# Deep Learning

**Recurrent Neural Network**

# Examples of sequence data

| | | | |
|---|---|---|---|
| Speech recognition |  | → | "The quick brown fox jumped over the lazy dog." |
| Music generation | ∅ | → |  |
| Sentiment classification | "There is nothing to like in this movie." | → | ★☆☆☆☆ |
| DNA sequence analysis | AGCCCCTGTGAGGAACTAG | → | AGCCCCTGTGAGGAACTAG |
| Machine translation | Voulez-vous chanter avec moi? | → | Do you want to sing with me? |
| Video activity recognition |  | → | Running |
| Name entity recognition | Yesterday, Harry Potter met Hermione Granger. | → | Yesterday, Harry Potter met Hermione Granger. |

# Motivating example

x: Harry Potter and Hermione Granger invented a new spell.

$x^{<1>}$ $x^{<2>}$ $x^{<3>}$ $\cdots$ $x^{<9>}$

$T_x=9$

y: 1 1 0 1 1 0 0 0 0

$y^{<1>}$ $y^{<2>}$ $y^{<3>}$ $y^{<4}$ ……. $y^{<9>}$

$T_y=9$

$x^{(i)<t>}$ $T_X^{(i)}$

$y^{(i)<t>}$ $T_y^{(i)}$

# Notation

## Representing words

x: Harry Potter and Hermione Granger invented a new spell.

$x^{<1>}$ $x^{<2>}$ $x^{<3>}$ ... $x^{<9>}$

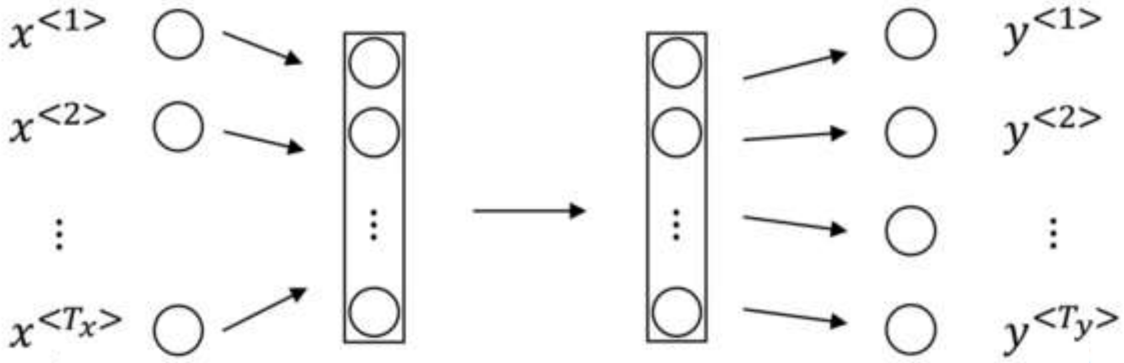
Vocabulary

$$\begin{bmatrix} word1 \\ \text{and} \\ . \\ . \\ . \\ \text{harry} \\ . \\ \text{potter} \\ . \\ wordn \end{bmatrix} \begin{array}{c} 1 \\ 2 \\ . \\ . \\ . \\ 4075 \\ . \\ 6015 \\ . \\ n \end{array} \qquad \begin{bmatrix} 0 \\ 0 \\ . \\ . \\ . \\ 1 \\ . \\ . \\ . \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 0 \\ . \\ . \\ . \\ . \\ . \\ 1 \\ . \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 1 \\ . \\ . \\ . \\ . \\ . \\ . \\ . \\ 0 \end{bmatrix} \qquad \begin{bmatrix} 0 \\ 0 \\ . \\ . \\ . \\ 1 \\ . \\ . \\ . \\ 0 \end{bmatrix}$$

# RNN

EACH x<i> IS 10000 dim vector because on one hot vector encoding

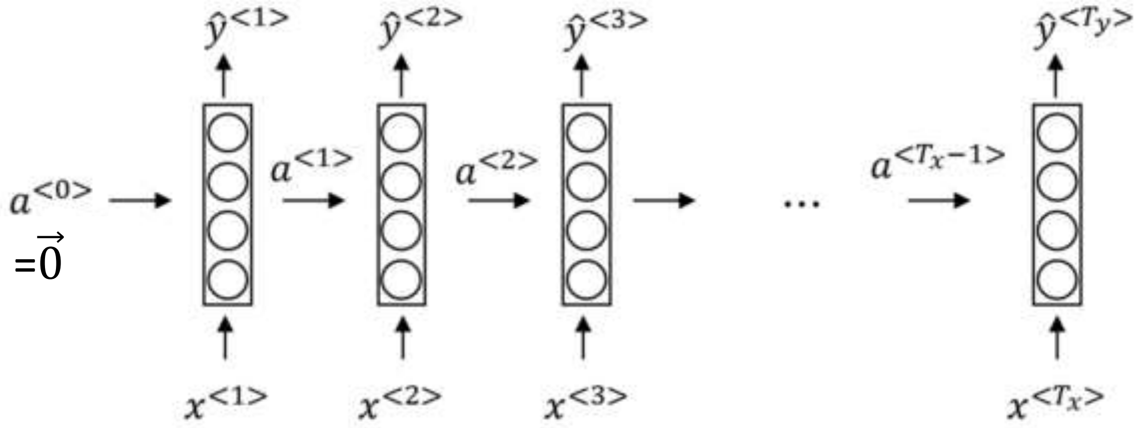
## Why not a standard network?




Problems:

- Inputs, outputs can be different lengths in different examples.
- Doesn't share features learned across different positions of text.

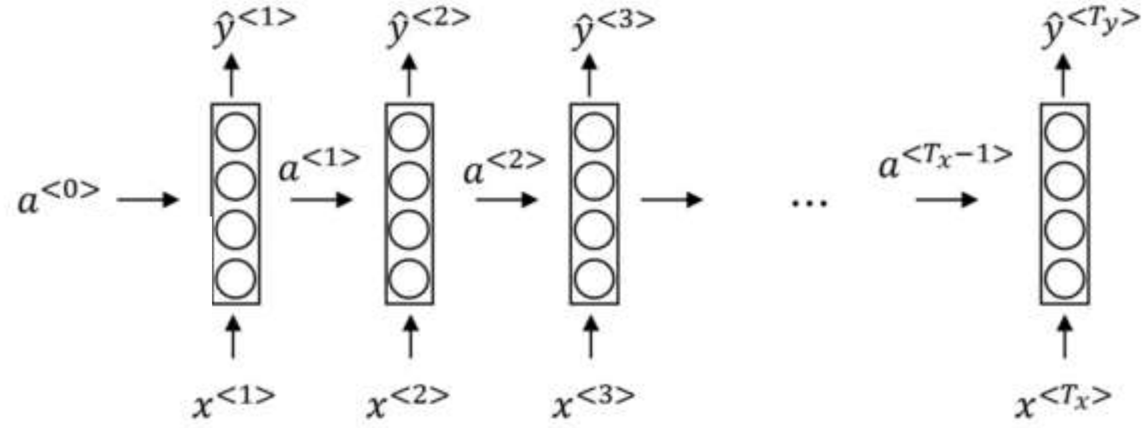
# Forward Propagation




Bidirectional RNN

He said, “Teddy Roosevelt was a great President.”
He said, “Teddy bears are on sale!”

