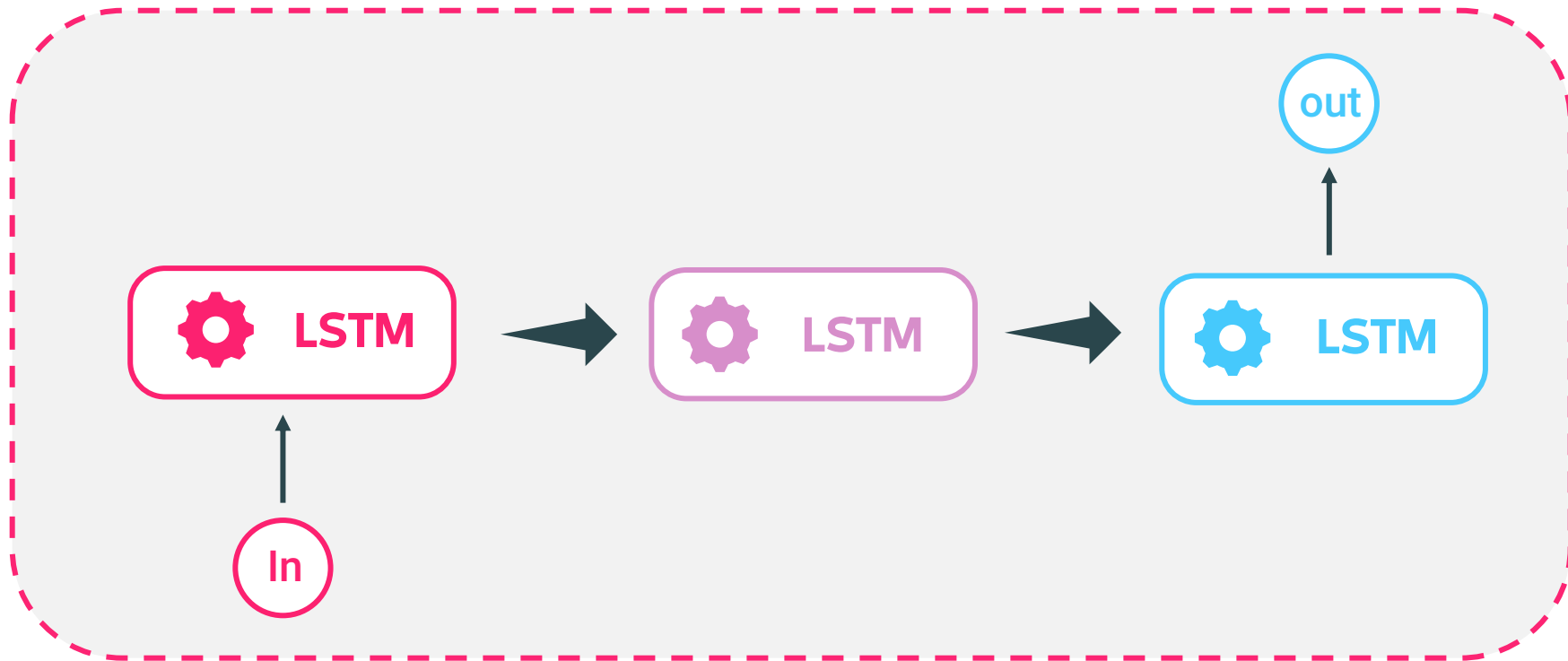


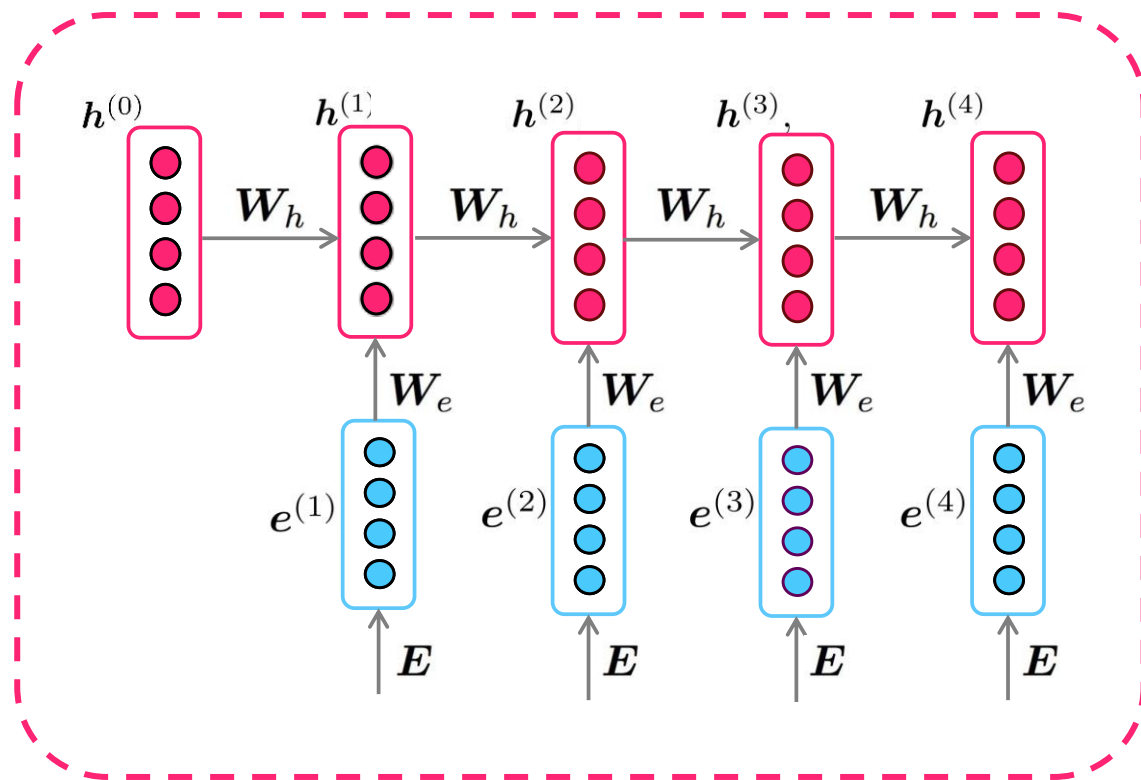
آکادمی رباتک

دوره یادگیری عمیق پیشرفته - شبکه های بازگشتی

جلسه دوم: بزرگی به نام LSTM

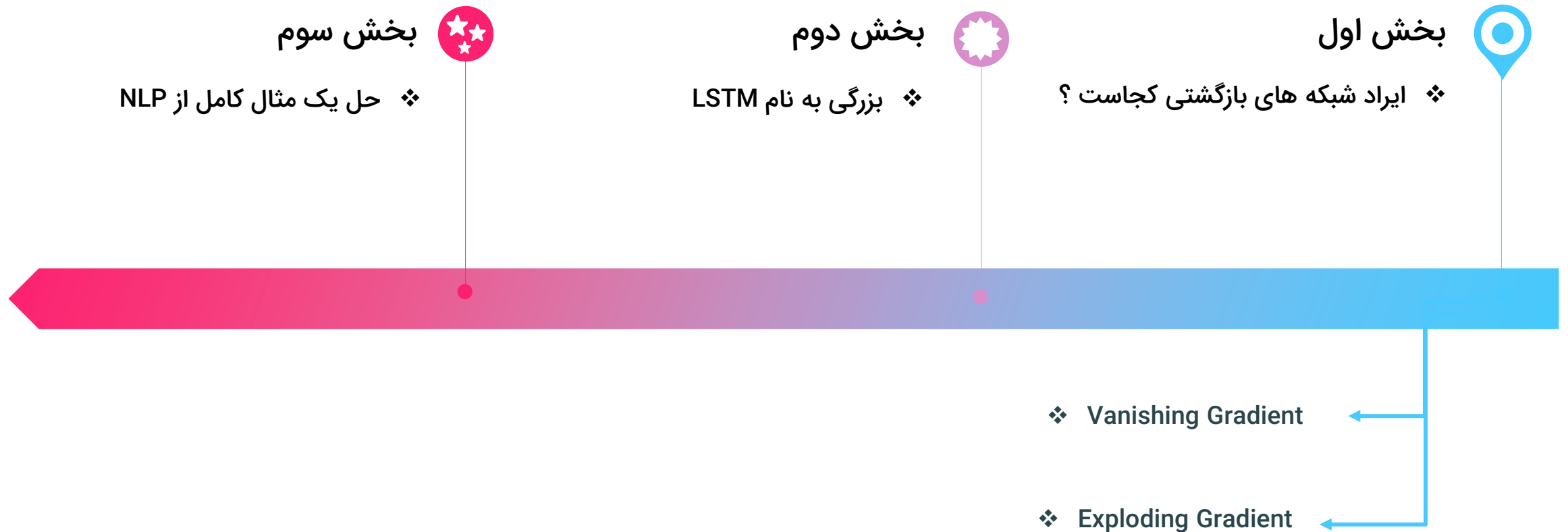


آنچه گذشت:

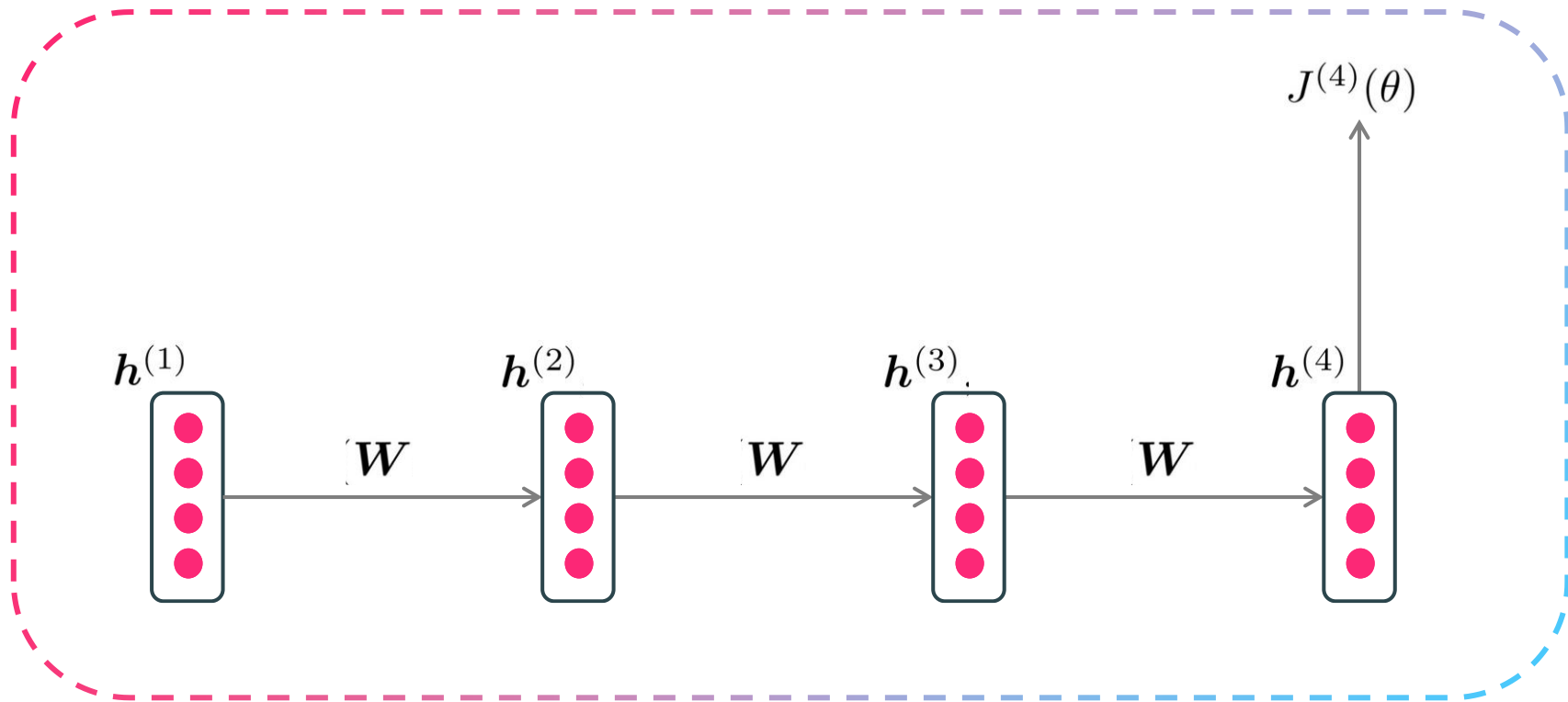


$$h^{(t)} = \sigma(W_h h^{(t-1)} + W_e e^{(t)} + b_1)$$

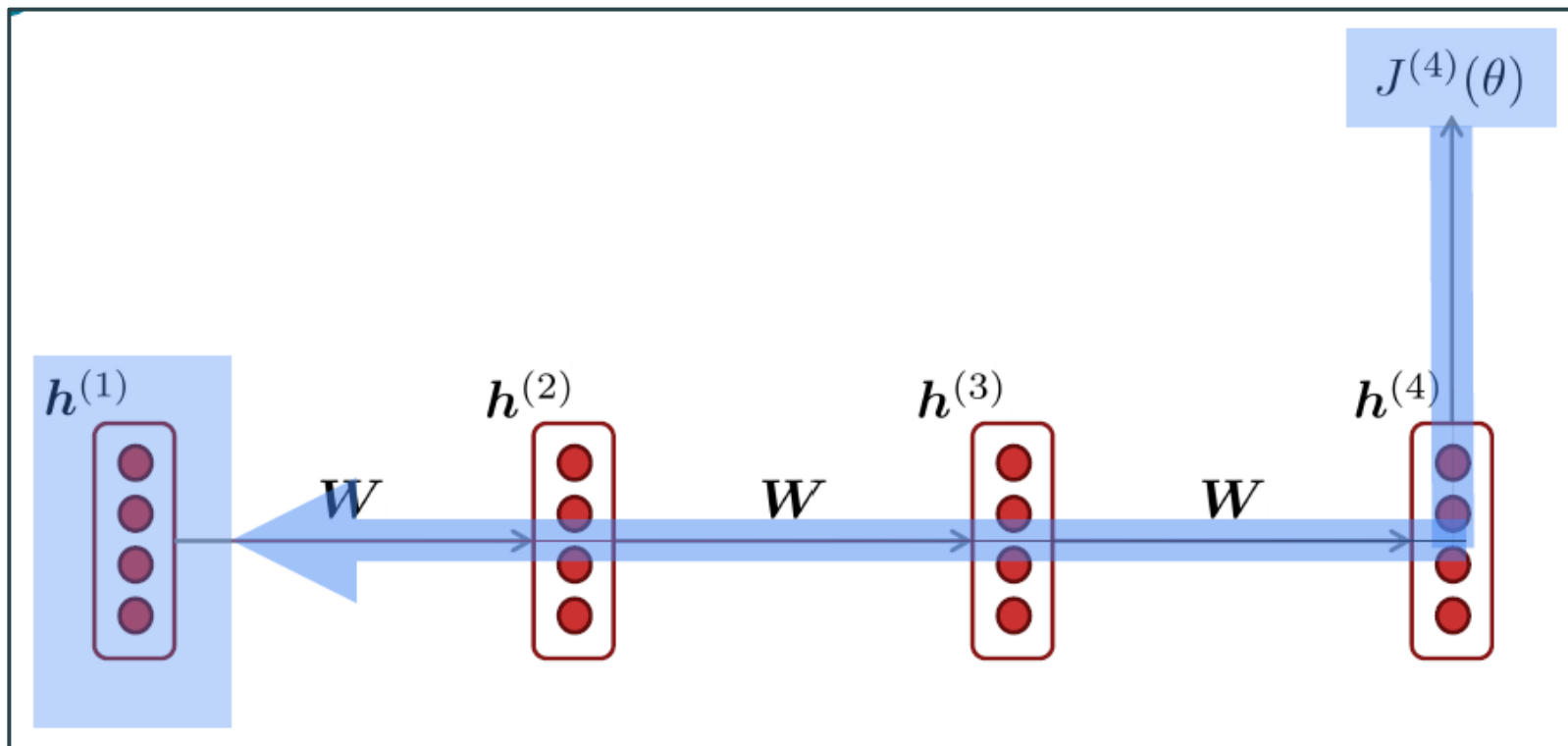
آنچه در این جلسه گفته خواهد شد :



