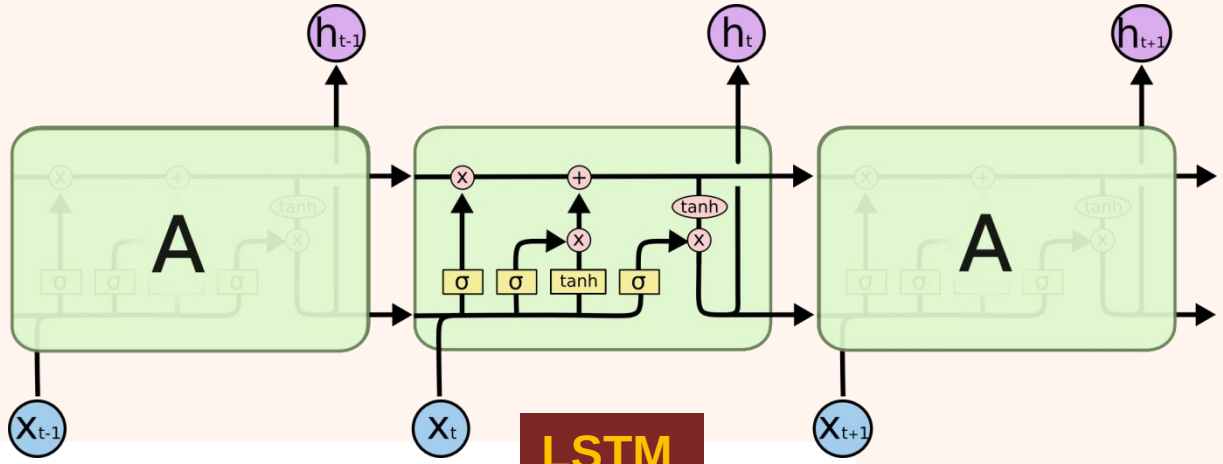
LSTM

حافظه کوتاه‌مدت ماندگار

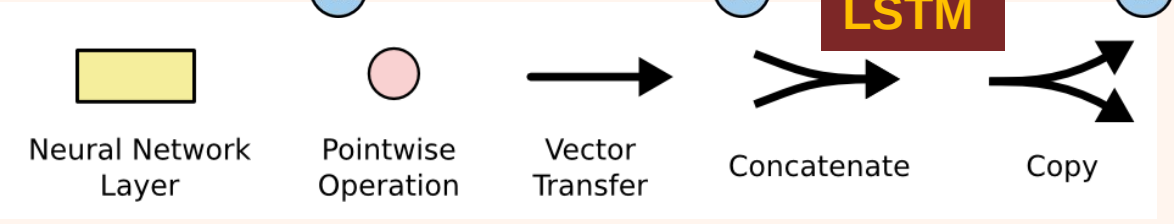
Long Short Term Memory networks



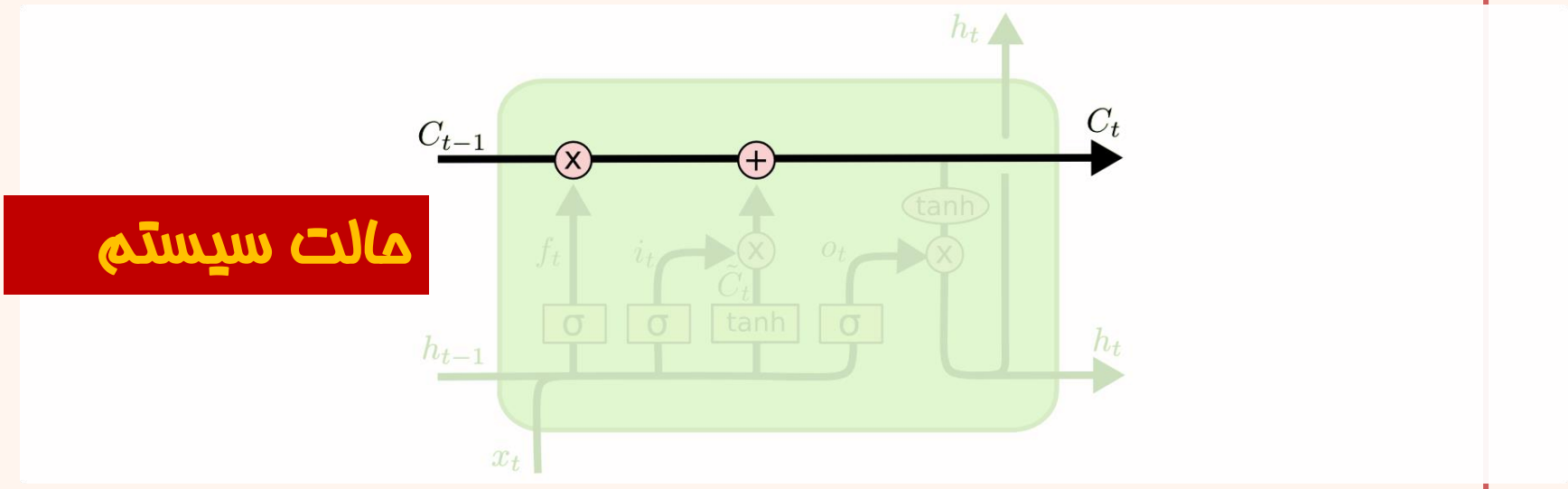
standard RNN



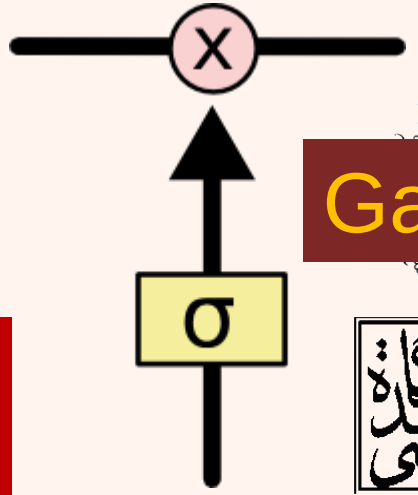
LSTM



cell state



حالت سیستم

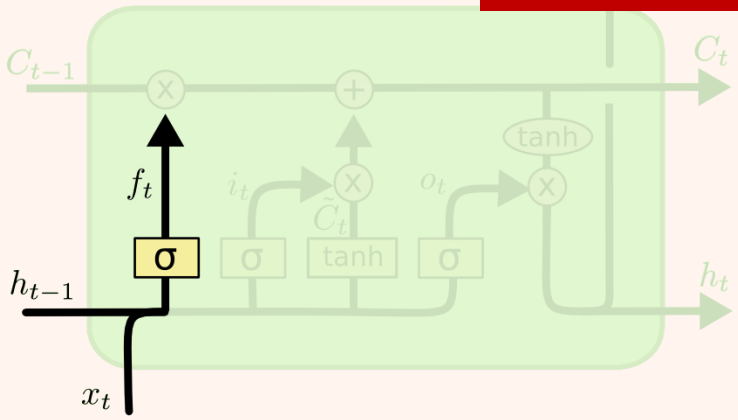


Gates

مشخص می‌کند چه بخشی از اطلاعات می‌تواند عبور کنند.

قدم اول: از حالت قبلی چه اطلاعاتی حفظ شود

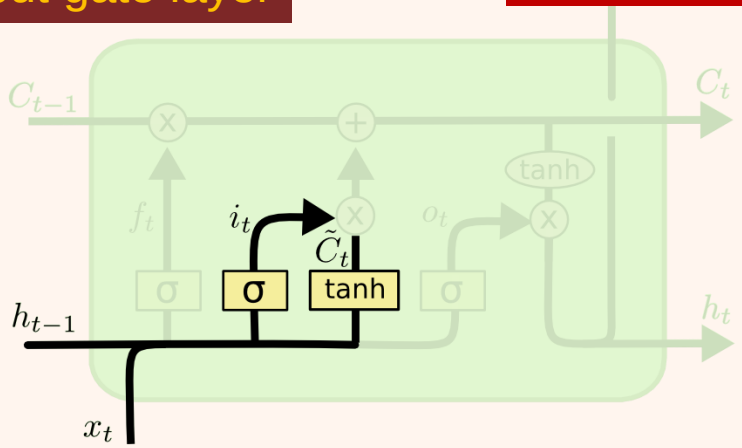
forget gate layer



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

گام بعدی: چه اطلاعاتی به حالت جدید اضافه شود

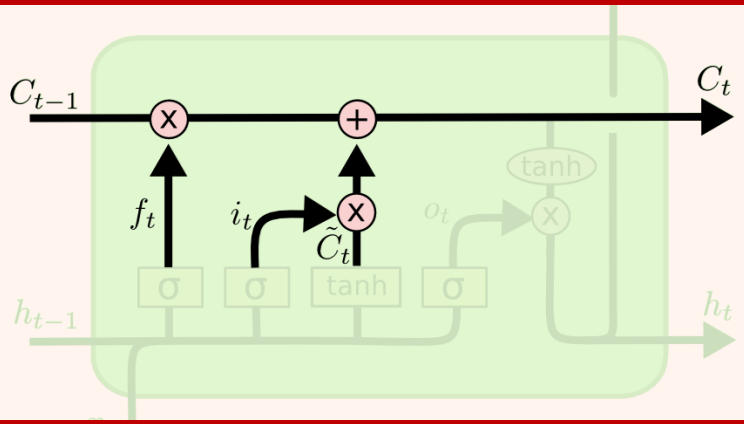
input gate layer



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

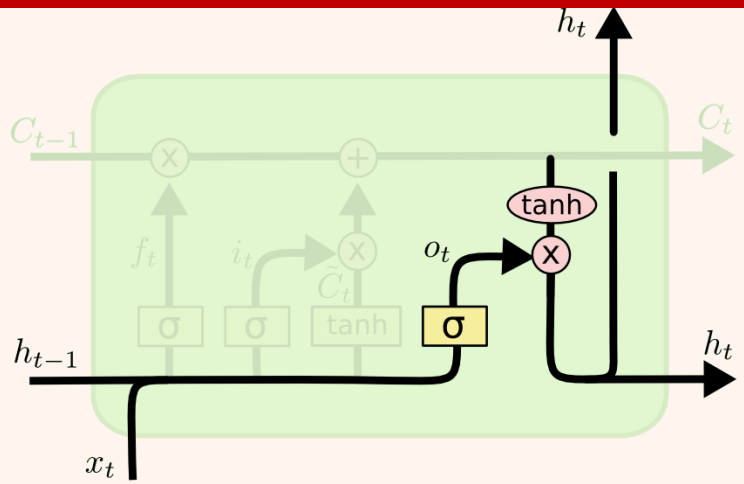
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

