Deep Learning

CS256 - Topics in Artificial Intelligence

March 7, 2018

Overview

Sequence Recognition with Neural Networks
- 1D Convolution
- Sequence Recognition
- Recurrent neural networks
- LSTM neural network

Big Picture

So far single element recognition

Sequences

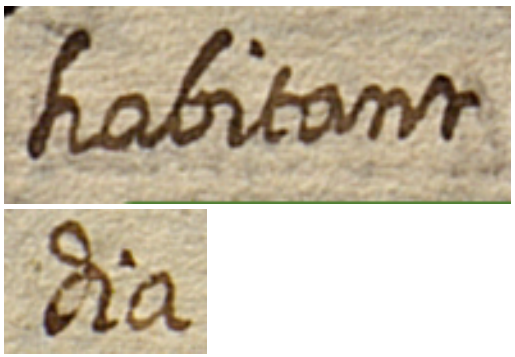
Some sequences

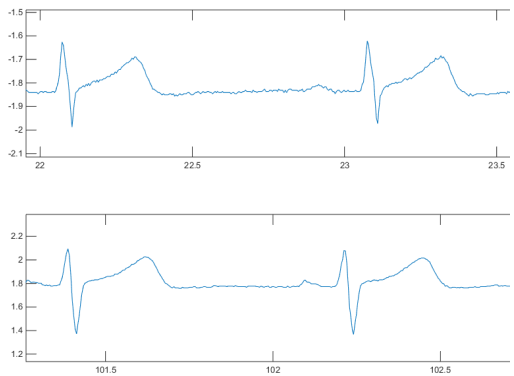
- Sound waves
- Videos
- Stock market data
- Handwriting (pen trajectory)
- Network traffic
- DNA
- Text
- Medical data (ECG, etc.)

Sequence problems come in 3 flavors

Simple sequence classification

- Classifying an entire sequence into one class
  - Digit/word recognition
  - One cycle of ECG data
  - Analysing one tweet (serious, sarcastic, happy, sad, etc.)
  - ...

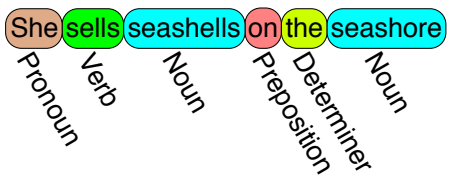




Sequence problems come in 3 flavors

Element-wise sequence classification

- Each element of a sequence is assigned a class/value
  - Part-of-Speech tagging
  - Stock market (daily report)
  - Tweet-sequence classification
  - ..



Sequence problems come in 3 flavors

Sequence-to-sequence mapping

- Input and output sequences are not of the same length
  - Continuous speech recognition
  - Action Recognition
  - Machine Translation
  - ...

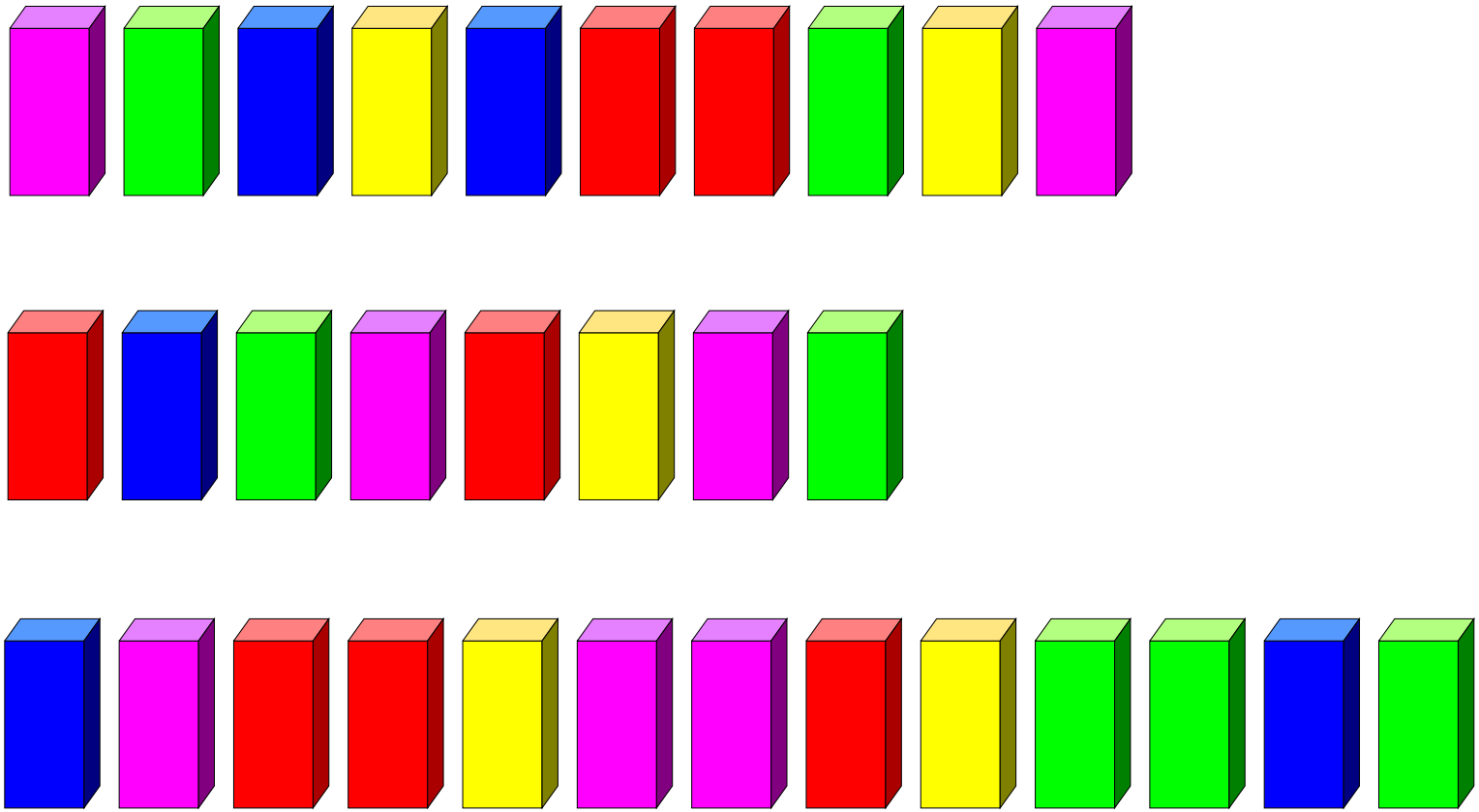
A mí me gusta reconocimiento de patrones

I like pattern recognition

Challenges with Sequences

- 1) Variable length
- 2) Beginning and ending of sub-elements not known
- 3) Long-term dependencies

1) Variable length



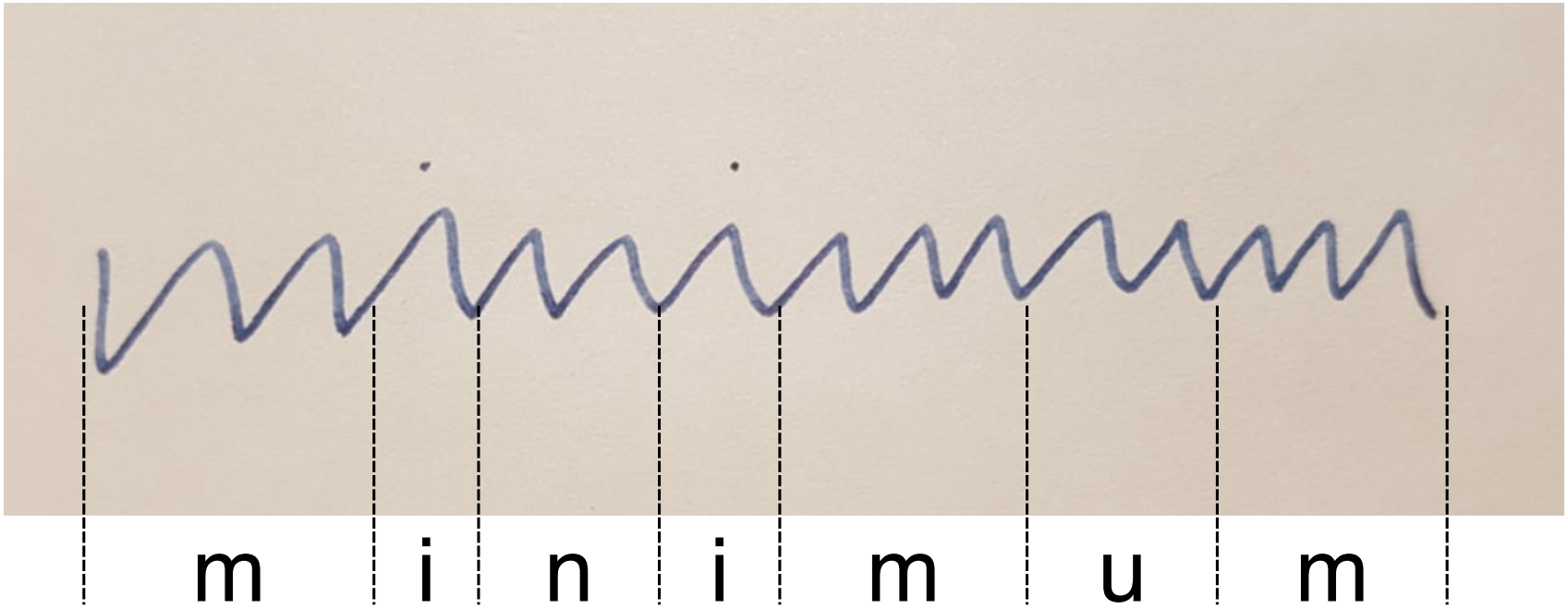
Methods we discussed so far require that each samples has the same size