RNN ها چه ایراداتی دارند؟

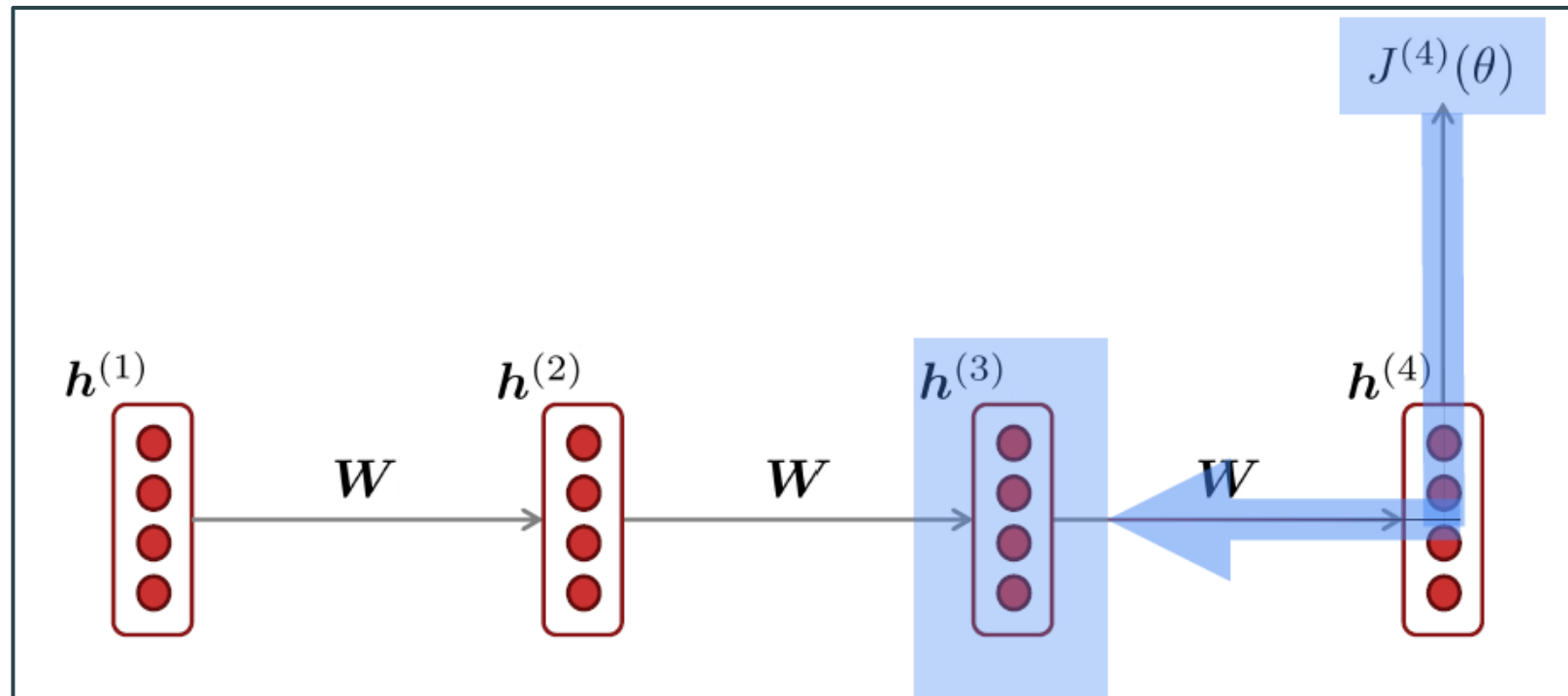


مشکلی بزرگ به نام Vanishing Gradient



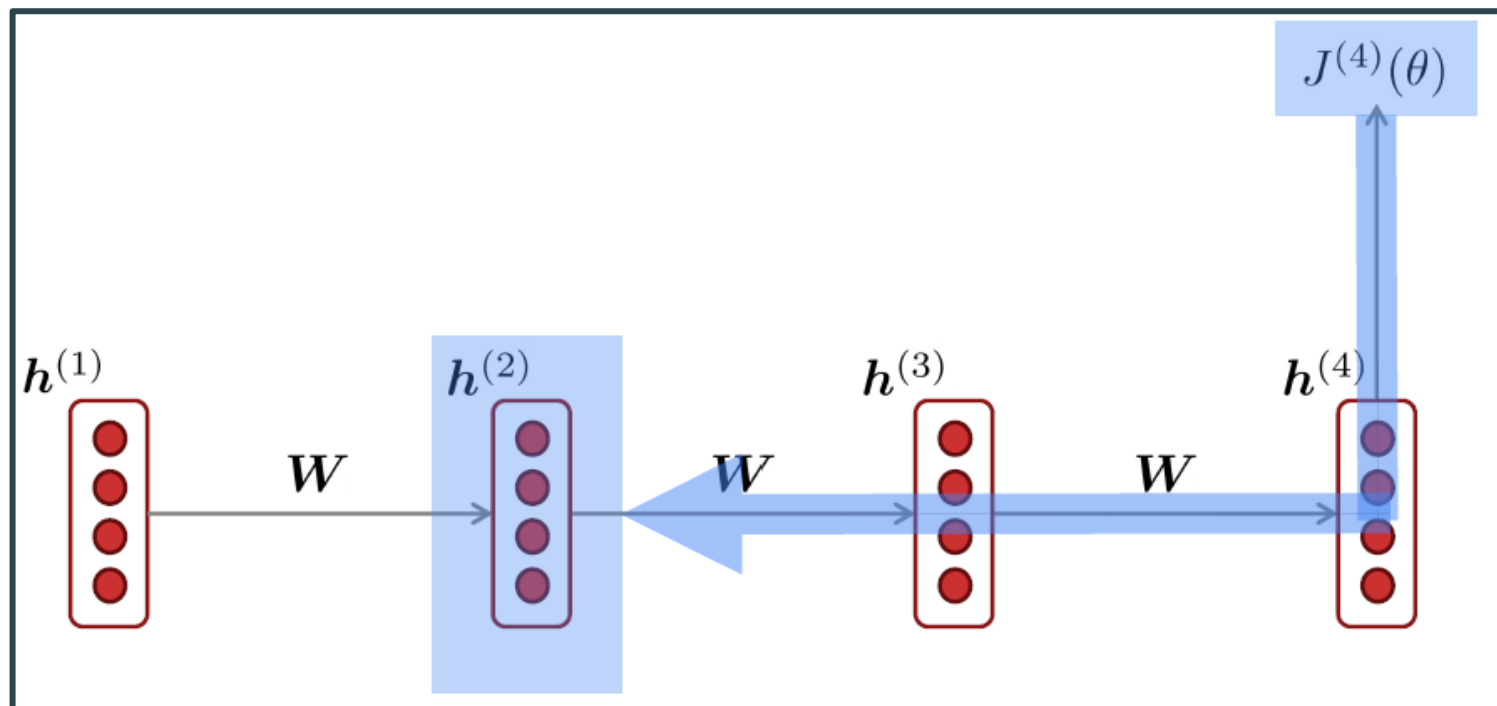
$$\frac{\partial J^{(4)}}{\partial h^{(1)}} = ?$$

قاعده ضرب زنجیره ای



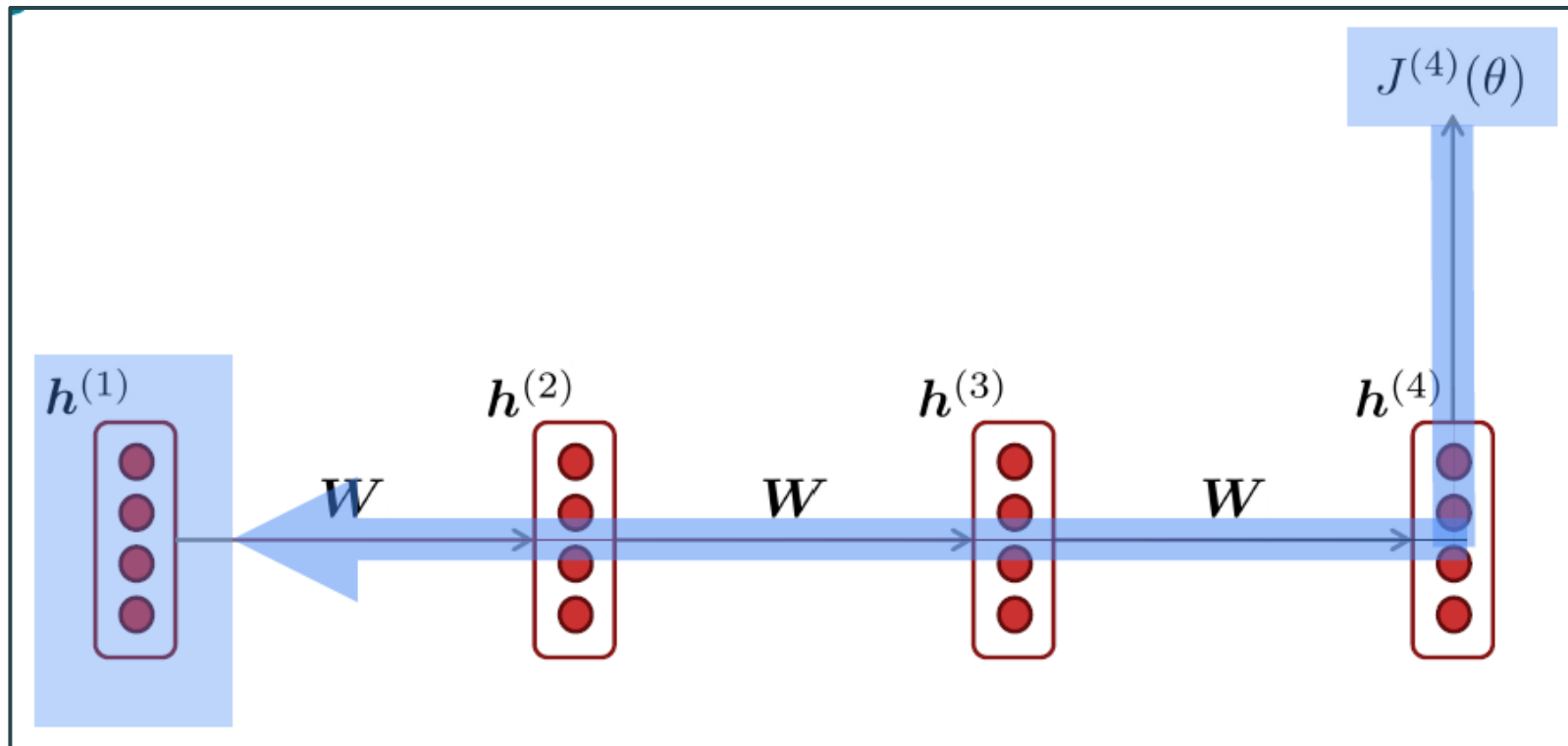
$$\frac{\partial J^{(4)}}{\partial h^{(1)}} = \frac{\partial J^{(4)}}{\partial h^{(4)}} \times \frac{\partial h^{(4)}}{\partial h^{(3)}}$$

مجدد قاعده ضرب زنجیره ای



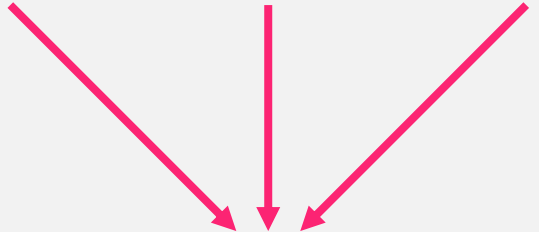
$$\frac{\partial J^{(4)}}{\partial h^{(1)}} = \frac{\partial J^{(4)}}{\partial h^{(4)}} \times \frac{\partial h^{(4)}}{\partial h^{(3)}} \times \frac{\partial h^{(3)}}{\partial h^{(2)}}$$

دوباره ضرب زنجیره ای



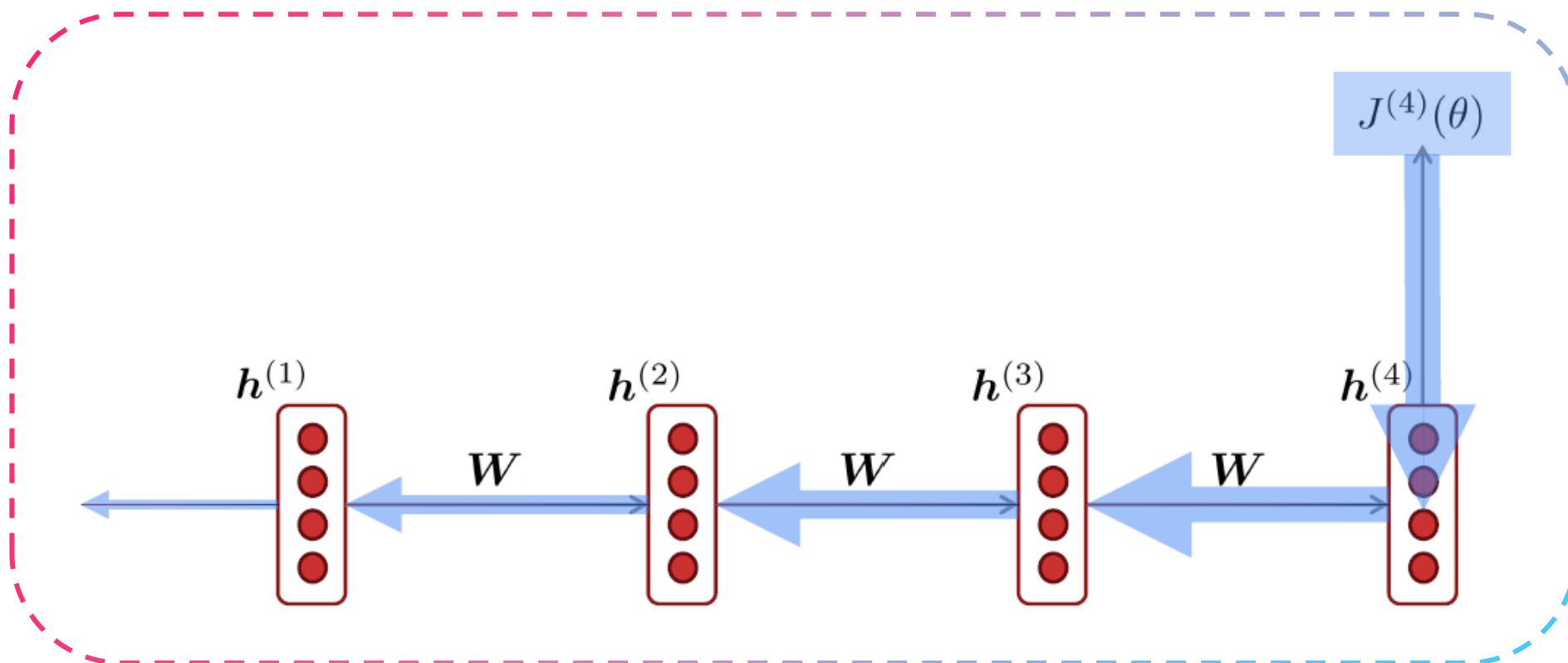
$$\frac{\partial J^{(4)}}{\partial h^{(1)}} = \frac{\partial J^{(4)}}{\partial h^{(4)}} \times \frac{\partial h^{(4)}}{\partial h^{(3)}} \times \frac{\partial h^{(3)}}{\partial h^{(2)}} \times \frac{\partial h^{(2)}}{\partial h^{(1)}}$$

سوال

$$\frac{\partial J^{(4)}}{\partial h^{(1)}} = \frac{\partial J^{(4)}}{\partial h^{(4)}} \times \frac{\partial h^{(4)}}{\partial h^{(3)}} \times \frac{\partial h^{(3)}}{\partial h^{(2)}} \times \frac{\partial h^{(2)}}{\partial h^{(1)}}$$


اگر همه اینها اعداد کوچکی باشند چه می شود ؟

گرایان ها کوچک و کوچکتر می شود و اصطلاحاً محو می شوند !

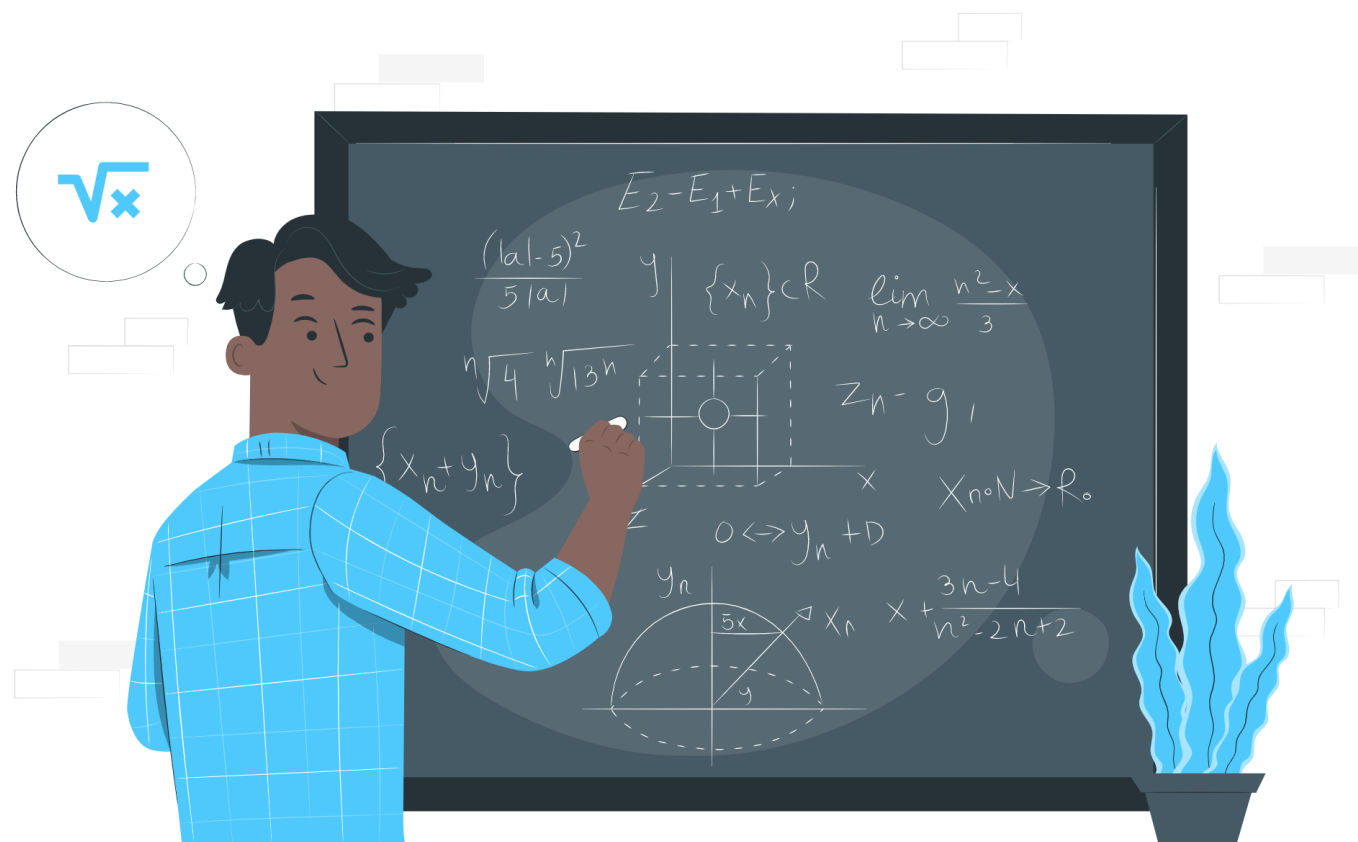


یعنی ما واقعا اینقدر بد شانس هستیم که همه اینها با هم عدد کمی بشن؟



بدشانس نیستیم ولی با ریاضیات نشون میدیم که دقیقا همین اتفاق میفته.

برای این که این رو نشون بدیم ابتدا باید به قضیه ریاضیاتی رو معرفی کنیم.

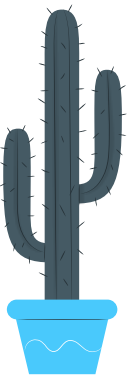
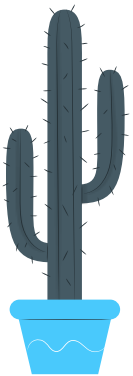


قضیه

اگر $y = Ax$ باشد که در آن $y \in R^{m \times 1}$ و $x \in R^{n \times 1}$ و $A \in R^{m \times n}$ و x تابعی از z باشد و A مستقل از z باشد. آنگاه داریم :



$$y = Ax \rightarrow \frac{\partial y}{\partial z} = A \frac{\partial x}{\partial z}$$



یک فرض σ : یک تابع خطی است (تابع فعال سازی نداریم).

