CS 231 Lecture 10

Fei-Fei Li & Justin Johnson & Serena teung*(2017 Stanford)*

CS 231 Study week 5

Presenter : 허환

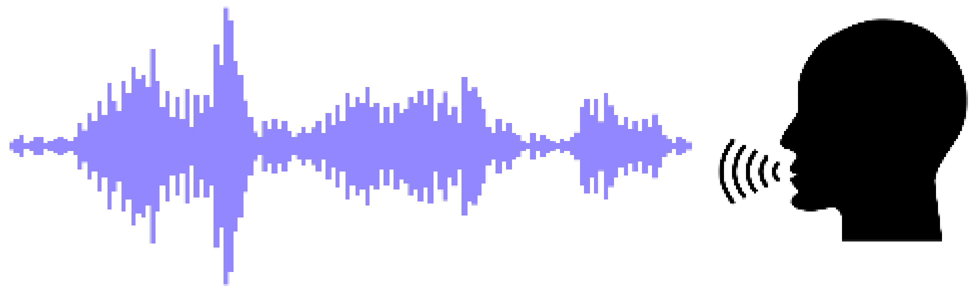
Table of content

1. Introduction

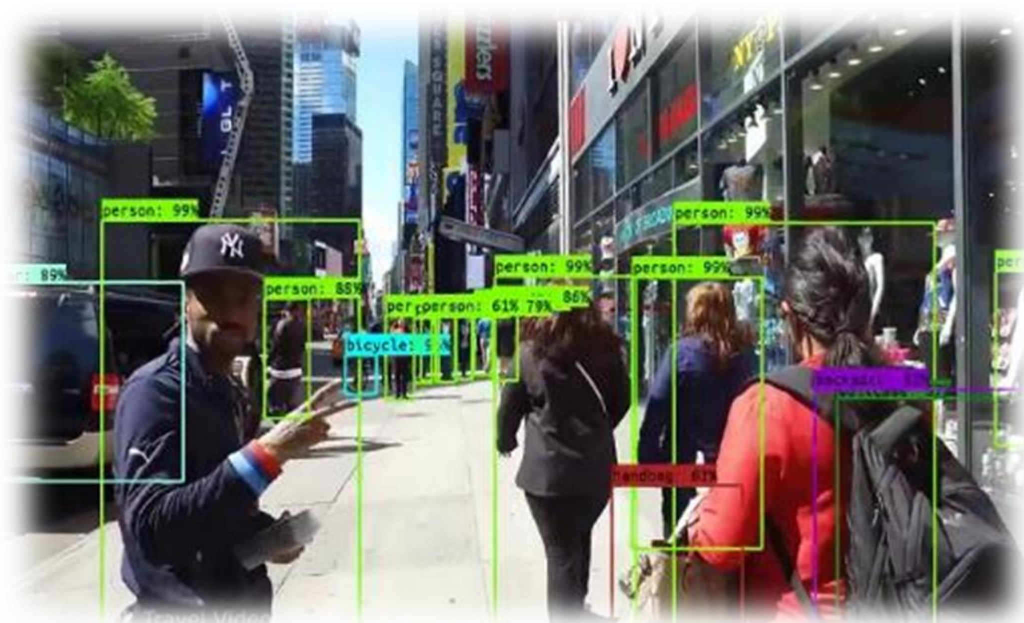
- 1.1 CNN’s area
- 1.2 CNN’s limitation

1.1 CNN’s Area

- Speech recognition

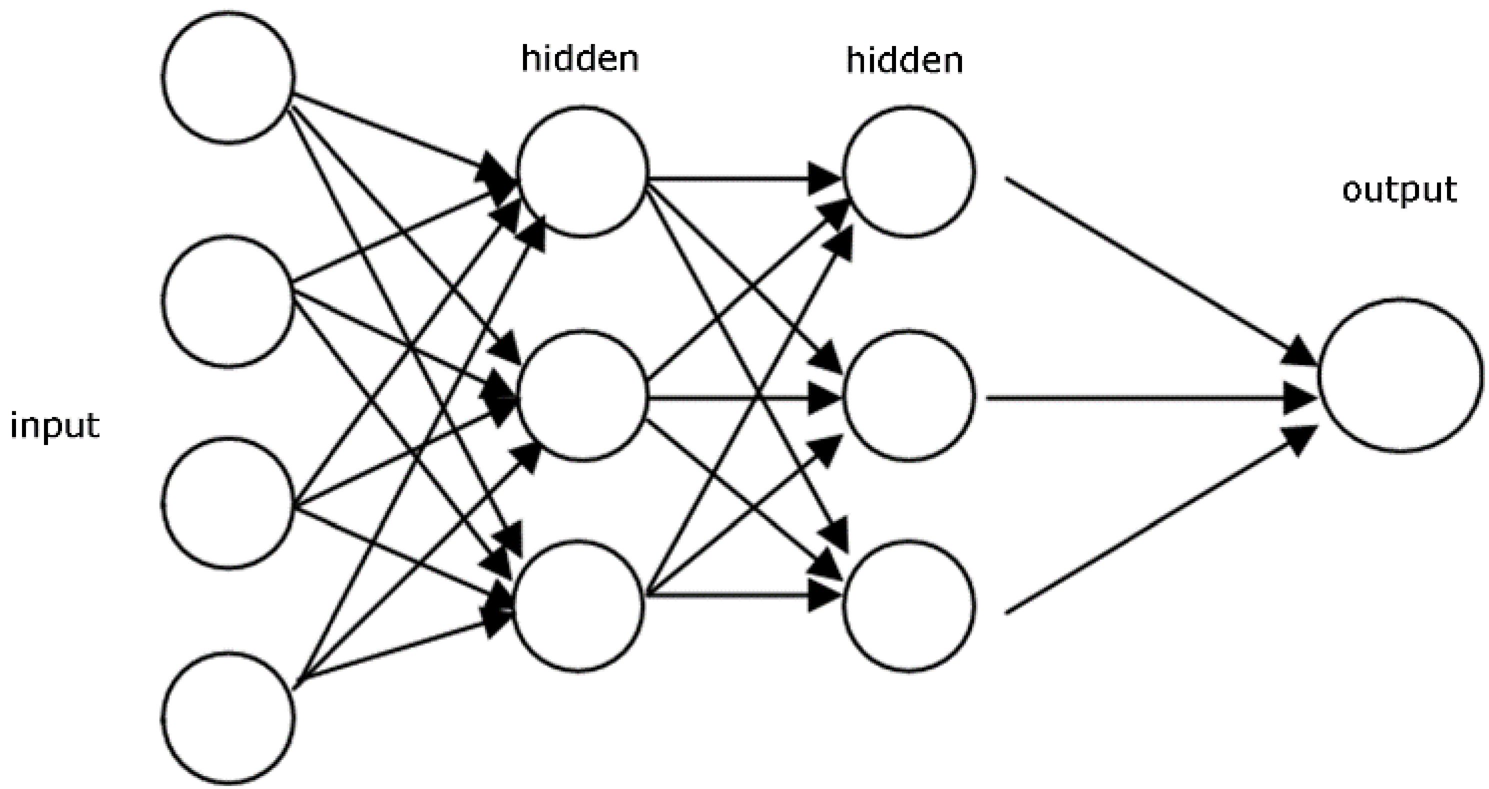


- Vision recognition



1.1 CNN's Area

- Feed Forward Network (One to One)



1.2 CNN's limitation

- Well with large/labeled dataset, with fixed length input

  - Vision recognition, Speech recognition

- How about variable length?

  - Sequence mapping (Machine Translation), Question Answering (Chatbot)
  - Image captioning, Sentiment Classification …

