Recurrent Neural Networks (RNNs)

Alvaro Soto

Computer Science Department, PUC

RNNs: neural networks models for processing sequential data (Rumelhart et al., 1986a)

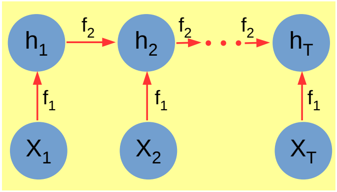
Sequential Data?

Our big world of data is full of sequences like text, audio, and videos. So RNNs are highly useful.

Let's use a RNN to encode a sequence: $X = \{x_1, x_2, \ldots, x_T\}$

x_t: input vector encoding input sequence at time t

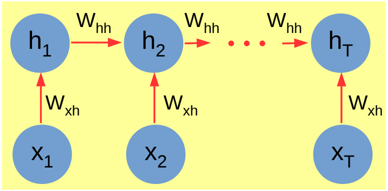
A RNN uses a recurrent encoding: $h_t = f(h_{t-1}, x_t)$



- h_t: hidden (latten/internal) vector at time t encoding history of input sequence until time t.
- f_1 and f_2 : parametric functions adjusted during learning. These are the key functions to model the input sequence. Usually f_1 and f_2 are given by MLPs (single layer NNs).
- t: sequence step, it usually means temporal steps, but it can represent other type of relation (spatial, ranking, etc).

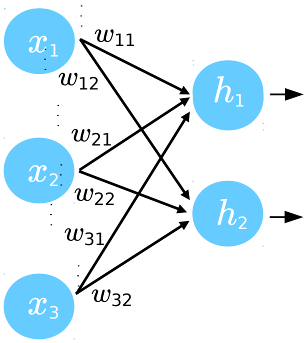
Common Configuration: Lineal models and sigmoid activations

$$h_t = \sigma(\, W_{hh}\; h_{t-1} + W_{xh}\; x_t)$$



$$x \in \mathbb{R}^{d_x},\, h \in \mathbb{R}^{d_h},\; W_{xh} \in \mathbb{R}^{d_h \times d_x},\; W_{hh} \in \mathbb{R}^{d_h \times d_h}.$$

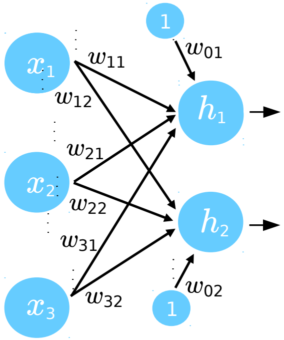
Lineal models and sigmoid activations



$$\begin{pmatrix} h_1 \\ h_2 \end{pmatrix} = \sigma \begin{pmatrix} w_{11} & w_{21} & w_{31} \\ w_{12} & w_{22} & w_{32} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

$$\begin{pmatrix} h_1 \\ h_2 \end{pmatrix} = \sigma \left(W \, X \right)$$

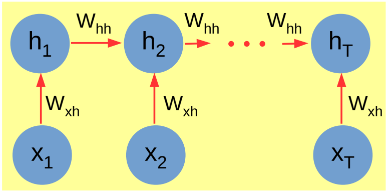
Lineal models and sigmoid activations

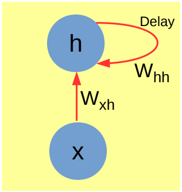


$$\begin{pmatrix} h_1 \\ h_2 \end{pmatrix} = \sigma \begin{pmatrix} w_{11} & w_{21} & w_{31} & w_{01} \\ w_{12} & w_{22} & w_{32} & w_{02} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} h_1 \\ h_2 \end{pmatrix} = \sigma \left(W \; X \right)$$

A recurrent encoding: $h_t = f(h_{t-1}, x_t)$

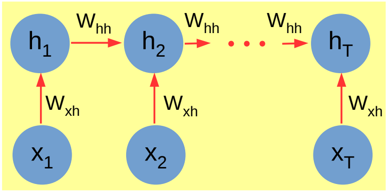




A recurrent application of models W_{xh} and W_{hh}

Common Configuration: Lineal models and sigmoid activations

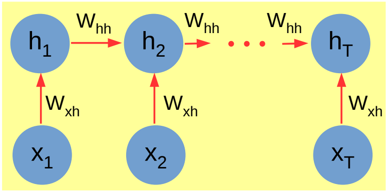
$$h_t = \sigma(W_{hh}\ h_{t-1} + W_{xh}\ x_t)$$



- Interestingly, a RNN generates a deep architecture without a substantial increase in the number of parameters, why?.
- Great!, by using a recurrent layer, we can obtain longer data dependencies while controlling model capacity.

