Recurrent Neural Nets (RNN)

L-1- Introduction Why RNN?

MLP x (vector) → □ → y

CNN x (Image) → □ → y

RNN x (Sequence) → □ → y

$x \rightarrow$ sequence

eg. (1) Sequence of words → NLP

eg → Amazon fine food reviews. +ve/-ve

This phone is very fast → sequence of words.

fast is very this phone → this makes no sense

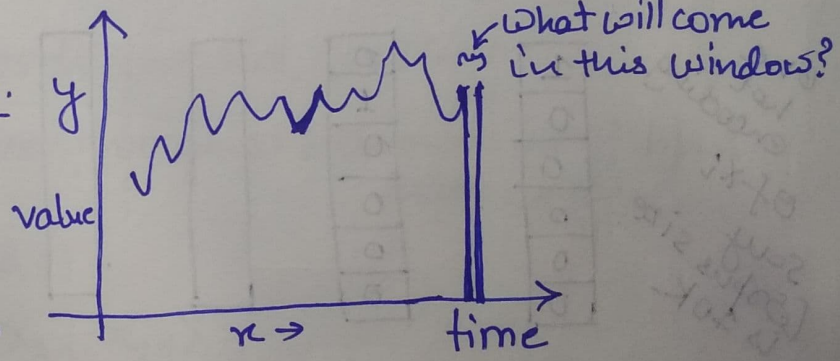
- All the techiques seens till now we have discarded the sequence information.
  - → {BOW, TF-idf, Avg W2V}

(2) Time series Data.

→ 'window' the data

→ "freq" domain

→ Stock market/car cab prediction.



(3) Machine translation

English → M.L. → Spanish

Seq → Seq

(4) Speech Recognition

Sequence in Audio → English text (sequence)

(5) Image → Captioning → {english text} → text search like google search.

this is useful to google image search

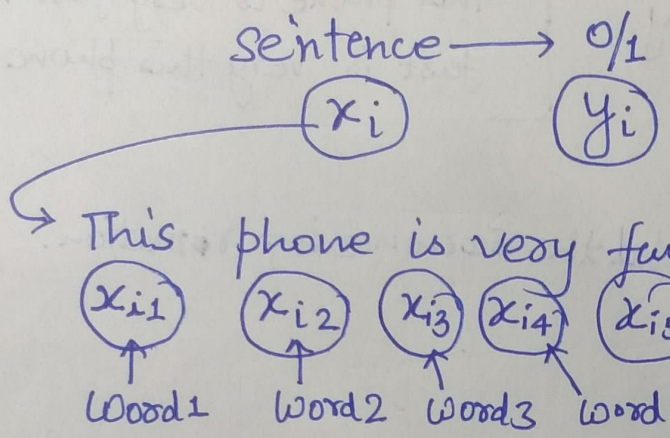
Core Idea:→ When output depends on the Seq. of input rather than set of inputs.

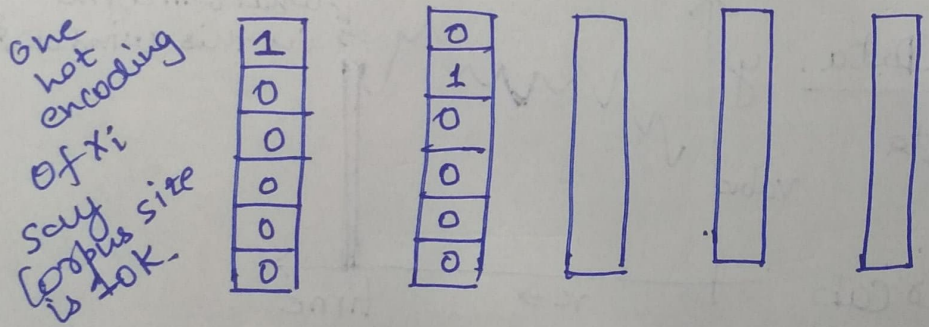
↓

new type of NN

→{retains & leverage the stepINfo}

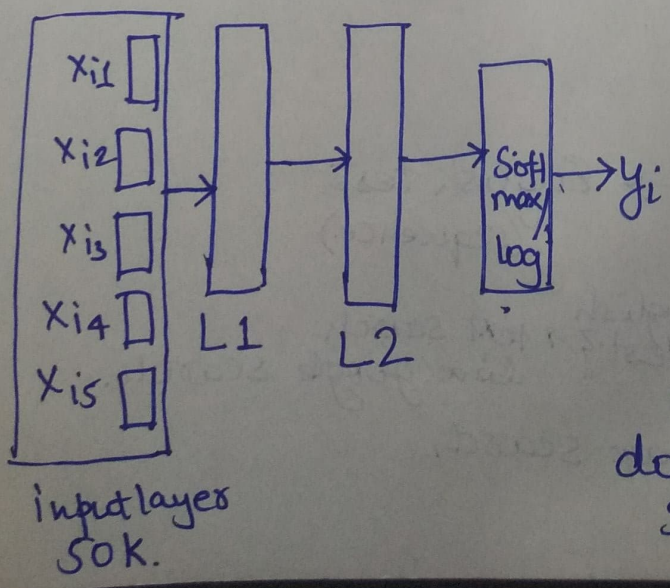
eg.→ Amazon food reviews





Now for 1st point I have input of 5x10K.

So why cannot we train simple MLP if will also have sequence info?



But problem →

x_i → This is very good phone

x_2 → This phone is good.

① Varing words size.
(All sentense of different length)

now you can say we can do zero padding for shorter Sentence.

Now if testsentence can of even bigger length than what will be our input dimention.

So, In short you will be having unnecessarily lot of zeros in input data & V.V. large input dimension. and Billion of weights to train.

L-2. RNN ("Recurrent" Neural Network)

↓ repeating

(e.g.) Amazon food reviews → binary classification {0,1}

$\mathcal{D}\{x_i, y_i\}$

- x_i → sentence (seq of words)
- y_i → binary

$x_{11}, x_{12}, x_{13}, x_{14} \rightarrow X_1 \rightarrow y_i$

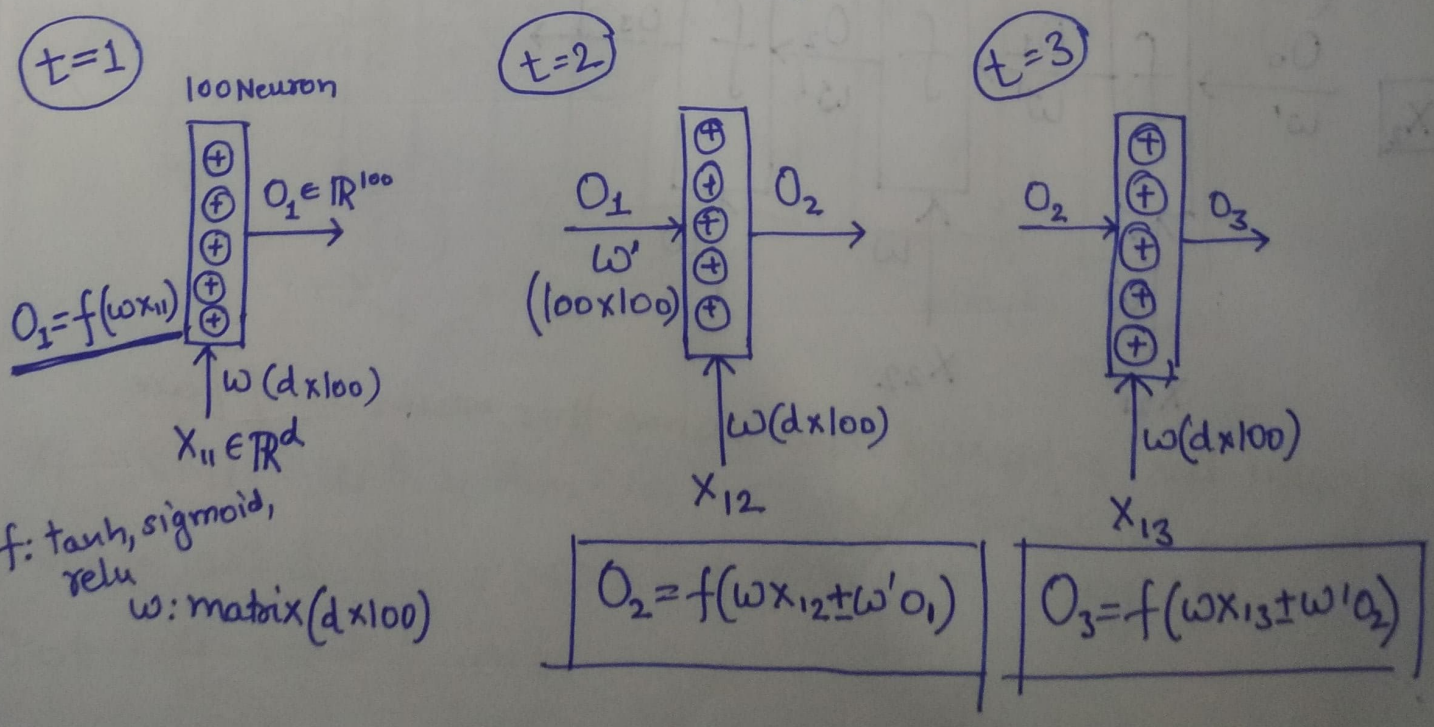
$x_{21}, x_{22}, x_{23} \rightarrow X_2 \rightarrow y_i$

