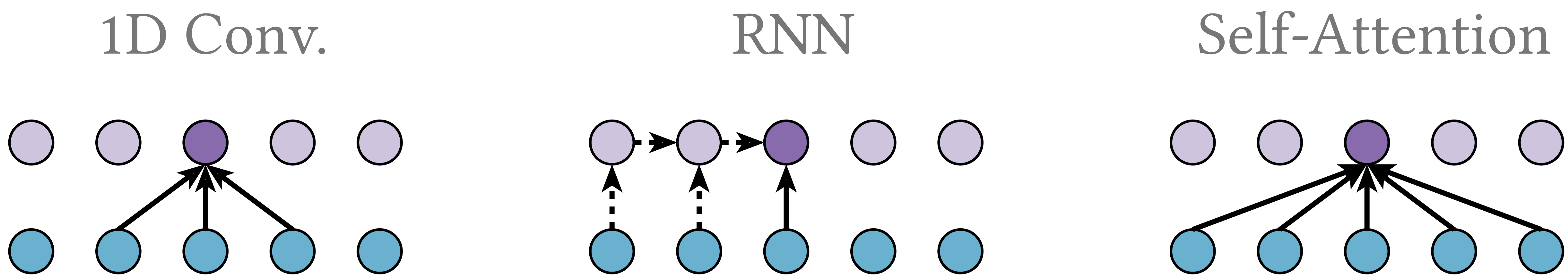
Deep Learning
5. Time series & sequences

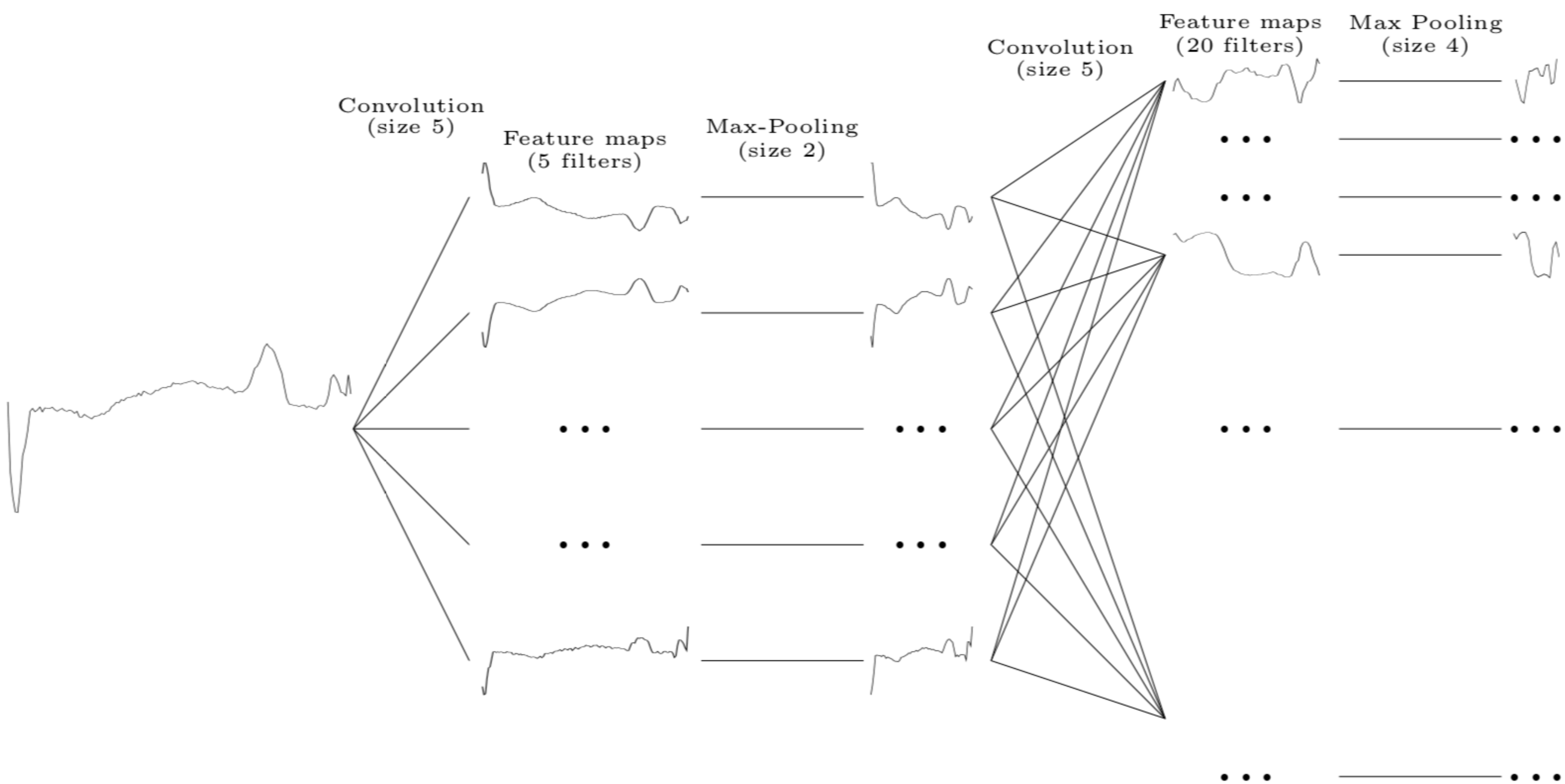
A course @EDHEC
by Romain Tavenard (Prof. @Univ. Rennes 2)

Slides on attention are widely inspired from François Fleuret's excellent Deep Learning cours at UNIGE: https://fleuret.org/dlc/

Neural network architectures and inductive biases



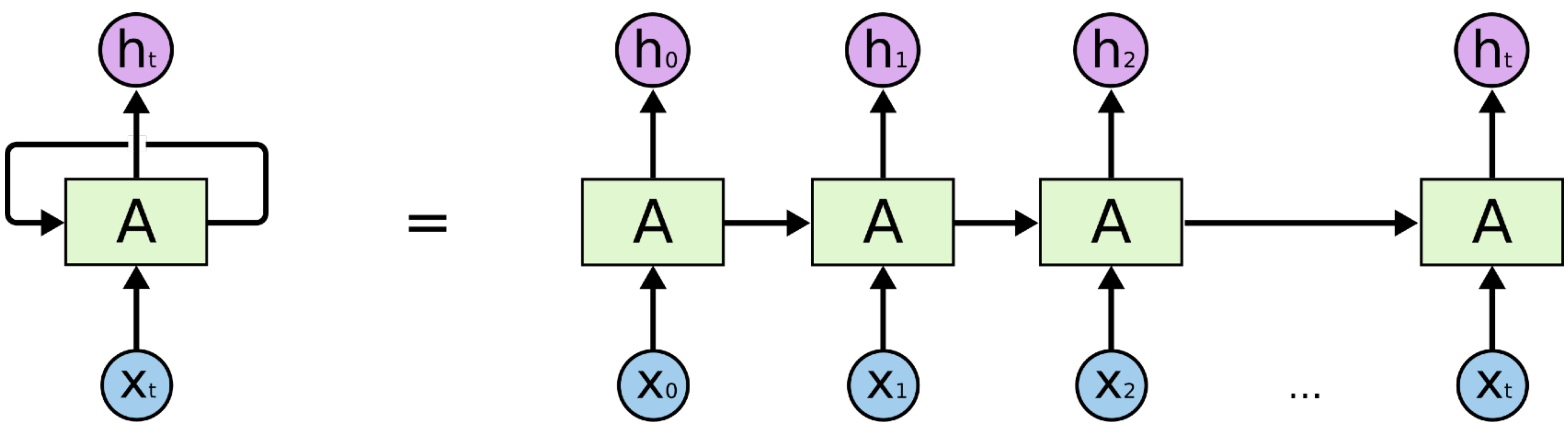
Convolutional neural nets for time series



