

# Recurrent Neural Networks

## Machine Learning

(Largely based on slides from Fei-Fei Li & Andrej Karpathy & Justin Johnson & Serena Yeung)

**Prof. Sandra Avila**  
Institute of Computing (IC/Unicamp)

MC886, November 4, 2019

# Today's Agenda

---

- Recurrent Neural Networks
  - An Intuitive Explanation
  - **A More Formal Explanation**
  - Vanilla vs LSTMs

**RNNs:**

**A More Formal Explanation**

# Recurrent Neural Networks: Process Sequences

one to one



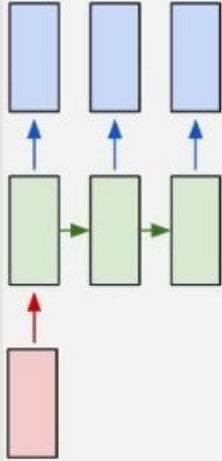
Vanilla Neural  
Networks

# Recurrent Neural Networks: Process Sequences

one to one



one to many



Vanilla Neural  
Networks

**Image Captioning**  
image  $\Rightarrow$  seq. words

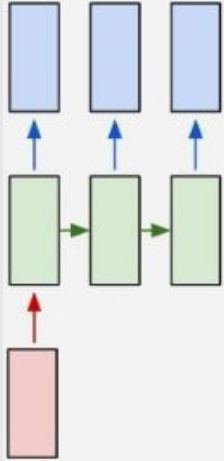
# Recurrent Neural Networks: Process Sequences

one to one



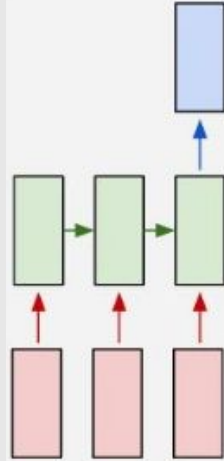
Vanilla Neural  
Networks

one to many



**Image Captioning**  
image  $\Rightarrow$  seq. words

many to one



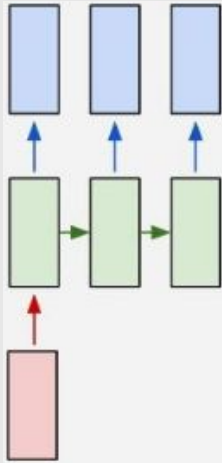
**Sentiment Classification**  
seq. words  $\Rightarrow$  sentiment

# Recurrent Neural Networks: Process Sequences

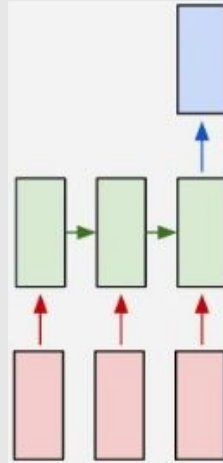
one to one



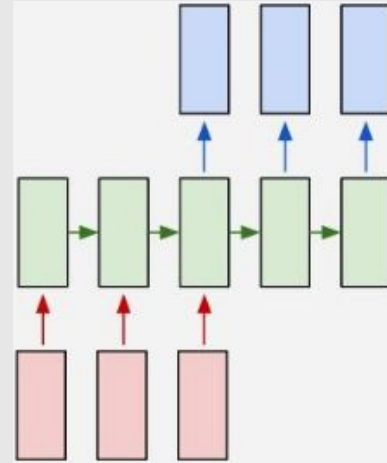
one to many



many to one



many to many



Vanilla Neural  
Networks

**Image Captioning**  
image  $\Rightarrow$  seq. words

**Sentiment Classification**  
seq. words  $\Rightarrow$  sentiment

**Machine Translation**  
seq. words  $\Rightarrow$  seq. of words

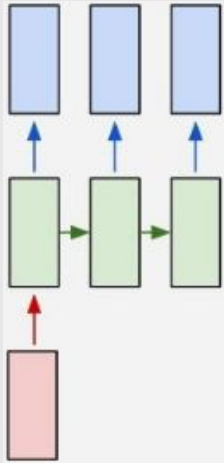
# Recurrent Neural Networks: Process Sequences

one to one



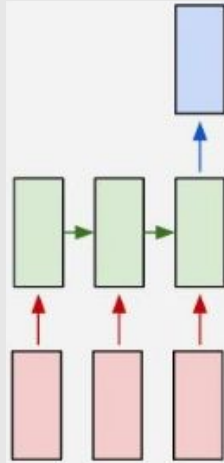
Vanilla Neural  
Networks

one to many



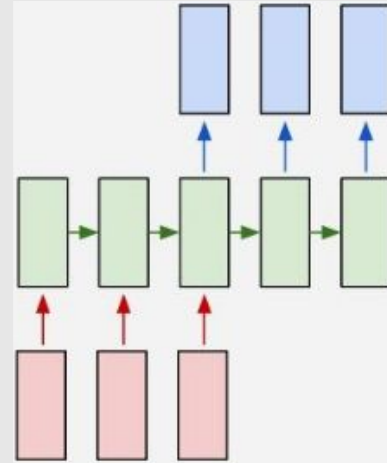
**Image Captioning**  
image  $\Rightarrow$  seq. words

many to one



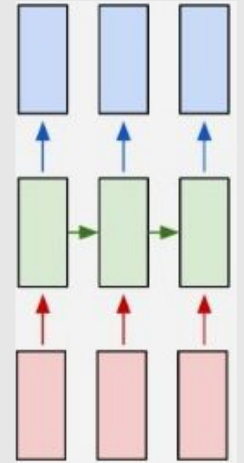
**Sentiment Classification**  
seq. words  $\Rightarrow$  sentiment

many to many



**Machine Translation**  
seq. words  $\Rightarrow$  seq. of words

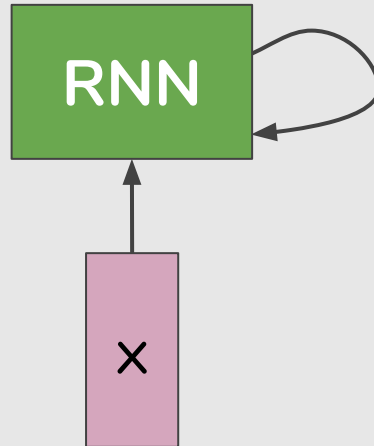
many to many



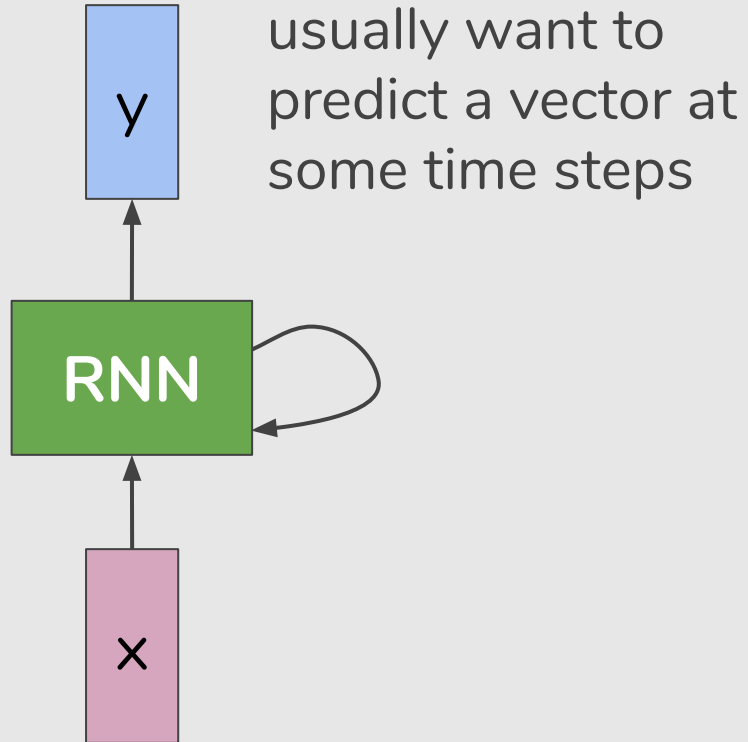
**Video classification  
on frame level**



# Recurrent Neural Network



# Recurrent Neural Network



# Recurrent Neural Network

We can process a sequence of vectors  $\mathbf{x}$  by applying a **recurrence formula** at every time step:

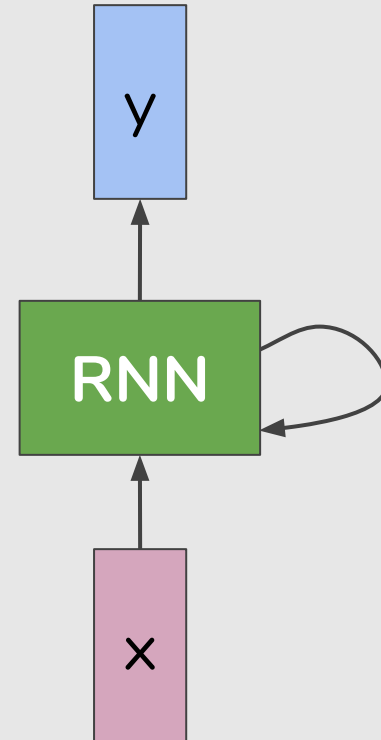
$$h_t = f_W(h_{t-1}, x_t)$$

new state

old state

input vector at  
some time step

some function  
with parameters  $W$

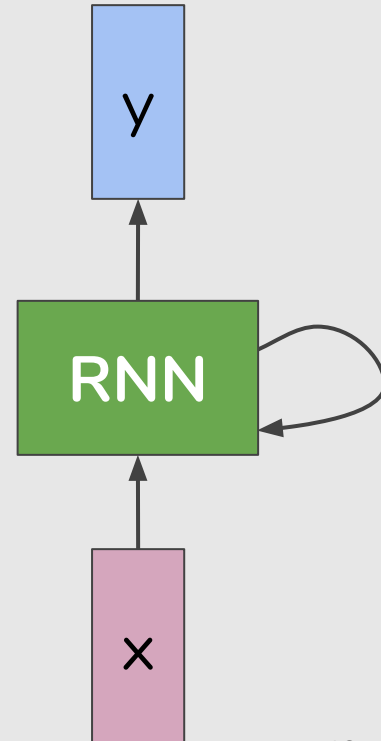


# Recurrent Neural Network

We can process a sequence of vectors  $\mathbf{x}$  by applying a **recurrence formula** at every time step:

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

**Notice:** the same function and the same set of parameters are used at every time step.



# Recurrent Neural Network

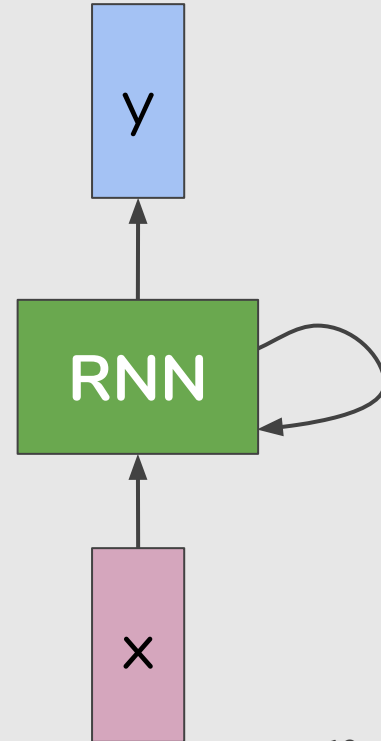
The state consists of a single “hidden” vector  $h$ :

