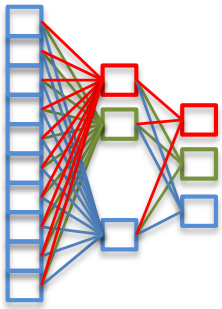
AI and Data Analytics

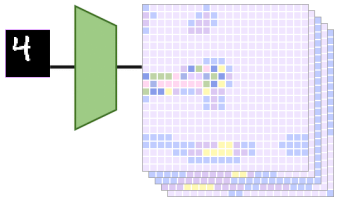
4.2 Recurrent Neural Networks

Christian Wolf

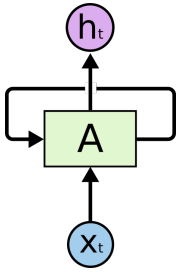
The Deep Toolbox



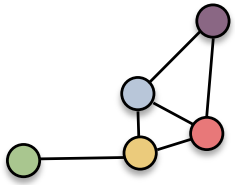
MLP



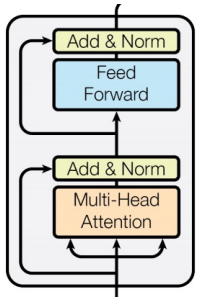
CNN / Convolutions



RNN / Recurrence



GN, GCN / Graphs, geometry

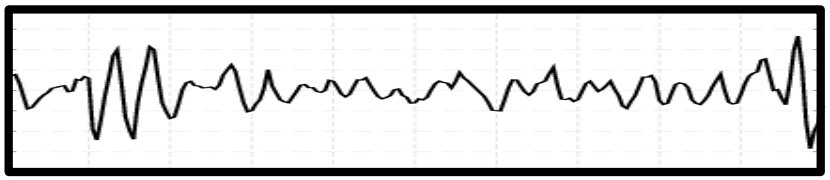


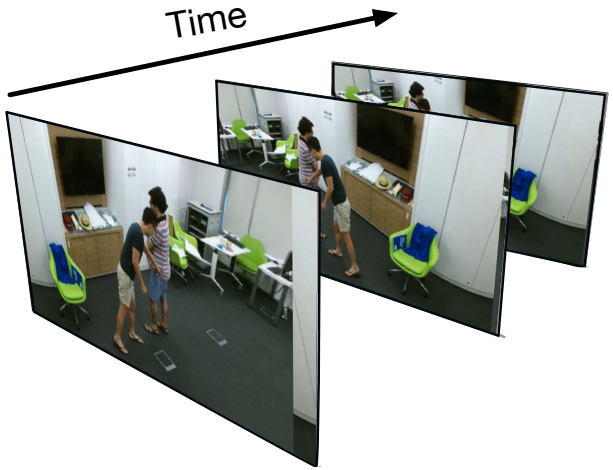
Transformers / Self-attention

What do I know about the data and the task?

Nothing (vector space)	*Translation equivariance*	*Sequential data, Markov property*	*Graph structured data*	*Permutation equivariance*

