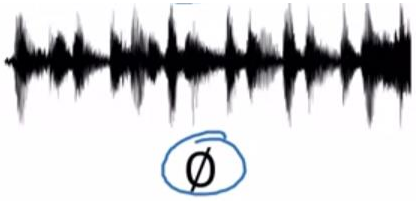
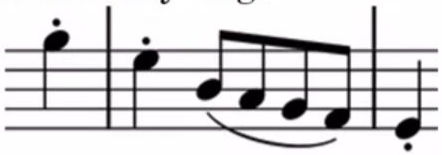
NLP Study Week 1

Sequence

- sequence는 시계열적인 특성을 가짐
  > time에 따라 변해가는 양상을 모델에 표현할 수 있음
- 자연어 처리도 마찬가지로 단어의 순서에 따라 영향을 미치기 때문에 적용할 필요 있음
- 이미지 모델에서는 sequence를 어떻게 적용할까?
  > 동영상과 같은 움직임이 들어가 있는 이미지의 경우 적용 가능
- 시퀀스 모델의 타입은 3가지로 볼수 있음
  > one to sequence
  > sequence to one
  > sequence to sequence

Examples of sequence data

Speech recognition		→	"The quick brown fox jumped over the lazy dog."
Music generation		→	
Sentiment classification	"There is nothing to like in this movie."	→	
DNA sequence analysis	AGCCCCTGTGAGGAACTAG	→	AGCCCCTGTGAGGAACTAG
Machine translation	Voulez-vous chanter avec moi?	→	Do you want to sing with me?
Video activity recognition		→	Running
Name entity recognition	Yesterday, Harry Potter met Hermione Granger.	→	Yesterday, Harry Potter met Hermione Granger.

Andrew Ng

Notation

X : Harry Potter and Harmione Granger invented a new spell

$x^{<1>}$ $x^{<2>}$ …… $x^{<9>}$

Y : 1 1 0 1 1 …… 0

$y^{<1>}$ $y^{<2>}$ …… $y^{<9>}$

$X^{(i)<t>}$ $Y^{(i)<t>}$: i번째 문장 t번째 단어

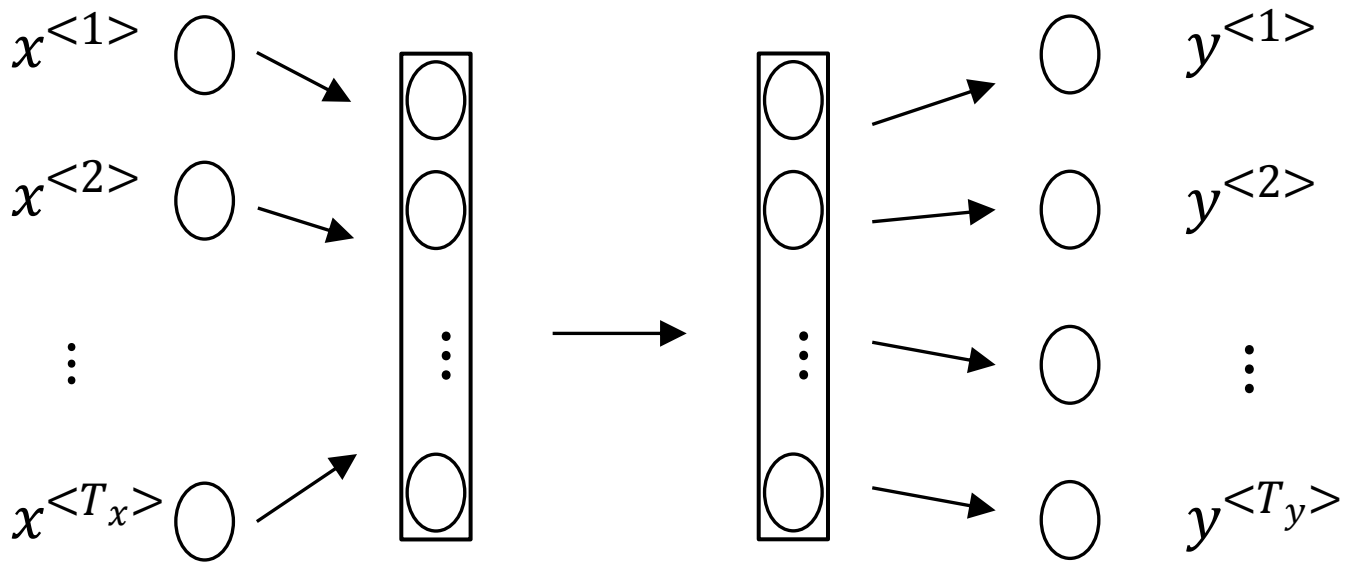
$T_x^{(i)} = 9$ $T_y^{(i)} = 9$: 갯수

Notation

x: Harry Potter and Hermione Granger invented a new spell.

