Recurrent neural network

Deep learning

ANN ✓ CNN ✓ RNN

RNN

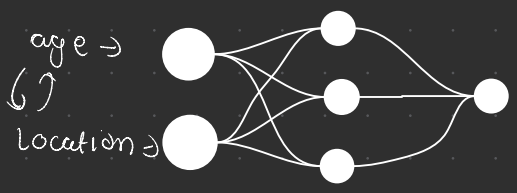
Recurrent nural network

This RNN works on sequential data and creates a sequential model.

Sequential Data

age	Location	Salary
-	-	-
-	-	-

→ this is non sequential data



Text

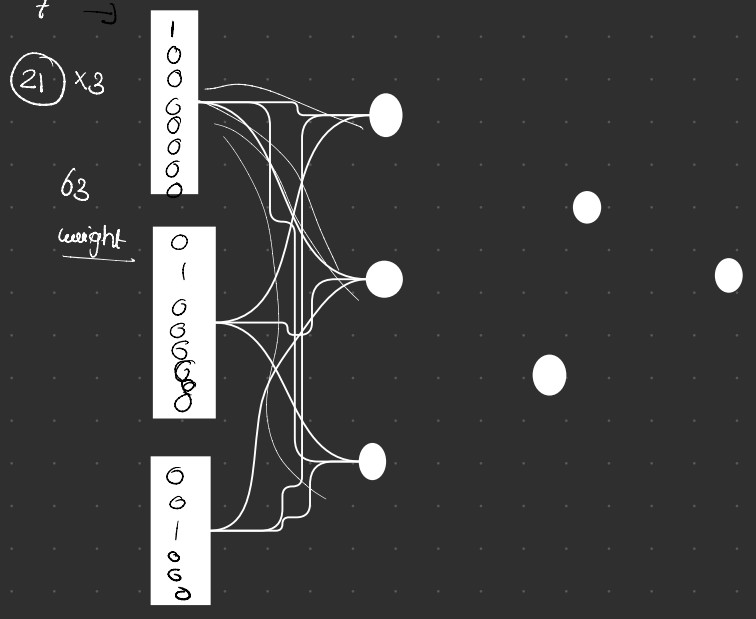
→ hello Subscribe to SAS

when we have textual data we will use sequential model or RNN

why use RNN instead of ANN

1) → (I love sheryians)
2) → sheryians Provide quality content
3) → I hate sheryians

1
1
0



OHE

Vocabulary size → 7 words

Sentence 1 → [[1,0,0,0,0,0,0],
[0,1,0,0,0,0,0],
[0,0,1,0,0,0,0]]

Sentence 2 → [ ]

1. Text input size may vary
2. Computational Power
3. Prediction Problem → Input → Testing input
4. Semantic meaning - Sequence is not retained

RNN Architecture

→ Data feeding

	comments	sentiment
In 1 →	you are good	1
	you are Bad	0
	you are not good	0

vocabulary = 5

you `[1,0,0,0,0]` are `[0,1,0,0,0]` good `[0,0,1,0,0]`

Bad `[0,0,0,1,0]` not `[0,0,0,0,1]`

first Review

`[[1,0,0,0,0], [0,1,0,0,0], [0,0,1,0,0]]`

Timestamp = 3

Input features = 6

Keras → (batch size, timestamp, input features)

(3, 4, 6)

