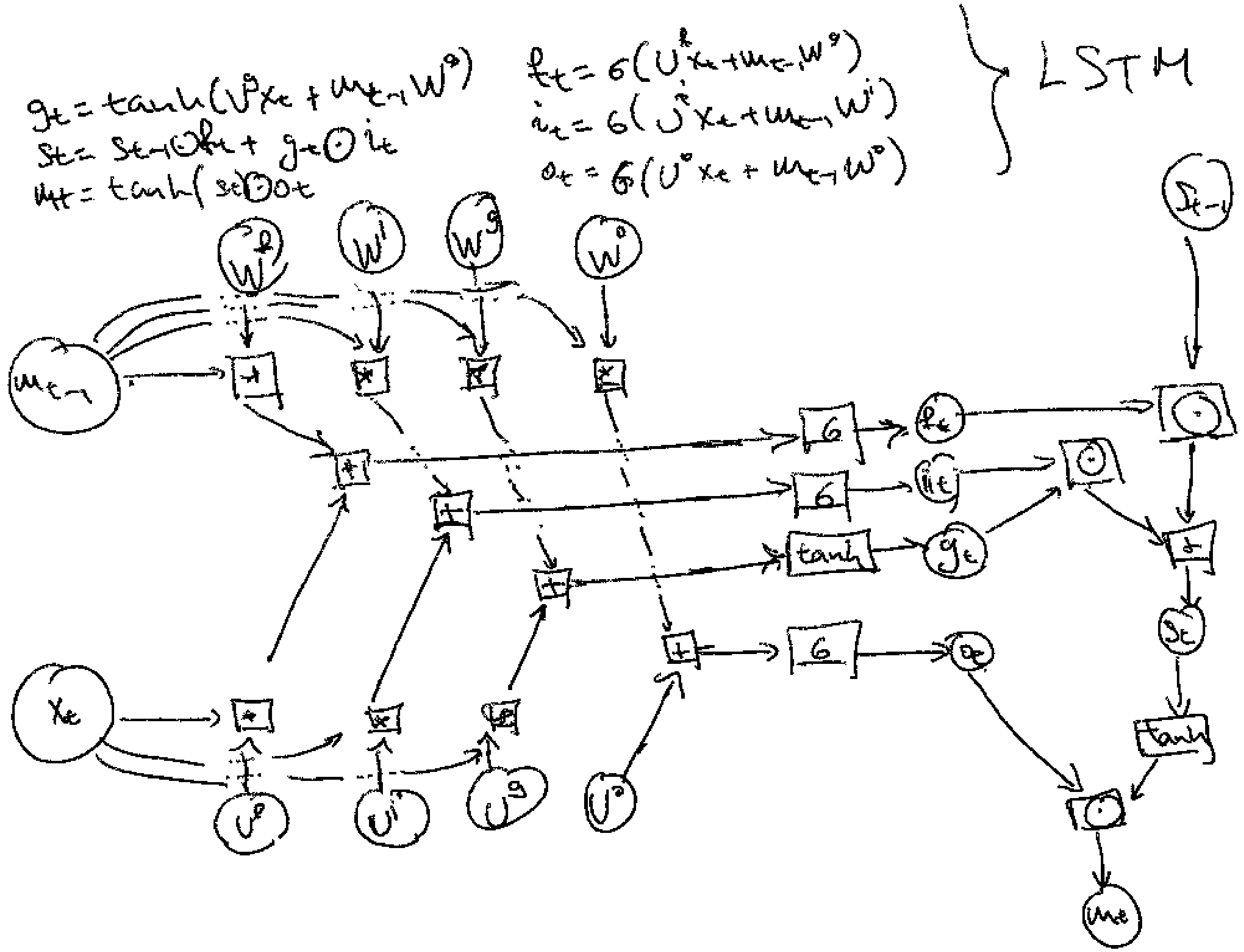
# LSTM and variants



# Recap: vanishing gradients in RNNs

- RNN equations

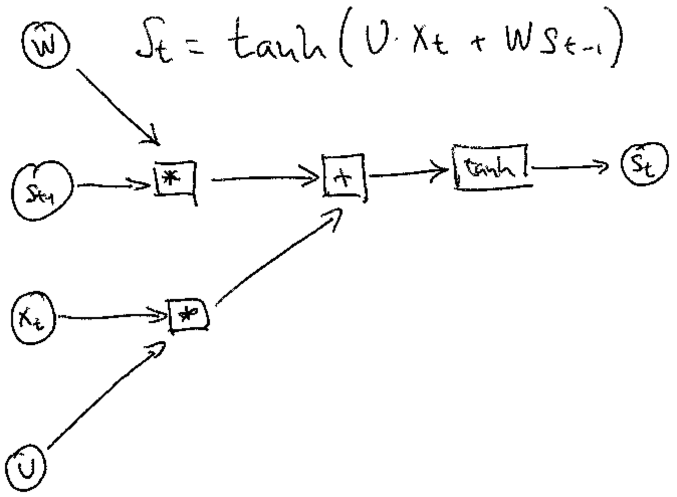
$$\boldsymbol{s}_t = \tanh(U \cdot \boldsymbol{x}_t + W \cdot \boldsymbol{s}_{t-1})$$