RNNs are sequential memory devices

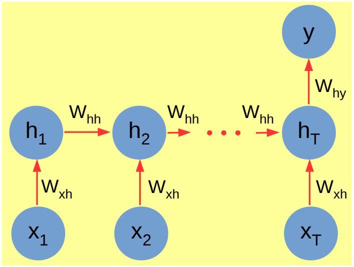
$$h_t = \sigma(\, W_{hh}\; h_{t-1} + W_{xh}\; x_t)$$



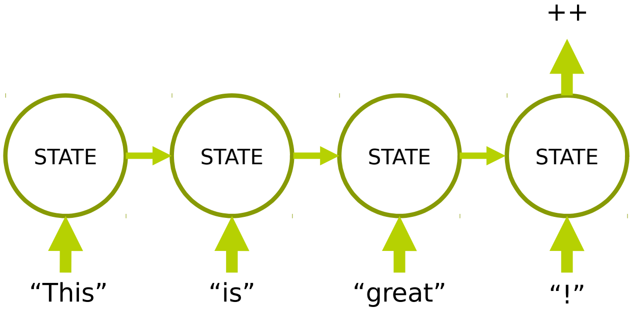
- The network learns to use the hidden state h_t as a lossy summary of the inputs until step t.
- This provides a mechanism to learn sequence order (similar to convolution masks in CNNs) (key property 1).

Most applications need to incorporate the coding of an output

$$h_t = \sigma(\, W_{hh}\; h_{t-1} + W_{xh}\; x_t)$$
$$y = \sigma(\, W_{hy}\; h_T)$$

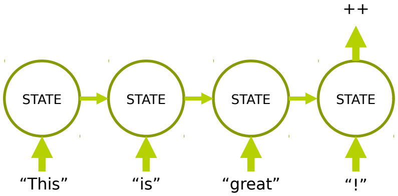


Application: Sentiment Analysis in Text



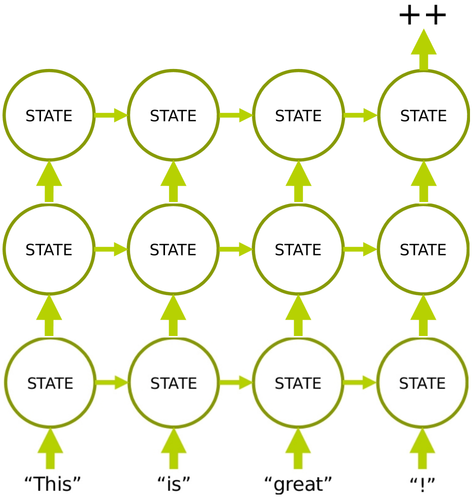
What is the math behind this model?

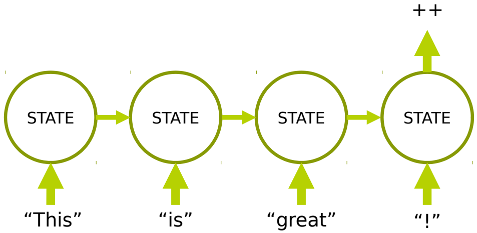
Let's see a possible implementation



- How can we implement the initial encoding (embedding) of the input words?
- How can we implement the encoding of the output?
- What dimensionality can we use for the internal state of the RNN?
- How can we manage sequence length (differente sentences have different lenght)?
- Can we implement a deeper model?

A deeper RNN


++
STATE
STATE
STATE
STATE
STATE
STATE
STATE
STATE
STATE
STATE
STATE
STATE
“This”
“is”
“great”
“!”

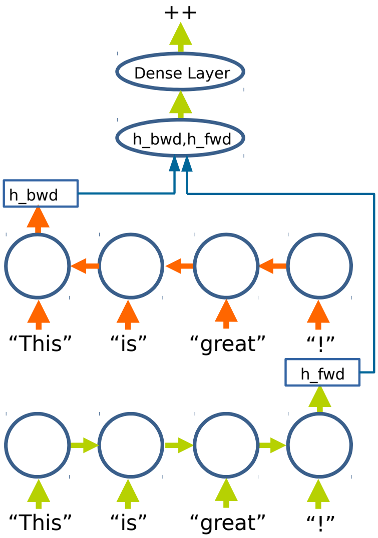


Each time step t, the hidden state h_t models the history of the sequence.