⇒ BAD !   Need domain independent method

2. RNN

- 2.1 RNN
- 2.2 RNN’s back-porpagation
- 2.3 Sequence to Sequence Model

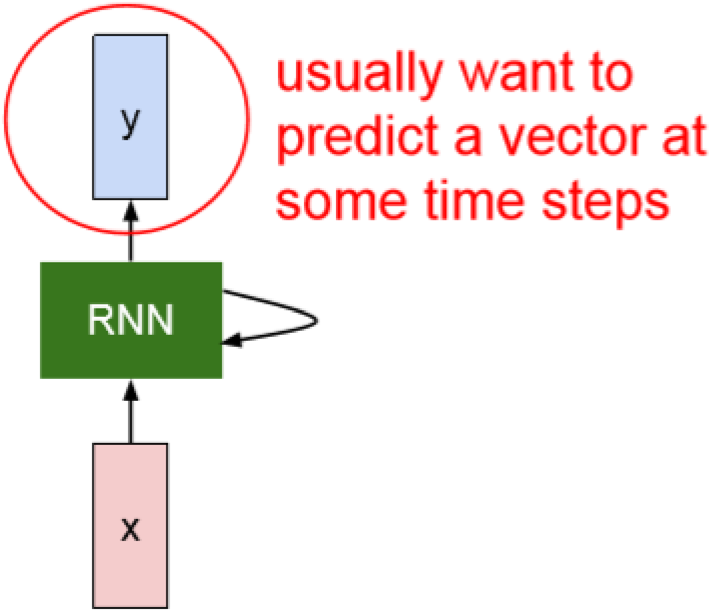
2.1 RNN

- RNN: Timestamp Output Calculation

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

$$h_{t-1} : (t-1)'th\ hidden\ state$$

$$x_t : t'th\ input$$

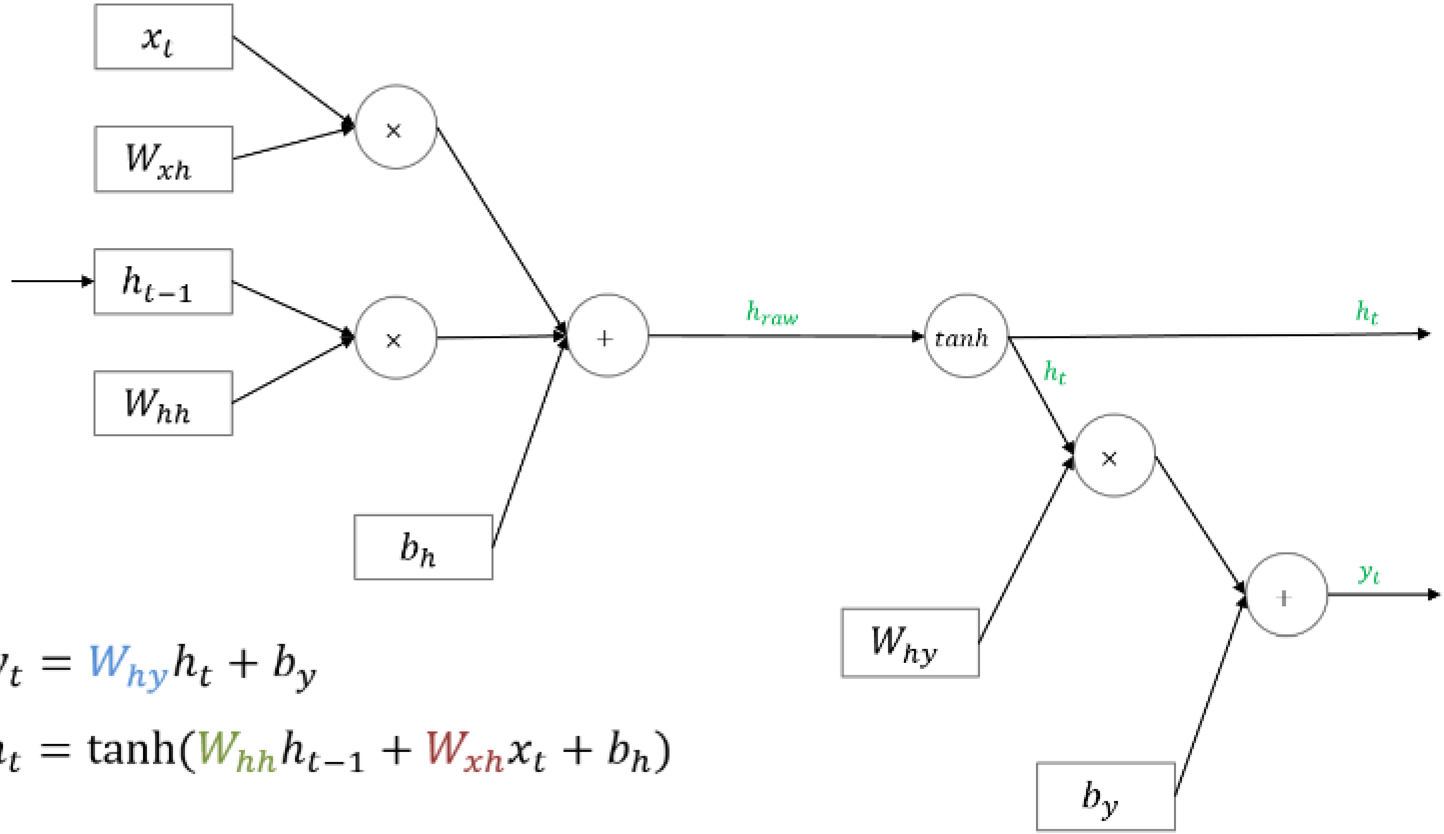


The result of the hidden layer(past step) enters the input of the next calculation!!

2.1 RNN

- RNN's computational graph

  : Forward compute pass

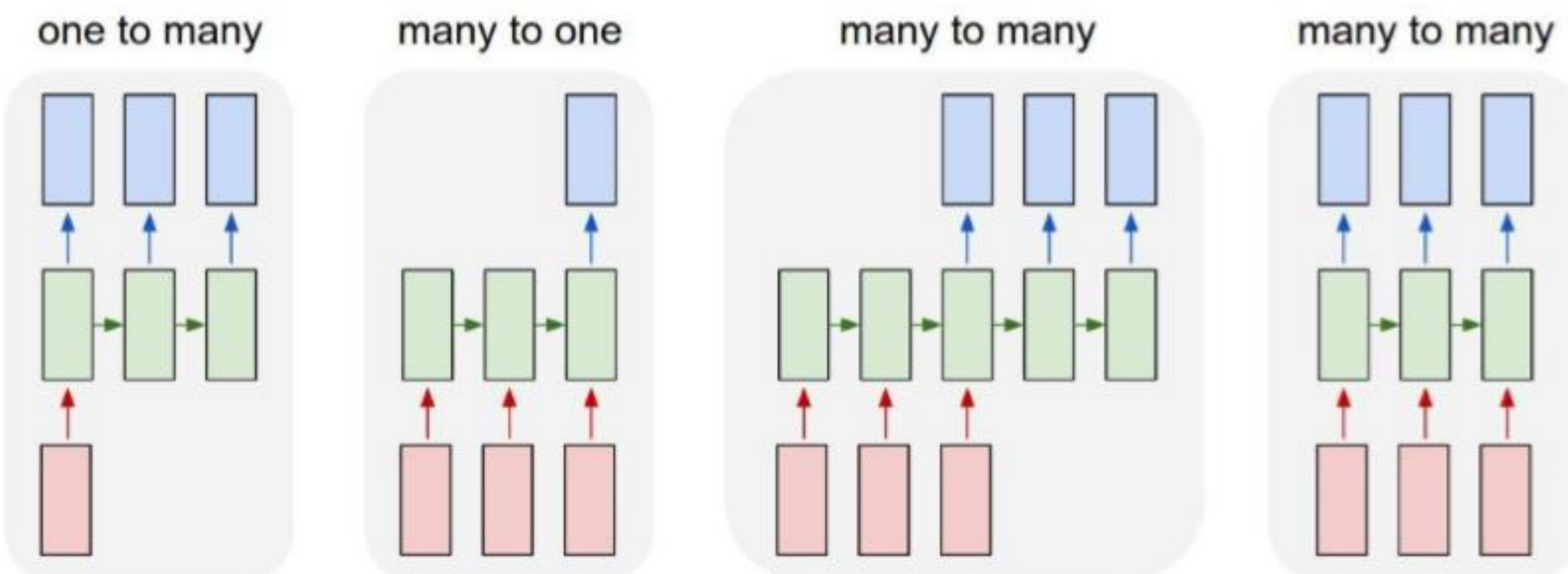


$$y_t = W_{hy}h_t + b_y$$

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

2.1 RNN

- RNN's Area

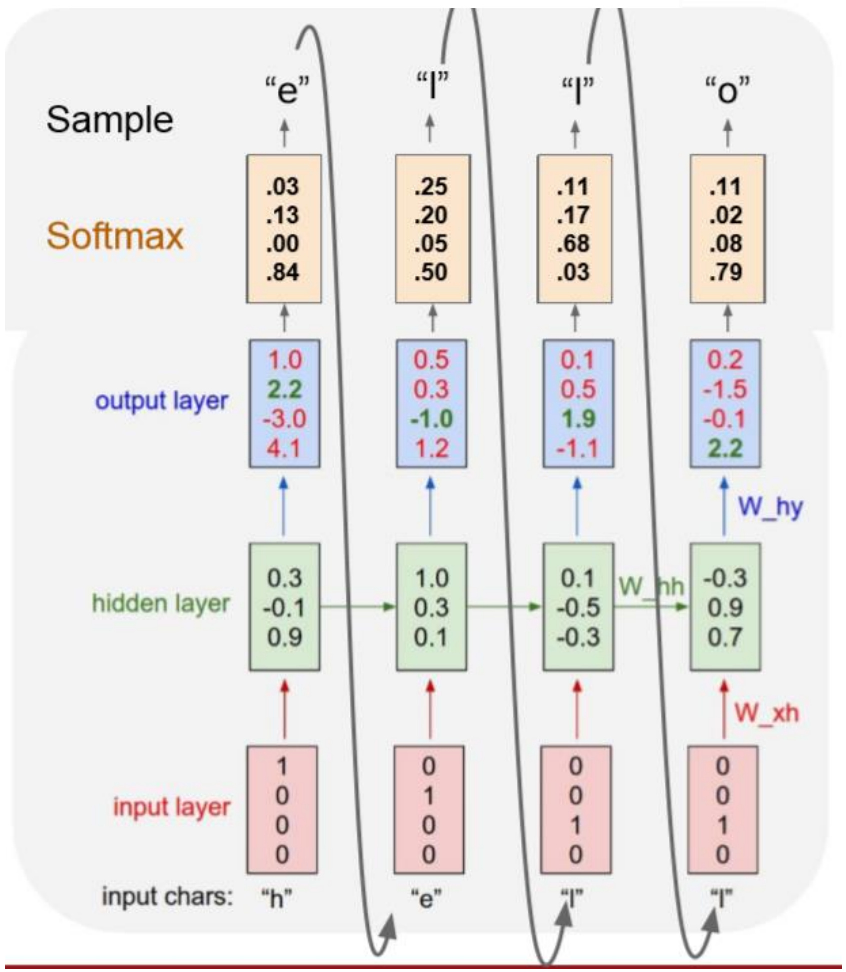


(Image Captioning) (Sentiment Classification) (machine translation) (Video Classification)

2.1 RNN

- Example : ‘hello’

