CoE202
Fundamentals of Artificial intelligence
<Big Data Analysis and Machine Learning>

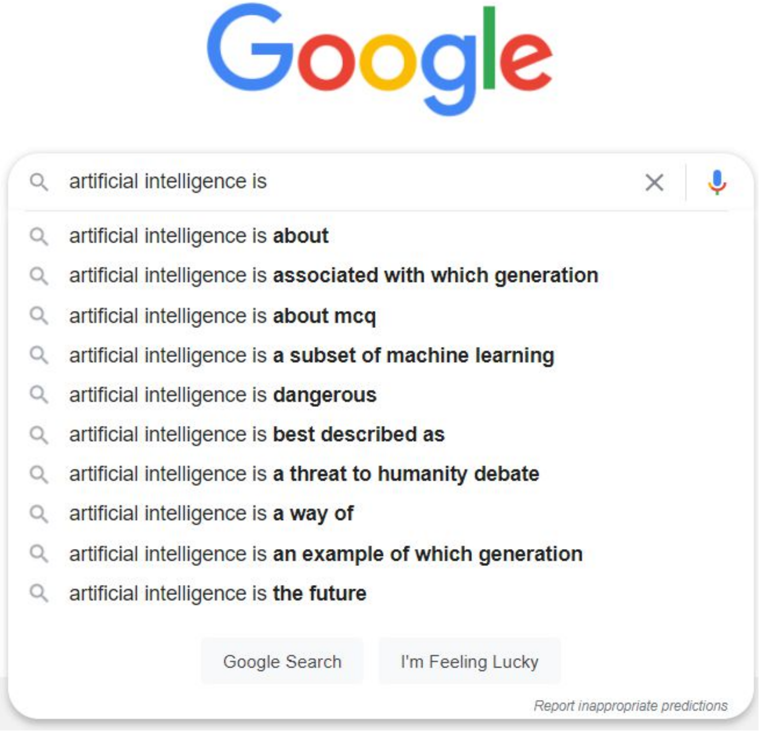
Recurrent Neural Network

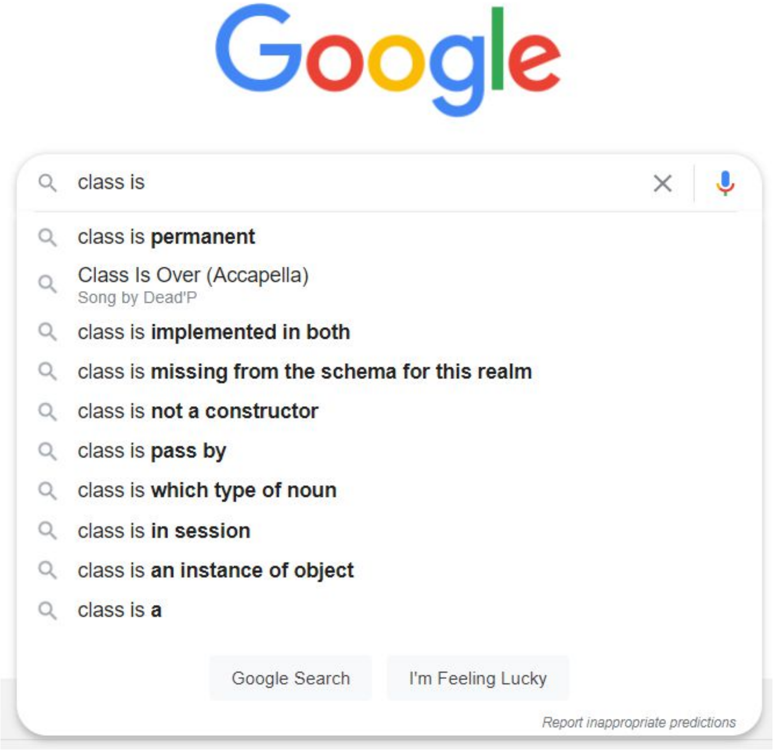
Prof. Young-Gyu Yoon
School of EE, KAIST

Contents

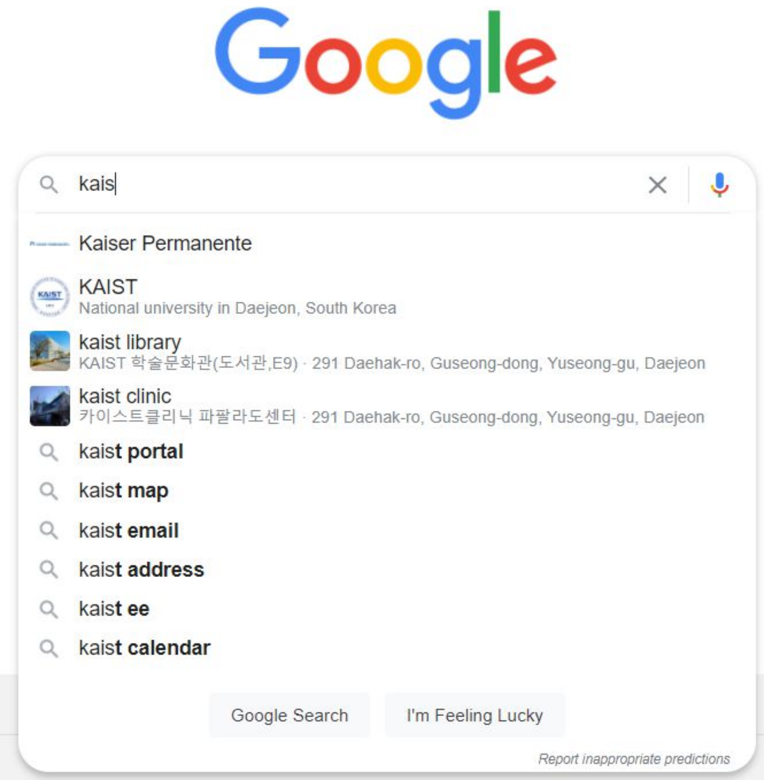
- Recap
  - Joint probability distribution function
  - Discriminative model vs generative model
  - Generative adversarial network
    - Based on two competing networks
- Sequence prediction problem
- Recurrent neural network
  - Concept
  - Unit cell
- Training RNN
- Advanced unit cells