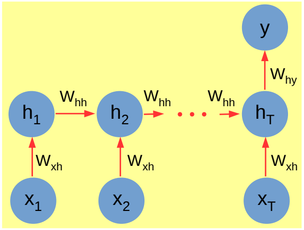
What about the future?

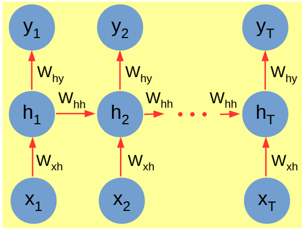
Answer: One can use a bidirectional RNN.

Sentiment Analysis Using a Bidirectional RNN



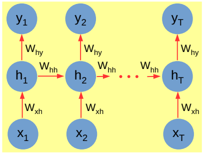
Depending of the application, we can define several configurations for the RNN

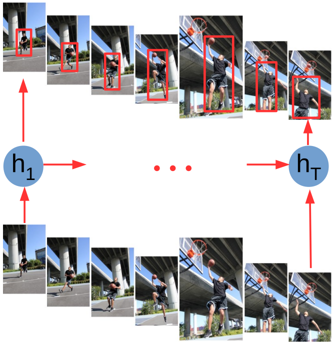




- Notice that this configuration leads to a RNN that maps an input sequence to an output sequence of the same length.
- However, RNNs are very flexible and they can accomodate to model sequences of different size.

Example: Target tracking in video, Seq2Seq decoding



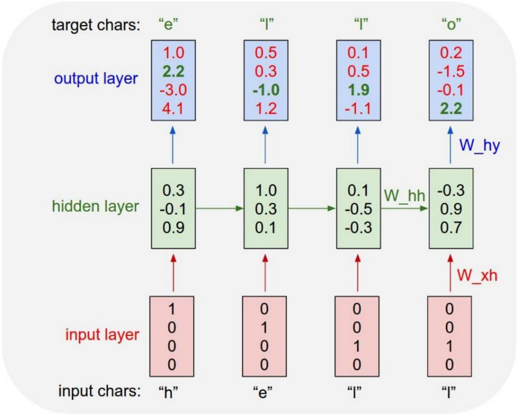


Example: Text Generation. Output seq has arbitrary size

Example: Character-level Language Model

Vocabulary: [h,e,l,o]

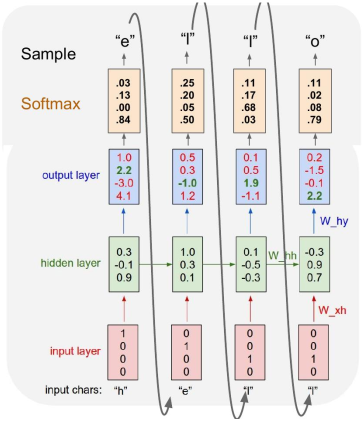
Example training sequence: **“hello”**



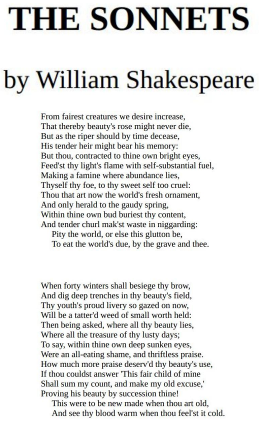
Example: Character-level Language Model Sampling

Vocabulary: [h,e,l,o]

At test-time sample characters one at a time, feed back to model



Ex. Text generation



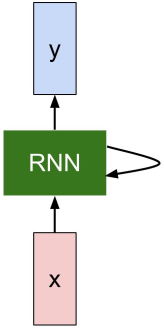
THE SONNETS

by William Shakespeare

From fairest creatures we desire increase,
That thereby beauty's rose might never die,
But as the riper should by time decease,
His tender heir might bear his memory:
But thou, contracted to thine own bright eyes,
Feed'st thy light's flame with self-substantial fuel,
Making a famine where abundance lies,
Thyself thy foe, to thy sweet self too cruel:
Thou that art now the world's fresh ornament,
And only herald to the gaudy spring,
Within thine own bud buriest thy content,
And tender churl mak'st waste in niggarding:
  Pity the world, or else this glutton be,
  To eat the world's due, by the grave and thee.