→ words: one-hot encoding

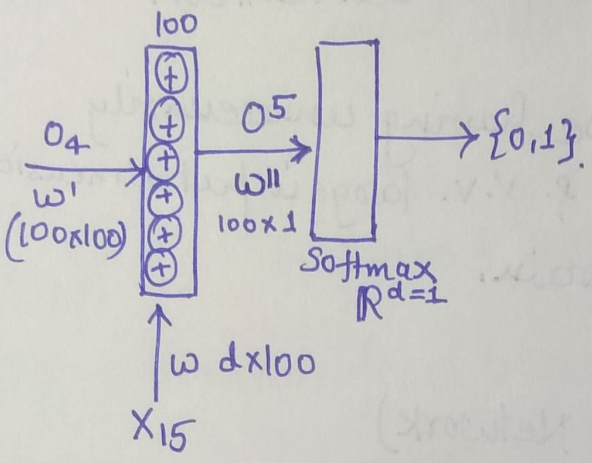
{size of vocablar :- dimension of input corpous. $\mathbb{R}^d$.

Task :- seq of words → {0,1}

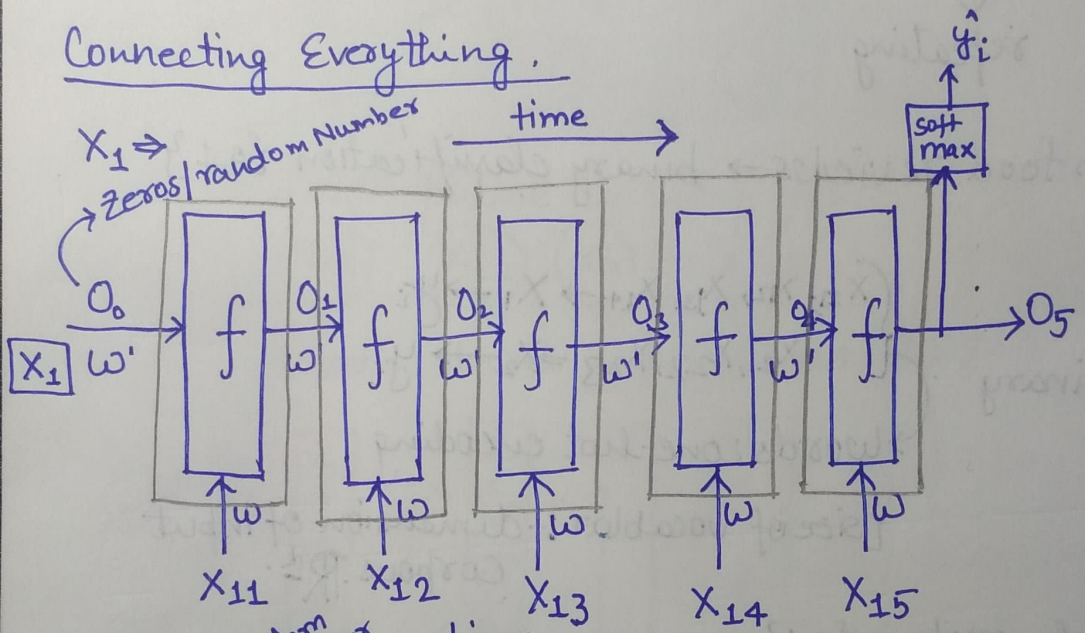
$X_1 \rightarrow \langle x_{11}, x_{12}, x_{13}, x_{14}, x_{15} \rangle, y_i$

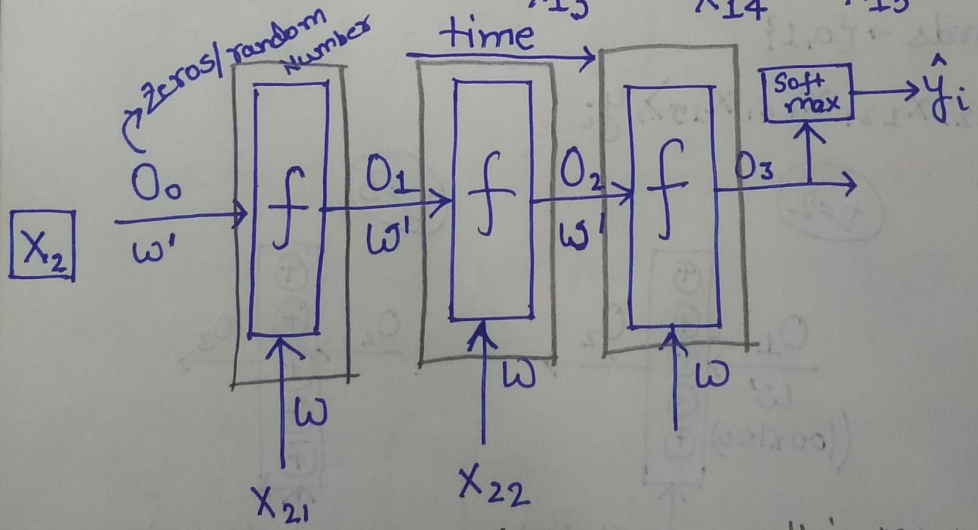


$t = 5$



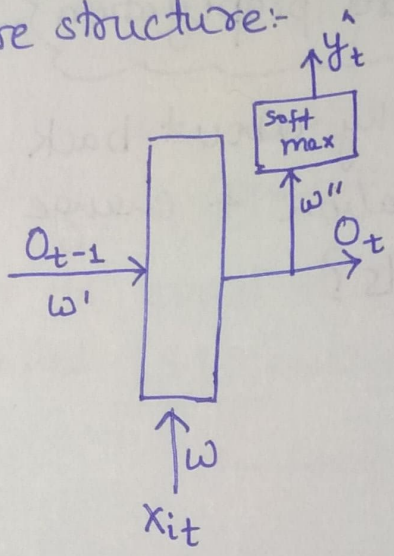
Connecting Everything.