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

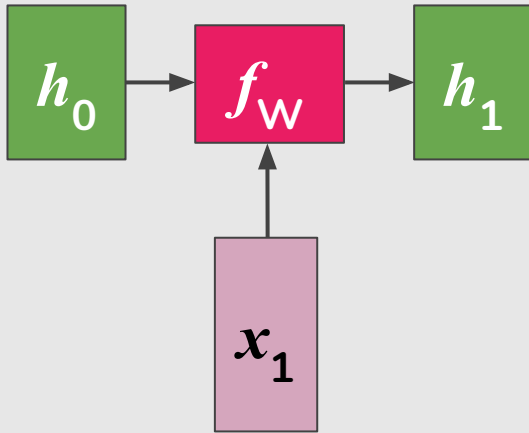


$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

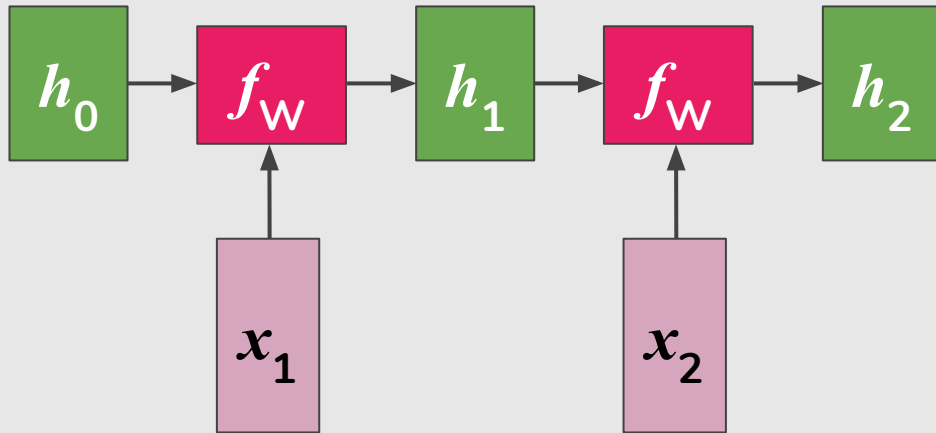
$$y_t = W_{hy}h_t$$



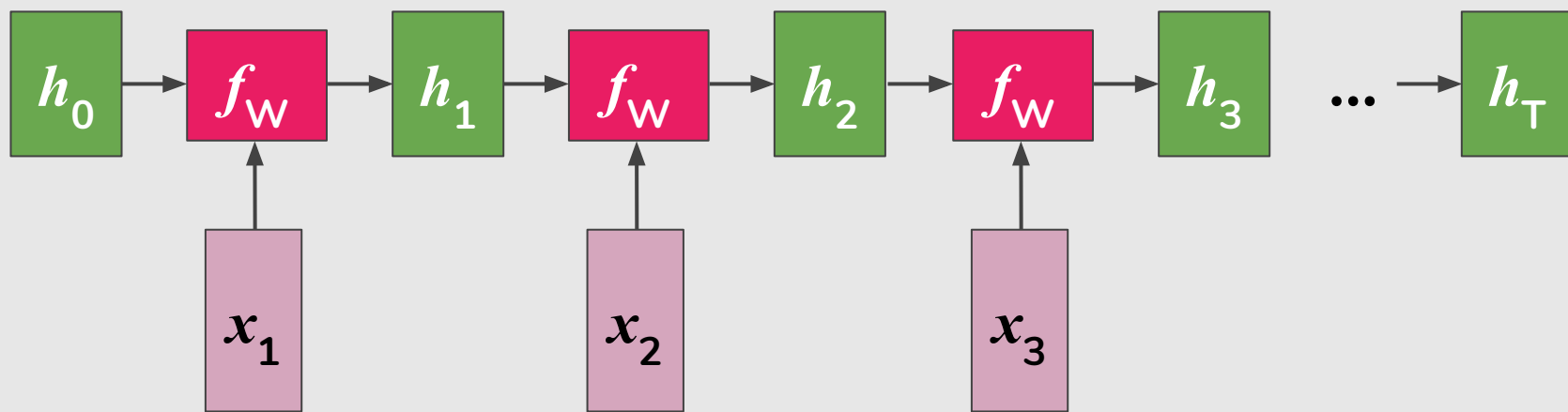
# RNN: Computational Graph



# RNN: Computational Graph



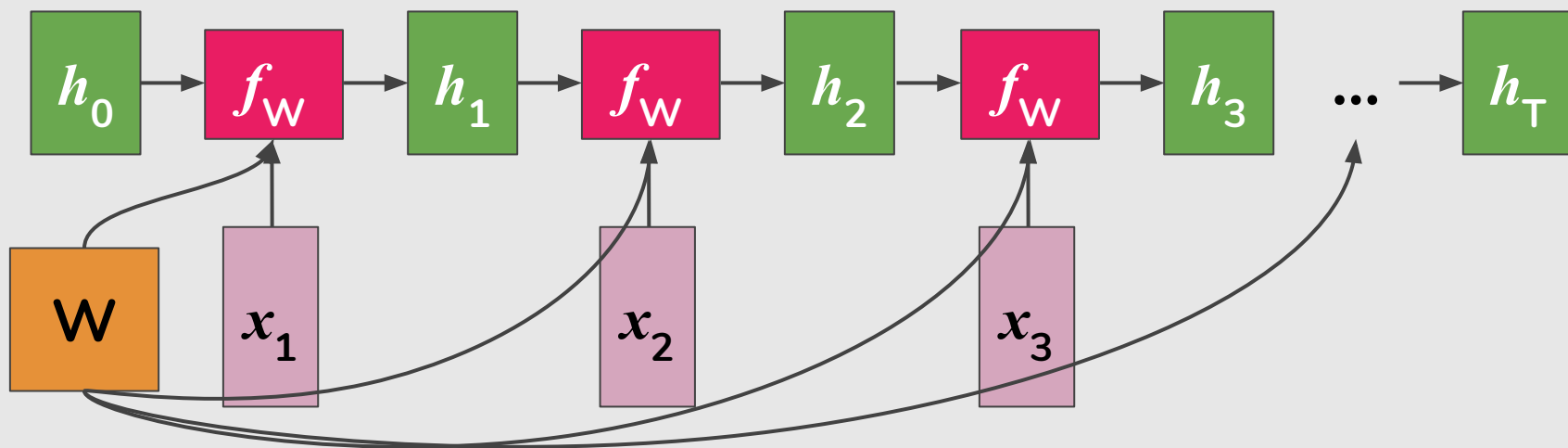
# RNN: Computational Graph



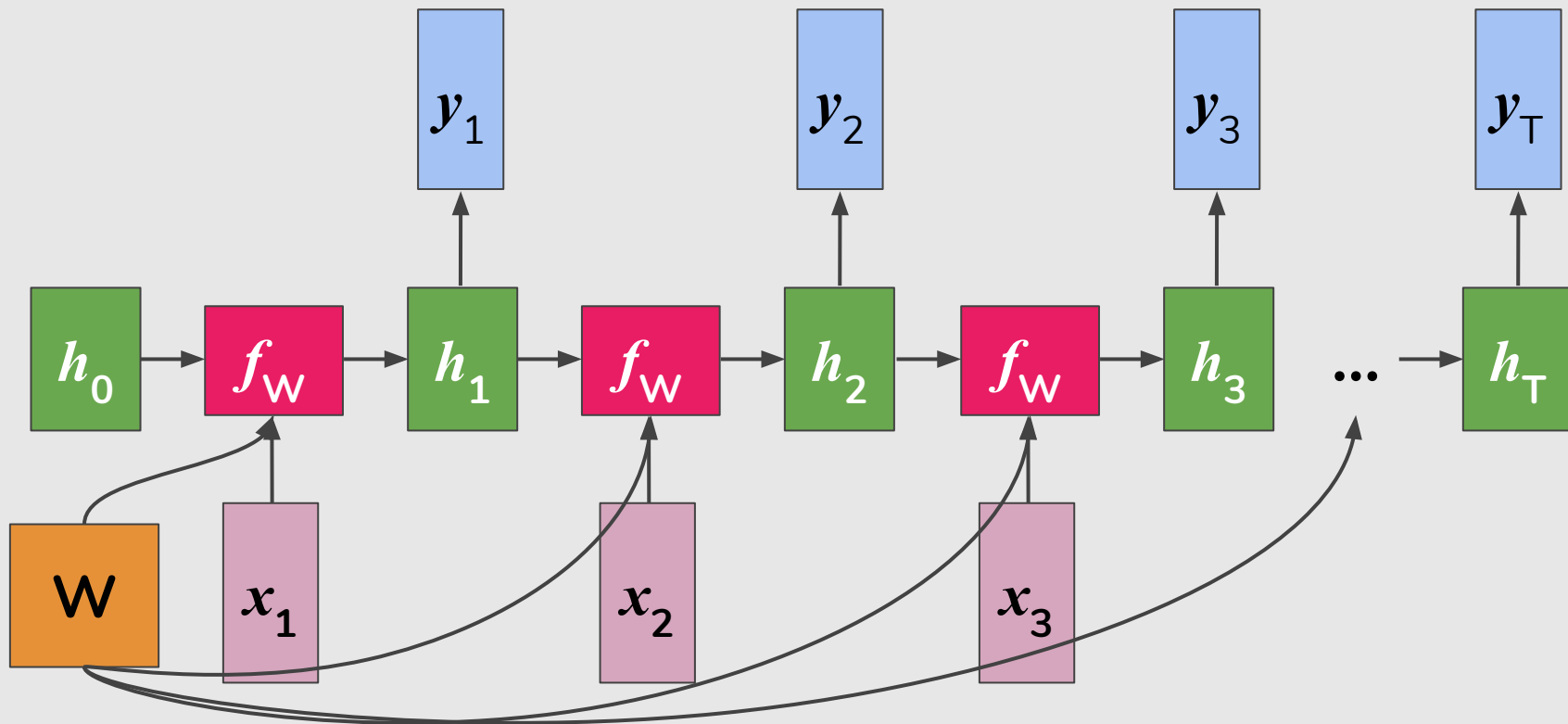


# RNN: Computational Graph

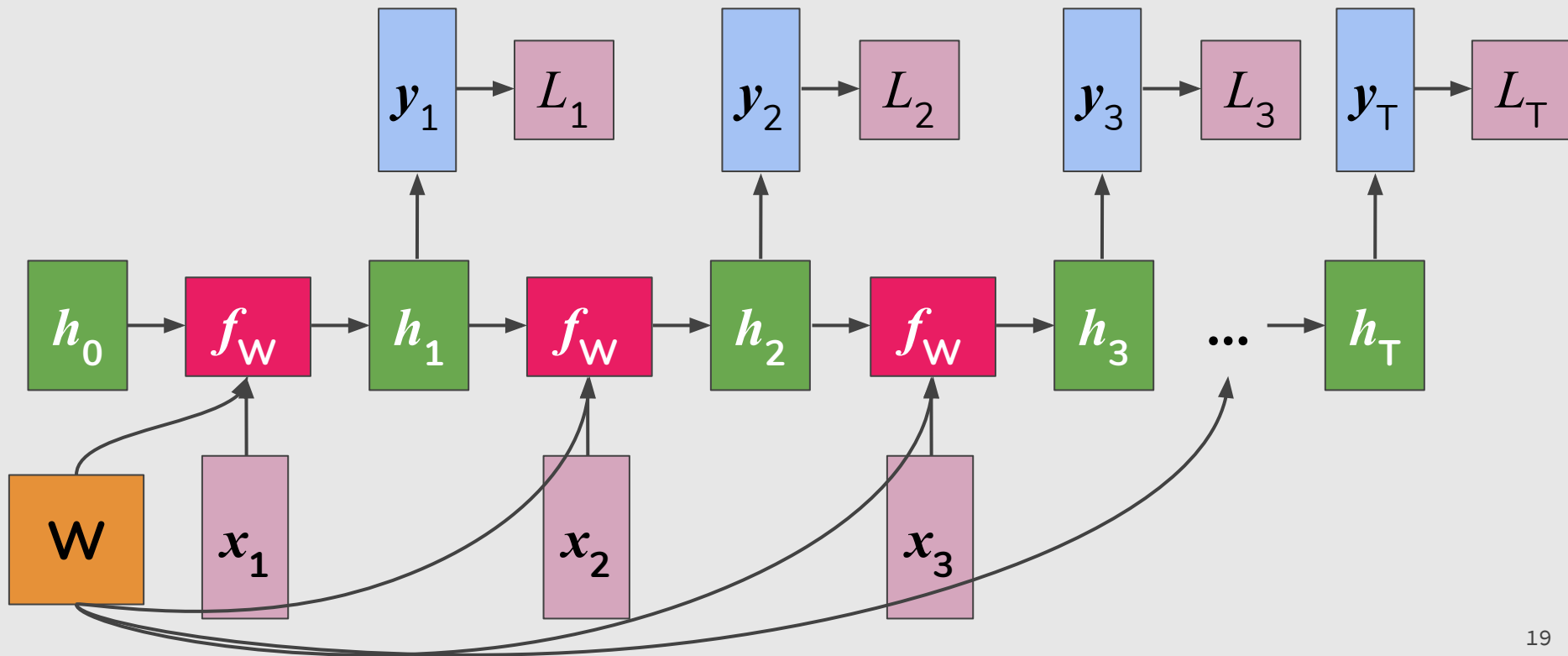
Re-use the same weight matrix at every time-step