When forty winters shall besiege thy brow,
And dig deep trenches in thy beauty's field,
Thy youth's proud livery so gazed on now,
Will be a tatter'd weed of small worth held:
Then being asked, where all thy beauty lies,
Where all the treasure of thy lusty days;
To say, within thine own deep sunken eyes,
Were an all-eating shame, and thriftless praise.
How much more praise deserv'd thy beauty's use,
If thou couldst answer 'This fair child of mine
Shall sum my count, and make my old excuse,'
Proving his beauty by succession thine!
  This were to be new made when thou art old,
  And see thy blood warm when thou feel'st it cold.

$\longrightarrow$



Ex. Text generation

```
tyntd-iafhatawiaoihrdemot  lytdws  e ,tfti, astai f ogoh eoase rrranbyne 'nhthnee e
plia tklrgd t o idoe ns,smtt   h ne etie h,hregtrs nigtike,aoaenns lng
```

↓ train more

```
"Tmont thithey" fomesscerliund
Keushey. Thom here
sheulke, anmerenith ol sivh I lalterthend Bleipile shuwy fil on aseterlome
coaniogennc Phe lism thond hon at. MeiDimorotion in ther thize."
```

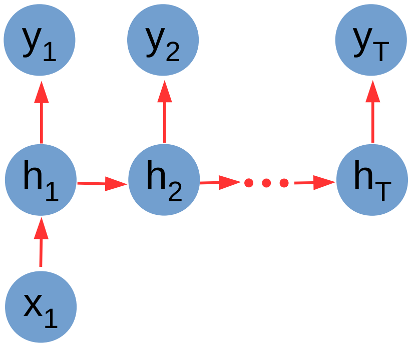
↓ train more

```
Aftair fall unsuch that the hall for Prince Velzonski's that me of
her hearly, and behs to so arwage fiving were to it beloge, pavu say falling misfort
how, and Gogition is so overelical and ofter.
```

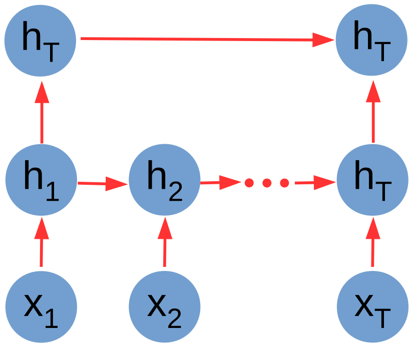
↓ train more

```
"Why do what that day," replied Natasha, and wishing to himself the fact the
princess, Princess Mary was easier, fed in had oftened him.
Pierre aking his soul came to the packs and drove up his father-in-law women.
```

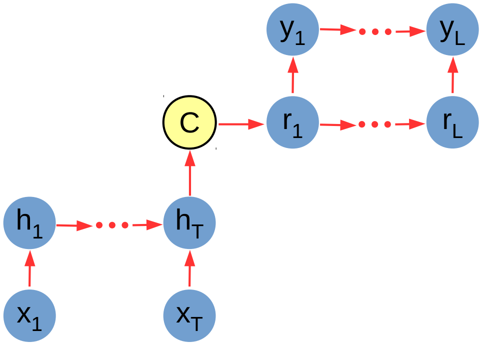
Depending of the application, we can define several configurations for the RNN



Depending of the application, we can define several configurations for the RNN.



Depending of the application, we can define several configurations for the RNN



RNNs are a powerful modelling framework.

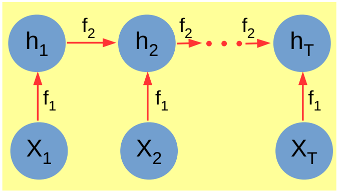
Lot of flexibility.

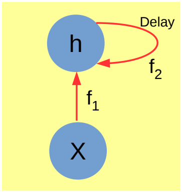
Let´s briefly discuss some further structural issues about RNNs.

RNNs can have different configurations. Key idea is to provide deep parameter sharing by including cycles

Typical Configuration

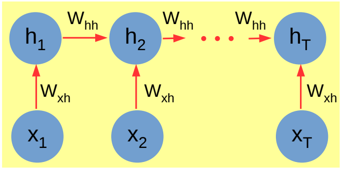
$$h_t = f(h_{t-1}, x_t)$$





Key step is the cycle or recurrent connection that provides some degree of "memory".

RNNs can have different configurations. The key idea is to provide deep parameter sharing by including cycles.



Parameter sharing:

- Provides a suitable mechanism that allows us to model sequences with different length (key property 2).
- Actually, an unfolded RNNs can be pictured as a deep feedforward network where all the layers share the same weights (why?).

Deep parameter sharing:

- While CNN parameter sharing is shallow (convolution mask), RNN sharing is deep (Why?).
- Deep parameter sharing allows us to model interactions that are far away (deeper) (key property 3).