2) Segmentation

- Beginning and ending of elements in a sequences are not known a priori
  - (for sequence-to-sequence approaches)
- In fact, this can be quite a severe problem (Sayre's Paradox):
  - Segmentation requires recognition
  - Piece-wise recognition requires segmentation

2) Sayre's Paradox

- Segmentation requires recognition
  - without knowing the characters `[m i n i m u m]`, impossible to segment into characters



2) Sayre's Paradox

- Without knowing the segmentation, piece-wise recognition not possible

3) Long-term dependencies

- To predict the next element, not only the immediate past but the distant past may be important

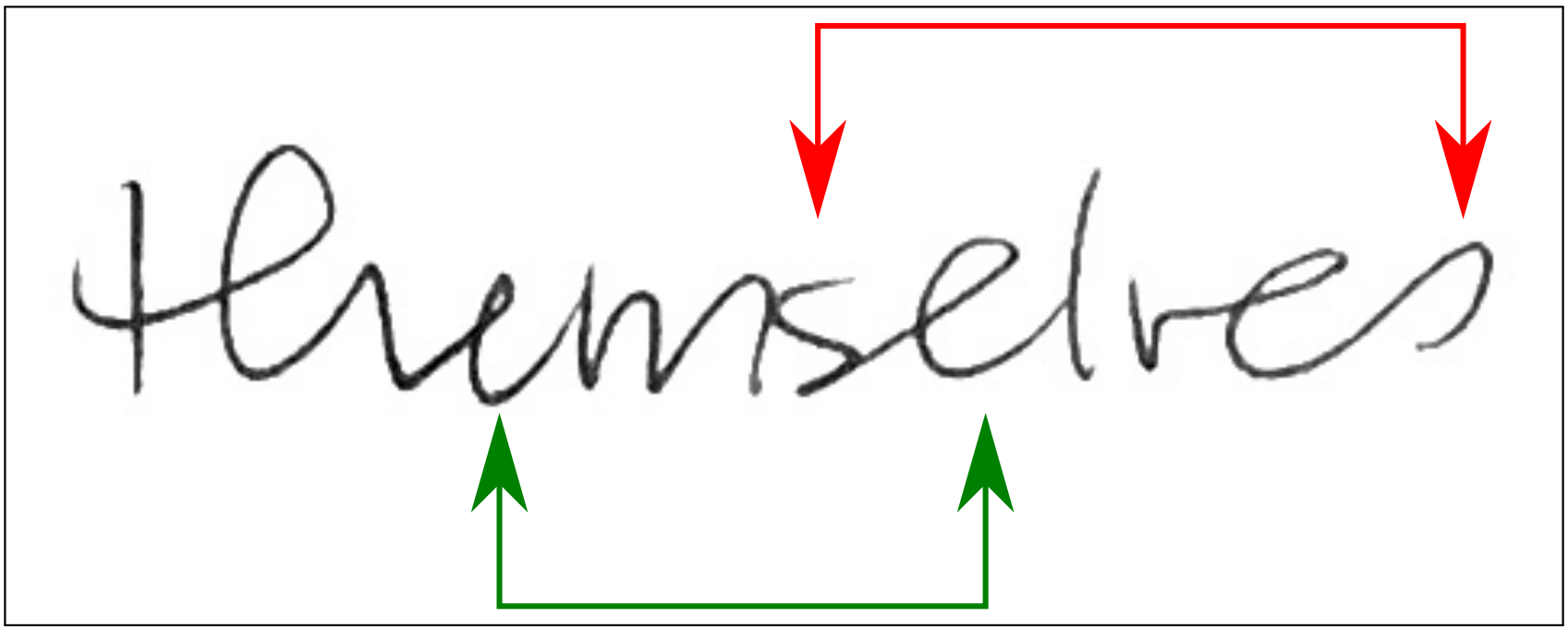
Bear market?

3) Long-term dependencies

- To predict the next element, not only the immediate past but the distant past may be important

3b) Often, the future is helpful as well

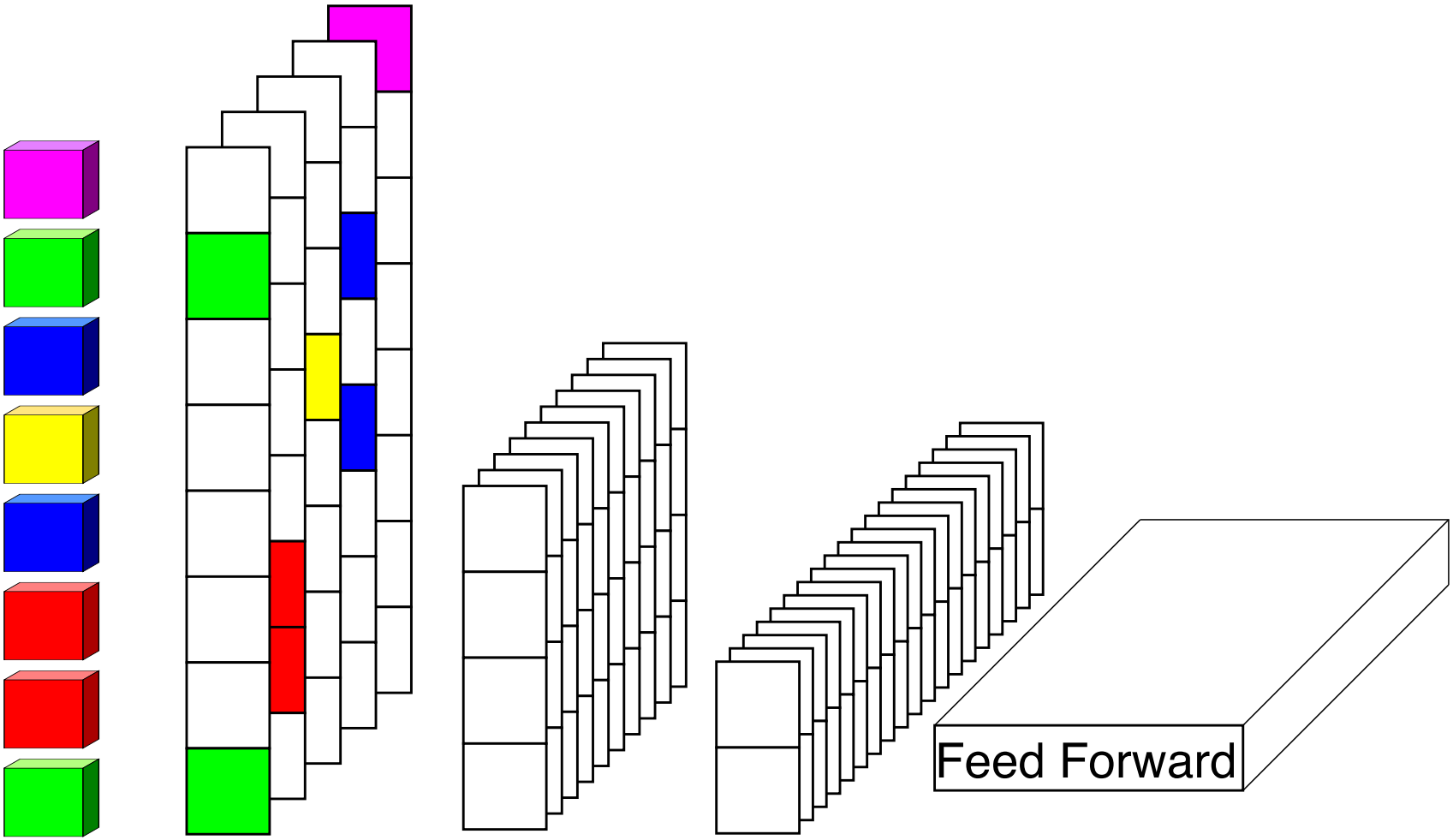
- For example, the shape of a character depends not only on the previous characters, but also the following character



