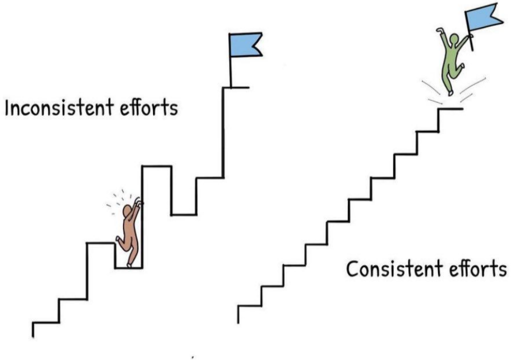
Deep Sequence Modelling

M.Tech. Data Science, Second Year, NMIMS

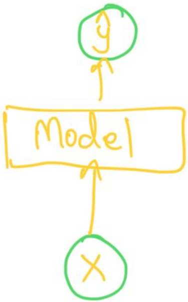
By,

Bilal Hungund, Data Scientist, Halliburton

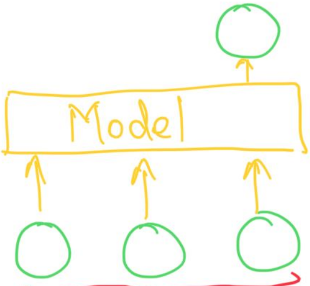


Sequencing Modelling Applications

One to one
(Classification / Regression)



Many to One
(Sentimental Analysis)



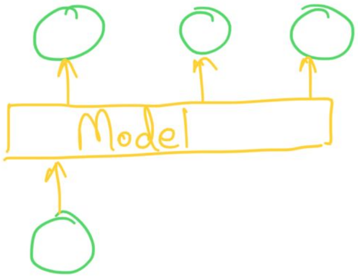
Sequences

BioDecoded
@BioDecoded

ScanNet uncovers binding motifs in protein structures with deep learning | Nature Methods
nature.com/articles/s4159...

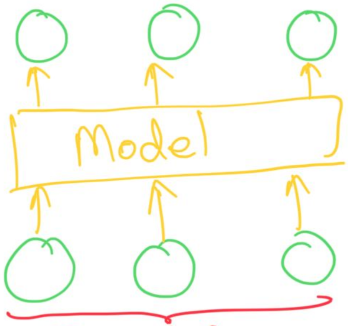
#DeepLearning

One to Many
(Image Captioning)



Cat is sleeping

Many to Many
(Language Modelling)

