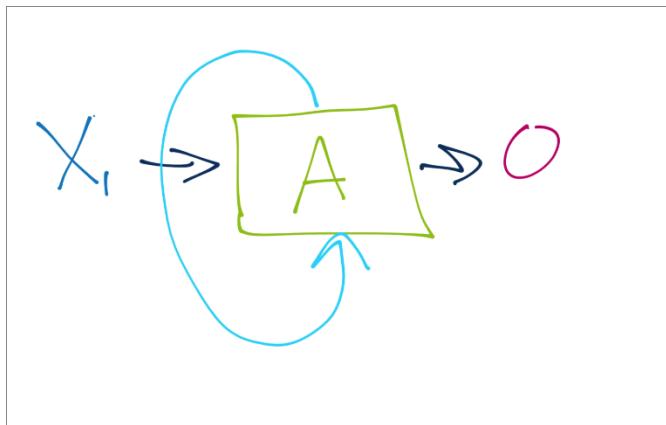


# Recurrent Neural Network

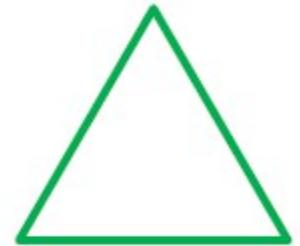


Fast Campus  
Start Deep Learning with Tensorflow

# Sequence Data

- Alphabet 을 z부터 a까지 거꾸로 외워봅시다
- 어려운 이유는?
- Traditional multilayer perceptron neural networks make the assumption that all inputs are independent of each other
- This assumption breaks down in the case of sequence data

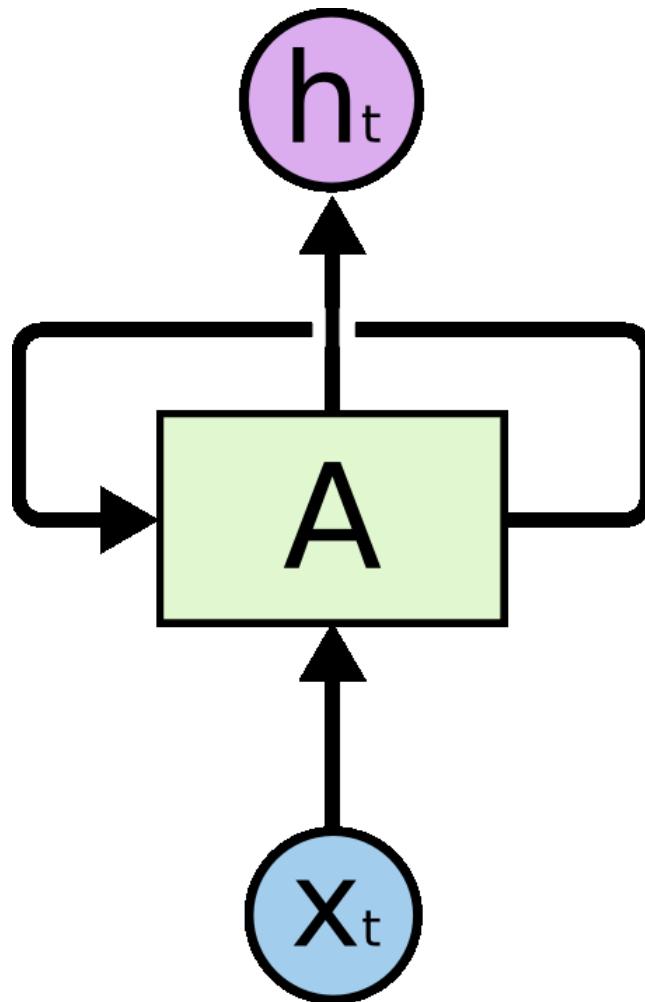
# Anagram



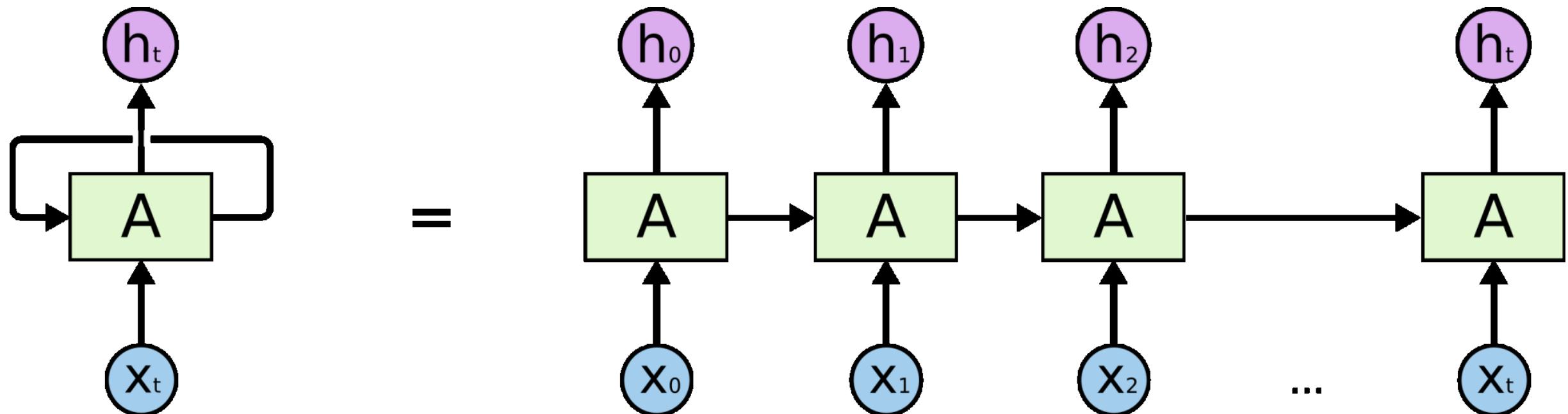
# Fill the Blank

- I am a student. Every morning I go to the \_\_\_\_.
- MLP로 풀 수 있을까요?
  - 모든 단어는 vector로 표현 가능하다고 가정해봅시다
  - 모든 단어를 모두 합쳐서(concatenation?) MLP에 입력으로 넣는 방법
  - 한 단어씩 MLP에 순차적으로 넣는 방법

# Recurrent Neural Network

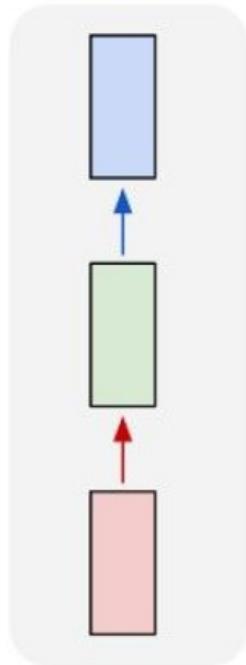


# Recurrent Neural Network

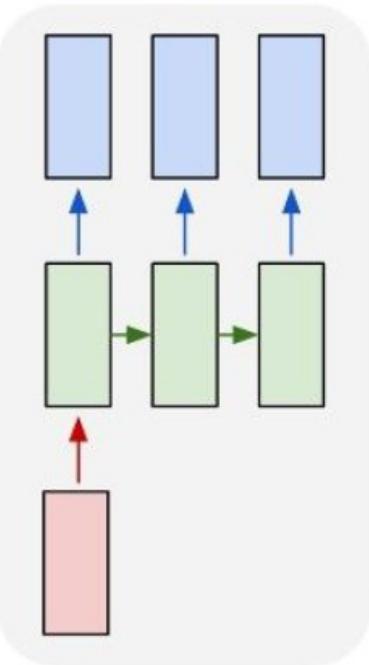


# RNN's flexibility

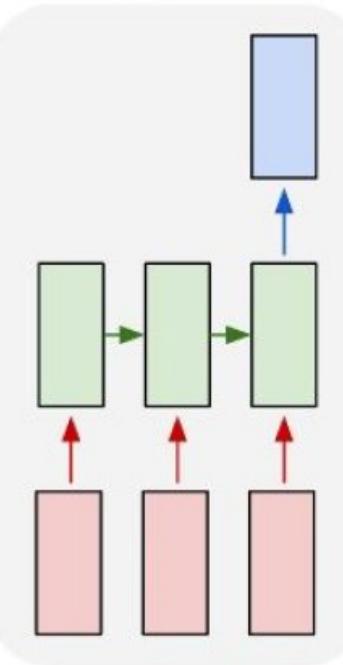
one to one



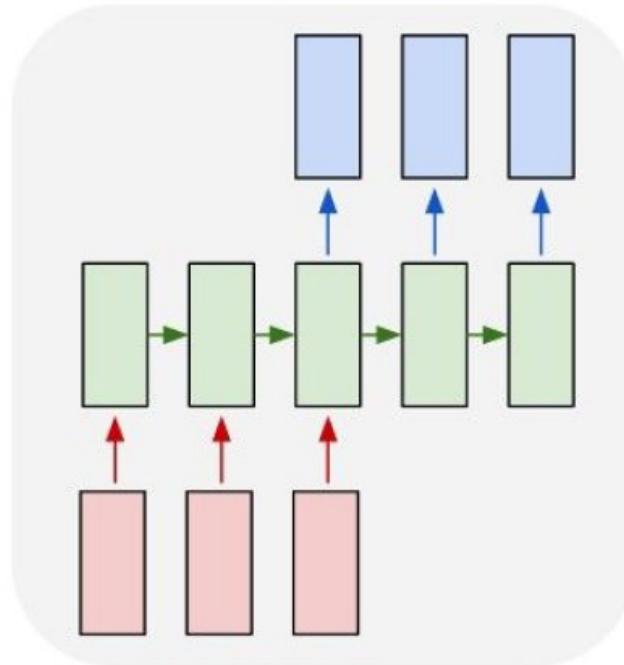
one to many



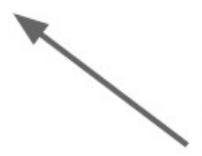
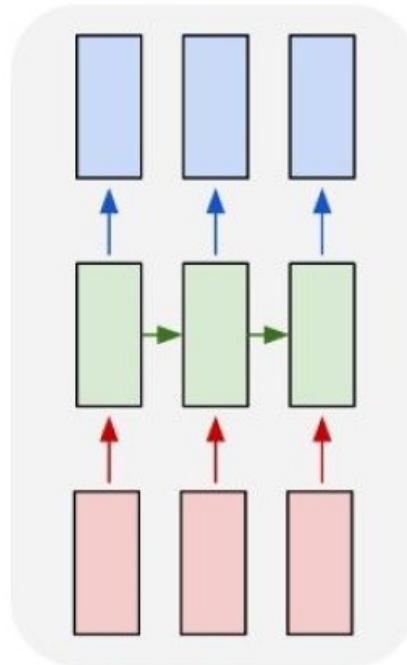
many to one



many to many



many to many



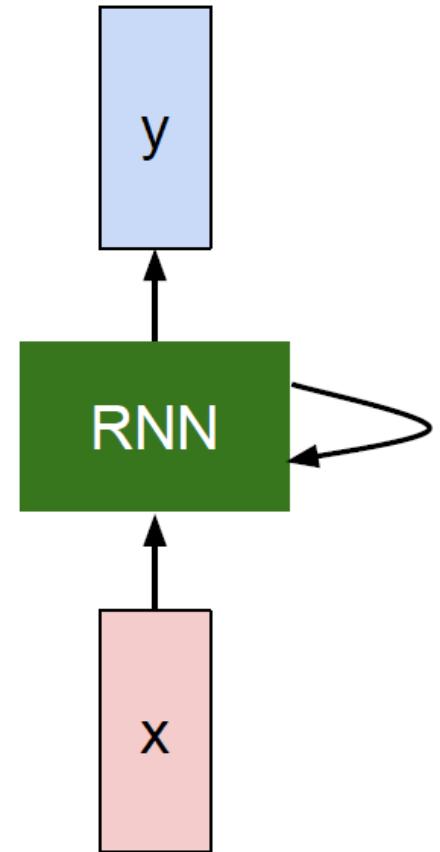
**Vanilla Neural Networks**

# Recurrent Neural Network

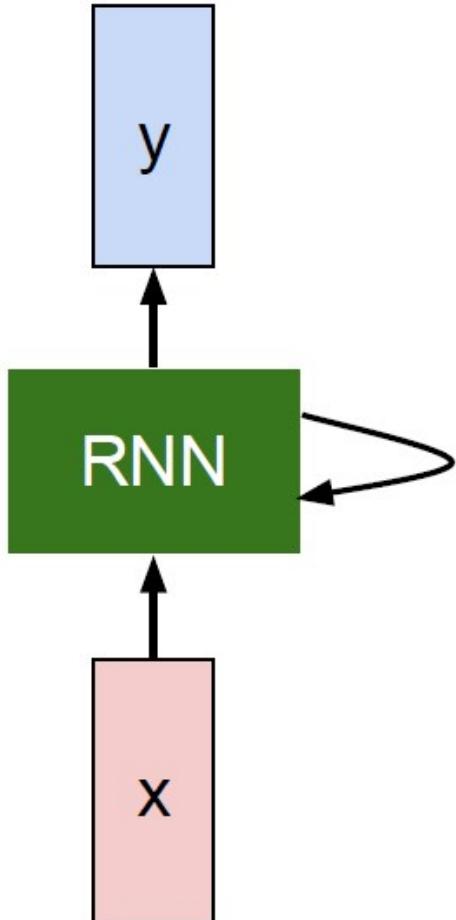
We can process a sequence of vectors  $\mathbf{x}$  by applying a recurrence formula at every time step:

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

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



# Recurrent Neural Network



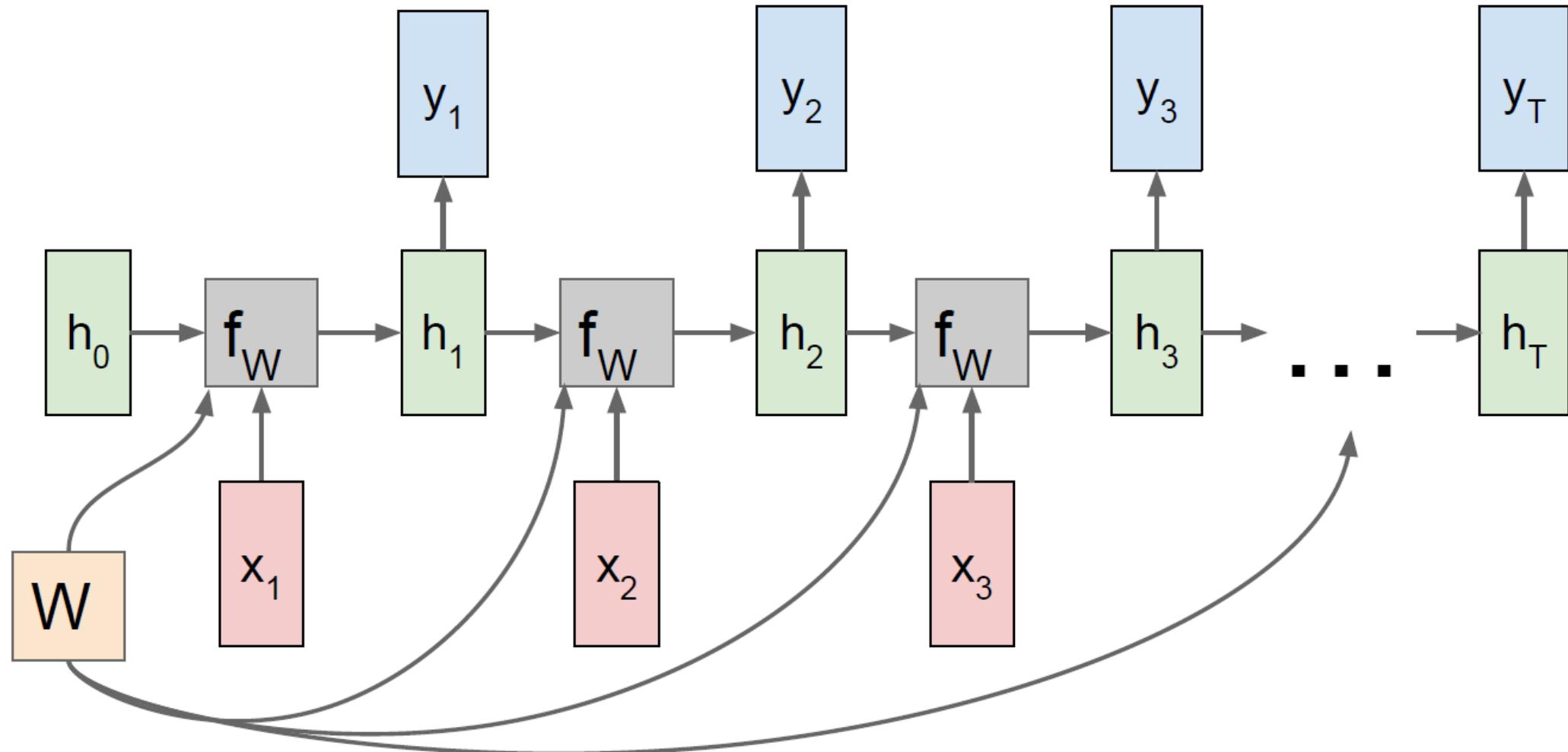
$$h_t = f_W(h_{t-1}, x_t)$$



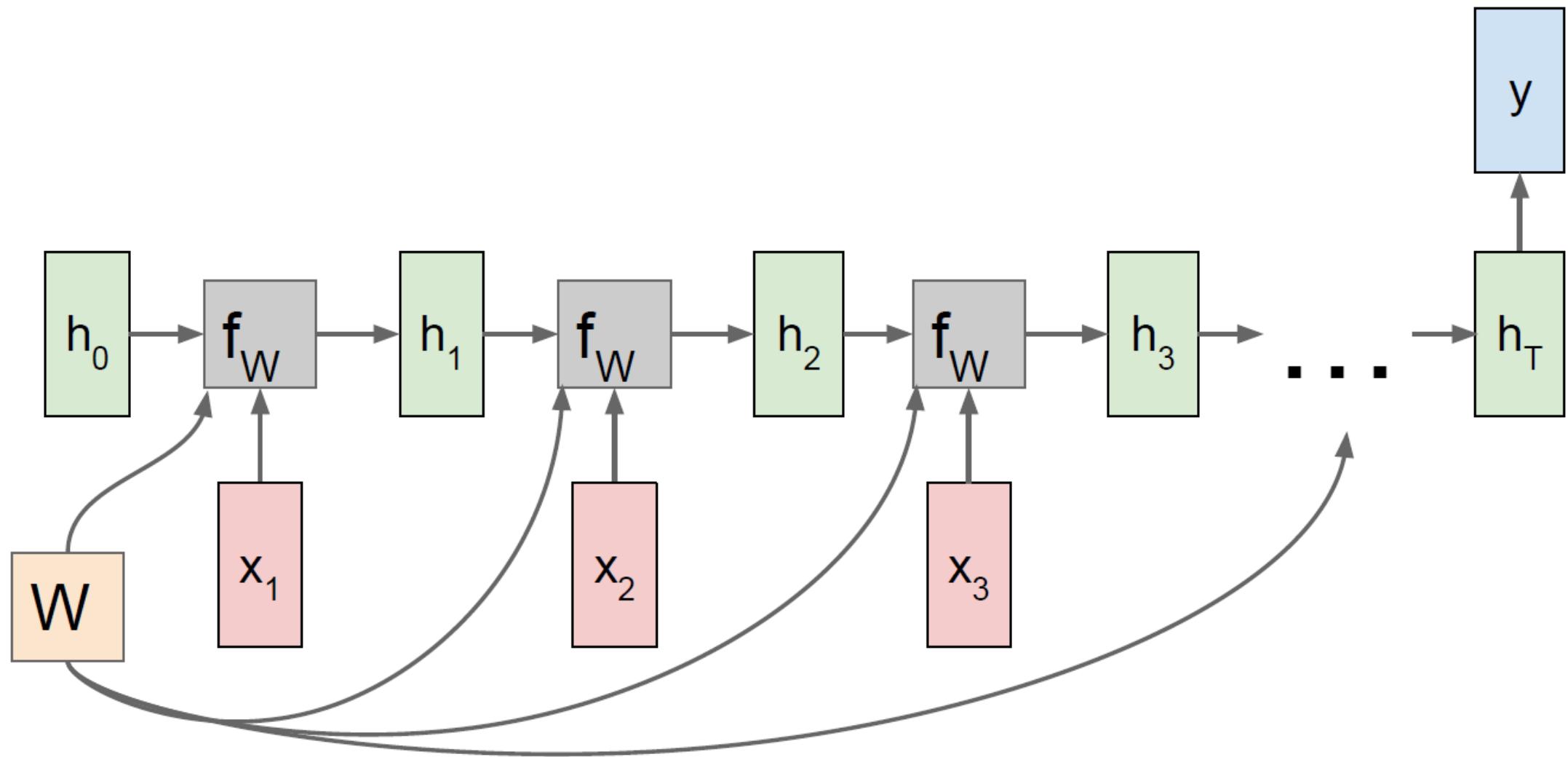
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t$$

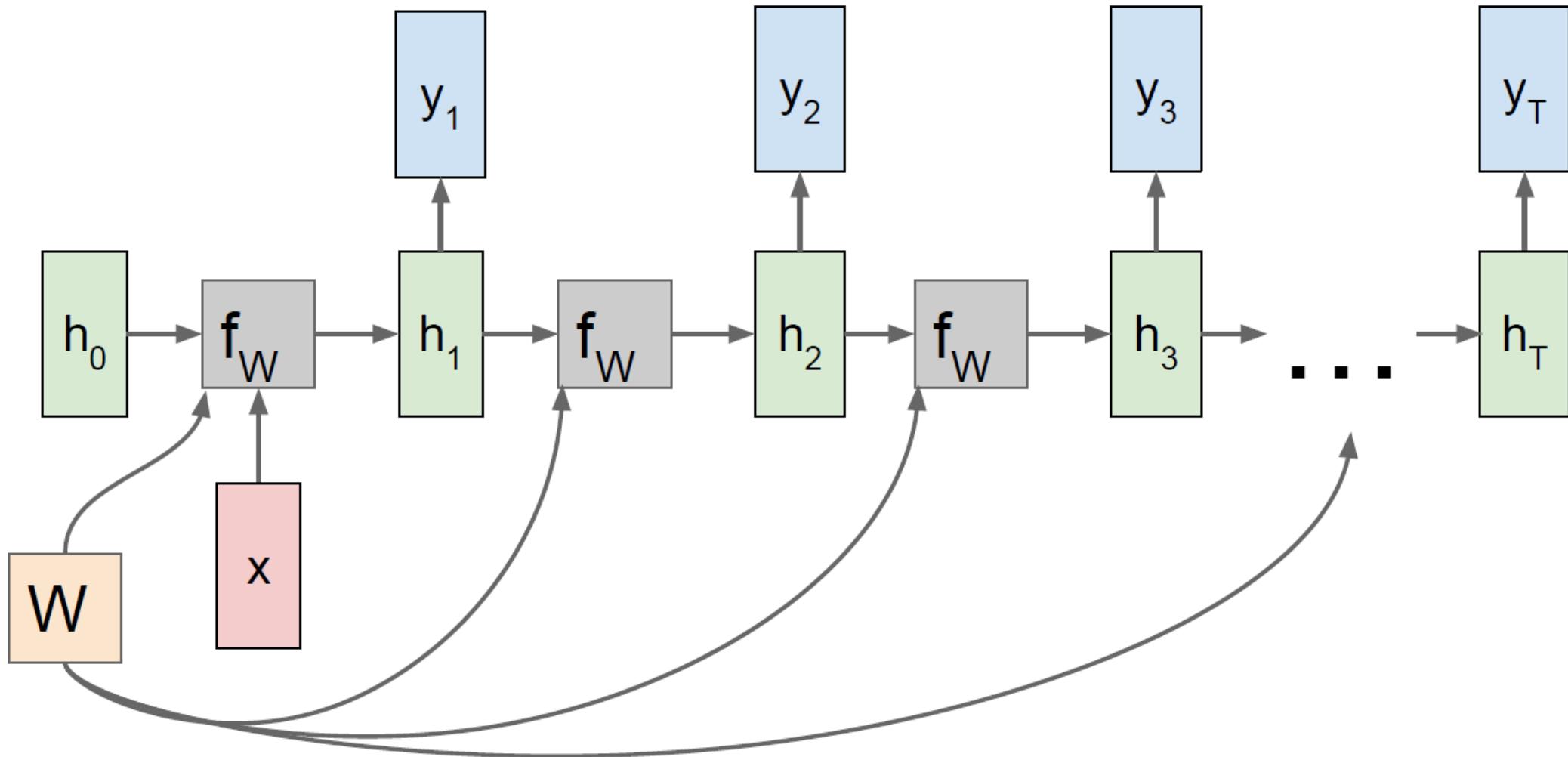
# RNN: Computational Graph – Many to Many



