Time Series Forecasting(LSTM & GRU)

Zeham Management Technologies BootCamp by SDAIA

August 19th, 2024

SDAIA
الهيئة السعودية للبيانات والذكاء الاصطناعي
Saudi Data & AI Authority

Introduction to LSTM & GRU

Let’s start together…

Agenda

- Introduction to LSTM
- Architecture of LSTM
- Advantage over standard RNN’s
- Introduction to GRUs.
- GRU Architecture.
- Applications and Performance Comparisons.

Introduction Long Short Term Memory (LSTM)

Introduction to LSTM

LSTM excels in sequence prediction tasks, capturing long-term dependencies. Ideal for time series, machine translation, and speech recognition due to order dependence. The article provides an in-depth introduction to LSTM, covering the LSTM model, architecture, working principles, and the critical role they play in various applications.

What is LSTM?

Long Short-Term Memory (LSTM) is an advanced variant of recurrent neural networks (RNN) developed by Hochreiter and Schmidhuber. While traditional RNNs utilize a single hidden state that propagates through time, this approach struggles with learning long-term dependencies due to issues like vanishing gradients. LSTM networks overcome this limitation by incorporating a memory cell, a mechanism that can store information over longer periods, enabling more effective learning of long-term dependencies.

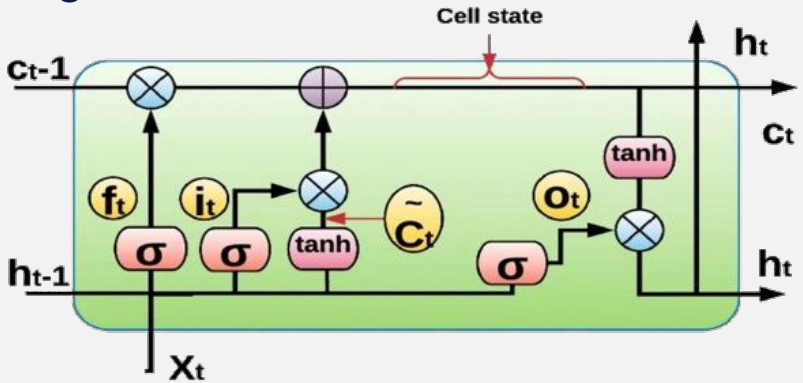
LSTM architectures are adept at learning long-term dependencies in sequential data. This capability makes them highly suitable for various tasks, including language translation, speech recognition, and time series forecasting.

What is LSTM?

LSTMs can also be used in combination with other neural network architectures, such as Convolutional Neural Networks (CNNs) for image and video analysis.

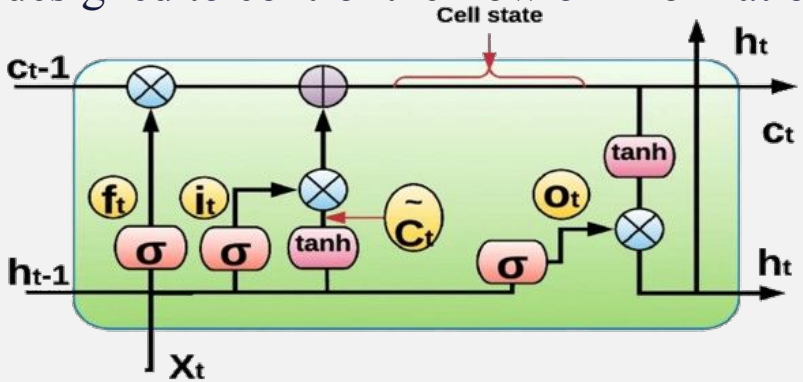
In summary ,

- **L**ong **S**hort-**T**erm **M**emory (LSTM) is a special type of Recurrent Neural Network (RNN) architecture.
- LSTMs implements feedback loops to solve with data sequences through temporal dependencies found in typical Feedforward Neural Networks.
- It is designed to solve the limitation of vanishing or exploding gradients happening in sequence training



In summary ,

- LSTM is useful for tasks that involve sequential data, such as natural language processing (NLP), Speech Recognition, and time series analysis.
- Memory cells are added in LSTM networks to maintain sequenced data.
- Each memory cell has three primary components: an input gate, a forget gate, and an output gate.
- These gates are designed to control the flow of information in/out of the memory cell.



Architecture of LSTM

Architecture of LSTM

LSTM architectures incorporate a memory cell managed by three gates: the input gate, the forget gate, and the output gate. These gates determine what information is added to, removed from, and output from the memory cell.

- Input Gate: Determines what new information is added to the memory cell.
- Forget Gate: Controls what information is discarded from the memory cell.
- Output Gate: Regulates what information is output from the memory cell.

Architecture of LSTM

This allows LSTM networks to selectively retain or discard information as it flows through the network, which allows them to learn long-term dependencies.

The LSTM maintains a hidden state, which acts as the short-term memory of the network. The hidden state is updated based on the input, the previous hidden state, and the memory cell's current state.

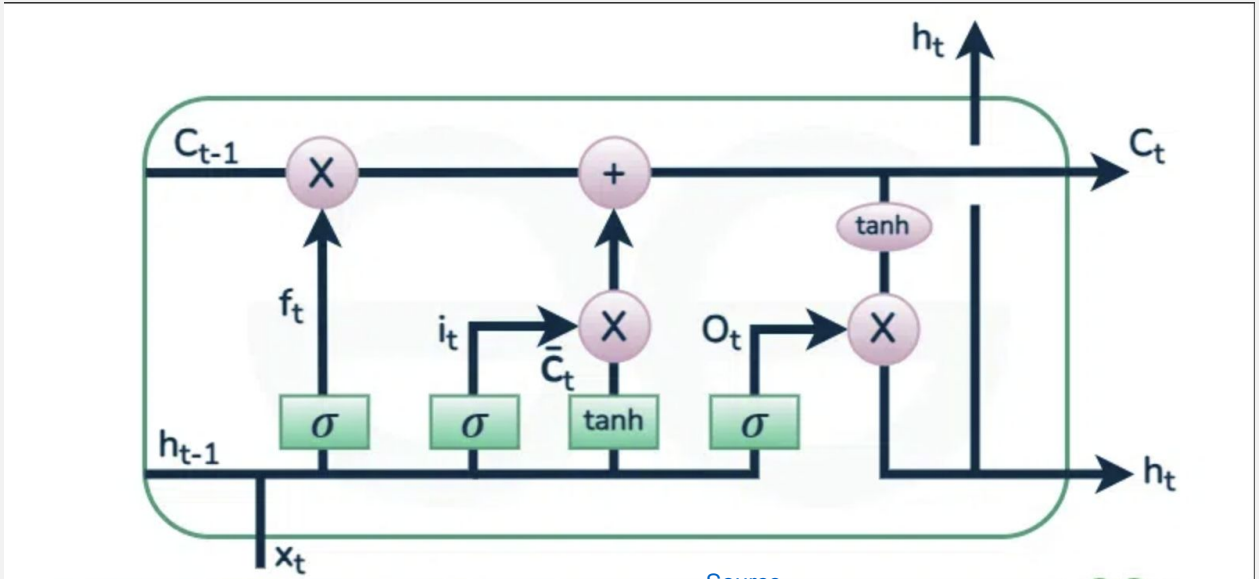
Bidirectional LSTM Model

Bidirectional LSTM (Bi LSTM/BLSTM) is a type of recurrent neural network (RNN) capable of processing sequential data in both forward and backward directions. This dual-directional approach allows Bi LSTMs to capture longer-range dependencies in sequential data more effectively than traditional LSTMs, which process data in only one direction.

- Bi LSTMs consist of two LSTM networks: one processes the input sequence in the forward direction, while the other processes it in the backward direction.
- The outputs from both LSTM networks are then combined to generate the final output.

Architecture of LSTM

LSTM architecture has a chain structure that contains four neural networks and different memory blocks called cells.

