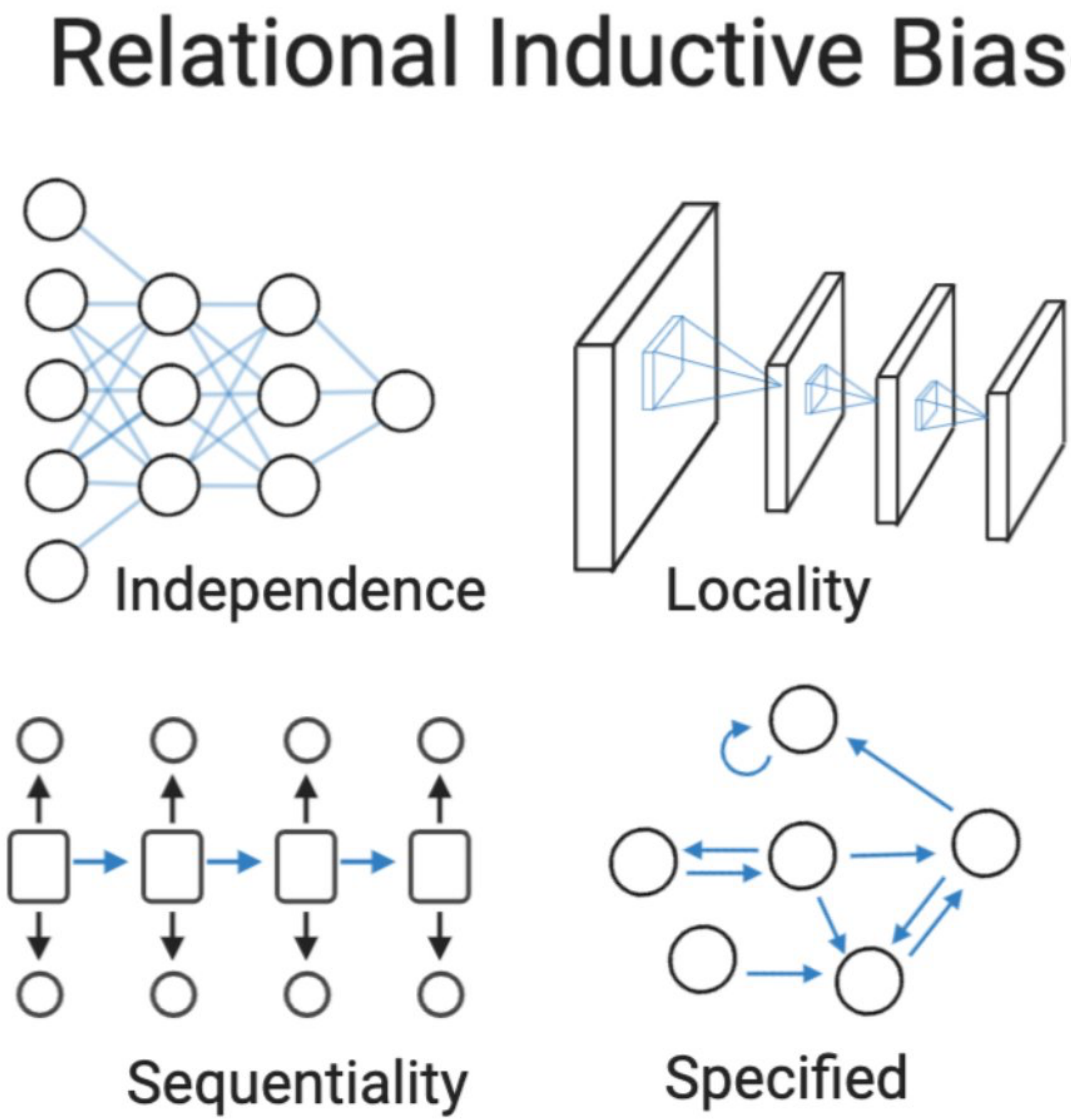
Deep Learning
5. Time series & sequences

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

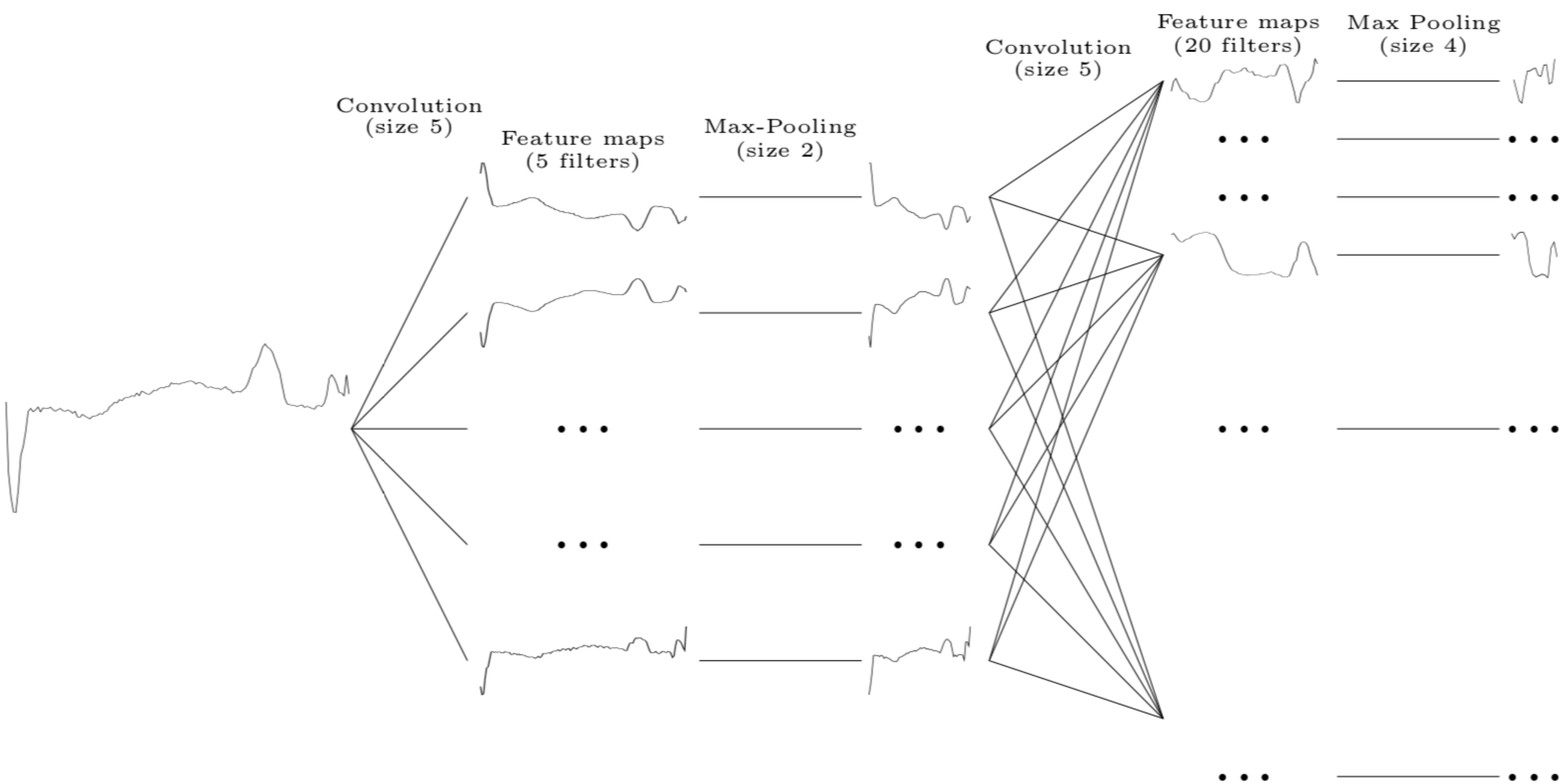
Slides on attention are widely inspired from François Fleuret's excellent Deep Learning cours at UNIGE: https://fleuret.org/dlc/

