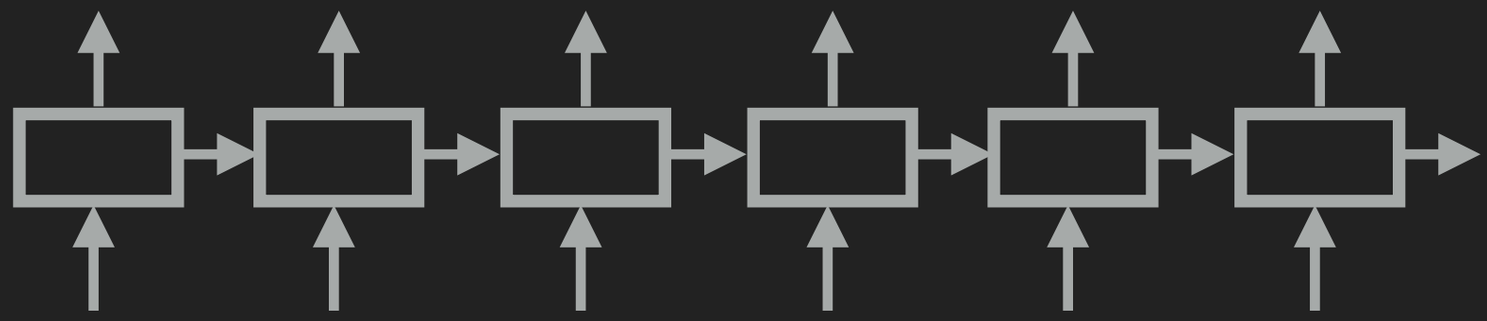
FERRET



ANDRII GAKHOV

RECURRENT NEURAL NETWORKS

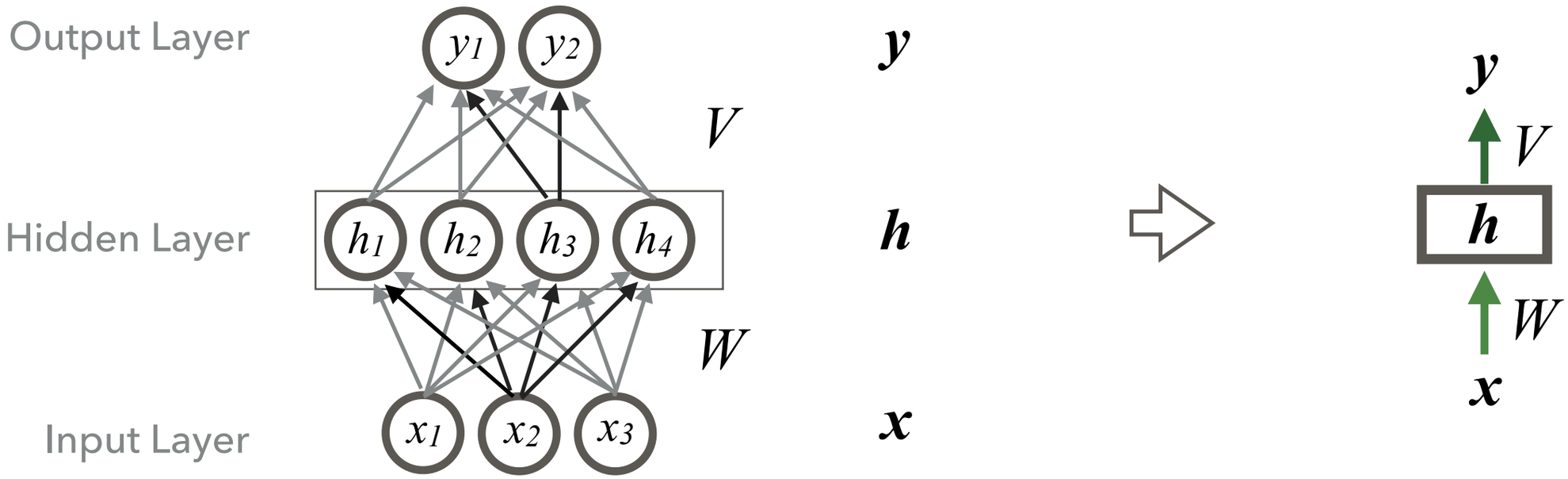
PART 1: THEORY

10.12.2015
Techtalk @ ferret

FEEDFORWARD NEURAL NETWORKS

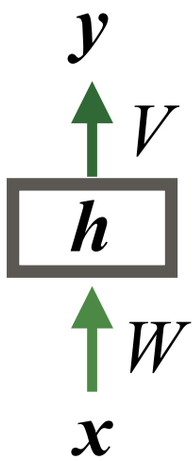
NEURAL NETWORKS: INTUITION

- **Neural network** is a computational graph whose nodes are computing units and whose directed edges transmit numerical information from node to node.
- Each computing unit (neuron) is capable of evaluating a single primitive function (activation function) of its input.
- In fact the network represents a chain of function compositions which transform an input to an output vector.



NEURAL NETWORKS: FNN

- The Feedforward neural network (FNN) is the most basic and widely used artificial neural network. It consists of a number of layers of computational units that are arranged into a layered configuration.



$$h = g(Wx)$$

$$y = f(Vh)$$

- g - activation function for the hidden layer units
- f - activation function for the output layer units

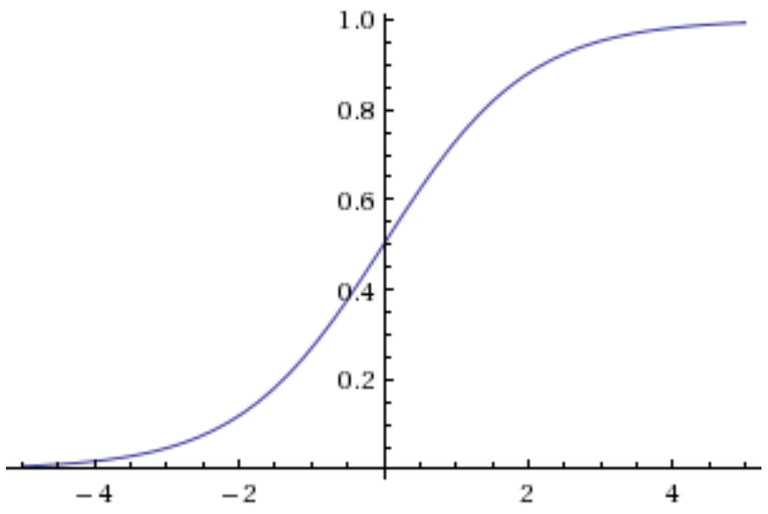
Unlike all hidden layers in a neural network, **the output layer units** most commonly have as activation function:

- **linear identity function** (for regression problems)
- **softmax** (for classification problems).

NEURAL NETWORKS: POPULAR ACTIVATION FUNCTIONS

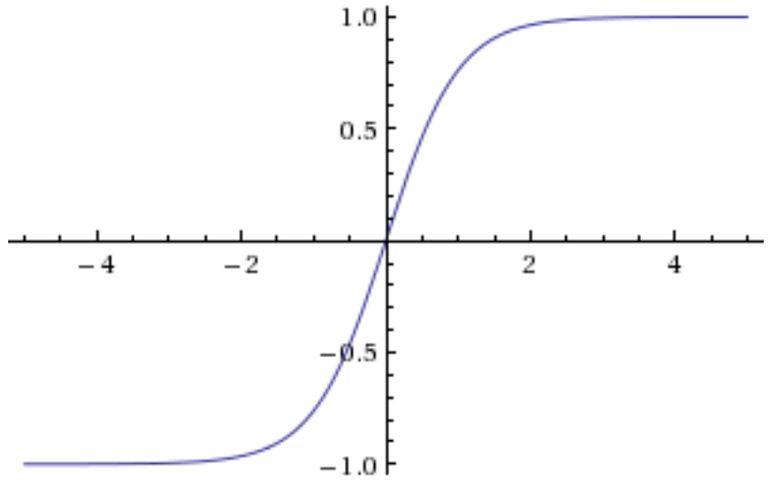
sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}, \quad \sigma : \mathbb{R} \to (0,1)$$



tanh

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}, \quad \tanh : \mathbb{R} \to (-1,1)$$



- Nonlinear “squashing” functions
- Easy to find the derivative
- Make stronger weak signals and don’t pay too much attention to already strong signals

NEURAL NETWORKS: HOW TO TRAIN

- The main problems for neural network training:
  - billions parameters of the model
  - multi-objective problem
  - requirement of the high-level parallelism
  - requirement to find wide domain where all minimising functions are close to their minimums

In general, training of neural network is an **error minimization problem**

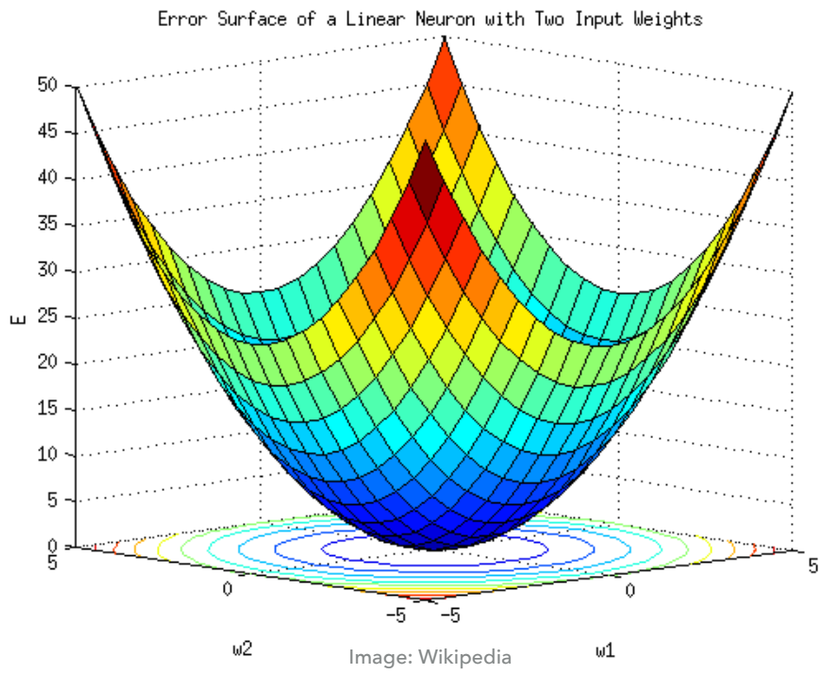


Image: Wikipedia

NEURAL NETWORKS: BP

- Feedforward neural network can be trained with **backpropagation algorithm** (BP) - propagated **gradient descent algorithm**, where gradients are propagated **backward**, leading to very efficient computing of the higher layer weight change.
- **Backpropagation algorithm consists of 2 phases:**
  - **Propagation.** Forward propagation of inputs through the neural network and generation output values. Backward propagation of the output values through the neural network and calculate errors on each layer.
  - **Weight update.** Calculate gradients and correct weights proportional to the negative gradient of the cost function