# Forward Propagation

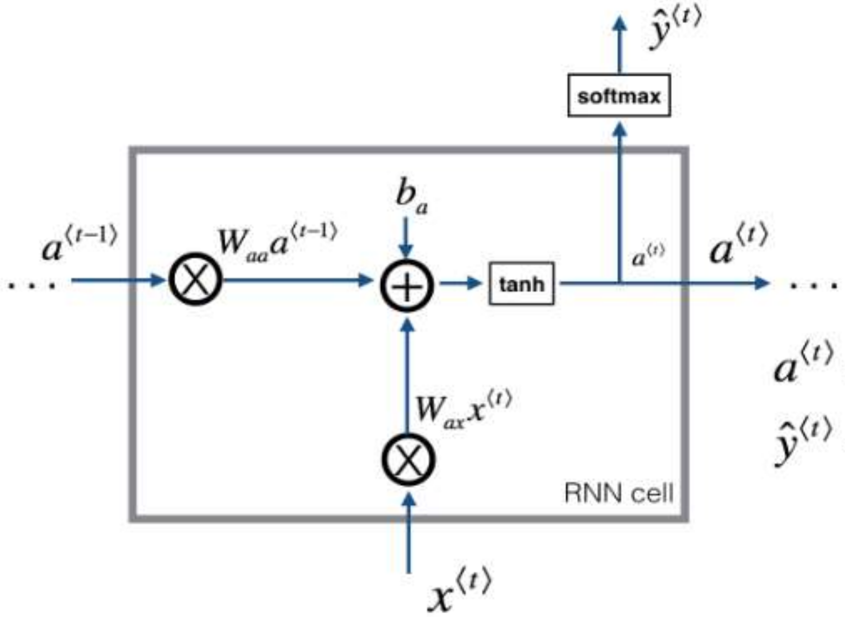



# Simplified RNN notation

$$a^{<t>} = g(W_{aa}a^{<t>} + W_{ax}x^{<t>} + b_a)$$

$$\hat{y}^{<t>} = g(W_{ya}a^{<t>} + b_y)$$

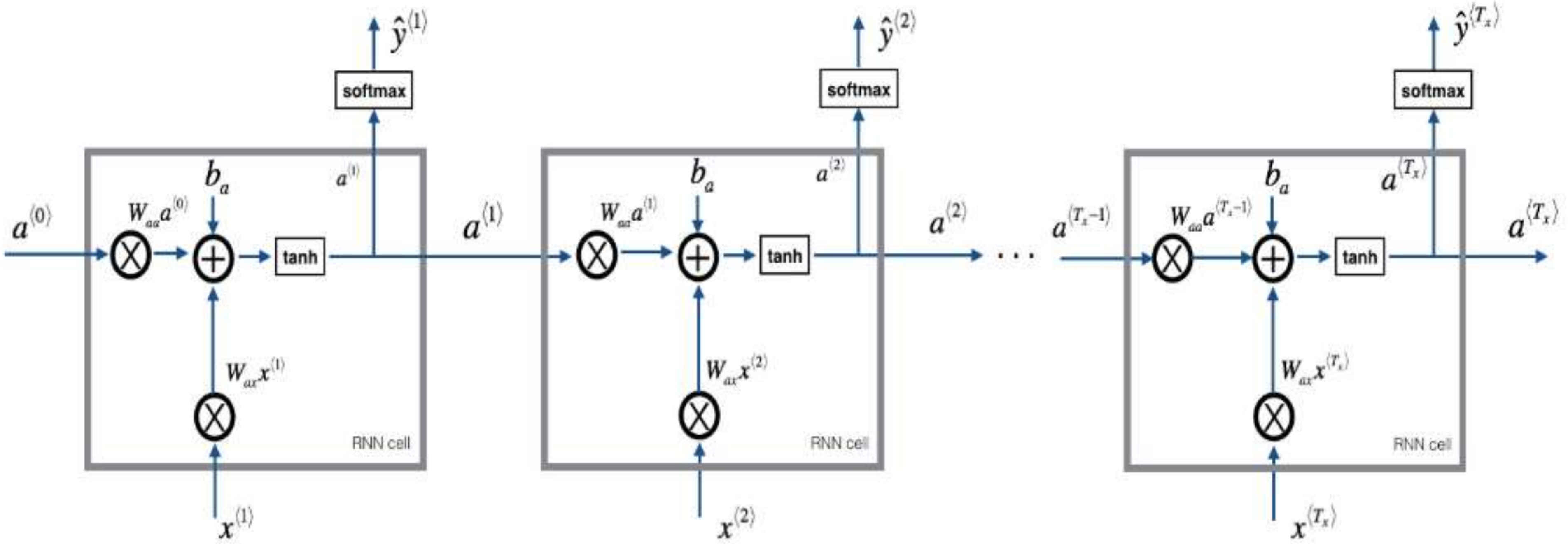
# RNN Cell




$$a^{\langle t \rangle} = \tanh(W_{ax}x^{\langle t \rangle} + W_{aa}a^{\langle t-1 \rangle} + b_a)$$

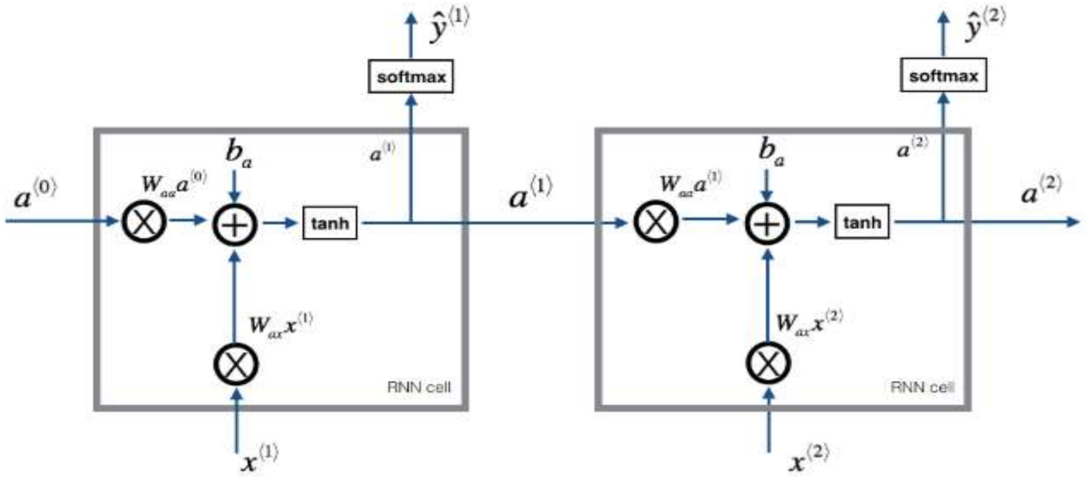
$$\hat{y}^{\langle t \rangle} = soft\max(W_{ya}a^{\langle t \rangle} + b_y)$$

# RNN Forward pass




Basic RNN. The input sequence $x = (x^{\langle 1 \rangle}, x^{\langle 2 \rangle}, \ldots, x^{\langle T_x \rangle})$ is carried over $T_x$ time steps. The network outputs $y = (y^{\langle 1 \rangle}, y^{\langle 2 \rangle}, \ldots, y^{\langle T_x \rangle})$