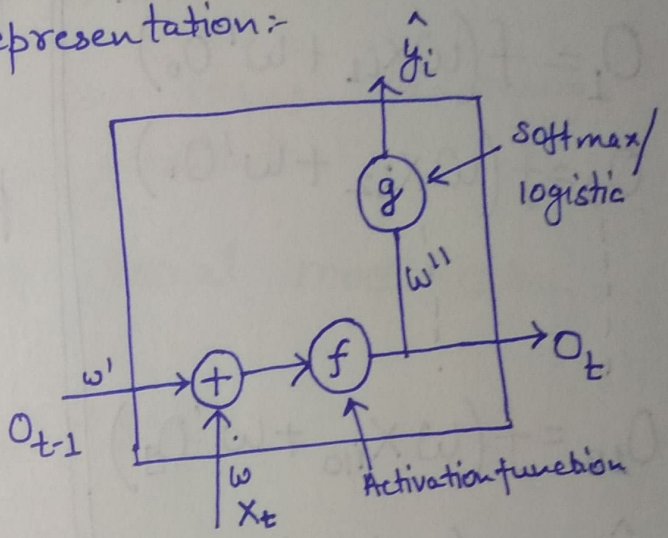


persil box represent the same this. means all [] are pointing to same layer.

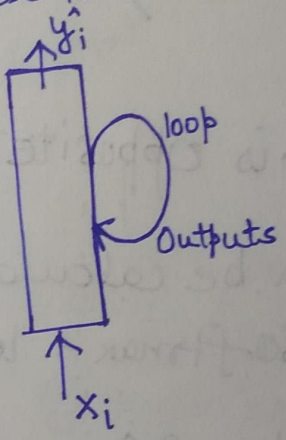
Core structure:-



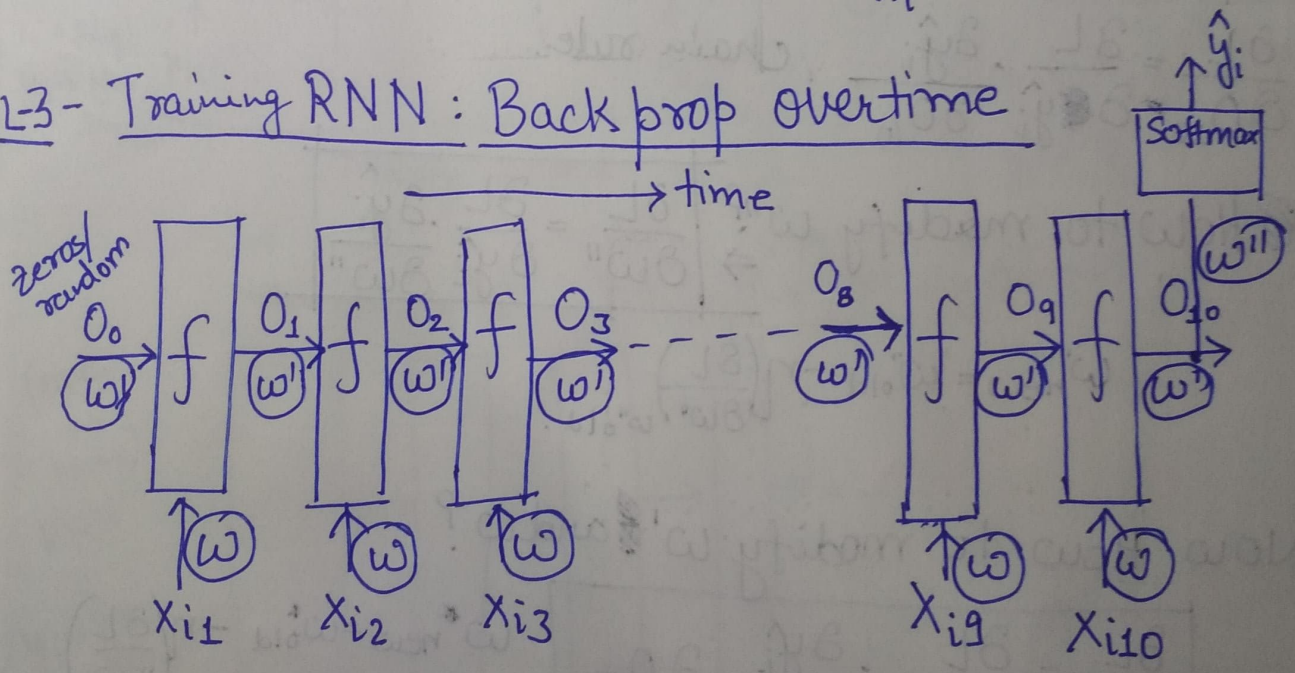
Representation:-



other representation:



L-3 - Training RNN : Backprop overtime



$x_i = \langle x_{i1}, x_{i2}, x_{i3}, x_{i4}, x_{i5}, x_{ic} \text{---} x_{i10} \rangle$

this is just 1 layer repeating overtime. they are not different layers.

$O_1 = f(wx_{i1} + w' O_0)$

$O_2 = f(wx_{i2} + w' O_1)$

$\vdots \qquad \vdots$

$O_{10} = f(wx_{i10} + w' O_9)$

$\hat{y}_i = g(w'' . O_{10})$

forward propogation

But why about back propopogation & change weights?

Backprop is opposite direction of arrows.

① $\frac{\partial L}{\partial \hat{y}_i}$ can be calculated as loss function of Softmax & log-loss is derivative.

② $\frac{\partial L}{\partial O_{10}} = \frac{\partial L}{\partial \hat{y}_i} \cdot \frac{\partial \hat{y}_i}{\partial O_{10}}$, chain rule.

③ How to modify w''? $\Rightarrow \boxed{\frac{\partial L}{\partial w''} = \frac{\partial L}{\partial \hat{y}_i} \cdot \frac{\partial \hat{y}_i}{\partial w''}}$

$$w''_{new} = w''_{old} - \eta \left(\frac{\partial L}{\partial w''}\right)_{w''_{old}}.$$

Now How to modify w' and w?

$$\boxed{\frac{\partial L}{\partial w} = \frac{\partial L}{\partial \hat{y}_i} \cdot \frac{\partial \hat{y}_i}{\partial O_{10}} \cdot \frac{\partial O_{10}}{\partial w}} \qquad w^*_{new} = w^*_{old} - \eta \left(\frac{\partial L}{\partial w}\right)_{w_{old}}$$

similarly $\boxed{\frac{\partial L}{\partial w'} = \frac{\partial L}{\partial \hat{y}_i} \cdot \frac{\partial \hat{y}_i}{\partial O_{10}} \cdot \frac{\partial O_{10}}{\partial w'}}$ $\qquad w'_{new} = w'_{old} - \eta \left(\frac{\partial L}{\partial w'}\right)_{w'_{old}}$

And this has to be done with each x_{ij} $i \in$ dataset $j \in$ length of each point

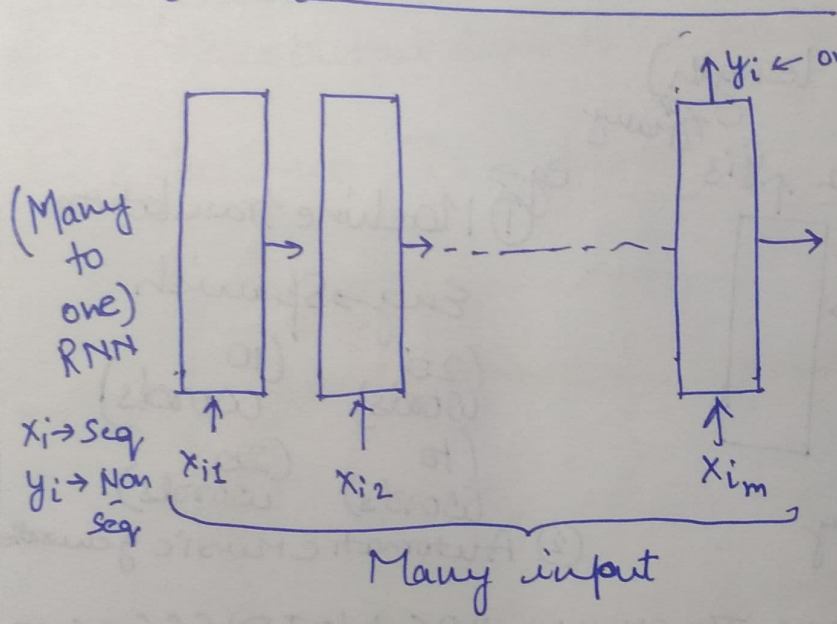
Now if you backprop again for deep network

$\frac{\partial L}{\partial w}$ for 1st word in X_i can cause vanishing gradient & exploxing gradient.

