Perceptron Networks and Applications

M. Ali Akcayol
Gazi University
Department of Computer Engineering

Content

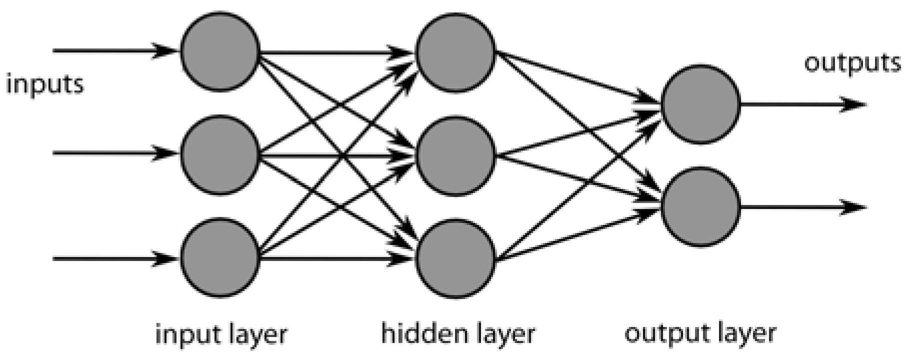
- Recurrent neural networks
- Structure of RNNs
- Feed-forward in RNNs
- RNN training
- RNN architectures
- RNN applications

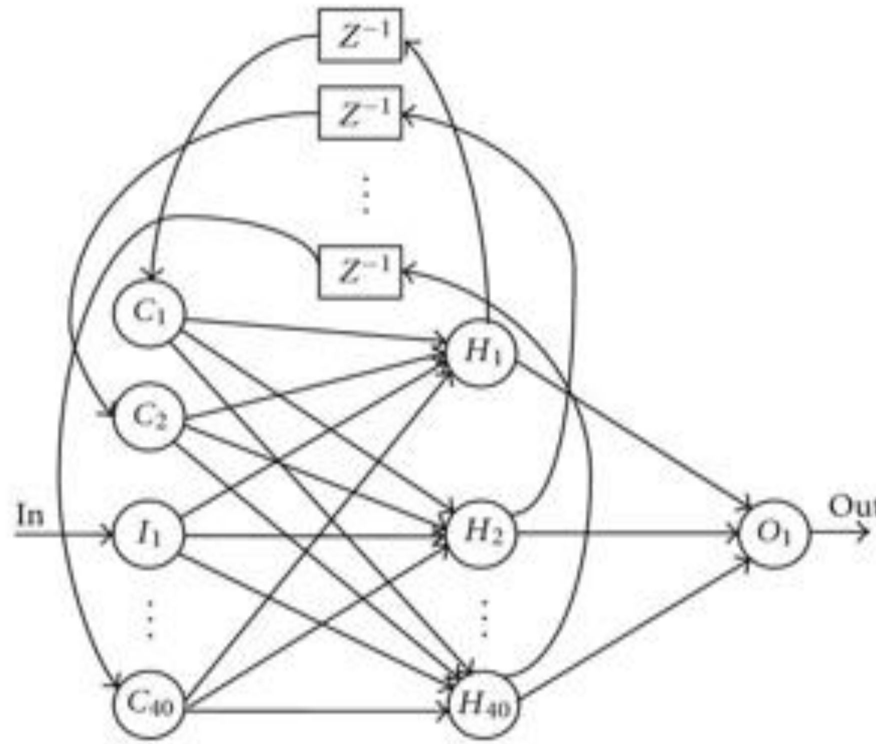
Recurrent neural networks

- All problems can not be expressed with fixed-length inputs and outputs.
- For example, if the number 1 in the input bit sequence is even the output is YES, if odd NO. The previous information should be stored in the system that produces the output (1000010101 -> YES, 100011 -> NO).
- In some problems, a fixed-length input may not always be possible and the input size may be different from the previous ones.
- Recurrent neural networks take the previous output or previous states of the hidden layer as input.
- An input at any time t is a combination of past information and current input.

Recurrent neural networks

- In classical neural networks, there is no correlation between previous states or inputs and current inputs.
- RNNs associate previous inputs or states with the current inputs.



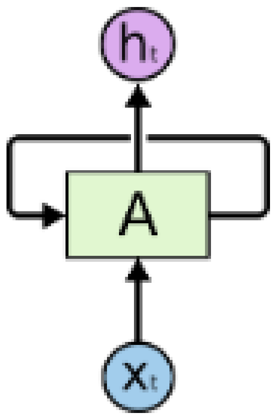


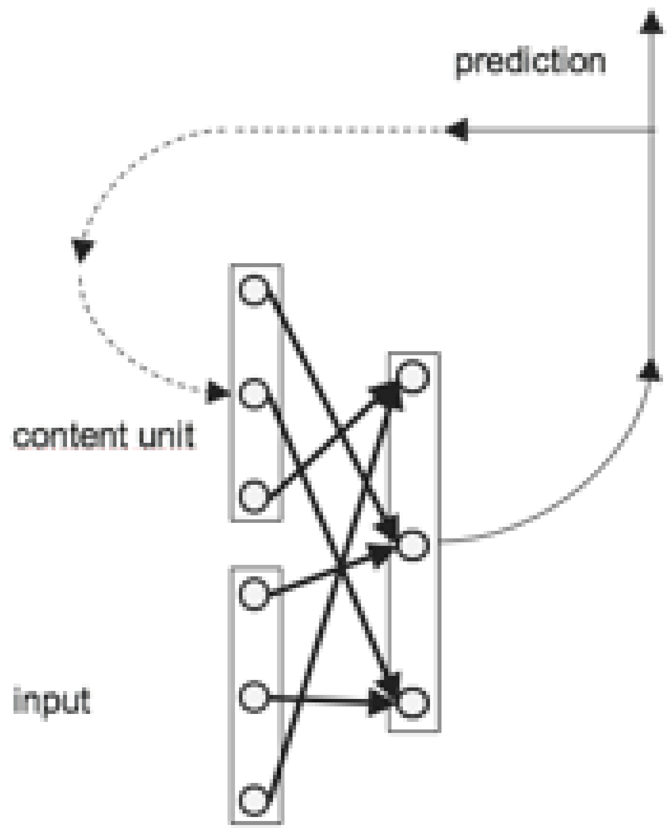
Content

- Recurrent neural networks
- Structure of RNNs
- Feed-forward in RNNs
- RNN training
- RNN architectures
- RNN applications

Structure of RNNs

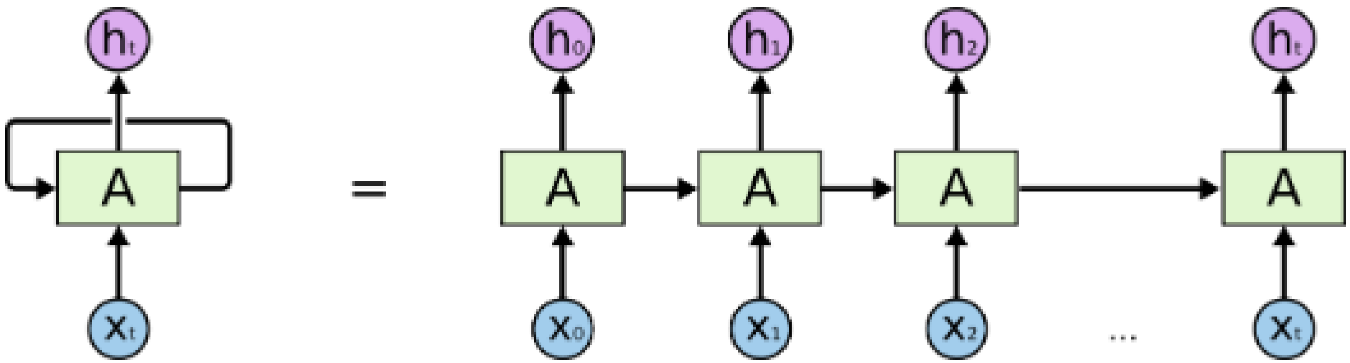
- RNNs have loops.
- In the figure, A shows a neural network, x_t inputs and h_t output.