Why we do use a RNN instead of a CNN to model a sequence?

RNNs exploit deep parameter sharing:

- It is important to understand the relevance of parameter sharing to model input sequences of variable length.
- In general, CNNs are applied to problems whose inputs and targets are encoded using vectors of fixed dimensionality.
- This is a significant limitation, since many important problems are best expressed as sequences whose lengths are not known a-priori. E.g., speech recognition, machine translation, Q&A, etc.
- In the case of RNNs, output vectors can be generated sequentially, so one can obtain an output sequence of variable length.

Why we do use a RNN instead of a CNN to model a sequence?

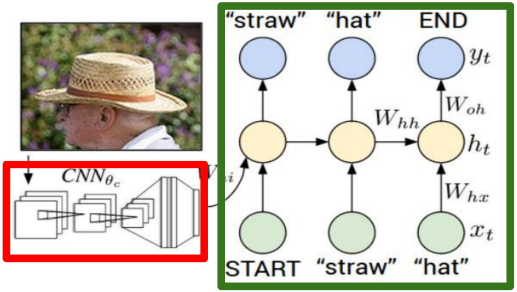
Deep parameter sharing and distance relations among data:

- It is also important to understand the relevance of modelling far interactions among data.
- CNNs are esentially hierarchical compositional models, while this is great and follows the pattern generation of a large list of domains, CNNs do not incorporate an explicit mechanism to model distant interactions among input data.
- Several applications need these types of models, such as text understanding, Q&A systems, end-to-end dialog agents, among many others.
- In general, the ability to distinguish distance context is key to create intelligent agents.

Instead of choosing between RNNs and CNNs, usually we use both in combination

Ex. Application: Image captioning

Recurrent Neural



Convolutional Neural Network

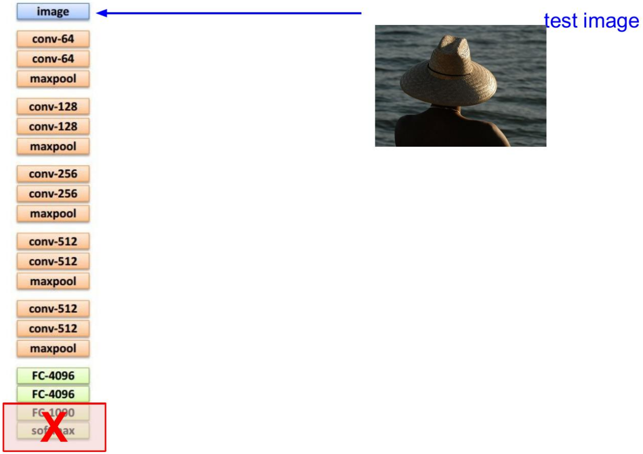
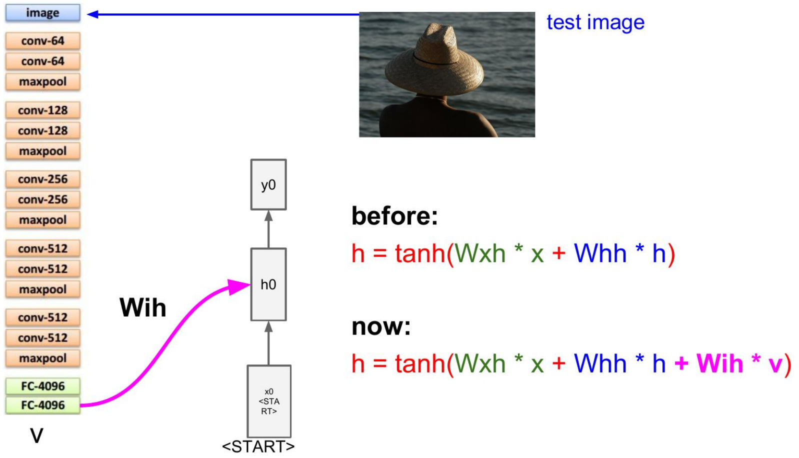


Figure from Stanford CS-231 class



before:

h = tanh(Wxh * x + Whh * h)

now:

h = tanh(Wxh * x + Whh * h + **Wih * v**)

