Recurrent Neural Network

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

Language Model

Language Modeling

- Language Modeling is the task of predicting what word comes next.

the students opened their _____ → books, laptops, exams, minds
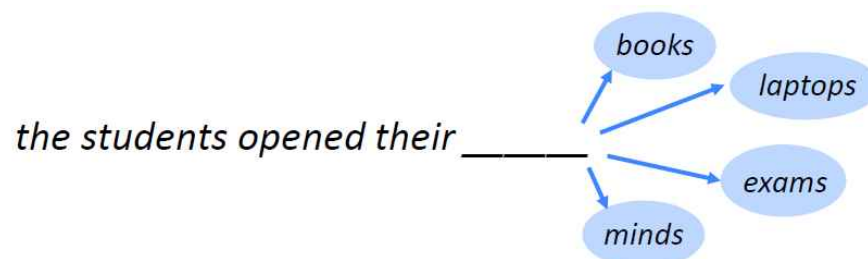
- More formally: given a sequence of words $x^{(1)}, x^{(2)}, \ldots, x^{(t)}$, compute the probability distribution of the next word $x^{(t+1)}$:

$$P(\boldsymbol{x}^{(t+1)} \mid \boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(1)})$$

where $x^{(t+1)}$ can be any word in the vocabulary $V = \{w_1, \ldots, w_{|V|}\}$

- A system that does this is called a Language Model.

Language Modeling

- You can also think of a Language Model as a system that assigns probability to a piece of text.

- For example, if we have some text $x^{(1)}, x^{(2)}, \ldots, x^{(T)}$, then the probability of this text(according to the Language Model) is :

$$P(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(T)}) = P(\boldsymbol{x}^{(1)}) \times P(\boldsymbol{x}^{(2)} \mid \boldsymbol{x}^{(1)}) \times \cdots \times P(\boldsymbol{x}^{(T)} \mid \boldsymbol{x}^{(T-1)}, \ldots, \boldsymbol{x}^{(1)})$$

$$= \prod_{t=1}^{T} P(\boldsymbol{x}^{(t)} \mid \boldsymbol{x}^{(t-1)}, \ldots, \boldsymbol{x}^{(1)})$$

This is what our LM provides

N-gram Language Models

The students opened their _____

- Definition: A n-gram is a chunk of n consecutive words.
    - Unigrams: "the", "students", "opened", "their"
    - Bigrams: "the students", "students opened", "opened their"
    - Trigrams: "the students opened", "students opened their"
    - 4-grams: "the students opened their"

- Idea: Collect statistics about how frequent different n-grams are, and use these to predict next word.

N-gram Language Models

- First we make a simplifying assumption: $x^{(t+1)}$ depends only on the preceding *n-1* words.

n-1 words

$$P(\boldsymbol{x}^{(t+1)}|\boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(1)}) = P(\boldsymbol{x}^{(t+1)}|\boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(t-n+2)}) \qquad \text{(assumption)}$$

prob of a n-gram

prob of a (n-1)-gram

$$= \frac{P(\boldsymbol{x}^{(t+1)}, \boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(t-n+2)})}{P(\boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(t-n+2)})} \qquad \begin{array}{l}\text{(definition of} \\ \text{conditional prob)}\end{array}$$

- **Question:** How do we get these *n*-gram and (*n*-1)-gram probabilities?
- **Answer:** By counting them in some large corpus of text!

$$\approx \frac{\text{count}(\boldsymbol{x}^{(t+1)}, \boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(t-n+2)})}{\text{count}(\boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(t-n+2)})} \qquad \begin{array}{l}\text{(statistical} \\ \text{approximation)}\end{array}$$

N-gram Language Models

- Suppose we are learning 4-gram Language Model.

~~as the proctor started the clock, the~~ *students opened their* _____

discard

condition on this

$$P(\boldsymbol{w}|\text{students opened their}) = \frac{\text{count}(\text{students opened their } \boldsymbol{w})}{\text{count}(\text{students opened their})}$$

For example, suppose that in the corpus:

- "students opened their" occurred 1000 times
- "students opened their books" occurred 400 times
  - → P(books | students opened their) = 0.4
- "students opened their exams" occurred 100 times
  - → P(exams | students opened their) = 0.1

Should we have discarded the "proctor" context?

Problems of a N-gram Language Models

- Sparsity Problems
  - What if "students open their (w)" never occurred in data? Then (w) has probability 0!

$$P(\boldsymbol{w}|\text{students opened their}) = \frac{\text{count}(\text{students opened their } \boldsymbol{w})}{\text{count}(\text{students opened their})}$$
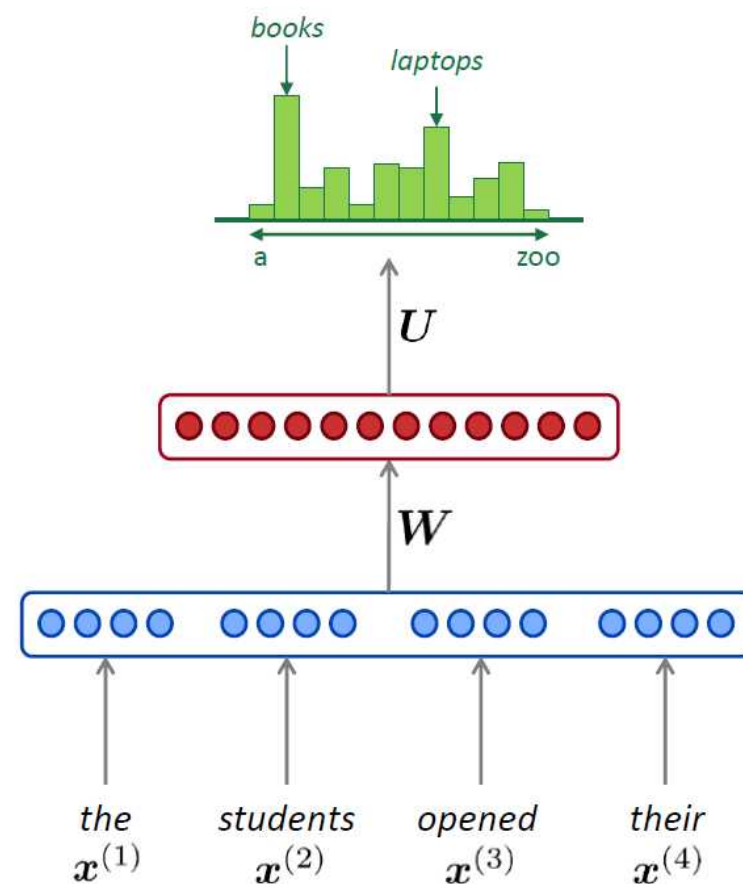
- Inability to capture long-term dependencies.

~~as the proctor started the clock, the~~ *students opened their* _____

discard

condition on this

- Need to store count for all n-grams you saw in the corpus. Increasing n or increasing corpus increases model size.

How to Build a Neural Language Model?

- Recall the Language Modeling task:
  - Input: sequence of words $x^{(1)}, x^{(2)}, \ldots, x^{(t)}$
  - Output: prob dist of the next word
    $P\big(x^{(t+1)}\big|x^{(t)}, \ldots, x^{(1)}\big)$

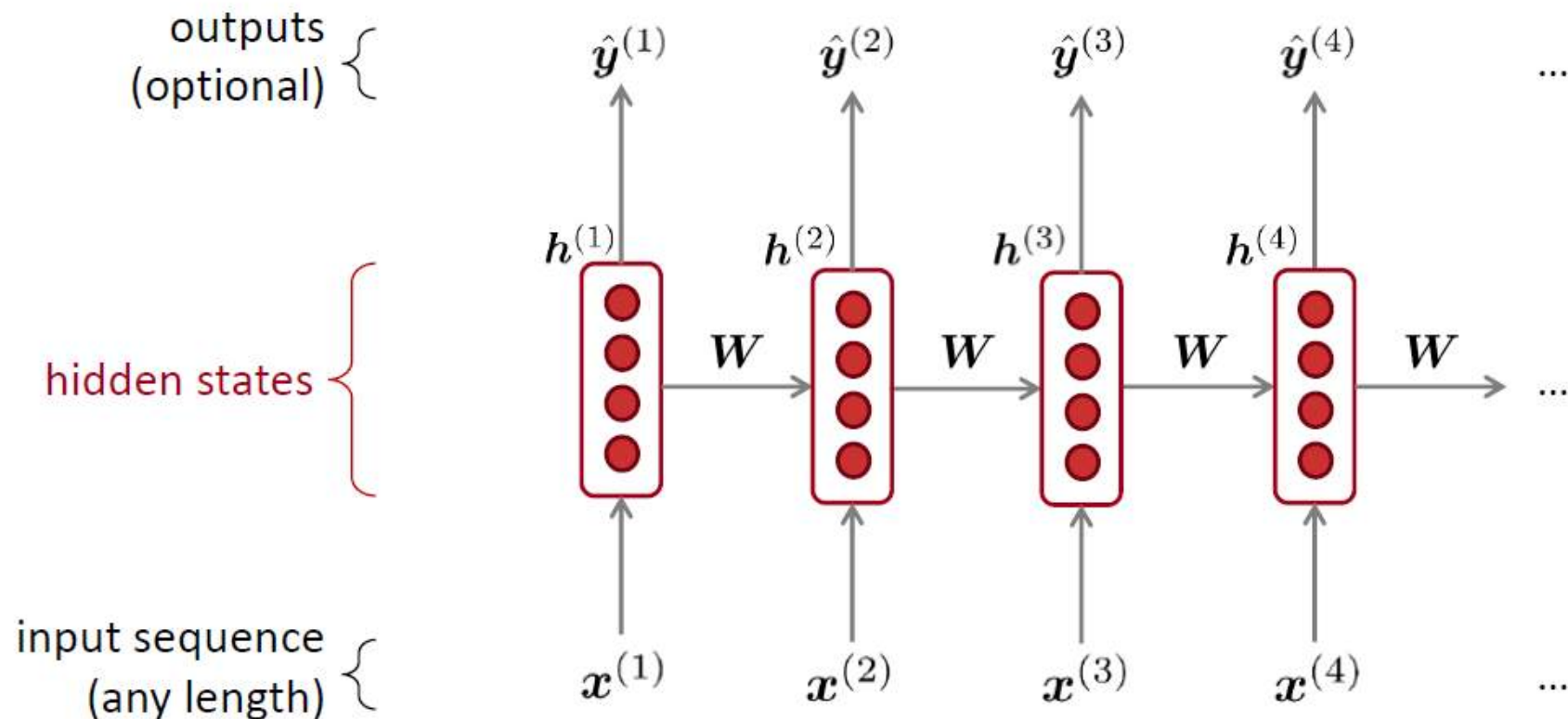- How about a window-based neural model?

A Fixed-Window Neural Language Model

- Improvement over n-gram LM:
  - No sparsity problem
  - Don't need to store all observed n-grams

- Problems
  - Fixed window is too small
  - Long term dependencies cannot be captured
  - Enlarging window enlarges W
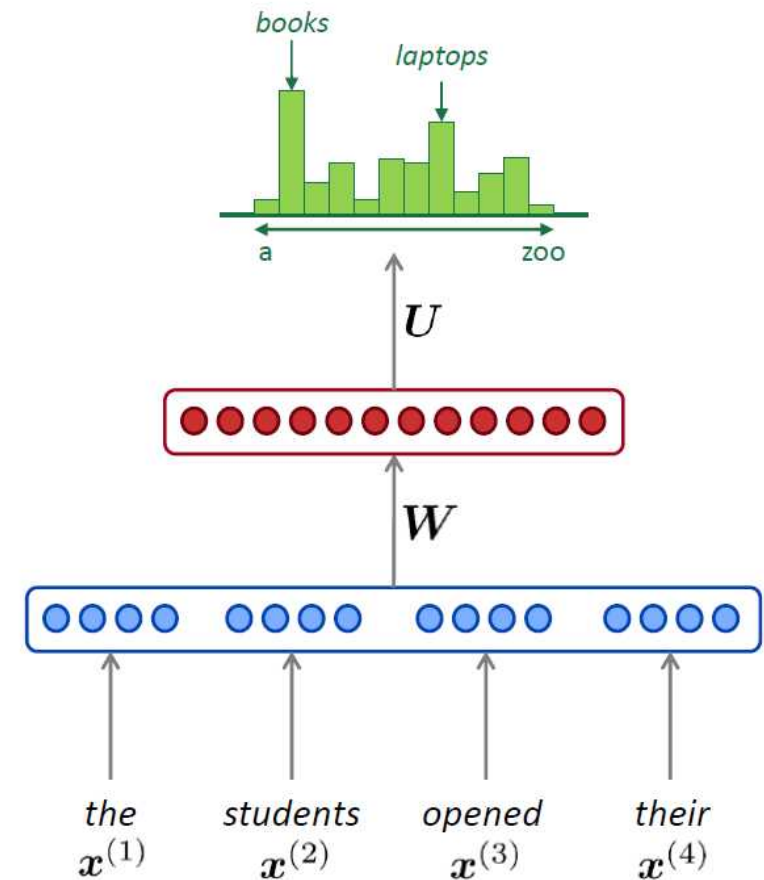
Recurrent Neural Network

- Core idea: Apply the same weights W repeatedly

A RNN Language Model

$\hat{y}^{(4)} = P(x^{(5)}|\text{the students opened their})$

output distribution

$$\hat{y}^{(t)} = \text{softmax}\left(Uh^{(t)} + b_2\right) \in \mathbb{R}^{|V|}$$

hidden states

$$h^{(t)} = \sigma\left(W_h h^{(t-1)} + W_e e^{(t)} + b_1\right)$$

$h^{(0)}$ is the initial hidden state

word embeddings

$$e^{(t)} = Ex^{(t)}$$

words / one-hot vectors

$$x^{(t)} \in \mathbb{R}^{|V|}$$
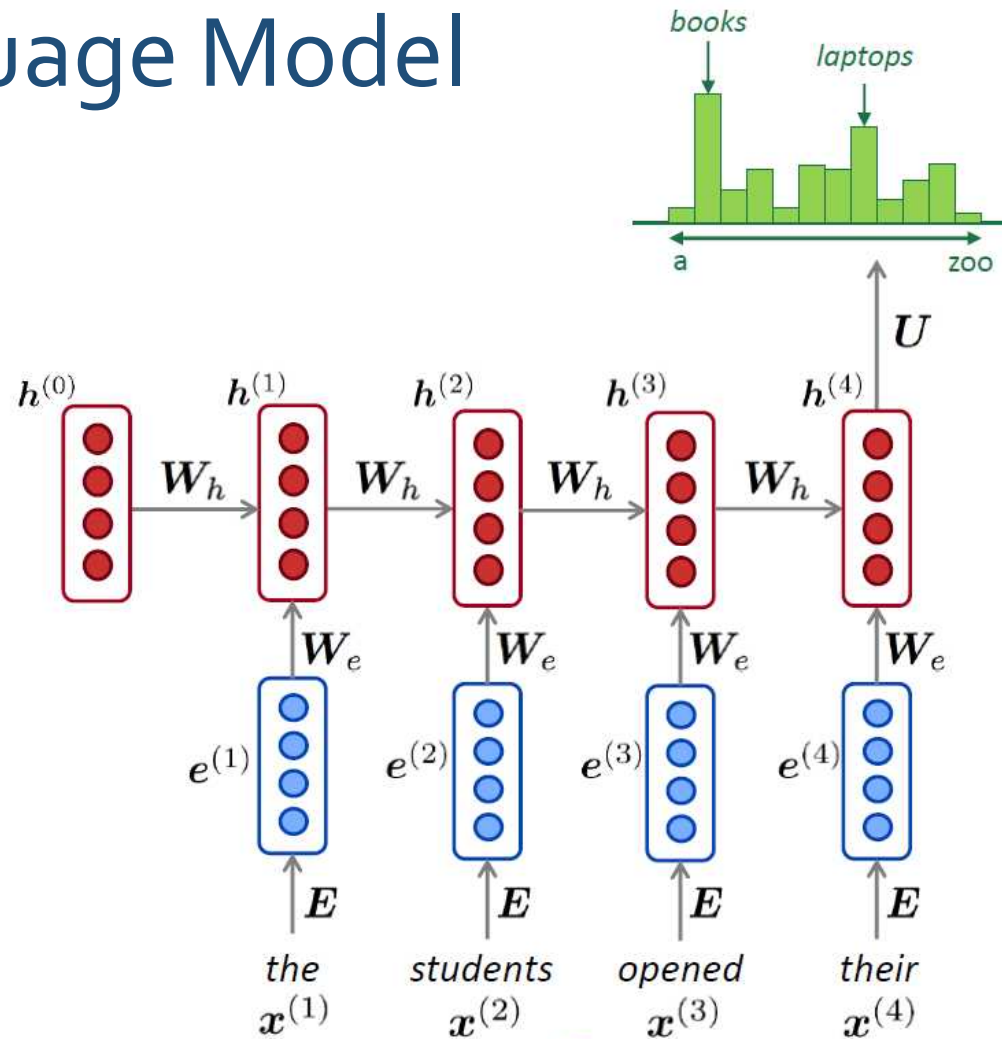
Note: this input sequence could be much longer, but this slide doesn't have space!