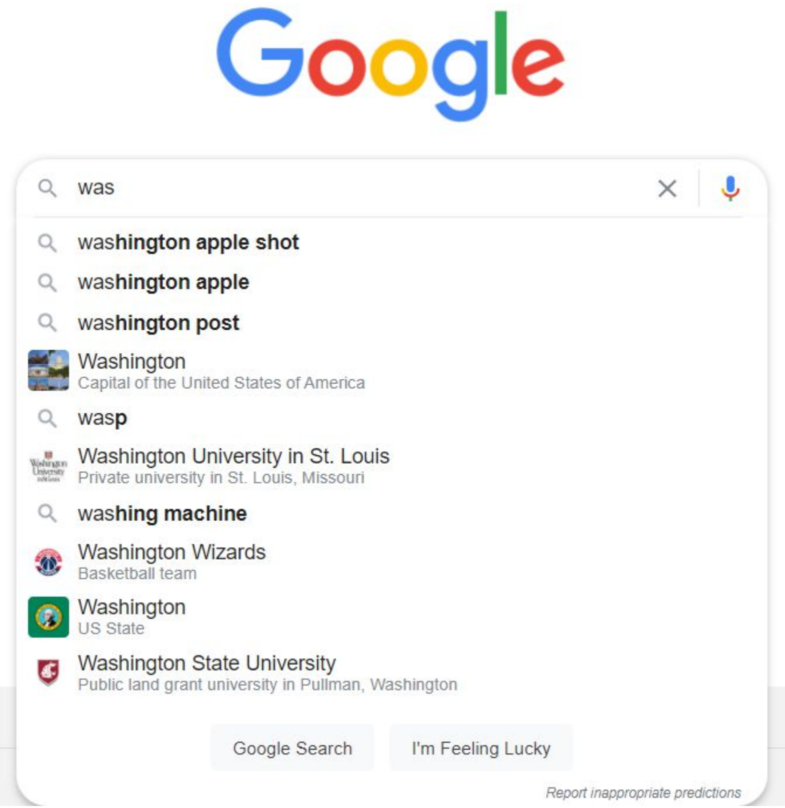
Word prediction problem





and it doesn’t have to be a word



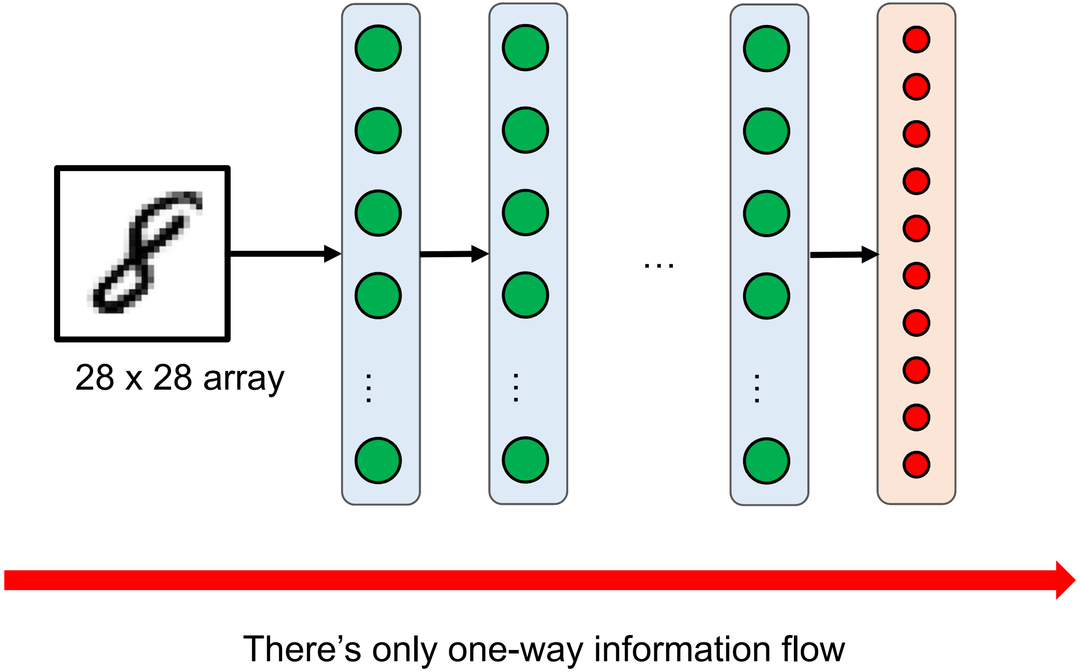


Characteristics of the problem

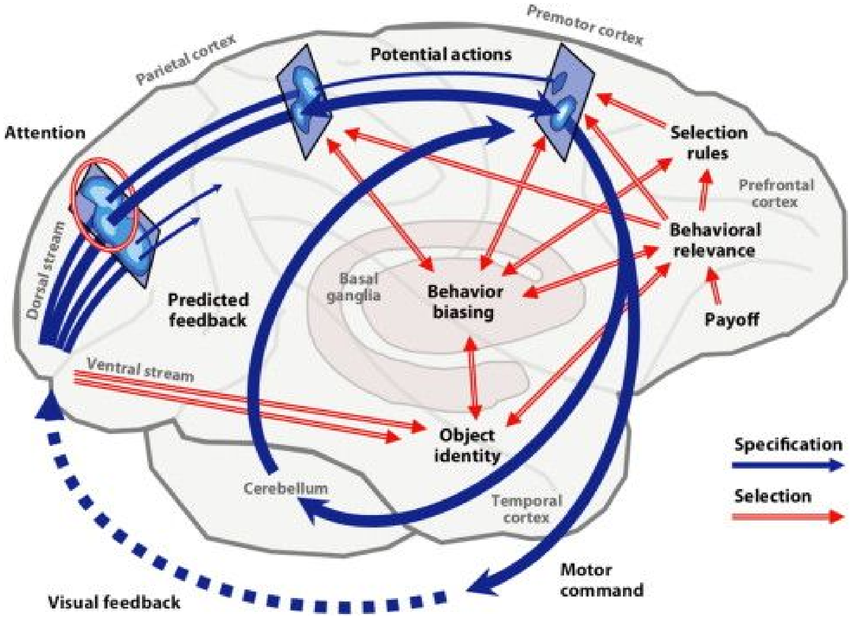
- Input is a sequence of characters or words
- Input length can be arbitrary
- The prediction may depend on the entire input (at least it does not depend solely on the latest input)

