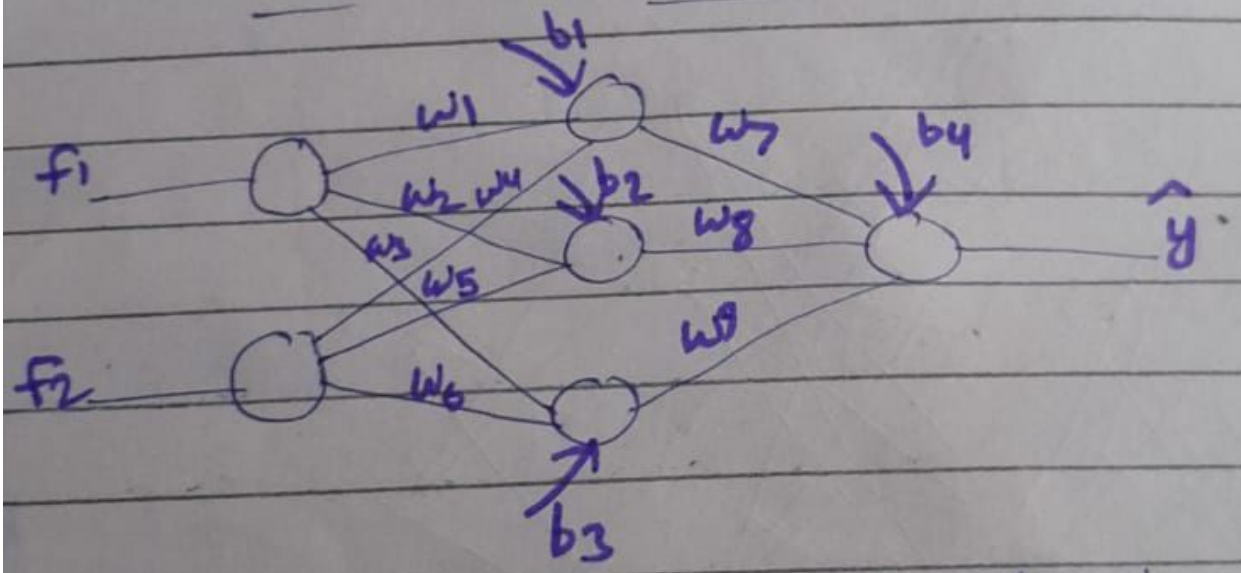
⇒ NLP in Deep Learning:

① ANN → Artificial Neural Network

» Tabular data → Classification
→ Regression

e.g House Price Prediction

f_1	f_2	y
House Size	No. of Rooms	Price



Here sequence of data doesn't matter we can interchange f_1 and f_2. This is called Non-Sequential data.

② CNN (Convolutional Neural Network)
↳ Images → Image classification/ object detection

③ RNN → data (Sequential data)
↓
Sequence/order is important (changing sequence change meaning)

e.g (sequential data) Input

(i) Text Generation → This is an apple
output
↳ juice

(ii) Chatbot conversation → Q and A
(sequential Data → Sequence is also important here)

ii) Language Translation
↳ [English] → [French]
(Here also sequence is important)

) Autogeneration
Sales Data → Time Series (Datatime)
Sales Forcasting → Sequence matters

Can we solve with ANN?
↳ Sequential Data
e.g Sentiment Analysis

	Text	Output
S1	The food is good	1
S2	The food is bad	0
S3	The food is not good	0

Step 01: Text → Vectors
We have 4 unique words after ignoring stop words ⇒ Vocabulary size = 4

