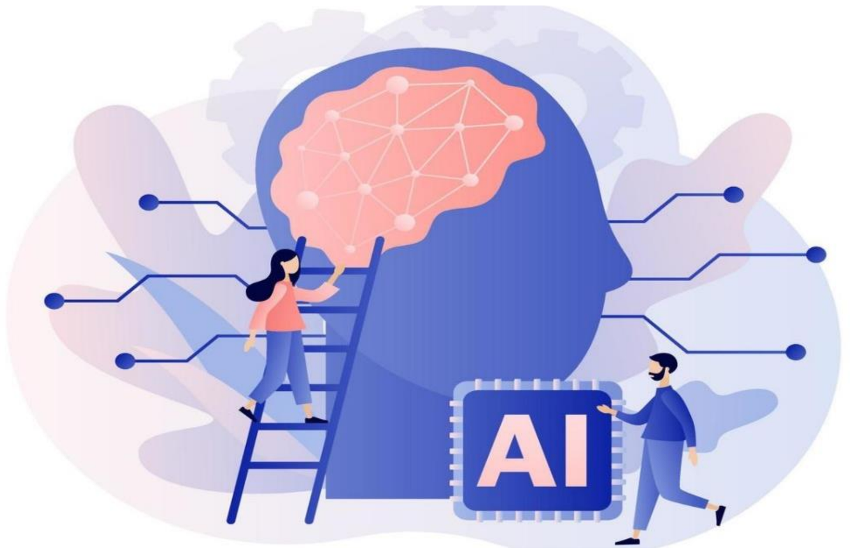
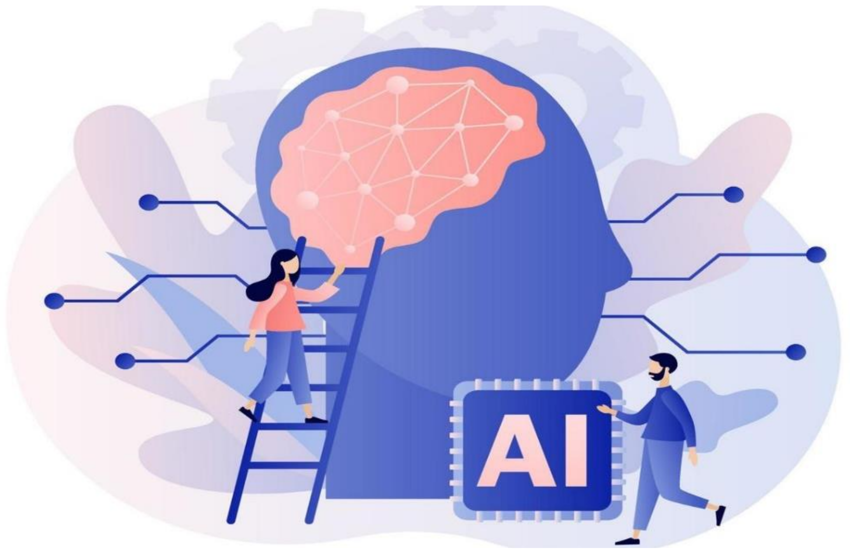


(RNN-LSTM-GRU)



PRESENTATION

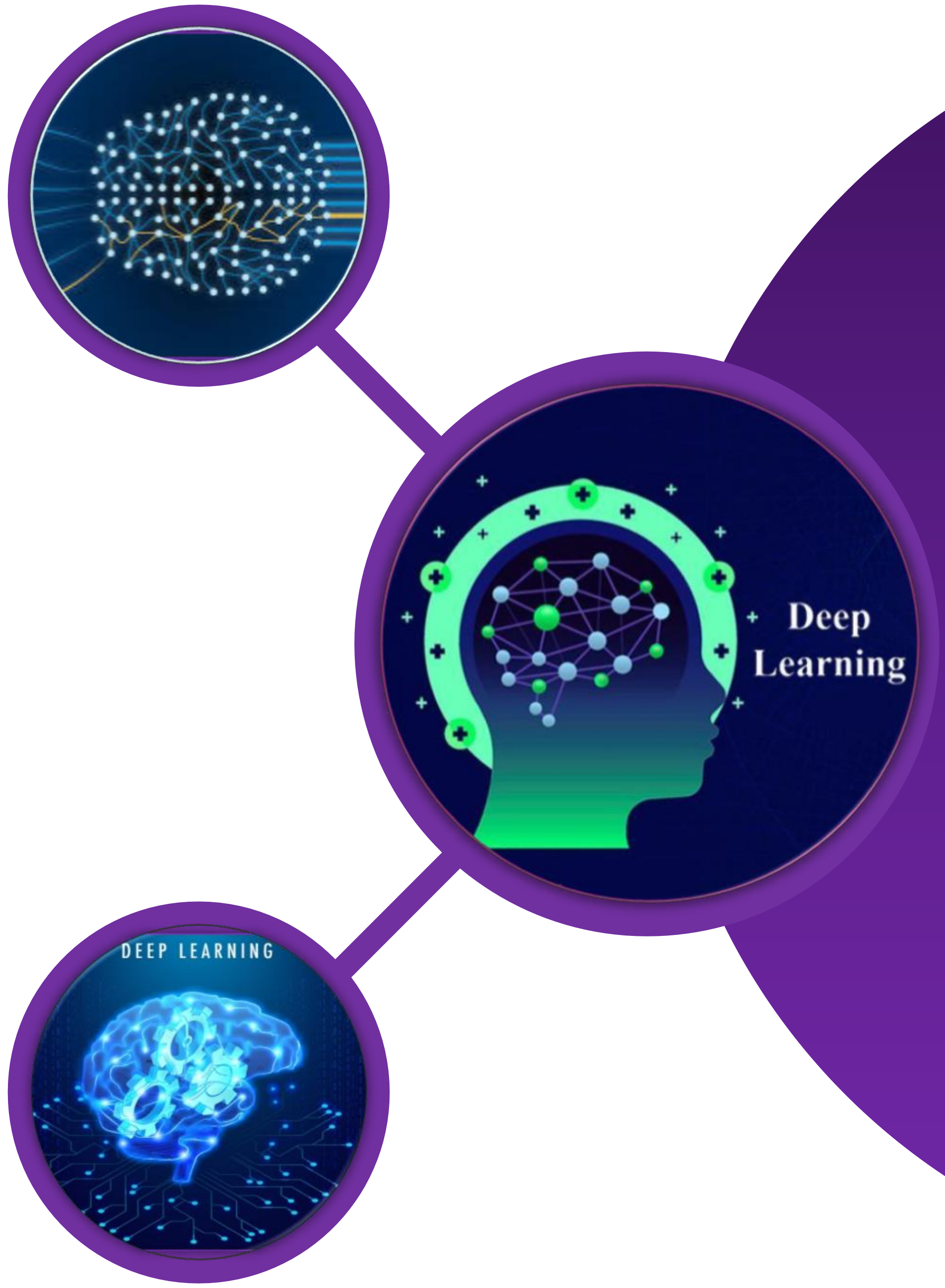
CONTENT LIST

01 RNN

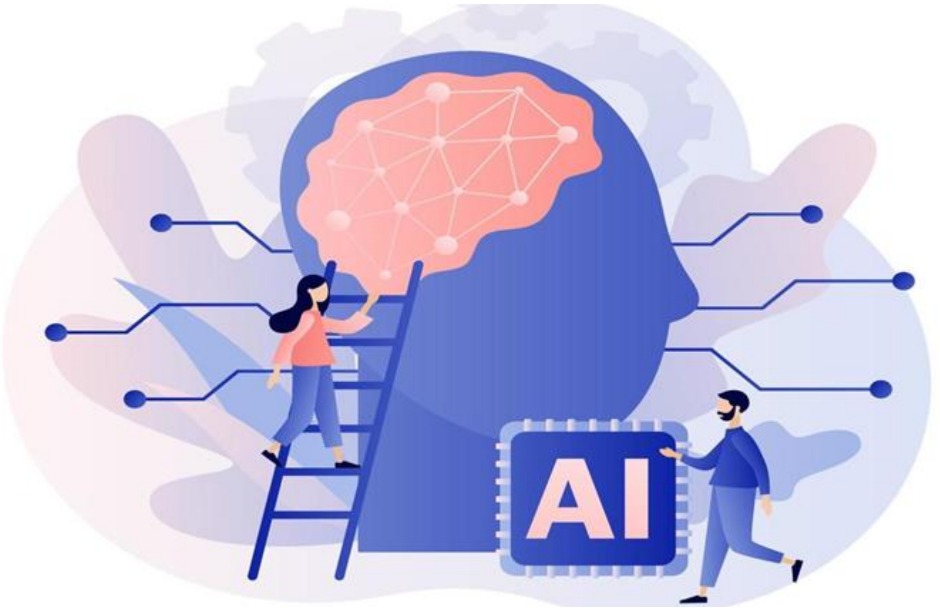
02 LSTM

03 GRU

04 RNN Topologies

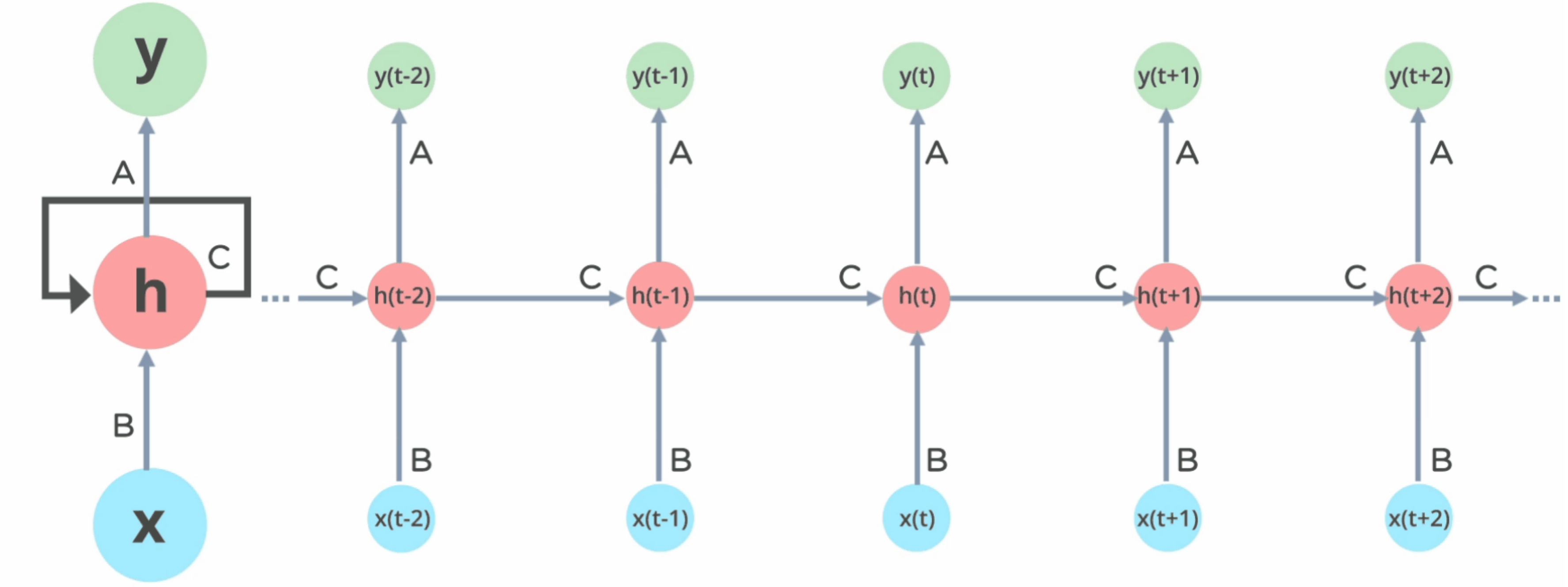


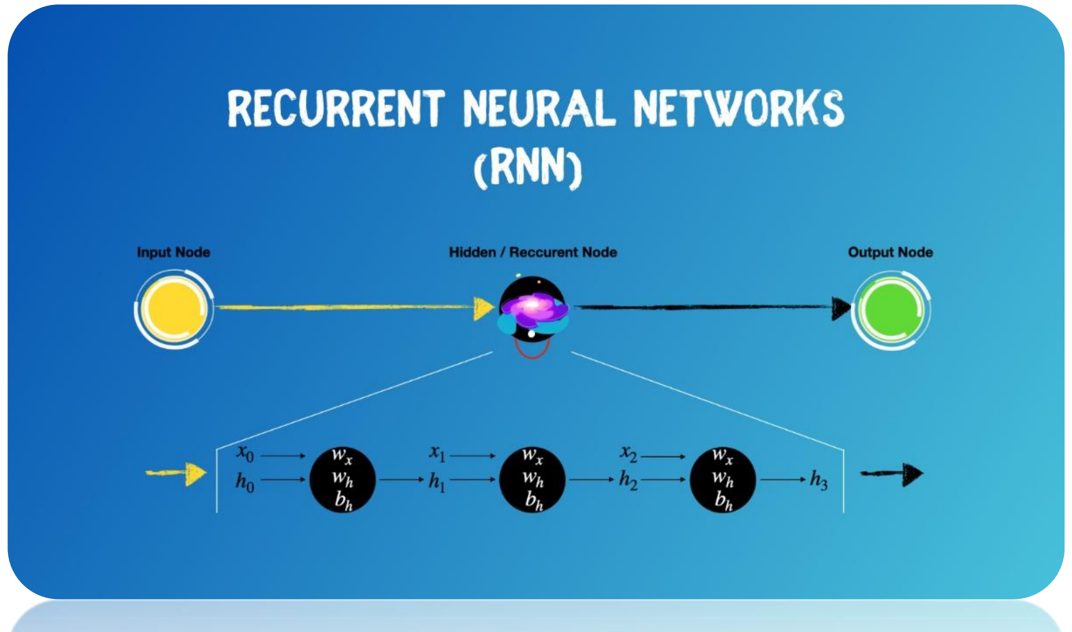
01 RNN

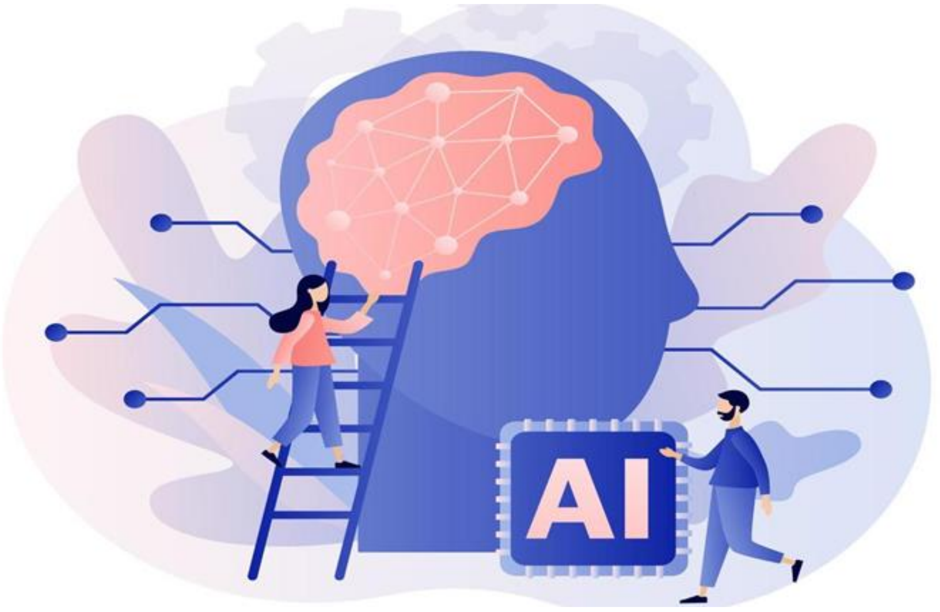


Recurrent Neural Networks (RNNs): are a family of networks that are suitable for learning representations of sequential data like text in Natural Language Processing (NLP).

- The idea behind RNNs is to make use of sequential information.
- In a traditional neural network, we assume that all inputs (and outputs) are independent of each other.
- But for many tasks that is a very bad idea. If we want to predict the next word in a sentence, we better know which words came before it.