A RNN Language Model

$$\hat{y}^{(4)} = P(x^{(5)}|\text{the students opened their})$$

RNN **Advantages**:
- Can process any length input
- Computation for step t can (in theory) use information from many steps back
- Model size doesn't increase for longer input
- Same weights applied on every timestep, so there is symmetry in how inputs are processed.
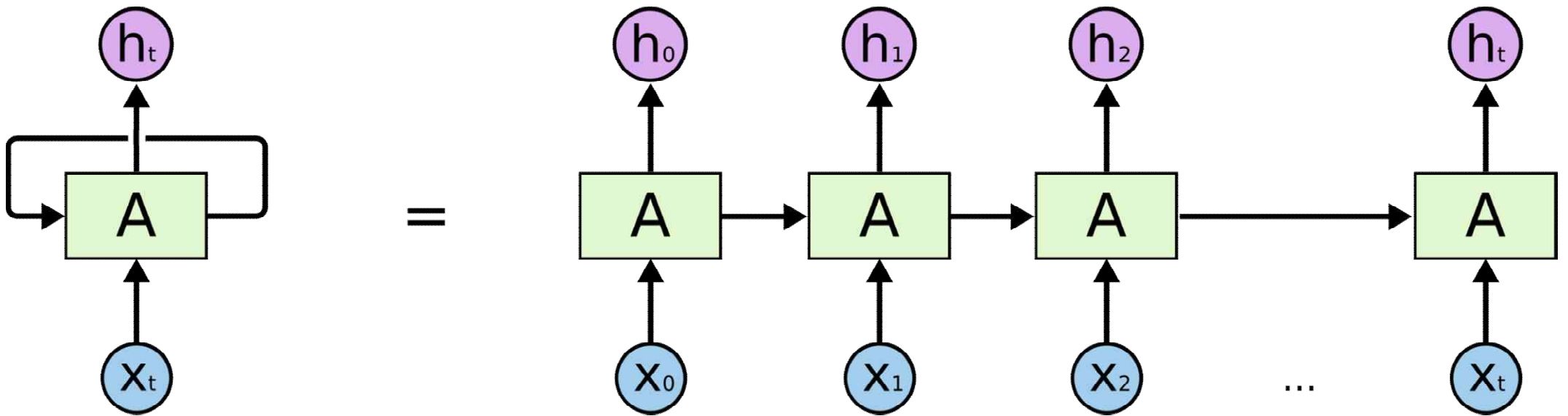
RNN **Disadvantages**:
- Recurrent computation is slow
- In practice, difficult to access information from many steps back

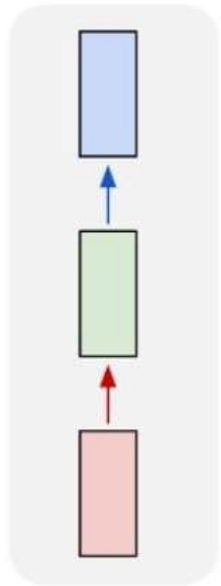More on these later in the course
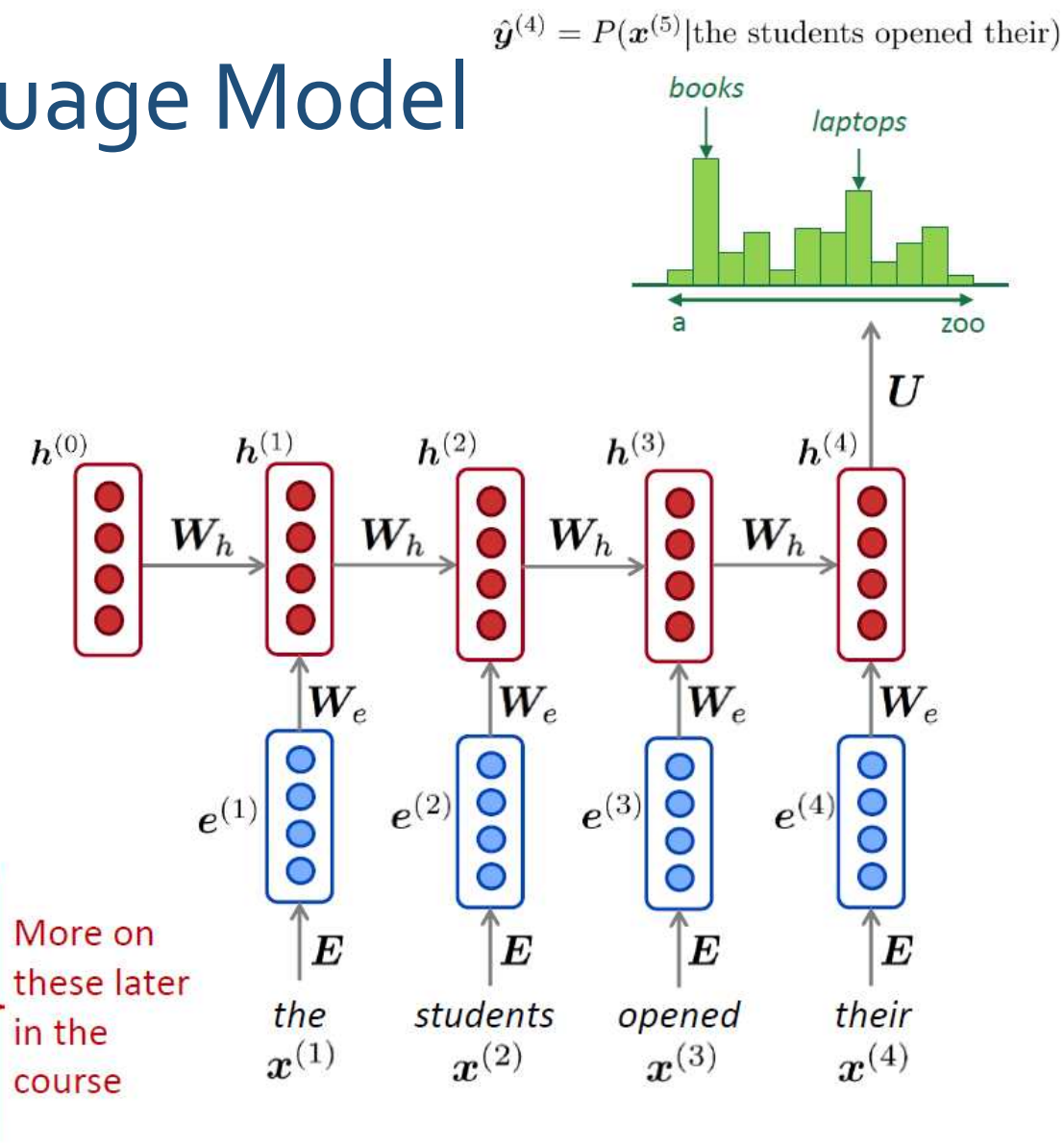
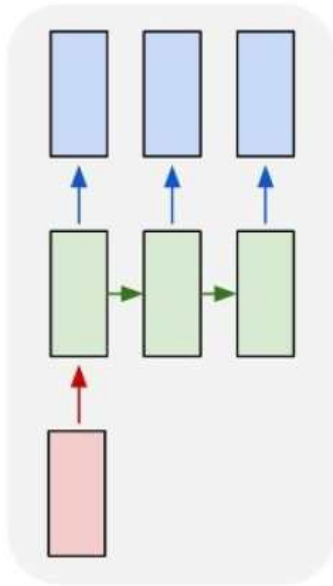# Recurrent Neural Network
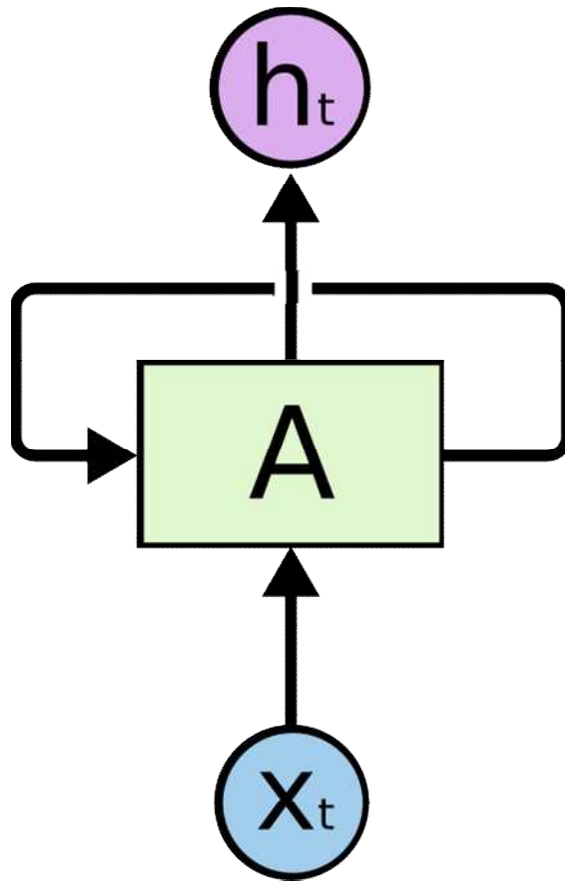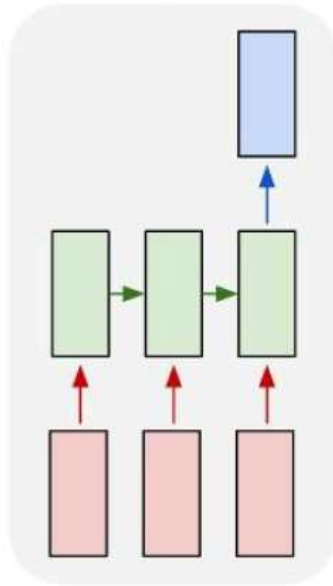
Recurrent Neural Network

RNN's flexibility
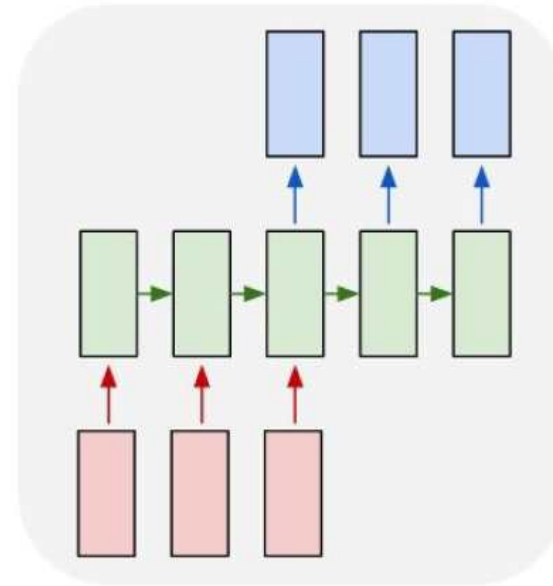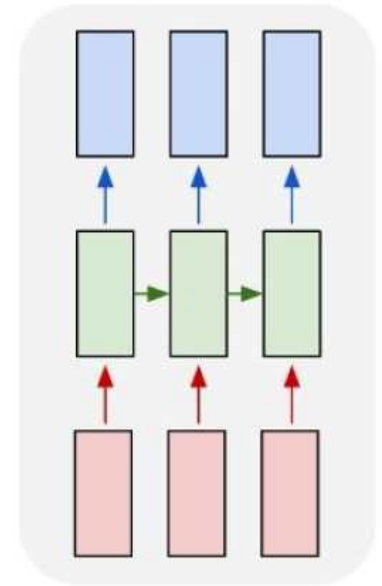

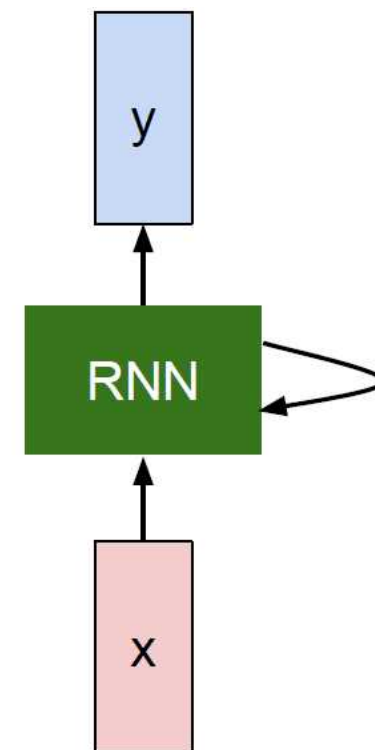one to one   one to many   many to one   many to many   many to many
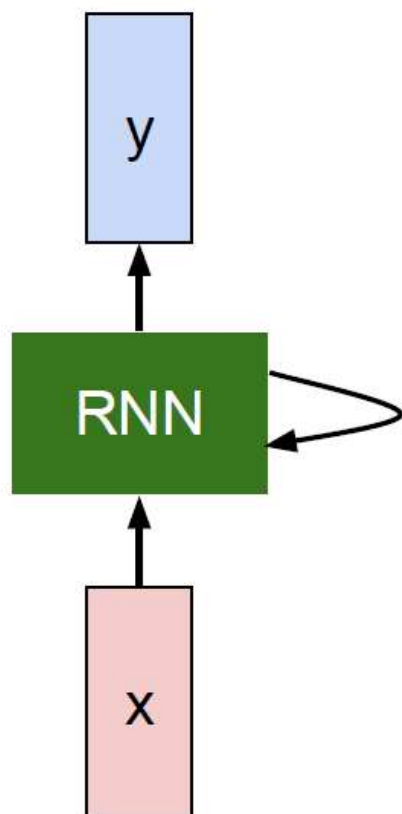
Vanilla Neural Networks

Recurrent Neural Network

We can process a sequence of vectors **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
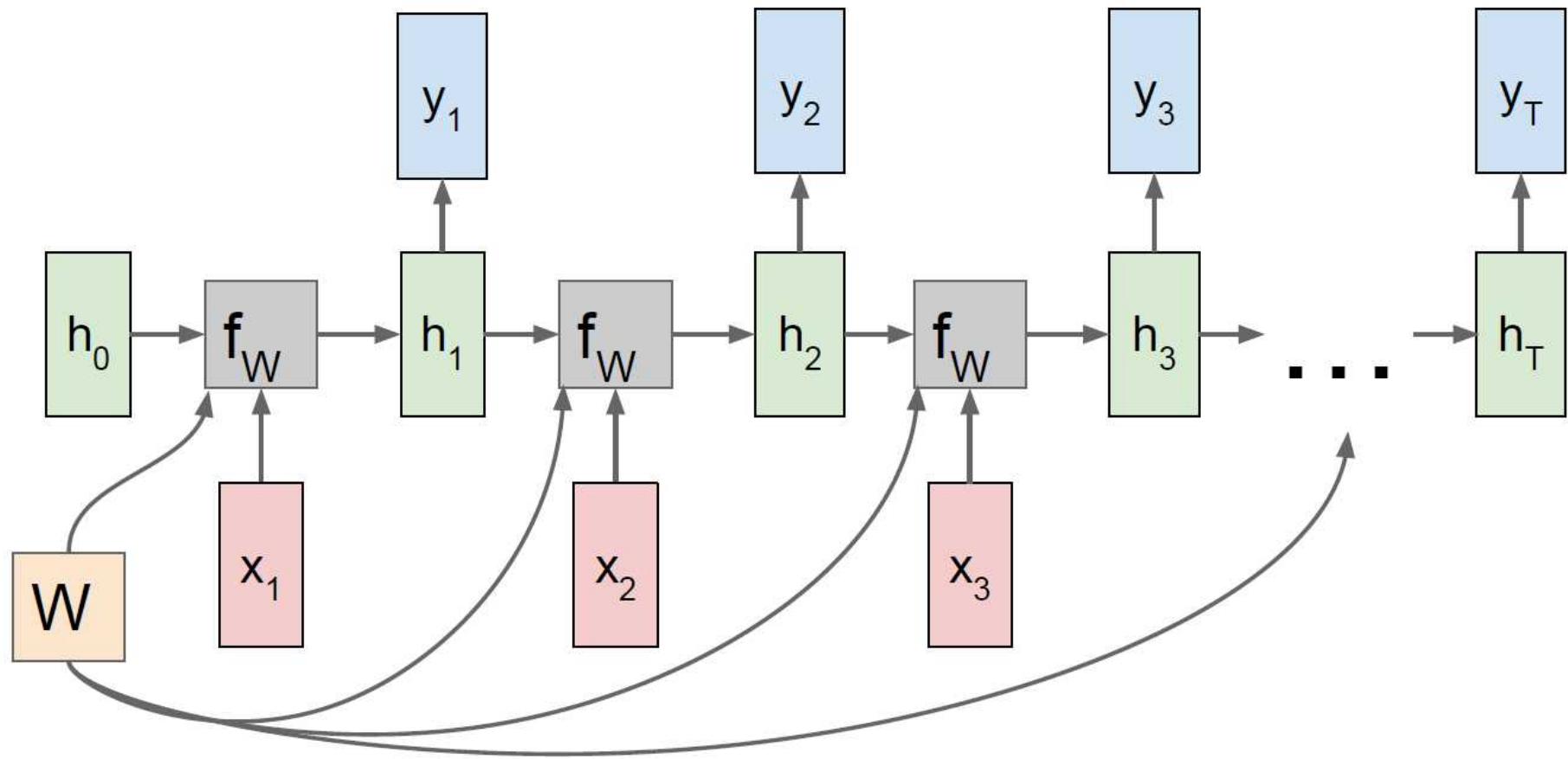of parameters are used at every time step.

