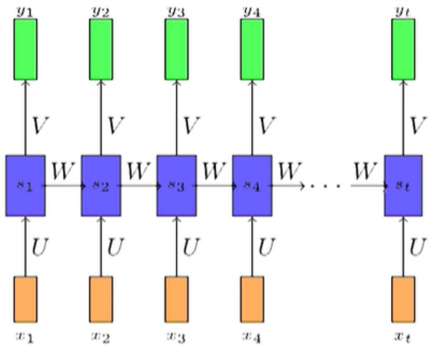
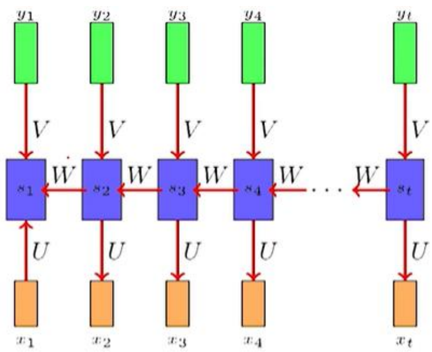
Motivation for the need of LSTMs and GRUs

Selective Read, Selective Write, Selective Forget - The Whiteboard Analogy



- The state (s_i) of an RNN records information from all previous time steps
- At each new timestep the old information gets morphed by the current input
- One could imagine that after t steps the information stored at time step $t-k$ (for some $k < t$) gets completely morphed

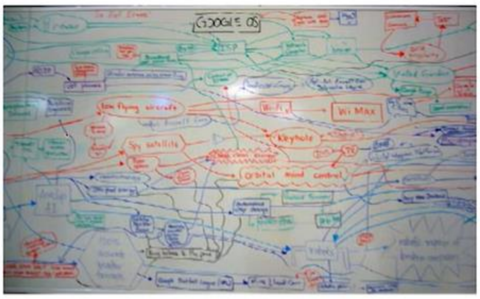
  so much that it would be impossible to extract the original information stored at time step $t-k$



- A similar problem occurs when the information flows backwards (backpropagation)
- It is very hard to assign the responsibility of the error caused at time step t to the events that occurred at time step $t-k$
- This responsibility is of course in the form of gradients and we studied the problem in backward flow of gradients
- We saw a formal argument for this while discussing vanishing gradients

- Let us see an analogy for this
- We can think of the state as a fixed size memory
- Compare this to a fixed size white board that you use to record information
- At each time step (periodic intervals) we keep writing something to the board
- Effectively at each time step we morph the information recorded till that time point
- After many timesteps it would be impossible to see how the information at time step $t-k$ contributed to the state at timestep t

4:16 / 9:17



- Continuing our whiteboard analogy, suppose we are interested in deriving an expression on the whiteboard
- We follow the following strategy at each time step
- Selectively write on the board
- Selectively read the already written content
- Selectively forget (erase) some content

$a = 1 \quad b = 3 \quad c = 5 \quad d = 11$

Compute $ac(bd + a) + ad$

Say "board" can have only 3 statements at a time.

1. ac
2. bd
3. $bd + a$
4. $ac(bd + a)$
5. ad
6. $ac(bd + a) + ad$

$a=1$

$c=5$

$\cancel{ac = 5}$ $\quad a \times c = 5$

$bd = 33$ $\quad b =$

$b \times d = 33$

Selective write

- There may be many steps in the derivation but we may just skip a few
- In other words we select what to **write**

$a = 1 \quad b = 3 \quad c = 5 \quad d = 11$

Compute $ac(bd + a) + ad$

Say "board" can have only 3 statements at a time.