# RNN: Computational Graph – Many to One

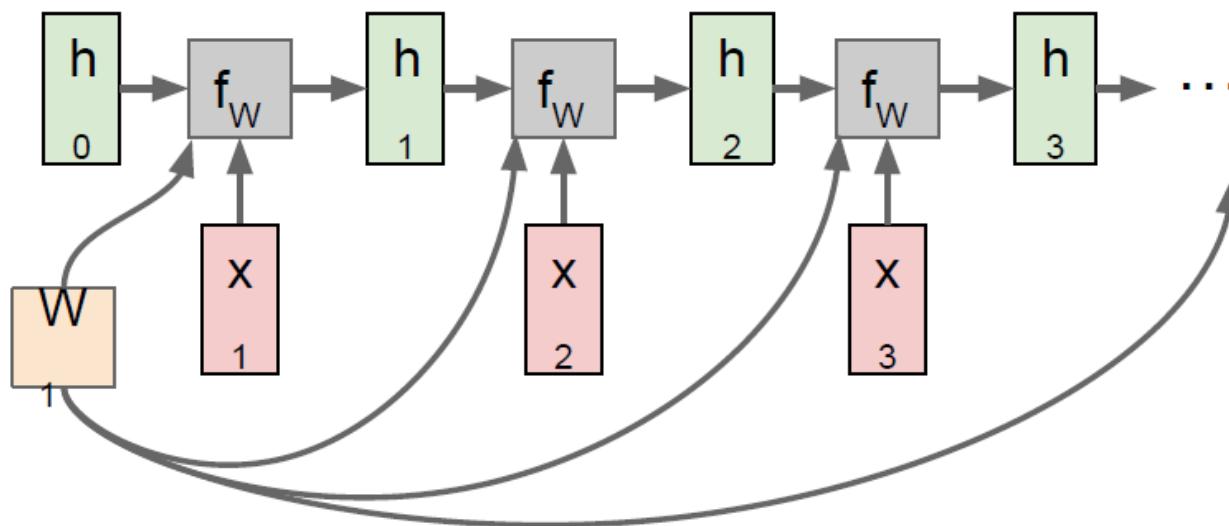


# RNN: Computational Graph – One to Many

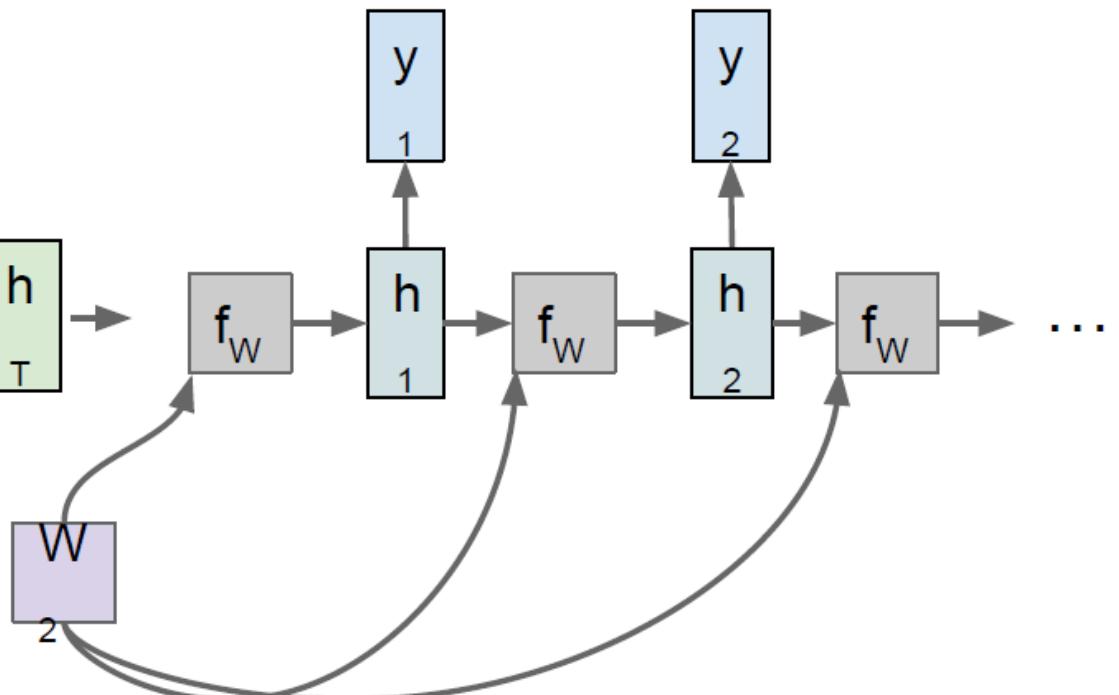


# Sequence to Sequence: Many to One + One to Many

**Many to one:** Encode input sequence in a single vector



**One to many:** Produce output sequence from single input vector

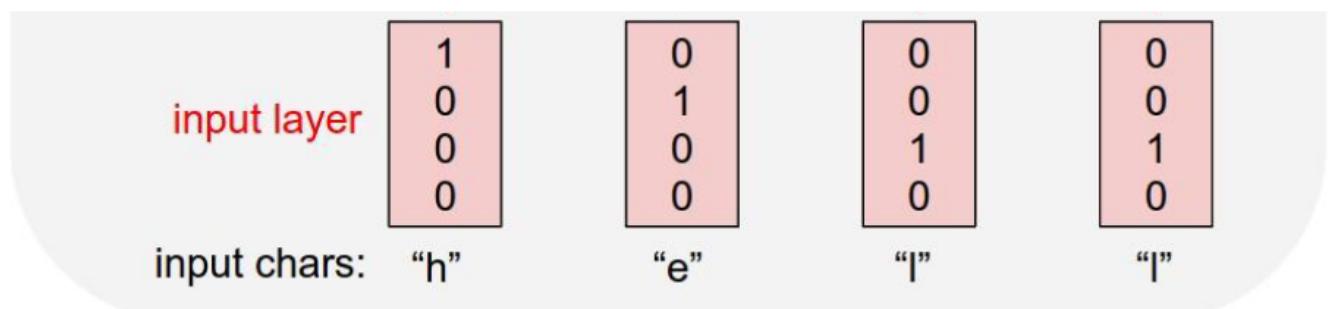


# RNN

**Character-level  
language model  
example**

Vocabulary:  
[h,e,l,o]

Example training  
sequence:  
**“hello”**



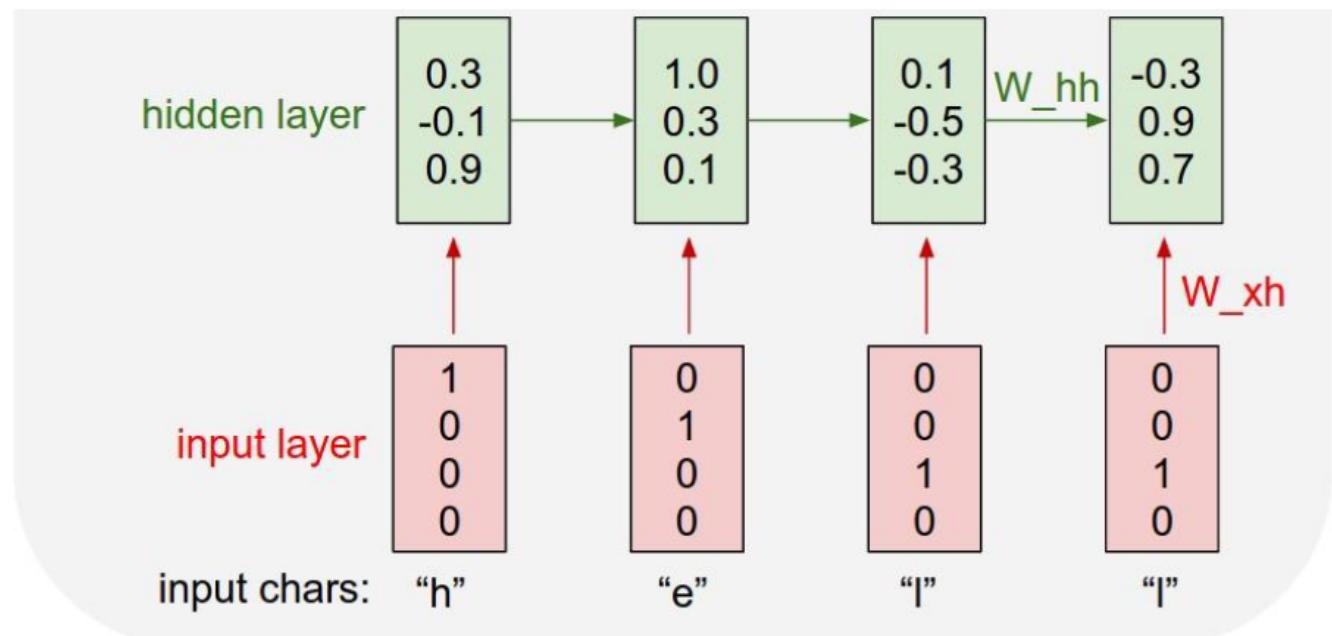
# RNN

## Character-level language model example

Vocabulary:  
[h,e,l,o]

Example training  
sequence:  
“hello”

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

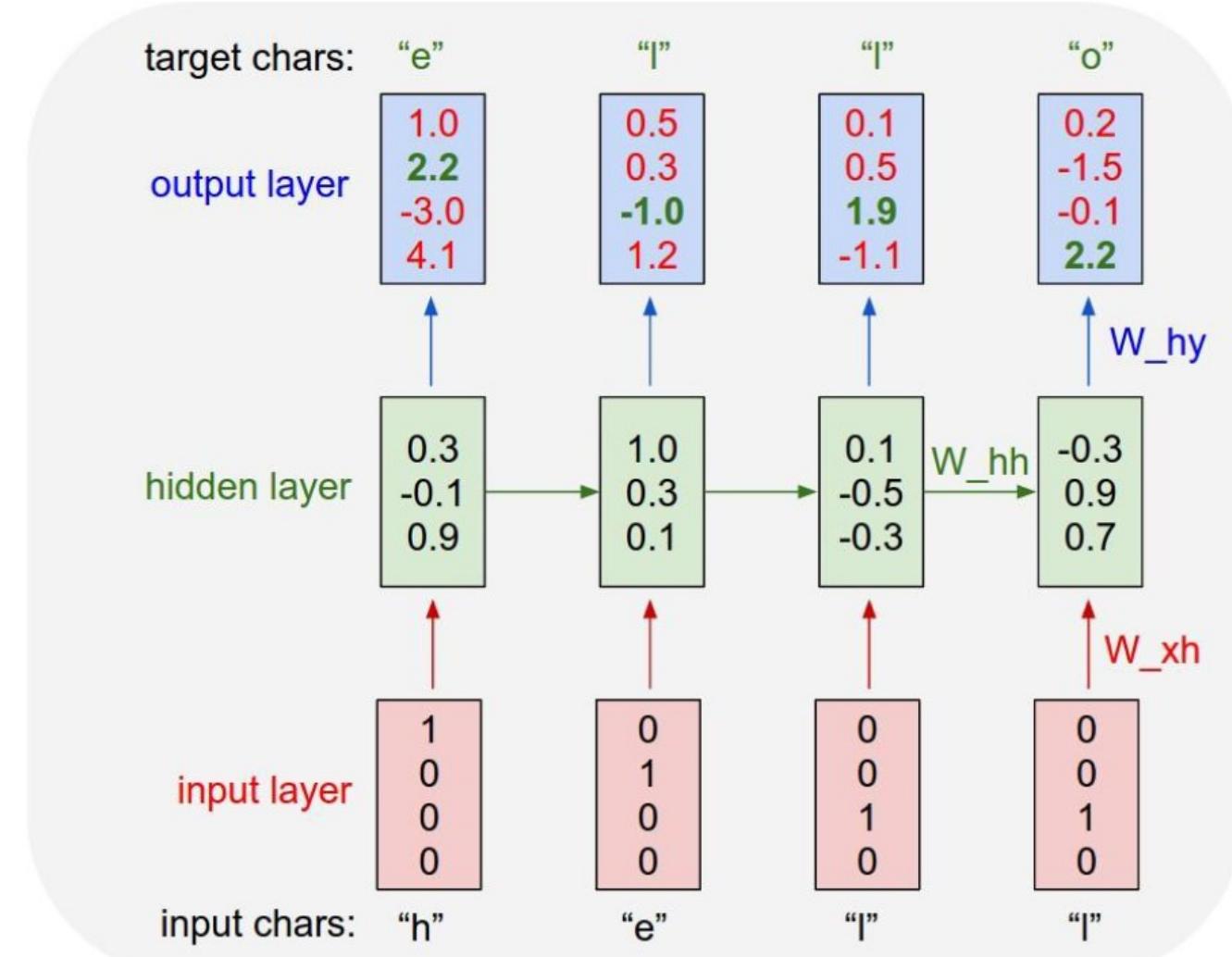


# RNN

## Character-level language model example

Vocabulary:  
[h,e,l,o]

Example training  
sequence:  
“hello”



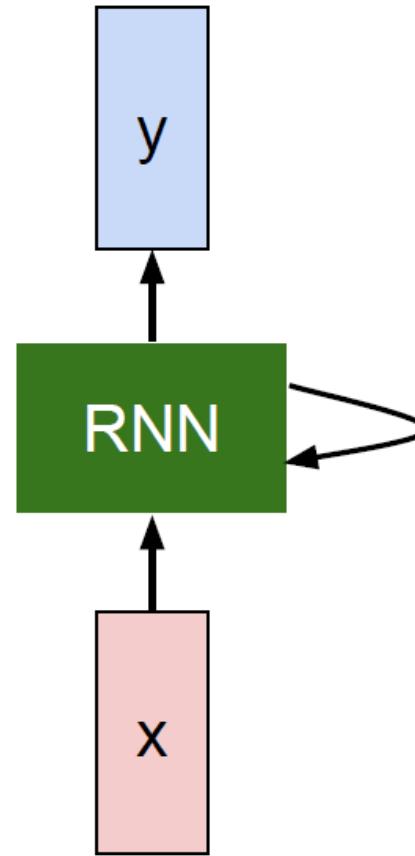
# Language Model

## THE SONNETS

by William Shakespeare

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

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



# Language Model

at first:

tyntd-iafhatawiaoahrdemot lytdws e ,tfti, astai f ogoh eoase rrranbyne 'nhthnee e  
plia tkldg 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.

