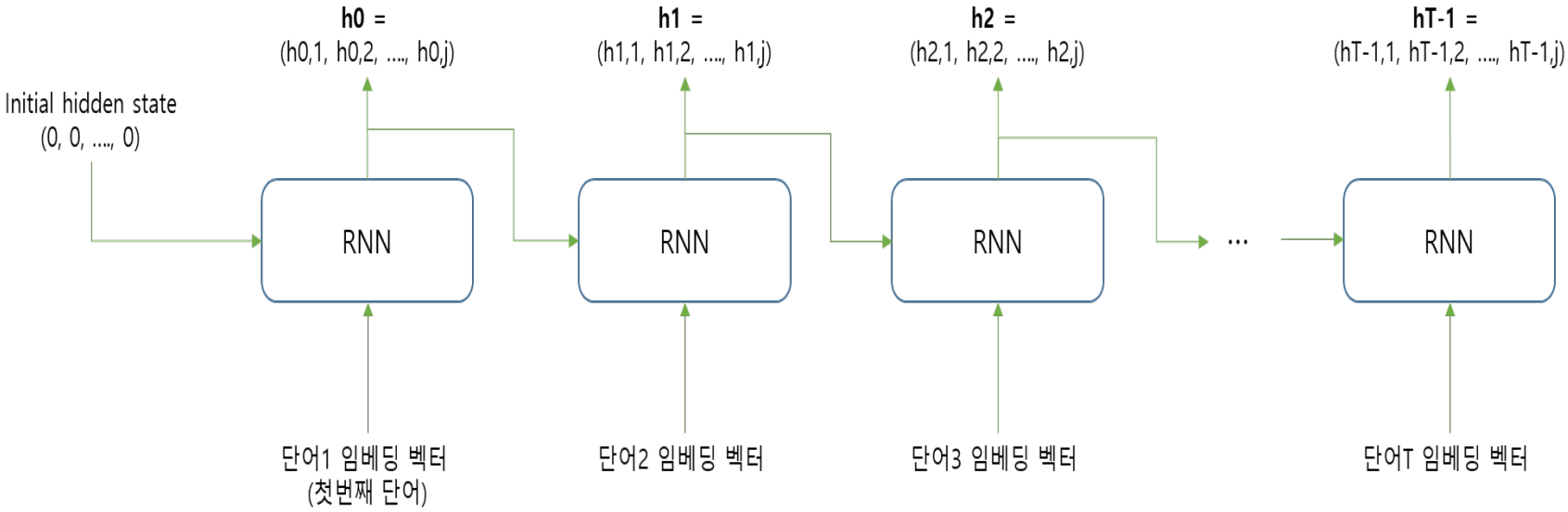
Long short-term memory

Sang Yup Lee

RNN의 문제점

- RNN 구조



LSTM

- LSTM 소개
  - RNN 기반의 신경망 알고리즘
  - Simple RNN의 문제를 보완하기 위해 제안
    - Problem of long term dependency (장기의존문제) (혹은 short-term memory라고 함)
      - 입력된 문서에서 상대적으로 오래전에 사용된 단어의 정보가 잘 전달 되지 않는다.
      - 분류 문제의 경우
        - 마지막 time step에 대한 RNN층에서 출력되는 hidden state만 다음 층으로 전달하는 경우, 입력된 문서에서 앞부분에서 사용된 단어들의 정보가 잘 반영되지 못한다는 문제도 존재

LSTM

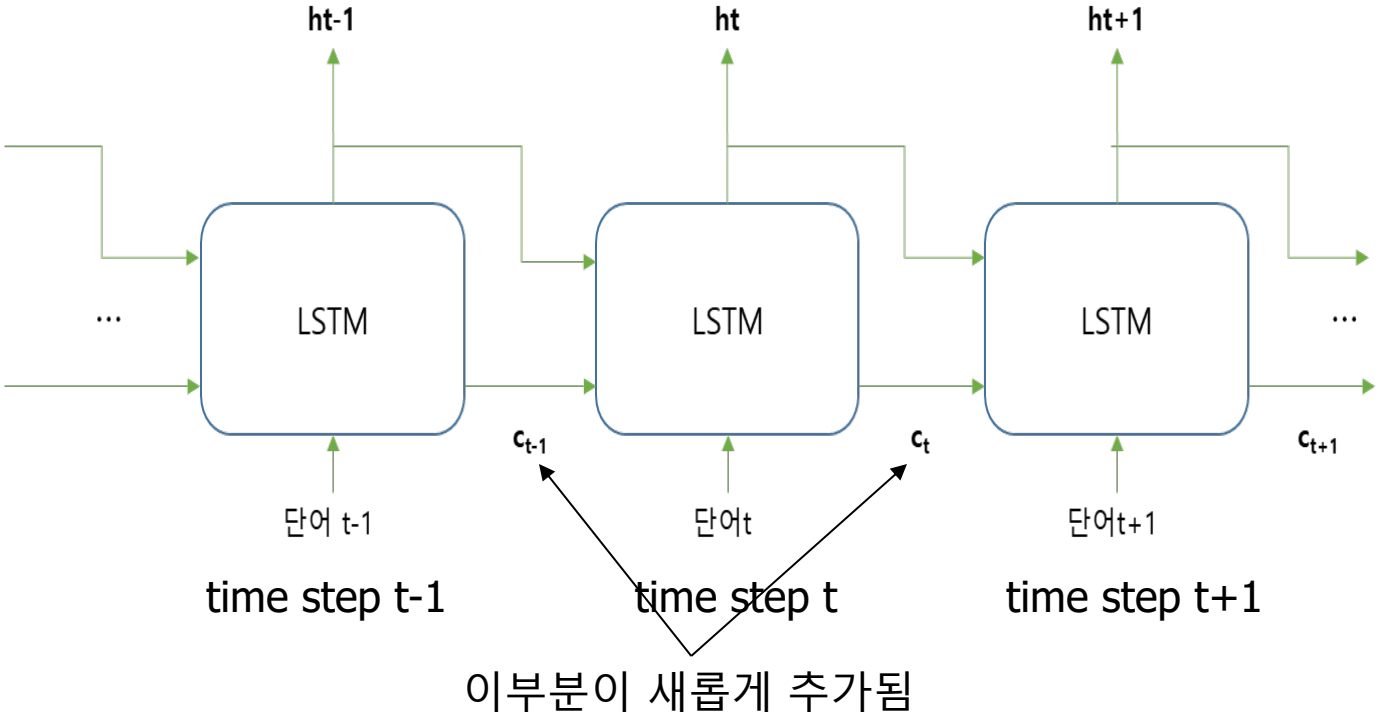
- LSTM 소개
  - RNN 기반의 신경망 알고리즘
  - Simple RNN의 문제를 보완하기 위해 제안
    - Problem of long term dependency (cont'd)
      - LSTM과 GRU 등이 제안
        - 둘의 공통점은 오래전 단어의 정보를 효율적으로 사용하기 위해서 게이트(gate)라는 것을 사용한다는 것
        - 둘의 작동원리가 비슷하고 일반적으로 LSTM 의 성능이 더 우수하기 때문에 여기서는 LSTM 중심으로 설명