- The RNN gradient of the memory parameters

$$\frac{\partial \mathcal{L}_t}{\partial \boldsymbol{W}} = \sum_{i=0}^{t} \frac{\partial \mathcal{L}_t}{\partial y_t} \frac{\partial y_t}{\partial \boldsymbol{s}_t} \left( \prod_{j=i+1}^{t} \frac{\partial \boldsymbol{s}_j}{\partial \boldsymbol{s}_{j-1}} \right) \frac{\partial \boldsymbol{s}_i}{\partial \boldsymbol{W}}$$

- Chain multiplications: $\frac{\partial \boldsymbol{s}_j}{\partial \boldsymbol{s}_{j-1}} < 1 \rightarrow$ vanishing gradients

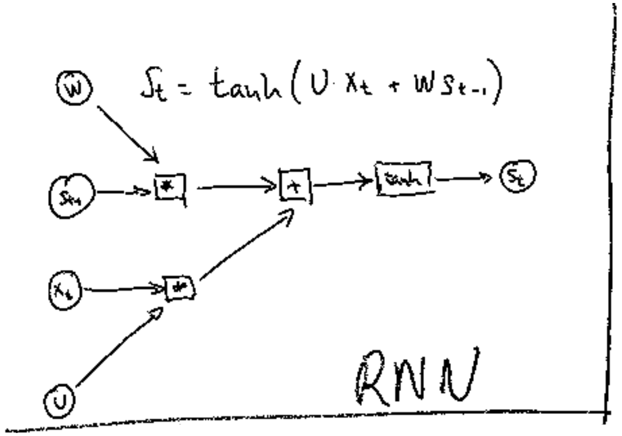
# RNN → LSTM: Key idea




$$y_t = \text{softmax}(V \cdot s_t)$$
$$\boldsymbol{s}_t = \tanh(U \cdot \boldsymbol{x}_t + W \cdot \boldsymbol{s}_{t-1})$$

- Setting $\frac{\partial s_j}{\partial s_{j-1}} = 1$ → no vanishing and exploding gradients
- Remove immediate nonlinear relation between $s_t$ and $s_{t-1}$
  - Replace tanh between $s_t$ and $s_{t-1}$ with identity
- Also, avoid continuous overwriting of state
  - Modulate the importance of new input by a gate
  - Modulate the importance of new output by a gate
  - Modulate the importance of past memories by a gate

# Designing an LSTM




$$g_t = U \cdot x_t + W s_{t-1}$$

$$s_t = \tanh(g_t)$$

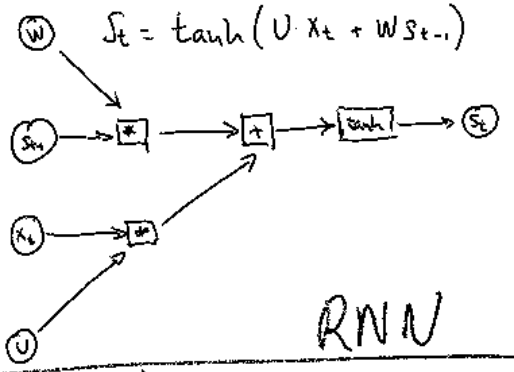
1. Drop nonlinearity between $s_t$ and $s_{t-1}$

$$g_t = U \cdot x_t + W s_{t-1}$$

$$s_t = g_t$$

# Designing an LSTM

$S_t = \tanh(U \cdot X_t + W S_{t-1})$




$g_t = U \cdot X_t + W S_{t-1}$

$S_t = \tanh(g_t)$

1. Drop nonlinearity between $S_t$ and $S_{t-1}$

$g_t = U \cdot X_t + W S_{t-1}$

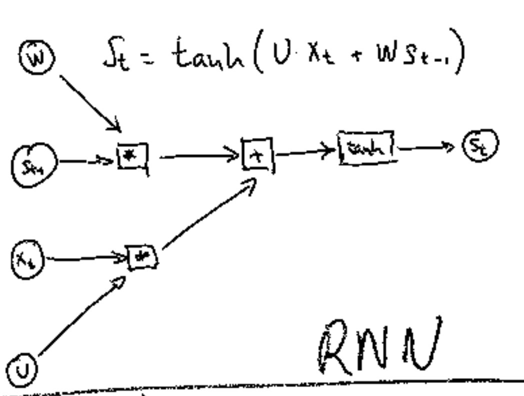
$S_t = g_t$

2. Now there is no nonlinearity in the neural network. Not good.
Add a nonlinearity on the term that does not depend on $S_{t-1}$

$g_t = \tanh(U X_t)$

$S_t = S_{t-1} + g_t$

# Designing an LSTM




$$g_t = U \cdot X_t + W S_{t-1}$$
$$S_t = \tanh(g_t)$$

1. Drop nonlinearity between $S_t$ and $S_{t-1}$

$$g_t = U \cdot X_t + W S_{t-1}$$
$$S_t = g_t$$

2. Now there is no nonlinearity in the neural network. Not good.
Add a nonlinearity on the term that does not depend on $S_{t-1}$

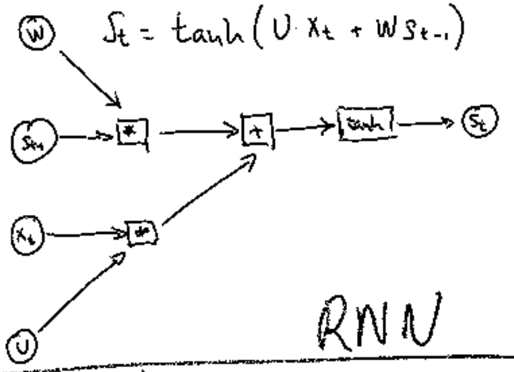
$$g_t = \tanh(U X_t)$$
$$S_t = S_{t-1} + g_t$$