→ RNN Architecture

	comments	sentiment
In 1 →	you are good	1
	you are Bad	0
	you are not good	0

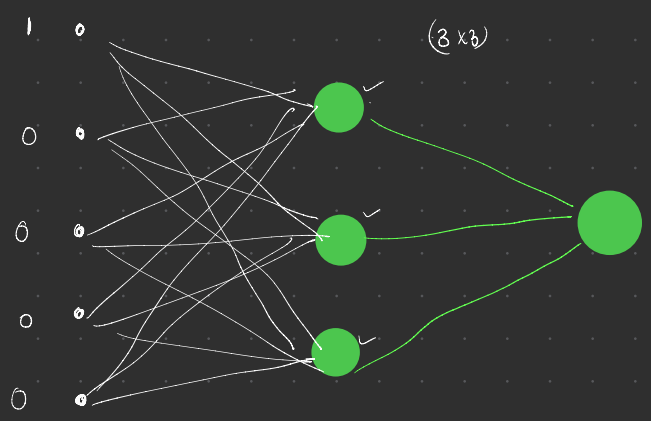
vocabulary = 5

you `[1,0,0,0,0]` are `[0,1,0,0,0]` good `[0,0,1,0,0]`

Bad `[0,0,0,1,0]` not `[0,0,0,0,1]`

Feed forward Network

Feedback Network
Recurrent



(1,3)

(3x3)

you + are + good

↳ is ↳ is

zeros

3 times terms

18 weights + 9 weights = 27 weights

18 weights

$5 \times 3 = 15$ weights + 3 weights

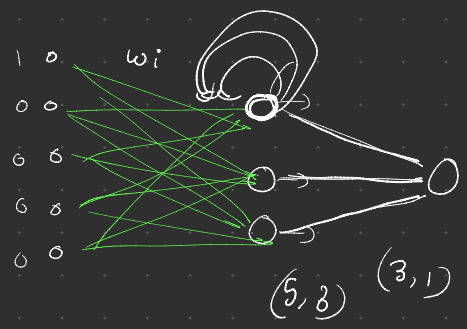
$= (3 \times 3) = 9$

$= 27$ weights

you are good

T=1 you → [1,0,0,0,0]

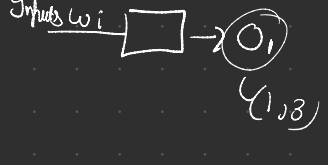
↳ (1,5)



Recurrent neural network have an activation function inside their hidden n default (tanh)

$\phi(you_i \times w_i + b_i)$

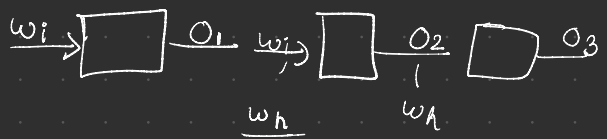
$((1,5) \times (5,3)) = (1,3)$



T2

are

↳ [0,1,0,0,0]



$\phi((are(i) \times w_i) + (w_h O_1) + b)$

$(1,5) \times (5,3) + [(3,3) + 1,3)] + b)$

$(1,3) + (1,3) \rightarrow (1,3)$

$T_3 \rightarrow f(good(i) \times w_i) + (w_h O_2) + b)$

good $\Rightarrow (1,3)$

$(1,3)\ (3,1)$

$= (1,1) \rightarrow$

Backpropogation

BPTT ⇒ Backpropogation through time

text

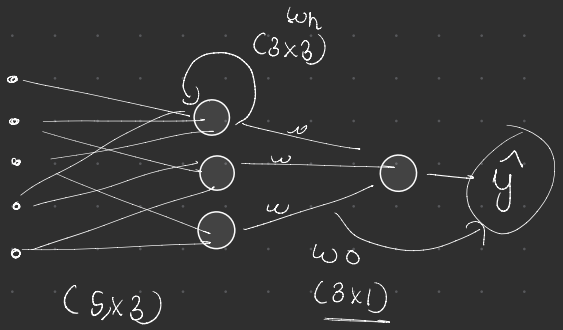
Sheryians	coding	school	1
Sheryian	AI	School	1
Sheryians	cyber	school	0

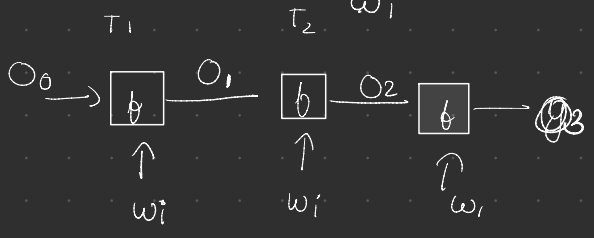
[10000] [01000] [00100]

[10000] [00010] [00100]

[10000] [00001] [00100]

