

# 程式人《十分鐘系列》



## 深度學習的 RNN/LSTM 循環神經網路

( 使用 node.js 的 neataptic 套件實作 )

陳鍾誠

2017 年 7 月 6 日

# 前天、我們講解了反傳遞演算法

程式人《十分鐘系列》



專為程式人寫的神經網路導論

(以反傳遞演算法為入門磚)

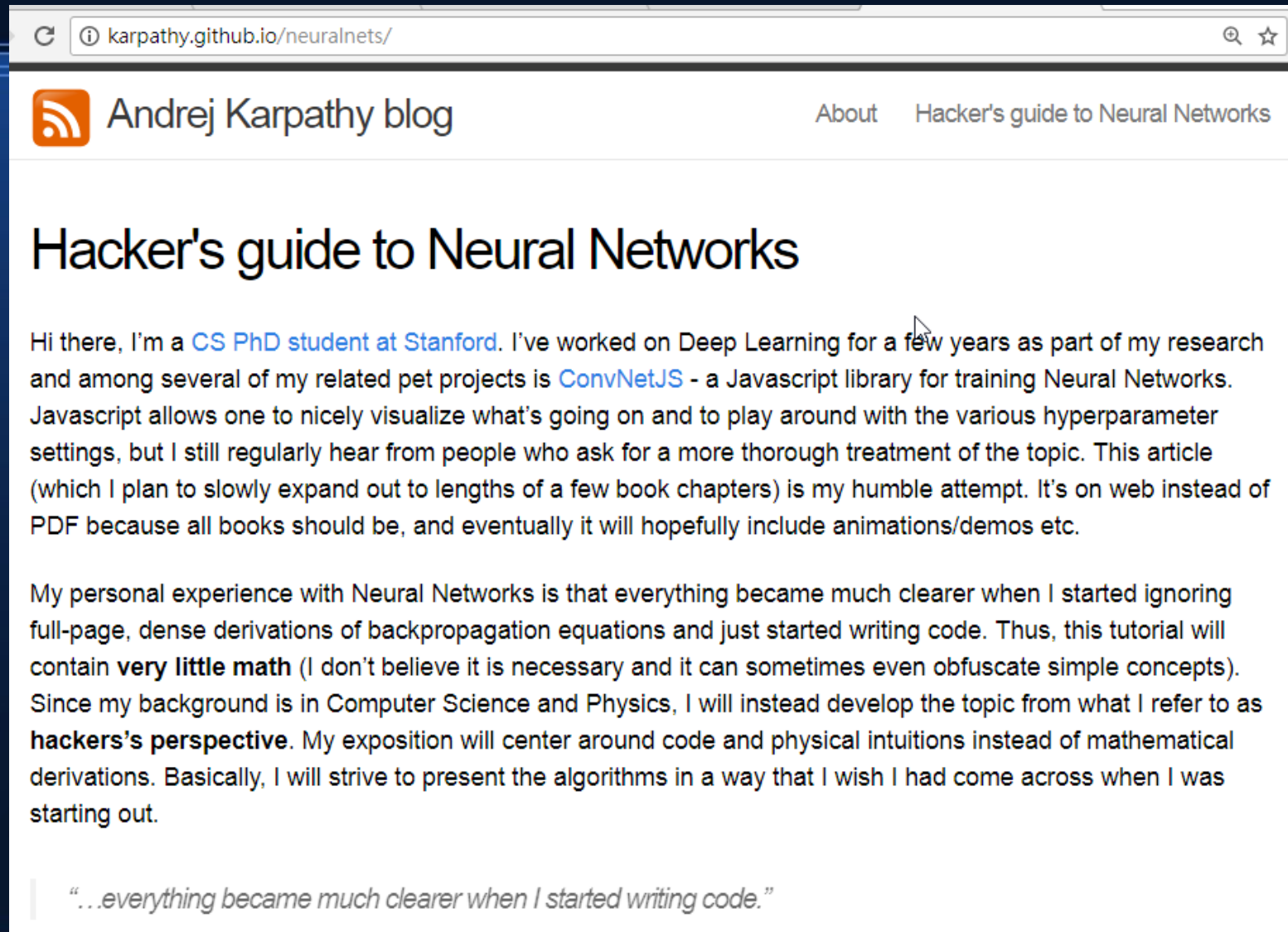
陳鍾誠

2017 年 7 月 4 日

# 那篇十分鐘系列

- 是根據 Andrej Karpathy 的  
Hacker's guide to Neural Networks  
這篇文章所改寫的！

# 就是這篇



The image is a screenshot of a web browser displaying the blog of Andrej Karpathy. The browser's address bar shows the URL `karpathy.github.io/neuralnets/`. The page header includes the blog's name, "Andrej Karpathy blog", and navigation links for "About" and "Hacker's guide to Neural Networks". The main heading of the article is "Hacker's guide to Neural Networks". The text of the article begins with a greeting and an introduction to the author's background as a CS PhD student at Stanford, followed by a discussion of his research in Deep Learning and his pet project, ConvNetJS. He explains that while Javascript allows for visualization and experimentation with hyperparameters, he still receives requests for a more thorough treatment of the topic. This article, he states, is his humble attempt, which he plans to expand into a book. He notes that the article is on the web instead of PDF to allow for future updates like animations and demos. The second paragraph describes his personal experience with Neural Networks, stating that everything became clearer when he started ignoring complex mathematical derivations and focusing on writing code. He promises that the tutorial will contain very little math and will be developed from a hacker's perspective, centered around code and physical intuitions rather than mathematical derivations. At the bottom, there is a quote: "...everything became much clearer when I started writing code."

karpathy.github.io/neuralnets/

Andrej Karpathy blog

About Hacker's guide to Neural Networks

## Hacker's guide to Neural Networks

Hi there, I'm a [CS PhD student at Stanford](#). I've worked on Deep Learning for a few years as part of my research and among several of my related pet projects is [ConvNetJS](#) - a Javascript library for training Neural Networks. Javascript allows one to nicely visualize what's going on and to play around with the various hyperparameter settings, but I still regularly hear from people who ask for a more thorough treatment of the topic. This article (which I plan to slowly expand out to lengths of a few book chapters) is my humble attempt. It's on web instead of PDF because all books should be, and eventually it will hopefully include animations/demos etc.

My personal experience with Neural Networks is that everything became much clearer when I started ignoring full-page, dense derivations of backpropagation equations and just started writing code. Thus, this tutorial will contain **very little math** (I don't believe it is necessary and it can sometimes even obfuscate simple concepts). Since my background is in Computer Science and Physics, I will instead develop the topic from what I refer to as **hackers's perspective**. My exposition will center around code and physical intuitions instead of mathematical derivations. Basically, I will strive to present the algorithms in a way that I wish I had come across when I was starting out.

*"...everything became much clearer when I started writing code."*

# 但是

- 那篇是講解《基礎的神經網路》  
的運作與訓練方法！

# 而我真正的目標

- 是要用深度學習的循環神經網路，來實作機器翻譯系統。

# Andrej Karpathy

- 不愧是深度學習技術專家

- 他的另一篇文章




The Unreasonable Effectiveness of Recurrent Neural Networks


- 就是在介紹《循環神經網路》！



# 今天

- 我們就來介紹這篇文章！

 [karpathy.github.io/2015/05/21/rnn-effectiveness/](https://karpathy.github.io/2015/05/21/rnn-effectiveness/)  

 Andrej Karpathy blog About Hacker's guide to Neural Networks

## The Unreasonable Effectiveness of Recurrent Neural Networks

May 21, 2015

There's something magical about Recurrent Neural Networks (RNNs). I still remember when I trained my first recurrent network for [Image Captioning](#). Within a few dozen minutes of training my first baby model (with rather arbitrarily-chosen hyperparameters) started to generate very nice looking descriptions of images that were on the edge of making sense. Sometimes the ratio of how simple your model is to the quality of the results you get out of it blows past your expectations, and this was one of those times. What made this result so shocking at the time was that the common wisdom was that RNNs were supposed to be difficult to train (with more experience I've in fact reached the opposite conclusion). Fast forward about a year: I'm training RNNs all the time and I've witnessed their power and robustness many times, and yet their magical outputs still find ways of amusing me. This post is about sharing some of that magic with you.

*We'll train RNNs to generate text character by character and ponder the question "how is that even possible?"*



# 問題是

- 到底甚麼是《循環神經網路》呢？

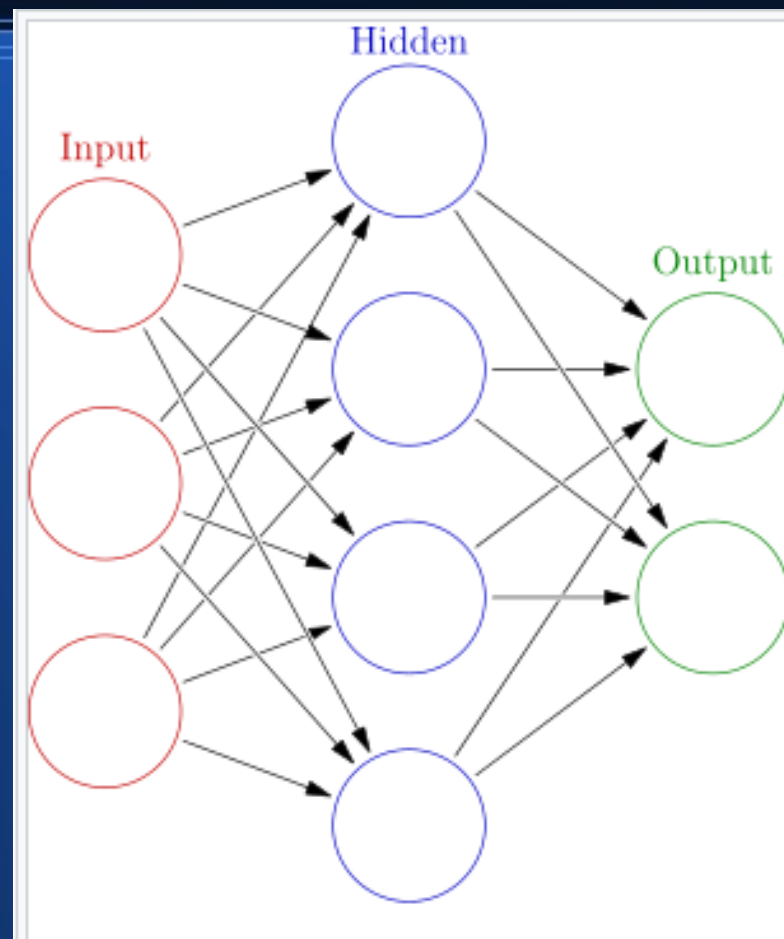
# 這個問題

- 先讓我們反過來問？

甚麼是《非循環神經網路》呢？

# 下圖就是個非循環神經網路

- 很清楚的  
這個網路裡  
沒有循環！



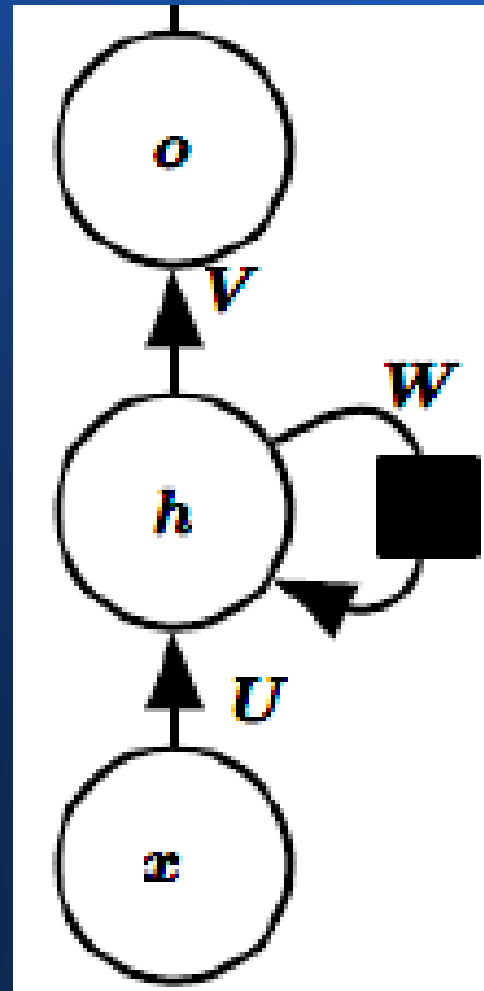
An artificial neural network is an interconnected group of nodes, akin to the vast network of neurons in a brain. Here, each circular node represents an artificial neuron and an arrow represents a connection from the output of one neuron to the input of another.