1. ac
2. bd
3. $bd + a$
4. $ac(bd + a)$
5. ad
6. $ac(bd + a) + ad$

$ac = 5$

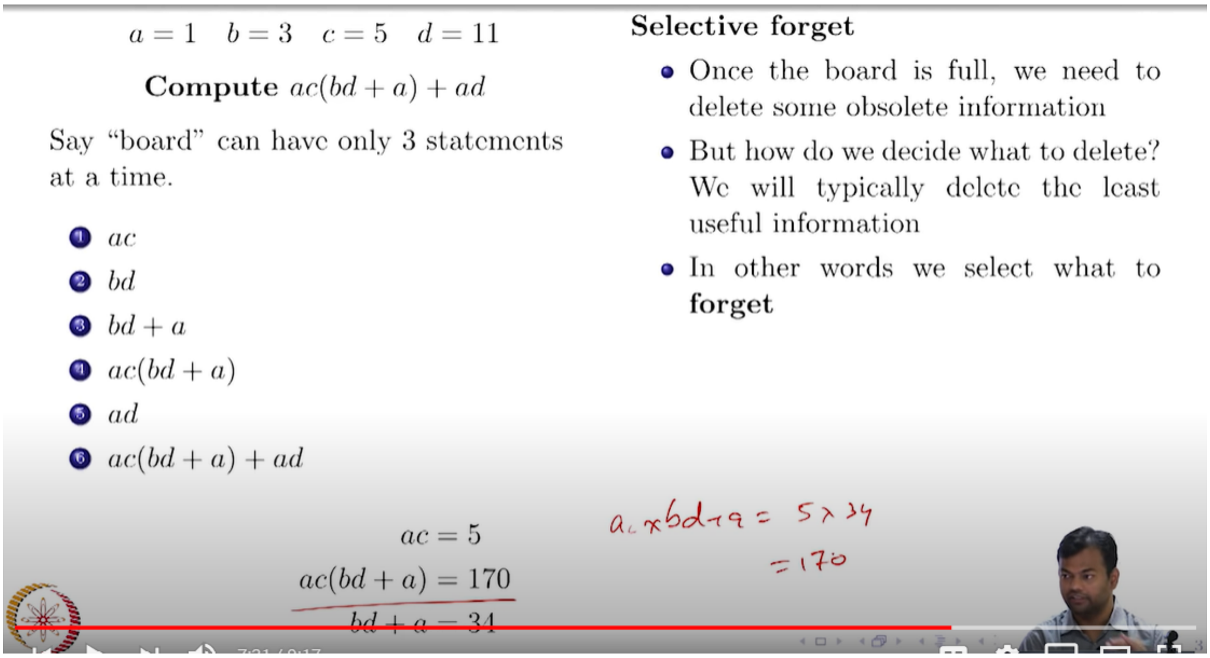
$bd = 33$

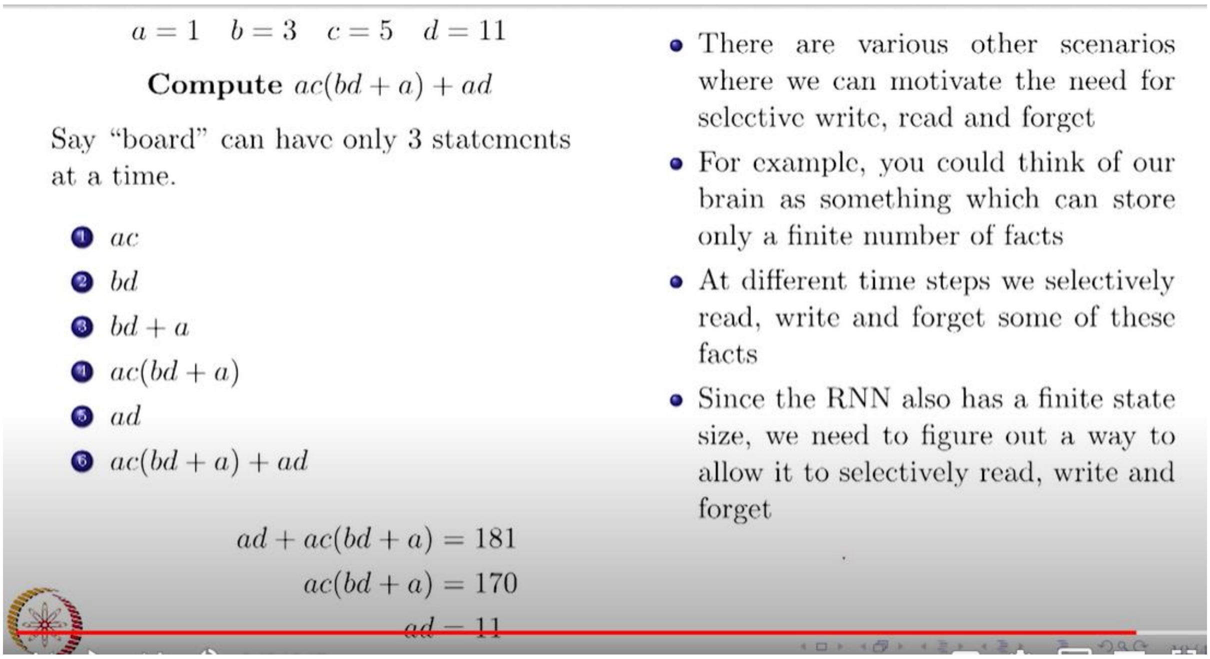
$bd + a = 34$

6:35 / 9:17

Selective read

- While writing one step we typically read some of the previous steps we have already written and then decide what to write next
- For example at Step 3, information from Step 2 is important
- In other words we select what to **read**



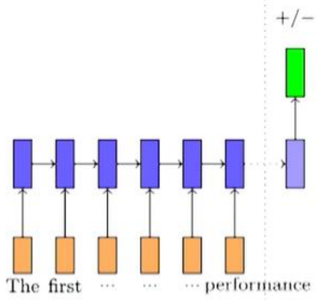


LSTMs and GRUs

Long Short Term Memory (LSTM) and Gated Recurrent Units (GRUs)

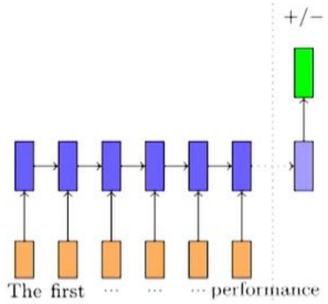
Questions

- Can we give a concrete example where RNNs also need to selectively read, write and forget ?
- How do we convert this intuition into mathematical equations ? We will see this over the next few slides



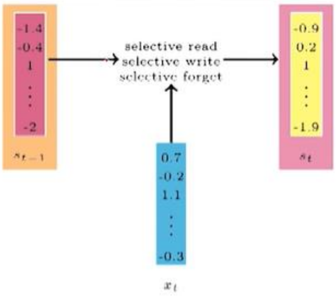
Review: The first half of the movie was dry but the second half really picked up pace. The lead actor delivered an amazing performance

- Consider the task of predicting the sentiment (positive/negative) of a review
- RNN reads the document from left to right and after every word updates the state
- By the time we reach the end of the document the information obtained from the first few words is completely lost
- Ideally we want to
