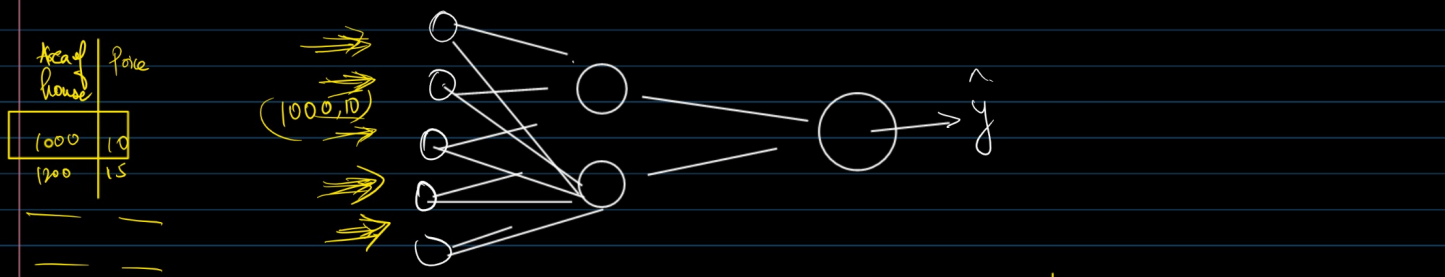
NLP

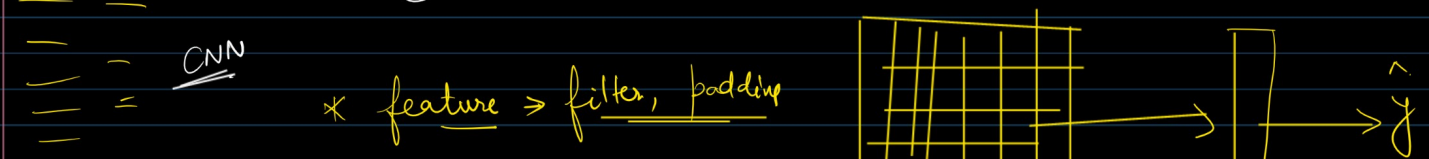
Word / sentence → Vector representation

RNN – LSTM – GRU

ANN:



CNN



* feature → filter, padding

Why RNN is needed? → for a sequential data, you can't ANN or CNN

↓

Performance will be low.

"Cat sits on Mat"

→ RNN performs very well for sequential

→ Sequence has a meaning

"India will win the match"

"match will the India win"

Review	Sentiment
movie was okay	0
movie was bad	0
movie was good	1

↓

tokenization

↓

Word2vec / glove

movies was okay [[1 00 00], [0 1 0 00], [0 0 1 00]]

