



UE21CS343BB2

Topics in Deep Learning

Dr. Shylaja S S

Director of Cloud Computing & Big Data (CCBD), Centre
for Data Sciences & Applied Machine Learning (CDSAML)
Department of Computer Science and Engineering
shylaja.sharath@pes.edu

**Ack:Divija L,
Teaching Assistant**

GRU - Gated Recurrent Unit

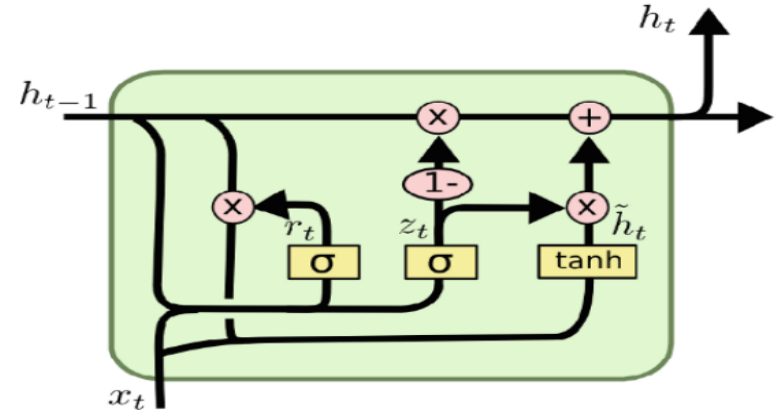


Revision about RNN's

- RNNs are designed to process sequential data by maintaining a form of memory based on previous inputs.
- This allows them to exhibit temporal dynamic behavior and capture information about a sequence's history. RNNs have complex architecture and the no. of parameters are more that makes it hard to train to do this.
- However, RNNs often struggle with learning long-term dependencies due to the **vanishing gradient problem**, where the contribution of information decays geometrically over time, making it difficult for the RNN to maintain a long-term memory.

Solution : **Gated Recurrent Unit(GRU)**

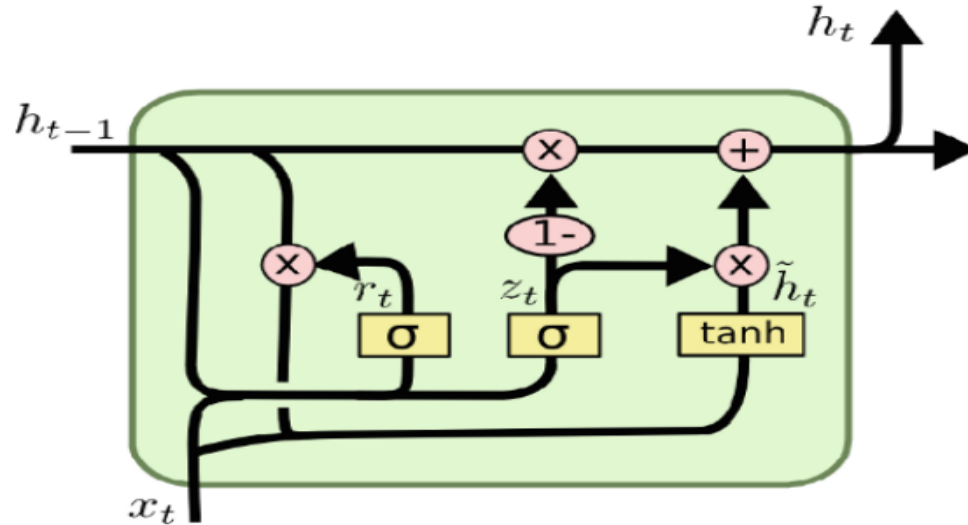
- Gated Recurrent Units were introduced by Kyunghyun Cho et al. in 2014.
- GRU has a simpler architecture compared to RNN (3 gates while GRU has 2 gates).
- Has few parameters compared to RNN making the training time lesser with comparable performance to LSTM.