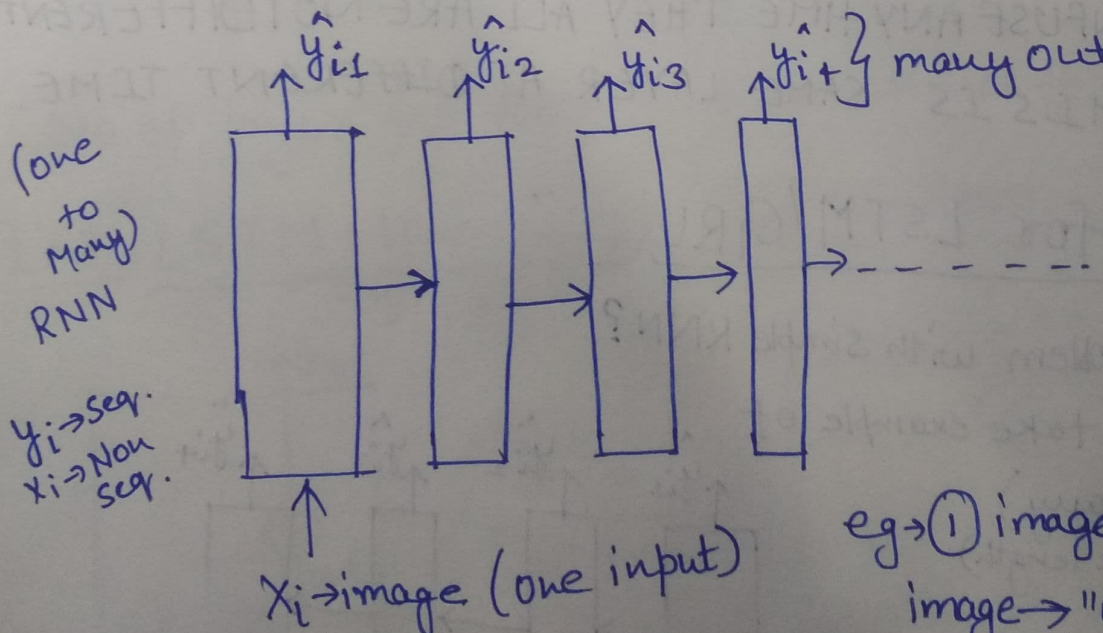
So to avoid this we have special modification Called LSTM's and GRU's.

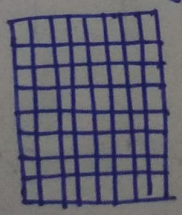
→special RNN's.

L4- Types of RNN Architecture



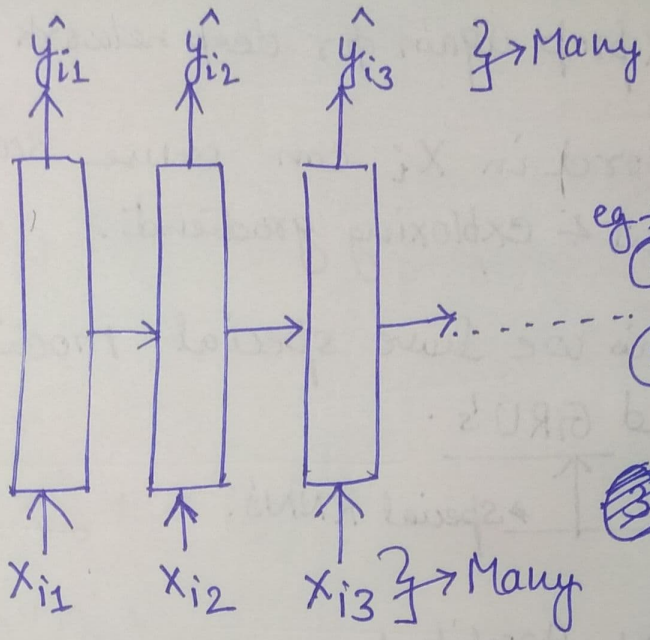
eg→ ① Sentiment Analysis
$y_i \in \{0,1\}$
② movie Review→star Rating
$y_i \in \{1,2,3,4,5\}$
$y_i \in \mathbb{R}$





eg→① image captioning.
image→"what is in the image"

(Many to Many RNN)



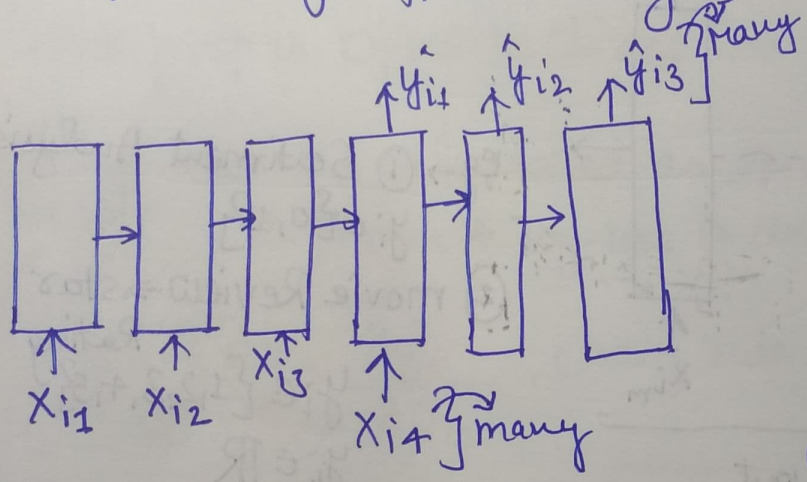
input → seq
Output-seq
Also called (Seq2seq model).

eg → ① input-output same length.

② part of speech (same length) NLP task

~~③ Automatic Music generation.~~

(Many to Many different length)



eg → ① Machine translation

Eng → Spanish

(20 words) (10 words)

(10 words) (20 words)

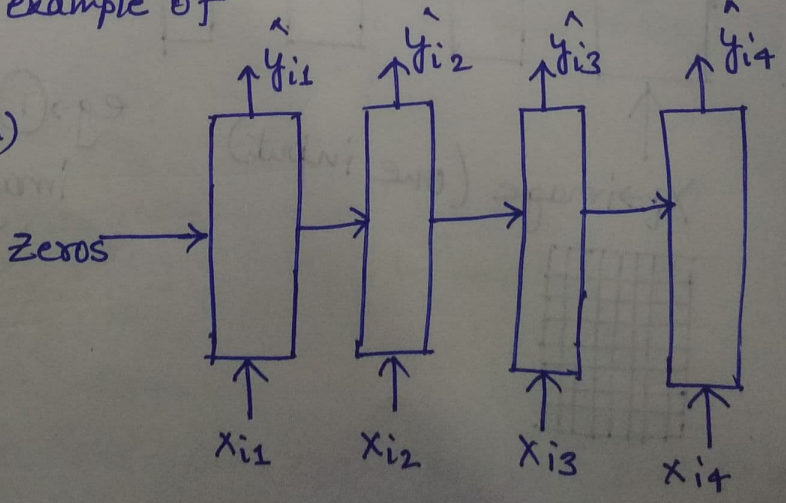
② Automatic Music generation

* DO NOT CONFUSE ANY TIME THEY ALL ARE NOT DIFFERENT LAYER. THIS IS SAME LAYER AT DIFFERENT TIME.

L-5- Need for LSTM/GRU

What is the problem with simple RNN?

