RNNs, GRUs, and LSTMs….

Making the world a better place through Artificial Intelligence!

Agenda

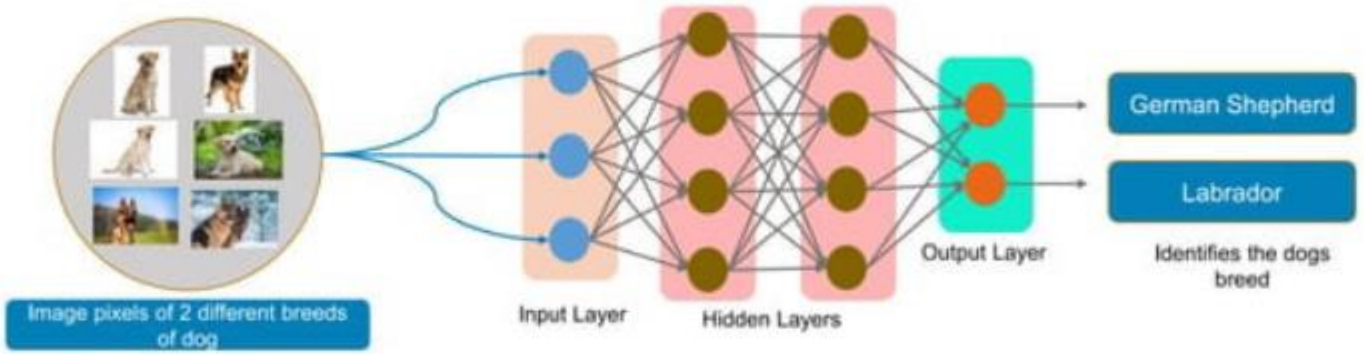
- Neural Network
- Challenges in Neural Network
- RNNs
- GRU
- LSTM
- Comparisons
  - RNN Vs LSTM
  - GRU Vs LSTM

Neural Networks

Making the world a better place through Artificial Intelligence!

Neural Networks (NN)

- NN consists of different layers connected to each other.
- NN learns from huge volume of data.
- It requires complex algorithms (Gradient Descent, Stochastic Gradient Descent, Adam, etc.) to train NN on a data.

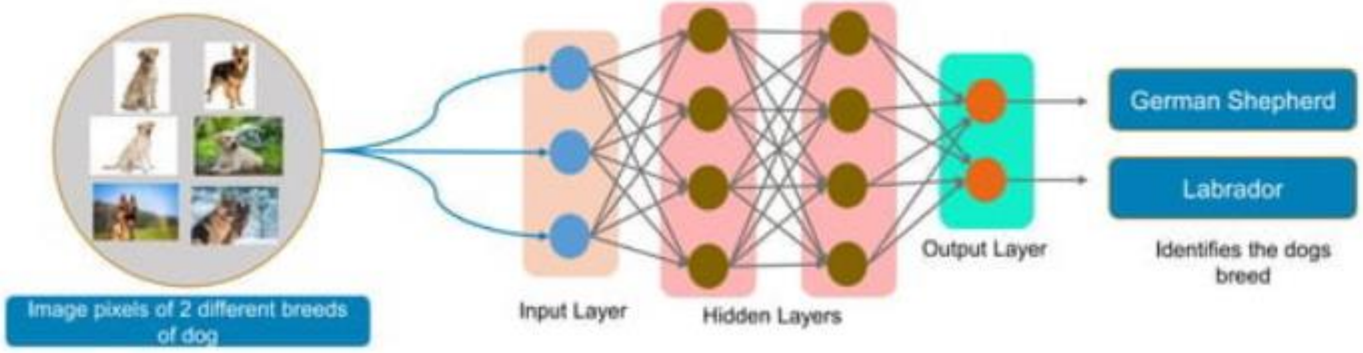


Neural Networks (NN)- Challenges

Making the world a better place through Artificial Intelligence!

