



CS 7643: Deep Learning

Topics:

- Recurrent Neural Networks (RNNs)
- BackProp Through Time (BPTT)
- Vanishing / Exploding Gradients

Dhruv Batra
Georgia Tech

Administrativa

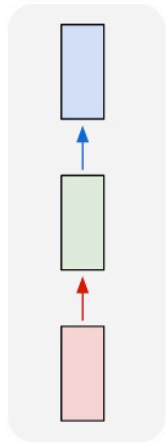
- HW3 + PS3 out
 - Due 10/28
 - Last PS and HW
 - Focus on projects after that

Plan for Today

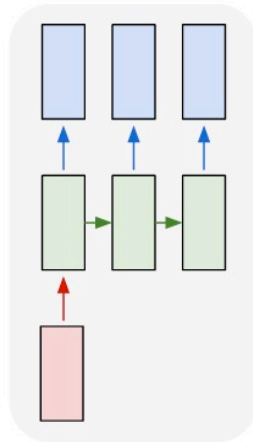
- Model
 - Recurrent Neural Networks (RNNs)
- Learning
 - BackProp Through Time (BPTT)
 - Vanishing / Exploding Gradients

New Topic: RNNs

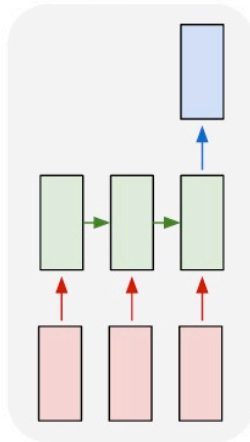
one to one



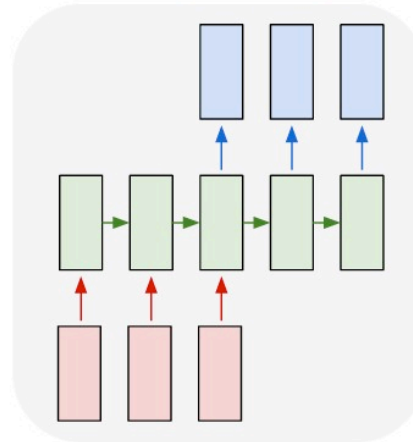
one to many



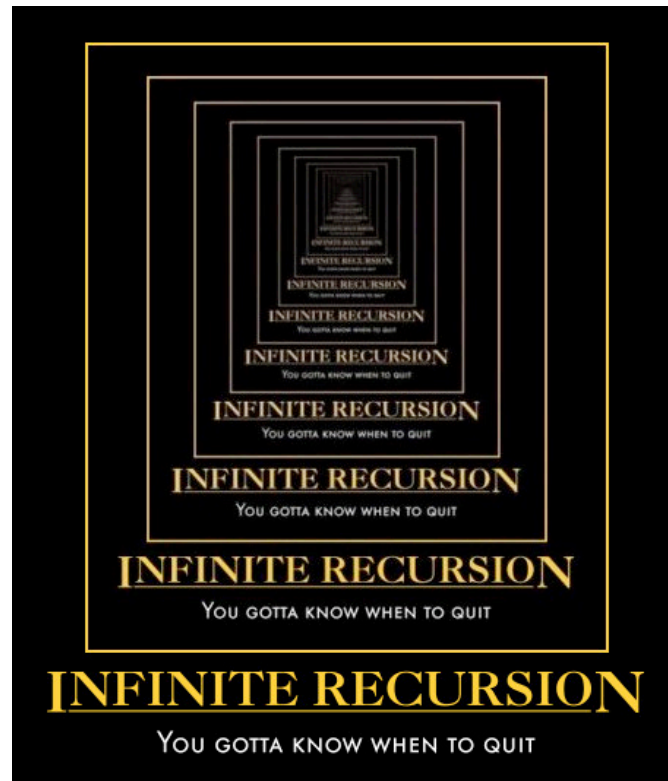
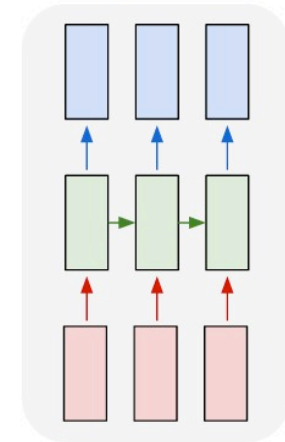
many to one



many to many



many to many



Synonyms

- Recurrent Neural Networks (RNNs)
- Recursive Neural Networks
 - General family; think graphs instead of chains
- Types:
 - “Vanilla” RNNs
 - Long Short Term Memory (LSTMs)
 - Gated Recurrent Units (GRUs)
 - ...
- Algorithms
 - BackProp Through Time (BPTT)
 - BackProp Through Structure (BPTS)

Recurrent
←

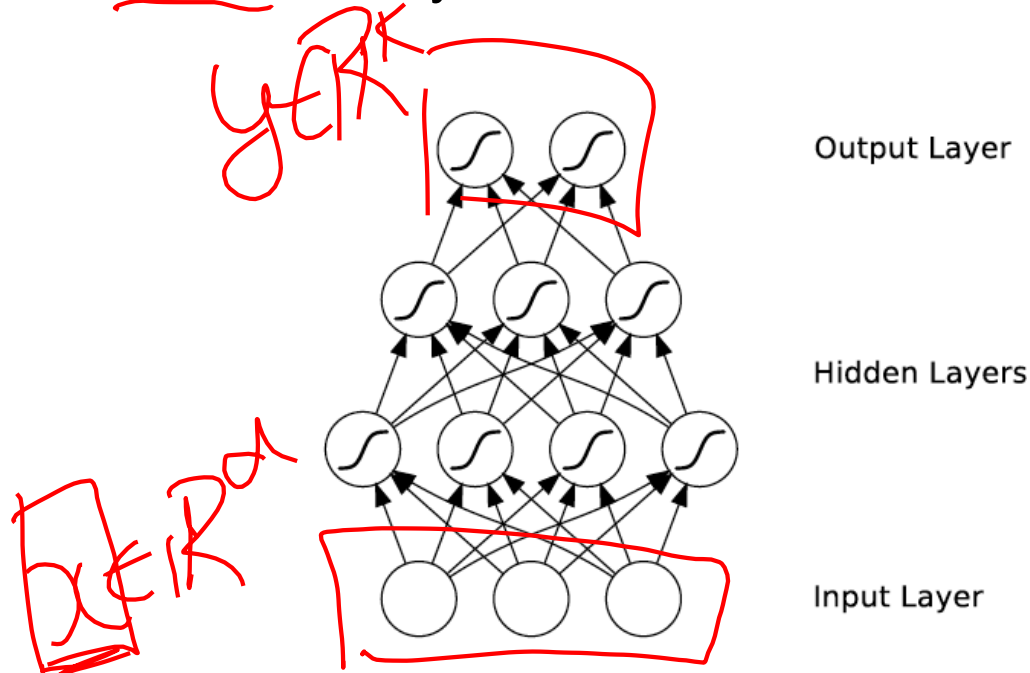
What's wrong with MLPs?

- Problem 1: Can't model sequences

- Fixed-sized Inputs & Outputs
- No temporal structure

- Problem 2: Pure feed-forward processing

- No "memory", no feedback



Sequences are everywhere...

Foreign Minister. → FOREIGN MINISTER.

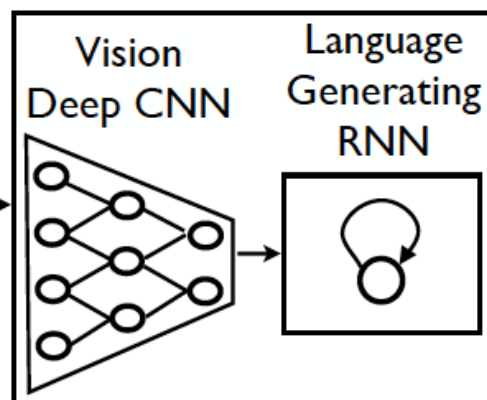
[Waveform] → THE SOUND OF

$x =$ $a_1=2$ $a_2=0$ $a_3=1$ $a_4=3$ $a_5=4$ $a_6=2$ $a_7=5$
bringen sie bitte das auto zurück .

↙

$y =$ please return the car .

Even where you might not expect a sequence...

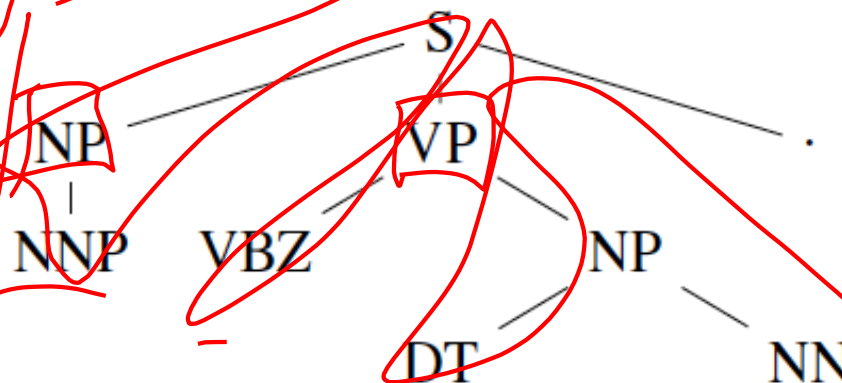


A group of people shopping at an outdoor market.

There are many vegetables at the fruit stand.

John has a dog .

→



John has a dog .

→

(S (NP NNP)_{NP} (VP VBZ (NP DT NN)_{NP})_{VP} .)_S

Even where you might not expect a sequence...

Classify images by taking a series of “glimpses”



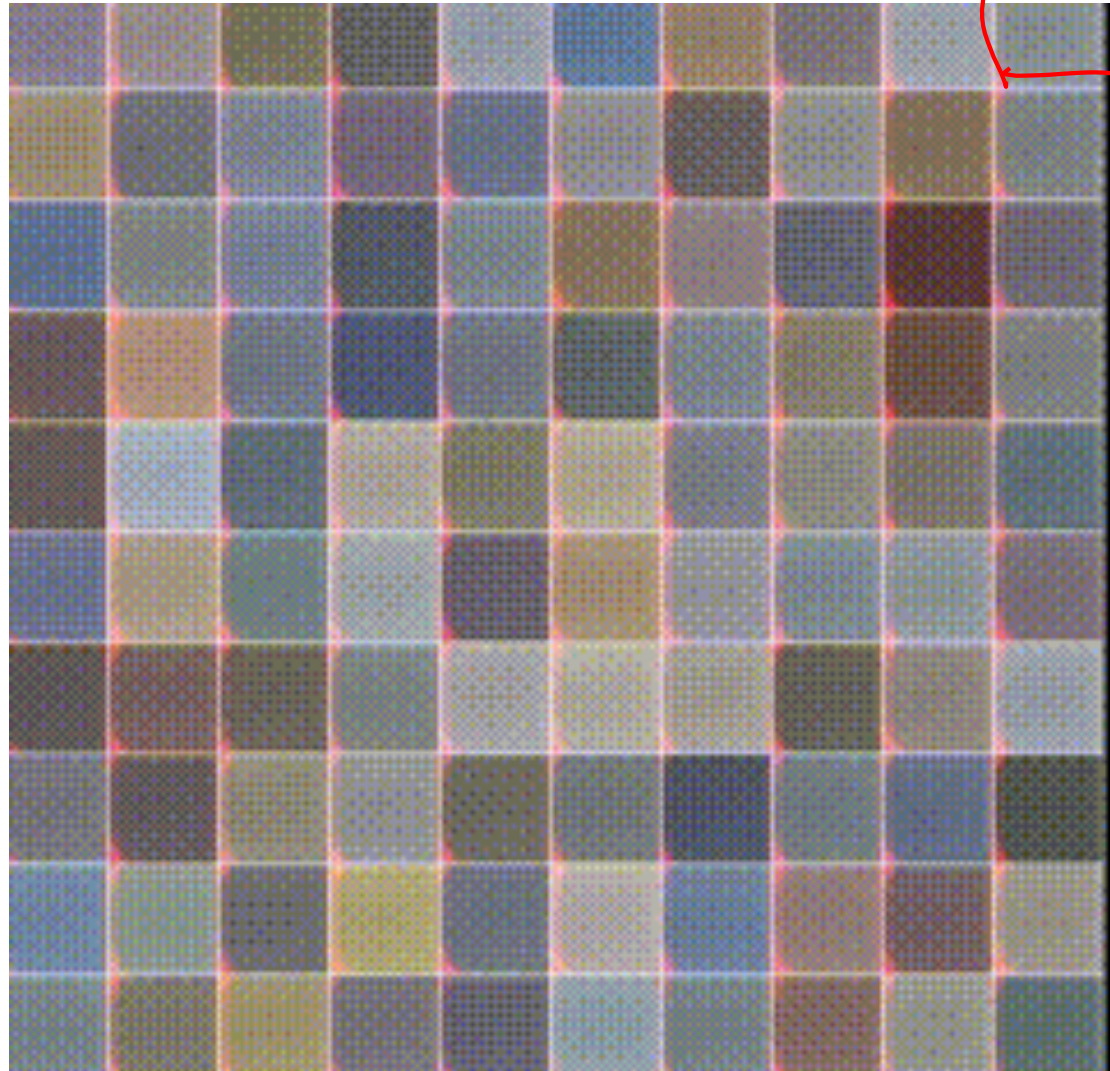
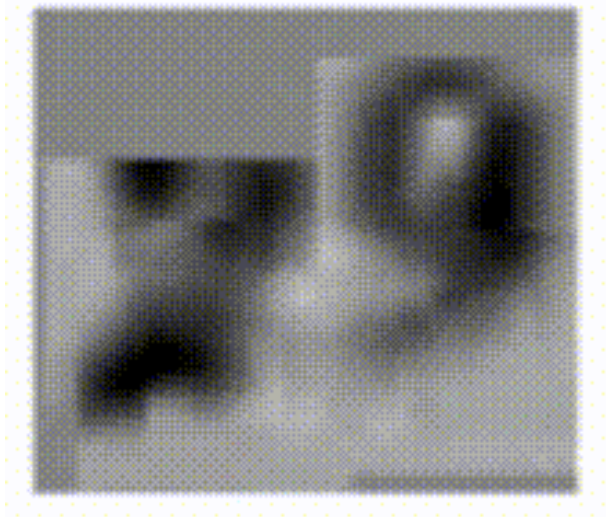
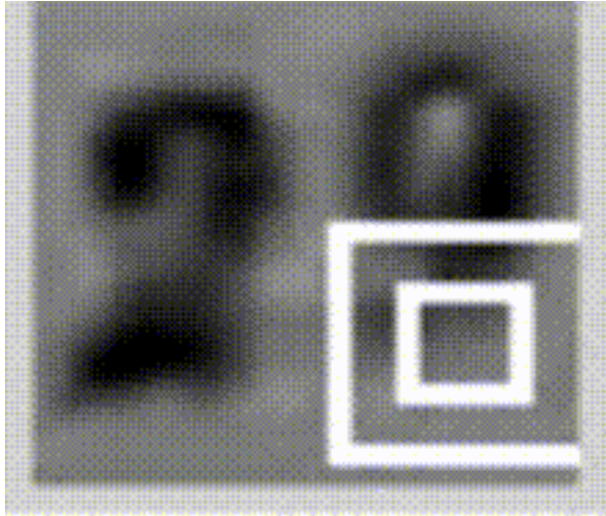
Ba, Mnih, and Kavukcuoglu, “Multiple Object Recognition with Visual Attention”, ICLR 2015.

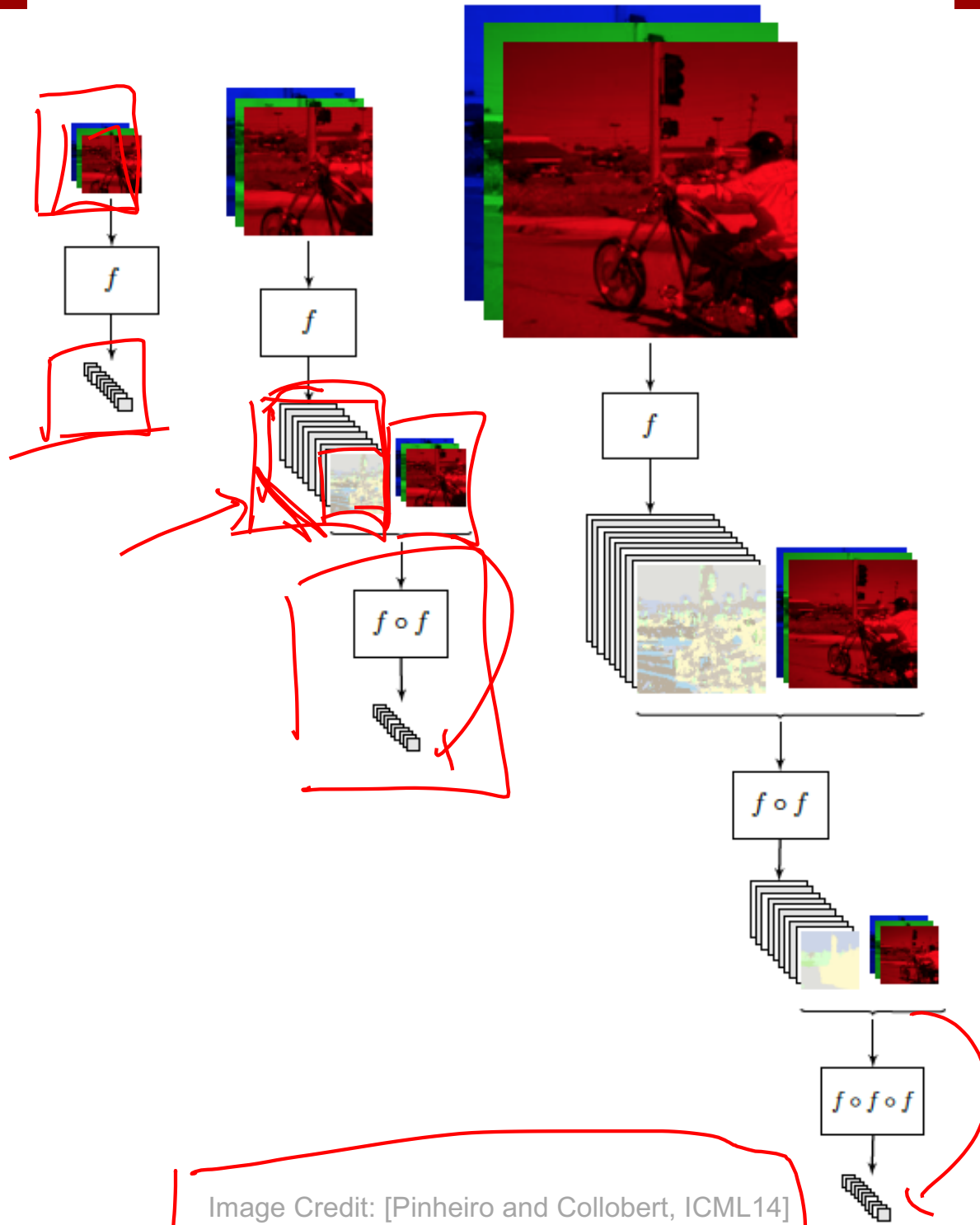
Gregor et al, “DRAW: A Recurrent Neural Network For Image Generation”, ICML 2015

Figure copyright Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra, 2015. Reproduced with permission.

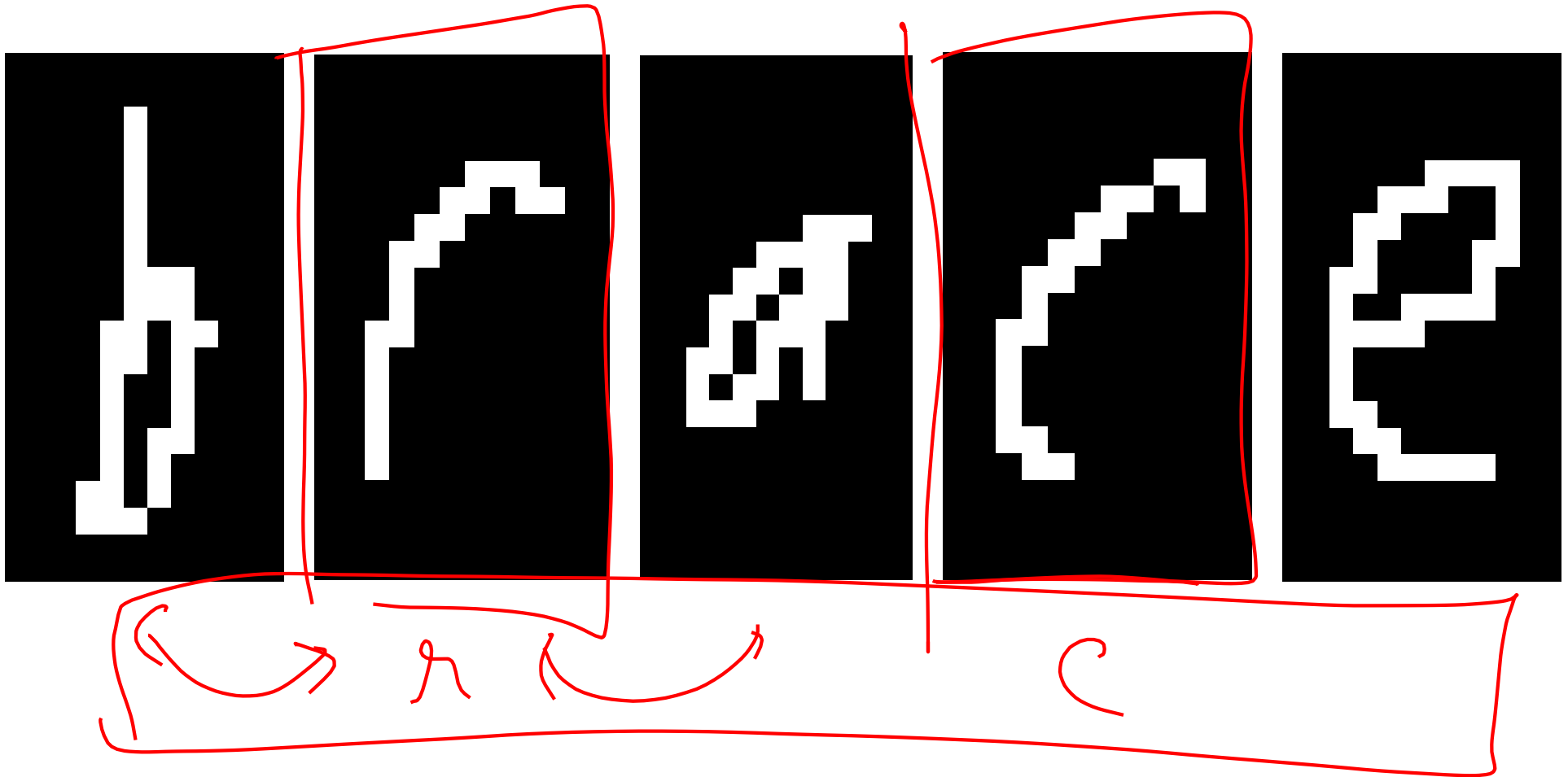
Even where you might not expect a sequence...

- Output ordering = sequence

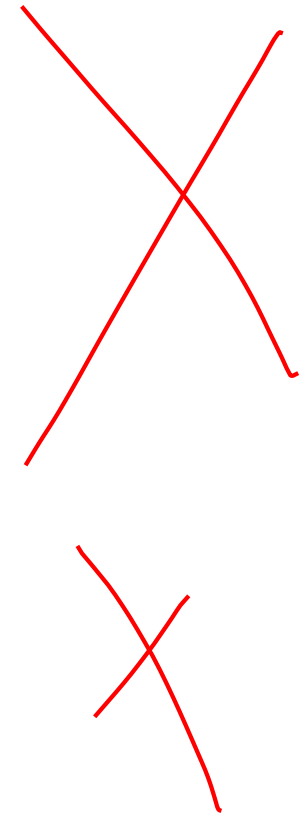
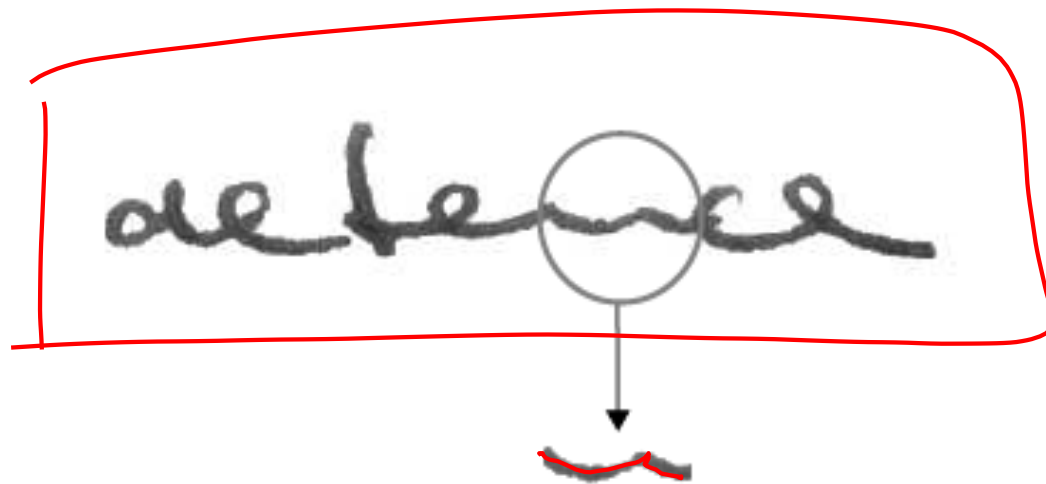




Why model sequences?



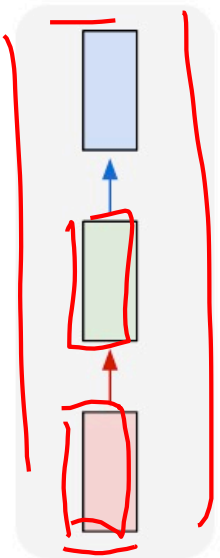
Why model sequences?



Sequences in Input or Output?