# Language Model

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.

# Language Model

*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  $X_{\text{étale}}$  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. □

**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

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

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  $U$ . □

*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  $C$ . The functor  $\mathcal{F}$  is a “field”

$$\mathcal{O}_{X,x} \rightarrow \mathcal{F}_{\bar{x}} \rightarrow (\mathcal{O}_{X_{\text{étale}}})^{-1}(\mathcal{O}_{X_\lambda}(\mathcal{O}_{X_\eta}^{\bar{v}}))$$

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

# Language Model

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

"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

# 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

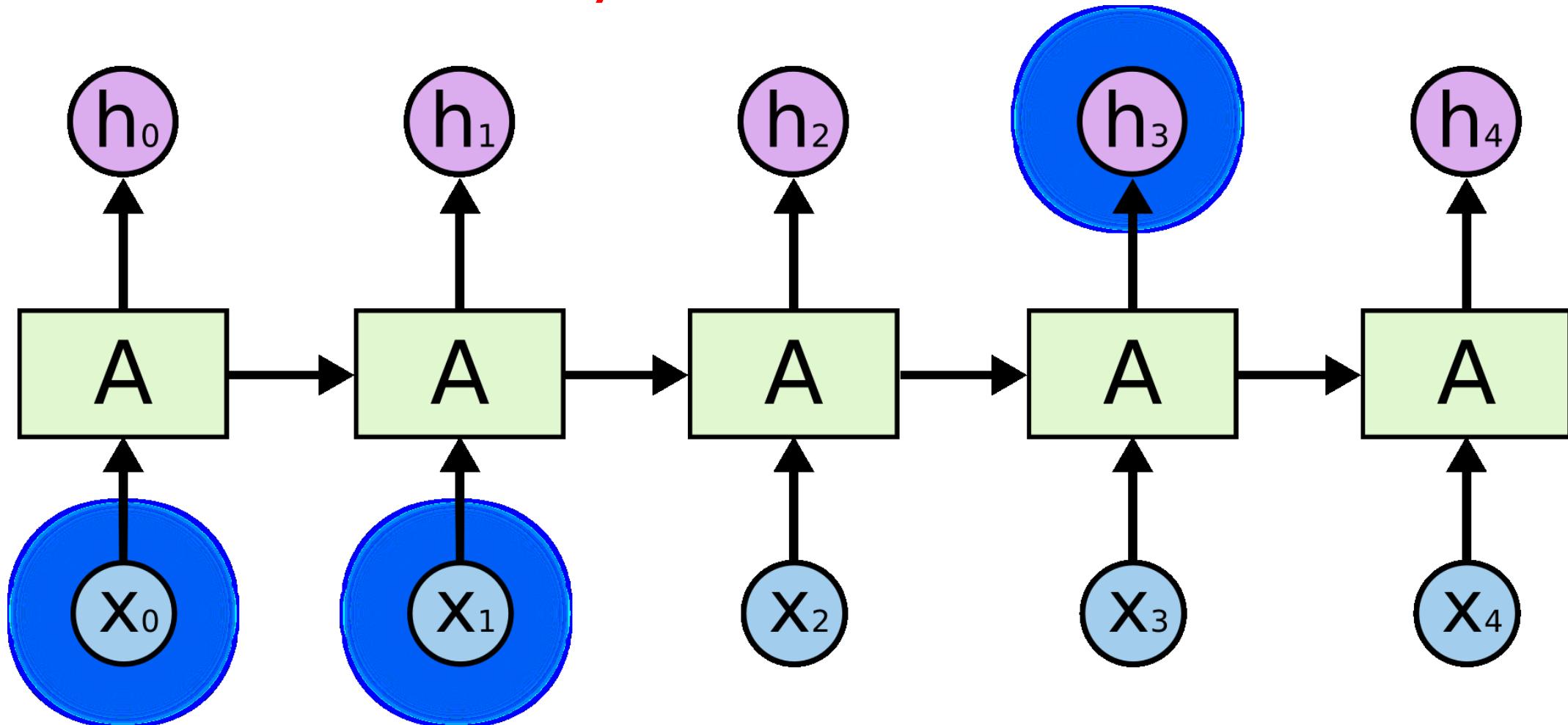
# Searching for Interpretable Cells

```
#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

# Long-Term Dependencies

- The **clouds** are in the **sky**



## Longer-Term Dependencies

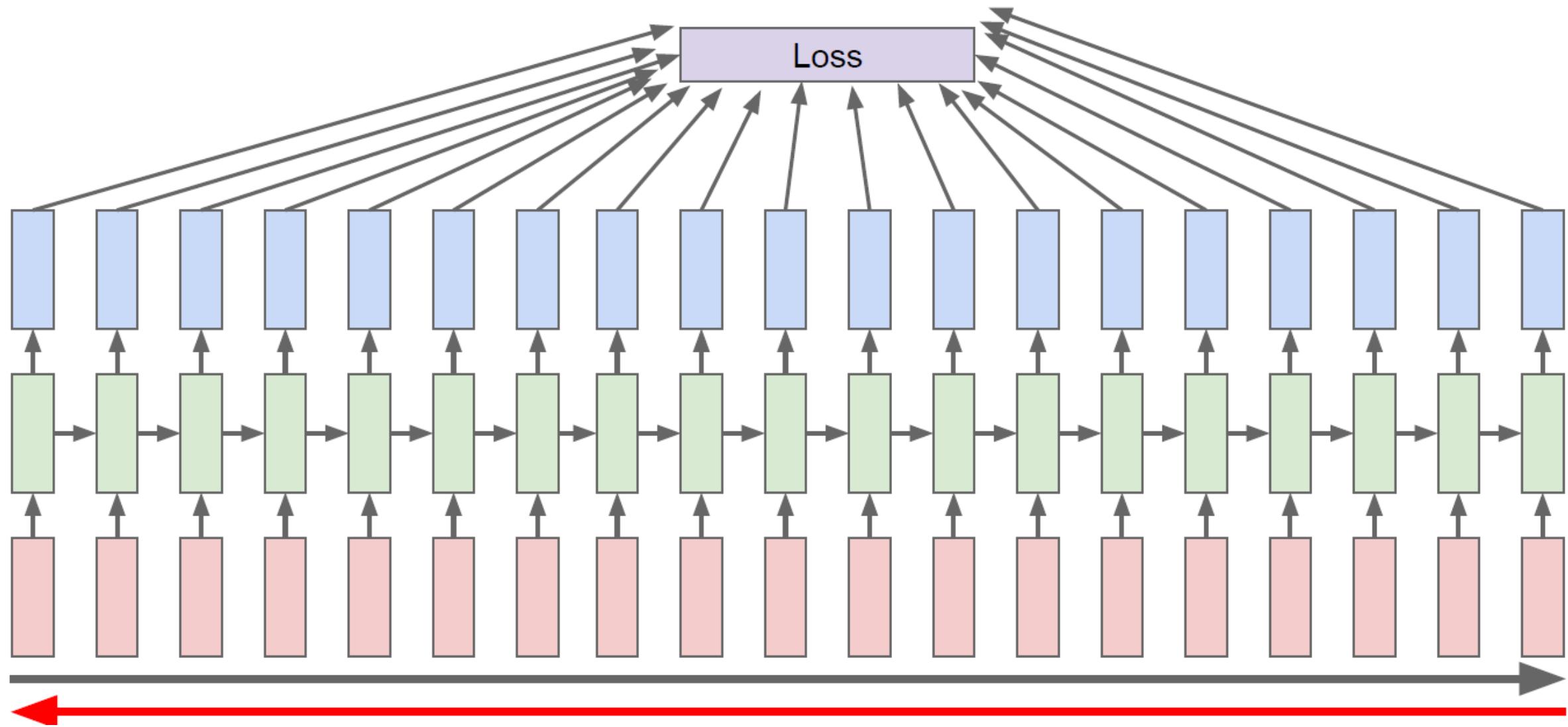
Lenny Khazan is a criminal, and  
should be sent to school.

Lenny Khazan is a criminal, and  
