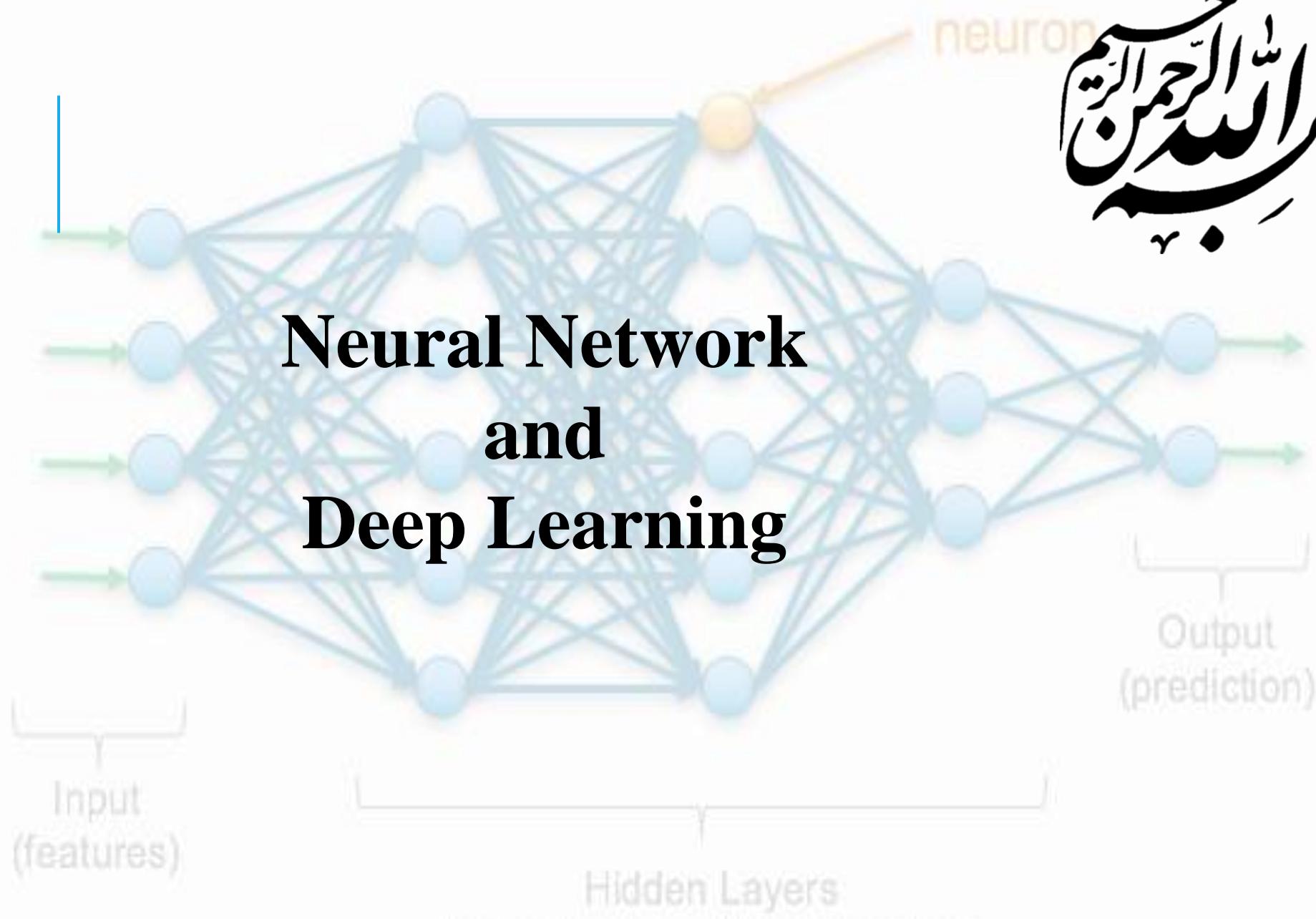


بِسْمِ اللّٰهِ الرَّحْمٰنِ الرَّحِيْمِ

# Neural Network and Deep Learning



# Lecture 6:

# Recurrent Neural Networks (RNN)

**lecture inspiration:**

Fei-Fei Li & Justin Johnson & Serena Yeung, 2019

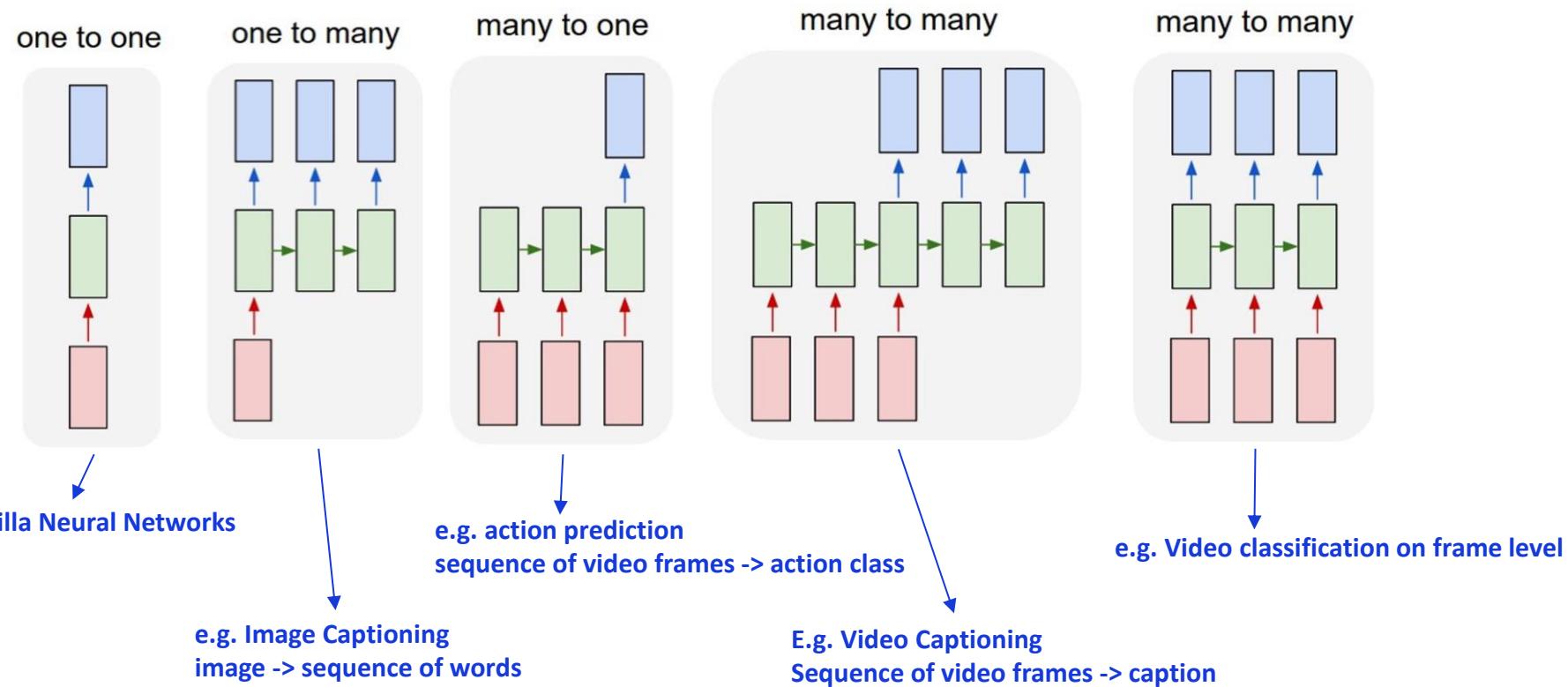
Fei-Fei Li, Ehsan Adeli, 2023

Some papers

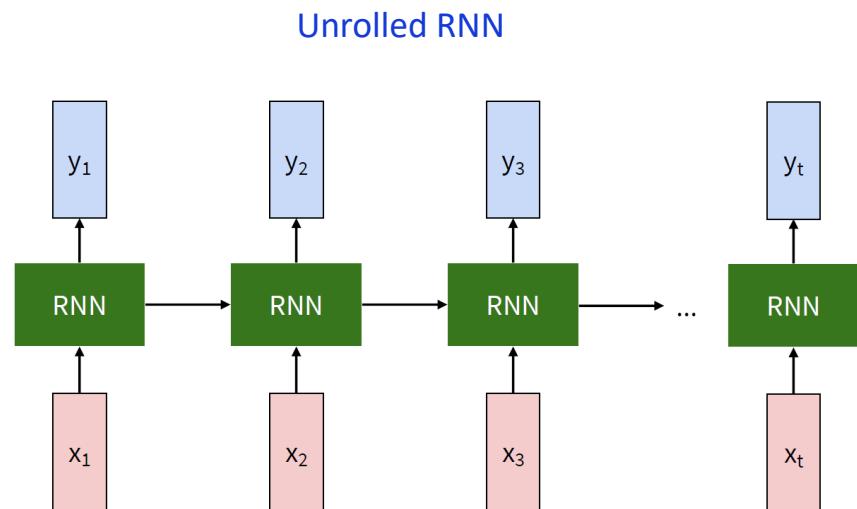
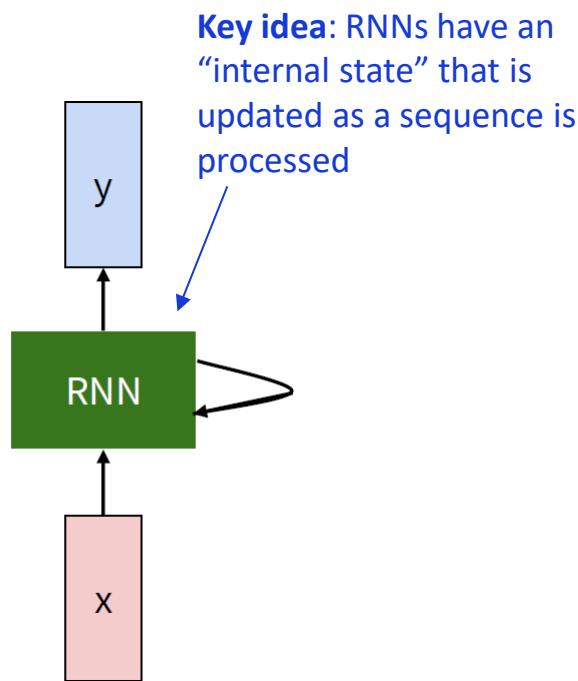
# OUTLINE

- Recurrent Neural Networks
- Long Short Term Memory (LSTM)

# RECURRENT NEURAL NETWORKS: PROCESS SEQUENCES



# RECURRENT NEURAL NETWORK

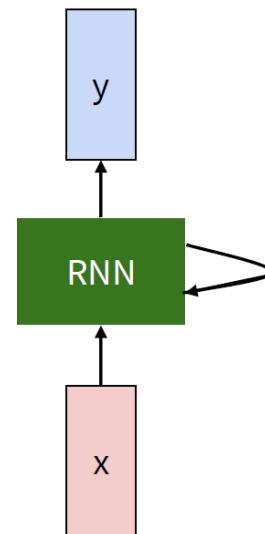


# RNN HIDDEN STATE UPDATE

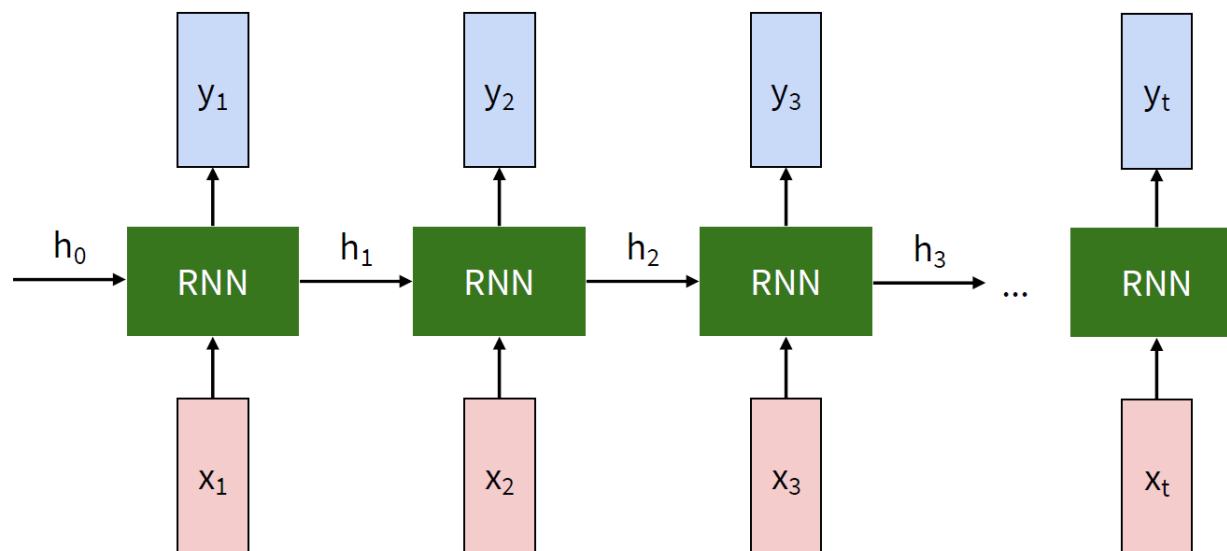
- We can process a sequence of vectors  $x$  by applying a recurrence formula at every time step:

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

new state      \ /      old state      input vector at  
some function      |      some time step  
with parameters W



# RECURRENT NEURAL NETWORK

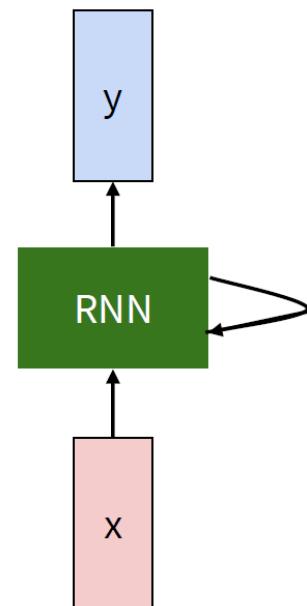


# RECURRENT NEURAL NETWORK

- We can process a sequence of vectors  $x$  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.