- It's a spectrum...

one to one

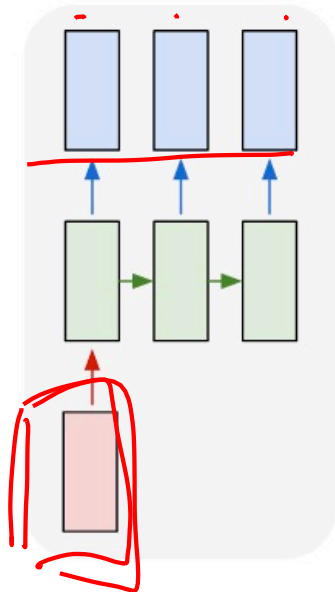


Input: No sequence

Output: No sequence

Example: "standard" classification / regression problems

one to many

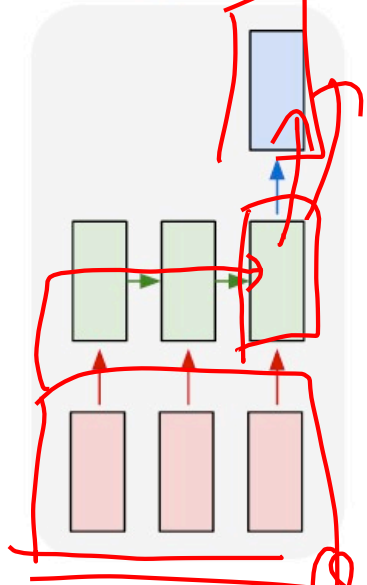


Input: No sequence

Output: Sequence

Example: Im2Caption

many to one

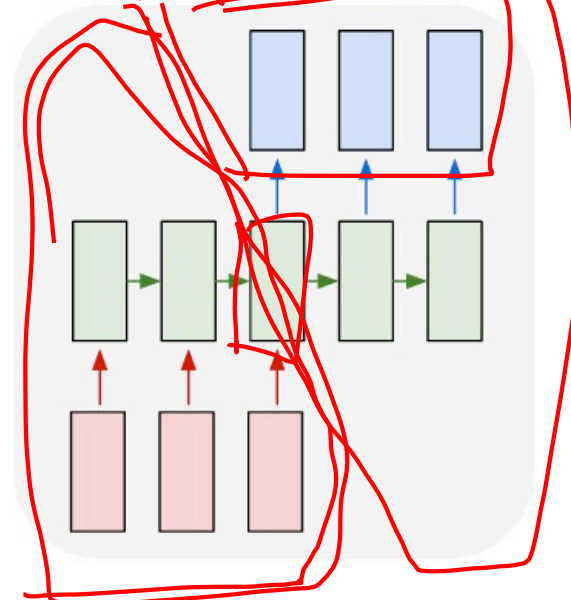


Input: Sequence

Output: No sequence

Example: sentence classification, multiple-choice question answering

many to many

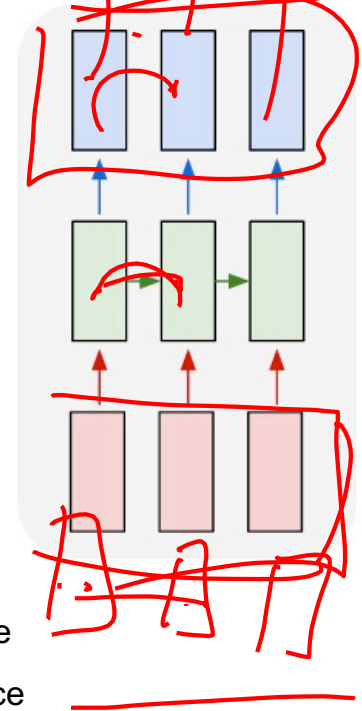


Input: Sequence

Output: Sequence

Example: machine translation, video captioning, open-ended question answering, video question answering

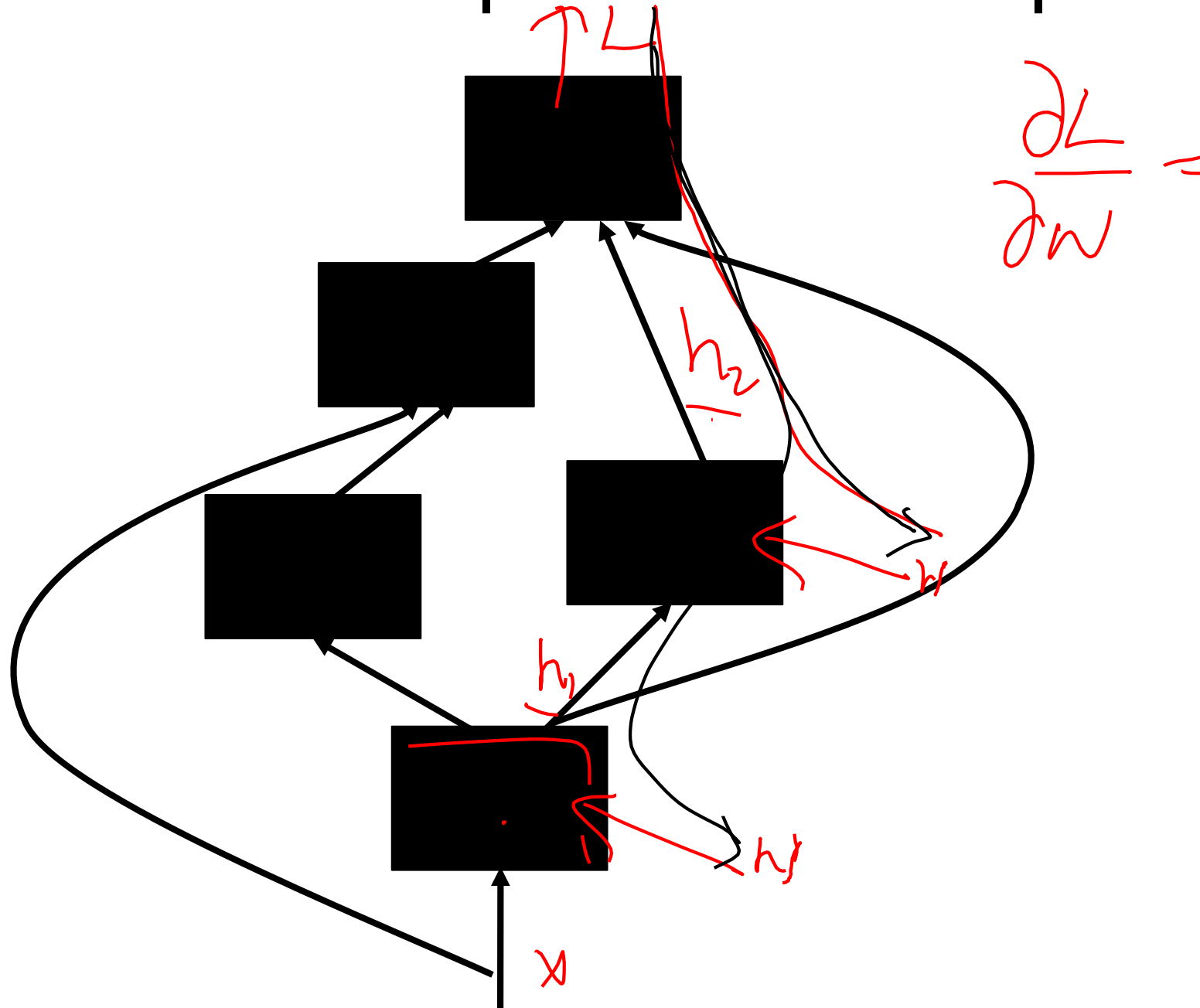
many to many



2 Key Ideas

- Parameter Sharing
 - in computation graphs = adding gradients

Computational Graph



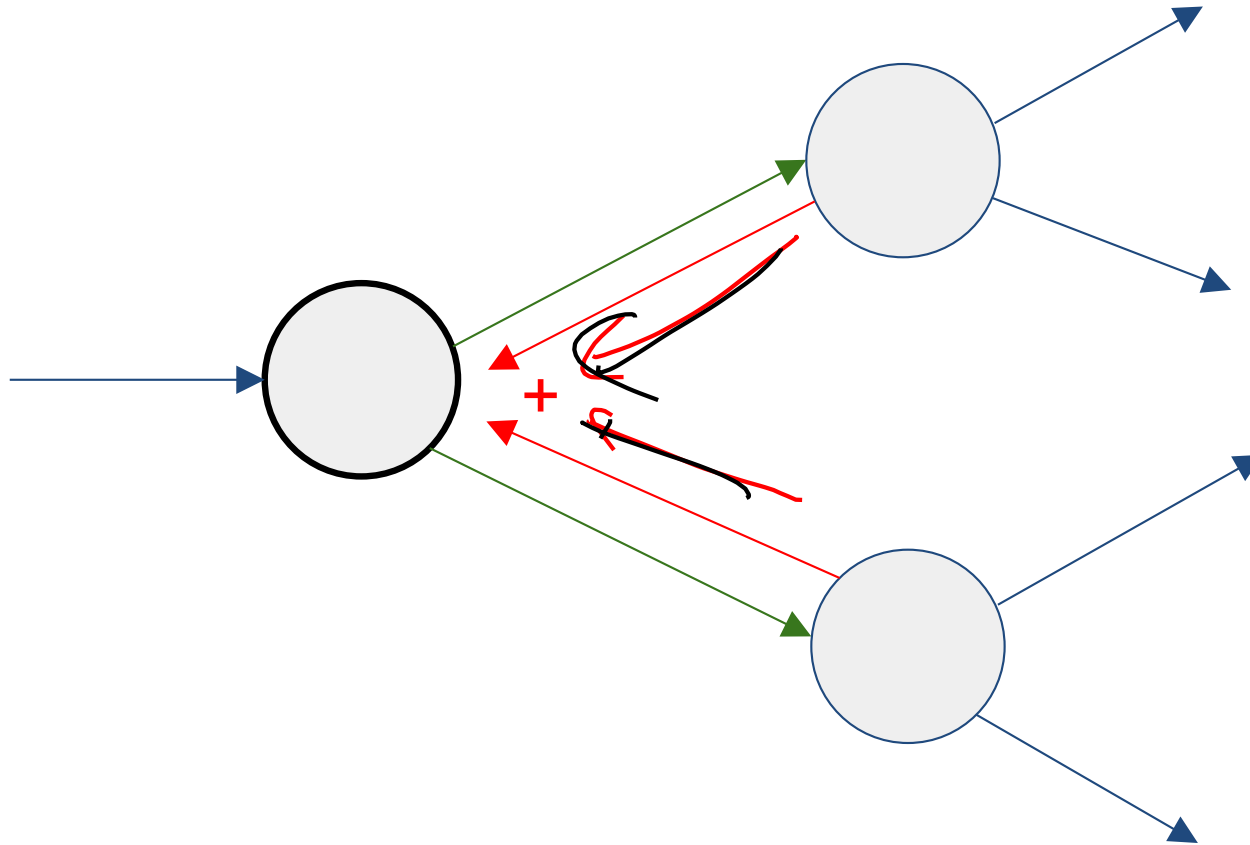
$$f(w_1^{(t)}, w_2^{(t)})$$

$$\frac{\partial f}{\partial w_1} = \lim_{h \rightarrow 0} \frac{f(w_1 + h, w_2)}{h}$$

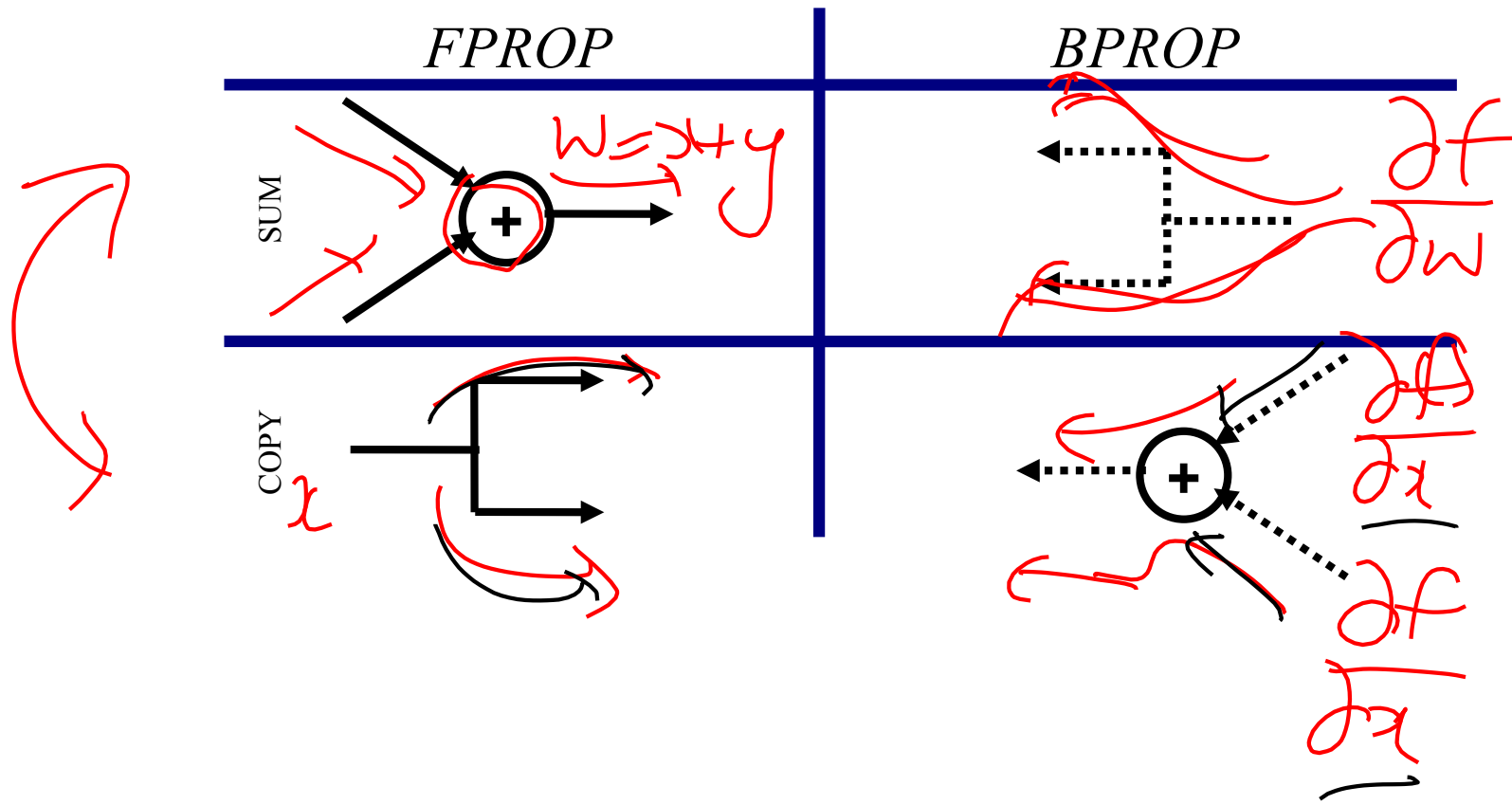
$$\frac{df}{dt} = \underbrace{\frac{\partial f}{\partial w_1}}_{\text{bracket}} \frac{\partial w_1}{\partial t} + \underbrace{\frac{\partial f}{\partial w_2}}_{\text{bracket}} \frac{\partial w_2}{\partial t}$$

$$\cancel{\frac{\partial f}{\partial w}} \quad \frac{df}{dw}$$

Gradients add at branches



Duality in Fprop and Bprop

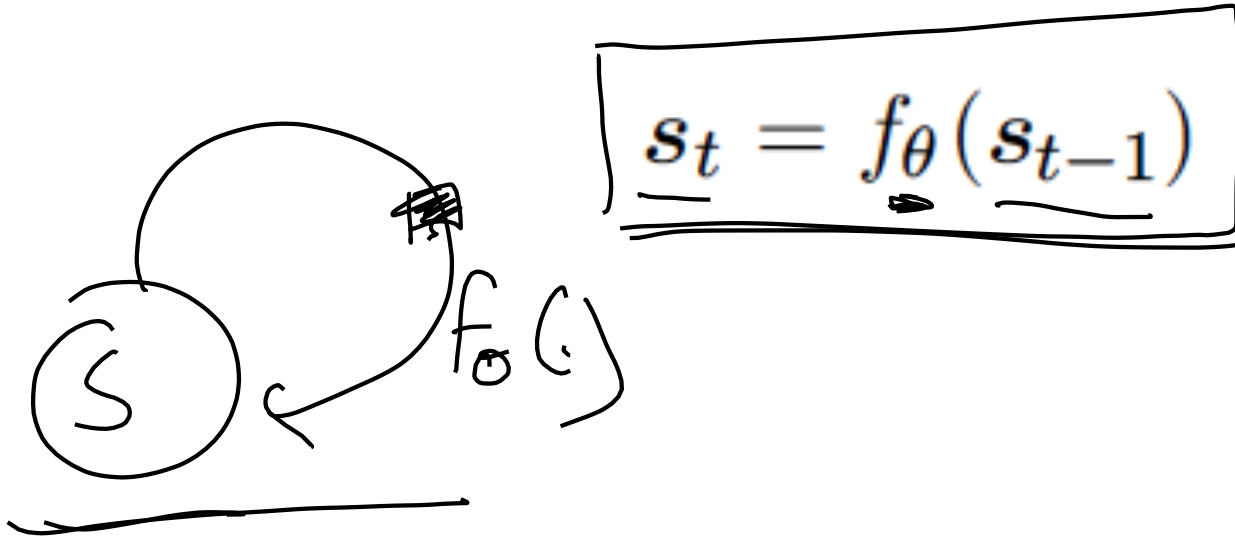


2 Key Ideas

- Parameter Sharing
 - in computation graphs = adding gradients
- “Unrolling”
 - in computation graphs with parameter sharing

How do we model sequences?

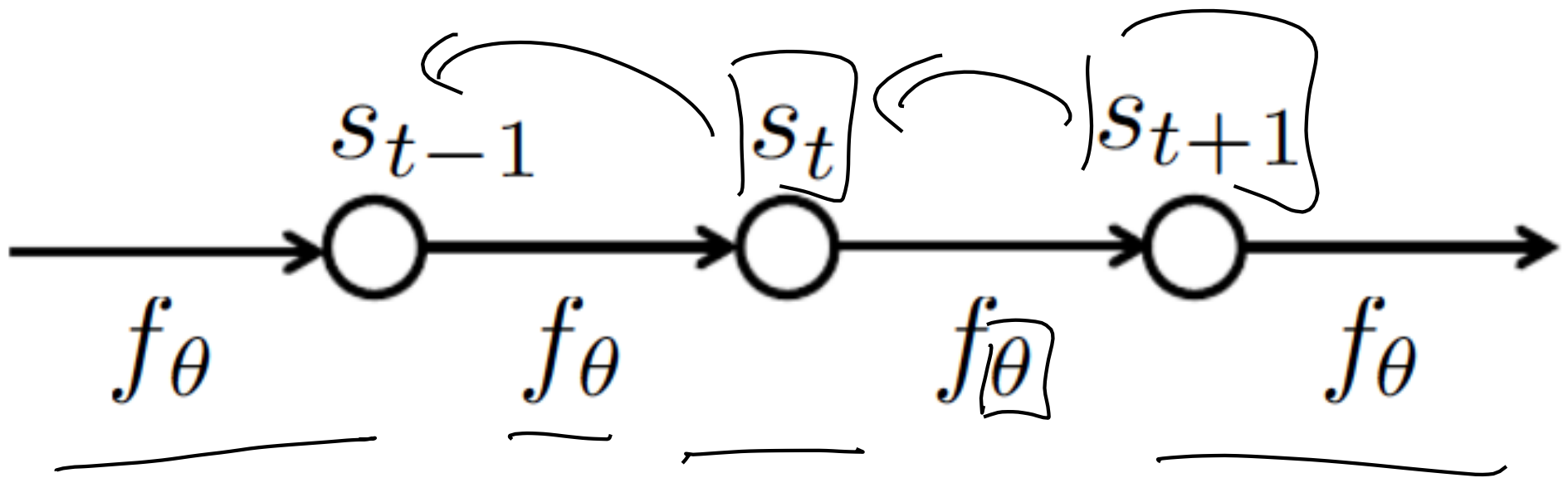
- No input



How do we model sequences?

- No input

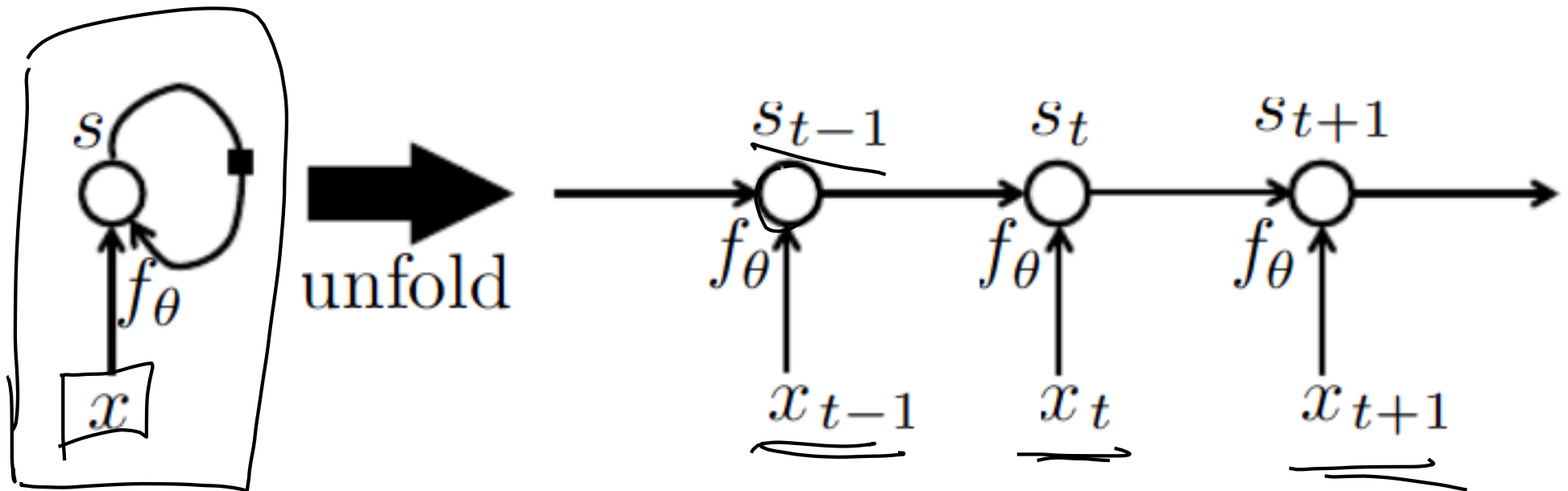
$$s_t = f_{\theta}(s_{t-1})$$



How do we model sequences?

- With inputs

$$\underline{s_t} = \underline{f_\theta(s_{t-1}, \underline{x_t})}$$



2 Key Ideas

- Parameter Sharing

- in computation graphs = adding gradients

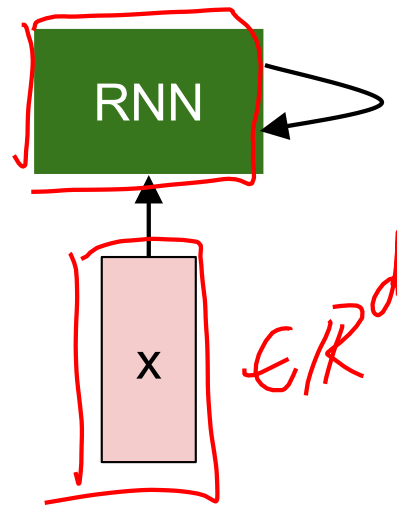
- “Unrolling”

- in computation graphs with parameter sharing

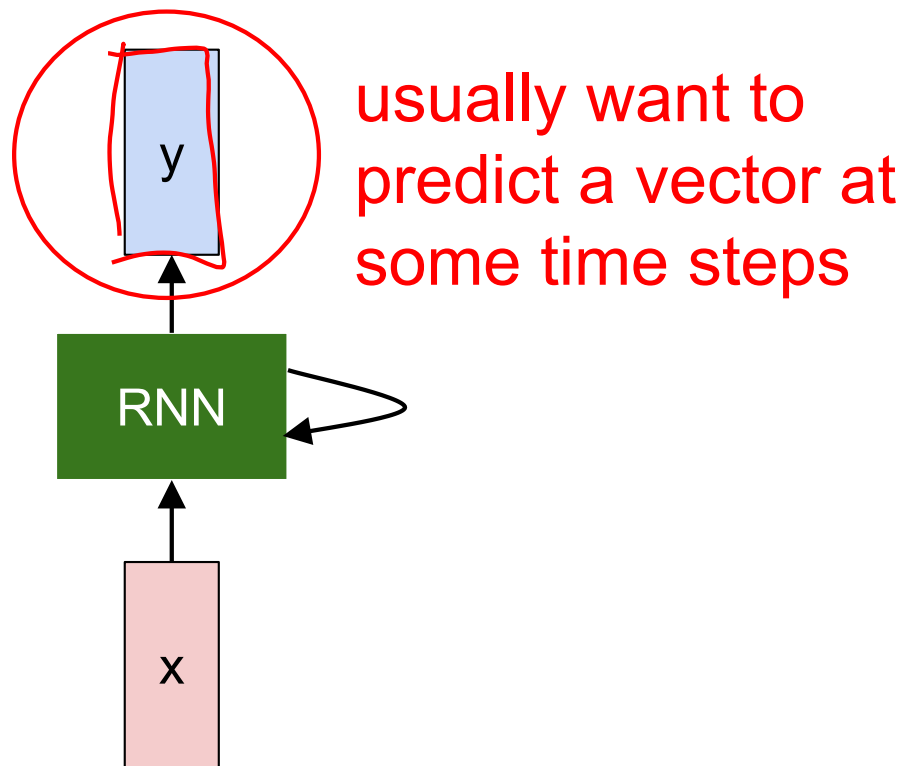
- Parameter sharing + Unrolling

- Allows modeling arbitrary sequence lengths!
- Keeps numbers of parameters in check

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:

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

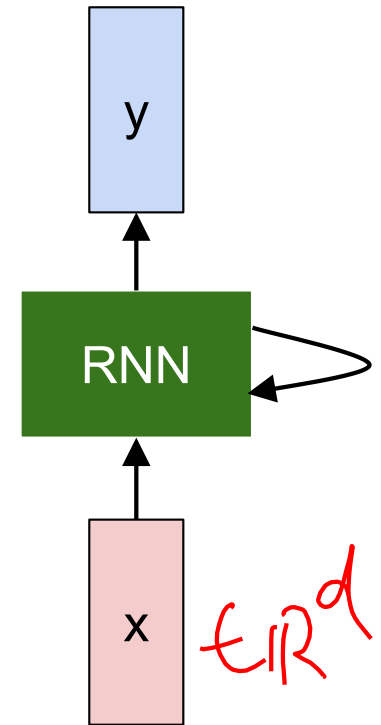
new state

some function with parameters W

old state

input vector at some time step

h

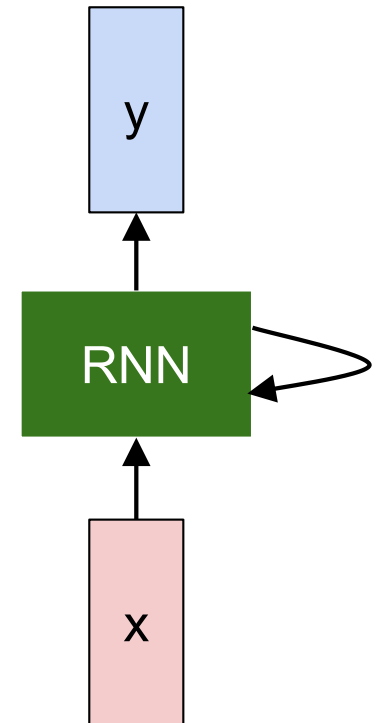


Recurrent Neural Network

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

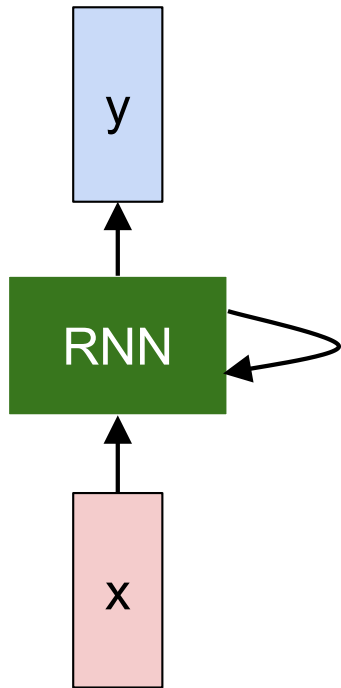
$$h_t = \underline{f_W(h_{t-1}, x_t)}$$

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



(Vanilla) Recurrent Neural Network

The state consists of a single "hidden" vector h :

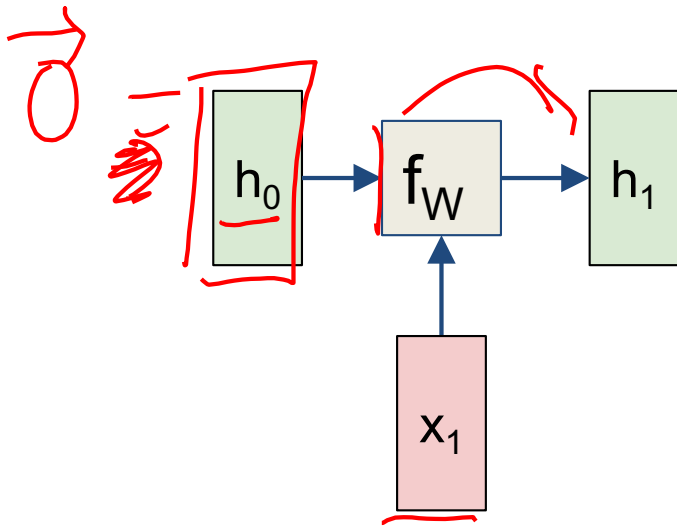


$$y_t = W_{hy}h_t + b_y$$

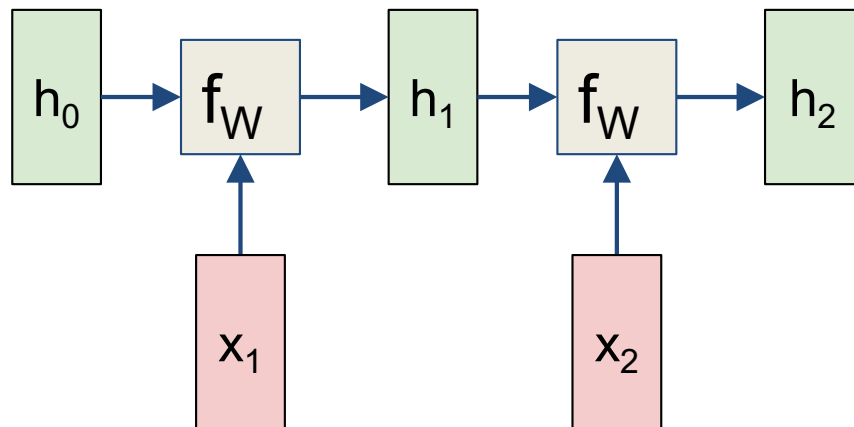
$d_2 \times (d_1 + d_2)$ \tanh

$$h_t = f_W(h_{t-1}, x_t)$$
$$\vec{h}_t = \tanh(W_{hh}\vec{h}_{t-1} + W_{xh}\vec{x}_t + \vec{b}_h)$$

RNN: Computational Graph

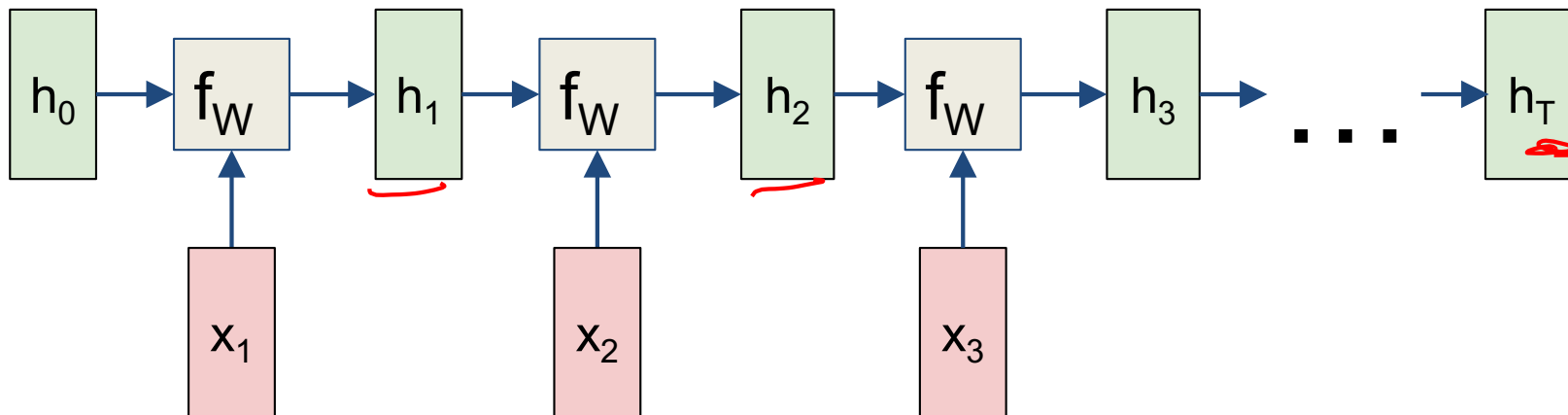


RNN: Computational Graph



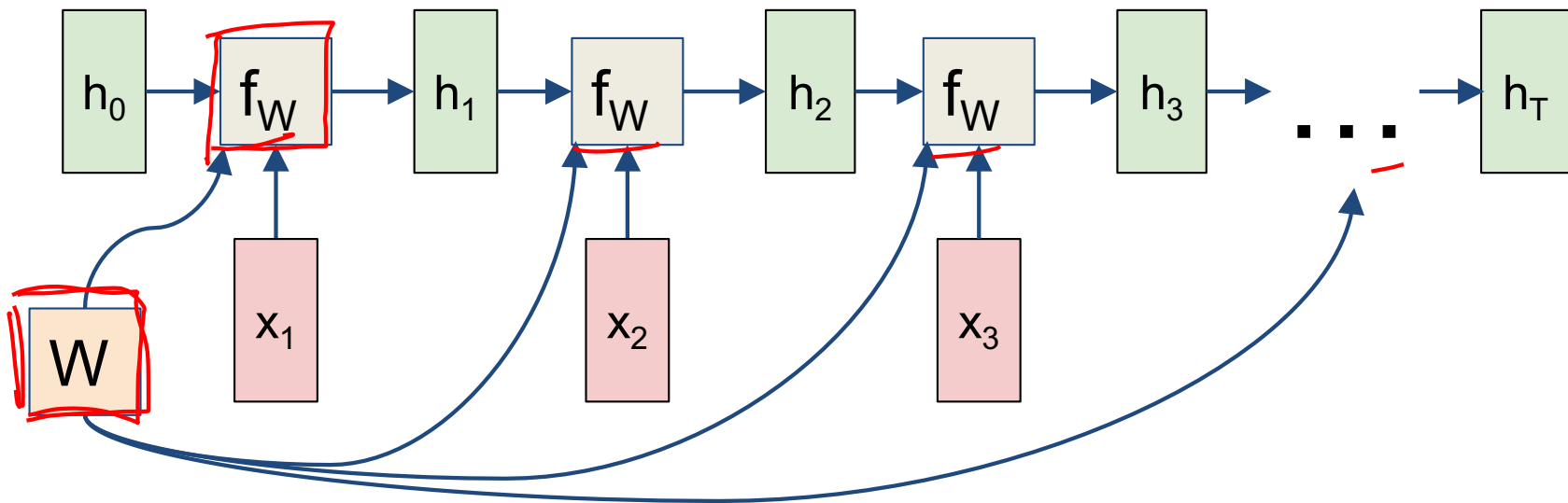
RNN: Computational Graph

$$h_2 = f_W(h_1, x_2)$$

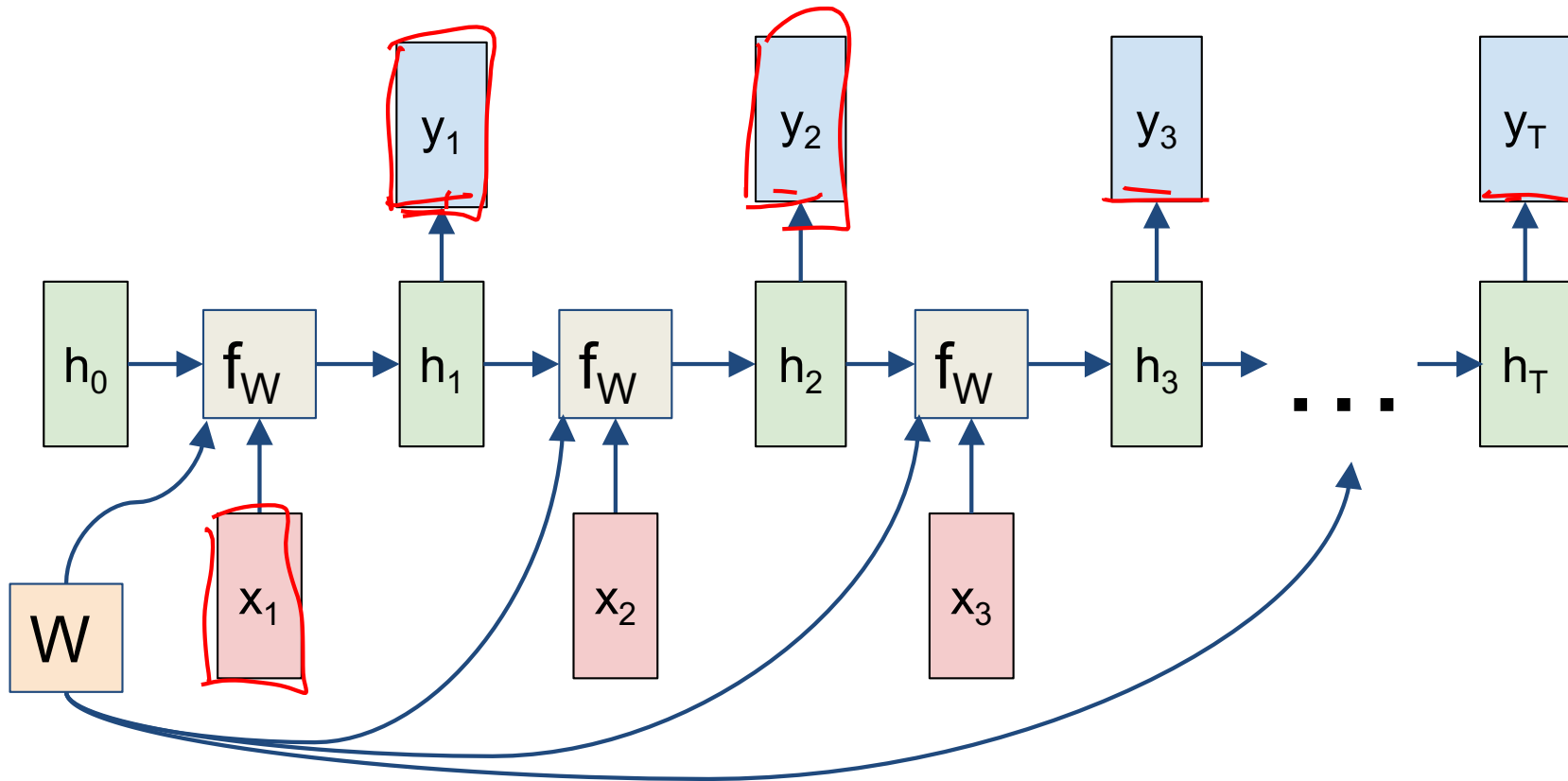


RNN: Computational Graph

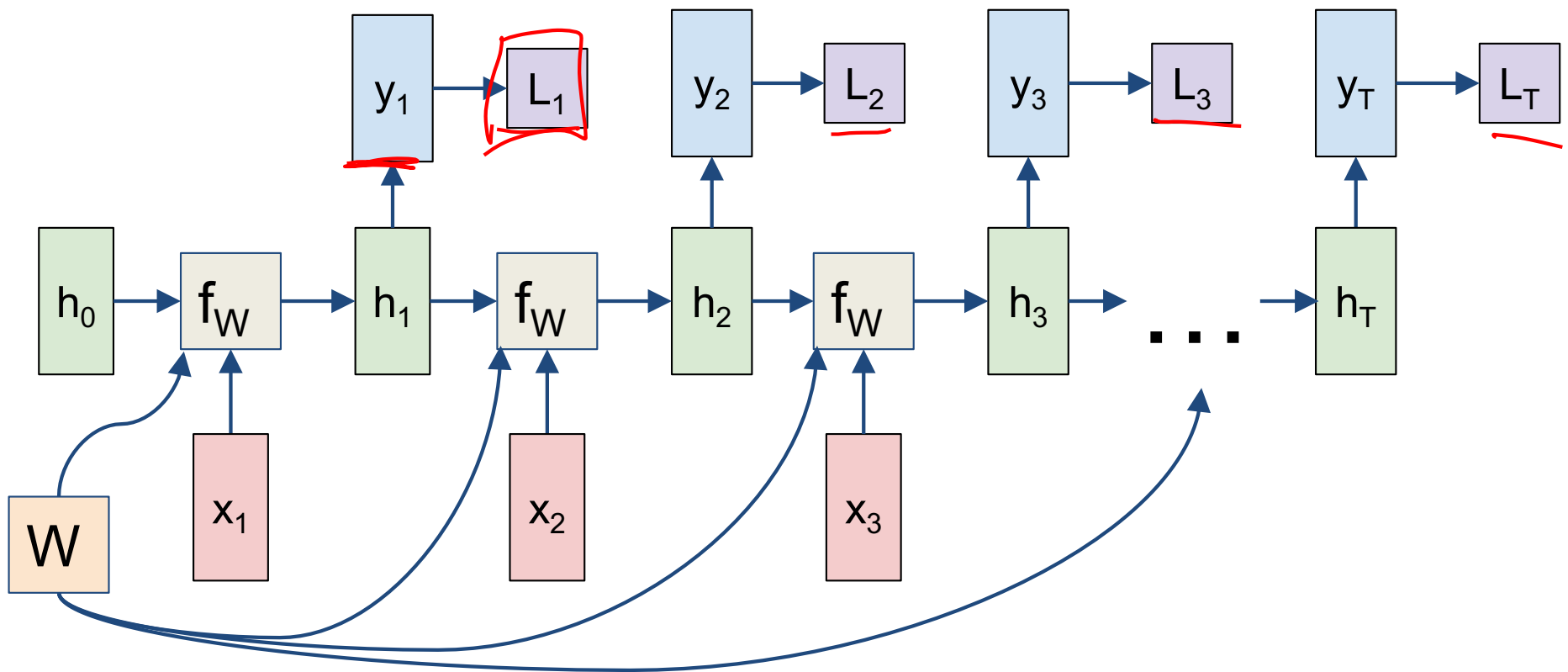
Re-use the same weight matrix at every time-step