Figure from Stanford CS-231 class

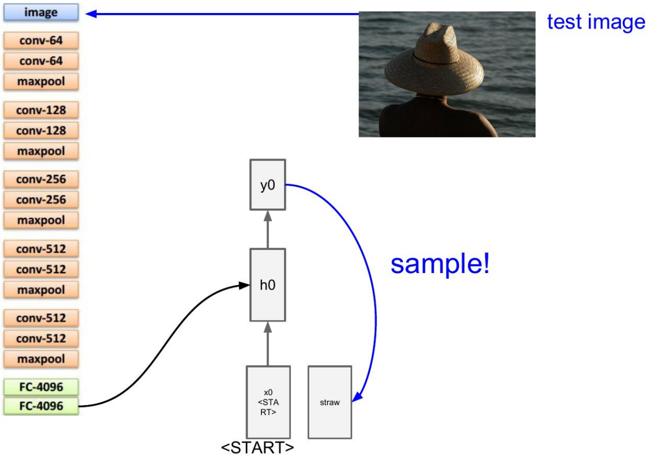


Figure from Stanford CS-231 class

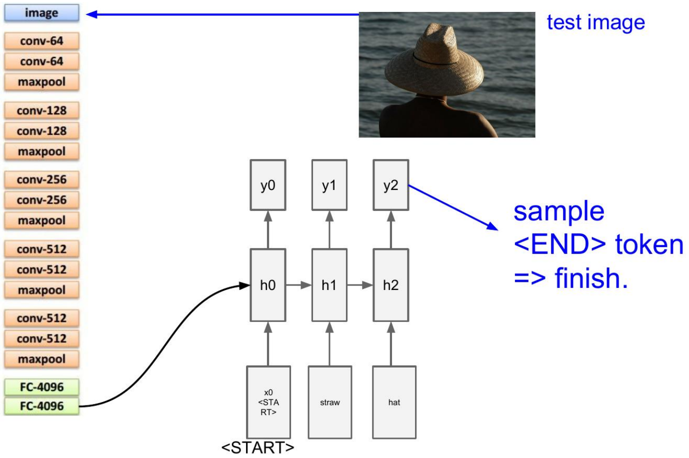
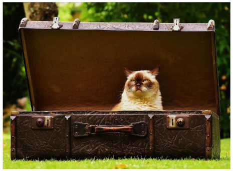
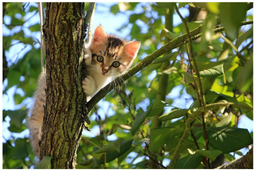


Figure from Stanford CS-231 class

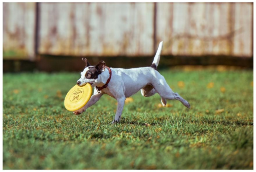
Ex. Image captioning: Results



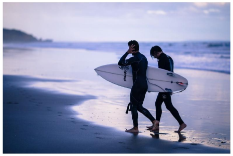
A cat sitting on a suitcase on the floor



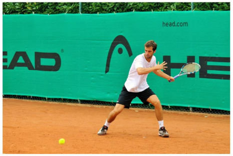
A cat is sitting on a tree branch



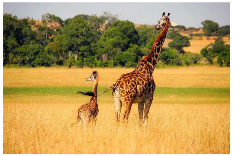
A dog is running in the grass with a frisbee



