GRU, Seq to Seq mapping

First see this practical ipynb file:
Deep_and_Bi_directional_RNN_LSTM_GRU.ipynb

Following are the 3 types of Sequence to Sequence mappings:

1. One to many
2. many to one
3. many to many

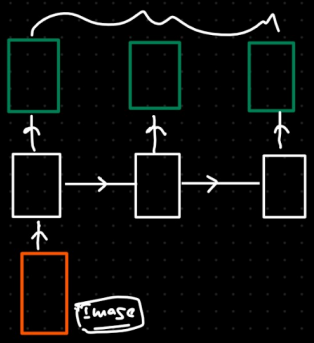
Encoder and decoder (2014)

- Seq to Seq

RNN => 1980-1990

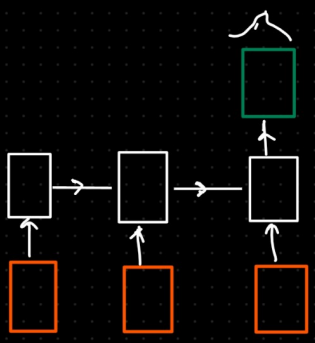
Seq to Seq => RNN, LSTM, GRU

One to many



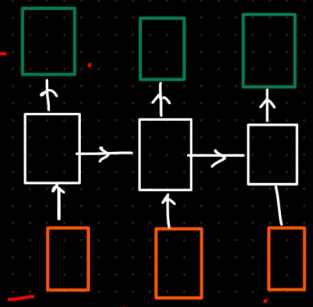
Application => Image captioning

many to one



Application => Sentiment analysis

many to many



Application => machine translation (Google Translator)

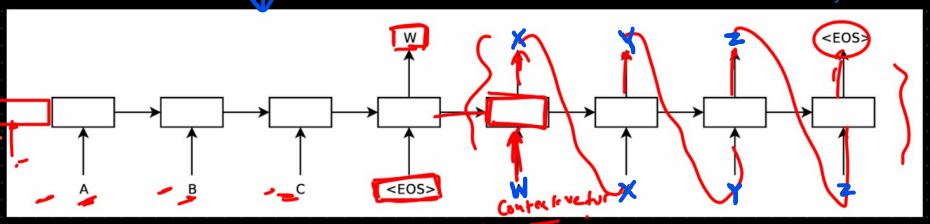
Seq to Seq => classical => RNN, LSTM, GRU

- Fix length output

Seq to Seq => variable length input & o/p

Encoder_Decoder Architecture

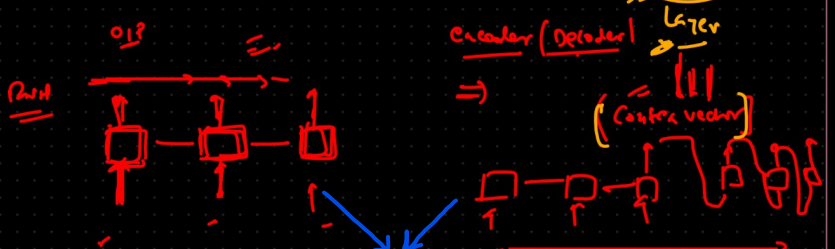
I/P: A B C
O/P: W X Y Z
Variable Dimensionality



EOS represents the End of Sentence. Cells upto first EOS represents Encoder side whereas, beyond that cells are of Decoder side. Output of Encoder is denoted by W which is context vector which is then passed to the decoder side in cascading fashion (Observe passing of W, X, Y, Z)

RNN, LSTM, GRU are considered as the classical seq to seq mapping models. Lets understand the drawback with this classical model.
So, these models works fine in case of one to many(Image captioning where images can be flatten and then passed) and many to one use case. But in case of many to many these are going to work where we have fixed length input and output. But in real word scenario Input and output length may varry. For eg: English to Hindi Google translator - "I wanna be" ->"Mai banna cahta hu" we can observe that when we passed input of len 3 in get translated into hindi with len 4. Classical models will fail in such case since they work only in case of fixed length.

The problem that has been stated above gave birth to modern day used architecture of seq to seq mapping called Encoder Decoder.



RNN/LSTM/GRU were the timestamp based approach where in each timestamp we get an output. There is no single vector obtained containing the context of whole input passed. Whereas, In Encoder Decoder which is not a timestamp based approach, we are wrapping up the entire Input info into a single vector called context vector and then using decoder to basically decode it.

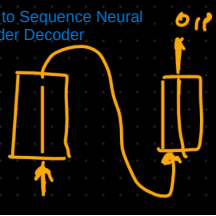
For evaluating classification based use case we were using performance metrices like accuracy, precision, recall, zscore etc.