$x^{<1>}$ $x^{<2>}$ $x^{<3>}$ ... $x^{<9>}$

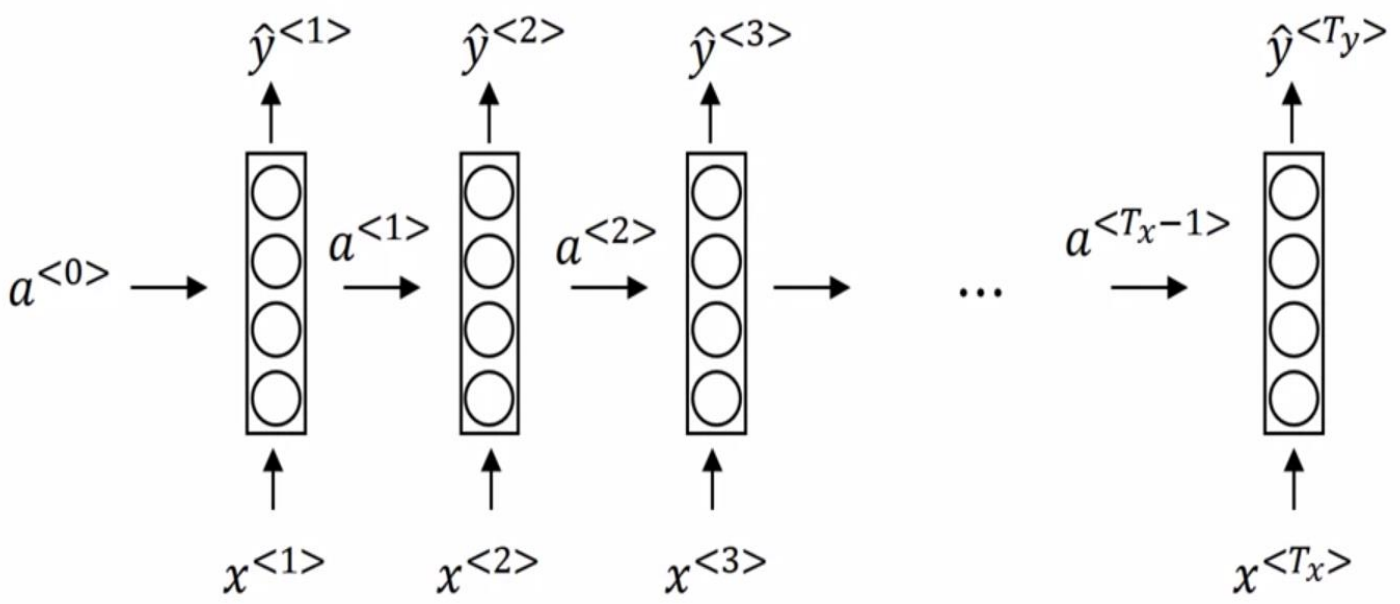
Why not a standard network?



Problems:

- Inputs, outputs can be different lengths in different examples.
- Doesn’t share features learned across different positions of text.
- Too many parameters

RNN



$a^{<0>} = \vec{0}$.

$$a^{<1>} = g_1(W_{aa} a^{<0>} + W_{ax} x^{<1>} + b_a) \leftarrow \tanh / \text{ReLU}$$

$$\hat{y}^{<1>} = g_2(W_{ya} a^{<1>} + b_y) \leftarrow \text{sigmoid}$$

$$a^{<t>} = g(W_{aa} a^{<t-1>} + W_{ax} x^{<t>} + b_a)$$

$$\hat{y}^{<t>} = g(W_{ya} a^{<t>} + b_y)$$

RNN

$$a^{<t>} = g(W_{aa}a^{<t>} + W_{ax}x^{<t>} + b_a)$$

$$\hat{y}^{<t>} = g(W_{ya}a^{<t>} + b_y)$$

$$\hat{y}^{<t>} = g(W_y a^{<t>} + b_y)$$

$$a^{<t>} = g(W_a [a^{<t-1>}, x^{<t>}] + b_a)$$

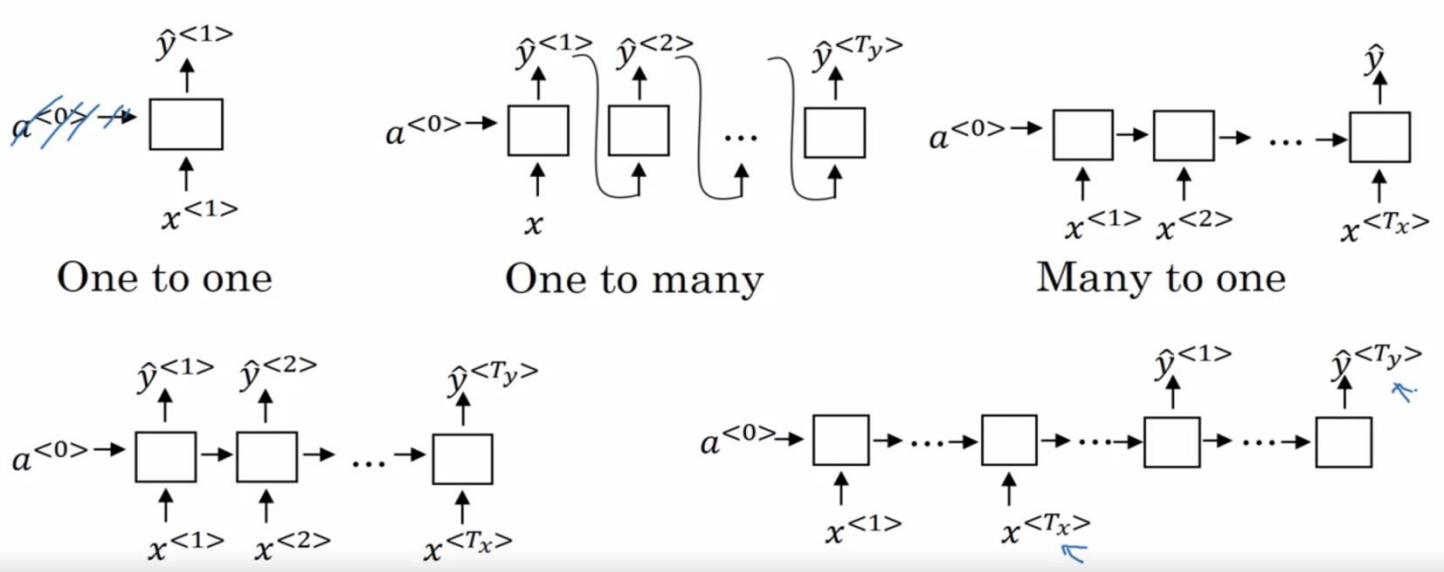
$$\begin{bmatrix} W_{aa} ; W_{ax} \end{bmatrix} = W_a$$