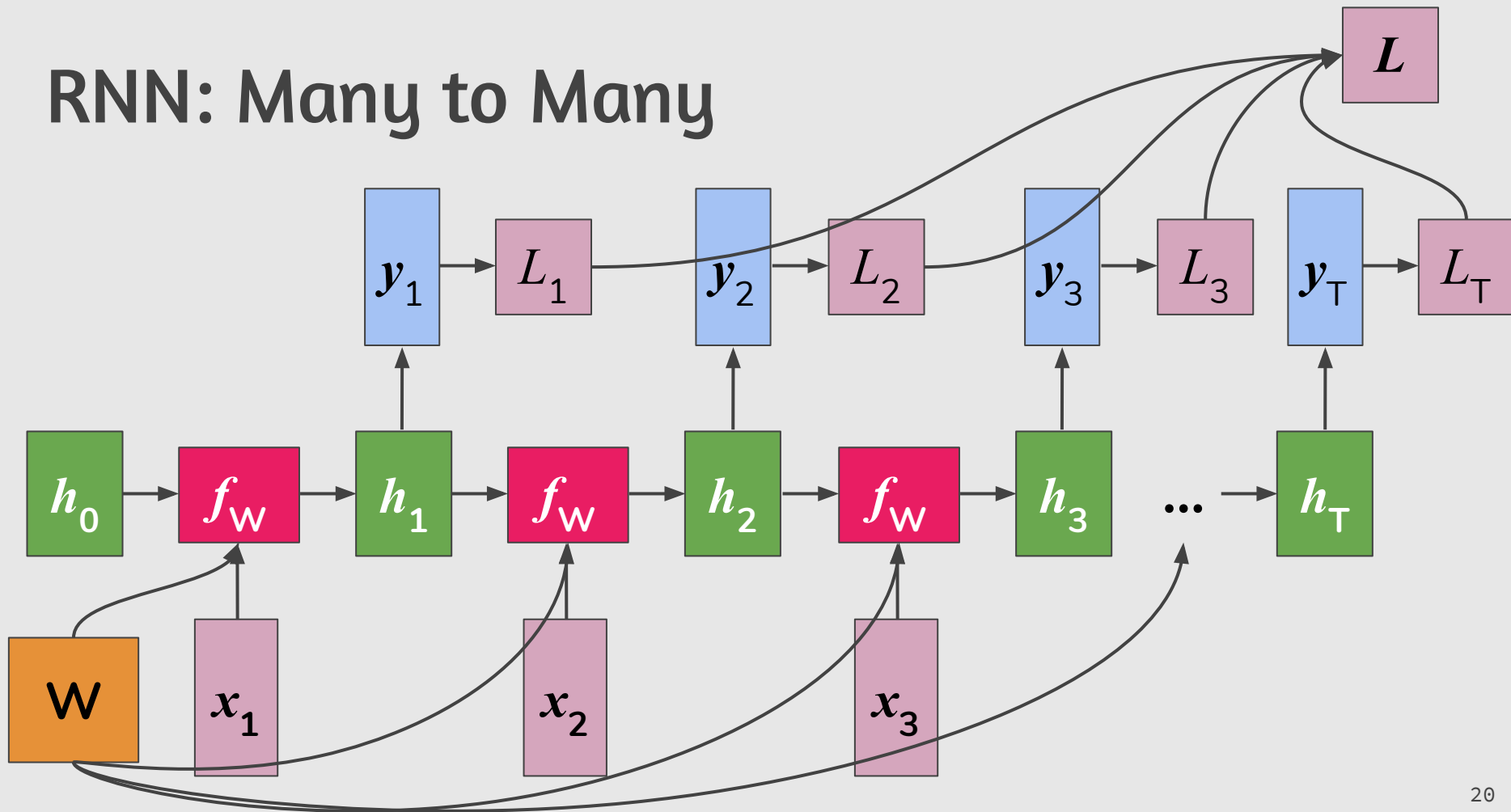
# RNN: Computational Graph



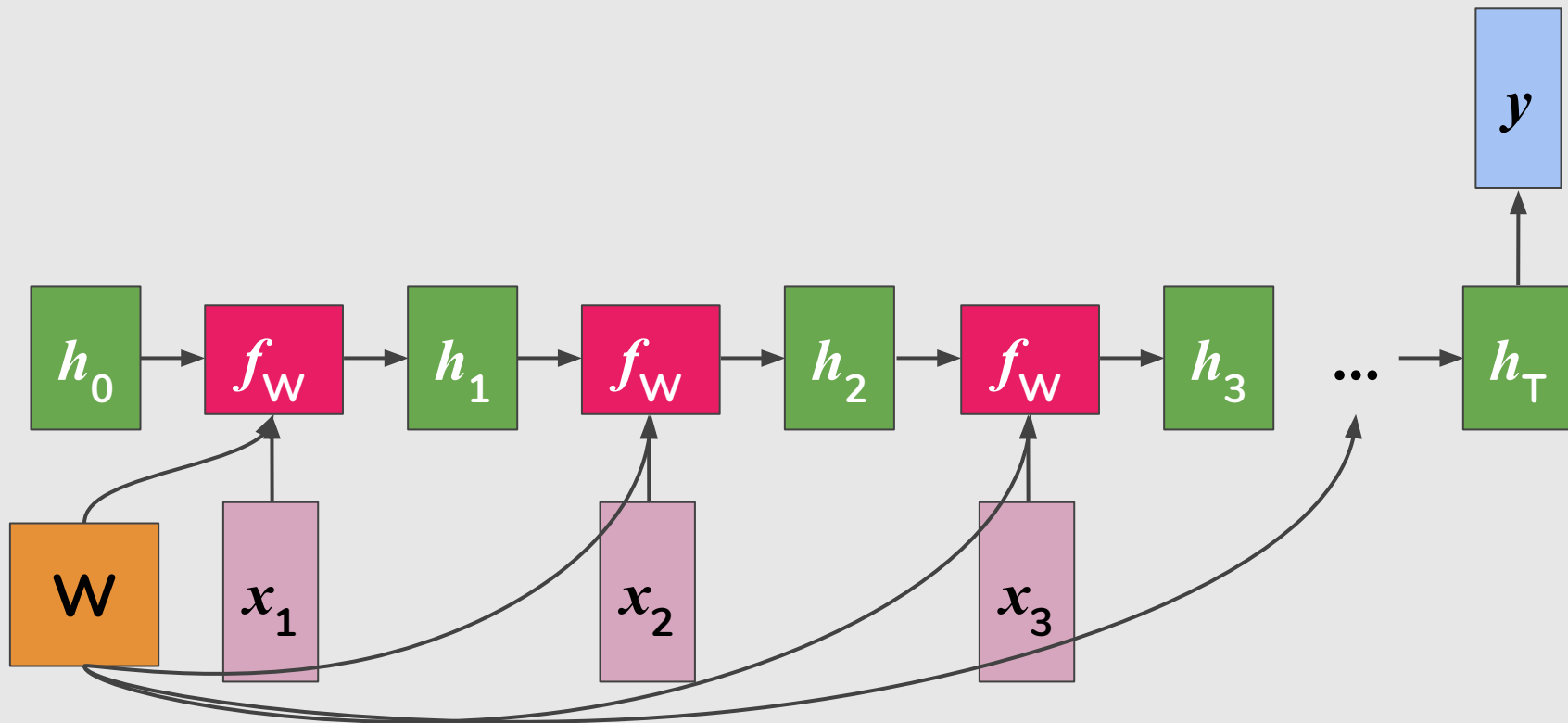
# RNN: Many to Many



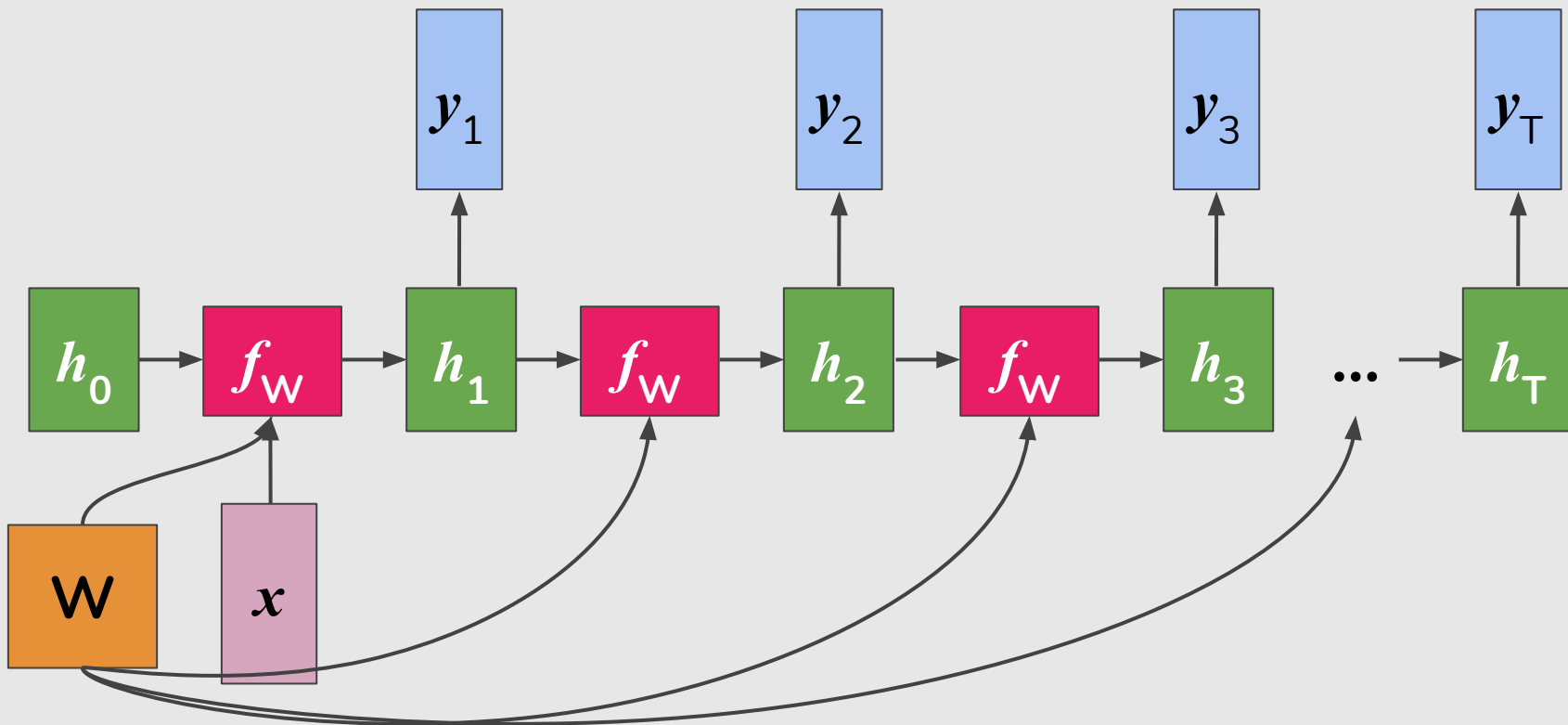
# RNN: Many to Many



# RNN: Many to One



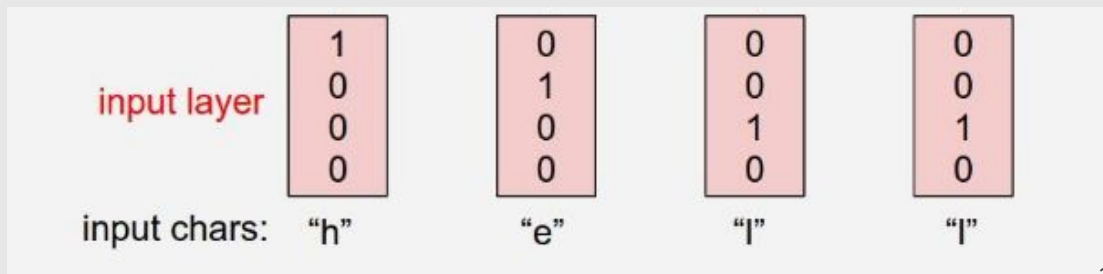
# RNN: One to Many



# Character-level Language Model

Vocabulary:  
[h,e,l,o]

Example training  
sequence:  
“hello”

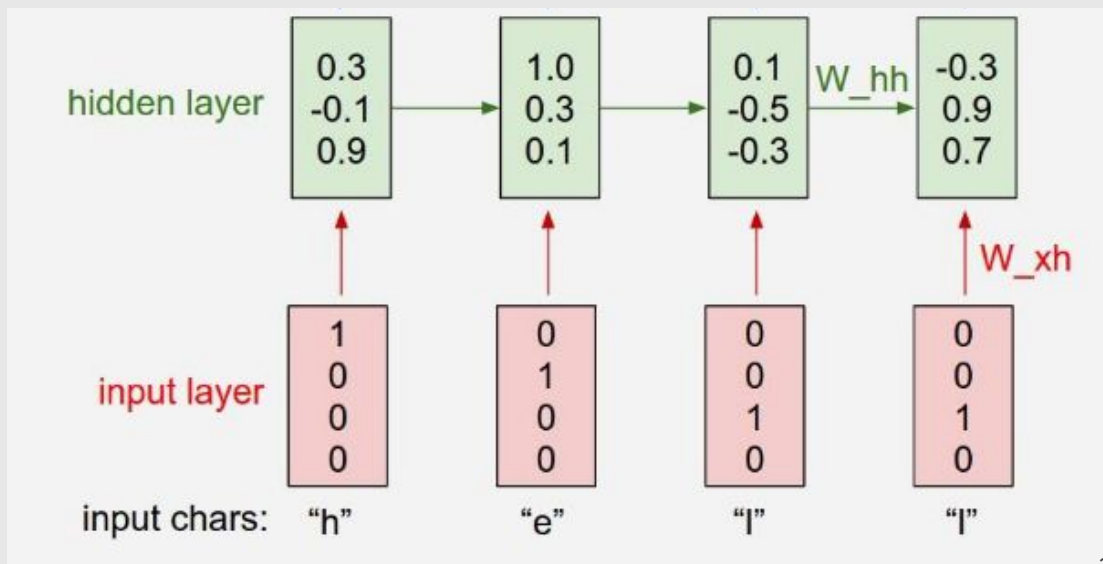


# Character-level Language Model

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

Vocabulary:  
[h,e,l,o]

Example training  
sequence:  
“hello”

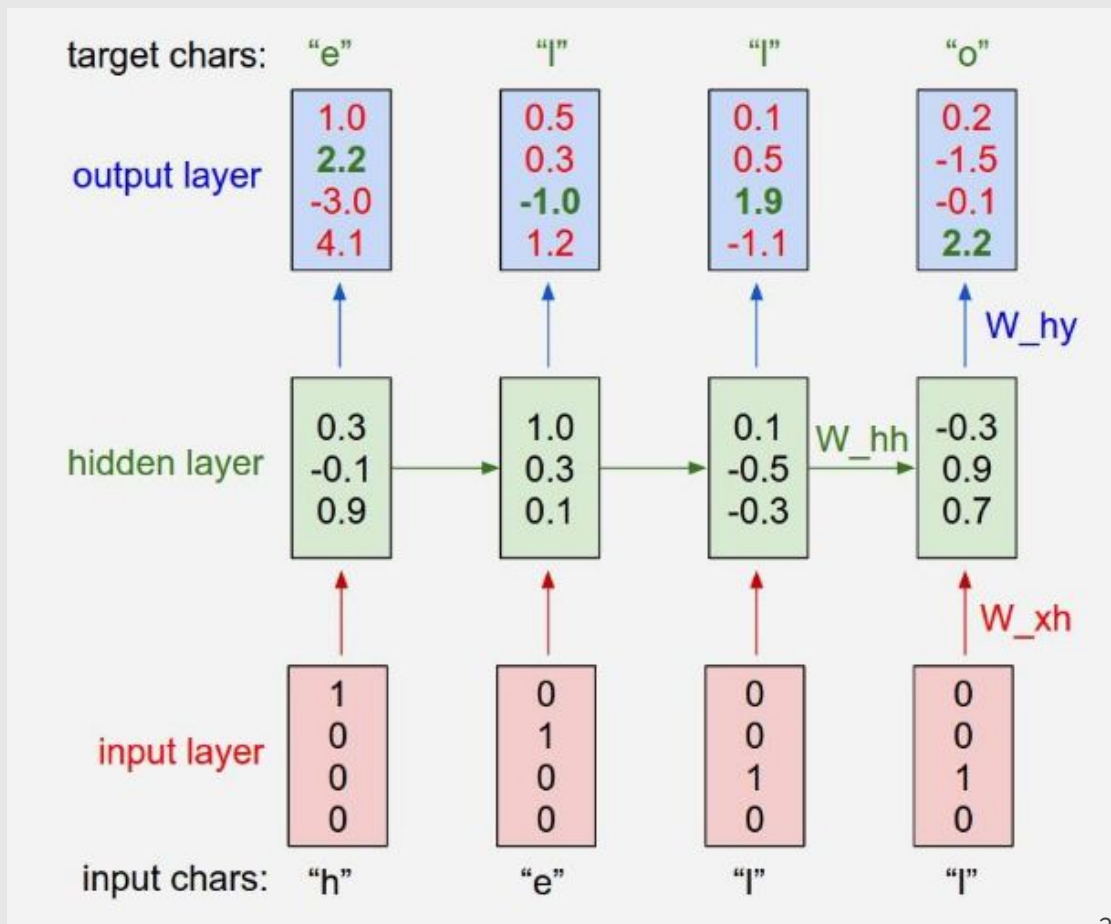


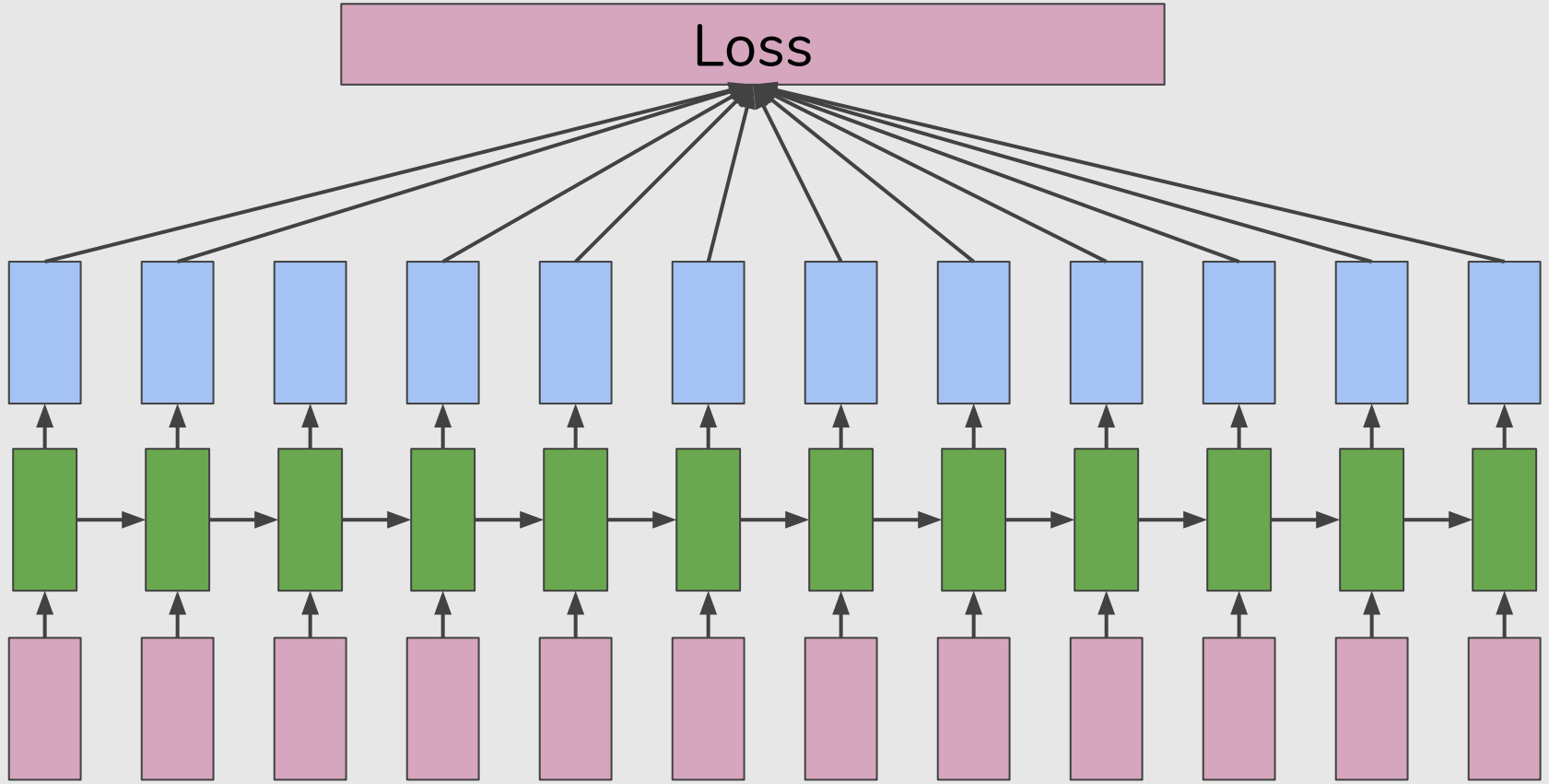


# Character-level Language Model

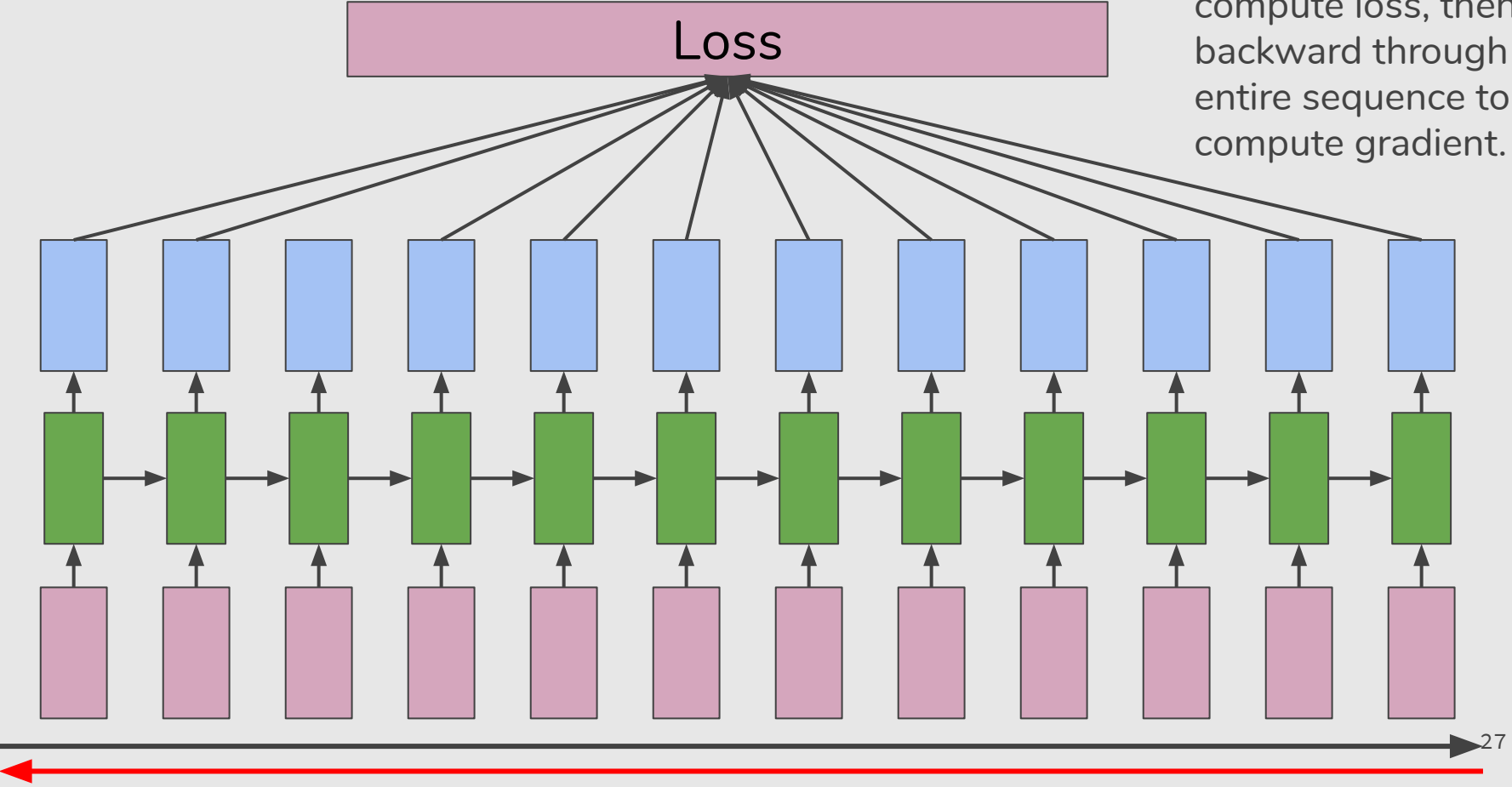
Vocabulary:  
[h,e,l,o]

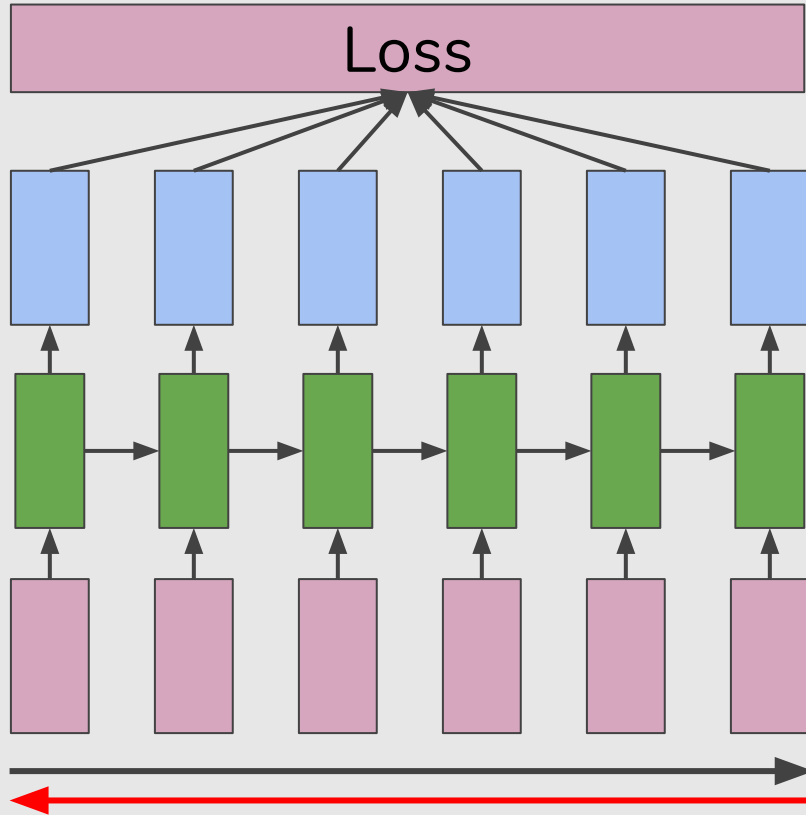
Example training  
sequence:  
"hello"





Forward through entire sequence to compute loss, then backward through entire sequence to compute gradient.