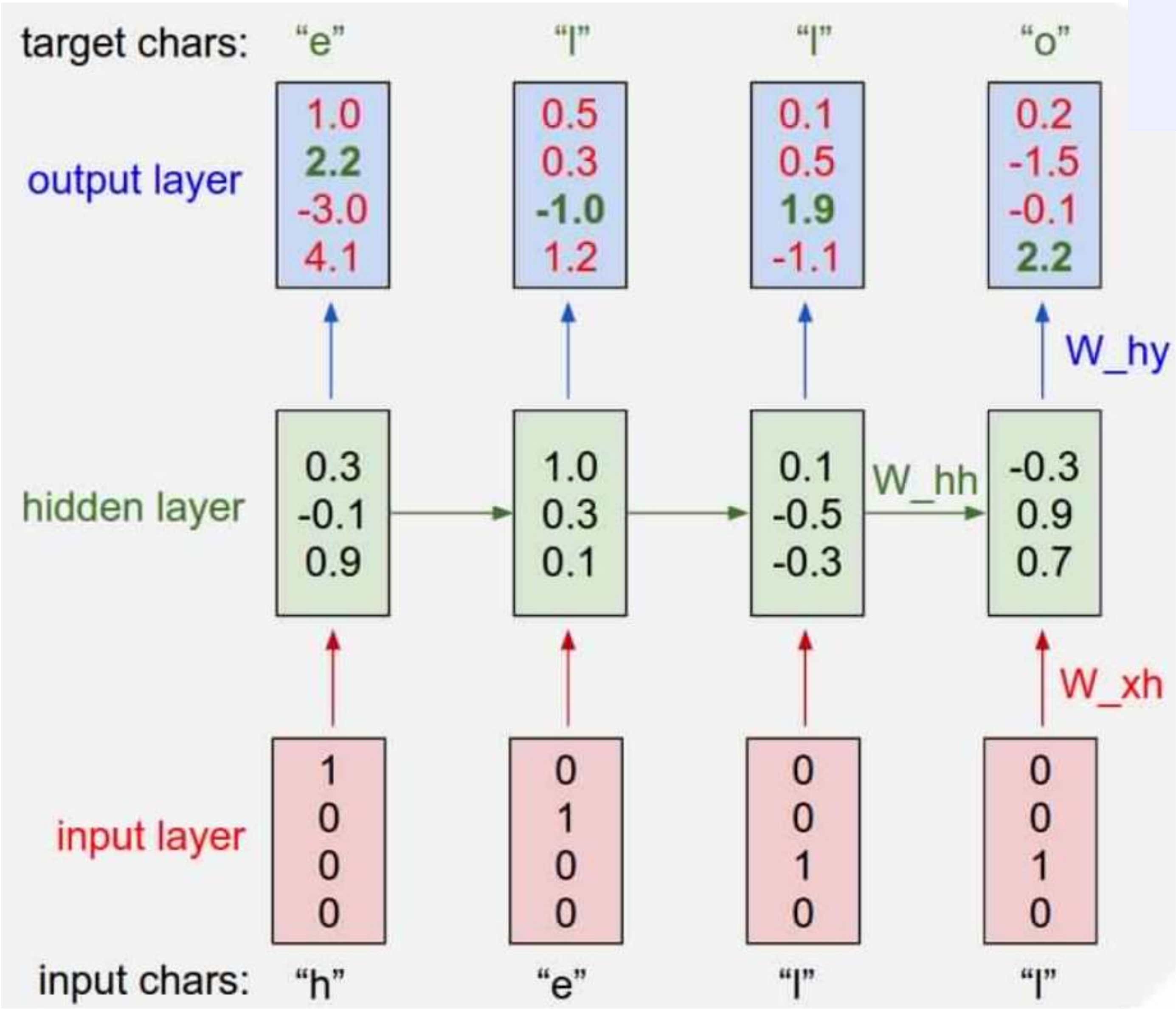
# Example of RNN API




```
rnn = RNN()
y = rnn.step(x)
```

```
class RNN:
  # ...
  def step(self, x):
    # update the hidden state
    self.h = np.tanh(np.dot(self.W_hh, self.h) + np.dot(self.W_xh, x))
    # compute the output vector
    y = np.dot(self.W_hy, self.h)
    return y
```

# Example of RNN API




```python
class RNN:
  # ...
  def step(self, x):
    # update the hidden state
    self.h = np.tanh(np.dot(self.W_hh, self.h) + np.dot(self.W_xh, x))
    # compute the output vector
    y = np.dot(self.W_hy, self.h)
    return y
```

