Deep learning
lecture 9, spring 2018

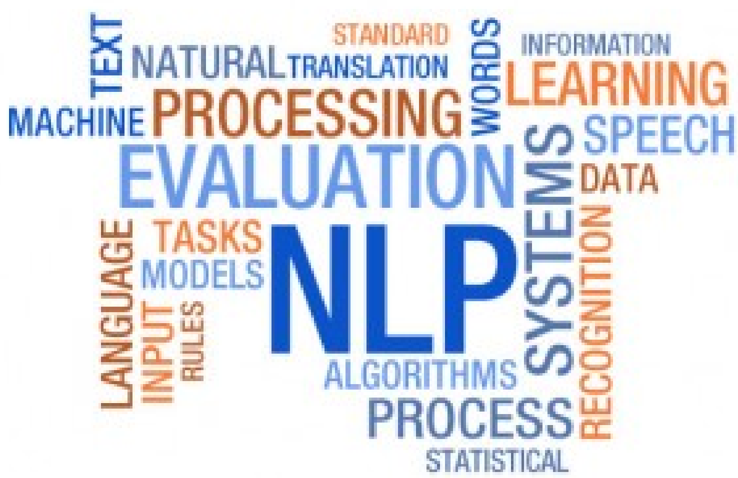
NLP basics,
Recurrent neural networks

British Hedgehog Preservation Society

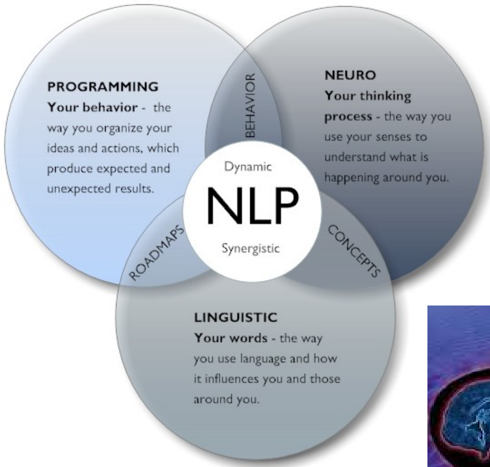
Skoltech
Skolkovo Institute of Science and Technology

LAMBDA

What is NLP?

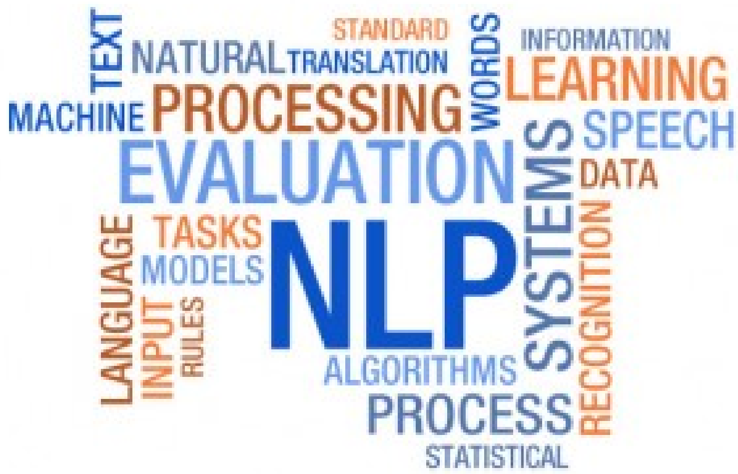


Natural Language Processing



Neuro-Linguistic Programming

What is NLP?



Natural Language Processing

Example: classification/regression

Why bother:

- Any ideas?

Example: classification/regression

Why bother:

- Adult content filter (safe search)
- Detect age/gender/interests by search querries
- Convert movie review text into “stars”
- Survey public opinion about the new iphone vs old one

...

Text 101

text

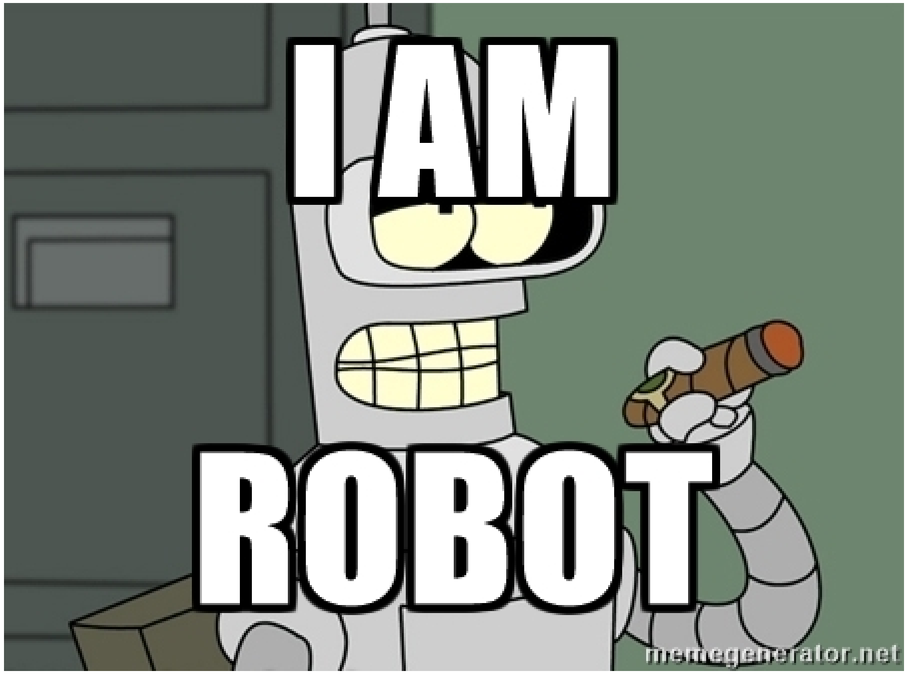
/tɛkst/

noun

1. a book or other written or printed work, regarded in terms of its content rather than its physical form.
   "a text which explores pain and grief"
   synonyms: written work, book, work, printed work, narrative
   "a text which explores pain and grief"

2. the main body of a book or other piece of writing, as distinct from other material such as notes, appendices, and illustrations.
   "the pictures are clear and relate well to the text"
   synonyms: words, wording; More

Text 101: nlp perspective



Text:
A sequence of tokens(words).

Token/word:
A sequence of characters.

Character:
An atomic element of text.
¯_(ツ)_/¯

NLP problems

One way of classifying

Sequence to answer:

Given input sequence, produce one answer

movie review → positive?; ***more ideas?***

Sequential problems

One way of classifying

Sequence to answer:

Given input sequence, produce one answer

movie review → positive?; mail → is spam?

job offer → salary($); blog entry → #likes;

Sequential problems

One way of classifying

Sequence to answer:

Given input sequence, produce one answer

movie review → positive?; mail → is spam?

job offer → salary($); blog entry → #likes;

Sequence labeling:

Given input sequence, produce one answer for each input

Part-Of-Speech tagging; ***???***

Sequential problems

One way of classifying

Sequence to answer:

Given input sequence, produce one answer

movie review → positive?; mail → is spam?

job offer → salary($); blog entry → #likes;

Sequence labeling:

Given input sequence, produce one answer for each input

Part-Of-Speech tagging; Named Entity Recognition;

Speech recognition (with a twist), Video segmentation;

Sequence generation:

Given some condition (optional), generate output sequence

Ideas?

Sequential problems

One way of classifying

Sequence to answer:

Given input sequence, produce one answer

movie review → positive?; mail → is spam?

job offer → salary($); blog entry → #likes;

Sequence labeling:

Given input sequence, produce one answer for each input

Part-Of-Speech tagging; Named Entity Recognition;

Speech recognition (with a twist), Video segmentation;

Sequence generation:

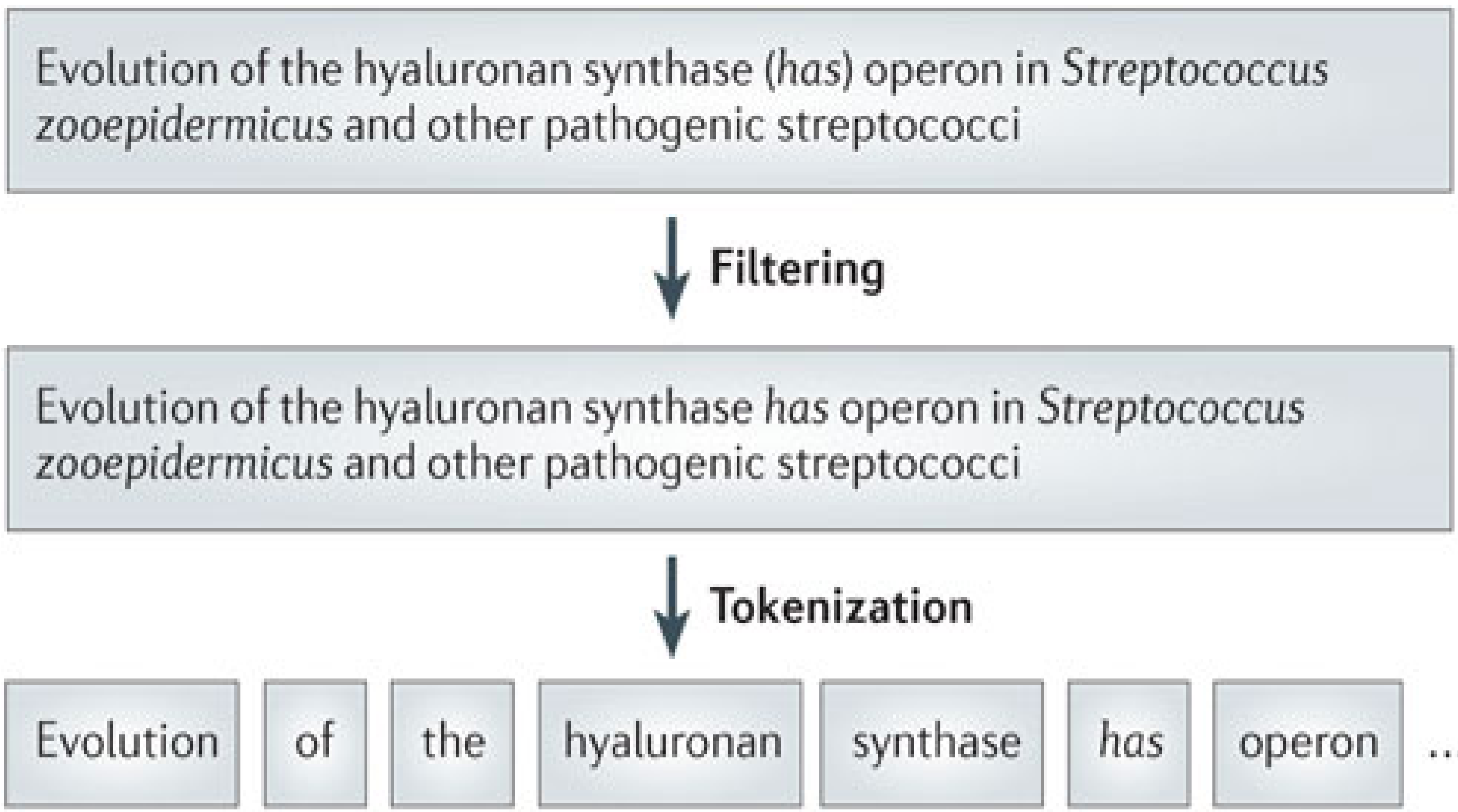
Given some condition (optional), generate output sequence

Image → caption; machine translation; conversation systems;

generating clickbait ads, arxiv articles, molecules (SMILES), etc.

Other: Document retrieval(ranking), recsys, topic modelling, ...

Text 101: tokens

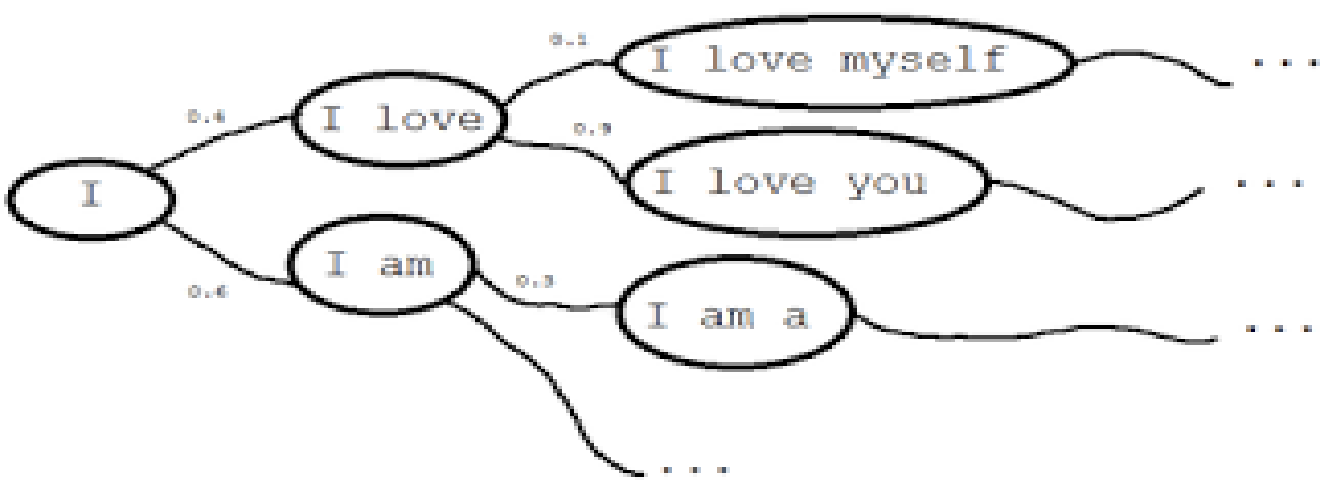


Language model

Objective:

- Learn P(text)

$$P(text) = P(w_{0,} w_{1,} \ldots, w_n) = P(w_0) \cdot P(w_1 | w_0) \cdot P(w_2 | w_1 w_0) \cdot \ldots \cdot P(w_n | \ldots)$$



Language model

Why learning it?

- Detect languages as P(text|language)
- Sentiment analysis P(text|happy)
- Any text analysis you can imagine
- Generate texts!
  - Cool article http://bit.ly/1K610Ie
  - Generating clickbait: http://bit.ly/21cZM70

Language model

- Actual distribution

$$P(text) = P(w_{0,} w_{1,} \ldots, w_n) = P(w_0) \cdot P(w_1 | w_0) \cdot P(w_2 | w_1 w_0) \cdot \ldots \cdot P(w_n | \ldots)$$

- Bag of words assumption (independent words)

$$P(text) = P(w_{0,} w_{1,} \ldots, w_n) = P(w_0) \cdot P(w_1) \cdot P(w_2) \cdot \ldots \cdot P(w_n)$$

- **Anything better?**

Language model

- Actual distribution

$$P(text) = P(w_{0,} w_{1,} \ldots, w_n) = P(w_0) \cdot P(w_1|w_0) \cdot P(w_2|w_1 w_0) \cdot \ldots \cdot P(w_n|\ldots)$$

- Bag of words assumption (independent words)

$$P(text) = P(w_{0,} w_{1,} \ldots, w_n) = P(w_0) \cdot P(w_1) \cdot P(w_2) \cdot \ldots \cdot P(w_n)$$

- Markov assumption

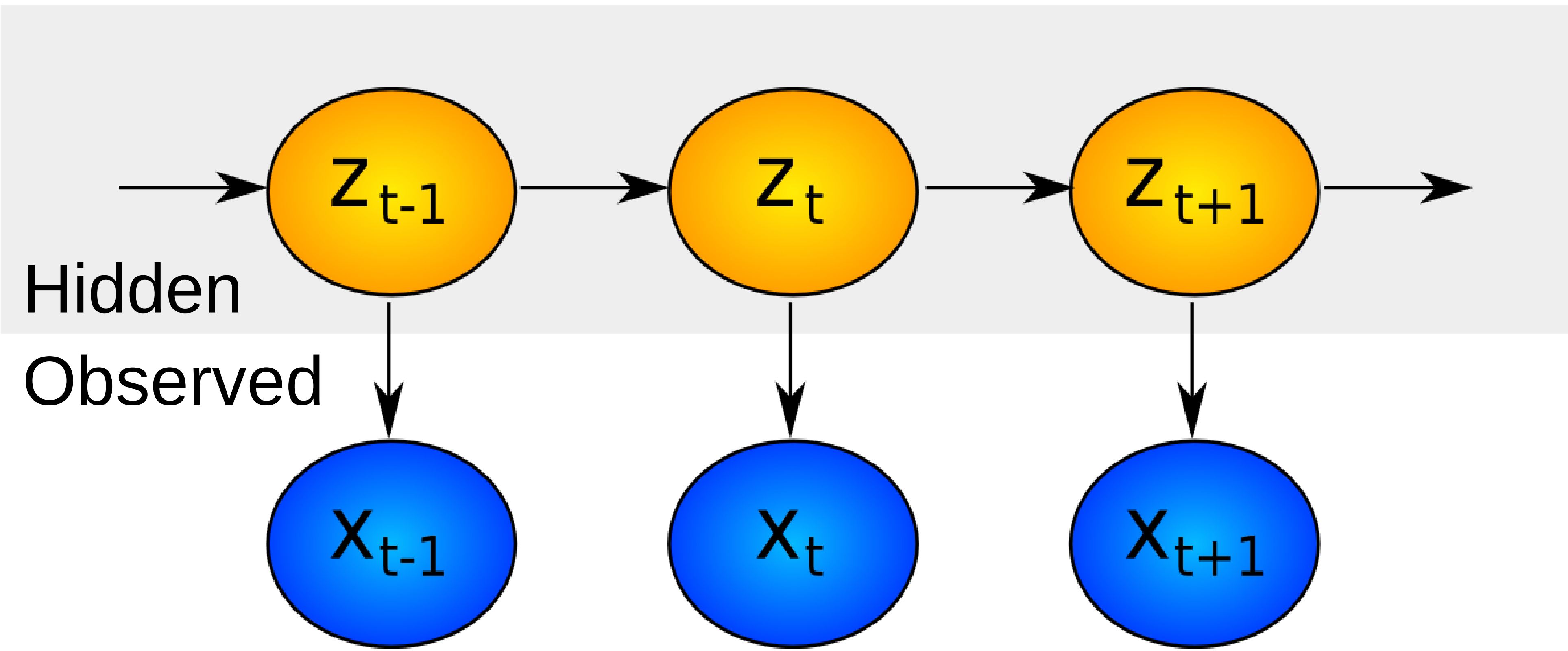
$$P(text) = P(w_{0,} w_{1,} \ldots, w_n) = P(w_0) \cdot P(w_1|w\,0) \cdot P(w_2|w_1) \cdot \ldots \cdot P(w_n|w_{n-1})$$

  - also 3-gram, 5-gram, 100-gram

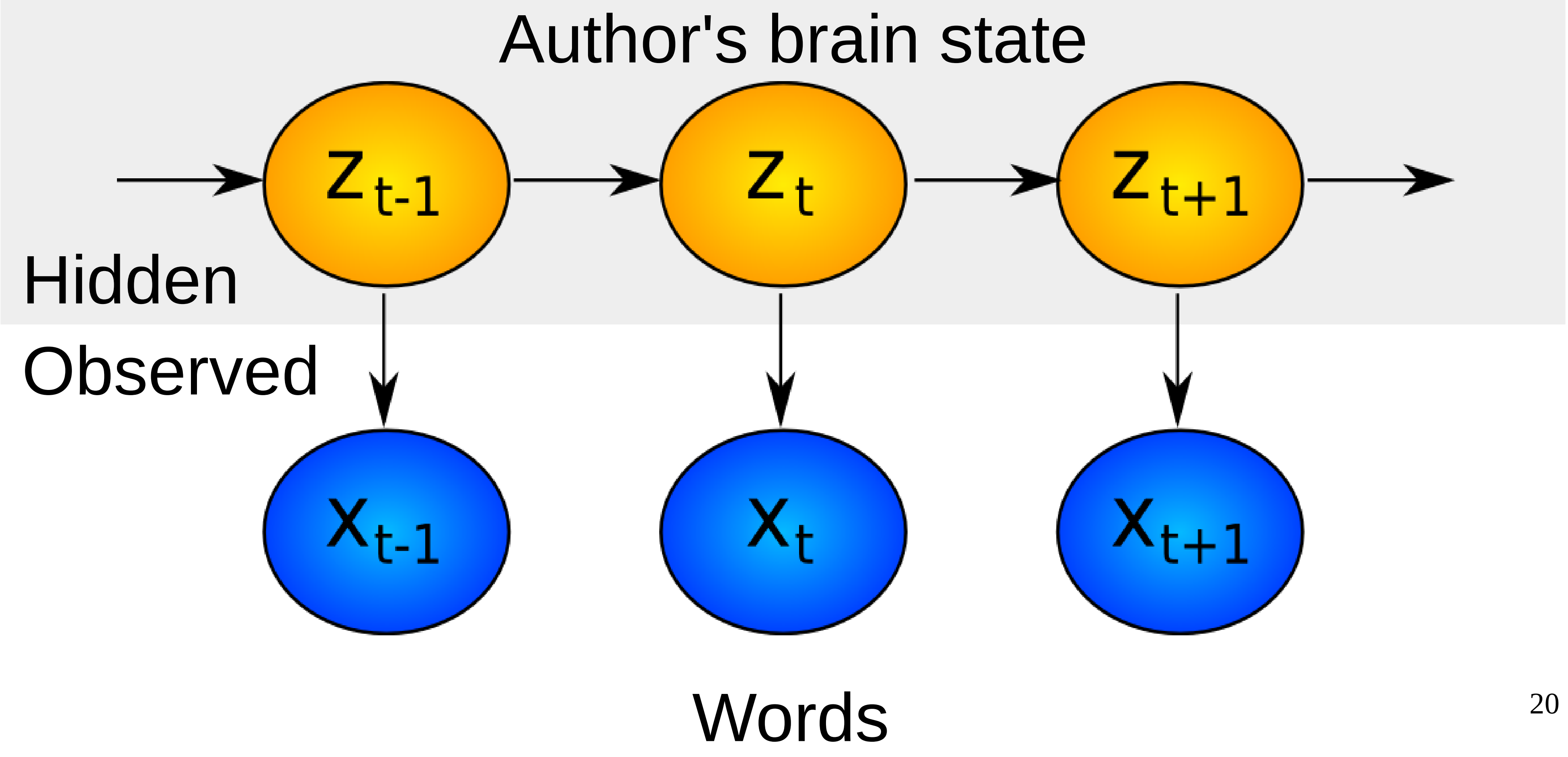
Can we learn* arbitrarily long dependencies?