[Sheryians coding School AI cyber]





$O_1 = f(x_{i1} \times w_i + O_0 w_h) + b$

$O_2 = f(x_{i2} w_i + O_1 w_h) + b$

$O_3 = f(x_{i3} w_i + O_2 w_h) + b$

$\hat{y} = \sigma(O_3 w_o)$

$y - \hat{y}$ ← Loss function

$= -y_i \log(\hat{y}_i) - (1-y_i) \log(1-\hat{y}_i)$ - Loss

The main motive is to minimize the loss

So we will use gradient Descent. → Optimizer

↓

to optimize the weights

w_i (6×3) w_0 (3×1) w_h (3×3)

18 + 3 + 9 = 27

$w_i = w_i - \eta \frac{\partial L}{\partial w_i}$

$w_h = w_h - \eta \frac{\partial L}{\partial w_h}$

$w_0 = w_0 - \eta \left[\frac{\partial L}{\partial w_0}\right] \longrightarrow \frac{\partial L}{\partial w_0} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial w_0}$

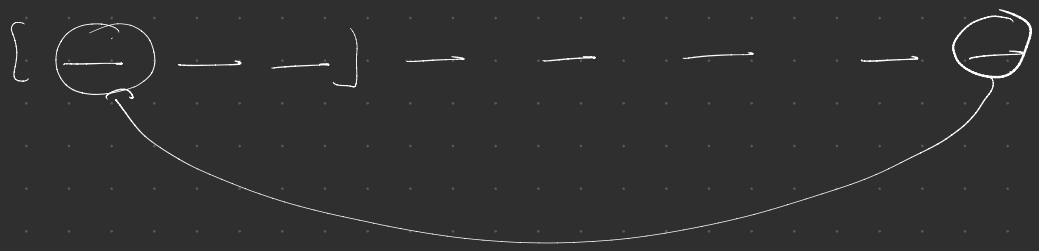
[I love this Product]

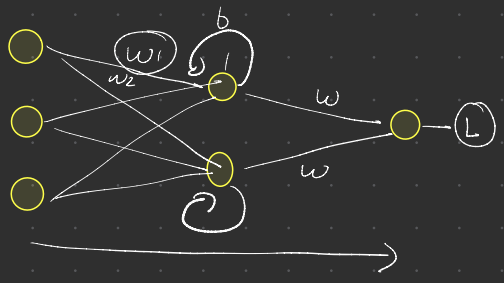
[[ ] [ ] [ ] [ ]] – T4

[5, 17, 9, 10, 0, 0, 0, 0] → [[0.21, 0.77, -0.83]
[
[
]
]
]

Problems with RNN

1. Long term memory is weak.
2. Vanishing Gradient problem
3. Exploding Grading
4. No long dependencies are getting captured
5. Sequential process is slow.



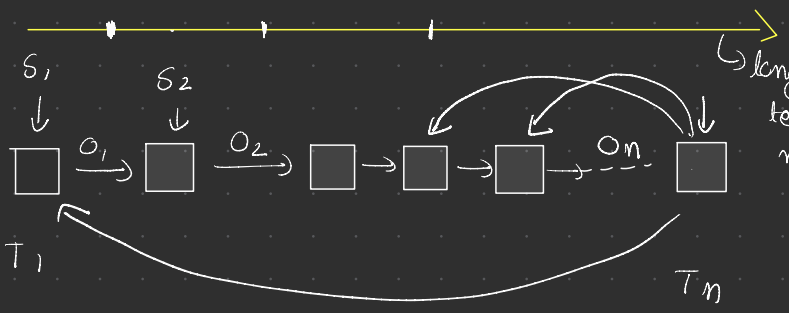


$$w_{new} = w_{old} - \eta \left(\frac{\partial loss}{\partial w_{old}} \right)$$

$$\frac{\partial loss}{\partial w_{old}} = \frac{\partial loss}{\partial o_3} \times \frac{\partial o_3}{\partial o_2} \times \frac{\partial o_2}{\partial o_1} \times \frac{\partial o_1}{\partial w_{old}}$$

We don't use simple RNNs much today because they forget information very fast and can't handle long sentences or long-term patterns the gradients either vanish or explode. LSTM and GRU were invented to solve exactly this problem. They have special gates that decide what to remember and what to forget, so they can learn long-range dependencies, capture context better, and train more stably. In simple words: **RNN gets confused in long sequences, but LSTM/ GRU remember things smartly**, which is why we prefer them now.