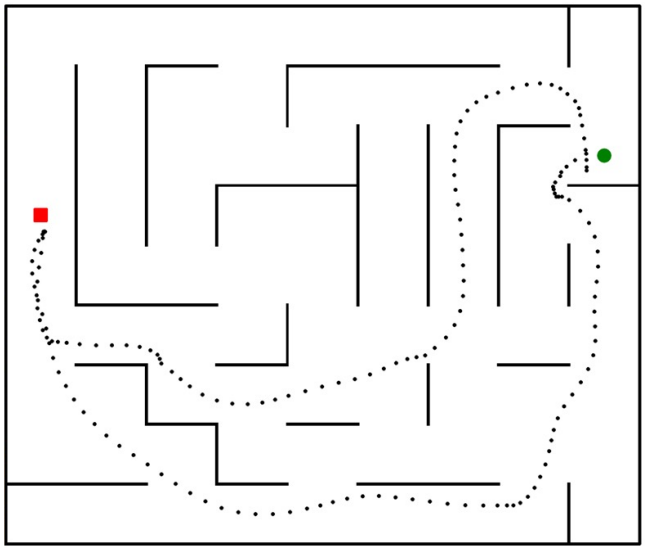
Problem: dealing with sequences



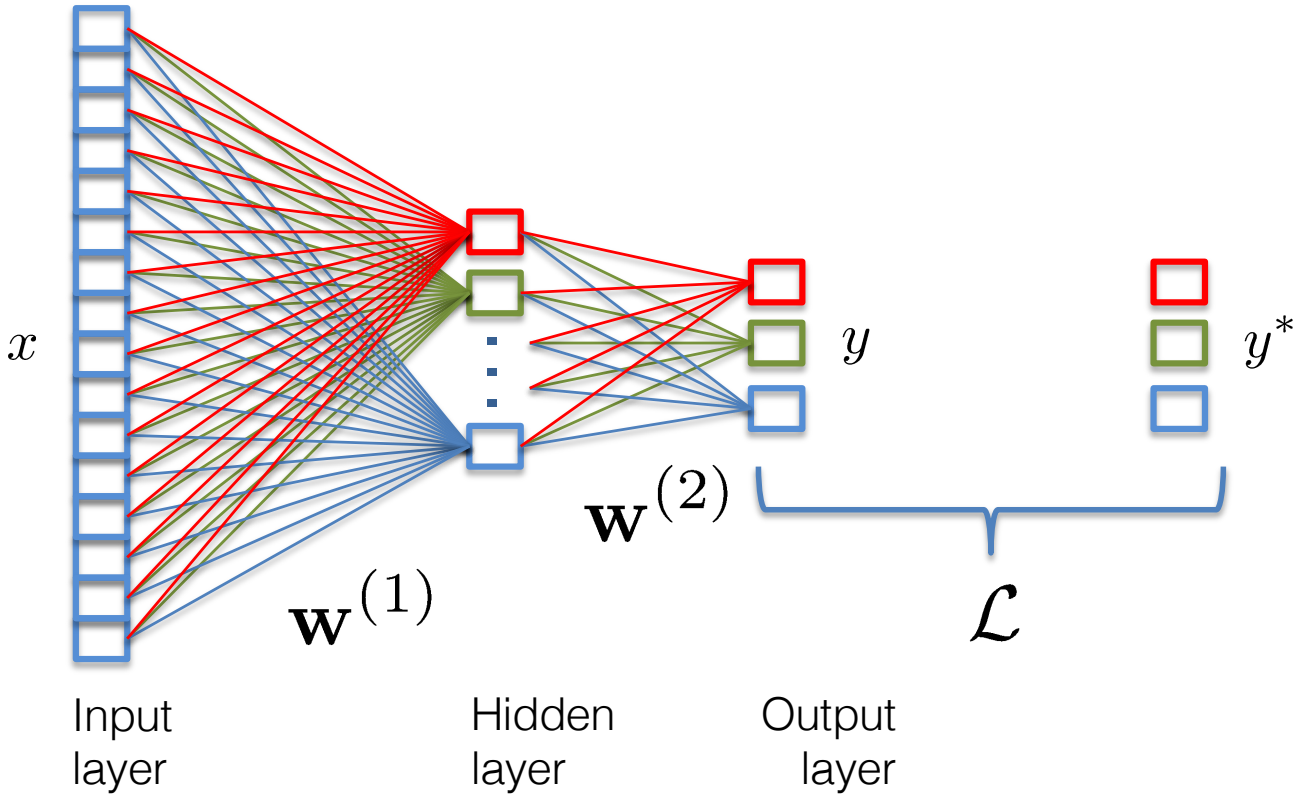


Toutes les familles heureuses se ressemblent. Chaque famille malheureuse, au contraire, l'est à sa façon.

Happy families are all alike. Every unhappy family is unhappy in its own way.

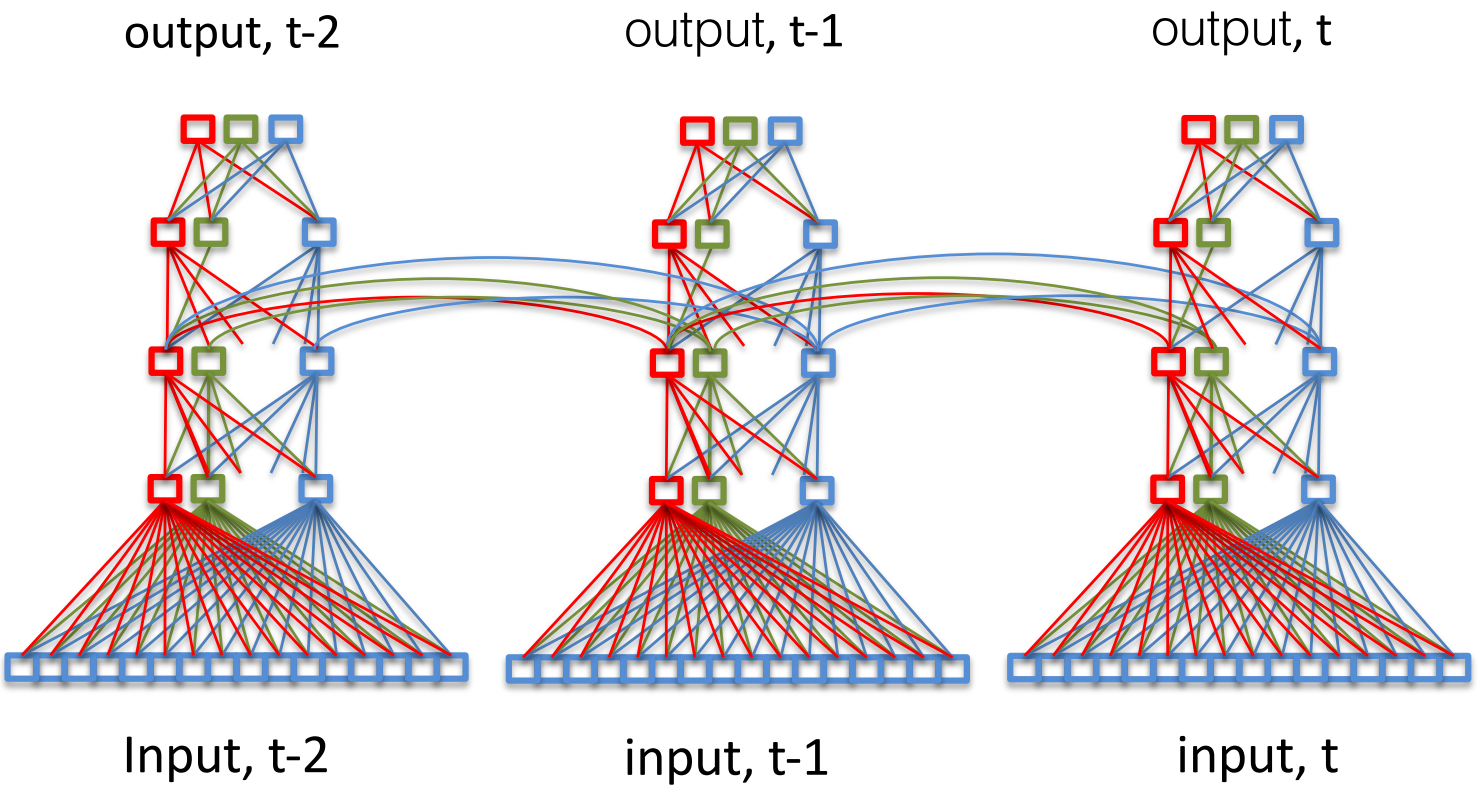


Feed Forward Networks



Prediction: **feed-forward** computation in a DAG (directed asyclic graph).

Recurrent neural networks



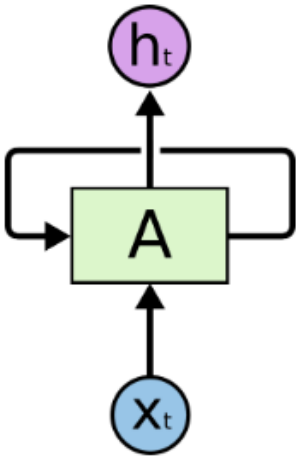
A shout-out ...

... to Chris Olah's excellent blog post on RNN and LSTM networks, which completely dominates lectures on this topic. The following couple of slides are based on his excellent drawings.