should be sent to the military.

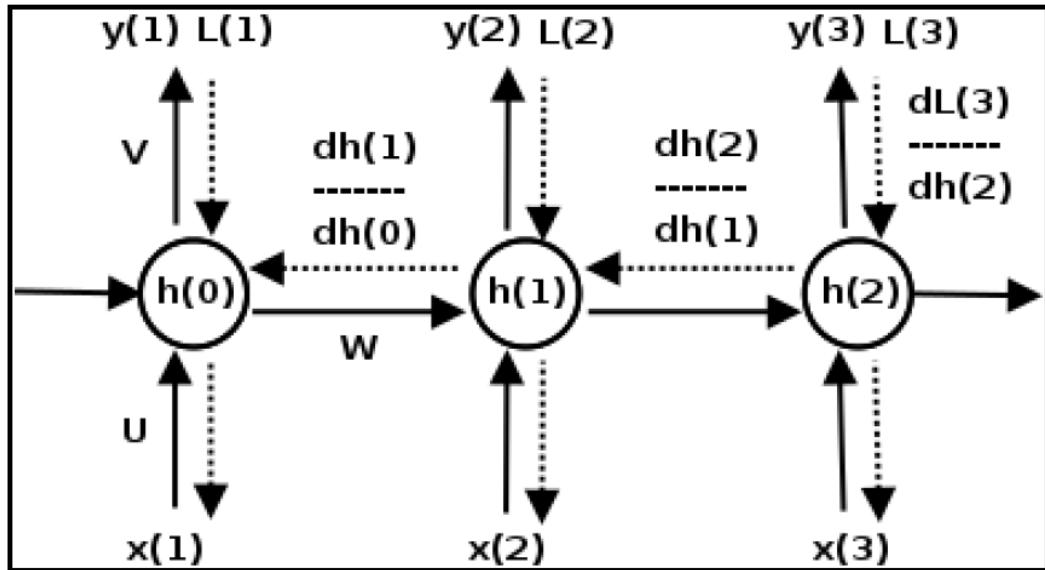
Lenny Khazan is a criminal, and  
should be sent to heaven.

Lenny Khazan is a criminal, and  
should be sent to jail.

# Back Propagation Through Time



# Vanishing/Exploding Gradient



$$h_t = \tanh(Wh_{t-1} + Ux_t)$$
$$y_t = \text{softmax}(Vh_t)$$

$$\frac{\partial L}{\partial W} = \sum_t \frac{\partial L_t}{\partial W}$$

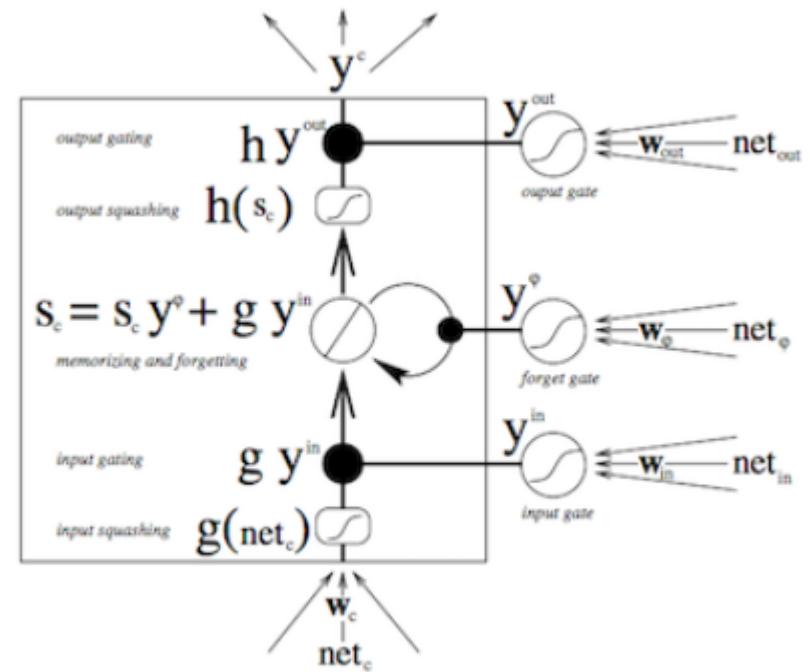
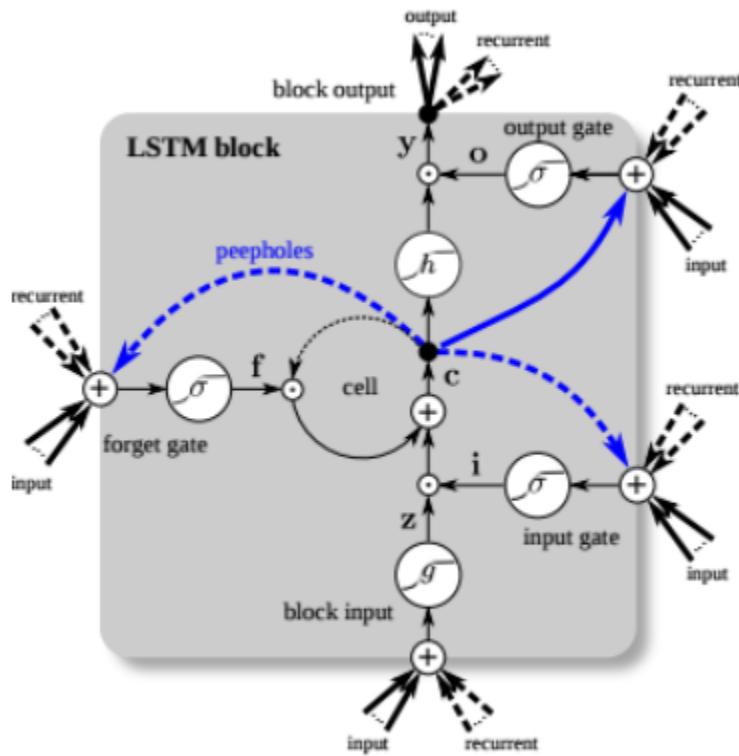
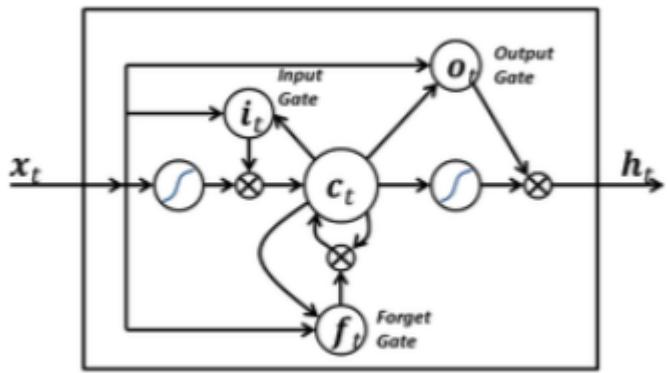
$$\frac{\partial L_3}{\partial W} = \frac{\partial L_3}{\partial y_3} \cdot \frac{\partial y_3}{\partial h_2} \cdot \frac{\partial h_2}{\partial W}$$

$$= \sum_{t=0}^2 \frac{\partial L_3}{\partial y_3} \cdot \frac{\partial y_3}{\partial h_2} \cdot \frac{\partial h_2}{\partial h_t} \cdot \frac{\partial h_t}{\partial W}$$

$$= \sum_{t=0}^2 \frac{\partial L_3}{\partial y_3} \cdot \frac{\partial y_3}{\partial h_2} \cdot \left( \prod_{j=t+1}^2 \frac{\partial h_j}{\partial h_{j-1}} \right) \cdot \frac{\partial h_t}{\partial W}$$

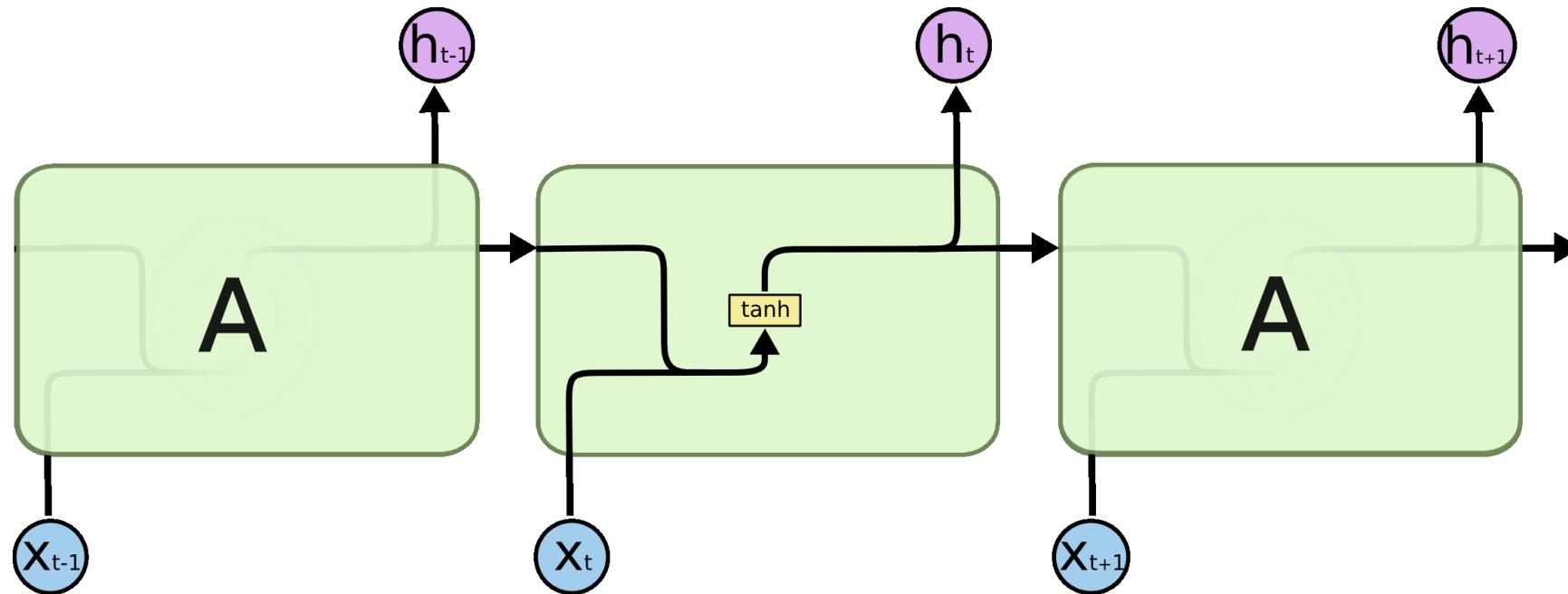
**Vanishing or Exploding  
Gradient!!**

# Long Short Term Memory



# LSTM

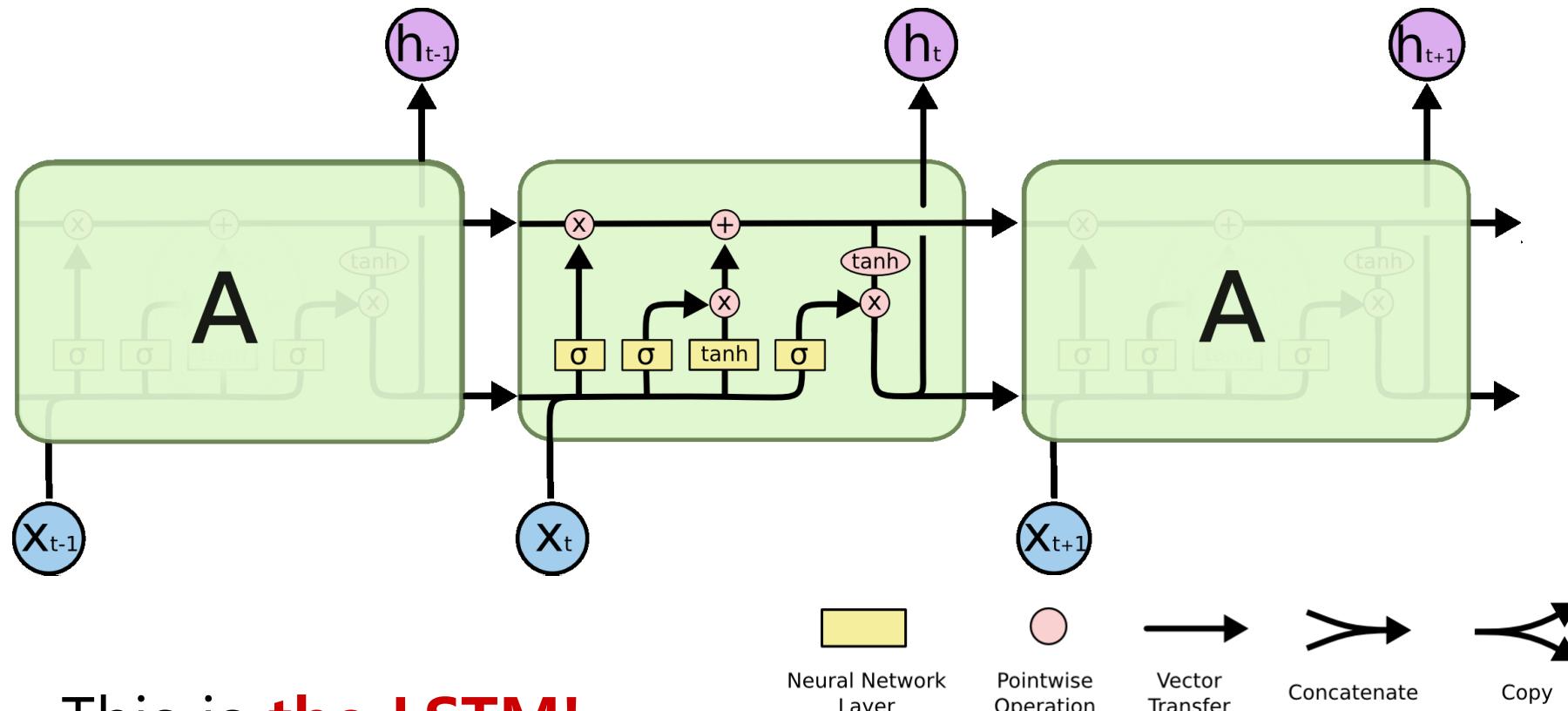
## Long Short Term Memory



This is just a standard RNN.