LSTM
(Long short term memory)



Whatsapp message

2000 words

Rahul lives in Delhi

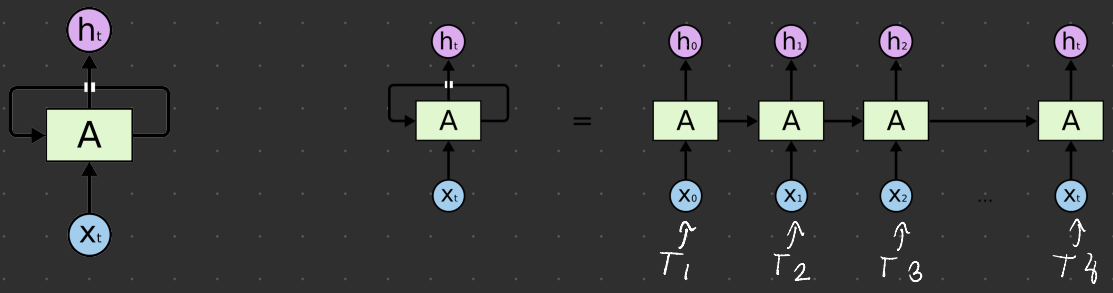
where Rahul lives?

RNN X → (Short term memory)

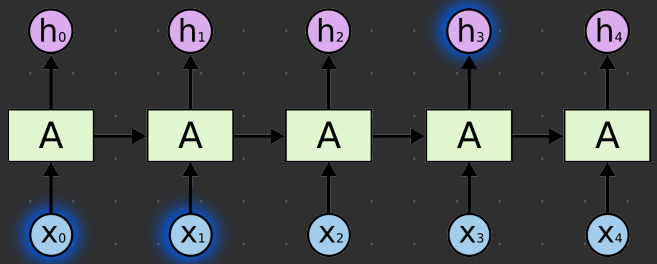
LSTM → long term memory
Short term memory

Architecture

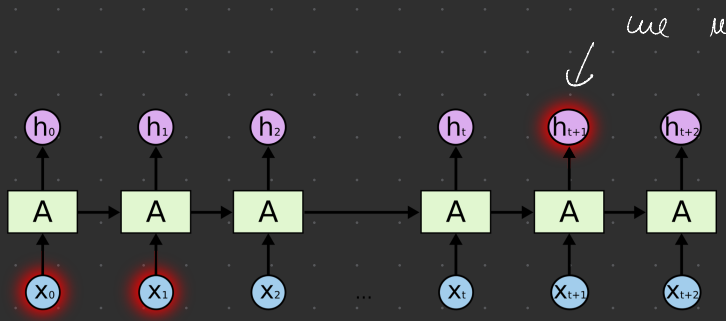
RNN



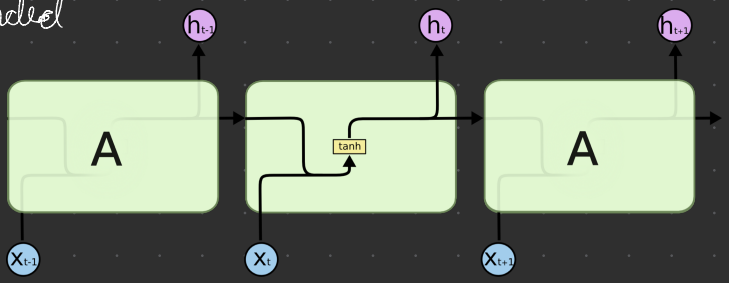
(1) The clouds are in the sky .



