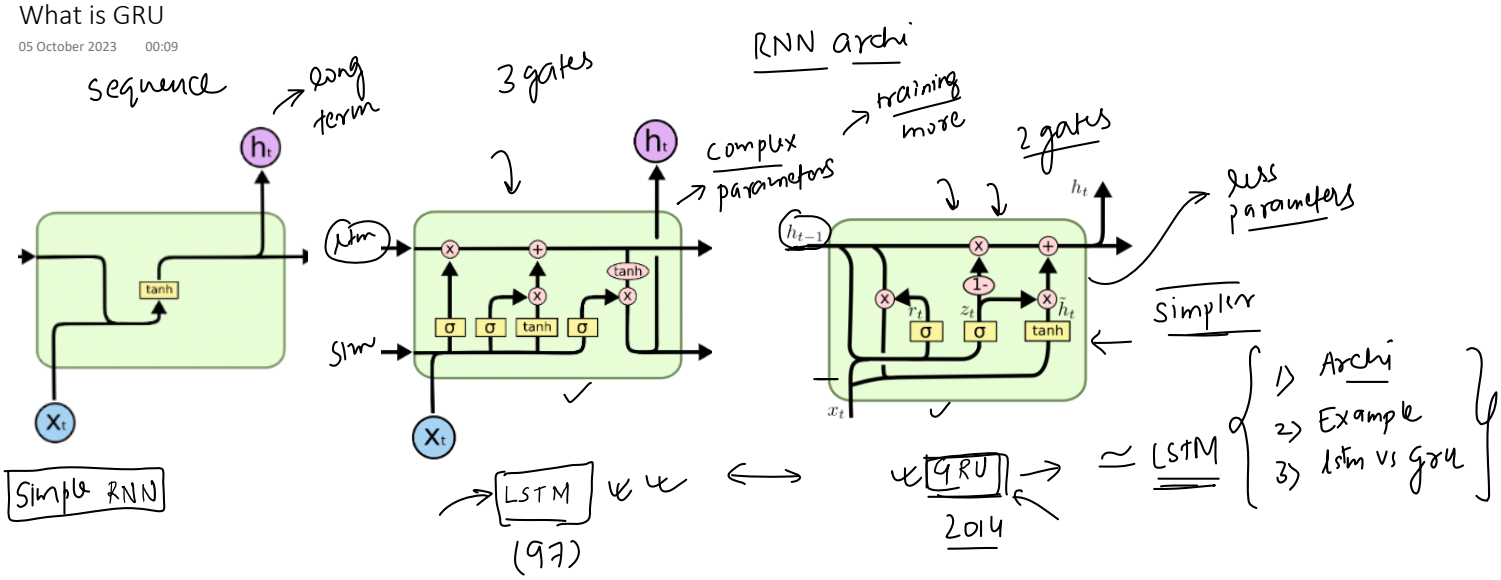
Lecture-64 (GRU)

What is GRU

05 October 2023 00:09



→ LSTM এসেছে 1997 আর GRU 2014 সালে।

→ LSTM এর Architecture complex parameter অনেক বেশি। এজন্য LSTM এর থেকে simple Architecture হচ্ছে GRU।

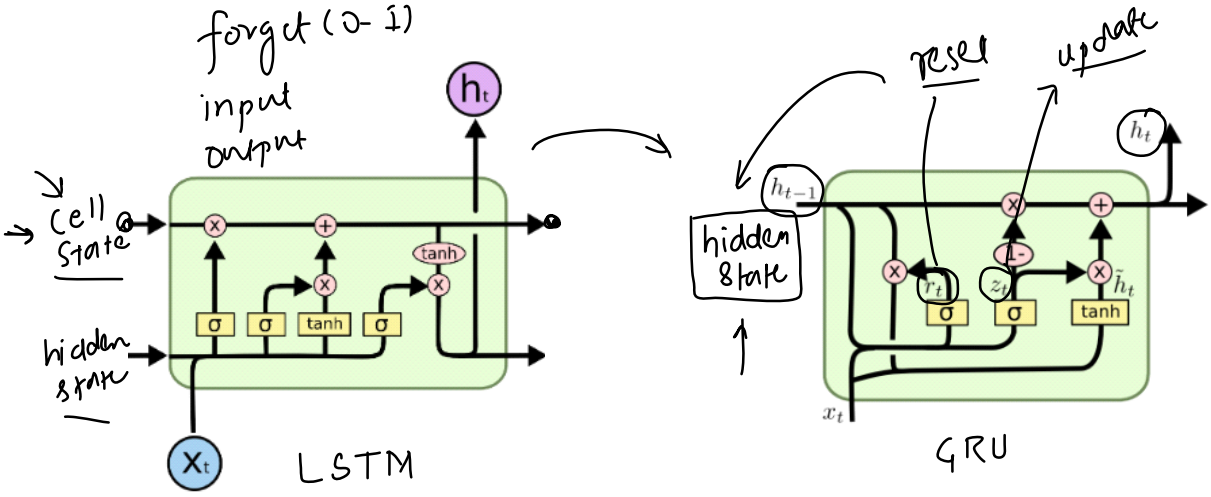
→ LSTM এ gate 3 টি অন্যদিকে GRU এর gate 2টি (Reset + Update)।

→ LSTM এ দুই ধরনের context maintain করা হয় (LTM, STM) অন্যদিকে GRU তে একটি context ই follow করা হয়।

→ এমন হয় যে GRU এর performance অনেকের ভালো হবে। LSTM এর টাও হতে পারে আমাদের check করে দেখতে হবে কোনটা ভালো কাজ করে। অনেক অনেক বড় and complex data set এ LSTM train হতে অনেক বেশি সময় লাগে।