那麼、什麼是循環神經網路呢？

# 顧名思義

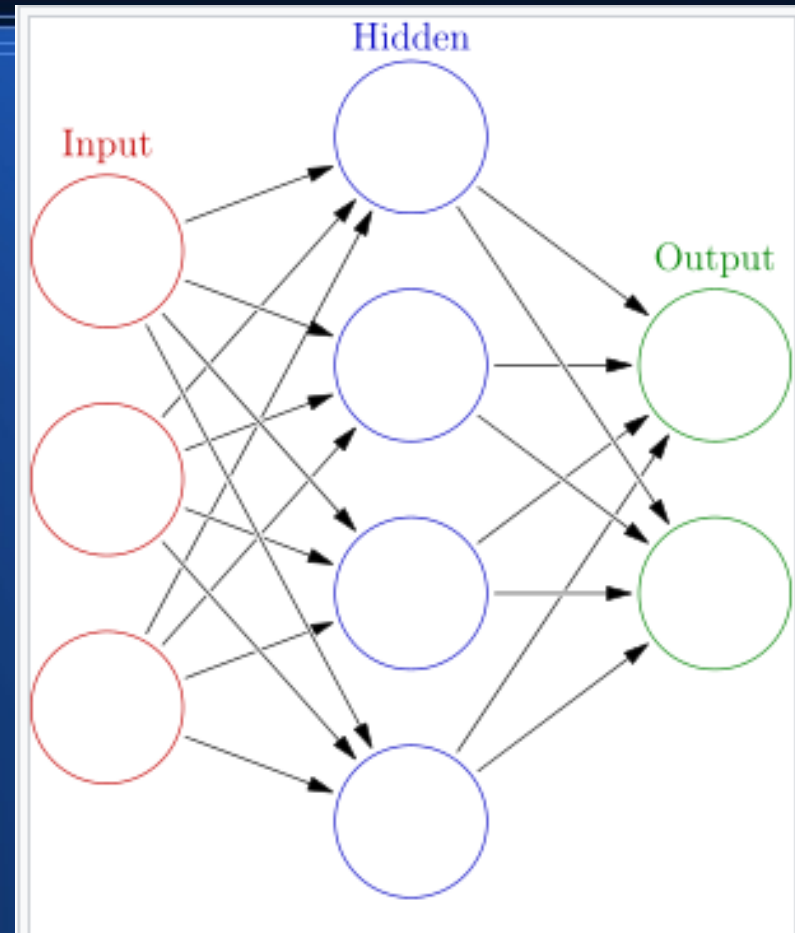
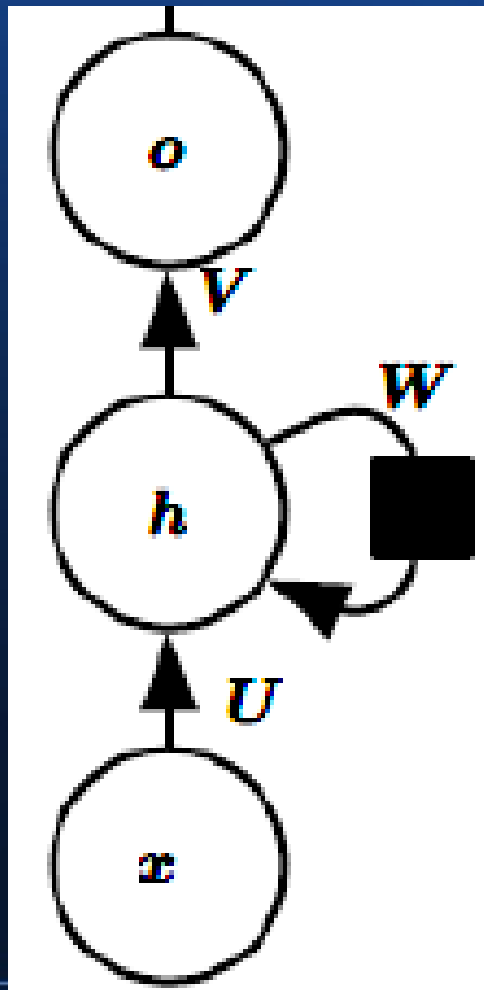
- 就是有循環的神經網路！

像這樣





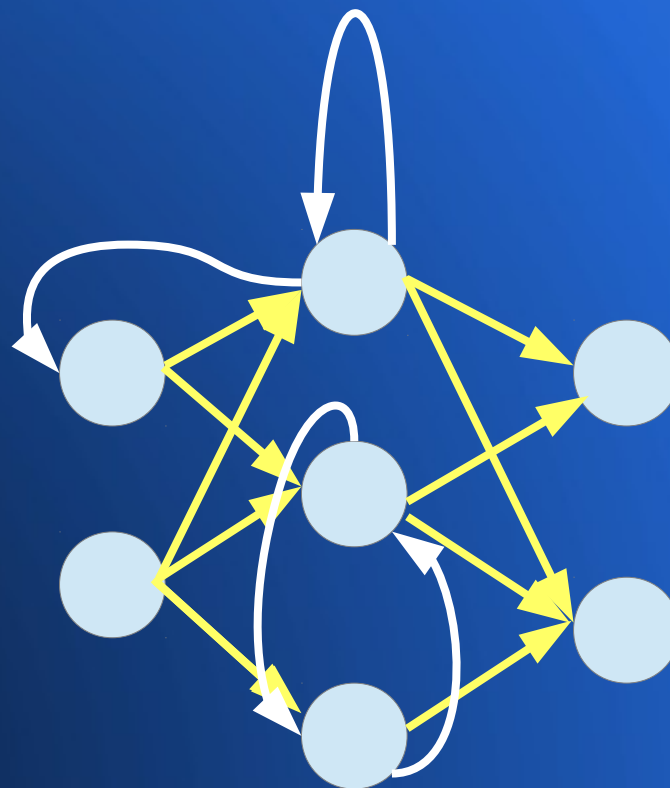
# 比較一下



An artificial neural network is an interconnected group of nodes, akin to the vast network of **neurons** in a **brain**. Here, each circular node represents an artificial neuron and an arrow represents a connection from the output of one neuron to the input of another.

# 其實只要拉幾條倒鉤的線

神經網路就會產生循環



但是

- 為何需要有循環呢？

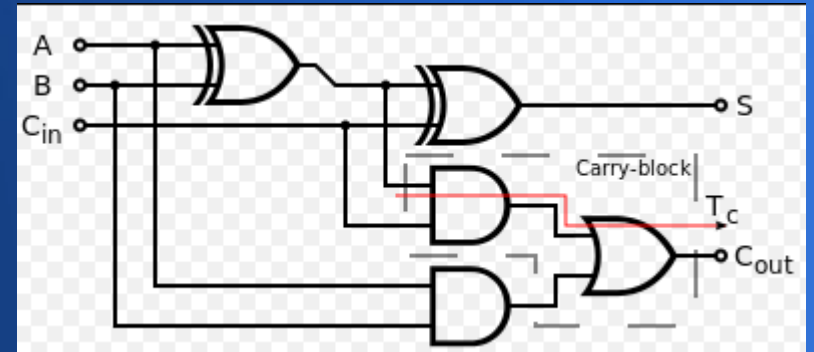
有循環的網路有何好處呢？

# 如果您曾經學過數位電路

- 應該會記得《組合邏輯電路》  
和《循序邏輯電路》

# 其中的組合邏輯電路

- 像是 AND, OR, NOT, XOR, MUX 多工器, DMUX 解多工器, HalfAdder 半加器, FullAdder 全加器, 16 位元加法器等等。
- 都是沒有循環的電路！



# 這類電路

- 只要輸入相同
- 輸出一定是相同的！



# 因為電路裏

- 不會偷偷的暫存一些東西
- 沒有任何記憶元件！

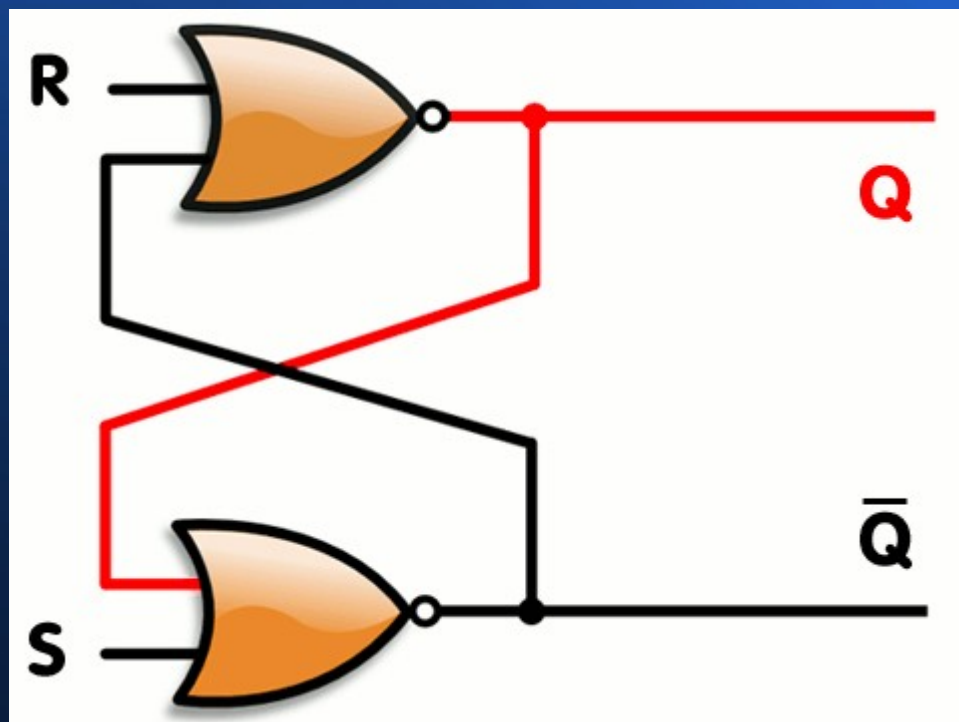
# 所以執行結果

- 不會有記憶所造成的效應
- 於是相同的輸入，就會有相同的輸出！

# 但是有一類數位電路

- 稱為循序邏輯電路
- 這種電路就有循環！

像是正反器就是典型有循環的電路



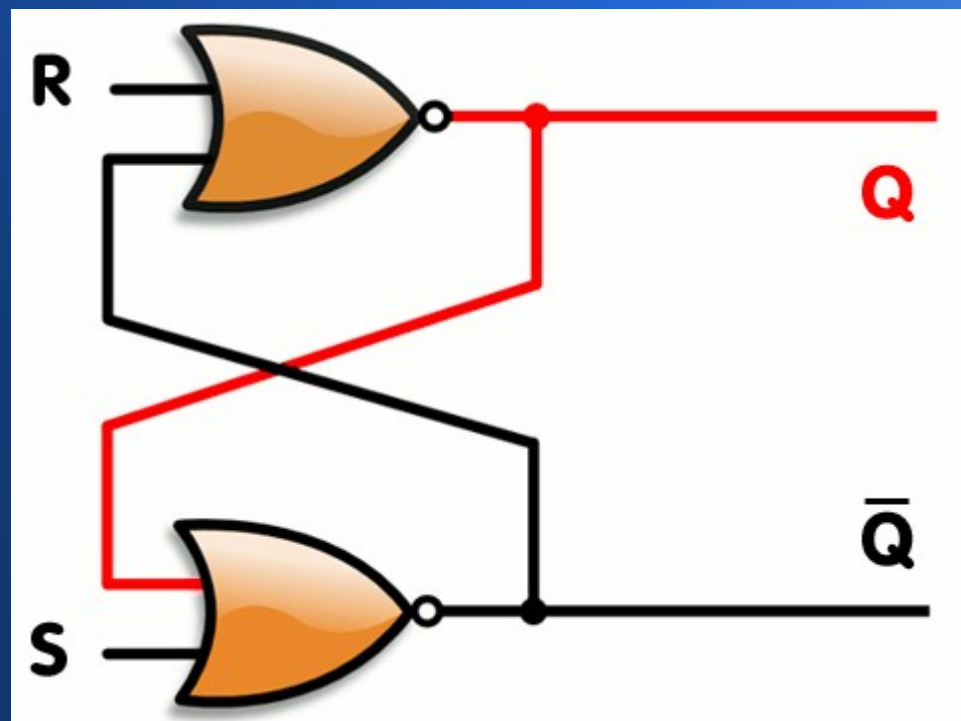
# 這些循環

- 可以用來把訊號鎖死在電路裏，造成記憶效應！

SR門鎖運算<sup>[2]</sup>

狀態轉移表				激發表			
S	R	$Q_{next}$	動作	Q	$Q_{next}$	S	R
0	0	Q	保持	0	0	0	X
0	1	0	重設	0	1	1	0
1	0	1	設置	1	0	0	1
1	1	X	不允許的輸入	1	1	X	0

