Recurrent Neural Networks

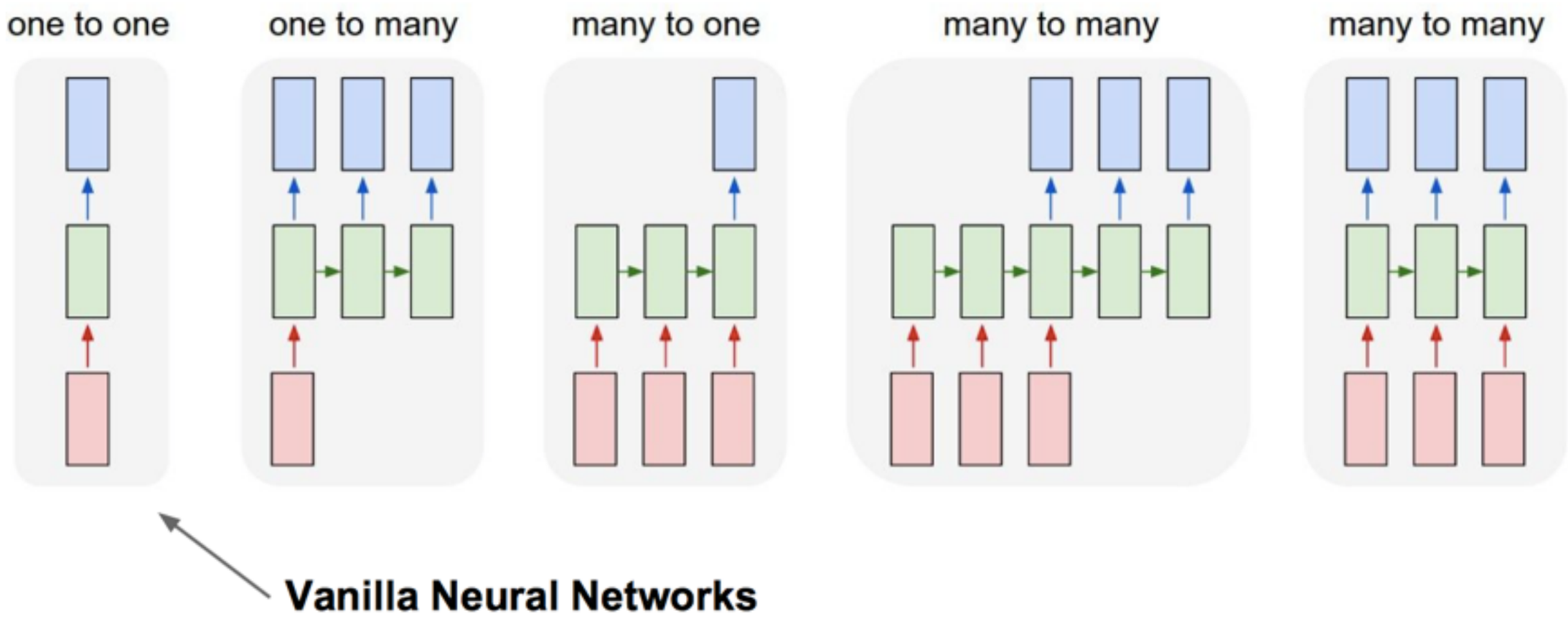
Recitation 3
14/10/2016

Limitations of CNNs

- Fixed input length
- Adapt to time series data
- Convolution corresponds to strong prior
- Requires a lot of training data

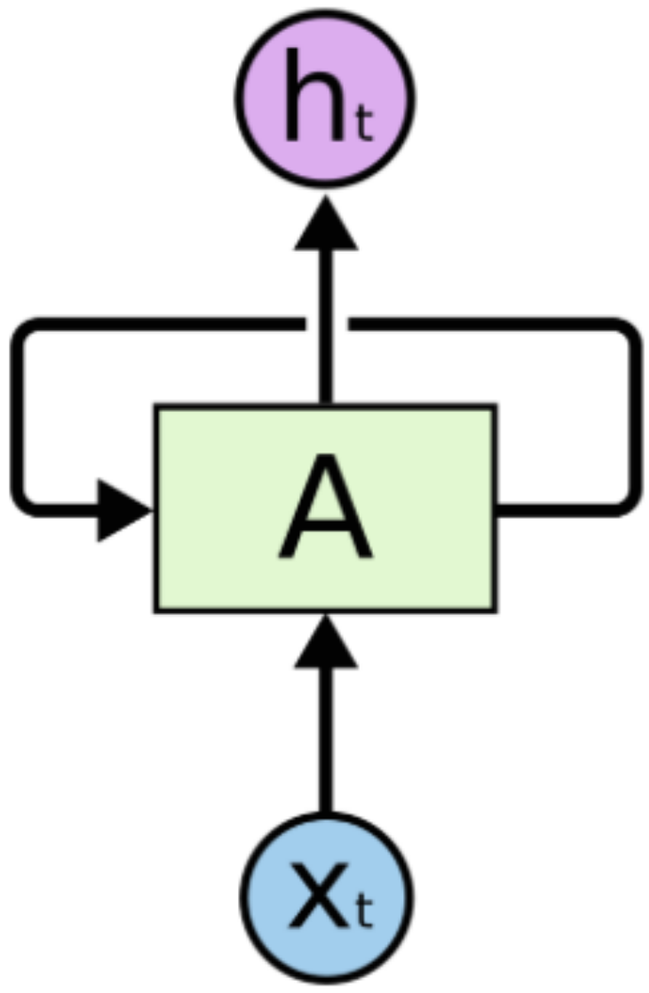
Recurrent Neural Network

- Allows a lot of flexibility:



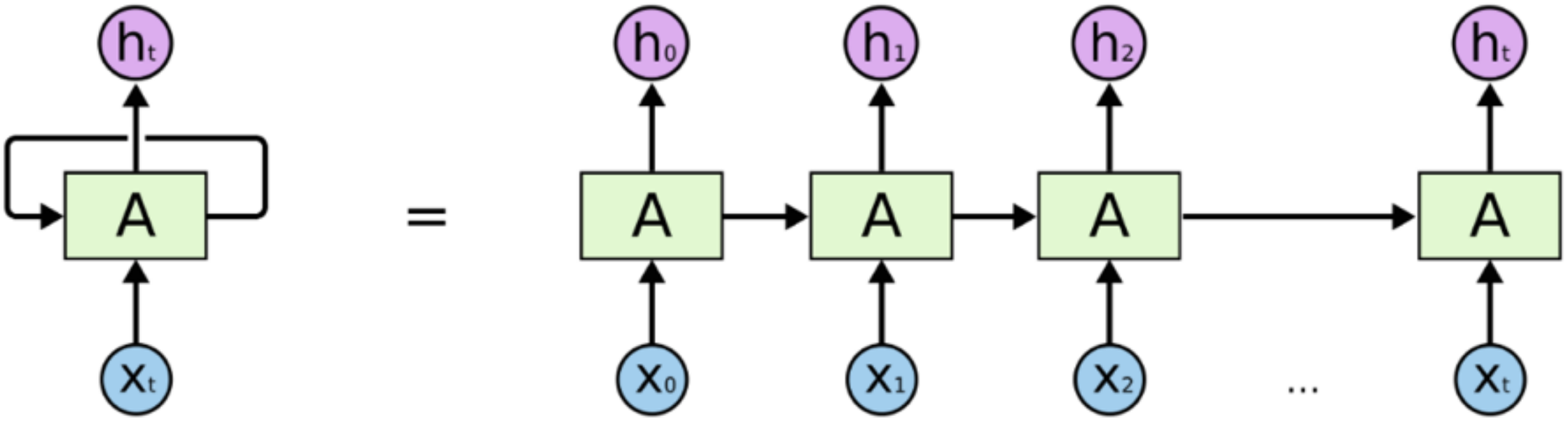
RNNs

- NNs assume that all inputs are independent of each other
- Want to make use of sequential information
- Recurrent because perform the same task on each sequence element
- Loops allow information to persist
- Output depends on previous computation