Neural network architectures and inductive biases



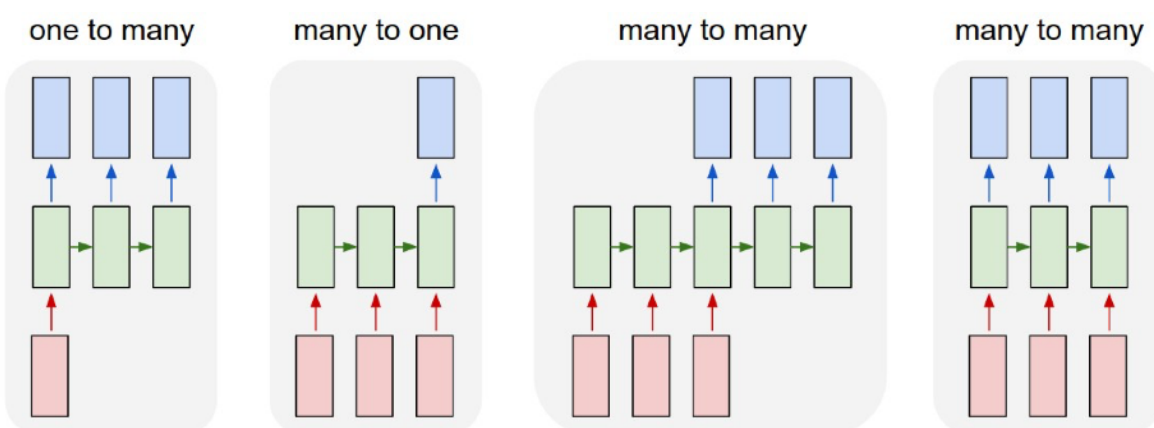
Source: https://sgfin.github.io/2020/06/22/Induction-Intro/

Convolutional neural nets for time series



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

Recurrent neural nets




```
PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.

Second Lord:
They would be ruled after this chamber, and
my fair nues begun out of the fact, to be conveyed,
Whose noble souls I'll have the heart of the wars.
```

Sample text generated by a RNN
trained on Shakespeare words

For $\bigoplus_{n=1,\ldots,m}$ where $\mathcal{L}_{m_\bullet} = 0$, hence we can find a closed subset $\mathcal{H}$ in $\mathcal{H}$ and any sets $\mathcal{F}$ on X, U is a closed immersion of S, then $U \to T$ is a separated algebraic space.

Proof. Proof of (1). It also start we get

$$S = \mathrm{Spec}(R) = U \times_X U \times_X U$$

and the comparicoly in the fibre product covering we have to prove the lemma generated by $\coprod Z \times_U U \to V$. Consider the maps M along the set of points Sch_{fppf} and $U \to U$ is the fibre category of S in U in Section, ?? and the fact that any U affine, see Morphisms, Lemma ??. Hence we obtain a scheme S and any open subset $W \subset U$ in $Sh(G)$ such that $\mathrm{Spec}(R') \to S$ is smooth or an

$$U = \bigcup U_i \times_{S_i} U_i$$

which has a nonzero morphism we may assume that f_i is of finite presentation over S. We claim that $\mathcal{O}_{X,x}$ is a scheme where $x, x', s'' \in S'$ such that $\mathcal{O}_{X,x'} \to \mathcal{O}'_{X',x'}$ is separated. By Algebra, Lemma ?? we can define a map of complexes $\mathrm{GL}_{S'}(x'/S'')$ and we win. □

To prove study we see that $\mathcal{F}|_U$ is a covering of $\mathcal{X}'$, and $\mathcal{T}_i$ is an object of $\mathcal{F}_{X/S}$ for $i > 0$ and $\mathcal{F}_p$ exists and let $\mathcal{F}_i$ be a presheaf of $\mathcal{O}_X$-modules on $\mathcal{C}$ as a $\mathcal{F}$-module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

$$\widetilde{M}^\bullet = \mathcal{I}^\bullet \otimes_{\mathrm{Spec}(k)} \mathcal{O}_{S,s} - i_X^{-1}\mathcal{F})$$

is a unique morphism of algebraic stacks. Note that

$$\mathrm{Arrows} = (Sch/S)^{opp}_{fppf}, (Sch/S)_{fppf}$$

and

$$V = \Gamma(S, \mathcal{O}) \longmapsto (U, \mathrm{Spec}(A))$$

is an open subset of X. Thus U is affine. This is a continuous map of X is the inverse, the groupoid scheme S.