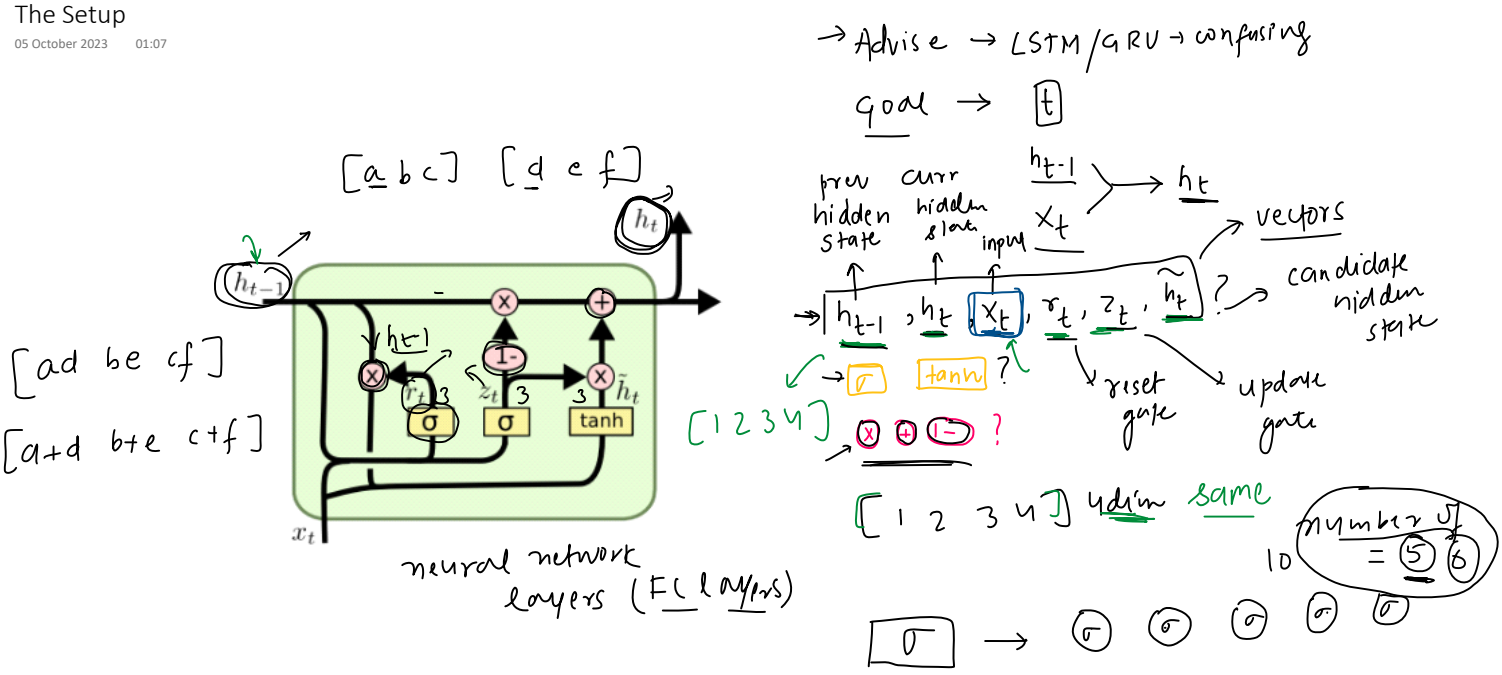
The Big Idea Behind GRU

05 October 2023 00:47



The Setup

05 October 2023 01:07



LSTM পড়ার সময় আমরা দেখেছি, ছোট বক্সগুলো neural network represent করে। আর এই neural network এ কতগুলো node বা neuron থাকবে তা আমরা বলে দেই। আর সবগুলো neural network এই সমান সংখ্যক node থাকবে। আর যদি node 4 টা থাকে তাহলে, h_{t-1}, h_t, r_t, z_t, $\tilde{h}_t$ তৈরা হবে vector এবং সেই vector এর dimention হবে 4।

h_{t-1} → Previous hidden state

h_t → Current hidden state

r_t → reset gate

z_t → Update gate

$\tilde{h}_t$ → Candidate hidden state

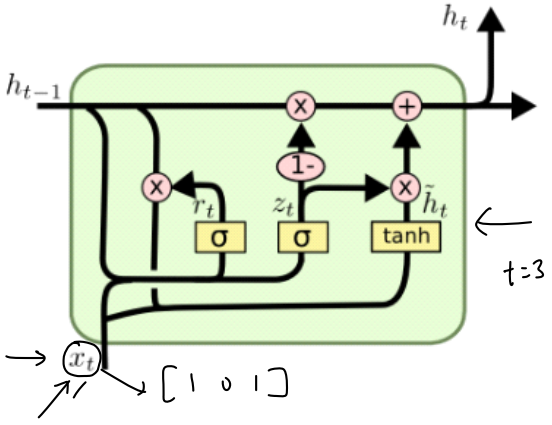
x_t → (Input in current time step) [কত dimention হিসু হয়]

⊕ Pointwise addition

⊗ pointwise multiplication

The Input Xt

05 October 2023 01:52



t input → vector

Sentiment

text	review
cat mat rat	1
cat rat rat	0
→ mat rat mat	1

OHE / BOW / Word2Vec

S1 [1 0 0] [0 1 0] [0 0 1]

S2 [1 0 0] [0 0 1] [0 0 1]

S3 [0 1 0] [0 0 1] [0 1 0]

প্রত্যেক time step এ আলাদা আলাদা word input যাবে। যেমন, sentence এর মতো,

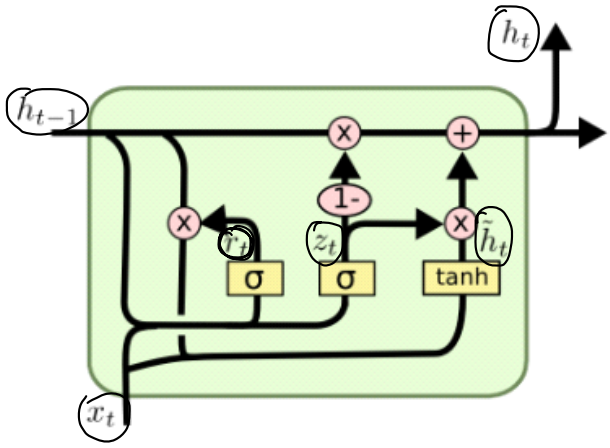
t=1, x_1= [1 0 0]

t=2, x_2= [0 1 0]

t=3, x_3= [0 0 1]

Architecture

05 October 2023 02:10



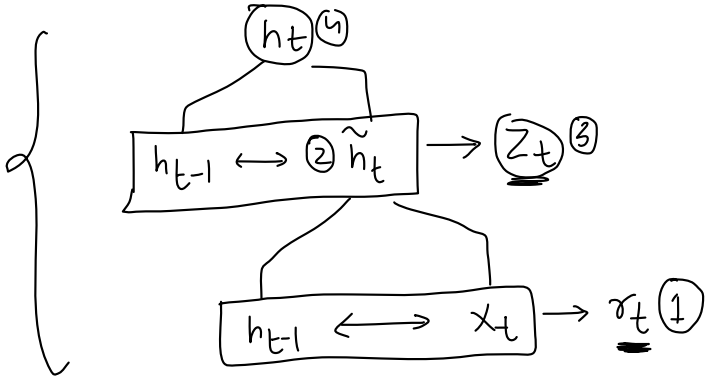
h_{t-1} , x_t ⟶ h_t

→ 1) Calculate r_t (reset gate)

→ 2) Calculate $\tilde{h}_t$ (candidate hidden state)

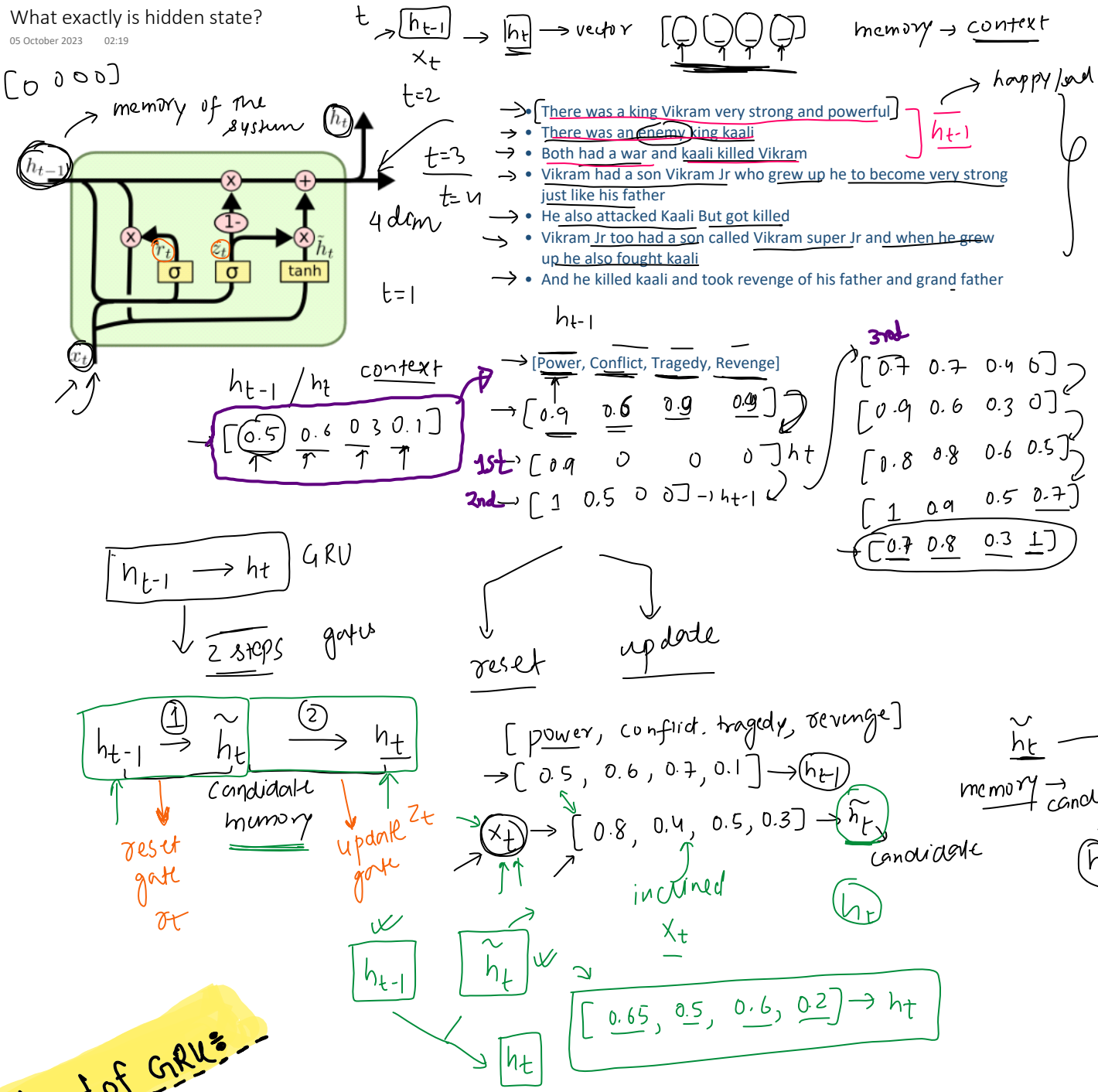
→ 3) Calculate z_t (update gate)