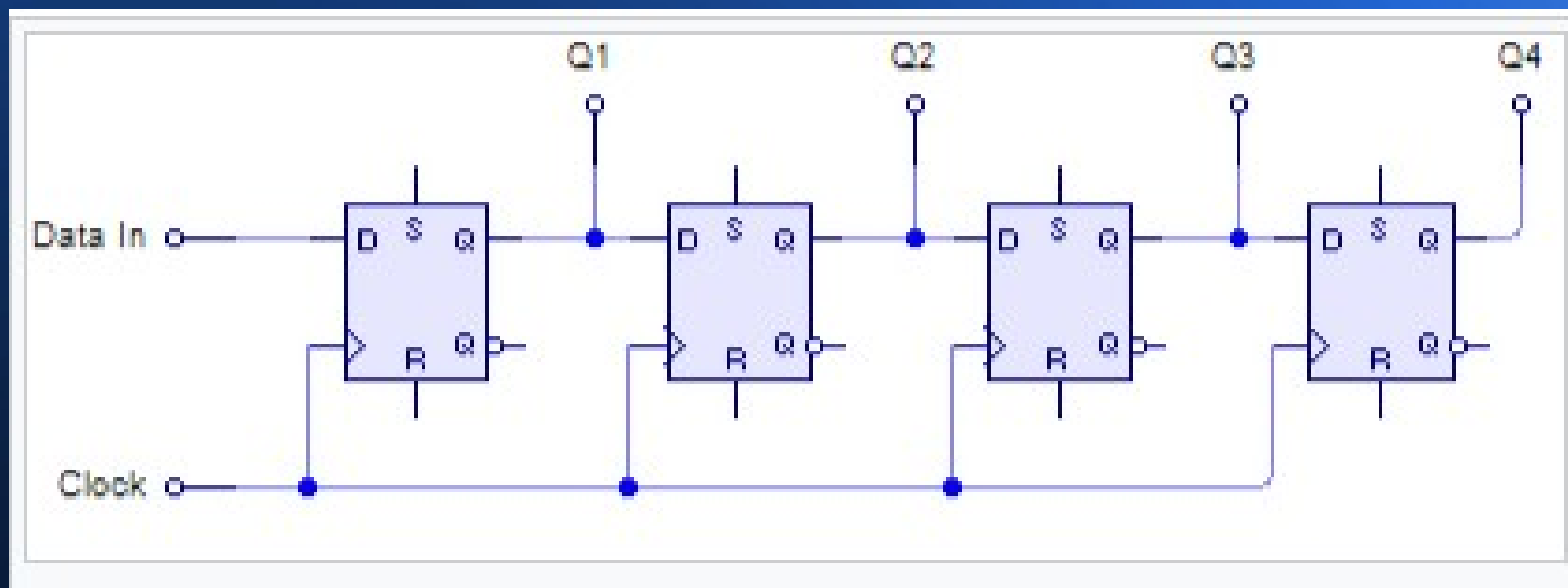
特徵方程為  $Q_{next} = S + \bar{R}Q$ ，且  $RS=0$ 。



# 因此

- 電路的輸出除了受輸入影響之外，還會受目前的（記憶）狀態所影響

暫存器就是用很多正反器做出來的！





記憶體也是類似

# 而循環神經網路的好處

- 就是可以記憶一些歷史訊息
- 並透過歷史訊息預測未來！

# 非循環神經網路

- 輸出完全由輸入決定
- 可以寫成  $y=f(x)$
- 或者寫成  $s_t=f(\theta)$

# 但是循環神經網路

- 數學式通常寫為

$$-s_t = f(s_{t-1}, \theta)$$

- 而不是  $s_t = f(\theta)$

$s_t = f(s_{t-1}, \theta)$  中的  $s_t$

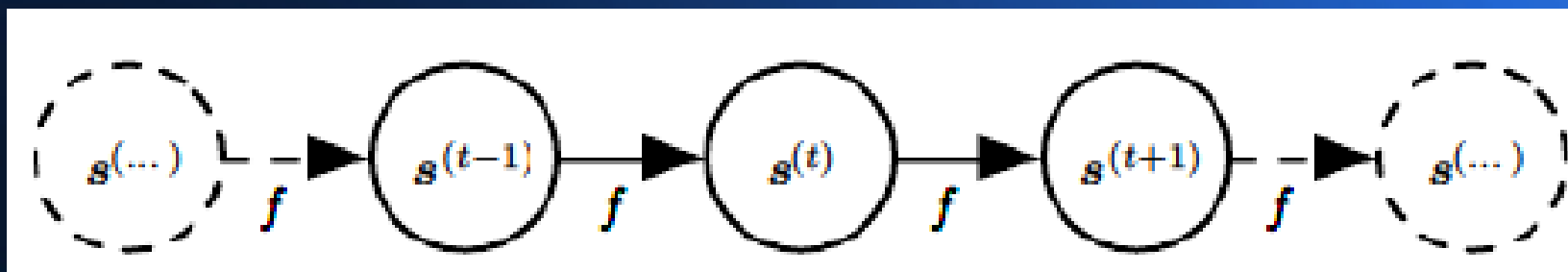
- 就代表了  $t$  時刻的記憶狀態

# 透過這種記憶狀態 $s_t$

歷史資訊就能影響後續的決策

$$s^{(t)} = f(s^{(t-1)}; \theta)$$

$$\begin{aligned} s^{(3)} &= f(s^{(2)}; \theta) \\ &= f(f(s^{(1)}; \theta); \theta) \end{aligned}$$



# 如果加入外部輸入 $\mathbf{x}$

- 循環神經網路的函數就會改寫如下

$$\mathbf{s}^{(t)} = f(\mathbf{s}^{(t-1)}, \mathbf{x}^{(t)}; \boldsymbol{\theta})$$



我們可以用  $h$  取代  $s$   
代表非外部輸入的隱含狀態

$$s^{(t)} = f(s^{(t-1)}, x^{(t)}; \theta)$$



$$h^{(t)} = f(h^{(t-1)}, x^{(t)}; \theta)$$

# 根據 $t$ 畫成展開圖就變這樣

黑色框框代表延遲一單位時間

網路歷史展開後的結果

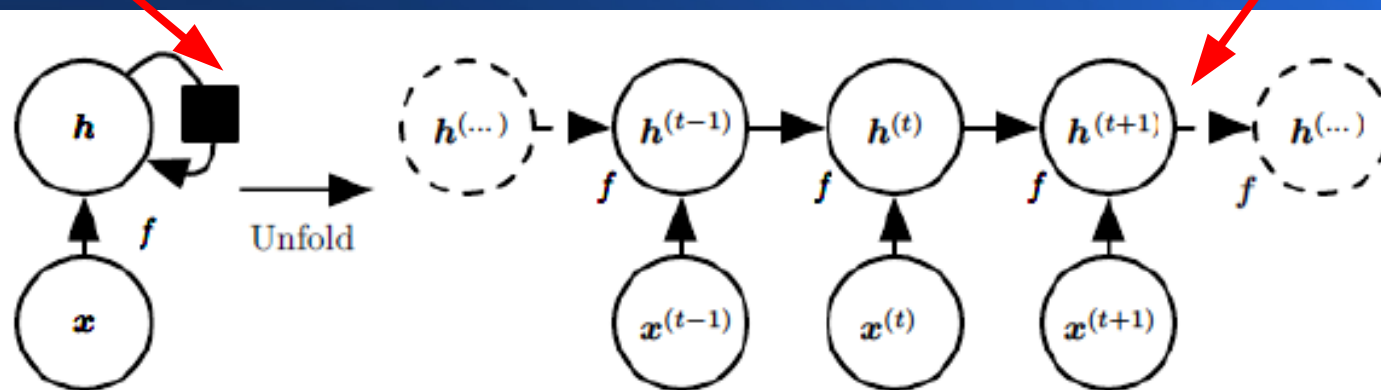
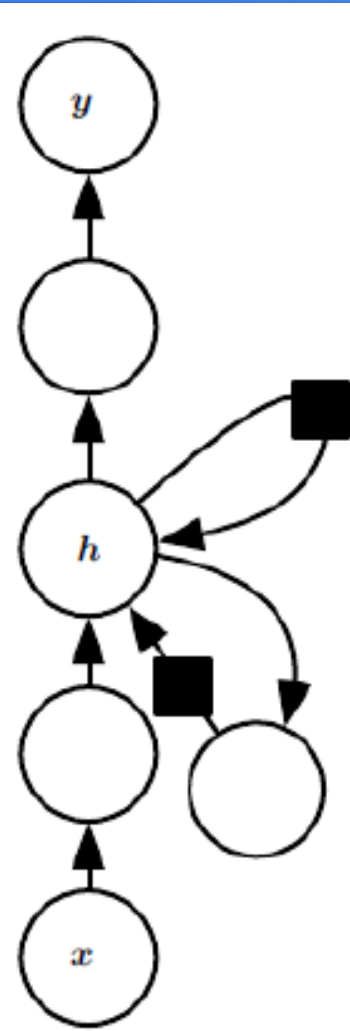
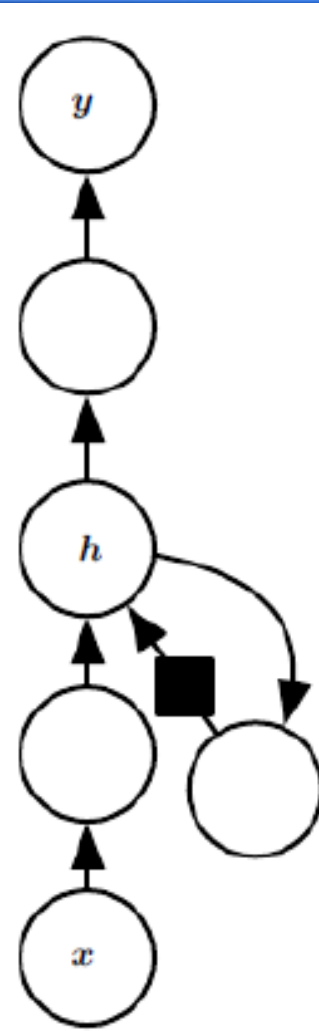
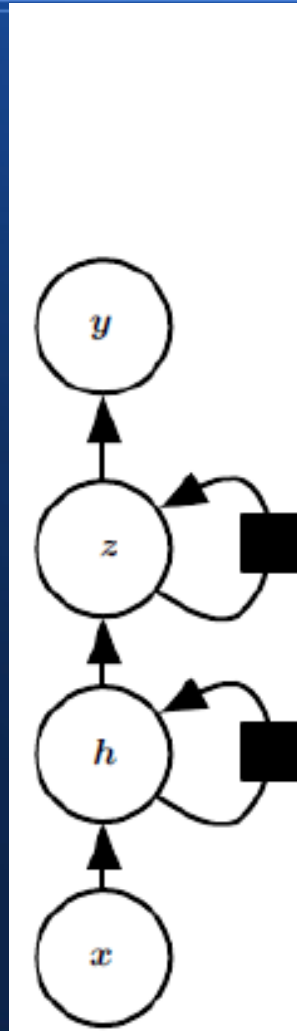
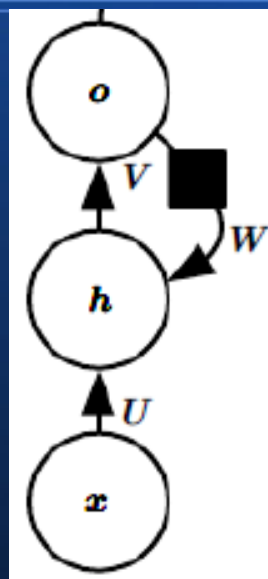
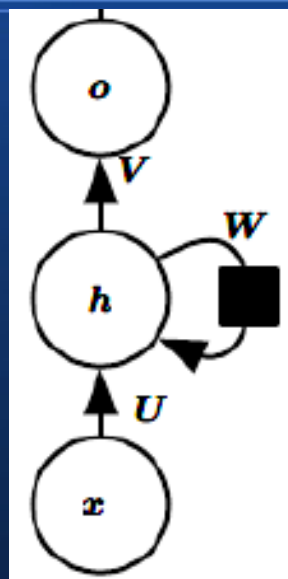


Figure 10.2: A recurrent network with no outputs. This recurrent network just processes information from the input  $\mathbf{x}$  by incorporating it into the state  $\mathbf{h}$  that is passed forward through time. (Left) Circuit diagram. The black square indicates a delay of 1 time step. (Right) The same network seen as an unfolded computational graph, where each node is now associated with one particular time instance.

$$\begin{aligned} \mathbf{h}^{(t)} &= g^{(t)}(\mathbf{x}^{(t)}, \mathbf{x}^{(t-1)}, \mathbf{x}^{(t-2)}, \dots, \mathbf{x}^{(2)}, \mathbf{x}^{(1)}) \\ &= f(\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}; \boldsymbol{\theta}) \end{aligned}$$