(100, 10100)

$$[a^{<t-1>}, x^{<t>}] = \begin{bmatrix} a^{<t-1>} \\ x^{<t>} \end{bmatrix}$$

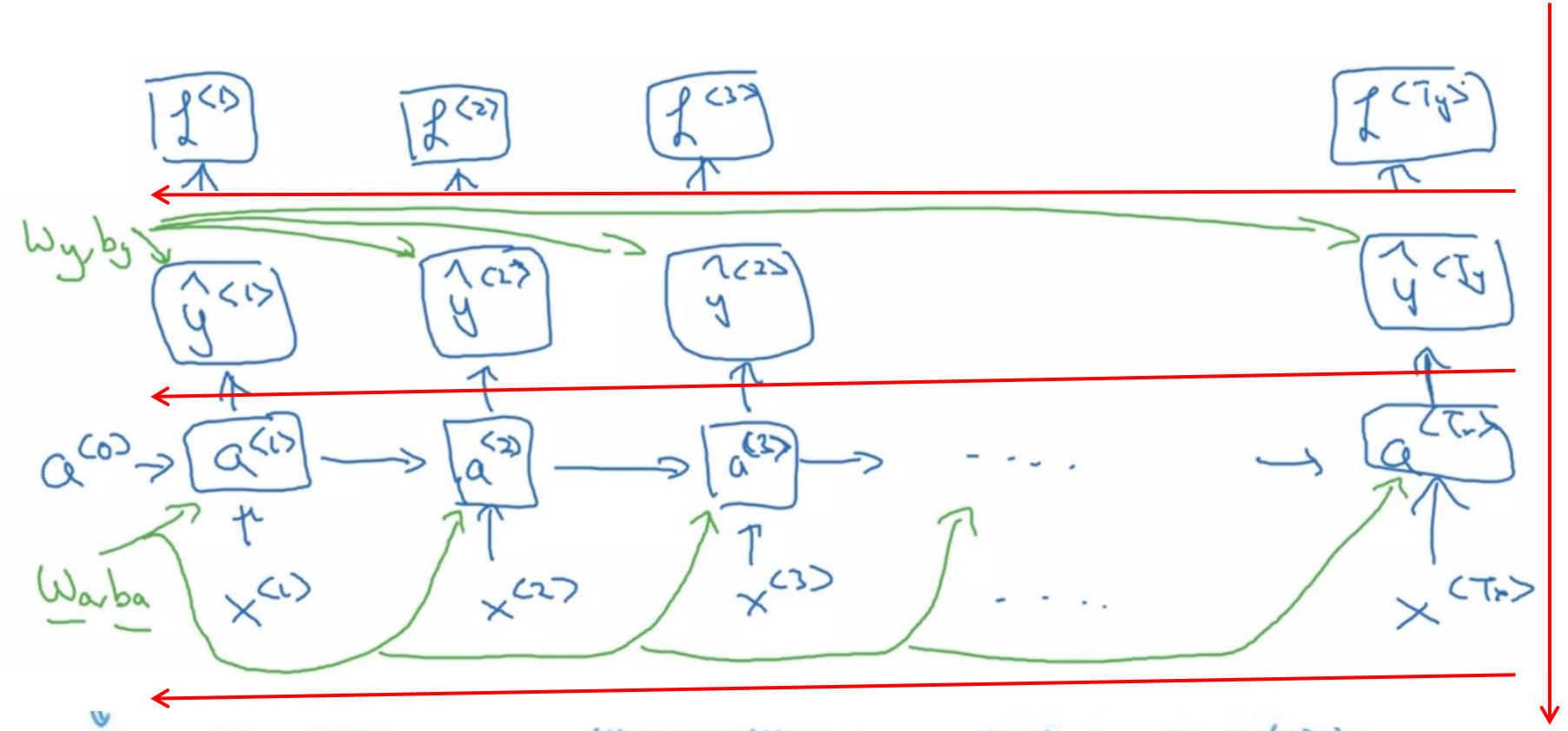
100, 10000, 10100

RNN



one to one : 일반적인 신경 네트워크
one to many : (음악 생성), (언어모델링), (이미지캡션),
many to one : 동영상 및 텍스트 입력하여 감성분석
many to many : 이름 엔터티 인식
many to many (encoder-decoder) : 기계번역

Backpropagation



$$\mathcal{L}^{<t>}(\hat{y}^{<t>}, y^{<t>}) = -y^{<t>} \log \hat{y}^{<t>} - (1 - y^{<t>}) \log (1 - \hat{y}^{<t>})$$

$$\mathcal{L}(\hat{y}, y) = \sum_{t=1}^{T_y} \mathcal{L}^{<t>}(\hat{y}^{<t>}, y^{<t>})$$

Language Model with RNN

Speech recognition

The apple and pair salad.