LSTM

- LSTM 특징
  - LSTM과 기본 RNN의 가장 큰 차이는 LSTM은 기억셀 (memory cell)을 가지고 있고, 이를 이용하여 오래전 단어의 내용을 보다 잘 기억할 수 있다는 것
  - 하지만 기본 원리는 유사
    - 즉, time step t에서 단어 t에 대한 벡터 정보 (즉, **xt**)와 이전 은닉층 (즉, time step t-1)에서 전달하는 hidden state 정보를 담고 있는 벡터 **ht-1**를 같이 입력받는다는 방식은 동일 (즉, LSTM 층이 순차적으로 적용)
    - 여기에 기억셀 (memory cell)이 추가된 구조
      - 즉, 기존의 hidden state를 나타내는 ht와 더불어 기억셀을 나타내는 ct가 더 추가된 것이라고 생각할 수 있음
      - 기억셀은 하나의 벡터 (원소의 수 = hidden state 원소의 수 = LSTM 층에 존재하는 은닉 노드의 수)

LSTM

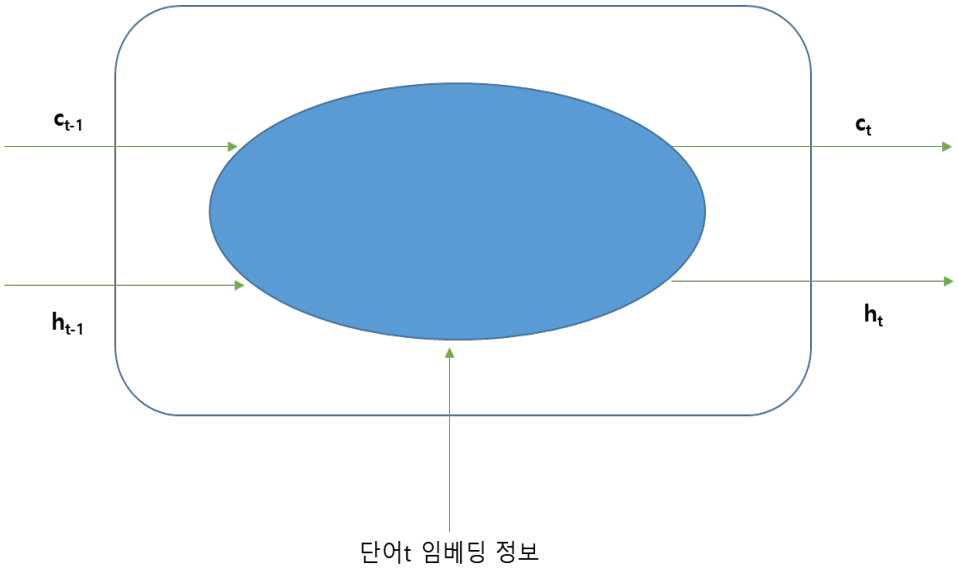
- LSTM의 구조



LSTM

- LSTM의 작동원리
  - 파란색 타원에서는 발생하는 주요한 일은 다음 두가지
    - **기억셀 업데이트: ct-1**(이전 기억셀)이 **ht-1**(이전 단계에서 전달되는 hidden state)과 **단어t의 임베딩 정보**를 사용하여 **ct**로 업데이트
    - 단어t에 대한 hidden state 값 출력: 단어t의 임베딩 정보, **ht-1**, 그리고 **ct**의 정보를 이용하여 **ht**가 계산
  - 위의 두가지의 작업을 하는데 중요한 역할을 하는 것이 게이트(gate)라고 하는 것

time step t에서의 LSTM



LSTM

- 게이트의 역할
  - 기억셀 update: 기억셀이 가지고 있는 이전 정보를 삭제(또는 기억)하거나, 새로운 정보를 추가
    - 다시 말하면, 이전 정보 중에서 정답을 맞히는데 필요한 중요한 정보는 기억하고, 필요없는 정보는 잊어버리고, 새롭게 입력되는 정보들 중에서 중요한 정보는 추가하는 역할한다고 생각
  - (업데이트된 기억셀 정보를 이용해서) 새로운 hidden state 계산
  - LSTM에는 서로 다른 역할을 하는 게이트가 3개가 존재
    - forget, input, output 게이트
  - 결국 이러한 게이트의 역할은 ct-1를 업데이트하여 ct를 계산하고, ct와 단어t의 정보, 그리고 ht-1을 이용하여 ht를 계산하는 것

LSTM

- 게이트의 역할 (cont'd)
  - 먼저 해야하는 작업: 이전 기억셀의 정보를 업데이트하는 것(즉, ct-1 → ct)
    - 이전에 가지고 있던 정보의 일부를 삭제하고, 새로운 정보를 추가
    - forget 게이트와 input 게이트를 사용