1D Convolution

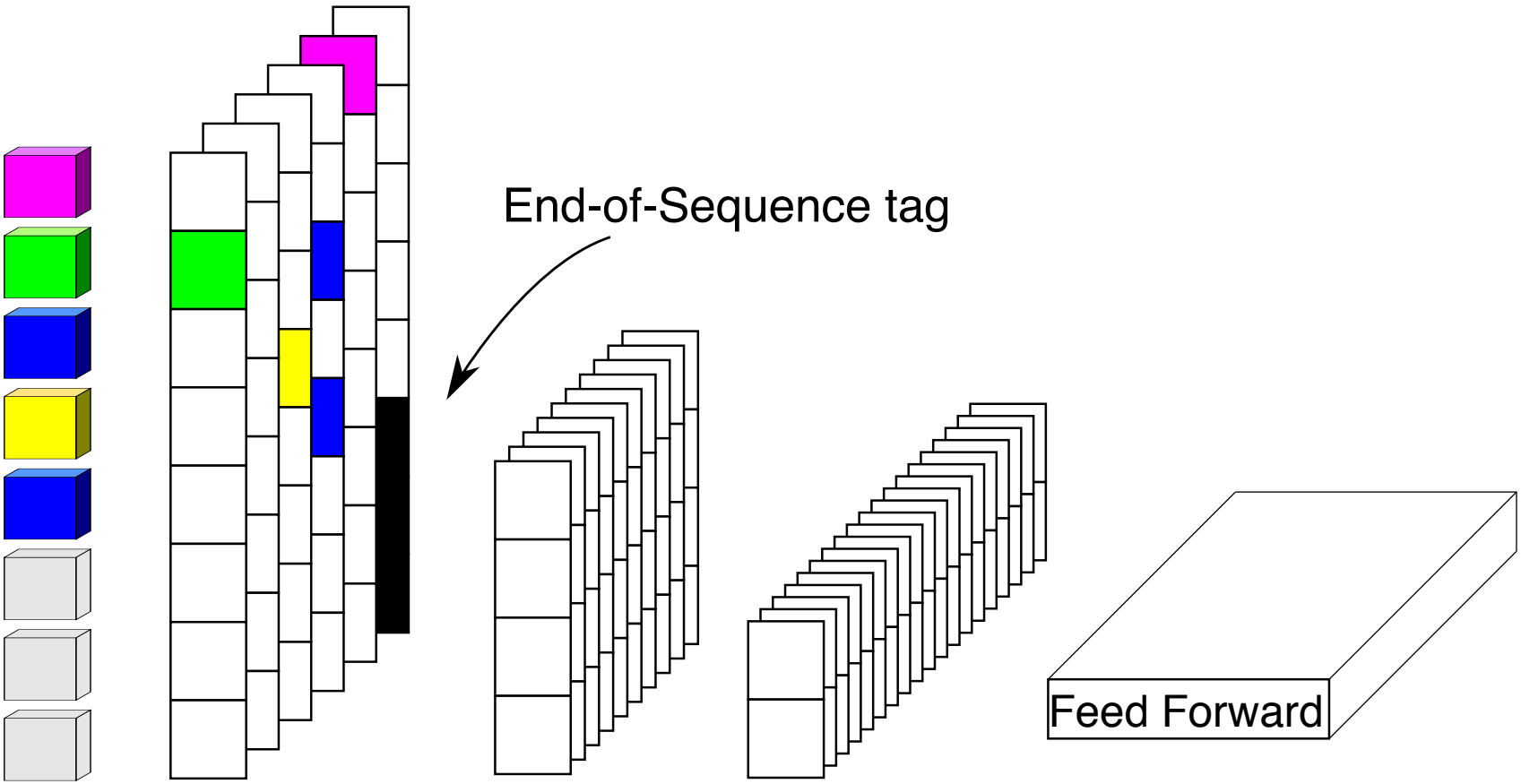
- One possible way to approach sequences are Convolutional neural Networks
  - After all, images are two-dimensional sequences of 3D vectors:
  - i.e. a 3-dim vector (RGB) at position $i_{(x,y)}$ (with x and y within certain bounds)
- For a (normal) n-dim sequence ($s_i \in \mathbb{R}^n$) we could construct the following setup
  - The sequence is represented as a $l \times n \times 1$ tensor for a fixed l
  - (we care later for the case of sequences that are longer/shorter than l)
  - We build a convolutional network, with one dimension constantly 1

1D Convolution



1D Convolution - Padding

- If the input sequence happens to be shorter than slots in our 1D conv network?
  - We pad the remaining cells



1D Convolution - Bucketing

- If the input sequence is longer than the slots in our 1D conv network, we have a problem
- The solution is to train a second, longer network
  - Reuse the kernel weights of the non-feed-forward layers
- We train, e.g., a network with the following buckets

```
[(5), (10), (15), (20), (25), (30), (35), (40)]
```

- If the sequence is of length 17, chose the bucket (20) and pad the end of the sequence with 3 end-of-sequence tags
- Bucketing is helpful to combine sequence learning and batch processing
  - This way, we can group input sequences into the same buckets, and process all sequences in a batch in parallel

Non Convolutional Solutions

- Convolutional neural networks are not the only player on the field
- The other big player are recurrent neural networks, particularly LSTM
- There is an ongoing debate over which is better

Non Convolutional Solutions

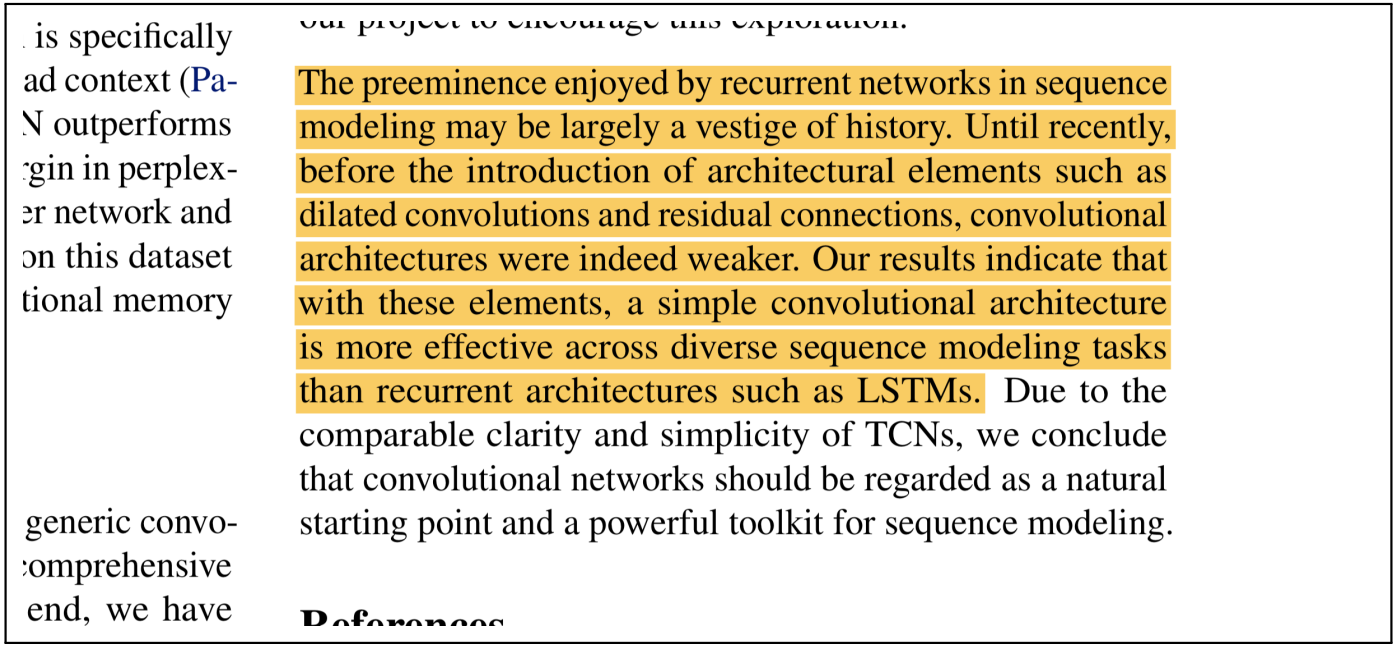
A paper on arxiv, uploaded 3 days ago (this Monday):

An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling

---

Shaojie Bai [1] **J. Zico Kolter** [2] **Vladlen Koltun** [3]

arguing in favor of CNN



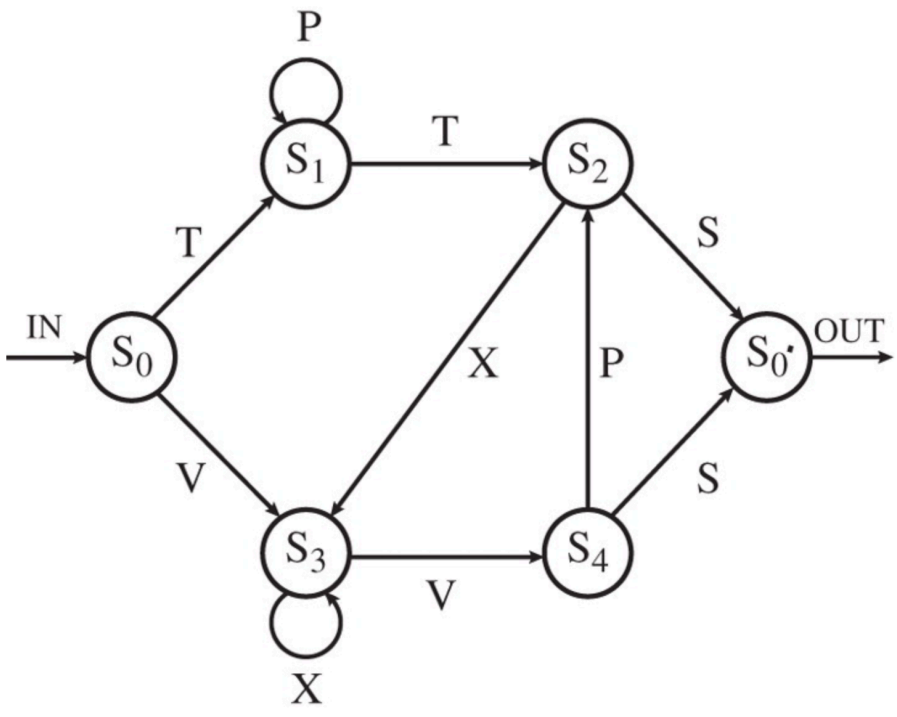
. is specifically
ad context (Pa-
N outperforms
gin in perplex-
er network and
on this dataset
tional memory

generic convo-
comprehensive
end, we have

our project to encourage this exploration.