# (VANILLA) RECURRENT NEURAL NETWORK

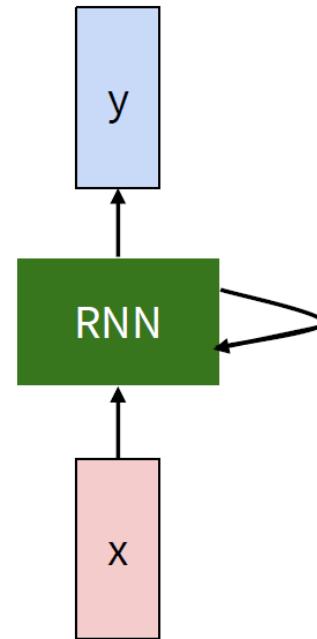
➤ The state consists of a single “hidden” vector  $h$ :

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



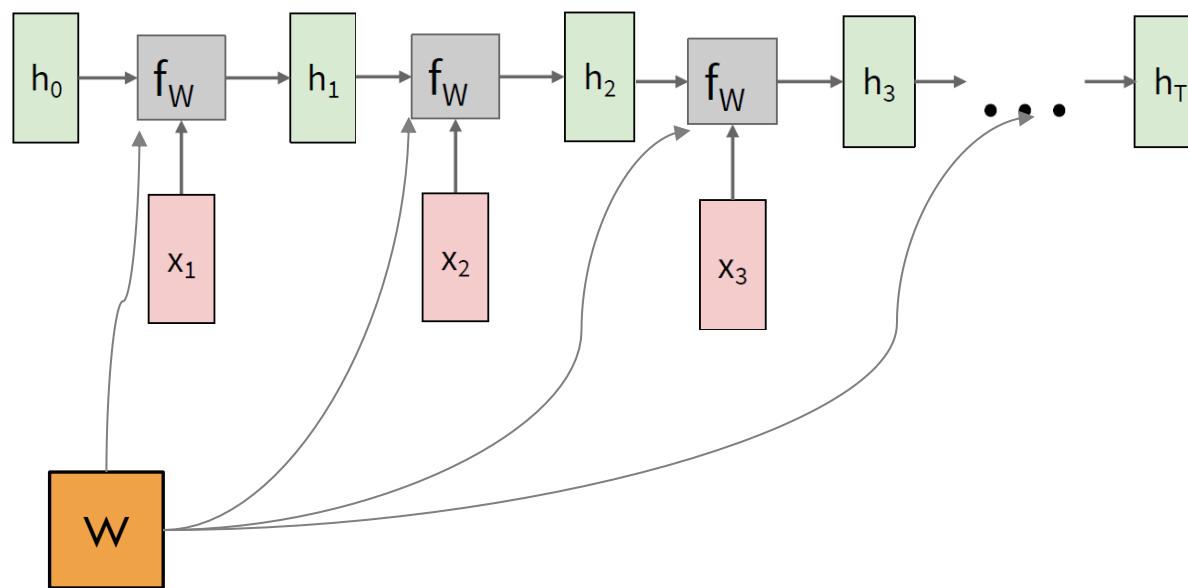
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t$$

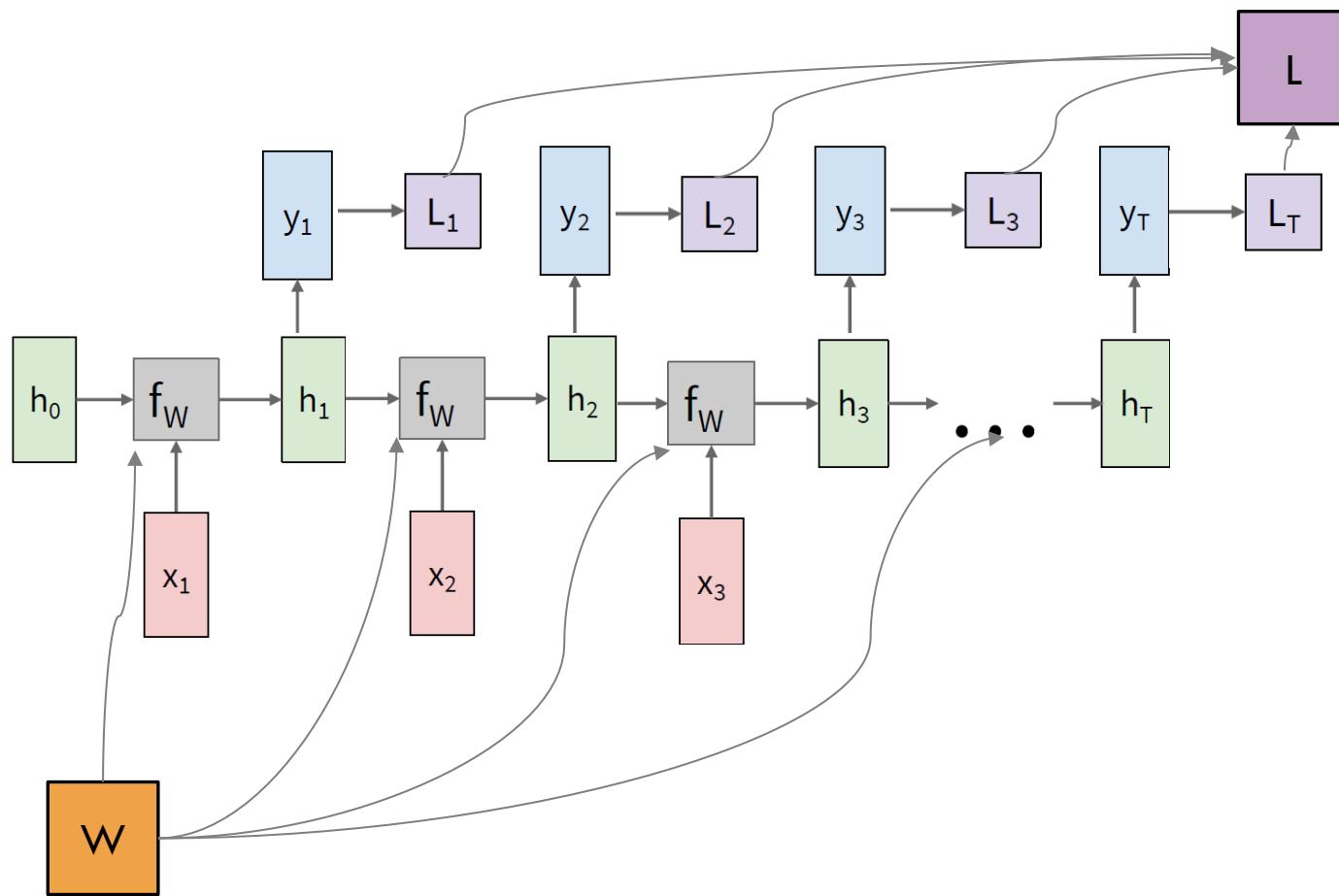


# RNN: COMPUTATIONAL GRAPH

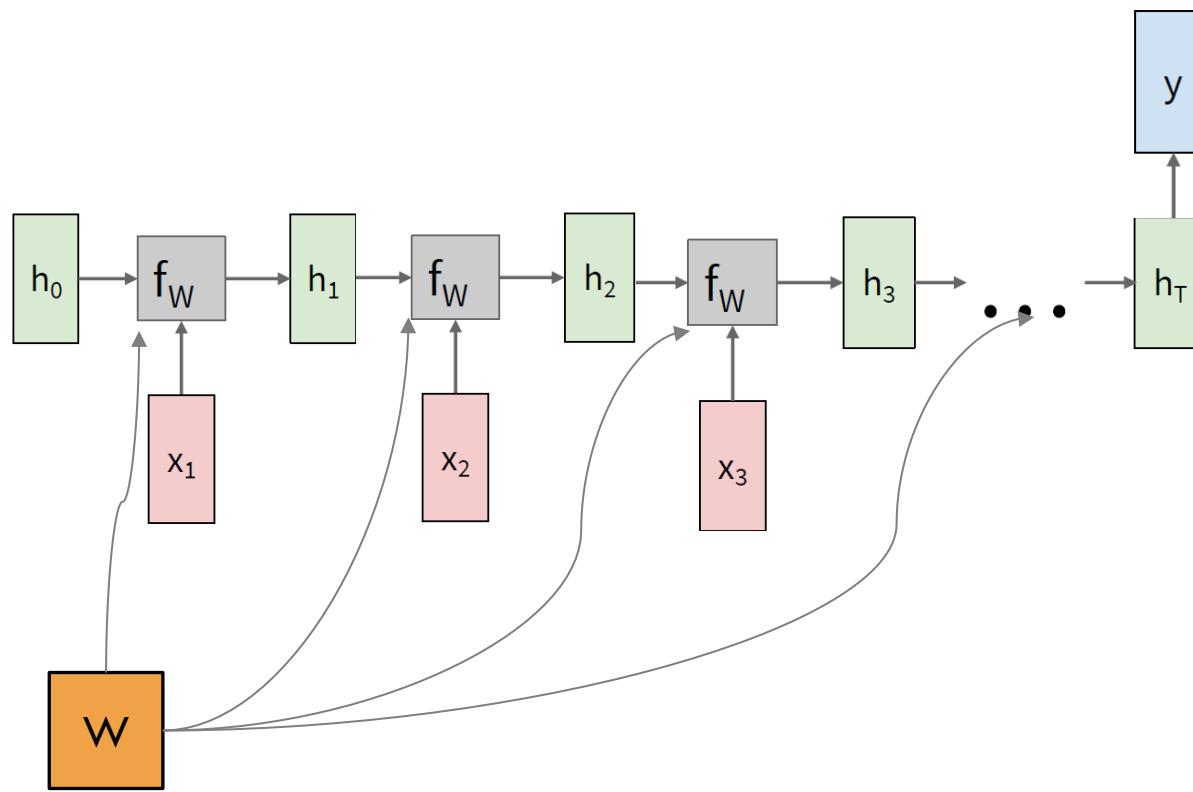
Re-use the same weight matrix at every time-step



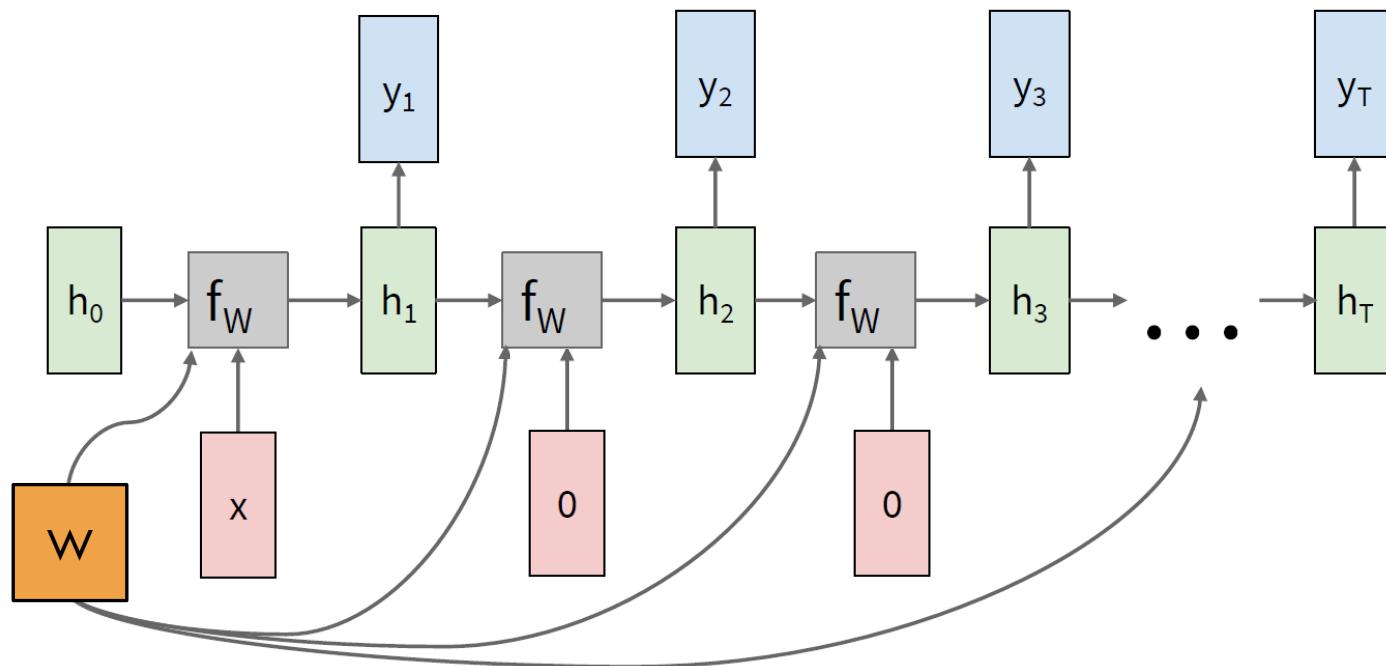
# RNN: COMPUTATIONAL GRAPH: MANY TO MANY



