Text generation

Agenda

- Generation problems
- N-gram generation model
- Recurrent Neural Networks

What generation is used for?

Word auto-completion

Quick red fox jumps

What generation is used for?

Word auto-completion

Quick red fox jumps

Phrase auto-completion

Why do birds fly
sing
fly in a wedge

What generation is used for?

Word auto-completion

Quick red fox jumps

Phrase auto-completion

Why do birds fly
sing
fly in a wedge

Dialog systems

– What is the weather in Moscow?

It's 15 degrees Celsius in Moscow –

Problem statement

The text must meet the following requirements:

- Logical coherence
- Compliance with language norms

We can try to set the rules manually.
But there are too many rules, so nothing good will come of it.

Problem statement

We will learn to **mimic** human speech

To do this, we will learn to evaluate the probability of texts

$$p(\text{Quick brown fox jumps}) \qquad p(\text{Cat sits on a mat})$$

Problem statement

We will learn to **mimic** human speech

To do this, we will learn to evaluate the probability of texts

$$p(\text{Quick brown fox jumps}) \quad \text{V} \quad p(\text{Cat sits on a mat})$$

$$p(\text{Quick brown fox jumps}) = \frac{1}{|\text{dataset}|} \qquad p(\text{Cat sits on a mat}) = \frac{1}{|\text{dataset}|}$$

Problem statement

We will learn to **mimic** human speech

To do this, we will learn to evaluate the probability of texts

$$p(\text{Quick brown fox jumps}) \quad \vee \quad p(\text{Cat sits on a mat})$$

$$\frac{1}{|\text{dataset}|} \qquad\qquad\qquad \frac{1}{|\text{dataset}|}$$

<u>It is impossible to work with text as a whole!</u>

Problem statement

Let's divide the text into words

x – text with m words.

We will train a model to estimate the probability of a set of words.

$$p(x) = p(x_1, \ldots, x_m)$$

Problem statement

Let's divide the text into words

x – text with m words.

We will train a model to estimate the probability of a set of words.

$$p(x) = p(x_1, \ldots, x_m)$$

Replace the joint probability with the product of conditional probabilities

$$p(x_1, \ldots, x_m) = p(x_1) \cdot p(x_2 | x_1) \cdot p(x_3 | x_1, x_2) \cdot \ldots \cdot p(x_m | x_1, \ldots, x_{m-1}) = \prod_{i=1}^{m} p(x_i | x_{<i})$$

Problem statement

$$p(x_1, \dots, x_m) = \prod_{i=1}^{m} p(x_i \mid x_{<i})$$

It is enough to train the model to estimate the probability $p(x_i \mid x_{<i})$

Problem statement

$$p(x_1, \ldots, x_m) = \prod_{i=1}^{m} p(x_i \mid x_{<i})$$

It is enough to train the model to estimate the probability $p(x_i \mid x_{<i})$

It is still difficult because you have to take into account <u>all</u> the previous words

Let's simplify the task - we'll only look at the previous n words.

$$p(x_1, \ldots, x_m) \approx \prod_{i=1}^{m} p(x_i \mid x_{i-1}, \ldots, x_{i-n})$$

This model is called <u>n-gram</u>

N-gram generation model

We assume that the next word depends only on the previous ones

$$p(x_1, \ldots, x_m) \approx \prod_{i=1}^{m} p(x_i \,|\, x_{i-1}, \ldots, x_{i-n})$$

N-gram generation model

We assume that the next word depends only on the previous ones

$$p(x_1, \dots, x_m) \approx \prod_{i=1}^{m} p(x_i \mid x_{i-1}, \dots, x_{i-n})$$

If a word has less than n of the previous ones, we fill in the gaps with the <PAD> token

$$\begin{aligned} p(\text{Quick}, \text{brown}, \text{fox}, \text{jumps}) = \\ & p(\text{Quick} \mid \text{<PAD>}, \text{<PAD>}) \\ & \cdot\, p(\text{brown} \mid \text{<PAD>}, \text{Quick}) \\ & \cdot\, p(\text{fox} \mid \text{Quick}, \text{brown}) \\ & \cdot\, p(\text{jumps} \mid \text{brown}, \text{fox}) \end{aligned}$$

N-gram generation model: training

- We need to estimate the probability $p(x_i \mid x_{i-1}, \ldots, x_{i-n})$.
- Let's manually count how many times x_i occurred after $x_{i-1}, \ldots, x_{i-n}$.

$$p(x_i \mid x_{i-1}, \ldots, x_{i-n}) = \frac{p(x_i, x_{i-1}, \ldots, x_{i-n})}{p(x_{i-1}, \ldots, x_{i-n})} = \frac{N(x_i, x_{i-1}, \ldots, x_{i-n})}{N(x_{i-1}, \ldots, x_{i-n})}$$

N-gram generation model: training

- We need to estimate the probability $p(x_i \mid x_{i-1}, \ldots, x_{i-n})$.

- Let's manually count how many times x_i occurred after $x_{i-1}, \ldots, x_{i-n}$.

$$p(x_i \mid x_{i-1}, \ldots, x_{i-n}) = \frac{p(x_i, x_{i-1}, \ldots, x_{i-n})}{p(x_{i-1}, \ldots, x_{i-n})} = \frac{N(x_i, x_{i-1}, \ldots, x_{i-n})}{N(x_{i-1}, \ldots, x_{i-n})}$$

$$N(\text{Brown}, \text{fox}, \text{jumps}) = 2$$

$$N(\text{Brown}, \text{fox}, \text{runs}) = 5$$

$$N(\text{Brown}, \text{fox}, \text{lies}) = 1$$

$$N(\text{Brown}, \text{fox}) = 8$$

N-gram generation model: training

- We need to estimate the probability $p(x_i \mid x_{i-1}, \ldots, x_{i-n})$.

- Let's manually count how many times x_i occurred after $x_{i-1}, \ldots, x_{i-n}$.

$$p(x_i \mid x_{i-1}, \ldots, x_{i-n}) = \frac{p(x_i, x_{i-1}, \ldots, x_{i-n})}{p(x_{i-1}, \ldots, x_{i-n})} = \frac{N(x_i, x_{i-1}, \ldots, x_{i-n})}{N(x_{i-1}, \ldots, x_{i-n})}$$

$N(\text{Brown, fox, jumps}) = 2$

$N(\text{Brown, fox, runs}) = 5$

$N(\text{Brown, fox, lies}) = 1$

$N(\text{Brown, fox}) = 8$

$\longrightarrow$

$p(\text{jumps} \mid \text{Brown, fox}) = \frac{2}{8}$

$p(\text{runs} \mid \text{Brown, fox}) = \frac{5}{8}$

$p(\text{lies} \mid \text{Brown, fox}) = \frac{1}{8}$

N-gram generation model: generation

Generate words one by one according to probabilities.

N-gram generation model: generation

Generate words one by one according to probabilities.

<PAD> <PAD> _

p(Quick | <PAD> <PAD>) = 0.3
p(My | <PAD> <PAD>) = 0.2
p(When | <PAD> <PAD>) = 0.15

N-gram generation model: generation

Generate words one by one according to probabilities.

<PAD> <PAD> Quick _

p(fox | <PAD> B) = 0.34
p(brown | <PAD> B) = 0.23
p(bird | <PAD> B) = 0.09

N-gram generation model: generation

Generate words one by one according to probabilities.

<PAD> <PAD> Quick brown _

p(cat | В лесу) = 0.4
p(fox | В лесу) = 0.23
p(dog | В лесу) = 0.09

N-gram generation model: generation

Generate words one by one according to probabilities.

<PAD> <PAD> Quick brown fox

Advantages of n-gram model

- Texts consist of existing n-grams
- Therefore, the sentences are grammatically correct

- The model is easy to implement and very fast.

Disadvantages of n-gram model

- When generating, it looks only at the last n words
- This results in logically incoherent texts

- As n increases, the probabilities of words are estimated worse

- Due to the large size of the dictionary, many n-grams are very rare

Disadvantages of n-gram model

- When generating, it looks only at the last n words
- This results in logically incoherent texts

- As n increases, the probabilities of words are estimated worse

- Due to the large size of the dictionary, many n-grams are very rare

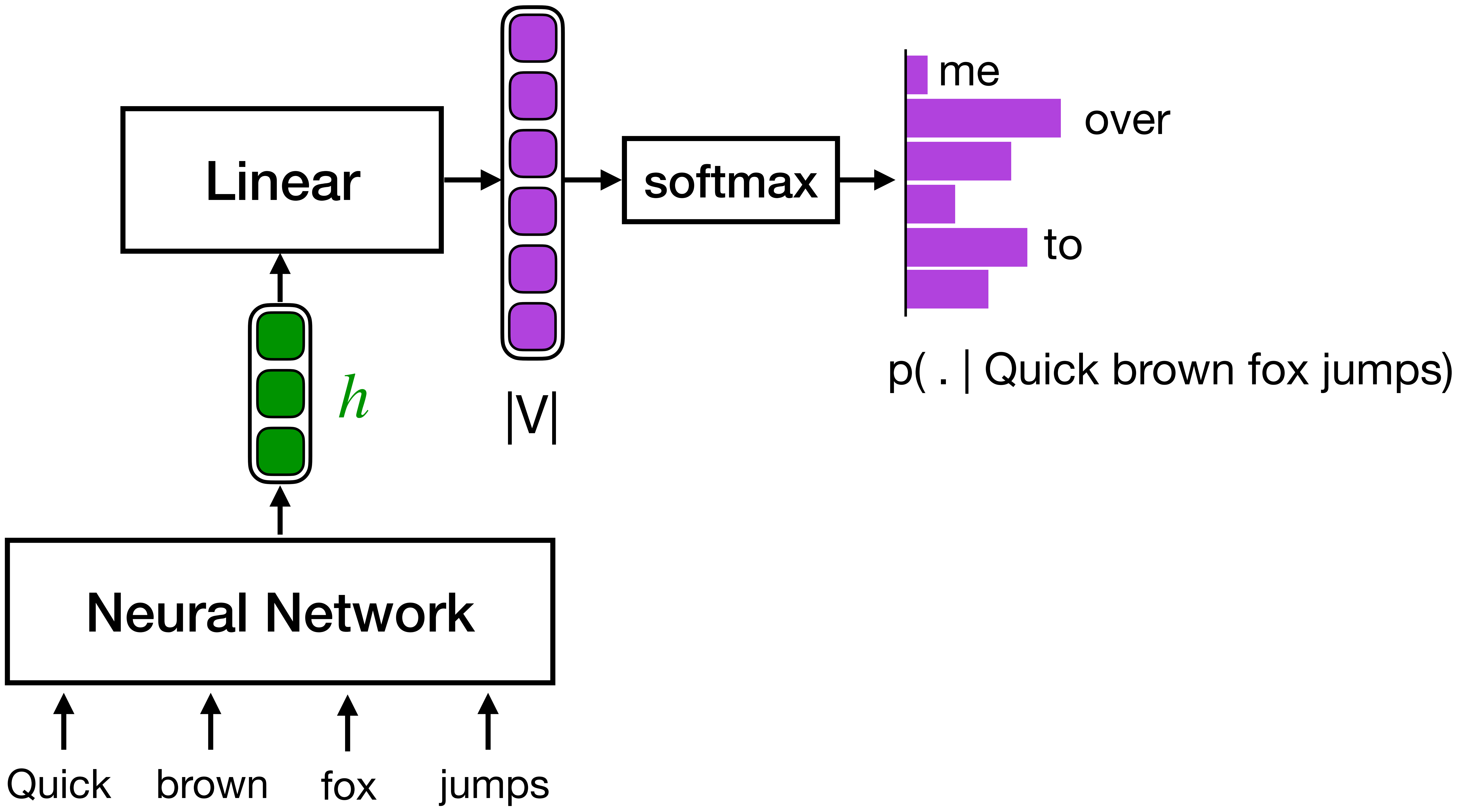
How to reduce the size of the dictionary?

Dictionary size reduction

- Stop word removal
- Lemmatization
- Stemming
- Tokenization: using word parts instead of whole words

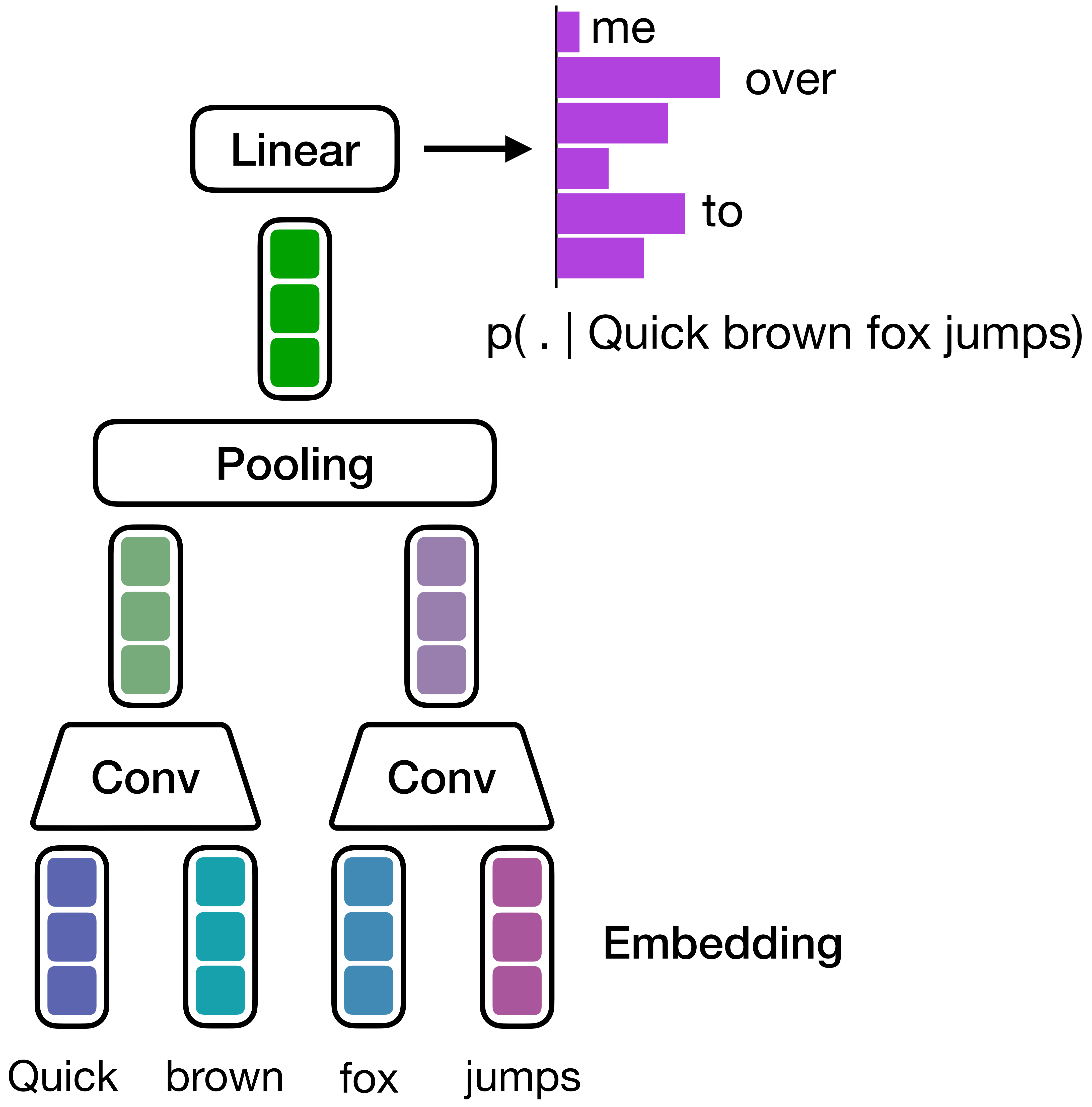
Recurrent Neural Network
(RNN)