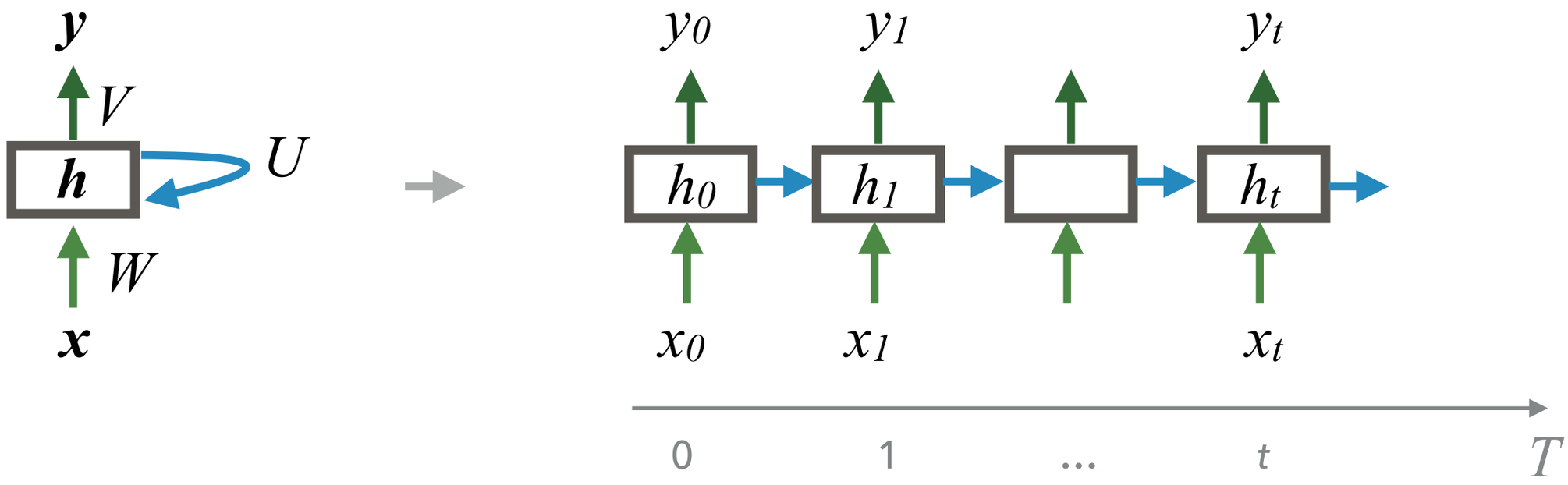
NEURAL NETWORKS: LIMITATIONS

- Feedforward neural network has several limitations due to its architecture:
  - accepts a **fixed-sized** vector as input (e.g. an image)
  - produces a **fixed-sized** vector as output (e.g. probabilities of different classes)
  - performs such input-output mapping using a fixed amount of computational steps (e.g. the number of layers).
- These limitations make it really hard to model time series problems when input and output are real-valued sequences

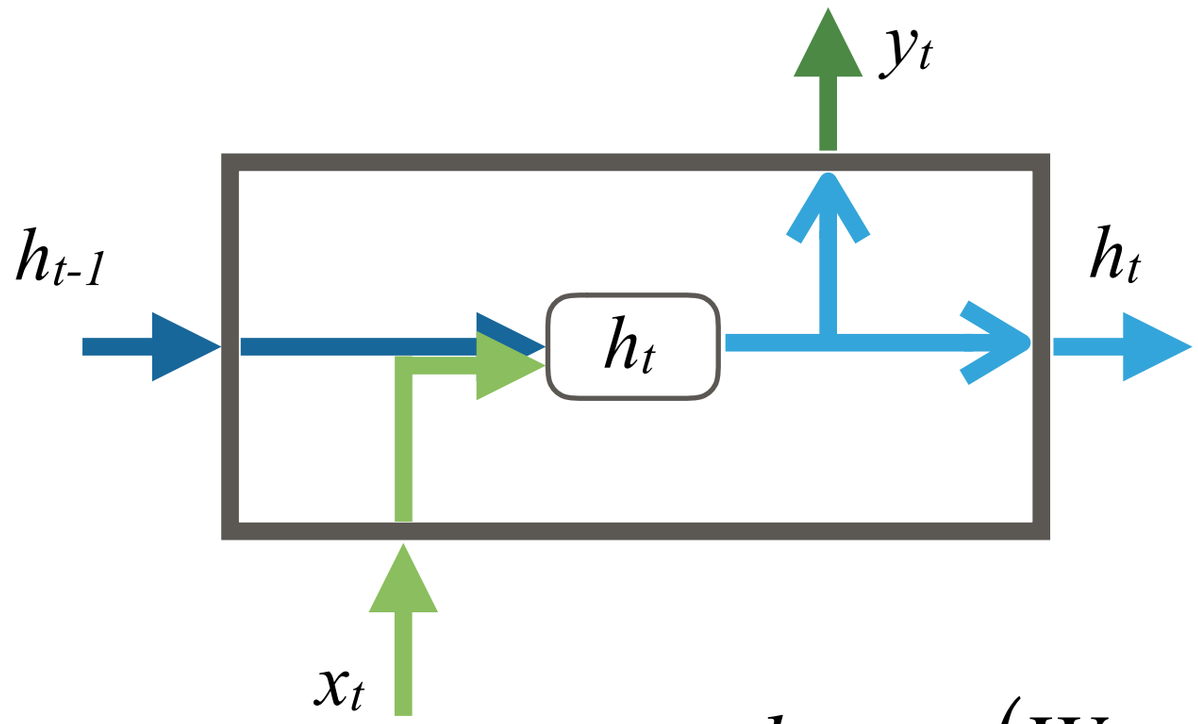
RECURRENT NEURAL NETWORKS

RECURRENT NEURAL NETWORKS: INTUITION

- **Recurrent neural network** (RNN) is a neural network model proposed in the 80's for modelling time series.
- The structure of the network is similar to feedforward neural network, with the distinction that it allows a **recurrent hidden state** whose activation at each time is dependent on that of the previous time (cycle).



RECURRENT NEURAL NETWORKS: SIMPLE RNN



$$h_t = \sigma\left(Wx_t + Uh_{t-1}\right)$$
$$y_t = f\left(Vh_t\right)$$

- The time recurrence is introduced by relation for hidden layer activity h_t with its past hidden layer activity h_{t-1}.
- This dependence is **nonlinear** because of using a logistic function.

RECURRENT NEURAL NETWORKS: BPTT

The unfolded recurrent neural network can be seen as a deep neural network, except that the recurrent weights are tied. To train it we can use a modification of the BP algorithm that works on sequences in time - **backpropagation through time (BPTT).**

- **For each training epoch:** start by training on shorter sequences, and then train on progressively longer sequences until the length of max sequence (1, 2 … N-1, N).
- **For each length of sequence k:** unfold the network into a normal feedforward network that has k hidden layers.
- **Proceed with a standard BP algorithm.**

RECURRENT NEURAL NETWORKS: TRUNCATED BPTT

One of the main problems of BPTT is the high cost of a single parameter update, which makes it impossible to use a large number of iterations.

- For instance, the gradient of an RNN on sequences of length 1000 costs the equivalent of a forward and a backward pass in a neural network that has 1000 layers.

Truncated BPTT processes the sequence one timestep at a time, and every T_1 timesteps it runs BPTT for T_2 timesteps, so a parameter update can be cheap if T_2 is small.

Truncated backpropagation is arguably the most practical method for training RNNs

Read more: http://www.cs.utoronto.ca/~ilya/pubs/ilya_sutskever_phd_thesis.pdf

RECURRENT NEURAL NETWORKS: PROBLEMS

While in principle the recurrent network is a simple and powerful model, in practice, it is unfortunately hard to train properly.

- **PROBLEM:** In the gradient back-propagation phase, the **gradient** signal **multiplied a large number of times** by the *weight matrix* associated with the recurrent connection.
  - If $|\lambda_1| < 1$ (weights are small) => **Vanishing gradient problem**
  - If $|\lambda_1| > 1$ (weights are big) => **Exploding gradient problem**
- **SOLUTION:**
  - Exploding gradient problem => clipping the norm of the exploded gradients when it is too large
  - Vanishing gradient problem => relax nonlinear dependency to liner dependency => LSTM, GRU, etc.