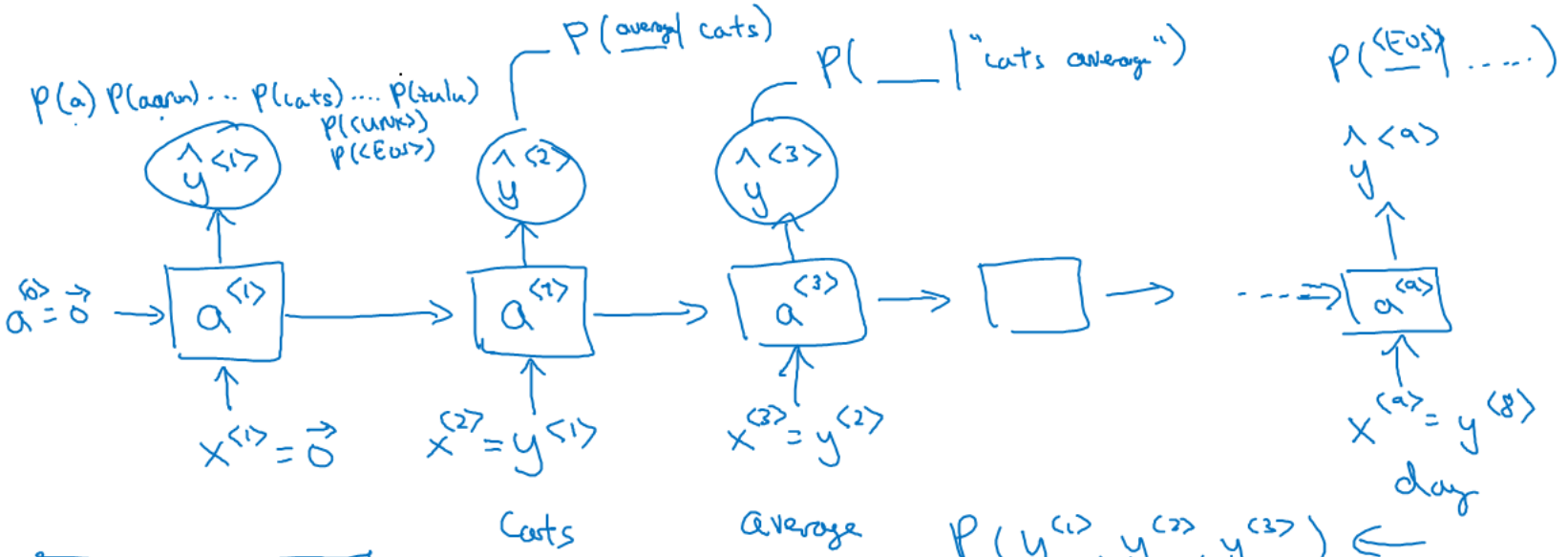
→ The apple and pear salad.

$P(\text{The apple and pair salad}) = 3.2 \times 10^{-13}$

$P(\text{The apple and pear salad}) = 5.7 \times 10^{-10}$

$P(\text{Sentence}) = ?$ $\quad P(y^{\langle 1 \rangle}, y^{\langle 2 \rangle}, \ldots, y^{\langle T_y \rangle})$

Language Model with RNN



Cats average 15 hours of sleep a day. <EOS>

$$\mathcal{L}(\hat{y}^{<t>}, y^{<t>}) = -\sum_i y_i^{<t>} \log \hat{y}_i^{<t>}$$

$$\mathcal{L} = \sum_t \mathcal{L}^{<t>}(\hat{y}^{<t>}, y^{<t>})$$

$$P(y^{<1>}, y^{<2>}, y^{<3>}) = P(y^{<1>})\, P(y^{<2>} \mid y^{<1>})\, P(y^{<3>} \mid y^{<1>}, y^{<2>})$$

Andrew Ng

Language Model with RNN

Character level laanguage model

- 장점

"알 수 없는 단어토큰"에 대해 걱정할 필요 없음

- 단점

훨씬 더 긴 배열로 모델이 끝난다.

=> 문장의 초기 부분이 문장의 뒷부분에도 어떻게 영향을 주는지, long range dependency 캡처할 때 word level만큼 좋지 않음

computationally expensive

Word level $\text{Vocabulary} = [\text{a, aaron, …, zulu, <UNK>}]$

Character level Vocabulary = [a, b, … , A, B, …, 0, … 9, …]

Problems of RNN

Vanishing Gradient

- Long range dependency capturing 약하다
- Layer 많이 쌓기 힘들다

⇒GRU, LSTM 등으로 해결

Exploding Gradient

- Parameter 엉망
- NaN 등, 숫자 아닌게 나올수도

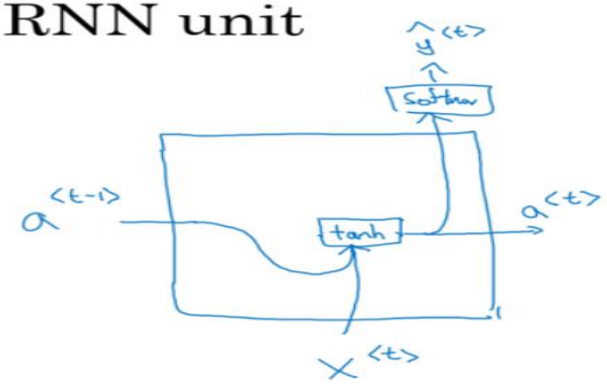
⇒Gradient clipping 통해 한계점보다 큰 벡터 조정

Vanishing이 더 해결하기 어렵고 중요한 문제

GRU

The cat, which already ate …, was full.
The cats, which already ate …, were full.