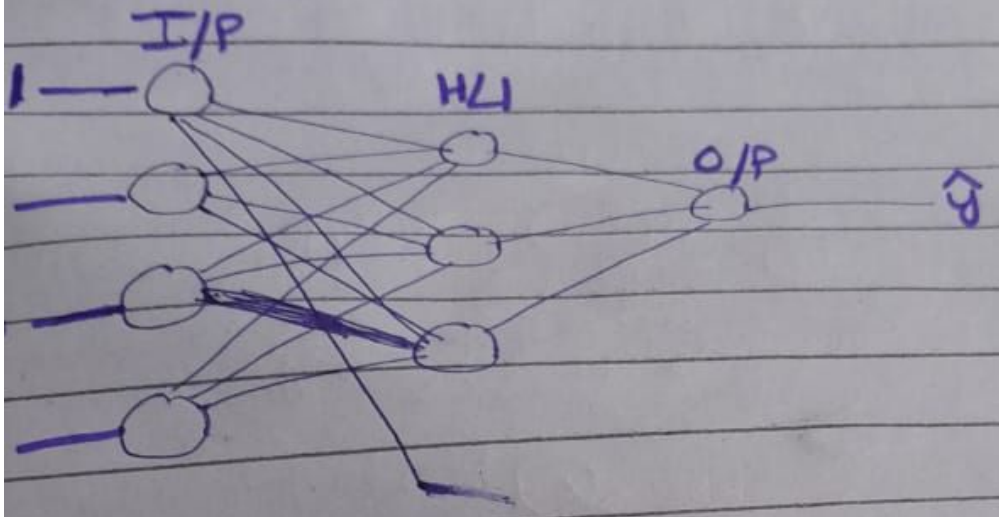
Bag of Words (BoW)

	food	good	bad	not
S1	1	1	0	0
S2	1	0	1	0
S3	1	1	0	1

Text Data → Sequence Informa
is important

- Here in ANN, we are losing

Sequence Information → Meaning/Context is losing.



Text Data:

Sequence of Information

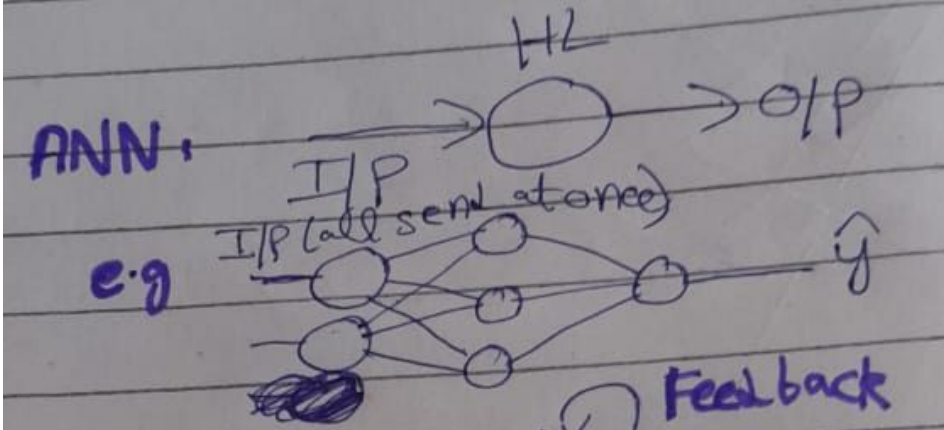
"The food is good"

Here we are giving whole sentence at once in ANN so sequence lost. The best practice is that we should give word by word one word at a time to maintain sequence and context.

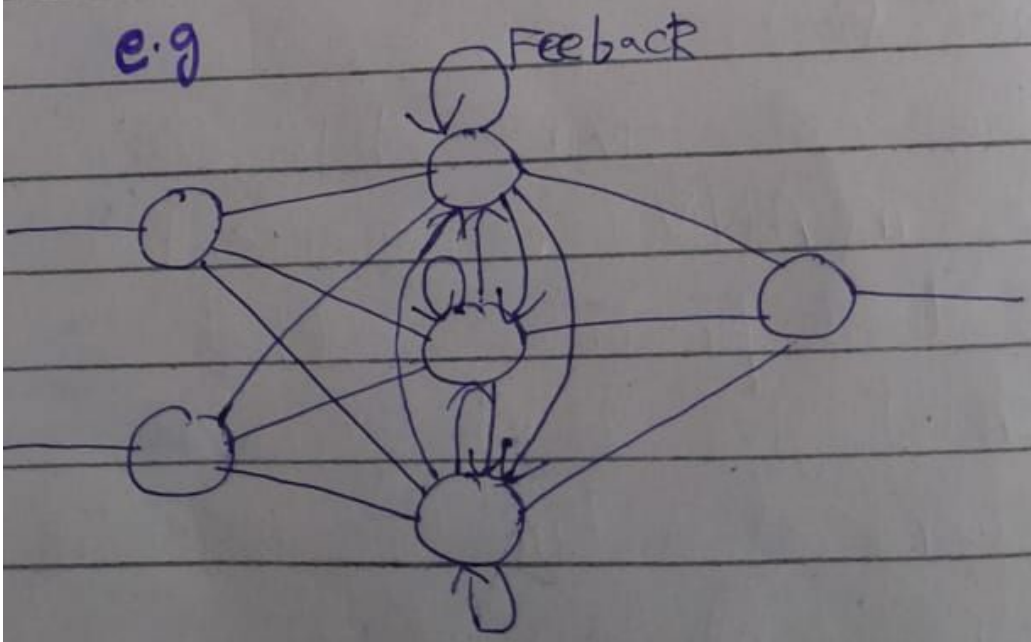
That's why to solve this kind of problem we use