`Chris Olah, https://colah.github.io/posts/2015-08-Understanding-LSTMs/`

Recurrent Neural Networks (RNNs)

As feed-forward networks, Recurrent Neural Networks (RNNs) predict some output from a given input.
However, they also pass information over time, from instant $(t-1)$ to (t):

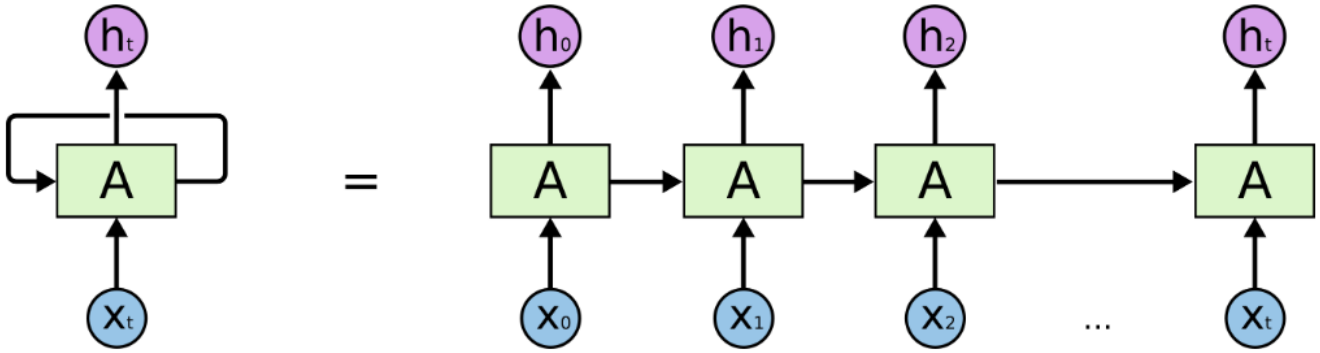


Here, we write h_t for the output, since these networks can be stacked into multiple layers, i.e. h_t is input into a new layer.

`Chris Olah, https://colah.github.io/posts/2015-08-Understanding-LSTMs/`

Recurrent Neural Networks (RNNs)

A more intuitive view of an RNN is to unroll it over time:



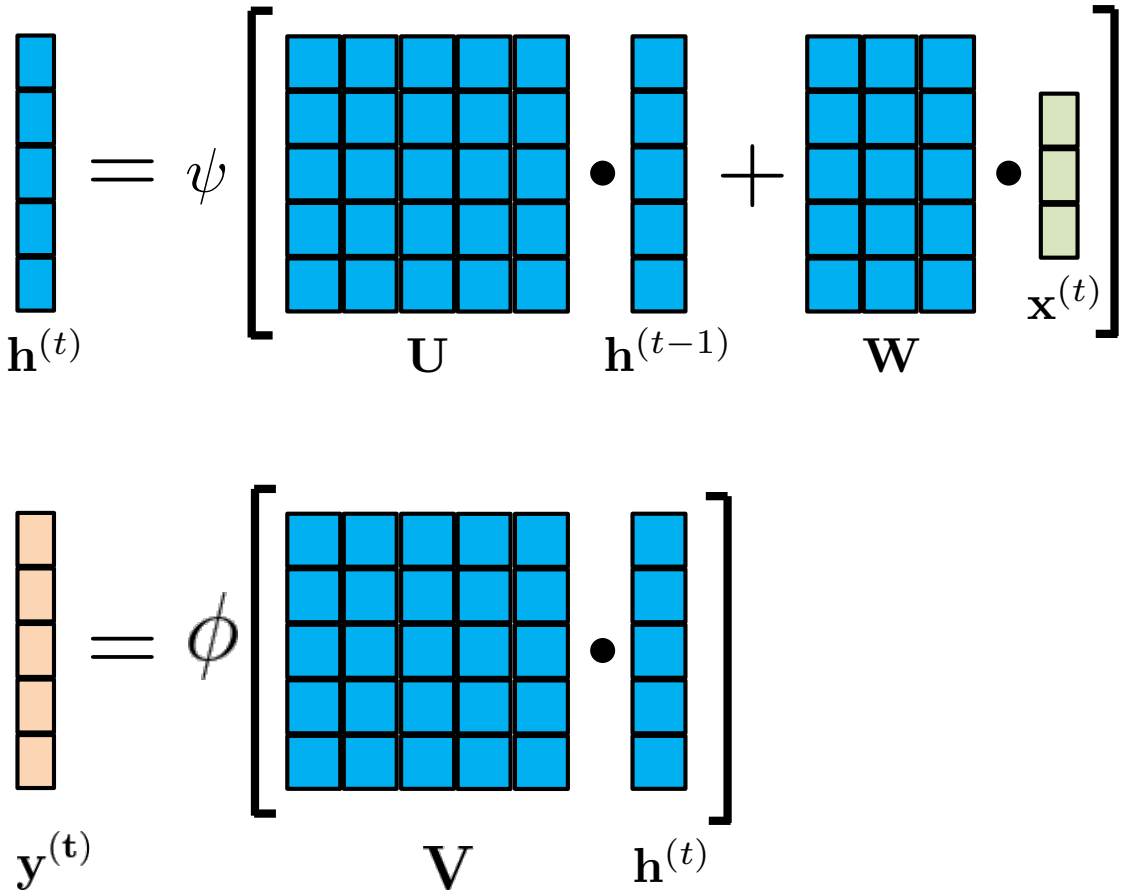
The update equations are:

$$h_t = \sigma\left(h_{t-1} \cdot W_h + x_t \cdot W_x\right)$$

$\Rightarrow$ two weight matrices: one for the recurrent connections over time, one for the input connections.
σ is an activation function.