RNN: Computational Graph: Many to Many

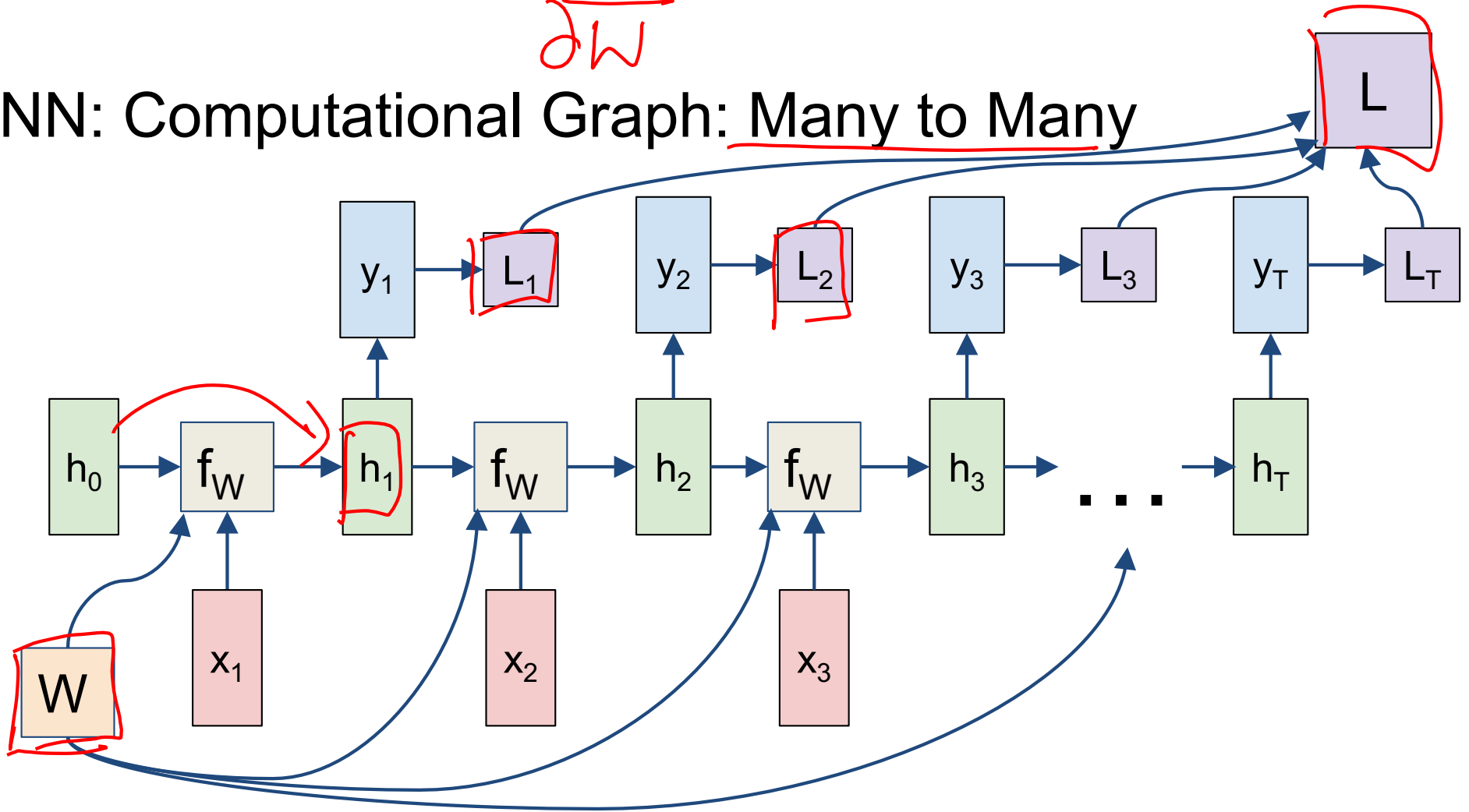


RNN: Computational Graph: Many to Many

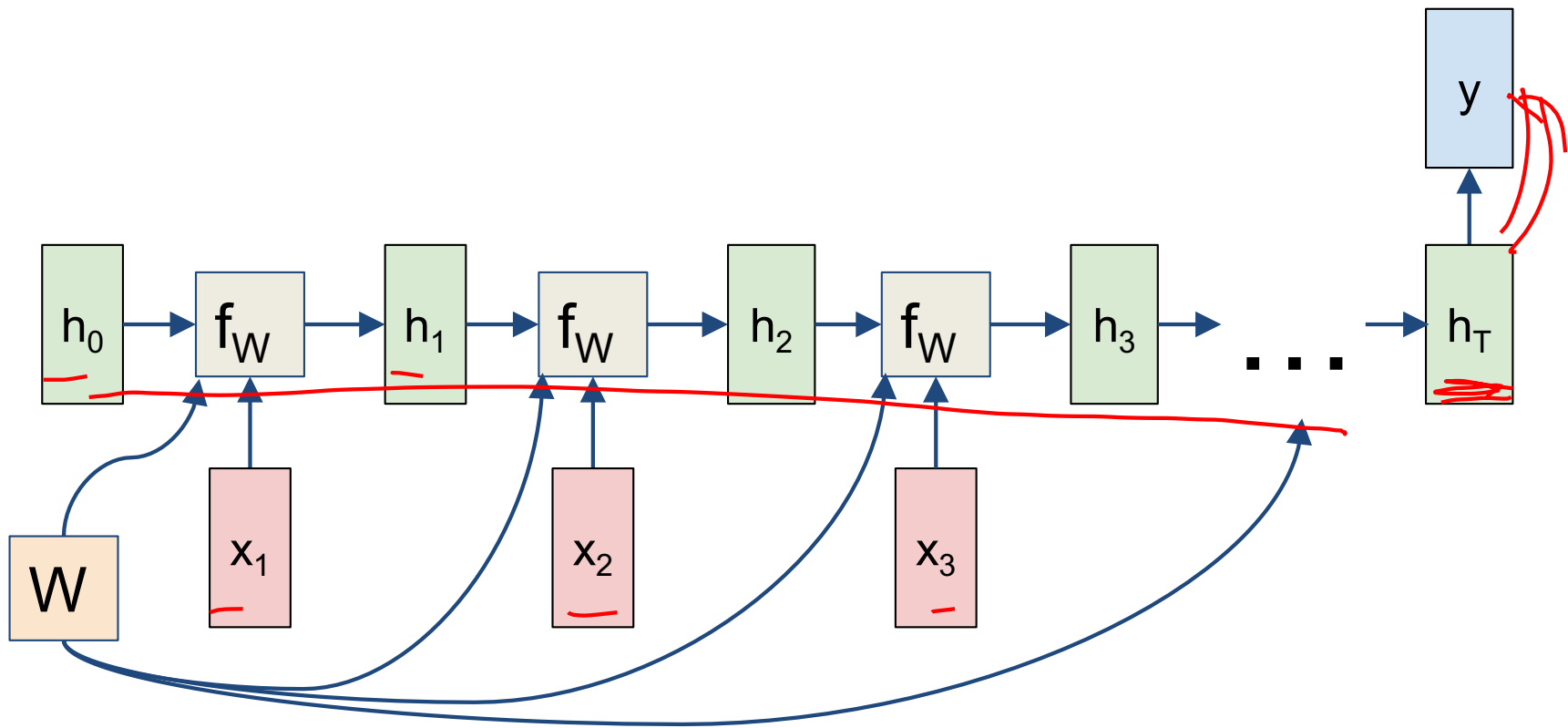


$$\frac{\partial L}{\partial W}$$

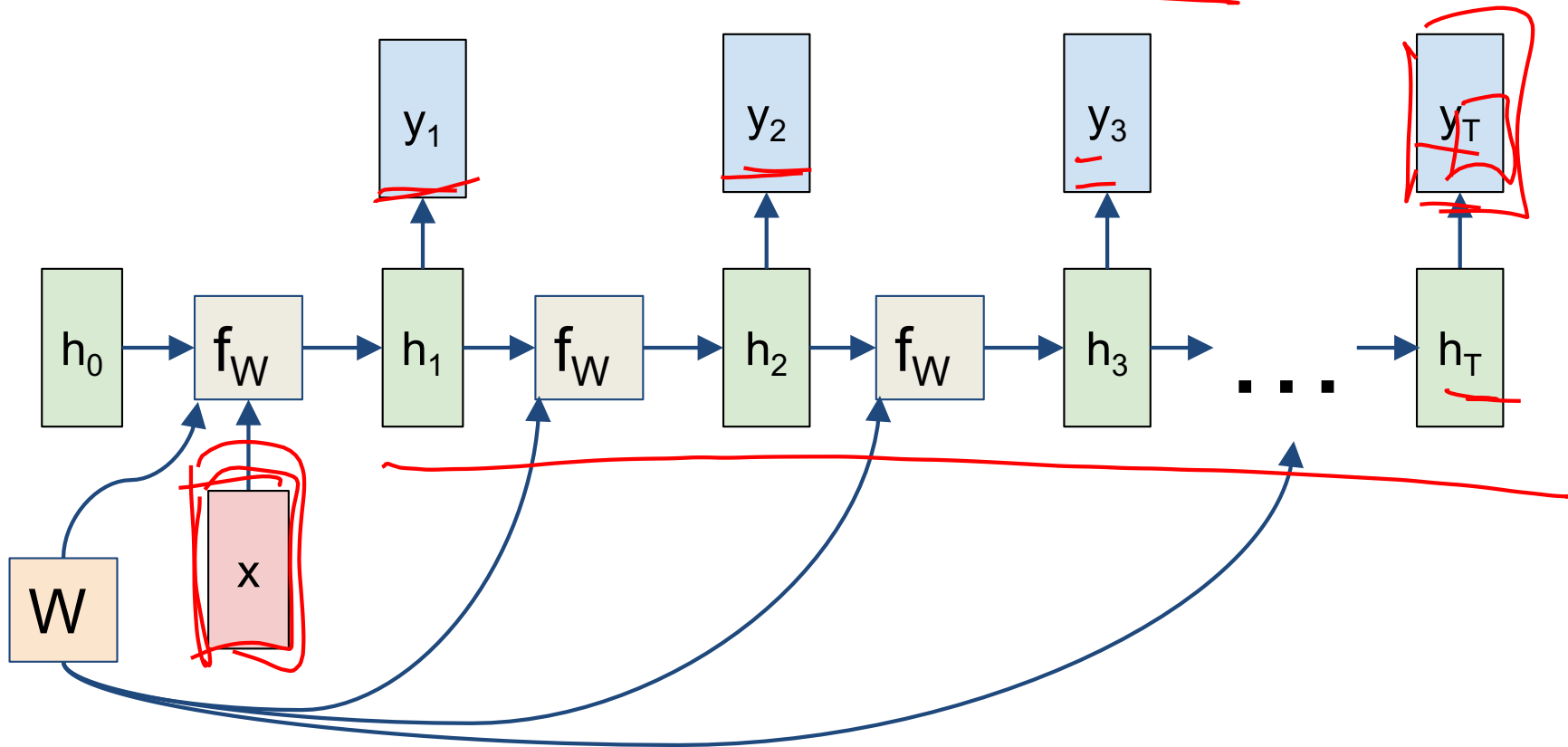
RNN: Computational Graph: Many to Many



RNN: Computational Graph: Many to One

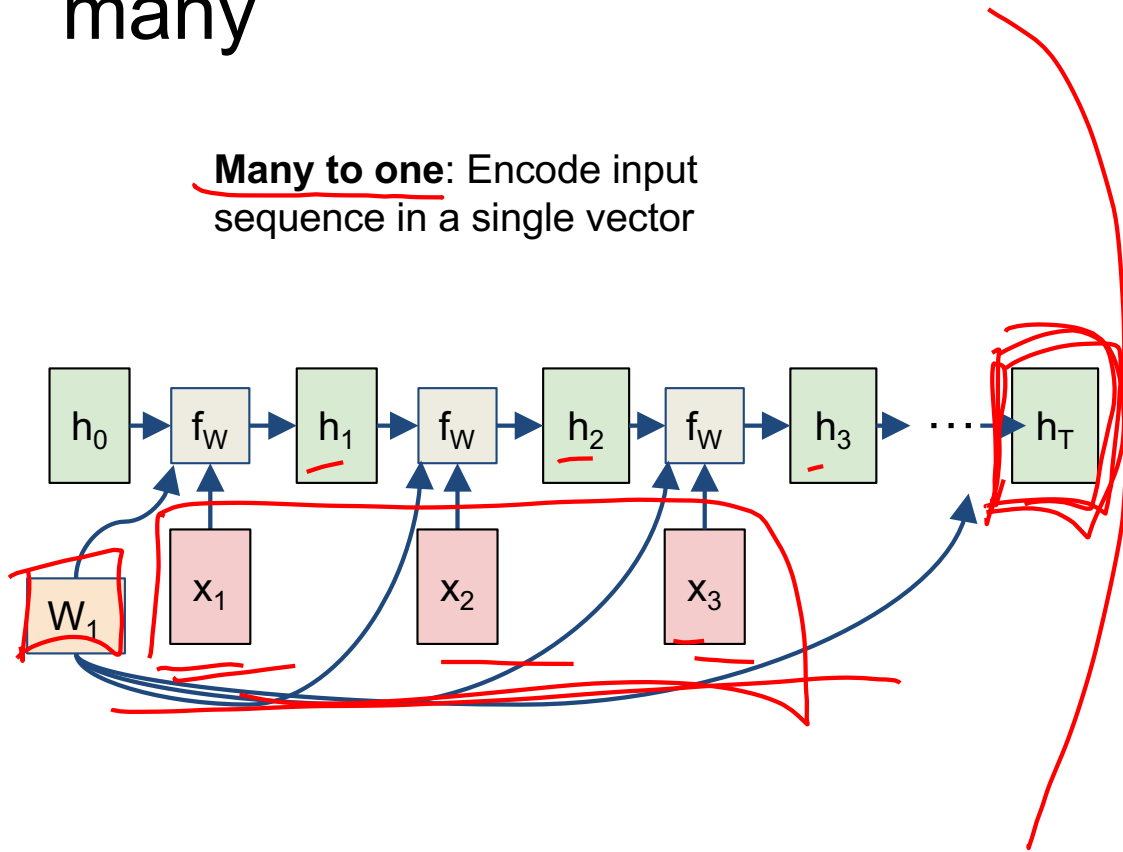


RNN: Computational Graph: One to Many



Sequence to Sequence: Many-to-one + one-to-many

Many to one: Encode input sequence in a single vector

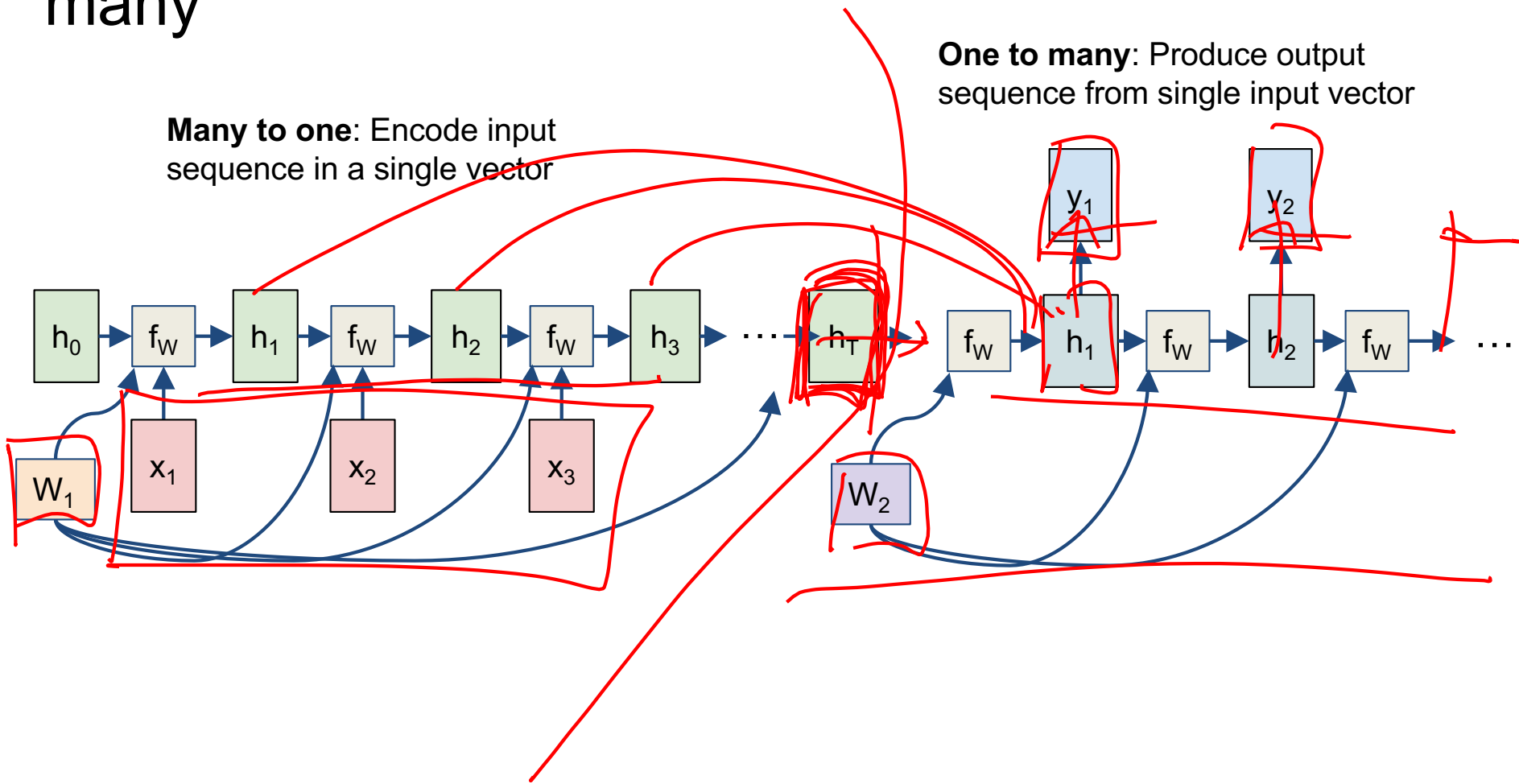


Sequence to Sequence: Many-to-one + one-to-many

$\min L(W_1, W_2)$

Many to one: Encode input sequence in a single vector

One to many: Produce output sequence from single input vector

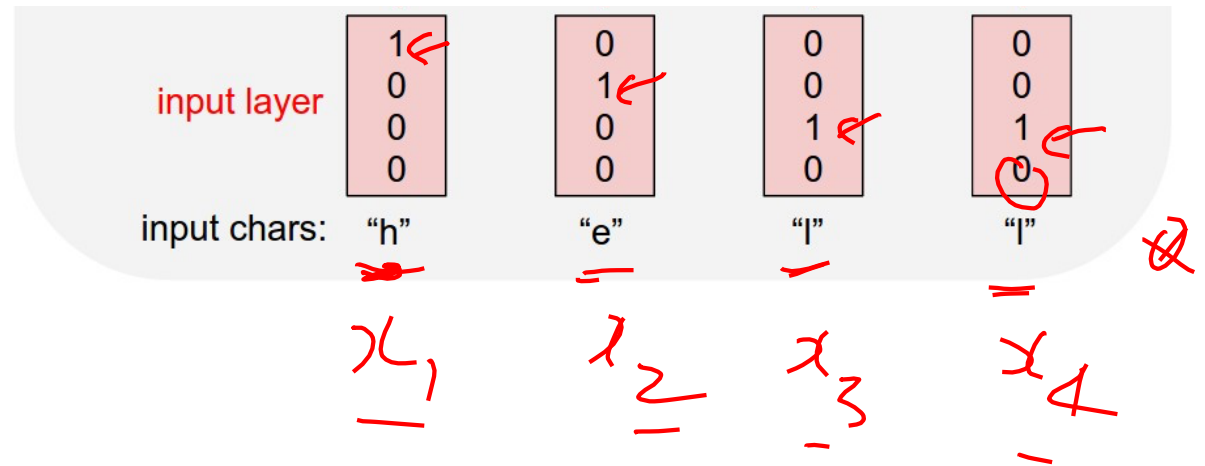


$$P(x_1, x_2, x_3, x_4)$$

Example:
Character-level
Language Model

Vocabulary:
[h,e,l,o]

Example training
sequence:
"hello"

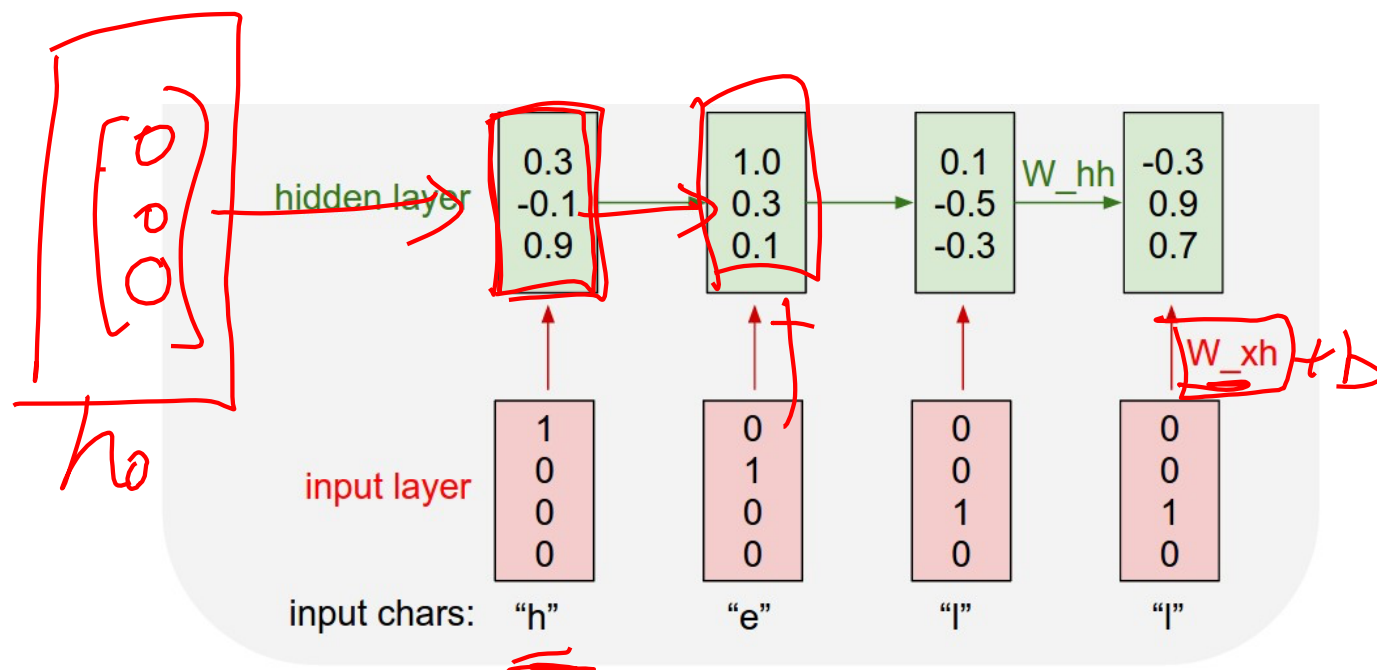


Example: Character-level Language Model

Vocabulary:
[h,e,l,o]

Example training
sequence:
"hello"

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

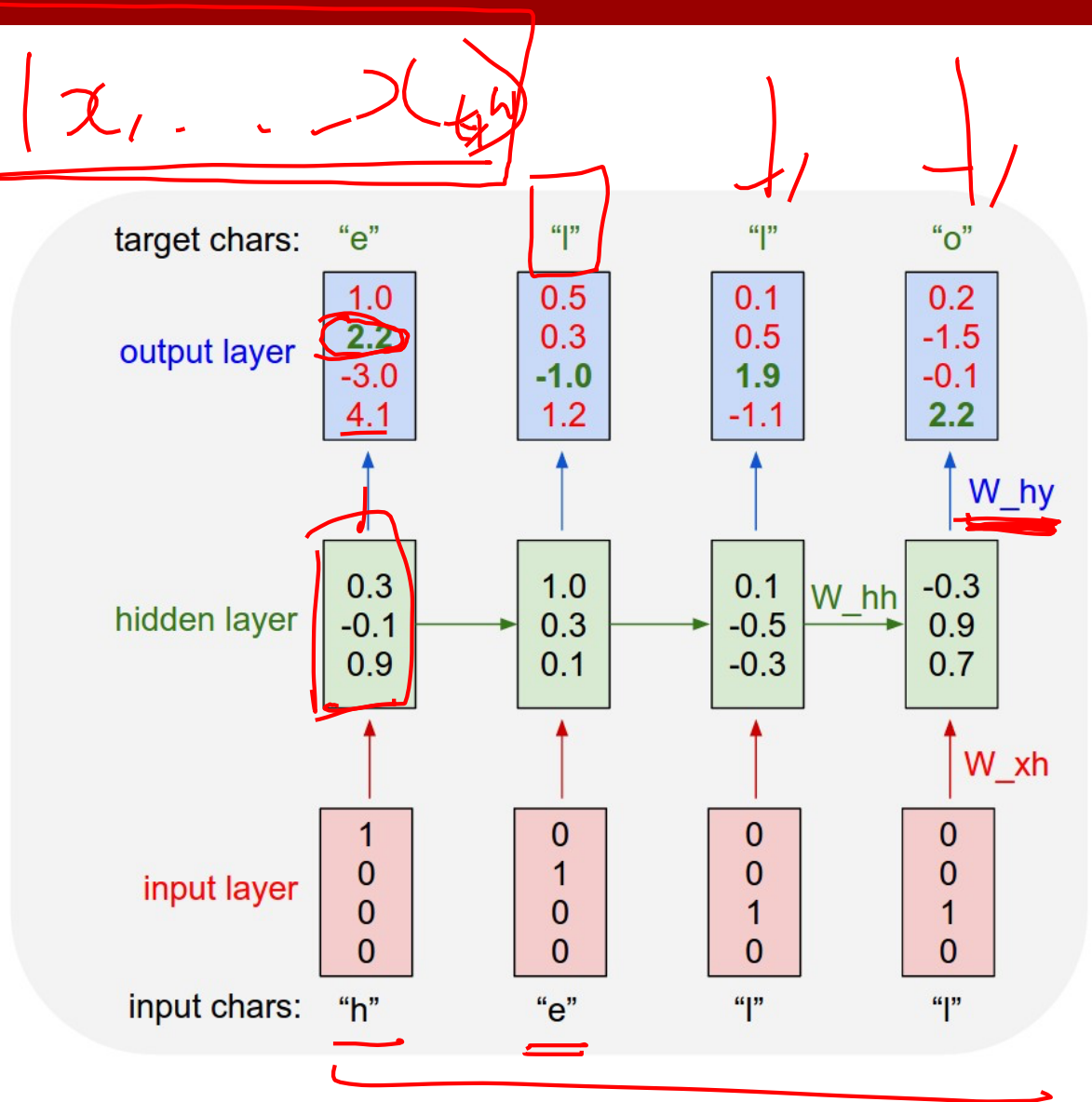


$$\arg \max_w \log P(x_t | x_1, \dots, x_{t-1})$$

Example: Character-level Language Model

Vocabulary:
[h,e,l,o]

Example training
sequence:
"hello"

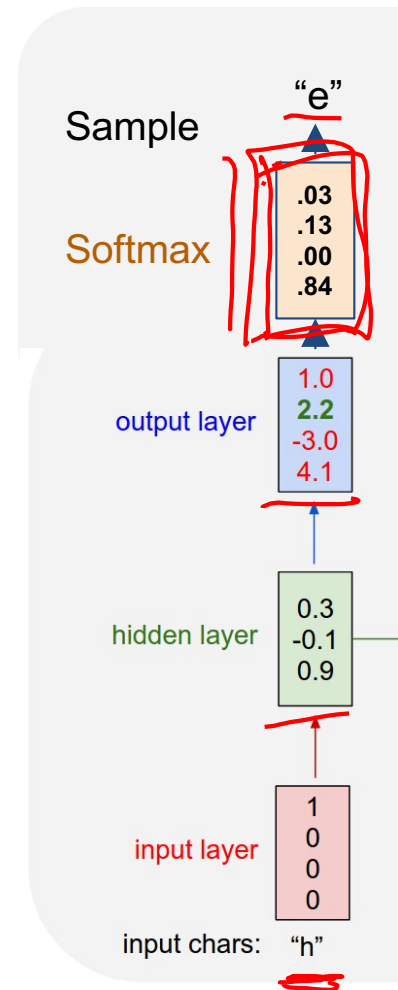


$$\underline{x}_2 \sim P(\underline{x}_2 \mid x_1 = \text{h}, w)$$

Example: Character-level Language Model Sampling

Vocabulary:
[h,e,l,o]

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

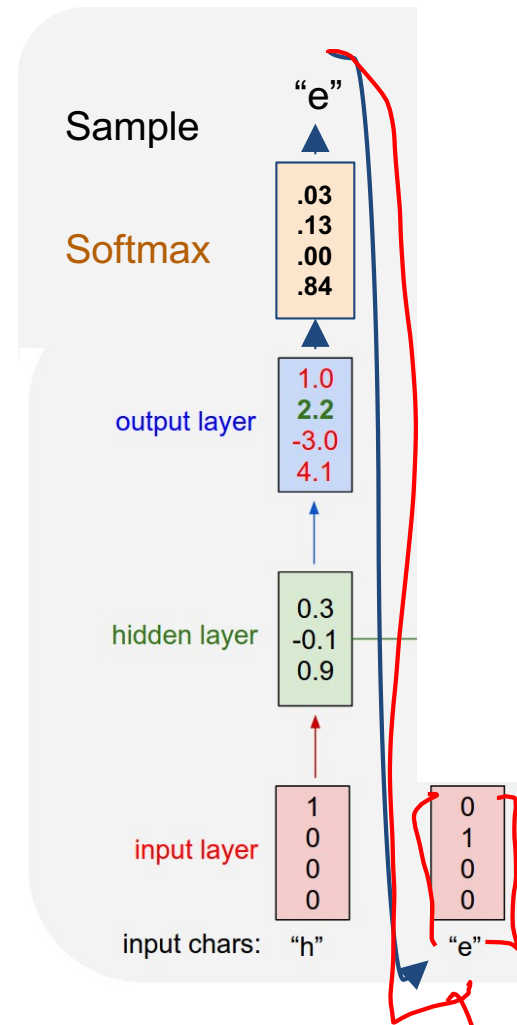


x_1 x_2

Example: Character-level Language Model Sampling

Vocabulary:
[h,e,l,o]

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

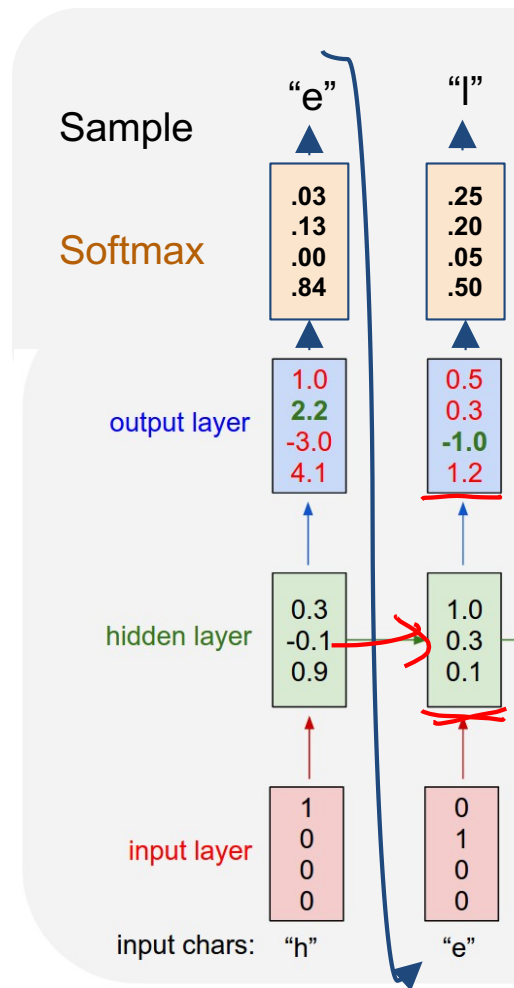


$$\underline{x}_3 \sim P(x_3 | x_1 = x_1, x_2 = x_2, w)$$

Example: Character-level Language Model Sampling

Vocabulary:
[h,e,l,o]

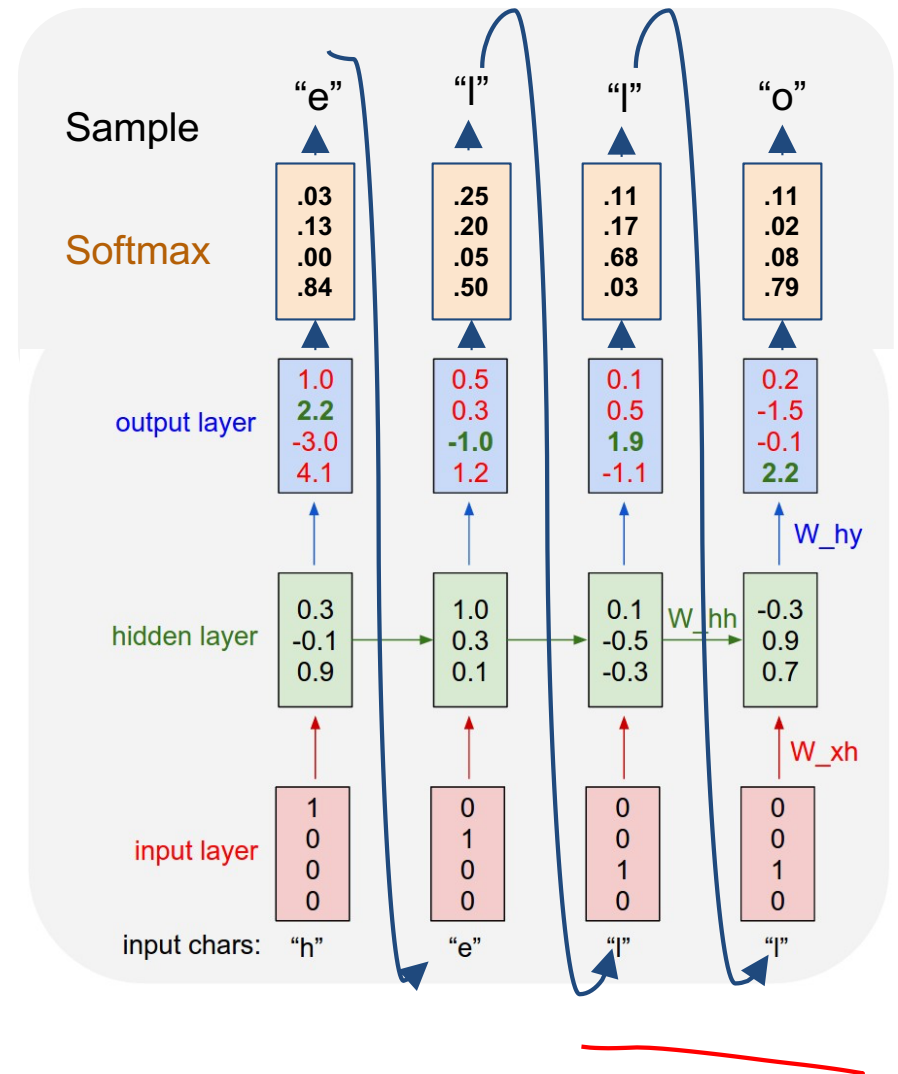
At test-time sample
characters one at a
time, feed back to
model



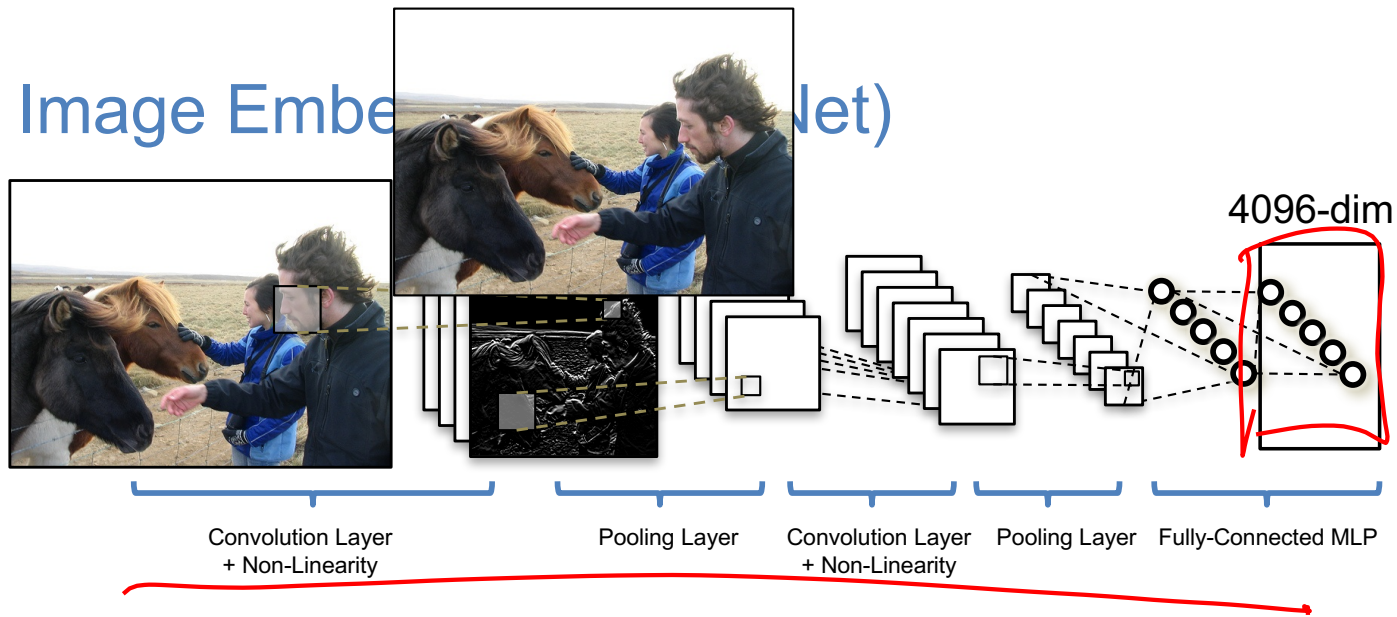
Example: Character-level Language Model Sampling