* without infinitely many parameters

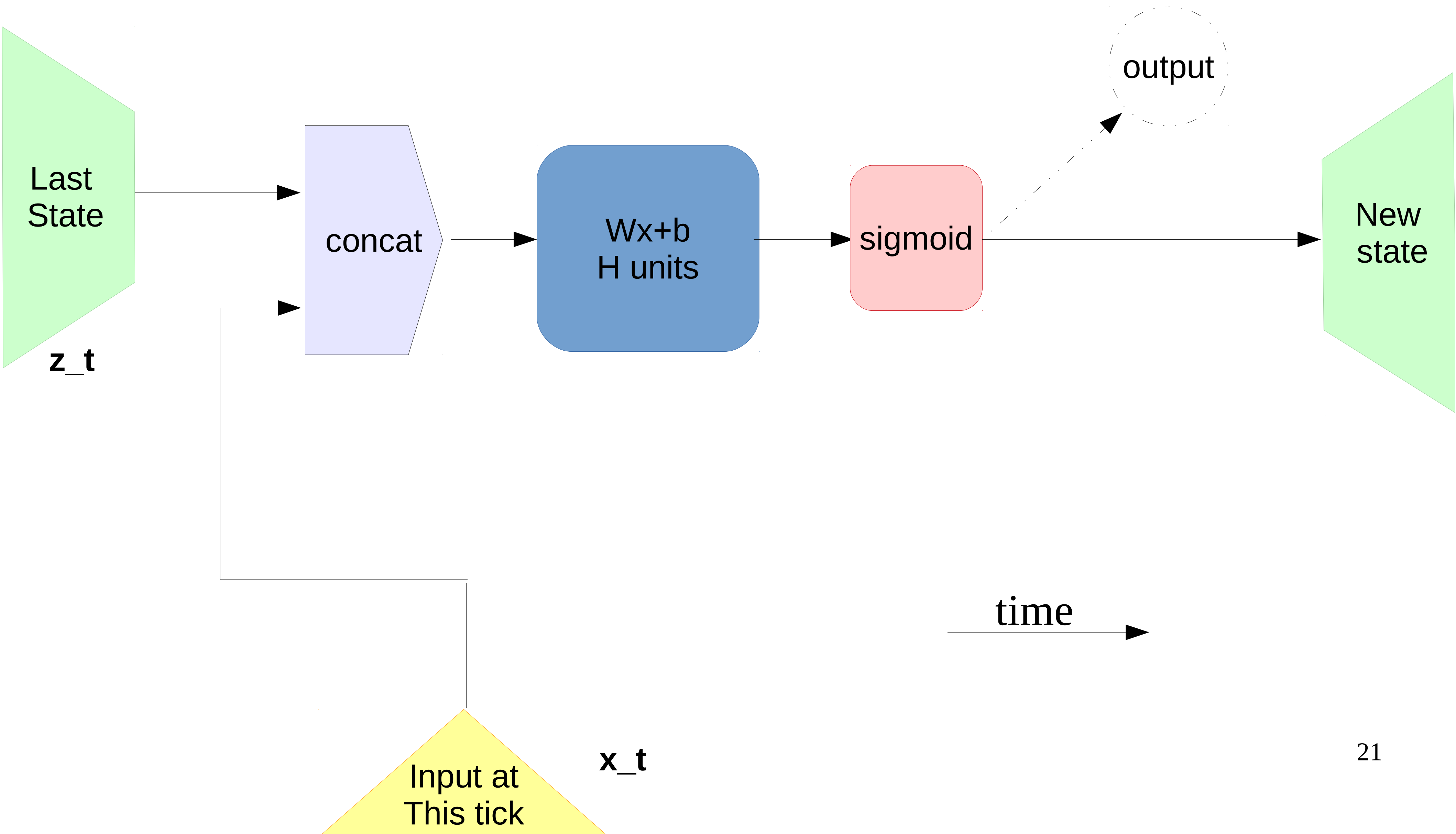
Hidden Markov Models: what's hidden



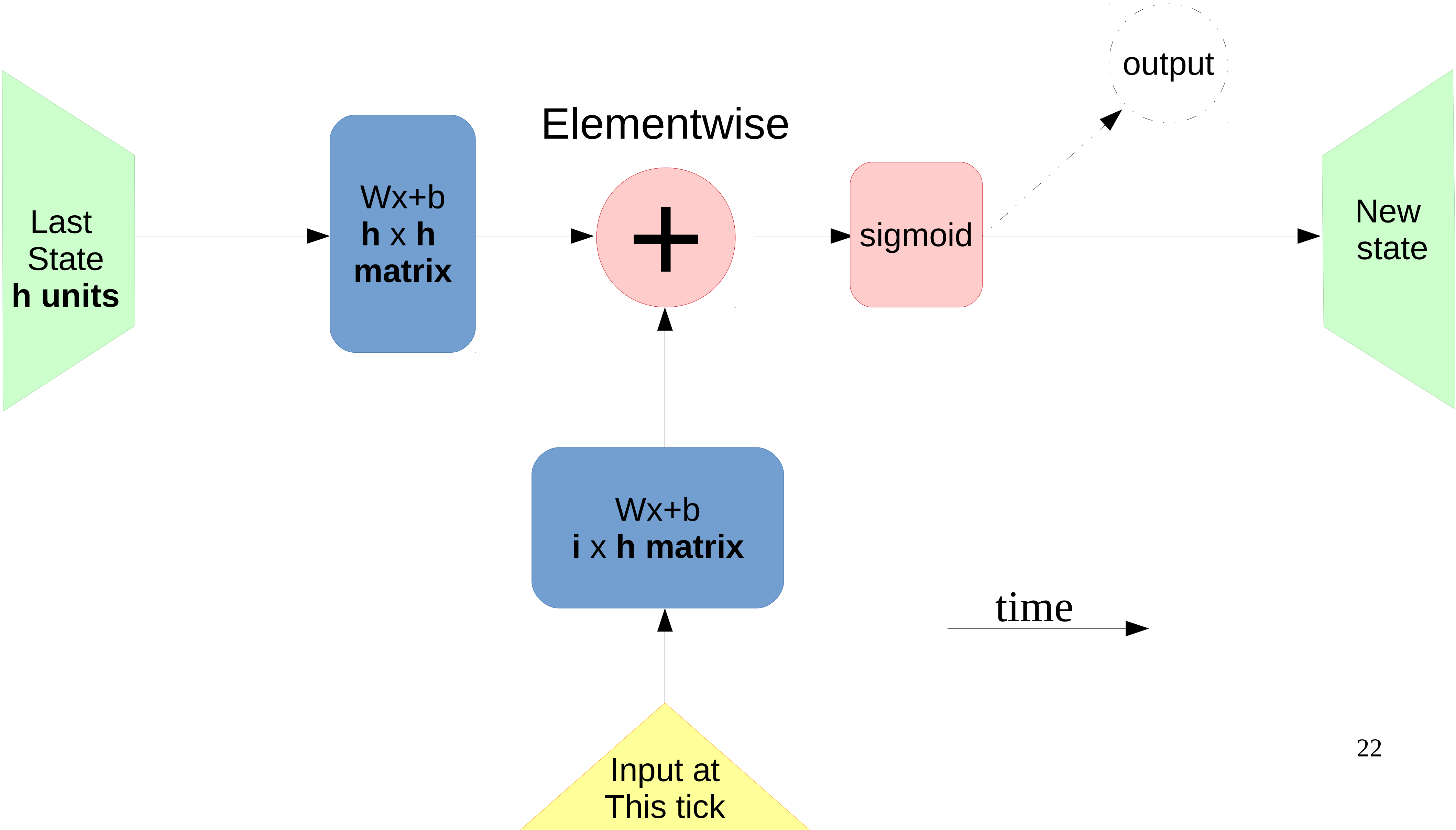
Hidden Markov Models: what is hidden



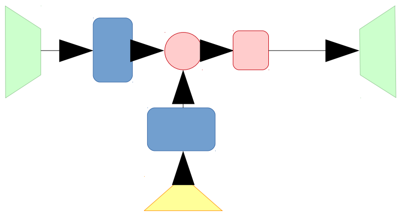
Recurrent neural network: one step



Recurrent neural network: one step

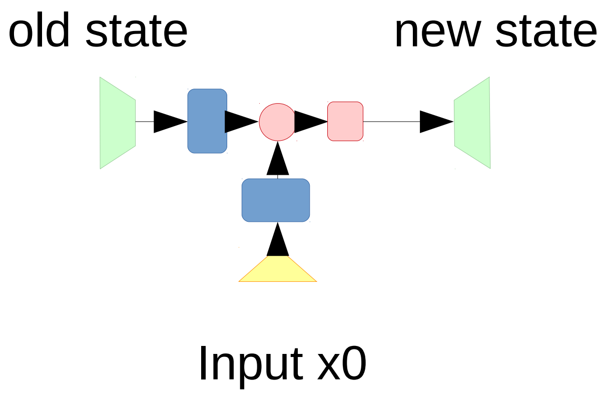


Recurrent neural network

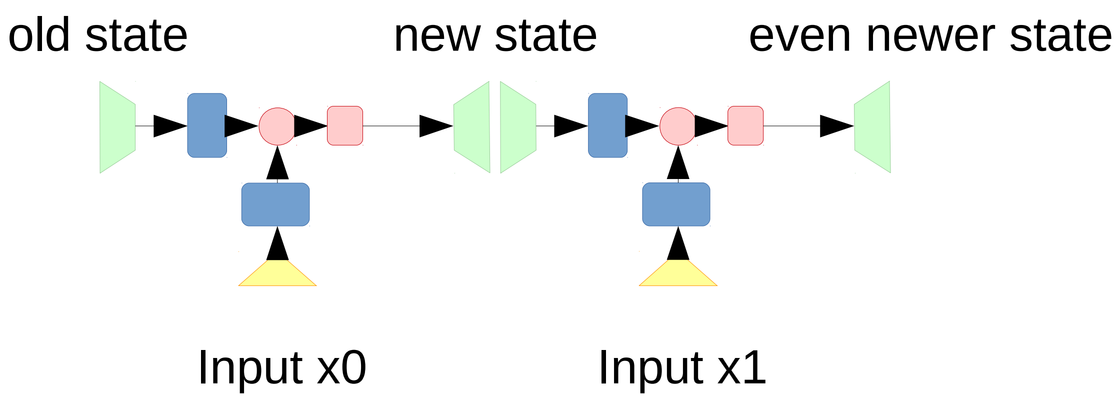


Zoom-out
of previous slide

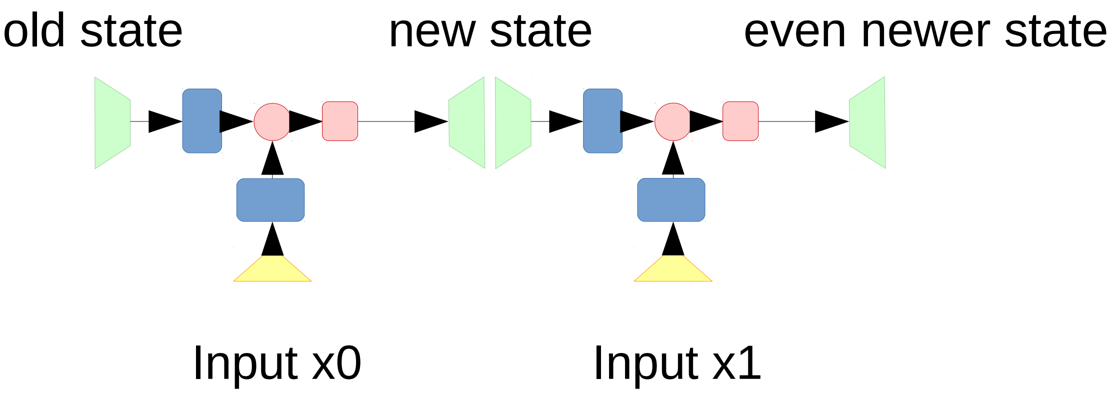
Recurrent neural network



Recurrent neural network

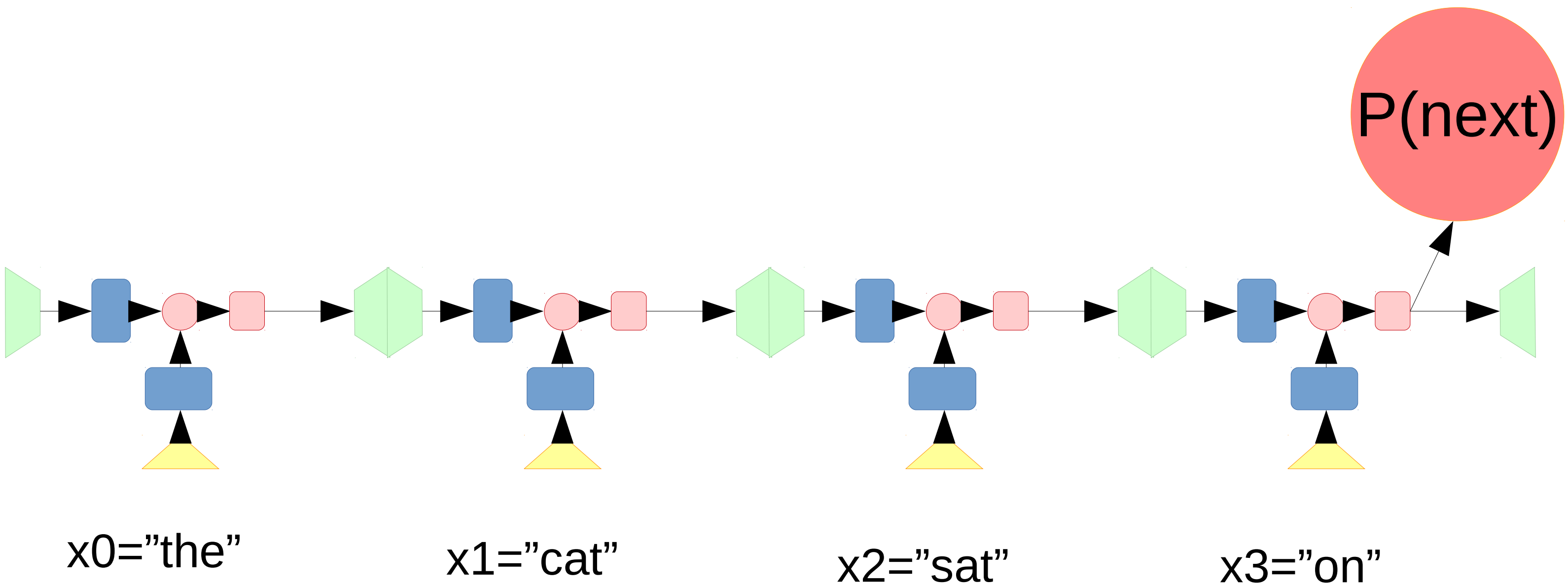


Recurrent neural network

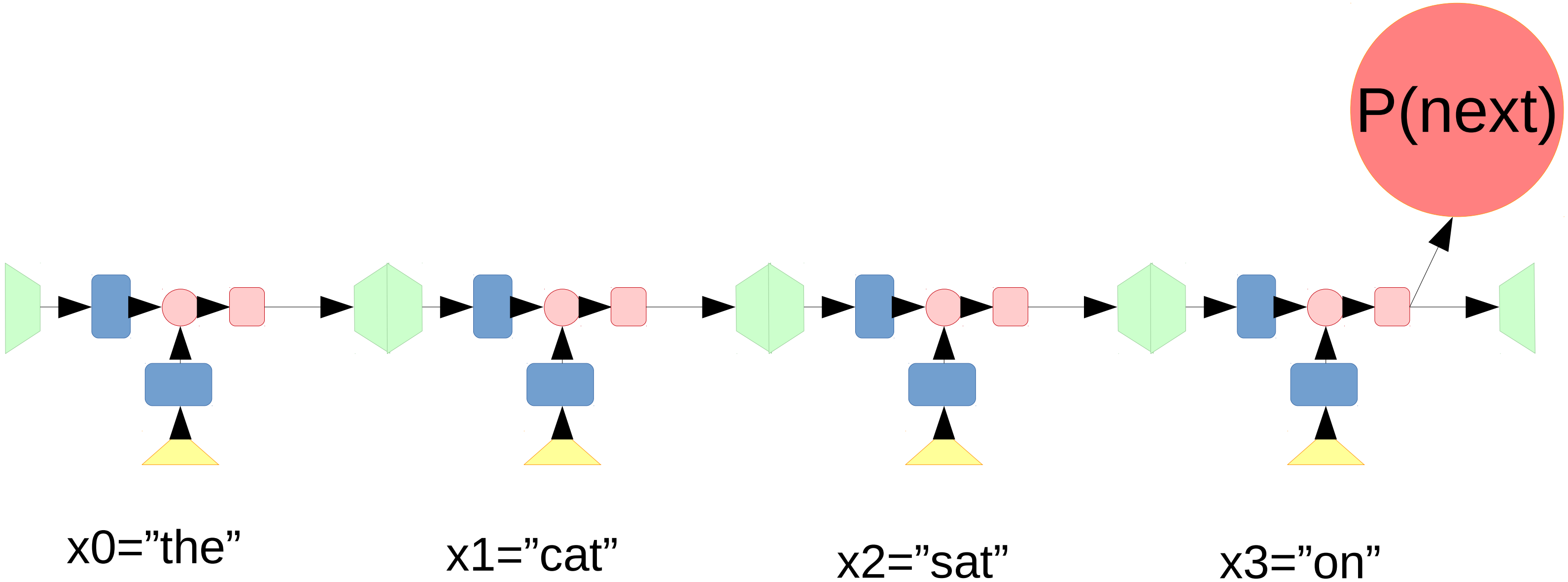


We use **same weight matrices** for all steps

Recurrent neural network

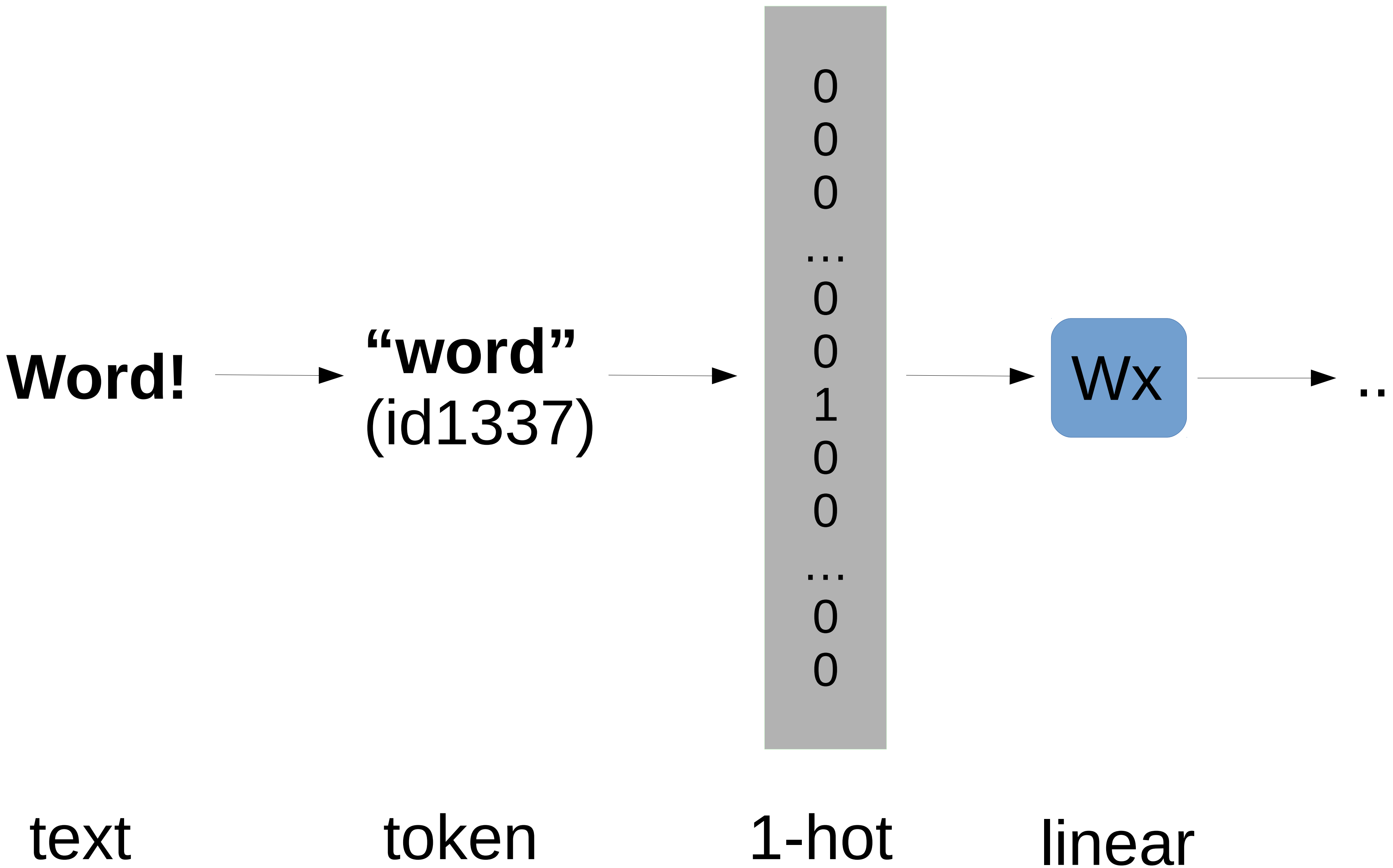


Recurrent neural network

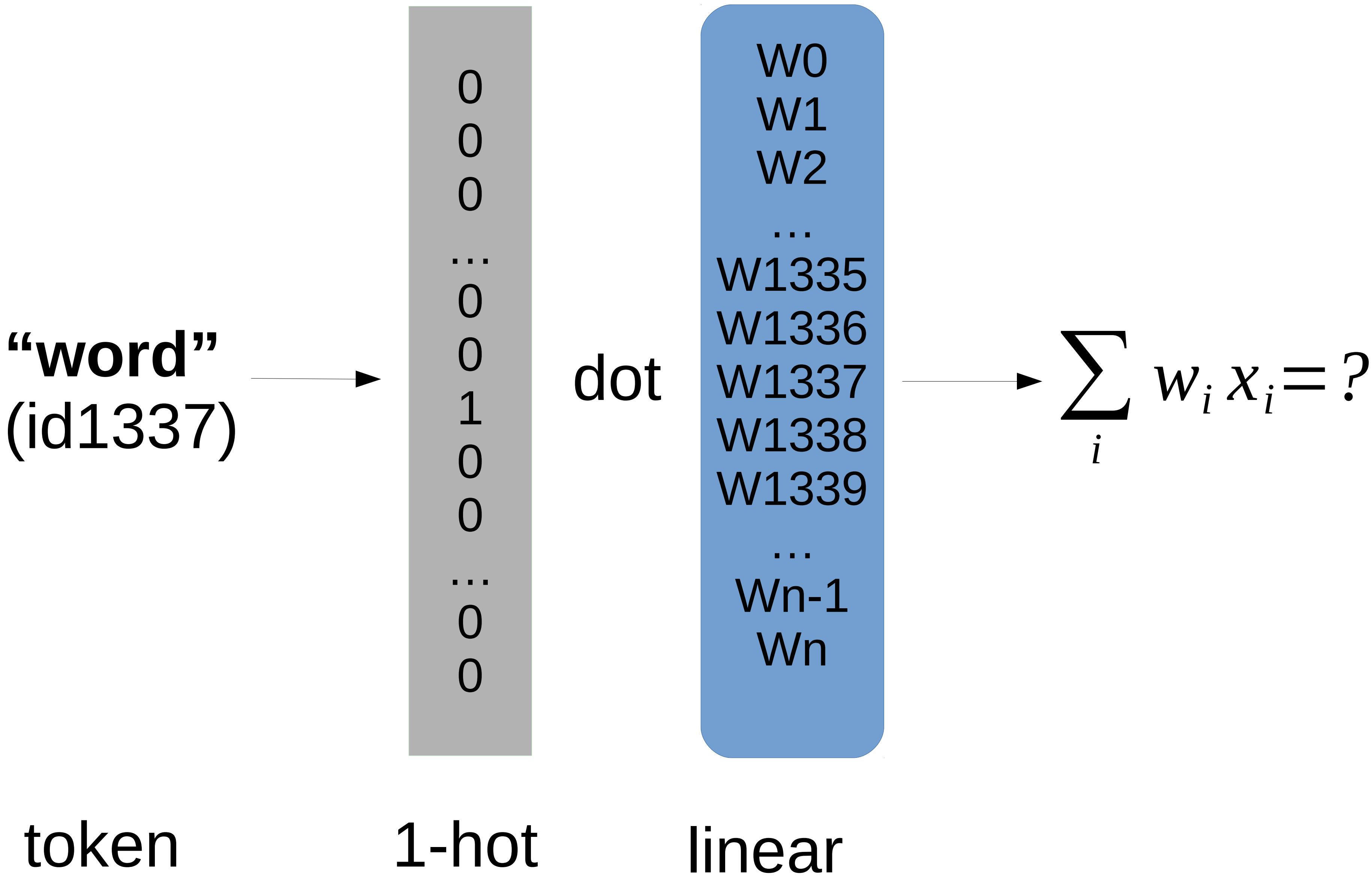


How can we represent words?

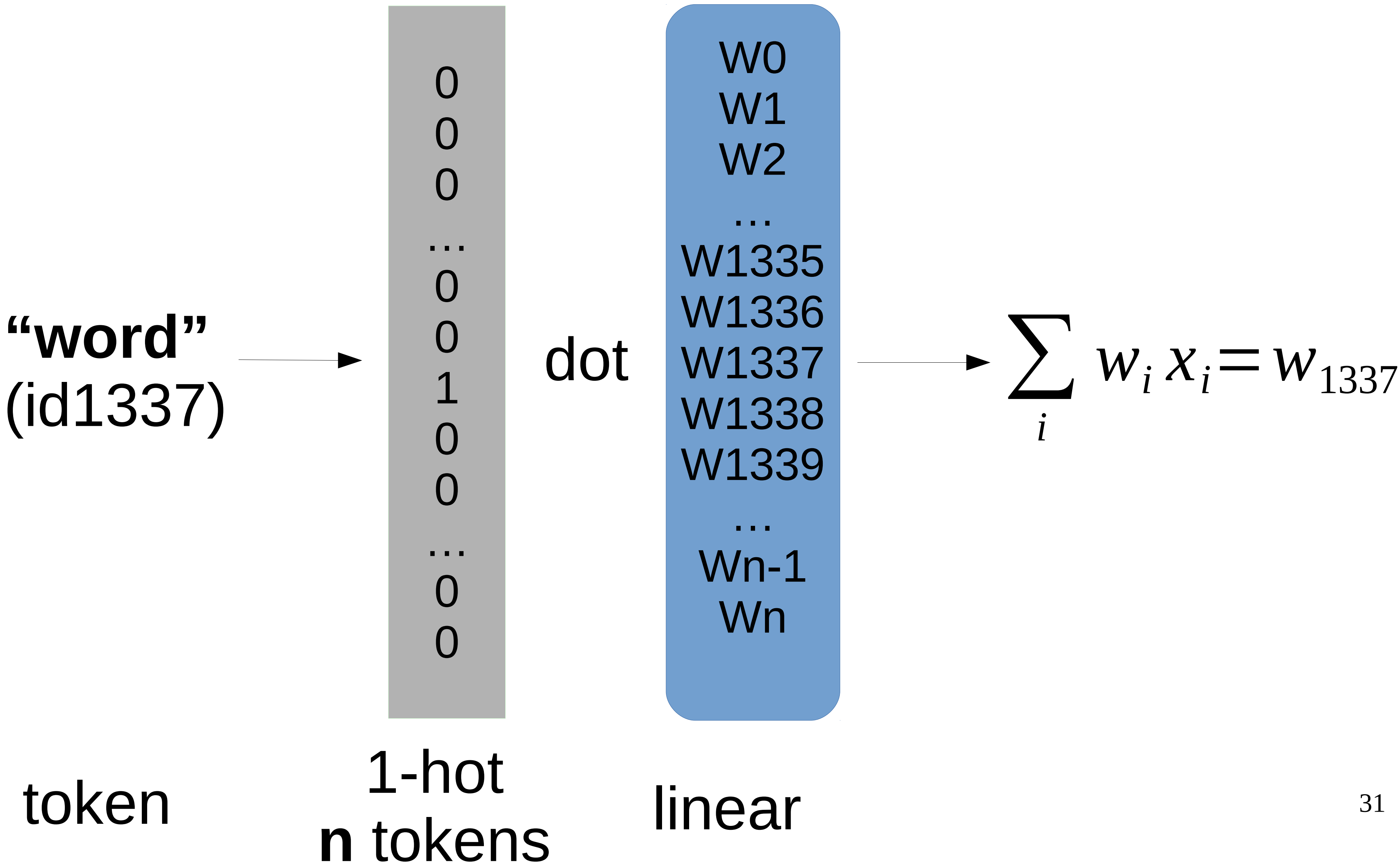
Sparse vector products



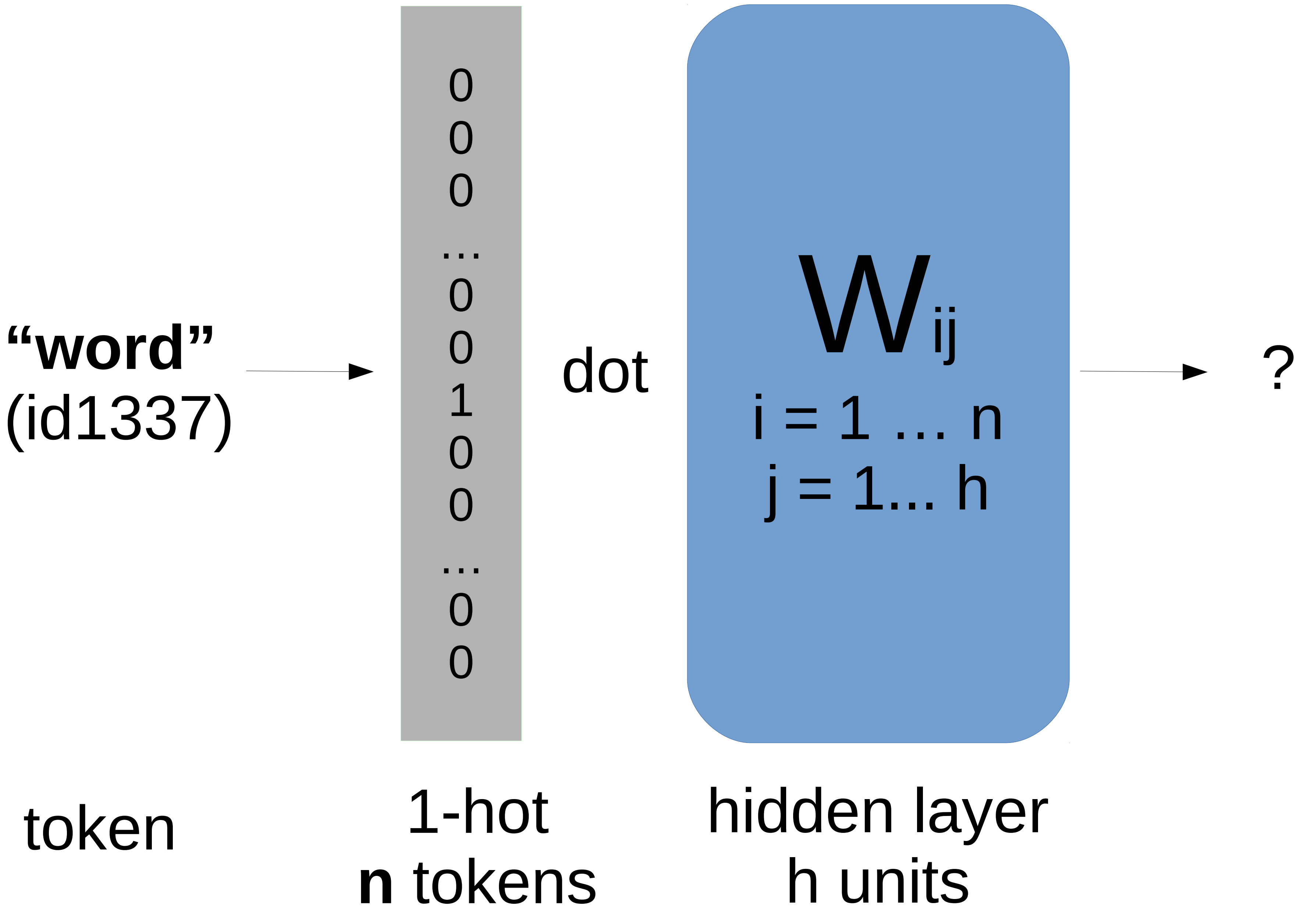
How to represent words?



