RECURRENT NEURAL NETWORKS (RNN)

DEEPLEA17EM

TEXT REPRESENTATION

- corpus - the given text we use for ML
- tokenization - split of the text to words
- stemming - converting everything to singular & removing affixations (eq going -> go, dogs -> dog)
- vocabulary - unique set of the stemmed tokens

The quick brown fox jumps over the lazy dog.

[The] [quick] [brown] [fox] [jumps] [over] [the] [lazy] [dog]

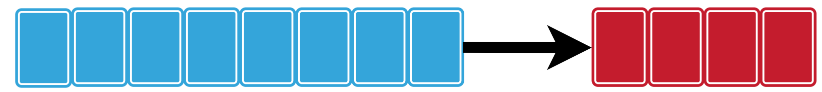
[The] [quick] [brown] [fox] [jump] [over] [the] [lazy] [dog]

[the] [quick] [brown] [fox] [jump] [over] [the] [lazy] [dog]

	dog
brown	0
dog	1
fox	0
jump	0
lazy	0
over	0
quick	0
the	0

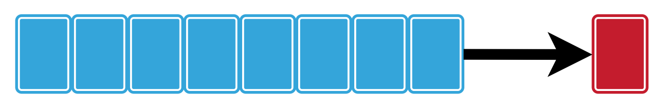
SEQUENCE AS INPUT OR OUTPUT

- **Speech recognition**
  - **Input: sequence of pressure values**
  - **Output: sequence of words**

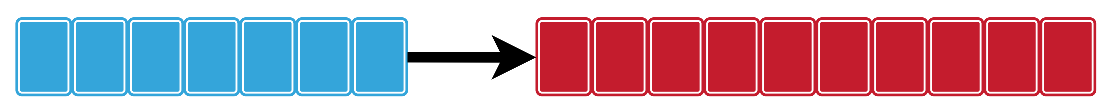


- **Music generation**
  - **Input: 0**
  - **Output sequence of notes**

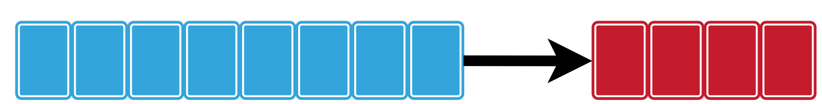
- **Sentiment classification**
  - **Input: sequence of words**
  - **Output: rating (1-5)**



- **Machine translation**
  - **Input: sequence of words**
  - **Output: sequence of words**

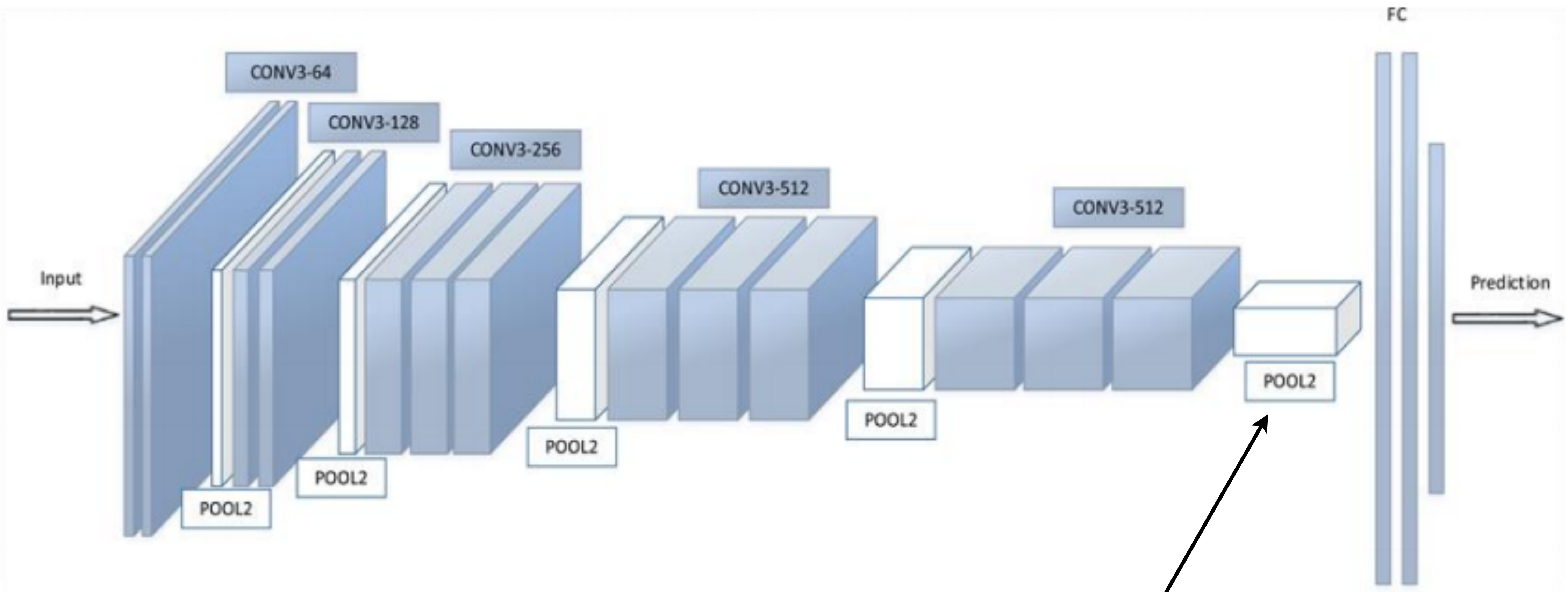


- **Video activity recognition, summarization, etc.:**
  - **Input: sequence of pictures**
  - **Output: labels, sequence of words, etc.**



CNNs lack this flexibility. For some cases you can hack around, sometimes you can't.

CNN INPUT-OUTPUT FLEXIBILITY



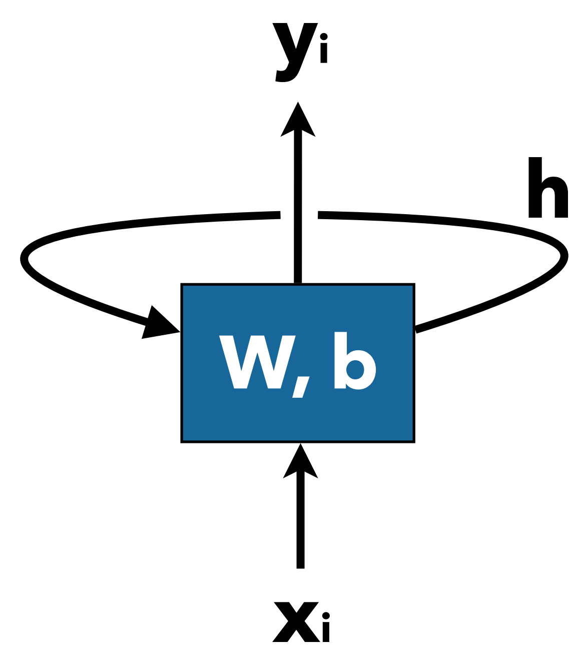
[http://file.scirp.org/Html/4-7800353_65406.htm]

globalmaxpooling instead of maxpooling opens some flexibility for the input image size

One can also slice the input sequence to fix sized chucks and assemble the prediction of the chunks.

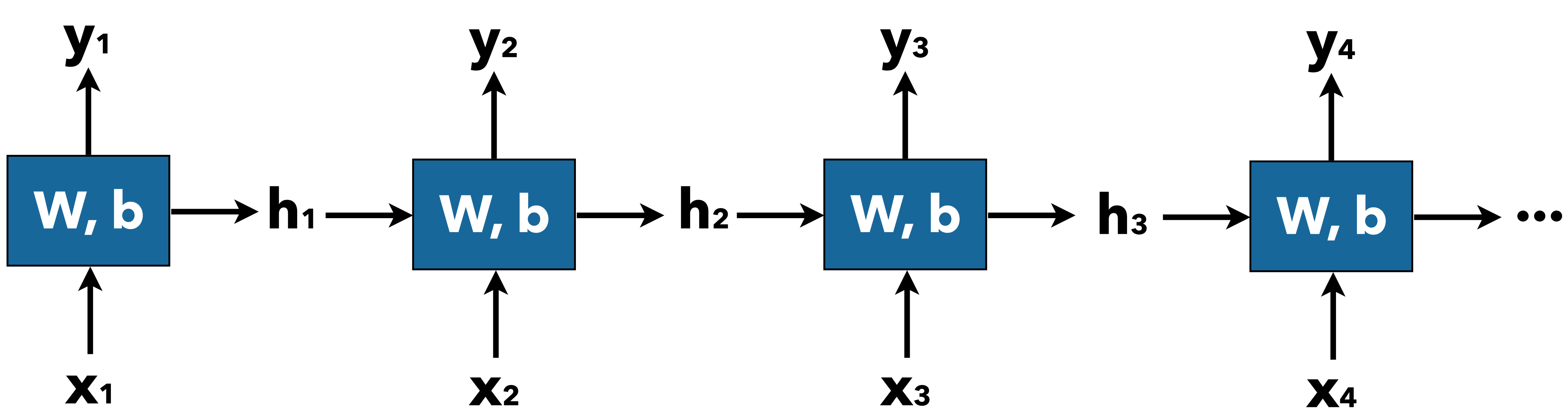
Requirements:

- flexible input sequence size (**x**)
- flexible output sequence size (**y**)
- some knowledge kept from previous state (**h**)

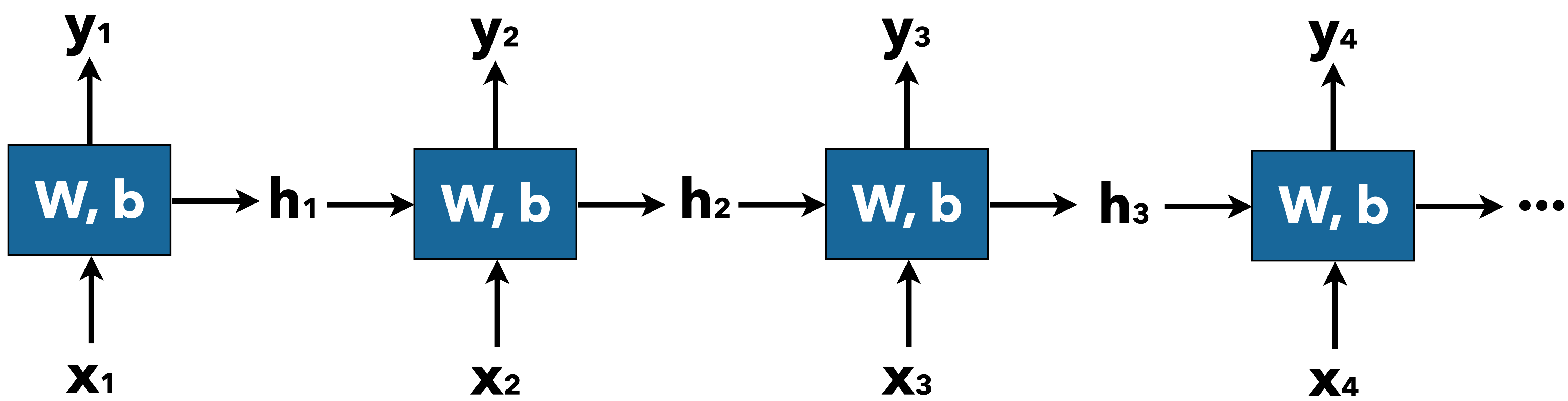


Two representation:

- compact
- roll-out



SIMPLEST RNN CELL



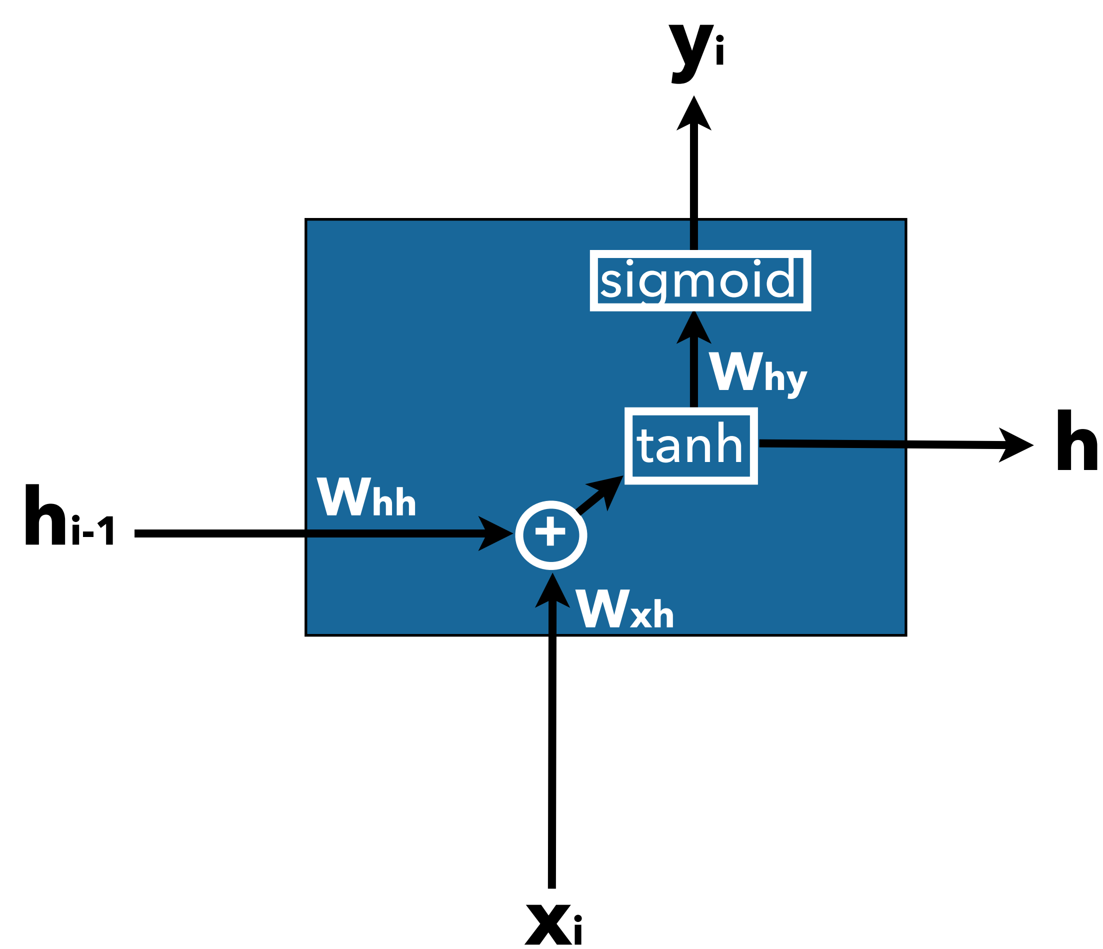
$\mathbf{h}_1 = g_h(W_{xh}\mathbf{X}_1 + W_{hh}\mathbf{h}_0 + b_h)$

$\mathbf{y}_1 = g_y(W_{hy}\, g_h(W_{xh}\mathbf{X}_1 + W_{hh}\mathbf{h}_0 + b_h) + b_y) = g_y(W_{hy}\mathbf{h}_1 + b_y)$

$\mathbf{h}_2 = g_h(W_{xh}\mathbf{X}_2 + W_{hh}\mathbf{h}_1 + b_h)$

$\mathbf{y}_2 = g_y(W_{hy}\, g_h(W_{xh}\mathbf{X}_2 + W_{hh}\mathbf{h}_1 + b_h) + b_y) = g_y(W_{hy}\mathbf{h}_2 + b_y)$

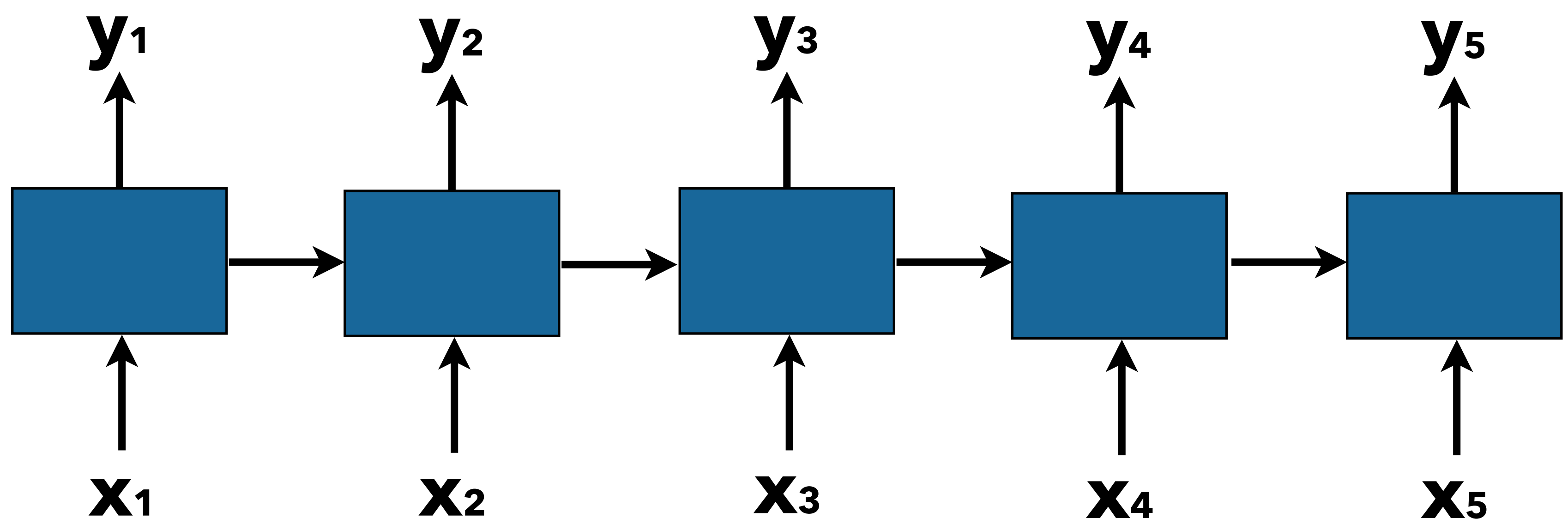
...