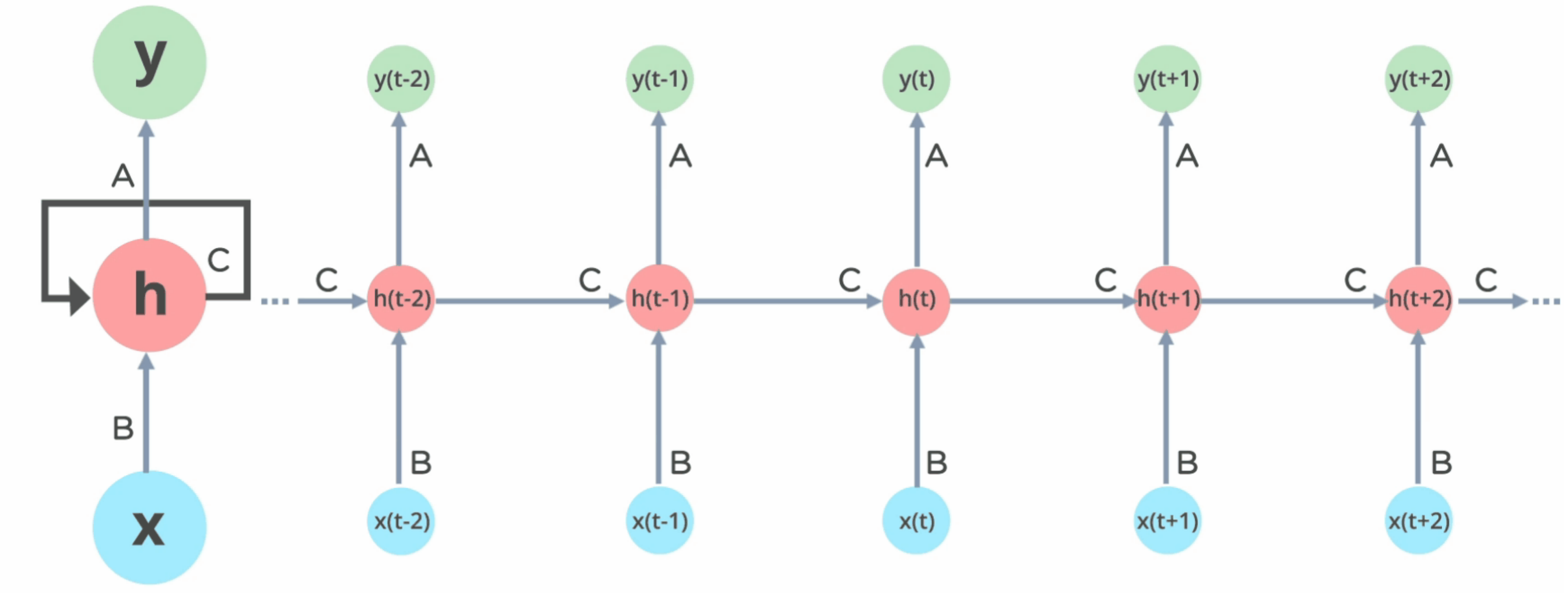


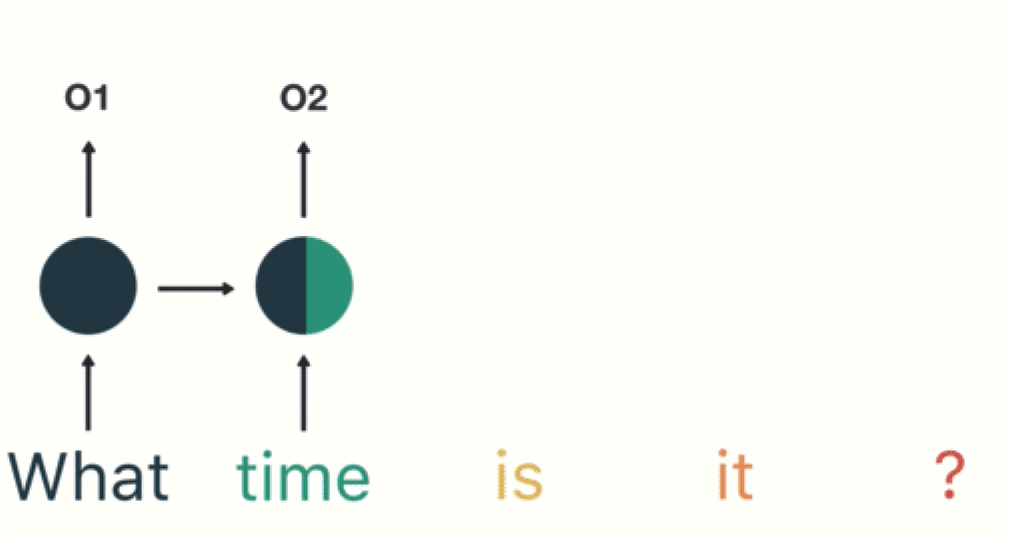


01 RNN

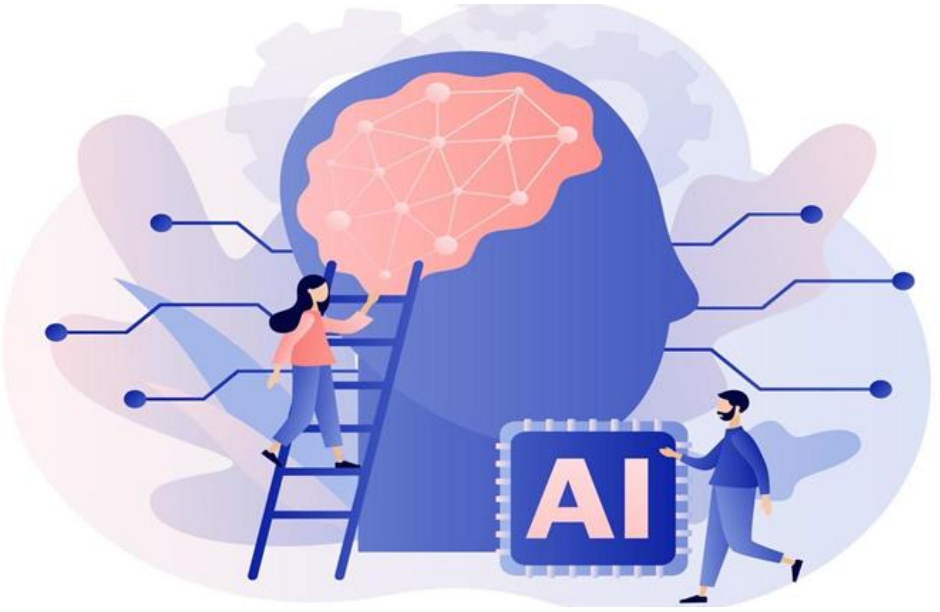
Recurrent Neural Networks (RNNs): are a family of networks that are suitable for learning representations of sequential data like text in Natural Language Processing (NLP).

- RNNs are called recurrent because they perform the same task for every element of a sequence, with the output being depended on the previous computations.
- Another way to think about RNNs is that they have a “memory” which captures information about what has been calculated so far.
- In theory RNNs can make use of information in arbitrarily long sequences, but in practice they are limited to looking back only a few steps.



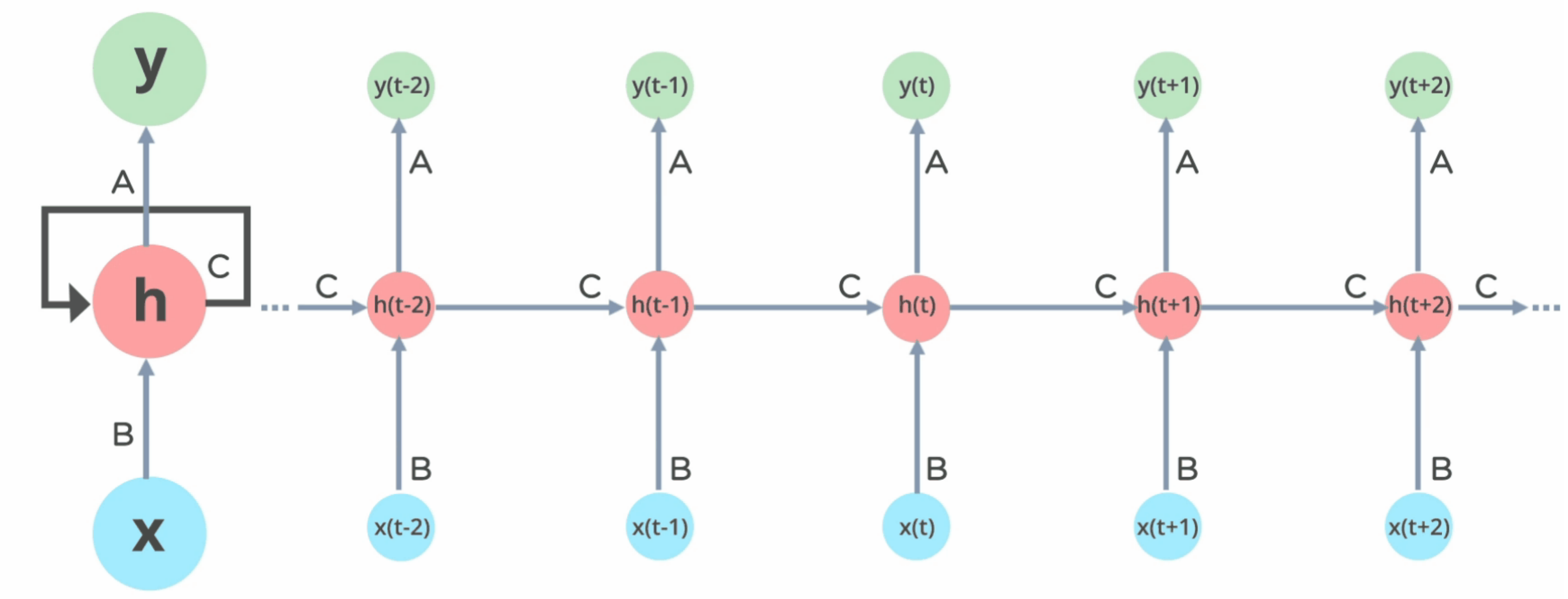


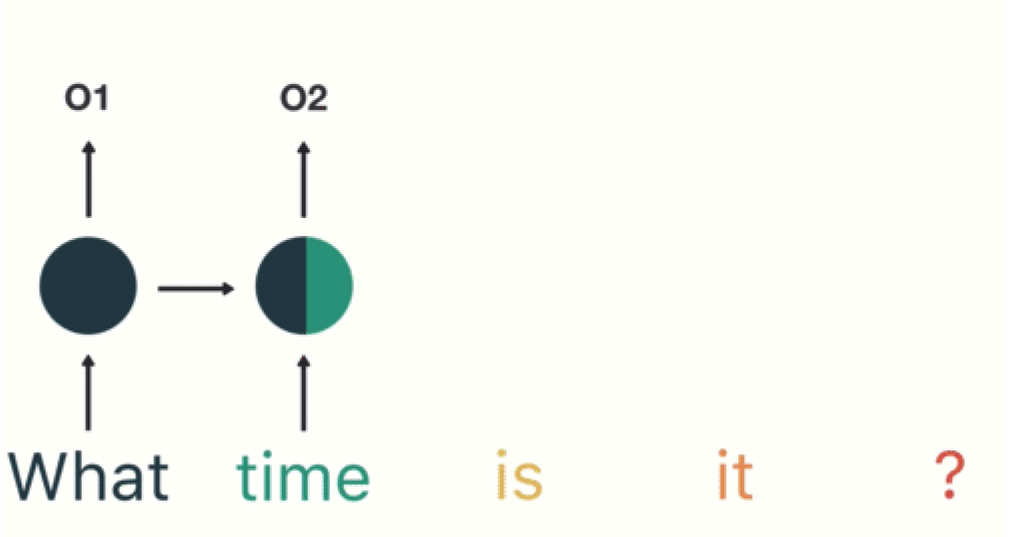
01 RNN



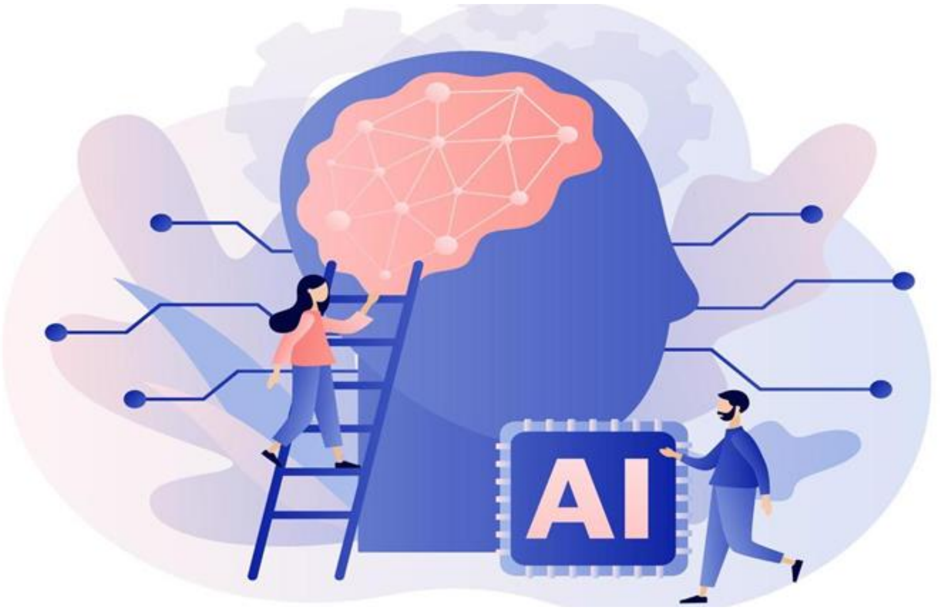
Recurrent Neural Networks (RNNs): are a family of networks that are suitable for learning representations of sequential data like text in Natural Language Processing (NLP).

- The above diagram shows a RNN being unrolled (or unfolded) into a full network.
- For example, if the sequence we care about is a sentence of 5 words, the network would be unrolled into a 5-layer neural network, one layer for each word.



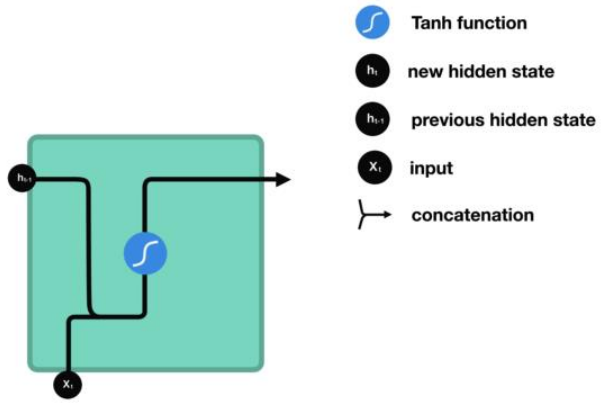


01 RNN

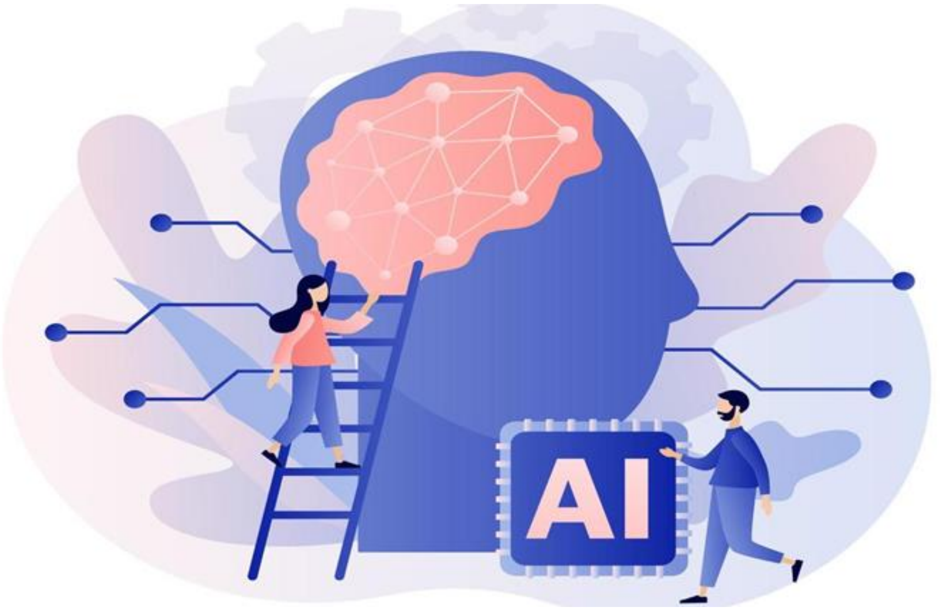


Recurrent Neural Networks (RNNs): are a family of networks that are suitable for learning representations of sequential data like text in Natural Language Processing (NLP).

- Time series data, such as stock prices, also exhibit a dependence on past data, called the secular trend.
- RNN cells incorporate this dependence by having a hidden state, or memory, that holds the essence of the past.
- The value of the hidden state at any point in time is a function of the value of the hidden state at the previous time step and the value of the input at the current time step, that is: $h_t = \tanh(W * h_{t-1} + U * X_t)$
- h_t and h_{t-1} are the values of the hidden states at the time steps t and t-1 respectively, and x_t is the value of the input at time t.
- The above equation is recursive, that is, h_{t-1} can be represented in terms of h_{t-2} and x_{t-1}, and so on, until the beginning of the sequence.



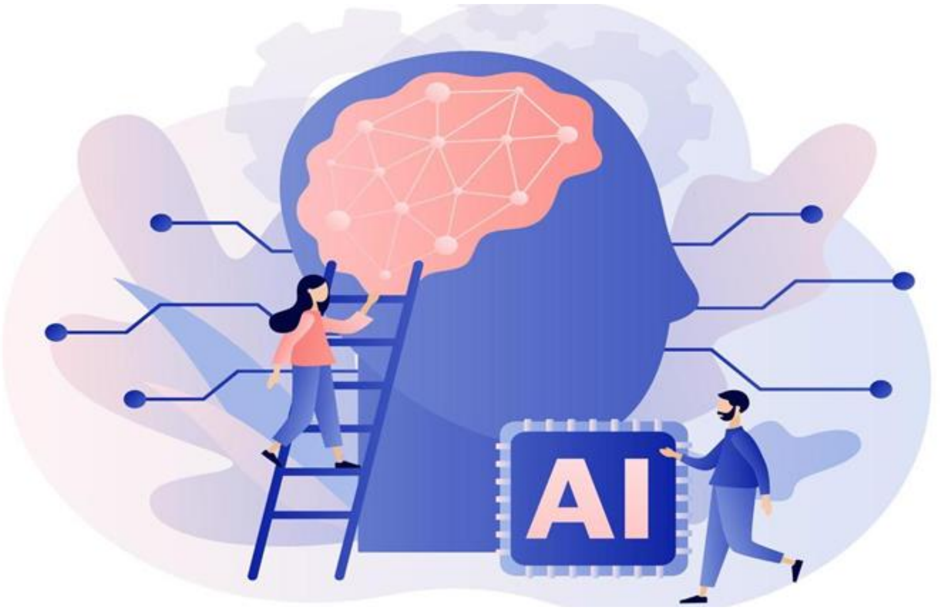
01 Vanishing gradient



When doing back propagation, each node in a layer calculates its gradient with respect to the effects of the gradients in the layer before it.

