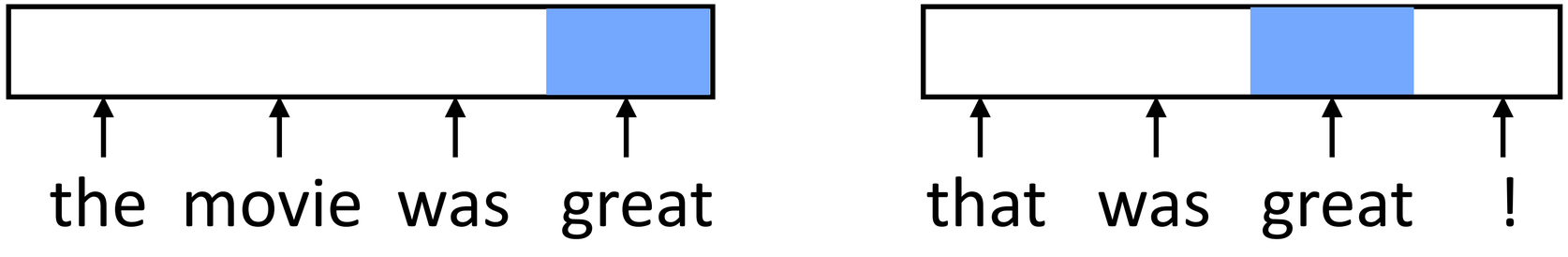
RNN Motivation

- Feedforward NNs can’t handle variable length input: each position in the feature vector has fixed semantics



- These don’t look related (*great* is in two different orthogonal subspaces)

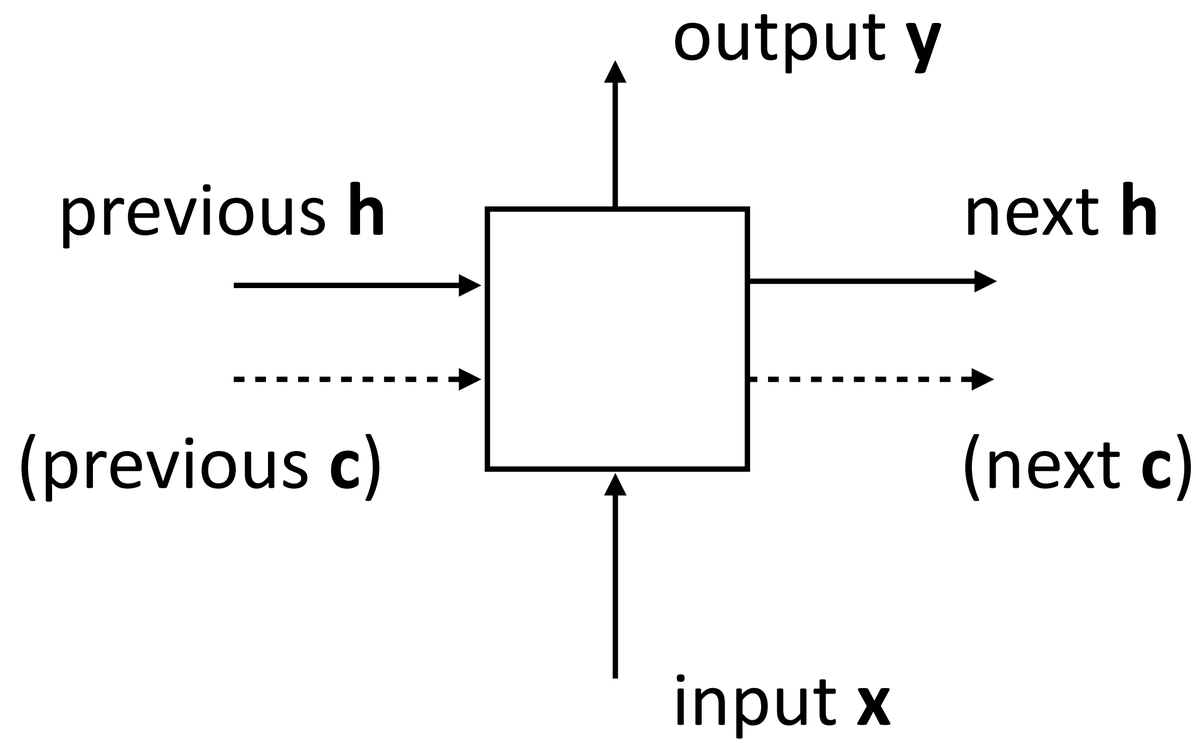
- Instead, we need to:

  1) Process each word in a uniform way

  2) …while still exploiting the context that that token occurs in

RNN Abstraction

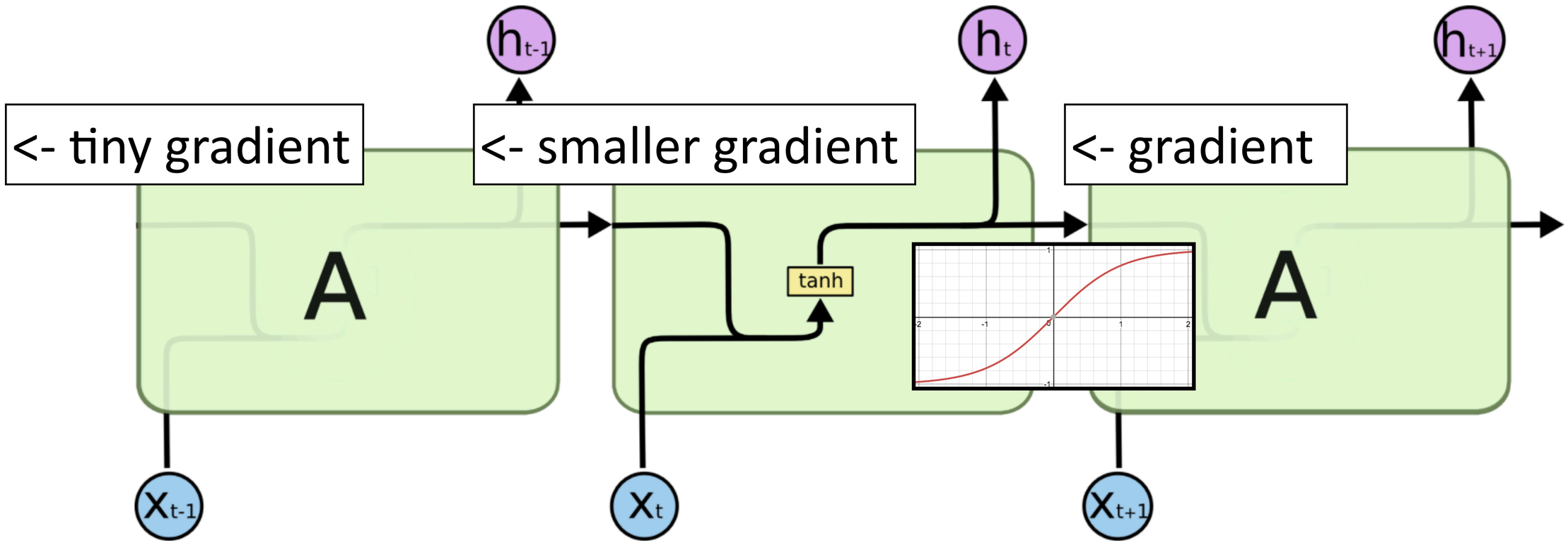
- Cell that takes some input **x**, has some hidden state **h**, and updates that hidden state and produces output **y** (all vector-valued)
- Optionally: cell state **c** (used in LSTMs but not all architectures)



- Example:

the movie was great it was not great

Vanishing Gradient

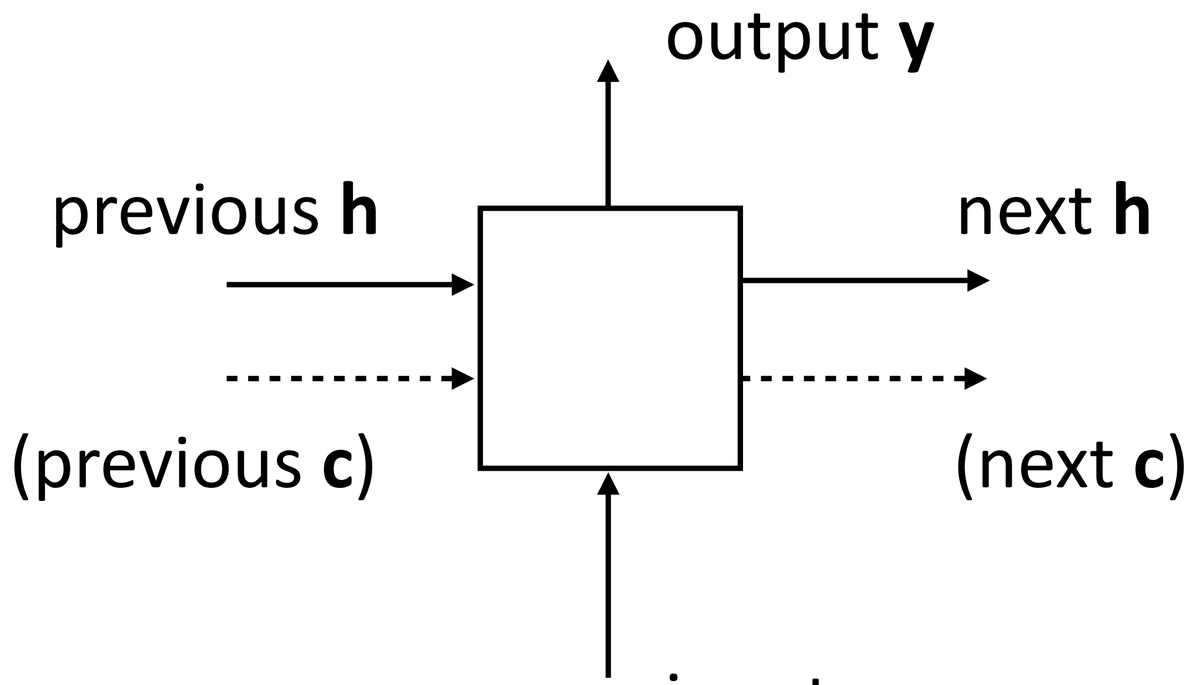


$$\mathbf{h}_t = \tanh(W\mathbf{x}_t + V\mathbf{h}_{t-1} + \mathbf{b}_h)$$

- Gradient diminishes going through tanh; if not in [-2, 2], gradient is almost 0
- Repeated multiplication by V causes problems

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

RNNs: Why not?



- Vanishing gradient makes it hard to learn. LSTMs can help…but not enough*
- Slow. They do not parallelize and there are O(n) non-parallel operations to encode n items
- Solution: Transformers. They can scale to thousands of words!

*This is somewhat addressed by recent innovations like state-space models