بر این مبنا حالت جدید به روز می‌شود



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

خروجی بر اساس ورودی و حالت به روز شده محاسبه می‌شود

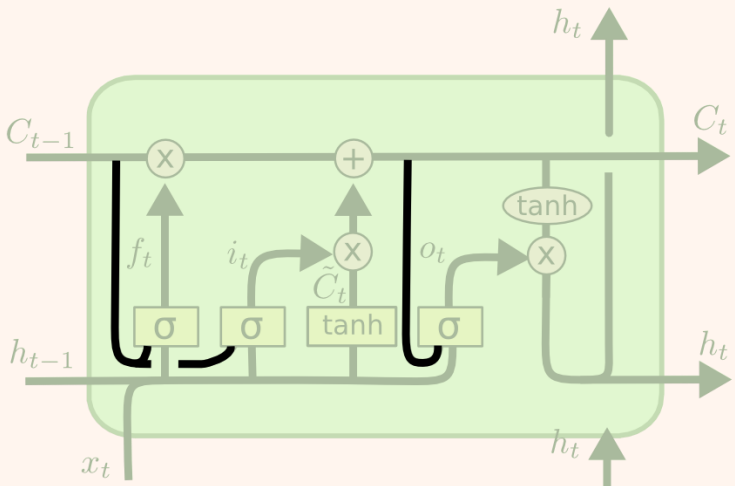


$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

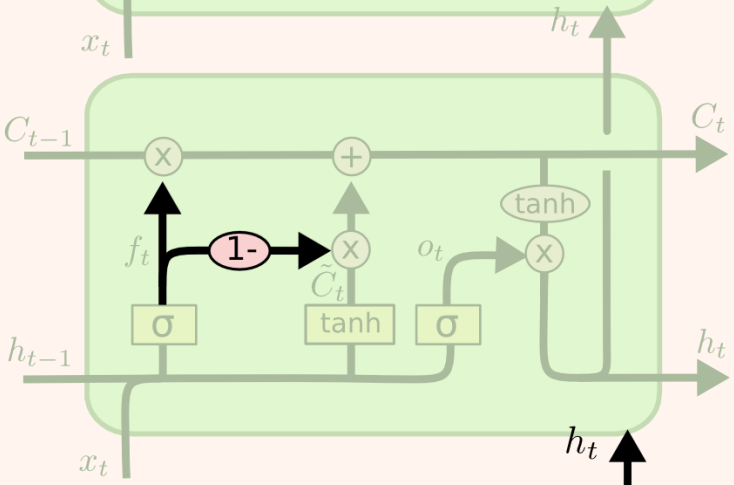
$$h_t = o_t * \tanh\left(C_t\right)$$

می‌توان گفت که lstm هم به نوعی از ایده bypass استفاده می‌کند.

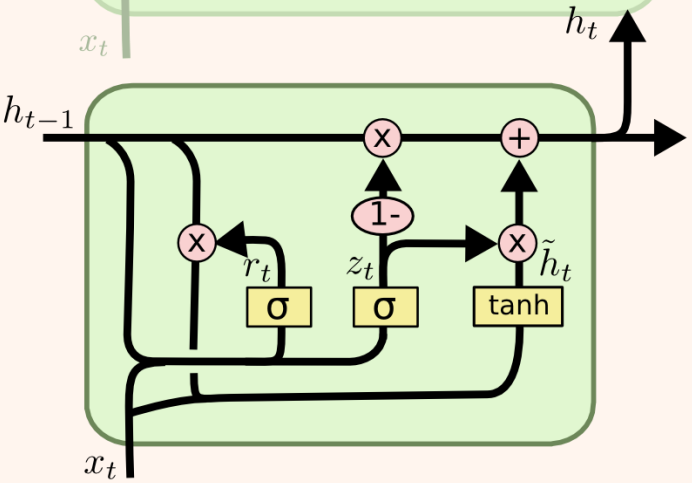
نمونه‌های دیگر LSTM



$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \; + \; b_o\right)$$



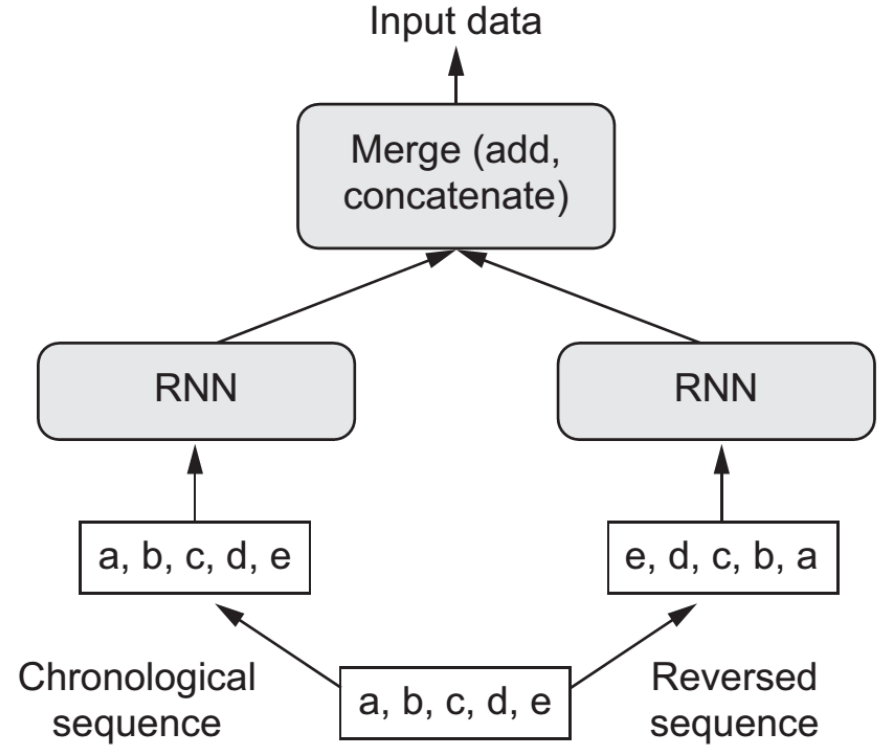
$$C_t = f_t * C_{t-1} + (1 - \boldsymbol{f_t}) * \tilde{C}_t$$



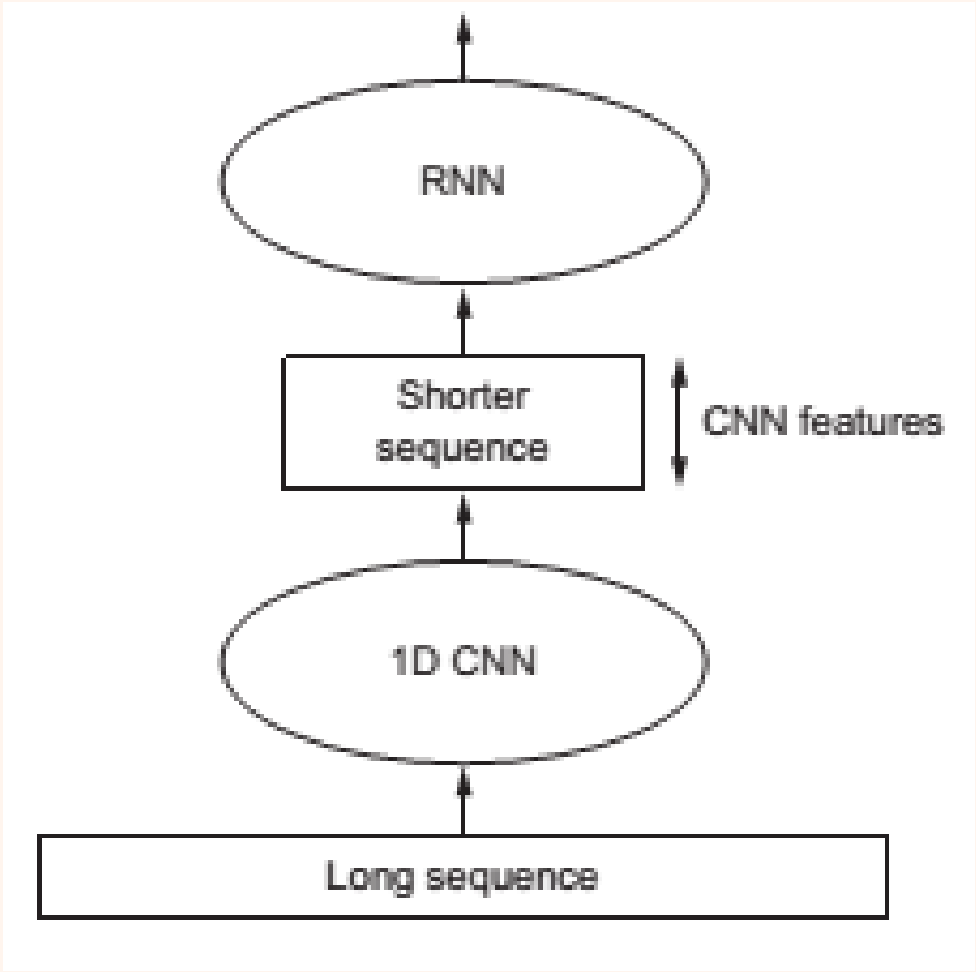
Gated Recurrent Unit, or GRU, introduced by Cho, et al. (2014)

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

bidirectional RNNs



ترکیب کانولوشن یک‌بعدی و شبکه‌ی بازگشت‌کننده



Reference window

Jane went to Africa last September, and enjoyed the culture and met many wonderful people; she came back raving about how wonderful her trip was, and is tempting me to go too.

RNN

Jane went to Africa last September, and enjoyed the culture and met many wonderful people; she came back raving about how wonderful her trip was, and is tempting me to go too.

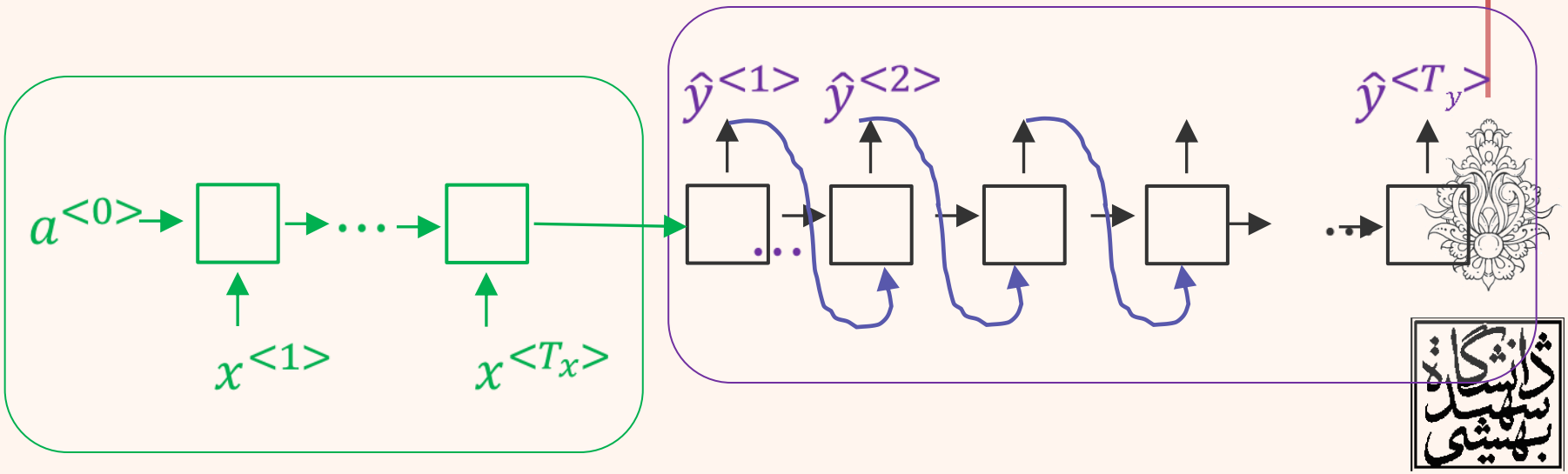
GRU and LSTM

Jane went to Africa last September, and enjoyed the culture and met many wonderful people; she came back raving about how wonderful her trip was, and is tempting me to go too.

Transformer

به واسطه‌ی بررسی گام به گام با افزایش طول جمله محاسبات بیشتری خواهیم داشت.