- The main idea behind GRU is does not have cell state like LSTM it has only hidden state to capture the long and short term context dependencies.

The Gated Recurrent Unit Neural Networks basically consist of two gates i.e., Reset Gate and Update Gate. Reset Gates help capture short-term dependencies in sequences and Update Gates help capture long-term dependencies in sequences. Both the gates control how much each hidden unit has to remember or forget while working on the sequence.


GRU



Neural network fully connected layers with some activation functions with all 3 of same dimensionality


Neural Network Layer


Pointwise Operation


Vector Transfer


Concatenate

Applications of GRU

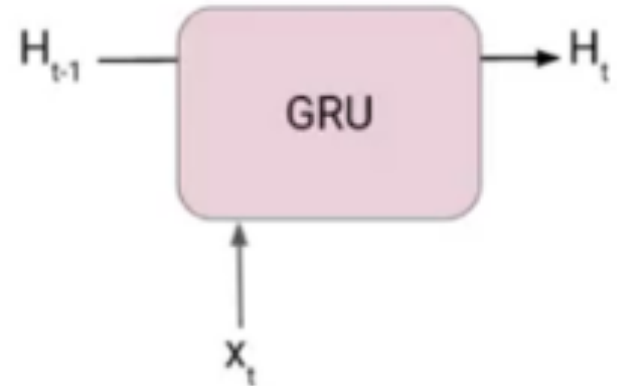
GRUs are used in tasks where sequence data is prevalent. Some applications include:

Language Modeling:

GRUs can predict the [probability](#) of a sequence of words or the next word in a sentence, which is useful for tasks like text generation or auto-completion.

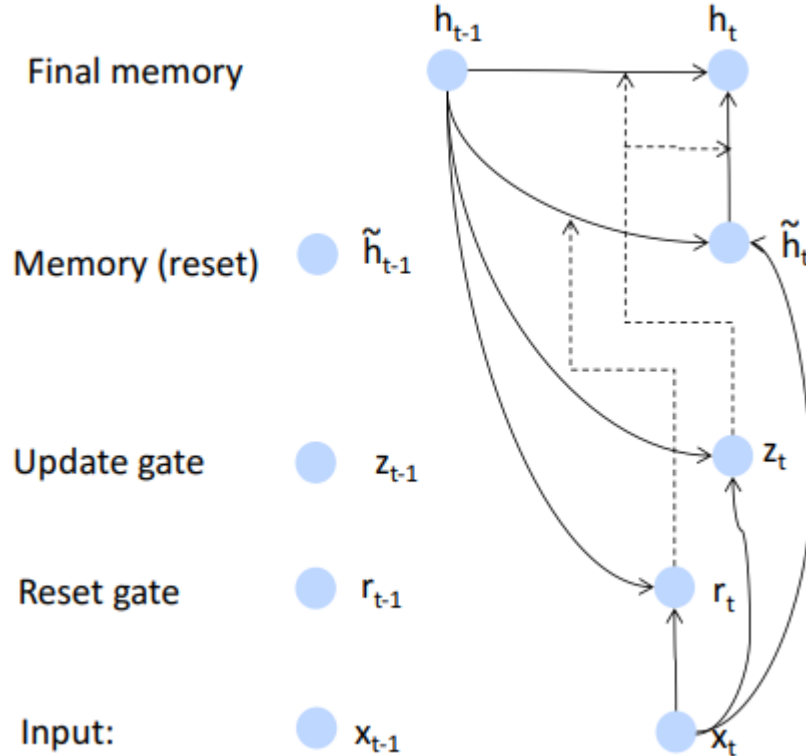
- **Machine Translation:** They can be used to translate text from one language to another by capturing the context of the input sequence.
- **Speech Recognition:** GRUs can process audio data over time to transcribe spoken language into text.
- **Time Series Analysis:** They are effective for predicting future values in a time series, such as stock prices or weather forecasts.

