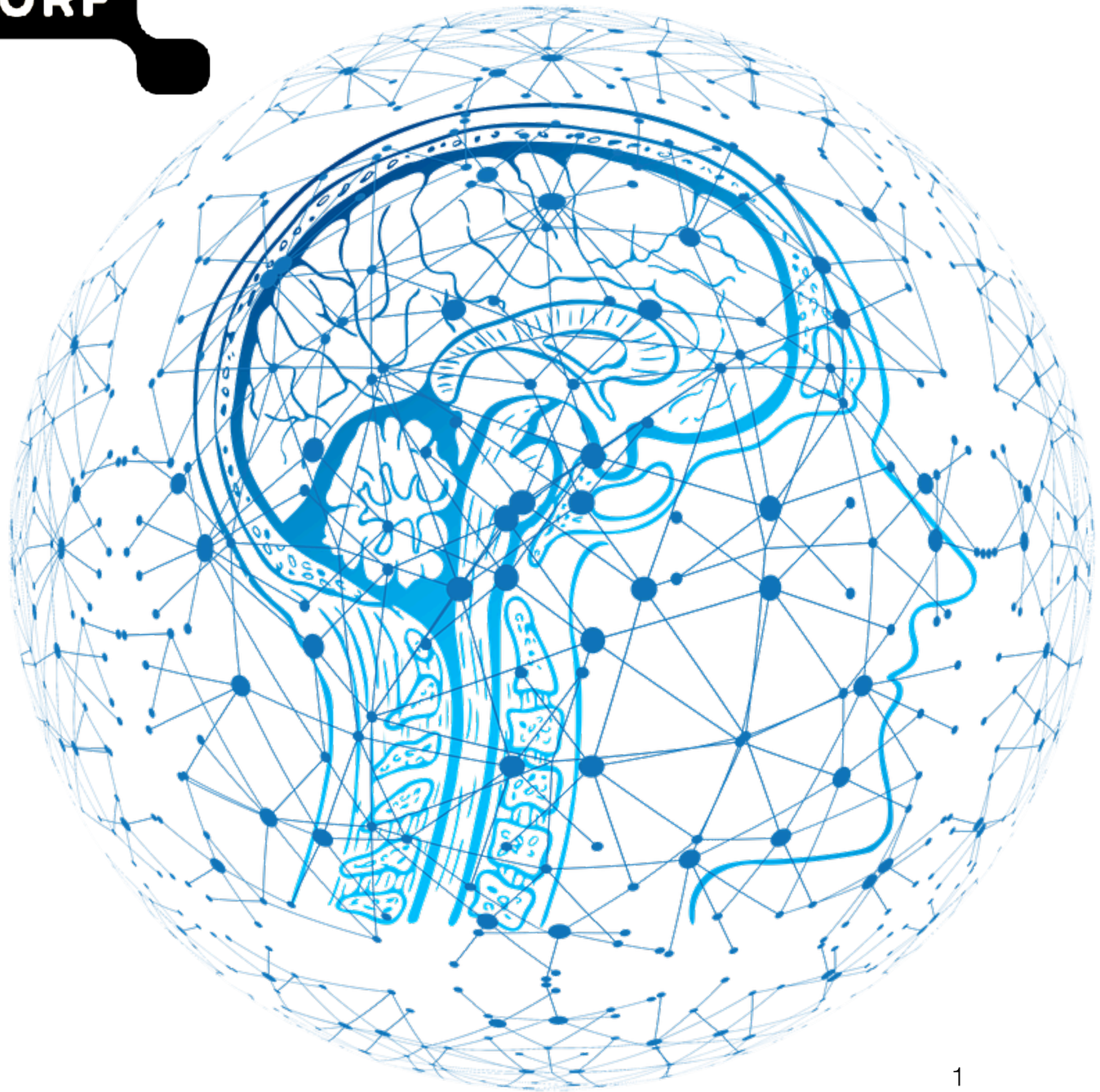
SURF



Introduction to Deep Learning

Recurrent Neural Networks

Maxwell Cai

maxwellcai.com

Wouldn’t it be easier to handle sequences (1D) than images (2D)?

Yes and no…

Sequence Modeling

Language translation

Speech recognition

Music generation

Sentiment classification

Video captioning

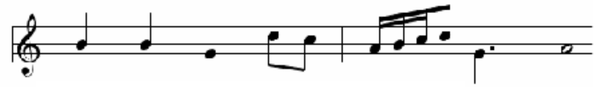
Chatbot

Text summary

...

“Hoi, hoe gaat het?” → “Hi, how are you?”

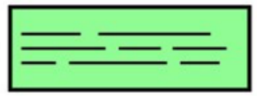
 → “Hi, how are you?”

Style / nothing → 

“Hoi, hoe gaat het?” → ★★★★★

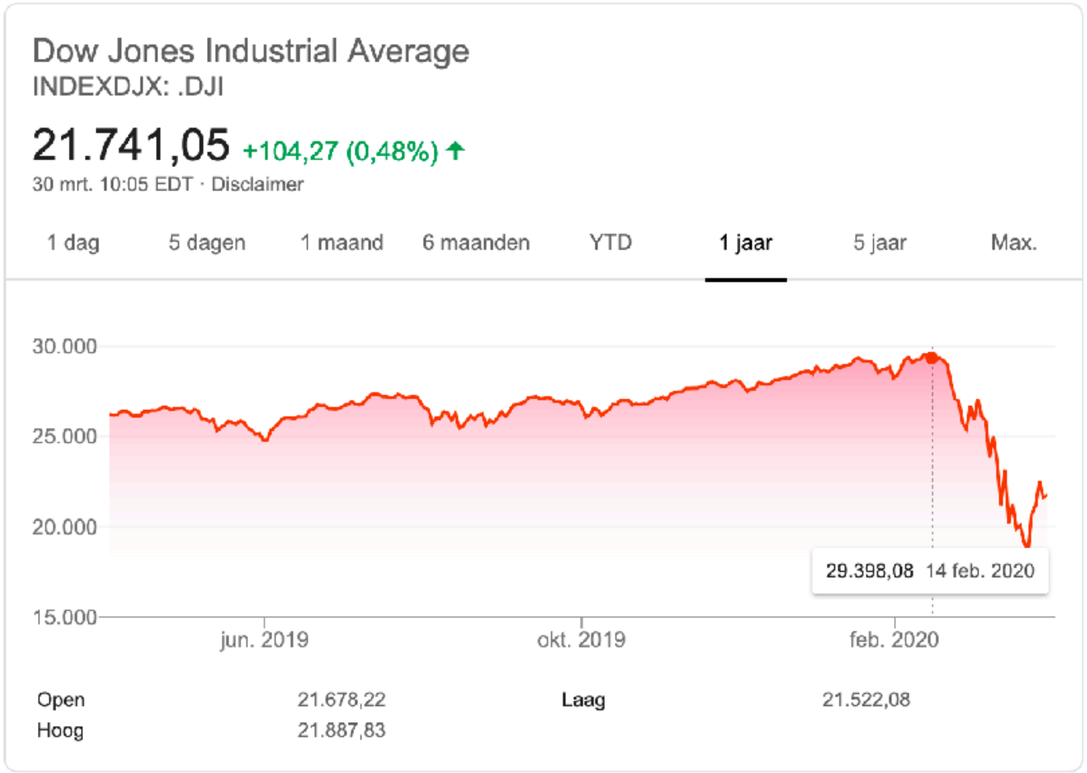
 → A person is running

“Tell me my day” → You have two appointments.

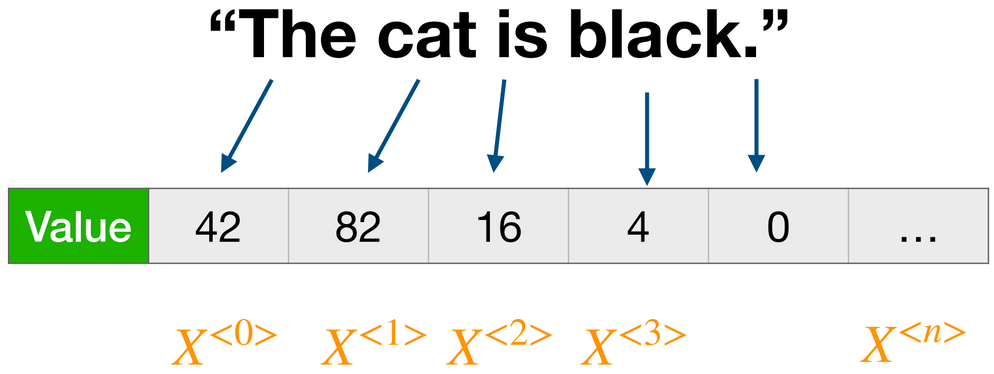
 → 

Sequential Data

Sequential data are represented with **1D or 2D arrays**.



Time	24/3	25/3	26/3	27/3	28/3	…
Value	19.722	21.050	21.188	22.532	21.678	…

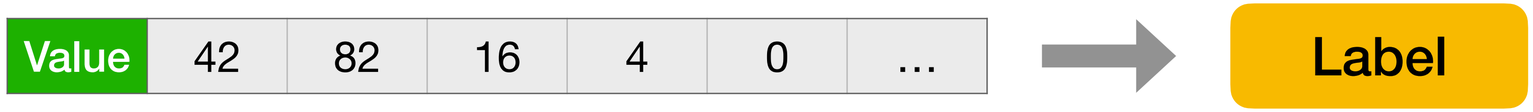


In **natural language processing** (NLP),

- Each **word** is converted into a **token**;
- A token is typically represented as a **numerical** value;
- A **sentence** is a **sequence** of tokens;
- A sequence can have **arbitrary length**;
- Tokens may be **logically related** to each other.