Neural networks for text generation



Convolutional neural networks

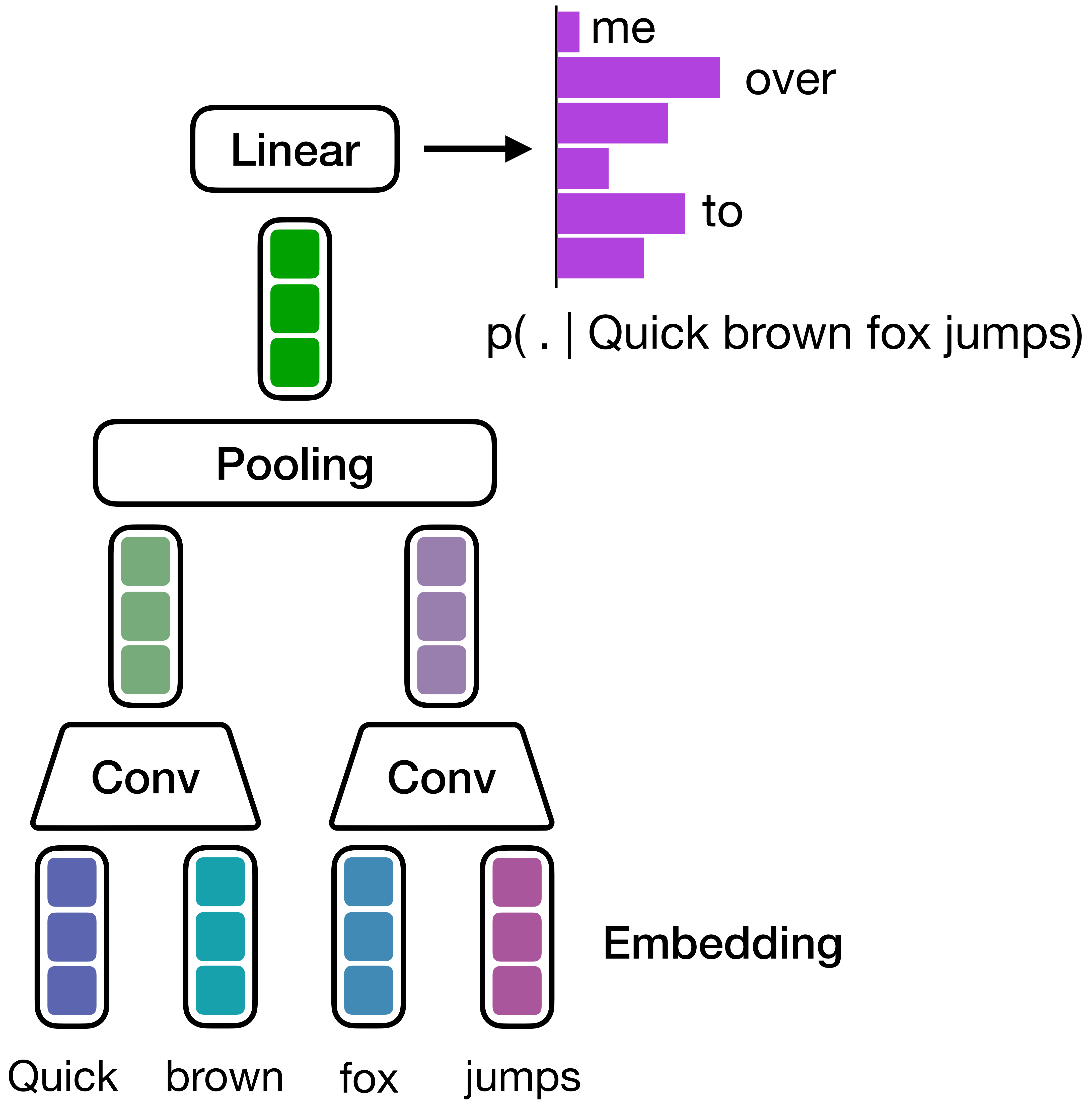
- Can CNN be used for generation?



Convolutional neural networks

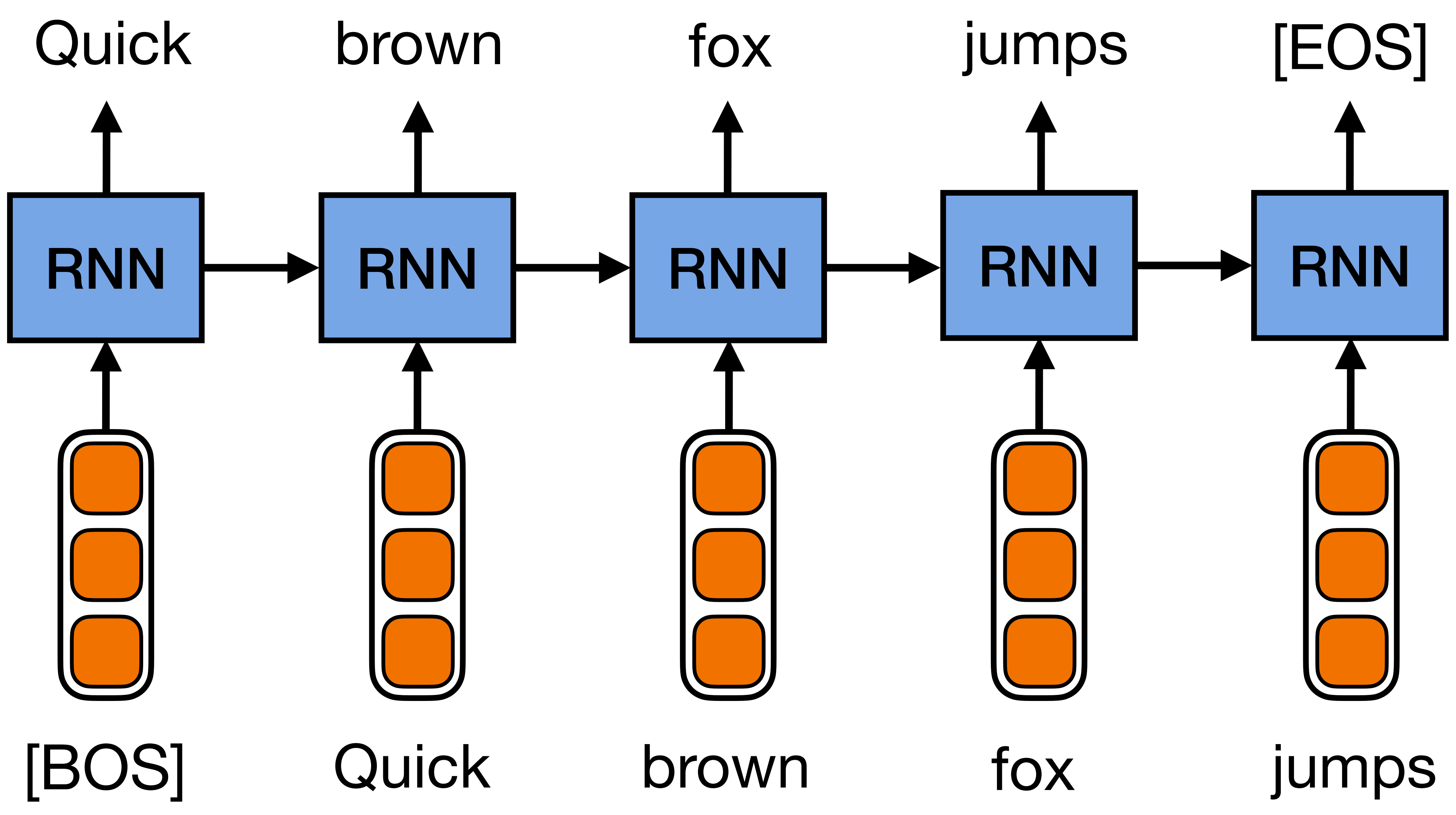
- Can CNN be used for generation?

- Yes, but should not
- Information will blur as text length increases
- Training and generation is not effective



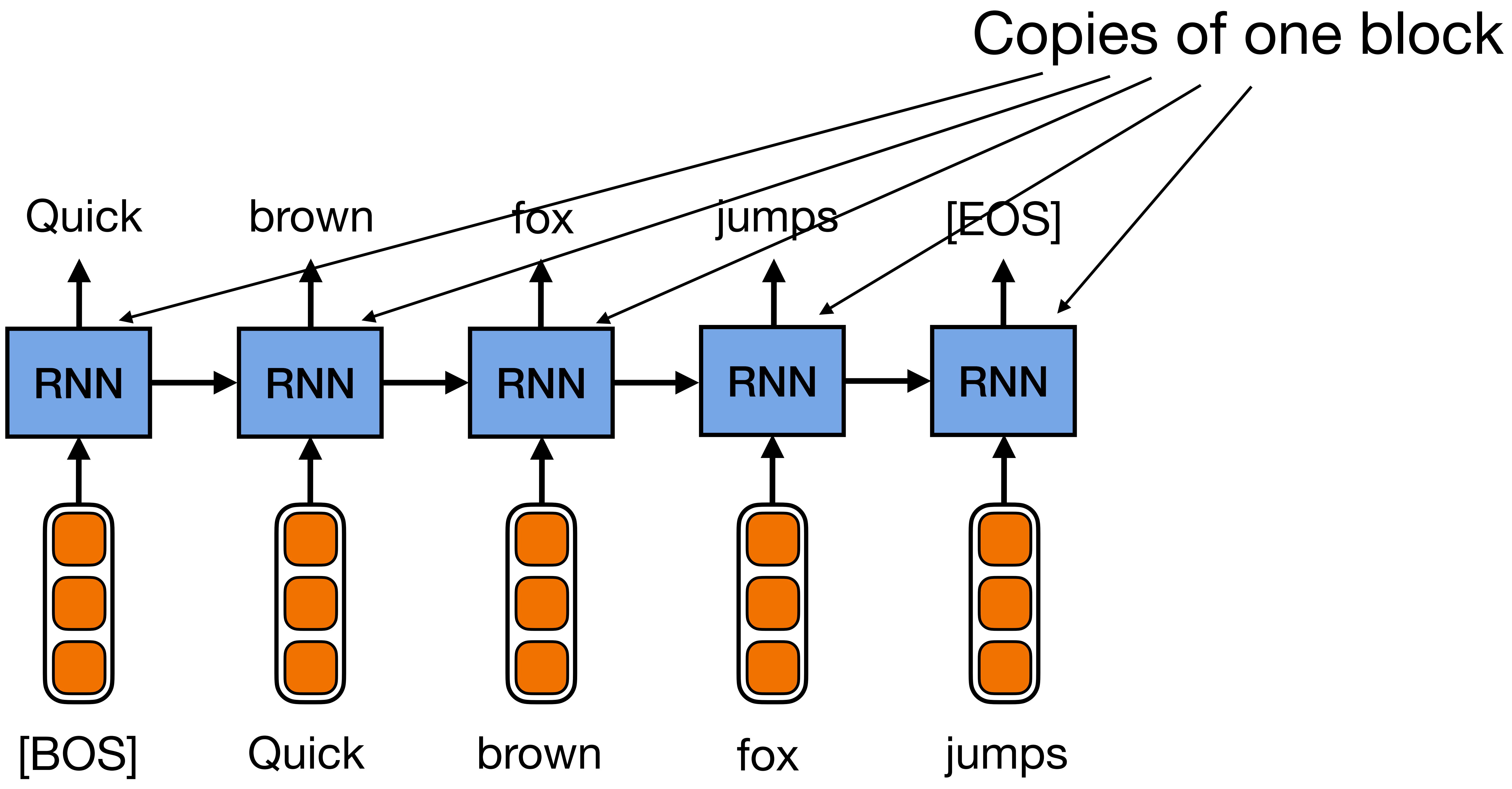
Recurrent neural networks (RNN)

- Designed to work with sequential data
- Each block predicts the next token



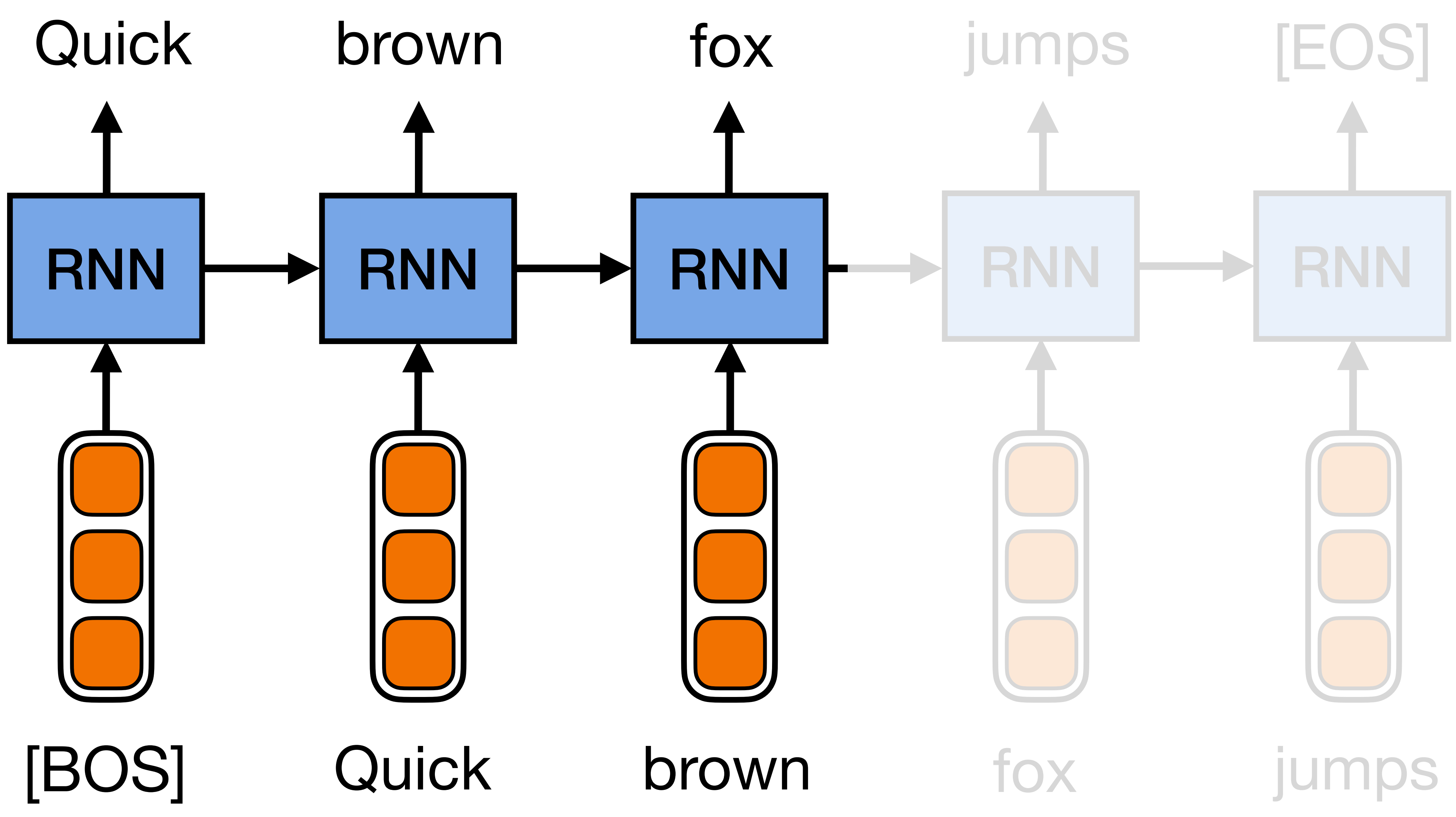
Recurrent neural networks (RNN)