3. No vanishing gradients, but also no way to modulate input/memory
Add different gates to modulate/attend with (0,1) values the key variables

$$g_t = \tanh(U^g X_t) \qquad f_t = \sigma(U^f X_t)$$
$$S_t = S_{t-1} \odot f_t + g_t \odot i_t \qquad i_t = \sigma(U^i X_t)$$

# Designing an LSTM

$s_t = \tanh(U \cdot x_t + W s_{t-1})$




$$g_t = U \cdot x_t + W s_{t-1}$$
$$s_t = \tanh(g_t)$$

1. Drop nonlinearity between $s_t$ and $s_{t-1}$

$$g_t = U \cdot x_t + W s_{t-1}$$
$$s_t = g_t$$

2. Now there is no nonlinearity in the neural network. Not good.
Add a nonlinearity on the term that does not depend on $s_{t-1}$

$$g_t = \tanh(U x_t)$$
$$s_t = s_{t-1} + g_t$$

3. No vanishing gradients, but also no way to modulate input/memory
Add different gates to modulate/attend with $(0,1)$ values the key variable

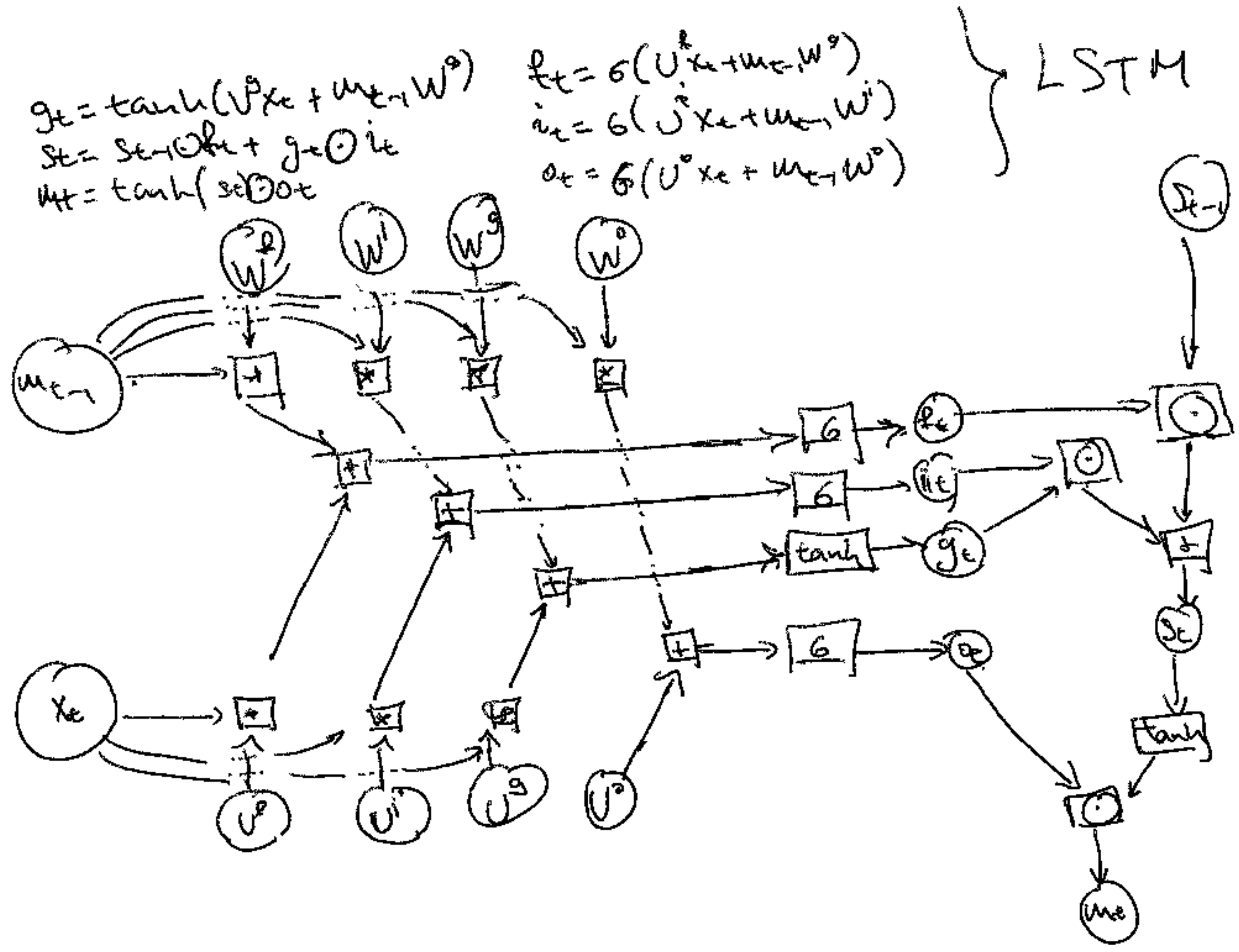
$$g_t = \tanh(U^g x_t) \qquad f_t = \sigma(U^f x_t)$$
$$s_t = s_{t-1} \odot f_t + g_t \odot i_t \qquad i_t = \sigma(U^i x_t)$$

