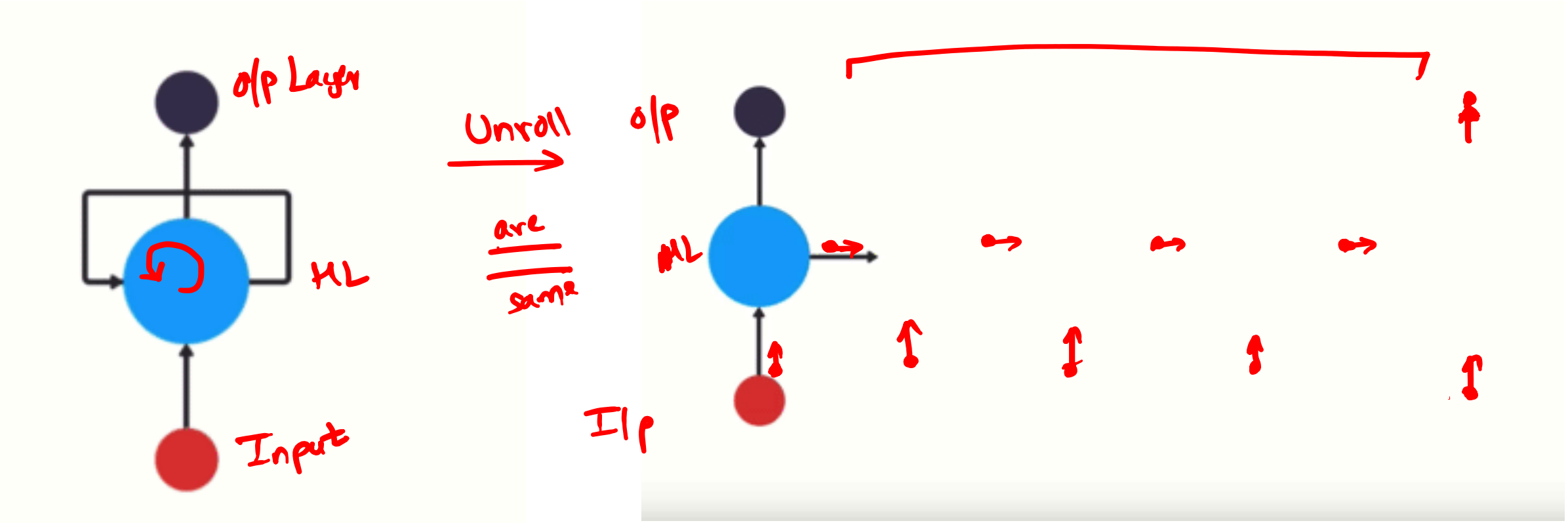
DEEP LEARNING – Recurrent Neural Network

Trainer: Dr. Darshan Ingle.

Recurrent Neural Network



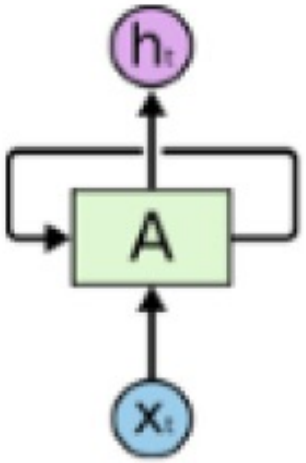
Recurrent Neural Network (RNN)

Humans dont start thinking from d scratch every second.

Our thoughts hv persistence.

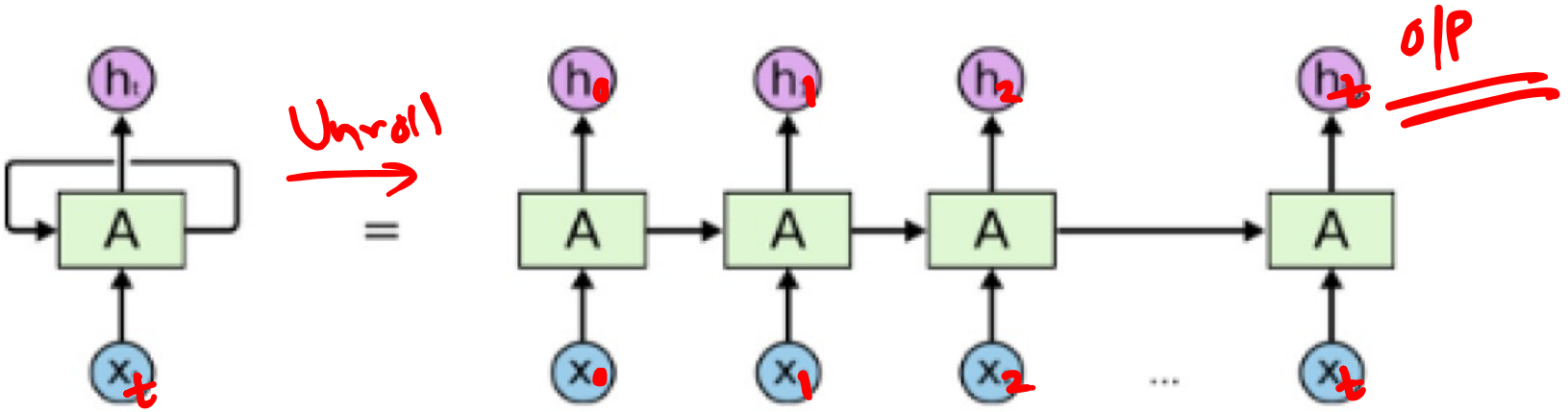
eg: what kind of event is happening at every point in a movie.