  - **forget** the information added by stop words (a, the, etc.)
  - **selectively read** the information added by previous sentiment bearing words (awesome, amazing, etc.)
  - **selectively write** new information from the current word to the state

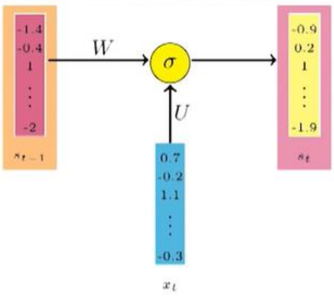


Review: The first half of the movie was dry but the second half really picked up pace. The lead actor delivered an amazing performance

- Recall that the blue colored vector (s_t) is called the state of the RNN
- It has a finite size ($s_t \in \mathbb{R}^n$) and is used to store all the information upto timestep t
- This state is analogous to the whiteboard and sooner or later it will get overloaded and the information from the initial states will get morphed beyond recognition
- **Wishlist:** selective write, selective read and selective forget to ensure that this finite sized state vector is used effectively

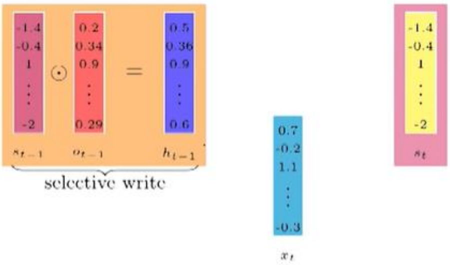


- Just to be clear, we have computed a state s_{t-1} at timestep $t-1$ and now we want to overload it with new information (x_t) and compute a new state (s_t)
- While doing so we want to make sure that we use selective write, selective read and selective forget so that only important information is retained in s_t
- We will now see how to implement these items from our wishlist