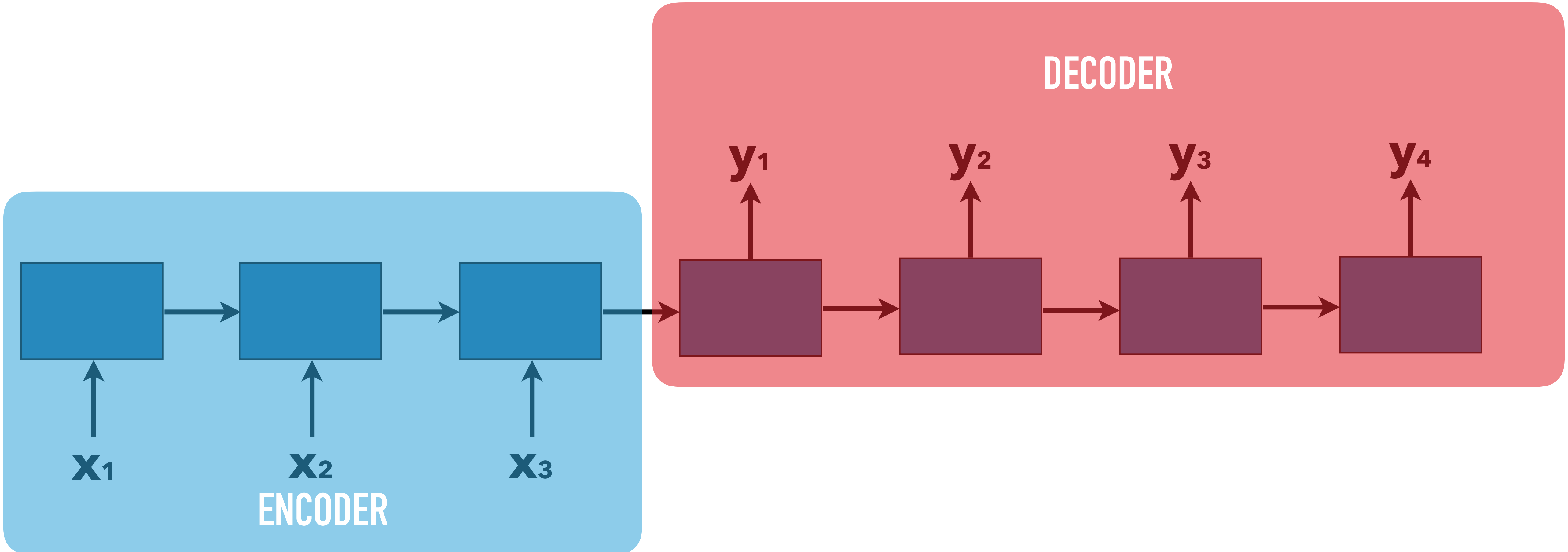
W_{xh} embedding matrix can be transferred from word2vec or from training an RNN on a large corpus