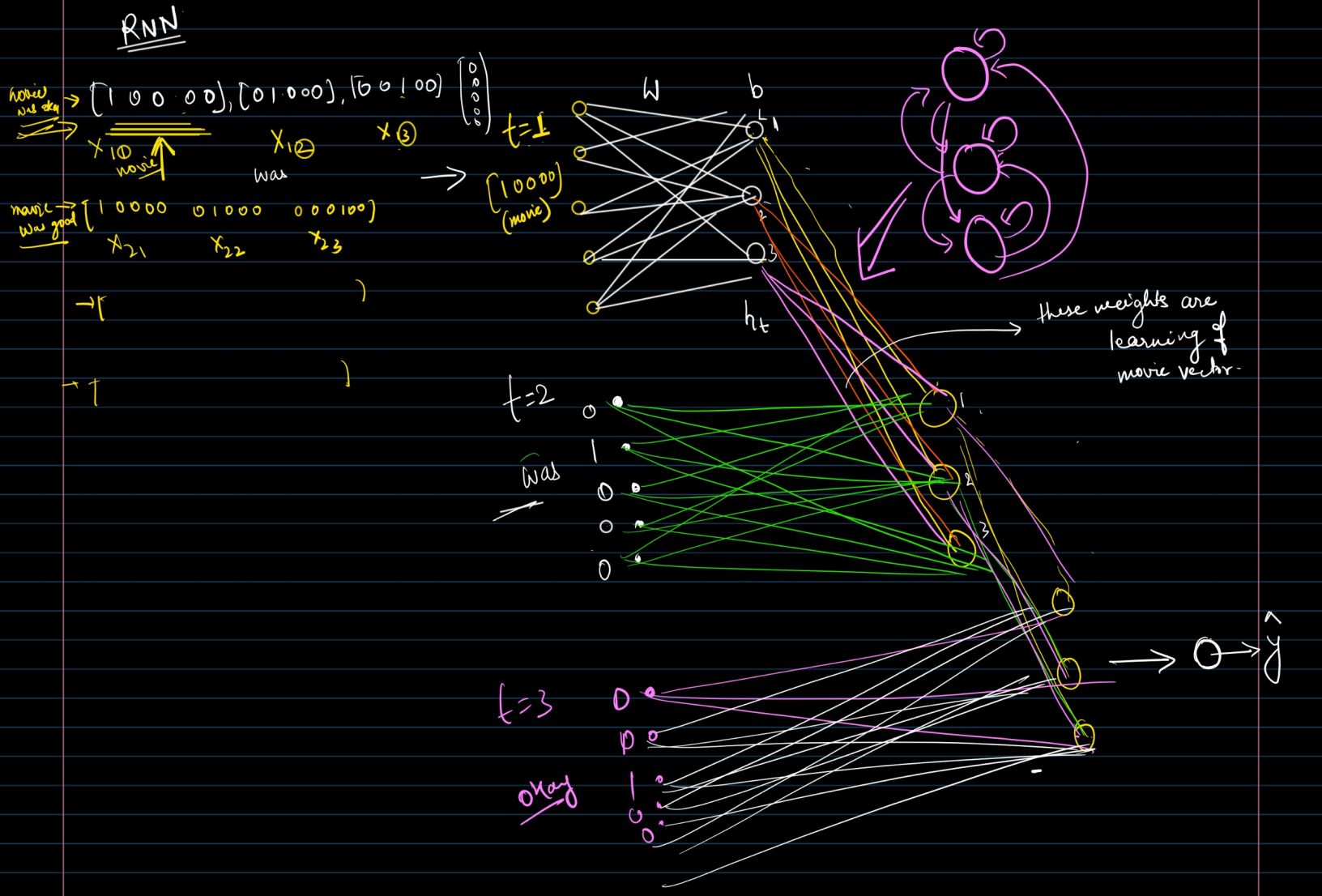
movies → movie

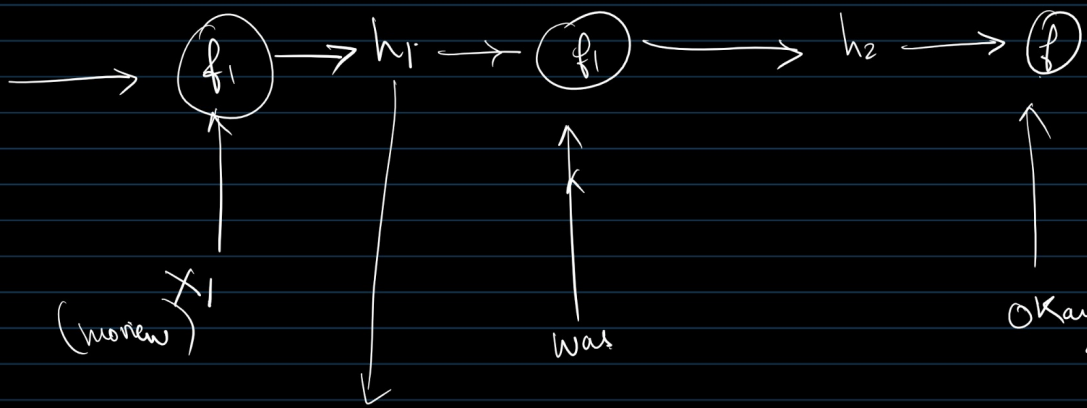
was → was

okay → okay

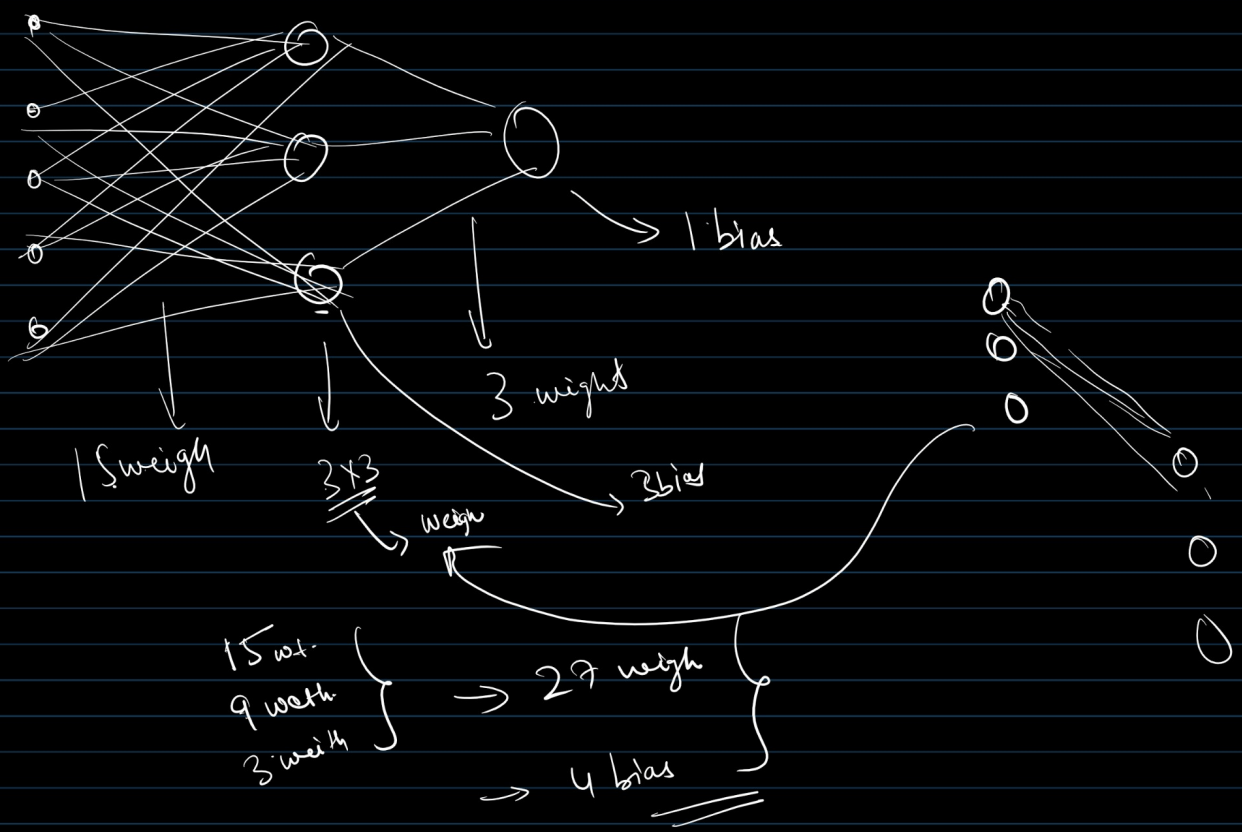
	movies	was	okay	bad	good
movie	1	0	0	0	0
was	0	1	0	0	0
okay	0	0	1	0	0