Sequential Data Modeling

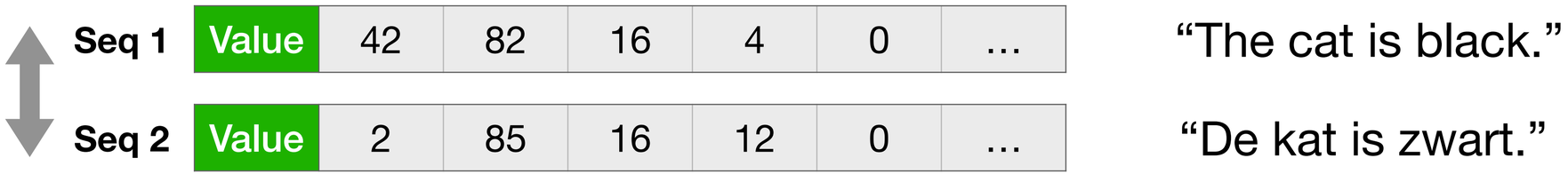
Classification



Generation

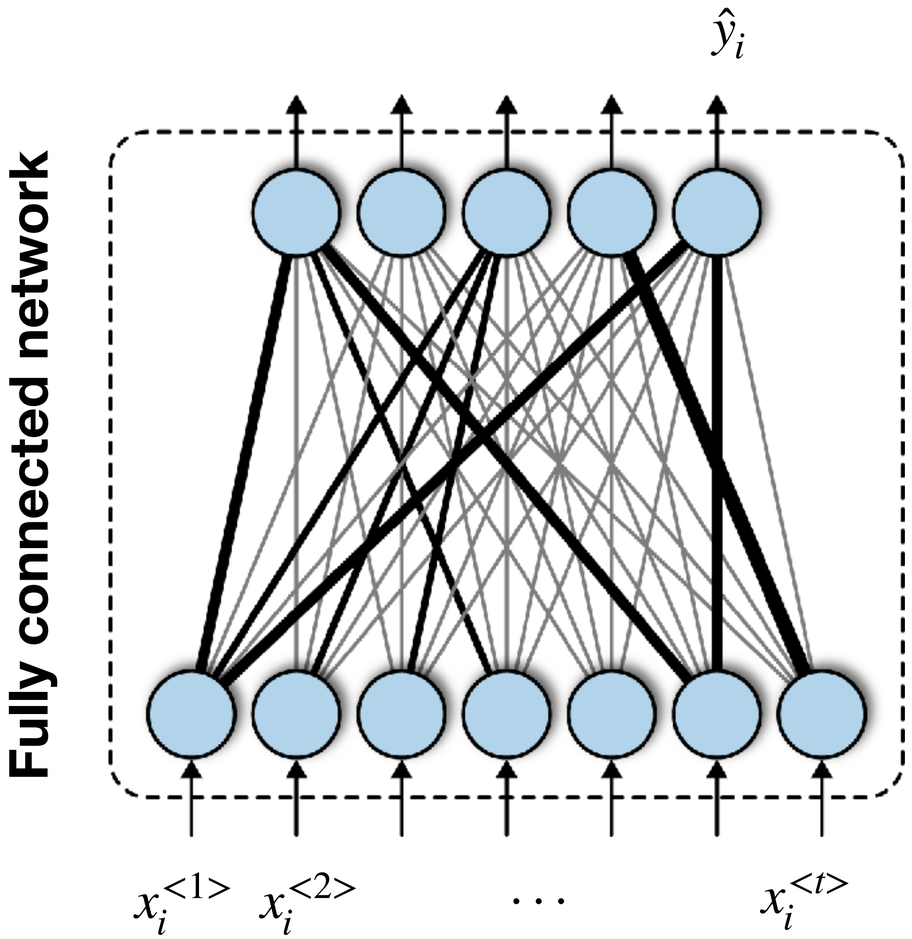
Regression

Sequence to Sequence (e.g., Translation)



The fundamental problem in sequence model is to understand the **relations between tokens**/elements.

Naive Approaches



Modeling the **full sentence** using a fully connected network requires a **huge number of parameters**!

A FC network needs to model the pairwise relation between each word, thus the network scales as $\mathcal{O}(k^n)$, where k is size of the vocabulary and n is the length of the sequence.

For example, consider that in the English language, there are $k = 2000$ words. To model a sentence of $n = 100$ words, one single FC layer requires 2000^{100} parameters!!

Sequence modeling with 1D convolution

CNNs works because they can take into account the relation between **nearby** tokens.

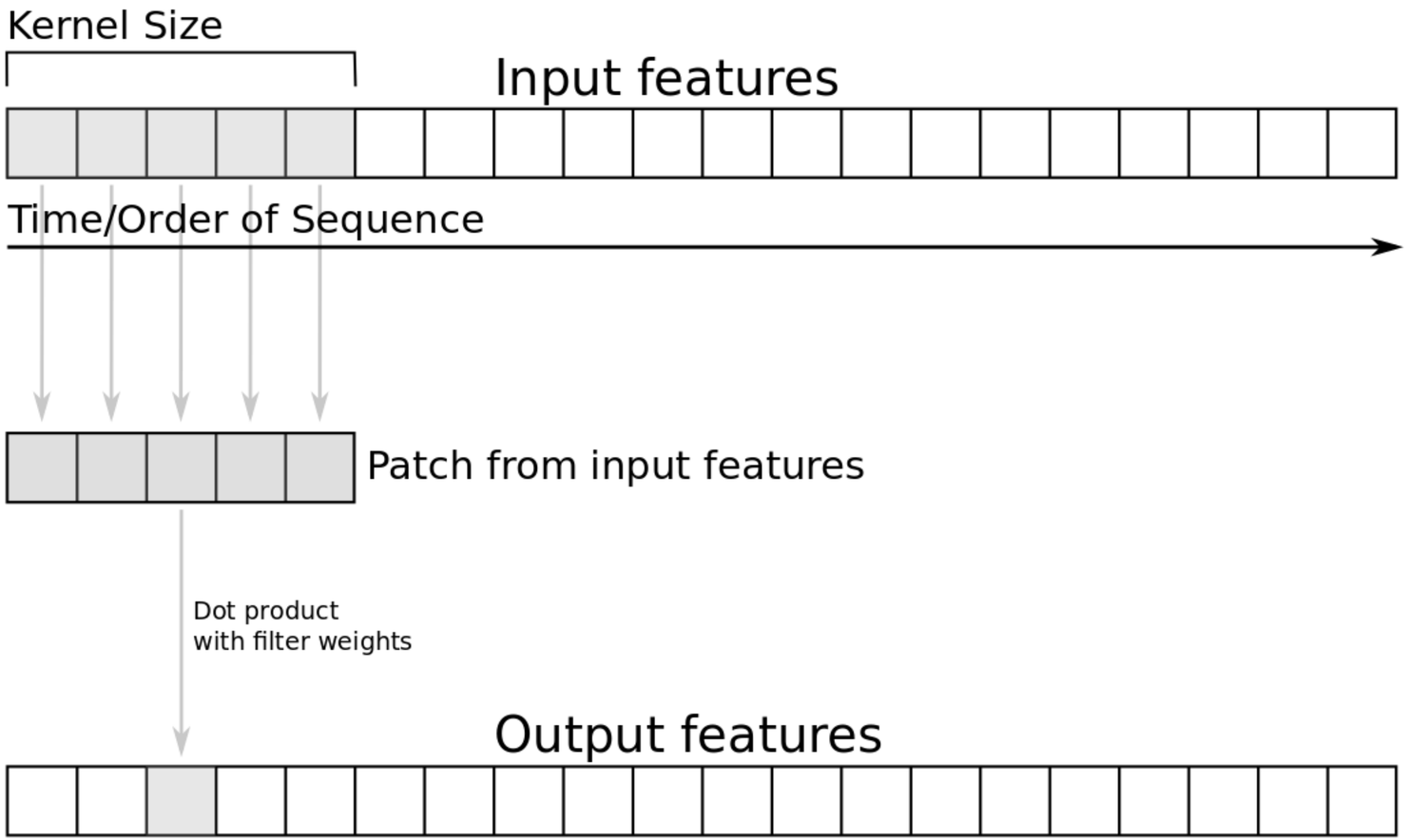
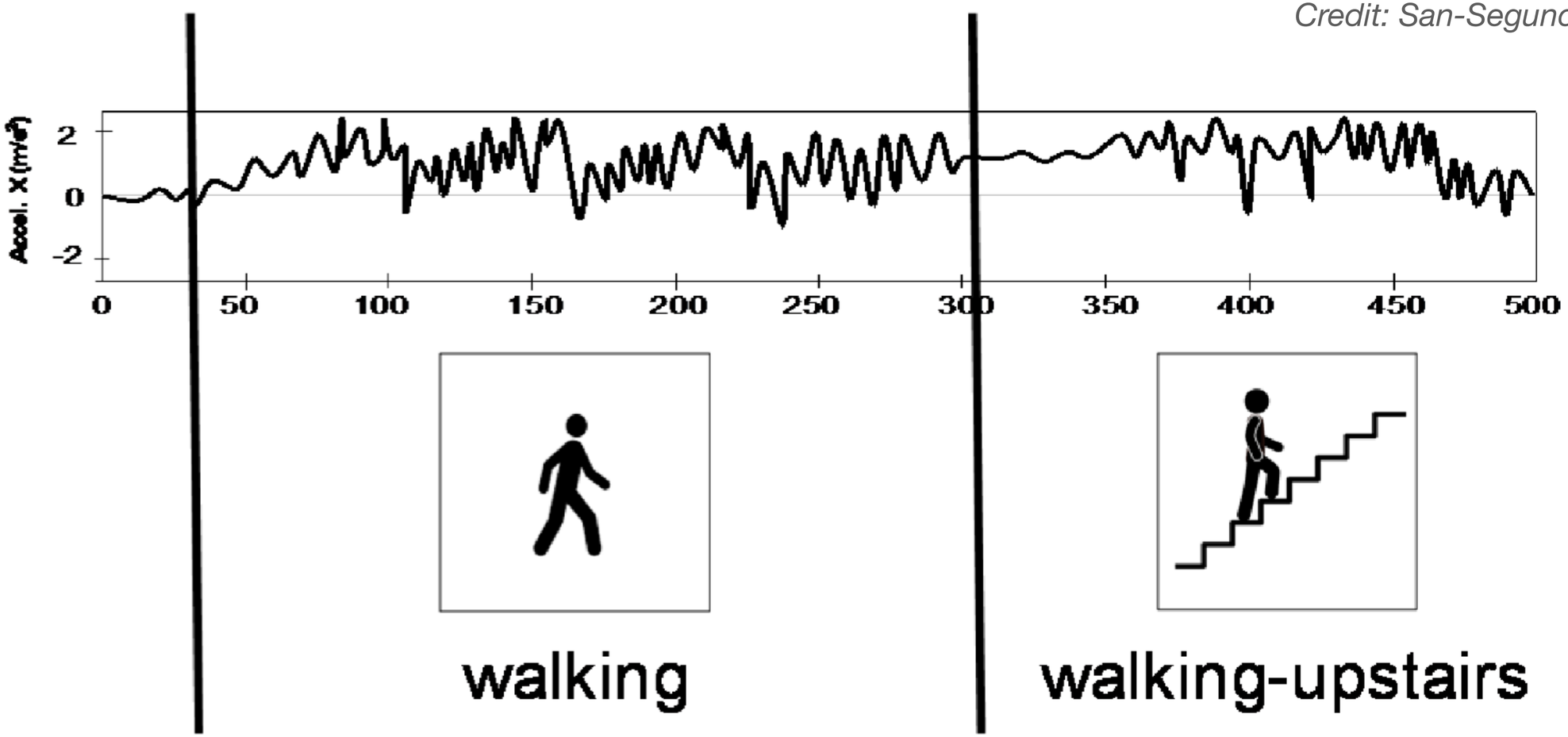