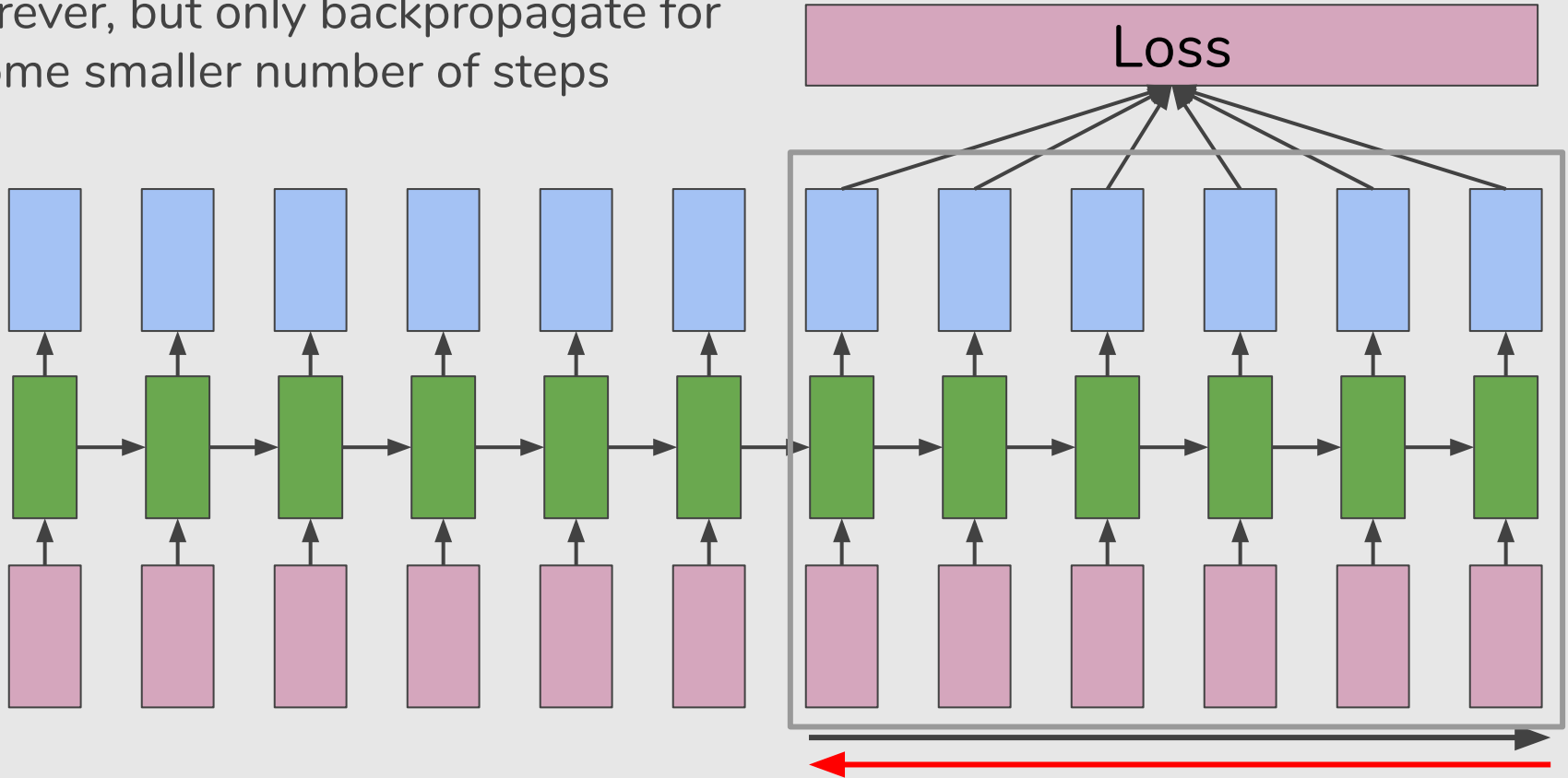




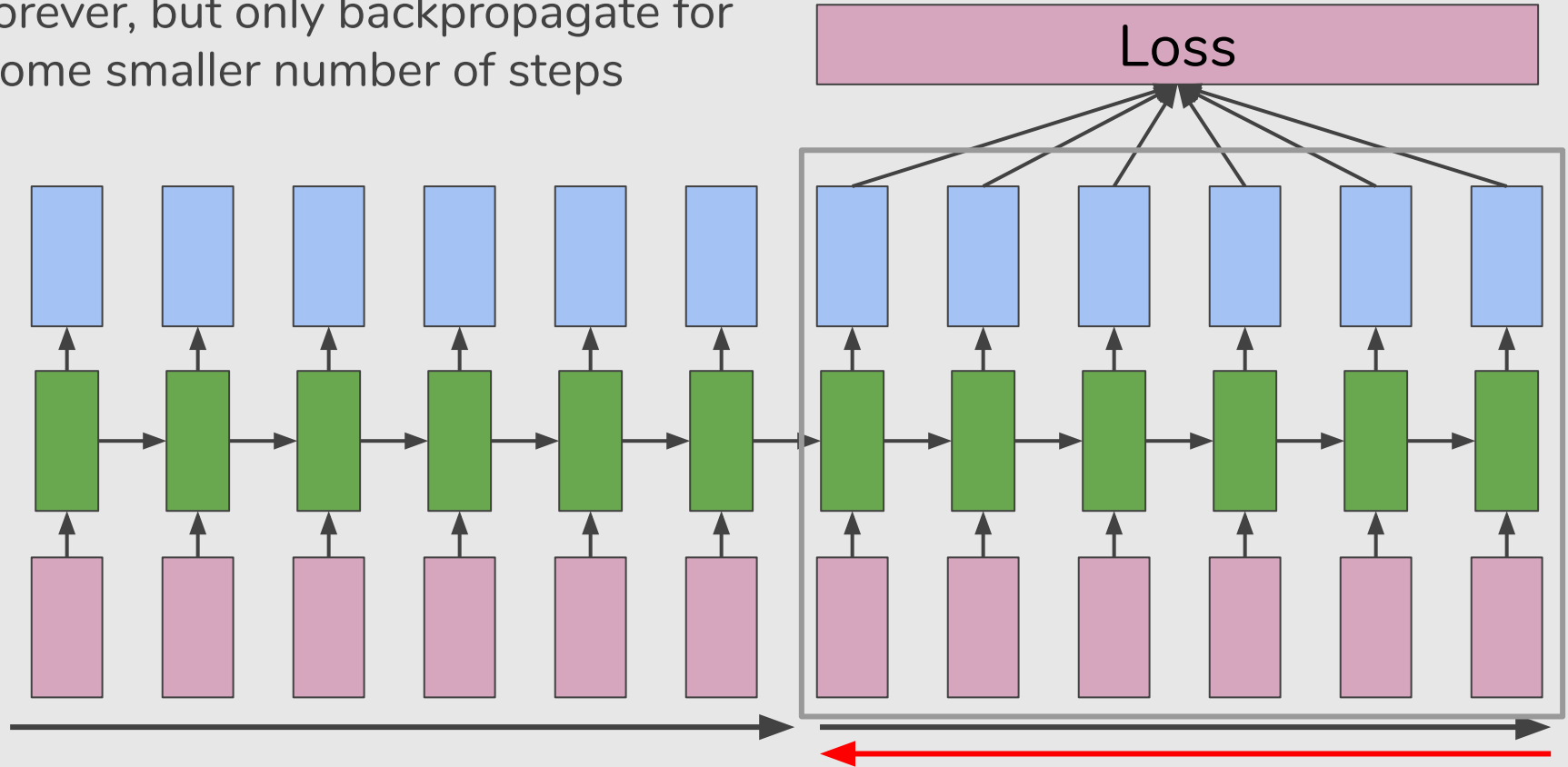
## Truncated backpropagation through time (TBTT)

Run forward and backward through chunks of the sequence instead of whole sequence

Carry hidden states forward in time forever, but only backpropagate for some smaller number of steps



Carry hidden states forward in time forever, but only backpropagate for some smaller number of steps



# https://gist.github.com/karpathy/d4dee566867f8291f086



karpathy / min-char-rnn.py

Last active 33 minutes ago • Report abuse

Subscribe

★ Star

3k

Fork

1.1k

Code

Revisions 7

★ Stars 3029

Forks 1077

Embed

<script src="https://gis



Download ZIP

Minimal character-level language model with a Vanilla Recurrent Neural Network, in Python/numpy

min-char-rnn.py

Raw

```
1 """
2 Minimal character-level Vanilla RNN model. Written by Andrej Karpathy (@karpathy)
3 BSD License
4 """
5 import numpy as np
6
7 # data I/O
8 data = open('input.txt', 'r').read() # should be simple plain text file
9 chars = list(set(data))
10 data_size, vocab_size = len(data), len(chars)
11 print 'data has %d characters, %d unique.' % (data_size, vocab_size)
12 char_to_ix = { ch:i for i,ch in enumerate(chars) }
13 ix_to_char = { i:ch for i,ch in enumerate(chars) }
14
15 # hyperparameters
16 hidden_size = 100 # size of hidden layer of neurons
17 seq_length = 25 # number of steps to unroll the RNN for
18 learning_rate = 1e-1
19
20 # model parameters
21 Wxh = np.random.randn(hidden_size, vocab_size)*0.01 # input to hidden
22 Whh = np.random.randn(hidden_size, hidden_size)*0.01 # hidden to hidden
23 Why = np.random.randn(vocab_size, hidden_size)*0.01 # hidden to output
24 bh = np.zeros((hidden_size, 1)) # hidden bias
25 by = np.zeros((vocab_size, 1)) # output bias
26
27 def lossFun(inputs, targets, hprev):
28     """
29     inputs, targets are both list of integers.
30     hprev is Hx1 array of initial hidden state
31     returns the loss, gradients on model parameters, and last hidden state
32     """
33     xs, hs, ys, ps = {}, {}, {}, {}
```

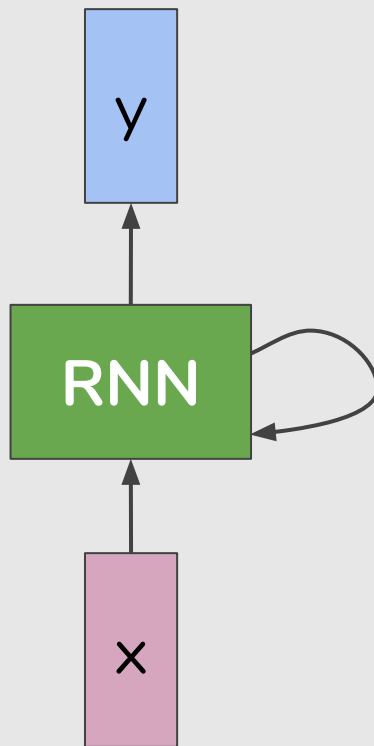
## Min-char-rnn.py 112 lines of Python

# THE SONNETS

by William Shakespeare

From fairest creatures we desire increase,  
That thereby beauty's rose might never die,  
But as the ripper should by time decrease,  
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.





At first:

tyntd-iafhatawiaoihrdemot lytdws e ,tfti, astai f ogoh eoase rrranbyne 'nhtnee e  
plia tklrqd 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 shuw y 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 offer.

↓ 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.

PANDARUS:

Alas, I think he shall be come approached and the day  
When little strain would be attain'd into being never fed,  
And who is but a chain and subjects of his death,  
I should not sleep.

Second Senator:

They are away this miseries, produced upon my soul,  
Breaking and strongly should be buried, when I perish  
The earth and thoughts of many states.

DUKE VINCENTIO:

Well, your wit is in the care of side and that.

Second Lord:

They would be ruled after this chamber, and  
my fair nudes begun out of the fact, to be conveyed,  
Whose noble souls I'll have the heart of the wars.

Clown:

Come, sir, I will make did behold your worship.

VIOLA:

I'll drink it.











VIOLA:

Why, Salisbury must find his flesh and thought  
That which I am not apt, not a man and in fire,  
To show the reining of the raven and the wars  
To grace my hand reproach within, and not a fair are hand,  
That Caesar and my goodly father's world;  
When I was heaven of presence and our fleets,  
We spare with hours, but cut thy council I am great,  
Murdered and by thy master's ready there  
My power to give thee but so much as hell:  
Some service in the noble bondman here,  
Would show him to her wine.

KING LEAR:

O, if you were a feeble sight, the courtesy of your law,  
Your sight and several breath, will wear the gods  
With his heads, and my hands are wonder'd at the deeds,  
So drop upon your lordship's head, and your opinion  
Shall be against your honour.

## Browse chapters

| Part          | Chapter                 | online                 | TeX source  | view pdf  |
|---------------|-------------------------|------------------------|---|---|
| Preliminaries |                         |                        |   |   |
|               | 1. Introduction         | <a href="#">online</a> | <a href="#">tex</a>  | <a href="#">pdf</a>  |
|               | 2. Conventions          | <a href="#">online</a> | <a href="#">tex</a>  | <a href="#">pdf</a>  |
|               | 3. Set Theory           | <a href="#">online</a> | <a href="#">tex</a>  | <a href="#">pdf</a>  |
|               | 4. Categories           | <a href="#">online</a> | <a href="#">tex</a>  | <a href="#">pdf</a>  |
|               | 5. Topology             | <a href="#">online</a> | <a href="#">tex</a>  | <a href="#">pdf</a>  |
|               | 6. Sheaves on Spaces    | <a href="#">online</a> | <a href="#">tex</a>  | <a href="#">pdf</a>  |
|               | 7. Sites and Sheaves    | <a href="#">online</a> | <a href="#">tex</a>  | <a href="#">pdf</a>  |
|               | 8. Stacks               | <a href="#">online</a> | <a href="#">tex</a>  | <a href="#">pdf</a>  |
|               | 9. Fields               | <a href="#">online</a> | <a href="#">tex</a>  | <a href="#">pdf</a>  |
|               | 10. Commutative Algebra | <a href="#">online</a> | <a href="#">tex</a>  | <a href="#">pdf</a>  |

## Parts

1. [Preliminaries](#)
2. [Schemes](#)
3. [Topics in Scheme Theory](#)
4. [Algebraic Spaces](#)
5. [Topics in Geometry](#)
6. [Deformation Theory](#)
7. [Algebraic Stacks](#)
8. [Miscellany](#)

## Statistics

- The Stacks project now consists of
- 455910 lines of code
  - 14221 tags (56 inactive tags)
  - 2366 sections

Latex source



For  $\bigoplus_{n=1, \dots, m}$  where  $\mathcal{L}_{m_\bullet} = 0$ , hence we can find a closed subset  $\mathcal{H}$  in  $\mathcal{H}$  and any sets  $\mathcal{F}$  on  $X$ ,  $U$  is a closed immersion of  $S$ , then  $U \rightarrow T$  is a separated algebraic space.