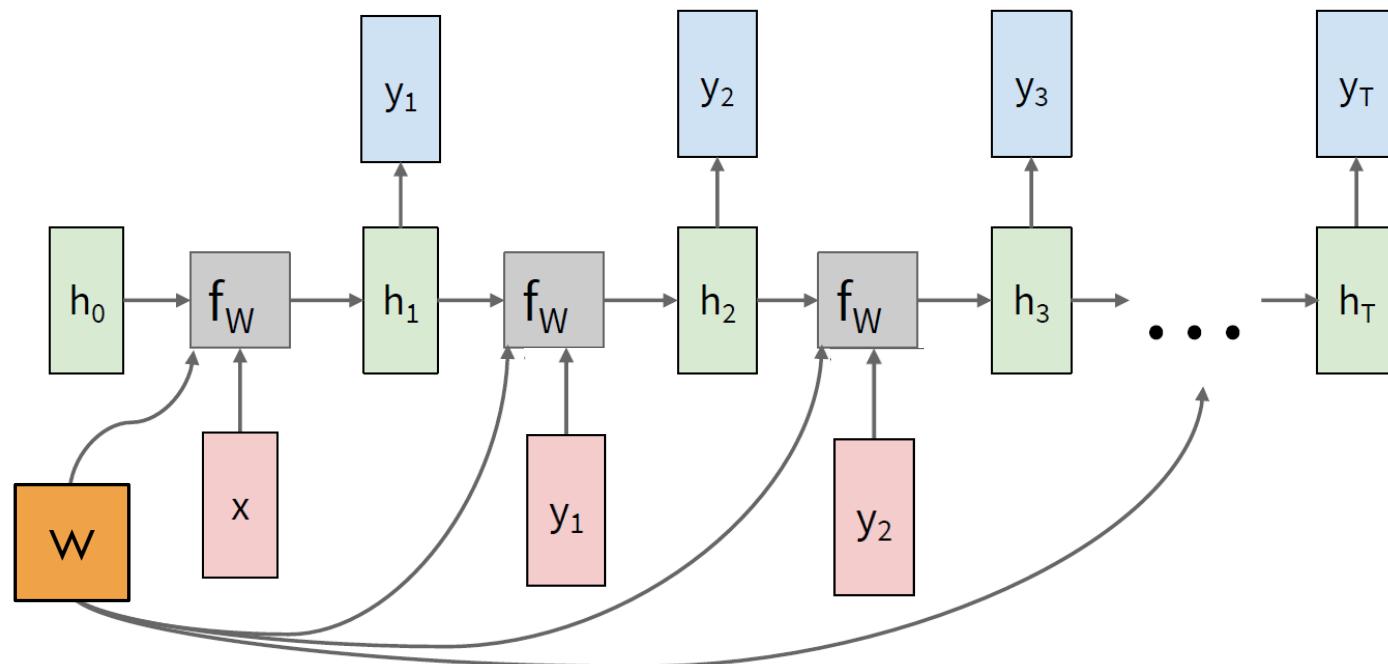
# RNN: COMPUTATIONAL GRAPH: MANY TO ONE



# RNN: COMPUTATIONAL GRAPH: ONE TO MANY

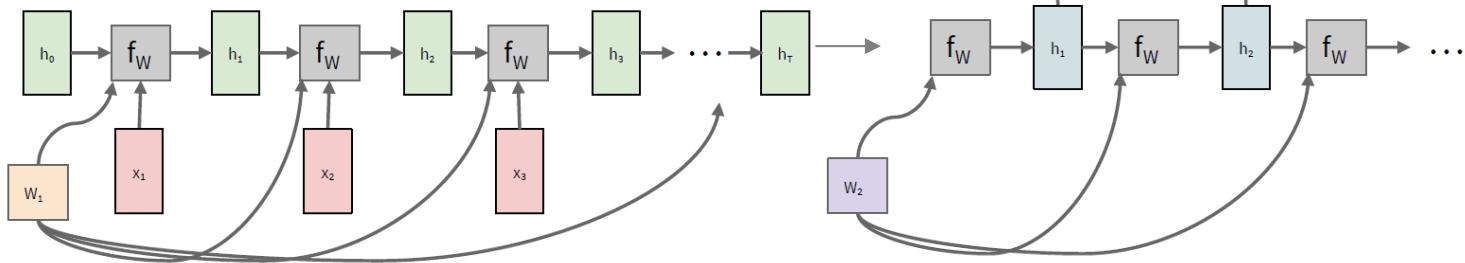


# RNN: COMPUTATIONAL GRAPH: ONE TO MANY



# SEQUENCE TO SEQUENCE: MANY-TO-ONE + ONE-TO-MANY

Many to one: Encode input sequence  
in a single vector



One to many: Produce output  
sequence from single input vector

Sutskever et al, "Sequence to Sequence Learning with  
Neural Networks", NIPS 2014

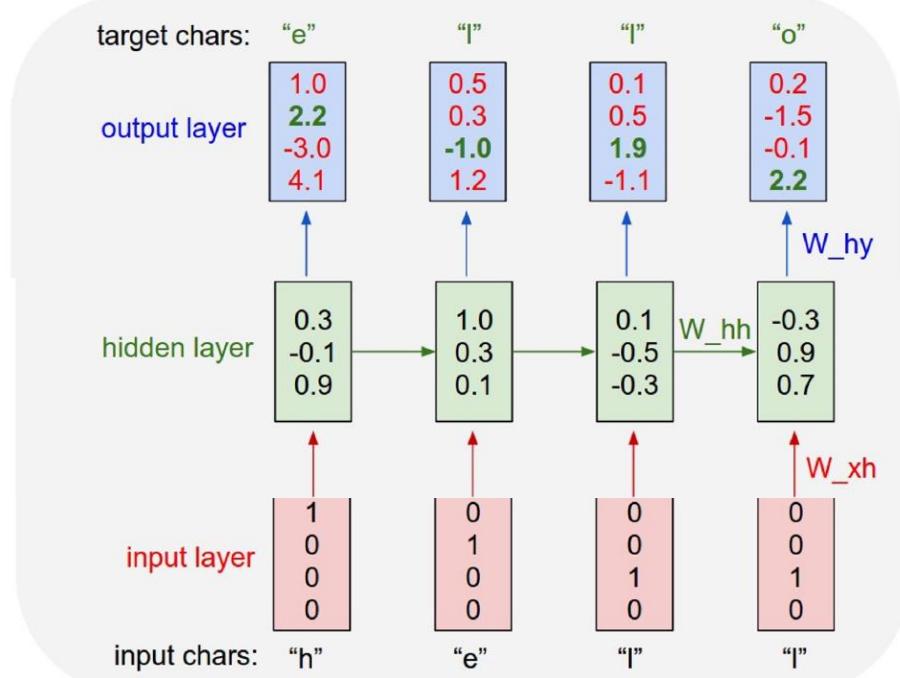
# EXAMPLE: CHARACTER-LEVEL LANGUAGE MODEL

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

$$y_t = W_{hy}h_t$$

Vocabulary: [h,e,l,o]

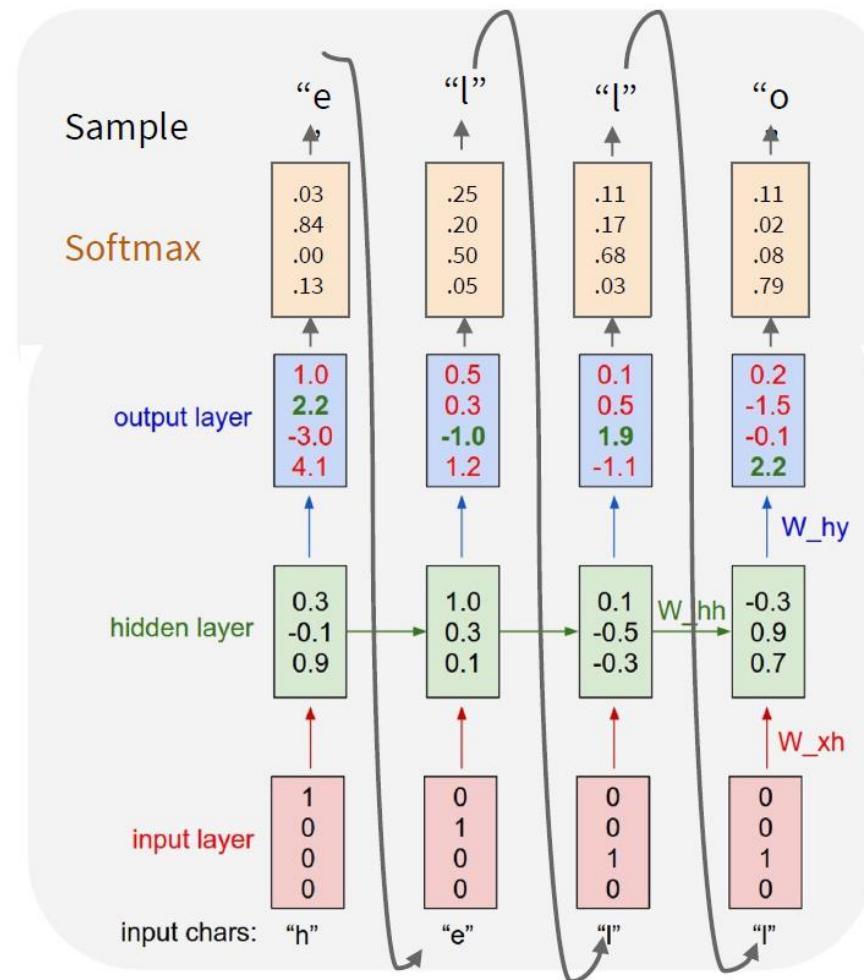
Example training sequence: “hello”



# EXAMPLE: CHARACTER-LEVEL LANGUAGE MODEL SAMPLING

Vocabulary: [h,e,l,o]

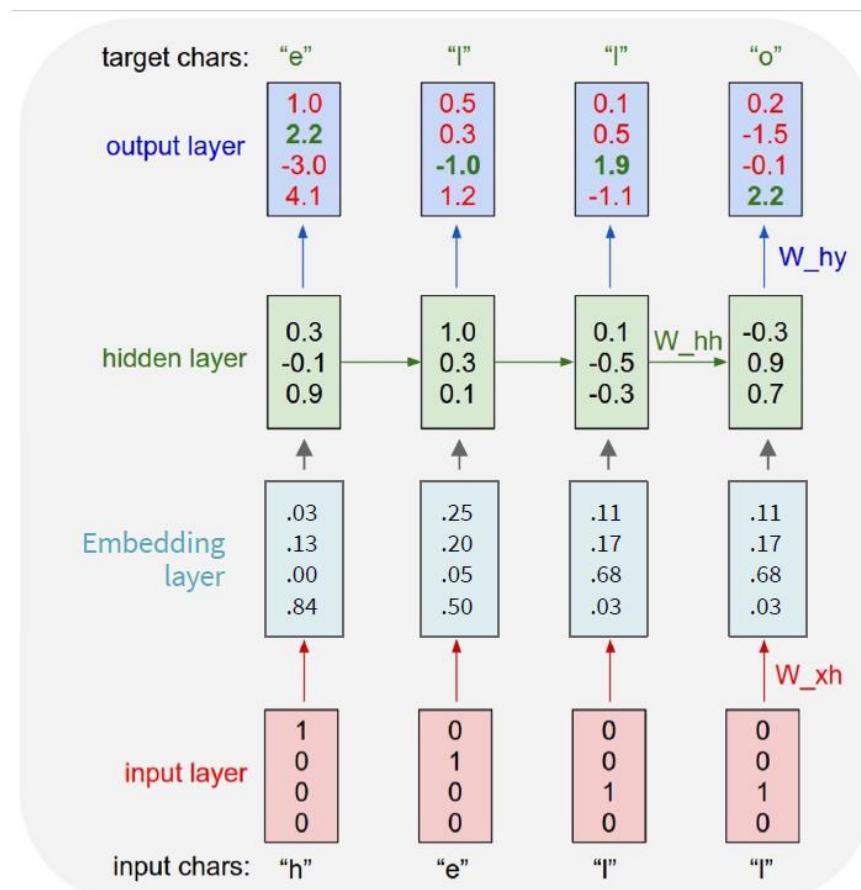
At **test-time** sample characters one at a time, feed back to model