Image source: “Deep Learning With Python” (Francois Chollet)

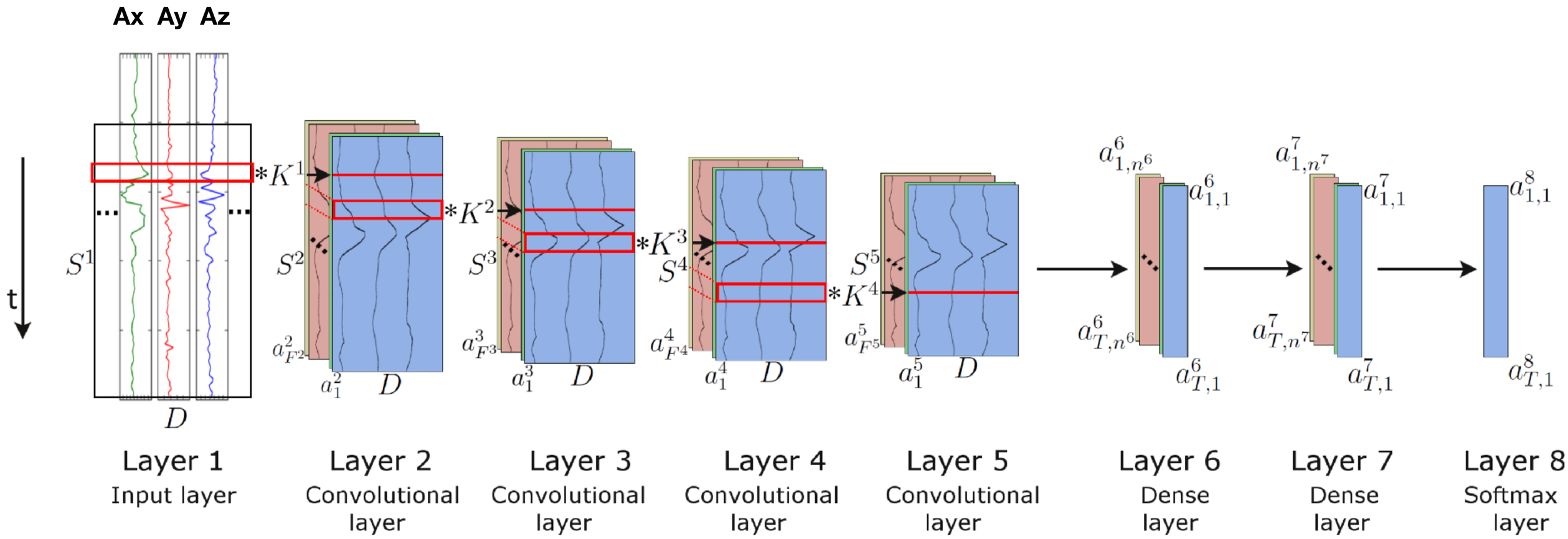
Example: human activity recognition

Credit: San-Segundo et al. (2016)



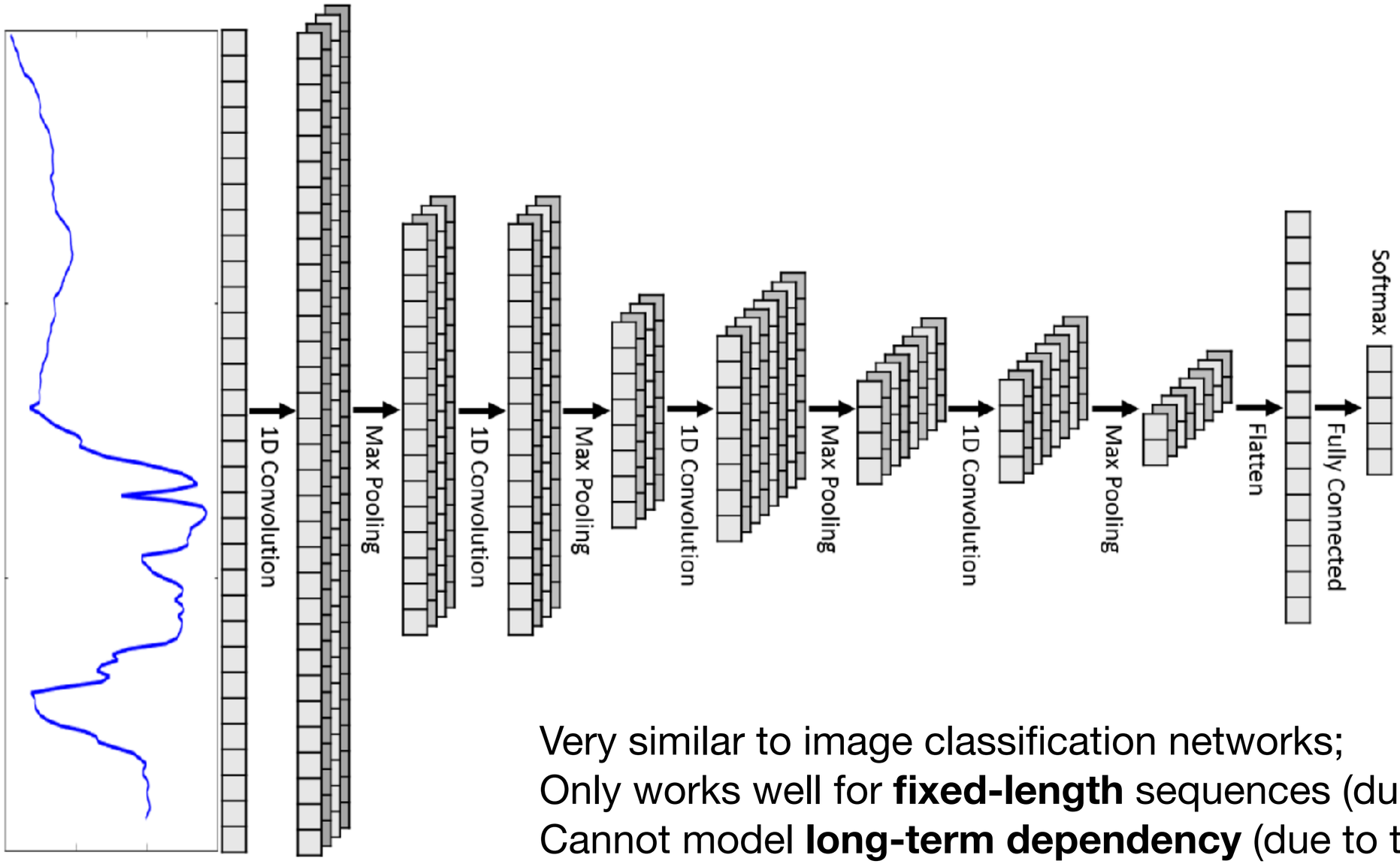
The DNN can be trained with **existing** open datasets!
e.g., https://archive.ics.uci.edu/ml/datasets/human+activity+recognition+using+smartphones

Human activity recognition with 1D Convolution



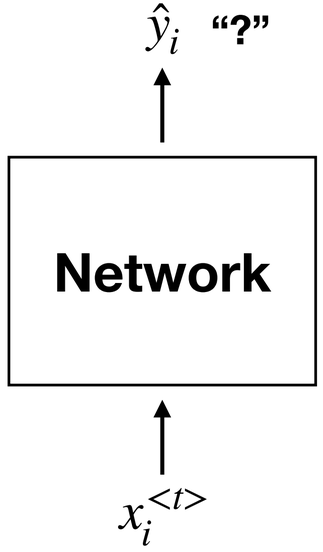
Ordóñez et al. (2016)

Sequence modeling with 1D convolution



Very similar to image classification networks;
Only works well for **fixed-length** sequences (due to the FC layer);
Cannot model **long-term dependency** (due to the limited kernel size).