→ 4) Calculate h_t (current hidden state)



What exactly is hidden state?

05 October 2023 02:19



h_{t-1}

Mathematically, hidden state হচ্ছে vector. But vector এর number গুলোর significant কি? h_{t-1} সেই system এর memory হিসেবে আচরণ করে। উদাহরণ সরূপ যদি 6,7 নং লাইন GRU input হিসেবে দেয় তাহলে আমাদের জানা দরকার কোন Kaai কে মারা হচ্ছে আর এর উত্তর উদাহরণের লাইনগুলোতে আছে, সেইটা memory হিসেবে বা context হিসেবে store করে h_{t-1}।

ধরি, $h_{t-1} = [0.3 \quad 0.2 \quad 0.5 \quad 0.7]$ = [power, conflict, strength, revenge] হতে পারে (যেকোনো কিছু)

$[0.9 \quad 0 \quad 0 \quad 0]$ → h_{t-1}, এখন mean করছে যে বাচ্চা মানুষকে বলা হয়েছে, তিনি অনেক powerful।

$[0.1 \quad 0.2 \quad 0 \quad 0]$

এখন, যদি Revenge এর value বাড়ে বুঝবো কোন প্রতিশোধ নেওয়া হয়েছে।

এখানে, $0.9 \to 0.1$, $0 \to 0.2$ যে হচ্ছে বা update হচ্ছে reset & update gate এর সাহায্যে।

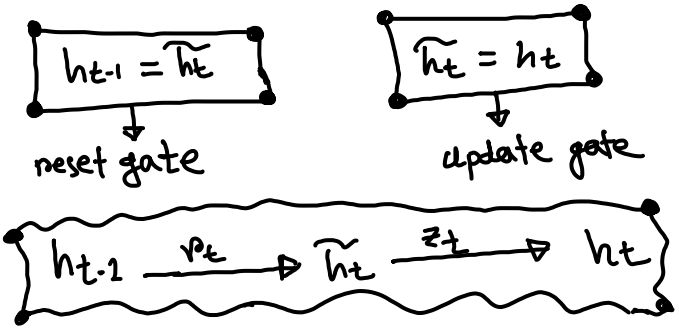
এখন, h_{t-1} থেকে h_t কিভাবে calculation হয়?

(i) $h_{t-1} = \tilde{h}_t$ (ii) $\tilde{h}_t = h_t$

h_{t-1} (Past memory)

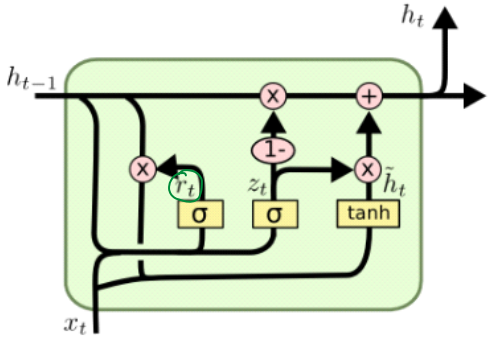
x_t (current state input)

h_{t-1} ও x_t এর basis এ $\tilde{h}_t$ (candidate hidden state তৈরি হয়)। $\tilde{h}_t$ current input এর basis এ তৈরি হয়েছে তাকে আমরা সরাসরি h_t তে convert করতে পারি না। কারণ, কোন কিছু current input এর basis এ important হলেও overall important নাও হতে পারে। তাই, আমরা আবার, h_{t-1} ও $\tilde{h}_t$ কে balance করি।



Calculating the reset gate

05 October 2023 14:24



reset

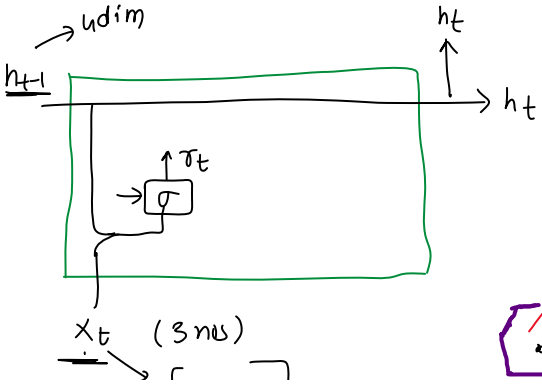
r_t → vector → h_{t-1}

↳ gate

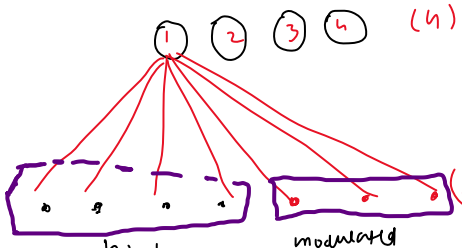
$h_{t-1} = [0.6, 0.6, 0.7, 0.1]$ (lower lower higher)

$r_t = [0.8\ 0.2\ 0.1\ 0.9]$

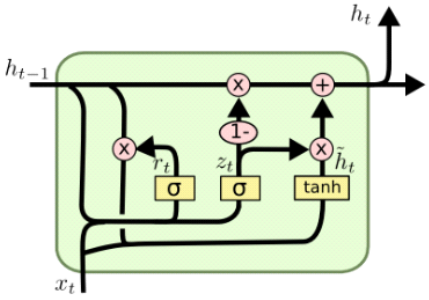
gate — 80% percent — 20% — 90%



$$r_t = \sigma(W_r[h_{t-1}, x_t] + b_r)$$



← (7x4) = 28 + 4



modulated reseted

h_{t-1} ⊗ r_t → $\tilde{h}_t$

(1x7) (7x4)

(1x4) + (1x4) → ((1x4))

(1x4) → r_t

h_t

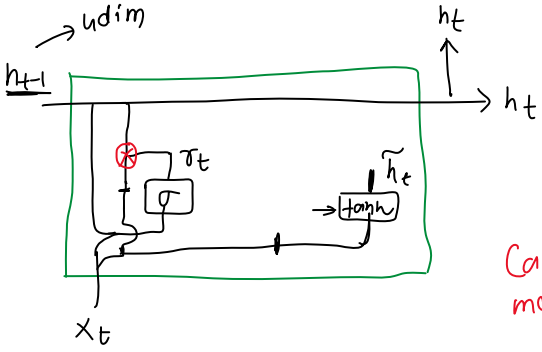
$h_{t-1} = [0.6, 0.6, 0.7, 0.1]$

$r_t = [0.8\ 0.2\ 0.1\ 0.9]$

$\tilde{h}_t = [0.7\ 0.2\ 0.1\ 0.2]$ ↑ x_t

power conflict tragedy revenge

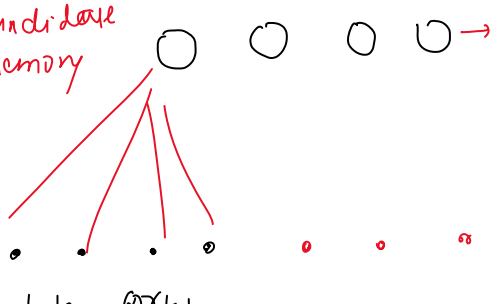
$h_{t-1} \otimes r_t = [0.48, 0.12, 0.07, 0.09]$ → x_t