Source: [Le Guennec *et al.*, 2014]

Recurrent neural nets

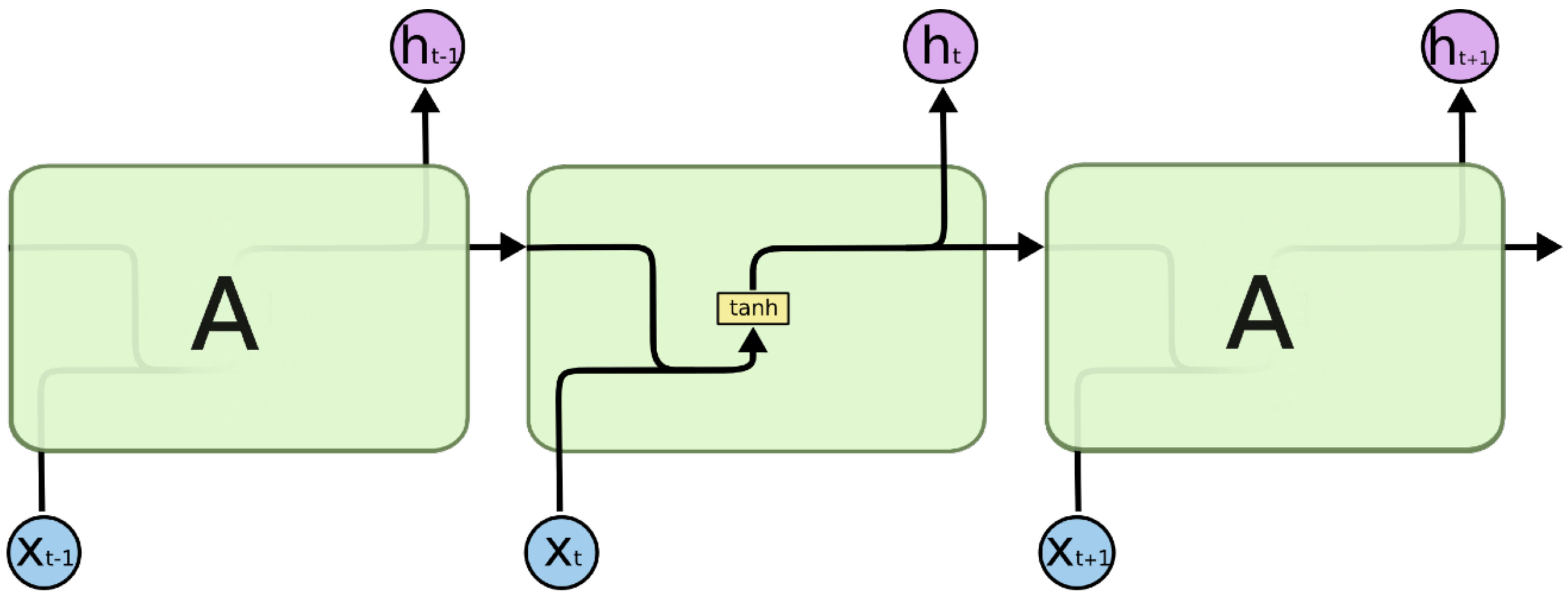
- Very flexible model (any length, let the model learn its memory needs, …)