Simple RNN (Recurrent Neural Network)

↳ Data is sequential



e.g



- Actually in RNN, a neuron shares it's output or feedback to itself

and all other neurons in the same hidden layer

eg

x_{11} x_{12} x_{13} x_{14}

The food is good

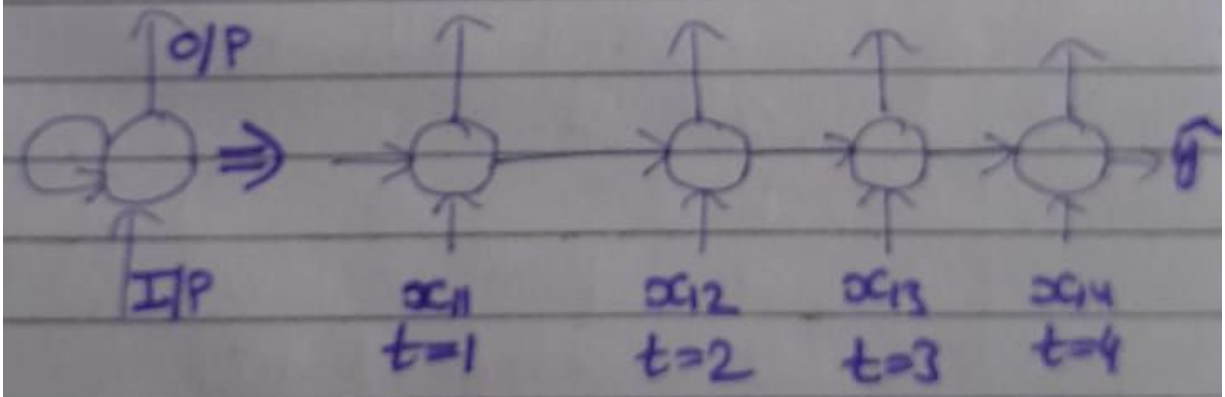
- In RNN, we pass each word at a particular time stamp (one word at a time)

At $t=1$, x_{11}

$t=2$, x_{12}

$t=3$, x_{13}

$t=4$, x_{14}



⇒ Working of Simple RNN with Forward Propogation:

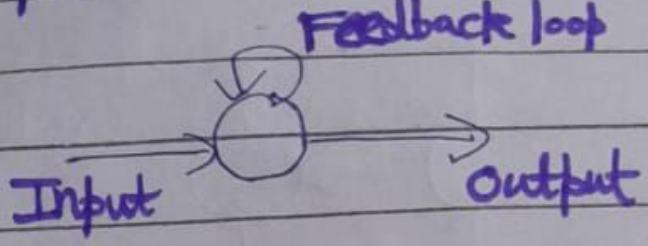
Dataset

Text	Output
The food is good	1
The food is bad	0
The food is not good	0

Words → Vectors (using one Hot encoding)

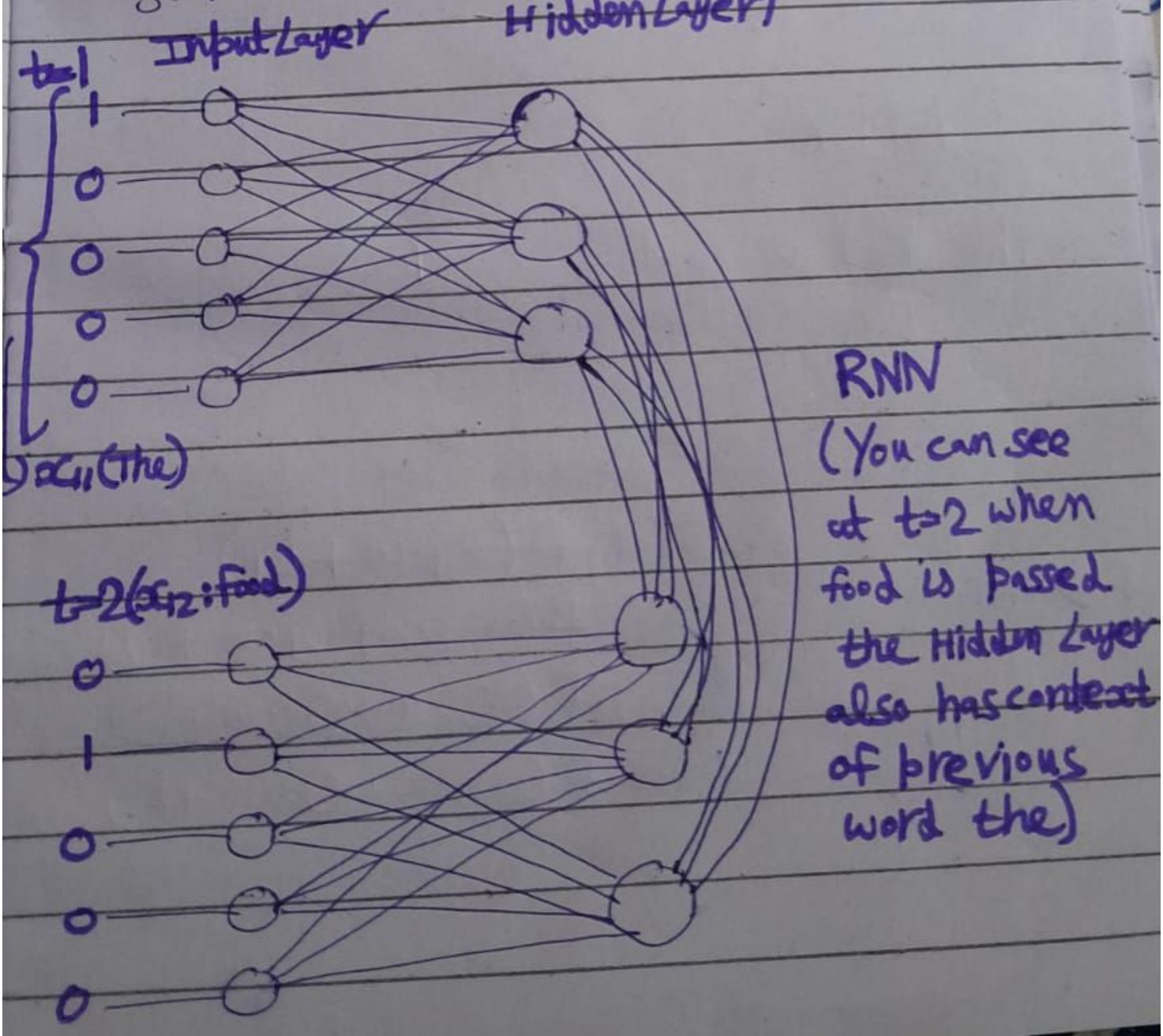
	The	food	good	bad	not
The (x_{11}) ⇒	1	0	0	0	0
food (x_{12}) ⇒	0	1	0	0	0
good (x_{13}) ⇒	0	0	1	0	0
bad (x_{14}) ⇒	0	0	0	1	0
not (x_{15}) ⇒	0	0	0	0	1

- Simple RNN Architecture



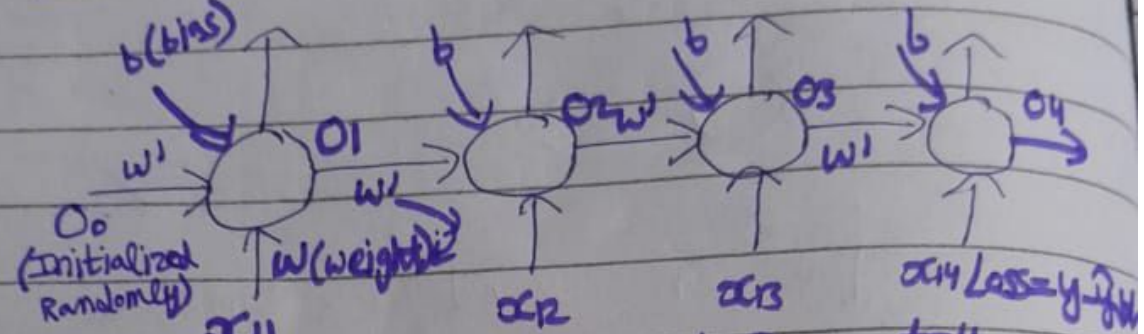
As we pass one word at a time and one word is representing by 5 inputs