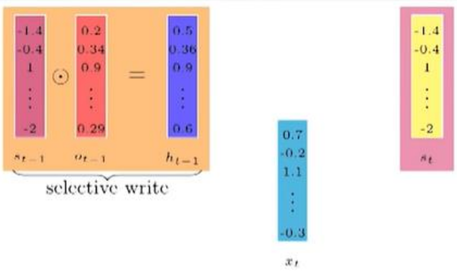
Selective Write

- Recall that in RNNs we use s_{t-1} to compute s_t
  $s_t = \sigma(Ws_{t-1} + Ux_t)$ *(ignoring bias)*
- But now instead of passing s_{t-1} as it is to s_t we want to pass (write) only some portions of it to the next state
- In the strictest case our decisions could be binary (for example, retain 1st and 3rd entries and delete the rest of the entries)
- But a more sensible way of doing this would be to assign a value between 0 and 1 which determines what fraction of the current state to pass on to the next state



Selective Write

- We introduce a vector o_{t-1} which decides what fraction of each element of s_{t-1} should be passed to the next state
- Each element of o_{t-1} gets multiplied with the corresponding element of s_{t-1}
- Each element of o_{t-1} is restricted to be between 0 and 1
- But how do we compute o_{t-1}? How does the RNN know what fraction of the state to pass on?



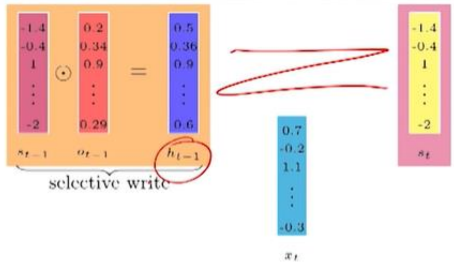
Selective Write

- Well the RNN has to learn o_{t-1} along with the other parameters (W, U, V)
- We compute o_{t-1} and h_{t-1} as

$$o_{t-1} = \sigma(W_o h_{t-2} + U_o x_{t-1} + b_o)$$
$$h_{t-1} = o_{t-1} \odot \sigma(s_{t-1})$$

R R R F R R
The movie was long but really amazing
R

o_{t-1}

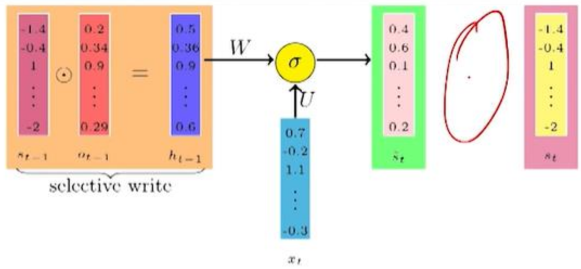


Selective Write

- Well the RNN has to learn o_{t-1} along with the other parameters (W, U, V)
- We compute o_{t-1} and h_{t-1} as

$$o_{t-1} = \sigma(W_o h_{t-2} + U_o x_{t-1} + b_o)$$
$$h_{t-1} = o_{t-1} \odot \sigma(s_{t-1})$$

- The parameters W_o, U_o, b_o need to be learned along with the existing parameters W, U, V
- The sigmoid (logistic) function ensures that the values are between 0 and 1
- o_t is called the output gate as it decides how much to pass (write) to the next time step

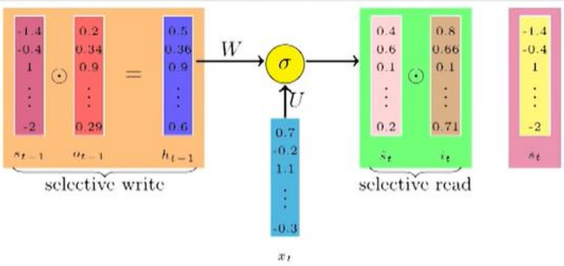


Selective Read

- We will now use h_{t-1} to compute the new state at the next time step
- We will also use x_t which is the new input at time step t