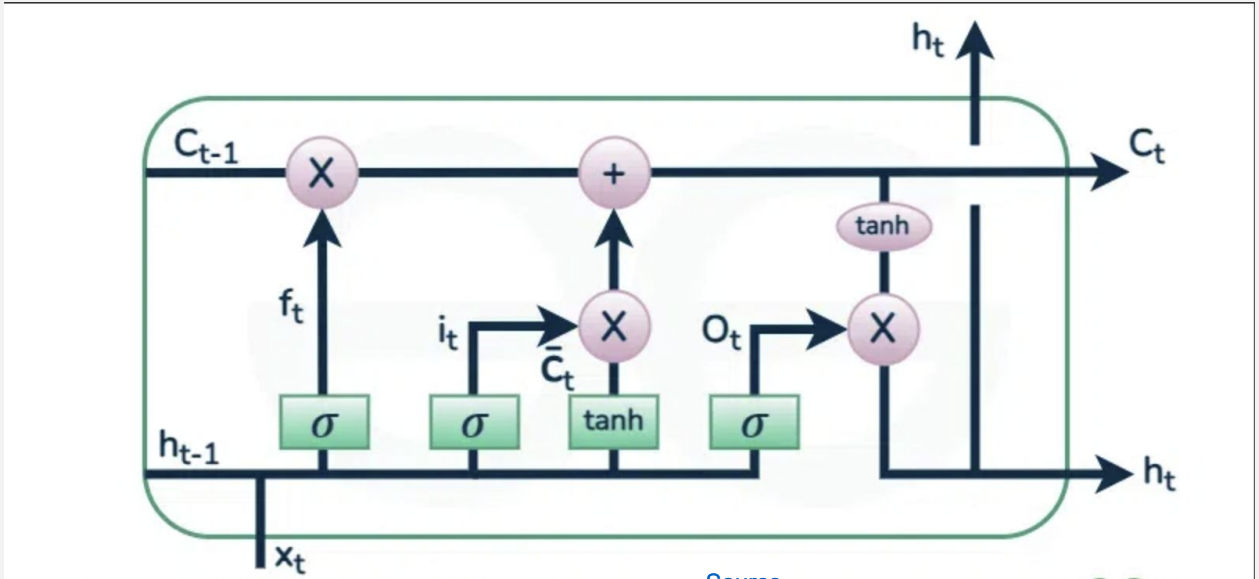


Source

Architecture of LSTM

Information is retained by the cells and the memory manipulations are done by the gates.

There are three gates –



Source

Architecture of LSTM

1) The input gate:

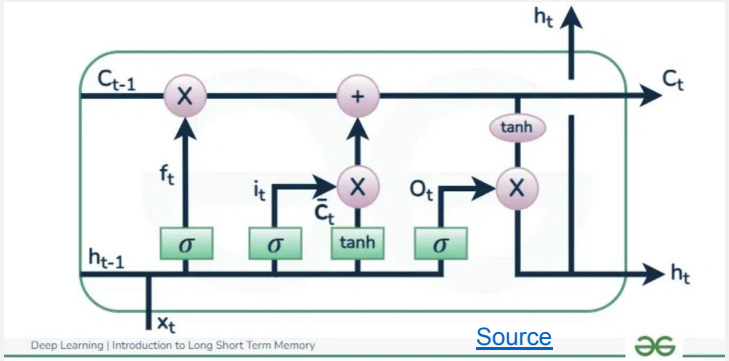
- The addition of useful information to the cell state is done by the input gate. First, the information is regulated using the sigmoid function and filter the values to be remembered similar to the forget gate using inputs ht-1 and xt. . Then, a vector is created using tanh function that gives an output from -1 to +1, which contains all the possible values from ht-1 and xt. At last, the values of the vector and the regulated values are multiplied to obtain the useful information.

Architecture of LSTM

1) The input gate:

The equation for the input gate is:

- $t=\sigma(Wi \cdot [ht-1,xt]+bi)$
- $C\hat{}t=\tanh(Wc \cdot [ht-1,xt]+bc)$



Source

We multiply the previous state by ft, disregarding the information we had previously chosen to ignore. Next, we include it∗Ct. This represents the updated candidate values, adjusted for the amount that we chose to update each state value.

Architecture of LSTM

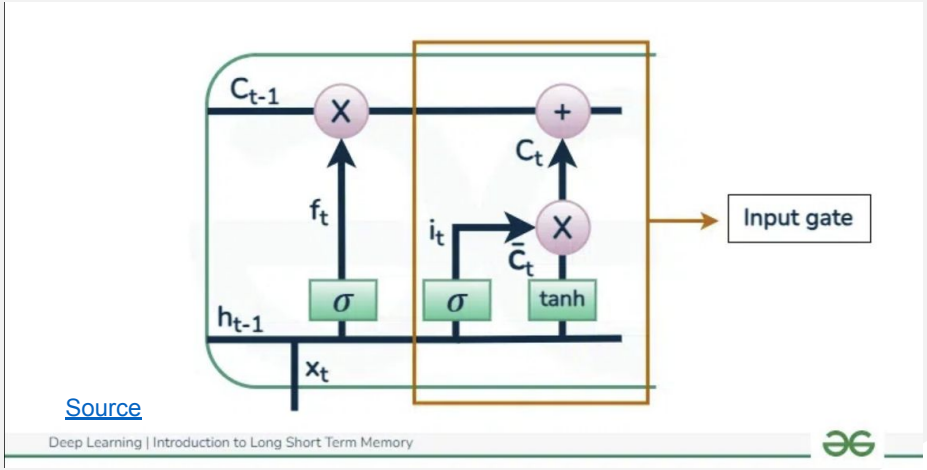
1) The input gate:

$C_t = f_t \odot C_{t-1} + i_t \odot \hat{C}_t$

where

⊙ denotes element-wise multiplication

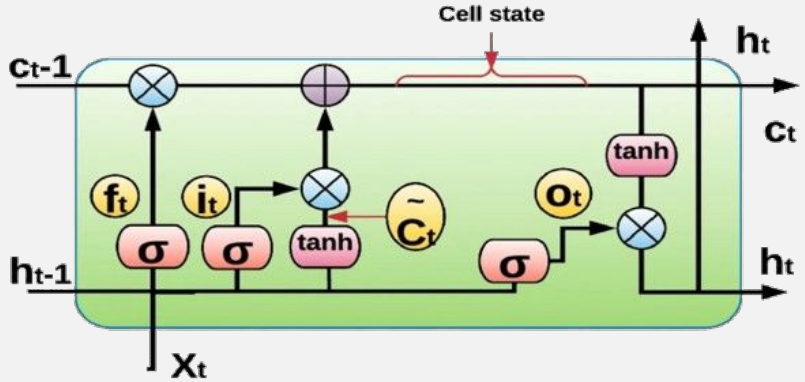
tanh is tanh activation function



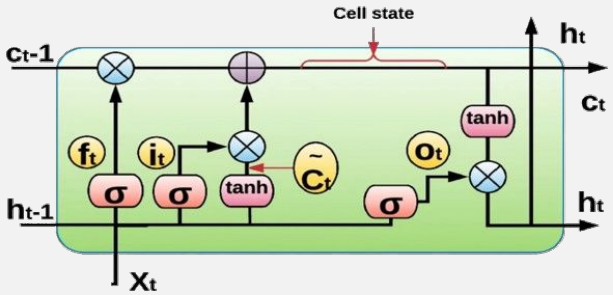
Architecture of LSTM

1) The input gate:

- control the new input that should be incorporated into the memory cell.
- It processes the current input and the previous hidden state, generating a value from 0 to 1 for each element of the memory cell, where each value dictates how much of the input is retained.



Architecture of LSTM



1) The input gate:

- Determines which new information should be stored in the cell state, involving two distinct

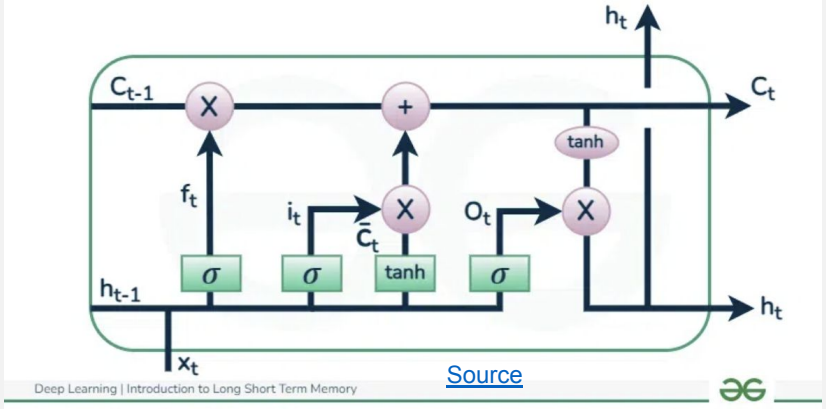
  components:

  1. sigmoid layer, known as the "input gate layer," that determines which values in the cell state are to be updated.
  2. tanh layer that produces a vector of new candidate values, which could potentially be added to the cell state.

Architecture of LSTM

2) The Forget Gate:

- The information that is no longer useful in the cell state is removed with the forget gate. Two inputs xt (input at the particular time) and ht-1 (previous cell output) are fed to the gate and multiplied with weight matrices followed by the addition of bias. The resultant is passed through an activation function which gives a binary output. If for a particular cell state the output is 0, the piece of information is forgotten and for output 1, the information is retained for future use.



Source

Architecture of LSTM

2) The Forget Gate:

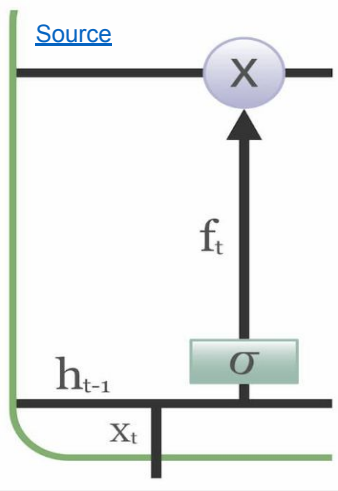
- The equation : $f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$

where:

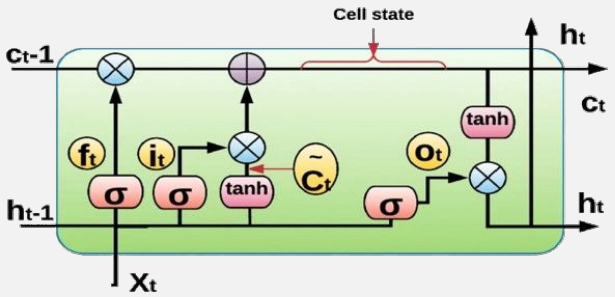
- W_f represents the weight matrix associated with the forget gate.
- [h_t-1, x_t] denotes the concatenation of the current

input and the previous hidden state.