RNN ARCHITECTURES

many-to-many, input - output size is the same

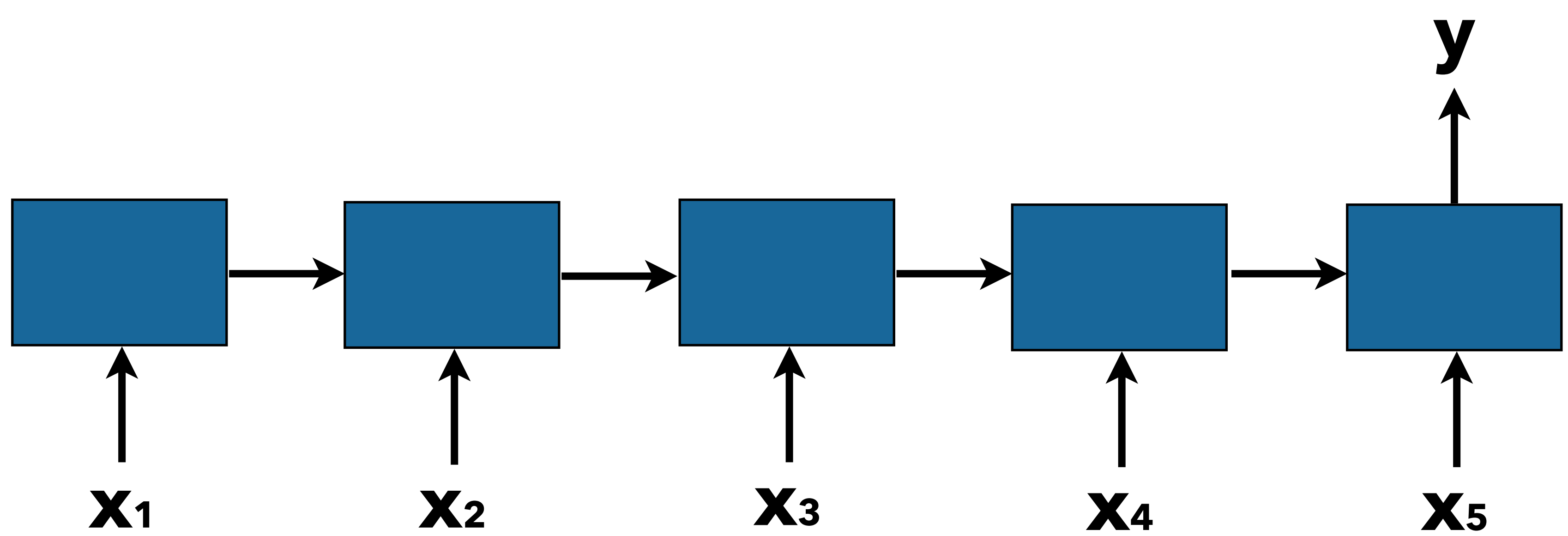


many-to-many, input - output size is not the same

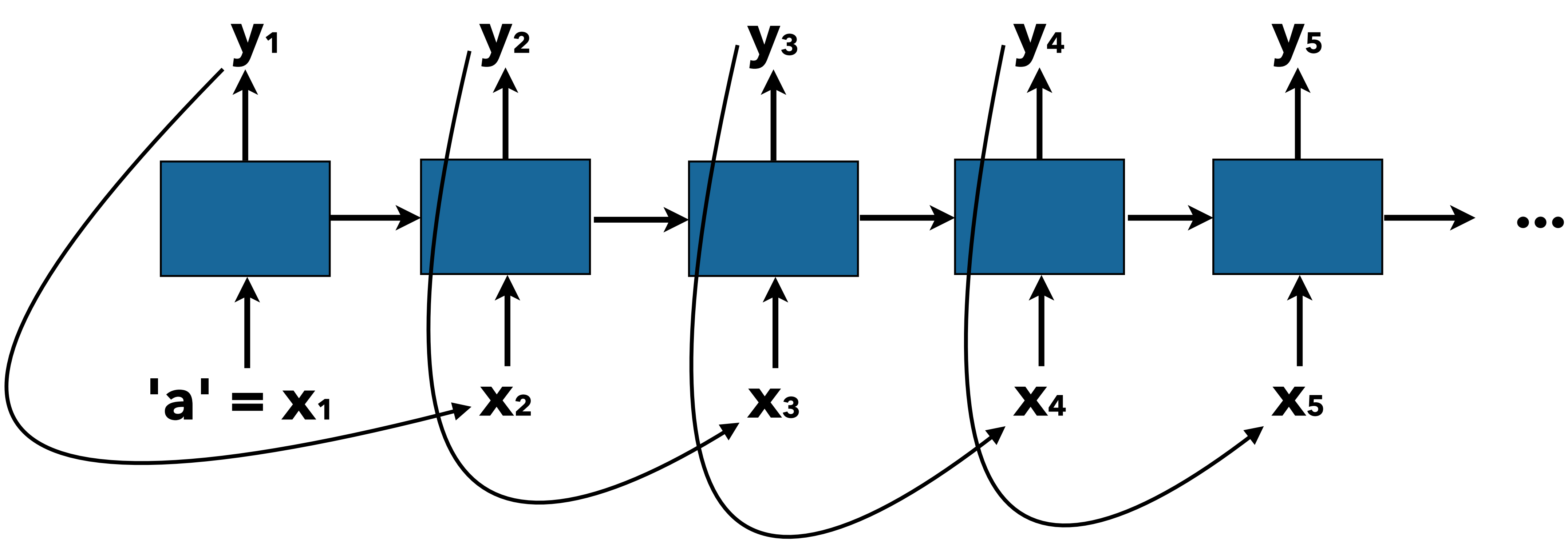


RNN ARCHITECTURES

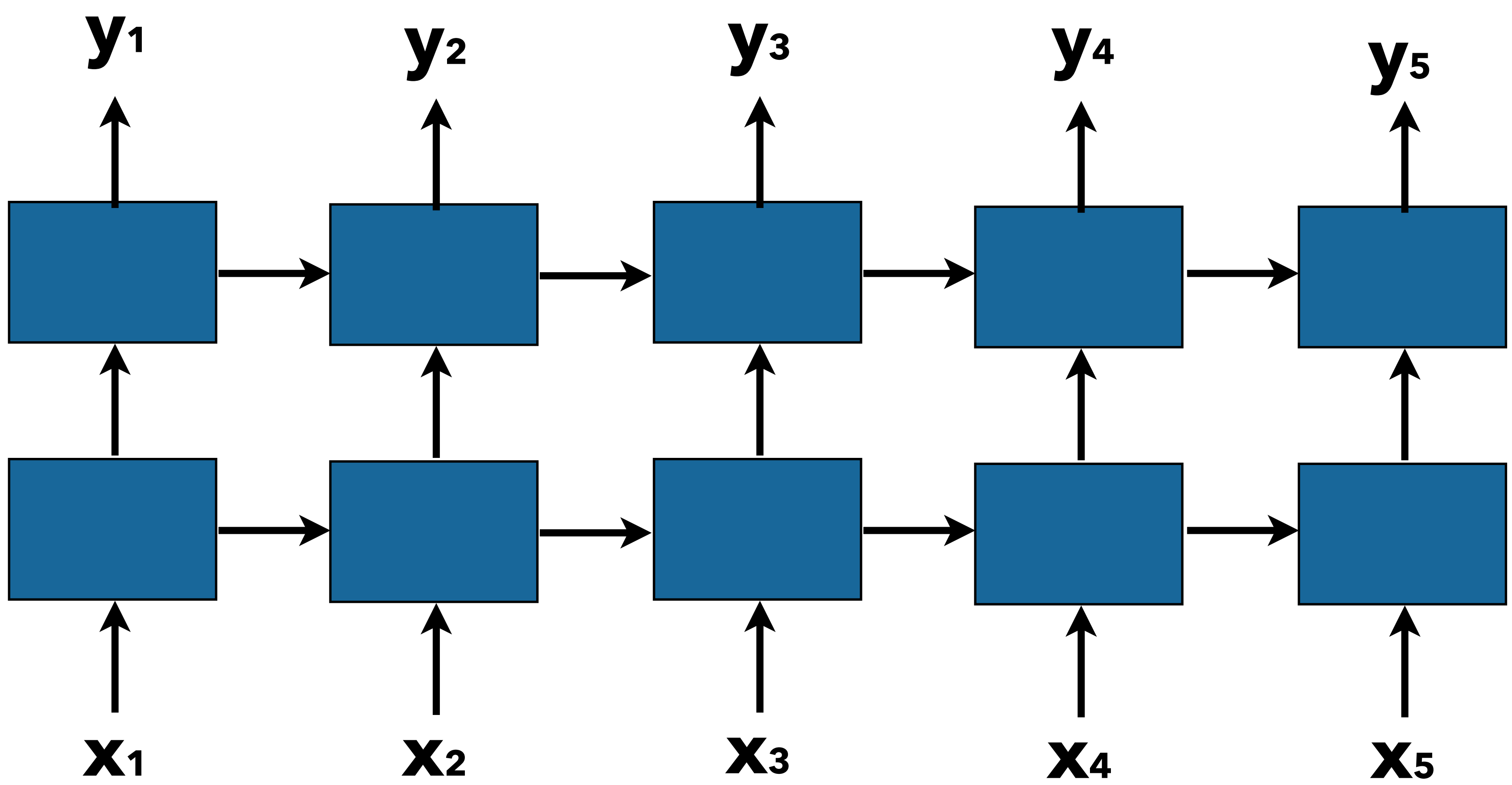
many-to-one

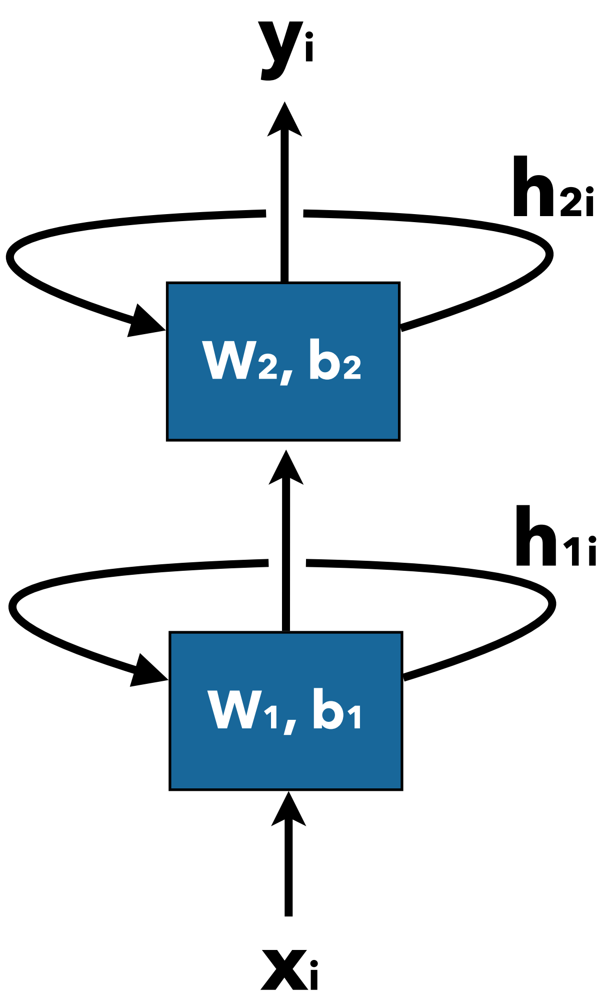


one-to-many, input is just a seed

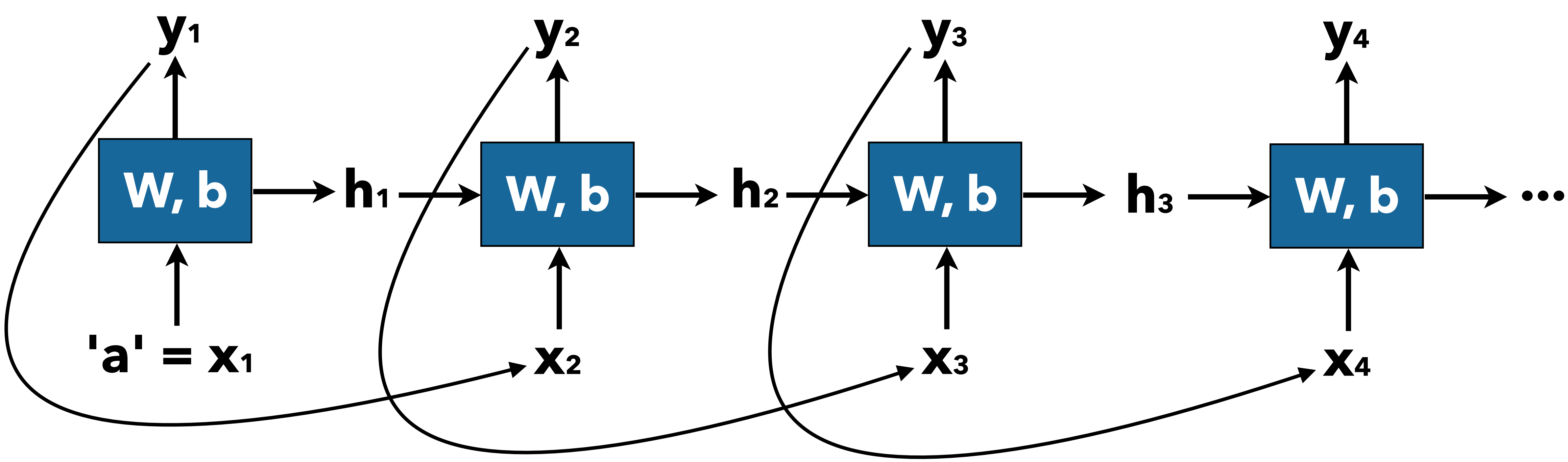


multi layer RNN





CHARACTER LEVEL RNN



~100 lines of Python code w/ only numpy: https://gist.github.com/karpathy/d4dee566867f8291f086

- Inputs are the characters (not the words!)
- Train time:
  - predict the next character in a text!
- Test time:
  - start from some seed
  - generate text
  - output from the previous step is the input in the actual step.
  - sampling the output as a probability distribution - to increase diversity - not to stuck in loops
    - otherwise it can happen that it can happen that it can happen that...