RNN_ can address this issue.



loop (they allow info. to persist)

RNN



An unrolled recurrent neural network.

Why RNN?

Sequence: It is a stream of data (finite / infinite) which are interdependent.
eg: Time Series, Conversations, informative pieces of strings, etc.

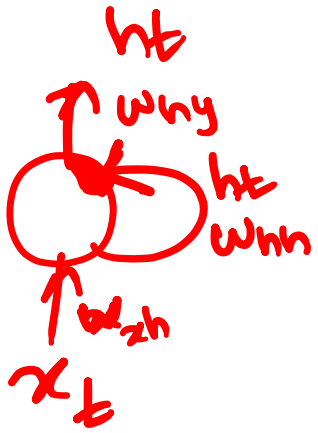
Why not CNN? Why go to RNN?

Soln: CNN — dnt perform well when input data is interdependent in a sequential pattern.
∴ all the outputs are self dependent.

Example

Predict what is the next letter, after a sequence of letters.

N A M A S T E ___
1 2 3 4 5 6 7 8th



$$h_t = f(h_{t-1}, x_t)$$

where h_t = new ~~state~~

h_{t-1} = previous state

x_t = input at time t.

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

$$y_t = w_{hy} \cdot h_t$$

Example

RNN Types

Each rectangle is a vector.

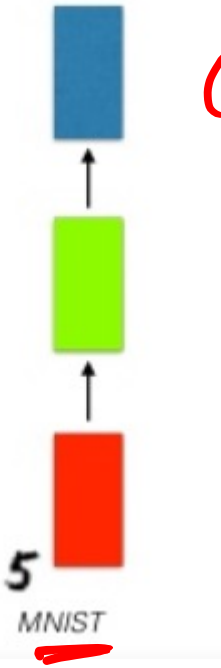
Arrows represents functions (eg: Mat. Mult)

Input vectors → Red

O/P vectors → Blue

RNN State → Green.

eg: Img Classification



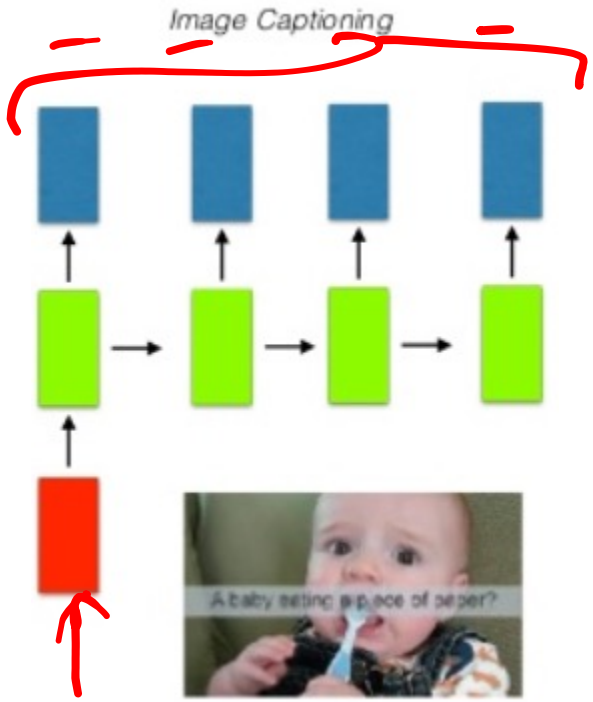
① One to One: Vanilla mode of processing w/o RNN.

