

1. (30%) Given an LSTM model as shown in Figure 1, point out (i.e., write down the symbols) which is the input gate, output gate, and forget gate, respectively (10%). Then explain the function of each gate (10%). Suppose that we have an application that needs to predict **an output  $y$**  for a sequence of **three inputs ( $x_1, x_2, x_3$ )** (i.e., three time steps); draw an unfolded figure of LSTM for this application. (10%)

$f_t$ : forget gate, for controlling the amount of information of the cell state “flow in the current state” from the previous state

$i_t$ : input gate, for controlling the amount of the contribution to the current cell state by the current input.

$o_t$ : output gate, for controlling the amount of state information of the cell state to hidden state.

$a_t$ : used to be called gate gate, which is a  $\tanh$  activation function to suppress the contribution of the input between -1 and +1.

So totally, we have four different gates and their corresponding parameter matrices.

For the described application, the layout of the LSTMs are as follows:

$f_t$ : 忘記門，用於控制單元狀態“當前狀態”與前一個狀態的資訊量

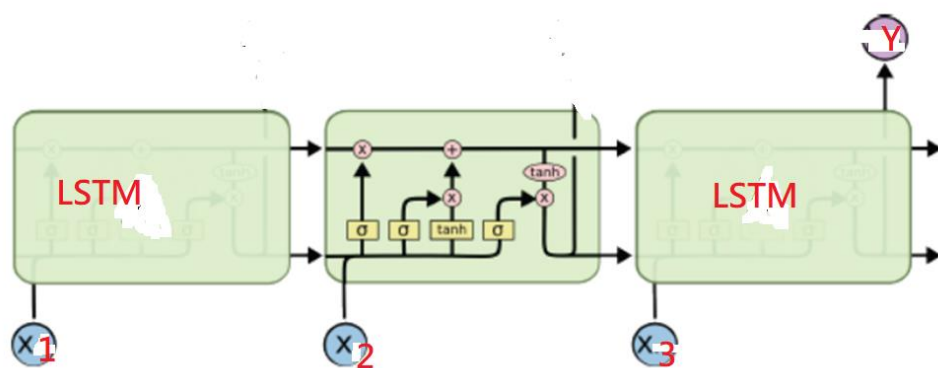
$i_t$ : 輸入門，用於控制當前輸入對當前單元狀態的貢獻量。

$o_t$ : 輸出門，用於控制單元狀態到隱藏狀態的狀態資訊量。

$a_t$ : 以前稱為柵極門，它是一種  $\tanh$  啟動函數，用於抑制 -1 和 +1 之間的輸入貢獻。

因此，我們總共有四個不同的門及其相應的參數矩陣。

對於所描述的應用程式，LSTM 的佈局如下：



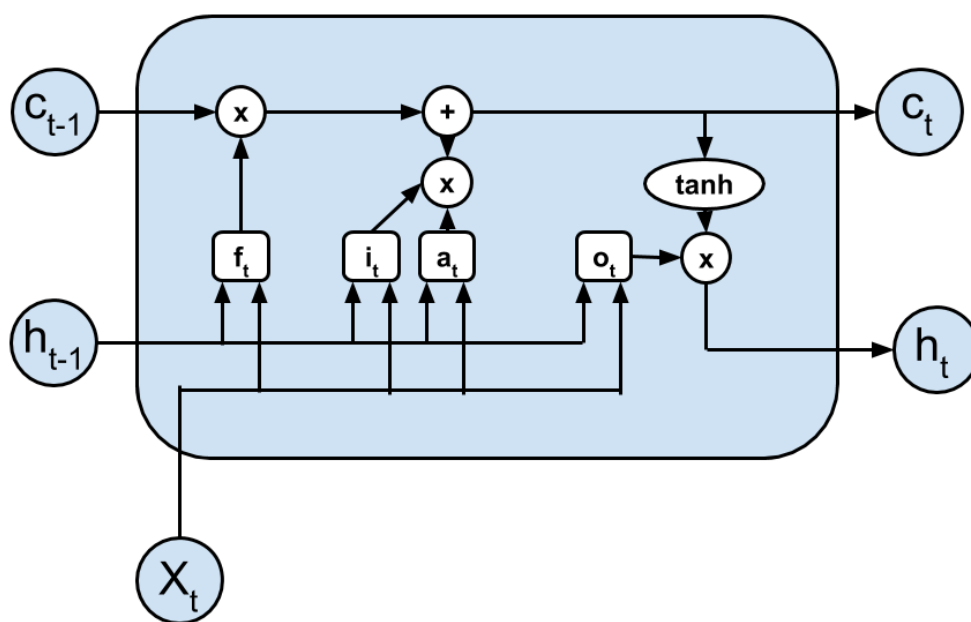


Figure 1. LSTM

2. (10%) What is the problem with a simple (or vanilla) RNN? How to solve it?