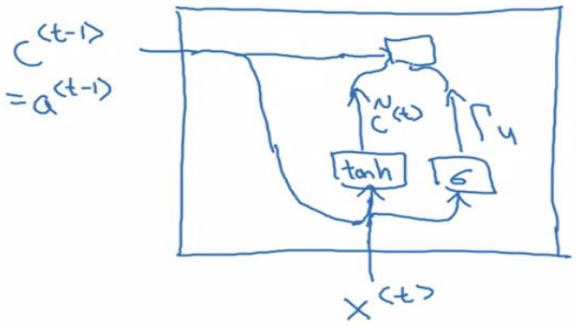
RNN unit



$$a^{\langle t \rangle} = g(W_a[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_a)$$

tanh

GRU (simplified)



c = memory cell

$\rightarrow c^{\langle t \rangle} = a^{\langle t \rangle}$

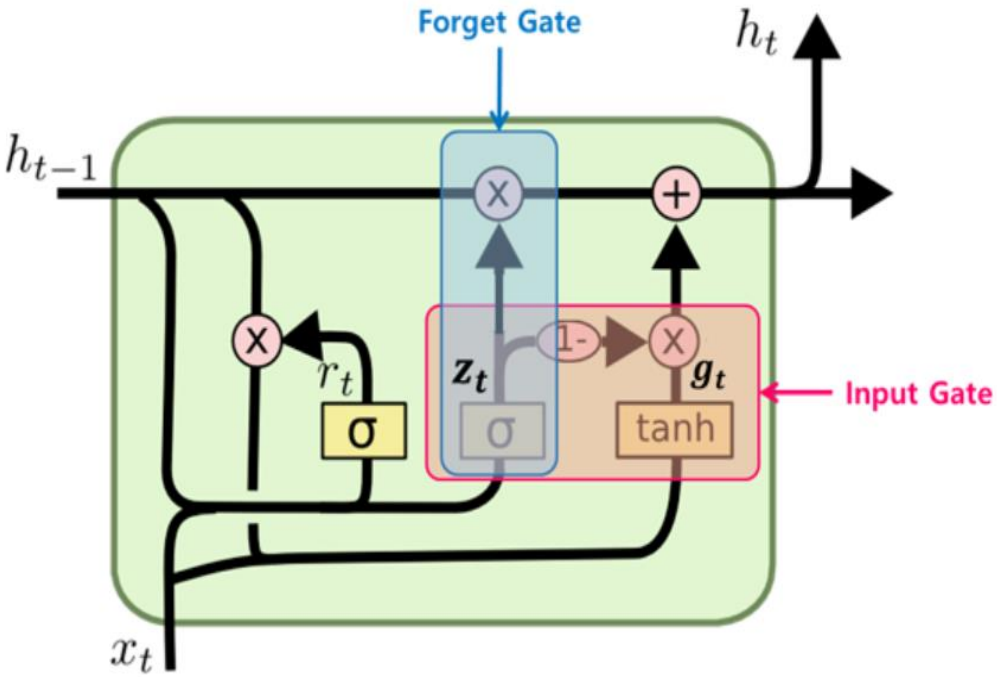
$\rightarrow \tilde{c}^{\langle t \rangle} = \tanh(W_c[c^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_c)$

$\rightarrow \Gamma_u = \sigma(W_u[c^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_u)$

Γ_u ← "update"

$c^{\langle t \rangle} = \Gamma_u * \tilde{c}^{\langle t \rangle} + (1-\Gamma_u) * c^{\langle t-1 \rangle}$

GRU



$$\mathbf{r}_t = \sigma\left(\mathbf{W}_{xr}^T \cdot \mathbf{x}_t + \mathbf{W}_{hr}^T \cdot \mathbf{h}_{t\ 1} + \mathbf{b}_r\right)$$

$$\mathbf{z}_t = \sigma\left(\mathbf{W}_{xz}^T \cdot \mathbf{x}_t + \mathbf{W}_{hz}^T \cdot \mathbf{h}_{t-1} + \mathbf{b}_z\right)$$

$$\mathbf{g}_t = \tanh\left(\mathbf{W}_{xg}^T \cdot \mathbf{x}_t + \mathbf{W}_{hg}^T \cdot \left(\mathbf{r}_t \otimes \mathbf{h}_{t-1}\right) + \mathbf{b}_g\right)$$

$$\mathbf{h}_t = \mathbf{z}_t \otimes \mathbf{h}_{t-1} + \left(1 - \mathbf{z}_t\right) \otimes \mathbf{g}_t$$

- 하나의 gate controller인 z가 **forget**과 **input** 게이트(gate)를 모두 제어한다.
  z가 1을 출력하면 forget 게이트가 열리고 input 게이트가 닫히며,
  z가 0일 경우 반대로 forget 게이트가 닫히고 input 게이트가 열린다.
  즉, 이전(t-1)의 기억이 저장 될때 마다 타임 스텝 의 입력은 삭제된다.

- GRU 셀은 output 게이트가 없어 전체 상태 벡터 h가 타임 스텝마다 출력되며,
  이전 상태 h의 어느 부분이 출력될지 제어하는 새로운 gate controller인 r이가 있다.

LSTM

GRU

$$\tilde{c}^{<t>} = \tanh(W_c[\Gamma_r * c^{<t-1>}, x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[\, c^{<t-1>}, x^{<t>}] + b_u)$$

$$\Gamma_r = \sigma(W_r[\, c^{<t-1>}, x^{<t>}] + b_r)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * c^{<t-1>}$$

$$a^{<t>} = c^{<t>}$$

LSTM

$$\tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[\, a^{<t-1>}, x^{<t>}] + b_u)$$