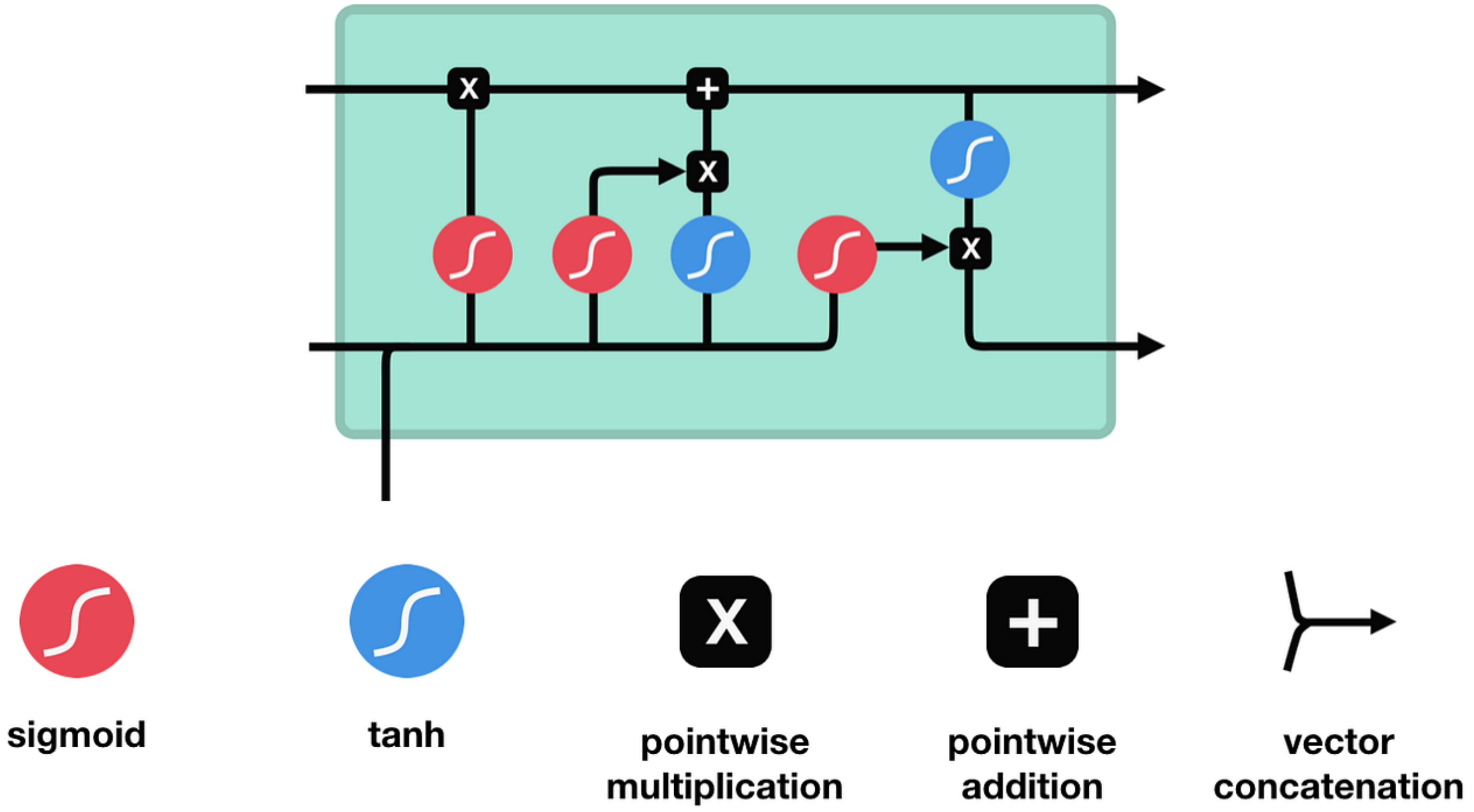
- Just like traditional neural networks, training the RNN also involves backpropagation.
- The difference in this case is that since the parameters are shared by all time steps, the gradient at each output depends not only on the current time step, but also on the previous ones.
- This process is called backpropagation through time (BPTT).
- Regular RNNs might have a difficulty in learning long range dependencies.
- This kind of dependencies between sequence data is called long-term dependencies because the distance between the relevant information and the point where it is needed to make a prediction is very wide.
- ***RNNs can't learn long-range dependencies!***

02 LSTM

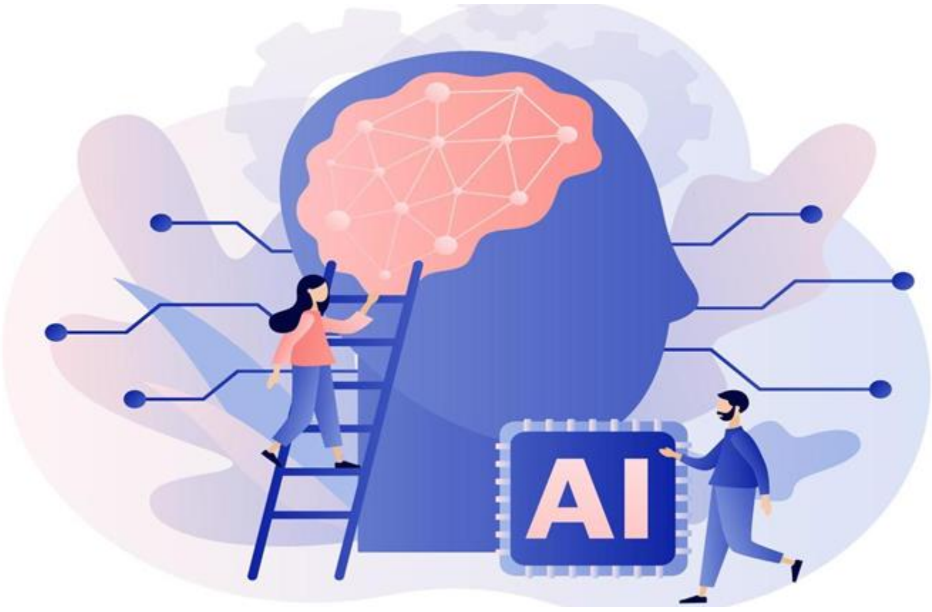


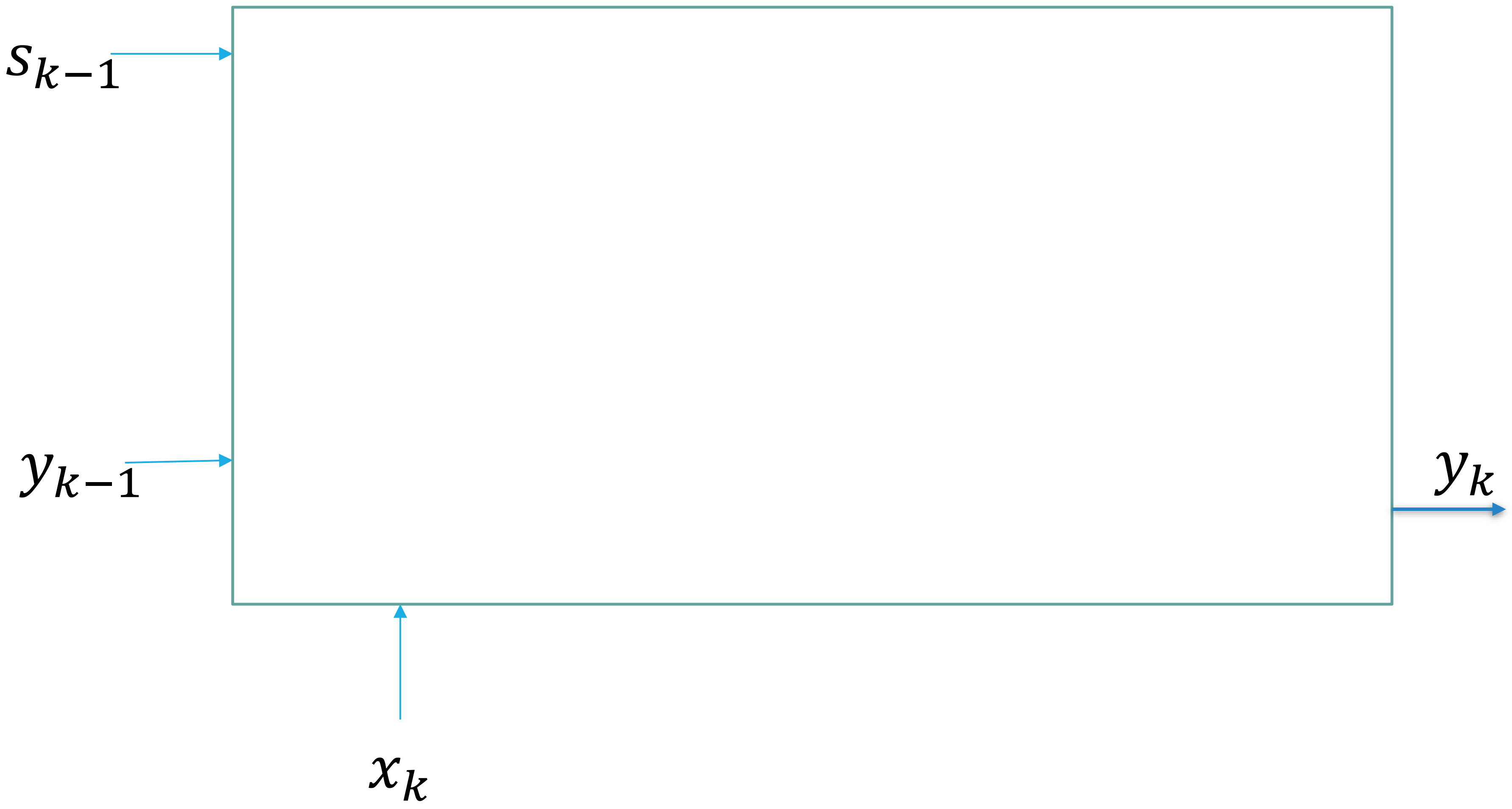
Similar in nature to RNNs but have additional features to help fight the short-term memory issue of RNNs.

While there are a few approaches to minimize the problem of vanishing gradients, such as proper initialization of the W matrix, using a ReLU instead of tanh layers, and pre-training the layers using unsupervised methods, the most popular solution is to use the LSTM or GRU architectures.



02 LSTM





انا دلوقتى عايز اخد قرار معين y_k

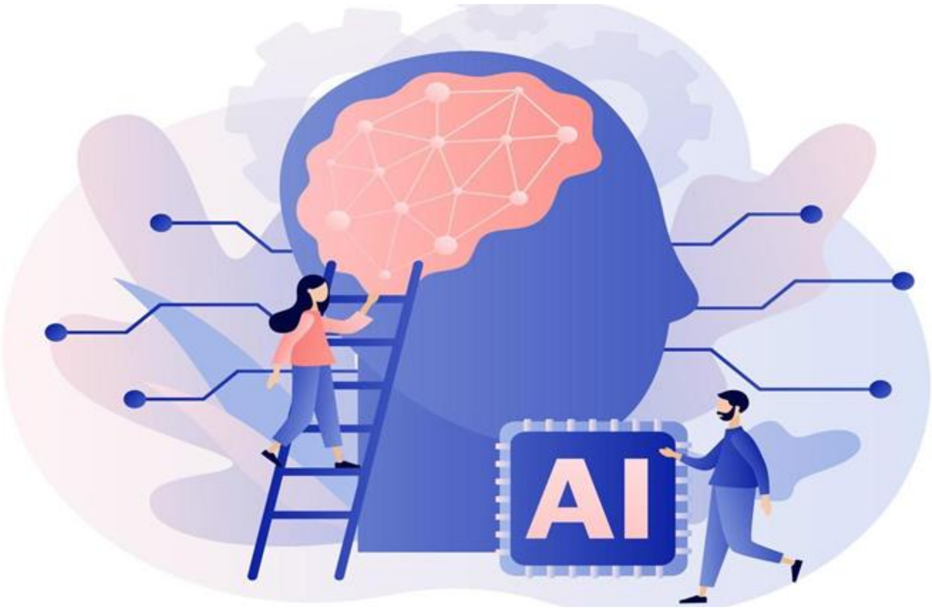
القرار ده هيعتمد على 3 حاجات:

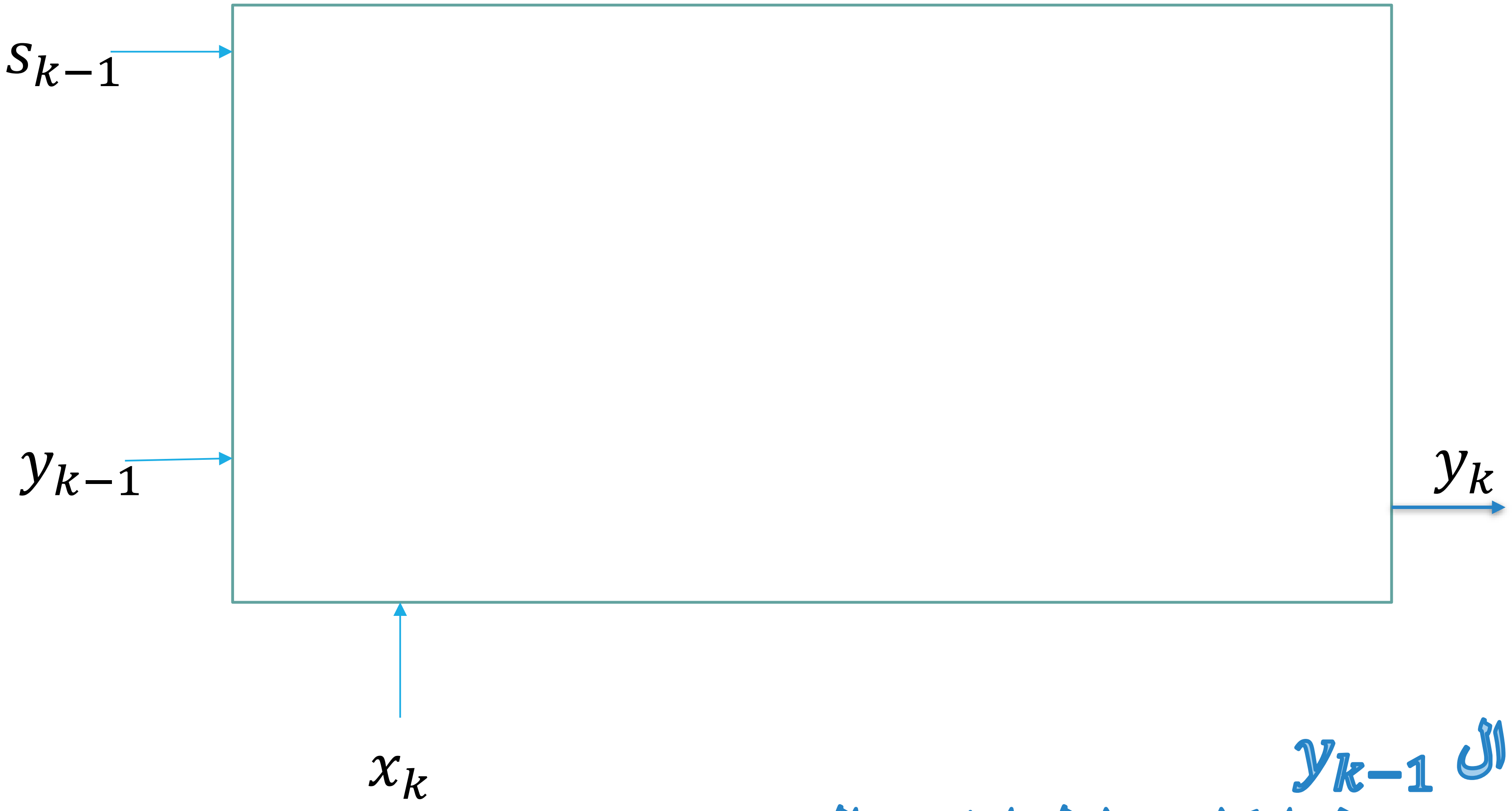
1- الانبوت اللى جايلى x_k اللى انا شايفه دلوقتى

2- اخر قرار انا اخدته y_{k-1}

3- وكل القرارات اللى اخدتها قبل كده ال history بتاعي يعني s_{k-1}

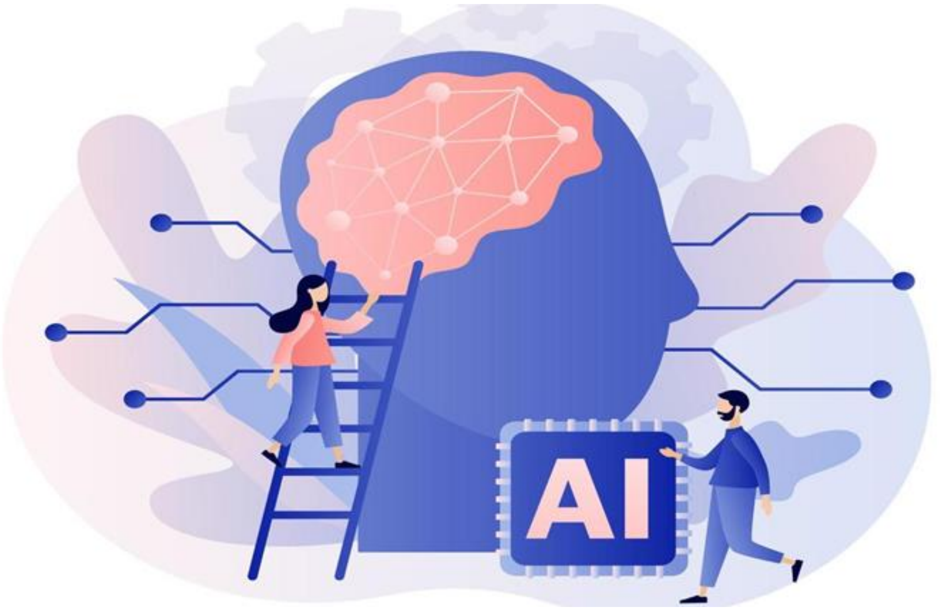
02 LSTM

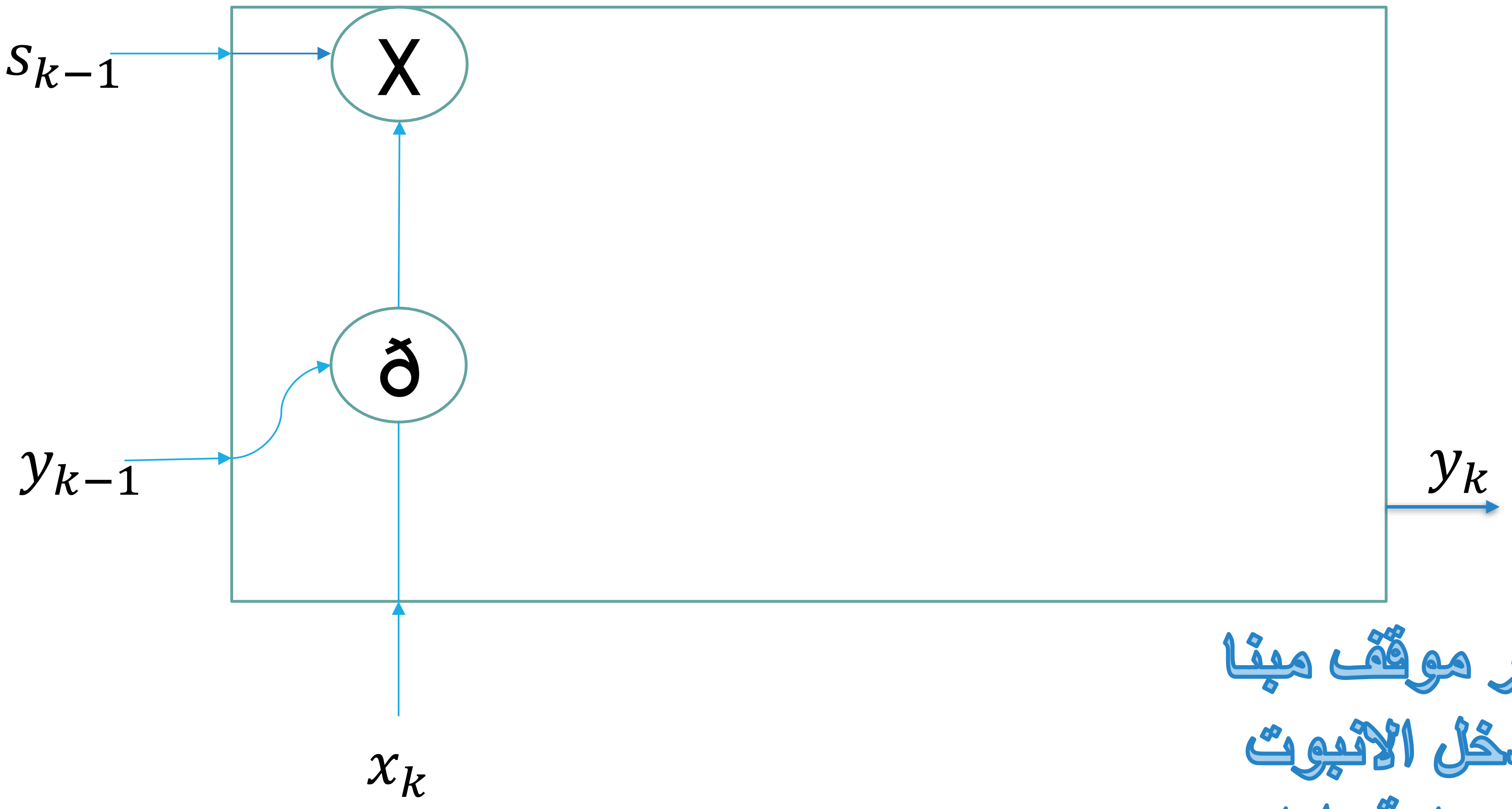




1- وليكن قولتلك فلان بيقول عليك كلام وحش يبقى ده ال x_k
2- فلان ده انا اخر موقف كان معاه كان ايه موقف رجوله ولا بيع ده ال y_{k-1}
3- طب تعالى بقى نرجع بالذاكره ونقلب في الماضى كل اللى فات مبينى ومبينوا كان عامل ايه ده ال s_{k-1}
يبقى هتربط الموقف اللى حصل دلوقتى باخر موقف حصل وبكل المواقف القديمه اللى مبنكوا من يوم معرفته
وعلى أساسه هاخد قرار اللى هو مثلا عفا الله عما سلف ولا اقطع علاقتي بيه ولا احذؤ من التعامل بس معاه ولا ايه