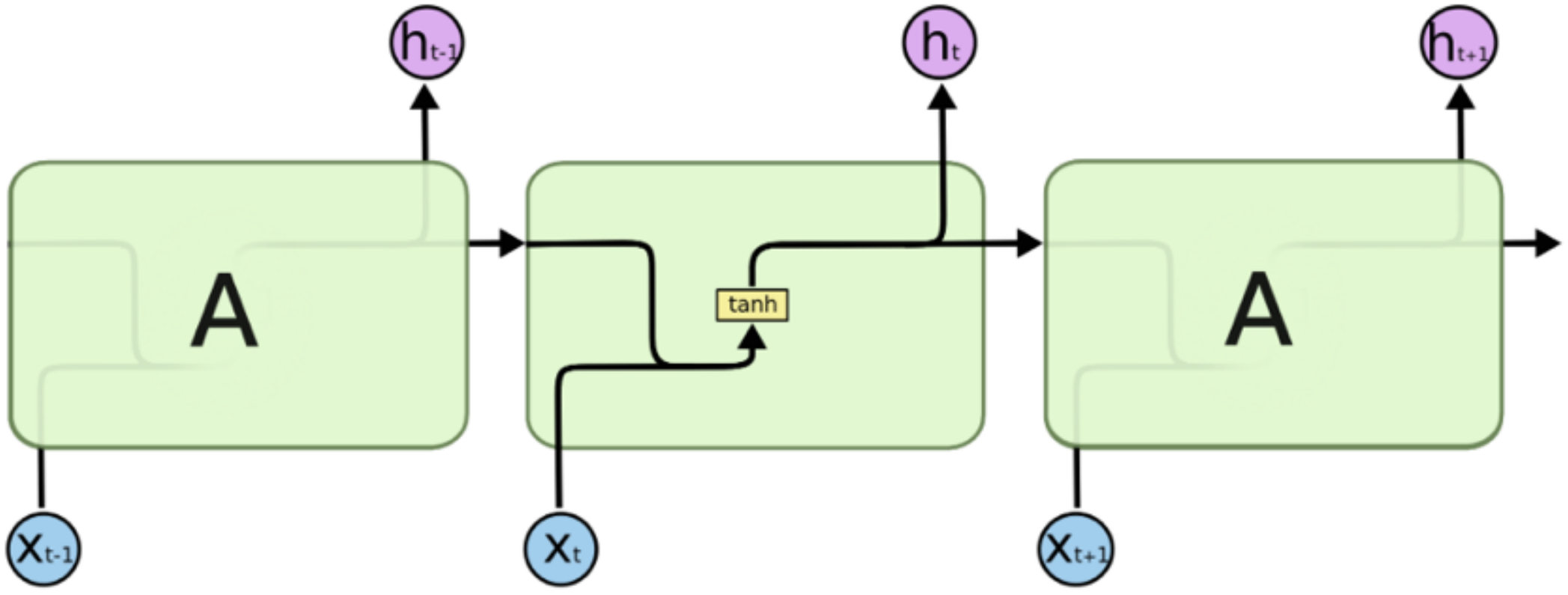
Unrolled RNN

- Multiple copies of the same network

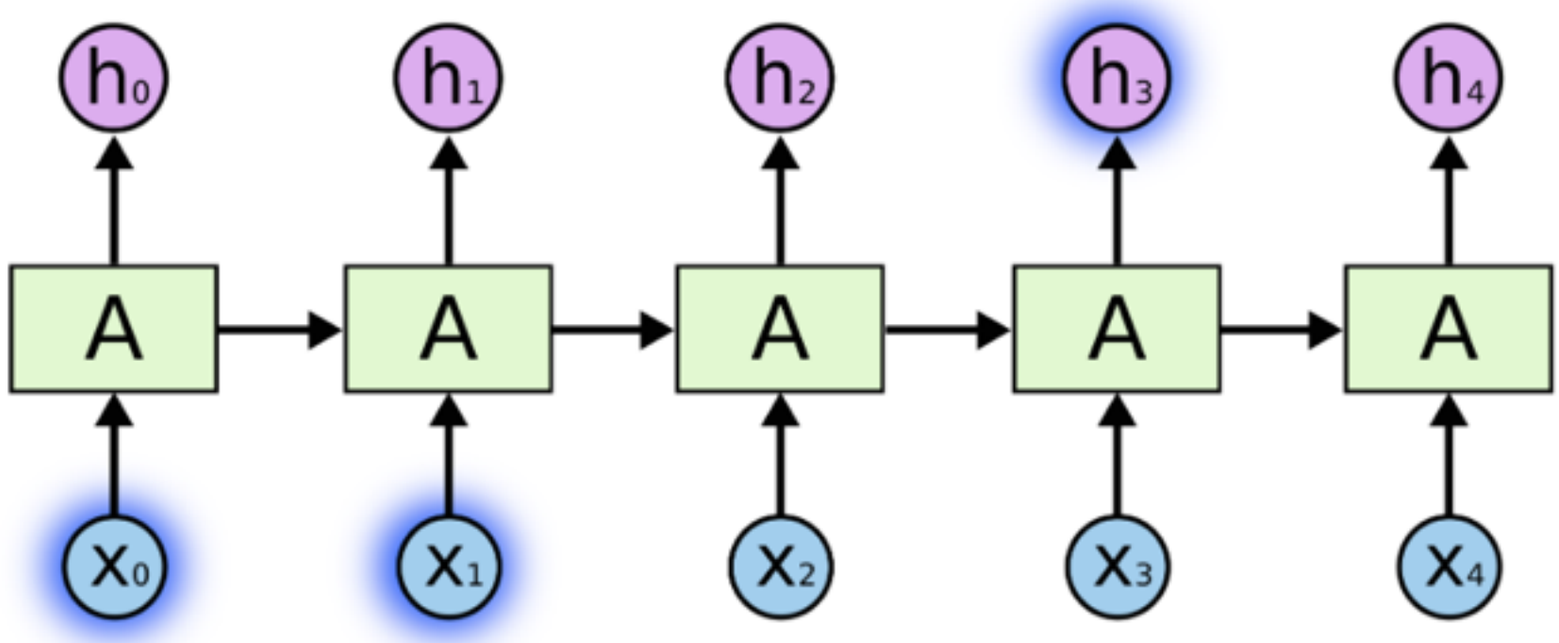


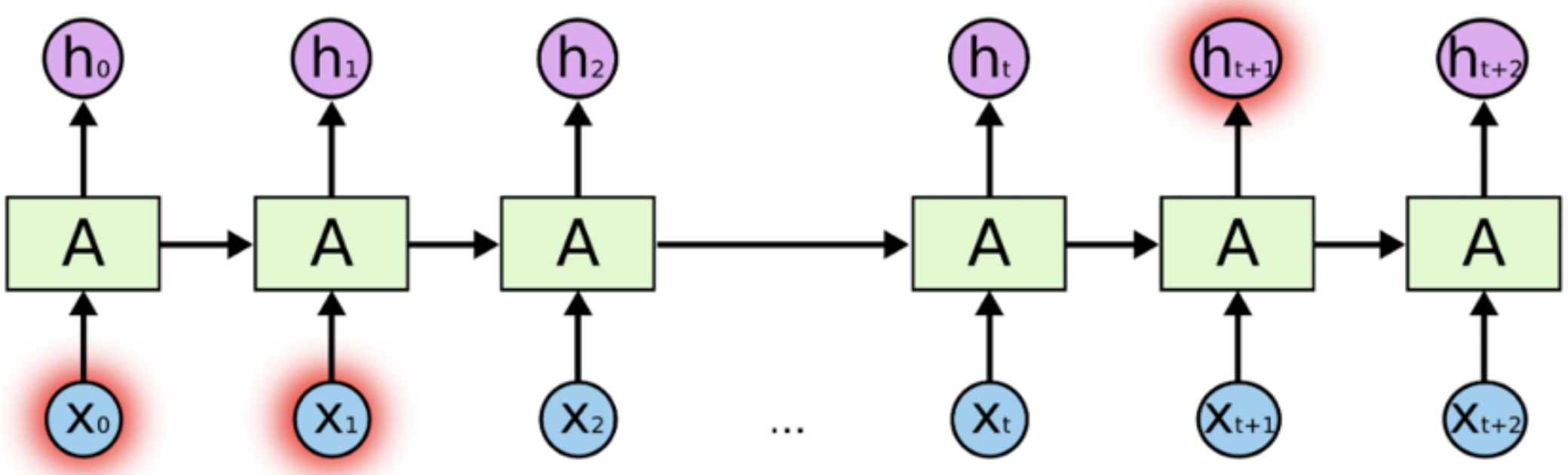
RNNs

- Inside of RNN black-box



The repeating module in a standard RNN contains a single layer.


h0
h1
h2
h3
h4
A
A
A
A
A
X0
X1
X2
X3
X4


h0
h1
h2
ht
ht+1
ht+2
A
A
A
A
A
A
X0
X1
X2
...
Xt
Xt+1
Xt+2

RNN limitations

- Doesn't capture long term dependencies
- Vanishing/exploding gradient problem in BPPT

$$\left\| \frac{\partial h_t}{\partial h_k} \right\| = \left\| \prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}} \right\| \leq (\beta_W \beta_h)^{t-k}$$

Solutions

- Long Short Term Memory networks (LSTMs)

