

Deep Learning

LSTM 모델 개요

(Long-Short Term Memory)

강사 양석환



RNN 모델의 문제점

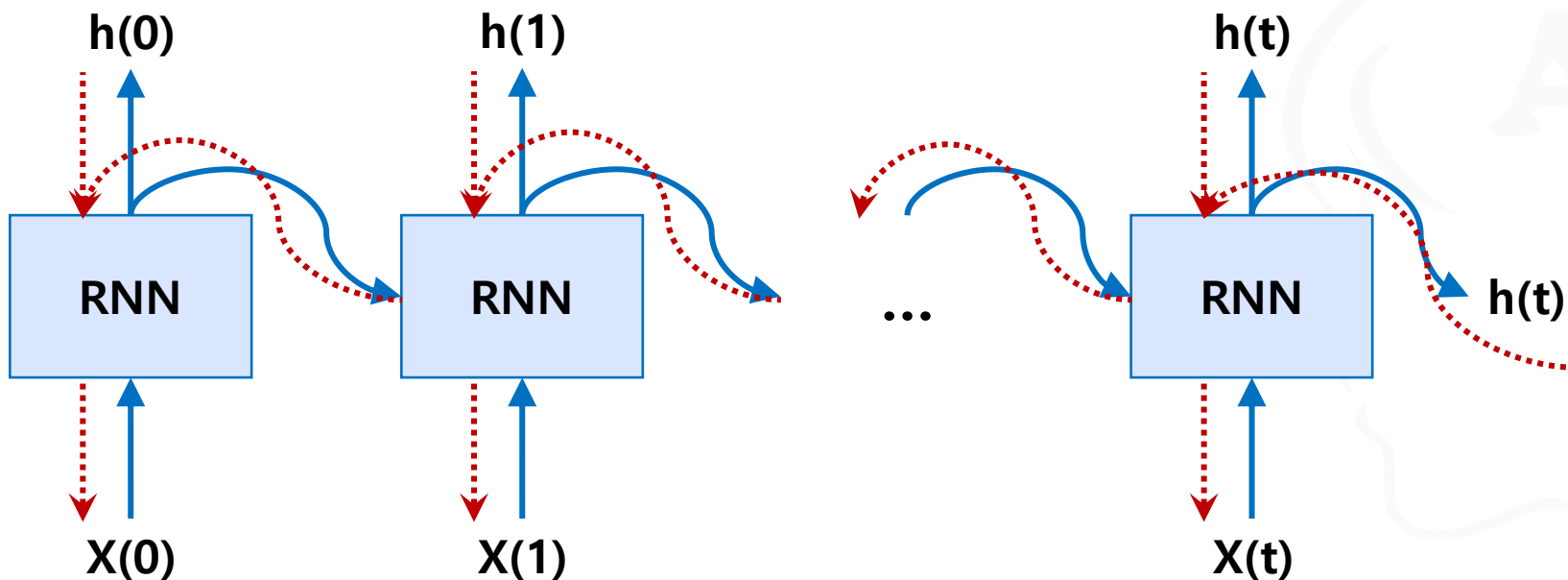


- RNN 모델의 학습 방법

- RNN 계층은 가로로 펼쳐 놓은 신경망과 동일하다고 가정할 수 있음

→ 학습 방법도 동일하게 적용할 수 있음(오류 역전파 방식 등) → **BPTT 방법**

BPTT: Back Propagation Through Time



- BPTT (Back Propagation Through Time)의 문제점

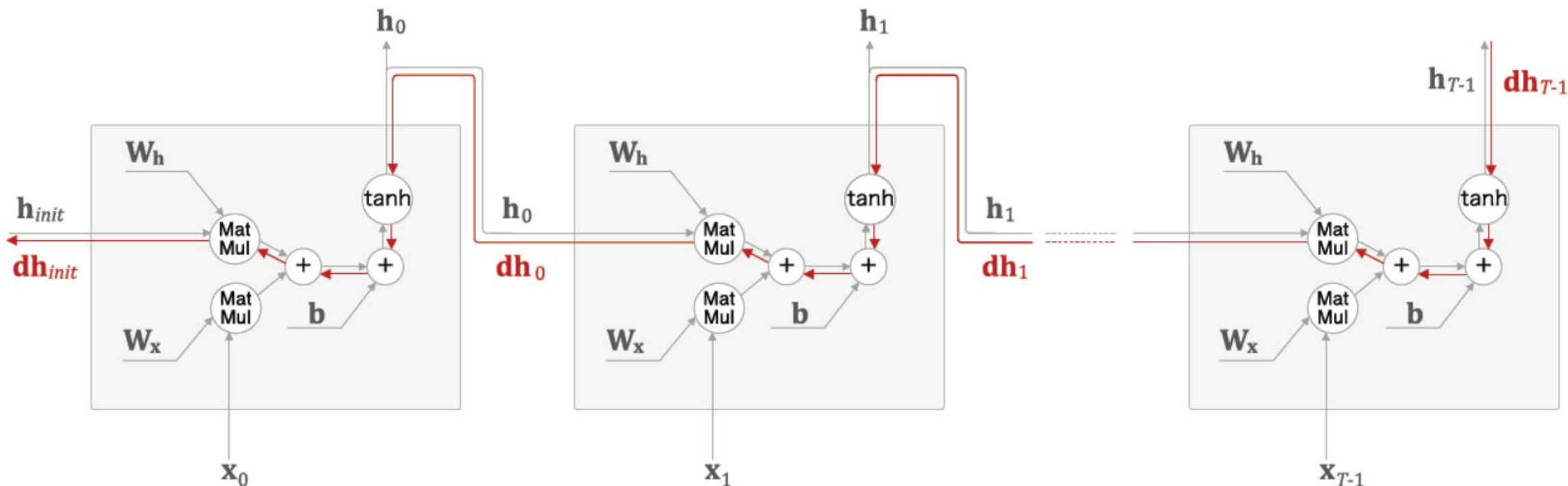
- **긴** 시계열 데이터를 학습할 때 → 장기기억 문제

- 시계열 데이터의 시간 크기가 커질수록 BPTT가 **소비하는 컴퓨팅 자원도 비례하여 증가함**
- 시간의 크기가 커질수록 역전파 시의 비율 조정을 위한 기울기가 불안정해짐
→ 기울기 폭발 또는 기울기 소실 문제 발생

BPTT를 이용하여 기울기를 구할 때, 매 시각의 RNN 계층의 중간 데이터를 메모리에 유지해 두어야 함
→ 시계열 데이터가 길어질 수록 계산량 및 메모리의 사용량이 증가함

- **개선을 위하여 Truncated BPTT 기법 제안**

- 기울기 폭발, 소실 문제는 왜 발생하는가?