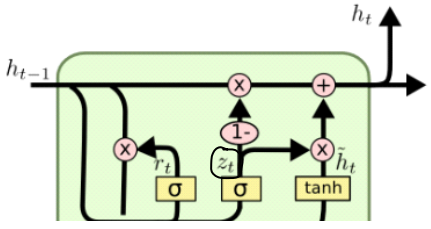


h_{t-1} / $\tilde{h}_t$ → z_t

$$\tilde{h}_t = \tanh(W_c[h_{t-1} \otimes r_t, x_t] + b_c)$$

Candidate memory





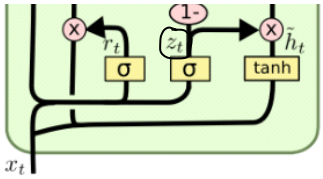
h_{t-1} — h_{t-1}, r_t → $r_t \otimes h_{t-1}$ → $\tilde{h}_t$

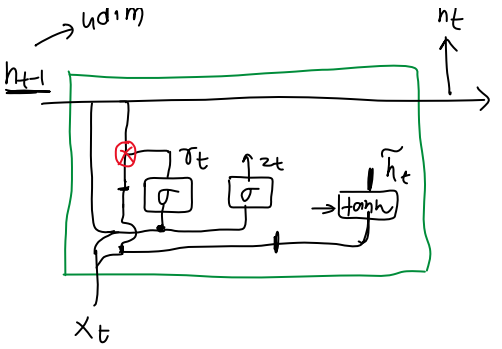
x_t — x_t

$\tilde{h}_t = [0.7\ 0.2\ 0.1\ 0.2]$

$h_{t-1} = [0.6\ 0.6\ 0.7\ 0.1]$

h_{t-1} → z_t





x_t

h_{t-1} x_t $\rightarrow$ z_t

$[[- - - -]]$ $(0-1)$

gate $z_t = \sigma(W_z[h_{t-1}, x_t] + b_z)$

h_{t-1} and $\tilde{h}_t$ (h_t)

$x_t \rightarrow$ important

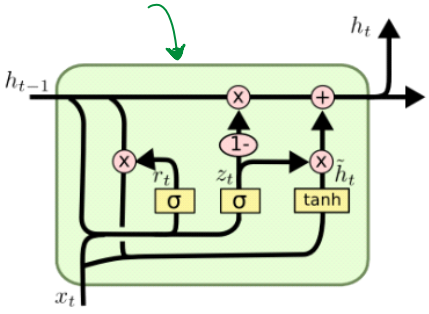
$x_t \rightarrow$ not imp

(1×4)

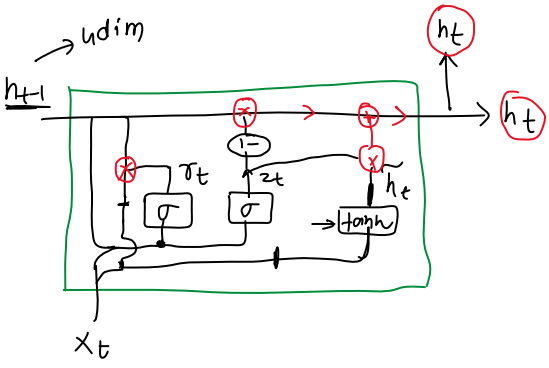
$\rightarrow$ bias(4)

$\rightarrow (7 \times 4)$

h_{t-1}



x_t



x_t

$\tilde{h}_t = [0.7 \ 0.2 \ 0.1 \ 0.2]$

$h_{t-1} = [0.6 \ 0.6 \ 0.7 \ 0.1]$

$z_t = [0.1 \ 0.7 \ 0.8 \ 0.2]$

h_t $[0.07 \ 0.14 \ 0.08 \ 0.04]$

$h_t = (1 - z_t) \otimes h_{t-1} \oplus z_t \otimes \tilde{h}_t$

0.1 0.1

0.1 z_t কম আমরা $(1-z_t)$ কে high value দিচ্ছি। যে কারণে h_t কে বেশি weight দিচ্ছি।

$(1-z_t) \rightarrow 0.9$

$(1-z_t)$ $[0.9 \ 0.3 \ 0.2 \ 0.8]$ $[0.1 \ 0.6 \ 0.7 \ 0.1]$

$[0.54 \ 0.18 \ 0.14 \ 0.08]$

h_t $[0.61 \ 0.32 \ 0.22 \ 0.12]$

power conflict tragedy revenge

h_{t-1} x_t $\rightarrow$ r_t $\rightarrow$ $r_t \otimes h_{t-1}$ $\rightarrow$ $\tilde{h}_t$

h_{t-1} x_t $\rightarrow$ z_t

h_{t-1} z_t $\tilde{h}_t$ $\rightarrow$ h_t

h_{t-1} & x_t থেকে r_t calculate করছি। current input এর কতটুকু পরিমাণে ঠিক করতে কোন value গুলো reset করতে হবে। আবার, h_{t-1} & x_t থেকে z_t update gate calculate করবো। যেটা ঠিক করবে কাকে বেশি value দিতে হবে। h_{t-1} or h_t।

LSTM vs GRU

05 October 2023 16:45

Here are the main differences between LSTM and GRU:

1. Number of Gates:
   - LSTM: Has three gates — input (or update) gate, forget gate, and output gate.
   - GRU: Has two gates — reset gate and update gate.

2. Memory Units:
   - LSTM: Uses two separate states - the cell state (ct) and the hidden state (ht). The cell state acts as an "internal memory" and is crucial for carrying long-term dependencies.
   - GRU: Simplifies this by using a single hidden state (ht) to both capture and output the memory.

3. Parameter Count:
   - LSTM: Generally has more parameters than a GRU because of its additional gate and separate cell state. For an input size of d and a hidden size of h, the LSTM has 4×((d×h)+(h×h)+h)4×((d×h)+(h×h)+h)parameters.
   - GRU: Has fewer parameters. For the same sizes, the GRU has 3×((d×h)+(h×h)+h)3×((d×h)+(h×h)+h)parameters.

4. Computational Complexity:
   - LSTM: Due to the extra gate and cell state, LSTMs are typically more computationally intensive than GRUs.
   - GRU: Is simpler and can be faster to compute, especially on smaller datasets or when computational resources are limited.

5. Empirical Performance:
   - LSTM: In many tasks, especially more complex ones, LSTMs have been observed to perform slightly better than GRUs.
   - GRU: Can perform comparably to LSTMs on certain tasks, especially when data is limited or tasks are simpler. They can also train faster due to fewer parameters.