4. All good, but the output plays no role in the modulation. Add an output variable

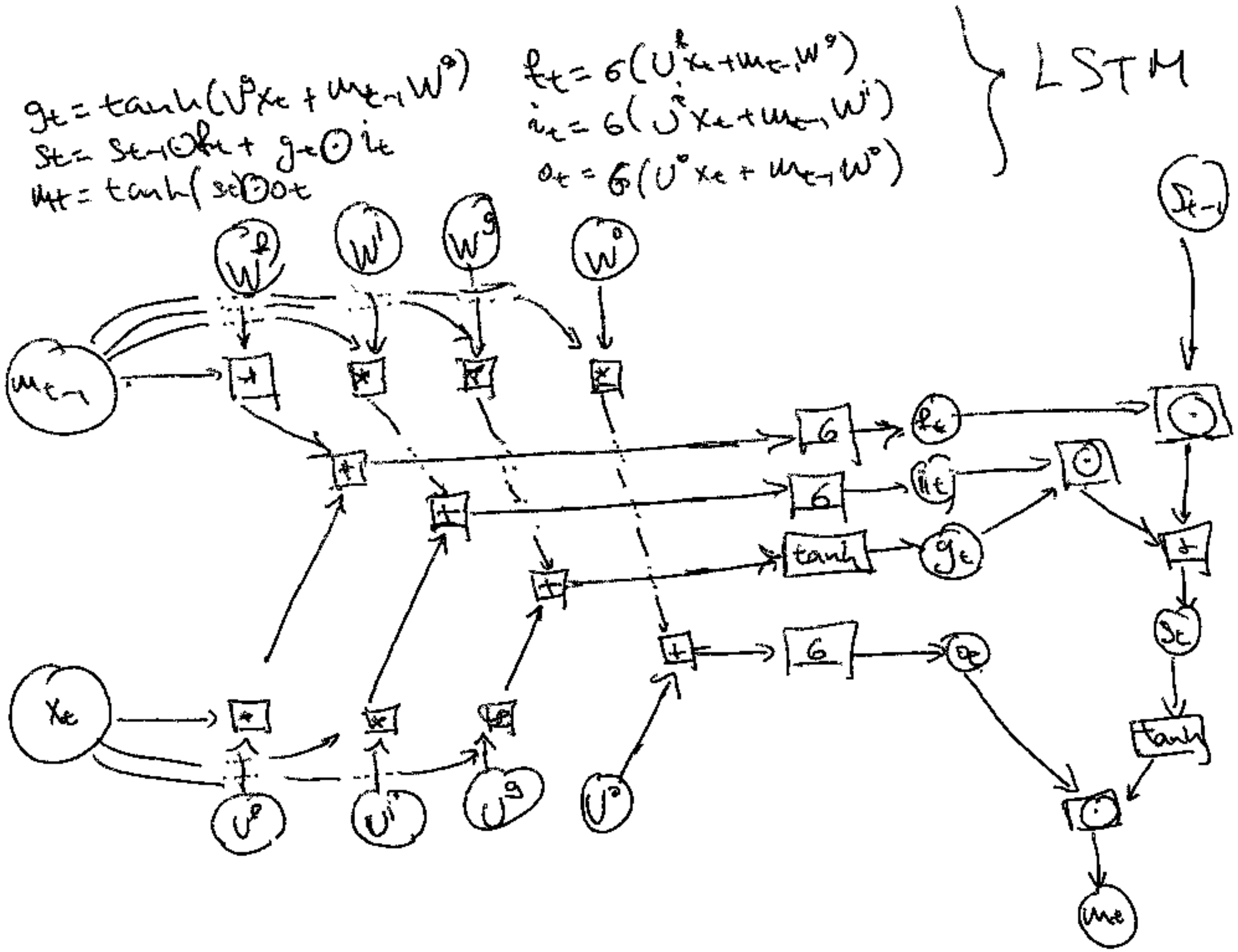
$$g_t = \tanh(U^g x_t + m_{t-1} W^g) \qquad f_t = \sigma(U^f x_t + m_{t-1} W^f)$$
$$s_t = s_{t-1} \odot f_t + g_t \odot i_t \qquad i_t = \sigma(U^i x_t + m_{t-1} W^i)$$
$$m_t = \tanh(s_t) \odot o_t \qquad o_t = \sigma(U^o x_t + m_{t-1} W^o)$$