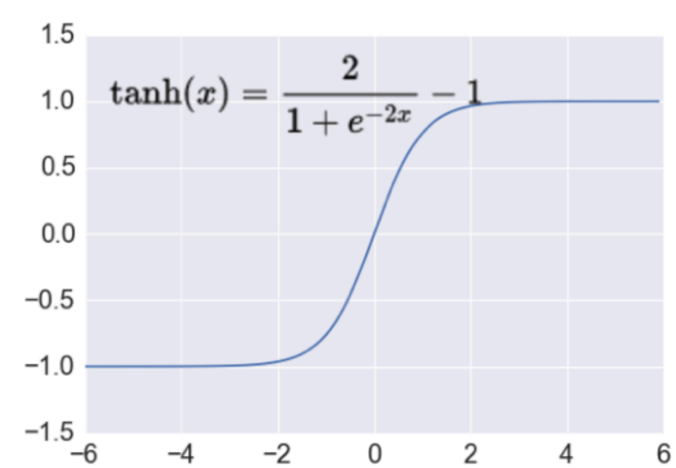
Source: Christopher Olah's blog

Recurrent neural nets

- "Vanilla" RNN in more details



$$h_t = \tanh(W_h h_{t-1} + W_x x_t + b)$$

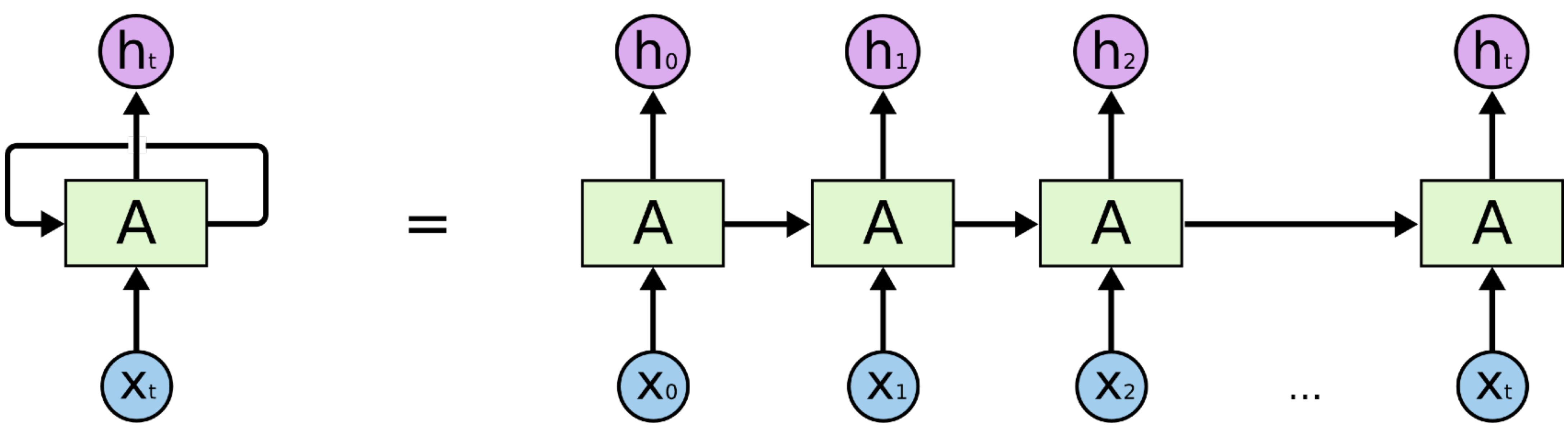


tanh

Illustration: RNN cell, source: Christopher Olah's blog

Recurrent neural nets

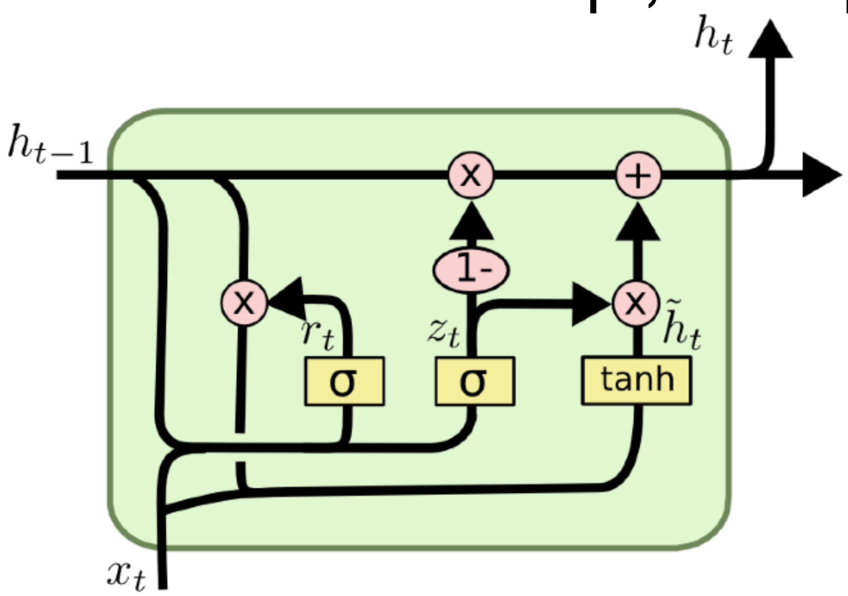
- Very flexible model (any length, let the model learn its memory needs, …)
- Difficult to learn in practice
    - Slow (lack of parallelism)
    - Vanishing gradients (hard to learn long-term dependencies)



Source: Christopher Olah's blog

Recurrent neural nets

- Gated Recurrent Unit (GRU)
- Principle
  - At each time step, keep only part of the information



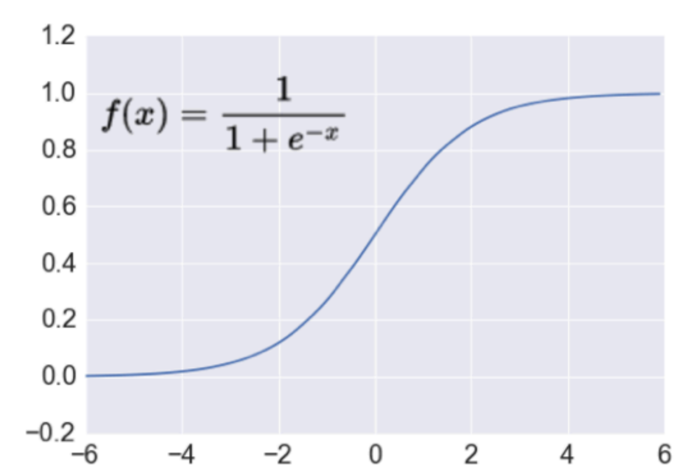
$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Illustration: GRU cell, source: Christopher Olah's blog



sigmoid σ

Recurrent neural nets

- Long Short-Term Memory (LSTM)
- Principle: similar to GRU

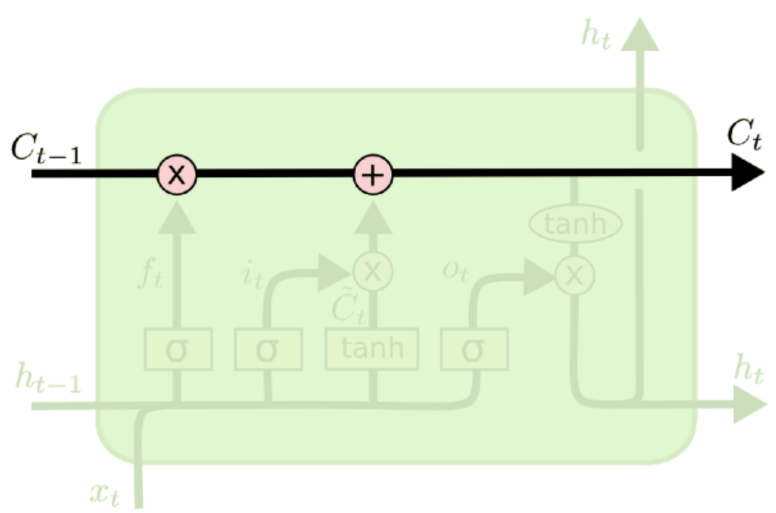
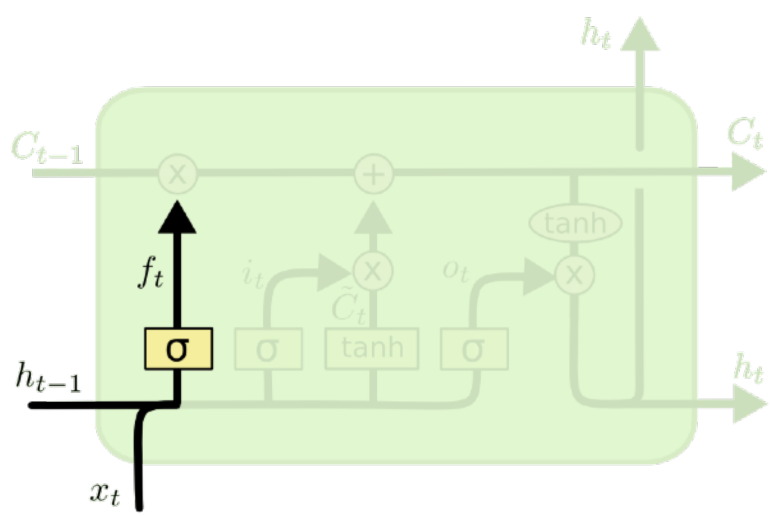


