Deep Learning Applications for collider physics Lecture 3

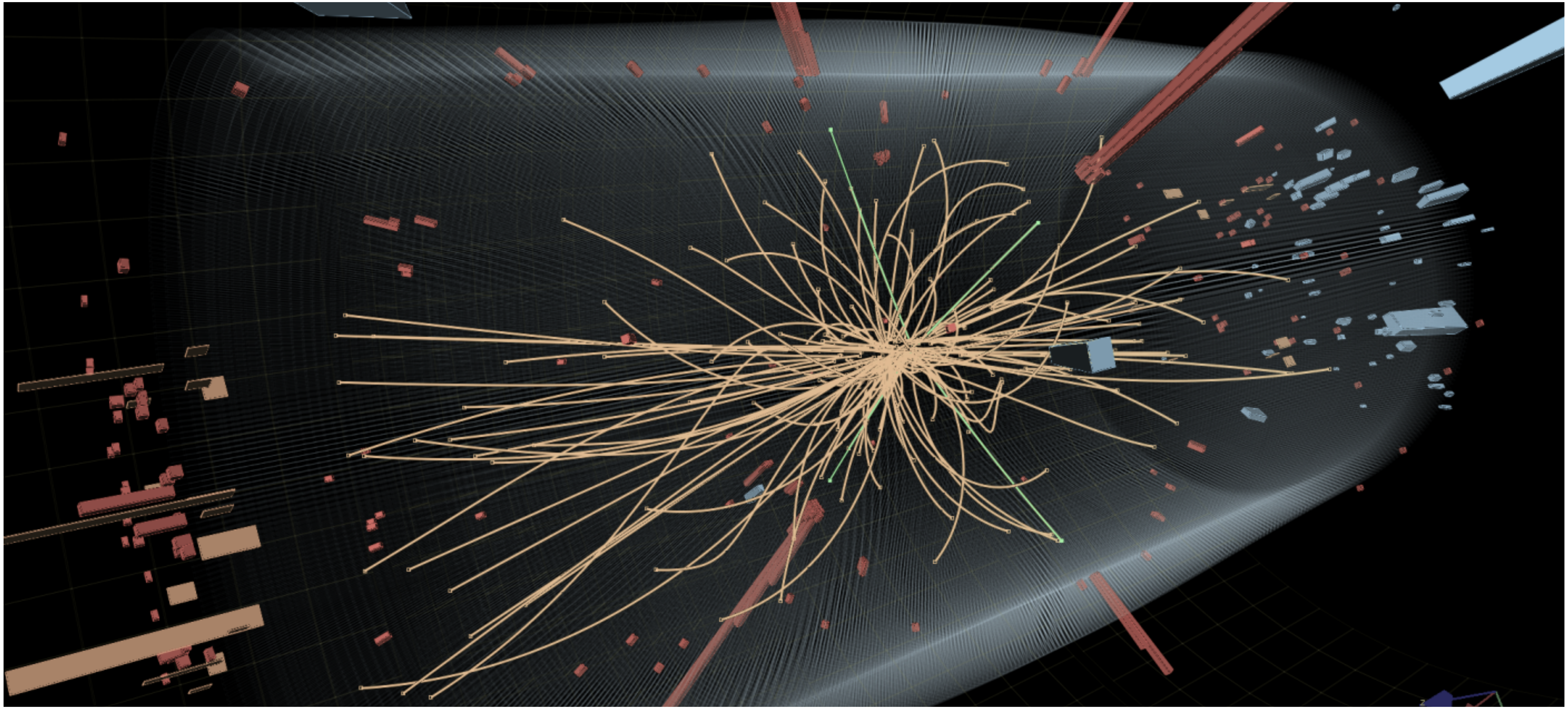
Maurizio Pierini

European Research Council

Plan for this week

	Day1	Day2	Day3	Day4	Day5
Lecture	Introduction	ConvNN	RNNs	Graphs	Unsupervised Learning
Tutorial	Fully Connected Classifier	ConvNN Classifier	RNNs Classifier	Graphs Classifier	Anomaly Detection

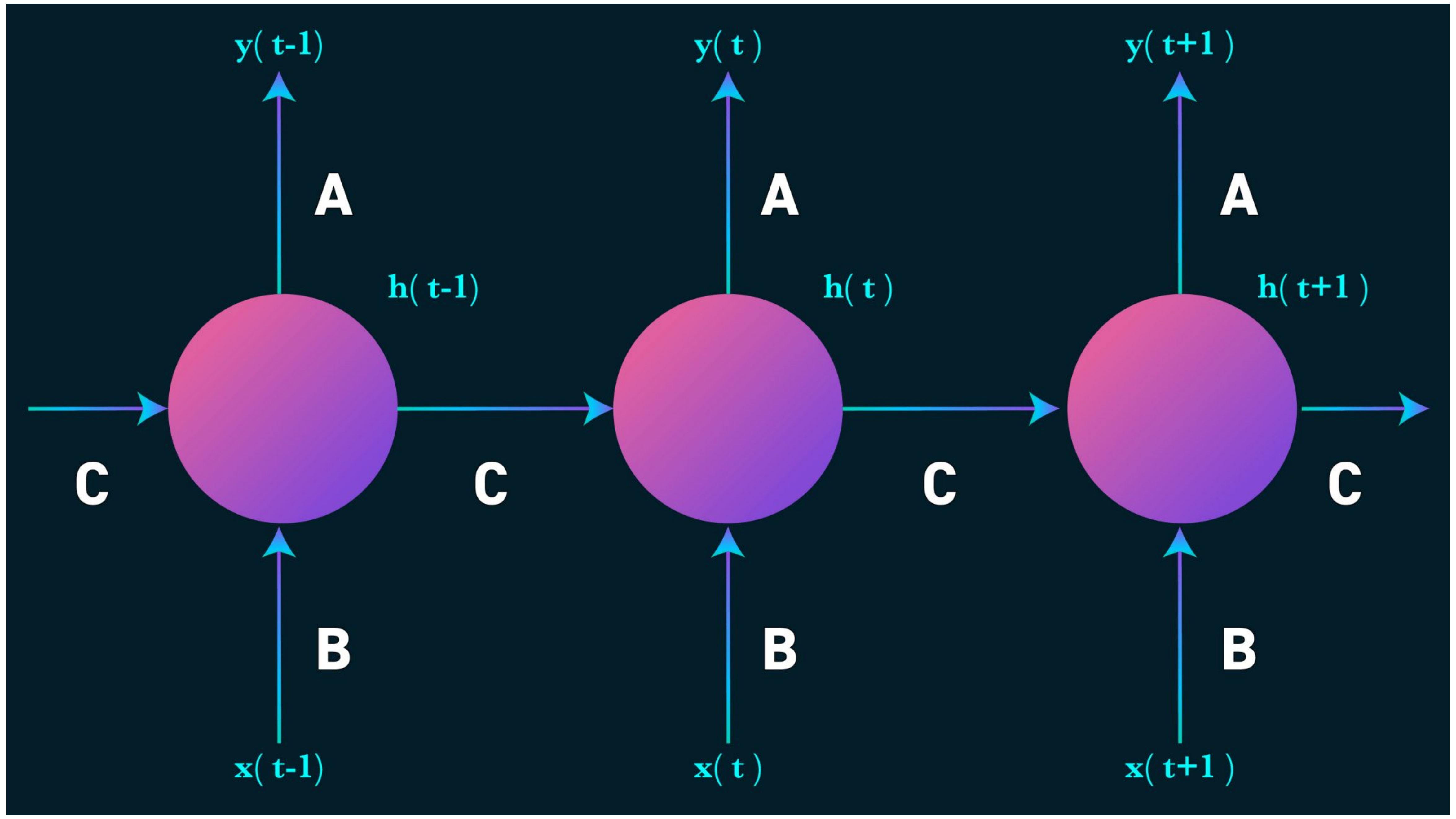
European Research Council



Language processing for particle physics

Recurrent networks

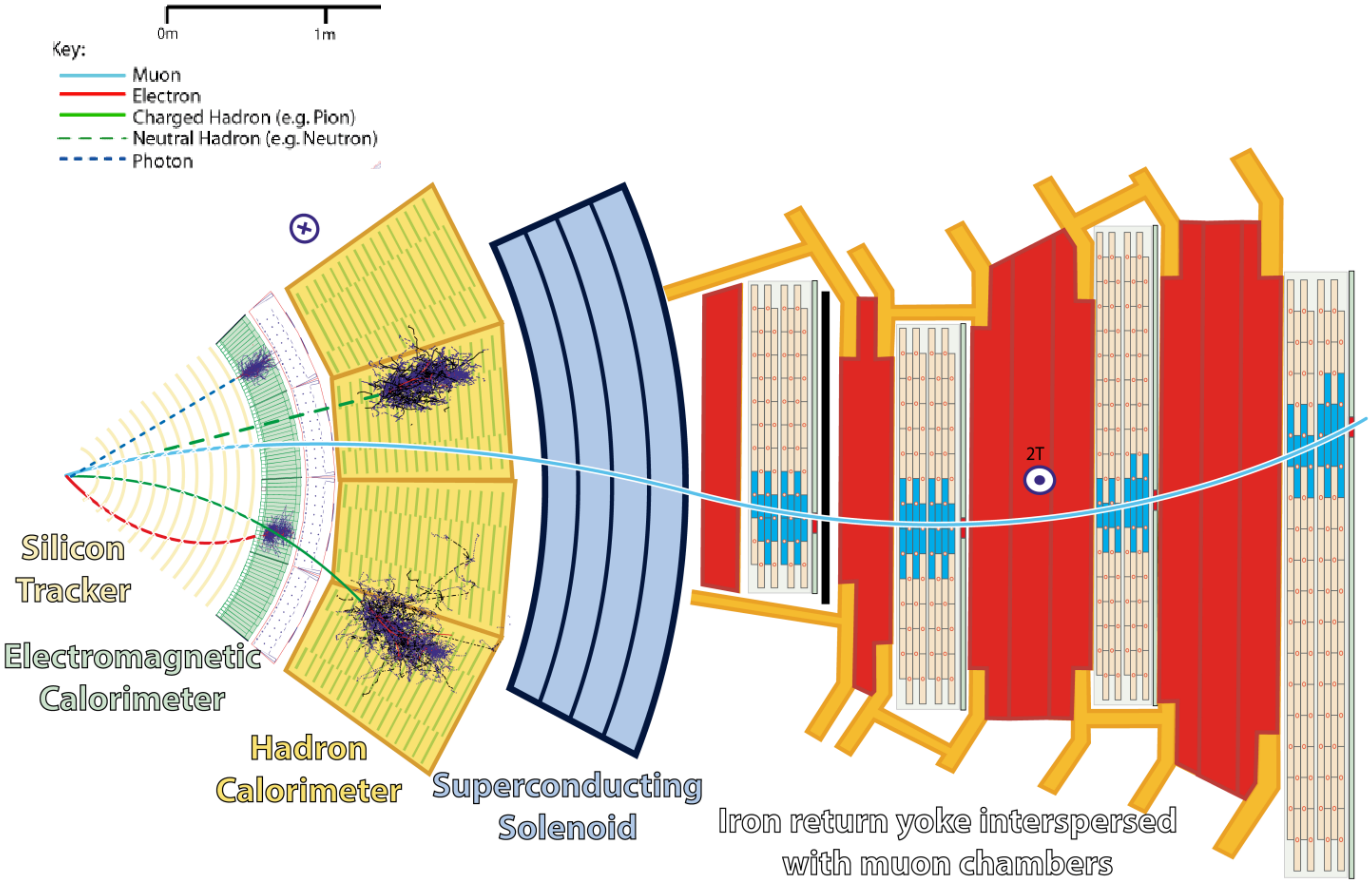
- Recurrent architectures are designed to process sequences of data
- Then idea is to have information flowing in the network while the sequence is sequentially processed
- Through this idea, recurrent networks mimic memory persistence
- Advantages
  - the input is not fixed-sized



https://towardsdatascience.com/understanding-rnn-and-lstm-f7cdf6dfc14e

RNNs for particle physics

- RNN can be used to deal with list of reconstructed particles
- Their architecture better fits our needs
  - No underlying assumption on the detector geometry
- This comes at two costs
  - Some ordering principle needs to be specified
  - Inference is sequential (not ideal for L1 real time)



European Research Council

LHC events & language processing

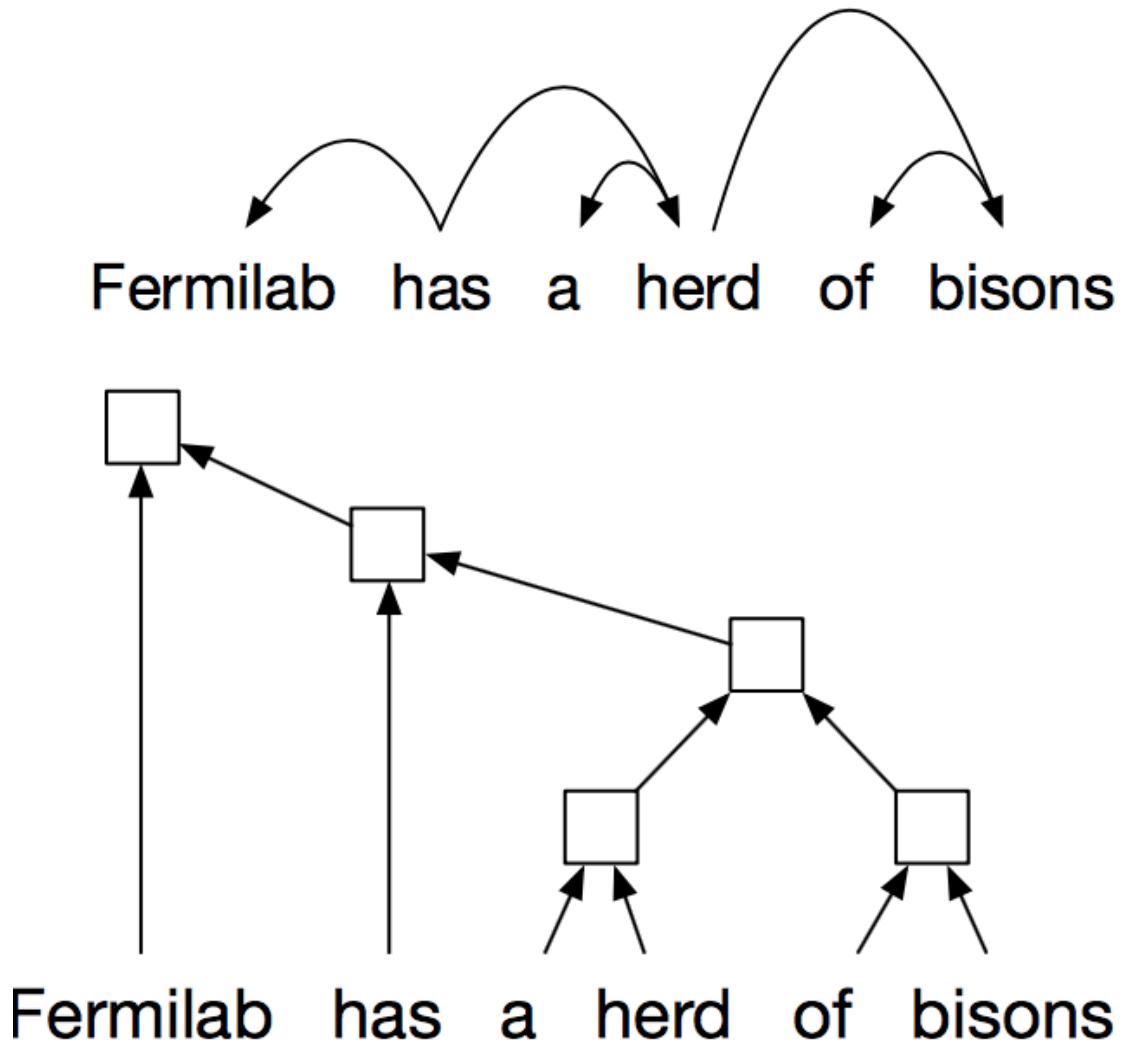
- *RNNs learn sentence meaning by order, which (for a given language) carries a lot of the information about the underlying grammar*

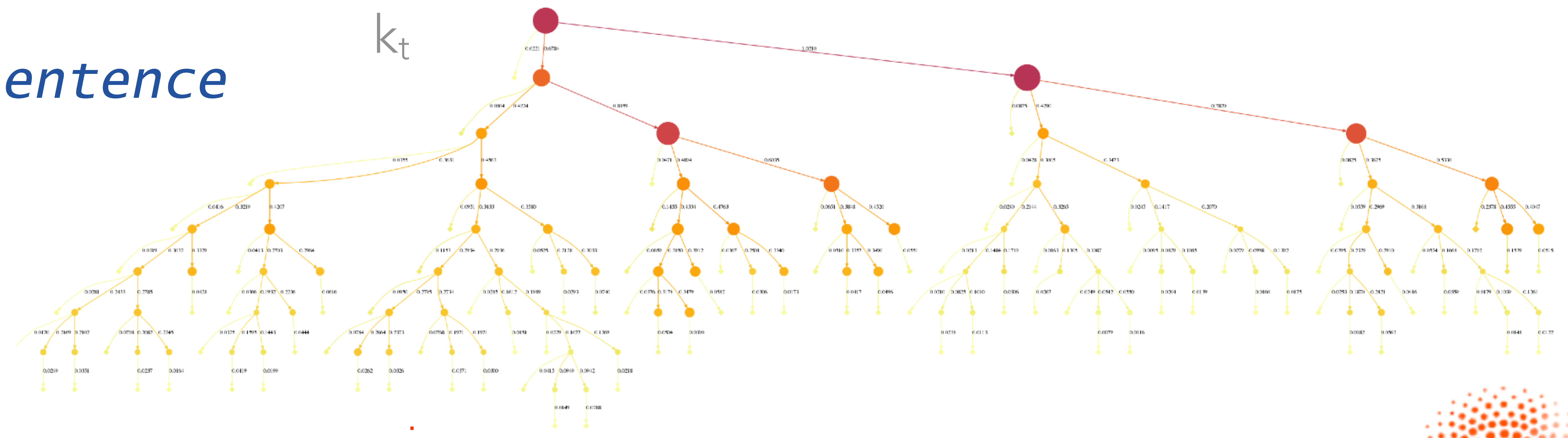
- *One could attempt something similar with physics, e.g., understanding the hadronization processes in jets from a sequence of jet constituents*