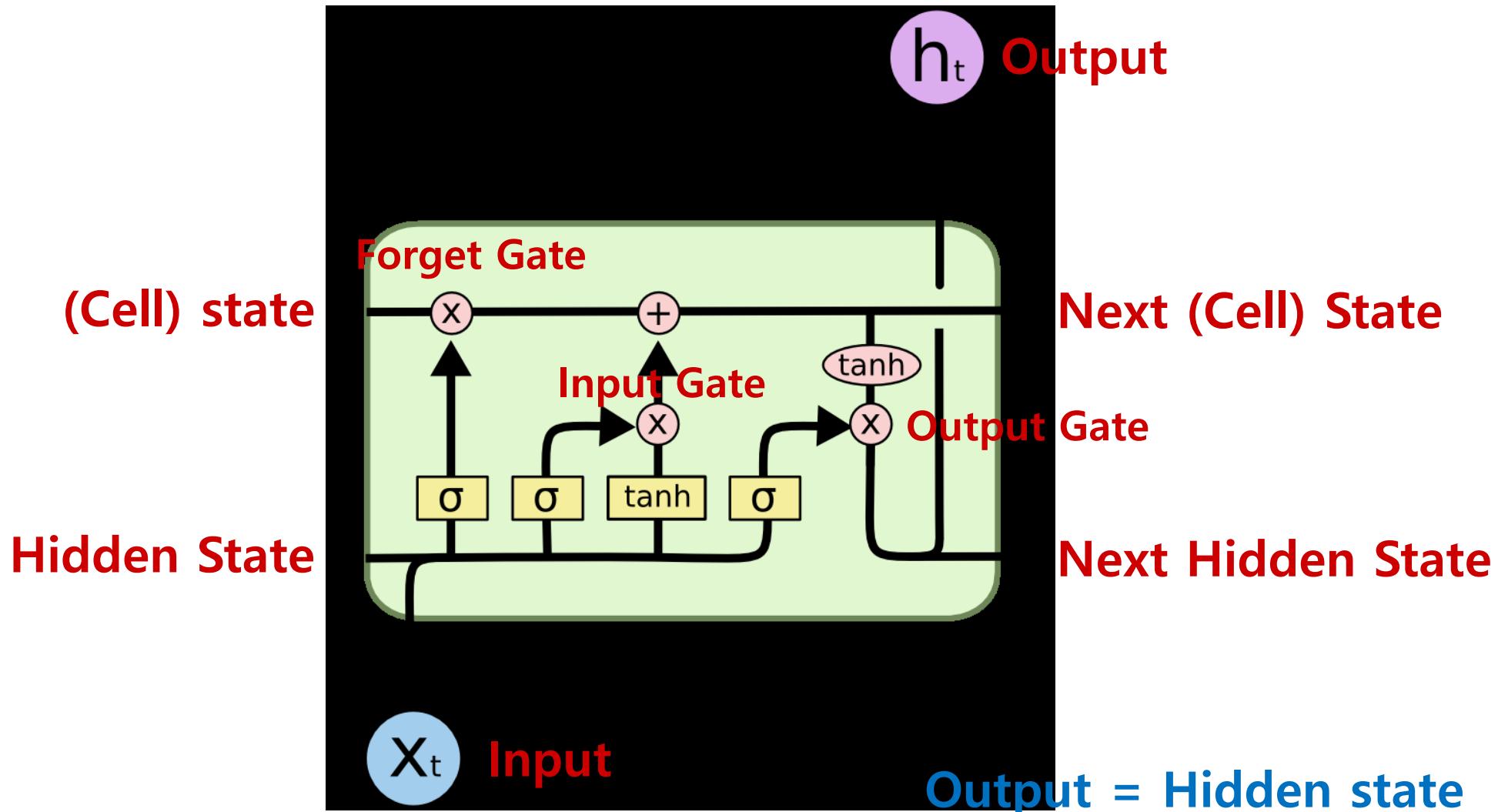
# LSTM

## Long Short Term Memory

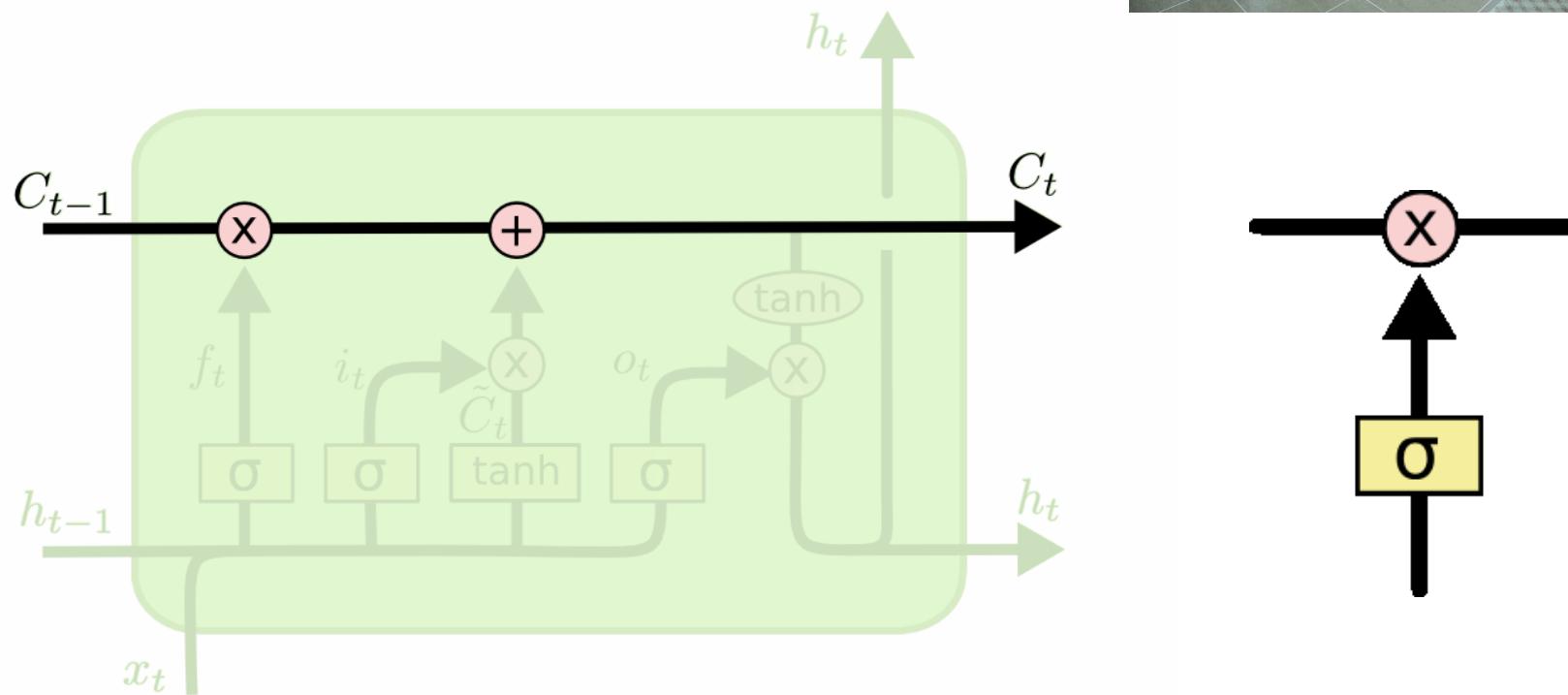


This is **the LSTM!**

# Overall Architecture

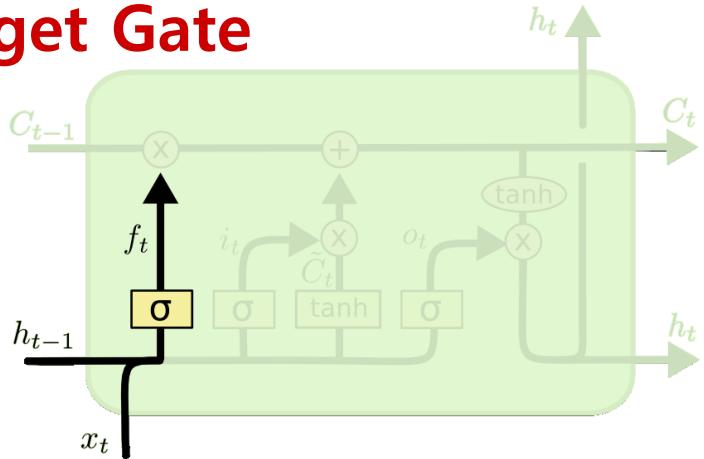


# The Core Idea



# Forget Gate & Input Gate

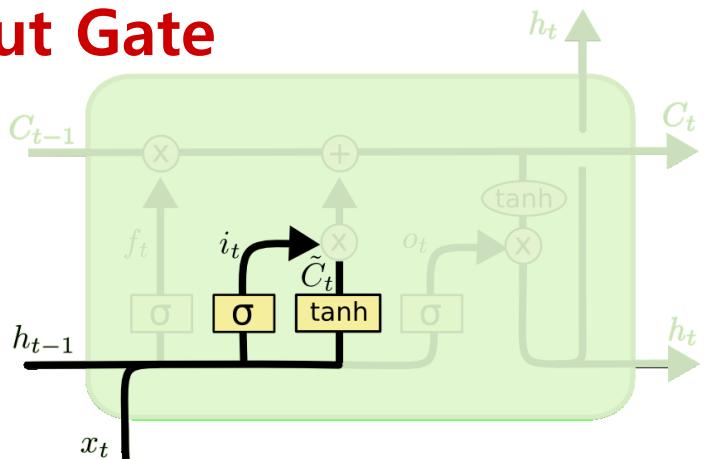
## Forget Gate



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

Decide what information we're going to **throw away** from the cell state.

## Input Gate



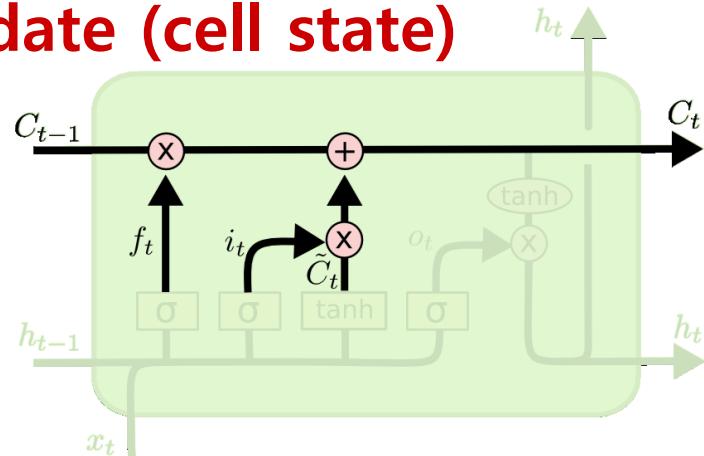
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

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

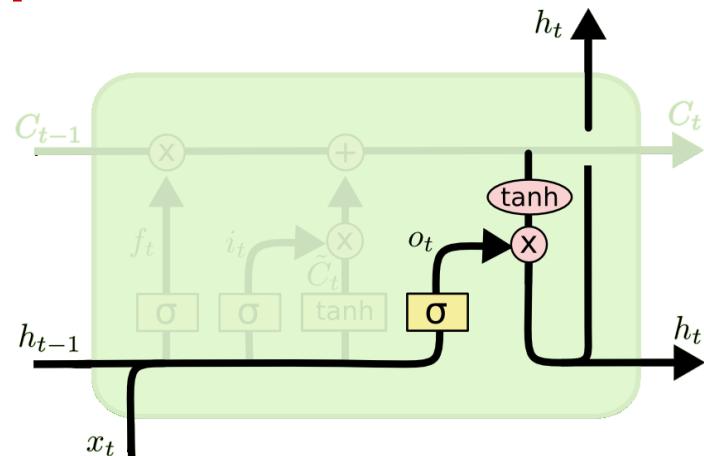
Decide what new information we're going to **store** in the cell state.

# Update Cell State & Output Gate

## Update (cell state)



## Output Gate (hidden state)



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

Update, scaled by how much we decide to update

: `input_gate*curr_state + forget_gate*prev_state`

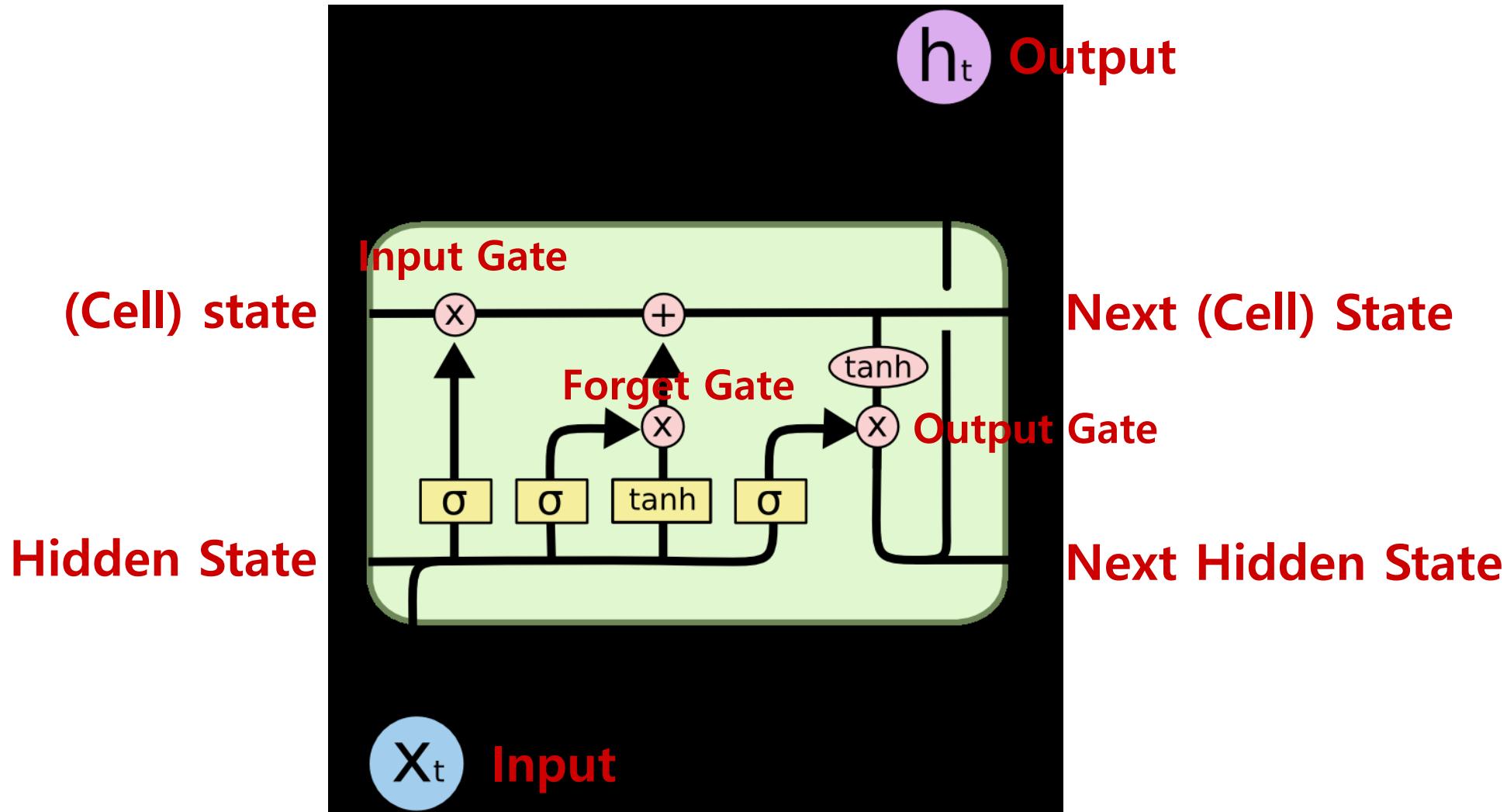
$$o_t = \sigma(W_o [ h_{t-1}, x_t ] + b_o)$$

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

Output based on the updated state

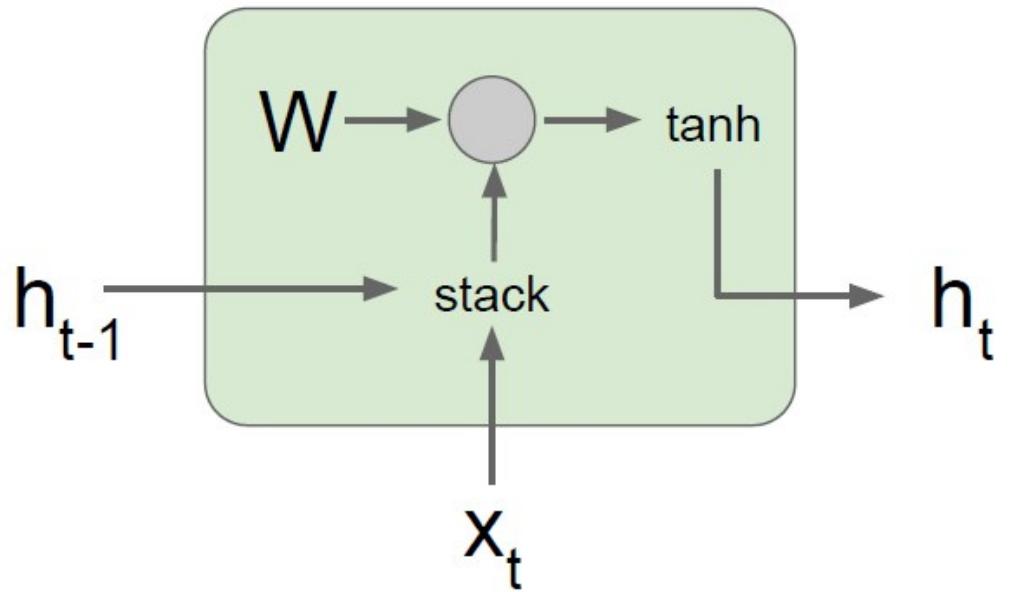
: `output_gate*updated_state`

# Putting It Together



Is LSTM free from Vanishing Gradient Problems?