The answer is (答案是:): It may cause the gradient vanishing or gradient exploding problems. Use "gradient clipping" to solve the gradient exploding problem and use LSTM or GRU to solve the gradient vanishing problem. 它可能會導致梯度消失或分級爆炸問題。使用「梯度裁剪」求解梯度爆炸問題，使用 LSTM 或 GRU 求解梯度消失問題。

3. (10%) (下面兩個圖中，何者有使用 Attention---The answer is b) The following diagrams show a sequence-to-sequence translation application of an RNN with or without using attention. Point out which (Fig.2(a) or Fig.2(b)) is the RNN with attention model and which is not. Explain what is the difference between them.

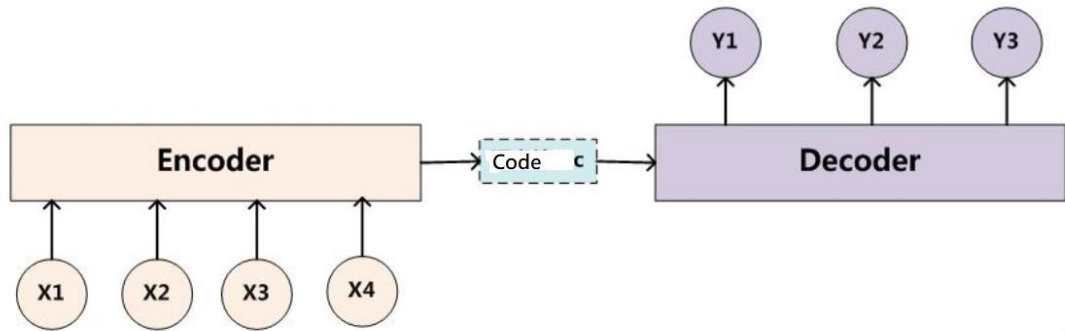
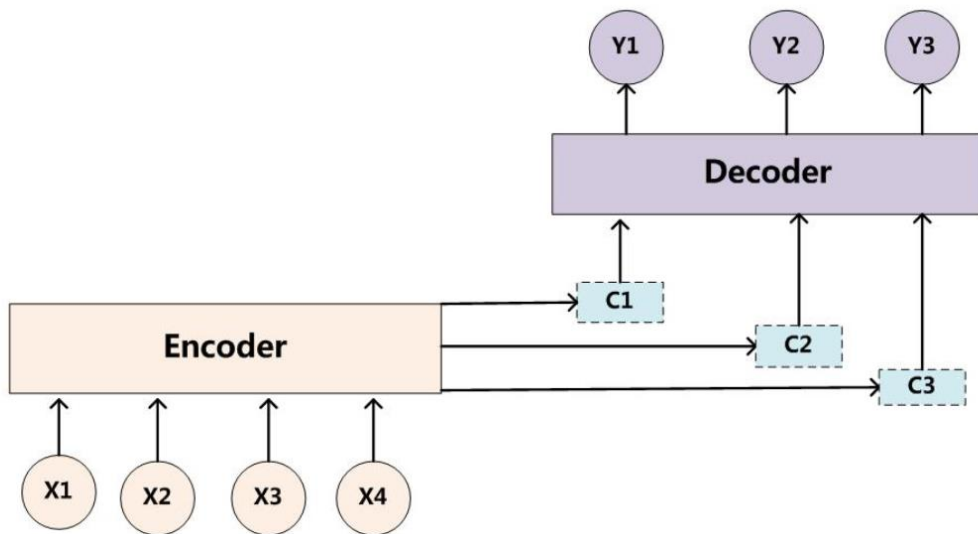


Fig. 2(a)



✓

Fig. 2(b)

4. (20%) (RNN 計算範例) A simple RNN with an initial hidden state of  $h_0=[0.0, 0.0]$ ,

$U=[0.5,0.6]$ ,  $V=[1.0,2.0]$ , Hidden layer **bias**=[0.1,-0.1], Output bias=[0.1]