The preeminence enjoyed by recurrent networks in sequence modeling may be largely a vestige of history. Until recently, before the introduction of architectural elements such as dilated convolutions and residual connections, convolutional architectures were indeed weaker. Our results indicate that with these elements, a simple convolutional architecture is more effective across diverse sequence modeling tasks than recurrent architectures such as LSTMs. Due to the comparable clarity and simplicity of TCNs, we conclude that convolutional networks should be regarded as a natural starting point and a powerful toolkit for sequence modeling.

References

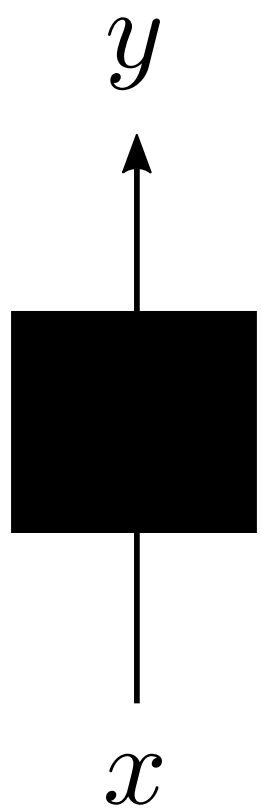


example strings:

TTS
TPPTS
TPPPTXVS
VXVS
VVS
TTXVPXVPS
TPPPPPPPTS
VXXXXXXVS
TPPTXVPXXXXXXVPXVPS

Classification Task

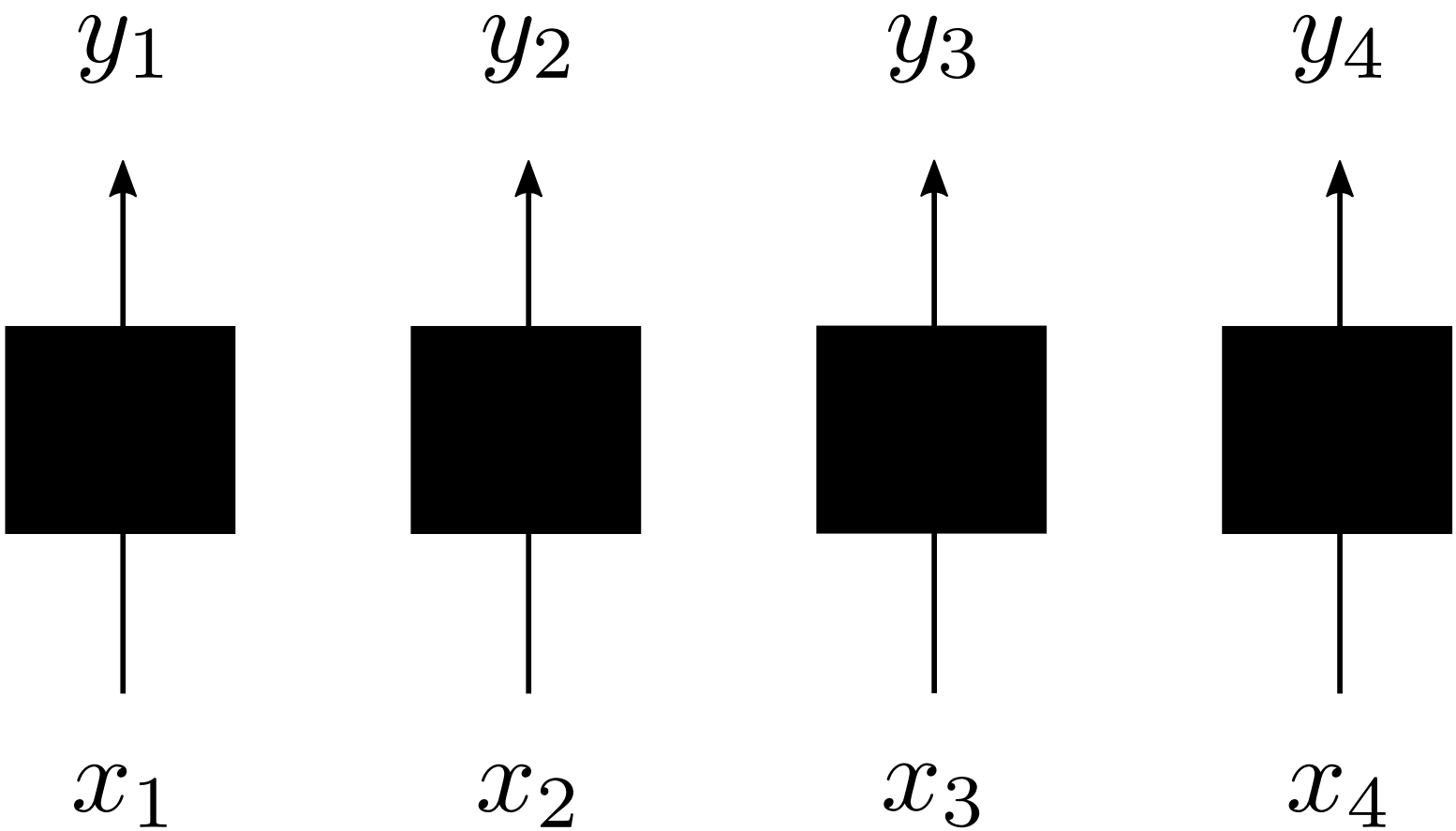
mapping x to y



$$\mathcal{F} : x \in \mathbb{X} \rightarrow y \in \mathbb{Y}$$

Independent Sequence Classification Task

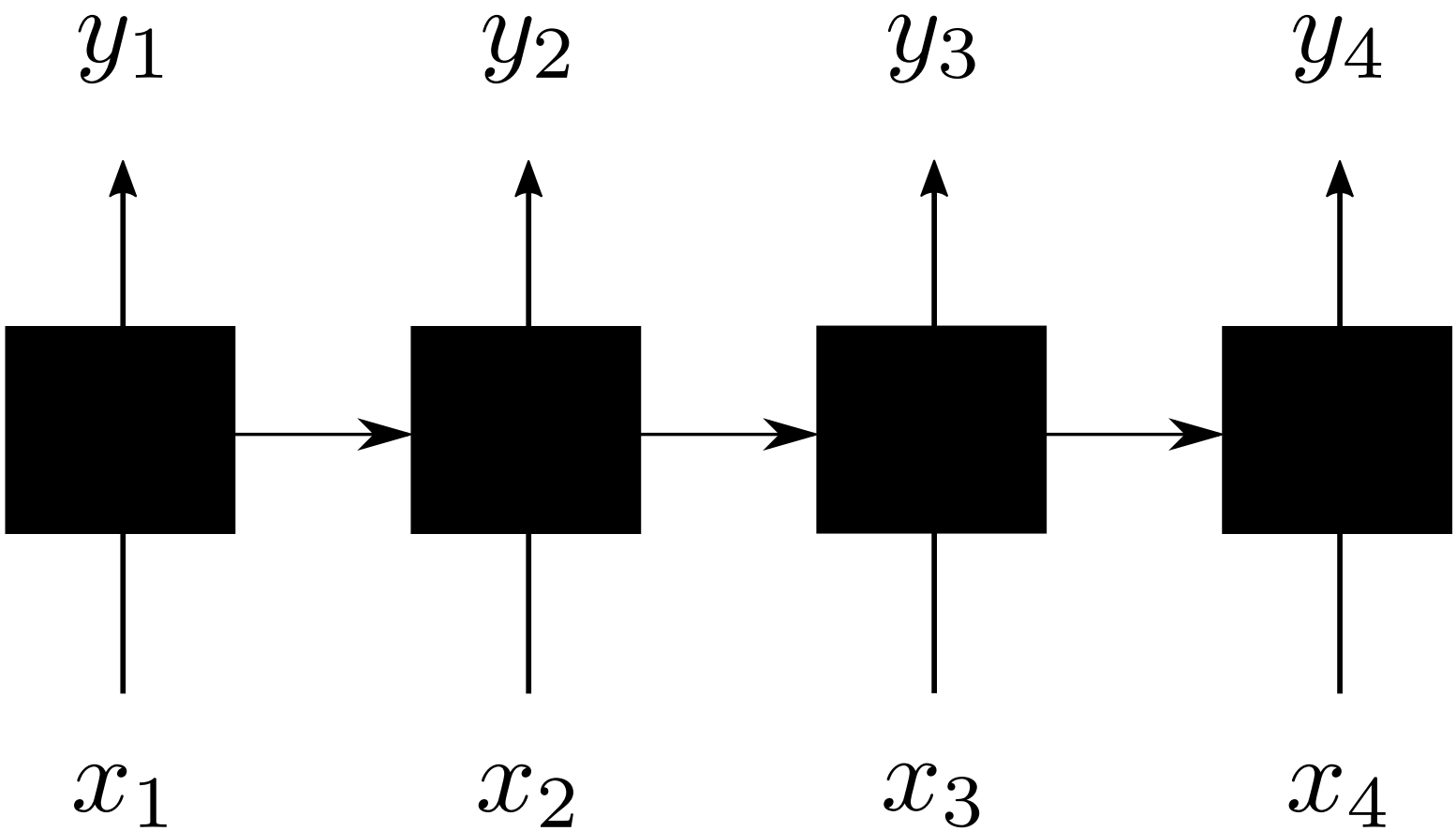
mapping $x_1 x_2 x_3 \ldots$ to $y_1 y_2 y_3 \ldots$



$$\mathcal{F} : \mathbf{x} \in \mathbb{X}^* \rightarrow \mathbf{y} \in \mathbb{Y}^*$$