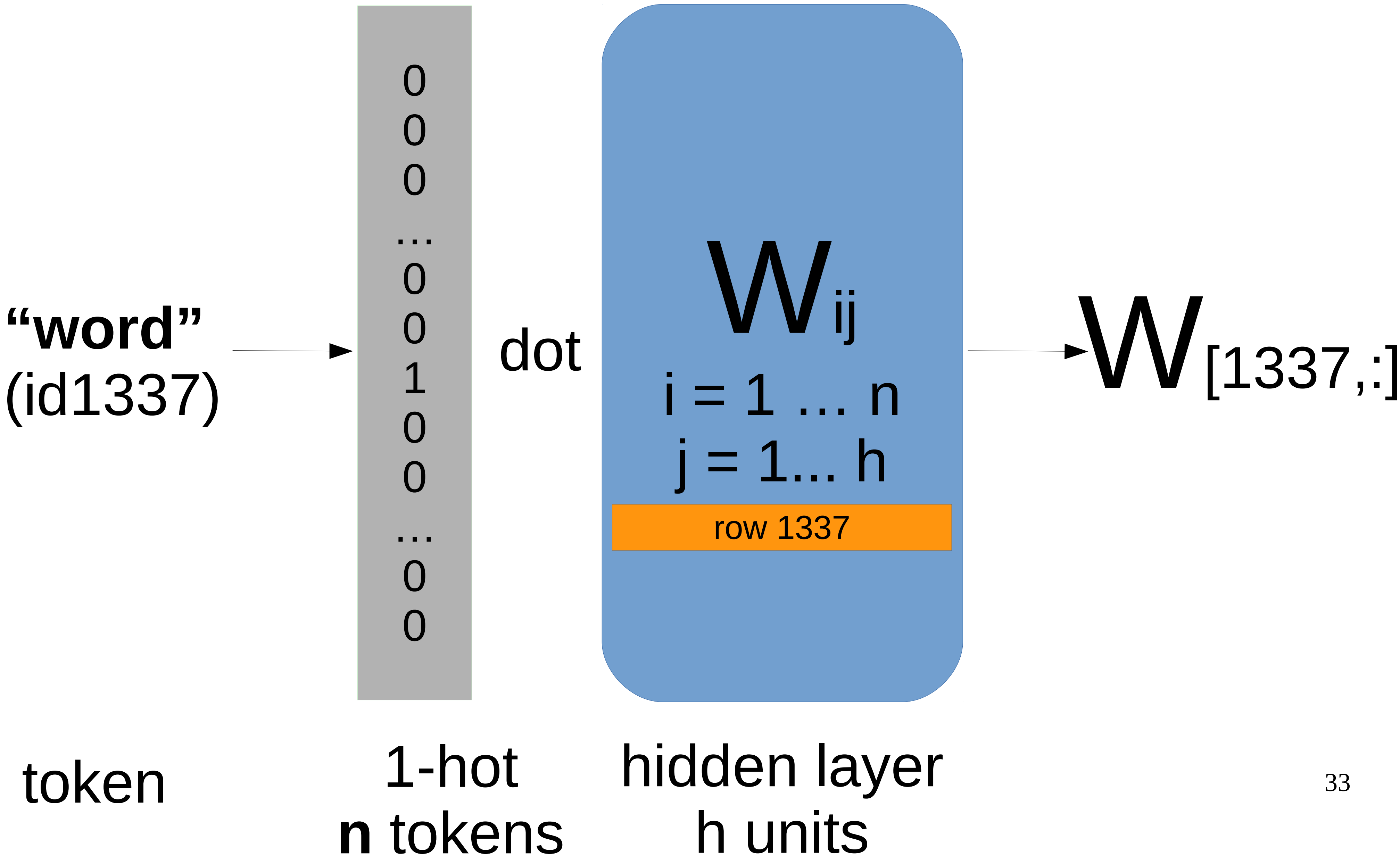
How to represent words?



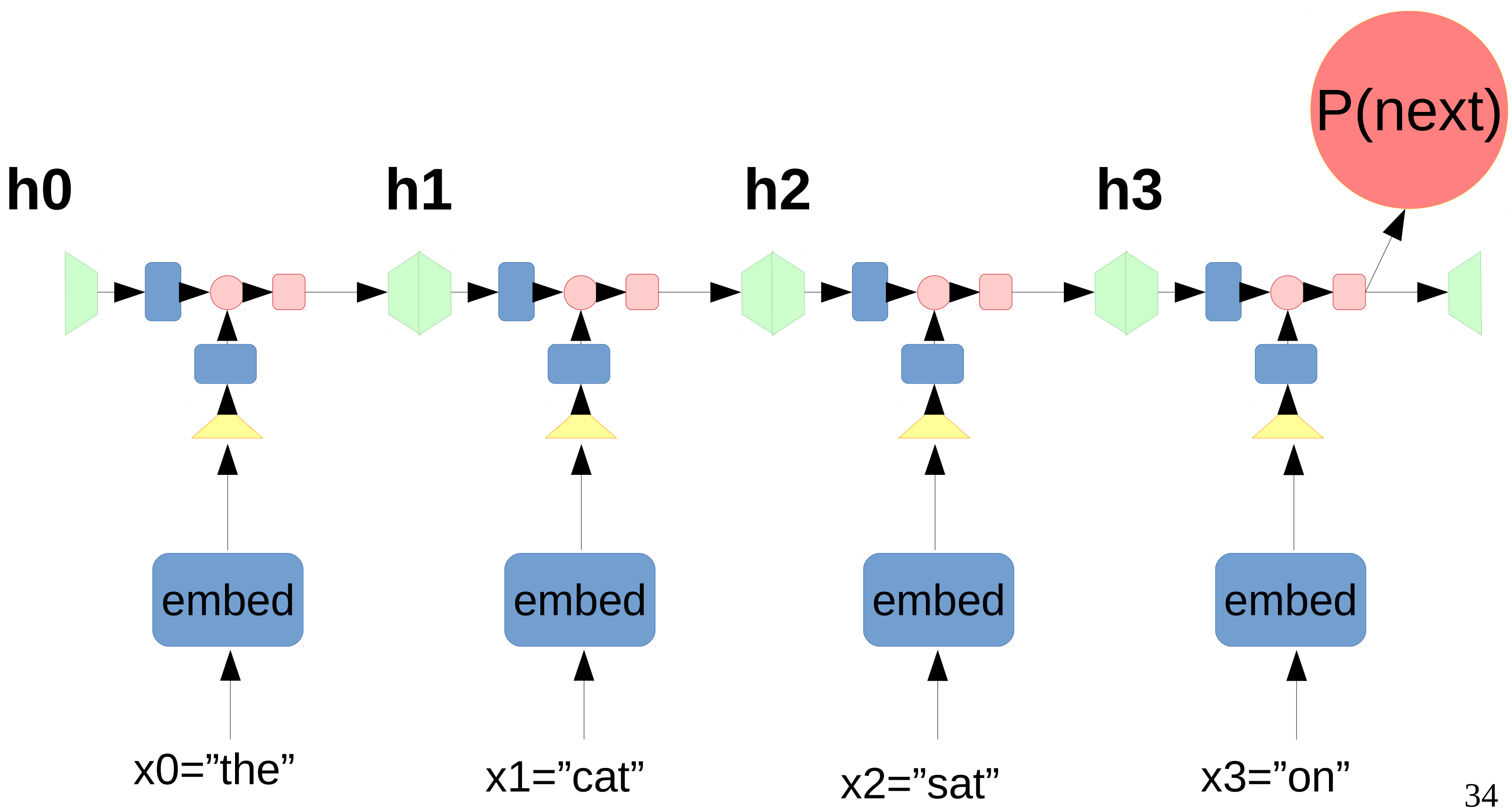
How to represent words?



“Embedding Layer”



Recurrent neural network

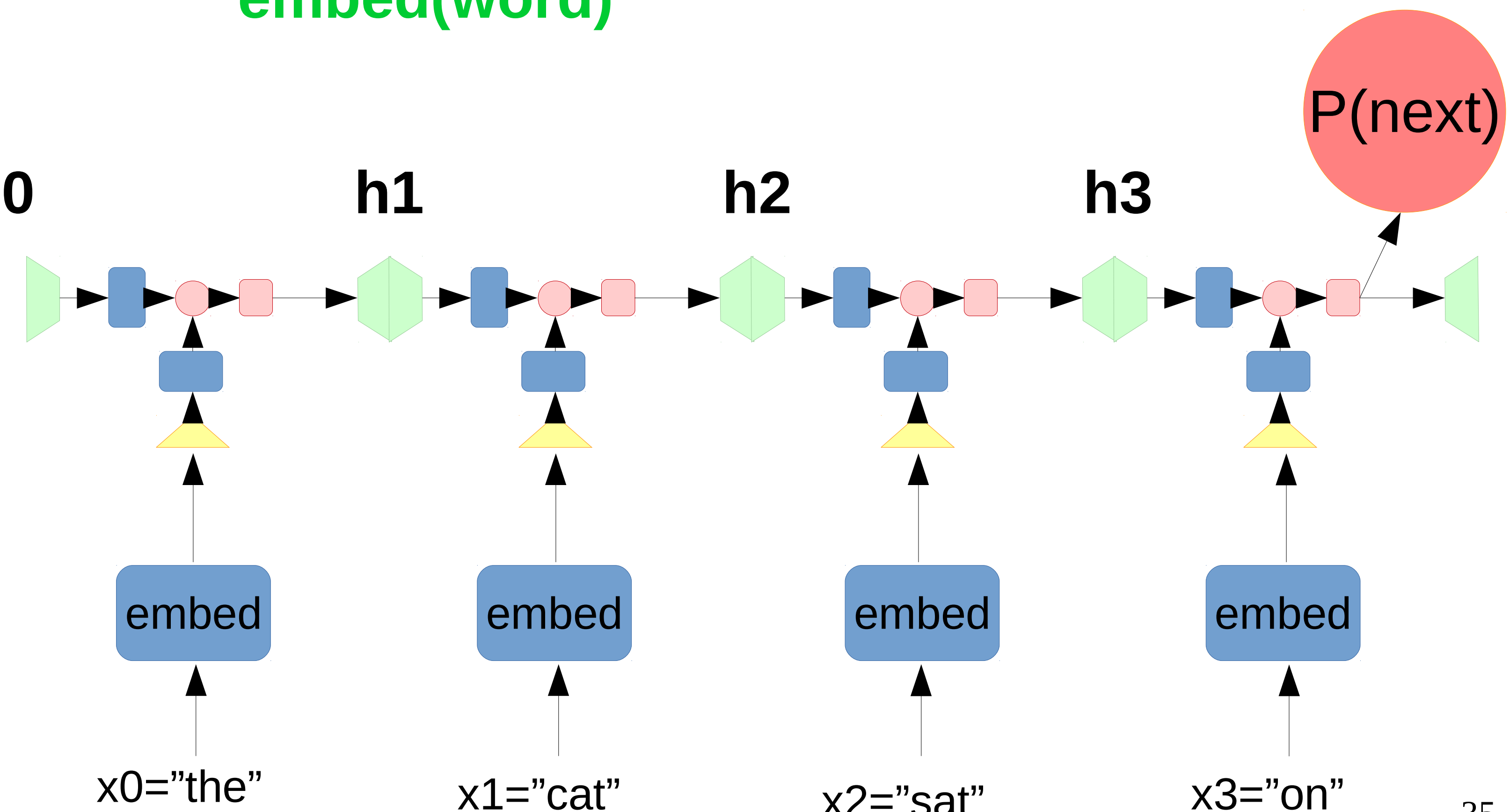


Recurrent neural network

$$h_0 = \overline{0}$$

$$h_1 = \sigma\left(W_{hid} \cdot h_0 + W_{inp} \cdot x_0 + b\right)$$

embed(word)

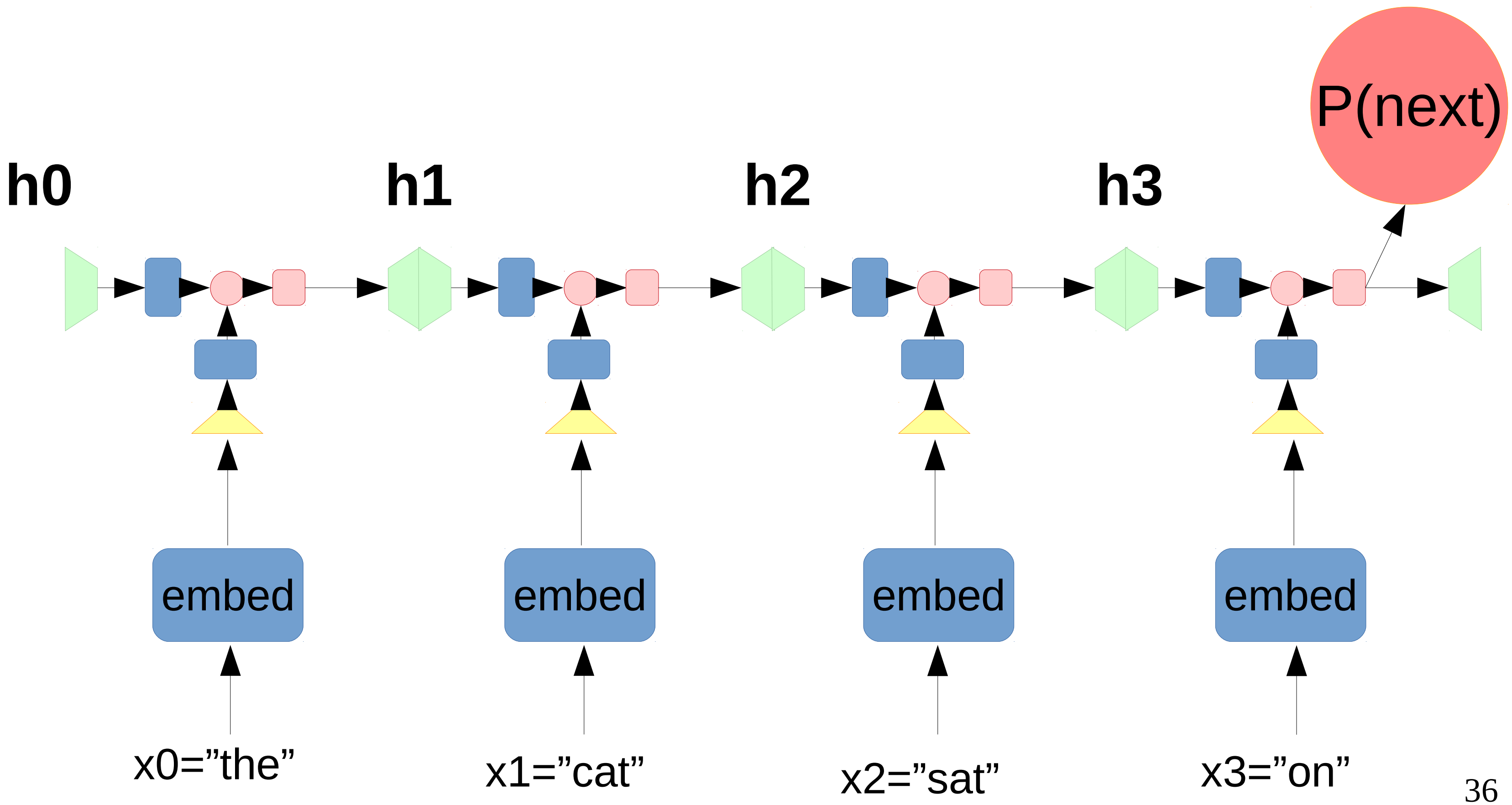


Recurrent neural network

$$h_0 = \bar{0}$$

$$h_1 = \sigma\left(W_{hid} \cdot h_0 + W_{inp} \cdot x_0 + b\right)$$

$$h_2 = ?$$



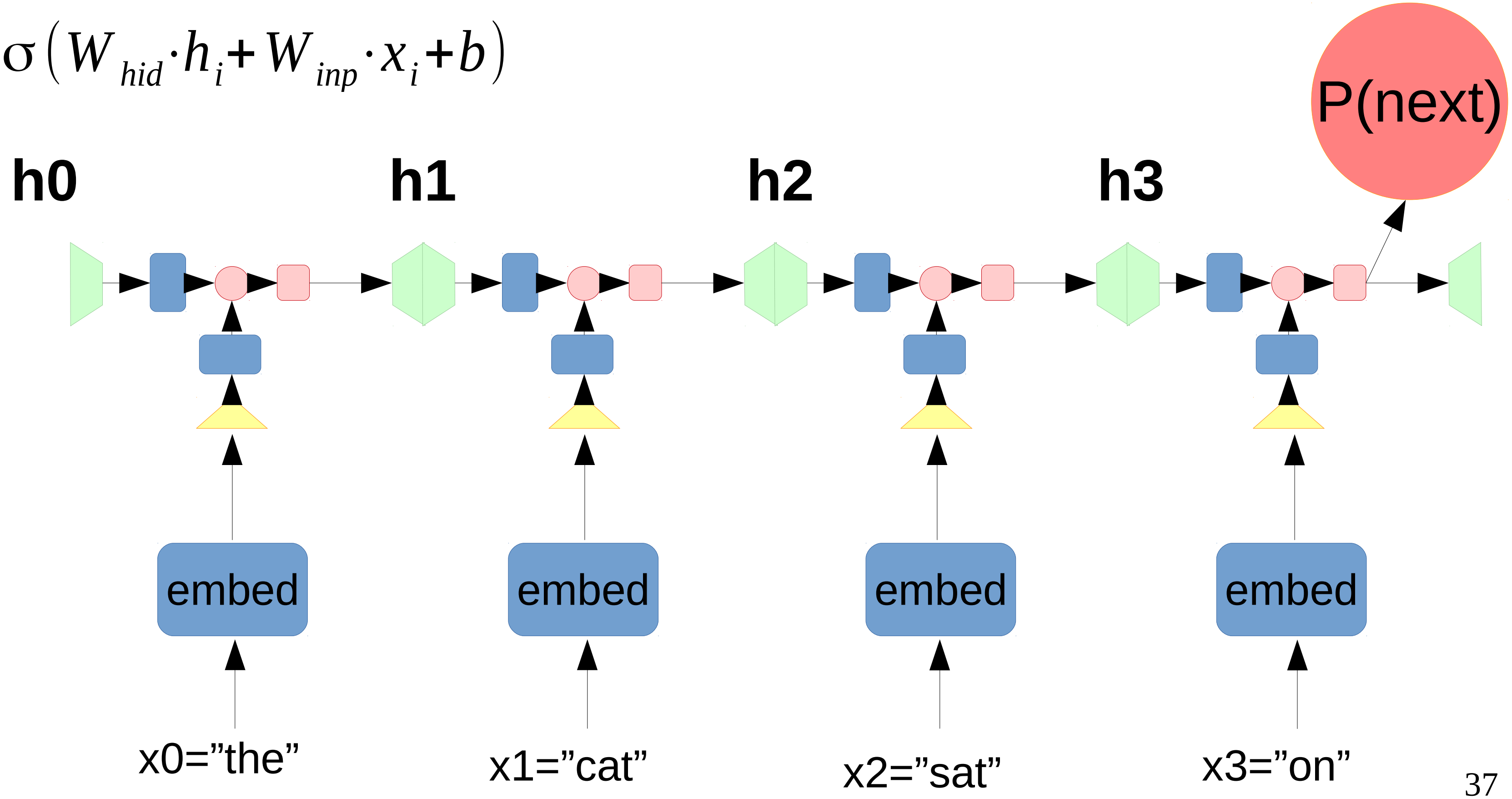
Recurrent neural network

$$h_0 = \bar{0}$$

$$h_1 = \sigma\left(W_{hid} \cdot h_0 + W_{inp} \cdot x_0 + b\right)$$

$$h_2 = \sigma\left(W_{hid} \cdot h_1 + W_{inp} \cdot x_1 + b\right) = \sigma\left(W_{hid} \cdot \sigma\left(W_{hid} \cdot h_0 + W_{inp} \cdot x_0 + b\right) + W_{inp} \cdot x_1 + b\right)$$

$$h_{i+1} = \sigma\left(W_{hid} \cdot h_i + W_{inp} \cdot x_i + b\right)$$



Recurrent neural network

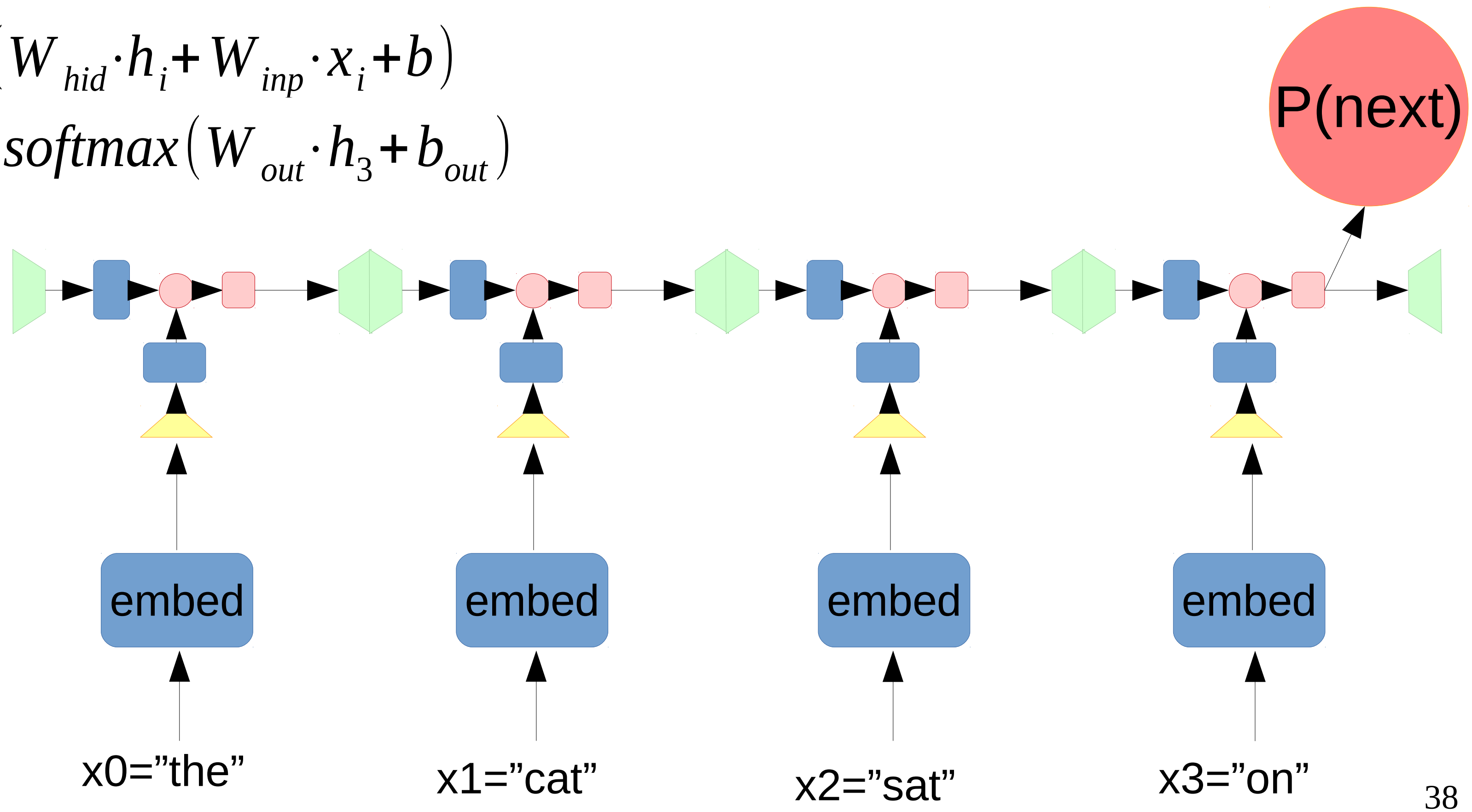
$$h_0 = \bar{0}$$

$$h_1 = \sigma(W_{hid} \cdot h_0 + W_{inp} \cdot x_0 + b)$$

$$h_2 = \sigma(W_{hid} \cdot h_1 + W_{inp} \cdot x_1 + b) = \sigma(W_{hid} \cdot \sigma(W_{hid} \cdot h_0 + W_{inp} \cdot x_0 + b) + W_{inp} \cdot x_1 + b)$$

