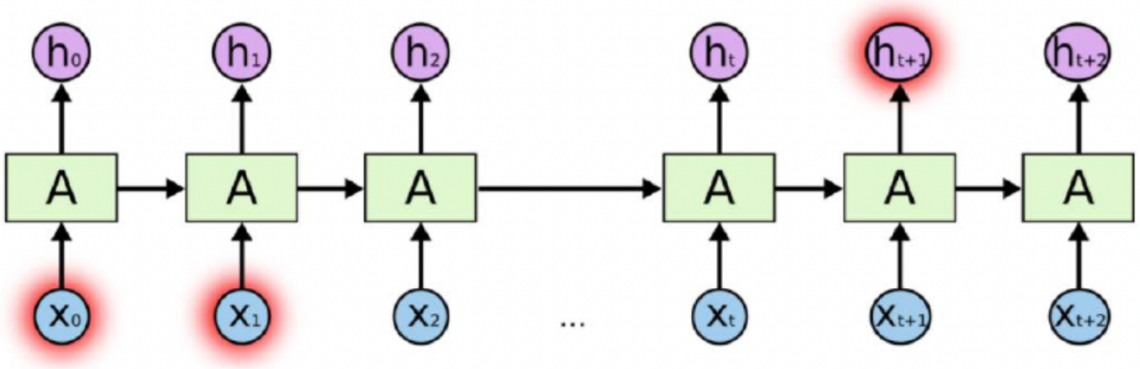
RNN

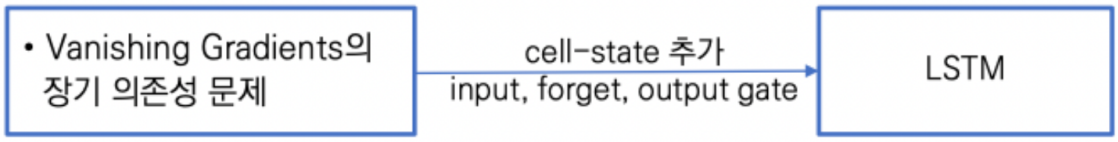
:순차적인 데이터를 처리하는 데 강점을 가지고 있지만, 한계점이 존재



하나의 네트워크가 여러개의 복사된 형태를 띄고 있고, 각각의 네트워크는 다음 단계로 정보를 넘겨줌. 시간에 따라, 학습이 진행됨에 따라 앞에서 input으로 받은 정보가 학습에 미치는 여향이 점점 감소하다가 결국 사라져버림.