Illustration: LSTM cell, source: Christopher Olah's blog

Recurrent neural nets

- Long Short-Term Memory (LSTM)
- Principle: similar to GRU

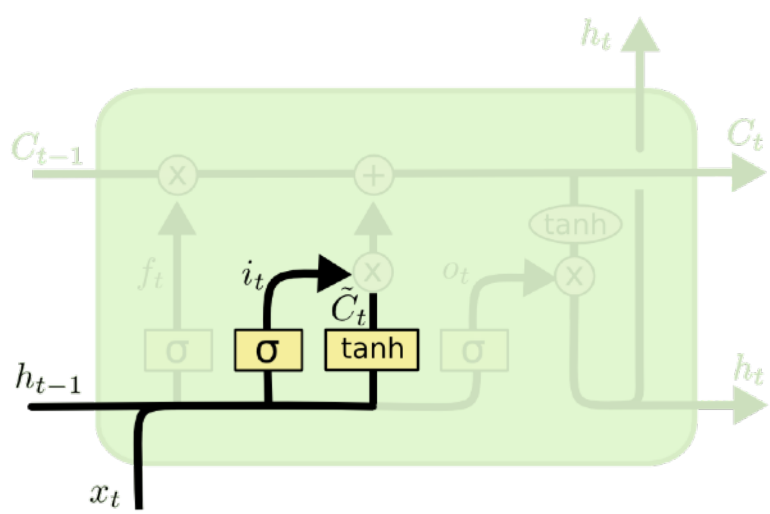


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$

Illustration: LSTM cell, source: Christopher Olah's blog

Recurrent neural nets

- Long Short-Term Memory (LSTM)
- Principle: similar to GRU

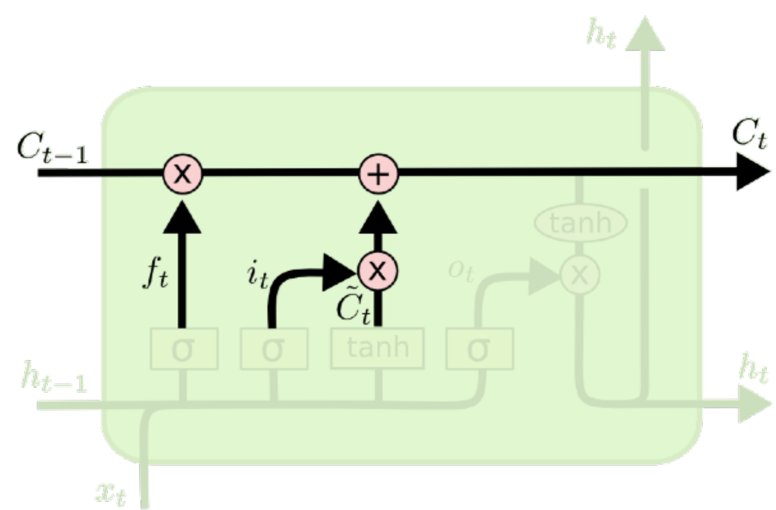


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

Illustration: LSTM cell, source: Christopher Olah's blog

Recurrent neural nets

- Long Short-Term Memory (LSTM)
- Principle: similar to GRU

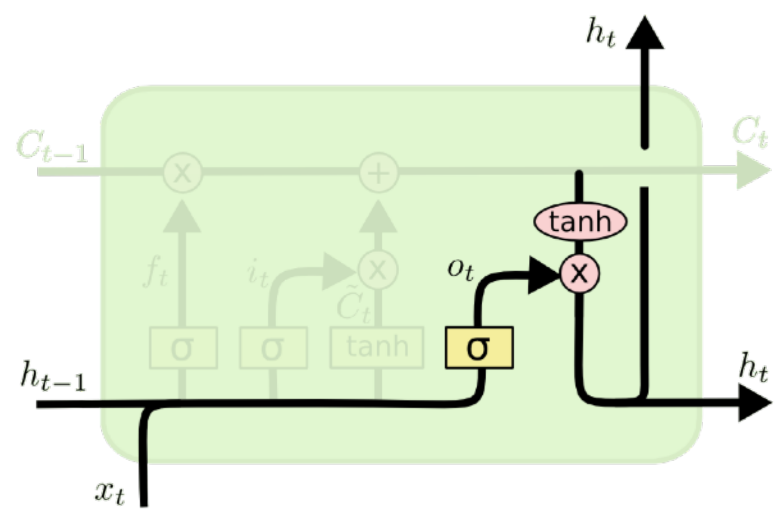


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Illustration: LSTM cell, source: Christopher Olah's blog

Recurrent neural nets

- Long Short-Term Memory (LSTM)
- Principle: similar to GRU



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

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

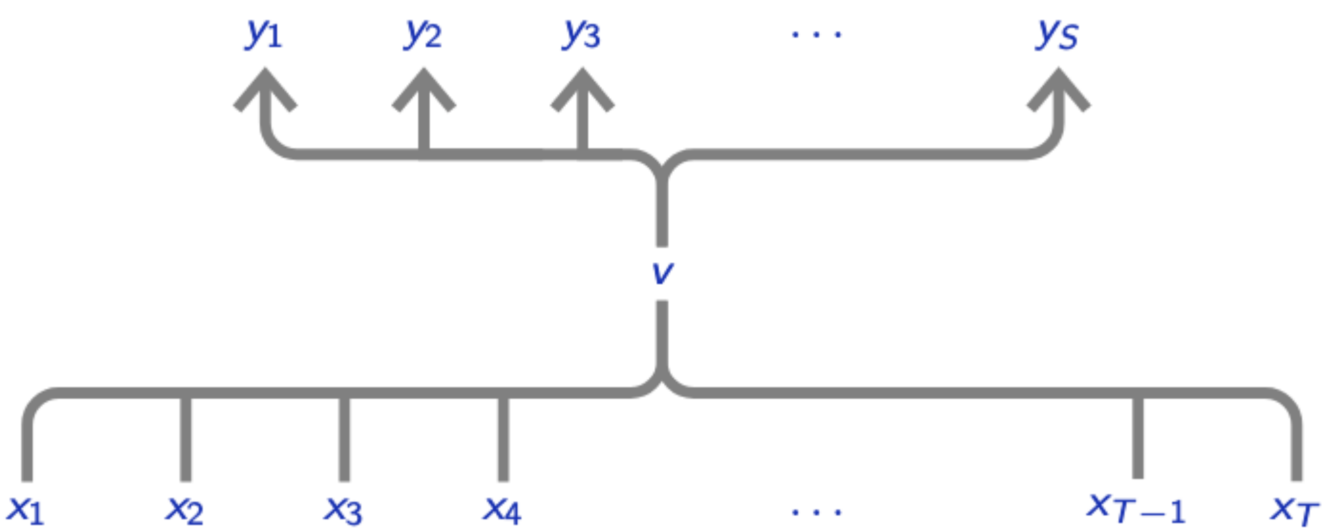
Illustration: LSTM cell, source: Christopher Olah's blog