CHARACTER LEVEL RNN

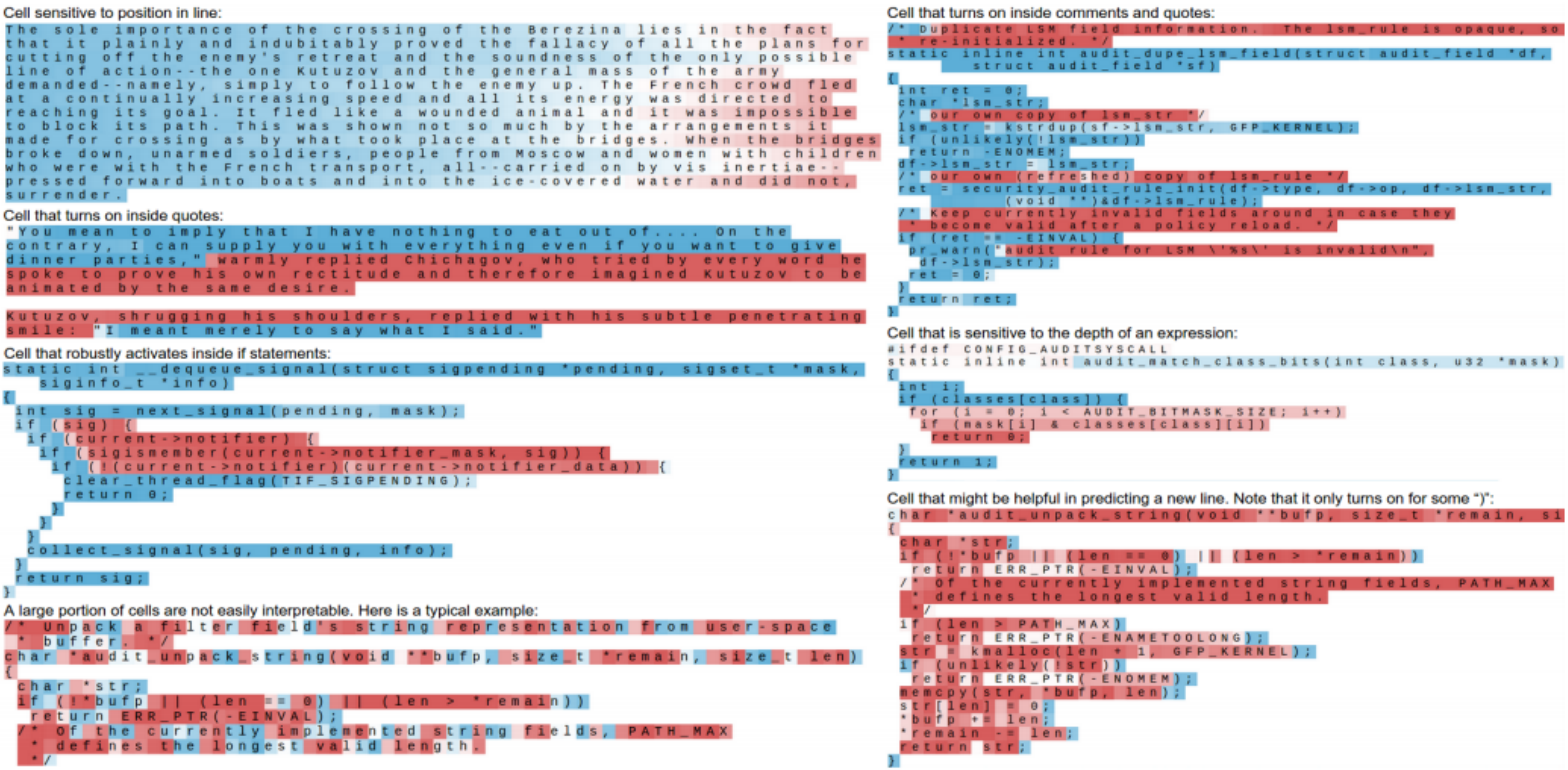
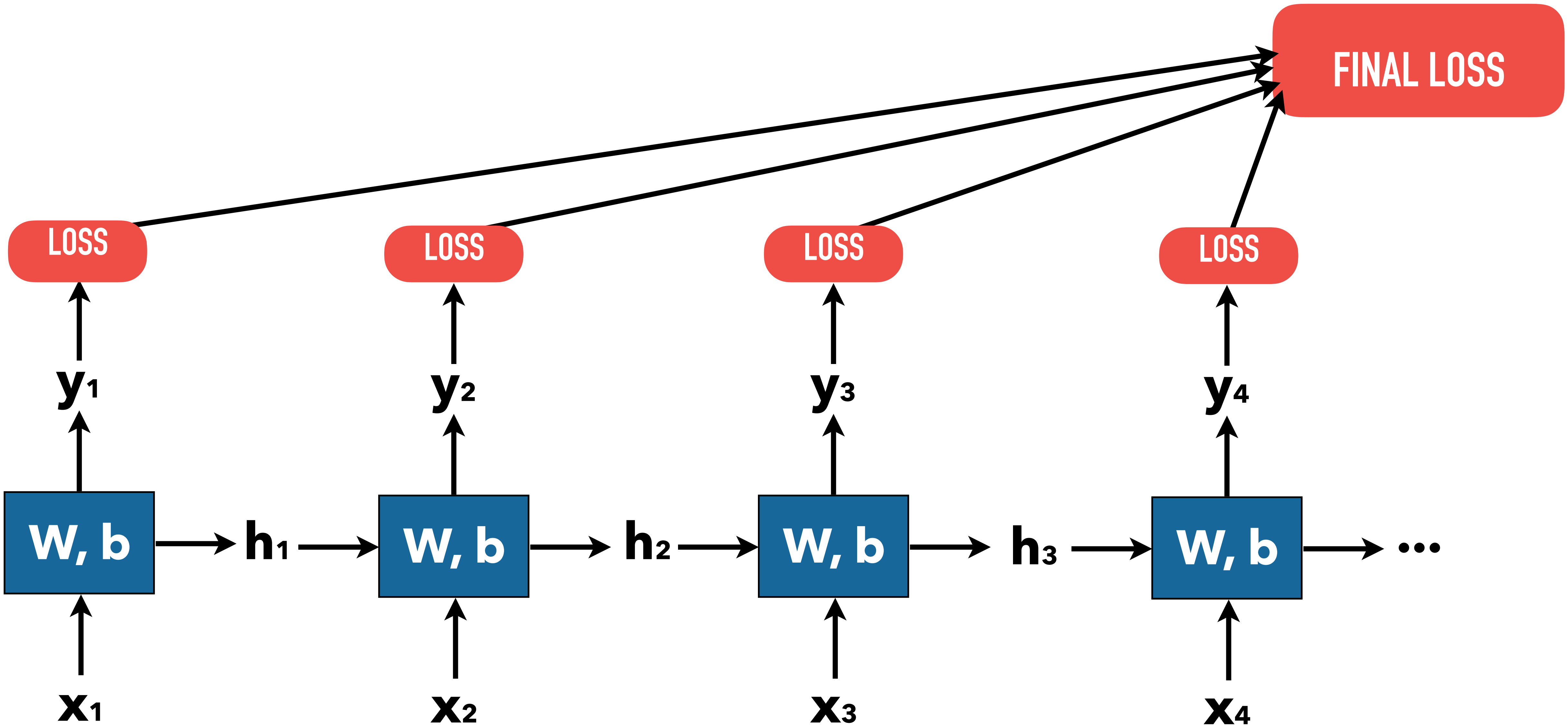


Figure 2: Several examples of cells with interpretable activations discovered in our best Linux Kernel and War and Peace LSTMs. Text color corresponds to $tanh(c)$, where -1 is red and +1 is blue.

Karpathy, Johnson, Fei-Fei: VISUALIZING AND UNDERSTANDING RECURRENT NETWORKS, 2016

PREDICT THE NEXT ELEMENT IN A SEQUENCE. HOW TO TRAIN SUCH AN RNN?

backpropagation through time

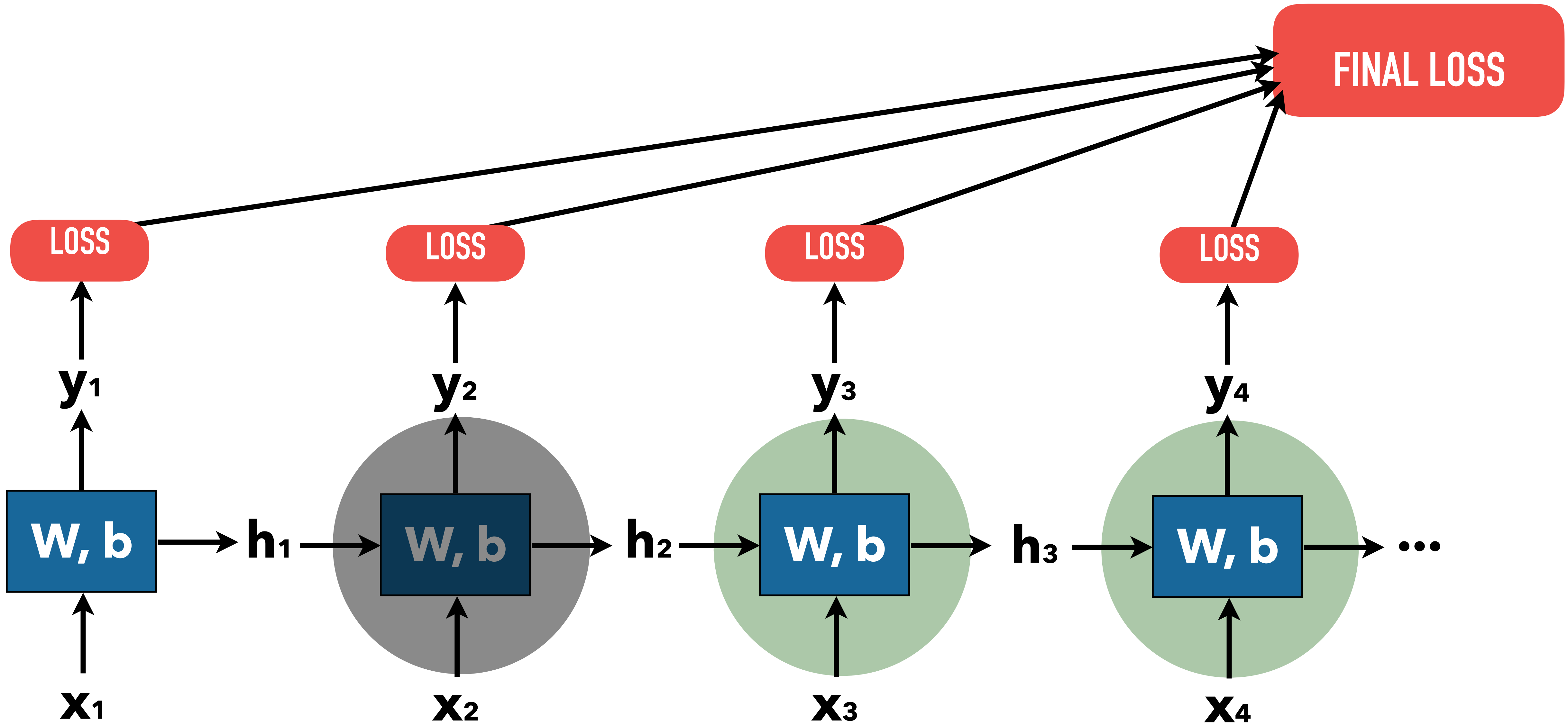


an update after each N step (similar to mini-batches):

- we do not want one update per the whole data -- super slow
- gradient vanishing / gradient exploding

PREDICT THE NEXT ELEMENT IN A SEQUENCE. HOW TO TRAIN SUCH AN RNN?

backpropagation through time



changing weights in the grey circle affects losses computed for the green ones!