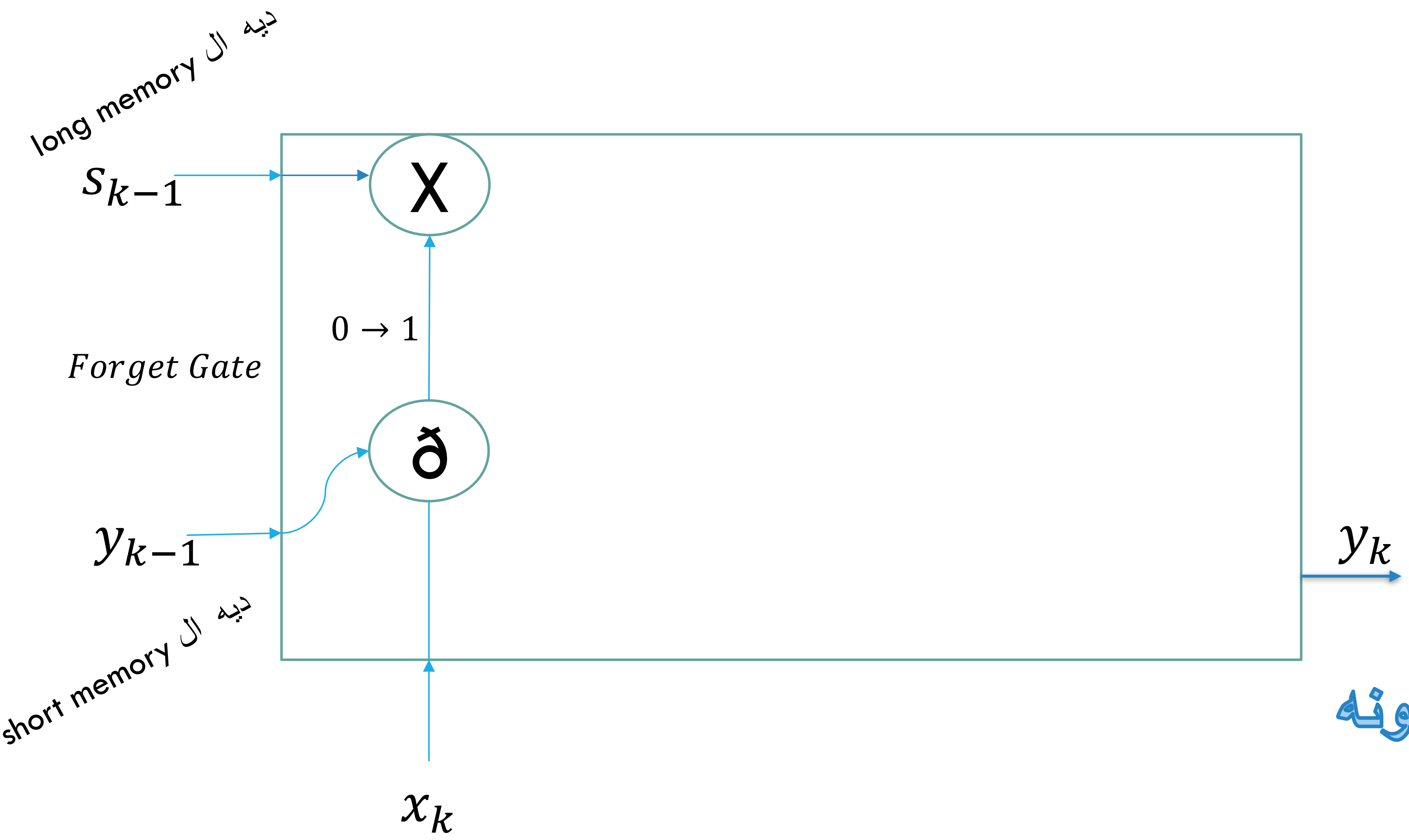
02 LSTM

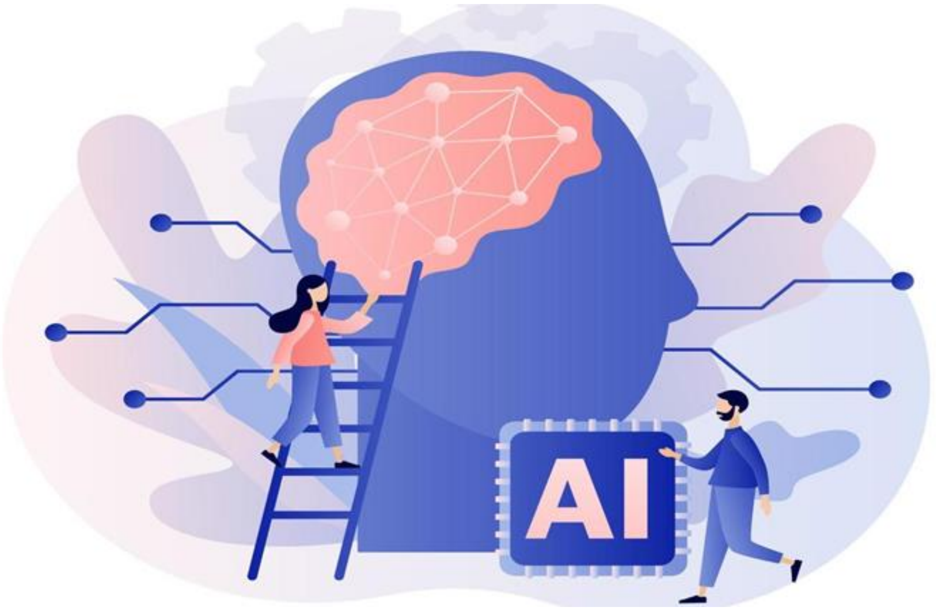


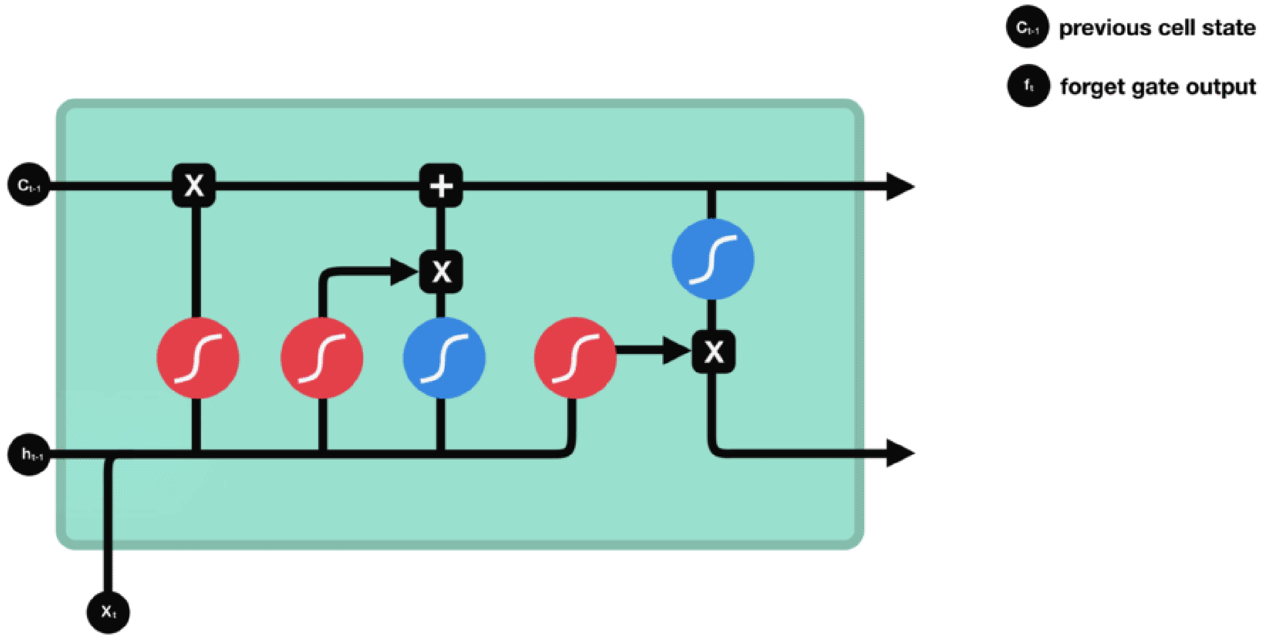


اكتر حاجه هيبقى ليها تاثير على قرارى هيبقى الكلام اللى جايلى واخر موقف مبنا
انما التاريخ الأسود اللى مبنا مش هيبقى ليه تاثير قوى عشان كده هدخل الانبوت
والاوت السابق على sigmoid بحيث اشوف التاريخ الأسود هياثر بنسبه قد ايه
مش يمكن الموقف اللى حصل واخر موقف مبنا يعمينى انى اشوف التاريخ
فالاوتبوت بتاع ال sigmoid وقتها هيبقى بصفر فمش هبص عالهيستورى واقطع علاقتى بيه

02 LSTM



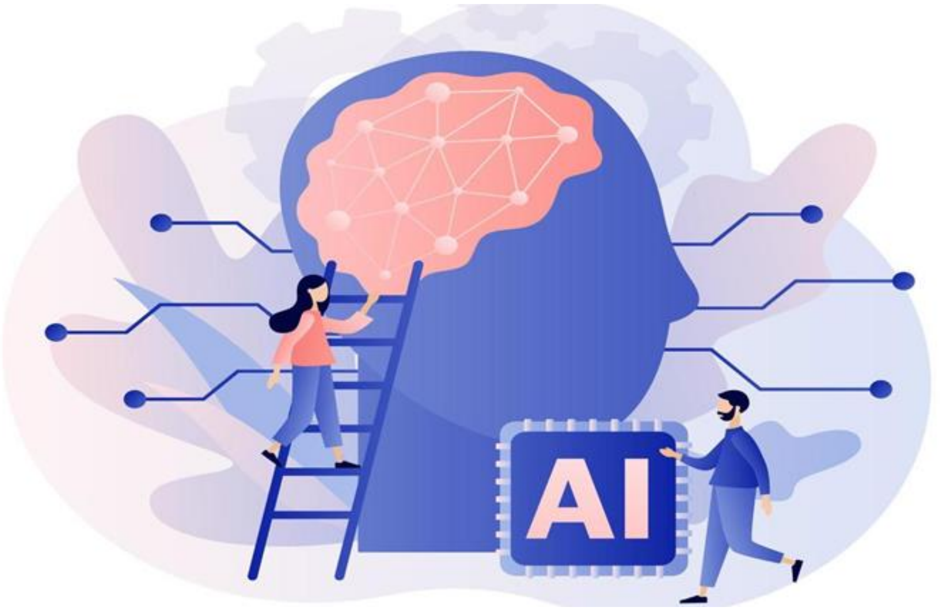


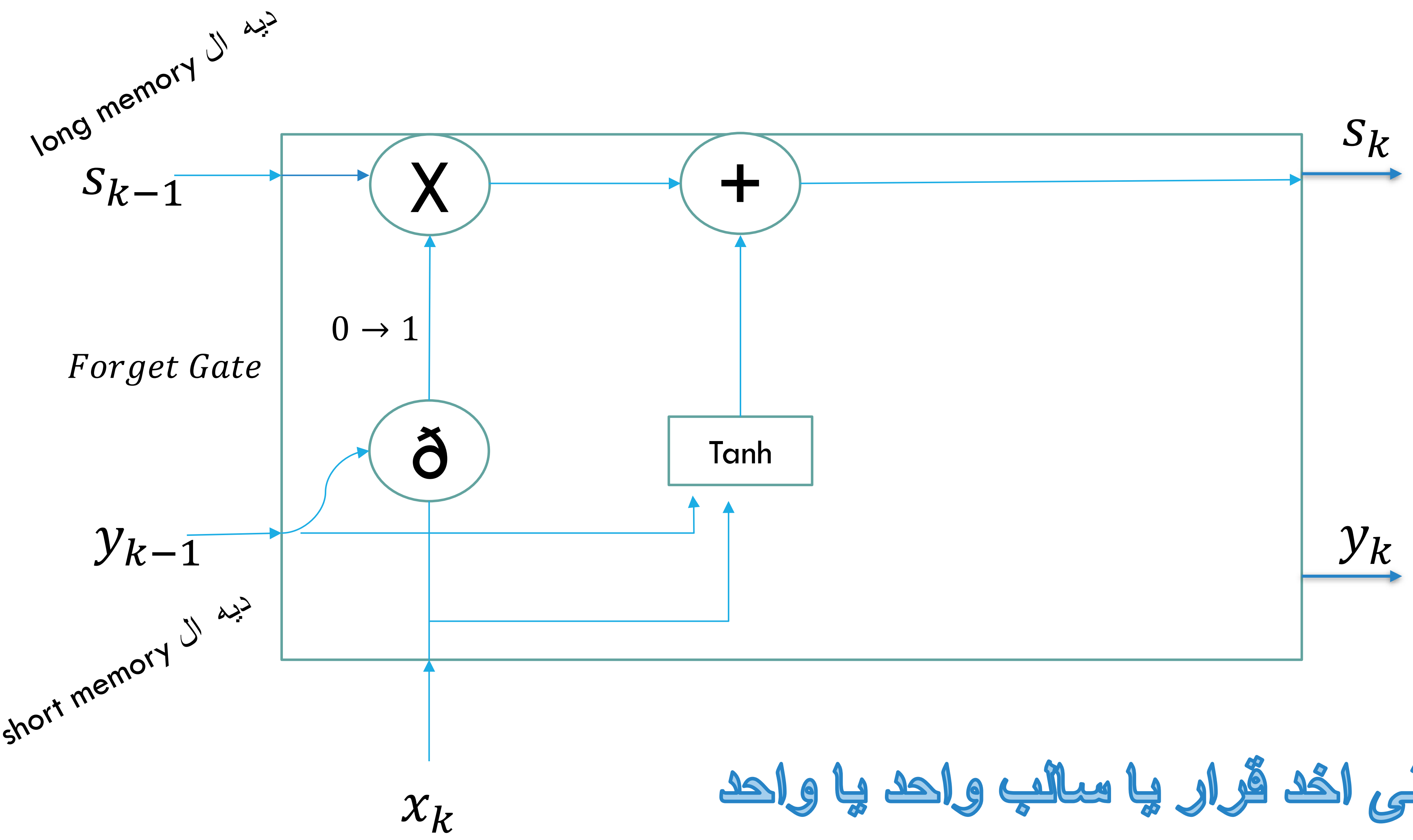


بما ان ديه بتعتمد على انى هاخد قد ايه من الذكرايات والحاجات المدفونه فهنسميها forget gate

- **Forget gate:** Information from the previous hidden state and information from the current input is passed through the sigmoid activation function.
- Values come out between 0 and 1
- Decides what information should be thrown away or kept