LSTM

- 게이트의 종류와 역할
  - Forget 게이트
    - 역할: 이전 기억 셀 (즉, ct-1)이 가지고 있는 정보 중에서 정답을 맞히는데 불필요한 정보는 잊어버리는 (혹은 삭제하는) 역할
    - ct-1이 가지고 있는 각 원소의 정보 중에서 몇 %를 잊어버릴 것이냐를 결정하기 위해서 0~1사이의 값 사용
    - Example: 아래와 같은 ct-1이 있다고 가정

$$\begin{bmatrix} c_{t-1,0} \\ c_{t-1,1} \\ c_{t-1,2} \\ \vdots \end{bmatrix}$$

- $c_{t-1,0}$ 은 중요한 정보를 조금만 가지고 있다 ⇒ 많은 양을 삭제
- $c_{t-1,1}$ 은 중요한 정보를 많이 가지고 있다 ⇒ 조금만 삭제

$$\begin{bmatrix} c_{t-1,0} \\ c_{t-1,1} \\ c_{t-1,2} \\ \vdots \end{bmatrix} \odot \begin{bmatrix} 0.2 \\ 0.9 \\ 0.3 \\ \vdots \end{bmatrix} = \begin{bmatrix} 0.2 * c_{t-1,0} \\ 0.9 * c_{t-1,1} \\ 0.3 * c_{t-1,2} \\ \vdots \end{bmatrix}$$

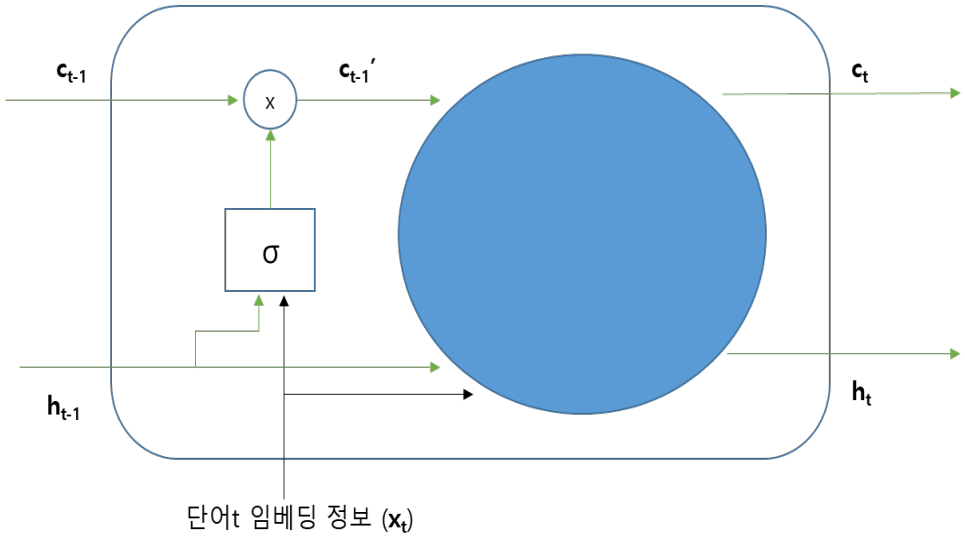
원소별로 곱하는 연산 ⇒ 아마다르 곱

LSTM

- 게이트의 종류와 역할
  - Forget 게이트 (cont'd)
    - 0~1사이의 값을 반환하는 sigmoid함수를 사용
      - ct-1 내용 (즉, 각 원소) 중 얼마만큼을 잊을 것인가를 결정하기 위해 ht-1과 단어t의 임베딩 벡터(xt)를 이용
      - $\sigma\left(\mathbf{x}_t \cdot \mathbf{W}_x^{\mathrm{f}} + \mathbf{h}_{t-1} \cdot \mathbf{W}_h^{\mathrm{f}} + \mathbf{b}^{\mathrm{f}}\right)$
        - 임베딩 벡터의 차원 = k
        - 은닉 노드의 수 = h (=히든 스테이트 벡터의 차원 = 기억셀 벡터의 차원)
        - 각 가중치 행렬의 크기는?
      - 값이 0에 가까울수록 더 많이 잊는다는 것을 의미
    - 출력되는 값
      - $\mathbf{c}_{t-1}' = \sigma\left(\mathbf{x}_t \cdot \mathbf{W}_x^{\mathrm{f}} + \mathbf{h}_{t-1} \cdot \mathbf{W}_h^{\mathrm{f}} + \mathbf{b}^{\mathrm{f}}\right) \odot \boldsymbol{c}_{t-1}$
      - 여기에서 $\mathbf{c}_{t-1}'$는 forget 게이트를 통해 업데이트된 $\mathbf{c}_{t-1}$를 의미
      - 각 원소들에 대해서 상대적으로 덜 중요한 역할을 하는 원소들의 내용을 더 많이 삭제한다는 의미

LSTM

- 게이트의 종류와 역할
  - ① Forget 게이트 (cont'd)
    - $\mathbf{c}_{t-1}{}' = \sigma\left(\mathbf{x}_t \cdot \mathbf{W}_x^{\mathrm{f}} + \mathbf{h}_{t-1} \cdot \mathbf{W}_h^{\mathrm{f}} + \mathbf{b}^{\mathrm{f}}\right) \odot \boldsymbol{c}_{t-1}$