$$h_{i+1} = \sigma(W_{hid} \cdot h_i + W_{inp} \cdot x_i + b)$$

$$P(x_4) = softmax(W_{out} \cdot h_3 + b_{out})$$

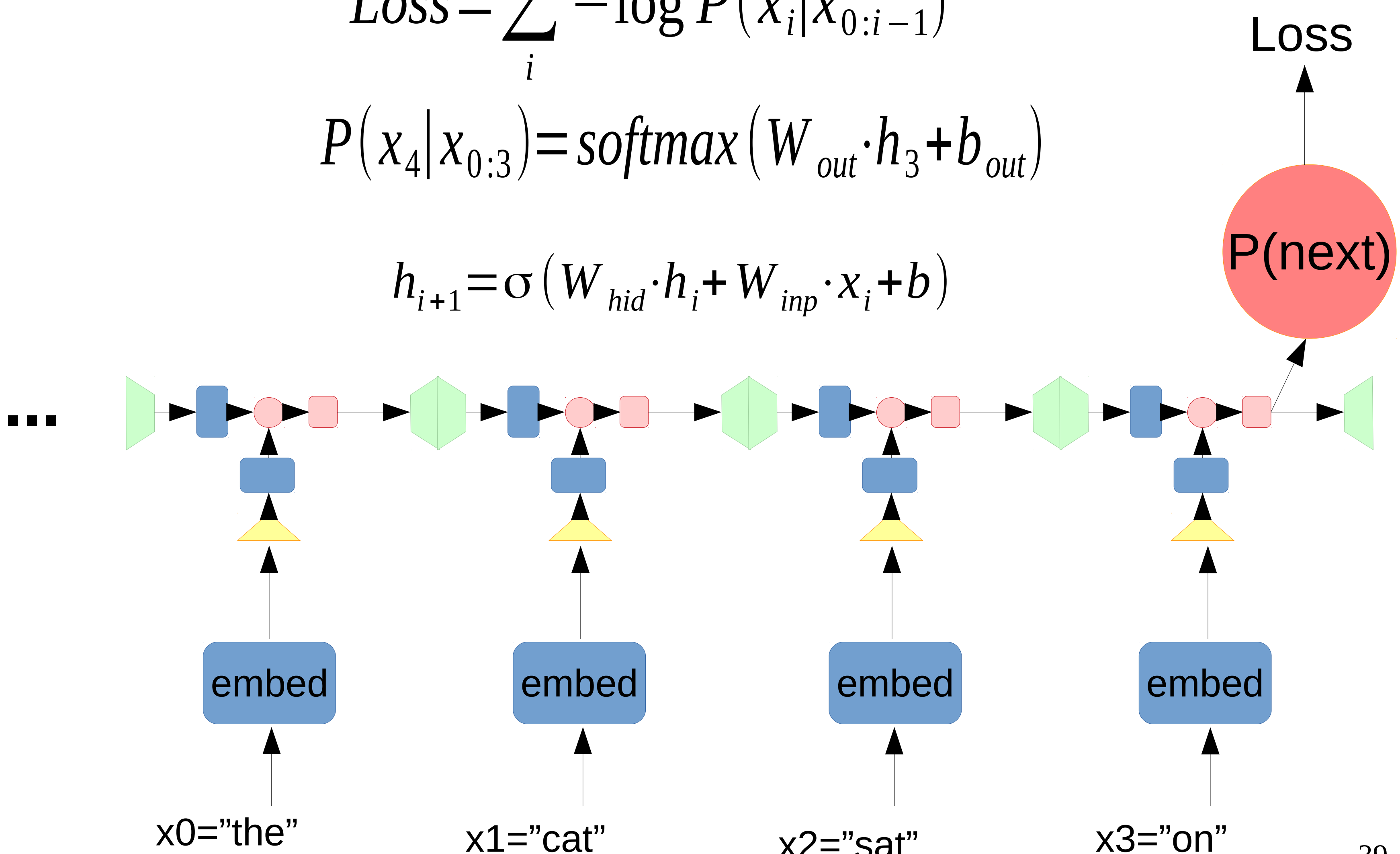


Recurrent neural network

$$Loss = \sum_i -\log P(x_i | x_{0:i-1})$$

$$P(x_4 | x_{0:3}) = softmax(W_{out} \cdot h_3 + b_{out})$$

$$h_{i+1} = \sigma(W_{hid} \cdot h_i + W_{inp} \cdot x_i + b)$$



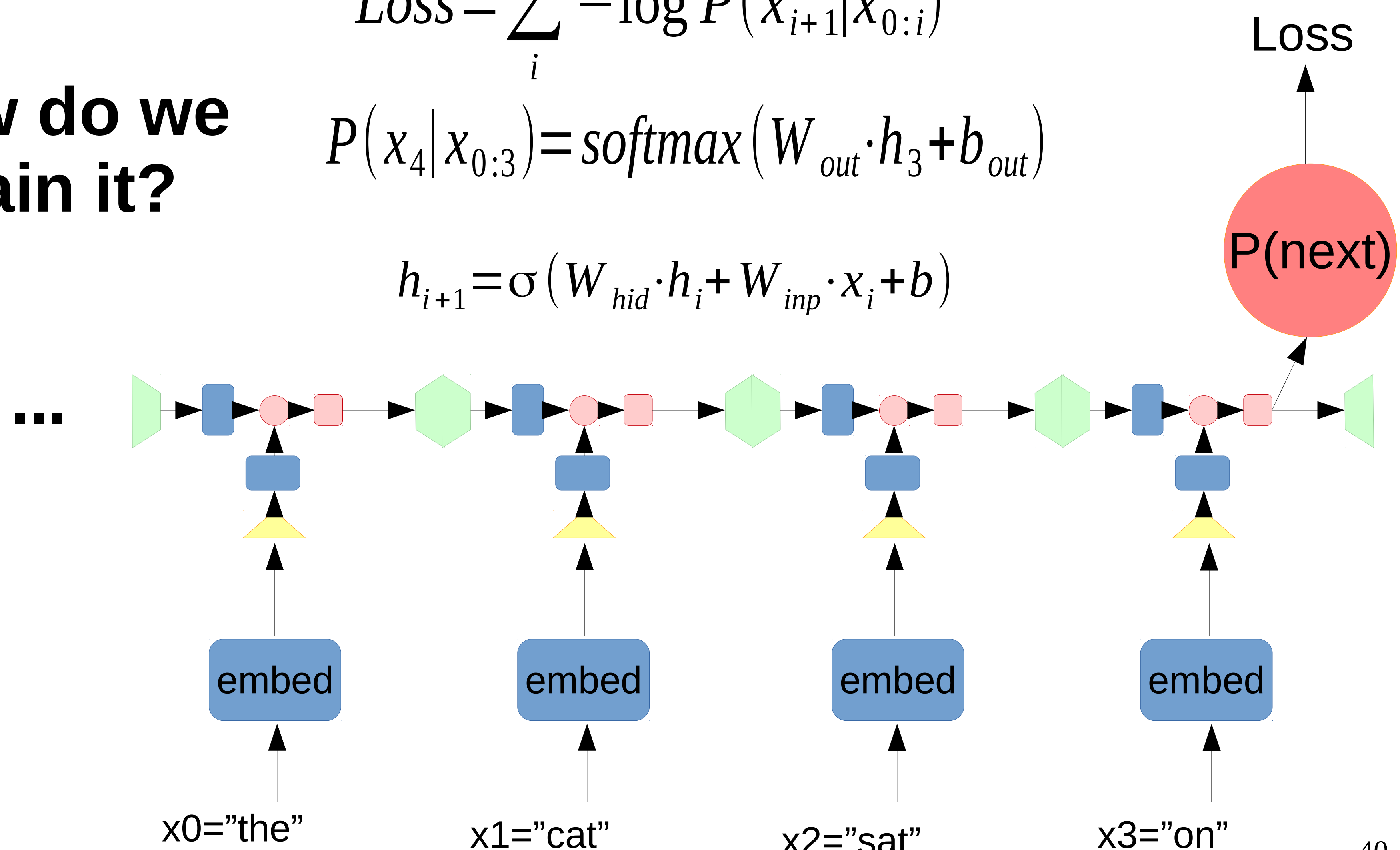
Recurrent neural network

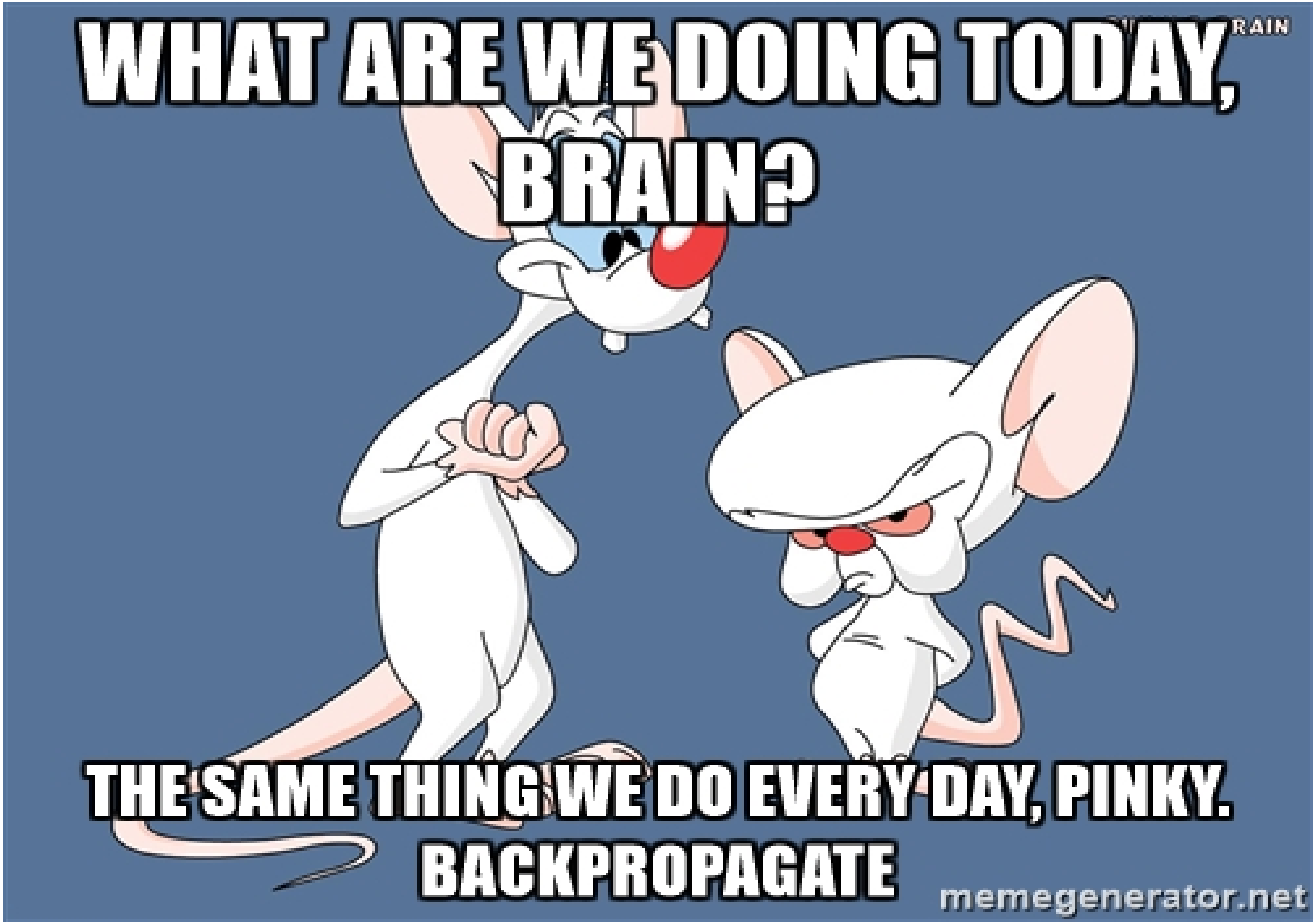
$$Loss = \sum_i -\log P(x_{i+1} | x_{0:i})$$

How do we train it?

$$P(x_4 | x_{0:3}) = softmax(W_{out} \cdot h_3 + b_{out})$$

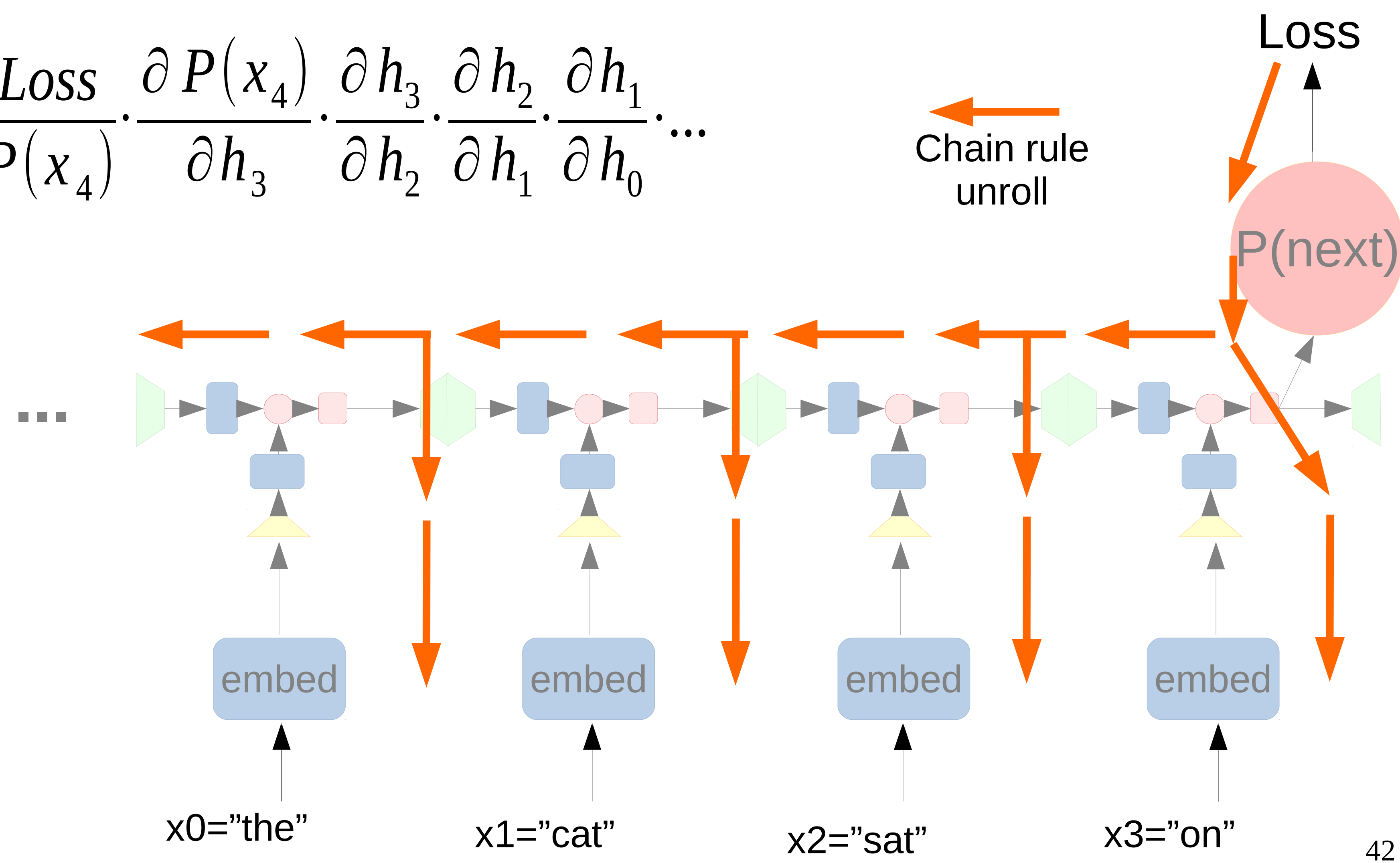
$$h_{i+1} = \sigma(W_{hid} \cdot h_i + W_{inp} \cdot x_i + b)$$




WHAT ARE WE DOING TODAY, BRAIN?
THE SAME THING WE DO EVERY DAY, PINKY. BACKPROPAGATE
memegenerator.net

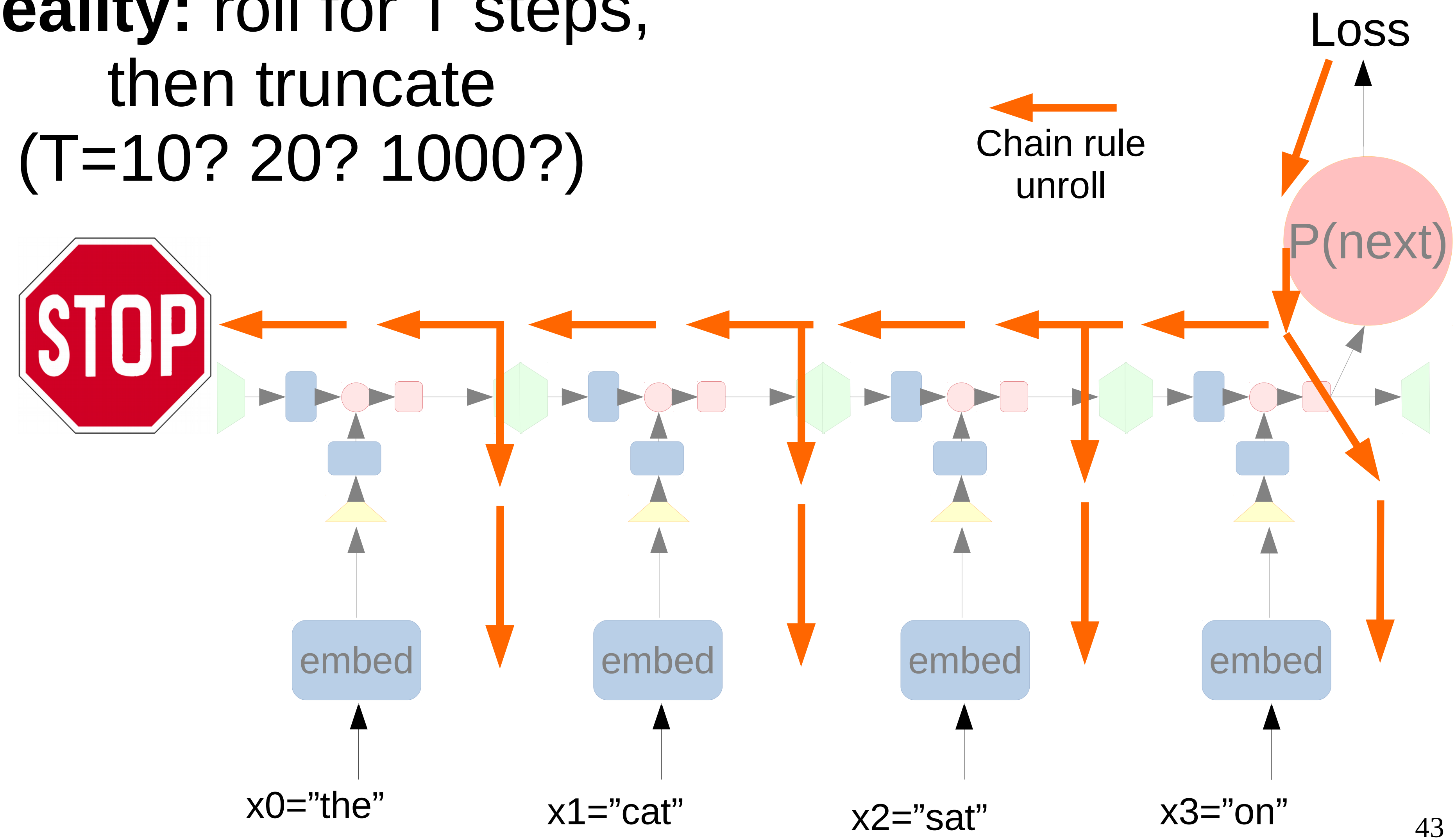
Backpropagation through time

$$\frac{\partial Loss}{\partial P(x_4)} \cdot \frac{\partial P(x_4)}{\partial h_3} \cdot \frac{\partial h_3}{\partial h_2} \cdot \frac{\partial h_2}{\partial h_1} \cdot \frac{\partial h_1}{\partial h_0} \cdot \ldots$$



Truncated BPTT

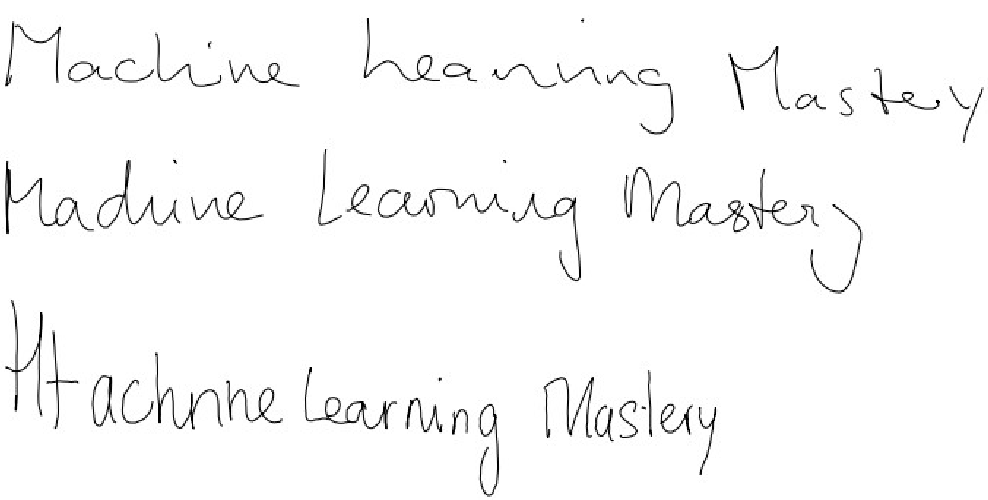
Reality: roll for T steps,
then truncate
(T=10? 20? 1000?)



End of part 1

Questions for coffee break:

A) how would you apply RNN to generate random handwriting?



B) how would you apply RNN for sentiment classification?

End of part 1

Questions for coffee break:

A) how would you apply RNN to generate random handwriting?

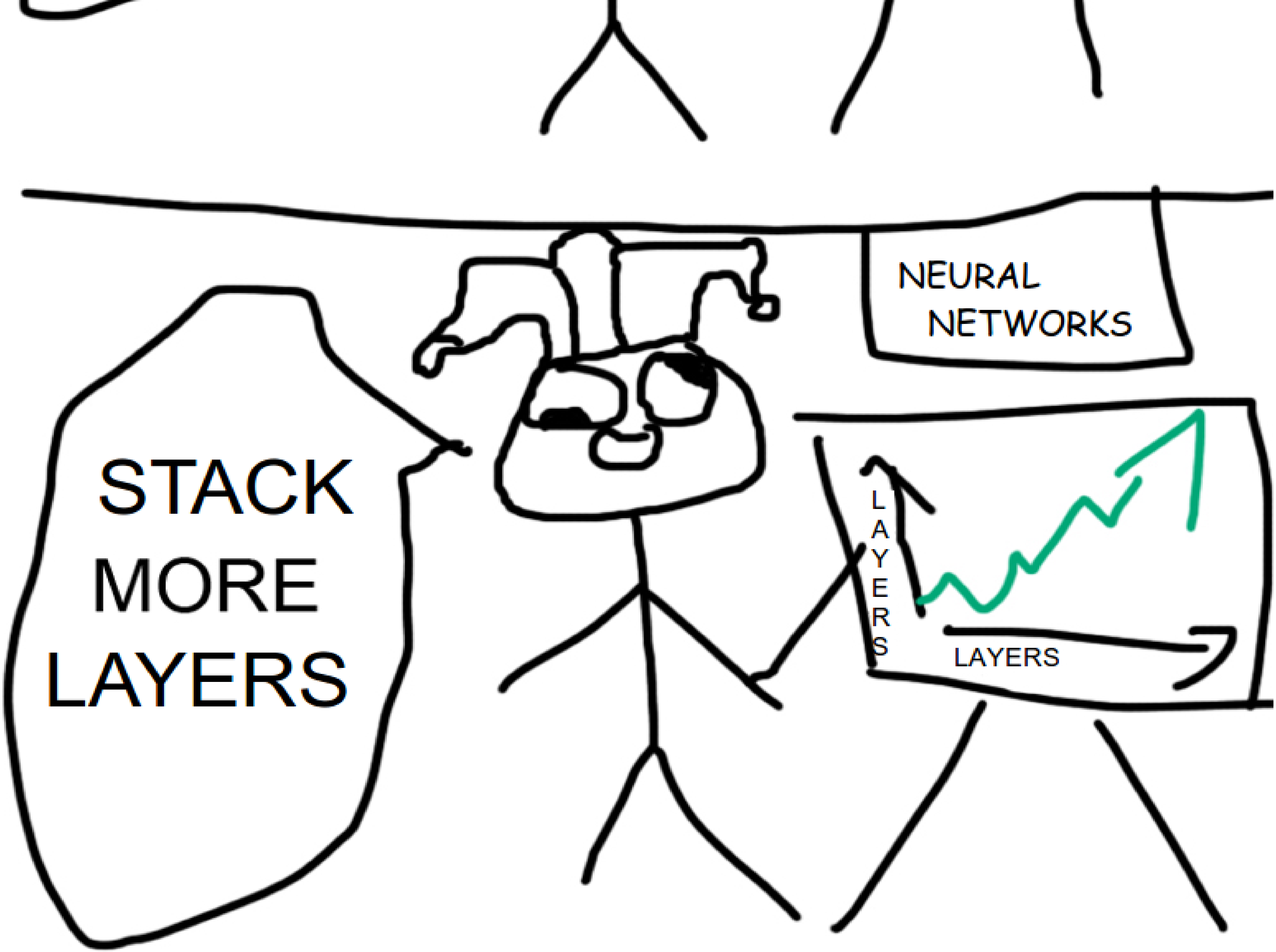
predict next pen position (or diff) minimize MSE

See bit.ly/2qq57wy

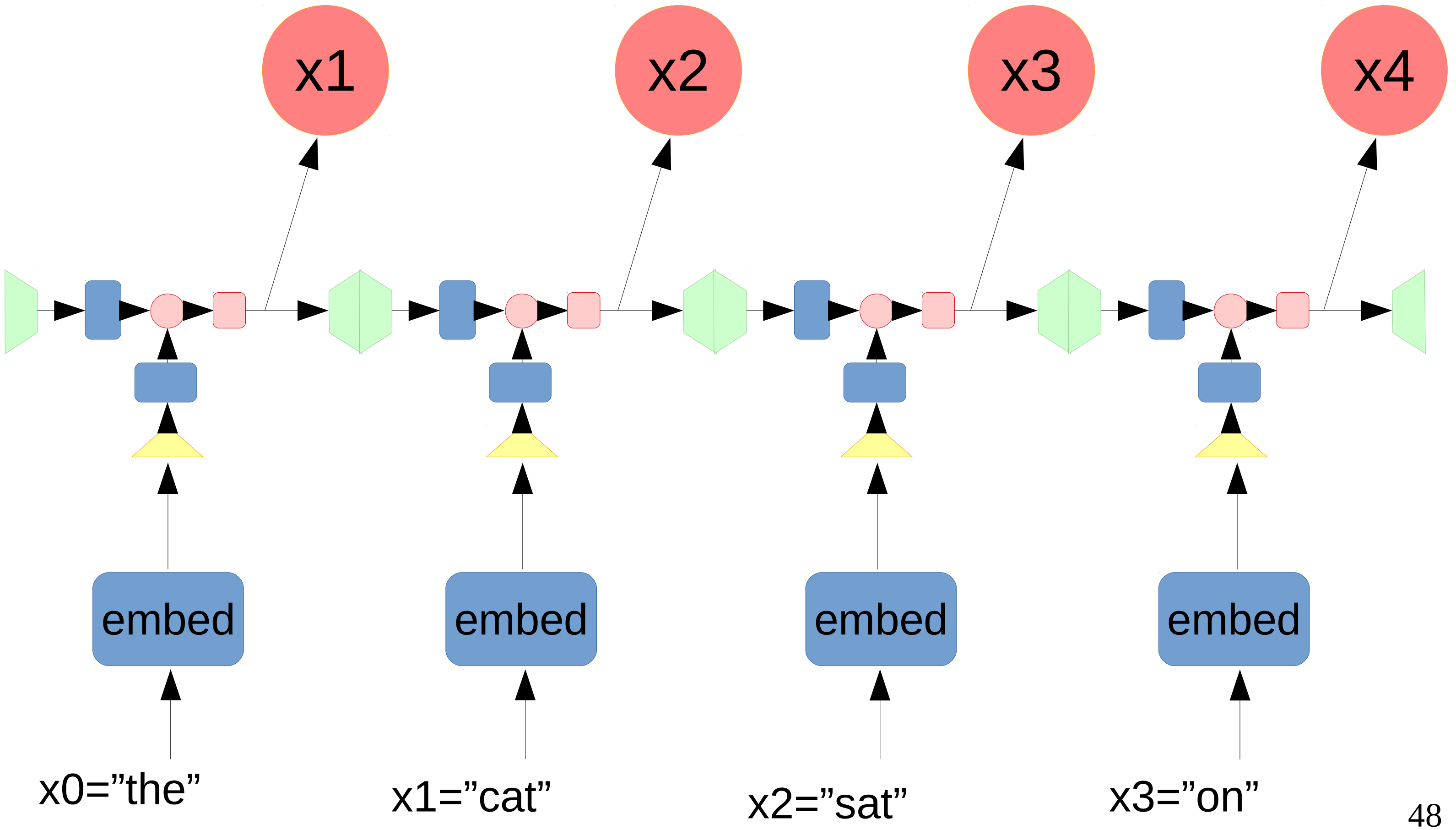
B) how would you apply RNN for sentiment classification?