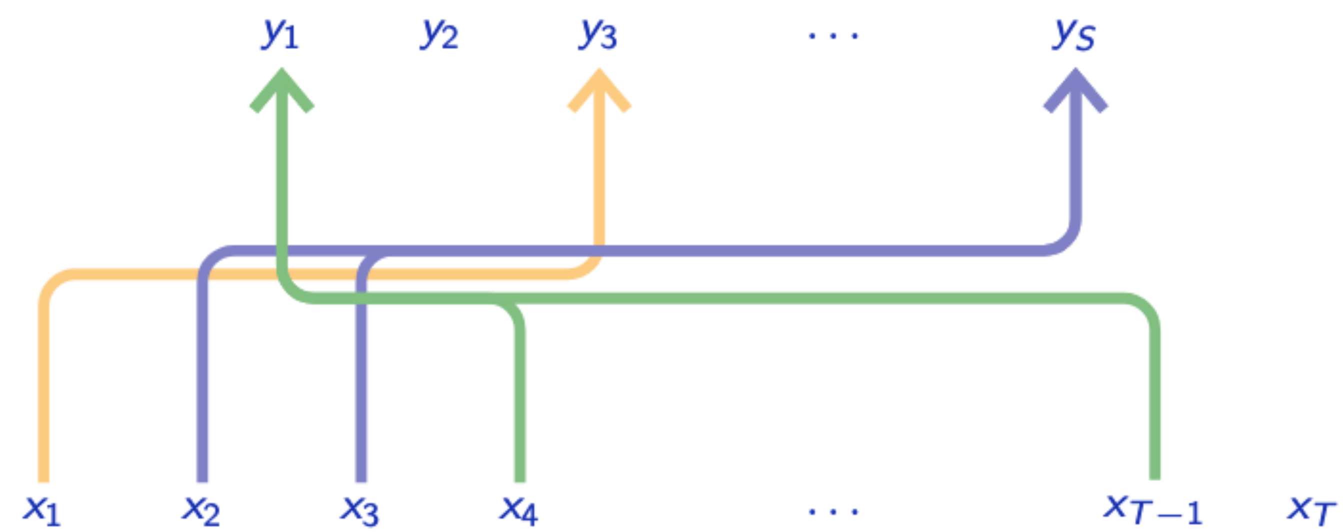
Attention-based models
Motivating example

- Consider the following translation task:
  - From English: "**An apple** that had been on the tree in the garden for weeks had finally been **picked up**."
  - To French: "**Une pomme** qui était sur l’arbre du jardin depuis des semaines avait finalement été **ramassée**."

- Both recurrent and convolutional architectures fail at modelling long-range dependencies (for different reasons)
  - Attention aims at tackling this limitation

Attention for Sequence-to-Sequence (seq2seq) tasks





<u>Recurrent models</u>

- Compute a bottleneck representation
- Generate output sequence from this bottleneck

<u>Attention-based models</u>

- For each token in the output sequence, aggregate input features
- Aggregation is importance-based
- Importance depends on the features, not their localization

Assessing importance: Queries, Keys & Values

- The Query / Key / Value metaphor: Python dictionaries

```
d = {"a": 12, "b": 7}
print(d["a"])
```

- Queries, Keys & Values: the continuous case
  - Look for keys that are **similar** to the query (not equal)
  - Retrieved value is a weighted average of values associated to keys that are close to the query
    - Could be seen as weighted k-nearest neighbors