Vocabulary:
[h,e,l,o]

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

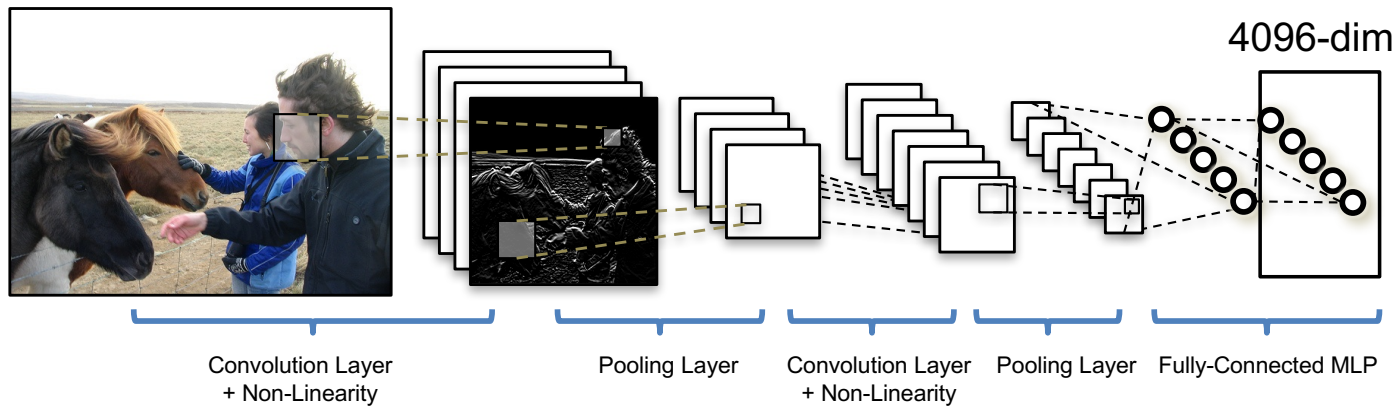


Neural Image Captioning

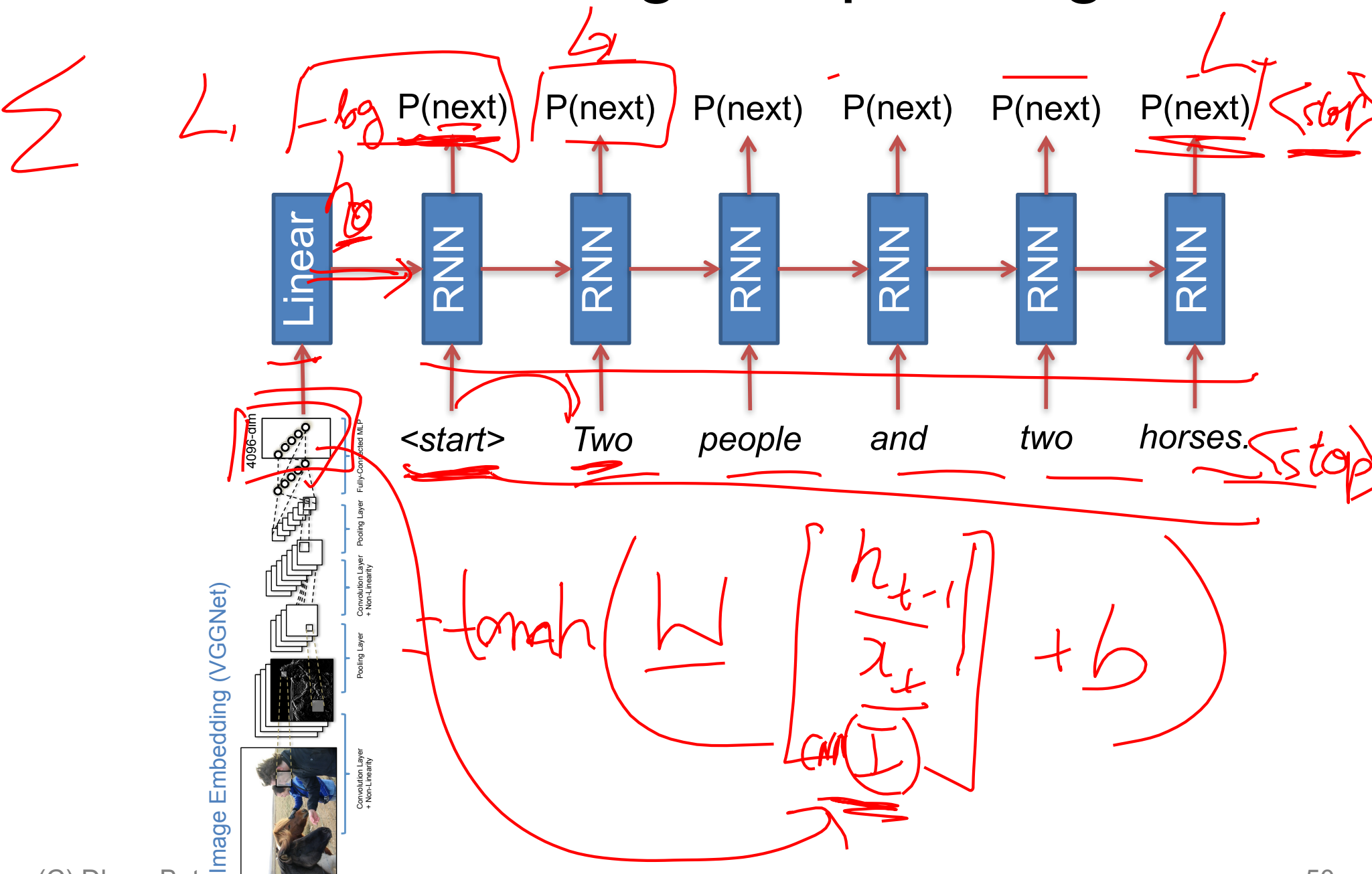


Neural Image Captioning

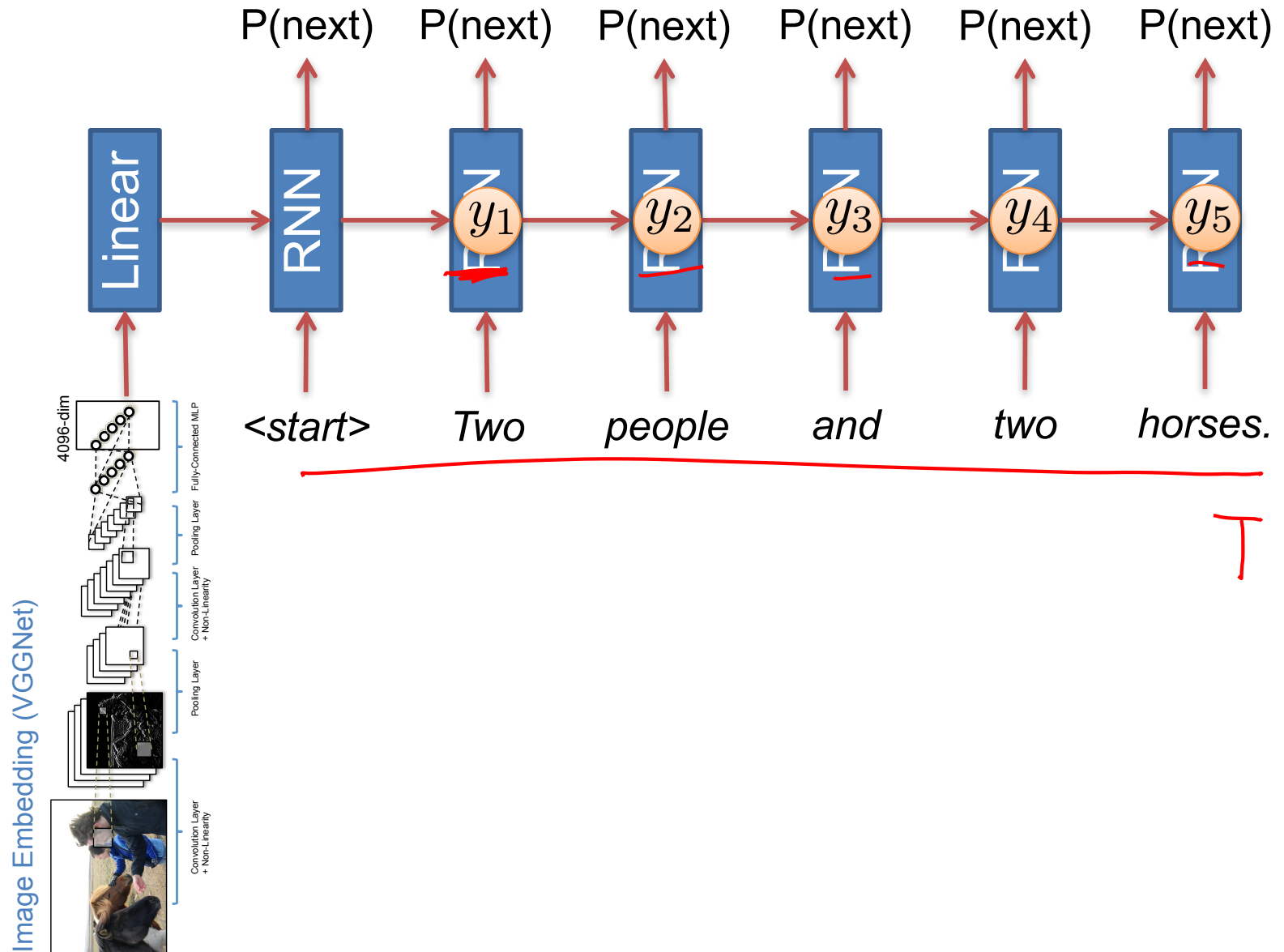
Image Embedding (VGGNet)



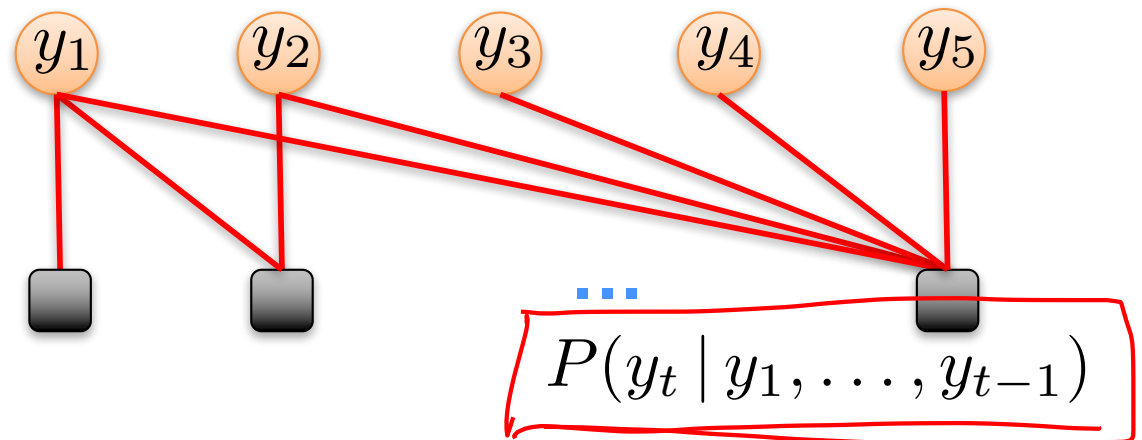
Neural Image Captioning



Neural Image Captioning



Sequence Model Factor Graph



MAP

$$\underset{y_1, \dots, y_T}{\operatorname{argmax}}$$

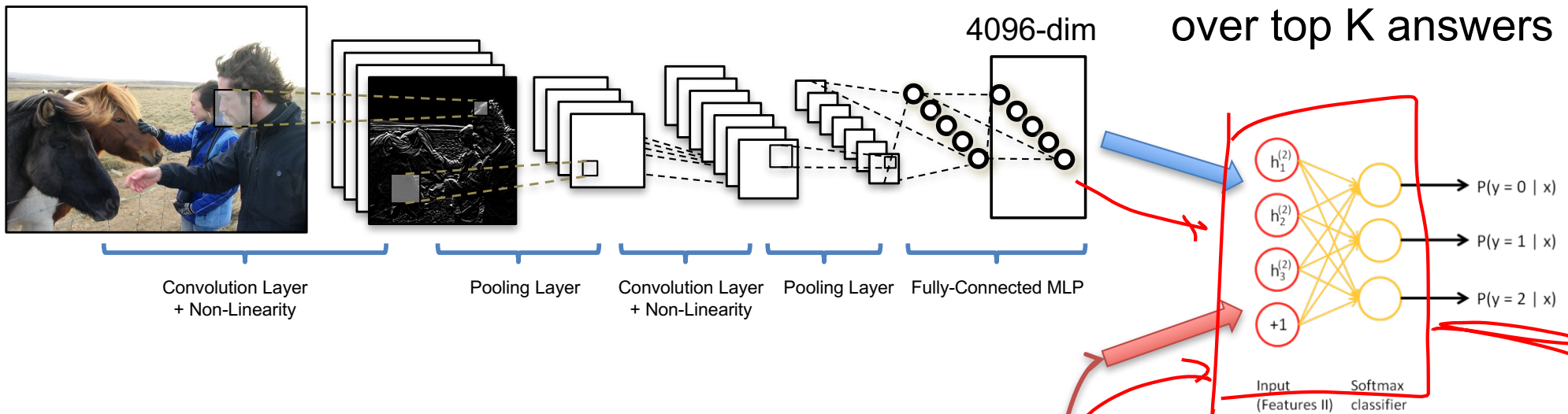
$$P(y_1, \dots, y_T | x, w)$$

Beam Search Demo

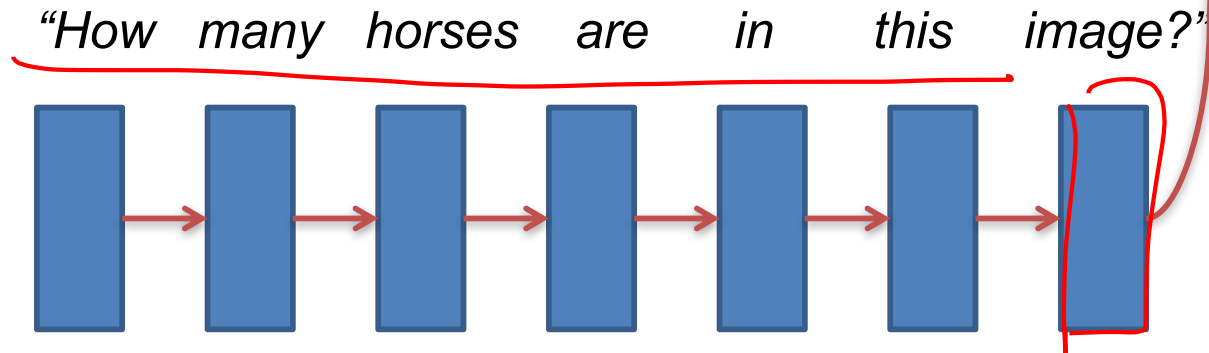
- <http://dbs.cloudcv.org/captioning&mode=interactive>

Typical VQA Models

Image Embedding (VGGNet)

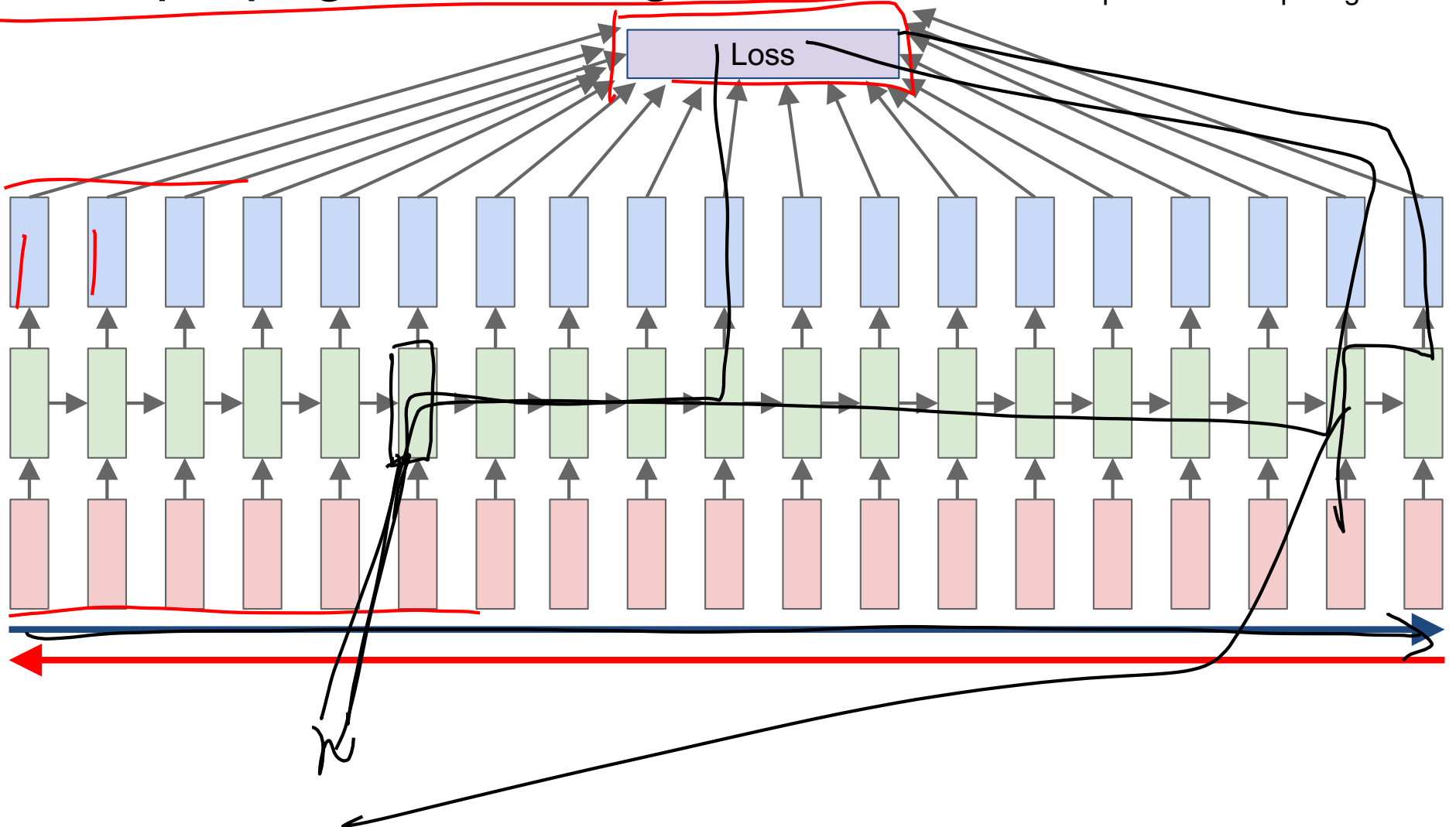


Question Embedding (LSTM)

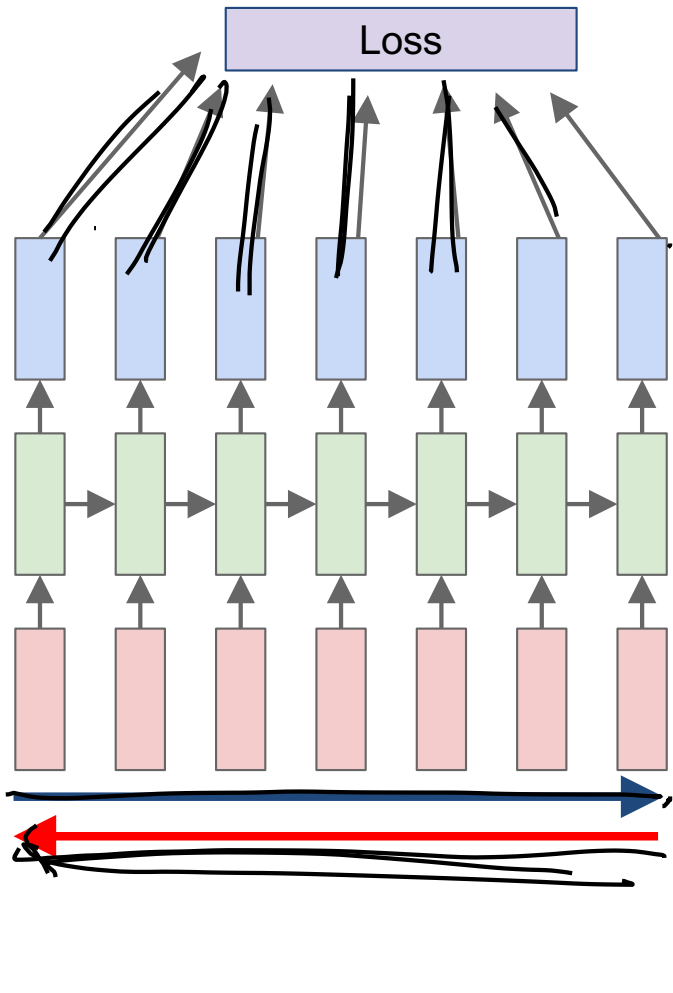


Backpropagation through time

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

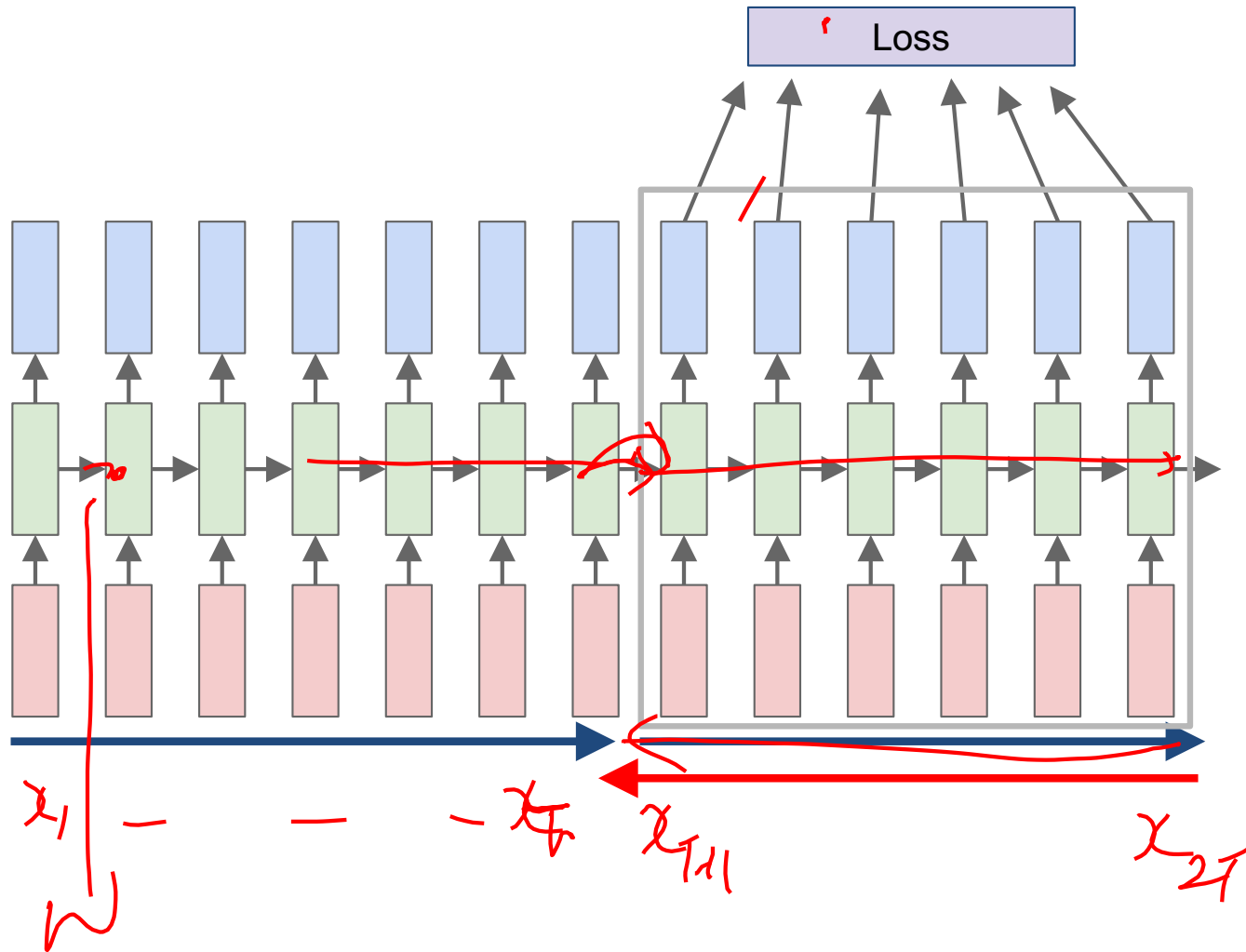


Truncated Backpropagation through time



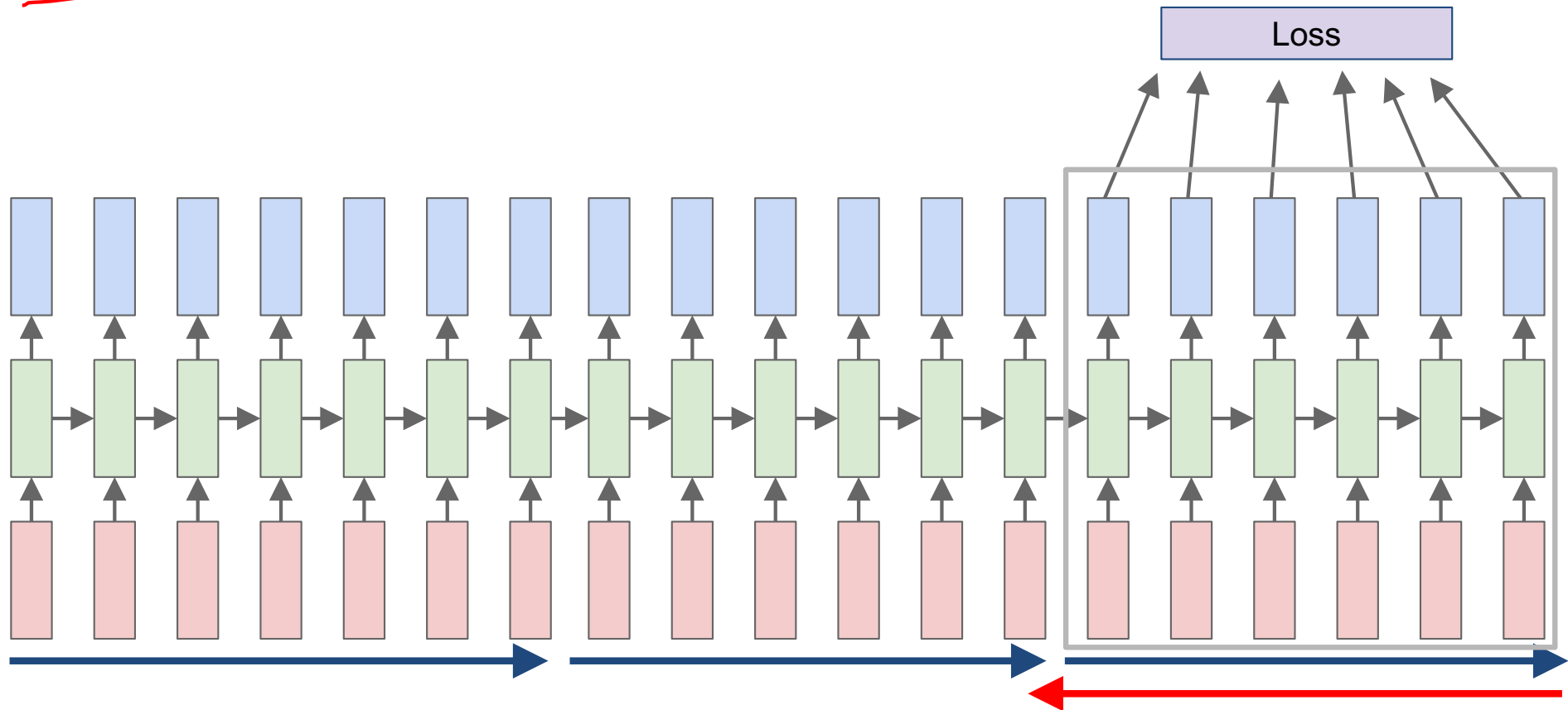
Run forward and backward through chunks of the sequence instead of whole sequence