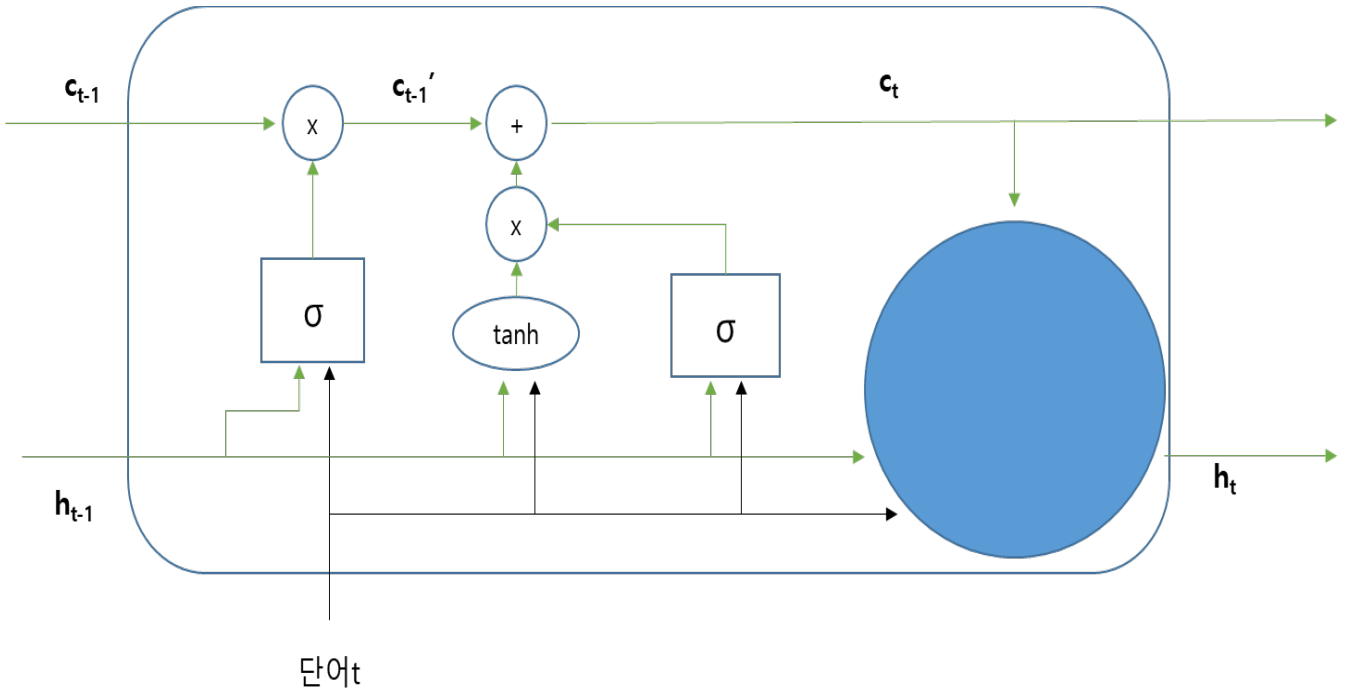


LSTM

- 게이트의 종류와 역할
  - ② Input 게이트
    - 일부의 정보가 삭제된 ct-1 (즉, $\mathbf{c}_{t-1}'$)에 새로운 정보를 추가하는 역할
    - 일단, 추가하고자 하는 정보를 계산: ht-1와 단어t의 정보를 사용
    - 새롭게 추가되는 정보들의 긍부정 역할을 나타내기 위해 -1 ~ 1의 값을 출력하는 tanh()를 사용
    - $\tanh(\mathbf{x}_t \cdot \mathbf{W}_x^{\mathrm{n}} + \mathbf{h}_{t-1} \cdot \mathbf{W}_h^{\mathrm{n}} + \mathbf{b}^{\mathrm{n}})$
    - 그대로 반영되는 것이 아니라, 정답을 맞히는데 있어서 기여하는 정도에 따라서 적용되는 비율을 다르게 함 ⇒ 이를 위해 sigmoid 함수 사용
    - $\sigma\left(\mathbf{x}_t \cdot \mathbf{W}_x^{\mathrm{i}} + \mathbf{h}_{t-1} \cdot \mathbf{W}_h^{\mathrm{i}} + \mathbf{b}^{\mathrm{i}}\right)$
    - $\mathbf{c}_t = \mathbf{c}_{t-1}' + \tanh(\mathbf{x}_t \cdot \mathbf{W}_x^{\mathrm{n}} + \mathbf{h}_{t-1} \cdot \mathbf{W}_h^{\mathrm{n}} + \mathbf{b}^{\mathrm{n}}) \odot \sigma\left(\mathbf{x}_t \cdot \mathbf{W}_x^{\mathrm{i}} + \mathbf{h}_{t-1} \cdot \mathbf{W}_h^{\mathrm{i}} + \mathbf{b}^{\mathrm{i}}\right)$

LSTM

- 게이트의 종류와 역할
  - Input 게이트 도식화 (Forget 게이트 부분도 포함)