# EXAMPLE: CHARACTER-LEVEL LANGUAGE MODEL SAMPLING

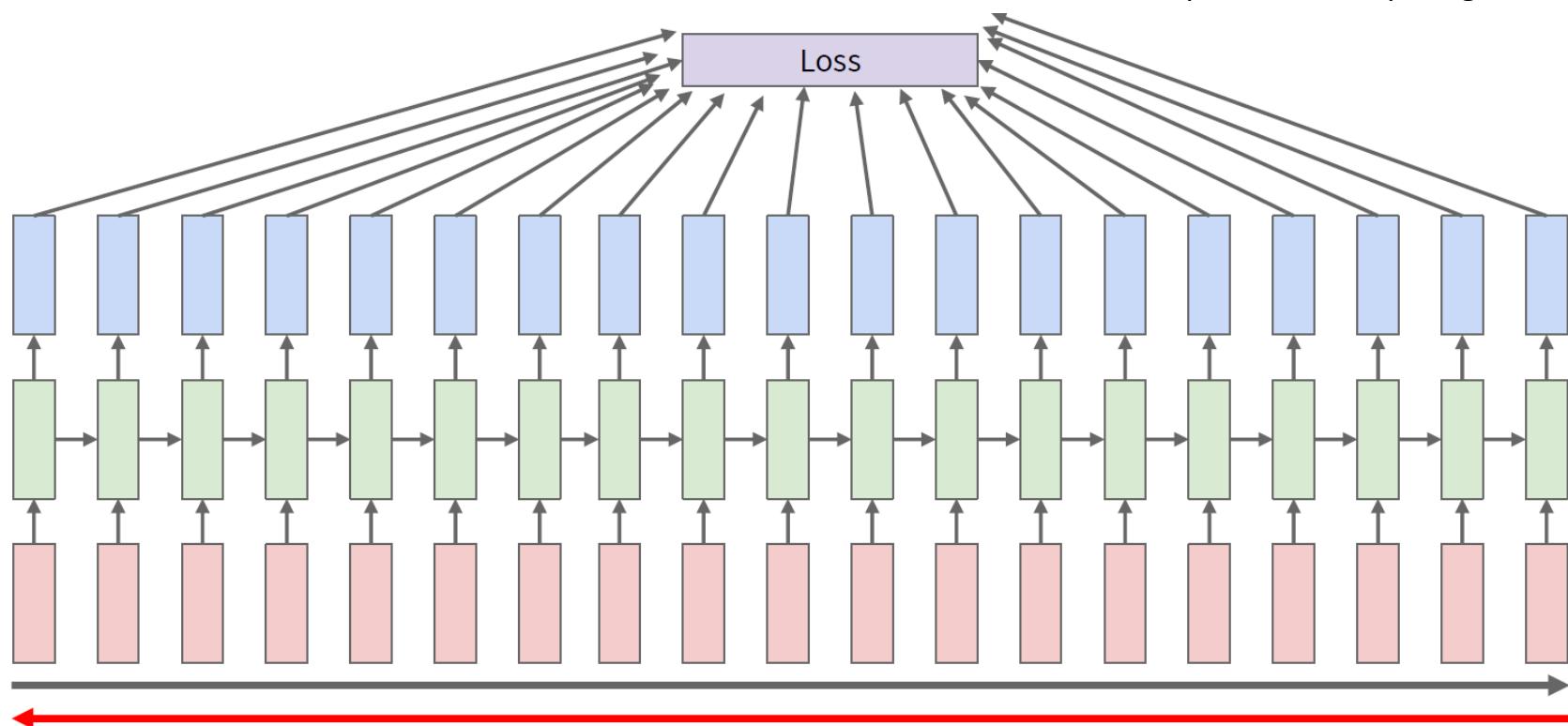
$$\begin{aligned}
 [W_{11} W_{12} W_{13} W_{14}] [1] &= [w_{11}] \\
 [W_{21} W_{22} W_{23} W_{14}] [0] &= [w_{21}] \\
 [W_{31} W_{32} W_{33} W_{14}] [0] &= [w_{31}] \\
 [W_{41} W_{42} W_{43} W_{44}] [0] &= [w_{41}]
 \end{aligned}$$

- ❖ Matrix multiplication with a one-hot vector just extracts a column from the weight matrix
- ❖ We often put a separate embedding layer between the input and hidden layers.



# BACKPROPAGATION THROUGH TIME

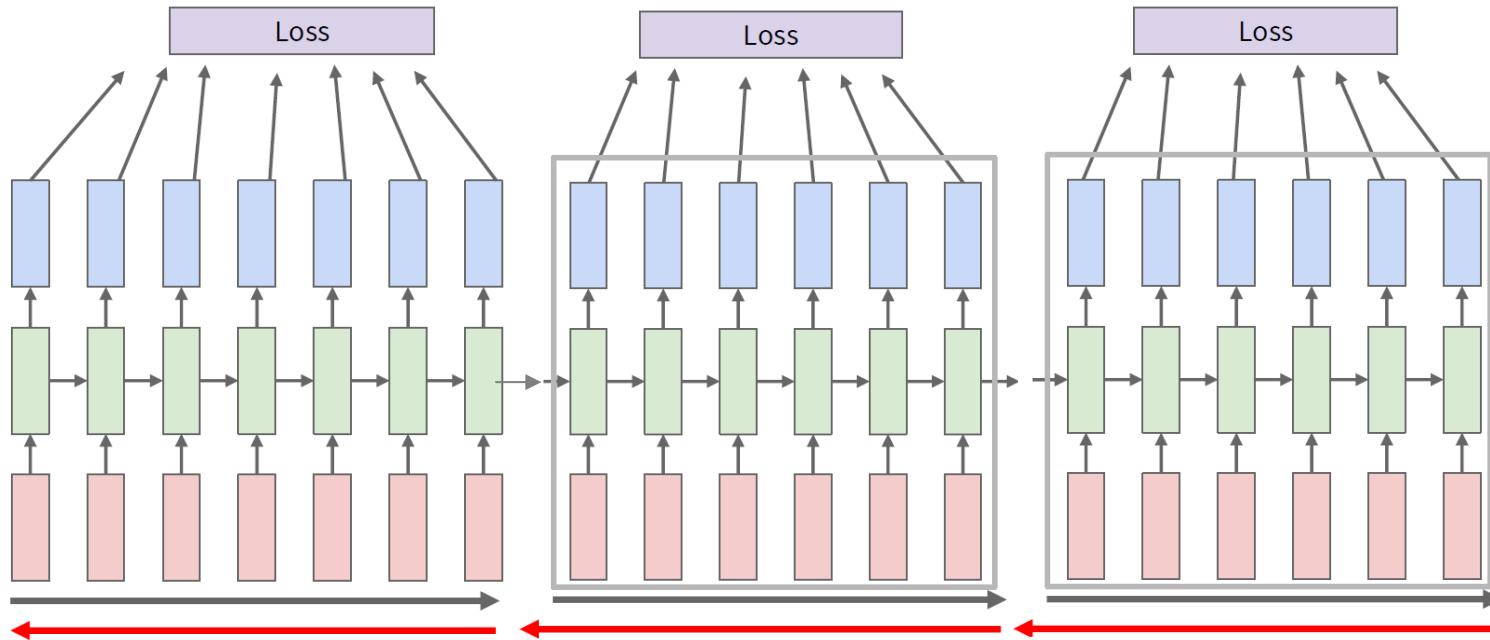
Forward through entire sequence to compute loss, then backward through entire sequence to compute gradient



# TRUNCATED BACKPROPAGATION THROUGH TIME

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

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



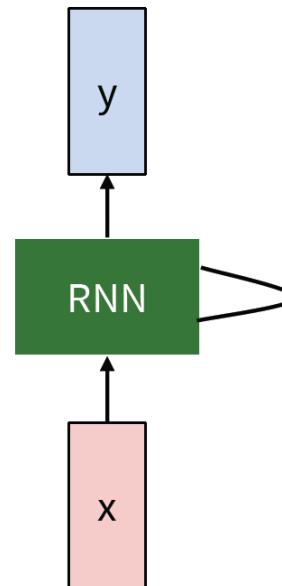
# THE SONNETS

by William Shakespeare

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



When forty winters shall besiege thy brow,  
And dig deep trenches in thy beauty's field,  
Thy youth's proud livery so gazed on now,  
Will be a tatter'd weed of small worth held:  
Then being asked, where all thy beauty lies,  
Where all the treasure of thy lusty days;  
To say, within thine own deep sunken eyes,  
Were an all-eating shame, and thriftless praise.  
How much more praise deserved 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.



# RNN EXAMPLE: MIN-CHAR-RNN.PY

at first:

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

↓ train more

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

↓ train more

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

↓ train more

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

# RNN EXAMPLE: MIN-CHAR-RNN.PY

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 nues 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 aps, 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 bar with links: home, about, tags explained, tag lookup, browse, search, bibliography, recent comments, blog, and add slogans. Below the navigation bar, there is a section titled "Browse chapters". This section contains a table with two columns: "Part" and "Chapter". The "Part" column lists "Preliminaries", and the "Chapter" column lists numbered topics from 1 to 10. For each chapter, there are three links: "online", "TeX source", and "view pdf". To the right of the table, there is a sidebar titled "Parts" which lists eight categories: Preliminaries, Schemes, Topics in Scheme Theory, Algebraic Spaces, Topics in Geometry, Deformation Theory, Algebraic Stacks, and Miscellany. Below the sidebar, there is a section titled "Statistics" which provides information about the project's size: 455910 lines of code, 14221 tags (56 inactive tags), and 2366 sections.

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

Latex source

<http://stacks.math.columbia.edu/>  
The stacks project is licensed under the [GNU Free Documentation License](#)

For  $\bigoplus_{n=1,\dots,m} \mathcal{L}_{m_n} = 0$ , 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  $\text{Sch}_{fppf}$  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

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

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

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

and

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

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  $X_{\text{spaces},\text{étale}}$  which gives an open subspace of  $X$  and  $T$  equal to  $S_{\text{Zar}}$ , 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  $\underline{\text{Proj}}_X(\mathcal{A}) = \text{Spec}(B)$  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  $i_0 \leq p$  is a subset of  $\mathcal{J}_{n,0} \circ \overline{A}_2$  works.

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

*Proof.* We will use the property we see that  $p$  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.  $\square$

**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  $X_{\text{étale}}$  we have

$$\mathcal{O}_X(\mathcal{F}) = \{\text{morph}_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}$$

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

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

*Proof.* See Spaces, Lemma ???.  $\square$

**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.  $\square$

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

$$\begin{array}{ccccc}
 S & \xrightarrow{\quad} & & & \\
 \downarrow & & & & \\
 \xi & \xrightarrow{\quad} & \mathcal{O}_{X'} & \xleftarrow{\quad} & \\
 \text{gor}_s & & \uparrow & & \\
 & & =\alpha' \xrightarrow{\quad} & & \\
 & & \downarrow & & \\
 & & =\alpha' \xrightarrow{\quad} \alpha & & \\
 & & & & \\
 \text{Spec}(K_\psi) & & \text{Mor}_{\text{Sets}} & & d(\mathcal{O}_{X/k}, \mathcal{G}) \\
 & & & & \downarrow \\
 & & & & X
 \end{array}$$

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  $U$ .  $\square$

*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”

$$\mathcal{O}_{X,x} \longrightarrow \mathcal{F}_{\bar{x}} \dashrightarrow (\mathcal{O}_{X_{\text{étale}}}) \longrightarrow \mathcal{O}_{X_\ell}^{-1} \mathcal{O}_X(\mathcal{O}_{X_n}^{\text{ur}})$$

is an isomorphism of covering of  $\mathcal{O}_{X_\ell}$ . 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  $S$ .