y

RNN

x

Recurrent Neural Network

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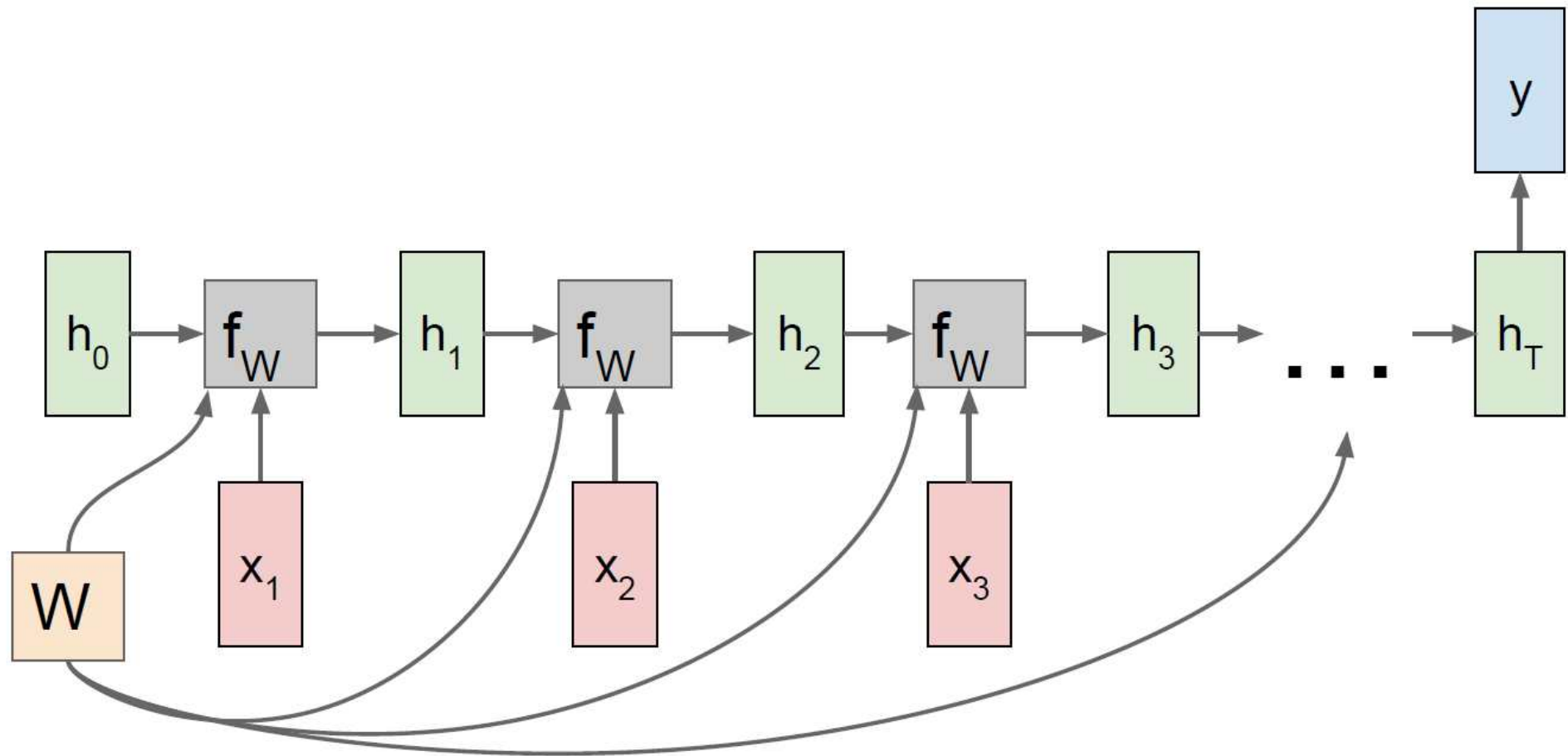
$$\downarrow$$

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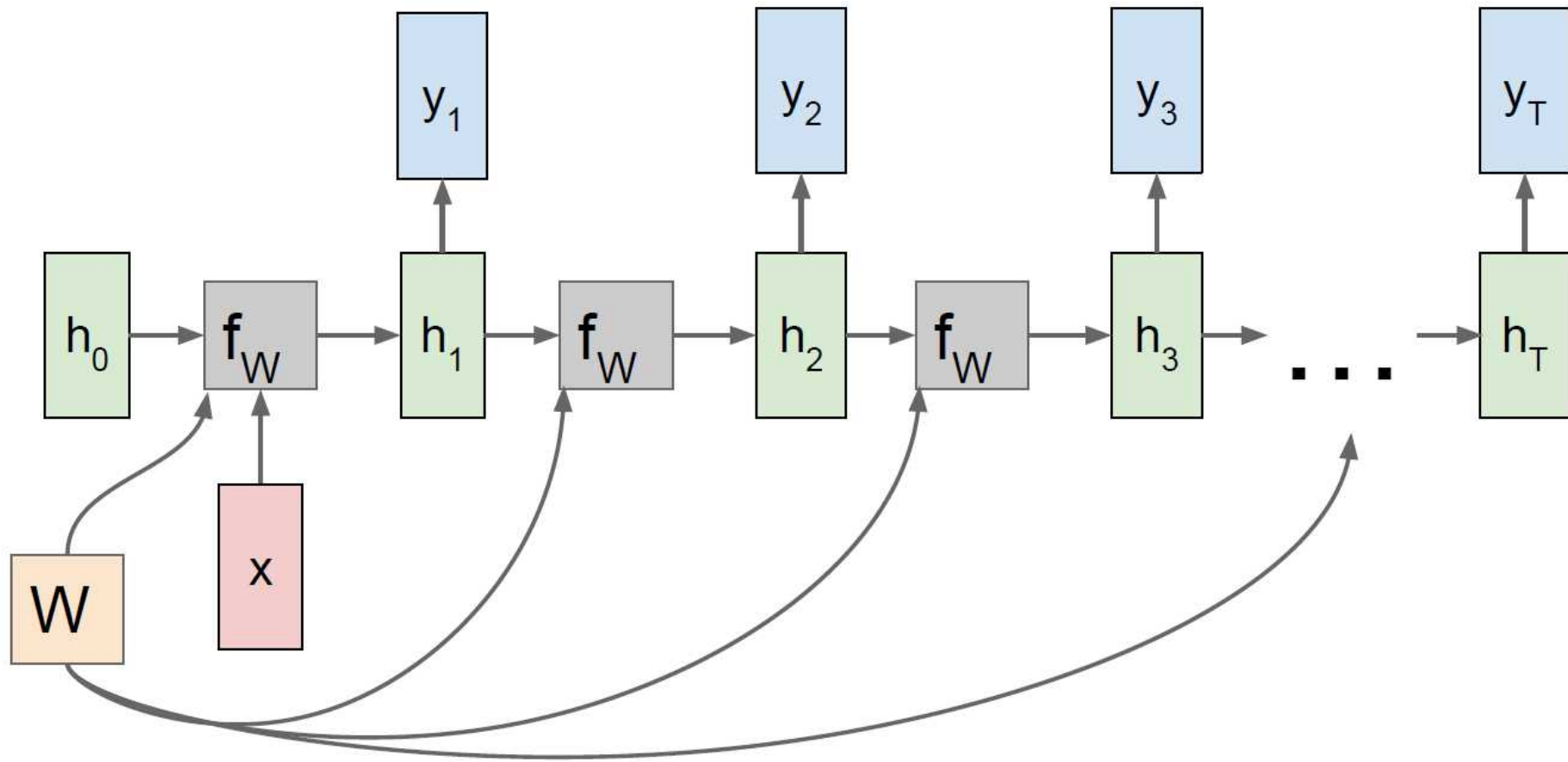
$$y_t = W_{hy}h_t$$

RNN: Computational Graph – Many to Many

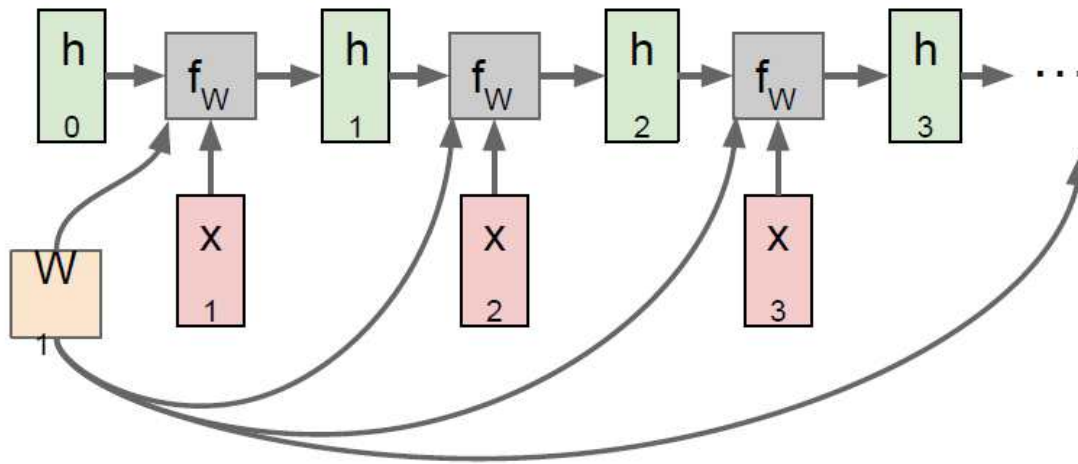# RNN: Computational Graph – Many to One
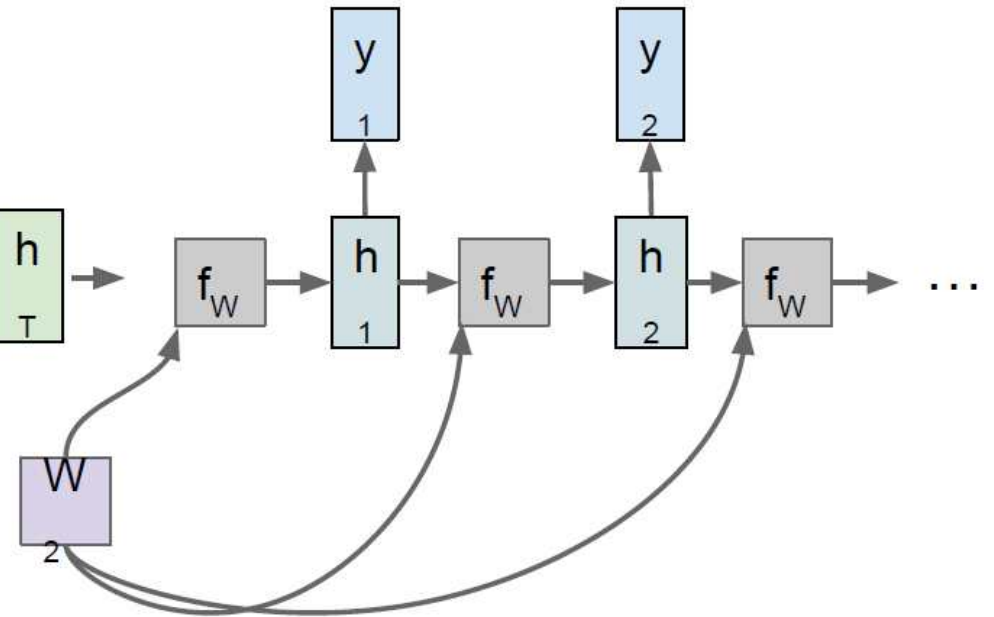
RNN: Computational Graph – One to Many

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
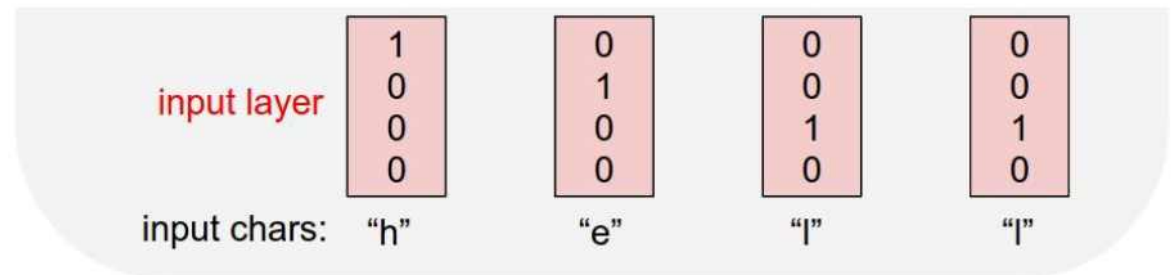
Character-level language model example

Vocabulary:
[h,e,l,o]

Example training sequence:
"hello"

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$
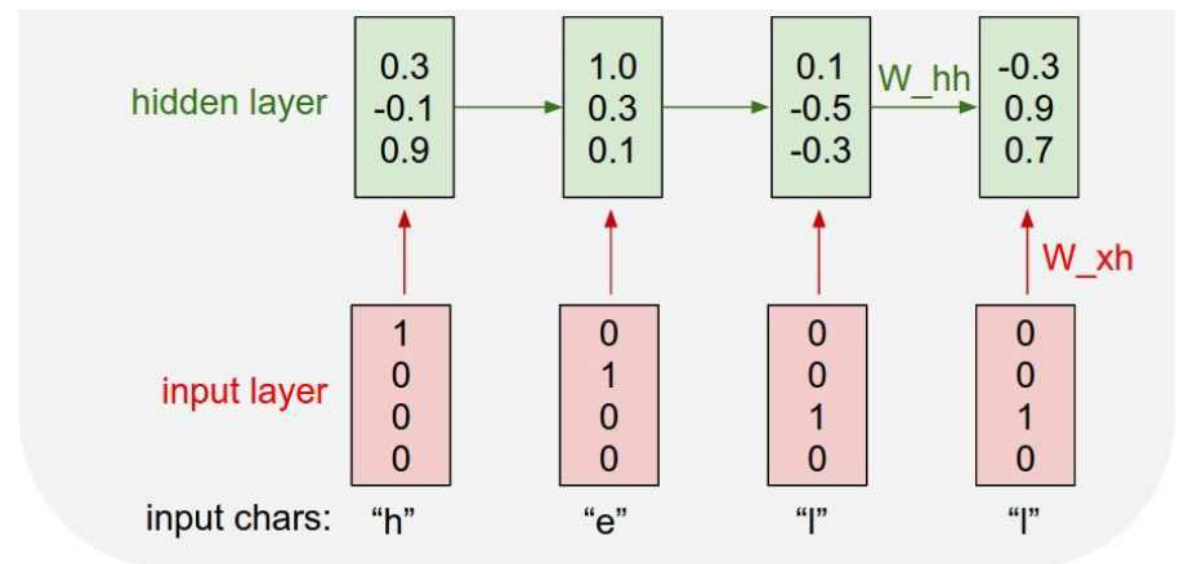
hidden layer	0.3 -0.1 0.9	1.0 0.3 0.1	0.1 -0.5 -0.3	W_hh	-0.3 0.9 0.7

W_xh

| input layer | 1 0 0 0 | 0 1 0 0 | 0 0 1 0 | 0 0 1 0 |

input chars: "h"   "e"   "l"   "l"

RNN

Character-level language model example

Vocabulary: [h,e,l,o]