Chris Olah, https://colah.github.io/posts/2015-08-Understanding-LSTMs/

From state h to output y



Toy example with handcrafted parameters

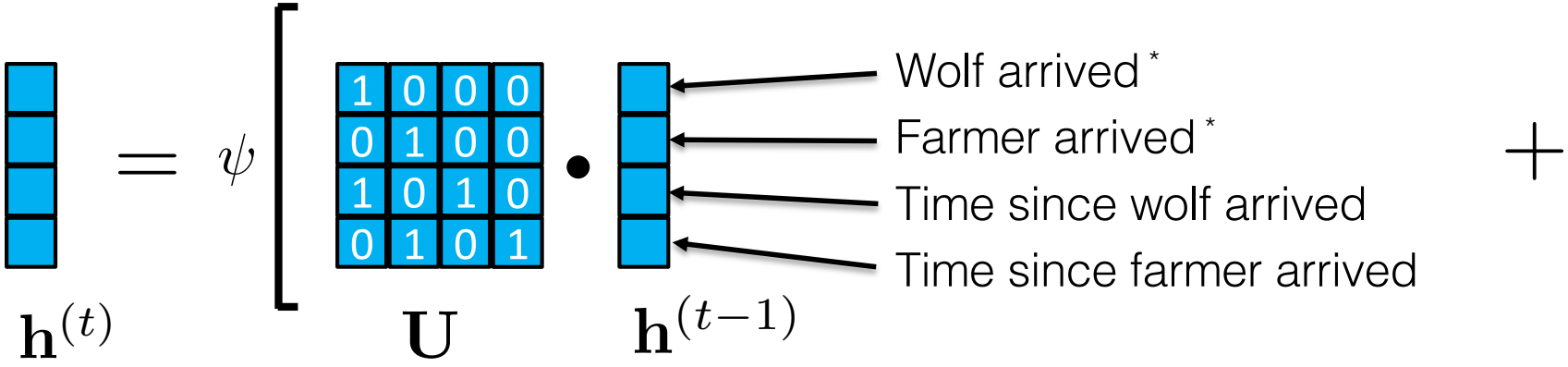
In a sequential problem, we surveil a farm and watch for the appearance of objects. At each instant t we oberve a vector 🎚 which can indicate the apparence of

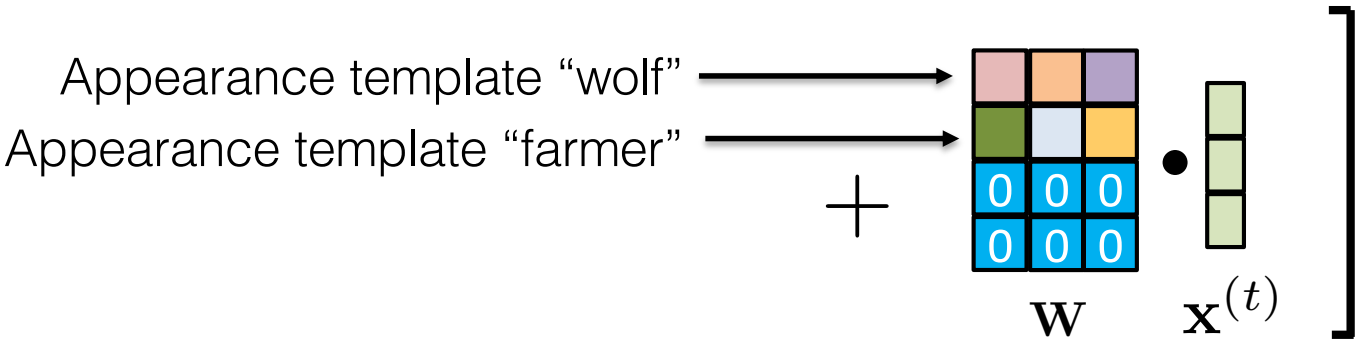
- a wolf 🐺
- a farmer 👩‍🌾

The objective is to output an estimate of danger, i.e.

- presence of the wolf w/o the farmer, or
- presence of both, arrival of the wolf before the farmer.

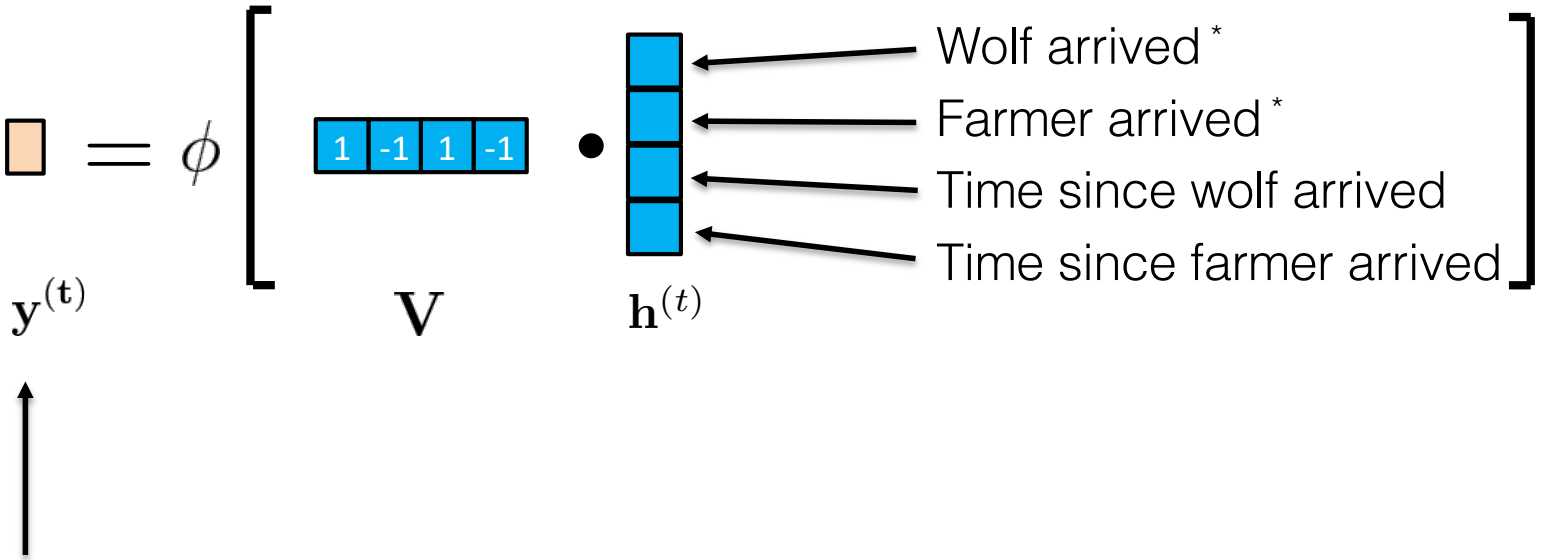
Toy example with handcrafted parameters





*Appropriate activation functions required to normalize state values

Toy example with handcrafted parameters



Output: danger!

RNN training & problems

RNNs are trained with backpropagation through time (BPTT): the graph is unrolled, and the loss derived w.r.t. the parameters of all different time instants.
Standard vanilla RNNs are difficult to train and suffer from shortcomings:

- Vanishing gradients: small gradients vanish over long time ranges.
- Exploding gradients: high gradients explode over long ranges.
- Lack of long-term dependency handling: short-term updates between individual time instances dominate.

