RNN (Recurrent Neural Networks)

(Recursion (call again and again))

(1980)

→ Recurrent Neural Networks are a powerful subset of machine learning algorithm that allow for the processing of sequential data by introducing feed back loops into the network architecture.

→ Vanishing Gradient : →

→ When the gradient becomes too small, the network may become unable to learn.

→ Exploding Gradient →

→ When the gradient becomes too large, the network may become unstable and fail to converge.

→ Solutions : →

→ Techniques such as weight initialization, gradient clipping, and careful network architecture design can mitigate these issues.

→ Application of RNNs →

(1) Language Modeling → Predicting the likelihood of the next word in a sequence.

(2) Machine Translation → Translating the text from one language to another

(3) Named Entity Recognition →

Identifying entities such as people, places, and organizations in text.

(4) Sentiment Analysis →

Determining the sentiment expressed in a piece of text.

* Challenges and Limitations ⇒

(1) Capturing Long-Term Dependencies ⇒

→ Vanishing gradient can make it difficult for RNNs to capture long-term dependencies.

(2) Training on Large Dataset →

RNNs require large amounts of data to train effectively, which can be time-consuming and resource-intensive.

* LSTM (Long-Short-Term Memory) →

LSTMs use a memory cell that can selectively forget or store information, allowing them to capture longer-term dependencies.

GRU (Gated Recurrent Unit) →

GRU simplify the LSTM architecture while still incorporating gating mechanisms to selectively update or pass on information.

Bi RNNs →

→ the outputs of the forward and backward RNNs are concatenated to form the final output.

* Deep RNNs →

→ Deep RNNs can capture more complex relationships in the data and improve performance.

* Stacking Layers →

→ Multiple RNN layers can be stacked to create a deeper network.

* Computational Cost : →

→ Deeper networks may require more computational resources, making them more difficult to train.

* Sequence to Sequence Learning : →

→ Encoder → the input sequence is encoded into a fixed-length representation.

→ Decoder → the output sequence is generated based on the encoded representation and optionally previous outputs.

* Attention Mechanisms →

① Global Attention → the decoder attends the entire input sequence at each time step.

② Local Attention → the decoder attends to a specific section of the input sequence at each time step.

③ Self Attention →

An attention mechanism where input is compared to itself to compute attention weights.

* Need for memory in Sequential Data →

The Challenge →

→ Traditional Recurrent Neural Networks (RNNs) suffer from vanishing or exploding gradients, making it hard for them to store long-term memory in sequential data models.

The Solution →

LSTMs address this challenge by introducing memory cells that keep track of past information, making them ideal for NLP applications such as machine translation.

* LSTM Cell Architecture →

① Input Gate → A gate that determines which values to update and how much.

② Forget Gate → A gate that removes irrelevant information from the cell state.

(3) Output Gate → A gate that controls the amount of information that flows out of the cell state.

(4) Memory Cell →

A cell that stores long-term dependencies.

* Vanilla RNN →

Has a simple architecture well-suited for small datasets, but can't store long-term dependencies in sequential data.

LSTM →

Has a more complex architecture that allows for the storage and retrieval of long-term dependencies in sequential data, making it ideal for NLP tasks.

* Peephole Connections in LSTM →

Peephole Connections →

Peephole connections are additional connections between the cell state and gates that directly interact with one another.

Q Why are they useful?

Peephole connections improve the LSTM's performance by allowing it to directly influence the gating mechanism's activation.

* Applications of LSTM in NLP →

(1) Sentiment Analysis using LSTM.

(2) Named Entity Recognition with LSTM

(3) Language Generation with LSTM.

* LSTM is an architecture that allows for the storage and retrieval of long-term memory to handle.

* NLP networks Bi LSTM →

Bidirectional LSTMs → Process sequences in both forward and backward directions, encoding past and future information.

* Bidirectional Sequential Data Analysis →

Challenges →

Sequential data often has complex dependencies and long-range connections that are difficult to capture using traditional machine learning approaches.

Solutions → Bidirectional LSTMs have taken NLP research by storm, enabling researchers to model complex sequences and make predictions with impressive accuracy.

* Architecture of Bidirectional LSTM →

Input layer → Accepts input sequences and processes them in a forward and backward direction.

LSTM layer → Composed of memory cells, forget gates, and input gates, BiLSTM layers are able to "remember" important features from past and future timesteps.

Output layer → Combine the output from the forward and backward LSTMs and outputs are prediction.

* Training Bidirectional LSTMs :→

(1) **Back propagation** → Optimizes the weights in the network by comparing predictions to actual outcomes.