Naive Approaches v2



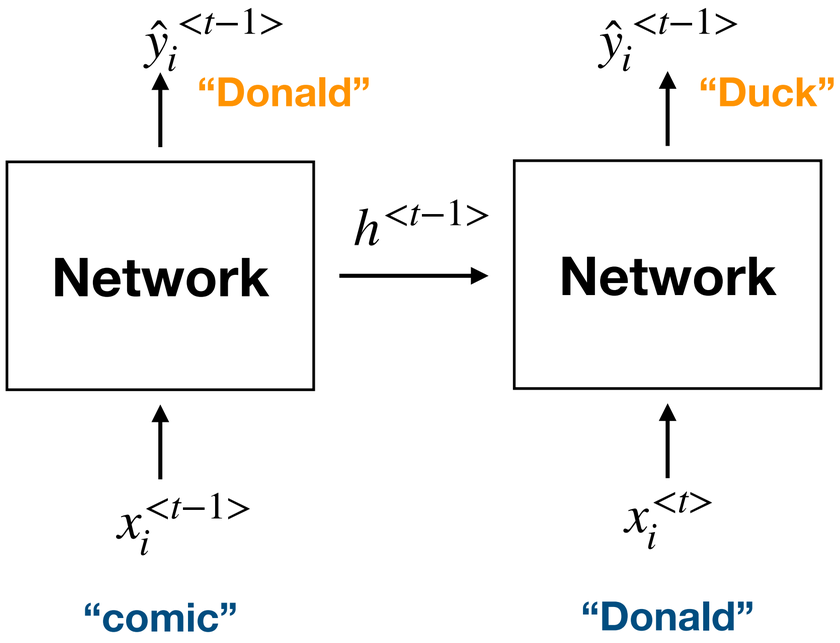
How about modeling just the next word?

The formal president of the United States is Donald Trump.
Wherever Donald Trump went, security was tight.

The comic Donald Duck is very famous.

Context matters to the prediction!

Context Propagation



The meaning of the sentence depends on the **context**.

The context at $x_i^{<t-1>}$ is propagated to the next time step via the **hidden state** $h_i^{<t-1>}$.

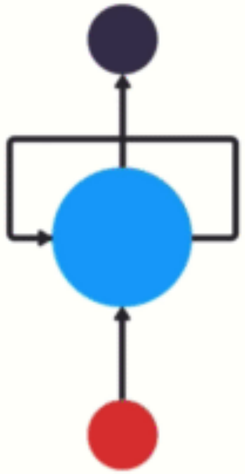
The next prediction is then modeled with both $x_i^{<t>}$ and $h_i^{<t-1>}$.

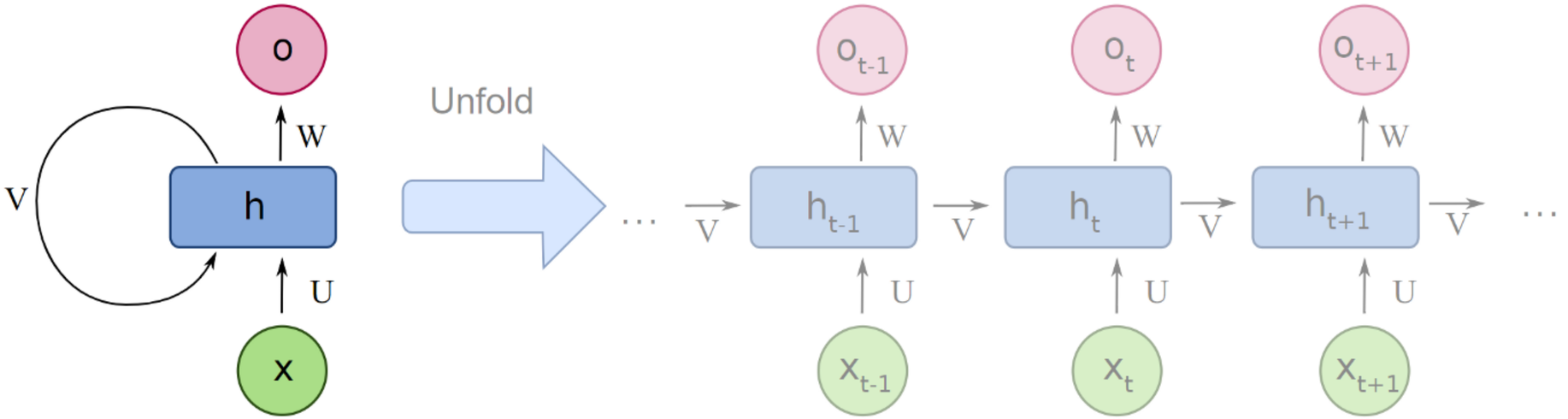
The introduction of the hidden state h_i allows the network to have **memory**.

"The comic Donald Duck is very famous."

Recurrent Neural Networks

In recurrent neural networks (RNNs), weights are **reused** across multiple time steps.