Solution: gating mechanisms (LSTM and GRU networks).

Gating

Problem: hidden state is updated at **each** time step

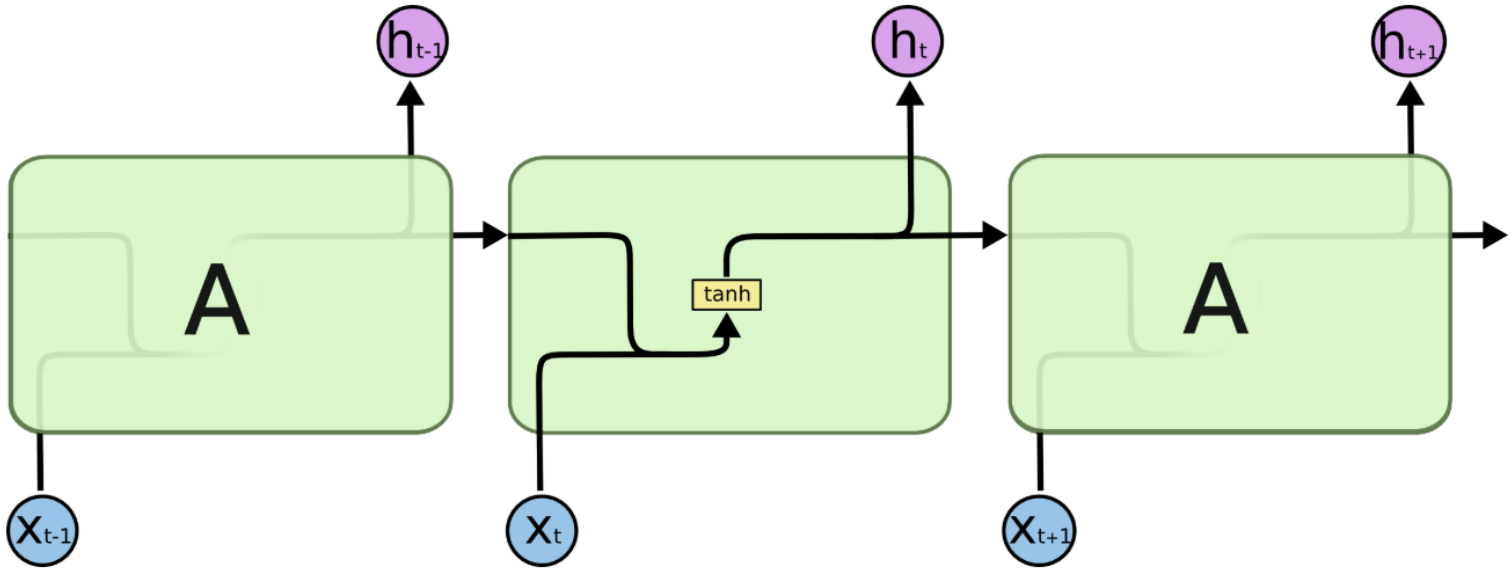
Solution: at each instant, for each state value, decide how much to

- input
- forget
- output

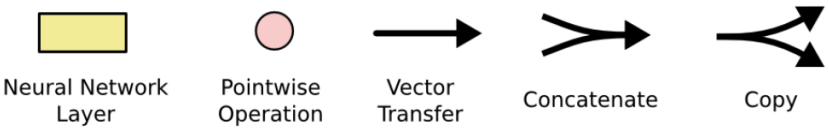
- These decisions are also learned

LSTM Networks: walk through

We start by illustrating a vanilla RNN in a new way:



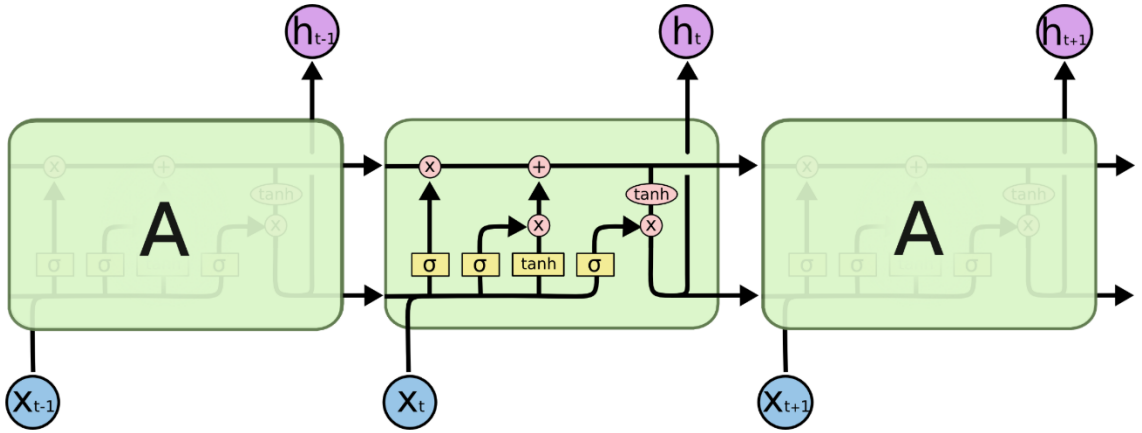
where:



Chris Olah, https://colah.github.io/posts/2015-08-Understanding-LSTMs/

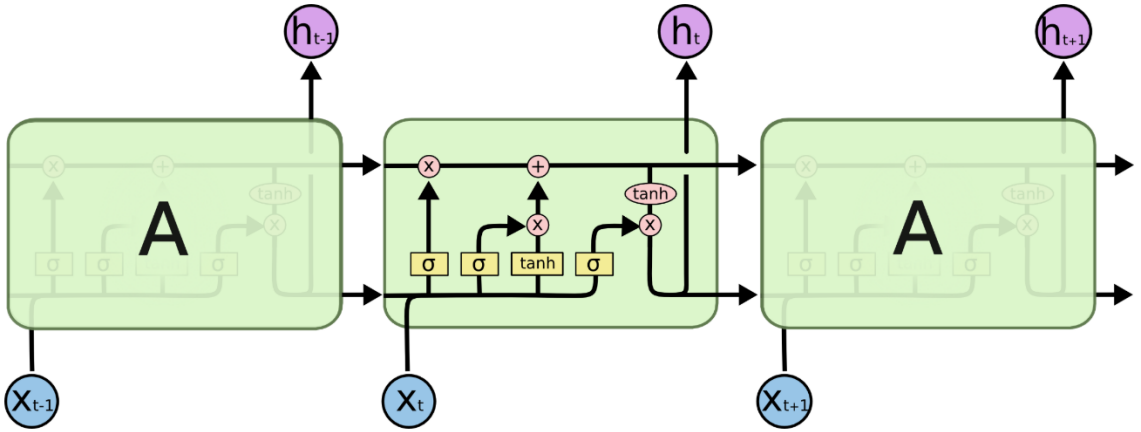
LSTM Networks

LSTM (=Long-short term Memory) networks use gating mechanisms, which handle information flow in a fully trainable way.



