Neural Network Langauge Models

Feedforward LM

Chia-Chi Yu

Feed-Forward Neural Network Language Model

- Language modeling can predict upcoming words from prior word context.
- The advantage of neural net-based language models compared with n-gram language models.
  - Don't need smoothing.

    - not smooth, small changes in discrete context may result in large changes in probability estimate.
  - Handle much longer histories
  - Generalize over contexts of similar words

    - Today's presentation is given by John.

    p(John|...) = ?, P(Mary|...) = ?
- Neural net language models are strikingly slower to train than traditional language models.

Feed-Forward Neural Network Language Model

- Takes as input at time *t* a representation of some number of previous words (w_{t-1}, w_{t-2}, etc.) outputs a probability distribution over possible next words.
- Approximates the probability of a word given the entire prior context $P(w_t|w_{t-N+1}^{t-1})$ by approximating based on the N previous words:

$$P(w_t|w_1^{t-1}) \approx P(w_t|w_{t-N+1}^{t-1})$$

Word Embeddings

- In neural language models, the prior context is represented by embeddings of the previous words.
- But what is embedding?

  -Convert text to number
- Why embedding?

  -generalize unseen data much better than n-gram language models which representing the prior context by exact words.

  Ex. I have to make sure when I get home to feed the cat.

  N-gram LM: I have to make sure when I get home to feed the ___. => “cat”

  Neural LM: I have to make sure when I get home to feed the___. =>“cat or dog”

A one-hot vector

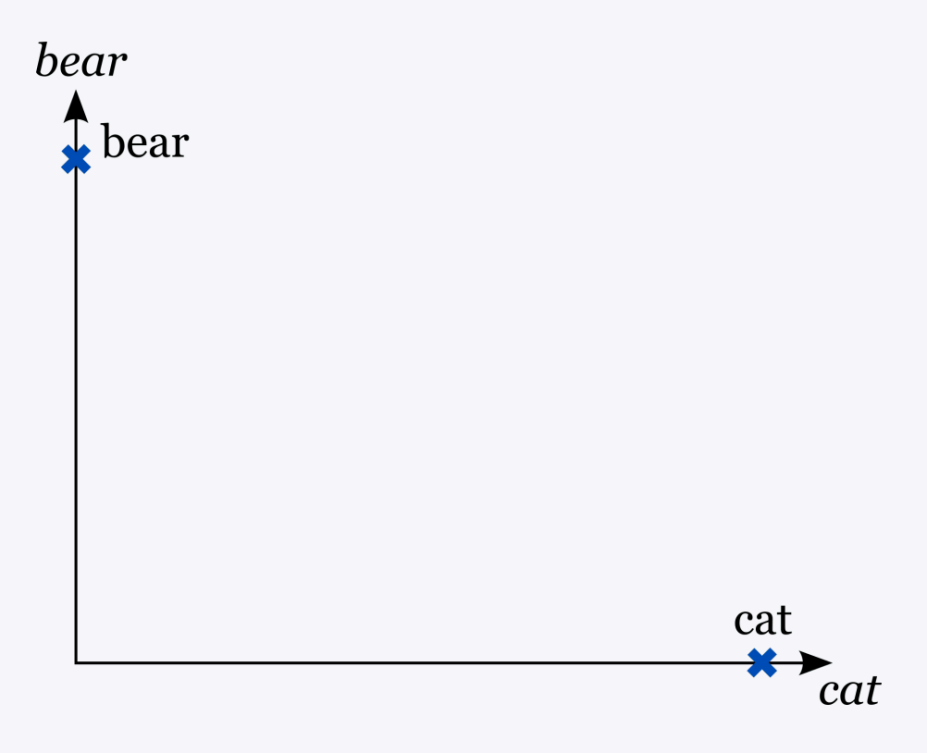
- The vector is lexicon size.
- Each dimension corresponds to a word in the lexicon.
- The dimension for the word is 1, and others are 0.

	Bear	Cat	Frog
Bear	1	0	0
Cat	0	1	0
Frog	0	0	1

Bear = [1, 0, 0] cat = [0, 1, 0]. Frog = [0, 0, 1]

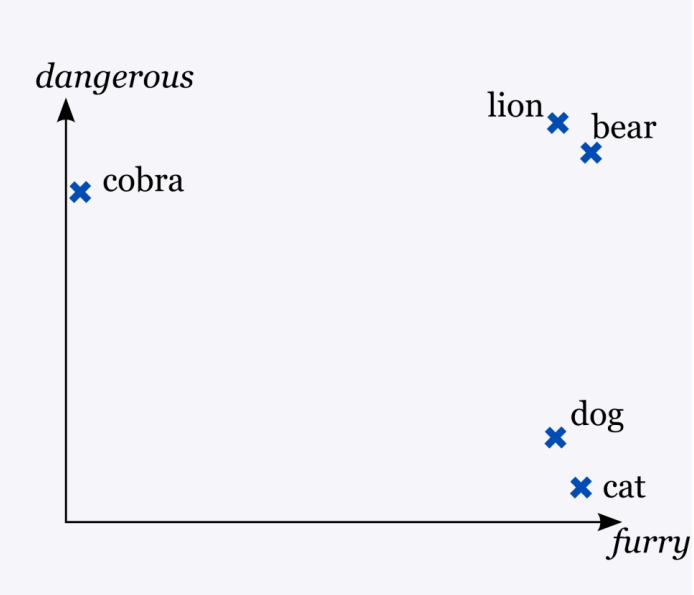
A one-hot vector

- It's a very large vector!
- It tells us very little
- Does not show similarity.