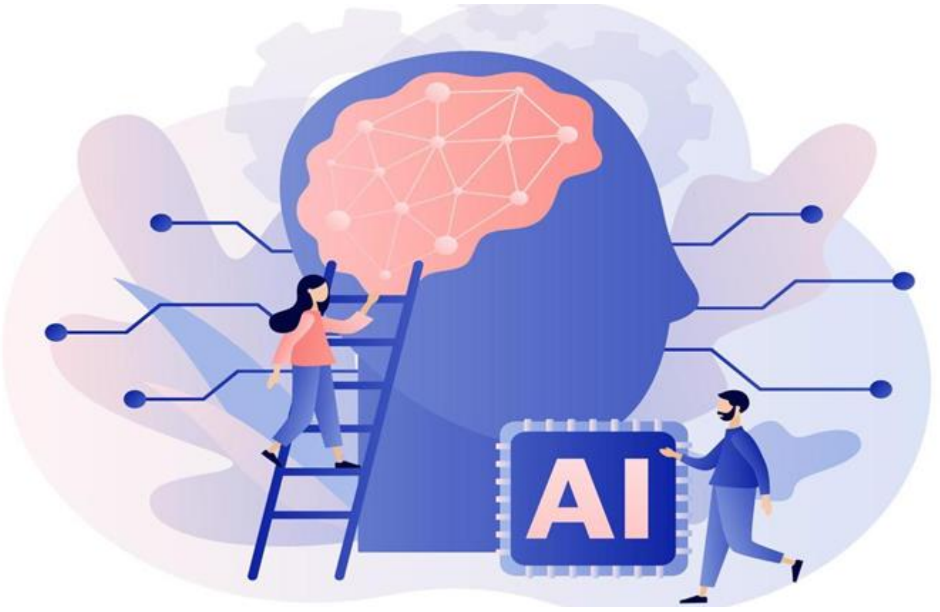
02 LSTM

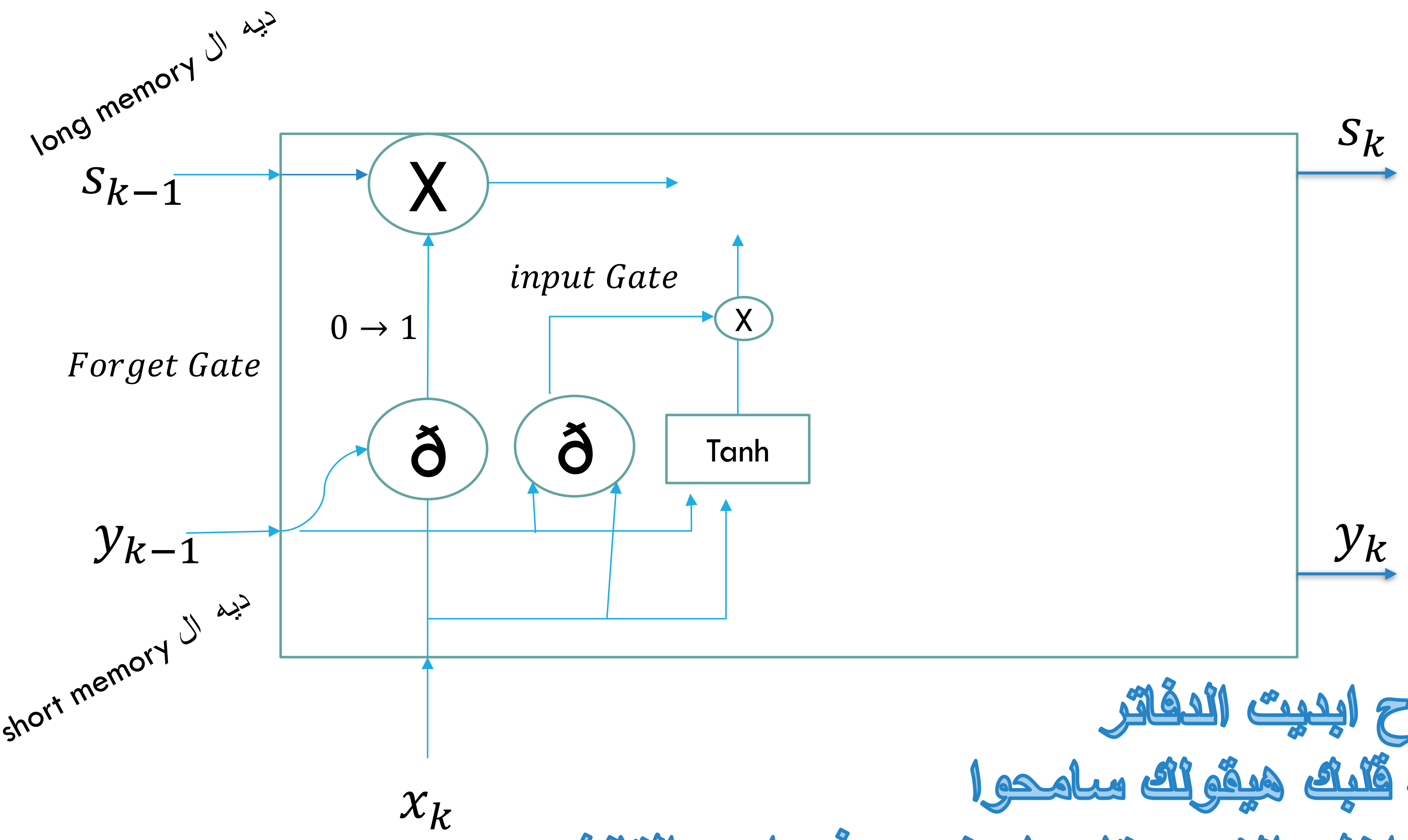


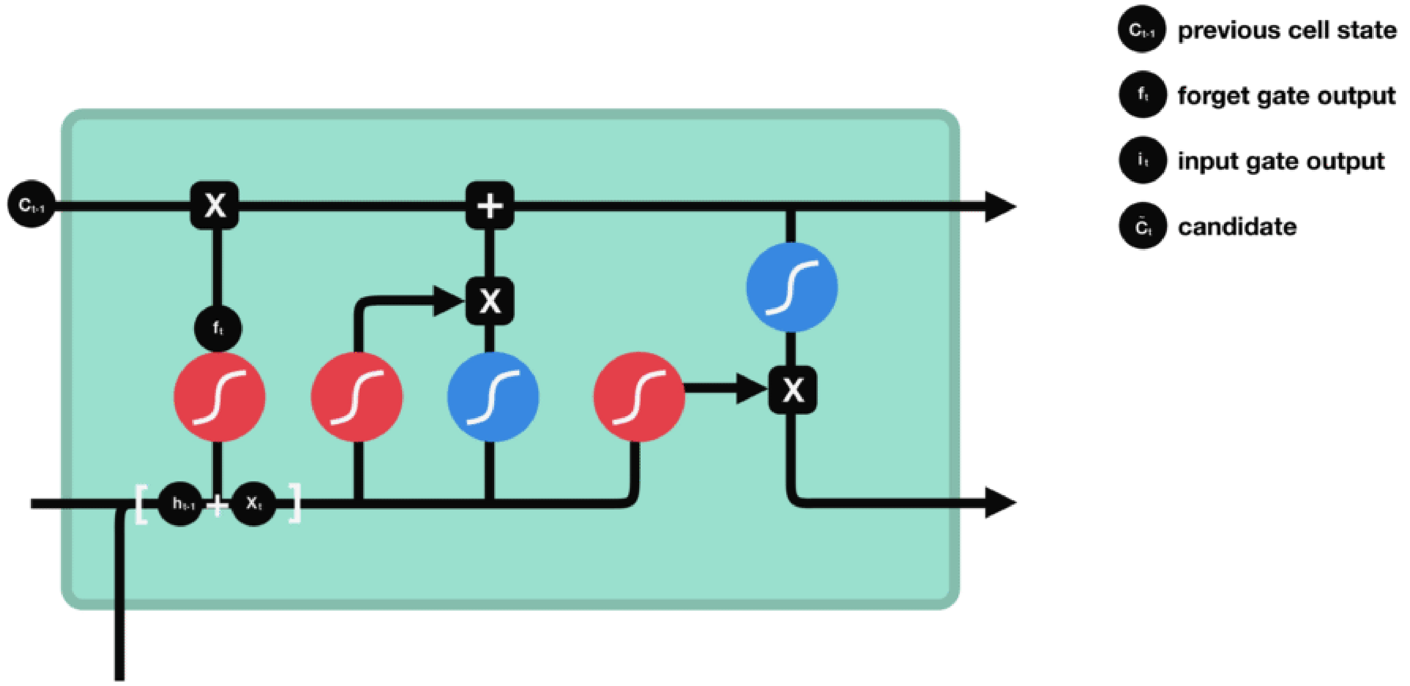


دلوقتى عايز أكون ال state الجديده بمعنى بابديت الدفاتر القديمه
فهاخد الانبوت والاوتبوت القديم وادخلهم على tanh ديه اللى هتخلينى اخد قرار يا سالب واحد يا واحد
فهاخد القرار ده اجمعوا على الاوتبوت اللى جايلى من ال history

02 LSTM



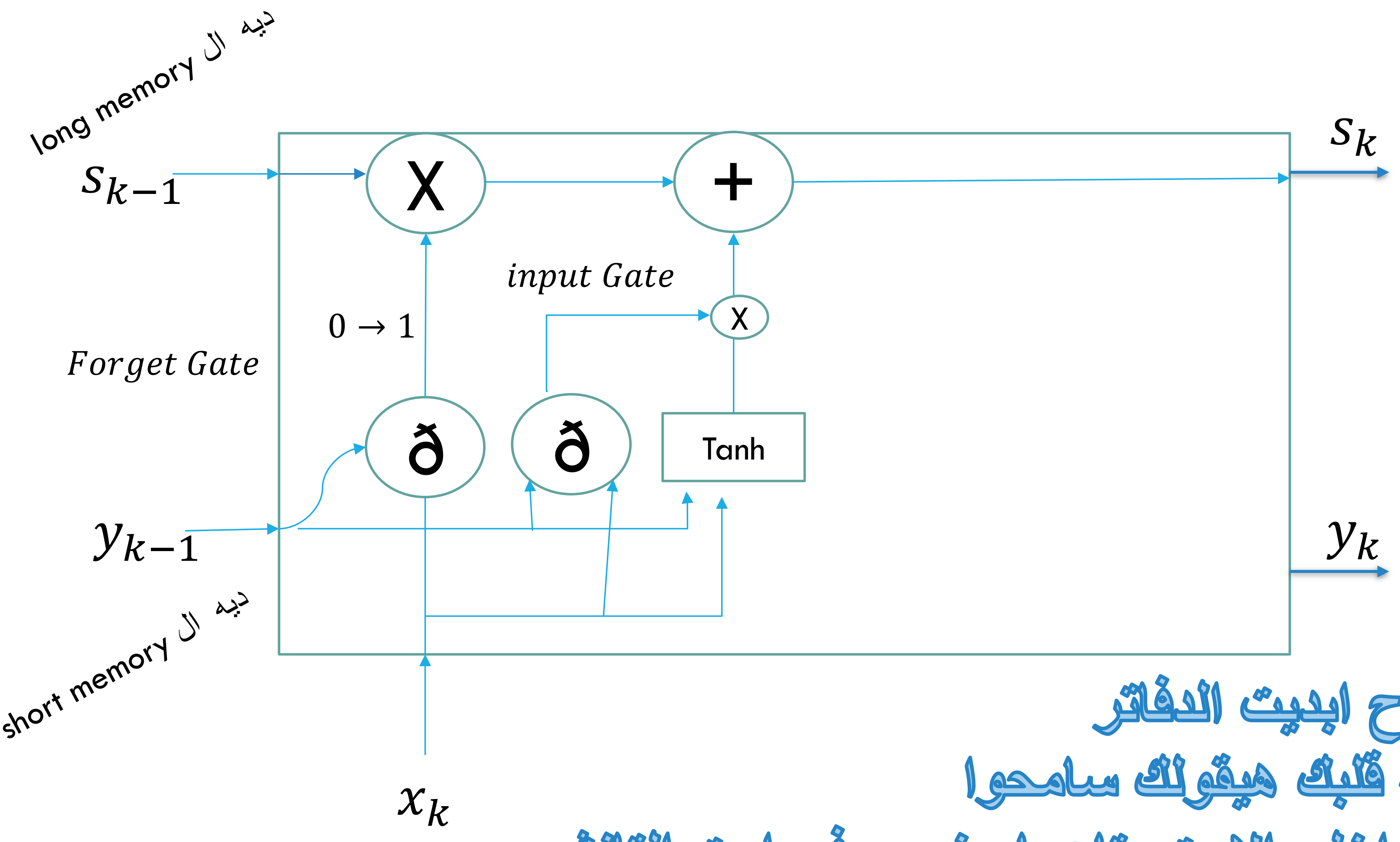


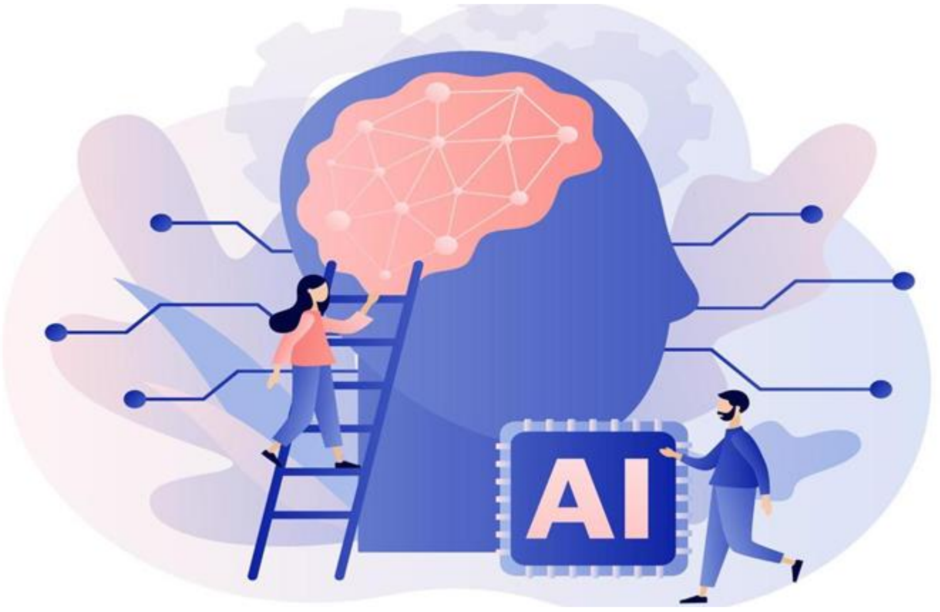


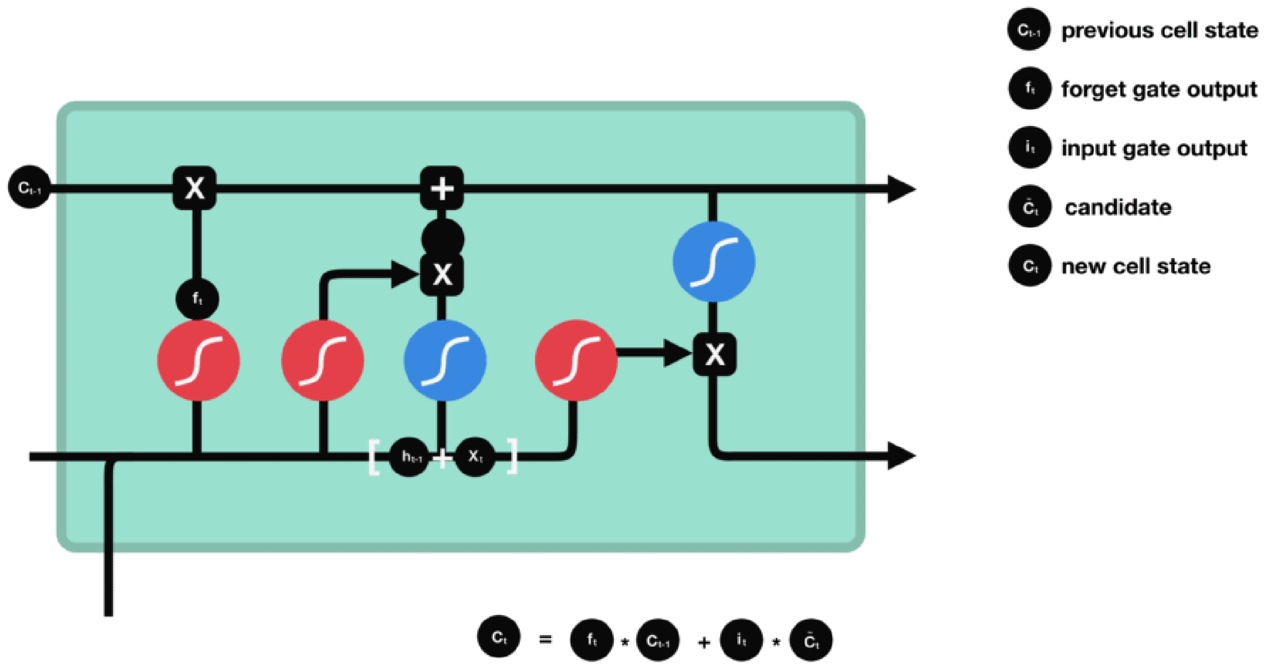
مين قال يعنى ان انت بناء على اللى انت شايفه والموقف الماضى روح ابديت الدفاتر
فانا طيب حبتين فالقرار بتاع ال tanh اللى هيقولى اقطع علاقتك بيه قلبك هيقولك سامحوا
فهاخد الانبوت الحالي والاوت السابق وادخله على sigmoid جديده اخلى الاوت بتاعها يضرب في اوت التانش
فديه هسميها input gate اللى هو انبوت جديد للتاريخ بتاعى

- Input gate
  Decides what new information we're going to store in the cell state

02 LSTM





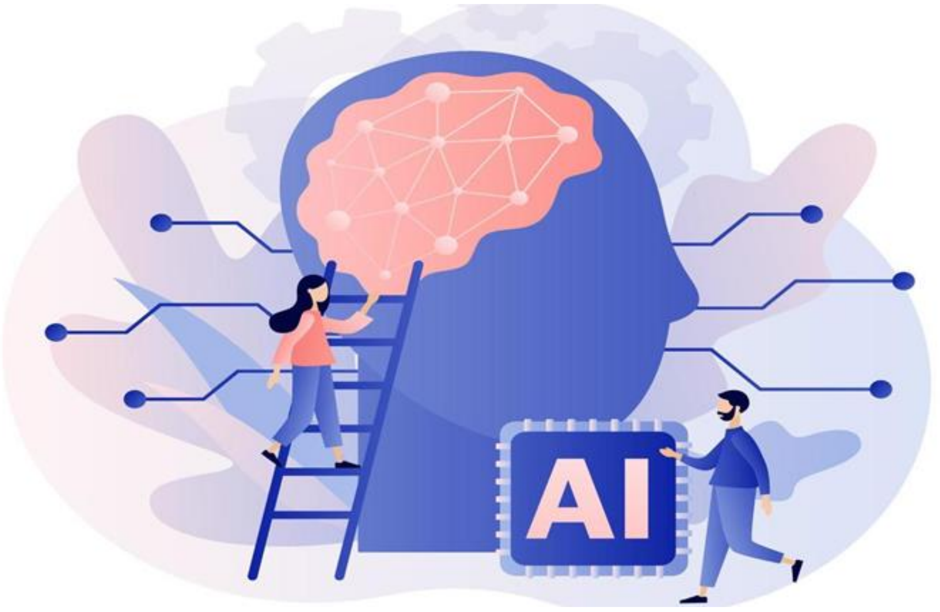


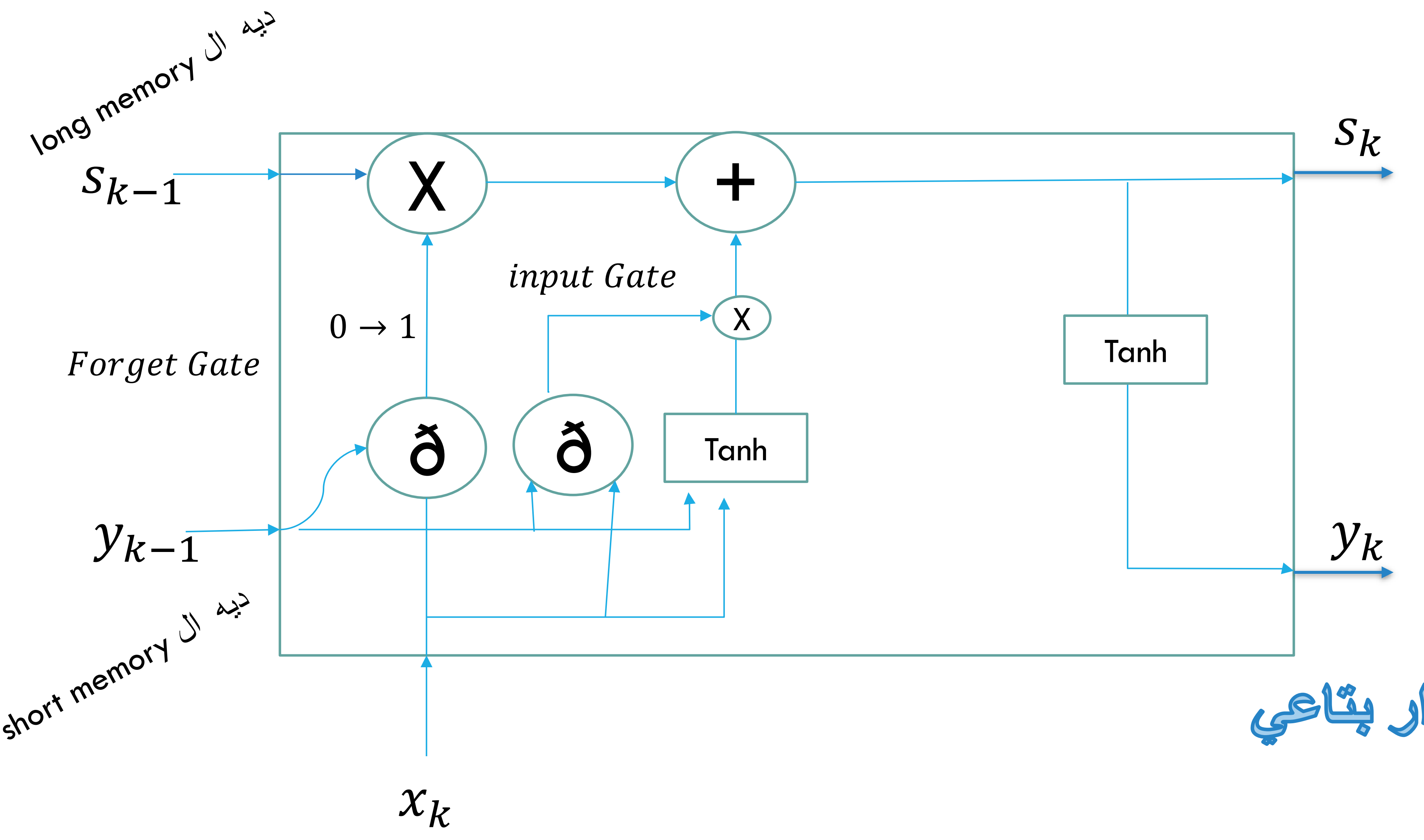
مين قال يعني ان انت بناء على اللى انت شايفه والموقف الماضى روح ابديت الدفاتر
فانا طيب حبتين فالقرار بتاع ال tanh اللى هيقولى اقطع علاقتك بيه قلبك هيقولك سامحوا
فهاخد الانبوت الحالي والاوت السابق وادخله على sigmoid جديده اخلى الاوت بتاعها يضرب في اوت التانش
فديه هسميها input gate اللى هو انبوت جديد للتاريخ بتاعى