RNN 계층에서 시간 방향으로의 기울기 전파

학습을 위해 역전파 할 때, “ $\tanh \rightarrow + \rightarrow \text{MatMul}$ ”의 순서로 연산을 수행함

- 역전파 할 때, “tanh → + → MatMul”의 순서로 연산을 수행함

- “+” : 그냥 기울기의 정보를 전달할 뿐 기울기의 크기가 바뀌지 않음

- “tanh” : 역전파 할 때, 미분 연산을 수행함

- $y = \tanh(x)$ 일 때의 미분은 $\frac{\partial y}{\partial x} = 1 - y^2$

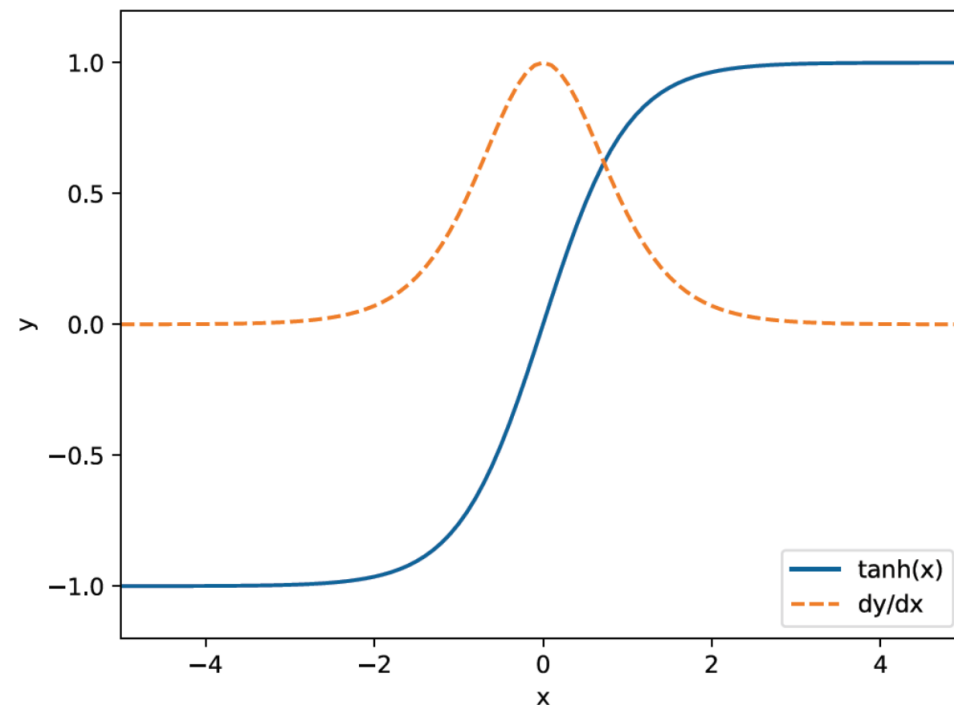
- 각각 그래프를 그려보면

- 값의 범위는 1.0 이하

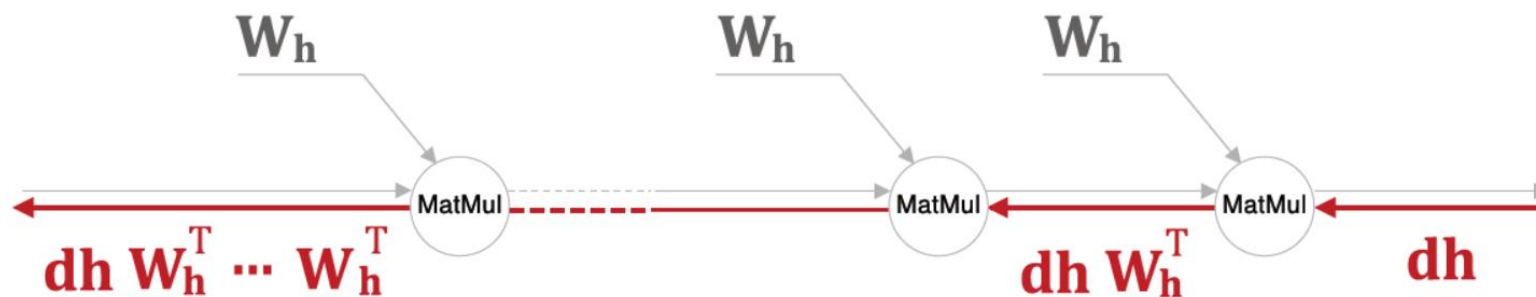
- x 가 0에서 멀어질수록 작아짐

- 역전파를 반복할수록(= \tanh 함수를 지나칠수록)

- 계속 작아짐

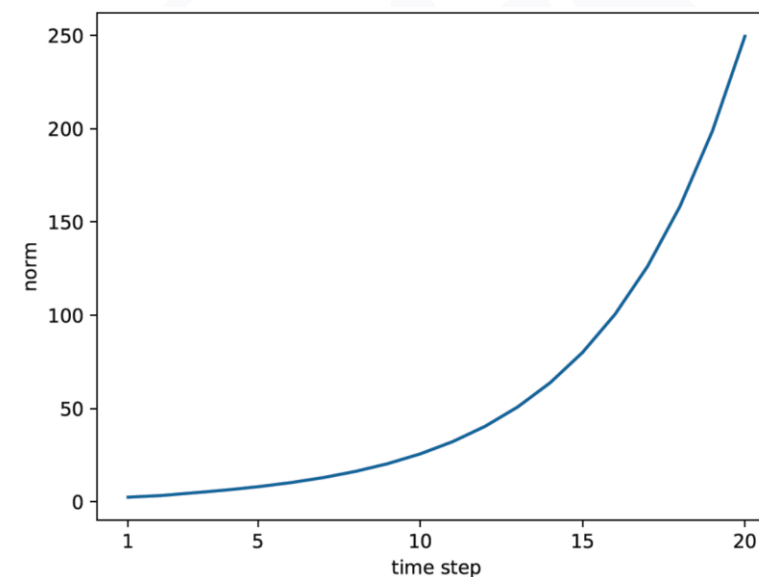


- “MatMul” : $\tanh(x)$ 를 무시하고 “MatMul”만 집중해서 보면..