Distributed Vectors

- Each element represents a property, and they are shared between the words.
- Group similar words/objects together

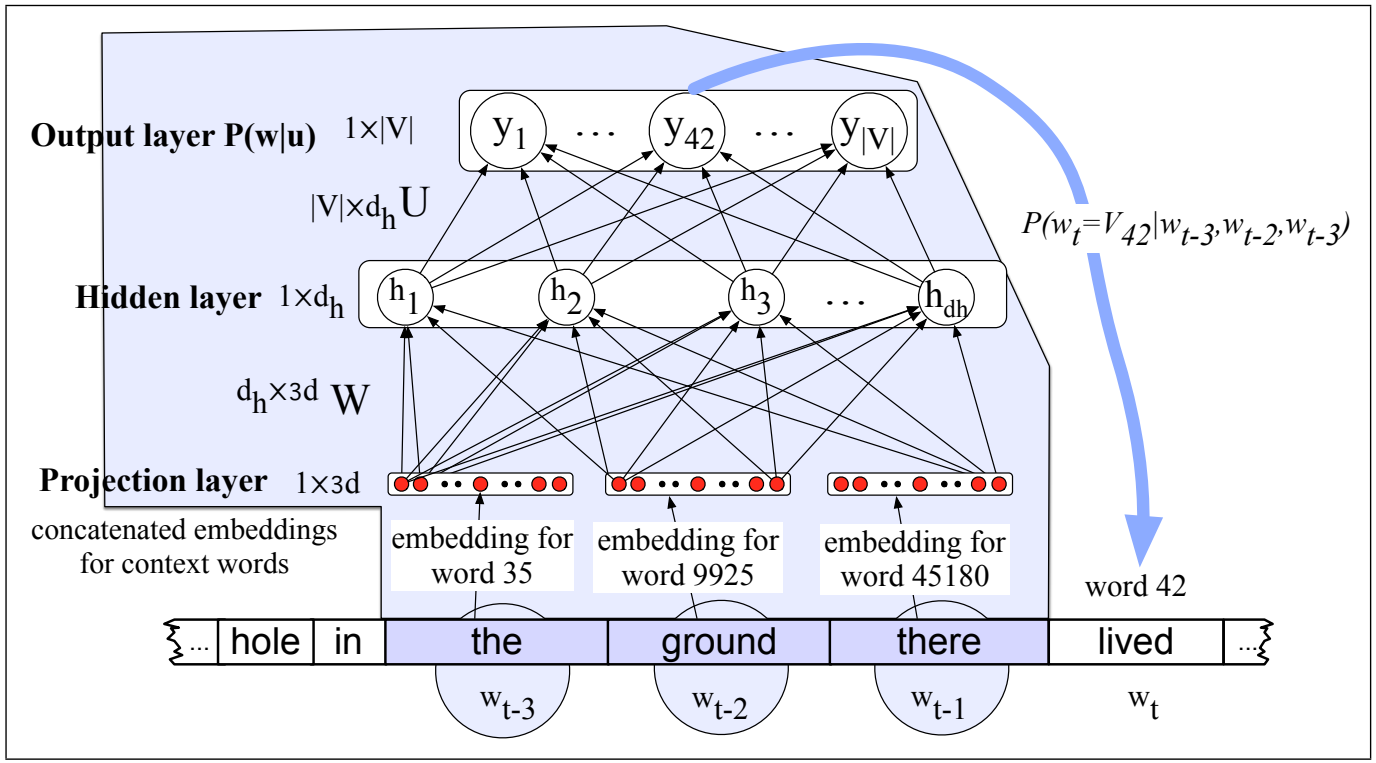


	furry	*dangerous*
bear	0.9	0.85
cat	0.85	0.15
cobra	0.0	0.8
lion	0.85	0.9
dog	0.8	0.15