LSTM Networks

LSTM (=Long-short term Memory) networks use gating mechanisms, which handle information flow in a fully trainable way.



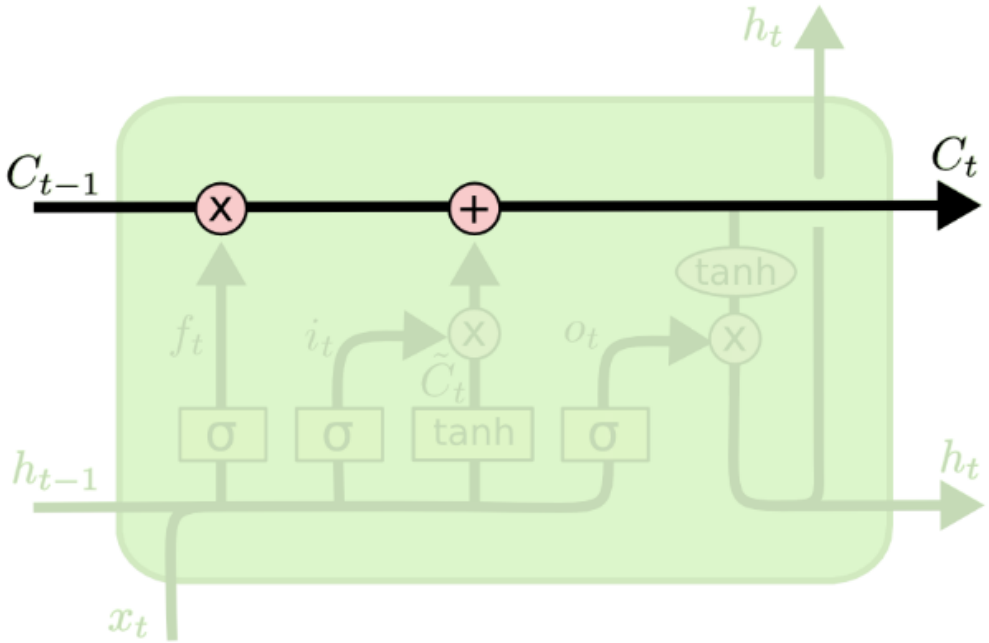
$$
\begin{aligned}
f_t &= \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \\
i_t &= \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \\
\tilde{C}_t &= \tanh\left(W_C \cdot [h_{t-1}, x_t] + b_C\right) \\
C_t &= f_t * C_{t-1} + i_t * \tilde{C}_t \\
o_t &= \sigma\left(W_o \left[h_{t-1}, x_t\right] + b_o\right) \\
h_t &= o_t * \tanh\left(C_t\right)
\end{aligned}
$$

Chris Olah, https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM Networks: walk through

An LSTM has two different memory representations:

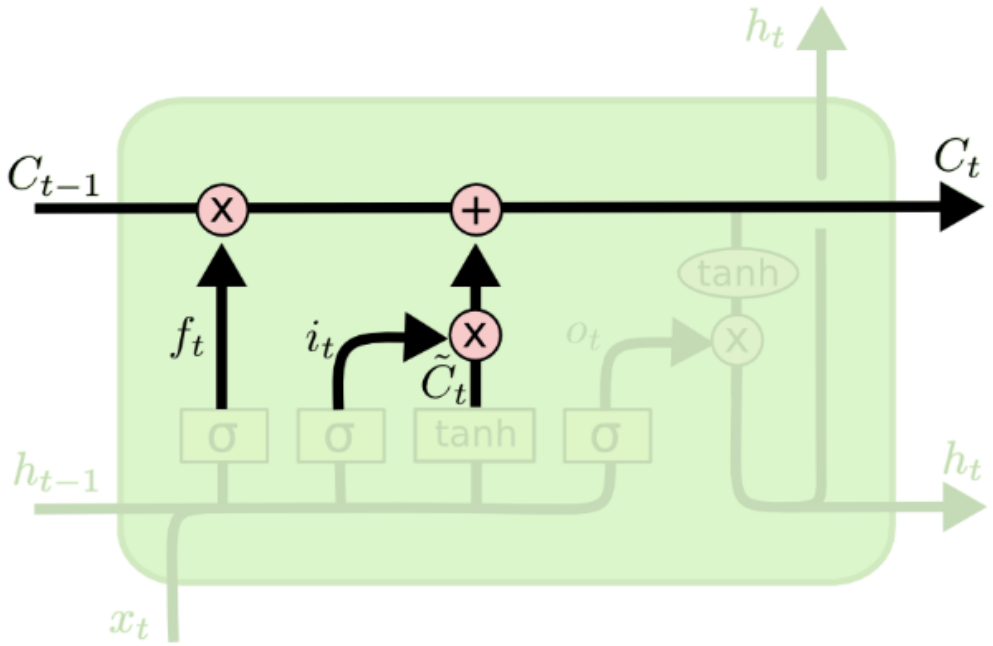
- A cell state C,
- the hidden state h.