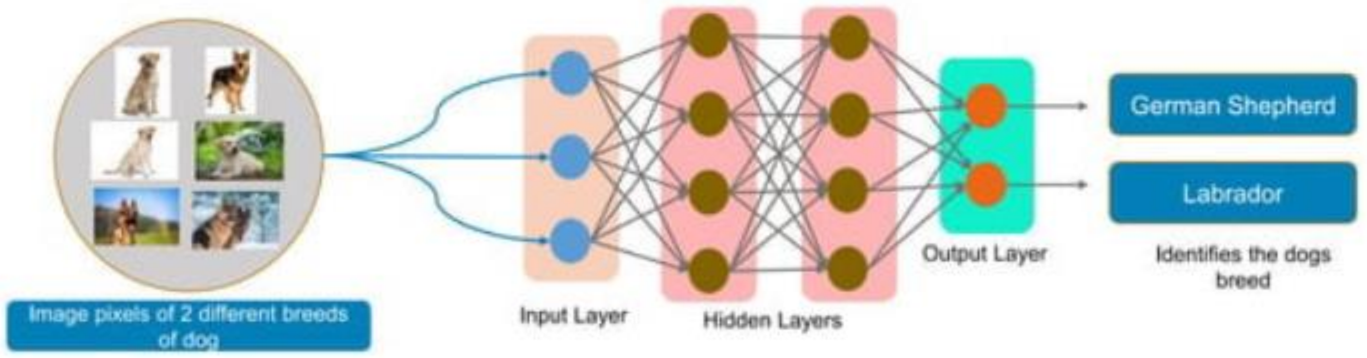
Neural Networks (NN)- Challenges

- Decision of NN are based on single input.
- They don’t remember past input.
- In case of NLP, a sentence is made of multiple words where each word depends on other words to make sense. But, NN can handle one input at a time (word).



Neural Networks (NN)- Challenges

- NN cannot handle sequential data (such as text).
- Cannot memorize previous inputs.
- Considers only current input.



But how to represent words ?

- One-hot encoding or
- Get already learned Embedding from existing methods such as Glove, word2vec, etc.
- Embedding are nothing but numerical features.

The cat sat on the mat

The: [0 1 0 0 0 0 0]

cat: [0 0 1 0 0 0 0]

sat: [0 0 0 1 0 0 0]

on: [0 0 0 0 1 0 0]

the: [0 0 0 0 0 1 0]

mat: [0 0 0 0 0 0 1]

Notice that in the image to the left the words 'The' and 'the' have different

One-hot encoding example

Word	Word Embedding						
man →	0.6	-0.2	0.8	0.9	-0.1	-0.9	-0.7
women →	0.7	0.3	0.9	-0.7	0.1	-0.5	-0.4
king →	0.5	-0.4	0.7	0.8	0.9	-0.7	-0.6
queen →	0.8	-0.1	0.8	-0.9	0.8	-0.5	-0.9

Word and its already learned embedding example

Note: if one-hot encoding is used, then we need to use embedding layer ourselves (using pytorch or keras)