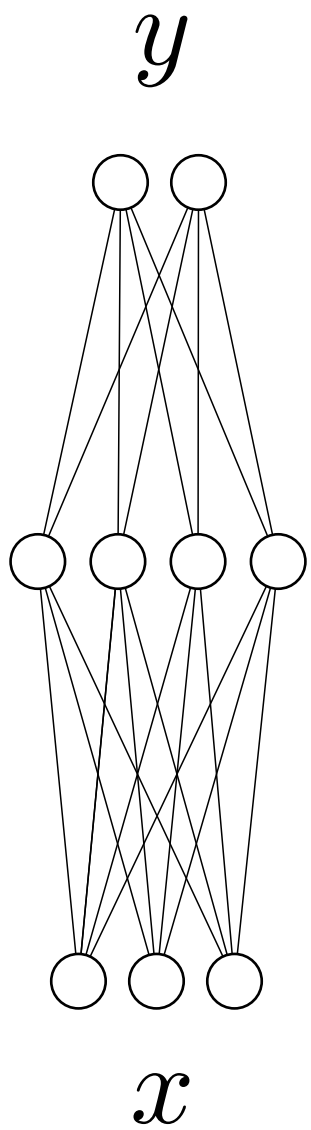
Sequence Classification Task

mapping $x_1 x_2 x_3 \ldots$ to $y_1 y_2 y_3 \ldots$



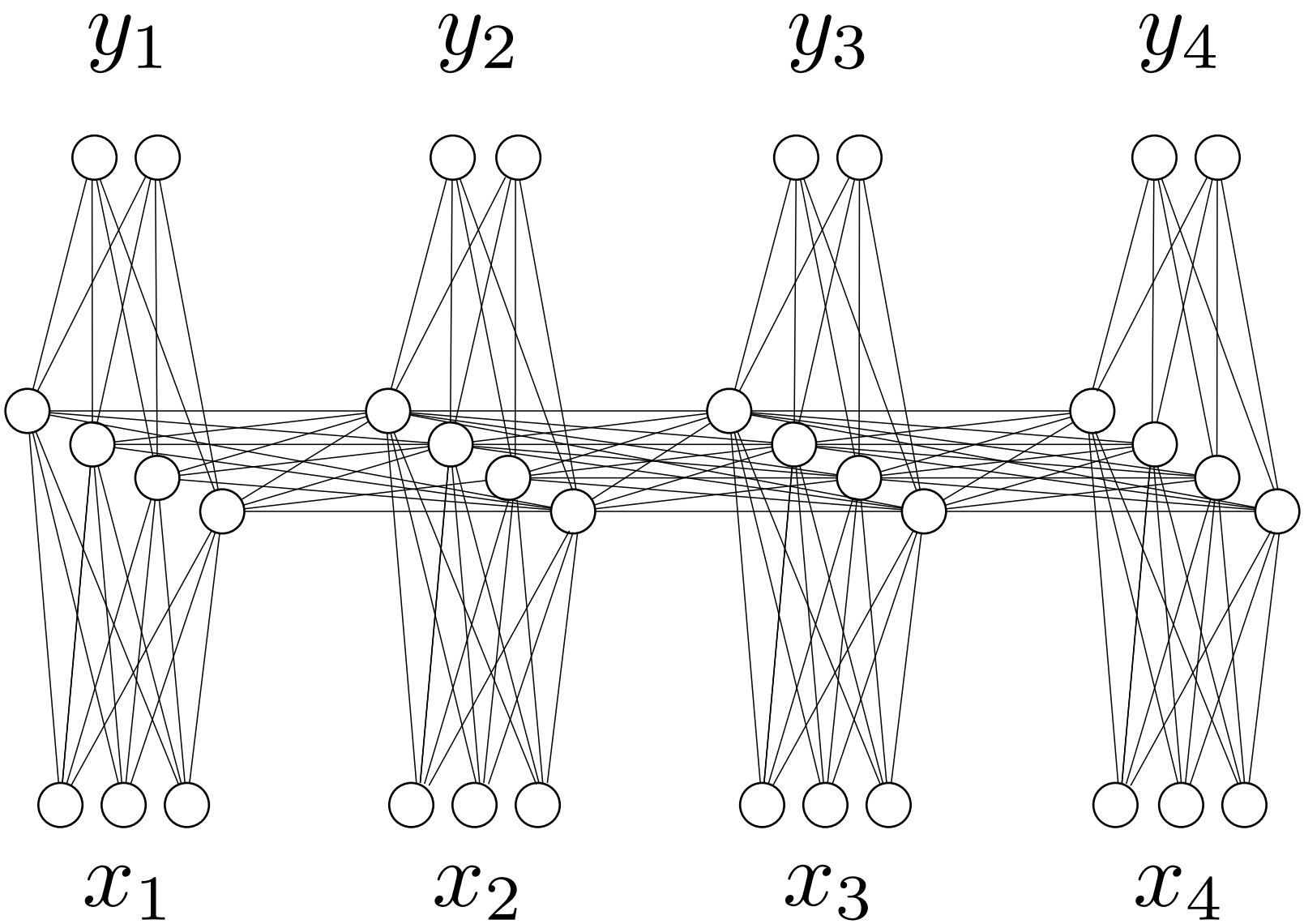
$$\mathcal{F} : \mathbf{x} \in \mathbb{X}^* \rightarrow \mathbf{y} \in \mathbb{Y}^*$$