Chris Olah, https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM Networks: walk through

The new cell state C_t is a linear combination of the old cell state C_{t-1} and some new updated information $\tilde{C}_t$ (described later):



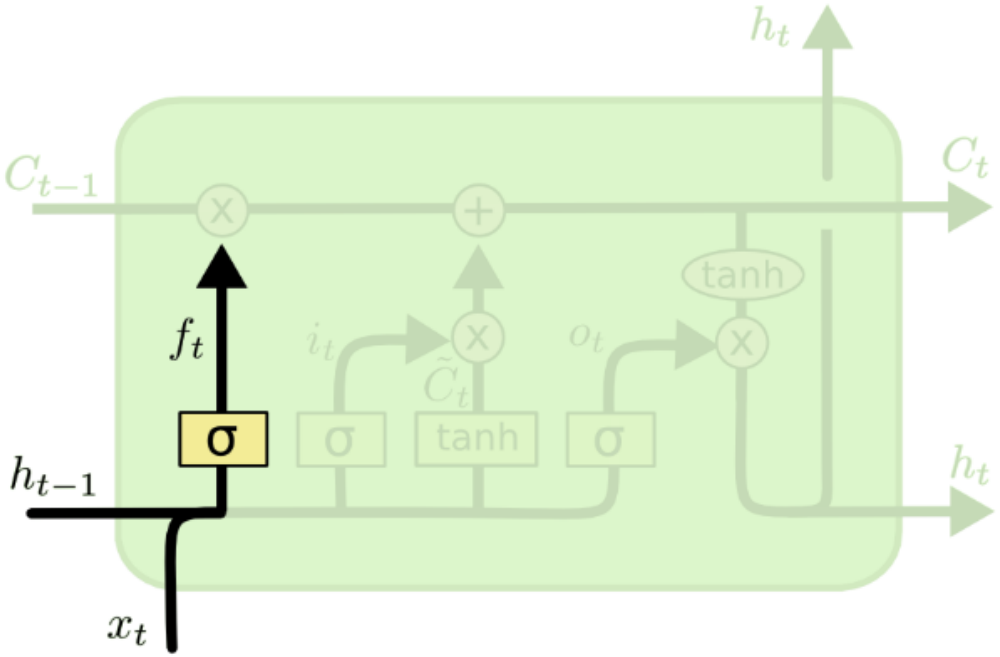
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

f_t and i_t are "**gates**" (trainable functions), which govern the information flow (based on the hidden state h_t).

Chris Olah, https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM Networks: walk through

The forget gate controls how much of the old cell state is forgotten or passed through:



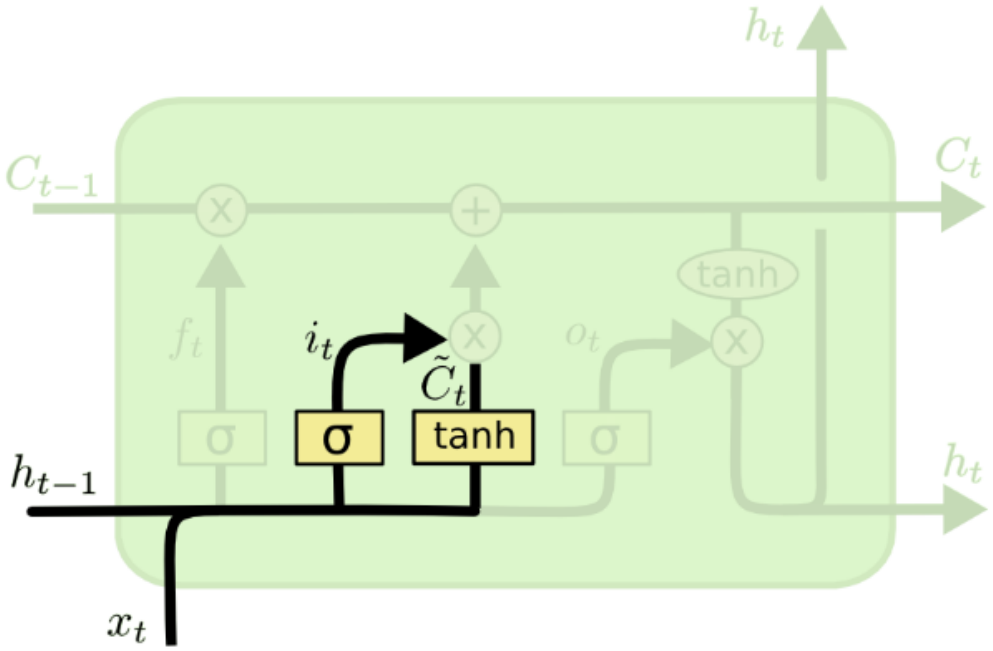
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$

Chris Olah, https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM Networks: walk through

The input gate controls how much new information is passed over to the cell state.

The new information is predicted from the hidden state h:

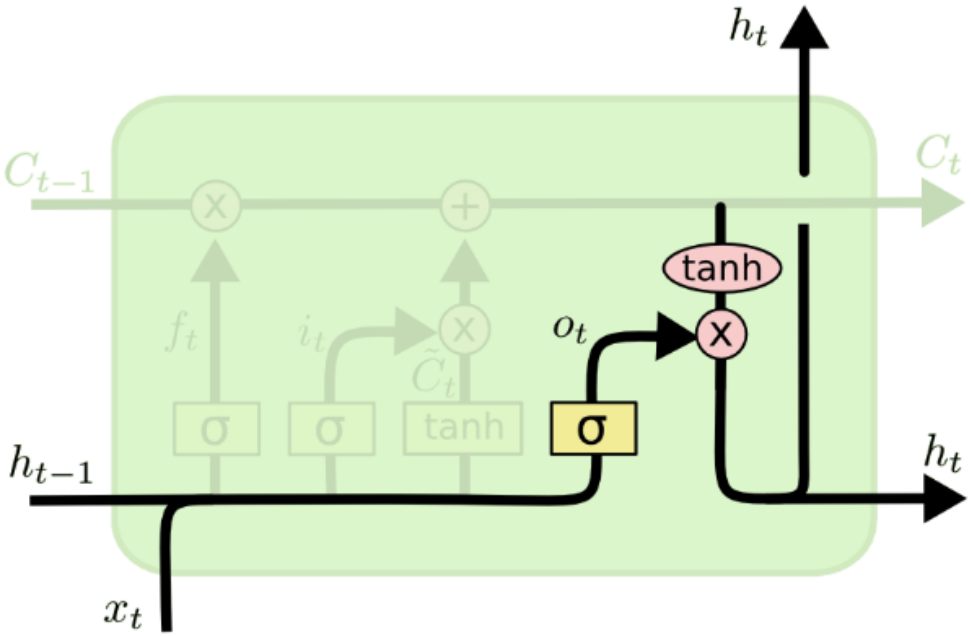


$$
\begin{aligned}
i_t &= \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \\
\tilde{C}_t &= \tanh\left(W_C \cdot [h_{t-1}, x_t] + b_C\right)
\end{aligned}
$$

Chris Olah, https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM Networks: walk through