A simplified version of a feedforward neural language model

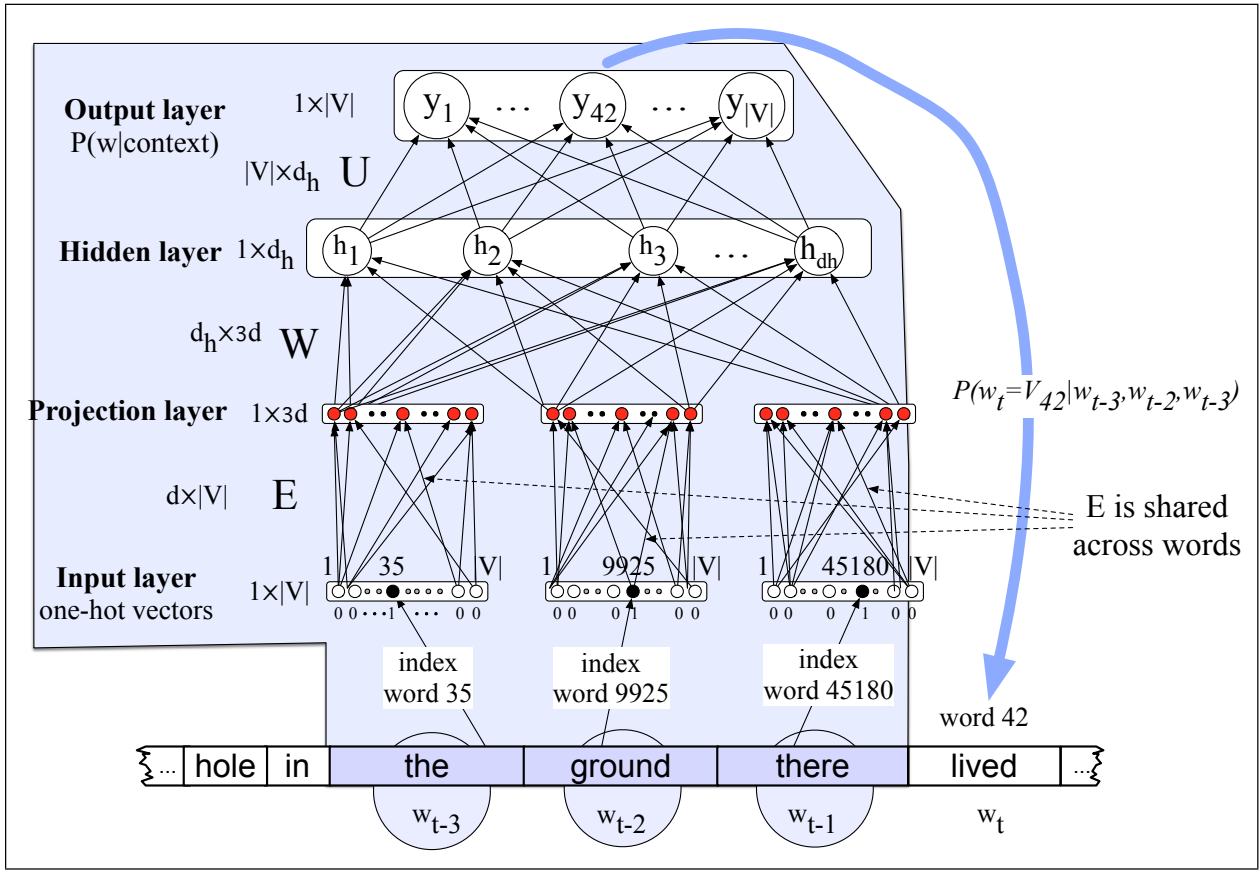
Inputs: vector representations of previous words w_{t-3}, w_{t-2}, w_{t-1}

Output: the conditional probability of w_t being the next word



Learn the embedding during LM training

1. Adding one-hot vectors as an input layer
2. Select three embeddings from E
3. Multiply by W
4. Multiply by U
5. Apply softmax



Training the neural language model

- To train model, set all the parameters $\theta = E, W, U, b$
- Do gradient descent
- Use error back propagation to compute the gradient
- Generally training proceeds by taking as input a very long text, concatenating all the sentences, starting with random weights, and iteratively moving through the text predicating each word w_t.

$$L = -\log p(w_t|w_{t-1}, ..., w_{t-n+1})$$

$$\theta_{t+1} = \theta_t - \eta \frac{\partial -\log p(w_t|w_{t-1}, ..., w_{t-n+1})}{\partial \theta}$$

LSTM Networks

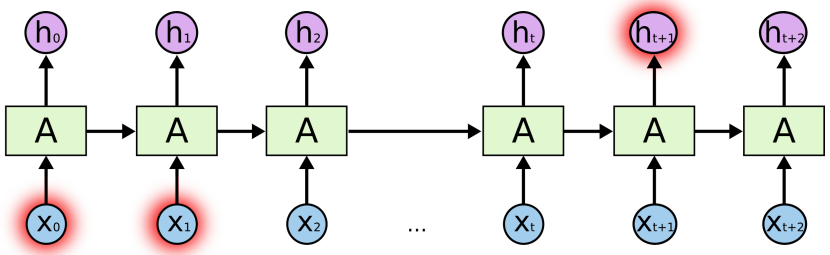
Long Short-term Memory

What is LSTM?

- Solve two problems in the context management
  1. Removing information no longer needed from the context
  2. Adding information likely to be needed for later decision making.

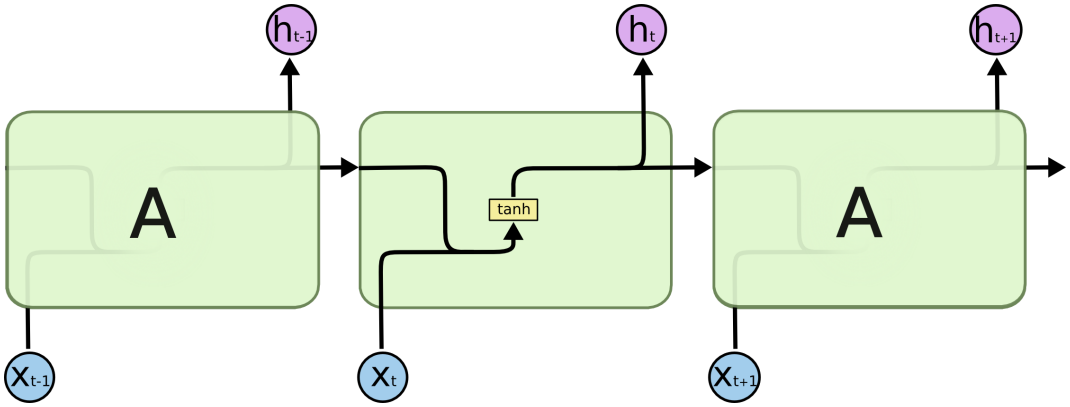
  ex. “I grew up in France... I speak fluent French.”

  Sometimes, it’s possible for the gap between the relevant information and the point where it is needed to become very large.
- As the gap grows, RNNs become unable to learn to connect the information.