*Proof.* Proof of (1). It also start we get

$$S = \text{Spec}(R) = U \times_X U \times_X U$$

and the comparico in the fibre product covering we have to prove the lemma generated by  $\coprod Z \times_U U \rightarrow V$ . Consider the maps  $M$  along the set of points  $\text{Sch}_{fppf}$  and  $U \rightarrow U$  is the fibre category of  $S$  in  $U$  in Section, ?? and the fact that any  $U$  affine, see Morphisms, Lemma ???. Hence we obtain a scheme  $S$  and any open subset  $W \subset U$  in  $\text{Sh}(G)$  such that  $\text{Spec}(R') \rightarrow S$  is smooth or an

$$U = \bigcup U_i \times_S U_i$$

which has a nonzero morphism we may assume that  $f_i$  is of finite presentation over  $S$ . We claim that  $\mathcal{O}_{X, x}$  is a scheme where  $x, x', s'' \in S'$  such that  $\mathcal{O}_{X, x'} \rightarrow \mathcal{O}'_{X', x'}$  is separated. By Algebra, Lemma ?? we can define a map of complexes  $\text{GL}_{S'}(x'/S'')$  and we win.  $\square$

To prove study we see that  $\mathcal{F}|_U$  is a covering of  $\mathcal{X}'$ , and  $\mathcal{T}_i$  is an object of  $\mathcal{F}_{X/S}$  for  $i > 0$  and  $\mathcal{F}_p$  exists and let  $\mathcal{F}_i$  be a presheaf of  $\mathcal{O}_X$ -modules on  $\mathcal{C}$  as a  $\mathcal{F}$ -module. In particular  $\mathcal{F} = U/\mathcal{F}$  we have to show that

$$\widetilde{M}^\bullet = \mathcal{I}^\bullet \otimes_{\text{Spec}(k)} \mathcal{O}_{S, s} - i_X^{-1} \mathcal{F}$$

is a unique morphism of algebraic stacks. Note that

$$\text{Arrows} = (\text{Sch}/S)_{fppf}^{opp}, (\text{Sch}/S)_{fppf}$$

and

$$V = \Gamma(S, \mathcal{O}) \mapsto (U, \text{Spec}(A))$$

is an open subset of  $X$ . Thus  $U$  is affine. This is a continuous map of  $X$  is the inverse, the groupoid scheme  $S$ .

*Proof.* See discussion of sheaves of sets.  $\square$

The result for prove any open covering follows from the less of Example ??. It may replace  $S$  by  $X_{spaces, \acute{e}tale}$  which gives an open subspace of  $X$  and  $T$  equal to  $S_{Zar}$ , see Descent, Lemma ??. Namely, by Lemma ?? we see that  $R$  is geometrically regular over  $S$ .

**Lemma 0.1.** Assume (3) and (3) by the construction in the description.

Suppose  $X = \lim |X|$  (by the formal open covering  $X$  and a single map  $\text{Proj}_X(\mathcal{A}) = \text{Spec}(B)$  over  $U$  compatible with the complex

$$\text{Set}(\mathcal{A}) = \Gamma(X, \mathcal{O}_{X, \mathcal{O}_X}).$$

When in this case of to show that  $\mathcal{Q} \rightarrow \mathcal{C}_{Z/X}$  is stable under the following result in the second conditions of (1), and (3). This finishes the proof. By Definition ?? (without element is when the closed subschemes are catenary. If  $T$  is surjective we may assume that  $T$  is connected with residue fields of  $S$ . Moreover there exists a closed subspace  $Z \subset X$  of  $X$  where  $U$  in  $X'$  is proper (some defining as a closed subset of the uniqueness it suffices to check the fact that the following theorem

(1)  $f$  is locally of finite type. Since  $S = \text{Spec}(R)$  and  $Y = \text{Spec}(R)$ .

*Proof.* This is form all sheaves of sheaves on  $X$ . But given a scheme  $U$  and a surjective étale morphism  $U \rightarrow X$ . Let  $U \cap U = \coprod_{i=1, \dots, n} U_i$  be the scheme  $X$  over  $S$  at the schemes  $X_i \rightarrow X$  and  $U = \lim_i X_i$ .  $\square$

The following lemma surjective restrocomposes of this implies that  $\mathcal{F}_{x_0} = \mathcal{F}_{x_0} = \mathcal{F}_{X, \dots, 0}$ .

**Lemma 0.2.** Let  $X$  be a locally Noetherian scheme over  $S$ ,  $E = \mathcal{F}_{X/S}$ . Set  $\mathcal{I} = \mathcal{J}_1 \subset \mathcal{I}'_n$ . Since  $\mathcal{I}^n \subset \mathcal{I}'_n$  are nonzero over  $i_0 \leq \mathfrak{p}$  is a subset of  $\mathcal{J}_{n,0} \circ \overline{A}_2$  works.

**Lemma 0.3.** In Situation ??. Hence we may assume  $\mathfrak{q}' = 0$ .

*Proof.* We will use the property we see that  $\mathfrak{p}$  is the next functor (??). On the other hand, by Lemma ?? we see that

$$D(\mathcal{O}_{X'}) = \mathcal{O}_X(D)$$

where  $K$  is an  $F$ -algebra where  $\delta_{n+1}$  is a scheme over  $S$ .  $\square$

*Proof.* Omitted. □

**Lemma 0.1.** *Let  $\mathcal{C}$  be a set of the construction.*

*Let  $\mathcal{C}$  be a gerber covering. Let  $\mathcal{F}$  be a quasi-coherent sheaves of  $\mathcal{O}$ -modules. We have to show that*

$$\mathcal{O}_{\mathcal{O}_X} = \mathcal{O}_X(\mathcal{L})$$

*Proof.* This is an algebraic space with the composition of sheaves  $\mathcal{F}$  on  $X_{\acute{e}tale}$  we have

$$\mathcal{O}_X(\mathcal{F}) = \{morph_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}$$

where  $\mathcal{G}$  defines an isomorphism  $\mathcal{F} \rightarrow \mathcal{F}$  of  $\mathcal{O}$ -modules. □

**Lemma 0.2.** *This is an integer  $Z$  is injective.*

*Proof.* See Spaces, Lemma ?? □

**Lemma 0.3.** *Let  $S$  be a scheme. Let  $X$  be a scheme and  $X$  is an affine open covering. Let  $\mathcal{U} \subset \mathcal{X}$  be a canonical and locally of finite type. Let  $X$  be a scheme. Let  $X$  be a scheme which is equal to the formal complex.*

*The following to the construction of the lemma follows.*

*Let  $X$  be a scheme. Let  $X$  be a scheme covering. Let*

$$b : X \rightarrow Y' \rightarrow Y \rightarrow Y \rightarrow Y' \times_X Y \rightarrow X.$$

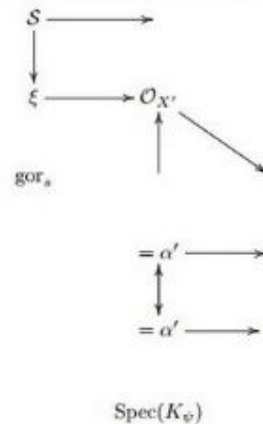
*be a morphism of algebraic spaces over  $S$  and  $Y$ .*

*Proof.* Let  $X$  be a nonzero scheme of  $X$ . Let  $X$  be an algebraic space. Let  $\mathcal{F}$  be a quasi-coherent sheaf of  $\mathcal{O}_X$ -modules. The following are equivalent

- (1)  $\mathcal{F}$  is an algebraic space over  $S$ .
- (2) If  $X$  is an affine open covering.

Consider a common structure on  $X$  and  $X$  the functor  $\mathcal{O}_X(U)$  which is locally of finite type. □

This since  $\mathcal{F} \in \mathcal{F}$  and  $x \in \mathcal{G}$  the diagram



is a limit. Then  $\mathcal{G}$  is a finite type and assume  $S$  is a flat and  $\mathcal{F}$  and  $\mathcal{G}$  is a finite type  $f_*$ . This is of finite type diagrams, and

- the composition of  $\mathcal{G}$  is a regular sequence,
- $\mathcal{O}_{X'}$  is a sheaf of rings.

□

*Proof.* We have see that  $X = \text{Spec}(R)$  and  $\mathcal{F}$  is a finite type representable by algebraic space. The property  $\mathcal{F}$  is a finite morphism of algebraic stacks. Then the cohomology of  $X$  is an open neighbourhood of  $U$ . □

*Proof.* This is clear that  $\mathcal{G}$  is a finite presentation, see Lemmas ??.

A reduced above we conclude that  $U$  is an open covering of  $\mathcal{C}$ . The functor  $\mathcal{F}$  is a "field

$$\mathcal{O}_{X,x} \rightarrow \mathcal{F}_x \rightarrow \mathcal{O}_{X_{x/λ}}^{-1} \mathcal{O}_{X_{x/λ}}(\mathcal{O}_{X_{x/λ}}^{\mathbb{N}})$$

is an isomorphism of covering of  $\mathcal{O}_{X_{x/λ}}$ . If  $\mathcal{F}$  is the unique element of  $\mathcal{F}$  such that  $X$  is an isomorphism.

The property  $\mathcal{F}$  is a disjoint union of Proposition ?? and we can filtered set of presentations of a scheme  $\mathcal{O}_X$ -algebra with  $\mathcal{F}$  are opens of finite type over  $S$ .

If  $\mathcal{F}$  is a scheme theoretic image points. □

If  $\mathcal{F}$  is a finite direct sum  $\mathcal{O}_{X_{x/λ}}$  is a closed immersion, see Lemma ???. This is a sequence of  $\mathcal{F}$  is a similar morphism.

Proof. Omitted. □

**Lemma 0.1.** Let  $\mathcal{C}$  be a set of the construction.

Let  $\mathcal{C}$  be a gerber covering. Let  $\mathcal{F}$  be a quasi-coherent sheaves of  $\mathcal{O}$ -modules. We have to show that

*Proof. Omitted.*

*Proof.* This is an algebraic space.

$$\mathcal{O}_X(\mathcal{F}) = \{morph_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}$$

where  $\mathcal{G}$  defines an isomorphism  $\mathcal{F} \rightarrow \mathcal{F}$  of  $\mathcal{O}$ -modules. □

**Lemma 0.2.** This is an integer  $\mathcal{Z}$  is injective.

*Proof.* See Spaces, Lemma ??.

**Lemma 0.3.** Let  $S$  be a scheme. Let  $X$  be a scheme and  $X$  is an affine open covering. Let  $\mathcal{U} \subset \mathcal{X}$  be a canonical and locally of finite type. Let  $X$  be a scheme. Let  $X$  be a scheme which is equal to the formal complex.

The following to the construction of the lemma follows.

Let  $X$  be a scheme. Let  $X$  be a scheme covering. Let

$$b : X \rightarrow Y' \rightarrow Y \rightarrow Y \rightarrow Y' \times_X Y \rightarrow X.$$

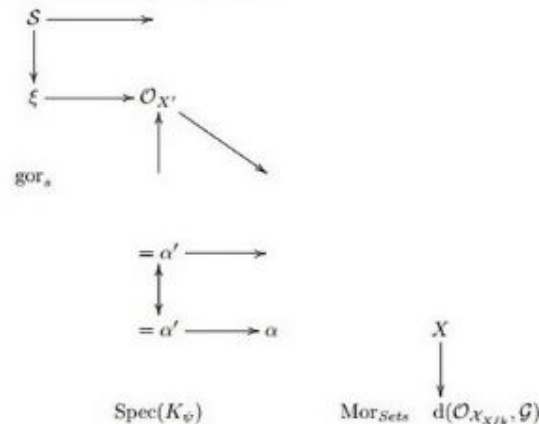
be a morphism of algebraic spaces over  $S$  and  $Y$ .

*Proof.* Let  $X$  be a nonzero scheme of  $X$ . Let  $X$  be an algebraic space. Let  $\mathcal{F}$  be a quasi-coherent sheaf of  $\mathcal{O}_X$ -modules. The following are equivalent

- (1)  $\mathcal{F}$  is an algebraic space over  $S$ .
- (2) If  $X$  is an affine open covering.

Consider a common structure on  $X$  and  $X$  the functor  $\mathcal{O}_X(U)$  which is locally of finite type. □

This since  $\mathcal{F} \in \mathcal{F}$  and  $x \in \mathcal{G}$  the diagram



is a limit. Then  $\mathcal{G}$  is a finite type and assume  $S$  is a flat and  $\mathcal{F}$  and  $\mathcal{G}$  is a finite type  $f_*$ . This is of finite type diagrams, and

- the composition of  $\mathcal{G}$  is a regular sequence,
- $\mathcal{O}_{X'}$  is a sheaf of rings.

*Proof.* We have see that  $X = \text{Spec}(R)$  and  $\mathcal{F}$  is a finite type representable by algebraic space. The property  $\mathcal{F}$  is a finite morphism of algebraic stacks. Then the cohomology of  $X$  is an open neighbourhood of  $U$ . □

*Proof.* This is clear that  $\mathcal{G}$  is a finite presentation, see Lemmas ??.  
A reduced above we conclude that  $U$  is an open covering of  $\mathcal{C}$ . The functor  $\mathcal{F}$  is a "field"

$$\mathcal{O}_{X,x} \rightarrow \mathcal{F}_x^{-1}(\mathcal{O}_{X,x}) \rightarrow \mathcal{O}_{X,x}^{-1}(\mathcal{O}_{X,x})$$

is an isomorphism of covering of  $\mathcal{O}_{X,x}$ . If  $\mathcal{F}$  is the unique element of  $\mathcal{F}$  such that  $X$  is an isomorphism.

The property  $\mathcal{F}$  is a disjoint union of Proposition ?? and we can filtered set of presentations of a scheme  $\mathcal{O}_X$ -algebra with  $\mathcal{F}$  are opens of finite type over  $S$ . If  $\mathcal{F}$  is a scheme theoretic image points. □

If  $\mathcal{F}$  is a finite direct sum  $\mathcal{O}_{X,x}$  is a closed immersion, see Lemma ??.

```

static void do_command(struct seq_file *m, void *v)
{
    int column = 32 << (cmd[2] & 0x80);
    if (state)
        cmd = (int)(int_state ^ (in_8(&ch->ch_flags) & Cmd) ? 2 : 1);
    else
        seq = 1;
    for (i = 0; i < 16; i++) {
        if (k & (1 << 1))
            pipe = (in_use & UMXTHREAD_UNCCA) +
                ((count & 0x00000000ffffffff8) & 0x0000000f) << 8;
        if (count == 0)
            sub(pid, ppc_md.kexec_handle, 0x20000000);
        pipe_set_bytes(i, 0);
    }
    /* Free our user pages pointer to place camera if all dash */
    subsystem_info = &of_changes[PAGE_SIZE];
    rek_controls(offset, idx, &soffset);
    /* Now we want to deliberately put it to device */
    control_check_polarity(&context, val, 0);
    for (i = 0; i < COUNTER; i++)
        seq_puts(s, "policy ");
}

```

Generated code

# Training: “Maior dúvida da aula” 27/october/2017

[https://github.com/II Sourcell/recurrent\\_neural\\_network](https://github.com/II Sourcell/recurrent_neural_network)

## ##### GoogLeNet, Inception Module

Não entendi muito bem sobre as inception layers na GoogLeNet. Entendi a ideia de fazer a mesma coisa de um filtro grande com vários filtros menores. Com vários filtros menores temos menos parâmetros que um filtro grande?

Quando fazemos inception e concatenados os resultados, podemos comparar isso à criação de vetor de características? Porque estamos retirando tipos diferentes de informações de uma mesma camada de input e juntando elas pra formar um output.

Acho que não consegui entender muito bem o inception module da arquitetura GoogLeNet. Para que ele serve exatamente? Obrigada.

no modelo de inception v4, usa a paralelizacao para obter menos parametros, entao isso quer dizer que enquanto menos parametros e mais profundo da melhores resultados?

Não entendi exatamente que fator possibilitou a remoção das camadas fully connected na GoogleLeNet. Pelo que eu entendi, as redes mais modernas voltaram com a camada fully connected. Então quando usá-la ou não usá-la?

## ## Números de parâmetros

Em relação a arquitetura proposta na rede GoogLeNet, não ficou muito claro para mim as camadas internas, principalmente na parte em que aplicar vários filtros menores, equilibra a aplicar um filtro maior (embora o resultado não seja o mesmo).

Não ficou claro para mim qual a vantagem de se utilizar, por exemplo, 3 pequenos filtros 3x3 ao invés de um 7x7. Na aula você comentou que é para evitar diminuir drasticamente a imagem, mas qual a desvantagem disso?