NO Parallelism



The fall of RNN / LSTM

شبکه بازگشت‌کننده hardware-friendly نیستند.

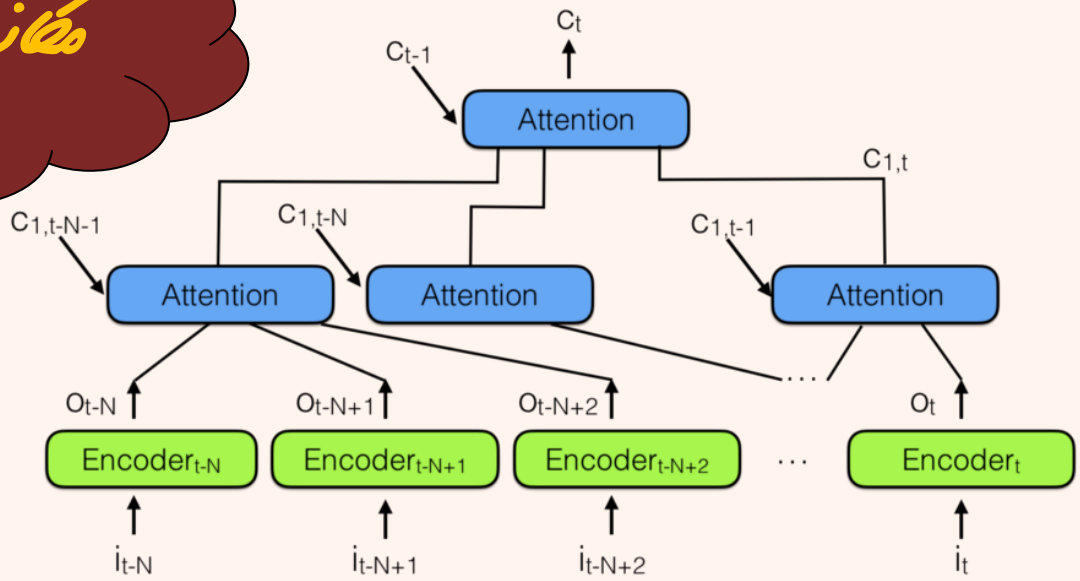
مقیاس‌پذیر نیستند

پردازش ترتیبی ← پردازش موازی

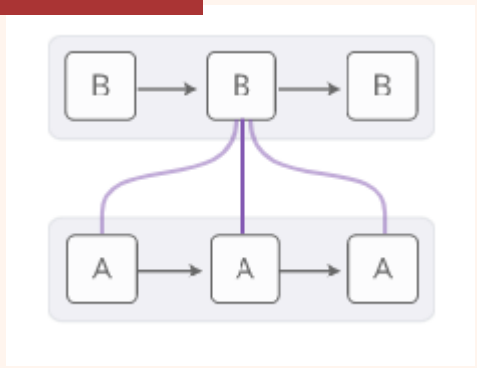
نگاه رو به جلو/نگاه رو به عقب

نحوه ترکیب؟؟؟

مکانیزم توجه

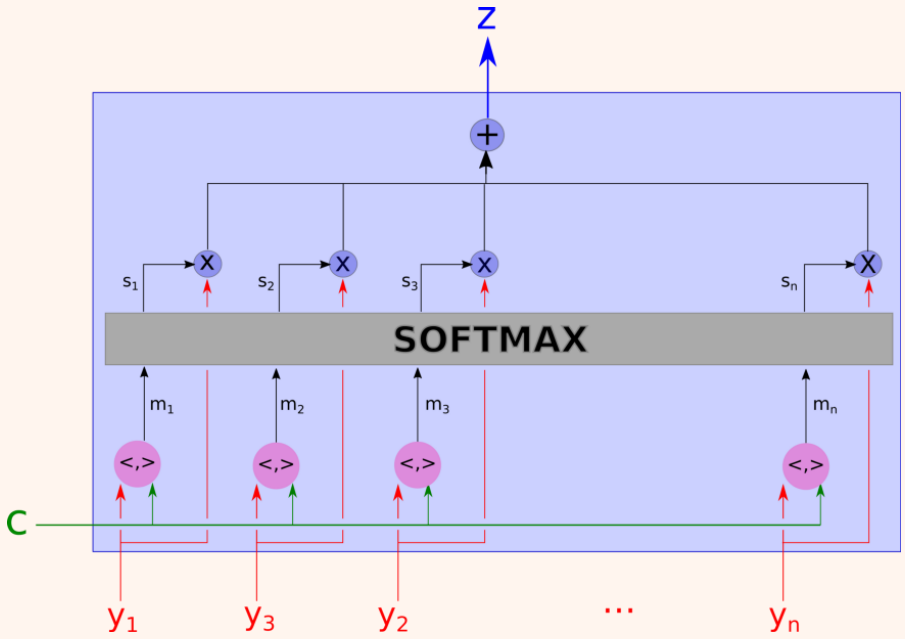


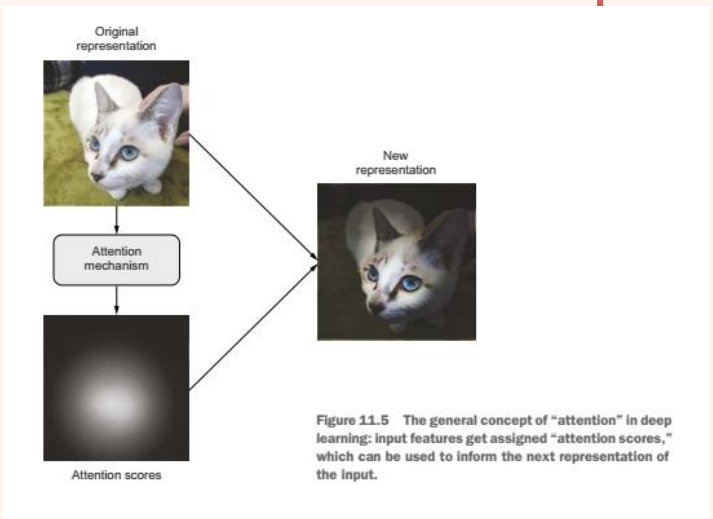
Attention mechanism



کدام بخش‌ها از داده مورد توجه بیشتری قرار گیرد.

https://distill.pub/2016/augmented-rnns/





https://towardsdatascience.com/memory-attention-sequences-37456d271992

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

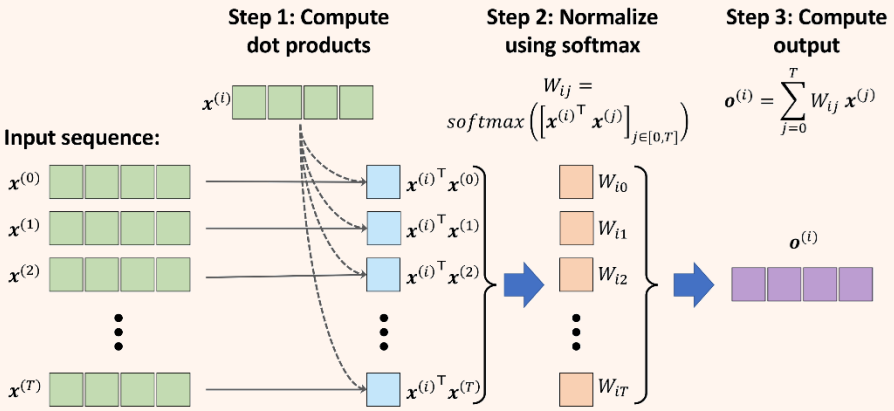
Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

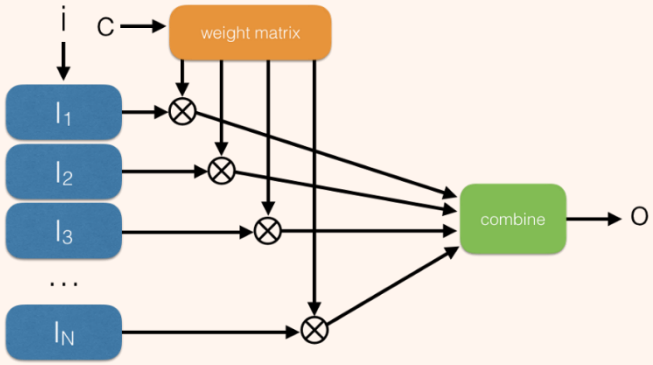
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

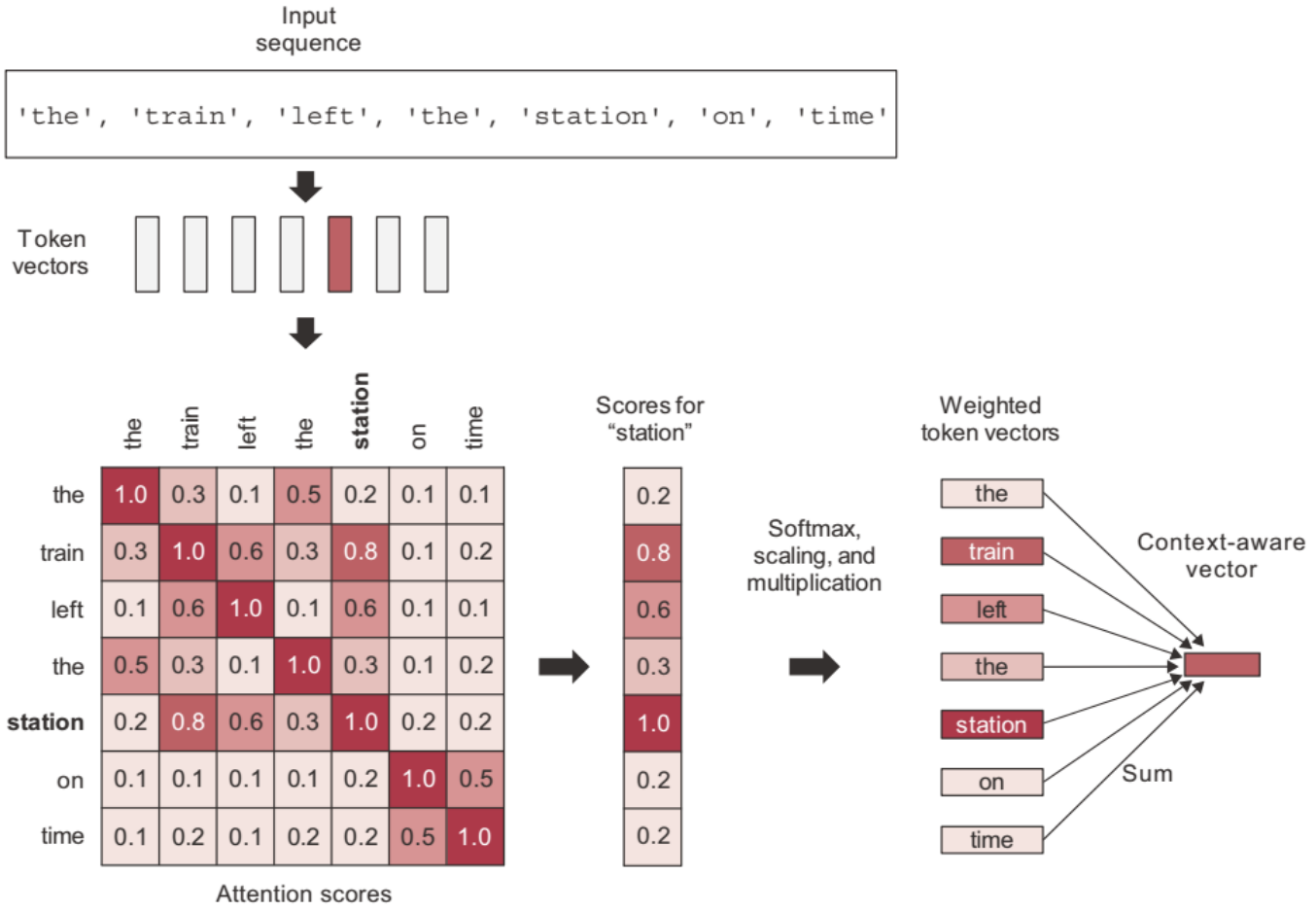
Illia Polosukhin* ‡ illia.polosukhin@gmail.com