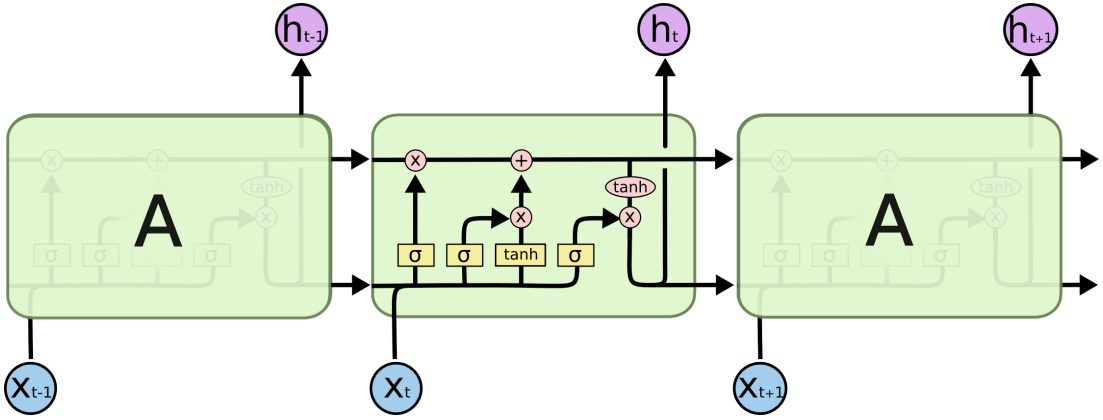


How LSTM works?

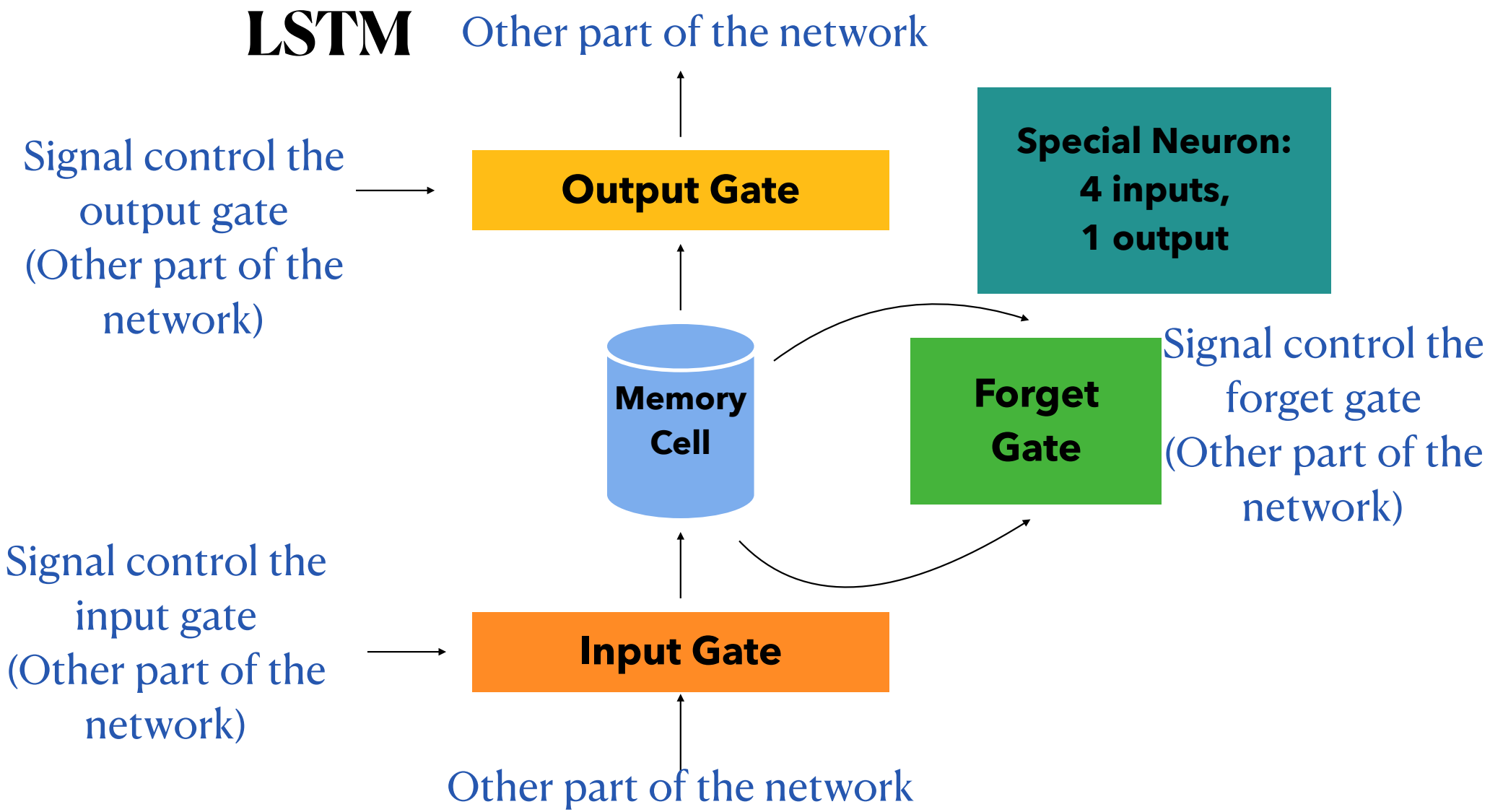
- LSTMs have a chain-like structure. Instead of having a single neural network layer, there are four, interacting in a very special way.



