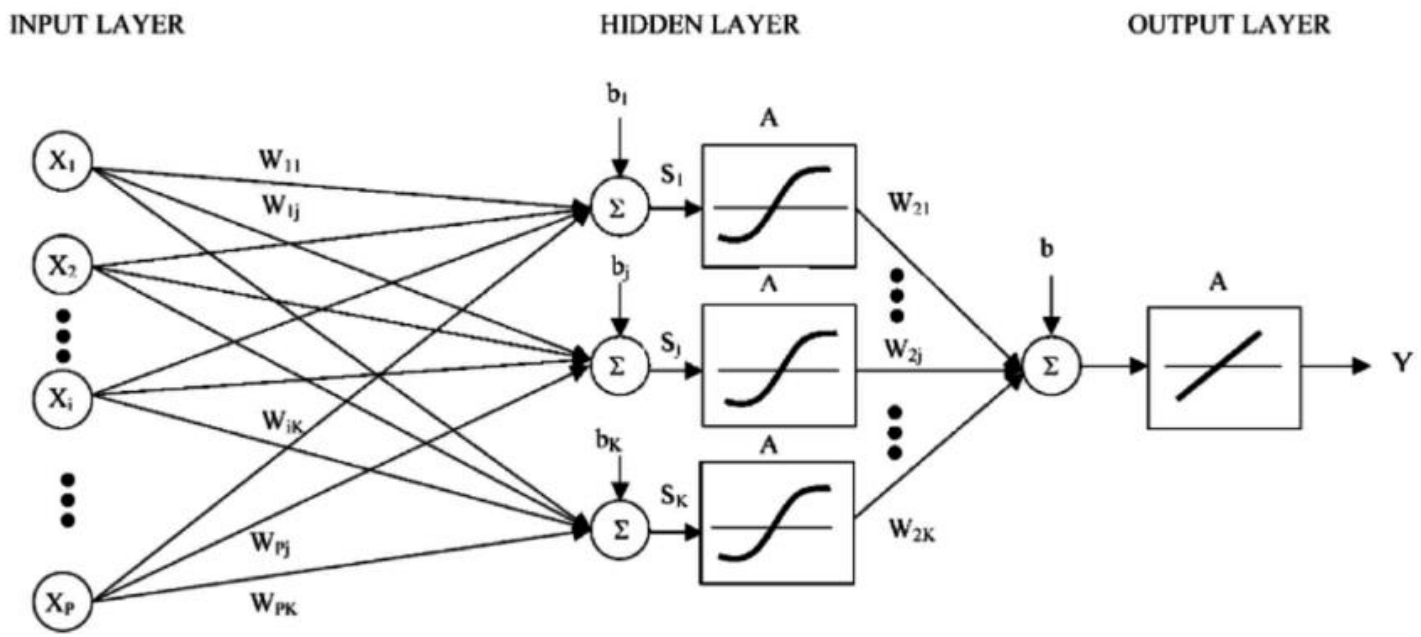
Deep learning in NLP

Part II: Introduction to RNN and LSTM

Zeinab Rahimi, Feb 2022

شبکه های عصبی سنتی

- لایه ورودی
- لایه هیدن
- لایه خروجی
- فعالسازی

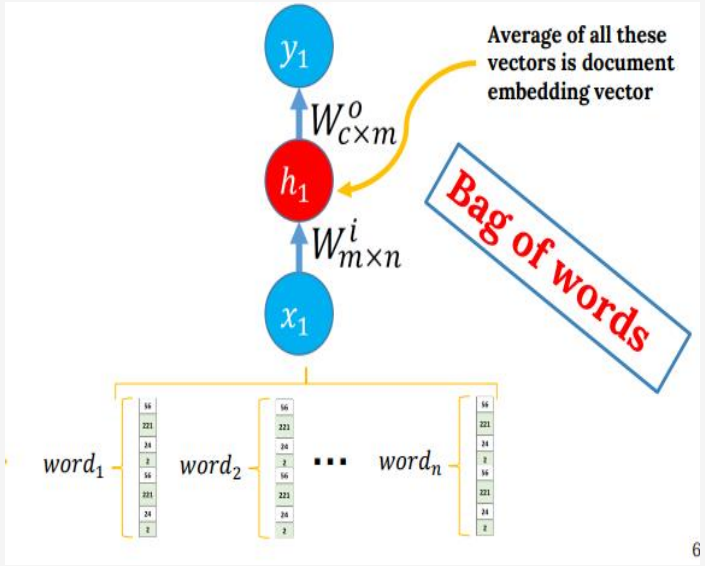


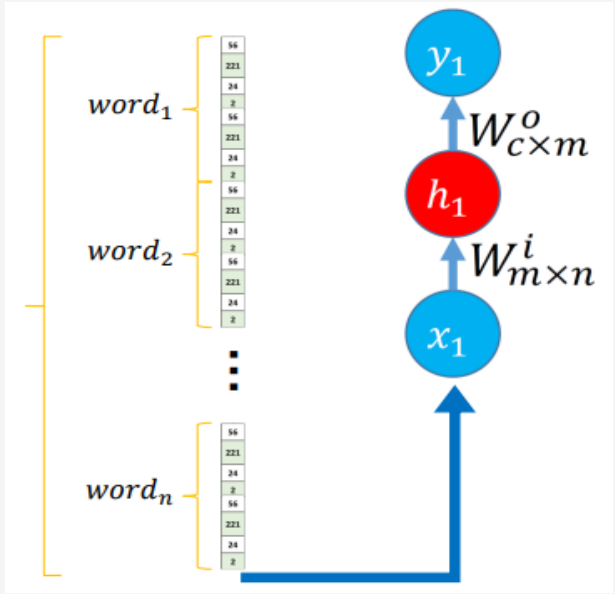
مشکلات شبکه های قبلی برای NLP

- در تسک های قبلی مثل Word2vec از شبکه های FullyConnected استفاده می شد:
  - ورود یک کلمه به شبکه و انجام تسک طبقه بندی
  - در هر مرحله خروجی بر اساس ورودی همان مرحله مثلا دسته بندی بر اساس آخرین ورودی

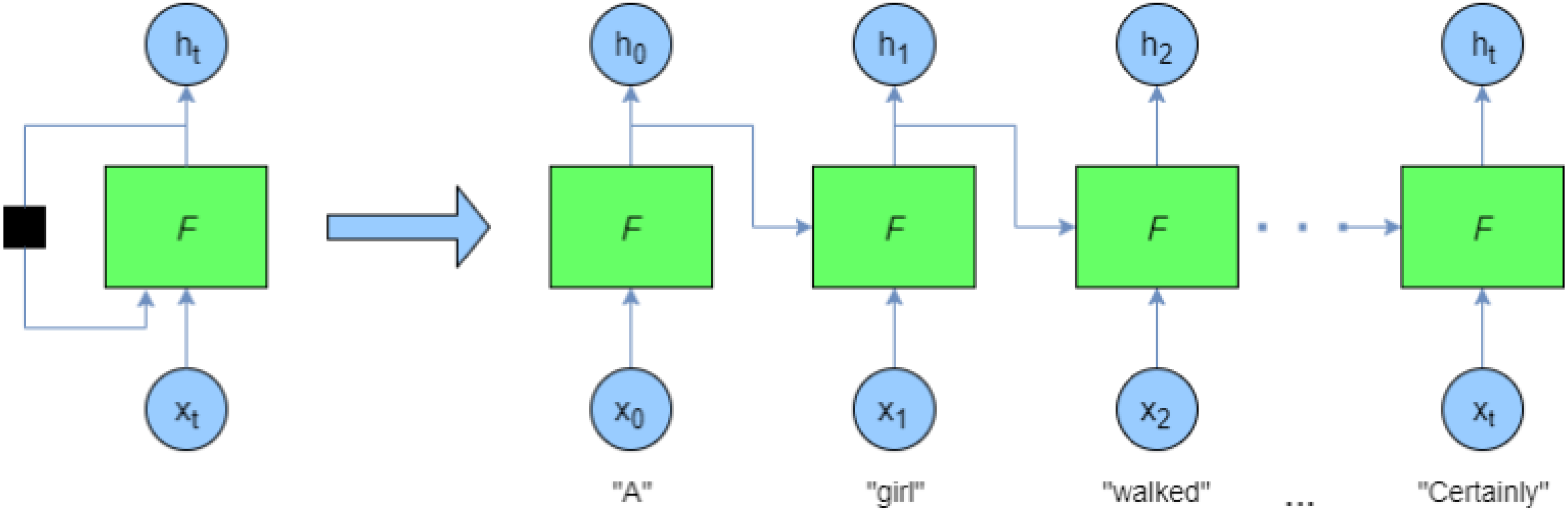
- نیاز به تصمیم گیری بعد از دیدن یک رشته توکن با ترتیب (seq)
  - نیاز به ورود یک جمله یا سند برای تسک های پیچیده تر
  - نیاز به داشتن اطلاعات از کلمات قبل و بعد در رشته ورودی
  - نیاز به در نظر گرفتن ترتیب توکن ها
  - خروجی باید به ورودی فعلی و خروجی قبلی وابسته باشد.

راه حل برای نواقص شبکه های سنتی





- میانگین گرفتن از بردارهای جاسازی
  - در نظر نگرفتن توالی و ساختار جمله
  - مناسب نبودن برای تسک های پیچیده تر
- به هم وصل کردن بردارهای جاسازی
  - ثابت نبودن طول ورودی شبکه
  - زیاد بودن طول ورودی
- معرفی شبکه های بازگشتی (RNN)
  - همان FC ها با این تفاوت که لایه های هیدن آنها به هم وصل است.
  - در FC اطلاعی از کلمات قبل و بعد نداریم اما در RNN در هر لحظه از قبلی ها مطلعیم.
  - در هر زمان با ورود هر توکن اطلاعاتی در لایه هیدن ذخیره می شود و به لایه هیدن در زمان بعدی منتقل می شود.

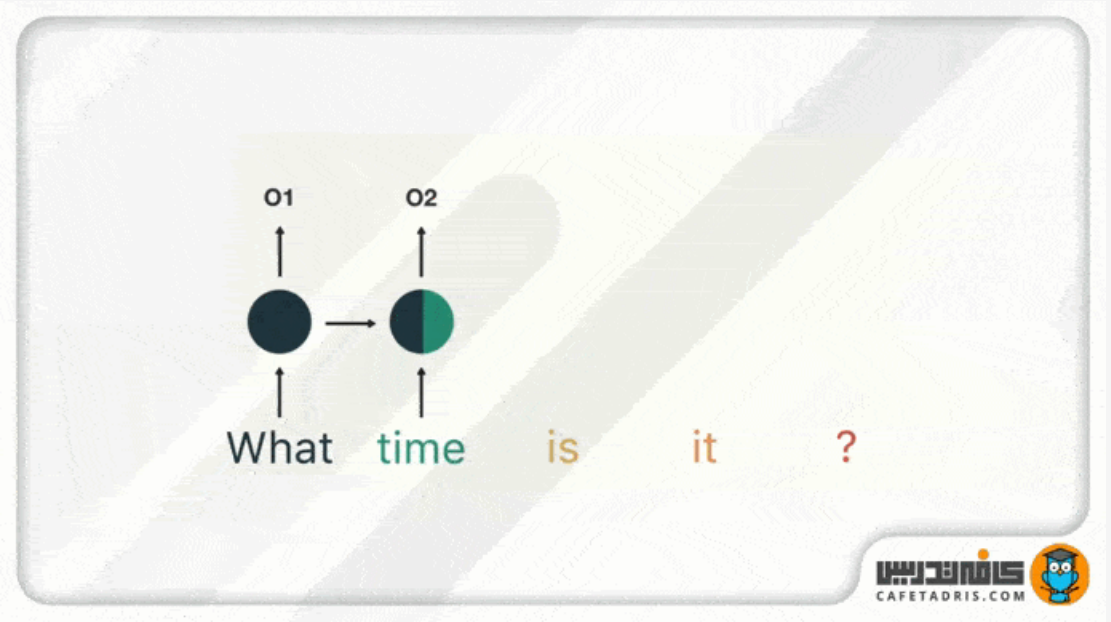


Recurrent Neural Network

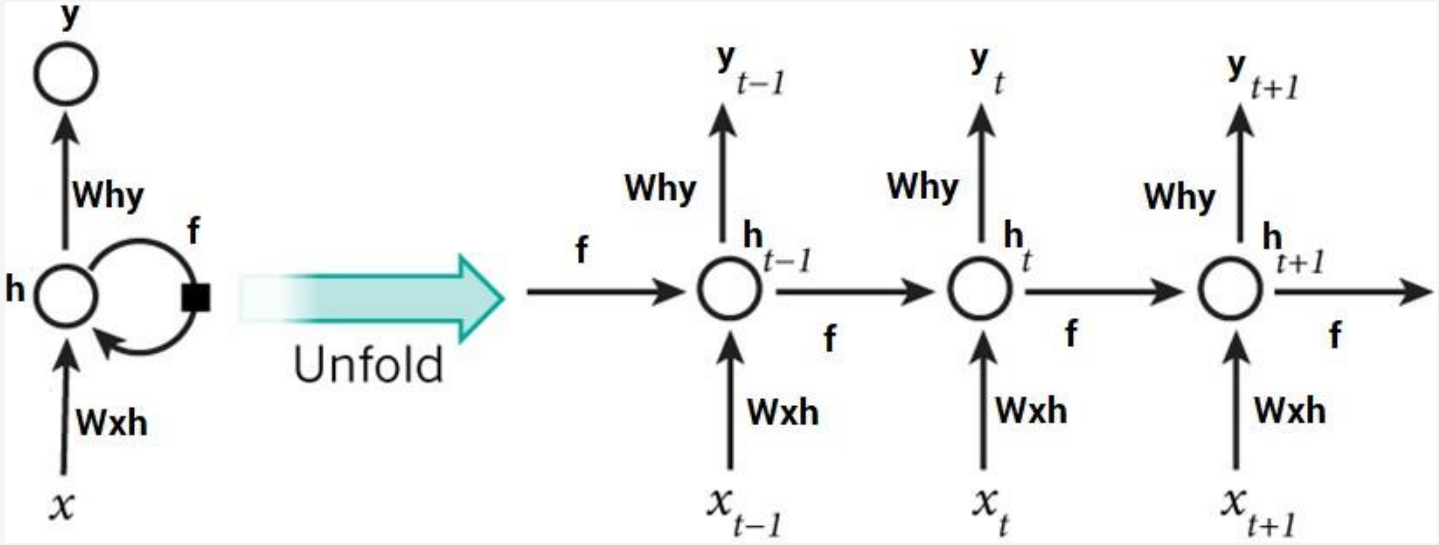
از خروجی قبلی به عنوان ورودی بعدی استفاده می کند.

شبکه های عصبی بازگشتی

- شکستن جمله به کلمات
- شبکه‌ی RNN به صورت ترتیبی کار می‌کند، در هر گام یک کلمه را به آن وارد می شود.(بردار کلمه)
- زمانی‌که به آخر جمله برسیم می‌بینیم که شبکه‌ی اطلاعات مربوط به تمامی کلمه‌های موجود در جمله را پردازش کرده است.