Visual explanation of the Attention mechanism

$$A_{i,j} = \mathrm{softmax}\left(Q_i \cdot K_j\right)$$

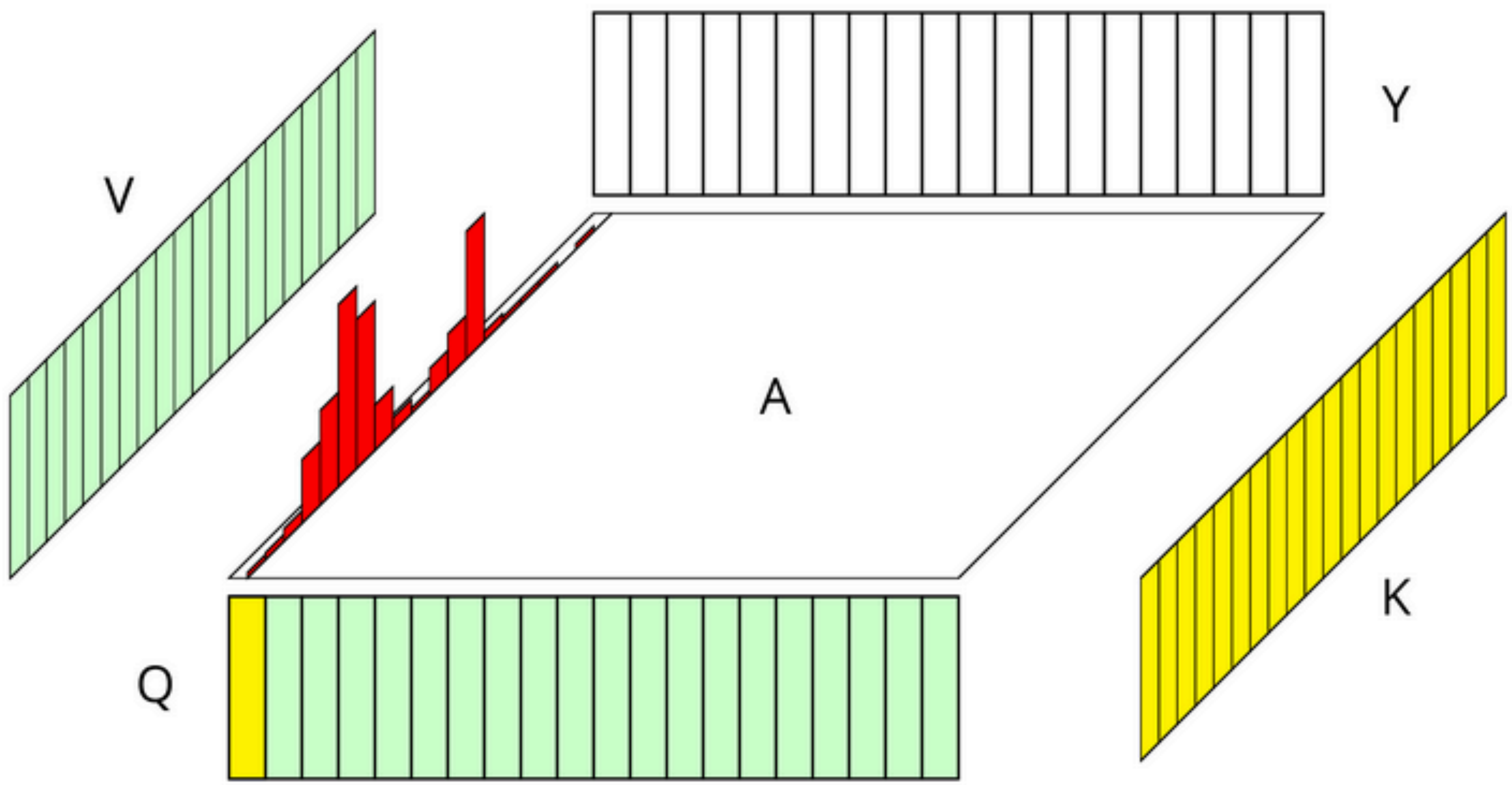


Image credit: François Fleuret

Visual explanation of the Attention mechanism

$$Y_i = \sum_j A_{i,j} V_j$$

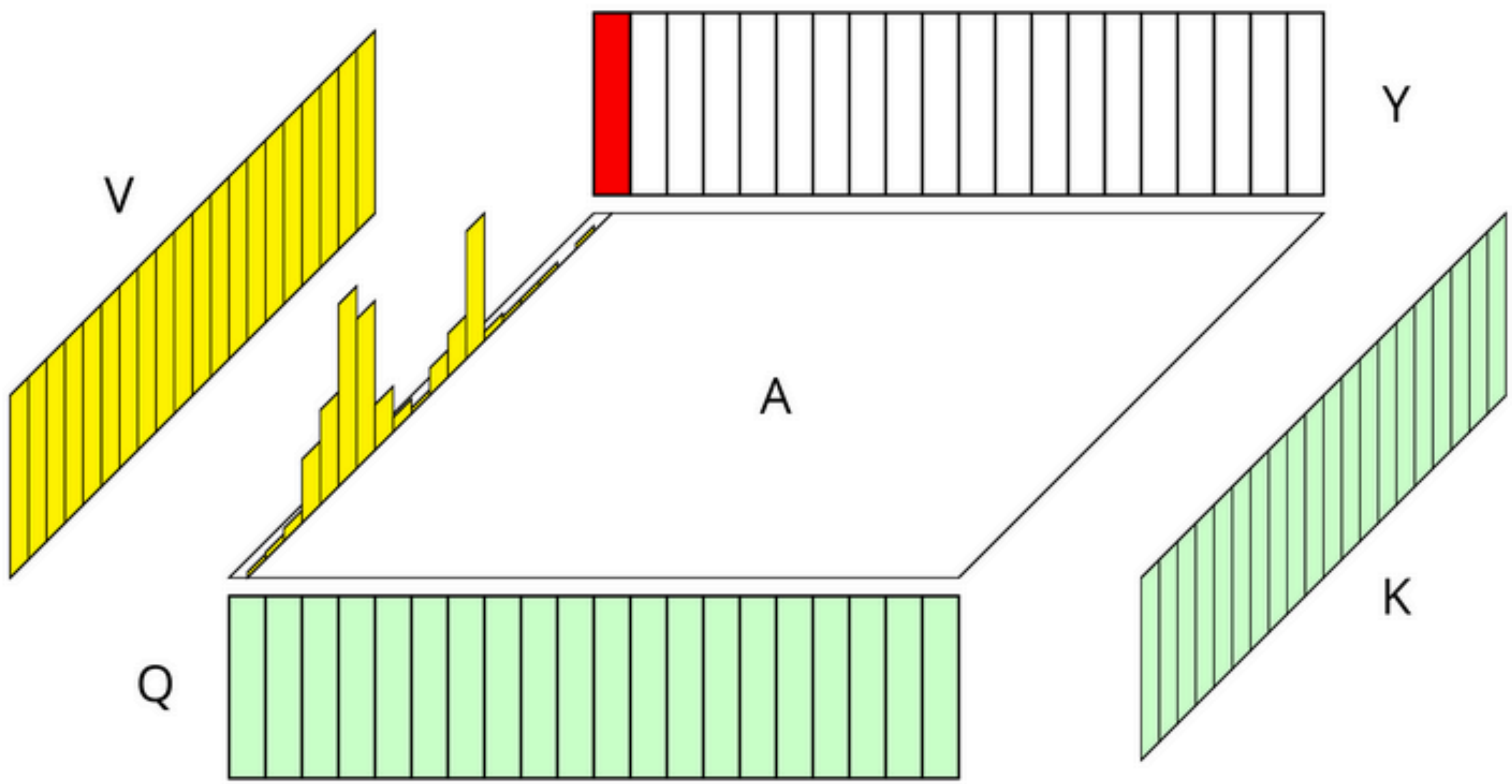


Image credit: François Fleuret

Visual explanation of the Attention mechanism

$$A_{i,j} = \mathrm{softmax}\left(Q_i \cdot K_j\right)$$

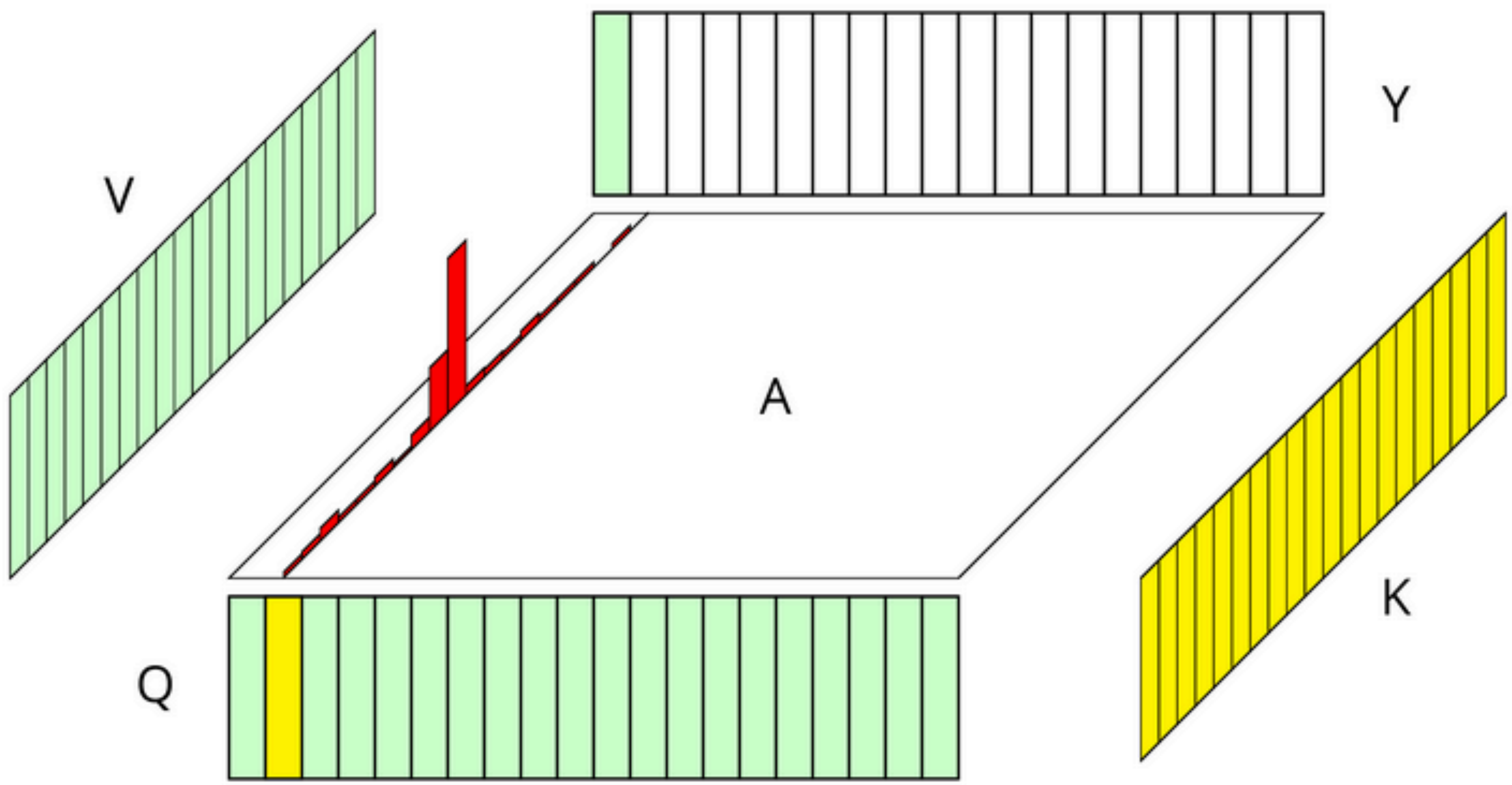


Image credit: François Fleuret

Visual explanation of the Attention mechanism

$$Y_i = \sum_j A_{i,j} V_j$$