- Designed to work with sequential data
- Each block predicts the next token

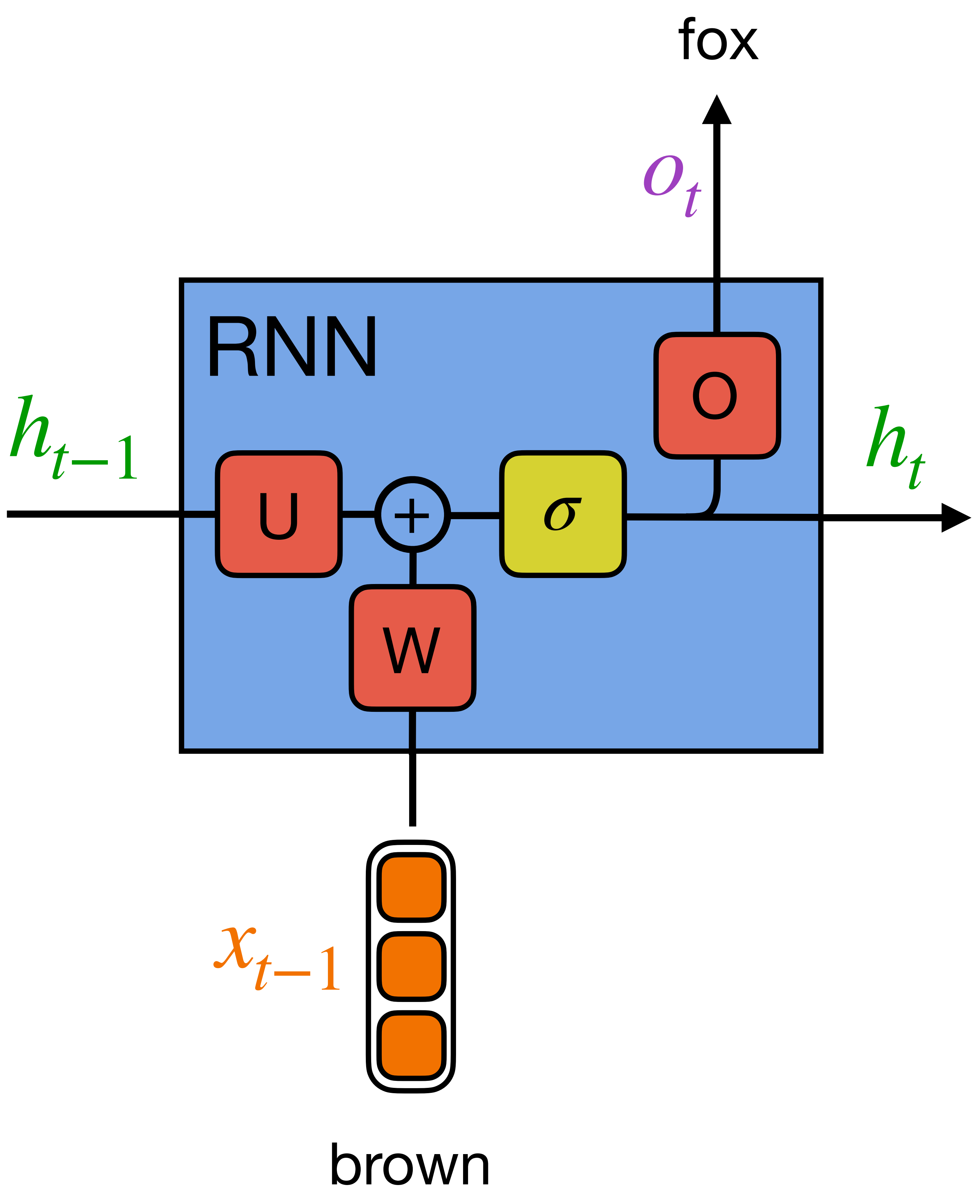


Recurrent neural networks (RNN)

- Designed to work with sequential data
- Each block predicts the next token
- The generation process is intuitive



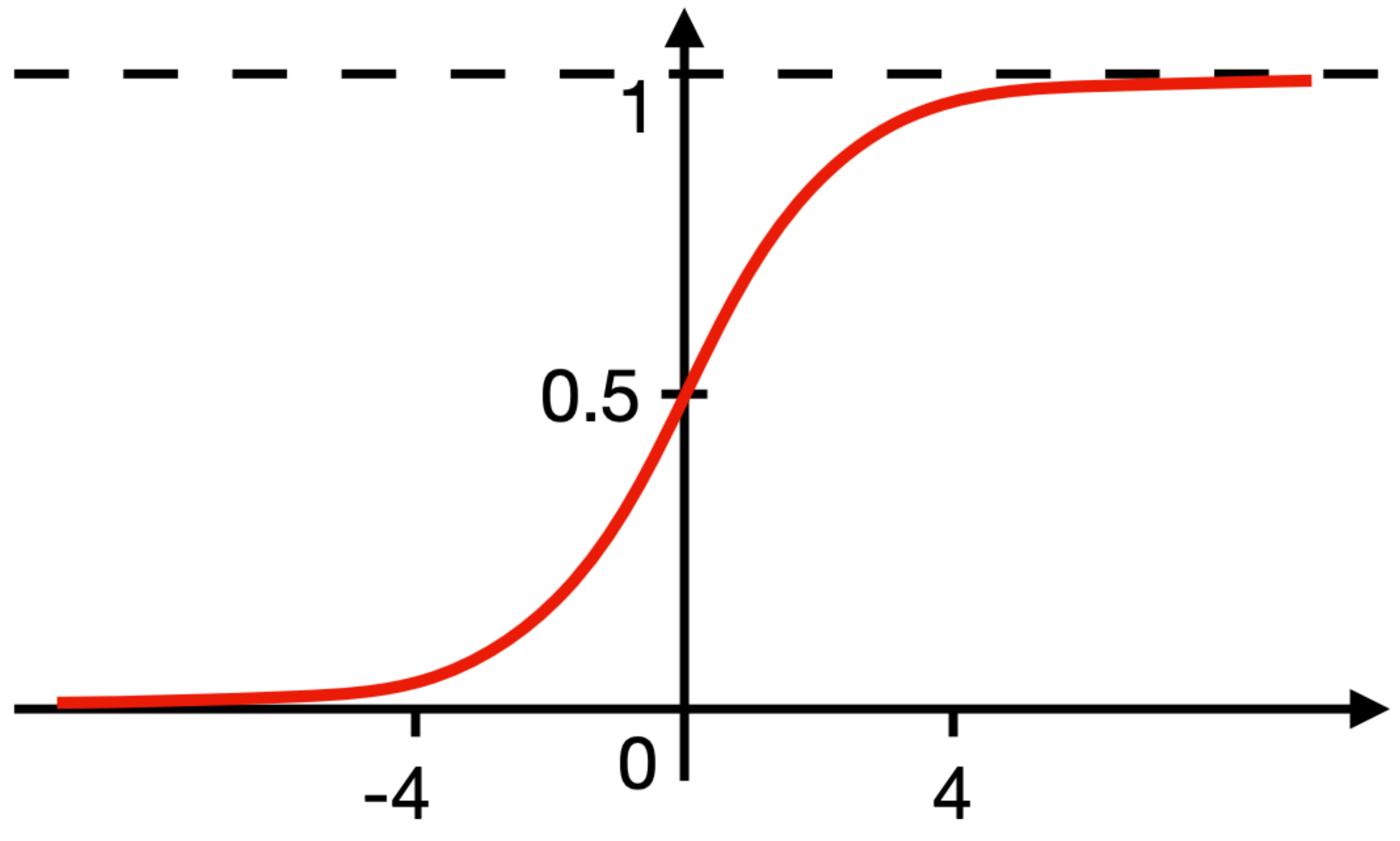
RNN block



$$h_t = \sigma(Wx_{t-1} + Uh_{t-1} + b_h)$$

$$o_t = Oh_t + b_o$$

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$



RNN: training

$$p(x_1, \ldots, x_m) = \prod_{t=1}^{m} p(x_t | x_{<t}) \rightarrow \max$$

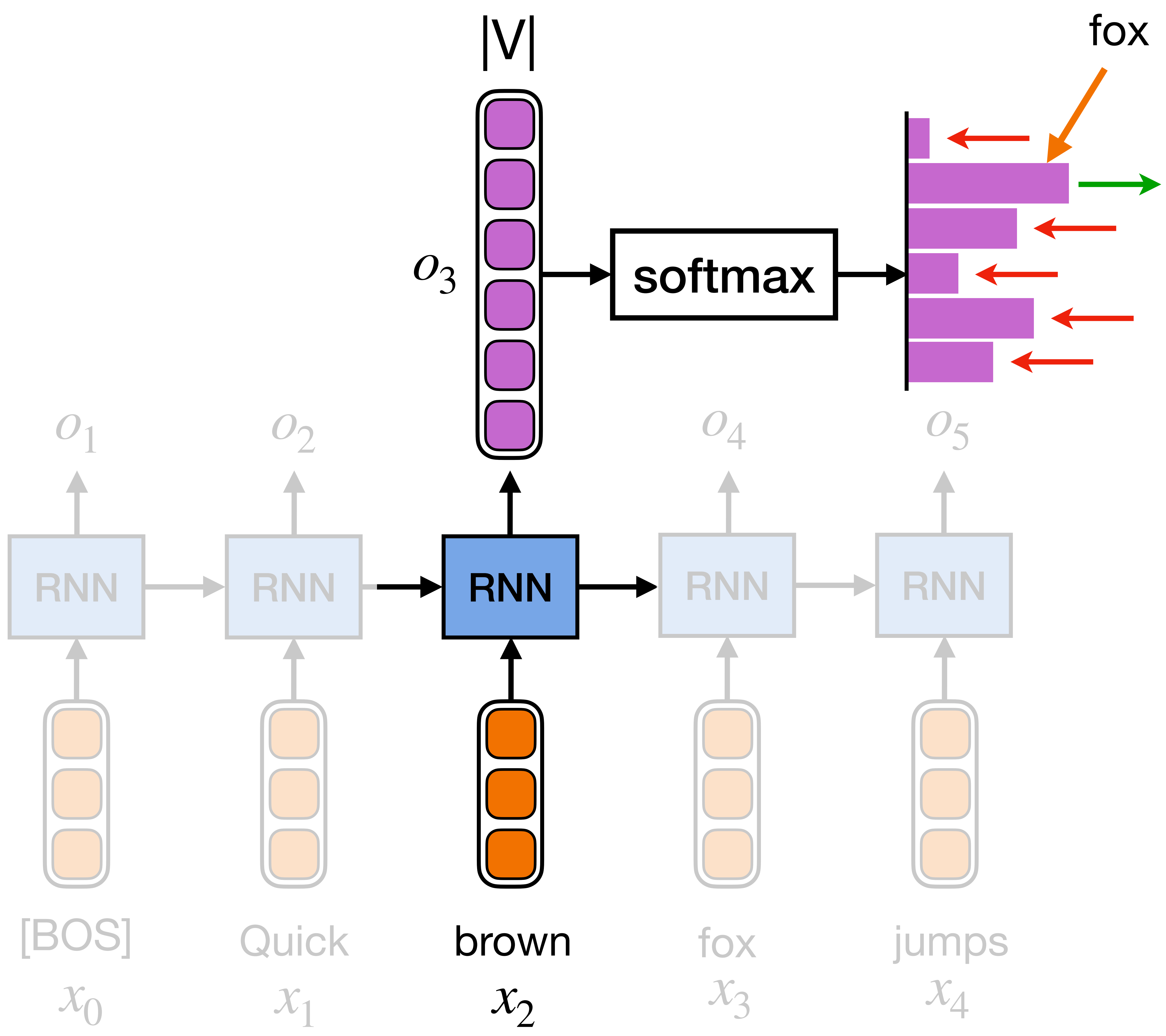
$$p(\,.\,| x_{<t}) = \mathrm{softmax}(\mathrm{o_t})$$

Add logarithm and negation to get the cross-entropy

$$L(x) = -\sum_{t=1}^{m} \log p(x_t | x_{<t}) \rightarrow \min$$

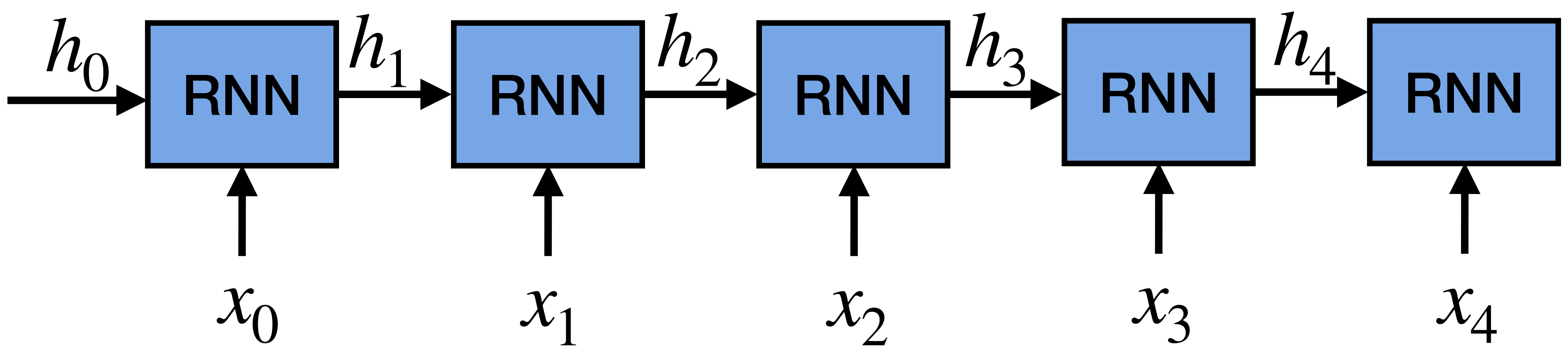
Loss for the whole corpus

$$L(X) = -\frac{1}{|X|} \sum_{x \in X} \sum_{t=1}^{m} \log p(x_t | x_{<t}) \rightarrow \min$$