(2) **Vanishing Gradient** → Occurs when the gradients in the network become too small to make sufficient changes to the weights.

(3) **L_1/ L_2 Regularisation** →

Prevents overfitting and improves generalisation by adding a penalty to the weight update equation.

Application →

(1) **Text Classification** → Classify text based on its content, such as sentiment analysis or spam detection.

(2) Named Entity Recognition → Identify important entities in text like people, organizations, and locations.

(3) Machine Translation → Translate text from one language to another based on contextual cues.

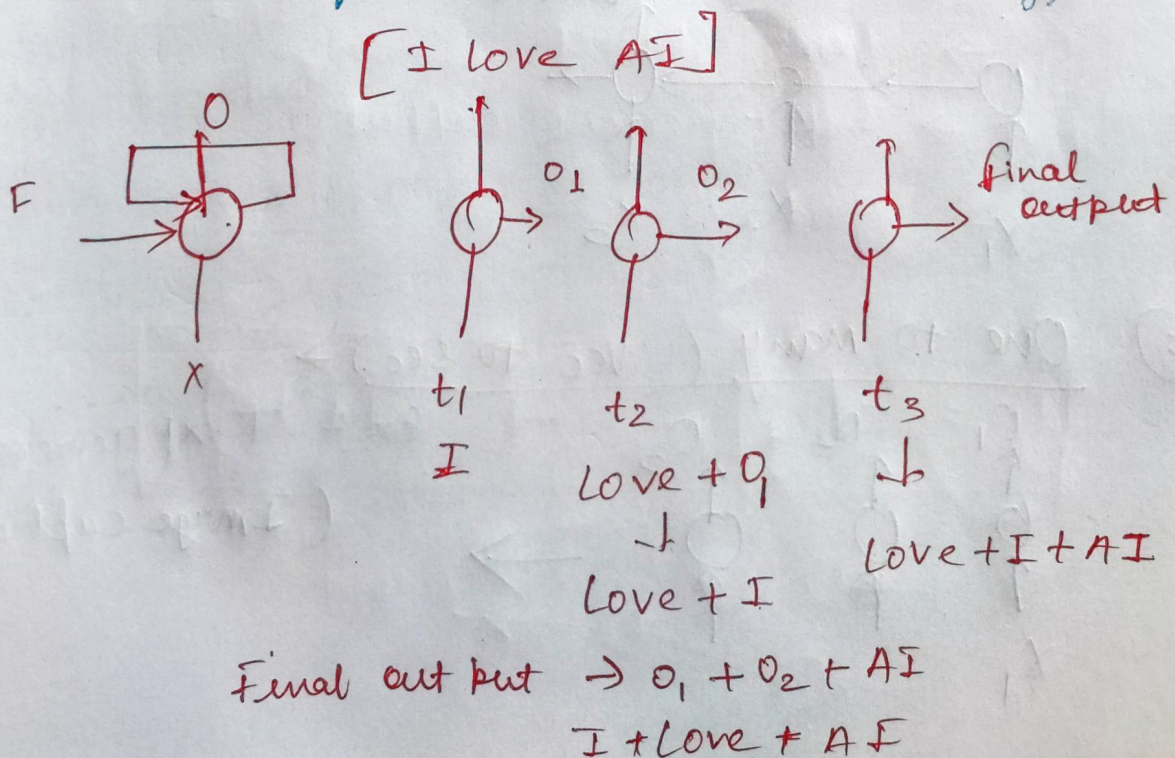
(4) Summarization → Create a shorter, more concise version of a text while preserving its meaning.

* RNN overcomes many problems →

(1) Text input → Varying Size

(2) Zero padding → (spause matrix) → Unnecesary computation

(3) Loosing sequential Info (Context Meaning)



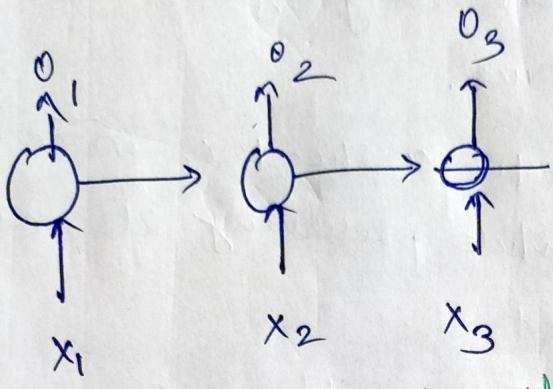
Final out put → $O_1 + O_2 + AI$

I + Love + AI

Types of RNN | Sequential Architecture →

(1) Many to Many Architecture → (See to see)

→ Feed many data and get many outputs.