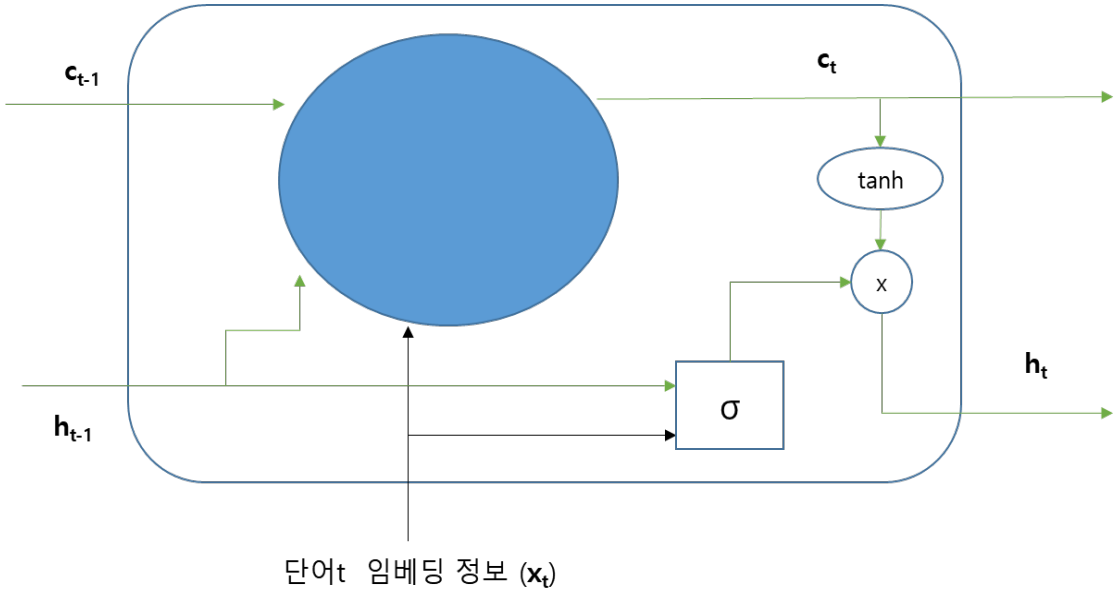


LSTM

- 게이트의 종류와 역할
  - ③ Output 게이트
    - Output 게이트의 역할은 forget 게이트와 input 게이트를 이용하여 업데이트된 기억셀의 정보, 즉 ct,를 이용하여 ht를 계산하는 것
      - ht는 현재 LSTM층에서 출력 (output)되는 값
    - Output 게이트는 ct가 갖는 원소의 값들을 조정하여 ht를 계산
    - ct 원소들의 긍부정 역할을 구분하기 위해 tanh 적용: $\tanh(\boldsymbol{c}_t)$
    - ct가 갖고 있는 원소들 중에서 정답을 맞히는데 있어 기여하는 정도를 반영하기 위해 sigmoid 함수 사용: 현재 LSTM층에 입력되는 ht-1과 단어t의 정보(xt) 사용: $\sigma(\mathbf{x}_t \cdot \mathbf{W}_x^o + \mathbf{h}_{t-1} \cdot \mathbf{W}_h^o + \mathbf{b}^o)$
    - 둘의 아마다르곱하기 수행
    - $\mathbf{h}_t = \sigma(\mathbf{x}_t \cdot \mathbf{W}_x^o + \mathbf{h}_{t-1} \cdot \mathbf{W}_h^o + \mathbf{b}^o) \odot \tanh(\boldsymbol{c}_t)$

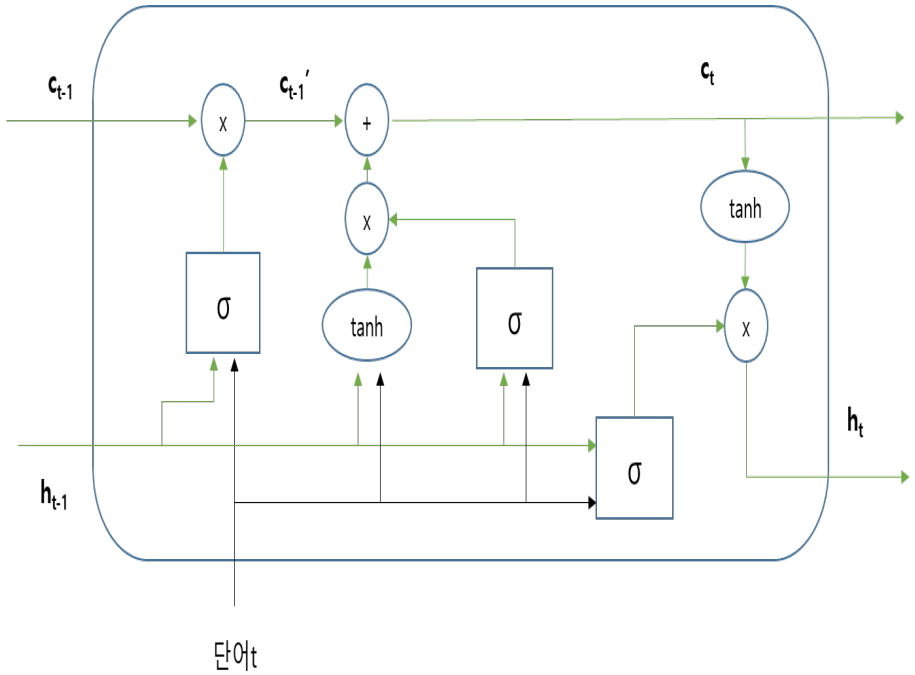
LSTM

- 게이트의 종류와 역할
  - ③ Output 게이트
    - $\mathbf{h}_t = \sigma(\mathbf{x}_t \cdot \mathbf{W}_x^o + \mathbf{h}_{t-1} \cdot \mathbf{W}_h^o + \mathbf{b}^o) \odot \tanh(\boldsymbol{c}_t)$



LSTM

- Time step t에서의 LSTM의 구조



forget gate: $\mathbf{f} = \sigma(\mathbf{x}_t\mathbf{W}_{\mathrm{x}}^{(\mathrm{f})} + \mathbf{h}_{t-1}\mathbf{W}_{\mathrm{h}}^{(\mathrm{f})} + \mathbf{b}^{(\mathrm{f})})$

새로운 정보: $\mathbf{g} = \tanh(\mathbf{x}_t\mathbf{W}_{\mathrm{x}}^{(\mathrm{n})} + \mathbf{h}_{t-1}\mathbf{W}_{\mathrm{h}}^{(\mathrm{n})} + \mathbf{b}^{(\mathrm{n})})$

input gate: $\mathbf{i} = \sigma(\mathbf{x}_t\mathbf{W}_{\mathrm{x}}^{(\mathrm{i})} + \mathbf{h}_{t-1}\mathbf{W}_{\mathrm{h}}^{(\mathrm{i})} + \mathbf{b}^{(\mathrm{i})})$