Two people walking on the beach with surfboards

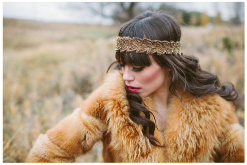


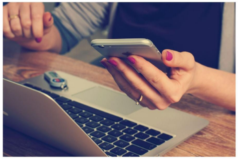
A tennis player in action on the court

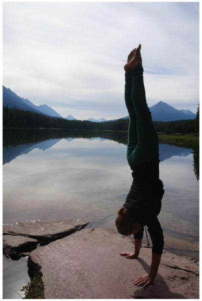


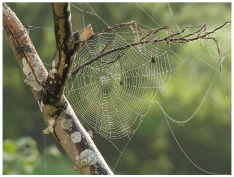
Two giraffes standing in a grassy field

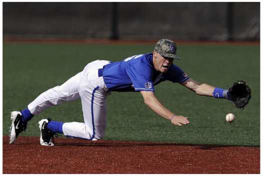
Ex. Image captioning: Failure Cases


A woman is holding a cat in her hand


A person holding a computer mouse on a desk


A woman standing on a beach holding a surfboard


A bird is perched on a tree branch


A man in a baseball uniform throwing a ball

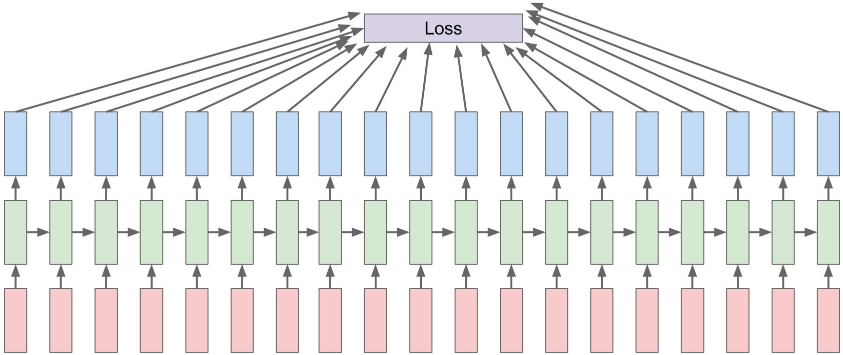
RNN Training

RNN Training

- Computation in most RNNs can be decomposed into three blocks of parameters and associated transformations:
  1. From input to hidden state W_{xh},
  2. From previous hidden state to next hidden state W_{hh}, and
  3. From hidden state to output W_{hy}.
- Training requires to establish a suitable loss function that relates RNN outputs to desired labeled sequences. Ex. cross entropy, ranking loss, etc.
- Minimization of this loss function is usually performed through mini-batch sthocastic gradient steps.
- Computing the gradient through a RNN is straightforward. One simply applies the generalized backpropagation algorithm to the unrolled computational graph.
- The use of backpropagation on the unrolled graph is called the backpropagation through time (BPTT) algorithm.

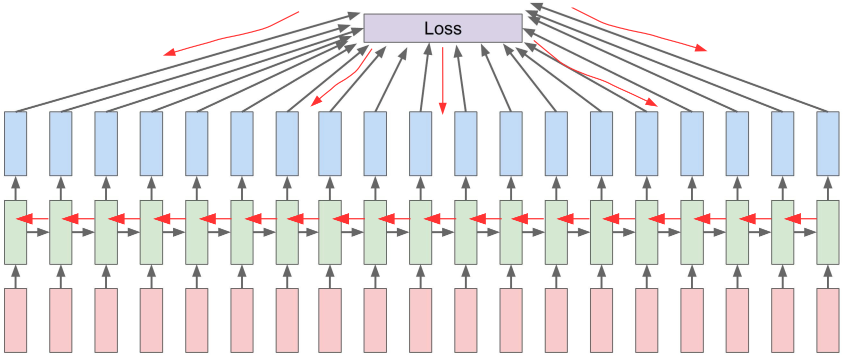
Backpropagation Through Time (BPTT)

Forward input data and hidden states through entire sequence to compute the “Loss” function



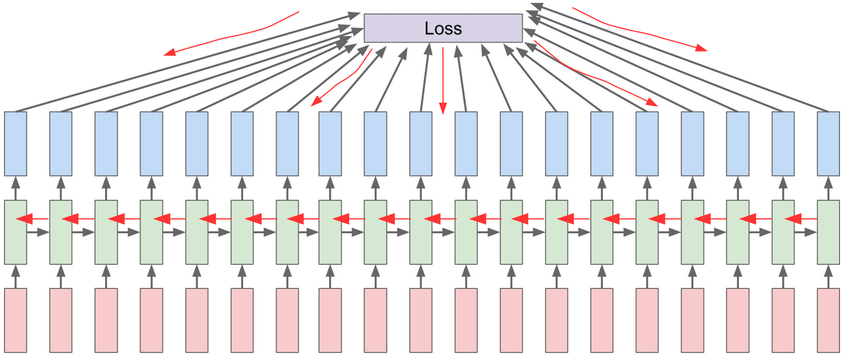
Backpropagation Through Time (BPTT)

Then backward error signal through entire sequence to compute gradients and perform weight updates



Backpropagation Through Time (BPTT)

Then backward error signal through entire sequence to compute gradients and perform weight updates



Big potential problems:

- Exploding gradient.
- Vanishing gradient.

The Challenge of Long-Term Dependencies

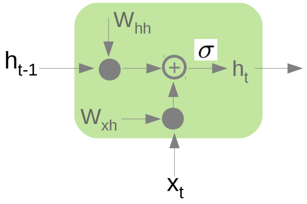
Backpropagation through time

- Exploding gradient problem.
- Vanishing gradient problem.
- Vanishing and exploding gradient problems depends on the largest singular value of W (>1 exploding gradients; <1 vanishing gradients).

- Exploding gradient problem can be controlled by gradient clipping (gradient clipping?).
- Vanishing gradient problem can be controlled using LSTMs or related gated architectures (LSTMs?).