- Cell state

  Use outputs of previous gates to update current cell state

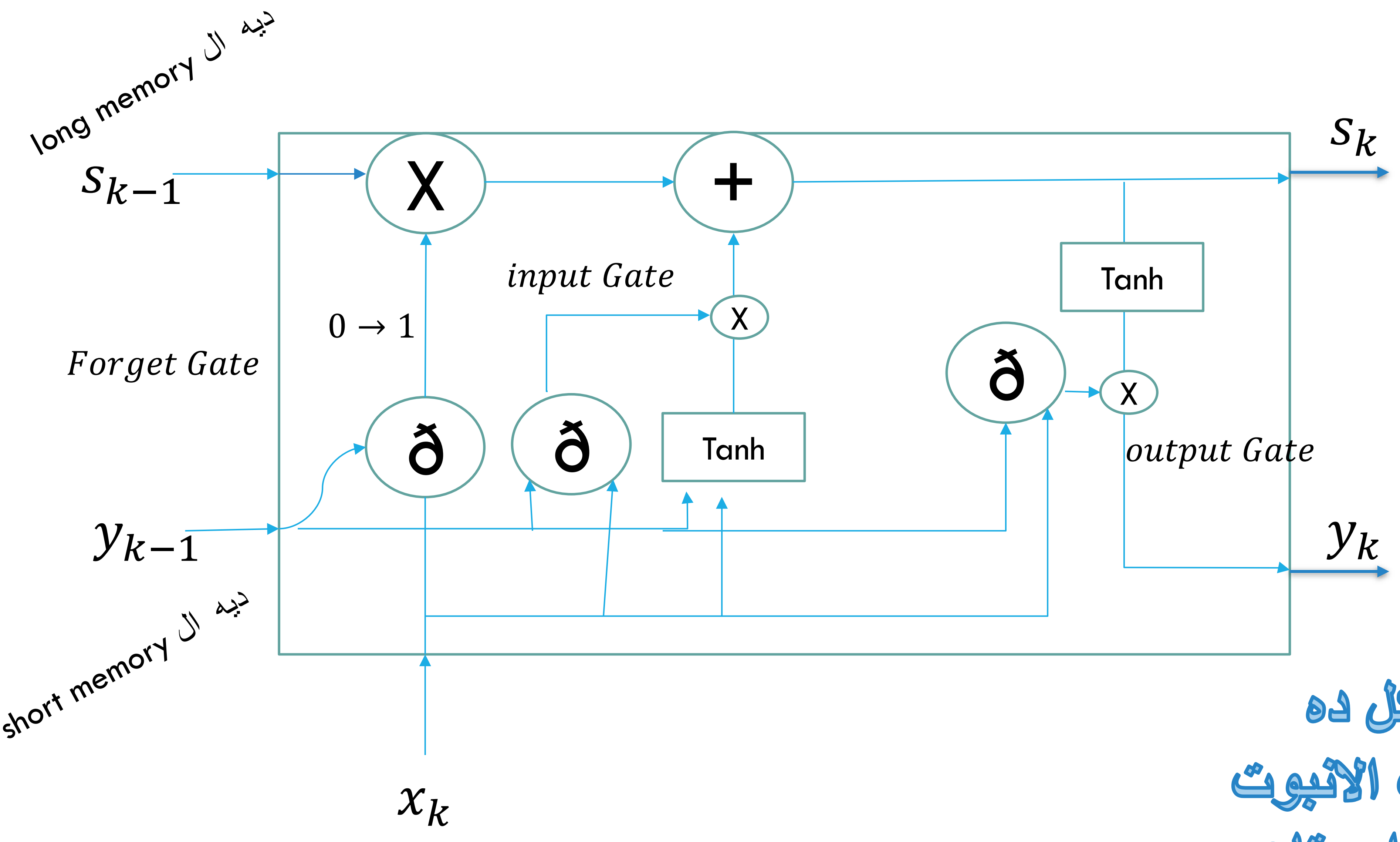
02 LSTM

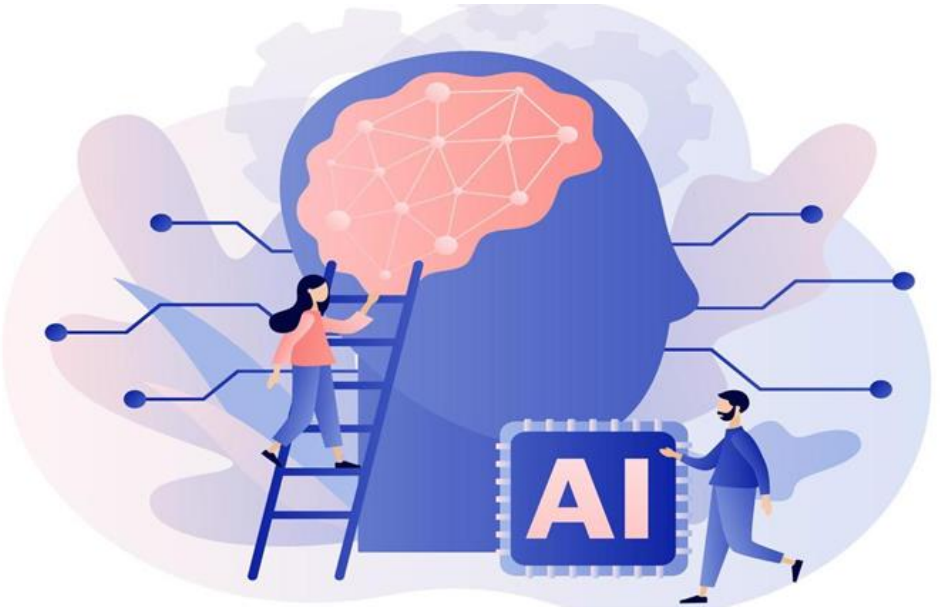


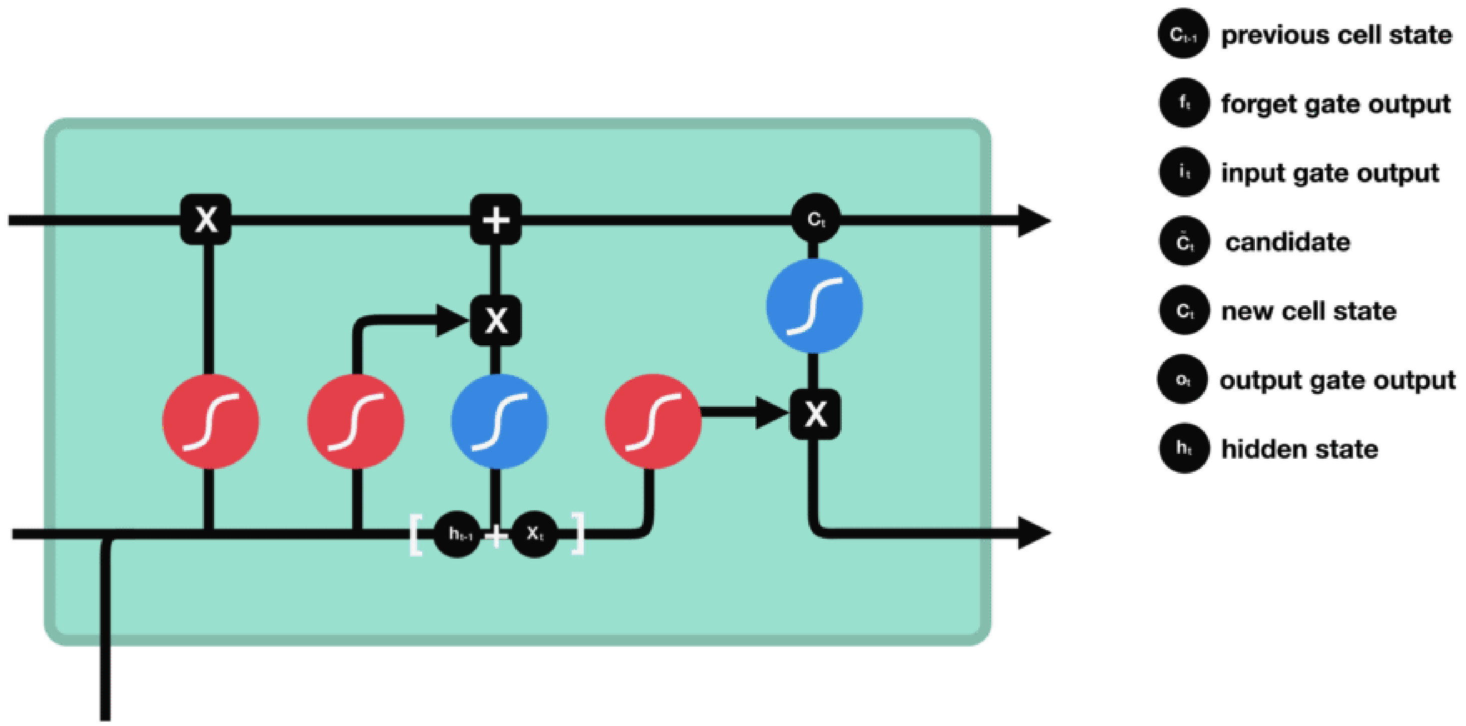


بما ان ديه مساعده ان انا اخد قرار فليه مطلعش منها الاوتبوت القرار بتاعي
يعني ادخلوا على tanh بالشكل ده

02 LSTM





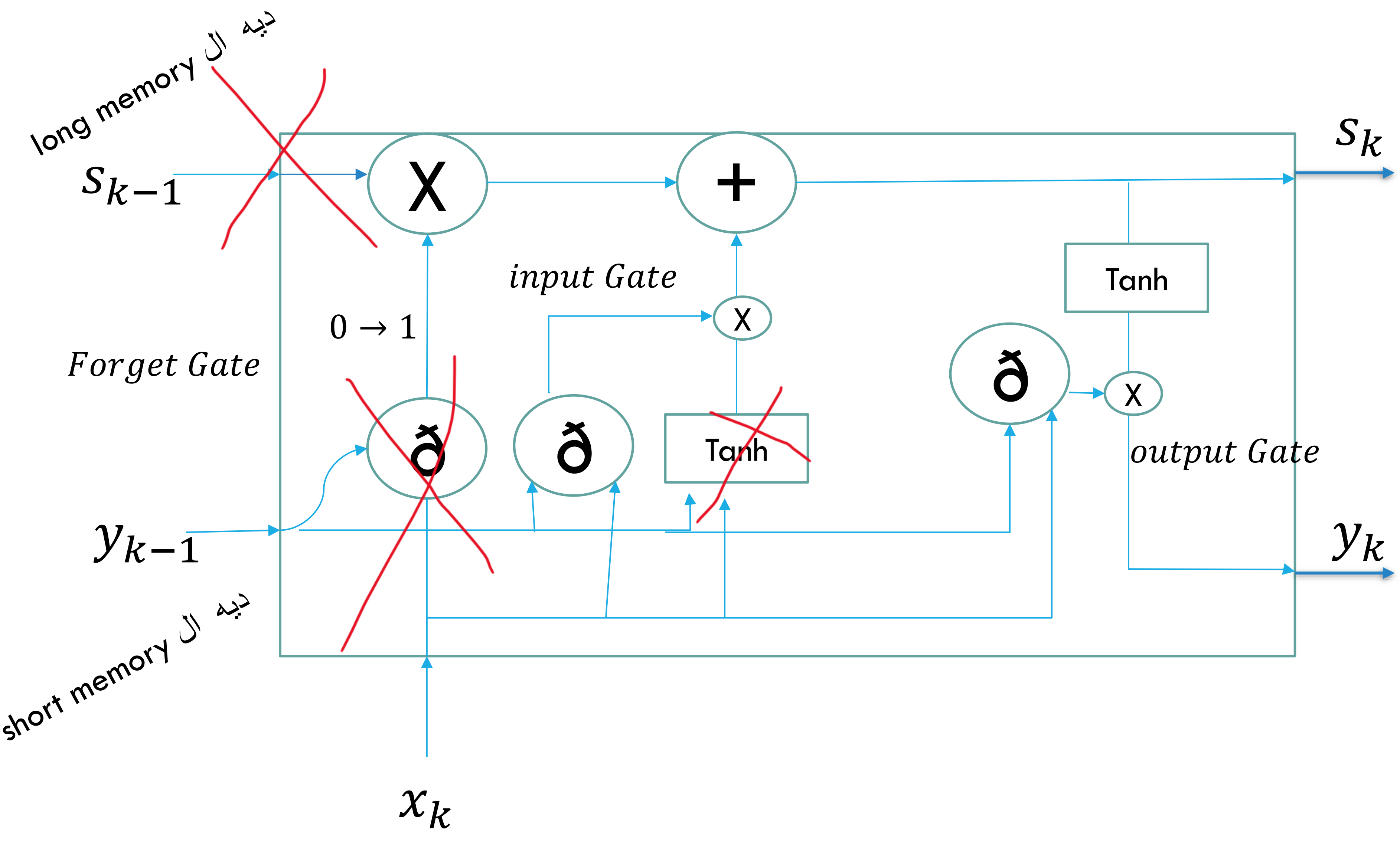


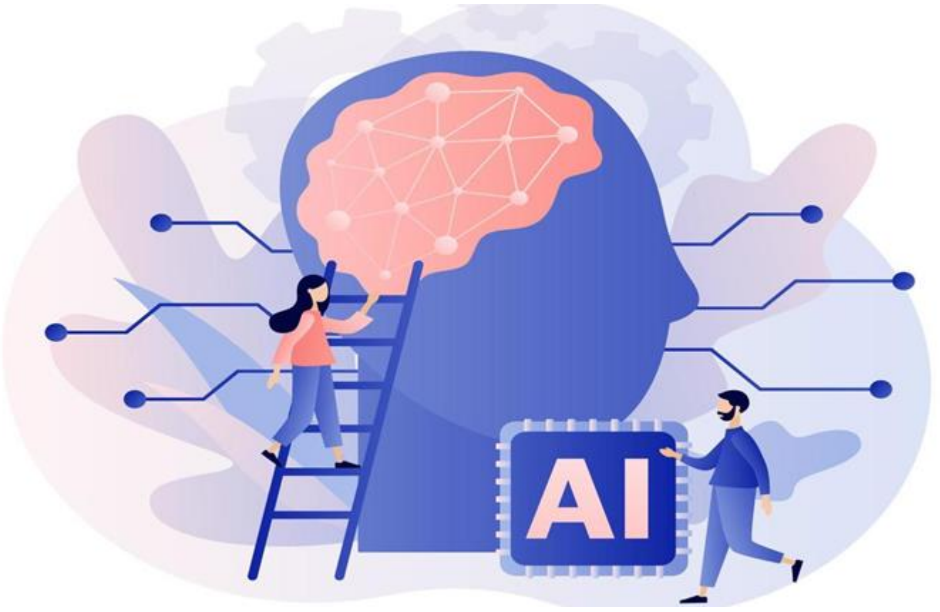
بما ان ديه مساعده ان انا اخد قرار فليه مطلعش منها الاوتبوت بالشكل ده قلبك الحنين هيقولك لا القرار اللى اخدته متاخدوش كله فرجعت اخدت الاتبوت مع الاوتبوت السابق دخلتهم على sigmoid وضربت الناتج في القرار بتاعى ديه اللى هي ال output gate بتاعتى.

