[😇Mission Torch 2 (RNN, GRU, LSTM) | Kaggle](https://www.kaggle.com/code/rushinaik/mission-torch-2-rnn-gru-lstm)

Architecture of RNN :

1. Input layer : Vectors which we will feed in RNN layer
2. previous hidden state : Previous hidden state is the output of previous RNN which are supposed to feed in second layer. It works like we will take input and then we will process the data. we will feed the processed data to next layer in that way the information in sequence vector will be used
3. Tanh activation : The tanh activation is used to help regulate the values flowing through the network. Tanh function transform the vectors values between -1 to 1. so if we have multplication operation in our NN it will be huge after certain iterations, so here tanh will come handy and convert it in to -1 to 1
4. Concatenation : General Addition of two vectors

# But Why RNN failed and LSTM & GRU invented? 🧐

### **Vanishing Gradient:**

* so as we start to feeding the long sentences to the RNN model the 1st hidden layers sends the information to 2nd hidden layer and 2nd to 3rd hidden layer and this goes on.
* So while backpropagation we adjust the weights but as coming till first layer the gradient descent value becomes so small that it will not do any learning and the information will get lost
* And this was the main reason of born of LSTM & GRU: GATED RECURRENT NEURAL NETWORK Which has update gate, forget gate etc that will look next

### **LSTM :**

LSTM’s are the cell state, and it’s various gates. The cell state act as a transport highway that transfers relative information all the way down the sequence chain. You can think of it as the “memory” of the network. The cell state, in theory, can carry relevant information throughout the processing of the sequence. So even information from the earlier time steps can make it’s way to later time steps, reducing the effects of short-term memory. As the cell state goes on its journey, information get’s added or removed to the cell state via gates. The gates are different neural networks that decide which information is allowed on the cell state. The gates can learn what information is relevant to keep or forget during training.

* will update the theory in details

## **GRU**

* The GRU is the newer generation of Recurrent Neural networks and is pretty similar to an LSTM. GRU’s got rid of the cell state and used the hidden state to transfer information. It also only has two gates, a reset gate and update gate.
* Update Gate

The update gate acts similar to the forget and input gate of an LSTM. It decides what information to throw away and what new information to add.

* Reset Gate

The reset gate is another gate is used to decide how much past information to forget.

And that’s a GRU. GRU’s has fewer tensor operations; therefore, they are a little speedier to train then LSTM’s. There isn’t a clear winner which one is better. Researchers and engineers usually try both to determine which one works better for their use case

### **Class Sequences :**

* it will take text dataset as input and processed the text, tokenize it to sequences, pad it
* function **getitem** willl return the item at particular index
* **len** return lenght of the sequence

[🤖LSTM+BERT💬 | 🔴IMDB Sentiment Analysis🟢 | Kaggle](https://www.kaggle.com/code/shivanirana63/lstm-bert-imdb-sentiment-analysis)

The data consists of 50000 comments. There are 25000 negative and 25000 positive comments. Comments are of different lengths and are given in sentences. Using the imdb dataset, we will train our LSTM model with positive and negative comments about movies. First, we will make comments available for the LSTM model. After training our model, we will examine the accuracy and loss values on the chart.

## **Text Cleaning**

linkcode

* process of clearing punctuation marks in data
* cleaning unnecessary marks in data. </p> </li>
* capitalization to lowercase. </p> </li>
* cleaning extra spaces. </p> </li>
* removal of stopwords in sentences. </p> </li>

[LSTM based 50K IMDB movie reviews classification | Kaggle](https://www.kaggle.com/code/akanksha125/lstm-based-50k-imdb-movie-reviews-classification)

**PROBLEM DESCRIPTION**

This notebook demostrates a **sequence classification of IMDB movie reviews** dataset by creating a simple LSTM based classifier.

Each movie review is a variable sequence of words, and the tone of each movie review must be classified. The large movie reviews dataset (sometimes referred to as the IMDB dataset) contains *25,000 film reviews (good or bad) for training and 25,000 reviews for testing*. The problem is deciding whether a given movie review is positive or negative. The data were collected by researchers at Stanford and used in a 2011 paper that used 50-50 data for training and testing. An accuracy of 88.89% is achieved.

linkcode

**Import modules**

Let's start off with the basic step of importing all the relevant modules and functions required for this particular classifier.

**Load and Clean Dataset**

In the original dataset, the reviews are still dirty. There are still html tags, numbers, uppercase, and punctuations. This will not be good for training, so in load\_dataset() function, beside loading the dataset using pandas, I also pre-process the reviews by removing html tags, non alphabet (punctuations and numbers), stop words, and lower case all of the reviews.

**Tokenize and Pad/Truncate Reviews**

A Neural Network only accepts numeric data, so we need to encode the reviews.

Each reviews has a different length, so we need to add padding (by adding 0) or truncating the words to the same length (in this case, it is the mean of all reviews length) using tensorflow.keras.preprocessing.sequence.pad\_sequences.

**Define the LSTM model**

The first layer is the Embedded layer that uses 32 length vectors to represent each word. The next layer is the LSTM layer with 100 memory units (smart neurons). Subsequently, you can add more than one LSTM layer. Finally, because this is a classification problem we use a Dense output layer with a single neuron and a sigmoid activation function to make 0 or 1 predictions for the two classes (good and bad) in the problem.

Because it is a *binary classification problem*, log loss is used as the loss function (binary\_crossentropy in Keras). The efficient ADAM optimization algorithm is used. The number of epochs and batch size can be increased as per the requirement.

[IMDb sentiment analysis: Keras LSTM | Kaggle](https://www.kaggle.com/code/sumishog/imdb-sentiment-analysis-keras-lstm)

Text cleaning

# Baseline linear model

Before we build time-consuming neural network model, simple classifier is tried. Tfidf method is used for text vectorization.

Logistic regression provides 90% accuracy for test set, which is fast and adapted for such large datasets.

# 3. LSTM model

linkcode

To quickly explore LSTM model, we firstly take only 500 characters. That is much faster way than taking full sentences.

### **Tokenize & Padding**

linkcode

Let's convert tokens to ID. We will use keras Tokenizer which can filter punctuations and take only most-counted 10000 words.

Sentiment is converted to 0 (negative) or 1 (positive).

Let's build the simple LSTM model using first 500 characters datasets.