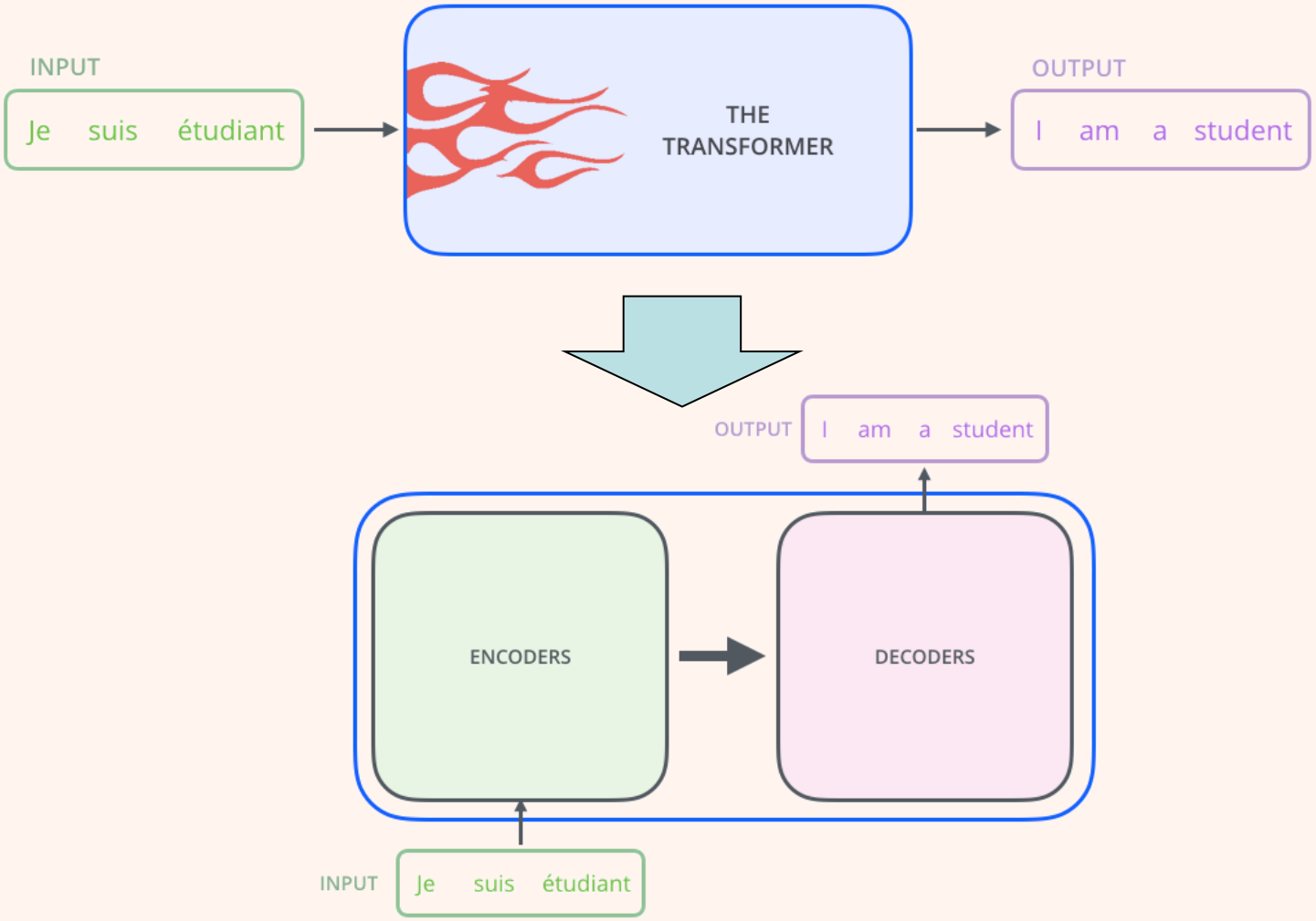
هدف self-attention مدل کردن ارتباط مولفه‌های مختلف خروجی با مولفه‌های مختلف ورودی است.



مثال

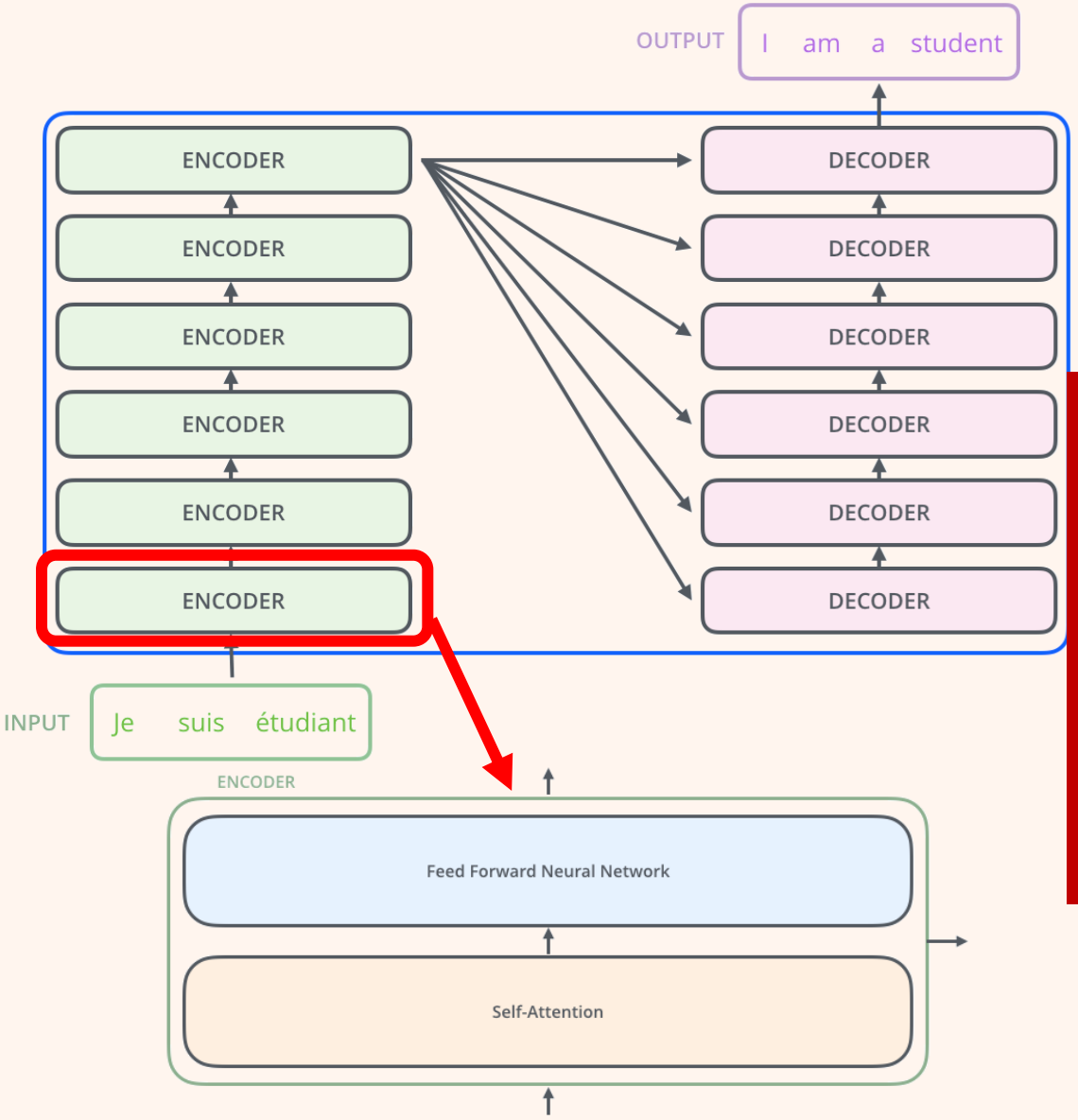

Input sequence
'the', 'train', 'left', 'the', 'station', 'on', 'time'
Token vectors
the train left the station on time
the 1.0 0.3 0.1 0.5 0.2 0.1 0.1
train 0.3 1.0 0.6 0.3 0.8 0.1 0.2
left 0.1 0.6 1.0 0.1 0.6 0.1 0.1
the 0.5 0.3 0.1 1.0 0.3 0.1 0.2
station 0.2 0.8 0.6 0.3 1.0 0.2 0.2
on 0.1 0.1 0.1 0.1 0.2 1.0 0.5
time 0.1 0.2 0.1 0.2 0.2 0.5 1.0
Attention scores
Scores for "station"
0.2
0.8
0.6
0.3
1.0
0.2
0.2
Softmax, scaling, and multiplication
Weighted token vectors
the
train
left
the
station
on
time
Context-aware vector
Sum

Transformer

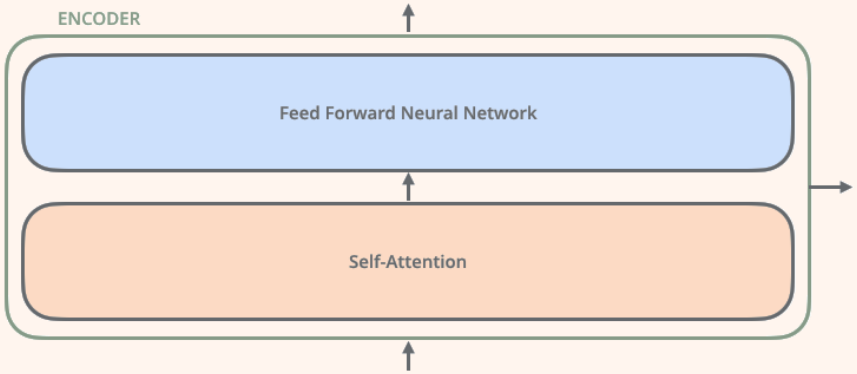


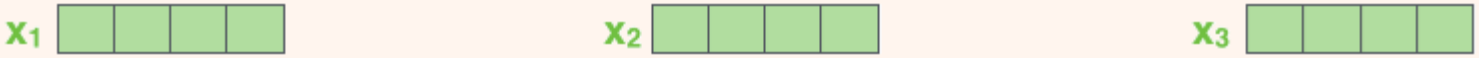
در این جا نیز ساختار کدگذار-کدگشا وجود دارد.