The goal of the GRU is for any timestamp t , the two inputs previous timestamp h_{t-1} And current timestamp x_t are used to find the current hidden state timestamp h_t .



-> Hidden state is the memory of the system

Flowchart of GRU



- Previous timestamp h_{t-1}
- Current timestamp h_t
- Current state x_t
- Reset Gate R_t
- Update Gate Z_t
- Candidate Hidden State

\tilde{h}_t

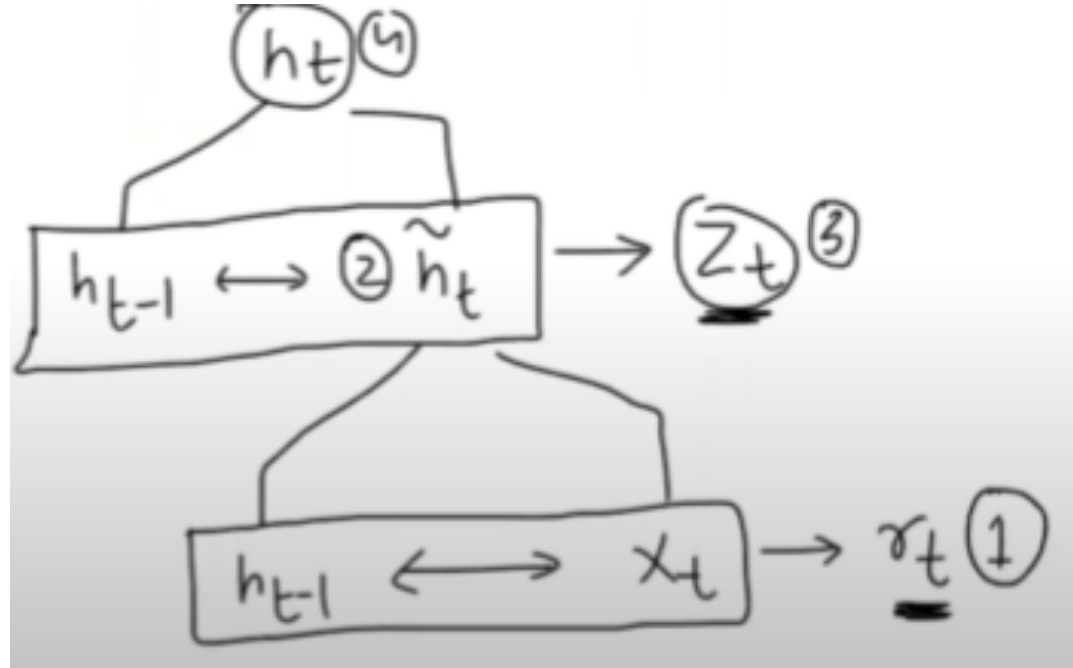
Steps of GRU

1) Calculate r_t reset gate

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

3) Calculate z_t update gate

4) Calculate h_t current
hidden state



Why are gates needed in GRU ?

h_{t-1} to \tilde{h}_t -> step 1 cannot be directly used as it is

too inclined towards X_t

Therefore we are using **reset gate**.