بنابراین



$$h^{(t)} = \sigma(W_h h^{(t-1)} + W_x x^{(t)} + b_1) \Rightarrow h^{(t)} = W_h h^{(t-1)} + W_x x^{(t)} + b_1$$

میخواهیم رابطه زیر را محاسبه کنیم:

$$\frac{\partial h^{(t)}}{\partial h^{(t-1)}}$$



چرا مهمه ؟

$$\frac{\partial J^{(4)}}{\partial h^{(1)}} = \frac{\partial J^{(4)}}{\partial h^{(4)}} \times \frac{\partial h^{(4)}}{\partial h^{(3)}} \times \frac{\partial h^{(3)}}{\partial h^{(2)}} \times \frac{\partial h^{(2)}}{\partial h^{(1)}}$$

برای محاسبه داریم

$$\mathbf{h}^{(t)} = \sigma(\mathbf{W}_h \mathbf{h}^{(t-1)} + \mathbf{W}_x \mathbf{x}^{(t)} + \mathbf{b}_1) \Rightarrow \mathbf{h}^{(t)} = \mathbf{W}_h \mathbf{h}^{(t-1)} + \mathbf{W}_x \mathbf{x}^{(t)} + \mathbf{b}_1$$

$$y = Ax \rightarrow \frac{\partial y}{\partial z} = A \frac{\partial x}{\partial z} \quad \Downarrow$$

$$\frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(t-1)}} = \mathbf{W}_h \frac{\partial \mathbf{h}^{(t-1)}}{\partial \mathbf{h}^{(t-1)}} = \mathbf{W}_h$$

بنابراین برای رابطه گفته شده در اسلایدهای قبل داریم :

$$\frac{\partial J^{(4)}}{\partial h^{(1)}} = \frac{\partial J^{(4)}}{\partial h^{(4)}} \times \frac{\partial h^{(4)}}{\partial h^{(3)}} \times \frac{\partial h^{(3)}}{\partial h^{(2)}} \times \frac{\partial h^{(2)}}{\partial h^{(1)}}$$



$$\frac{\partial J^{(4)}}{\partial h^{(1)}} = \frac{\partial J^{(4)}}{\partial h^{(4)}} \times W_h \times W_h \times W_h$$

و در حالت کلی :

$$\frac{\partial J^{(i)}(\theta)}{\partial \mathbf{h}^{(j)}} = \frac{\partial J^{(i)}(\theta)}{\partial \mathbf{h}^{(i)}} \prod_{j < t \leq i} \frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(t-1)}} = \frac{\partial J^{(i)}(\theta)}{\partial \mathbf{h}^{(i)}} \prod_{j < t \leq i} \mathbf{W}_h = \frac{\partial J^{(i)}(\theta)}{\partial \mathbf{h}^{(i)}} \mathbf{W}_h^\ell, \quad \ell = i - j$$

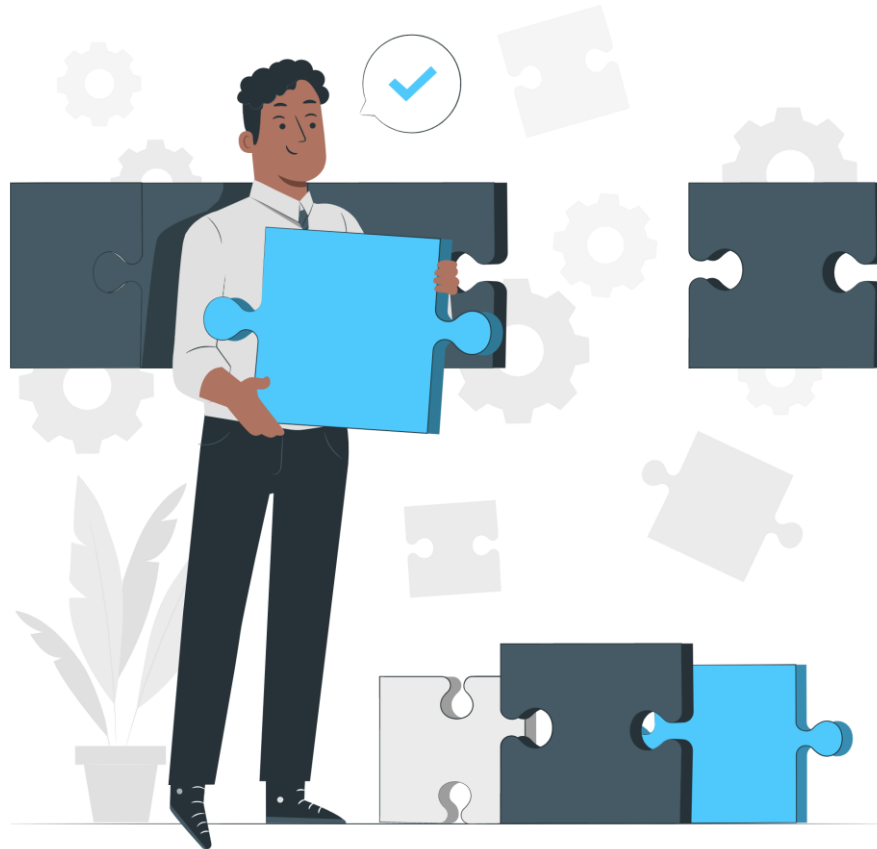


حالا وقتی که ماتریس \mathbf{W} کوچک باشد مقدار مشتق کم و کمتر میشود و اصطلاحاً **vanishing gradient** رخ می دهد.

اصلا سوال: ماتریس کوچک باشه یعنی چی؟

اگر به دنبال این سوال هستید و کمی ریاضیاتی تر میخواهید مساله را ببینید
اسلاید ۲۸ از جلسه ششم دوره CS224n را مشاهده کنید.

حالا چرا این Vanishing gradient در دسر ساز هست ؟



حالا چرا این Vanishing gradient در دسر ساز هست ؟

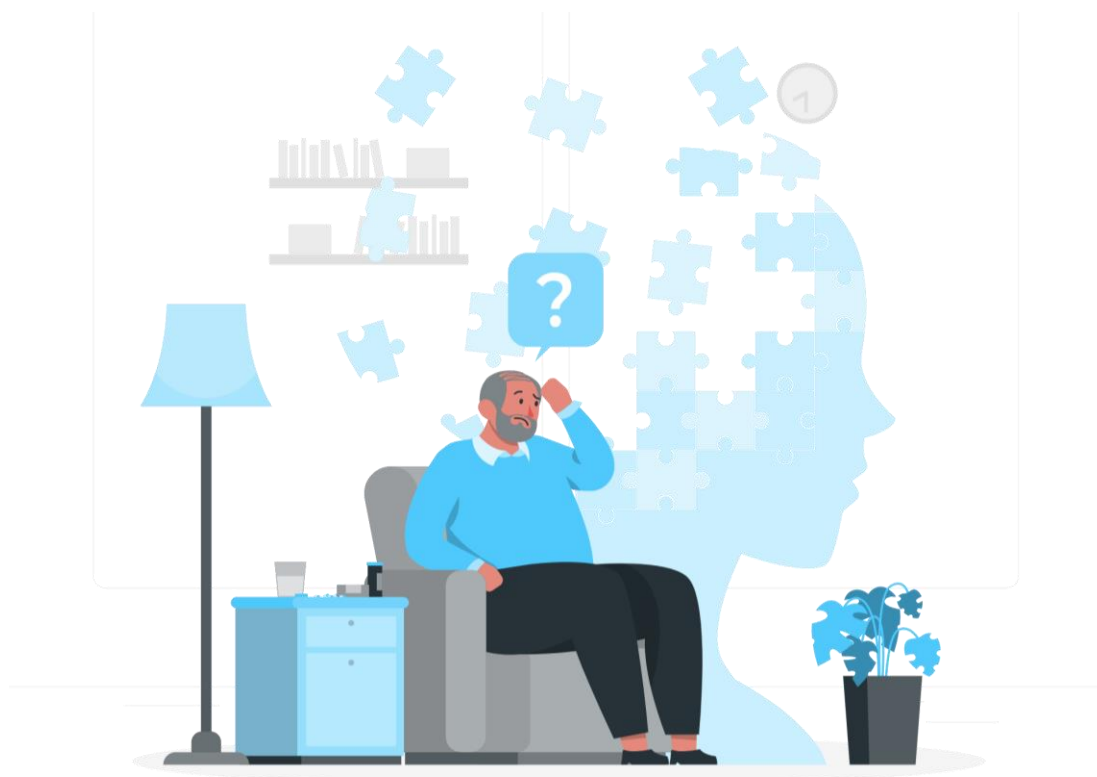
مشتق هایی که مربوط به قدیم هستند (توان w عدد بالایی هست) عملا مقدار کمی پیدا میکنند و نقشی در تعیین وزن ندارند در حالی که مشتق های جدیدتر نقش زیاد تری دارند.



سیگنال های قدیمی نقشی در تعیین وزن ندارند و سیگنال های جدید نقش عمده ای ایفا میکنند.

به عبارت دیگر

سلول RNN ساده Long-Term Dependency یا وابستگی طولانی مدت ندارد.



يعنی چی ؟ یہ مثال

LM task: When she tried to print her tickets, she found that the printer was out of toner.

She went to the stationery store to buy more toner. It was very overpriced. After

installing the toner into the printer, she finally printed her _____

توضیح

برای پر کردن جای خالی نیاز است تا مدل کلمه tickets در هفتمین جایگاه را در نظر داشته باشد.



اما اگر گرادیان ها کوچک باشند و ارتباطات long-distance نتواند برقرار کند ، نمیتواند چنین پیشبینی را انجام دهد.

زمان توقف : سوالی بحثی ...



آنچه در این جلسه گفته خواهد شد :



مشکل دیگر Exploding Gradient

$$\theta^{new} = \theta^{old} - \alpha \nabla_{\theta} J(\theta)$$

وقتی که ایشون خیلی بزرگ بشه



$$\frac{\partial J^{(i)}(\theta)}{\partial \mathbf{h}^{(j)}} = \frac{\partial J^{(i)}(\theta)}{\partial \mathbf{h}^{(i)}} \prod_{j < t \leq i} \frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(t-1)}} = \frac{\partial J^{(i)}(\theta)}{\partial \mathbf{h}^{(i)}} \prod_{j < t \leq i} \mathbf{w}_h = \frac{\partial J^{(i)}(\theta)}{\partial \mathbf{h}^{(i)}} \mathbf{w}_h^\ell, \quad \ell = i - j$$



حالا وقتی که ماتریس \mathbf{w} خیلی بزرگ شود چی ؟

Gradient Clipping : راه حل


Algorithm 1 Pseudo-code for norm clipping

$\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta}$
if $\|\hat{\mathbf{g}}\| \geq threshold$ **then**
 $\hat{\mathbf{g}} \leftarrow \frac{threshold}{\|\hat{\mathbf{g}}\|} \hat{\mathbf{g}}$
end if



نکته :

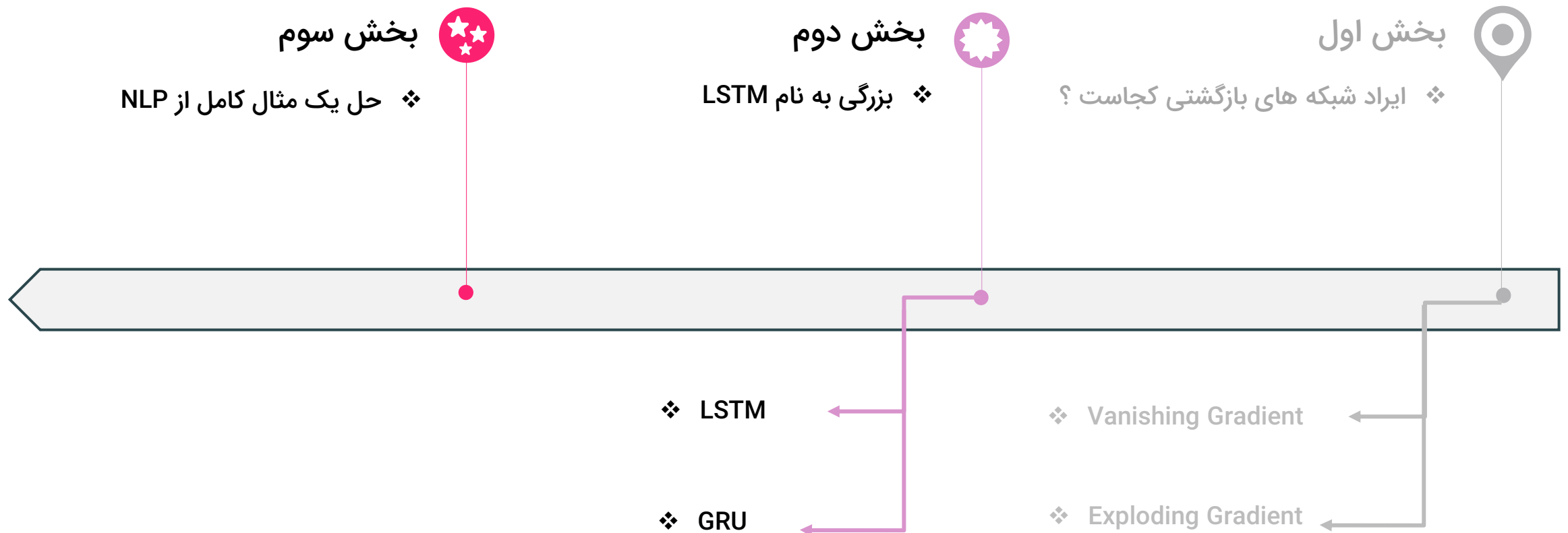
به صورت کلی مساله **Exploding Gradient** ساده تر از **Vanishing Gradient** است و راحت تر میتوان آن را حل کرد.



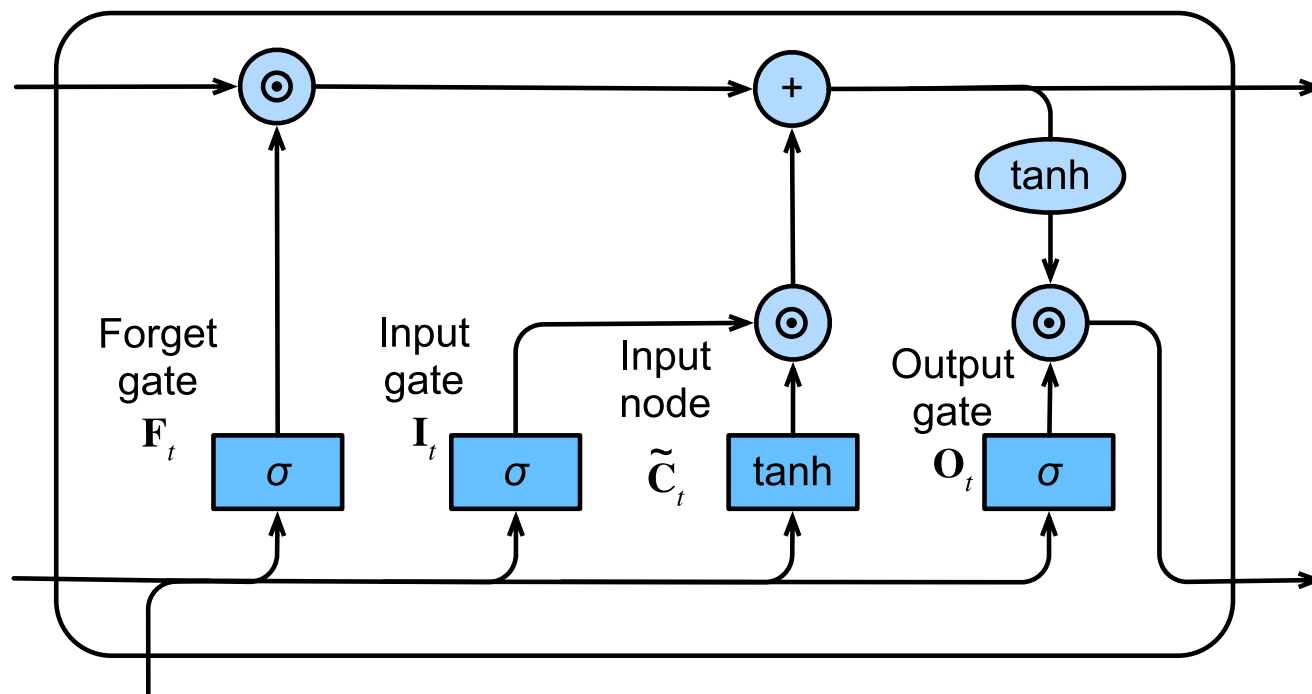
حالا Vanishing Gradient رو چطور حل کنیم؟



آنچه در این جلسه گفته خواهد شد :



Vanishing Gradient راه حلی برای LSTM



مقاله اصلی LSTM

long short-term memory-1997

Sepp hochreiter, jurgen schmidhuber



مقاله ای که کمی بهش لطفی شده

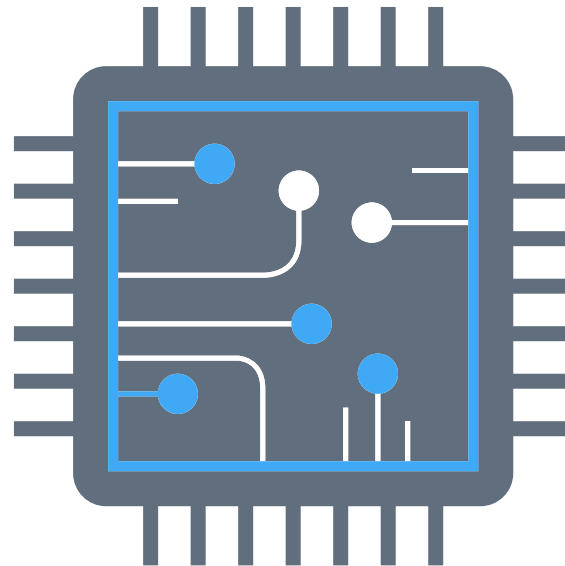
Learning to Forget: Continual Prediction with LSTM-2000

Felix A. Gers Jurgen Schmidhuber Fred Cummins

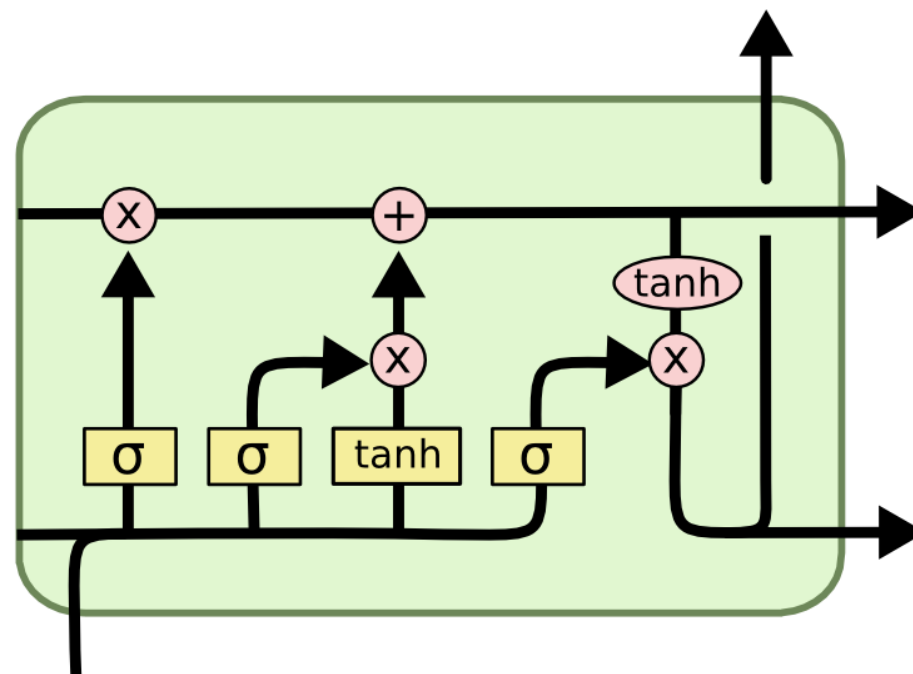


حالت ایده آل ما چیه

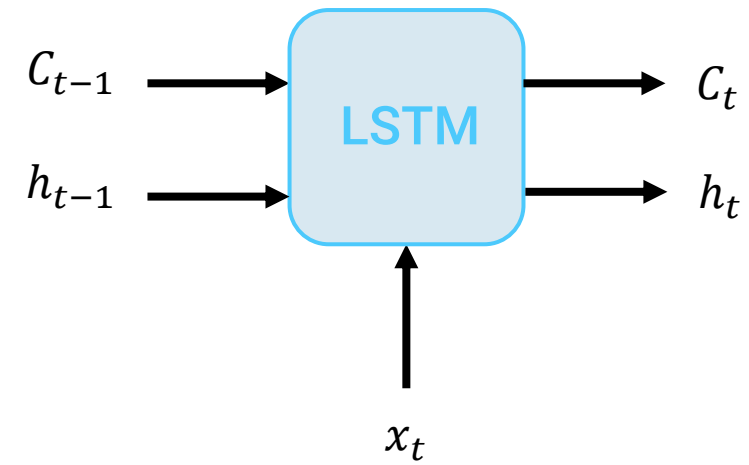
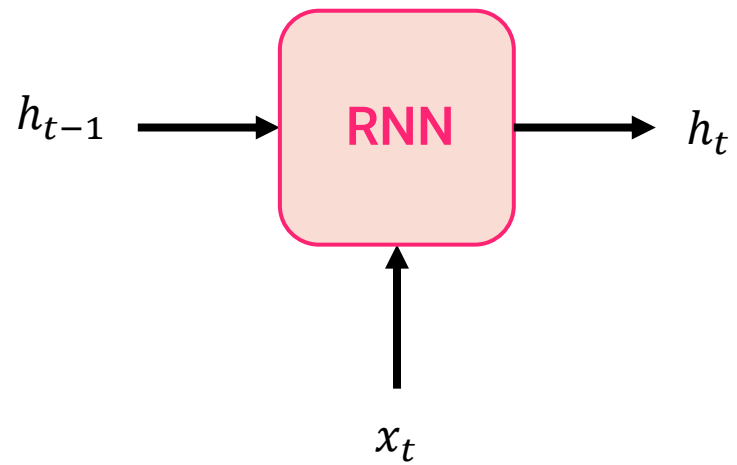
یک حافظه داشته باشیم که اطلاعات مهم و کلیدی را در آن ذخیره کنیم. فرقی ندارد از گذشته باشد یا حال.