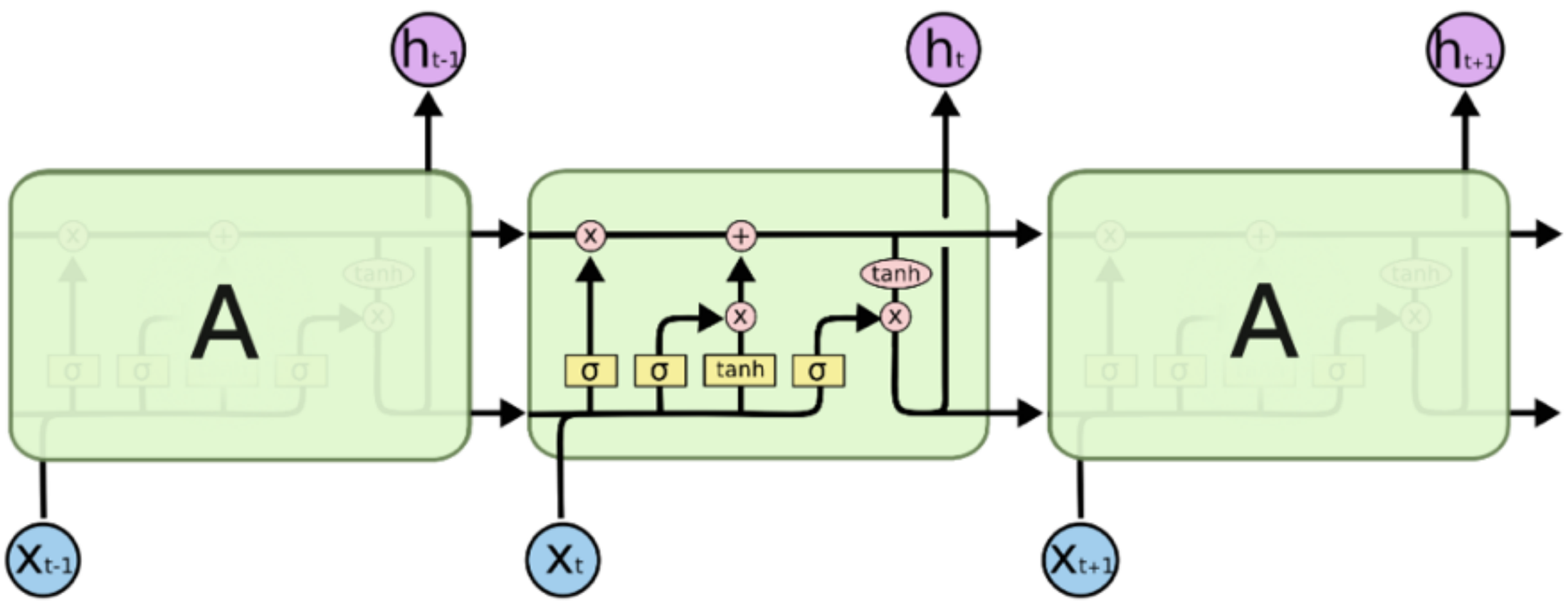
- Gated Recurrent Units (GRUs)

LSTM

- Capable of learning long-term dependencies
- Widely used
- Avoid vanishing gradient problem

LSTM black box

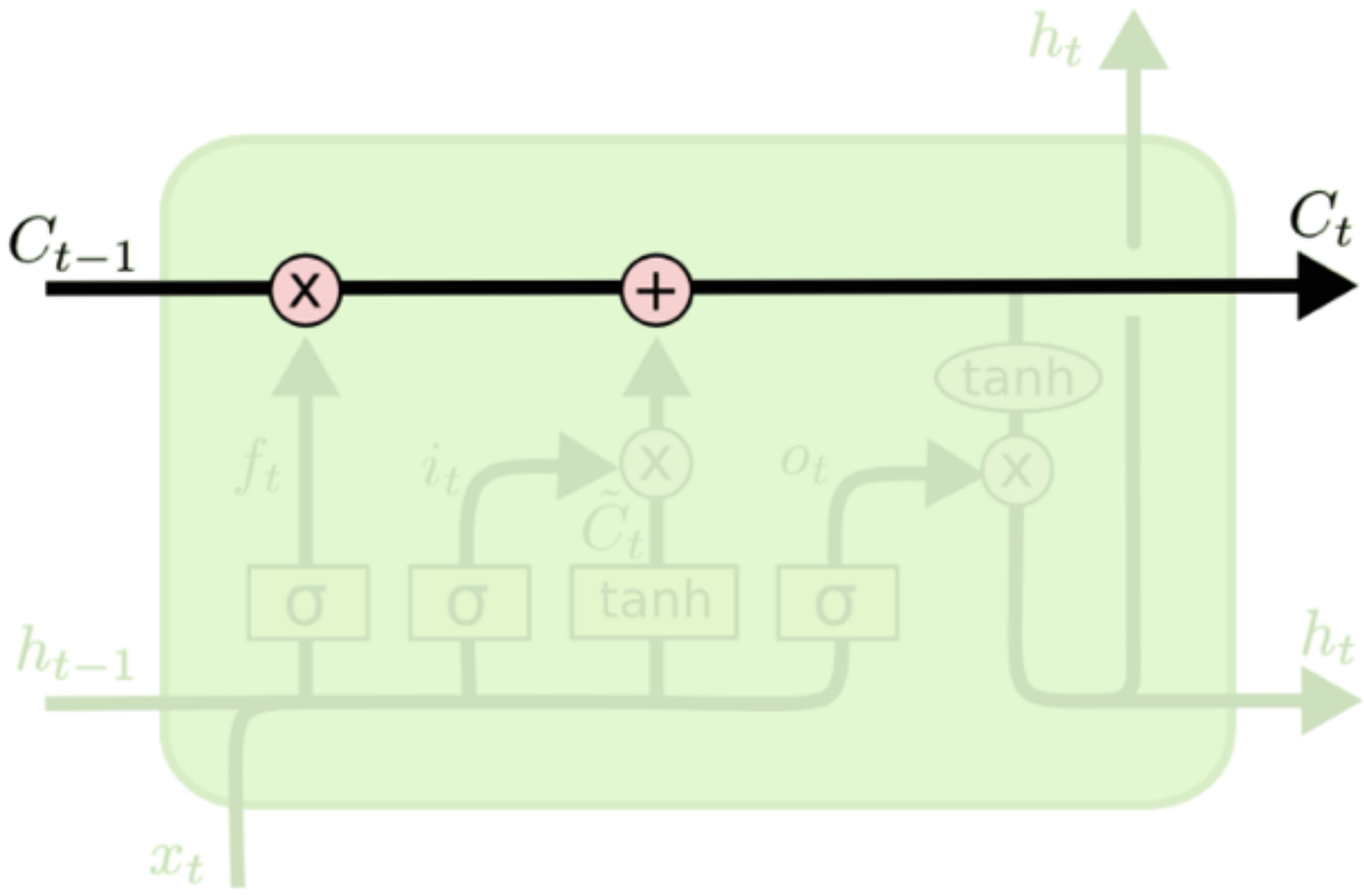
- Has 4 neural network layers instead of 1



The repeating module in an LSTM contains four interacting layers.