Feed-Forward Neural Networks



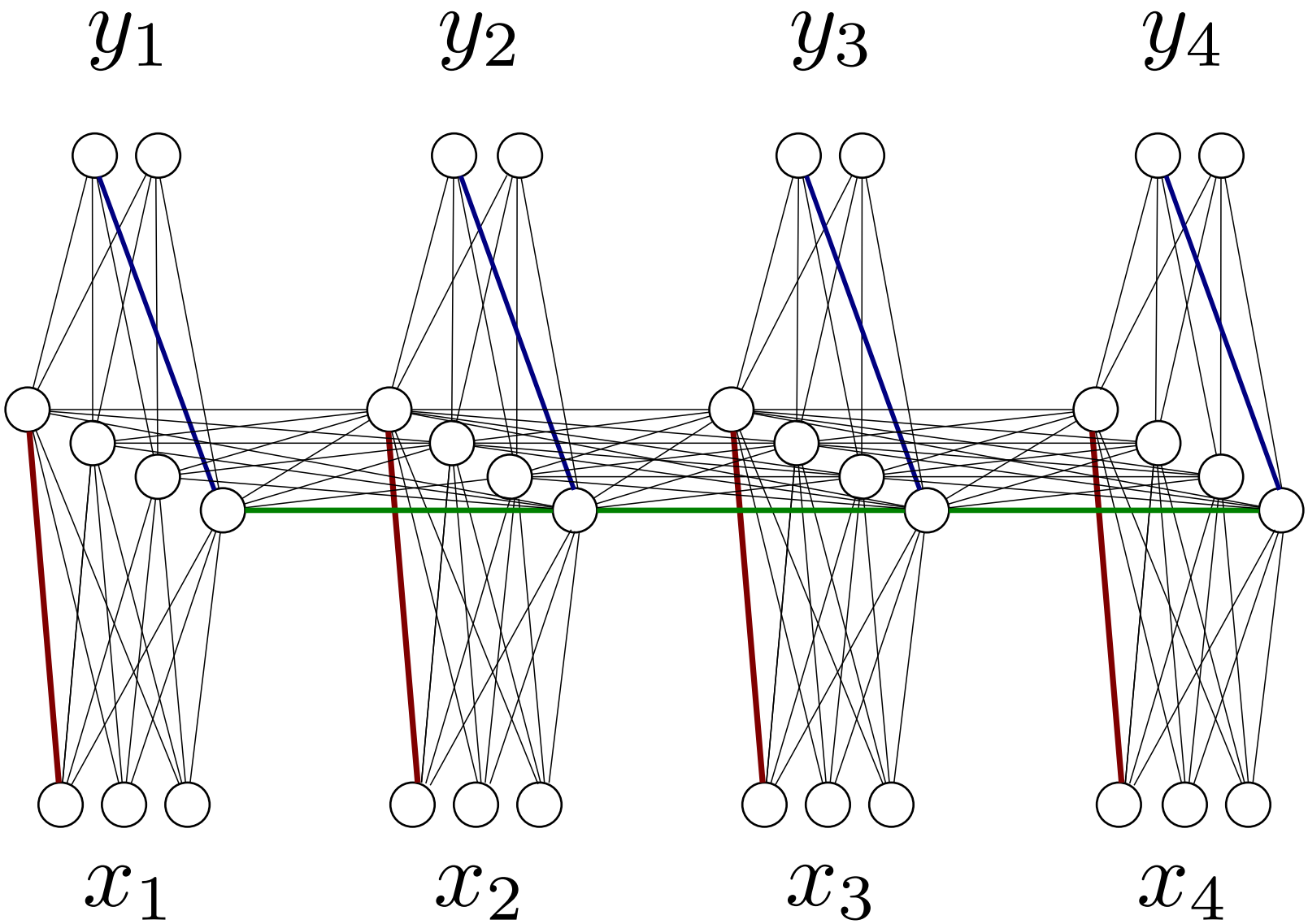
Recurrent Neural Networks

Unfolded in time



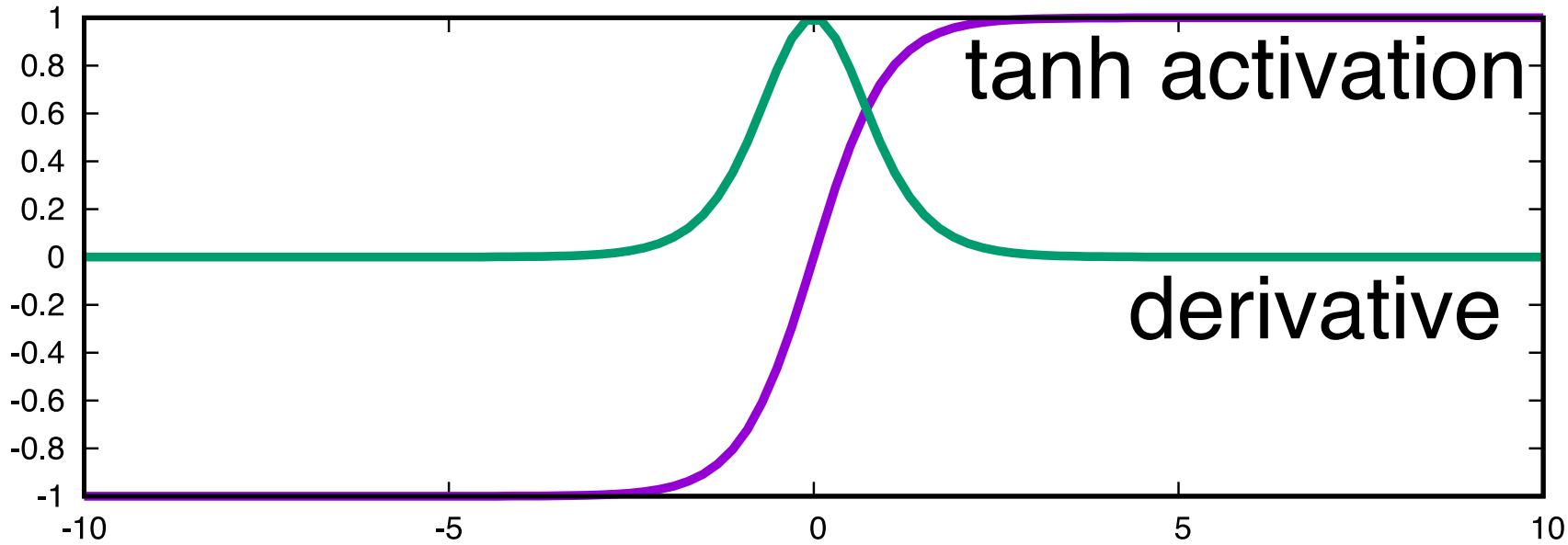
Recurrent Neural Networks

For every element, use the same weights



Problems with simple RNN architectures

- Vanishing Gradient Problem
  - Back-propagation with unfolding in time
  - At each step in the back-propagation phase, the gradient passes through the hidden node
  - Imagine the activation function is $f(z) = \tanh(z)$. Derivative is $1 - \tanh^2(z)$
  - Each time, the gradient is multiplied by a value $\in [0, 1]$
  - Exponentially decreasing gradient, after a few steps inaccuracies in representing floats (double) have more impact

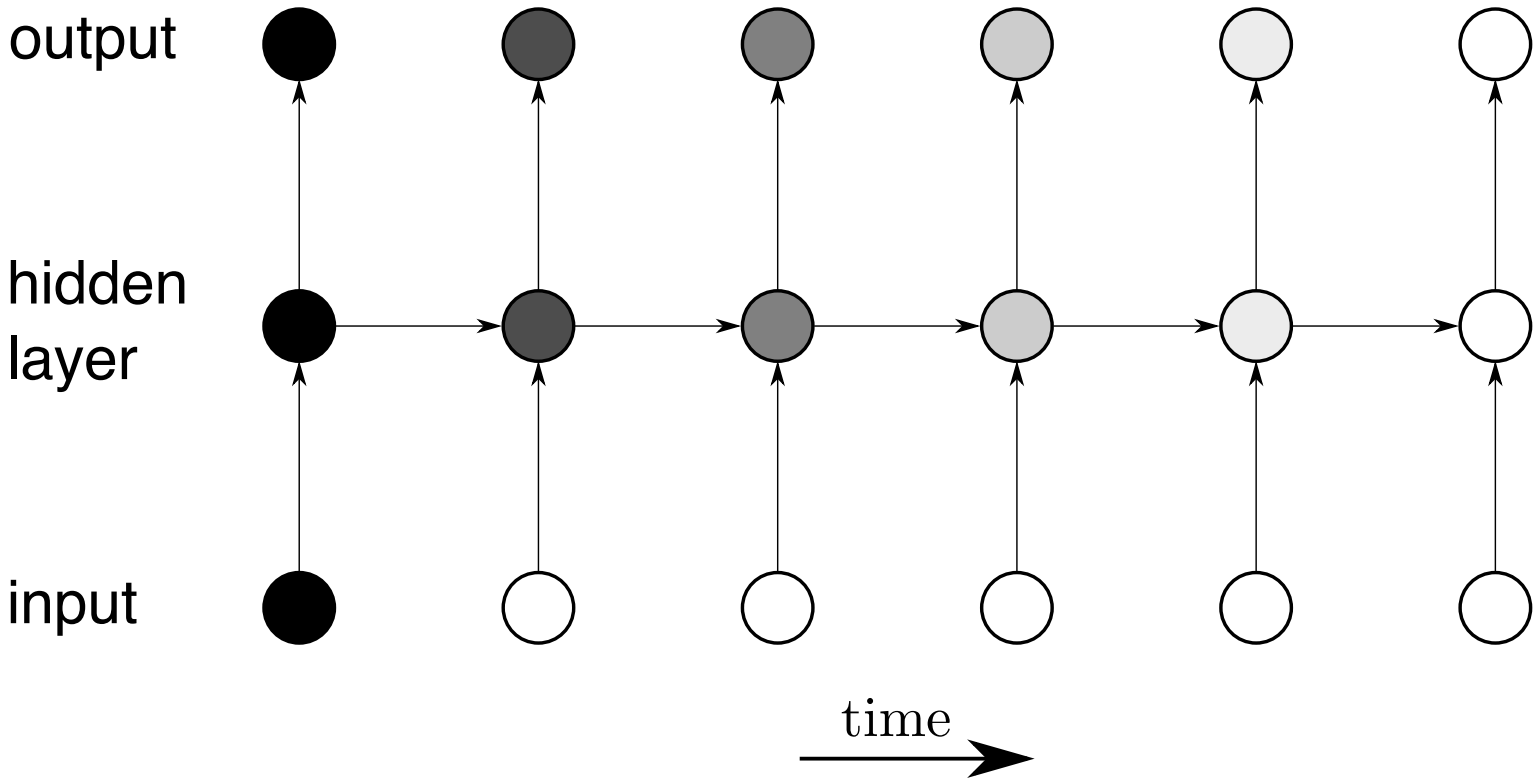


Problems with simple RNN architectures

- Because of the vanishing gradient problem, the sensitivity to input at time $t$decreases rapidly
- Only limited context if effectively usable
- Long-term storage of information is not possible

Sensitivity to input at $t = 0$

The darker the more sensitive



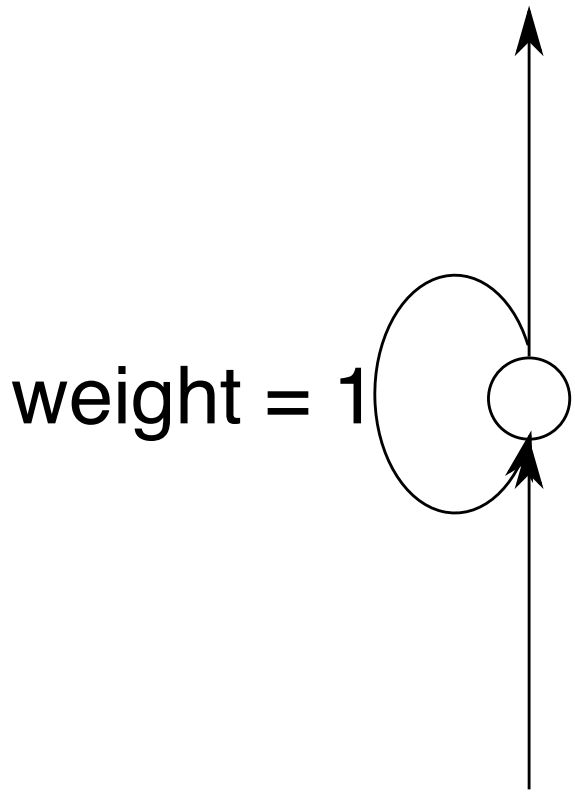
Solution to the vanishing gradient problem

What we need is some form of memory cell

- Set a value
- Reset the cell
- Read from the cell
- Fully differentiable (for back-propagation)
  - We need to compose it out of differentiable functions

