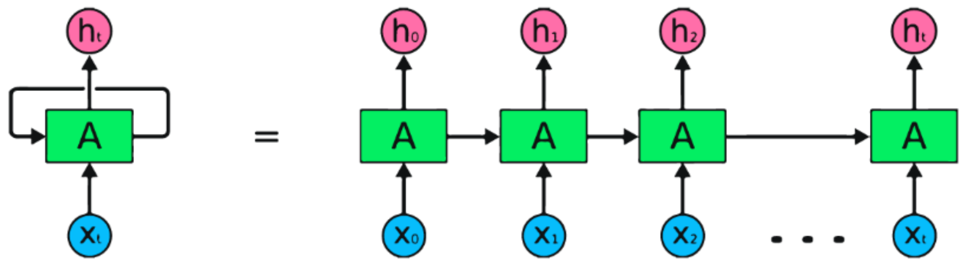
Recurrent Neural Network (RNN)

- Recurrent Neural Network(RNN) is a type of Neural Network
- it is used in Apple's Siri and Google's voice search.
- RNN remembers past inputs due to an internal memory which is useful for predicting stock prices, generating text, transcriptions, and machine translation.
- In the traditional neural network, the inputs and the outputs are independent of each other, whereas the output in RNN is dependent on prior elementals within the sequence.



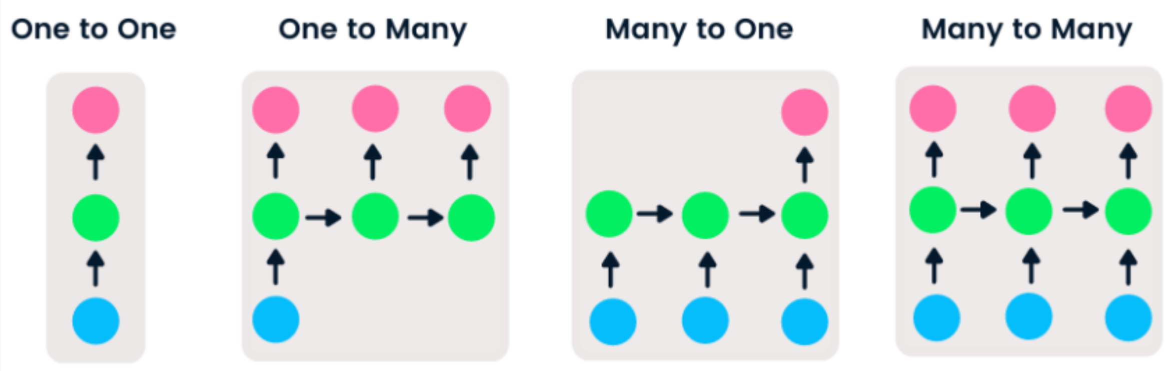
RNN

Types of Recurrent Neural Networks

Feedforward networks have single input and output, while recurrent neural networks are flexible as the length of inputs and outputs can be changed. This flexibility allows RNNs to generate music, sentiment classification, and machine translation.

There are four types of RNN based on different lengths of inputs and outputs.

- **One-to-one** is a simple neural network. It is commonly used for machine learning problems that have a single input and output.
- **One-to-many** has a single input and multiple outputs. This is used for generating image captions.
- **Many-to-one** takes a sequence of multiple inputs and predicts a single output. It is popular in sentiment classification, where the input is text and the output is a category.
- **Many-to-many** takes multiple inputs and outputs. The most common application is machine translation.



Advantages

- **The processing of sequential data.**

- Ability to remember and preserve primary outcomes.
- When calculating new results, consider the most recent and previous results.
- The model size does not change as the input size does. Over time, it distributes weights to other components.