ICRL
Intelligent Criminology Research Lab

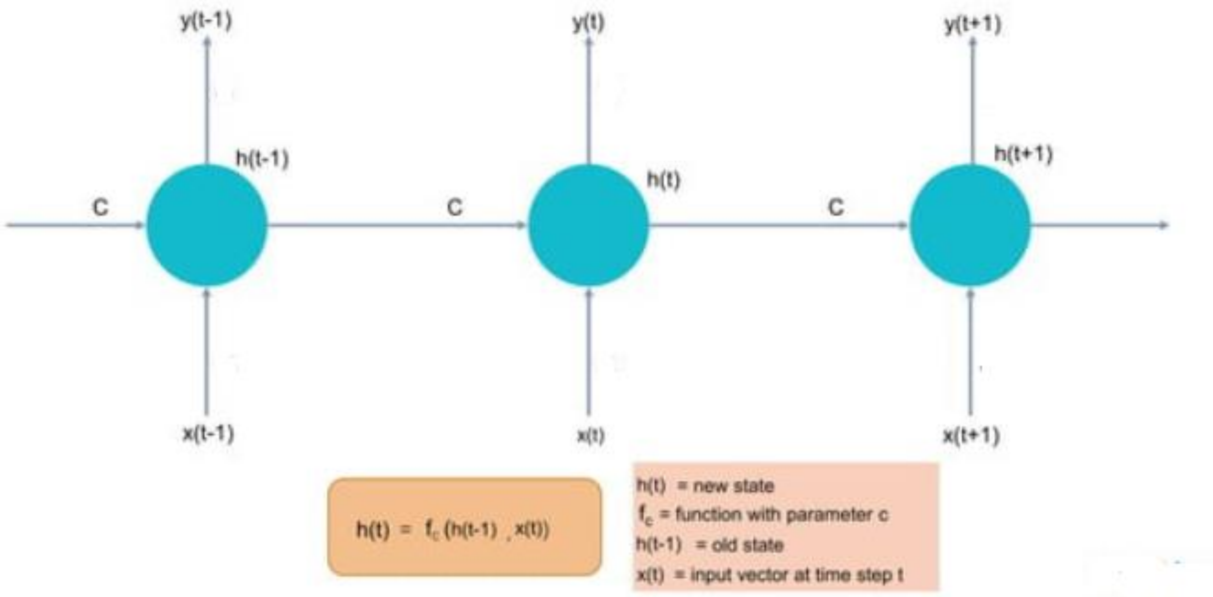
NCAI
NATIONAL CENTRE OF
ARTIFICIAL INTELLIGENCE

Recurrent Neural Network (RNN)

Making the world a better place through Artificial Intelligence!

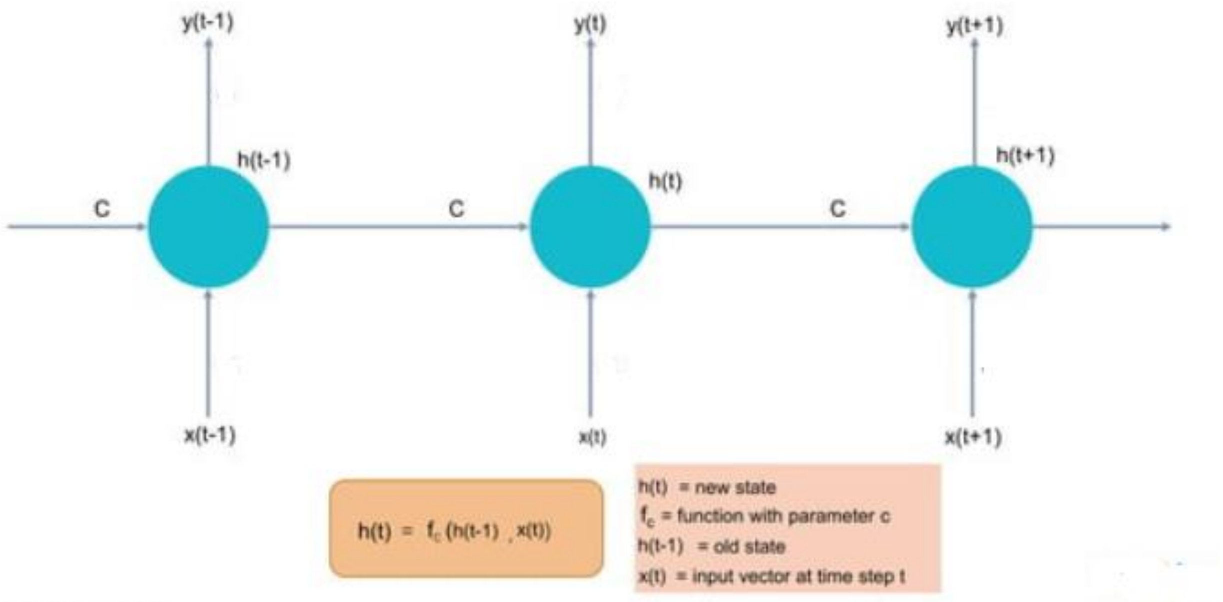
Recurrent Neural Network (RNN)

- RNN is a modification to neural network.
- It uses two inputs (current input or word, and the output of previous input or word) to compute output. C is the weight or parameter matrix similar to weights in neural networks.
- Output is calculated using neural network (but it uses two inputs instead of one).



