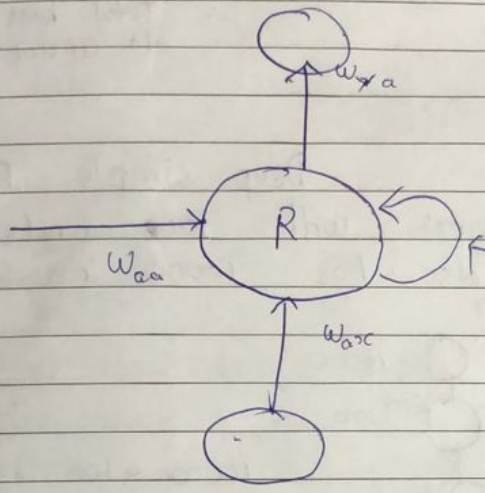
Q A simple RNN has 10000 neurons in i/p layer. There is a single recurrent layer with 100 units. The output layer also has 10000 neurons. Numbers of parameters involved is _____. Ignore bias param.

Soln



10000 × 100 (Input) + 100 × 10000 (output) + 100 × 100 (without bias)

= 2010000

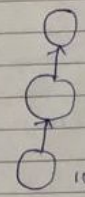
Q2 Simple RNN has 10000 Neurons in i/p layer. Single layer has 1000 units. Output layer has 1 neuron. Number of parameters.

Soln

Input: 10000 × 1000 + Self loop: 1000 × 1000 + Output: 1000 × 1

= 11001000

For bias

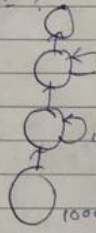


Bias for
1) Input to Hidden = 1000
2) Hidden to Output = 1

∴ Total bias to be added to answer is 1001.

Q I/p = 10000 Deep simple RNN
2 recurrent layers with 100 units each.
Output layer also has 10000 neuron.
Parameters = ?

Soln



10000 × 100
+ 100 × 100 + 100 × 100 + 100 × 10000
Self Self

= 100(20300)

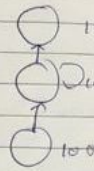
= 2030000

With bias

2030000 + 100 + 100 + 10000
= 2040200.

Q Bidirectional RNN
i/p = 10000 neuron
output = 10000 "
Recurrent layer = 100 units.

Soln



= 2(10000 × 100
+ 100 × 100
+ 100 × 10000)

= 2 × 2020000

= 4020000

Q LSTM

i/p = 10000 ~~units~~ neurons
Hidden layer = 100 units
Output = 10000 neurons.

Soln

~~1000~~ 10000 × 100

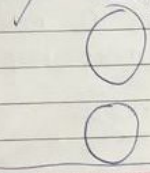
5080000

$g(h(x_t) + h$

4(100 × 100 + 10000) + 100)

= 4(10010000 + 100)

= 10010100 = 4060400



because for output there is only one dense layer

Sharing 4 weights LSTM
4 × 10000 × 100 (Ip)
+ 4 × 100 × 100 (hidden)
+ 100 × 10000 (output)
→ No sharing of weights in output

= 5040000

With bias

5040000 + 4×100 + 10000

Q GRU
i/p = 10000
hidden = 100 units
o/p = 10000

Sol^n 3 × 10000 × 100
+ 3×100×100
+ 10000×100 = 4030000

UNET
LeNet
Backpropagation
Skip.
Steep fun^n in UNET is inevitable.

⇒ LSTM

1) Forget Gate

→ Information from the previous hidden state & info from the current input is passed through the sigmoid fun^n
→ Value closer to 0 means to forget & closer to 1 means keep.

2) Input Gate

S → Imp
T → Regulate

→ Info from previous hidden state & info from current input is passed through sigmoid.
→ 0 means not imp & 1 means imp.
→ Also, we pass the hidden state & current input into tanh fun^n to help regulate the network
→ Then multiply tanh output with sigmoid output.
→ The sigmoid output will decide which info is imp to keep from tanh output.

3) Cell State

→ Cell state is multiplied by forget vector
→ Then we take output from input gate & do pointwise add^n which updates cell state to new values that NN finds relevant.

4) Output Gate

→ The output gate decides what next hidden state should be
→ First we pass previous hidden state & current input to sigmoid fun^n.
→ Then we pass newly modified cell state to tanh fun^n.
→ Then we multiply both of them to decide what information hidden state should keep.
→ New cell state & a new hidden state is carried over to next timestamp.