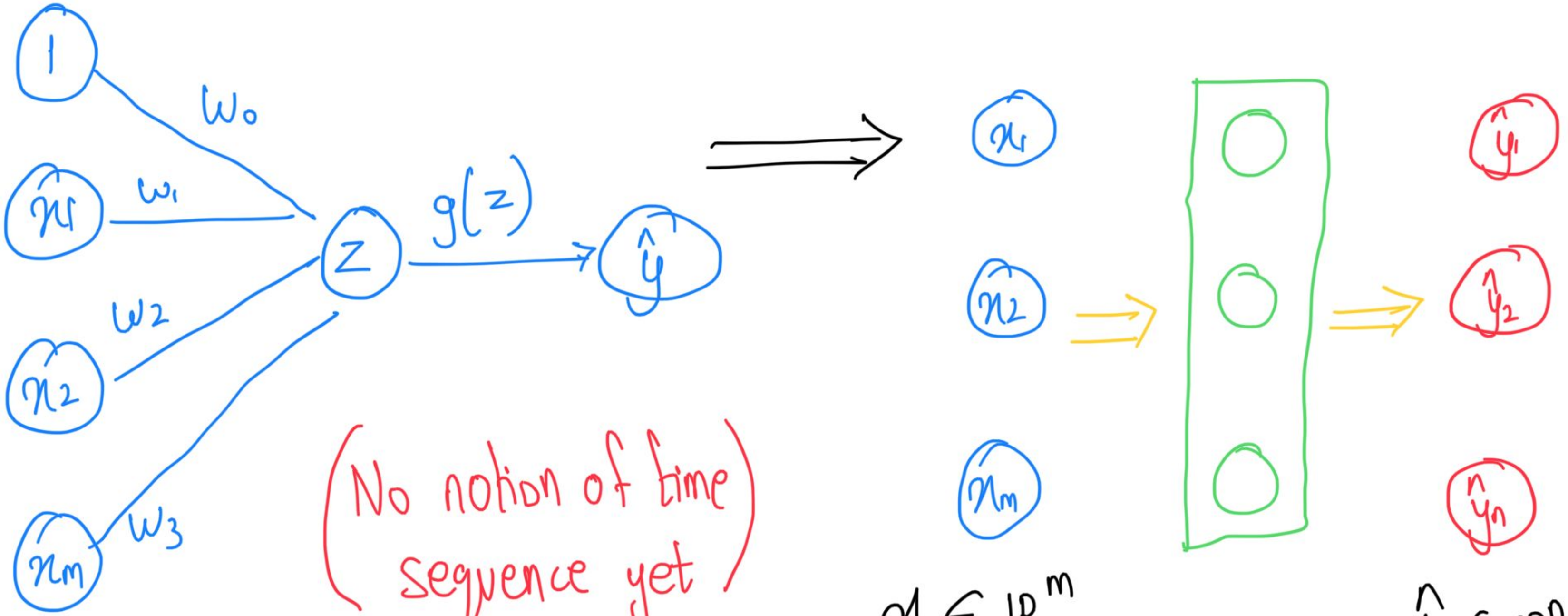


Temporal Space

English to Hindi Text

Perceptrons Revisited

Feed Forward Models



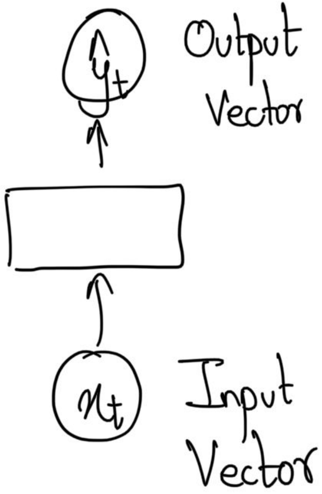
(No notion of time sequence yet)

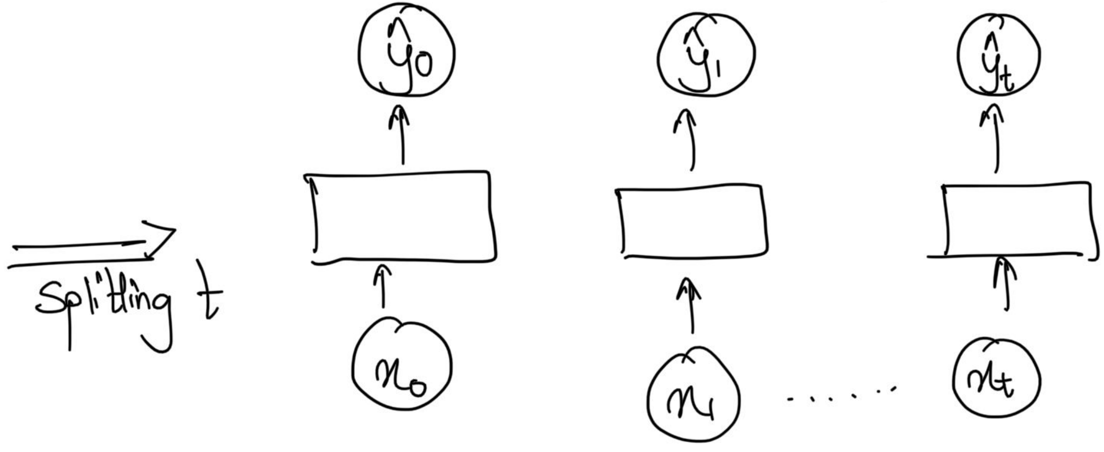
$x \in \mathbb{R}^m$ $\qquad$ $\hat{y} \in \mathbb{R}^n$

Handling with time steps

(How to relate network computations?)