Example training sequence: **"hello"**
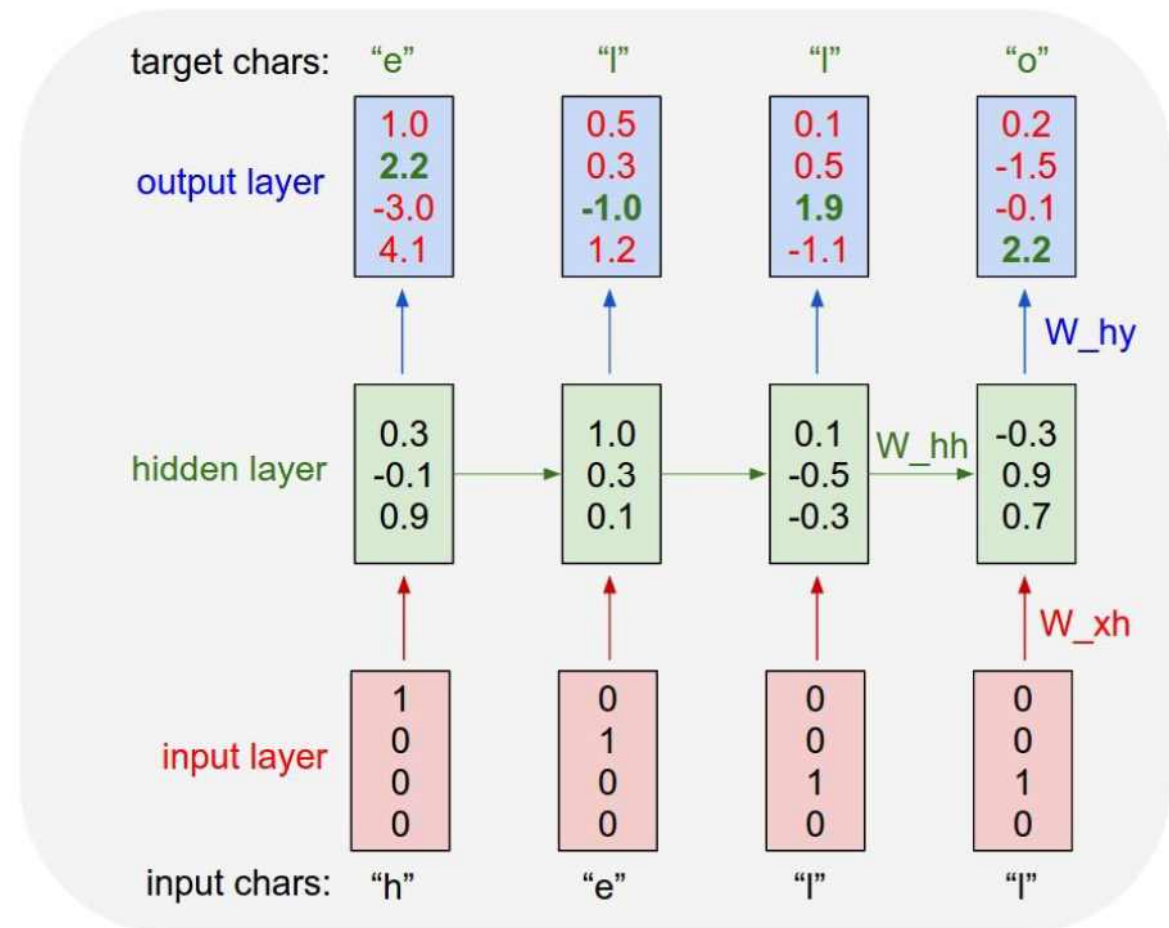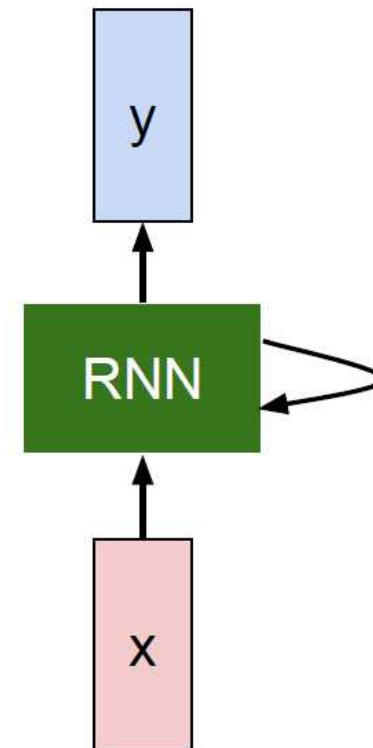
Language Model

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

train more

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

train more

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

train more

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

Language Model

PANDARUS:
Alas, I think he shall be come approached and the day
When little srain 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 C be a set of the construction.*
  Let 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 $\mathcal{F} \to \mathcal{F}$ 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

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

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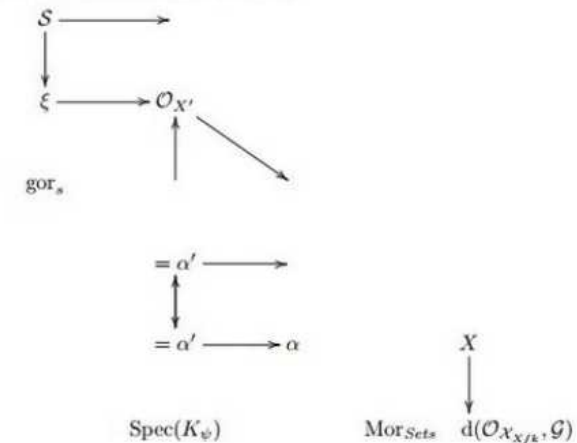
is a limit. Then $\mathcal{G}$ is a finite type and assume S is a flat and $\mathcal{F}$ and $\mathcal{G}$ is a finite type f_*. This is of finite type diagrams, and

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

□

Proof. We have see that $X = \mathrm{Spec}(R)$ 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

$$\mathcal{O}_{X,x} \longrightarrow \mathcal{F}_{\overline{x}} \ \ -1(\mathcal{O}_{X_{\acute{e}tale}}) \longrightarrow \mathcal{O}_{X_{\acute{e}}}^{-1}\mathcal{O}_{X_\lambda}(\mathcal{O}_{X_\eta}^{\overline{v}})$$

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

```c
#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

"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:

line length tracking cell

Searching for Interpretable Cells

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;
}
```
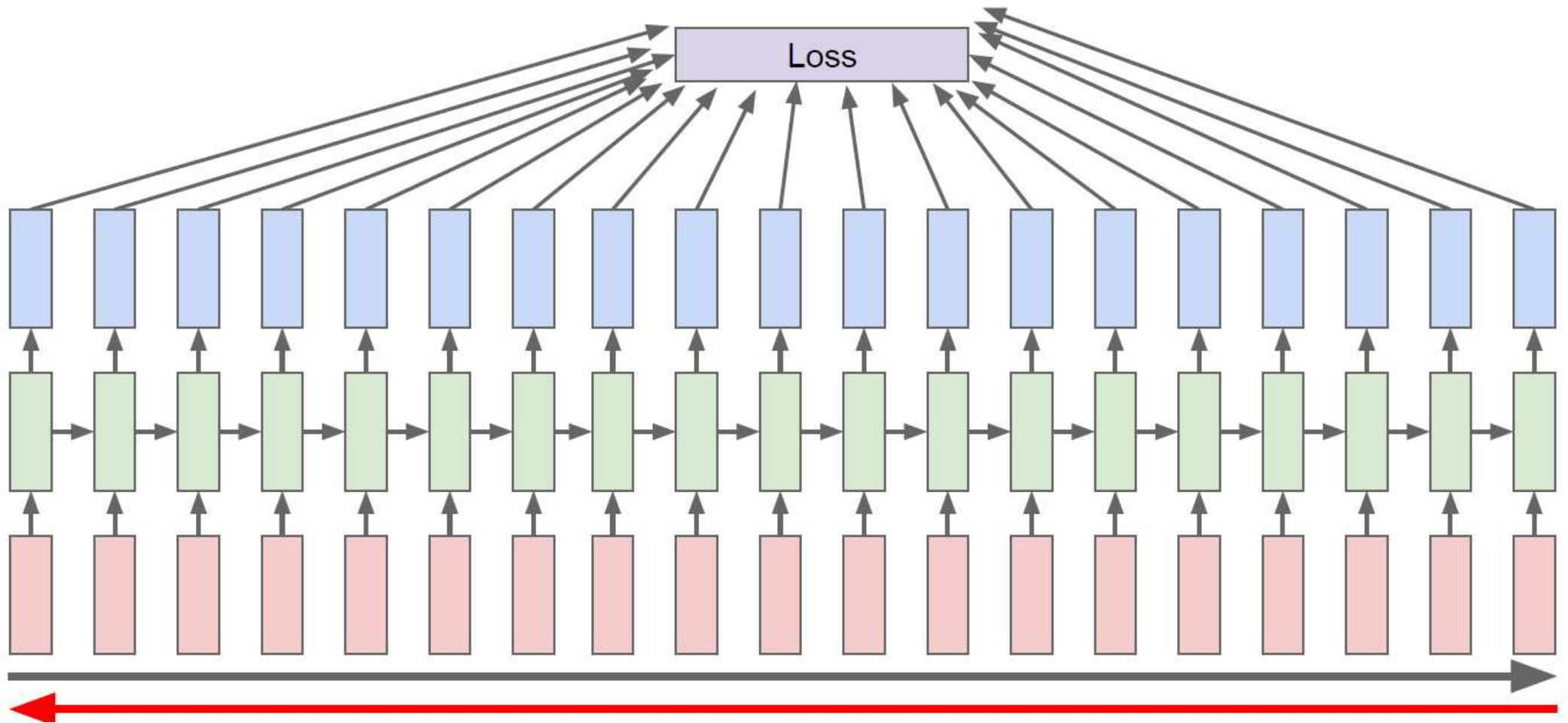
code depth cell

Is Long Term Dependency Problem Resolved?

The writer of the books ___

is

are
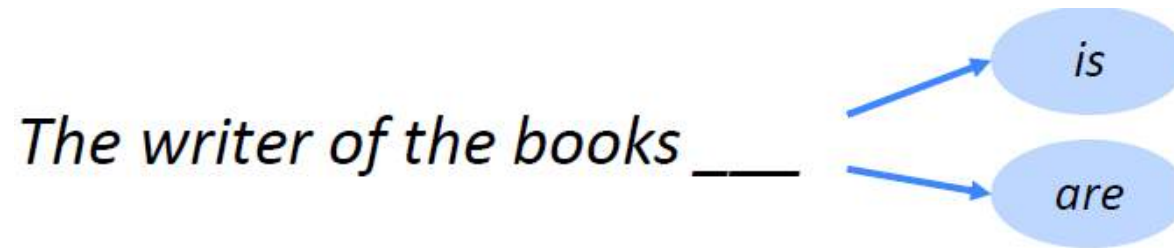
- RNN language model can choose "are".

- Why? How can we train RNN?

Back Propagation Through Time

Backpropagation of RNNs
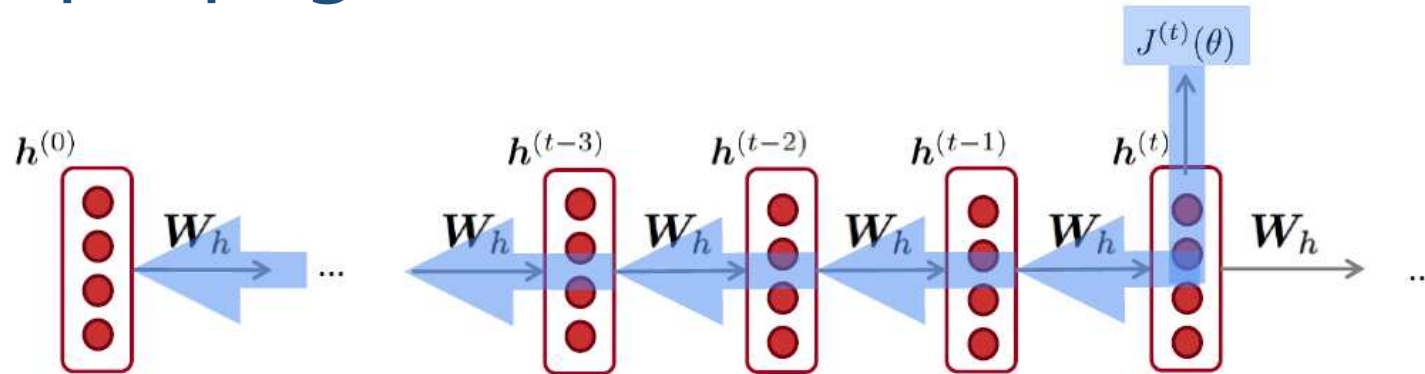
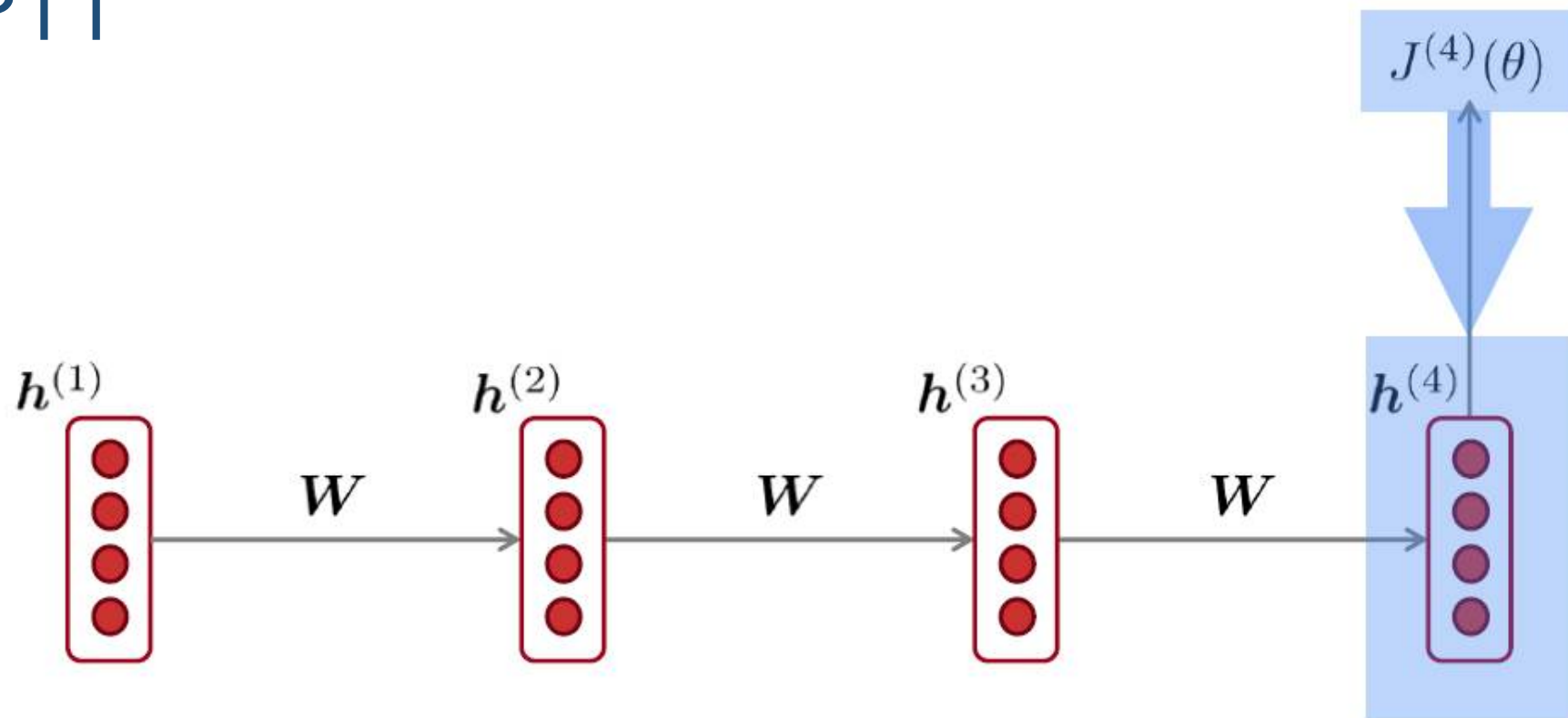$$\frac{\partial J^{(t)}}{\partial \boldsymbol{W_h}} = \left. \sum_{i=1}^{t} \frac{\partial J^{(t)}}{\partial \boldsymbol{W_h}} \right|_{(i)}$$

Question: How do we calculate this?