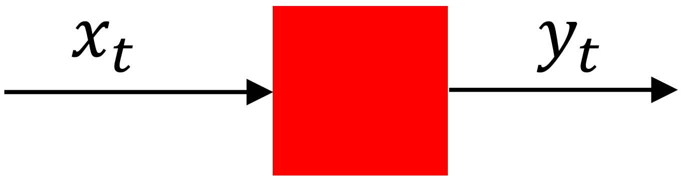
Structure of RNNs

- An RNN can be thought of as multiple copies of a neural network.
- Each neural network passes the information to the next (input).



Structure of RNNs

- In simple feed-forward networks, each output is calculated for its own input.



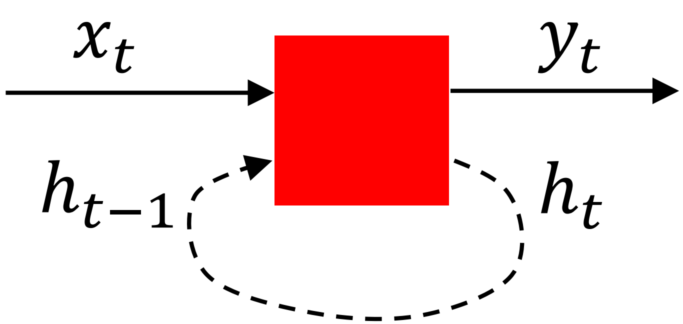
$$y_0 = f(W_x X_0)$$

$$y_1 = f(W_x X_1)$$

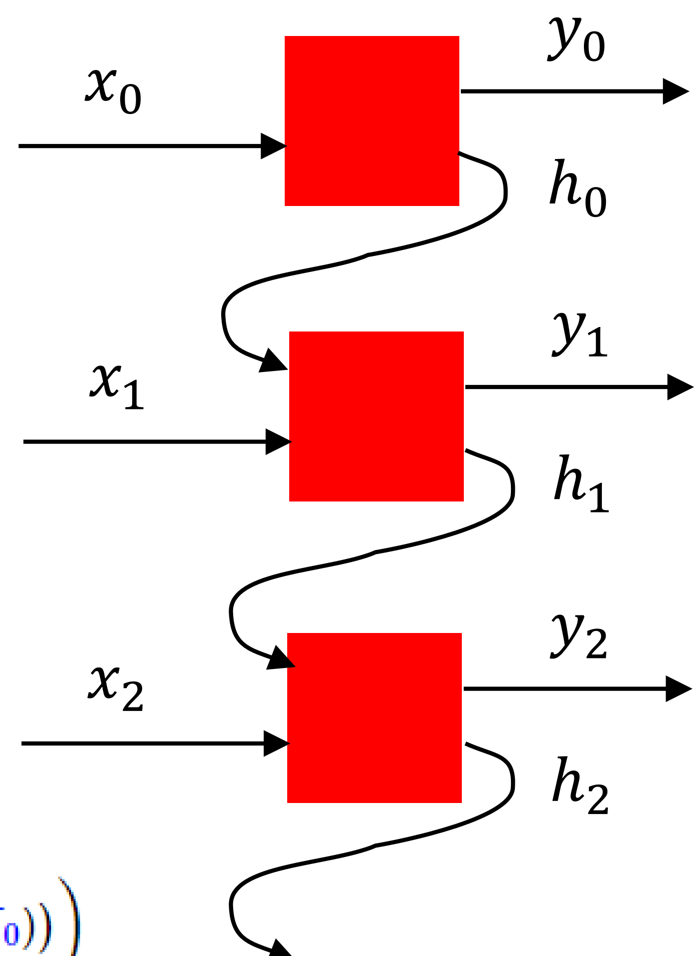
$$y_2 = f(W_x X_2)$$

Structure of RNNs

- In RNNs, each output is calculated based on its own input and the previous output.



One-step delay



$$y_0 = f(W_x X_0)$$

$$y_1 = f\big(W_x X_1 + W_h\, f(W_x X_0)\big)$$

$$y_2 = f\Big(W_x X_2 + W_h\, f\big(W_x X_1 + W_h\, f(W_x X_0)\big)\Big)$$

Structure of RNNs

- The same function and same parameters are used in each discrete time.
- The weights are used by sharing between layers.

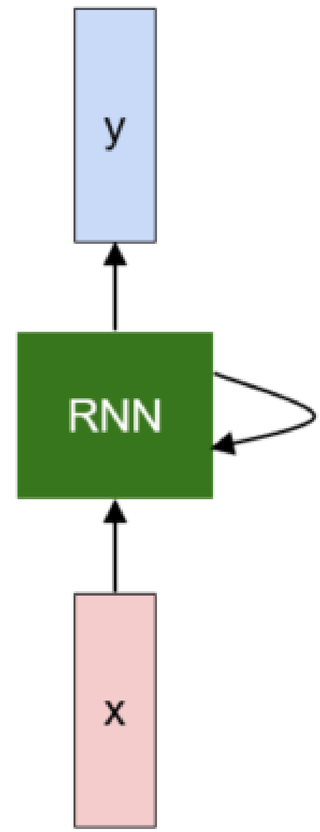
$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

new state

some function with parameters W

old state

input vector at some time step



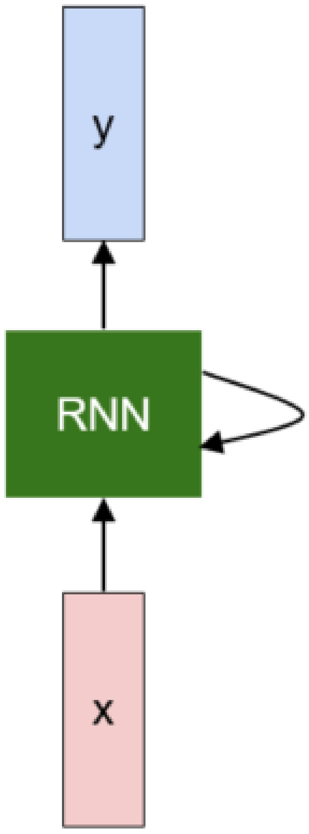
Structure of RNNs

- In RNNs, previous status information affects subsequent outputs at a certain weight.