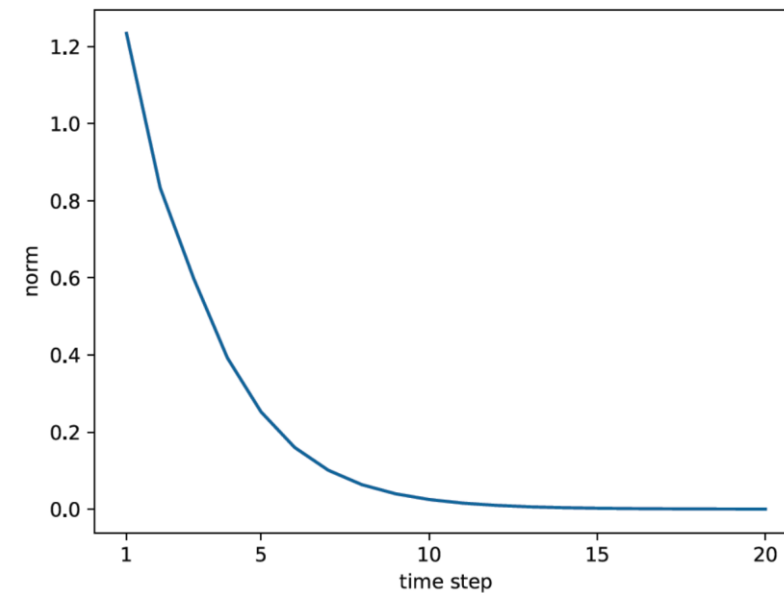


- 몇 번을 반복하더라도 가중치 W_h 는 역전파를 지나는 도중에 변하지 않고 계속 그대로 사용됨
→ 반복적으로 곱해질 경우의 그래프를 그려보면 →

지수증가 → 기울기 폭발



- W_h 의 초기 값을 절반으로 줄여서 다시 반복시켜보면
지수감소 → 기울기 소실
- W_h 가 스칼라 값이면 $W_h > 1$ 이면 지수증가,
 $W_h < 1$ 이면 지수감소 하게 됨
- W_h 가 행렬이면 “특잇값”에 따라 지수증가/감소가 결정됨
 - 행렬의 특잇값: 데이터가 얼마나 퍼져있는지(분포)에 따라 변화



• 기울기 폭발의 대책

• 기울기 클리핑(Gradients Clipping)

- 신경망에서 사용되는 모든 매개변수에 대한 기울기를 하나로 처리한다고 가정함 (\hat{g} 로 표시)
- 기울기의 L2-Norm이 임계값(Threshold)을 초과하면 두 번째 줄의 수식과 같이 기울기를 수정
- 간단한 알고리즘이지만 많은 경우에 잘 작동함

if $\|\hat{g}\| \geq threshold :$

$$\hat{g} = \frac{threshold}{\|\hat{g}\|} \hat{g}$$

- 기울기 소실의 대책

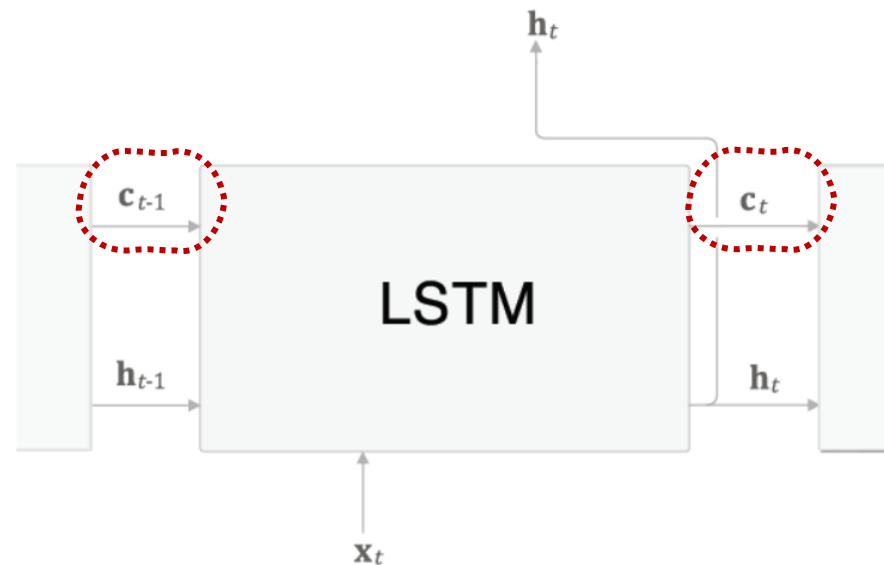
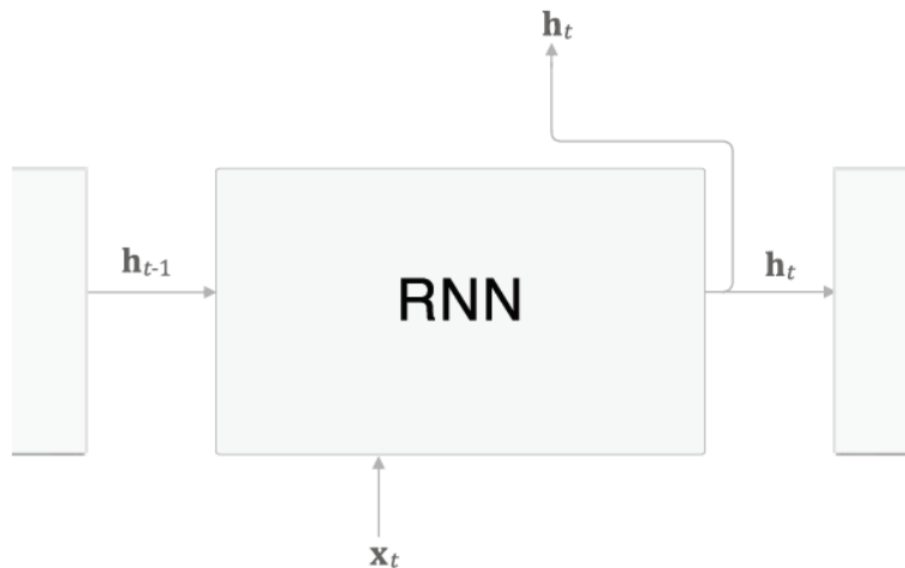
- 간단한 방법으로는 대책을 찾을 수 없음(RNN 모델의 근본부터 뜯어고쳐야..)
- “**기울기 소실이 일어나지 않도록 그 값을 직접 제어해 주겠다**”라는 아이디어
 - **제어를 위한 Gate**를 추가
 - 다양한 알고리즘이 등장했으며 그 중 대표적인 알고리즘이 **LSTM, GRU**