RNN





$$h_t = \tanh\left(W_{hh} \cdot h_{t-1} + W_{xh} X_t\right)$$



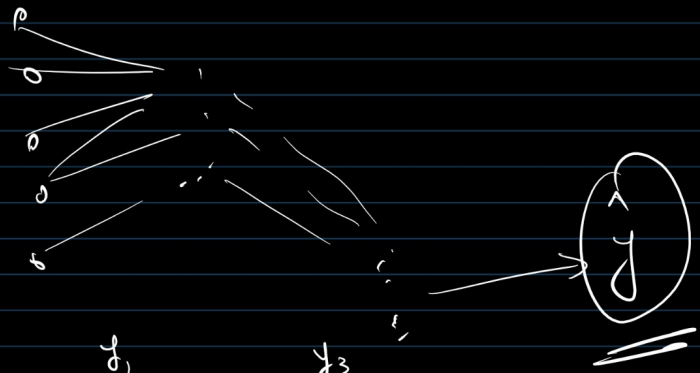
Types of RNN

① many to one

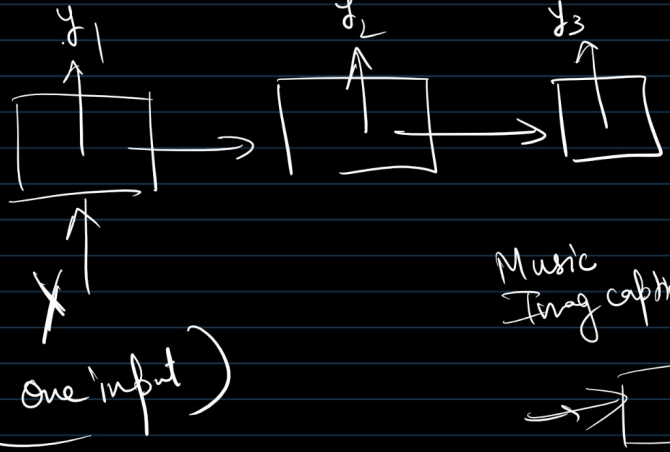
movie was okay

input

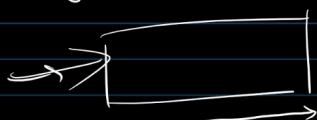
output



One to Many

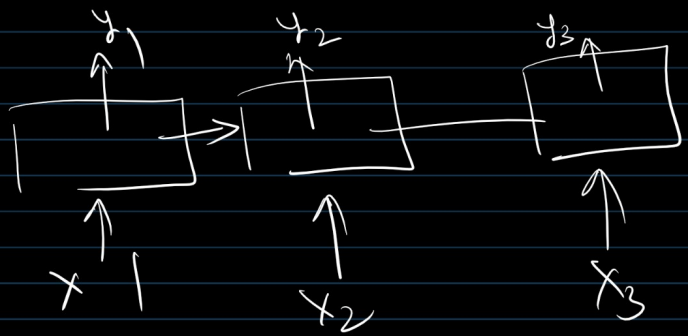


Music
Image captioning

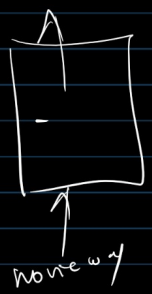


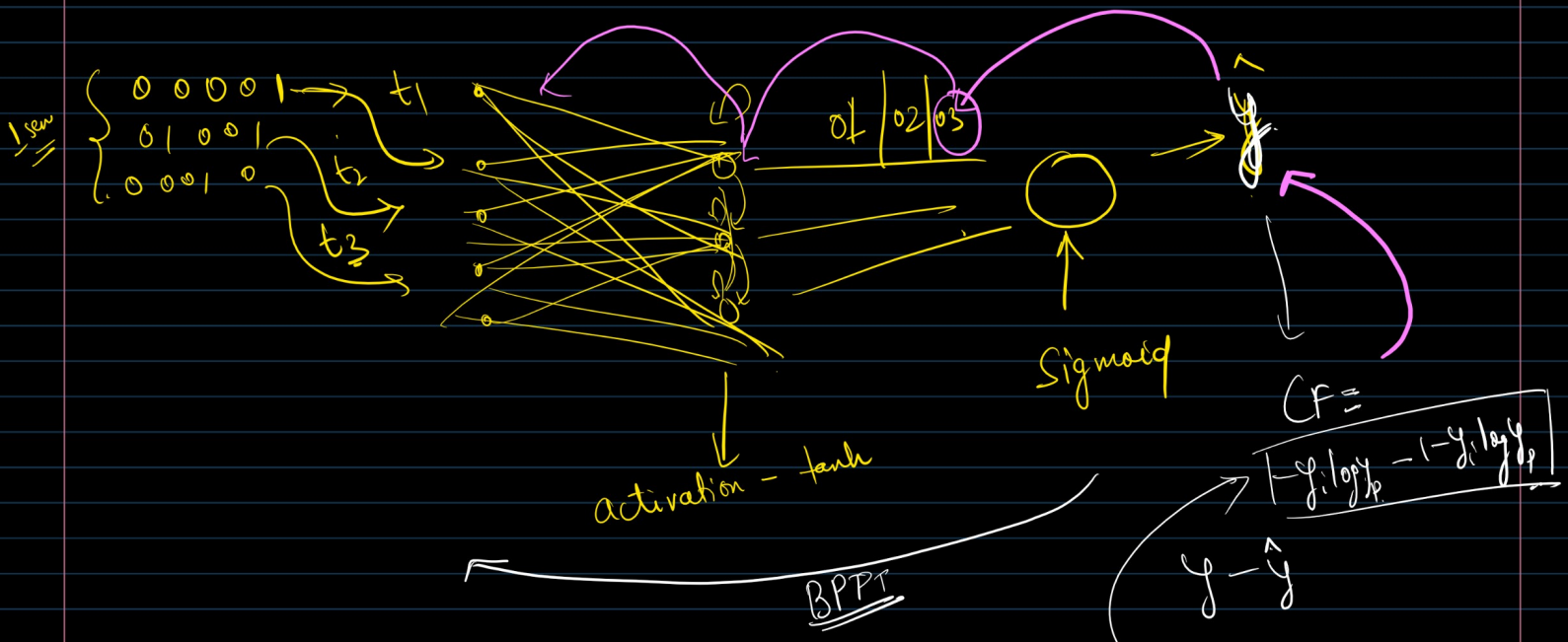
The donkey is silly

Many to Many



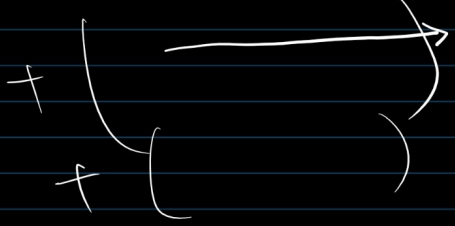
One to one





$$\theta_0 = \eta \frac{\partial L}{\partial \theta_0}$$