In this context take example of (Many to Many) (Same length)

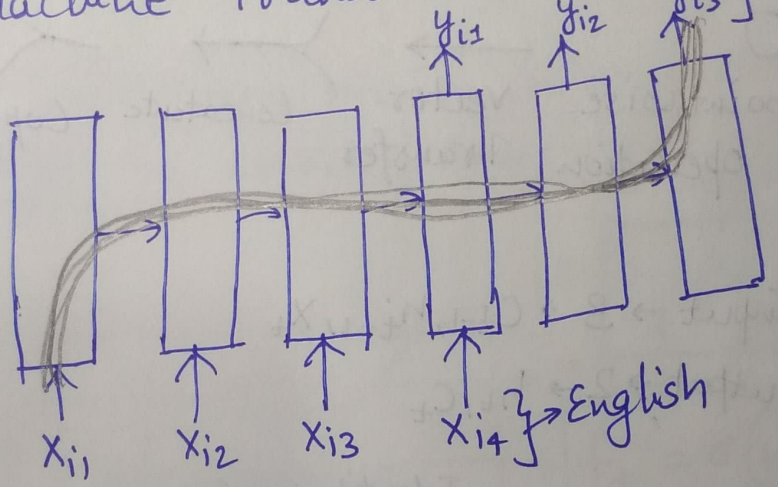


$\hat{y}_{i4}$ depends on x_{i4} & O_3 more in this case.

→ There can be real world applications where $\hat{y}_{i4}$ must depend more on x_{i1} rather than x_{i4}.
Ex → part of speech and in that architecture described before $\hat{y}_{i4}$ & x_{i1} are very far away.

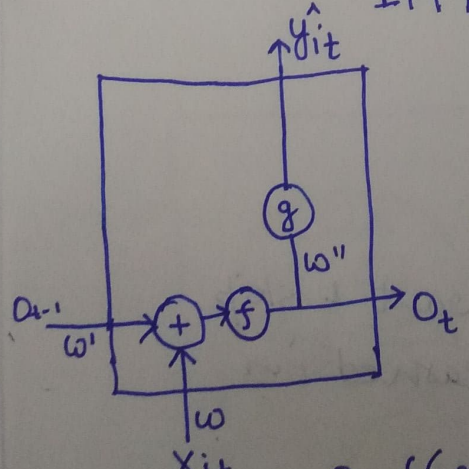
So key problem "Simple RNN's cannot handle long term dependencies".
↳ later output depends on earlier input.

Ex → Machine Translation



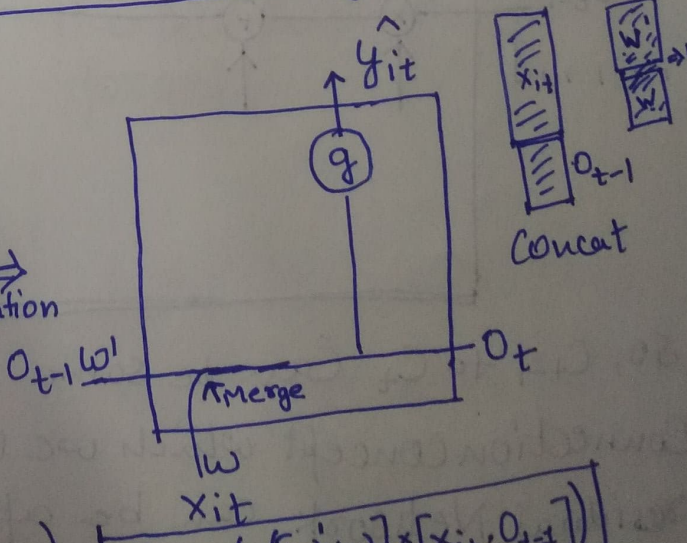
No. of layer far more and dependies got ever far.