LSTM 모델

AiDA
Lab.

- RNN과 LSTM의 구조 비교



- LSTM 계층의 인터페이스에는 c 라는 경로가 있다!
 - c : Cell 또는 Memory Cell

• Memory Cell의 특징

- 데이터를 자기 자신으로만(LSTM 계층 내에서만) 주고 받음
→ LSTM 계층 내에서만 완결되고, 다른 계층으로는 출력하지 않음
- Memory Cell c_t 에는 시각 t 에서의 LSTM의 기억을 저장함
 - 과거로부터 시각 t 까지에 필요한 모든 정보가 저장되어 있다고 가정함
 - 또는 그렇게 저장되도록 학습 수행 → c_t 의 정보가 지속적으로 갱신되고 있음을 의미
- 그리고 c_t 의 정보를 바탕으로 외부 계층과 다음 시각의 LSTM 계층에 h_t 를 계산하여 출력함

- Memory Cell은 어떤 값을 계산하는가?

- c_t 는 3개의 입력 (c_{t-1}, h_{t-1}, x)로 부터 어떤 계산을 수행함
- 출력되는 은닉상태 $h_t = \tanh(c_t)$
 - c_t 의 각 요소에 \tanh 를 적용함을 의미
- 은닉상태 h_t 는 단순히 c_t 에 \tanh 를 적용한 것 뿐이다
 - c_t 와 h_t 를 구성하는 원소의 수는 같다

그리고 h_t 는 c_t 의
영향을 받는다는 소리임



- Memory Cell은 무엇으로부터 제어를 받는가?

- 3개의 Gate와 1개의 Updater의 제어를 받음

- Output Gate

- Forget Gate

- Input Gate

- Updater → New Memory Cell

Gate: 데이터의 흐름을 제어함. 열기/닫기 및 어느 정도 열지를 조절

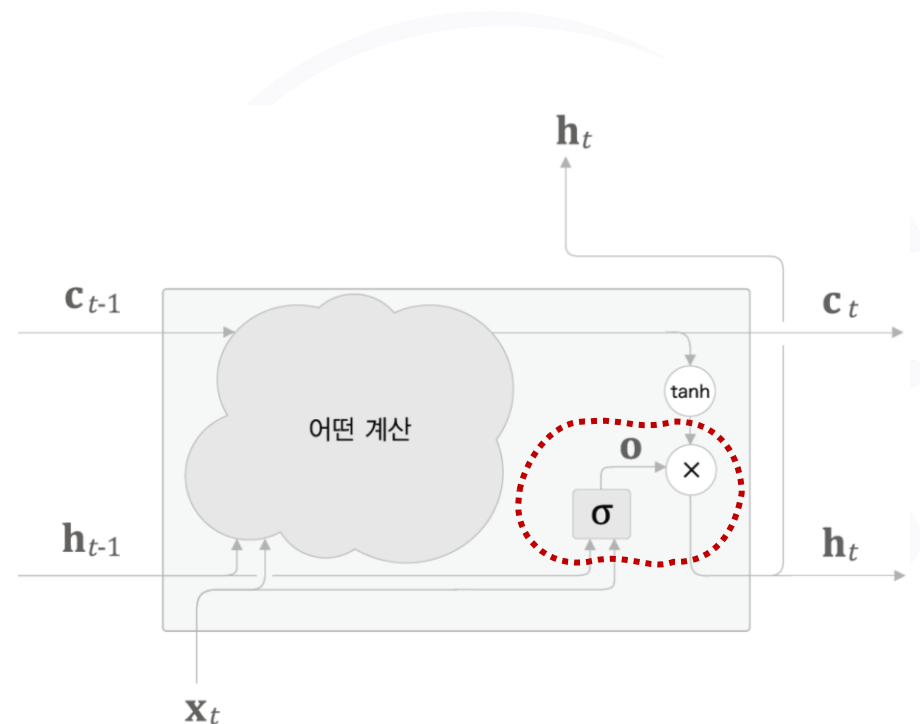
열림 상태

• Output Gate

- $\tanh(c_t)$ 의 각 원소에 대하여 “그것이 다음 시각의 은닉상태에 얼마나 중요한가?”를 조정
- 다음 은닉상태 h_t 의 출력을 담당하므로 Output Gate라고 부름
- Output Gate의 열림 상태는 입력 x_t 와 이전 상태 h_{t-1} 로부터 구함

$$\mathbf{o} = \sigma(\mathbf{x}_t \mathbf{W}_x^{(o)} + \mathbf{h}_{t-1} \mathbf{W}_h^{(o)} + \mathbf{b}^{(o)})$$