$$h_t = f_W(h_{t-1}, x_t)$$

$$h_t = \tanh(\boxed{W_{hh}} h_{t-1} + W_{xh} x_t)$$

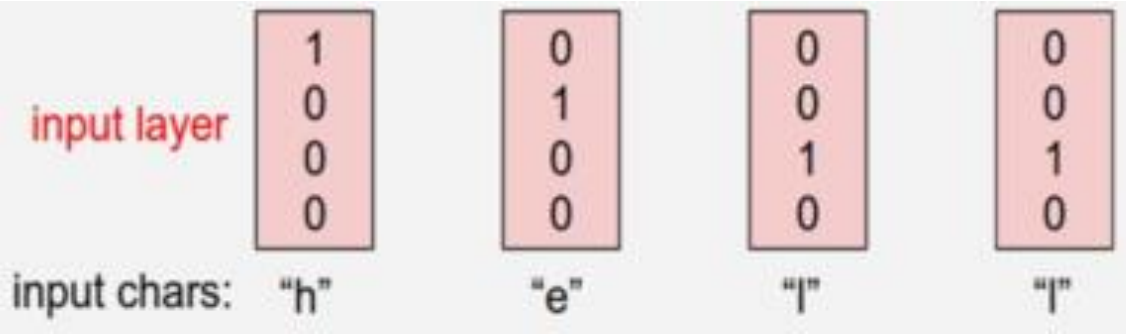
$$y_t = W_{hy} h_t$$



Structure of RNNs

Example

- Let there be 4 letters {h, e, l, o} in the dictionary.
- Let's create an RNN for the word "hello".
- The letters are converted to vector for input.
- Input vectors are created with 1 for each letter in the word and 0 for the others.

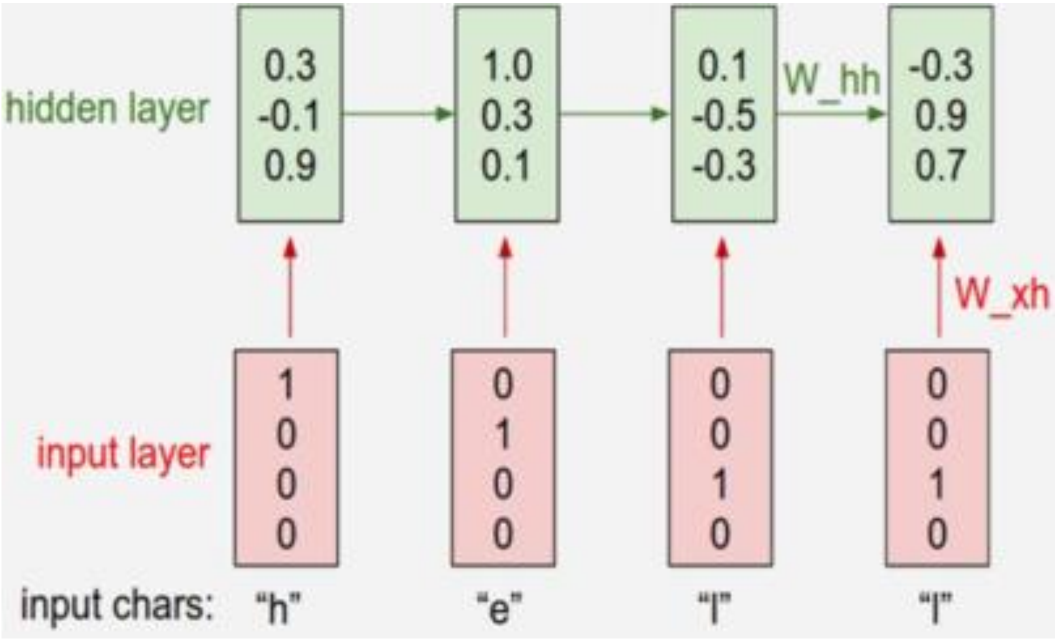


Structure of RNNs

Example – cont.

- The hidden layer outputs are calculated by using the transfer function.

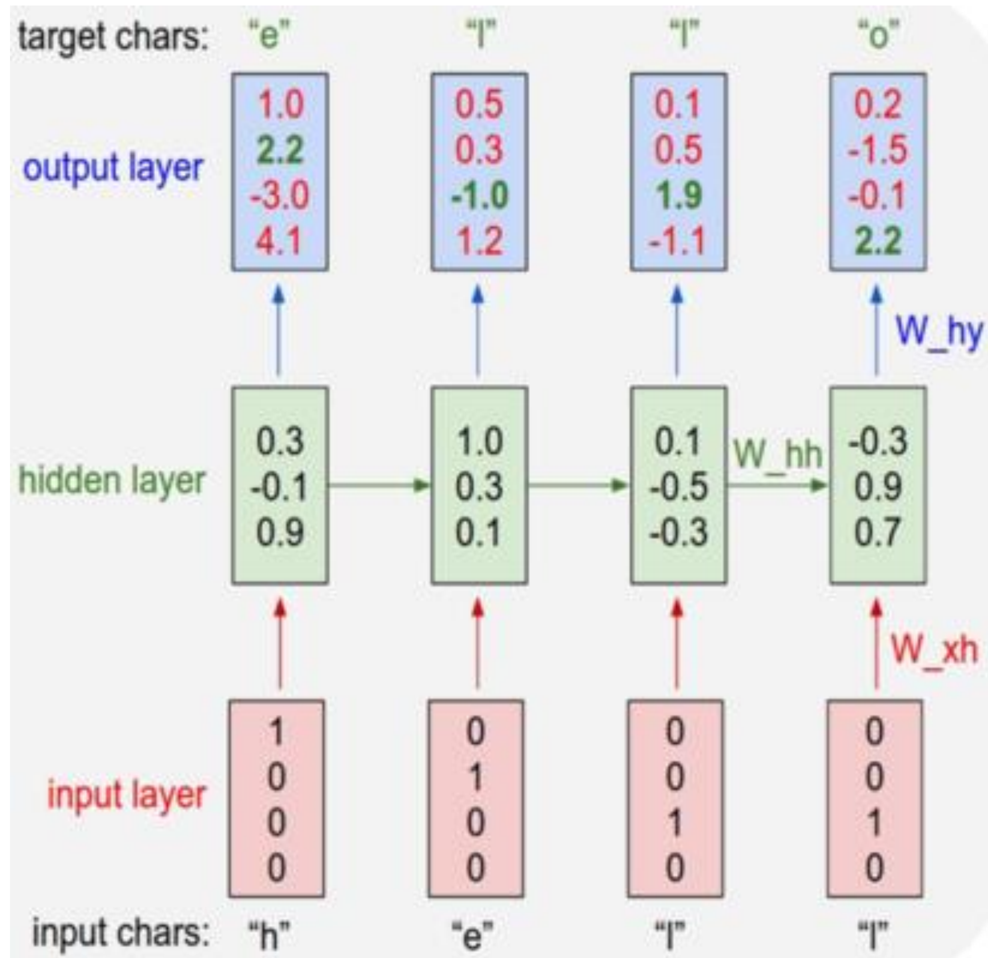
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$



Structure of RNNs

Example – cont.