So,



RNN
(You can see at t=2 when food is passed the Hidden Layer also has context of previous word the)

Forward Propogation with Time:



Dataset

Text	Output
The food is good	1
x_{11} x_{12} x_{13} x_{14}	

f ⇒ Activation function

Forward Propogation:

$$O_1 = f(x_{11} \cdot w + b_1)$$

$$O_2 = f(x_{12} \cdot w + O_1 \cdot w' + b_2)$$

$$O_3 = f(x_{13} \cdot w + O_2 \cdot w' + b_3)$$

$$O_4 = f(x_{14} \cdot w + O_3 \cdot w' + b_4)$$

RNN Back Propogation with time:

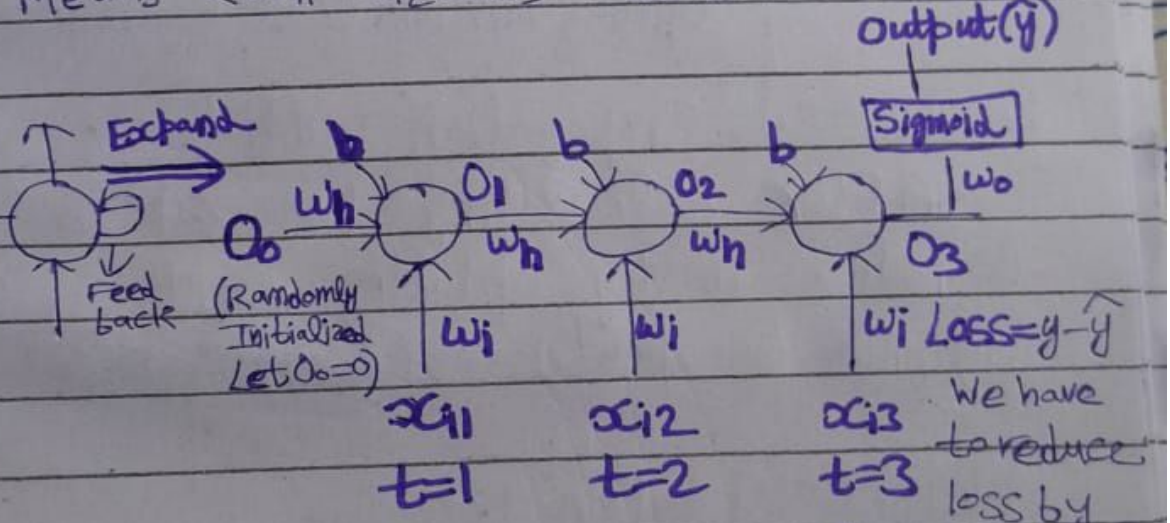
Let we have sentences

$S1 \langle x_{11}\ x_{12}\ x_{13} \rangle$

$S2 \langle x_{21}\ x_{22}\ x_{23} \rangle$

$S3 \langle x_{31}\ x_{32}\ x_{33} \rangle$

Means $\langle x_{i1}\ x_{i2}\ x_{i3} \rangle$ where i = sentence no.



We have ~~to reduce~~ loss by updating weight [Wi, Wh, Wo]

Back Propogation

Forward Propogation:

→ Activation function

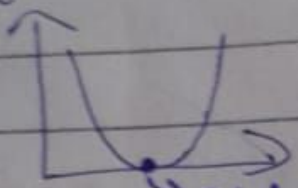
$$O_1 = f(x_{i1} * w_i + O_0 * w_h + b)$$

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

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

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

We will repeat forward and back propogation until convergence happens means until we reach global minima in gradient descent



Global minima (loss is minimum)

Backward Propogation with time:

After calculating loss we have to reduce it by updating weights
Weights to update are: $[w_i, w_h, w_o]$

Weight Updation Formula:

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

η: Learning rate

$\frac{\partial L}{\partial w_{old}}$: Derivative/ Slope of Gradient Descent

1. Updating w_o:

$$w_{o_{new}} = w_{o_{old}} - \eta \frac{\partial L}{w_{o_{old}}}$$

Chain rule of derivative

$$\frac{\partial L}{\partial w_{oow}} = \frac{\partial L}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial w_{oow}}$$

(2) Update w_h [Hidden Layer weights]