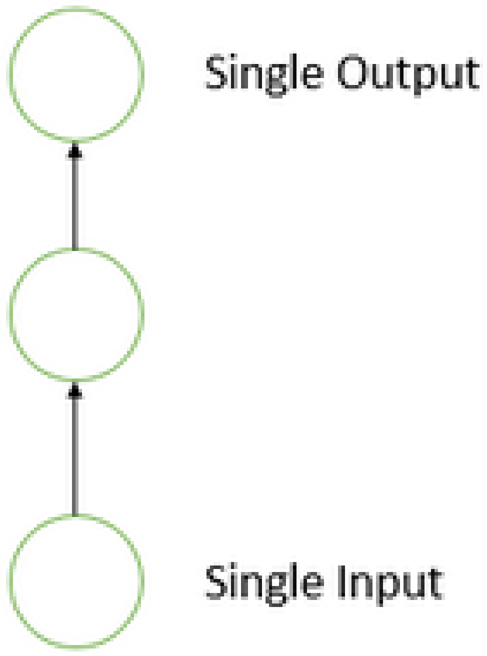
Recurrent Neural Network (RNN)

- d) Q: What will be the 2nd input at start ? We don’t have any previous output at start?
- e) Answer: A matrix with random values.



Types of Recurrent Neural Network (RNN)

1. One-to-One RNN:
   a) It takes one input and produces one output. E.g. image classification.
   b) It is like a traditional Neural Network.

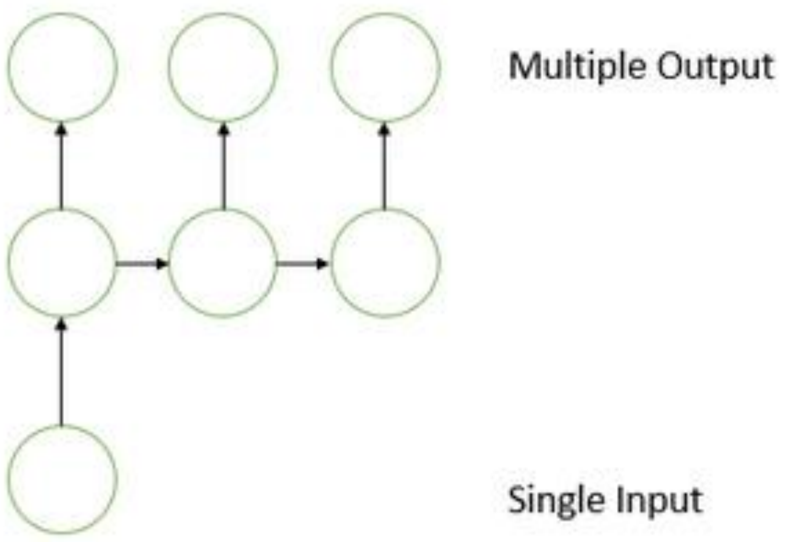


One-to-One RNN

Types of Recurrent Neural Network (RNN)

3. One-to-Many RNN:

a. It takes single input and produces multiple outputs. E.g. Image captioning where an image is a single input, and a sentence of words (many words) is output.

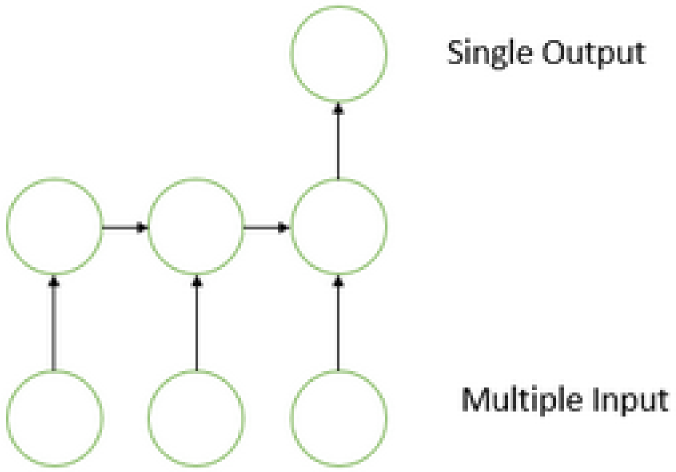


One-to-Many RNN

Types of Recurrent Neural Network (RNN)

2. Many-to-One RNN:

a. It takes multiple inputs and produces single output. E.g. Sentiment analysis
b. where input is a sentence (multiple words), and single output (+ve, -ve, or neutral).