from fixed sized i/p to fixed size o/p

RNN Types

2. One to Many: Sequence o/p.

eg: Img Captioning

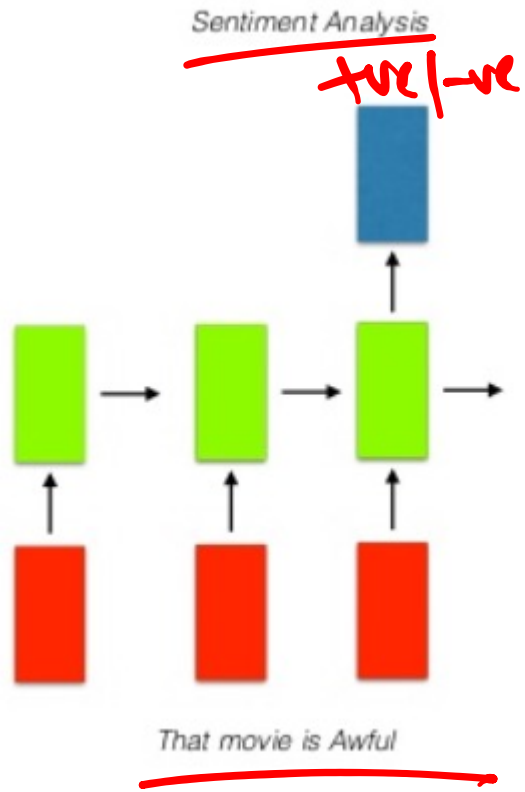


A Baby Eating a piece of paper

RNN Types

3. Many to One: Seq. input

eg: Sentiment analysis

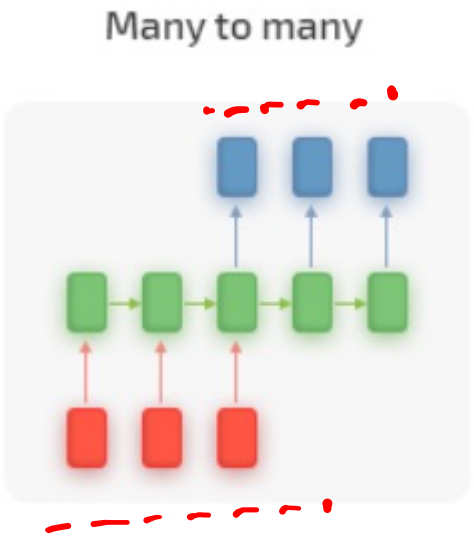


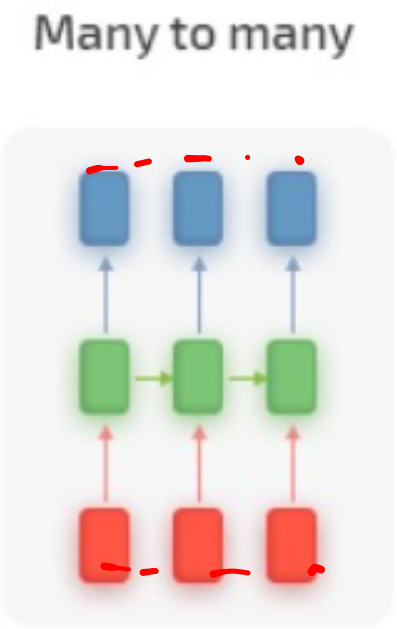
RNN Types

4.) Many to Many:

a.) Seq. input Seq. output

eg: M/c Translation.





b.) Synced Seq. i/p Seq. o/p:

eg: Video classification.

Why RNN?

NLP: Input Data

Used popularly used in NLP Problems

X = Text format
processed X = Number format ← preprocessing + Bag of Words (CountVectorizer)

Eg: 0 - I love playing
1 - We love playing cricket
2 - Playing is fun.

Preprocess. ↓ lower, rem. stopwords, apply stemming

0 - love play
1 - love play cricket
2 - play fun

TrainBayes Algo.

	love	play	cricket	fun
0 love play	1	1	0	0
1 love play cricket	1	1	1	0
2 play fun	0	1	0	1
Sum:	2	3	1	1

Desc. order.

Seq. info is lost.

	play	love	Cricket	fun
0	1	1	0	0
1	1	1	1	0
2	1	0	0	1

	x_0	x_1	x_2	x_3
0 - I love playing →	1	1	0	0
1 - We love playing cricket →	1	1	1	0
2 - Playing is fun →	1	0	0	1

Why RNN?