Read more: Razvan Pascanu et al. http://arxiv.org/pdf/1211.5063.pdf

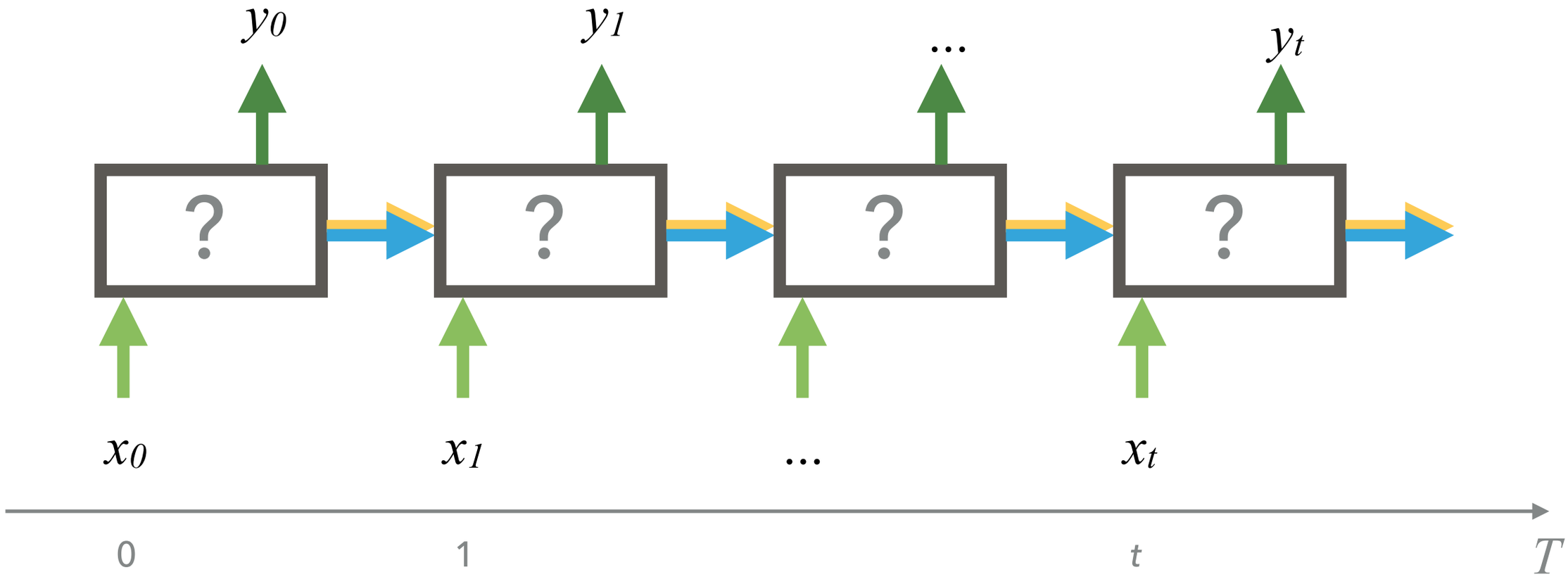
LONG SHORT-TERM MEMORY

LONG SHORT-TERM MEMORY (LSTM)

- Proposed by Hochreiter & Schmidhuber (1997) and since then has been modified by many researchers.
- The LSTM architecture consists of a set of recurrently connected subnets, known as **memory blocks**.
- Each memory block consists of:
  - **memory cell** - stores the state
  - **input gate** - controls what to learn
  - **forget get** - controls what to forget
  - **output gate** - controls the amount of content to modify
- Unlike the traditional recurrent unit which overwrites its content each timestep, the LSTM unit is able to decide whether to keep the existing memory via the introduced gates

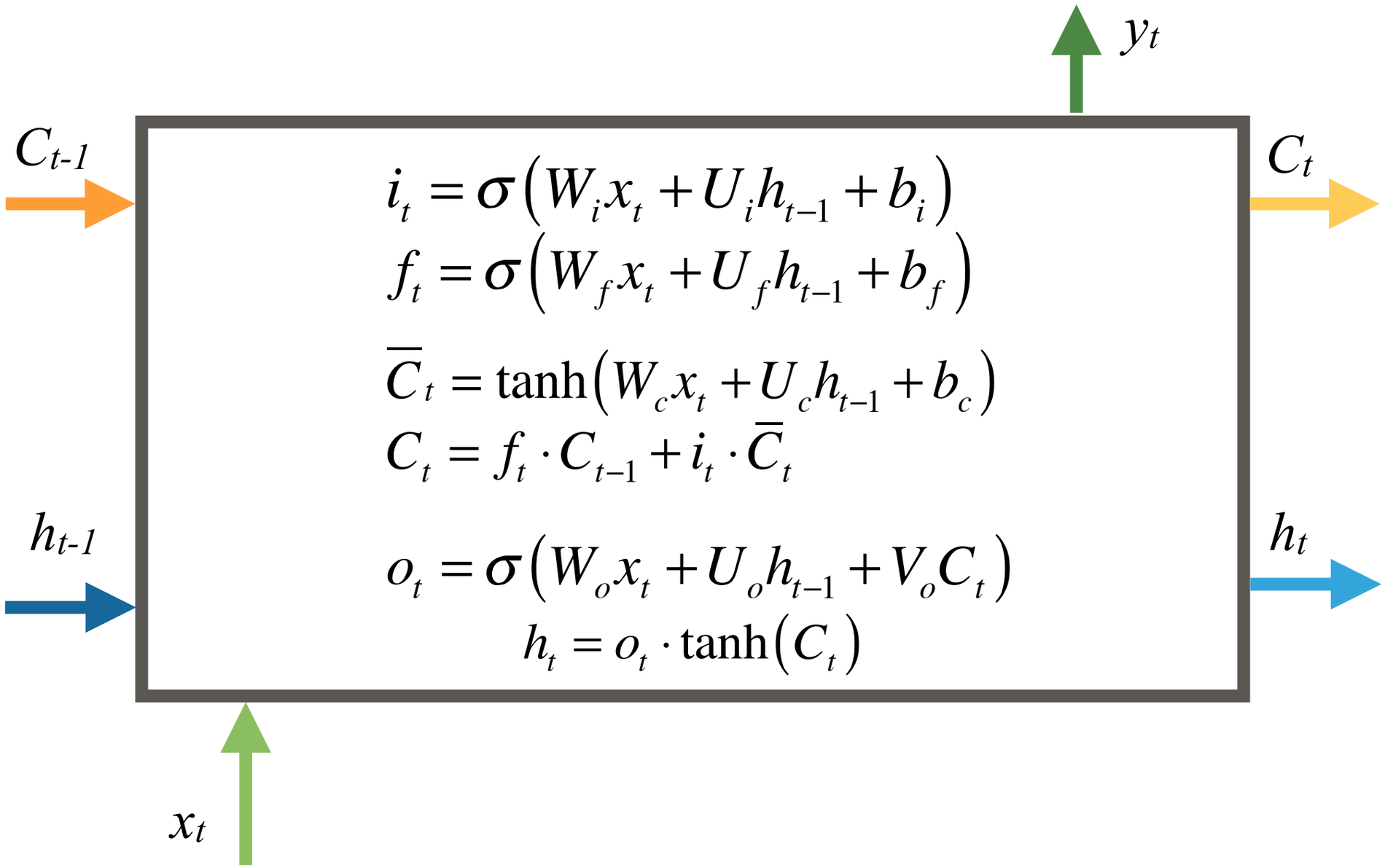
LSTM

The basic unit in the hidden layer of an LSTM is the memory block that replaces the hidden units in a "traditional" RNN



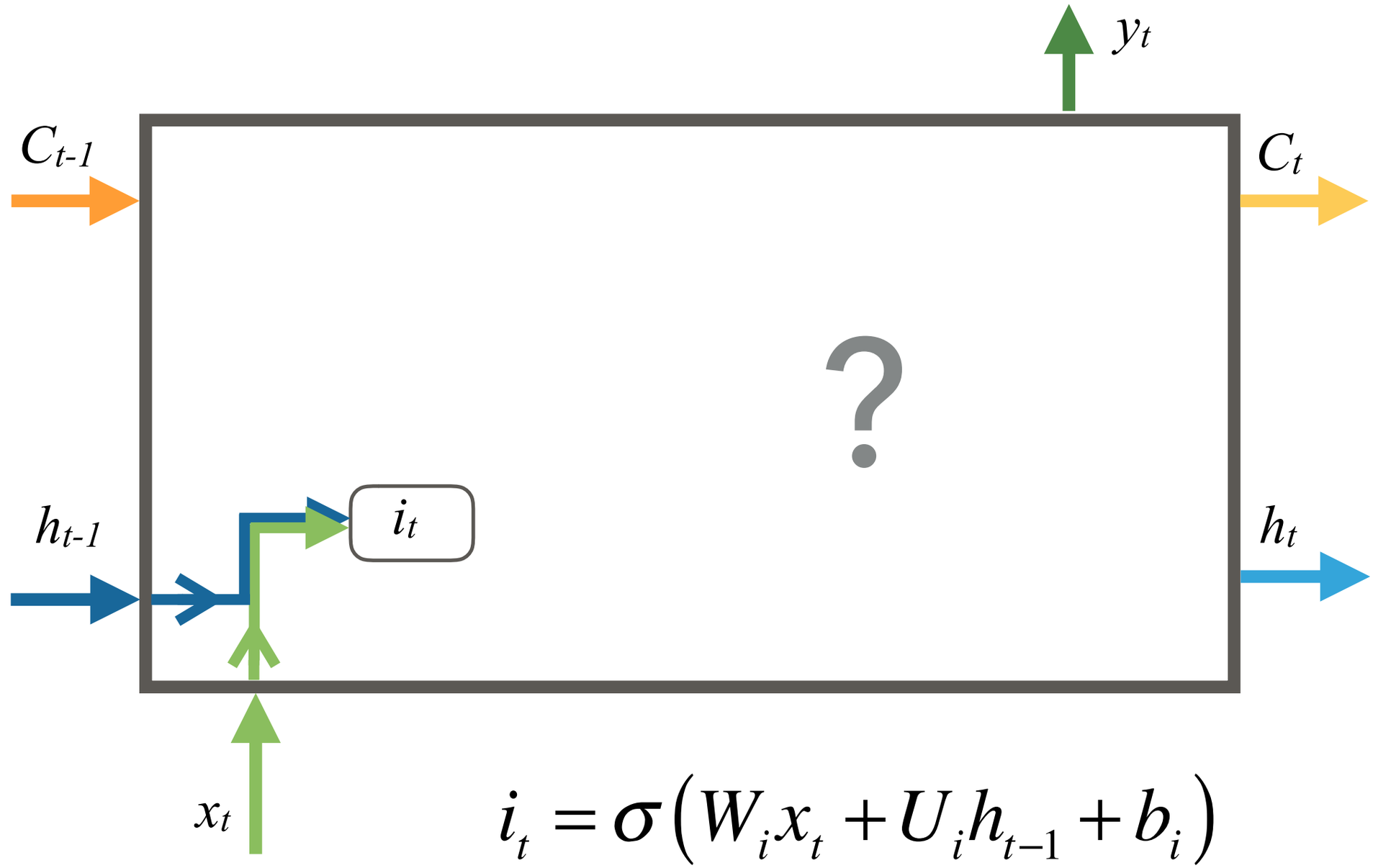
- Model parameters:
  - x_t is the input at time t
  - Weight matrices: $W_i, W_f, W_c, W_o, U_i, U_f, U_c, U_o, V_o$
  - Bias vectors: b_o, b_f, b_c, b_o

LSTM MEMORY BLOCK



- Memory block is a subnet that allow an LSTM unit to adaptively forget, memorise and expose the memory content.