Answer: Backpropagate over timesteps $i=t,...,0$, summing gradients as you go.
This algorithm is called **"backpropagation through time"**

BPTT

$$J^{(4)}(\theta)$$

$\boldsymbol{h}^{(1)}$

$\boldsymbol{h}^{(2)}$

$\boldsymbol{h}^{(3)}$

$\boldsymbol{h}^{(4)}$

$\boldsymbol{W}$ $\boldsymbol{W}$ $\boldsymbol{W}$

$$\frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(1)}} = \ ?$$

BPTT

$$J^{(4)}(\theta)$$

$$\boldsymbol{h}^{(1)} \qquad \boldsymbol{h}^{(2)} \qquad \boldsymbol{h}^{(3)} \qquad \boldsymbol{h}^{(4)}$$

$$\boldsymbol{W} \qquad \boldsymbol{W} \qquad \boldsymbol{W}$$
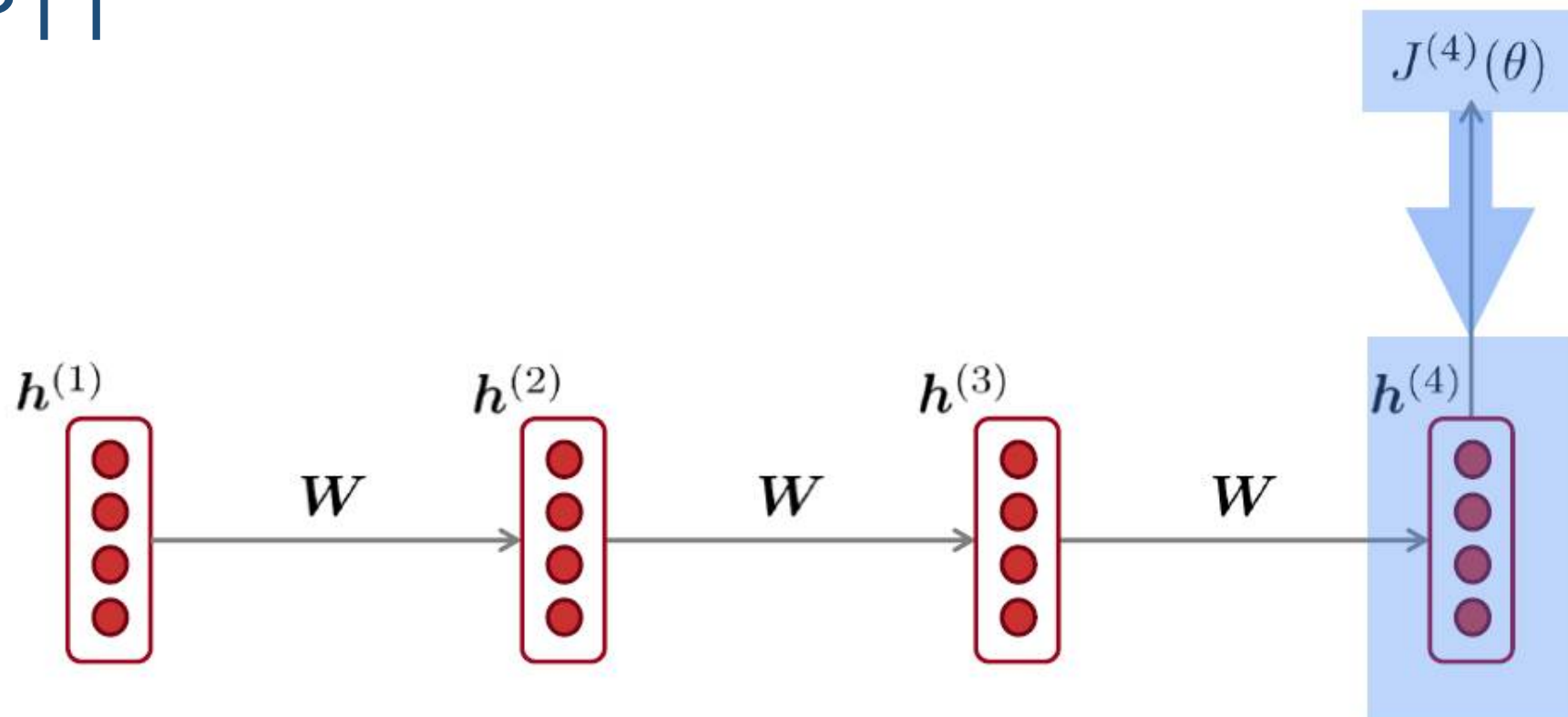
$$\frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(1)}} = \qquad\qquad \frac{\partial \boldsymbol{h}^{(4)}}{\partial \boldsymbol{h}^{(3)}} \times \frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(4)}}$$

chain rule!

BPTT

$$J^{(4)}(\theta)$$

$$\boldsymbol{h}^{(1)} \qquad \boldsymbol{h}^{(2)} \qquad \boldsymbol{h}^{(3)} \qquad \boldsymbol{h}^{(4)}$$

$$\boldsymbol{W} \qquad \boldsymbol{W} \qquad \boldsymbol{W}$$

$$\frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(1)}} = \qquad\qquad \frac{\partial \boldsymbol{h}^{(3)}}{\partial \boldsymbol{h}^{(2)}} \times \qquad\qquad \frac{\partial \boldsymbol{h}^{(4)}}{\partial \boldsymbol{h}^{(3)}} \times \frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(4)}}$$

chain rule!
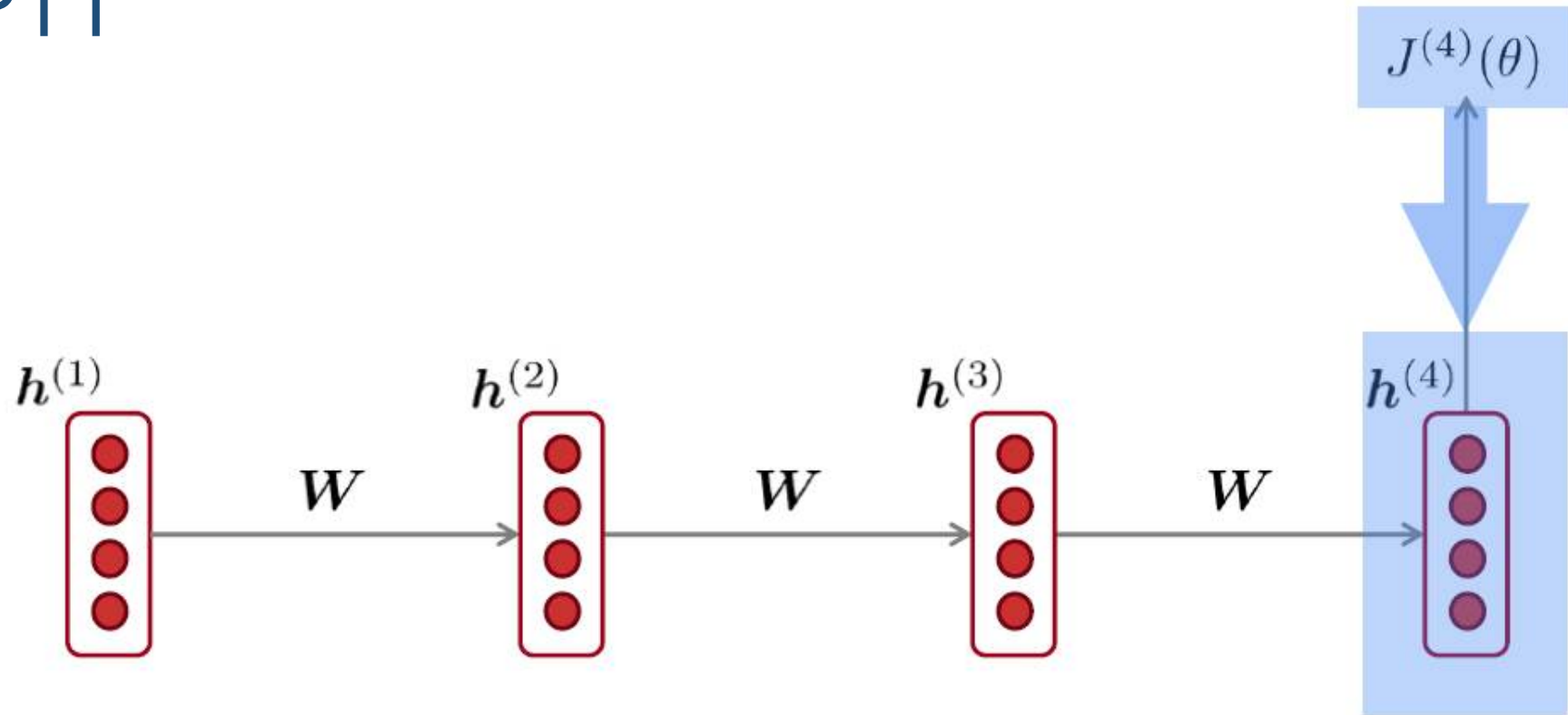
BPTT

$$\frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(1)}} = \quad \frac{\partial \boldsymbol{h}^{(2)}}{\partial \boldsymbol{h}^{(1)}} \times \qquad \frac{\partial \boldsymbol{h}^{(3)}}{\partial \boldsymbol{h}^{(2)}} \times \qquad \frac{\partial \boldsymbol{h}^{(4)}}{\partial \boldsymbol{h}^{(3)}} \times \frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(4)}}$$

chain rule!

BPTT
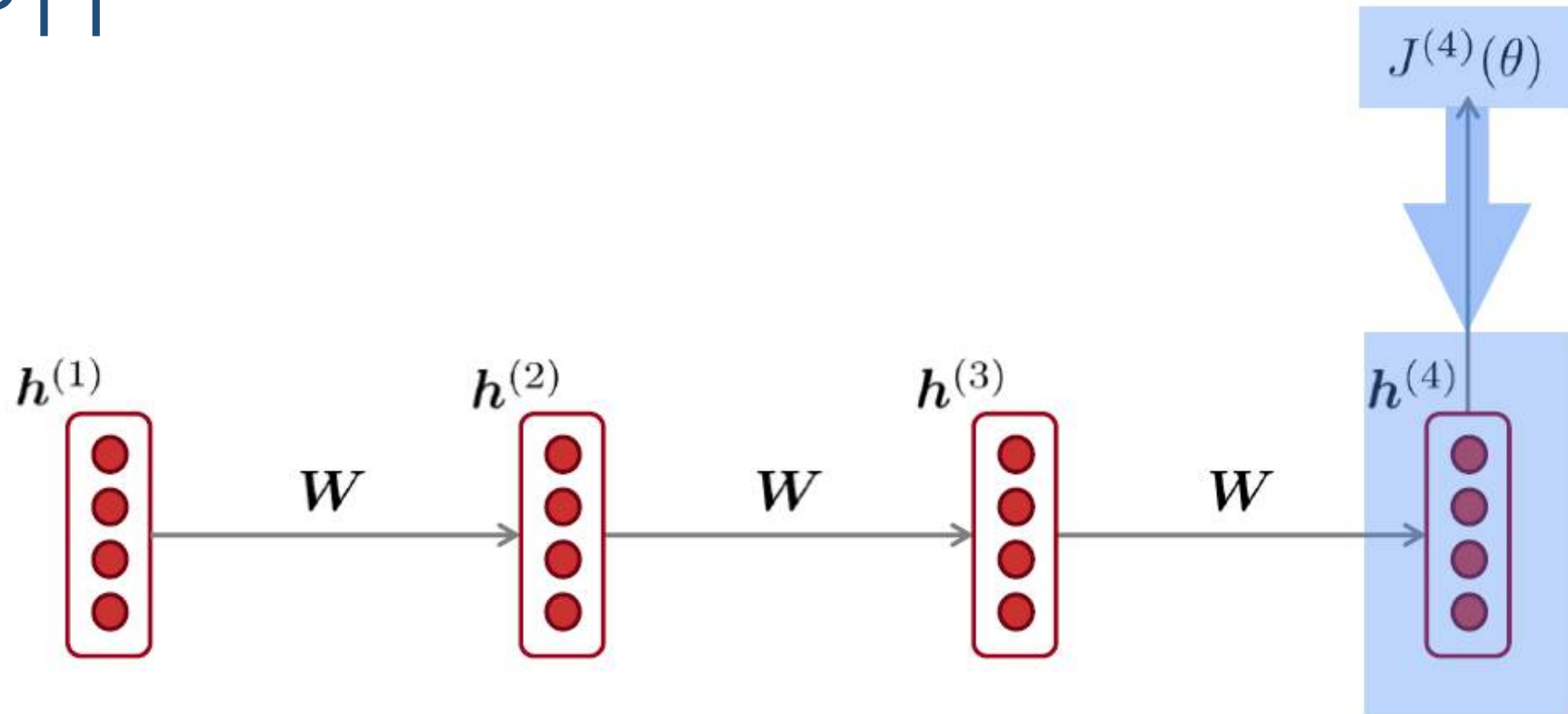
$$\frac{\partial J^{(4)}}{\partial h^{(1)}} = \boxed{\frac{\partial h^{(2)}}{\partial h^{(1)}}} \times \boxed{\frac{\partial h^{(3)}}{\partial h^{(2)}}} \times \boxed{\frac{\partial h^{(4)}}{\partial h^{(3)}}} \times \frac{\partial J^{(4)}}{\partial h^{(4)}}$$

What happens if these are smaller than 1 or larger than 1?

Vanishing/Exploding Gradient Problem

- $\dfrac{\partial J^{(4)}}{\partial h^{(1)}} = \dfrac{\partial J^{(4)}}{\partial h^{(4)}} \cdot \prod_{i=2}^{4} \dfrac{\partial h^{(t)}}{\partial h^{(t-1)}}$

- Recall: $h^{(t)} = \tanh\left(W_h h^{(t-1)} + W_x x^{(t)} + b\right)$

- $\dfrac{\partial h^{(t)}}{\partial h^{(t-1)}} = W_h \cdot \tanh'\left(W_h h^{(t-1)} + W_x x^{(t)} + b\right)$

- Therefore: $\dfrac{\partial J^{(4)}}{\partial h^{(1)}} = \dfrac{\partial J^{(4)}}{\partial h^{(4)}} \cdot W_h^{\,3} \cdot \tanh'^3$

- Generally, $\dfrac{\partial J^{(n)}}{\partial h^{(m)}} = \dfrac{\partial J^{(n)}}{\partial h^{(n)}} \cdot W_h^{\,(n-m)} \cdot \tanh'^{(n-m)}$

What happens if (n-m) is getting larger?

Vanishing/Exploding Gradient Problem

$$\frac{\partial J^{(n)}}{\partial h^{(m)}} = \frac{\partial J^{(n)}}{\partial h^{(n)}} \cdot W_h^{(n-m)} \cdot \tanh'^{(n-m)}$$

- tanh' is always less than equal to 1 → vanishing gradient
- If the largest eigenvalue of W_h < 1 → vanishing gradient
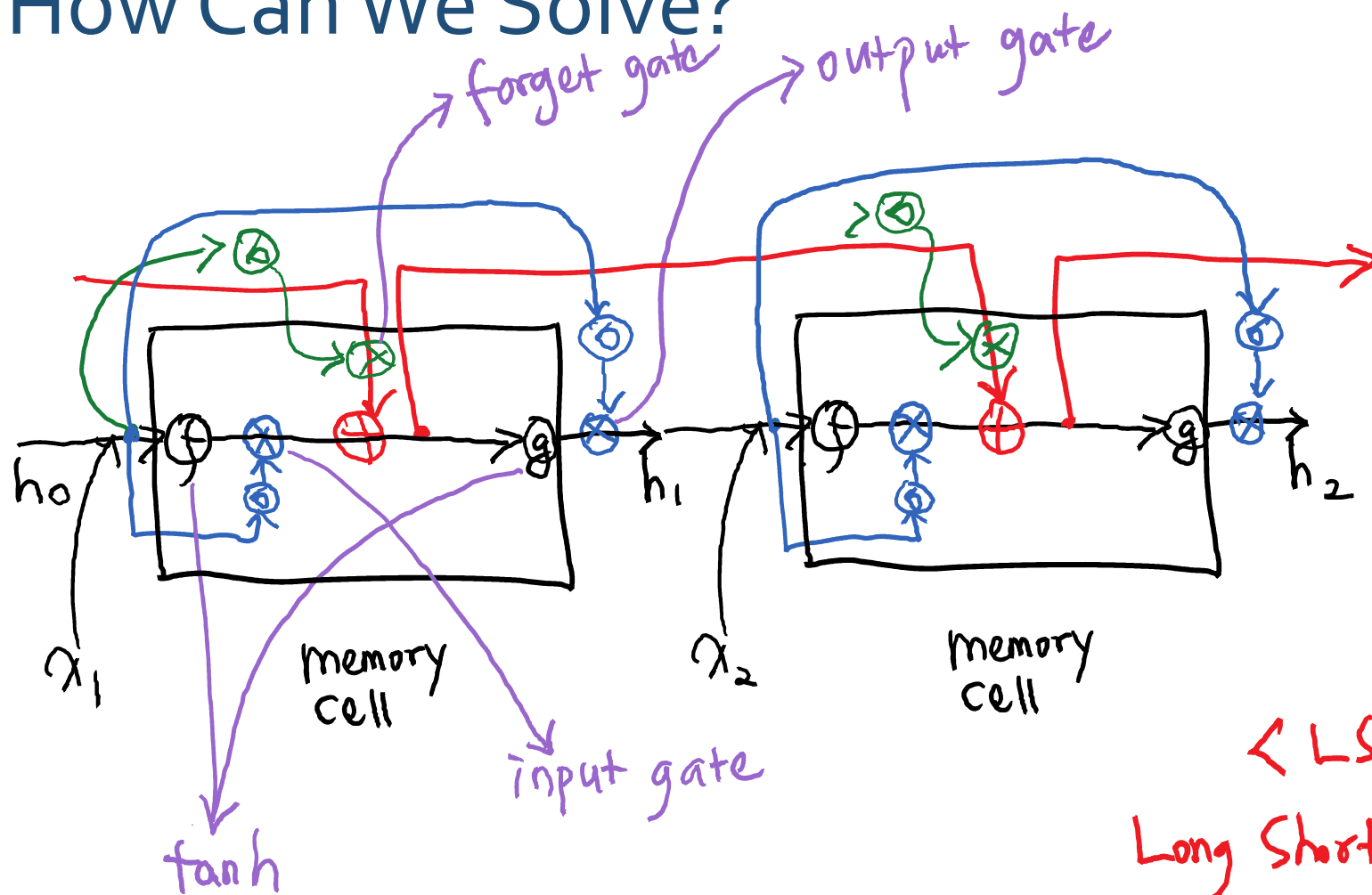- If the largest eigenvalue of W_h > 1 → exploding gradient

How Can We Solve?