# 循環神經網路有很多類

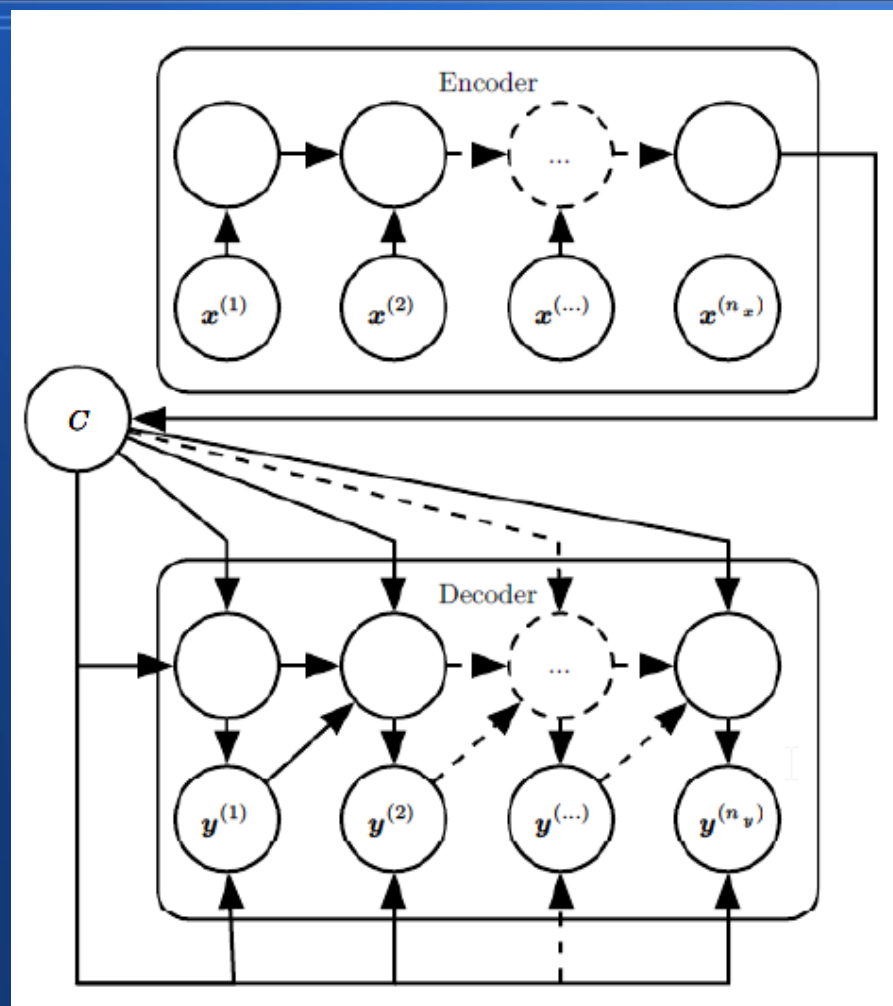


# 其中有一類展開圖長這樣

這種 Encoder/Decoder  
( 編碼 / 解碼 ) 網路

在機器翻譯的領域非常重要！

也是我目前所學習的重點。



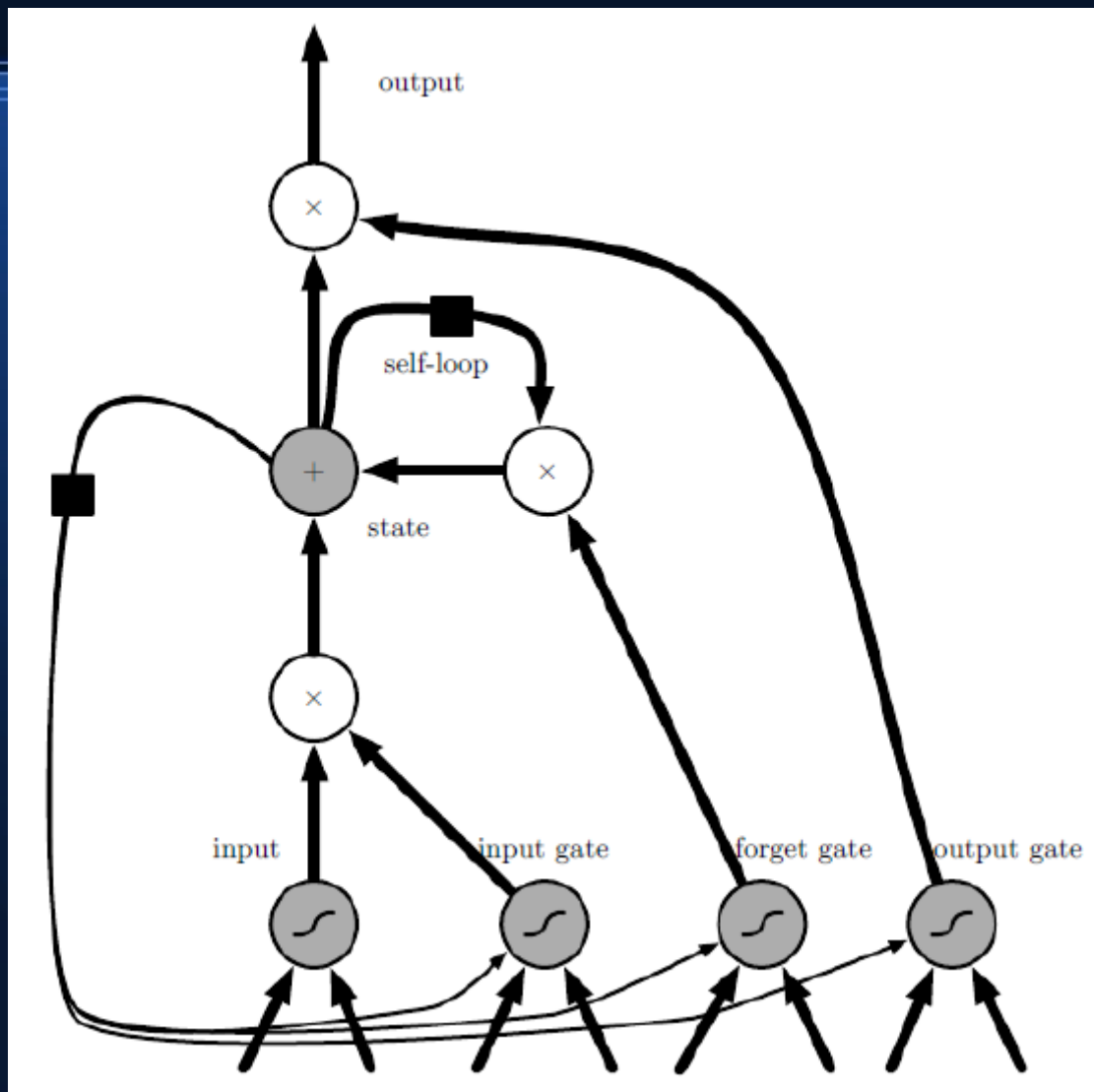
# 但是簡易的循環神經網路

- 通常有個缺點
- 就是沒有長期記憶！

# 為了保住長期記憶

- 有人發明了 LSTM 長短期記憶網路  
(Long-Short Term Memory)
- 以便讓夠強的資訊可以進入長期記憶  
不夠強的資訊可以被遺忘！

# LSTM 長短期記憶網路長這樣



# 其數學式如下

$$s_i^{(t)} = f_i^{(t)} s_i^{(t-1)} + g_i^{(t)} \sigma \left( b_i + \sum_j U_{i,j} x_j^{(t)} + \sum_j W_{i,j} h_j^{(t-1)} \right)$$

$$f_i^{(t)} = \sigma \left( b_i^f + \sum_j U_{i,j}^f x_j^{(t)} + \sum_j W_{i,j}^f h_j^{(t-1)} \right)$$

$$g_i^{(t)} = \sigma \left( b_i^g + \sum_j U_{i,j}^g x_j^{(t)} + \sum_j W_{i,j}^g h_j^{(t-1)} \right)$$



# 現在

- 讓我們先忘了那些惱人的數學式

# 把 RNN 循環網路實作為程式

```
rnn = RNN()
y = rnn.step(x) # x is an input vector, y is the RNN's output vector
```

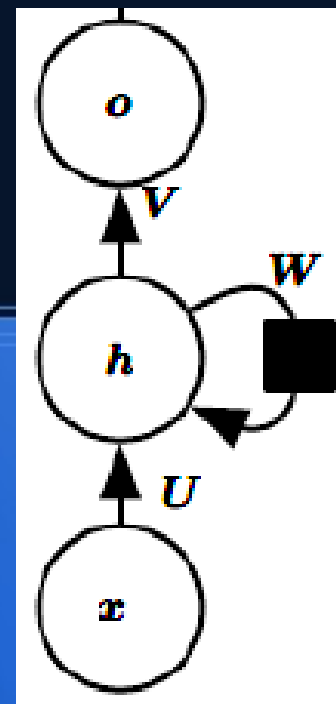
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

```
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
```

對應到上圖的  $V$

對應到上圖的  $W$

對應到上圖的  $U$



# 然後

- 你還可以疊合兩層以上的 RNN

```
y1 = rnn1.step(x)  
y = rnn2.step(y1)
```

接著讓我們想想

這樣的網路到底有甚麼用？

# 其實

- 我也想不出來！

還是看個實例好了

# RNN 可以用來記住序列

- 例如當我們想記住 hello 這個詞
- 但是我們每個時間點只會看到一個字母！



# 於是我們的網路展開圖看來像這樣

目標：記住 hello 這個單字並且能重新產生 hello!

