Recurrent Neural Network

Dr. Muhammad Safyan

Recurrent Neural Network

- Sequential Data
- Why Use RNN
- Applications of RNN
- Simple RNN
- Forward Propagation
  Types of RNN
- Back Propagation of RNN

Sequential Data

- Sequential Data→
  - ANN, Tabular data
- CNN→
  - Grid Type Data
- RNN→
  - Sequential Data,
- Tabular data is not Positional Sensitive.
  - e.g.

I Q	Gender	Marks

- Sequential data is Positional Sensitive
  - e.g. Hi, My Name is Safyan
- Time Series data(Stock Exchange)
  - Speech
  - DNA Sequence
- **RNN is special kind of data that is sequence specialist**

RNN

- RNN is favorite of NLP Domain
  - Initially NLP uses ML, ANN
  - CNN→ Image

Why Use RNN

- e.g. we want classification

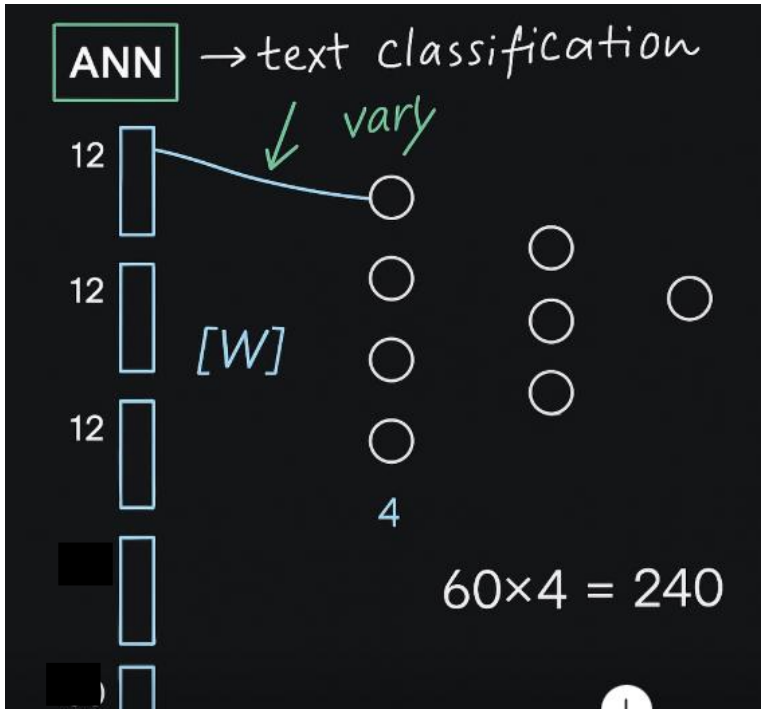
Sentiment Classification Example

Input Text	Output
Hi my name is Safyan	0
I love GCU	0
Pakistan won the match	1

- vectorize it(one-hot-encoding, Tf-idf, BoW, W2V) change each word in to vector :ONE
- Hi[1,0,0,……]
- My[0,1,0,0 ……..]

Why Use RNN

- Variable Length input(Textual Data).
  - Take longest sentence as input length.
- Zero Padding.
  - What if the size is 10 k
  - Max length is 100 words(Zero Padding)
  - Sparse (10*100),
  - Imagine weight Matrix.
  - Too Much Weight to Process
    Unnecessary computation
- Prediction Problem
  take features independent of order
- Disregarding of Sequential Information
  - All word of Sentence go alongside in ANN.
  - Loose Semantic meaning.
- Need a New Architecture



Application of RNN

- Sentiment Analysis – Free Online Demo
- https://sentivisor.com/sentiment-analysis-free-online-demo/
- Sentence Completion
  - Email/chrome Typing

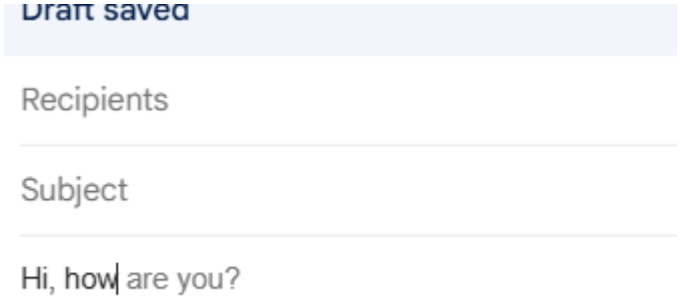
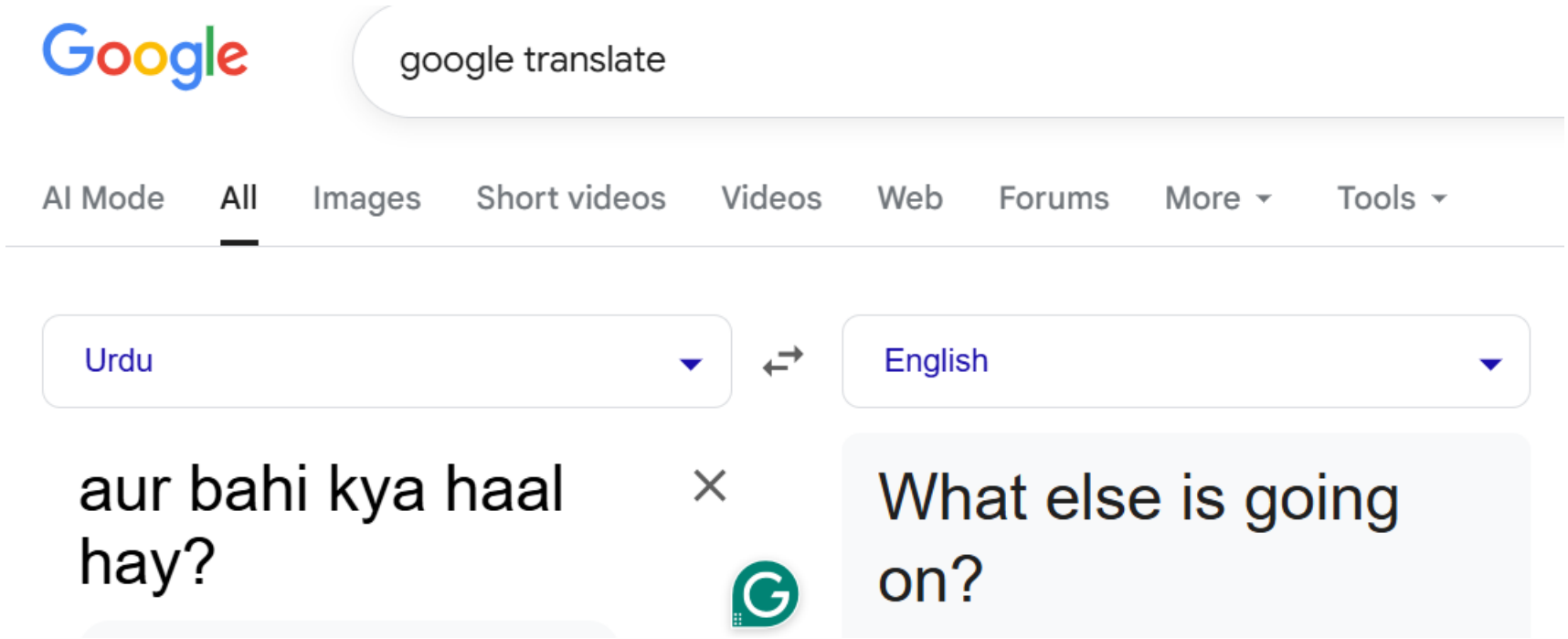


Image Caption

Application: for Blind person

Language Translation

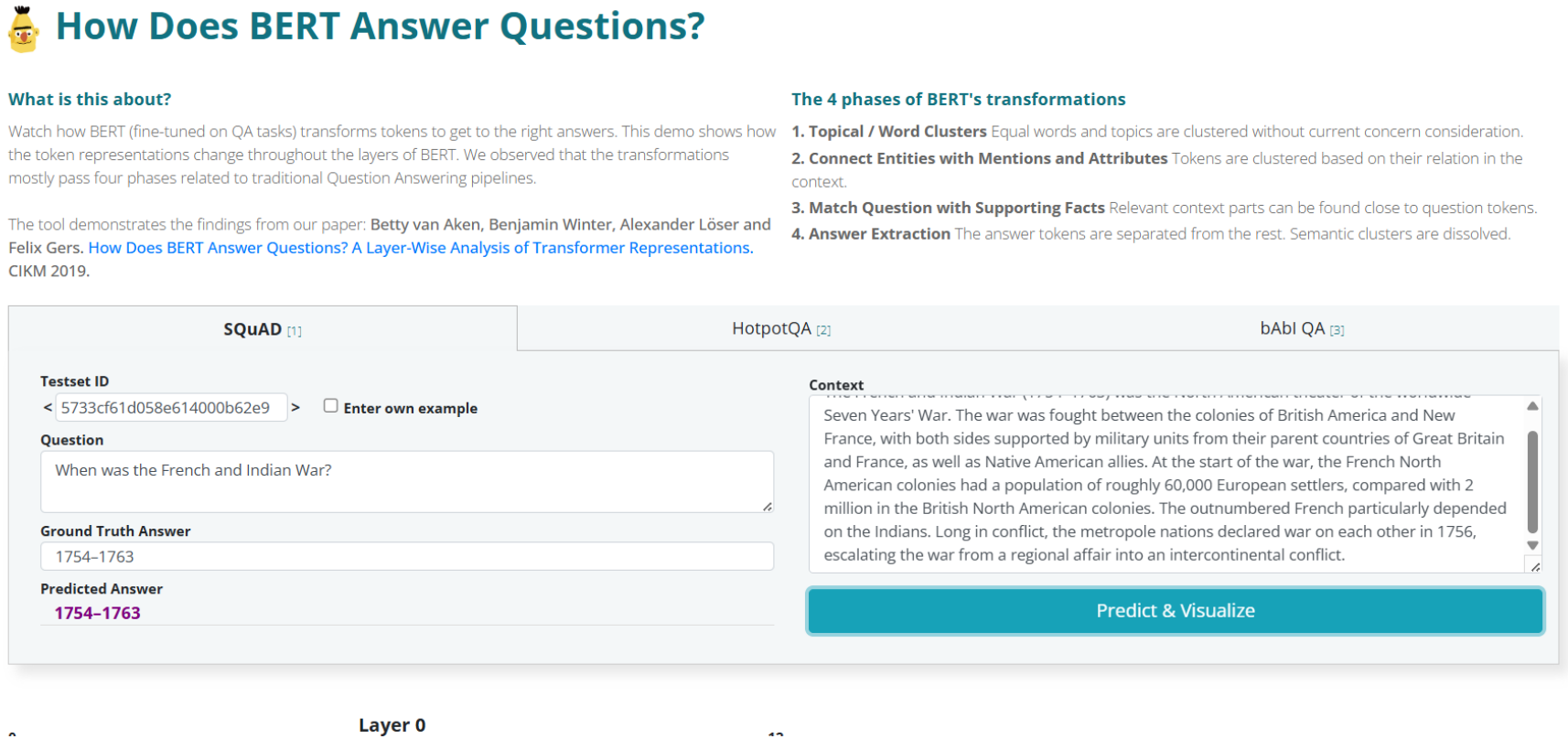
- Google Translation



- Email Translation

Question Answering

- **BERT Answer Questions?**
- https://visbert.demo.datexis.com/



Simple RNN Architecture

- Sequence data can be of any length
  - Movie Reviews,
- Sequence Contains Some Meanings
  - ANN loose meaning and unable to grasp the context.
  - RNN have the capability of hold some memory

RNN Data

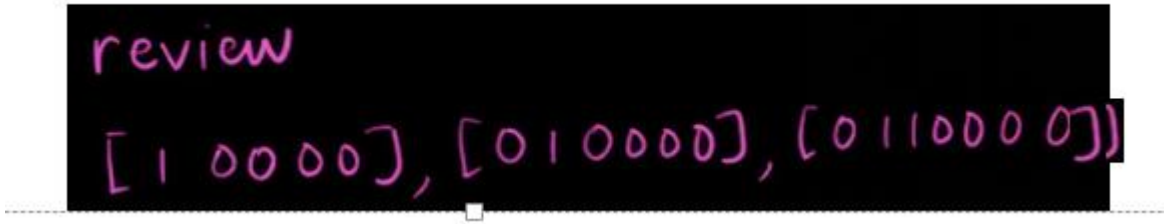
- (Timsteps, Input Features)

Review	Sentiment
movie was good	1
movie bad was	0
movie was not good	0

- Vocabulary size=5
- Movie=[1,0,0,0,0]
- Was=[0,1,0,0,0]
- --- ----- -
- --- ------

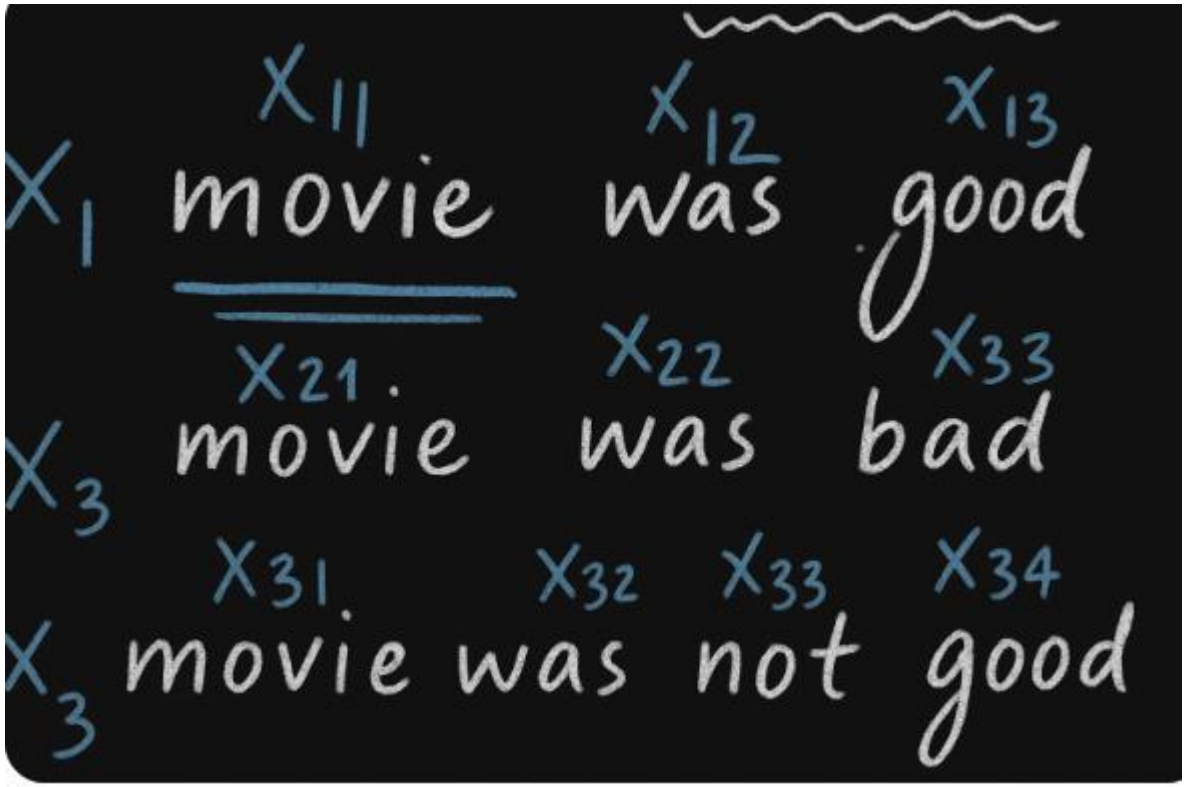
RNN

- Send the word in RNN one by one
- T=1, T=2, T=3



- First Review is of (3 * 5)
  - Where
    - 3 is the # of time steps
    - 5 is the # of Features
- 2nd Review (3*5)
- 3rd Review (4*5)
- Keras: batch size, time step, input features
    - (3,4,5) → 3 D Tensor→raw
    - 3 sentences, 4 words per sentence, 5 features per word

Data Layout


x_{11} x_{12} x_{13}
x_1 movie was good
x_{21} x_{22} x_{33}
x_3 movie was bad
x_{31} x_{32} x_{33} x_{34}
x_3 movie was not good

Difference in Structure of RNN& ANN

- ANN
  - Input layer → Hidden layers → Output Layer
- RNN
  - Input layer → Hidden layers → Output Layer
- 1st Difference:

  Pick your first review and pass on time basis

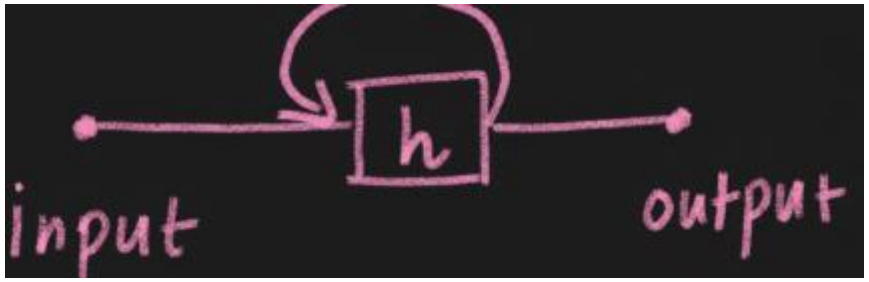
  t=1 → x1

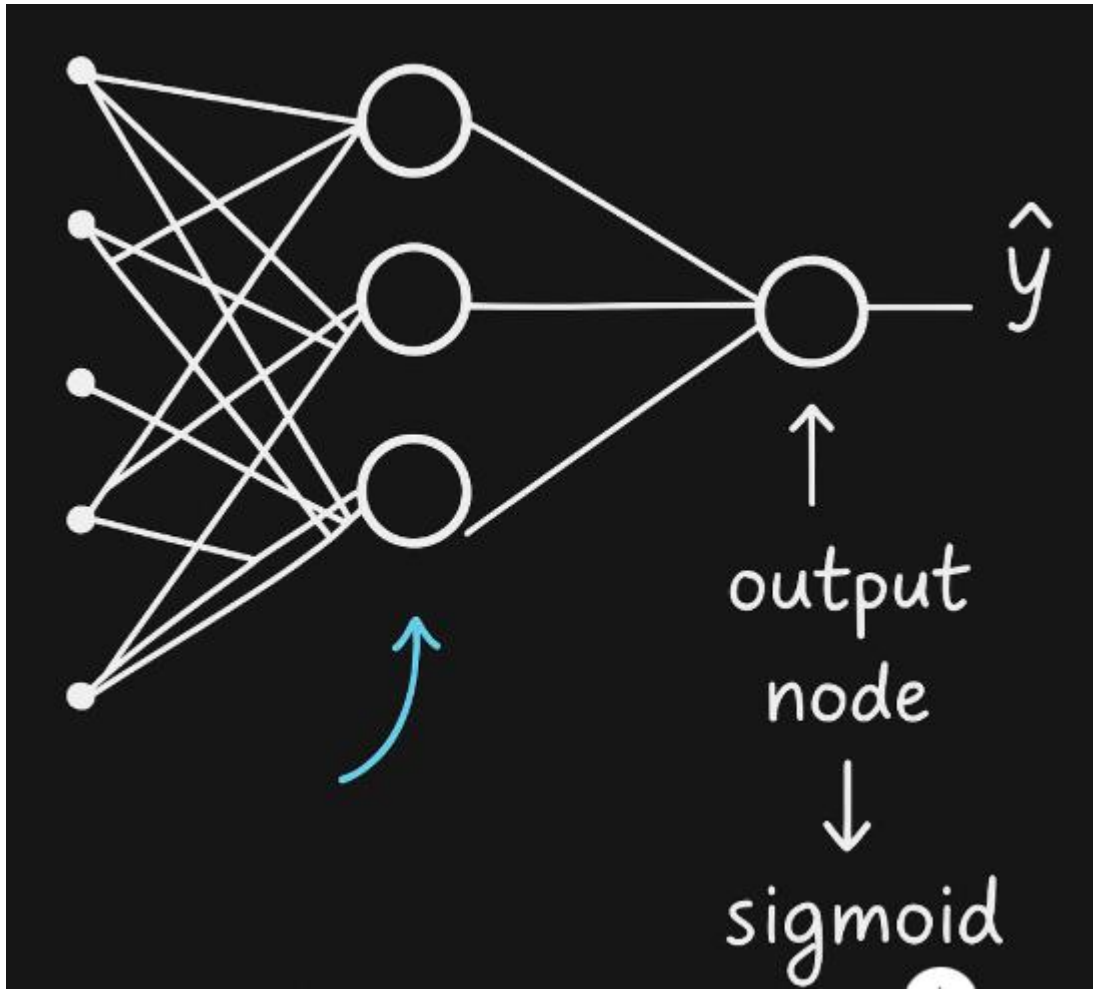
  t=2→ x2

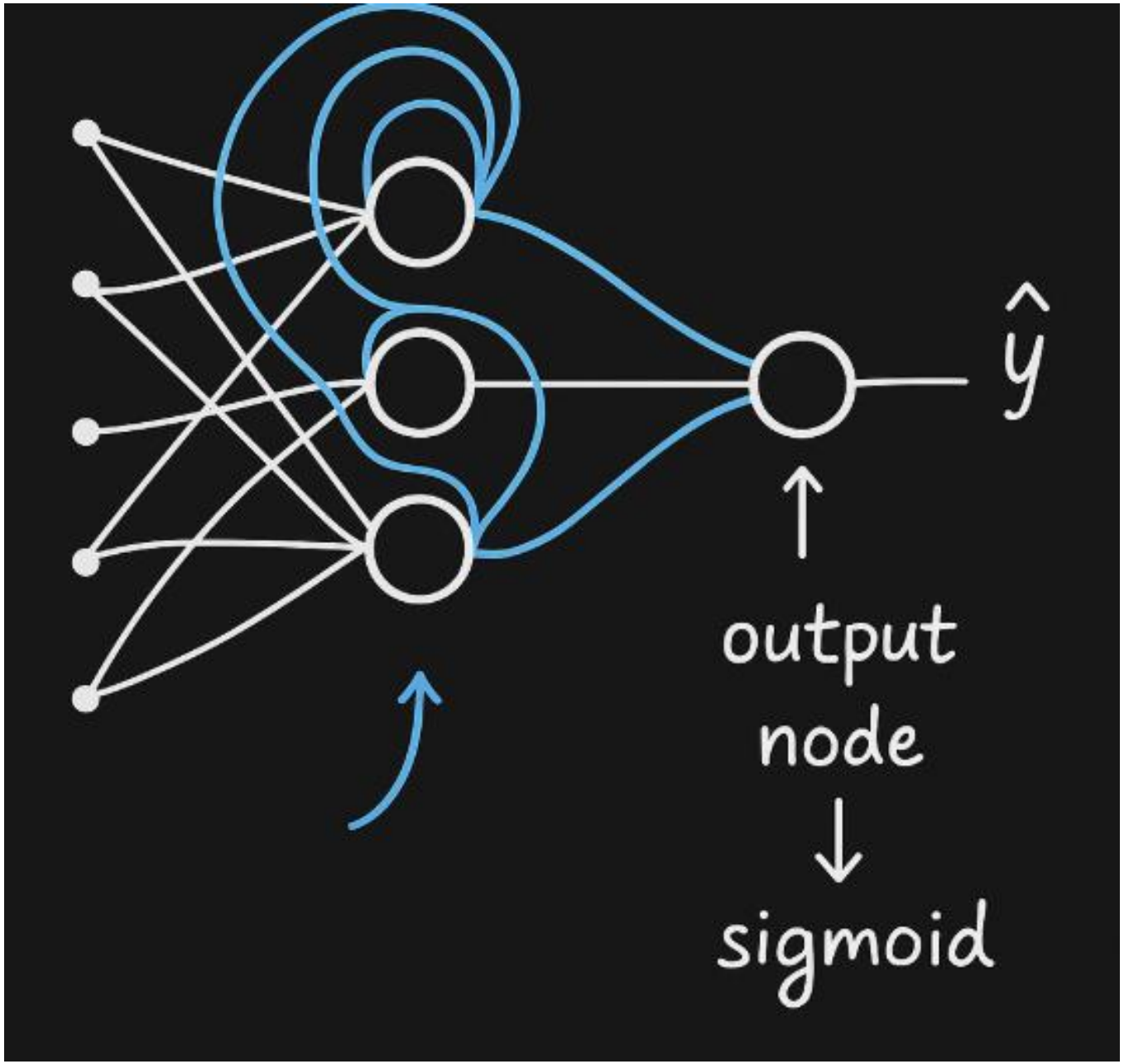
  t-=3 → x3
- 2nd Difference
  - ANN is feed forward NN

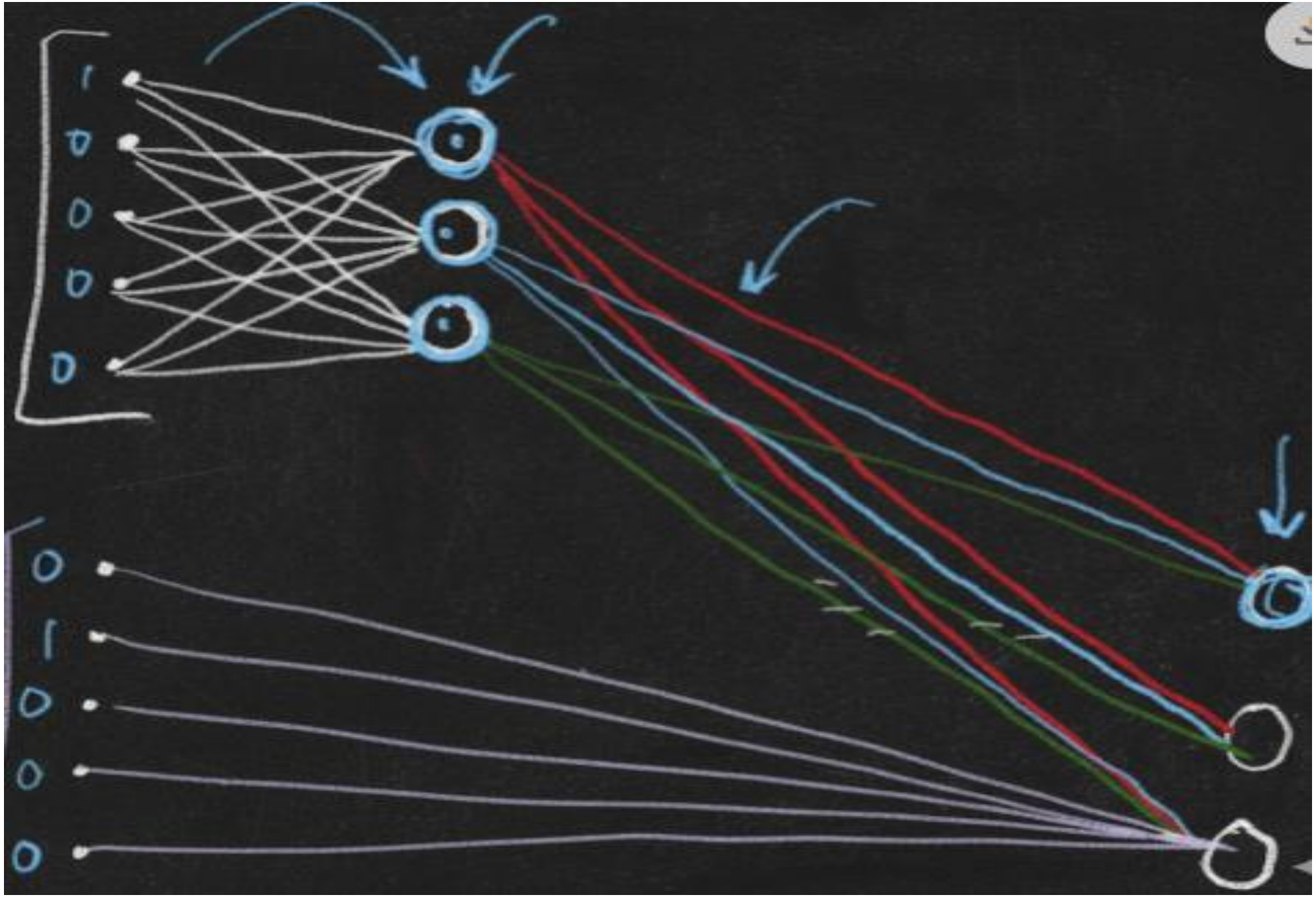
Difference in Structure of RNN& ANN

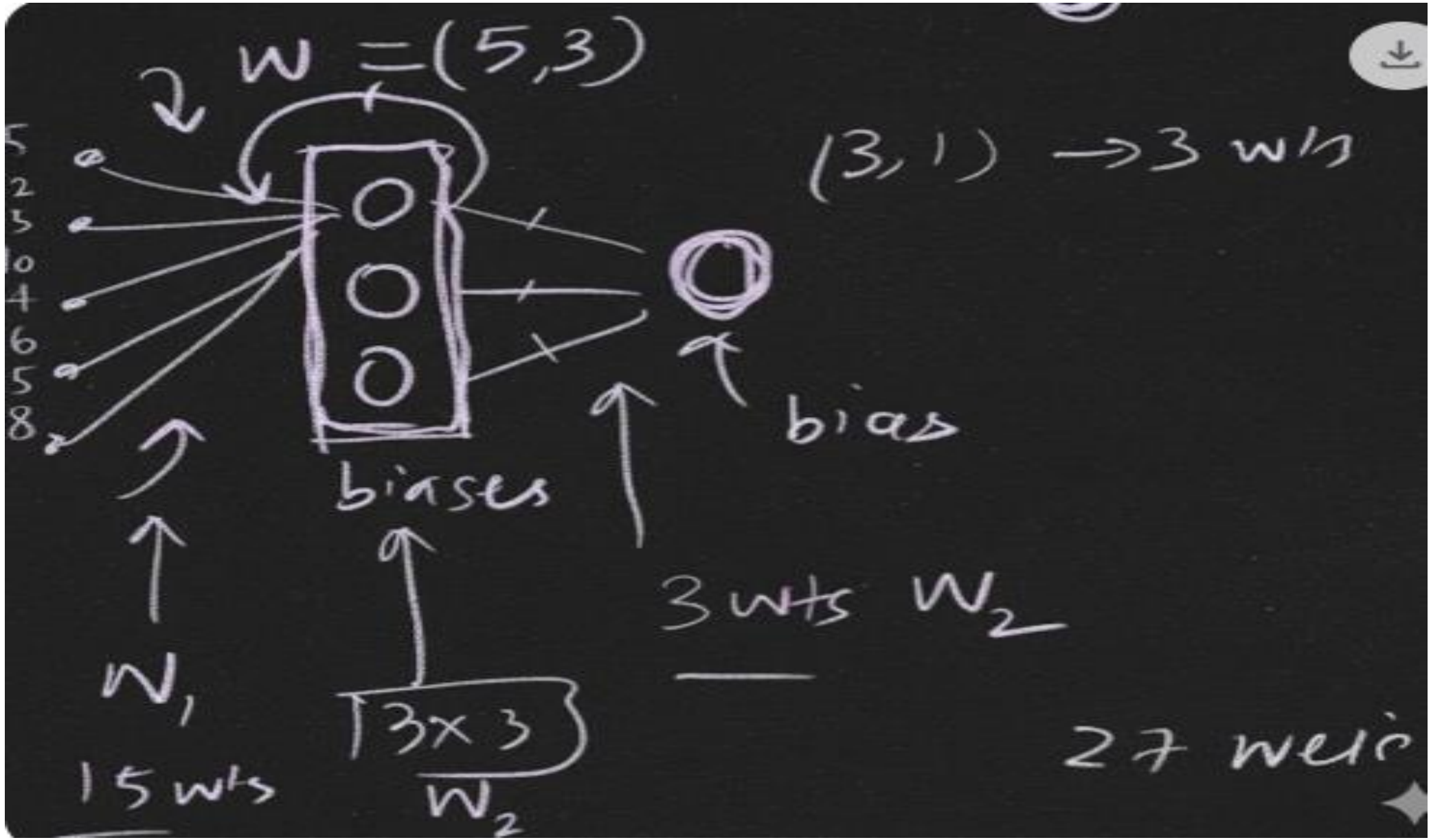
- RNN is not Feed forward
  - There is a concept of State
  - Hidden layer return back to its self that create the big difference




$\hat{y}$
output
node
sigmoid

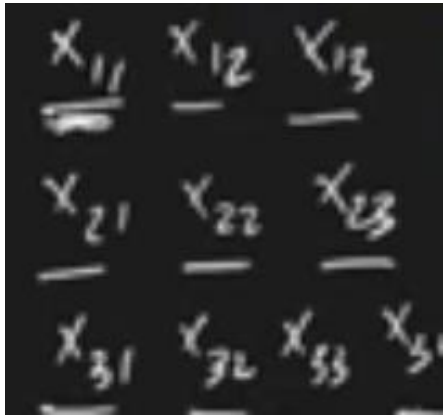

$\hat{y}$
output
node
sigmoid




2 W = (5,3)
(3,1) → 3 wts
bias
biases
3 wts W2
W1
15 wts
3x3
W2
27 weig

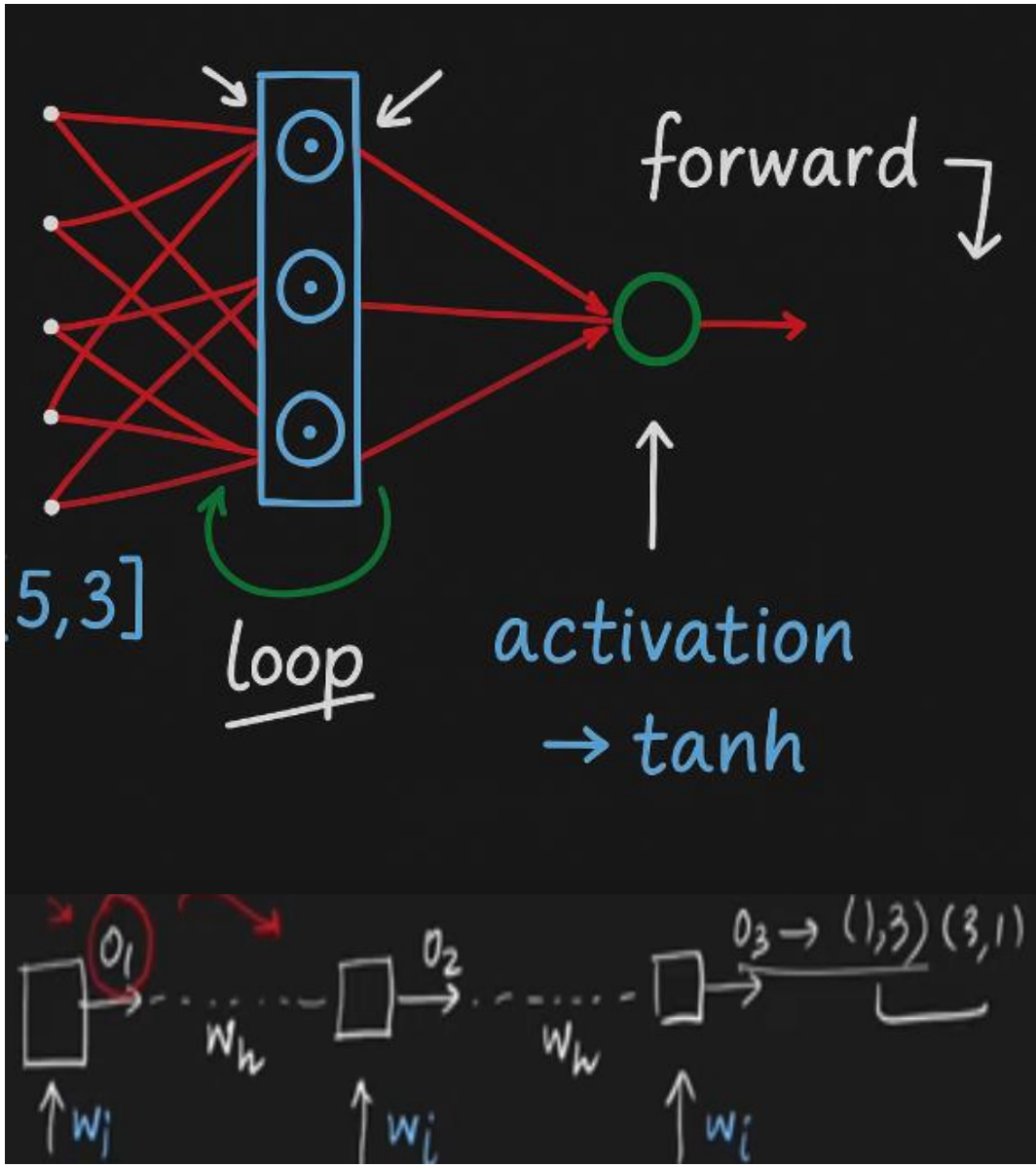
RRN-Forward Propagation

X11(5,1)
Wi(5,3)
At t=1
F(x11, w1)=(1,3)→O1
At t=2
Wh=(3,3)
F((x12,w1)+(O1,wh))
X12=(1,5)
Wi(5,3)
Res=1,3
O1(1,3)

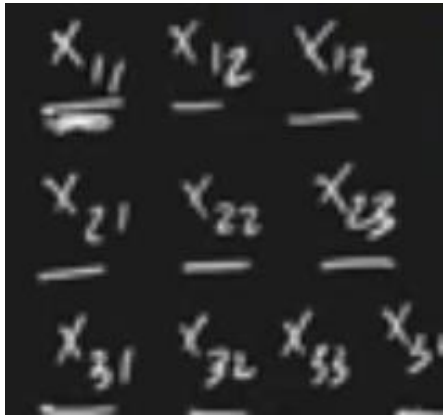


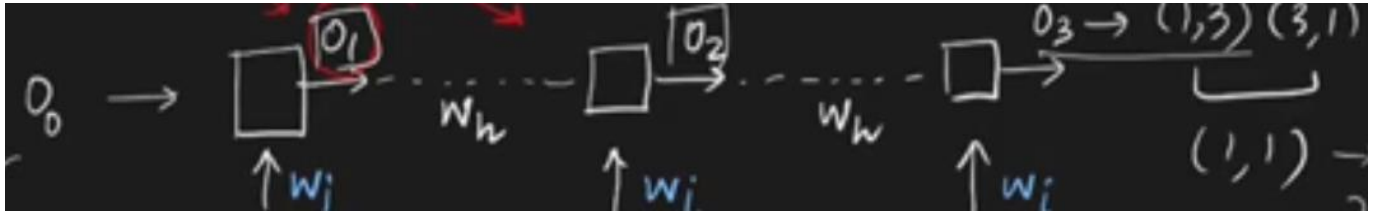
Wh=(3,3)
res2=1,3
02=(1,3)+(1,3)

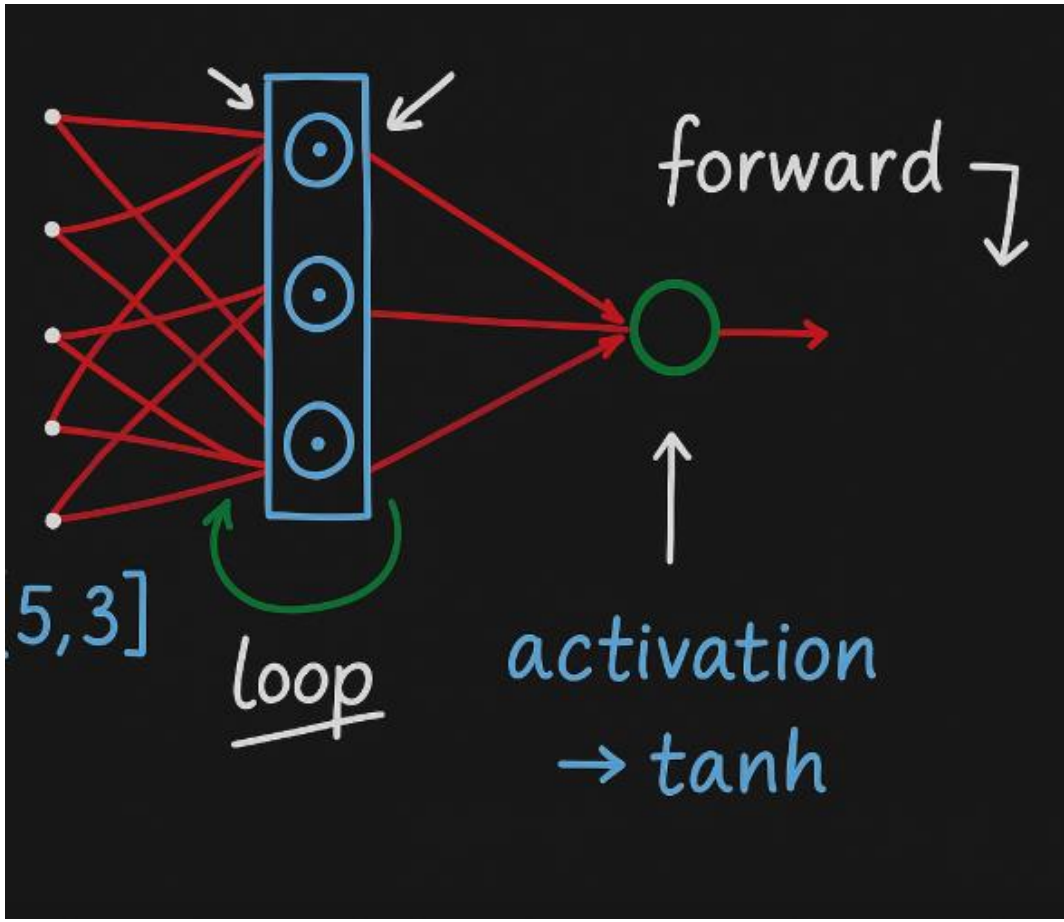
At t=3
X13
O3=F((x13,wi)+(o2,wh))
O3=(1,3)
Now
Wo=(3,1)
Final Output=Sigmoid(1,1)



RRN-Forward Propagation





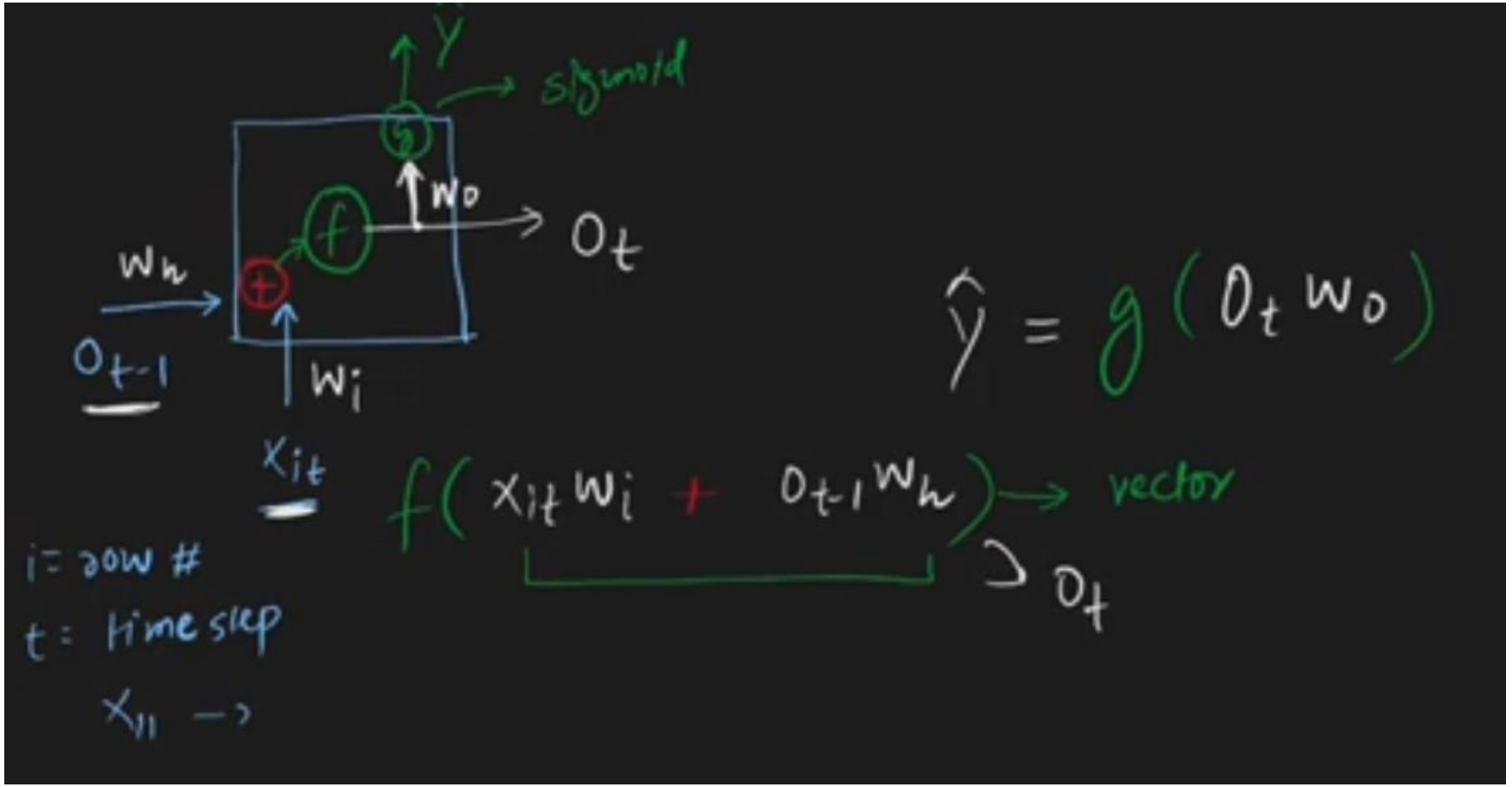


O0=[0,0,0]

Final Formula=F(x11*Wi+ O0 *Wh)

- **Only One layer is used again and again -→ Recurrent NN.**
- **Almost reminder last ten steps**
- **Very good on sequence data**

Simplified Representation

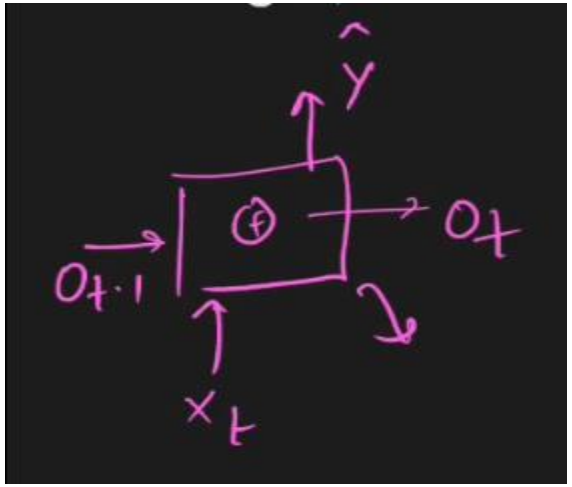


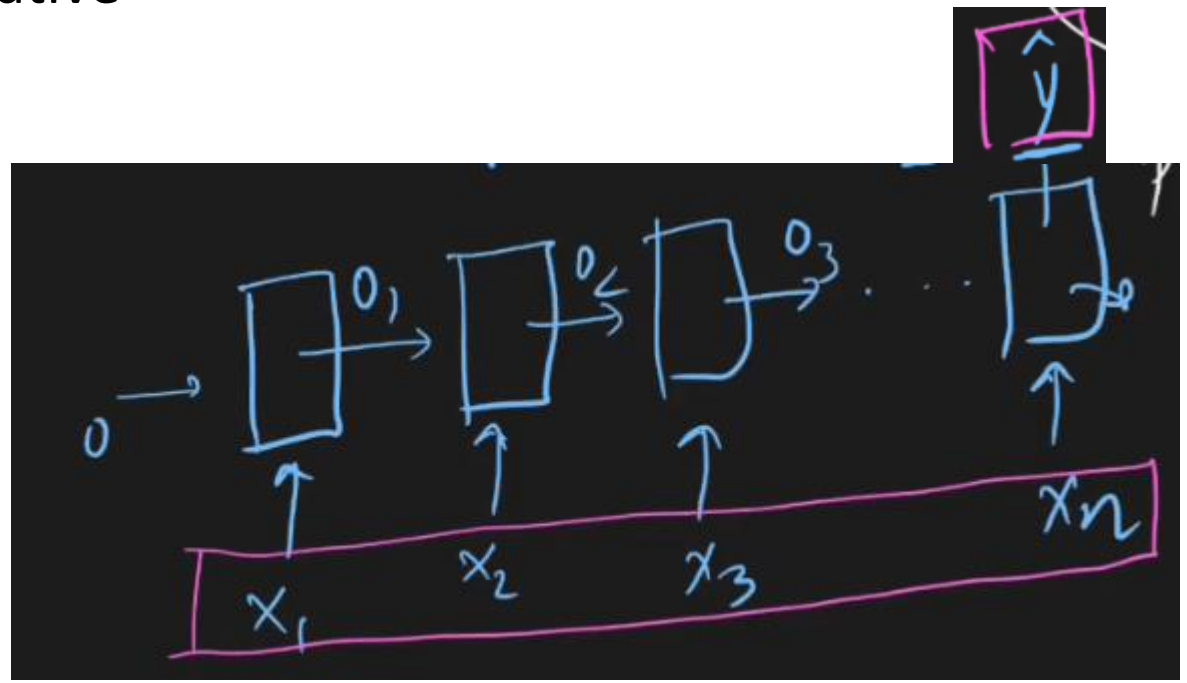
Types of RNNs

- Back propagation varies on different types of RNN.
- 4 Types of RNN
  - Many to one
  - One to many
  - Many to many
  - One to one

Many to one

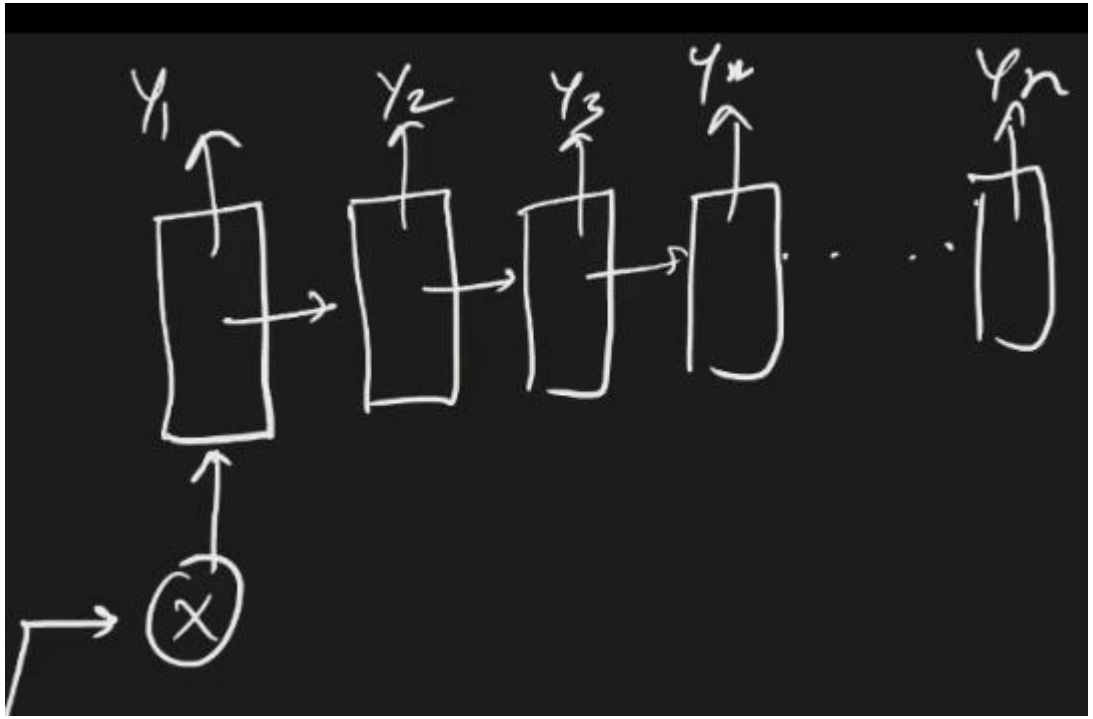
- Input =sequence
- Output= Non sequence(integer, scalar)
  e.g.
  - Sentiment Analysis→ Positive/Negative
  - Rating Prediction-> ****





One-to-Many

- Input=non sequential(number , image)
  output= Sequential



One-to-Many

- Input=non sequential(number , image)
  output= Sequential
  - Image Captioning
- Input= image of a person play cricket
- Output=“Man is playing cricket”
- Another example:

  music generation

Many-to-Many

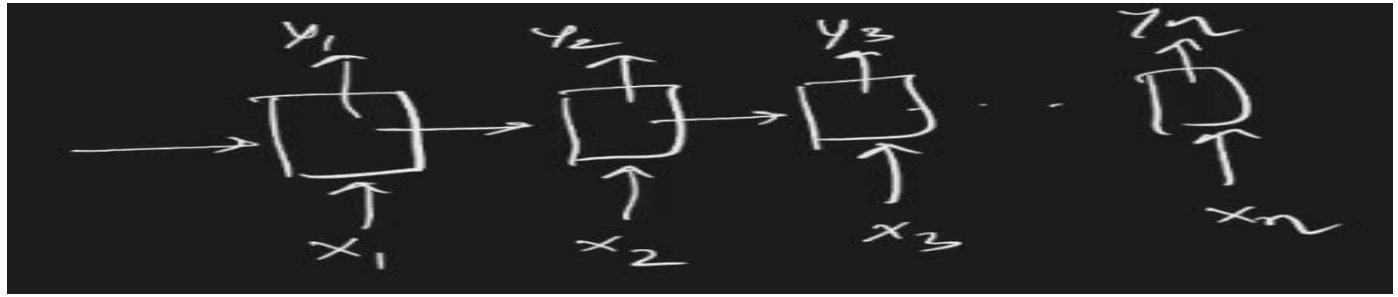
- Input =sequential data
- Output =sequential data
- Also called Seq2Seq
- Two Types:

  Same Length

  - Input size=equal to output size

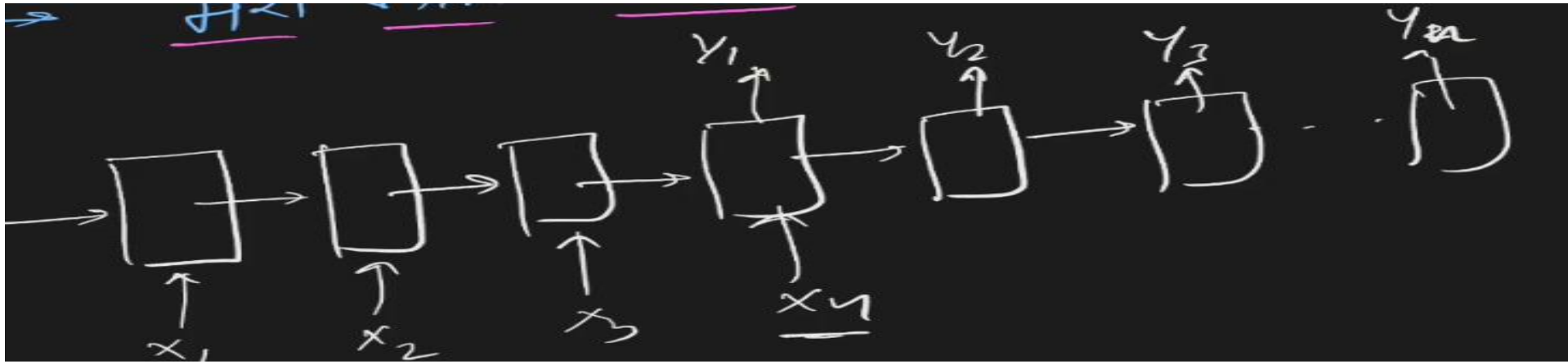
My	Name	Is	Safyan
Pronoun	Noun	Verb	Noun

  - Name Entity Recognition



Many-to-Many

- Variable Length
  input size!=output size
- Language Translation
- The boy is playing with a red ball.
- لڑکا ایک سرخ گیند کے ساتھ کھیل رہا ہے
- Encoder/Decoder



One-to-One

- Input=Non Sequential
- Output=non Sequential
  - Image Classification

RNN-Back Propagation

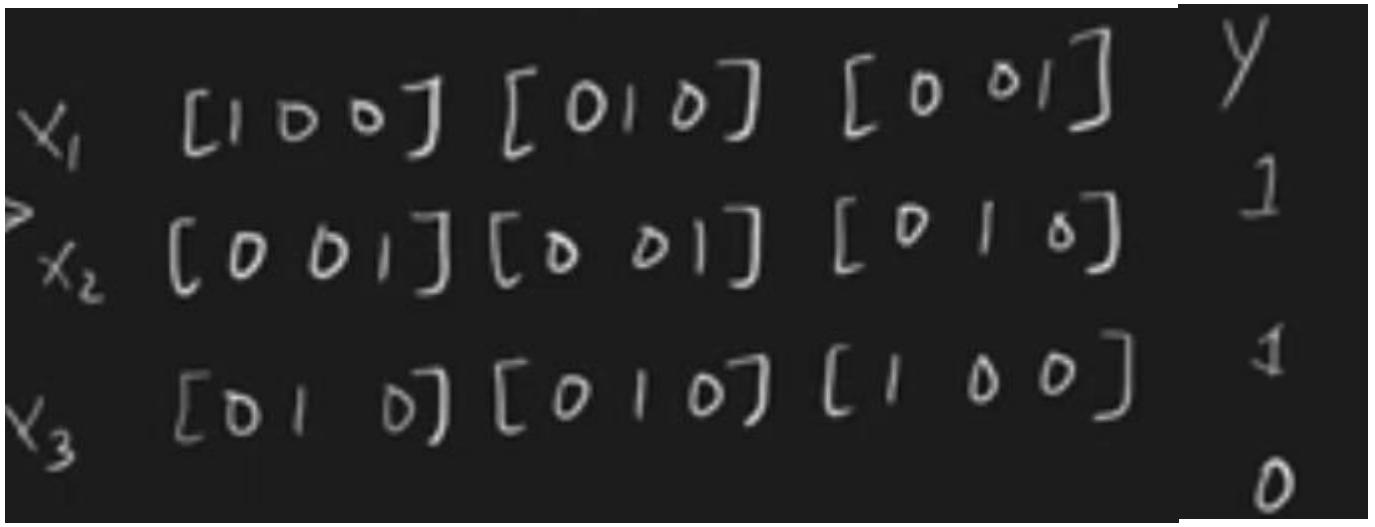
- Learning is attained with back propagation.
- Back Propagation Through Time(BPTT).
- Took an example of Many-to-One.
- Text---→ 1/0

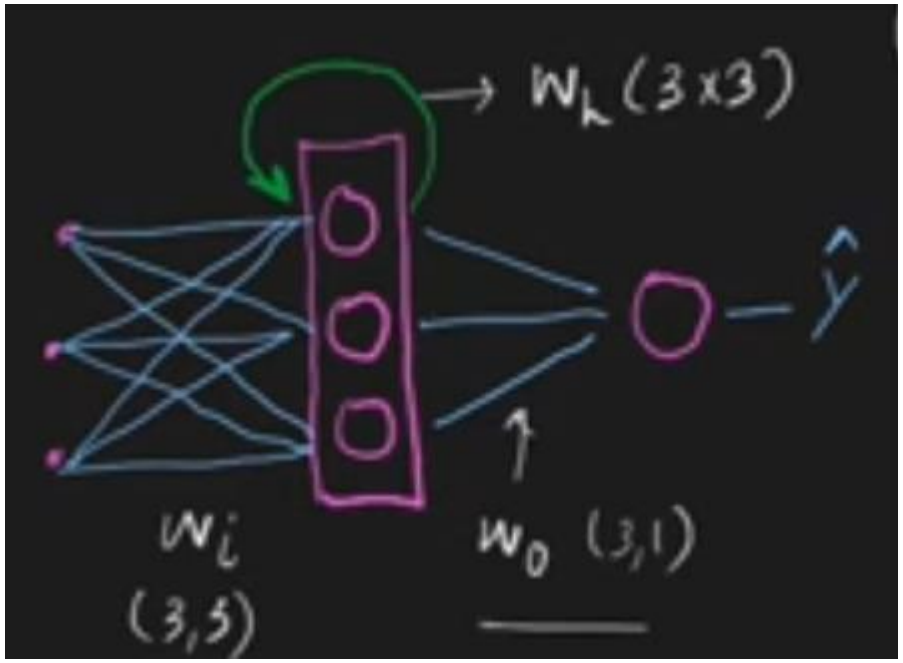
F1	F2	F3	Outptut
Cat	mat	rat	1
rat	rat	mat	1
mat	mat	cat	0

- First step: Convert text into embedding
  - Total words=3

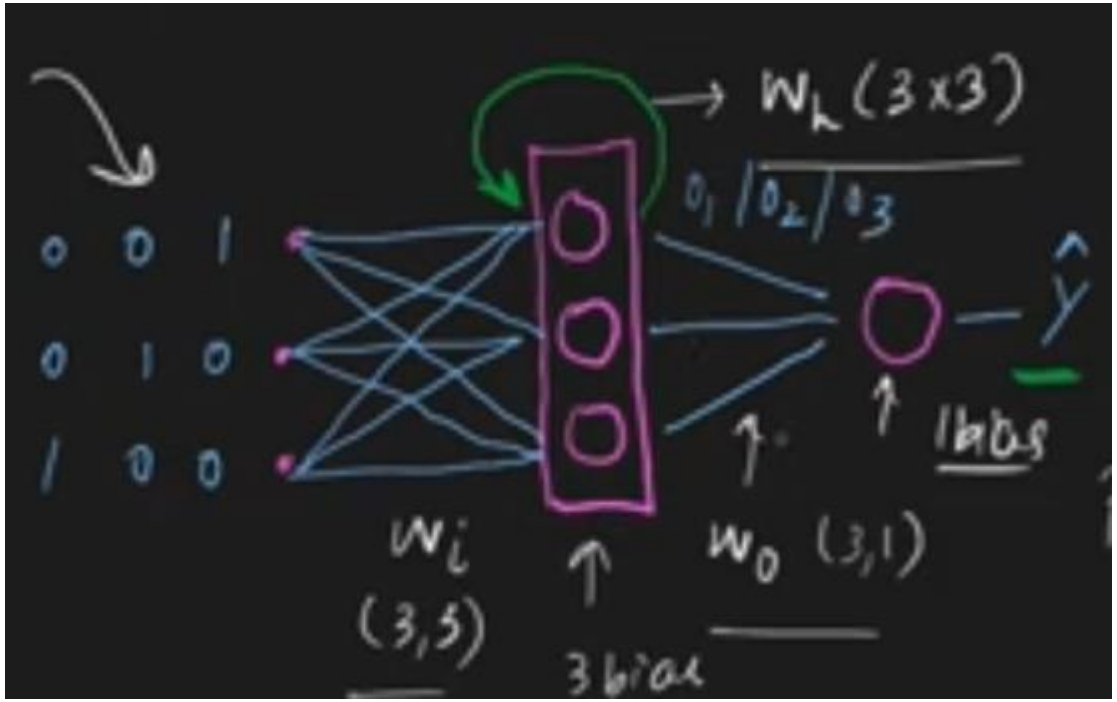
RNN-Back Propagation

- Cat=[1,0,0], Mat=[0,1,0] Rat=[0,0,1]

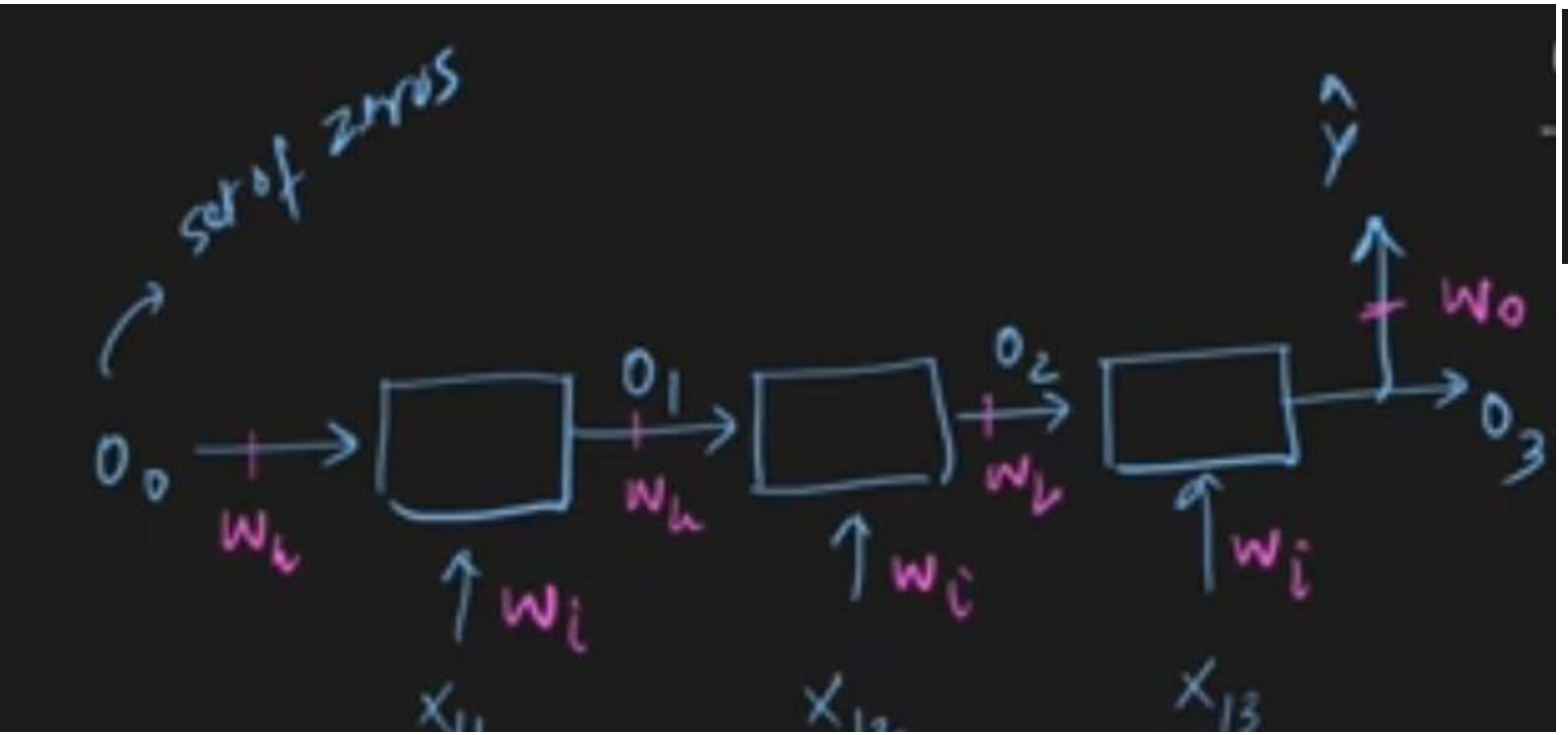


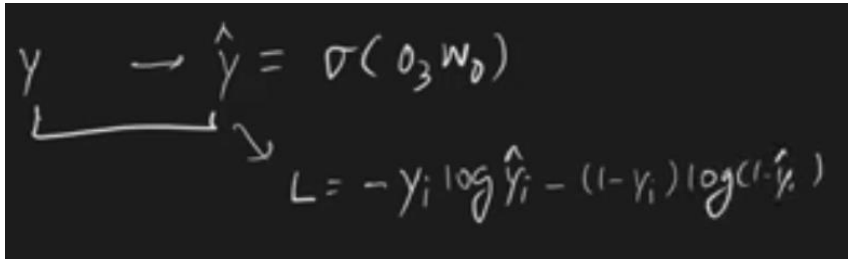


RNN-Back Propagation



RNN-Back Propagation





Final Formula=F(x11*Wi+ O0 *Wh)

- **Objective Function:**
  $Loss_{min}$ = by adjusting/tuning weights of w_i, w_h, w_o

  Possible with Gradient Descent

Gradient Descent

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

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

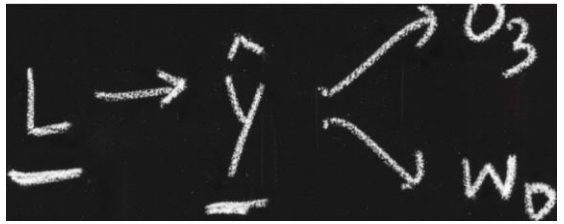
$$w_o = w_o - \eta \frac{\partial L}{\partial w_o}$$

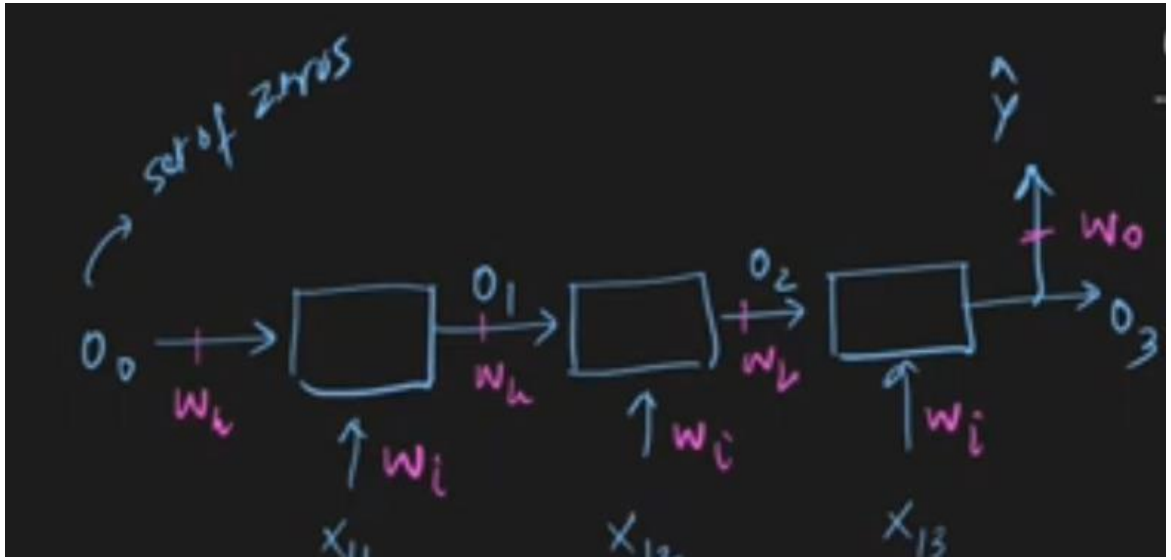
$$O_1 = f(x_{i1} w_i + O_0 w_h)$$
$$O_2 = f(x_{i2} w_i + O_1 w_h)$$
$$O_3 = f(x_{i3} w_i + O_2 w_h)$$
$$\hat{y} = \sigma(O_3 w_o)$$

- O_t: The **output** or hidden state at time step t.
- f: An **activation function** (e.g., tanh, ReLU).
- x_{it}: The **input** at time step t.
- w_i, w_h, w_o: **Weight matrices** for input, hidden (recurrent), and output layers, respectively.
- O_0: The initial hidden state or previous hidden state.
- $\hat{y}$: The **final prediction** or output.
- σ: A final activation function (e.g., **sigmoid** or **softmax**).

Gradient Descent

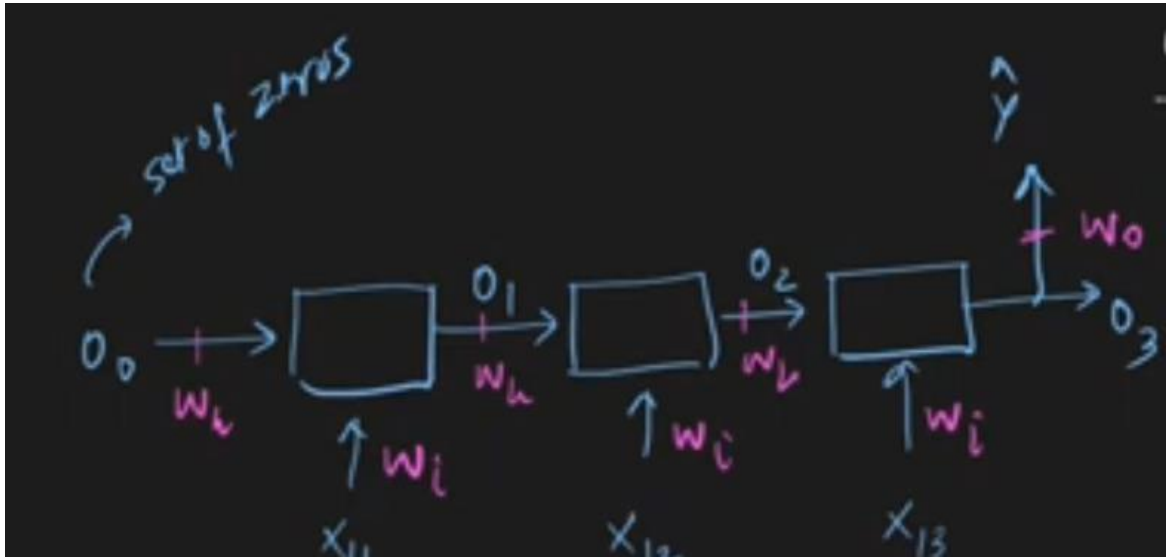
$$\frac{\partial L}{\partial w_o} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w_o}$$

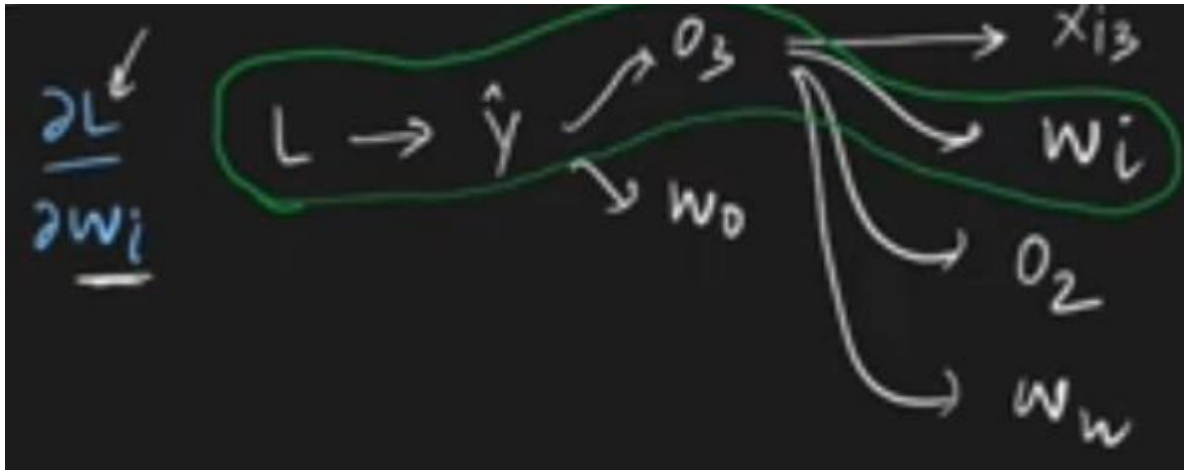


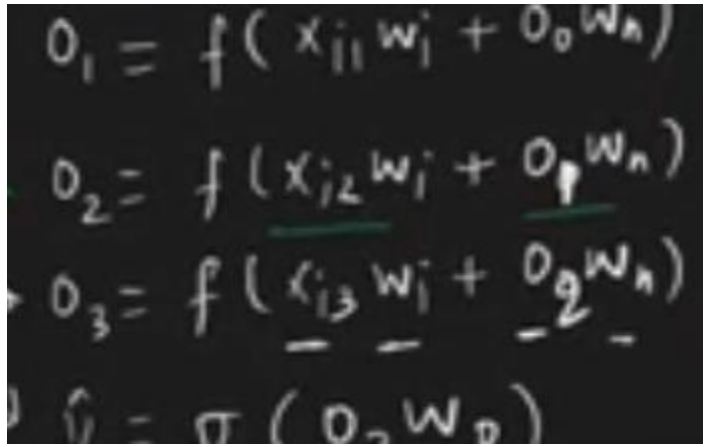


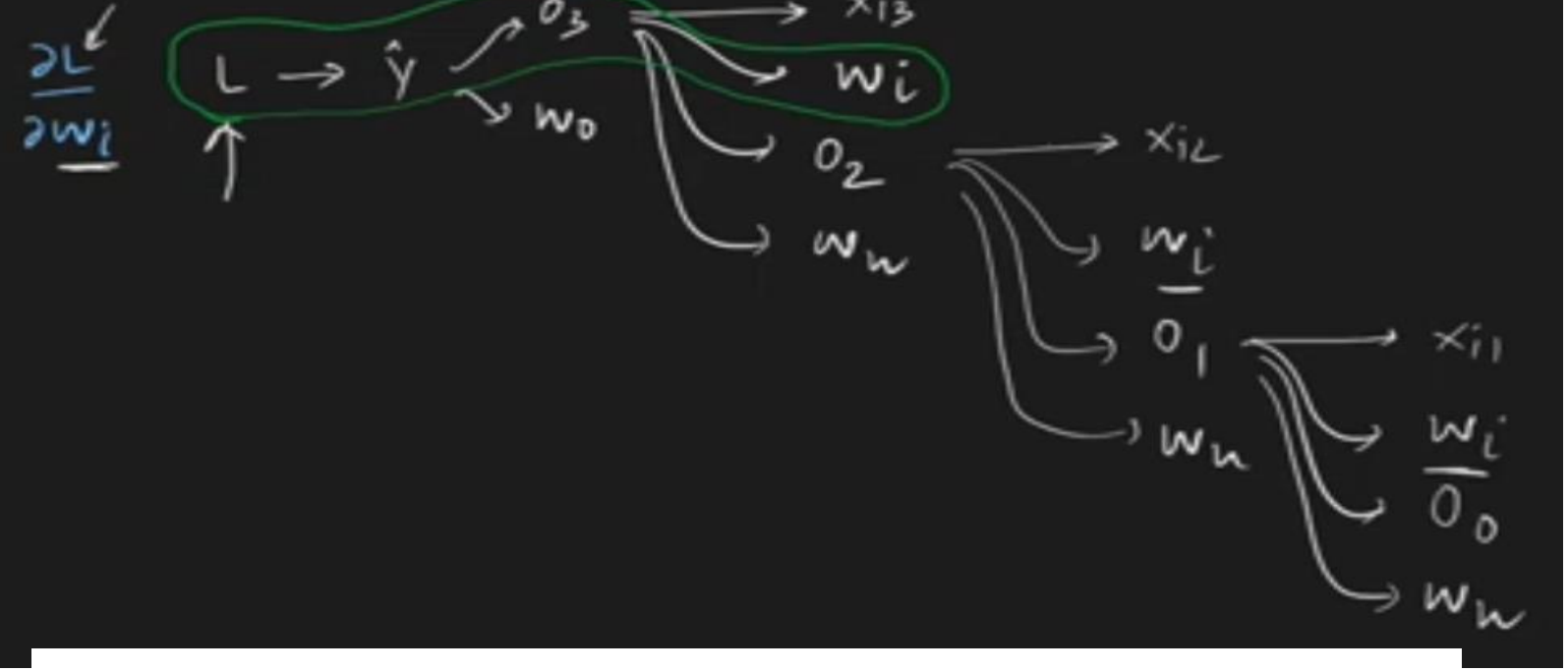
Gradient Descent

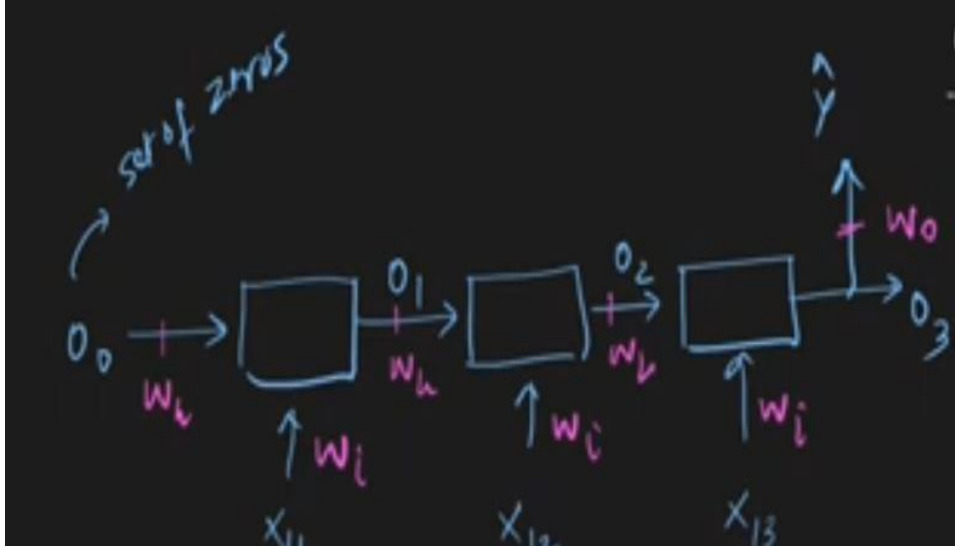
$$\frac{\partial L}{\partial w_i} =$$











$$O_1 = f(x_{i1}w_i + O_0w_h)$$
$$O_2 = f(x_{i2}w_i + O_1w_h)$$
$$O_3 = f(x_{i3}w_i + O_2w_h)$$
$$\hat{y} = \sigma(O_3w_o)$$

The total gradient is:

$$\begin{aligned}\frac{\partial L}{\partial w_i} &= \frac{\partial L}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial O_3}\frac{\partial O_3}{\partial w_i} && \text{(Path through } O_3 \text{ only)}\\ &+ \frac{\partial L}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial O_3}\frac{\partial O_3}{\partial O_2}\frac{\partial O_2}{\partial w_i} && \text{(Path through } O_2 \text{ and } O_3\text{)}\\ &+ \frac{\partial L}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial O_3}\frac{\partial O_3}{\partial O_2}\frac{\partial O_1}{\partial w_i} && \text{(Path through } O_1, O_2, \text{ and } O_3\text{)}\end{aligned}$$

Mathematical Expression (Summation Form)

The expression is:

$$\frac{\partial L}{\partial w_i} = \sum_{j=1}^{3} \frac{\partial L}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial O_j}\frac{\partial O_j}{\partial w_i}$$

Mathematical Expression (Generalized BPTT)

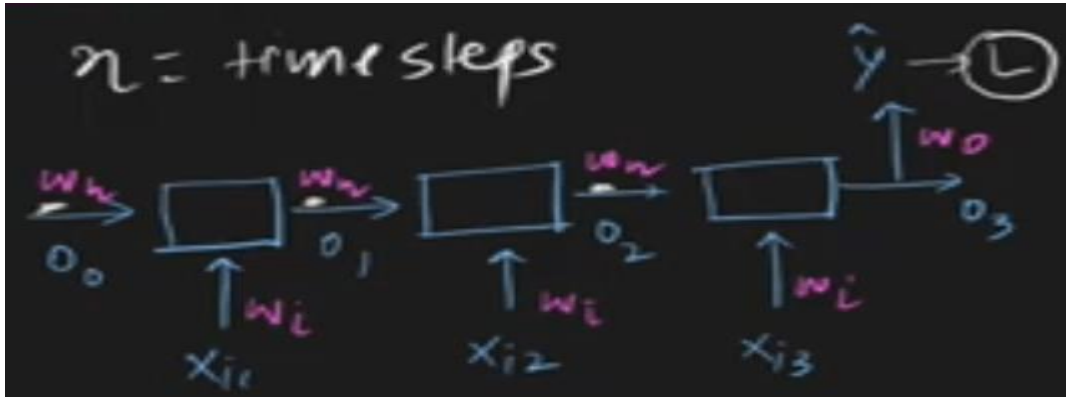
The expression is:

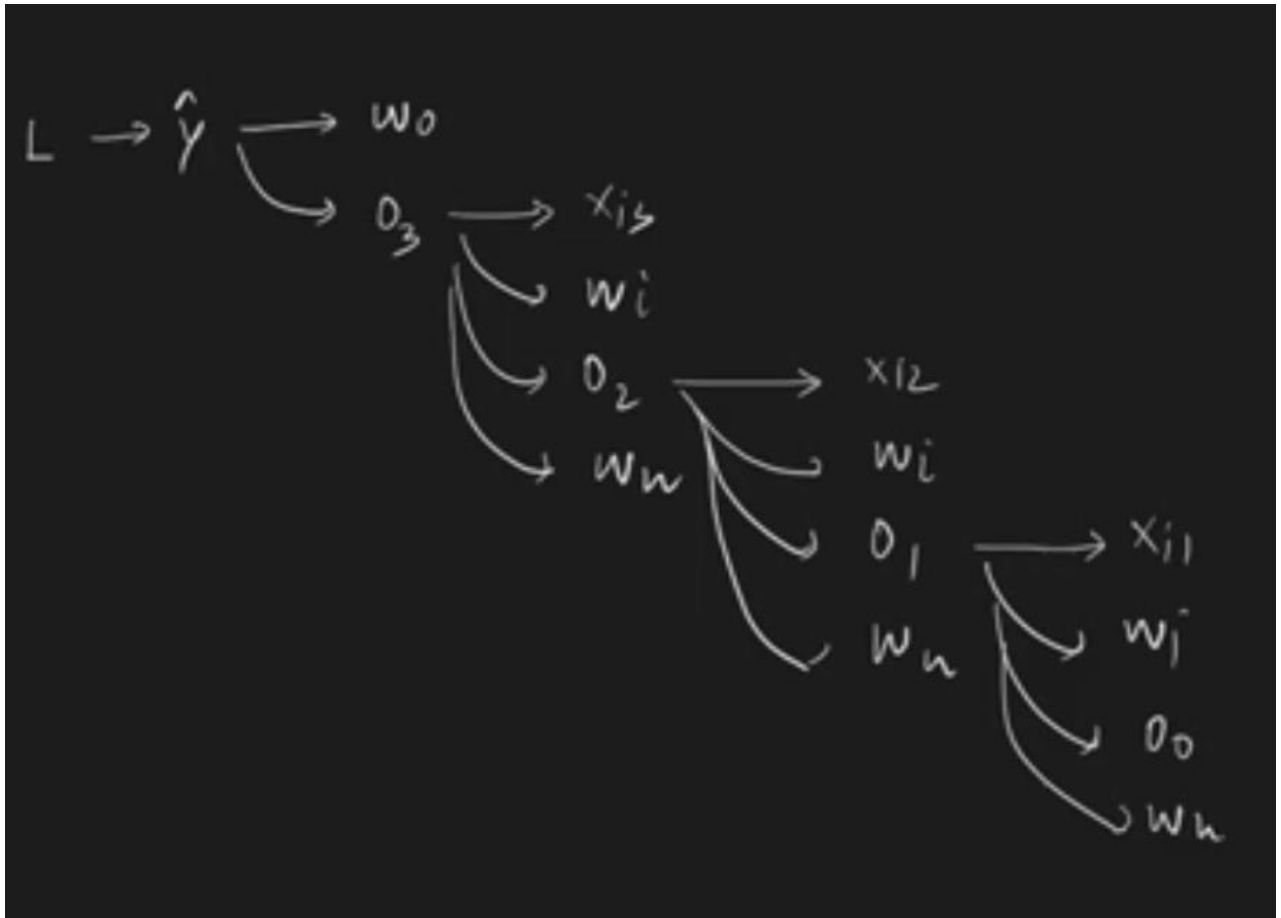
$$\frac{\partial L}{\partial w_i} = \sum_{j=1}^{n} \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial O_j} \frac{\partial O_j}{\partial w_i}$$

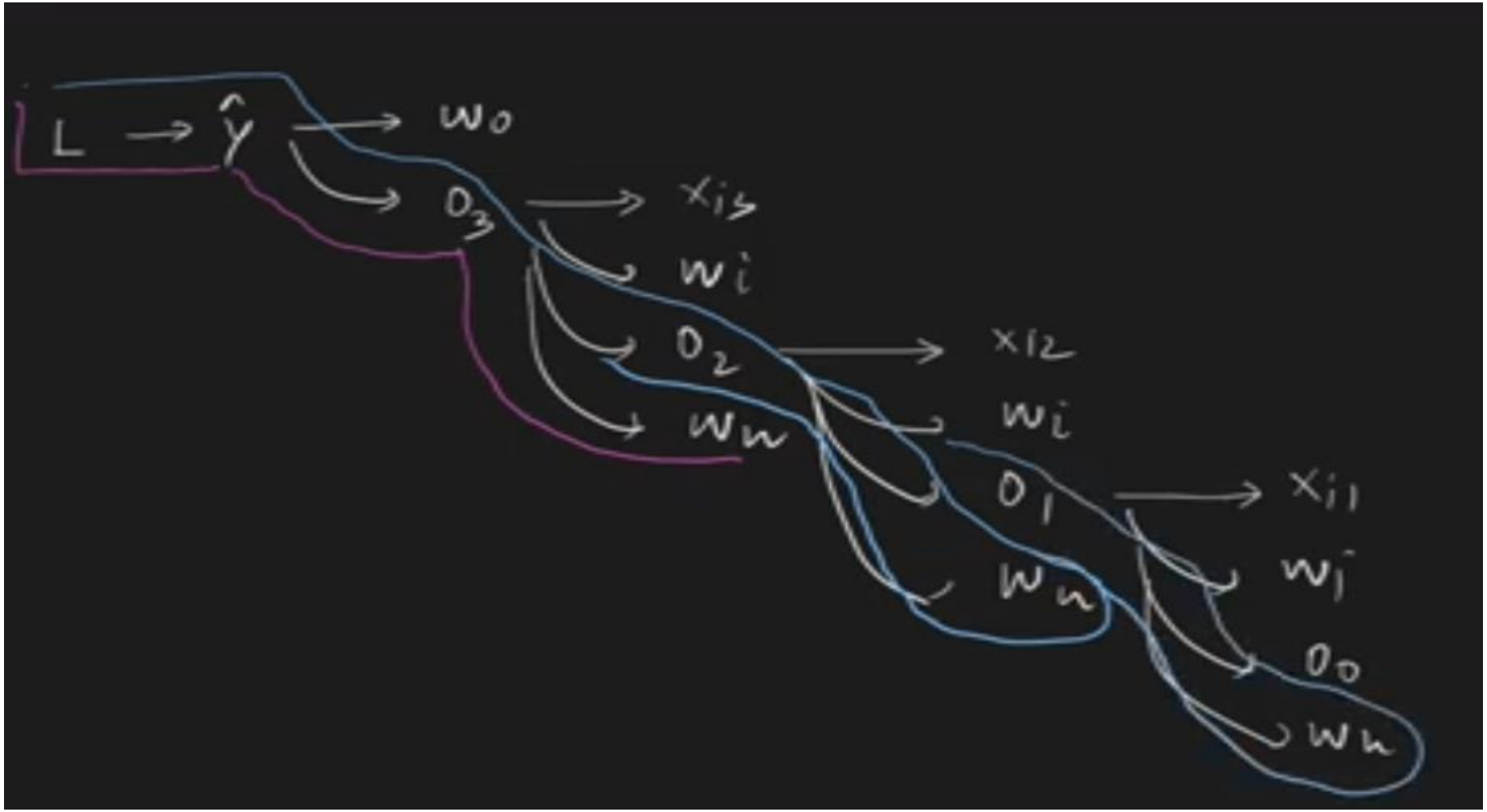
The image also provides the definition:

$$n = \text{time steps}$$

$$\frac{\partial L}{\partial w_h}$$






L
ŷ
w_0
o_3
x_{i3}
w_i
o_2
x_{i2}
w_i
w_n
o_1
x_{i1}
w_i
w_n
o_0
w_n

The total gradient $\frac{\partial L}{\partial w_h}$ is given by the sum of three terms (representing three time steps, O_1, O_2, O_3):

$$\begin{aligned}\frac{\partial L}{\partial w_h} &= \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial O_3} \frac{\partial O_3}{\partial w_h} && \text{(Path through } O_3 \text{ only)} \\ &+ \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial O_3} \frac{\partial O_3}{\partial O_2} \frac{\partial O_2}{\partial w_h} && \text{(Path through } O_2 \text{ and } O_3\text{)} \\ &+ \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial O_3} \frac{\partial O_3}{\partial O_2} \frac{\partial O_2}{\partial O_1} \frac{\partial O_1}{\partial w_h} && \text{(Path through } O_1, O_2, \text{ and } O_3\text{)}\end{aligned}$$

$$\frac{\partial L}{\partial w_h} = \sum_{j=1}^{n} \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial O_j} \frac{\partial O_j}{\partial w_h}$$