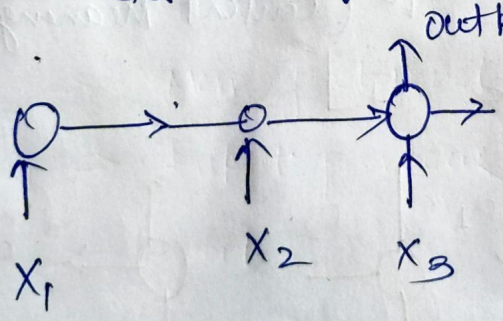
Ex → (Translation)

Ex :- Language translation

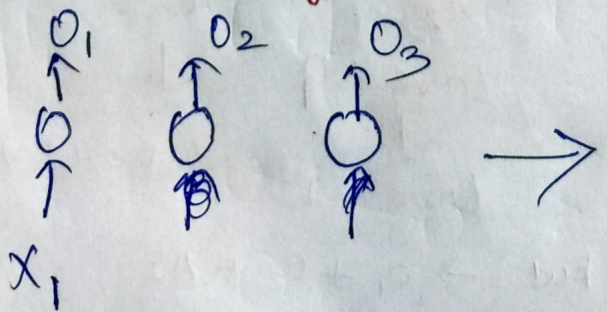
(2) Many to One (See to Vec) →

→ Feed many data and gets only one output

Ex :- Text classification / Sentimental Analysis)



(3) One to Many (Vec to See) →



Ex :- Text generation

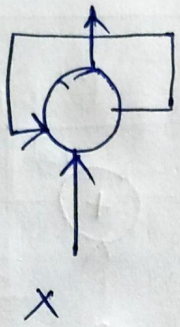
(Image captioning)

(4) One to one (Vector to Vector) →

→ Feed one input and gets one output.

(Vanilla neural network)

Ex:- RNN



X

→ Simple RNN is one to one

* Keras Embedding Layer →

→ By using this text Representation is performed.

→ This layer is used in RNN, LSTM & GRU.

* RNN can't remember longer context.

* LSTM overcomes the drawbacks of RNN.