- Simple neural network (MLP) or vanilla convolutional neural network is not suited for solving this type of problem

Feedforward neural network

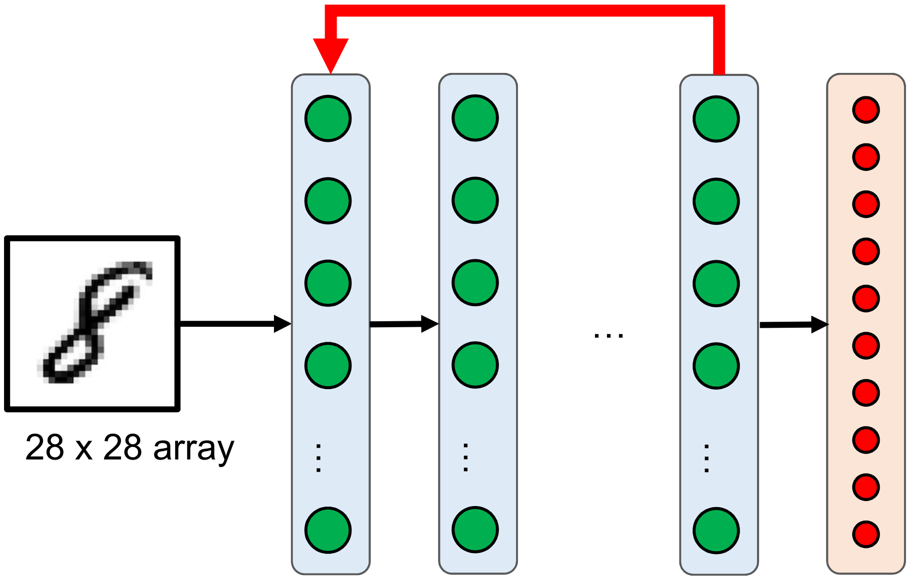


Information flow in a biological brain…



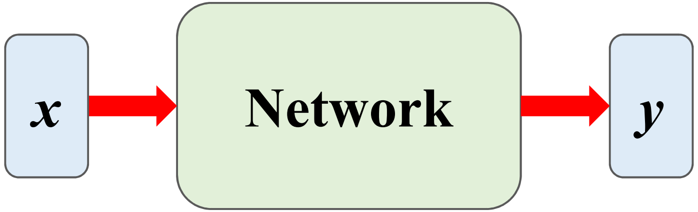
…is not one-way, obviously

RECURENT neural network?

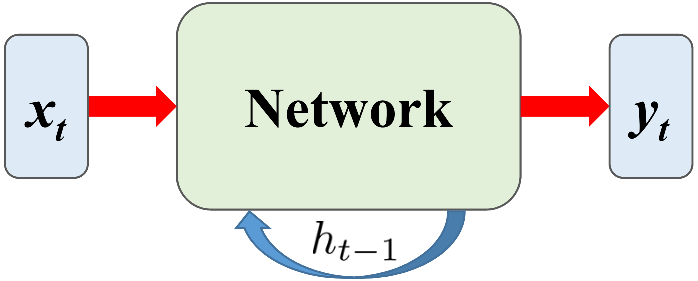


What if we add a **recurrent** path like this?

Feedforward vs. Recurrent

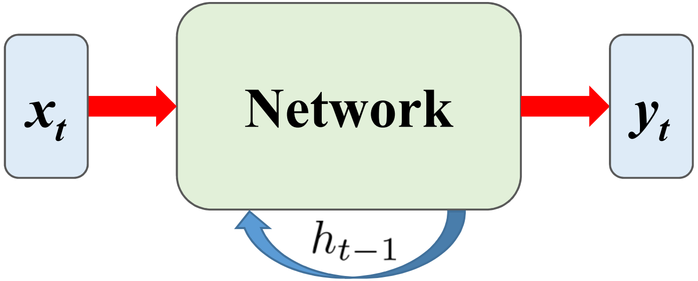


$$y = f(x)$$



$$y_t = f(x_t, h_{t-1})$$

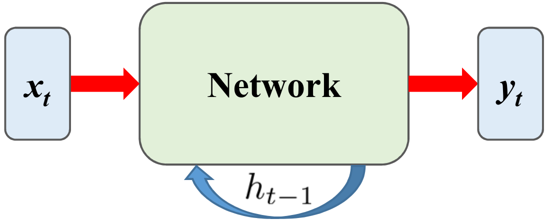
Recurrent neural network (RNN)



$$y_t = f(x_t, h_{t-1})$$

- **RNN**: a type of neural network that contains loops, allowing information to be stored within the network