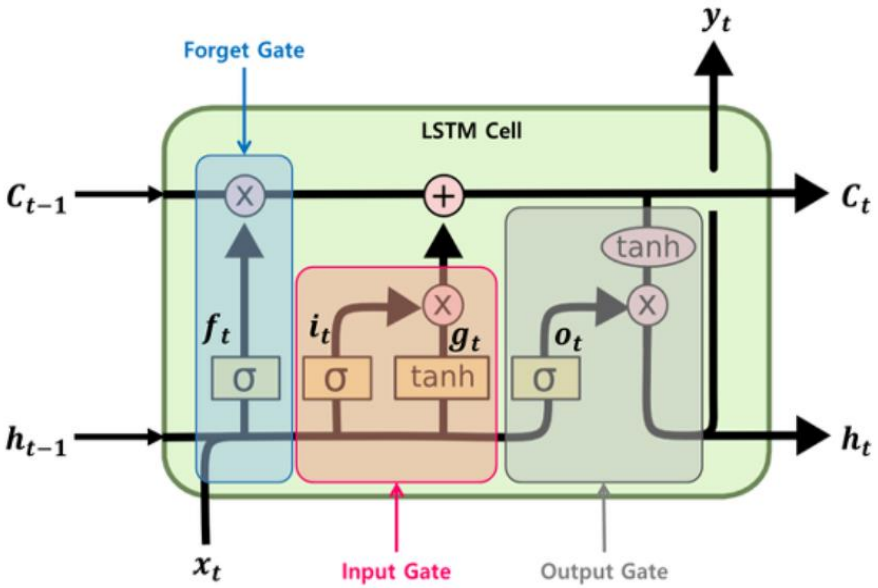
$$\Gamma_f = \sigma(W_f[\, a^{<t-1>}, x^{<t>}] + b_f)$$

$$\Gamma_o = \sigma(W_o[\, a^{<t-1>}, x^{<t>}] + b_o)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>}$$

$$a^{<t>} = \Gamma_o * \tanh(c^{<t>})$$

LSTM

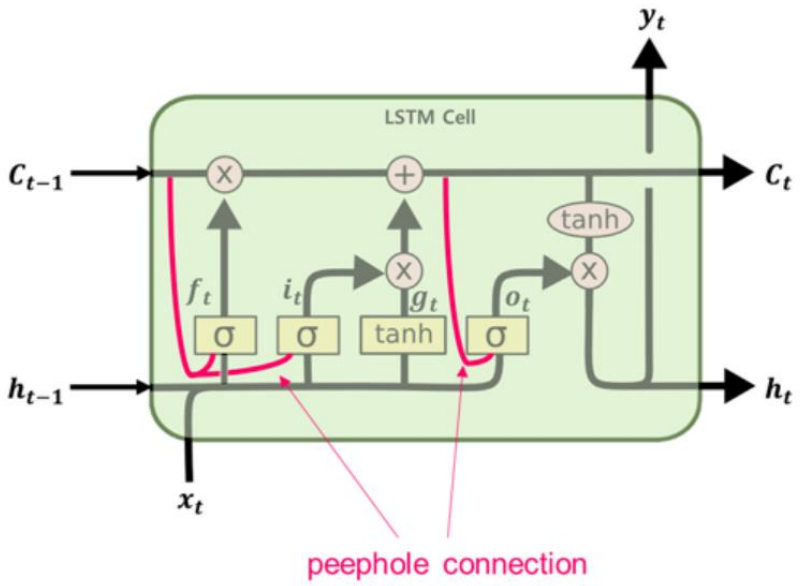


$$\mathbf{f}_t = \sigma\left(\mathbf{W}_{xf}^T \cdot \mathbf{x}_t + \mathbf{W}_{hf}^T \cdot \mathbf{h}_{t\ 1} + \mathbf{b}_f\right)$$
$$\mathbf{i}_t = \sigma\left(\mathbf{W}_{xi}^T \cdot \mathbf{x}_t + \mathbf{W}_{hi}^T \cdot \mathbf{h}_{t-1} + \mathbf{b}_i\right)$$
$$\mathbf{o}_t = \sigma\left(\mathbf{W}_{xo}^T \cdot \mathbf{x}_t + \mathbf{W}_{ho}^T \cdot \mathbf{h}_{t-1} + \mathbf{b}_o\right)$$
$$\mathbf{g}_t = \tanh\left(\mathbf{W}_{xg}^T \cdot \mathbf{x}_t + \mathbf{W}_{hg}^T \cdot \mathbf{h}_{t-1} + \mathbf{b}_g\right)$$
$$\mathbf{c}_t = \mathbf{f}_t \otimes \mathbf{c}_{t-1} + \mathbf{i}_t \otimes \mathbf{g}_t$$
$$\mathbf{y}_t, \mathbf{h}_t = \mathbf{o}_t \otimes \tanh(\mathbf{c}_t)$$

- h(단기 상태), c(장기 상태) 나누어짐
- 장기 상태에서 기억할 부분, 삭제할 부분(f), 그리고 읽어 들일 부분(i)을 학습

Forget gate : f에 의해 제어되며 장기 상태의 어느 부분을 삭제할지 제어.
Input gate : i에 의해 제어되며 g의 어느 부분이 장기 상태 에 더해져야 하는지 제어
Output gate : o는 장기 상태 의 어느 부분을 읽어서 h로 출력해야 하는지 제어

LSTM



$$\mathbf{f}_t = \sigma\left(\mathbf{W}_{cf}^T \cdot \mathbf{c}_{t-1} + \mathbf{W}_{xf}^T \cdot \mathbf{x}_t + \mathbf{W}_{hf}^T \cdot \mathbf{h}_{t-1} + \mathbf{b}_f\right)$$

$$\mathbf{i}_t = \sigma\left(\mathbf{W}_{ci}^T \cdot \mathbf{c}_{t-1} + \mathbf{W}_{xi}^T \cdot \mathbf{x}_t + \mathbf{W}_{hi}^T \cdot \mathbf{h}_{t-1} + \mathbf{b}_i\right)$$

$$\mathbf{o}_t = \sigma\left(\mathbf{W}_{co}^T \cdot \mathbf{c}_t + \mathbf{W}_{xo}^T \cdot \mathbf{x}_t + \mathbf{W}_{ho}^T \cdot \mathbf{h}_{t-1} + \mathbf{b}_o\right)$$

- 기존의 LSTM에서 gate controller(f, I, o)는 입력 x와 이전 타임스텝의 단기 상태 h만 입력으로 받는다. 하지만 핍홀(peephole) 연결 해주면 gate controller에 이전 타임 스텝의 장기 상태 c가 입력으로 추가되며, 좀 더 많은 맥락(context)를 인식할 수 있다.