RNN gradients

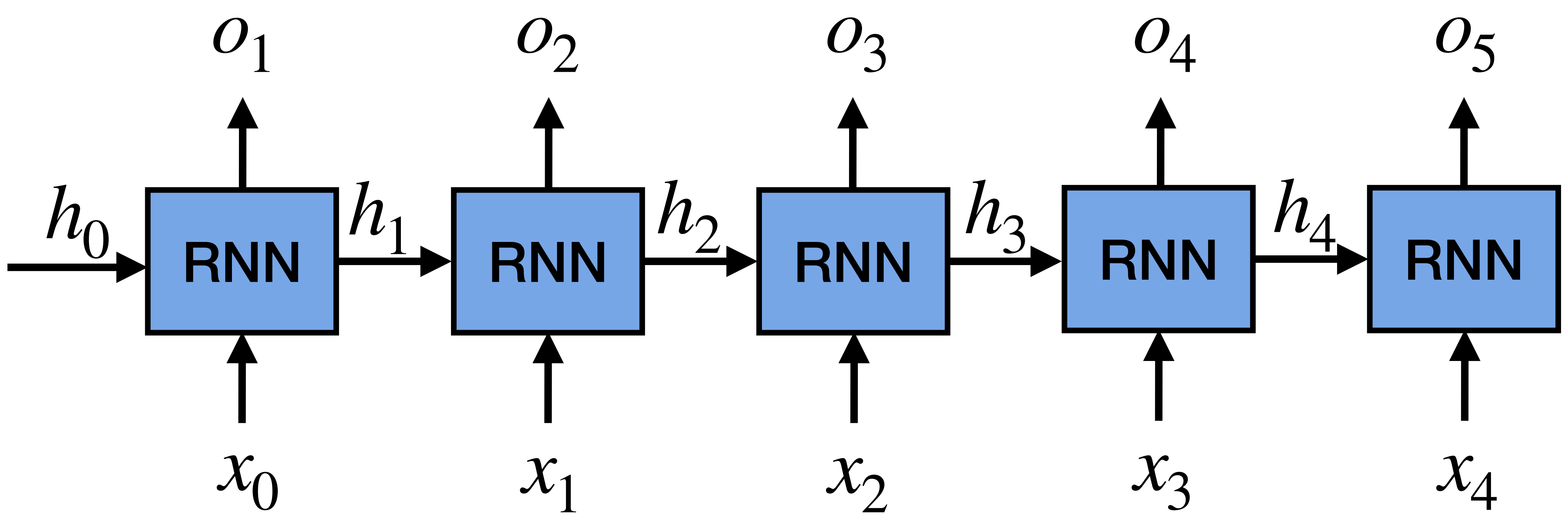
$$h_t = \sigma(Wx_t + Uh_{t-1} + b_h)$$



RNN gradients

$$h_t = \sigma(Wx_t + Uh_{t-1} + b_h)$$

$$o_t = Oh_t + b_o$$

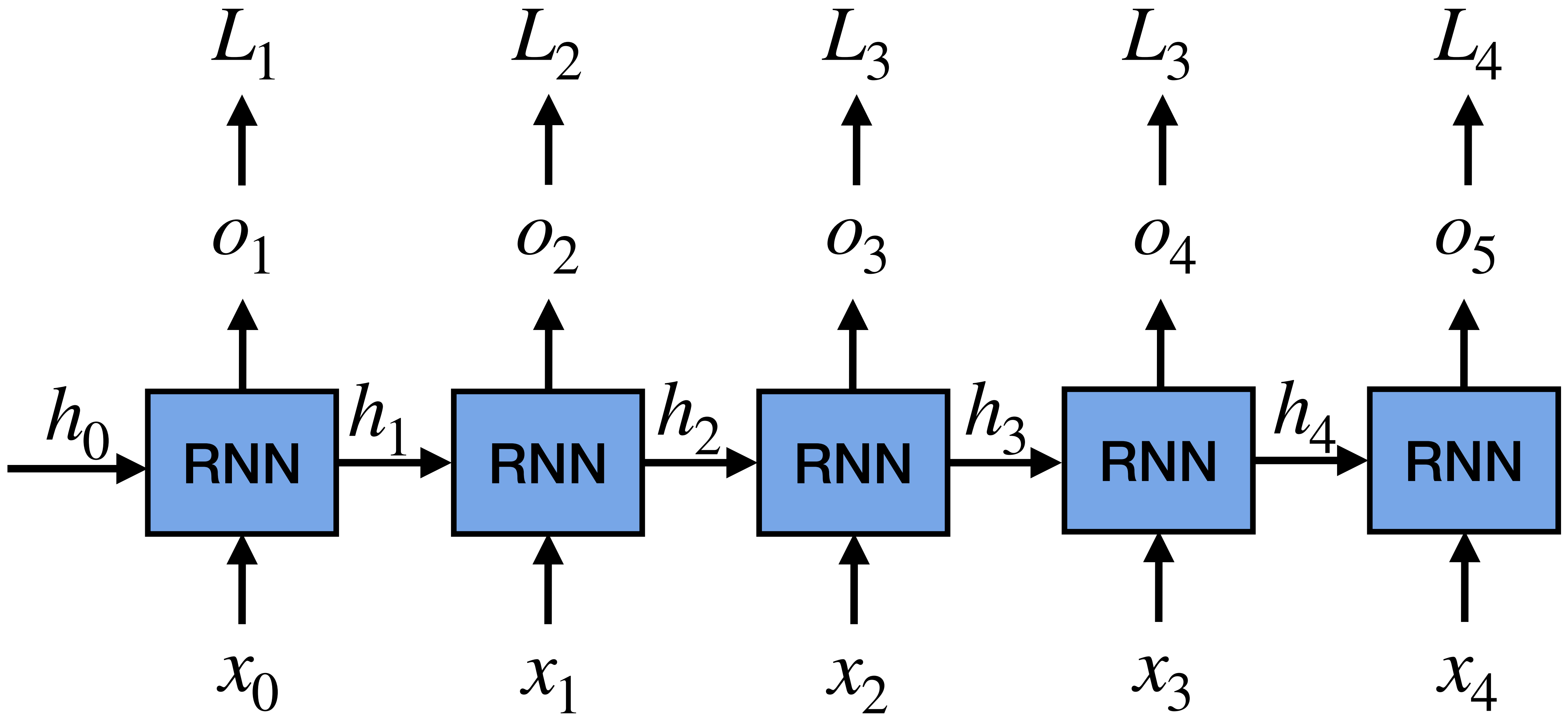


RNN gradients

$$h_t = \sigma(Wx_t + Uh_{t-1} + b_h)$$

$$o_t = Oh_t + b_o$$

$$L_t = -\log p(x_t \,|\, x_{<t}) = -\log \mathrm{softmax}(\mathrm{o_t})_{\mathrm{x_t}}$$



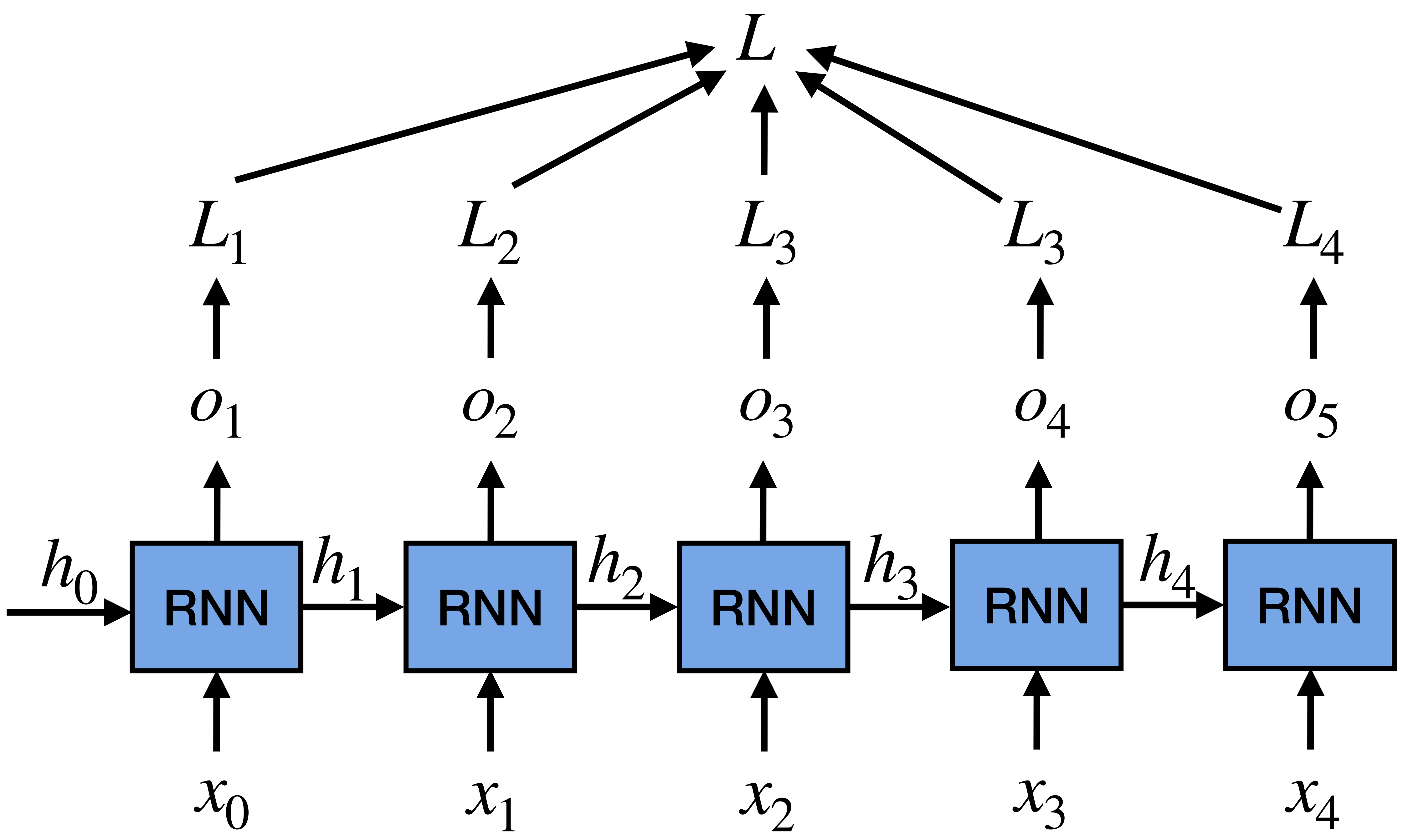
RNN gradients

$$h_t = \sigma(Wx_t + Uh_{t-1} + b_h)$$

$$o_t = Oh_t + b_o$$

$$L_t = -\log p(x_t \,|\, x_{<t}) = -\log \mathrm{softmax}(\mathrm{o_t})_{\mathrm{x_t}}$$

$$L = \sum_{t=1}^{m} L_t$$



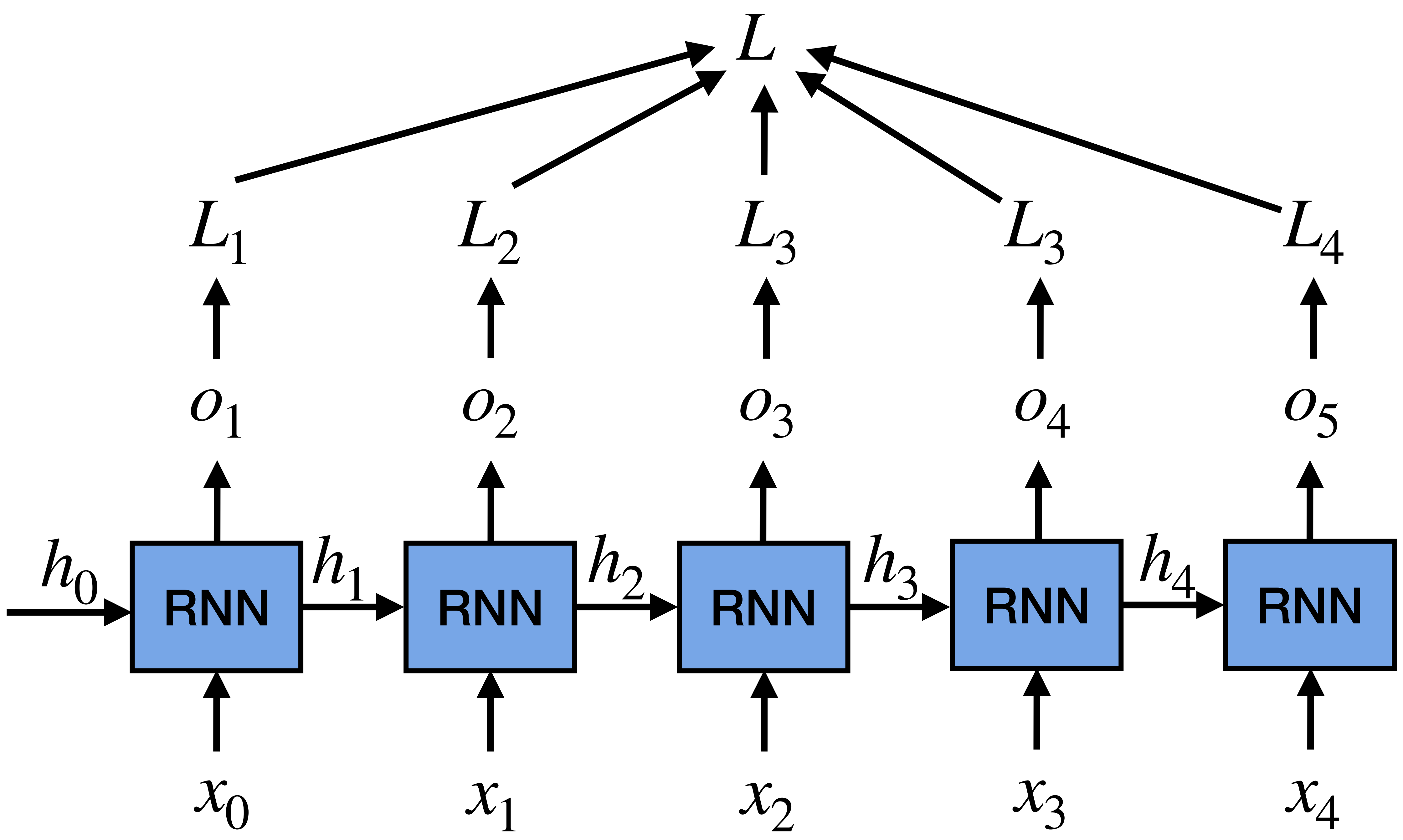
RNN gradients

$$h_t = \sigma(Wx_t + Uh_{t-1} + b_h)$$

$$o_t = Oh_t + b_o$$

$$L_t = -\log p(x_t \mid x_{<t}) = -\log \mathrm{softmax}(\mathrm{o_t})_{\mathrm{x_t}}$$

$$L = \sum_{t=1}^{m} L_t$$



Let's see how gradients behave. It's going to hurt a little.