The repeating module in a standard RNN contains a single layer.

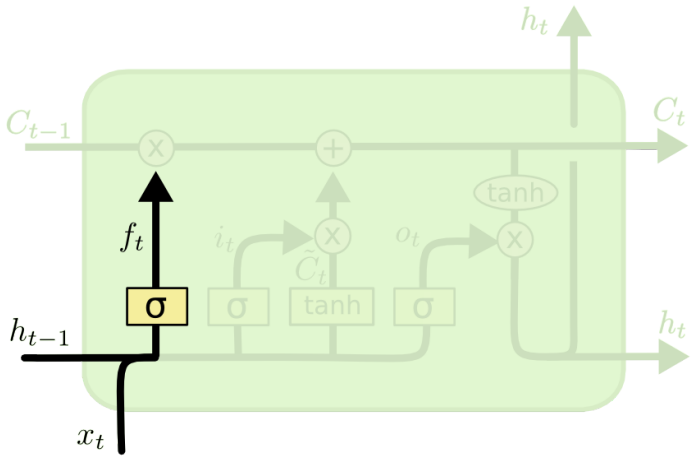


LSTM


Other part of the network
Signal control the output gate (Other part of the network)
Output Gate
Special Neuron: 4 inputs, 1 output
Memory Cell
Forget Gate
Signal control the forget gate (Other part of the network)
Signal control the input gate (Other part of the network)
Input Gate
Other part of the network

Step-by-Step LSTM Walk Through

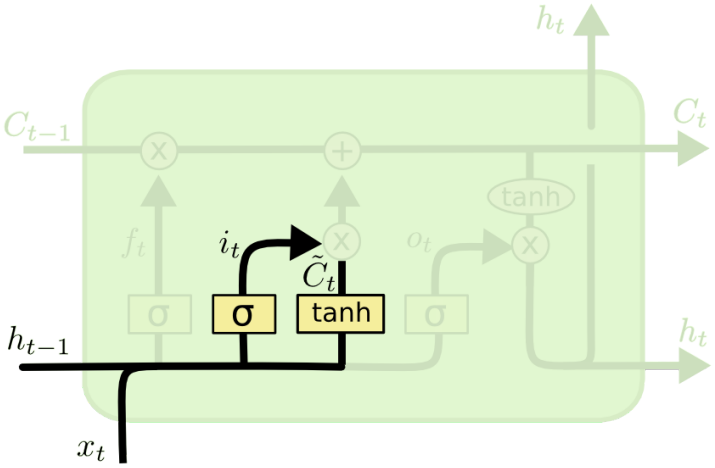
- The first step is to decide what information we're going to throw away from the cell memory state.
- The decision is made by a sigmoid layer called the "**forget gate layer**".
- Looks at h_{t-1} and x_t, and outputs a number between 0 and 1 for each number in the cell memory state C_{t-1}.
- A **1** represents "completely keep this" while a **0** represents "completely get rid of this."



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

Step-by-Step LSTM Walk Through

- The second step is to decide what new information we're going to store in the cell memory state.
- A sigmoid layer called the "input gate layer" decides which values we'll update.
- A tanh layer creates a vector of new candidate values, $\tilde{C}_t$ that could be added to the state.
- Next, we combine these two to create an update to the state.