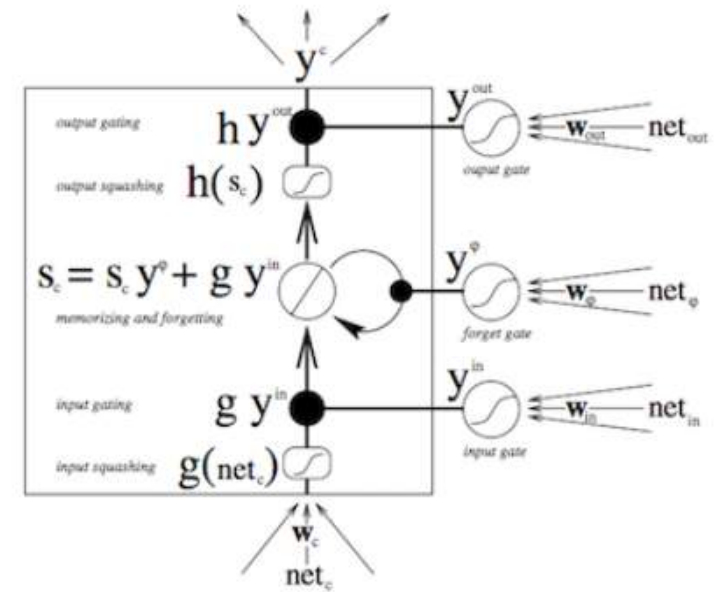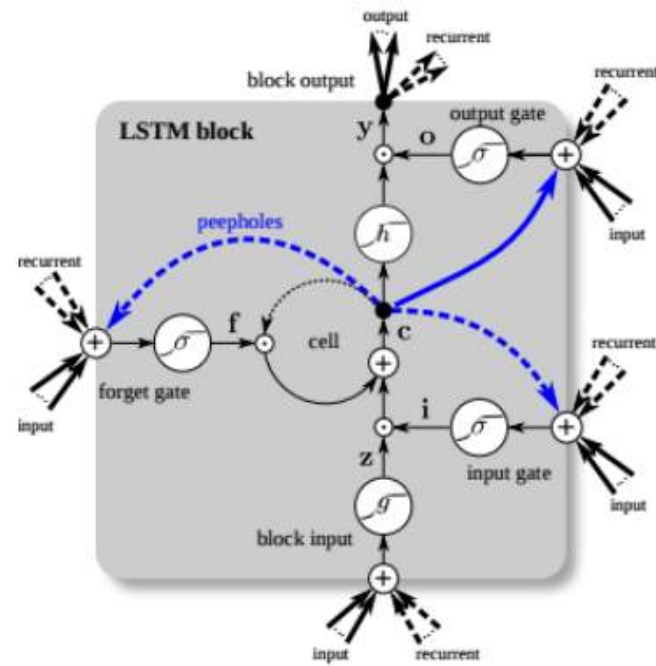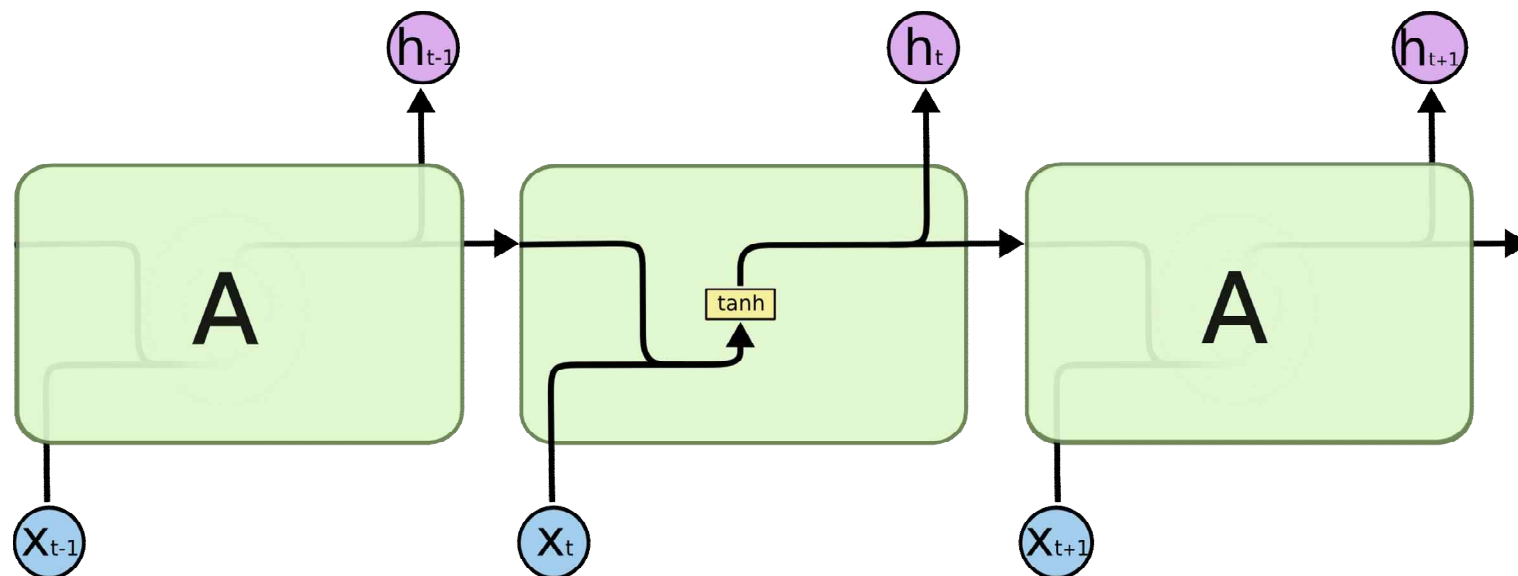# How Can We Solve?

How Can We Solve?

How Can We Solve?

How Can We Solve?

forget gate

output gate

h_0

x_1

memory cell

h_1

x_2

memory cell

h_2

input gate

tanh
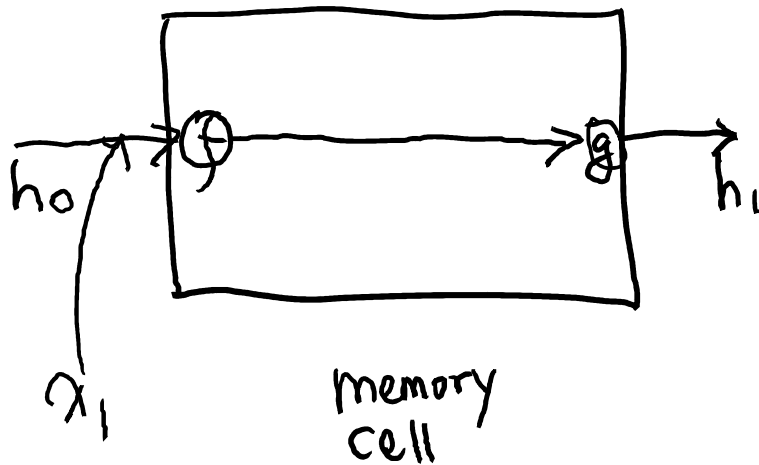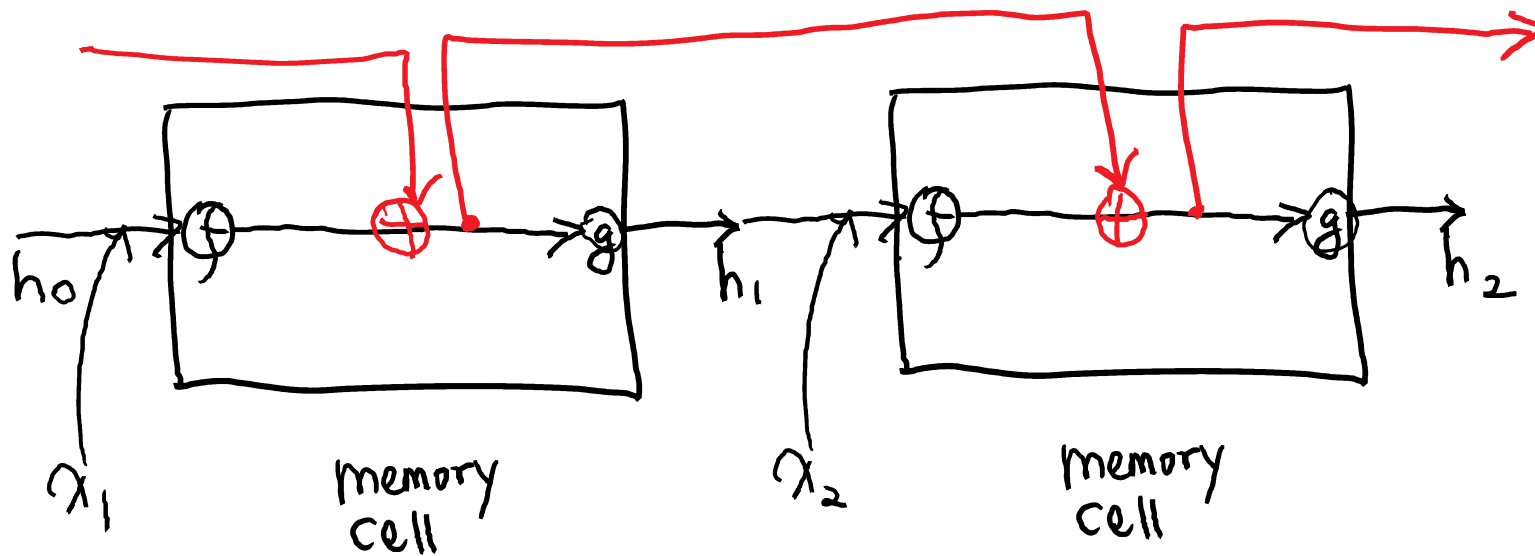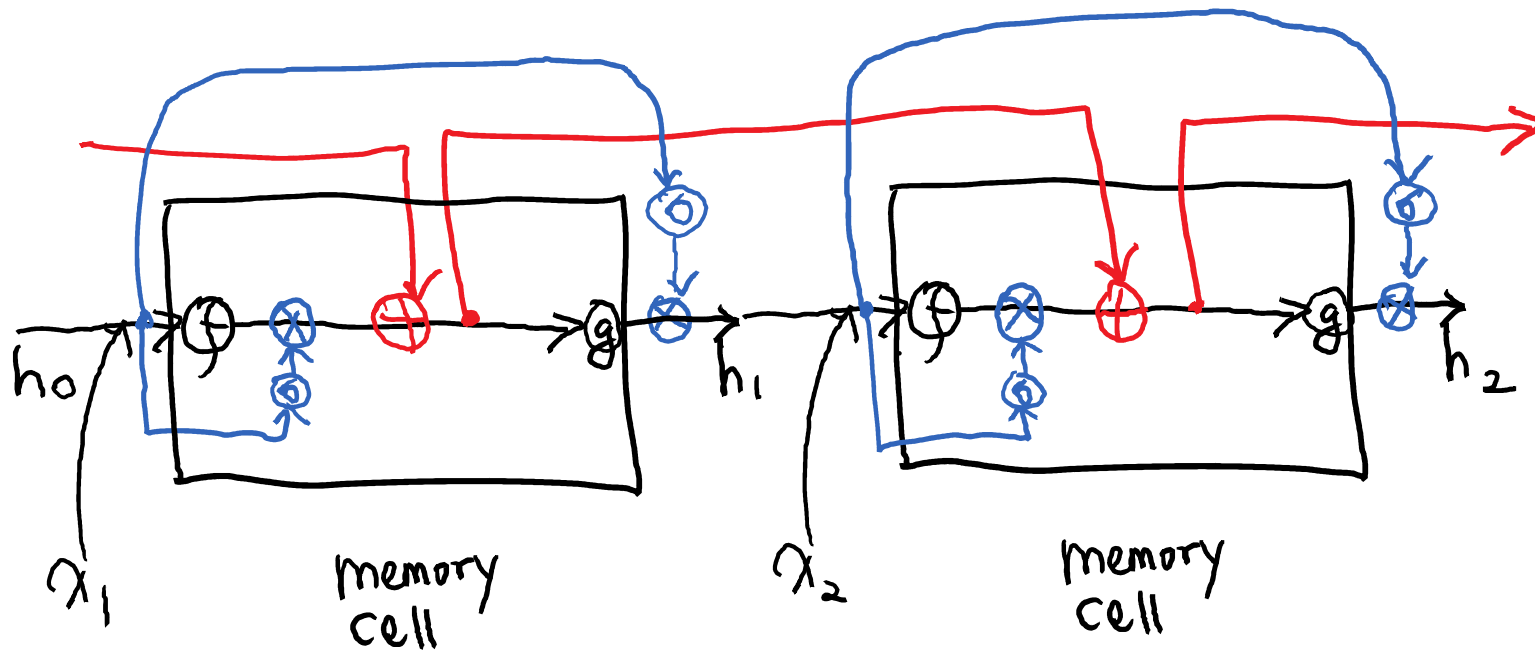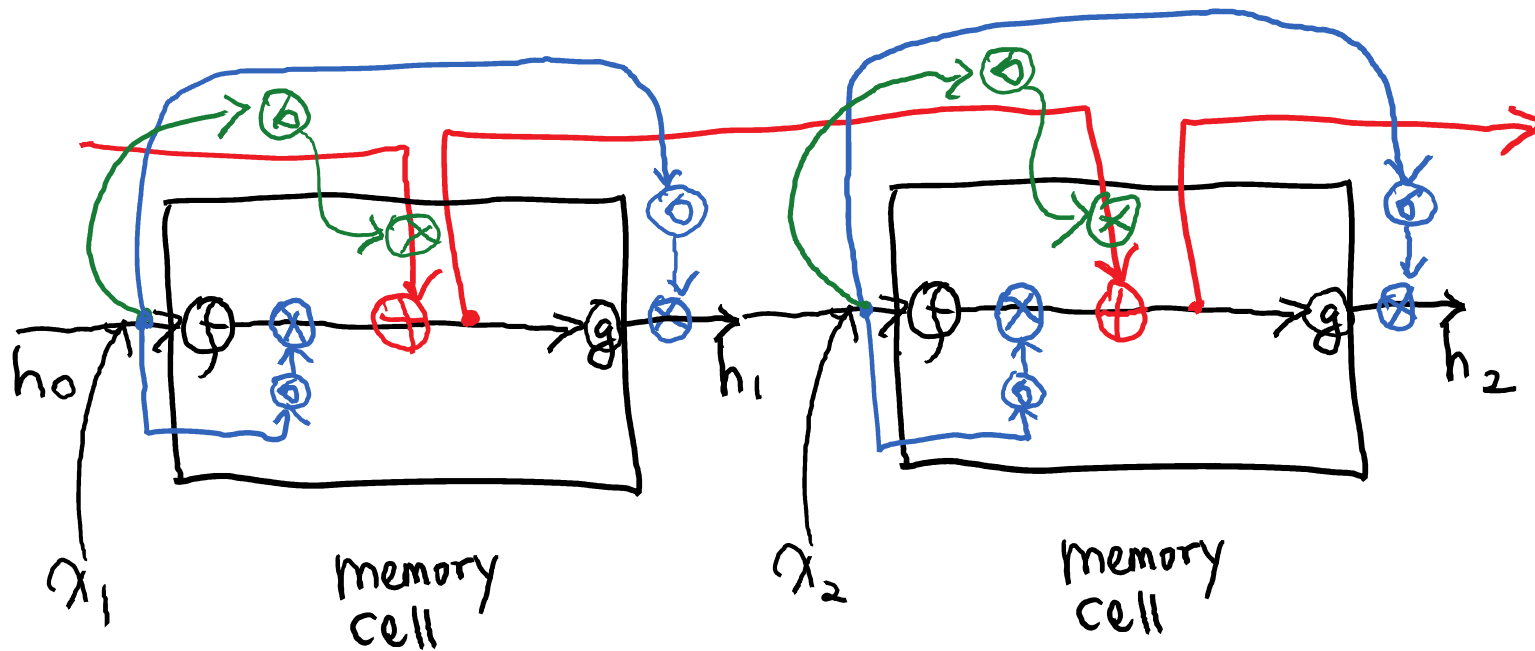
< LSTM >