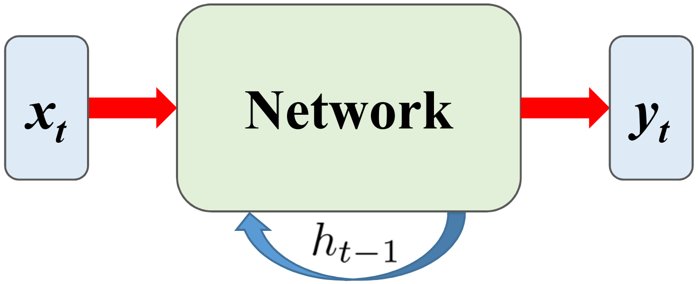
Recurrent neural network (RNN)



$$y_t = f(x_t, h_{t-1})$$

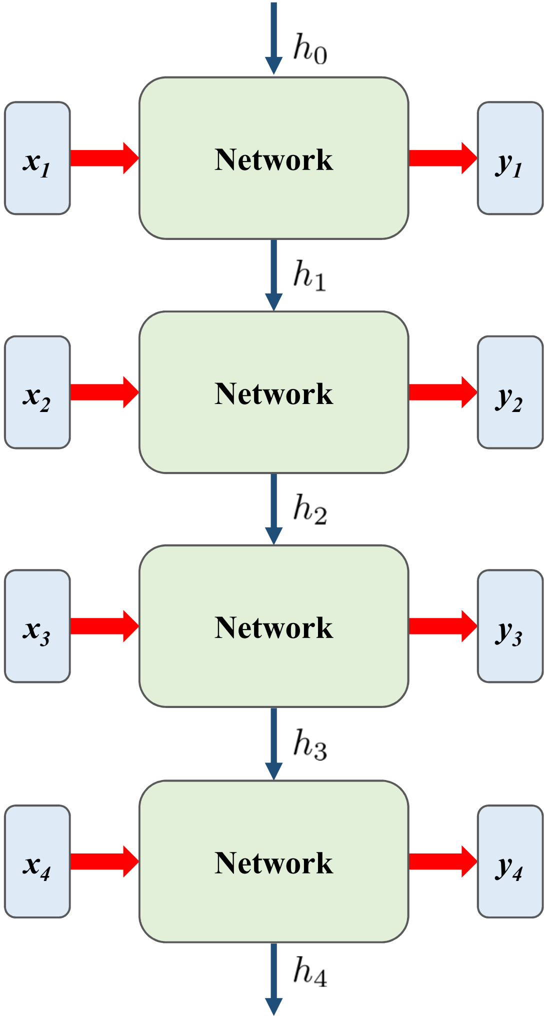
- RNN can take a ‘sequence’ of input
- RNN has “memory”
- RNN can handle input with ‘arbitrary length’

Unfolding RNN

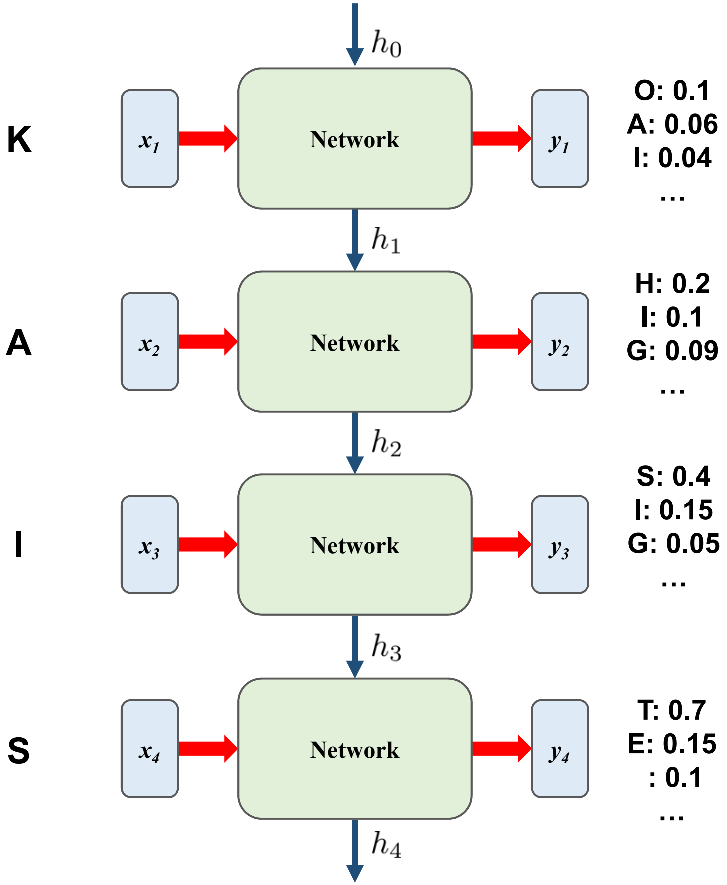


$$y_t = f(x_t, h_{t-1})$$

equivalent



RNN for sequence prediction



Character-by-character prediction example

Possible use of RNN

One to many

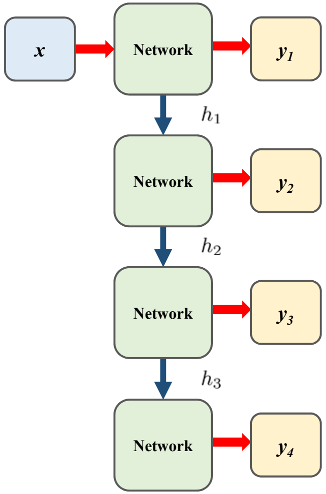
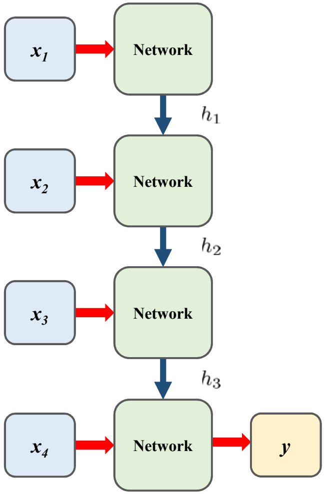