Need some prior history

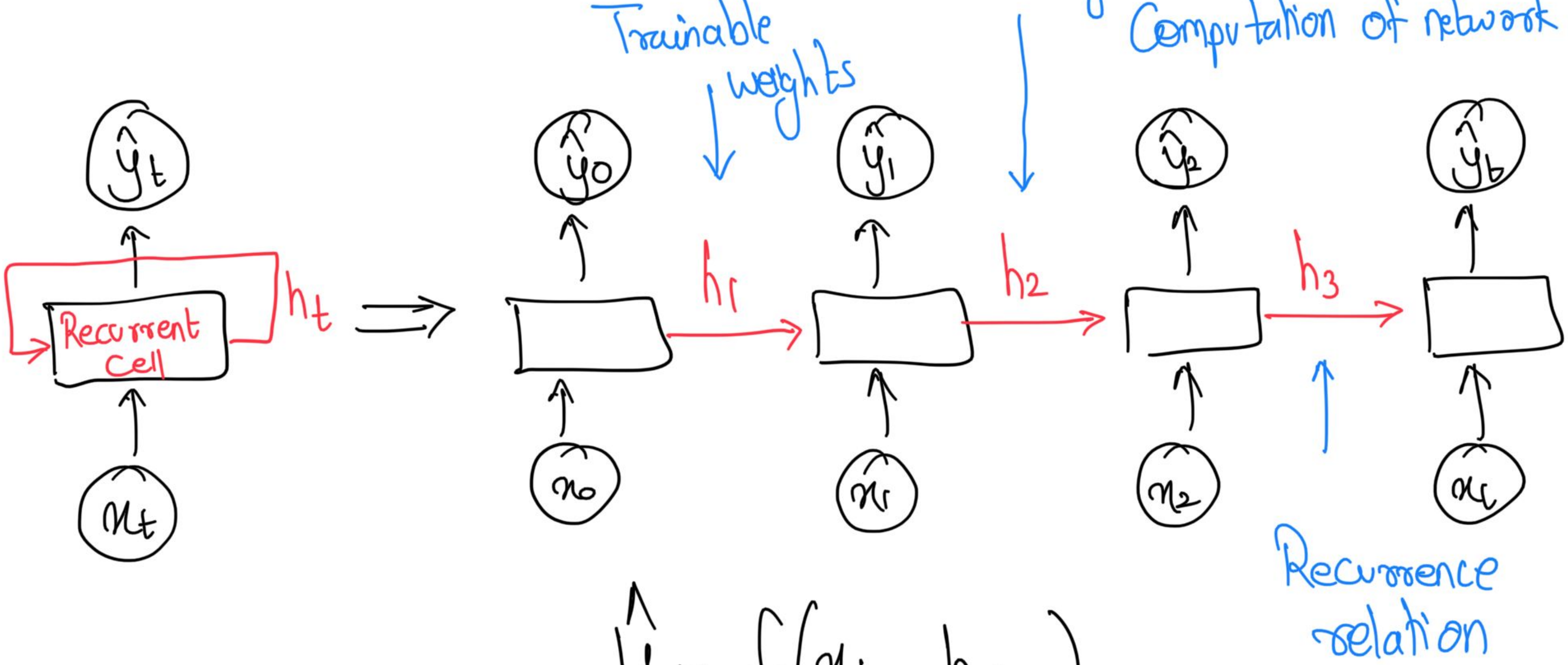




$$\hat{y}_t = f(x_t)$$

No interdependence or interconnectedness added yet

Neurons with Recurrence



$$\underbrace{\hat{y}}_{\text{Output}} = f(\underbrace{x_t}_{\text{Input}}, \underbrace{h_{t-1}}_{\text{past memory}})$$

Recurrence Neural Networks (RNNs)