- Output gate

  Decides what new information to pass along as our hidden state

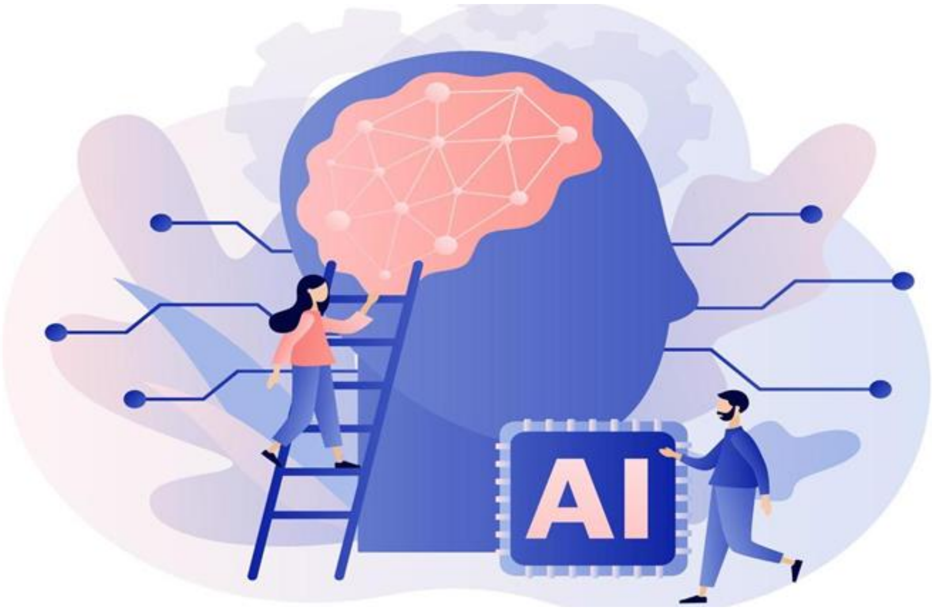
03 GRU

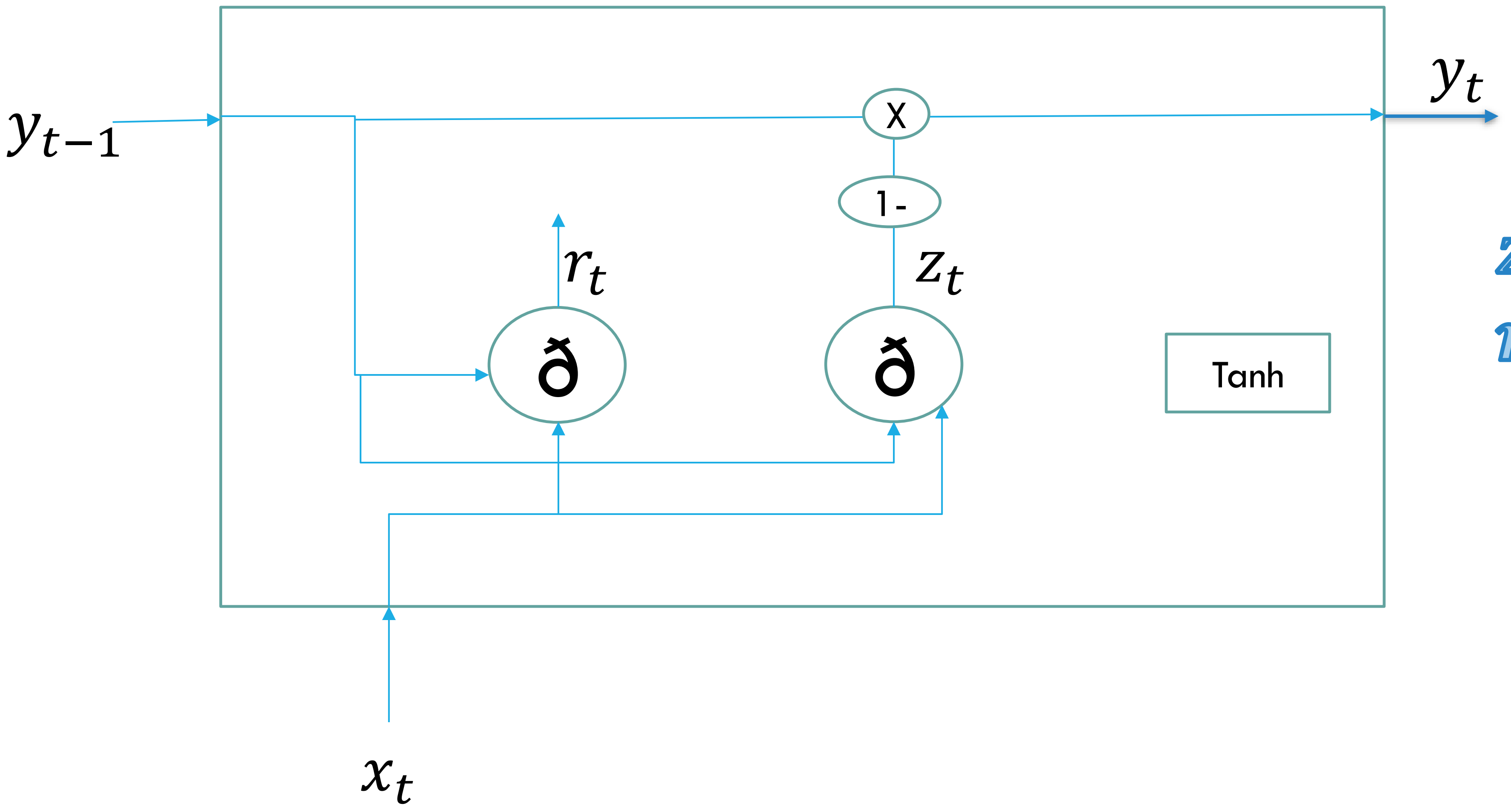




طب ليه التعقيد ده كله مش ممكن أقول بس ان $y_k = F(x_k, y_{k-1})$
وكده كده ال y_{k-1} بتعتمد على ال y_{k-2} وهكذا.....
يبقى هنلغى الجزء بتاع ال history وبالتابعيه هنلغى جزء ال sigmoid بتاعه
وكمان جزء ال tanh اللى كان بيابديت ال history

03 GRU





$z_t = \eth(y_{t-1}, x_t)$

$r_t = \eth(y_{t-1}, x_t)$

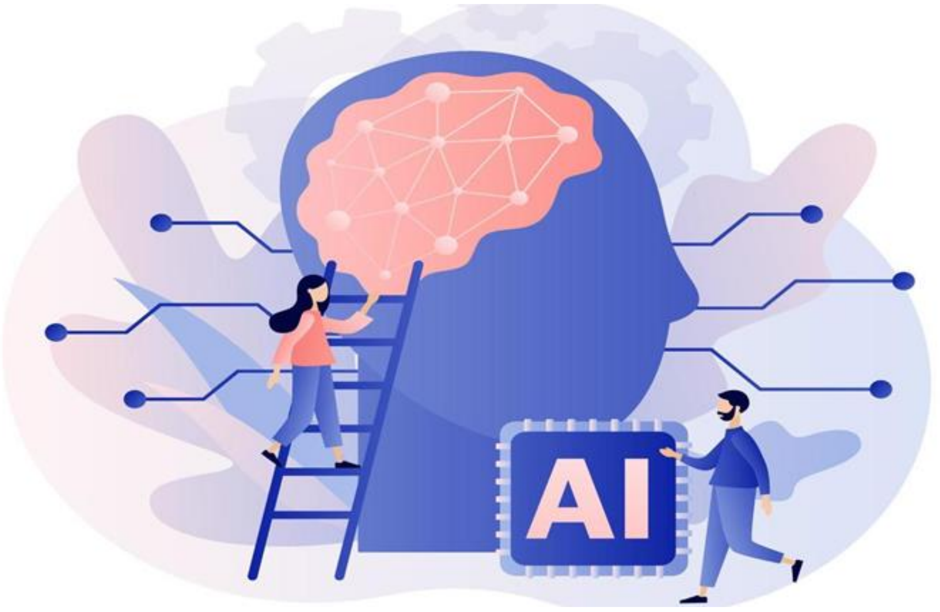
طب ليه التعقيد ده كله مش ممكن أقول بس ان $y_t = F(x_t, y_{t-1})$

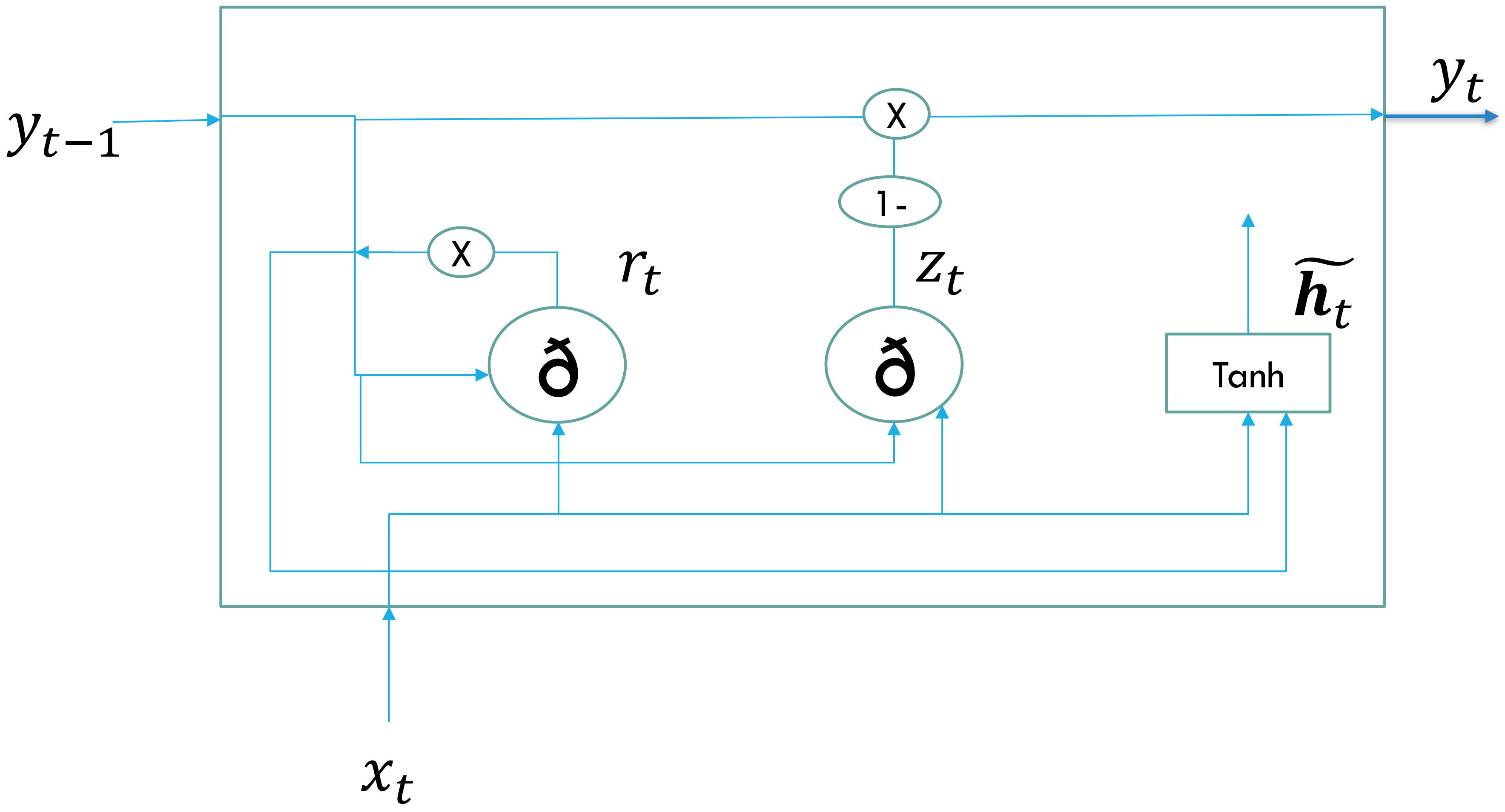
وكده كده ال y_{t-1} بتعتمد على ال y_{t-2} وهكذا.....

يبقى هنلغى الجزء بتاع ال history وبالتابعيه هنلغى جزء ال sigmoid بتاعه

وكمان جزء ال tanh اللى كان بيابديت ال history

03 GRU

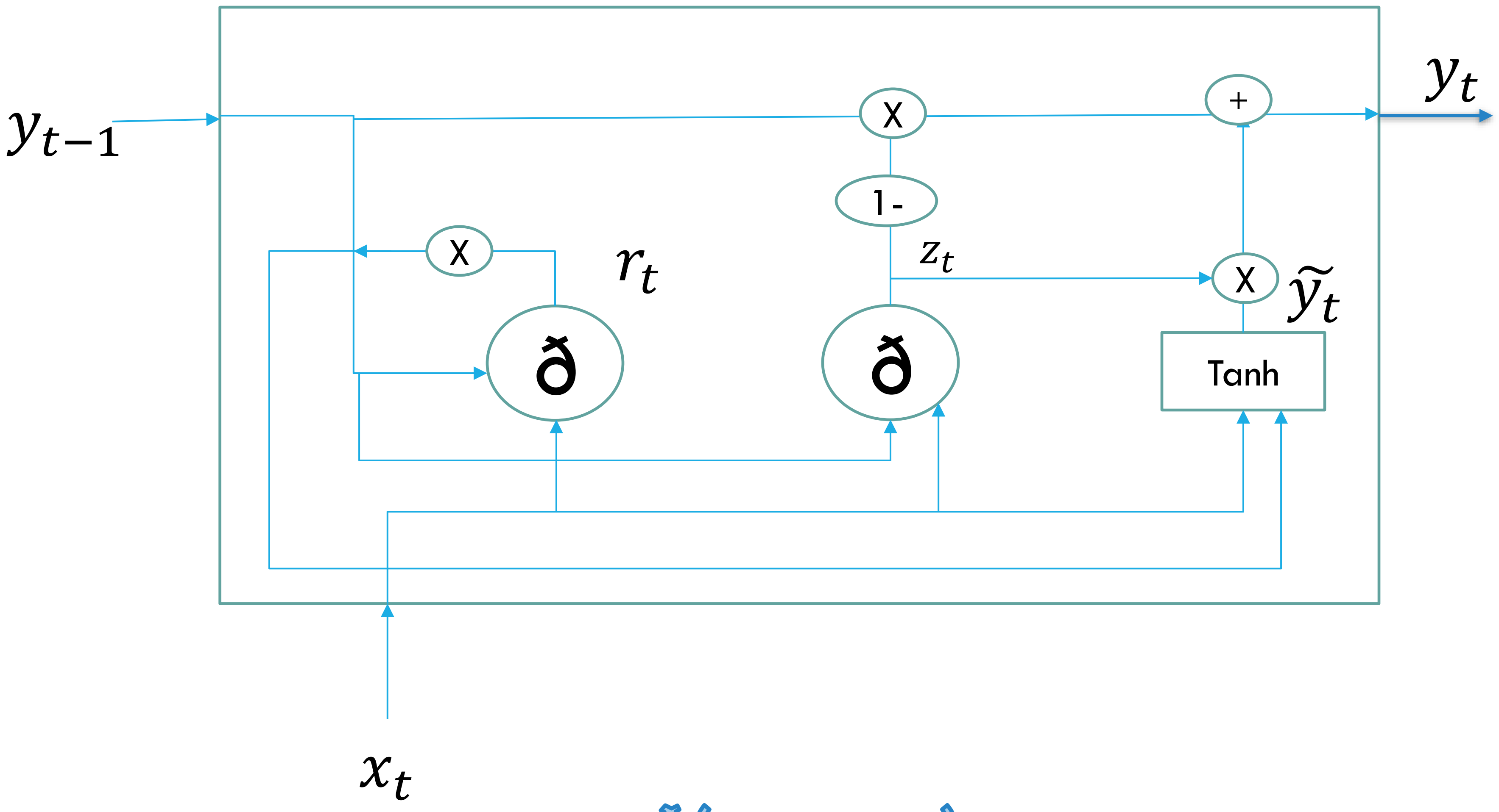


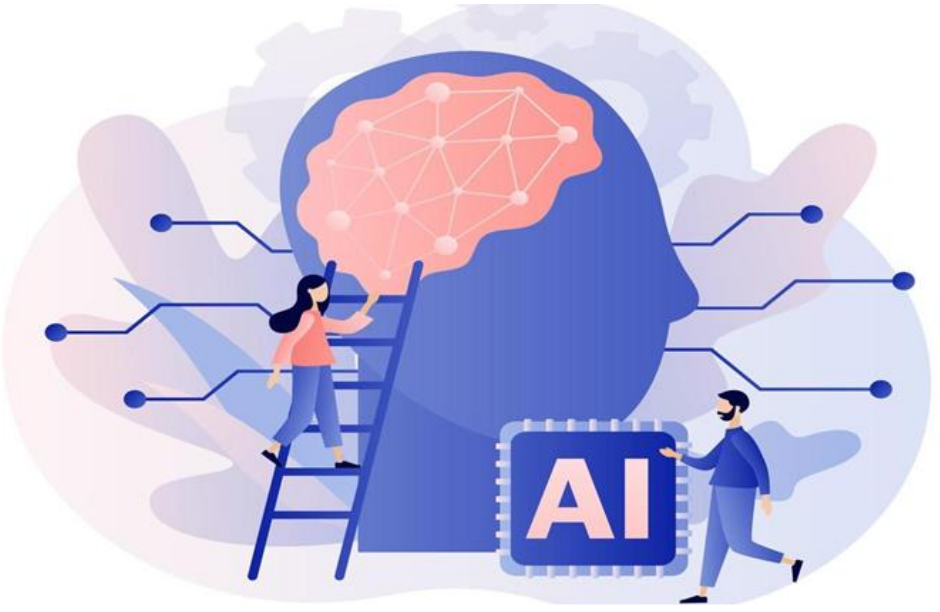


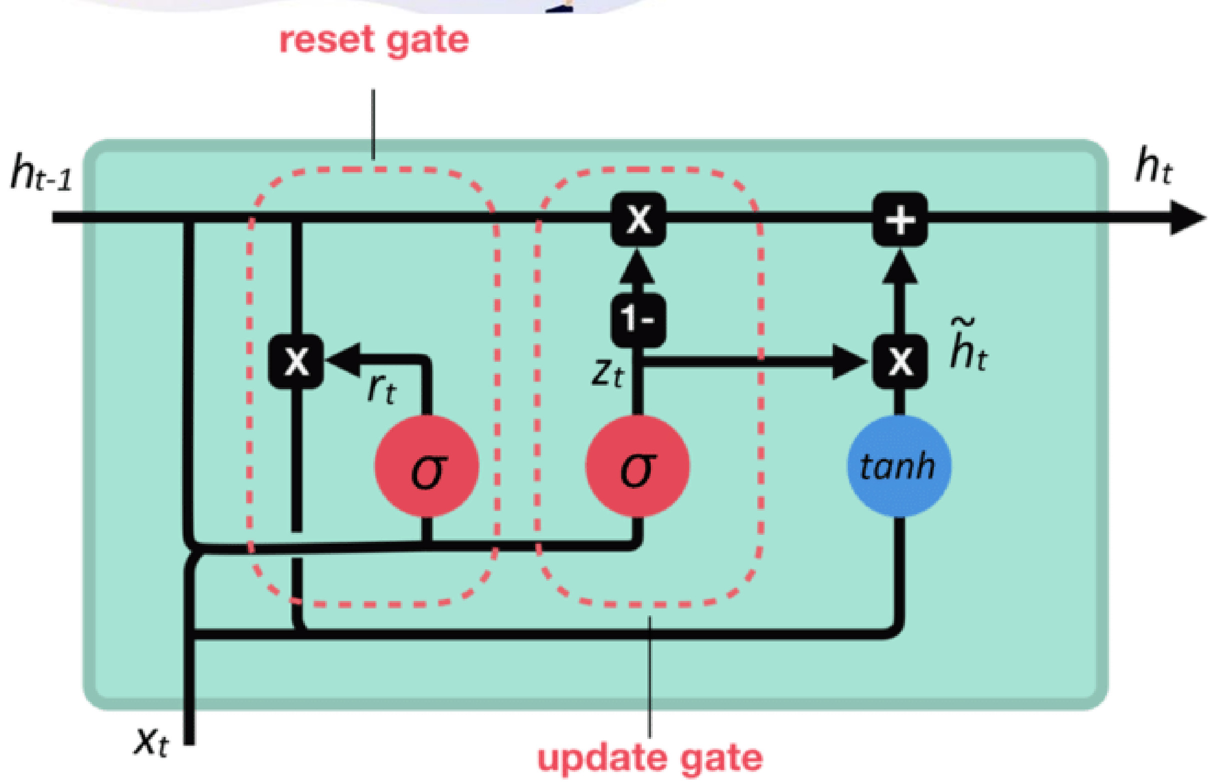
$$z_t = \eth(y_{t-1}, x_t)$$
$$r_t = \eth(y_{t-1}, x_t)$$
$$\widetilde{h}_t = \tanh(r_t * y_{t-1}, x_t)$$