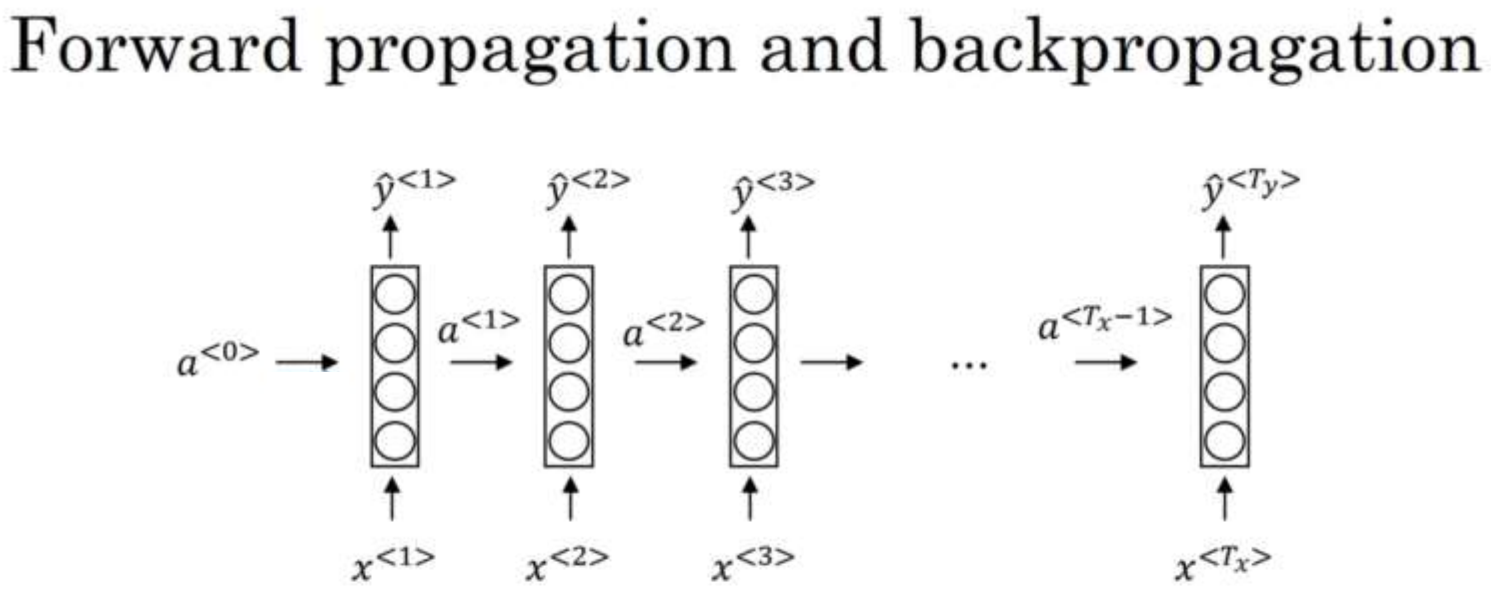
# Simplified RNN notation

$$a^{<t>} = g(W_{aa}a^{<t>} + W_{ax}x^{<t>} + b_a)$$

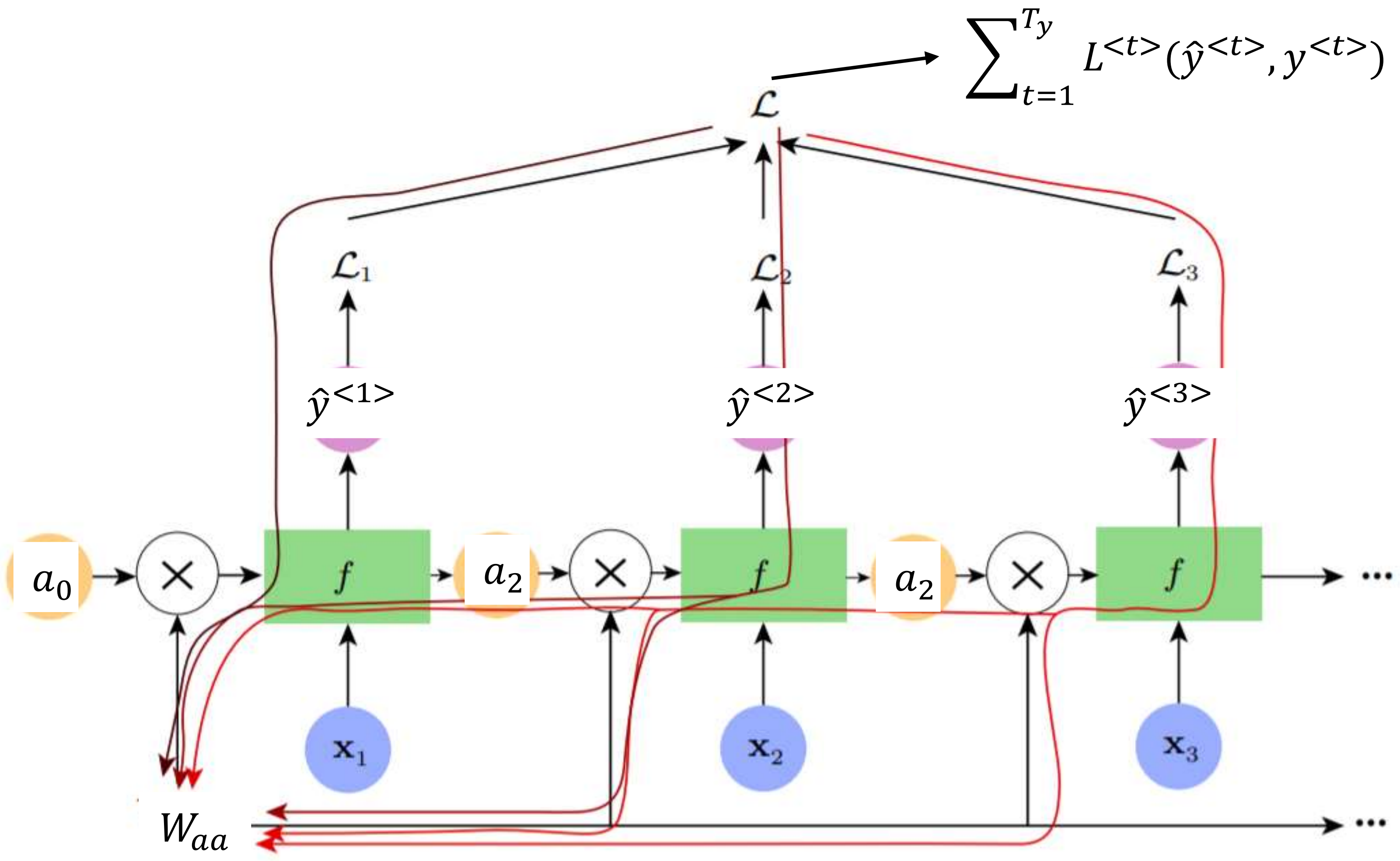
$$\hat{y}^{<t>} = g(W_{ya}a^{<t>} + b_y)$$

$$g(W_a[a^{<t-1>}, x^{<t>}] + b_a)$$
$$W_a = [W_{aa} \mid W_{ax}]$$

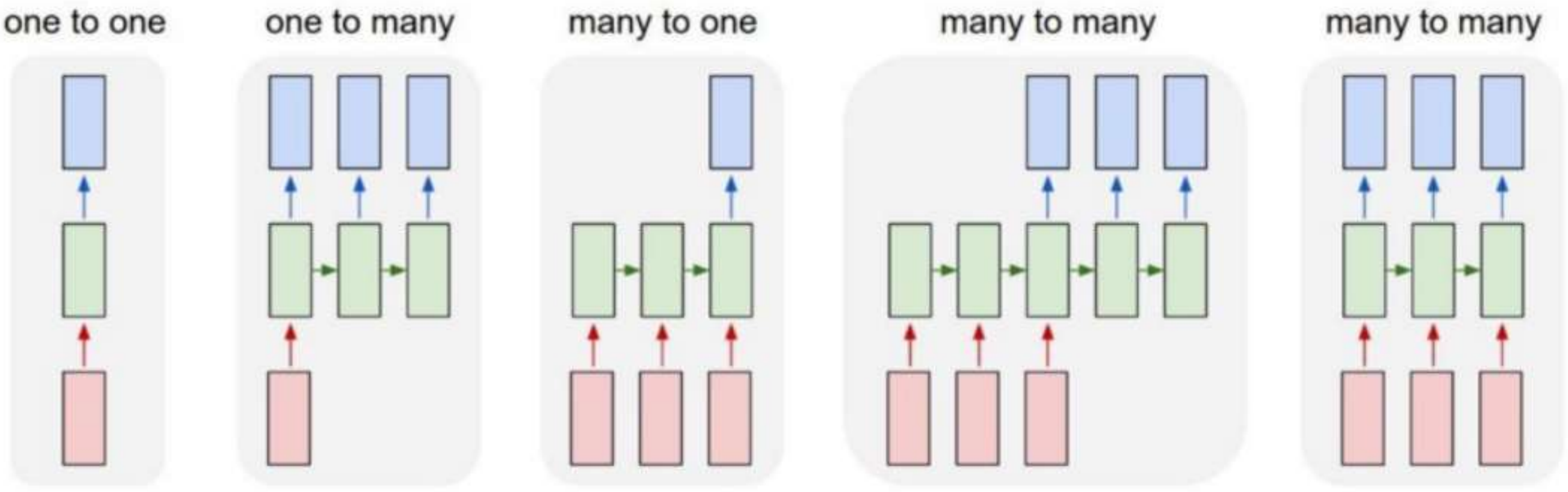
# Backpropagation through time



# Backpropagation through time



# Different types of RNN



# Language model and sequence generation

## What is language modelling?

Speech recognition

The apple and pair salad.

The apple and pear salad.

$P(\text{The apple and pair salad}) =$ 3.2 X $10^{-13}$

$P(\text{The apple and pear salad}) =$ 5.7 X $10^{-10}$

P(Sentence) = $P(\hat{y}^{<1>}, \hat{y}^{<2>}, \ldots, \hat{y}^{<t>})$

# Language modelling with an RNN

Training set: large corpus of english text.

Tokenization

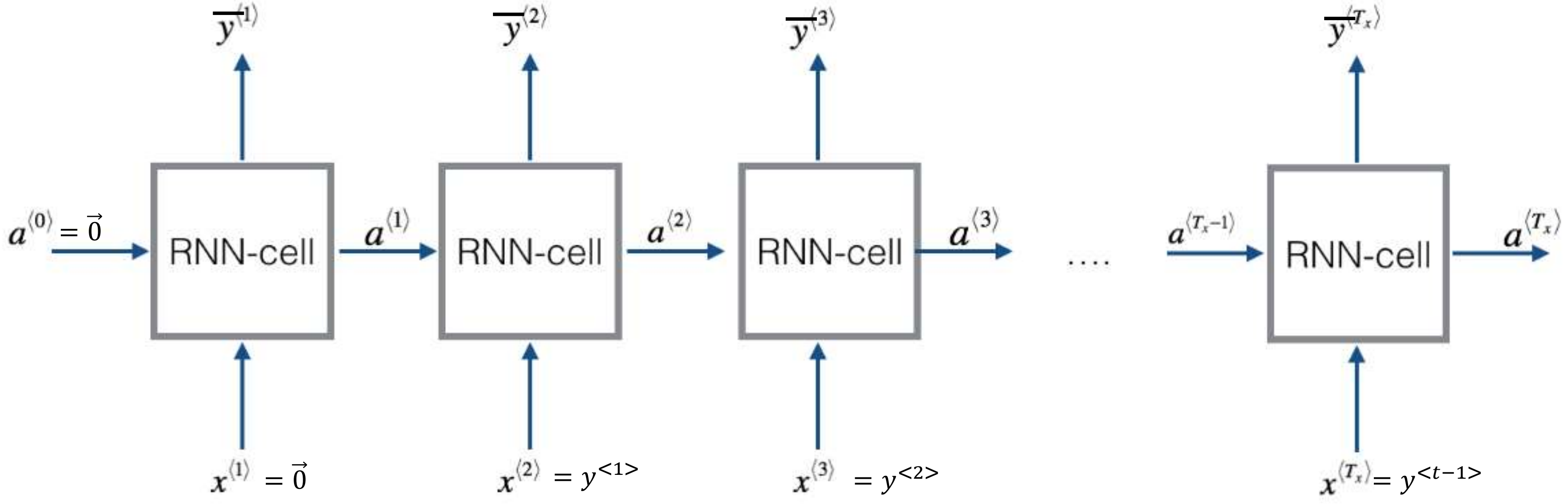
Cats average 15 hours of sleep a day. <EOS>