Summary

Summary

Red Ink: FP
Blue Ink: BP

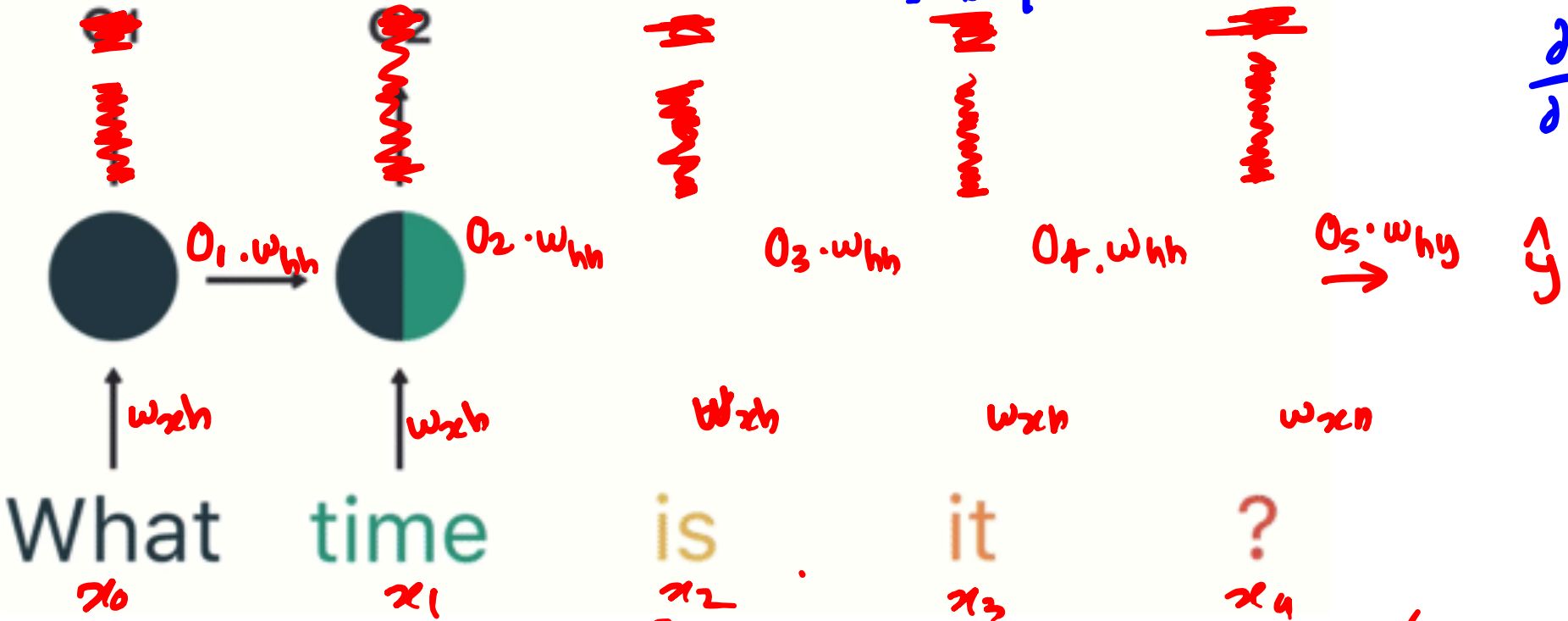
I/p to HL: w_{xh}, HL2HL: w_{hh}, HL2O/p: w_{hy}.

$$\frac{\partial L}{\partial w_{hy}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w_{hy}}$$

$$w_{hy\,new} = w_{hy\,old} - \eta \cdot \frac{\partial L}{\partial w_{hy\,old}}$$

$$\frac{\partial L}{\partial w_{xh}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial O_4} \cdot \frac{\partial O_4}{\partial w_{xh}}$$

$$w_{xh\,new} = w_{xh\,old} - \frac{\partial L}{\partial w_{xh}}$$



$\frac{\partial L}{\partial \hat{y}}$ can be computed directly w/o chain rule.

$$O_1 = f[(x_0 \cdot w_{xh} + b_{xh}) + (O_0 \cdot w_{hh} + b_{hh})]$$

$$O_2 = f[(x_1 \cdot w_{xh} + b_{xh}) + (O_1 \cdot w_{hh} + b_{hh})]$$

$$O_3 = f[(x_2 \cdot w_{xh} + b_{xh}) + (O_2 \cdot w_{hh} + b_{hh})]$$

$$\hat{y} = f[O_5 \cdot w_{hy} + b_{hy}]$$

RNN Architecture (Fwd Prop over Time)

RNN Architecture (Fwd Prop over Time)

RNN Architecture (Back Prop over Time)

RNN Architecture (Back Prop over Time)

Problems with RNN

① tanh : [-1, +1]

Der. of tanh : [0, 1]

→ Vanishing Gradient Problem.

- * - * - * .... = v.v. small gradient.

Exploding Gradient Problem

- caused due to hugely assigned weights