$$\frac{\partial L}{\partial w} = \left(\frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_3} \cdot \frac{\partial o_3}{\partial w_i} \right)$$ ← one word (movies)

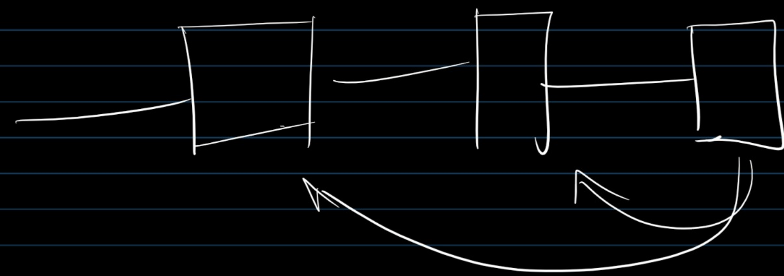


3 word → 3 eqn

10 word → 10 eqn

Problem with RNN

↳ long term dependency is not captured (vanishing gradient)



Ram is going to delhi. Delhi is a good place. He wants to visit Qutub Minar.

* The gradient contribution of previous words keeps decreasing.

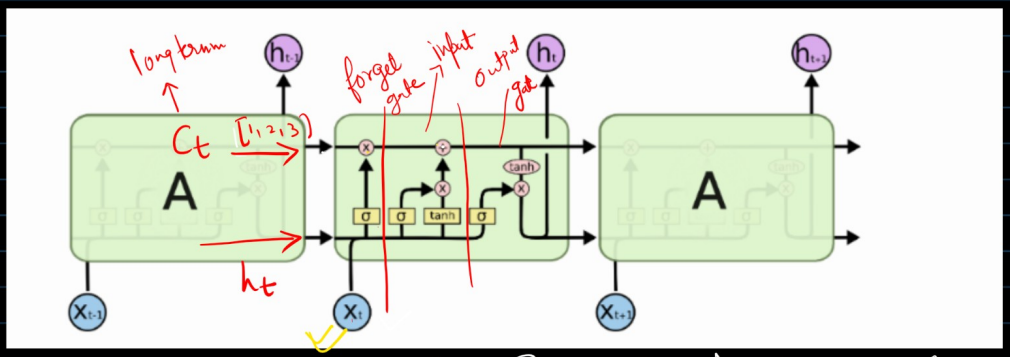
L S T M

↓ Long ↓ Short ↓ term ↓ Memory

Mon Kakd

✓ Kakuda went to delhi.
He met Monjolika.
Monjolika is beautiful.
She loves biryani.
Kakuda love biryani
both love each other.