$y^{<1>}$ $y^{<2>}$ $y^{<3>}$ $y^{<4>}$ .... $y^{<8>}$ $y^{<9>}$

$x^{<t>} = y^{<t-1>}$

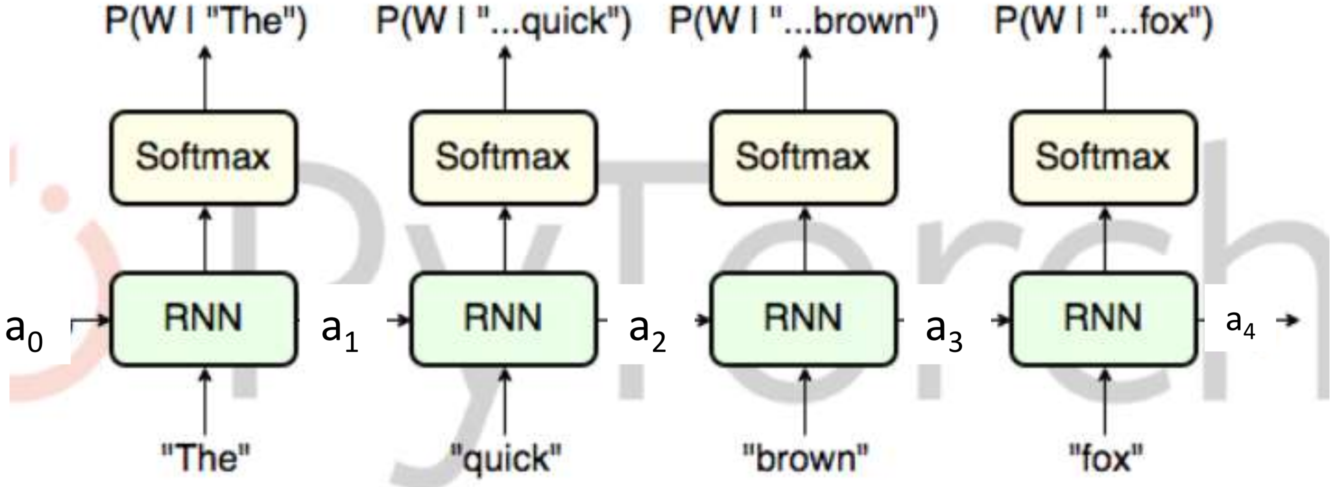
The Egyptian Mau is a bread of cat. <EOS>

# RNN Language Model



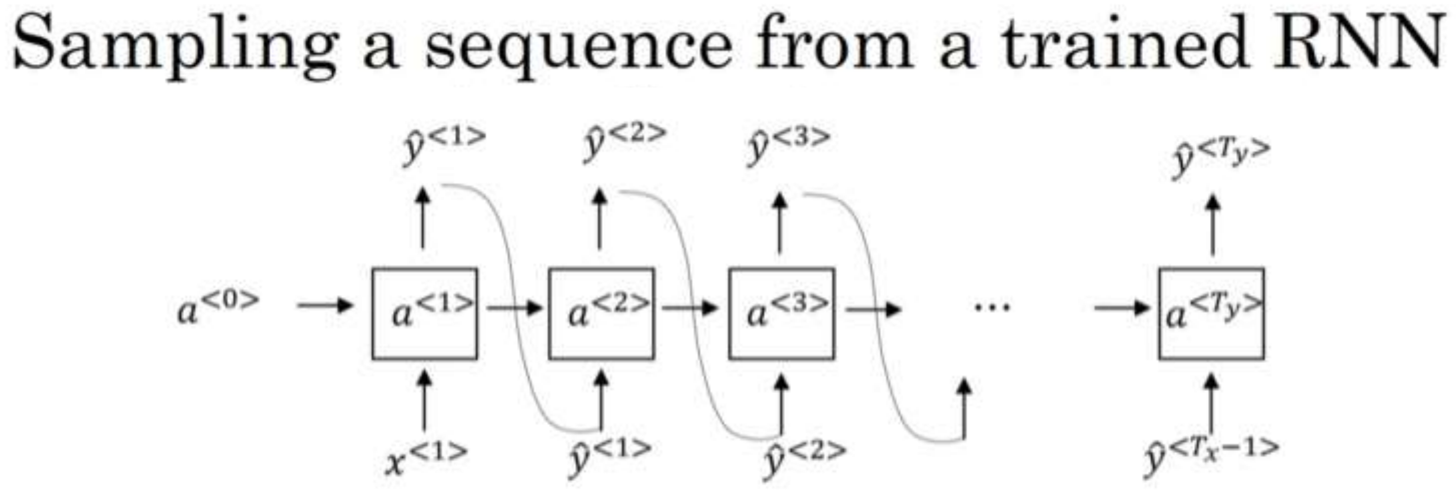

$$\mathcal{L}(\overline{y}^{<t>}, y^{<t>}) = -\sum_i y_i^{<t>} \log \overline{y}_i^{<t>}$$