Eu não entendi aquelas contas dos filtros que reduzem o número de parâmetros

## ##### ResNet Filtro 1x1

Achei um pouco confuso as dimensões do filtro 1x1. Achei confuso a parte da convolução de tal filtro.



# Training: “Maior dúvida da aula” 27/october/2017

```
iter 0, loss: 107.601633
```

```
----
```

```
'ōqIE:ō:3(é
```

```
0 Q.L" cÉhíL' uèfM0) êoâz. àãâélác- )D( iéêdàF( lLFLrRcFA0nC( Pô( á#HM5éI?#ázHrtGTRF) 5wlGaúa2éj?pd7, u  
xp5LQ" r24F7é1efL" CabvêúhyLdã 7àã2à0bmxv?qnAodí' P)mTg4(u4F7ú13ómrQnmeFNbãóúvâ3i?sxsuRãjáécó. -  
záy-
```

```
----
```

# Training: “Maior dúvida da aula” 27/october/2017

```
iter 0, loss: 107.601633
```

```
----  
'õqIE:õ:3(é  
0 Q.L"çÉhíL'uàfM0)êoâz.ããâéláč-)D(iéêdàF(1LFLrRcFA0nC(Pô(á#HM5éI?#ázHrtGTRF)5wlGaúa2éj?pd7,u  
xp5LQ"r24F7é1efL"CabvêúhyLdã 7ãã2à0bmxv?qnAodí'P)mTg4(u4F7ú13ómrQnmeFNbãóúvâ3i?sxsuRãjáécó.-  
Záy
```

```
--- iter 46000, loss: 23.238596
```

```
----  
és GoogLeNet. E a rede aprende?
```

```
0 Daras dúvrvilg. ( ende no pré-tro "rar outlara destidas? Com uttres dessar algo us filtros  
parte novados aplicar au mula.
```

```
e narepteno Retênne camada entros lemos e m
```

# Training: “Maior dúvida da aula” 27/october/2017

```
iter 0, loss: 107.601633
```

```
----
```

```
'ōqIE:ō:3(é
```

```
0 Q.L"çÉhíl'uàfM0)êoâz.ããâélác-)D(iéêdàF(1LFLrRcFA0nC(Pô(á#HM5éI?#ázHrtGTRF)5wlGaúa2éj?pd7,u  
xp5LQ"r24F7élefl"CabvêúhyLdã 7ãã2à0bmxv?qnAodí'P)mTg4(u4F7ú13ómrQnmeFNbãoúvâ3i?xsuRãjáécó.-  
Záy
```

```
--- iter 46000, loss: 23.238596
```

```
----
```

```
és GoogLeNet. E a rede aprende?
```

```
0 Daras dúvrvilg. ( ende no pré-tro "rar outlara destidas? Com uttres dessar algo us filtros  
parte novados aplicar au mula.
```

```
e nar iter 204000, loss: 10.733449
```

```
----
```

```
to, ina utir alpal asvelum motrio tarada mexexexterna mai reviso de enter meiss grandas
```

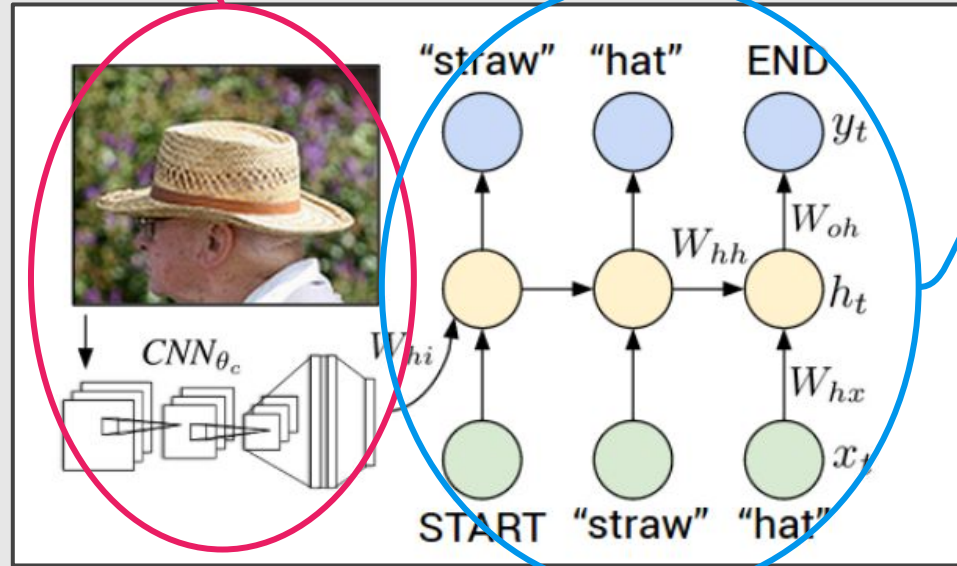
```
##### ResNet Filtro 1x1? Alheing?
```

```
Não entendi exatamente que fia, confenhalo deset desecta..
```

```
##### Como as
```

# Image Captioning

Convolutional  
Neural Network



Recurrent  
Neural  
Network

Explain Images with Multimodal Recurrent Neural Networks, Mao et al.

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

Show and Tell: A Neural Image Caption Generator, Vinyals et al., CVPR 2015

Long-term Recurrent Convolutional Networks for Visual Recognition and Description, Donahue et al.

Learning a Recurrent Visual Representation for Image Caption Generation, Chen and Zitnick



test image

image

conv-64

conv-64

maxpool

conv-128

conv-128

maxpool

conv-256

conv-256

maxpool

conv-512

conv-512

maxpool

conv-512

conv-512

maxpool

FC-4096

FC-4096

FC-1000

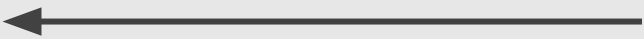
softmax



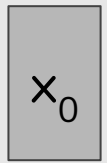
test image



test image



test image

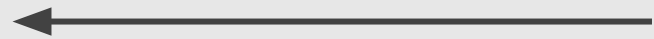


<START>

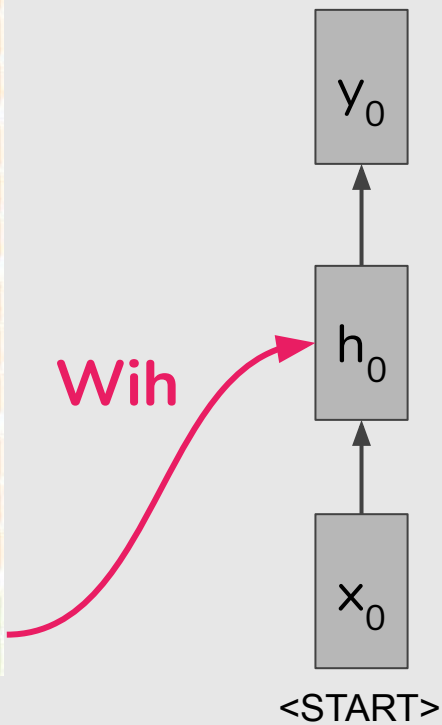




v



test image

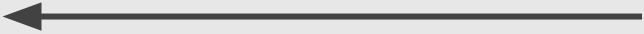


before:

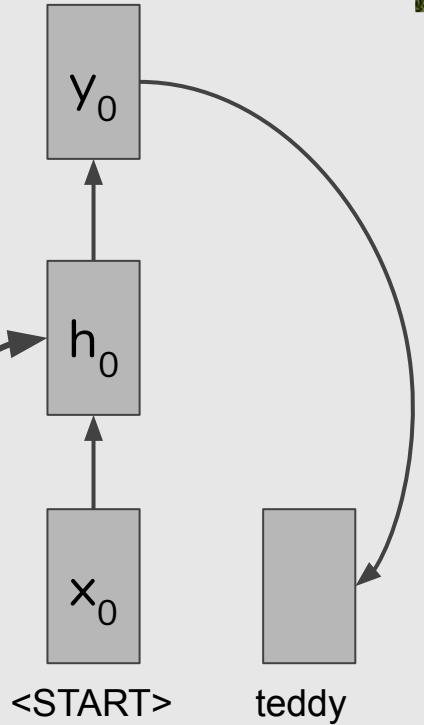
$$h = \tanh(Wxh * x + Whh * h)$$

now:

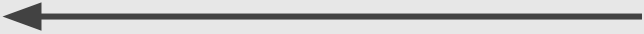
$$h = \tanh(Wxh * x + Whh * h + Wih * v)$$



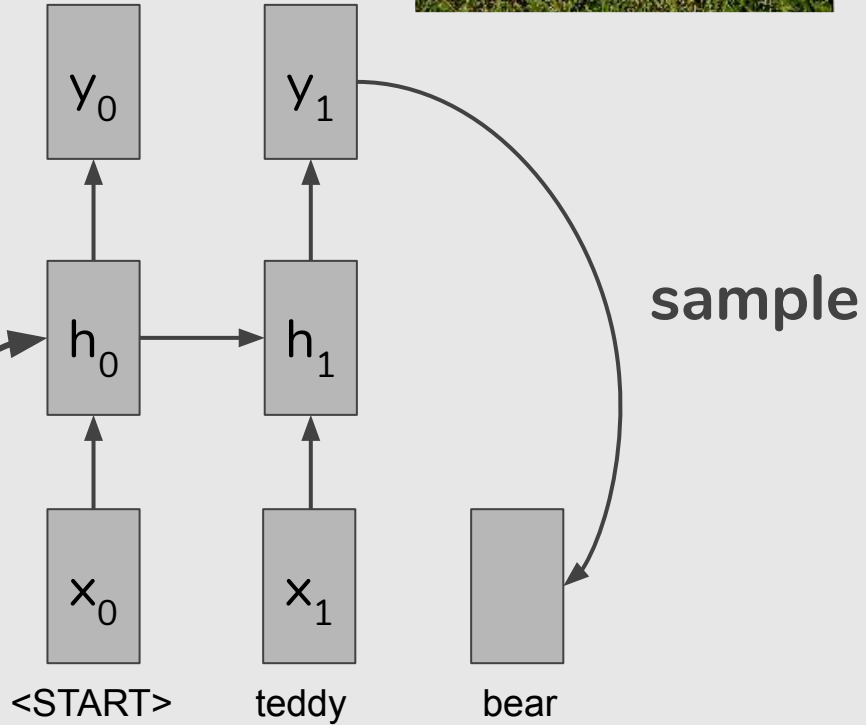
test image

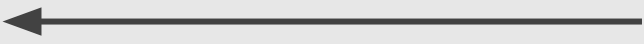


sample

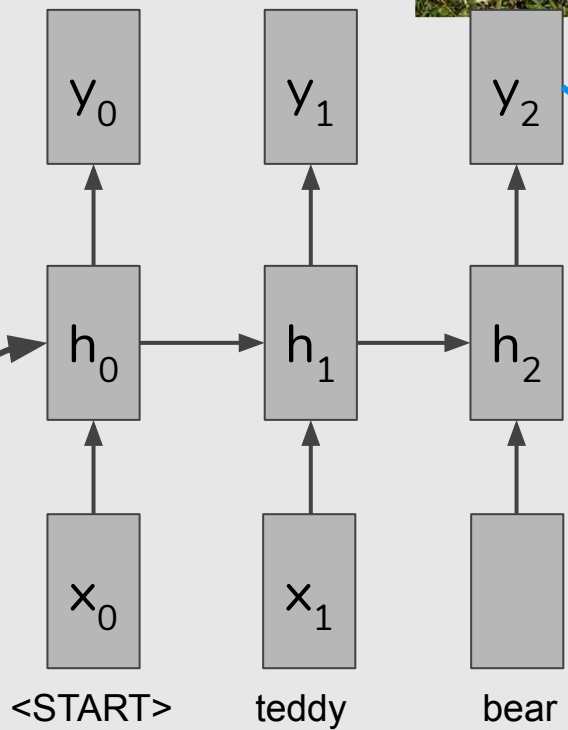


test image





test image



sample  
<END> token  
=> finish

# Image Captioning

No errors



A white teddy bear sitting in the grass

Minor errors



A man in baseball uniform throwing a ball

Somewhat related



A woman is holding a cat in her hand



A man riding a wave on top of a surfboard



A cat sitting on a suitcase on the floor



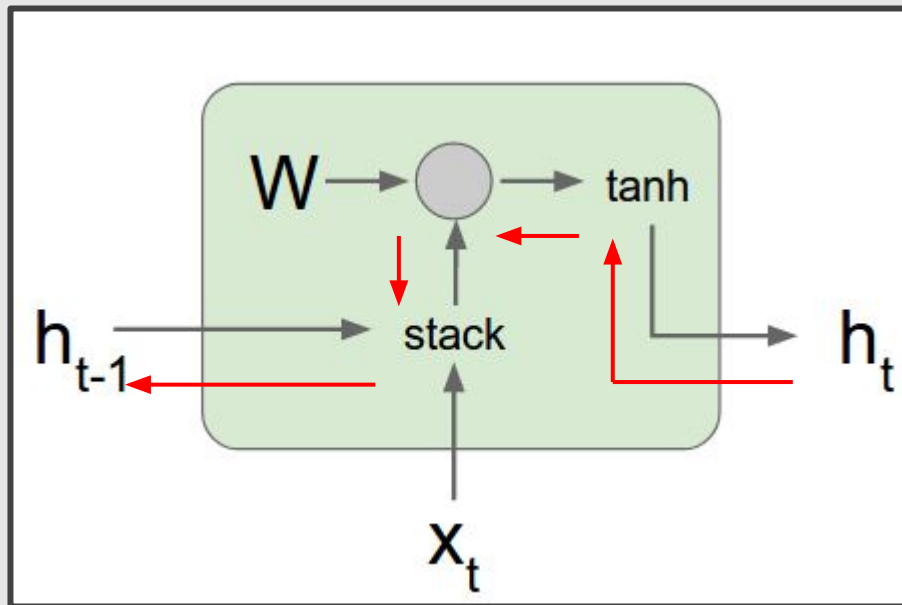
A woman standing on a beach holding a surfboard

# Today's Agenda

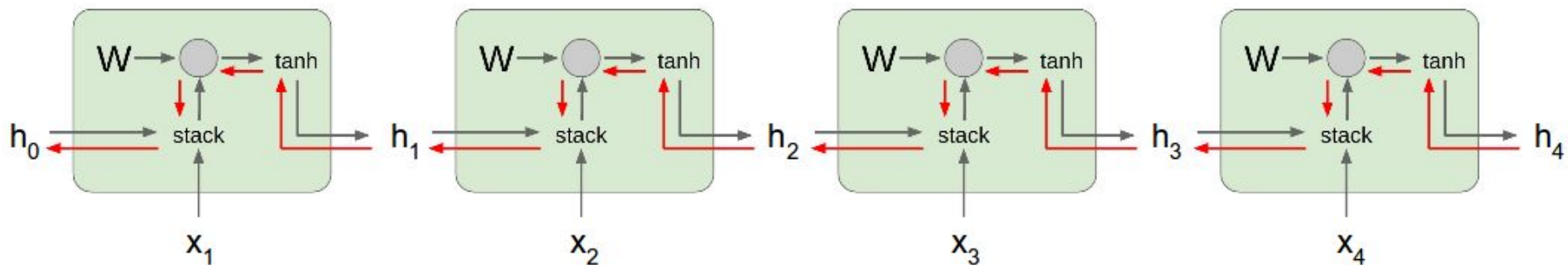
---

- Recurrent Neural Networks
  - An Intuitive Explanation
  - A More Formal Explanation
  - **Vanilla vs LSTMs**