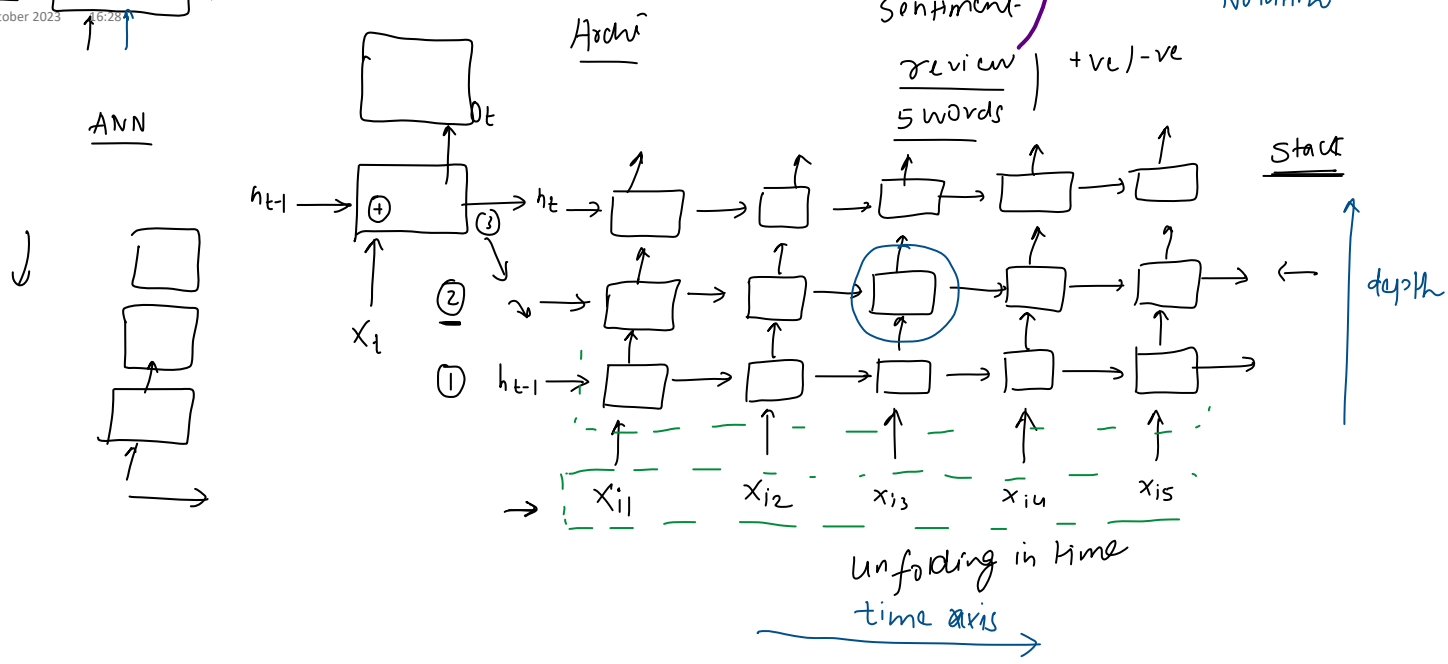
6. Choice in Practice:
   - The choice between LSTM and GRU often comes down to empirical testing. Depending on the dataset and task, one might outperform the other. However, GRUs, due to their simplicity, are often the first choice when starting out.

Deep RNN, Stacked RNN, Stacked LSTM, Stacked GRU

What is Deep RNN →

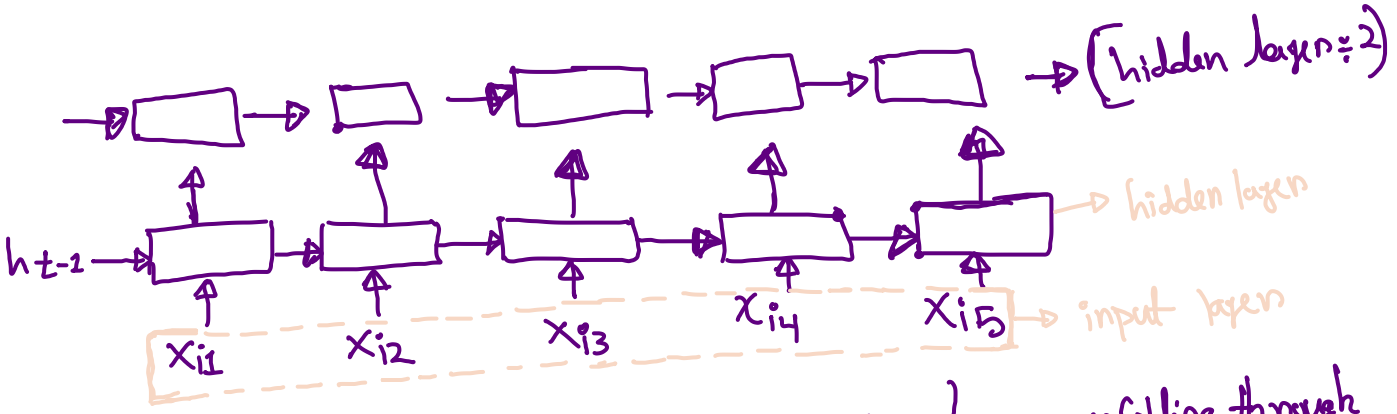
17 October 2023 16:28

ধরি sentement analysis করবো। review তে ৫টা word আছে। Notation



→ playground tensorflow (see the website)

ANN এ আমরা data এর complex pattern খোঁজার জন্য hidden layer ও Node add করি।
যদি RNN এ আমরা একই concept ব্যবহার করি তাহলে একে Deep RNN বলি।



x_i প্রথম review x_{i1} (প্রথম review এর প্রথম word)

↳ একে unfolding through time বলে।

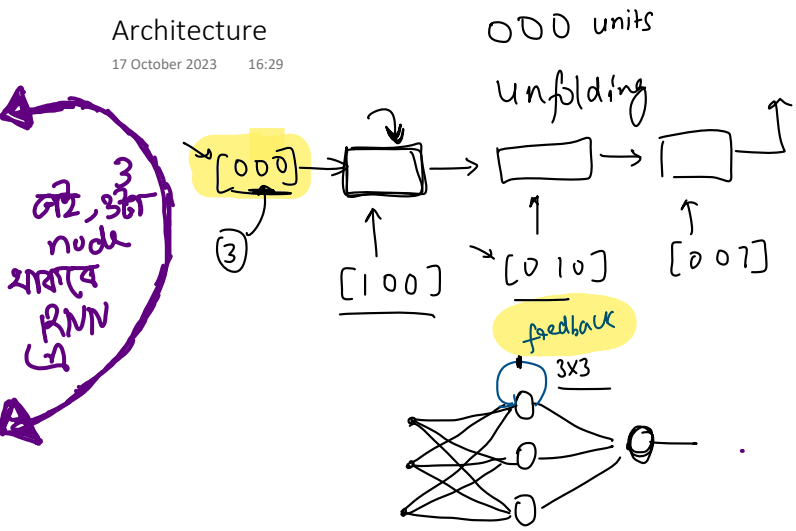
আমাদের এই RNN আমাদেরকে output দিচ্ছে না। এখন, আমরা hidden layer আরো add করতে পারবো accuracy improve করার জন্য। যদি, hidden layer add করি তাহলে সেই hidden layer এ input আগের layer থেকে যাবে।

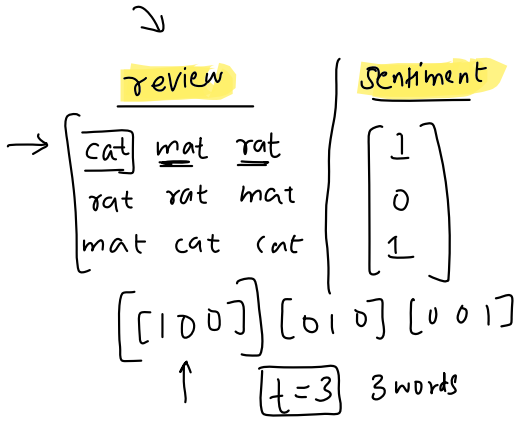
অর্থাৎ, Deep RNN হচ্ছে এটা RNN এর উপর আরো RNN stack করা এবং unfolding করা through time.