$$\mathcal{L} = \sum_t \mathcal{L}^{<t>}(\overline{y}^{<t>}, y^{<t>})$$

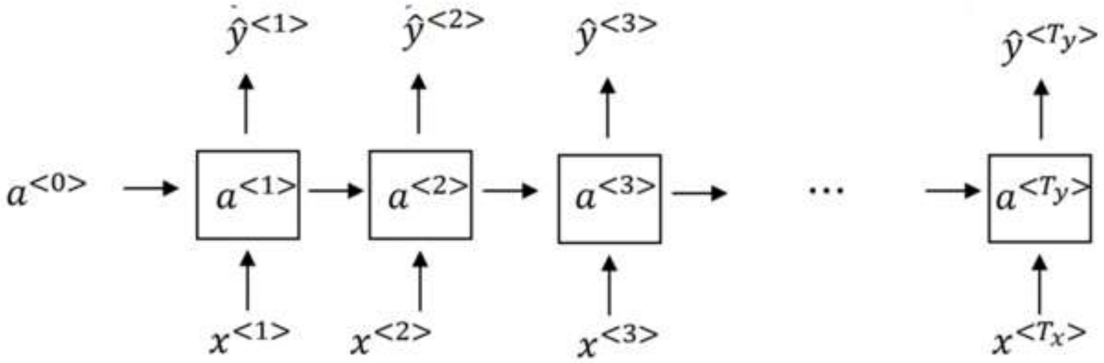


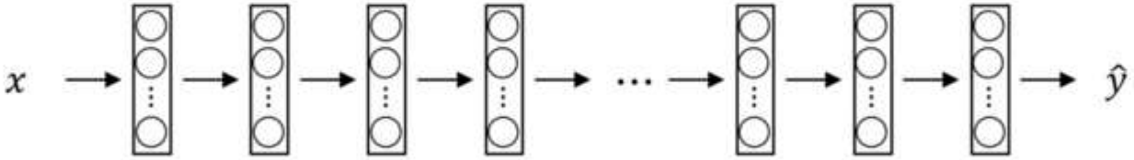

$$P(A,B,C,D) = P(A)P(B|A)P(C|A,B)P(D|A,B,C)$$

# Sampling novel sequences



# Vanishing gradients with RNNs

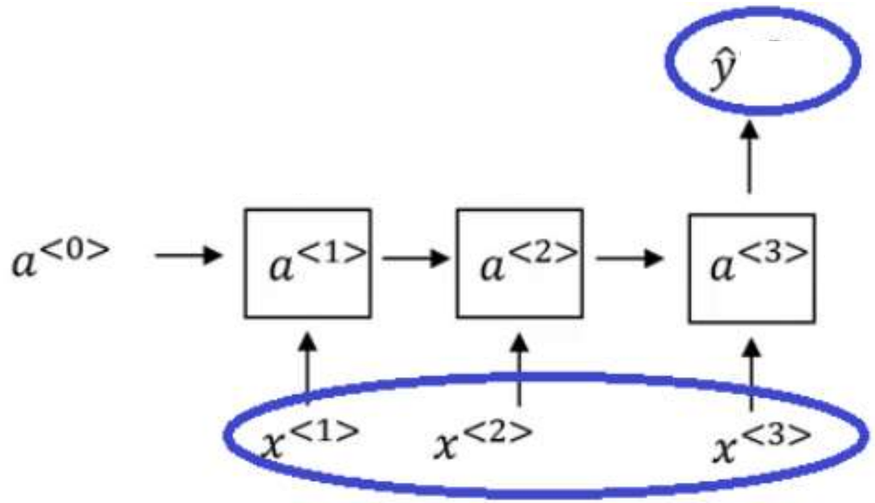


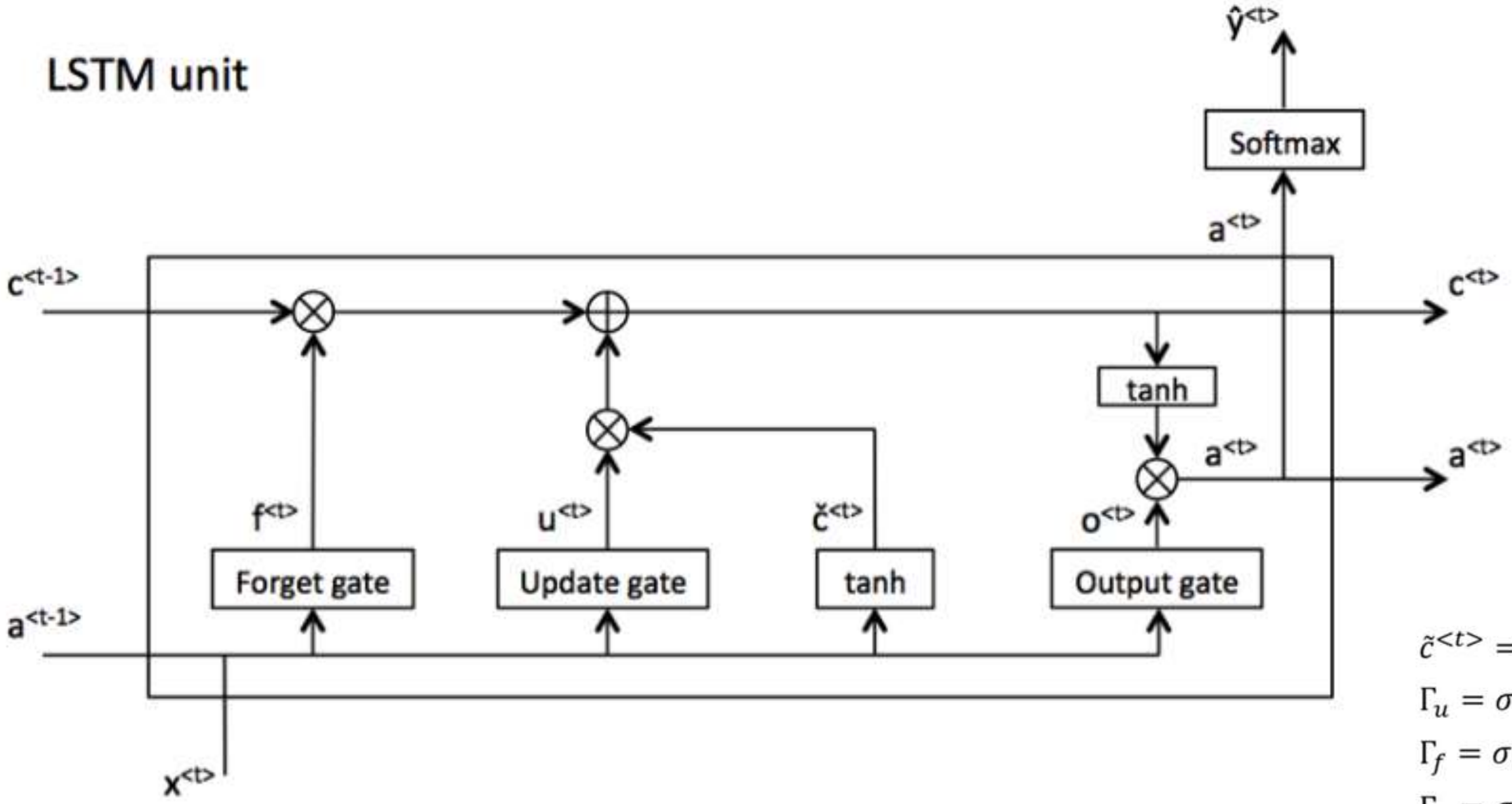

Example 1:The ***cat***, which already ate, ...................................***was*** full.

Example 2:The ***cats***, which already ate, ...................................***were*** full.

# Local Influence



LSTM unit




$$\tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c)$$
$$\Gamma_u = \sigma(W_u[\, a^{<t-1>}, x^{<t>}] + b_u)$$
$$\Gamma_f = \sigma(W_f[\, a^{<t-1>}, x^{<t>}] + b_f)$$
$$\Gamma_o = \sigma(W_o[\, a^{<t-1>}, x^{<t>}] + b_o)$$
$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>}$$
$$a^{<t>} = \Gamma_o * \tanh\, c^{<t>}$$

# LSTM in pictures

$$\tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c)$$

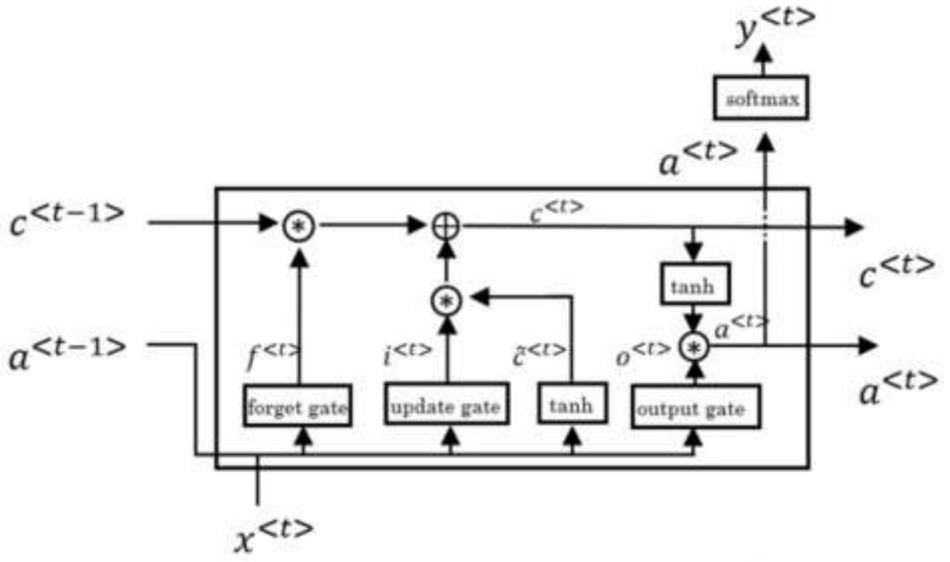
$$\Gamma_u = \sigma(W_u[\, a^{<t-1>}, x^{<t>}] + b_u)$$