RNN

$$h_t = \sigma(Ux_t + Vh_{t-1} + b_h)$$

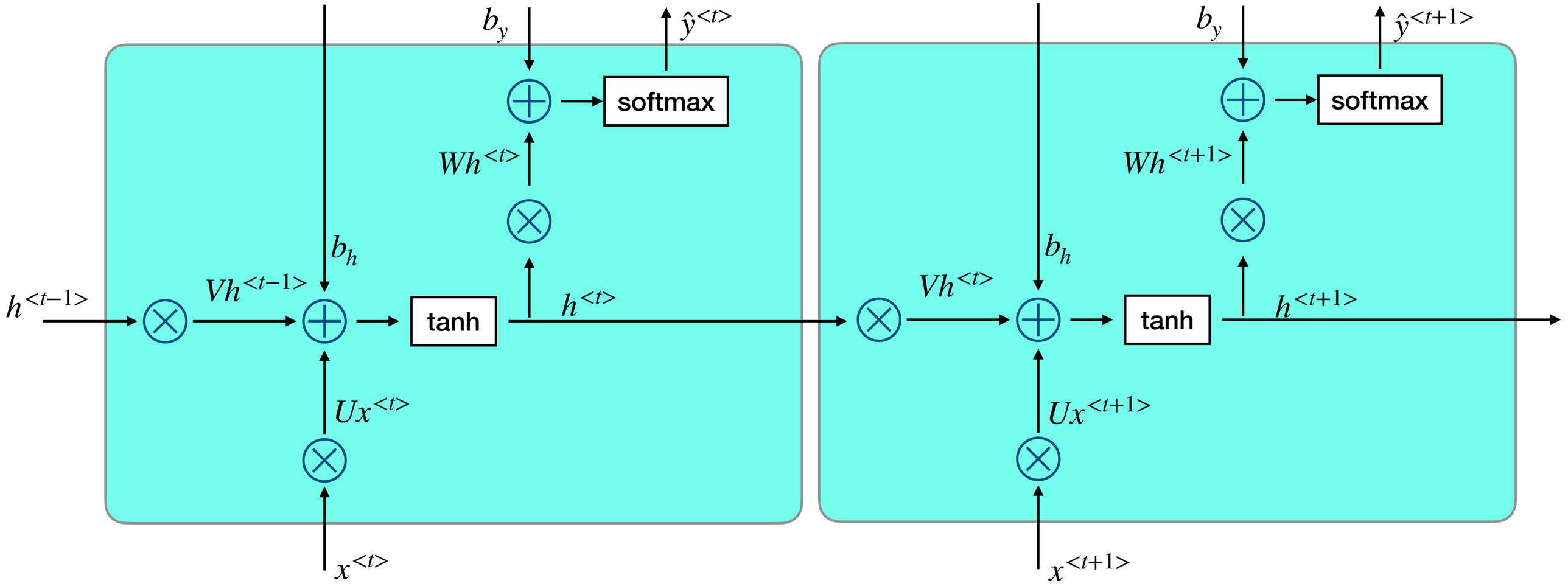
$$O_t = \sigma(Wh_t + b_o)$$

Back propagation through **time**

Non-RNN $O_t = \sigma(Wx_t + b)$

Back propagation through **layers**

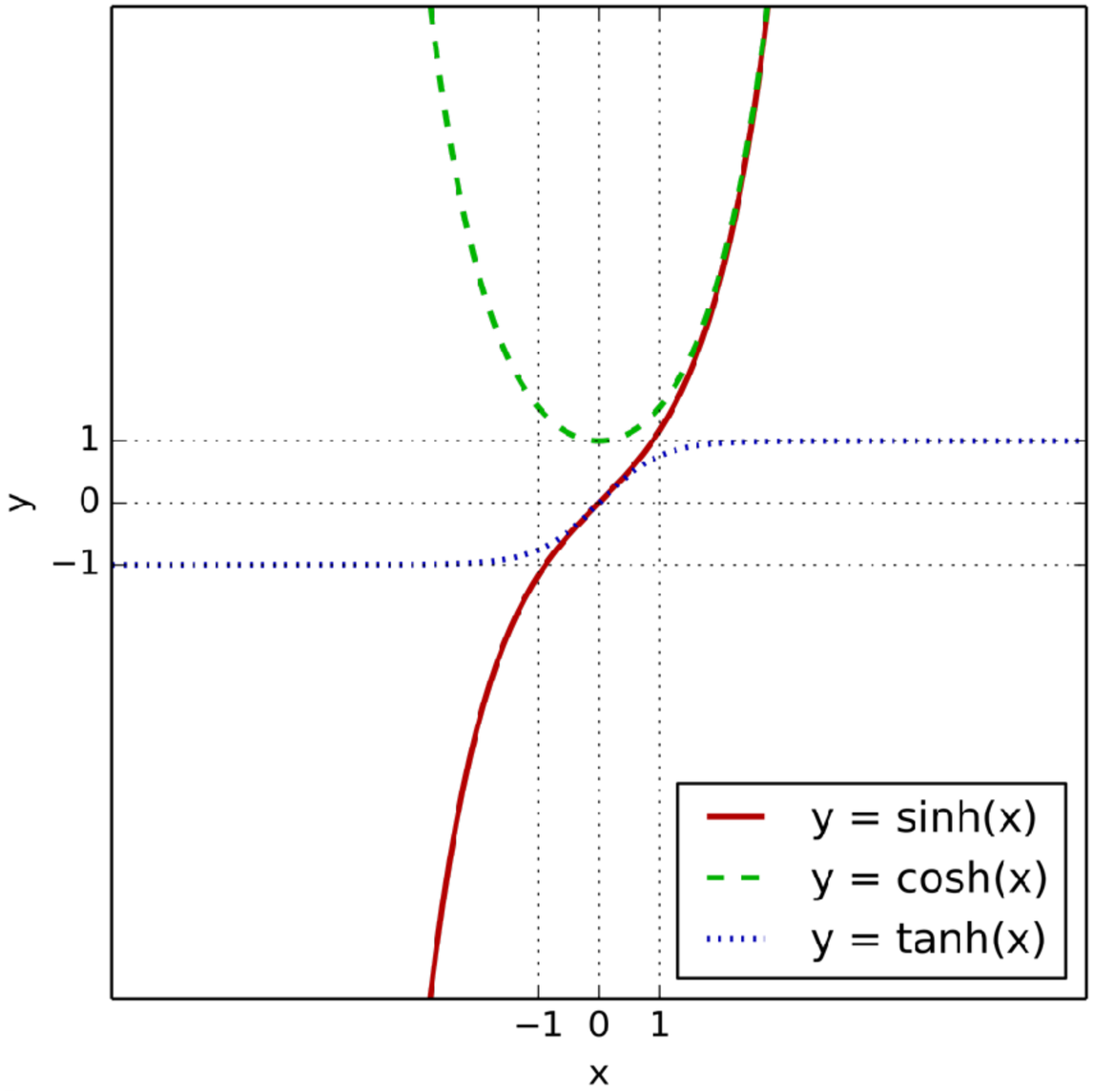
Anatomy of an RNN Cell



$$h^{<t>} = \tanh(Ux^{<t>} + Vh^{<t-1>} + b_h)$$

$$\hat{y}^{<t>} = \text{softmax}(Wh^{<t>} + b_y)$$

Hyperbolic Tangent as an Activation Function



$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

Between **[-1, 1]**, the function behaves **linearly** (more or less).

As long as the input range is within [-1,1], the derivative in this range is approximately 1.

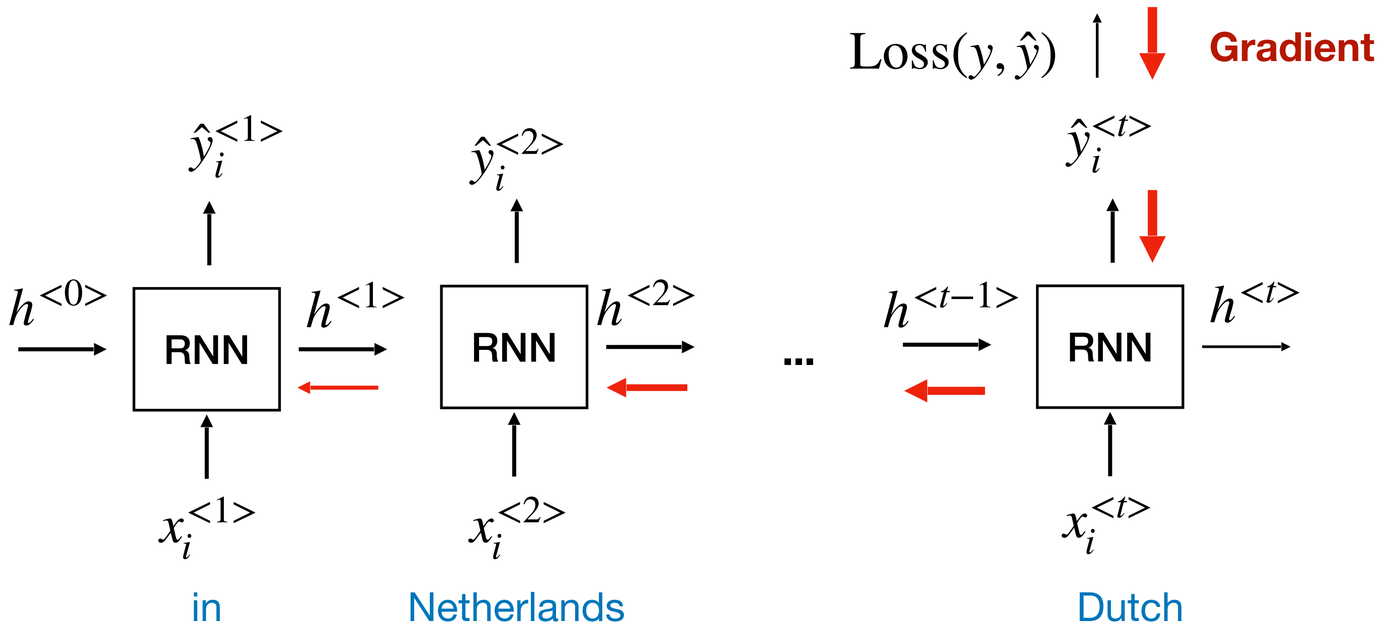
The function always scales the input to the well-controlled range of [-1,1].

Long-term Dependency

Sometimes, the contexts are far away.

The clouds are in the *sky*.

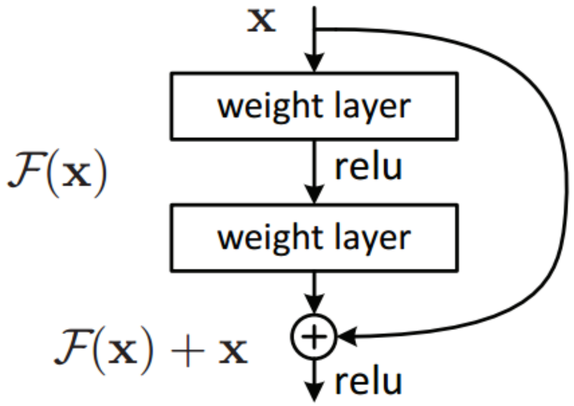
I grew up in the Netherlands… I speak fluent *Dutch*.



Contexts can be **forgotten**; the **gradients** are **vanishing** during back propagation.

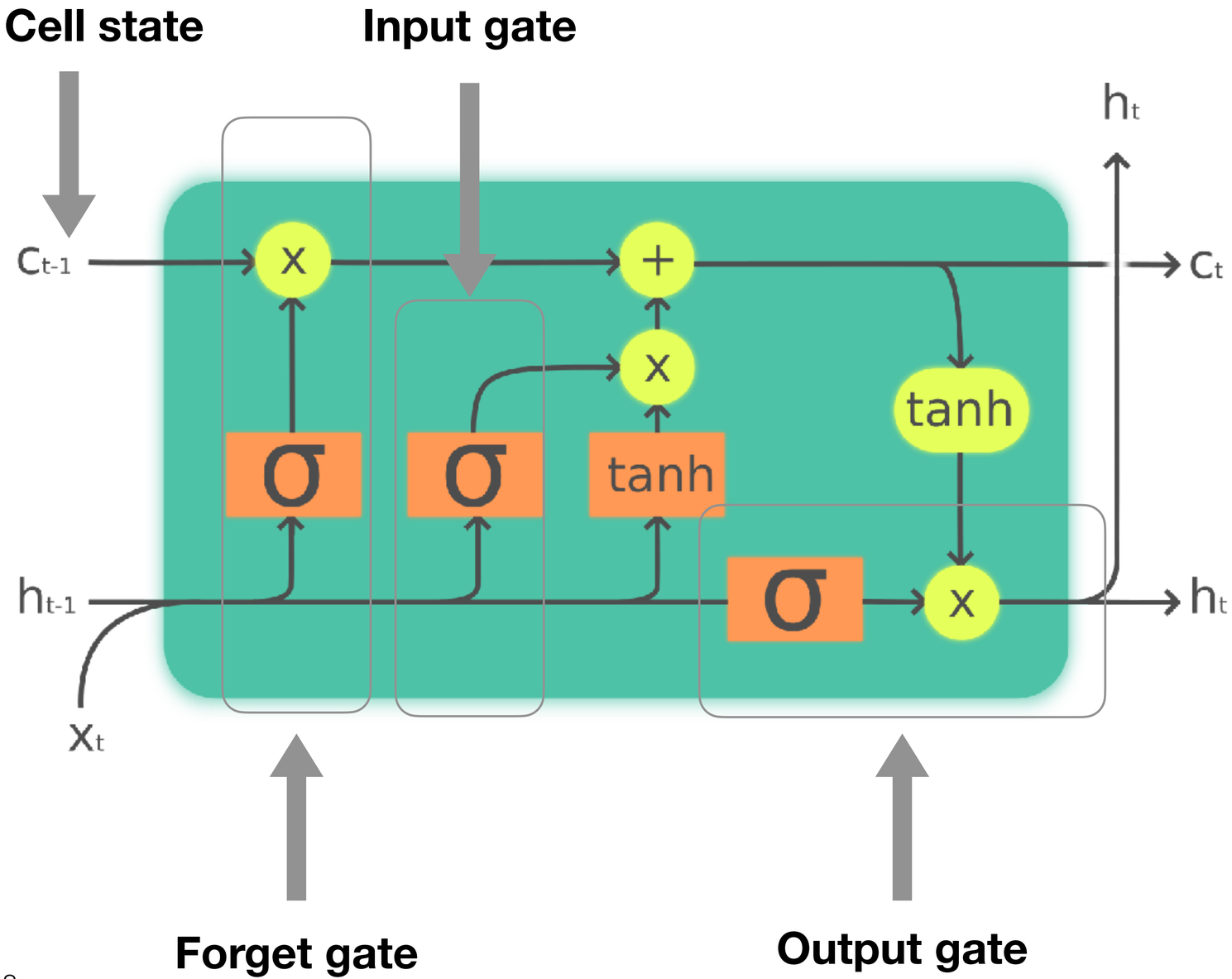
Vanishing Gradient Problem in RNNs

Residual connection is a general solution to the vanishing gradient problem.