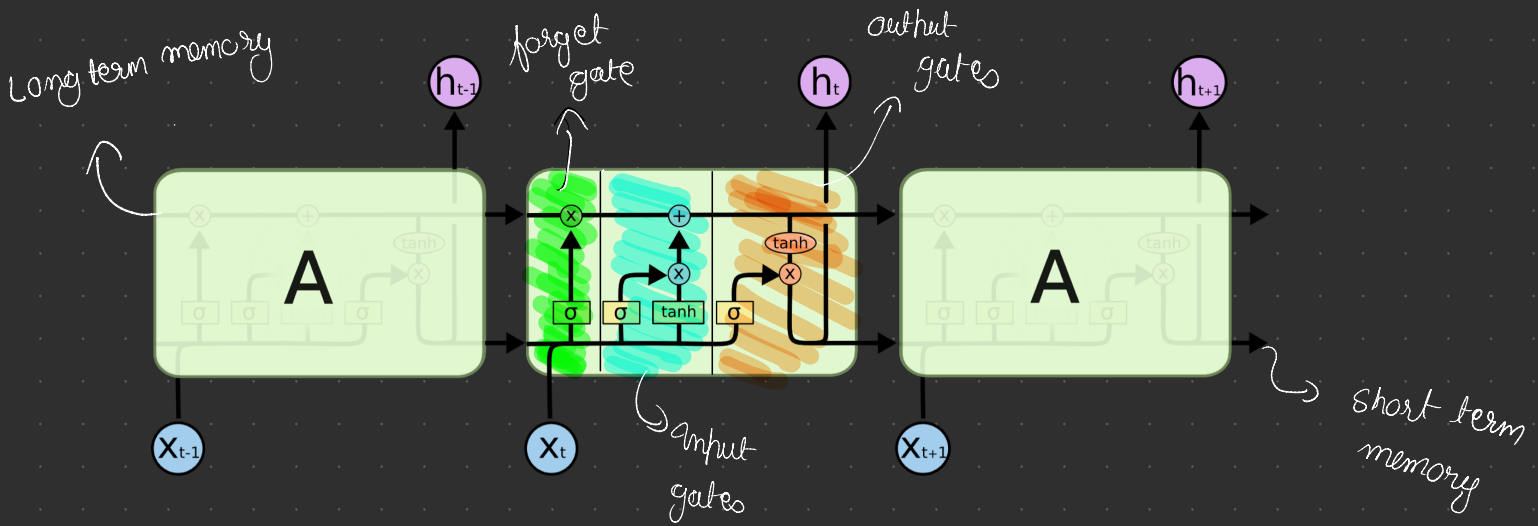
② long sentences.



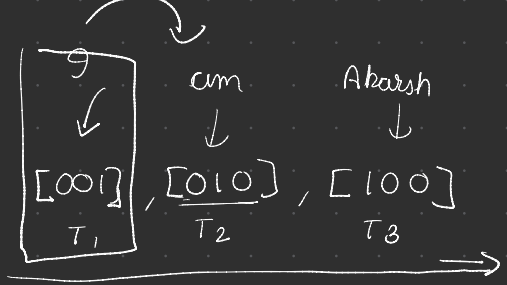
RNN more expanded Version

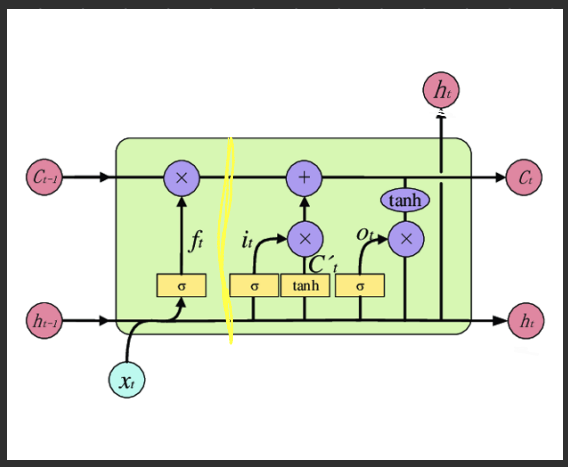


LSTM → Arch.