RNN gradients

$$h_t = \sigma(Wx_t + Uh_{t-1} + b_h)$$

$$o_t = Oh_t + b_o$$

$$L_t = -\log p(x_t \mid x_{<t}) = -\log \mathrm{softmax}(\mathrm{o_t})_{\mathrm{x_t}}$$

$$\boxed{L = \sum_{t=1}^{m} L_t}$$

chain rule

$$\boxed{\frac{dL}{dU} = \sum_{t=1}^{m} \frac{dL_t}{dU}} = \sum_{t=1}^{m} \frac{dL_t}{do_t}\frac{do_t}{dh_t}\frac{dh_t}{dU}$$

RNN gradients

$$\boxed{h_t = \sigma(Wx_t + Uh_{t-1} + b_h)}$$

$$o_t = Oh_t + b_o$$

$$L_t = -\log p(x_t \mid x_{<t}) = -\log \mathrm{softmax}(\mathrm{o_t})_{\mathrm{x_t}}$$

$$L = \sum_{t=1}^{m} L_t$$

$$\frac{dL}{dU} = \sum_{t=1}^{m} \frac{dL_t}{dU} = \sum_{t=1}^{m} \frac{dL_t}{do_t}\frac{do_t}{dh_t}\frac{dh_t}{dU}$$

$$\boxed{\frac{dh_t}{dU}} = \frac{\partial h_t}{\partial U} + \frac{\partial h_t}{\partial h_{t-1}}\boxed{\frac{dh_{t-1}}{dU}}$$

Moving from derivatives to partial derivatives

RNN gradients

$$h_t = \sigma(Wx_t + Uh_{t-1} + b_h)$$

$$o_t = Oh_t + b_o$$

$$L_t = -\log p(x_t \mid x_{<t}) = -\log \mathrm{softmax}(\mathrm{o_t})_{\mathrm{x_t}}$$

$$L = \sum_{t=1}^{m} L_t$$

$$\frac{dL}{dU} = \sum_{t=1}^{m} \frac{dL_t}{dU} = \sum_{t=1}^{m} \frac{dL_t}{do_t}\frac{do_t}{dh_t}\frac{dh_t}{dU}$$

$$\frac{dh_t}{dU} = \frac{\partial h_t}{\partial U} + \frac{\partial h_t}{\partial h_{t-1}}\boxed{\frac{dh_{t-1}}{dU}} =$$

$$= \frac{\partial h_t}{\partial U} + \frac{\partial h_t}{\partial h_{t-1}}\boxed{\left(\frac{\partial h_{t-1}}{\partial U} + \frac{\partial h_{t-1}}{\partial h_{t-2}}\frac{dh_{t-2}}{dU}\right)}$$

RNN gradients

$$h_t = \sigma(Wx_t + Uh_{t-1} + b_h)$$

$$o_t = Oh_t + b_o$$

$$L_t = -\log p(x_t \,|\, x_{<t}) = -\log \mathrm{softmax}(\mathrm{o_t})_{\mathrm{x_t}}$$

$$L = \sum_{t=1}^{m} L_t$$

$$\frac{dL}{dU} = \sum_{t=1}^{m} \frac{dL_t}{dU} = \sum_{t=1}^{m} \frac{dL_t}{do_t}\frac{do_t}{dh_t}\frac{dh_t}{dU}$$

$$\begin{aligned}
\frac{dh_t}{dU} &= \frac{\partial h_t}{\partial U} + \frac{\partial h_t}{\partial h_{t-1}}\frac{dh_{t-1}}{dU} = \\
&= \frac{\partial h_t}{\partial U} + \frac{\partial h_t}{\partial h_{t-1}}\left(\frac{\partial h_{t-1}}{\partial U} + \frac{\partial h_{t-1}}{\partial h_{t-2}}\frac{dh_{t-2}}{dU}\right) = \\
&= \frac{\partial h_t}{\partial U} + \frac{\partial h_t}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial U} + \frac{\partial h_t}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial h_{t-2}}\frac{dh_{t-2}}{dU}
\end{aligned}$$

RNN gradients

$$h_t = \sigma(Wx_t + Uh_{t-1} + b_h)$$

$$o_t = Oh_t + b_o$$

$$L_t = -\log p(x_t \mid x_{<t}) = -\log \mathrm{softmax}(\mathrm{o_t})_{\mathrm{x_t}}$$

$$L = \sum_{t=1}^{m} L_t$$

$$\frac{dL}{dU} = \sum_{t=1}^{m} \frac{dL_t}{dU} = \sum_{t=1}^{m} \frac{dL_t}{do_t}\frac{do_t}{dh_t}\frac{dh_t}{dU}$$

$$\frac{dh_t}{dU} = \frac{\partial h_t}{\partial U} + \frac{\partial h_t}{\partial h_{t-1}}\frac{dh_{t-1}}{dU} =$$

$$= \frac{\partial h_t}{\partial U} + \frac{\partial h_t}{\partial h_{t-1}}\left(\frac{\partial h_{t-1}}{\partial U} + \frac{\partial h_{t-1}}{\partial h_{t-2}}\frac{dh_{t-2}}{dU}\right) =$$

$$= \frac{\partial h_t}{\partial U} + \frac{\partial h_t}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial U} + \frac{\partial h_t}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial h_{t-2}}\frac{dh_{t-2}}{dU} =$$

$$= \sum_{k=1}^{t}\left(\prod_{j=k+1}^{t}\frac{\partial h_j}{\partial h_{j-1}}\right)\frac{\partial h_k}{\partial U}$$

RNN gradients

$$h_t = \sigma(Wx_t + Uh_{t-1} + b_h)$$

$$o_t = Oh_t + b_o$$

$$L_t = -\log p(x_t \mid x_{<t}) = -\log \mathrm{softmax}(\mathrm{o_t})_{\mathrm{x_t}}$$

$$L = \sum_{t=1}^{m} L_t$$

$$\frac{dL}{dU} = \sum_{t=1}^{m} \frac{dL_t}{dU} = \sum_{t=1}^{m} \frac{dL_t}{do_t}\frac{do_t}{dh_t}\frac{dh_t}{dU}$$

$$\begin{aligned}
\frac{dh_t}{dU} &= \frac{\partial h_t}{\partial U} + \frac{\partial h_t}{\partial h_{t-1}}\frac{dh_{t-1}}{dU} = \\
&= \frac{\partial h_t}{\partial U} + \frac{\partial h_t}{\partial h_{t-1}}\left(\frac{\partial h_{t-1}}{\partial U} + \frac{\partial h_{t-1}}{\partial h_{t-2}}\frac{dh_{t-2}}{dU}\right) = \\
&= \frac{\partial h_t}{\partial U} + \frac{\partial h_t}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial U} + \frac{\partial h_t}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial h_{t-2}}\frac{dh_{t-2}}{dU} = \\
&= \sum_{k=1}^{t}\left(\prod_{j=k+1}^{t}\frac{\partial h_j}{\partial h_{j-1}}\right)\frac{\partial h_k}{\partial U}
\end{aligned}$$

$$\frac{dL}{dU} = \sum_{t=1}^{m} \frac{dL_t}{do_t}\frac{do_t}{dh_t}\left[\sum_{k=1}^{t}\left(\prod_{j=k+1}^{t}\frac{\partial h_j}{\partial h_{j-1}}\right)\frac{\partial h_k}{\partial U}\right]$$

Gradient explosion

$$\frac{dL}{dU} = \sum_{t=1}^{m} \frac{dL_t}{do_t} \frac{do_t}{dh_t} \left[\sum_{k=1}^{t} \underbrace{\left(\prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}} \right)}_{} \frac{\partial h_k}{\partial U} \right]$$

Chain of derivatives multiplication

$$\left\| \frac{\partial h_j}{\partial h_{j-1}} \right\| > 1$$

- Gradient $\frac{dL}{dU}$ explodes
- Model diverges, NaNs in weights

Gradient explosion

$$\frac{dL}{dU} = \sum_{t=1}^{m} \frac{dL_t}{do_t} \frac{do_t}{dh_t} \left[\sum_{k=1}^{t} \underbrace{\left(\prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}} \right)}_{\text{Chain of derivatives multiplication}} \frac{\partial h_k}{\partial U} \right]$$

$$\left\| \frac{\partial h_j}{\partial h_{j-1}} \right\| > 1$$

- Gradient $\frac{dL}{dU}$ explodes
- Model diverges, NaNs in weights

Solutions:

- Regularization
- Reducing learning rate
- Gradient clipping

Gradient explosion

$$\frac{dL}{dU} = \sum_{t=1}^{m} \frac{dL_t}{do_t}\frac{do_t}{dh_t}\left[\sum_{k=1}^{t}\underbrace{\left(\prod_{j=k+1}^{t}\frac{\partial h_j}{\partial h_{j-1}}\right)}\frac{\partial h_k}{\partial U}\right]$$

Chain of derivatives multiplication

$$\left\| \frac{\partial h_j}{\partial h_{j-1}} \right\| > 1$$

- Gradient $\frac{dL}{dU}$ explodes
- Model diverges, NaNs in weights

Solutions:

- Regularization
- Reducing learning rate
- Gradient clipping

1. $g \leftarrow \min\left(1, \frac{\text{max norm}}{\|g\|}\right) \cdot g$ Correct way

2. $g \leftarrow \mathrm{clip}(\mathrm{g}, -\mathrm{C}, \mathrm{C})$ Lazy way (changes gradient direction)

Gradient vanishing

$$\frac{dL}{dU} = \sum_{t=1}^{m} \frac{dL_t}{do_t} \frac{do_t}{dh_t} \left[\sum_{k=1}^{t} \left(\prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}} \right) \frac{\partial h_k}{\partial U} \right]$$

$$\left\| \frac{\partial h_j}{\partial h_{j-1}} \right\| < 1$$

- Gradient vanishes
- The model stops training
- The model does not capture distant dependencies!

Gradient vanishing

$$\frac{dL}{dU} = \sum_{t=1}^{m} \frac{dL_t}{do_t}\frac{do_t}{dh_t}\left[\sum_{k=1}^{t}\left(\prod_{j=k+1}^{t}\frac{\partial h_j}{\partial h_{j-1}}\right)\frac{\partial h_k}{\partial U}\right]$$

$$\left\| \frac{\partial h_j}{\partial h_{j-1}} \right\| < 1$$

- Gradient vanishes
- The model stops training
- The model does not capture distant dependencies!

Gradient decay is a common problem with RNNs.

It cannot be fixed with tricks.

Why gradient vanishes?

$$h_j = \sigma(\underbrace{Wx_j + Uh_{j-1} + b_h}_{z_j})$$

$$\frac{\partial h_j}{\partial h_{j-1}} = \frac{\partial \sigma(z_j)}{\partial h_{j-1}} = \frac{\partial \sigma(z_j)}{\partial z_j} \frac{\partial z_j}{\partial h_{j-1}}$$

Why gradient vanishes?

$$h_j = \sigma(\underbrace{Wx_j + Uh_{j-1} + b_h}_{z_j})$$

Element-wise multiplication

$$\frac{\partial h_j}{\partial h_{j-1}} = \frac{\partial \sigma(z_j)}{\partial h_{j-1}} = \frac{\partial \sigma(z_j)}{\partial z_j}\frac{\partial z_j}{\partial h_{j-1}} = \Big(\sigma(z_j) \odot (1 - \sigma(z_j))\Big)U$$

Why gradient vanishes?

$$h_j = \sigma(\underbrace{Wx_j + Uh_{j-1} + b_h}_{z_j})$$

Element-wise multiplication

$$\frac{\partial h_j}{\partial h_{j-1}} = \frac{\partial \sigma(z_j)}{\partial h_{j-1}} = \frac{\partial \sigma(z_j)}{\partial z_j} \frac{\partial z_j}{\partial h_{j-1}} = \Big(\sigma(z_j) \odot (1 - \sigma(z_j))\Big) U$$

Let's look at the spectral norm

$$\left\| \frac{\partial h_j}{\partial h_{j-1}} \right\| \leq \underbrace{\|\sigma(z_j) \odot (1 - \sigma(z_j))\|}_{< 1} \cdot \|U\|$$

т. к. $\sigma(z) \in [0,1]$

Why gradient vanishes?

$$h_j = \sigma(\underbrace{Wx_j + Uh_{j-1} + b_h}_{z_j})$$

Element-wise multiplication

$$\frac{\partial h_j}{\partial h_{j-1}} = \frac{\partial \sigma(z_j)}{\partial h_{j-1}} = \frac{\partial \sigma(z_j)}{\partial z_j}\frac{\partial z_j}{\partial h_{j-1}} = \Big(\sigma(z_j) \odot (1 - \sigma(z_j))\Big)U$$

Let's look at the spectral norm

$$\left\| \frac{\partial h_j}{\partial h_{j-1}} \right\| \le \underbrace{\|\sigma(z_j) \odot (1 - \sigma(z_j))\|}_{< 1} \cdot \|U\|$$