Disadvantages of Recurrent Neural Network

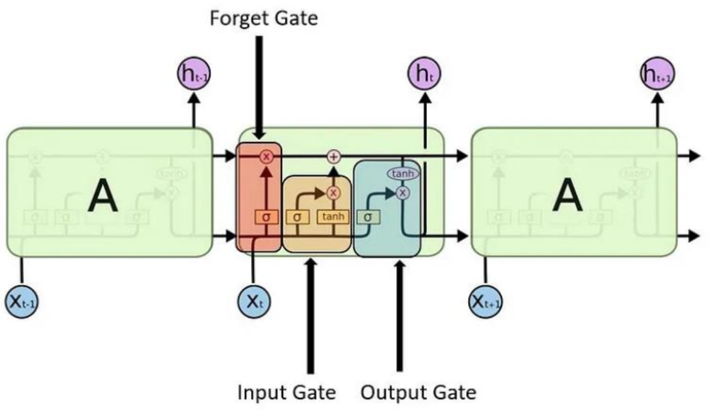
1. Gradient vanishing and exploding problems.
2. Training an RNN is a very difficult task.
3. It cannot process very long sequences if using tanh or relu as an activation function.

Applications of Recurrent Neural Network

1. Language Modelling and Generating Text
2. Speech Recognition
3. Machine Translation
4. Image Recognition, Face detection
5. Time series Forecasting

Recurrent Neural Network (LSTM):

- LSTM networks are a **modified version** of recurrent neural networks.
- It makes **easier to remember past data** in memory.
- **vanishing gradient** problem of RNN is resolved here.
- LSTM is **well-suited to classify**, **process** and **predict time series** given time lags of unknown duration.
- It trains the model by using **back-propagation**.
- widely used in Deep Learning.
- LSTM Stands for long short term memory.



LSTM gates

In an LSTM network, **three gates** are present:

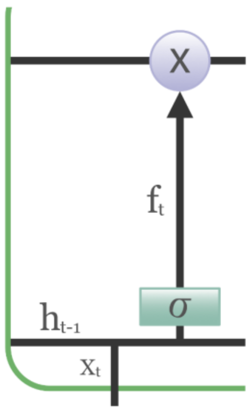
1. **Forget Gate:**
2. **Input gate:**
3. **Output gate:**

1. **Forget Gate:**
    - It is decided by the **sigmoid function.**
    - it looks at the previous state(**ht-1**) and the content input(**Xt**) and outputs a number between **0(***omit this*)and **1(***keep this***)**for each number in the cell state **Ct−1**.

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

- Wf represents the weight matrix associated with the forget gate.

- [ht-1, xt] denotes the concatenation of the current input and the previous hidden state.
- bf is the bias with the forget gate.
- σ is the sigmoid activation function.



2. **Input gate:**
    - Discover which value from input should be used to modify the memory.
    - **Sigmoid** function decides which values to let through **0,1.** and **tanh** function gives weightage to the values which are passed deciding their level of importance ranging from**-1** to **1.**

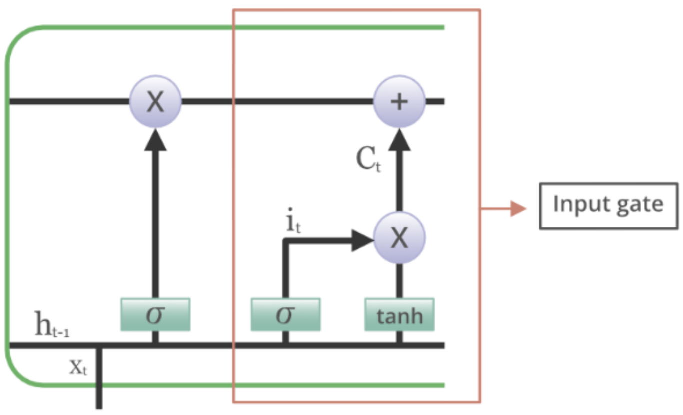
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

it = σ(Wi · [ht-1, xt] + bi)

Ĉt = tanh(Wc · [ht-1, xt] + bc)

Ct = ft ⊙ Ct-1 + it ⊙ Ĉt

- ⊙ denotes element-wise multiplication
- tanh is tanh activation function



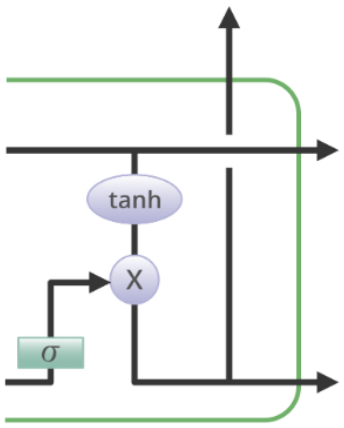
3. Output gate:

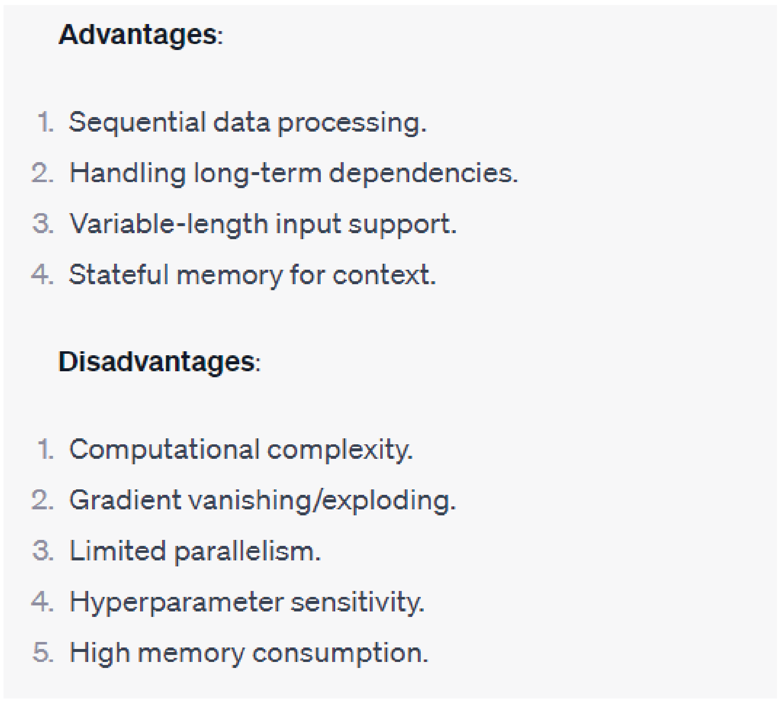
- the input and the memory of the block is used to decide the output.
- **Sigmoid** function decides which values to let through **0,1.** and **tanh** function gives weightage to the values which are passed deciding their level of importance ranging from**-1** to **1** and multiplied with output of **Sigmoid.**

$$o_t = \sigma\left(W_o \; \left[h_{t-1}, x_t\right] \; + \; b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

o_t = σ(W_o · [h_t-1, x_t] + b_o)





Advantages:

1. Sequential data processing.
2. Handling long-term dependencies.
3. Variable-length input support.
4. Stateful memory for context.

Disadvantages:

1. Computational complexity.
2. Gradient vanishing/exploding.
3. Limited parallelism.
4. Hyperparameter sensitivity.
5. High memory consumption.

Applications:

1. NLP (Natural Language Processing)
2. Speech Recognition
3. Time Series Analysis
4. Healthcare
5. Video Analysis
6. Autonomous Vehicles

Gated Recurrent Unit (GRU)

- GRU (Gated Recurrent Units) are similar to the LSTM networks.
- (GRU) is a type of recurrent neural network (RNN)
- GRU can process sequential data such as text, speech, and time-series data.

there are some differences between GRU and LSTM.