- *Which order: not a clear answer. One can use pT ordering, angular ordering, etc. as in parton-shower MC simulation (pythia, sherpa, …) and jet clustering algorithms (kT, anti-kT, etc.)*

  - *particles as words in a sentence*

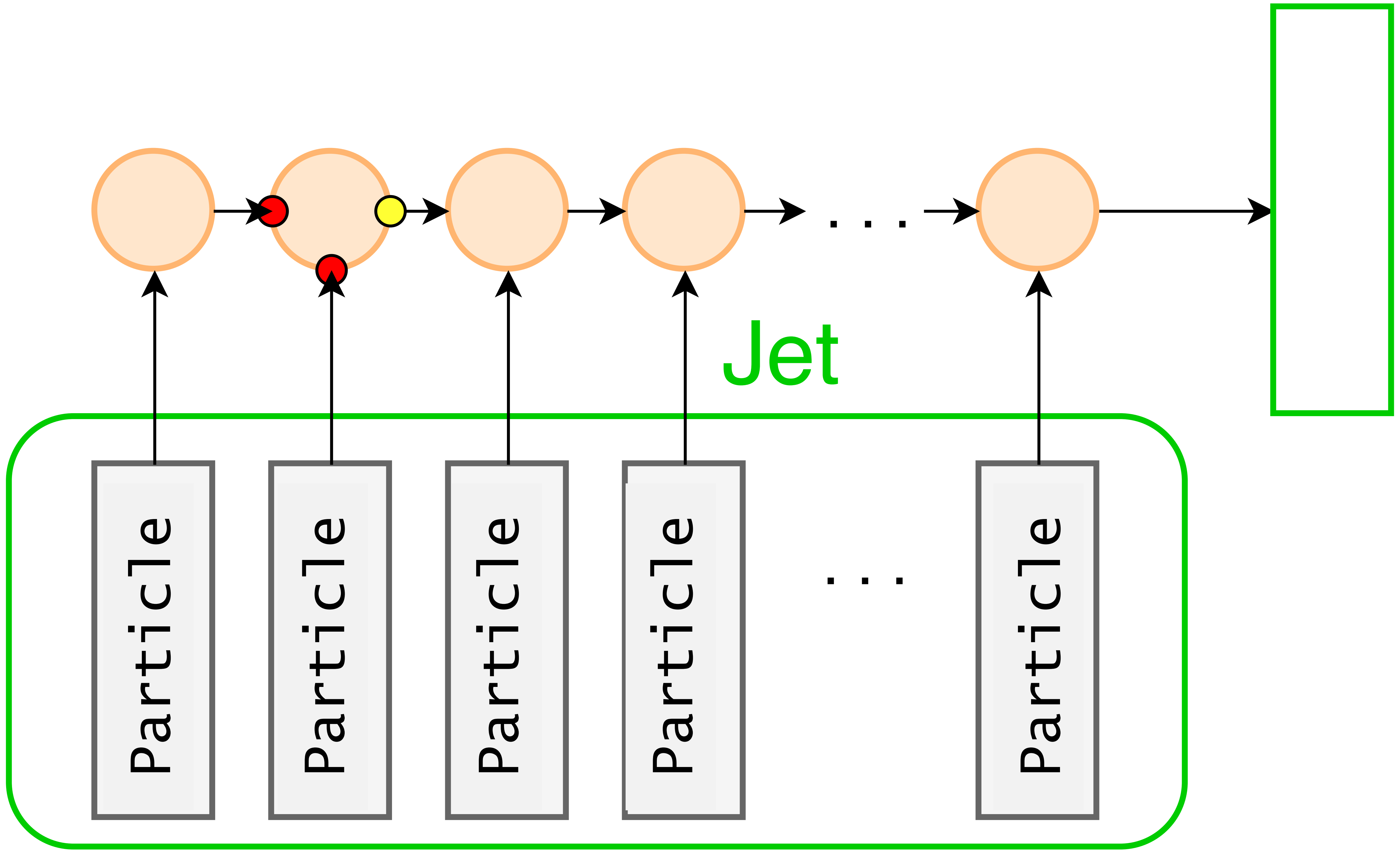
  - *QCD is the grammar*





Recurrent Neural Networks

- Given an ordered list of jet constituents, one could then learn jet features
  - Jet ID, as in our problem
  - Jet substructure quantities, jet kinematic, etc
- One could use the same approach on the whole event, w/o jet clustering (topology classifier)
- Not my best option as of today: graph networks emerged as a more suitable architecture for this (can process unordered sets)



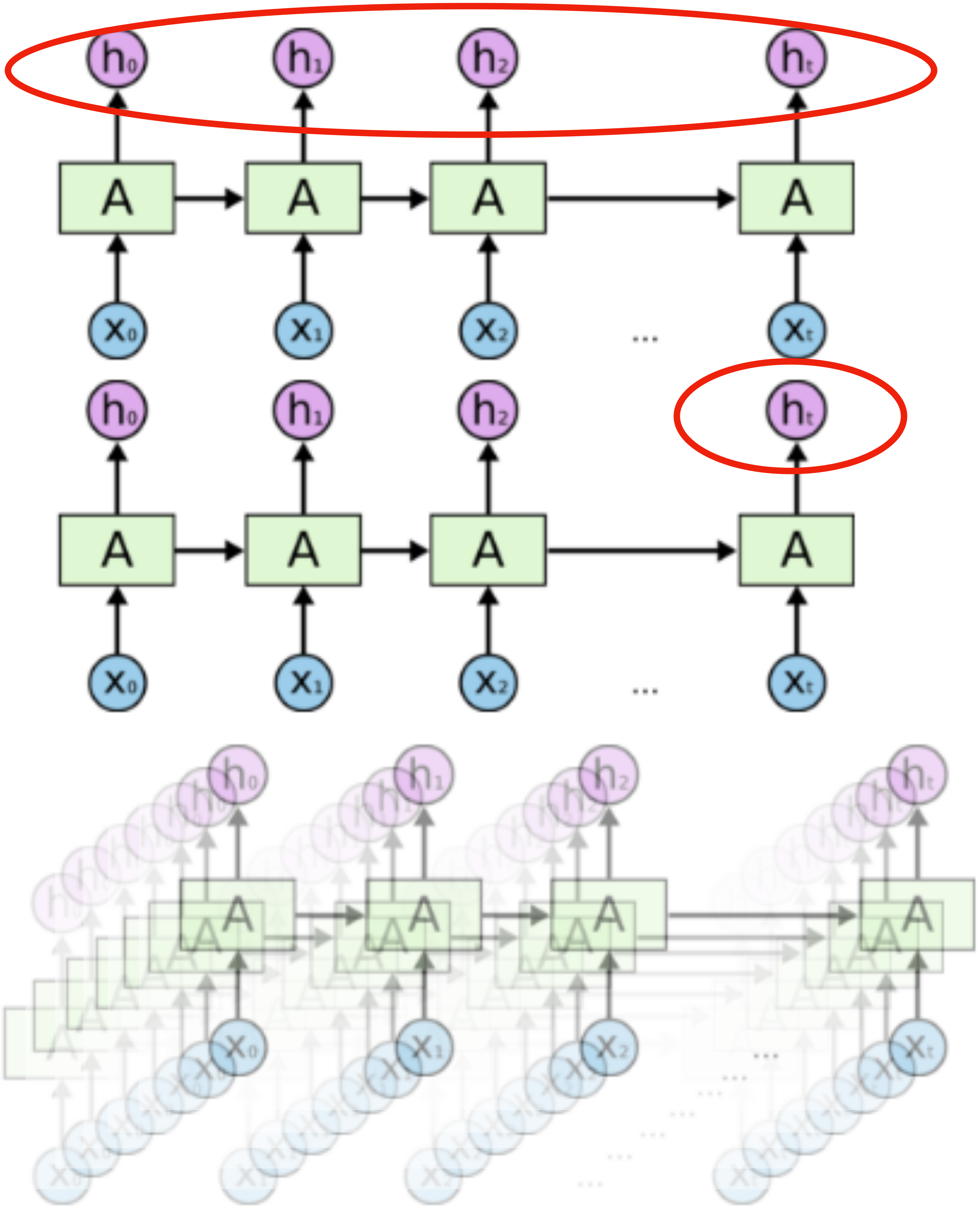
Recurrent networks

- *Recurrent networks can be operated in two ways*
  - *one can inject a sequence and received a sequence*
  - *one could just focus on the result of the last iteration, translating a sequence in a quantity (*)*

- **return_sequences**: Boolean. Whether to return the last output in the output sequence, or the full sequence. Default: `False`.

From Keras

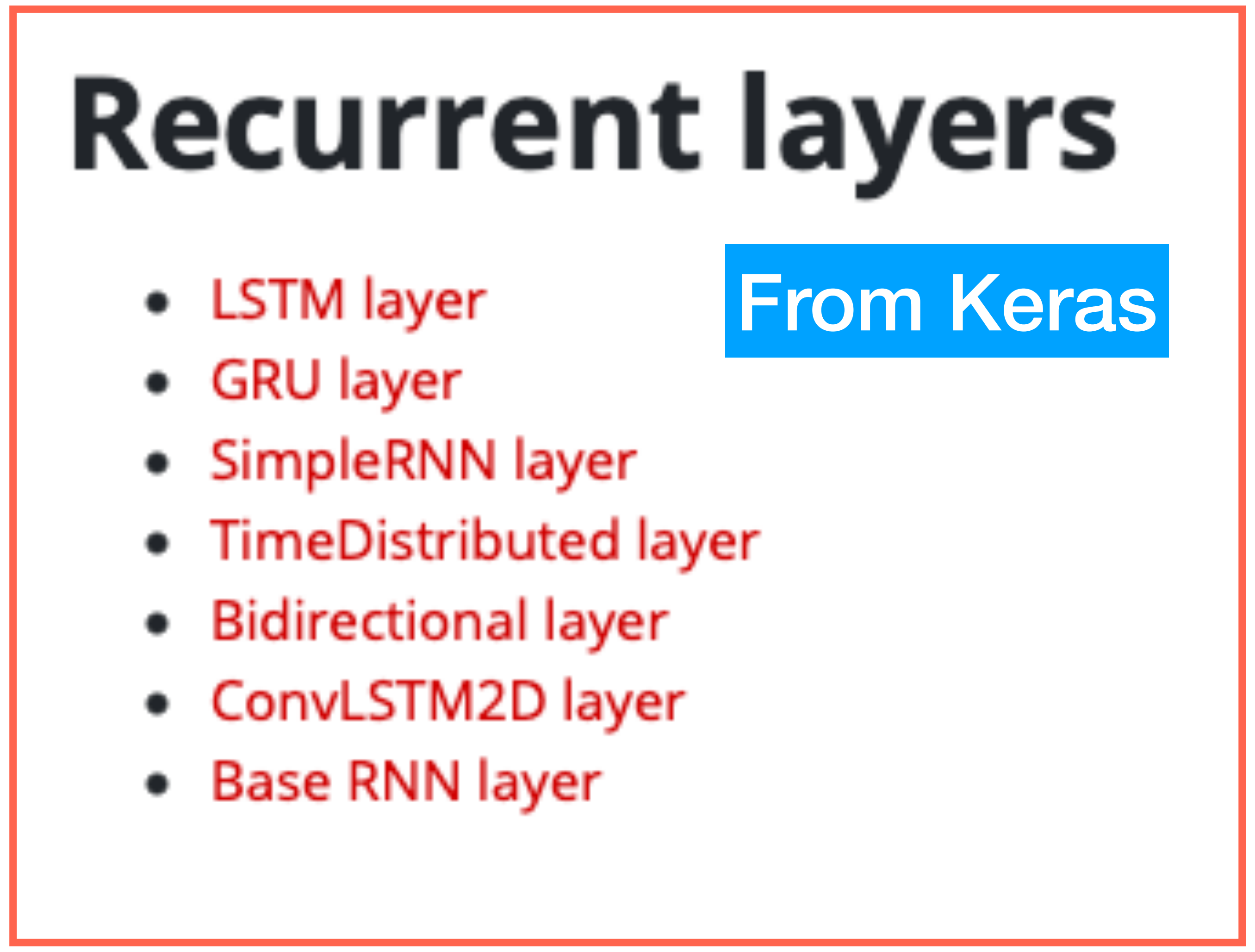
  - *One typically operates many recurrent units at once in a recurrent layer (like nodes in a dense layer, or kernels for CNN)*



(*) This is what we will be doing

Recurrent networks

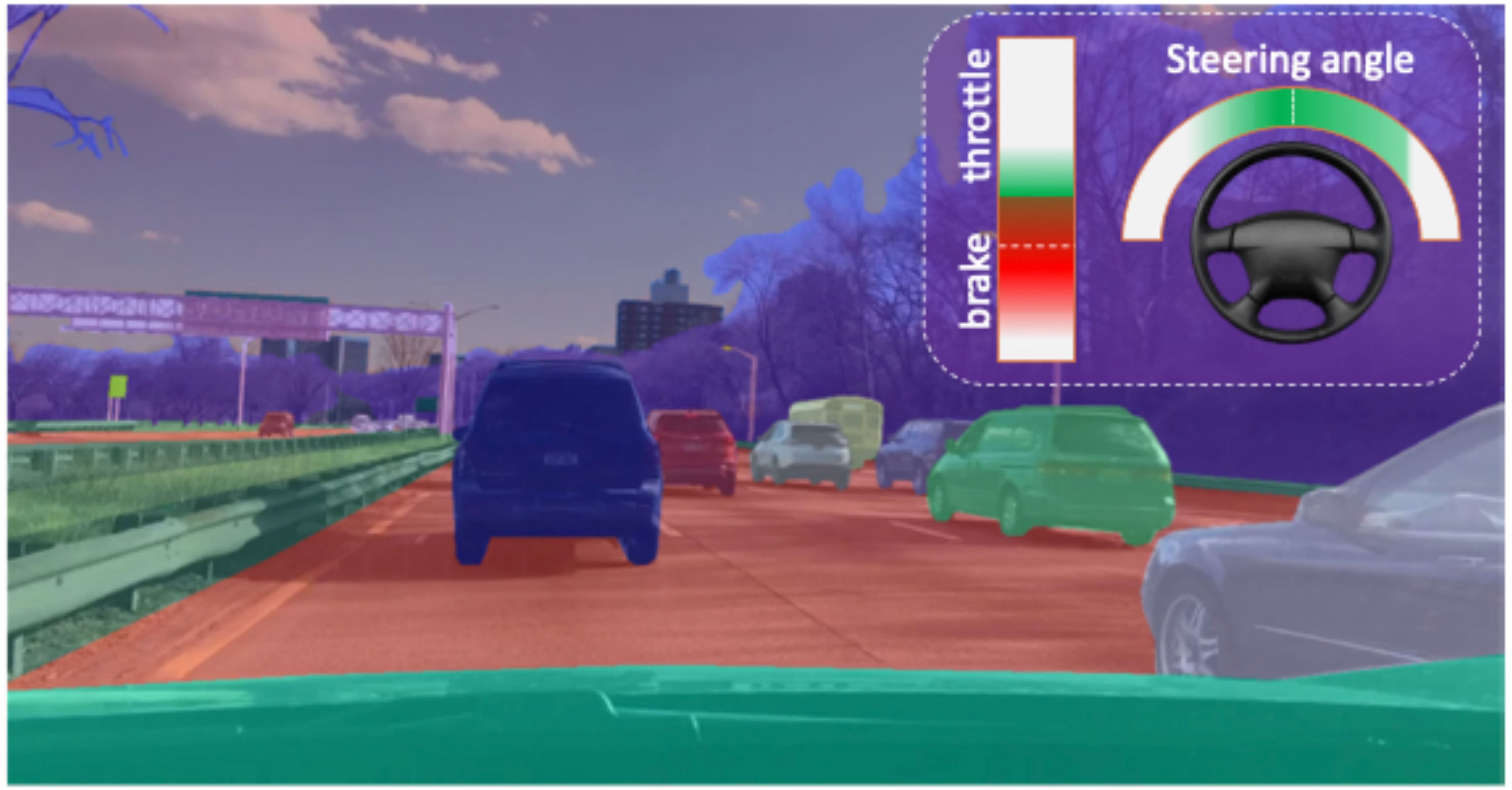
- Several architectures proposed
  - what changes is what happens in the A block
- We will focus on the first three
  - LSTMs (1995): the most popular (and performing) choice for serial-data processing
  - GRUs (2014): essentially an LSTM with a forget gate, with less parameters (and similar performance) as LSTM
  - SRNs, aka Elman (1990) and Jordan (1997) networks: simplest realisation of the idea



Recurrent layers

- LSTM layer
- GRU layer
- SimpleRNN layer
- TimeDistributed layer
- Bidirectional layer
- ConvLSTM2D layer
- Base RNN layer