- b_f is the bias with the forget gate.
- σ is the sigmoid activation function.



Architecture of LSTM



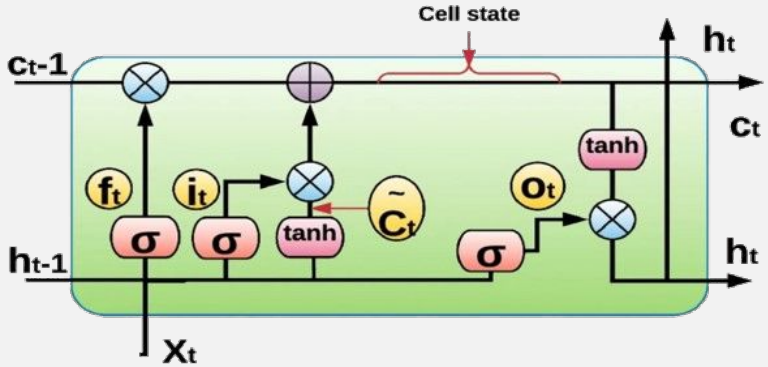
2) The Forget Gate:

- The forget gate determines what information to remove from the memory cell.
- It inputs the current input and the previous hidden state, setting a value between 0 and 1 for each element.
- A value of 0 indicates that the information will be discarded, whereas a value of 1 means it will be kept.
- Through these mechanisms, LSTM networks manage information, store, update, and access data across long sequences.
- This selective memory capability is particularly beneficial for applications requiring the modelling of long-term dependencies, such as language translation, speech recognition, and sentiment analysis.

Architecture of LSTM

2) The Forget Gate:

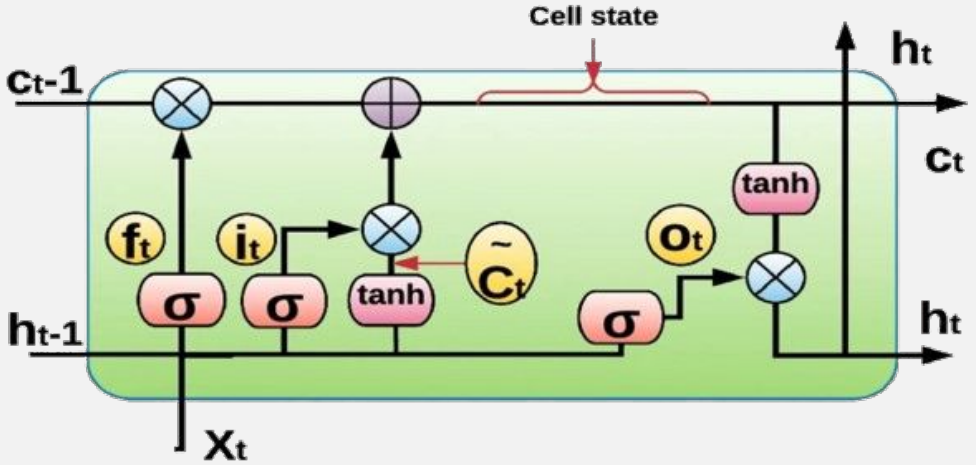
- This gate is responsible for deciding which information to remove from the cell state.
- It evaluates inputs from the current time step and the previous hidden state, generating a value ranging from 0 to 1 for each number in the cell state.
- Here, a value of 1 signifies "fully retain it" whereas a value of 0 means "completely discard it"



Architecture of LSTM

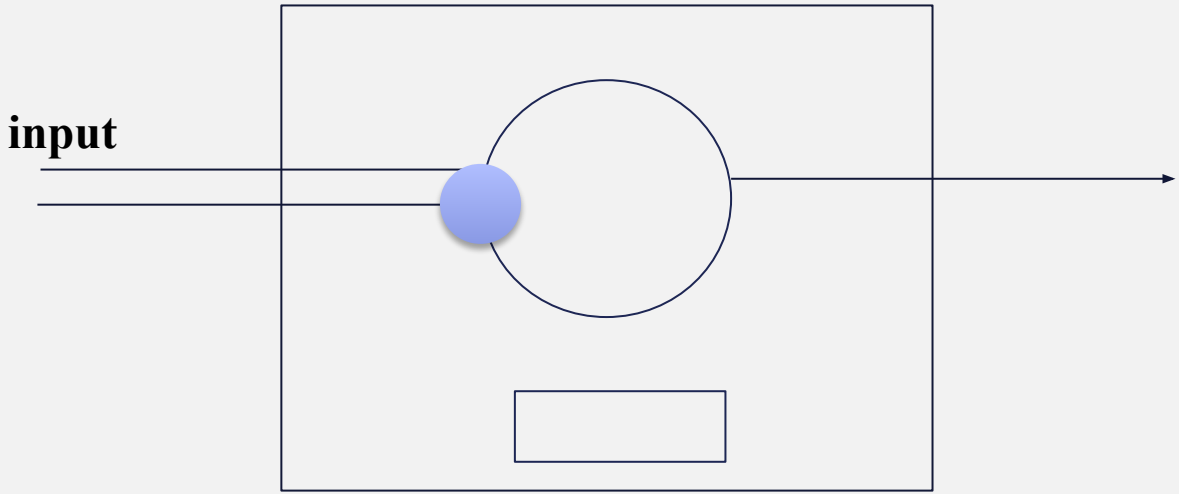
2) The Output Gate:

- Decides the subsequent hidden state using the updated cell state.
- It filters and selects the information that the LSTM will ultimately output, guided by the modifications in the cell state.



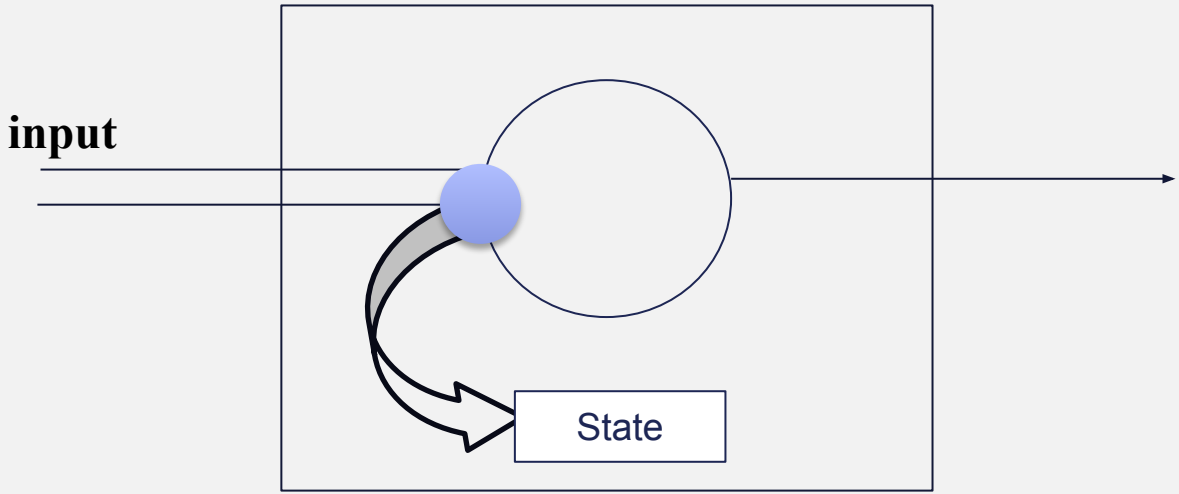
Activation function on Gates

1) Forget Act function:



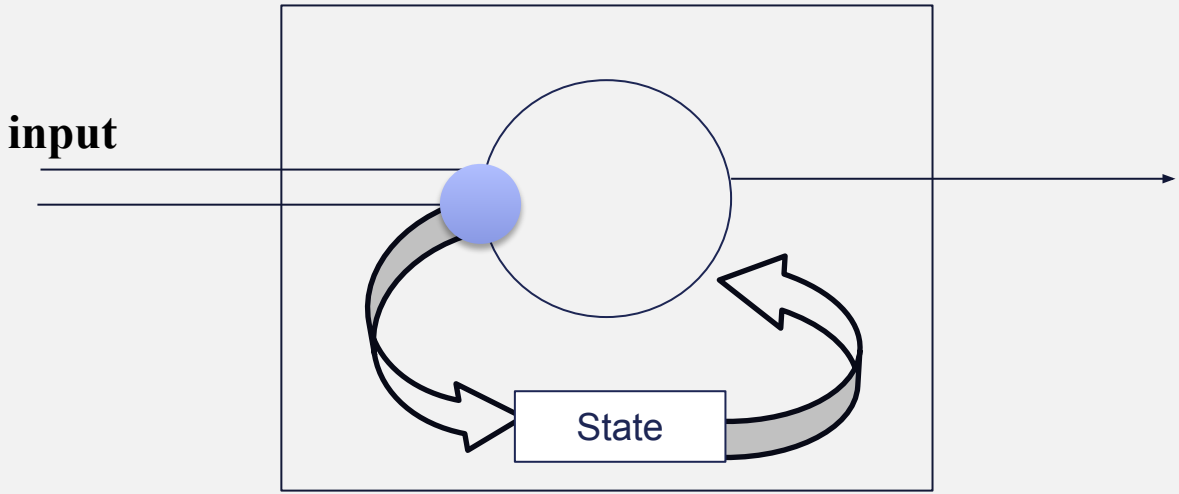
Activation function on Gates

1) Forget Act function:



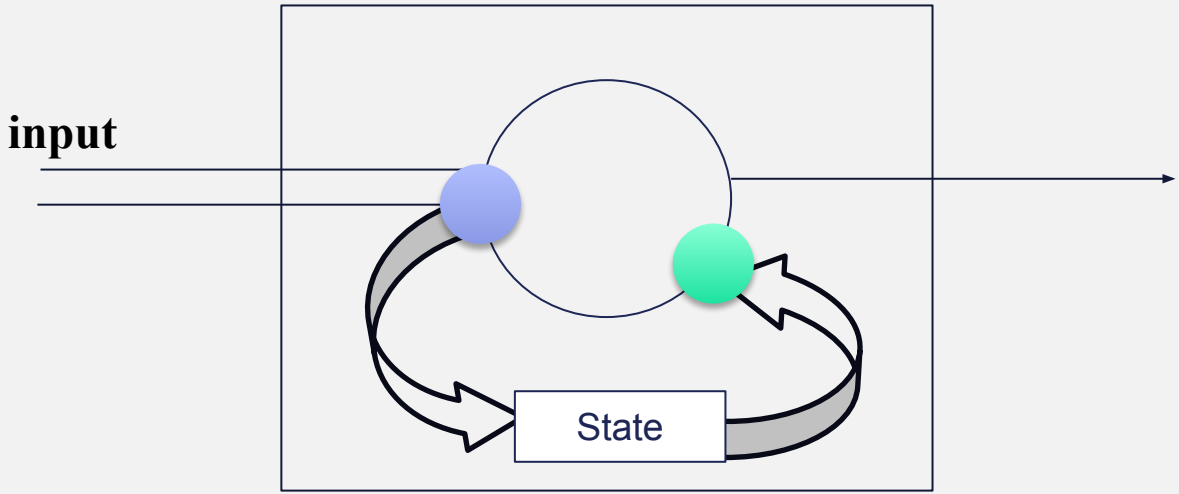
Activation function on Gates

1) Forget Act function:



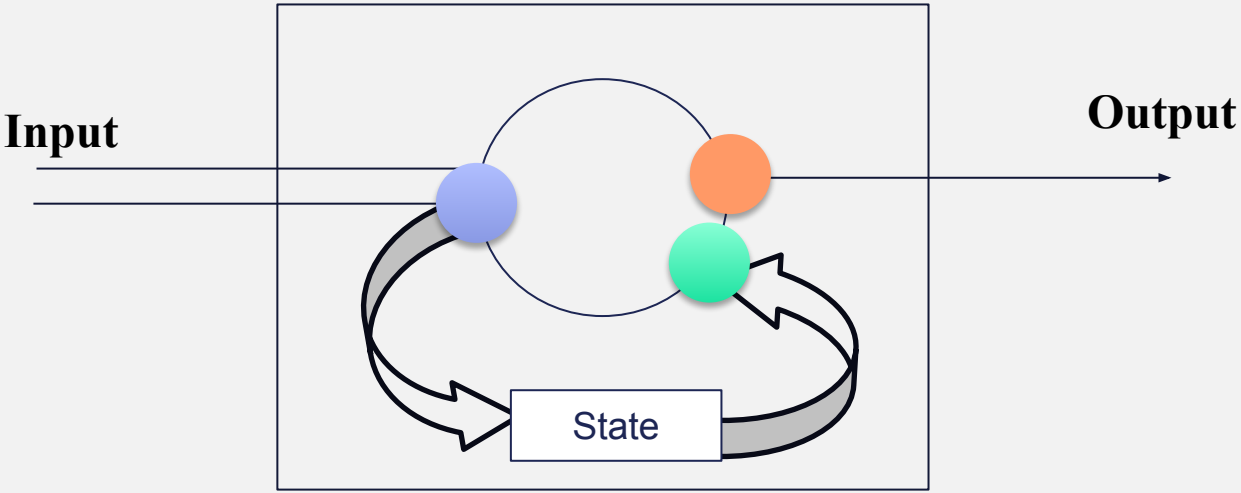
Activation function on Gates

1) Input Act function:



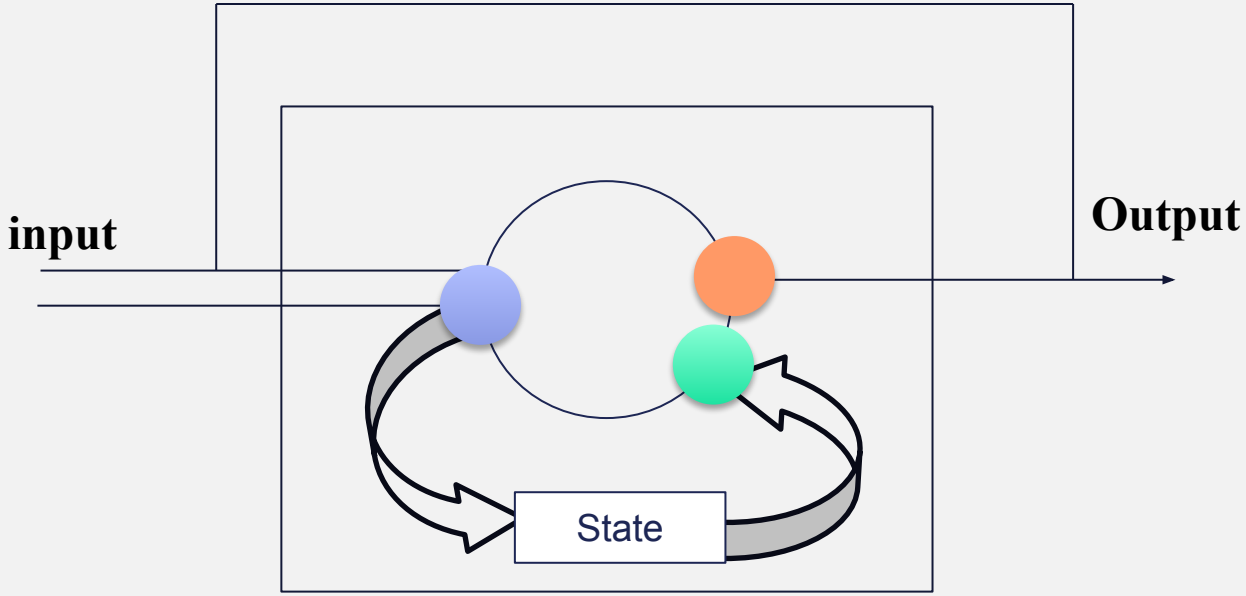
Activation function on Gates

1) Input Act function:



Activation function on Gates

1) Input Act function:



Key Components

Key Components of LSTM:

Cell State: flows throughout the entire LSTM chain, experiencing only slight linear interactions. It serves as the fundamental distinguishing feature of LSTMs, enabling them to manage and sustain long-term dependencies effectively.

Hidden State: represents the output of the LSTM at any given time step, determined by the current state of the cell.

Application of LSTM

Applications of LSTM

- Language Modeling: LSTMs have been used for natural language processing tasks such as language modeling, machine translation, and text summarization. They can be trained to generate coherent and grammatically correct sentences by learning the dependencies between words in a sentence.

- Speech Recognition: LSTMs have been used for speech recognition tasks such as transcribing speech to text and recognizing spoken commands. They can be trained to recognize patterns in speech and match them to the corresponding text.

Applications of LSTM

- Time Series Forecasting: LSTMs have been used for time series forecasting tasks such as predicting stock prices, weather, and energy consumption. They can learn patterns in time series data and use them to make predictions about future events.

- Anomaly Detection: LSTMs have been used for anomaly detection tasks such as detecting fraud and network intrusion. They can be trained to identify patterns in data that deviate from the norm and flag them as potential anomalies.

Applications of LSTM

- Recommender Systems: LSTMs have been used for recommendation tasks such as recommending movies, music, and books. They can learn patterns in user behavior and use them to make personalized recommendations.

- Video Analysis: LSTMs have been used for video analysis tasks such as object detection, activity recognition, and action classification. They can be used in combination with other neural network architectures, such as Convolutional Neural Networks (CNNs), to analyze video data and extract useful information.