이렇게 아래와 같은 문제가 발생함

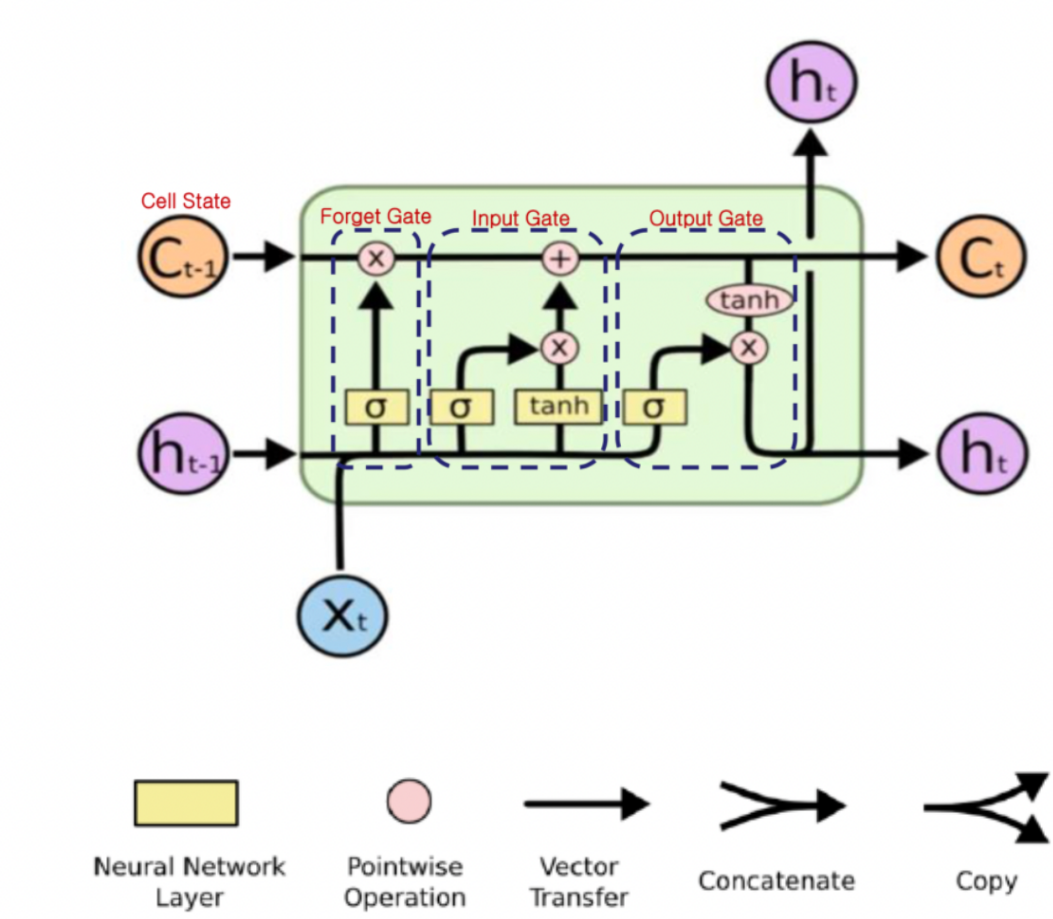
1. 장기 의존성 처리가 어려움
2. 그라디언트의 소실 및 폭발 문제
3. 계산 효율성 – 학습속도 저하



이를 극복하기 위해 나온 모델 중 하나

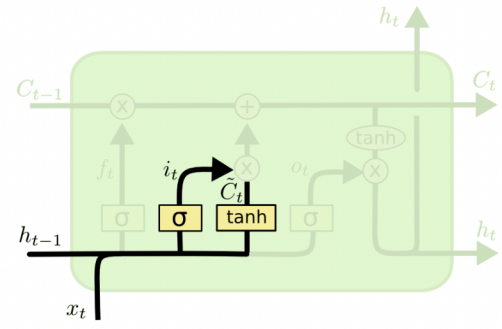
-> LSTM (Long Short-Term Memory)

LSTM의 핵심 구성 요소



셀 스테이트 : 장기 기억을 저장하고 유지하는 역할

⇨ LSTM이 장기 의존성을 다루는 핵심

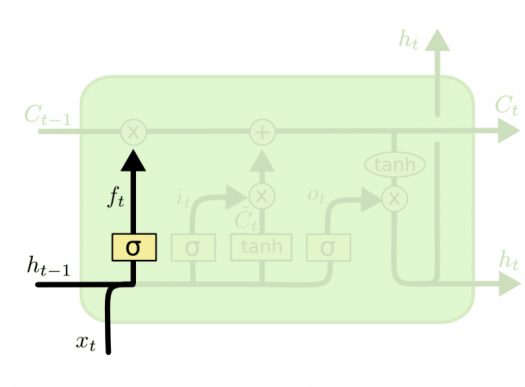


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

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

Input 게이트 (입력 게이트) : 현재 정보를 셀 스테이트에 추가하는 역할

어떤 정보를 받아들일지를 결정하는데, 이를 시그모이드 함수를 통해 수행

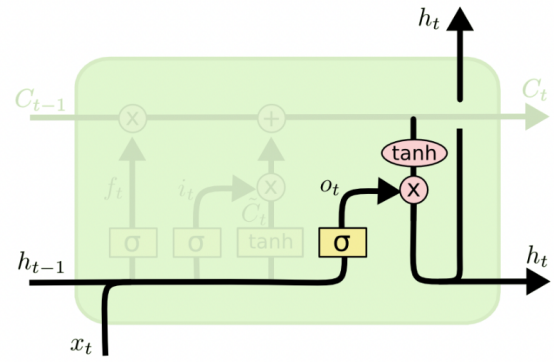


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Forget 게이트 (삭제 게이트) : 얼마나 많은 이전 정보를 잊을지 결정