  :Forward propagation

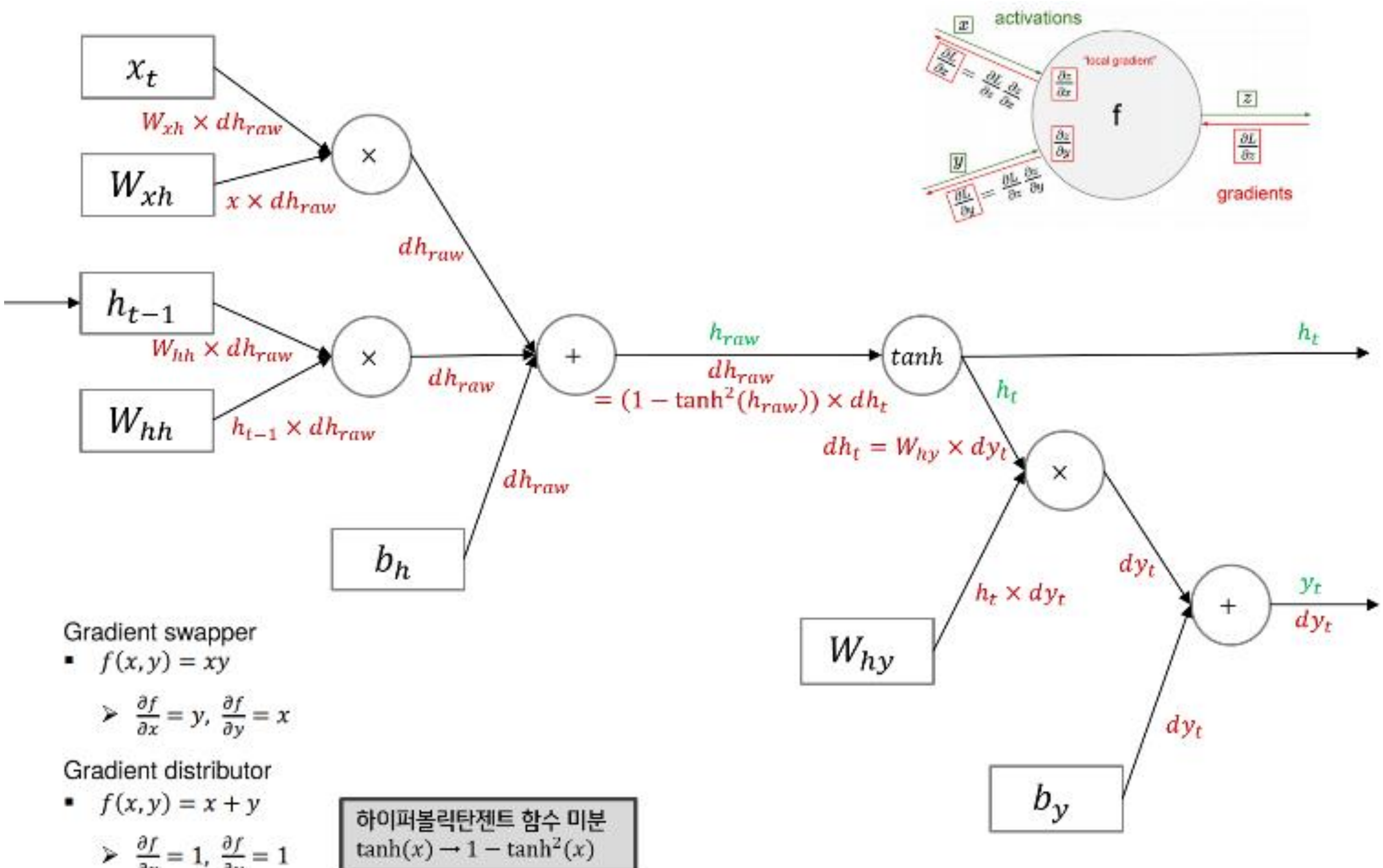


Conditional Probability Distribution

$$p(y_1, \ldots, y_{T'} | x_1, \ldots, x_T) = \prod_{t=1}^{T'} p(y_t | v, y_1, \ldots, y_{t-1})$$