Long Short Term Memory NWs or LSTMs (Hochreiter et al., 1997)

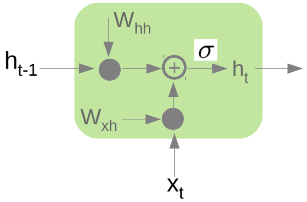
RNN



$$h_t = \sigma(\,W_{hh}\; h_{t-1} + W_{xh}\; x_t)$$

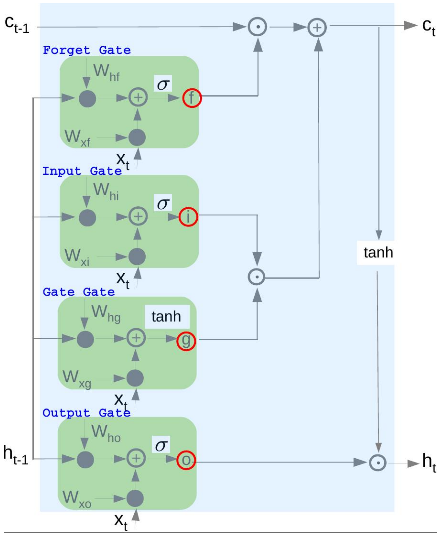
Long Short Term Memory NWs or LSTMs (Hochreiter et al., 1997)

RNN

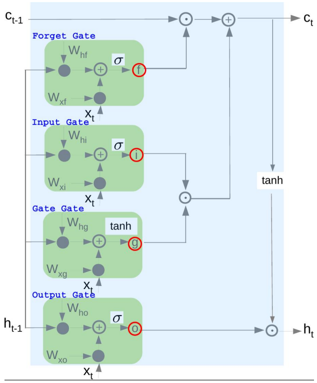


$$h_t = \sigma(\,W_{hh}\; h_{t-1} + W_{xh}\; x_t)$$

LSTM



Long Short Term Memory NWs or LSTMs (Hochreiter et al., 1997)

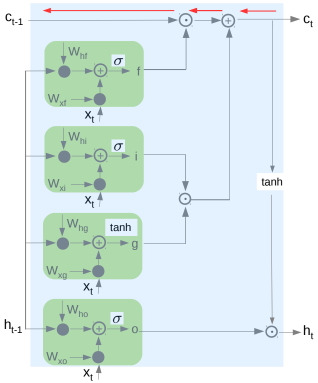


$$f = \sigma(W_{hf}\, h_{t-1} + W_{xf}\, x_t)$$
$$i = \sigma(W_{hi}\, h_{t-1} + W_{xi}\, x_t)$$
$$g = tanh(W_{hg}\, h_{t-1} + W_{xg}\, x_t)$$
$$o = \sigma(W_{ho}\, h_{t-1} + W_{xo}\, x_t)$$

$$c_t = f \odot c_{t-1} + i \odot g$$
$$h_t = o \odot tanh(c_t)$$

- f: Forget gate function, controls how to erase the cell.
- i: Input gate function, controls how to update the cell.
- g: Gate gate function, controls how much to update the cell.
- o: Output gate function, controls the cell output.
- c_t: internally accesed state of the LSTM cell.
- h_t: externally accesed output of the LSTM cell.

Long Short Term Memory NWs or LSTMs (Hochreiter et al., 1997)



$$f = \sigma(W_{hf}\, h_{t-1} + W_{xf}\, x_t)$$
$$i = \sigma(W_{hi}\, h_{t-1} + W_{xi}\, x_t)$$
$$g = tanh(W_{hg}\, h_{t-1} + W_{xg}\, x_t)$$
$$o = \sigma(W_{ho}\, h_{t-1} + W_{xo}\, x_t)$$

$$c_t = f \odot c_{t-1} + i \odot g$$
$$h_t = o \odot tanh(c_t)$$

- Backpropagation from c_t to c_{t-1} only elementwise multiplication by f, no matrix multiplication by W.

RNNs vs LSTMs.

- RNNs provide lot of flexibility in architectural design.
- Backward flow of gradients can explode or vanish.
- Exploding gradient problem can be controlled with gradient clipping.
- Vanishing gradient problem needs alternative solutions.
- In particular, LSTM and alternative gated architectures such as GRU (gated recurrent units) mitigate the vanishing gradient problem.

In summary, vanilla RNNs (basic RNNs) are simple, however, they don't work very well. LSTMs or GRUs are today the common choice.