Time Stamps

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

We have 3 time stamps $t = h2,3$

$$\frac{\partial L}{\partial w_{h old}} = \left[\frac{\partial L}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial o_3} * \frac{\partial o_3}{\partial w_h}\right] \Rightarrow t=3$$

$$t=2 \Leftarrow + \left[\frac{\partial L}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial o_3} * \frac{\partial o_3}{\partial o_2} * \frac{\partial o_2}{\partial w_h}\right]$$

$$t=1 \Leftarrow + \left[\frac{\partial L}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial o_3} * \frac{\partial o_3}{\partial o_2} * \frac{\partial o_2}{\partial o_1} * \frac{\partial o_1}{\partial w_h}\right]$$

Updating weights w_i: → Time stamp

$$w_{i new} = w_{i old} - \eta \frac{\partial L}{\partial w_{i old}}$$

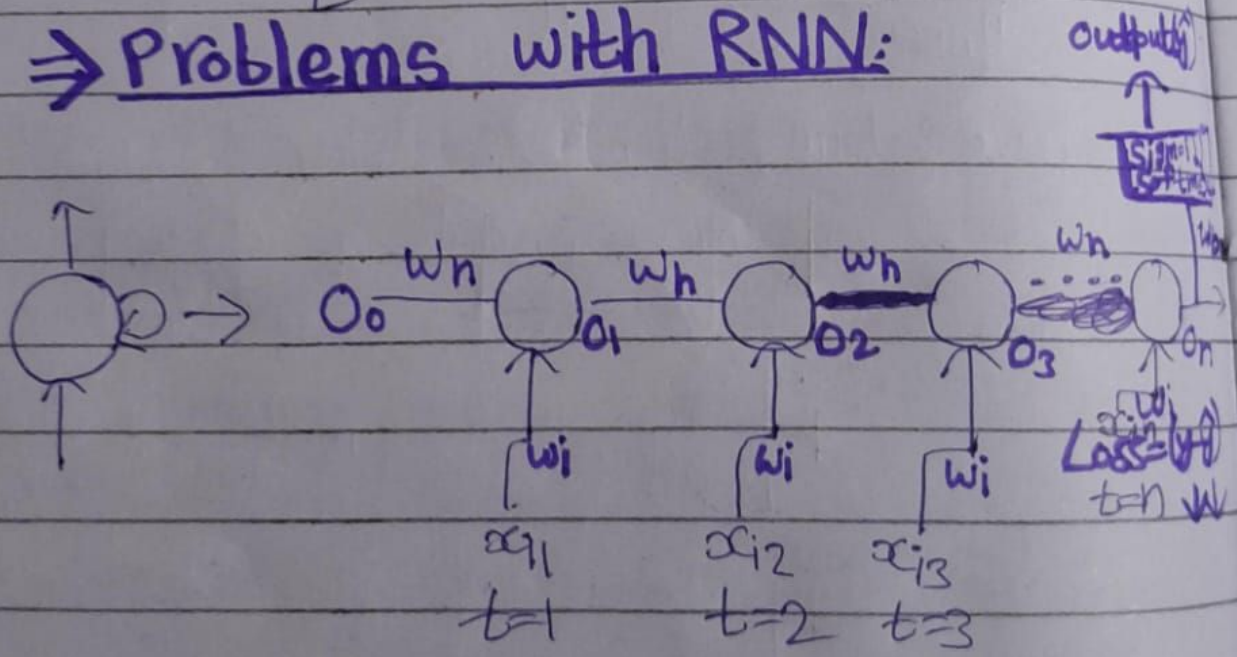
We have 3 timestamps $t=1,2,3$

$$\frac{\partial L}{\partial w_{i_{old}}} = \left[\frac{\partial L}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial O_3} * \frac{\partial O_3}{\partial w_{i_{old}}}\right]_{t=1} +$$

$$\left[\frac{\partial L}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial O_3} * \frac{\partial O_3}{\partial O_2} * \frac{\partial O_2}{\partial w_{i_{old}}}\right]_{t=2} +$$

$$\left[\frac{\partial L}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial O_3} * \frac{\partial O_3}{\partial O_2} * \frac{\partial O_2}{\partial O_1} * \frac{\partial O_1}{\partial w_{i_{old}}}\right]_{t=3}$$

⇒ Problems with RNN:



ANN → Vanishing Gradient Problem

Let we have dataset

Text	Output	
The food is good	1	} Small sentences
The food is bad	0	

Let we have to do Text Generation

I like to play ____

* Now the word which is to be generated/predicted can be dependent on any word of sentence.