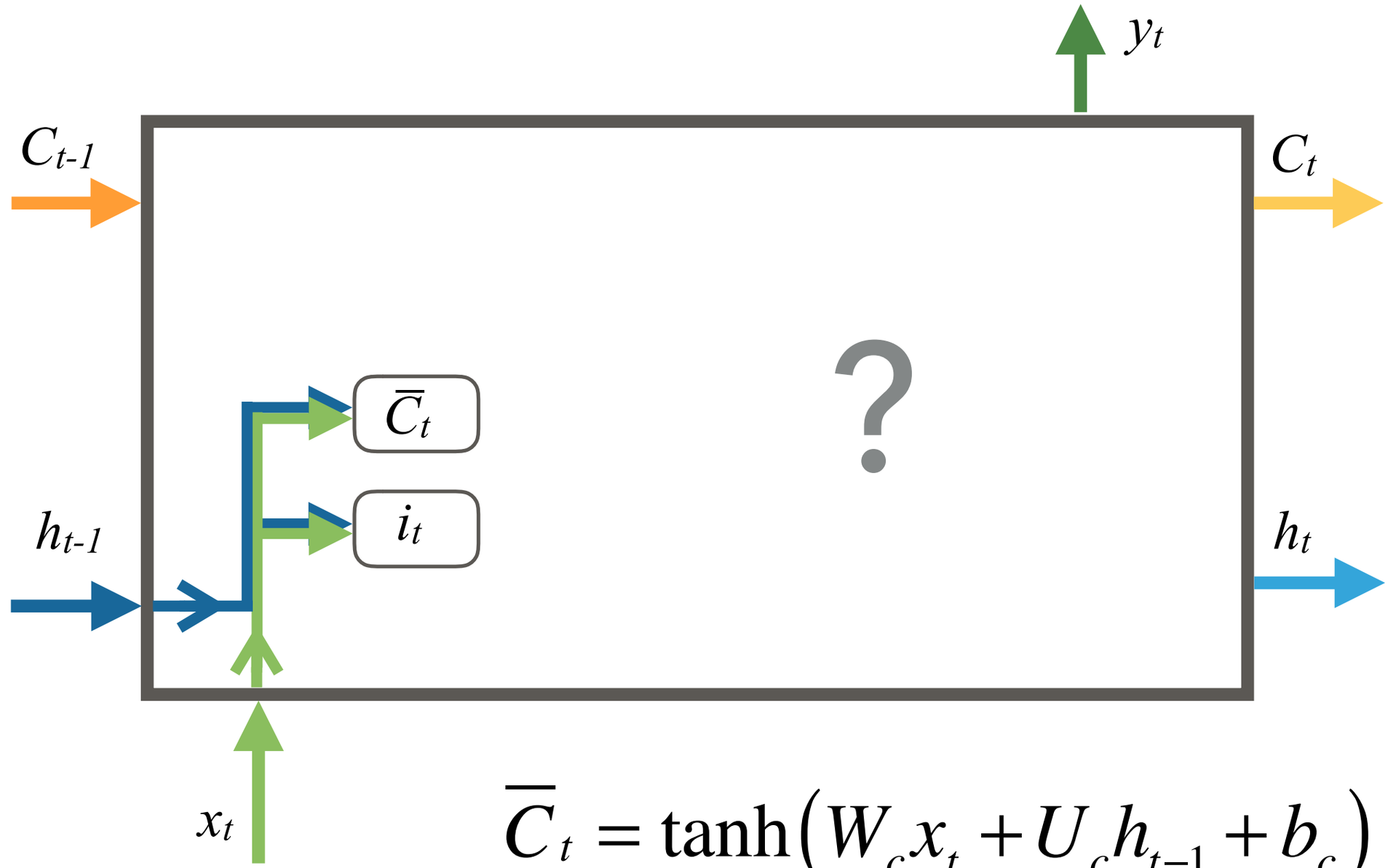
LSTM MEMORY BLOCK: INPUT GATE



$$i_t = \sigma\left(W_i x_t + U_i h_{t-1} + b_i\right)$$

- The input gate i_t controls the degree to which the new memory content is added to the memory cell

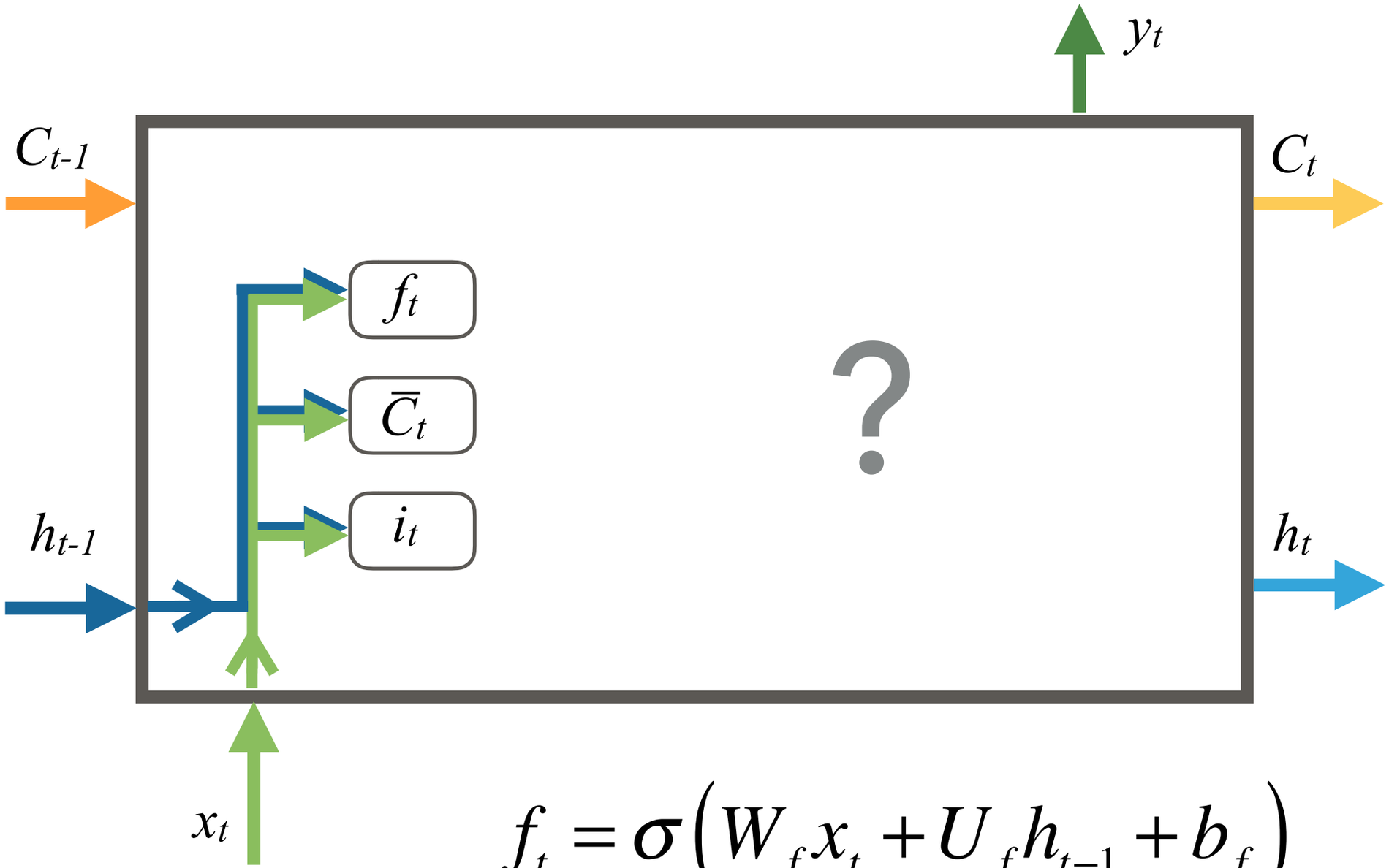
LSTM MEMORY BLOCK: CANDIDATES



$$\overline{C}_t = \tanh\left(W_c x_t + U_c h_{t-1} + b_c\right)$$

- The values $\overline{C}_t$ are candidates for the state of the memory cell (that could be filtered by input gate decision later)

LSTM MEMORY BLOCK: FORGET GATE



$$f_t = \sigma\left(W_f x_t + U_f h_{t-1} + b_f\right)$$

- If the detected feature seems important, the forget gate f_t will be closed and carry information about it across many timesteps, otherwise it can reset the memory content.