شبکه های عصبی بازگشتی

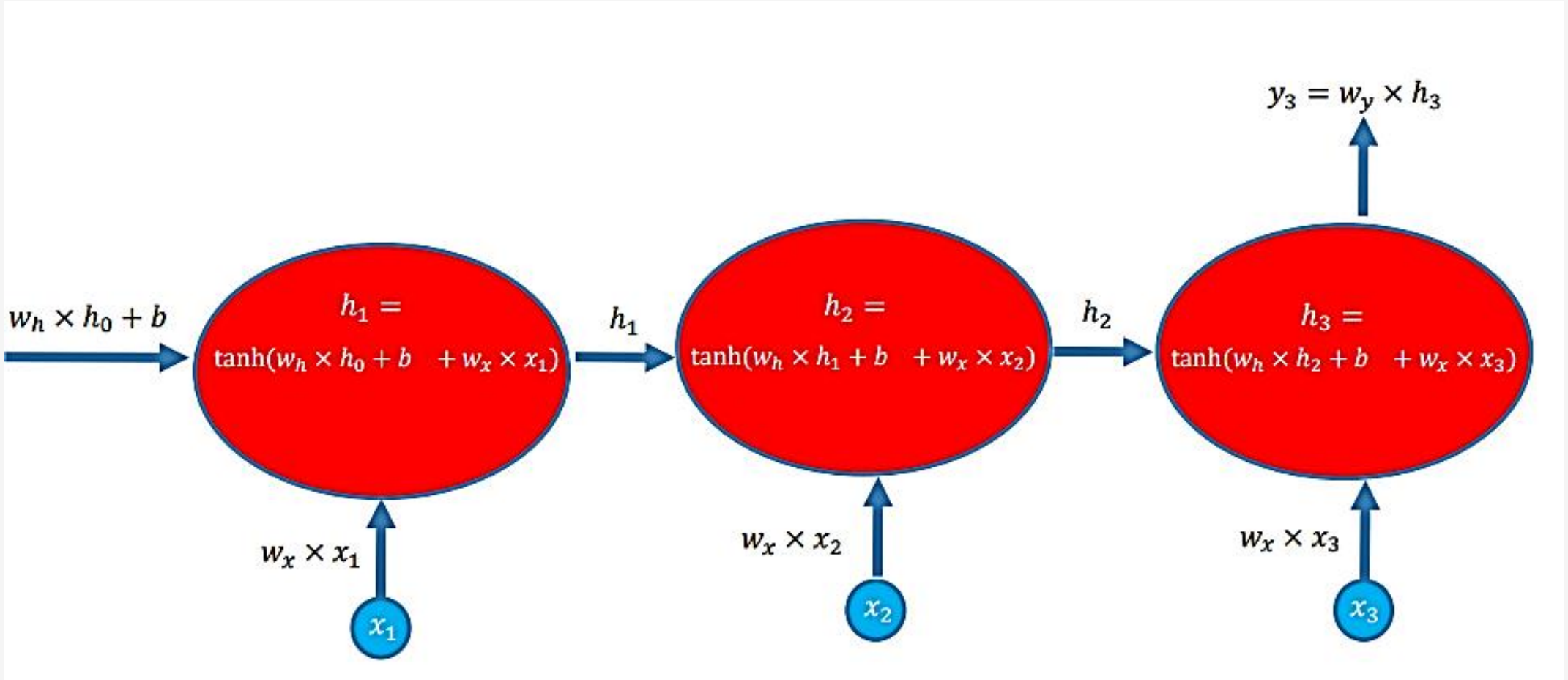


$h_t = f(h_{t-1}, x_t)$

$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$

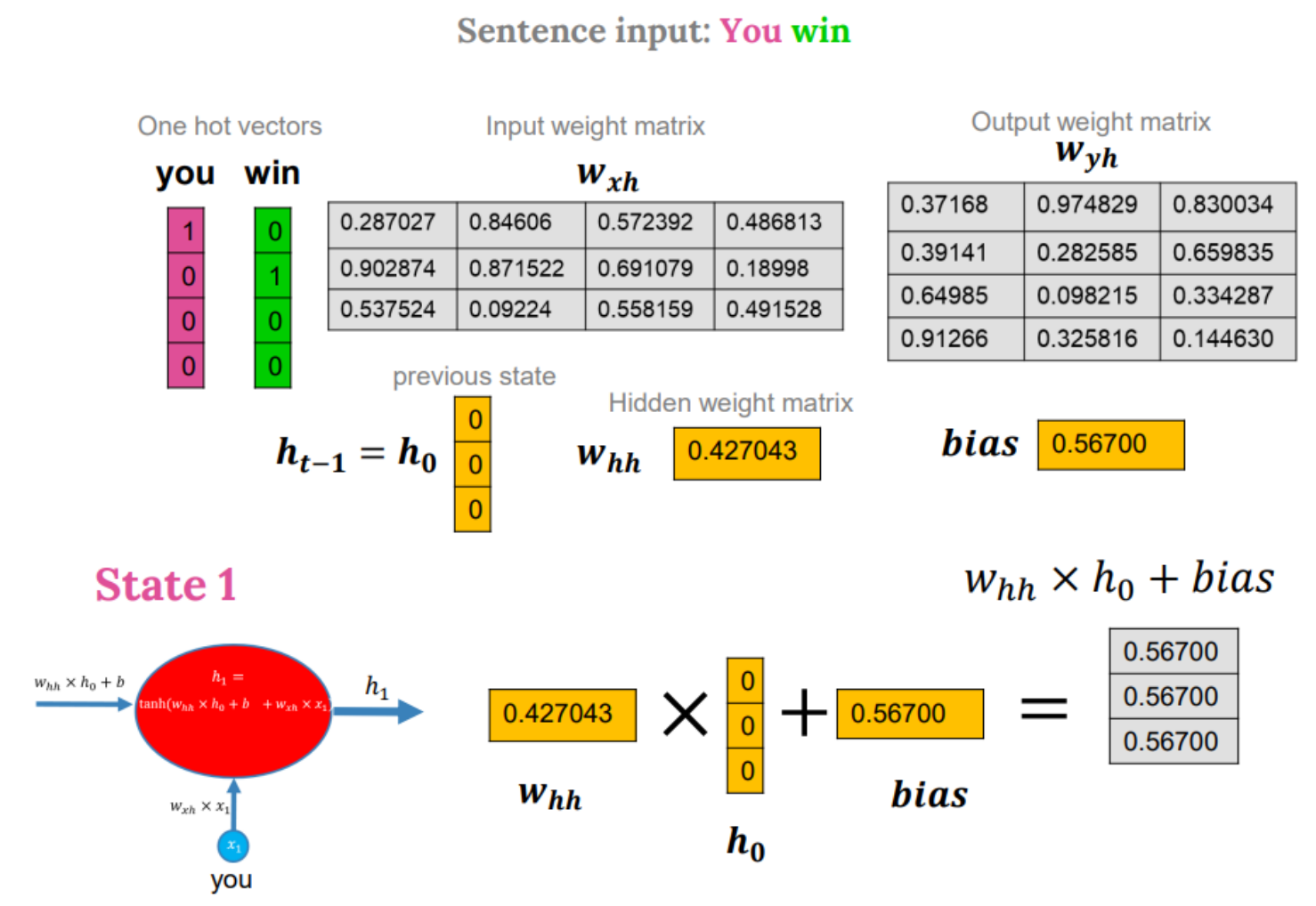
$y_t = W_{hy}h_t$

شبکه های عصبی بازگشتی: روابط



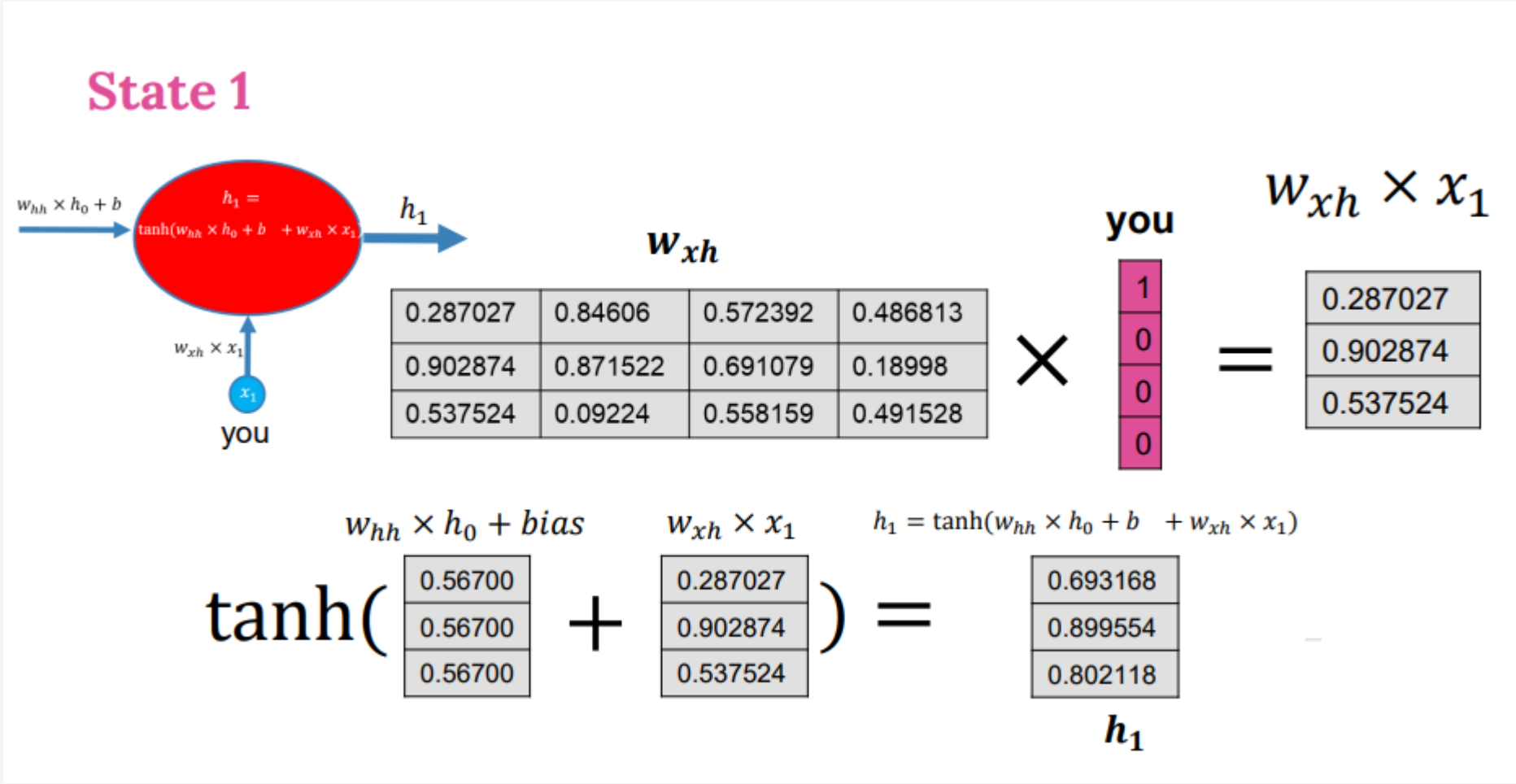
مثال برداشته شده از Deep NLP workshop (IKT,2021)

شبکه های عصبی بازگشتی: مثال

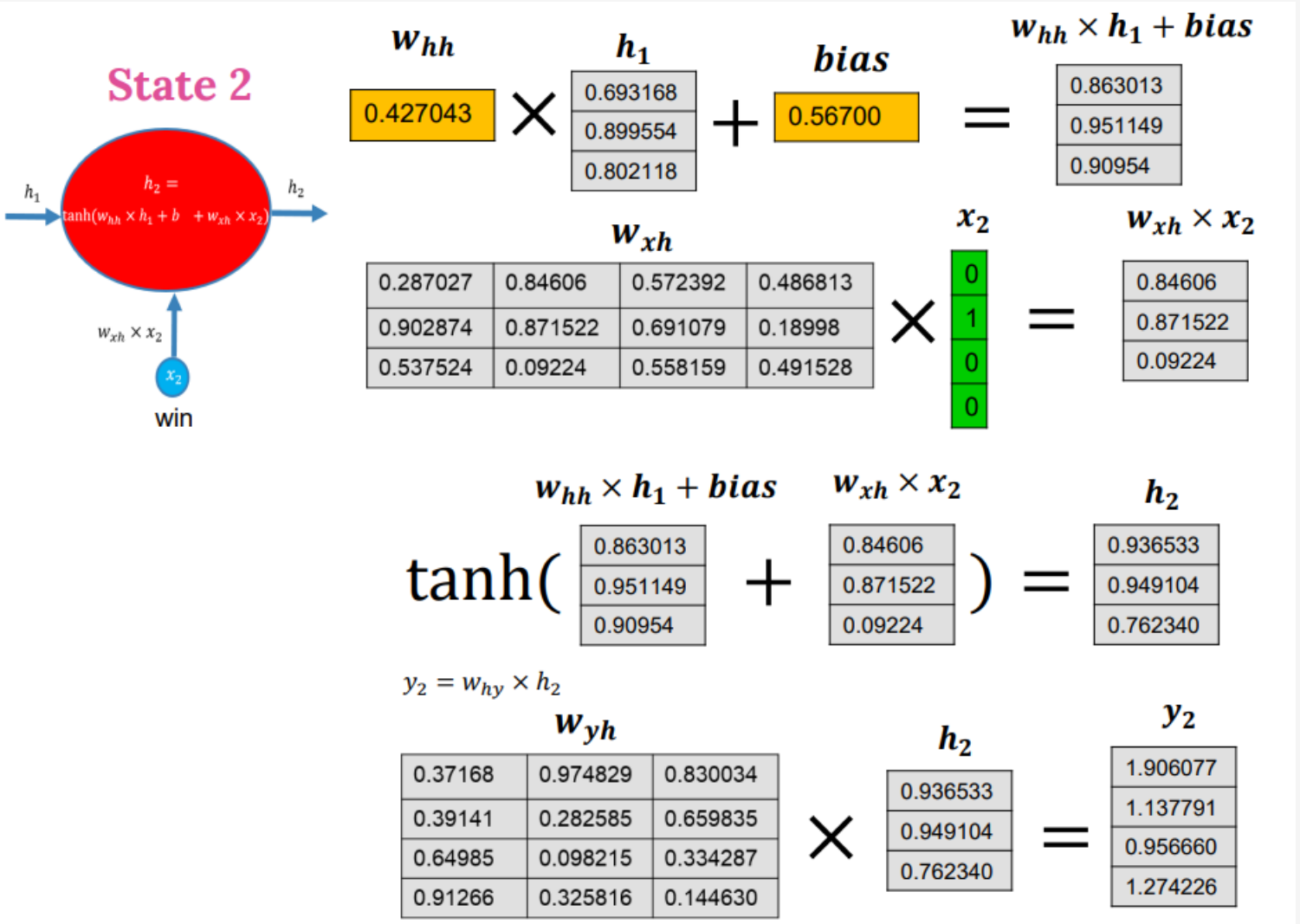


مثال برداشته شده از Deep NLP workshop (IKT,2021)

شبکه های عصبی بازگشتی: مثال



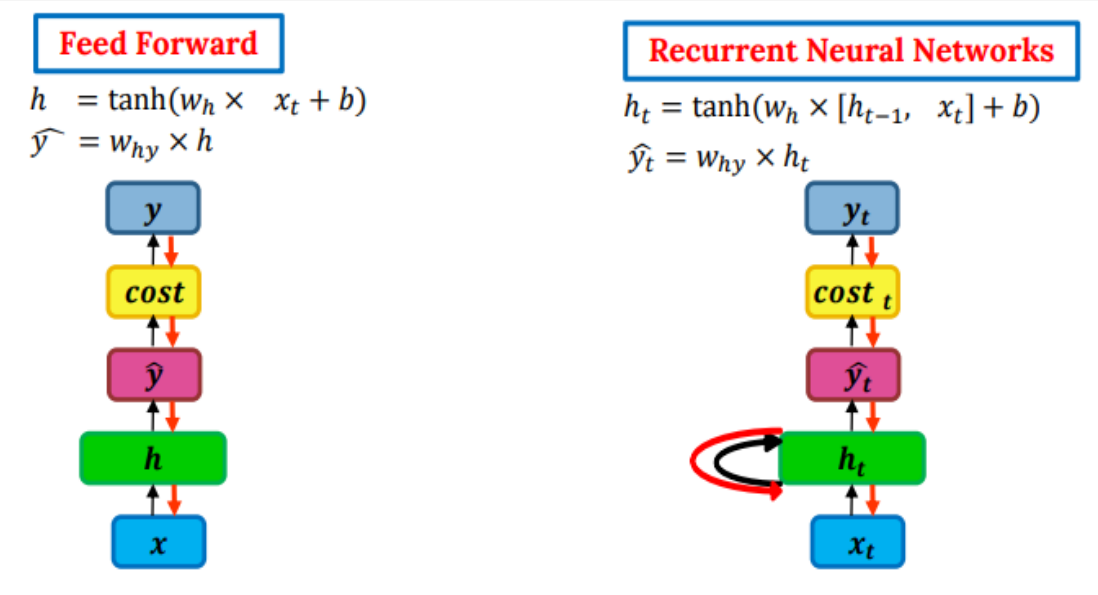
شبکه های عصبی بازگشتی: مثال

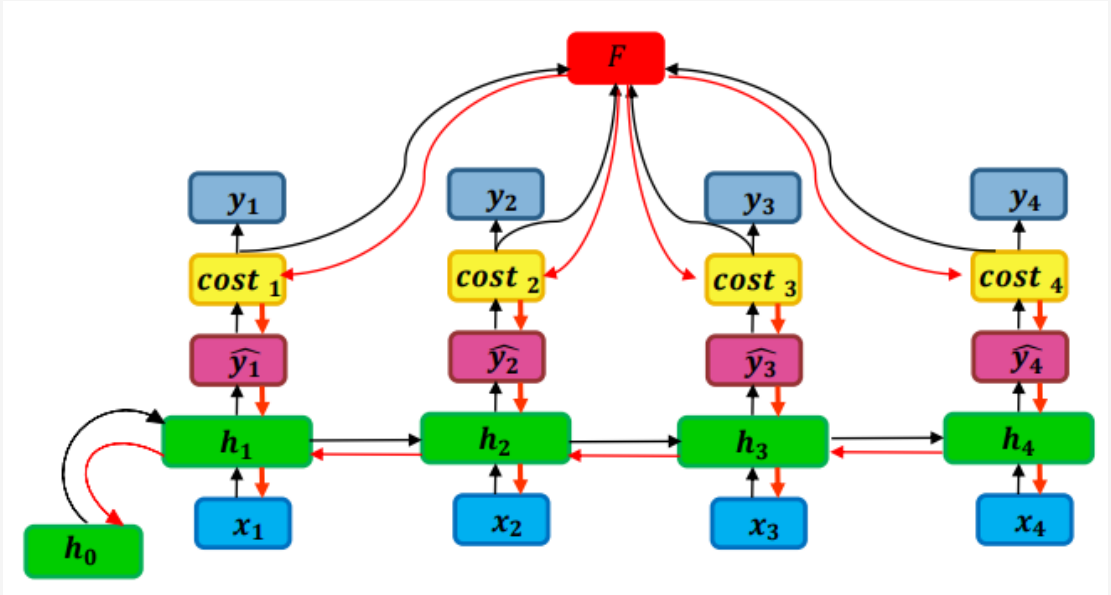


$$Softmax\left(\begin{matrix}1.906077\\1.137791\\0.956660\\1.274226\end{matrix}\right) = \begin{matrix}0.41974\\0.1946\\0.1624\\0.2231\end{matrix} \quad \begin{matrix}1\\0\\0\\0\end{matrix}$$