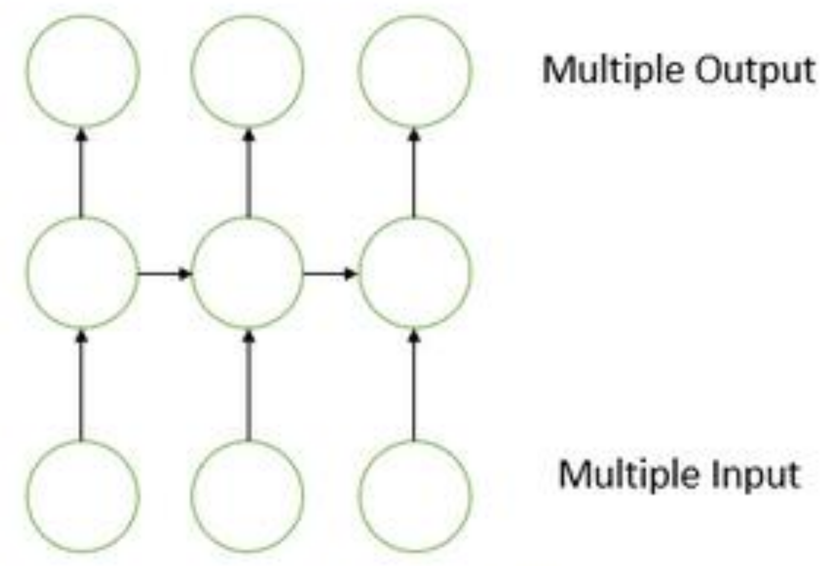


Many-to-One RNN

Types of Recurrent Neural Network (RNN)

2. Many-to-Many RNN:

a. It takes multiple inputs and produces multiple outputs. E.g. Machine Translation, in which the RNN scans any English text (many words) and then converts it to French (many words)



Many-to-Many RNN

Overall Training loop of Recurrent Neural Network (RNN)

- Input (consists of sequences such as words) is passed to RNN. RNN computes hidden state of current input by taking current input and hidden state of previous input. At the last layer final output is calculated, (such as +ve, -ve, or neutral sentiment of a sentence).
- Output is compared with the actual label. The aim is to predict output which is similar to the actual label. If predicted output is much different from the actual label , then loss is much higher, otherwise it is lower.
- The aim of training is to minimize loss. Loss is minimized by adjusting weights or parameters of RNN using Gradient descent of stochastic gradient descent.
- Gradients (gradient means a vector of derivatives), are back propagated (from the last layer to the first layer) using chain rule.

Overall Training loop of Recurrent Neural Network (RNN)

- Problem?
  - Gradients (gradient means a vector of derivatives), are back propagated (from the last layer to the first layer) using chain rule. Chain rule is a multiplication process of gradients.
    - If the gradients are too small, then multiplication of gradient in one layer with other gradient in other layer, makes them more smaller. This is called vanishing gradient problem.
- Due to vanishing gradient, impact of gradient is smaller. Hence, no useful training happen, because gradients are the key factor in updating weights.
- Exampe:
  - 1) input= 10, if we multiply input by 0.1, then 10 X 0.1= 1
  - 2) Input= 10, if 10 X 0.01= 0.1.
- Above example shows that multiplying input with smaller value makes it smaller.

Overall Training loop of Recurrent Neural Network (RNN)

Problem?

- c) Due to vanishing gradient, impact of gradient is smaller. Hence, no useful training happen, because gradients are the key factor in updating weights.

Exampe:

- 1) input= 10, if we multiply input by 0.1, then 10 X 0.1= 1
- 2) Input= 10, if 10 X 0.01= 0.1.

Above example shows that multiplying input with smaller value makes it smaller.