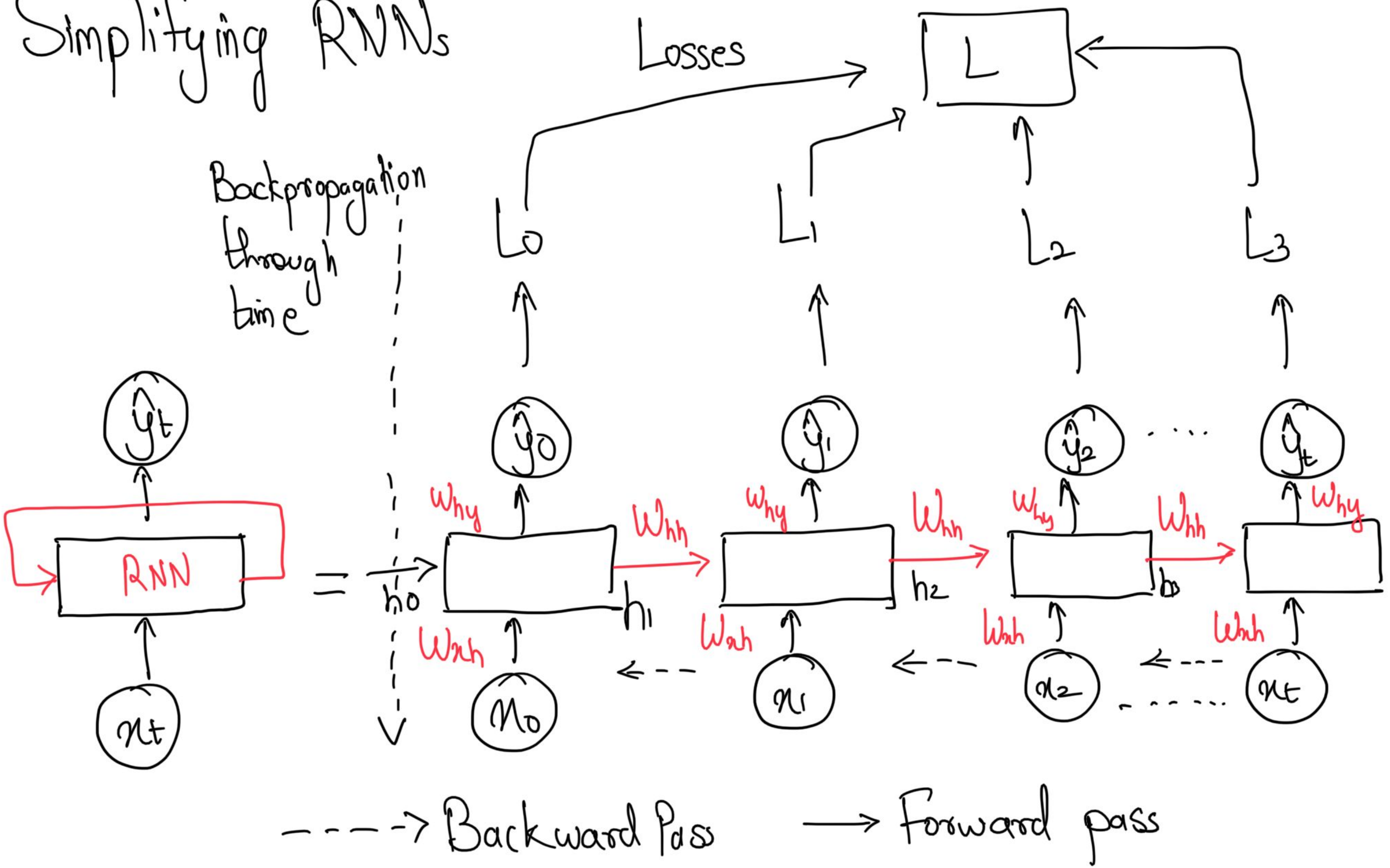
→ RNNs have a state h_t, that is updated at each time steps as a sequence