However, for very long dependency (which is common in RNNs), this solution is not sufficient.
We should instead regulate which information to **keep** and which to **forget**.

Long Short-term Memory



Hochreiter & Schmidhuber (1997)

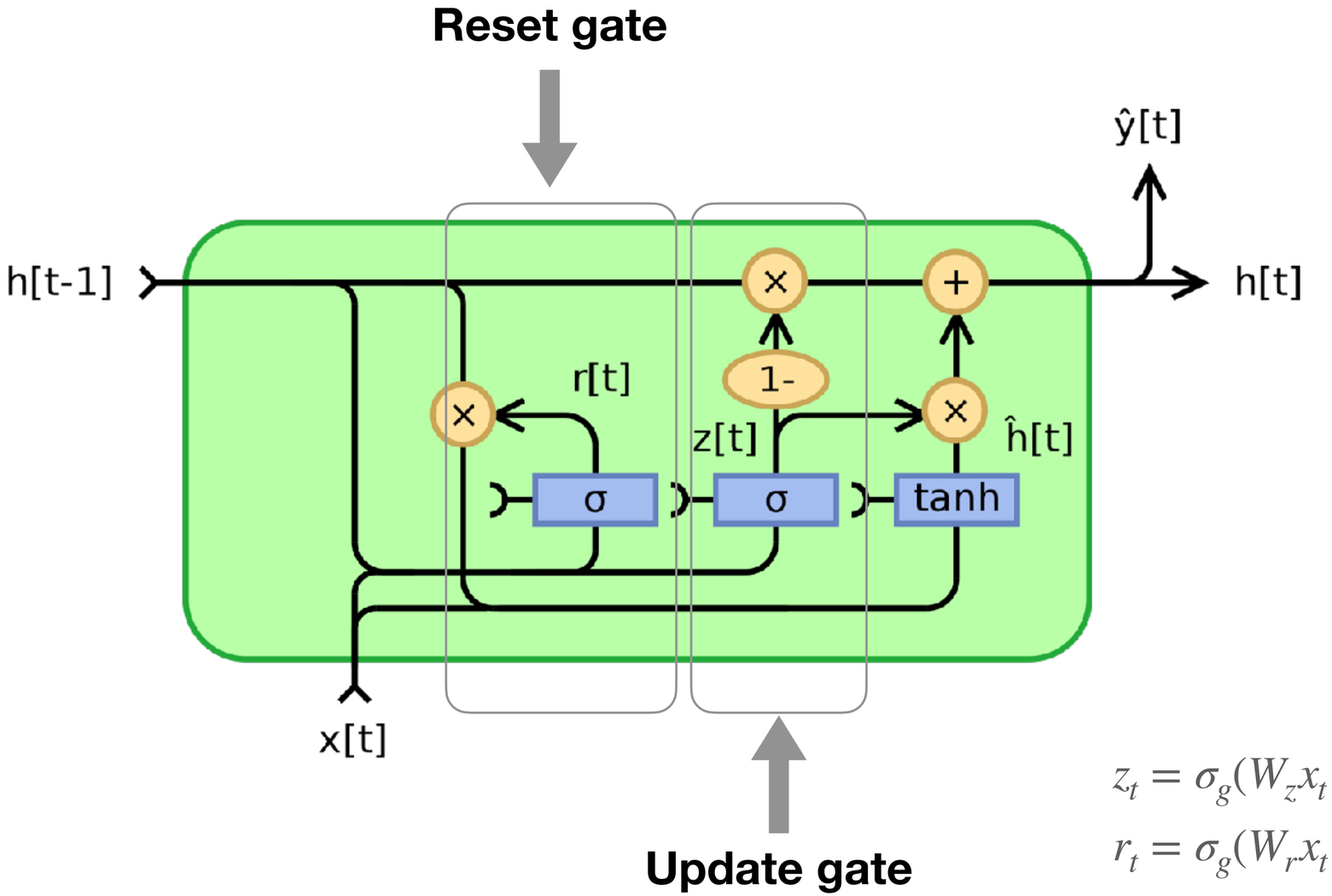
LSTM regulates the flow of information through gates.

- **Input gate**: controls the extent to which a new value flows into the cell;
- **Forget gate**: controls the extent a value remains in the cell;
- **Output gate**: controls the extent to which the value in the cell is used to calculate the output activation.

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f)$$
$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i)$$
$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o)$$
$$\tilde{c}_t = \sigma_h(W_c x_t + U_c h_{t-1} + b_c)$$
$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$$
$$h_t = o_t \circ \sigma_h(c_t)$$

Image source: Wikipedia

Gated Recurrent Unit



Cho et al. 2014 (arXiv:1406.1078)

GRU regulates the flow of information through gates:

- **Reset gate**: controls how much past information to forget;
- **Update gate**: controls what information to discard and what to keep (similar to the LSTM's input gate and forget gate combined)

$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z)$$
$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r)$$
$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \phi_h(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h)$$

Generating Input Sequences

To apply a long sequence to a RNN, we need to define the training data x and the labels y:

(1, ...) (2, ...) (3, ...) (4, ...) (5, ...) (6, ...) (7, ...) (8, ...) (9, ...) (10, ...) (11, ...) (12, ...) (13, ...) (14, ...)

We could either **reshape** it:

x (1, ...) (2, ...) (3, ...) (4, ...) (5, ...) (6, ...) (7, ...)

y (8, ...) (9, ...) (10, ...) (11, ...) (12, ...) (13, ...) (14, ...)

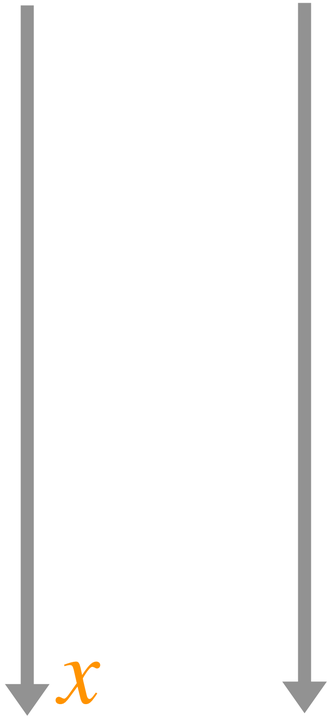
Or truncate it using a **rolling window**:

(1, ...) (2, ...) (3, ...) (4, ...) (5, ...) (6, ...) (7, ...)

(3, ...) (4, ...) (5, ...) (6, ...) (7, ...) (8, ...) (9, ...)

(5, ...) (6, ...) (7, ...) (8, ...) (9, ...) (10, ...) (11, ...)

(7, ...) (8, ...) (9, ...) (10, ...) (11, ...) (12, ...) (13, ...)



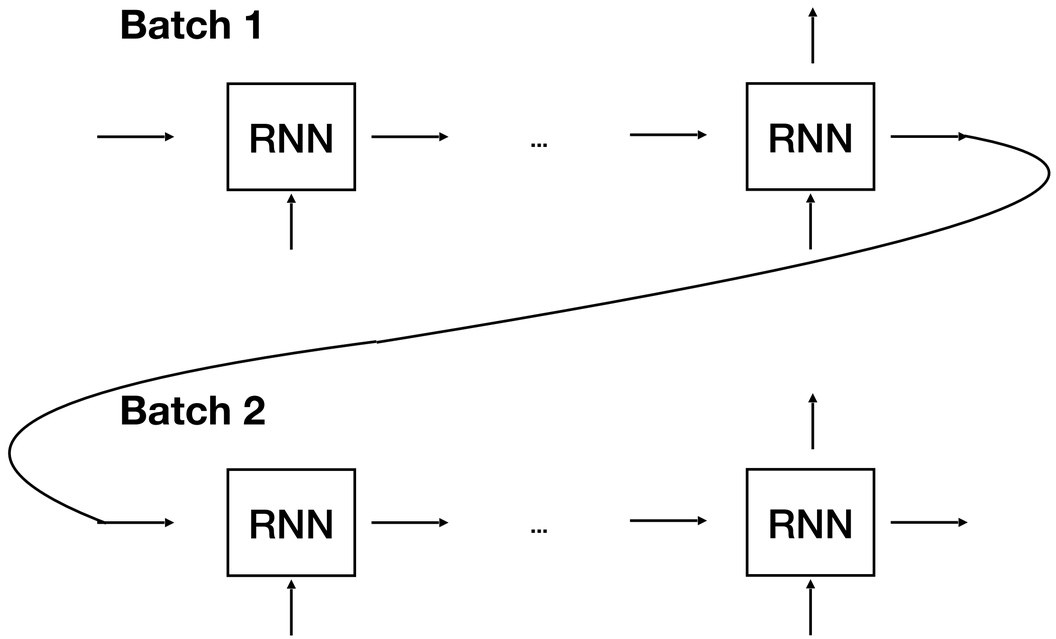
(13, ...)