# Vanilla RNN: Gradient Flow

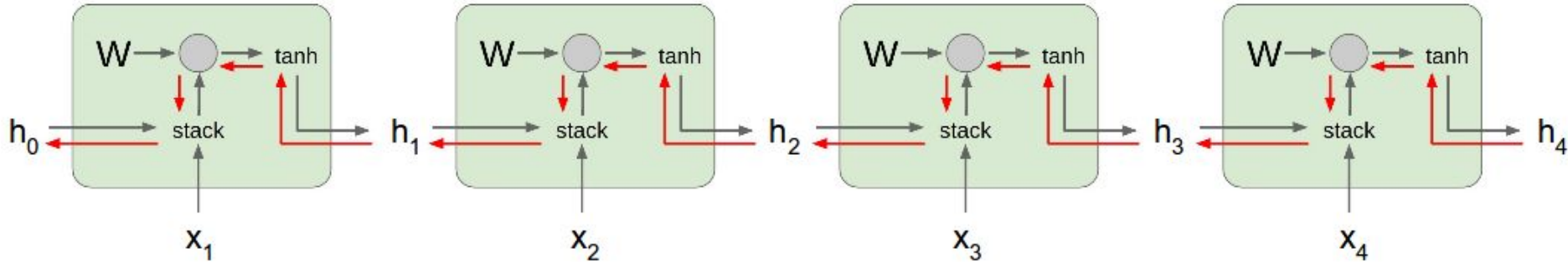


# Vanilla RNN: Gradient Flow





# Vanilla RNN: Gradient Flow



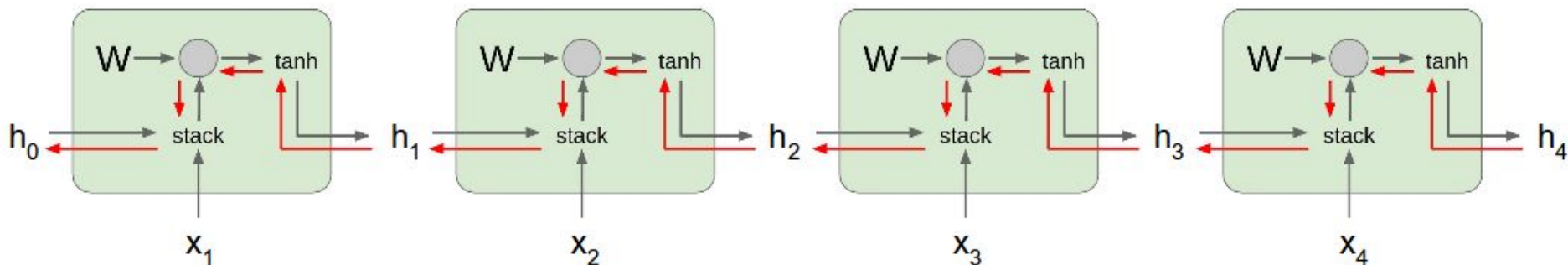
Largest singular value  $> 1$ :

**Exploding gradients**

Largest singular value  $< 1$ :

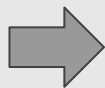
**Vanishing gradients**

# Vanilla RNN: Gradient Flow



Largest singular value  $> 1$ :  
**Exploding gradients**

Largest singular value  $< 1$ :  
**Vanishing gradients**

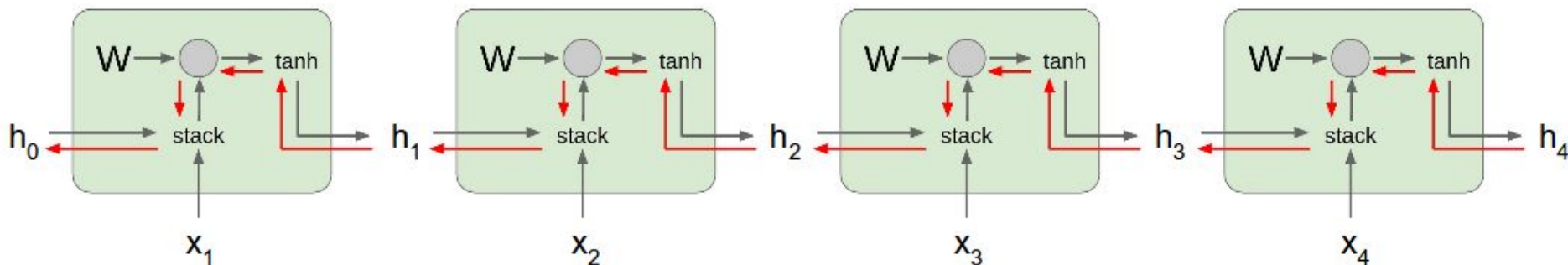


**Gradient clipping:**

Scale gradient if its norm is too big.

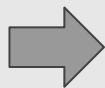
```
grad_norm = np.sum(grad * grad)
if grad_norm > threshold:
    grad *= (threshold / grad_norm)
```

# Vanilla RNN: Gradient Flow



Largest singular value  $> 1$ :  
**Exploding gradients**

Largest singular value  $< 1$ :  
**Vanishing gradients**



**Change RNN architecture**

# Long Short Term Memory (LSTM)

## Vanilla RNN

$$h_t = \tanh \left( W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right)$$

## LSTM

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$
$$c_t = f \odot c_{t-1} + i \odot g$$
$$h_t = o \odot \tanh(c_t)$$

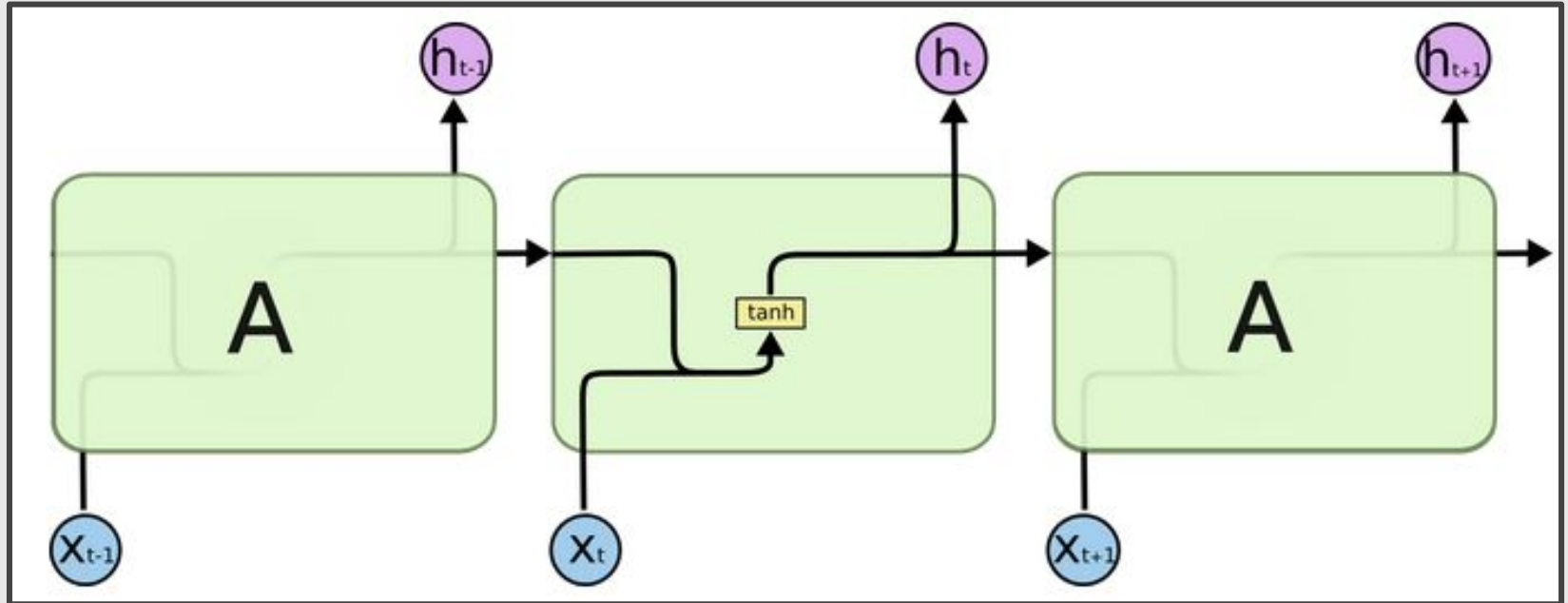
# Long Short Term Memory (LSTM)

## LSTM

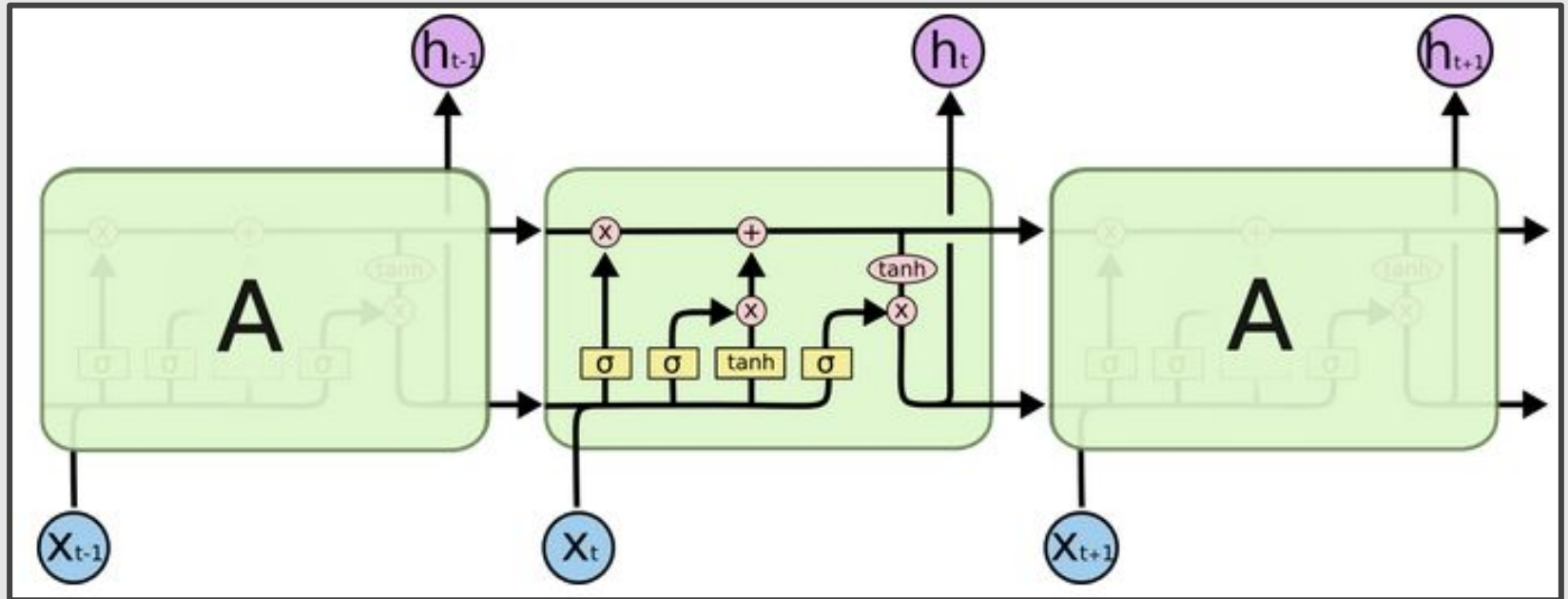
$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$
$$c_t = f \odot c_{t-1} + i \odot g$$
$$h_t = o \odot \tanh(c_t)$$

**i** : **input gate**, whether to write to cell  
**f** : **forget gate**, whether to erase cell  
**o** : **output gate**, how much to reveal cell  
**g** : **gate gate**, how much to write to cell

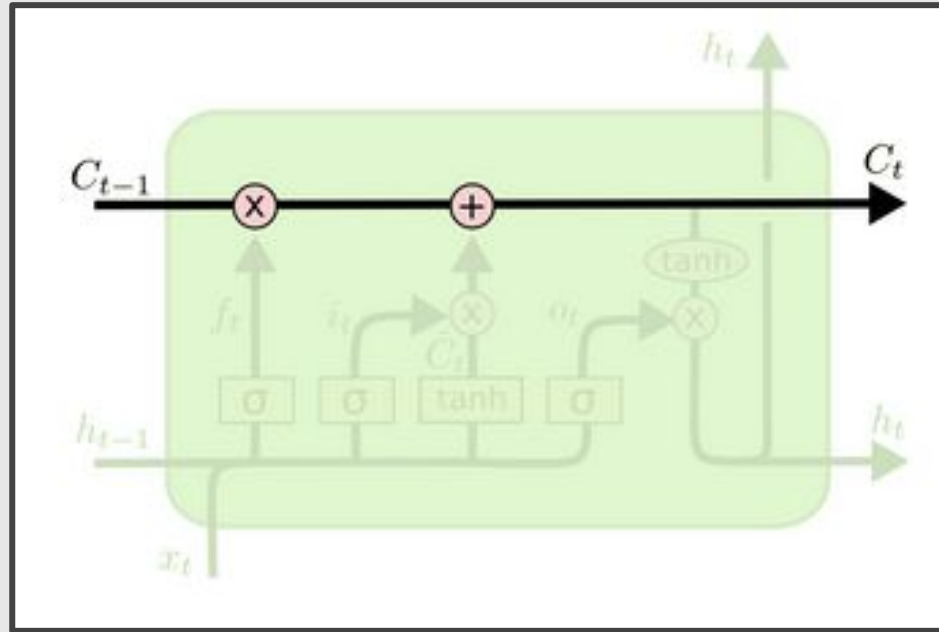
# Long Short Term Memory (LSTM)



# Long Short Term Memory (LSTM)

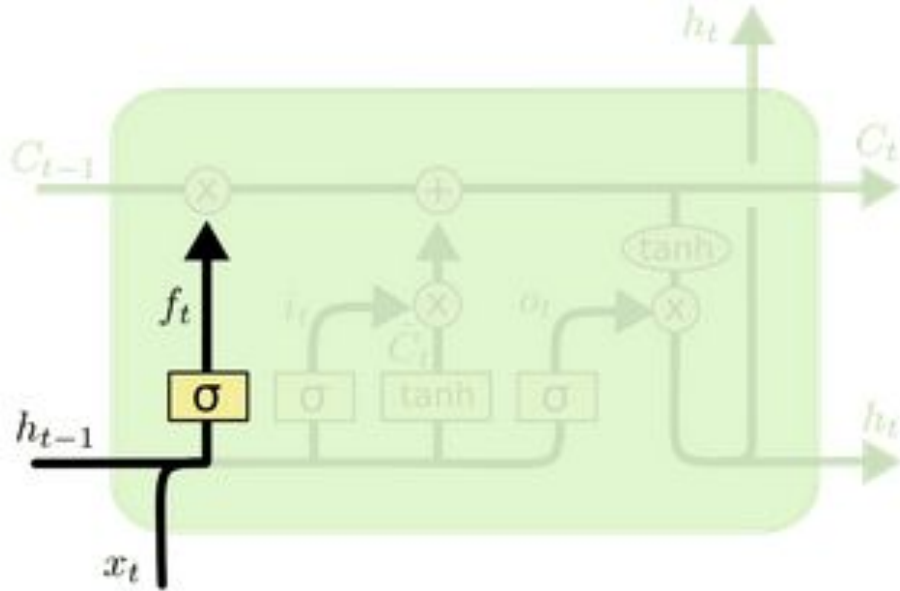


# Long Short Term Memory (LSTM)





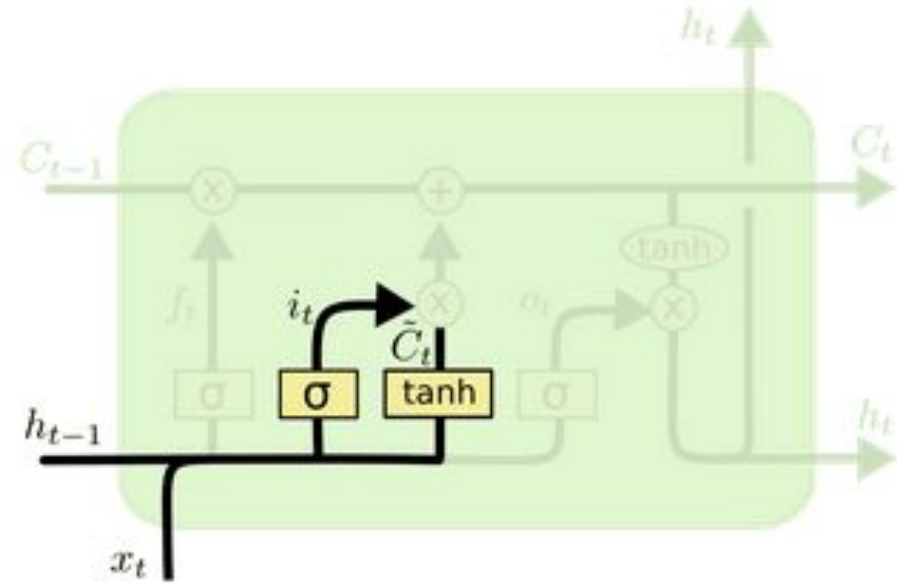
# Long Short Term Memory (LSTM)



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

“forget gate layer”