Proof. See discussion of sheaves of sets. □

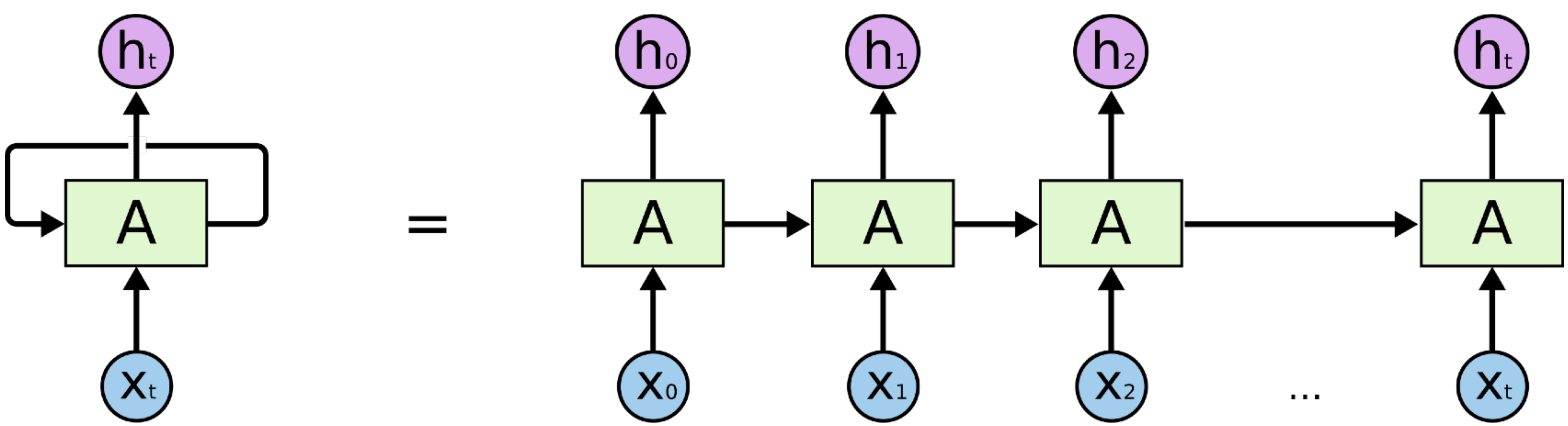
The result for prove any open covering follows from the less of Example ??. It may replace S by $X_{spaces,étale}$ which gives an open subspace of X and T equal to S_{Zar}, see Descent, Lemma ??. Namely, by Lemma ?? we see that R is geometrically regular over S.

Sample LaTeX generated by a RNN
trained on a book of algebraic geometry

Source: Andrej Karpathy's blog, http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Recurrent neural nets

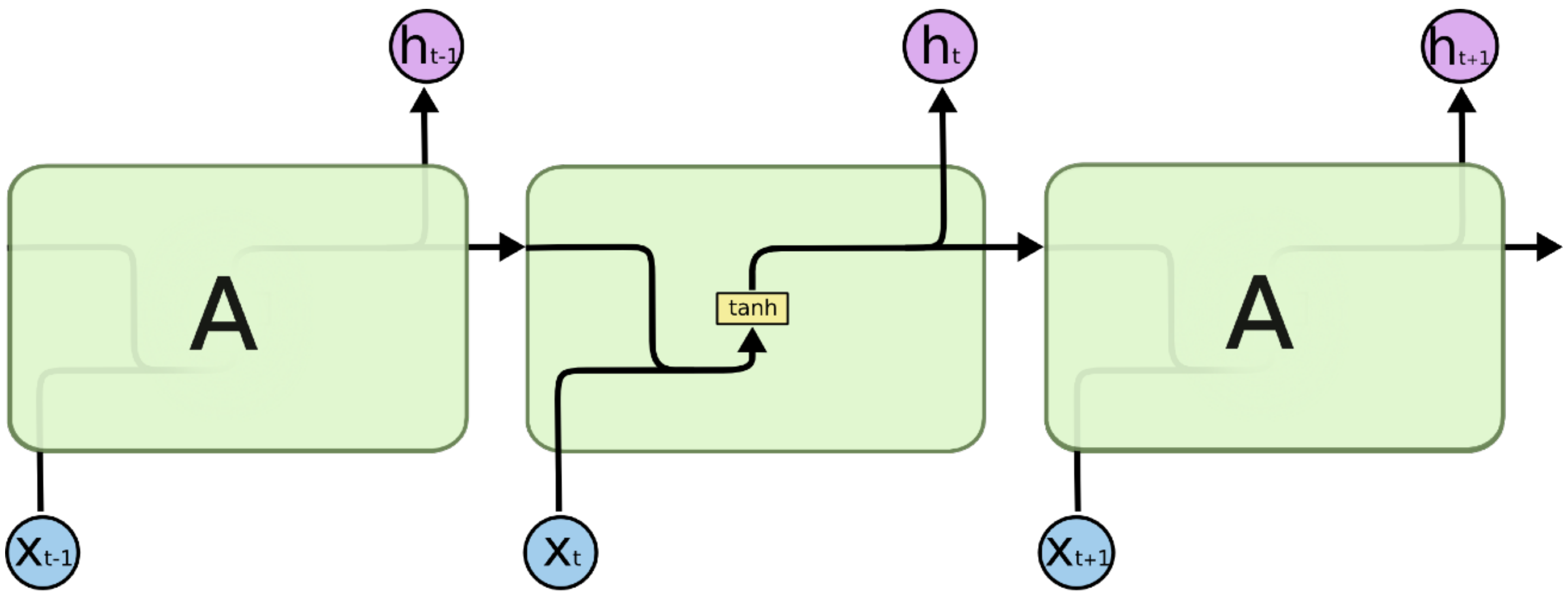
- Very flexible model (any length, let the model learn its memory needs, …)



Source: Christopher Olah’s blog

Recurrent neural nets

- "Vanilla" RNN in more details



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

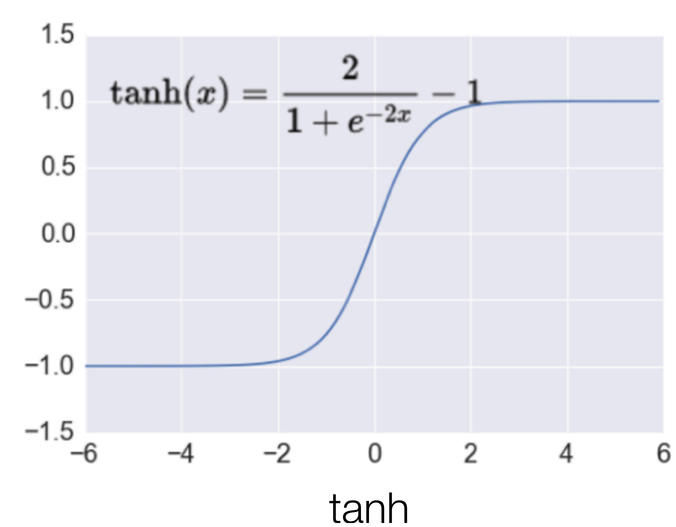
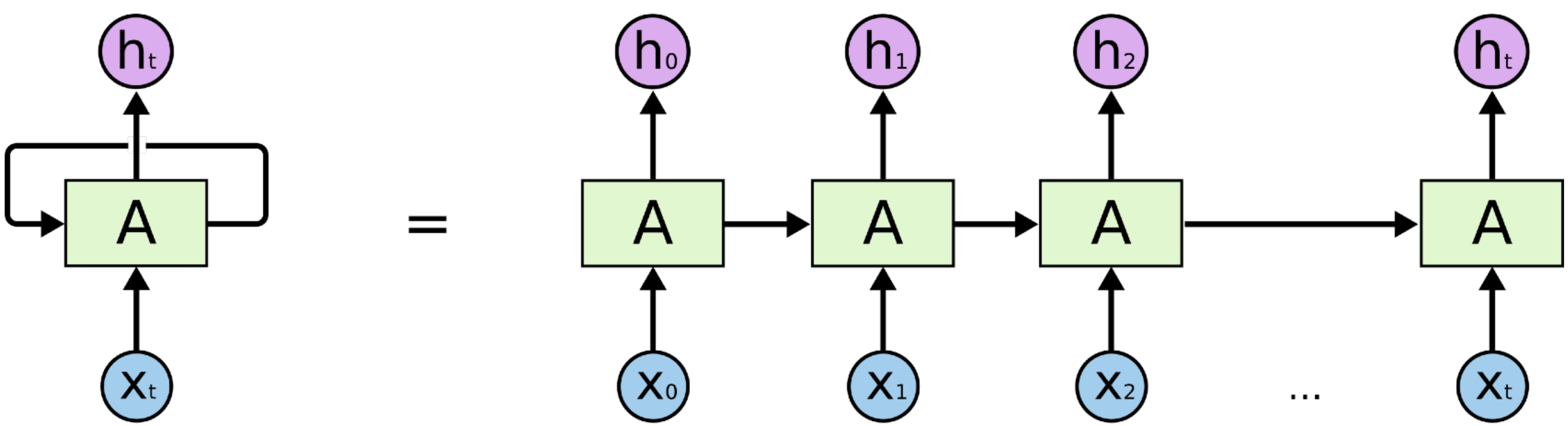