Advantage over standard RNN’s

Advantage over standard RNN's

LSTM cells have several advantages over simple RNN cells, such as their ability to learn long-term dependencies and capture complex patterns in sequential data. For example, they can predict the next word in a sentence based on the previous words and the context, or generate captions for images based on the visual features and the language model. Additionally, LSTM cells can avoid the vanishing or exploding gradient problem, allowing them to learn from longer sequences without losing or amplifying the information. This means they can be used to translate a long sentence from one language to another without forgetting or distorting the meaning. Furthermore, they can handle noisy or missing data better than simple RNN cells, such as filling in the blanks or correcting errors in a text based on the surrounding words and grammar.

RNN vs. LSTM

Feature	LSTM (Long Short-term Memory)	RNN (Recurrent Neural Network)
Memory	Has a special memory unit that allows it to learn long-term dependencies in sequential data	Does not have a memory unit
Directionality	Can be trained to process sequential data in both forward and backward directions	Can only be trained to process sequential data in one direction
Training	More difficult to train than RNN due to the complexity of the gates and memory unit	Easier to train than LSTM
Long-term dependency learning	Yes	Limited

Feature	LSTM (Long Short-term Memory)	RNN (Recurrent Neural Network)
Ability to learn sequential data	Yes	Yes
Applications	Machine translation, speech recognition, text summarization, natural language processing, time series forecasting	Natural language processing, machine translation, speech recognition, image processing, video processing

Let’s Practice

Tutorial:

5- Time series forecasting/2-2- Time Series Forecasting (LSTM and GRU)

/LAB/LSTM_Tutorial.ipynb

Exercise:

5- Time series forecasting/2-2- Time Series Forecasting (LSTM and GRU)

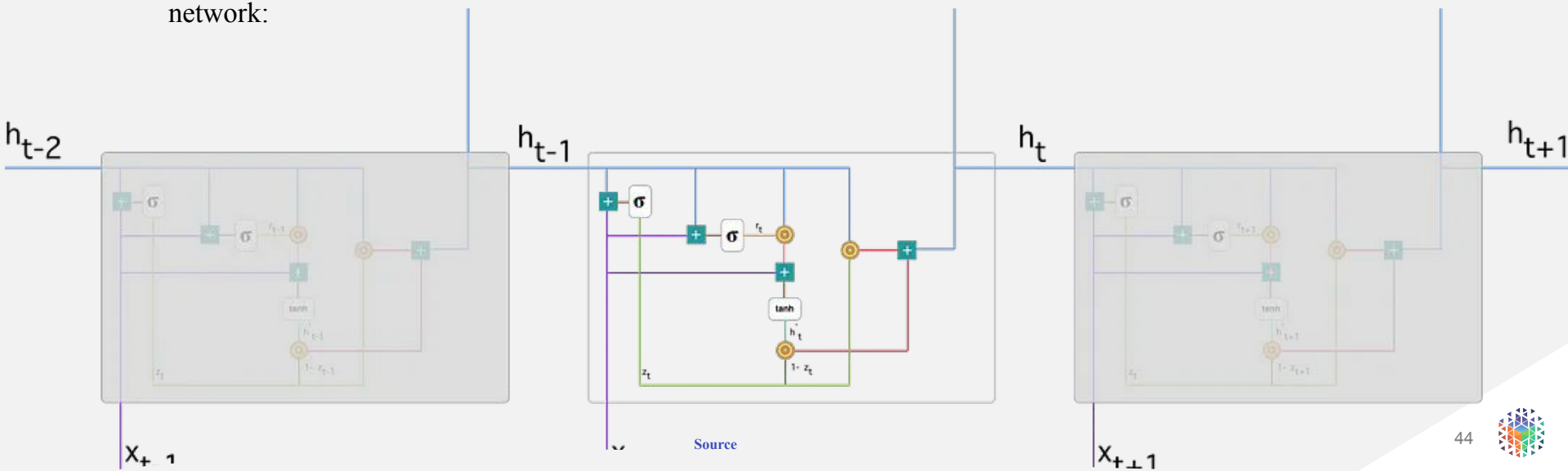
/LAB/LSTM_Exercise.ipynb

Introduction to GRUs.

GRUs

- **GRUs are improved version of standard recurrent neural network. Aims to solve the vanishing gradient problem** which comes with a standard recurrent neural network **.**
- **It's similar as LSTM because both are designed to produced equally excellent results.**
- To solve the vanishing gradient problem of a standard RNN, GRU uses, so-called, **update gate and reset gate.**

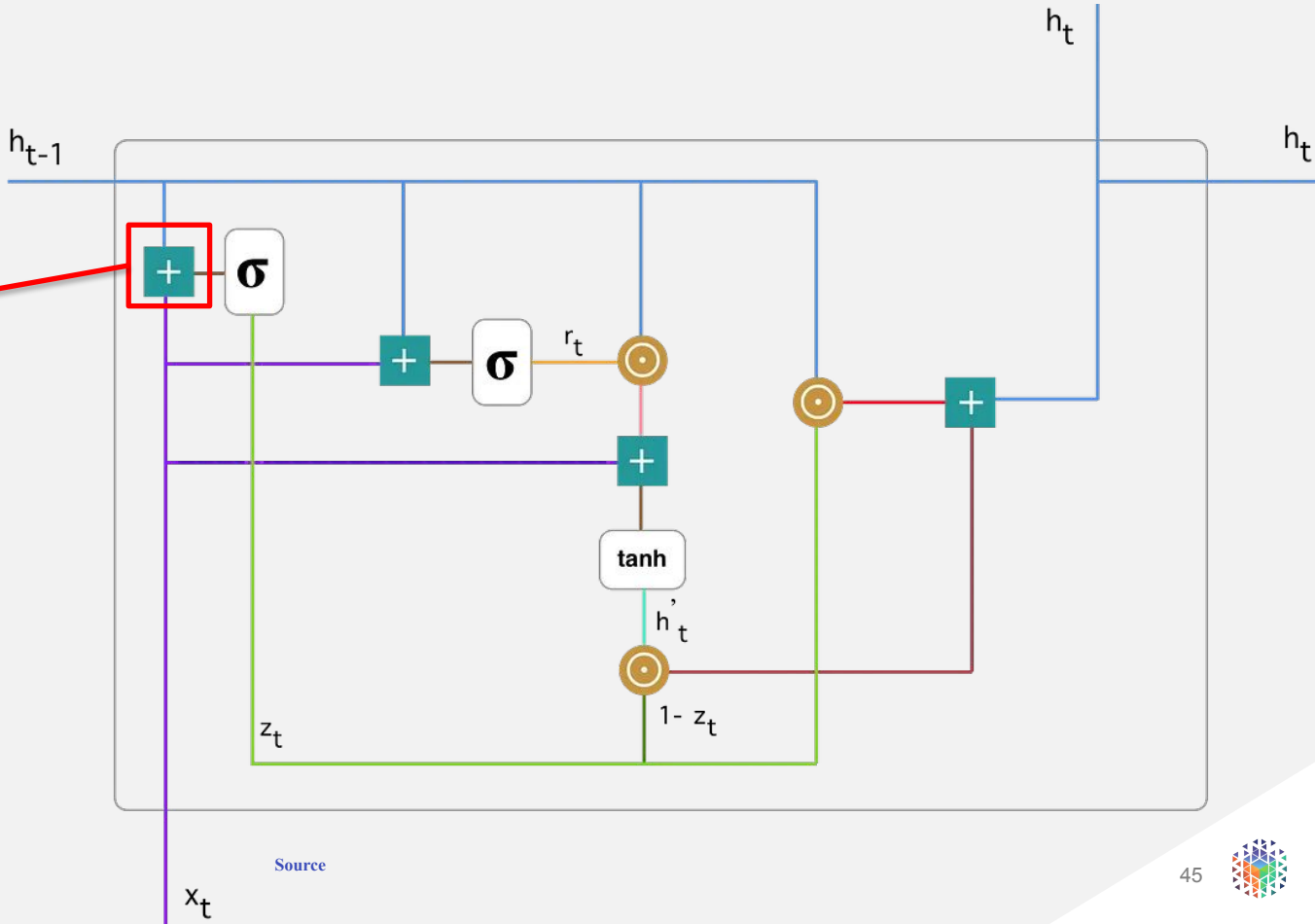
To explain the mathematics behind that process we will examine a single unit from the following recurrent neural network:



Source

GRUs

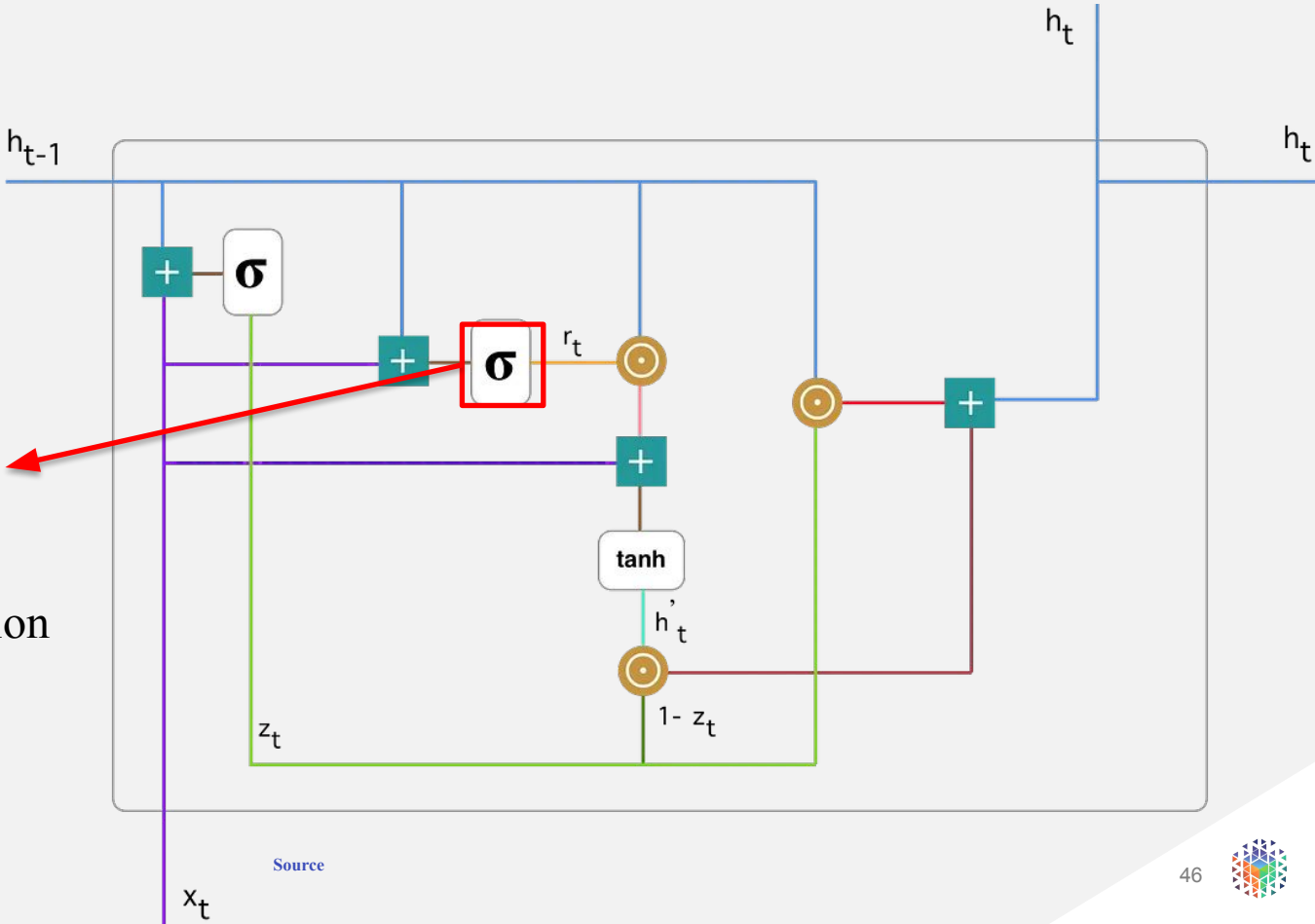
‘Plus’ Operation



Source

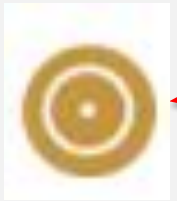
GRUs

‘Sigmoid’ Function

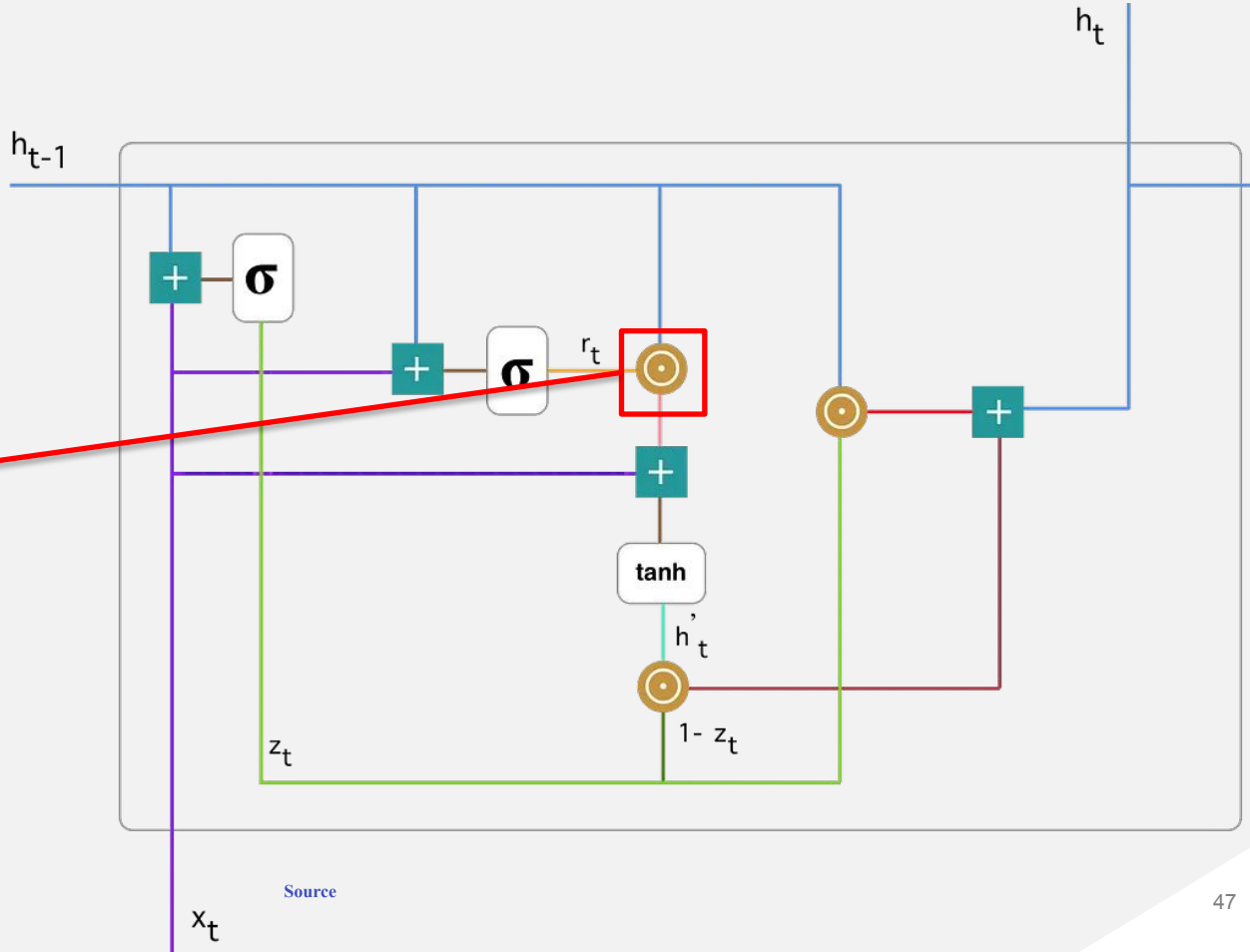


Source

GRUs

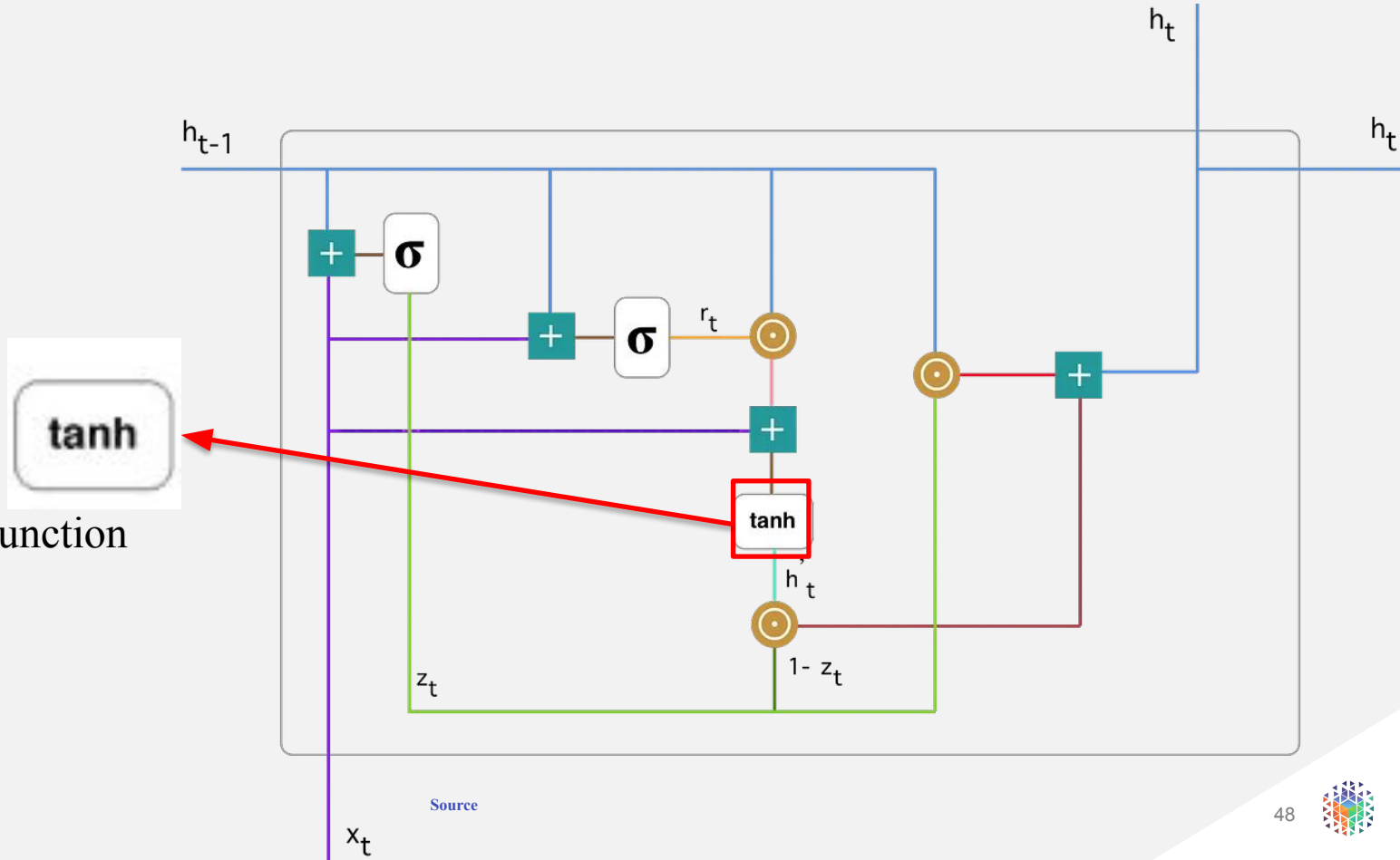


‘Hadamard product’ Operation



Source

GRUs



'Tanh' Function

Source

GRU Architecture.

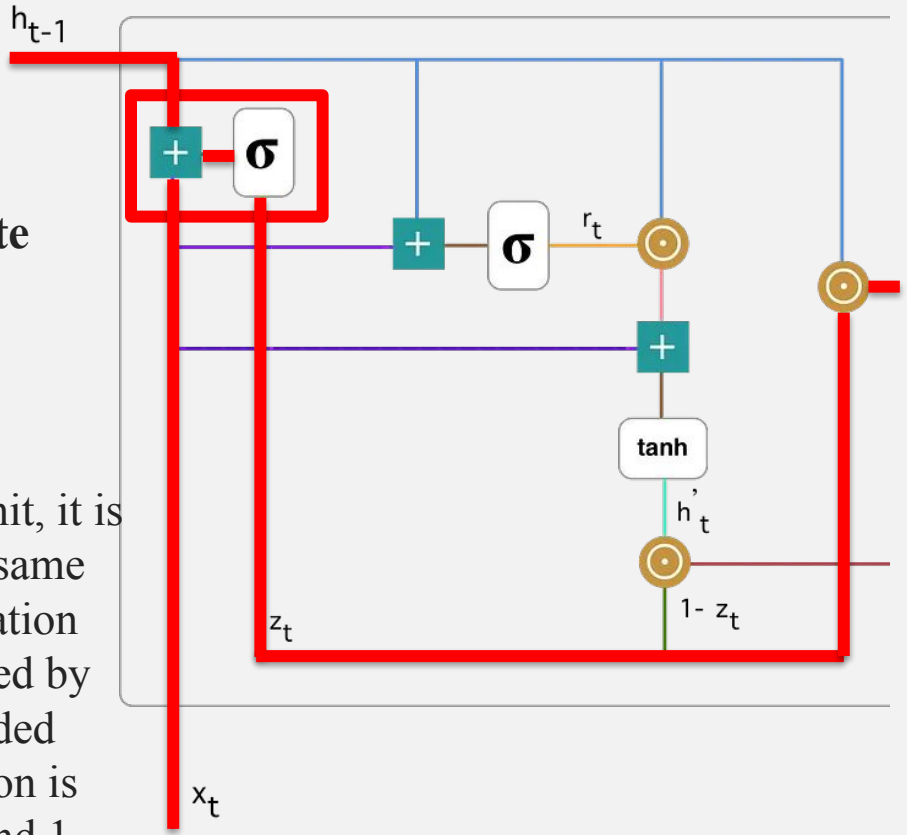
GRUs

Update gate

We start with calculating the **update gate z_t for time step t** using the formula:

$$z_t = \sigma(W^{(z)}x_t + U^{(z)}h_{t-1})$$

When *x_t* is plugged into the network unit, it is multiplied by its own weight *W(z)*. The same goes for *h_(t-1)* which holds the information for the previous *t-1* units and is multiplied by its own weight *U(z)*. Both results are added together and a sigmoid activation function is applied to squash the result between 0 and 1.