http://jalammar.github.io/illustrated-transformer/

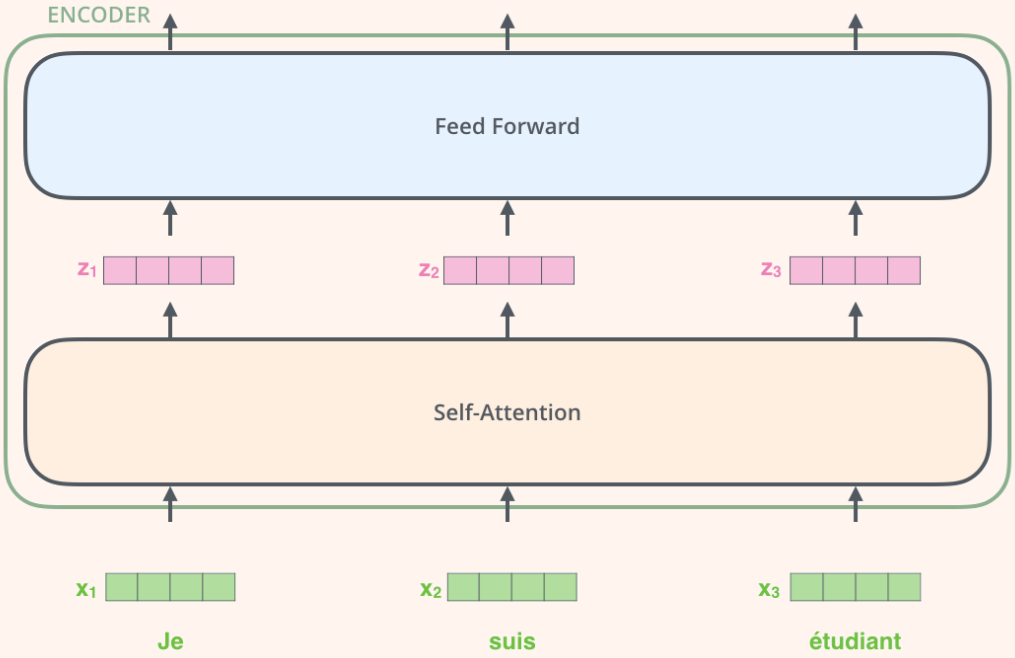


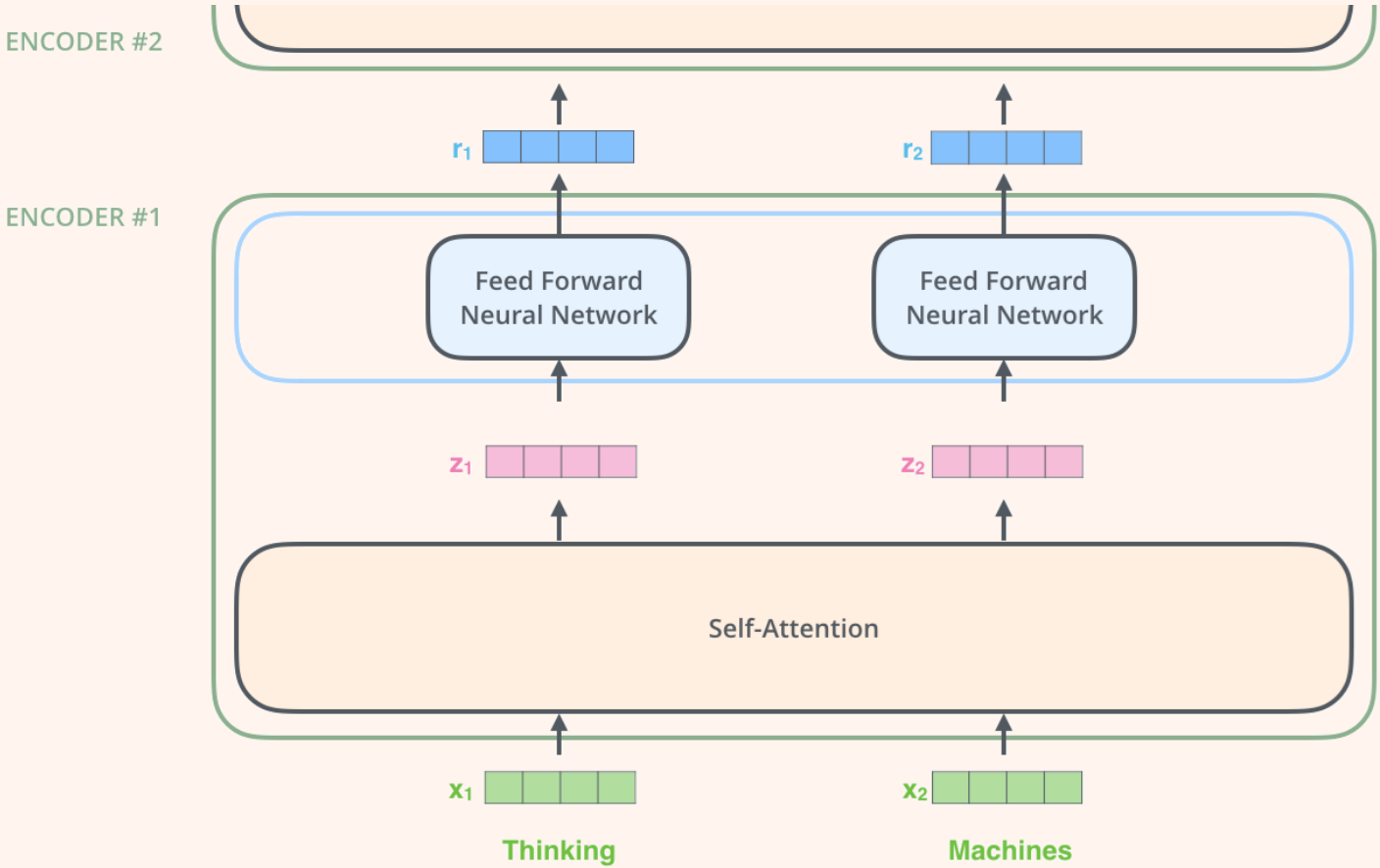
لایه self-attention به کدگذار کمک می‌کند تا هنگام بررسی یک کلمه، سایر کلمات و ارتباط آن‌ها را در نظر بگیرد.

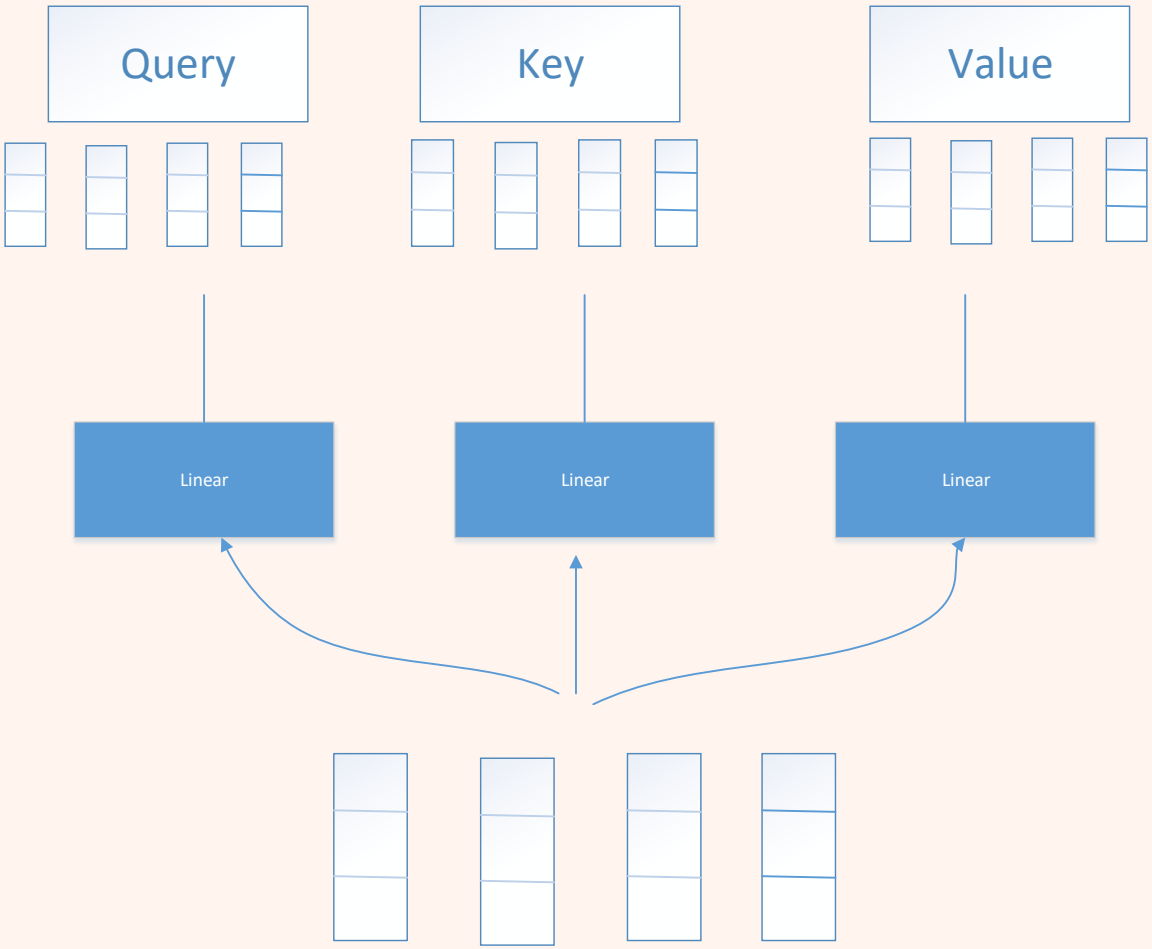




Je suis étudiant

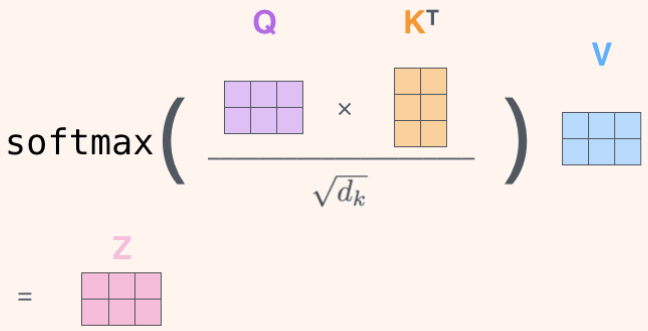

ENCODER
Feed Forward
z1
z2
z3
Self-Attention
x1
x2
x3
Je
suis
étudiant

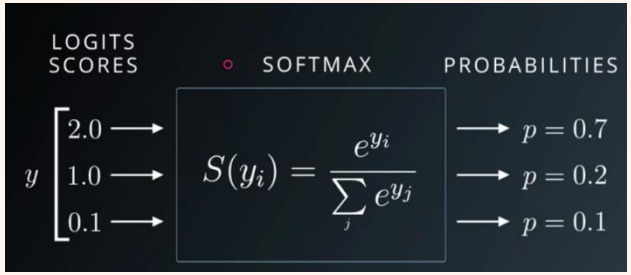

ENCODER #2
ENCODER #1
r1
r2
Feed Forward Neural Network
Feed Forward Neural Network
z1
z2
Self-Attention
x1
x2
Thinking
Machines