LSTM architecture



forget
↳ LSTM में क्या रहेगा क्या नहीं रहेगा

$X = [1, 2, 3] \rightarrow \sigma(1,2,3) \Rightarrow [0.2, 0.3, 0.1]$

Input → based on current input, what should be added in long term

[1 2 3]

[0.2, 0.6, 0.3]

Output → what h_t will be passed for next time step

LTM + STM

(C_t) LTM

STM

$h_t)$

X_t

LTM — cell state

STM — hidden state

update → LTM

LSTM

Forget + Input + Output gate

Forget → To remove something from LTM

Input → To add to LTM

Output gate → Output for next hidden state

Use case

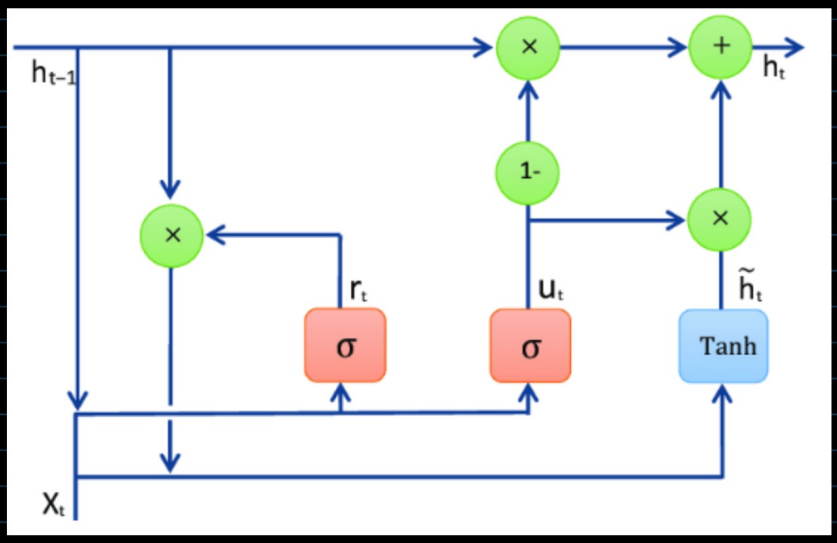
→ ~~Next~~ word prediction

→ translation

GRU

GRU → Gated Recurrent Unit → RNN architecture

LSTM & GRU → Simplified Architecture of LSTM

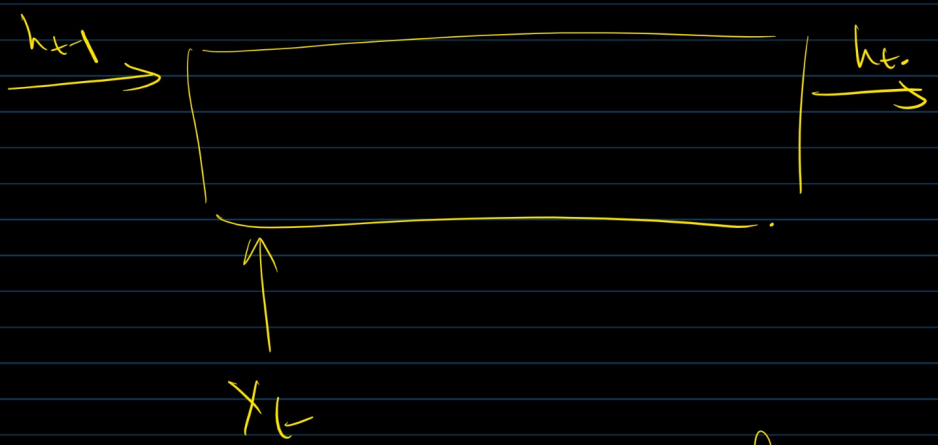


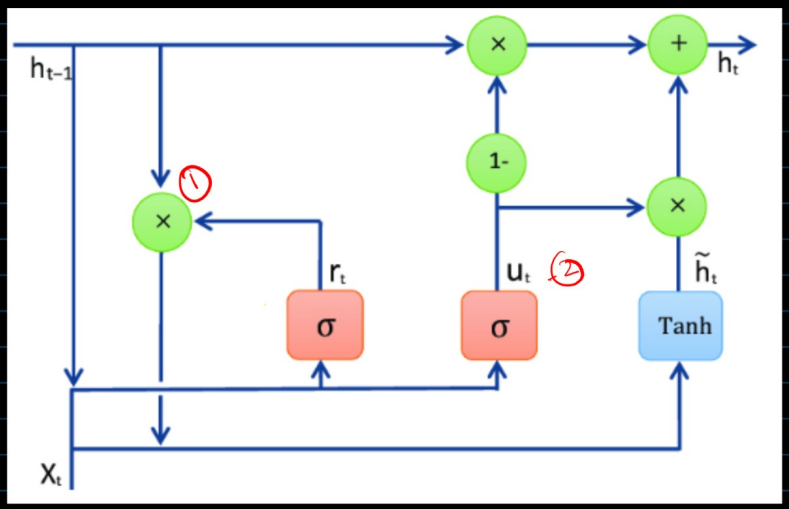
→ Simple version of RNN

→ less no of parameters as compared to LSTM → training time

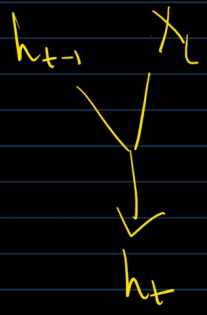
In LSTM → Long term state + Short term state → separate