Truncated Backpropagation through time



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

Truncated Backpropagation through time



[min-char-rnn.py gist: 112 lines of Python](https://gist.github.com/karpathy/d4dee566867f8291f086)

```
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 Nx1 array of initial hidden state
31     returns the loss, gradients on model parameters, and last hidden state
32     """
33     xs, hs, ys, ps = {}, {}, {}, {}
34     hs[-1] = np.copy(hprev)
35     loss = 0
36     # forward pass
37     for t in xrange(len(inputs)):
38         xs[t] = np.zeros((vocab_size,1)) # encode in 1-of-k representation
39         xs[t][inputs[t]] = 1
40         hs[t] = np.tanh(np.dot(wxh, xs[t]) + np.dot(whh, hs[t-1]) + bh) # hidden state
41         ys[t] = np.dot(why, hs[t]) + by # unnormalized log probabilities for next chars
42         ps[t] = np.exp(ys[t]) / np.sum(np.exp(ys[t])) # probabilities for next chars
43         loss += -np.log(ps[t][targets[t],0]) # softmax (cross-entropy loss)
44     # backward pass: compute gradients going backwards
45     dwhx, dwhh, dwhy = np.zeros_like(wxh), np.zeros_like(whh), np.zeros_like(why)
46     dbh, dby = np.zeros_like(bh), np.zeros_like(by)
47     dhnext = np.zeros_like(hs[0])
48     for t in reversed(xrange(len(inputs))):
49         dy = np.copy(ps[t])
50         dy[targets[t]] -= 1 # backprop into y
51         dwhy += np.dot(dy, hs[t].T)
52         dby += dy
53         dh = np.dot(why.T, dy) + dhnext # backprop into h
54         dhrw = (1 - hs[t] * hs[t]) * dh # backprop through tanh nonlinearity
55         dbh += dhrw
56         dwhx += np.dot(dhrw, xs[t].T)
57         dwhh += np.dot(dhrw, hs[t-1].T)
58         dhnext = np.dot(whh.T, dhrw)
59     for dparam in [dwhx, dwhh, dwhy, dbh, dby]:
60         np.clip(dparam, -5, 5, out=dparam) # clip to mitigate exploding gradients
61     return loss, dwhx, dwhh, dwhy, dbh, dby, hs[len(inputs)-1]
62
63 def sample(h, seed_ix, n):
64     """
65     sample a sequence of integers from the model
66     h is memory state, seed_ix is seed letter for first time step
67     """
68     x = np.zeros((vocab_size, 1))
69     x[seed_ix] = 1
70     ixes = []
71     for t in xrange(n):
72         h = np.tanh(np.dot(wxh, x) + np.dot(whh, h) + bh)
73         y = np.dot(why, h) + by
74         p = np.exp(y) / np.sum(np.exp(y))
75         ix = np.random.choice(range(vocab_size), p=p.ravel())
76         x = np.zeros((vocab_size, 1))
77         x[ix] = 1
78         ixes.append(ix)
79     return ixes
80
81 n, p = 0, 0
82 mxhx, mwhh, mwhy = np.zeros_like(wxh), np.zeros_like(whh), np.zeros_like(why)
83 mbh, mby = np.zeros_like(bh), np.zeros_like(by) # memory variables for Adagrad
84 smooth_loss = -np.log(1.0/vocab_size)*seq_length # loss at iteration 0
85 while True:
86     # prepare inputs (we're sweeping from left to right in steps seq_length long)
87     if p+seq_length+1 == len(data) or n == 0:
88         hprev = np.zeros((hidden_size,1)) # reset RNN memory
89         p = 0 # go from start of data
90     inputs = [char_to_ix[ch] for ch in data[p:p+seq_length]]
91     targets = [char_to_ix[ch] for ch in data[p+1:p+seq_length+1]]
92
93     # sample from the model now and then
94     if n % 100 == 0:
95         sample_ix = sample(hprev, inputs[0], 200)
96         txt = ''.join(ix_to_char[ix] for ix in sample_ix)
97         print '....\n%s\n....' % (txt, )
98
99     # forward seq_length characters through the net and fetch gradient
100     loss, dwhx, dwhh, dwhy, dbh, dby, hprev = lossFun(inputs, targets, hprev)
101     smooth_loss = smooth_loss * 0.999 + loss * 0.001
102     if n % 100 == 0: print 'iter %d, loss: %f' % (n, smooth_loss) # print progress
103
104     # perform parameter update with Adagrad
105     for param, dparam, mem in zip([wxh, whh, why, bh, by],
106                                 [dwhx, dwhh, dwhy, dbh, dby],
107                                 [mxhx, mwhh, mwhy, mbh, mby]):
108         mem += dparam * dparam
109         param += -learning_rate * dparam / np.sqrt(mem + 1e-8) # adagrad update
110
111     p += seq_length # move data pointer
112     n += 1 # iteration counter
```

<https://gist.github.com/karpathy/d4dee566867f8291f086>

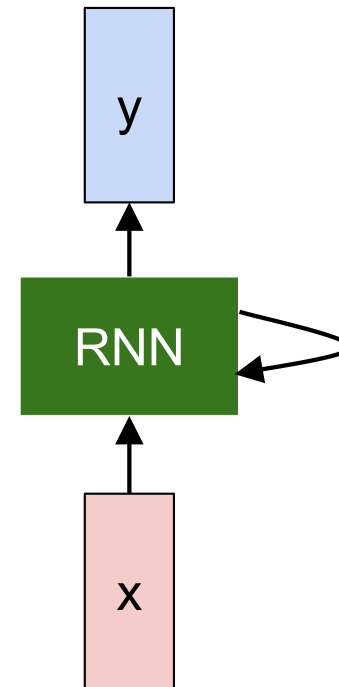
THE SONNETS

by William Shakespeare

37

From fairest creatures we desire increase,
That thereby beauty's rose might never die,
But as the ripper 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 buduriest 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 'nhthnee 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 oftended 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 news 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.

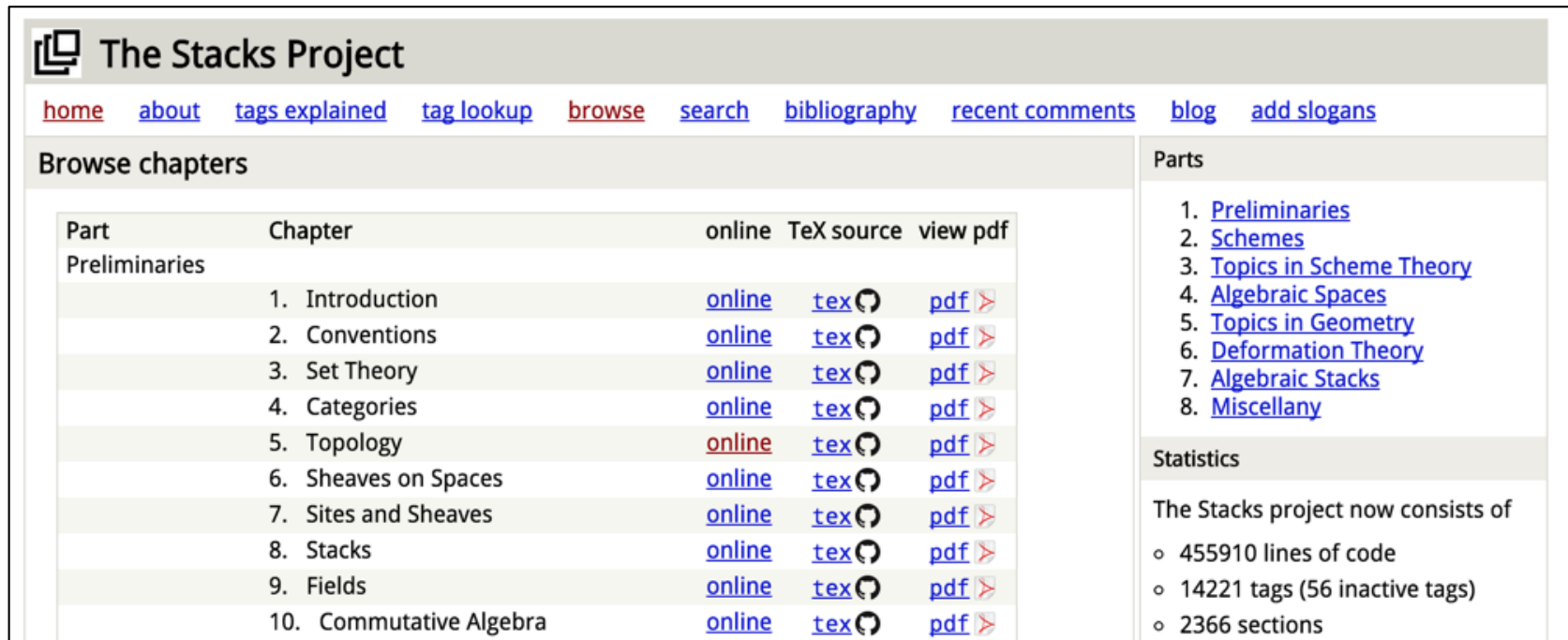
VIOLA:

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.

The Stacks Project: open source algebraic geometry textbook



The screenshot shows the homepage of The Stacks Project. At the top, there is a navigation menu with links: [home](#), [about](#), [tags explained](#), [tag lookup](#), [browse](#), [search](#), [bibliography](#), [recent comments](#), [blog](#), and [add slogans](#). Below the navigation is a section titled "Browse chapters" which contains a table with columns for "Part", "Chapter", "online", "TeX source", and "view pdf". The table lists 10 chapters, each with a corresponding "online" link, "TeX source" link (with a circular refresh icon), and "view pdf" link (with a document icon). To the right of the table is a section titled "Parts" which lists 8 parts: 1. [Preliminaries](#), 2. [Schemes](#), 3. [Topics in Scheme Theory](#), 4. [Algebraic Spaces](#), 5. [Topics in Geometry](#), 6. [Deformation Theory](#), 7. [Algebraic Stacks](#), and 8. [Miscellany](#). Below the "Parts" section is a "Statistics" section which states: "The Stacks project now consists of" followed by a list: 455910 lines of code, 14221 tags (56 inactive tags), and 2366 sections.

Part	Chapter	online	TeX source	view pdf
Preliminaries				
	1. Introduction	online	tex	pdf
	2. Conventions	online	tex	pdf
	3. Set Theory	online	tex	pdf
	4. Categories	online	tex	pdf
	5. Topology	online	tex	pdf
	6. Sheaves on Spaces	online	tex	pdf
	7. Sites and Sheaves	online	tex	pdf
	8. Stacks	online	tex	pdf
	9. Fields	online	tex	pdf
	10. Commutative Algebra	online	tex	pdf

Latex source

<http://stacks.math.columbia.edu/>

The stacks project is licensed under the [GNU Free Documentation License](#)

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 comparicoly 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 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

$$U = \bigcup U_i \times_{S_i} U_i$$

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 $\mathcal{F}_{x_0} = \mathcal{F}_{x_0} = \mathcal{F}_{\mathcal{X}, \dots, 0}$.

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 δ_{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 \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 $\mathcal{U} \subset X$ 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

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

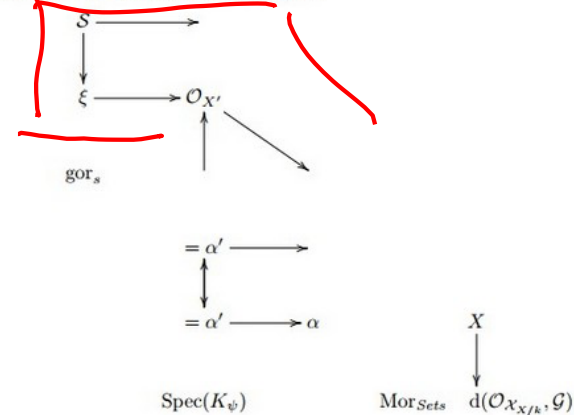
be a morphism of algebraic spaces over S and Y .

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"

$$\mathcal{O}_{X,x} \rightarrow \mathcal{F}_{\bar{x}}^{-1}(\mathcal{O}_{X_{\acute{e}tale}}) \rightarrow \mathcal{O}_{X_t}^{-1} \mathcal{O}_{X_\lambda}(\mathcal{O}_{X_n}^{\bar{v}})$$

is an isomorphism of covering of \mathcal{O}_{X_t} . 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_λ} is a closed immersion, see Lemma ?? . This is a sequence of \mathcal{F} is a similar morphism.



This repository Search

Explore Gist Blog Help



karpathy



torvalds / linux

Watch 3,711

Star 23,054

Fork 9,141

Linux kernel source tree

520,037 commits

1 branch

420 releases

5,039 contributors



branch: master

linux / +



Merge branch 'drm-fixes' of git://people.freedesktop.org/~airlied/linux



torvalds authored 9 hours ago

latest commit 4b1786927d

Documentation	Merge git://git.kernel.org/pub/scm/linux/kernel/git/hab/target-pending	6 days ago
arch	Merge branch 'x86-urgent-for-linux' of git://git.kernel.org/pub/scm/l...	a day ago
block	block: discard bdi_unregister() in favour of bdi_destroy()	9 days ago
crypto	Merge git://git.kernel.org/pub/scm/linux/kernel/git/herbert/crypto-2.6	10 days ago
drivers	Merge branch 'drm-fixes' of git://people.freedesktop.org/~airlied/linux	9 hours ago
firmware	firmware/ihex2fw.c: restore missing default in switch statement	2 months ago
fs	vfs: read file_handle only once in handle_to_path	4 days ago
include	Merge branch 'perf-urgent-for-linux' of git://git.kernel.org/pub/scm/l...	a day ago
init	init: fix regression by supporting devices with major:minor:offset fo...	a month ago
ipc	Merge branch 'for-linux' of git://git.kernel.org/pub/scm/linux/kernel...	a month ago

Code

Pull requests 74

Pulse

Graphs

HTTPS clone URL

https://github.c

You can clone with HTTPS, SSH, or Subversion.

Clone in Desktop

Download ZIP

```

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 & 0x00000000ffffffff) & 0x000000f) << 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 C code

```
/*
 * Copyright (c) 2006-2010, Intel Mobile Communications. All rights reserved.
 *
 * This program is free software; you can redistribute it and/or modify it
 * under the terms of the GNU General Public License version 2 as published by
 * the Free Software Foundation.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 *
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software Foundation,
 * Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
 */
```

```
#include <linux/kexec.h>
#include <linux/errno.h>
#include <linux/io.h>
#include <linux/platform_device.h>
#include <linux/multi.h>
#include <linux/ckevent.h>
```

```
#include <asm/io.h>
#include <asm/prom.h>
#include <asm/e820.h>
#include <asm/system_info.h>
#include <asm/setew.h>
#include <asm/pgproto.h>
```



```

#include <asm/io.h>
#include <asm/prom.h>
#include <asm/e820.h>
#include <asm/system_info.h>
#include <asm/setew.h>
#include <asm/pgproto.h>

#define REG_PG    vesa_slot_addr_pack
#define PFM_NOCOMP AFSR(0, load)
#define STACK_DDR(type)    (func)

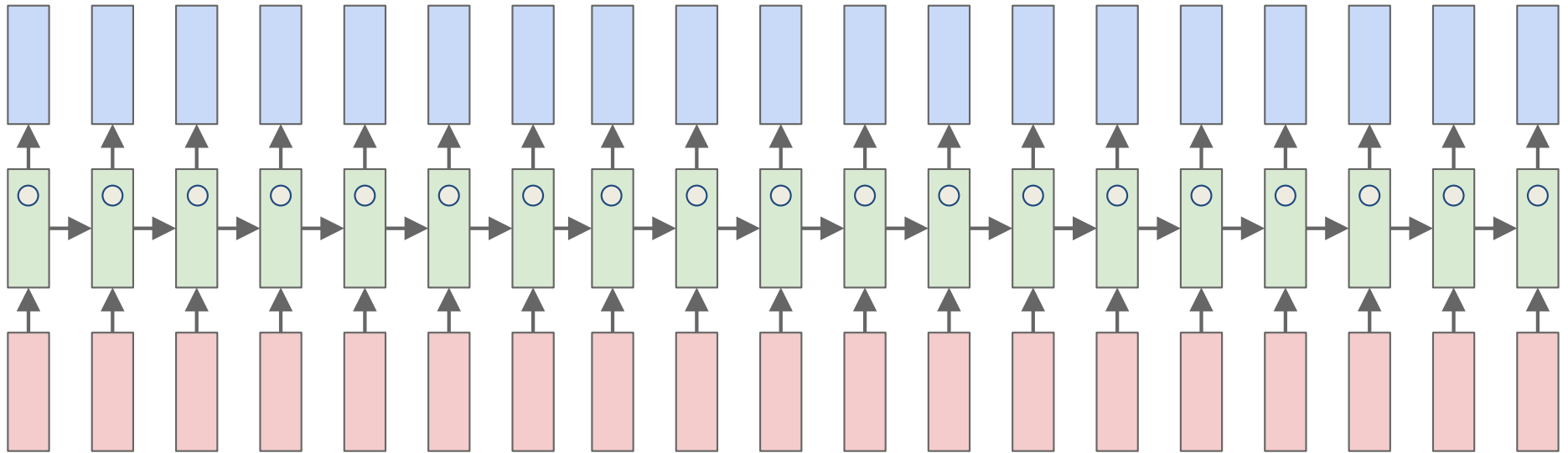
#define SWAP_ALLOCATE(nr)    (e)
#define emulate_sigs()    arch_get_unaligned_child()
#define access_rw(TST)    asm volatile("movd %!esp, %0, %3" : : "r" (0)); \
    if (__type & DO_READ)

static void stat_PC_SEC __read_mostly offsetof(struct seq_argsqueue, \
    pC>[1]);

static void
os_prefix(unsigned long sys)
{
#ifdef CONFIG_PREEMPT
    PUT_PARAM_RAID(2, sel) = get_state_state();
    set_pid_sum((unsigned long)state, current_state_str(),
        (unsigned long)-1->lr_full; low;
}

```

Searching for interpretable cells



Karpathy, Johnson, and Fei-Fei: Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016

Searching for interpretable cells

```
/* Unpack a filter field's string representation from user-space
 * buffer. */
char *audit_unpack_string(void **bufp, size_t *remain, size_t len)
{
    char *str;
    if (!*bufp || (len == 0) || (len > *remain))
        return ERR_PTR(-EINVAL);
    /* Of the currently implemented string fields, PATH_MAX
     * defines the longest valid length.
     */
}
```

Karpathy, Johnson, and Fei-Fei: Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016

Figures copyright Karpathy, Johnson, and Fei-Fei, 2015; reproduced with permission

Searching for interpretable cells

"You mean to imply that I have nothing to eat out of.... On the contrary, I can supply you with everything even if you want to give dinner parties," warmly replied Chichagov, who tried by every word he spoke to prove his own rectitude and therefore imagined Kutuzov to be animated by the same desire.

Kutuzov, shrugging his shoulders, replied with his subtle penetrating smile: "I meant merely to say what I said."

quote detection cell

Karpathy, Johnson, and Fei-Fei: Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016

Figures copyright Karpathy, Johnson, and Fei-Fei, 2015; reproduced with permission

Searching for interpretable cells

Cell sensitive to position in line:

The sole importance of the crossing of the Berezina lies in the fact that it plainly and indubitably proved the fallacy of all the plans for cutting off the enemy's retreat and the soundness of the only possible line of action--the one Kutuzov and the general mass of the army demanded--namely, simply to follow the enemy up. The French crowd fled at a continually increasing speed and all its energy was directed to reaching its goal. It fled like a wounded animal and it was impossible to block its path. This was shown not so much by the arrangements it made for crossing as by what took place at the bridges. When the bridges broke down, unarmed soldiers, people from Moscow and women with children who were with the French transport, all--carried on by vis inertiae--pressed forward into boats and into the ice-covered water and did not, surrender.

line length tracking cell

Karpathy, Johnson, and Fei-Fei: Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016

Figures copyright Karpathy, Johnson, and Fei-Fei, 2015; reproduced with permission

Searching for interpretable cells

```
static int __dequeue_signal(struct sigpending *pending, sigset_t *mask,
                           siginfo_t *info)
{
    int sig = next_signal(pending, mask);
    if (sig) {
        if (current->notifier) {
            if (sigismember(current->notifier_mask, sig)) {
                if (!(current->notifier)(current->notifier_data)) {
                    clear_thread_flag(TIF_SIGPENDING);
                    return 0;
                }
            }
        }
        collect_signal(sig, pending, info);
    }
    return sig;
}
```

if statement cell

Karpathy, Johnson, and Fei-Fei: Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016

Figures copyright Karpathy, Johnson, and Fei-Fei, 2015; reproduced with permission

Searching for interpretable cells

Cell that turns on inside comments and quotes:

```
/* Duplicate LSM field information. The lsm_rule is opaque, so
 * re-initialized. */
static inline int audit_dupe_lsm_field(struct audit_field *df,
                                     struct audit_field *sf)
{
    int ret = 0;
    char *lsm_str;
    /* our own copy of lsm_str */
    lsm_str = kstrdup(sf->lsm_str, GFP_KERNEL);
    if (unlikely(!lsm_str))
        return -ENOMEM;
    df->lsm_str = lsm_str;
    /* our own (refreshed) copy of lsm_rule */
    ret = security_audit_rule_init(df->type, df->op, df->lsm_str,
                                   (void **)&df->lsm_rule);
    /* Keep currently invalid fields around in case they
     * become valid after a policy reload. */
    if (ret == -EINVAL) {
        pr_warn("audit rule for LSM '%s' is invalid\n",
                df->lsm_str);
        ret = 0;
    }
    return ret;
}
```

quote/comment cell

Karpathy, Johnson, and Fei-Fei: Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016

Figures copyright Karpathy, Johnson, and Fei-Fei, 2015; reproduced with permission

Searching for interpretable cells