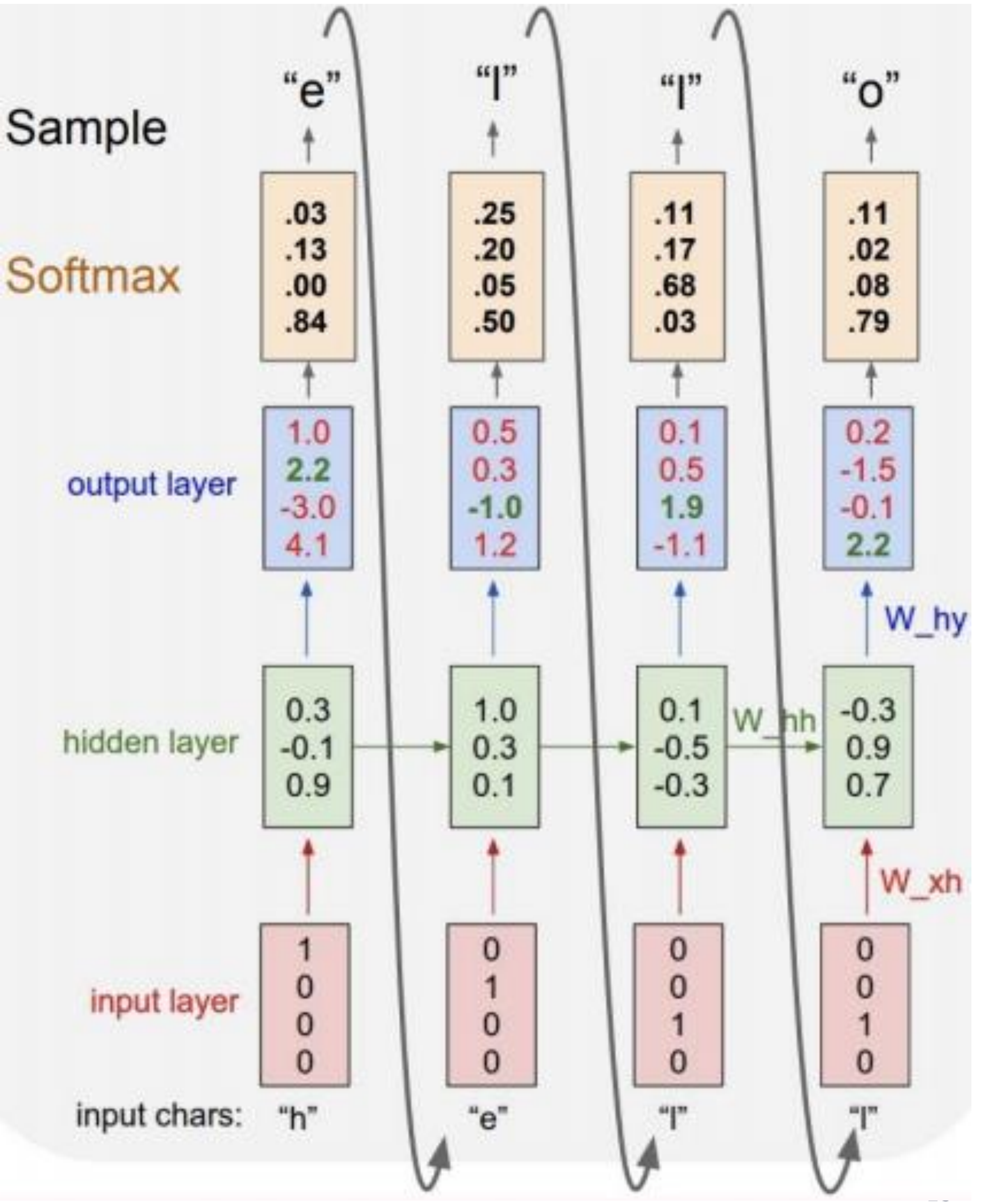
- The error is calculated according to the target output vector.
- The probability that the next character is "e" after the character "h" is given.
- The probability that the next character is "l" after the character "e" is given.
- The probability that the next character "l" is "l" after the character "l" is given.
- The probability that the next character is "o" after the character "l" is given.



Structure of RNNs

Example – cont.

- A word/sentence can be created by transferring the outputs to the input.

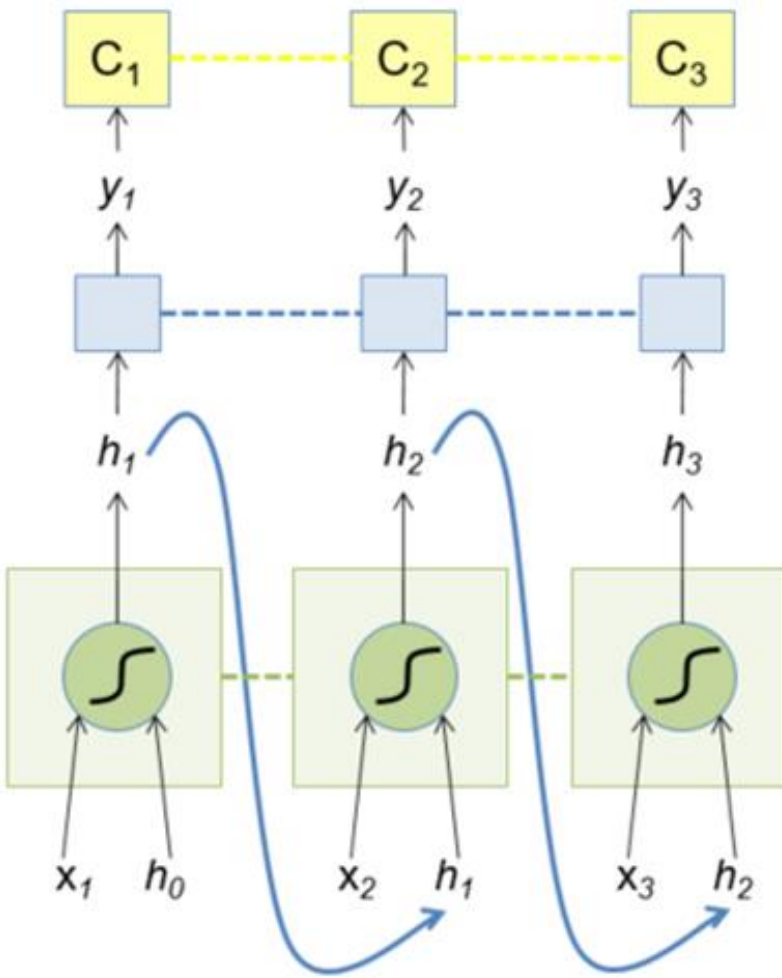


Content

- Recurrent neural networks
- Structure of RNNs
- Feed-forward in RNNs
- RNN training
- RNN architectures
- RNN applications

Feed-forward in RNNs

- The new output is calculated by combining the previous output with the next input.



$$h_t = \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

$$y_t = \mathrm{F}(h_t)$$

$$C_t = \mathrm{Loss}(y_t, \mathrm{T}_t)$$

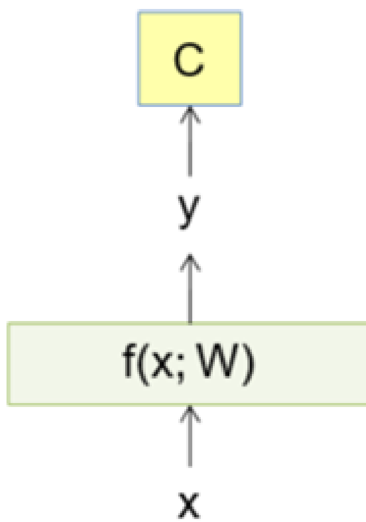
------ indicates shared weights

Content

- Recurrent neural networks
- Structure of RNNs
- Feed-forward in RNNs
- RNN training
- RNN architectures
- RNN applications

RNN training

- Training for RNNs is accomplished by the backpropagation Through Time (BPTT).
- The weights are changed according to the error at the output.