} LSTM

# An LSTM, graphically



# LSTM: Forward propagation step by step



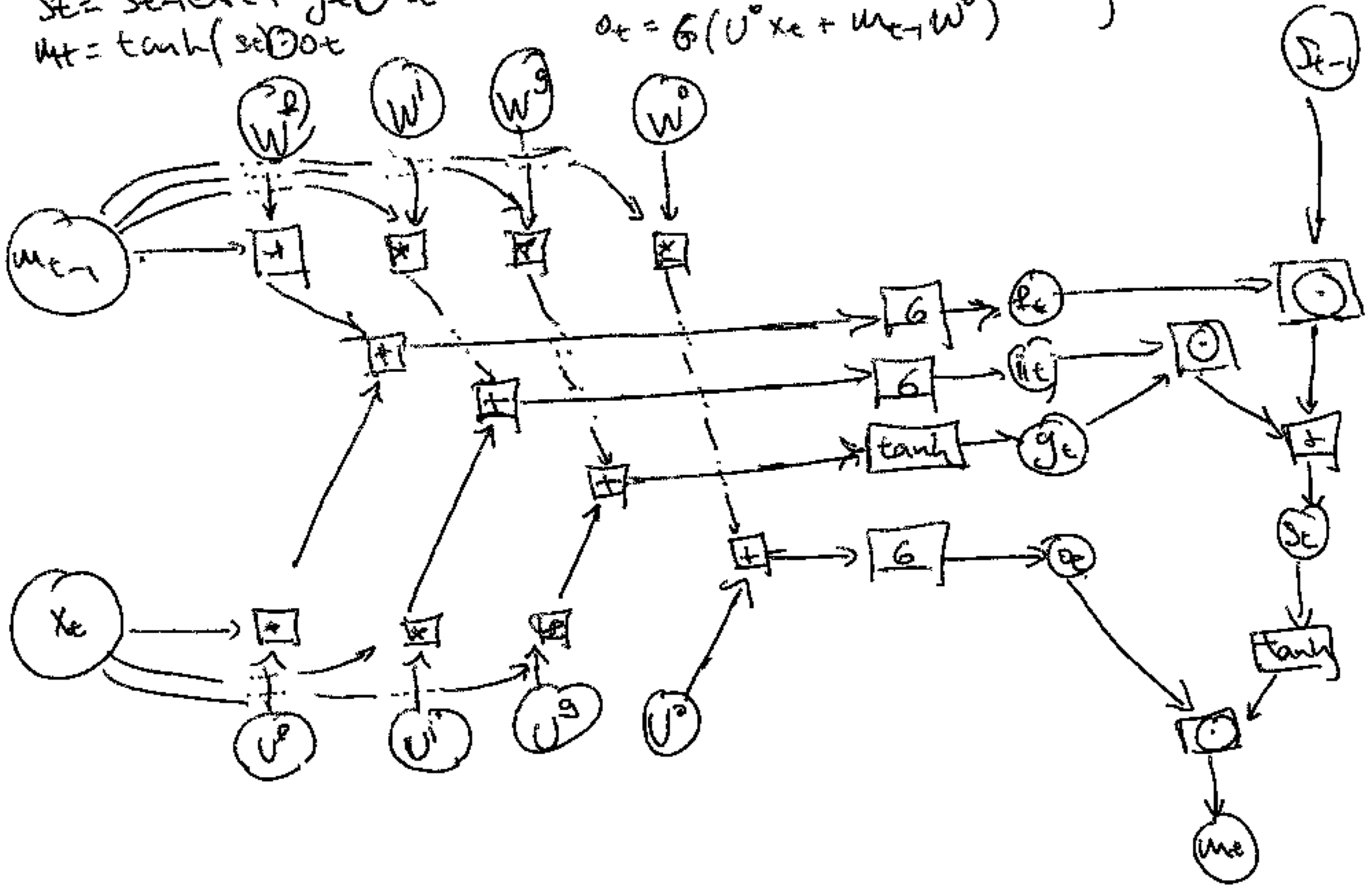

## LSTM speaking

How important is my input?

# LSTM: Forward propagation step by step

$$g_t = \tanh(U^g x_t + m_{t-1} W^g)$$
$$s_t = s_{t-1} \odot f_t + g_t \odot i_t$$
$$m_t = \tanh(s_t) \odot o_t$$

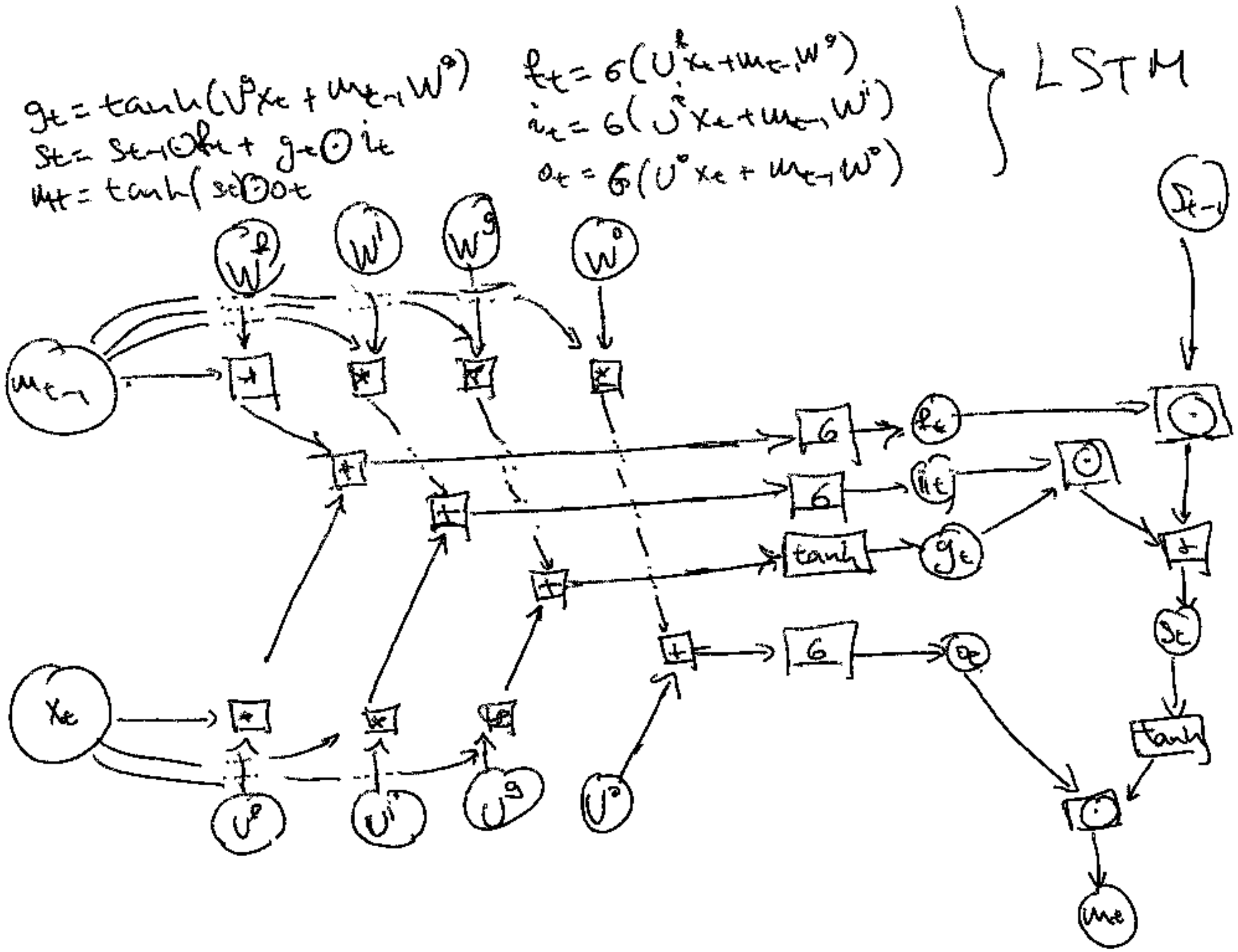
$$\left.\begin{aligned} f_t &= \sigma(U^f x_t + m_{t-1} W^f) \\ i_t &= \sigma(U^i x_t + m_{t-1} W^i) \\ o_t &= \sigma(U^o x_t + m_{t-1} W^o) \end{aligned}\right\} \text{LSTM}$$