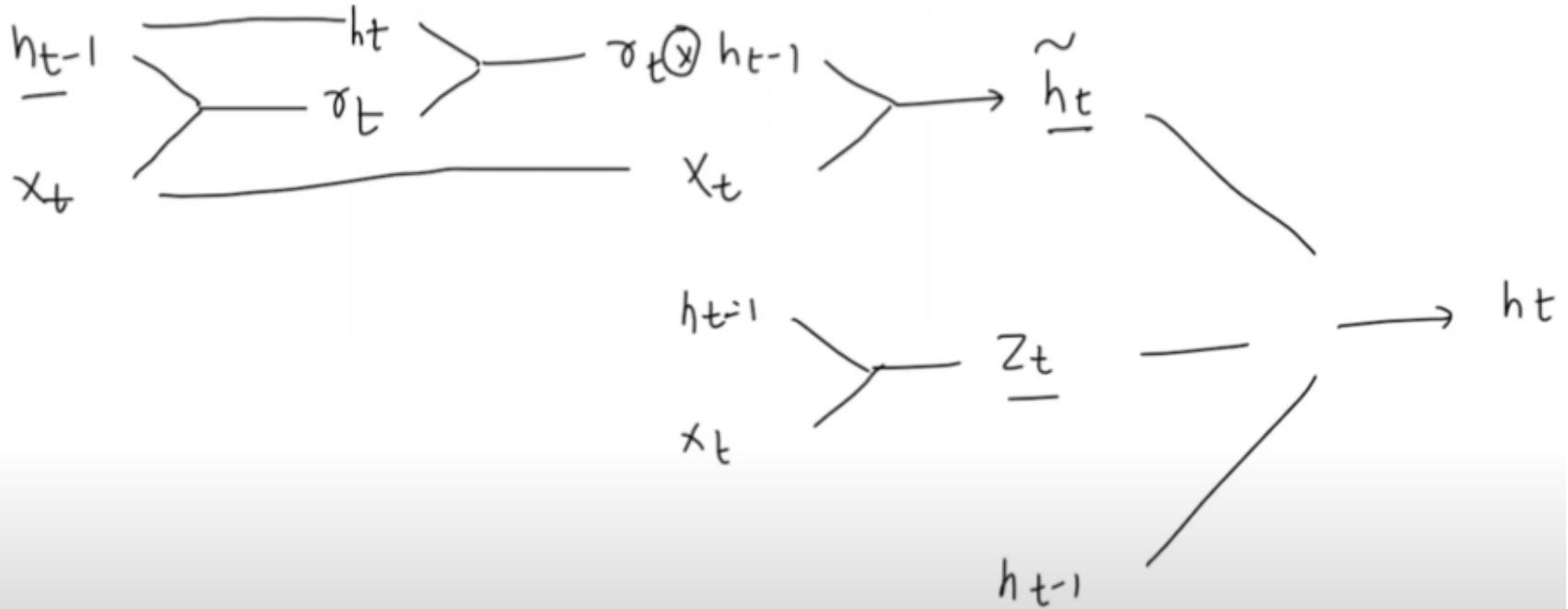
Why are gates needed in GRU

In step 2 : $\tilde{h}_t \rightarrow h_t \Rightarrow$ using **update gate**.

- The update gate helps the model decide how much information from the candidate hidden state should be incorporated into the current hidden state.
- It acts as a filter, determining the balance between retaining information from the previous hidden state and incorporating information from the current input.

h_{t-1} and \tilde{h}_t are balanced to eventually to achieve h_t .

Overall steps in GRU



Detailed steps in GRU

Step 1: Using h_{t-1} and X_t Reset gate r_t is calculated.

It has values lying between 0 to 1.

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

- If reset is close to 0, ignore previous hidden state. This allows model to drop information that is irrelevant in the future.
- helps the model decide how much information from the previous hidden state should be "reset" or forgotten, allowing the model to focus more on the current input X_t .
- This is important to avoid a bias towards the previous hidden state and make the model more adaptable to the current input.

Why are gates needed in GRU

Step 2: Modulated past memory with reset gate

$$r_t \otimes h_{t-1}$$

- The pointwise multiplication allows the GRU to be adaptive to the input sequence. It enables the model to learn which parts of the previous hidden state are relevant for the current input and which parts should be updated.
- This mechanism helps in addressing the vanishing gradient problem by allowing the model to selectively retain information from the past. It facilitates the learning of long-term dependencies in sequential data.

Why are gates needed in GRU

Step 3:

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

- The tanh function is applied element-wise to the linear combination of the concatenated input and the bias term.
- The purpose of using tanh is to squash the values to the range $(-1,1)$.
- Tanh \Rightarrow brings in Linearity as well as handling gradients.