L-6 LSTM - Long Short Term Memory
↳ 1997



Simple RNN unit

Basic idea of Modification



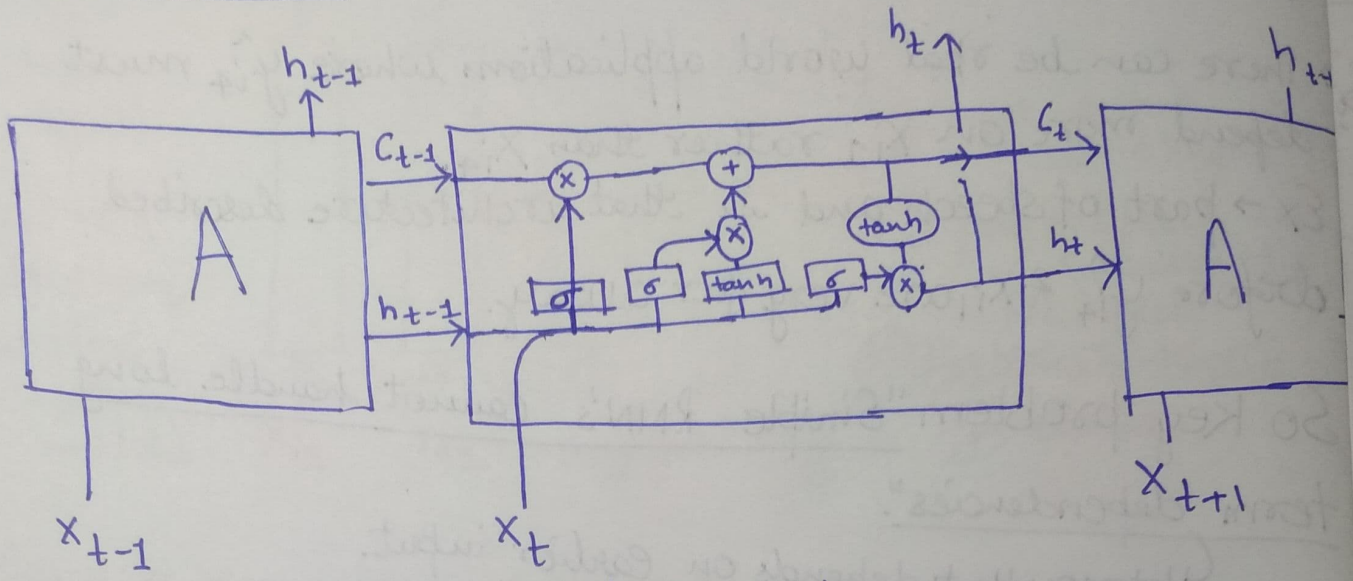
$O_t = f(wx_{it} + w'O_{t-1})$

$\hat{y}_{it} = g(w''O_t)$

$O_t = f([w', w] * [x_{it}, O_{t-1}])$

$y_{it} = g(w''O_t)$

LSTM Unit



LSTM unit

Neural network layer

pointwise operation

vector Transfer

Concatente

Copy

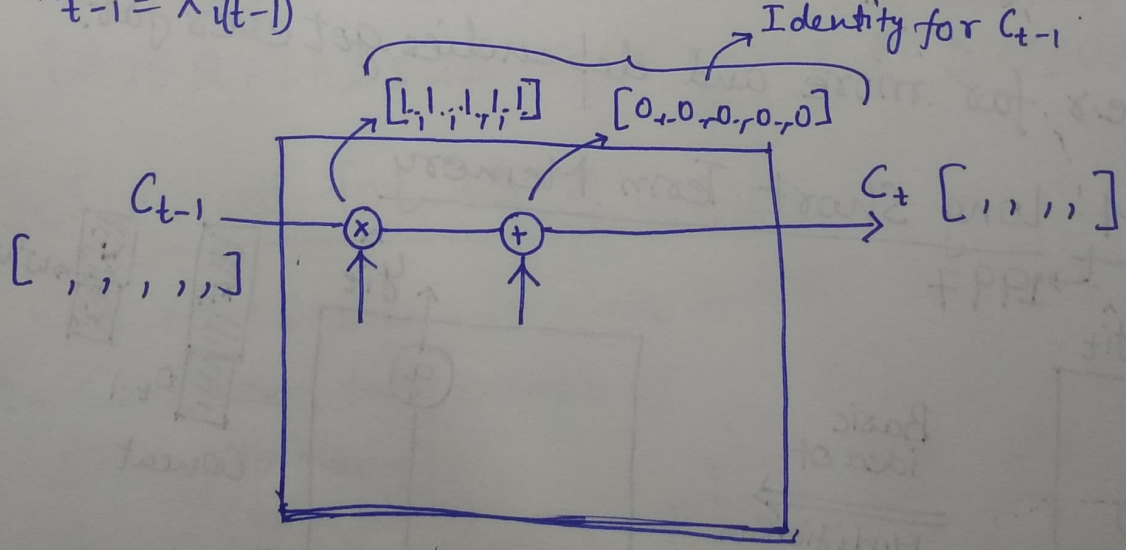
$O_{t-1} = C_{t-1}$

$h_{t-1} = \hat{y}_{t-1}$

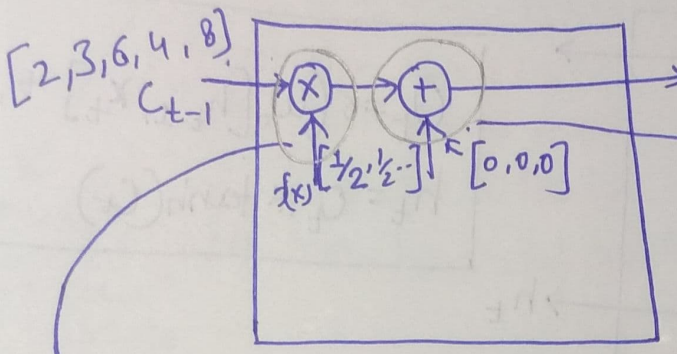
$X_{t-1} = X_{i(t-1)}$

Input → 3 → C_{t-1}, h_{t-1}, X_t

Output → 2 → h_t, C_t



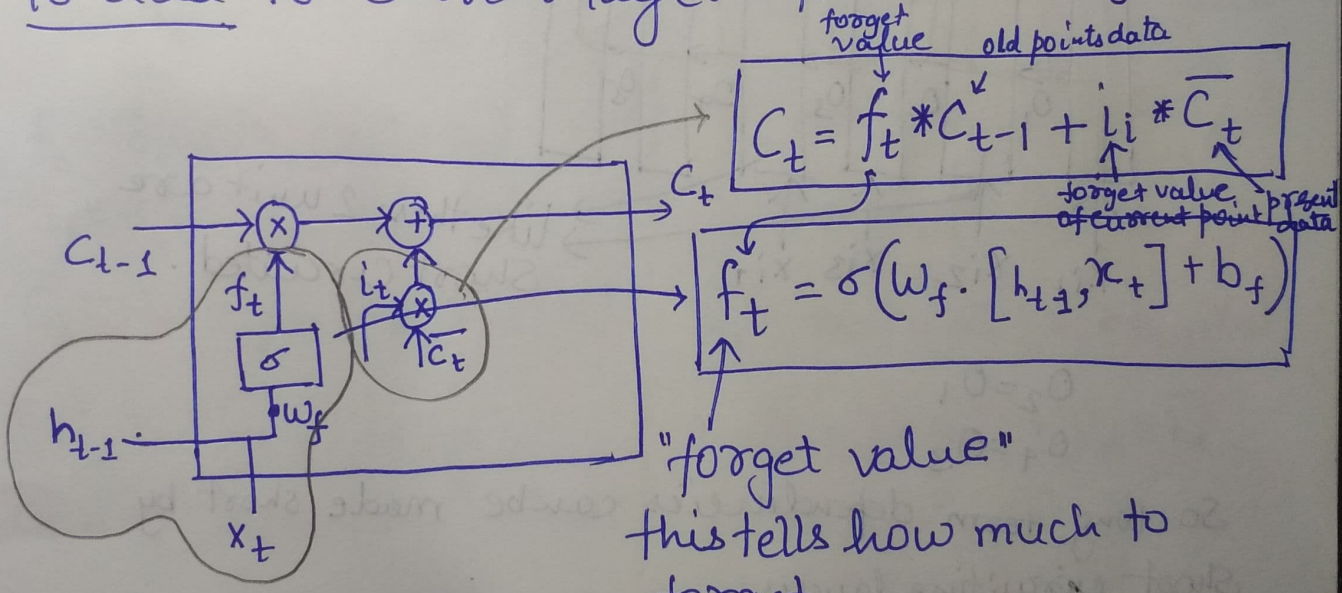
So, C_{t-1} to C_t can be same if we want & skip Connection concept which we have learned in residual Network can be applied.



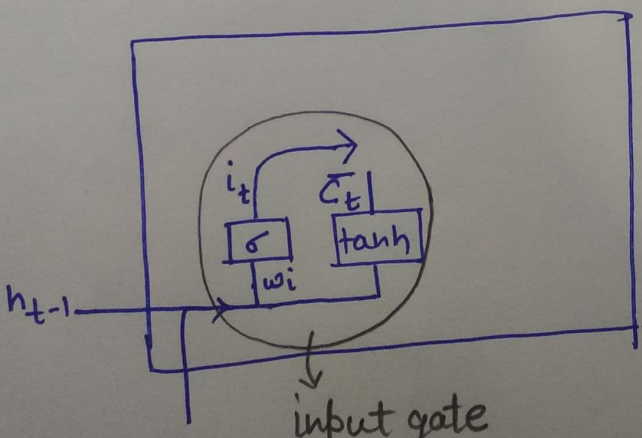
So this this how much of previous layer percent you want take forword for each bit.

How much to remember or forget of each bit also called forget gate.

this tells how much information to add to current layer after some forgets.



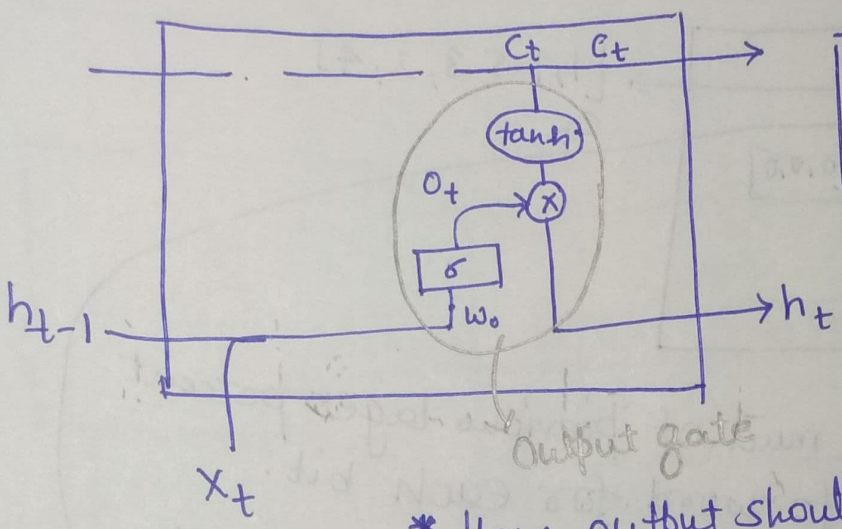
"forget value"
this tells how much to forget.