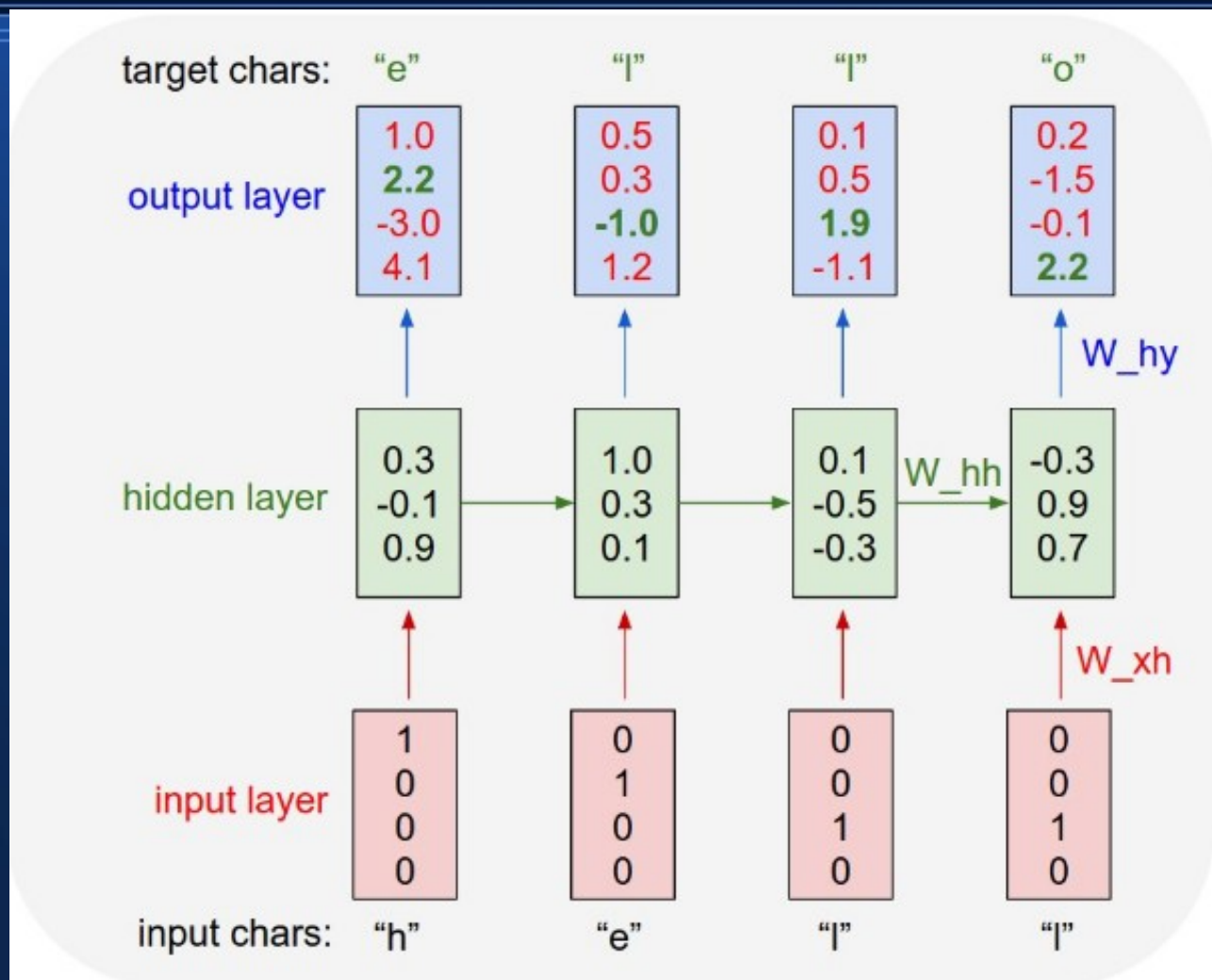
子目標：

輸入 h 時 => 輸出 e

再輸入 e 時 => 輸出 l

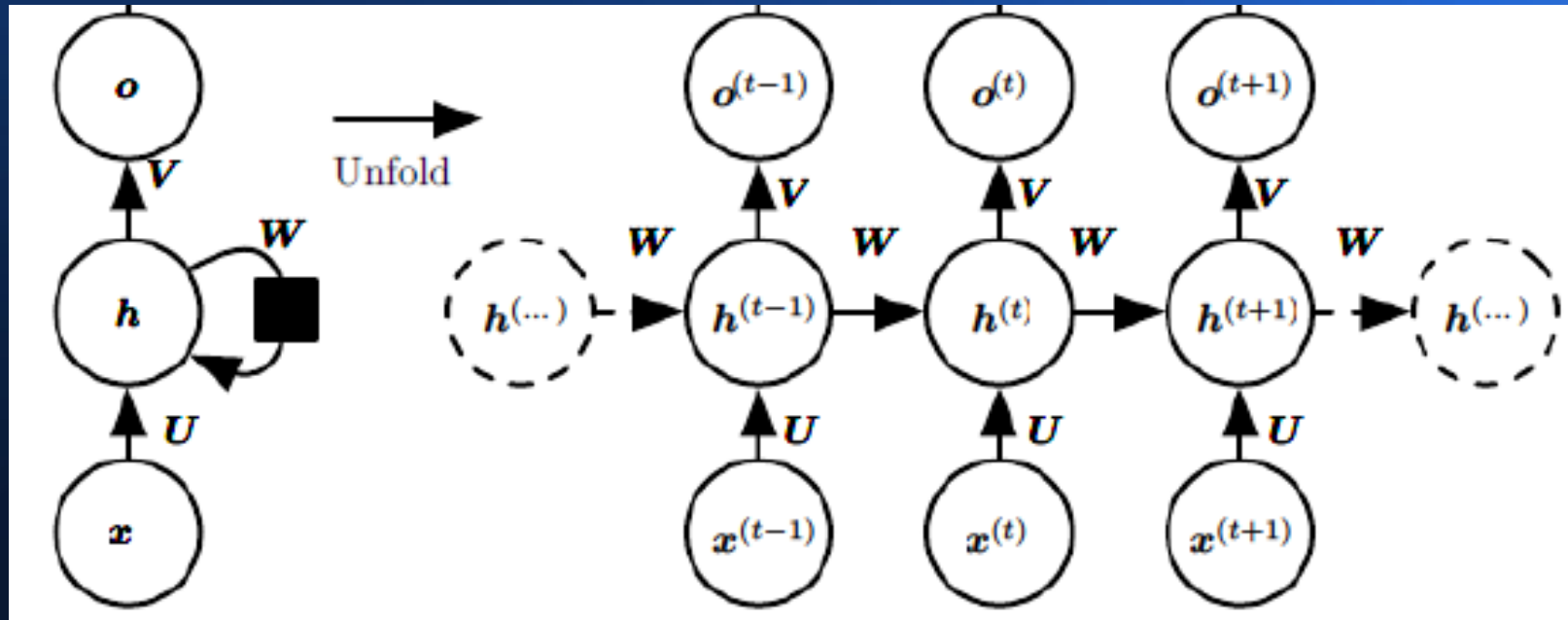
再輸入 l 時 => 輸出 l

再輸入 l 時 => 輸出 o



# RNN 網路展開後

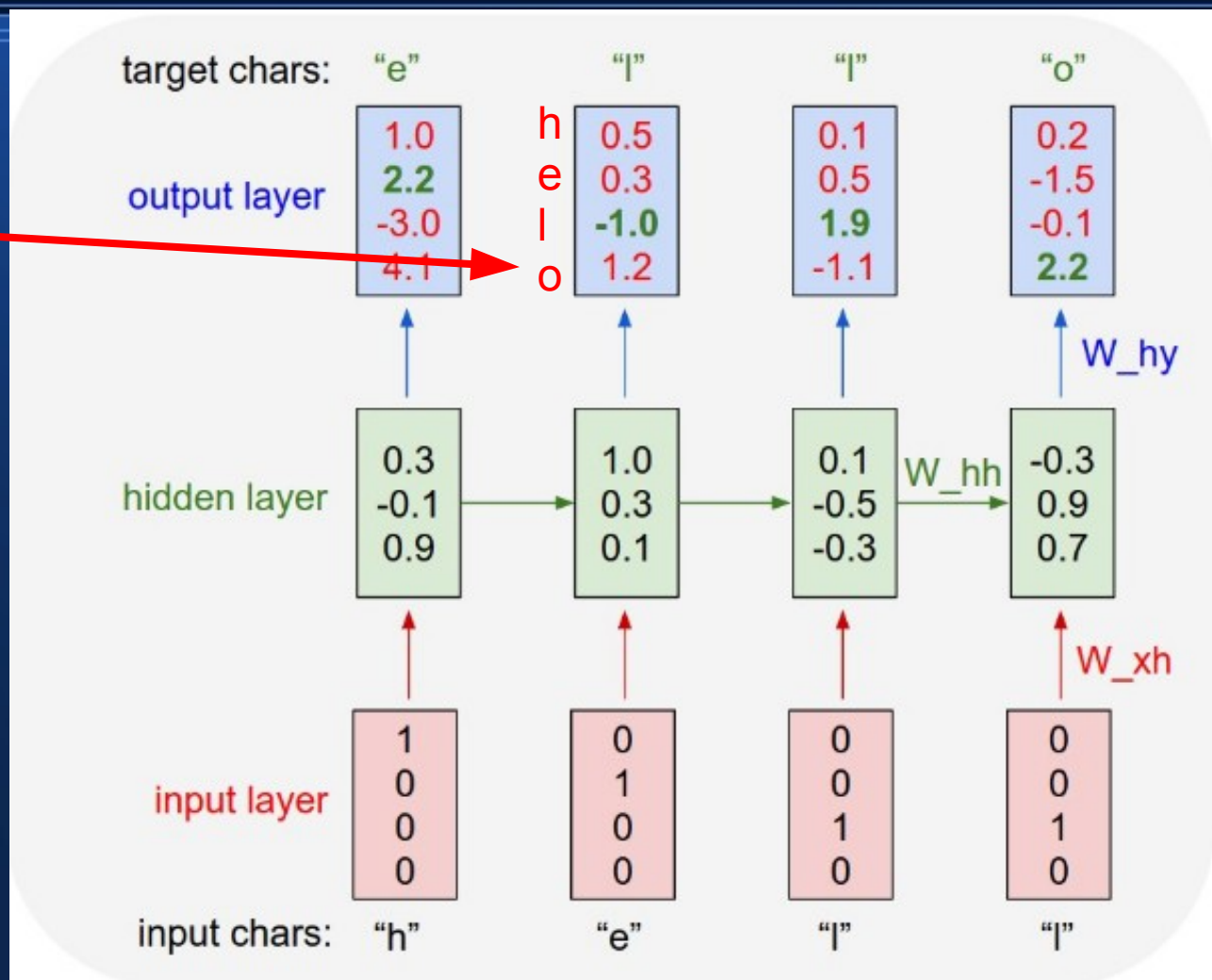
- 通常可以讓我們看得更清楚



但是、這個網路是還沒訓練完畢的

輸入 e 時輸出 o

應該要輸出 l 才對



# 所以循環神經網路

- 和非循環神經網路一樣
- 需要反向傳遞的學習演算法
- 該演算法稱為 BPTT  
(Backpropagation through time)

# 以下是 BPTT 的前向傳遞算法

```
1: for  $t$  from 1 to  $T$  do  
2:    $u_t \leftarrow W_{hv}v_t + W_{hh}h_{t-1} + b_h$   
3:    $h_t \leftarrow e(u_t)$   
4:    $o_t \leftarrow W_{oh}h_t + b_o$   
5:    $z_t \leftarrow g(o_t)$   
6: end for
```



# 然後是 BPTT 的反向傳遞算法

```
1: for  $t$  from  $T$  downto 1 do  
2:    $do_t \leftarrow g'(o_t) \cdot dL(z_t; y_t) / dz_t$   
3:    $db_o \leftarrow db_o + do_t$   
4:    $dW_{oh} \leftarrow dW_{oh} + do_t h_t^\top$   
5:    $dh_t \leftarrow dh_t + W_{oh}^\top do_t$   
6:    $dz_t \leftarrow e'(z_t) \cdot dh_t$   
7:    $dW_{hv} \leftarrow dW_{hv} + dz_t v_t^\top$   
8:    $db_h \leftarrow db_h + dz_t$   
9:    $dW_{hh} \leftarrow dW_{hh} + dz_t h_{t-1}^\top$   
10:   $dh_{t-1} \leftarrow W_{hh}^\top dz_t$   
11: end for  
12: Return  $d\theta = [dW_{hv}, dW_{hh}, dW_{oh}, db_h, db_o, dh_0]$ .
```

# 如果不要看數學就會變這樣

```
Back_Propagation_Through_Time(a, y)  // a[t] is the input at time t. y[t] is the output
  Unfold the network to contain k instances of f
  do until stopping criteria is met:
    x = the zero-magnitude vector; // x is the current context
    for t from 0 to n - k           // t is time. n is the length of the training sequence
      Set the network inputs to x, a[t], a[t+1], ..., a[t+k-1]
      p = forward-propagate the inputs over the whole unfolded network
      e = y[t+k] - p;                // error = target - prediction
      Back-propagate the error, e, back across the whole unfolded network
      Sum the weight changes in the k instances of f together.
      Update all the weights in f and g.
      x = f(x, a[t]);                // compute the context for the next time-step
```

# 背後的數學

- 仍然是微積分
- 還有根據鏈鎖規則的拆解！



# Karpathy 用 Python 寫了一個簡易的 RNN 程式，只有 112 行

```
→ ↻ 🔒 安全 | https://gist.github.com/karpathy/d4dee566867f8291f086

Minimal character-level language model with a Vanilla Recurrent Neural Network, in Python/numpy

min-char-rnn.py

1  """
2  Minimal character-level Vanilla RNN model. Written by Andrej Karpathy (@karpathy)
3  BSD License
4  """
5  import numpy as np
6
7  # data I/O
8  data = open('input.txt', 'r').read() # should be simple plain text file
9  chars = list(set(data))
10 data_size, vocab_size = len(data), len(chars)
11 print 'data has %d characters, %d unique.' % (data_size, vocab_size)
12 char_to_ix = { ch:i for i,ch in enumerate(chars) }
13 ix_to_char = { i:ch for i,ch in enumerate(chars) }
14
15 # hyperparameters
16 hidden_size = 100 # size of hidden layer of neurons
17 seq_length = 25 # number of steps to unroll the RNN for
18 learning_rate = 1e-1
```

<https://gist.github.com/karpathy/d4dee566867f8291f086>

您可以仔細研究看看

# 現在

- 再度讓我們拋開那些數學式
- 然後在甩掉那些程式碼！

# 純粹用使用者的角度

- 來看看 RNN 與 LSTM 這些循環神經網路，到底能做些甚麼事？

# Karpathy 拿這些程式來做了一堆測試

首先是把莎士比亞拿來學

# 結果電腦會自動寫小說了

PANDARUS:

Alas, I think he shall be come approached and the day  
When little strain would be attain'd into being never fed,  
And who is but a chain and subjects of his death,  
I should not sleep.

Second Senator:

They are away this miseries, produced upon my soul,  
Breaking and strongly should be buried, when I perish  
The earth and thoughts of many states.

DUKE VINCENTIO:

Well, your wit is in the care of side and that.

Second Lord:

They would be ruled after this chamber, and  
my fair nudes begun out of the fact, to be conveyed,  
Whose noble souls I'll have the heart of the wars.

Clown:

Come, sir, I will make did behold your worship.

VIOLA:

I'll drink it.

然後把維基百科原始檔拿來訓練



# 結果電腦也會寫維基文章了

Naturalism and decision for the majority of Arab countries' capitalide was grounded by the Irish language by [[John Clair]], [[An Imperial Japanese Revolt]], associated with Guangzham's sovereignty. His generals were the powerful ruler of the Portugal in the [[Protestant Immineners]], which could be said to be directly in Cantonese Communication, which followed a ceremony and set inspired prison, training. The emperor travelled back to [[Antioch, Perth, October 25|21]] to note, the Kingdom of Costa Rica, unsuccessful fashioned the [[Thrales]], [[Cynth's Dajoard]], known in western [[Scotland]], near Italy to the conquest of India with the conflict. Copyright was the succession of independence in the slop of Syrian influence that was a famous German movement based on a more popular servicious, non-doctrinal and sexual power post. Many governments recognize the military housing of the [[Civil Liberalization and Infantry Resolution 265 National Party in Hungary]], that is sympathetic to be to the [[Punjab Resolution]] (PJS)[<http://www.humah.yahoo.com/guardian.cfm/7754800786d17551963s89.htm> Official economics Adjoint for the Nazism, Montgomery was swear to advance to the resources for those Socialism's rule, was starting to signing a major tripad of aid exile.]]

# 有時竟然還寫出了 XML

```
<page>
  <title>Antichrist</title>
  <id>865</id>
  <revision>
    <id>15900676</id>
    <timestamp>2002-08-03T18:14:12Z</timestamp>
    <contributor>
      <username>Paris</username>
      <id>23</id>
    </contributor>
    <minor />
    <comment>Automated conversion</comment>
    <text xml:space="preserve">#REDIRECT [[Christianity]]</text>
  </revision>
</page>
```

應該是因為維基百科裏有些 XML 語法的文件吧！

然後把 Algebraic Geometry  
這本 latex 電子書拿來練

# 結果電腦寫出來的 latex

## ● 還可以正常的編譯成書！

For  $\bigoplus_{n=1, \dots, m} \mathcal{L}_{m,n} = 0$ , hence we can find a closed subset  $\mathcal{H}$  in  $\mathcal{H}$  and any sets  $\mathcal{F}$  on  $X$ ,  $U$  is a closed immersion of  $S$ , then  $U \rightarrow T$  is a separated algebraic space.