$W = [0.1,0.2]$

$[0.3,0.4]$

**Given the input vector [2, 3], calculates the corresponding output of y.**

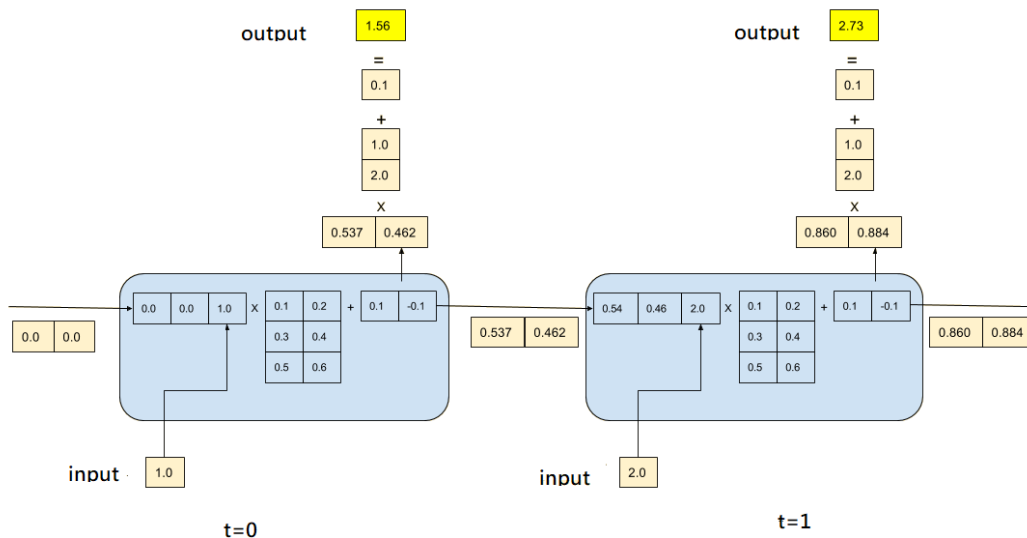
*For your reference, note that the activation function  $f$  is  $\tanh()$ .*

$$h_1 = f(U \cdot x_1 + W \cdot h_0)$$

$$y_1 = g(V \cdot h_1)$$

An example of calculation is shown in the following figure.

The answers are: (答案是:) 2.5, 3.0



5. (20%) The following piece of code defines a deep LSTM model for a time series prediction. It uses three time steps of historical time series values to predict the value of the next time step.

```

1 # reshape from [samples, timesteps] into [samples, timesteps, features]
2 n_features = 1
3 X = X.reshape((X.shape[0], X.shape[1], n_features))
4 # define model
5 print("n_steps", n_steps, "n_features", n_features)
6 model = Sequential()
7 model.add(LSTM(30, activation='relu', return_sequences=True, \
8               input_shape=(n_steps, n_features)))
9 model.add(LSTM(30, activation='relu'))
10 model.add(Dense(1))
11 model.summary()

```

n\_steps 3 n\_features 1

Answer the following questions:

- How many parameters are there in the first LSTM layer?
- How many parameters are there in the second LSTM layer?

*Hint: The input of the first layer is just a real number, and the input of the second layer is the hidden vector of the first layer, which has a dimension of 30.*

The answers are(答案是:) (a) 3840 (b) 7320

Figure 1 illustrates the architecture of the proposed network. The main path consists of a sequence of residual network layers (R1, R2, R3, R4, R5) and a final residual block (R6). A skip connection (S1) connects the output of the first residual block (R1) to the input of the first decoder block (D1). The decoder path consists of a sequence of residual network layers (D1, D2, D3, D4). The final output is the sum of the output of the last decoder block (D4) and the output of the last encoder block (R5). The legend defines the symbols: a green box for 'Residual Network Layer', a circle with 'S' for 'Skip Connection', a circle with 'V' for 'Vector Branch', a circle with 'C' for 'Concatenation', and a circle with 'C' for 'Copy'.

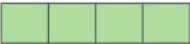
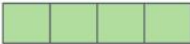



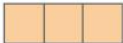
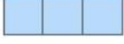
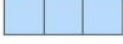
$$\begin{aligned} f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\ i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ \tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \\ C_t &= f_t * C_{t-1} + i_t * \tilde{C}_t \\ o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\ h_t &= o_t * \tanh(C_t) \end{aligned}$$

total number of  
parameters:

Note that  $x$  in the 2nd LSTM is the hidden state of the 1st LSTM. It has a dimension of 30.

6. (10%) Transformer is the foundation of the many recent large pre-trained language models, such as BERT and ChatGPT. The concept of self-attention is the core of the Transformer. Based on the following figure, please explain the relationships among query, key, value and the resulting representation  $Z$ . Please answer this question by considering only one-head attention. *Specifically, please answer how  $Z$ 's first row is*

derived from  $v_1$  and  $v_2$ .

Input	Thinking	Machines
Embedding	$x_1$ 	$x_2$ 
Queries	$q_1$ 	$q_2$ 
Keys	$k_1$ 	$k_2$ 
Values	$v_1$ 	$v_2$ 
Score	$q_1 \cdot k_1 = 112$	$q_2 \cdot k_2 = 96$
Divide by 8 ( $\sqrt{d_k}$ )	14	12
Softmax	0.88	0.12

The answer is :  $Z = 0.88 * v_1 + 0.12 * v_2$