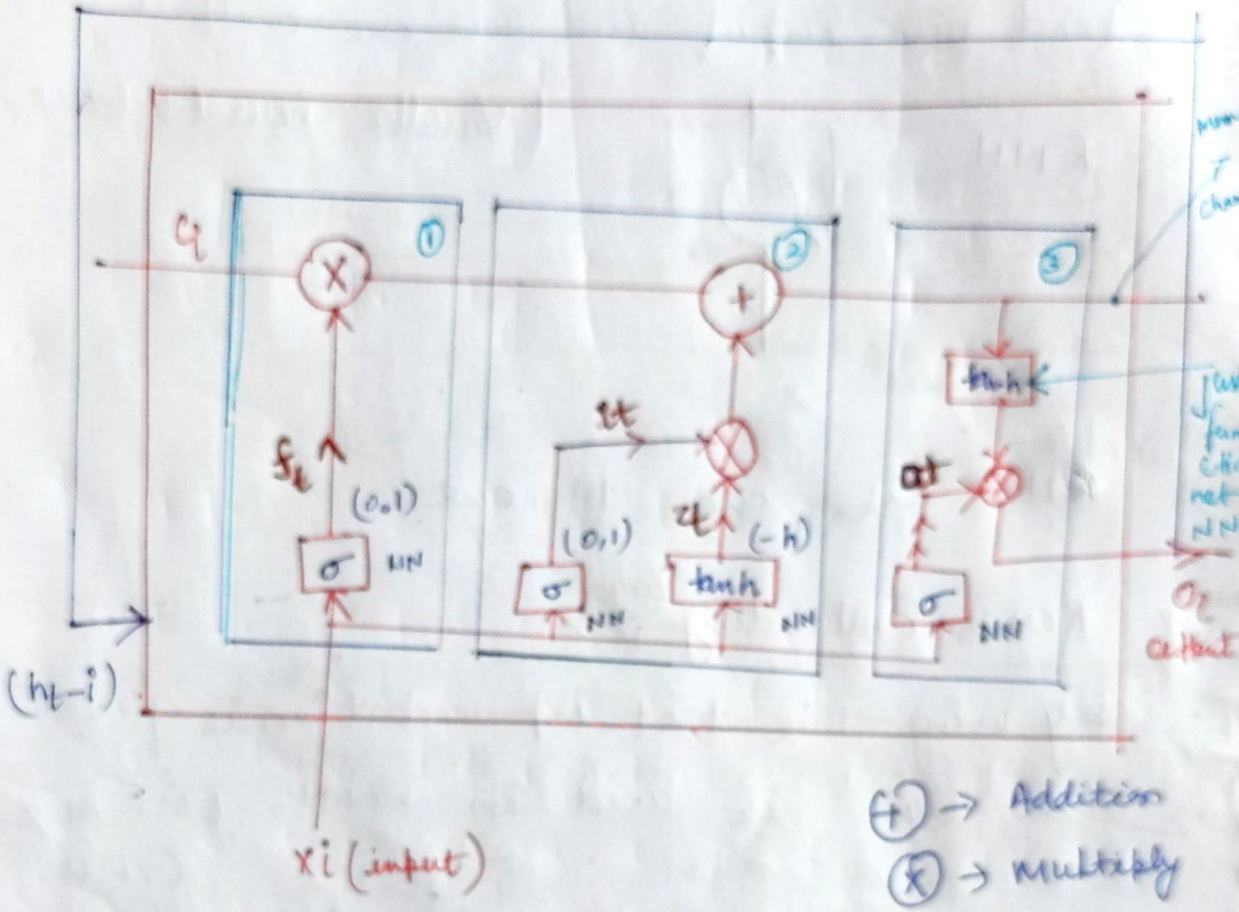
→ Components used in LSTM →

→ (1) forget gate

(2) Input gate or memory gate

(3) Output gate

LSTM Architecture



⊕ → Addition

⊗ → Multiply

x_i → input

o_i → output

c_t → Memory channel

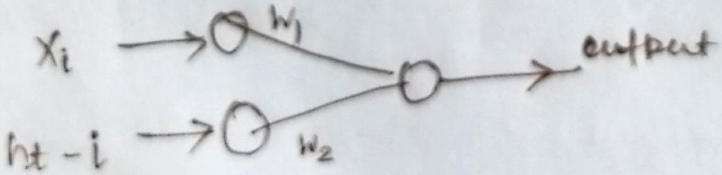
$h_t - i$ → Input previous output

σ → Sigmoid Activation fun^n

b_f → bias

W_f → Weight

$$f_t = \sigma\left(W_f\left[h_t - 1,\ x_t\right] + b_f\right)$$



@ $i_t = \sigma\left(W_t [h_{t-1}, x_t] + b_t\right)$

$Z_t = \tanh\left(W_t [h_{t-1}, x_t] + b_t\right)$

$O_t = \sigma\left(W_t (h_{t-1}, x_t) + b_t\right)$

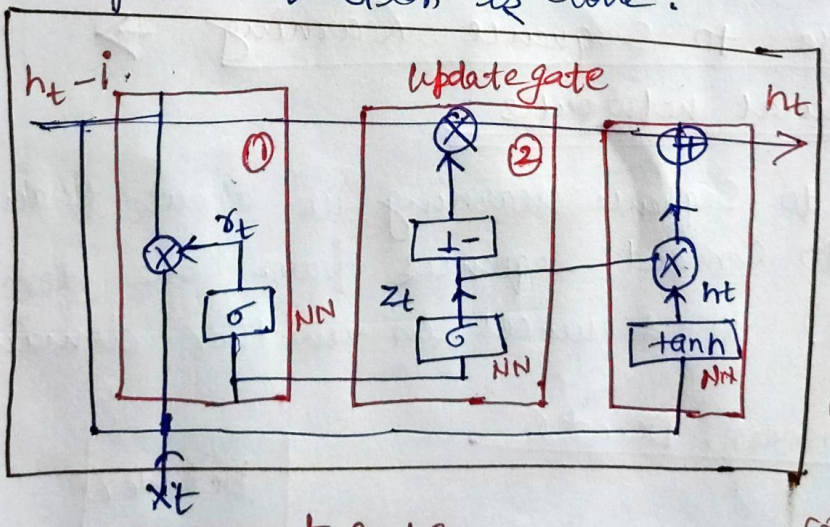
$O_i = O_t \times \tanh(C_t)$ → final output

Output

✳ GRU improves the drawback of LSTM

GRU (Gated Recurrent unit) →

↳ In GRU memory cell is not needed.
→ only classification is done.



① forgate gate

② Updategate

r_t → The reset gate

Z_t → Update gating Vector

Ex:- Sentement Analysis

↓

It tells about good word or bad word.

↓

GRU used mostly

Ex:- Chat Completion

Ex:- Hi how are you?

↓

Predict which context is important or not?