*Proof.* Proof of (1). It also start we get

$$S = \text{Spec}(R) = U \times_X U \times_X U$$

and the comparicoly in the fibre product covering we have to prove the lemma generated by  $\coprod Z \times_U U \rightarrow V$ . Consider the maps  $M$  along the set of points  $\text{Sch}_{fppf}$  and  $U \rightarrow U$  is the fibre category of  $S$  in  $U$  in Section, ?? and the fact that any  $U$  affine, see Morphisms, Lemma ?? . Hence we obtain a scheme  $S$  and any open subset  $W \subset U$  in  $\text{Sh}(G)$  such that  $\text{Spec}(R') \rightarrow S$  is smooth or an

$$U = \bigcup U_i \times_{S_i} U_i$$

which has a nonzero morphism we may assume that  $f_i$  is of finite presentation over  $S$ . We claim that  $\mathcal{O}_{X,x}$  is a scheme where  $x, x', s'' \in S'$  such that  $\mathcal{O}_{X,x'} \rightarrow \mathcal{O}'_{X',x'}$  is separated. By Algebra, Lemma ?? we can define a map of complexes  $\text{GL}_{S'}(x'/S'')$  and we win.  $\square$

To prove study we see that  $\mathcal{F}|_U$  is a covering of  $\mathcal{X}'$ , and  $\mathcal{T}_i$  is an object of  $\mathcal{F}_{X/S}$  for  $i > 0$  and  $\mathcal{F}_p$  exists and let  $\mathcal{F}_i$  be a presheaf of  $\mathcal{O}_X$ -modules on  $\mathcal{C}$  as a  $\mathcal{F}$ -module. In particular  $\mathcal{F} = U/\mathcal{F}$  we have to show that

$$\widetilde{M}^\bullet = \mathcal{I}^\bullet \otimes_{\text{Spec}(k)} \mathcal{O}_{S,s} - i_X^{-1} \mathcal{F}$$

is a unique morphism of algebraic stacks. Note that

$$\text{Arrows} = (\text{Sch}/S)_{fppf}^{\text{opp}}, (\text{Sch}/S)_{fppf}$$

and

$$V = \Gamma(S, \mathcal{O}) \mapsto (U, \text{Spec}(A))$$

is an open subset of  $X$ . Thus  $U$  is affine. This is a continuous map of  $X$  is the inverse, the groupoid scheme  $S$ .

*Proof.* See discussion of sheaves of sets.  $\square$

The result for prove any open covering follows from the less of Example ?? . It may replace  $S$  by  $X_{\text{spaces}, \text{étale}}$  which gives an open subspace of  $X$  and  $T$  equal to  $S_{Zar}$ , see Descent, Lemma ?? . Namely, by Lemma ?? we see that  $R$  is geometrically regular over  $S$ .

**Lemma 0.1.** Assume (3) and (3) by the construction in the description.

Suppose  $X = \lim |X|$  (by the formal open covering  $X$  and a single map  $\text{Proj}_X(\mathcal{A}) = \text{Spec}(B)$  over  $U$  compatible with the complex

$$\text{Set}(\mathcal{A}) = \Gamma(X, \mathcal{O}_{X, \mathcal{O}_X}).$$

When in this case of to show that  $\mathcal{Q} \rightarrow \mathbb{C}_{Z/X}$  is stable under the following result in the second conditions of (1), and (3). This finishes the proof. By Definition ?? (without element is when the closed subschemes are catenary. If  $T$  is surjective we may assume that  $T$  is connected with residue fields of  $S$ . Moreover there exists a closed subspace  $Z \subset X$  of  $X$  where  $U$  in  $X'$  is proper (some defining as a closed subset of the uniqueness it suffices to check the fact that the following theorem

(1)  $f$  is locally of finite type. Since  $S = \text{Spec}(R)$  and  $Y = \text{Spec}(R)$ .

*Proof.* This is form all sheaves of sheaves on  $X$ . But given a scheme  $U$  and a surjective étale morphism  $U \rightarrow X$ . Let  $U \cap U = \coprod_{i=1, \dots, n} U_i$  be the scheme  $X$  over  $S$  at the schemes  $X_i \rightarrow X$  and  $U = \lim_i X_i$ .  $\square$

The following lemma surjective restrocomposes of this implies that  $\mathcal{F}_{x_0} = \mathcal{F}_{x_0} = \mathcal{F}_{x, \dots, 0}$ .

**Lemma 0.2.** Let  $X$  be a locally Noetherian scheme over  $S$ ,  $E = \mathcal{F}_{X/S}$ . Set  $\mathcal{I} = \mathcal{I}_1 \subset \mathcal{I}_n$ . Since  $\mathcal{I}^n \subset \mathcal{I}^n$  are nonzero over  $i_0 \leq p$  is a subset of  $\mathcal{I}_{n,0} \circ \overline{A}_2$  works.

**Lemma 0.3.** In Situation ?? . Hence we may assume  $q' = 0$ .

*Proof.* We will use the property we see that  $\mathfrak{p}$  is the next functor (??). On the other hand, by Lemma ?? we see that

$$D(\mathcal{O}_{X'}) = \mathcal{O}_X(D)$$

where  $K$  is an  $F$ -algebra where  $\delta_{n+1}$  is a scheme over  $S$ .  $\square$

# 這是電腦產生的一段 latex 原始碼

```
\begin{proof}
We may assume that  $\mathcal{I}$  is an abelian sheaf on  $\mathcal{C}$ .
\item Given a morphism  $\Delta : \mathcal{F} \rightarrow \mathcal{I}$ 
is an injective and let  $\mathfrak{q}$  be an abelian sheaf on  $X$ .
Let  $\mathcal{F}$  be a fibered complex. Let  $\mathcal{F}$  be a category.
\begin{enumerate}
\item \hyperref[setain-construction-phantom]{Lemma}
\label{lemma-characterize-quasi-finite}
Let  $\mathcal{F}$  be an abelian quasi-coherent sheaf on  $\mathcal{C}$ .
Let  $\mathcal{F}$  be a coherent  $\mathcal{O}_X$ -module. Then
 $\mathcal{F}$  is an abelian catenary over  $\mathcal{C}$ .
\item The following are equivalent
\begin{enumerate}
\item  $\mathcal{F}$  is an  $\mathcal{O}_X$ -module.
\end{enumerate}
\end{enumerate}
\end{lemma}
```

然後 karpathy 把 linux 原始碼拿來訓練



# 訓練完後程式就會寫 C 語言了

只是這個 C 語言

應該是無法通過編譯

就算通過了編譯

應該執行下去也會

GG 的吧！

```
/*
 * Increment the size file of the new incorrect UI_FILTER group information
 * of the size generatively.
 */
static int indicate_policy(void)
{
    int error;
    if (fd == MARN_EPT) {
        /*
         * The kernel blank will coeld it to userspace.
         */
        if (ss->segment < mem_total)
            unblock_graph_and_set_blocked();
        else
            ret = 1;
        goto bail;
    }
    segaddr = in_SB(in.addr);
    selector = seg / 16;
    setup_works = true;
    for (i = 0; i < blocks; i++) {
        seq = buf[i++];
        bpf = bd->bd.next + i * search;
        if (fd) {
            current = blocked;
        }
    }
}
```

# 我覺得電腦寫的 C 語言

比我寫的漂亮很多  
看起來就很高級阿！

拿來交作業搞不好  
也可以呼嚨過的！

```
/*  
 * If this error is set, we will need anything right after that BSD.  
 */  
static void action_new_function(struct s_stat_info *wb)  
{  
    unsigned long flags;  
    int lel_idx_bit = e->edd, *sys & ~((unsigned long) *FIRST_COMPAT);  
    buf[0] = 0xFFFFFFFF & (bit << 4);  
    min(inc, slist->bytes);  
    printk(KERN_WARNING "Memory allocated %02x/%02x, "  
           "original MLL instead\n"),  
           min(min(multi_run - s->len, max) * num_data_in),  
           frame_pos, sz + first_seg);  
    div_u64_w(val, inb_p);  
    spin_unlock(&disk->queue_lock);  
    mutex_unlock(&s->sock->mutex);  
    mutex_unlock(&func->mutex);  
    return disassemble(info->pending_bh);  
}
```



然後 karpathy 寫了程式把  
神經網路的內部節點視覺化

# 結果看到這個

t	t	p	:	/	/	w	w	w	.	y	n	e	t	n	e	w	s	.	c	o	m	/	]	E	n	g	l	i	s	h	-	l	a	n	g	u	a	g	e		w	e	b	s	i	t	e		o	f	I	s	r	a	e	l	'	s		I	a	r		
t	p	:	/	/	w	w	w	.	b	a	c	a	h	e	t	s	.	c	o	m	/			-	x	g	l	i	s	h		l	i	n	g	u	a	g	e	s	a	i	r	s	i	t	e		o	f	t	s	I	a	e	l	i	s		s	i	n	g	
	d	:	x	n	e	.	w	a	e	a	.	.	a	w	a	t	o	a	.		s		&	n	t	i	a	c	a	-	s	a	r	d	e	e	l	h		o	a	n		t	b	i	s	a	n	f	a	n	r	e	i	f		'		a	a	t	d	
m	w	-	2		p	i	i	i	s	o	e	s	s	i	s	.	/	e	r	n	.	c	]	(	d	c	e	e	n		e	p	e	s	a	a	i	k	i		i	e	e	l	e	d	h	,	i	r	t	h	r	a	o	n	s	e		, c	o	s	e	
d	r	.	<	:	a	h	b	-	n	p	t	w	t	.	x	i		g	h	/	m	a	)	T	v	d	r	y	z	i		c	o	u	e	d	l	s	u	:	t	h	a	-	o	o		t	u	,	s	t	u	i	f		l	v	e	p	e	r	y	
s	t	p	,	t	c	o	a	2	d	r	u	l	w	o	c	l	e	n	s	r	]	p	.	I	l	v	a	o	d	,		e	y	t	c	-	n		d	m	-	o	i	b	u	v	s	]	b	b		i	m	s	u	l	t	a		t	l	y	b	n

g	e	s	t		n	e	w	s	p	a	p	e	r		'	'	[	[	Y	e	d	i	o	t	h		A	h	r	o	n	o	t	h	]	]	'	'		'	'	'	H	e	b	r	e	w	-	l	a	n	g	u	a	g	e		p	e	r	i	o	d			
e	t		a	a	w	s	p	a	p	e	r	s		o	'	[	[	T	e	l		t			i	(	f	e	a	n	e	m	t	i	]	'	'		*	'	'	[	e	r	r	e	w	s	l	e	n	g	u	a	g	e	:	a	r	o	s	o	d	i			
i	r		s	c	o	e			e	n	a			i	T	T	h	A	o	a	i	n	n	h		S	r	m	u	w	]		e	y			s				[	'	i	n	e	i	a	'	s	i	w	d	d	e	'	h	s	o	l	r	i	f	r	:			
u	s	.	.	s	e	t	l	g	o	r		s	.	a	s	a	t	C	a	r	e	e	g	'	a	C	l	r	i	s	z	]	i	e	'	:	:	,	#	:	T	A	a	a	a	a	t		B	a	s	e	e	i	l	o	'	i	a	n	f	v	l				
		-			t	u	a	e	v	r	t	i	d	,	t	B	A	m	S	u	s	y	u	t	]	]	A	s	a	o	i	g	s	]	]	,	.		.	:	s	M	B	o	l	o	u	s	:	T	o	u	a	-	n	:	d		w	o	a		p	n	u		
a	,	d	,	i	i	u	i	t	i	c	p	.	]	(	l	S	v	H	v	t	u	s	u	i	e	D	n	o	e	g	a	n	o			.	,	]	:	{			C	C	u	i	b	o	h	e	C	y	b	k	s	l	s	:	r	-	e	p	c	n	t	s	

i	c	a	s	:	'	'	*	'	'	[	[	G	l	o	b	e	s	]	]	'	'	[	h	t	t	p	:	/	/	w	w	w	.	g	l	o	b	e	s	.	c	o	.	i	l	/	]	b	u	s	i	n	e	s	s	d	a							
c	a	l	:	'	'	'	*	'	'	[	T	a	a	b	a	]	'	'	(	[	t	t	p	:	/	/	w	w	w	.	b	u	o	b	a	l	.	c	o	m	u	n	/	s	A	-	y	t	i	n	e	s	s	a	e	t								
s	t	l	'							[	h	A	e	o	v	e	l	t		s	a	h	a	d	:	x	g	e	.	w	a	o	i	r	.	r	t	o	a	.	e	l	.	i	T	&	a	i	e	g		e	o	o	y									
t	t	'	'							'	&	[	&	&	m	C	o	e	r	o	n	e	'	:	:	,	i	'	o	d	w	.	:	n	i	i	i	s	a	a	u	e	.	e	n	i	/	o	m	l	c	C	.	(	e	f	t	g	i	r		i	i	u
a	'	n	:	,	C	:	&	:	#	*	:	a	f	D	r	u	s	u	]	l	,		.	o	m	e	l		p	<	,	d	h	a	:	d	e	u	o	o	t	/	i	h	n	c	s	i	f	S	,	u	r	h	o	s	t	,	t	u	n			
n	k	i		<	]	:	&	1	1	s		T	G	u	i	t	r	s	i	,			:	b	a	c	m	r	-	x	t	p	o	b	-	g	r	e	s	i	s	l	e	r	l	n	a	f	a	D	]	l	o	s	p	t	a	d	,	i	f	r	m	

i	l	y	*	'	'	[	[	H	a	a	r	e	t	z		H	a	'	A	r	e	t	z	]	]	'	'	[	h	t	t	p	:	/	/	w	w	w	.	h	a	a	r	e	t	z	.	c	o	.	i	l	/	]	R	e	l	a	t	i	v			
l	y	*	'	'	[	[	T	e	r	r	d	n		F	e	r	a	n	t	a	h	]	]	'	'	(	[	t	t	p	:	/	/	w	w	w	.	b	o	n	m	d	s	t	.	c	o	m	u	n	/	s	-	e	s	a	t	e	o	i				
r	e		'	'	h	A	i	l	n	n	t	t	e	H	a	l	s	r	c	n	o	l	'		s	a	h	a		d	:	x	n	e	.	w	a	a	m	r	t	d	h	e	o	h	.	o	l	.	c	&	o	p	i	n	i	v	e					
k	i	.	:	*	s	C	O	S	a	n	l	t		h	i	T	i	m	'	l	i	]	e			:	,	i	m	c	d	w	-	2	◆	p	h	i	i	s	e	r	d	i	t	.	i	n	a	/	c	m	f	i	.	(	a	f	l	c	a	n	a	
d	s	-	!	[	t	B	T	C	o	m	m	g	d	]	]	W	o	n		a		a	e	,	:		.	b	a	e	r	.	<	t	a	i	b	-	d	u	l	c	n	n	c	/	a	r	n	e	s	i	]	l	i	c	e	y	s	t	o			
n	d	s	#	&	:	G	I	D	u	v	c	c	s	a	o	S	u	c	l	t	e	l	]	z		,		:	o	'	o	m	t	]	,	:	e	o	a	2	n	i	v	f	s	r	o	o	e	i	u	n	a	l	a	)		u	v	v	r	o		