$$y = f(x; W)$$

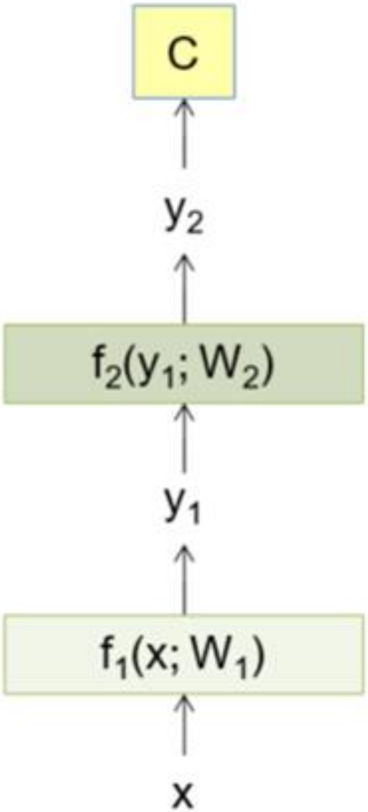
$$C = \mathrm{Loss}(y, y_T)$$

$$W \leftarrow W - \eta \frac{\partial C}{\partial W}$$

$$\frac{\partial C}{\partial W} = \left(\frac{\partial C}{\partial y}\right)\left(\frac{\partial y}{\partial W}\right)$$

RNN training

- In multilayer structures, the weights are changed by back propagation.



$$y_1 = f_1(x; W_1)$$
$$y_2 = f_2(y_1; W_2)$$
$$C = \mathrm{Loss}(y, y_T)$$

$$W_2 \leftarrow W_2 - \eta \frac{\partial C}{\partial W_2}$$
$$W_1 \leftarrow W_1 - \eta \frac{\partial C}{\partial W_1}$$

$$\frac{\partial C}{\partial W_2} = \left(\frac{\partial C}{\partial y_2}\right)\left(\frac{\partial y_2}{\partial W_2}\right)$$

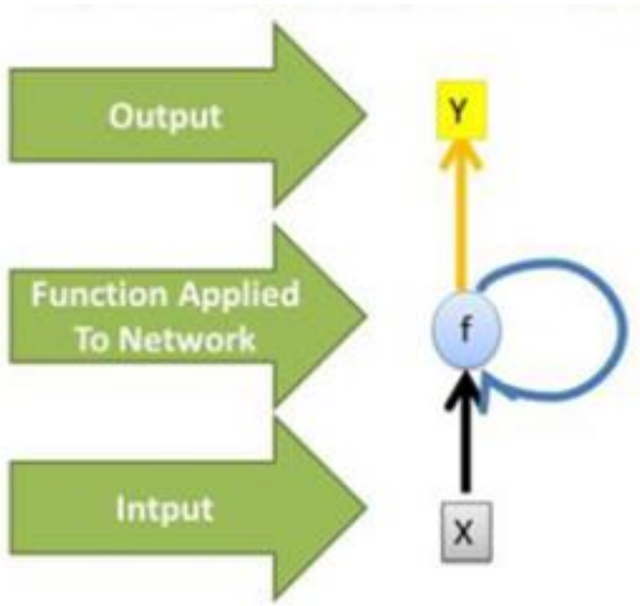
$$\frac{\partial C}{\partial W_1} = \left(\frac{\partial C}{\partial y_1}\right)\left(\frac{\partial y_1}{\partial W_1}\right) = \left(\frac{\partial C}{\partial y_2}\right)\left(\frac{\partial y_2}{\partial y_1}\right)\left(\frac{\partial y_1}{\partial W_1}\right)$$

Content

- Recurrent neural networks
- Structure of RNNs
- Feed-forward in RNNs
- RNN training
- RNN architectures
- RNN applications

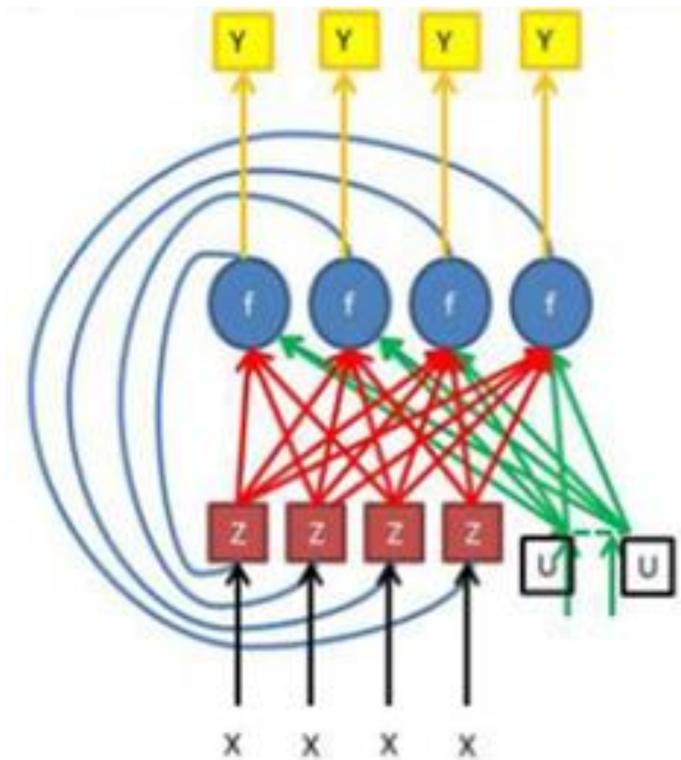
RNN architectures

- Simple RNN architecture is as follows.
- The input, output and previous state.
- The previous state is transferred to the entry with the next entry.



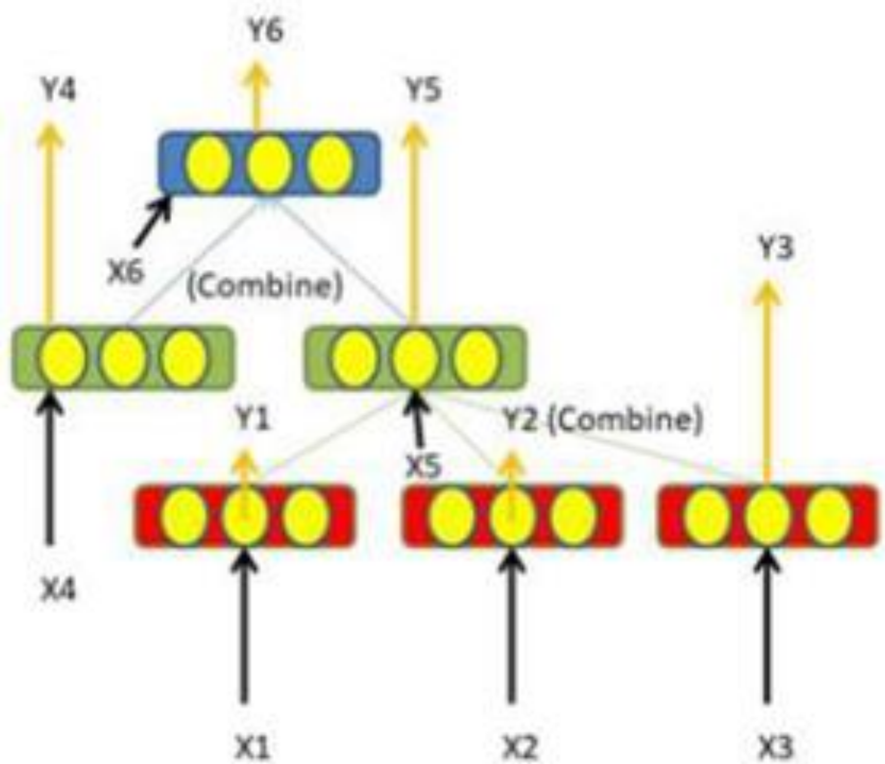
RNN architectures

- In fully connected RNNs, all outputs from the previous state are transferred to inputs.
- The feedback weight values decide the effect of the previous outputs on the next input values.