- d) Also, the impact of previous inputs is reduced if RNN contains too many layers.

- e) Other problem is exploding gradient. This means gradient values are too high.

Overall Training loop of Recurrent Neural Network (RNN)

- Problem?

  e) Other problem is exploding gradient. This means gradient values are too high.

  f) Exploding gradient can be detected by monitoring training loss. Training loss will be Nan.

  g) The solution of exploding gradient is gradient clipping. This means we manually reduce gradient value if it exceeds a certain threshold?

  h) For vanishing gradient, manually increasing gradient values does not work.

  i) Vanishing gradient is also detected by monitoring loss. Loss will not decrease due to it.

  j) Different solutions have been proposed for vanishing gradients but in RNN, generally GRUs or LSTMs are used.

Overall Training loop of Recurrent Neural Network (RNN)

- Example of long sequences:

  a) Bob is nice person. Dan, is evil.

  In above example, 2nd sentence doesn't have depend on the 1st sentence. Hence, we can forget about 1st sentence while working on 2nd sentence.

  b) Bob knows swimming. He told me over the phone that he had served the navy for four long years.

  In above example, both sentences are dependent on each other, so vanishing gradient will not allow to learn this dependency.

ICRL
Intelligent Criminology Research Lab

NCAI NATIONAL CENTRE OF ARTIFICIAL INTELLIGENCE

Gated Recurrent Unit (GRU)

Making the world a better place through Artificial Intelligence!

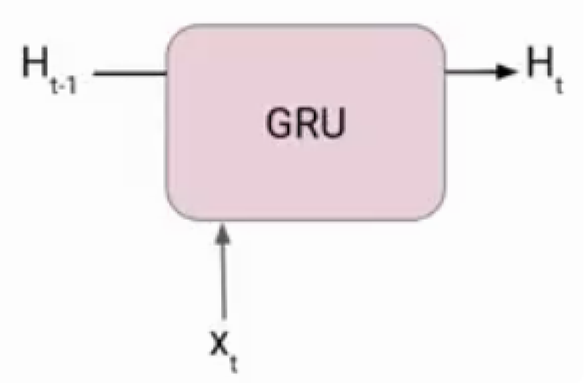
Gated Recurrent Unit (GRU)

- It is a variant of RNN, it gives memory to RNN. This memory is not a RAM or physical memory. It simply means it allows the model to retain information from previous layers.
- Introduced in 2014.
- It uses two gates to control flow of information from one layer to another. Hence, it can handle the required information from previous layers (this required information was reduced previously due to vanishing gradient problem).
- GRU has two gates:
- Reset Gate
- Update Gate

Gated Recurrent Unit (GRU)

- Similar to RNN, it takes an input Xt and the hidden state Ht-1 from the previous timestamp t-1
- Additionally, it contains two gates.
- 1) Reset Gate (Short term memory):
- It is responsible for the short-term memory of the network i.e the hidden state (Ht). Here is the equation of the Reset gate.

$$r_t = \sigma(x_t * U_r + H_{t-1} * W_r)$$



- The value of rt will range from 0 to 1 because of the sigmoid function.
- Here Ur and Wr are weight matrices for the reset gate.

Gated Recurrent Unit (GRU)

- 2) Update Gate (Long Term Memory)
  - Here is the equation of the Update gate. The only difference with reset gate is of weight metrics i.e Uu and Wu.

$$u_t = \sigma(x_t * U_u + H_{t-1} * W_u)$$

- How GRU Works?
  - It works in two steps:
    - a) First, it computes Candidate or Possible Hidden State
    - b) Then it computes actual current hidden state using candidate hidden state.

Gated Recurrent Unit (GRU)

- How GRU Works?
  - a) Candidate Hidden State:

$$\hat{H}_t = \tanh(x_t * U_g + (r_t \circ H_{t-1}) * W_g)$$

- Hidden state from the previous timestamp Ht-1 is multiplied (element-wise) by the reset gate output rt (please visit equation in previous slides).
- Wg and Ug are weight matrix, learnt automatically during training.
- After multiplication and summation, all the results are passed to tanh function.
- Tanh converts the input to anywhere between -1 and +1, using some formula.