## LSTM speaking

How important is my past state?

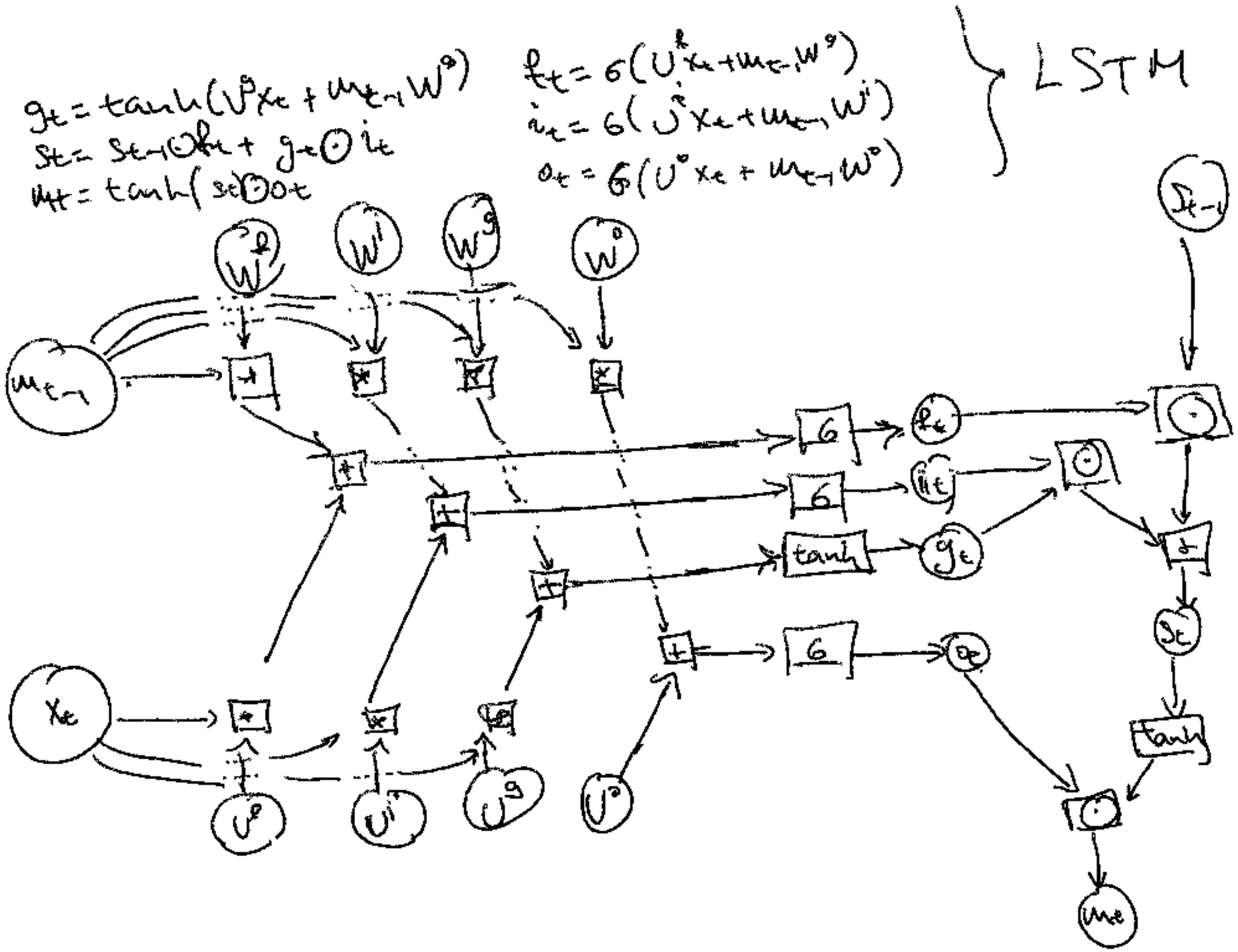
# LSTM: Forward propagation step by step




## LSTM speaking

What could be a relevant new memory?