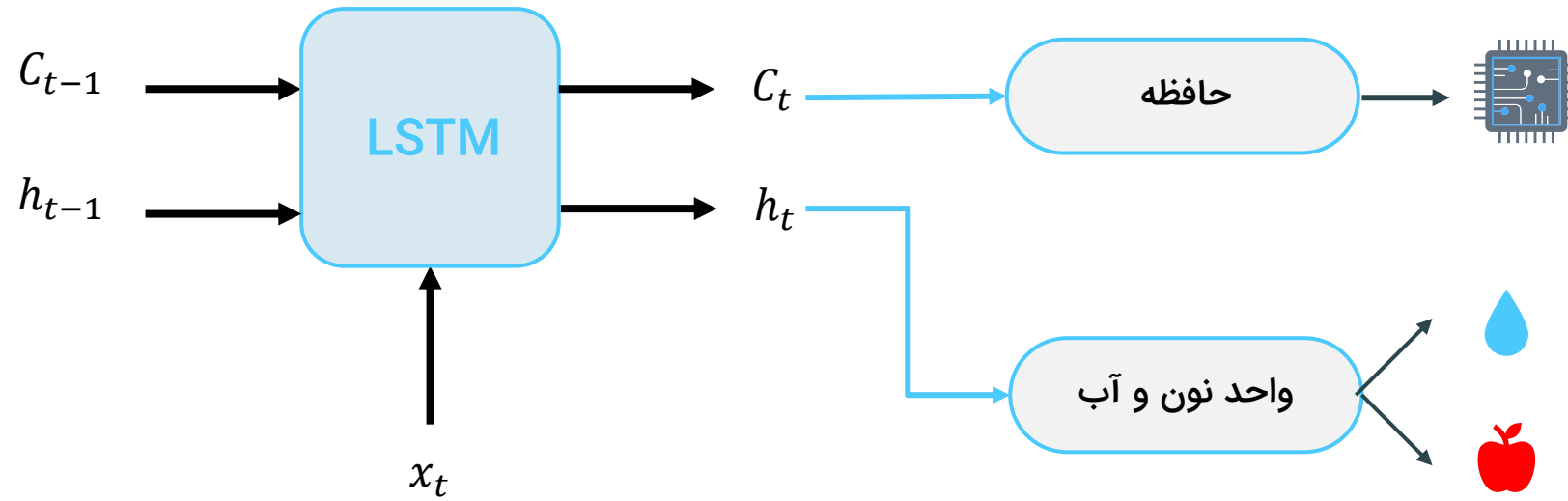
LSTM ، شخصیتی دارای حافظه



از نگاهی کلی



یه ذره دقیق تر



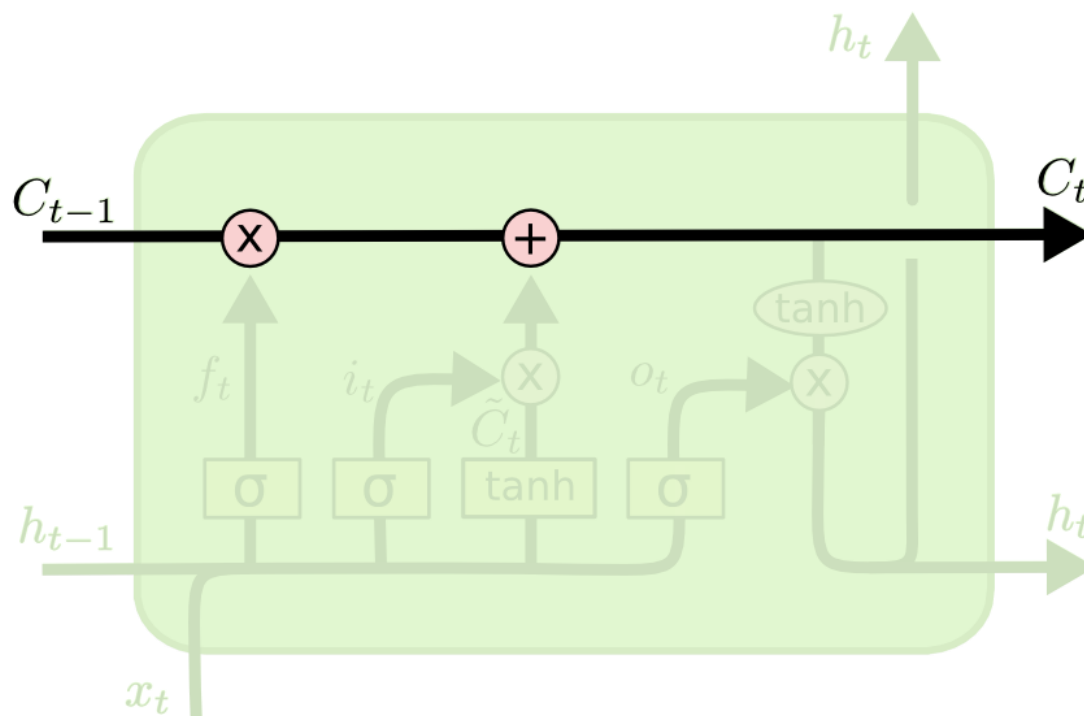
این حافظه قراره دقیقا چیکار کنه ؟

قراره از اول Sequence تا آخر Sequence اطلاعات مفید رو توی خودش ذخیره کنه. دیگه نگران حذف اطلاعات مفید قدیمی نباشیم.

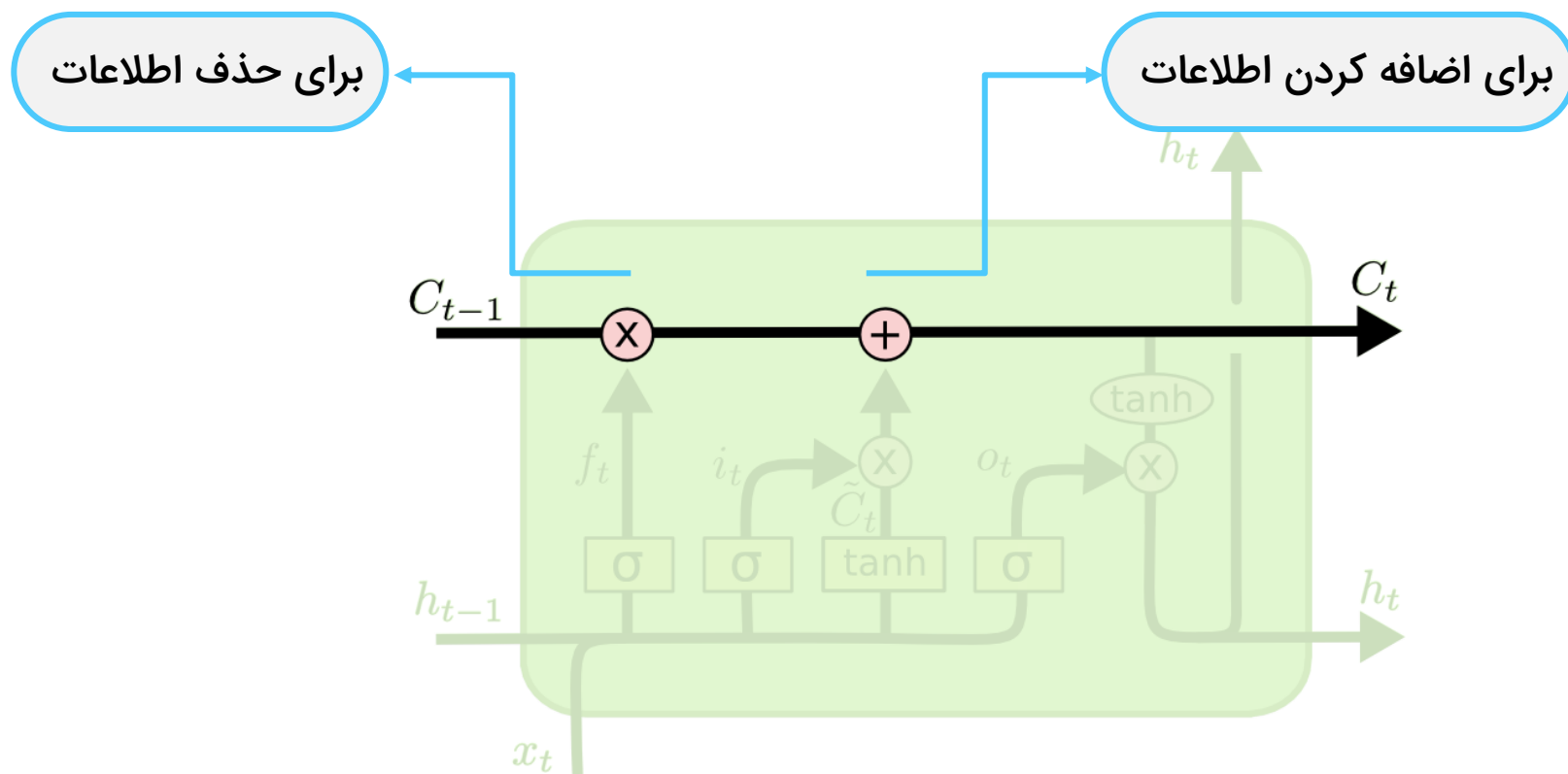


پس این حافظه باید دو تا کار مهم انجام بده: اول حذف اطلاعات و دوم اضافه کردن اطلاعات

و دقیقا همین دو عملیات رو در دل خودش داره. با اون **علامت ضرب** حذف میکنه و با اون **جمع** هم اضافه میکنه اطلاعات رو.



به عبارت دیگر



اول بینیم چطور حذف میکنه



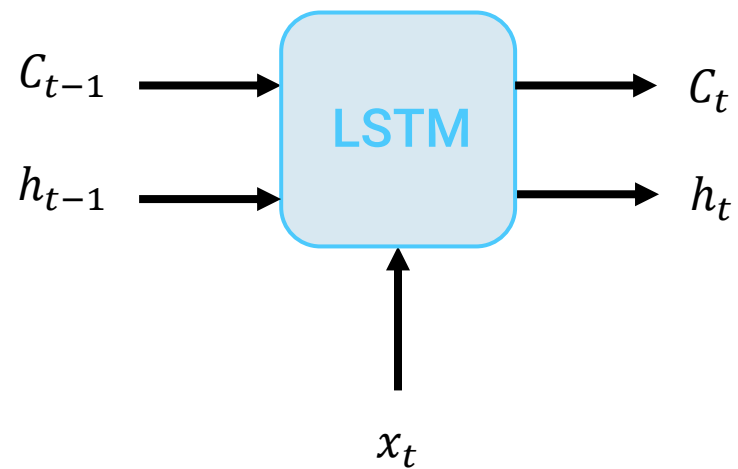
فقط باید بفهمیم این بردار f چطور تولید میشه ؟

f_t

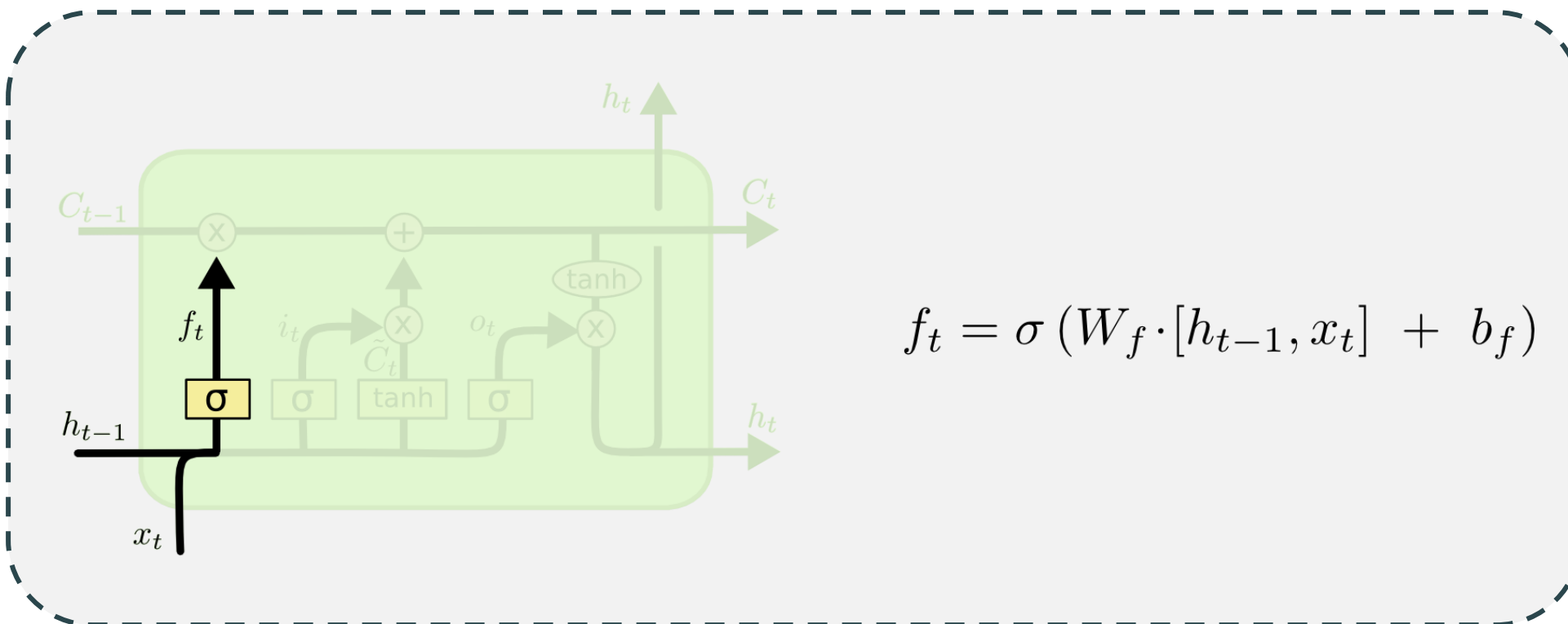


ایده شما چیست ؟

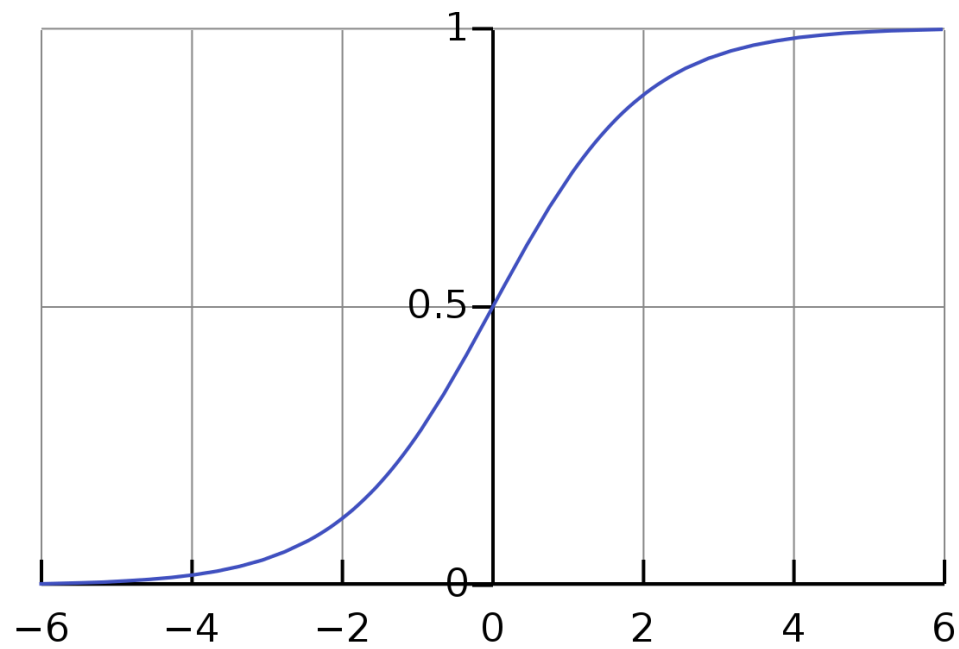
ما چه ابزارهایی در اختیار داریم برای ساخت f ؟ از همون ها استفاده کنیم !



گیت فراموشی ، ابزار LSTM برای حذف اطلاعات قدیمی از حافظه (Cell State)



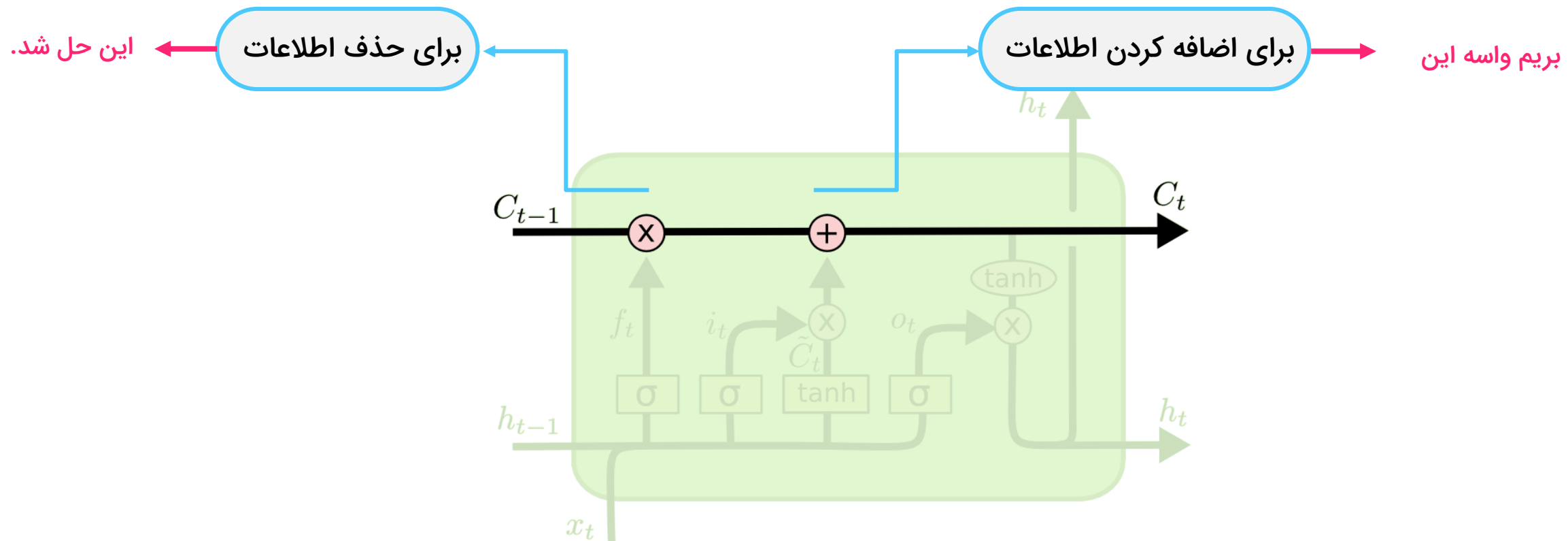
سیگموید چون خروجی بین صفر تا یک دارد میتواند، میتواند در نقش یک گیت عمل کند.