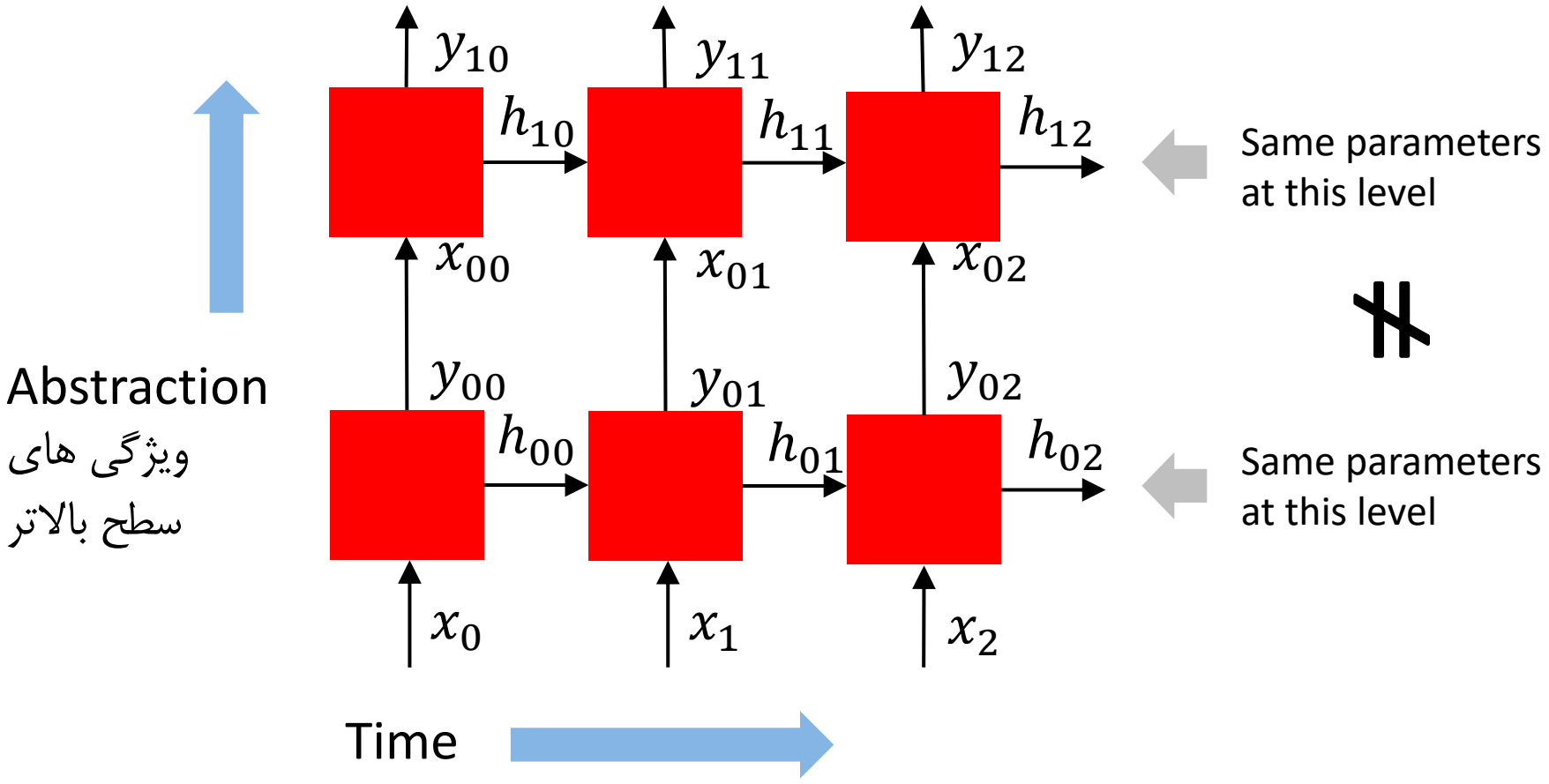
y_2 $Softmax(y_t)$

شبکه های عصبی بازگشتی: Back Propagation

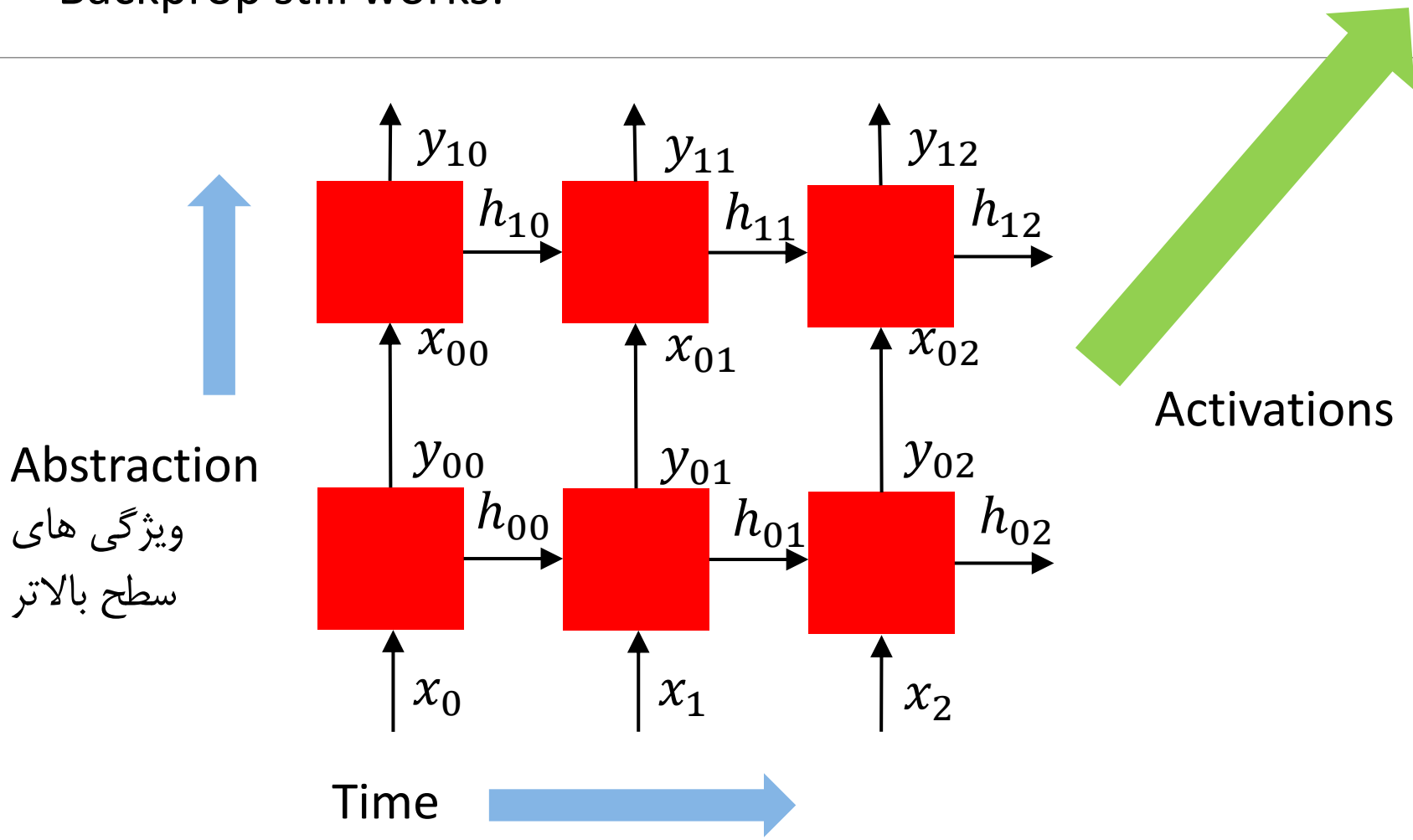




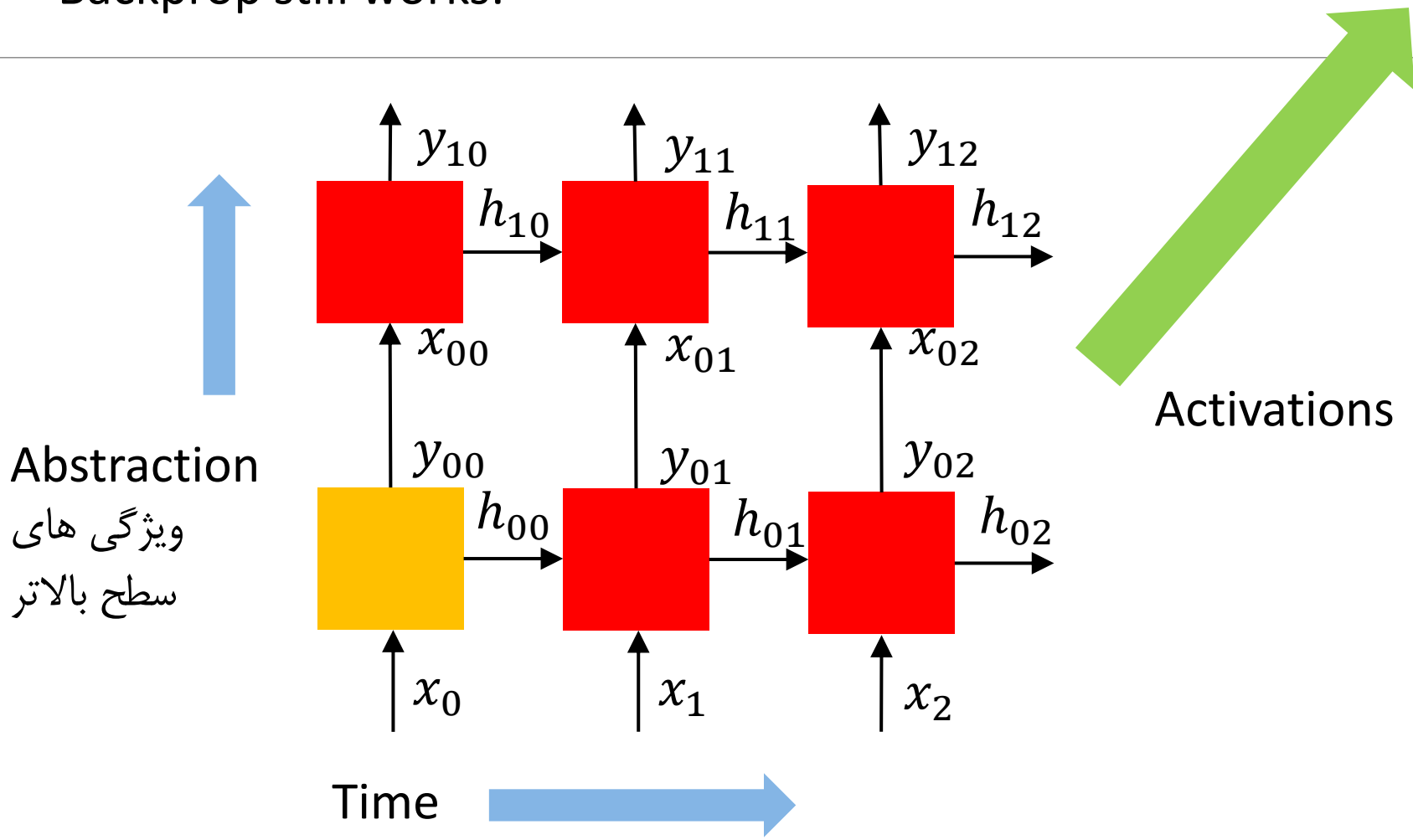
در RNN عمیق غالبا لایه ها به صورت عمودی استک می شوند.



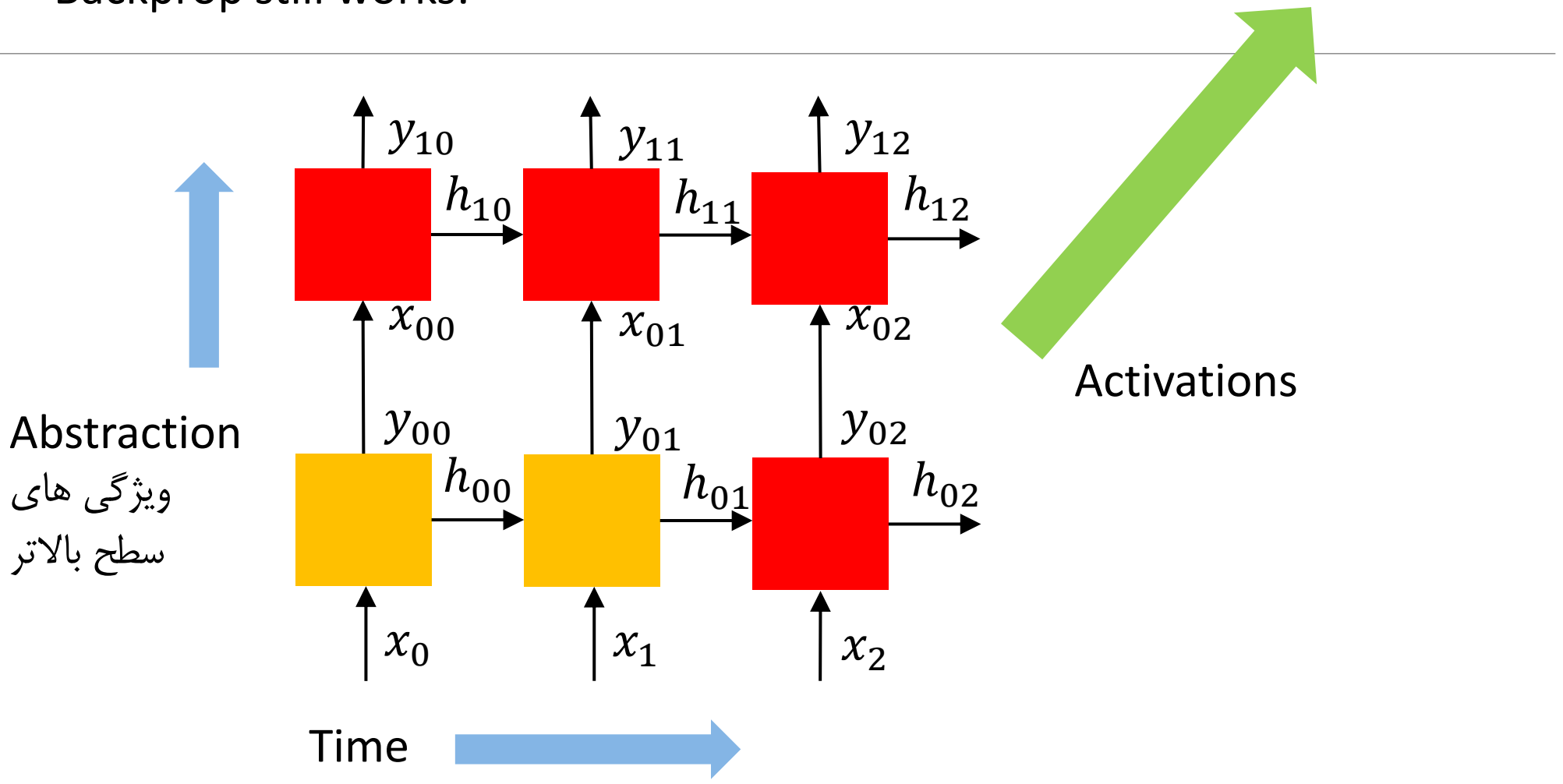
Backprop still works:



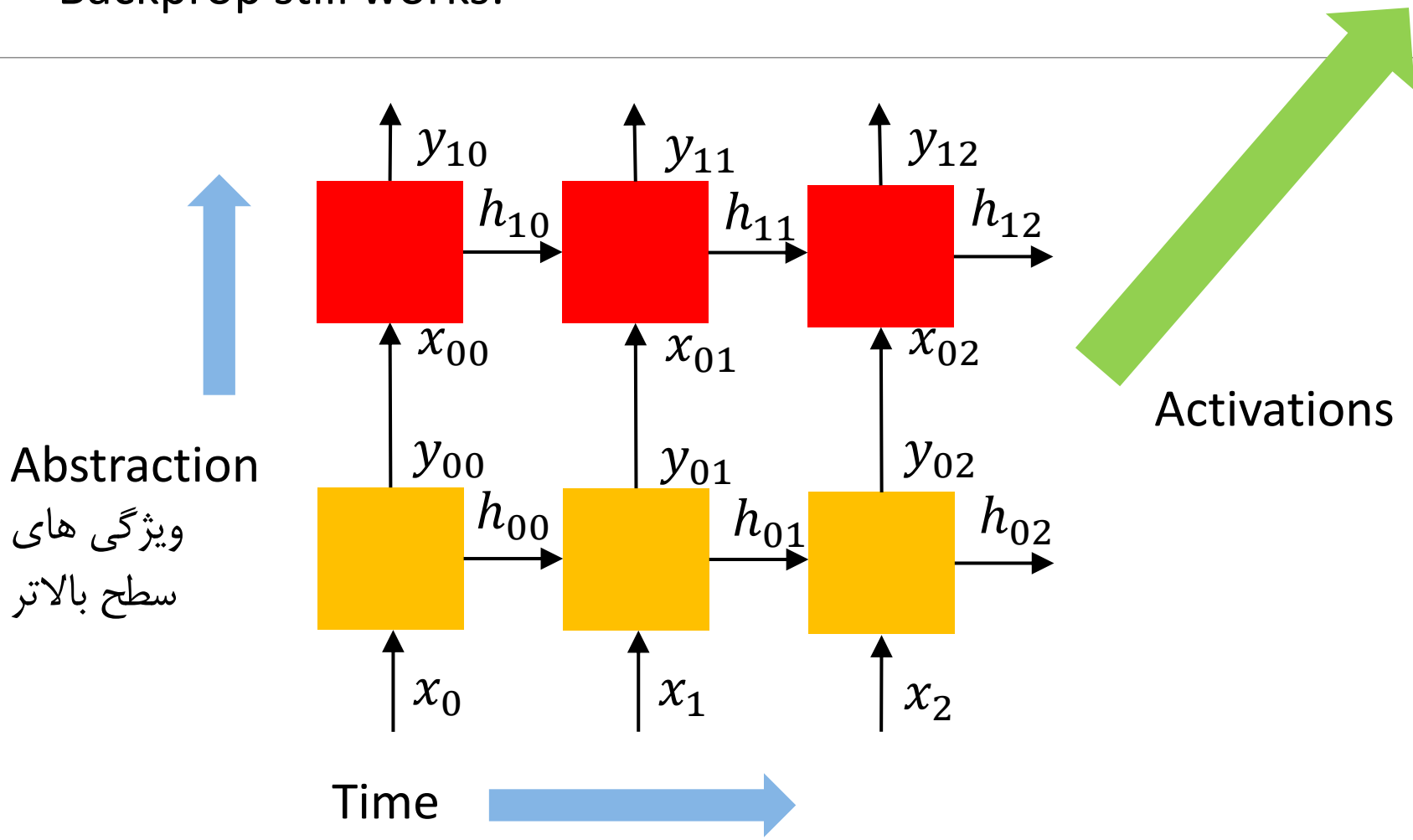
Backprop still works:



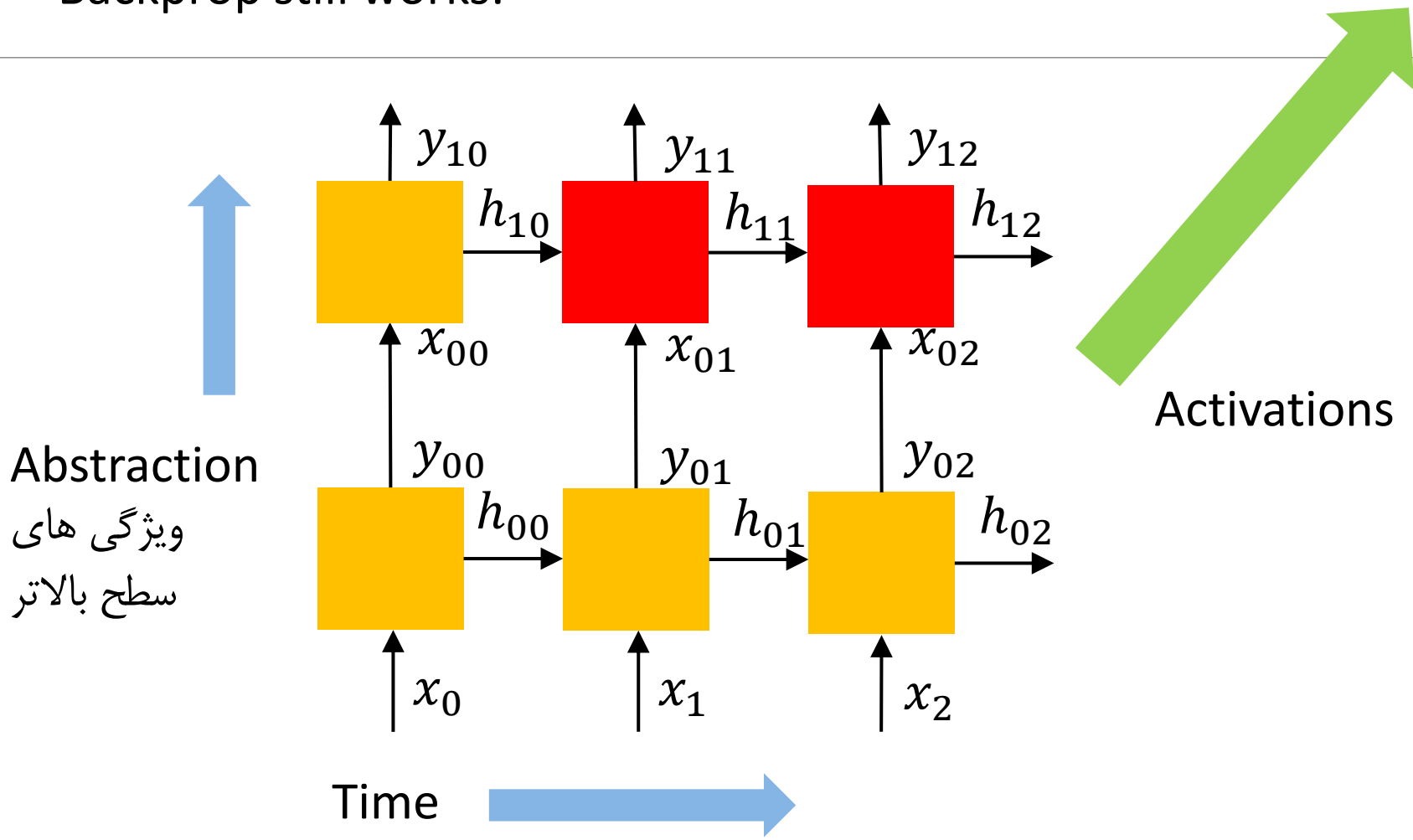
Backprop still works:



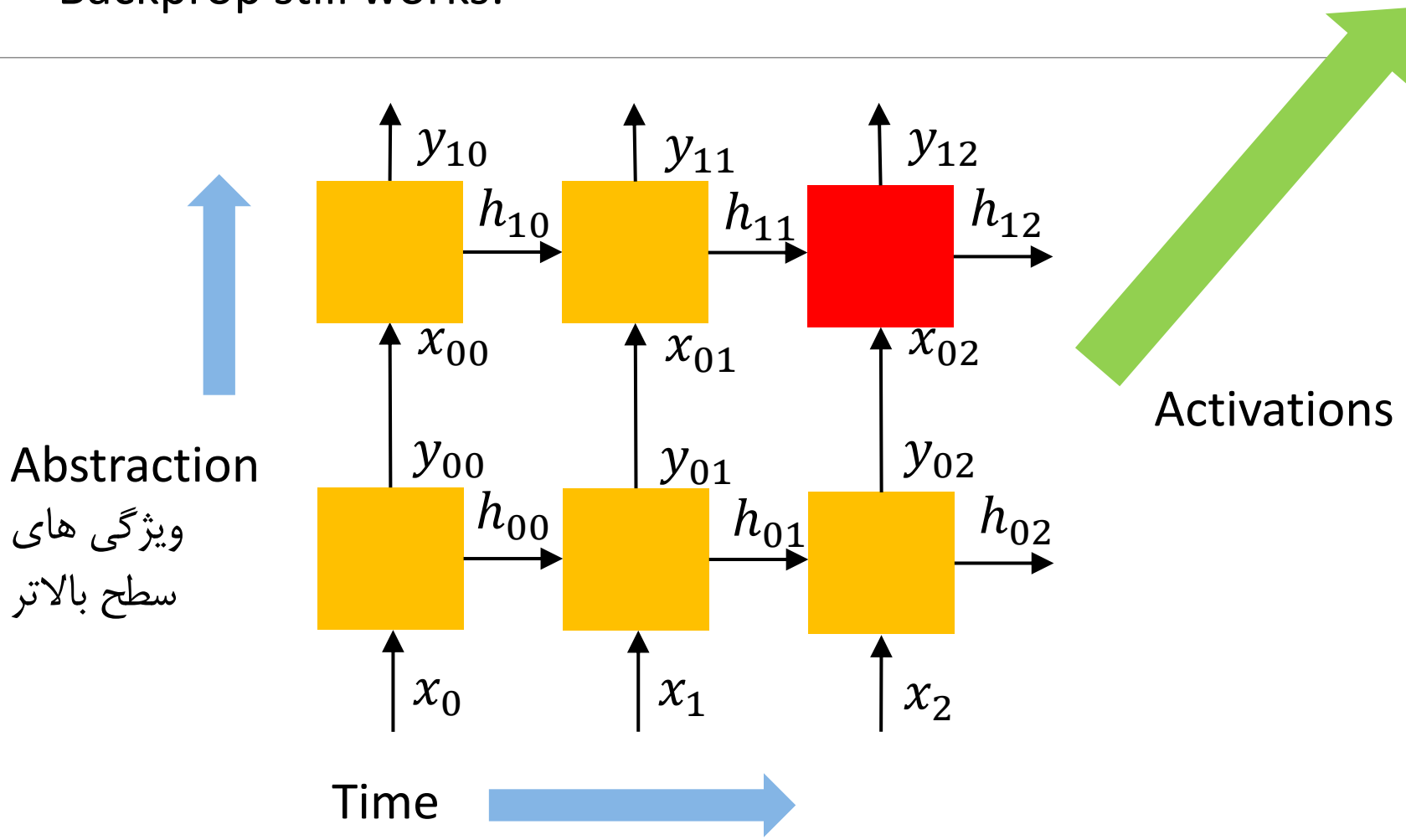
Backprop still works:



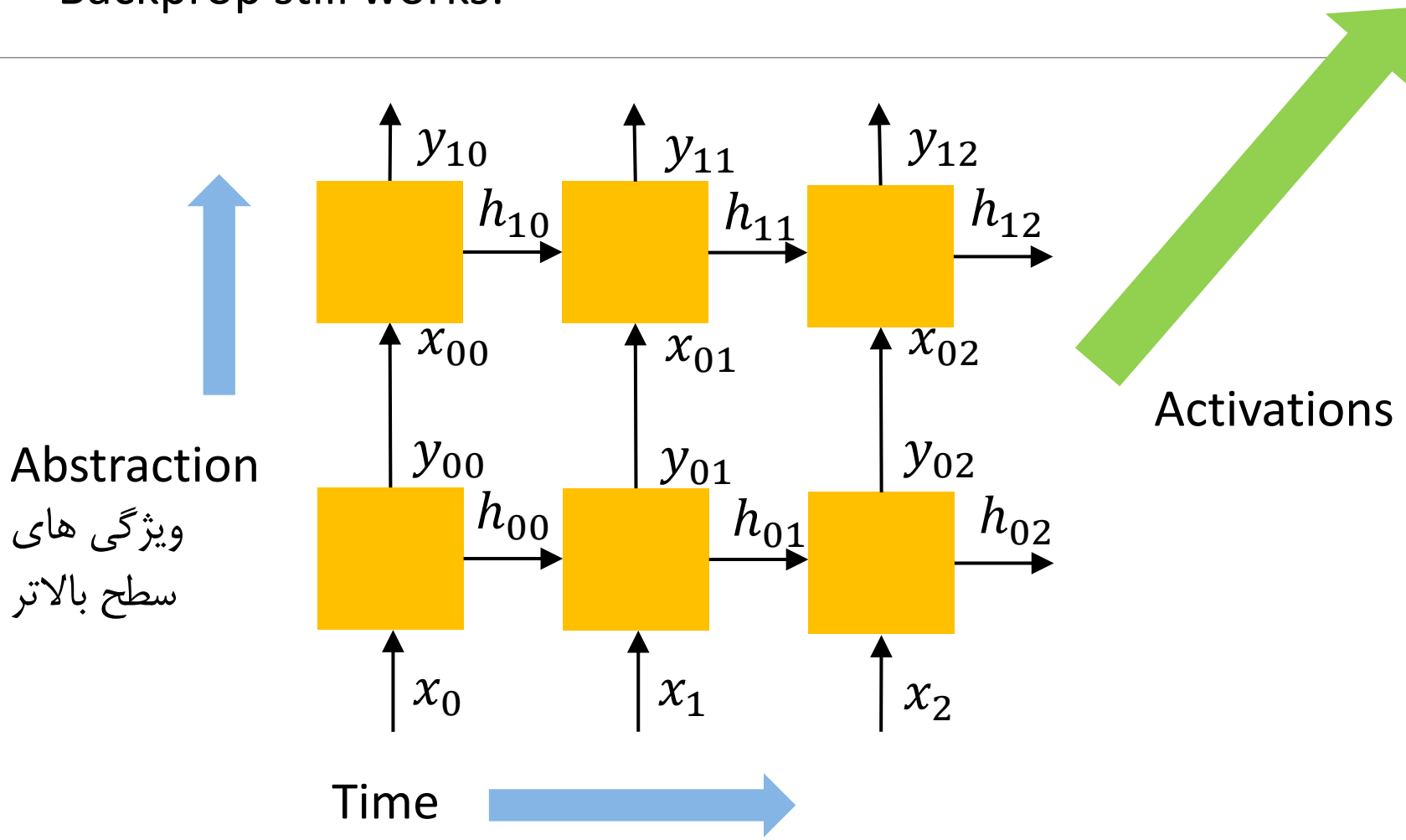
Backprop still works:



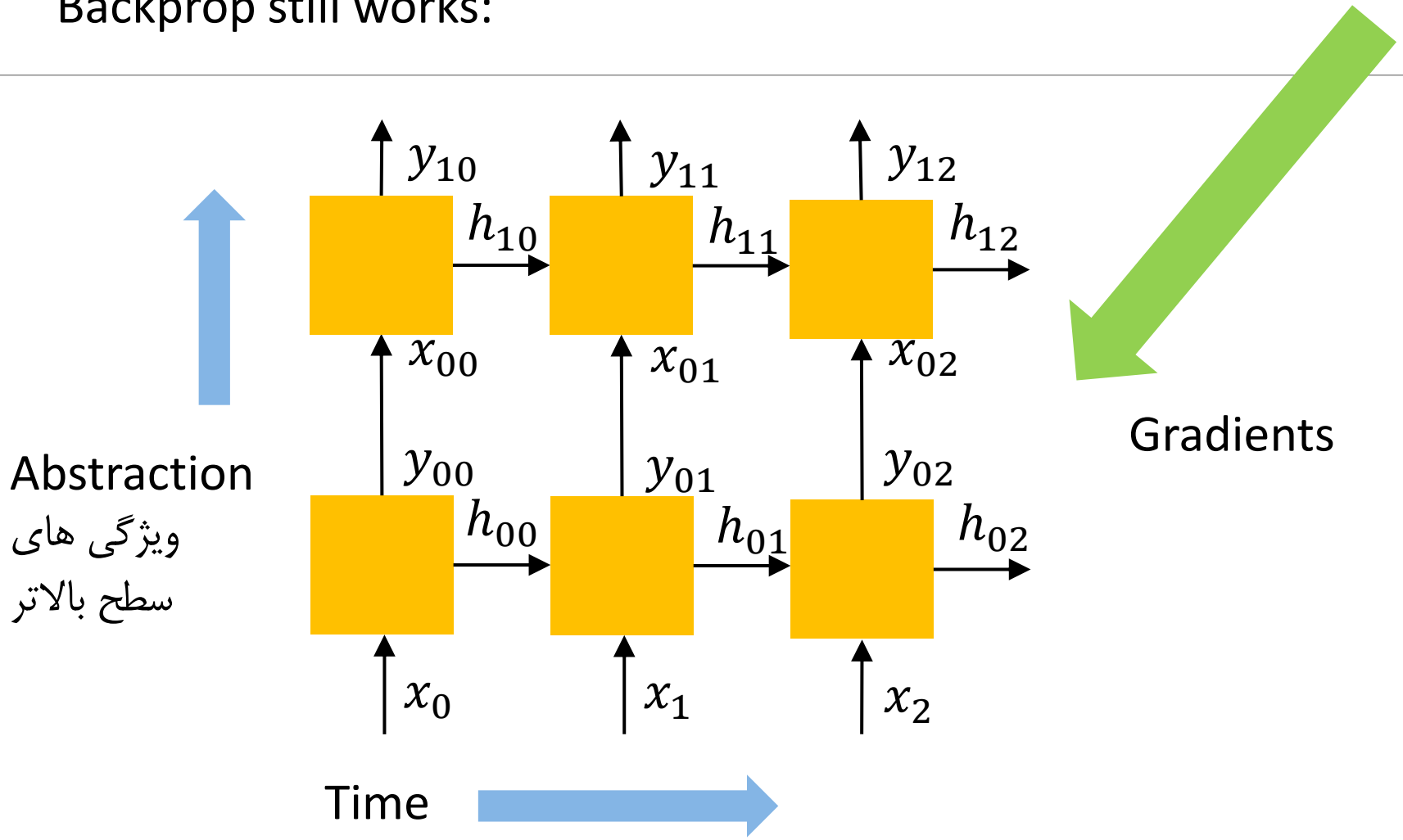
Backprop still works:



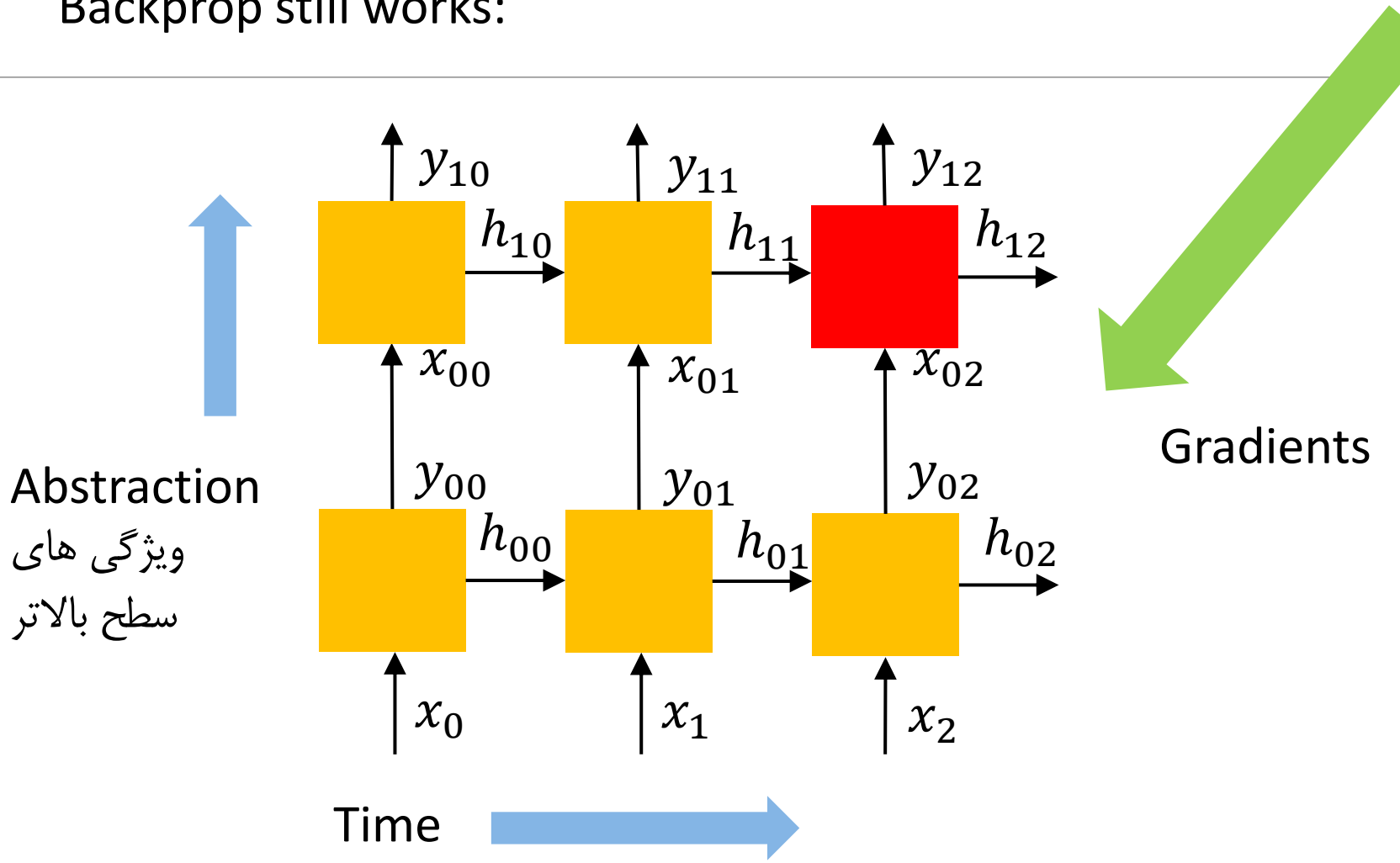
Backprop still works:



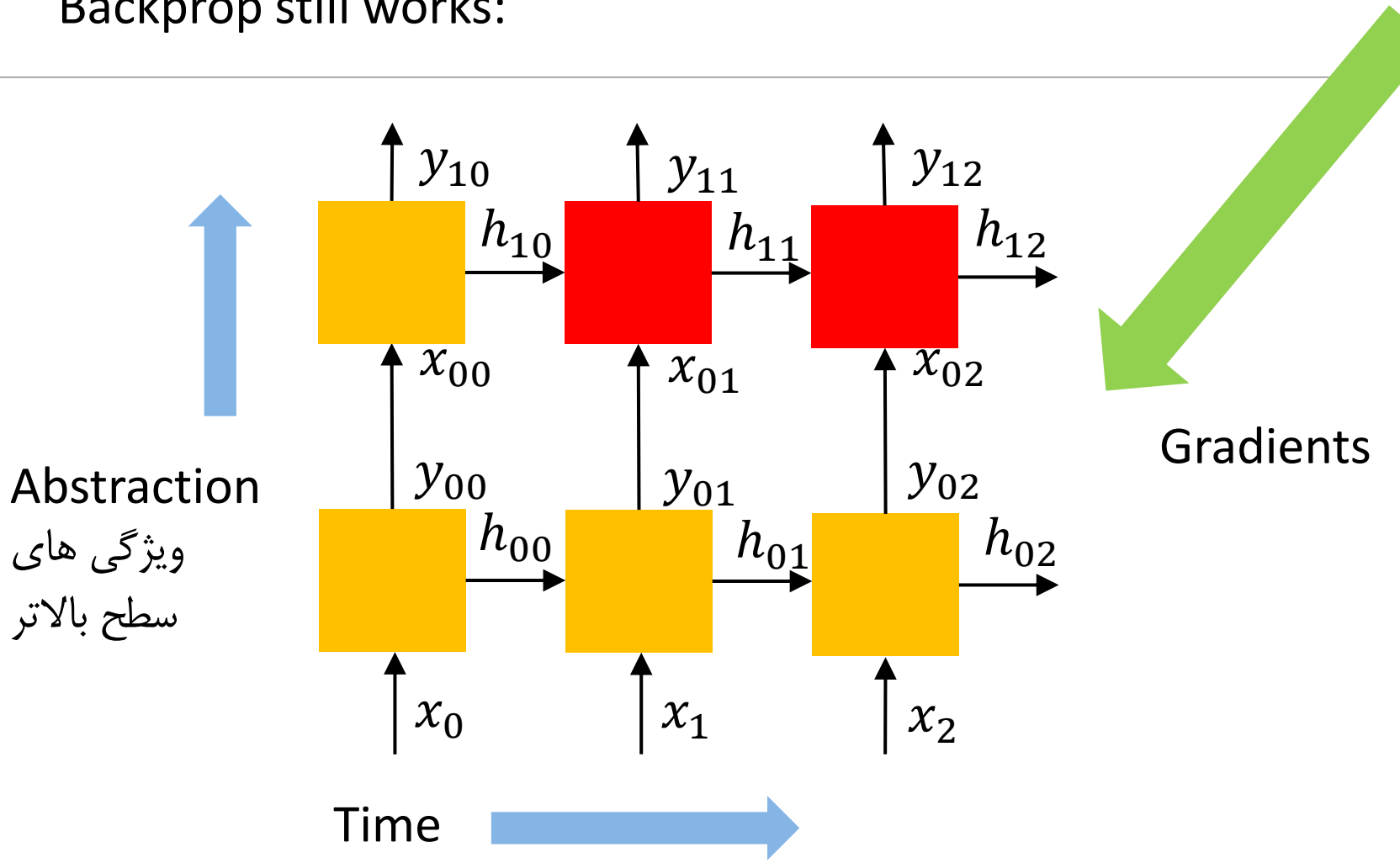
Backprop still works:



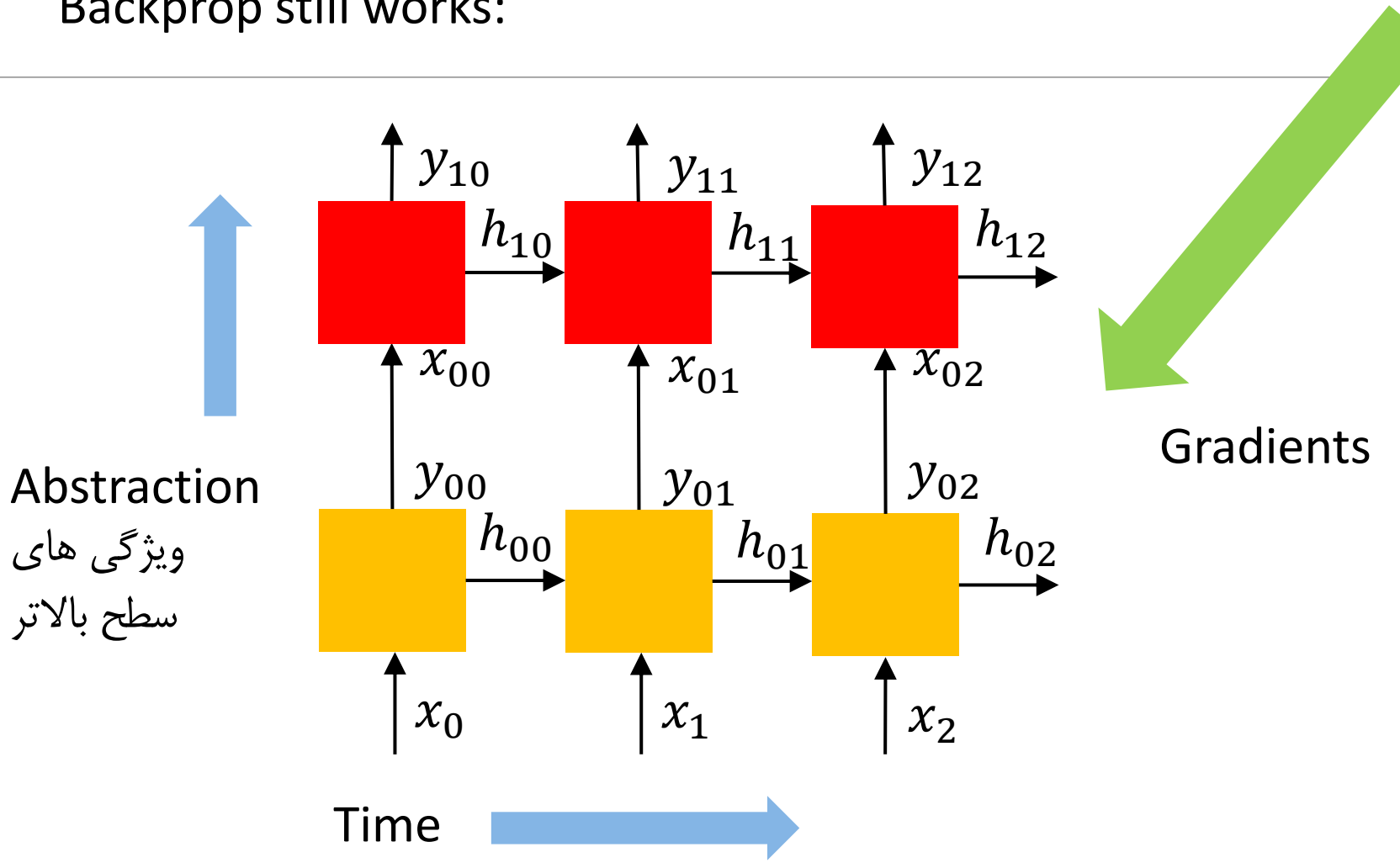
Backprop still works:



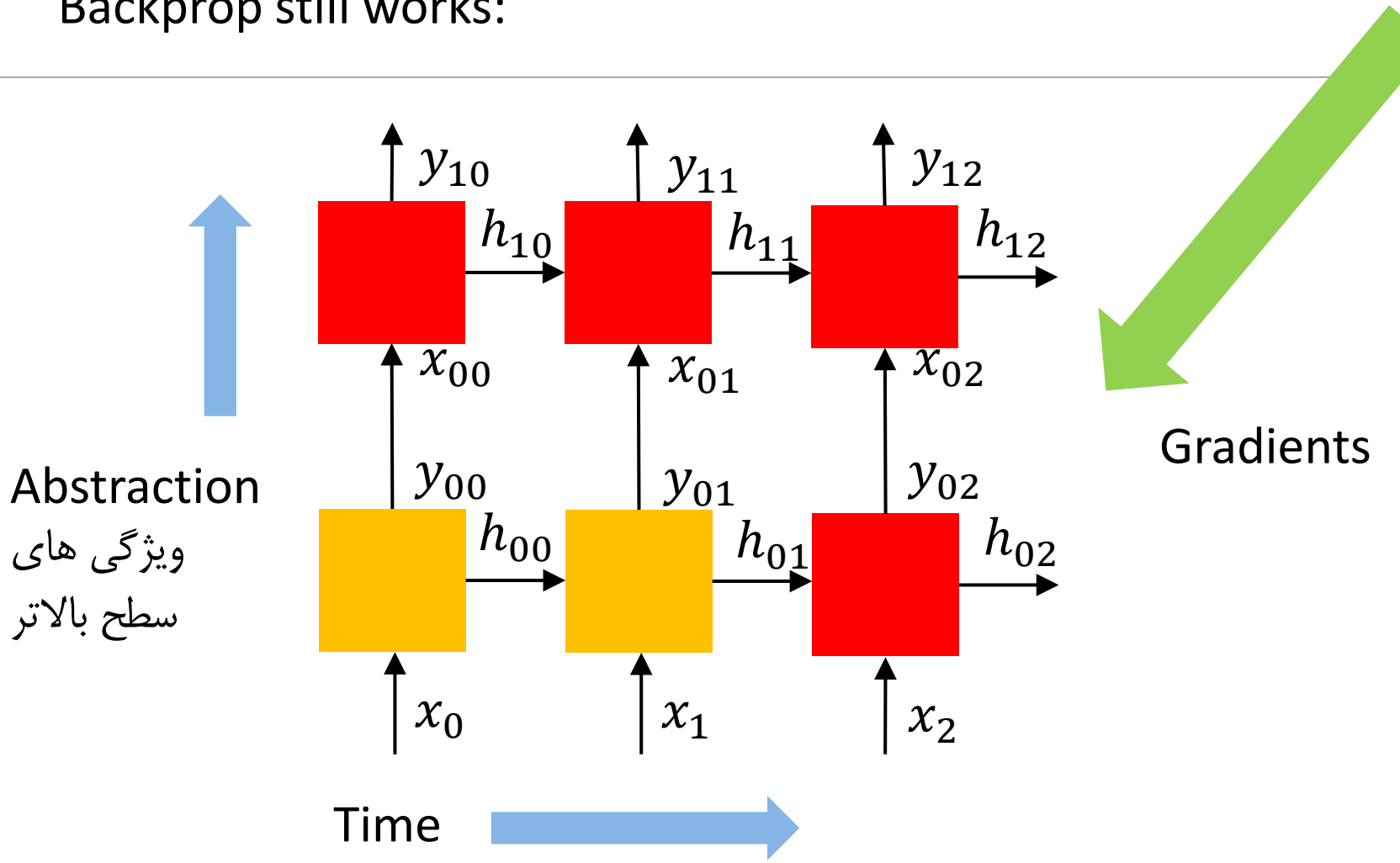
Backprop still works:



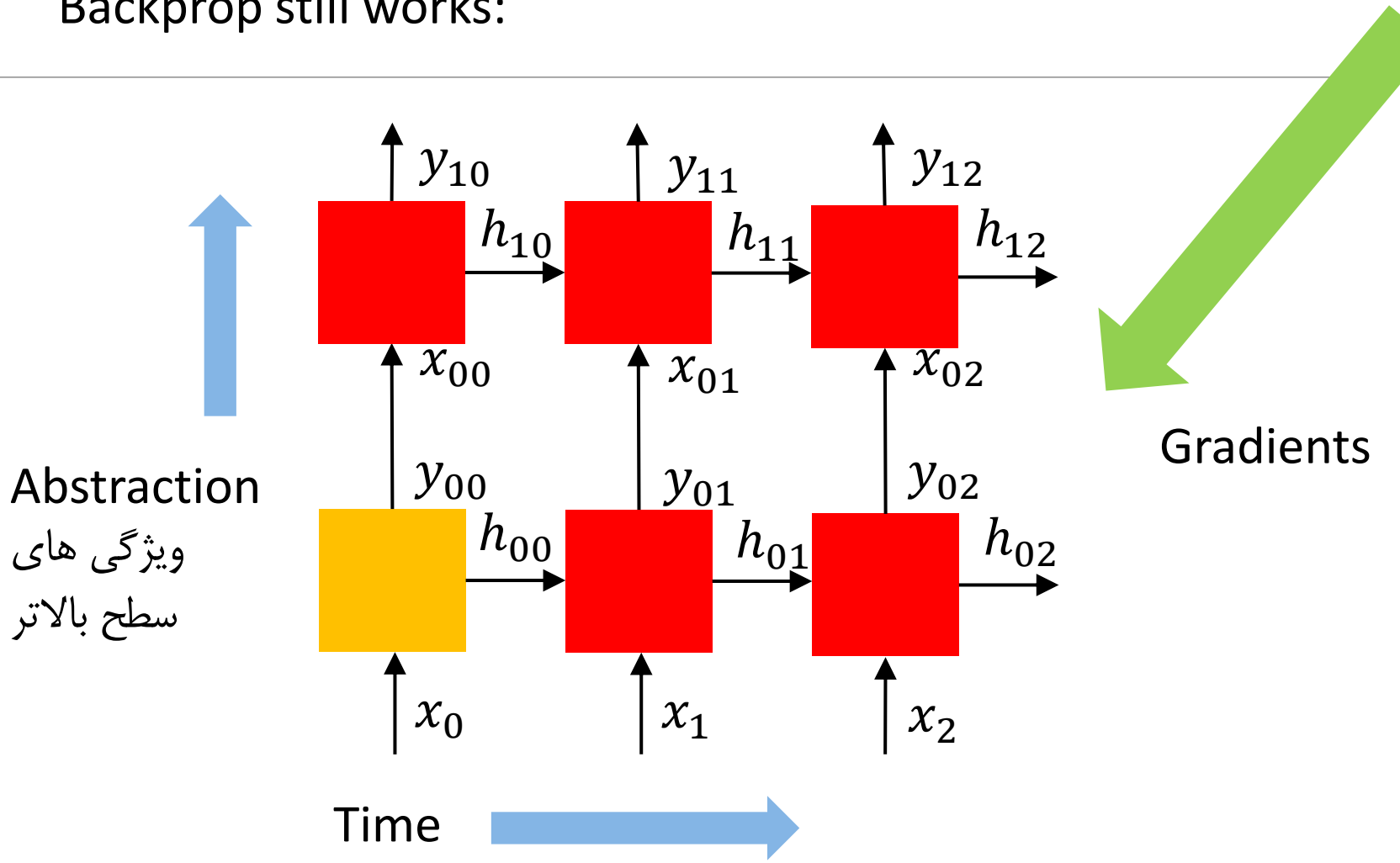
Backprop still works:



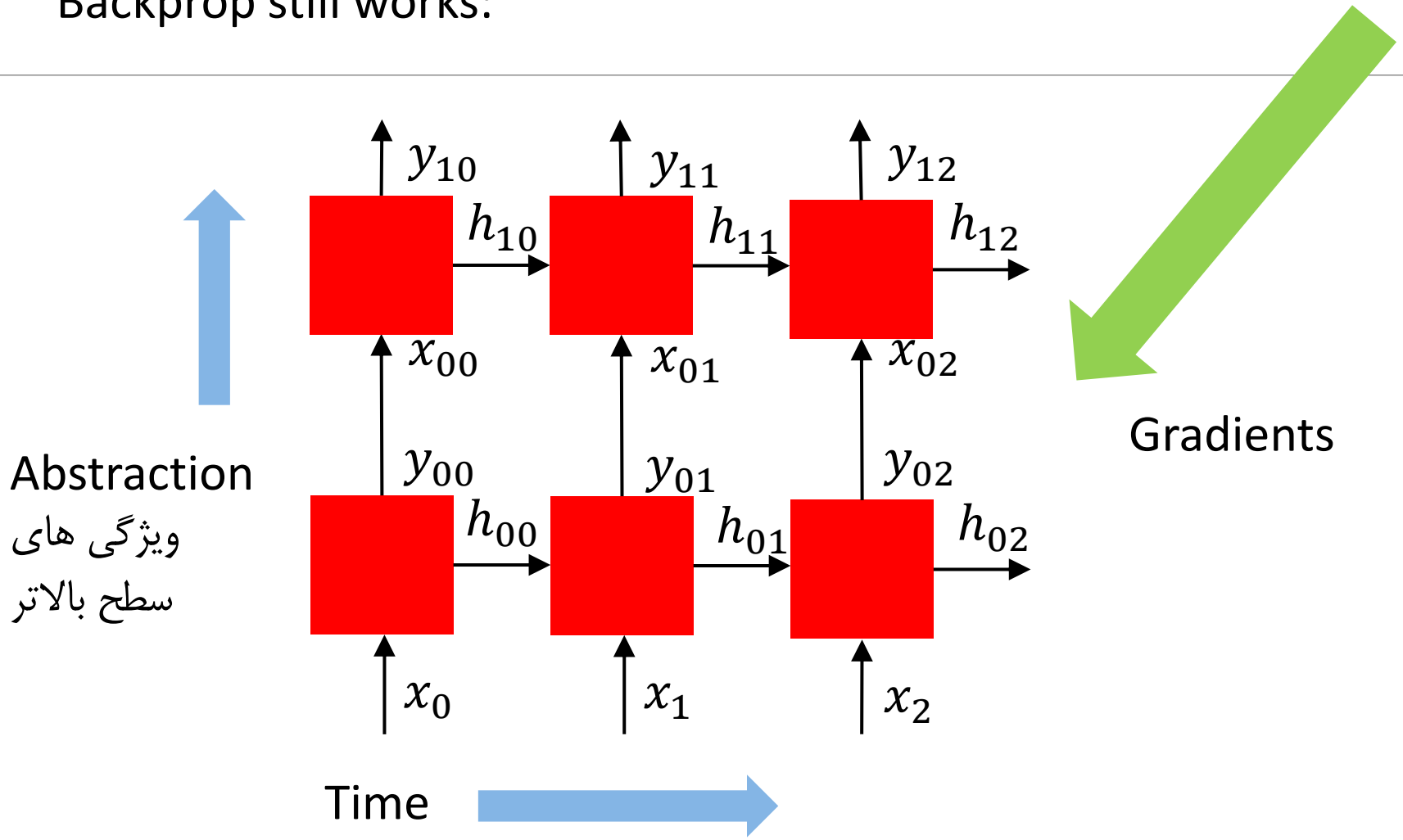
Backprop still works:



Backprop still works:

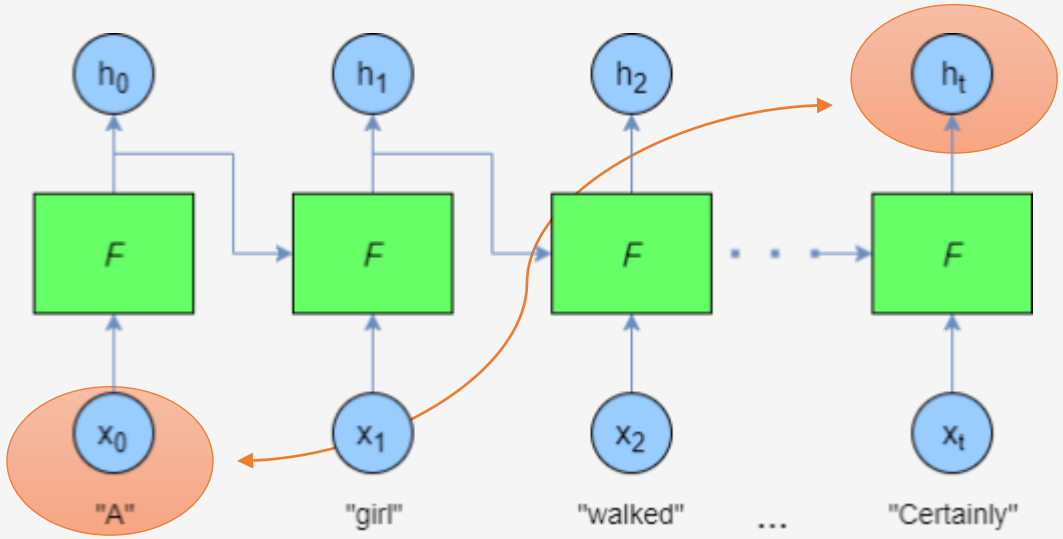


Backprop still works:



مشکلات RNN

- مفهوم و ایده پشت شبکه های بازگشت عالی است.
- مشکل عملیاتی هنگام آموزش وجود دارد.



ناپدید شدن گرادیان

- وابستگی بین h_t و x_0 به صورت زیر قابل تعریف است: $\frac{\partial h_t}{\partial x_0}$
- می تواند به صورت زیر خلاصه شود: $\frac{\partial h_t}{\partial x_0} = \frac{\partial h_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial h_{t-2}} \frac{\partial h_{t-2}}{\partial h_{t-3}} \dots \frac{\partial h_0}{\partial x_0}$

$$h_n = \tanh(\underbrace{w_{hh}h_{n-1} + w_{xh}x_n + b}_{z})$$

$$h_n = \tanh(z)$$

$$\frac{\partial z}{\partial h_{n-1}} = w_{hh}$$

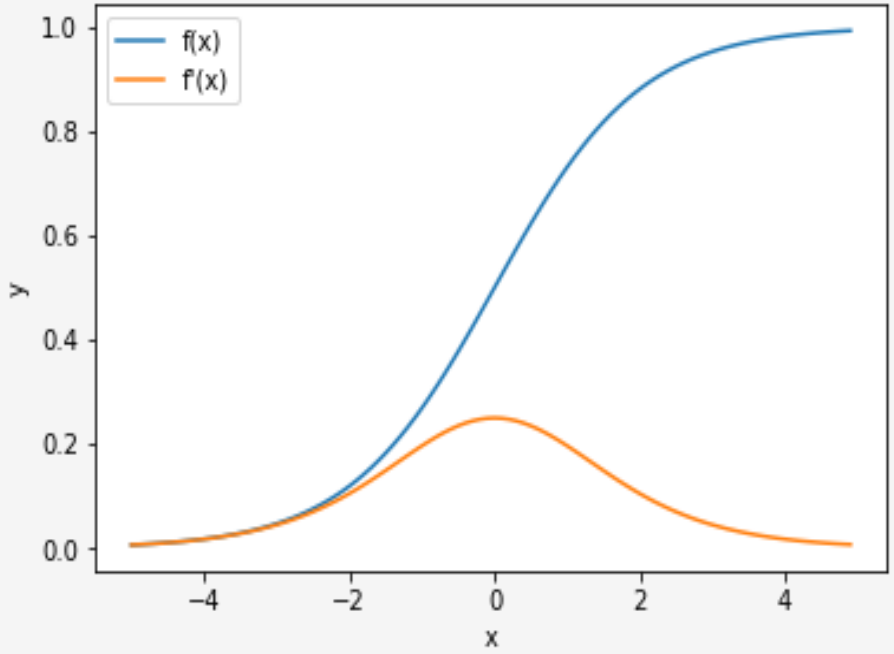
$$\frac{\partial h_n}{\partial z} = 1 - \tanh^2(z)$$

- مشتق tanh عددی بین صفر و یک است.
  - بخش اصلی حاصلضرب مکرر w_{hh} است. اگر بزرگترین مقدار ویژه w_{hh} برابر:
    - ۱، سپس گرادیان منتشر می شود،
    - > ۱، نتیجه به صورت تصاعدی رشد می کند (منفجر می شود)،
    - < ۱، نتیجه به صورت تصاعدی کوچک می شود (ناپدید می شود).

مشکل گرادیان

- نتیجه
  - مشتق جزئی نزدیک به صفر است.
  - خروجی به ورودی های فاصله دور قبلی مربوط نیست.
  - وابستگی های طولانی در حال از بین رفتن است.

نتیجه مشتق با توجه به تابع فعالسازی

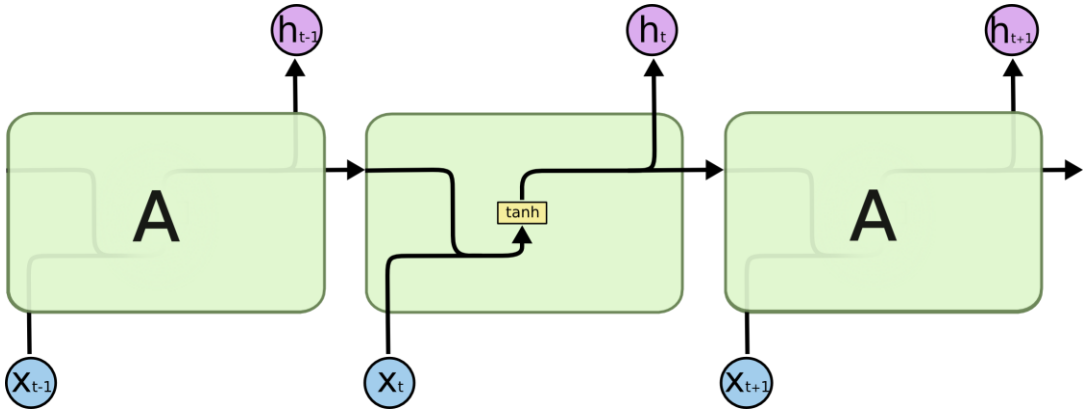


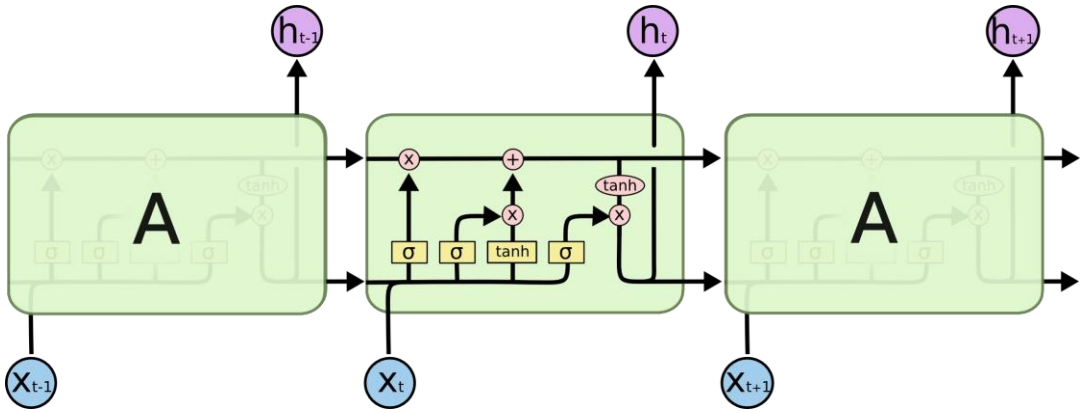
new weight = weight - learning rate*gradient

2.0999 = 2.1 - 0.001

Not much of a difference **update value**

- راه حل
  - تغییر توابع فعال سازی
    - تعداد کمی از توابع فعالسازی گرادیان دقیقا یک دارند:
      - محدودیت زیاد
      - مشکل را می توان تا حدودی کاهش داد اما برطرف نشد.
      - ممکن است بر عملکرد یادگیری تأثیر بگذارد.
      - پیشنهاد تغییر معماری: LSTM