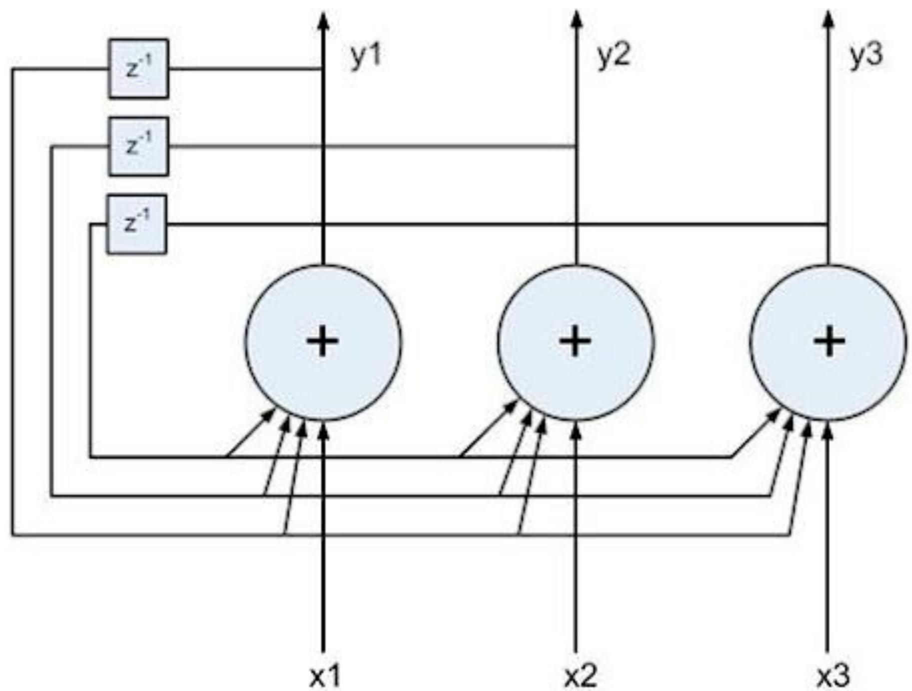
RNN architectures

- In recursive neural networks, the specified layer can be used as input and output values can be obtained from the determined layer.
- Each layer combines the previous layers as input.



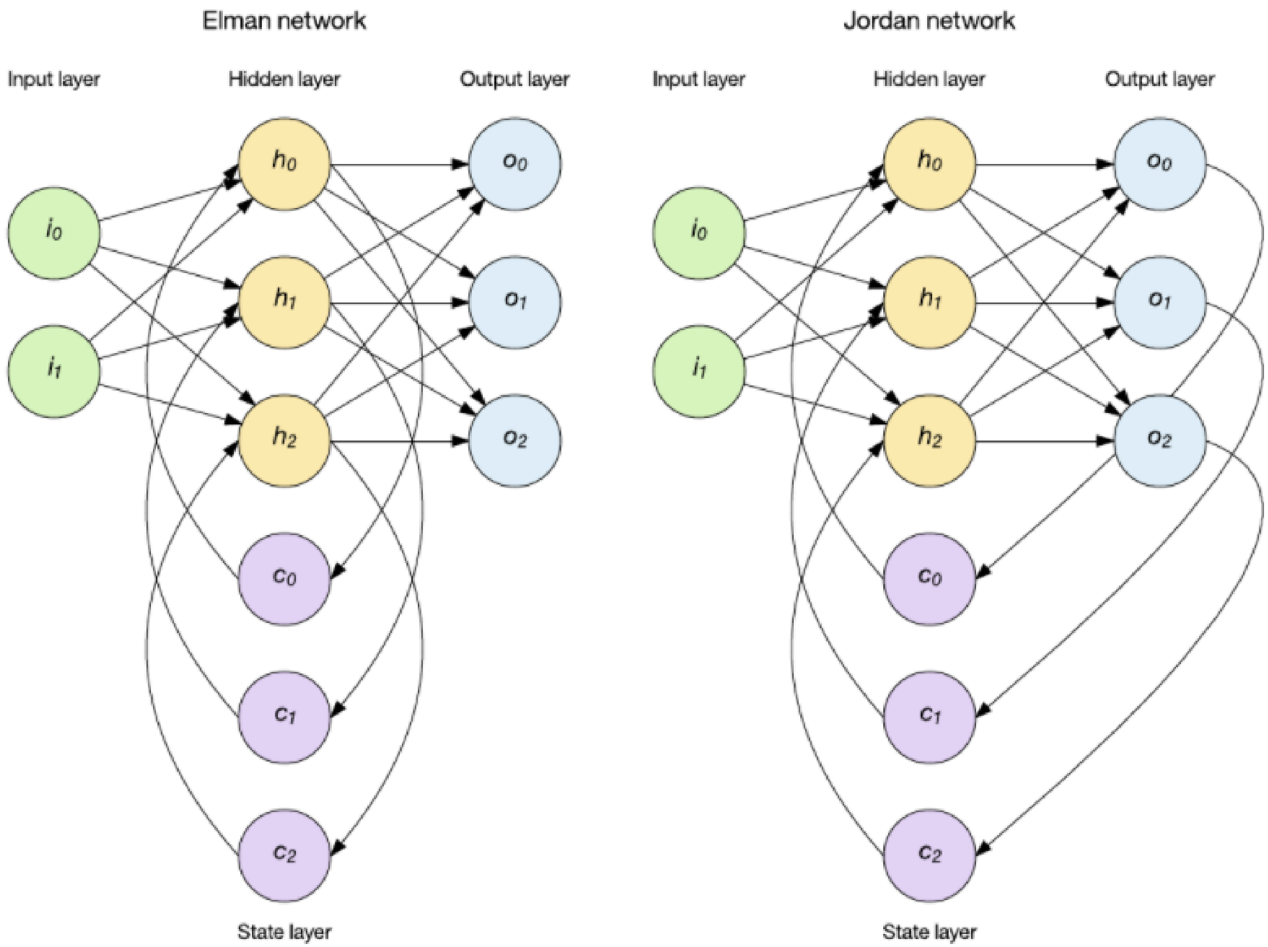
RNN architectures

- In the Hopfield network, all outputs are transferred to all inputs to combine with the next input.
- Depending on the problem type, some outputs can be transferred only selected input nodes.