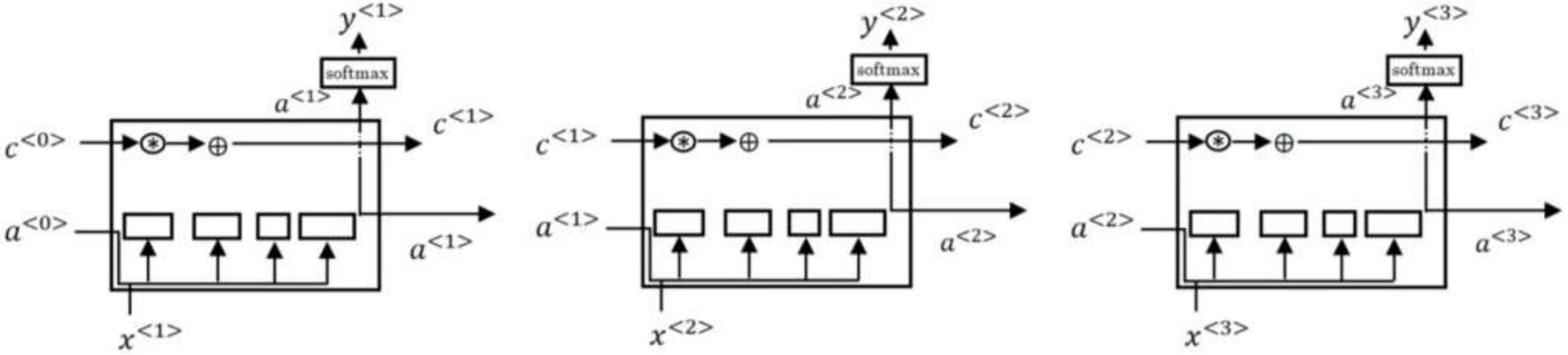
$$\Gamma_f = \sigma(W_f[\, a^{<t-1>}, x^{<t>}] + b_f)$$

$$\Gamma_o = \sigma(W_o[\, a^{<t-1>}, x^{<t>}] + b_o)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>}$$

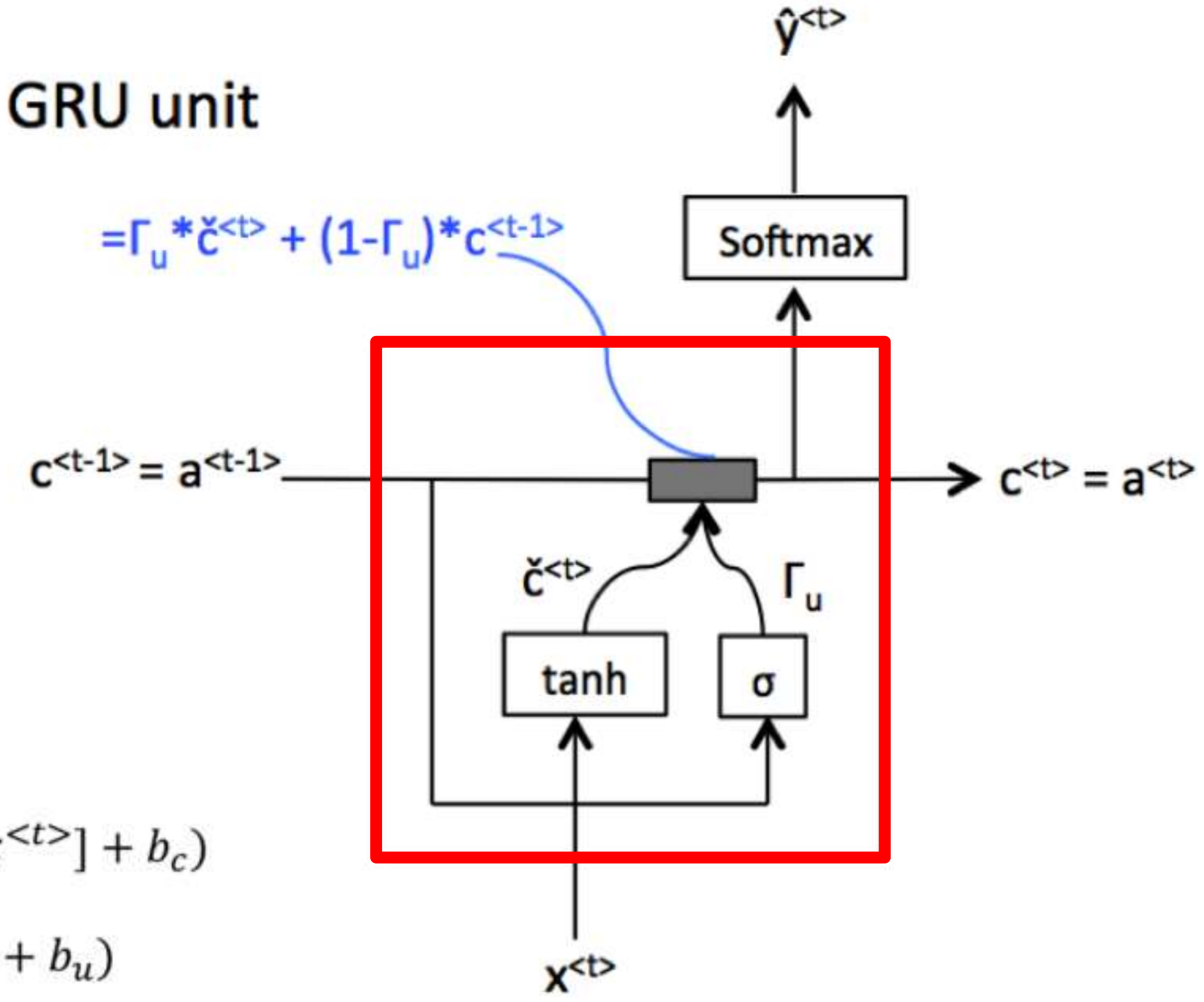
$$a^{<t>} = \Gamma_o * \tanh c^{<t>}$$



https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be

# GRU unit




$$\tilde{c}^{<t>} = \tanh(W_c[\, c^{<t-1>}, x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[\, c^{<t-1>}, x^{<t>}] + b_u)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * c^{<t-1>}$$


[Cho et al., 2014. On the properties of neural machine translation: Encoder-decoder approaches]
[Chung et al., 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling]

https://medium.com/swlh/tensorflow-vs-pytorch-for-text-classification-using-gru-e95f1b68fa2d

# Intuition of vanishing gradient in GRU

$$\frac{\partial L}{\partial c^{<t-1>}} = \frac{\partial L}{\partial c^{<t>}} \frac{\partial c^{<t>}}{\partial c^{<t-1>}}$$

and since:

$$\frac{\partial c^{<t>}}{\partial c^{<t-1>}} = 1 - \Gamma_u$$

we have:

$$\frac{\partial L}{\partial c^{<t-1>}} \approx \frac{\partial L}{\partial c^{<t>}} \text{ when } \Gamma_u \approx 0$$

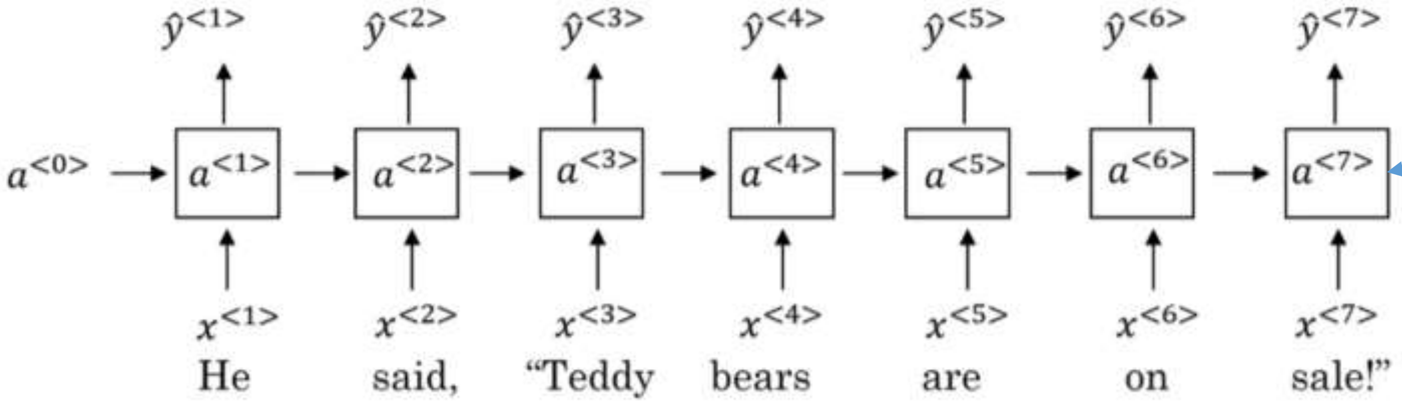
In the notation we are using for gradient descent, this means $dc^{<t-1>} \approx dc^{<t>}$