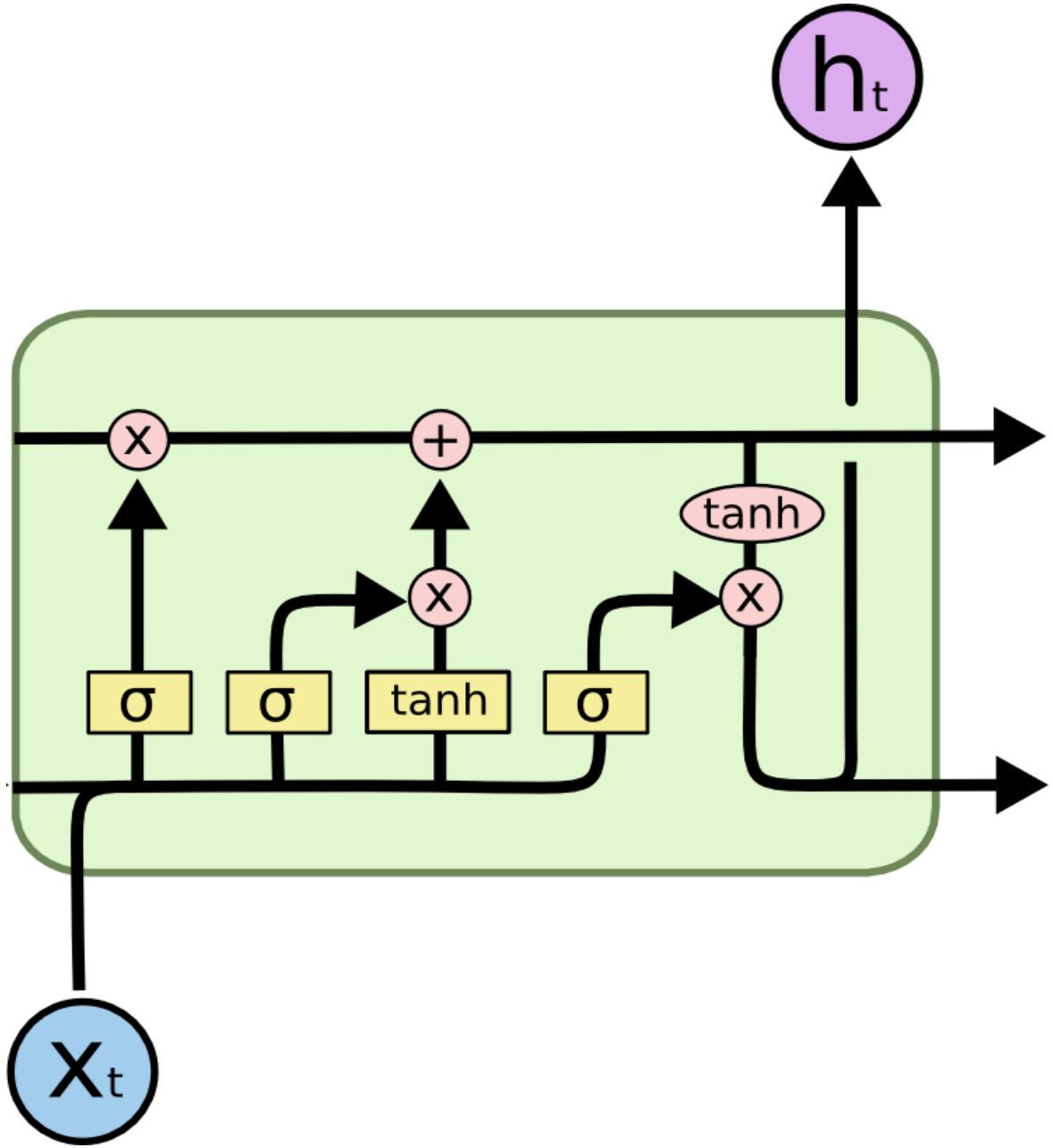
Long Short-Term Memory

ایده اصلی: ایجاد یک مسیر مستقیم از ورودی های قبلی به خروجی فعلی

حفظ اطلاعات مفید و حذف غیرمفیدها

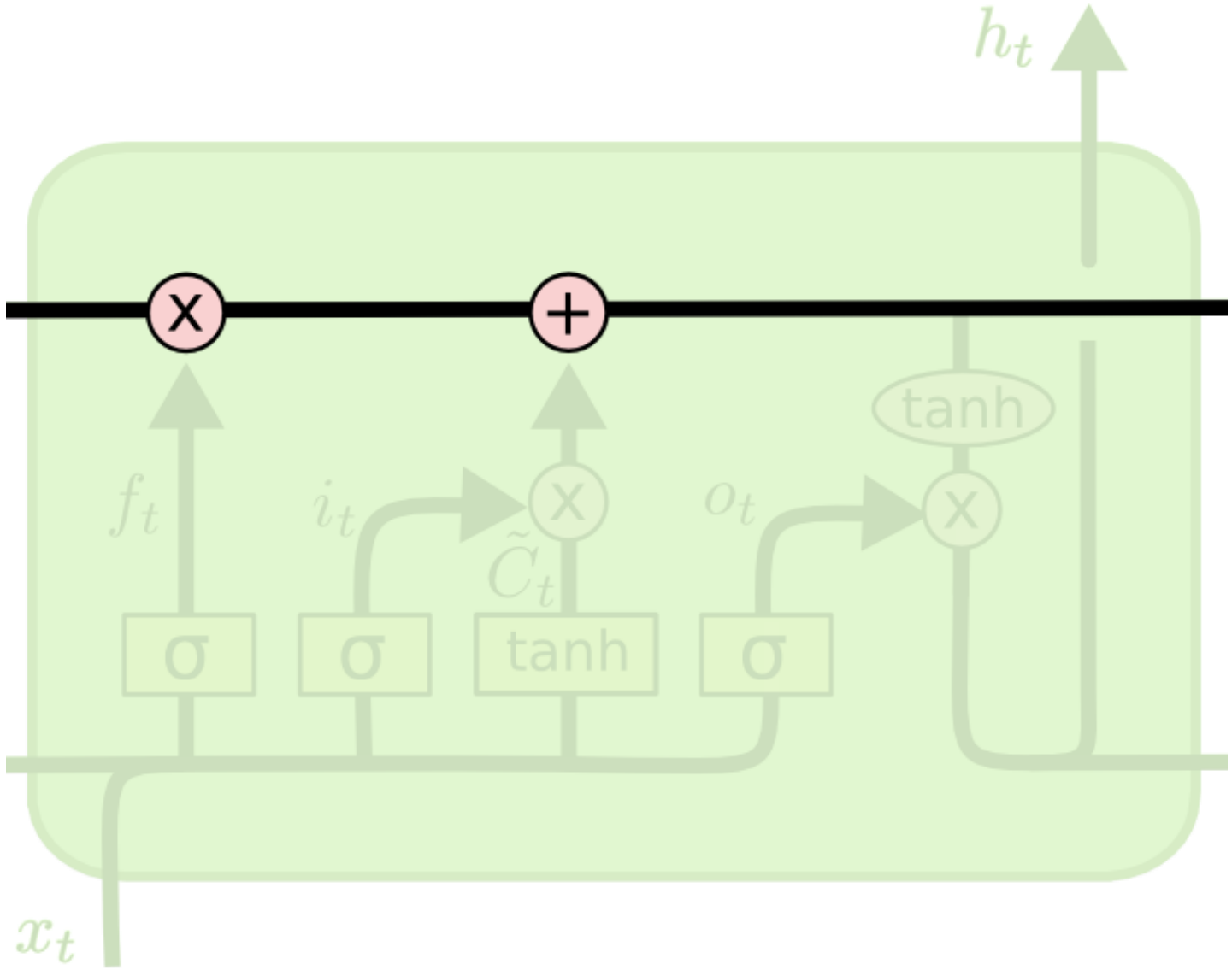
LSTM

- Hidden states
- Input gates
- Forget gates
- Output gates

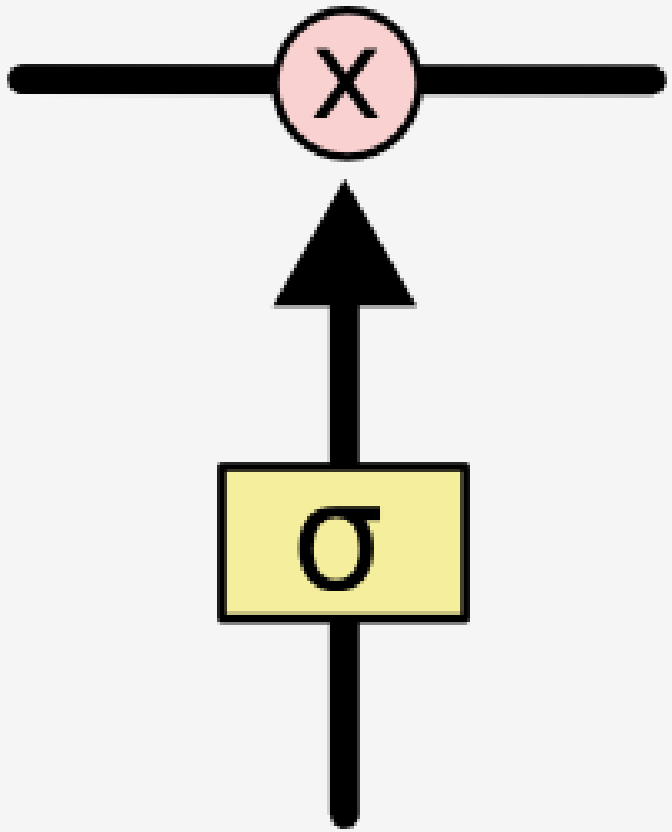


Hidden States

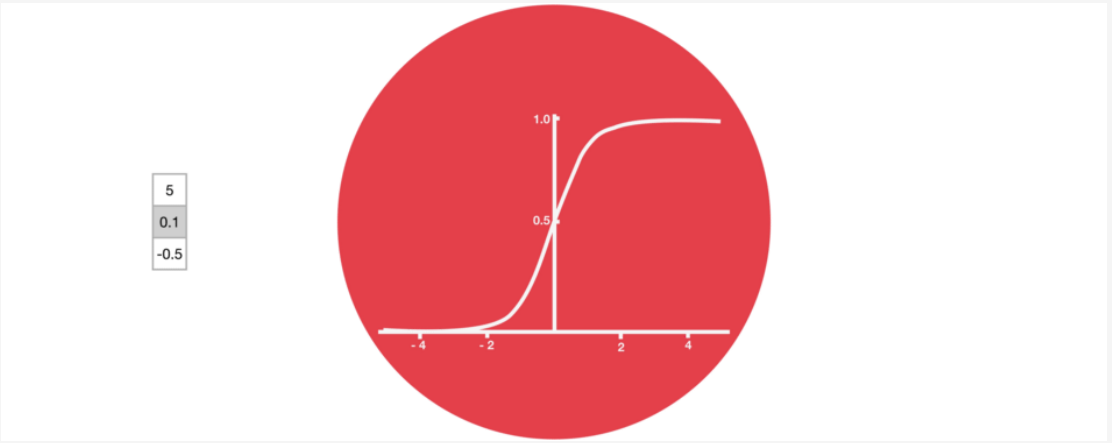
- حمل اطلاعات قبلی
- ساخت مسیر مستقیم



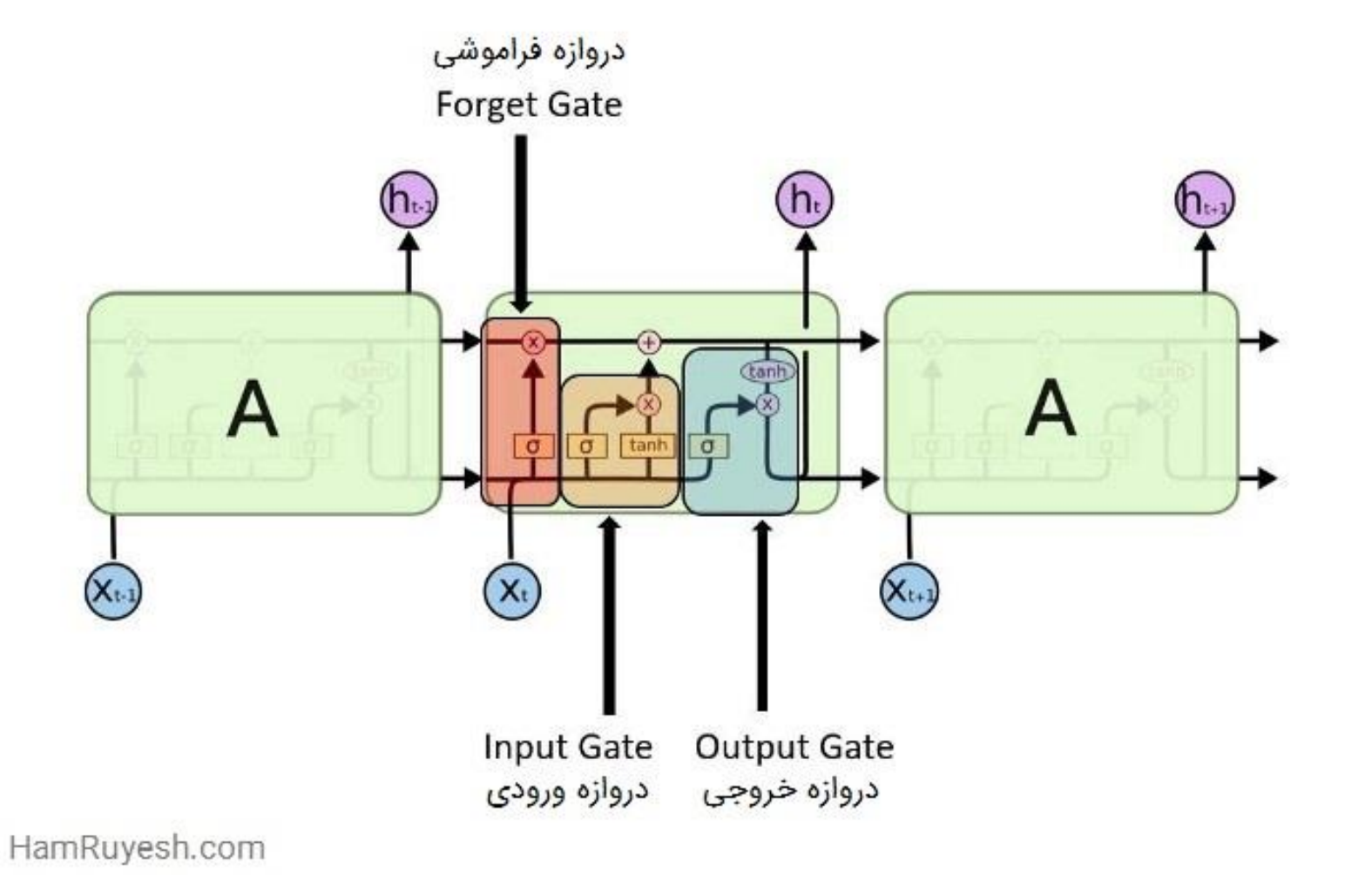
ها Gate

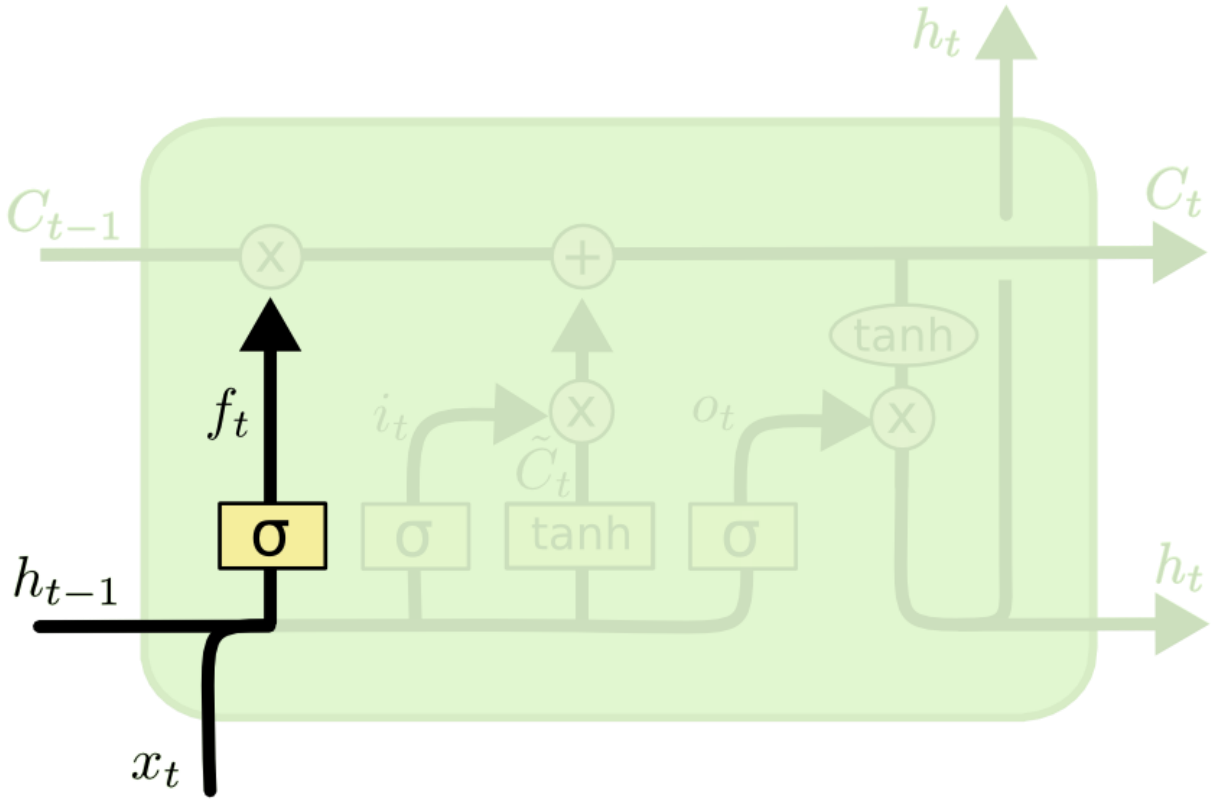


- یک شبکه عصبی سیگموئید
- رنج خروجی [0,1]
- ضرب نقطه ای
- تابع فیلترکننده



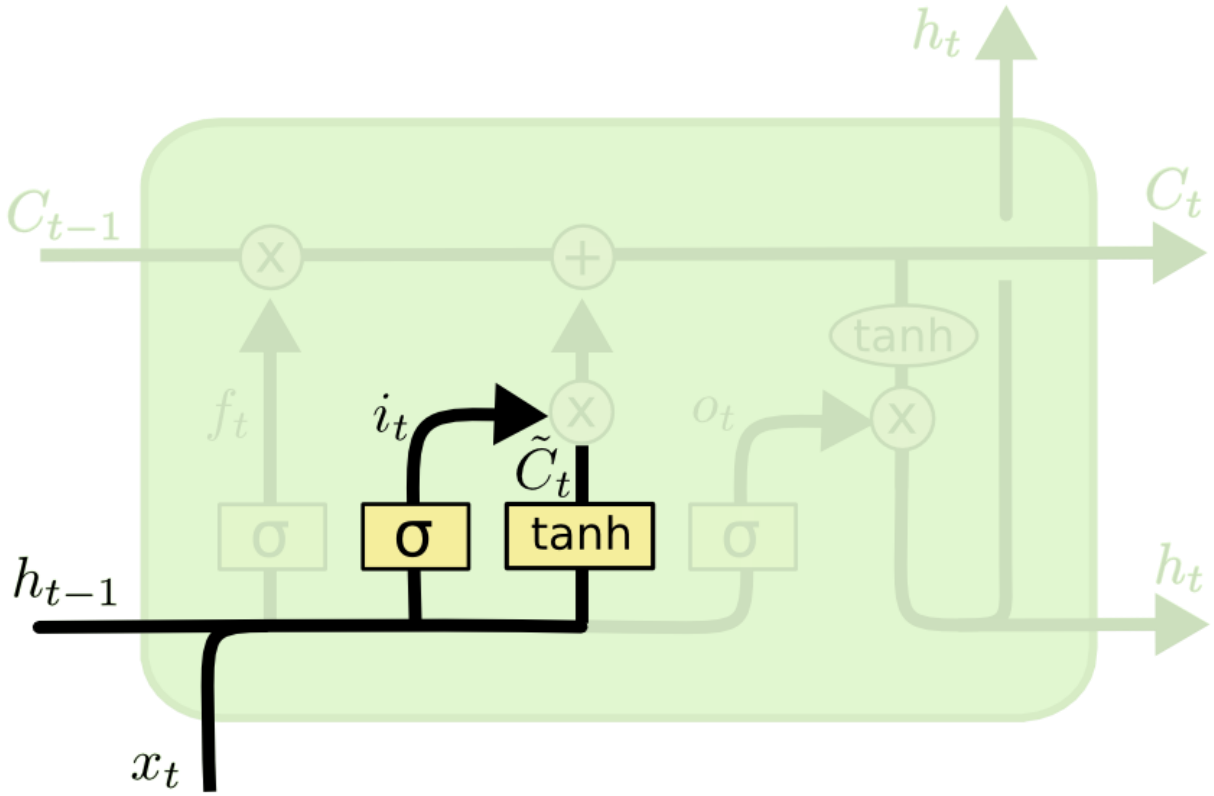
ها Gate


دروازه فراموشی
Forget Gate
A
A
tanh
σ
σ
tanh
σ
Input Gate
دروازه ورودی
Output Gate
دروازه خروجی
HamRuyesh.com



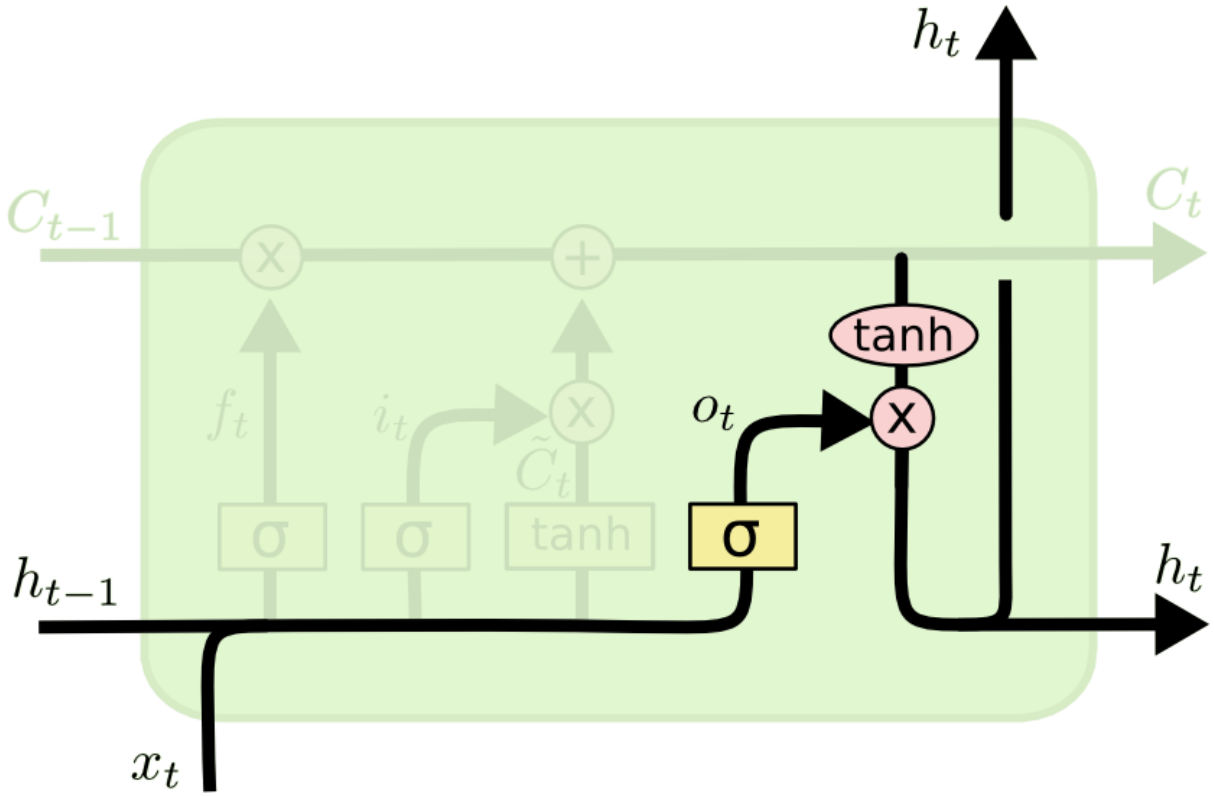
Input Gates

چیزهایی که از حالت های قبلی حفظ شده



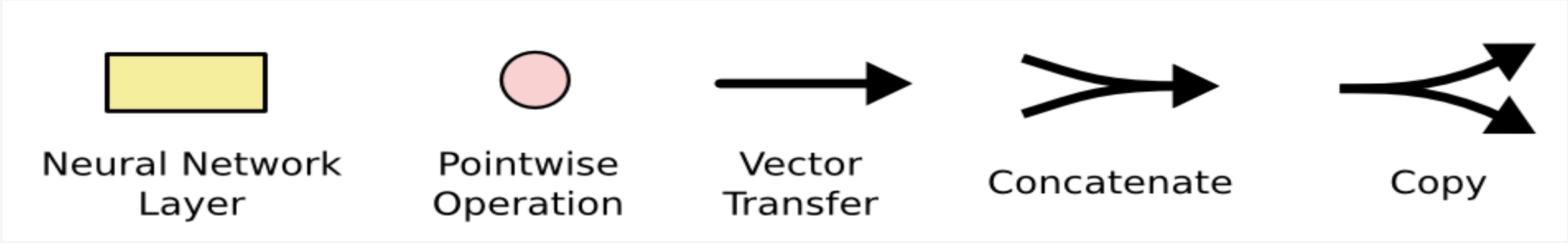
Forget Gate

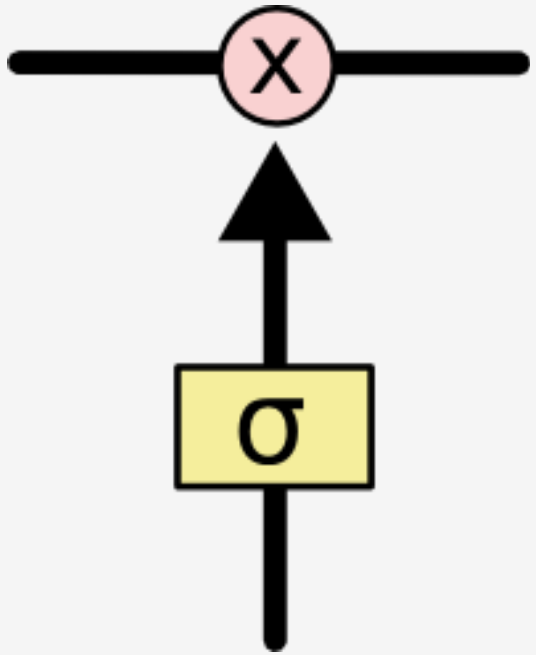
چیزی که به لایه هیدن اضافه می شود.