т. к. $\sigma(z) \in [0,1]$

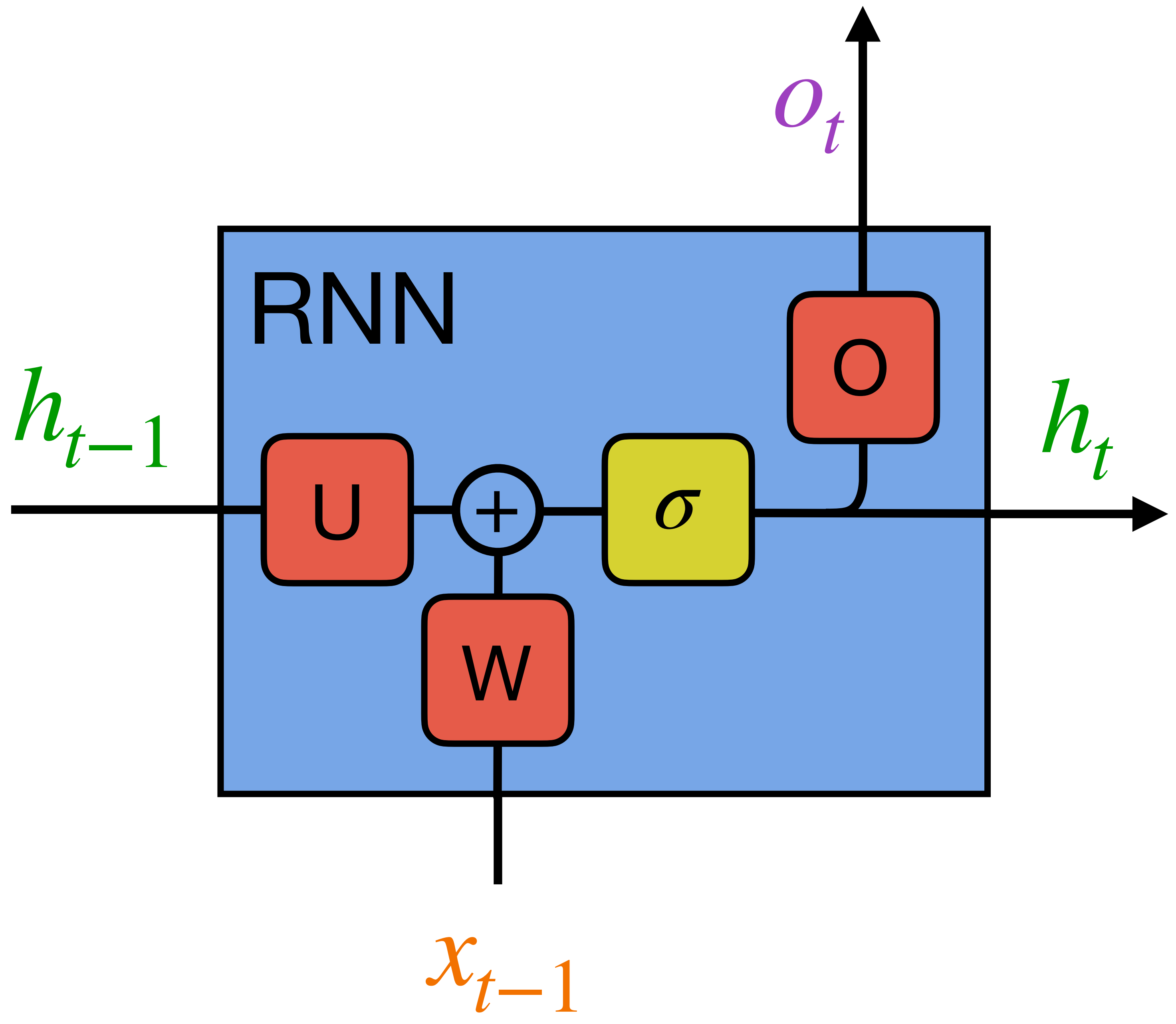
If U is orthogonal, then

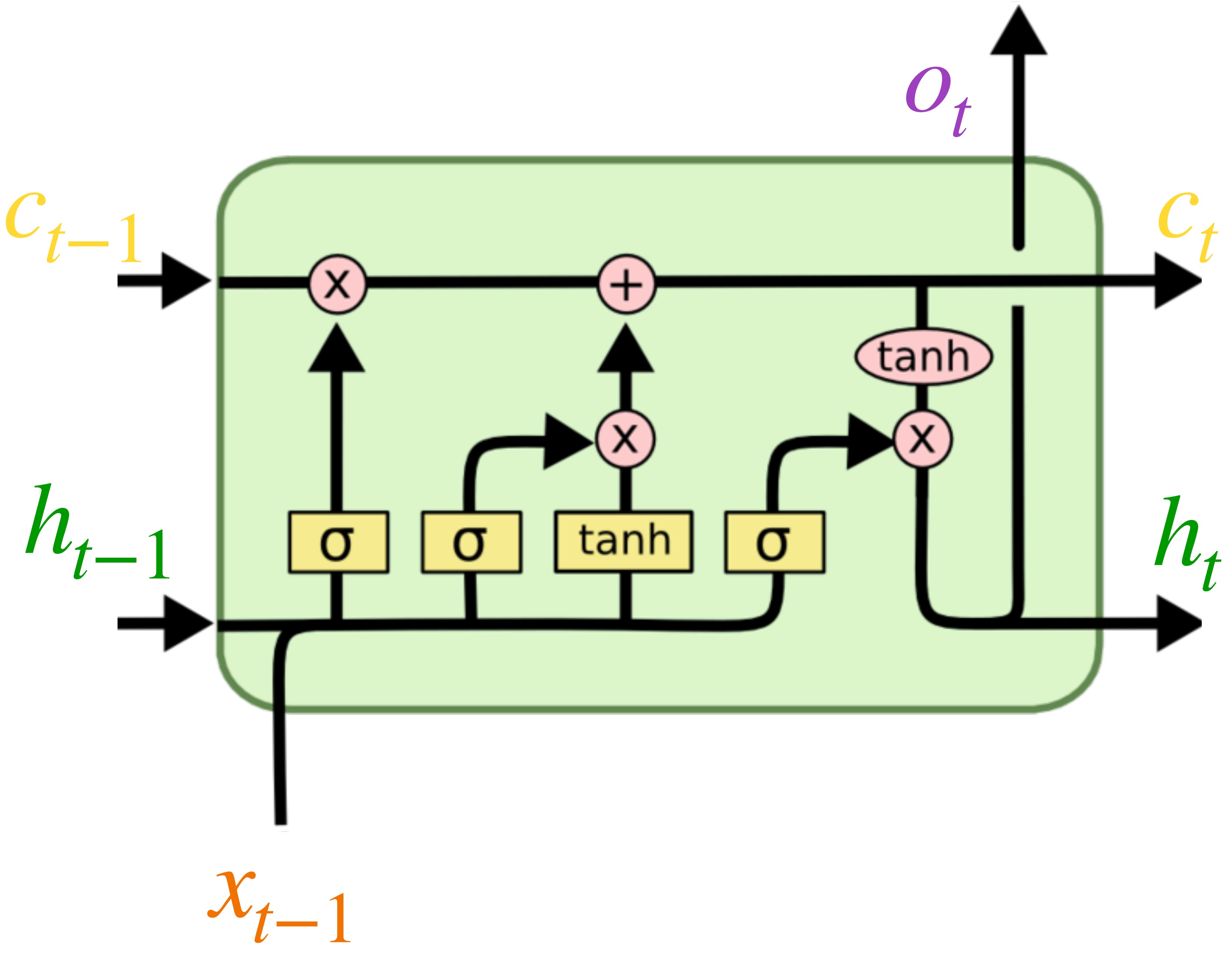
$$\left\| \frac{\partial h_j}{\partial h_{j-1}} \right\| \le \|\sigma(z_j) \odot (1 - \sigma(z_j))\| < 1$$

Long shot-term memory (LSTM)

A memory cell c_t is added to LSTM

Thanks to it, the model does not forget old information.

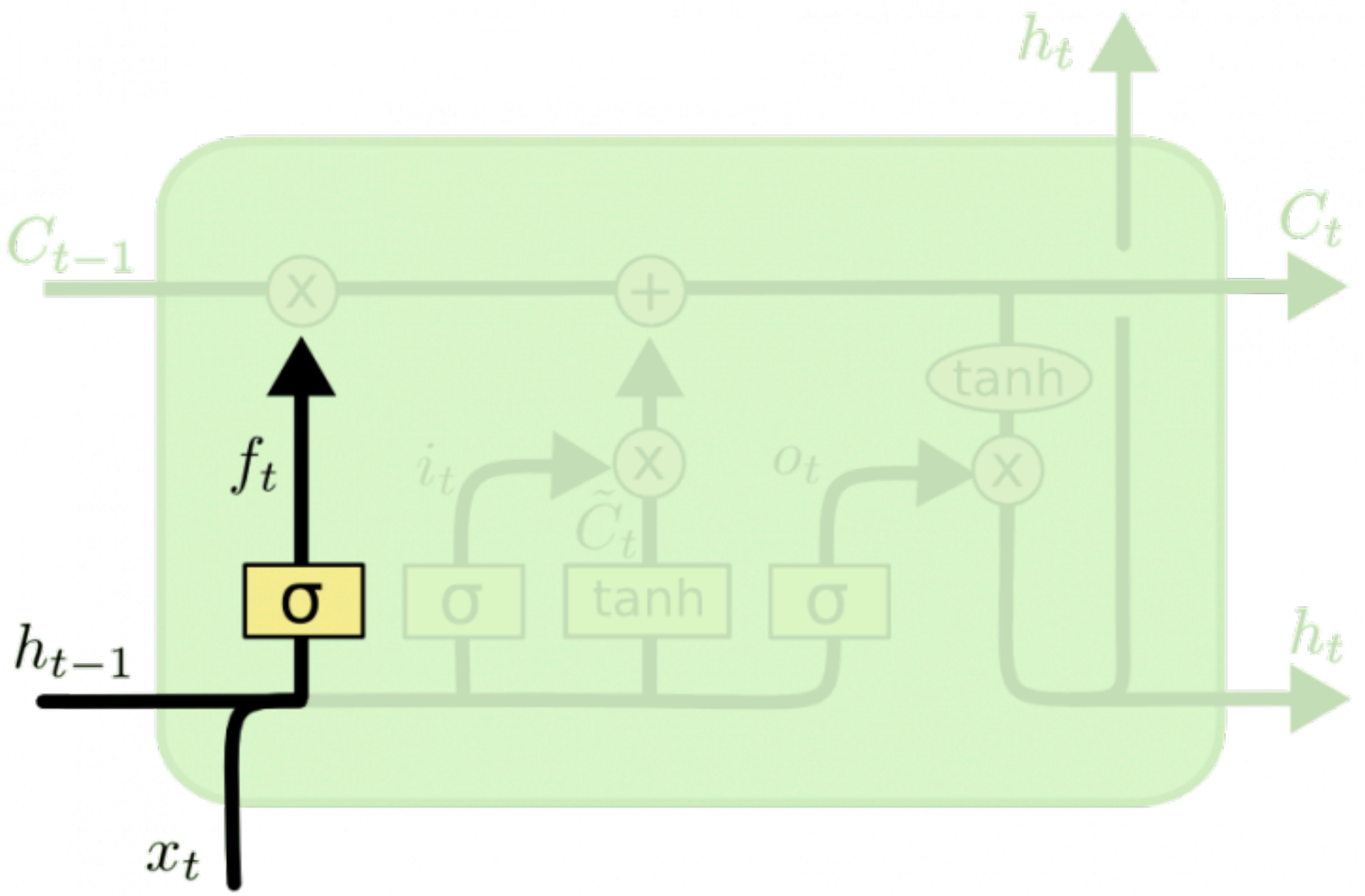




No activations are applied to c_t. Gradients don't vanish

LSTM: forget gate

Controls what information should be forgotten and what should be left.

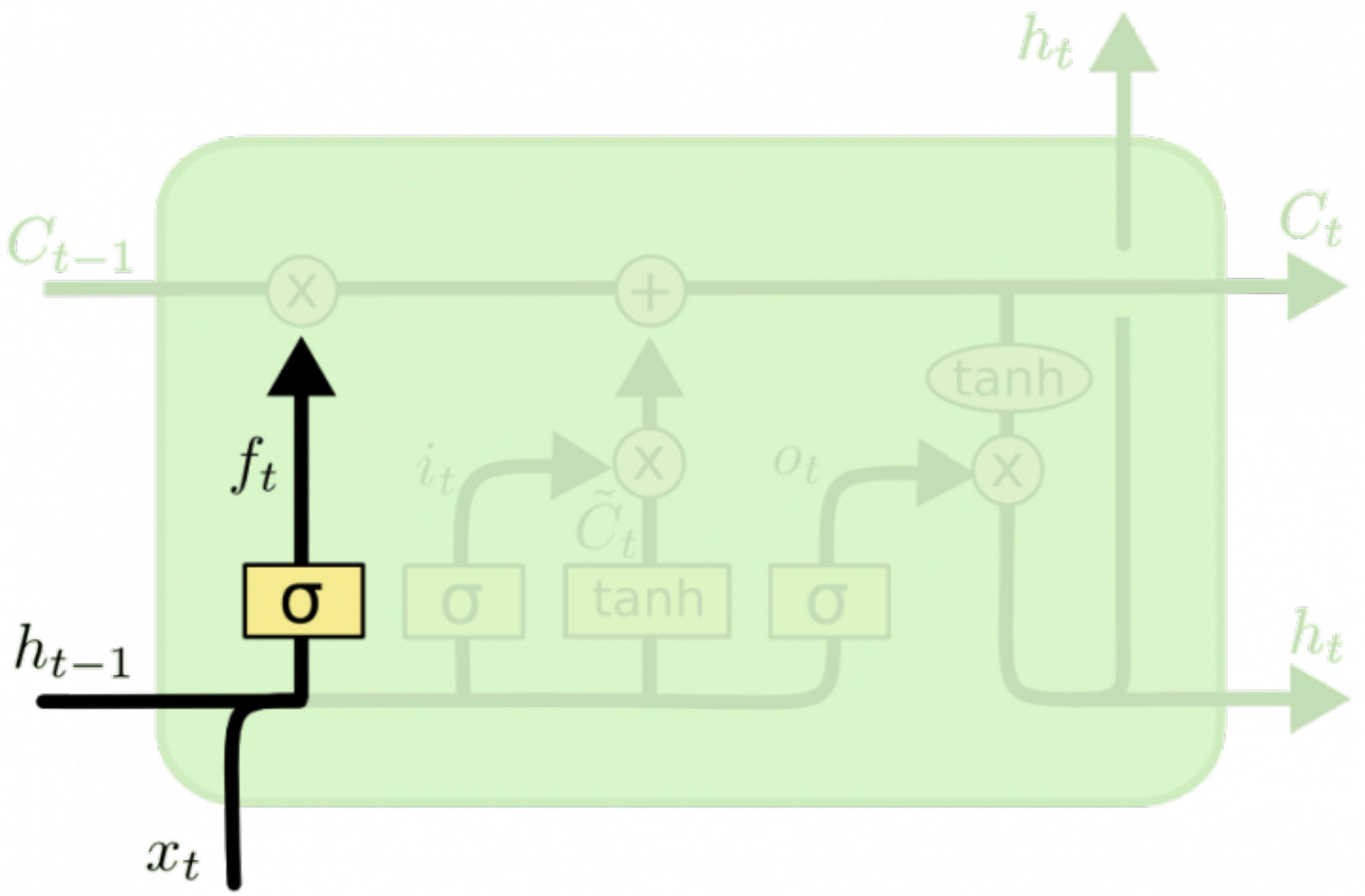


$$f_t = \sigma(W_f x_{t-1} + U_f h_{t-1} + b_f)$$

$$f_t \in [0,1]$$

LSTM: forget gate

Controls what information should be forgotten and what should be left.



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

$$f_t \in [0,1]$$

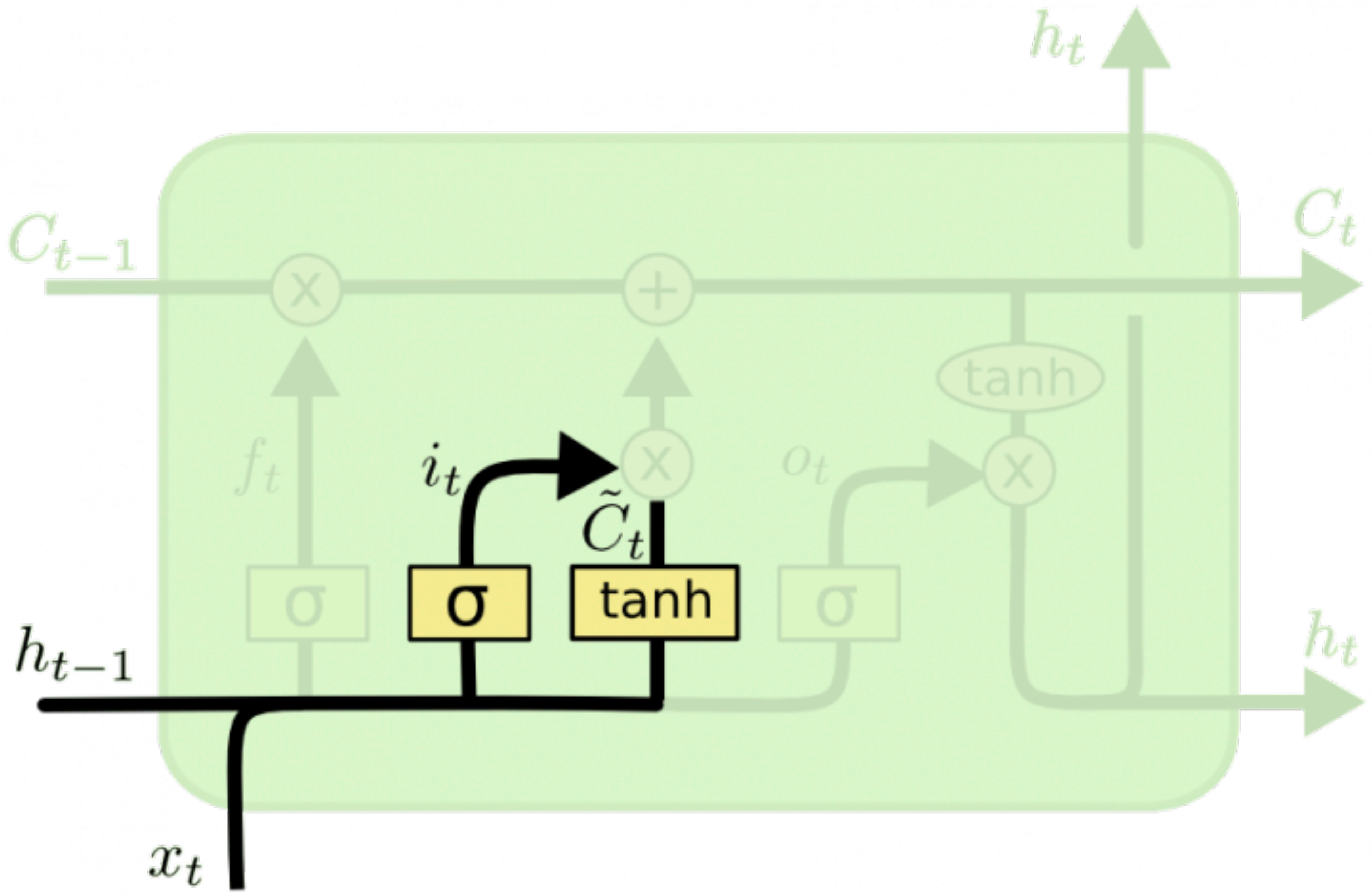
The food was tasty and we were not disappointed.

x_3 – marker word, we can forget everything before it

x_7 – негативный маркер, но до него идет "не". Знаем об этом из h_7.