ICRL
Intelligent Criminology Research Lab

Gated Recurrent Unit (GRU)

- How GRU Works?
  - a) Candidate Hidden State:

$$\hat{H}_t = \tanh(x_t * U_g + (r_t \circ H_{t-1}) * W_g)$$

- Multiplication with reset gate dictates how much information from previous hidden state should be considered.
- If rt is 0, then nothing is remembered from previous state because information from previous state become 0 after multiplication.
- If rt is 1, then everything from previous state is remembered.
- Usually, the value of rt will be somewhere between 0 and 1.

Gated Recurrent Unit (GRU)

- How GRU Works?
- Candidate Hidden State
- Actual current hidden state using candidate hidden state:

$$H_t = u_t \circ H_{t-1} + (1-u_t) \circ \hat{H}_t$$

- Ut is the update gate (described in previous slides)
- If Ut is 1, current hidden state will entirely depend on previous hidden state because 2nd term will become 0.
- If Ut is 0, then the first part of the equation becomes 0, and only 2nd part of equation remains relevant. Only a little information from the previous hidden state is used. That little information is present in the candidate hidden state.
- Generally, Ut will be between 0 and 1, hence the impact of both terms will be present.

Gated Recurrent Unit (GRU)

- Q: How reset and update gate values are decided?
- A:
  - These are calculated using the equations given in the previous slides. As during training, weight matrix and hidden states Ht are adjusted and learned, the values of these gates also change and learned during training.
  - Hopefully, at the end of training we will have ideal values of these gates. Otherwise, loss will be high, and model will not predict correct outputs.

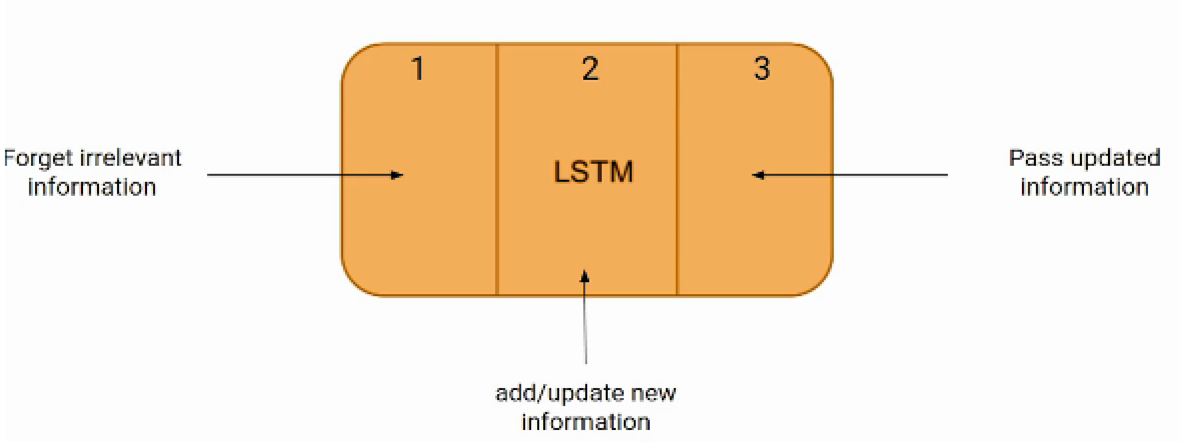
ICRL
Intelligent Criminology Research Lab

Long Short-Term Memory (LSTM)

Making the world a better place through Artificial Intelligence!

Long Short-Term Memory (LSTM)

- It consists of three gates (units). Gate means they control the flow of information.
- These gates dictate:
- What to forget and remember from previous state.
- How to get new information.
- Pass the overall information to next LSTM cell.



Long Short-Term Memory (LSTM)

- Forget Gate: Decides what to remember and forget from previous input?

$$f_t = \sigma (x_t * U_f + H_{t-1} * W_f)$$

- Xt: input to the current timestamp.
- Uf: weight associated with the input
- Ht-1: The hidden state of the previous timestamp
- Wf: It is the weight matrix associated with the hidden state

- Similar to GRU, it uses sigmoid function, hence output is between 0 and 1.
- We will use computed ft in future equations.