به بیان دیگر

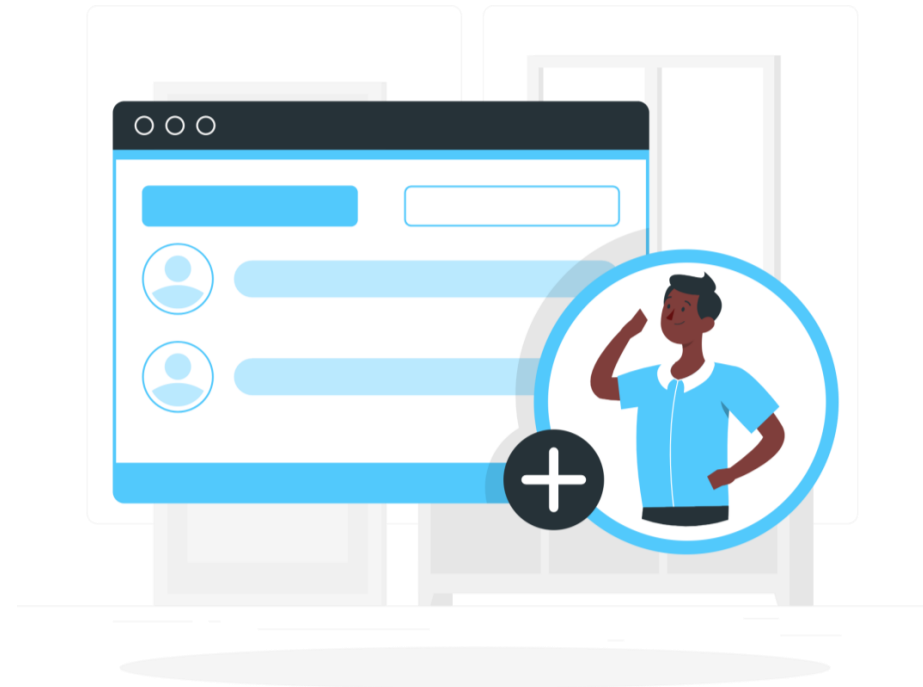


تا اینجا



اما چطور اطلاعات اضافه کنیم به Cell State ؟

باید به دو سوال مهم پاسخ بدیم: **چپو** میخوایم اضافه کنیم و **چقدر از اون رو** میخوایم اضافه کنیم ؟

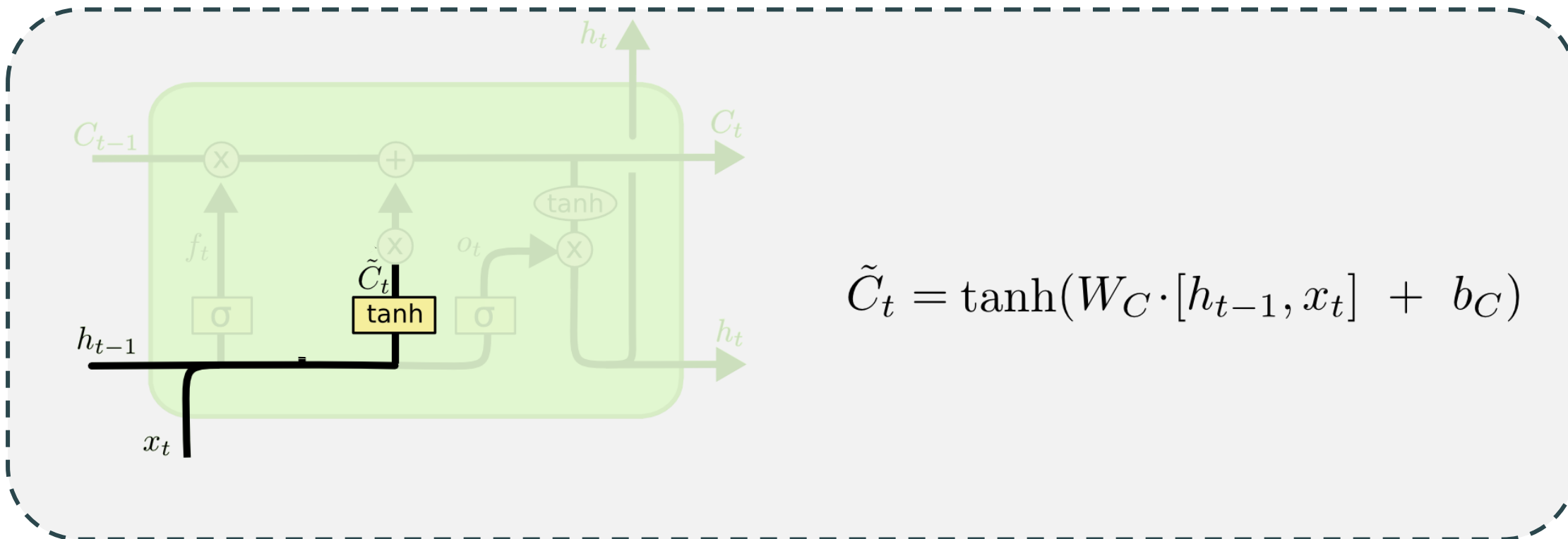


چیو میخوایم اضافه کنیم و چقدر از اون چیز رو ؟

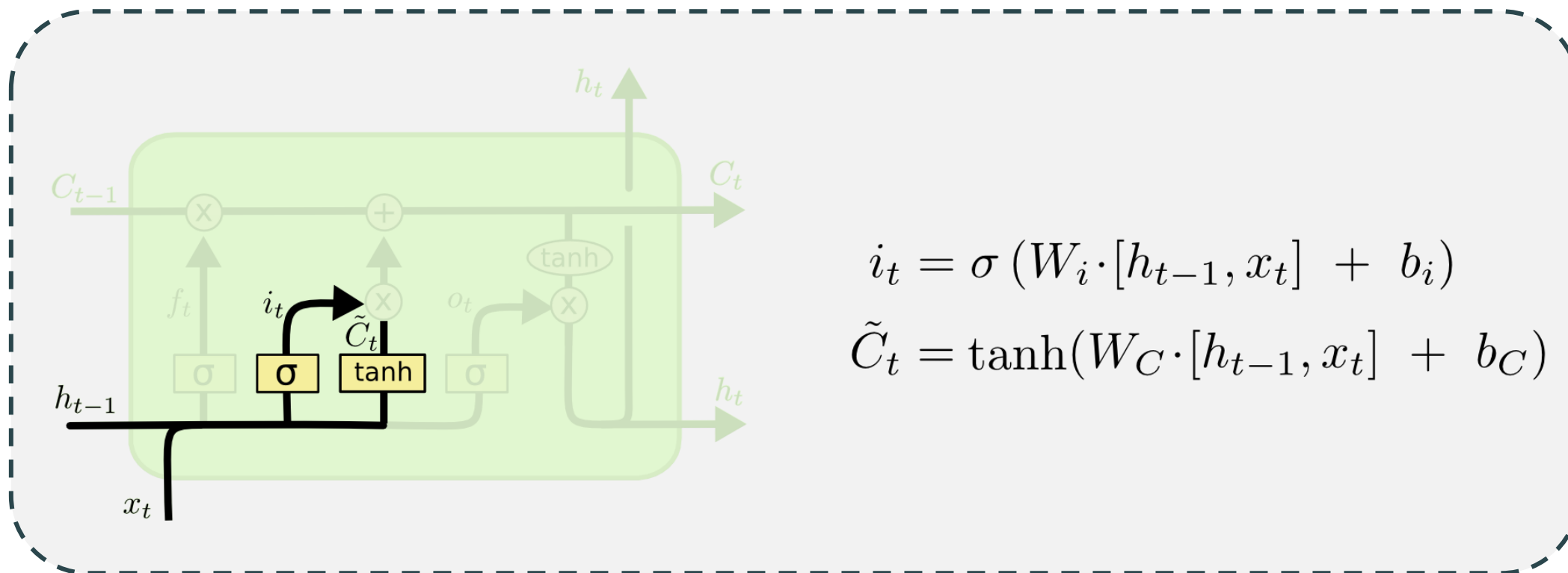


دوباره بیاییم از همون ابزارهای خودمون استفاده کنیم ! xt و $ht-1$

چيو ميخوايم اضافه كنيم؟



چقدر از اون رو میخوایم اضافه کنیم؟



همه چیز رو نمیخواد اضافه کنی به حافظه. شاید بی ارزش باشه



LM task: When she tried to print her tickets, she found that the printer was out of toner.

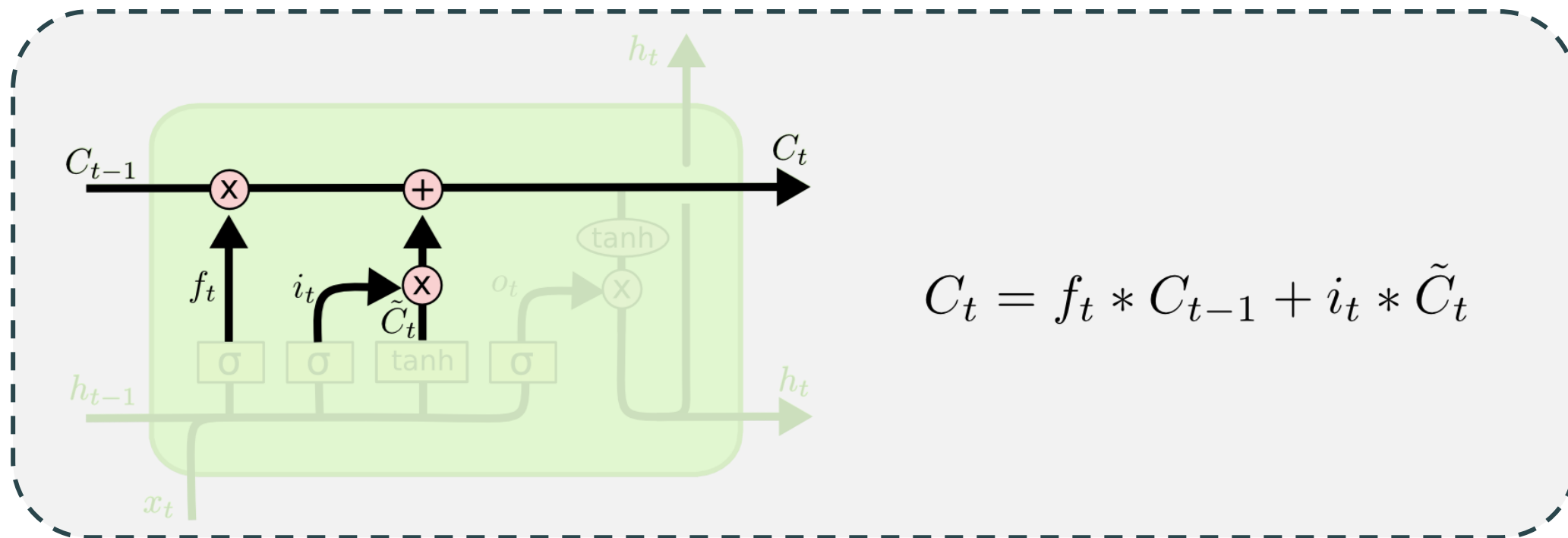
She went to the stationery store to buy more toner. It was very overpriced. After

installing the toner into **the** printer, she finally printed her _____

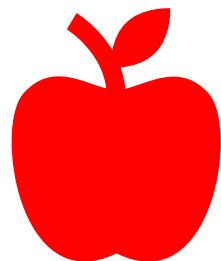
الکی چرا حافظه رو مشغول کنیم؟



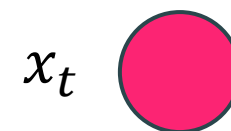
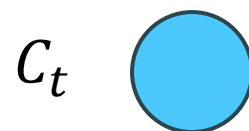
حالا بیاییم و رابطه کلی Cell State رو بنویسیم



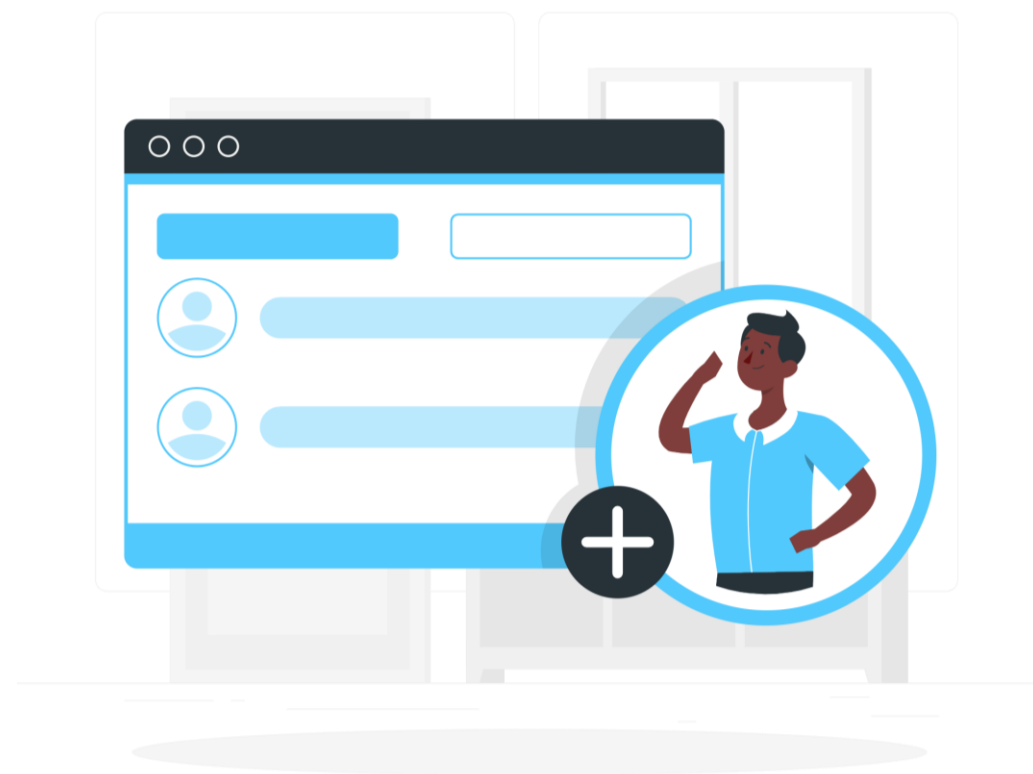
همه چیز محاسبه شد الا یک چیز؟ چه چیزی؟



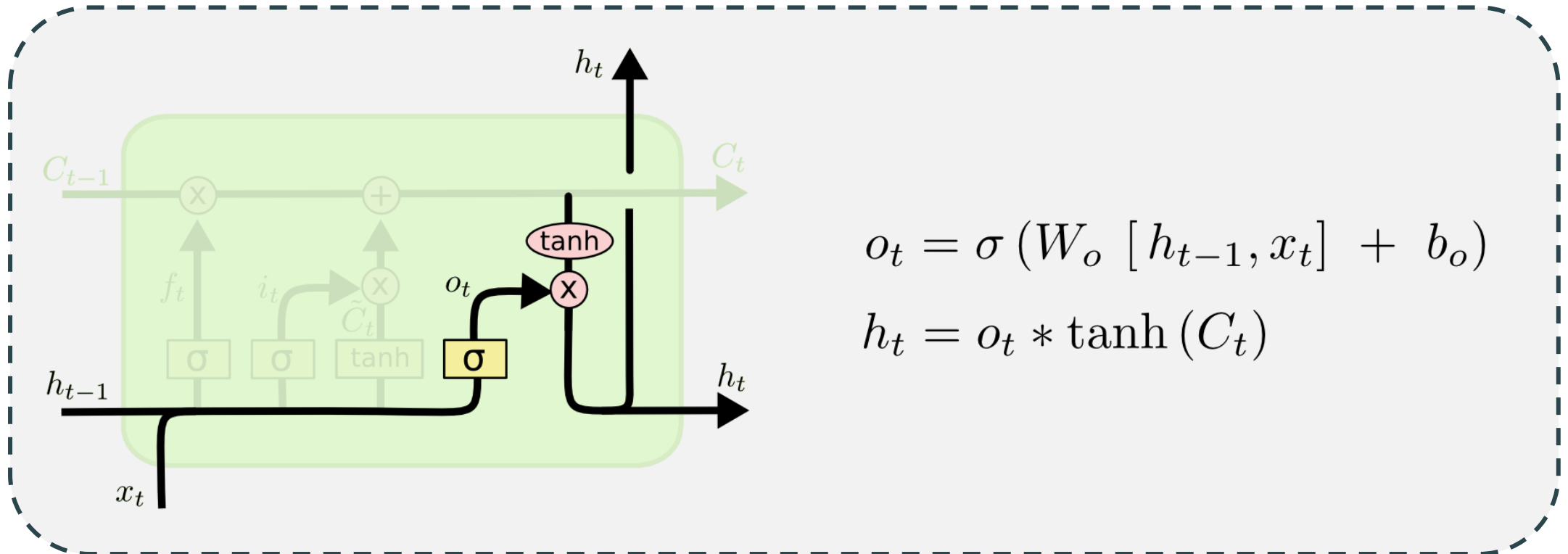
اوکی. حالا به نظر شما بین سه شخصیت زیر کدام یک شایسته ترین شخصیت
جهت تولید خروجی (h_t) است.