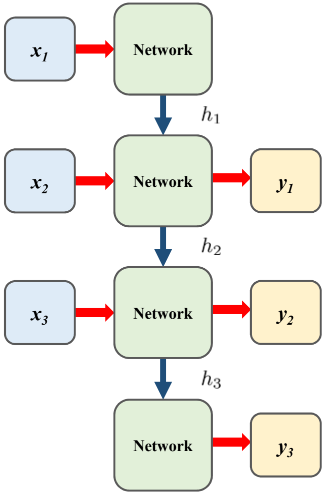
image captioning

many to one



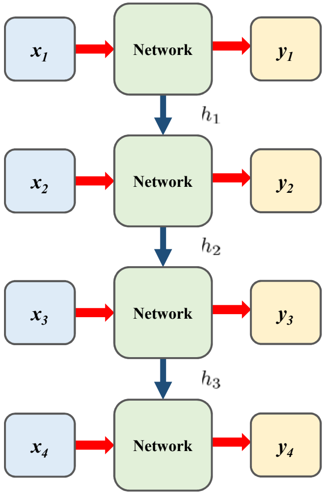
language detection
meaning of word

many to many



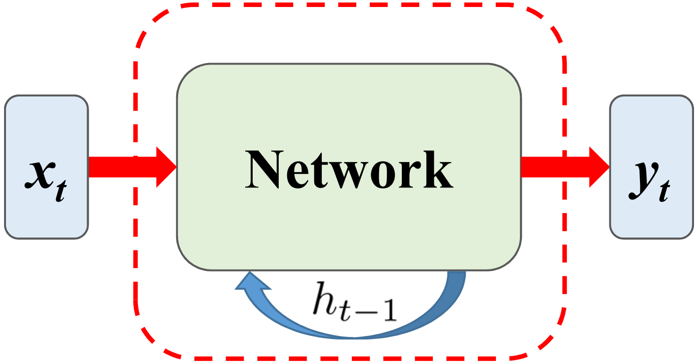
language translation

many to many



character prediction

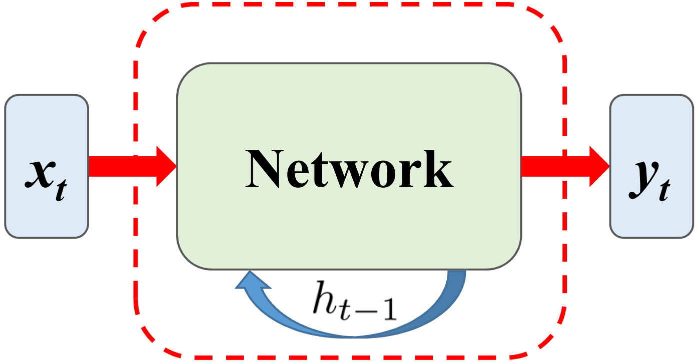
Unit cell of RNN?



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

- Constraint
  - Takes two inputs: x_t, h_{t-1}
  - Returns two outputs: y_t, h_t
- …and that's pretty much it!

Unit cell of RNN



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

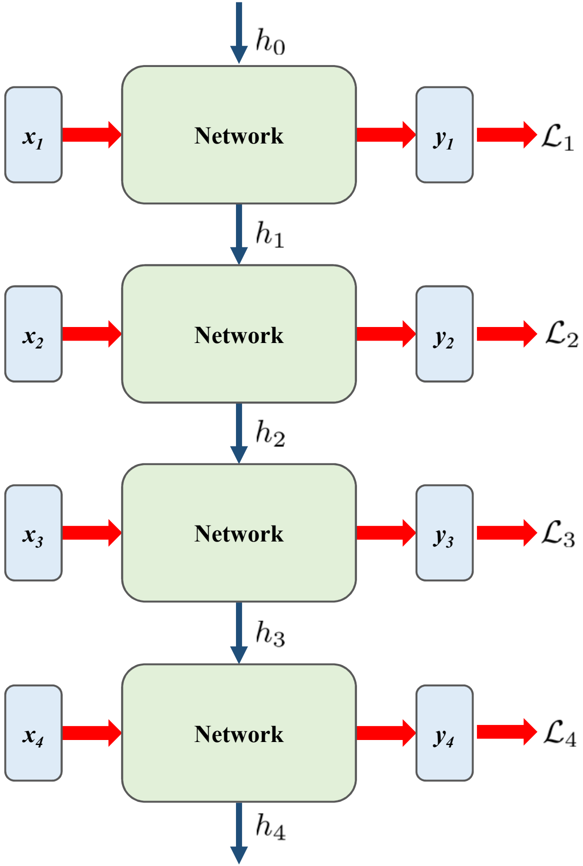
$$h_t = g_1(W_1 x_t + W_2 h_{t-1} + b_1)$$