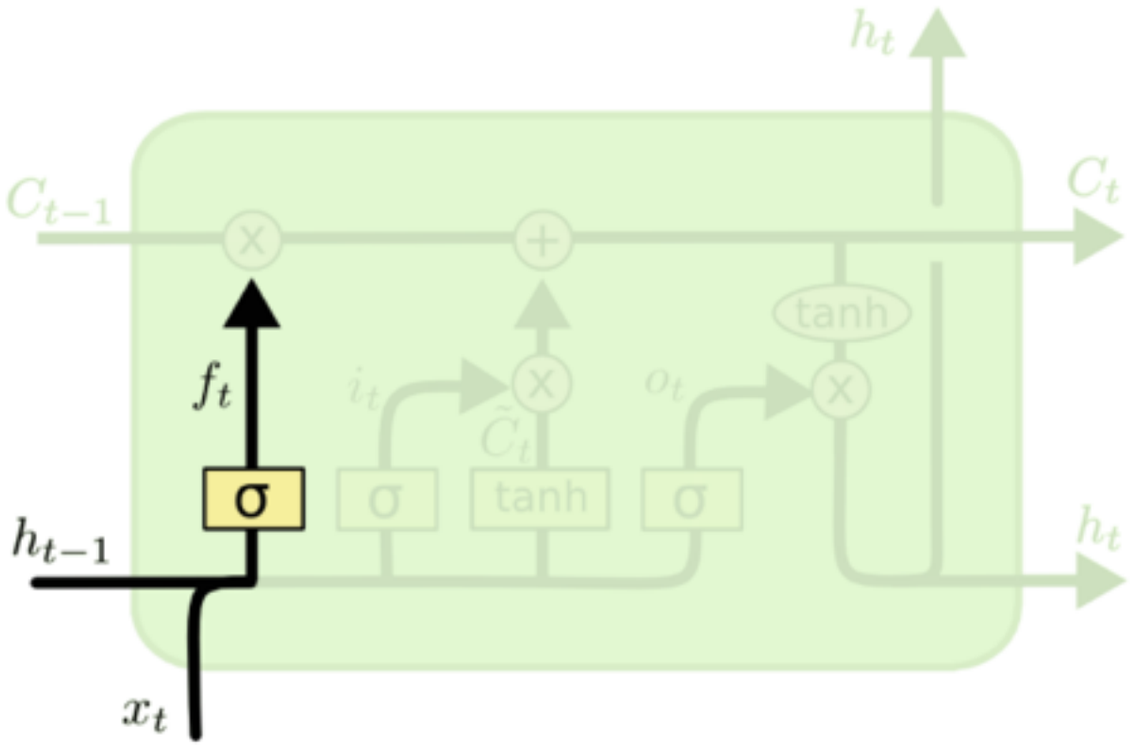
LSTM state

- Minor linear interactions regulated by gates



LSTM forget

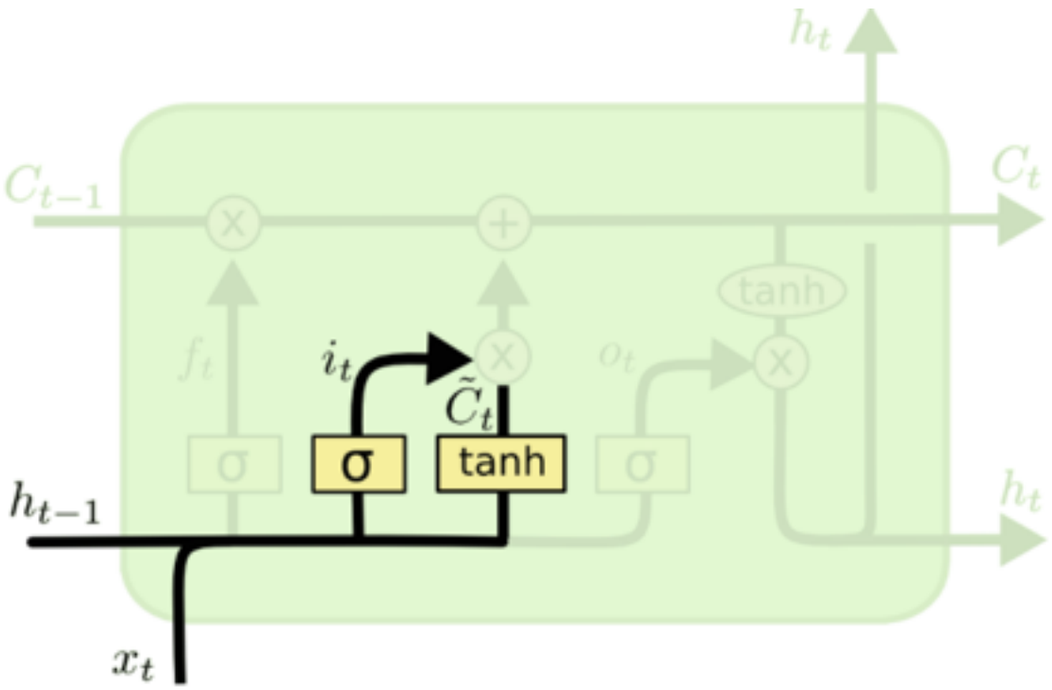
- Determines what information to throw away from the cell state