綠色部分是辨識區：這個節點似乎是在辨識超連結！



這個

*	'	'	[	[	J	e	r	u	s	a	l	e	m	R	e	p	o	r	t	]	]	'	'	[	h	t	t	p	:	/	/	w	w	w	.	j	r	e	p	.	c	o	m	/	]	L	e	f	t	-	o	f	-	c	e	n	t	e	r	E	n				
*	'	'	[	h	T	o	a	u	s	a	l	e	m	a	o	g	u	r	t	]	]	'	'	(	h	t	t	p	:	/	/	w	w	w	.	b	s	i	n	i	o	o	m	/	-	i	a	t	a	f	t	e	n	t	e	r	(	n	g						
			[		'	[	C	a	s	s	m	e	n	e	]	B	e	a	o	n	d	s			s	a	[	a	d	:	x	n	e	.	w	a	a	a	a	o	c	a	.	s	&	a	t	o	-	-	n	f	h	h	l	s	u	m	-	o	u	c			
			'	s		m	F	u	r	n	l	s	i	a	e	t	a	l	l	s	a	'	:	:	,	i	'	c	d	w	-	2	t	p	i	i	i	s	o	e	g	.	e	r	/	.	a	]	(	o	s	e	s	w	r	-	c	i	d	d	r	s	[	m	t
:	*	:	A	q	D	e	n	e	b	i	u	t	n		C	i	p	r	e	e		,	.	b	1	e	m	r	.	9	:	a	h	b	-	n	p	u	m	u	g	h	n	m	p	)	T	e	i	r	e	t	u	:	e	o	s	e	o	d	s	a	l	d	
#	T	&	T	f	S	i	w	r	p	e	]	]	a	l	u	v	e	l	r	u	,	s	:	-	m	p	r	t	s	<	mo	a	2	d	e	y	s	h	i	l	r	]	c	.	A	u	g	l	,	1	p	,	l	a	r	c			:	f	a	e			

g	l	i	s	h	[	[	w	e	e	k	l	y	n	e	w	s	p	a	p	e	r	]	]	*	'	'	[	Y	N	e	t	N	e	w	s	]	]	'	'	[	h	t	t	p	:	/	/	w	w	.	y	n	e	t	n	e	w	s	.	c		
r	i	s	h	c	[	C	a	a	k	l	y	]c	a	w	s	p	a	p	e	r	]	]	*	'	'	[	h	T	A	A		a	t		]	'	'	(	h	t	t	p	:	/	/	w	w	.	b	a	c	a	h	e	t	s	.	c	o			
i	a	c	i	-	l	h	S	o	i	p	]i	s	e	c		]e	n	p	]s	.	'	'	'	[	C	o	*	w	e	s	s		s	a	[	a	d	:	x	n	e	.	w	a	e	a	.	a	w	a	t	o	a									
e	e	n	a	,	p	C	c	i	e	t	n	e	d	l	o	x	]g	i	c	i			s	'	[	s	A	m	F	e	S	a	h	o	n	]t	'	:	:	,	i	m	o	m	w	-	2	◆	p	i	i	s	o	e	s	s	i	s	.	/	e	r
s	y	z	.	s	f	p	e	n	n	a		r	u	e	l		r	r	a	.	'	#	*	:	o	D	u	F	r	e	i	u	e	p	,	:	b	1	e	d	r	.	<	:	a	h	b	-	n	p	t	w	t	.	x	i	g	h				
a	d	p	e	a	m	A	r	b	d	e	o	r	p	i	t	e	e	]d	t	s	-		T	{	[	B	a	A	v	T	p	o	S	w	a	o	,	.	.	o	a	c	s	t	p	,	t	c	o	a	2	d	r	u	l	w	o	c	l	e	n	s

o	m	/	]	E	n	g	l	i	s	h	-	l	a	n	g	u	a	g	e	w	e	b	s	i	t	e	o	f	i	s	r	a	e	l	'	s	l	a	r	g	e	s	t	n	e	w	s	p	a	p	e	r	'	'	[	[	Y	e	d									
m	/			-	x	g	l	i	s	h		l	i	n	g	u	a	g	e	s	a	i	r	s	i	t	e	o	f	t	s	l	a	e	l	i	s		s	i	n	g	e		t	a	a	w	s	p	a	p	e	r	s	o	'	[	[	T	e	l						
.		s		&	n	t	i	a	c	a	-	s	a	r	d	e	e	l	h		o	a	n		t	b	i	s	a	n	f	a	n	r	e	i	f		'		a	a	t	d	i	r		s	c	o	e		e	n	a			i	T	T	h	A	o	a	i			
n	.	c	]	(	d	c	e	e	n		e	p	e	s	a	a	i	k	i		i	e	e	l	e	d	h	,	i	r	t	h	r	a	o	n	s	e		,	c	o	s	e	u	s	.	.	s	e	t	l	g	o	r		s	.	a	s	a	t	C	a	r	e		
/	m	a	)	T	v	d	r	y	z	i		c	o	u	e	d	l	s	u	:	t	h	a	-	o	o		t	u	,	s	t	u	i	f		l	v	e	p	e	r	y	-			t	u	a	e	v	r	t	i	d	,	t	B	A	m	S	u	s	y				
r	]	p	.	l	l	v	a	o	d	,	,	e	y	t	c	-	n		d	m	-	o	i	b	u	v	s	]	b	b		i	m	s	u	l	t	a		t	l	y	b	n	a	,	d	,	i	i	u	i	t	i	c	p	.	]	(	l	S	v	H	v	t	u		

i	o	t	h	A	h	r	o	n	o	t	h	]	]	'	'	'	'	H	e	b	r	e	w	-	l	a	n	g	u	a	g	e	p	e	r	i	o	d	i	c	a	l	s	:	'	'	'	*	'	'	[	[	G	l	o	b	e	s	]	]	'							
t			i	(	f	e	a	n	e	m	t	i	]	'	'	*	'	'	[	e	r	r	e	w	s	l	e	n	g	u	a	g	e	:	a	r	o	s	o	d	i	c	a	l	:	'	'	'	*	'	'	'	[	T	a	a	b	a		]	'	'						
n	n	h		S	r	m	u	w	]	e	y		s					[	'	i	n	e	i	a	'	s	i	w	d	d	e	'	h	s	o	l	r	i	f	r	:	s	t	l	'									[	h	A	e	o	v	e	l	t			s			
e	g	'	a	C	l	r	i	s	z	]	i	e	'	:	:	,	#	:	T	A	a	a	a	a	t		B	a	s	e	e	i	l	o	'	i	a	n	f	v	l		t	t	'	'							'	&	[	&	&	m	C	o	e	r	o	n	e	'	:	:
u	t	]	]	A	s	a	o	i	g	s	]	]	,	.	.	:	s	M	B	o	l	o	u	s	:	T	o	u	a	-	n	:	d		w	o	a	p	n	u	a	'	n	:	,	C	:	&	:	#	*	:	a	f	D	r	u	s	u	]	l	,						
s	u	i	e	D	n	o	e	g	a	n	o		.	,	]	:	{		C	C	u	i	b	o	h	e	C	y	b	k	s	l	s	:	r	-	e	p	c	n	t	s	n	k		i		<	]	:	&	1	1	s		T	G	u	i	t	r	s	i	,				

似乎是辨識 `[[...]]` 的節點！

這個

*	'	'	[	[	J	e	r	u	s	a	l	e	m	R	e	p	o	r	t	]	]	'	'	[	h	t	t	p	:	/	/	w	w	w	.	j	r	e	p	.	c	o	m	/	]	L	e	f	t	-	o	f	-	c	e	n	t	e	r	E	n					
*			[	h	T	o	a	u	s	a	l	m	a	o	g	u	r	t	]	]	'	'	(	h	t	t	p	:	/	/	w	w	w	.	b	s	i	n	i	o	o	m	/		-	i	a	t	a	f	t	e	n	t	e	r	(	n	g							
	[		'	[	C	a	s	s	m	e	n	e	]	B	e	a	o	n	d	s		s	a	[	a	d	:	x	n	e	.	w	a	a	a	a	o	c	a	.	s	&	a	t	o	-	-	n	f	h	h	l	s	u	m	-	o	u	c							
	'	s		m	F	u	r	n	l	s		i	a	e	t	a	l	l	s	a		:	:	,	i	'	c	d	w	-	2	t	p	i	i	i	s	o	e	g	.	e	r	/	.	a	]	(	o	s	e	s	w	r	-	c	i	d	d	r	s	[	m	t		
:	*	:	A	q	D	e	n	e	b	i	u	t	n		C	i	p	r	e		e		,		.	b	1	e	m	r	.	9	:	a	h	b	-	n	p	u	m	u	g	h	n	m	p	)	T	e	i	r	e	t	u	:	e	o	s	e	o	d	s	a	l	d
#	T	&	T	f	S	i	w	r	p		e	]	]	a	l	u	v	e	l	r	u	,	s	:	-	m	p	r	t	s	<	mo	a	2	d	e	y	s	h	i	l	r	]	c	.	A	u	g	l	,	1	p	,	l	a	r	c		:	f	a	e				

g	l	i	s	h		[	[	w	e	e	k	l	y		n	e	w	s	p	a	p	e	r	]	]	*	'	'	[	[	Y	N	e	t	N	e	w	s	]	]	'	'	[	h	t	t	p	:	/	/	w	w	w	.	y	n	e	t	n	e	w	s	.	c
l	i	s	h		c	[	C	a	a	k	l	y	]	c	a	w	s	p	a	p	e	r	]	]	*	'	'	[	h	T	a	A		a	t		]	'	'	(	h	t	t	p	:	/	/	w	w	w	.	b	a	c	a	h	e	t	s	.	c	o		
i	a	c	i	-	l	h	S	o	i	p	]	i		s	e	c		]	e	n	p	]	s	.	'	'	'	[	C	o		*	w	e	s	s	]		s	a	[	a	d	:	x	n	e	.	w	a	e	a	.	a	w	a	t	o	a					
e	e	n	a	,	p	C	c	i	e	t	n	e	d	l	o	x	]	g	i	c	i			s	'	[	s	A	m	F	e	S	a	h	o	n	]	t	'	:	:	,	i	m	o	m	w	-	2	pi	i	i	s	o	e	s	s	i	s	.	/	e	r	
s	y	z		.	s	f	p	e	n	n	a		r	u	e	l		r	r	a		.	'	#	*	:	o	D	u	F	r	e	i	u	e	p	,	:	b	1	e	d	r	.	<	:	a	h	b	-	n	p	t	w	t	.	x	i		g	h			
a	d	p	e	a	m	A	r	b	d	e	o	r	p	i	t	e	e	]	d	t	s	-			T	{	[	B	a	A	v	T	p	o	S	w	a	o	,	.	.	o	a	c	s	t	p	,	t	c	o	a	2	d	r	u	l	w	o	c	l	e	n	s