$$y_t = g_2(W_3 h_t + b_2)$$

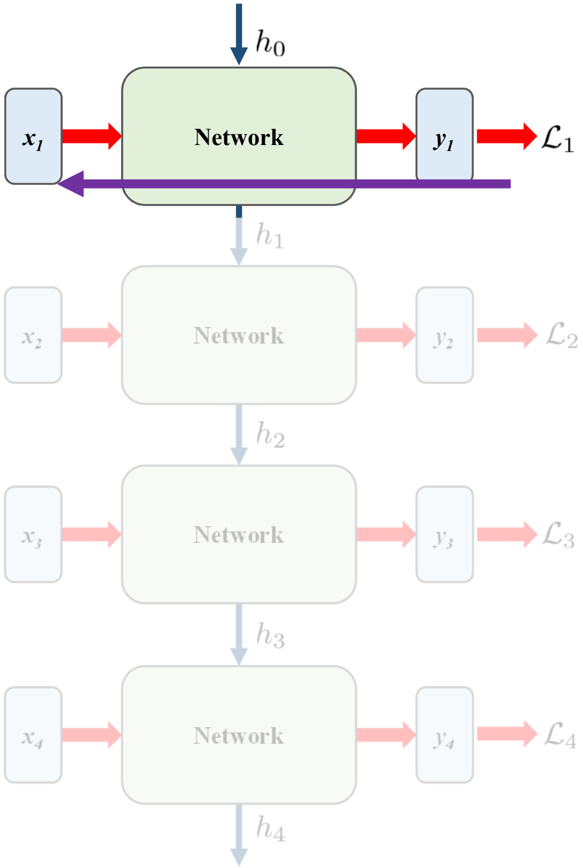
can be merged with the weight matrix

g_1, g_2 are activation functions

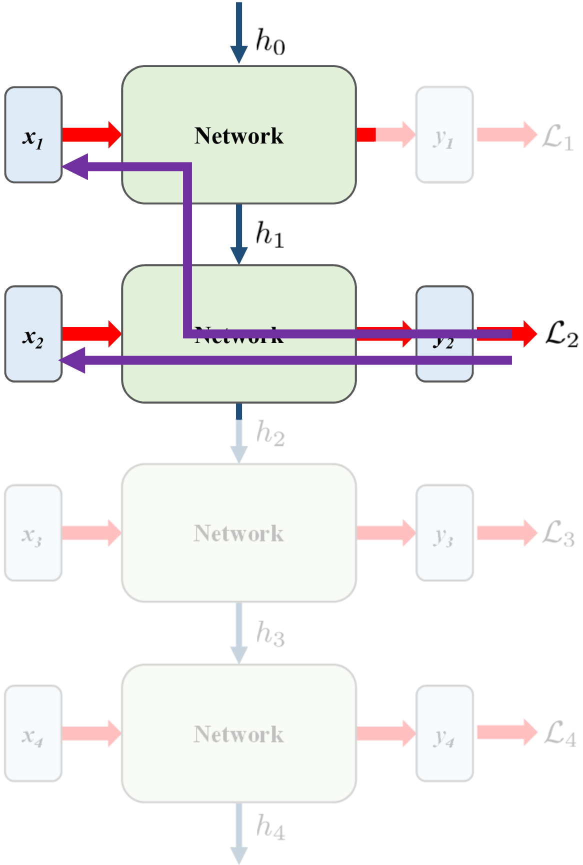
Training: backpropagation in RNN



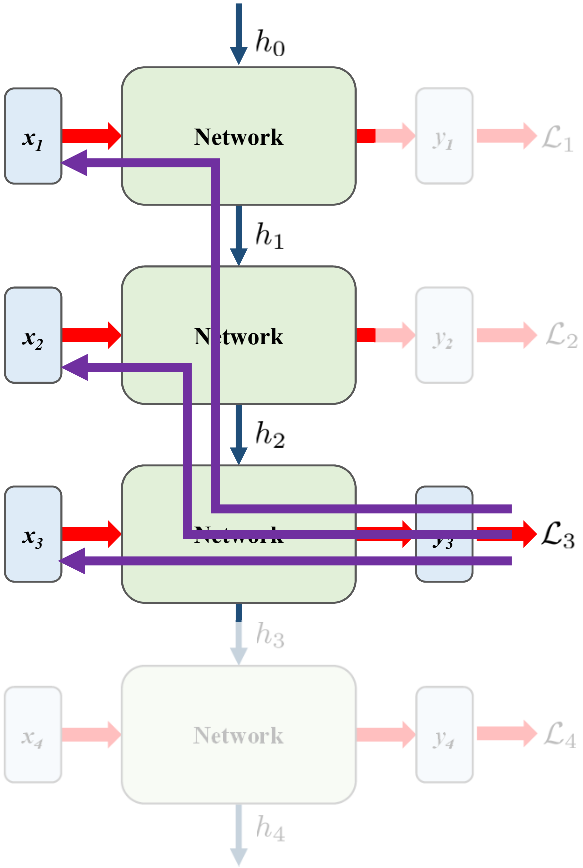
Backpropagation through time



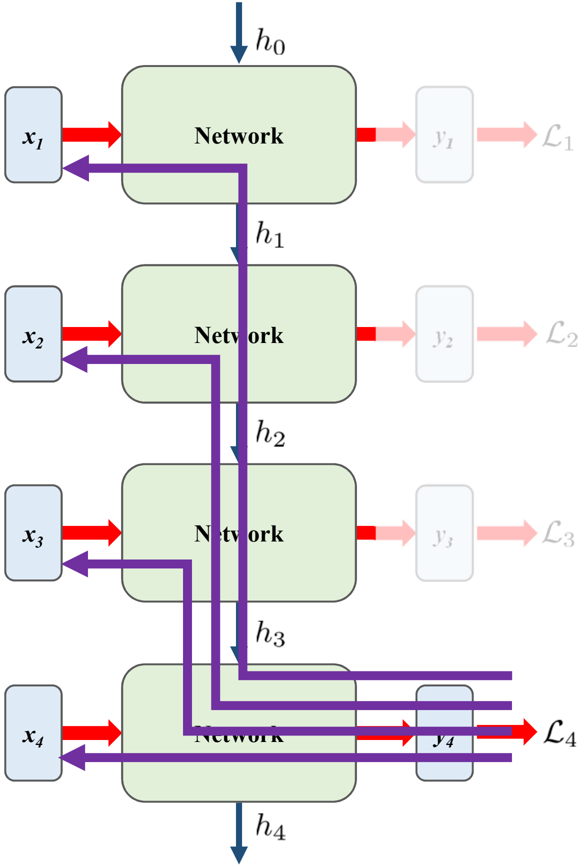
Backpropagation through time



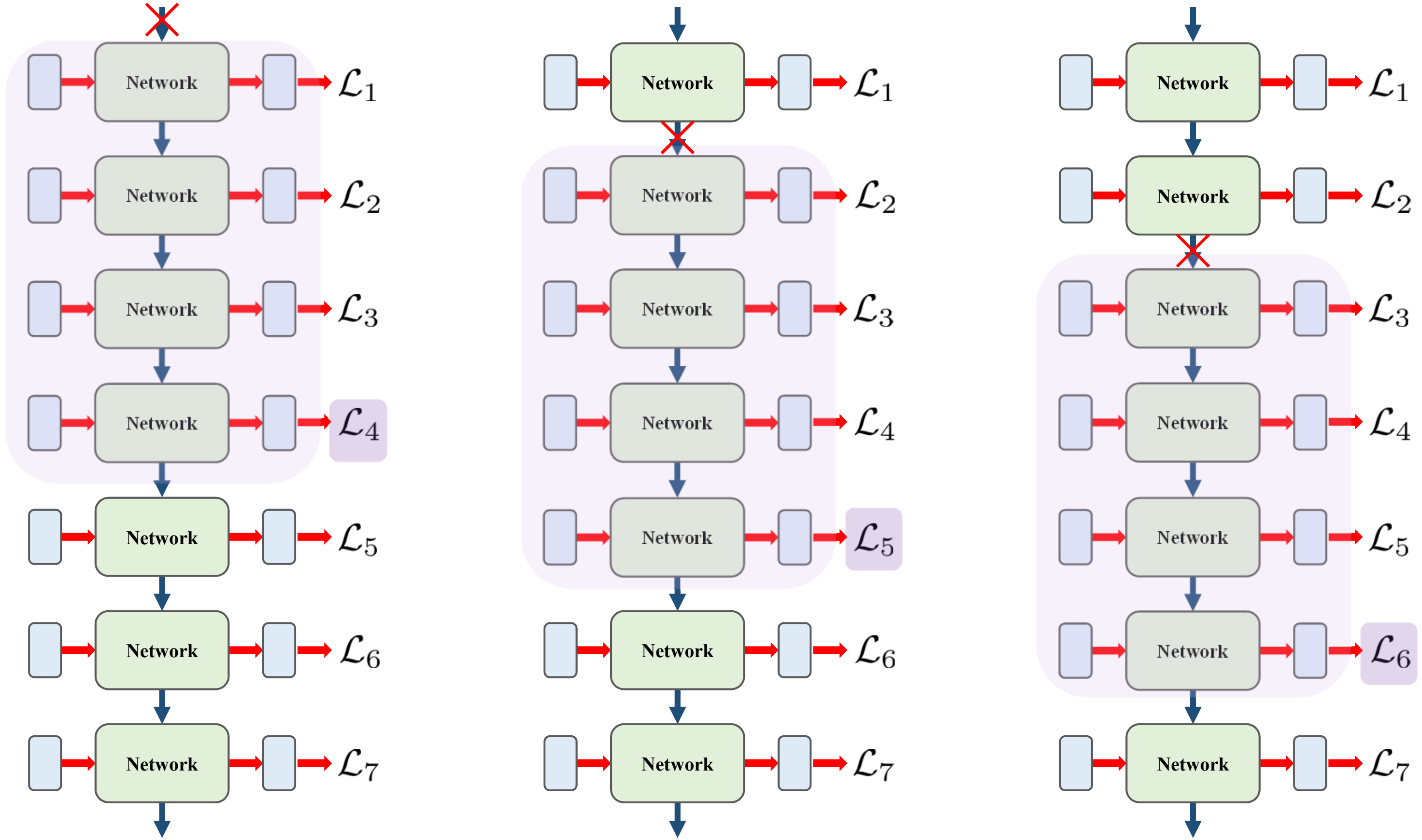
Backpropagation through time



Backpropagation through time