LSTM: input gate

Controls what information should be added to the memory cell c_t



Because of i_t we can avoid adding

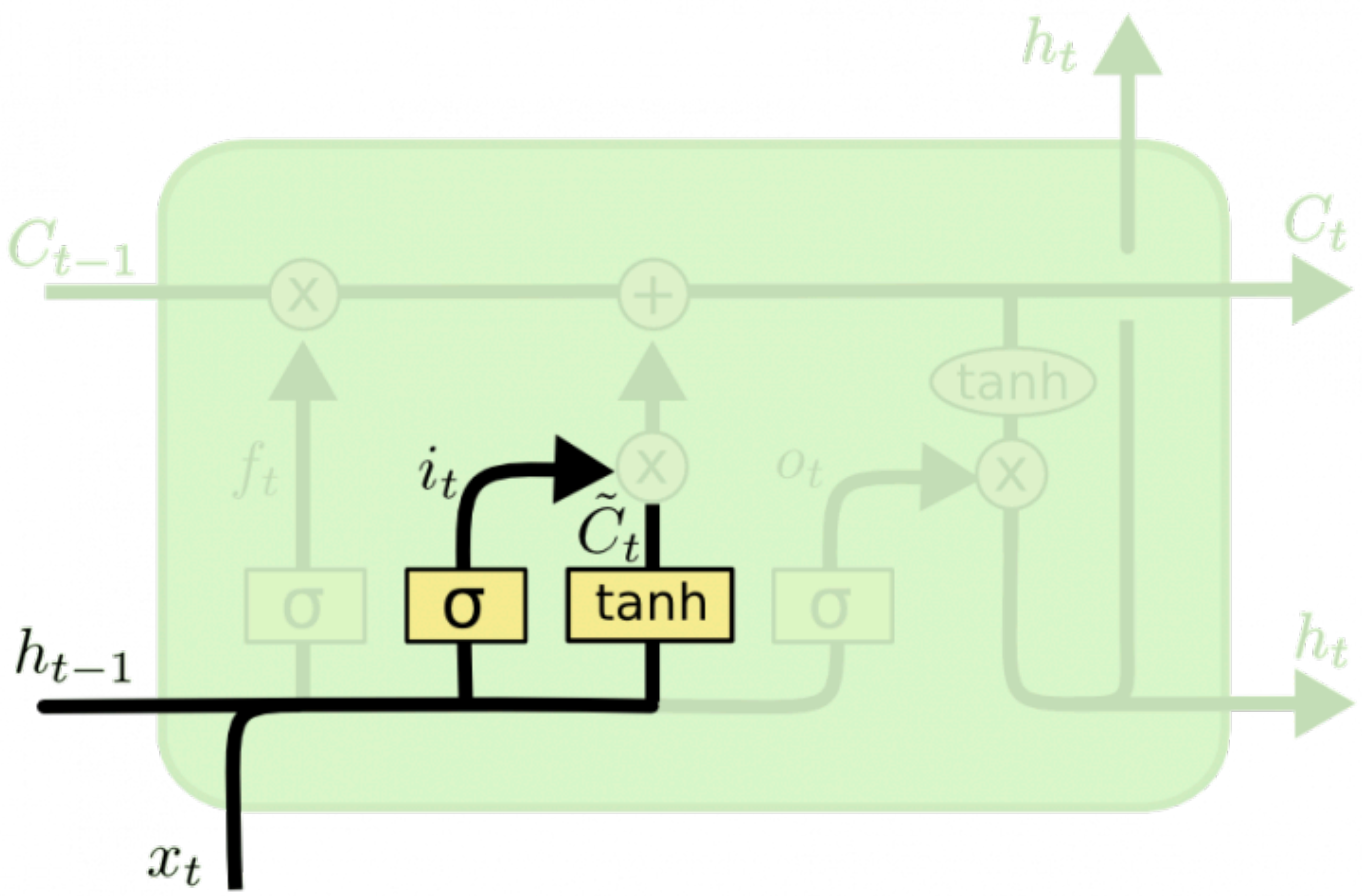
$$i_t = \sigma(W_i x_{t-1} + U_i h_{t-1} + b_i)$$

$$i_t \in [0,1]$$

$$\tilde{c}_t = \tanh(W_i x_{t-1} + U_i h_{t-1} + b_i)$$

LSTM: input gate

Controls what information should be added to the memory cell c_t



Because of i_t we can avoid adding

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

$$i_t \in [0,1]$$

$$\tilde{c}_t = \tanh(W_i x_{t-1} + U_i h_{t-1} + b_i)$$

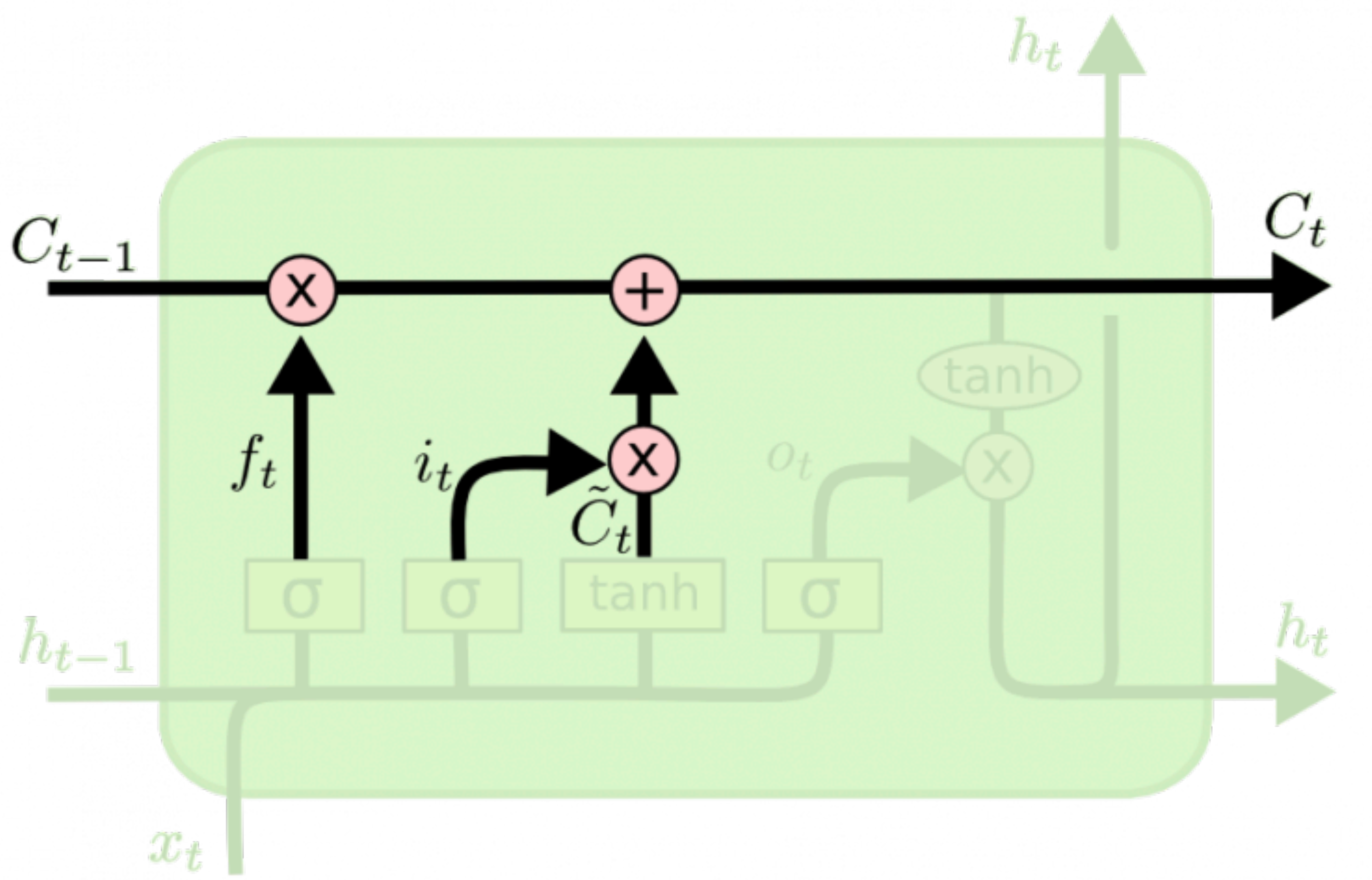
The food was tasty and we were not disappointed.

x_3 – marker word. Remember that the class is positive.

x_5 – negative marker, but before it comes "not". We know about this from h_t

LSTM: memory update

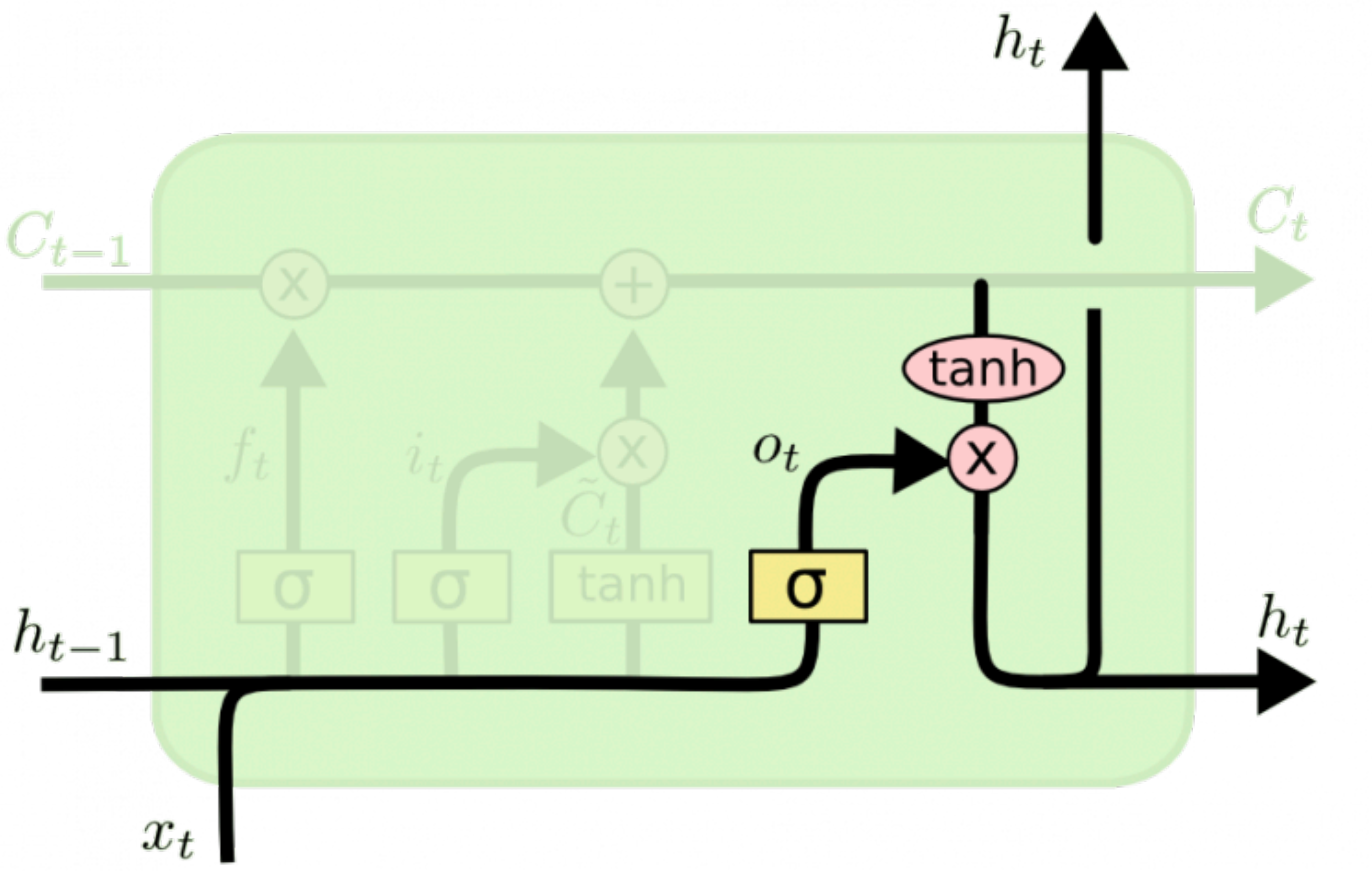
Remove unnecessary information and add new information.



$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$

LSTM: output gate

Controls the output of the current step.

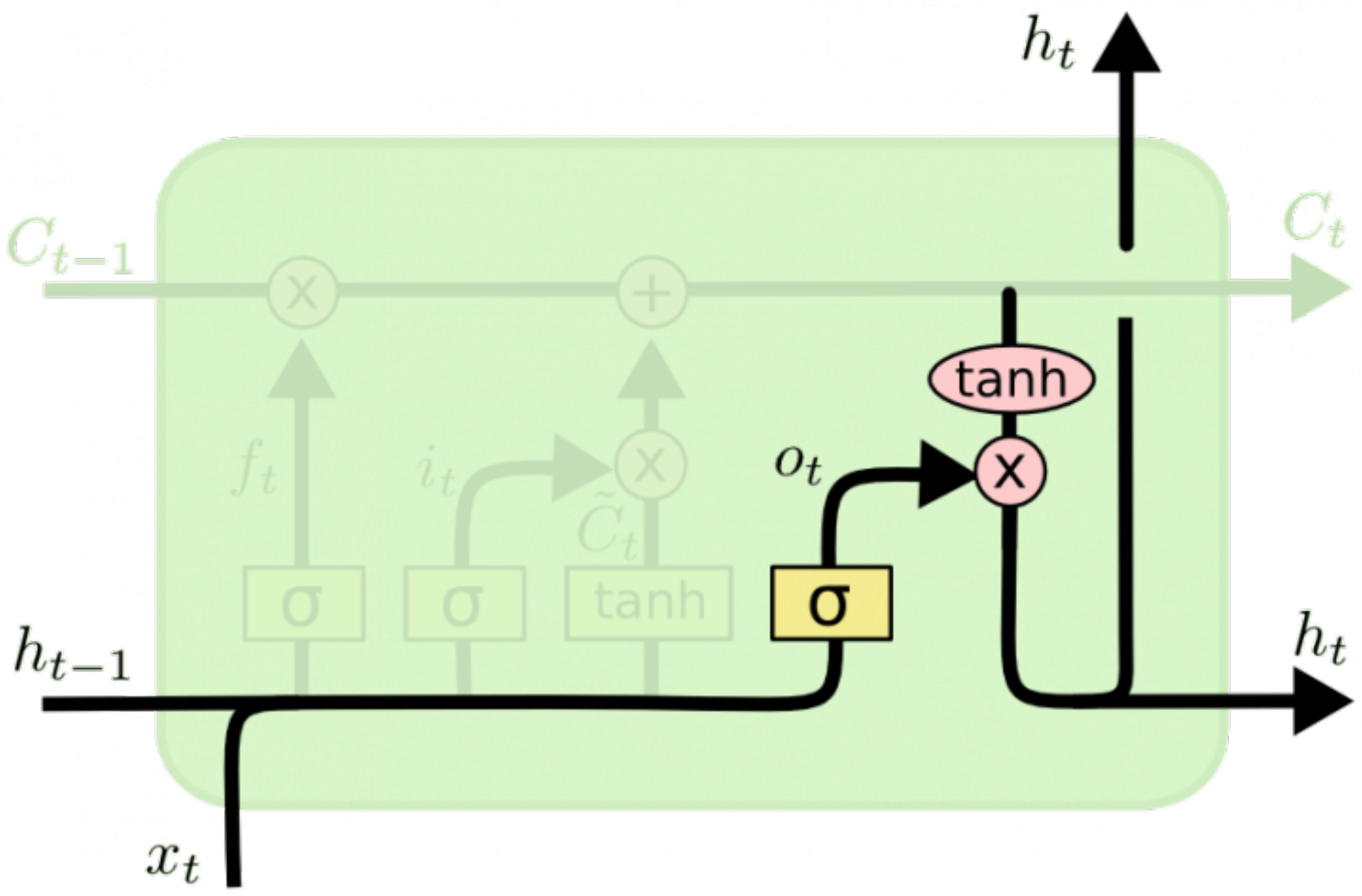


$$o_t = \sigma(W_t x_{t-1} + U_t h_{t-1} + b_t)$$

$$h_t = o_t \odot \tanh(c_t)$$

LSTM: output gate

Controls the output of the current step.