GRU → both lougth & Short term → hidden state





Goal ⇒ for any time stamp t, we give two inputs



h_{t-1} → previous hidden state

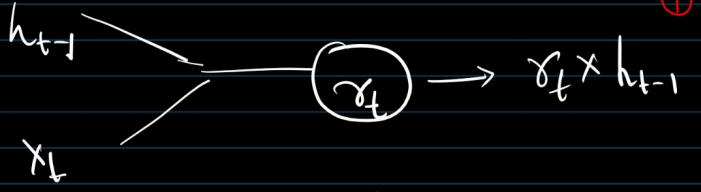
h_t → current hidden state

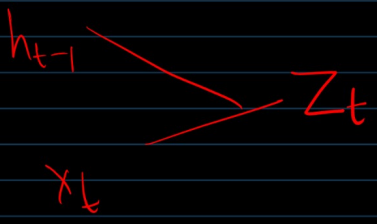
X_t → input

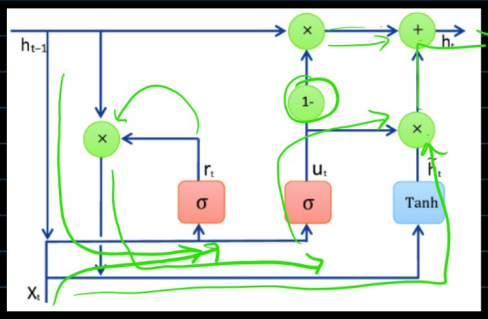
r_t → Reset gate

u_t / z_t → update gate

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

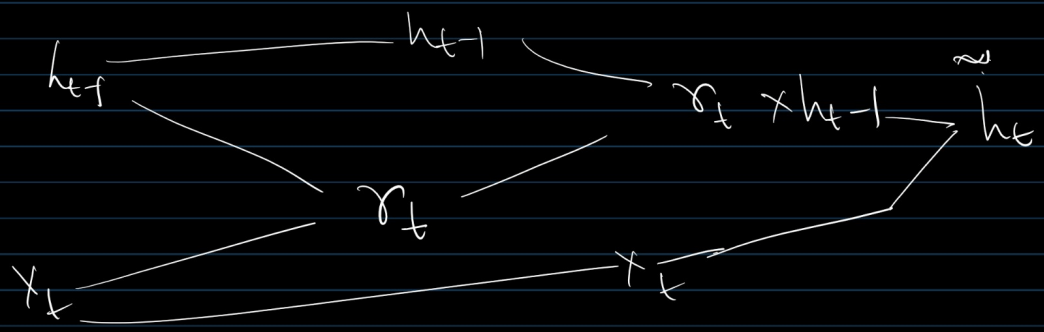


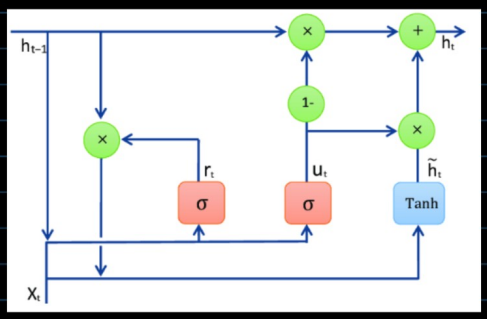




h_t

$1 - u_t$





$h_{t-1} \longrightarrow h_t$