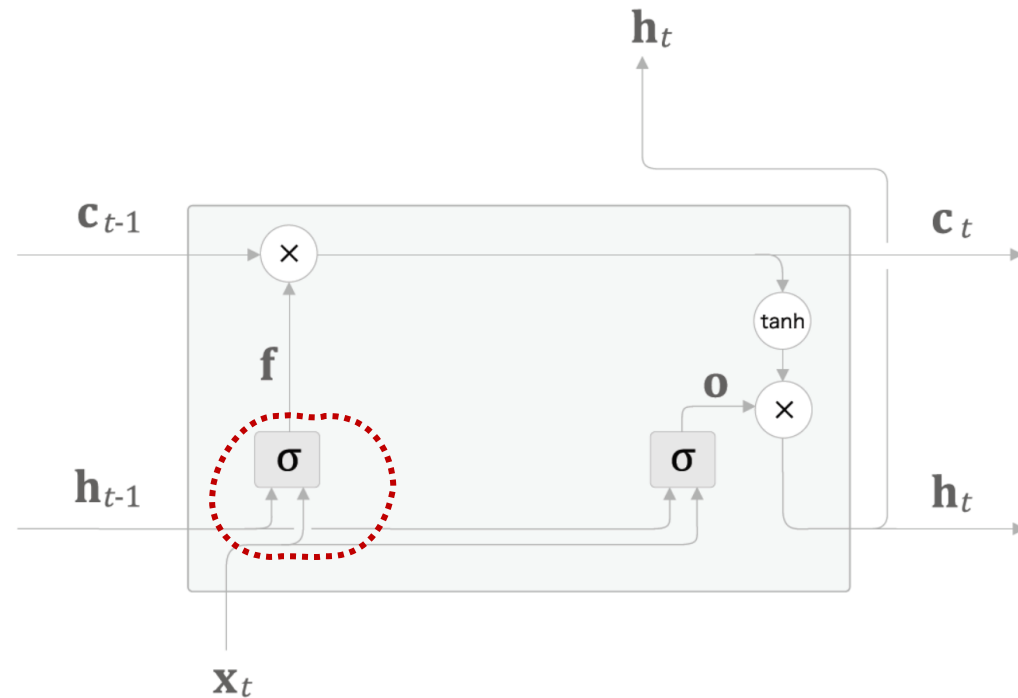
$$\mathbf{h}_t = \mathbf{o} \odot \tanh(\mathbf{c}_t)$$



- Forget Gate

- c_{t-1} 의 기억 중에서 불필요한 “기억의 어떤 것을 잊을까?”를 명확하게 지시하는 Gate
- LSTM 계층에 Forget Gate를 추가하면..

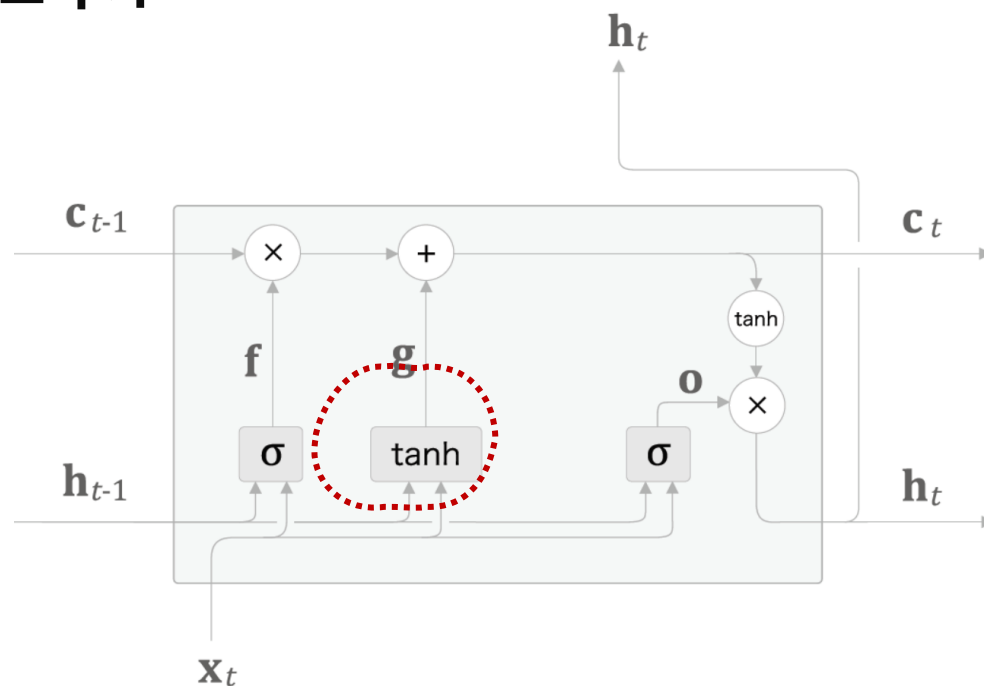
$$\mathbf{f} = \sigma(\mathbf{x}_t \mathbf{W}_x^{(f)} + \mathbf{h}_{t-1} \mathbf{W}_h^{(f)} + \mathbf{b}^{(f)})$$