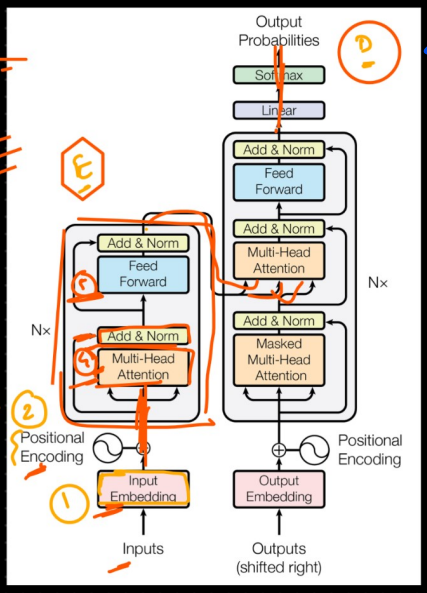
In the same way for machine/language translation use case we are going to use the performance metric called BLEU score.

If possible go on a high level through Sequence to Sequence Neural Network 2014 research paper which is for Encoder Decoder

Encoder_Decoder => LSTM, GRU, RNN

2018 => Google => transformer ~~LSTM~~ ~~GRU~~





Transformer Architecture. You may check you LinkedIn post for better understanding.

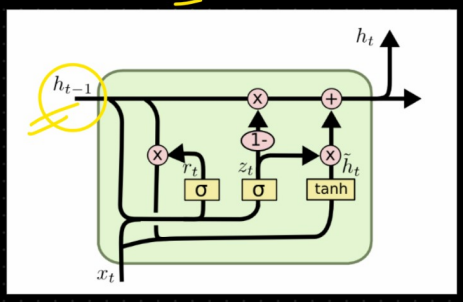
Note:
Please note that in the Encoder Decoder architecture we were still using RNN/LSTM/GRU as the hidden cell. There was one drawback of Encoder Decoder architecture, that is the context vector was able to hold the context of only 30-40 words at a time. But in real word scenario we can have words more than that. So, in 2018 Google published the research paper-"Attention is all you need" where they simply added an Attention layer on top of Encoder Decoder architecture and were able to remediate this issue. This new architecture is what we call as transformer that does not use RNN/LSTM/GRU and is used to power up the modern day LLMs.

Decoding GRU architecture:

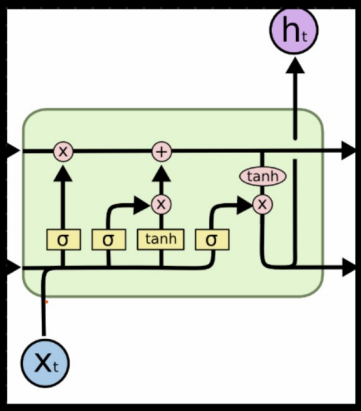
GRU

Gated Recurrent Unit

Difference b/w GRU and LSTM



GRU



LSTM

1. Number of gates

2 gates
1. Reset gate
2. Update

2. Memory Content
- Single hidden state

From architecture we can clearly observe that GRU only has hidden state from previous timestamp and not any separate cell/memory state. Therefore, it is going to use hidden state for both passing the output of current cell to next cell and also as a memory unit to hold the context. Due to this change only GRU is able to hold more long term dependencies as compared to LSTM

1. 3 gates
1. Forget
2. I/P or Update
3. O/P

2.
1. Cell state
2. Hidden State

Here we have both cell state Ct-1 and hidden state ht-1

3 Parameter Complexity

Less Parameter

Less trainable parameter since we have only 2 gates

4 Computation effecency

less intensive

5 Computation time

less

3

More Parameter

More trainable parameter since we have 3 gates

4 more Powerful and more intensive

5 more time

Community

- Github cont.
- Webmar
- Guest lecture

Education

- BE
- MS
- PHD

Awards | certifications | Activities

Sample

Assignment

1. Portfolio
2. Summary
3. Skill
4. Prof exp
5. Proj
6. Community cont / Awards / Activities / Certifications
7. education