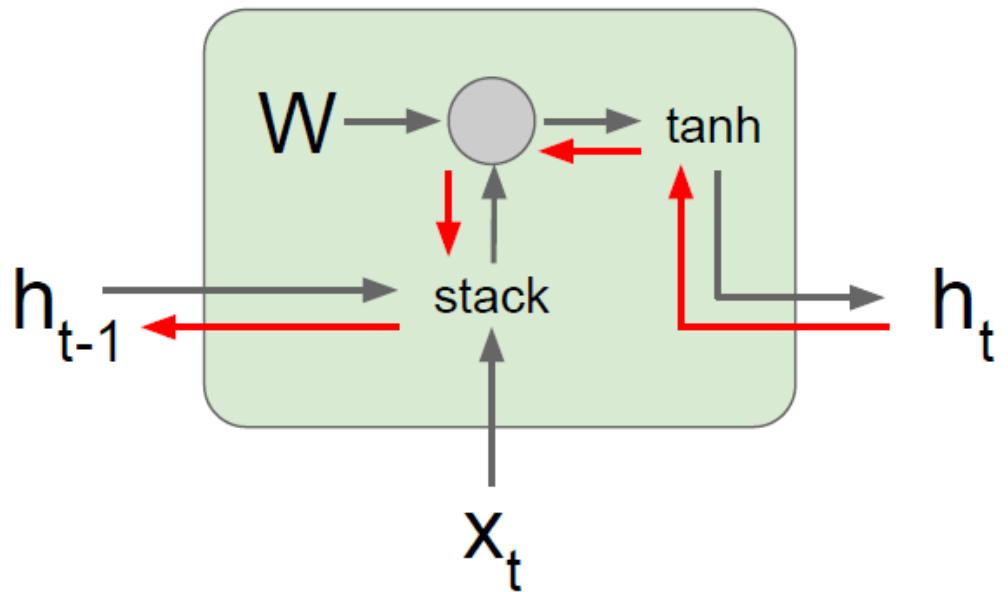
# Vanilla RNN Gradient Flow



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

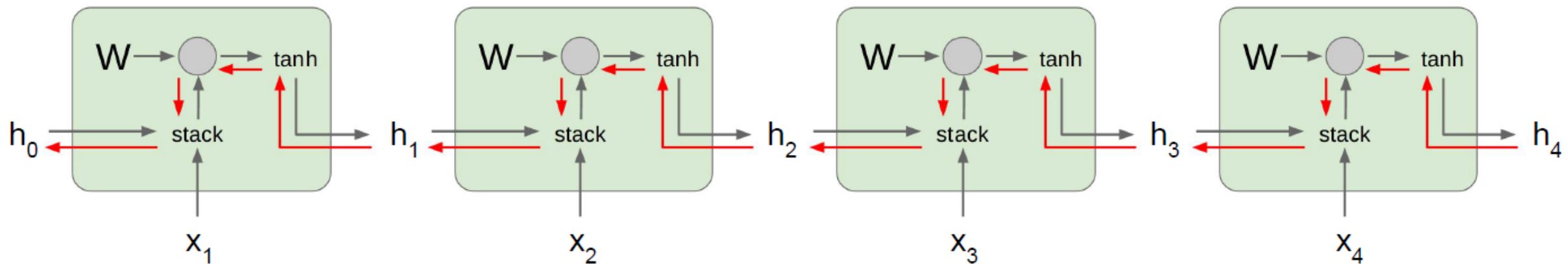
# Vanilla RNN Gradient Flow

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



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

# Vanilla RNN Gradient Flow



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

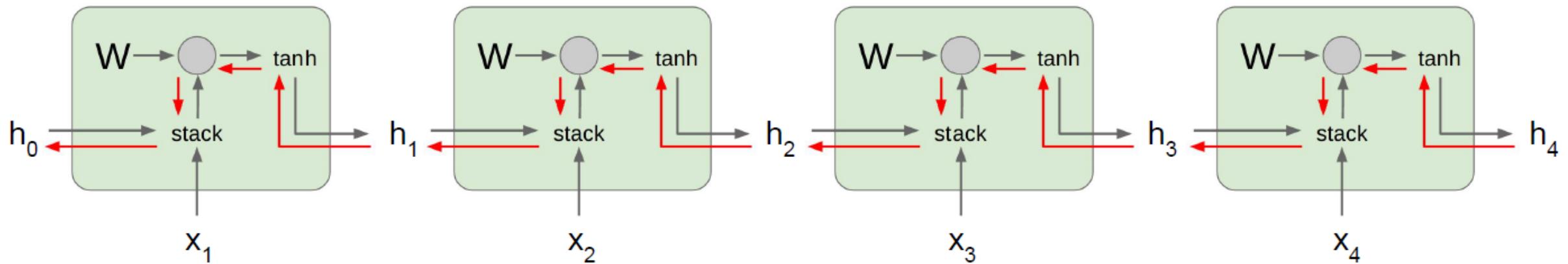
Largest singular value  $> 1$ :  
**Exploding gradients**

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



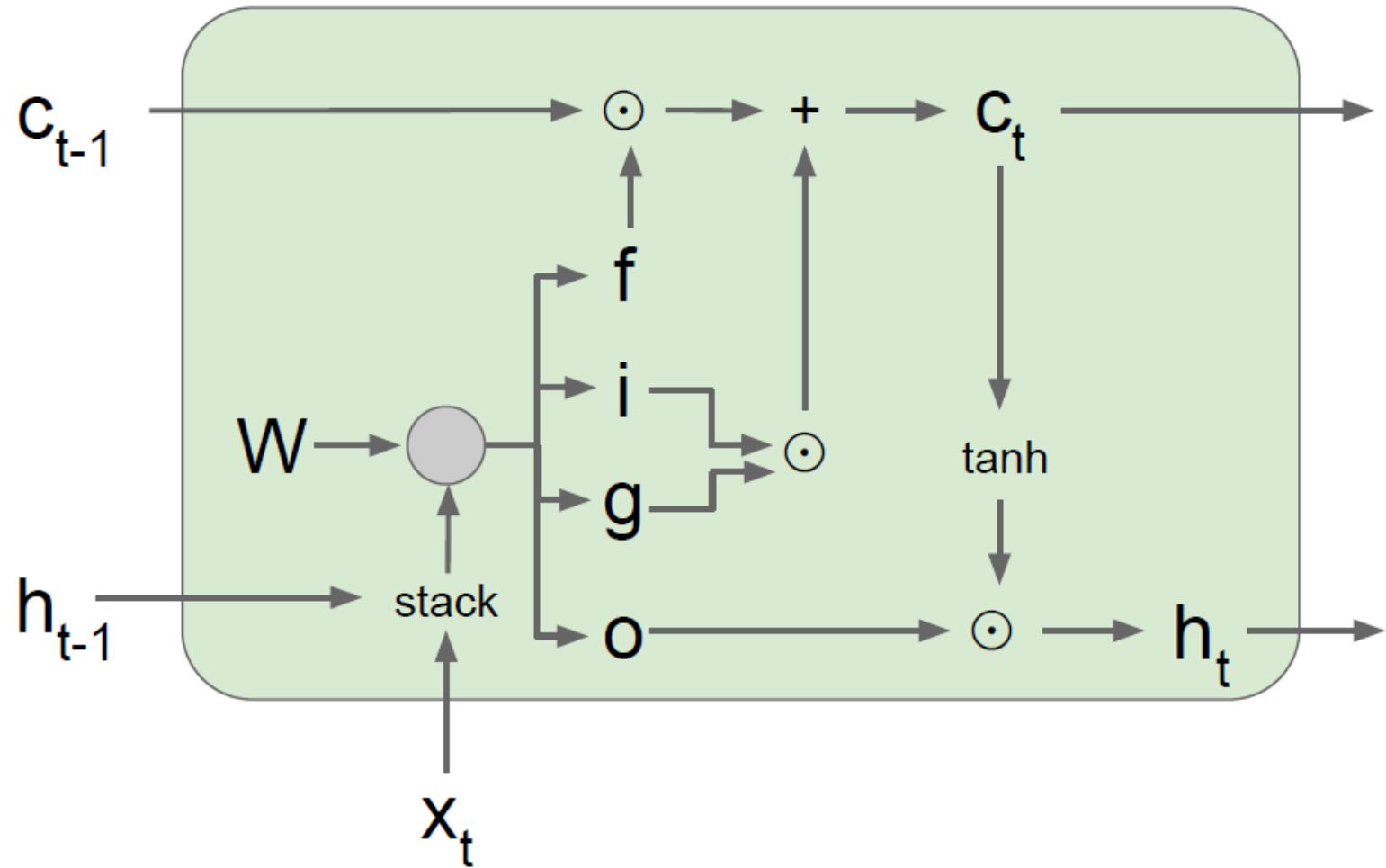
Computing gradient of  $h_0$  involves many factors of  $W$  (and repeated tanh)

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

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

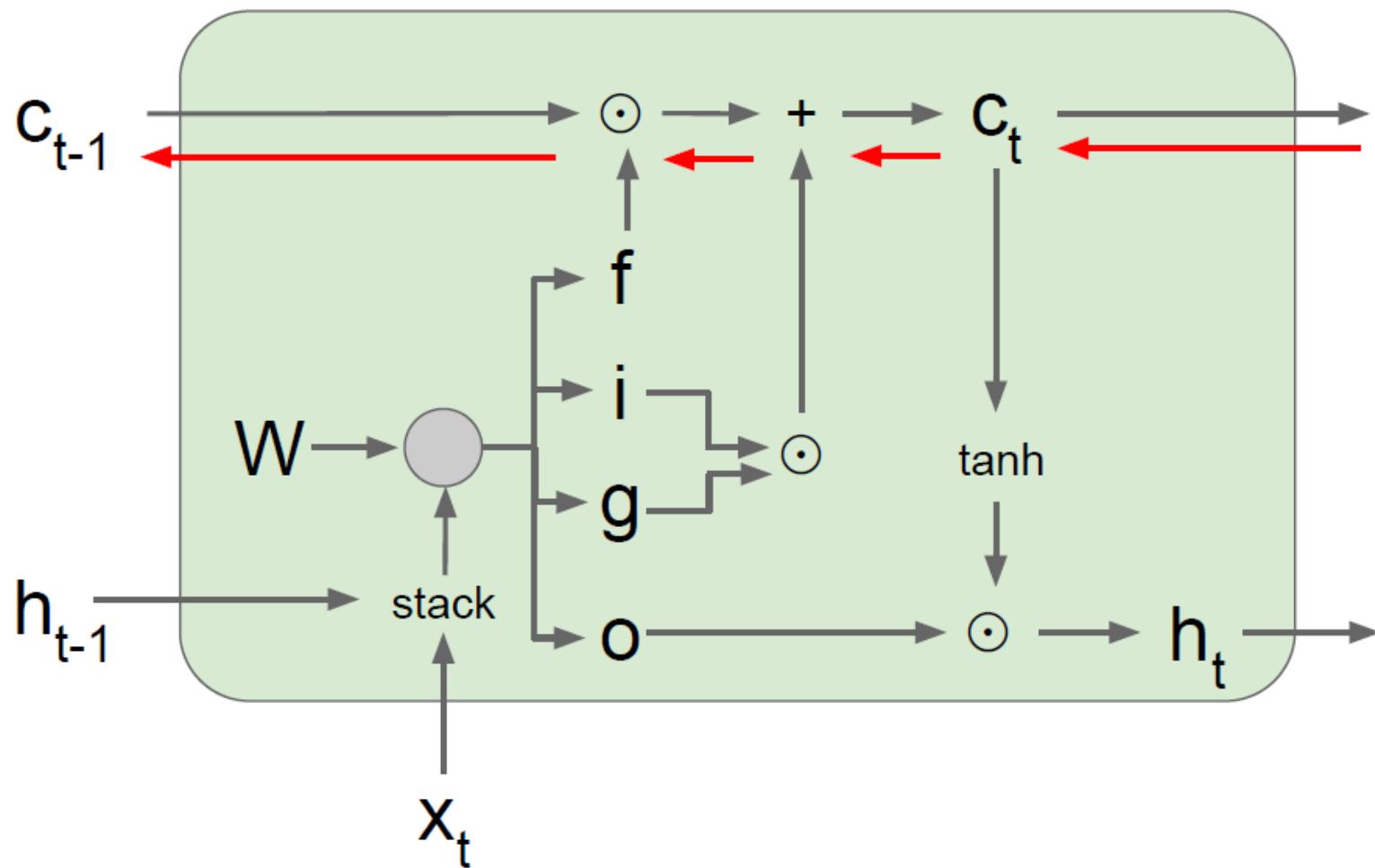
→ Change RNN architecture

# LSTM Gradient Flow



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

# LSTM Gradient Flow



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

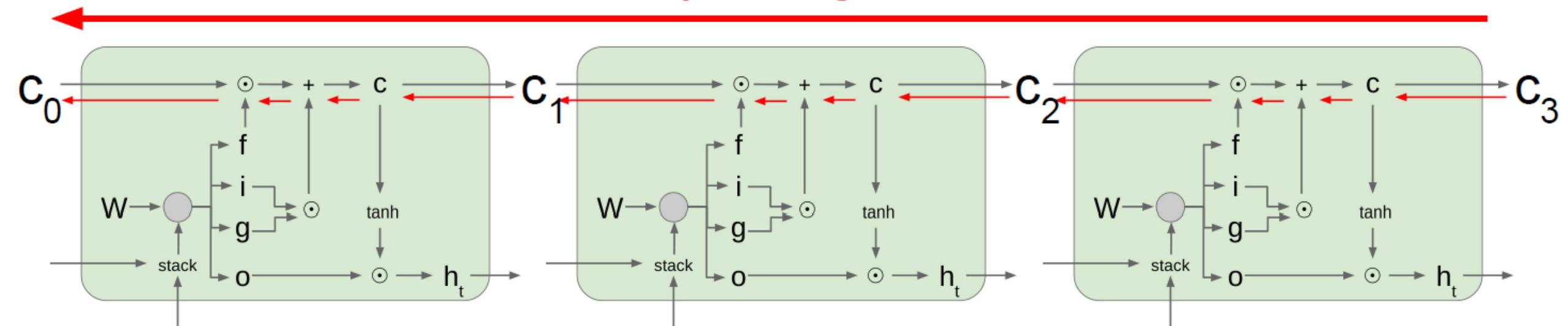
$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

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

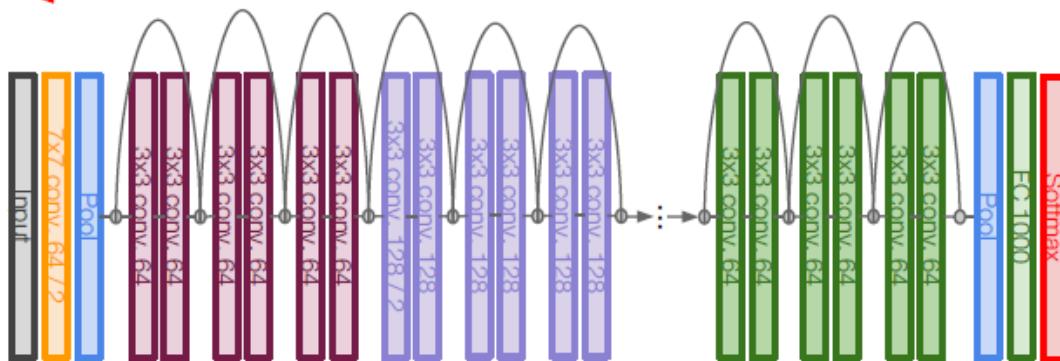
$$h_t = o \odot \tanh(c_t)$$

# LSTM Gradient Flow

Uninterrupted gradient flow!