# Bidirectional RNN

## Getting information from the future

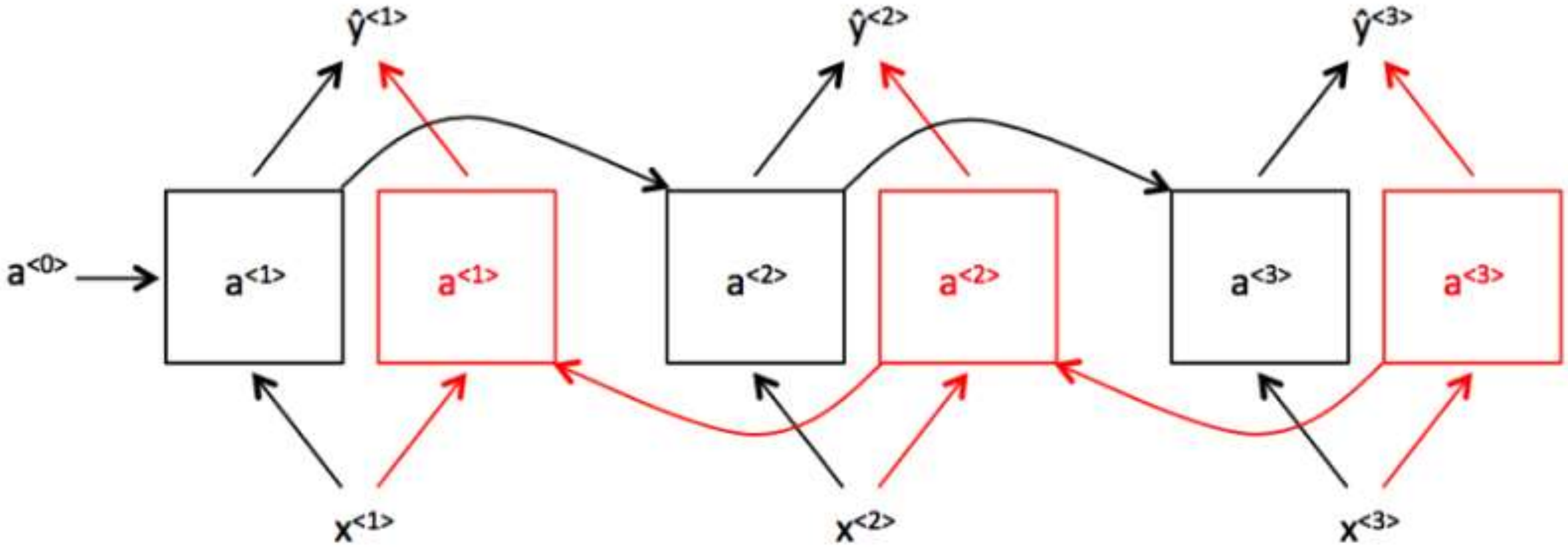
He said, “Teddy bears are on sale!”

He said, “Teddy Roosevelt was a great President!”



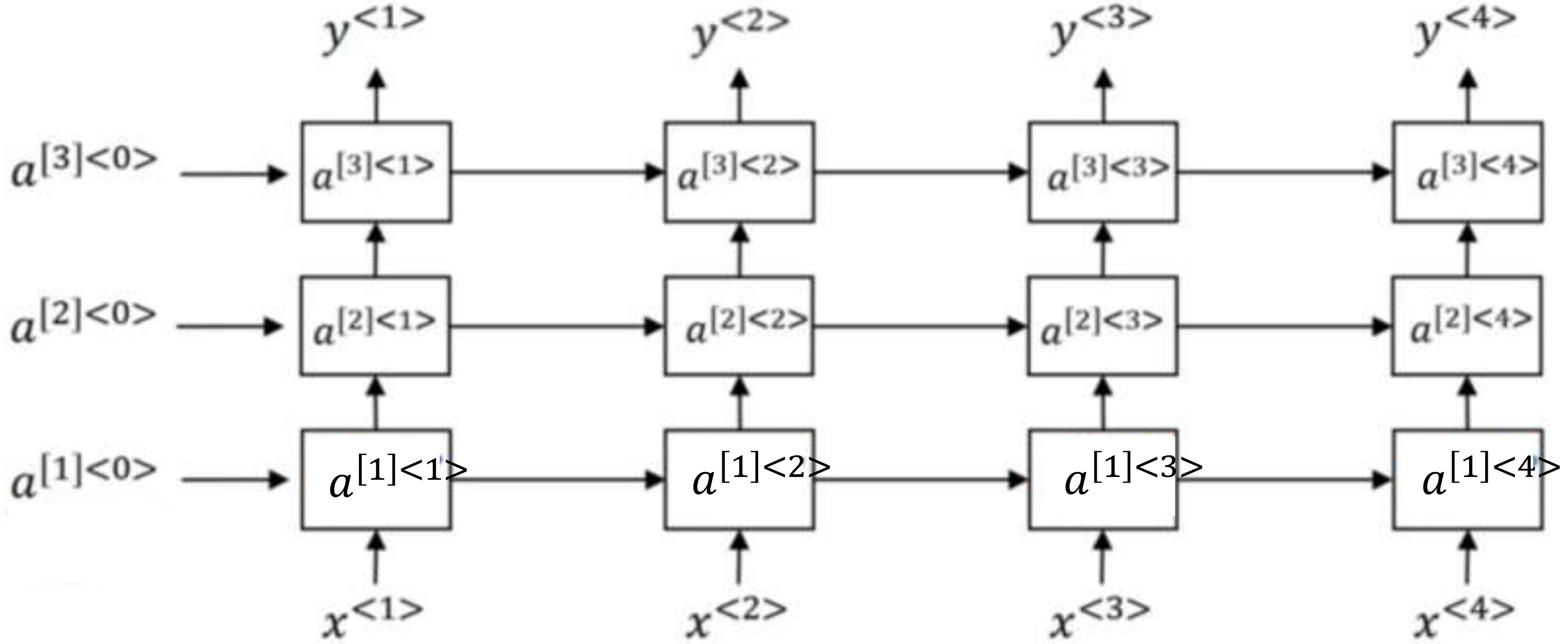

RNN
GRU
LSTM

# Bidirectional RNN




https://towardsdatascience.com/pytorch-basics-how-to-train-your-neural-net-intro-to-rnn-cb6ebc594677

# Deep RNN


$y^{<1>}$
$y^{<2>}$
$y^{<3>}$
$y^{<4>}$
$a^{[3]<0>}$
$a^{[3]<1>}$
$a^{[3]<2>}$
$a^{[3]<3>}$
$a^{[3]<4>}$
$a^{[2]<0>}$
$a^{[2]<1>}$
$a^{[2]<2>}$
$a^{[2]<3>}$
$a^{[2]<4>}$
$a^{[1]<0>}$
$a^{[1]<1>}$
$a^{[1]<2>}$
$a^{[1]<3>}$
$a^{[1]<4>}$
$x^{<1>}$
$x^{<2>}$
$x^{<3>}$
$x^{<4>}$

END