Portfolio

Summary ⇒ 3-5

Skills

Prof. exp — Internship

Proj — main, Personal, Poc, hackthons

Comm cont | Award | Activ | Certi | Cour

Education

Approach

1. Job
2. Chatgpt
3. Wol site
4. Skills

⇒ Resume → mail.id

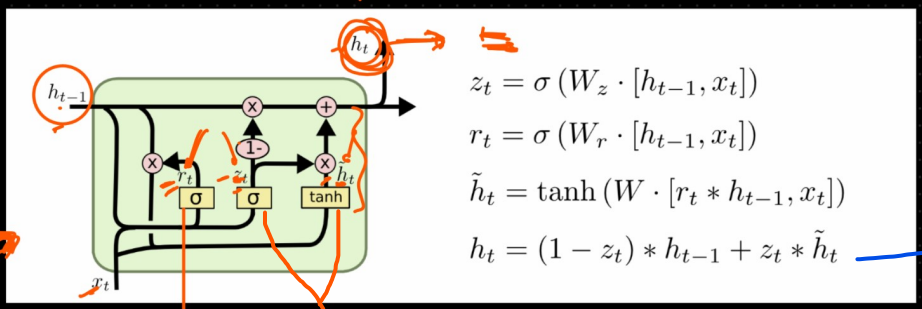
2 1-2

Long term dependency

Output

h

Mathematical Intuition of GRU:



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

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

Final output equation

hidden state

I/P

reset gate

update gate

word by word

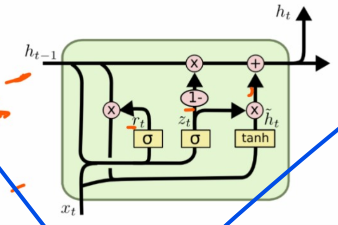
= on a diff timestamp

NN NN tanh

Acting Acting Acting

hidden layer hidden layer hidden layer

GRU



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

LSTM v/s GRU

this is happening

Why?

Scenario

$z_t = \sigma(W_z \cdot h_{t-1}, x_t])$

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

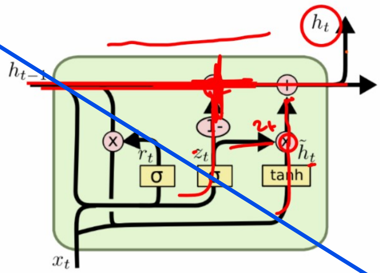
$\{\tilde{h}_t\}$ tanh $(W \cdot [r_t * h_{t-1}, x_t])$

Long-term Dependency

seq to seq   transformer

0

r_deg



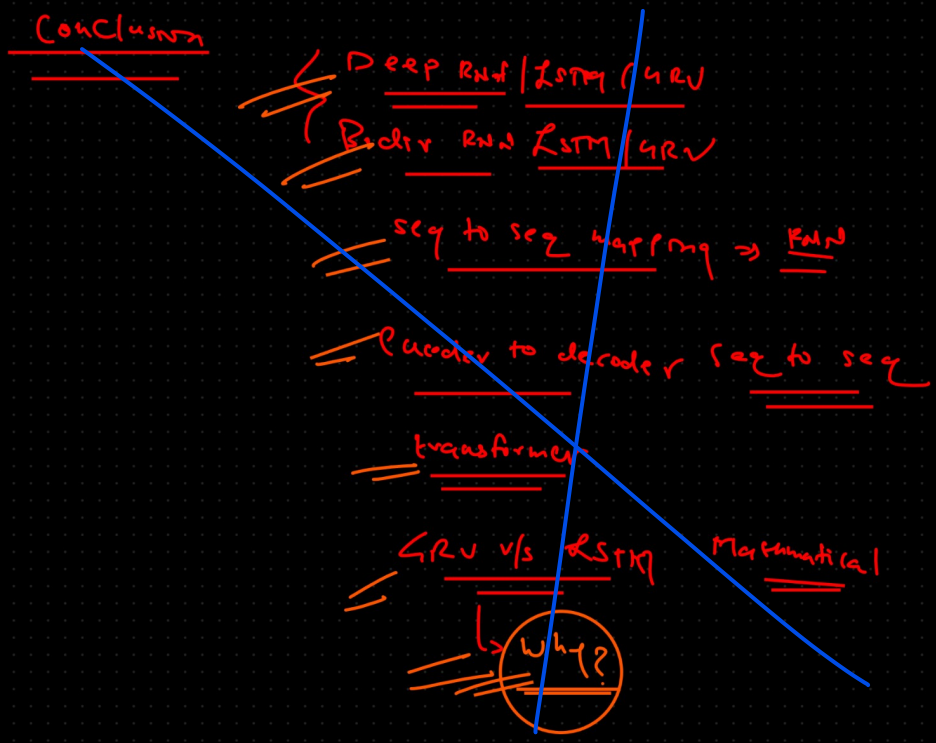
$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

(How)   Diff LSTM v/s GRU

Why? ⇒ Next → Context

=

theoretical ⇒ Project   Greats   OpenAI


Conclusion
Deep RNN / LSTM / GRU
Bi-dir RNN LSTM / GRU
seq to seq mapping ⇒ RNN
Encoder to decoder seq to seq
transformer
GRU v/s LSTM
Mathematical
Why?