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

LSTM input

- What information to add to the cell state

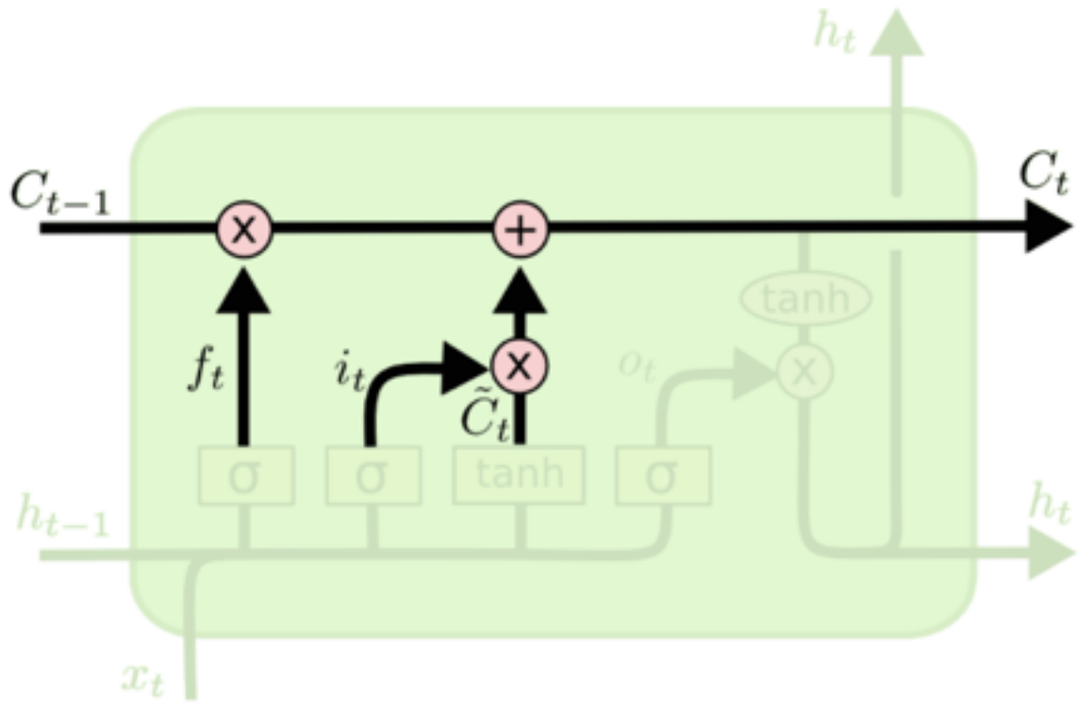


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

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

LSTM new state

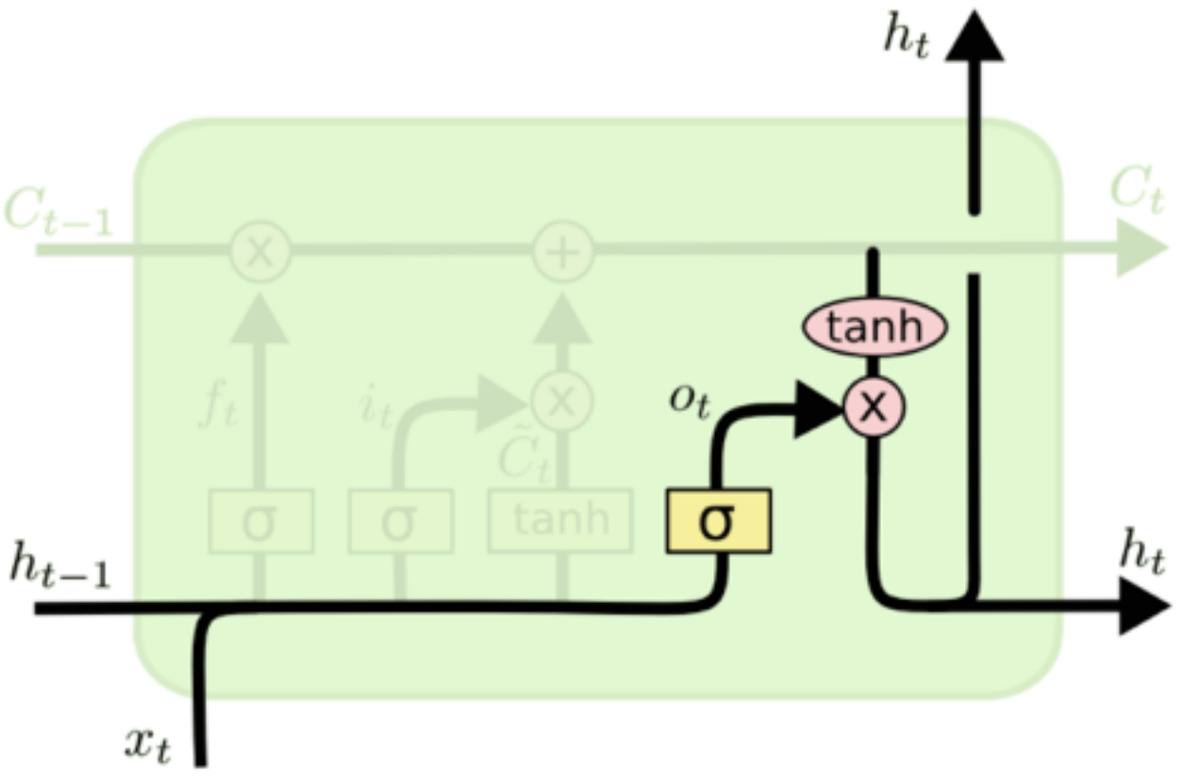
- Update old cell state using forget and input gates