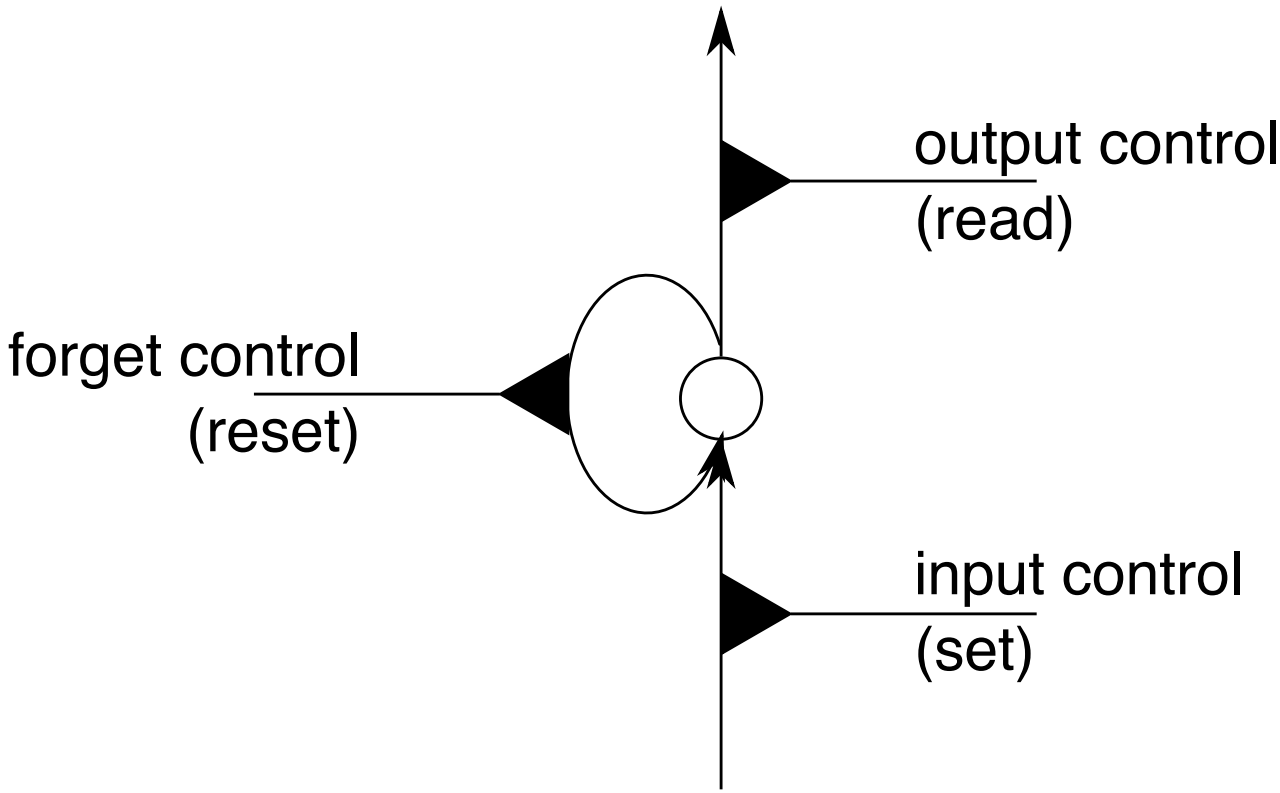
Constructing a differentiable memory cell

Starting point



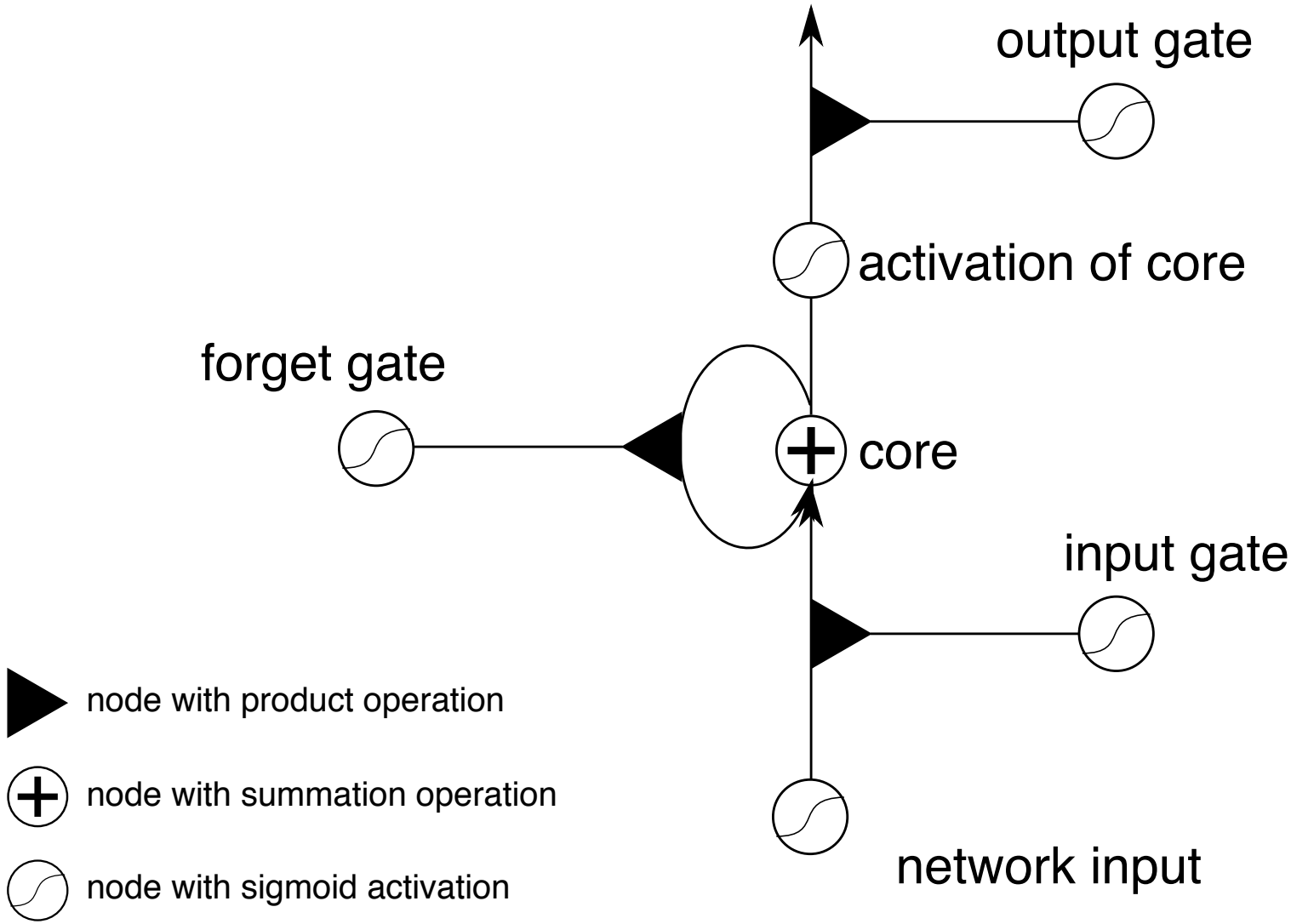
Constructing a differentiable memory cell

Adding controls



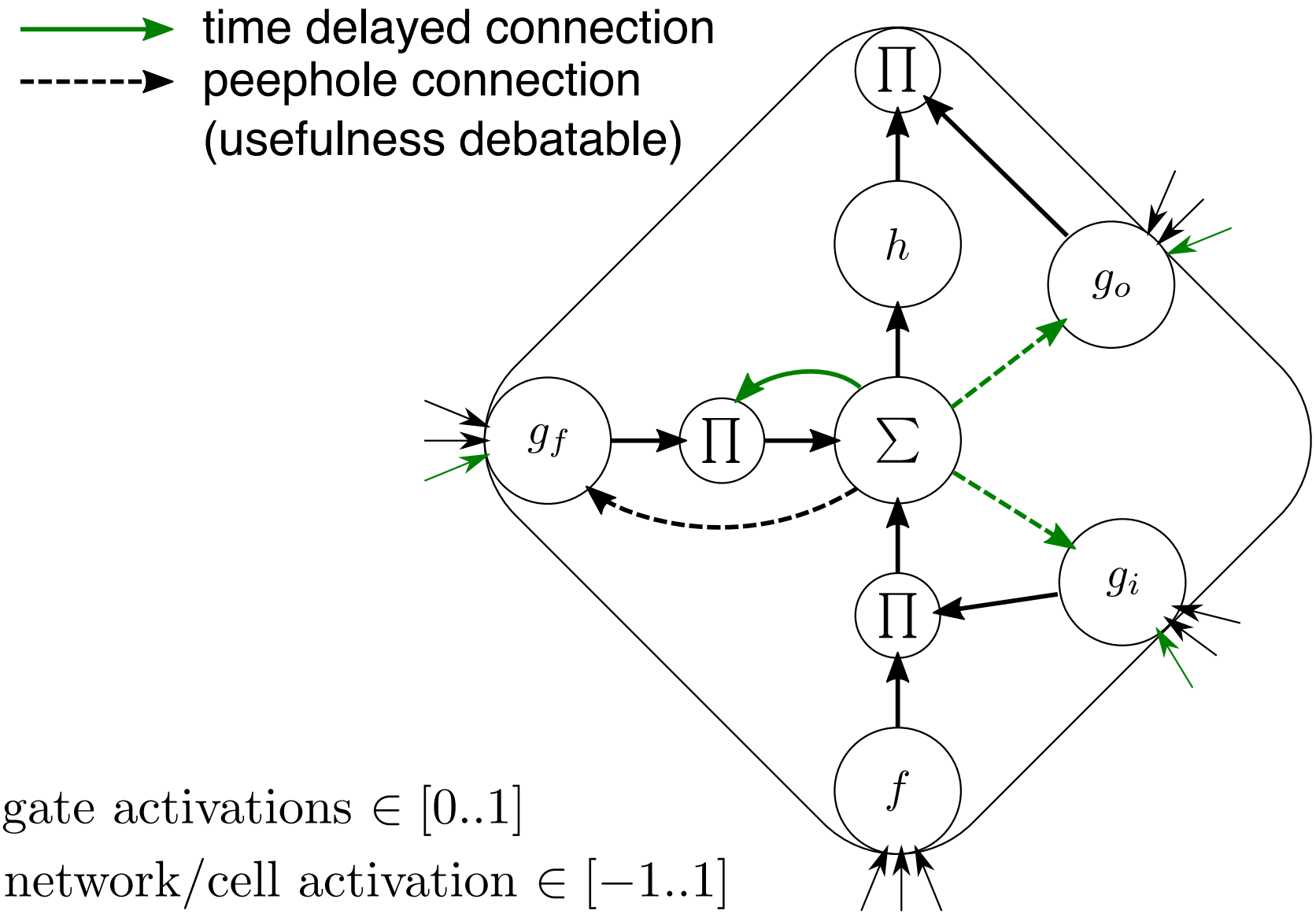
Constructing a differentiable memory cell