$$i_t = \sigma(w_i \cdot [h_{t-1}, x_t] + b_i)$$

$$C_t = \tanh(w_c \cdot [h_{t-1}, x_t] + b_c)$$

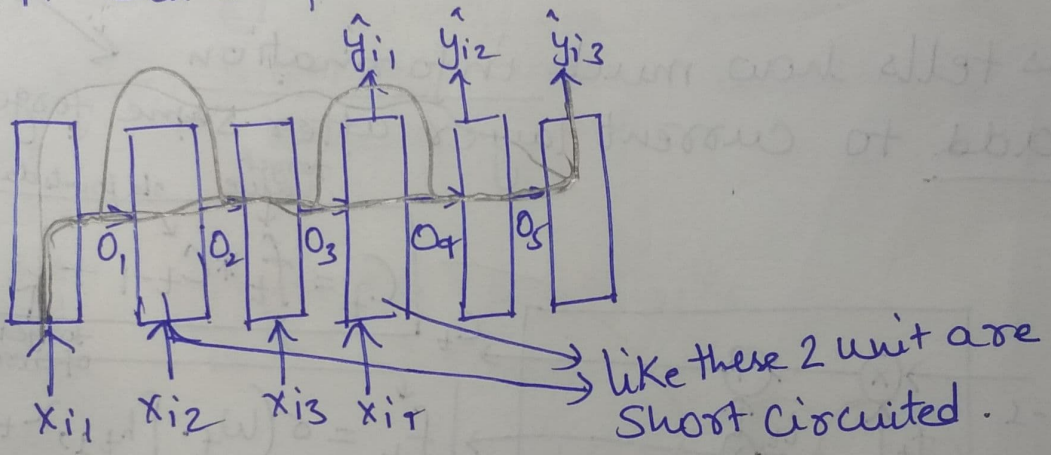
input gate
How much into to take forword of present datapoint to further.



$$O_t = \sigma(W_o[h_{t-1}, x_t] + b$$
$$h_t = O_t * \tanh(C_t)$$

* How output should be generated based on processed input.

* LSTM can help in short circuit connections.



$O_2 = O_1$

$O_4 = O_3$

So long term dependencies can be made short by short circuiting layers.

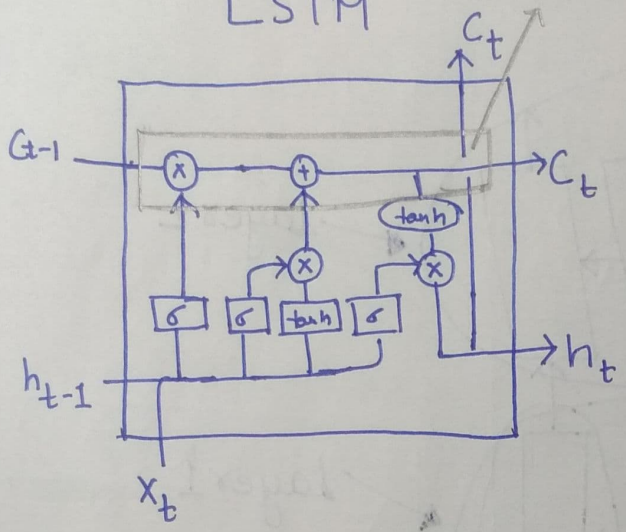
L-7 GRU (Gated Recurrent Unit)

- 2014 → morden than LSTM
  - → Simplified version
  - → faster to train lesser equations.
  - → as powerful as LSTM

Only 2 gates
- reset
- update

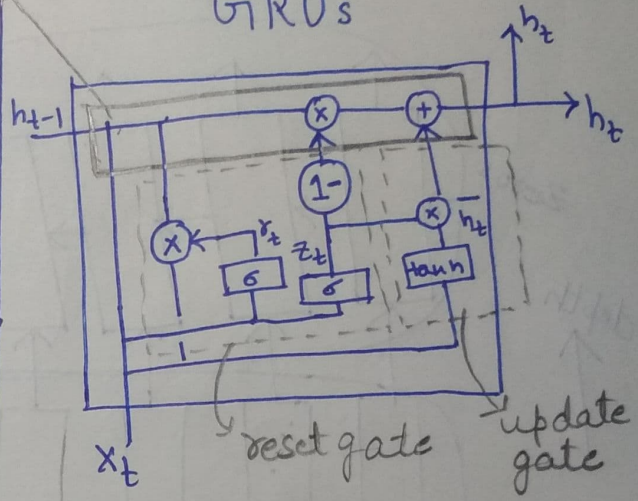
this kind of structure remain same because it helps to short circuit.

LSTM



$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\bar{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$$
$$C_t = f_t * C_{t-1} + i_t * \bar{C}_t$$
$$O_t = \sigma(W_o [h_{t-1}, x_t] + b_o)$$
$$h_t = O_t * \tanh(C_t)$$

GRU's



$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$
$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$
$$\bar{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t])$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \bar{h}_t$$

faster because of lesser equation.

L-8- Deep RNN's

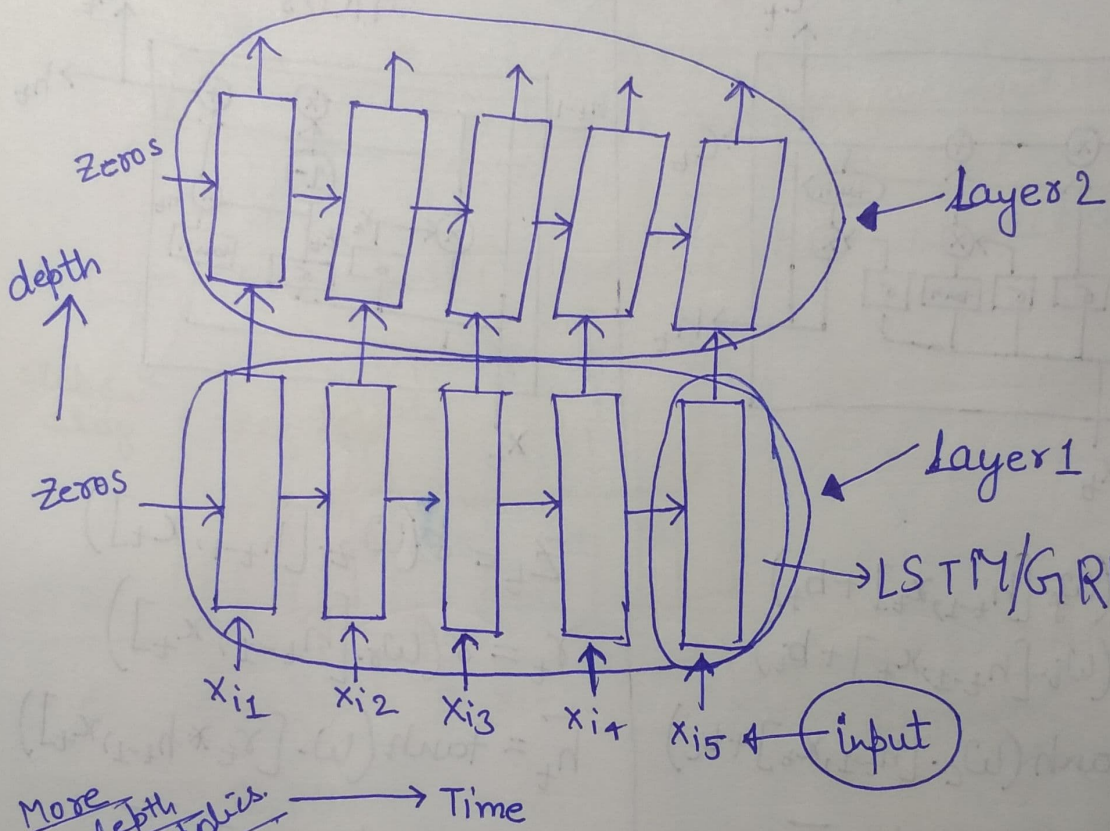
MLP → one layer → Deep MLP

CNN → Conv. layer & Maxpool → Deep CNN

RNN → Deep RNN ... ??