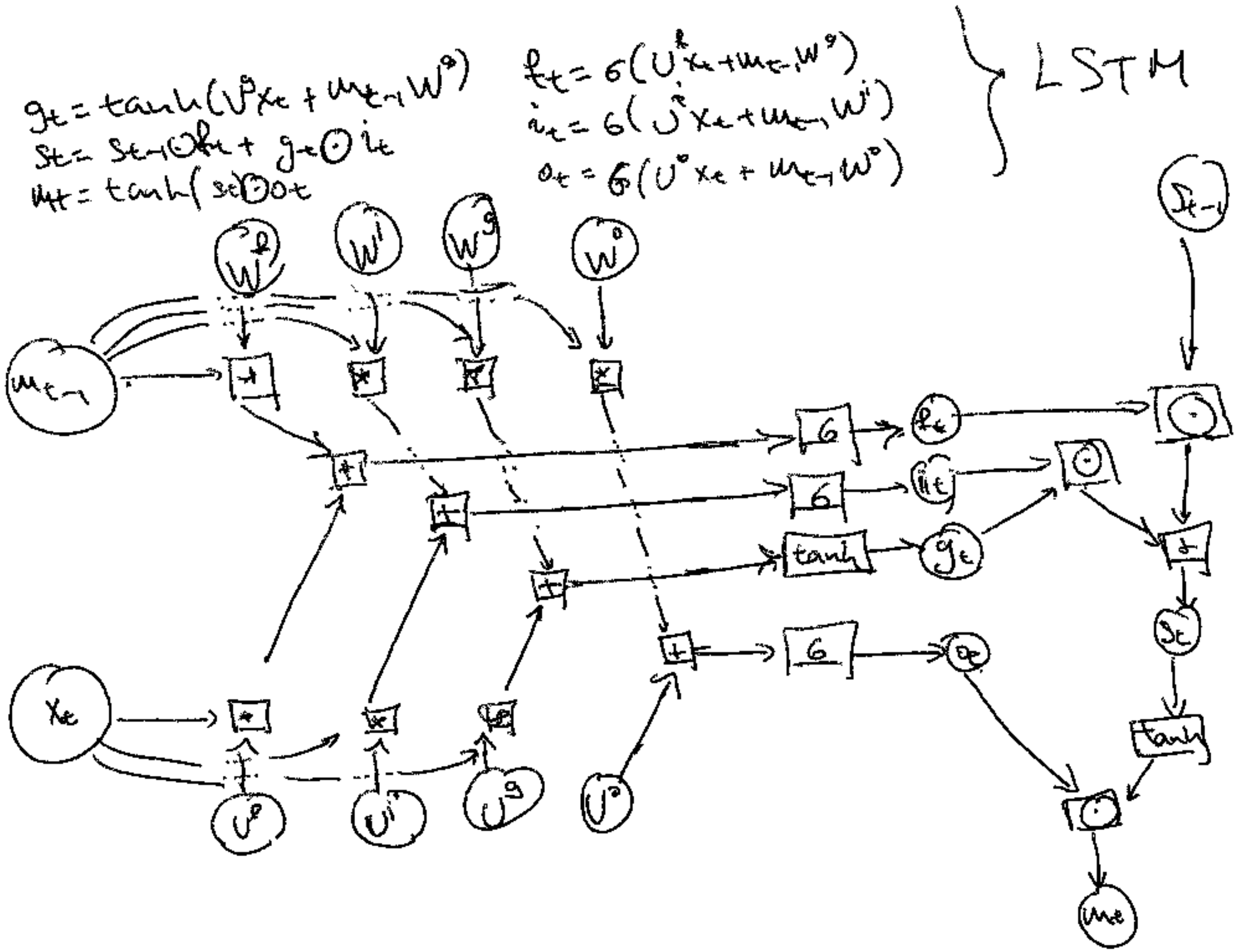
# LSTM: Forward propagation step by step