Output Gates

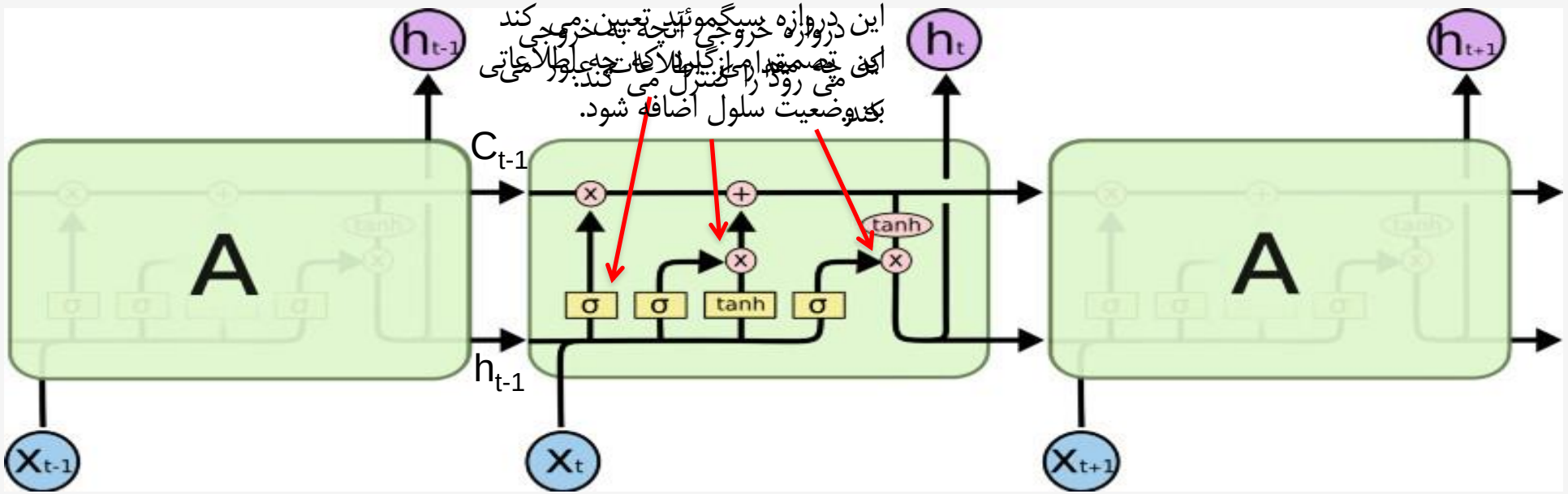
چیزی که به عنوان خروجی آماده می شود.



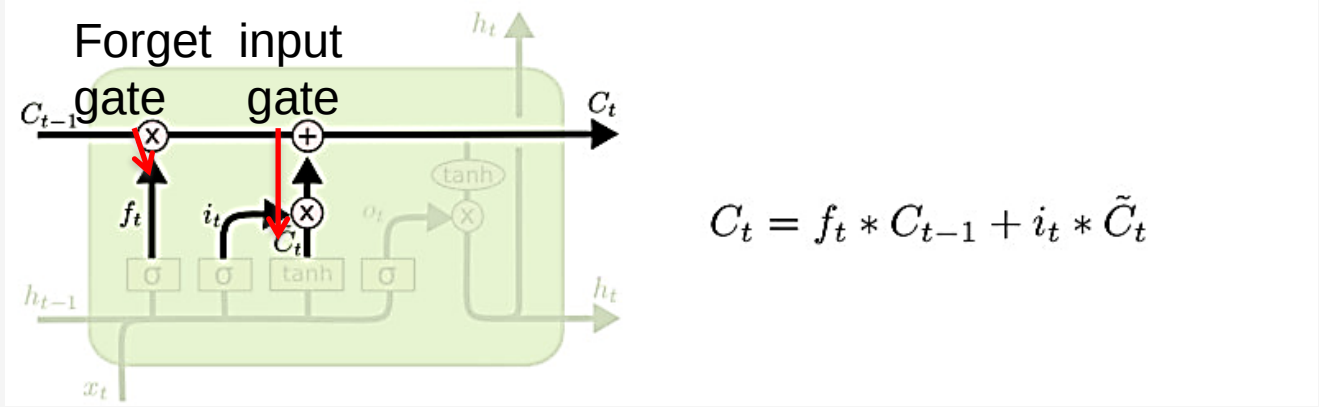


خروجی لایه سیگموئید با اعداد بین ۰–۱ تعیین می کند که هر جزء چقدر باید عبور کند. گیت X صورتی ضرب نقطه ای است.

LSTM

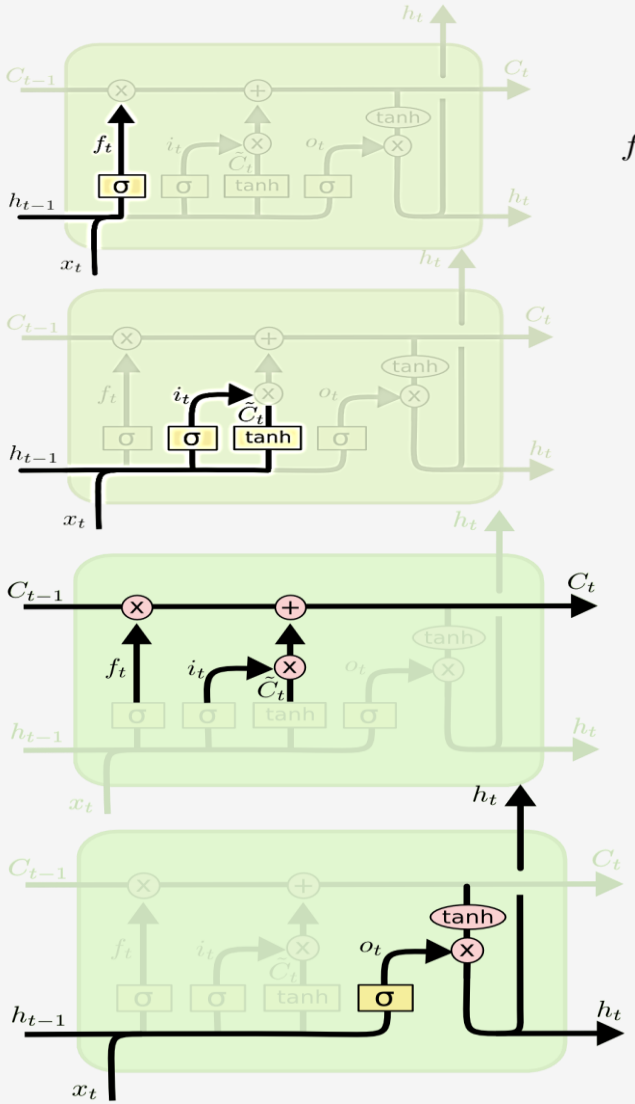


ایده اصلی این حالت سلولی C_t است، که به آرامی تغییر می کند، تنها با برهمکنش های خطی جزئی.
جریان اطلاعات بدون تغییر در امتداد آن بسیار آسان است.



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

LSTM

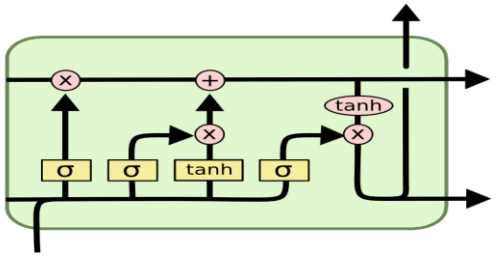


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

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

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

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

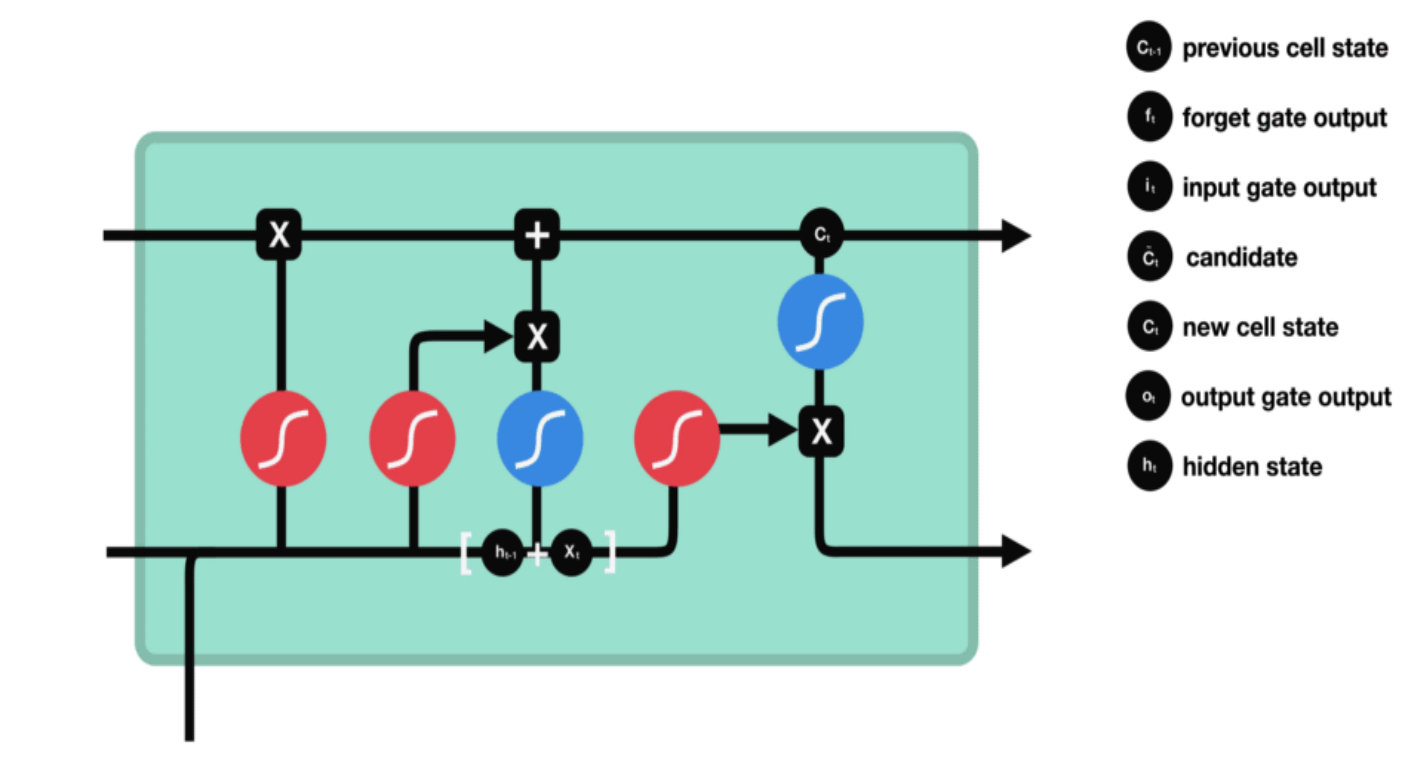


i_t تصمیم می گیرد چه مولفه ای به روز شود.
C'_t تغییر را اعمال می کند.

به روزرسانی cell state

o_t تصمیم می گیرد چه چیزی به خروجی برود.

LSTM به صورت متحرک



LSTM

Candidate input:

$$a_t = \tanh(W_a . x_t + U_a . h_{t-1} + b_a)$$

Input gate:

$$i_t = \sigma(W_i . x_t + U_i . h_{t-1} + b_i)$$

Forget gate:

$$f_t = \sigma(W_f . x_t + U_f . h_{t-1} + b_f)$$

Output gate:

$$o_t = \sigma(W_o . x_t + U_o . h_{t-1} + b_o)$$

Cell state:

$$C_t = a_t \odot i_t + f_t \odot C_{t-1}$$

Hidden state:

$$h_t = tan\, h(C_t) \odot o_t$$

مثال :LSTM

Internal weights

$W_a = \begin{matrix}0.45\\0.25\end{matrix}$ $U_a = 0.15$ $b_a = 0.2$

$W_i = \begin{matrix}0.95\\0.8\end{matrix}$ $U_i = 0.8$ $b_i = 0.65$

$W_f = \begin{matrix}0.7\\0.45\end{matrix}$ $U_f = 0.1$ $b_f = 0.15$

$W_o = \begin{matrix}0.6\\0.4\end{matrix}$ $U_o = 0.25$ $b_o = 0.1$

Input data

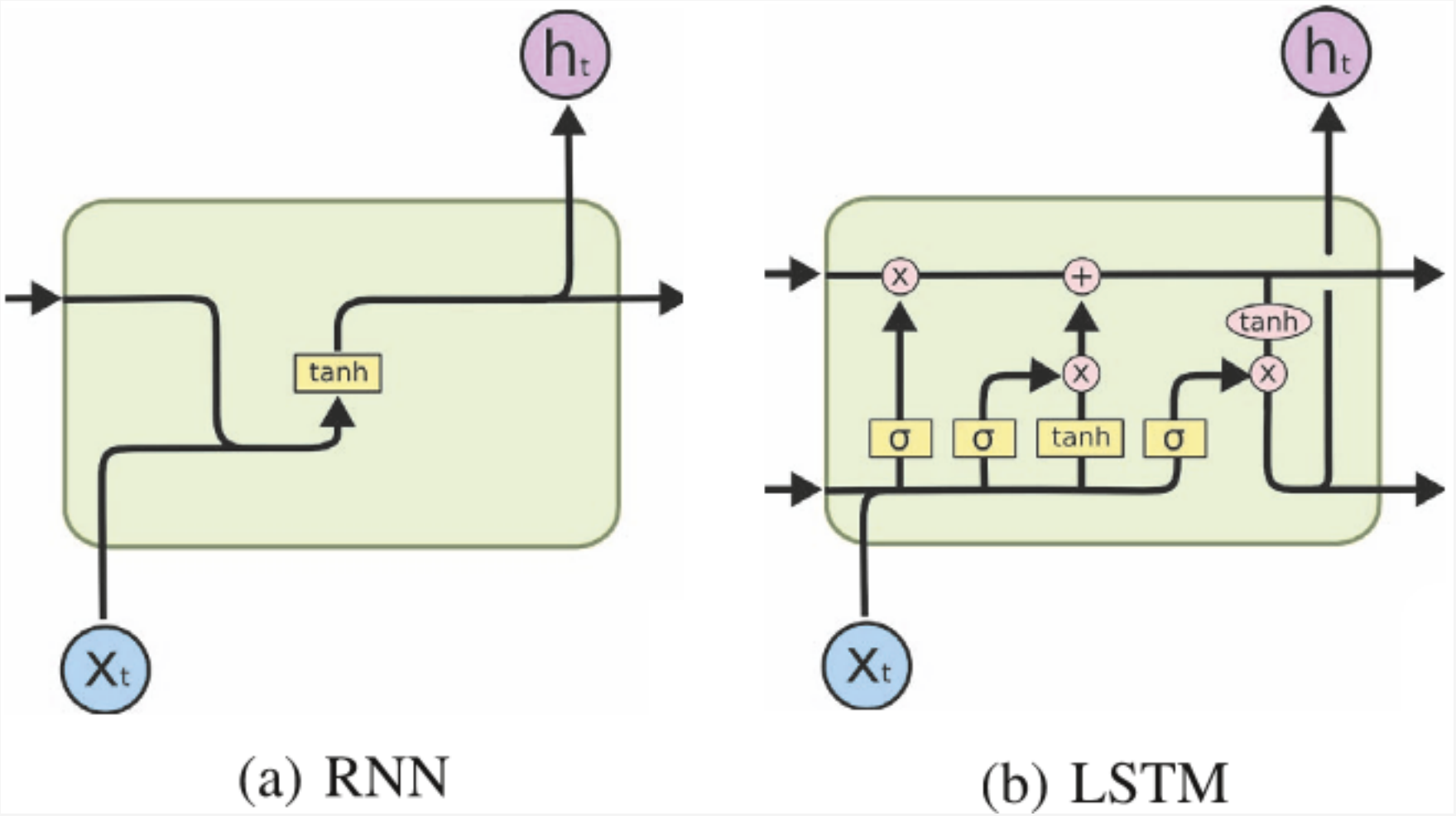
$x_0 = \begin{matrix}1\\2\end{matrix}$ With label: 0.5

$x_1 = \begin{matrix}0.5\\3\end{matrix}$ With label: 1.25

Forget gate:

$$f_t = \sigma(W_f . x_t + U_f . h_{t-1} + b_f) = \sigma\left(\begin{matrix}0.7 & 0.45\end{matrix}\begin{matrix}1\\2\end{matrix} + 0.1 \cdot 0 + 0.15\right) = 0.85195$$