Long Short Term Memory
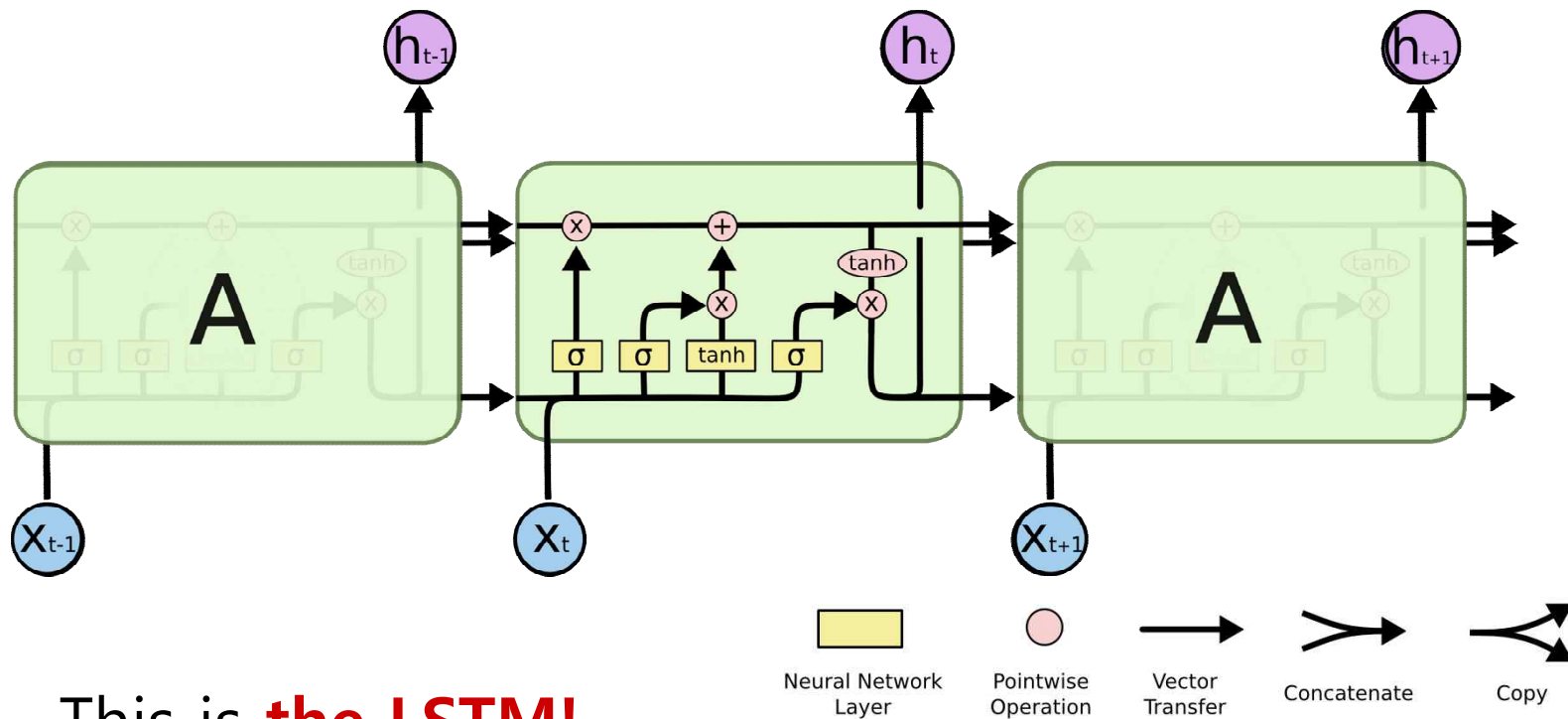
Long Short Term Memory
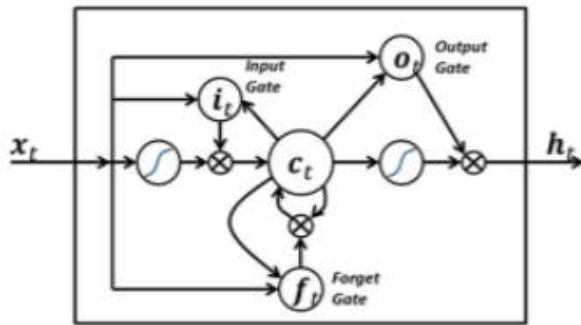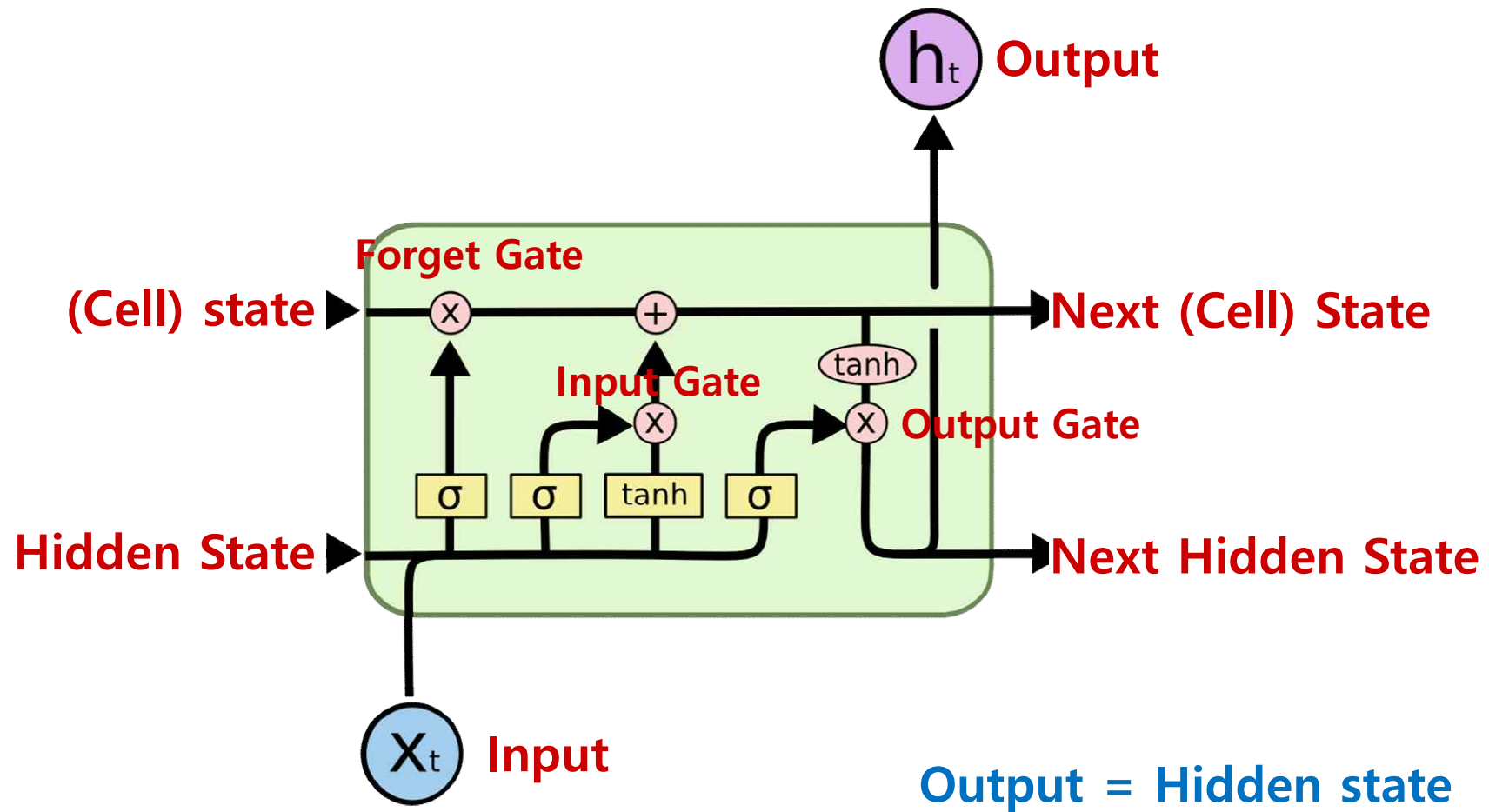
LSTM

Long Short Term Memory

This is just a standard RNN.

LSTM

Long Short Term Memory

This is **the LSTM!**

Overall Architecture
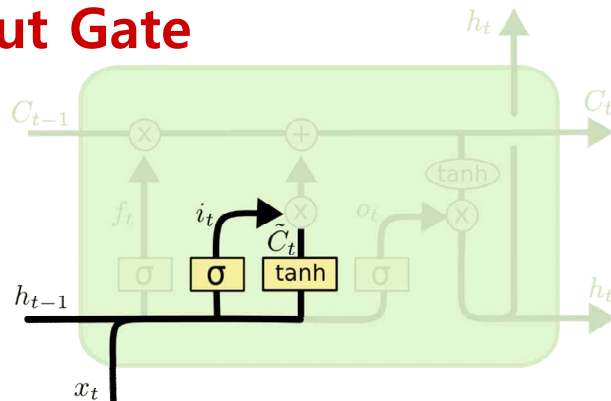
Forget Gate & Input Gate

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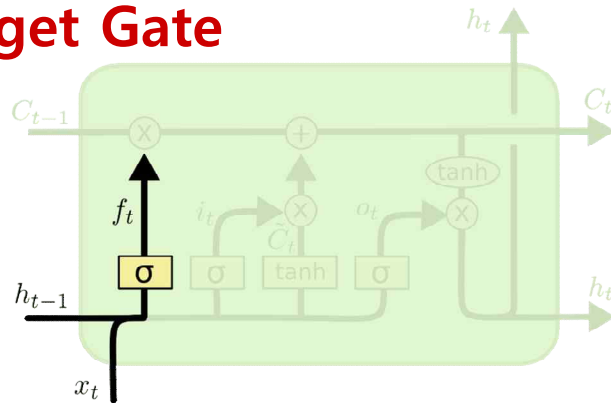
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

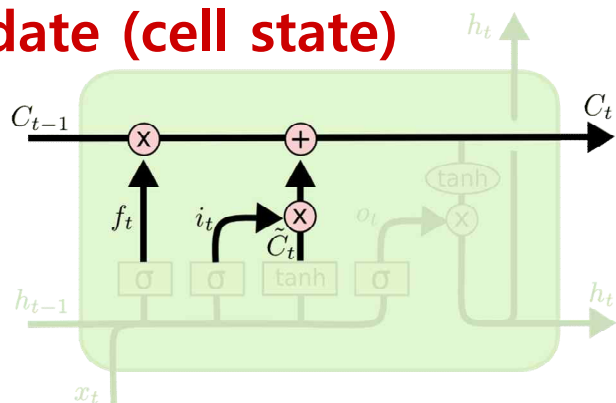Decide what information we're going to **throw away** from the cell state.

Input Gate

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

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

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

Update Cell State & Output Gate
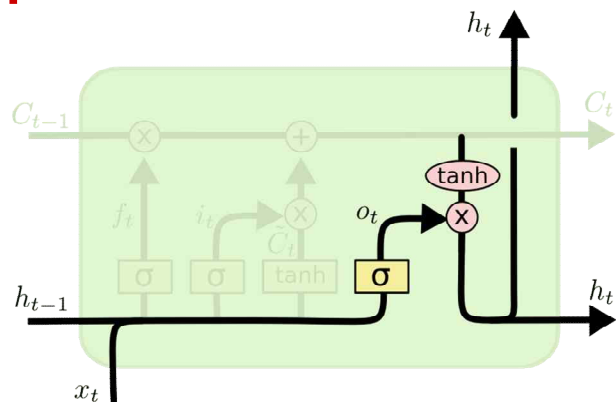
Update (cell state)

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

Update, scaled by how much we decide to update

: input_gate*curr_state + forget_gate*prev_state

Output Gate (hidden state)
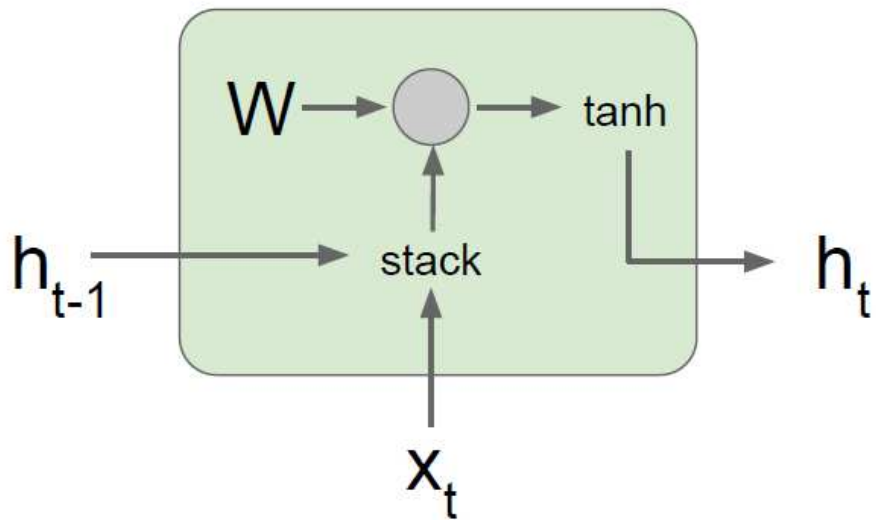

$$o_t = \sigma \left(W_o \left[h_{t-1}, x_t \right] + b_o \right)$$

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

Output based on the updated state

: output_gate*updated_state

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Is LSTM free from Vanishing Gradient Problems?