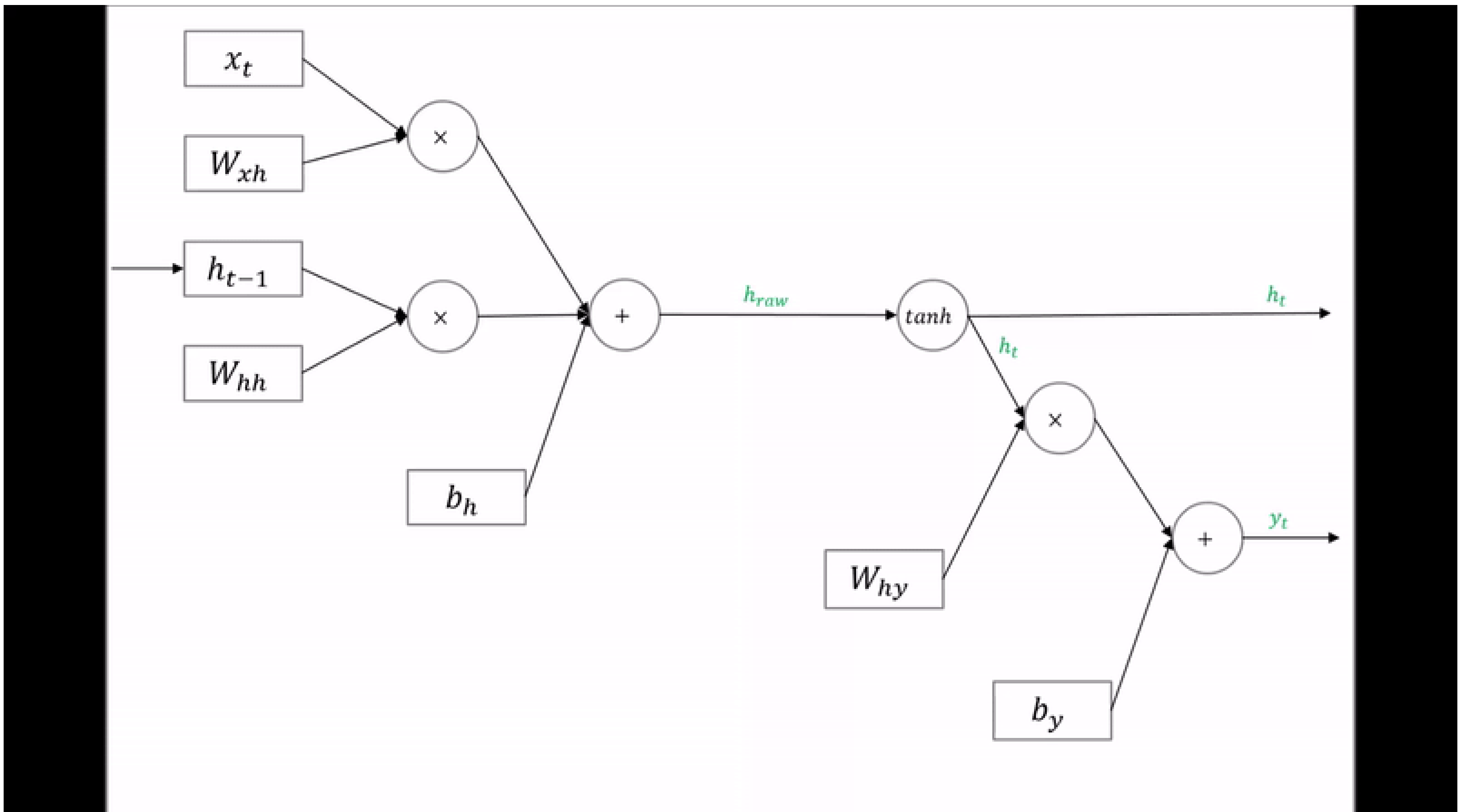
2.2 RNN's back-propagation

- RNN's back-propagation



2.2 RNN's back-propagation

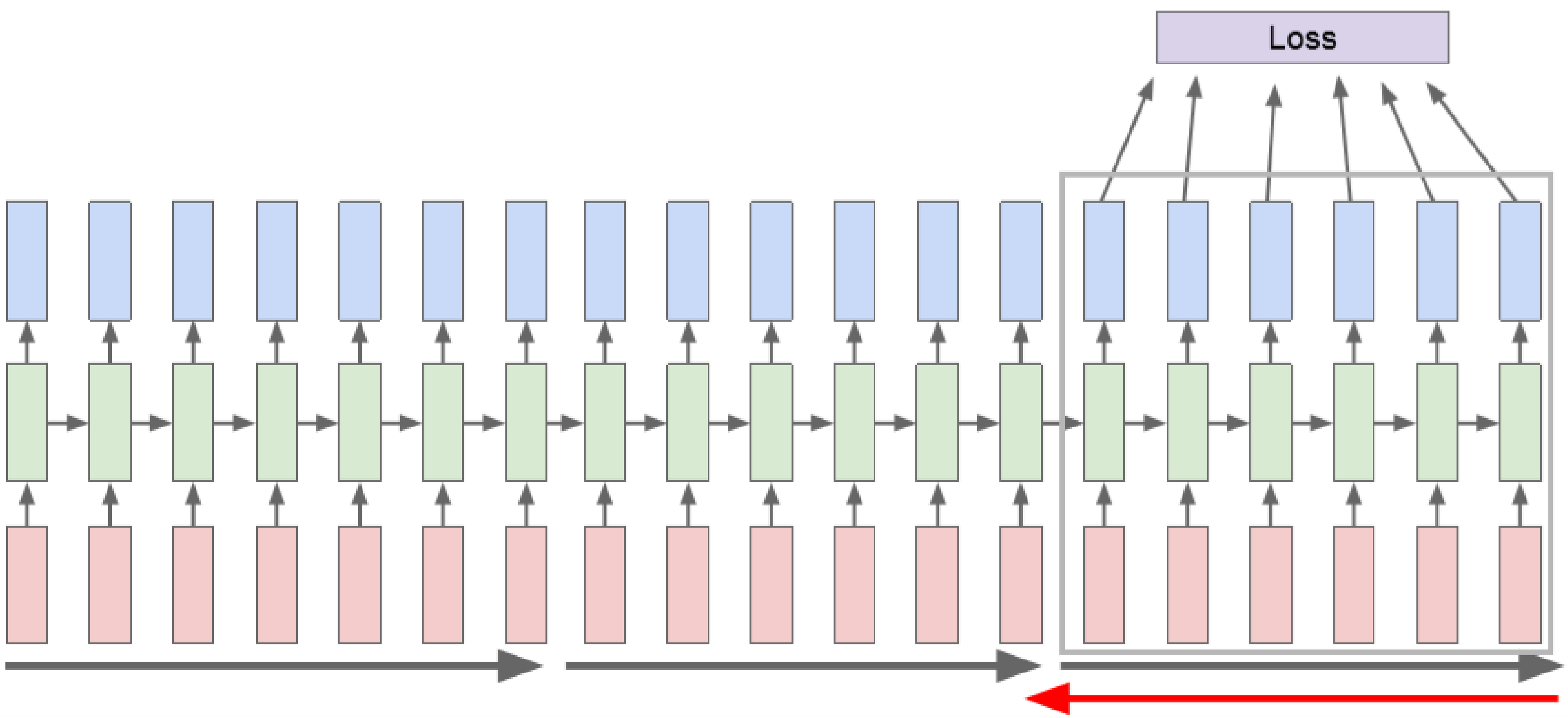
- RNN's back-propagation



2.2 RNN's back-propagation

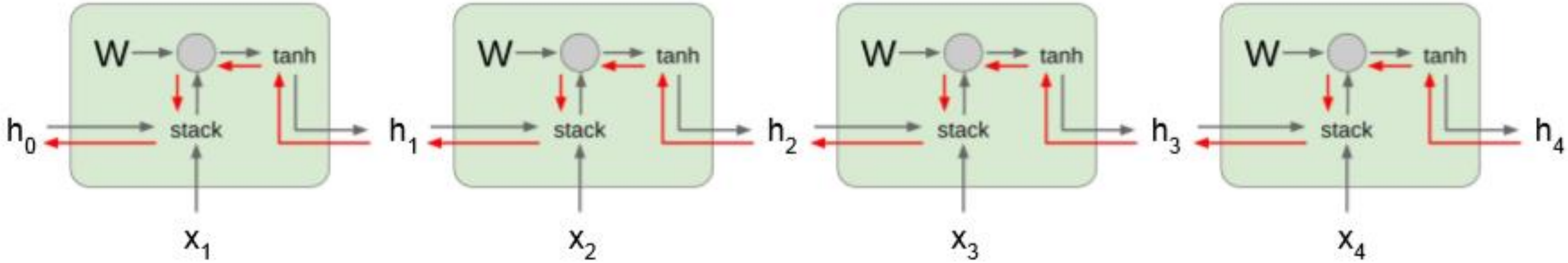
- Truncated Back-propagation

  : like mini-batch



2.2 RNN's back-propagation

- Vanishing or Exploding Gradient



(derivation with hidden state)

$$\frac{\partial h_T}{\partial h_t} = (W^{hh})^{T-t} * \prod_{i=t}^{T-1} \tanh'(W^{hh} h_i + W^{hx} x_{i+1})$$

2.2 RNN's back-propagation

- Exploding Gradient

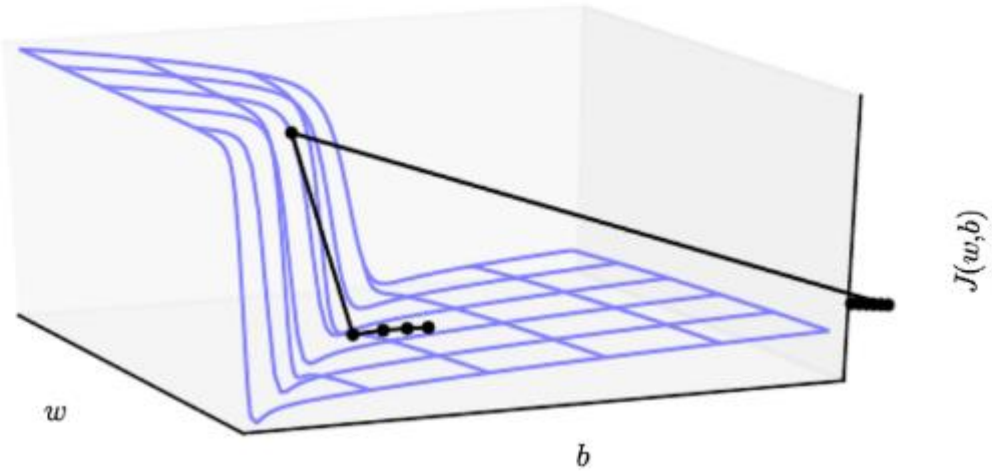
  Solution: Norm Clipping (scaling gradient)

Algorithm 1 Pseudo-code for norm clipping

$\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta}$
if $\|\hat{\mathbf{g}}\| \geq threshold$ **then**
  $\hat{\mathbf{g}} \leftarrow \frac{threshold}{\|\hat{\mathbf{g}}\|} \hat{\mathbf{g}}$
end if

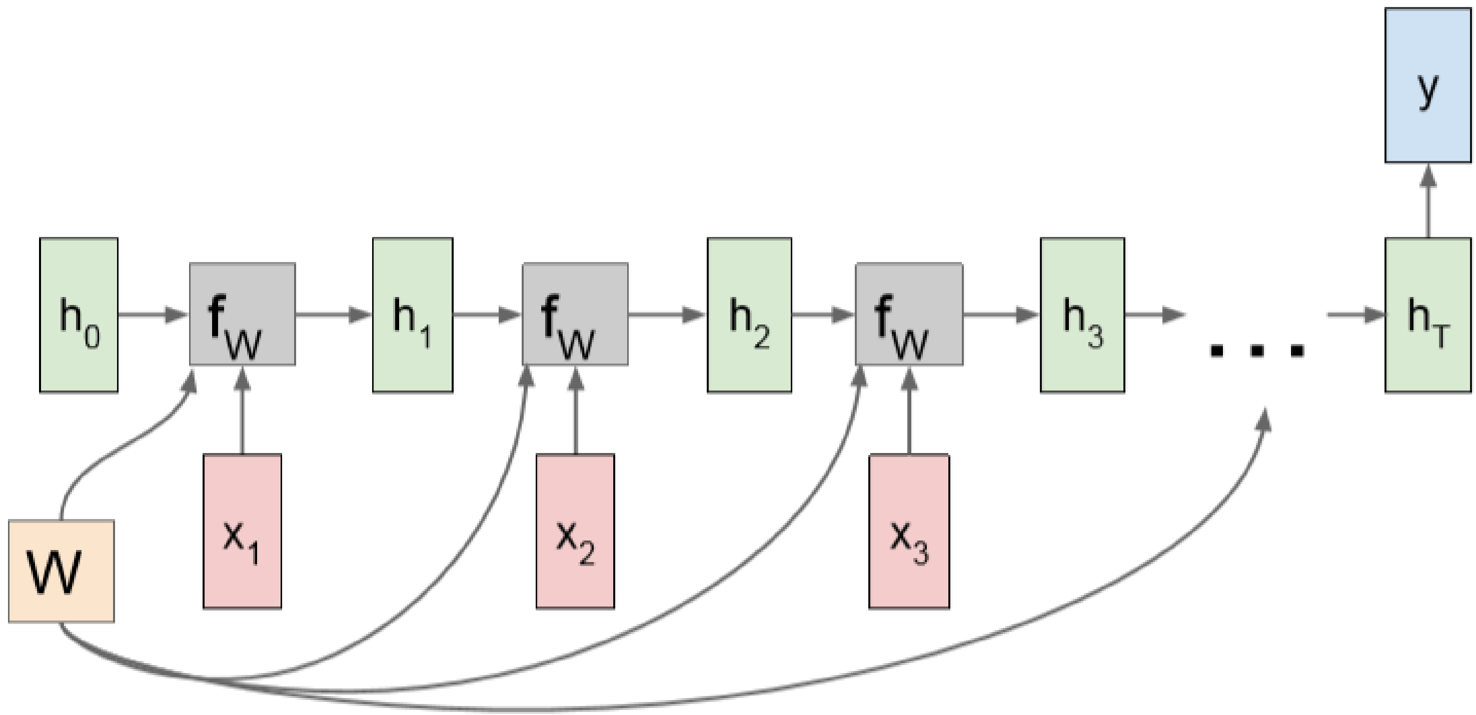
Just scaling :