Similar to ResNet!



# LSTM's Problems?

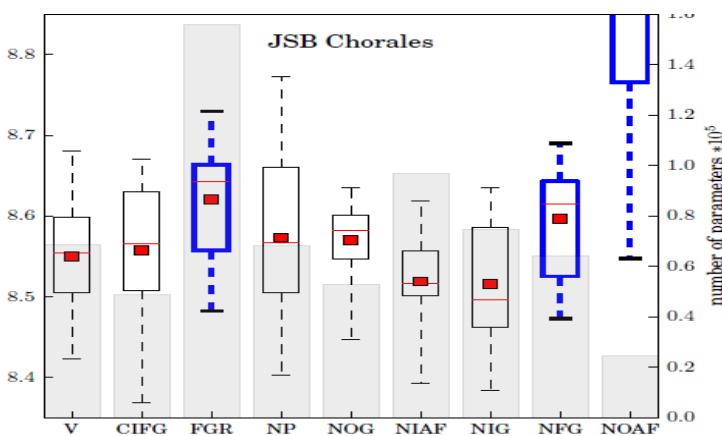
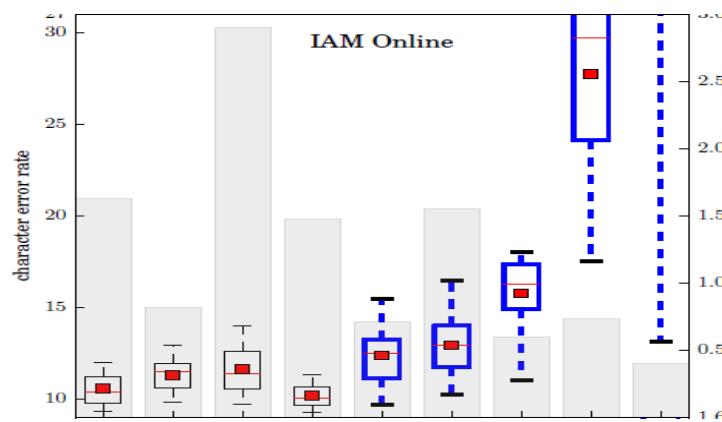
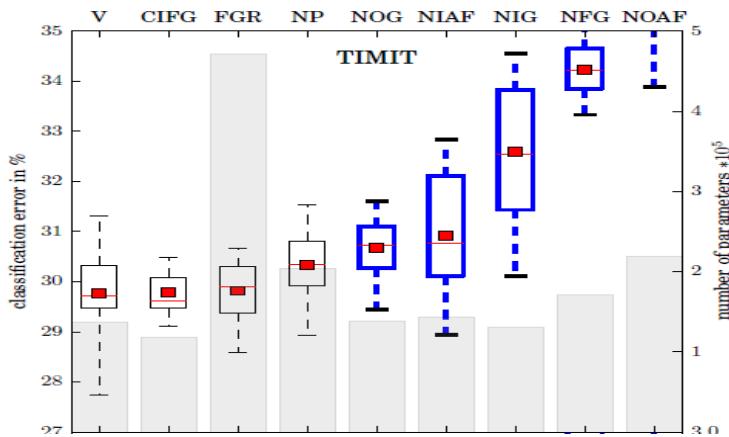
- Parameter가 너무 많고 복잡한데,  
뭔가 더 줄일 수 있는 여지가 없을까?
- gate를 곱해서 0~1사이의 non-linearity를 주는데,  
굳이 따로 activation function이 필요할까?
- Gate 수를 좀 줄여볼 수는 없을까?

# LSTM : A Search Space Odyssey

TIMIT : speech recognition database

IAM online : handwriting database ( pen movement )

JSB Chorales : polyphonic music database. 다음 음이 위 or 아래를 예측하는 문제, negative log likelihood를 측정했음.



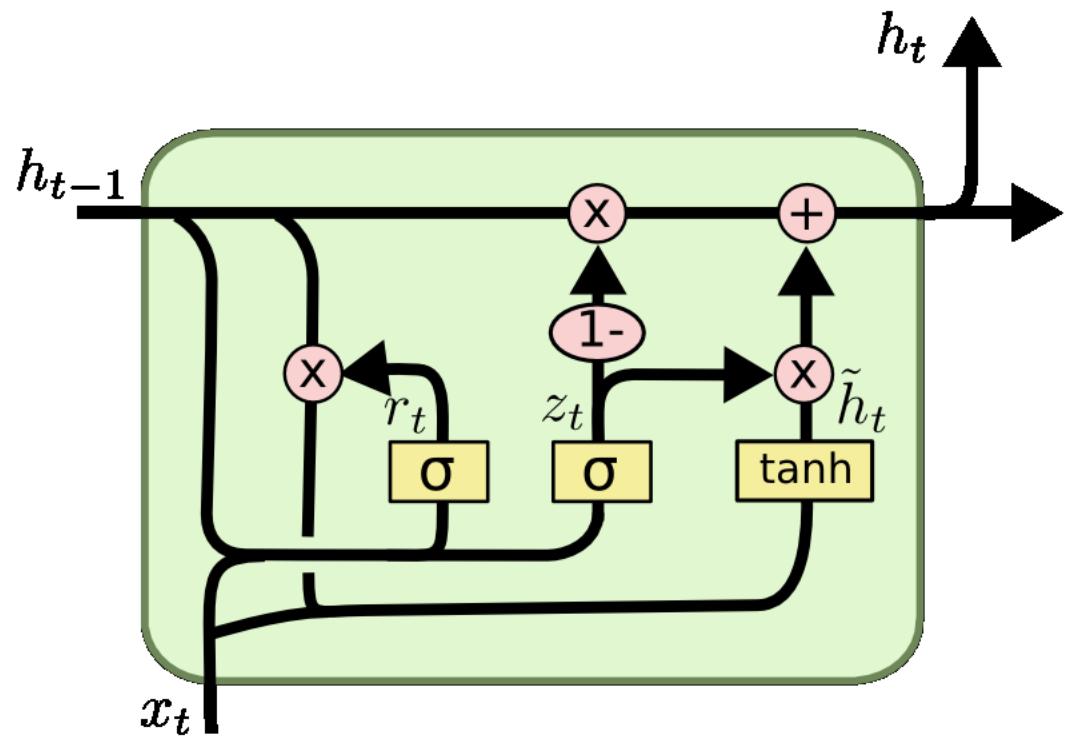
1. No Input Gate (NIG)
2. No Forget Gate (NFG)
3. No Output Gate (NOG)
4. No Input Activation Function (NIAF)
5. No Output Activation Function (NOAF)
6. No Peepholes (NP)
7. Coupled Input and Forget Gate (CIFG)
8. Full Gate Recurrence (FGR)

결론 :

- 1) peephole 은 성능에 별로 안 중요하다
- 2) Forget > Input > Output 게이트는 성능에 매우 중요하다
- 3) Activation function도 중요하다.
- 4) Full gate는 파라미터에 수에 비해서 잘 못한다.
- 5) Input하고 forget 게이트는 묶어도 되는 것 같다.

-> GRU로 발전

# Gated Recurrent Unit



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

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

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

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