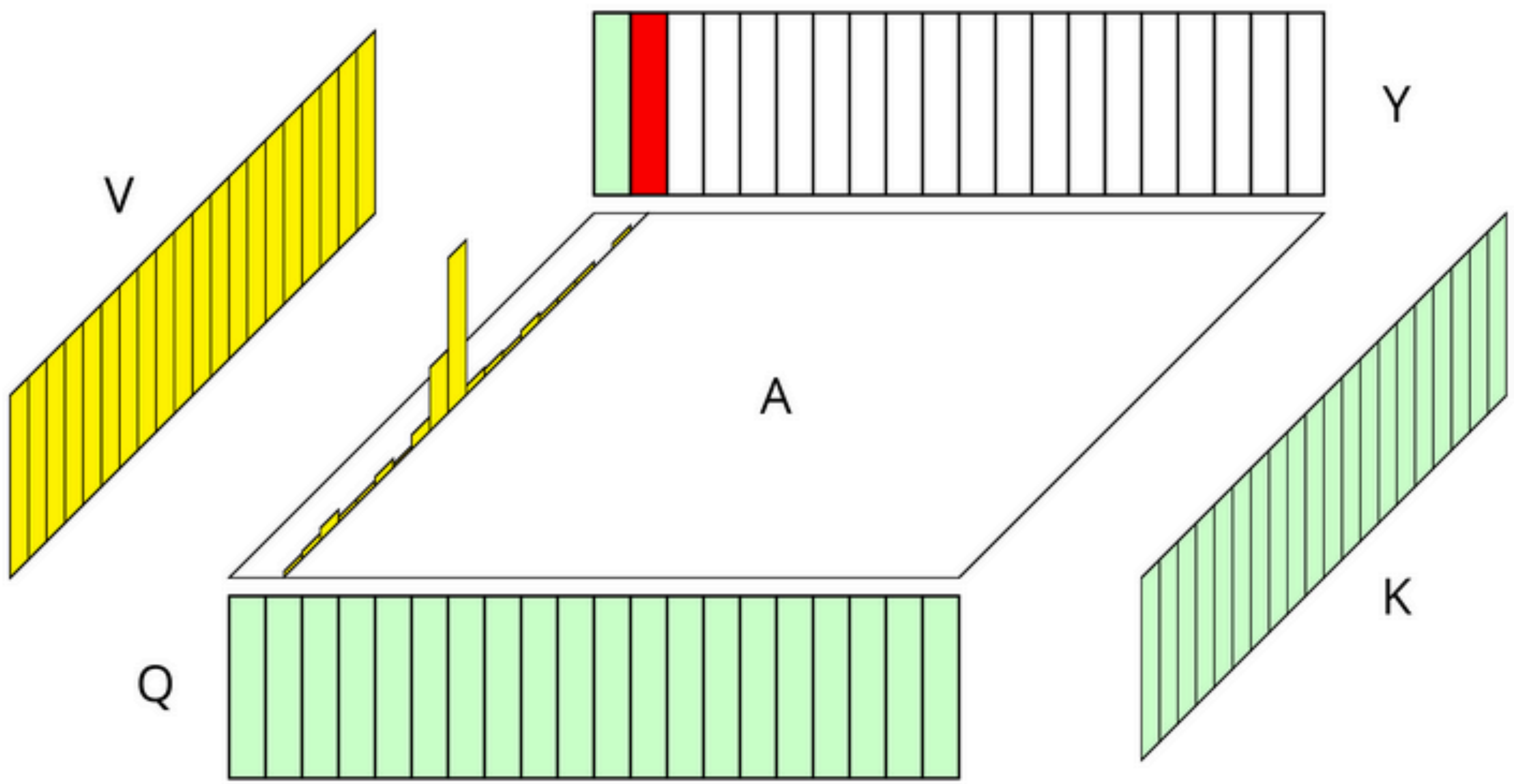


Image credit: François Fleuret

Standard attention layers

- <u>Standard Attention</u> layers
  - take as inputs 2 sequences X and X'
  - output a sequence Y

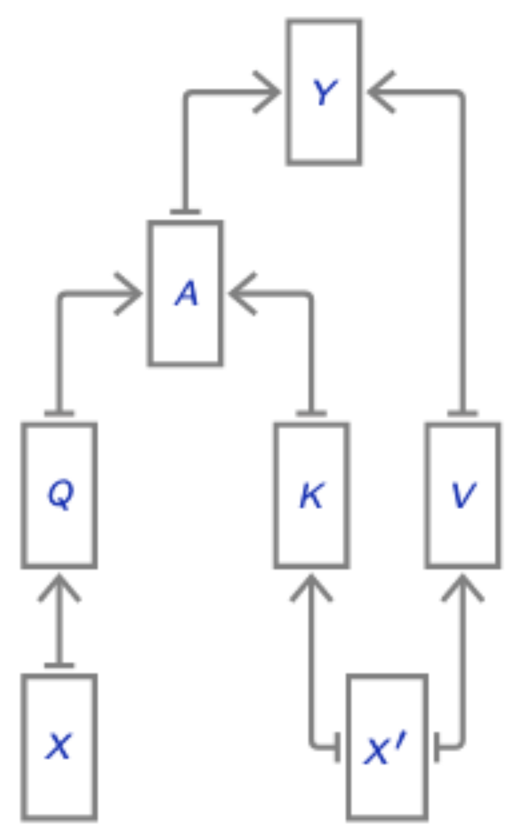


Image credit: François Fleuret

Standard attention layers

- In <u>self-attention layers</u>, we have X=X’

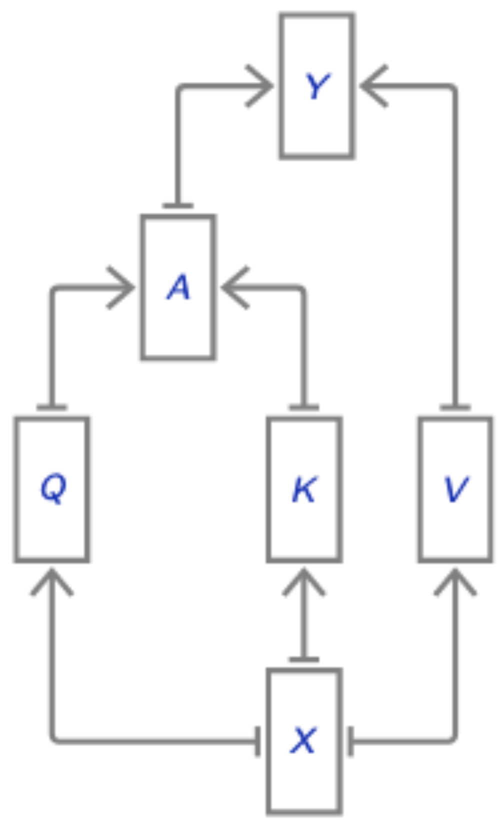


Image credit: François Fleuret

Standard attention layers

- In <u>multi-head attention layers</u>,
  - several (*h* here) such blocks operate in parallel
  - Their output is concatenated in feature dimension