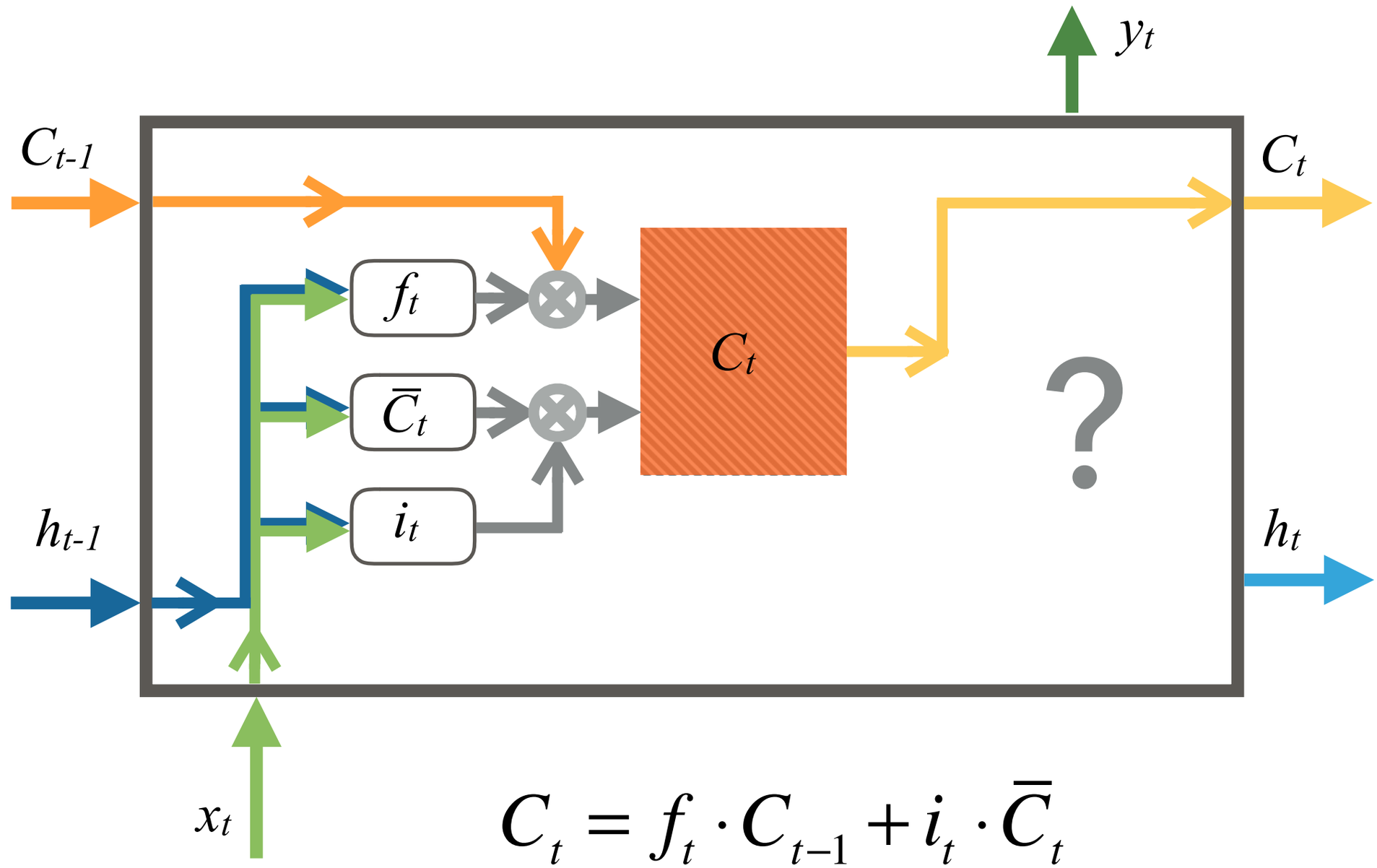
LSTM MEMORY BLOCK: FORGET GATE

- Sometimes it's good to forget.

  If you're analyzing a text corpus and come to the end of a document you may have no reason to believe that the next document has any relationship to it whatsoever, and therefore the memory cell should be reset before the network gets the first element of the next document.

- In many cases by *reset* we don't only mean immediate set it to 0, but also gradual resets corresponding to slowly fading cell states

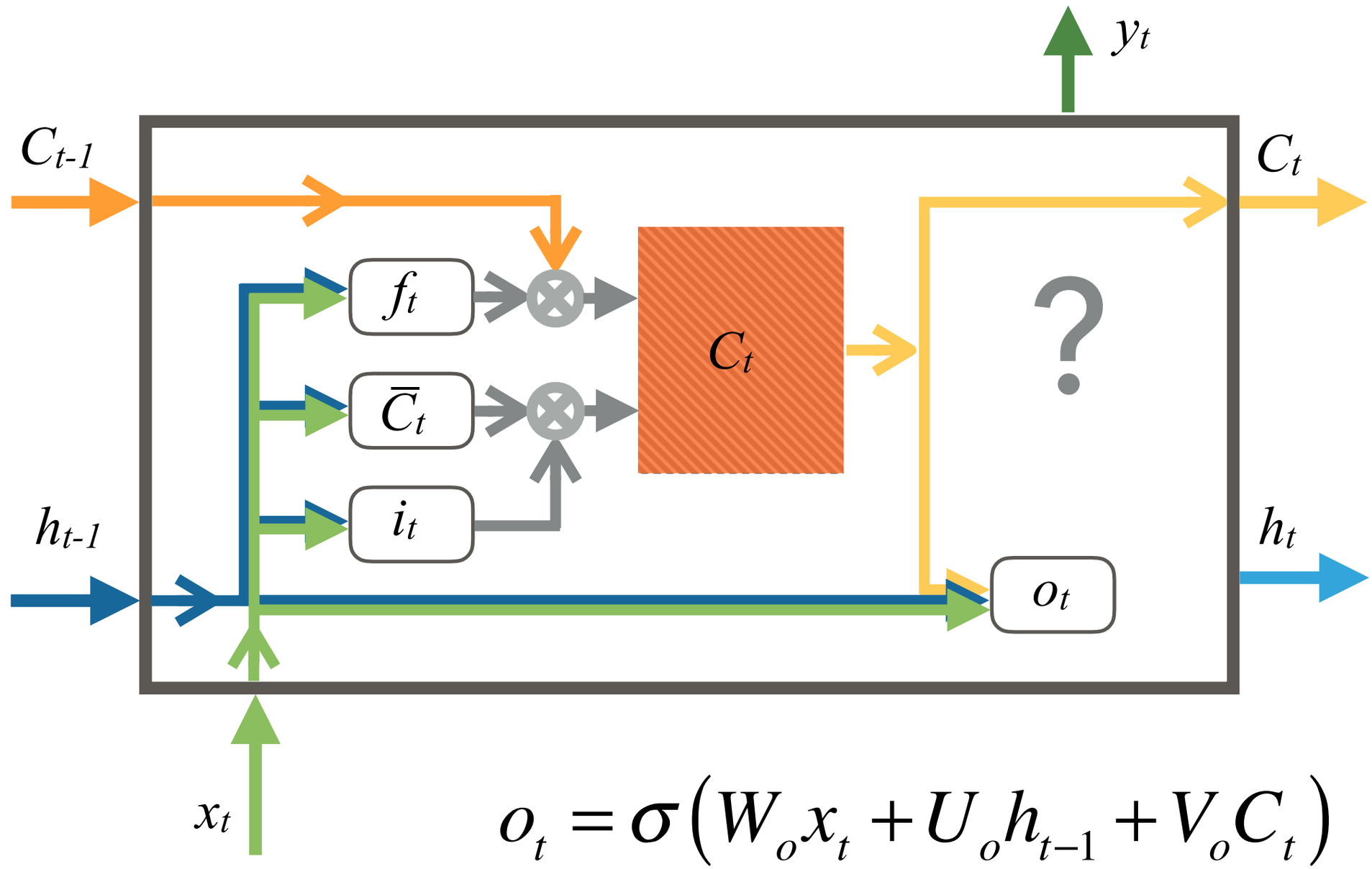
LSTM MEMORY BLOCK: MEMORY CELL



$$C_t = f_t \cdot C_{t-1} + i_t \cdot \bar{C}_t$$

- The new state of the memory cell C_t calculated by partially forgetting the existing memory content C_{t-1} and adding a new memory content $\bar{C}_t$