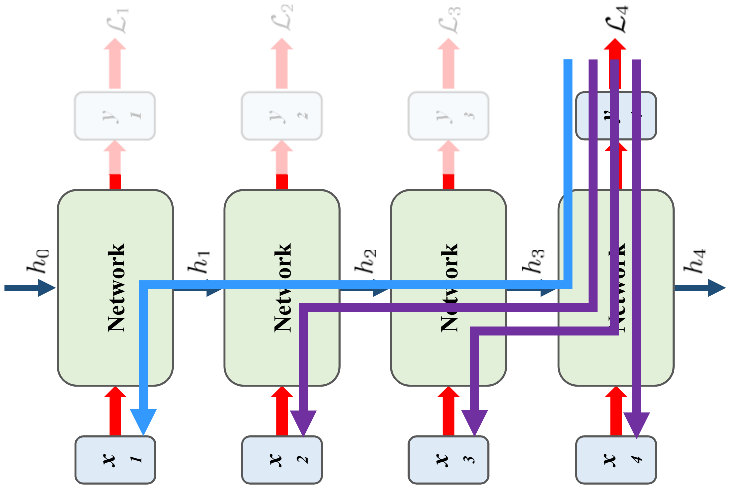
Truncated backpropagation through time



Issues of RNN

- **Vanishing gradient**
  - Gradient may become very small

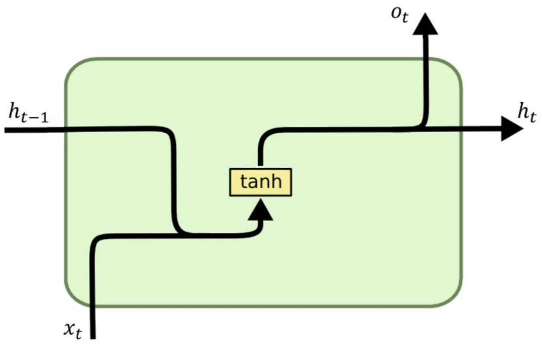
- **Exploding gradient**
  - Gradient may become very large



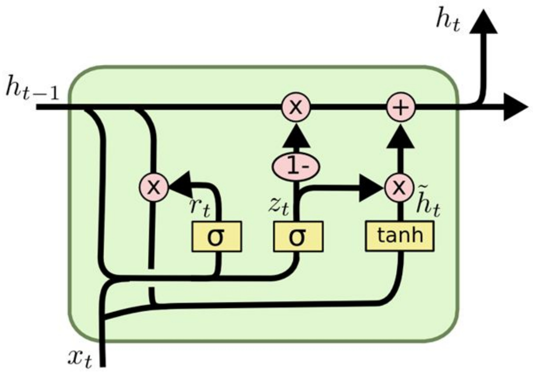
$$\frac{\partial \mathcal{L}}{\partial h_0} = \frac{\partial \mathcal{L}}{\partial h_3} \frac{\partial h_3}{\partial h_2} \frac{\partial h_2}{\partial h_1} \frac{\partial h_1}{\partial h_0}$$