RNN architectures

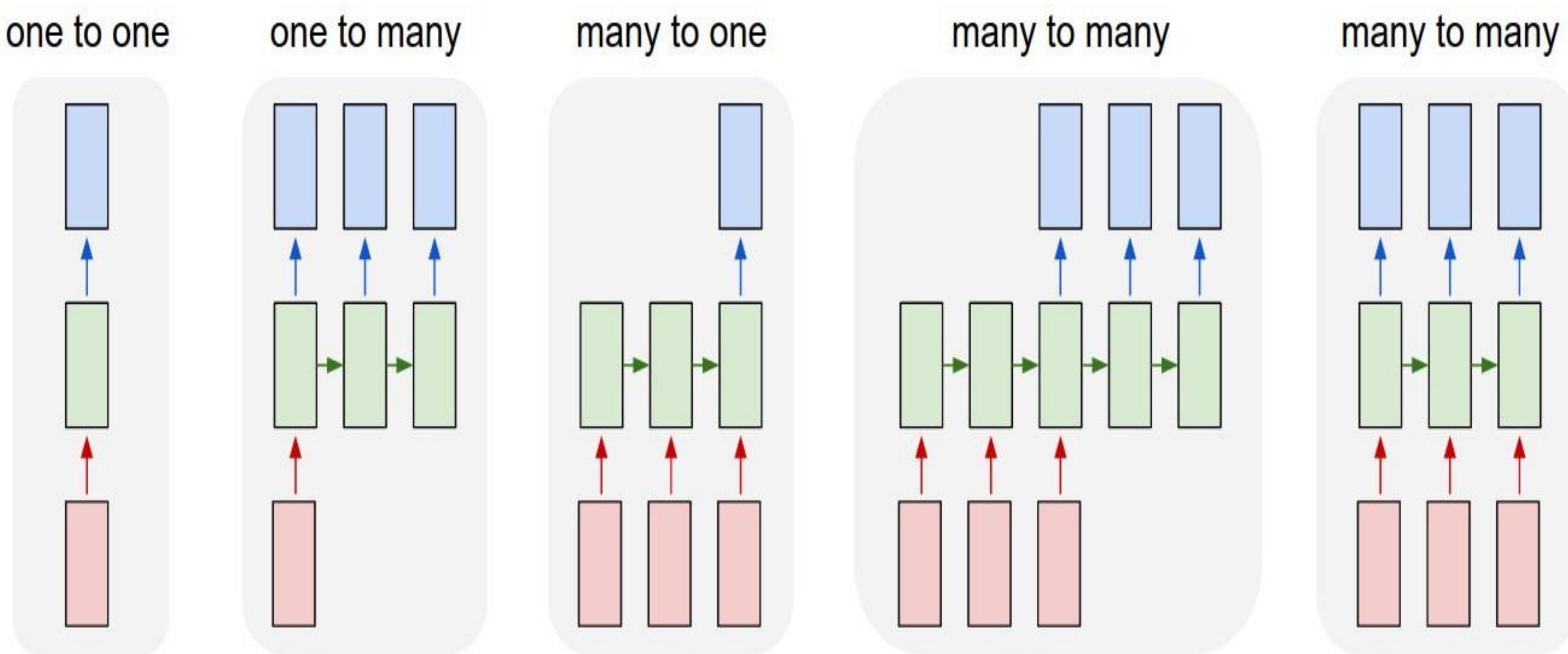
- In the Elman network, the output values in the hidden layer are transferred to the inputs.
- In the Jordan network, the output values are transferred to the inputs.



Content

- Recurrent neural networks
- Structure of RNNs
- Feed-forward in RNNs
- RNN training
- RNN architectures
- RNN applications

RNN applications



Vanilla Neural Networks
(image classification)

Sentiment analysis
sequence of words -> sentiment

Video classification
(Frame labelling)

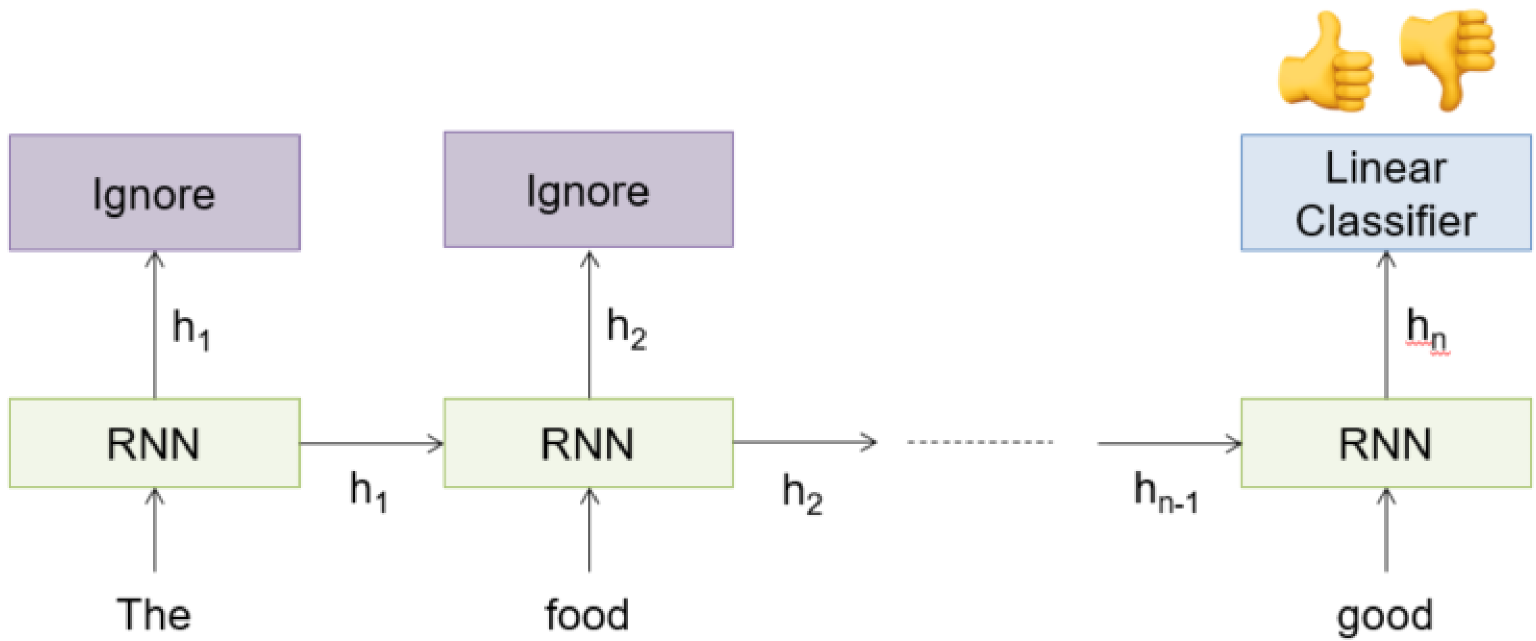
Image captioning
image -> sequence of words