Stateful RNNs

A stateful RNN passes the **last state** of the **previous batch** as the **initial state** of the **next batch**.

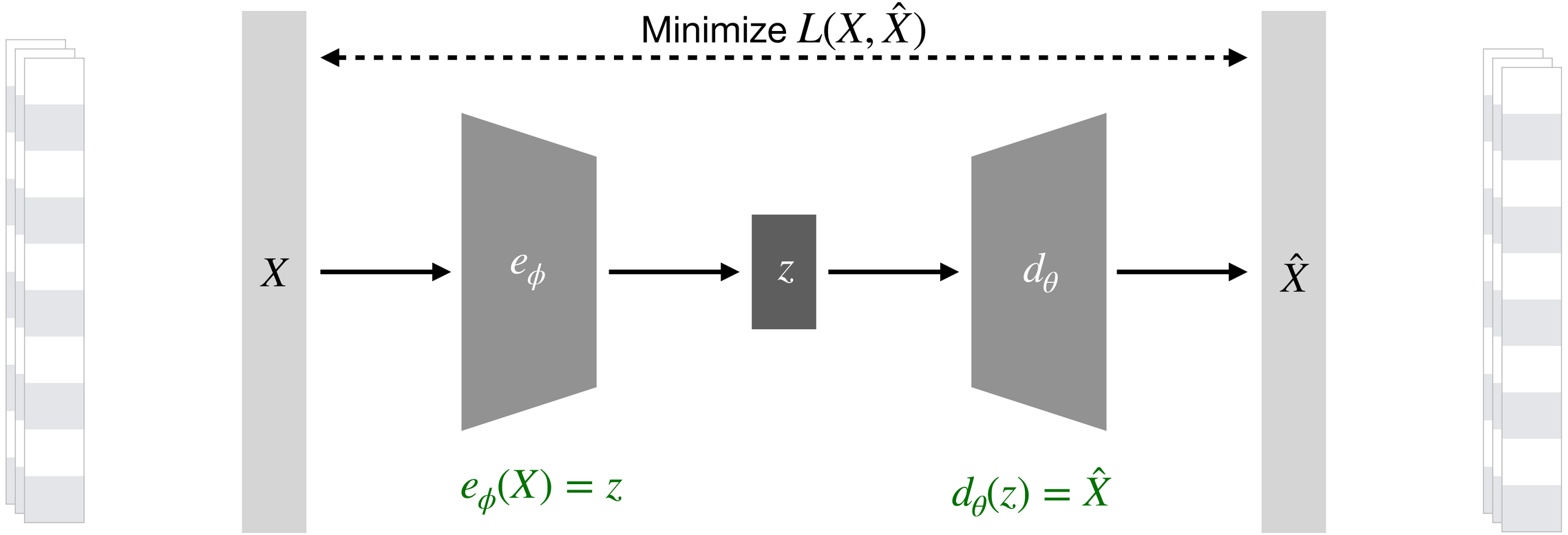
This is useful if there is some connections between batches. This feature essentially allows long sequences to be **folded** into batches.

A non-stateful RNN initialises state of each batch to zero.



Batch 1	(1, ...) (2, ...) (3, ...) (4, ...) (5, ...) (6, ...) (7, ...)
Batch 2	(8, ...) (9, ...) (10, ...) (11, ...) (12, ...) (13, ...) (14, ...)

LSTM autoencoder for rare event detection



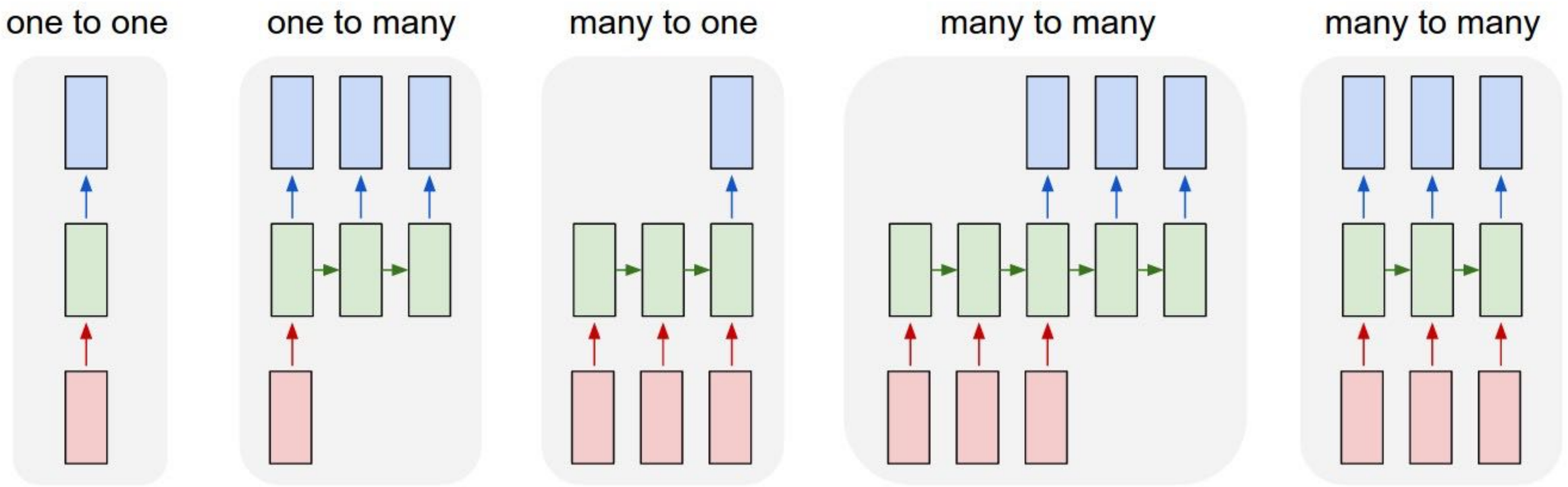
Input sequences

LSTM layers

LSTM layers

Reconstructed Input sequences

Topology of Sequence Processing Models

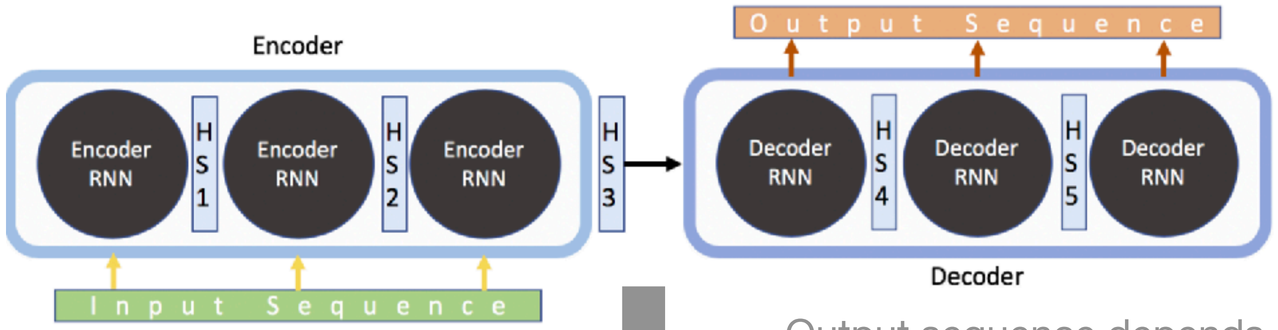


Non-RNN **RNN/Generation** **RNN/Classification** **RNN/Translation** **RNN/Captioning**

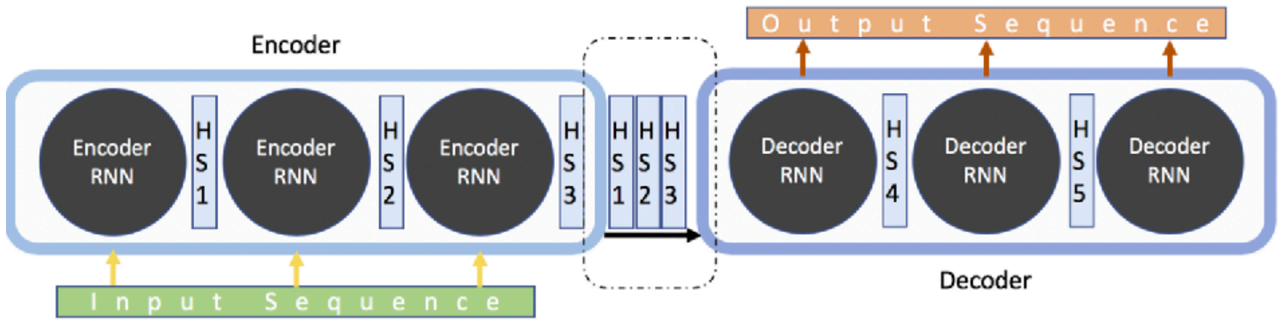
Image: Andrej Karpathy

Sequence-to-sequence Translation

… belongs to the “many-to-many” topology.