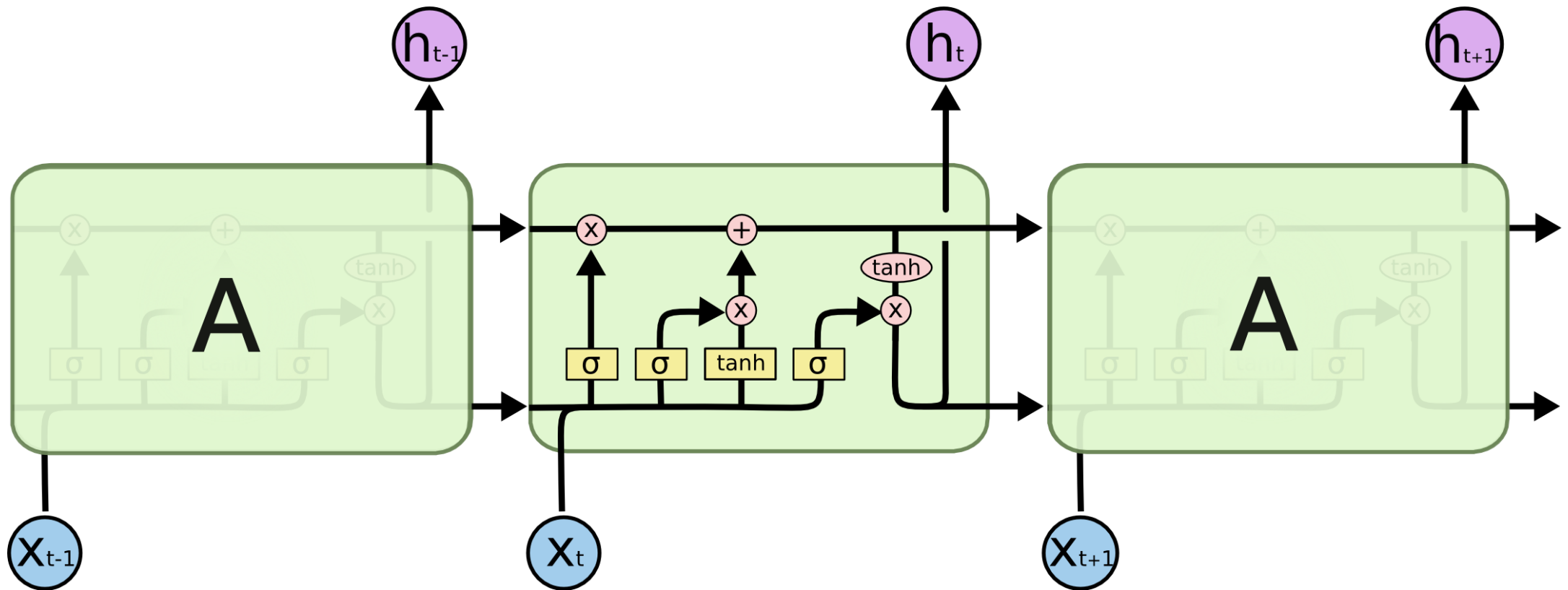
و سوال این هست : چقدر از Ct به خروجی برود ؟ این چقدر از Ct رو با چطور بسازم ؟



دوباره گیت ها ، ابزار کلیدی ما



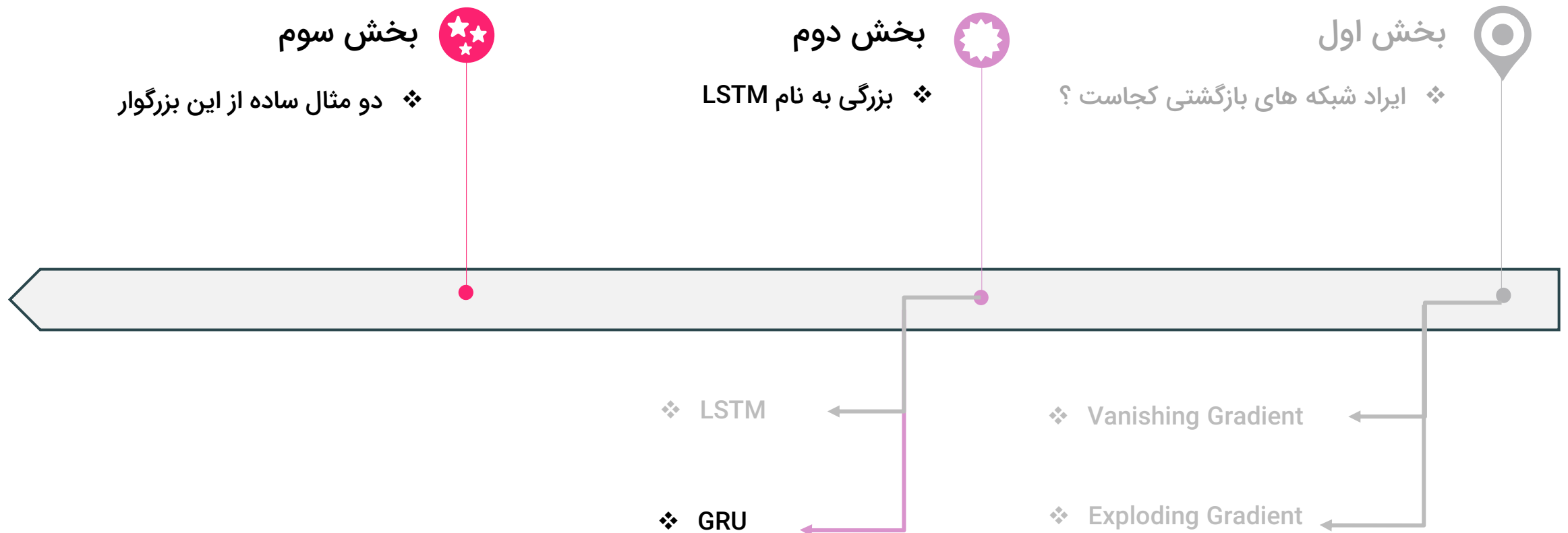
این بود ماجرای بزرگی به نام LSTM



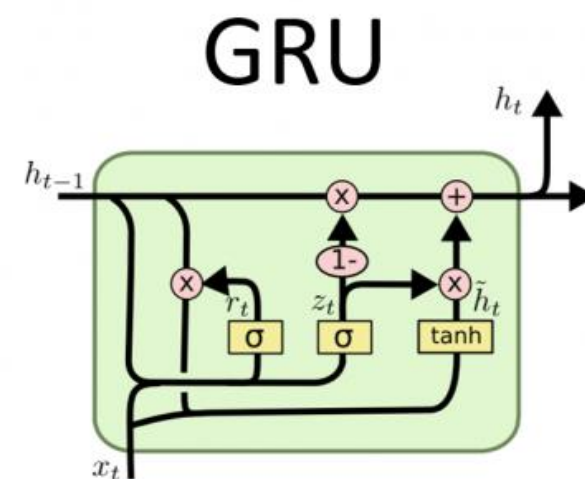
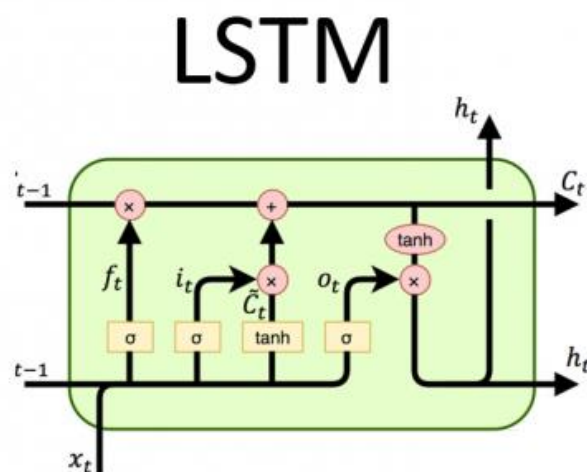
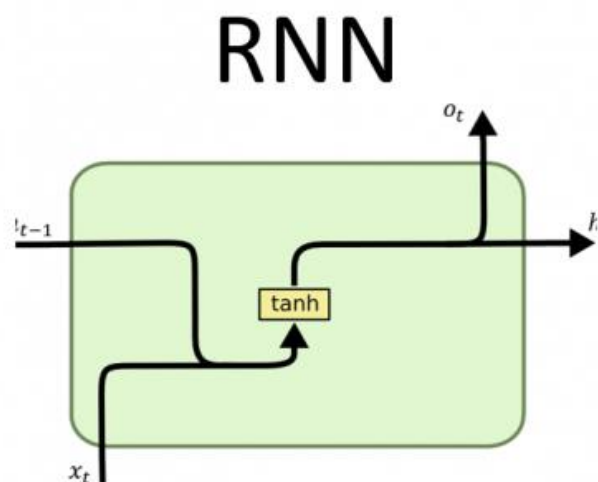
هر سوالی دارید بچه ها الان وقتش هست ... تاکید میکنم. هر سوالی



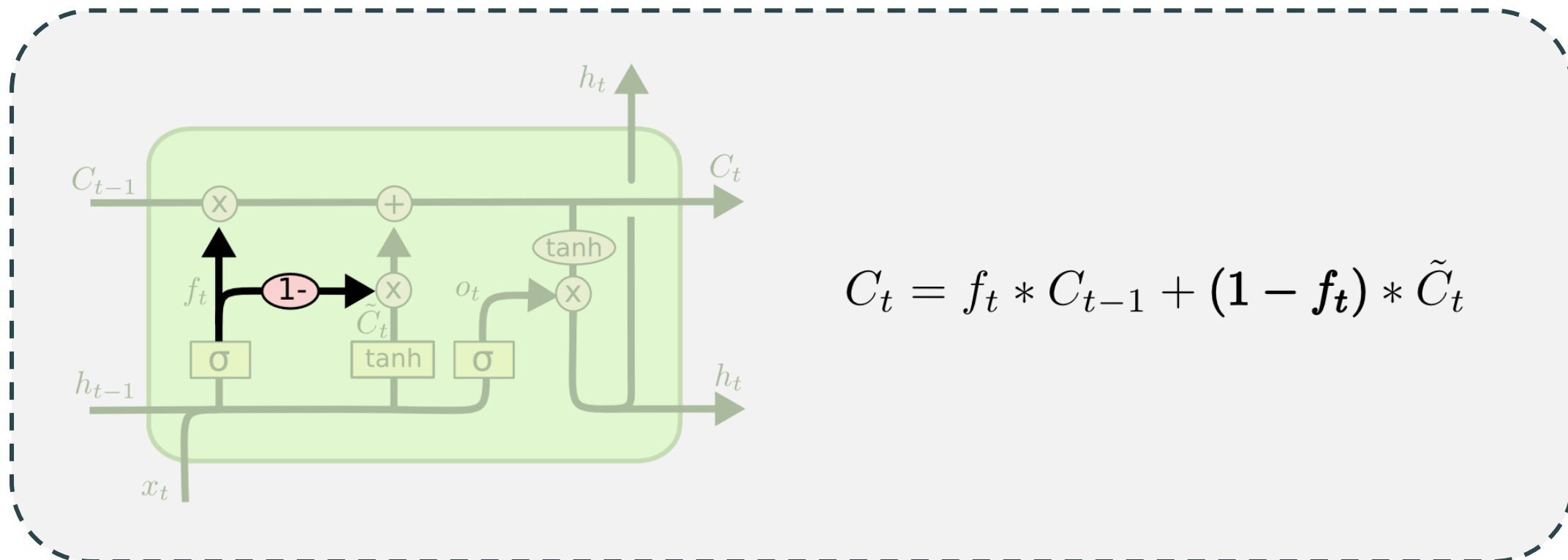
آنچه در این جلسه گفته خواهد شد :



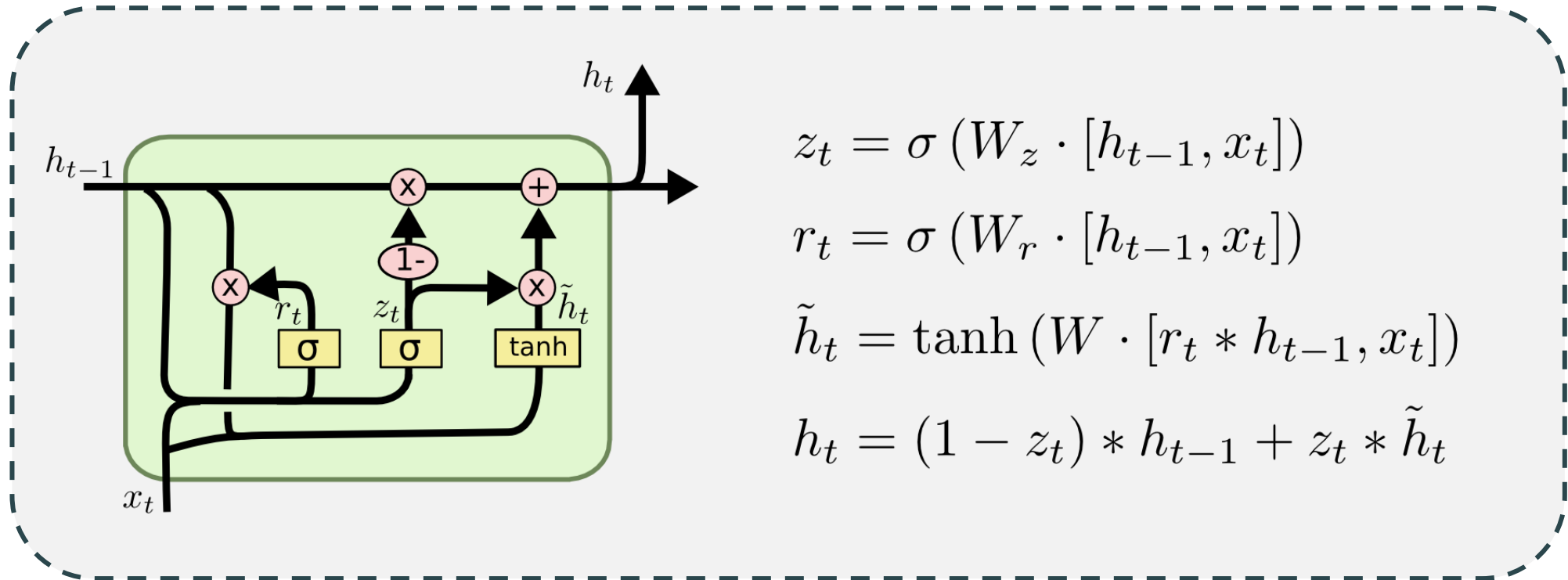
اما این پایان کار نبود و نسخه های مختلفی از LSTM ارایه شد...



ترکیب گیت های Input و forgot



GRU



قاعده سر انگشتی

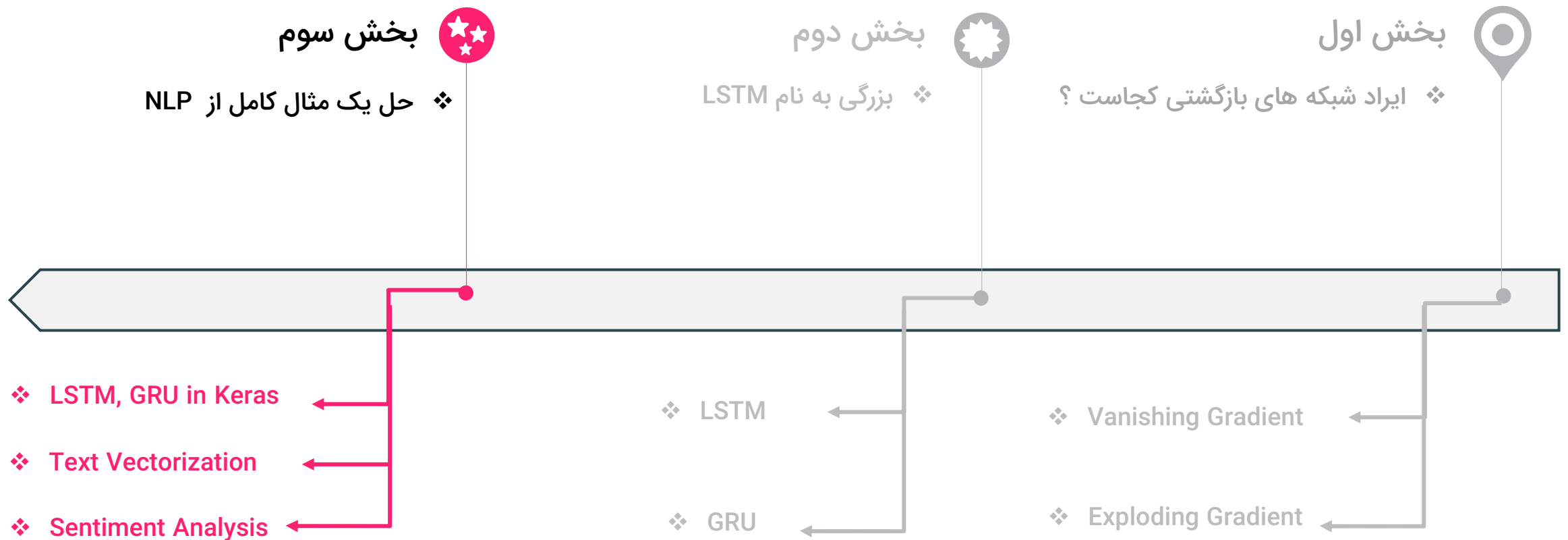
LSTM گزینه بسیار مناسبی است. (به خصوص وقتی **Long Sequence** دارید یا داده **Training** بالایی دارید.) اما اگر سرعت بیشتر و پارامترهای کمتری نیاز دارید گزینه بعدی **GRU** است.

تذکر

LSTM به ما گارانتی نمیده که مشکل **Vanishing Gradient** به صورت کامل و همیشگی حل میکنه.
فقط یک راه حل هست که این مشکل رو خیلی کمتر میکنه.