طب ليه التعقيد ده كله مش ممكن أقول بس ان $y_t = F(x_t, y_{t-1})$
وكده كده ال y_{t-1} بتعتمد على ال y_{t-2} وهكذا.....
يبقى هنلغى الجزء بتاع ال history وبالتابعيه هنلغى جزء ال sigmoid بتاعه
وكمان جزء ال tanh اللى كان بيابديت ال history

03 GRU







طب ليه التعقيد ده كله مش ممكن أقول بس ان $y_t = F(x_t, y_{t-1})$

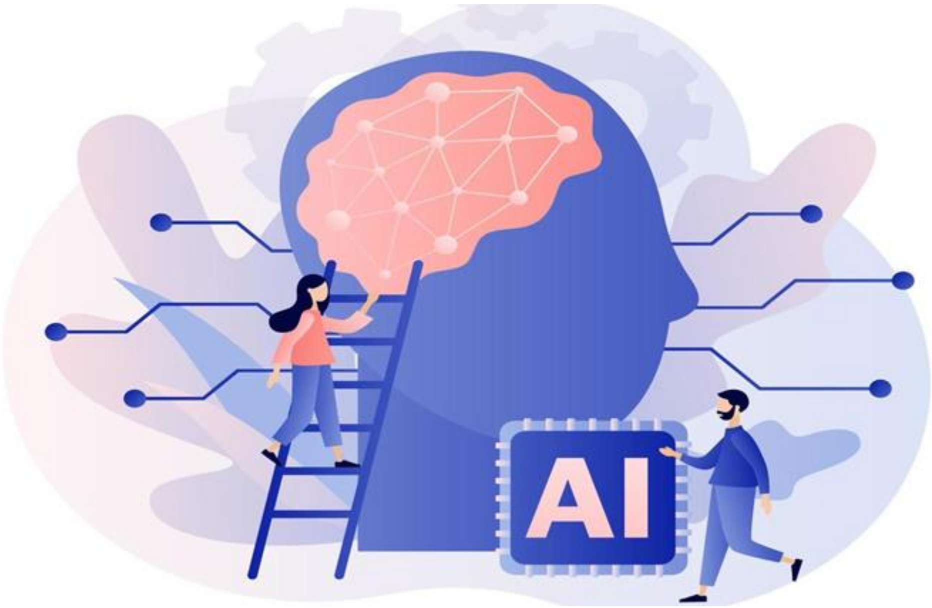
وكده كده ال y_{t-1} بتعتمد على ال y_{t-2} وهكذا.....

يبقى هنلغى الجزء بتاع ال history وبالتابعيه هنلغى جزء ال sigmoid بتاعه

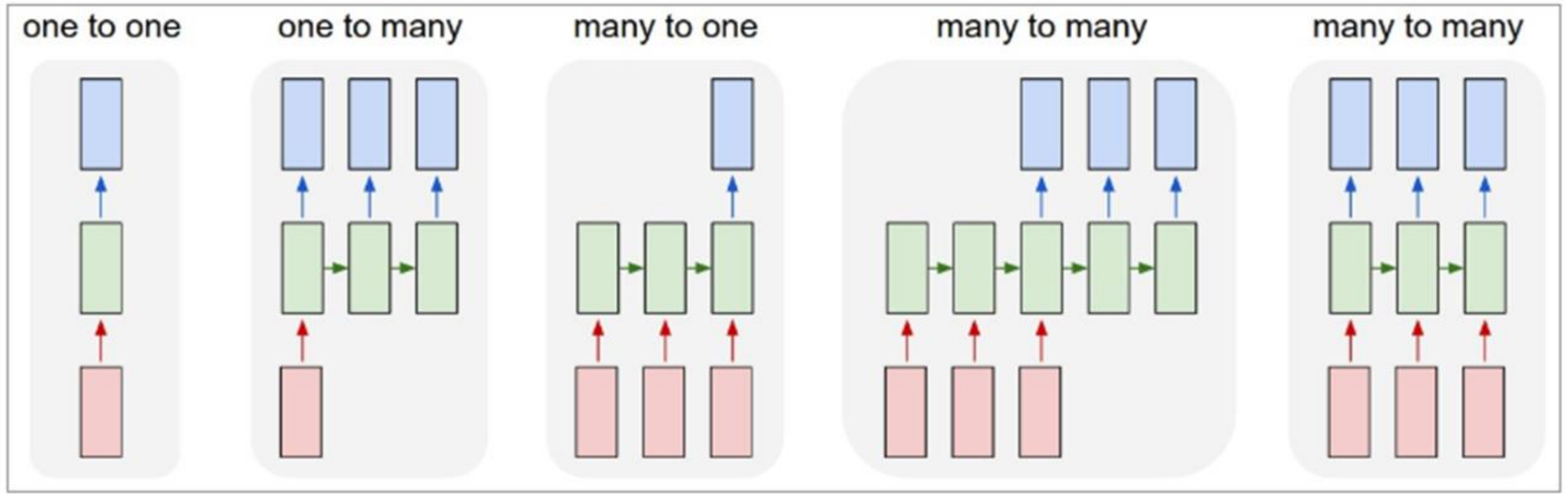
وكمان جزء ال tanh اللى كان بيابديت ال history

$$z_t = ð(y_{t-1}, x_t)$$
$$r_t = ð(y_{t-1}, x_t)$$
$$\tilde{y}_t = \tanh(r_t * y_{t-1}, x_t)$$
$$y_t = (1 - z_t) * y_{t-1} + z_t * \tilde{y}_t$$

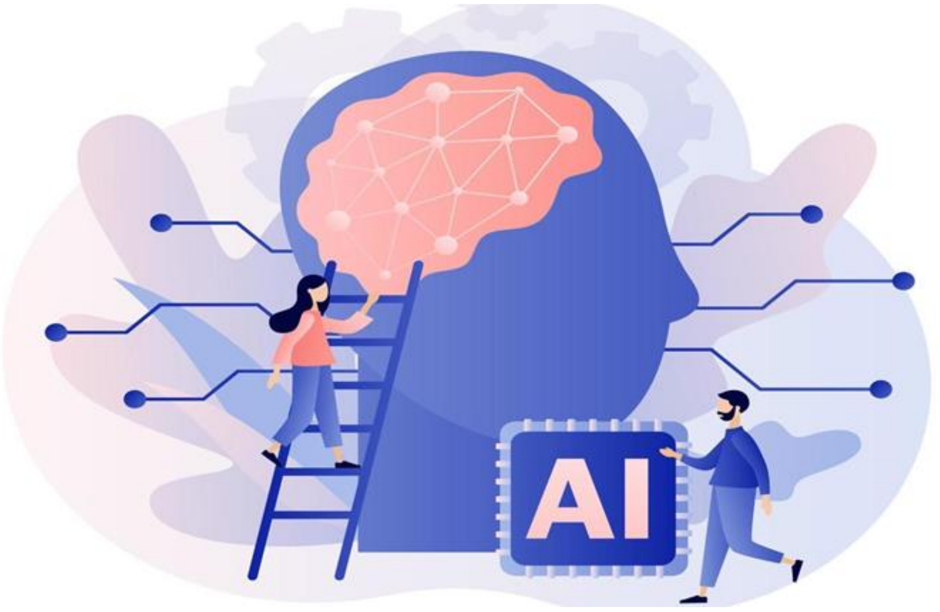
01 RNN topologies



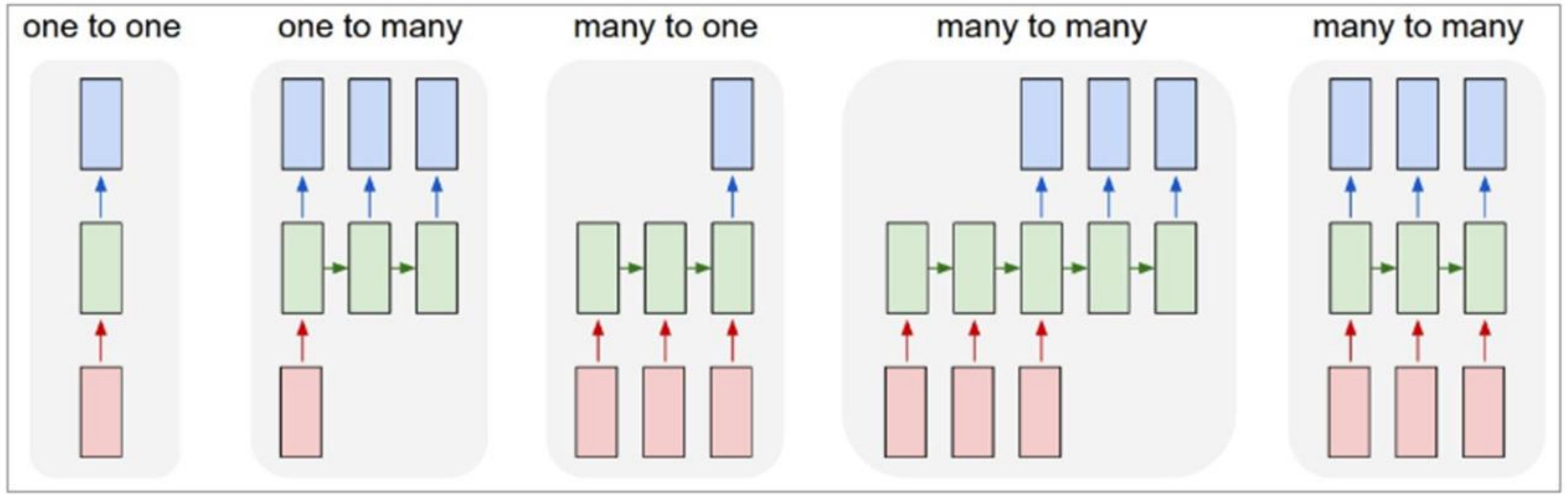
- The APIs for MLP and CNN architectures are limited. Both architectures accept a fixed-size tensor as input and produce a fixed-size tensor as output; and they perform the transformation from input to output in a fixed number of steps given by the number of layers in the model.
- RNNs don't have this limitation—you can have sequences in the input, the output, or both. This means that RNNs can be arranged in many ways to solve specific problems.
- RNNs combine the input vector with the previous state vector to produce a new state vector. This can be thought of as similar to running a program with some inputs and some internal variables.



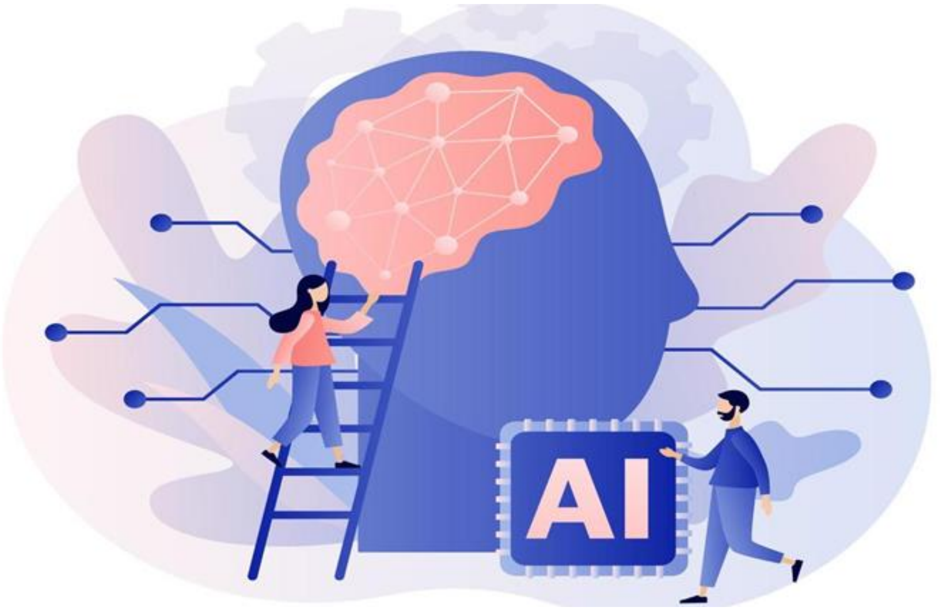
01 RNN topologies



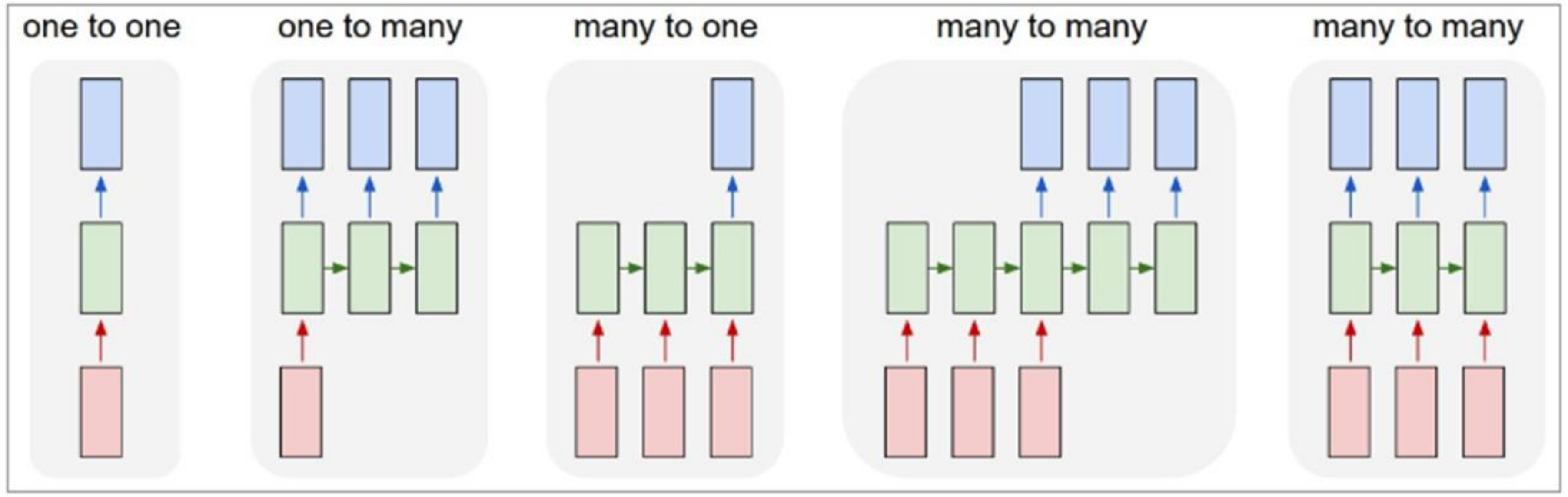
- The RNNs are more exciting because they allow us to operate over sequences of vectors: Sequences in the input, the output, or in the most general case both.
- This property of being able to work with sequences gives rise to a number of common topologies shown below:
- In the above diagram each rectangle is a vector and arrows represent functions (e.g. matrix multiply).
- Input vectors are in red, output vectors are in blue and green vectors hold the RNN’s state.

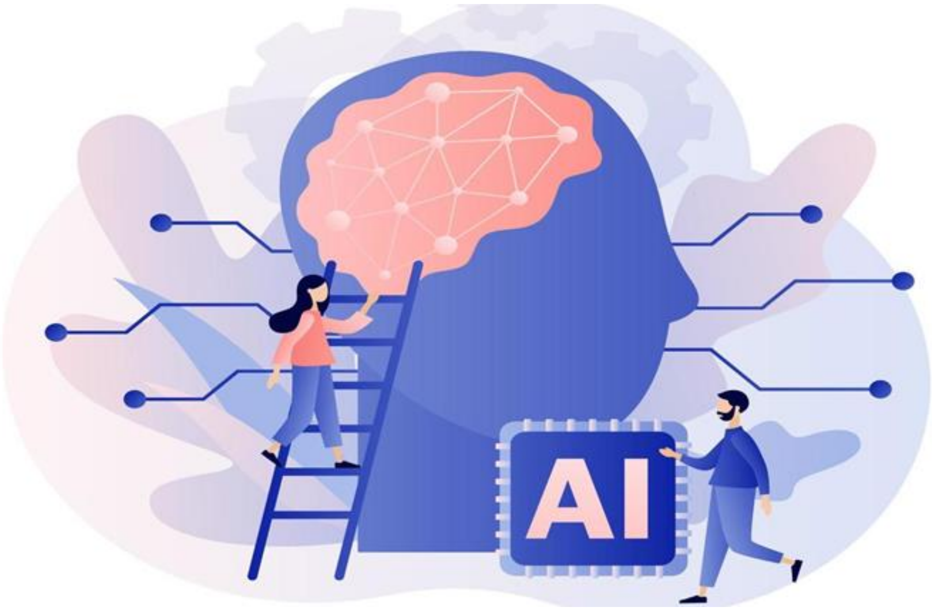


01 RNN topologies



- From left to right:
  1. Vanilla mode of processing without RNN, from fixed-sized input to fixed-sized output (e.g. image classification).
  2. Sequence output (e.g. image captioning takes an image and outputs a sentence of words).
  3. Sequence input (e.g. sentiment analysis where a given sentence is classified as expressing positive or negative sentiment).
  4. Sequence input and sequence output (e.g. Machine Translation: an RNN reads a sentence in English and then outputs a sentence in French).
  5. Synced sequence input and output (e.g. video classification where we wish to label each frame of the video).





Thank You...!

End