- 새로운 Memory Cell

- 잊어야 할 기억은 삭제되었는데 새로운 기억은?
- 새로운 기억을 Memory Cell에 추가하기 위해서 tanh 노드 추가

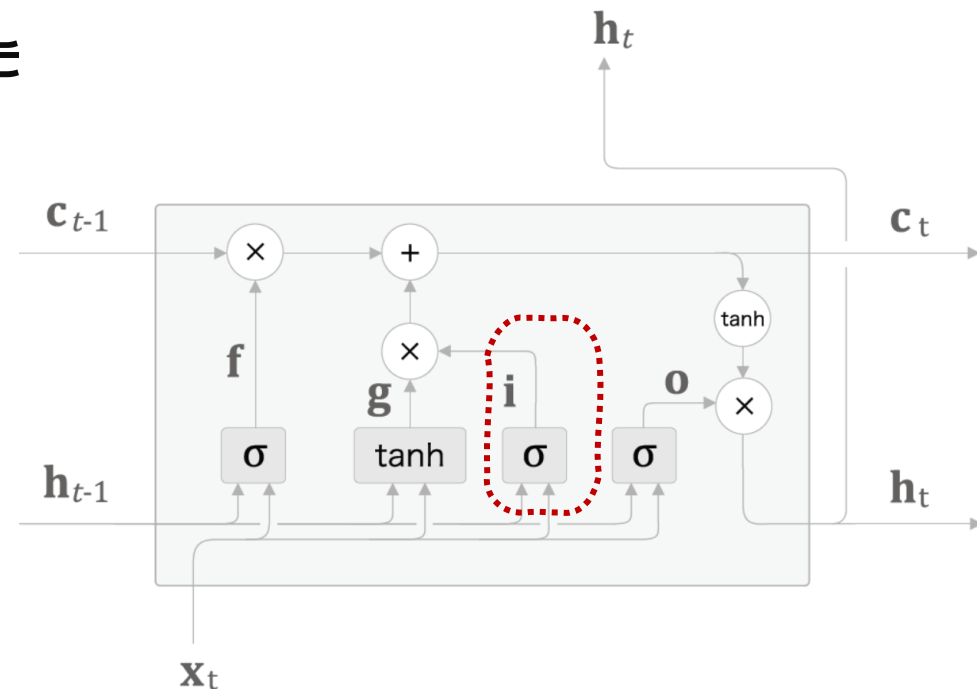
$$\mathbf{g} = \tanh(\mathbf{x}_t \mathbf{W}_x^{(g)} + \mathbf{h}_{t-1} \mathbf{W}_h^{(g)} + \mathbf{b}^{(g)})$$



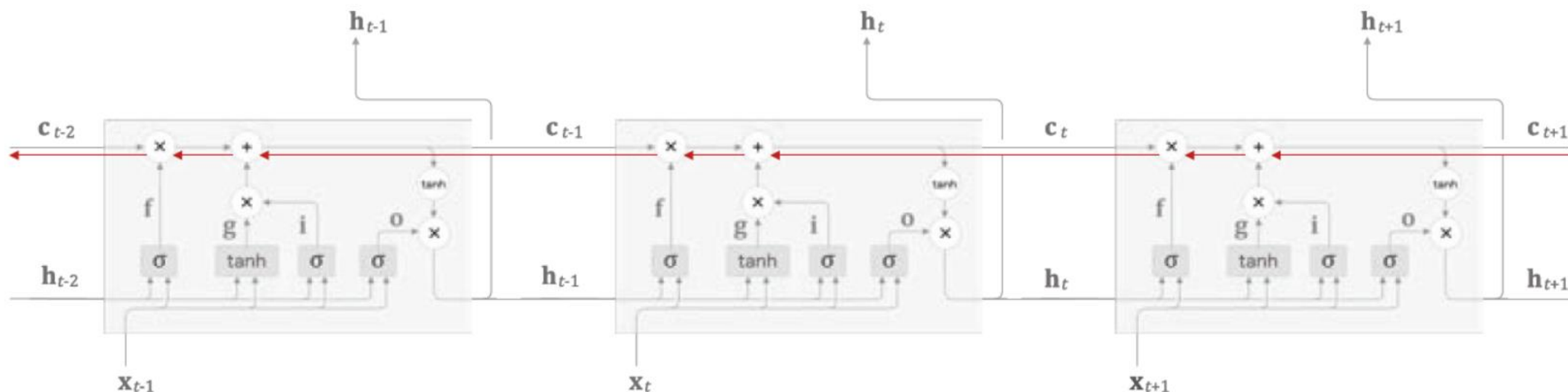
• Input Gate

- g에 Gate를 하나 추가함 → 입력 게이트(input gate)
- g의 각 원소가 새로 추가되는 정보로서의 가치가 얼마나 큰지를 판단
- 새 정보를 무작정 받는 것이 아니라 적절히 취사선택하는

$$\mathbf{i} = \sigma(\mathbf{x}_t \mathbf{W}_x^{(i)} + \mathbf{h}_{t-1} \mathbf{W}_h^{(i)} + \mathbf{b}^{(i)})$$



- 어떻게 해서 Gate들과 새로운 Memory Cell이 기울기 소실을 해결하는가?