gradient direction is unchanged



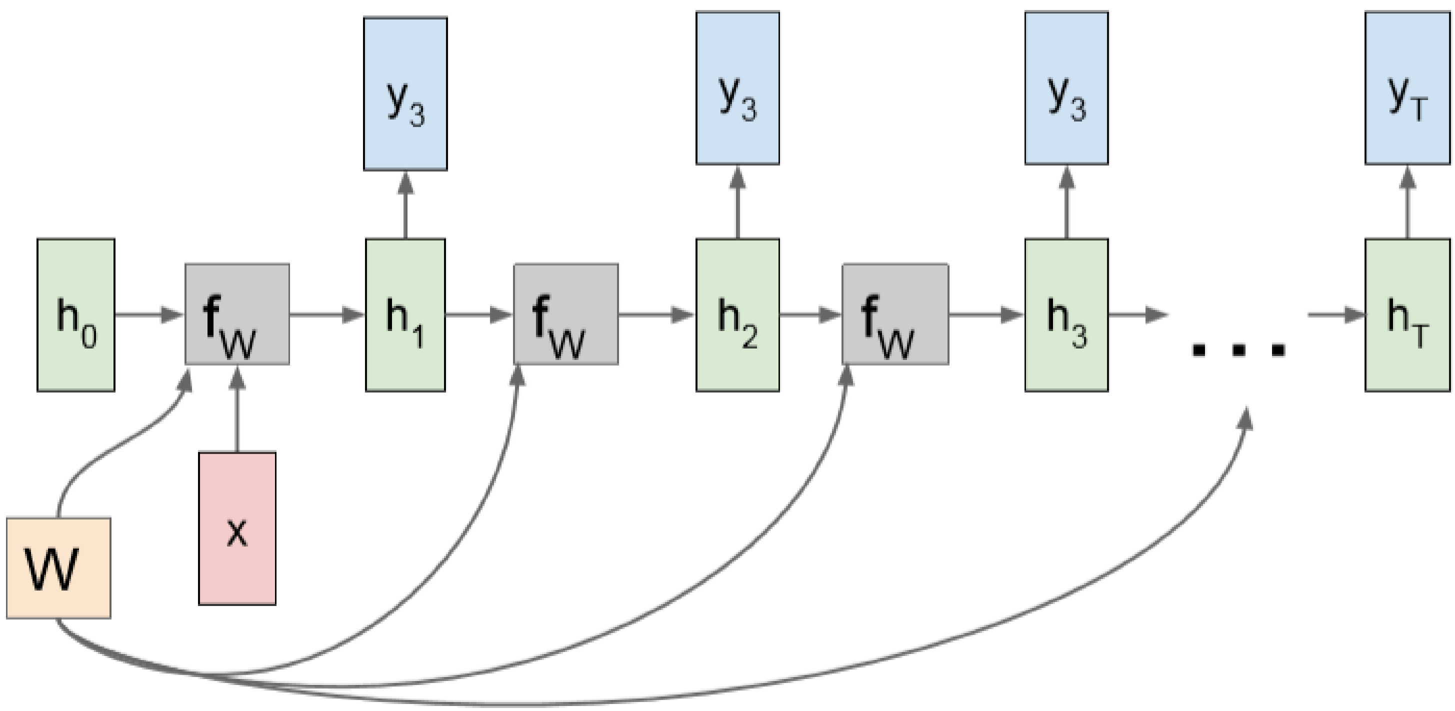
2.3 Sequence to Sequence Model

- RNN : Many to One



2.3 Sequence to Sequence Model

- RNN : One to Many

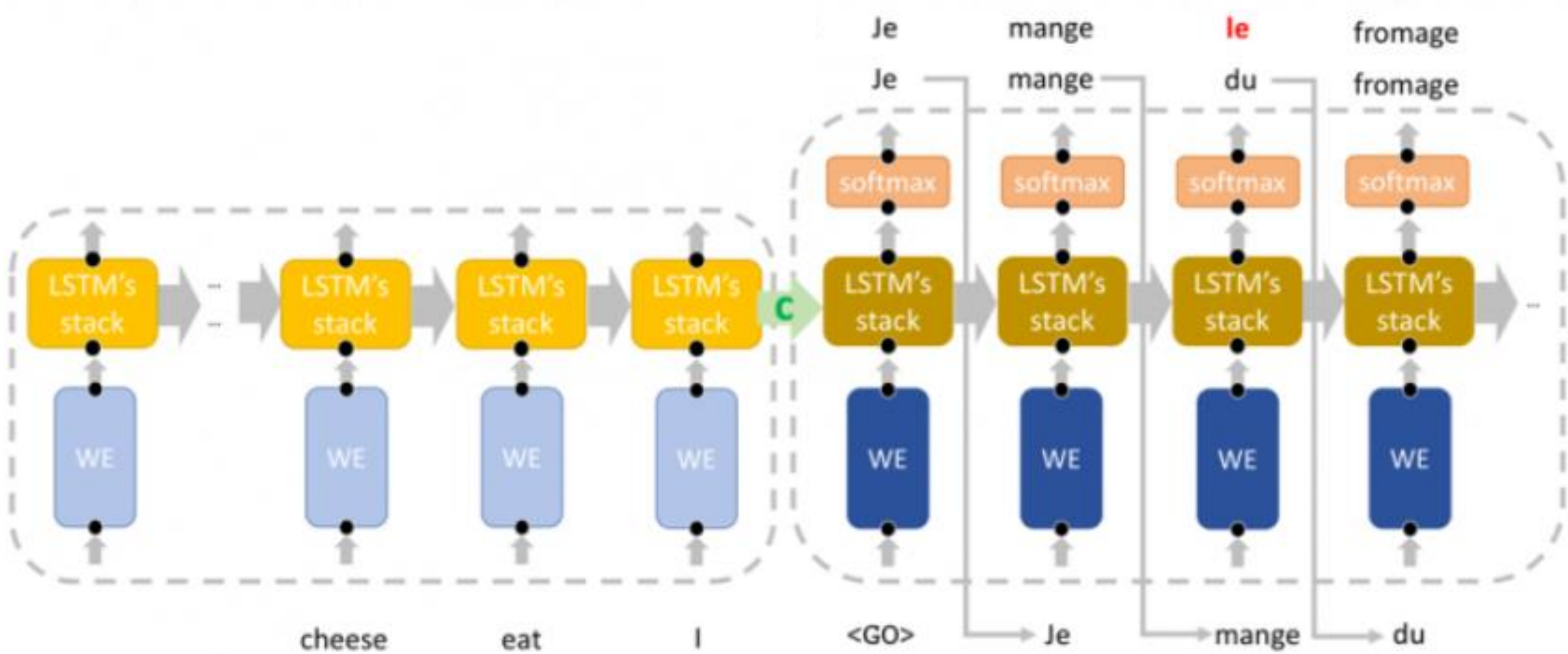


2.3 Sequence to Sequence Model

- Encoder – Decoder

    1) negligible computational cost

    2) train multiple language pairs simultaneously



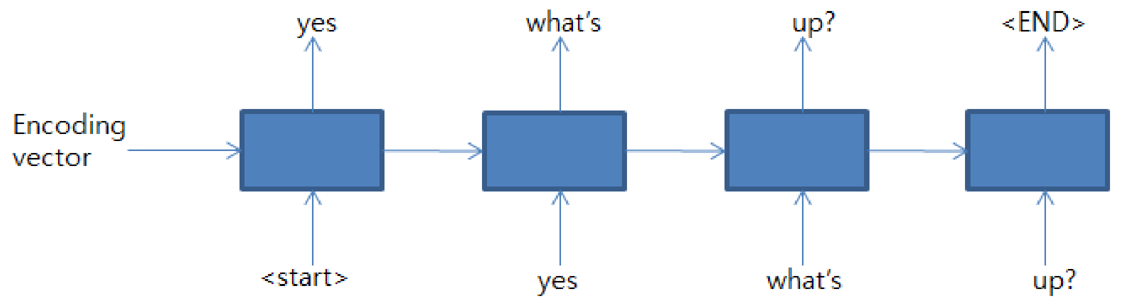
2.3 Sequence to Sequence Model

- Decoding and Rescoring

1. Train :

$$1/|\mathcal{S}| \sum_{(T,S)\in\mathcal{S}} \log p(T|S)$$

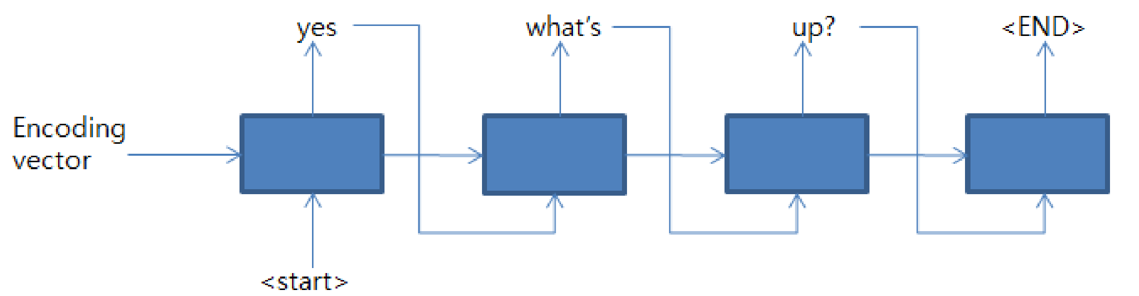
<TRAIN MODEL>



2. Test :

$$\hat{T} = \arg\max_{T} p(T|S)$$

<TEST MODEL>

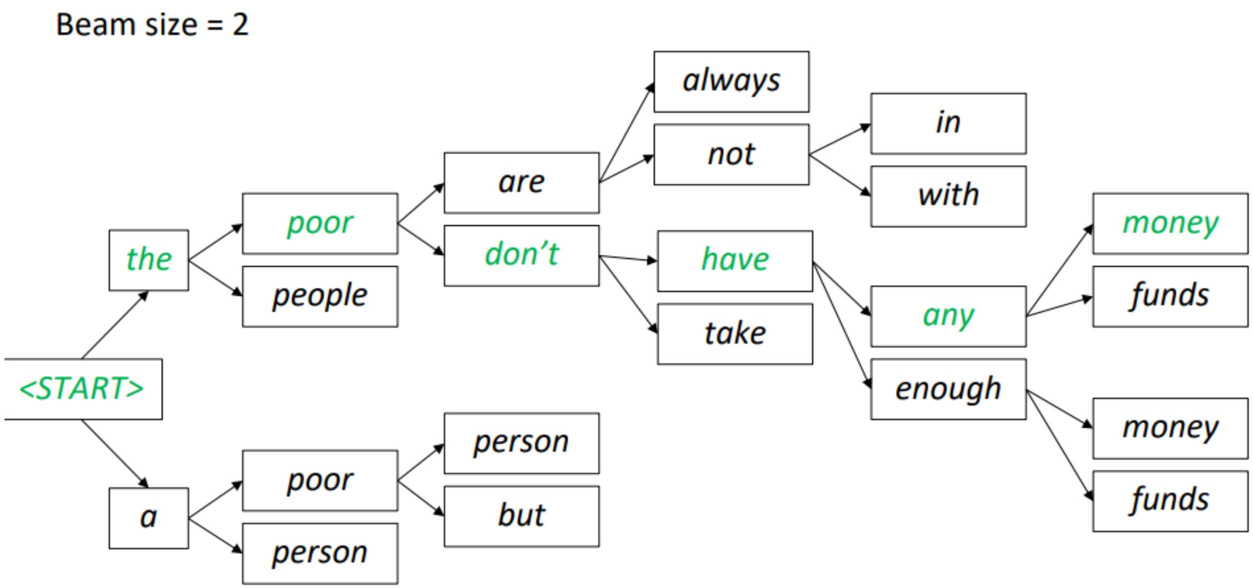


2.3 Sequence to Sequence Model

- Beam Search (Test)