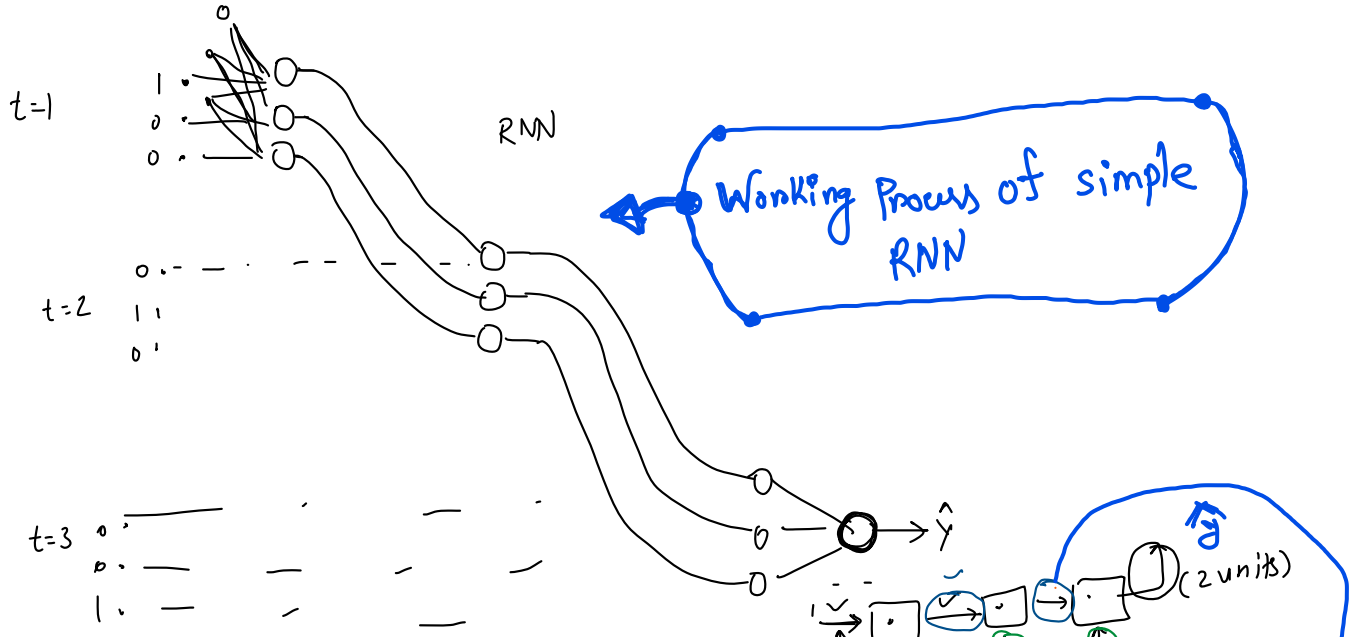
Architecture of deep RNN

Architecture

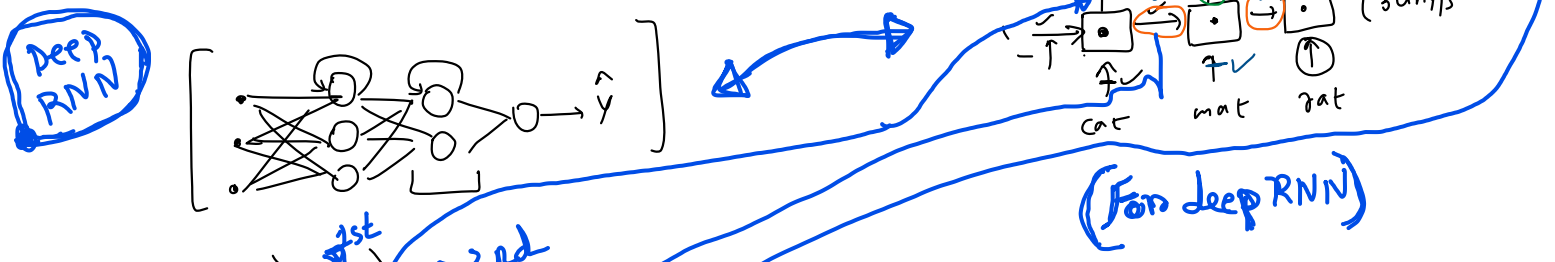
17 October 2023 16:29



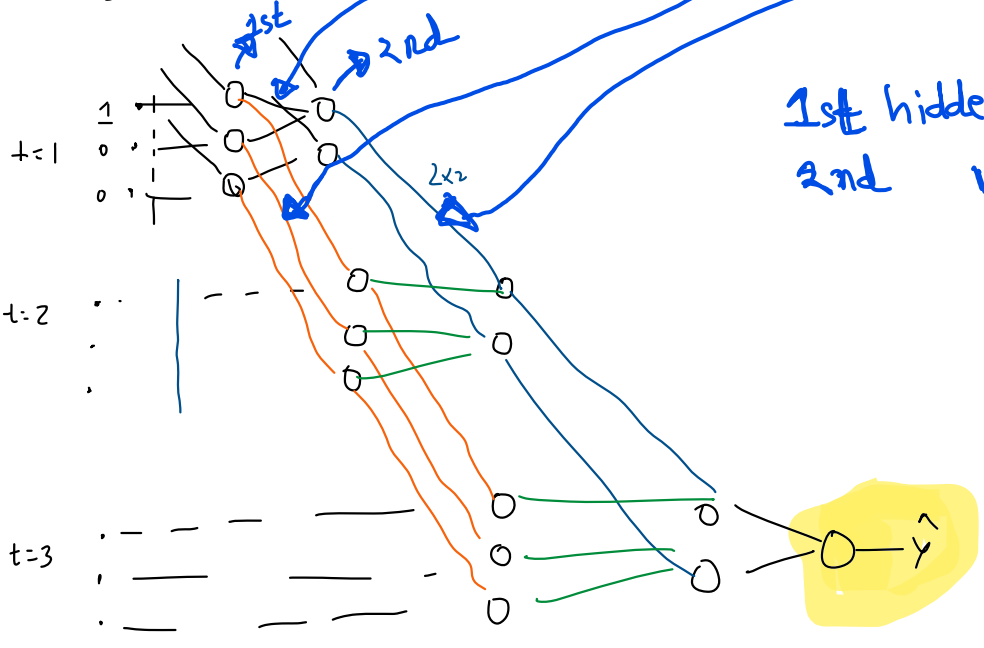




Working Process of simple RNN



(For deep RNN)



1st hidden layer RNN → 3 units

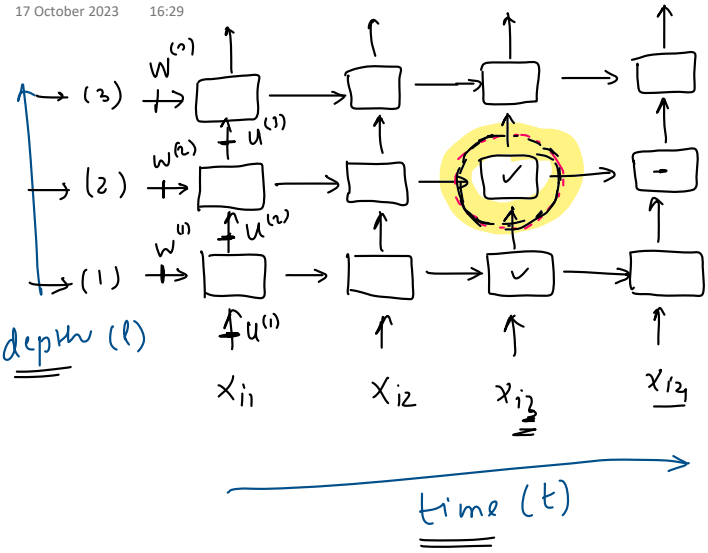
2nd " " " → 2 units

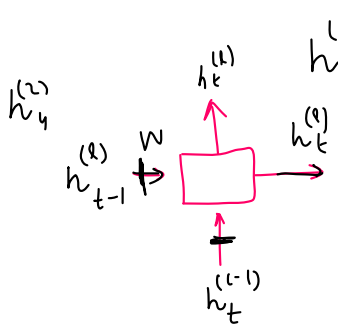

t=3

Take the middle node

Notation

17 October 2023 16:29





$$h_t^{(l)} = \tanh\left(W^{(l)} h_{t-1}^{(l)} + U^{(l)} h_t^{(l-1)} + b\right)$$

(এটা ভালো করে পড়বো যেন কোন documentation এ দেখতে পারি)

→ 2টা axis-এর x-axis এ time axis, y-axis এ depth axis.