Vanilla RNN Gradient Flow

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

$$= \tanh\left((W_{hh} \quad W_{hx}) \binom{h_{t-1}}{x_t}\right)$$
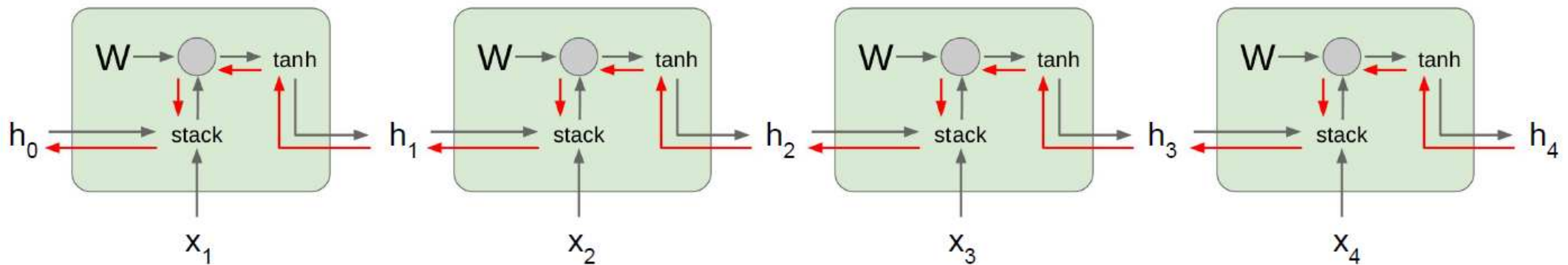
$$= \tanh\left(W \binom{h_{t-1}}{x_t}\right)$$

Vanilla RNN Gradient Flow

Backpropagation from h_t to h_{t-1} multiplies by W (actually W_{hh}^T)
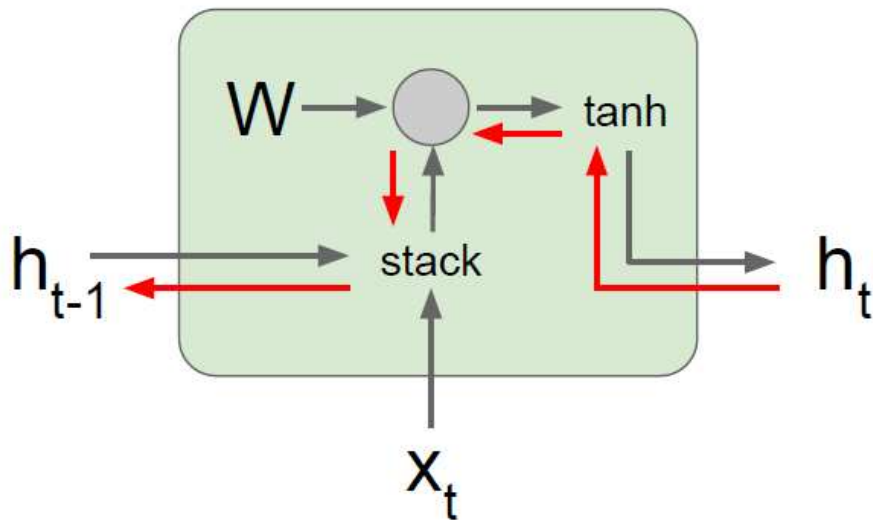
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

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

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

Vanilla RNN Gradient Flow

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

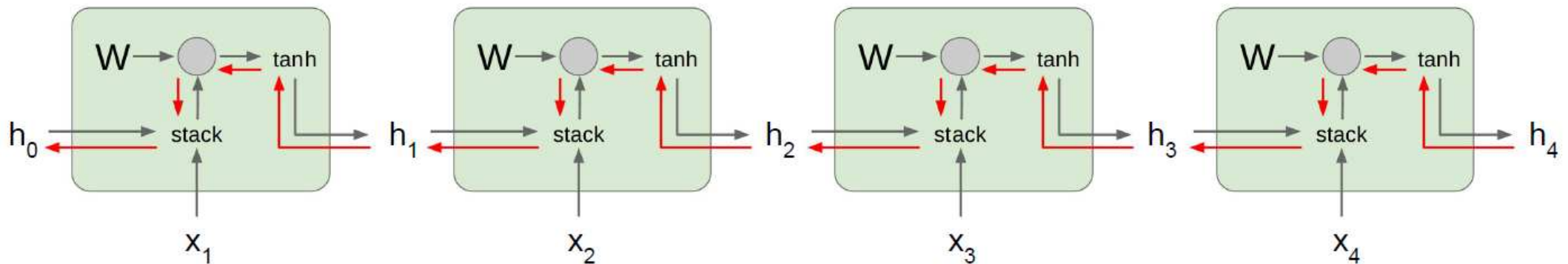Largest singular value > 1:
Exploding gradients

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
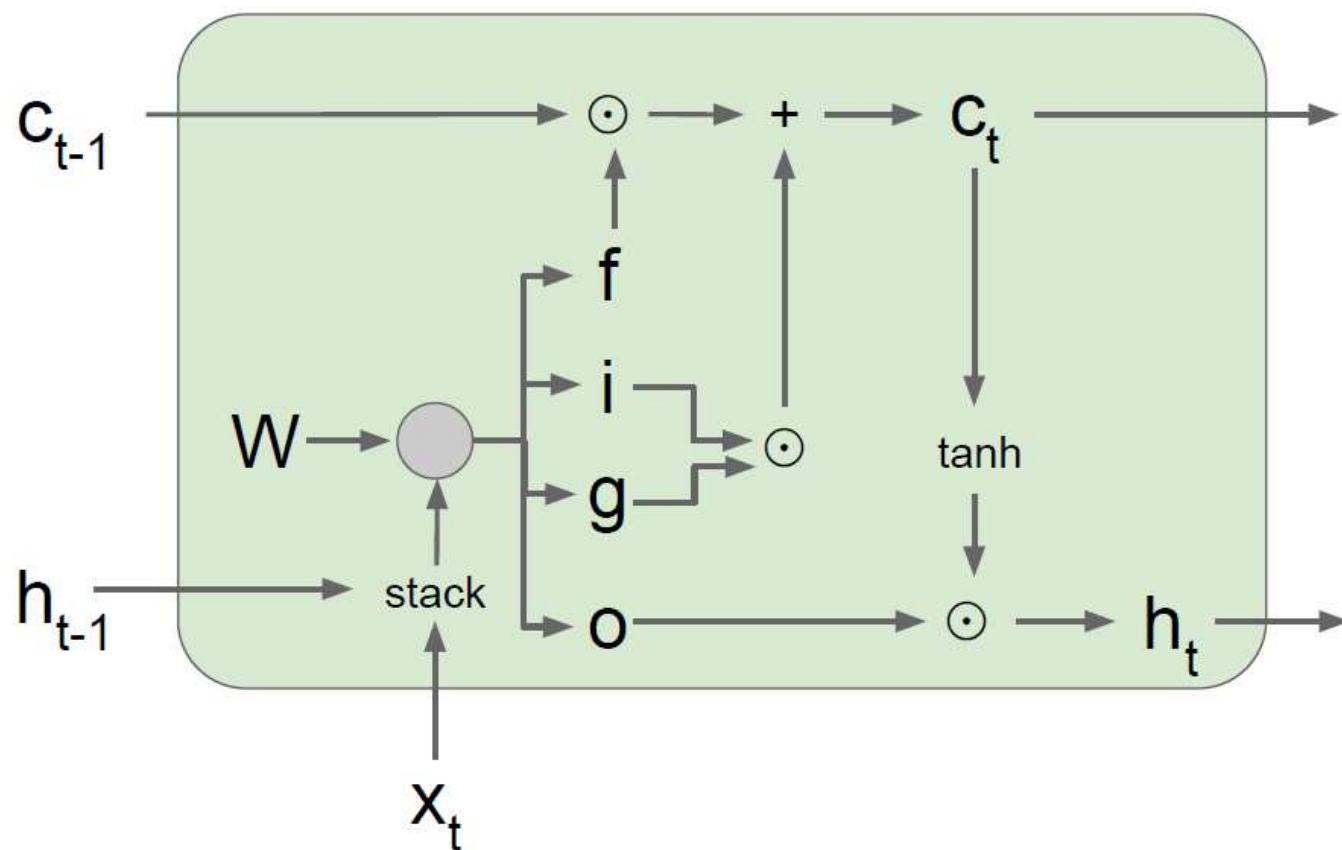
- Gradient가 흘러갈 때 같은 숫자(W, tanh의 미분)가 계속 곱해짐

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
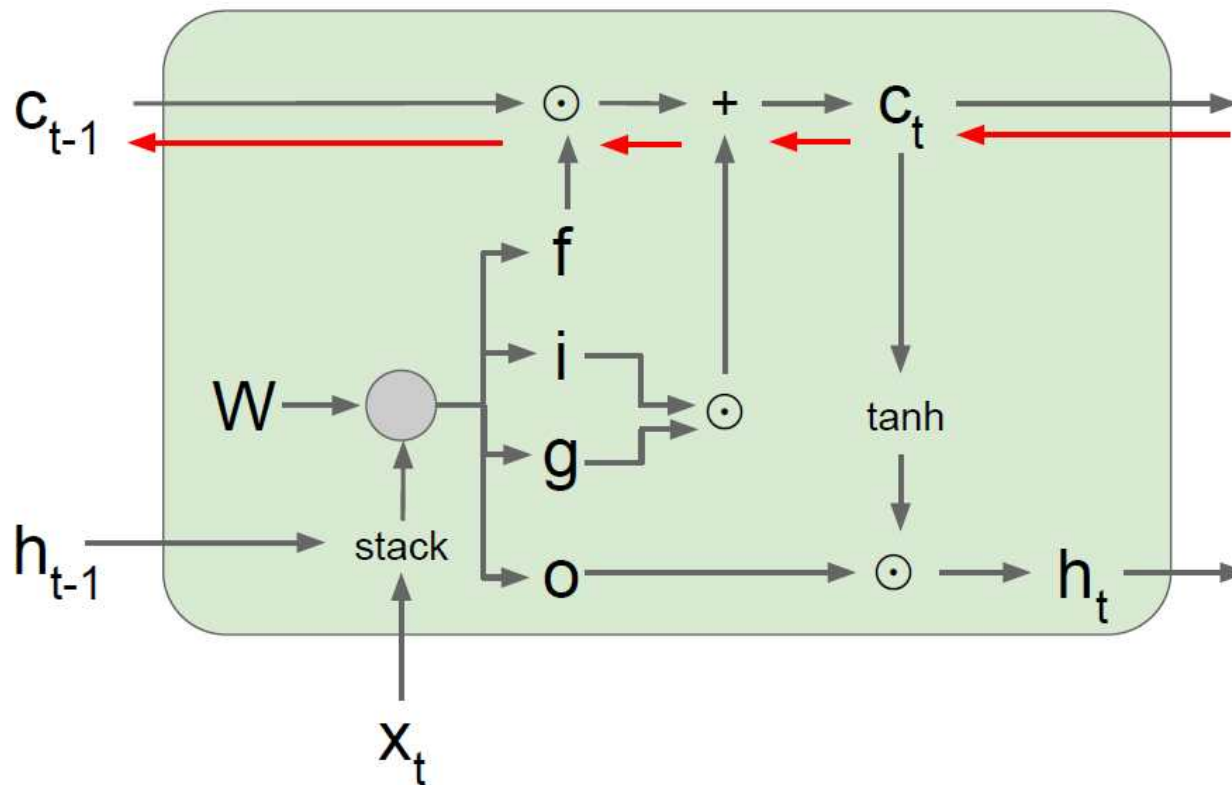
LSTM Gradient Flow

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

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

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

LSTM Gradient Flow

- Gradient가 흘러갈 때 forget gate이 값이 계속 곱해짐(매번 다른값)

Backpropagation from c_t to c_{t-1} only elementwise multiplication by f, no matrix multiply by W
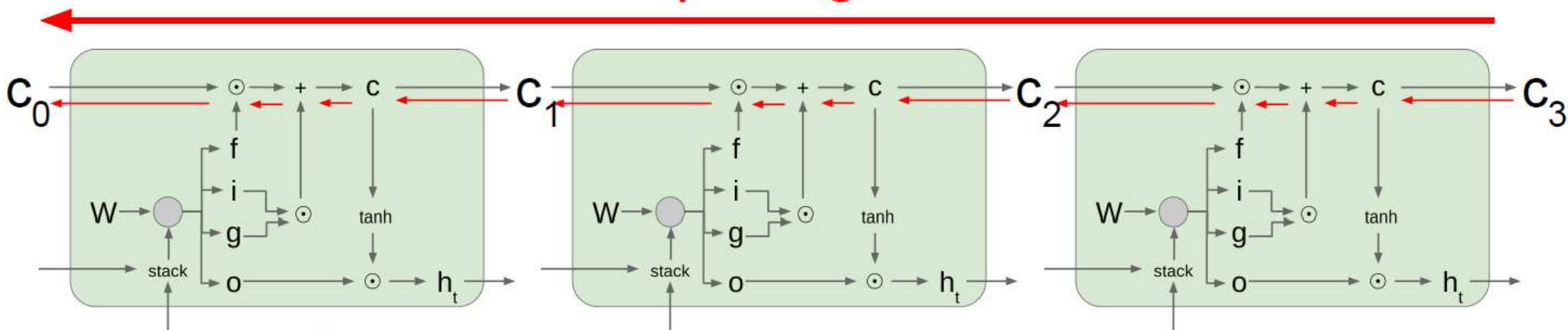
$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$
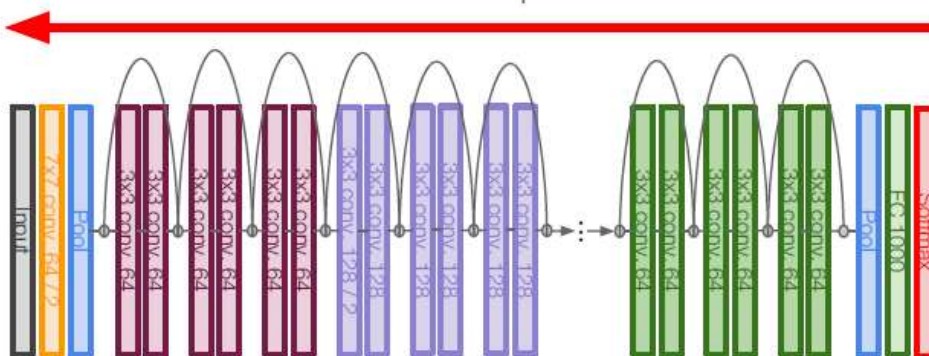
$$c_t = f \odot c_{t-1} + i \odot g$$
$$h_t = o \odot \tanh(c_t)$$

LSTM Gradient Flow

Uninterrupted gradient flow!

Similar to ResNet!
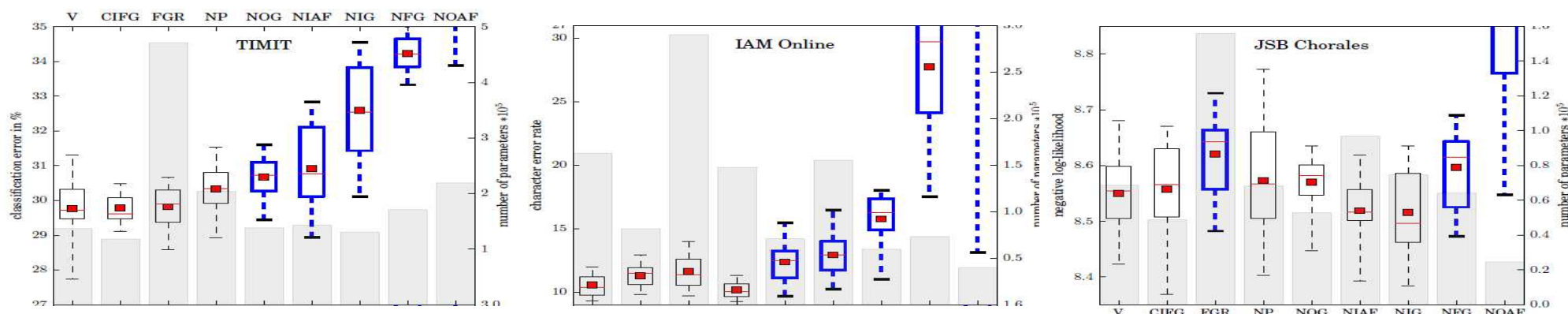
LSTM's Problems?

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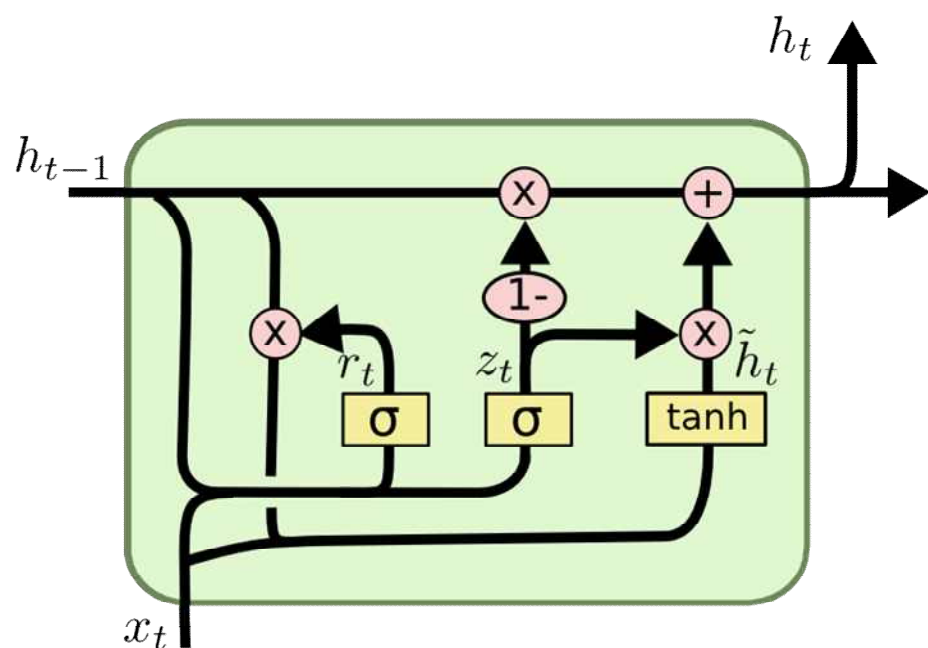
$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

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

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

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