GRU Vs. LSTM

2) LSTM vs GRU

- GRU

간단한 모델 : 쉽게 큰 네트워크 형성,

2개의 게이트만 사용 : 계산량 적다. 그래서 빠르다.

- LSTM

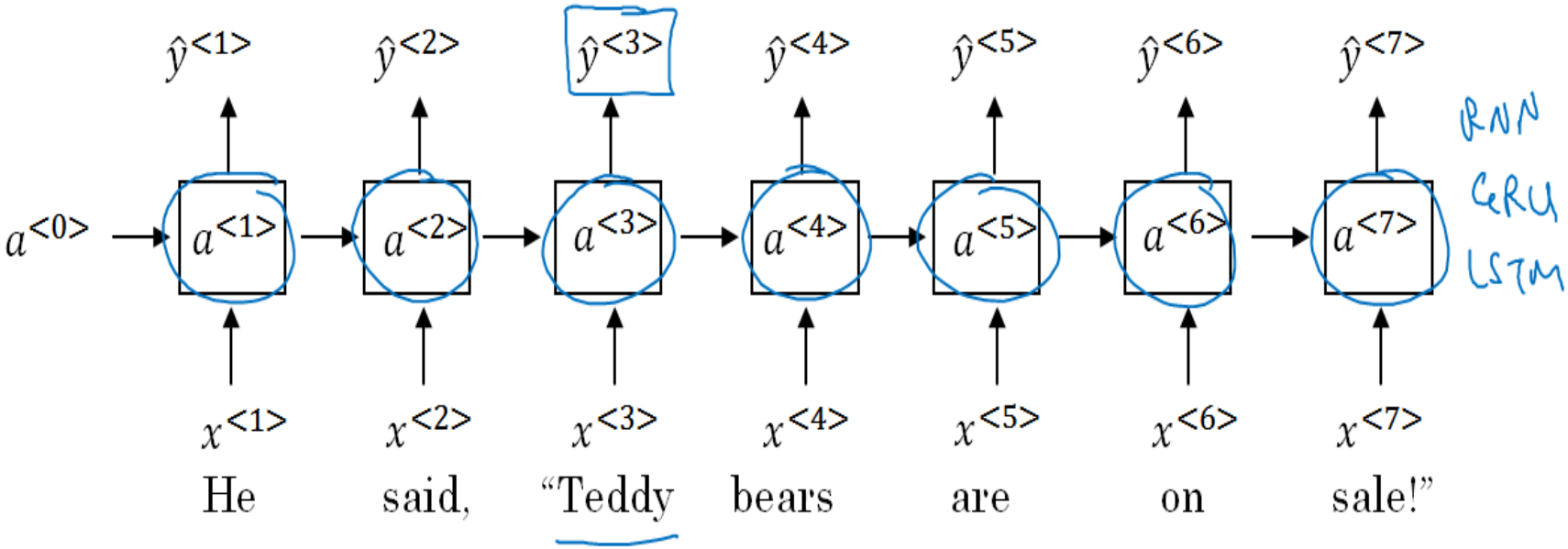
큰 모델 : 강하다.

3개의 게이트 사용 : 더 효과적이다.