آنچه در این جلسه گفته خواهد شد :



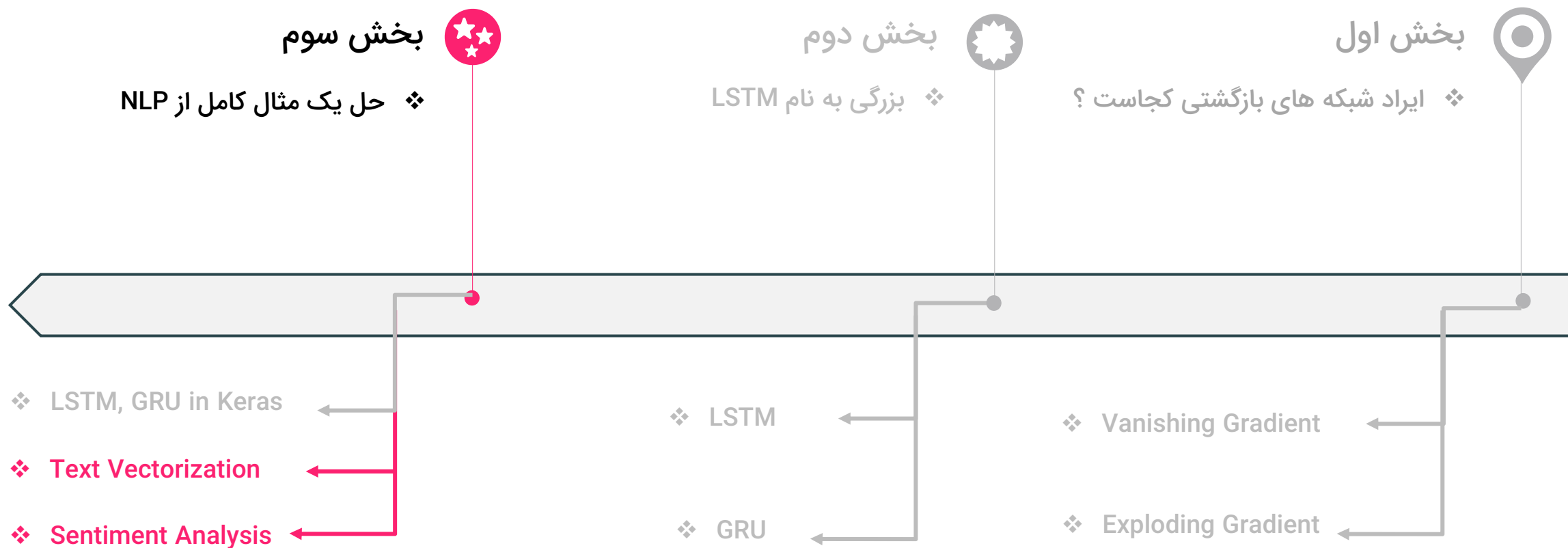
لایه LSTM در keras

```
>>> inputs = tf.random.normal([32, 10, 8])  
  
>>> lstm = tf.keras.layers.LSTM(4)  
  
>>> output = lstm(inputs)  
  
>>> print(output.shape)  
  
(32, 4)
```

خروجی cell state

```
>>> lstm = tf.keras.layers.LSTM(4, return_sequences=True, return_state=True)
>>> whole_seq_output, final_memory_state, final_carry_state = lstm(inputs)
>>> print(whole_seq_output.shape)
(32, 10, 4)
>>> print(final_memory_state.shape)
(32, 4)
>>> print(final_carry_state.shape)
(32, 4)
```

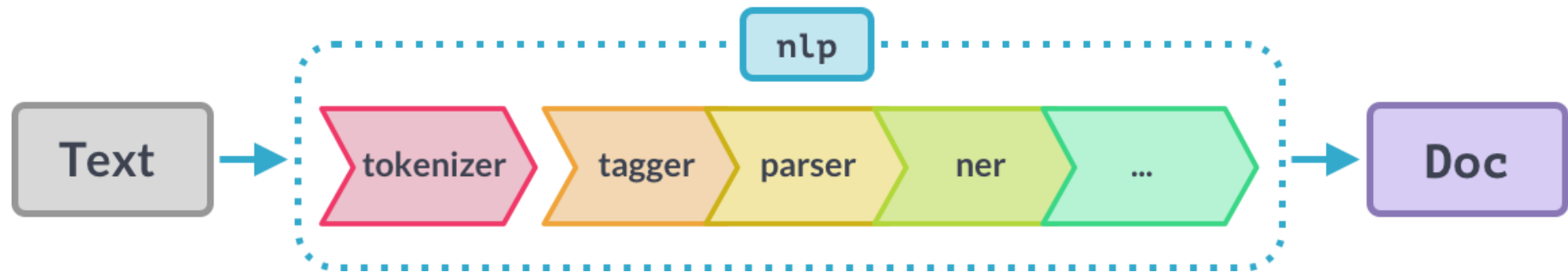
آنچه در این جلسه گفته خواهد شد :



مثال پردازش زبان طبیعی: Sentiment Analysis



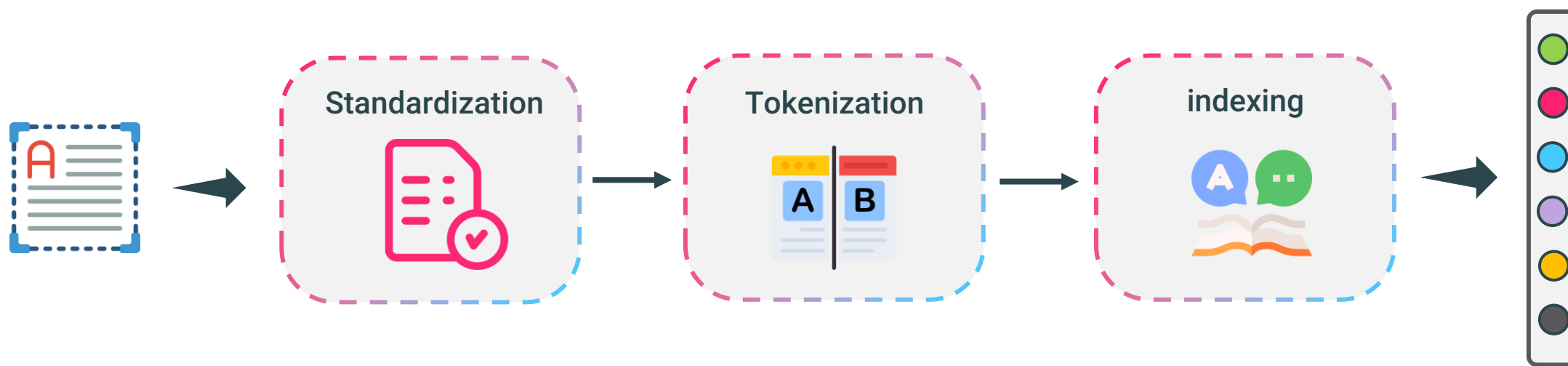
پیش نیاز : کار با داده های متنی



شبکه های عصبی عدد میفهمند. پس باید متن ها را به اعداد تبدیل کنیم.



Text Vectorizing چه مراحل دارد ؟



Standardization

“sunset came. i was staring at the Mexico sky. Isn't nature splendid??”

“Sunset came; I stared at the México sky. Isn't nature splendid?”



باید کاری کنیم که این دو تا جمله که یکی هستند واقعا ، به یک شکل در بیان !

حالا چطور ؟ دو تا کار ساده میتونیم انجام بدیم :

Convert to lowercase



Remove Punctuation characters



پس از انجام این کار

“sunset came i was staring at the mexico sky isnt nature splendid”

“sunset came i stared at the méxico sky isnt nature splendid”

خیلی بهتر شد ولی هنوز نه دقیق

“sunset came i was staring at the mexico sky isnt nature splendid”

“sunset came i stared at the méxico sky isnt nature splendid”

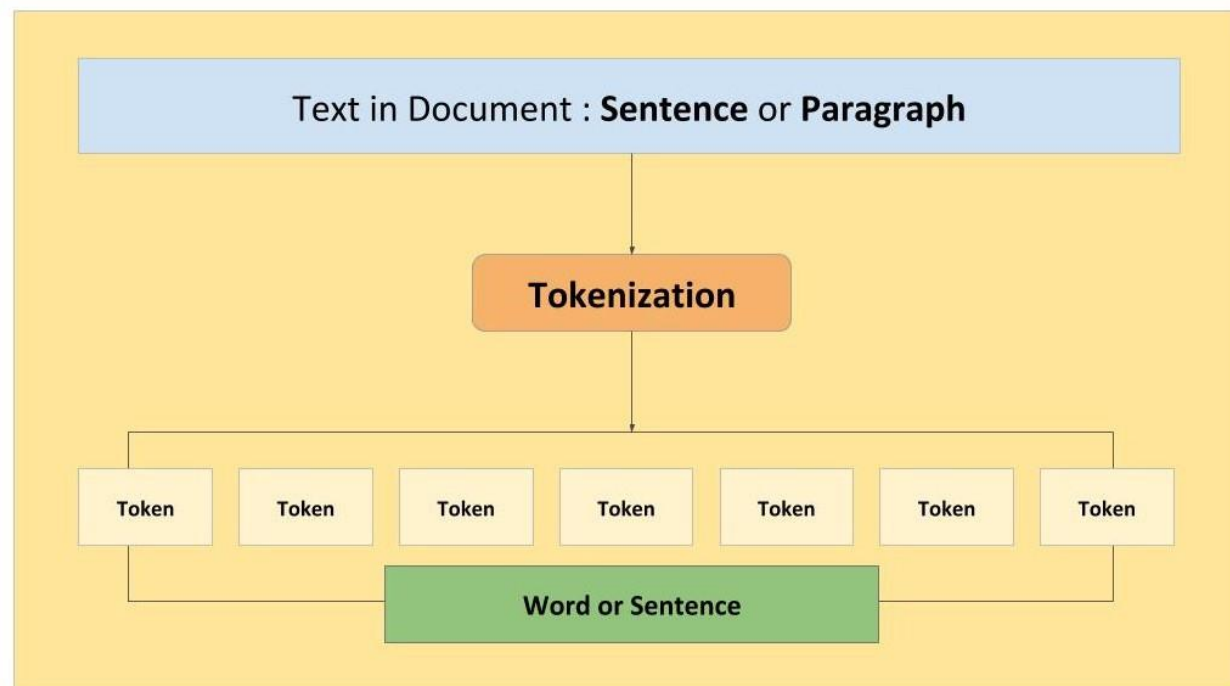
کار دیگه ای که میشه انجام داد

تبدیل کاراکترهای خاص به کاراکترهای استاندارد



“é” with “e,” “æ” with “ae,”

Text Splitting (Tokenization)



تقسیم بندی متن به بخش هایی کوچکتر که به هر بخش یک token میگوییم.

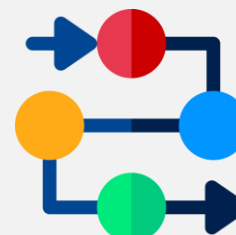
انواع روش ها :

Word Level Tokenization

H	C	W	B
A	K	O	I
L	D	R	O
U	P	D	M

تقسیم بندی جملات به
کلمات که با space یا ...
جدا شده اند.

N-gram Tokenization



هر توکن تشکیل شده از
گروهی از کلمات متوالی

N-gram Tokenization

This is Big Data AI Book

Uni-Gram

This	Is	Big	Data	AI	Book
------	----	-----	------	----	------

Bi-Gram

This is	Is Big	Big Data	Data AI	AI Book
---------	--------	----------	---------	---------

Tri-Gram

This is Big	Is Big Data	Big Data AI	Data AI Book
-------------	-------------	-------------	--------------

دو نوع نگاه در پردازش متن

Sequence model



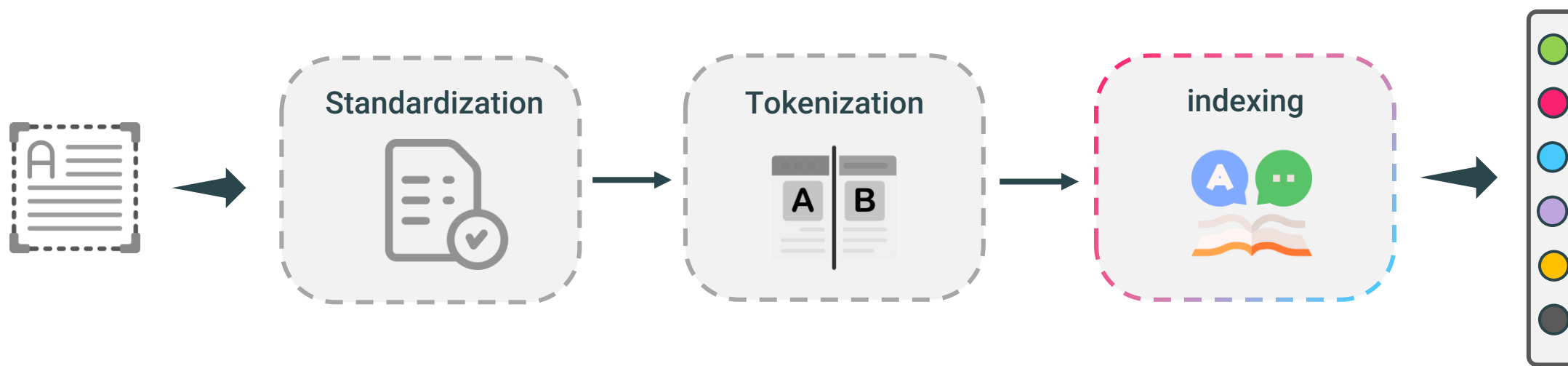
Word level Tokenization

Bag of words model



N-gram Tokenization

Text Vectorizing چه مراحل دارد ؟



Vocabulary Indexing

تبدیل هر کدام از کلمات داده های Train به یک عدد `int` منحصر به فرد



میتونیم هم محدود کنیم که پرتکرارترین کلمات فقط در `indexing` باشند.

سوال: اگه يه کلمه توی دیتای Train بود و توی دیتای تست نبود چی ؟ خطا میگیریم که !
یا مثلا خیلی نادر بود و جزو حذف شده ها بود. اونجا چی ؟

Index یک دیکشنری رو به این کلمات اختصاص میدیم. هر کلمه ای که نیست توی دیکشنری.

(out of vocabulary index -> OOV Index)

چرا Index صفر رو بهش اختصاص ندادیم ؟

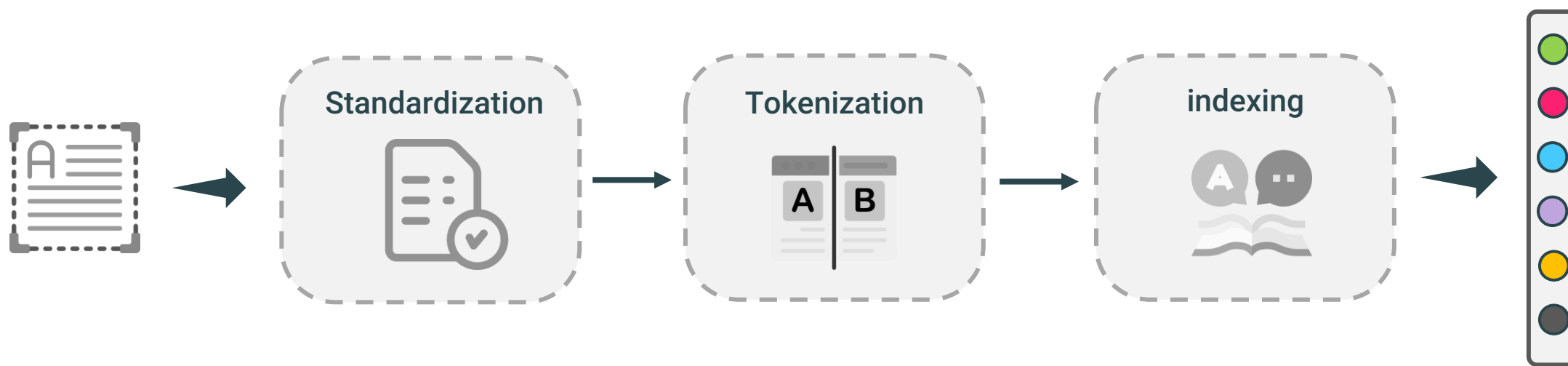


صفر رو دادیم به عزیز دیگری. یادتون هست چی؟

صفرهایی که در pad_sequence میدادیم.

```
[[5, 7, 124, 4, 89]  
 [8, 34, 21, 0, 0]]
```

Text Vectorizing چه مراحل دارد ؟



لايه TextVectorization ، جمع همه خوبان يکجا

```
from keras.layers import TextVectorization  
text_vectorization = TextVectorization(output_mode="int")
```

تنظیمات پیش فرض TextVectorization

Convert to Lowercase and remove punctuation

Split on whitespace



اما کاملاً customize میتوان کرد و توابع مدنظر خود را برای standardization و tokenization نوشت.

استفاده از متد adapt

```
dataset = ["I write, erase, rewrite", "Erase again, and then", "A poppy blooms"]  
text_vectorization.adapt(dataset)
```

نکته : اگه بخوايم از فضای int به کلمات برگرديم ميتونيم از متد `get_vocabulary()` استفاده کنيم.
درايه های vocabulary برحسب تعداد مرتب شده اند و به همين دليل متداول است که `the` و `a` ابتدا بيابند.



```
print(text_vectorization.get_vocabulary())
```

```
['', '[UNK]', 'erase', 'write', 'then', 'rewrite', 'poppy', 'i', 'blooms', 'and', 'again', 'a']
```

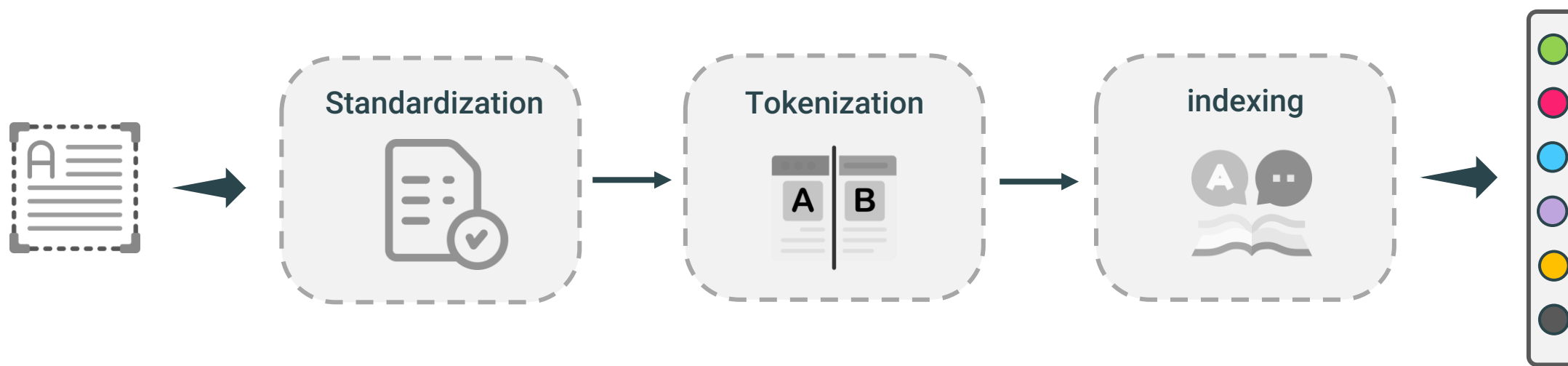
مشاهده خروجی از این لایه



```
test_data = ["write for natural language processing"]  
print(text_vectorization(test_data))
```


تمرین : استفاده از text_vectorization به عنوان یک لایه در شبکه عصبی

Text Vectorizing چه مراحل دارد ؟



تبدیل int ها به بردارهای عددی

تبدیل هر کدام از int ها به بردارهای عددی مثل One Hot یا ...



$[0, 1, \dots, 100]$



1	0	...	0	0
0	1		0	0
0	0		0	0
⋮	⋮		⋮	⋮
⋮	⋮		⋮	⋮
0	0		0	0
0	0		1	0
0	0		0	1

ایده اول : تبدیل کلمات به بردار one-hot

[1 , 4 , 2 , 0 , 3]

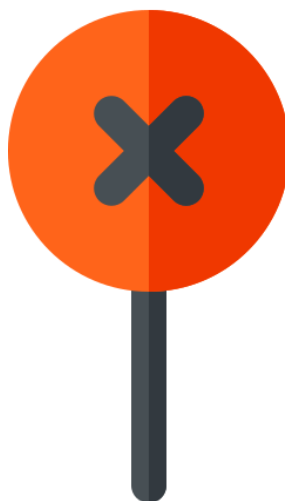
Normal array



$\begin{bmatrix} [0, 1, 0, 0, 0] \\ [0, 0, 0, 0, 1] \\ [0, 0, 1, 0, 0] \\ [1, 0, 0, 0, 0] \\ [0, 0, 0, 1, 0] \end{bmatrix}$

One hot encoding

ایراد این روش چیست ؟



توکن های مختلف مفهومی نسبت به یکدیگر ندارند. به عبارت همه مستقل از یکدیگر هستند.