  Extension of Greedy search

  : discard all but B most likely hypothesis

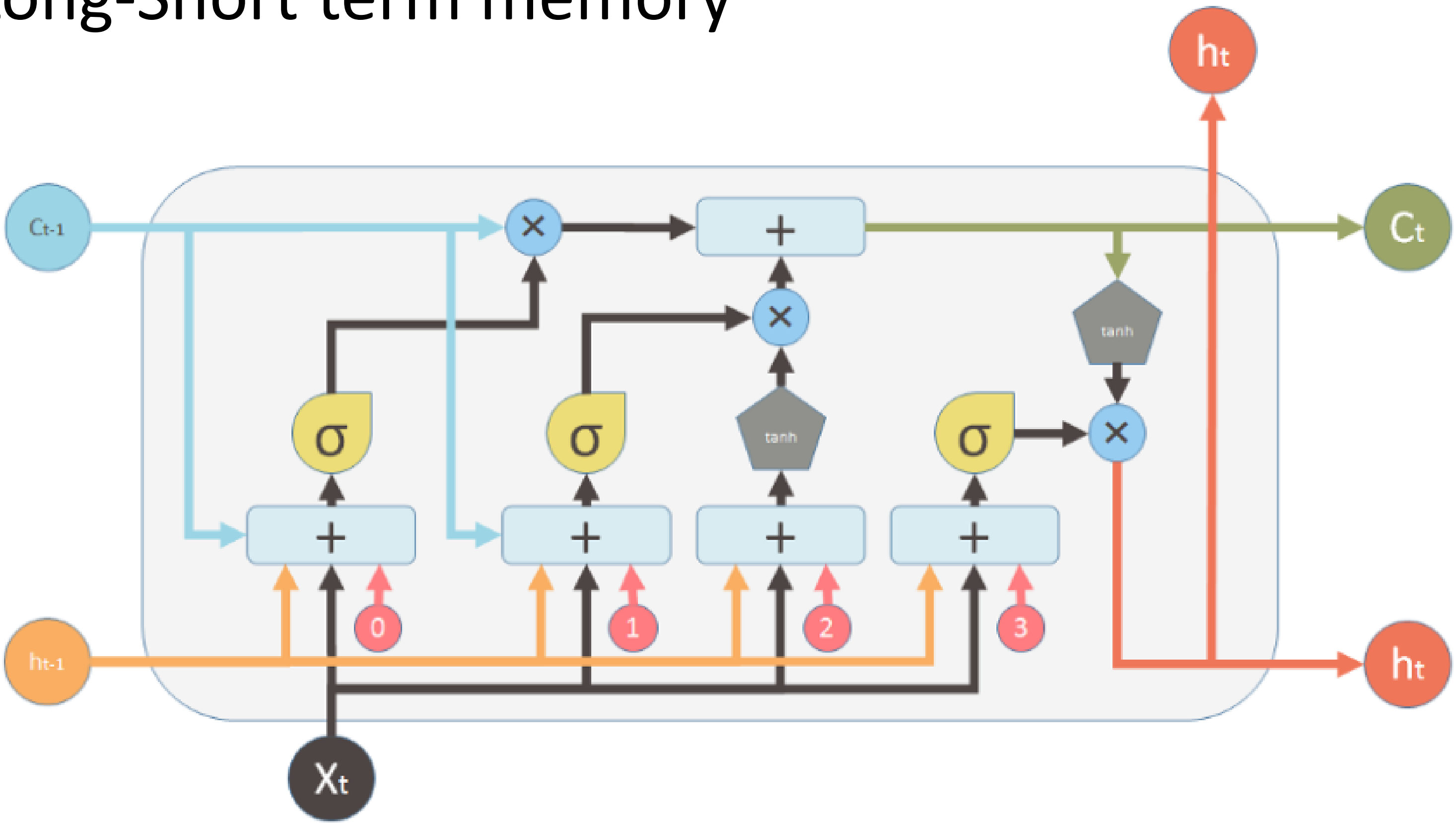


3. LSTM

- 3.1 LSTM
- 3.2 LSTM’s back propagation

3.1 LSTM

- Long-Short term memory

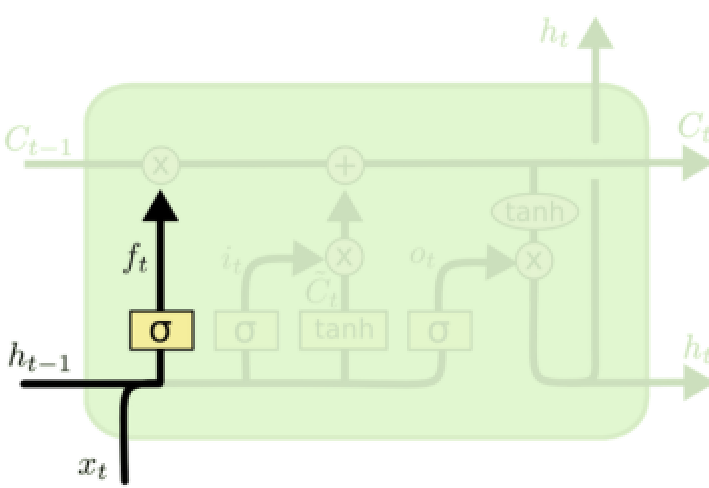


Add cell state that can memorize (I,F,O,G gate)

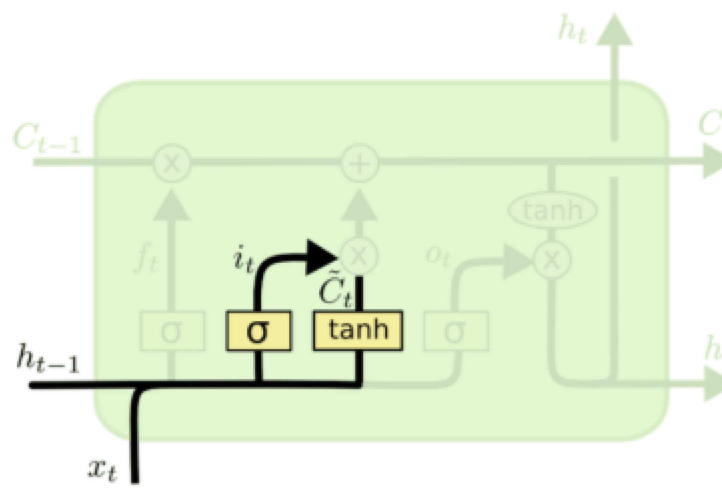
3.1 LSTM

- Forget & Input gate

$$
\begin{aligned}
f_t &= \sigma(W_{xh\_f} x_t + W_{hh\_f} h_{t-1} + b_{h\_f}) \\
i_t &= \sigma(W_{xh\_i} x_t + W_{hh\_i} h_{t-1} + b_{h\_i}) \\
g_t &= \tanh\left(W_{xh\_g} x_t + W_{hh\_g} h_{t-1} + b_{h\_g}\right)
\end{aligned}
$$



<forget gate>

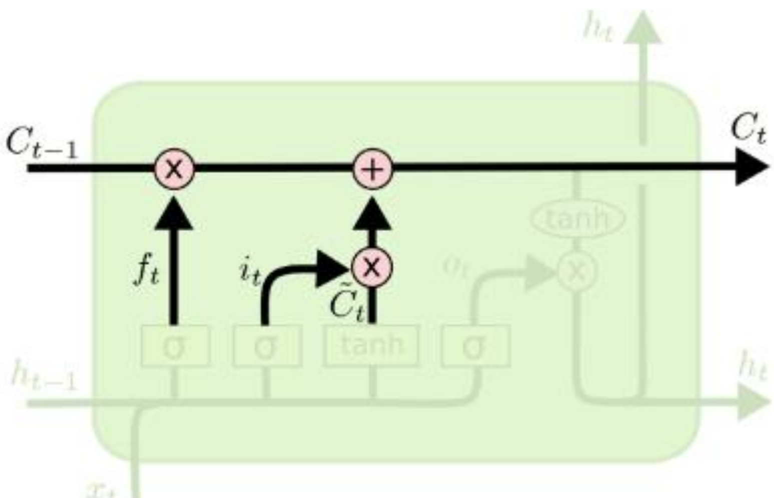


<input gate>

3.1 LSTM

- Cell state

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t$$ (hadarmard product)