The output gate controls, how the information from the cell state is translated into the hidden state h:

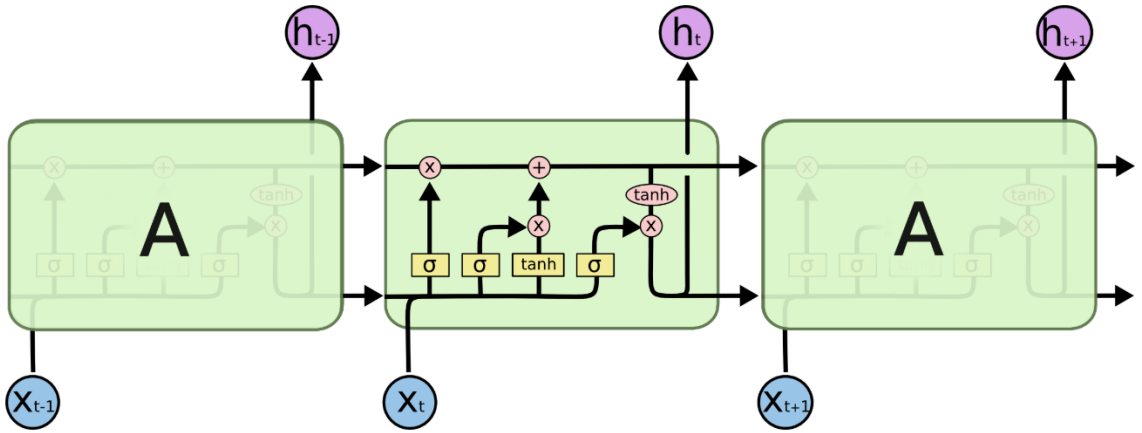


$$
\begin{aligned}
o_t &= \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right) \\
h_t &= o_t * \tanh\left(C_t\right)
\end{aligned}
$$

Chris Olah, https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM Networks

The full model, again:

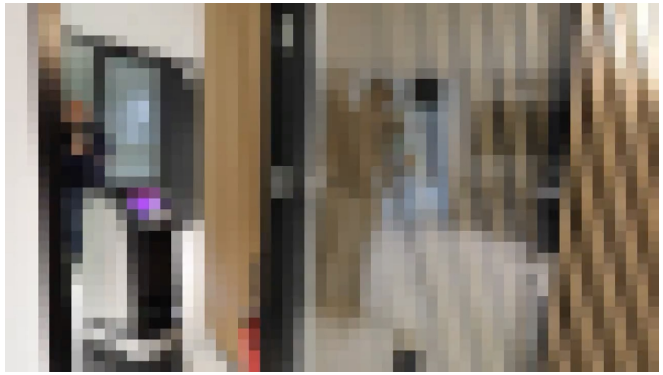


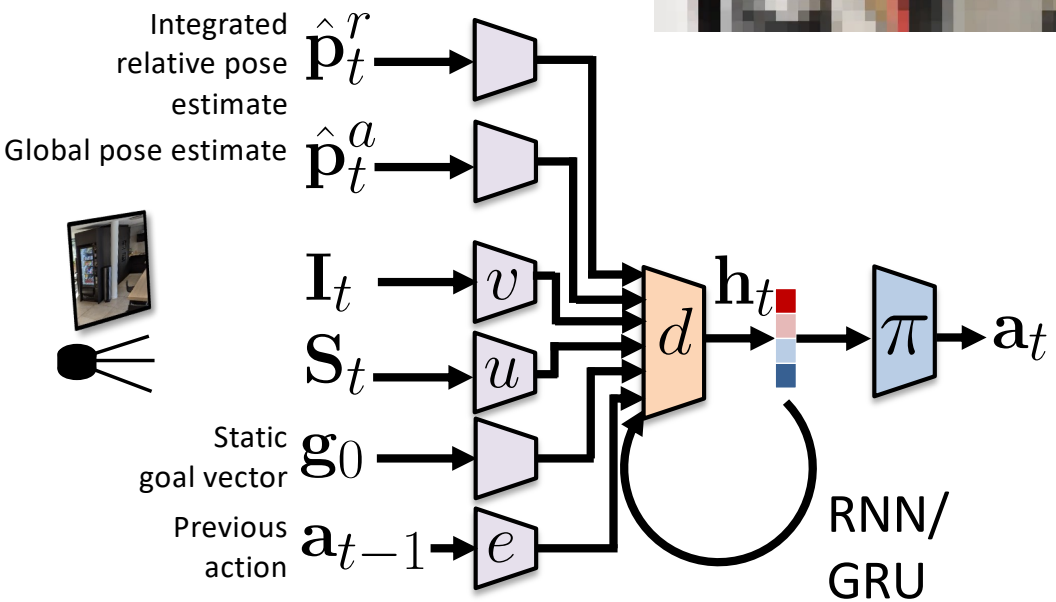
$$
\begin{aligned}
f_t &= \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \\
i_t &= \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \\
\tilde{C}_t &= \tanh\left(W_C \cdot [h_{t-1}, x_t] + b_C\right) \\
C_t &= f_t * C_{t-1} + i_t * \tilde{C}_t \\
o_t &= \sigma\left(W_o \left[h_{t-1}, x_t\right] + b_o\right) \\
h_t &= o_t * \tanh\left(C_t\right)
\end{aligned}
$$

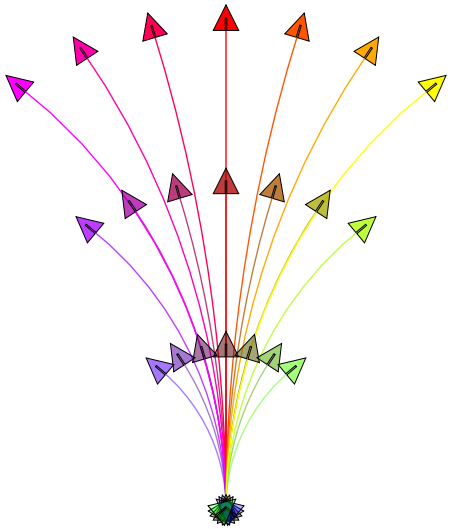
Chris Olah, https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Example applictions: Robotics

Navigation with recurrent memory







Input: RGB, Lidar, pose estimates from onboard sensors.
The agent does **not** have access to a map.

28 discrete actions
Pairs of linear+angular velocities