$$\tilde{s}_t = \sigma(W h_{t-1} + U x_t + b)$$

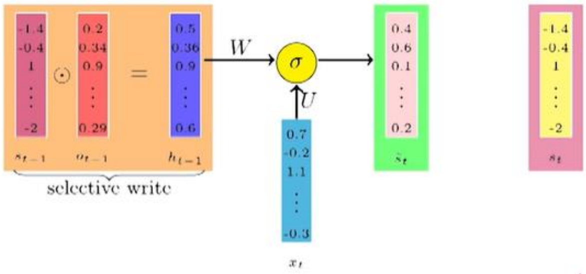
s_{t-1}



Selective Read

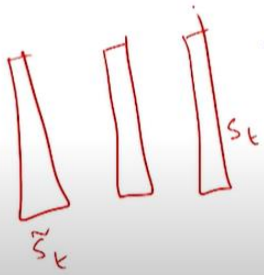
- $\tilde{s}_t$ thus captures all the information from the previous state (h_{t-1}) and the current input x_t
- However, we may not want to use all this new information and only selectively **read** from it before constructing the new cell state s_t

Selective Read

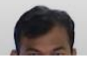


- We will now use h_{t-1} to compute the new state at the next time step
- We will also use x_t which is the new input at time step t

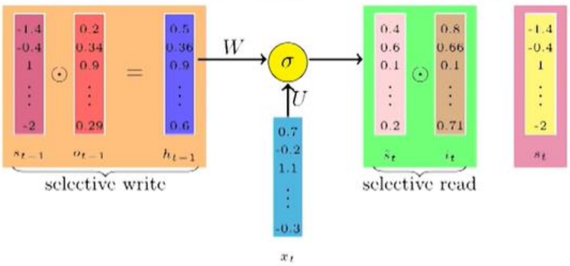
$$\tilde{s}_t = \sigma(Wh_{t-1} + Ux_t + b)$$



- Note that W, U and b are similar to the parameters that we used in RNN (for simplicity we have not shown the bias b in the figure)



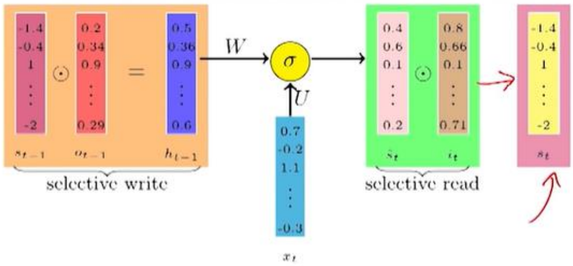
Selective Read



- $\tilde{s}_t$ thus captures all the information from the previous state (h_{t-1}) and the current input x_t
- However, we may not want to use all this new information and only selectively **read** from it before constructing the new cell state s_t
- To do this we introduce another gate called the input gate

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

- and use $i_t \odot \tilde{s}_t$ as the selectively read state information



- So far we have the following

Previous state:

s_{t-1}

Output gate:

$o_{t-1} = \sigma(W_o h_{t-2} + U_o x_{t-1} + b_o)$

Selectively Write:

$h_{t-1} = o_{t-1} \odot \sigma(s_{t-1})$

Current (temporary) state:

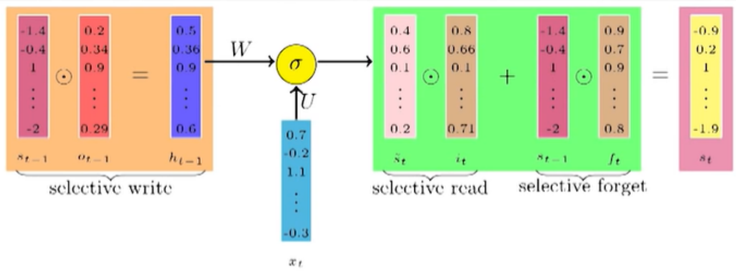
$\tilde{s}_t = \sigma(W h_{t-1} + U x_t + b)$

Input gate:

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

Selectively Read:

$i_t \odot \tilde{s}_t$



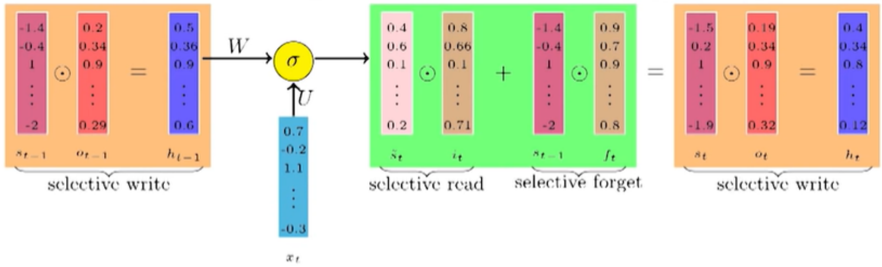
Selective Forget

- How do we combine s_{t-1} and $\tilde{s}_t$ to get the new state
- Here is one simple (but effective) way of doing this:

$$s_t = s_{t-1} + i_t \odot \tilde{s}_t$$

- But we may not want to use the whole of s_{t-1} but forget some parts of it
- To do this we introduce the forget gate