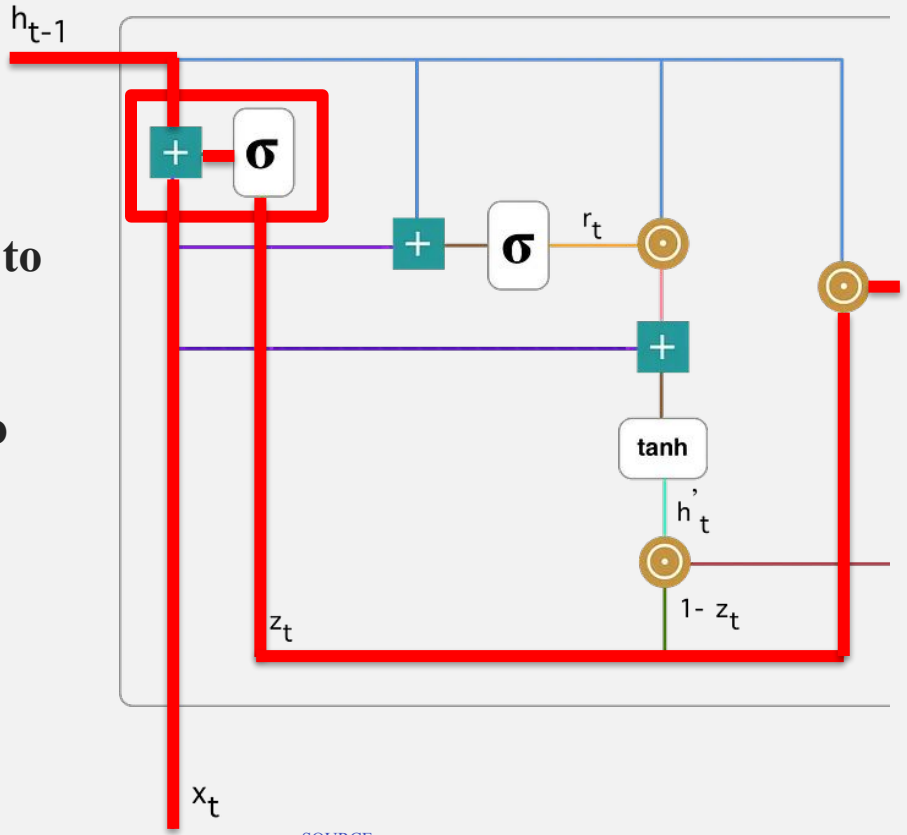


SOURCE

GRUs

Update gate

The update gate helps the model to determine how much of the past information (from previous time steps) needs to be passed along to the future.



SOURCE

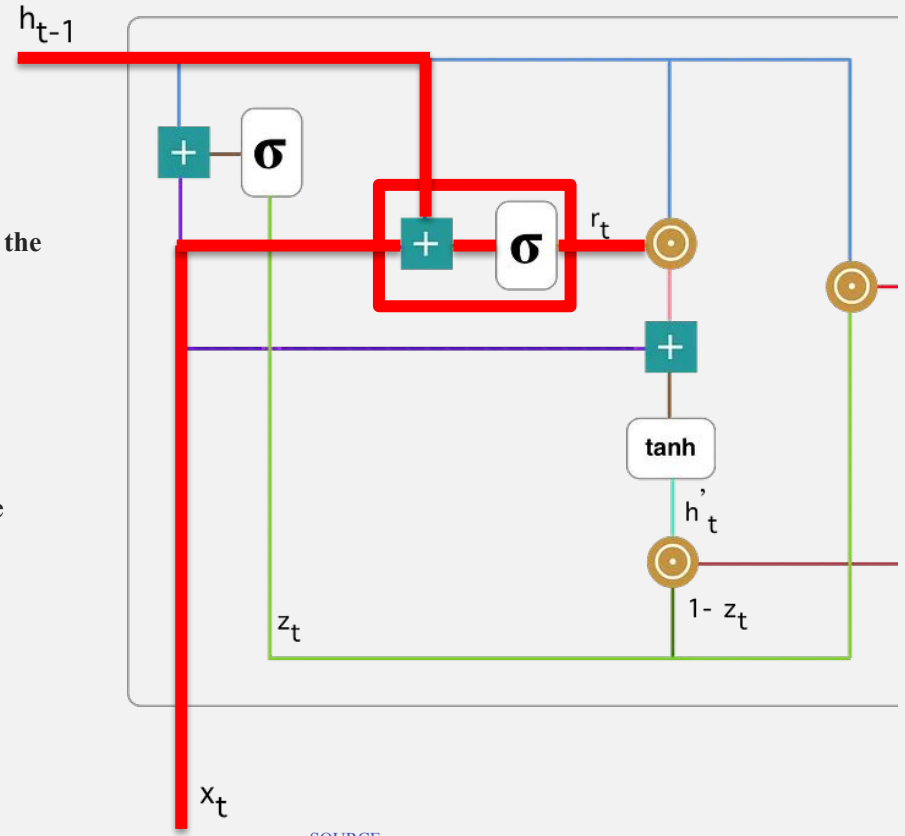
GRUs

Reset gate

This gate is used from the model to decide how much of the past information to forget.

$$r_t = \sigma(W^{(r)}x_t + U^{(r)}h_{t-1})$$

This formula is the same as the one for the update gate. The difference comes in the weights and the gate's usage



SOURCE

Current memory content

We introduce a new memory content which will use the reset gate to store the relevant information from the past.