<u>LSTM speaking</u>

Ok, let's compute my new state

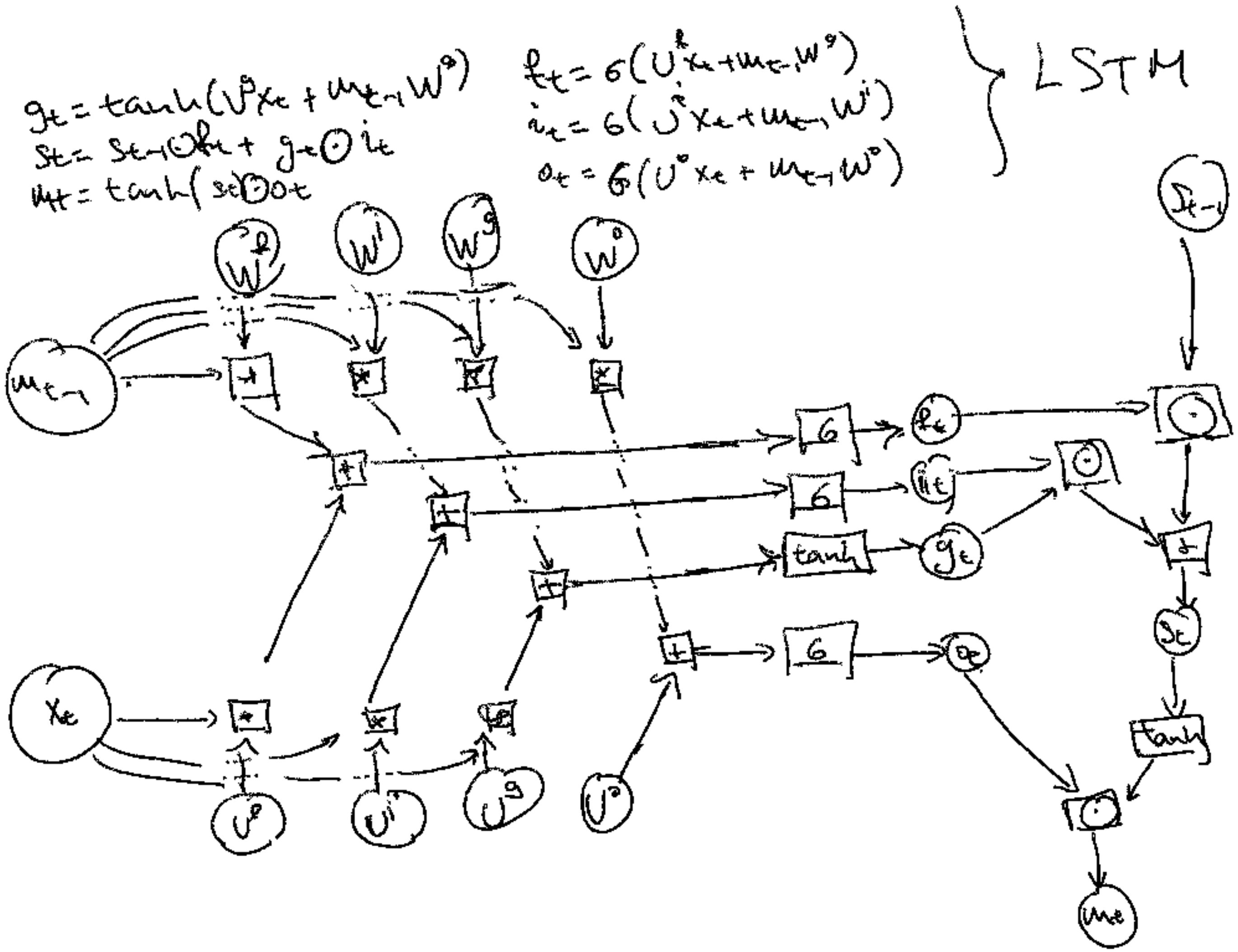
# LSTM: Forward propagation step by step




LSTM speaking

Is my new state useful for output? Check first how important is to give an output.

# LSTM: Forward propagation step by step




LSTM speaking

What is my new output?

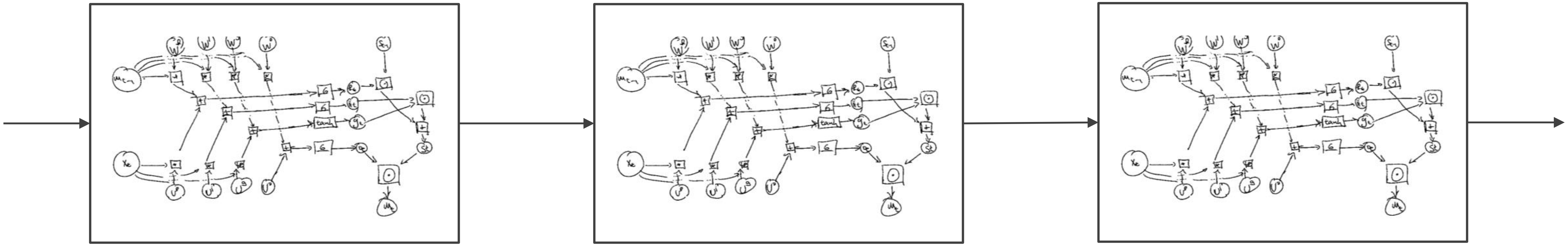
# LSTM insights

- Comparing the state equations between RNN and LSTM
  - RNN: $\boldsymbol{s}_t = \tanh(U \cdot \boldsymbol{x}_t + W \cdot \boldsymbol{s}_{t-1})$
  - LSTM: $\boldsymbol{s}_t = \boldsymbol{s}_{t-1} \odot \boldsymbol{f}_t + \boldsymbol{g}_t \odot \boldsymbol{i}_t$, $\boldsymbol{m}_t = \tanh(\boldsymbol{s}_t) \odot \boldsymbol{o}_t$
- The LSTM also has indirect nonlinear relation between $\boldsymbol{s}_t$ and $\boldsymbol{s}_{t-1}$ via $\boldsymbol{m}_t$
  - There is also direct linear relation → Strong gradients encouraged
- Use sigmoids for gating/squashing → (0,1) values
  - Use tanh as module's recurrence nonlinearity, instead

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

# Unrolling the LSTMs

- Just the same like for RNNs
- The internal equations are more complicated but the idea is the same
- Because of linear relation between $\boldsymbol{s}_t$ and $\boldsymbol{s}_{t-1}$ vanishing gradients mitigated
  - LSTM captures short-term, as well as long-term correlations



# LSTM variants

- LSTM with peephole connections
- Gates have access also to the previous cell states $c_{t-1}$ (not only memories)
- Bi-directional recurrent networks
- Gated Recurrent Units (GRU)
- Phased LSTMs
- Skip LSTMs
- And many more …