Use last RNN state and predict sentiment with yet another dense layer w/ softmax

And our goal for part 2 is...

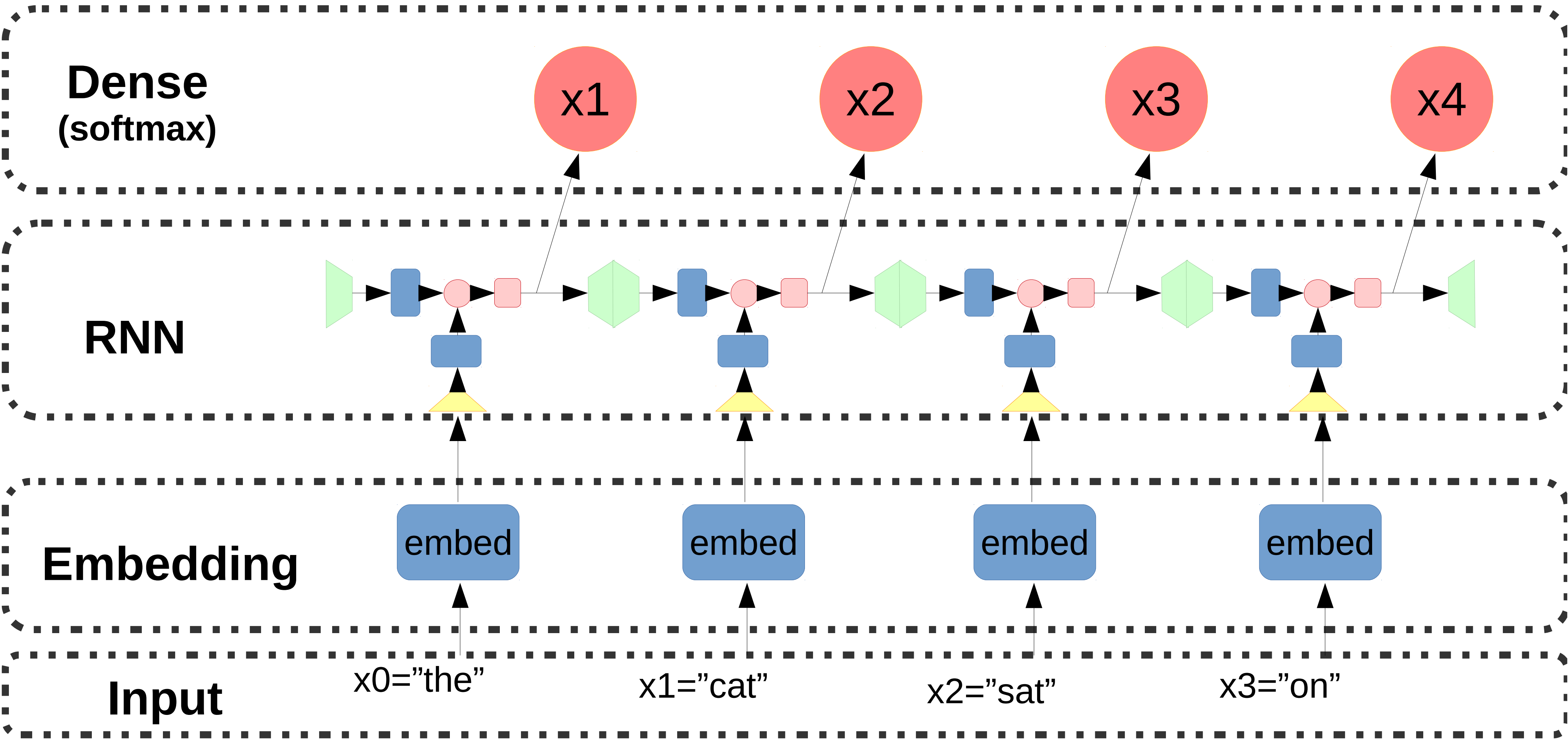

STACK MORE LAYERS
NEURAL NETWORKS
LAYERS
LAYERS

What is layer, again?

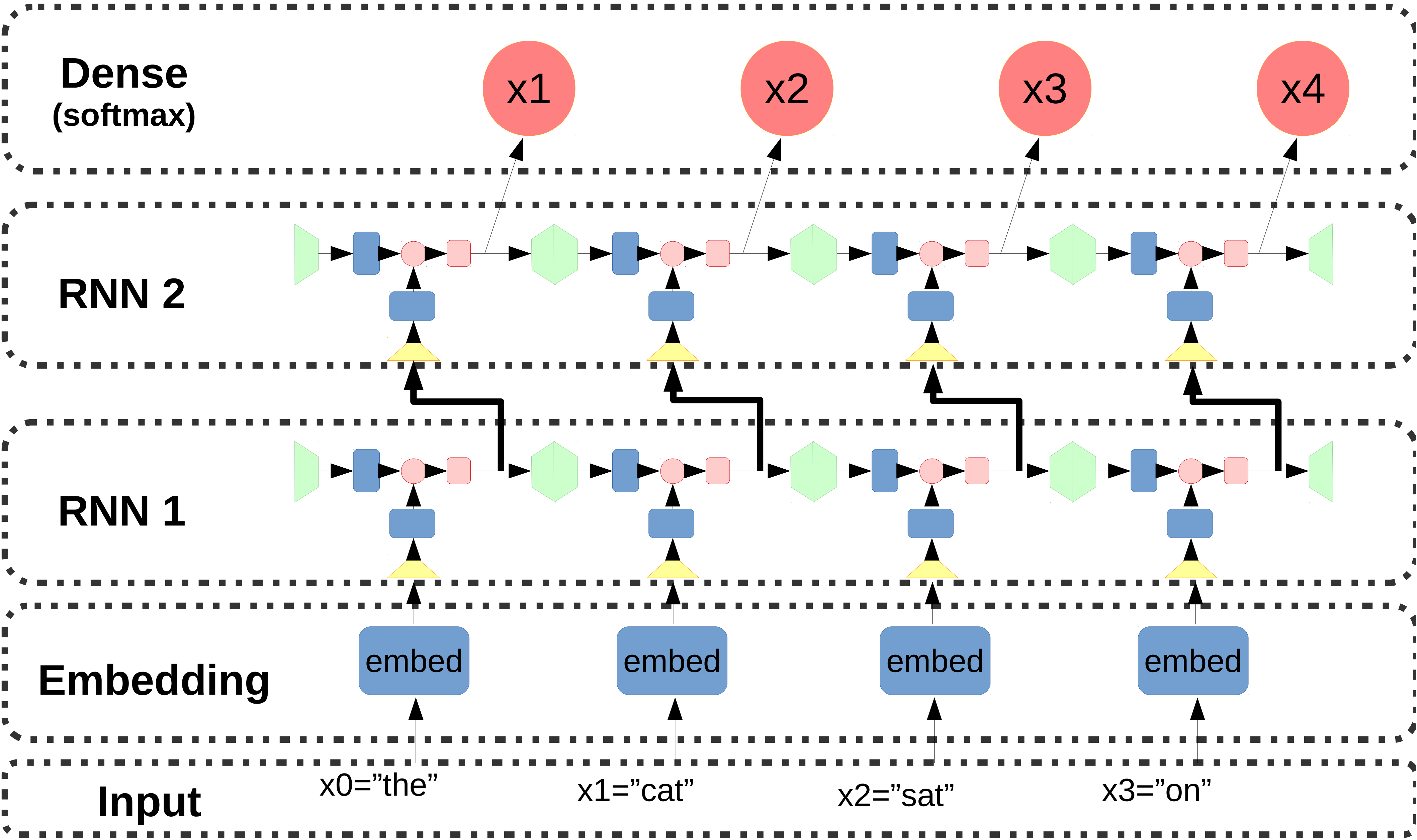


Layers

Where to stick more layers?

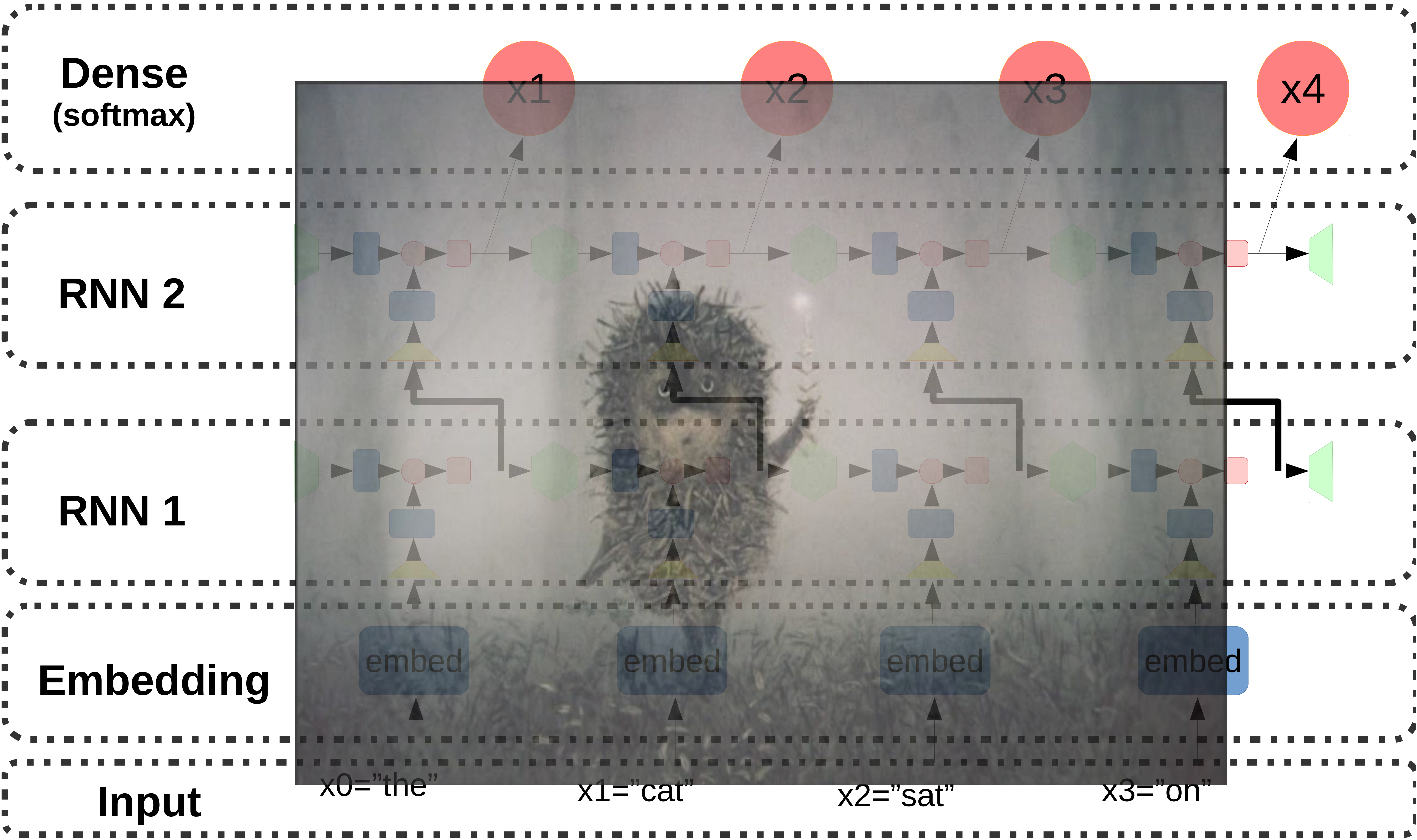


More layers

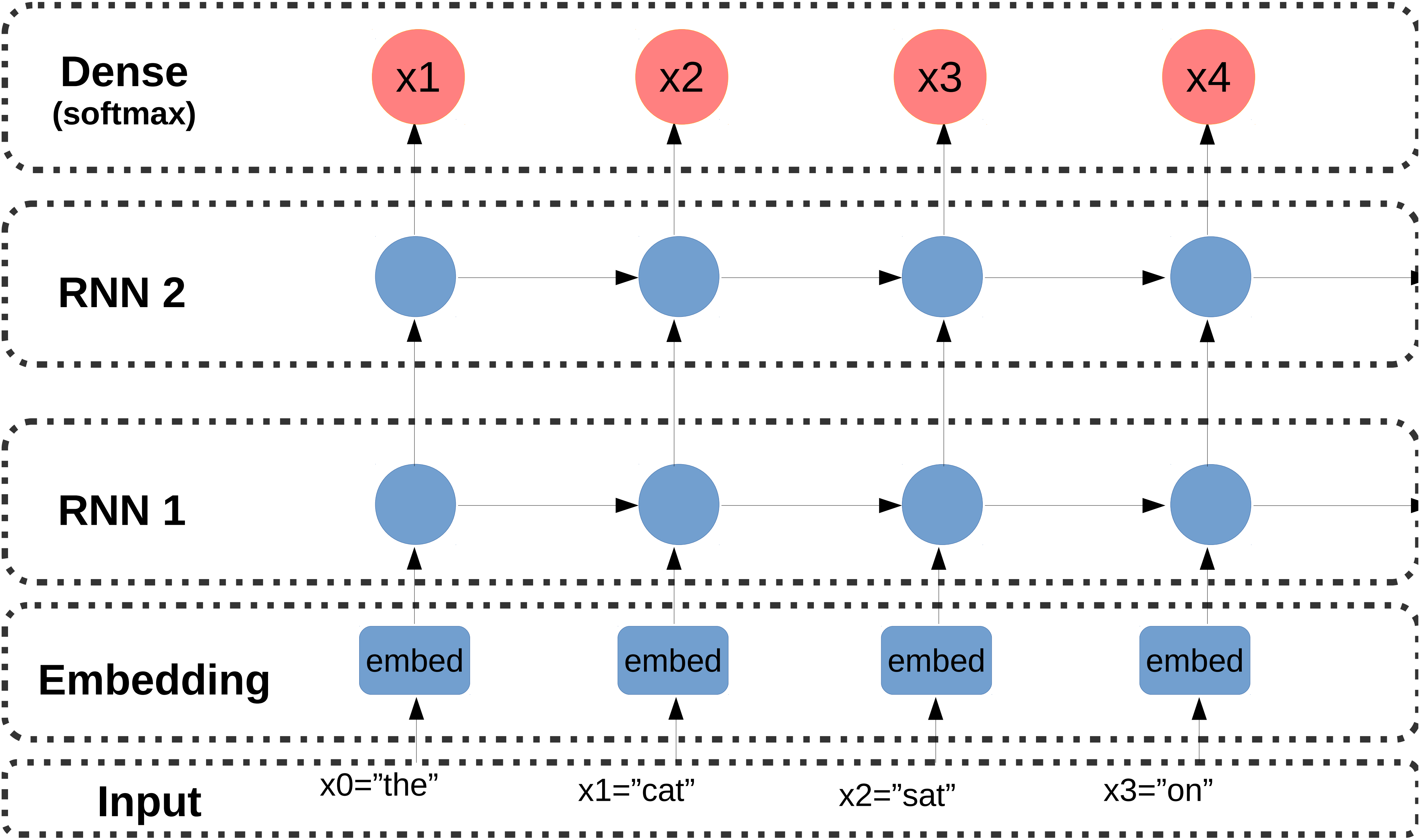


Dense
(softmax)
x1
x2
x3
x4
RNN 2
RNN 1
Embedding
embed
embed
embed
embed
Input
x0="the"
x1="cat"
x2="sat"
x3="on"

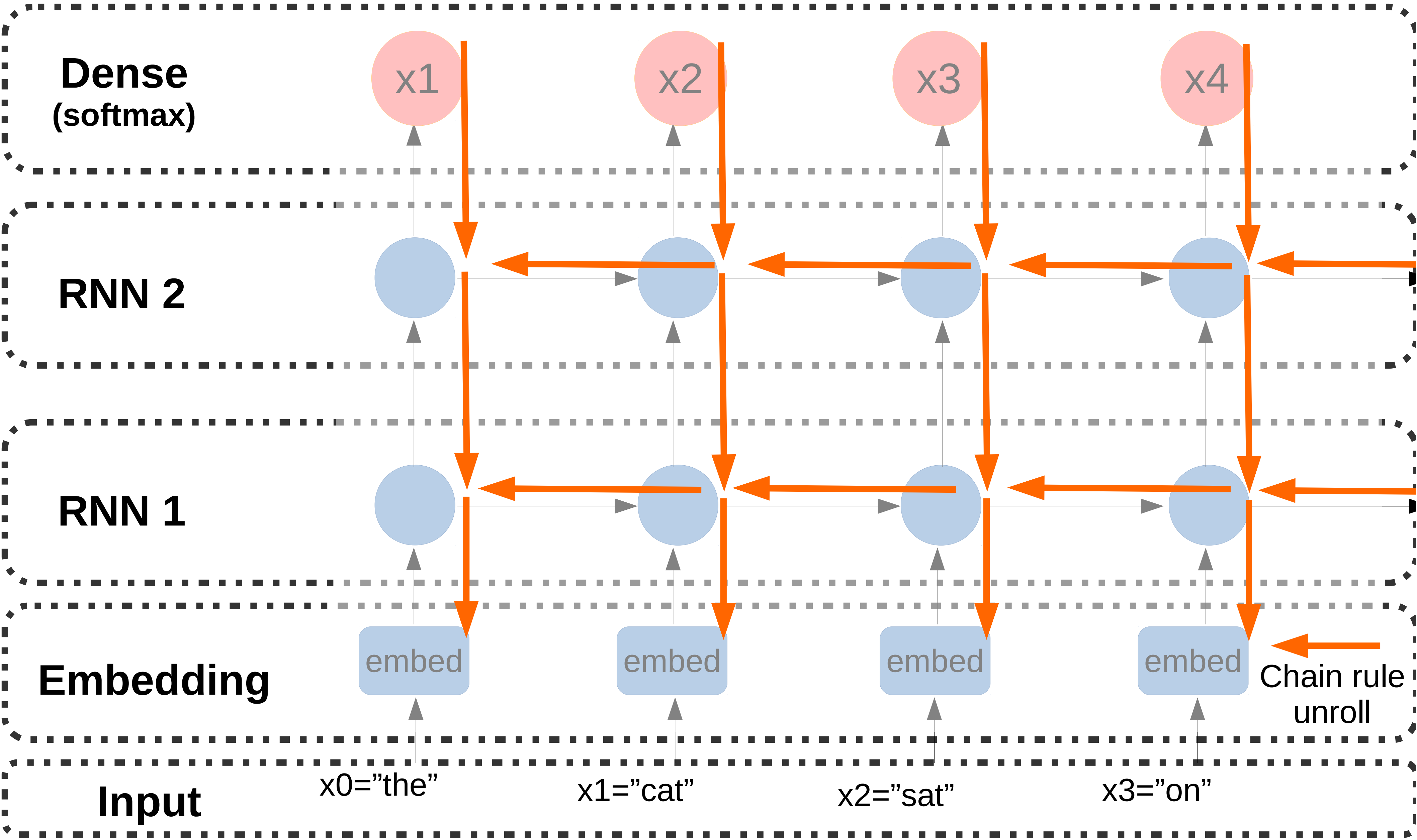
Too f**king complicated



2-layer RNN


Dense
(softmax)
x1
x2
x3
x4
RNN 2
RNN 1
Embedding
embed
embed
embed
embed
Input
x0=”the”
x1=”cat”
x2=”sat”
x3=”on”

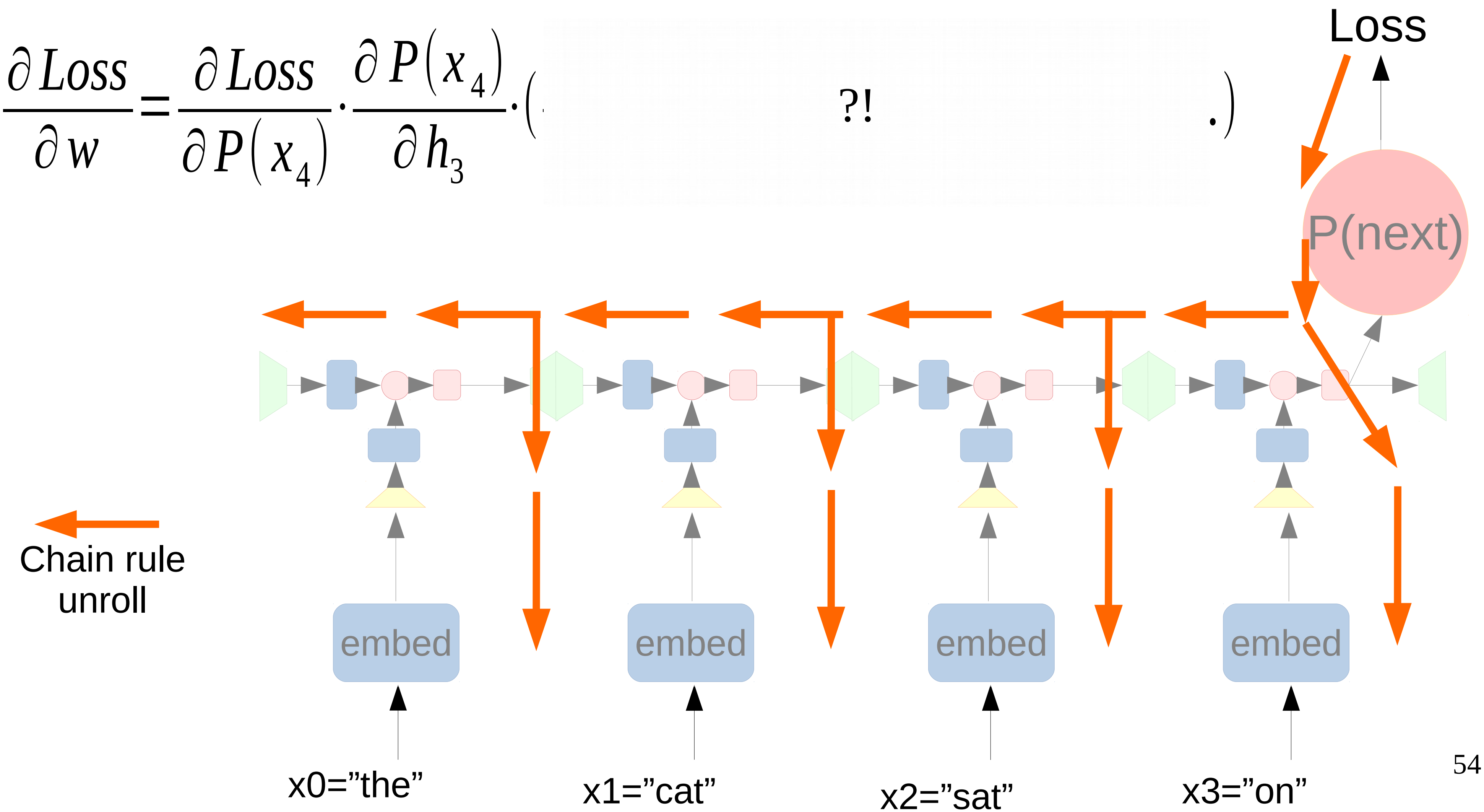
BPTT again


Dense
(softmax)
x1
x2
x3
x4
RNN 2
RNN 1
Embedding
embed
embed
embed
embed
Chain rule
unroll
Input
x0="the"
x1="cat"
x2="sat"
x3="on"

BPTT Again

$$h_{i+1} = \sigma\left(W_{hid} \cdot h_i + W_{inp} \cdot x_i + b\right)$$

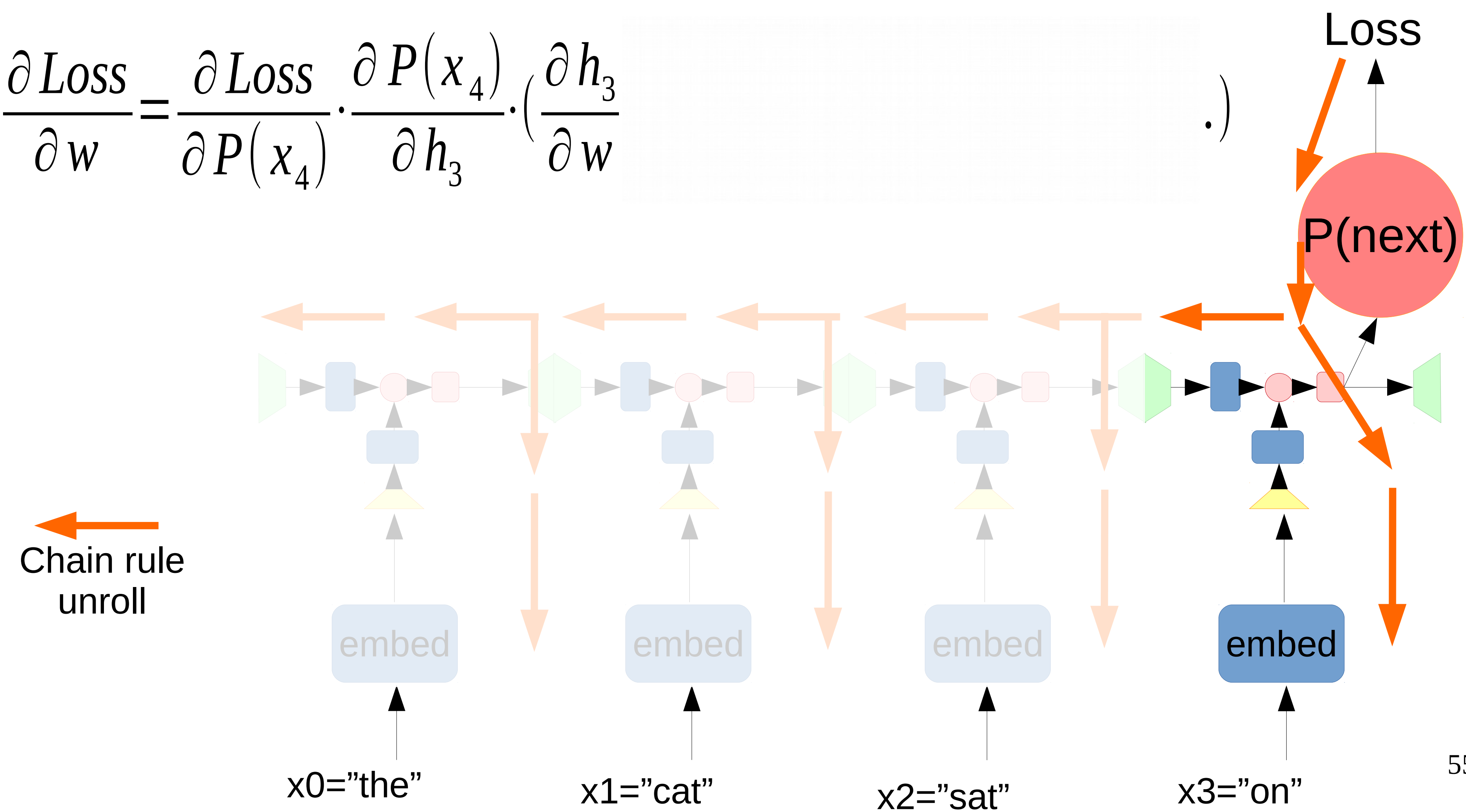
$$\frac{\partial Loss}{\partial w} = \frac{\partial Loss}{\partial P(x_4)} \cdot \frac{\partial P(x_4)}{\partial h_3} \cdot \left(\quad ?! \quad . \right)$$