→ Depth কে l আর time কে t দিয়ে।

Why and When to use?

17 October 2023 16:29

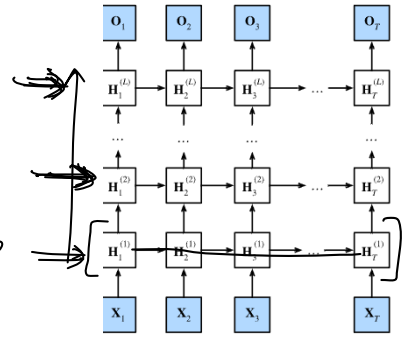
deep RNN

Stack

product

sentence

1. Hierarchical Representation ✓
2. Customization for Advanced Tasks



encoder-decoder

machine

deep RNNs

love hate amazing terrible

deep RNN

Audio is bad

Display is great

Overall I am happy

→ sentiment

When to use Deep RNNs?

Complex tasks

Speech recog

Machine translation

Large datasets

Overfitting

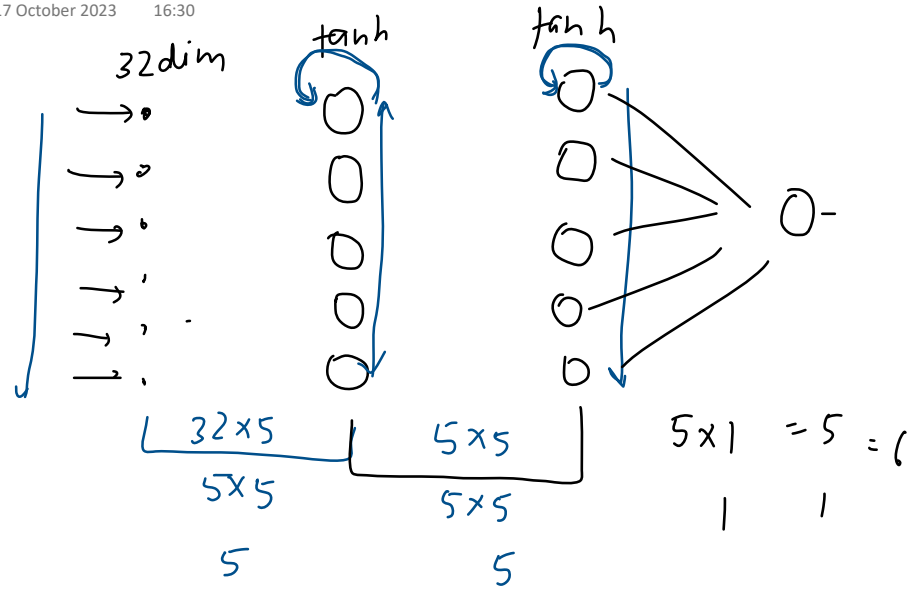
Computational

Simpler Models

Deep RNN

Code Example

17 October 2023 16:30