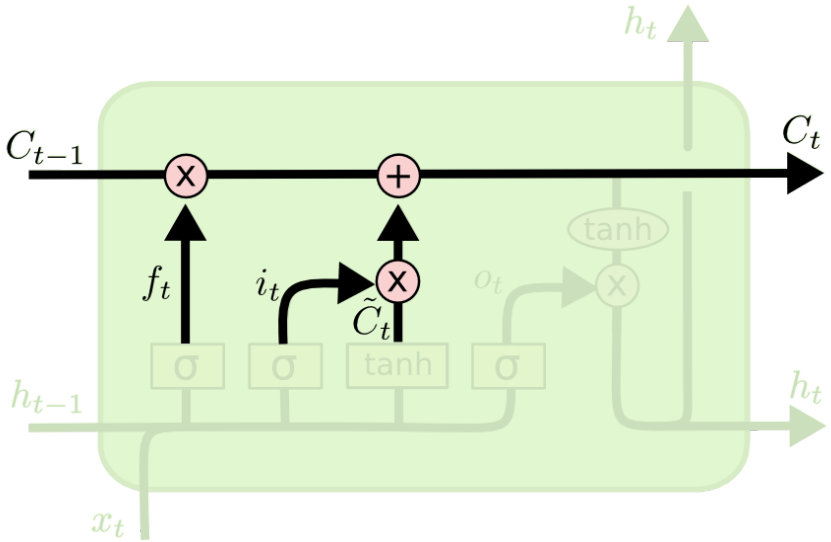


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

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

Step-by-Step LSTM Walk Through

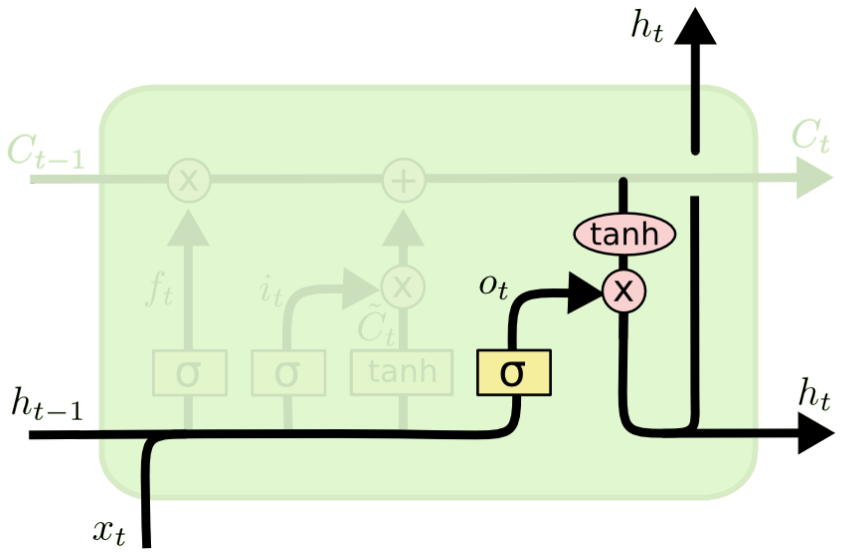
- Now, update the old cell state, C_{t-1} into the new cell state $\tilde{C}_t$.
- Multiply the old state by f_t, forgetting the things we decided to forget earlier.
- Add the new candidate values $i_t * \tilde{C}_t$, scaled by how much we decide to update each state value.



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

Step-by-Step LSTM Walk Through

- Finally, decide what we're going to output.
- Run a sigmoid layer which decides what parts of the cell state we're going to output.
- Put the cell state through tanh (to push the values to between -1 and 1) and multiply it by the output of the sigmoid gate, so that we only output the parts we decide to.