BPTT Again

$$h_{i+1} = \sigma\left(W_{hid} \cdot h_i + W_{inp} \cdot x_i + b\right)$$

$$\frac{\partial Loss}{\partial w} = \frac{\partial Loss}{\partial P(x_4)} \cdot \frac{\partial P(x_4)}{\partial h_3} \cdot \left(\frac{\partial h_3}{\partial w} \qquad .\right)$$

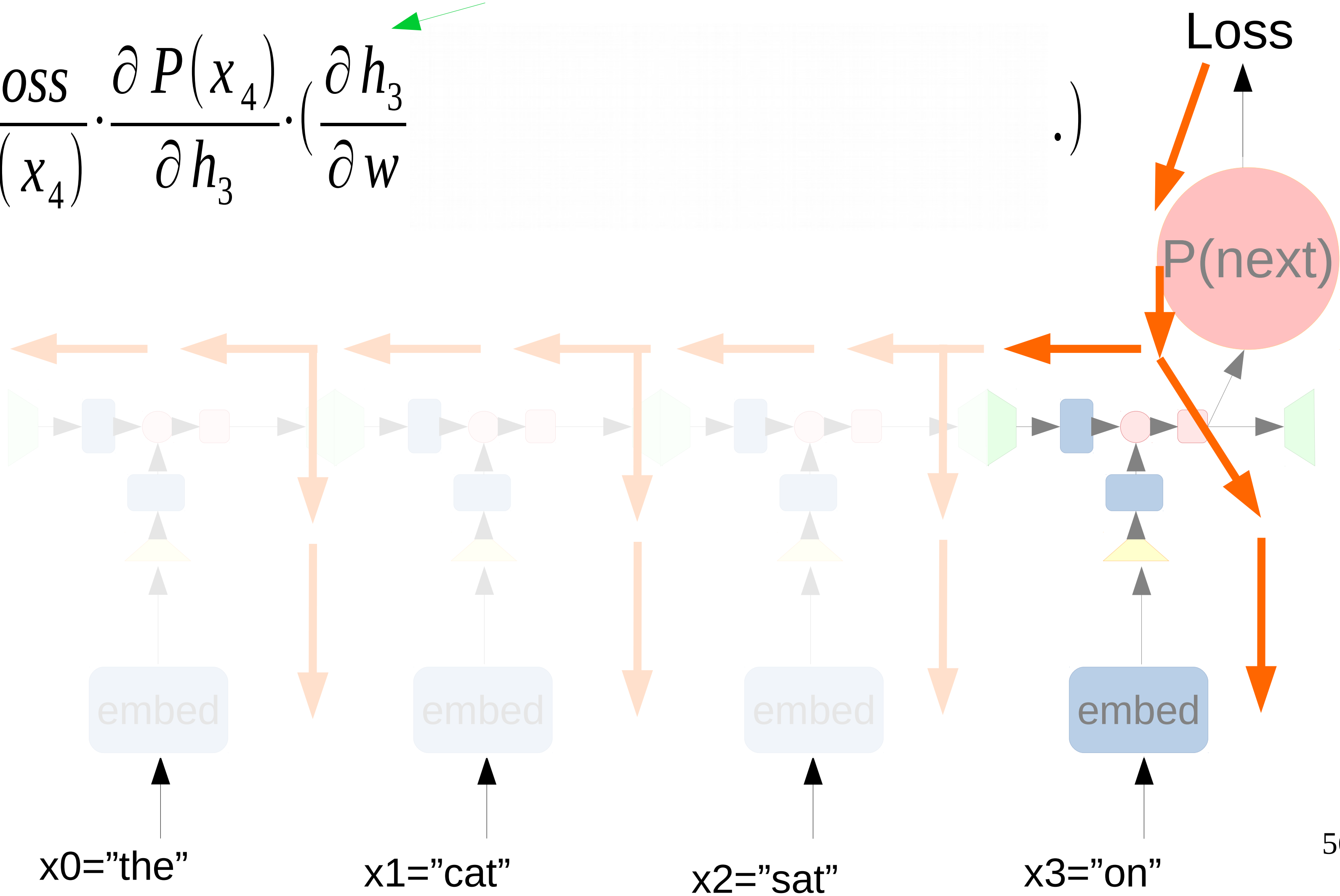


BPTT Again

$$h_{i+1} = \sigma\left(W_{hid} \cdot h_i + W_{inp} \cdot x_i + b\right)$$

consider h2 constant

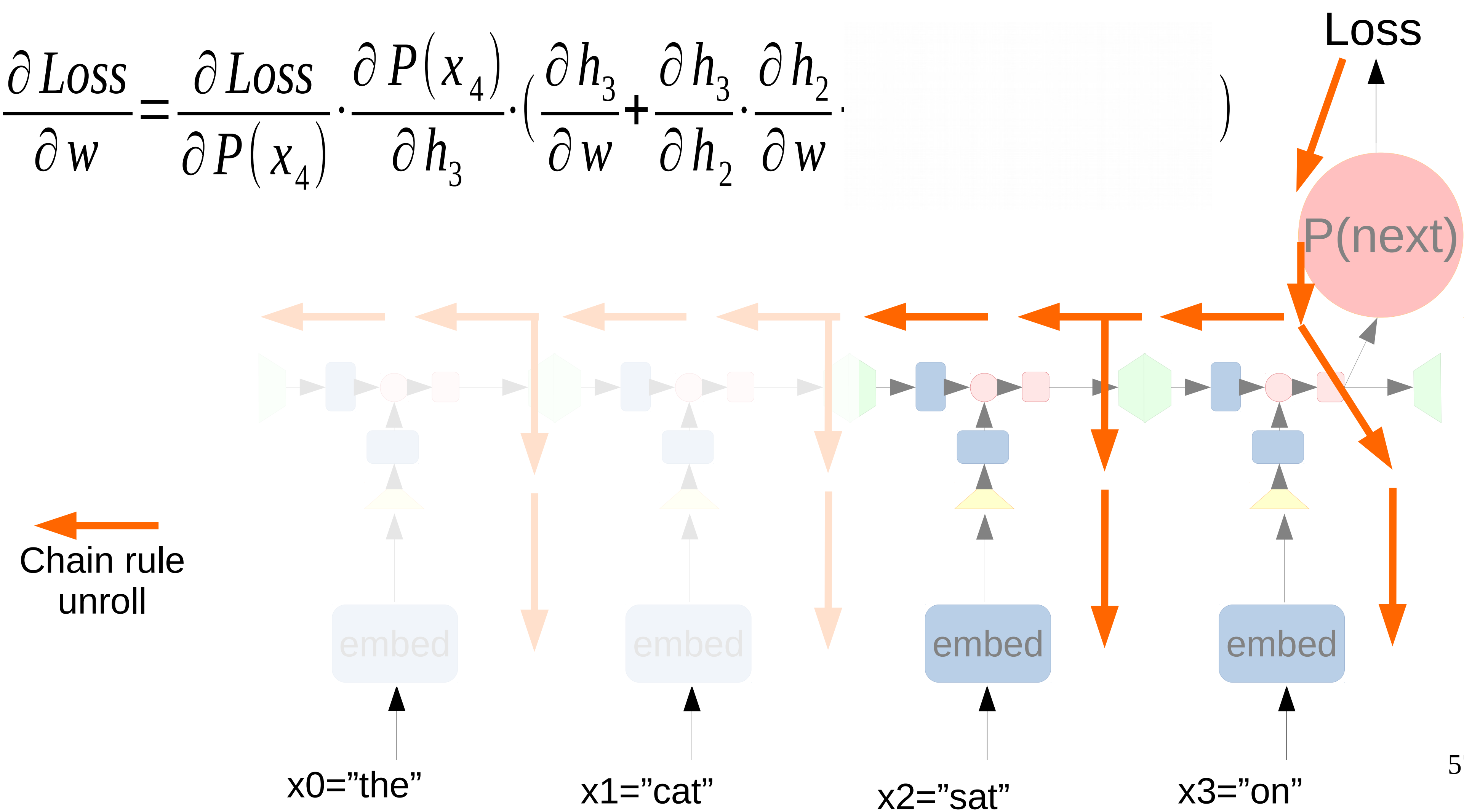
$$\frac{\partial Loss}{\partial w} = \frac{\partial Loss}{\partial P(x_4)} \cdot \frac{\partial P(x_4)}{\partial h_3} \cdot \left(\frac{\partial h_3}{\partial w} \qquad .\right)$$



BPTT Again

$$h_{i+1} = \sigma\left(W_{hid} \cdot h_i + W_{inp} \cdot x_i + b\right)$$

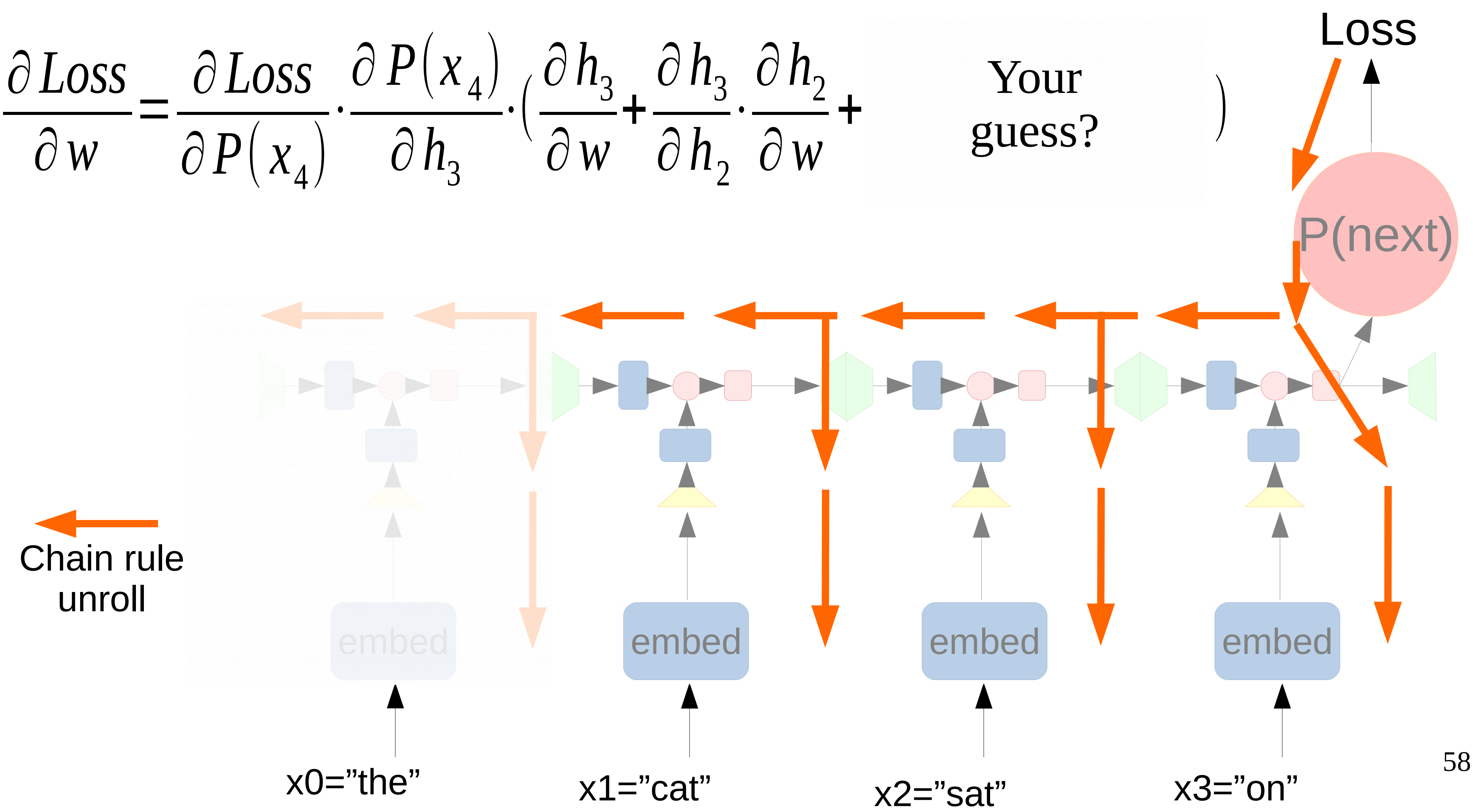
$$\frac{\partial Loss}{\partial w} = \frac{\partial Loss}{\partial P(x_4)} \cdot \frac{\partial P(x_4)}{\partial h_3} \cdot \left(\frac{\partial h_3}{\partial w} + \frac{\partial h_3}{\partial h_2} \cdot \frac{\partial h_2}{\partial w} \cdot \qquad \right)$$



BPTT Again

$$h_{i+1} = \sigma\left(W_{hid} \cdot h_i + W_{inp} \cdot x_i + b\right)$$

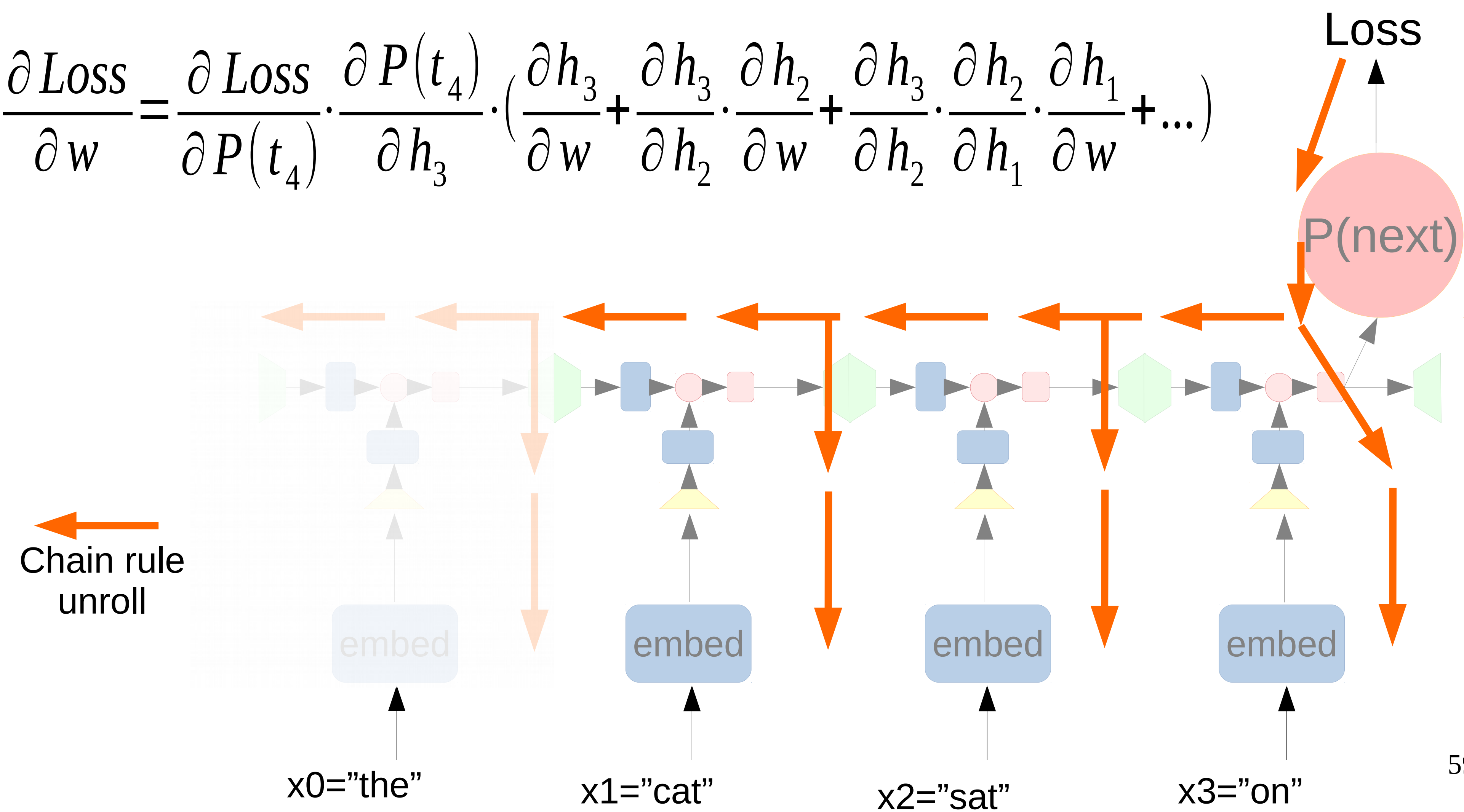
$$\frac{\partial Loss}{\partial w} = \frac{\partial Loss}{\partial P(x_4)} \cdot \frac{\partial P(x_4)}{\partial h_3} \cdot \left(\frac{\partial h_3}{\partial w} + \frac{\partial h_3}{\partial h_2} \cdot \frac{\partial h_2}{\partial w} + \quad \text{Your guess?} \quad \right)$$



BPTT Again

$$h_{i+1} = \sigma\left(W_{hid} \cdot h_i + W_{inp} \cdot x_i + b\right)$$

$$\frac{\partial Loss}{\partial w} = \frac{\partial Loss}{\partial P(t_4)} \cdot \frac{\partial P(t_4)}{\partial h_3} \cdot \left(\frac{\partial h_3}{\partial w} + \frac{\partial h_3}{\partial h_2} \cdot \frac{\partial h_2}{\partial w} + \frac{\partial h_3}{\partial h_2} \cdot \frac{\partial h_2}{\partial h_1} \cdot \frac{\partial h_1}{\partial w} + \ldots\right)$$



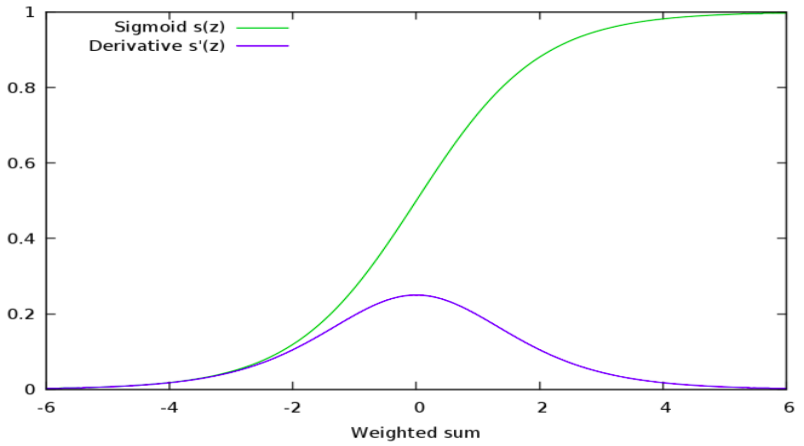
Gradient explosion and vanishing

$$h_{i+1} = \sigma\left(W_{hid} \cdot h_i + W_{inp} \cdot x_i + b\right)$$

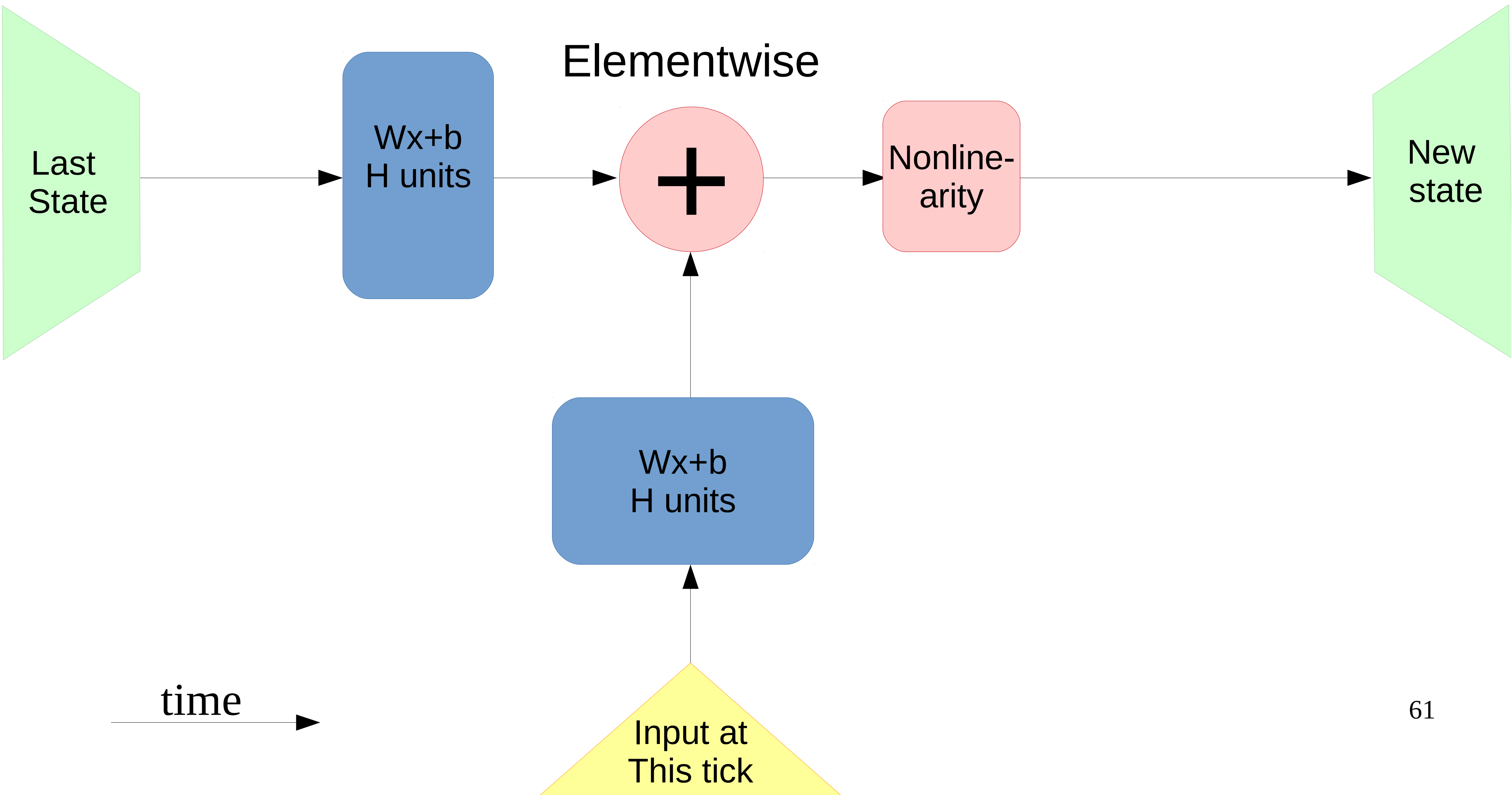
$$\frac{\partial Loss}{\partial w} = \frac{\partial Loss}{\partial P\left(x_4\right)} \cdot \frac{\partial P\left(x_4\right)}{\partial h_3} \cdot \left(\frac{\partial h_3}{\partial w} + \frac{\partial h_3}{\partial h_2} \cdot \frac{\partial h_2}{\partial w} + \frac{\partial h_3}{\partial h_2} \cdot \frac{\partial h_2}{\partial h_1} \cdot \frac{\partial h_1}{\partial w} + \ldots\right)$$

- Many sigmoids near 0 or 1
  - Gradients → 0
  - Not training for long-term dependencies

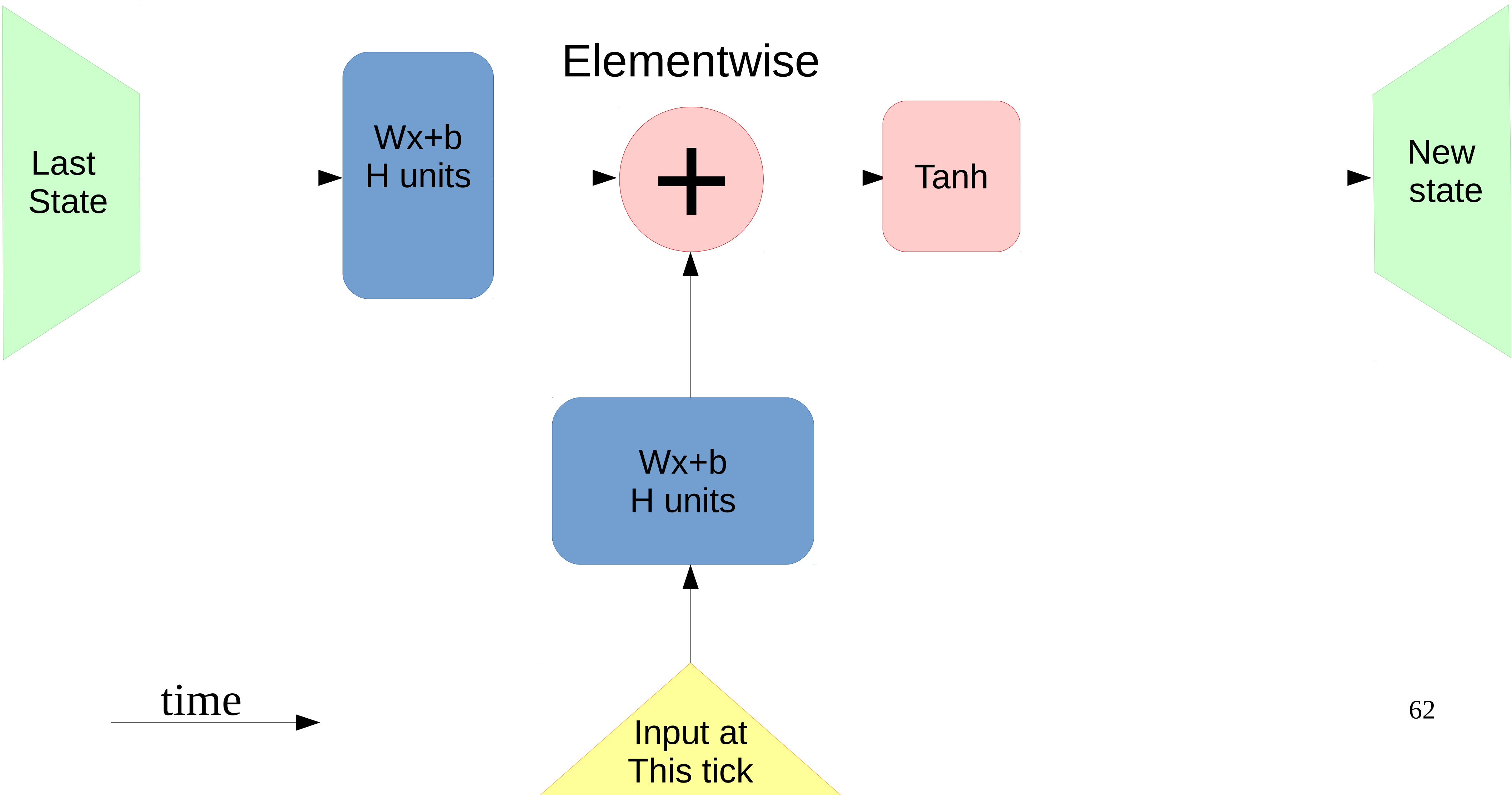
- Many nonzero values
  - Derivative stacks to >1
  - Gradients → inf
  - Weights → shit