PREDICT THE NEXT ELEMENT IN A SEQUENCE. HOW TO TRAIN SUCH AN RNN?

- Loss function:

$$L^{<t>}(\hat{y}^{<t>}, y^{<t>}) = -\sum_k y_k^{<t>} \log \hat{y}_k^{<t>}$$

$$L = \sum_{t=1}^{T_y} L^{<t>}(\hat{y}^{<t>}, y^{<t>})$$

- RNN: $h^{<t>} = g(W_{hh}h^{<t-1>} + W_{xh}\, x^{<t-1>} + b_h)$
- Let's say we know: $\frac{\partial L}{\partial h^{<t>}} = \frac{\partial L}{\partial y^{<t>}} \frac{\partial y^{<t>}}{\partial h^{<t>}}$
- We need: $\frac{\partial L}{\partial W_{hh}}, \frac{\partial L}{\partial W_{xh}}, \frac{\partial L}{\partial b_h}$ $\Leftarrow$ $\frac{\partial L^{<t>}}{\partial W_{hh}} = \frac{\partial L^{<t>}}{\partial h^{<t>}} \frac{\partial h^{<t>}}{\partial W_{hh}}$

$$\frac{\partial h^{<t>}}{\partial W_{hh}} \textit{ depends on } h^{<t-1>} \Rightarrow \text{Backprop trough time}$$

PREDICT THE NEXT ELEMENT IN A SEQUENCE. HOW TO TRAIN SUCH AN RNN?

- Loss function:

$$L^{<t>}(\hat{y}^{<t>}, y^{<t>}) = -\sum_k y_k^{<t>} \log \hat{y}_k^{<t>}$$

$$L = \sum_{t=1}^{T_y} L^{<t>}(\hat{y}^{<t>}, y^{<t>})$$

- RNN: $h^{<t>} = g(W_{hh} h^{<t-1>} + W_{xh}\, x^{<t-1>} + b_h)$

- Let's say we know: $\frac{\partial L}{\partial h^{<t>}} = \frac{\partial L}{\partial y^{<t>}} \frac{\partial y^{<t>}}{\partial h^{<t>}}$

- We need: $\frac{\partial L}{\partial W_{hh}}, \frac{\partial L}{\partial W_{xh}}, \frac{\partial L}{\partial b_h}$ $\Leftarrow$ $\frac{\partial L^{<t>}}{\partial W_{hh}} = \frac{\partial L^{<t>}}{\partial h^{<t>}} \frac{\partial h^{<t>}}{\partial W_{hh}}$

$\frac{\partial h^{<t>}}{\partial W_{hh}}$ *depends on* $h^{<t-1>}$ $\Rightarrow$ Backprop trough time

- gradient vanishing / gradient exploding --> gradient cliping

  0.99 to the power of 10000 is 2e-44 & 1.01 to the 10000 is 2e43

IMAGE CAPTIONING - THE TASK

COCO
Common Objects in Context

info@cocodataset.org

Home People Dataset Tasks Evaluate

COCO 2015 Image Captioning Task

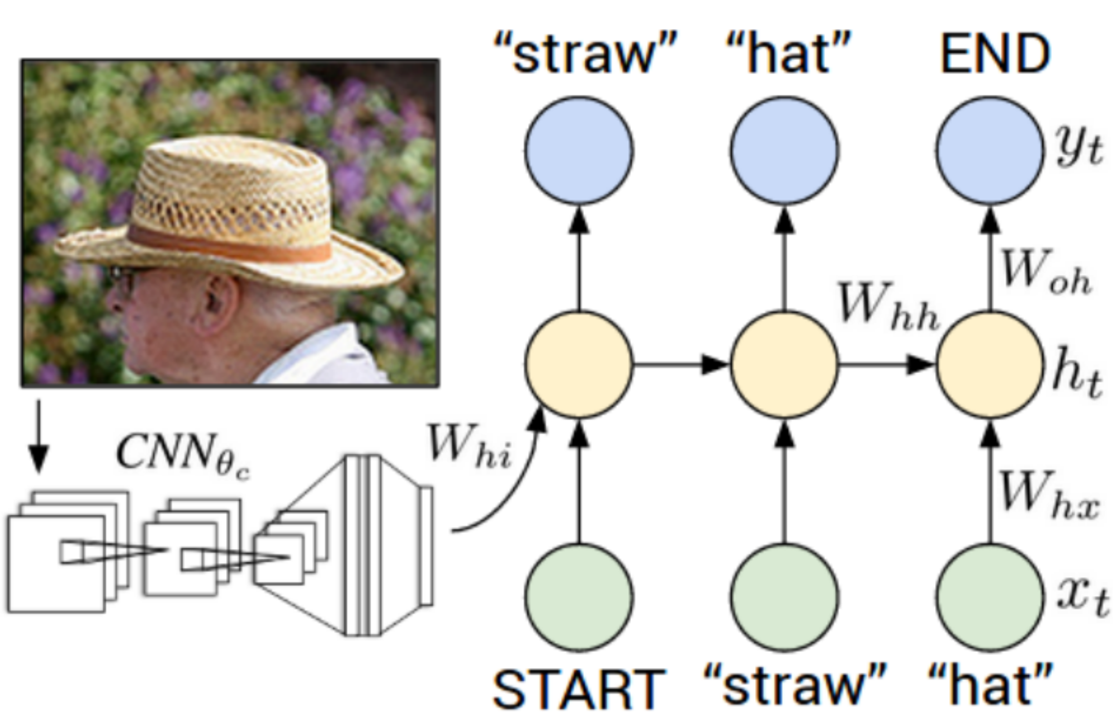
The man at bat readies to swing at the pitch while the umpire looks on.

A large bus sitting next to a very tall building.

http://cocodataset.org/#captions-2015

IMAGE CAPTIONING

- run a pre-trained CNN on the image and use it as a feature extractor
- feed the extracted features to the RNN



man in black shirt is playing guitar.

construction worker in orange safety vest is working on road.

Karpathy, Fei-Fei: Deep Visual-Semantic Alignments for Generating Image Descriptions, 2015