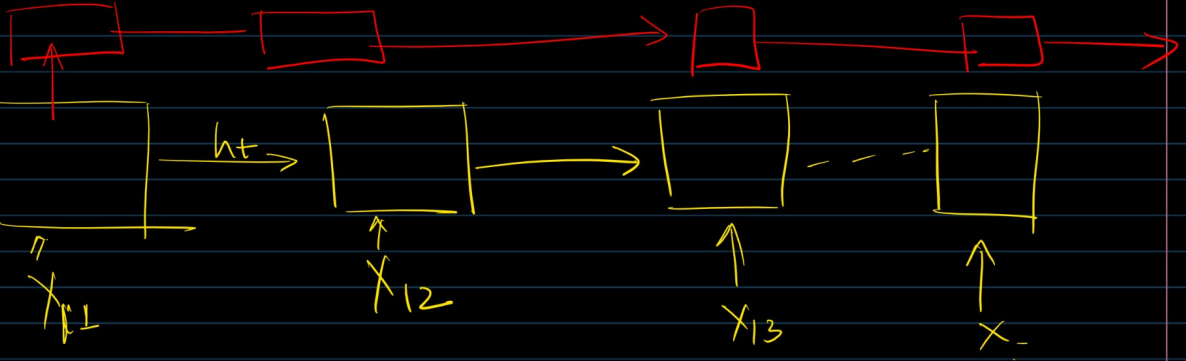
u_t / z_t

$\tilde{h}_t$

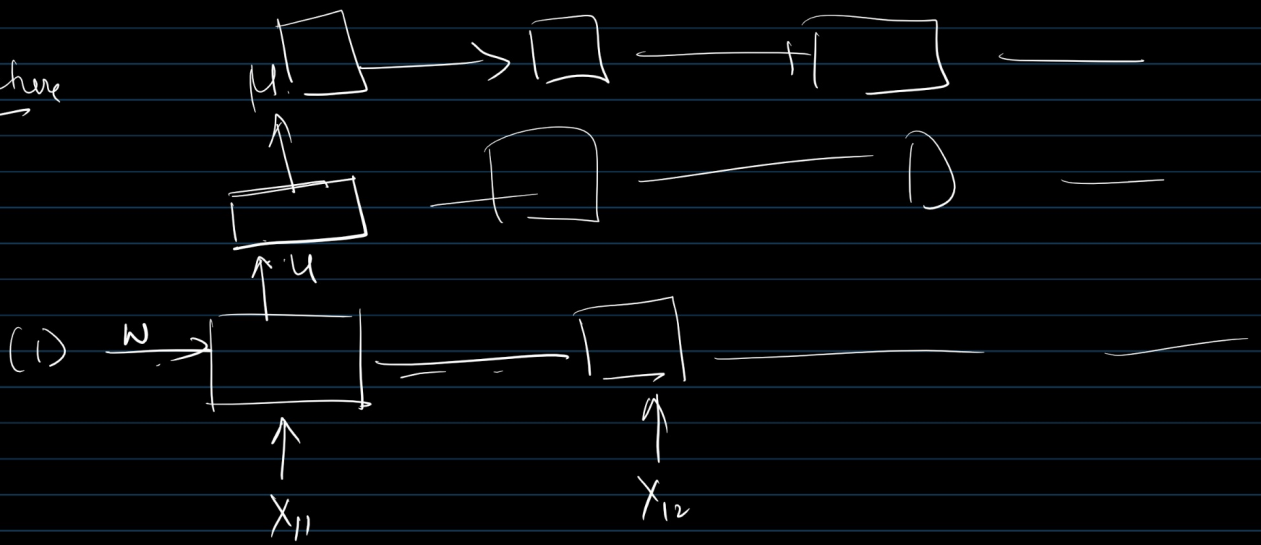
LSTM → more gates → parameter

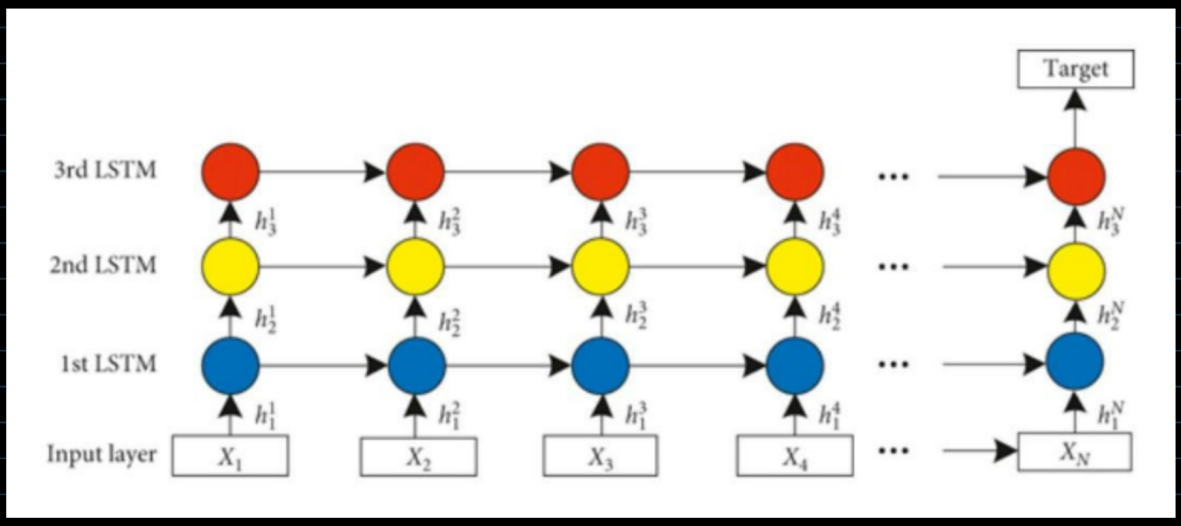
Deep RNN / stacked RNN

→ 2nd hidden layer.



Architecture





Disadvantage of dense RNN

↳ Overfitting & training time ↑.

Bidirection RNN

NER

India is great

↓

Location

Microsoft is good

↓

Org.

Statement 1 : I saw a bat, I played cricket.

(bat → Working thing)

Statement 2. I saw a bat, it flew from east to west.