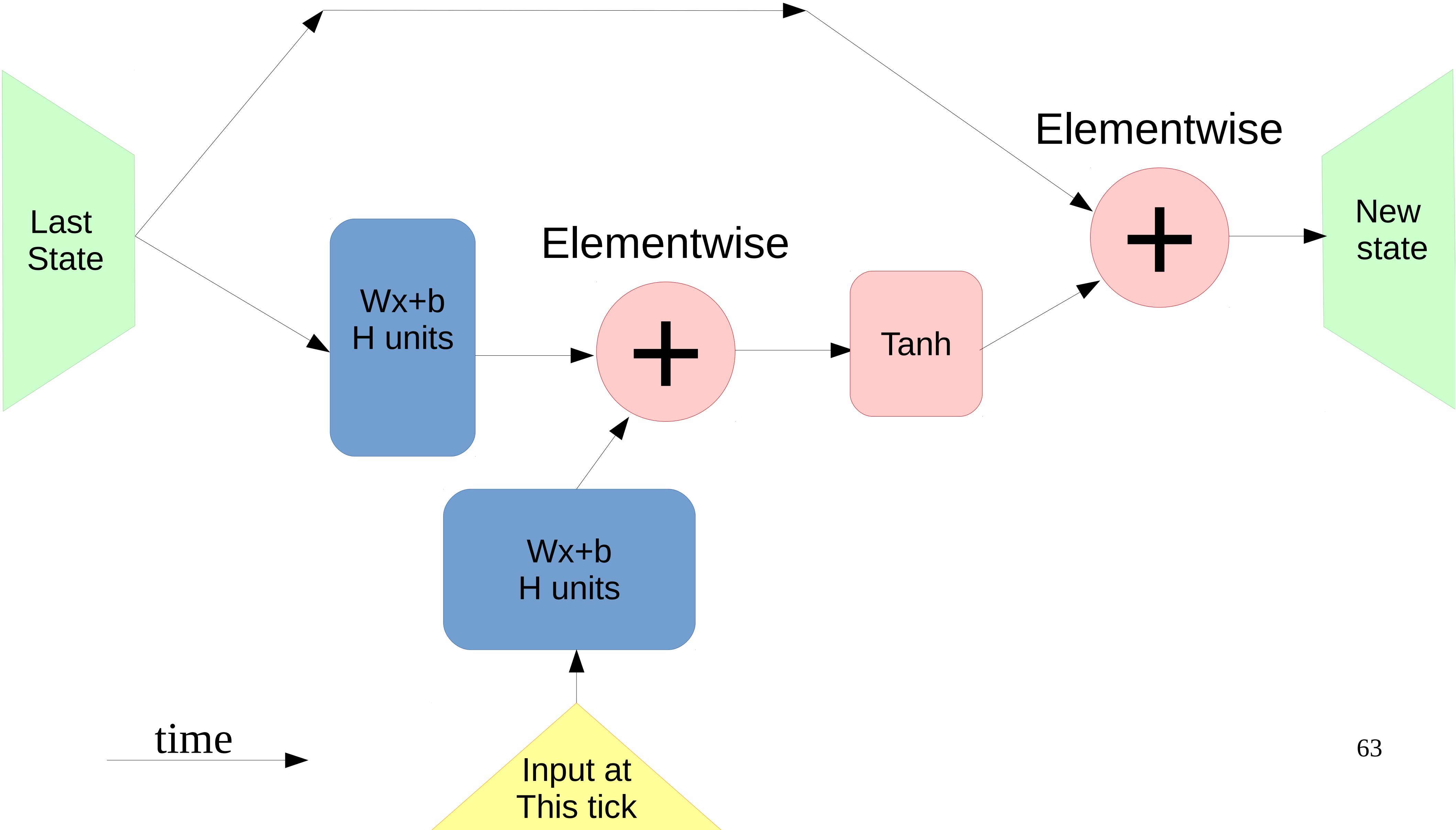
RNN step


Last
State
Wx+b
H units
Elementwise
+
Nonline-
arity
New
state
Wx+b
H units
Input at
This tick
time

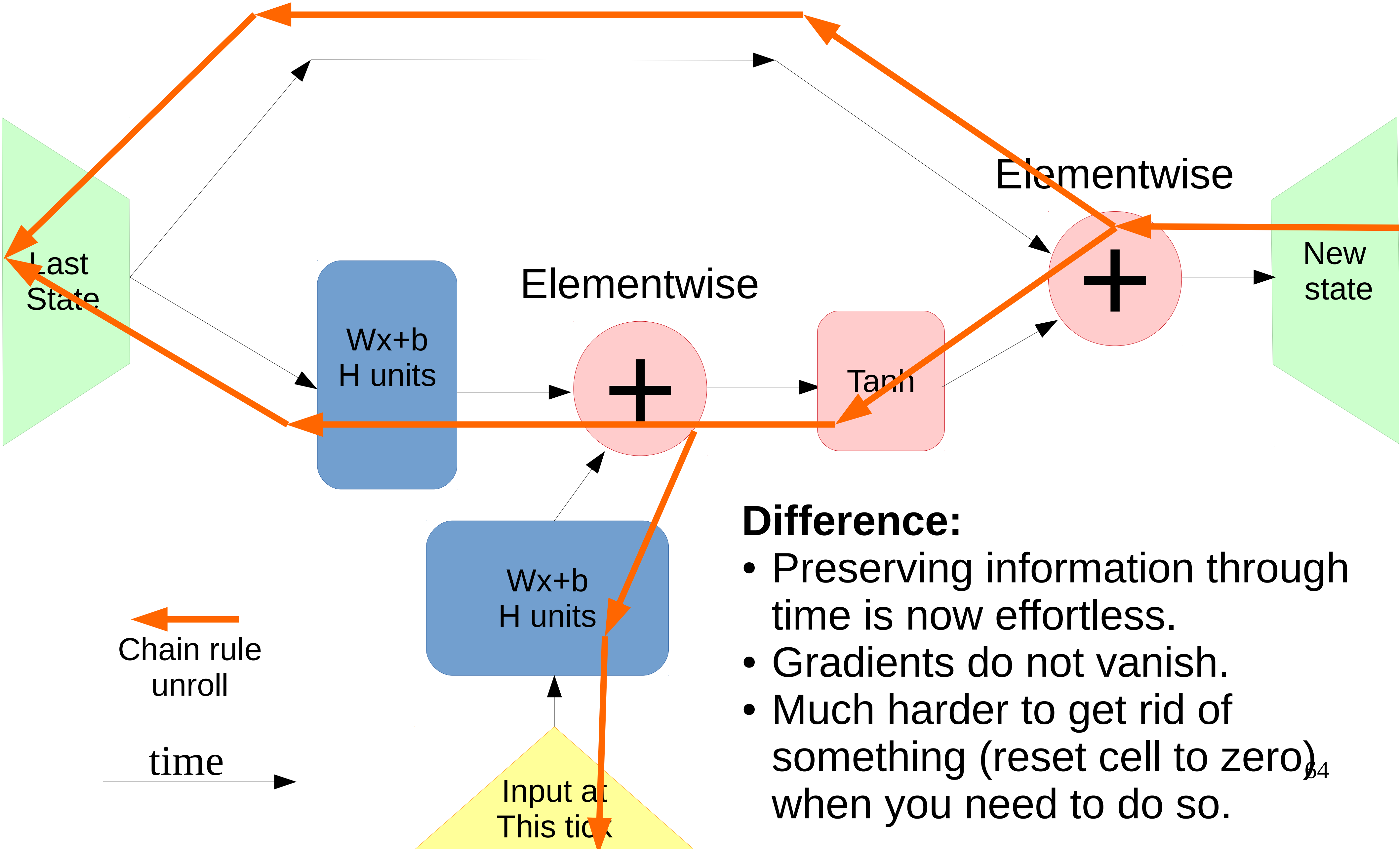
RNN step


Elementwise
Last
State
Wx+b
H units
+
Tanh
New
state
Wx+b
H units
Input at
This tick
time

Residual RNN step



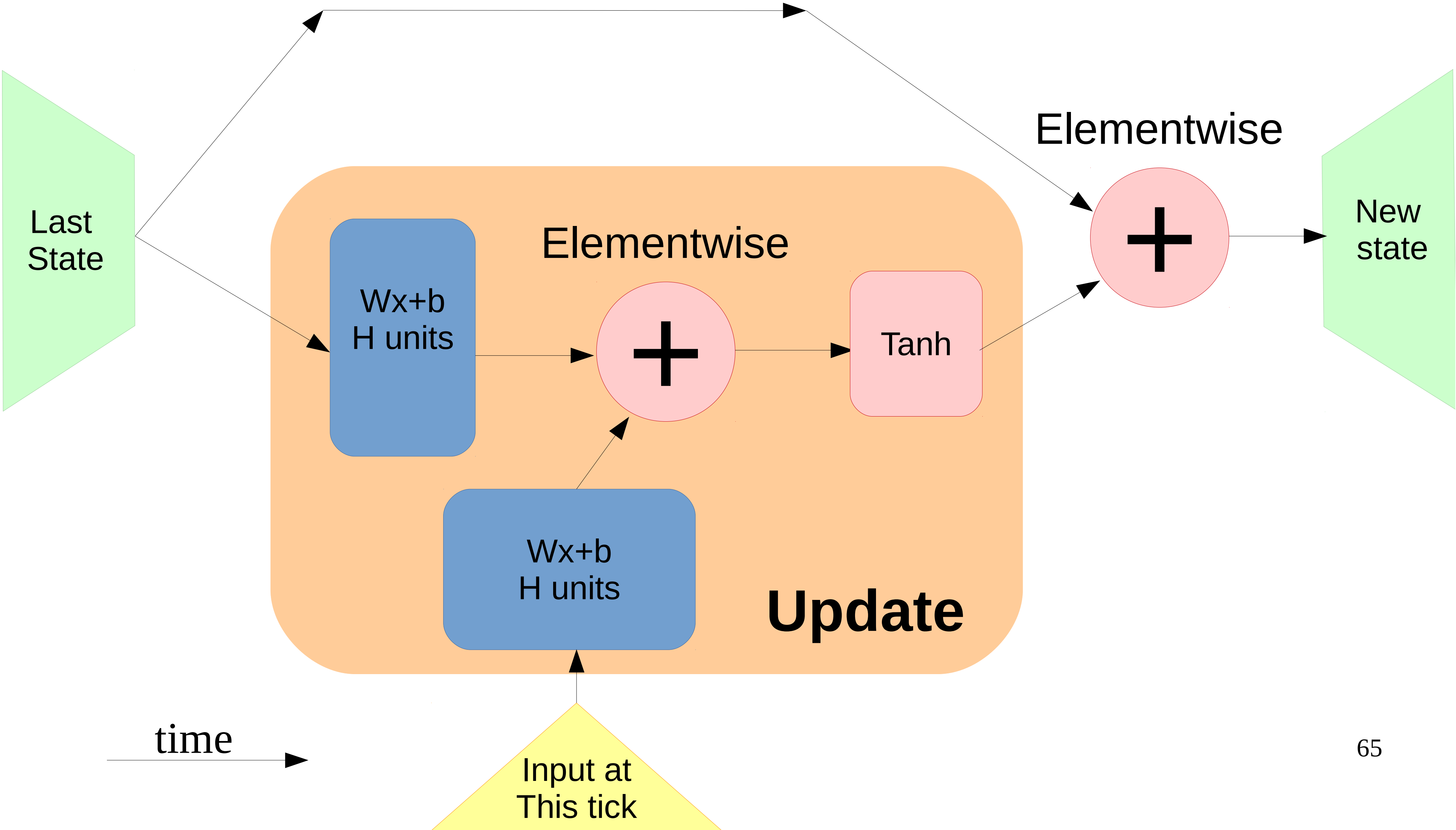
Residual RNN step



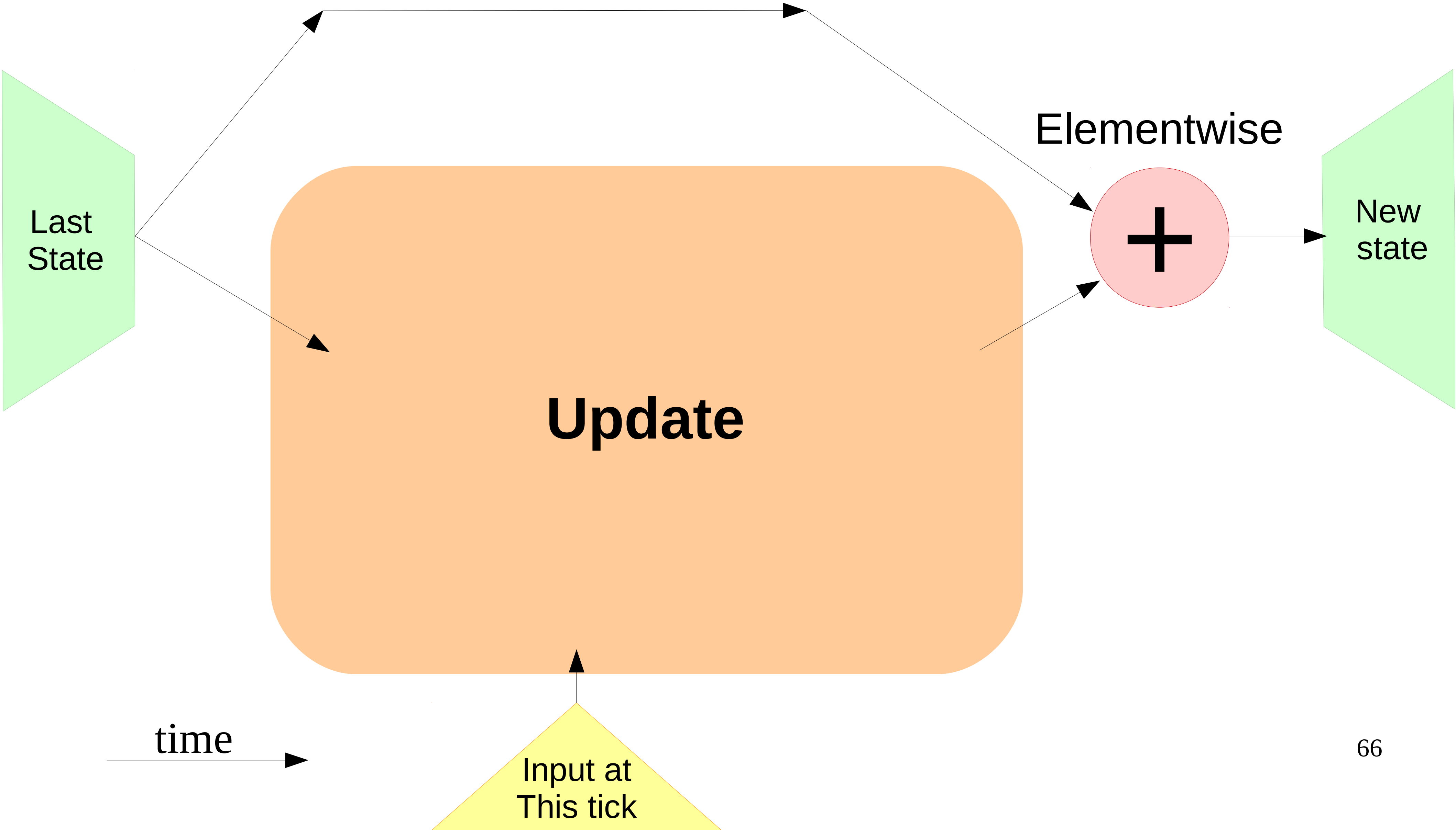
Difference:

- Preserving information through time is now effortless.
- Gradients do not vanish.
- Much harder to get rid of something (reset cell to zero) when you need to do so.

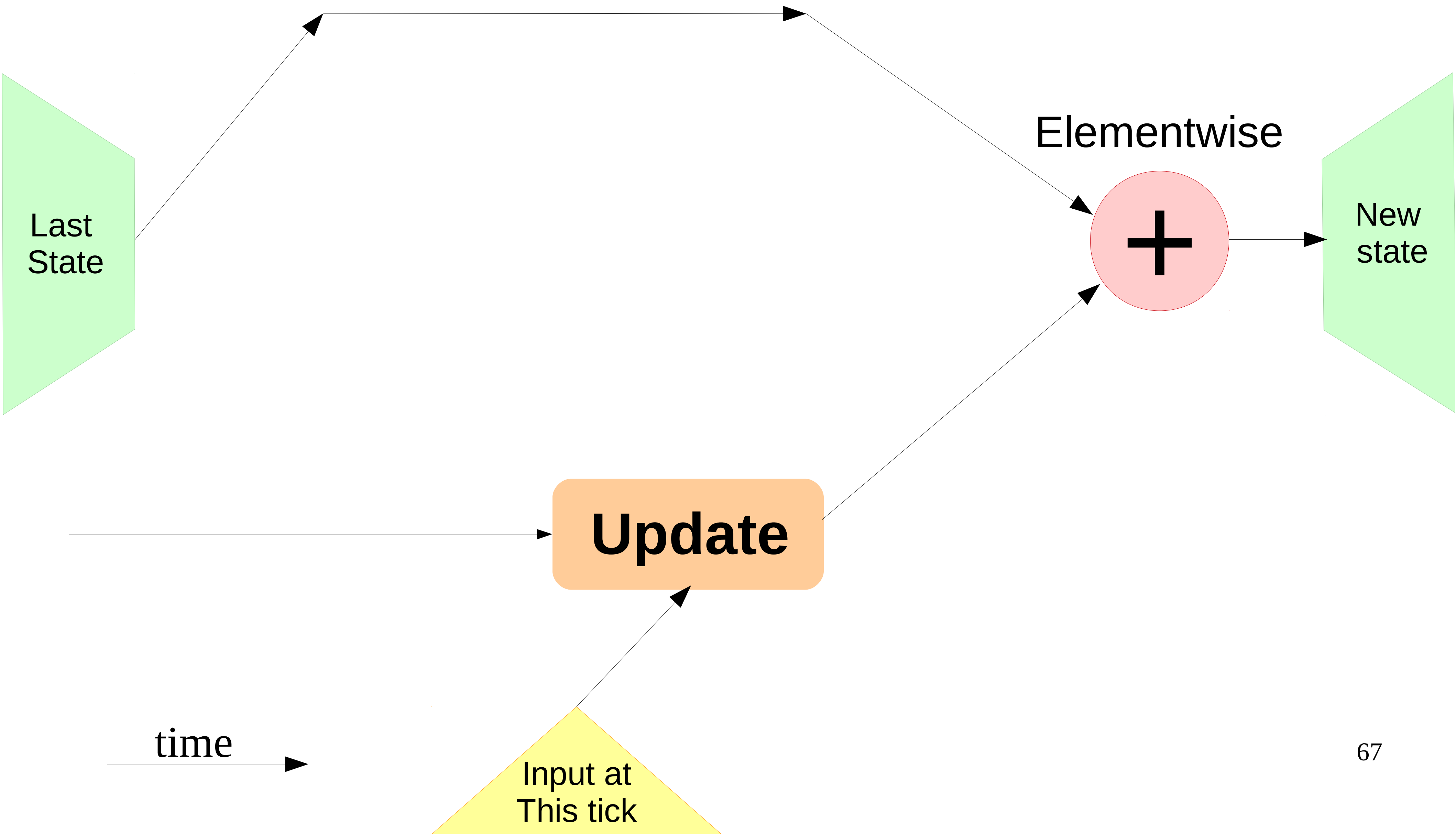
Residual RNN step



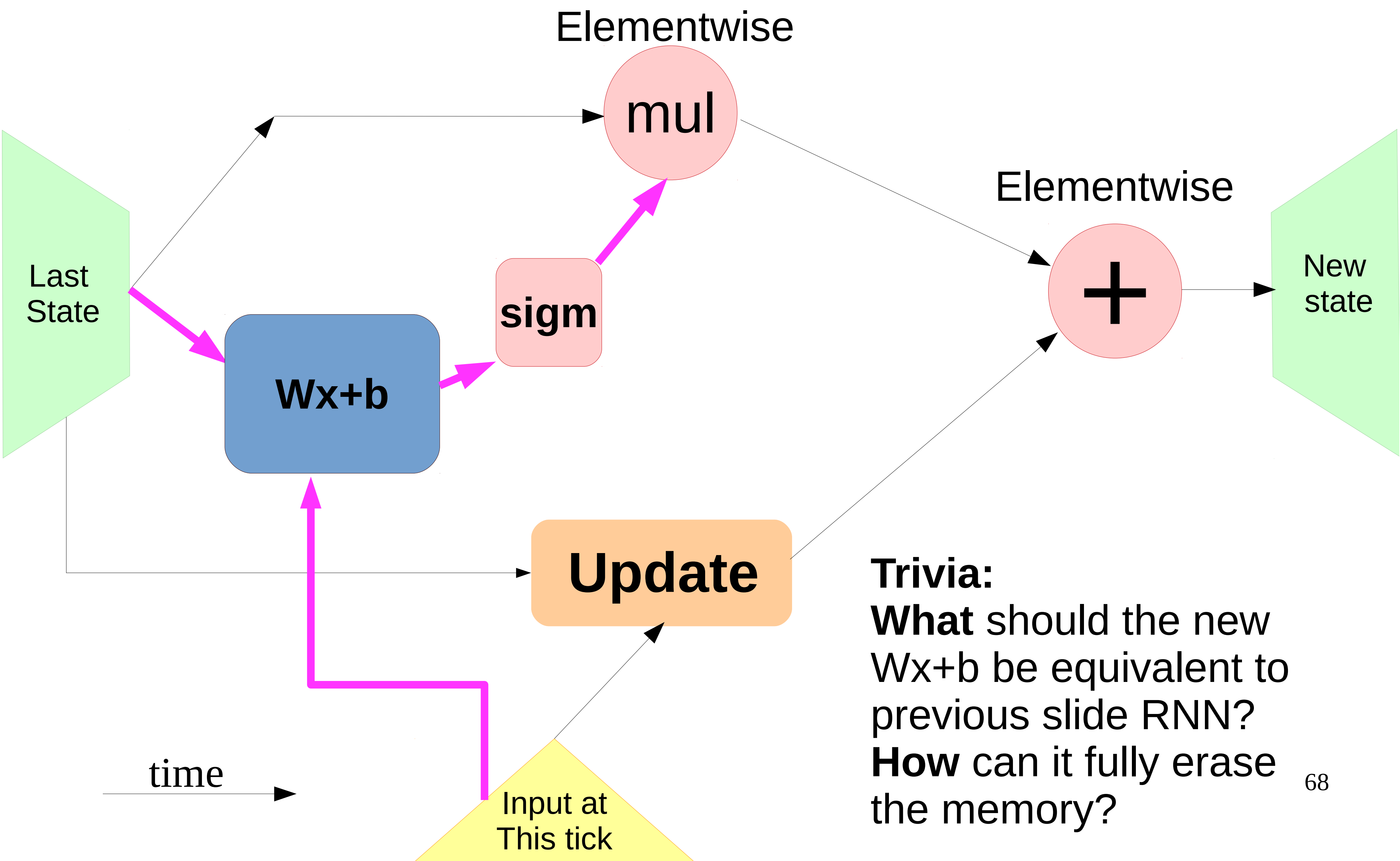
Residual RNN step



Residual RNN step

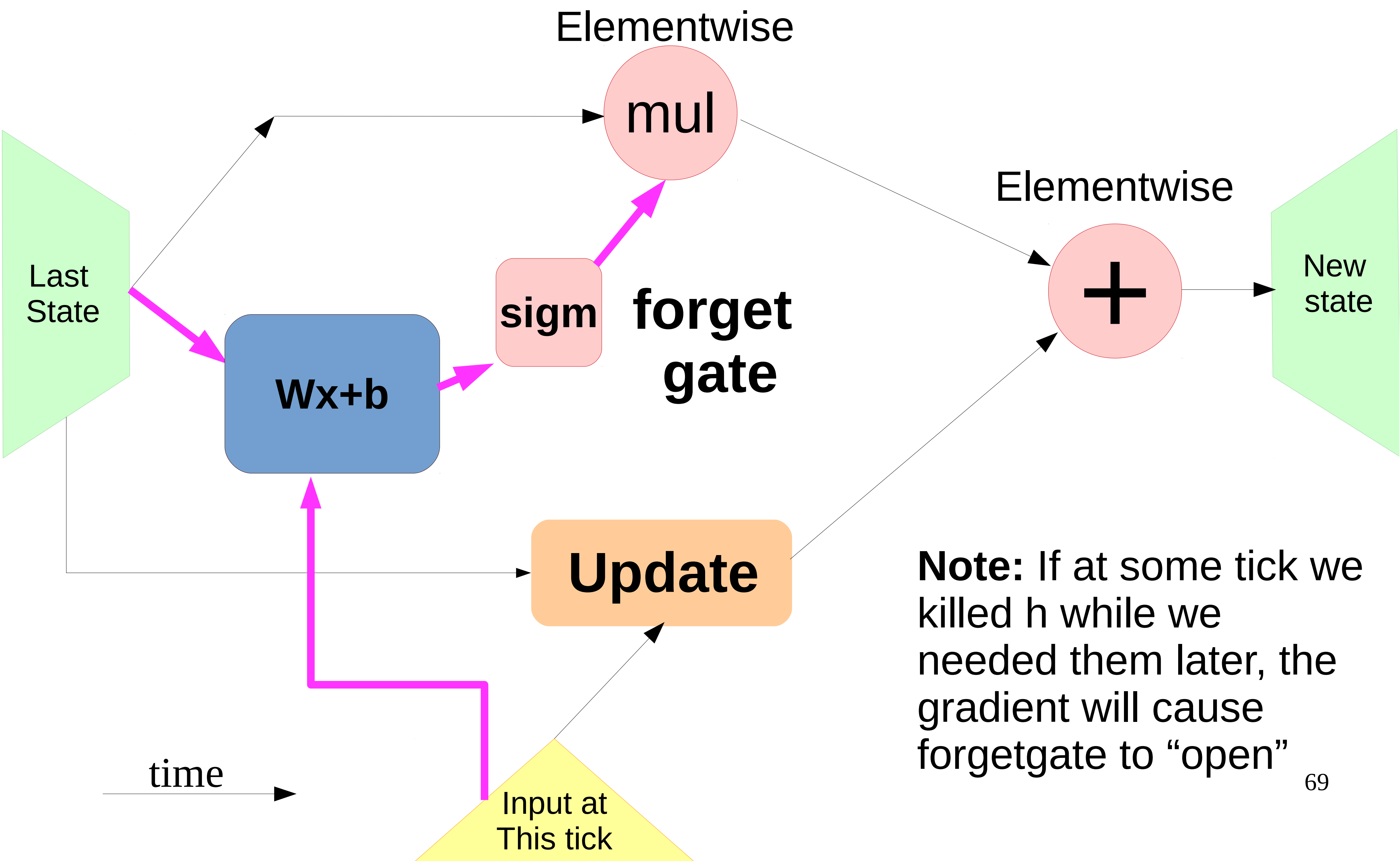


Residual RNN step



Trivia:
What should the new Wx+b be equivalent to previous slide RNN?
How can it fully erase the memory?