- Memory Cell “c”의 역전파에 주목해보면...
- ‘+’와 ‘x’ 노드 만을 지나게 됨

- 3개의 Gate와

1개의 업데이트 메모리셀로 구성된

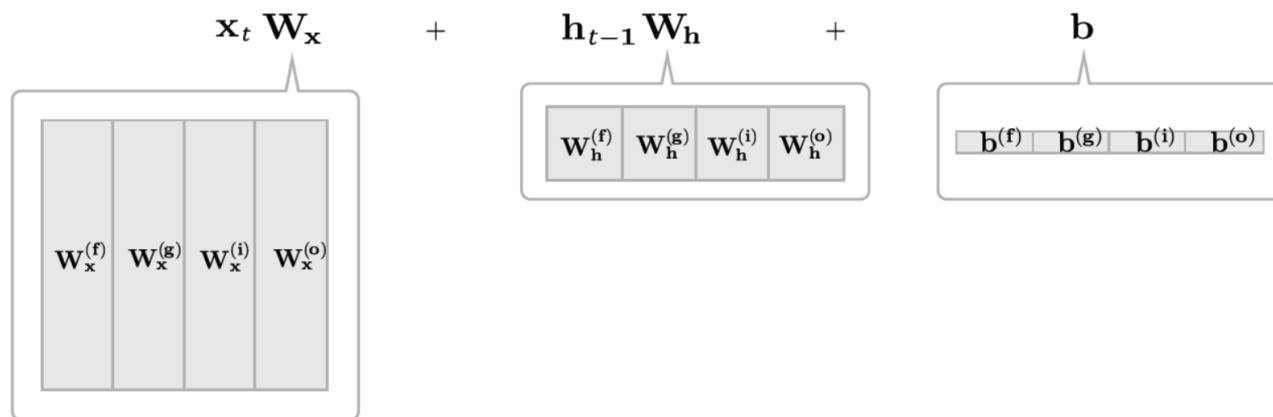
4개의 수식을 모아

단 한 번의 아핀변환으로 계산

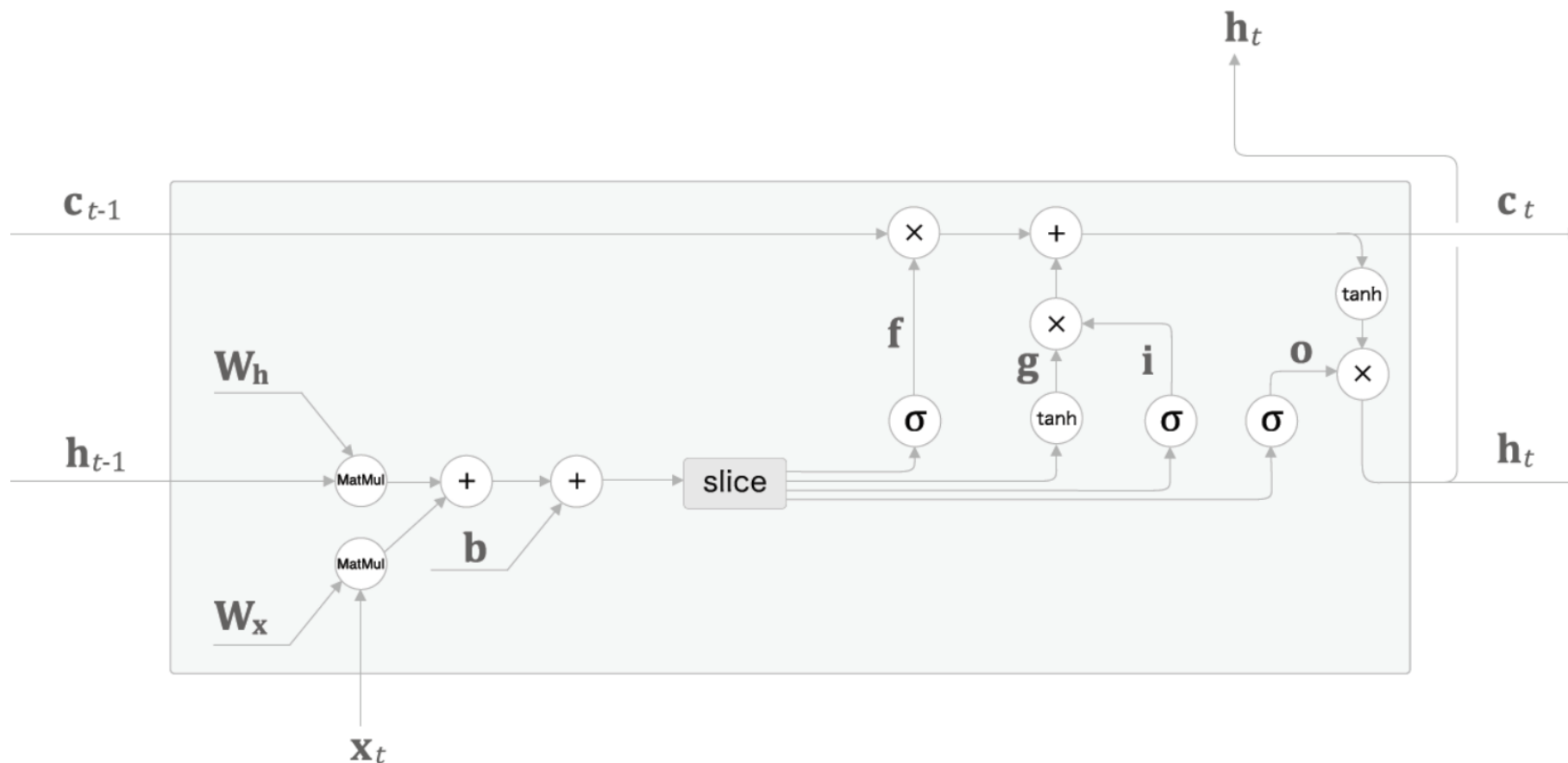
$$\begin{array}{rclclcl} x_t W_x^{(f)} & + & h_{t-1} W_h^{(f)} & + & b^{(f)} \\ x_t W_x^{(g)} & + & h_{t-1} W_h^{(g)} & + & b^{(g)} \\ x_t W_x^{(i)} & + & h_{t-1} W_h^{(i)} & + & b^{(i)} \\ x_t W_x^{(o)} & + & h_{t-1} W_h^{(o)} & + & b^{(o)} \end{array}$$