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

LSTM output

- Output filtered version of current cell state

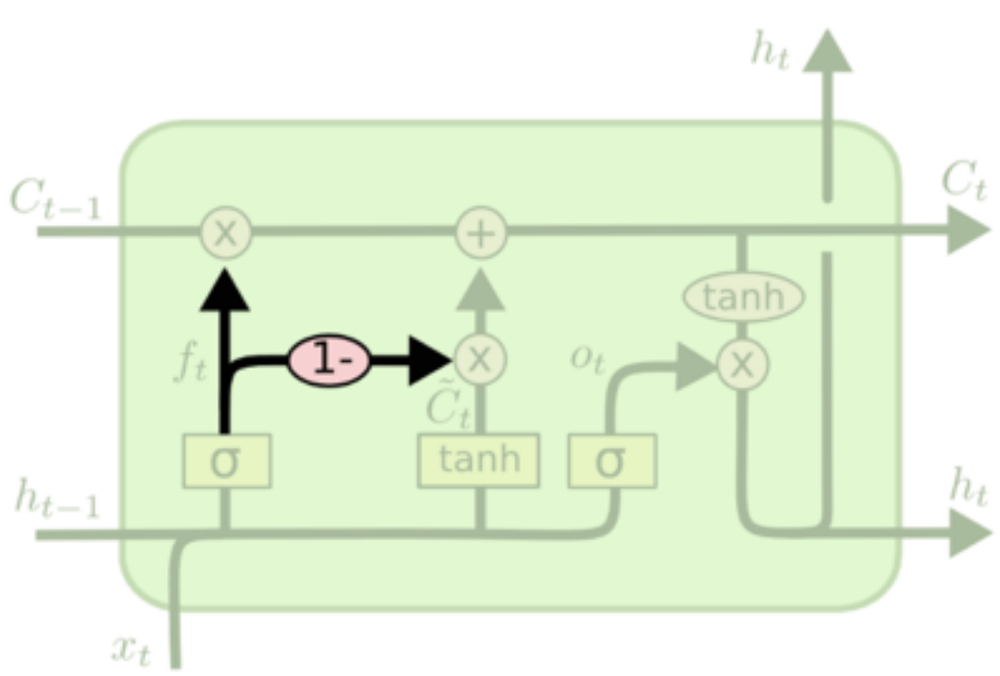


$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

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

LSTM variation

- Coupled forget and input gates

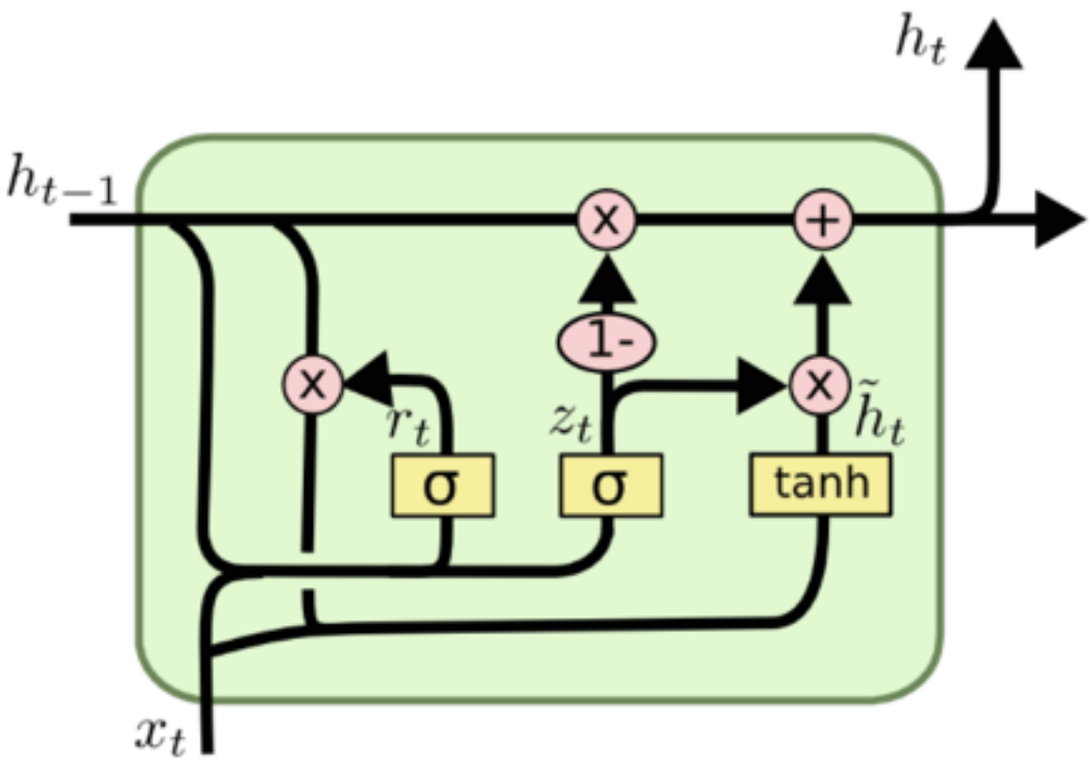


$$C_t = f_t * C_{t-1} + (1 - \boldsymbol{f_t}) * \tilde{C}_t$$

GRU

- Alternative to LSTM
- Simpler unit model
- Merges the cell state and the hidden state