$$o_t = \sigma(W_t x_{t-1} + U_t h_{t-1} + b_t)$$

$$h_t = o_t \odot \tanh(c_t)$$

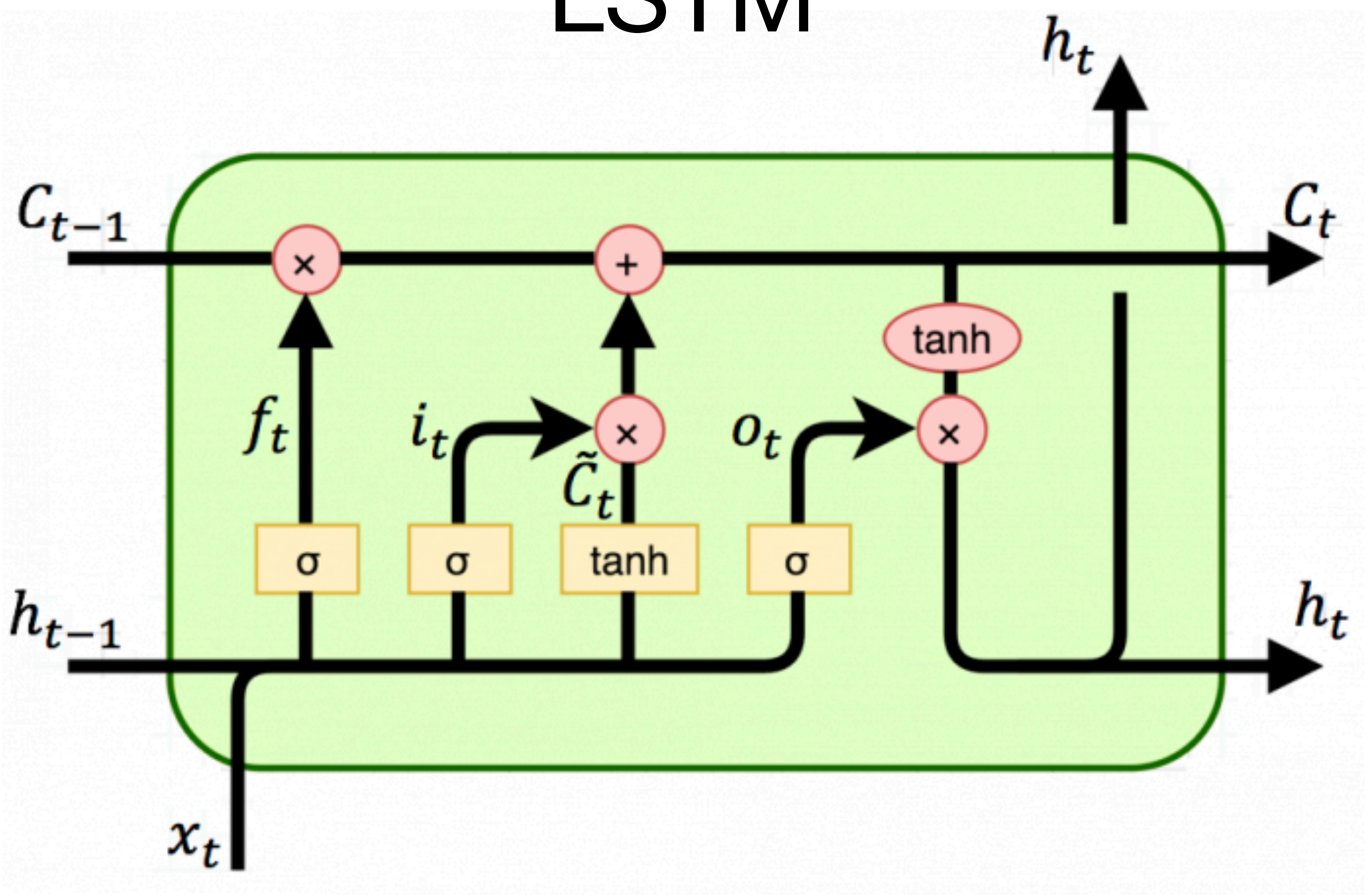
The teacher is teaching a lesson on recurrent models. _

Beginning of a new sentence. We must remember that we are talking about a teacher and a lesson.

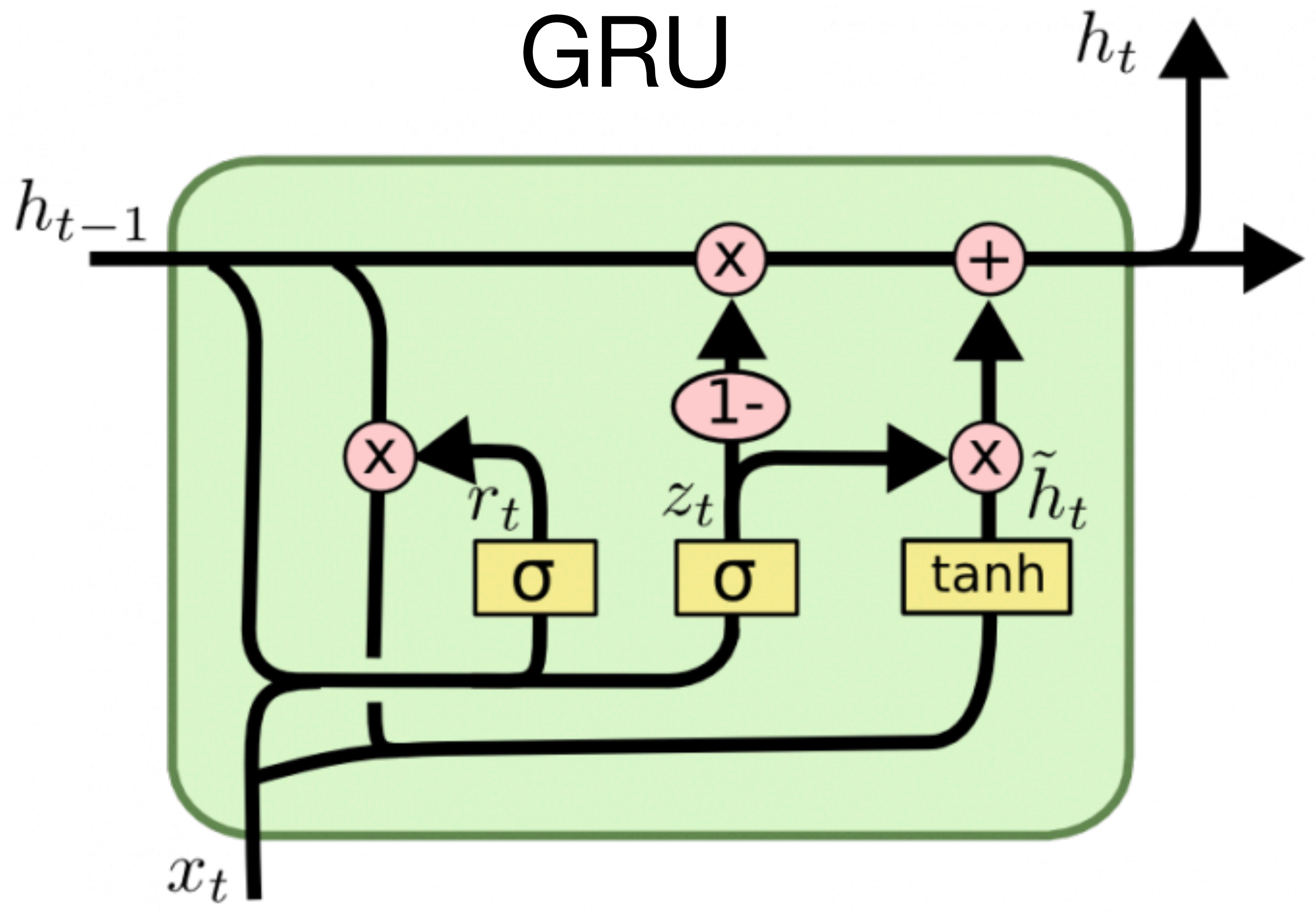
Gated recurrent units (GRU)

The most successful variation of LSTM for reducing the number of parameters.

LSTM



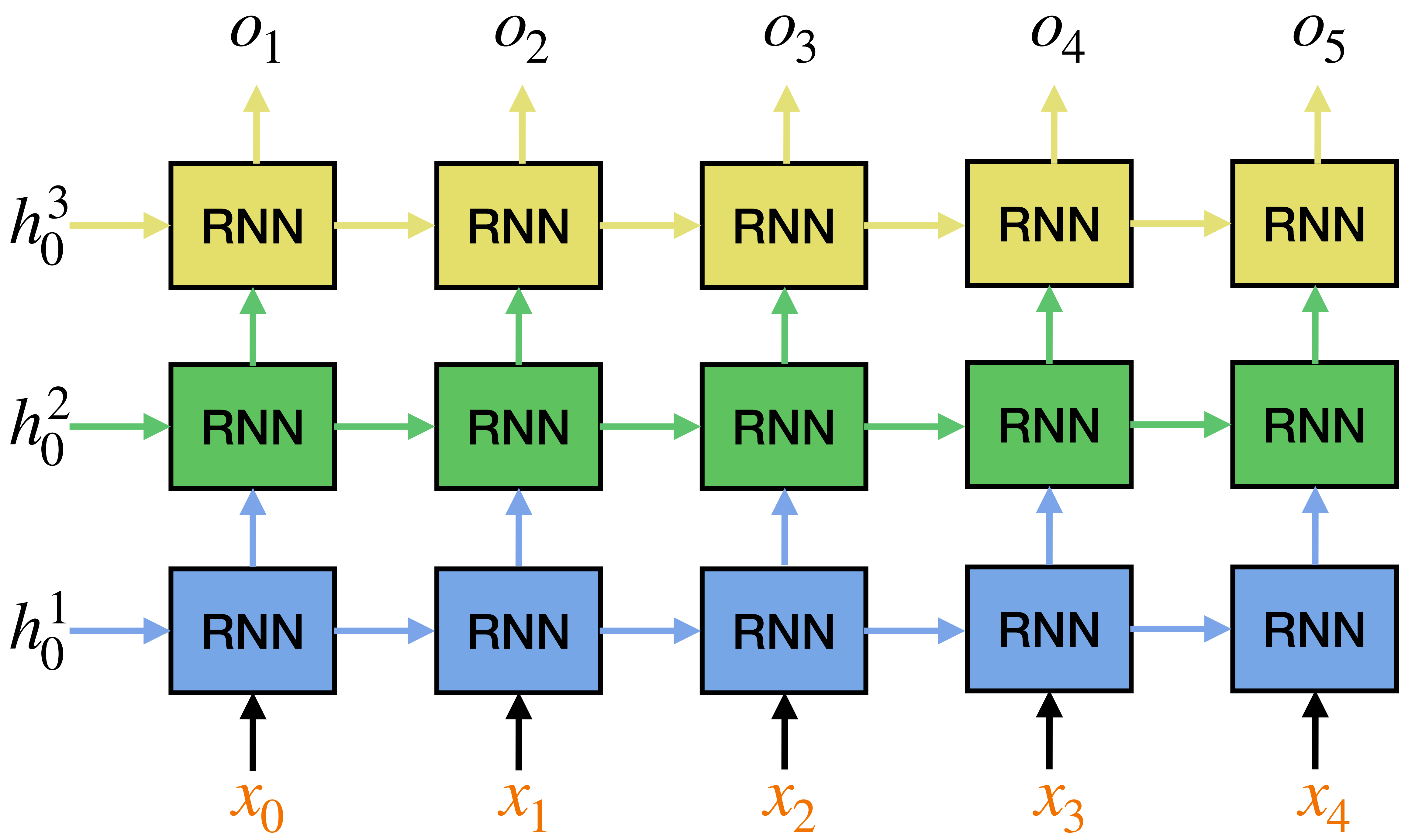
GRU



- Both models work well with remote dependencies.
- GRU has 3 layers, LSTM has 4.
- GRU trains better when corpus is limited

Multilayer recurrent networks

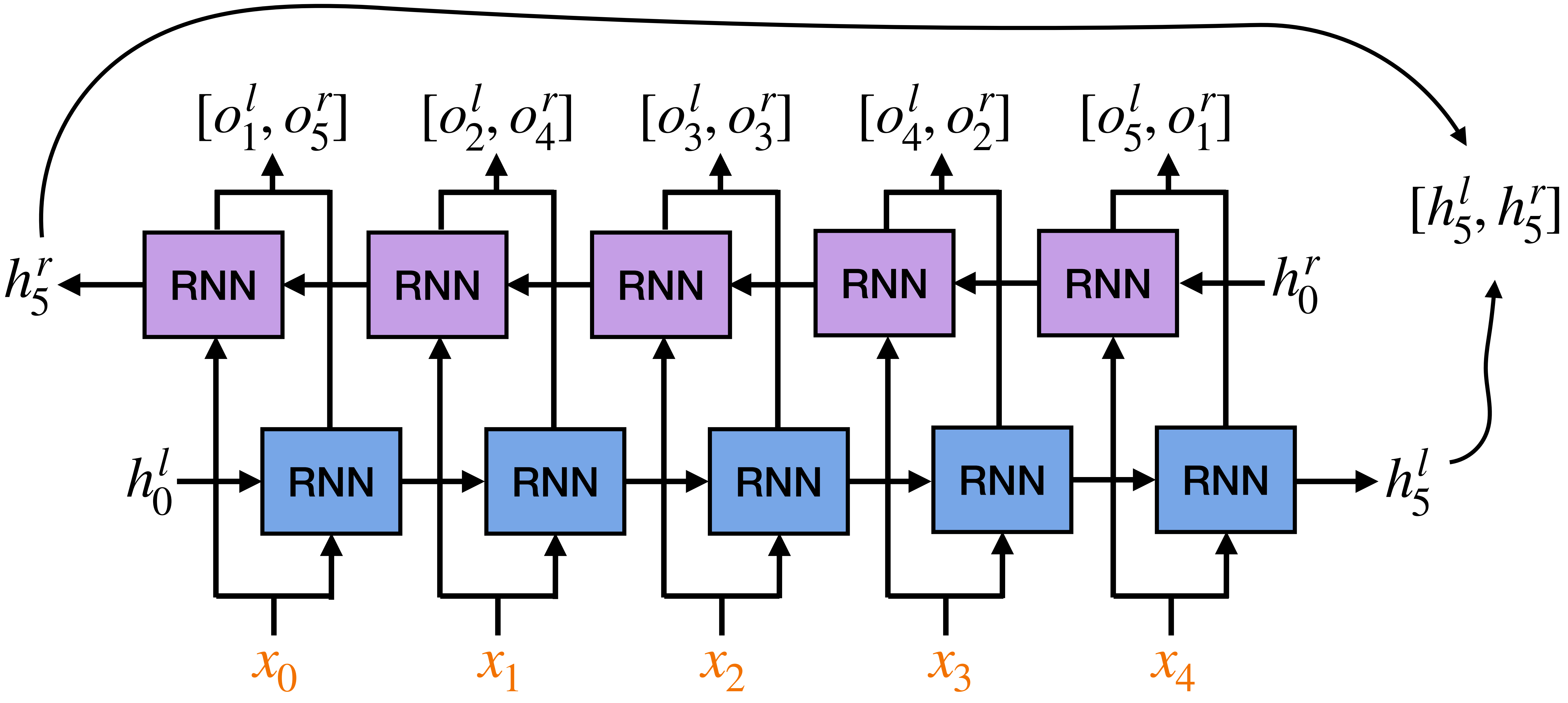
- The **outputs** of the current layer are the **inputs** of the next one
- More complex features are extracted



- The number of parameters increases by the number of layers
- Usually limited to **two** layers

Bidirectional recurrent networks (biRNN)

- Reads text from left to right and right to left.



- Has twice as many parameters
- Useless for generation