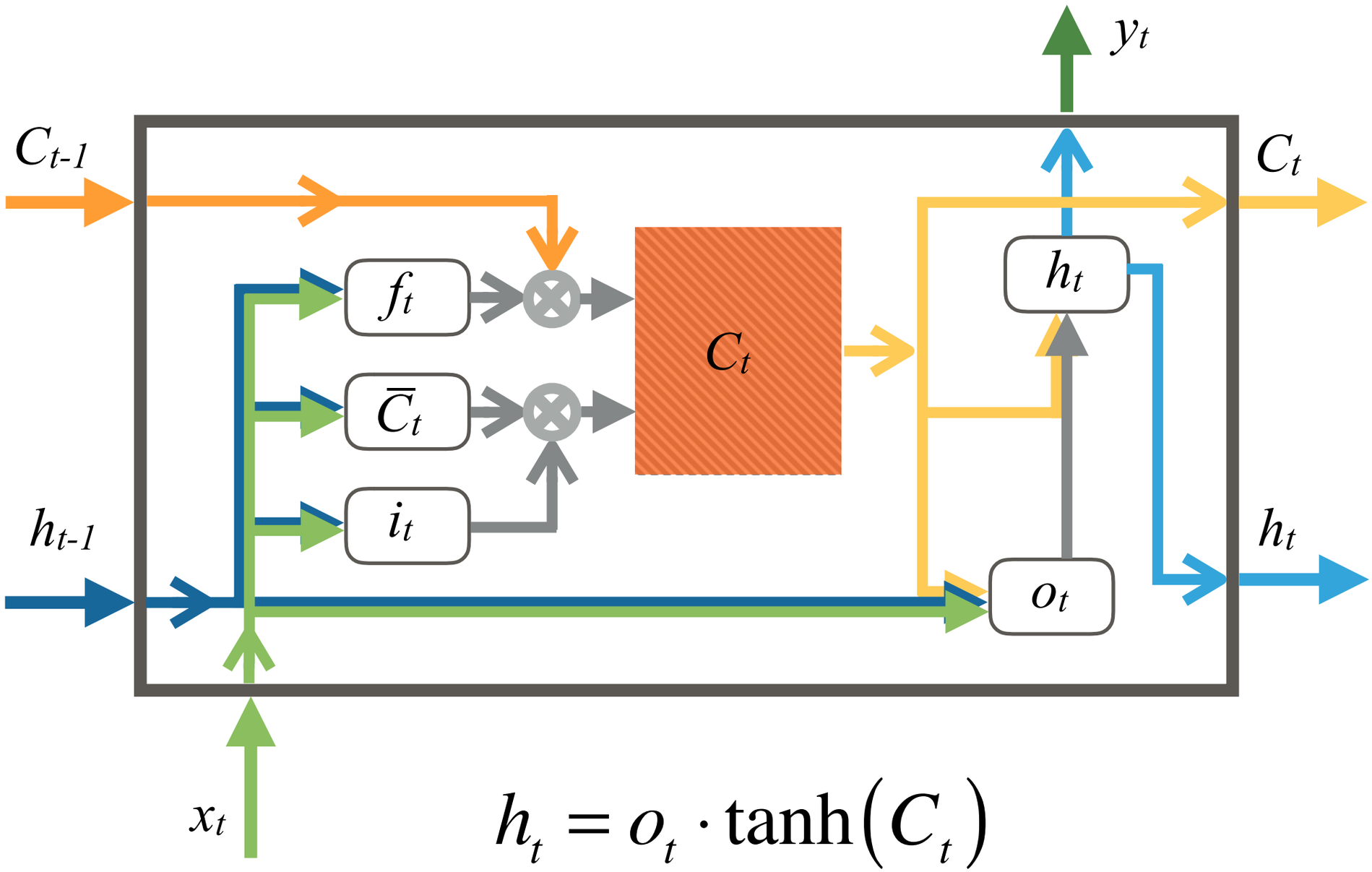
LSTM MEMORY BLOCK: OUTPUT GATE



$$o_t = \sigma\left(W_o x_t + U_o h_{t-1} + V_o C_t\right)$$

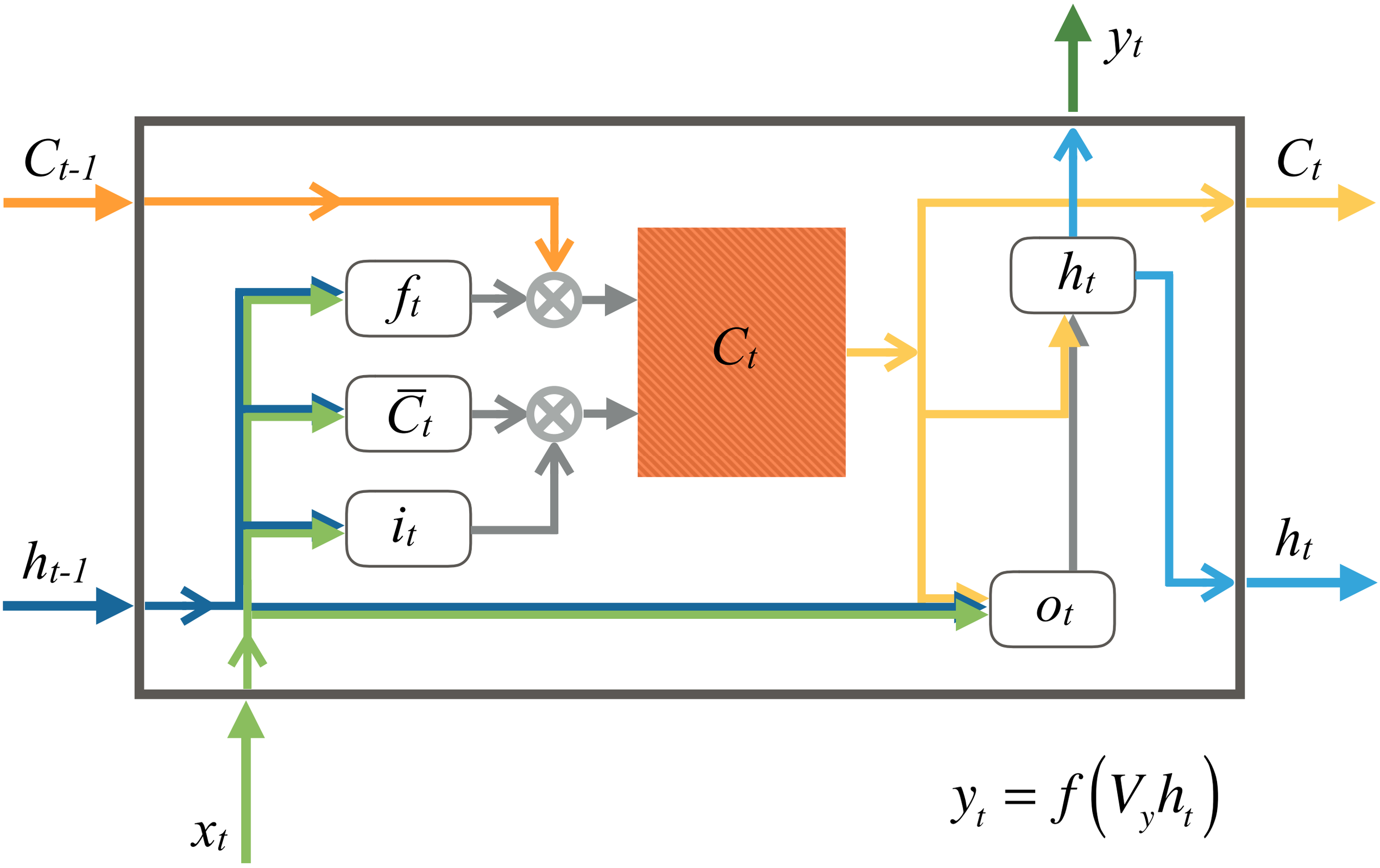
- The output gate o_t controls the amount of the memory content to yield to the next hidden state