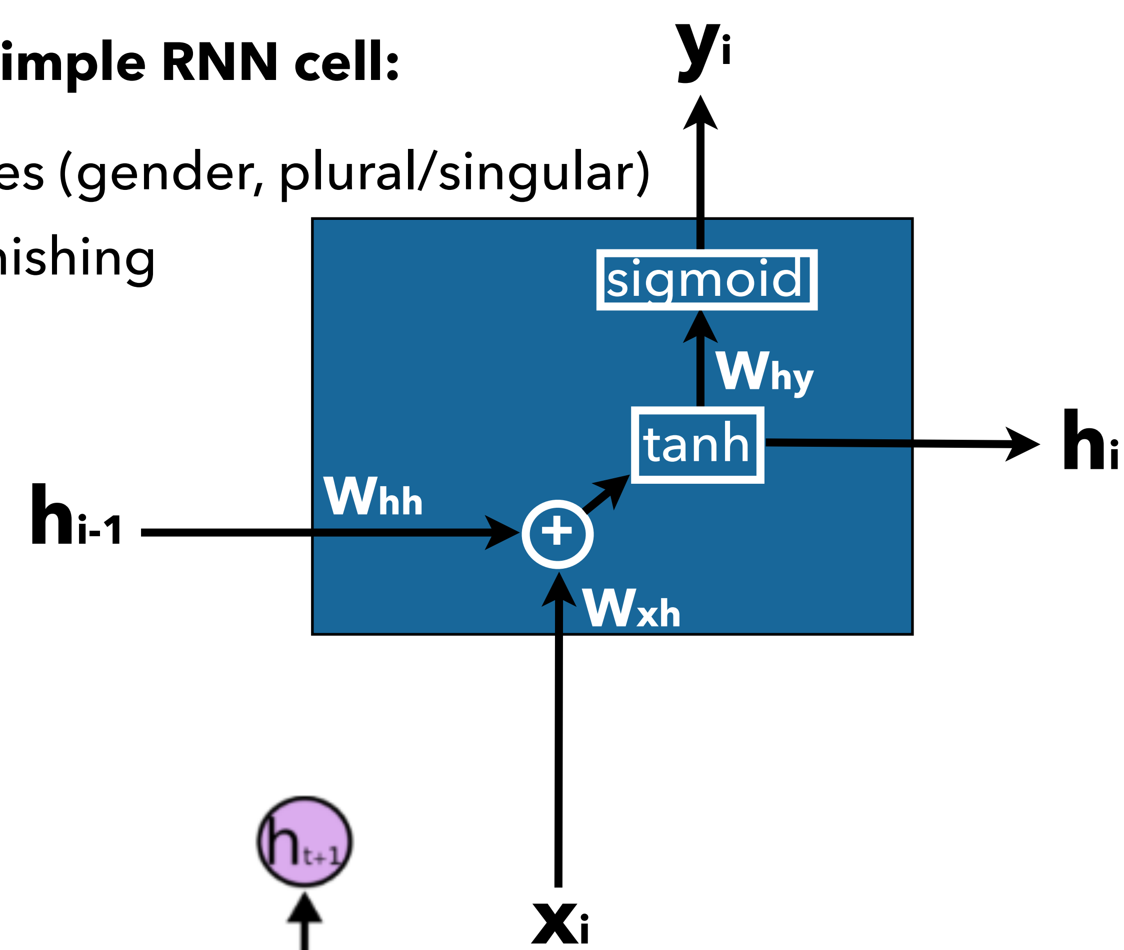
LSTM

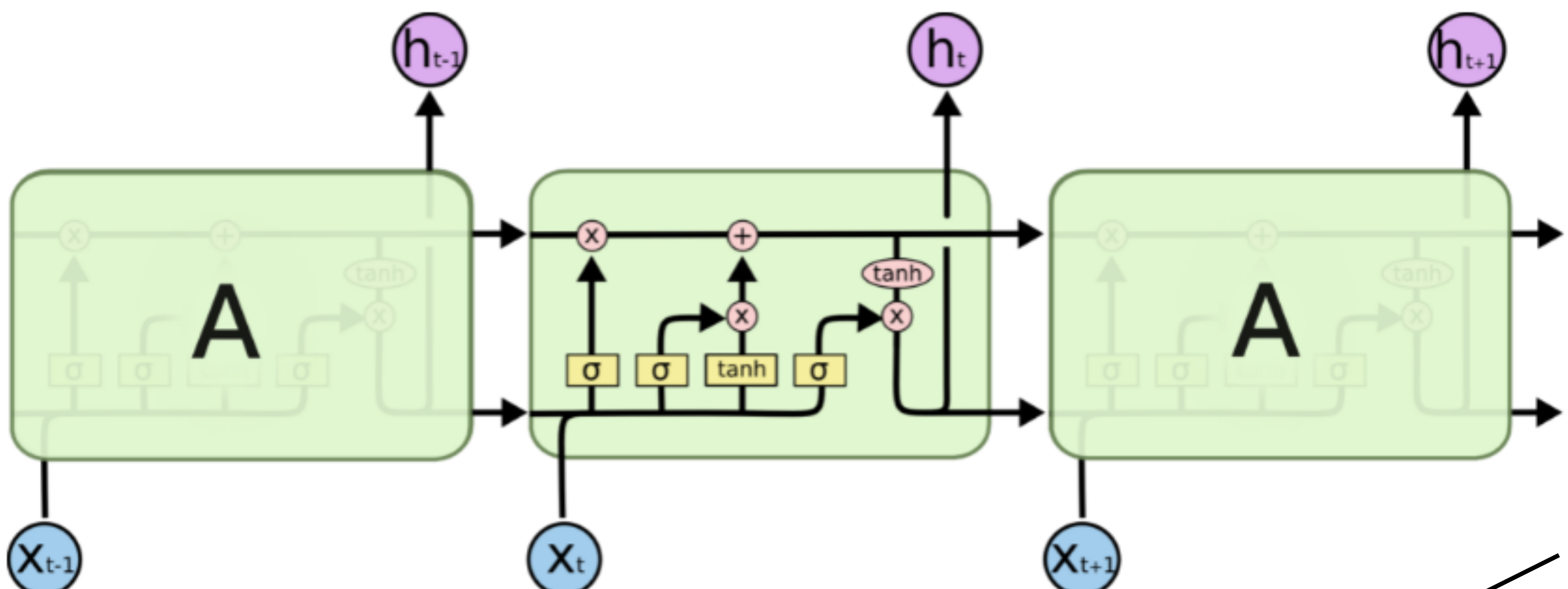
[Hochreiter, 1997]

Main problems with simple RNN cell:

long term dependencies (gender, plural/singular)

gradient explosion/vanishing





cool description & figures

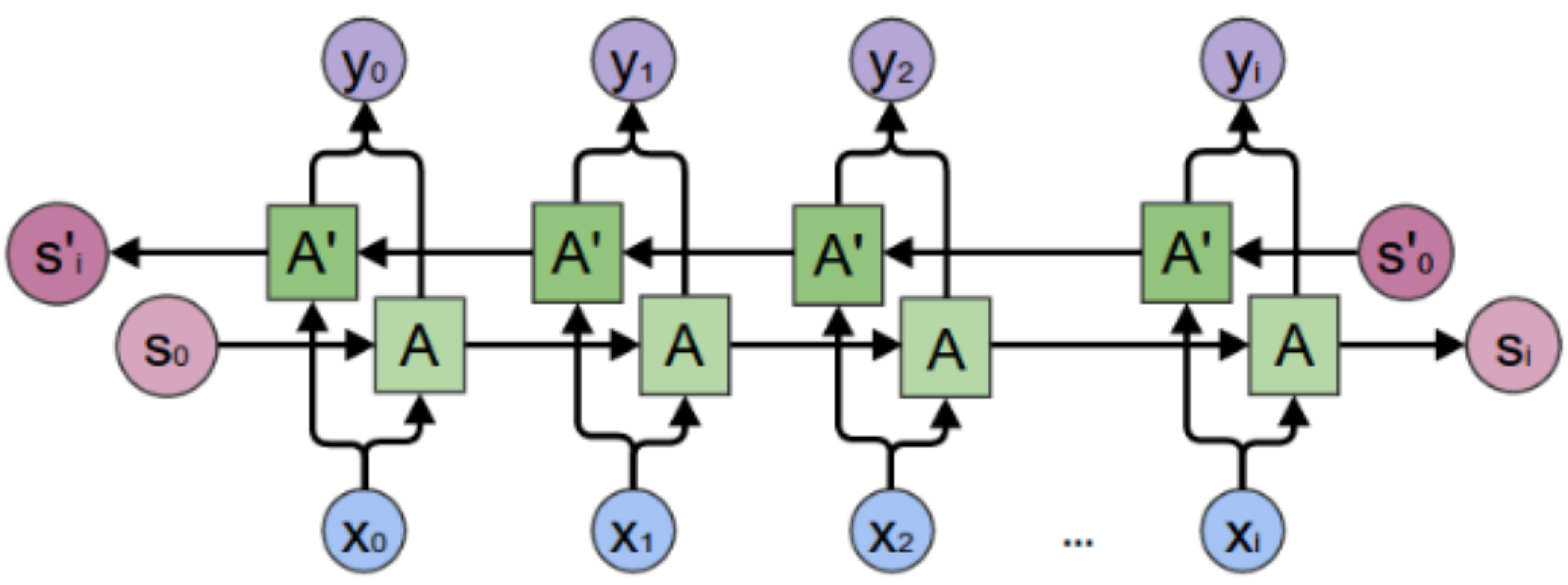
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

BIDIRECTIONAL MODELS

In many cases we do not want to strictly restrict ourself to the past.

- speech to text - conversion of a word knowing the pressure values after the word
- filling ____ in a sentence
- machine translation - usually you do not want to translate sentence on the fly word-by-word

2 RNN/LSTM: going to different direction



http://colah.github.io/posts/2015-09-NN-Types-FP/

OTHER RNN CELL ARCHITECTURES?

An Empirical Exploration of Recurrent Network Architectures

2015

Rafal Jozefowicz RAFALJ@GOOGLE.COM
Google Inc.

Wojciech Zaremba WOJ.ZAREMBA@GMAIL.COM
New York University, Facebook[1]

Ilya Sutskever ILYASU@GOOGLE.COM
Google Inc.

Abstract

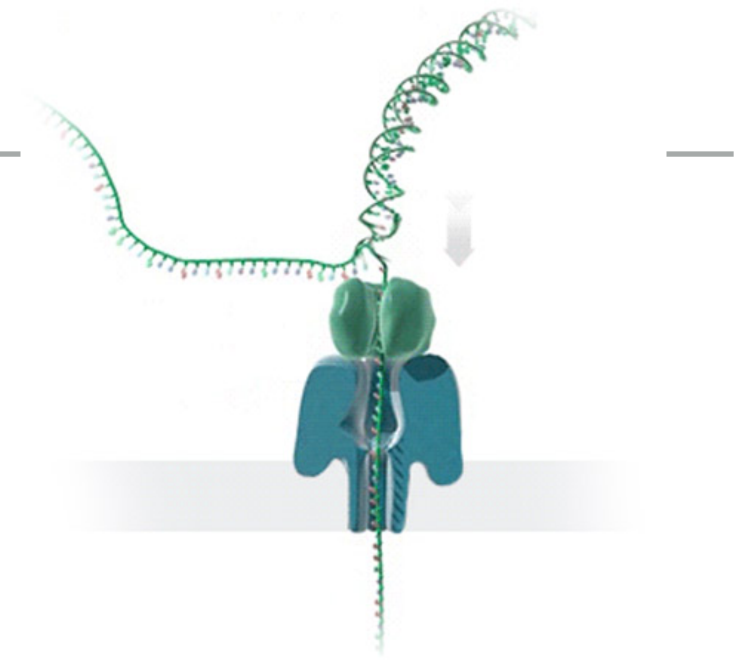
The Recurrent Neural Network (RNN) is an extremely powerful sequence model that is often difficult to train. The Long Short-Term Memory (LSTM) is a specific RNN architecture whose design makes it much easier to train. While wildly successful in practice, the LSTM's architecture appears to be ad-hoc so it is not clear if it is optimal, and the significance of its individual components is unclear.