↓

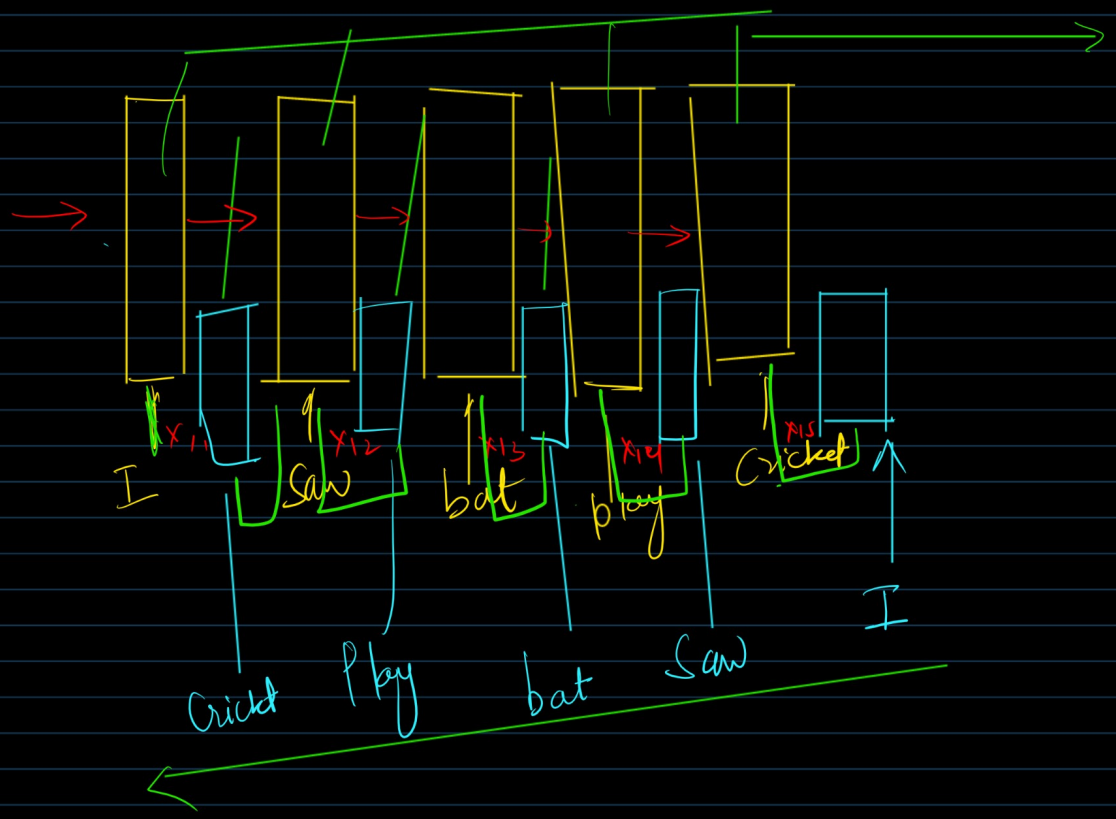
Animal

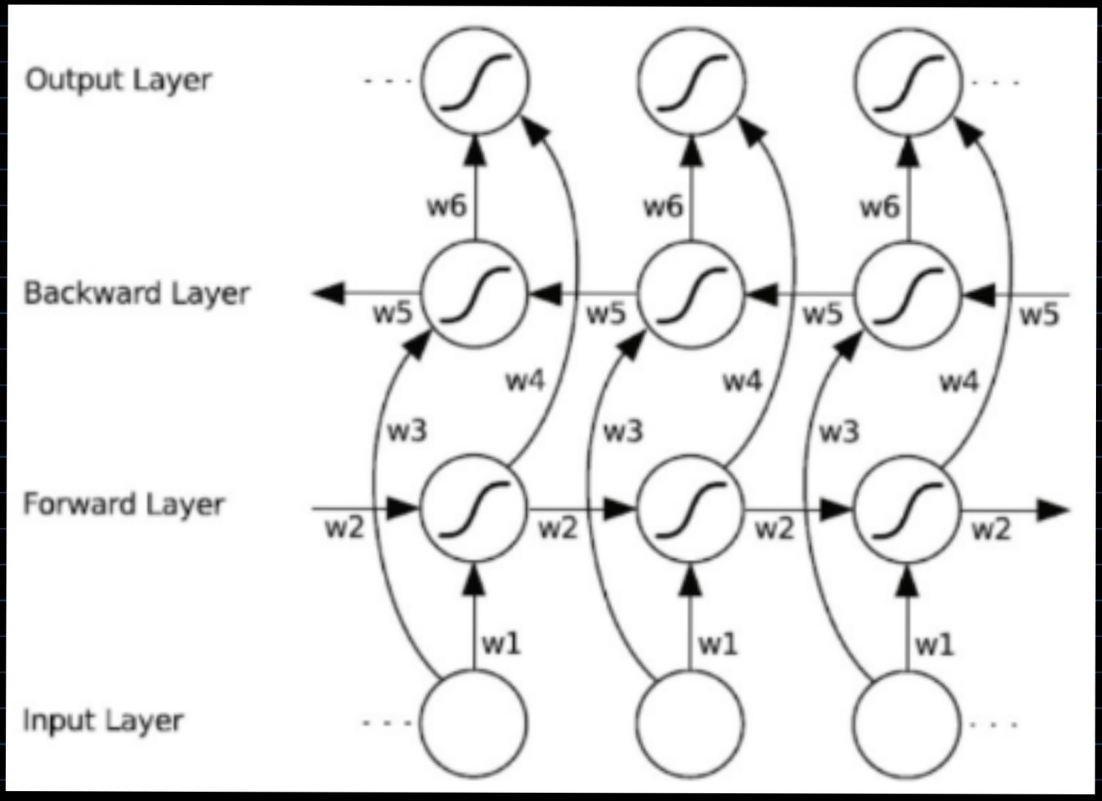
Predicting current based on future

Bi RNN

Forward RNN →

← Backward RNN





{ [India is great] → [1 2 3 0]
[I love playing cricket] [5 6 7 8]