LSTM MEMORY BLOCK: HIDDEN STATE



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

LSTM MEMORY BLOCK: ALL TOGETHER



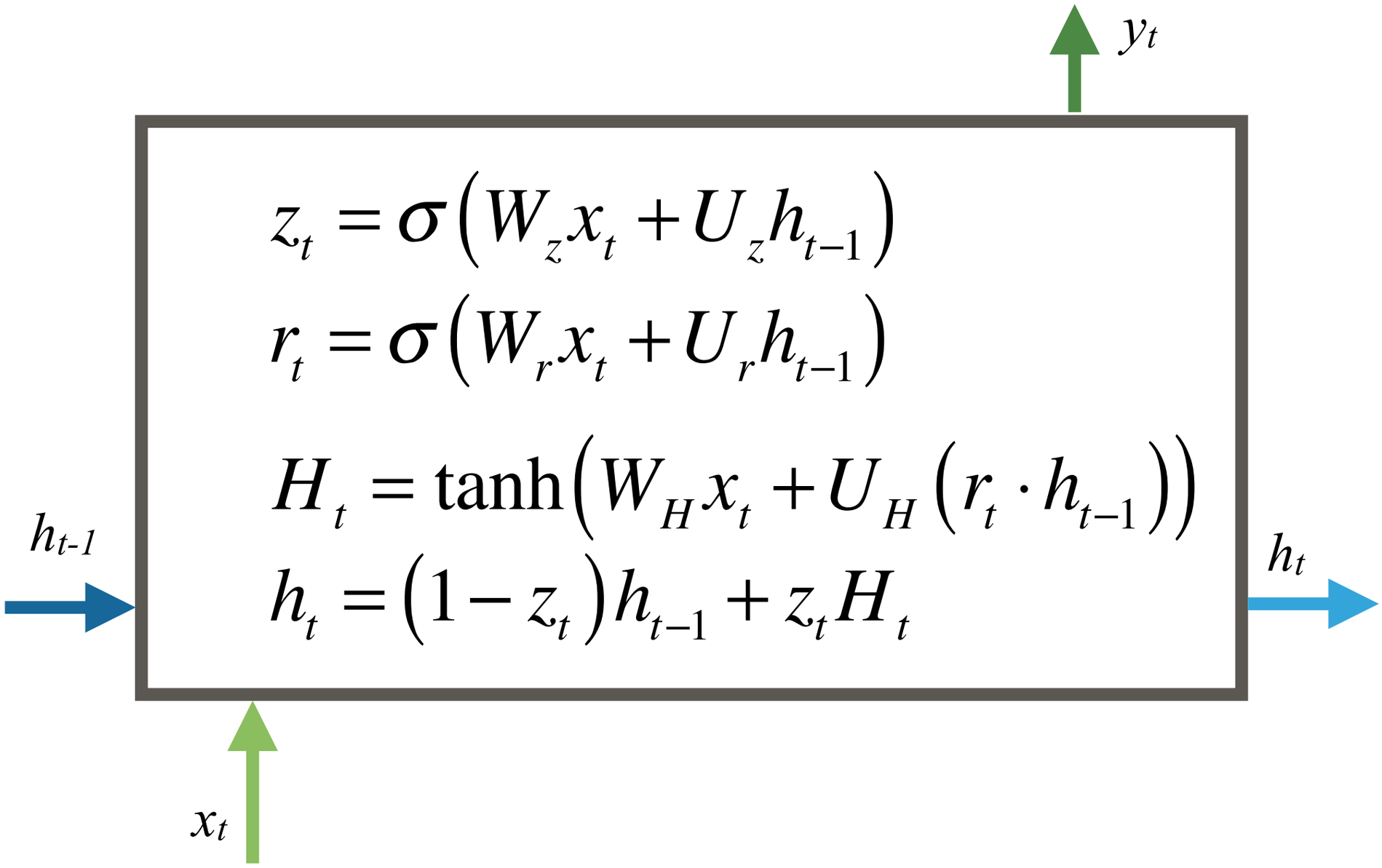
$$y_t = f\left(V_y h_t\right)$$

GATED RECURRENT UNIT

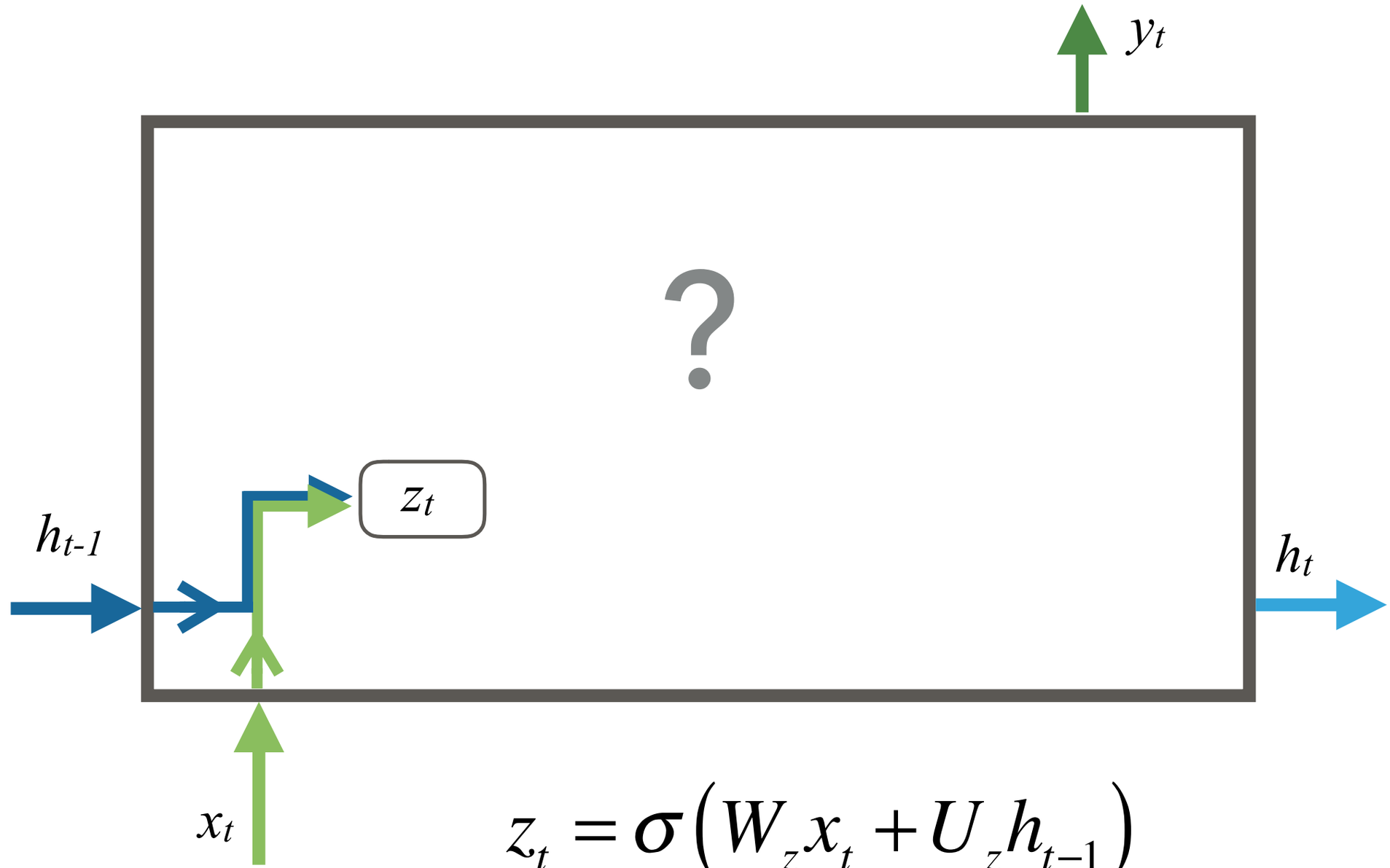
GATED RECURENT UNIT (GRU)

- Proposed by Cho et al. [2014].
- It is similar to LSTM in using gating functions, but differs from LSTM in that it doesn't have a memory cell.
- Each GRU consists of:
  - **update gate**
  - **reset get**
- Model parameters:
  - x_t is the input at time t
  - Weight matrices: $W_z, W_r, W_H, U_z, U_r, U_H$

GRU


y_t
$z_t = \sigma\left(W_z x_t + U_z h_{t-1}\right)$
$r_t = \sigma\left(W_r x_t + U_r h_{t-1}\right)$
$H_t = \tanh\left(W_H x_t + U_H\left(r_t \cdot h_{t-1}\right)\right)$
$h_t = \left(1 - z_t\right) h_{t-1} + z_t H_t$
h_{t-1}
h_t
x_t

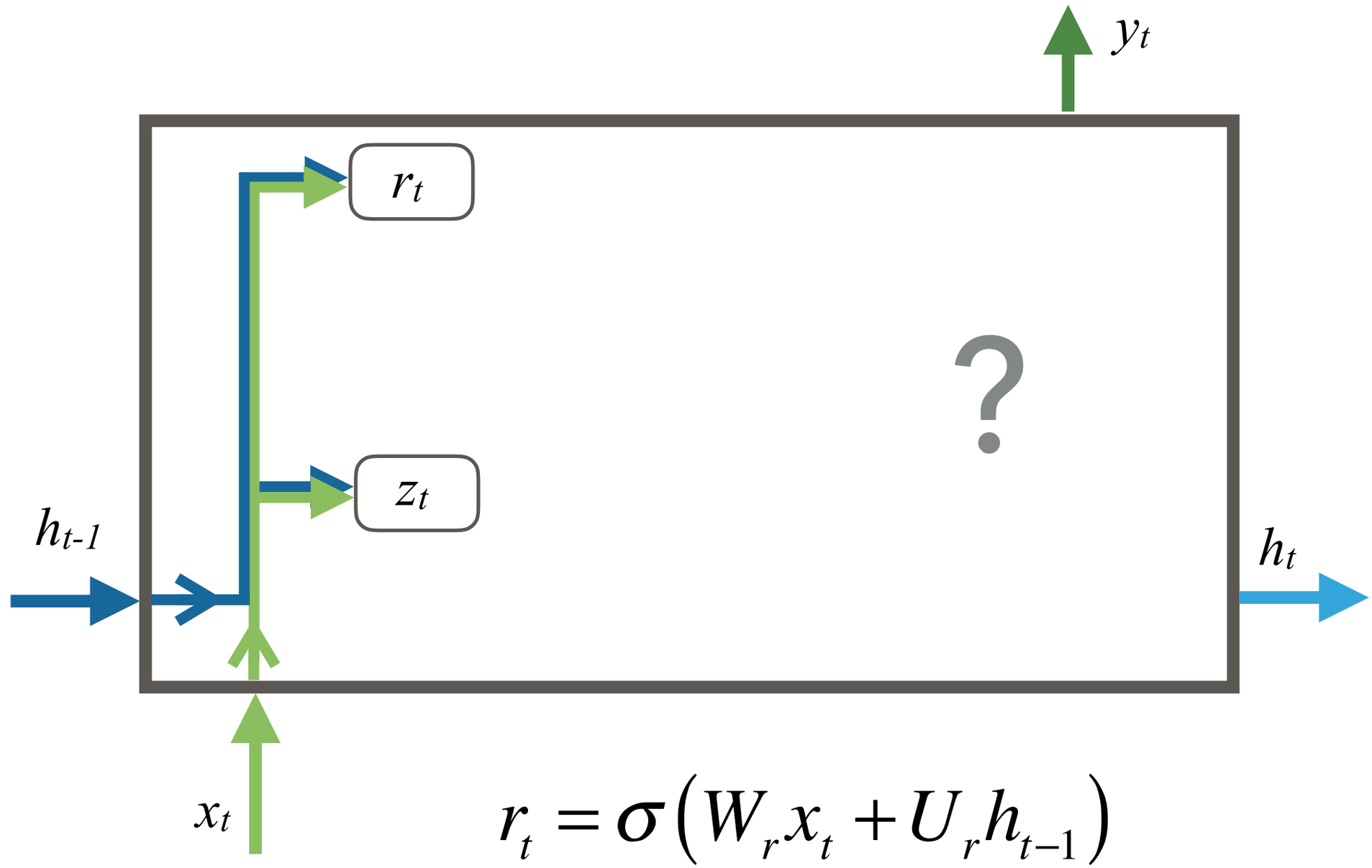
GRU: UPDATE GATE



$$z_t = \sigma\left(W_z x_t + U_z h_{t-1}\right)$$

- Update gate z_t decides how much unit update its activation or content

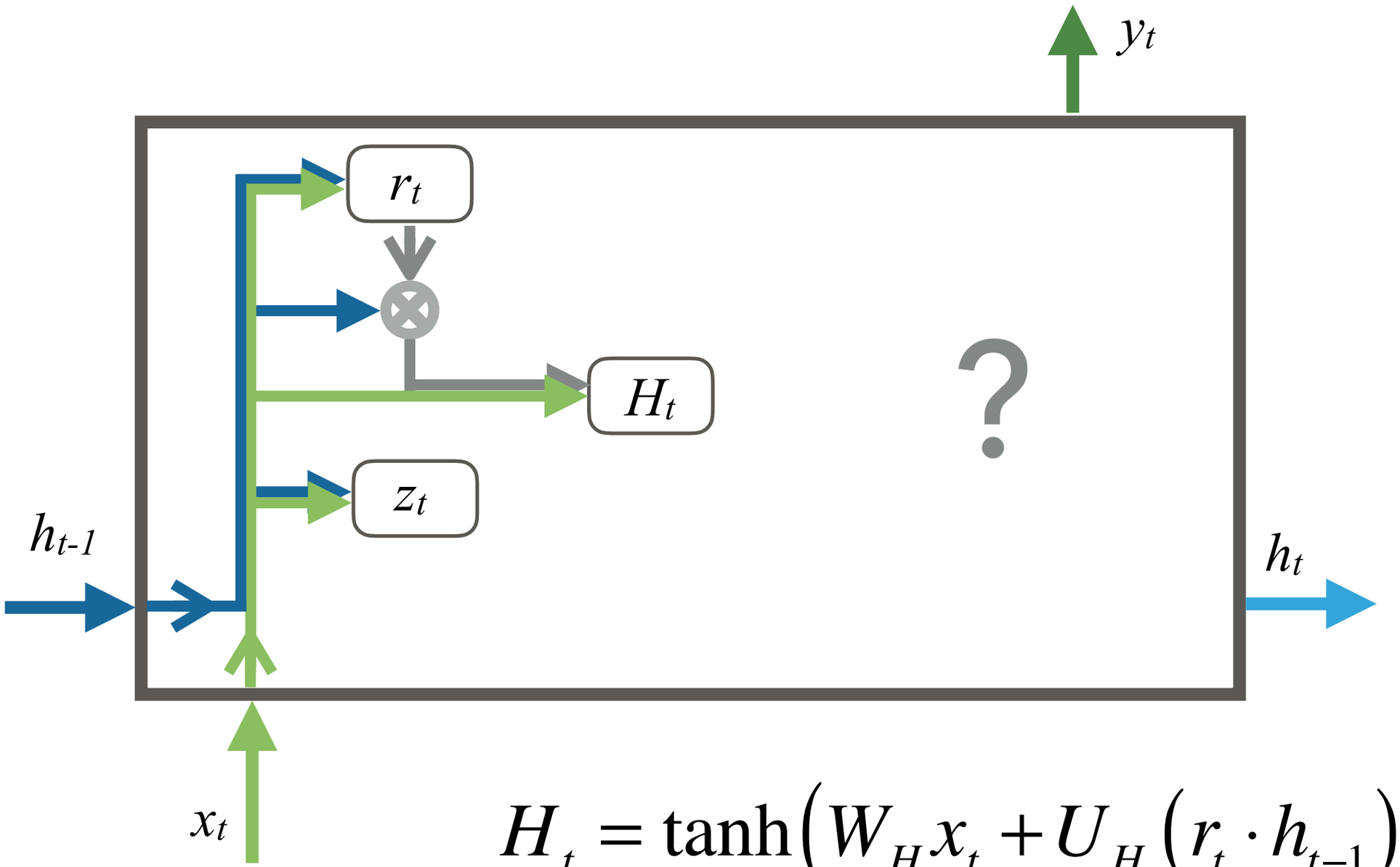
GRU: RESET GATE



$$r_t = \sigma\left(W_r x_t + U_r h_{t-1}\right)$$

- When r_t close to 0 (gate off), it makes the unit act as it's reading the first symbol from the input sequence, allowing it to forget previously computed states