In this work, we aim to determine whether the LSTM architecture is optimal or whether much better architectures exist. We conducted a thorough architecture search where we evaluated over ten thousand different RNN architectures, and identified an architecture that outperforms both the LSTM and the recently-introduced Gated Recurrent Unit (GRU) on some but not all tasks. We found that adding a bias of 1 to the LSTM's forget gate closes the gap between the LSTM and the GRU.

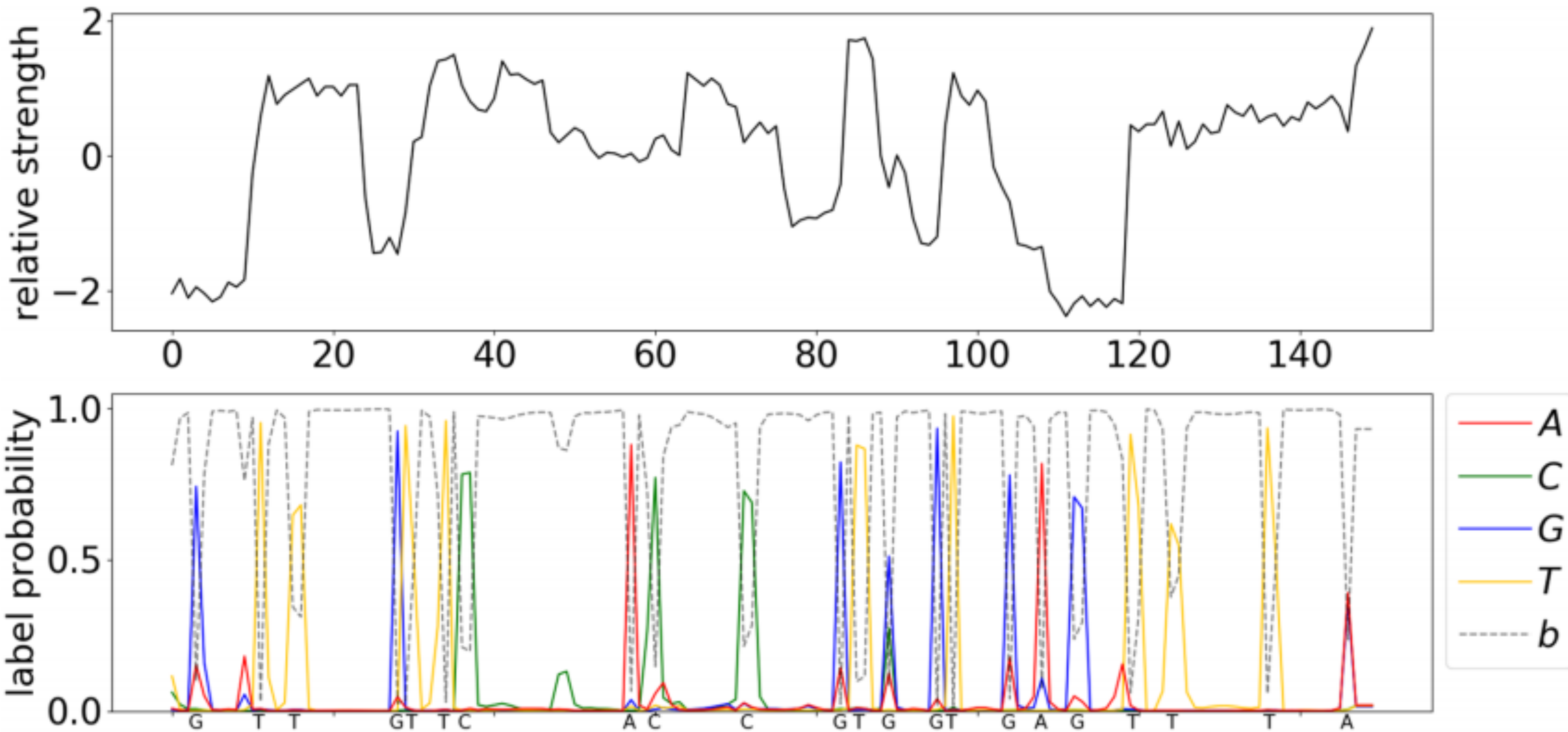
We didn't discuss GRUs, see:

Cho et al: Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation, 2014

SCIENTIFIC RELATION - NANOPORE SEQUENCING

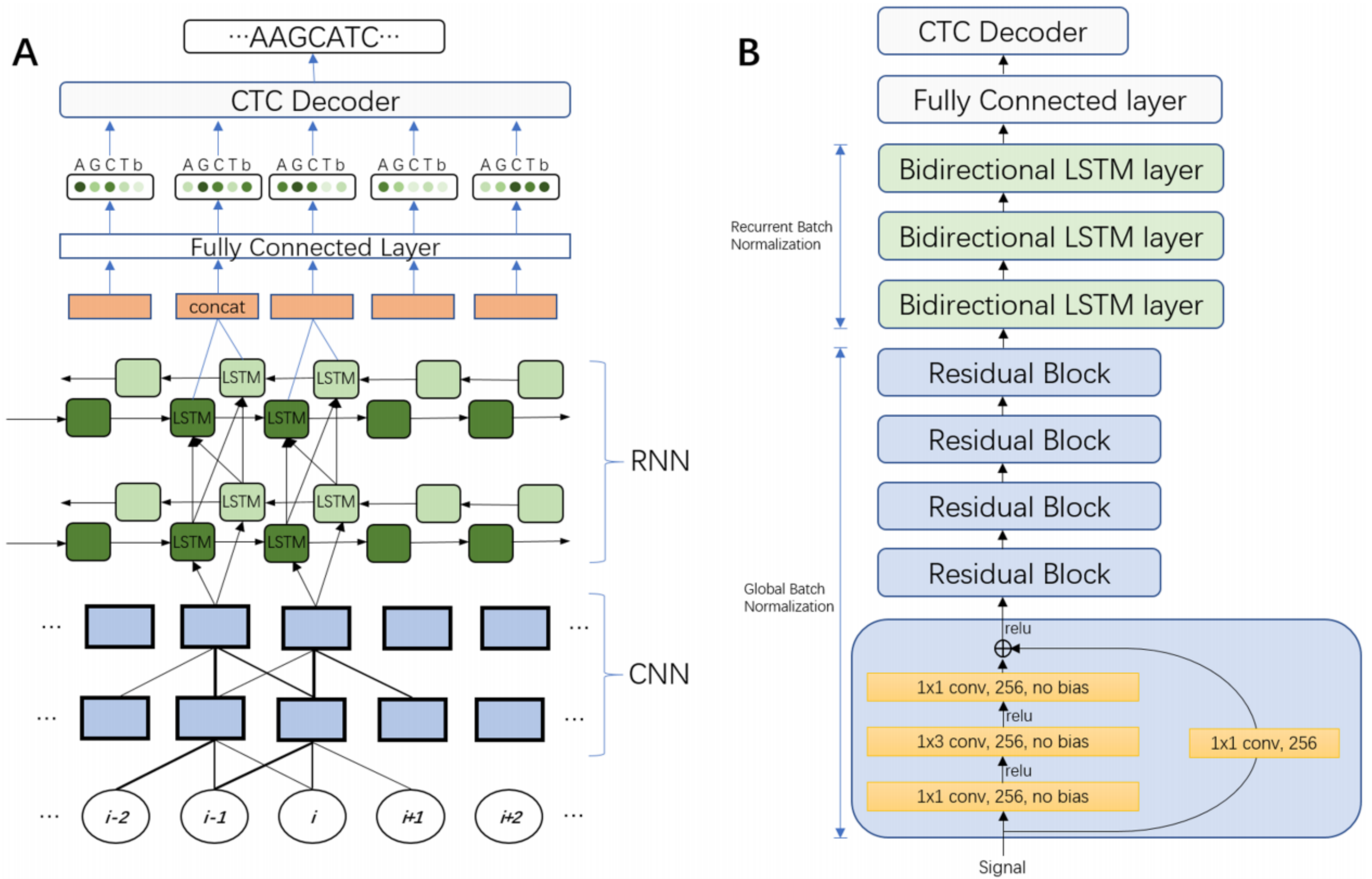


https://nanoporetech.com/learn-more



Teng et al: Chiron: translating nanopore raw signal directly into nucleotide sequence using deep learning, 2018

SCIENTIFIC RELATION - NANOPORE SEQUENCING



Teng et al: Chiron: translating nanopore raw signal directly into nucleotide sequence using deep learning, 2018

DEMO notebook