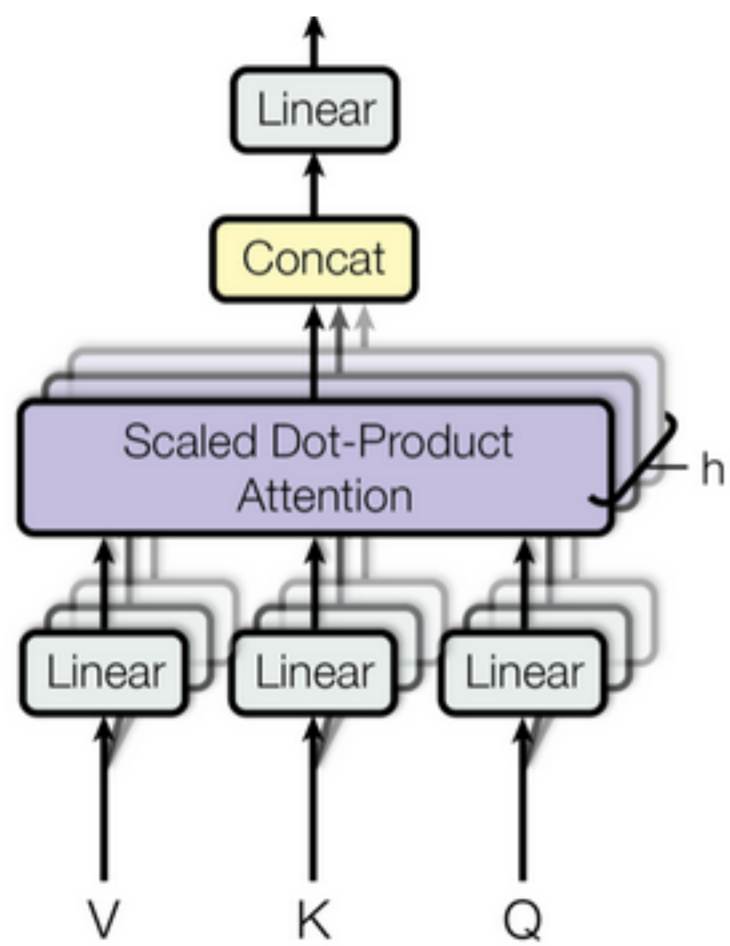


Image credit: Vaswani et al. 2017

The Transformer: The typical attention-based architecture

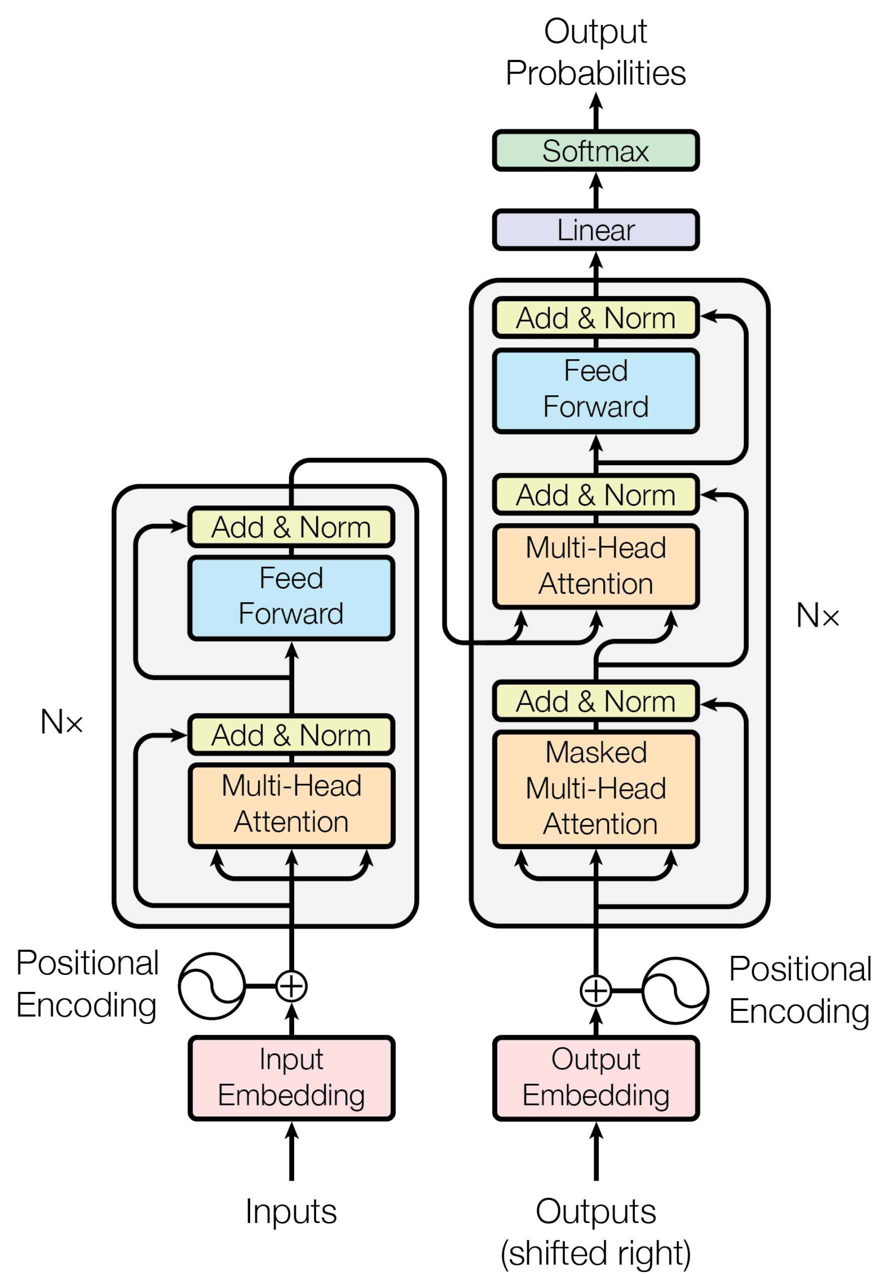


Image credit: Vaswani et al. 2017

Summary

- 1d-CNN, RNN and Attention-based models can be used
  - depends on the context
  - slightly different underlying assumptions
    - locality (ConvNets)
    - sequentiality (RNNs)
    - relationships between any set of items in the sequence (Attention-based models)
  - 1d-CNN are faster to train