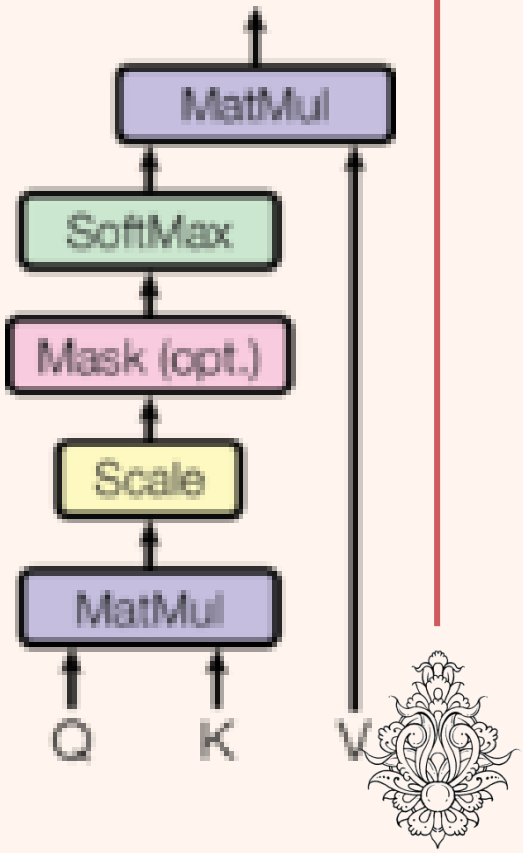

Query
Key
Value
Linear
Linear
Linear

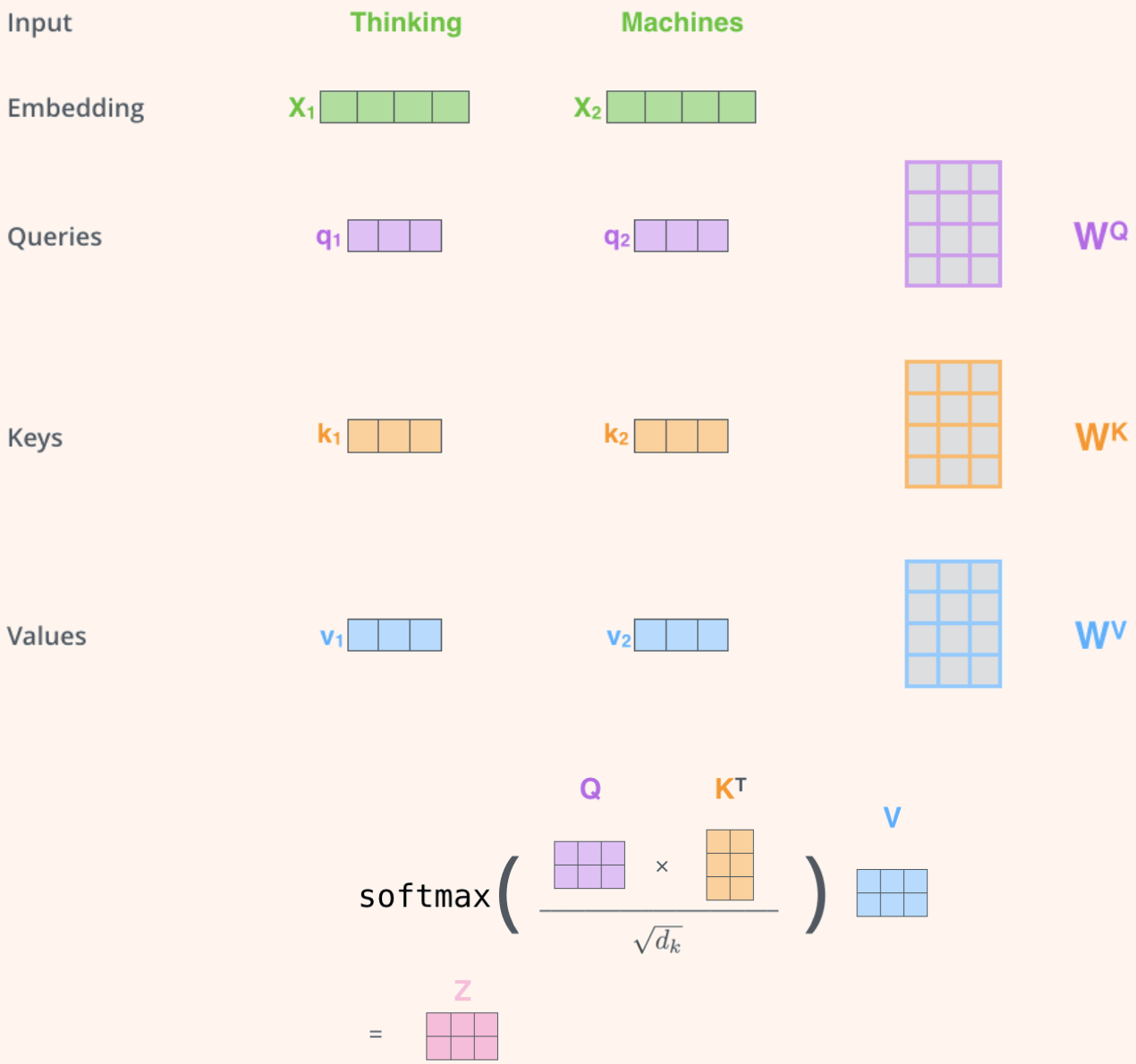
Scaled Dot-Product Attention

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

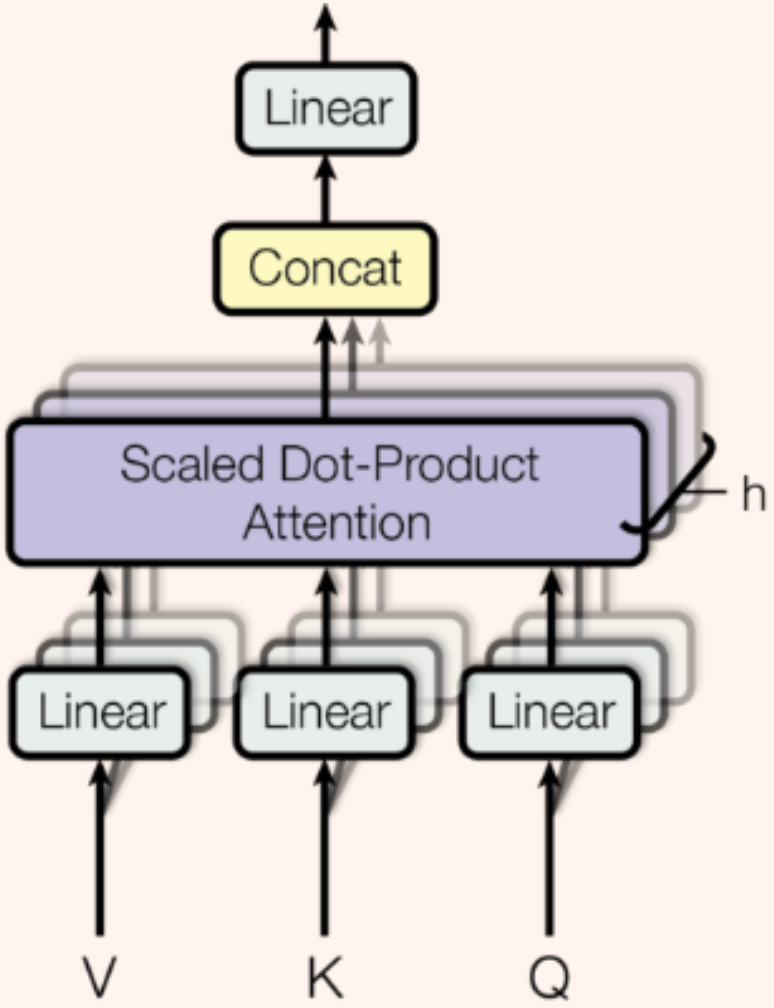




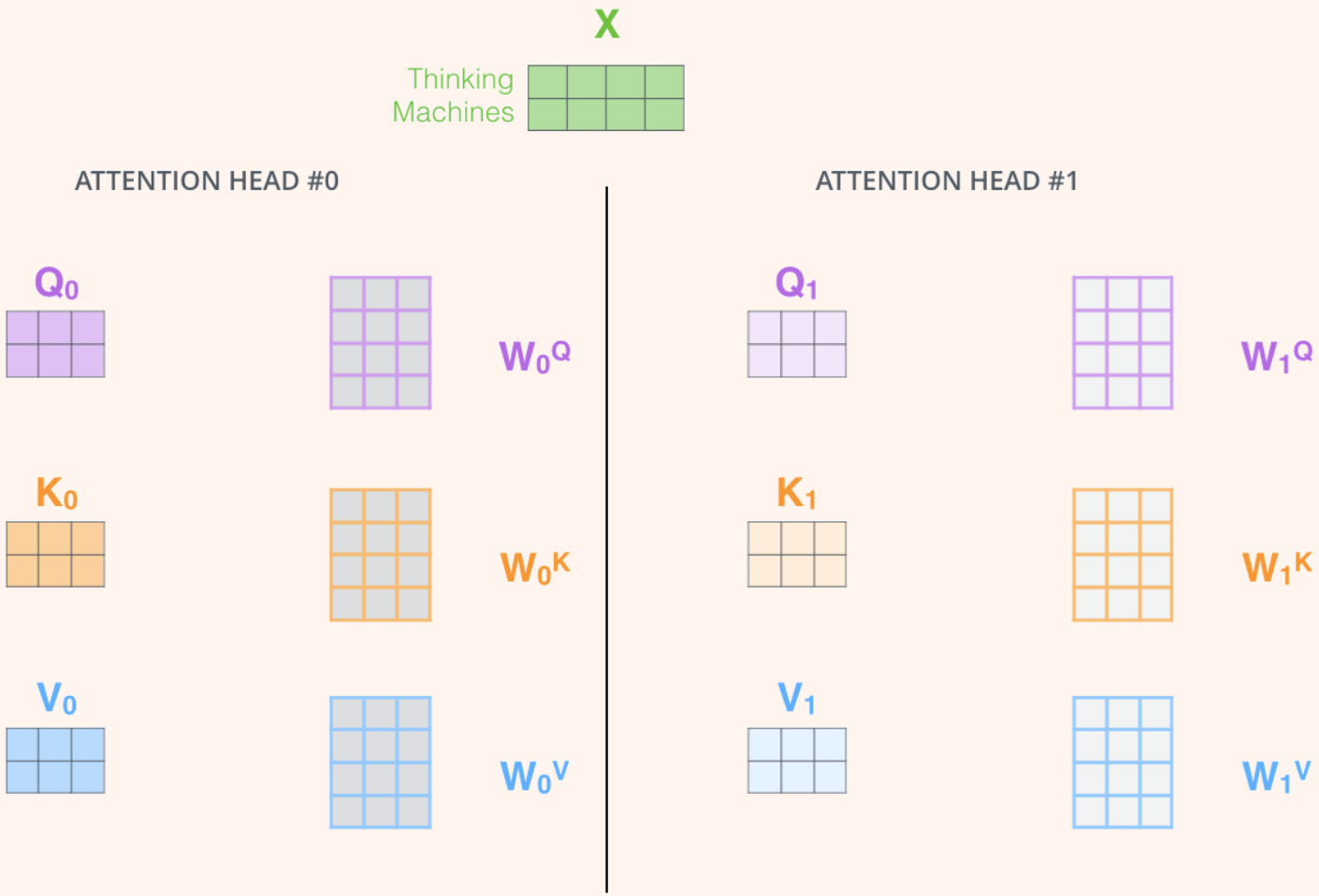



Input
Thinking
Machines
Embedding
X_1
X_2
Queries
q_1
q_2
W^Q
Keys
k_1
k_2
W^K
Values
v_1
v_2
W^V
$\text{softmax}\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) V$
$= Z$