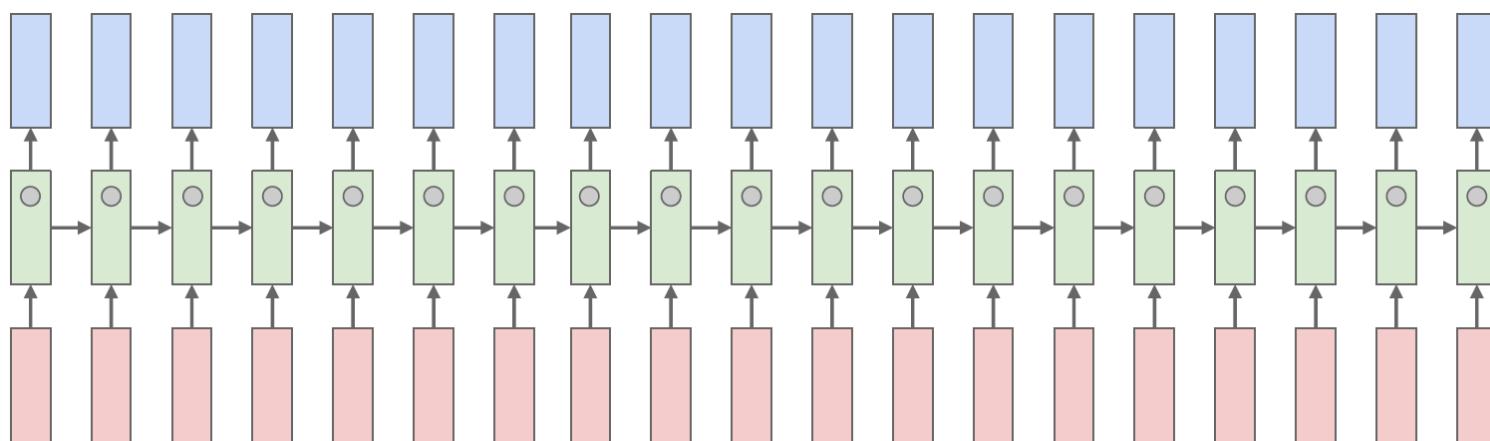
If  $\mathcal{F}$  is a scheme theoretic image points.  $\square$

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

# GENERATED C CODE

```
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 & 0x00000000fffffff8) & 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 ");
}
```