در حالی که در دنیای واقعی کلمات نزدیکی به هم دارد. مثلا انتظار داریم کلمات گیاه و گل به هم نزدیک باشند تا کلمات گیاه و میز که این اتفاق نمی افتد.



به بیان دیگر فاصله بین دو بردار مربوط به دو کلمه باید بتواند رابطه معنایی آنها را مشخص کند.

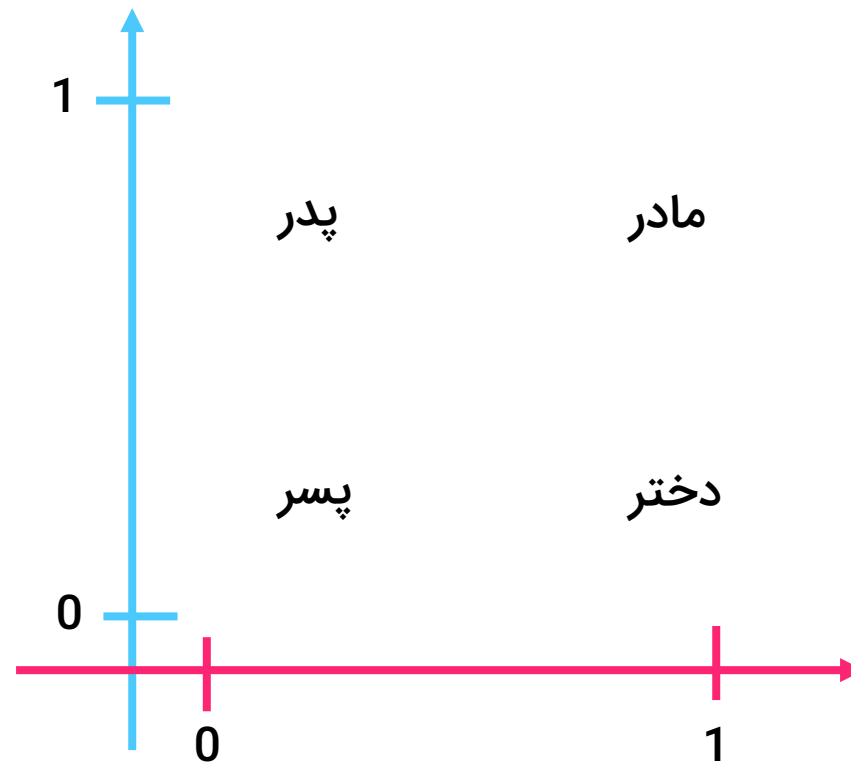
Word Embedding

روشی که در آن کلمات به یک فضای هندسی ساختار یافته map می شود که این قواعد و ساختارها رعایت می شوند.

تفاوت دیگر word embedding با one-hot encoding

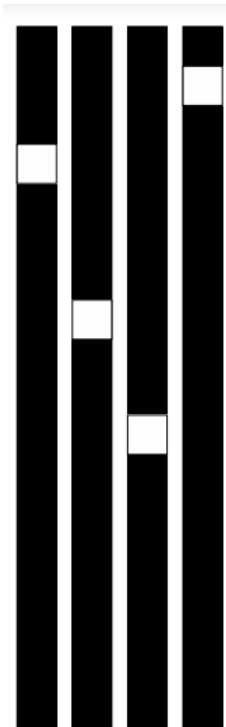
بعد بردار one hot به اندازه تعداد لغات یک دیکشنری است و یک بردار sparse است. در حالی که در word embedding با یک بردار Dense روبرو هستیم.

مثالی برای درک بهتر:



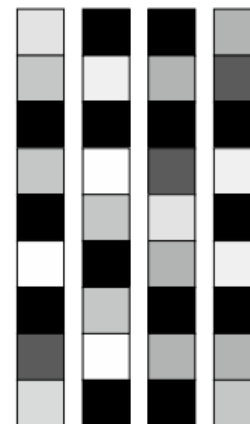
به صورت خلاصه

One Hot Encoding



- ❖ Sparse
- ❖ High-dimensional
- ❖ Hardcoded

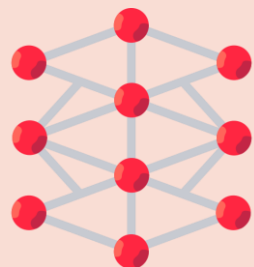
Word Embedding



- ❖ Dense
- ❖ Low-dimensional
- ❖ Learning from data

راه های تبدیل کلمات به بردارهای عددی

Train توسط خودمان !

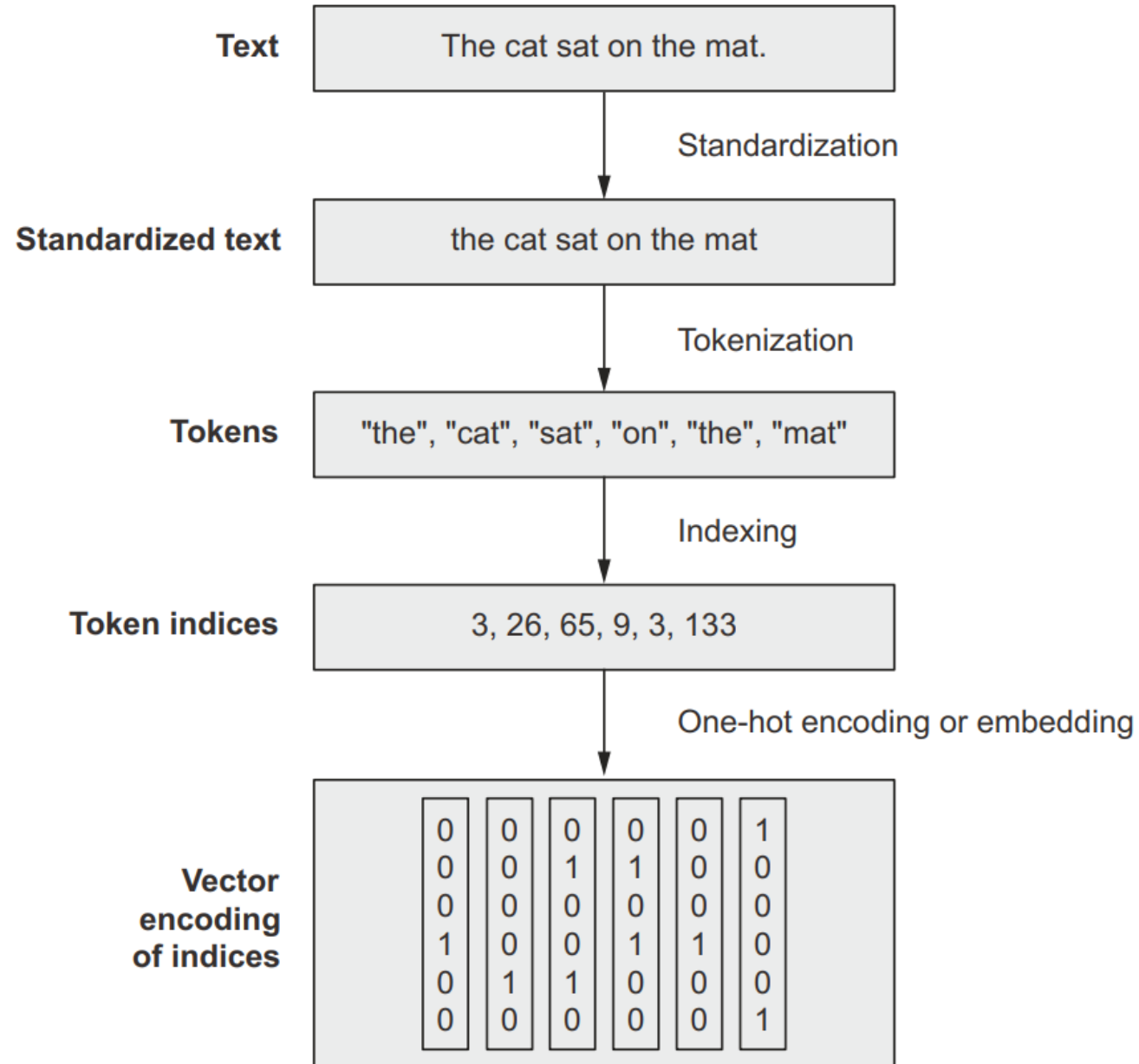


Embedding Layer
Keras

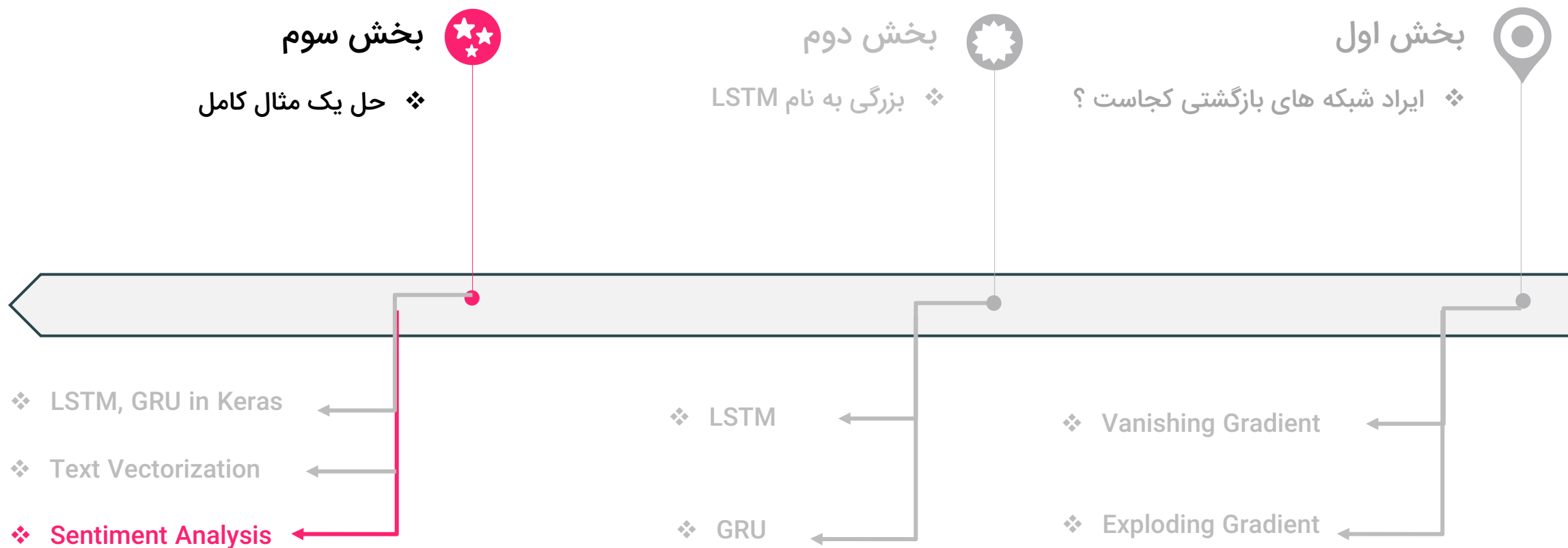
مدل های pre-train



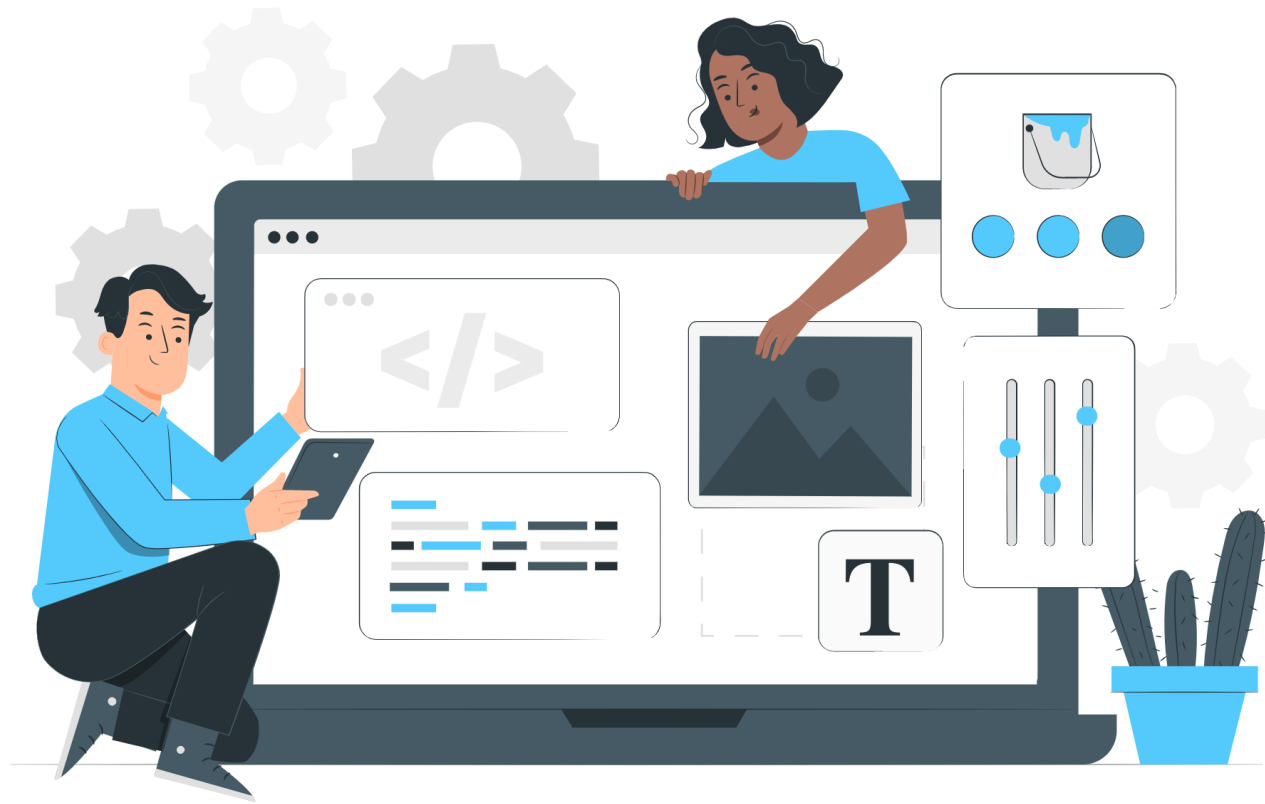
word2Vec, GloVe



آنچه در این جلسه گفته خواهد شد :



حالا بریم سر وقت کد 😊



تابع text_dataset_from_directory

```
main_directory/  
...class_a/  
.....a_text_1.txt  
.....a_text_2.txt  
...class_b/  
.....b_text_1.txt  
.....b_text_2.txt
```

```
train_ds = keras.utils.text_dataset_from_directory(  
    "aclImdb/train", batch_size=batch_size)
```

نتایج بر روی LSTM

```
Epoch 6/10
782/782 [=====] - 54s 69ms/step - loss: 0.5450 - accuracy: 0.7542 - val_loss: 0.5960 - val_accuracy: 0.7330
Epoch 7/10
782/782 [=====] - 55s 70ms/step - loss: 0.5569 - accuracy: 0.7332 - val_loss: 0.6006 - val_accuracy: 0.7168
Epoch 8/10
782/782 [=====] - 54s 69ms/step - loss: 0.5480 - accuracy: 0.7464 - val_loss: 0.5952 - val_accuracy: 0.7399
Epoch 9/10
782/782 [=====] - 56s 71ms/step - loss: 0.5394 - accuracy: 0.7564 - val_loss: 0.5816 - val_accuracy: 0.7342
Epoch 10/10
782/782 [=====] - 54s 69ms/step - loss: 0.5186 - accuracy: 0.7705 - val_loss: 0.5255 - val_accuracy: 0.7906
782/782 [=====] - 17s 22ms/step - loss: 0.5255 - accuracy: 0.7906
Test acc: 0.791
```


نتایج بر روی GRU

```
Epoch 6/10
782/782 [=====] - 54s 69ms/step - loss: 0.6667 - accuracy: 0.5227 - val_loss: 0.7214 - val_accuracy: 0.4961
Epoch 7/10
782/782 [=====] - 53s 67ms/step - loss: 0.7067 - accuracy: 0.5228 - val_loss: 0.7196 - val_accuracy: 0.4975
Epoch 8/10
782/782 [=====] - 52s 67ms/step - loss: 0.6912 - accuracy: 0.5266 - val_loss: 0.7251 - val_accuracy: 0.5029
Epoch 9/10
782/782 [=====] - 54s 68ms/step - loss: 0.7153 - accuracy: 0.5237 - val_loss: 0.7110 - val_accuracy: 0.4982
Epoch 10/10
782/782 [=====] - 56s 71ms/step - loss: 0.6712 - accuracy: 0.5244 - val_loss: 0.7264 - val_accuracy: 0.4983
```

معماری چند لایه GRU

```
x = layers.GRU(32, return_sequences=True)(embedded)
x = layers.GRU(64, return_sequences=True)(x)
x = layers.GRU(64)(x)
```

نتایج بر روی چند لایه از GRU

```
Epoch 1/10
782/782 [=====] - 141s 172ms/step - loss: 0.6938 - accuracy: 0.4987 - val_loss: 0.6931 - val_accuracy: 0.5029
Epoch 2/10
782/782 [=====] - 142s 182ms/step - loss: 0.6925 - accuracy: 0.5048 - val_loss: 0.6991 - val_accuracy: 0.5011
Epoch 3/10
782/782 [=====] - 141s 180ms/step - loss: 0.6845 - accuracy: 0.5133 - val_loss: 0.7112 - val_accuracy: 0.4982
Epoch 4/10
782/782 [=====] - 145s 186ms/step - loss: 0.6758 - accuracy: 0.5217 - val_loss: 0.7192 - val_accuracy: 0.4984
Epoch 5/10
782/782 [=====] - 142s 182ms/step - loss: 0.6718 - accuracy: 0.5238 - val_loss: 0.7224 - val_accuracy: 0.5028
Epoch 6/10
782/782 [=====] - 143s 183ms/step - loss: 0.6652 - accuracy: 0.5239 - val_loss: 0.7349 - val_accuracy: 0.5034
Epoch 7/10
782/782 [=====] - 158s 203ms/step - loss: 0.6621 - accuracy: 0.5301 - val_loss: 0.7511 - val_accuracy: 0.5027
Epoch 8/10
782/782 [=====] - 155s 198ms/step - loss: 0.6599 - accuracy: 0.5263 - val_loss: 0.7678 - val_accuracy: 0.5028
Epoch 9/10
782/782 [=====] - 144s 184ms/step - loss: 0.6592 - accuracy: 0.5276 - val_loss: 0.7625 - val_accuracy: 0.5031
Epoch 10/10
782/782 [=====] - 135s 172ms/step - loss: 0.6556 - accuracy: 0.5317 - val_loss: 0.7577 - val_accuracy: 0.5028
782/782 [=====] - 17s 21ms/step - loss: 0.6933 - accuracy: 0.5024
```

میتونید کاری کنید که GRU هم جواب بده ؟



نتایج بر روی Bidirectional LSTM

Epoch 1/10

782/782 [=====] - 103s 125ms/step - loss: 0.4628 - accuracy: 0.7960 - val_loss: 0.4484 - val_accuracy: 0.8272

Epoch 2/10

782/782 [=====] - 96s 122ms/step - loss: 0.3003 - accuracy: 0.8904 - val_loss: 0.3433 - val_accuracy: 0.8652

آنچه در این جلسه گفته خواهد شد :