Illustration: RNN cell, source: Christopher Olah's blog

Recurrent neural nets

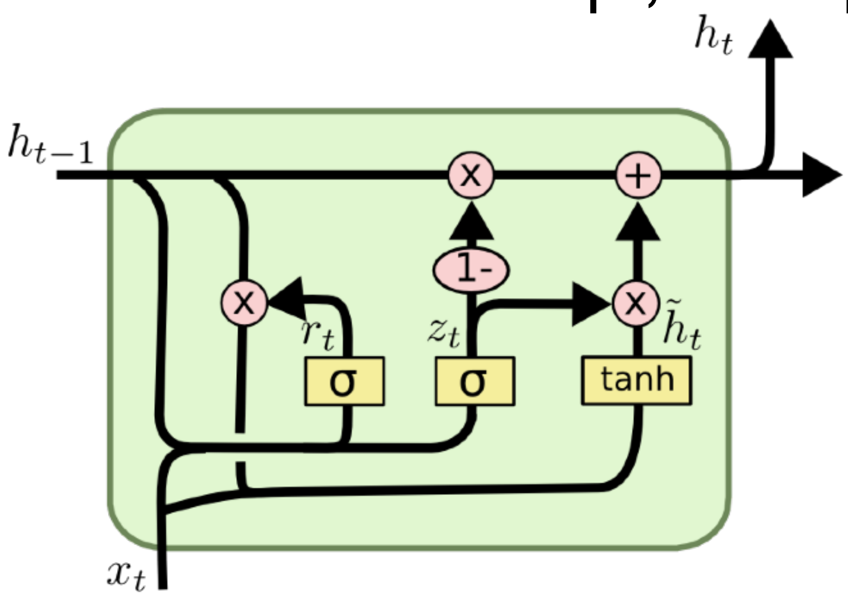
- Very flexible model (any length, let the model learn its memory needs, …)
- Difficult to learn in practice
  - Slow (lack of parallelism)
  - Vanishing gradients (hard to learn long-term dependencies)



Source: Christopher Olah's blog

Recurrent neural nets

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



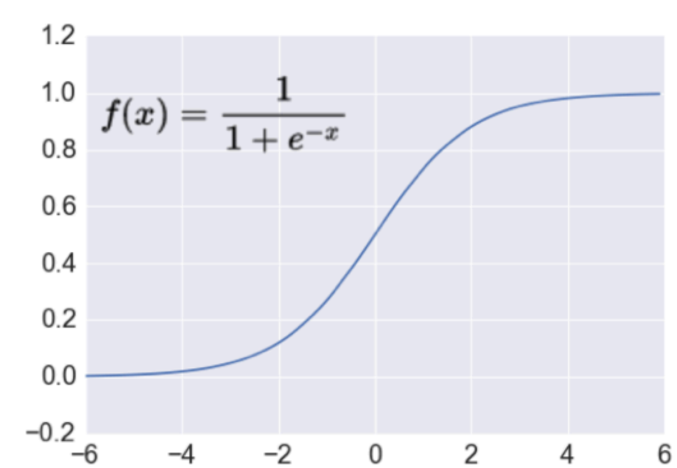
$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

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

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

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

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



sigmoid σ

Recurrent neural nets

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

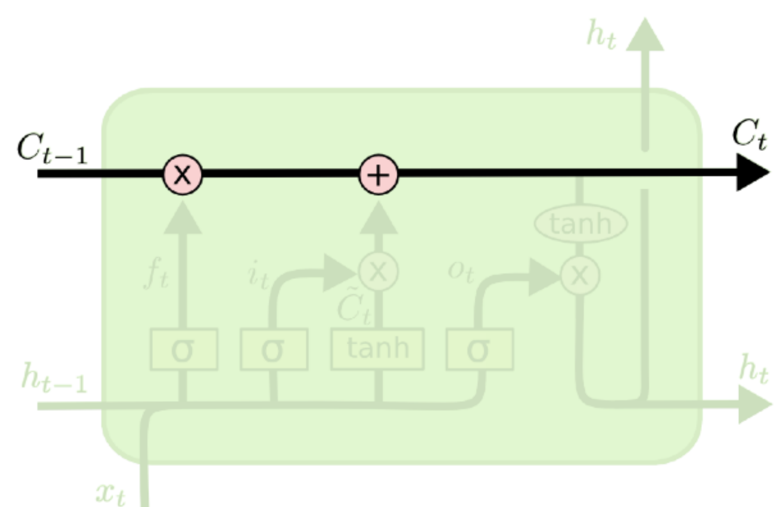
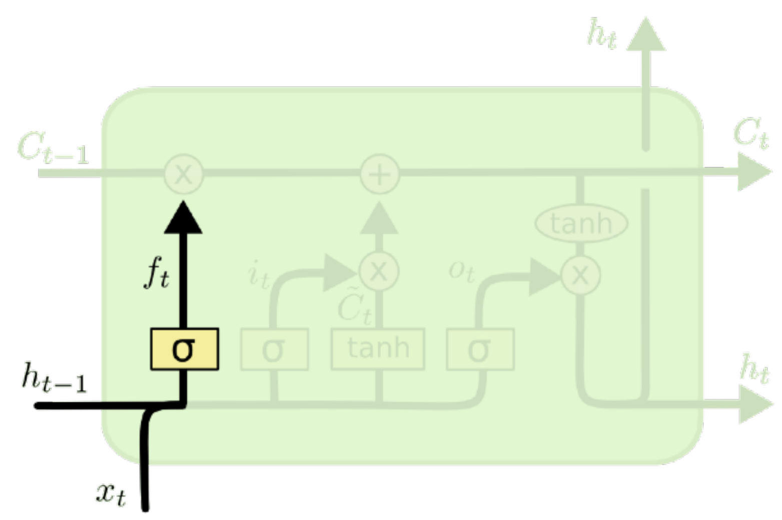