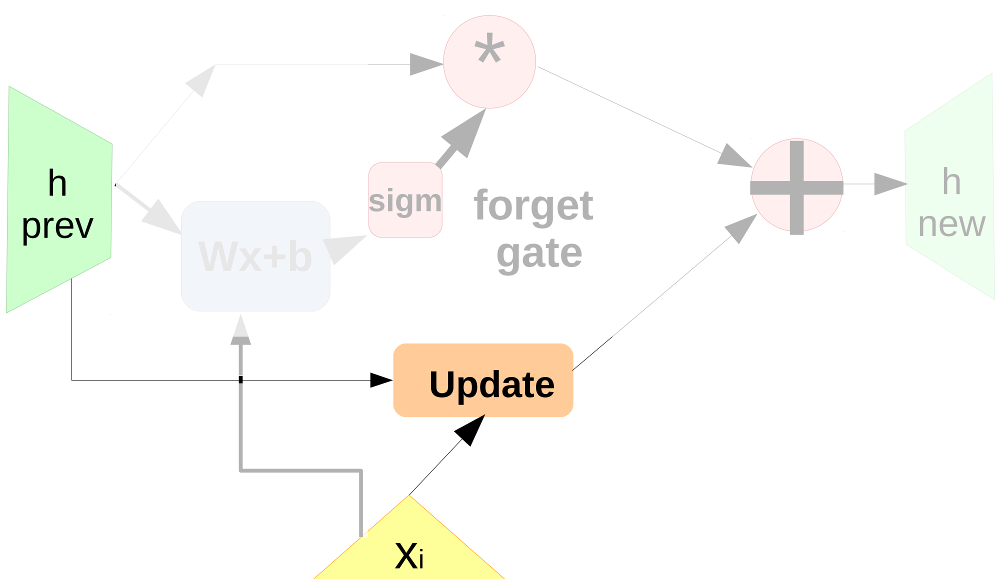
Residual RNN step



Note: If at some tick we killed h while we needed them later, the gradient will cause forgetgate to "open"

What we drew

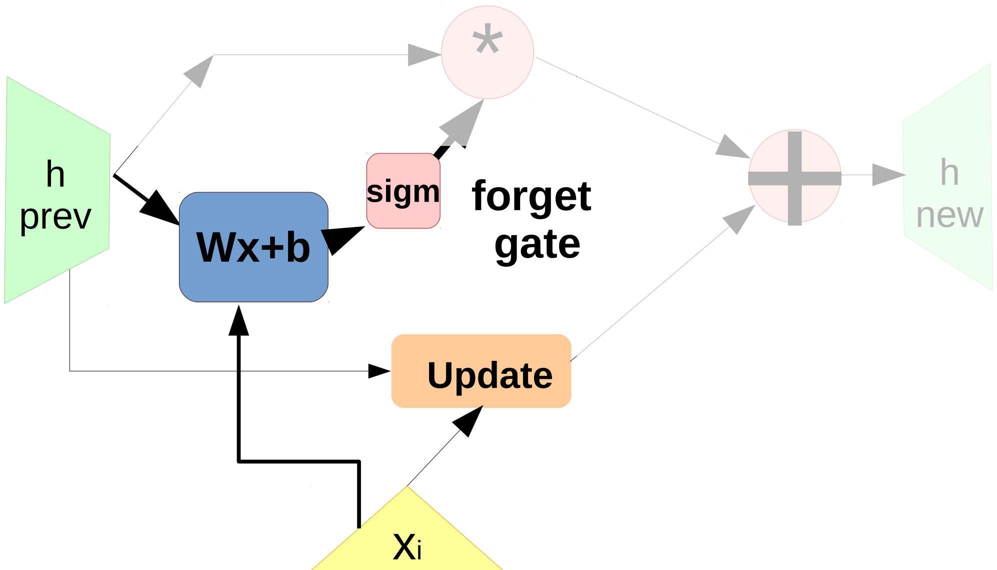
$$update\left(x_i, h_{i-1}\right) = \tanh\left(W_{hid}^{update} \cdot h_{i-1} + W_{inp}^{update} \cdot x_i + b^{update}\right)$$



What we drew

$$update(x_i, h_{i-1}) = \tanh(W_{hid}^{update} \cdot h_{i-1} + W_{inp}^{update} \cdot x_i + b^{update})$$

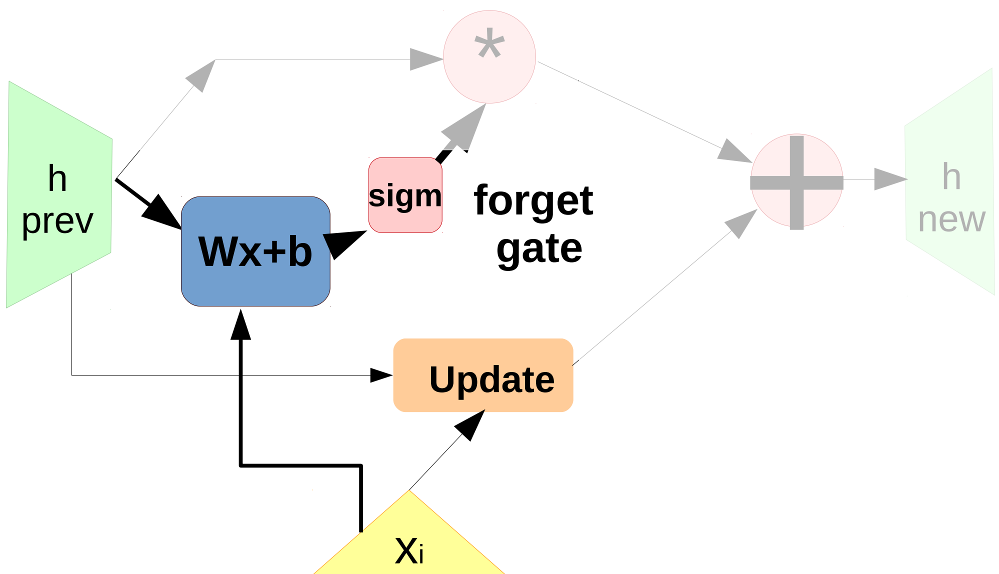
$$forget(x_i, h_{i-1}) = \sigma(W_{hid}^{forget} \cdot h_{i-1} + W_{inp}^{forget} \cdot x_i + b^{forget})$$



What we drew

$$update(x_i, h_{i-1}) = \tanh(W_{hid}^{update} \cdot h_{i-1} + W_{inp}^{update} \cdot x_i + b^{update})$$

$$forget(x_i, h_{i-1}) = \sigma(W_{hid}^{forget} \cdot h_{i-1} + W_{inp}^{forget} \cdot x_i + b^{forget})$$



How to compute h_new?

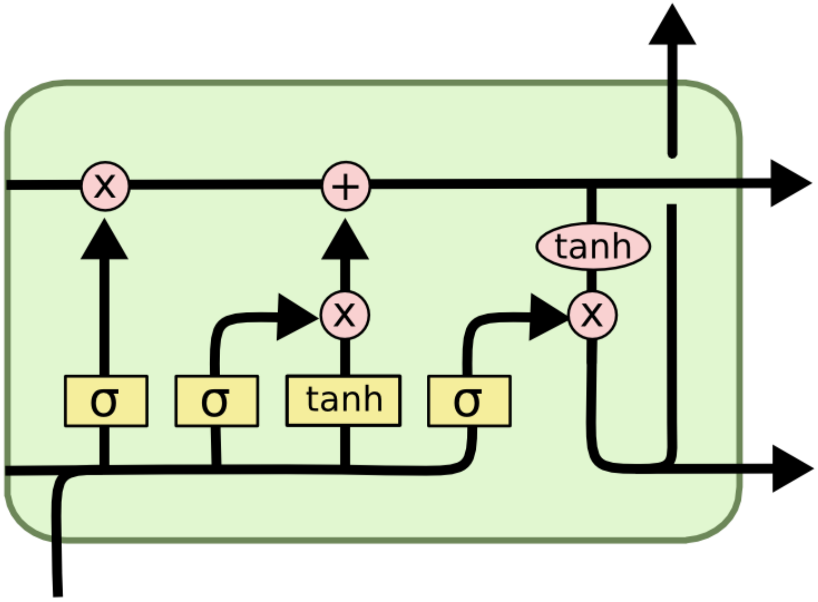
What we drew

$$update(x_i, h_{i-1}) = \tanh(W_{hid}^{update} \cdot h_{i-1} + W_{inp}^{update} \cdot x_i + b^{update})$$

$$forget(x_i, h_{i-1}) = \sigma(W_{hid}^{forget} \cdot h_{i-1} + W_{inp}^{forget} \cdot x_i + b^{forget})$$

$$h_i(x_i, h_{i-1}) = forget(x_i, h_{i-1}) \cdot h_{i-1} + update(x_i, h_{i-1})$$

LSTM



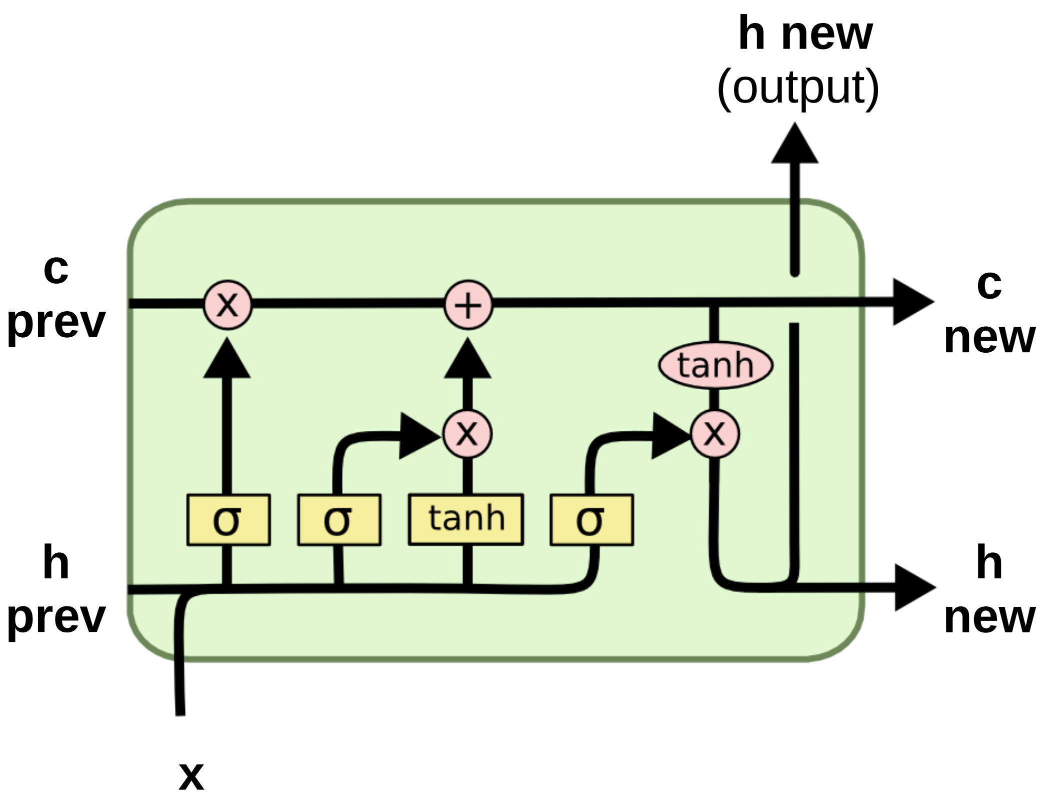
2 hidden states:

- Cell (“private” state)
- Output (“public” state)

4 blocks:

- Update
- Forget gate
- Input gate
- Output gate

LSTM



$$i_t = Sigm(\theta_{xi} x_t + \theta_{hi} h_{t-1} + b_i)$$

$$f_t = Sigm(\theta_{xf} x_t + \theta_{hf} h_{t-1} + b_f)$$

$$o_t = Sigm(\theta_{xo} x_t + \theta_{ho} h_{t-1} + b_o)$$

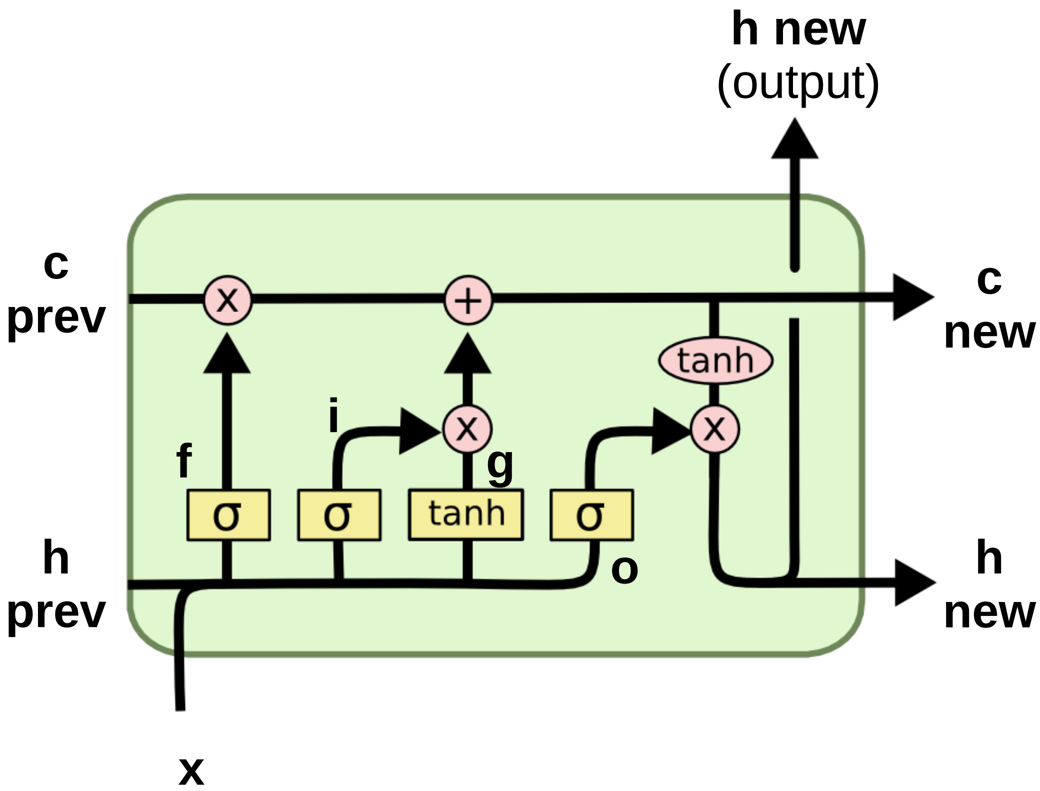
$$g_t = Tanh(\theta_{xg} x_t + \theta_{hg} h_{t-1} + b_g)$$

$$c_t = f_t \otimes c_{t-1} + i_t \otimes g_t$$

$$h_t = o_t \otimes Tanh(c_t)$$

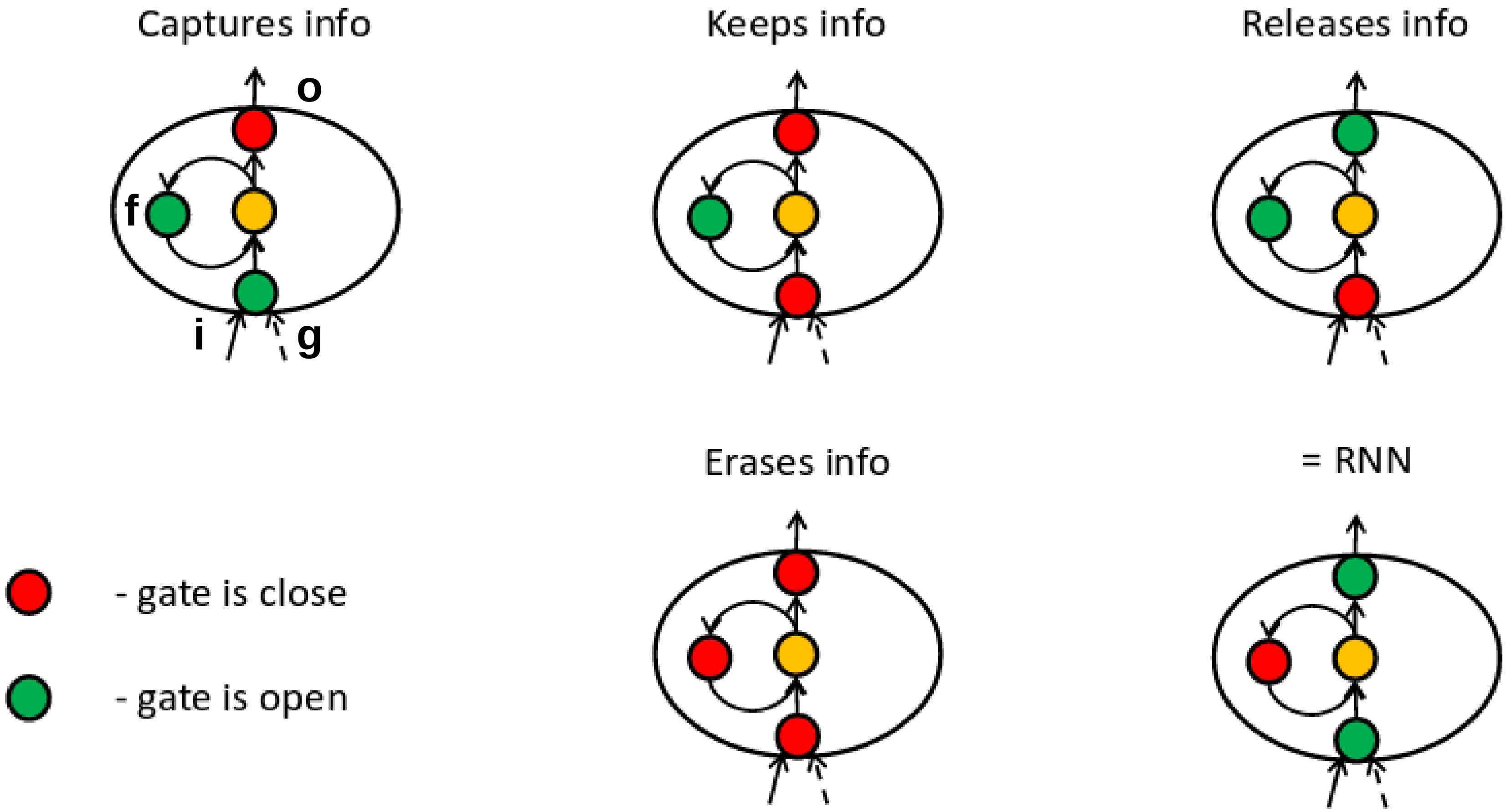
Where are the gates?

LSTM



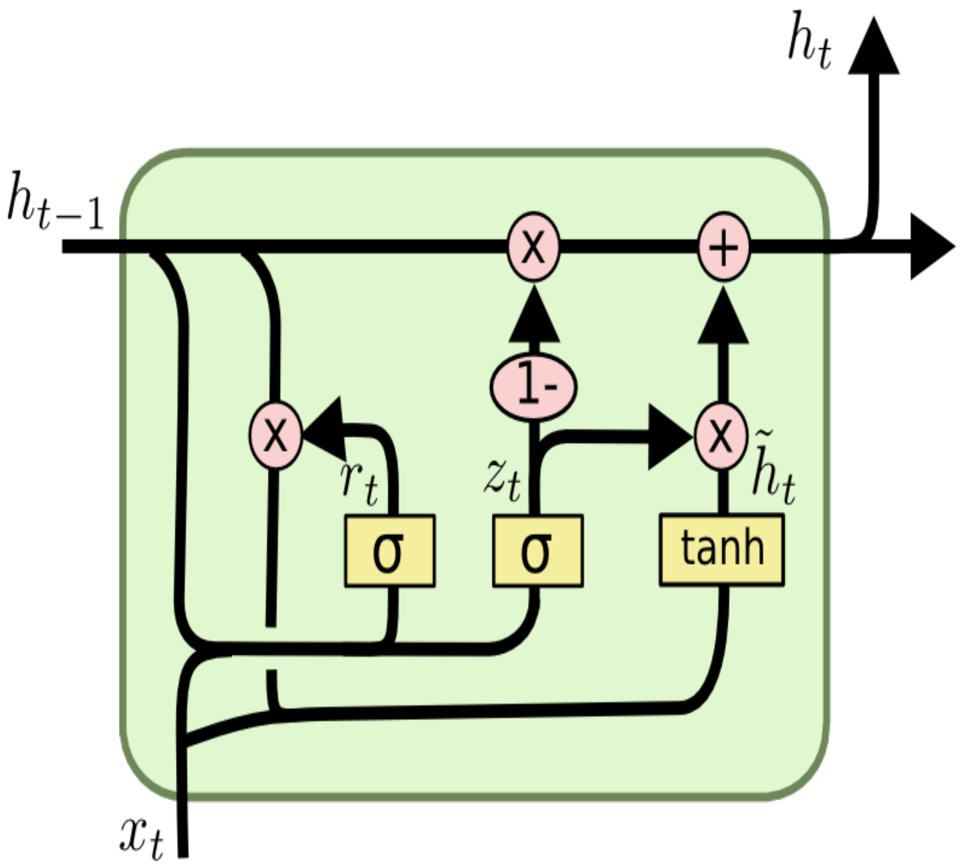
$$i_t = Sigm(\theta_{xi} x_t + \theta_{hi} h_{t-1} + b_i)$$
$$f_t = Sigm(\theta_{xf} x_t + \theta_{hf} h_{t-1} + b_f)$$
$$o_t = Sigm(\theta_{xo} x_t + \theta_{ho} h_{t-1} + b_o)$$
$$g_t = Tanh(\theta_{xg} x_t + \theta_{hg} h_{t-1} + b_g)$$
$$c_t = f_t \otimes c_{t-1} + i_t \otimes g_t$$
$$h_t = o_t \otimes Tanh(c_t)$$

LSTM: not a monster



[Pictures: E Lobacheva, D Vetrov]

GRU



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

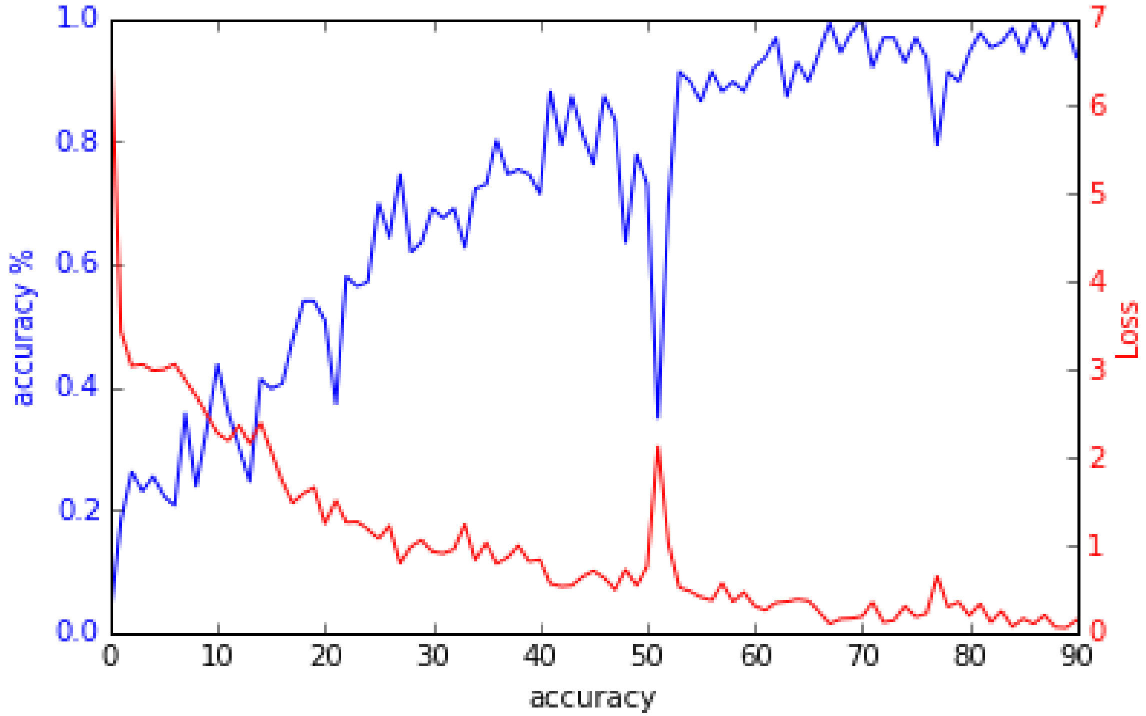
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Okay, the gradients no longer vanish
except they still do, if only slower

But how do we deal with exploding grads?



Ideas?

Gradient clipping

At each time tick,

- check if grad abs value is more than … 5?
- If so, clip it
  - large positive is now 5,
  - large negative is now -5

- How large is too large?
  - Reduce clipping threshold until explosions disappear

Gradient clipping

Where do I clip?

- Clip each element of $\delta L / \delta w$
- Clip each element of $\delta h_{i+1} / \delta h_i$
- Clip whole $\delta L / \delta w$ by norm
  - If $\left\| \frac{\delta L}{\delta w} \right\| > 5$, scale $\frac{\delta L}{\delta w} / \left\| \frac{\delta L}{\delta w} \right\| \cdot 5$

Generating stuff

Easy:

- Names, small phrases
- Arxiv article titles
- Orthographically correct delirium

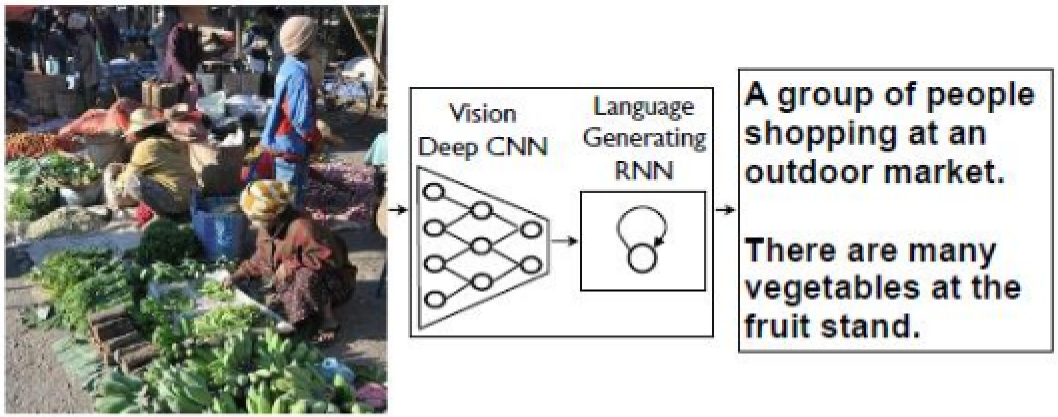
Medium:

- Music (notes)
- Organic molecules (SMILES)

Hard:

- C/C++ source code
- Articles (LaTeX full text)
- Your course projects
- Seq2Seq

Homework 4: Image Captioning



- Demo - http://stanford.io/2esMxOq
- Upload your image - http://bit.ly/2eAoueP

To be continued…

Lecture 10: embeddings, text convolutions

Lecture 11: seq2seq architectures, attention

Nuff

Coding time!