Long Short-Term Memory (LSTM)

- Input Gate: Decides, the hidden state calculation from current input.

$$i_t = \sigma(x_t * U_i + H_{t-1} * W_i)$$

- Xt: input to the current timestamp.
- Ui: weight associated with the input
- Ht-1: The hidden state of the previous timestamp
- Wi: It is the weight matrix associated with the hidden state

- It also uses sigmoid function, hence output is between 0 and 1.

Long Short-Term Memory (LSTM)

- New Information:

$$N_t = \tanh(x_t * U_c + H_{t-1} * W_c) \text{ (new information)}$$

- Xt: input to the current timestamp.
- Uc: weight associated with the input
- Ht-1: The hidden state of the previous timestamp
- Wc: It is the weight matrix associated with the hidden state
- It uses tanh function, hence output is between -1 and 1. This new information Nt is not directly used, instead it uses another equation to compute cell state:

$$C_t = f_t * C_{t-1} + i_t * N_t \text{ (updating cell state)}$$

- To compute cell state Ct, forget get and input gates are used:

ICRL
Intelligent Criminology Research Lab

Long Short-Term Memory (LSTM)

- Output gate:

$$o_t = \sigma(x_t * U_o + H_{t-1} * W_o)$$

- Xt: input to the current timestamp.
- Uo: weight associated with the input
- Ht-1: The hidden state of the previous timestamp
- Wo: It is the weight matrix associated with the hidden state
- Hidden state of current input is calculated as:

$$H_t = o_t * \tanh(C_t)$$

- Hidden state of current input uses the cell state Ct, which internally uses input gate, forget gate, and updating cell state. Moreover it also uses output gate Ot.

Long Short-Term Memory (LSTM)

- Hidden state of current input uses the cell state Ct, which internally uses input gate, forget gate, and updating cell state. These gates retain useful information from previous hidden states.
- Moreover it also uses output gate Ot.

$$H_t = o_t * \tanh(C_t)$$

- If, we need to calculate final output, Ht, can be passed to softmax, which converts it to probabilities. Then, loss can be computed.

LSTM VS RNN

Aspect	LSTM	RNN
Architecture	A type of RNN with additional memory cells.	RNN itself.
Memory Retention	Handles long-term dependencies in sequences.	Struggles with long-term dependencies.
Cell Structure	Complex cell structure with input, output, and forget gates	Simple cell structure with only hidden state
Training Efficiency	Slower training process due to increased complexity of gates.	Faster training process due to simpler architecture.
Performance on Long Sequences	Performs better on long sequences	Struggles to retain information on long sequences
Vanishing Gradient Problem	Addresses the vanishing gradient problem	Prone to the vanishing gradient problem

ICRL
Intelligent Criminology Research Lab

LSTM VS GRU

Aspect	LSTM	GRU
Architecture	A type of RNN with additional memory cells.	It is also type of RNN.
Memory Retention	Handles long-term dependencies in sequences.	It also handles long-term dependencies in sequences.
Cell Structure	Complex cell structure with input, output, and forget gates	Simple cell structure with only two gates.
Training Efficiency	Slower training process due to increased complexity of gates.	Faster training process due to compared to LSTM due to less gates.
Performance on Long Sequences	Performs better on long sequences	It also performs better on long sequences
Introduced in:	1998	2014
So use LSTM or GRU?	• Not clear guidelines. • Use LSTM if performance is more important. • Use GRU if training time is more important. • Both can be tried.	

ICRL
Intelligent Criminology Research Lab

Final Words

- Both LSTM and GRU learn sentence level representation, not individual word embedding (Though word embedding can also be updated).
- Generally, LSTMs and GRUs are applied after embedding of words. These embedding can be retrieved from Glove, Bag of words, word2vec etc.
- LSTM and GRU can also be applied on the final hidden layer of recently introduced transformer architectures.
- However, transformer architectures already solve vanishing gradient problems by using self-attention mechanism. Therefore, additional usage of LSTM and GRU may not improve results.