Let the controls be neurons



Constructing a differentiable memory cell

Final result

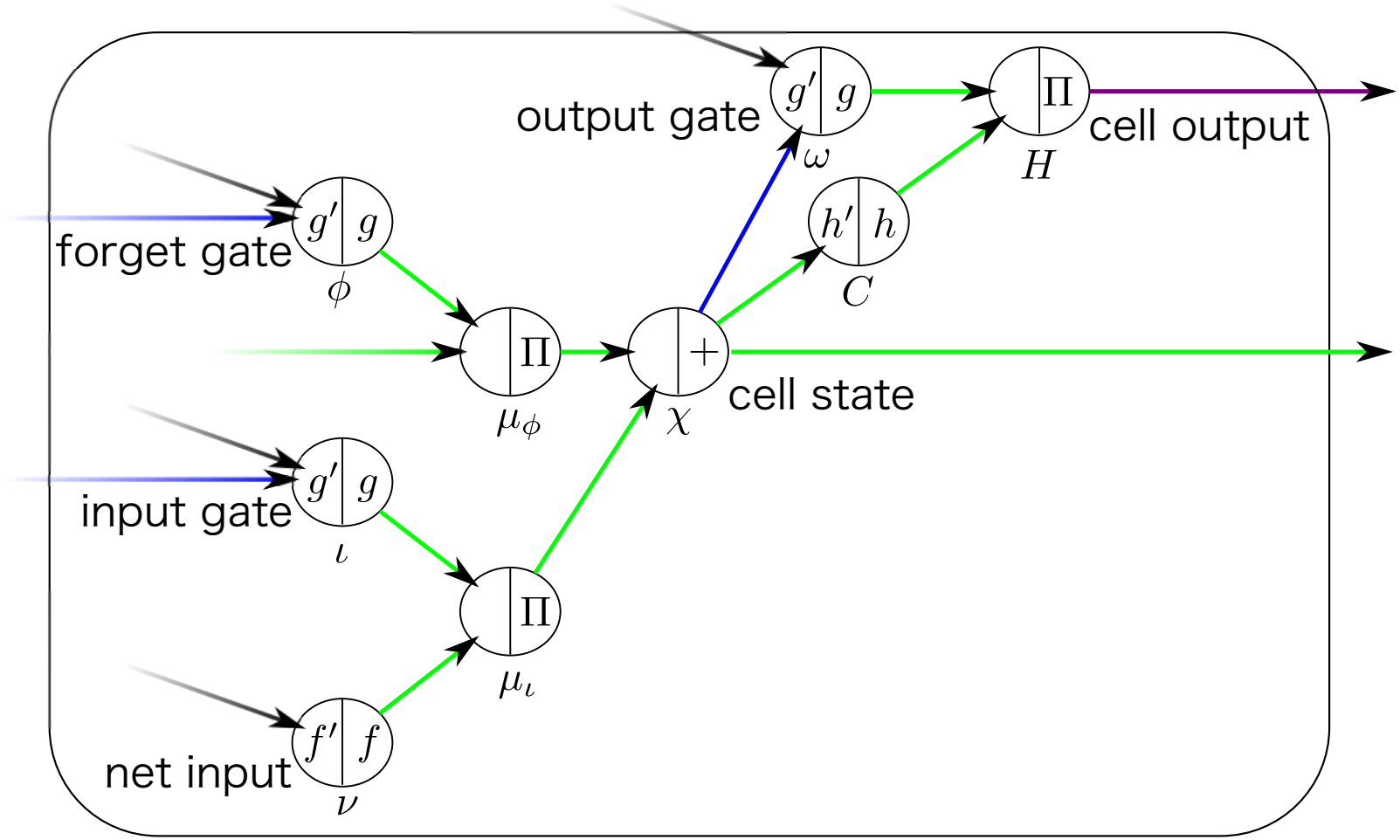


gate activations $\in [0..1]$

network/cell activation $\in [-1..1]$

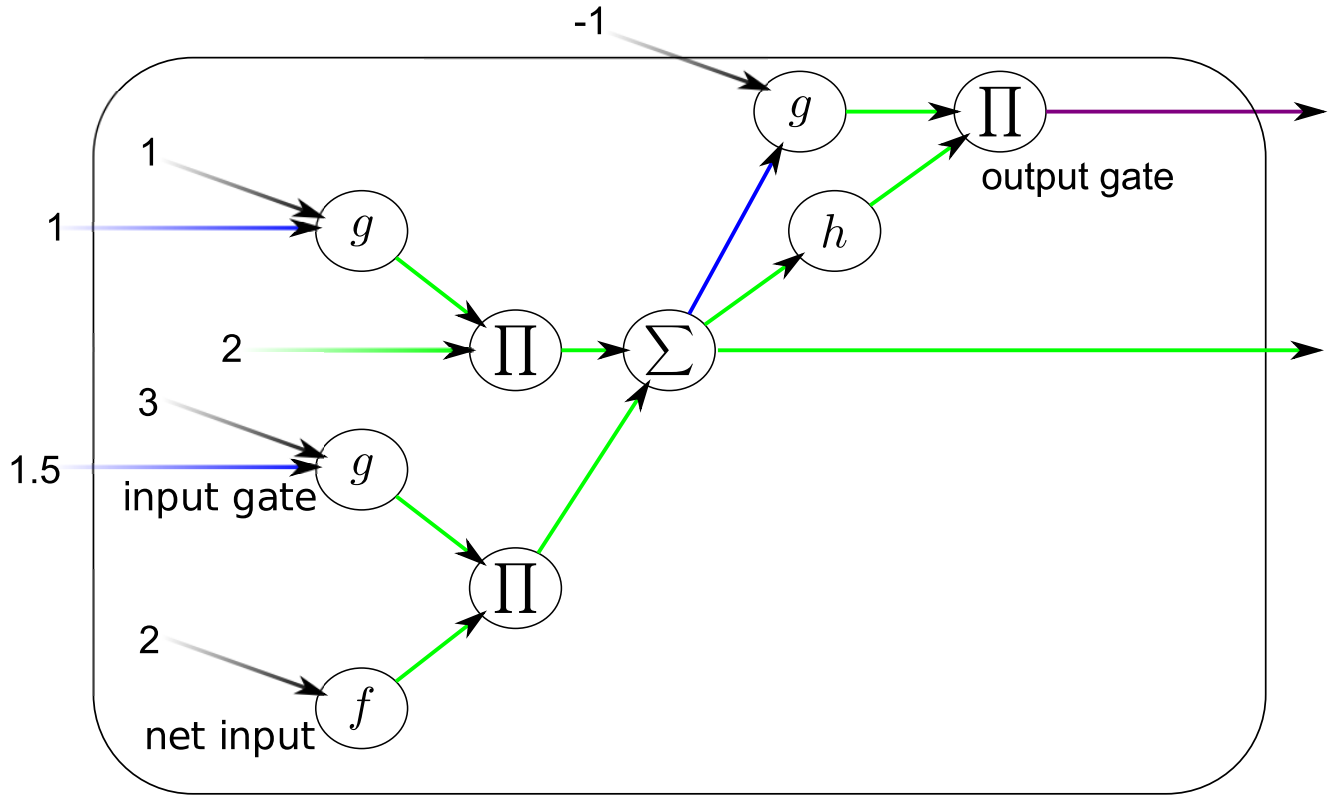
Constructing a differentiable memory cell

The final result



green lines are connection with a fixed weight = 1

Example



$$
\begin{aligned}
g(x) &= (1+\exp(-x))^{-1} \\
f(x) &= 2\,(1+\exp(-x))^{-1} - 1 \\
h(x) &= 2\,(1+\exp(-x))^{-1} - 1
\end{aligned}
$$