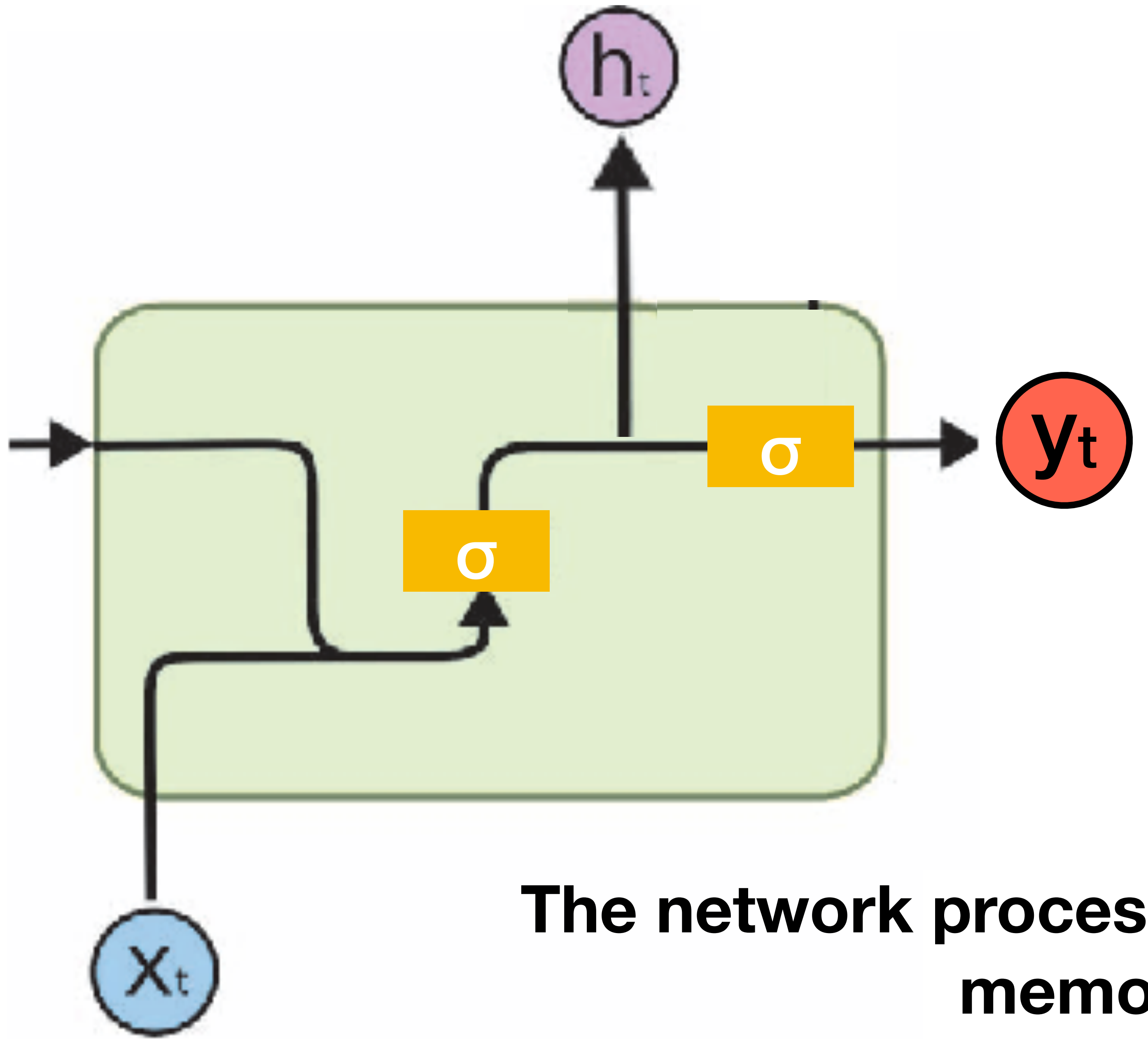
From Keras



(*) This is what we will be doing

SRN

- *The simplest recurrent architecture*
  - *The processing unit receives an input x and the output of the previous-particle processing h (the context)*
  - *The product of the two is activated by a tank function*
  - *the result can be used as it is and/or passed to the next processing*



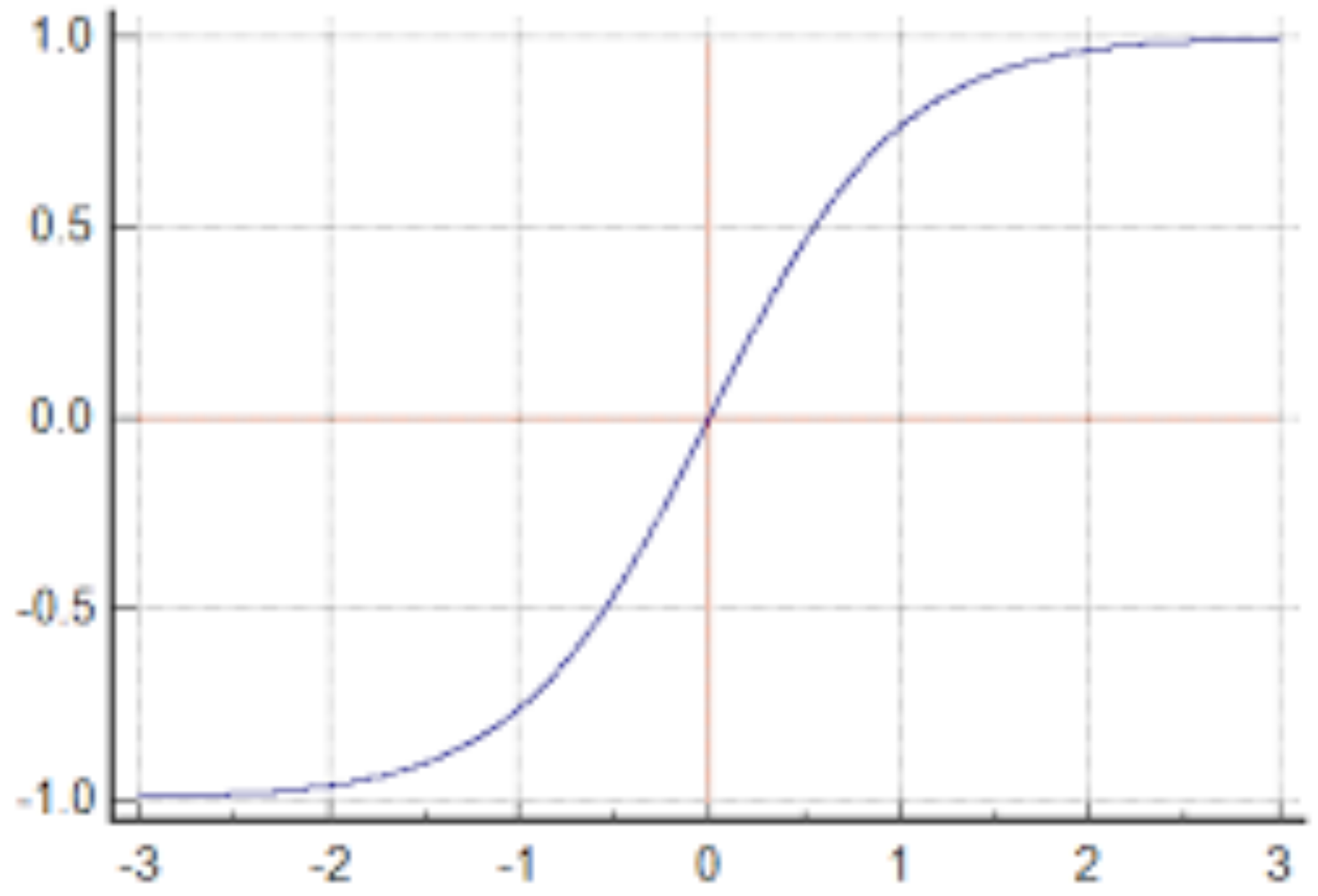
Elman network

$$h_t = \sigma_h(W_h x_t + U_h h_{t-1} + b_h)$$
$$y_t = \sigma_y(W_y h_t + b_y)$$

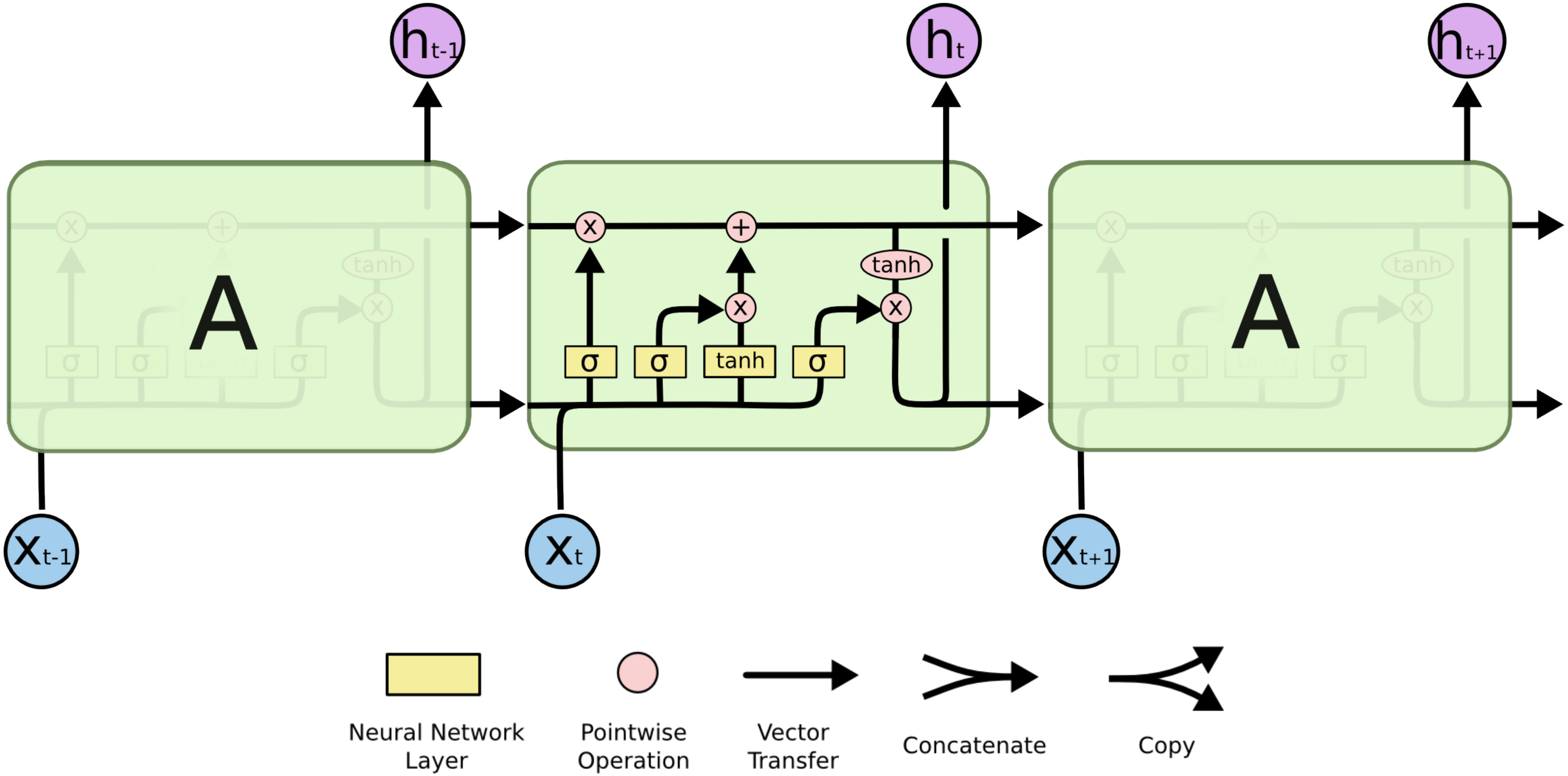
Jordan network

$$h_t = \sigma_h(W_h x_t + U_h y_{t-1} + b_h)$$
$$y_t = \sigma_y(W_y h_t + b_y)$$

The network processing essentially works as memory gates ->
One typically uses sigmoids or tanh as activation functions

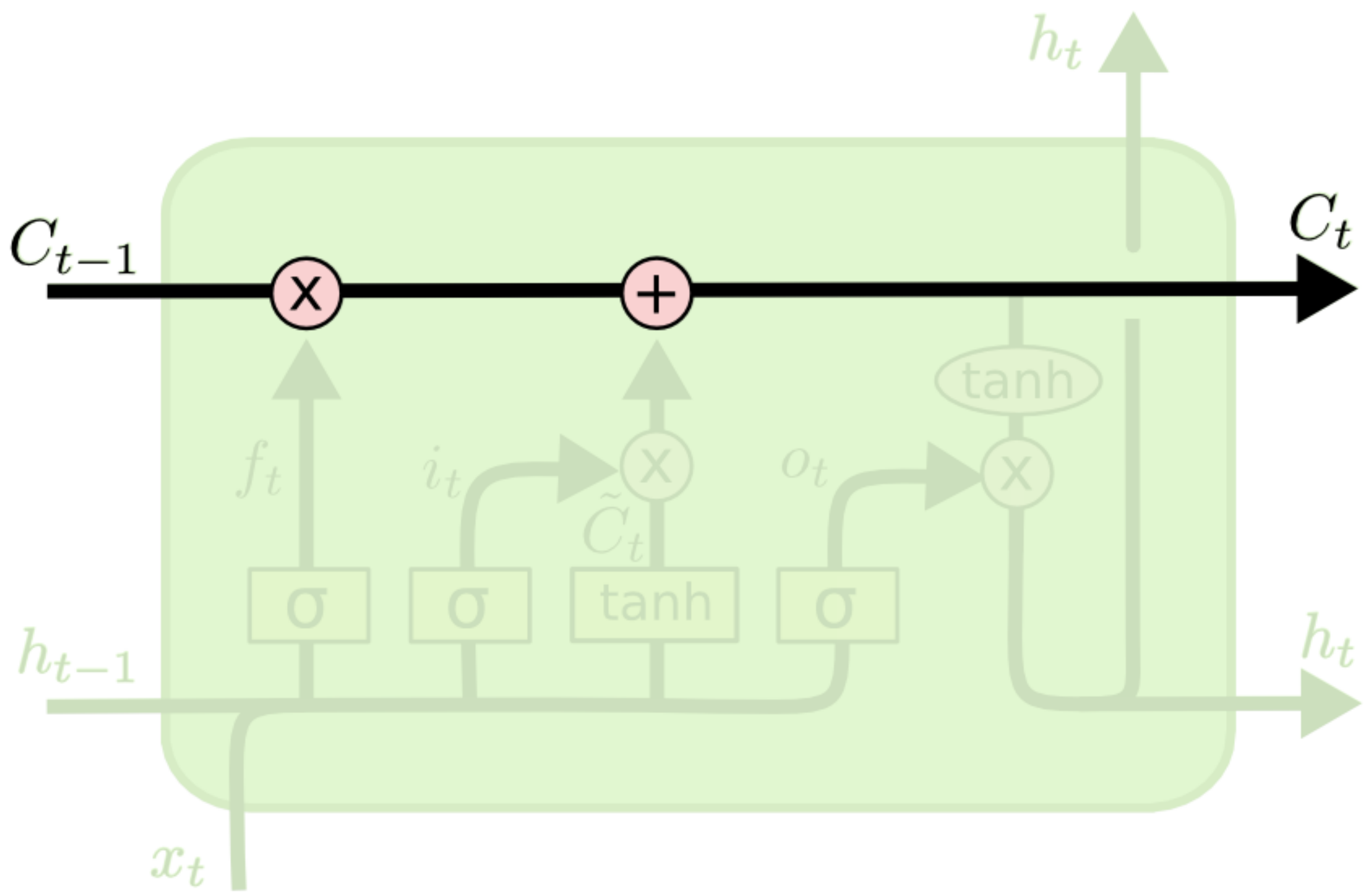


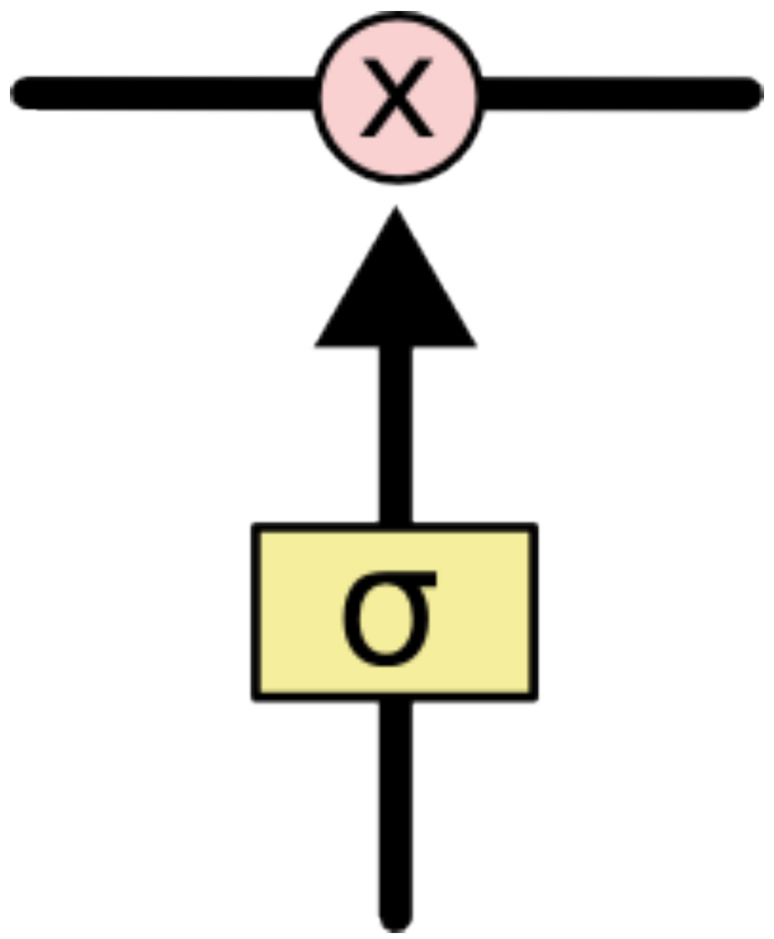
LSTM



h_{t-1}
h_t
h_{t+1}
A
σ
tanh
A
X_{t-1}
X_t
X_{t+1}
Neural Network Layer
Pointwise Operation
Vector Transfer
Concatenate
Copy

LSTM's memory cell

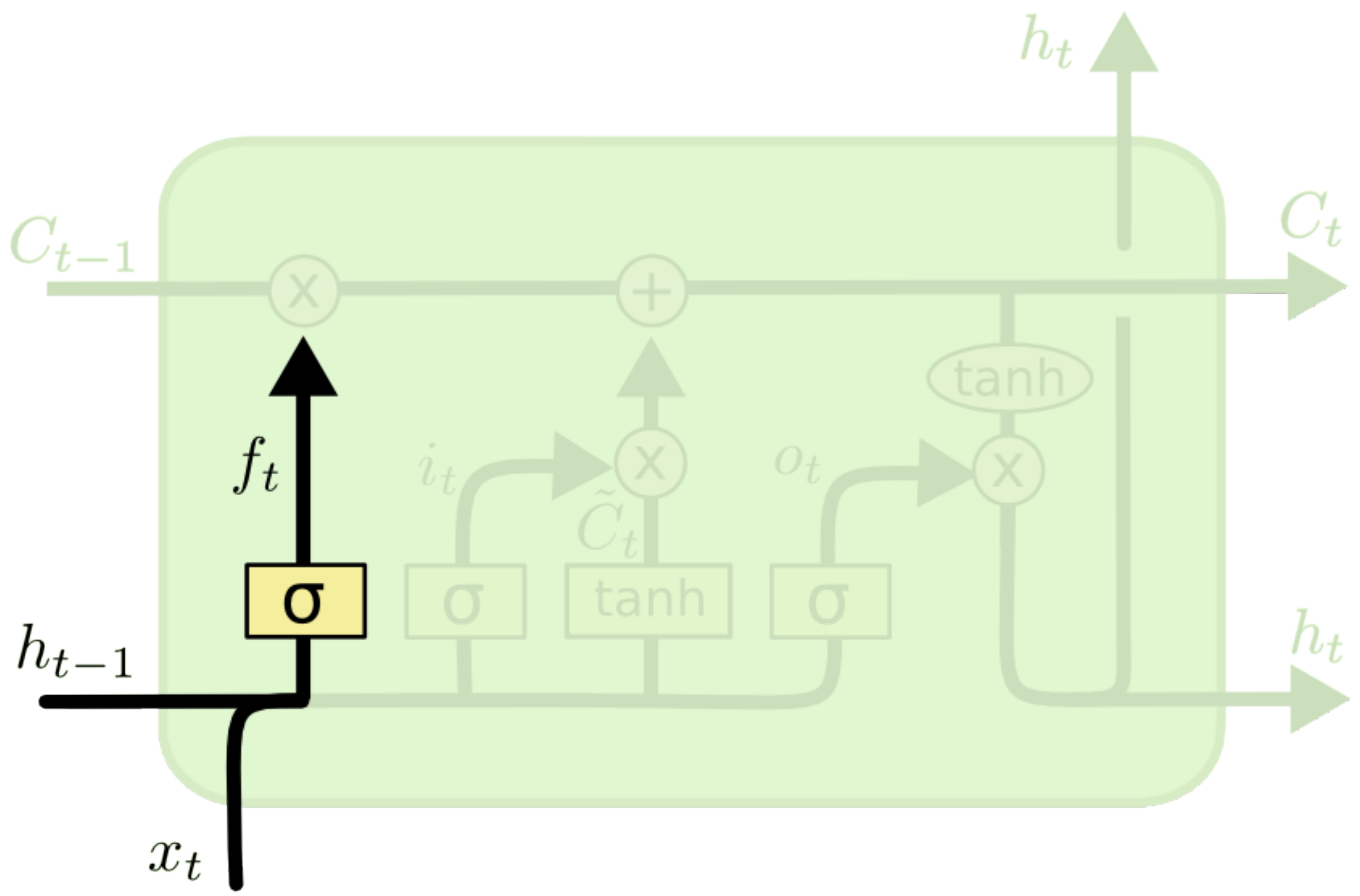




- *Information in the network flows through the memory cell*
  - *simple operations on it (to make the flow easy)*
  - *depending on gates (computed from next input), memory cell can be reset (yes/no decision taken by adequate activation function)*

LSTM processing

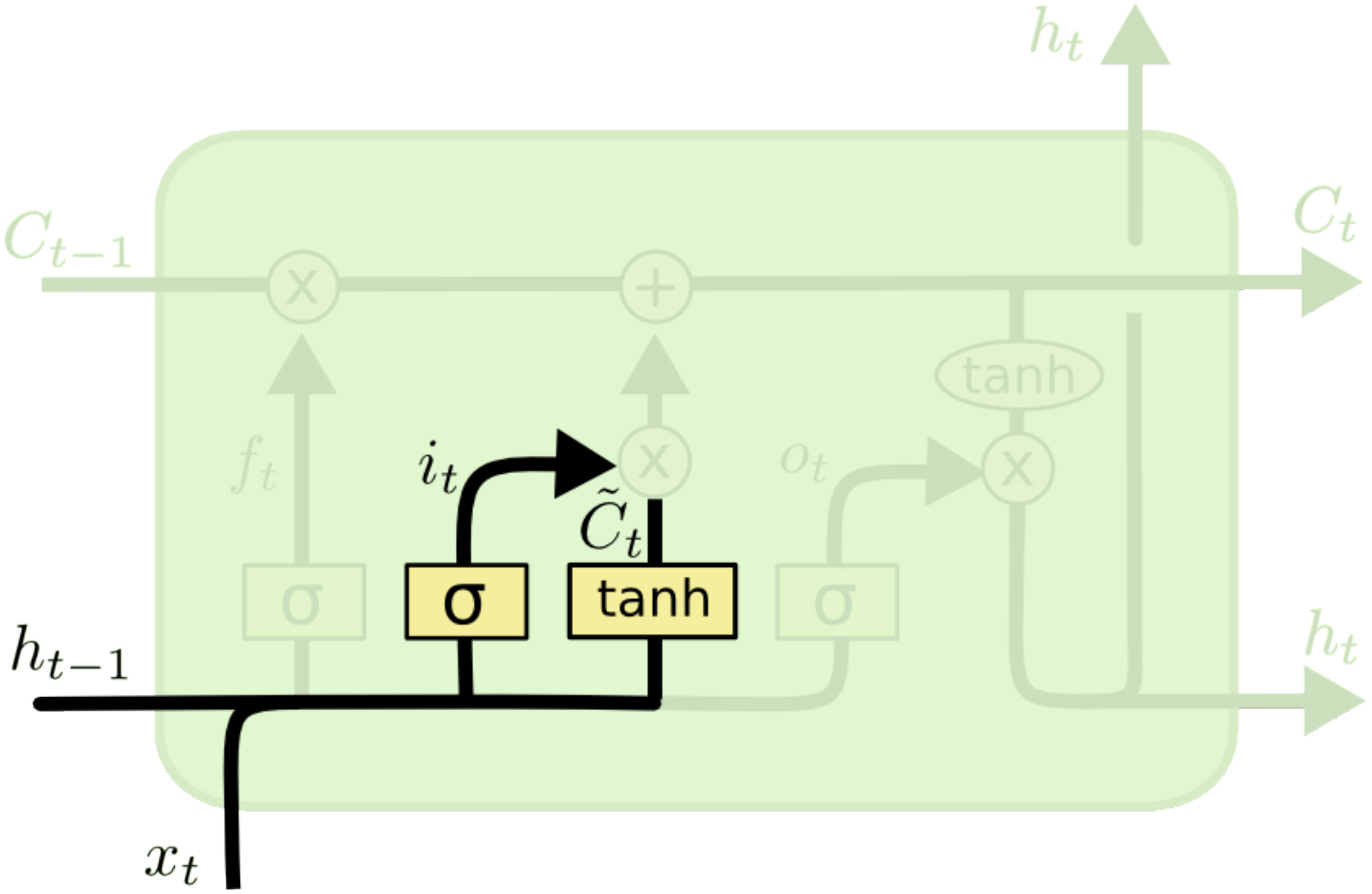
- ***Layer1:*** *decide if the context stored in the memory cell is to be forgotten or not*
  - *yes/no question -> sigmoid function*
  - *decision taken based on context and input*



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

LSTM processing

- ***Layer2:*** *two networks act to update the memory cell*
  - *a sigmoid (as in layer 1) decides which values in (0,1) to pass through*
  - *a tanh function assigns a weight in (-1,1) to it*

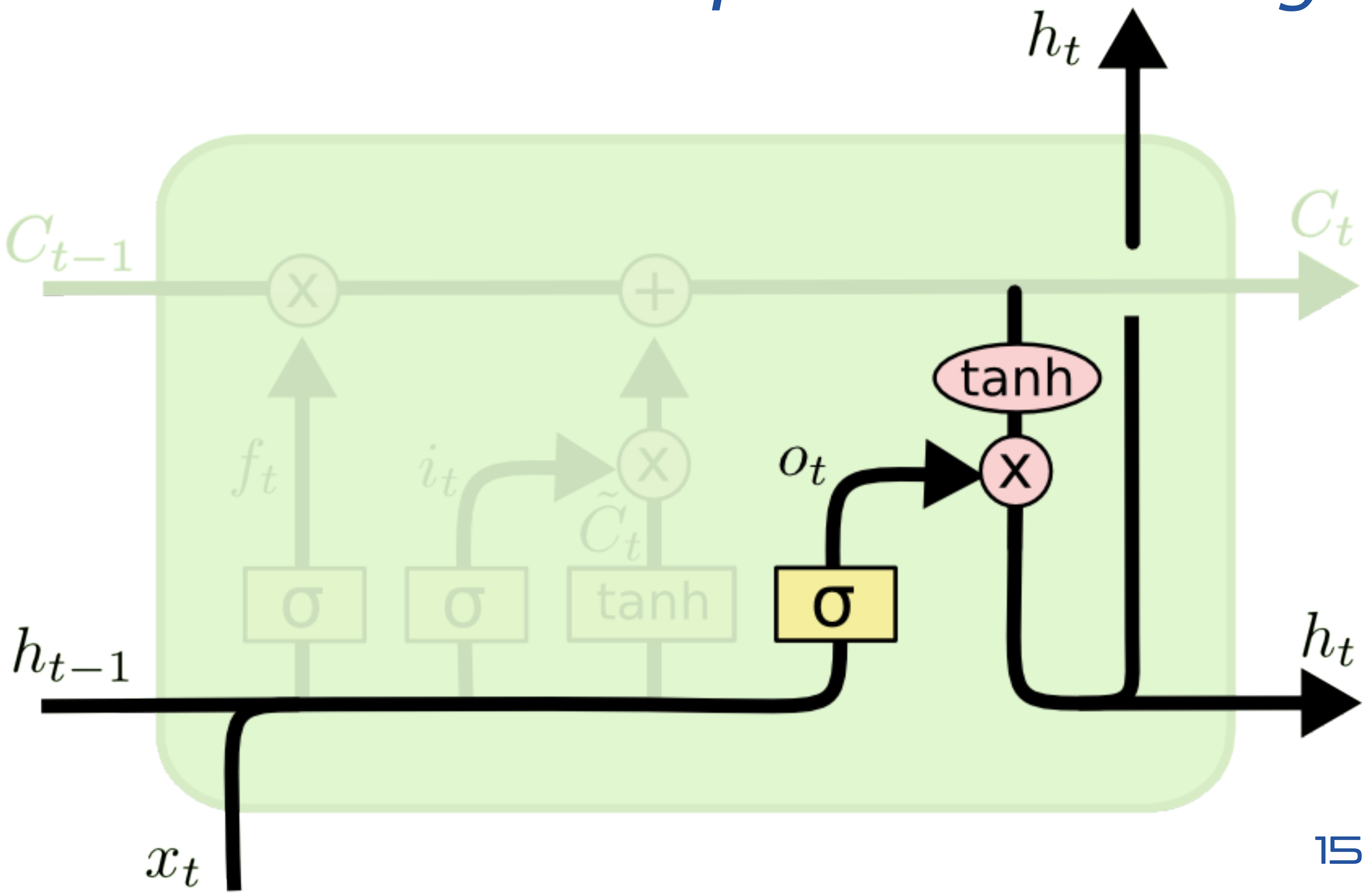


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

LSTM processing

- ***Layer3:*** *fixes the output value to be passed next*
  - *the cell content C_t is pushed in (-1,1) by a tanh*
  - *a gate function (sigmoid) sets which fraction of the cell content is passed through*

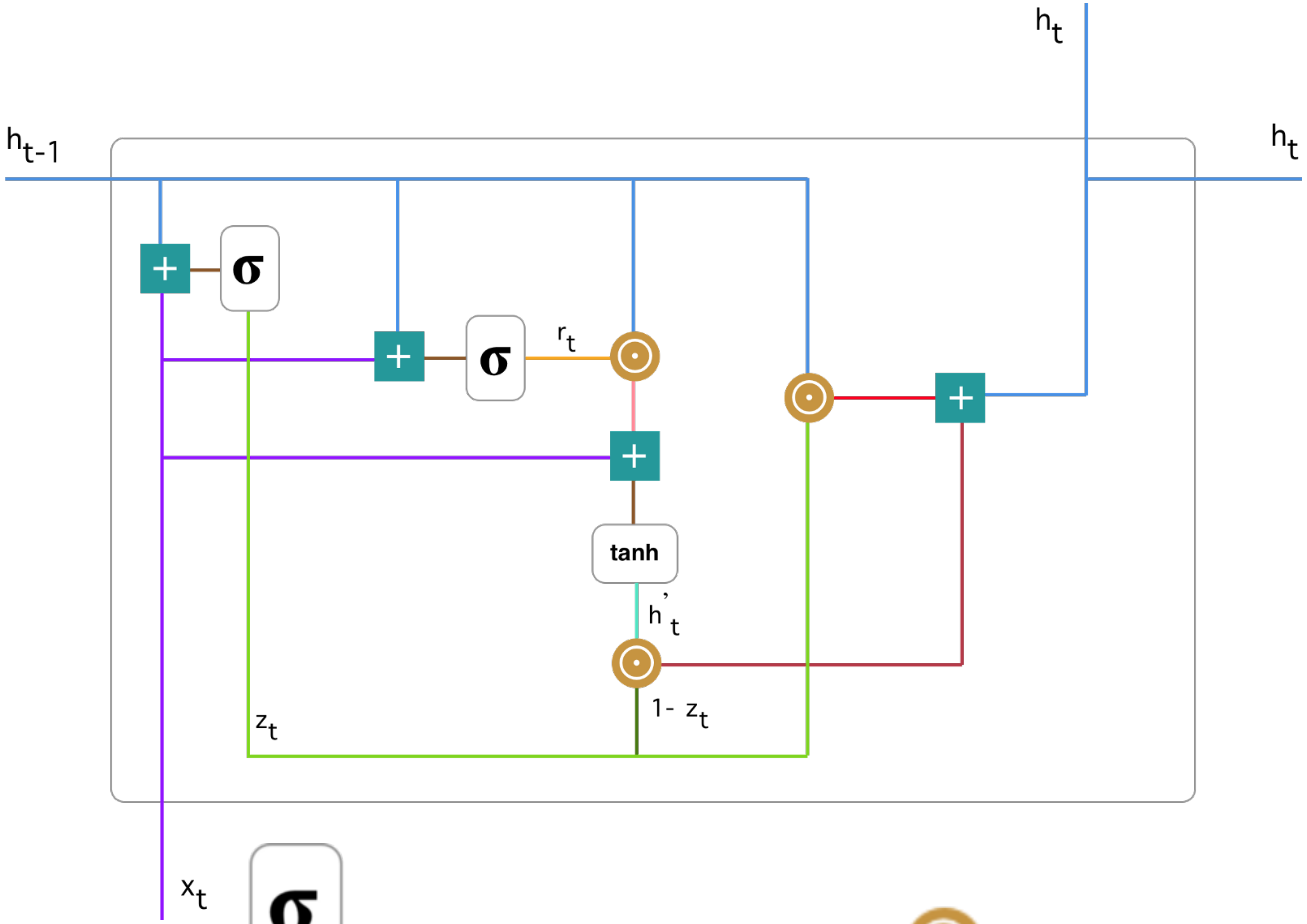


$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

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

European Research Council

Gated Recurrent Unit



https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be

“plus” operation

σ

“sigmoid” function

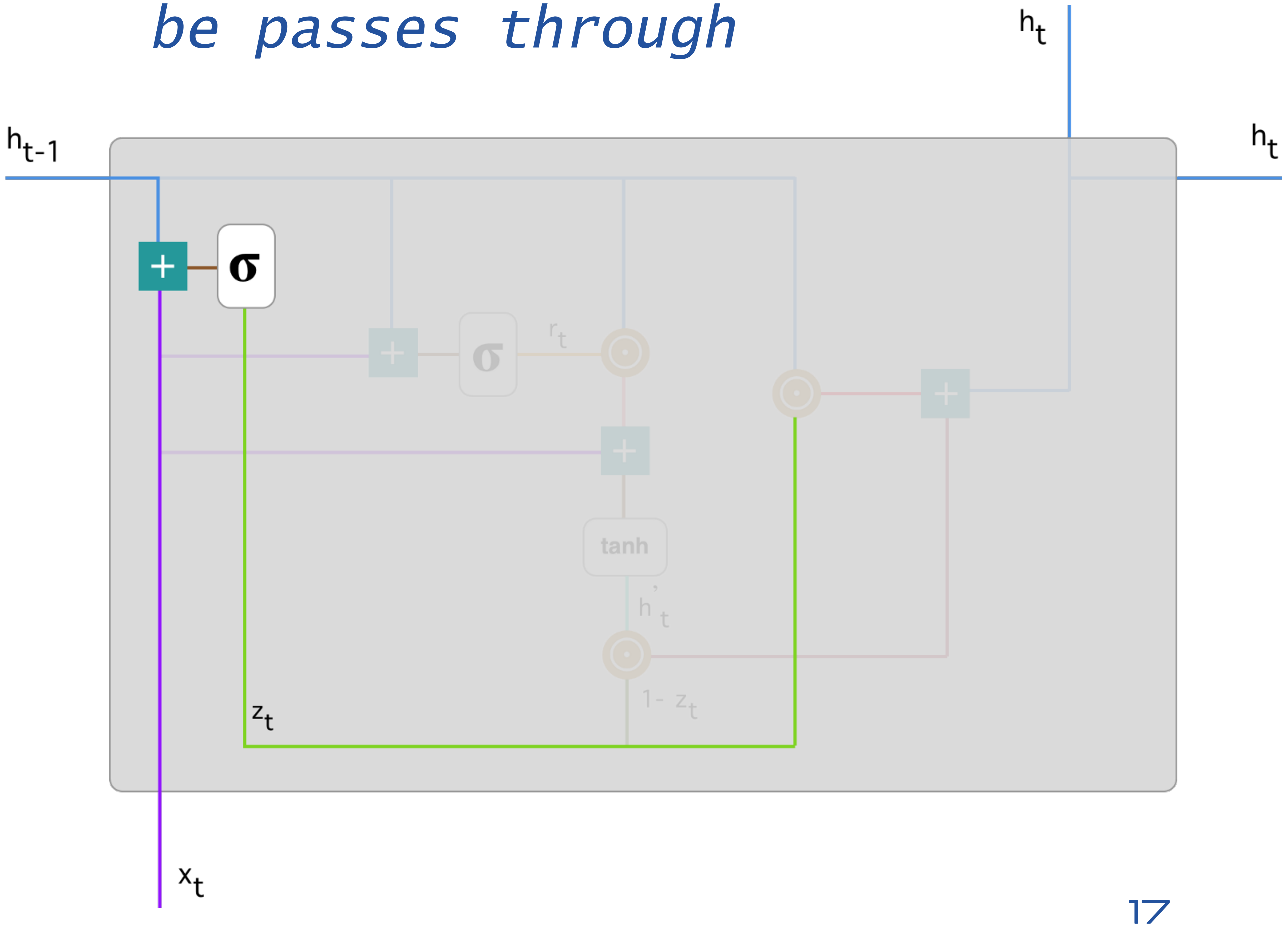
“Hadamard product” operation
(aka element-wise product)

tanh

“tanh” function

GRU processing

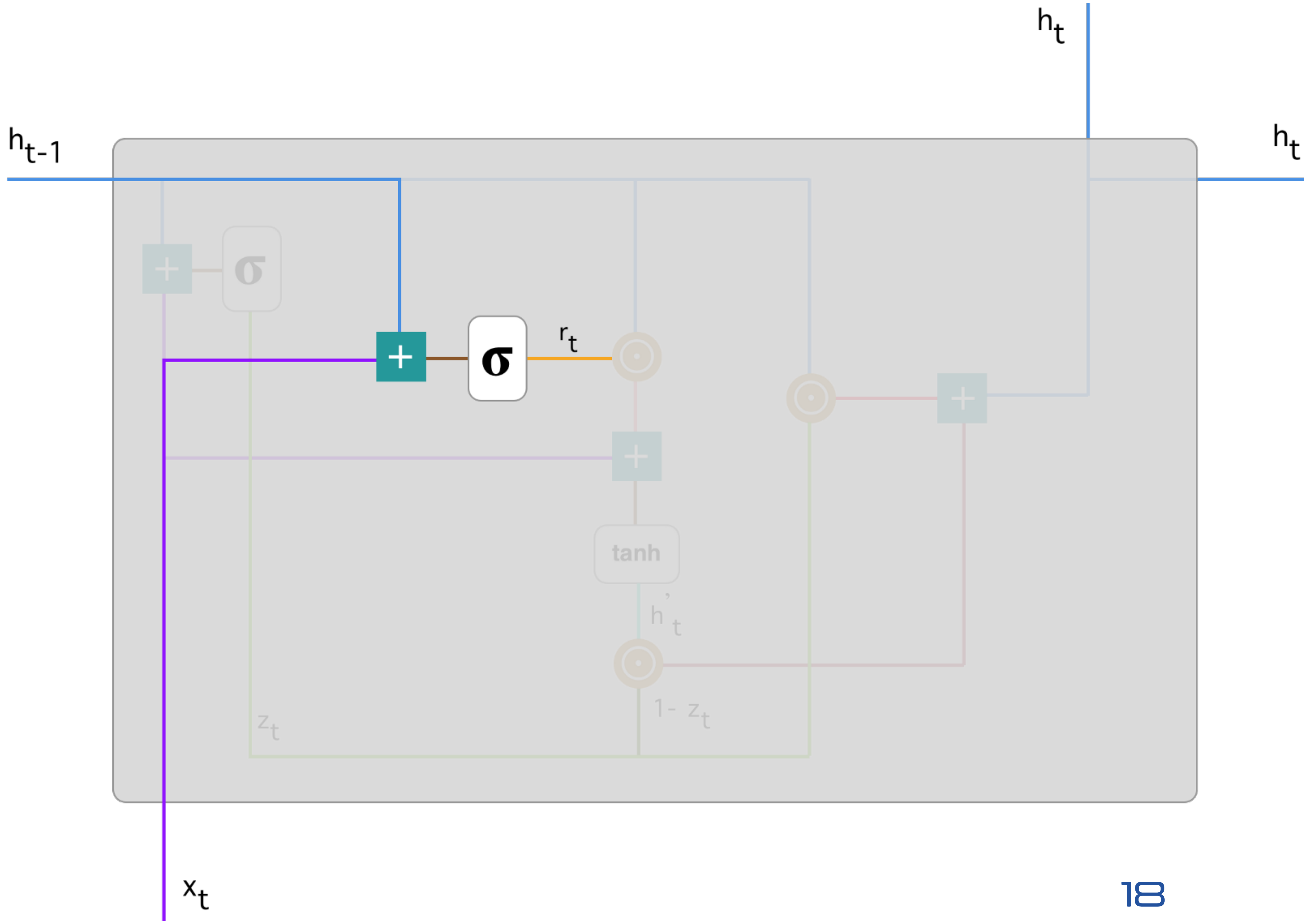
- ***Layer1:*** *update gate*
  - *multiply input and context by update weights*
  - *based on that, decide how much of the previous information should be passes through*



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

GRU processing

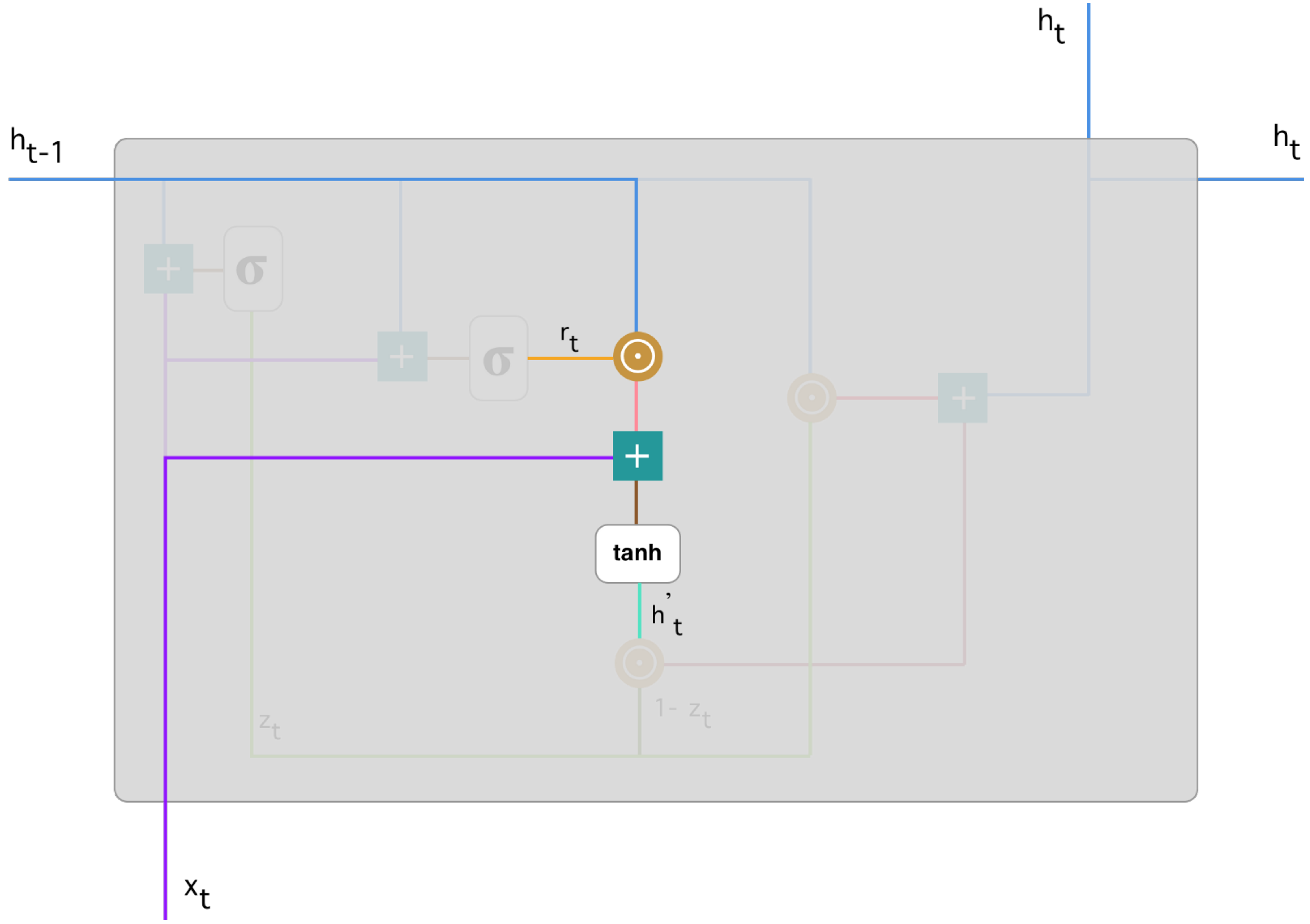
- ***Layer2:*** *reset gate*
  - *multiply input and context by reset weights*
  - *decide how much of the previous information is to be forgotten*



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

GRU processing

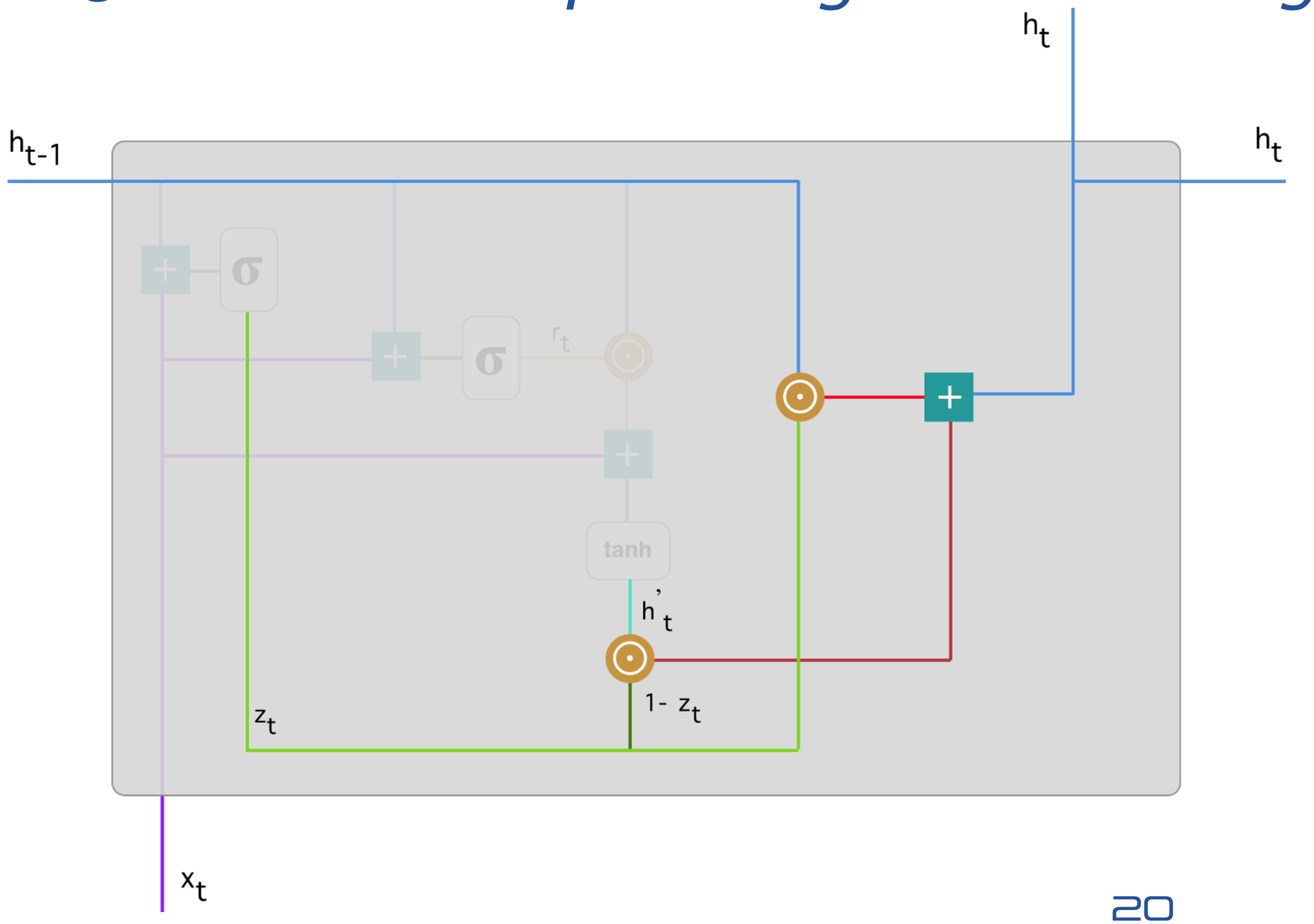
- ***Layer3:*** *determine the current memory content*
  - *the context information is partially removed to an element-by-element product with the reset gate*
  - *the result is summed to the (weighted) input and passed through a tank activation function*



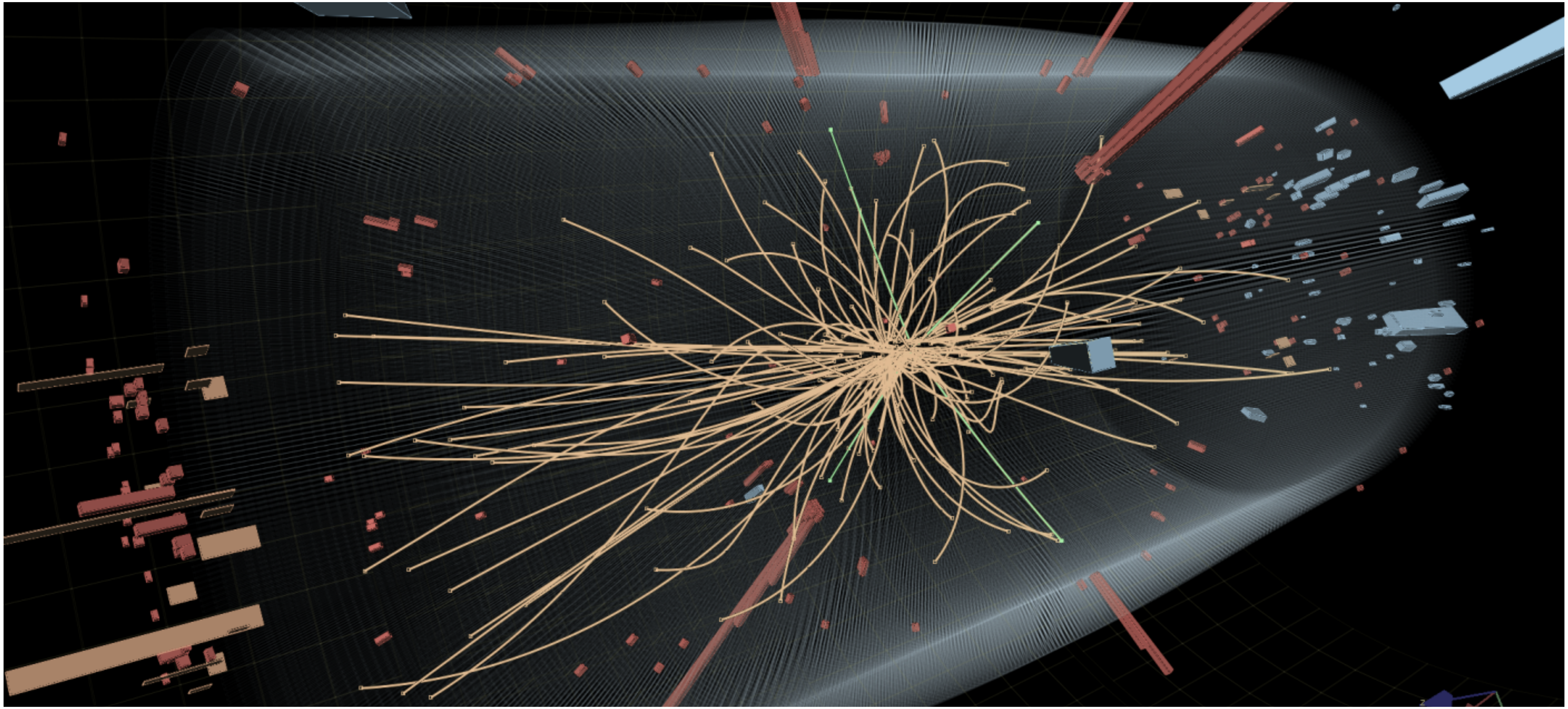
$$h_t^{'} = \tanh(Wx_t + r_t \odot Uh_{t-1})$$

GRU processing

- **Layer4:** *compute the new memory state, to pass through*
  - *mixes the current memory state with the input one*
  - *uses the update gate to weight the two contributions*



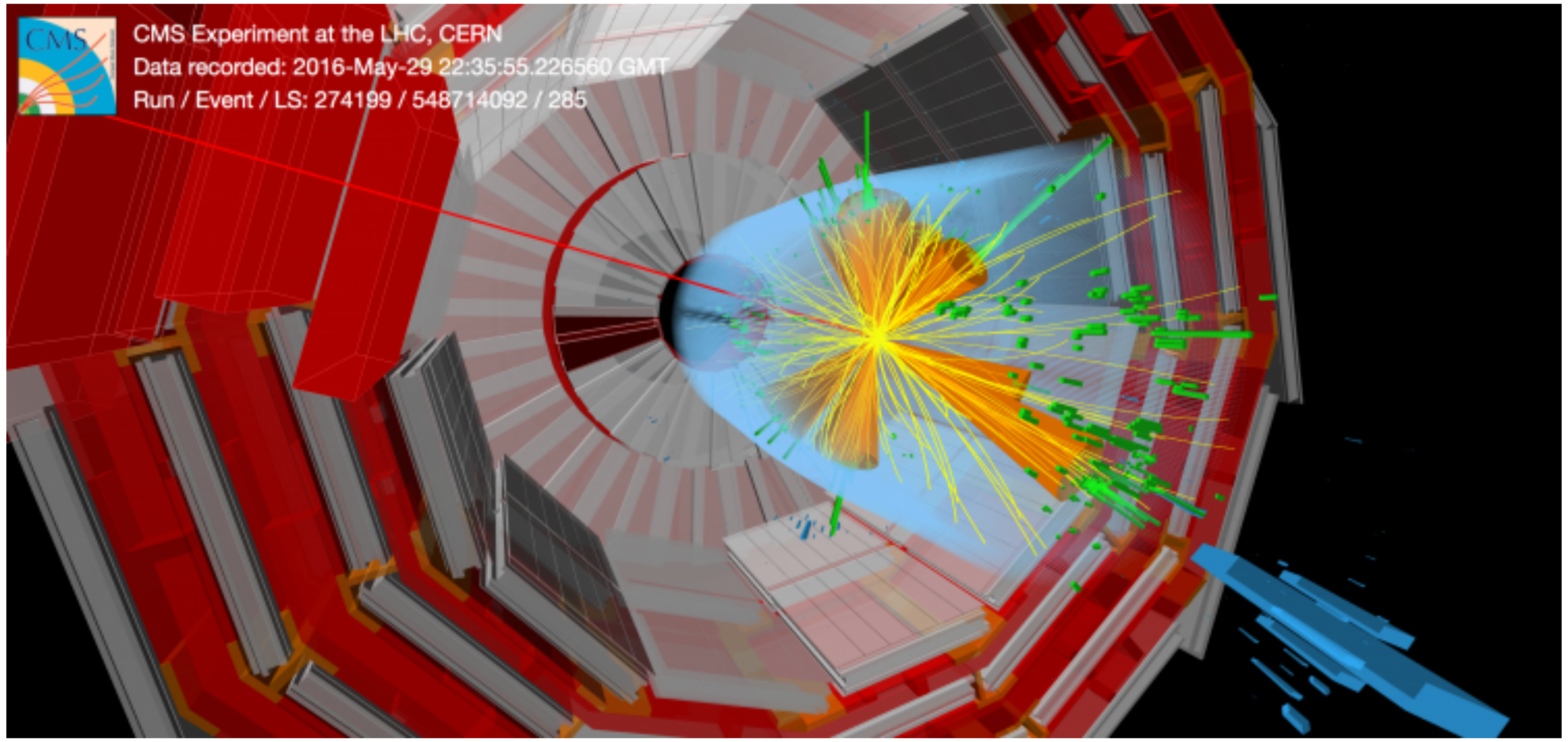
$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot h_t'$$



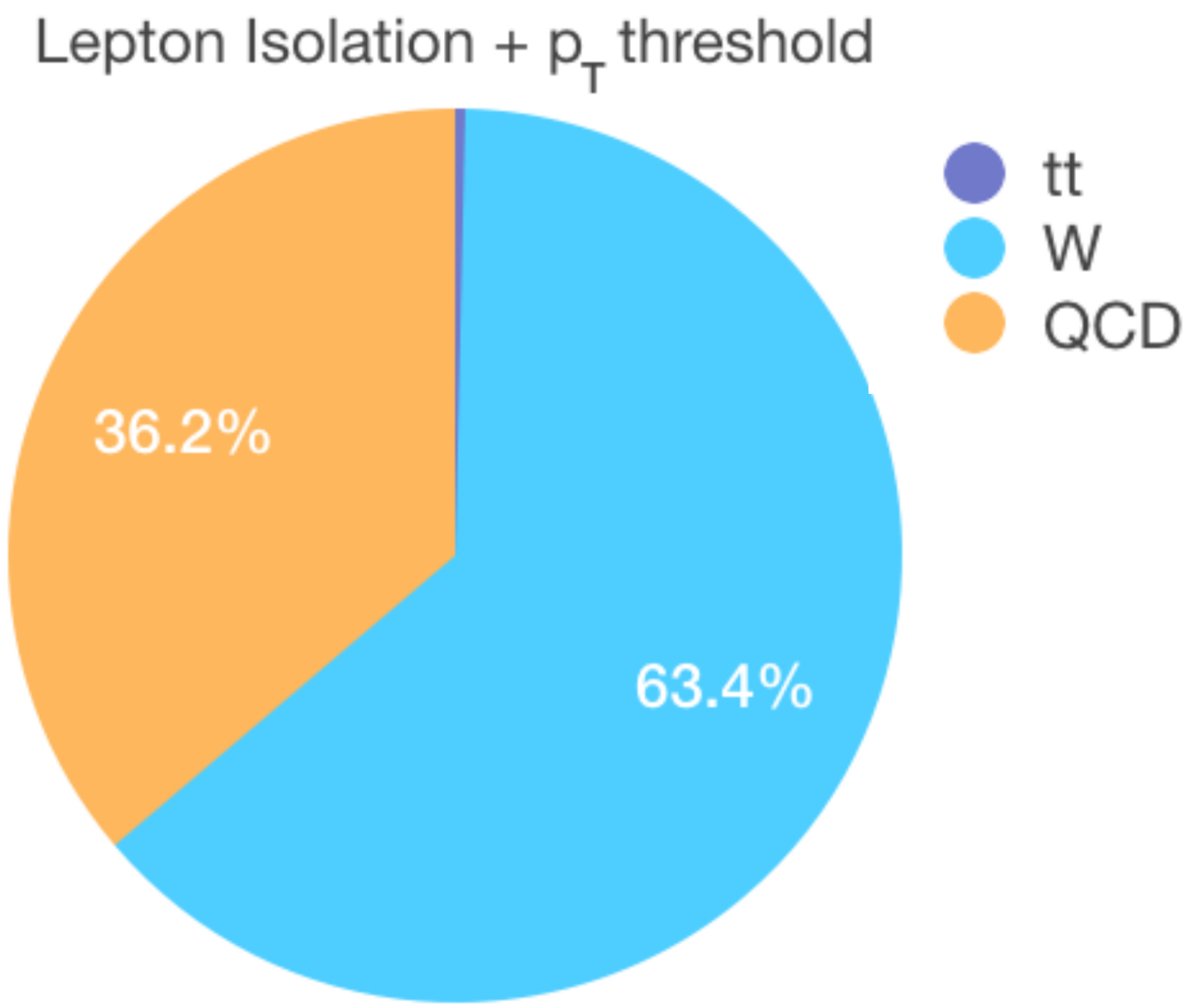
Application to HEP data

Example: topology classification @LHC

- *Finding which process generated an event is a typical task @LHC experiments*

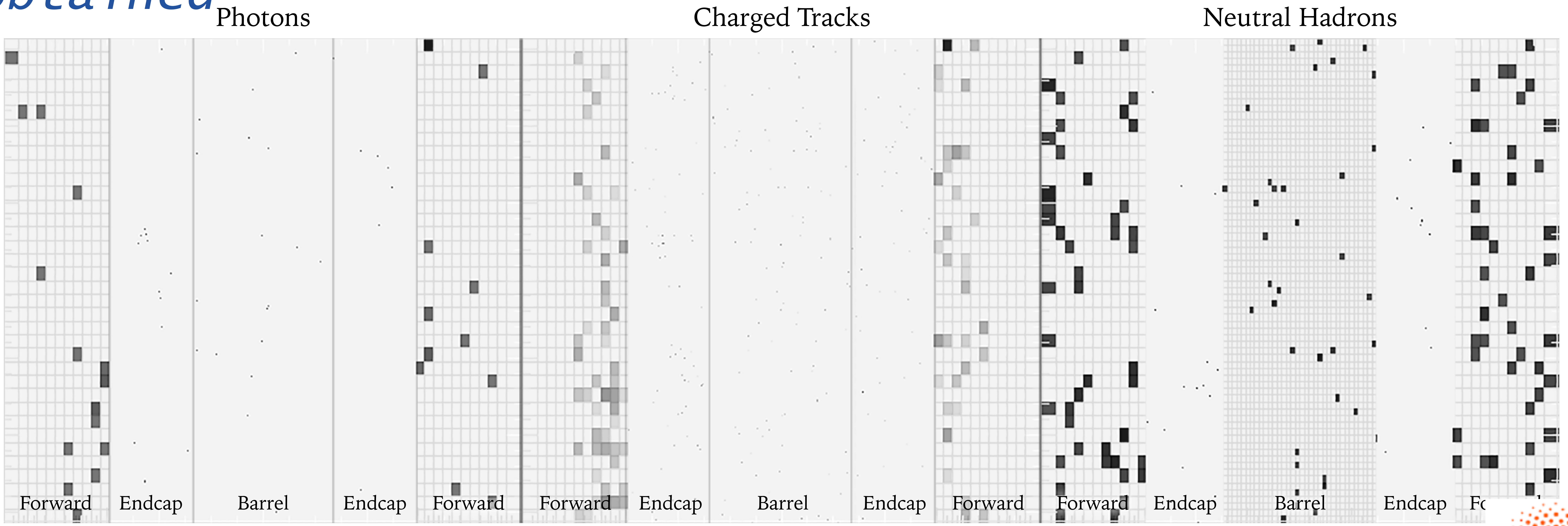


- *Let's take as input the events with one energetic electron or muon (lepton)*
  - *After some loose selection, mainly three processes left: jet production (QCD, the background) + W or tt productions (the signals)*
  - *We want to separate the three components*

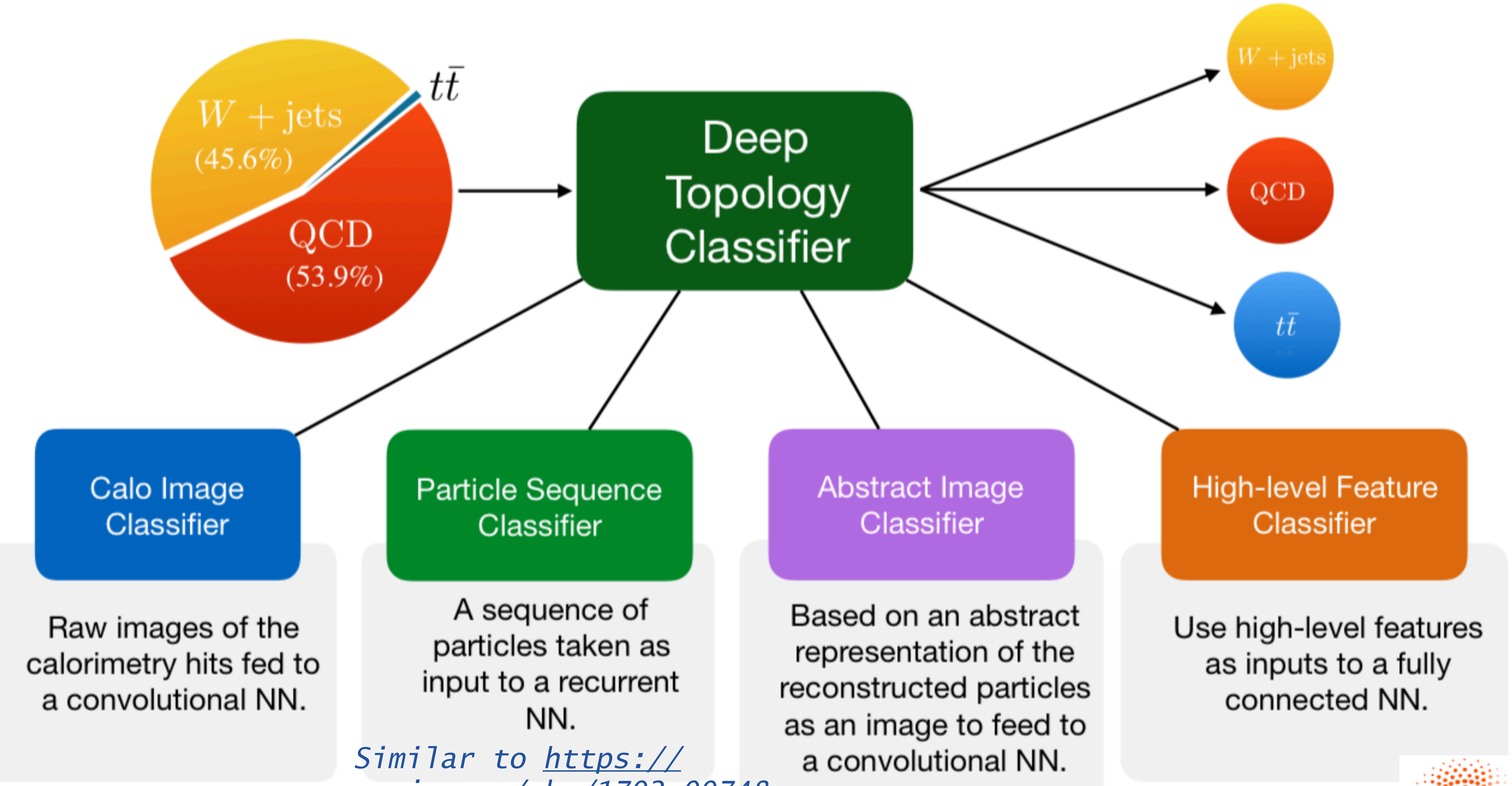


What the event looks like

- sparse image with many pixels
  - not the kind of image that CNNs usually deal with
  - still, reasonable performances (AUC~90%) can be obtained



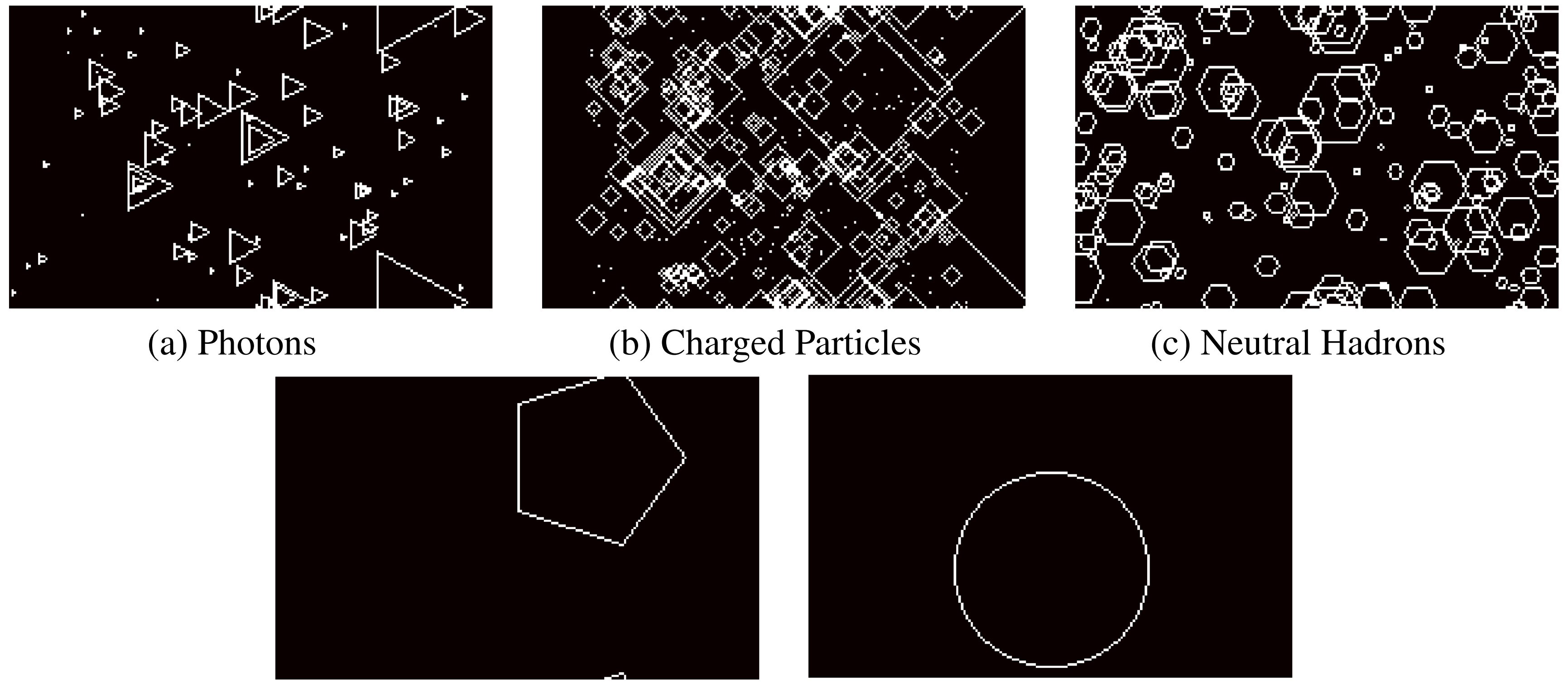
Event Representations



Similar to https://arxiv.org/abs/1702.00748

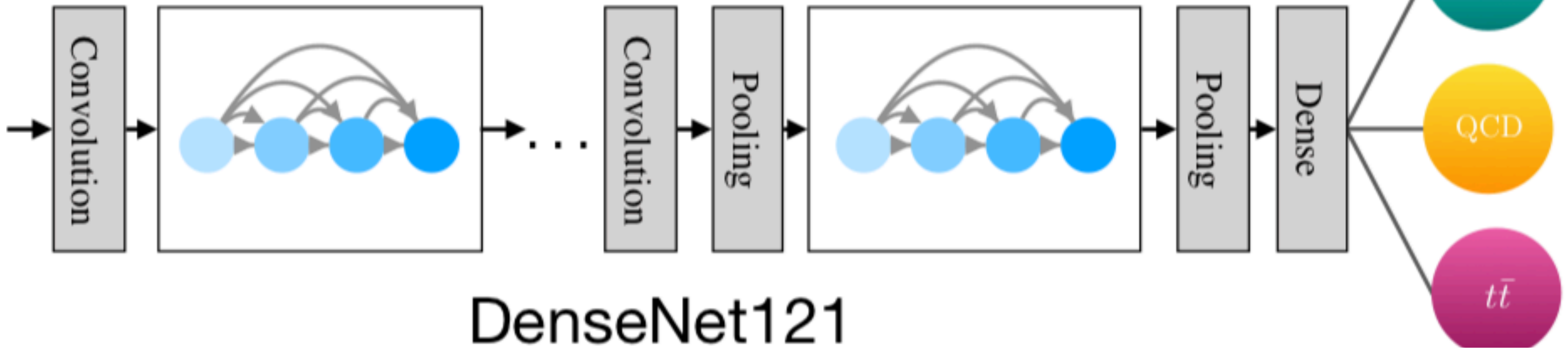
Inspired from https://arxiv.org/abs/1708.07034

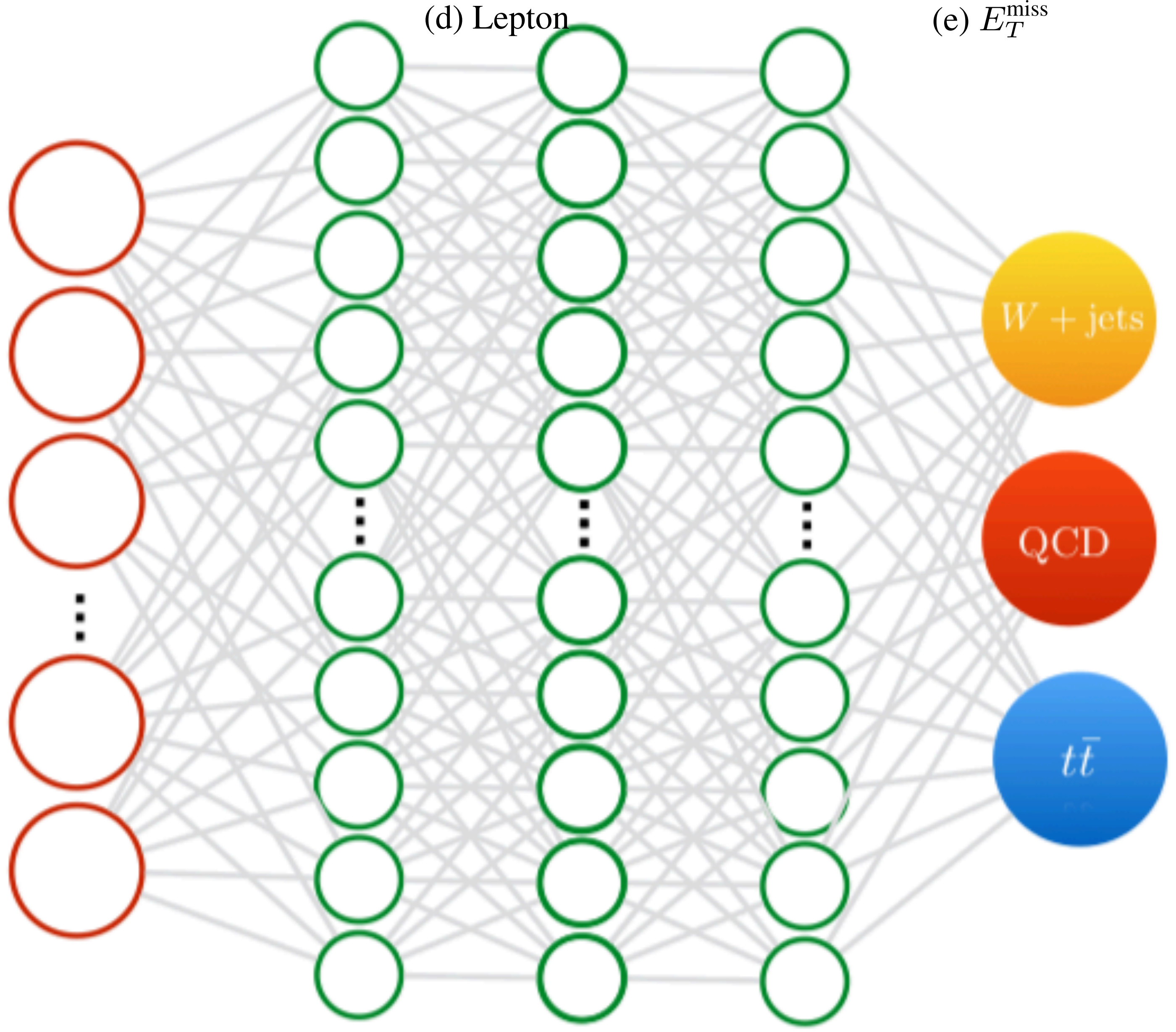
Event Representations



(a) Photons (b) Charged Particles (c) Neutral Hadrons (d) Lepton (e) E_T^{miss}

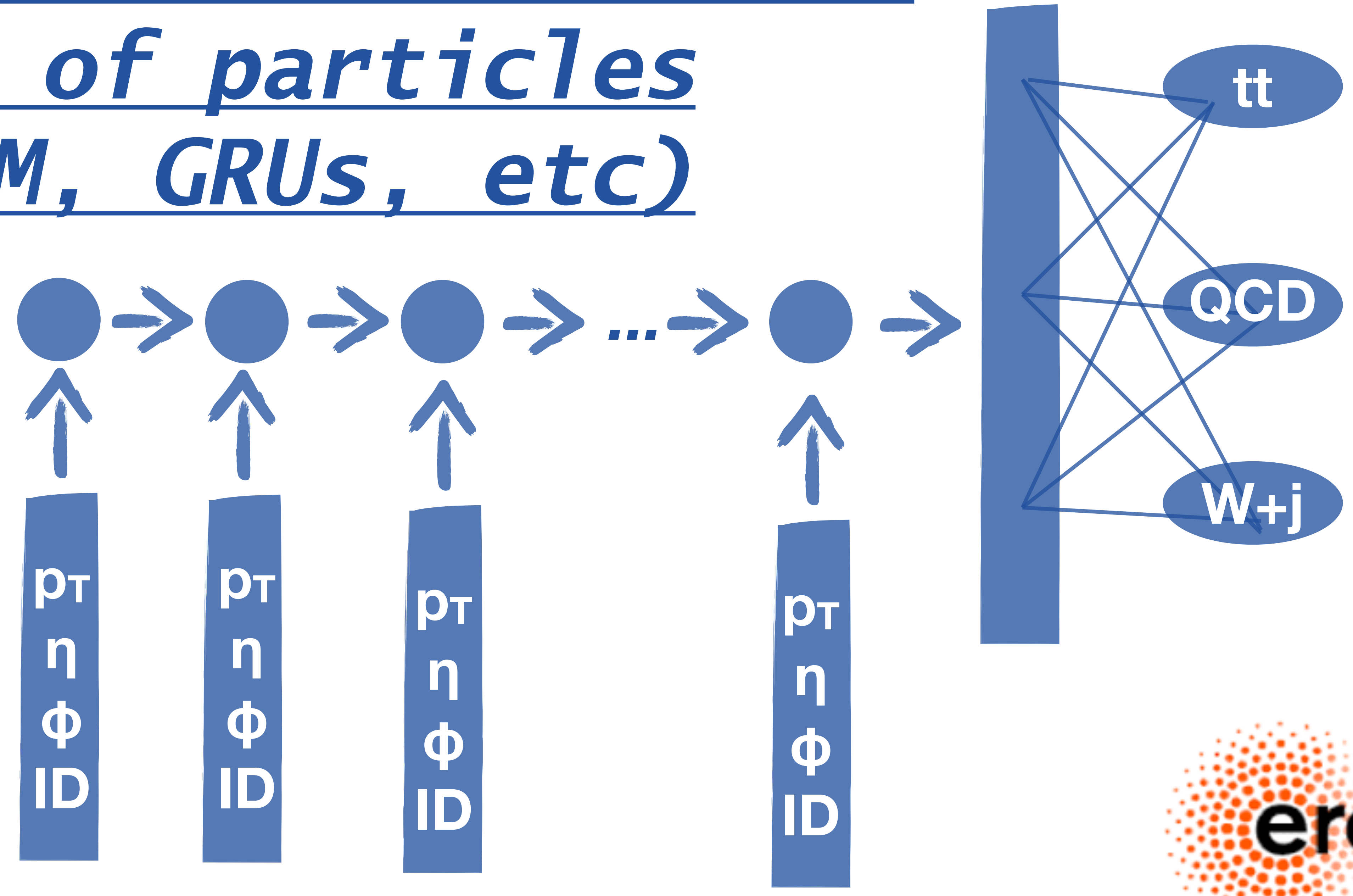
DenseNet on images





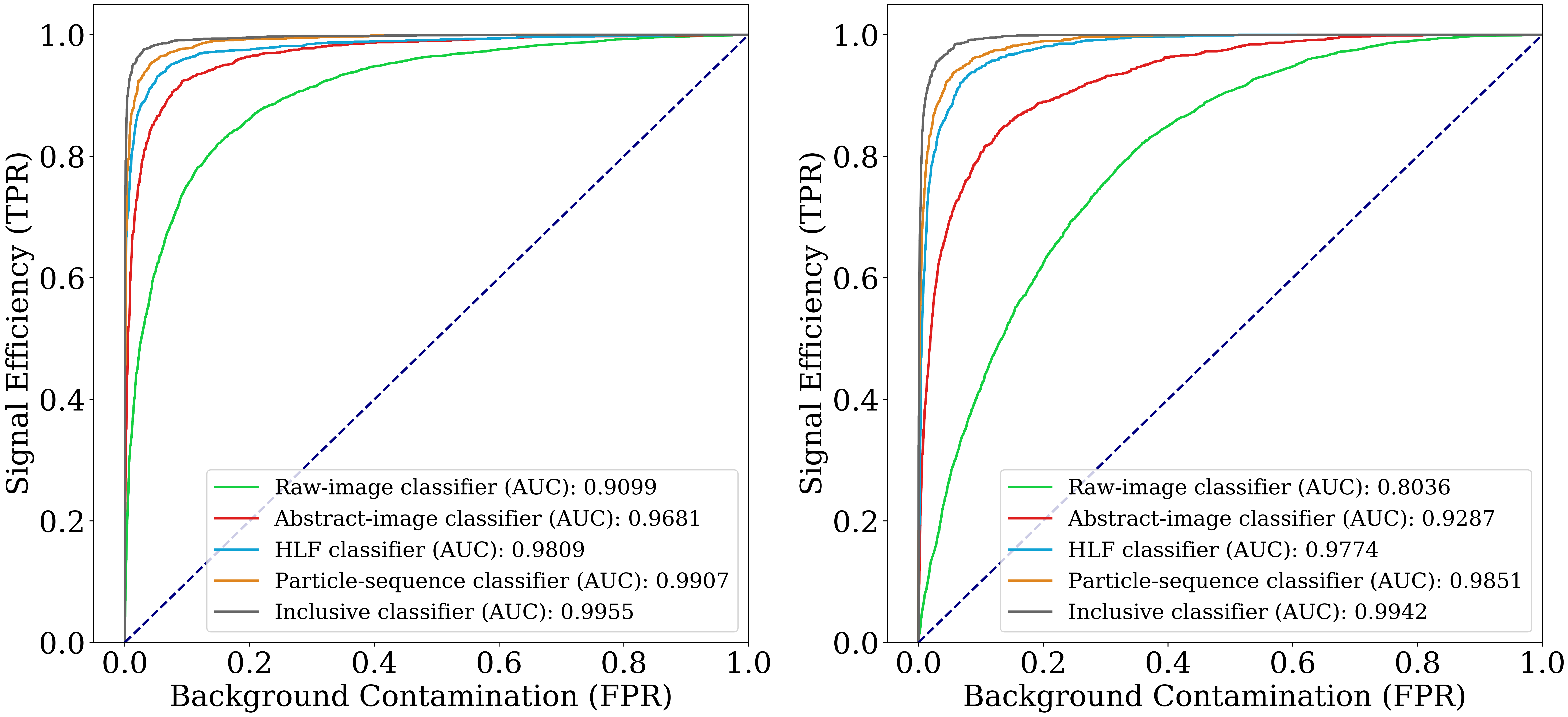
Fully-Connected classifier on physics-motivated features

Recurrent nets on the list of particles (LSTM, GRUs, etc)



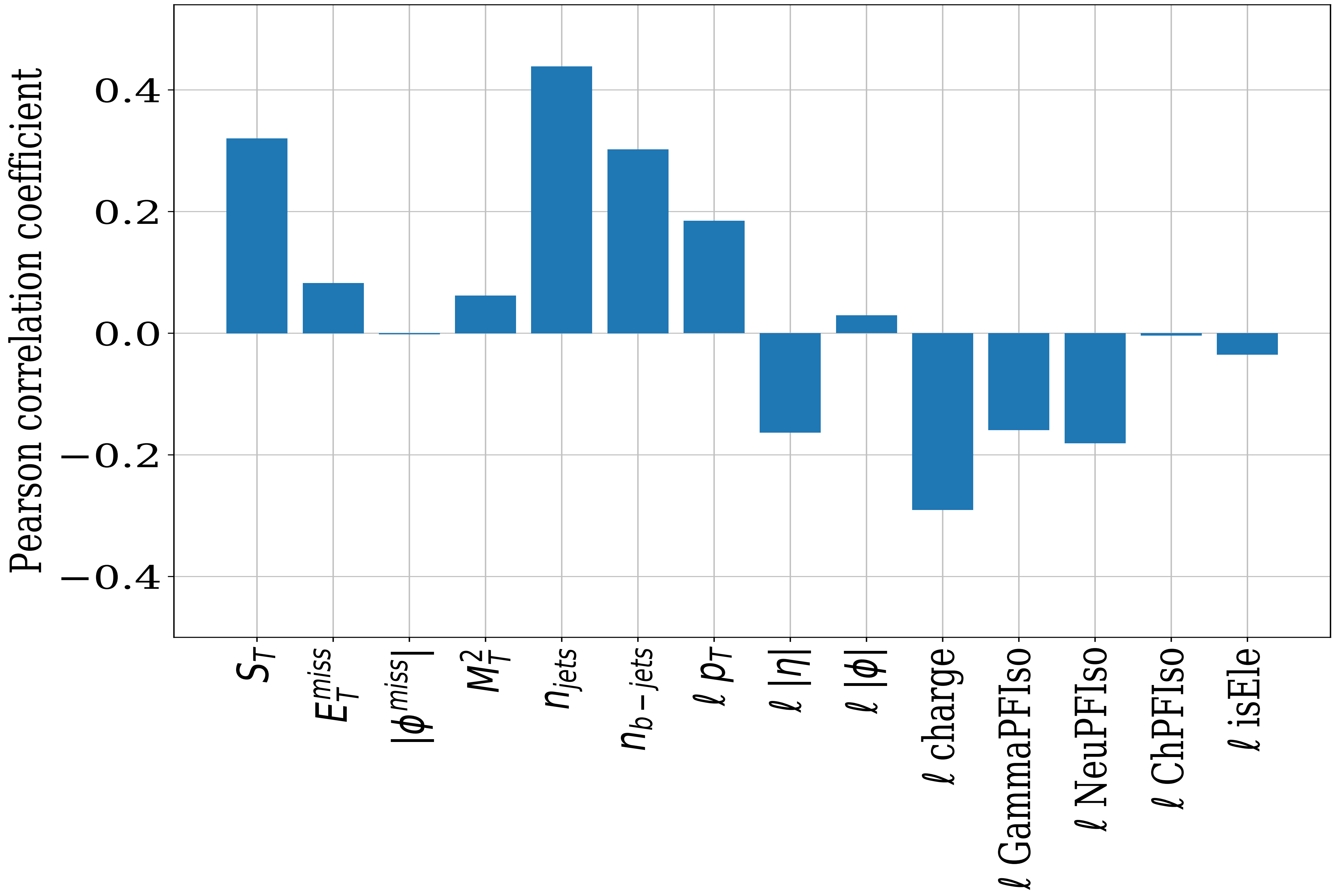
Performance

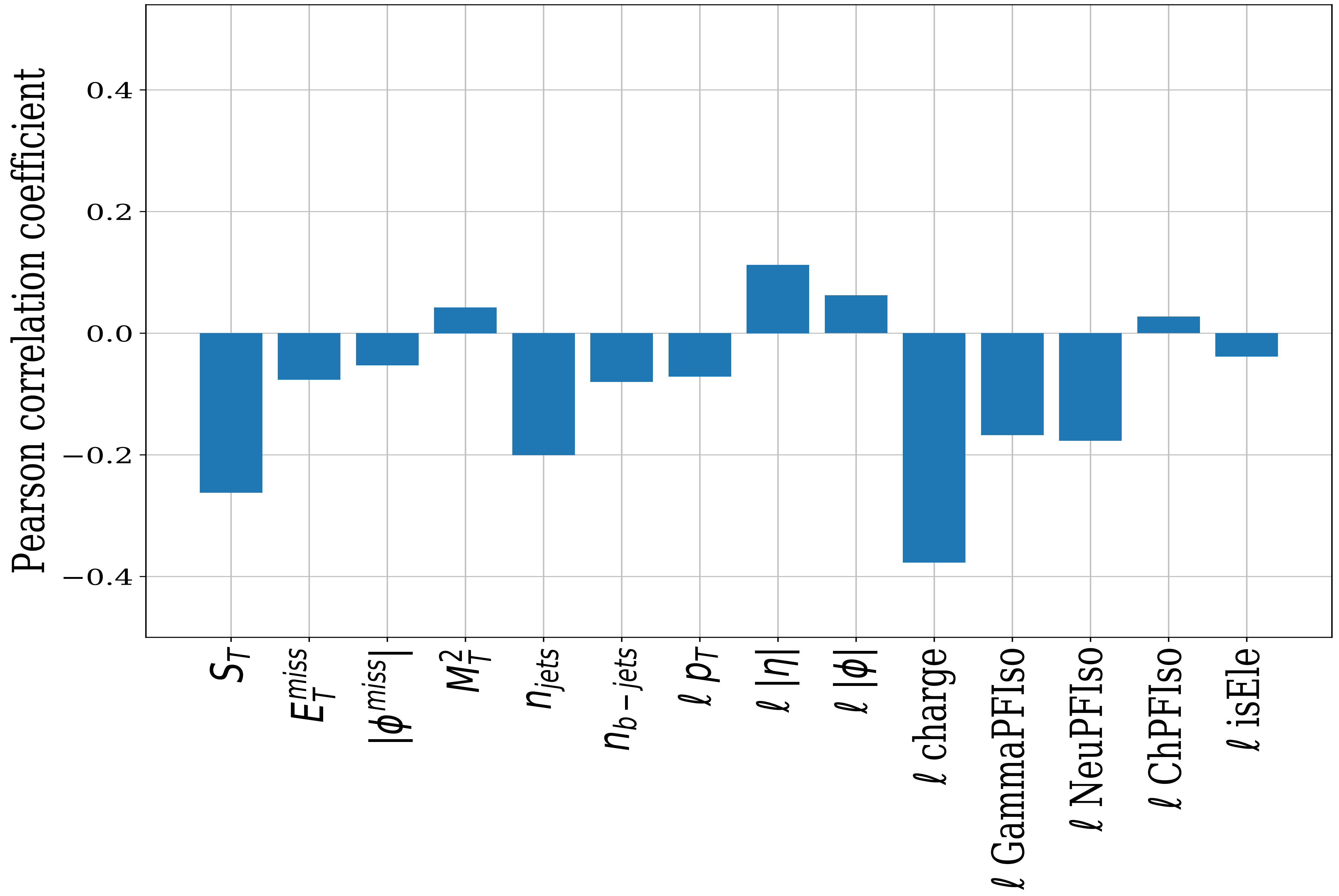
- The GRU provides the best discrimination power
- The HLFs come second
- The combination of the two further improve