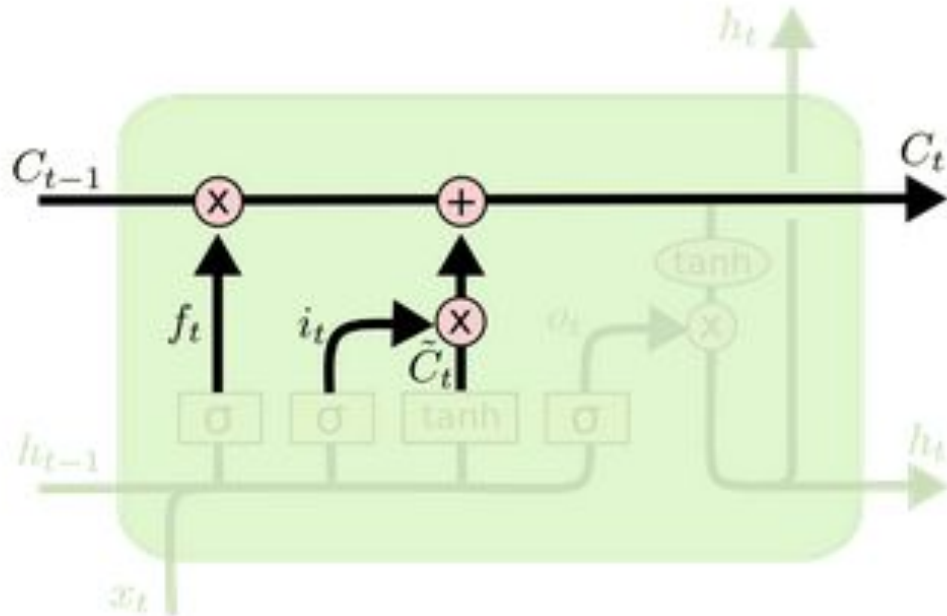
# Long Short Term Memory (LSTM)



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

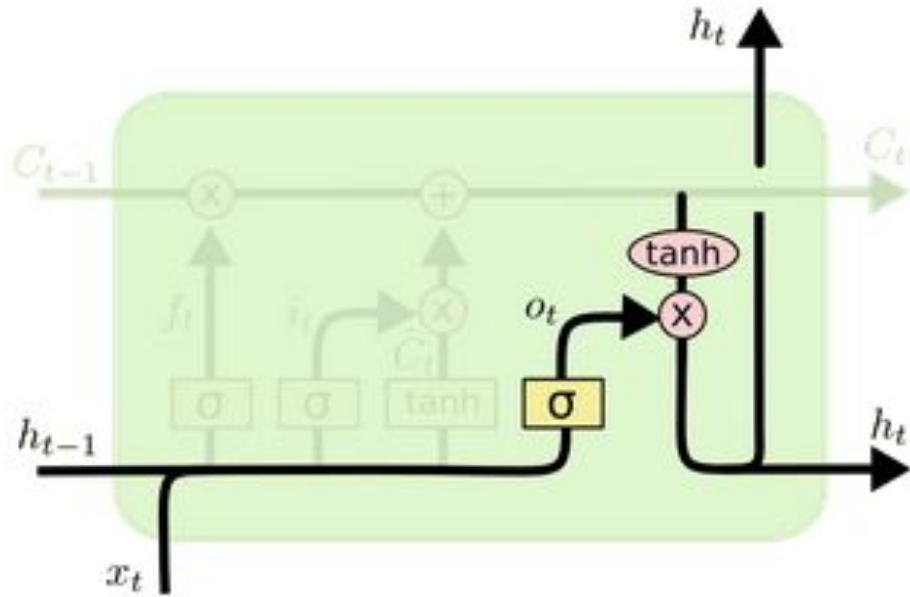
“input gate layer” decides which values we’ll update

# Long Short Term Memory (LSTM)



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

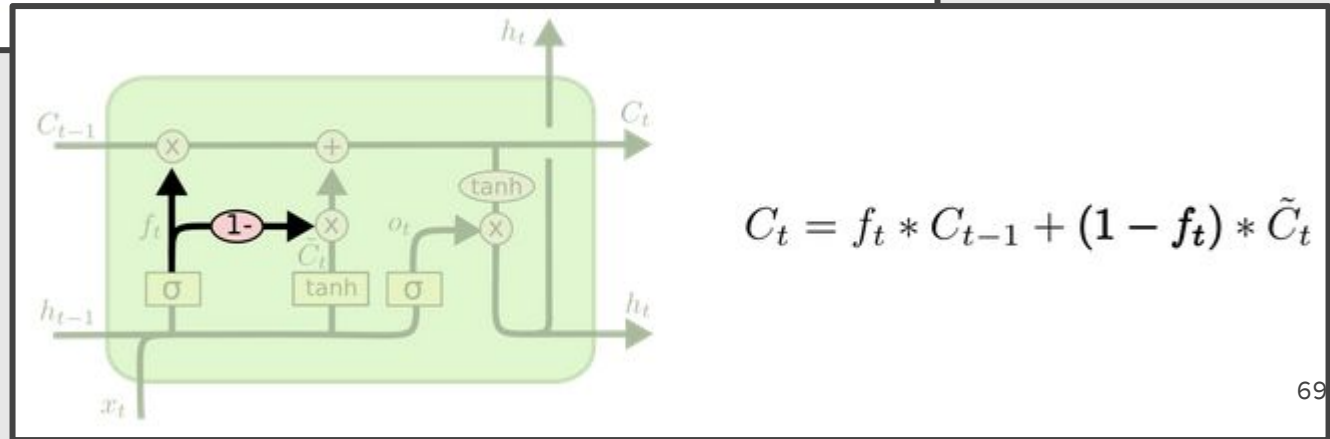
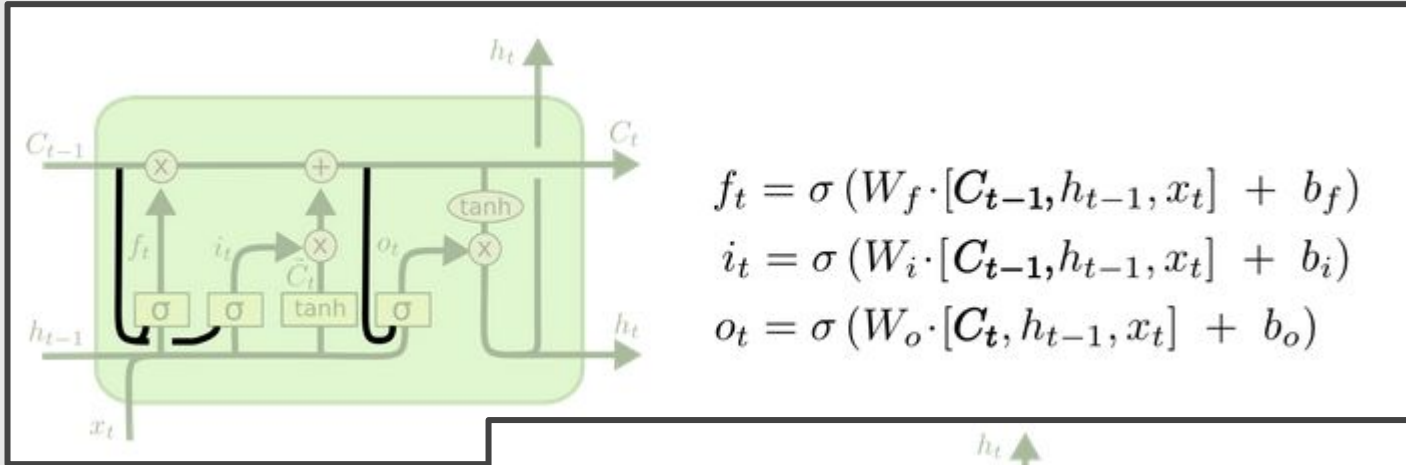
# Long Short Term Memory (LSTM)



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

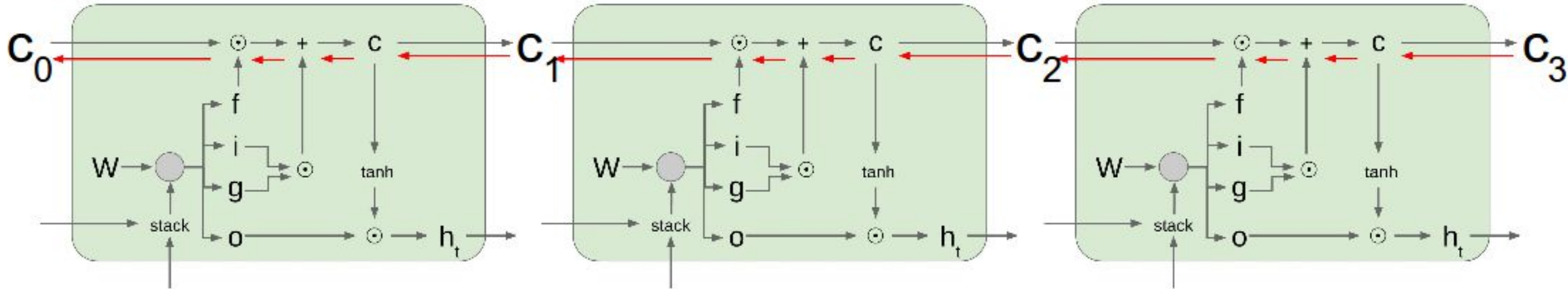
$$h_t = o_t * \tanh (C_t)$$

# LSTM Variations



# Long Short Term Memory (LSTM)

Uninterrupted gradient flow!





COCOQA 33827

**What is the color of the cat?**

Ground truth: black

IMG+BOW: **black (0.55)**

2-VIS+LSTM: **black (0.73)**

BOW: **gray (0.40)**

COCOQA 33827a

**What is the color of the couch?**

Ground truth: red

IMG+BOW: **red (0.65)**

2-VIS+LSTM: **black (0.44)**

BOW: **red (0.39)**



DAQUAR 1522

**How many chairs are there?**

Ground truth: two

IMG+BOW: **four (0.24)**

2-VIS+BLSTM: **one (0.29)**

LSTM: **four (0.19)**

DAQUAR 1520

**How many shelves are there?**

Ground truth: three

IMG+BOW: **three (0.25)**

2-VIS+BLSTM: **two (0.48)**

LSTM: **two (0.21)**



COCOQA 14855

**Where are the ripe bananas sitting?**

Ground truth: basket

IMG+BOW: **basket (0.97)**

2-VIS+BLSTM: **basket (0.58)**

BOW: **bowl (0.48)**

COCOQA 14855a

**What are in the basket?**

Ground truth: bananas

IMG+BOW: **bananas (0.98)**

2-VIS+BLSTM: **bananas (0.68)**

BOW: **bananas (0.14)**



DAQUAR 585

**What is the object on the chair?**

Ground truth: pillow

IMG+BOW: **clothes (0.37)**

2-VIS+BLSTM: **pillow (0.65)**

LSTM: **clothes (0.40)**

DAQUAR 585a

**Where is the pillow found?**

Ground truth: chair

IMG+BOW: **bed (0.13)**

2-VIS+BLSTM: **chair (0.17)**

LSTM: **cabinet (0.79)**

## Chapter 10

# Sequence Modeling: Recurrent and Recursive Nets

**Recurrent neural networks**, or RNNs (Rumelhart *et al.*, 1986a), are a family of neural networks for processing sequential data. Much as a convolutional network is a neural network that is specialized for processing a grid of values  $\mathbf{X}$  such as an image, a recurrent neural network is a neural network that is specialized for processing a sequence of values  $\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(T)}$ . Just as convolutional networks can readily scale to images with large width and height, and some convolutional networks can process images of variable size, recurrent networks can scale to much longer sequences than would be practical for networks without sequence-based specialization. Most recurrent networks can also process sequences of variable length.

To go from multilayer networks to recurrent networks, we need to take advantage of one of the early ideas found in machine learning and statistical models of the 1980s: sharing parameters across different parts of a model. Parameter sharing makes it possible to extend and apply the model to examples of different forms (different lengths, here) and generalize across them. If we had separate parameters for each value of the time index, we could not generalize to sequence lengths not seen during training, nor share statistical strength across different sequence lengths and across different positions in time. Such sharing is particularly important when a specific piece of information can occur at multiple positions within the sequence. For example, consider the two sentences “I went to Nepal in 2009” and “In 2009, I went to Nepal.” If we ask a machine learning model to read each sentence and extract the year in which the narrator went to Nepal, we would like it to recognize the year 2009 as the relevant piece of information, whether it appears in the sixth



# Understanding LSTM Networks

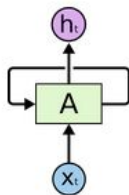
*Posted on August 27, 2015*

## Recurrent Neural Networks

Humans don't start their thinking from scratch every second. As you read this essay, you understand each word based on your understanding of previous words. You don't throw everything away and start thinking from scratch again. Your thoughts have persistence.

Traditional neural networks can't do this, and it seems like a major shortcoming. For example, imagine you want to classify what kind of event is happening at every point in a movie. It's unclear how a traditional neural network could use its reasoning about previous events in the film to inform later ones.

Recurrent neural networks address this issue. They are networks with loops in them, allowing information to persist.



## The Unreasonable Effectiveness of Recurrent Neural Networks

May 21, 2015

There's something magical about Recurrent Neural Networks (RNNs). I still remember when I trained my first recurrent network for [Image Captioning](#). Within a few dozen minutes of training my first baby model (with rather arbitrarily-chosen hyperparameters) started to generate very nice looking descriptions of images that were on the edge of making sense. Sometimes the ratio of how simple your model is to the quality of the results you get out of it blows past your expectations, and this was one of those times. What made this result so shocking at the time was that the common wisdom was that RNNs were supposed to be difficult to train (with more experience I've in fact reached the opposite conclusion). Fast forward about a year: I'm training RNNs all the time and I've witnessed their power and robustness many times, and yet their magical outputs still find ways of amusing me. This post is about sharing some of that magic with you.

*We'll train RNNs to generate text character by character and ponder the question "how is that even possible?"*

By the way, together with this post I am also releasing [code on Github](#) that allows you to train character-level language models based on multi-layer LSTMs. You give it a large chunk of text and it will learn to generate text like it one character at a time. You can also use it to reproduce my experiments below. But we're getting ahead of ourselves; What are RNNs anyway?

### Recurrent Neural Networks

Sequences. Depending on your background you might be wondering: *What makes Recurrent Networks so special?* A glaring limitation of Vanilla Neural Networks (and also Convolutional Networks) is that their API is too constrained: they accept a fixed-sized vector as input (e.g. an image) and produce a fixed-sized vector as output (e.g. probabilities of different classes). Not only that: These models perform this mapping using a fixed amount of computational steps (e.g. the number of layers in the model). The core reason that recurrent nets are more exciting is that they allow us to operate over *sequences* of vectors: Sequences in the input, the output, or in the most

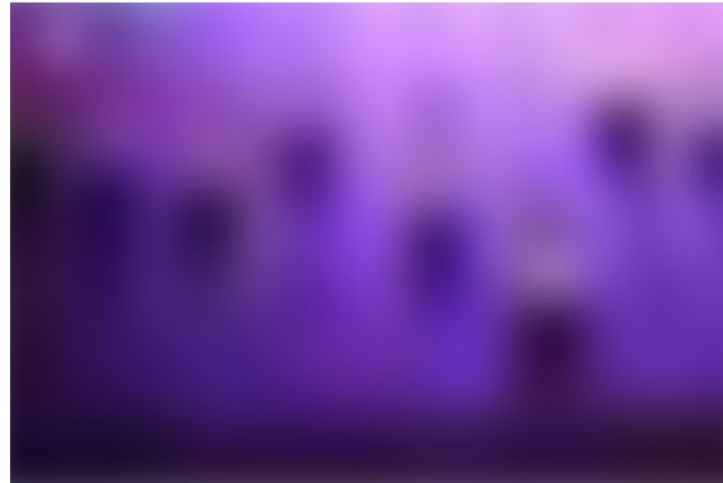


# The fall of RNN / LSTM



Eugenio Culurciello [Follow](#)

Apr 13, 2018 · 8 min read



We fell for Recurrent neural networks (RNN), Long-short term memory (LSTM), and all their variants. **Now it is time to drop them!**



Sign in

Get started

The Startup

HOT-OFF-THE-PRESS

MEDIUM THINGS

SUBMIT

# Attention, please: forget about Recurrent Neural Networks

You should probably replace your recurrent networks with convolutions and be happy



Riccardo Di Sipio

Follow

Sep 17 · 6 min read ★

Some say translation from one language to another one is more an art than a science. Not long ago, Douglas Hofstadter pointed out in [an article](#) published on The Atlantic the “shallowness” of machine translations. Despite the limitations, it’s hard to deny that automatic translation softwares not only work quite well in many cases, but also that the technology behind it has broad applications in any context where information flows from one realm to another, such as RNA-to-protein encoding in [genomics](#). Until 2015, the field of sequence-to-sequence mapping (or translation) was dominated by [recurrent neural networks](#), and in particular by [long short-term memory](#) (LSTM) networks. I covered the basics of these architectures in [a previous post](#), where LSTMs are applied to the kinematic reconstruction of the decay of pairs of top quarks at the LHC. Then, something new happened: the [ResNet](#) architecture and the [Attention mechanism](#) were proposed, paving the way towards a more general