似乎是辨識 [[....]] 《尾端》的節點！



# 還有這個

'	'	[	[	T	h	e		G	u	a	r	d	i	a	n	]	]	'	'	.		*	'	'	[	[	I	s	r	a	e	l	I	n	s	i	d	e	r	]	]	'	'	.	[	h	t	t	p	:	/	/	w	w	w	.	i	s	r	a	e	l	i	n	s
t	.	[	[	T	h	e		S	o	i	r	d	a	n	]	]	'	'	.							[	[	T	n	i	a	e	l										(	[	t	t	p	:	/	/	w	w	w	.	b	m	s	a	c	i	.	n	g	.	
a	i	G	h	A	o	i	o	M	r	i	t	r	i	n	]	]			s	,		*	[		'	h	C	s	i	f	i	D	a	t	s	i	]	]			s	,	a	h	a		d	:	x	n	e	.	w	s	n	o	o	n	s	i	t	e	t		
h	h	B	m	S	r	o	i	C	a	n	n	t	]e	c	s	,	.	.	#	T	[	a	m	D	m	a	e	i	t	]s	]t	a	n	s	'	:	:	.	i	'	o	l	.	:	g	i	i	;a	s	c	e	.	.	l	.	.	e								
s	m	T	I	G	e	u	y	L	u	s	i	n	s	c	a	.	:	]	=	S	:	C	T	A		o	o	n	f	e	]s	]c	s	,	,			o	1	e	c	w	-	x		p	o	b	i	g	c	k	.	r	t	a	a	d	o						
m	e	D	T	C	a	a	r	G	e	a	d	a	n	o	l	'	'	:	.	&	:	'	s	T	O	S	t	t	u	l	]c	e	o	]] i	.s	]'	b	m	p	d	r	,	<	t	m	h	d	-	n	e	l	i	e	f	s	s	s	a							

似乎是專門辨識 `www` 後面兩個 `w` 的節點！

# 有些神經網路節點

- 專門抓行尾

Cell sensitive to position in line:

The sole importance of the crossing of the Berezina lies in the fact that it plainly and indubitably proved the fallacy of all the plans for cutting off the enemy's retreat and the soundness of the only possible line of action--the one Kutuzov and the general mass of the army demanded--namely, simply to follow the enemy up. The French crowd fled at a continually increasing speed and all its energy was directed to reaching its goal. It fled like a wounded animal and it was impossible to block its path. This was shown not so much by the arrangements it made for crossing as by what took place at the bridges. When the bridges broke down, unarmed soldiers, people from Moscow and women with children who were with the French transport, all--carried on by vis inertiae--pressed forward into boats and into the ice-covered water and did not, surrender.

# 也有些節點專門抓 "... " 引用區

Cell that turns on inside quotes:

"You mean to imply that I have nothing to eat out of.... On the contrary, I can supply you with everything even if you want to give dinner parties," warmly replied Chichagov, who tried by every word he spoke to prove his own rectitude and therefore imagined Kutuzov to be animated by the same desire.

Kutuzov, shrugging his shoulders, replied with his subtle penetrating smile: "I meant merely to say what I said."

# 還有些專門抓

- if 裡面的運算式

Cell that robustly activates inside if statements:

```
static int __dequeue_signal(struct sigpending *pending, sigset_t *mask,
    siginfo_t *info)
{
    int sig = next_signal(pending, mask);
    if (sig) {
        if (current->notifier) {
            if (sigismember(current->notifier_mask, sig)) {
                if (!(current->notifier)(current->notifier_data)) {
                    clear_thread_flag(TIF_SIGPENDING);
                    return 0;
                }
            }
        }
        collect_signal(sig, pending, info);
    }
    return sig;
}
```



# 另一些專門抓註解

Cell that turns on inside comments and quotes:

```
/* Duplicate LSM field information. The lsm_rule is opaque, so
 * re-initialized. */
static inline int audit_dupe_lsm_field(struct audit_field *df,
                                       struct audit_field *sf)
{
    int ret = 0;
    char *lsm_str;
    /* our own copy of lsm_str */
    lsm_str = kstrdup(sf->lsm_str, GFP_KERNEL);
    if (unlikely(!lsm_str))
        return -ENOMEM;
    df->lsm_str = lsm_str;
    /* our own (refreshed) copy of lsm_rule */
    ret = security_audit_rule_init(df->type, df->op, df->lsm_str,
                                   (void *)&df->lsm_rule);
    /* Keep currently invalid fields around in case they
     * become valid after a policy reload. */
    if (ret == -EINVAL) {
        pr_warn("audit rule for LSM '%s' is invalid\n",
                df->lsm_str);
        ret = 0;
    }
    return ret;
}
```

但是也有些  
我們看不出到底在抓甚麼！

A large portion of cells are not easily interpretable. Here is a typical example:

```
/* Unpack a filter field's string representation from user-space
 * buffer. */
char *audit_unpack_string(void **bufp, size_t *remain, size_t len)
{
    char *str;
    if (!*bufp || (len == 0) || (len > *remain))
        return ERR_PTR(-EINVAL);
    /* Of the currently implemented string fields, PATH_MAX
     * defines the longest valid length.
     */
}
```

# 透過 karpathy 這篇文章的介紹

- The Unreasonable Effectiveness of Recurrent Neural Networks

# 相信

- 您應該會對 RNN 循環神經網路  
留下深刻的印象！

# 看到這裡

- 相信身為程式人的你  
一定會感覺到手癢了！

# 就讓我們

- 真正來寫程式感受一下，到底 RNN/LSTM 這類的循環神經網路是甚麼東西好了！

# 我們可以用 node.js

- 然後安裝 neataptic.js 這個循環神經網路的套件！

# 然後 寫個 小型 程式

```
GitHub, Inc. [US] | https://github.com/ccckmit/ai6/blob/master/book/nn/neataptic/lstmEx.js

2  var net = new neataptic.Architect.LSTM(1, 6, 1)
3
4  // Train a sequence: 000100010001....
5  var trainData = [
6    { input: [0], output: [0] },
7    { input: [0], output: [0] },
8    { input: [0], output: [1] },
9    { input: [1], output: [0] },
10   { input: [0], output: [0] },
11   { input: [0], output: [0] },
12   { input: [0], output: [1] }
13 ]
14 net.train(trainData, {
15   log: 500,
16   iterations: 6000,
17   error: 0.03,
18   clear: true,
19   rate: 0.05
20 })
21
22 var s = [0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0]
23 for (var i = 0; i < s.length; i++) {
24   console.log('s[%d]=%d next=%d', i, s[i], net.activate([s[i]]))
25 }
```



您可以從下列網址直接剪貼

<https://github.com/ccckmit/ai6/blob/master/book/nn/neataptic/lstmEx.js>

# 或者直接用我寫好的 ai6 專案

```
$ git clone https://github.com/ccckmit/ai6.git
```

```
$ cd ai6
```

```
$ npm install
```

```
$ cd book/nn/neataptic
```

# 這個程式會想辦法預測

- 000100010001 ...

這樣的規則序列！

# 執行結果如下

```
GitHub, Inc. [US] | https://github.com/ccckmit/ai6/tree/master/book/nn/neataptic

D:\Dropbox\github\ccc\course\ai6\book\nn\neataptic>node lstmEx
iteration 500 error 0.1943154782321371 rate 0.05
iteration 1000 error 0.19233348640050332 rate 0.05
iteration 1500 error 0.1894284988926231 rate 0.05
iteration 2000 error 0.17355676942961443 rate 0.05
iteration 2500 error 0.1276460523624396 rate 0.05
iteration 3000 error 0.09813105291227837 rate 0.05
iteration 3500 error 0.07907866945414652 rate 0.05
iteration 4000 error 0.06639723506046145 rate 0.05
iteration 4500 error 0.057299079889706485 rate 0.05
iteration 5000 error 0.05128957015820507 rate 0.05
iteration 5500 error 0.04784415549193767 rate 0.05
iteration 6000 error 0.04359995964360381 rate 0.05
s[0]=0 next=0.02618906821183275
s[1]=0 next=0.2596994965592306
s[2]=0 next=0.7876609870046056
s[3]=1 next=0.00003776187352897784
s[4]=0 next=0.037822070818615154
s[5]=0 next=0.3486835242452125
s[6]=0 next=0.8471095295847788
s[7]=1 next=0.000023611284930390403
s[8]=0 next=0.02272510397115269
s[9]=0 next=0.22632312738800156
s[10]=0 next=0.7465776770404702
```

您可以看到  
訓練完的程式  
總是領先預測  
下一個出現的  
位元

# 除了學習 01 序列以外

- 也可以用 LSTM 來學習字串序列！

# 像是這個範例

就會學習並預測 am i concious? or am I not? 這個字串序列

```
62
63 var text = `Am I concious? Or am I not?`.toLowerCase() // Text to learn
64 var characters = getCharSet(text)
65 var onehot = oneHotEncoding(characters)
66 var dataSet = text2dataSet(text, onehot)
67 var network = new Architect.LSTM(characters.length, 10, characters.length)
68
69 console.log('Characters=%s, count=%d', characters, Object.keys(onehot).length)
70 console.log('onehot=%j', onehot)
71 console.log('dataset.size=%d, dataSet=%j', dataSet.length, dataSet)
72 console.log('Network conns', network.connections.length, 'nodes', network.nodes.length)
73
74 network.train(dataSet, {
75   log: 1,
76   rate: 0.1,
77   cost: Methods.Cost.MSE,
78   error: 0.005,
79   clear: true
80 })
81
82 // writeSentence(text, onehot, dataSet)
83
84 writeSentence(text, onehot, dataSet)
```

<https://github.com/ccckmit/ai6/blob/master/book/nn/neataptic/lstmCbyc.js>

# 但是這樣並不能顯示

- LSTM 這類循環神經網路的威力

# 於是我寫了一組通用的

## ● 字串學習預測程式！

主模組：<https://github.com/ccckmit/ai6/blob/master/book/nn/neataptic/lstm.js>

訓練程式：<https://github.com/ccckmit/ai6/blob/master/book/nn/neataptic/lstmTrain.js>

預測程式：<https://github.com/ccckmit/ai6/blob/master/book/nn/neataptic/lstmPredict.js>



# 我們可以把一個文字檔當作輸入

- 程式就能學會該檔案的寫作風格！

# 像是學習簡單的狗世界英文語句

## 範例 1：狗世界的英文字串學習

輸入檔： [data/edog.txt](#)

```
a dog
little dog
little black dog
black dog
a little black dog
a little dog
```

用訓練完成的網路產生字串

```
$ node lstmPredict edog.lstm.json 100
===== gen (prefix=[]) =====
a dog
little dog
litle dog
a little dog
a dog
little dog
a litle dog
a litle dog
a little do
```

訓練過程：

```
$ node lstmTrain data/edog.txt edog.lstm.json
seqText="a dog\r\nlittle dog\r\nlittle black dog\r\nblack dog\r\na little black
dog\r\na little dog"
words = ["[#start#]", "a", " ", "d", "o", "g", "\r", "\n", "l", "i", "t", "t", "e", "b", "c", "k"]
Network conns 2025 nodes 90
iteration 1 error 0.07841929704993059 rate 0.1
iteration 2 error 0.05948436172466311 rate 0.1
```

# 或者是狗世界的《中文語句》

## 範例 2：狗世界的中文字串學習

輸入檔： [data/cdog.txt](#)

一隻狗  
小狗  
小黑狗  
黑狗  
一隻小黑狗  
一隻小狗

訓練過程：

```
$ node lstmTrain data/cdog.txt cdog.lstm.json
...
iteration 313 error 0.010488494419177973 rate 0.1
iteration 314 error 0.01030299447451737 rate 0.1
```

```
$ node lstmPredict cdog.lstm.json 100
===== gen (prefix=[]) =====
一隻狗
小狗
小黑狗
一隻小狗
一隻小狗
小黑狗
一隻小狗
小黑狗
一隻小狗
一隻小狗
小黑狗
一隻小狗
一隻狗
小黑狗
一隻小狗
小黑狗
一隻小狗
一隻小狗
小黑
```

# 又或者是數學運算式的學習

## 範例 3：數學運算式的學習

輸入檔： [data/exp.txt](#)

```
a
b
a+b
(a+b)+a
a+(b+a)
(a+b)+(b+a)
(a+(b+a))+b
((a+b)+a)+((b+a)+b)
```

訓練過程：

```
$ node lstmTrain data/exp.txt exp.lstm.json
...
iteration 384 error 0.01110337015812884 rate 0.1
iteration 385 error 0.010713149768480227 rate 0.1
iteration 386 error 0.012236864162624268 rate 0.1
```

用訓練完成的網路產生字串

```
$ node lstmPredict exp.lstm.json 100
===== gen (prefix=[]) =====
a
b
a+b
(a+b)+a
a+(b+a)
(a+b)+(b+a)
(a+b)+(b+a)
(a+b)+((a+b)+a)+(b+a)
(a+b)+(b+a)+b
(a+(b+a
```

# LSTM 神經網路

- 都可以在一兩分鐘內就透過  
很小的檔案學習完成
- 並且依樣畫葫蘆的產生很不錯的文章！

# 現在

- 您應該可以理解甚麼是 RNN/LSTM 神經網路了！

# 透過我們的 node. js 程式

- 您可以親自玩玩看！

<https://github.com/ccckmit/ai6/tree/master/book/nn/neataptic>

# 透過實際的體驗

- 您應該會更深入的瞭解深度學習的 RNN/LSTM 網路才對！



這就是我們今天的

# 十分鐘系列

希望您會喜歡！

我們下回見！

Bye Bye!