Output sequence depends **only** on HS3



Context vector

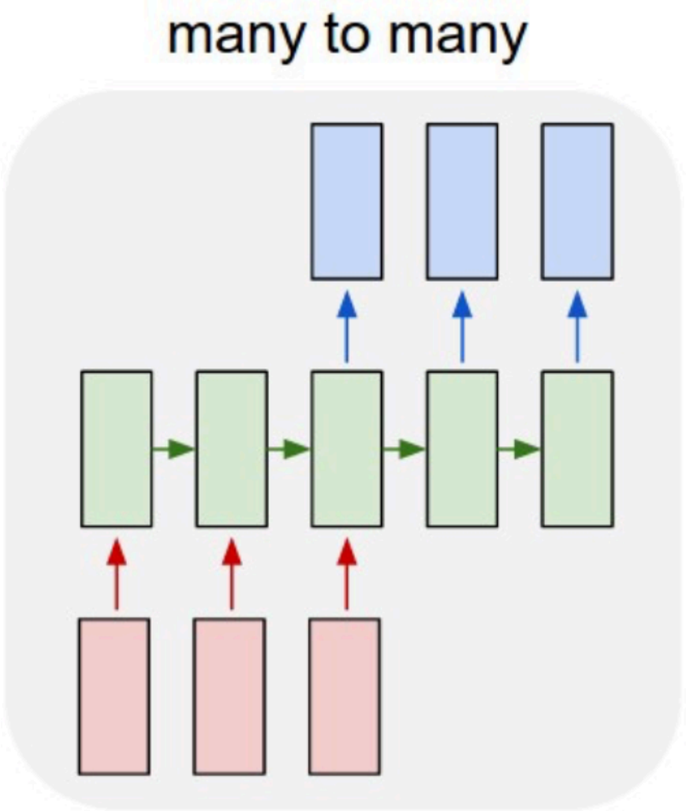
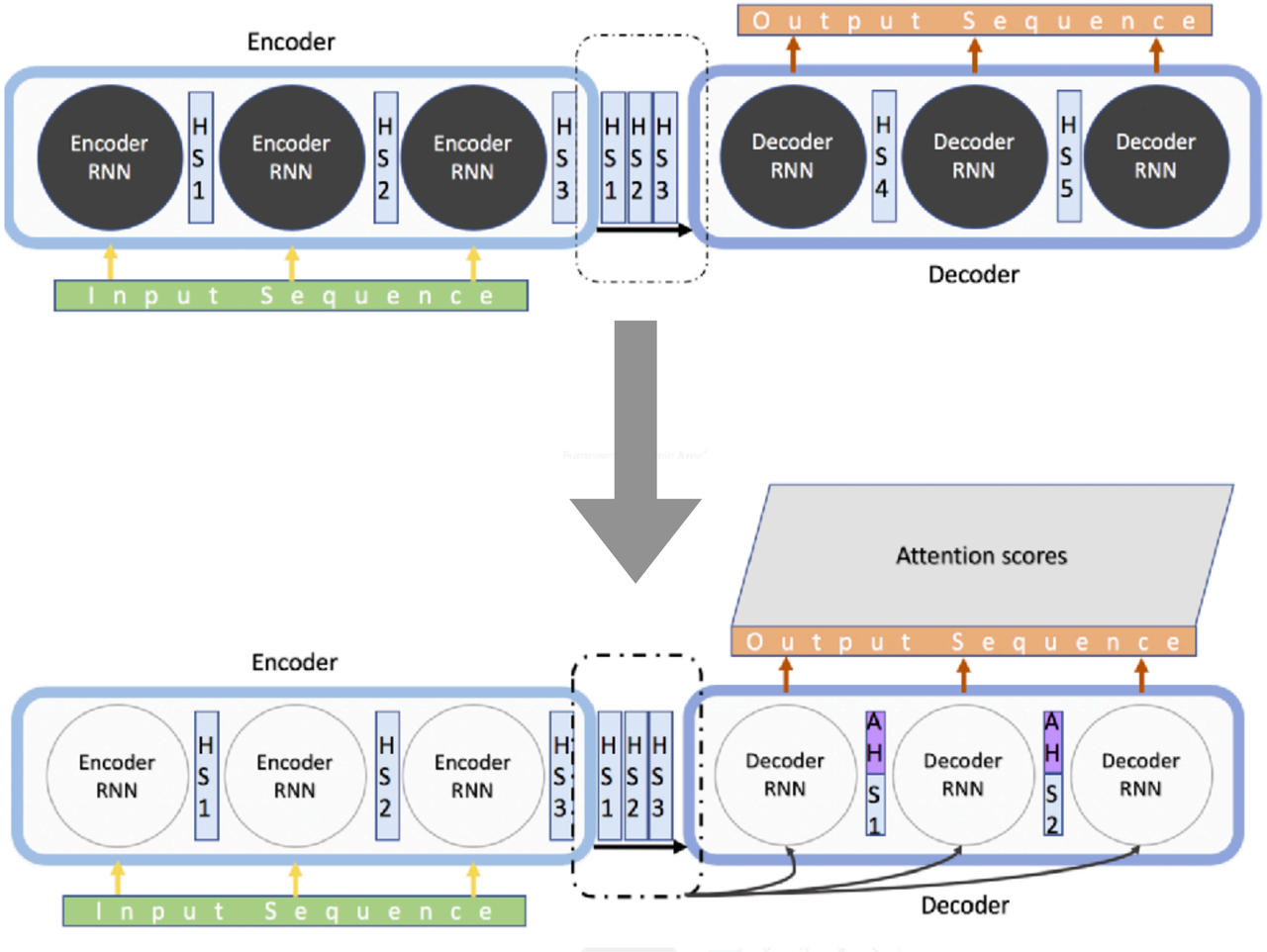


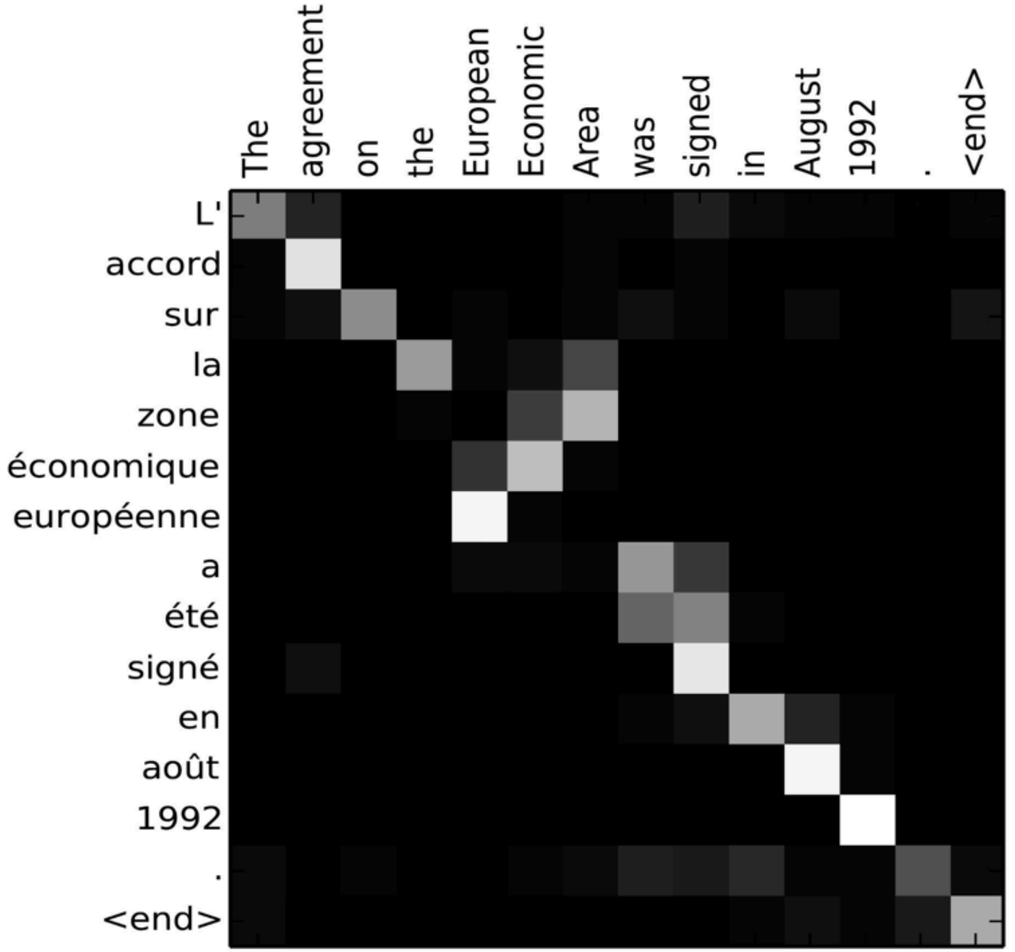
Image: Pranay Dugar

Seq2seq model: Sutskever et al., 2014, Cho et al., 2014, Vaswani et al. 2017)

Attention Mechanism

With **attention mechanism**, the decoder RNN **selectively** uses relevant data encoded in the **context vector** to generate the sequence.



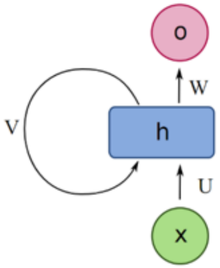


Bahdanau et al., 2015

Take Home Messages

Sequence modelling

… is a task that aims to solve the **dependency** between **tokens** in a sequence;
… requires passing information along the flow of the sequence;
… can be categorised into classification, regression, generation, and translation.



Recurrent neural networks (RNNs)

… are NN architectures that allow weights and states to be shared across multiple time steps;
… backward propagate through **time**;
… are usually slow to train because the time step dependency cannot be parallelised.

Miscellaneous

- Long short-term memory (LSTM) and gated recurrent units (GRU) are RNNs that designed to mitigate the **vanishing gradient problem**.
- LSTM and GRU use **gates** to regulate the flow of information.
- The **encoder-decoder** architecture is particular useful in RNNs for sequence to sequence translation.