Illustration: LSTM cell, source: Christopher Olah’s blog

Recurrent neural nets

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

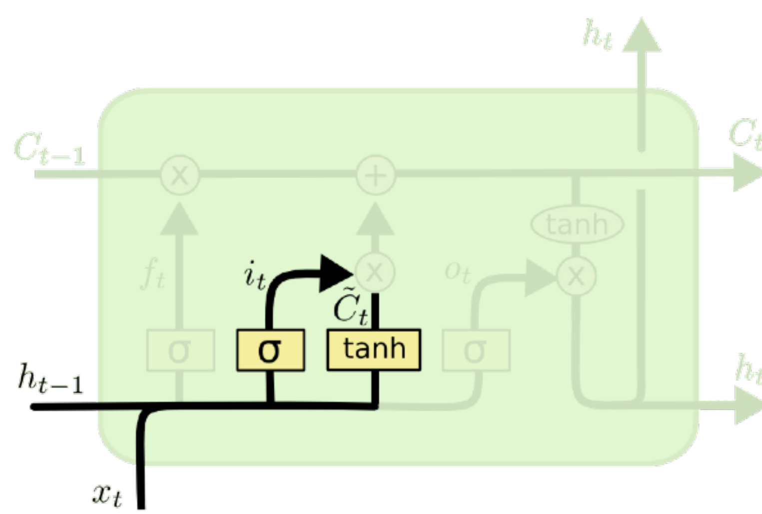


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

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

Recurrent neural nets

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

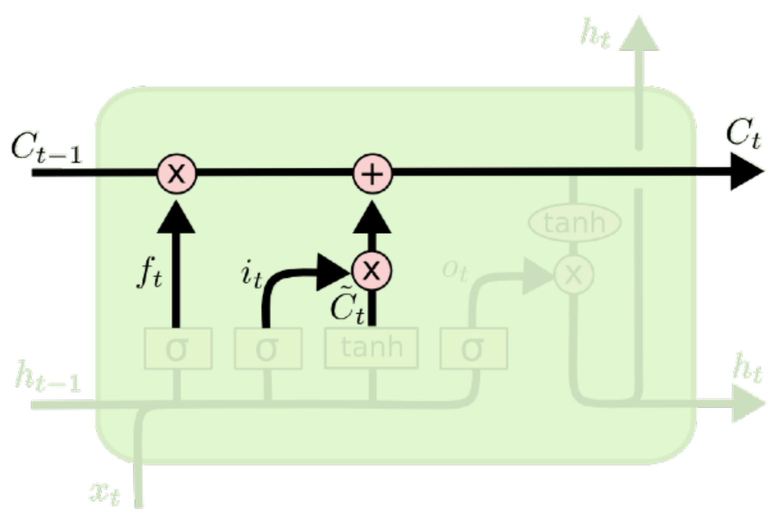


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

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

Recurrent neural nets

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

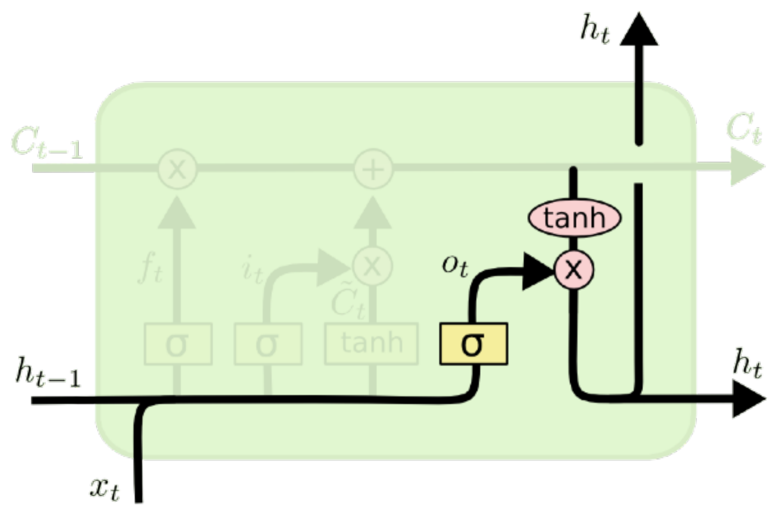


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

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

Recurrent neural nets

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



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

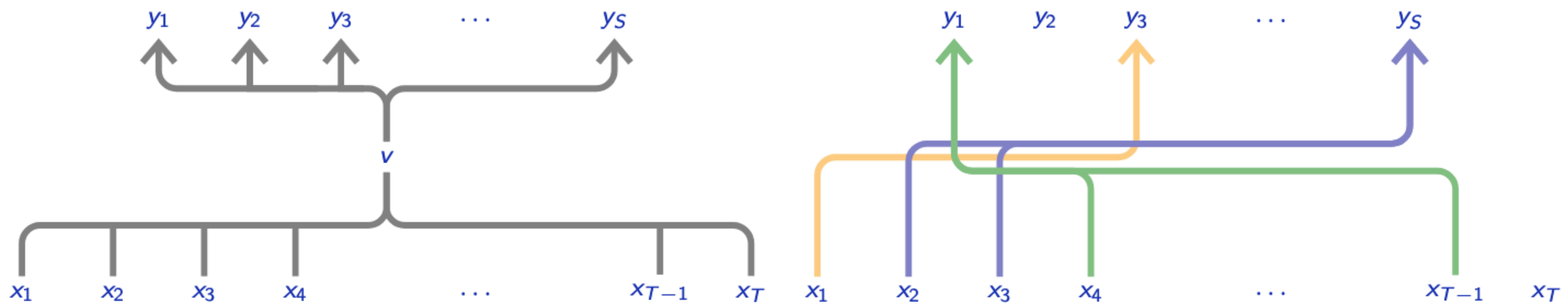
Illustration: LSTM cell, source: Christopher Olah's blog