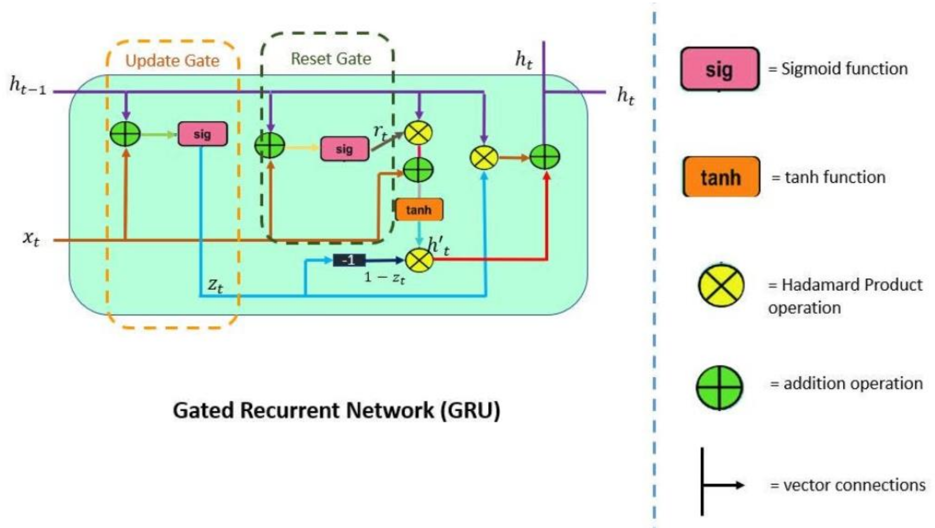
- GRU doesn't contain a cell state
- GRU uses its hidden states to transport information
- It Contains only 2 gates(Reset and Update Gate)
- GRU is faster than LSTM
- GRU has lesser tensor's operation that makes it faster.

1. Update Gate

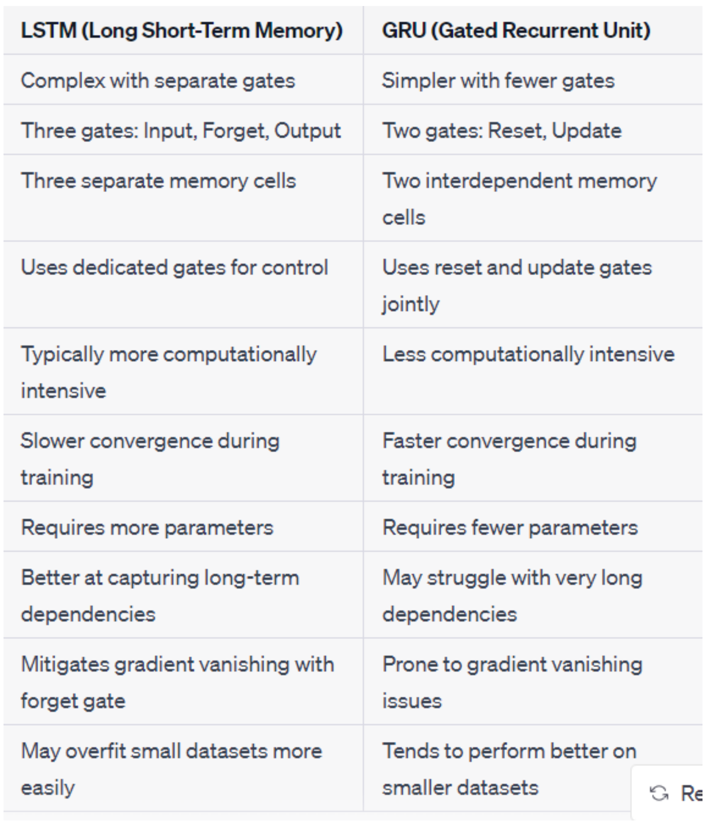
Update Gate is a combination of Forget Gate and Input Gate. Forget gate decides what information to ignore and what information to add in memory.

2. Reset Gate

This Gate Resets the past information in order to get rid of gradient explosion. Reset Gate determines how much past information should be forgotten.



Gated Recurrent Network (GRU)



LSTM (Long Short-Term Memory)	GRU (Gated Recurrent Unit)
Complex with separate gates	Simpler with fewer gates
Three gates: Input, Forget, Output	Two gates: Reset, Update
Three separate memory cells	Two interdependent memory cells
Uses dedicated gates for control	Uses reset and update gates jointly
Typically more computationally intensive	Less computationally intensive
Slower convergence during training	Faster convergence during training
Requires more parameters	Requires fewer parameters
Better at capturing long-term dependencies	May struggle with very long dependencies
Mitigates gradient vanishing with forget gate	Prone to gradient vanishing issues
May overfit small datasets more easily	Tends to perform better on smaller datasets