GRU black-box

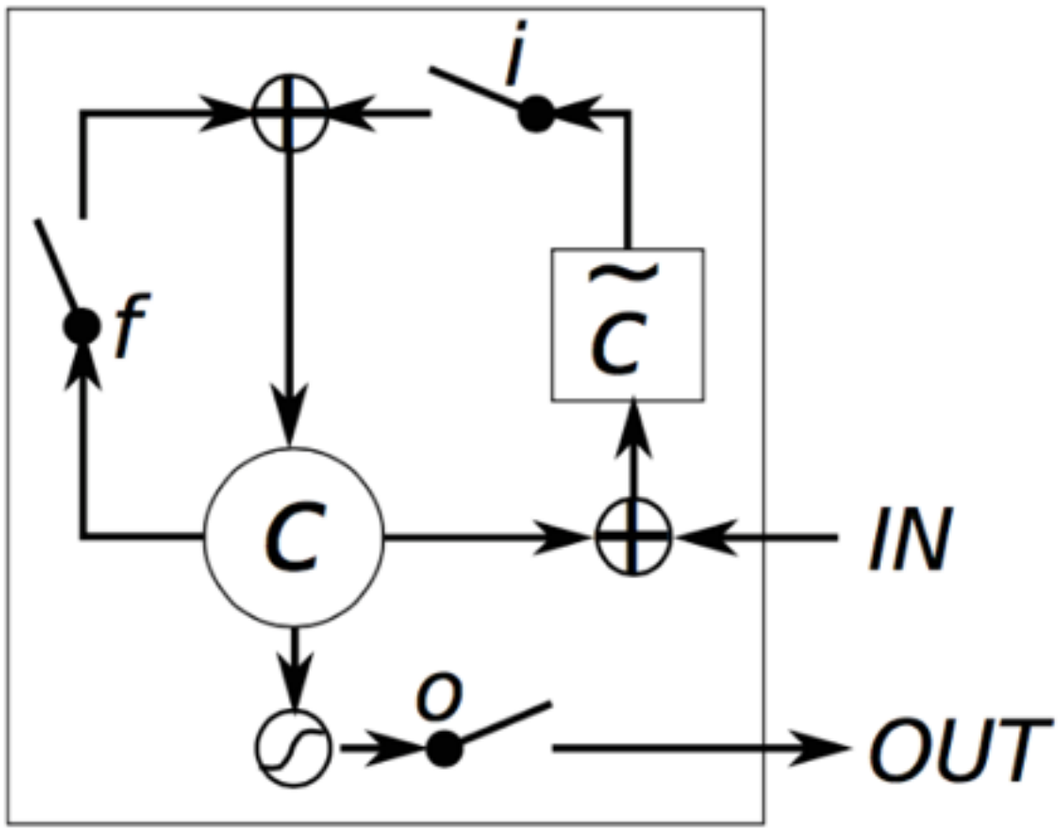


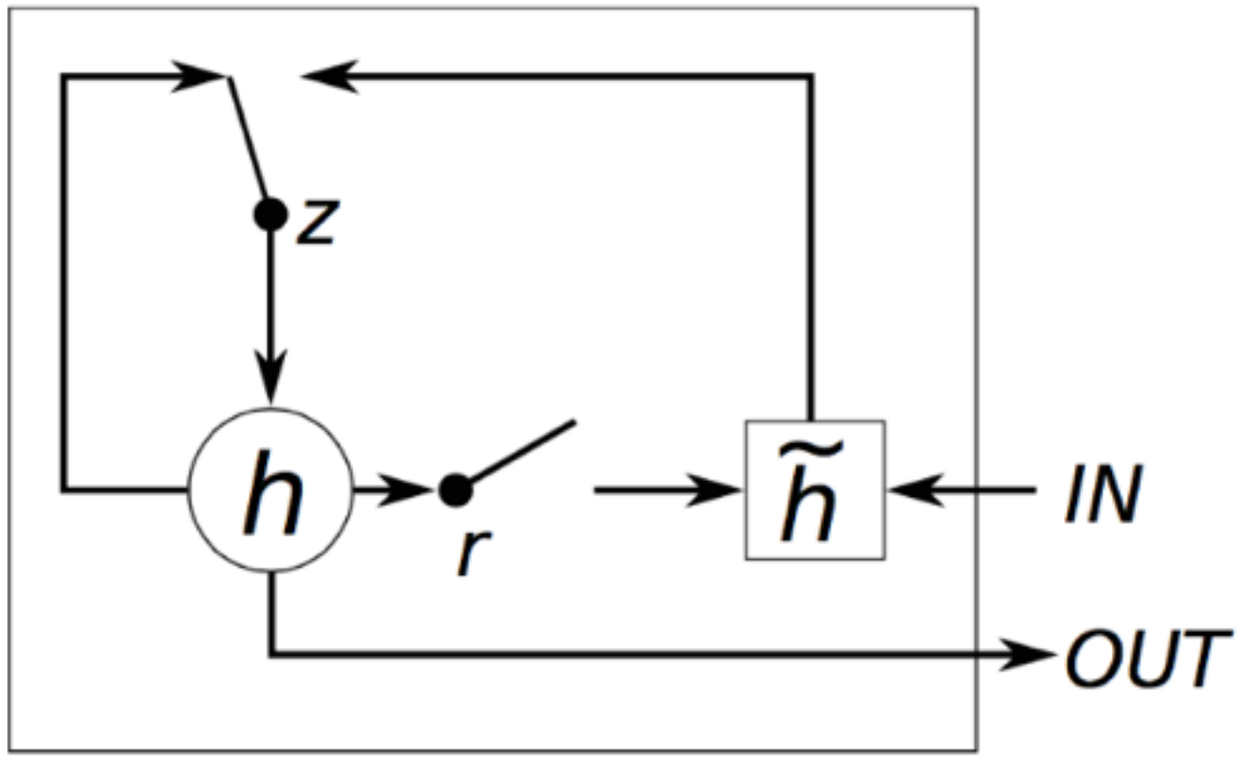
$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

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

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

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


i
f
C
c
IN
o
OUT


z
h
r
h
IN
OUT

LSTM/GRU comparison

- GRU has 2 gates, LSTM has 3
- GRU doesn't have an internal memory c_t
- Input and forget gates are coupled in an update gate
- No additional non linearity in computing the output

LSTM vs GRU

- No clear winner, comparable performance
- Hyperparameter tuning more important than architecture differences
- GRUs have less parameters:
    - trains faster
    - may need less data to generalize
- With enough data, LSTM may be better

Conclusions

- RNN allow flexibility. Vanilla RNNs don't work well
- LSTM/GRU alternatives: additive interactions improve gradient flow
  - Vanishing controlled by LSTM
  - Exploding by gradient clipping
- Simpler architecture still topic of research

Next

- Use word embedding to improve accuracy
- So far: one hot vectors to encode words in the vocabulary
- Use embedding to add semantic meaning (pre-training)

References

- CS231n lecture notes
- Understanding LSTM Networks, Colah's blog
- Recurrent Neural Networks tutorial, WILDML