It is calculated as follows:

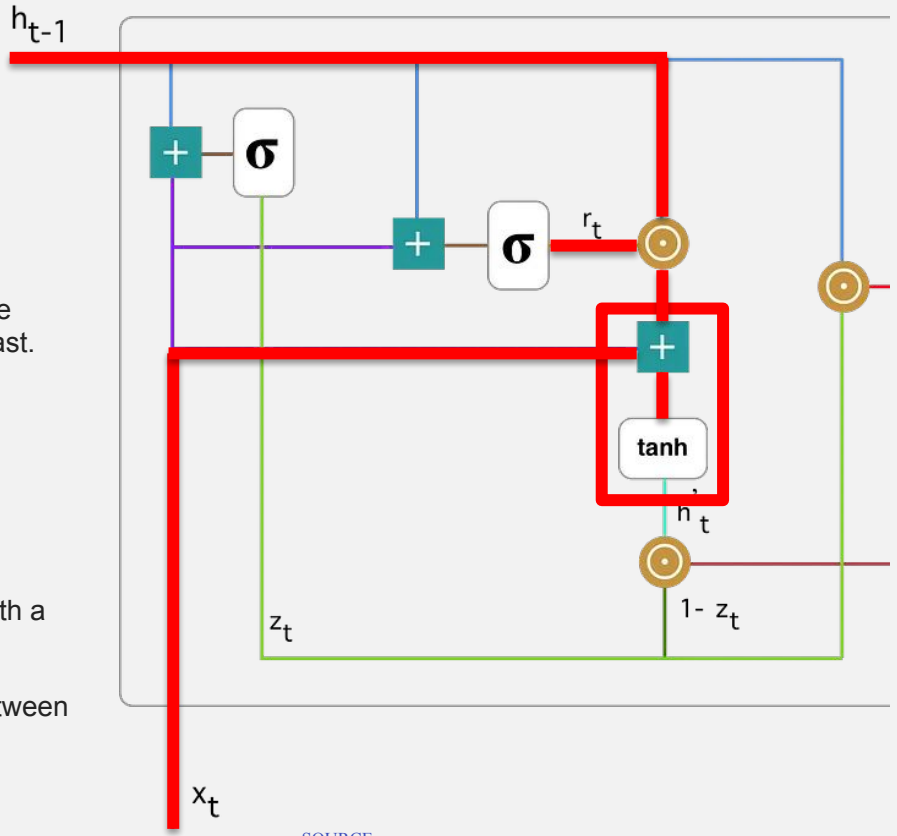
$$h_t' = \tanh(Wx_t + r_t \odot Uh_{t-1})$$

1- Multiply the input *x_t* with a weight *W* and *h_(t-1)* with a weight *U.*

2- Calculate the Hadamard (element-wise) product between the reset gate *r_t* and *Uh_(t-1).*

3- Sum up the results of step 1 and 2.

4- Apply the nonlinear activation function *tanh*.



SOURCE

GRUs

Final memory at current time step

It determines what to collect from the current memory content — *h'_t* and what from the previous steps — *h_(t-1)*.

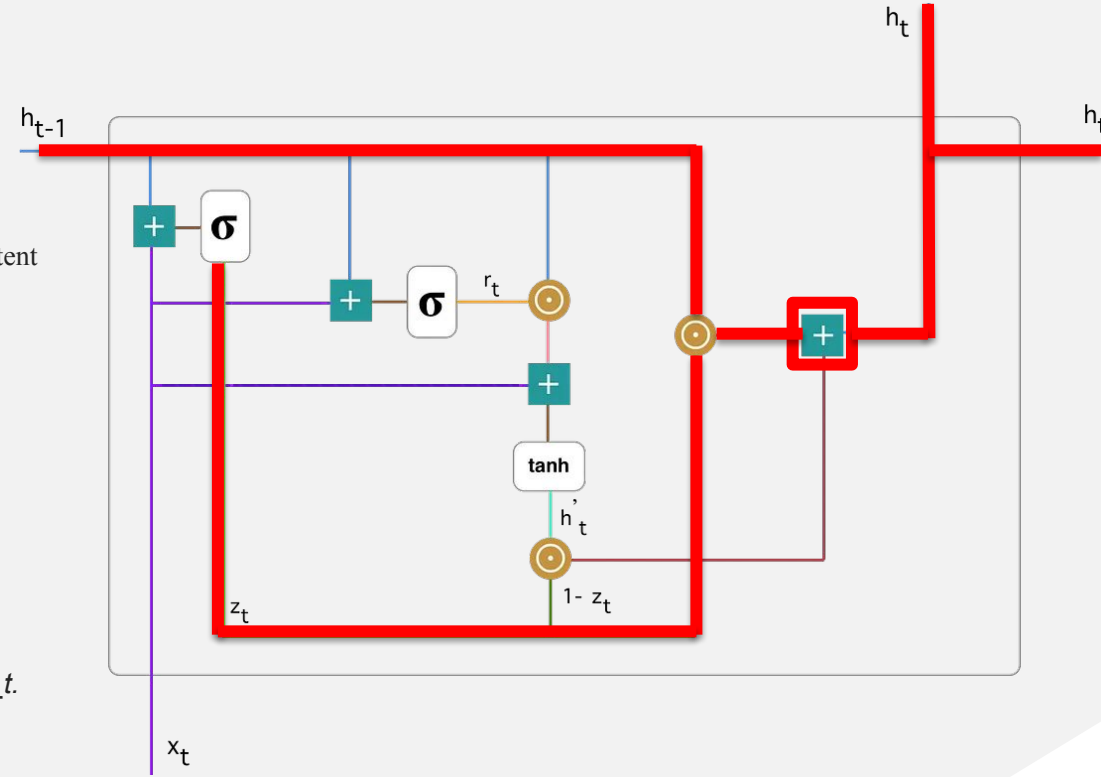
That is done as follows:

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot h_t^{'}$$

1- Apply element-wise multiplication to the update gate *z_t* and *h_(t-1)*.

2- Apply element-wise multiplication to *(1-z_t)* and *h'_t*.

3- Sum the results from step 1 and 2..



SOURCE

Applications and Performance Comparisons.

Applications and Performance Comparisons.

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

Language Modelling : 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.

Applications and Performance Comparisons.

Parameter	RNNs	LSTMs	GRU
Structure	Simple	More complex	Simpler than LSTM
Training	Can be difficult	Can be more difficult	Easier than LSTM
Performance	Good for simple tasks	Good for complex tasks	Can be intermediate between simple and complex tasks
Hidden state	Single	Multiple (memory cell)	Single
Gates	None	Input, output, forget	Update, reset
Ability to retain long-term dependencies	Limited	Strong	Intermediate between RNNs and LSTMs

Let’s Practice

Tutorial:

Traffic Prediction: GRU

Exercise:

5- Time series forecasting/2- Time Series Forecasting (LSTM and GRU)

/LAB/GRU_Exercise.ipynb

Let’s Practice

Tutorial:

5- Time series forecasting/2- Time Series Forecasting (LSTM and GRU)
/LAB/Time Series Forcasting using LSTM & GRU.ipynb

Dataset:

5- Time series forecasting/2- Time Series Forecasting (LSTM and GRU)
/LAB/Dataset/IBM_2006-01-01_to_2018-01-01.csv

Let's Practice

Exercise:

5- Time series forecasting/2- Time Series Forecasting (LSTM and GRU)
/LAB/Time Series Forcasting using LSTM & GRU_Exercise.ipynb

Dataset:

5- Time series forecasting/2- Time Series Forecasting (LSTM and GRU)
/LAB/Dataset/GOOGL_2006-01-01_to_2018-01-01.csv

THANK YOU

SDAIA
الهيئة السعودية للبيانات والذكاء الاصطناعي
Saudi Data & AI Authority