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

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

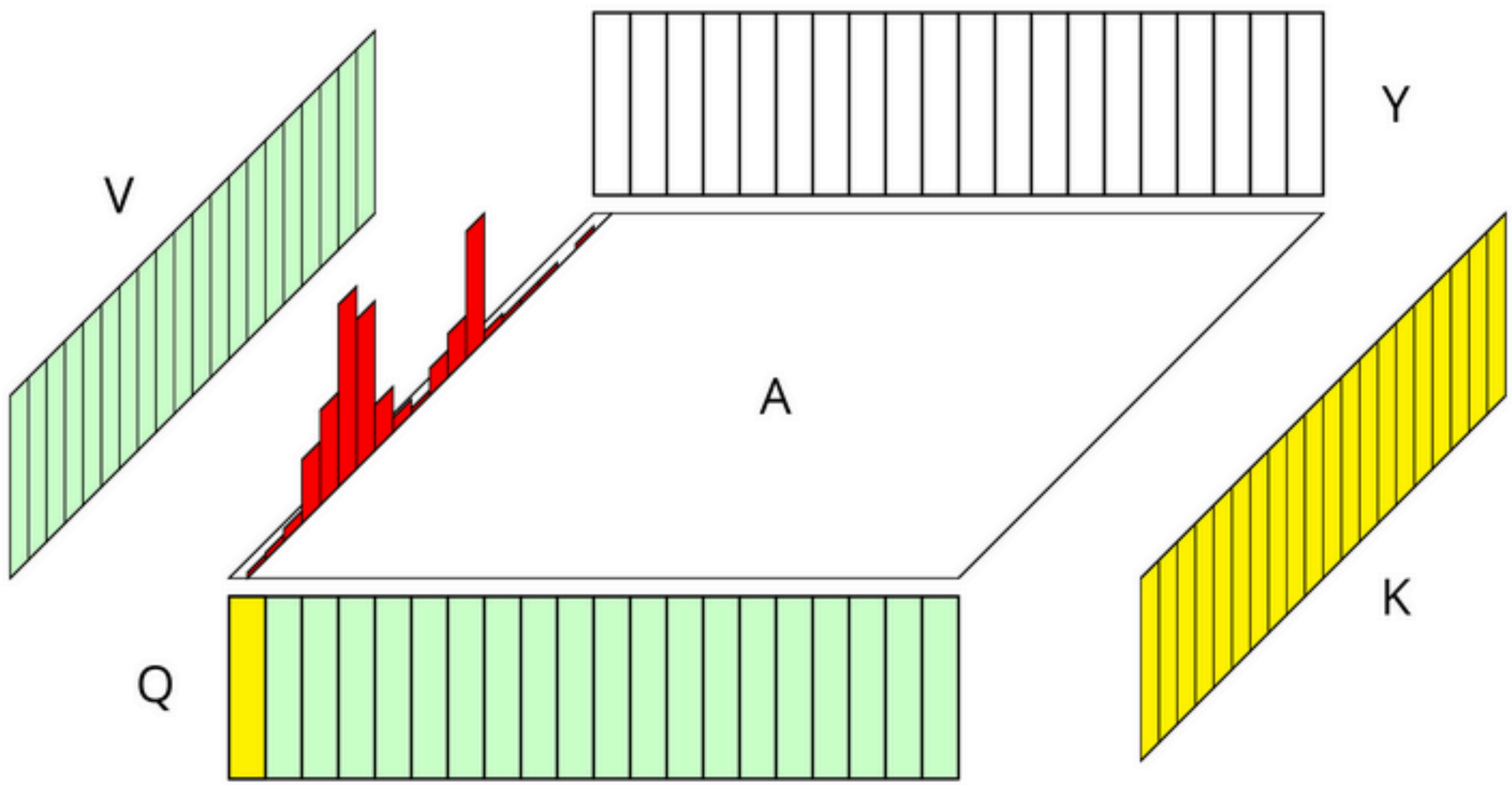


Image credit: François Fleuret

Visual explanation of the Attention mechanism

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

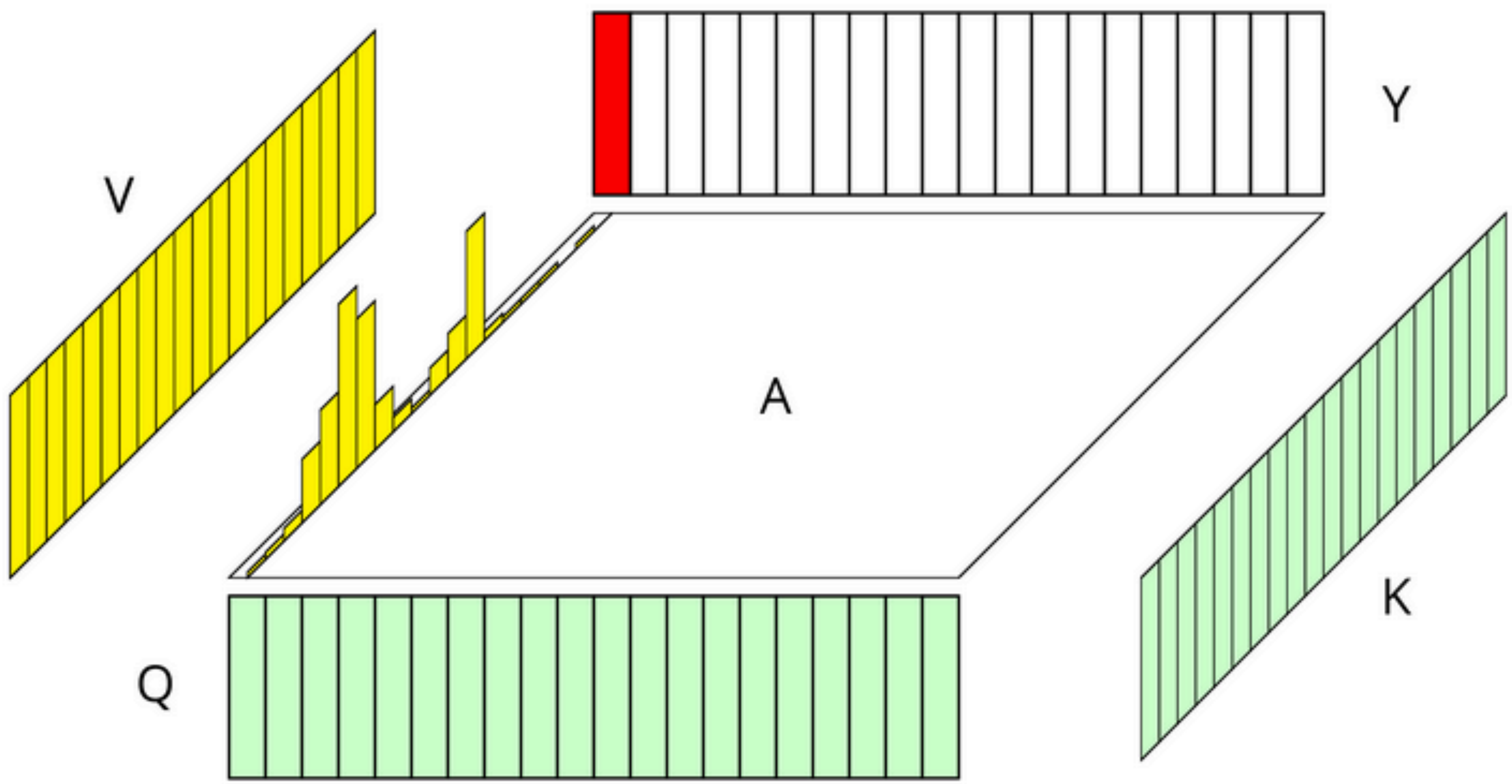


Image credit: François Fleuret

Visual explanation of the Attention mechanism