↓

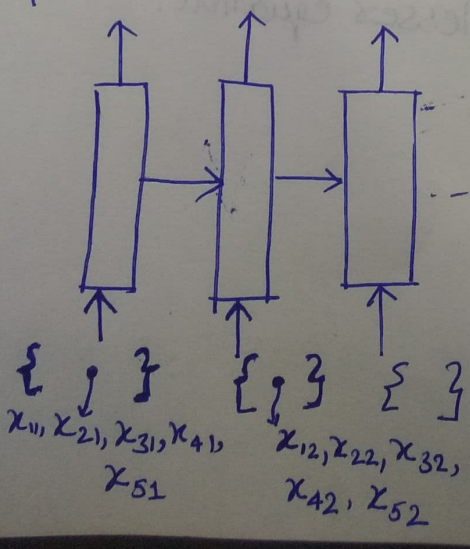
LSTM/GRU



More in depth in case statistics → Time

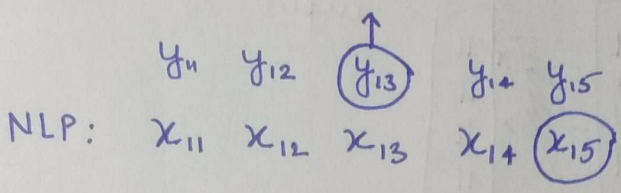
SGD with batch size = k(5) ← 5 points will go simultaneously.

speed up.



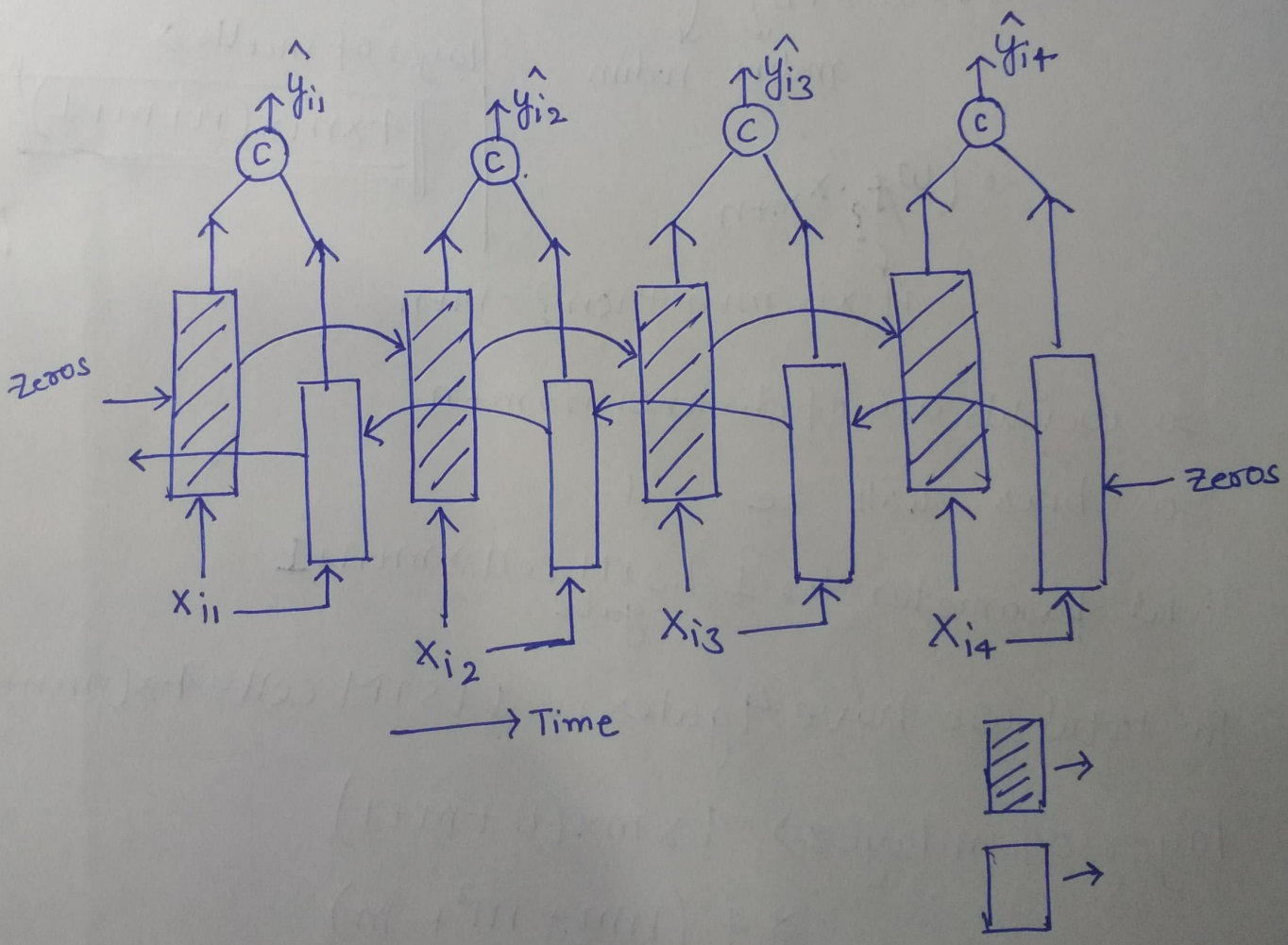
→ But to perform this speedup hack all the points must have same length so for that we can also use the concept of padding.

L-9 Bidirectional RNN

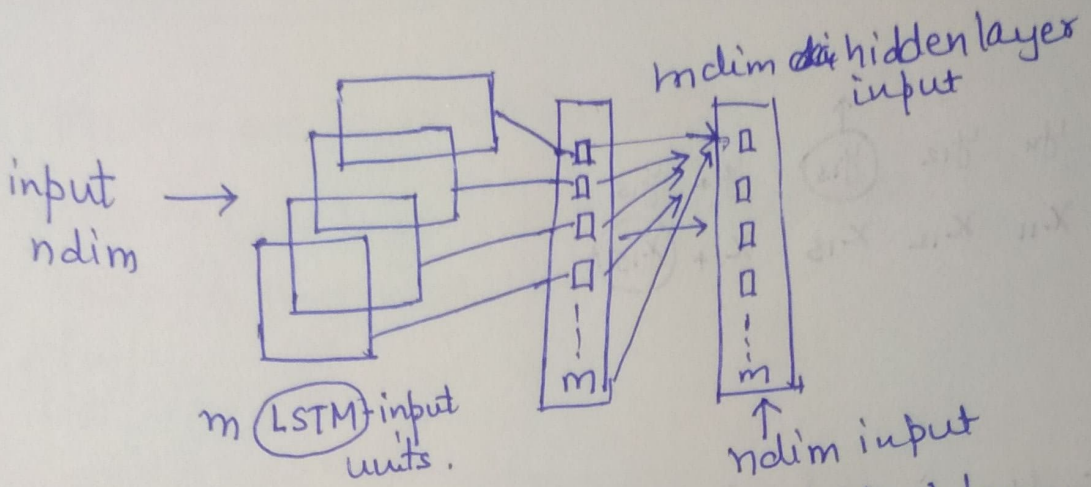


Q. What if y_{13} is dependent on x_{15}

y_{it} is dependent on x_{it+k}
(3) (5)



L-10 Numbers of parameter in LSTM



first lets see number of parameters in first layer 1 unit.

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$

look like $\Rightarrow W^T \cdot X + b$

W_f : mdim, $[h_{t-1}, x_t]$: ndim, b_f : 1

1 cell → 1 gate ⇒ n+m+1

all 4 gate → (n+m+1)4

layer of m cells ⇒

$$4 \times m \times (n+m+1)$$

$\sigma(W_{f_?} \cdot X_{m+n}$

if X is m+n then ? = m+n

So weights are of dimension ⇒ m+n

So bias will be ⇒ 1

Total parameter for 1 LSTM cell ⇒ m+n+1 gate.

In total we have 4 gates in 1 LSTM cell = 4x(m+n+1).

layer of m layer ⇒ $4 \times m \times (n+m+1)$

$\Rightarrow 4(nm + m^2 + m)$

if default Keras implementation bias ie. m is ignored.