```
#ifdef CONFIG_AUDITSYSCALL
static inline int audit_match_class_bits(int class, u32 *mask)
{
    int i;
    if (classes[class]) {
        for (i = 0; i < AUDIT_BITMASK_SIZE; i++)
            if (mask[i] & classes[class][i])
                return 0;
    }
    return 1;
}
```

code depth cell

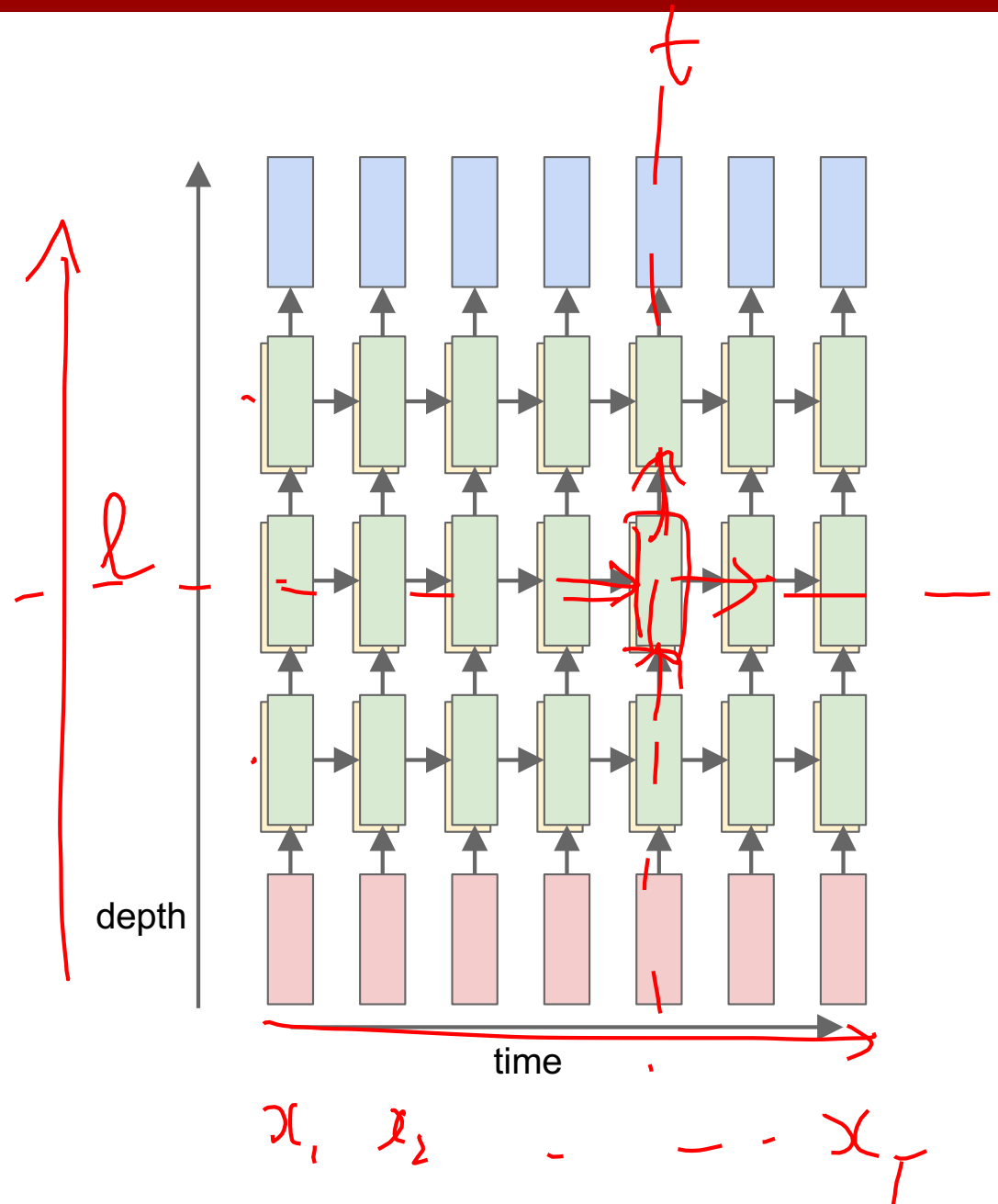
Karpathy, Johnson, and Fei-Fei: Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016

Figures copyright Karpathy, Johnson, and Fei-Fei, 2015; reproduced with permission

Multilayer RNNs

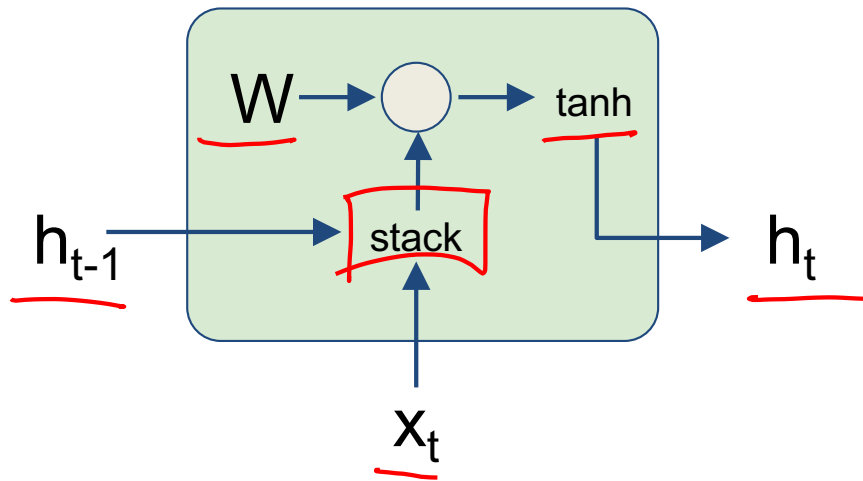
$$h_t^l = \tanh W^l \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix} + b^l$$

$h \in \mathbb{R}^n$ $W^l [n \times 2n]$



Vanilla RNN Gradient Flow

Bengio et al, "Learning long-term dependencies with gradient descent is difficult", IEEE Transactions on Neural Networks, 1994
Pascanu et al, "On the difficulty of training recurrent neural networks", ICML 2013

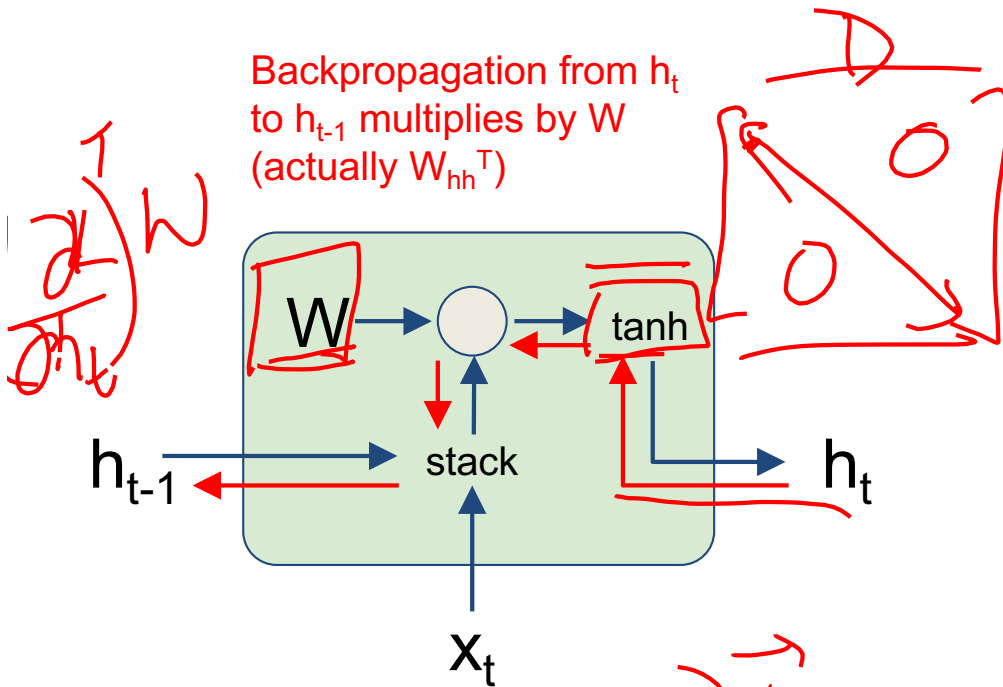


$$\begin{aligned} h_t &= \tanh(W_{hh}h_{t-1} + W_{hx}x_t) \\ &= \tanh\left(\begin{pmatrix} W_{hh} & W_{hx} \end{pmatrix} \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right) \\ &= \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right) \end{aligned}$$

Vanilla RNN Gradient Flow

Bengio et al, "Learning long-term dependencies with gradient descent is difficult", IEEE Transactions on Neural Networks, 1994
 Pascanu et al, "On the difficulty of training recurrent neural networks", ICML 2013

Backpropagation from h_t to h_{t-1} multiplies by W (actually W_{hh}^T)



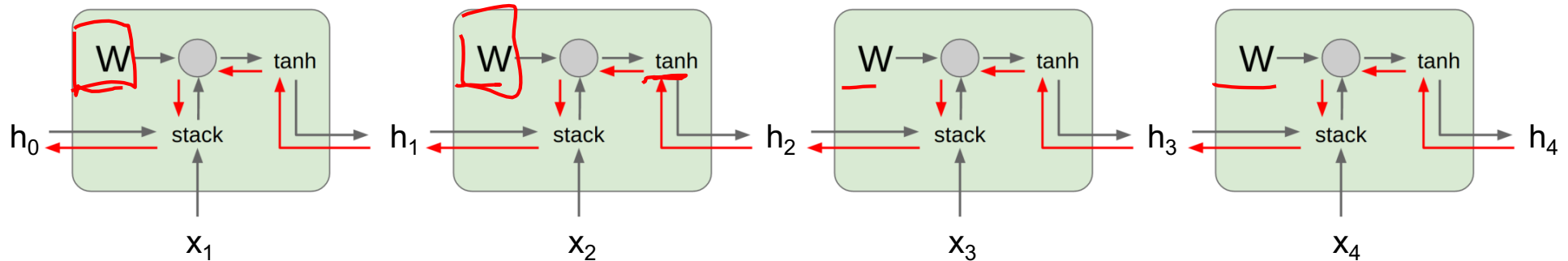
$$\begin{aligned}
 h_t &= \tanh(W_{hh}h_{t-1} + W_{hx}x_t) \\
 &= \tanh\left(\begin{pmatrix} W_{hh} & W_{hx} \end{pmatrix} \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right) \\
 &= \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)
 \end{aligned}$$

$$\frac{\partial L}{\partial h_{t-1}} = \left(\frac{\partial L}{\partial h_t} \right)^T W$$

$$\frac{\partial h_t}{\partial h_{t-1}} = W$$

Vanilla RNN Gradient Flow

Bengio et al, "Learning long-term dependencies with gradient descent is difficult", IEEE Transactions on Neural Networks, 1994
 Pascanu et al, "On the difficulty of training recurrent neural networks", ICML 2013

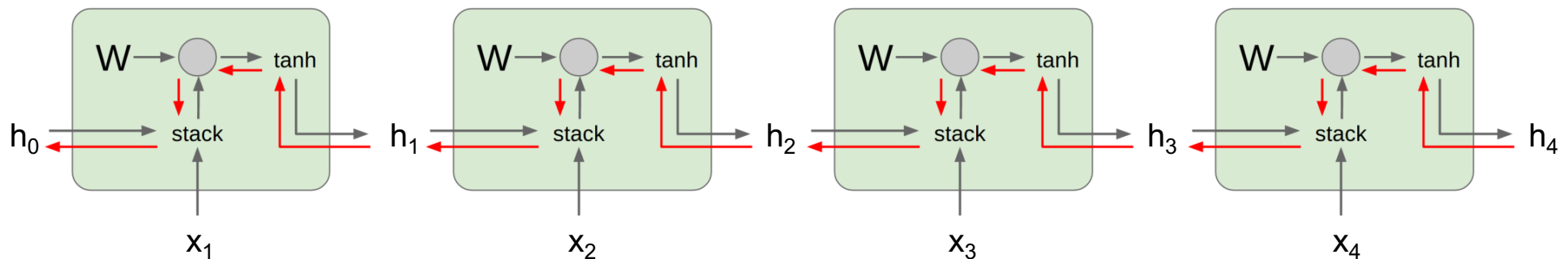


Computing gradient of h_0 involves many factors of W (and repeated tanh)

$$\frac{\partial h_4}{\partial h_0} = \begin{matrix} \frac{\partial h_4}{\partial h_3} \\ \vdots \\ \frac{\partial h_1}{\partial h_0} \end{matrix} = \underbrace{W^A}_{\text{red}} = \underbrace{Q \Lambda Q^T}_{\text{red}}$$

Vanilla RNN Gradient Flow

Bengio et al, "Learning long-term dependencies with gradient descent is difficult", IEEE Transactions on Neural Networks, 1994
Pascanu et al, "On the difficulty of training recurrent neural networks", ICML 2013



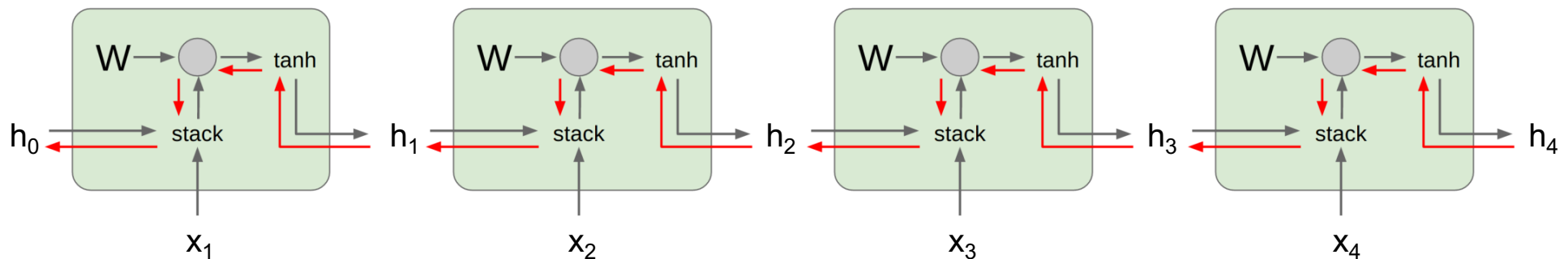
Computing gradient of h_0 involves many factors of W (and repeated tanh)

Largest singular value > 1 :
Exploding gradients

Largest singular value < 1 :
Vanishing gradients

Vanilla RNN Gradient Flow

Bengio et al, "Learning long-term dependencies with gradient descent is difficult", IEEE Transactions on Neural Networks, 1994
Pascanu et al, "On the difficulty of training recurrent neural networks", ICML 2013



Computing gradient of h_0 involves many factors of W (and repeated tanh)

Largest singular value > 1 :
Exploding gradients

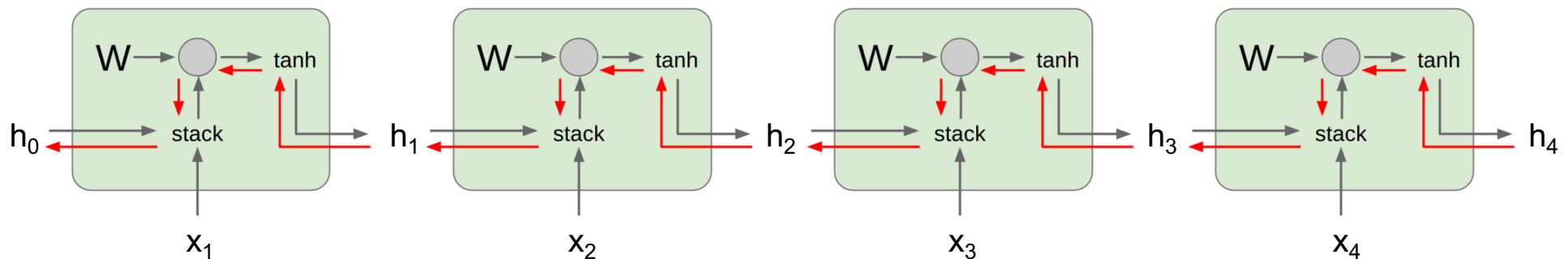
Largest singular value < 1 :
Vanishing gradients

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