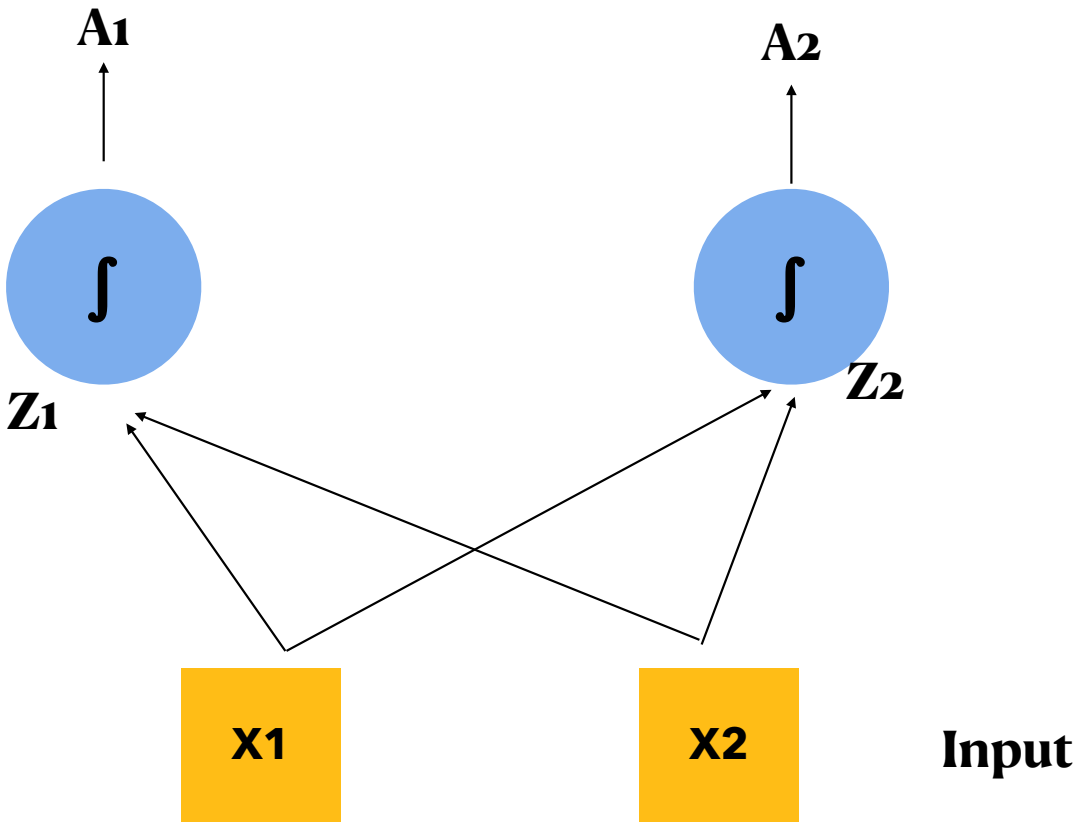


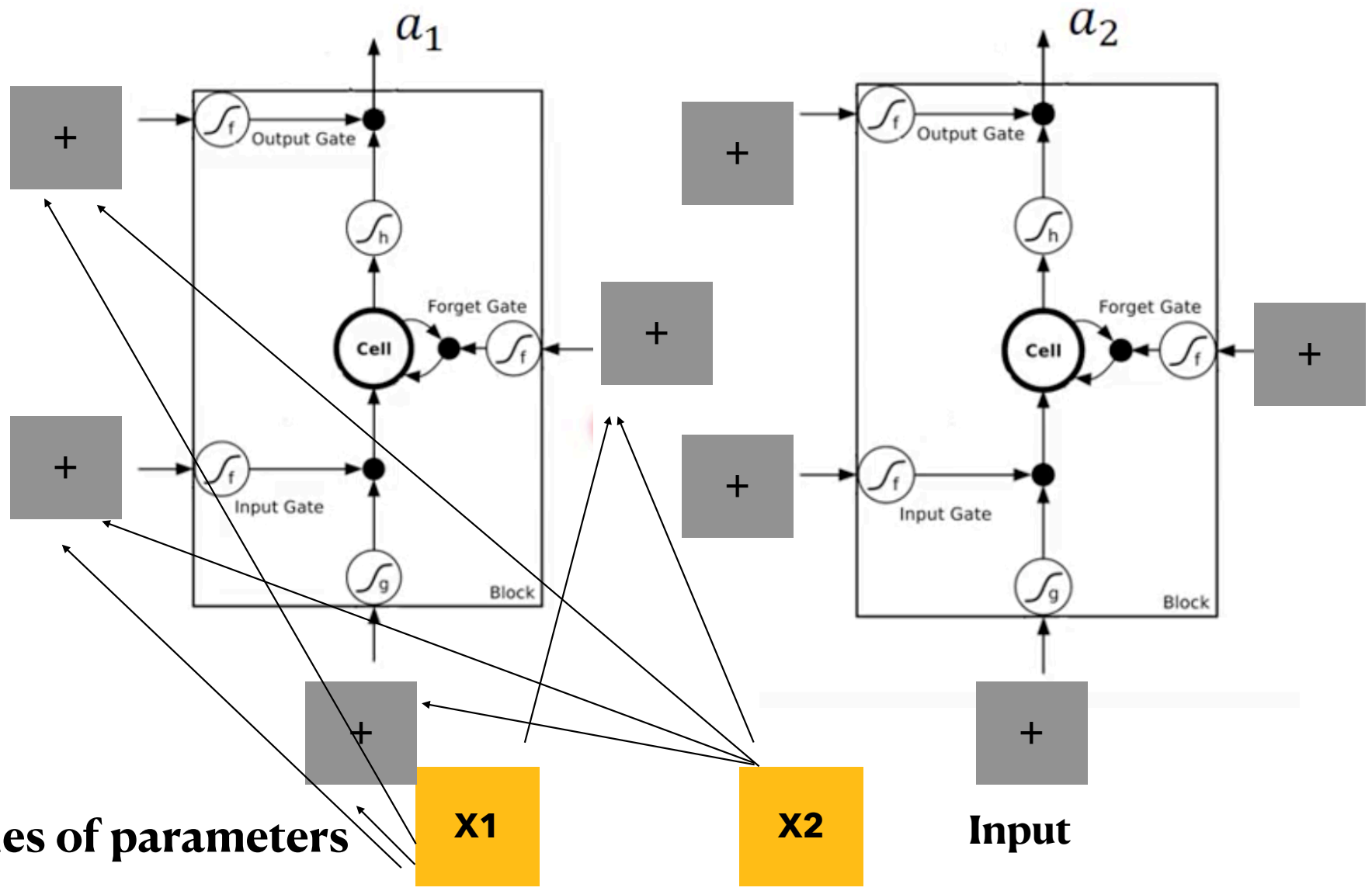
$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