Both make it difficult to learn long term dependencies

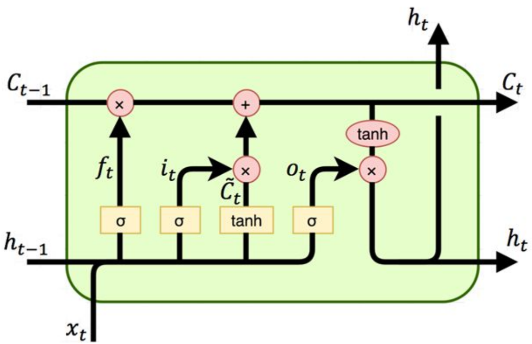
Advanced unit cells



Vanilla RNN



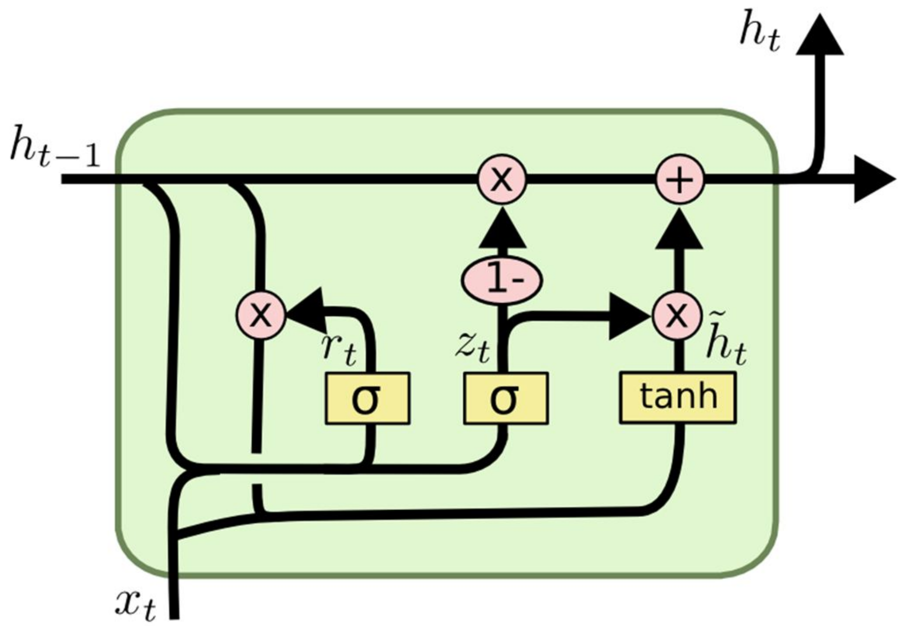
Gated Recurrent Unit



Long-Short Term Memory

- **Advanced unit cells have been developed to overcome the limitations of RNN**

Unit cell: Gated Rucurrent Unit (GRU)

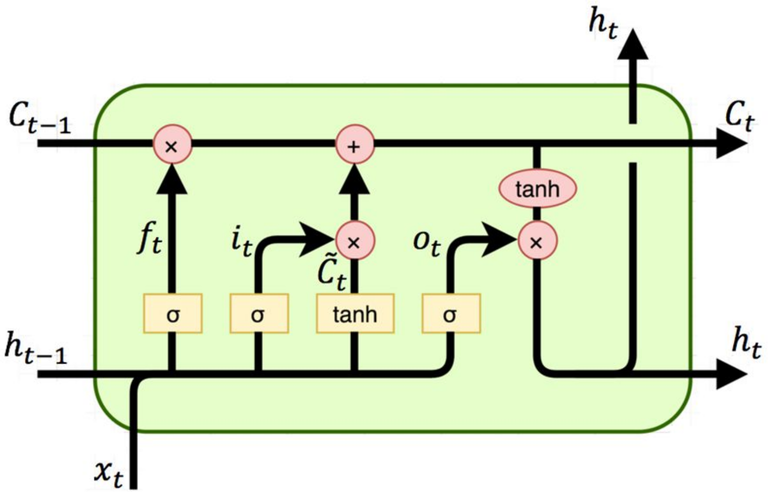


$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z)$$
$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r)$$
$$\hat{h}_t = \phi_h(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h)$$
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h}_t$$

- x_t: input vector
- h_t: output vector
- $\hat{h}_t$: candidate activation vector
- z_t: update gate vector
- r_t: reset gate vector
- W, U and b: parameter matrices and vector

https://en.wikipedia.org/wiki/Gated_recurrent_unit

Unit cell: LSTM



$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f)$$
$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i)$$
$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o)$$
$$\tilde{c}_t = \tanh_c(W_c x_t + U_c h_{t-1} + b_c)$$
$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$$
$$h_t = o_t \circ \sigma_h(c_t)$$

- $x_t \in \mathbb{R}^d$: input vector to the LSTM unit
- $f_t \in \mathbb{R}^h$: forget gate's activation vector
- $i_t \in \mathbb{R}^h$: input/update gate's activation vector
- $o_t \in \mathbb{R}^h$: output gate's activation vector
- $h_t \in \mathbb{R}^h$: hidden state vector also known as output vector of the LSTM unit
- $\tilde{c}_t \in \mathbb{R}^h$: cell input activation vector
- $c_t \in \mathbb{R}^h$: cell state vector
- $W \in \mathbb{R}^{h \times d}$, $U \in \mathbb{R}^{h \times h}$ and $b \in \mathbb{R}^h$: weight matrices and bias vector parameters which need to be learned during training

https://en.wikipedia.org/wiki/Long_short-term_memory

Summary

- Sequence prediction problem
- Recurrent neural network
- Training RNN
  - Backpropagation through time (BPTT)
  - Truncated BPTT
- Learning long-term dependency is difficult with vanilla RNN (gradient vanishing & gradient explosion)
- Advanced unit cells
  - Gated Recurrent Unit (GRU)
  - Long Short Term Memory (LSTM)

References

- Website
  - CS231n RNN lecture note: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf