The Ottawa Hospital | L'Hôpital d'Ottawa
RESEARCH INSTITUTE | INSTITUT DE RECHERCHE

uOttawa | Institut de recherche sur le cerveau
Brain and Mind Research Institute

Workshop on Applied Deep Learning in Intracranial Neurophysiology

Part 5 – Introduction to Recurrent Neural Networks

June 21, 2019

Presented by Chadwick Boulay, MSc, PhD

Sachs Lab

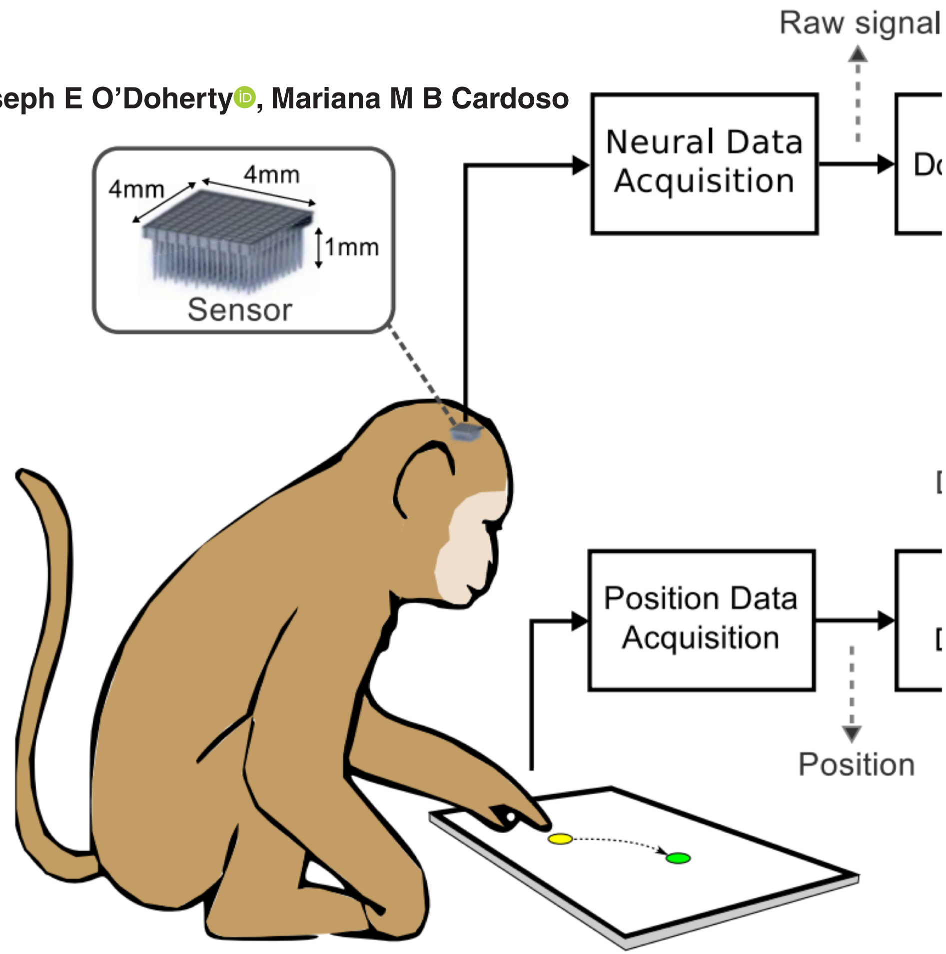
Superior arm-movement decoding from cortex with a new, unsupervised-learning algorithm

Joseph G Makin[1], Joseph E O'Doherty, Mariana M B Cardoso and Philip N Sabes

Dataset

2 96-ch Utah arrays: M1 and S1

- 24.4 kHz
- BP filt 500-5000 Hz
- Thresholded at 3.5-4.0 STDs

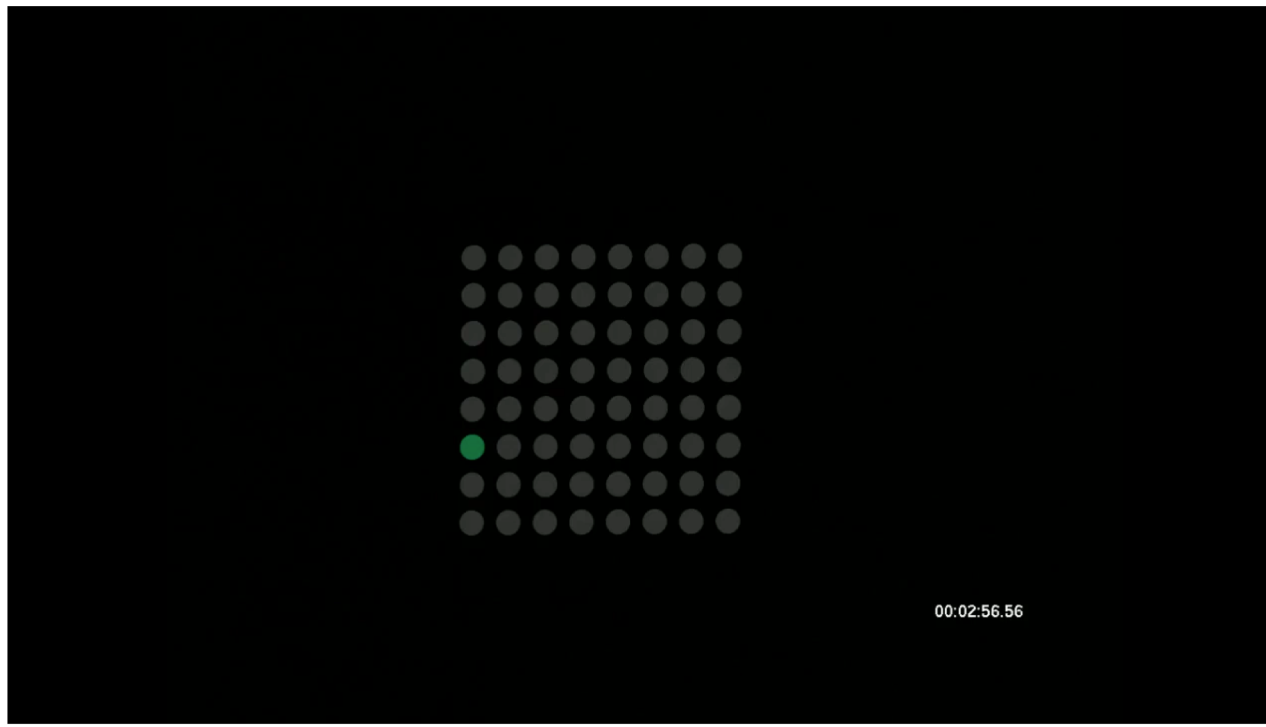


XYZ finger position recorded with Polhemus (magnetic) at 250 Hz

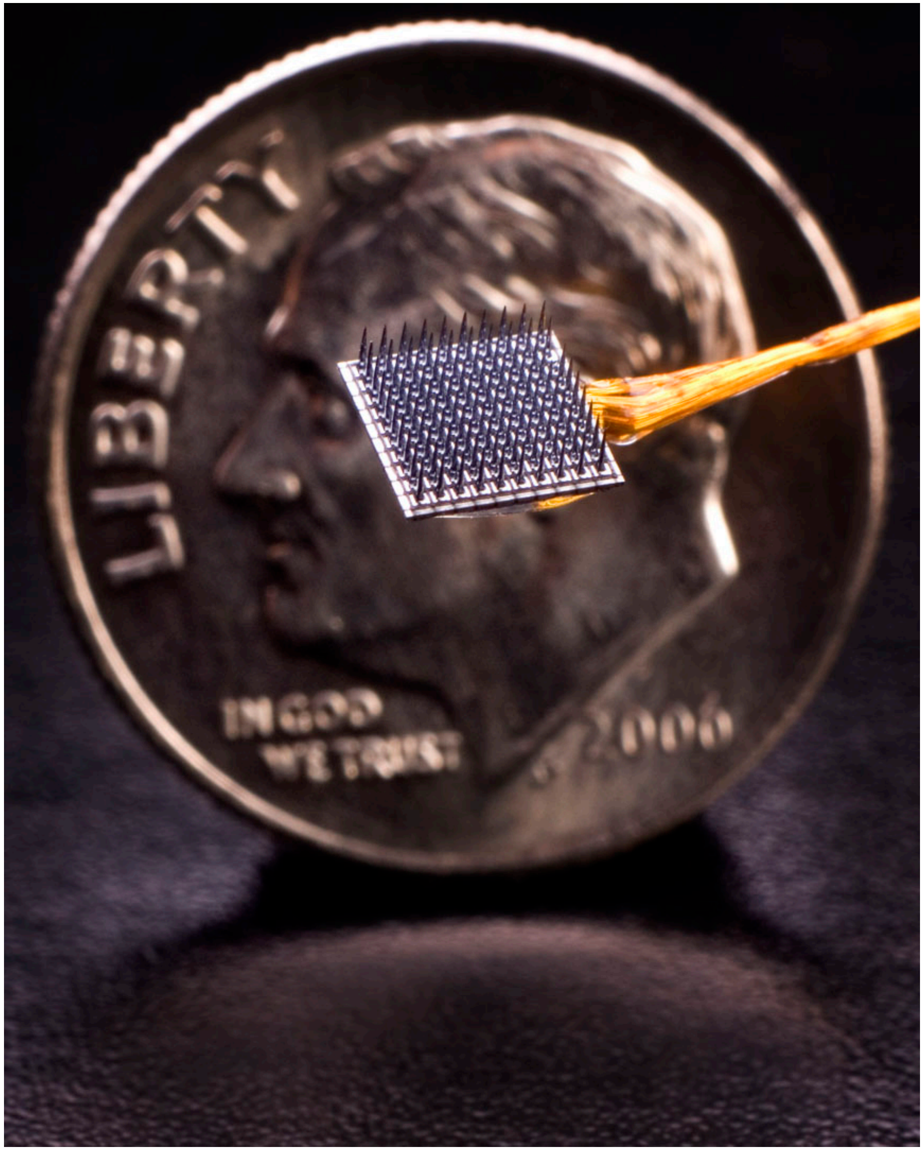
[Available online]

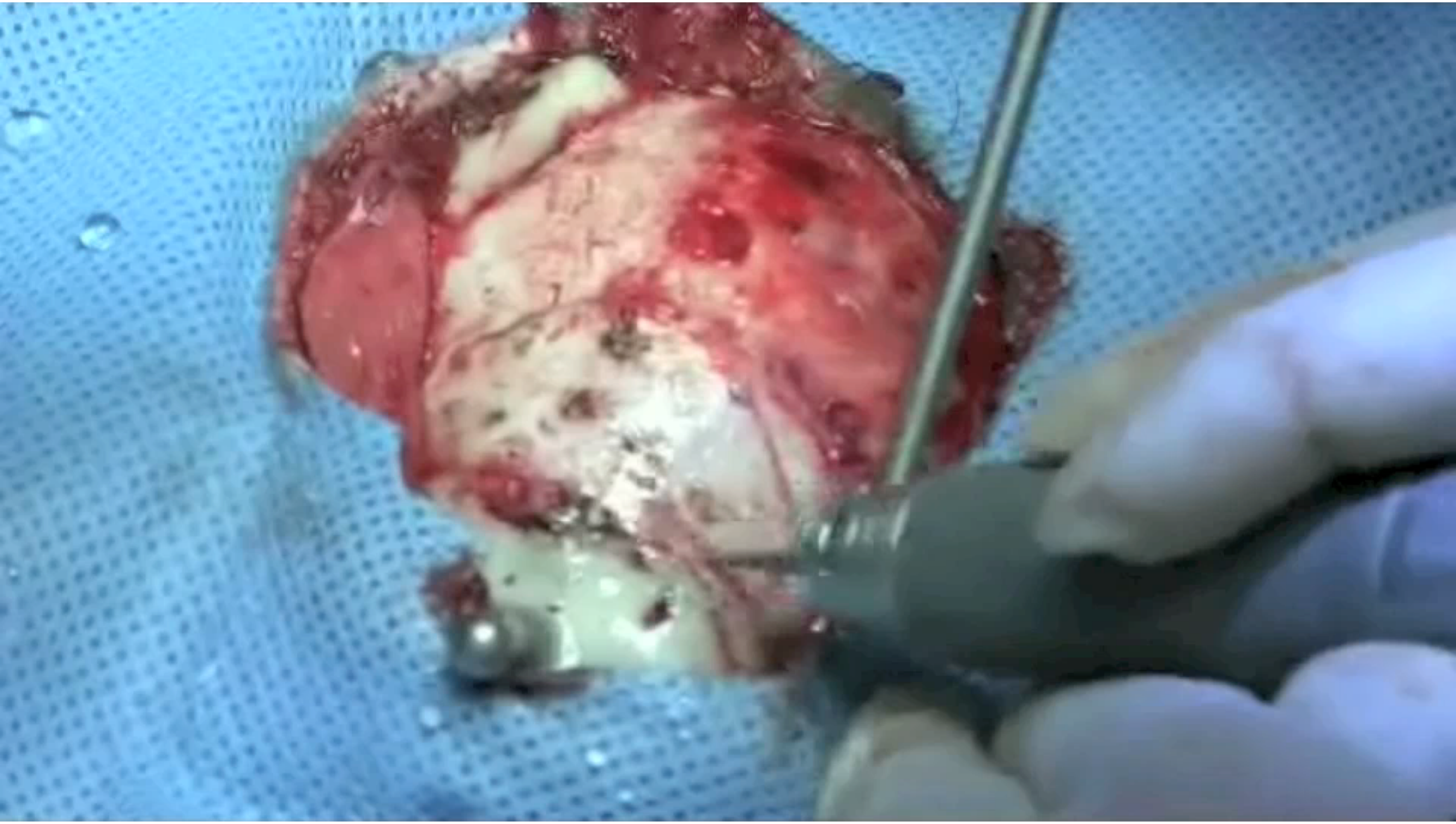
3D finger position projected onto 2D plane

$$\begin{pmatrix} 0 & 0 \\ -10 & 0 \\ 0 & -10 \end{pmatrix}$$



Utah Array



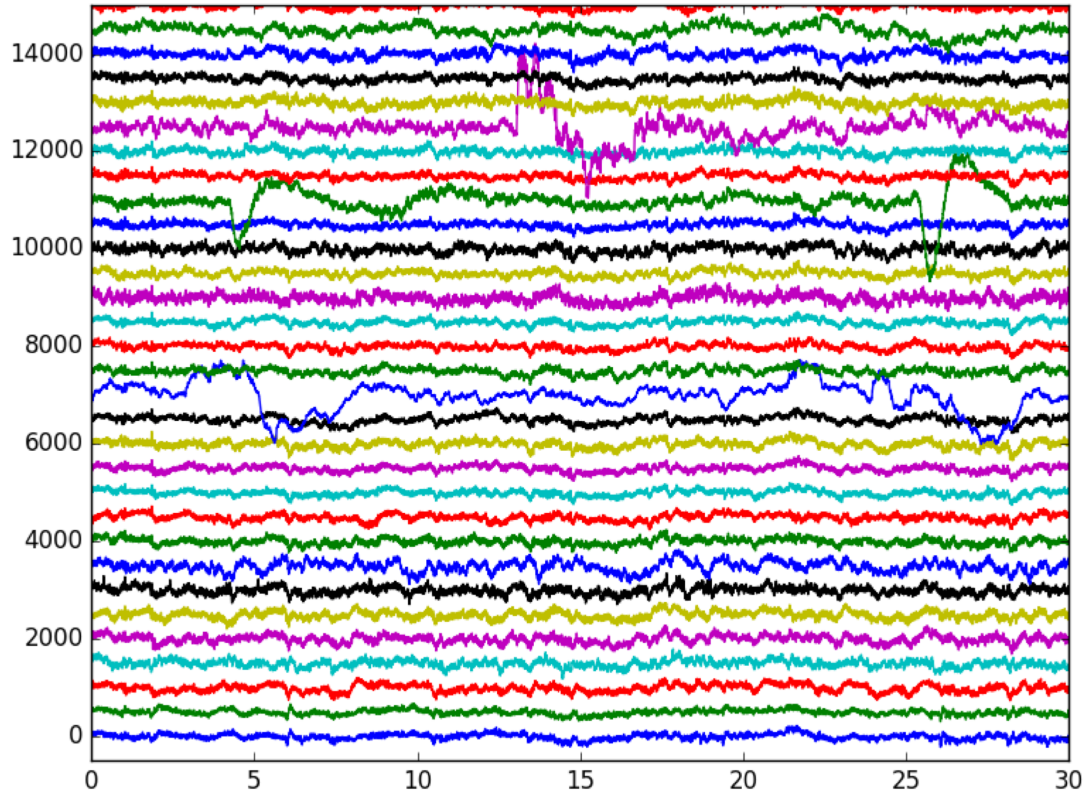


• Data contents:

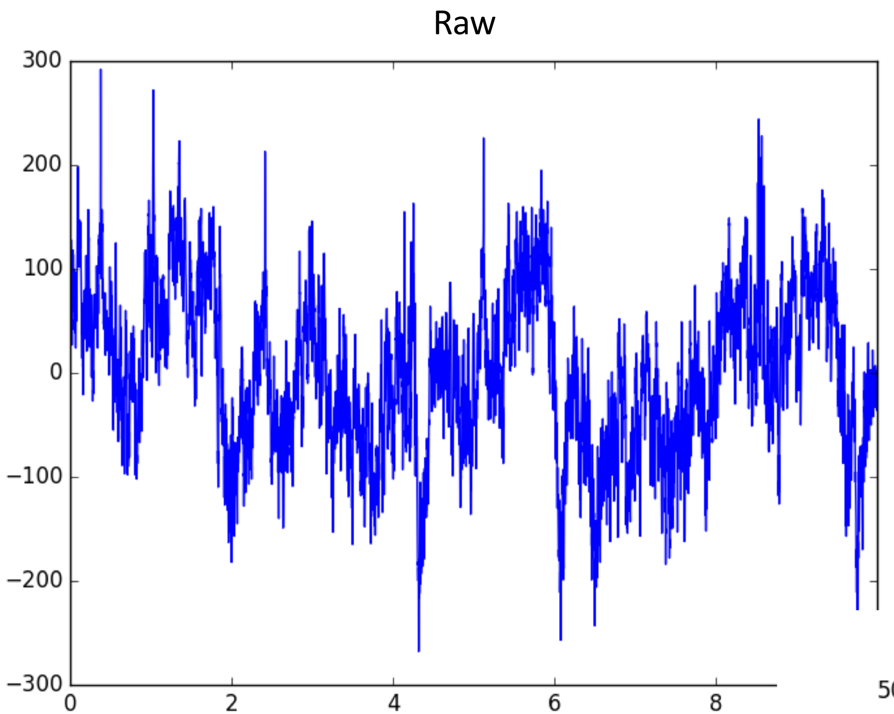
- Events

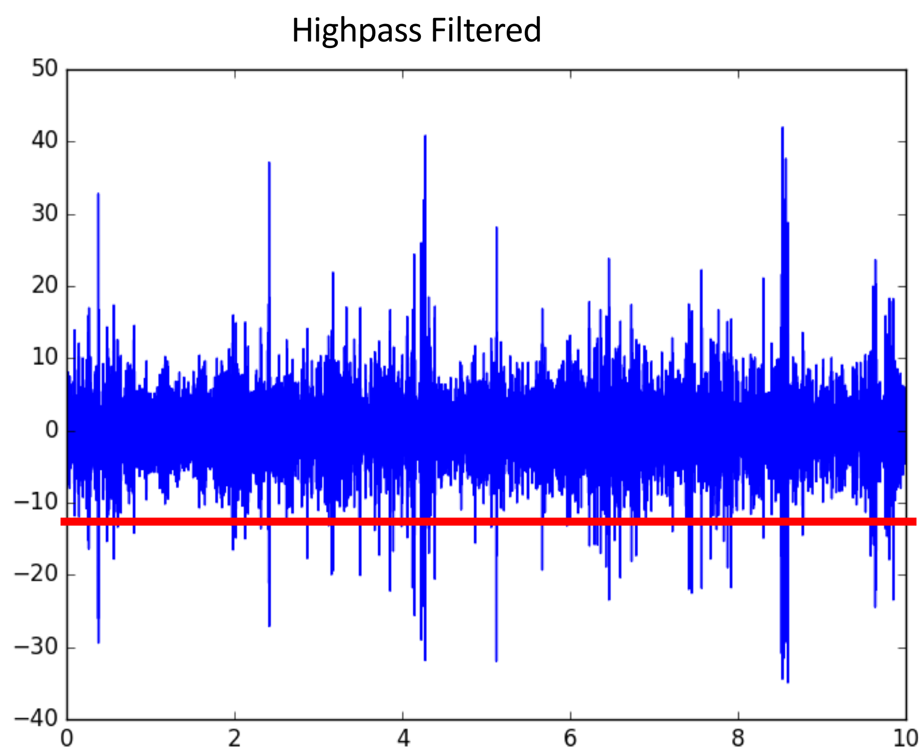
```
events = [(timestamp, event_string), ...]
```

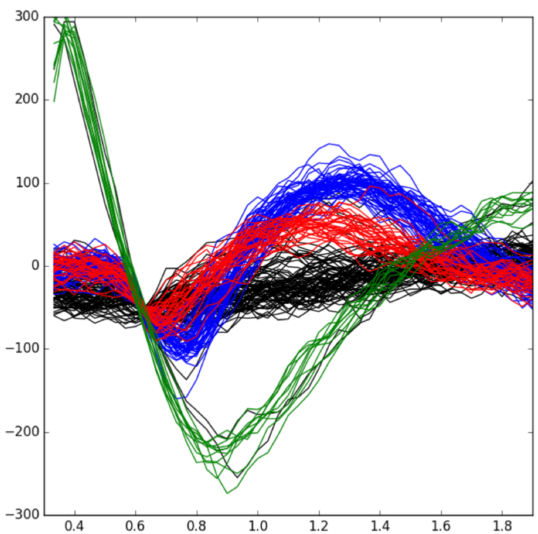
- Broadband Field Potentials
    - Low-pass filter
        - Downsample (decimate) to get LFP
    - High-pass filter
        - Get threshold crossing
        - Get spike waveforms



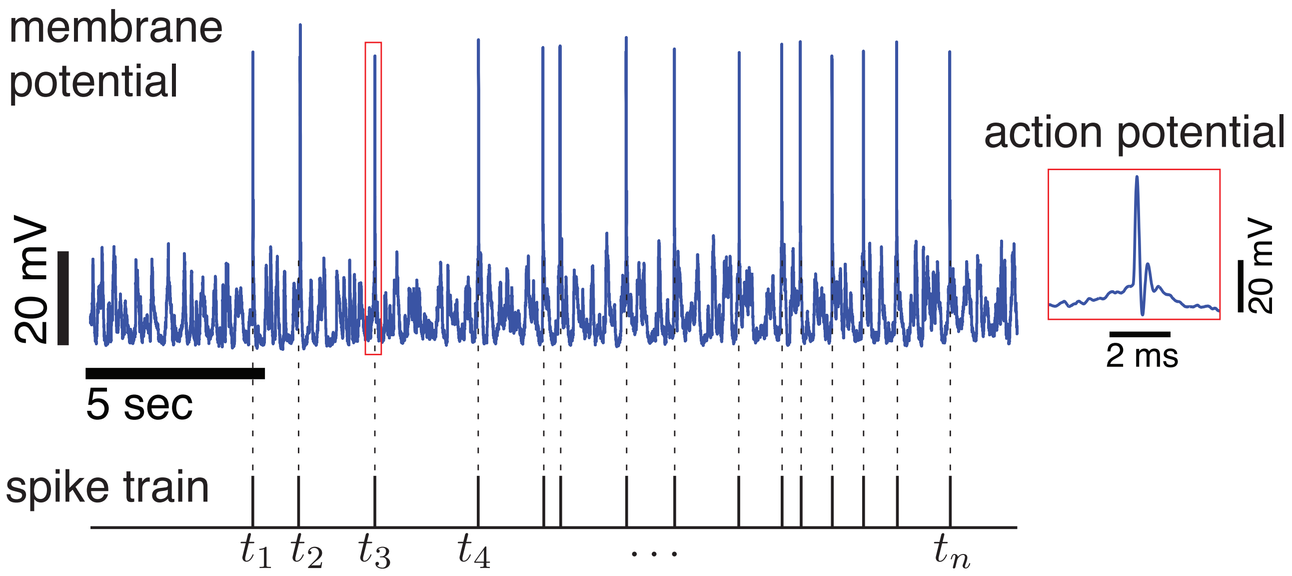
Preprocess





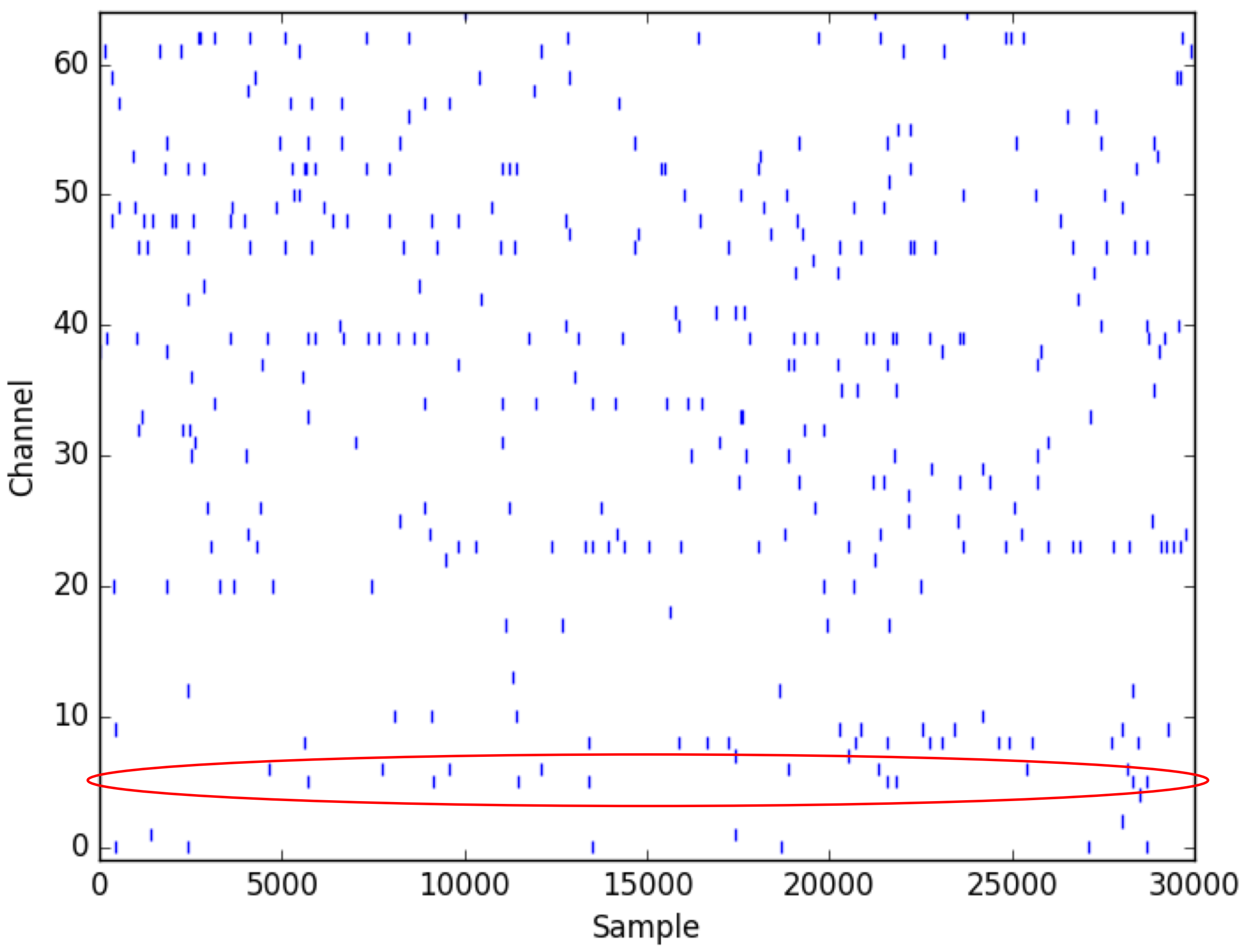


~ -3.5 – -4 SD

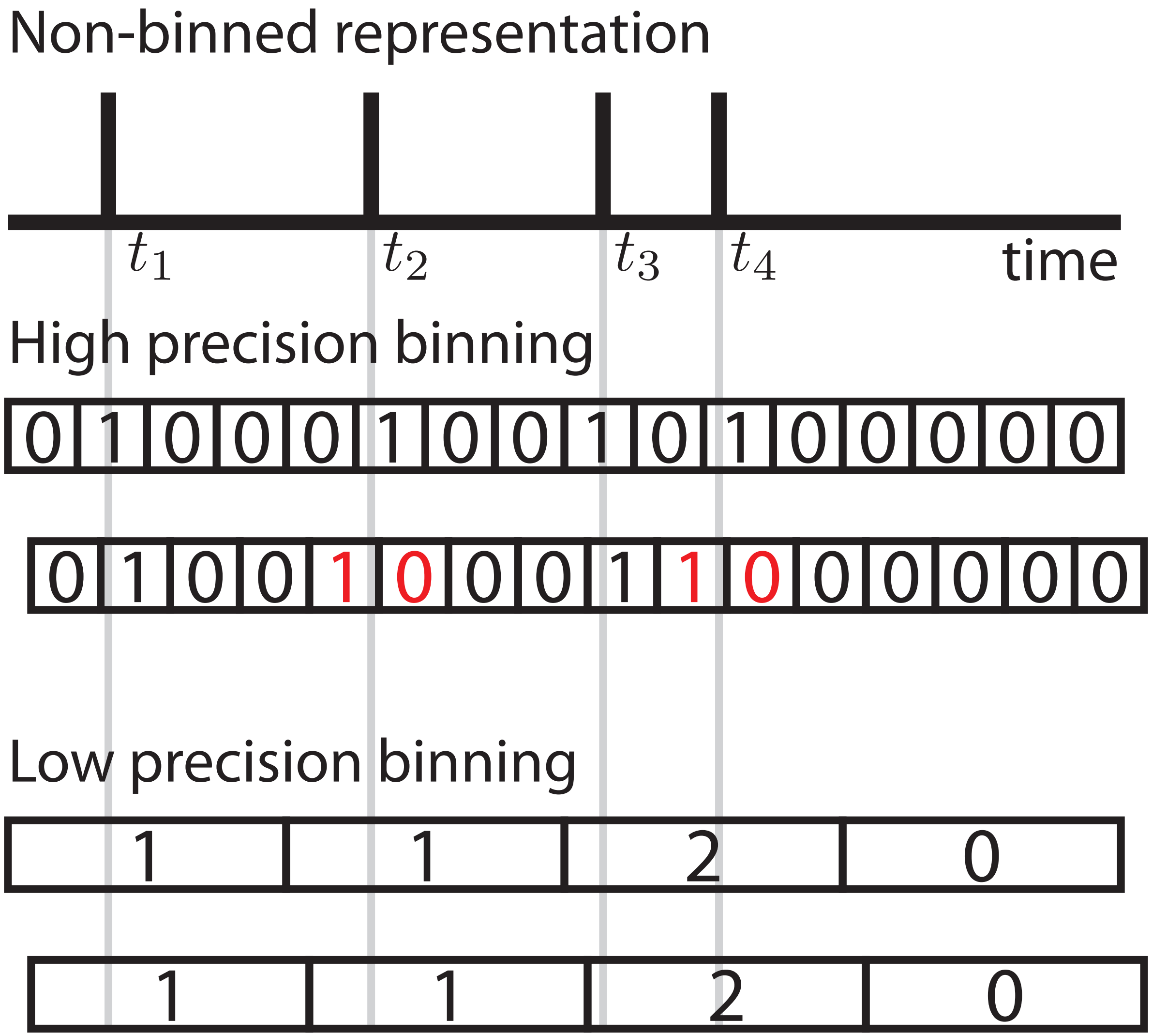


Park et al., IEEE Sig. Proc. Mag. 2013

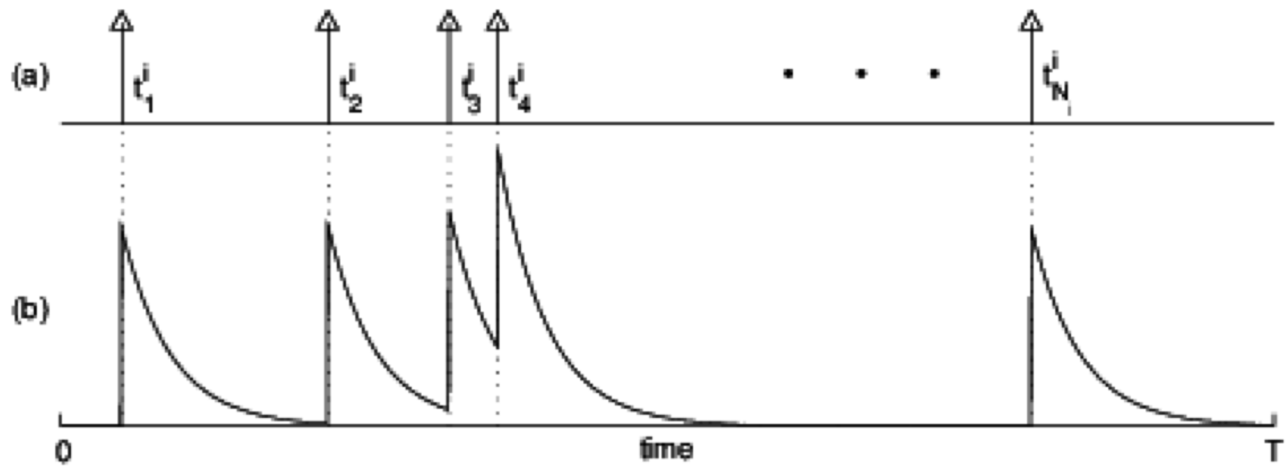
Spike Trains



Rate coding – Binned spike counts



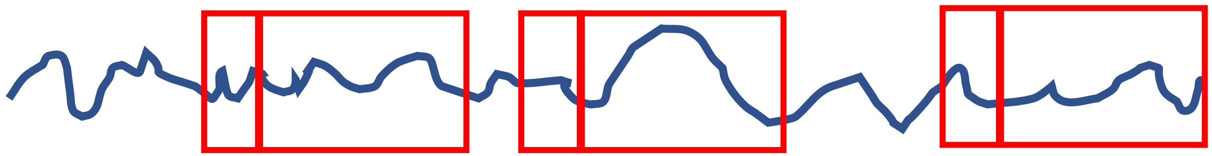
Rate coding – Kernel convolution

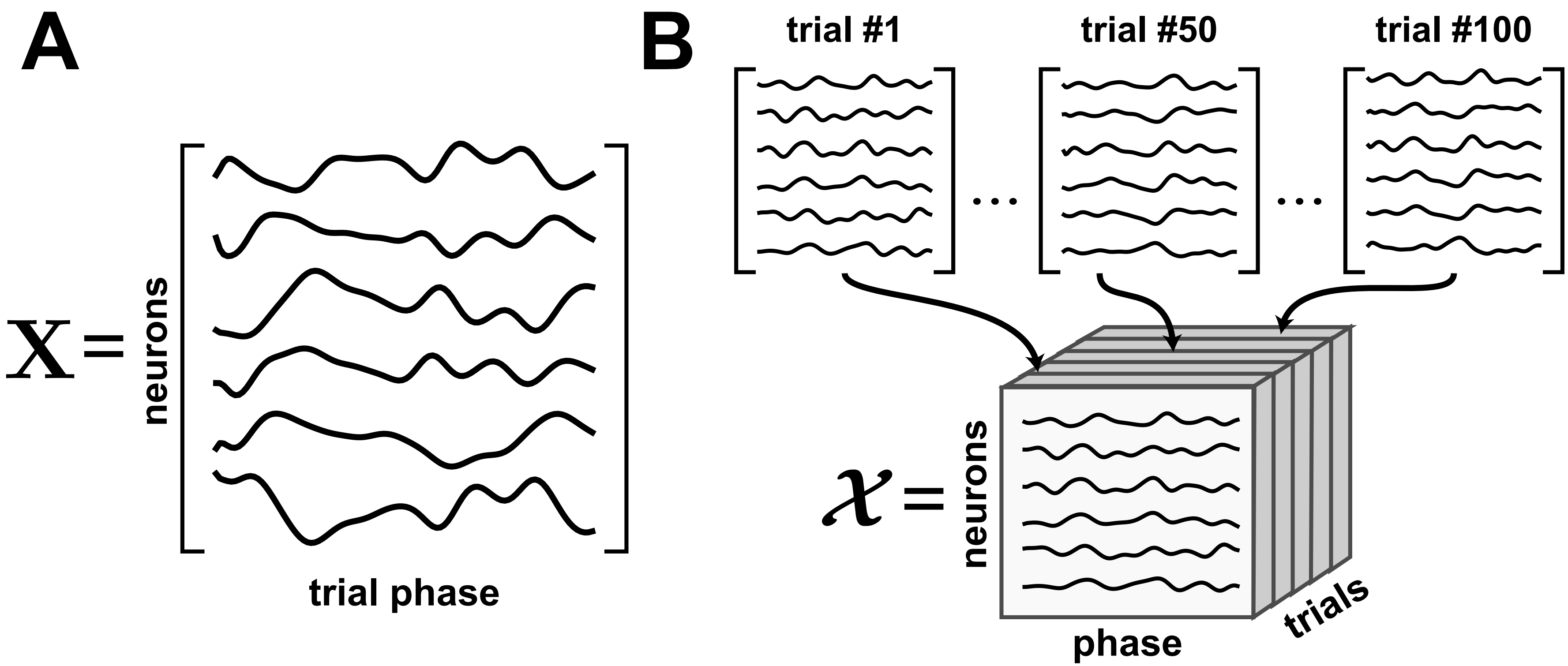


Paiva et al., Neural Computing and Applications, 2010

Kernels:
'auto', **'rectangular'**, **'gaussian'**, **'exponential'**, **'alpha'**

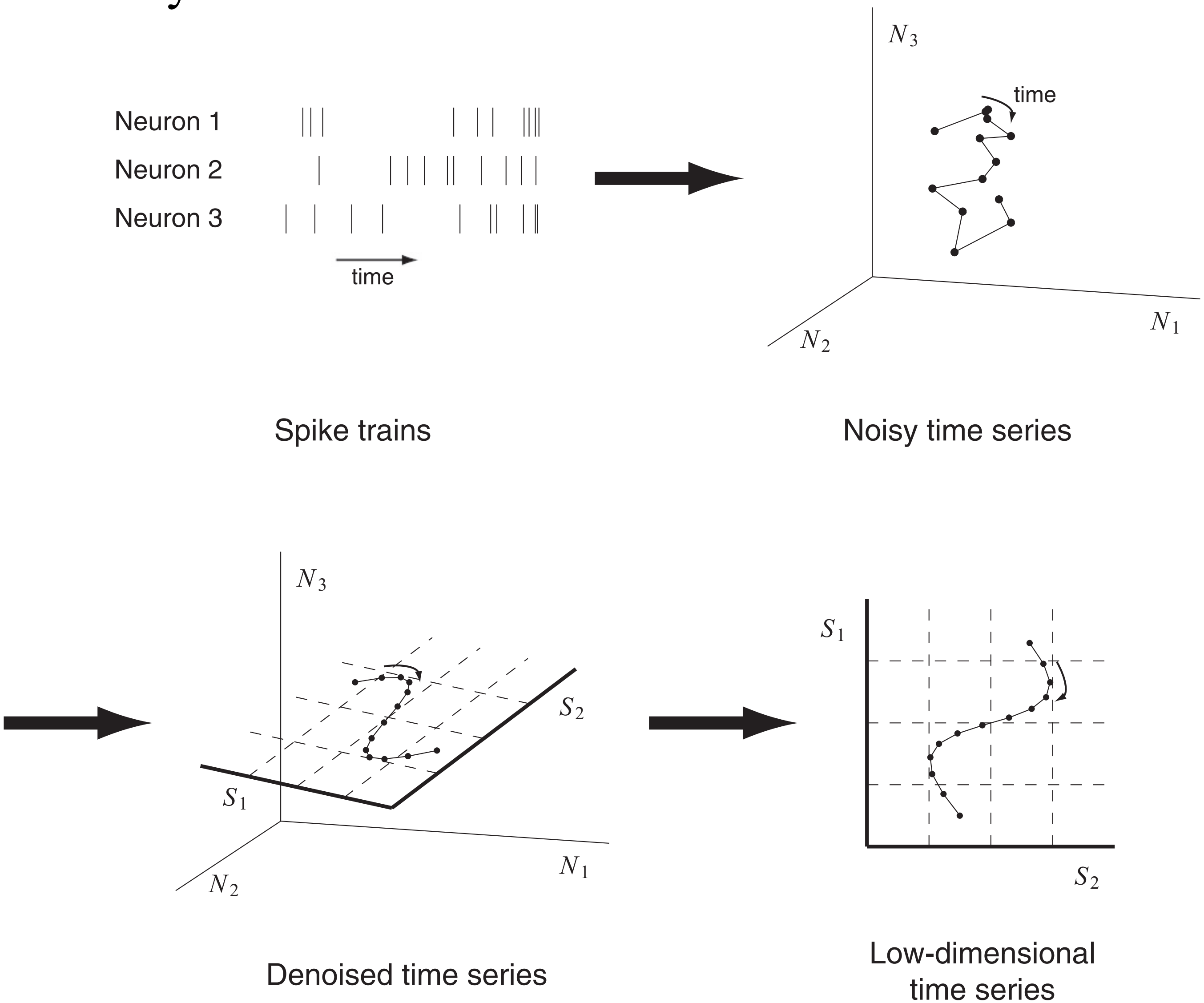
Auto: Gaussian with optimized standard deviation
H. Shimazaki and S. Shinomoto, "Kernel Bandwidth Optimization in Spike Rate Estimation," in Journal of Computational Neuroscience 29(1-2): 171–182, 2010




A
X =
neurons
trial phase
B
trial #1
trial #50
trial #100
…
…
𝒳 =
neurons
phase
trials

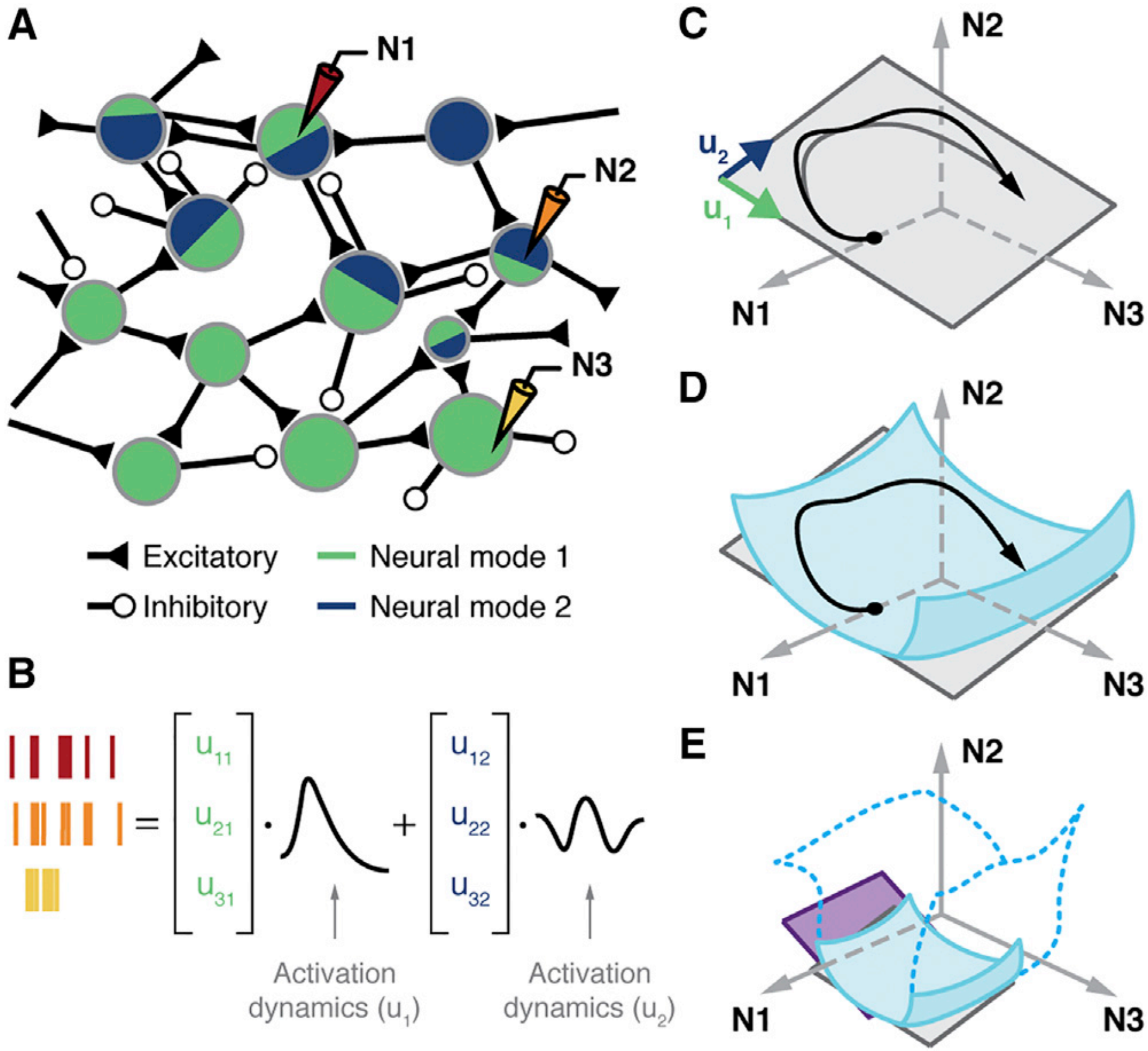
“tensor”

Standard Approach:
Dimensionality reduction



Yu et al., J Neurophysiol. 2009

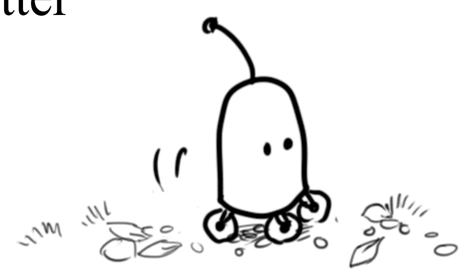
Standard Approach:
Dimensionality reduction



Gallego et al., Neuron 2017

Standard Approach:

- State Estimation (Kalman Filter)
  - Uncertain Information → Educated Guess about What System Will do Next
  - Example:
    - Robot moving in one direction with **fairly** constant speed
    - It has a GPS which tells us the exact position with 10 meters **uncertainty**
    - We define its state as $\overrightarrow{x_k} = (\vec{p}, \vec{v})$
    - Kinematics **OR** GPS ?
    - Using all available information would give us a better answer than either estimation



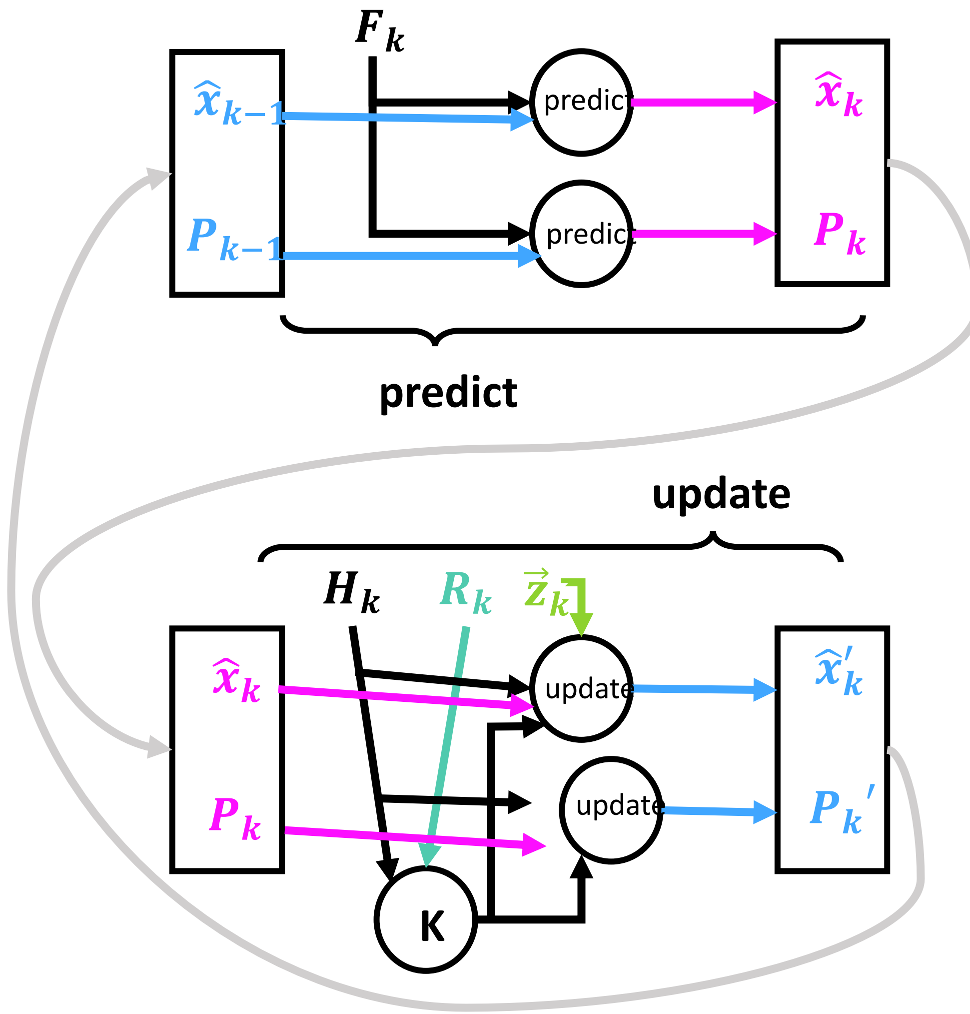
State Vector (x) comprises

- Position
- Velocity
- Acceleration

Predict next step (position = velocity * time_delta)

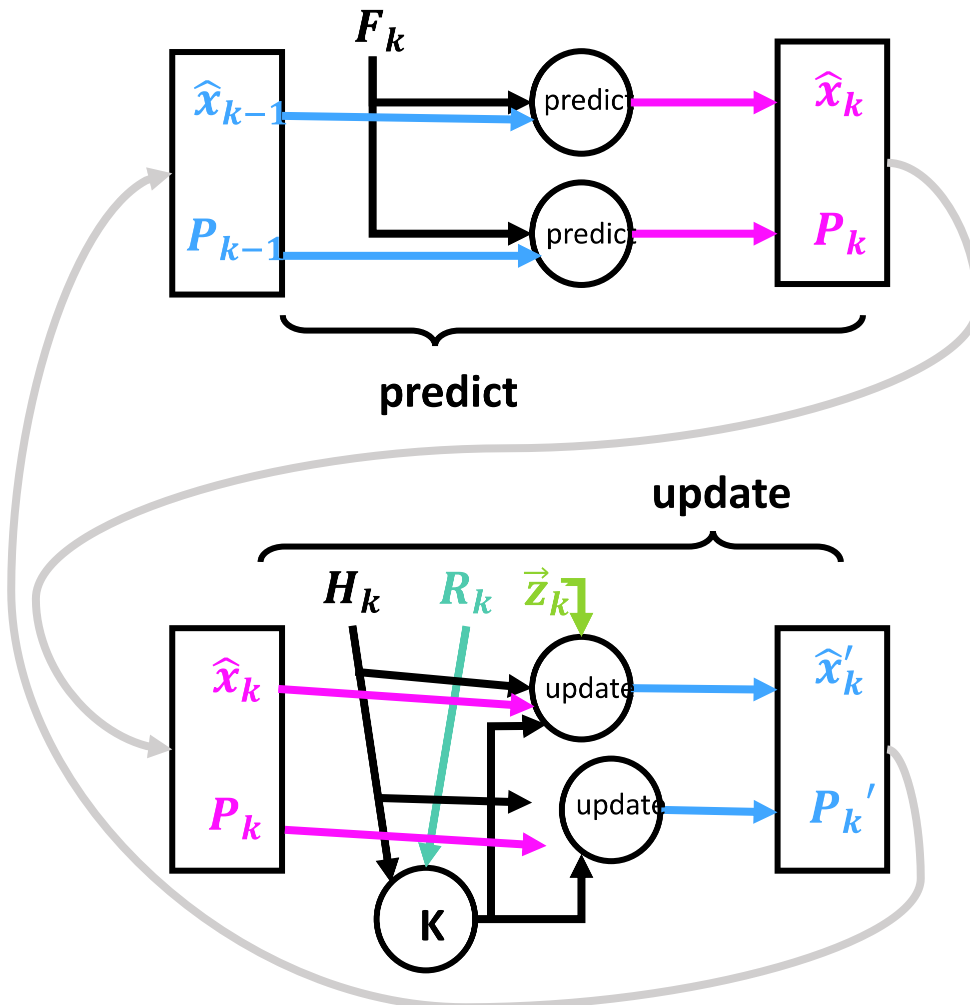
Update with neural input (z)

Optional: z can include history with previous time bins (taps)



Source: bzarg.com

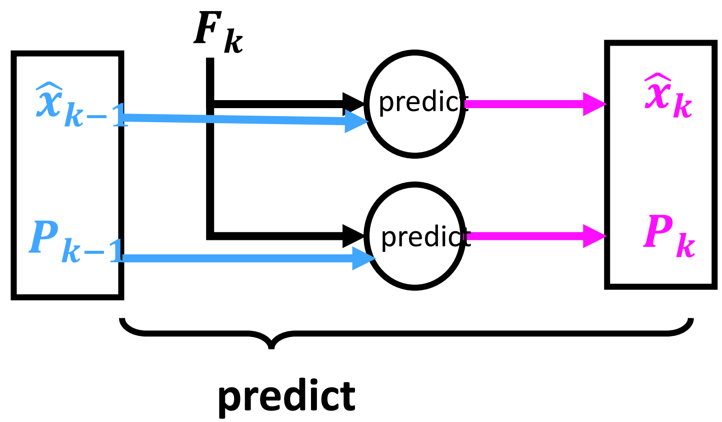
- State Estimation (Kalman Filter)



- **Predict**
- $\hat{x}_k = F_k \, \hat{x}_{k-1}$
- $P_k = F_k \, P_{k-1} F_k{}^T$

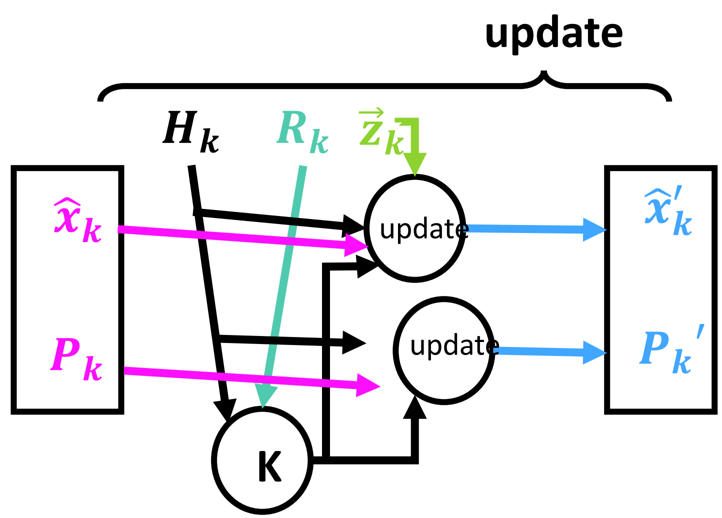
- **Update**
- $\hat{x}'_k = \hat{x}_k + K'(\vec{z}_k - H_k \, \hat{x}_k)$
- $P_k{}' = P_k - K' \, H_k \, P_k$
- $K' = P_k \, H_k{}^T (H_k P_k \, H_k{}^T + R_k)^{-1}$

- State Estimation (Kalman Filter)
  - **Predict**
  - New estimate is predicted from previous best estimate
  - New uncertainty is predicted from old uncertainty



  - **Update**
  - New best estimate is predicted from new estimate and sensor input
  - Overall new uncertainty is predicted from new uncertainty and

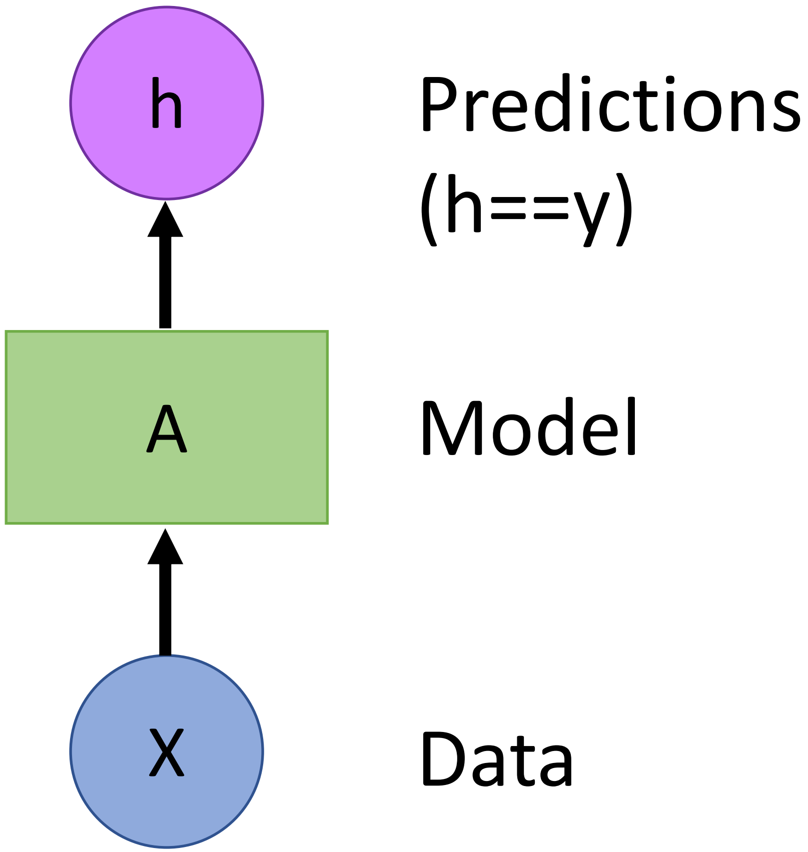
    sensor noise



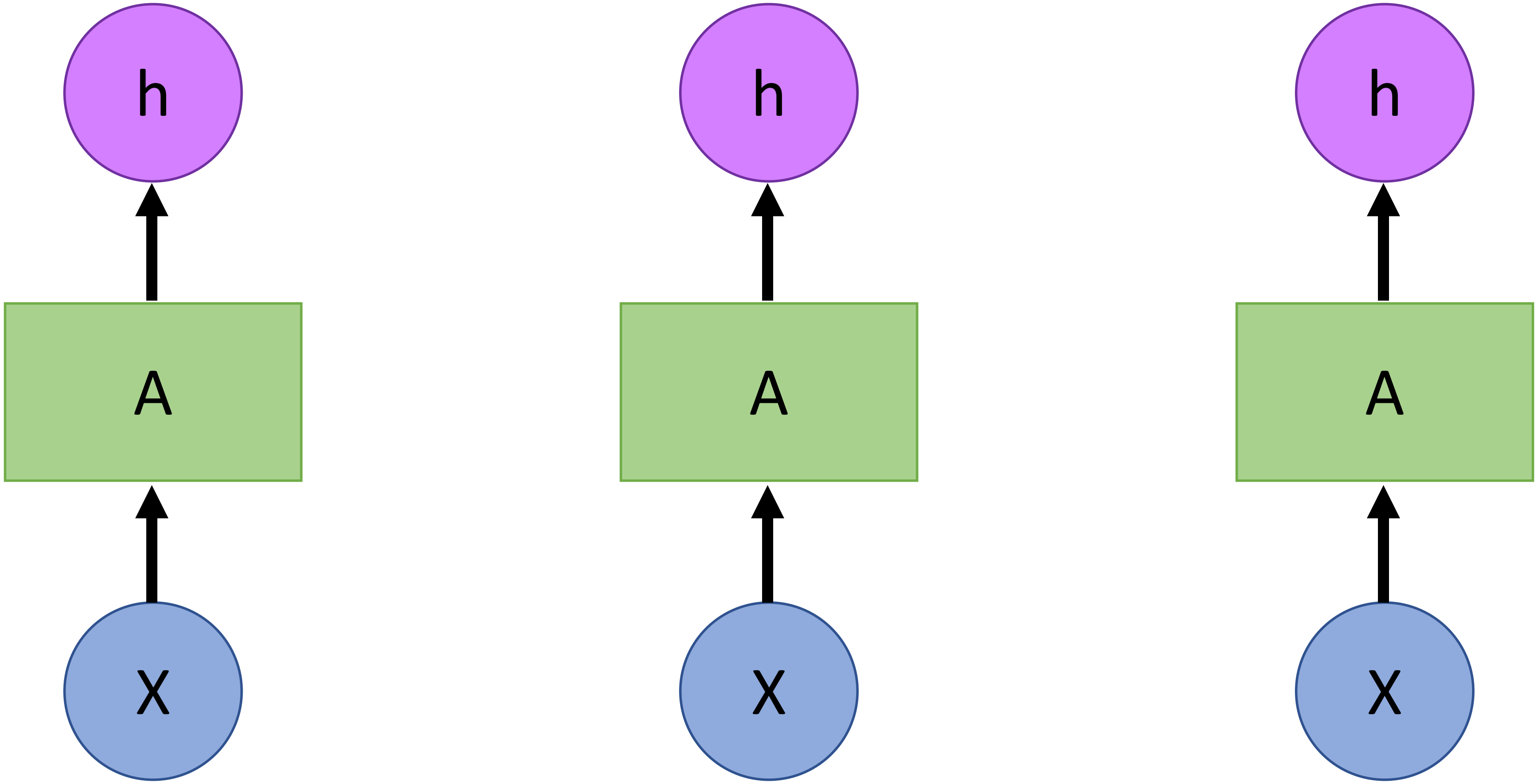
❖ F_k is how new prediction is related to previous (i.e. Kinematics)

❖ H_k is how we measure our state from sensor inputs

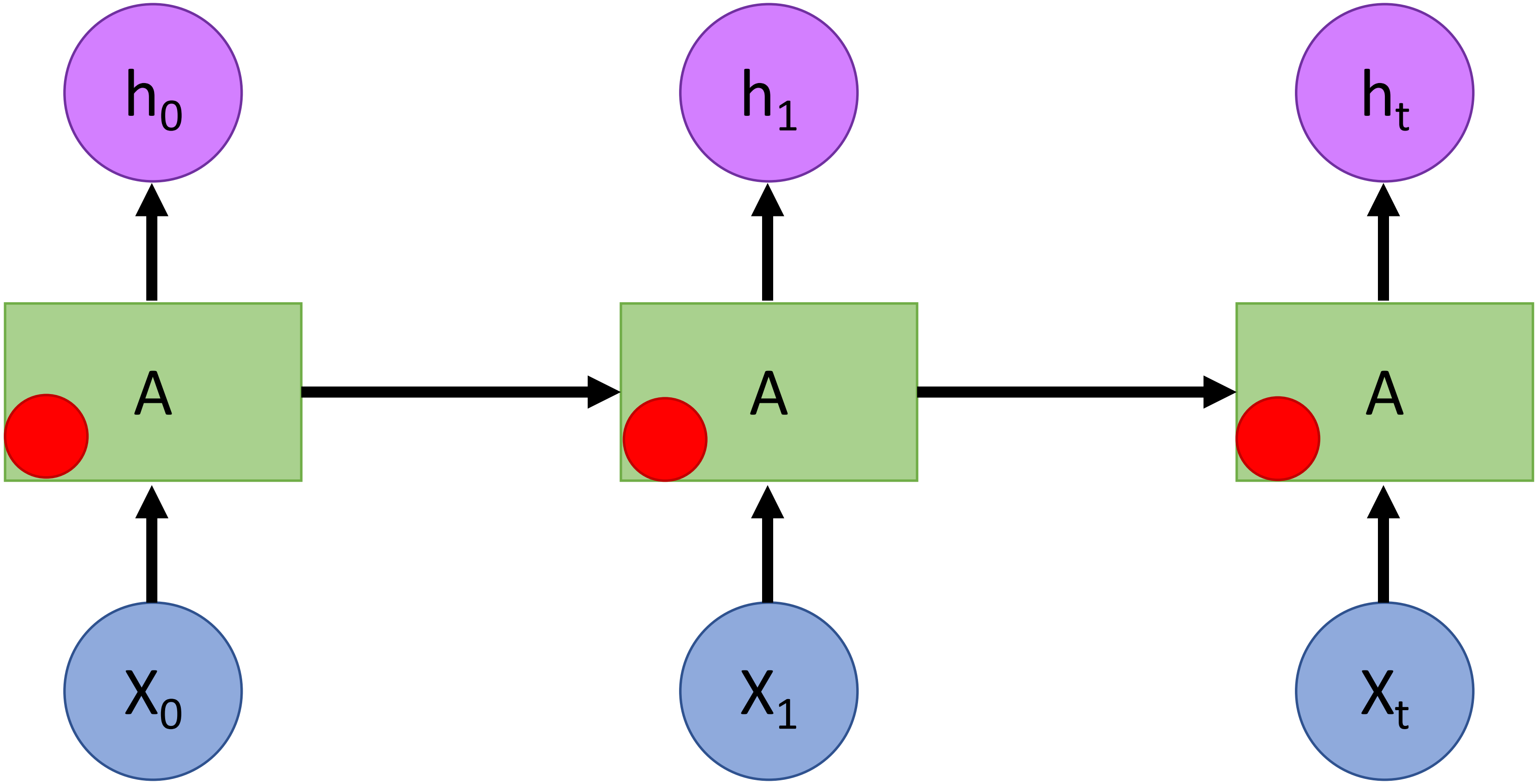
Vanilla CNN is stateless: no memory



Vanilla CNN is stateless: no memory



A model that remembers its (recent) data history is desirable.



Recurrent neural networks (RNNs) can use past information to make the current prediction

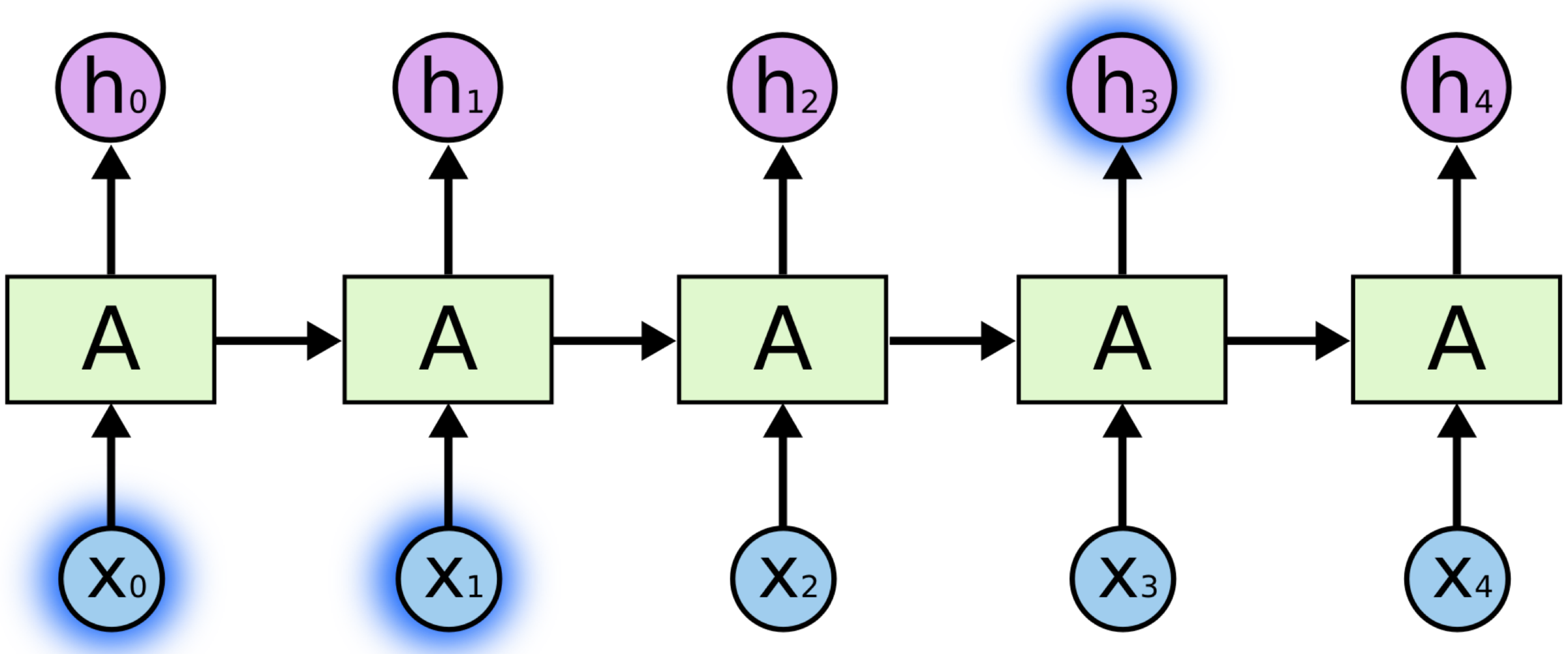
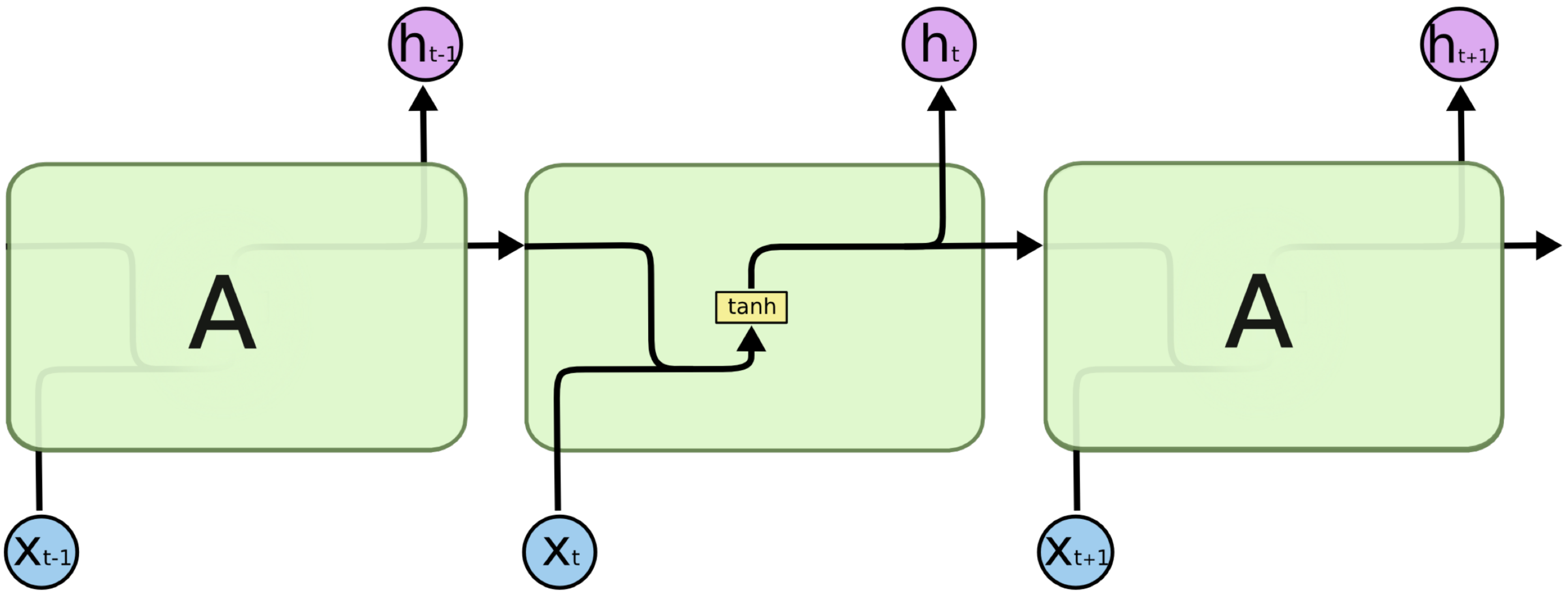
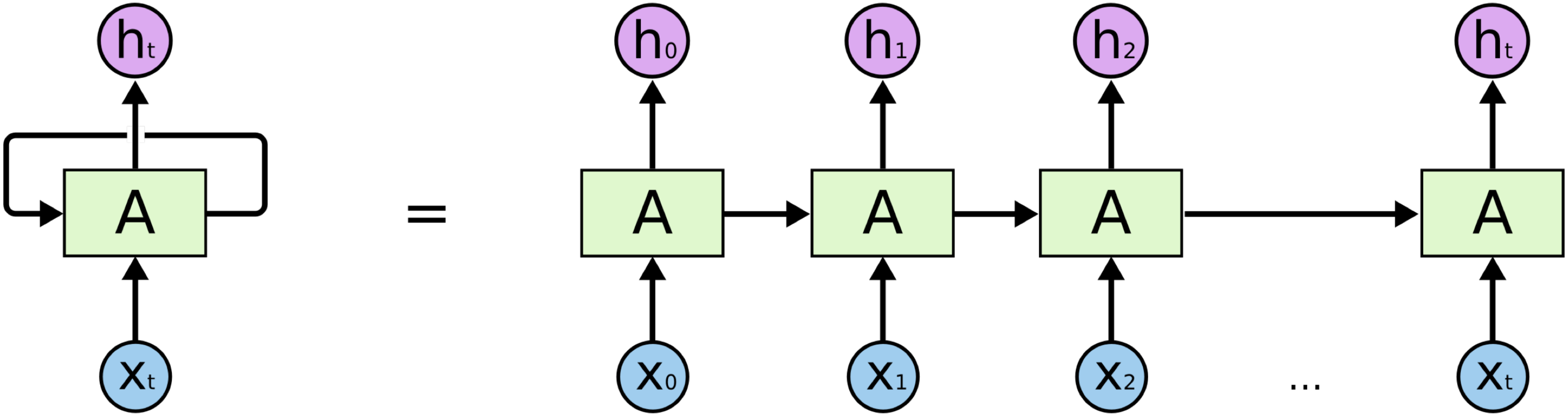


Figure source



h_{t-1}
h_t
h_{t+1}
A
tanh
A
X_{t-1}
X_t
X_{t+1}

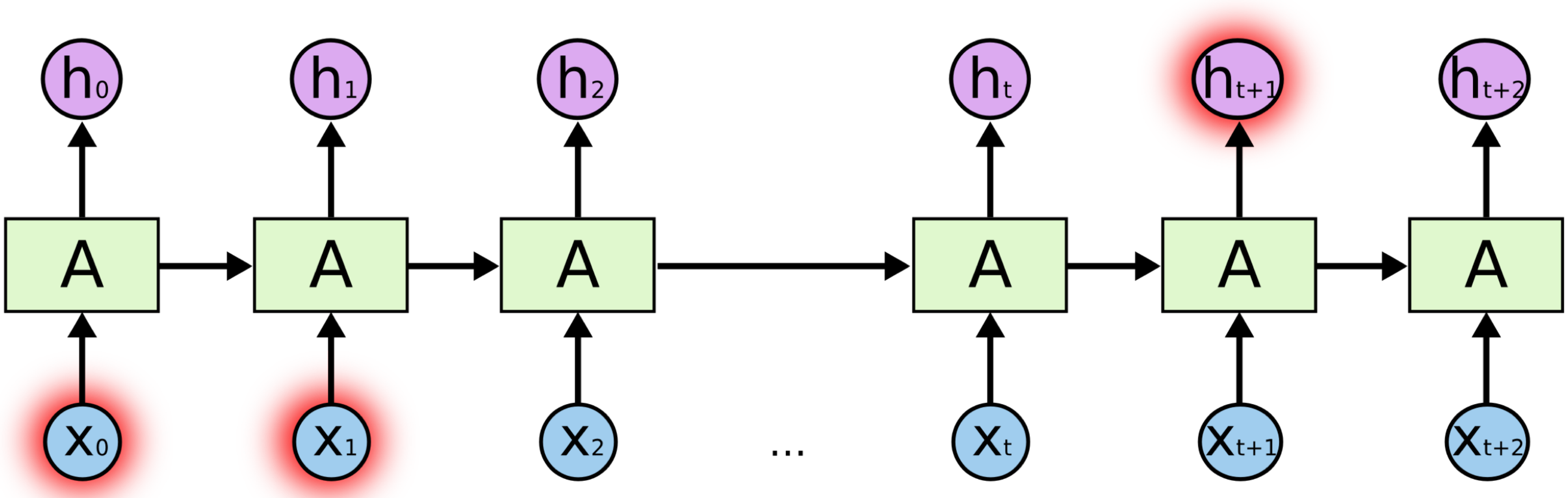
A model that maintains state and loops over its input is represented with a single looping model



- 05_01_intro_to_RNNs.ipynb
- When that’s done, open 05_02 and start downloading data, then resume here.

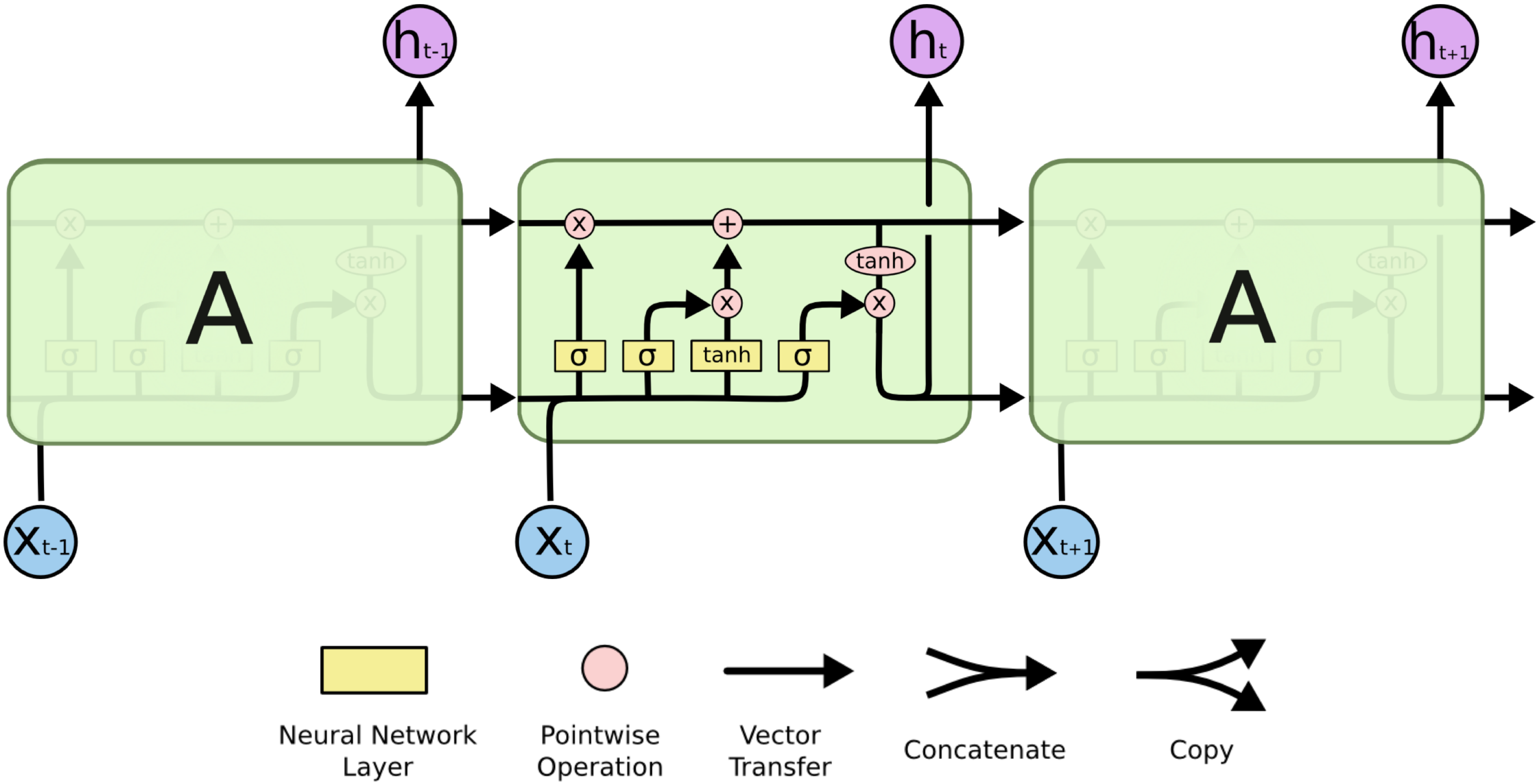
Learning Long-Term Dependencies with Gradient Descent is Difficult

Bengio et al., 1994

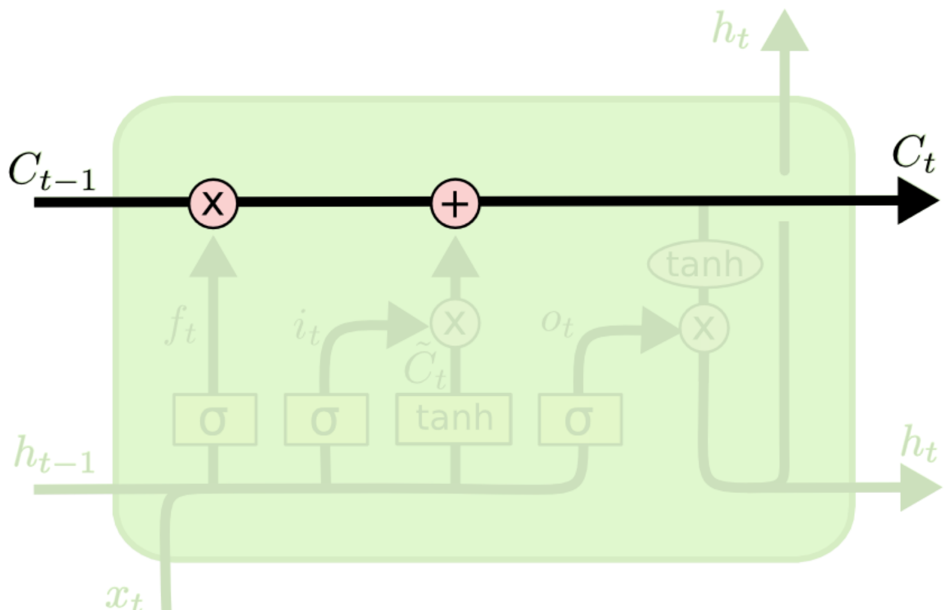


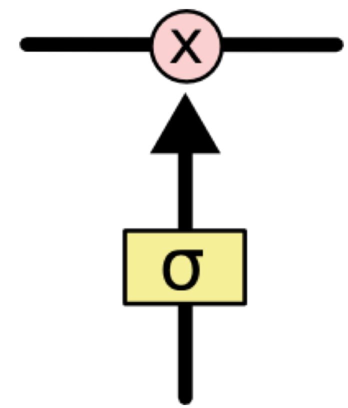
- In practice, simple RNNs cannot maintain information for more than a few steps
  - Exploding/Vanishing gradient

Long-Short Term Memory (LSTM) are a special kind of RNN capable of learning long(er)-term dependencies



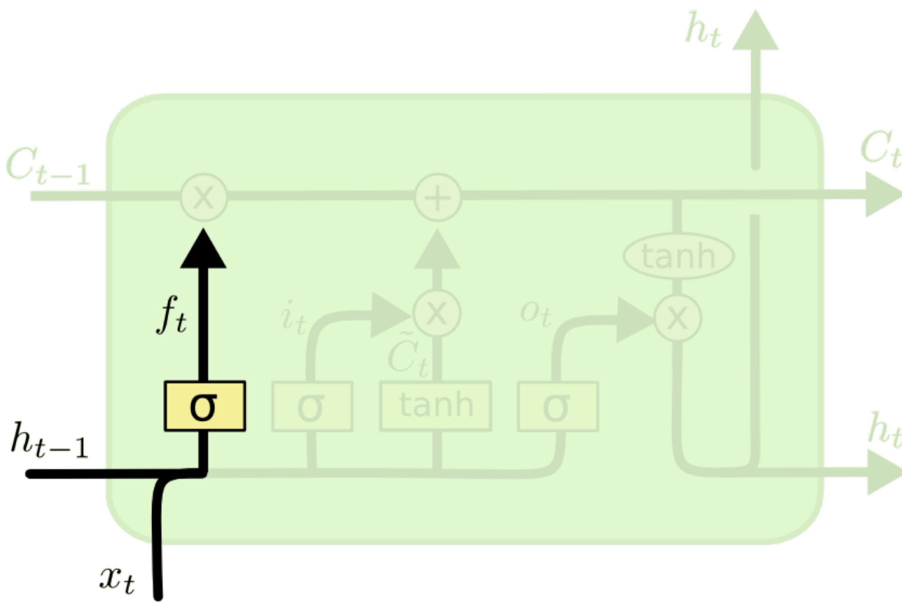
The cell state C persists across iterations. Its contents may be modified, determined by gates.





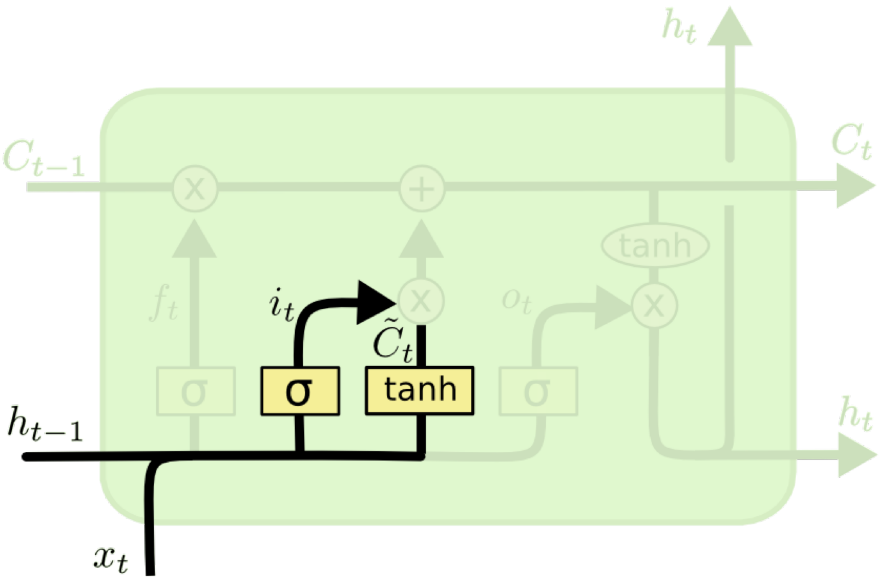
Sigmoid gating mechanism

LSTM Step 1: Forget Gate



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

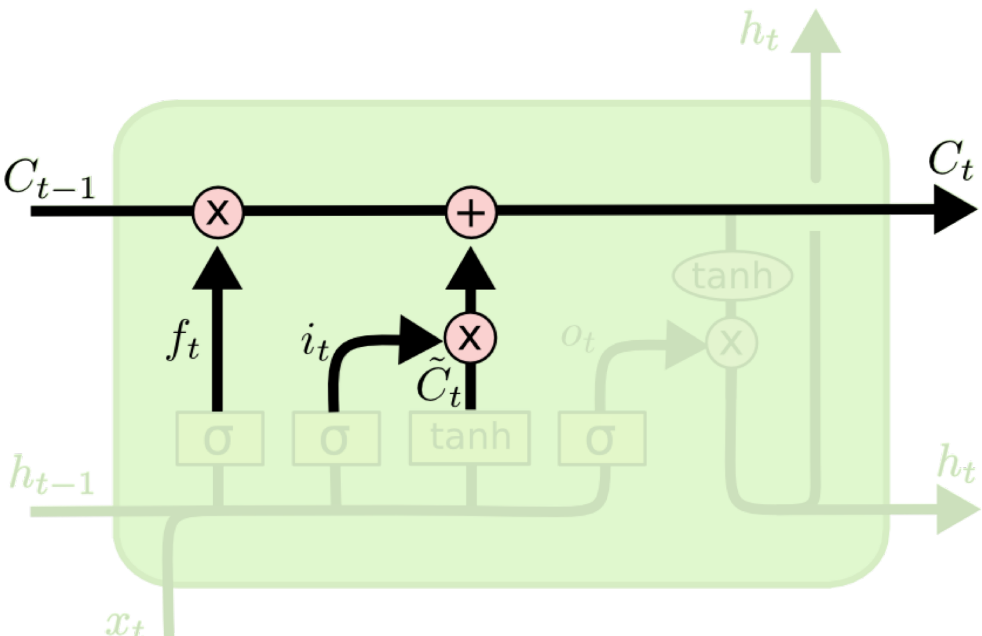
LSTM Step 2: Input Gate



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

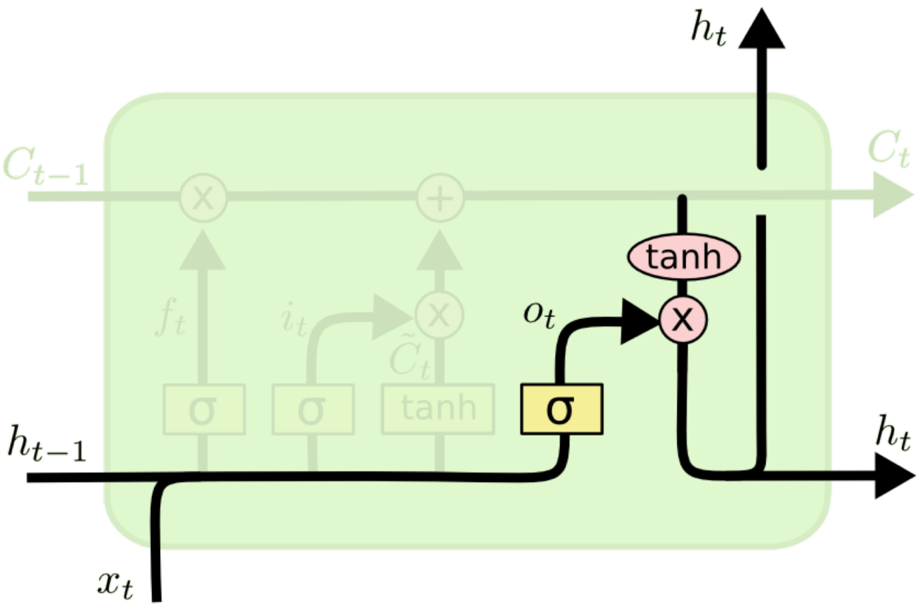
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

LSTM Step 3: Modify State



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

LSTM Step 4: Combine (new) state and input to produce output



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

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

Gated Recurrent Unit (GRU)

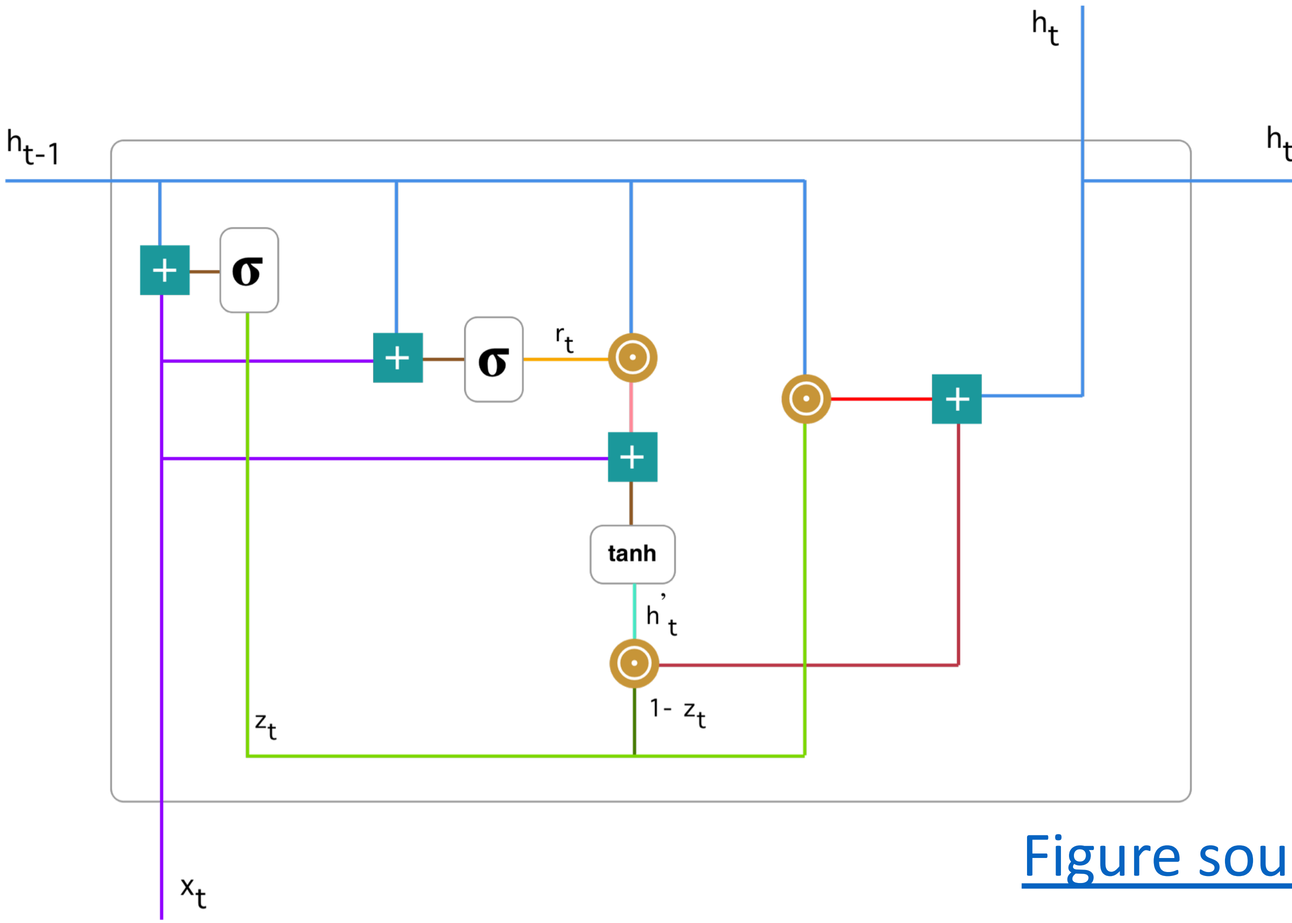
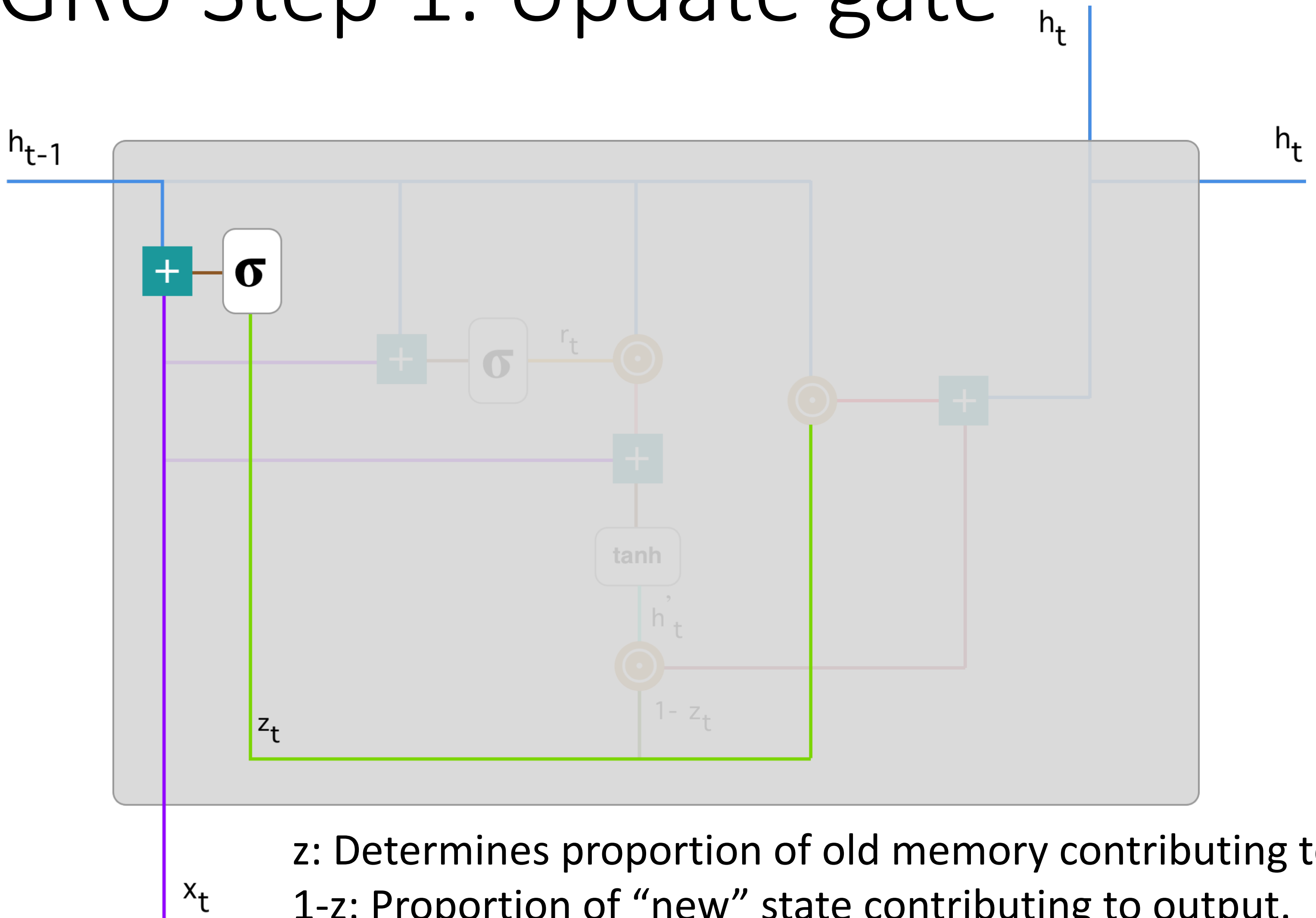


Figure source

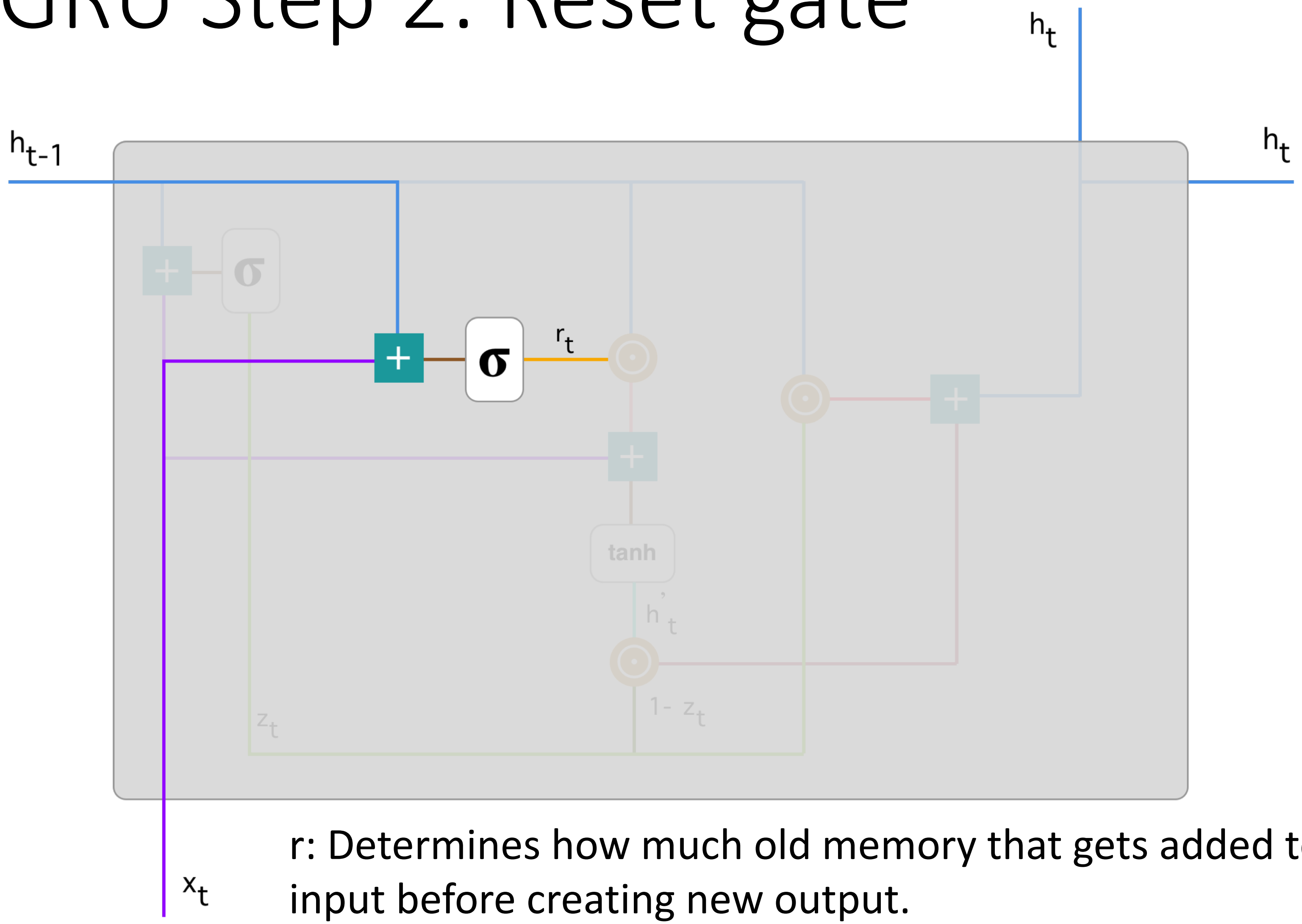
GRU Step 1: Update gate



z: Determines proportion of old memory contributing to output.
1-z: Proportion of “new” state contributing to output.

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

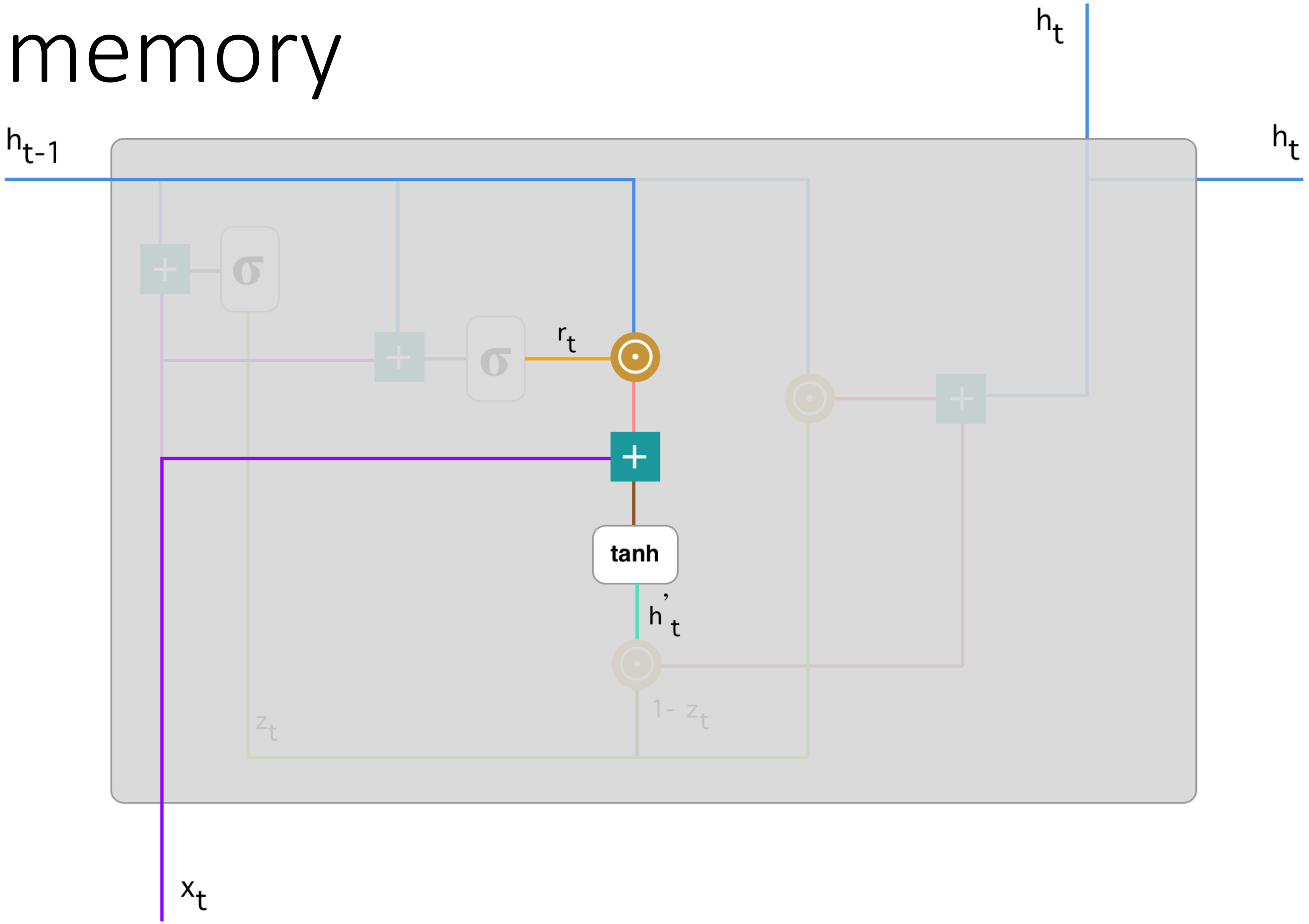
GRU Step 2: Reset gate



r: Determines how much old memory that gets added to new input before creating new output.

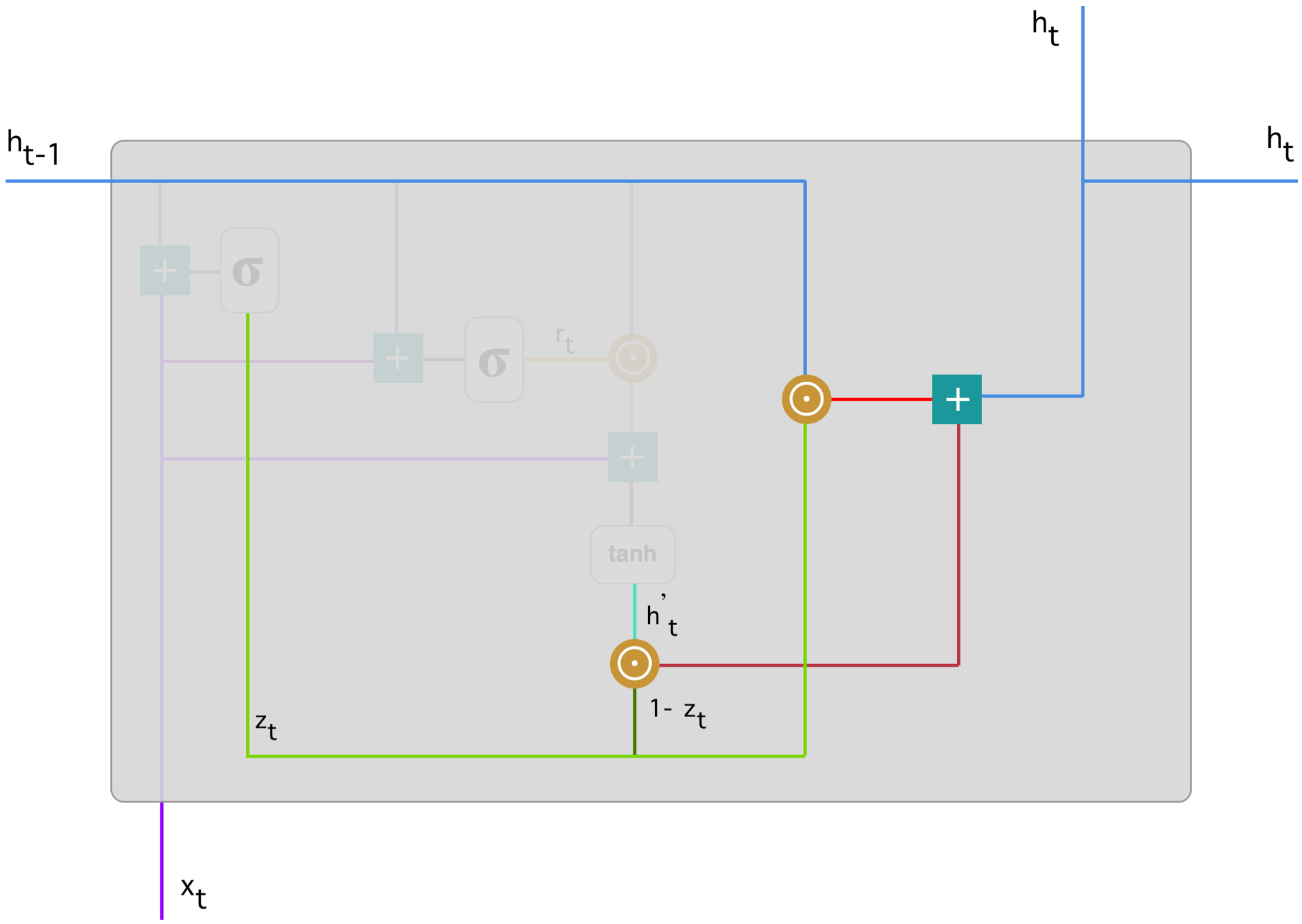
$$r_t = \sigma(W^{(r)}x_t + U^{(r)}h_{t-1})$$

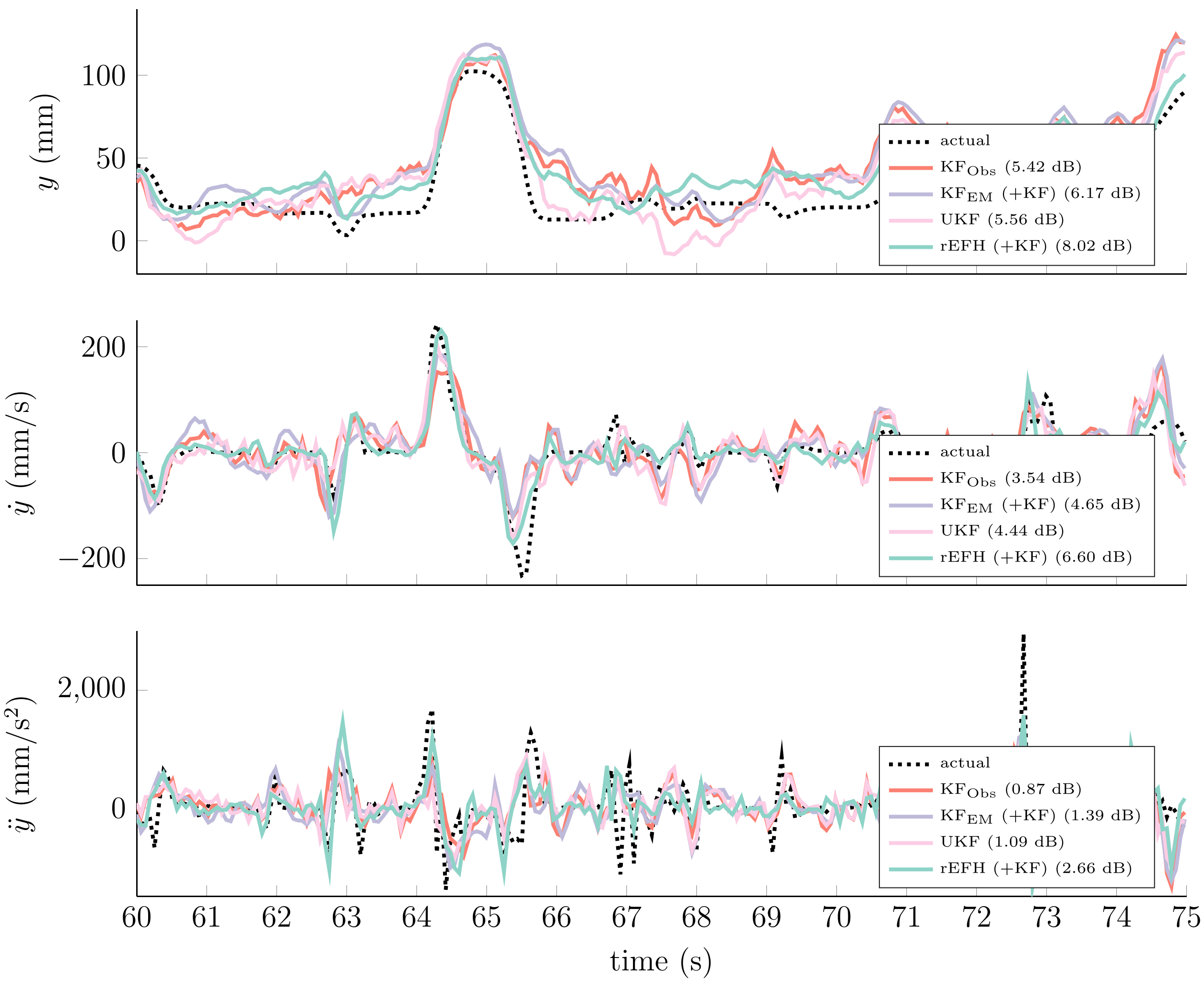
GRU Step 3: Combine input and memory



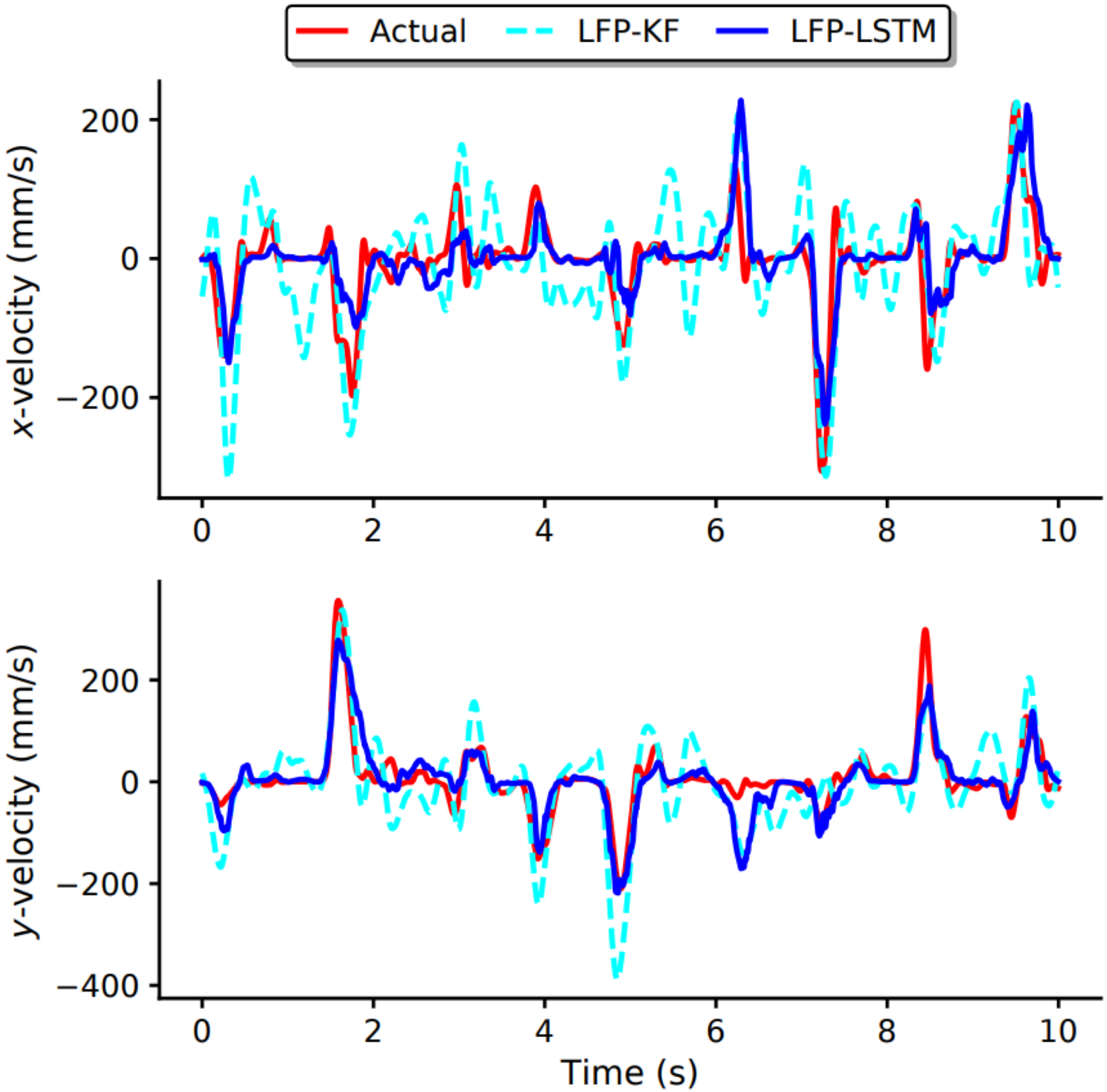
$$h_t^{'} = \tanh(Wx_t + r_t \odot Uh_{t-1})$$

GRU Step 4: Update memory



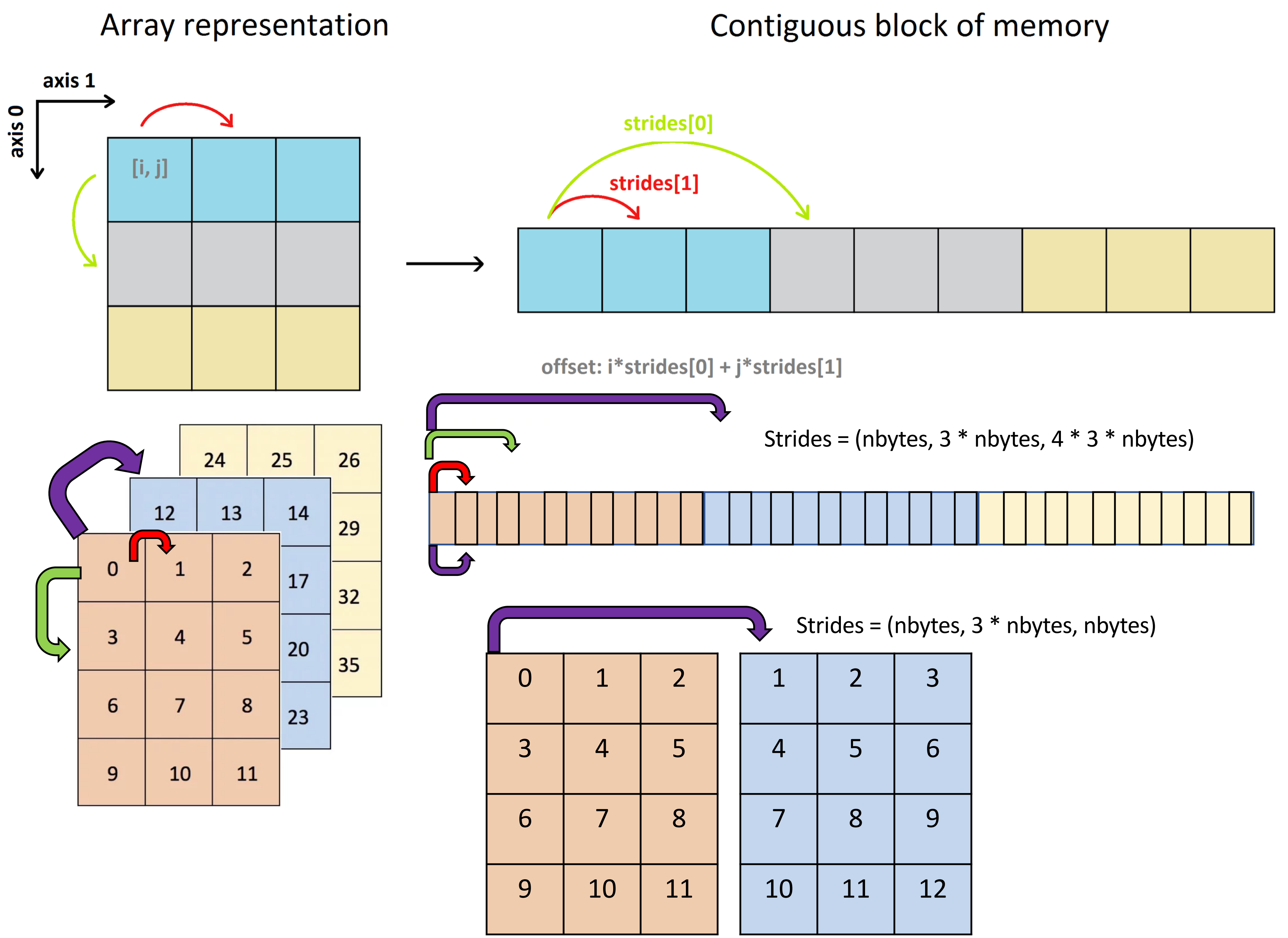


Result from Makin et al using same dataset



Result from Ahmadi et al., arXiv 2019

- 05_02_LSTM_and_GRU

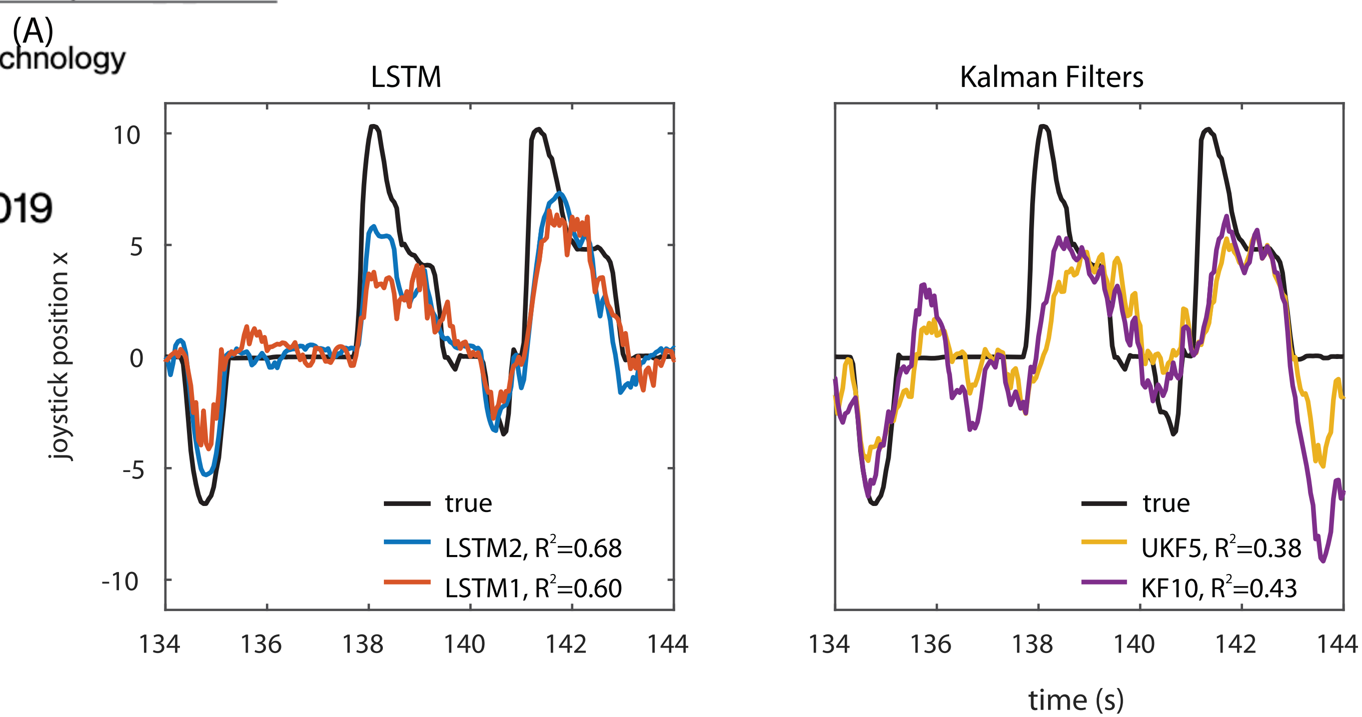

Array representation
Contiguous block of memory
axis 1
axis 0
[i, j]
strides[0]
strides[1]
offset: i*strides[0] + j*strides[1]
Strides = (nbytes, 3 * nbytes, 4 * 3 * nbytes)
Strides = (nbytes, 3 * nbytes, nbytes)
0 1 2
3 4 5
6 7 8
9 10 11
12 13 14
17
20
23
24 25 26
29
32
35
1 2 3
4 5 6
7 8 9
10 11 12

Decoding Movements from Cortical Ensemble Activity Using a Long Short-Term Memory Recurrent Network

Po-He Tseng, Núria Armengol Urpi, Mikhail Lebedev and Miguel Nicolelis

Posted Online May 21, 2019
https://doi-org.proxy.bib.uottawa.ca/10.1162/neco_a_01189

Neural Computation
Volume 31 | Issue 6 | June 2019
p.1085-1113



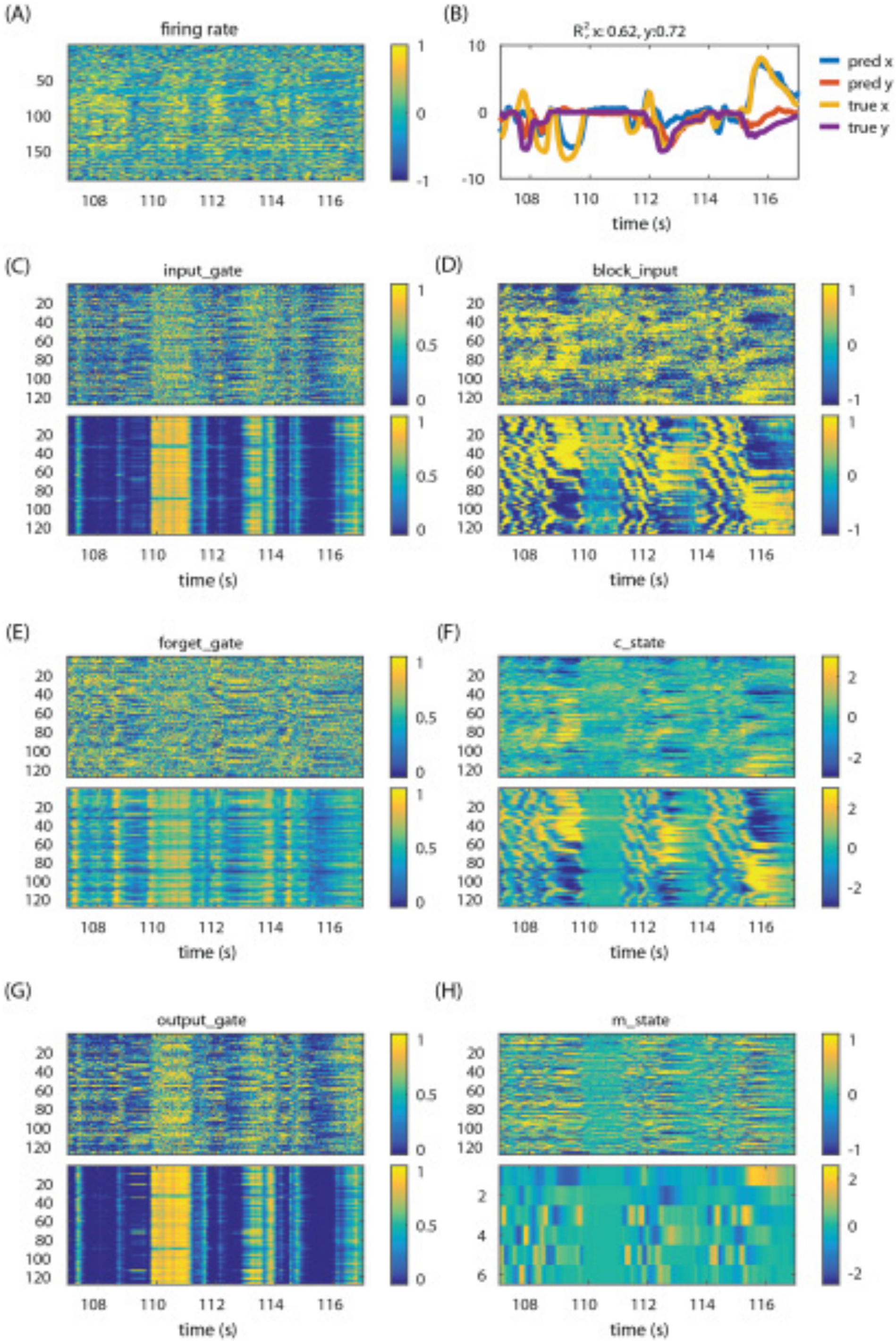
Model internal values in LSTM layers 1 and 2

Layers presented in pairs, with layer 1 on top.

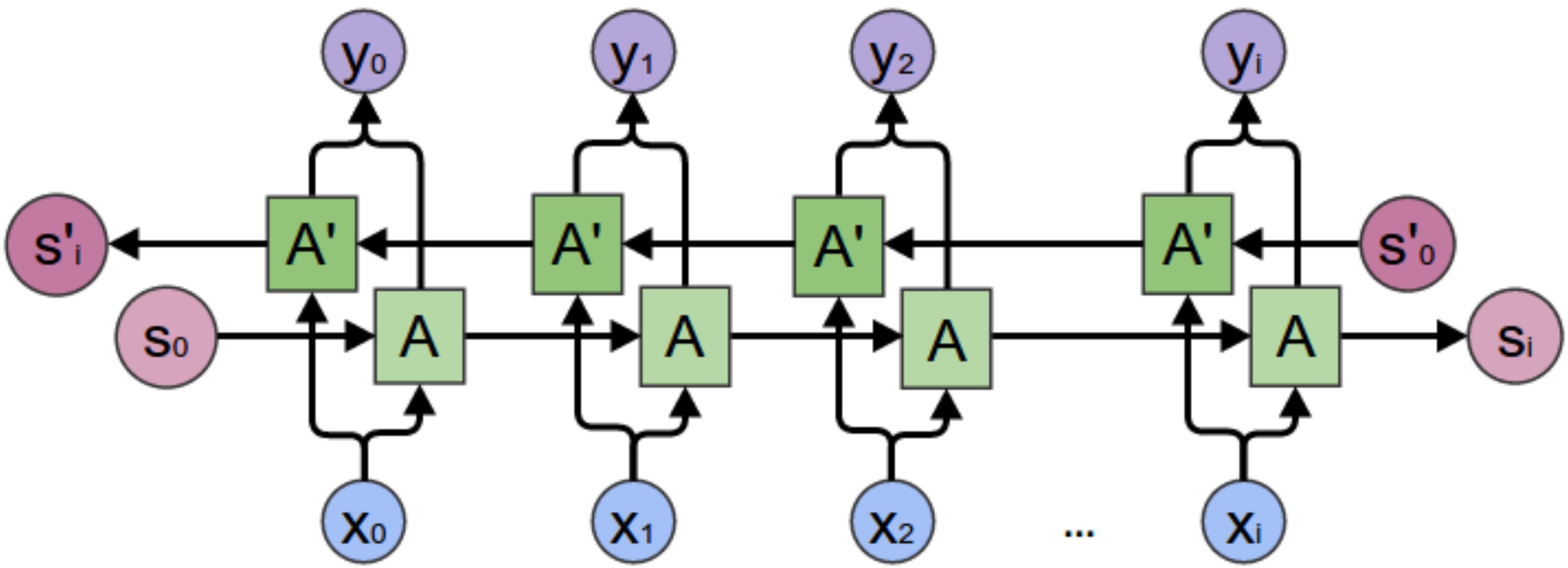
Layer 2 activations had higher contrast than layer 1, primarily because layer 2 gates were more fully closed.

(Not shown) layer 2 units were directionally tuned.

Does this tell us anything about how the brain works?

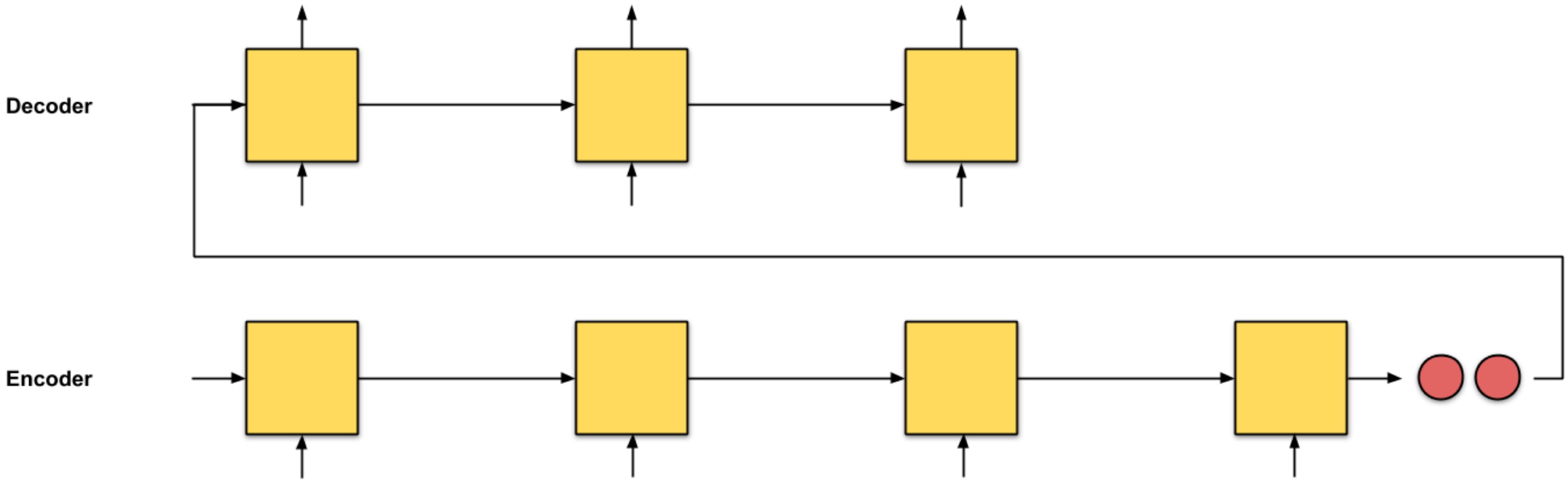


Bidirectional RNN



Loop forward over the sequence.
Loop backward over the sequence.
Combine output(s).

seq2seq (Sutskever et al., 2014)



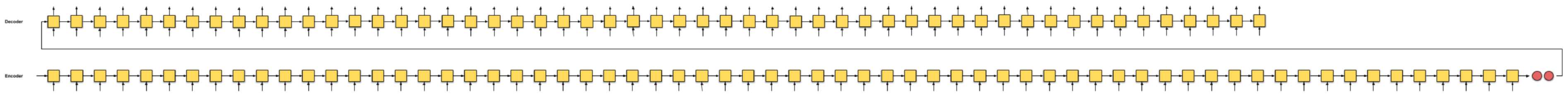
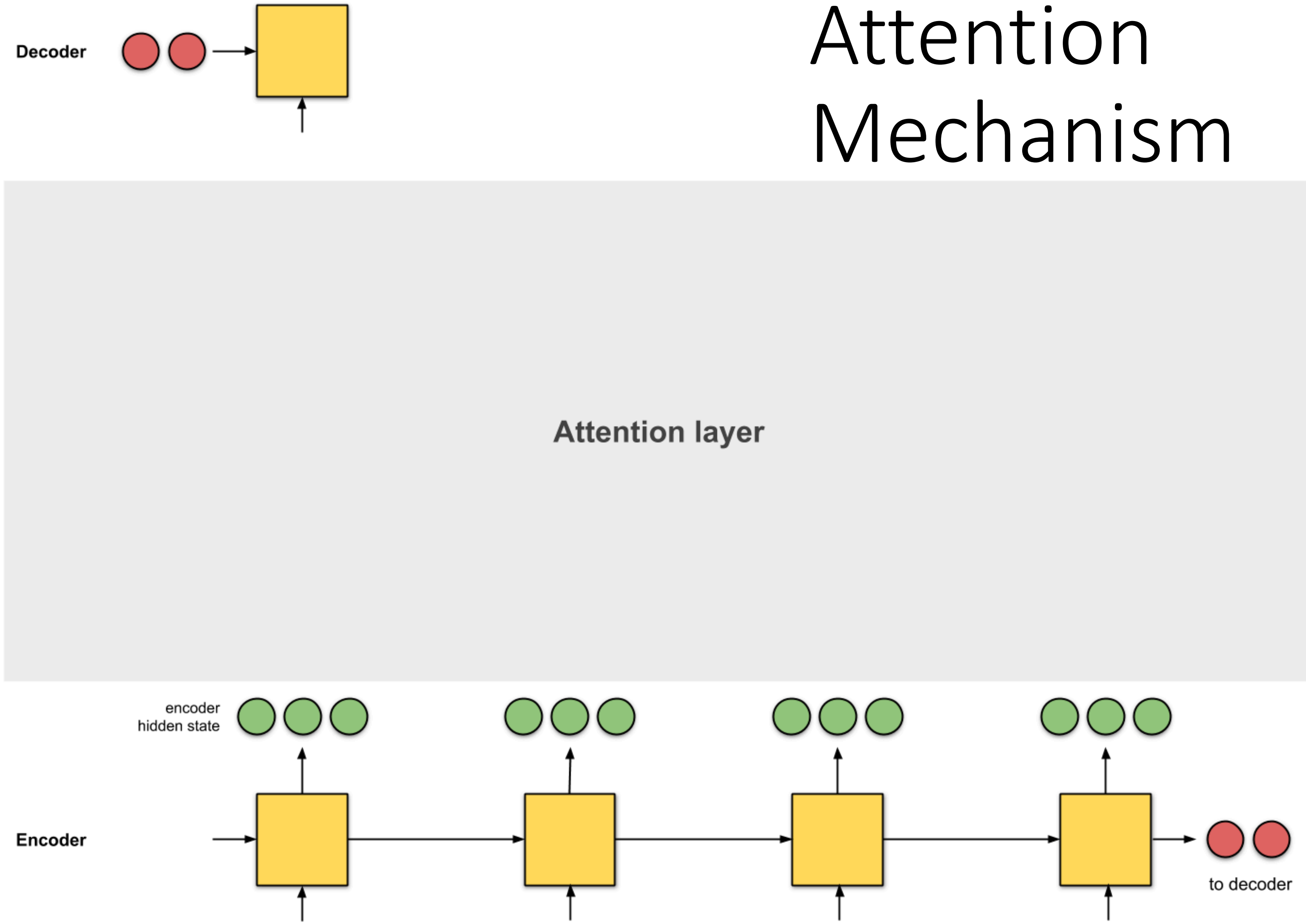


Diagram source

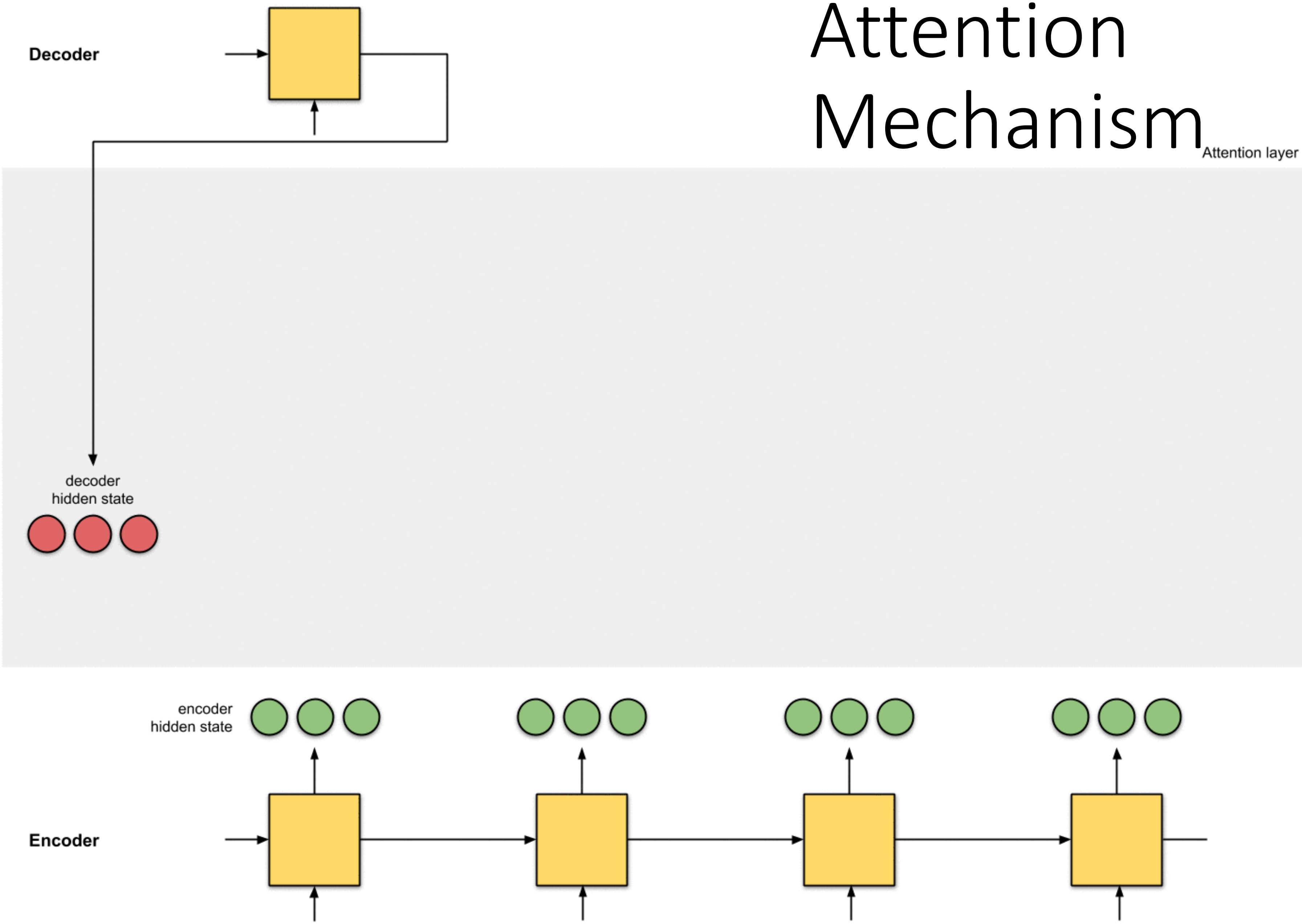
Attention Mechanism



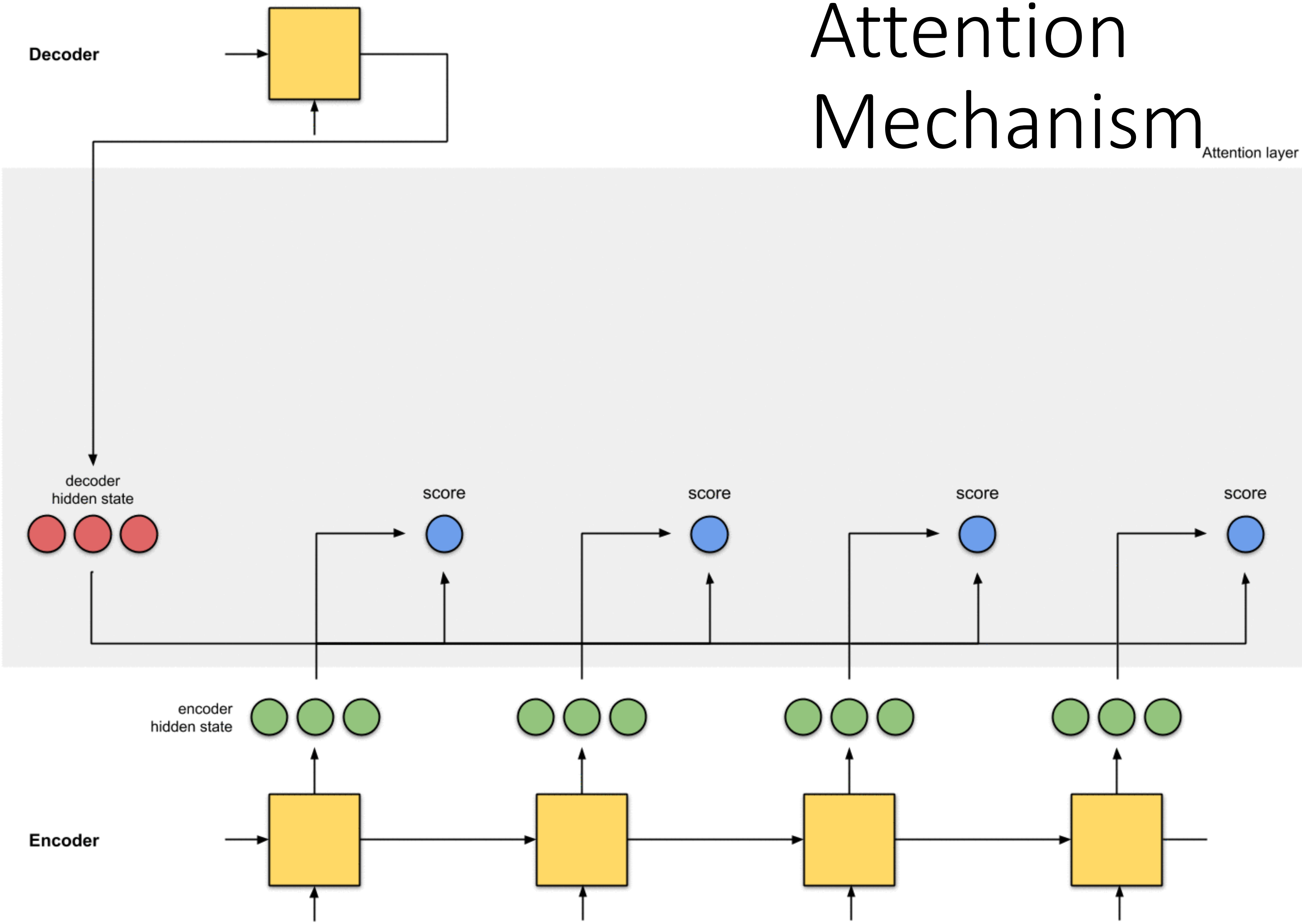
Attention Mechanism

Encoder

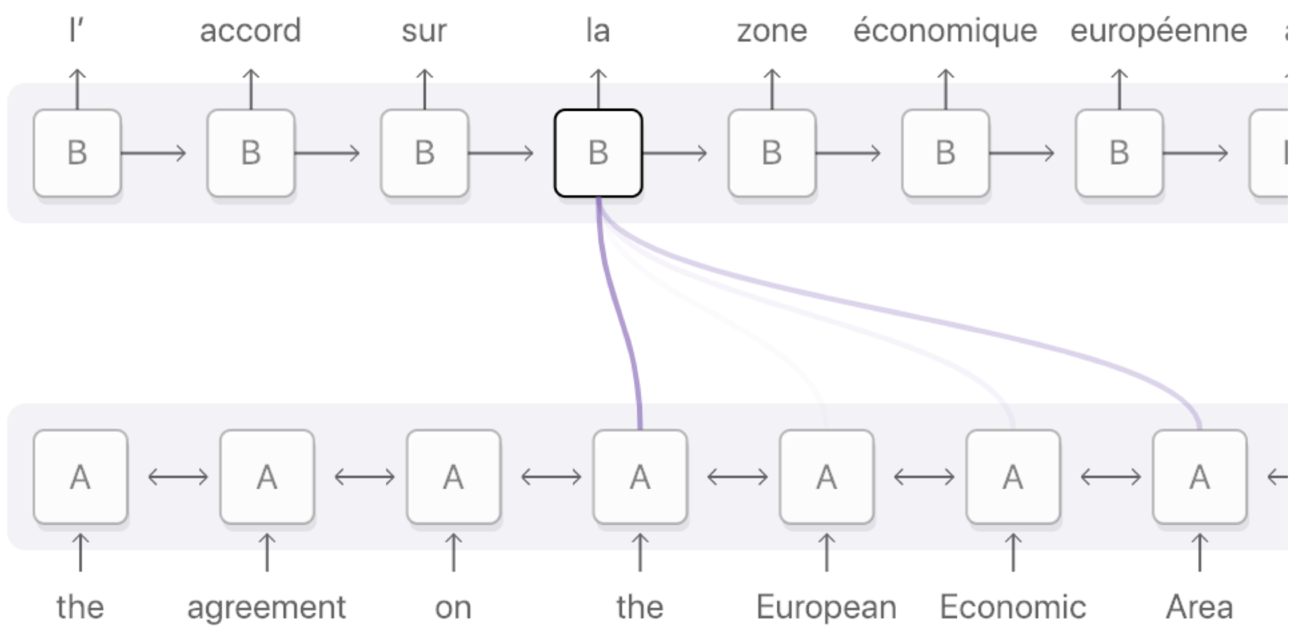
Attention Mechanism



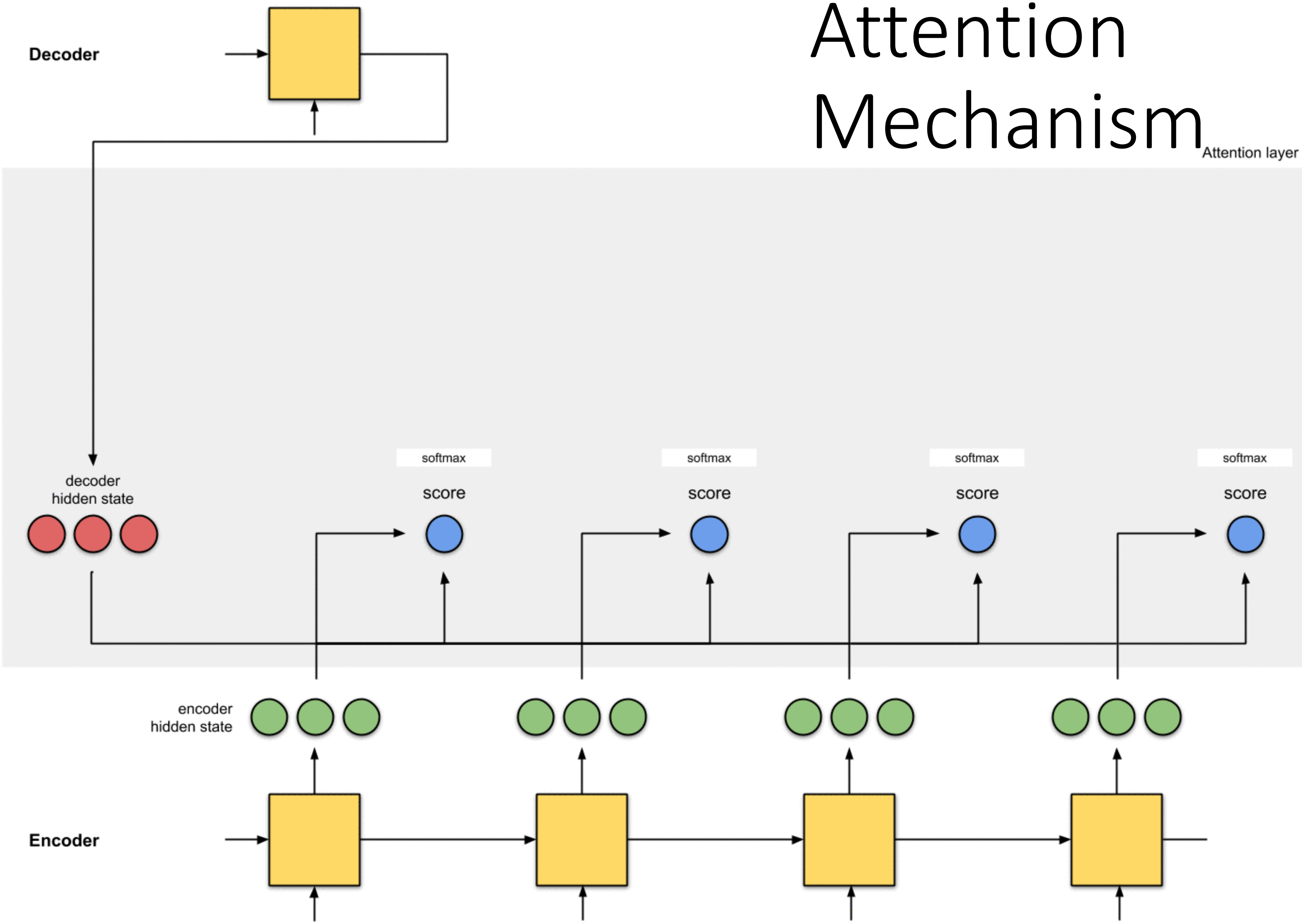
Attention Mechanism



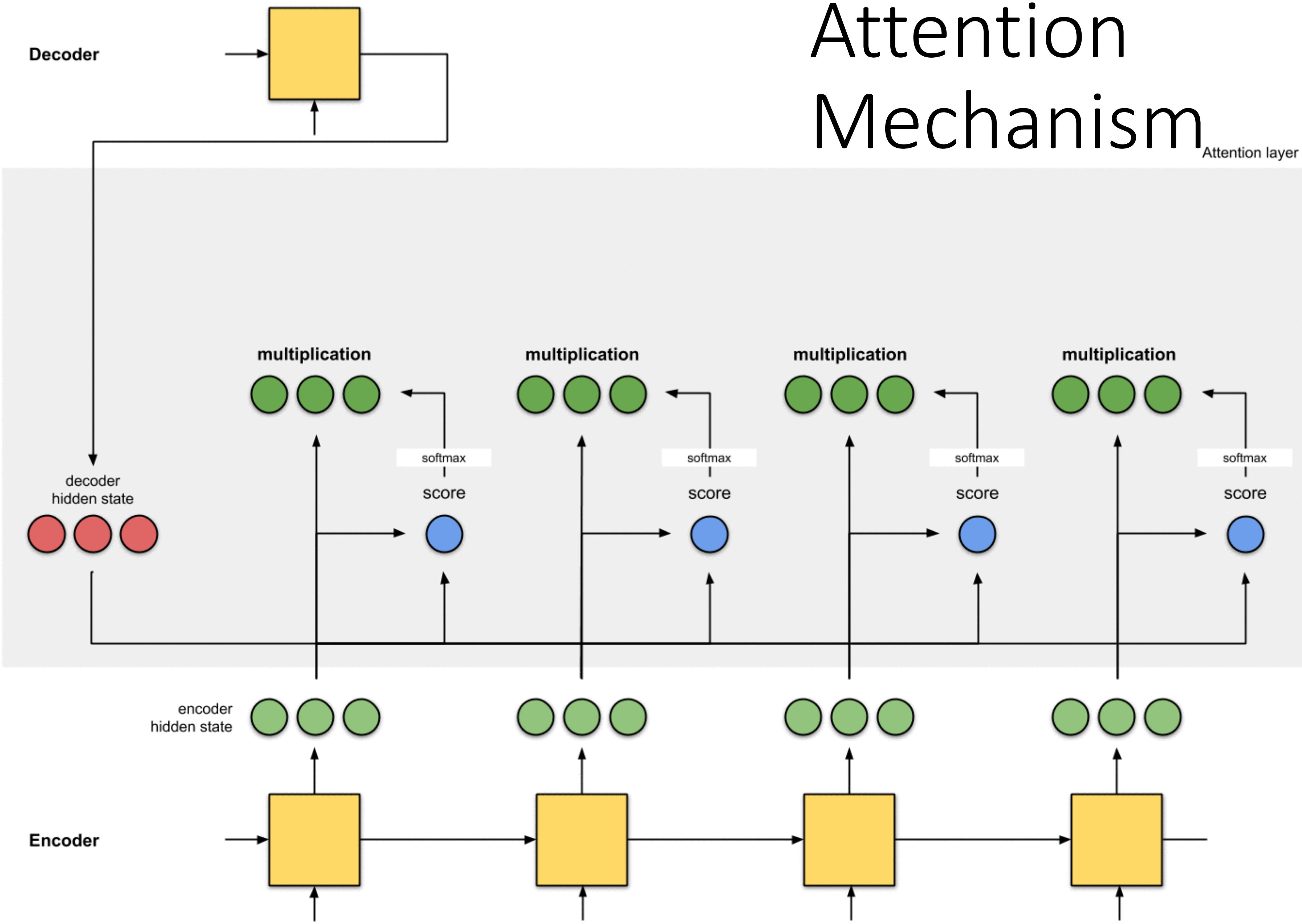
Attention Mechanism



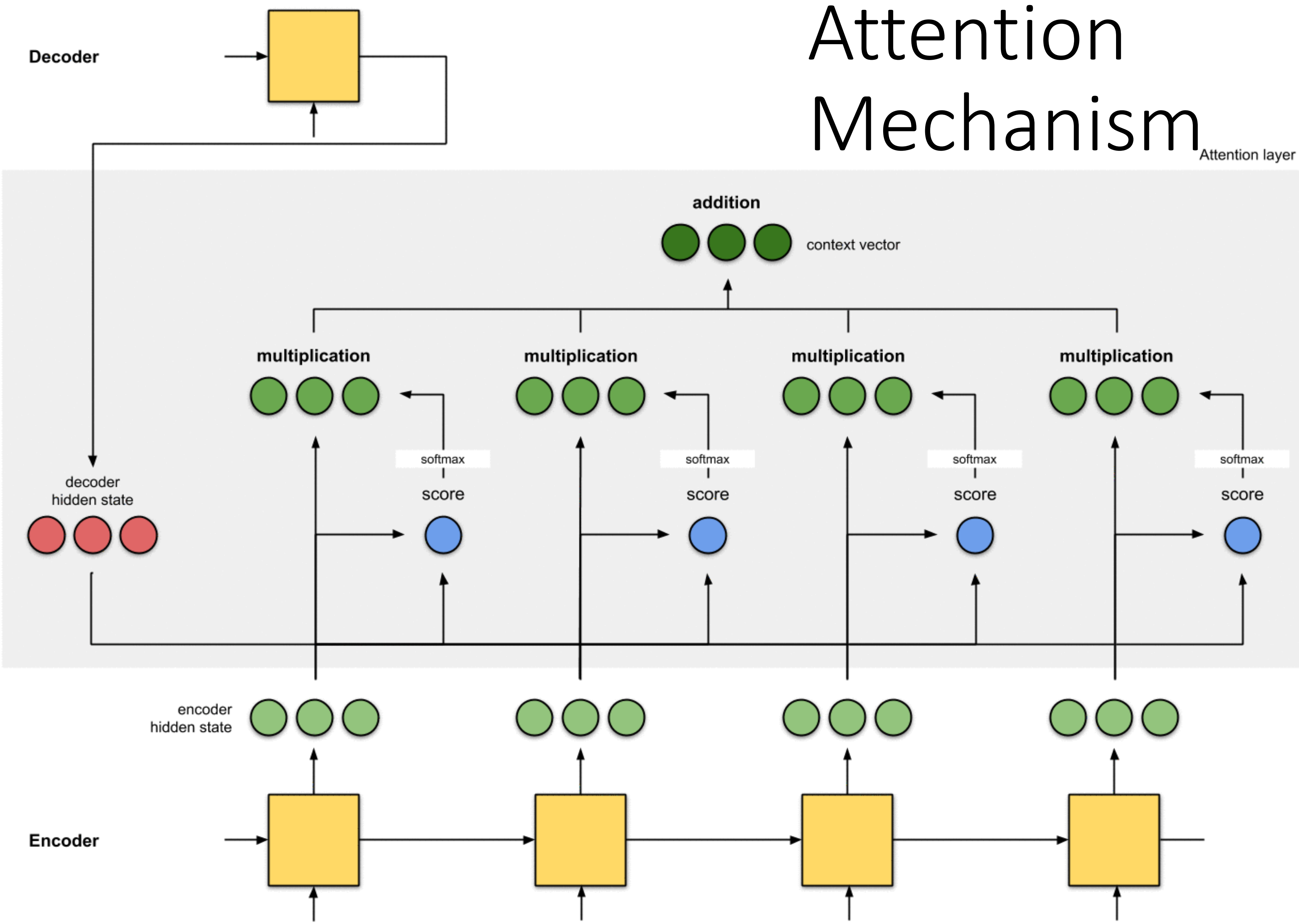
Attention Mechanism

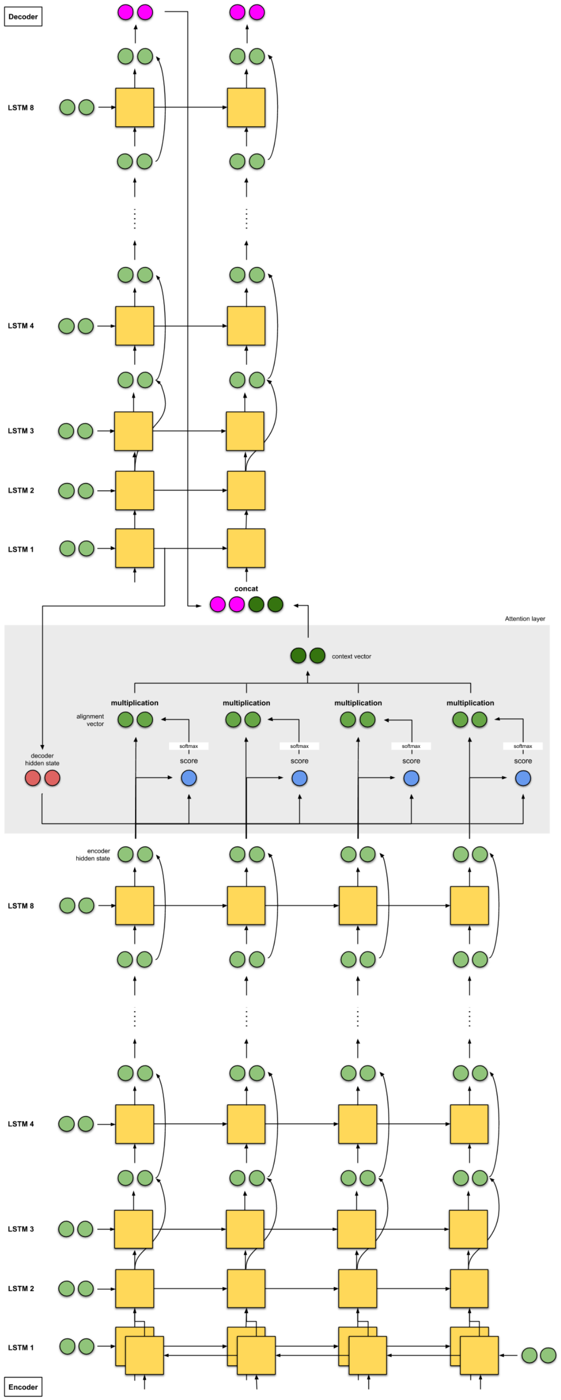


Attention Mechanism

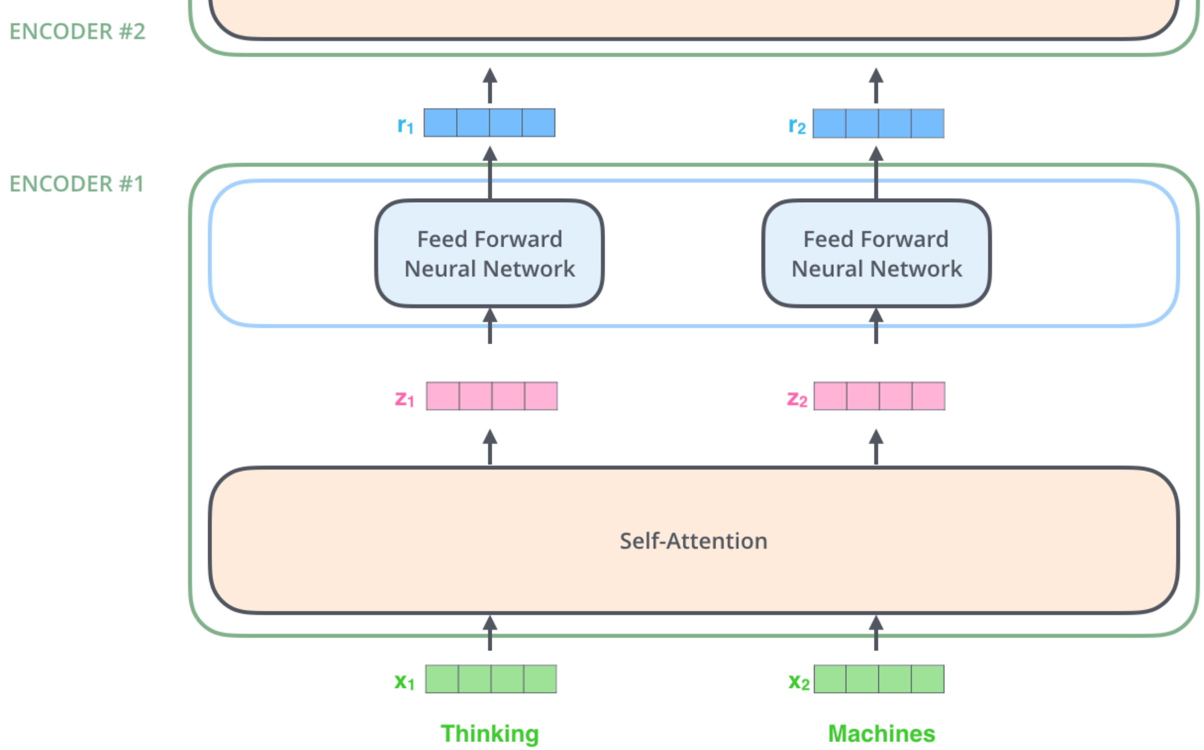


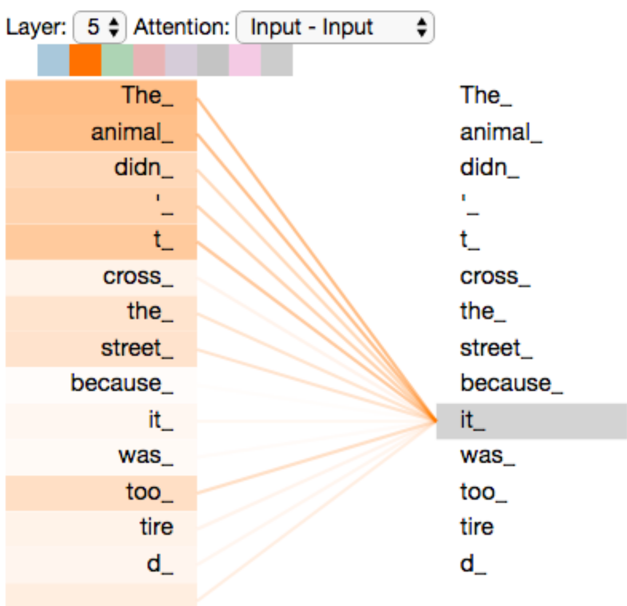
Attention Mechanism





Google's Neural Machine Translation





From [The Illustrated Transformer]