(a) $t\bar{t}$ selector (b) W selector

Selection performances





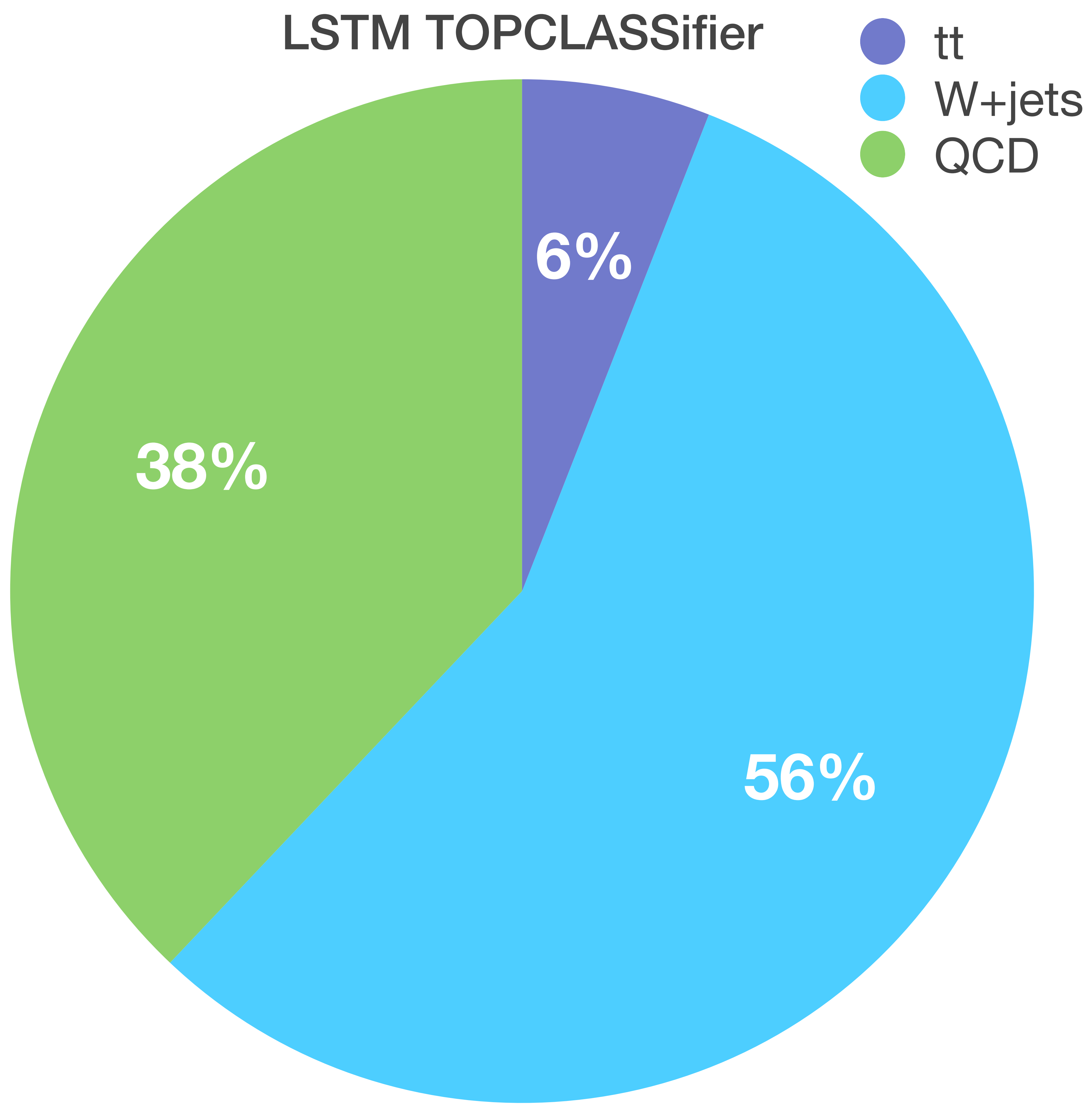
What is the network learning?

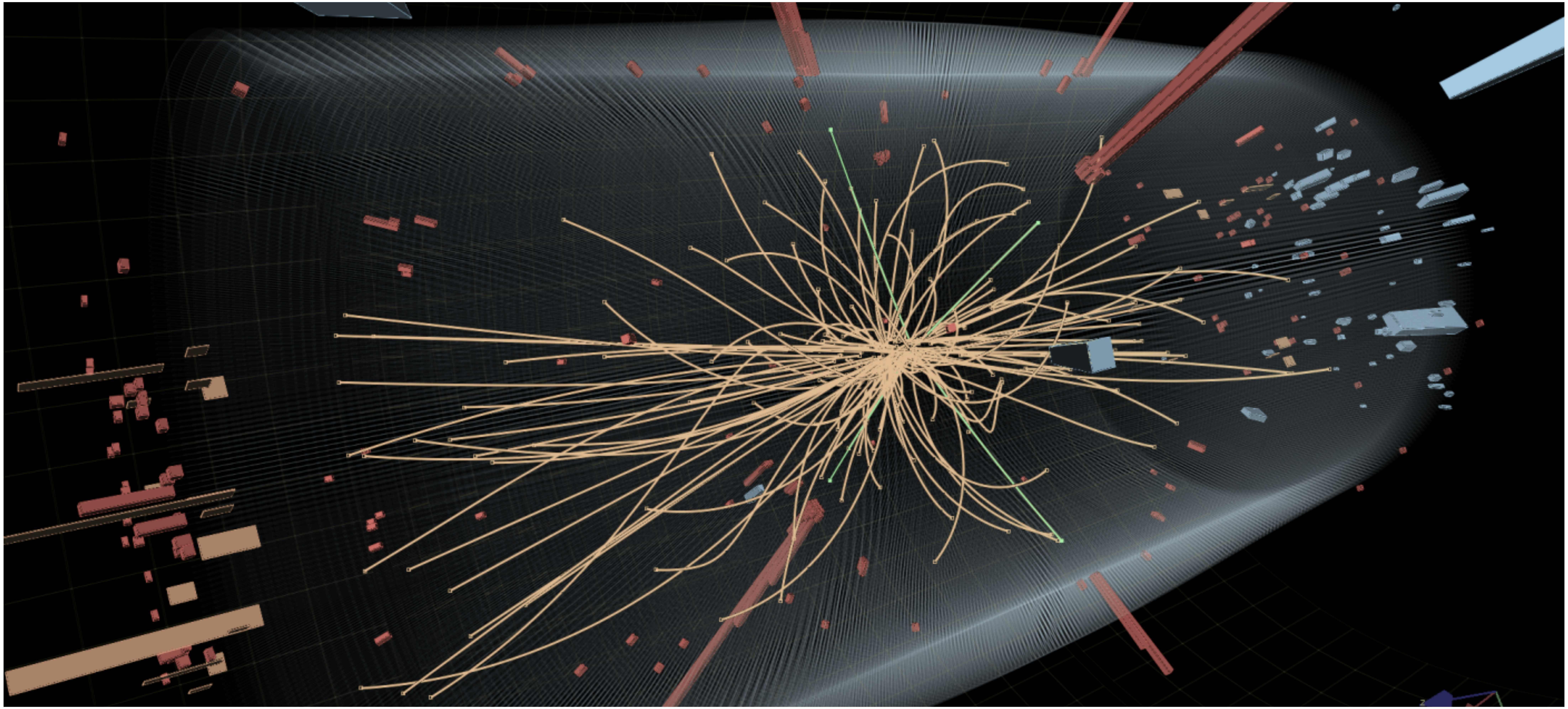
- tt events are more crowded that W events
- leptons in W and tt events are isolated from other particles

Cleaning up selected sample

A typical example: leptonic triggers

- *at the LHC, producing an isolated electron or muon is very rare. Typical smoking gun that something interesting happened (Z,W,top,H production)*
- *Triggers like those are very central to ATLAS/CMS physics*
- *The sample selected is enriched in interesting events, but still contaminated by non-interesting ones*
- *Contamination can be reduced with a DL classifier that rejects obvious false positives looking at the full event, not just at the lepton*

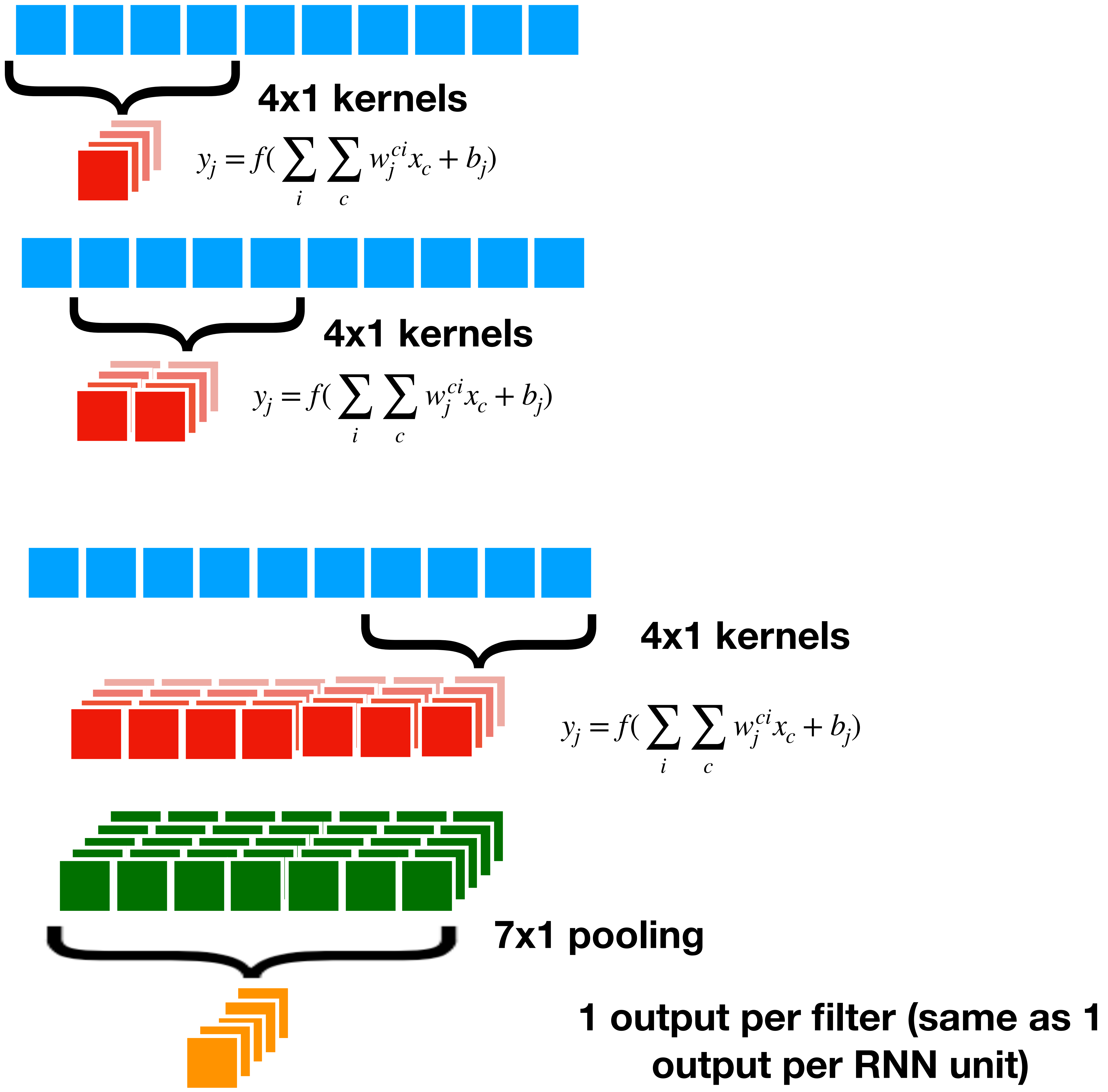




Alternative approach with 1D CNN

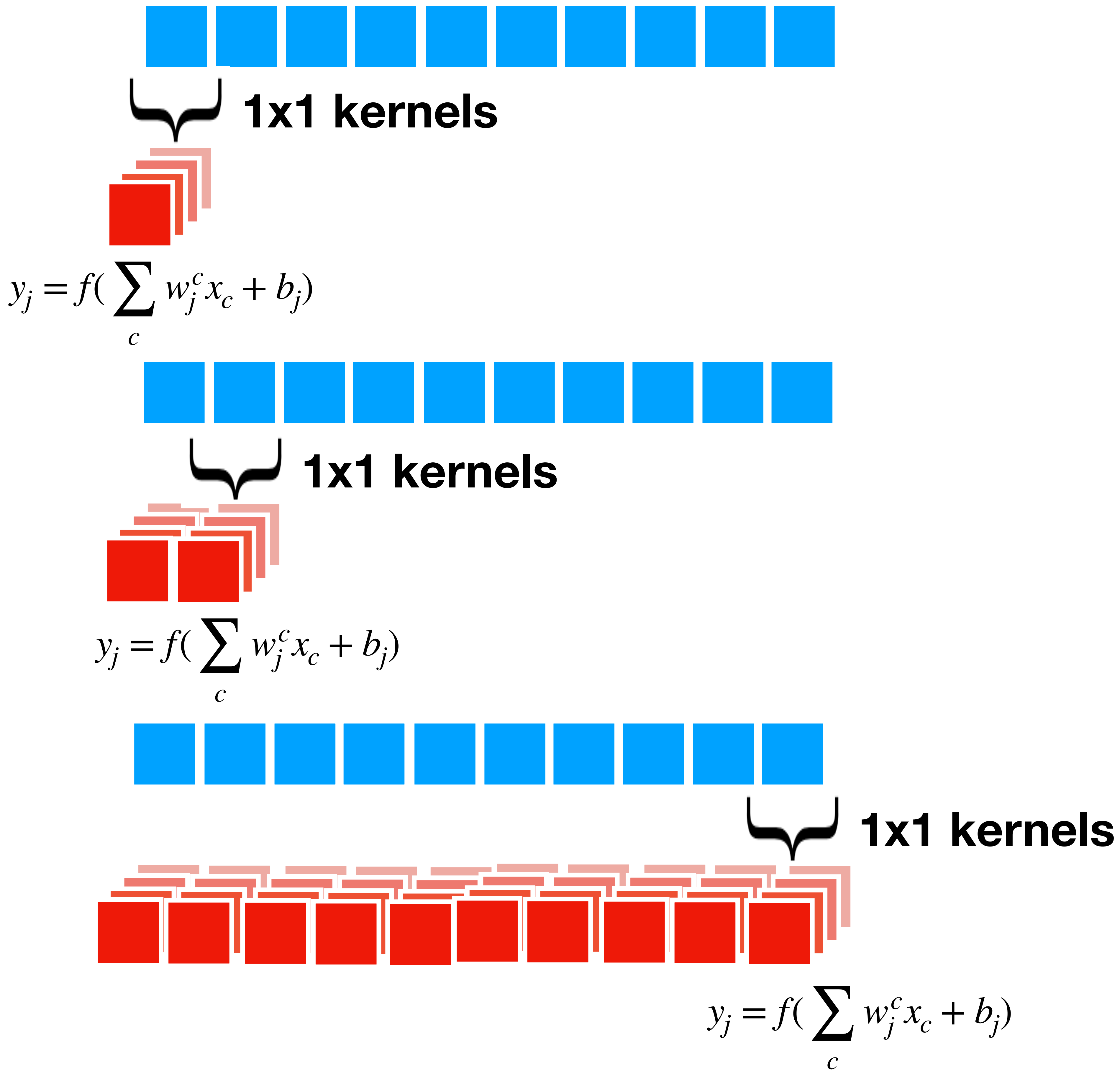
1D Convolution

- *Rather than a sequential processing, one could use a 1D kernel*
  - *Same as 2D conv, but acting on 1D sequence*
  - *The channels here are the features of the i-th element of the sequence (e.g., pT, η, φ of jet constituents)*
  - *Advantage comes from parallel processing (no memory cell) -> faster*
  - *Performance can be comparable, but usually LSTMs are better*



1x1 kernel for pre-processing

- *Sometimes, 1x1 kernels are used (in 1D, 2D, etc) as pre-processing networks*
- *If you look at the math, it's the same as running a DNN in parallel on the c features of each element*



Summary

- We discuss recurrent NNs and their potential usage to save processing resources @LHC
  - Learn underlying "particle physics grammar" by sequential processing of ordered lists of particles
  - Can be used in jets (our exercise of today)
  - Can be used on whole event
  - A step fwd wrt CNNs: don't require regular detector geometry
  - Not yet the ultimate solution: require ordering principle, not always uniquely defined
- We also saw 1D CNN as an alternative solution to the same problem
  - parallel process -> faster
  - no memory gate mechanism -> might not work that well