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

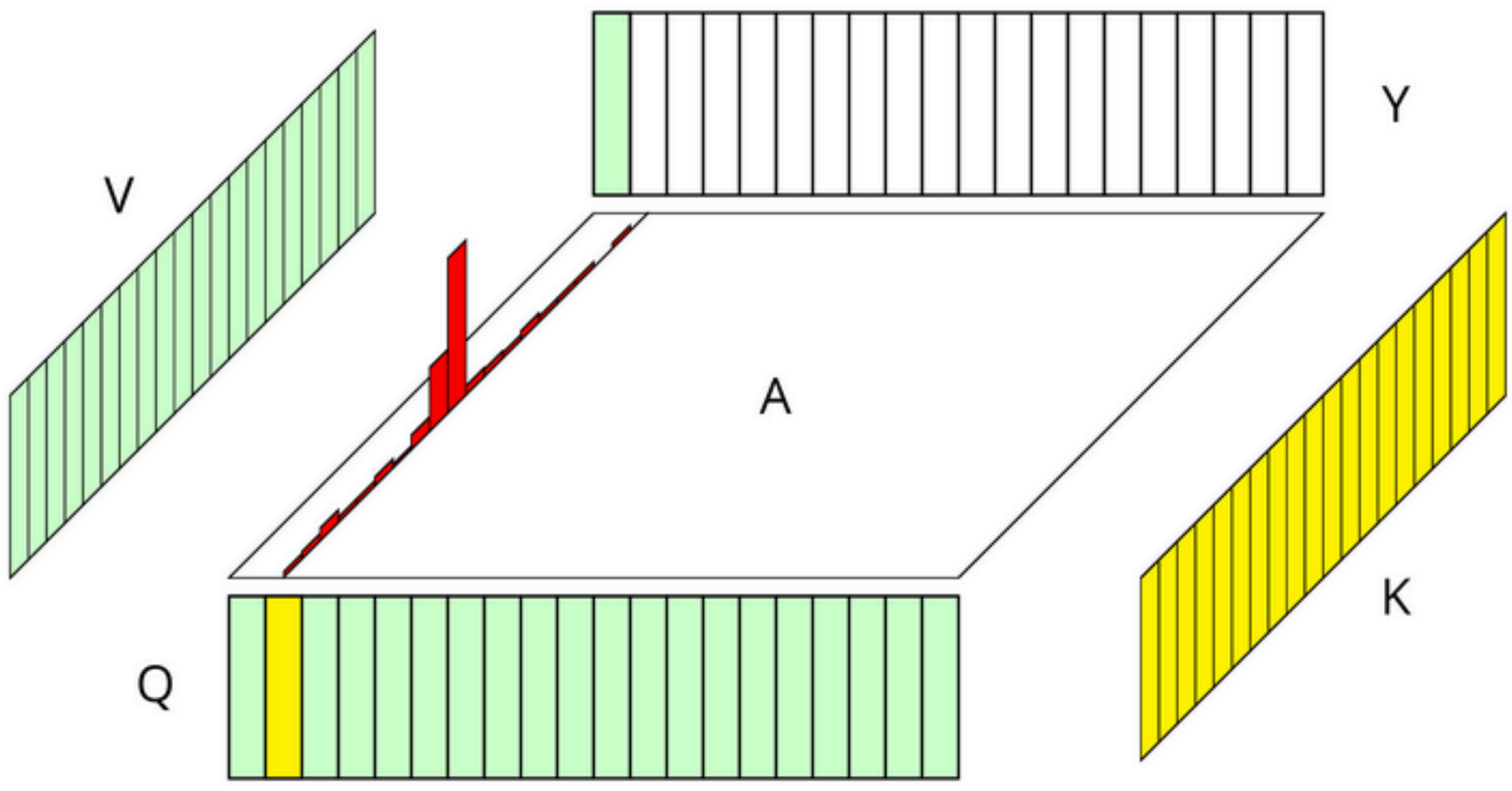


Image credit: François Fleuret

Visual explanation of the Attention mechanism

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

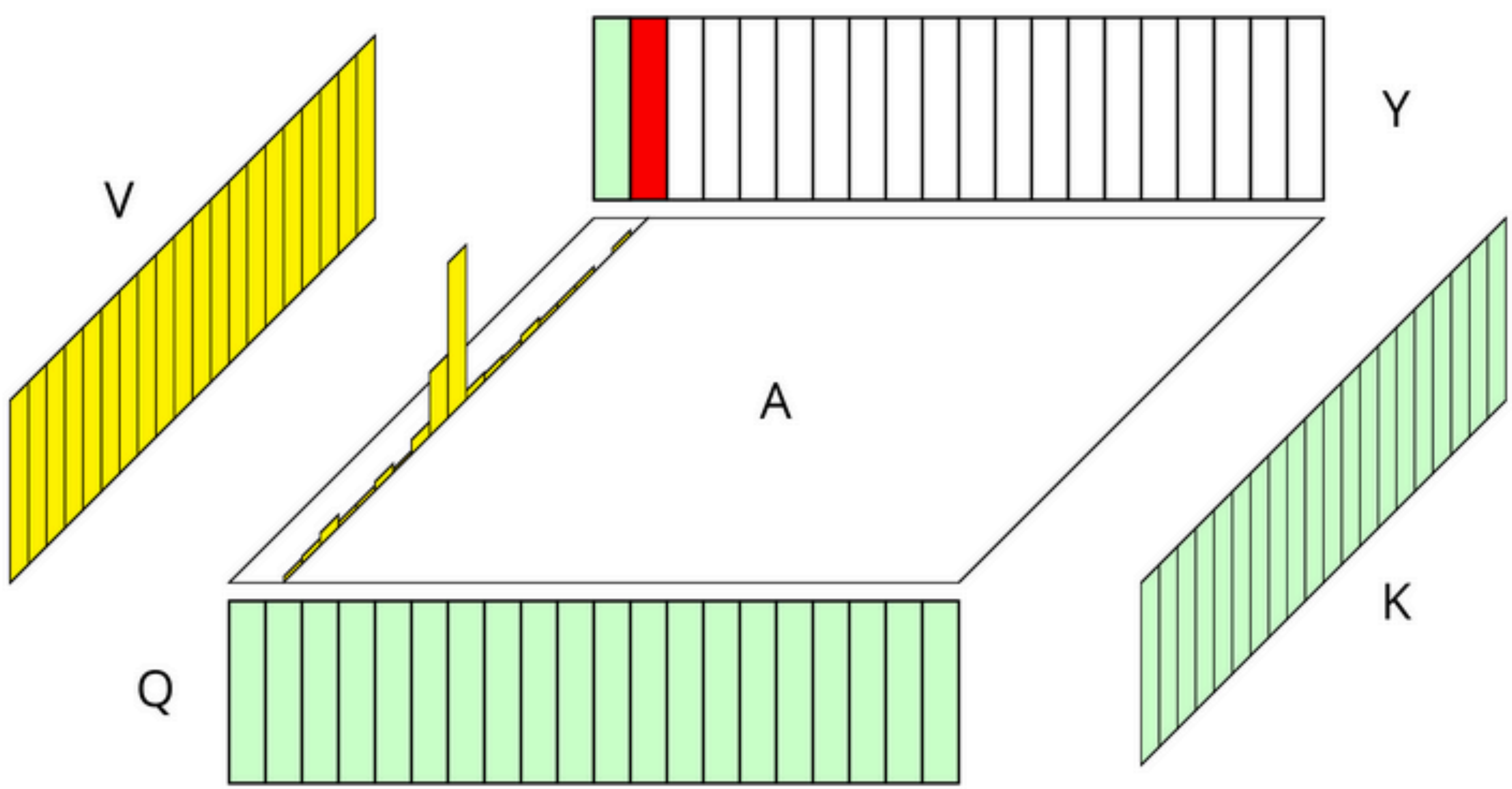


Image credit: François Fleuret

Standard attention layers

- Standard Attention layers
  - take as inputs 2 sequences X and X’
  - output a sequence Y