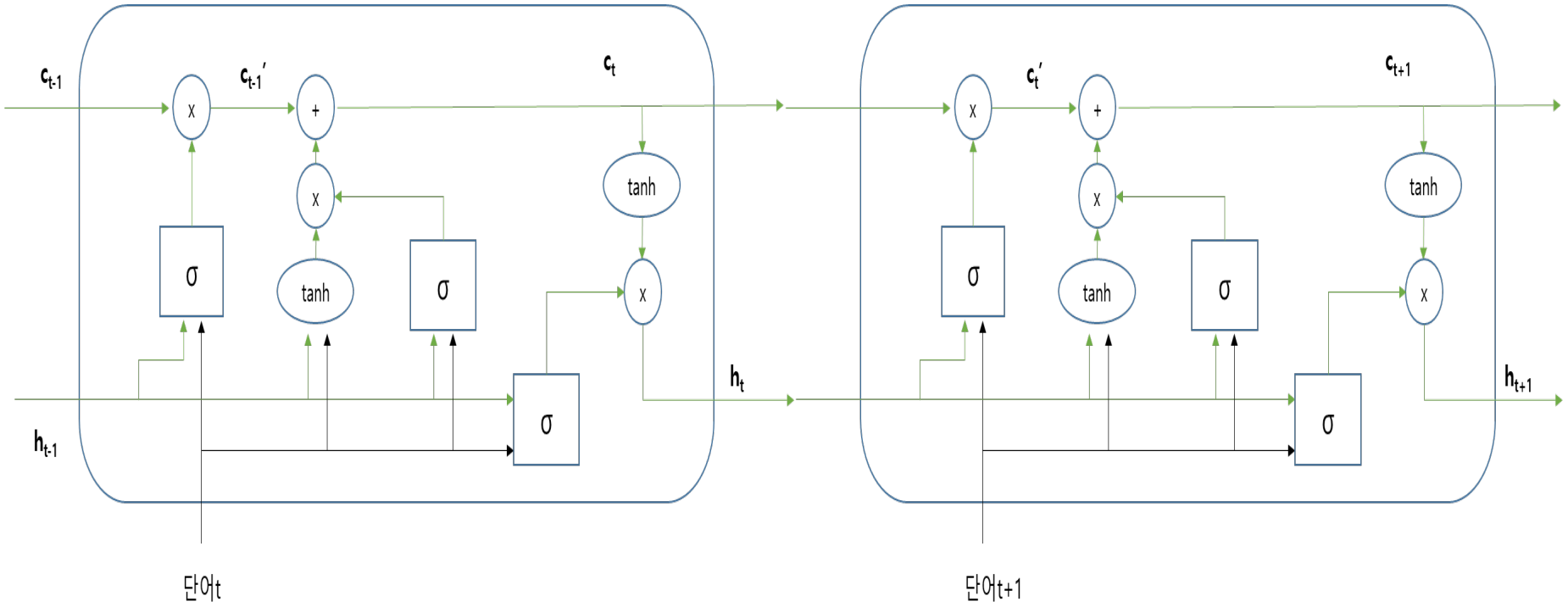
새로운 기억셀: $\mathbf{c_t} = \mathbf{f} \odot \mathbf{c_{t-1}} + \mathbf{g} \odot \mathbf{i}$

output gate: $\mathbf{o} = \sigma(\mathbf{x}_t\mathbf{W}_{\mathrm{x}}^{(\mathrm{o})} + \mathbf{h}_{t-1}\mathbf{W}_{\mathrm{h}}^{(\mathrm{o})} + \mathbf{b}^{(\mathrm{o})})$

(출력) 은닉 상태: $\mathbf{h_t} = \mathbf{o} \odot \tanh(\mathbf{c_t})$

LSTM

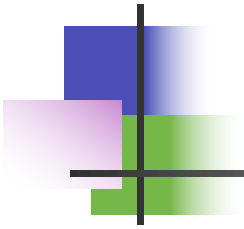
- 전체 LSTM 구조



LSTM

- LSTM 계층의 전체적인 작동 순서 정리
  - 다음과 같은 과정을 거쳐 기억셀의 정보를 업데이트한다 (즉, $\mathbf{c}_{t-1} \rightarrow \mathbf{c}_t$).
    - 기억셀의 일부 정보를 삭제한다 (잊어버린다) (forget gate 사용)
    - 기억셀에 새로운 정보를 추가한다 (input gate 사용)
  - 업데이트된 기억셀 정보를 이용해서 hidden state($\mathbf{h}_t$)를 출력한다 (output gate 사용)

- LSTM에서 다음 계층으로 출력되는 값은?
  - 일반적으로 기억 셀의 특징은 다음 계층(예, 출력층이나 또 다른 은닉층)으로 그 값이 전달되지 않는다
    - LSTM 계층 내에서만 이전 단어들의 정보를 기억하는 목적으로 사용
  - 그 다음 계층으로 전달되는 값은 hidden state, 즉, $\mathbf{h}_t$