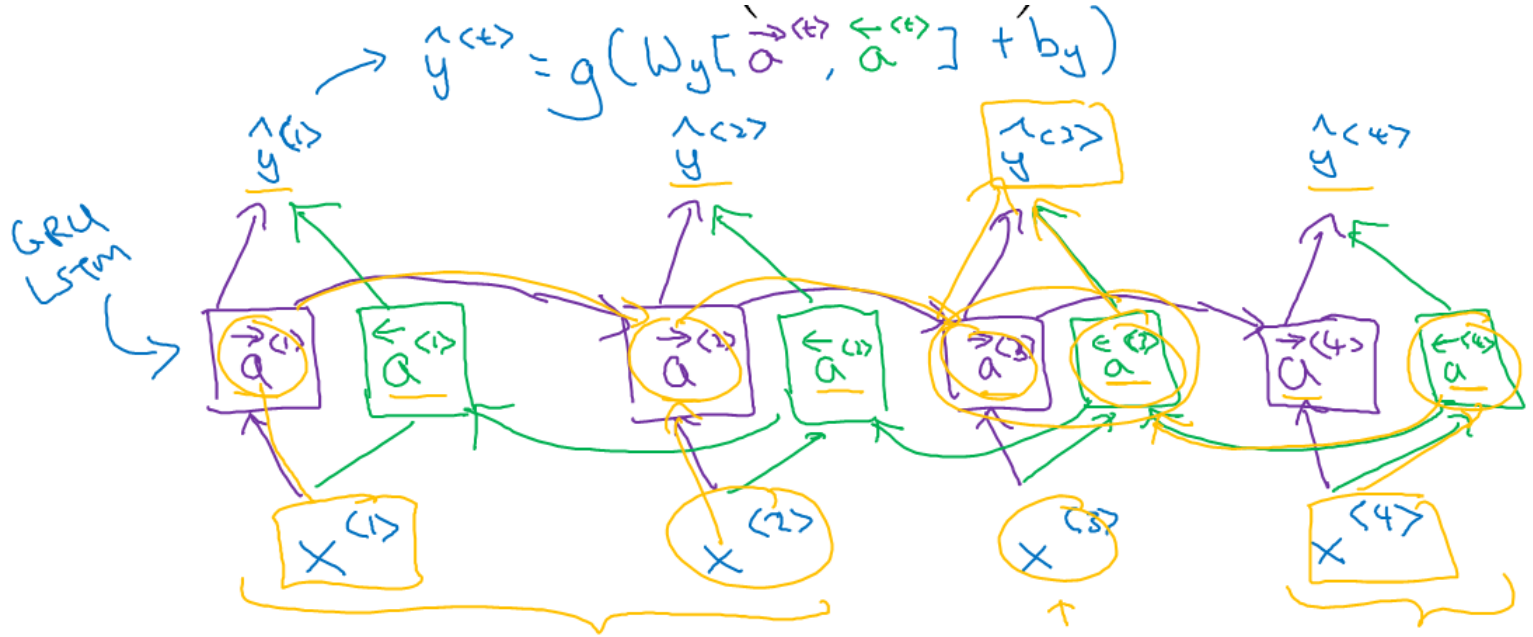
BRNN

He said, "Teddy bears are on sale!"

He said, "Teddy Roosevelt was a great President!"

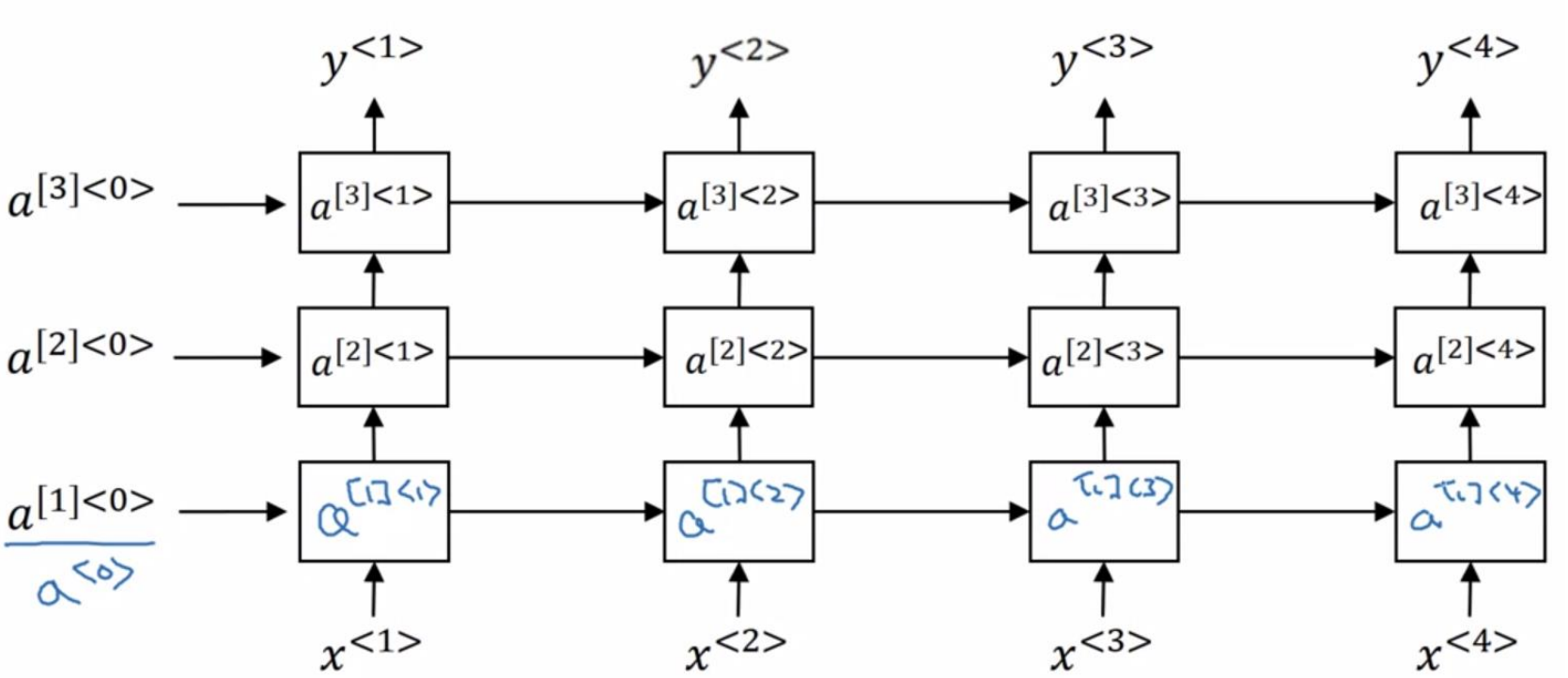


BRNN



음성 인식의 경우, 멈추는 지점이 없으면 BRNN 성능 안 좋다

Deep RNN



$$a^{[2]<3>} = g\left(W_a^{[2]}\left[a^{[2]<2>}, a^{[1]<3>}\right] + b_a^{[2]}\right)$$

너무 많이 쌓으면 안 좋다
3개도 충분히 많은 편

Reference

Coursera Sequence Models – Andrew Ng
(https://www.coursera.org/learn/nlp-sequence-models/home/week/1)

욱이의 따뜻한 감성
(https://m.blog.naver.com/ehdsnck/221778218877)

EXCELSIOR
(https://excelsior-cjh.tistory.com/185)