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



- We now have the full set of equations for LSTMs
- The green box together with the selective write operations following it, show all the computations which happen at timestep t

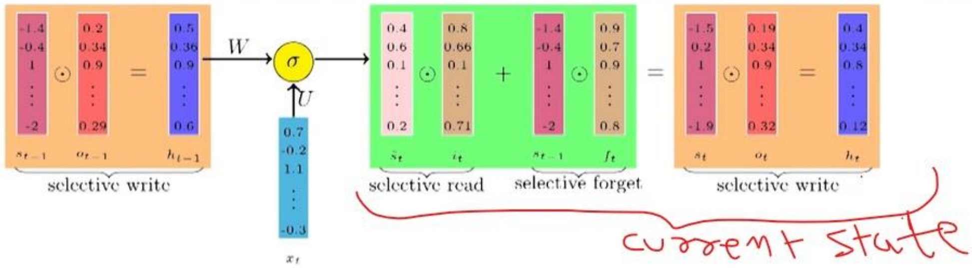
Gates:

$$o_t = \sigma(W_o h_{t-1} + U_o x_t + b_o)$$
$$i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i)$$
$$f_t = \sigma(W_f h_{t-1} + U_f x_t + b_f)$$

States:

$$\tilde{s}_t = \sigma(W h_{t-1} + U x_t + b)$$
$$s_t = f_t \odot s_{t-1} + i_t \odot \tilde{s}_t$$
$$h_t = o_t \odot \sigma(s_t)$$

(h_t is $a^{<t>}$ and s_t is $c^{<t>}$)
cell state



- We now have the full set of equations for LSTMs
- The green box together with the selective write operations following it, show all the computations which happen at timestep t

Gates:

$$o_t = \sigma(W_o h_{t-1} + U_o x_t + b_o)$$
$$i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i)$$
$$f_t = \sigma(W_f h_{t-1} + U_f x_t + b_f)$$

States:

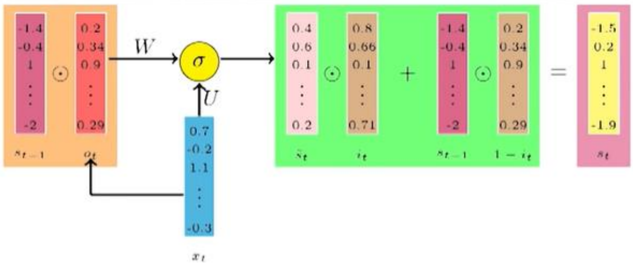
$$\tilde{s}_t = \sigma(W h_{t-1} + U x_t + b)$$
$$s_t = f_t \odot s_{t-1} + i_t \odot \tilde{s}_t$$
$$h_t = o_t \odot \sigma(s_t)$$

(h_t is $a^{<t>}$ and s_t is $c^{<t>}$)
cell state

24:31 / 31:19

Note

- LSTM has many variants which include different number of gates and also different arrangement of gates
- The one which we just saw is one of the most popular variants of LSTM
- Another equally popular variant of LSTM is Gated Recurrent Unit which we will see next



The full set of equations for GRUs

Gates:

$$o_t = \sigma(W_o s_{t-1} + U_o x_t + b_o)$$
$$i_t = \sigma(W_i s_{t-1} + U_i x_t + b_i)$$

States:

$$\tilde{s}_t = \sigma(W(o_t \odot s_{t-1}) + U x_t + b)$$
$$s_t = (1 - i_t) \odot s_{t-1} + i_t \odot \tilde{s}_t$$

- No explicit forget gate (the forget gate and input gates are tied)
- The gates depend directly on s_{t-1} and not the intermediate h_{t-1} as in the case of LSTMs