Understanding Gates





T2

C_{t-1} - Previous cellstate → long term memory

h_{t-1} → Previous hiddenstate → short term memory

x_t → Input at t_2

h_t → current hidden state

C_T → current cell state

① forget gate.

Riya is a doctor.
She lives in Delhi.
She works at a hospital.
Last year, Riya moved to London.
Now she works at a research lab

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

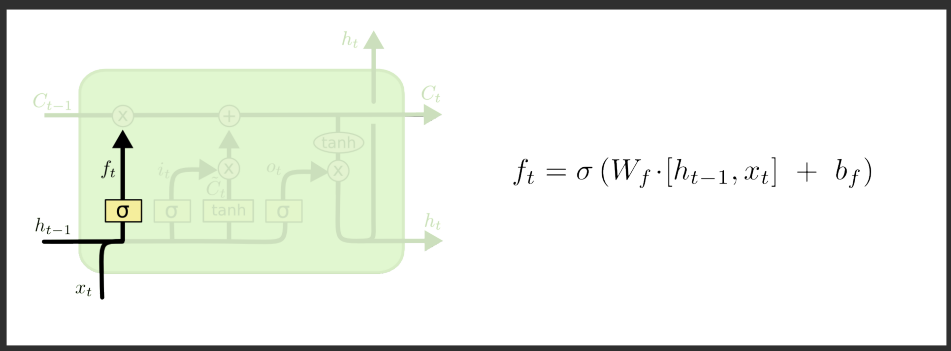
Gender → 0.99

Profession → 0.95

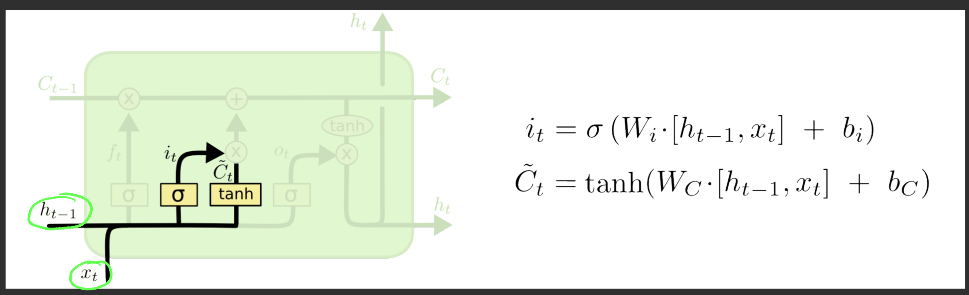
~~Delhi → 0.10~~

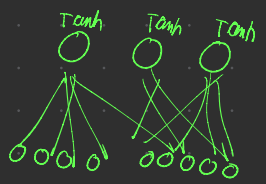
Hospital → 0.85

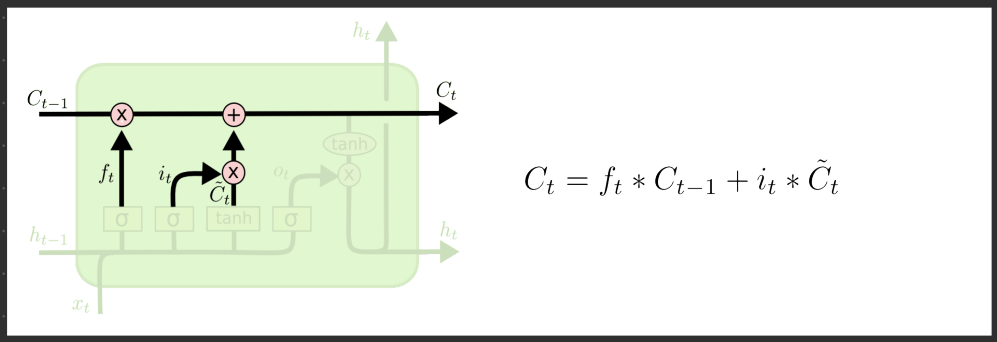
London → 0.90



② Input gate.