# SEARCHING FOR INTERPRETABLE CELLS



# 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.
    */
```

# 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 quote detection cell

# 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

# 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

# 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

# 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

# RNN TRADEOFFS

## ➤ RNN Advantages:

- Can process any length of the input
- Computation for step  $t$  can (in theory) use information from many steps back
- Model size does not increase for longer input
- The same weights are applied on every timestep, so there is symmetry in how inputs are processed.

## ➤ RNN Disadvantages:

- Recurrent computation is slow
- In practice, difficult to access information from many steps back

# IMAGE CAPTIONING

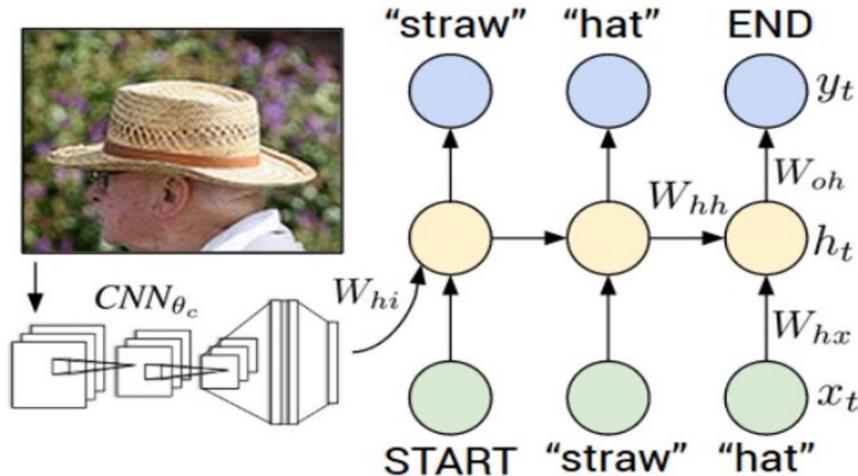


Figure from Karpathy et al, "Deep Visual-Semantic Alignments for Generating Image Descriptions", CVPR 2015; figure copyright IEEE, 2015.

## Some other papers:

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

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

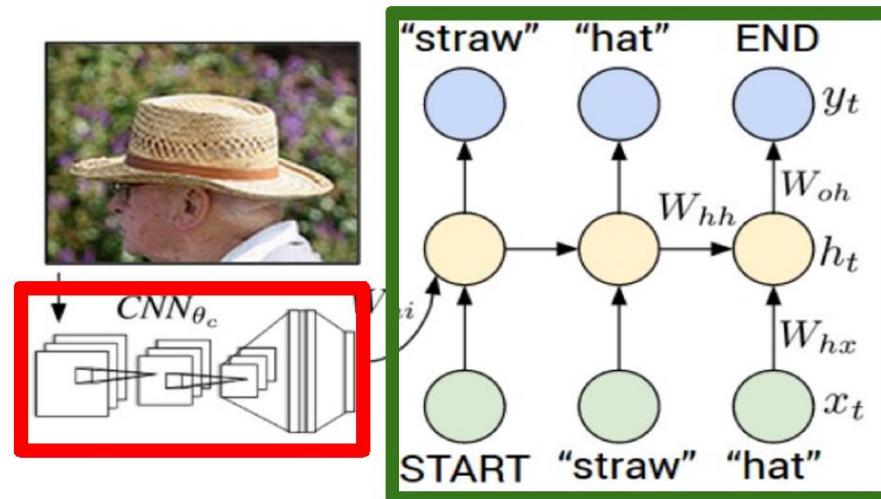
Show and Tell: A Neural Image Caption Generator, Vinyals et al.

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

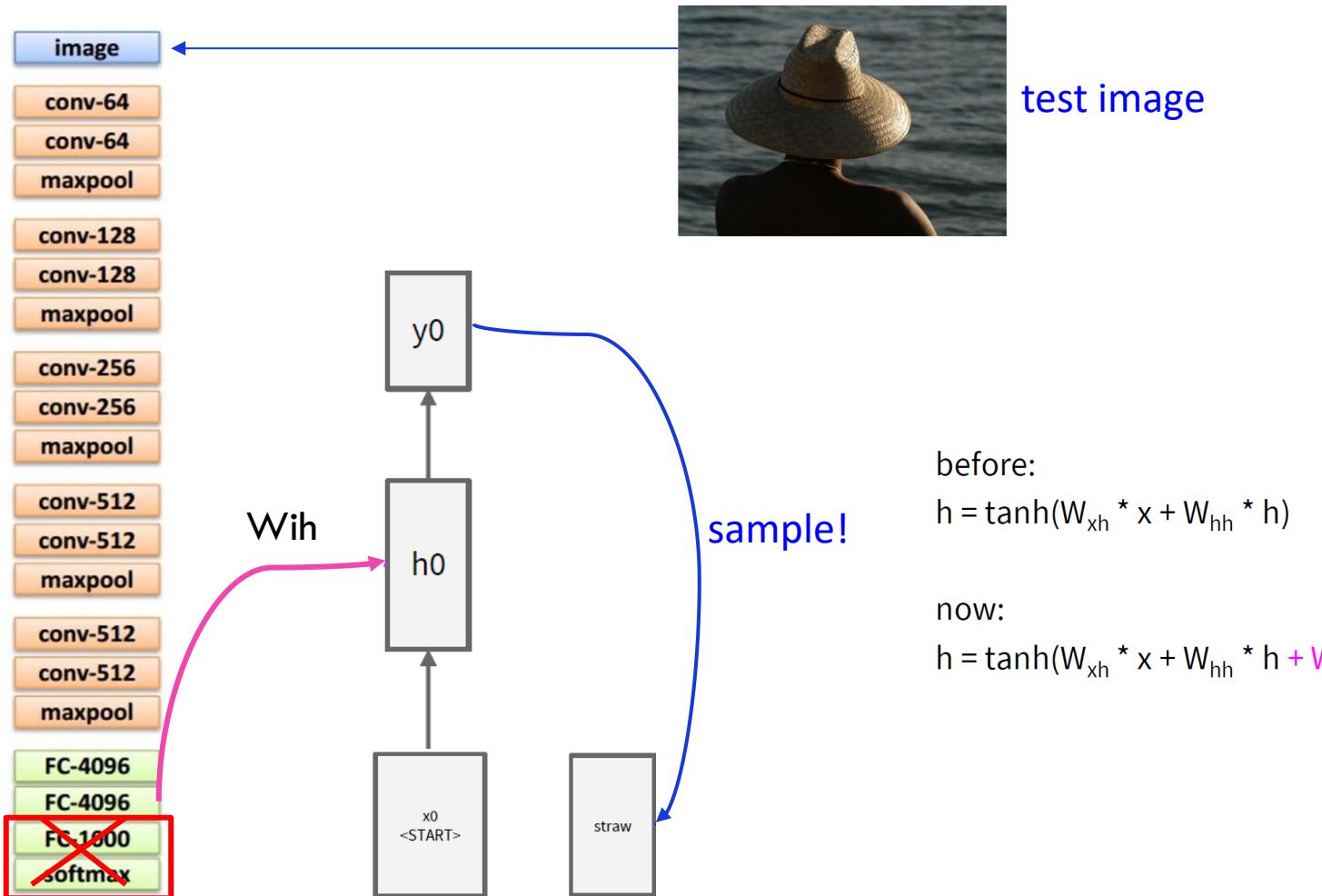
# IMAGE CAPTIONING

## Recurrent Neural Network

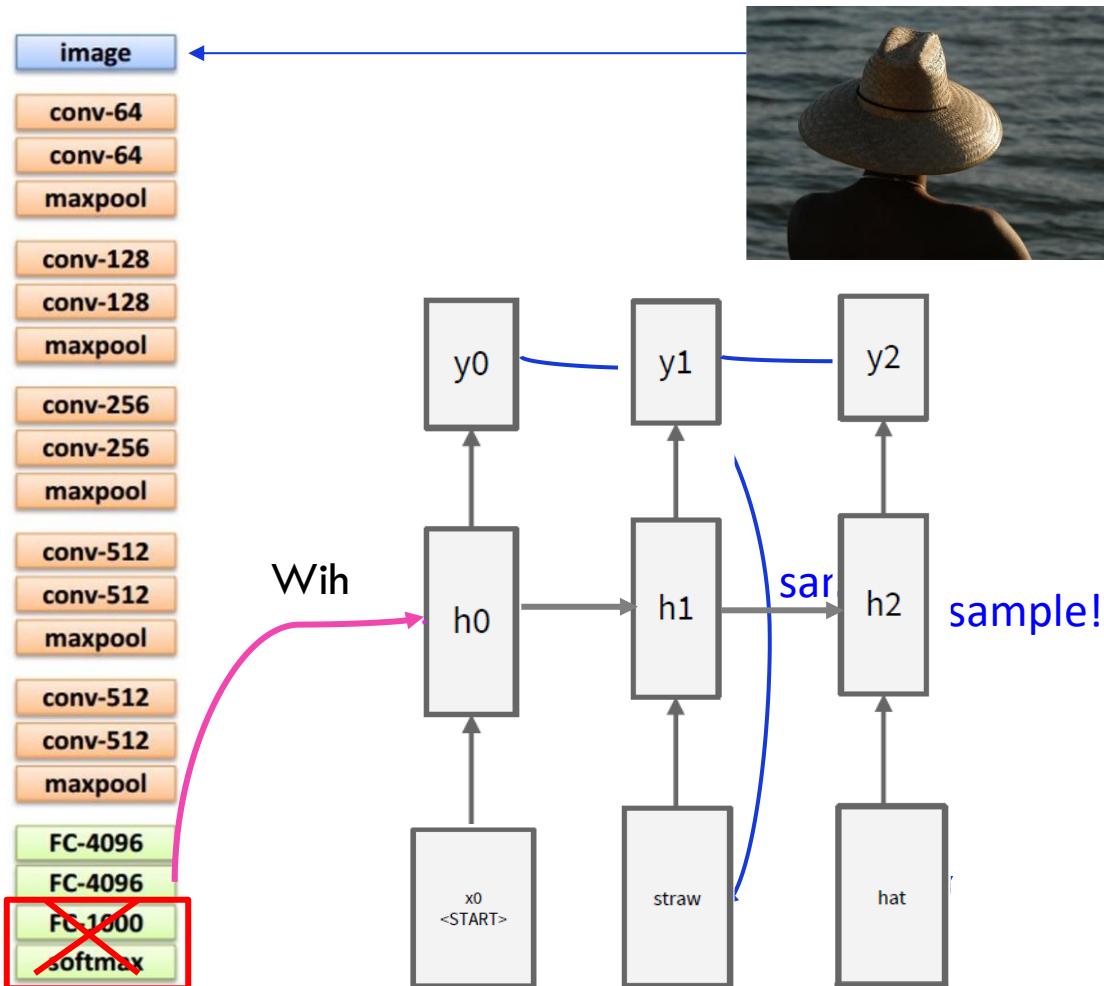


Convolutional Neural Network

# IMAGE CAPTIONING



# IMAGE CAPTIONING



test image

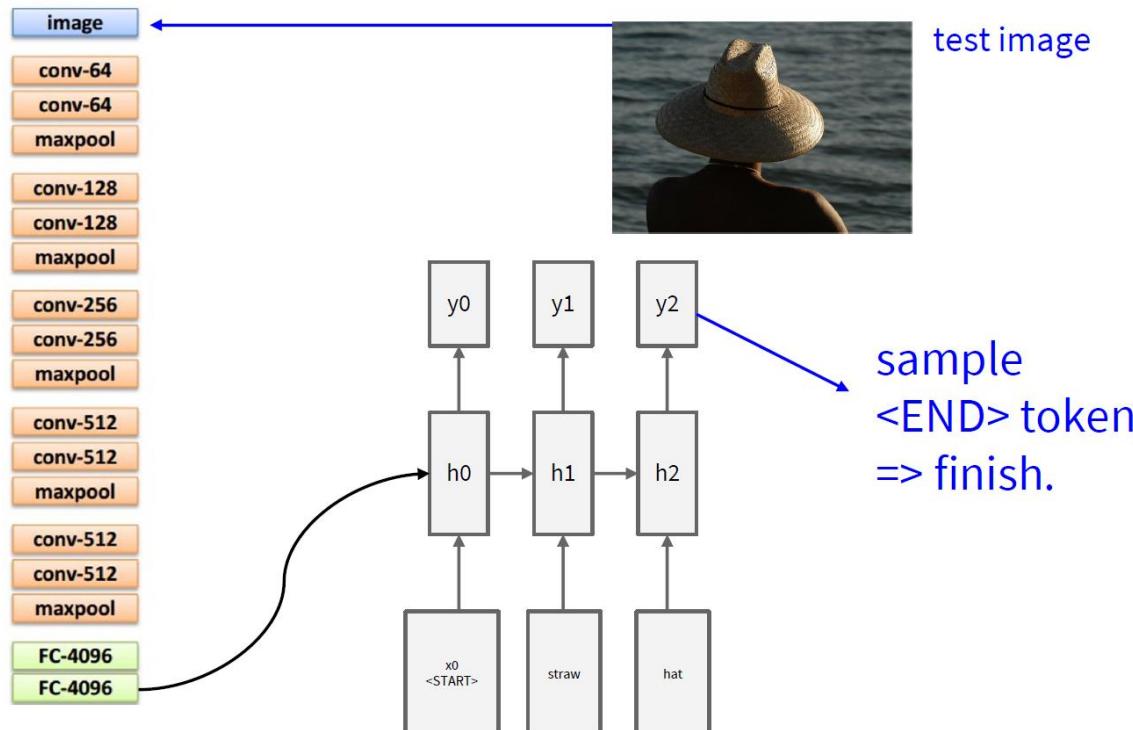
before:

$$h = \tanh(W_{xh} * x + W_{hh} * h)$$

now:

$$h = \tanh(W_{xh} * x + W_{hh} * h + W_{ih} * v)$$

# IMAGE CAPTIONING



# IMAGE CAPTIONING: EXAMPLE RESULTS



A cat sitting on a suitcase on the floor



A cat is sitting on a tree branch



A dog is running in the grass with a frisbee



A white teddy bear sitting in the grass



Two people walking on the beach with surfboards



A tennis player in action on the court



Two giraffes standing in a grassy field



A man riding a dirt bike on a dirt track

# IMAGE CAPTIONING: FAILURE CASES



A woman standing on a beach holding a surfboard

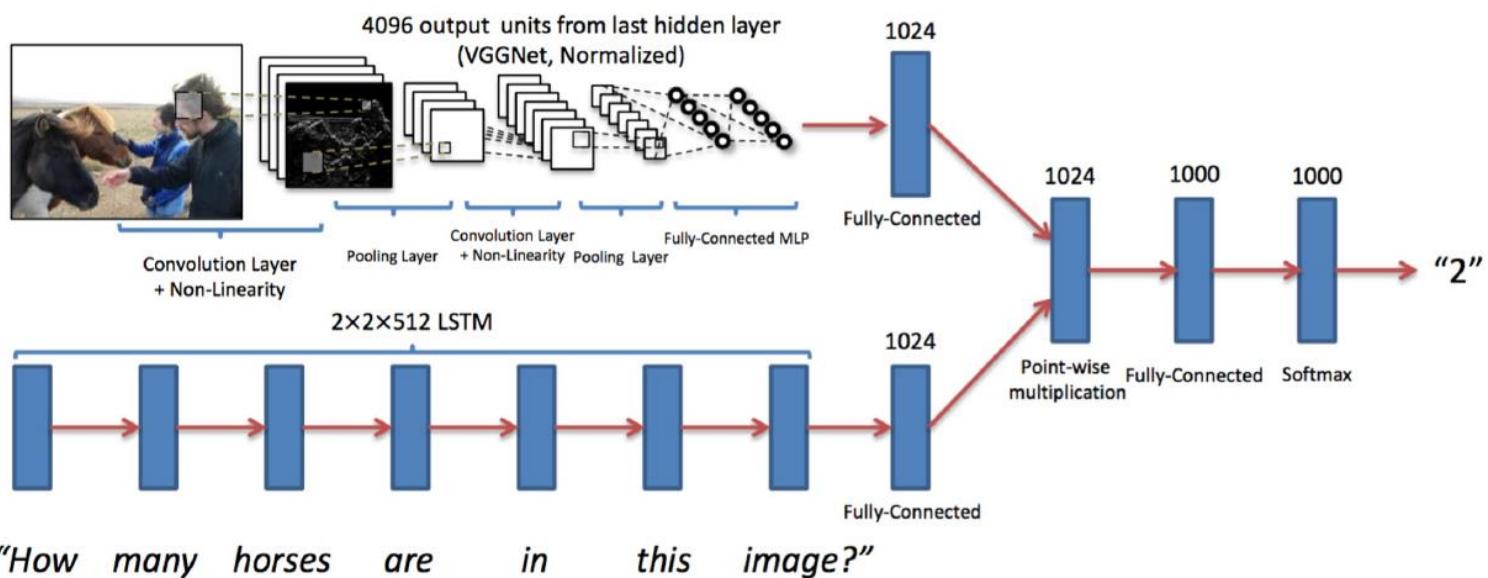


A bird is perched on a tree branch



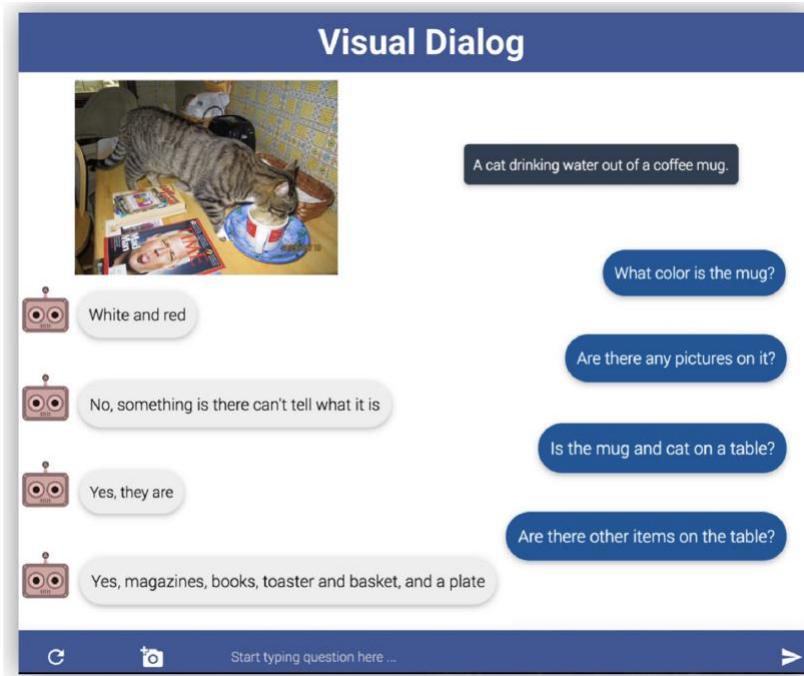
A man in a baseball uniform throwing a ball

# VISUAL QUESTION ANSWERING (VQA)



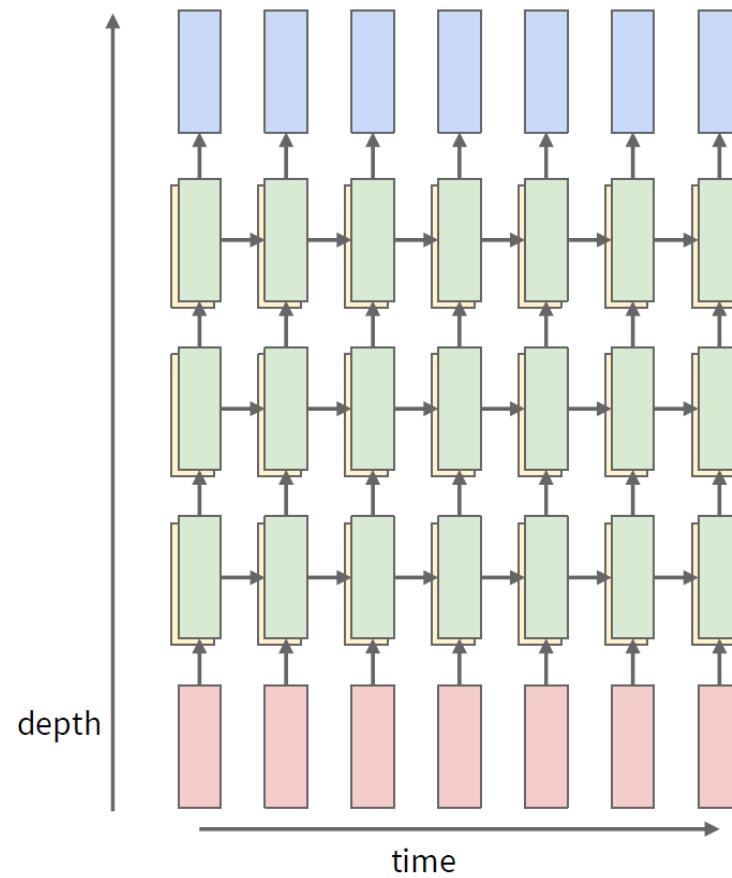
Agrawal et al, "Visual 7W: Grounded Question Answering in Images", CVPR 2015  
Figures from Agrawal et al, copyright IEEE 2015. Reproduced for educational purposes.

# VISUAL DIALOG: CONVERSATIONS ABOUT IMAGES



Das et al, "Visual Dialog", CVPR 2017  
Figures from Das et al, copyright IEEE 2017.

# MULTILAYER RNNs



# LONG SHORT TERM MEMORY (LSTM)

Vanilla RNN

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

LSTM

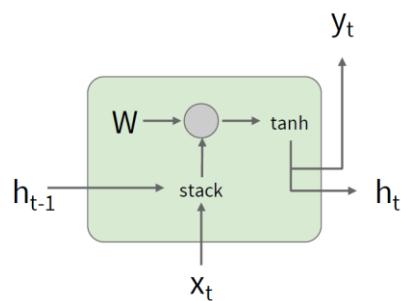
$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

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

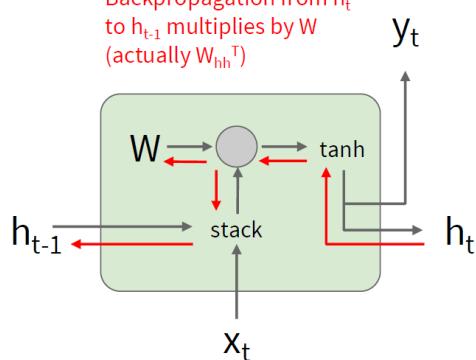
$$h_t = o \odot \tanh(c_t)$$

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

# VANILLA RNN GRADIENT FLOW



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

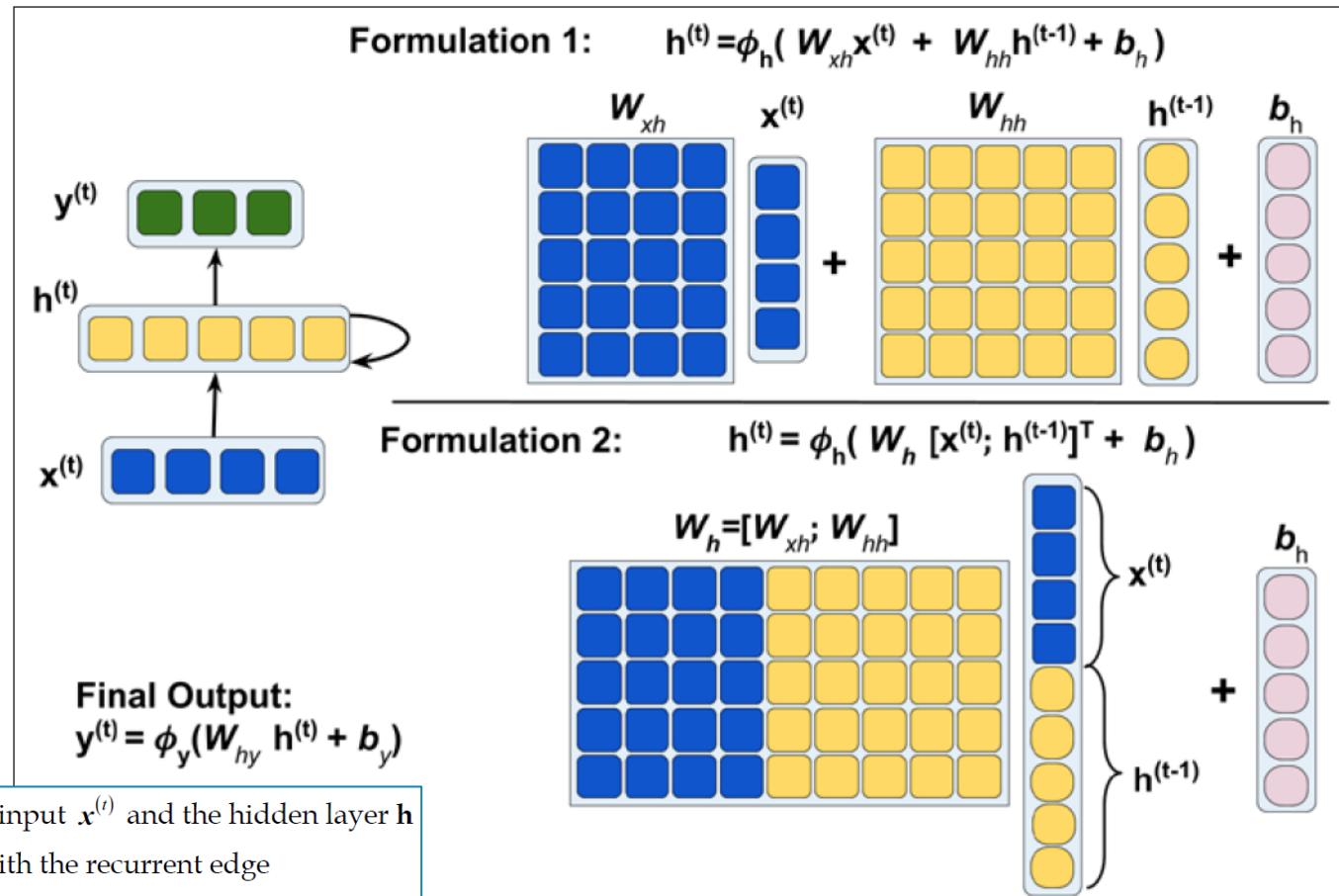


$$\frac{\partial h_t}{\partial h_{t-1}} = \tanh'(W_{hh}h_{t-1} + W_{xh}x_t)W_{hh}$$

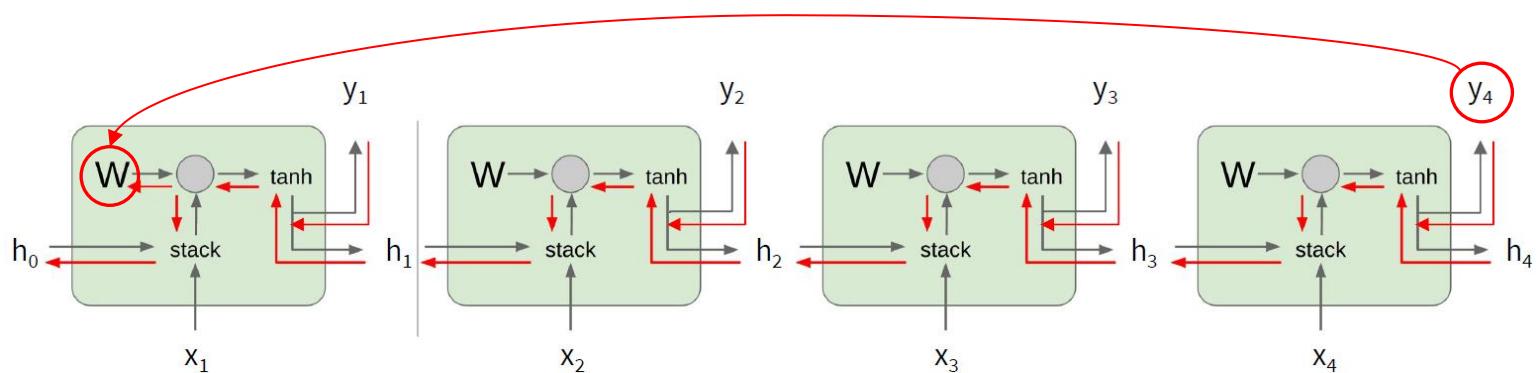
Bengio et al, "Learning long-term dependencies with gradient descent is difficult", IEEE Transactions on Neural Networks, 1994  
Pascanu et al, "On the difficulty of training recurrent neural networks", ICML 2013

# VANILLA RNN GRADIENT FLOW

➤ Computing activations in an RNN



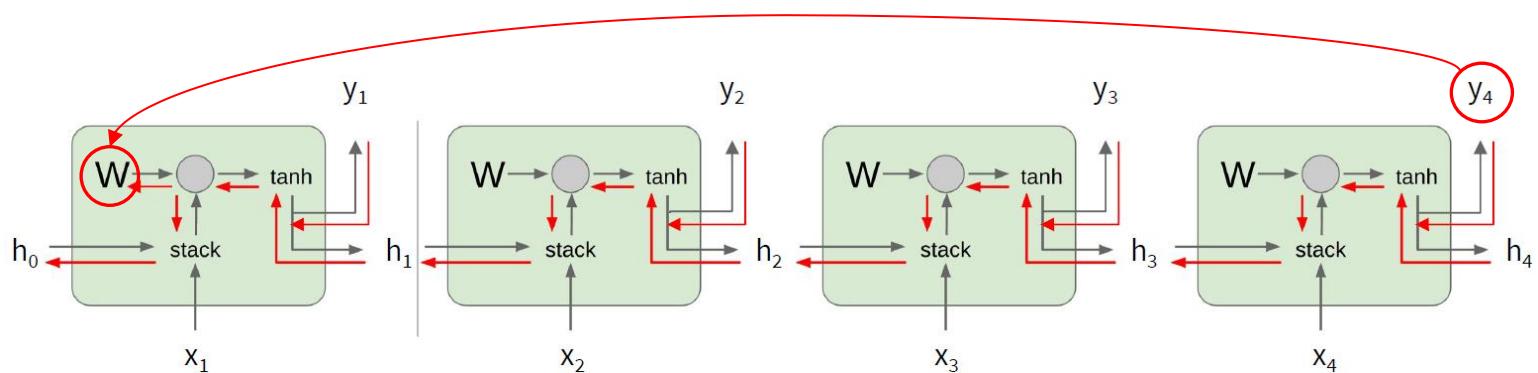