RNN و LSTM

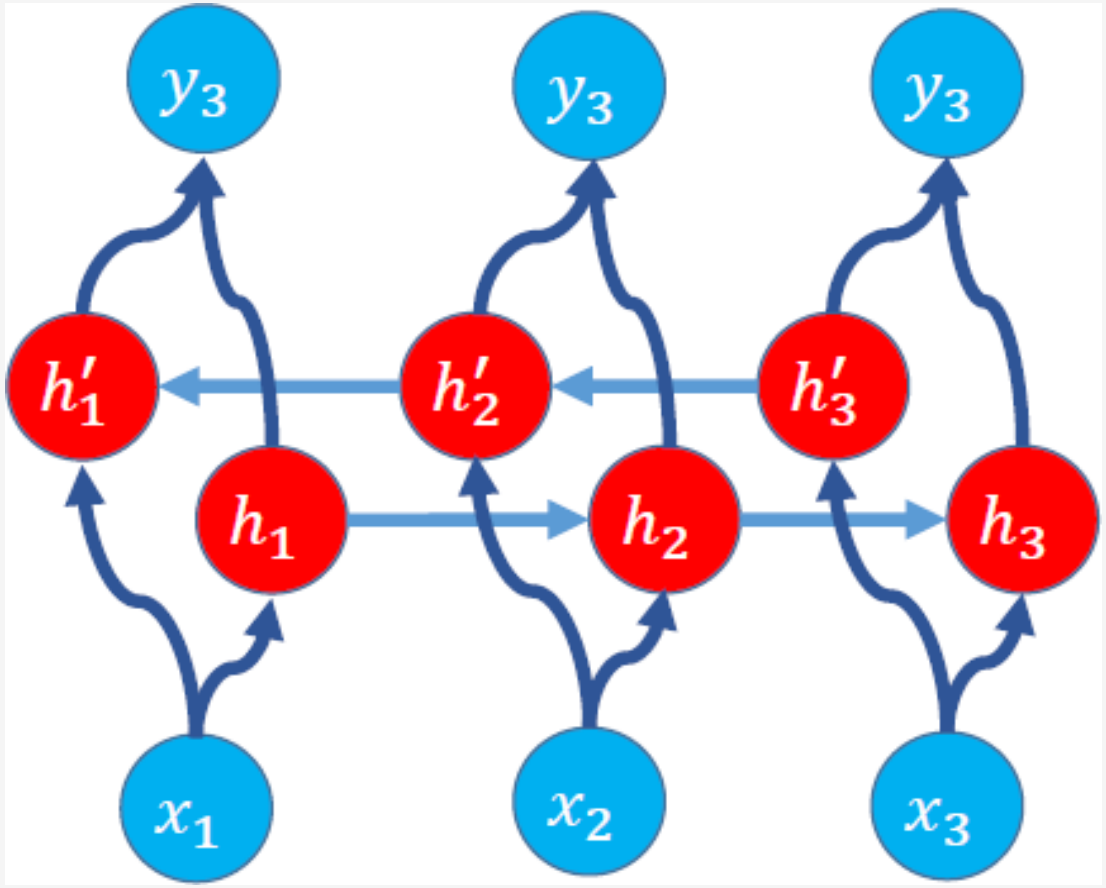


نتیجه

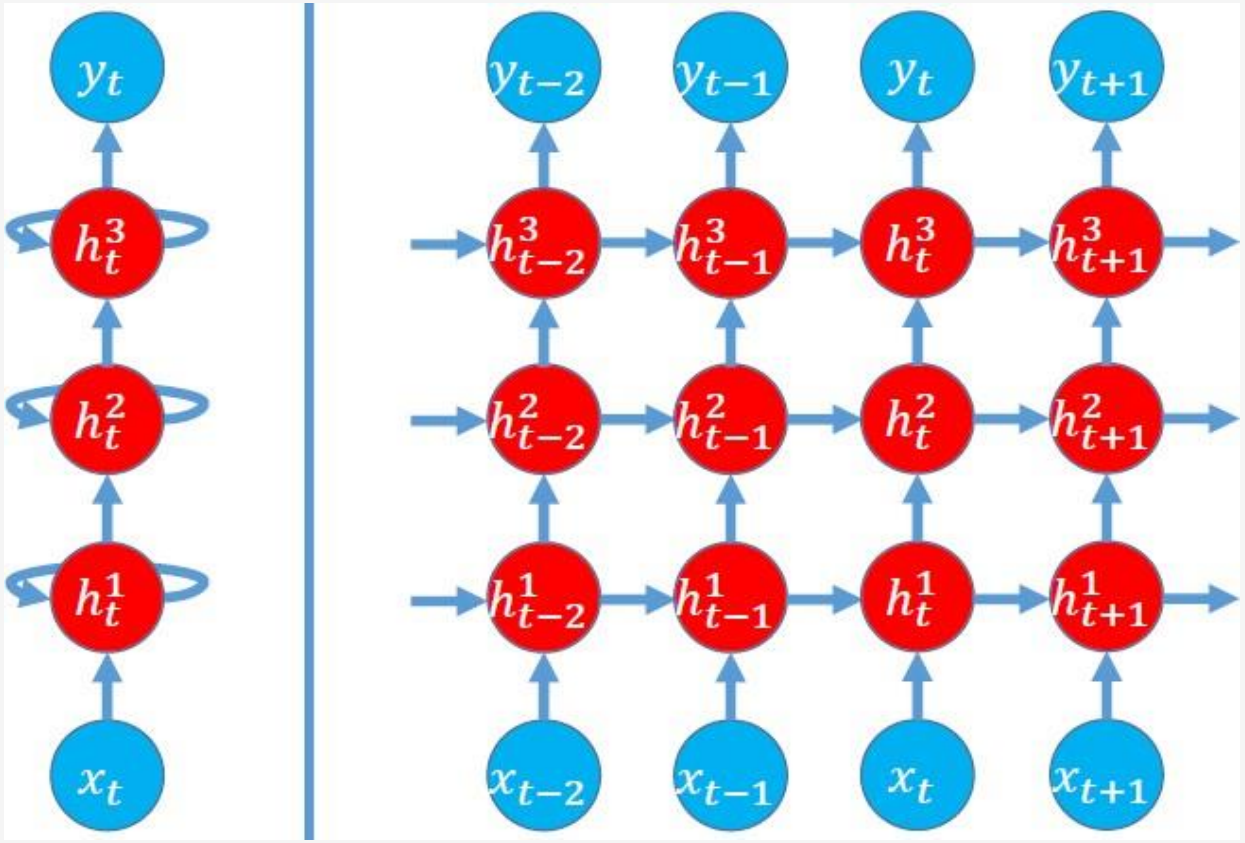
1. RNN برای مسئله توالی / سری زمانی استفاده می شود.
2. مشکل ناپدید شدن گرادیان اثربخشی آن را محدود می کند.
3. معرفی سلول LSTM برای از بین بردن مشکل
4. عملا LSTM بسیار بهتر از RNN کار می کند.

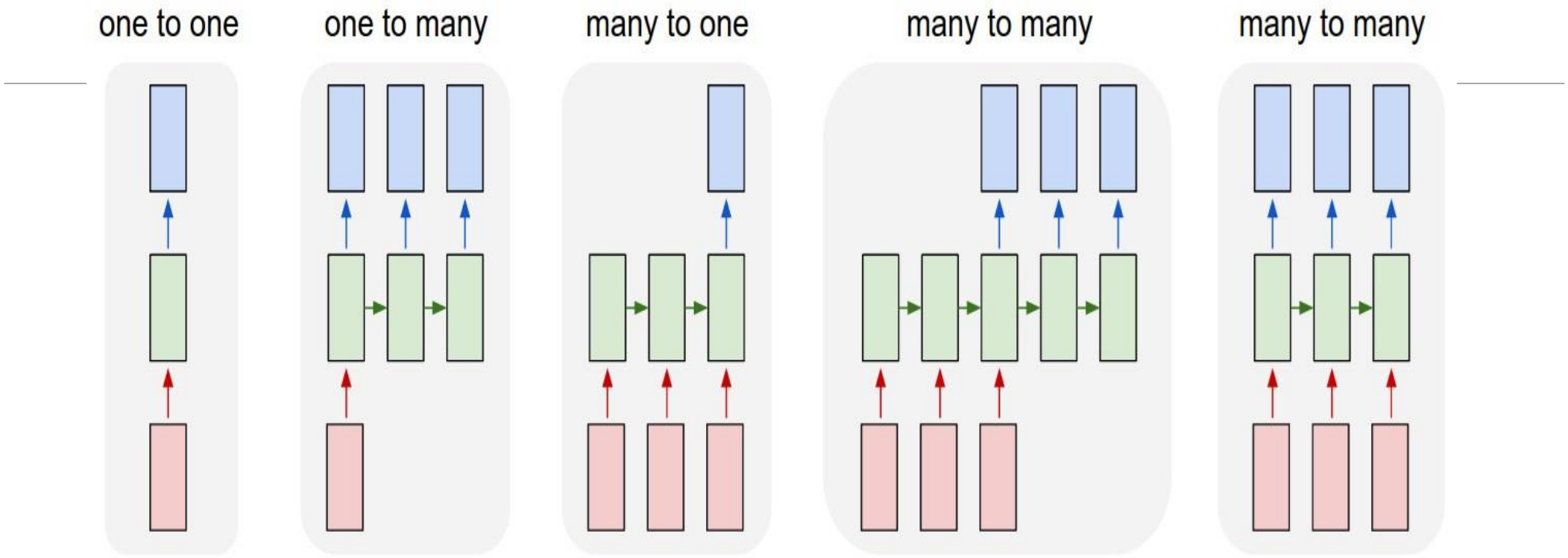
RNN دوطرفه

$$y = \text{softmax}(w[h_{t-1}, h'_{t-1}] + b)$$

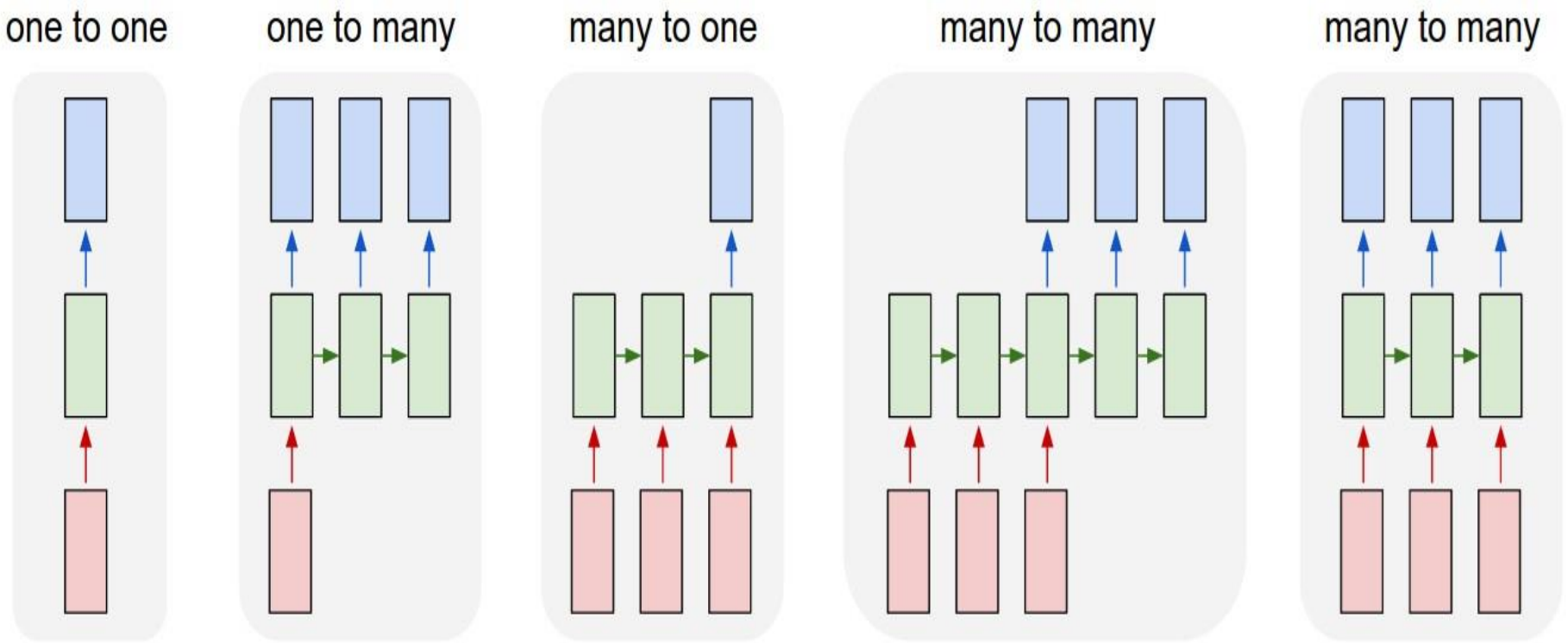


RNN عميق

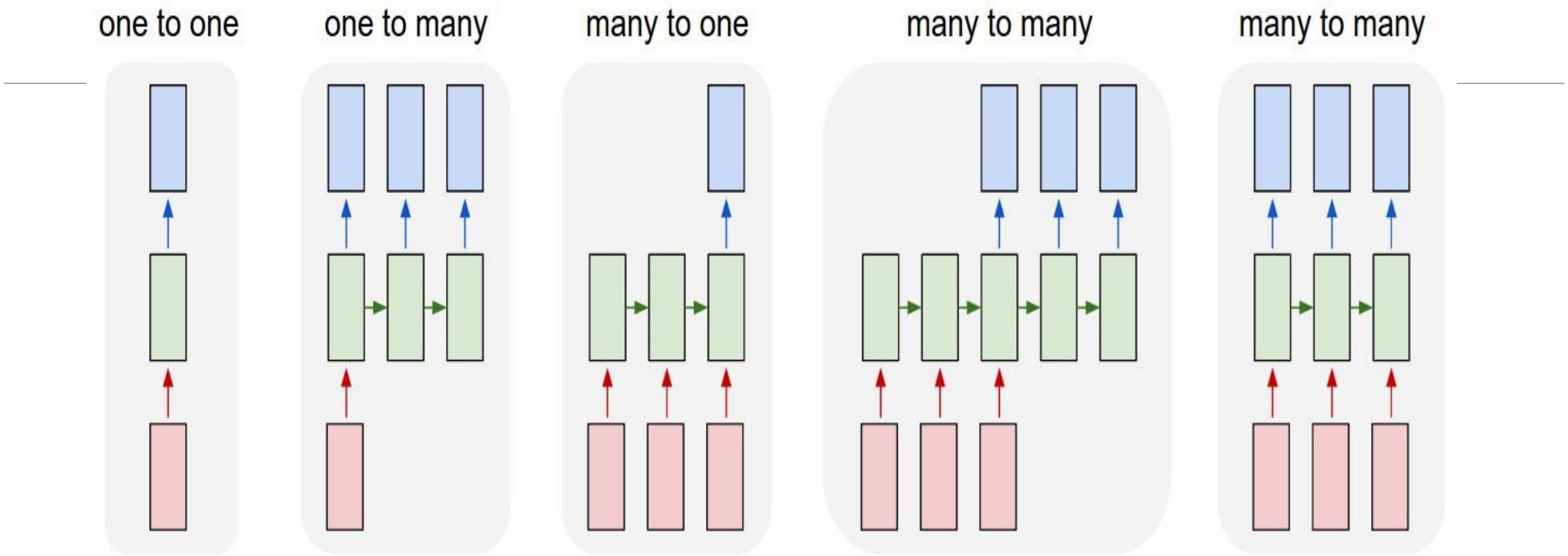

y_t
h_t^3
h_t^2
h_t^1
x_t
y_{t-2}
y_{t-1}
y_t
y_{t+1}
h_{t-2}^3
h_{t-1}^3
h_t^3
h_{t+1}^3
h_{t-2}^2
h_{t-1}^2
h_t^2
h_{t+1}^2
h_{t-2}^1
h_{t-1}^1
h_t^1
h_{t+1}^1
x_{t-2}
x_{t-1}
x_t
x_{t+1}



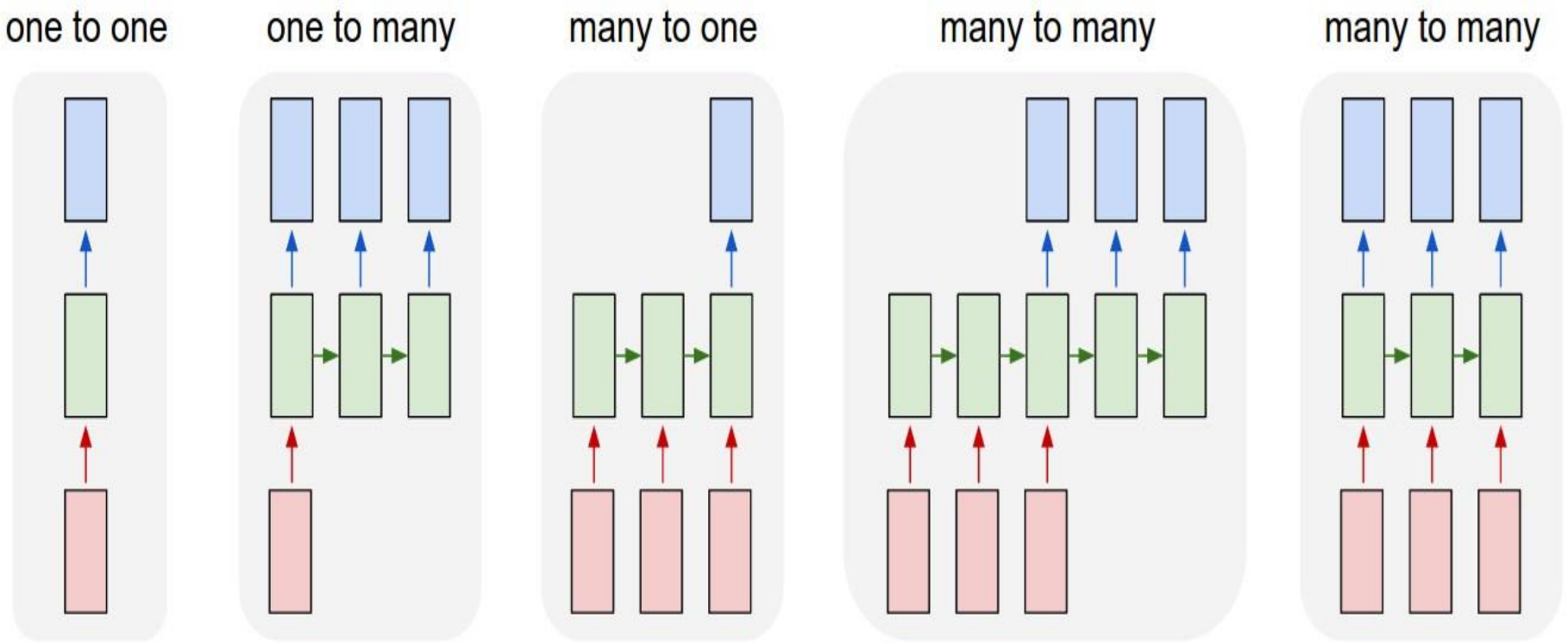
Vanilla Neural Networks



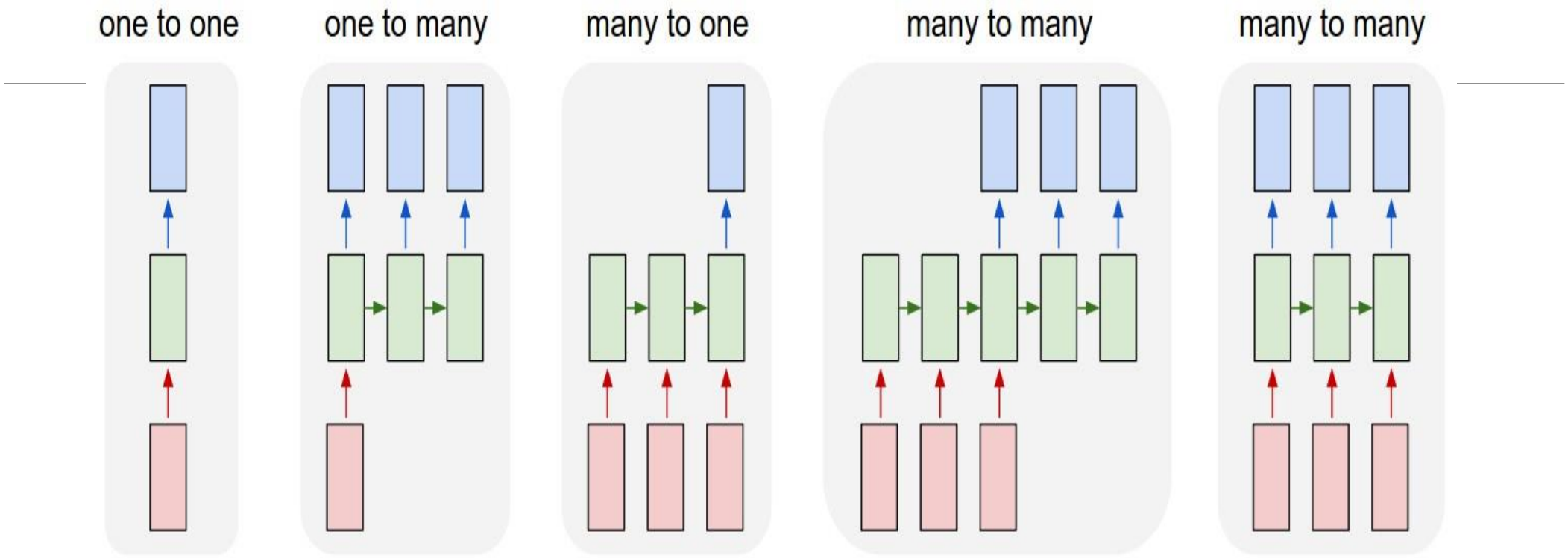
e.g. **Image Captioning**
image -> sequence of words



e.g. **Sentiment Classification**
sequence of words -> sentiment



e.g. **Machine Translation**
seq of words -> seq of words



e.g. **Video classification on frame level**