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

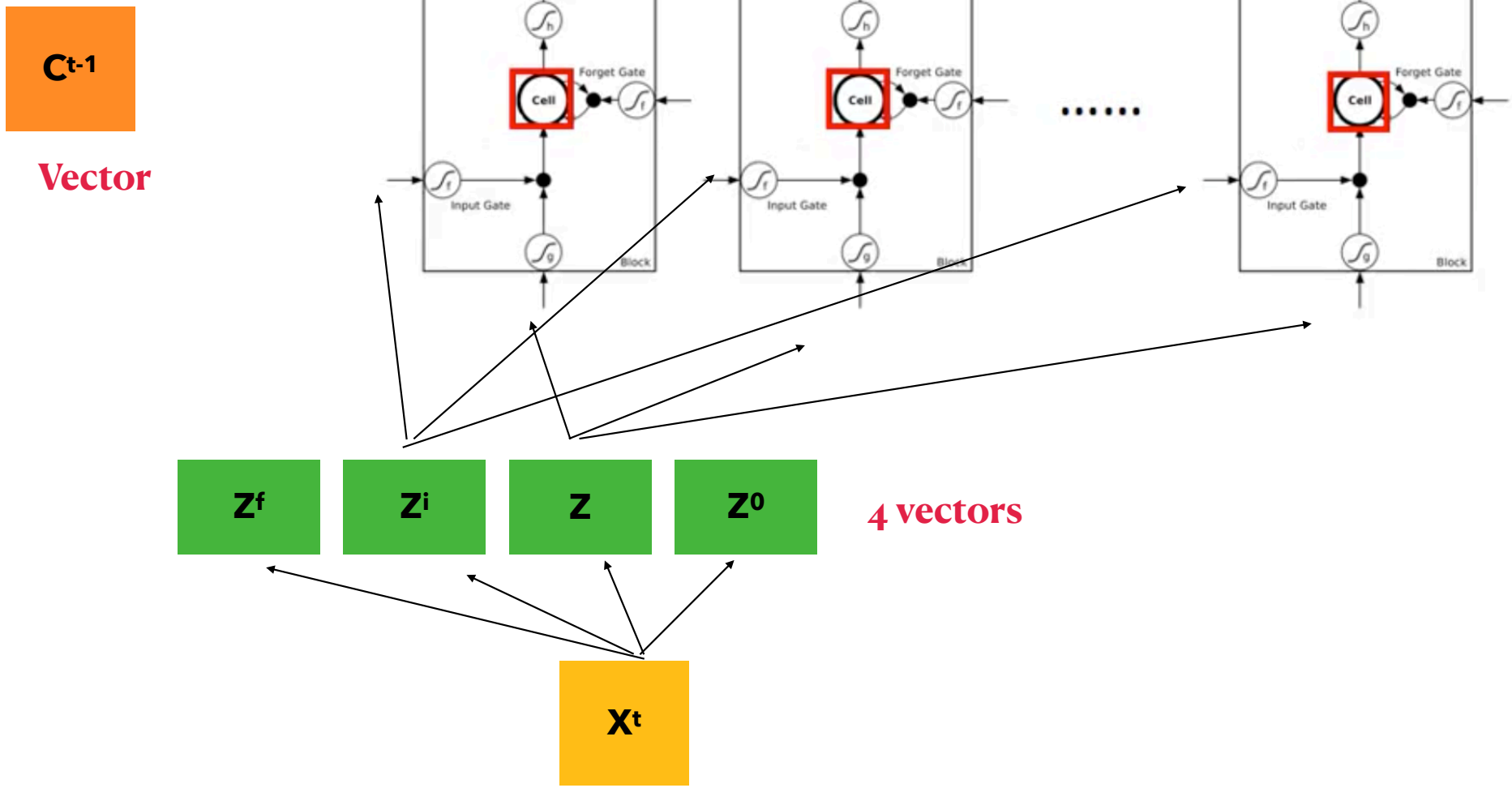
Original Network

- Simply replace the neurons with LSTM

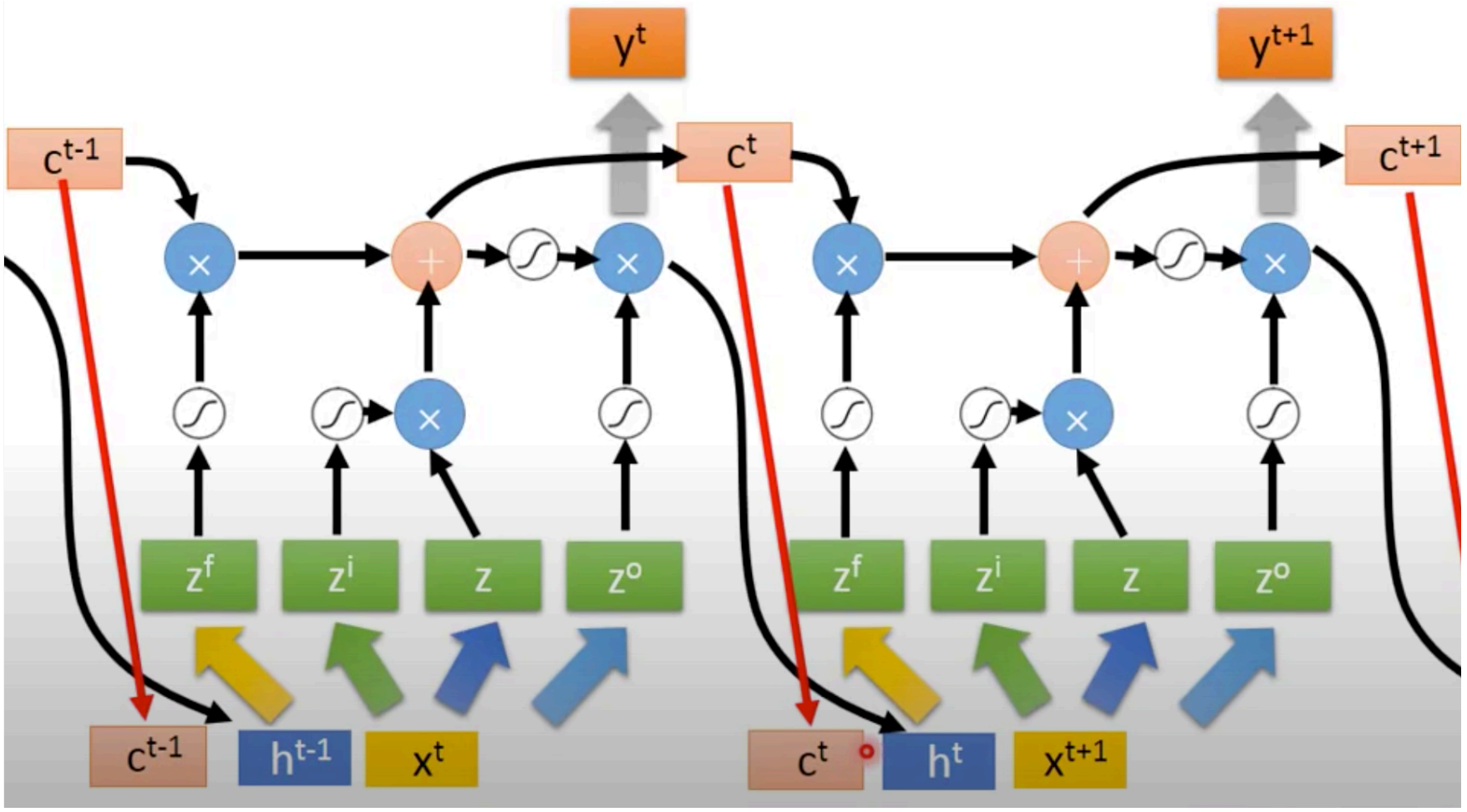



a_1
a_2
Output Gate
Forget Gate
Input Gate
Cell
Block
+
X1
X2
4 times of parameters
Input

LSTM


C^{t-1}
Vector
Output Gate
Forget Gate
Cell
Input Gate
Block
Z^f
Z^i
Z
Z^0
4 vectors
X^t

LSTM



Multiple-layer LSTM