Multi-head attention



multi-headed attention



1) This is our input sentence*

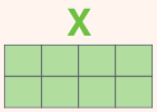
2) We embed each word*

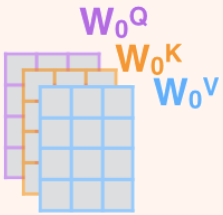
3) Split into 8 heads. We multiply X or R with weight matrices

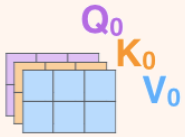
4) Calculate attention using the resulting Q/K/V matrices

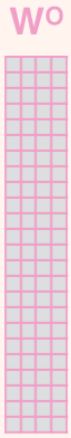
5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer

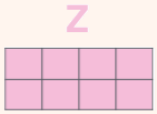
Thinking Machines





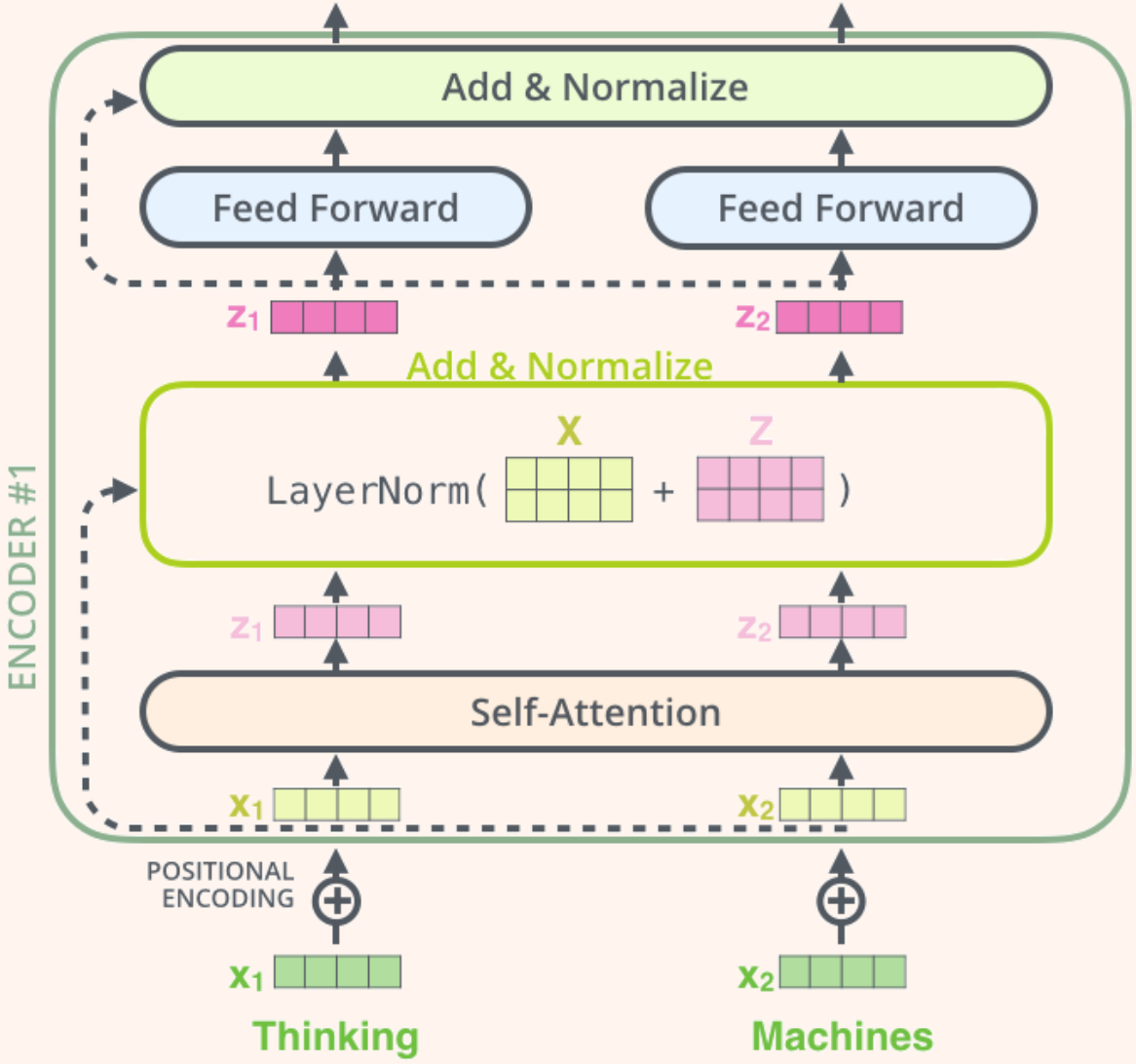




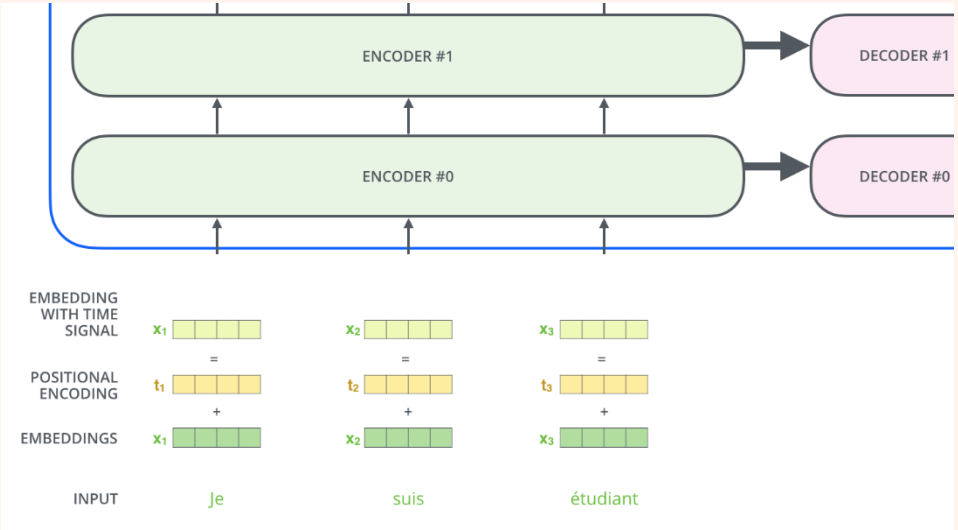


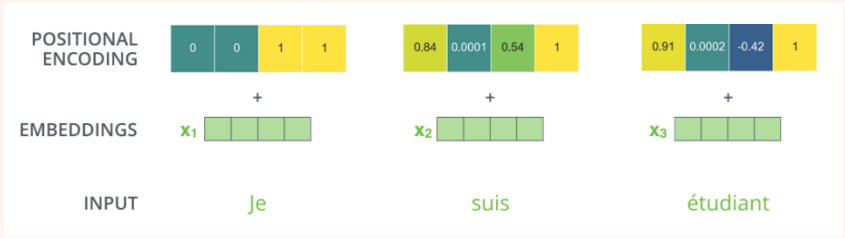
* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one