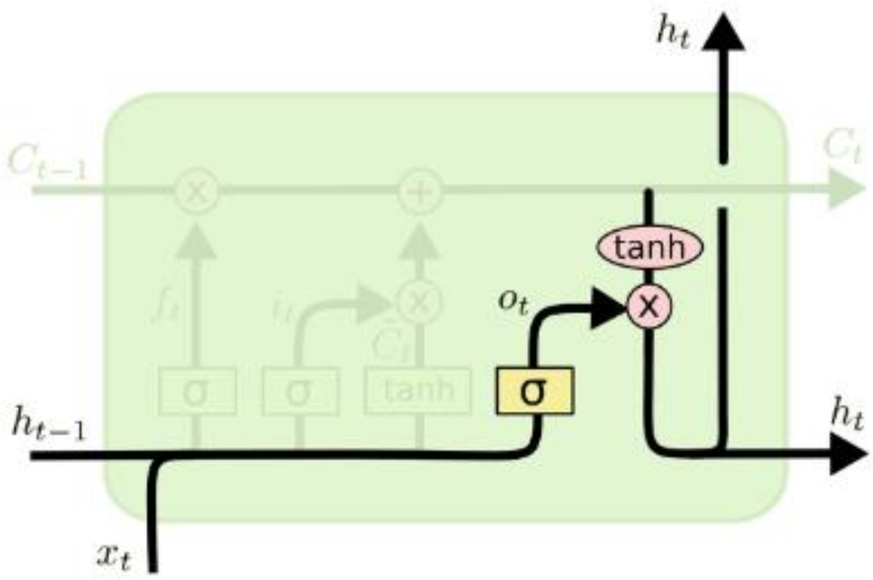


3.1 LSTM

- Output & Hidden state

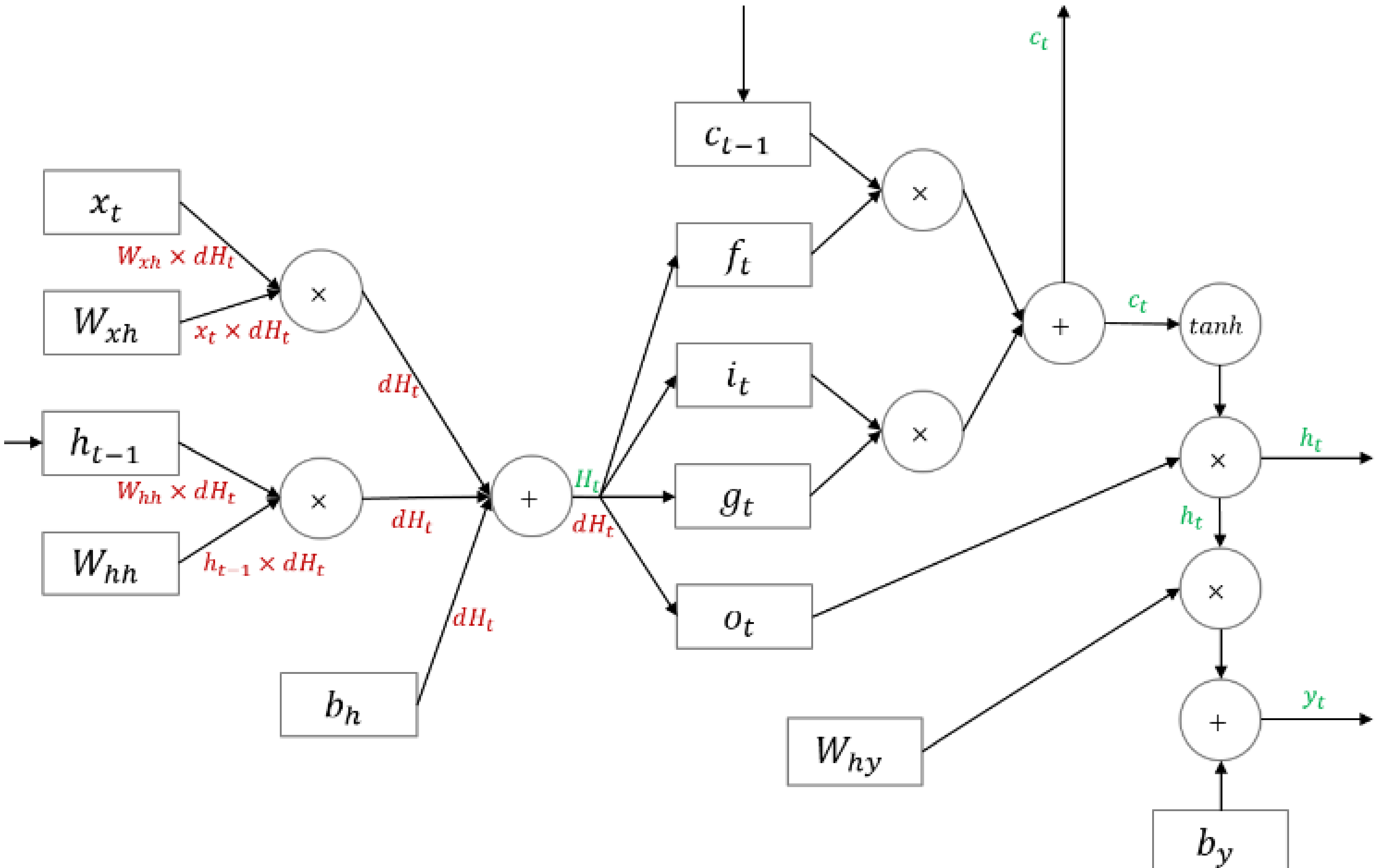
$$h_t = o_t \odot \tanh(c_t)$$

$$o_t = \sigma(W_{xh\_o} x_t + W_{hh\_o} h_{t-1} + b_{h\_o})$$



3.2 LSTM's back propagation

- LSTM's computational graph



3.2 LSTM’s back propagation

- Long-term dependency

  : partial solution of 'long-range dependency’ $$\frac{\partial C_T}{\partial C_t} = \prod_{i=t+1}^{T} f_i$$

- Pros and Cons
  1. Back Propagation
  2. Memory circuit + Neural Network
  3. Long-Term dependency
  4. Exploding Gradient

4. Example

- 4.1 Example
- 4.2 Searching interpretable cells

4.1 Example

- William Shakespeare

```
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.
```

4.1 Example

- Algebric geometry textbook

Proof. Omitted. □

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

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

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

.

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

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

where $\mathcal{G}$ defines an isomorphism $\mathcal{F} \to \mathcal{F}$ 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 $\mathcal{U} \subset \mathcal{X}$ 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

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

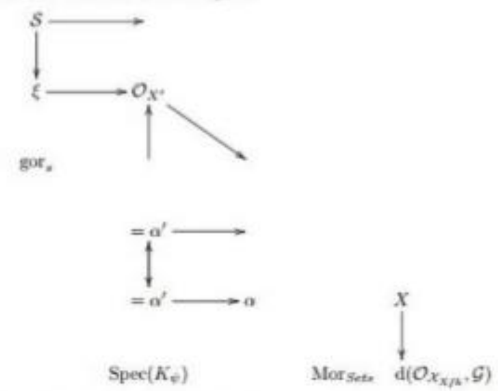
be a morphism of algebraic spaces over S and Y.

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

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

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.

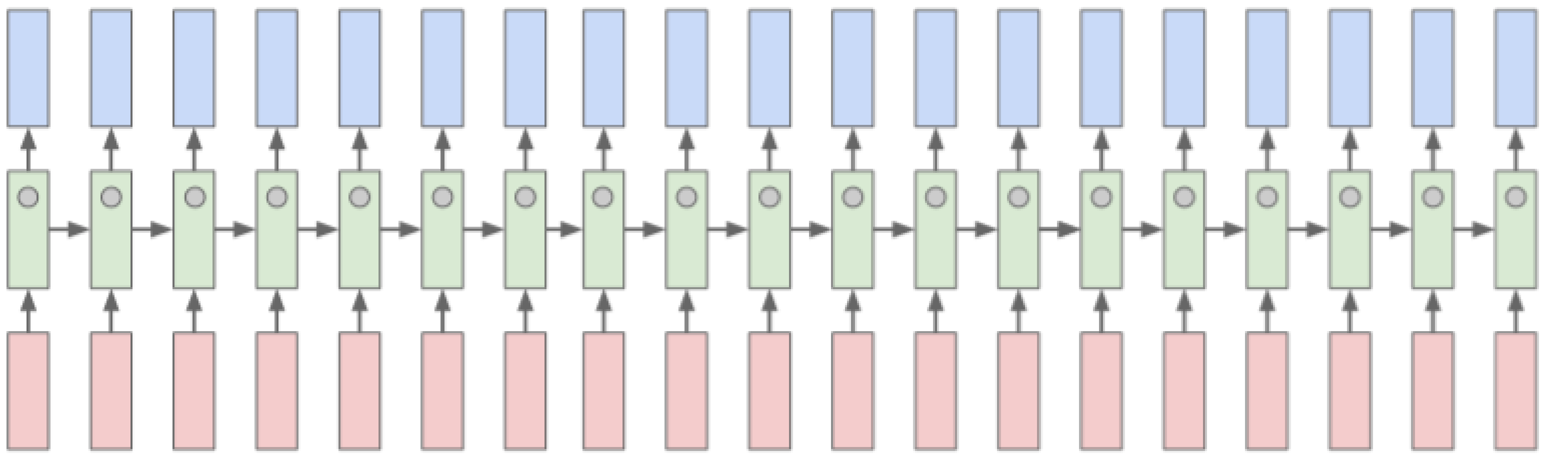
4.1 Example

- C code

```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;
}
```

4.2 Searching interpretable cells

- Finding hidden state



```
#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;
}
```