Why are gates needed in GRU

Step 4:

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

- Update gate z controls how much of the past state should matter now.
- If z close to 1, then we can copy information in that unit through many time steps! **Less vanishing gradient!**

Why are gates needed in GRU

Step 5:

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

$(1 - z_t) \otimes h_{t-1} \Rightarrow$ represents the contribution of the previous hidden state

$z_t \otimes \tilde{h}_t \Rightarrow$ represents the contribution of the candidate hidden state

- Units with short-term dependencies often have reset gates very active.
- Units with long-term dependencies have active update gates z .

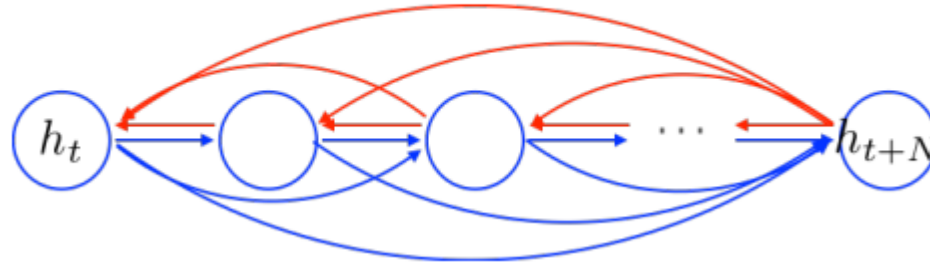
Is the problem with standard RNNs the naïve transition function?

$$h_t = f \left(W^{(hh)} h_{t-1} + W^{(hx)} x_t \right)$$

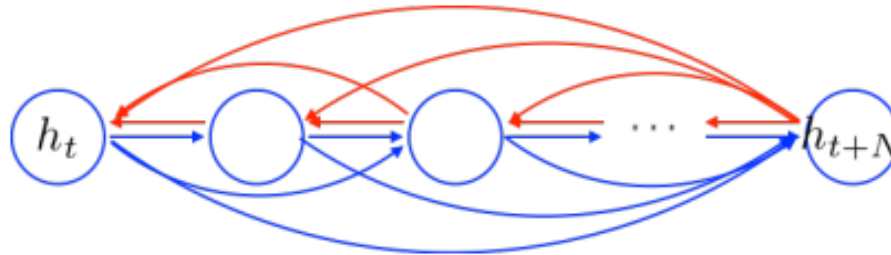
It implies that the error must backpropagate through all the intermediate nodes:



Perhaps we can create shortcut connections.



Perhaps we can create *adaptive* shortcut connections.
Let the net prune unnecessary connections *adaptively*.



That is what the gates do :

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

$$\tilde{h}_t = \tanh(W [x_t] + U(r_t \odot h_{t-1}) + b)$$

$$u_t = \sigma(W_u [x_t] + U_u h_{t-1} + b_u)$$

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

Difference between LSTM and GRU

1. NO OF GATES

LSTM : 3gates like input(or update) gate, forget gate, output gate

GRU : 2 gates like reset gate and update gate

1. MEMORY UNITS

LSTM: uses 2 separate states the cell state(ct) and hidden state (ht) the cell state acts as an internal memory and is crucial for carrying long term dependencies.

GRU: uses a single hidden state ht to both capture and output the memory

Difference between LSTM and GRU

3. PARAMETER COUNT:

LSTM: More parameters than GRU

For eg: input size is d and hidden state is h

$4 \times ((d \times h) + (h \times h) + h)$ parameters

GRU: few parameters

For eg: input size is d and hidden state is h

$3 \times ((d \times h) + (h \times h) + h)$ parameters

Difference between LSTM and GRU

4. COMPUTATIONAL COMPLEXITY:

LSTM: Due to extra gate and cell state, It is typically more computationally intensive than GRUs

GRU: is simpler and can be faster to compute, especially simple datasets or when computational resources are limited.