Machine translation
sequence of words -> sequence of words

RNN applications

Sentiment Classification

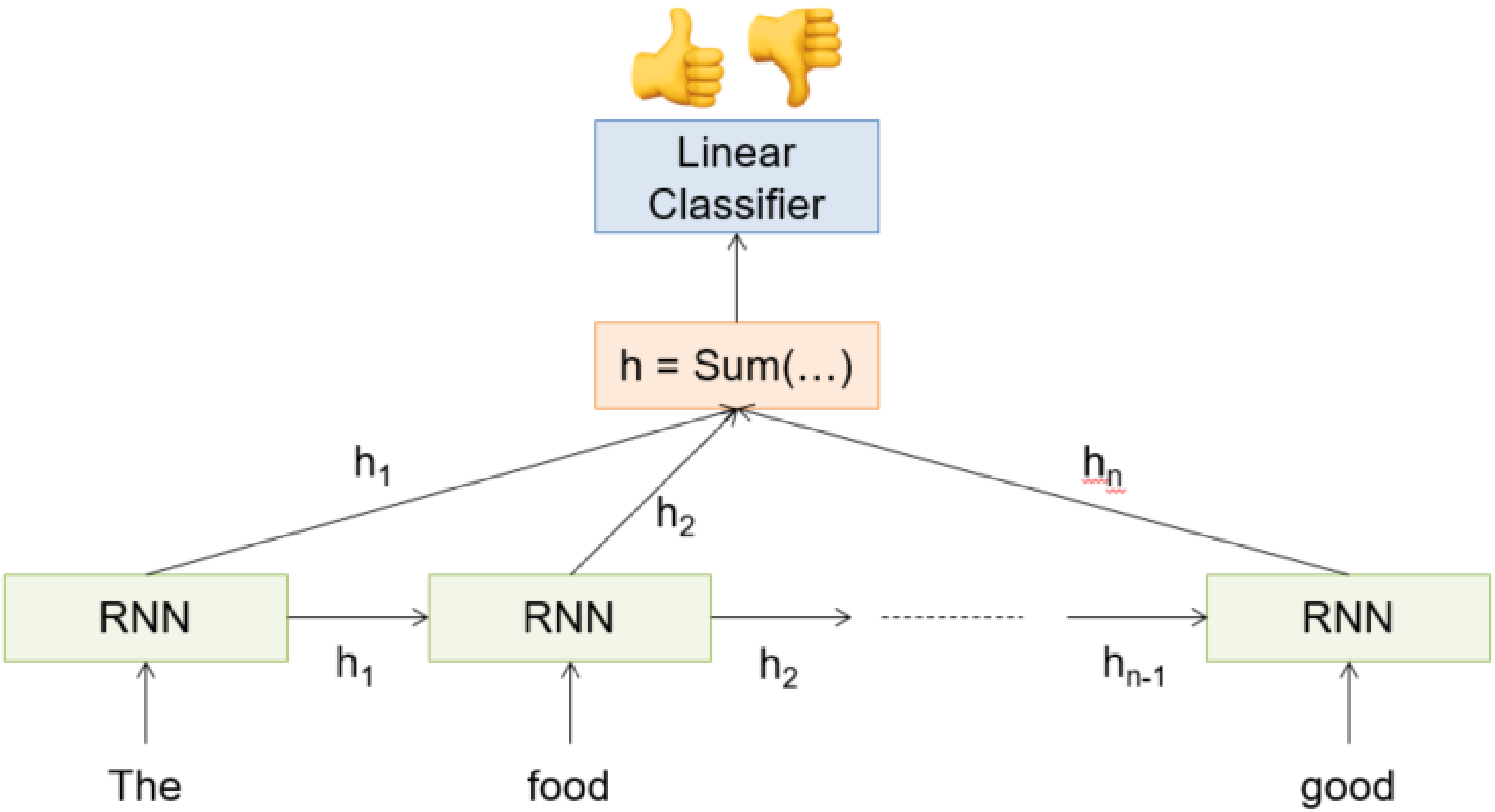
- The RNN is trained with a large number of sentences.
- Then, sentiment classification is predicted for the input sentences.
- One output can be taken and the others can be ignored.



RNN applications

Sentiment Classification

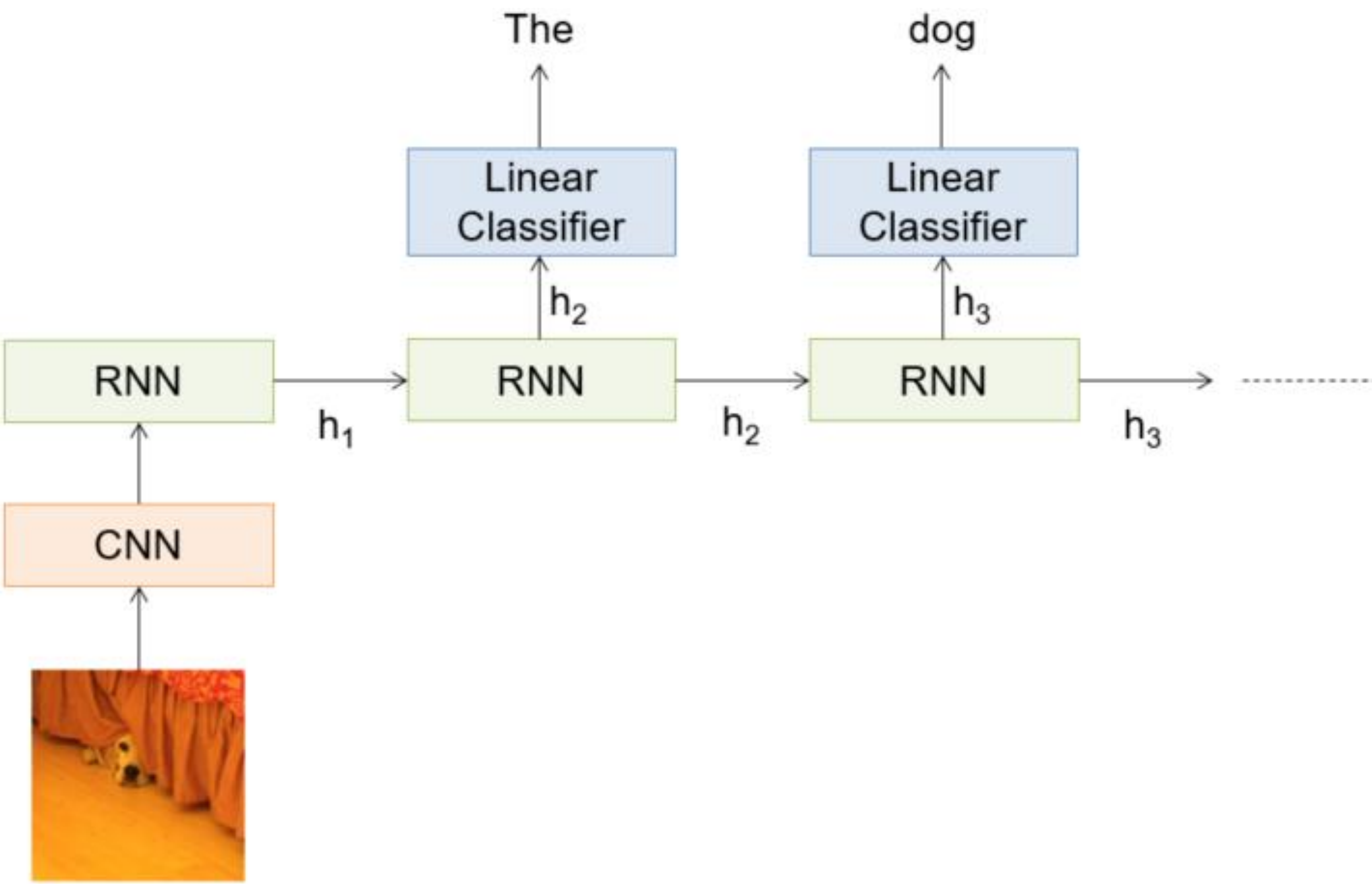
- The sum of all outputs can also be combined.



RNN applications

Image Captioning

- RNNs are used in image captioning applications with CNN.
- CNN is used to extract features from image, RNN is used to create caption for the image.



RNN applications

Image Captioning

- Image captioning applications with RNN.