→ embedded

10000 x 32 = 3,20,000

simple RNN 1

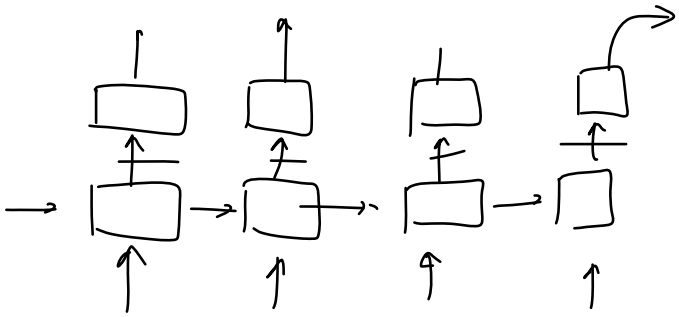
190

simple RNN 2

55

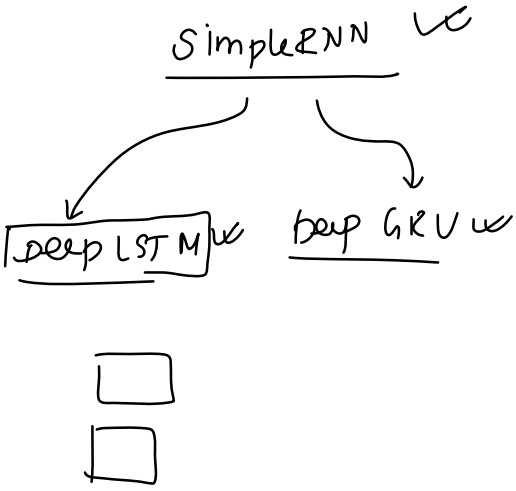
Dense

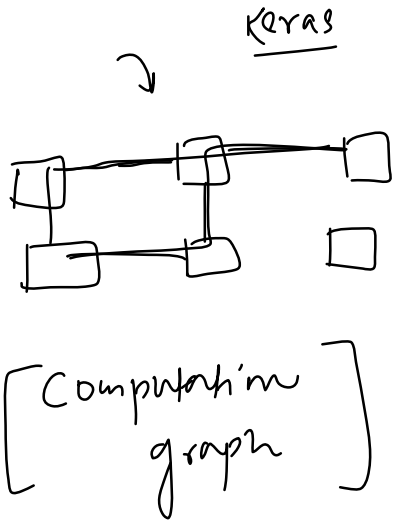
6



Variants

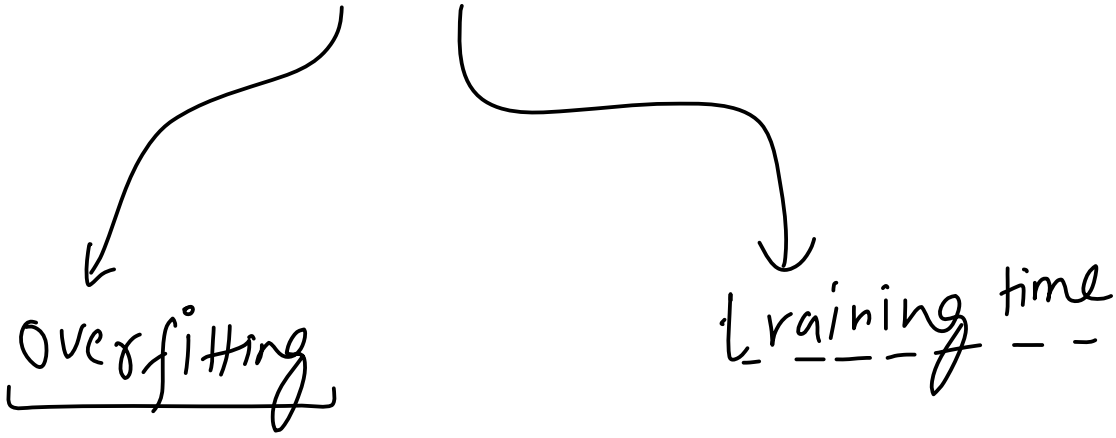
17 October 2023 16:30





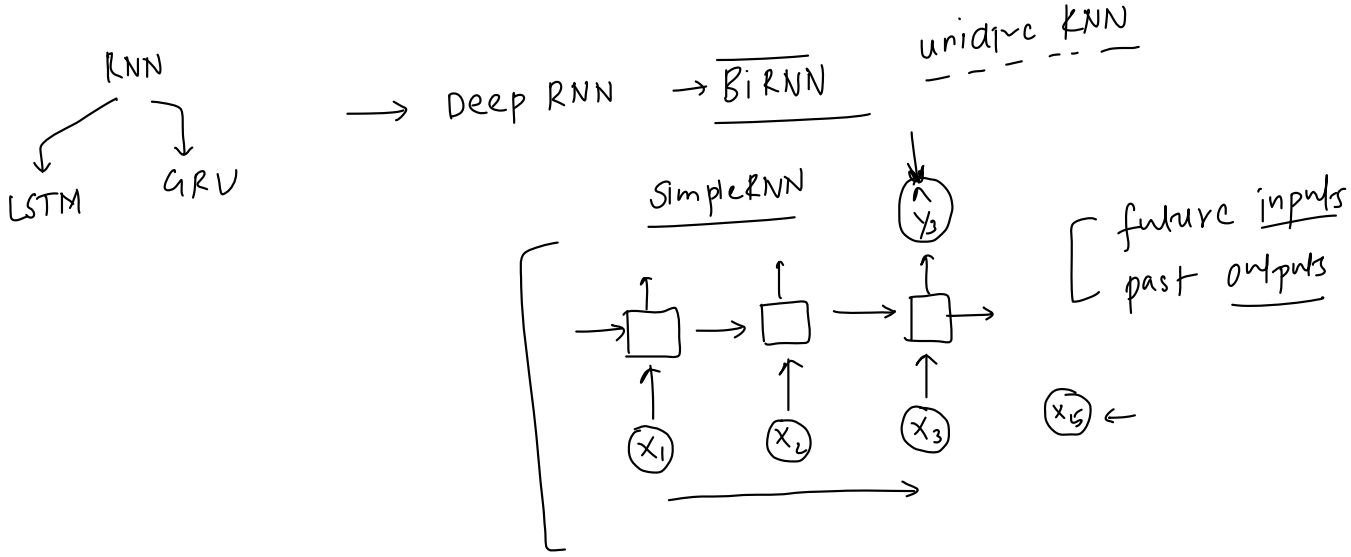
Disadvantages

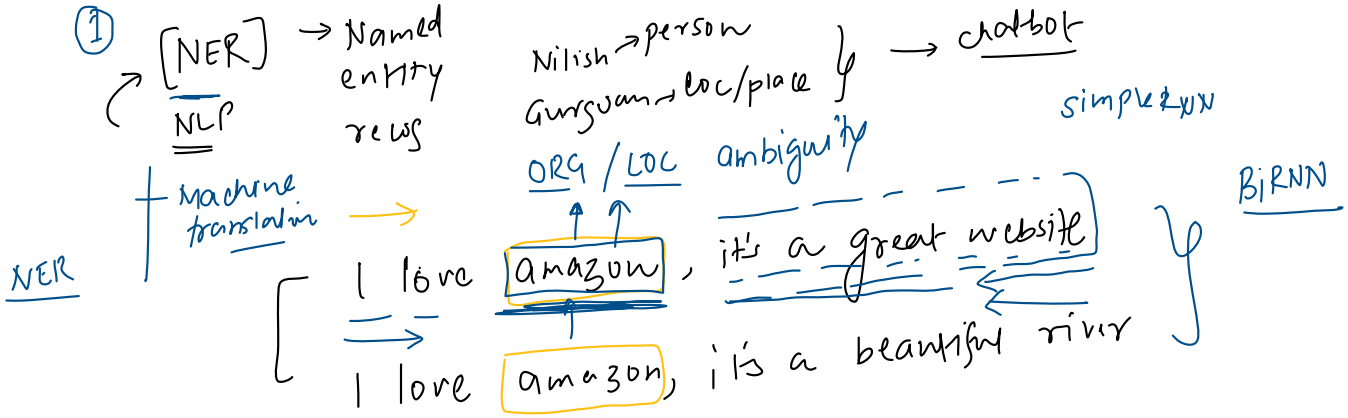
17 October 2023 16:30



The Why?
26 October 2023 15:19

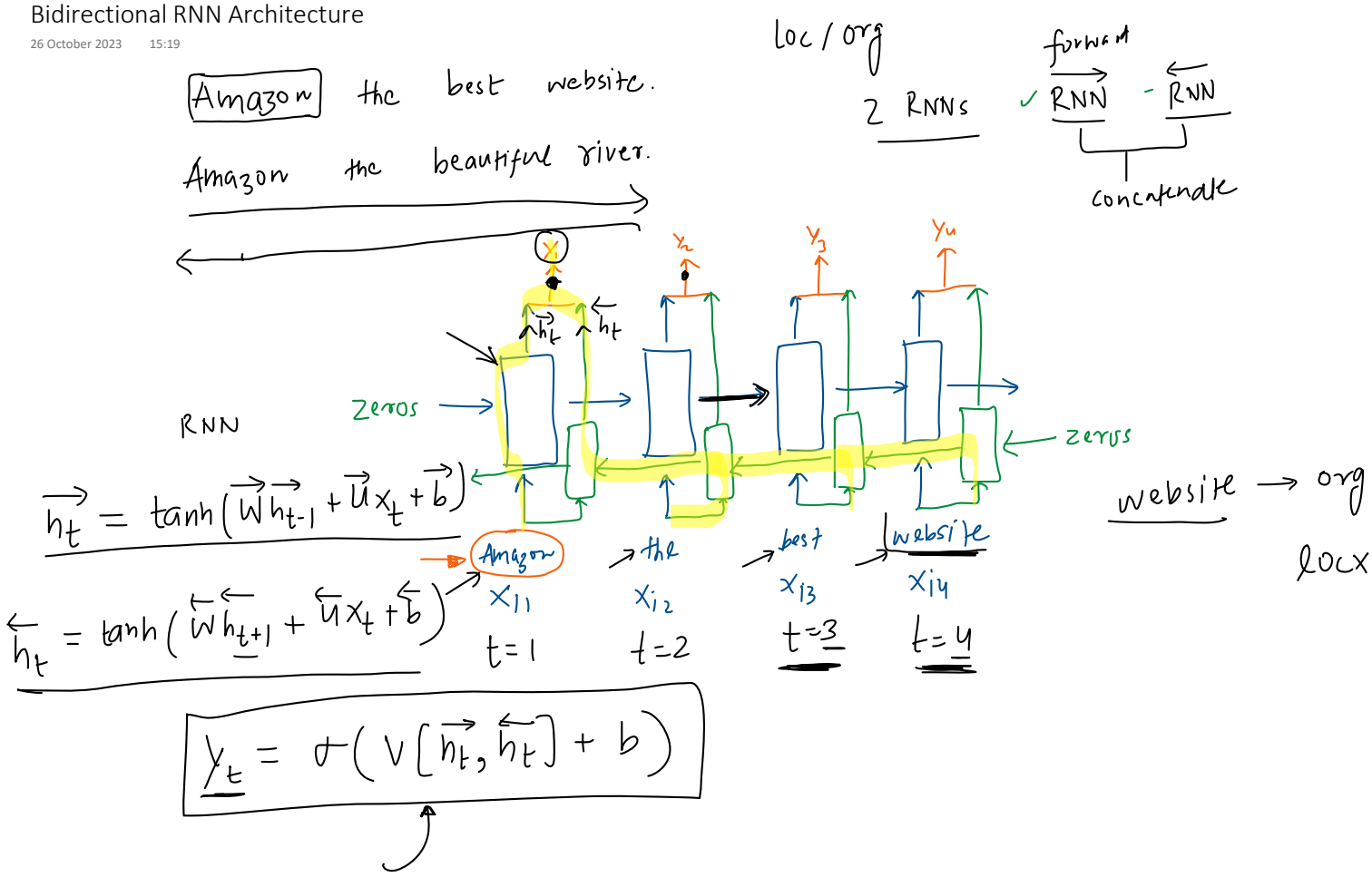
BiRNN





Bidirectional RNN Architecture

26 October 2023 15:19



Code

26 October 2023 15:21