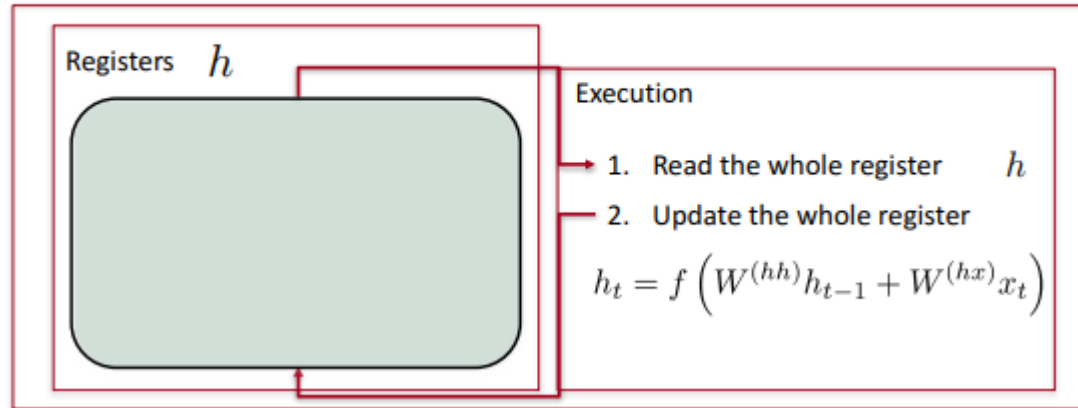
5. EMPIRICAL PERFORMANCE:

LSTM: In many tasks, especially complex ones, LSTM have been observed better GRUs

GRU: In certain tasks, GRU performs better , when data is limited or tasks are simpler. They can also train faster due to fewer parameters.

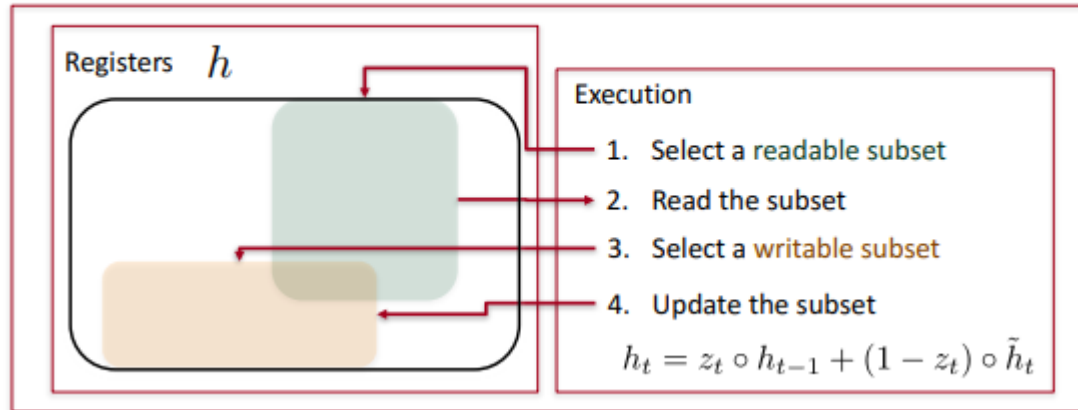
GRU Comparison to Standard tanh-RNN:

Vanilla RNN ...



GRU Comparison to Standard tanh-RNN:

GRU ...



Gated recurrent units are much more versatile and adaptive in which elements of the hidden vector h they update!



UE21CS343BB2

Topics in Deep Learning

Dr. Shylaja S S

Director of Cloud Computing & Big Data (CCBD), Centre
for Data Sciences & Applied Machine Learning (CDSAML)
Department of Computer Science and Engineering
shylaja.sharath@pes.edu

**Ack:Divija L,
Teaching Assistant**