Python coding

- 예제 파일
  - LSTM_imdb_example.ipynb 참고

LSTM

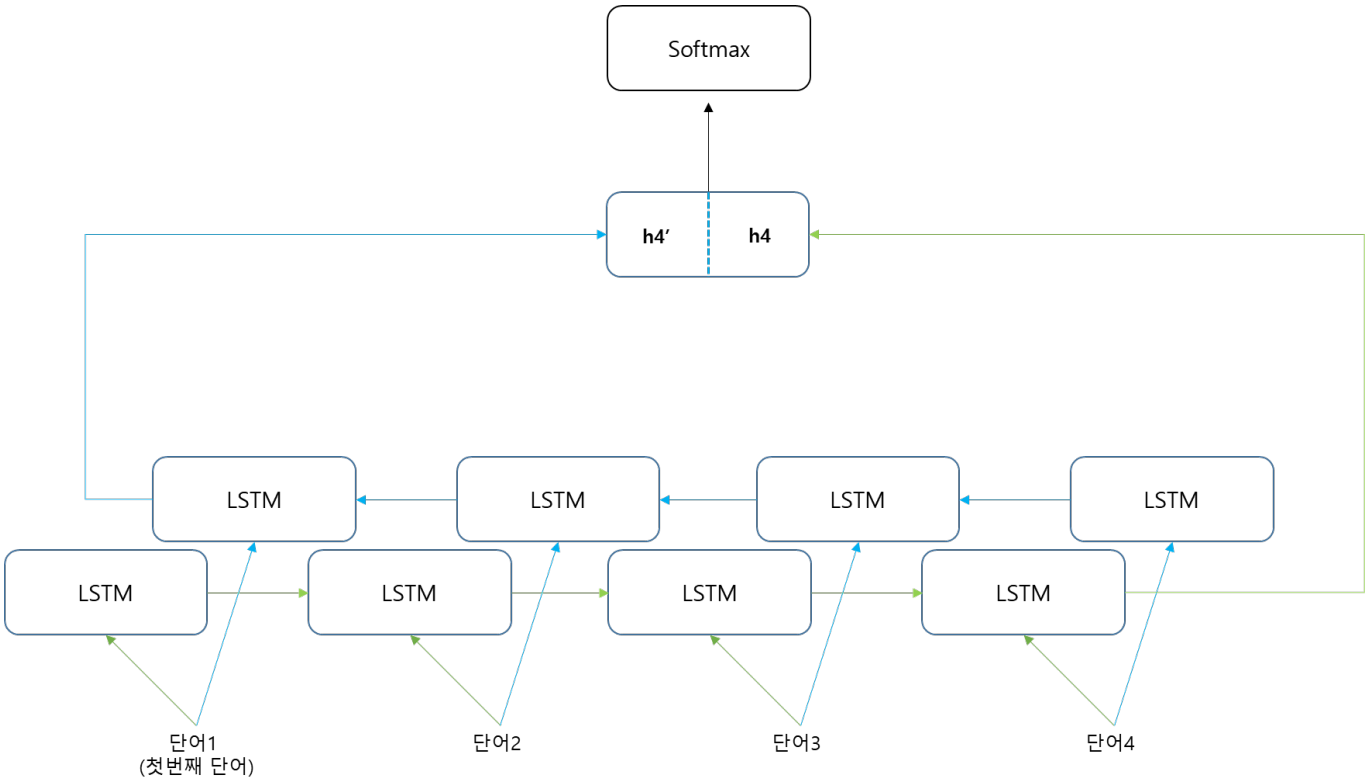
- 그 외
  - return_sequences = True 사용하는 것
  - Stacked LSTM 등은 RNN과 동일한 방법 사용
  - 한글 텍스트 분석은 “LSTM_sentiment_Korean.ipynb” 참고

Bidirectional LSTM

- Bidirectional LSTM
  - LSTM은 한 방향으로만 작동 (즉, 입력된 시퀀스 데이터의 왼쪽에서 오른쪽 방향으로)
  - 하지만, 그 다음에 나오는 단어들의 정보가 중요하거나 혹은 역순으로 문서를 읽으면 순방향으로 읽을 때 추출하지 못하는 정보를 추출할 수 있음 ⇒ 이러한 경우에는 반대방향으로 작동하는 LSTM도 같이 사용
  - 이렇게 작동 방향이 서로 반대인 두개의 서로 다른 LSTM 계층을 이용하는 LSTM을 양방향 LSTM

Bidirectional LSTM

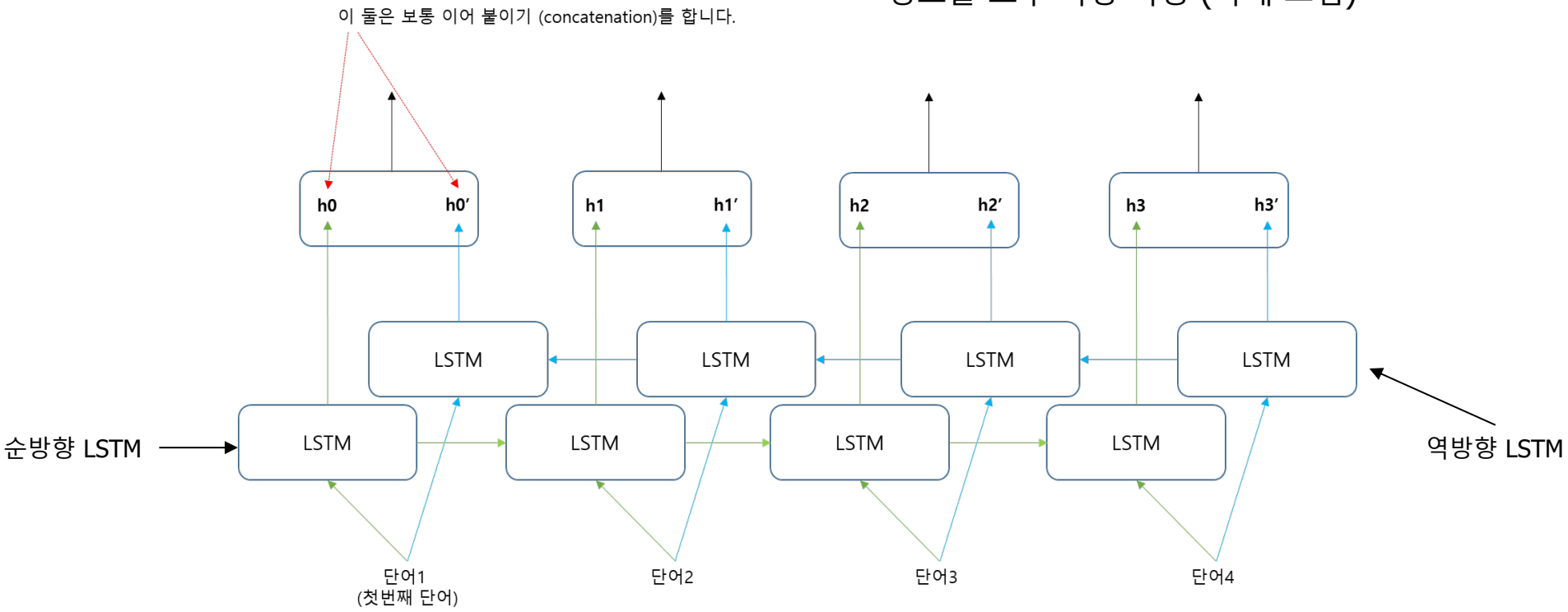
- 구조 (cont'd)