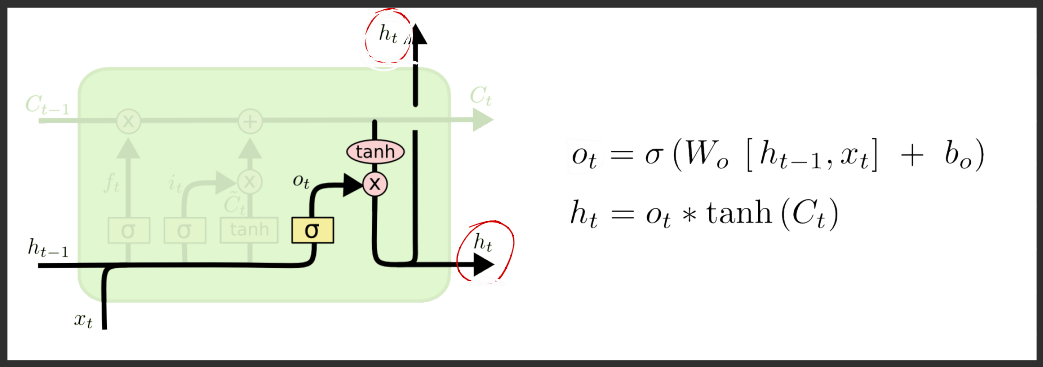
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

$\tilde{C}t$ → candidate cell state → filter out what we should add in cell state.

it → 2nd filtering

Ct → cell state

(3) output gates.

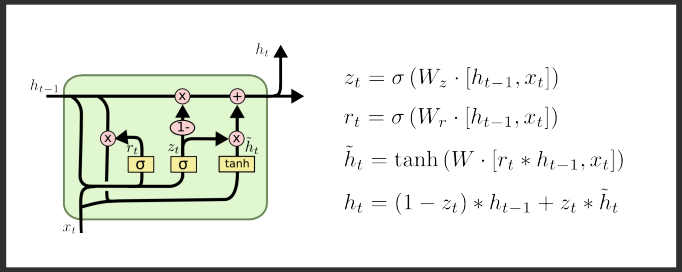


$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Ct → ht → tanh (Ct) → [-1, 1]

Ot =

GRU (Gated Recurrent unit)

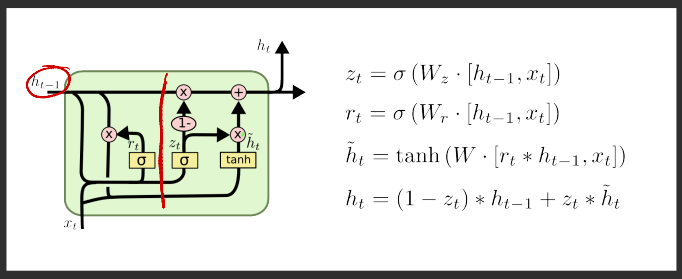


$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Problems with lstm

1. Complex Architecture
2. training Parameters
3. time complexity ↑

GRU → LSTM had 3 gates GRU has 2 Gates



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

[0, 1]

↑

σ - Sigmoid Activation function.

tanh → tanh Activation function

} neural network layers

[-1, 1]

h_{t-1} → Previous hidden State

x_t → Input current

h_t → current hidden state

r_t → reset Gate

z_t → Update gate

$\tilde{h}_t$ → candidate hidden state

(X) → element wise multiplication.

(+) → element wise addition

(1-) →

(X) → [1, 2, 3], [3, 4, 5]

= [3, 8, 15]

Riya lives in Delhi

Riya moved to London

↓ | |

T1 T2 T3

[100] [010] [001]

[ht-1] = [0.65, 0.30]

① xt = [1 0 0]

[1, 0, 0, 0.65, 0.30]

②, ③ → Update gate (do I keep the old memory or update it)

zt = [0.9, 0.9] - high value

(Keep the Past)

T3 → [0.1, 0.1] → low val

(Update the Past) → Delhi X → London.

Reset gate (how much past content should influence the new info)

① GRU creates new info. from the word
② mix it with old memory
③ mixing Ratio is decided by the Update Gate