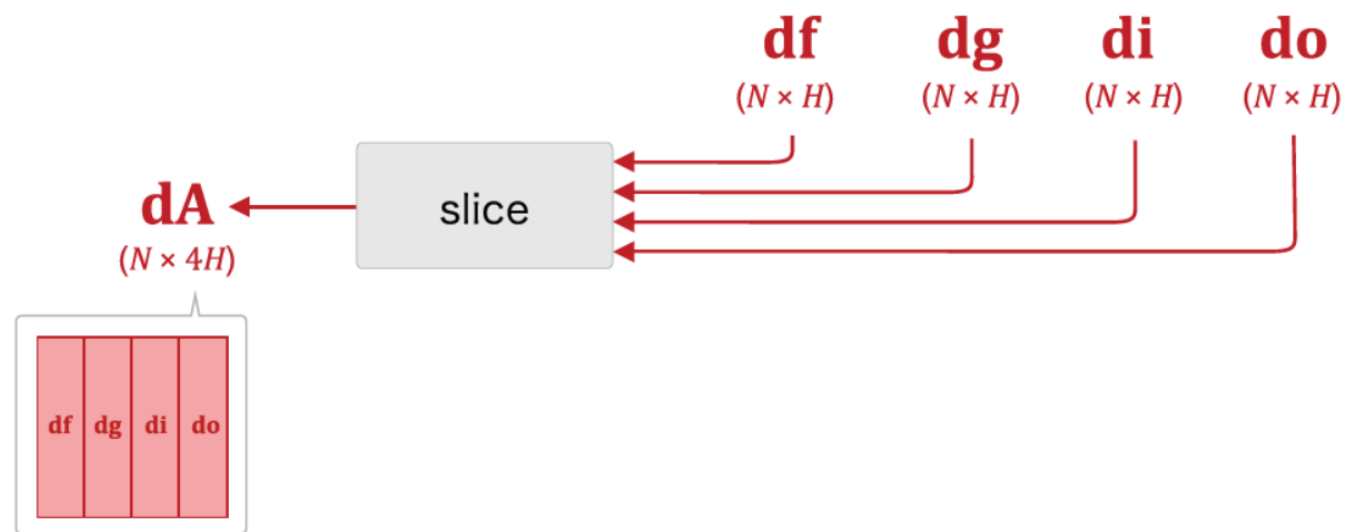
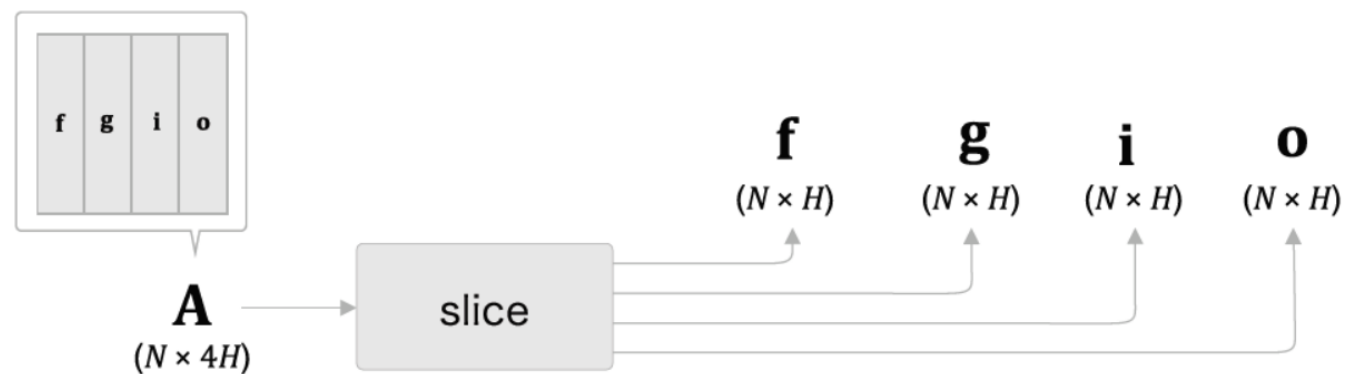


$$x_t \begin{bmatrix} W_x^{(f)} & W_x^{(g)} & W_x^{(i)} & W_x^{(o)} \end{bmatrix} + h_{t-1} \begin{bmatrix} W_h^{(f)} & W_h^{(g)} & W_h^{(i)} & W_h^{(o)} \end{bmatrix} + \begin{bmatrix} b^{(f)} & b^{(g)} & b^{(i)} & b^{(o)} \end{bmatrix}$$



- W_x, W_h, b 각각에 4개 분의 가중치(혹은 편향)가 포함되어 있다고 가정하고 Affine 변환을 수행하는 LSTM 계산 그래프

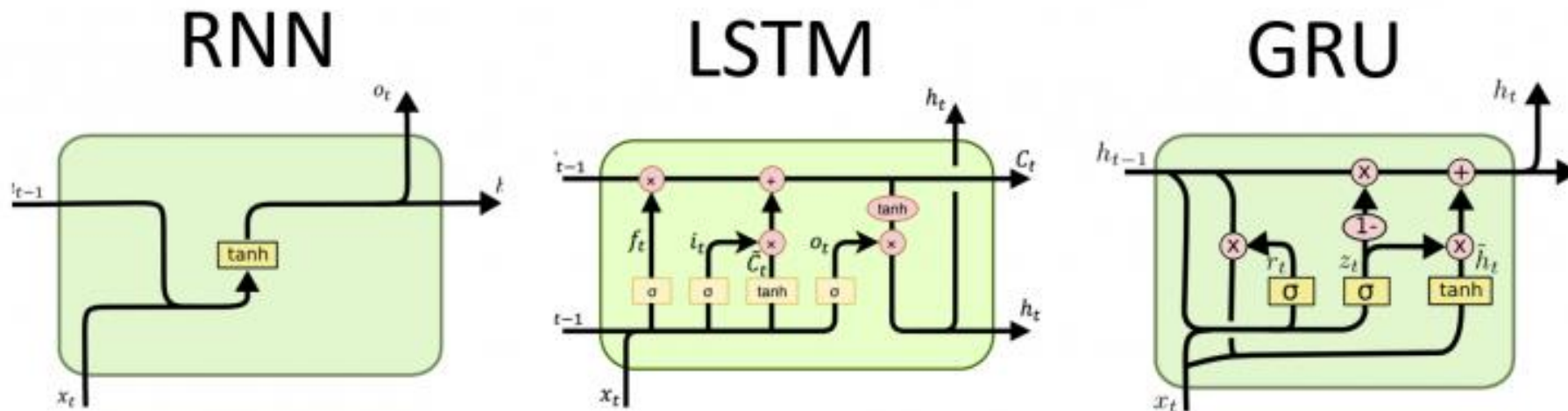




• 동작 원리

- Forget Gate를 통해 이전 특징 별로 기억 여부를 결정
- Input Gate를 통해 현 시점의 정보가 얼마나 중요한지를 반영하여 기록
- 이전 Cell State의 결과와 현 시점의 Forget Gate에서 잊고, Input Gate의 중요도만큼 곱한 값으로 현 시점의 Memory Cell값을 생성
- 현 시점의 Forget Gate, Input Gate에 의해서 변경된 현 시점의 Cell State 값을 얼마만큼 다음 레이어로 전달할지 계산

- LSTM에서 불필요한 부분을 제거하는 과정에서 생겨난 아키텍처
 - LSTM을 구성하는 Time-Step의 Cell을 조금 더 간소화한 버전
 - LSTM과 달리 Reset Gate(r)와 Update Gate(z)의 2개의 Gate만 가지고 있음
- 세 모델의 비교



THANK
YOU