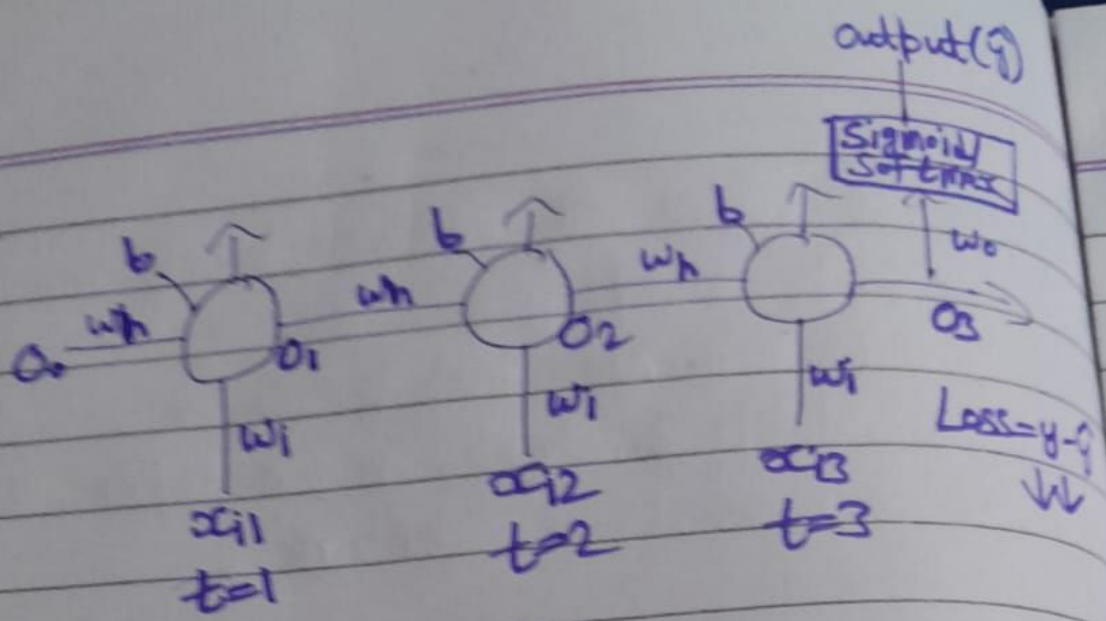
Now let we have a Long Sentence

" My name is Amir and I like cricket, football and also like to make ____ "

- Now the dependency is huge as word to be predicted can be dependent on any word

Issue:

Long Term dependency can't be captured by Simple RNN.



Backward propagation

Update weights: w_h, w_i, w_o

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

$$\frac{\partial L}{\partial w_{h_{old}}} = \frac{\partial L}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial w_{h_{old}}} \Rightarrow t=2$$

$$\frac{\partial L}{\partial w_{h_{old}}} = \left[\frac{\partial L}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial o_3} * \frac{\partial o_3}{\partial o_2} * \frac{\partial o_2}{\partial w_{h_{old}}} \right] + \ldots \Rightarrow t=1$$

You can see that the length of word is just 3 and we got so lengthy chain rule

What if length of words=50

At t=2

$$\frac{\partial L}{\partial w_{hh}} = \left[\frac{\partial L}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial o_{50}} * \frac{\partial o_{50}}{\partial o_{49}} * \frac{\partial o_{49}}{\partial o_{48}} \cdots \frac{\partial o_3}{\partial o_2} * \frac{\partial o_2}{\partial w_{hh}}\right] + \cdots$$

$$\frac{\partial o_3}{\partial o_2} = \frac{\partial \sigma(x_{13} * w_i + o_2 * w_h + b) \Leftarrow o_3}{\partial o_2}$$

We know

Derivative (Sigmoid) = [0 to 0.25]

When we keep on multiplying small values it will become ≈ 0

t=1 ⇒ The word is not participating to update the weights value.

⇓

Vanishing Gradient problem

⇒ So the last words in long sentences are effective but the first words are not participating and may be the word to predict depend on fist words.

⇒ So Simple RNN is unable to capture Long term dependency (when chain rule is very long)

To solve this problem we can use

① Relu, Leaky relu as activation function
② LSTM RNN → Long Short Term Memory N
③ GRU RNN