5. Attention

- 5.1 Attention
- 5.2 Image Captioning

5.1 Attention Mechanism

- LSTM’s long-term dependency

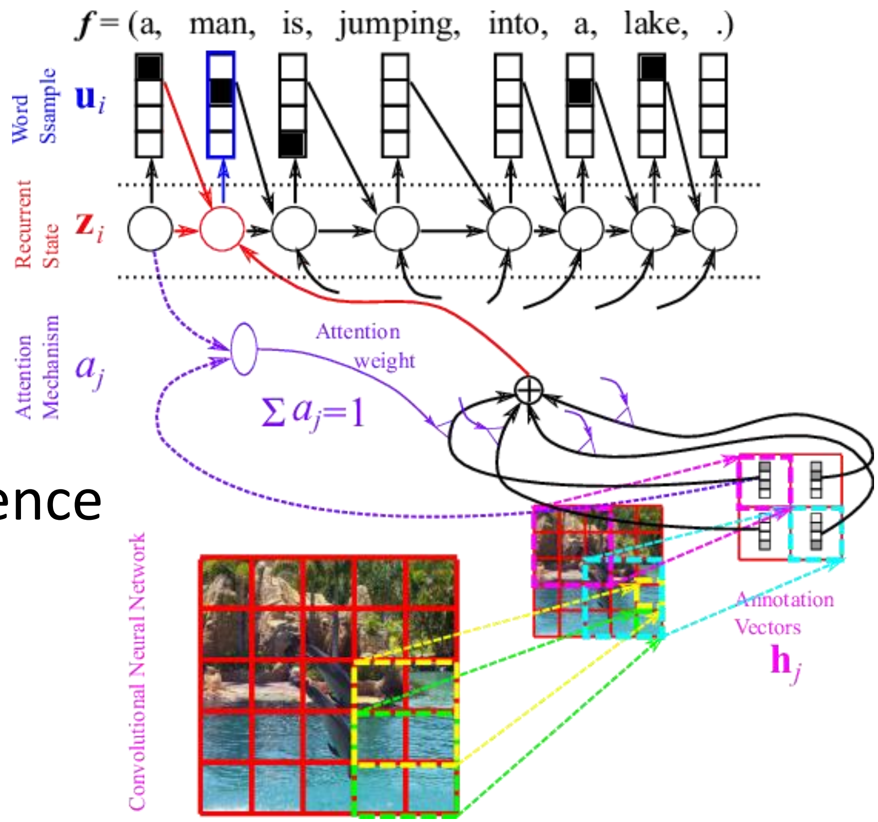
“BottleNeck” problem
:Need to know all the information in the input sentence to
single vector (context vector)

But the information needed for each word will be different.

5.1 Attention Mechanism

"Neural Machine Translation by Jointly Learning to Align and Translate" present attention mechanism

-Bahdanau, Cho (2015, ICLR)



- At every time-step,

  Encoder:

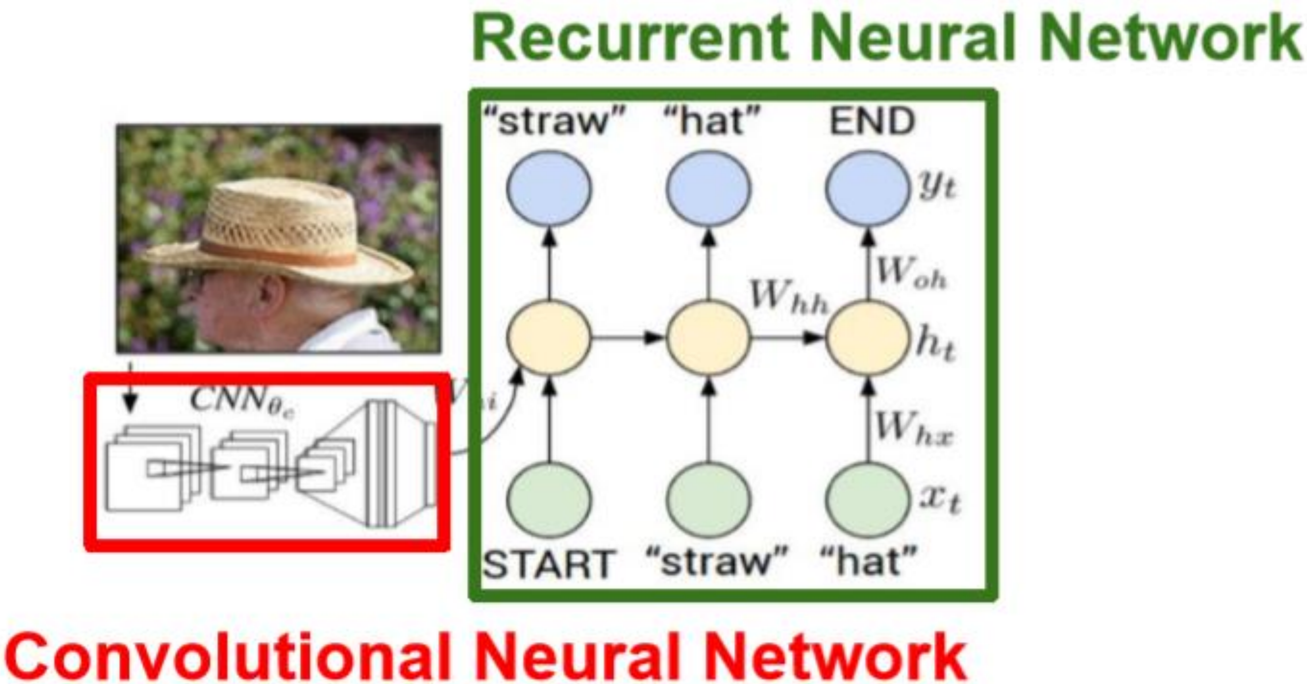
  refer to the entire input sentence

5.2 Image Captioning

- CNN + RNN

Input : Image (fixed-size vector)

Output : Natural Language

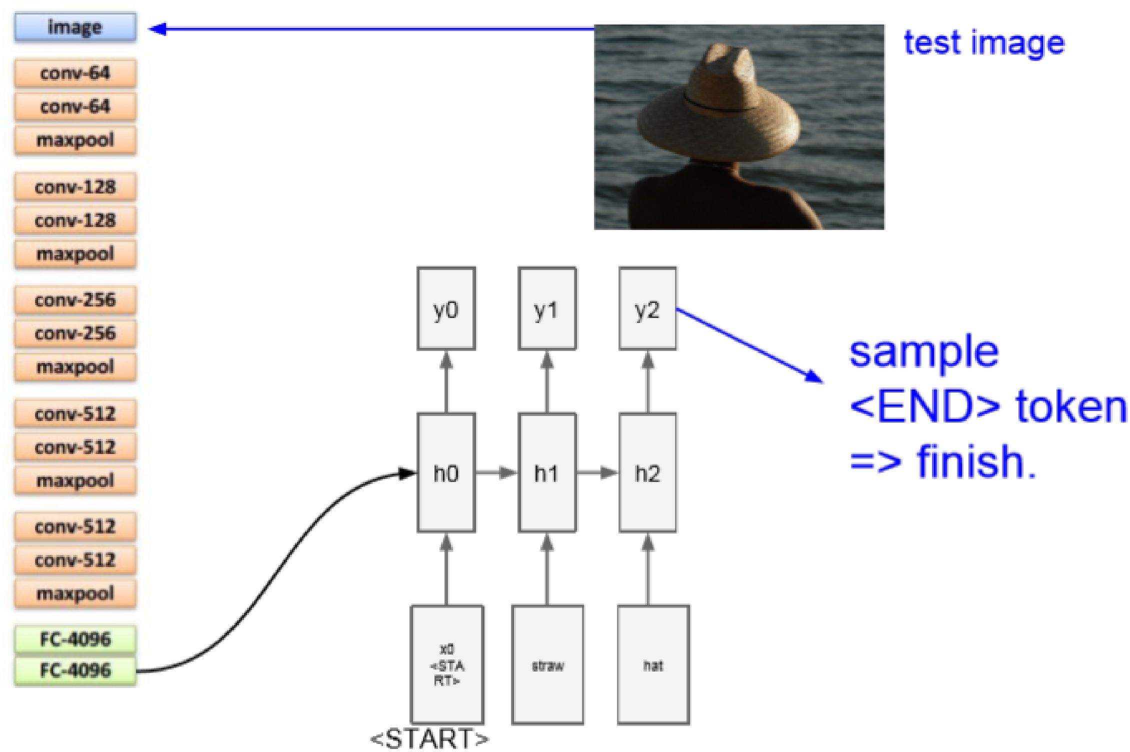


5.2 Image Captioning

- CNN's role :

Summarizing the information of an image like an encoder into a single vector

RNN start with <sos> token (start)



5.2 Image Captioning

- "Visual Relationship Detection with Language Priors"
  - Cewu Lu*, Ranjay Krishna*, Michael Bernstein, Li Fei-Fei

Using Word Vector's Information (word2vec)