i.e.,

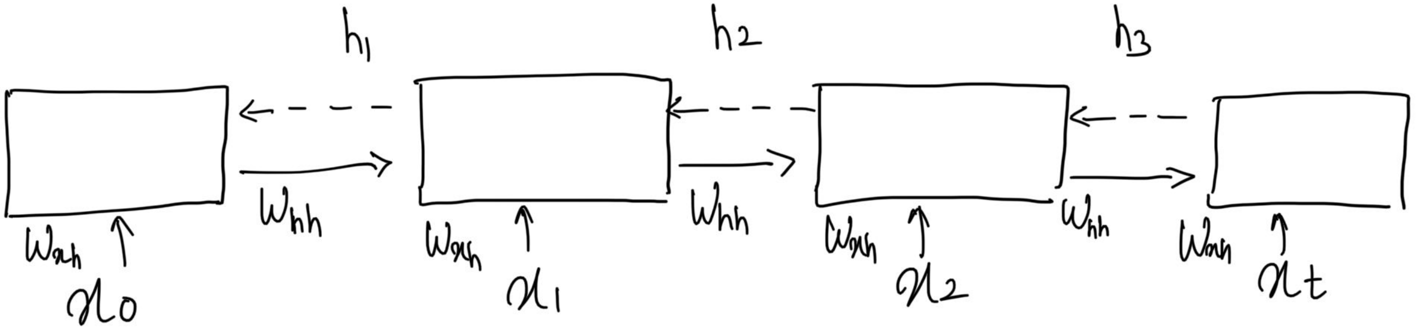
$$\underbrace{h_t}_{\text{cell state}} = \underbrace{f_w}_{\text{weight}} (\underbrace{x_t}_{\text{inputs}}, \underbrace{h_{t-1}}_{\text{old state}})$$

Input Vector	Update Hidden State	Output Vector
x_t	$h_t = \tanh(W_{hh}^T h_{t-1} + W_{xh}^T x_t)$	$\hat{y}_t = W_{hy}^T h_t$

Simplifying RNNs



Gradient issues



→ Exploding gradients (values > 1)

→ Gradient Clipping : Scale big gradients

→ Vanishing gradients (values < 1)

→ Activation Function, Weight initialization, Network architecture

Vanishing gradients

→ Focusing on short-term dependencies and ignoring long term dependencies

Short term dependencies $\xRightarrow{\text{Ignorance}}$ I studied data science and

I love neural ? (network)

I am fluent in ? (ML, DL, ...)

To alleviate the problem

→ Use of ReLU function as it prevents shrinking the gradients when $x > 0$

→ Weight Initialization:

Initialize bias to zero and weights to identity matrix

It prevents the weights from shrinking to zero

$$I_n = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & & 0 \\ \vdots & \vdots & \ddots & \\ 0 & 0 & & 1 \end{bmatrix}$$

→ Gated Cells : (LSTM)

Network Architecture

Use gates to add or remove information within each recurrent unit and to track information

Design Criteria of RNNs

- Handle variable-length sequences
- Track long term dependencies
- Maintain order of information
- Sharing parameters

Goals of Sequence Modelling

- Continuous stream
- Parallelization
- Long Memory

Limitations of RNNs

- Encoding bottleneck
- Slow, no parallelization
- Not long memory

NLP using RNN

Tokenization, pad Sequences and Embeddings

F L O W
001 002 003 004

Character Tokens

W O L F
004 003 002 001

Corpus/vocabulary

I love neural network
001 002 003 004

I love deep learning
001 002 005 006

Sequences

[[001, 002, 003, 004],
[001, 002, 005, 006]]

Tokenization, pad Sequences and Embeddings

building vocabulary

I love neural network
I love deep learning
You love neural network
Do you think deep learning is good?

Tokens ⟹

love - 1, i - 2, you - 3,
neural - 4, network - 5,
deep - 6, learning - 7,
do - 8, think - 9, good - 10
is - 11

Sequences

[2 1 4 5]
[2 1 6 7]
[3 1 4 5]
[8 3 9 6 7 11 10]

pad Sequences

$$\begin{bmatrix} 0 & 0 & 0 & 2 & 1 & 4 & 5 \\ 0 & 0 & 0 & 2 & 1 & 6 & 7 \\ 0 & 0 & 0 & 3 & 1 & 4 & 5 \\ 8 & 3 & 9 & 6 & 7 & 11 & 10 \end{bmatrix}$$

Pass to Embeddings