↓

LSTM used mostly

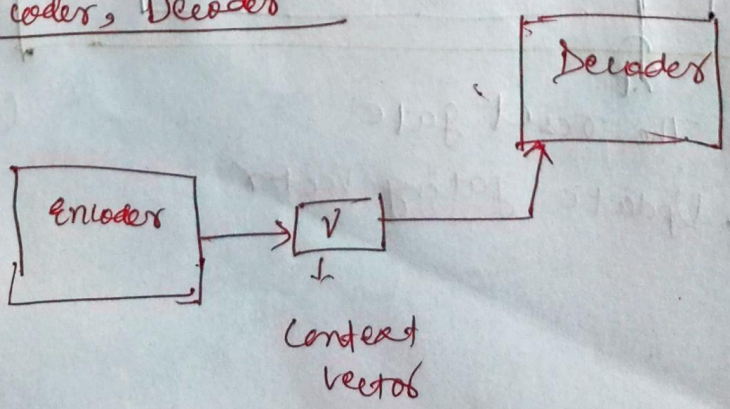
* Hate Speech Classification ⇒

→ Hate speech detection is the task of detecting if communication such as text, audio, and so on contains hatred and or encourages violence towards a person or a group of people.

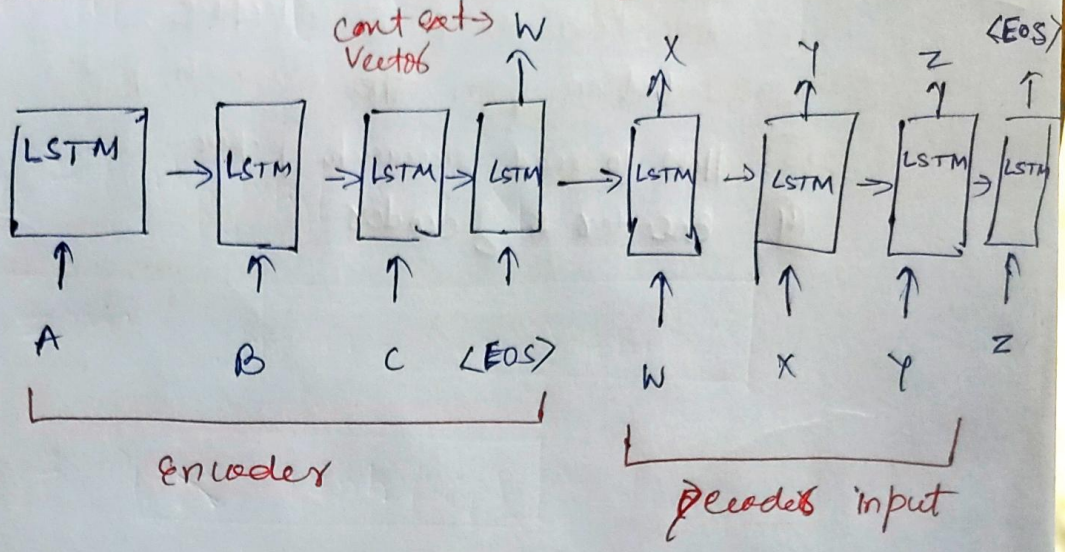
* Sequence to Sequence learning → in neural networks

→ Sequence to Sequence learning is about training models to convert sequences from one domain i.e. (english) to sequences in another domain (French).

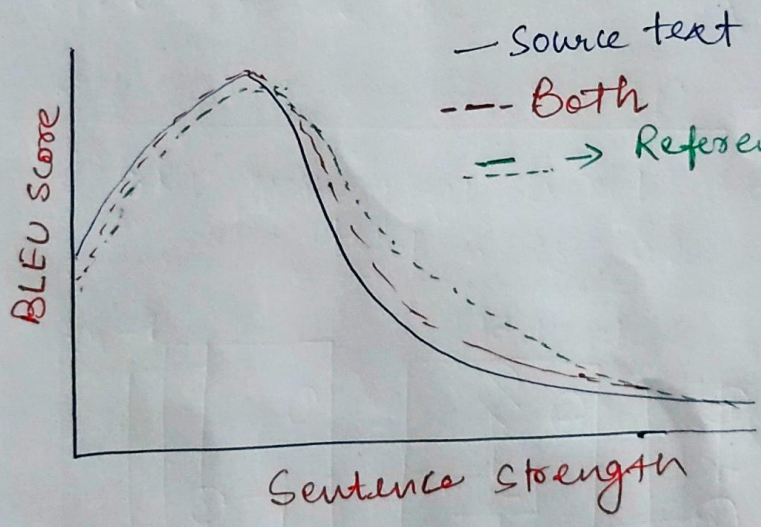
* Encoder, Decoder



Architecture of encoder and Decoder ⇒



* Problem with encoder & Decoder →



Sentence strength $\propto \frac{1}{\text{Blue score}}$