Bengio et al, "Learning long-term dependencies with gradient descent is difficult", IEEE Transactions on Neural Networks, 1994
Pascanu et al, "On the difficulty of training recurrent neural networks", ICML 2013



Computing gradient of h_0 involves many factors of W (and repeated tanh)

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

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

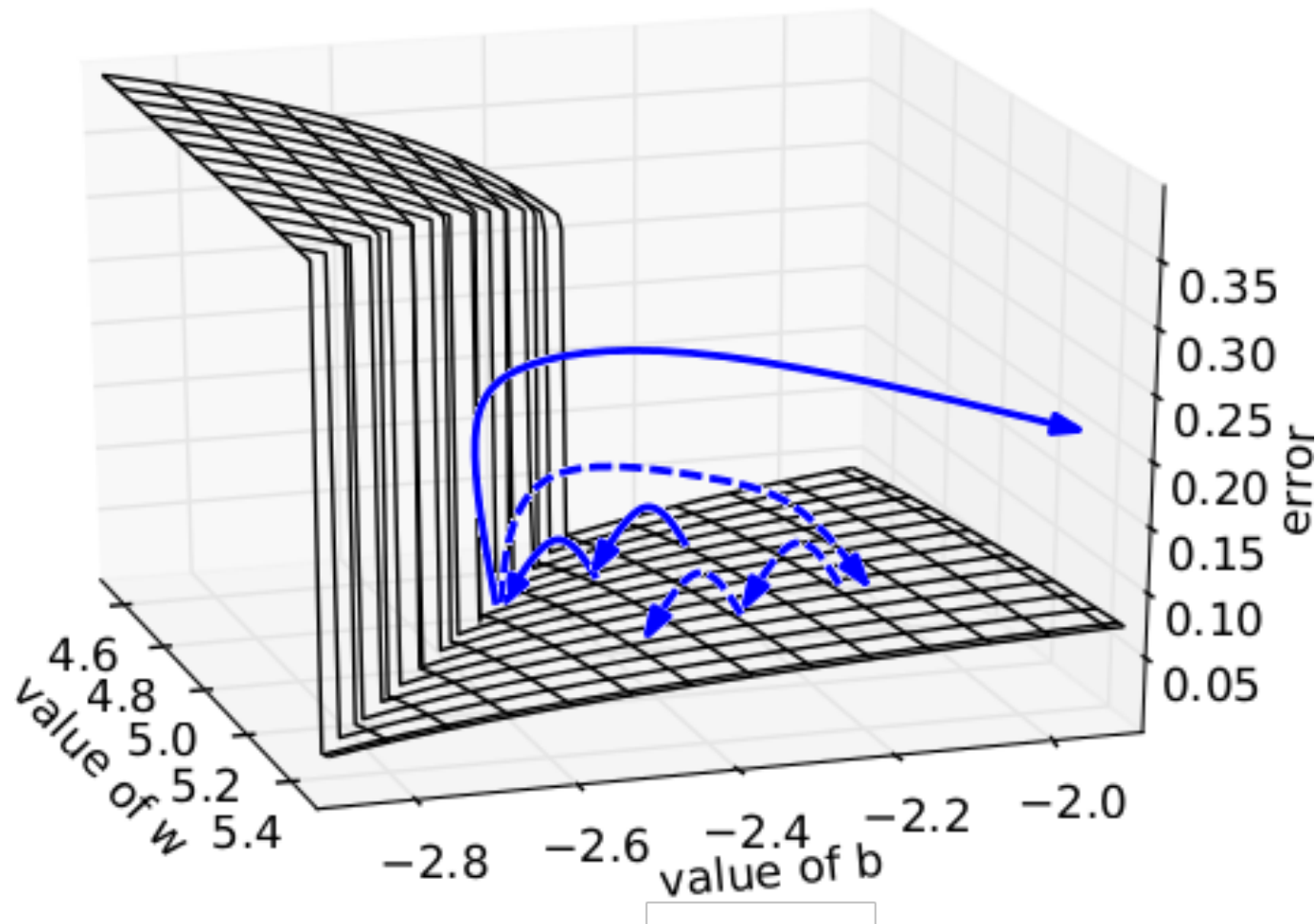
$$h_t = o \odot \tanh(c_t)$$

Hochreiter and Schmidhuber, "Long Short Term Memory", Neural Computation
1997

Illustration [Pascanu et al]

- Intuition

- Error surface of a single hidden unit RNN; High curvature walls
- Solid lines: standard gradient descent trajectories
- Dashed lines: gradient rescaled to fix problem



Fix #1

- Pseudocode

Algorithm 1 Pseudo-code for norm clipping the gradients whenever they explode

$$\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta}$$

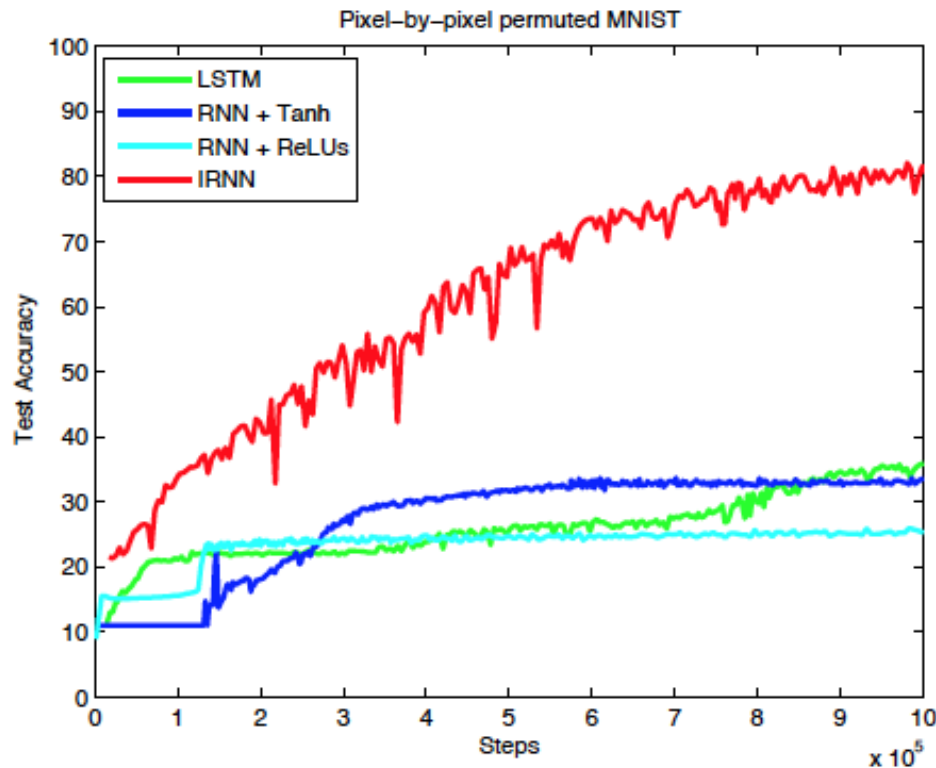
if $\|\hat{\mathbf{g}}\| \geq \textit{threshold}$ **then**

$$\hat{\mathbf{g}} \leftarrow \frac{\textit{threshold}}{\|\hat{\mathbf{g}}\|} \hat{\mathbf{g}}$$

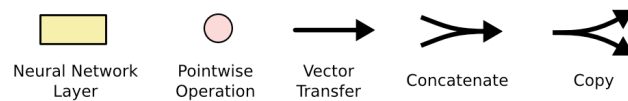
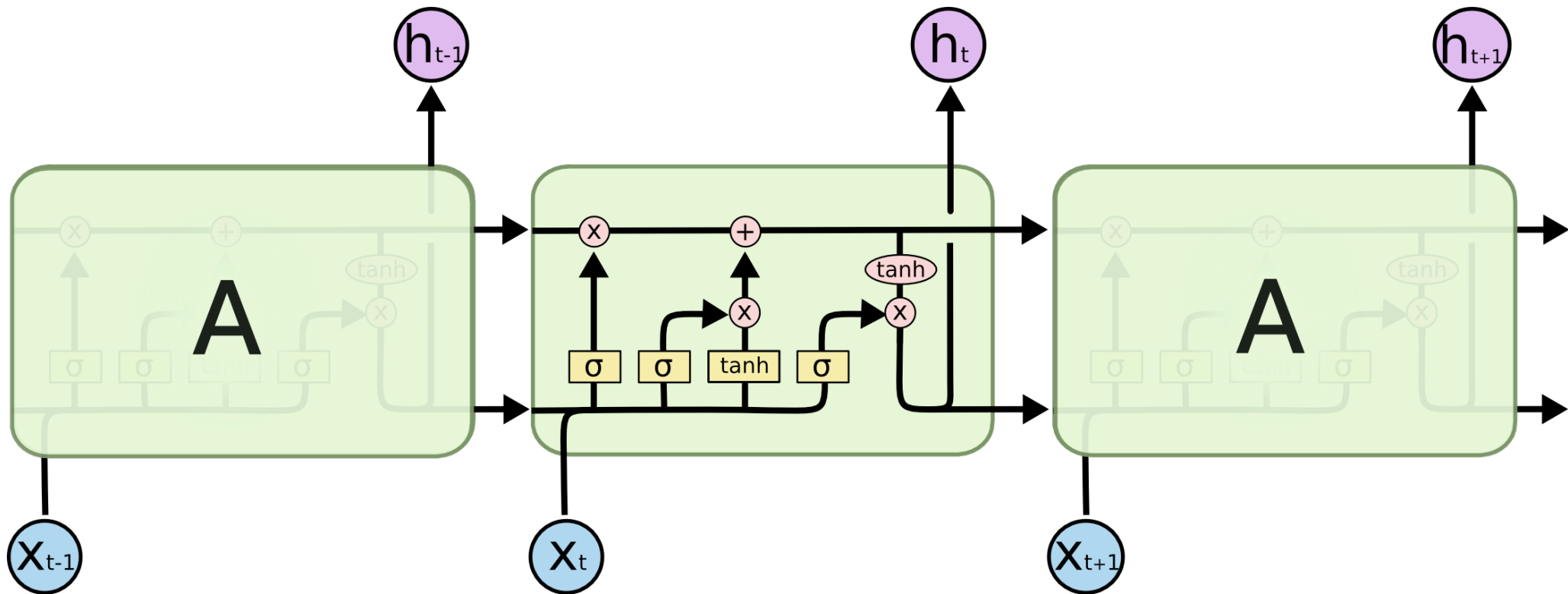
end if

Fix #2

- Smart Initialization and ReLus
 - [Socher et al 2013]
 - *A Simple Way to Initialize Recurrent Networks of Rectified Linear Units*, Le et al. 2015

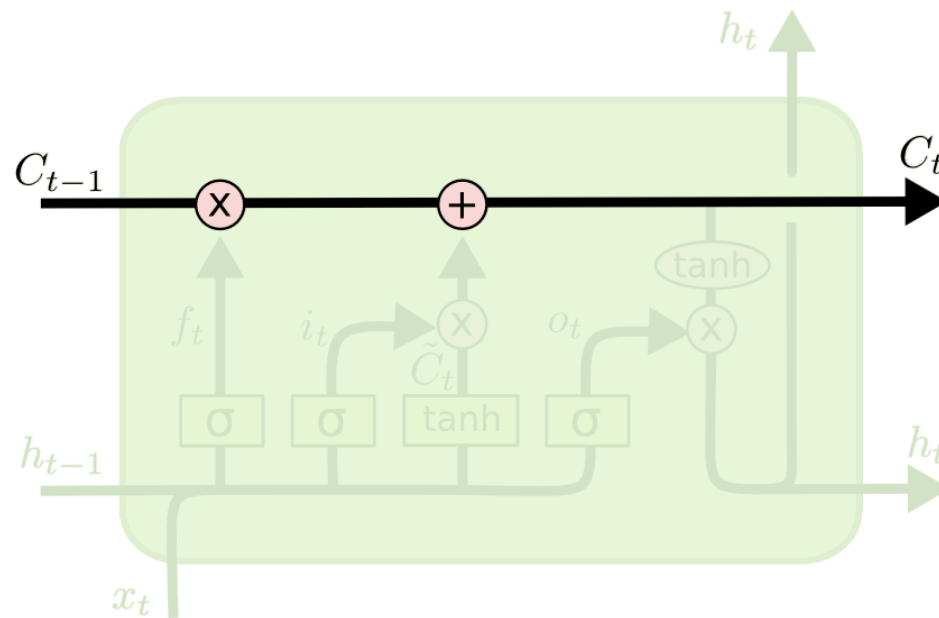


Meet LSTMs



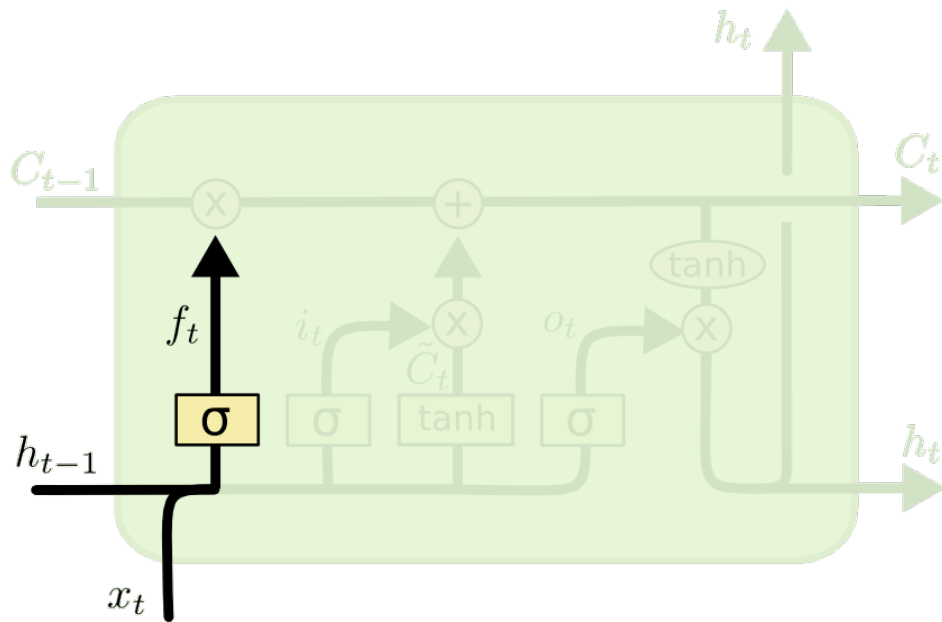
LSTMs Intuition: Memory

- Cell State / Memory



LSTMs Intuition: Forget Gate

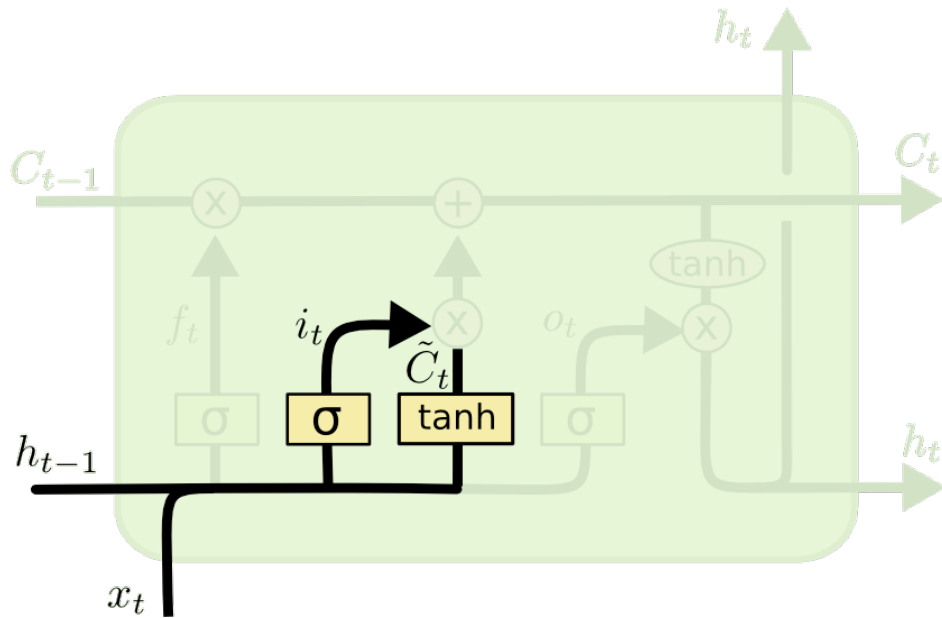
- Should we continue to remember this “bit” of information or not?



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

LSTMs Intuition: Input Gate

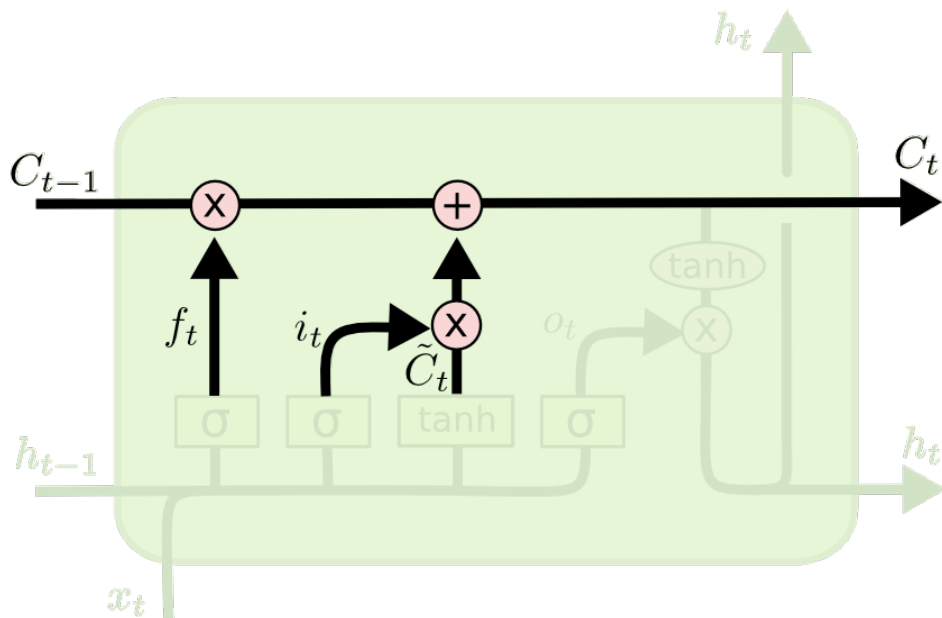
- Should we update this “bit” of information or not?
 - If so, with what?



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

LSTMs Intuition: Memory Update

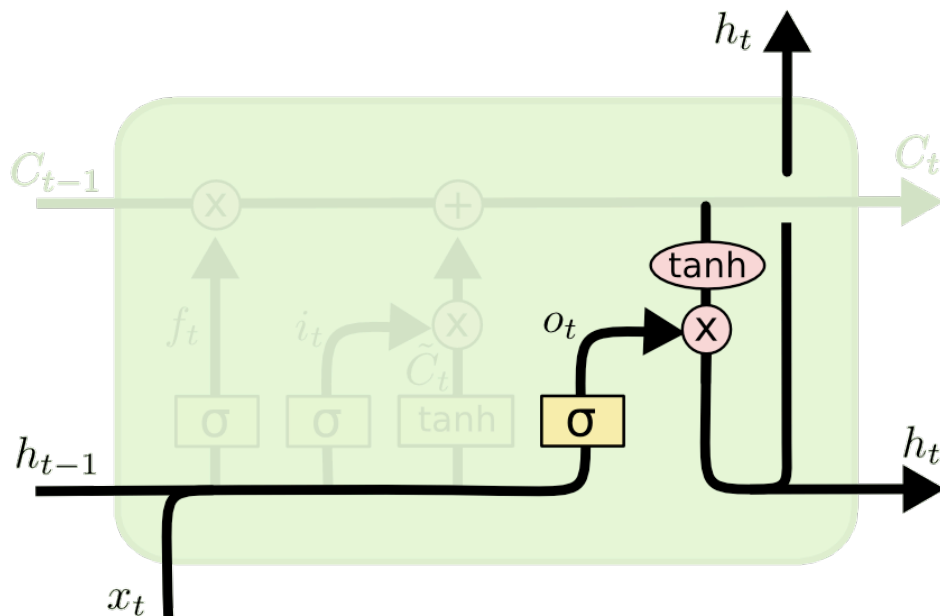
- Forget that + memorize this



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

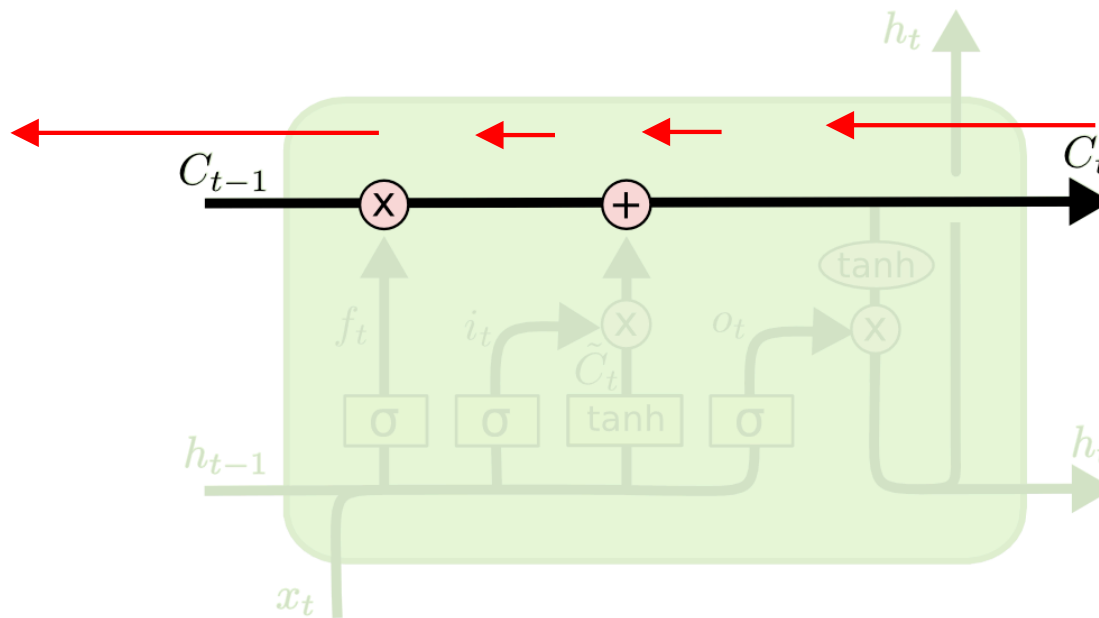
LSTMs Intuition: Output Gate

- Should we output this “bit” of information to “deeper” layers?



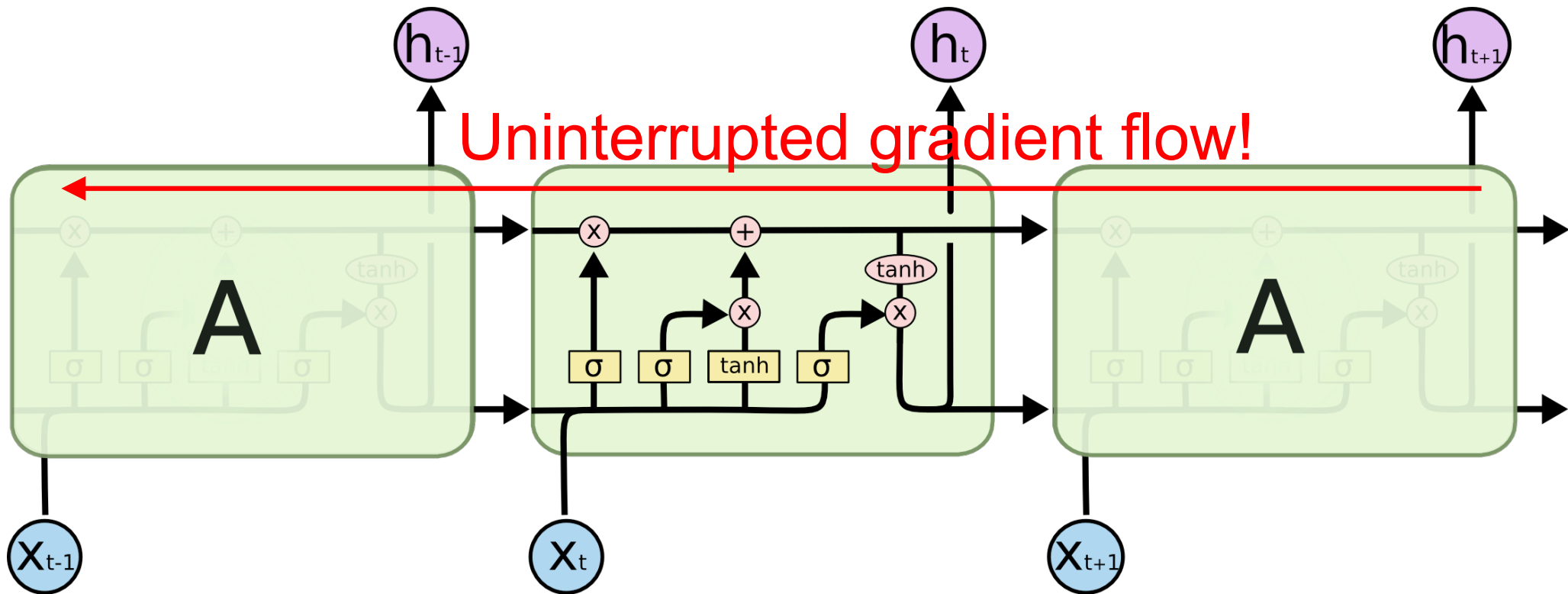
$$o_t = \sigma (W_o [h_{t-1}, x_t] + b_o)$$
$$h_t = o_t * \tanh (C_t)$$

LSTMs Intuition: Additive Updates

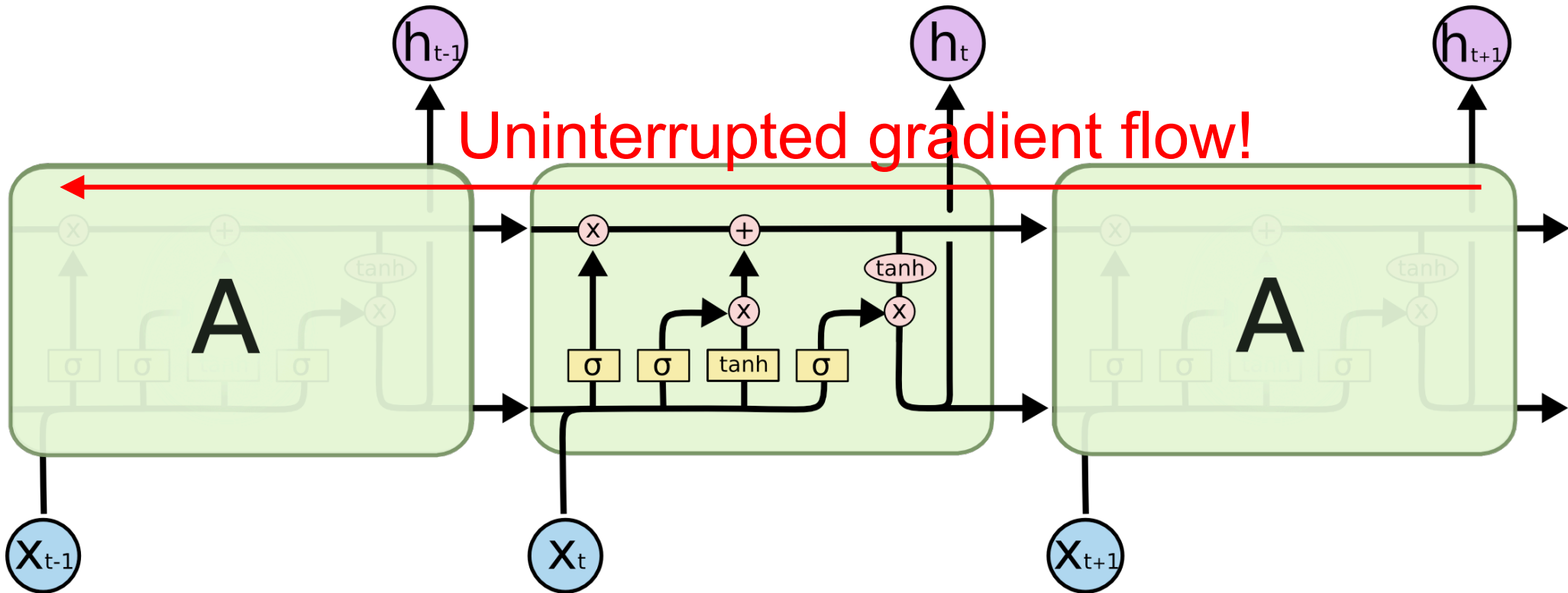


Backpropagation from c_t to c_{t-1} only
elementwise
multiplication by f , no
matrix multiply by W

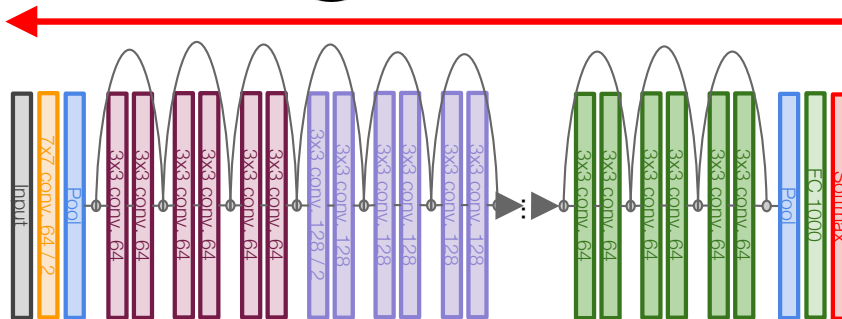
LSTMs Intuition: Additive Updates



LSTMs Intuition: Additive Updates

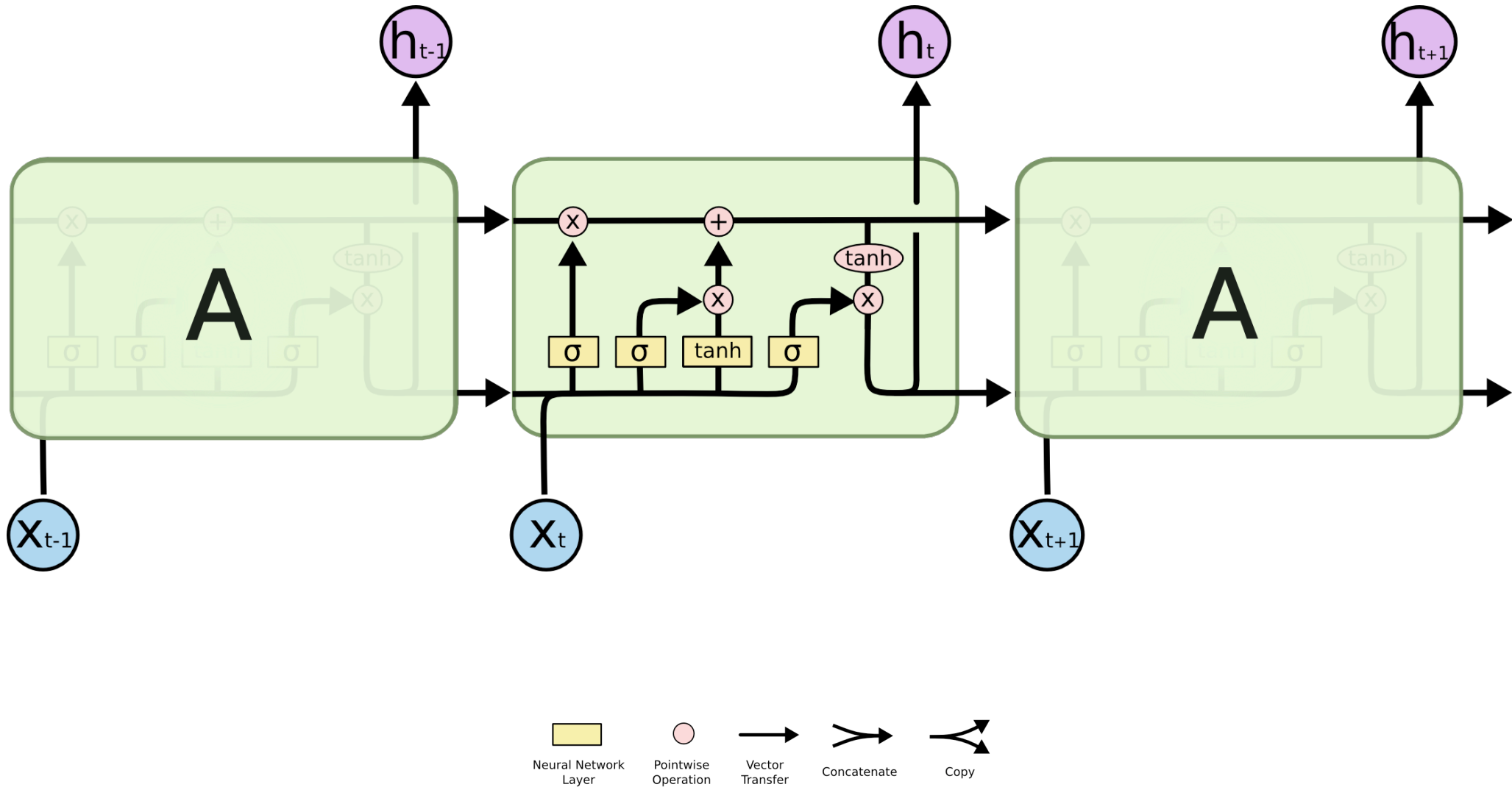


Similar to ResNet!



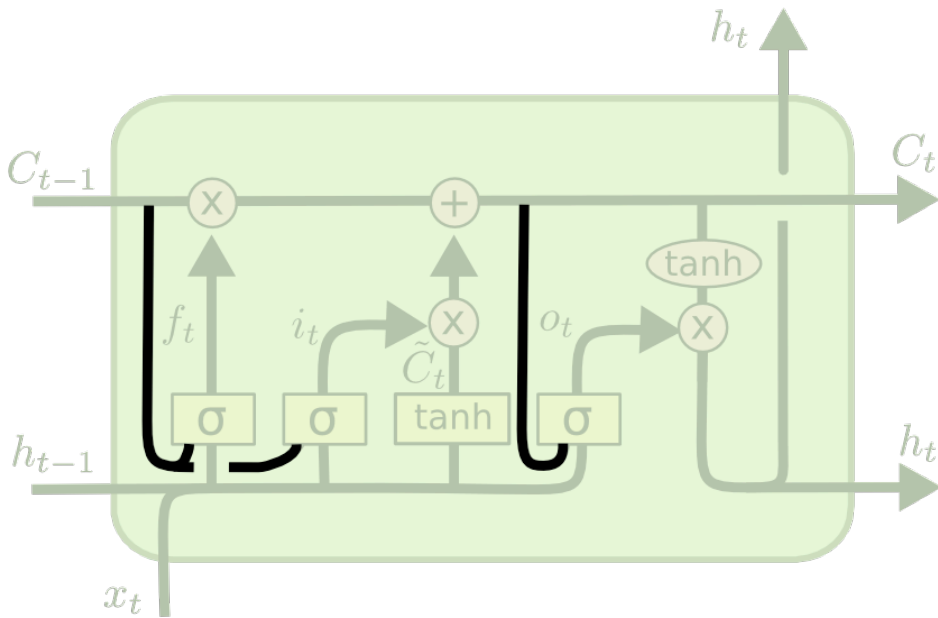
LSTMs

- A pretty sophisticated cell



LSTM Variants #1: Peephole Connections

- Let gates see the cell state / memory



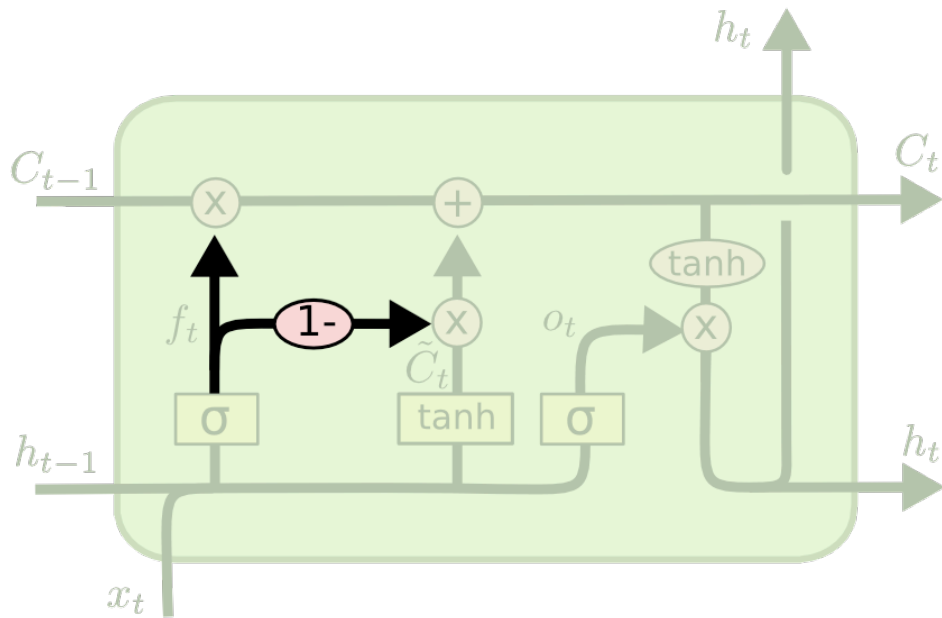
$$f_t = \sigma (W_f \cdot [C_{t-1}, h_{t-1}, x_t] + b_f)$$

$$i_t = \sigma (W_i \cdot [C_{t-1}, h_{t-1}, x_t] + b_i)$$

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

LSTM Variants #2: Coupled Gates

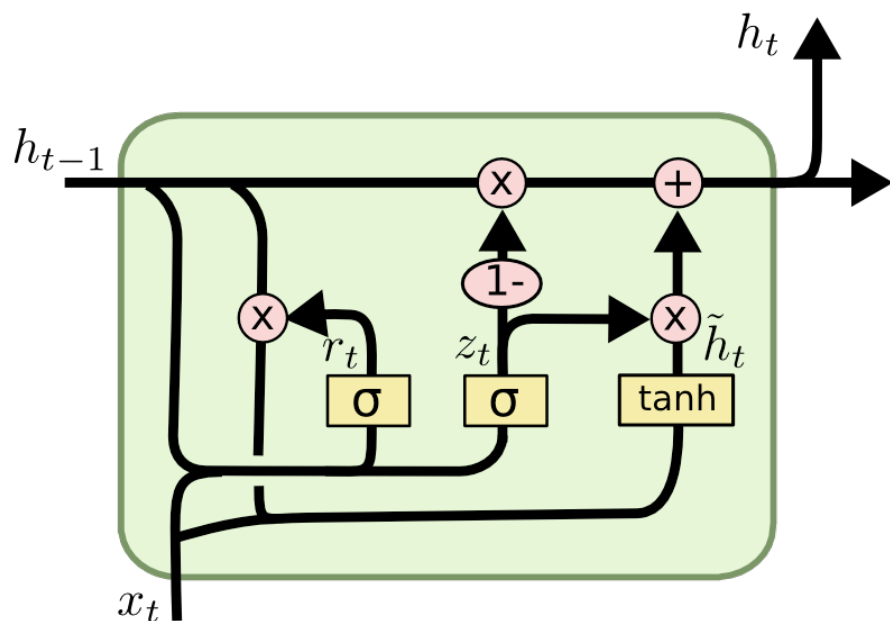
- Only memorize new if forgetting old



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

LSTM Variants #3: Gated Recurrent Units

- Changes:
 - No explicit memory; memory = hidden output
 - Z = memorize new and forget old



$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$

$$\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t])$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$