# VANILLA RNN GRADIENT FLOW



$$\frac{\partial L}{\partial W} = \sum_{t=1}^T \frac{\partial L_t}{\partial W} \quad \boxed{\frac{\partial h_t}{\partial h_{t-1}} = \tanh'(W_{hh}h_{t-1} + W_{xh}x_t)W_{hh}}$$

$$\frac{\partial L_T}{\partial W} = \frac{\partial L_T}{\partial h_T} \frac{\partial h_t}{\partial h_{t-1}} \cdots \frac{\partial h_1}{\partial W} = \frac{\partial L_T}{\partial h_T} \left( \prod_{t=2}^T \frac{\partial h_t}{\partial h_{t-1}} \right) \frac{\partial h_1}{\partial W}$$

# VANILLA RNN GRADIENT FLOW

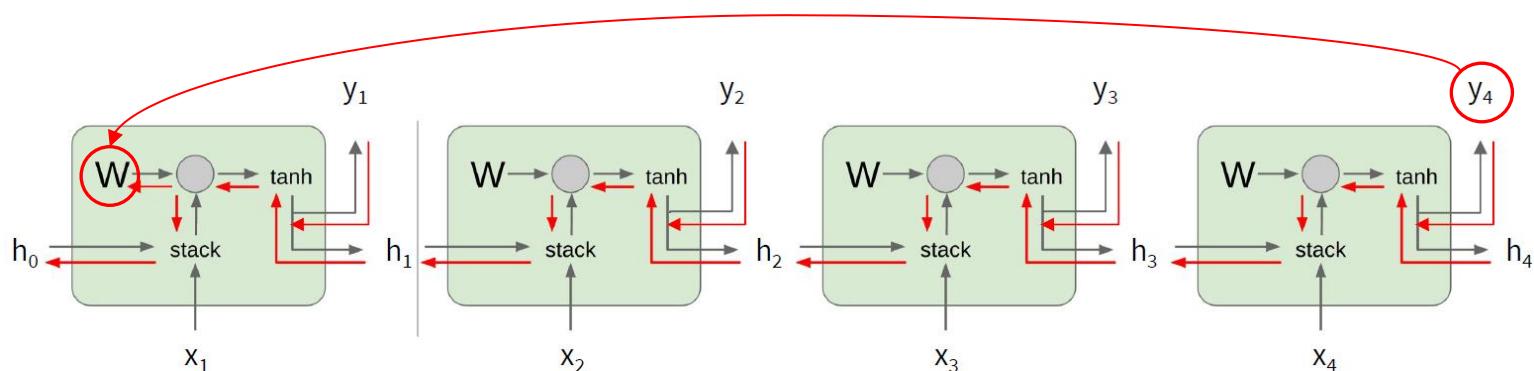


$$\frac{\partial L}{\partial W} = \sum_{t=1}^T \frac{\partial L_t}{\partial W}$$

Almost always  $< 1$   
Vanishing gradients

$$\frac{\partial L_T}{\partial W} = \frac{\partial L_T}{\partial h_T} \left( \prod_{t=2}^T \boxed{\tanh'(W_{hh}h_{t-1} + W_{xh}x_t)} \right) W_{hh}^{T-1} \frac{\partial h_1}{\partial W}$$

# VANILLA RNN GRADIENT FLOW



What if we assumed no non-linearity?

$$\frac{\partial L}{\partial W} = \sum_{t=1}^T \frac{\partial L_t}{\partial W}$$

Largest singular value > 1:  
Exploding gradients

Gradient clipping: Scale gradient  
if its norm is too big

$$\frac{\partial L_T}{\partial W} = \frac{\partial L_T}{\partial h_T} \boxed{W_{hh}^{T-1}} \frac{\partial h_1}{\partial W}$$

Largest singular value < 1:  
Vanishing gradients

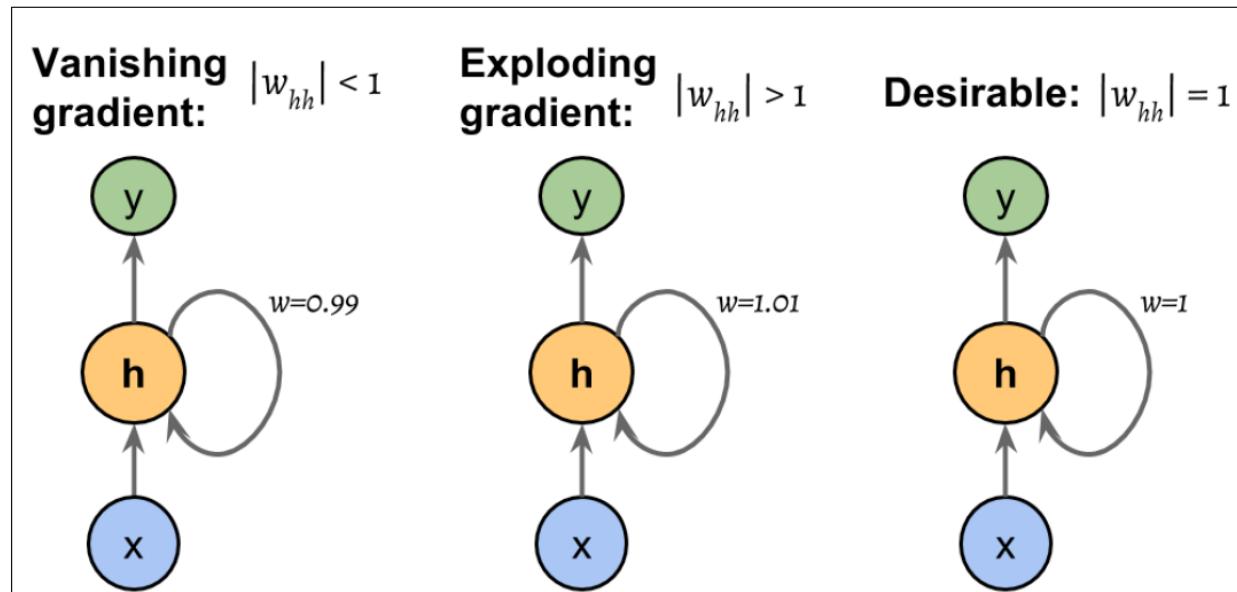
```
grad_norm = np.sum(grad * grad)
if grad_norm > threshold:
    grad *= (threshold / grad_norm)
```

Change RNN  
architecture

# RECURRENT NEURAL NETWORKS (RNNs)

- The challenges of learning long-range interactions

Because of the multiplicative factor  $\frac{\partial h^{(t)}}{\partial h^{(k)}}$  in the computing gradients of a loss function, the so-called **vanishing** or **exploding gradient** problem arises. This problem is explained through the examples in the following figure, which shows an RNN with only one hidden unit for simplicity:



# RECURRENT NEURAL NETWORKS (RNNs)

## ➤ The challenges of learning long-range interactions

Basically,  $\frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(k)}}$  has  $t - k$  multiplications; therefore, multiplying the  $w$  weight  $t - k$  times results in a factor  $-w^{t-k}$ . As a result, if  $|w| < 1$ , this factor becomes very small when  $t - k$  is large. On the other hand, if the weight of the recurrent edge is  $|w| > 1$ , then  $w^{t-k}$  becomes very large when  $t - k$  is large. Note that large  $t - k$  refers to long-range dependencies.

- In practice, there are two solutions to this problem:
  - Truncated backpropagation through time (TBPTT)
  - Long short-term memory (LSTM)
- TBPTT clips the gradients above a given threshold. While TBPTT can solve the exploding gradient problem, the truncation limits the number of steps that the gradient can effectively flow back and properly update the weights.
- On the other hand, LSTM, designed in 1997 by Hochreiter and Schmidhuber, has been more successful in modeling long-range sequences by overcoming the vanishing gradient problem.

# RECURRENT NEURAL NETWORKS (RNNs)

## ➤ Long Short-Term Memory (LSTM) units

- LSTMs were first introduced to overcome the vanishing gradient problem (Long Short-Term Memory, S. Hochreiter and J. Schmidhuber, Neural Computation, 9(8): 1735-1780, 1997). The building block of an LSTM is a memory cell, which essentially represents the hidden layer.
- In each memory cell, there is a **recurrent edge** that has the desirable weight  $w = 1$ , as we discussed previously, to overcome the vanishing and exploding gradient problems. The values associated with this recurrent edge is called cell state. The unfolded structure of a modern LSTM cell is shown in the following figure:

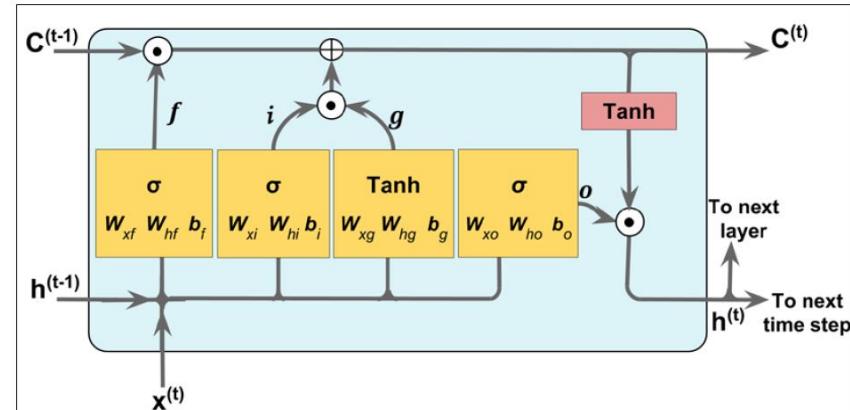