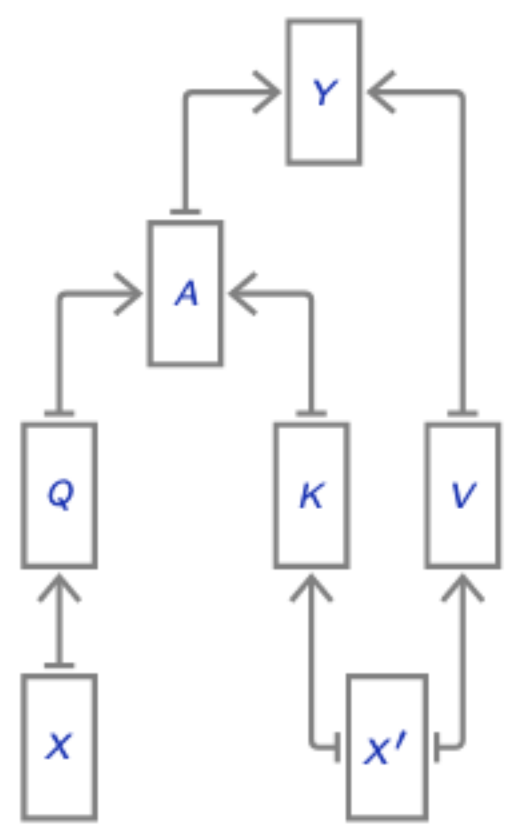


Image credit: François Fleuret

Standard attention layers

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

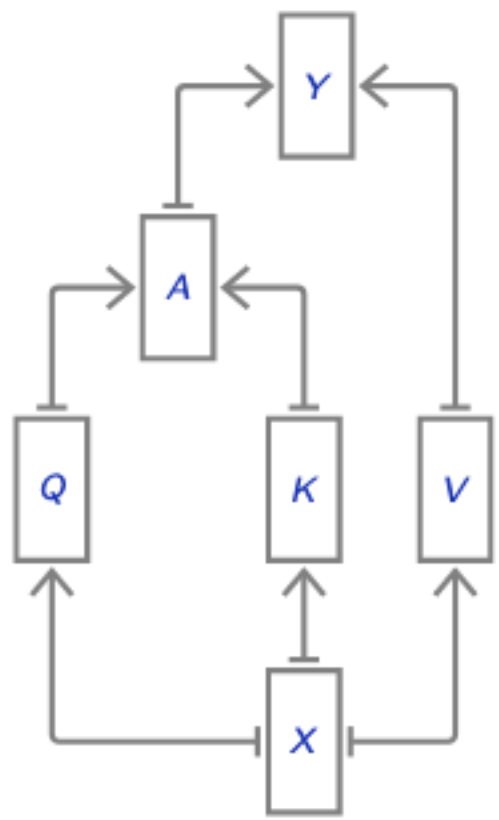


Image credit: François Fleuret

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

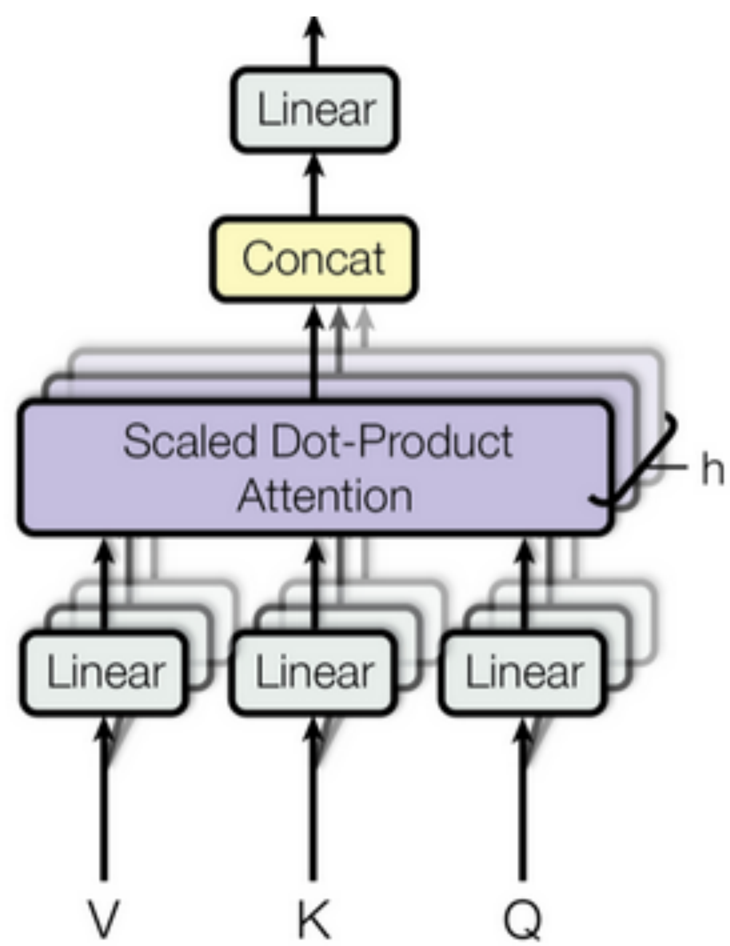


Image credit: Vaswani et al. 2017

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