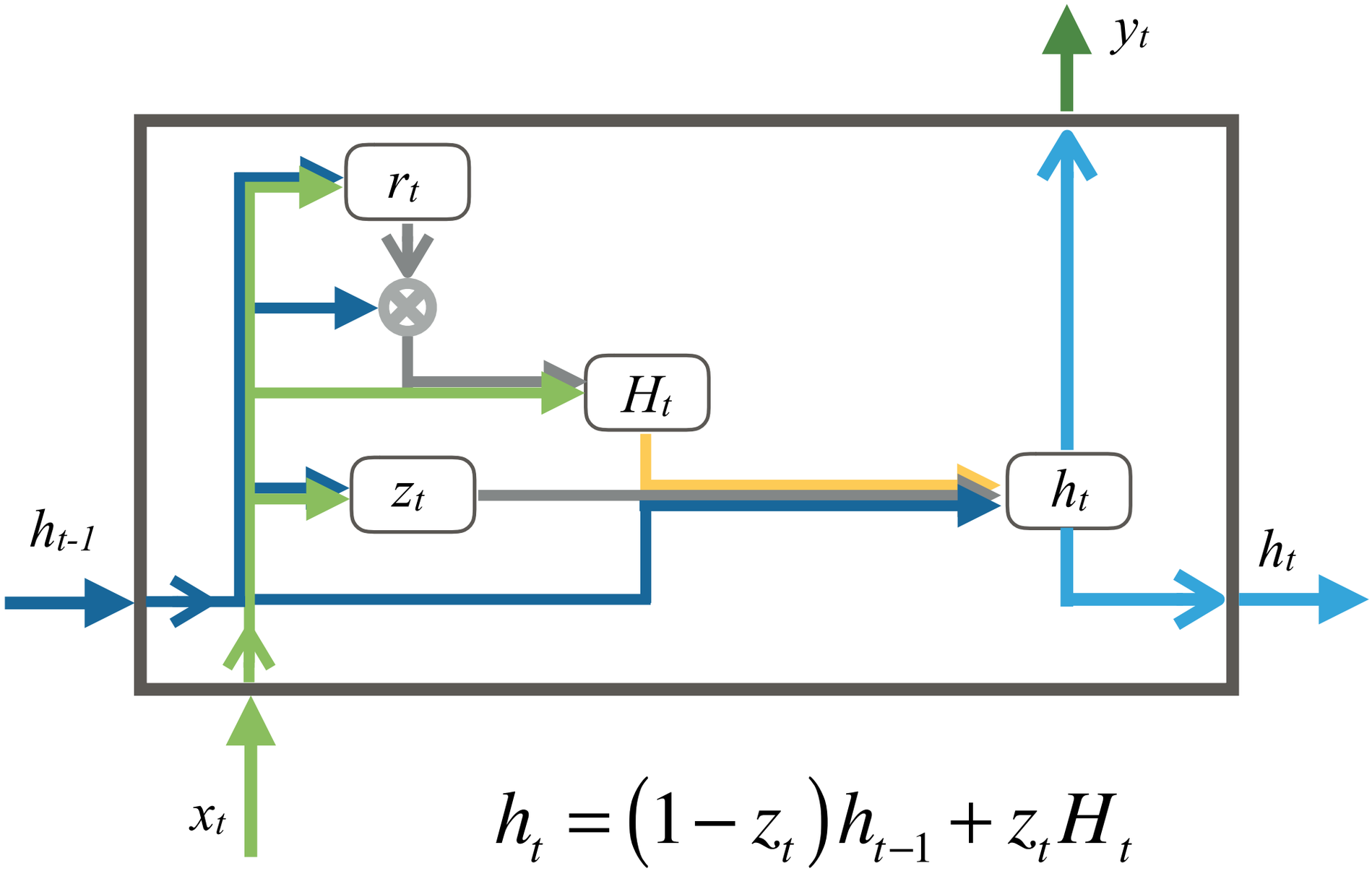
GRU: CANDIDATE ACTIVTION



$$H_t = \tanh\left(W_H x_t + U_H\left(r_t \cdot h_{t-1}\right)\right)$$

- Update gate z_t decides how much unit update its activation or content.

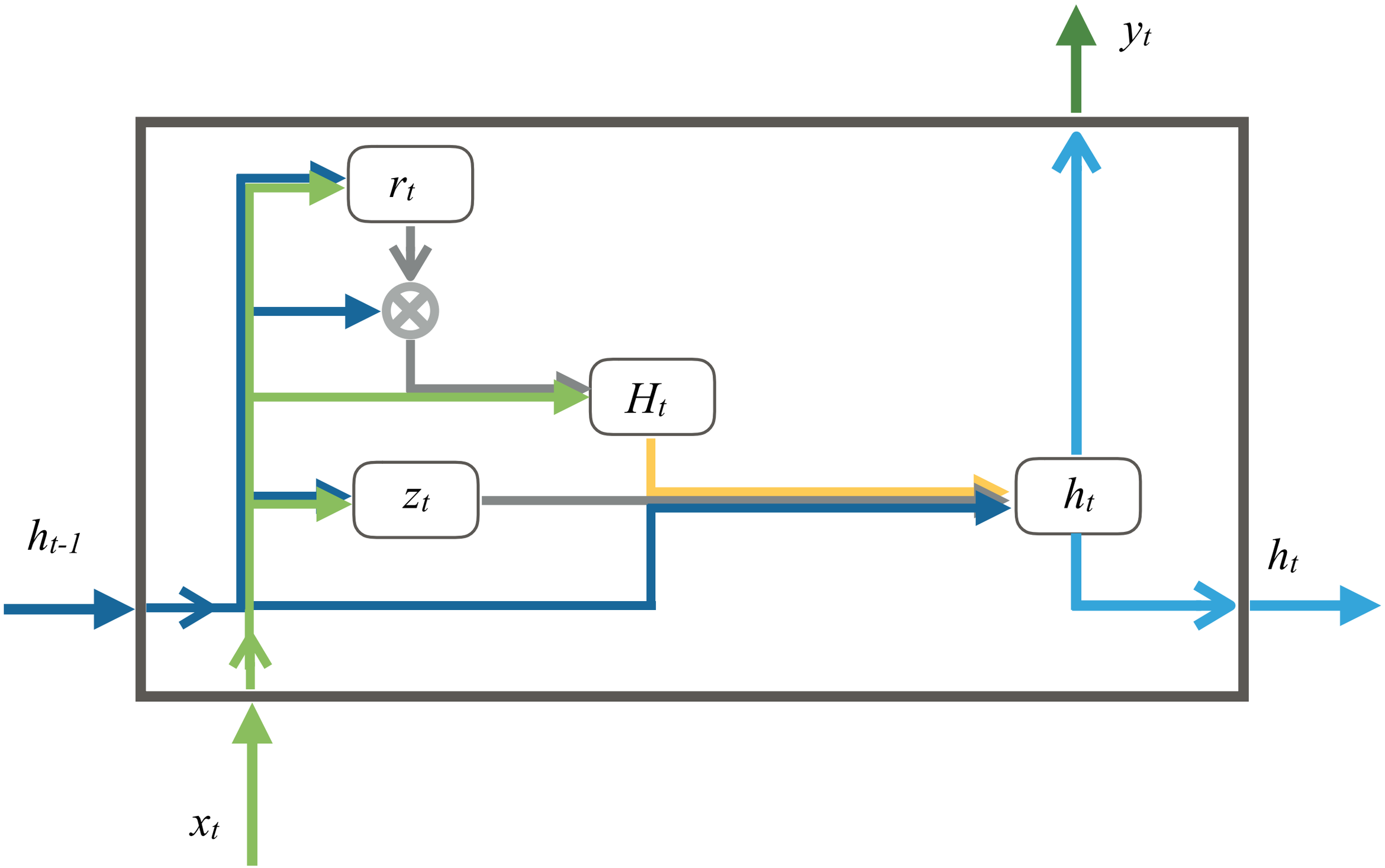
GRU: HIDDEN STATE



$$h_t = (1 - z_t) h_{t-1} + z_t H_t$$

- Activation at time t is the linear interpolation between previous activations h_{t-1} and candidate activation H_t

GRU: ALL TOGETHER



READ MORE

- Supervised Sequence Labelling with Recurrent Neural Networks
  http://www.cs.toronto.edu/~graves/preprint.pdf
- On the difficulty of training Recurrent Neural Networks
  http://arxiv.org/pdf/1211.5063.pdf
- Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling
  http://arxiv.org/pdf/1412.3555v1.pdf
- Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation
  http://arxiv.org/pdf/1406.1078v3.pdf
- Understanding LSTM Networks
  http://colah.github.io/posts/2015-08-Understanding-LSTMs/
- General Sequence Learning using Recurrent Neural Networks
  https://www.youtube.com/watch?v=VINCQghQRuM

END OF PART 1

THANK YOU

- @gakhov
- linkedin.com/in/gakhov
- www.datacrucis.com