✓ Notice that the cell state from the previous time step,  $C^{(t-1)}$ , is modified to get the cell state at the current time step,  $C^{(t)}$ , without being multiplied directly with any weight factor.

⊕: element-wise product

⊕: element-wise summation

$x^{(t)}$ : input data at time t

$h^{(t-1)}$ : hidden units at time t - 1



# LONG SHORT TERM MEMORY (LSTM)

Vanilla RNN

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

LSTM

Four gates

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

Cell state

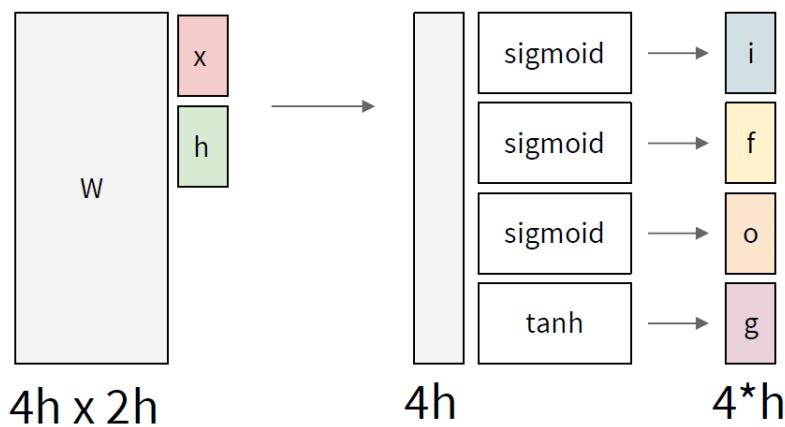
$$c_t = f \odot c_{t-1} + i \odot g$$

Hidden state

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

# LONG SHORT TERM MEMORY (LSTM)

[HOCHREITER ET AL., 1997]



i: Input gate, Whether to write to cell

f: Forget gate, Whether to erase cell

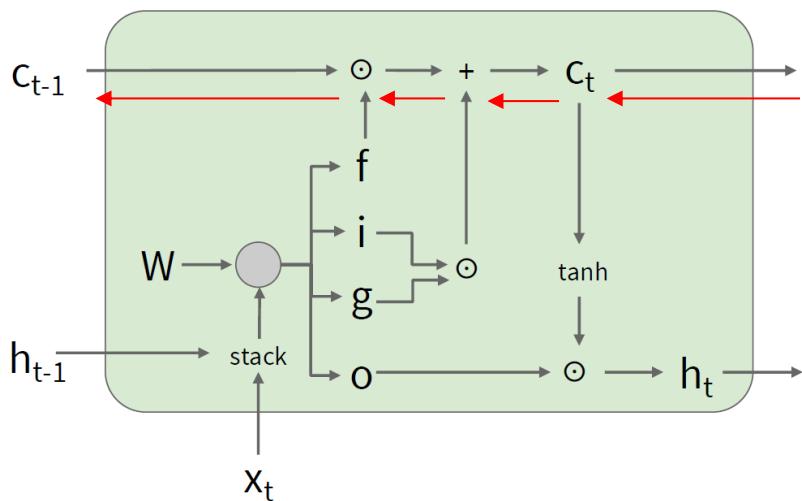
o: Output gate, How much to reveal cell

g: Gate gate (?), How much to write to cell

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

# LONG SHORT TERM MEMORY (LSTM)

[HOCHREITER ET AL., 1997]



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

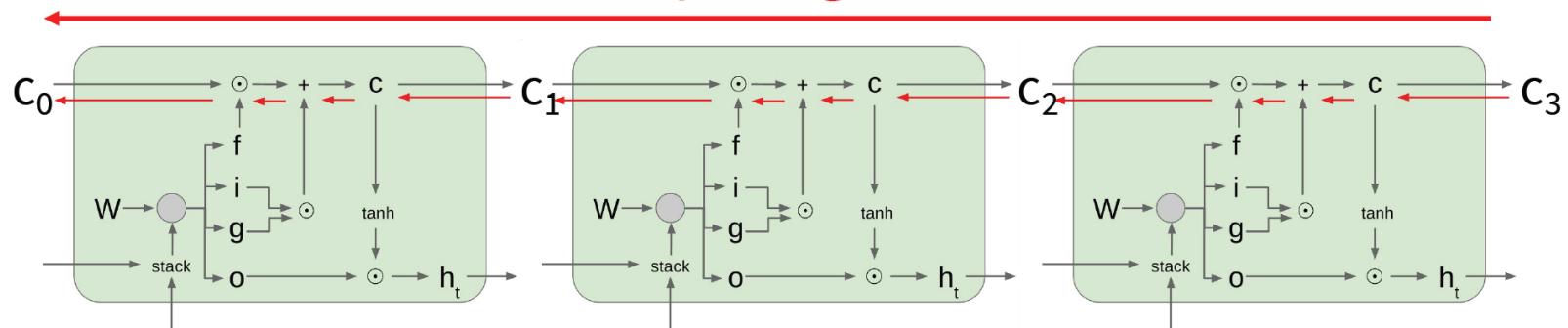
---

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

# LONG SHORT TERM MEMORY (LSTM)

[HOCHREITER ET AL., 1997]

Uninterrupted gradient flow!



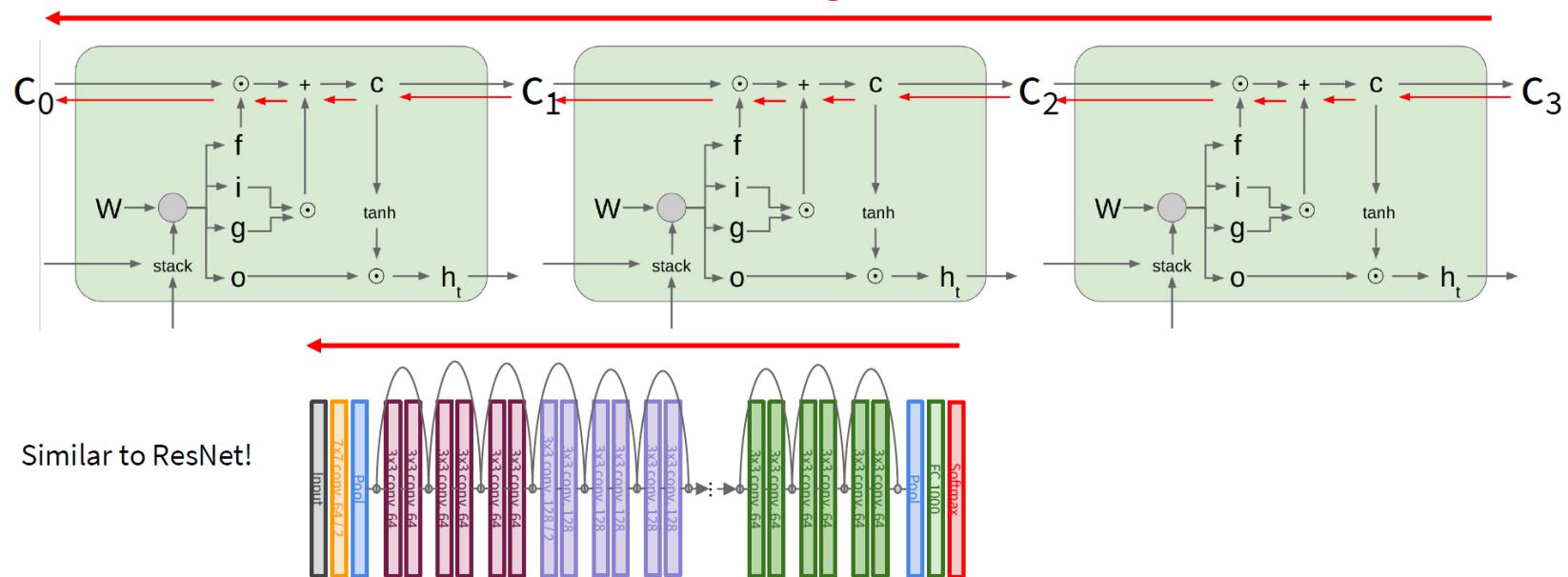
# DO LSTMS SOLVE THE VANISHING GRADIENT PROBLEM?

- The LSTM architecture makes it easier for the RNN to preserve information over many timesteps
  - e.g. if the  $f = 1$  and the  $i = 0$ , then the information of that cell is preserved indefinitely.
  - By contrast, it's harder for vanilla RNN to learn a recurrent weight matrix  $W_h$  that preserves info in hidden state
- LSTM doesn't guarantee that there is no vanishing/exploding gradient, but it does provide an easier way for the model to learn long-distance dependencies

# LONG SHORT TERM MEMORY (LSTM): GRADIENT FLOW

[HOCHREITER ET AL., 1997]

Uninterrupted gradient flow!



# OTHER RNN VARIANTS

GRU [Learning phrase representations using rnn encoderdecoder for statistical machine translation, Cho et al. 2014]

$$r_t = \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r)$$

$$z_t = \sigma(W_{xz}x_t + W_{hz}h_{t-1} + b_z)$$

$$\tilde{h}_t = \tanh(W_{xh}x_t + W_{hh}(r_t \odot h_{t-1}) + b_h)$$

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

[LSTM: A Search Space Odyssey, Greff et al., 2015]

Gated Recurrent Unit (GRU)

[An Empirical Exploration of Recurrent Network Architectures, Jozefowicz et al., 2015]

MUT1:

$$z = \text{sigm}(W_{xz}x_t + b_z)$$

$$r = \text{sigm}(W_{xr}x_t + W_{hr}h_t + b_r)$$

$$\begin{aligned} h_{t+1} = & \tanh(W_{hh}(r \odot h_t) + \tanh(x_t) + b_h) \odot z \\ & + h_t \odot (1 - z) \end{aligned}$$

MUT2:

$$z = \text{sigm}(W_{xz}x_t + W_{hz}h_t + b_z)$$

$$r = \text{sigm}(x_t + W_{hr}h_t + b_r)$$

$$\begin{aligned} h_{t+1} = & \tanh(W_{hh}(r \odot h_t) + W_{xh}x_t + b_h) \odot z \\ & + h_t \odot (1 - z) \end{aligned}$$

MUT3:

$$z = \text{sigm}(W_{xz}x_t + W_{hz}\tanh(h_t) + b_z)$$

$$r = \text{sigm}(W_{xr}x_t + W_{hr}h_t + b_r)$$

$$\begin{aligned} h_{t+1} = & \tanh(W_{hh}(r \odot h_t) + W_{xh}x_t + b_h) \odot z \\ & + h_t \odot (1 - z) \end{aligned}$$

# SUMMARY

- RNNs allow a lot of flexibility in architecture design
- Vanilla RNNs are simple but don't work very well
- Common to use LSTM or GRU: their additive interactions improve gradient flow
- Backward flow of gradients in RNN can explode or vanish. Exploding is controlled with gradient clipping. Vanishing is controlled with additive interactions (LSTM)
- Better/simpler architectures are a hot topic of current research, as well as new paradigms for reasoning over sequences
- Better understanding (both theoretical and empirical) is needed.