Bidirectional LSTM

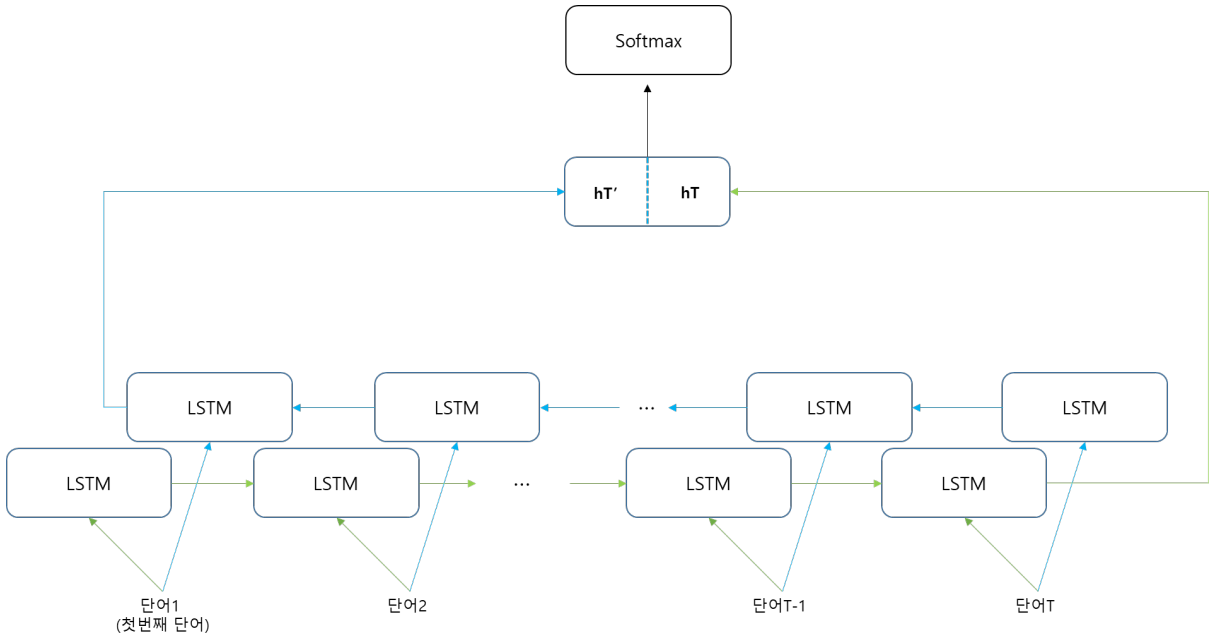
- 구조

모든 계층에서 출력하는 hidden state 정보를 모두 사용 가능 (아래 그림)



Python coding

- 예제 파일
  - Bidirectional_LSTM_example.ipynb



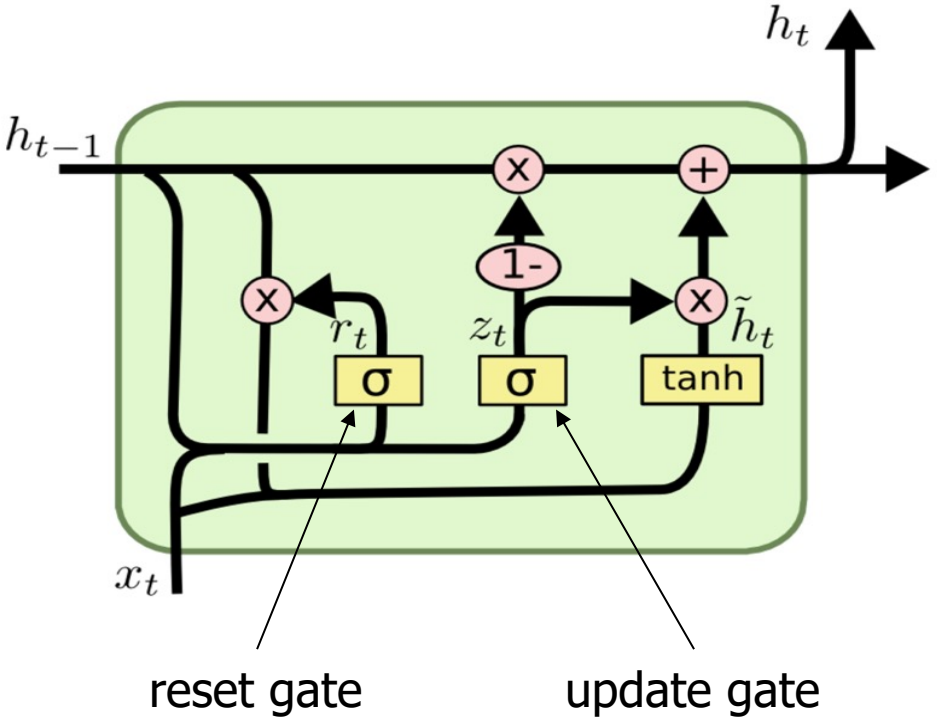
LSTM vs. GRU

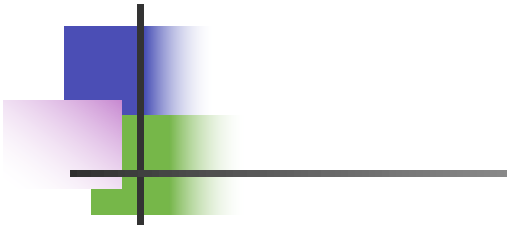
- GRU (Gated Recurrent Unit)
  - GRU는 LSTM과 마찬가지로 게이트 개념 사용
    - 기억셀을 사용하지 않음
    - reset 게이트와 update 게이트
      - hidden state 정보를 업데이트하기 위해서 reset 게이트와 update 게이트 사용
  - GRU는 LSTM 보다는 간단한 구조
    - 속도는 빠르나 정확도가 떨어짐

LSTM vs. GRU

- GRU 구조

$$
\begin{aligned}
z_t &= \sigma_g(W_z x_t + U_z h_{t-1} + b_z) \\
r_t &= \sigma_g(W_r x_t + U_r h_{t-1} + b_r) \\
\hat{h}_t &= \phi_h(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h) \\
h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h}_t
\end{aligned}
$$





Q & A