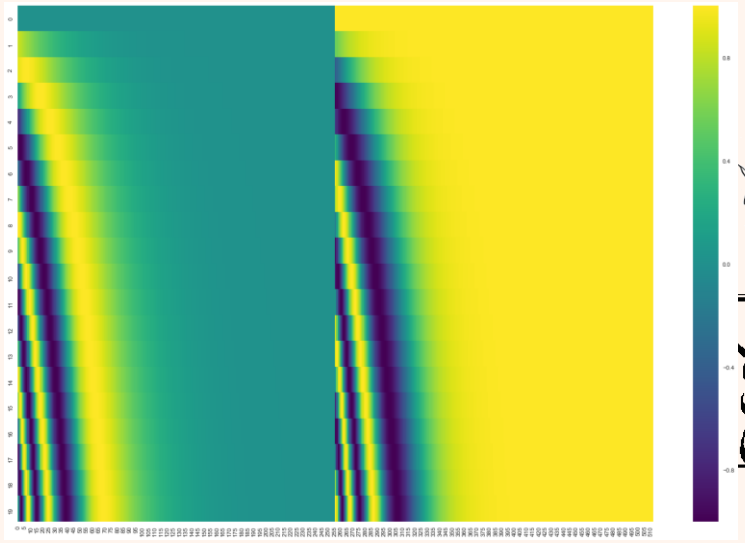
The Residuals



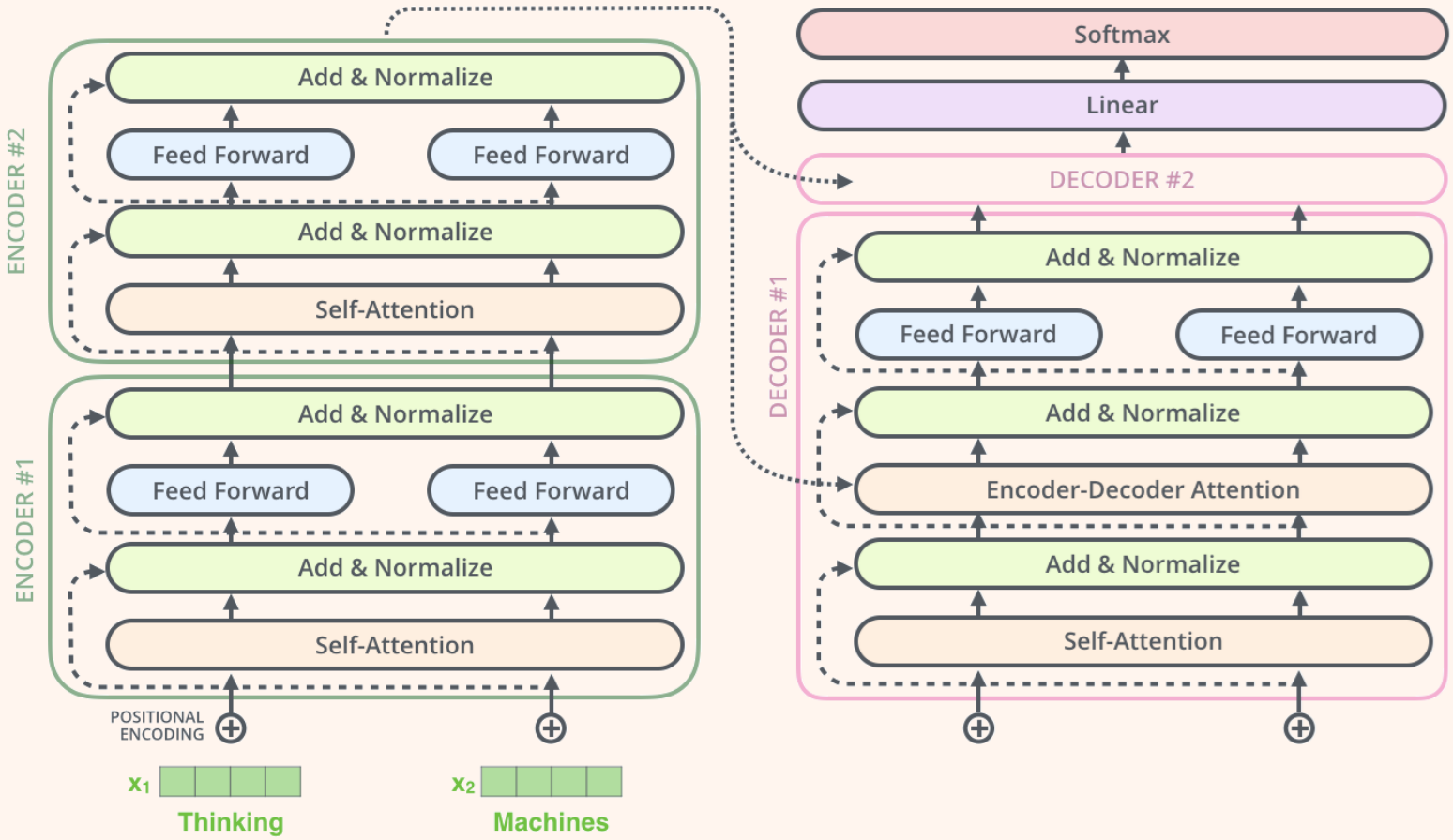
Positional Encoding

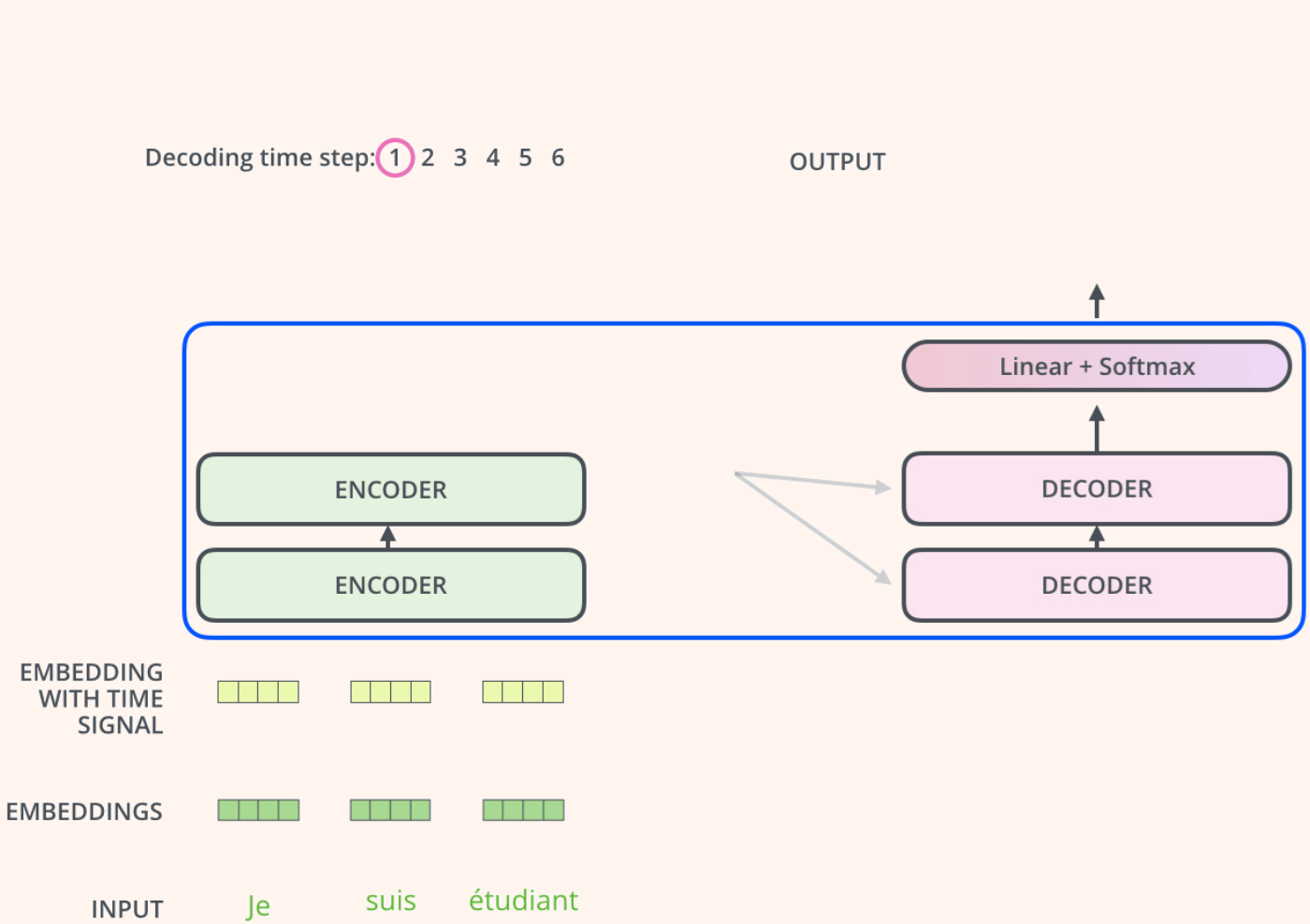


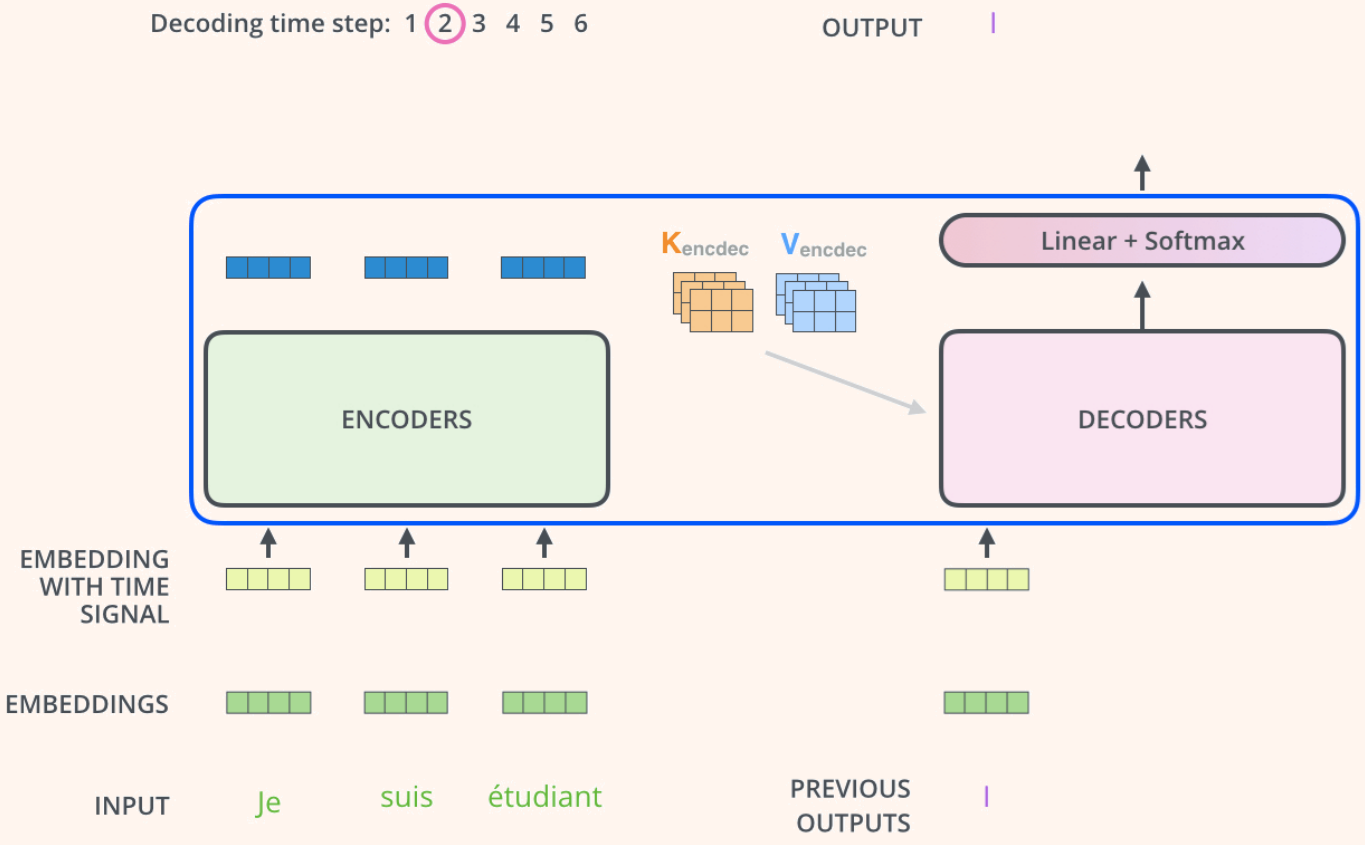




The Decoder Side




Decoding time step: 1 2 3 4 5 6
OUTPUT
Linear + Softmax
ENCODER
ENCODER
DECODER
DECODER
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant

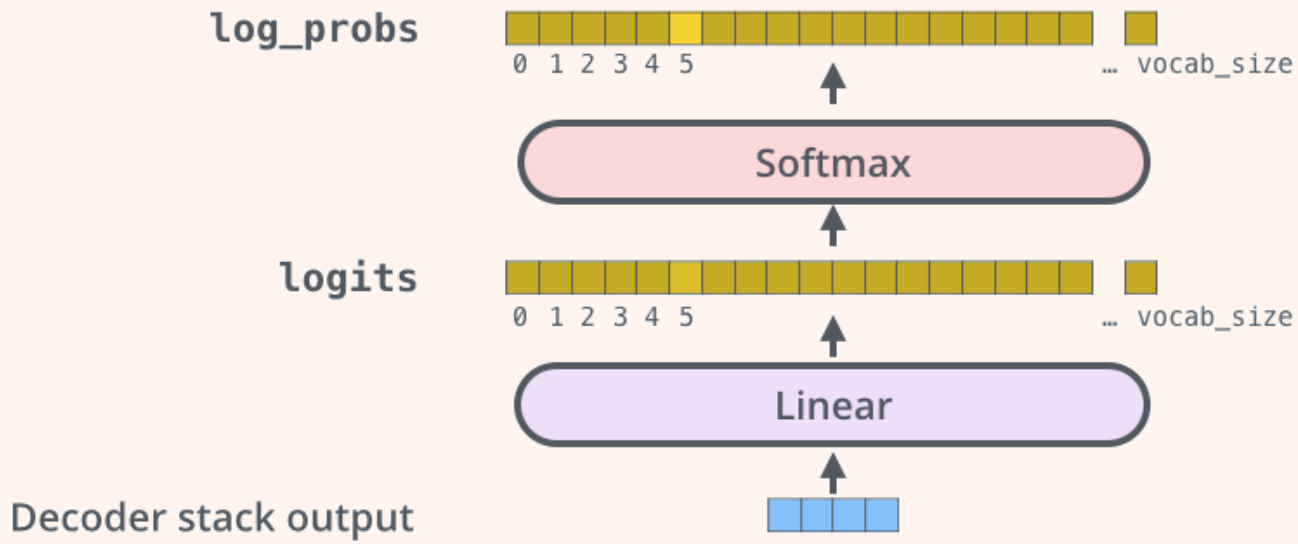

Decoding time step: 1 2 3 4 5 6
OUTPUT
I
Linear + Softmax
K encdec
V encdec
ENCODERS
DECODERS
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant
PREVIOUS OUTPUTS
I

Which word in our vocabulary is associated with this index?

am

Get the index of the cell with the highest value **(argmax)**

5



Published as a conference paper at ICLR 2021

AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE

Alexey Dosovitskiy*,†, Lucas Beyer*, Alexander Kolesnikov*, Dirk Weissenborn*,
Xiaohua Zhai*, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer,
Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, Neil Houlsby*,†
*equal technical contribution, †equal advising
Google Research, Brain Team
{adosovitskiy, neilhoulsby}@google.com