시그모이드 함수를 통해 이루어지며, 값이 0에 가까울수록 많은 정보를 잊게됨

sigmoid 를 통과하고 나온 각 원소 값이 0 에서 1 사이의 값을 가지는 비율이므로, cell state 의 원소에서 각 원소에 대응되는 비율만큼 남기고 나머지는 버리겠다는 의미에서 'Forget gate'로 볼 수 있음



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

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

Output 게이트 (출력 게이트) : 현재 셀 스테이트에 기반하여 모델의 출력을 결정

tanh 함수에 통과시켜 -1과 1사이의 값으로 변환하고,

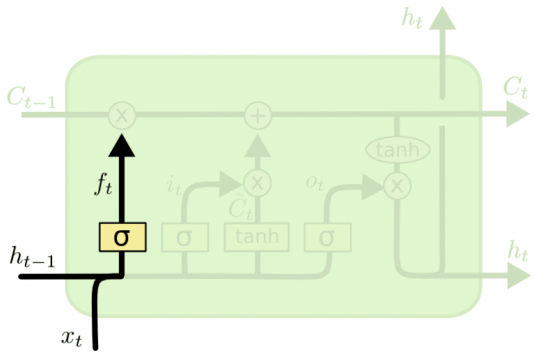
sigmoid 게이트와 곱하여 원하는 부분만 출력

Output gate의 과정은 이제까지의 좀 더 완전하고 많은 정보를 갖고 있는 곳에서 일부 정보만을 필터링하여 hidden state가 현재 time step에 직접적으로 필요한 정보만을 갖도록 하는 것으로 이해할 수 있음

LSTM 동작방식

1) 셀 스테이트에서 어떤 정보를 버릴지 선택

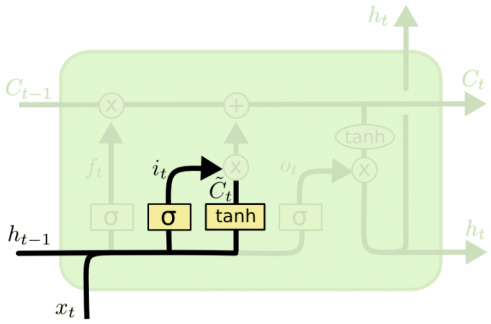
Forget gate layer라고 불리는 시그모이드 레이어로 만들어짐



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

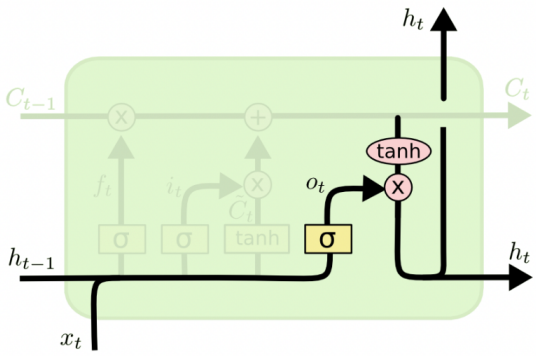
2) 새로운 정보가 셀 스테이트에 저장될지를 결정하는 단계

Input gate layer라고 불리는 시그모이드 레이어는 어떤 값을 우리가 업데이트 할지를 결정하는 역할을 함



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

3) 어떤 출력값을 출력할지 결정



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

https://pytorch.org/docs/stable/generated/torch.nn.LSTM.html

GRU

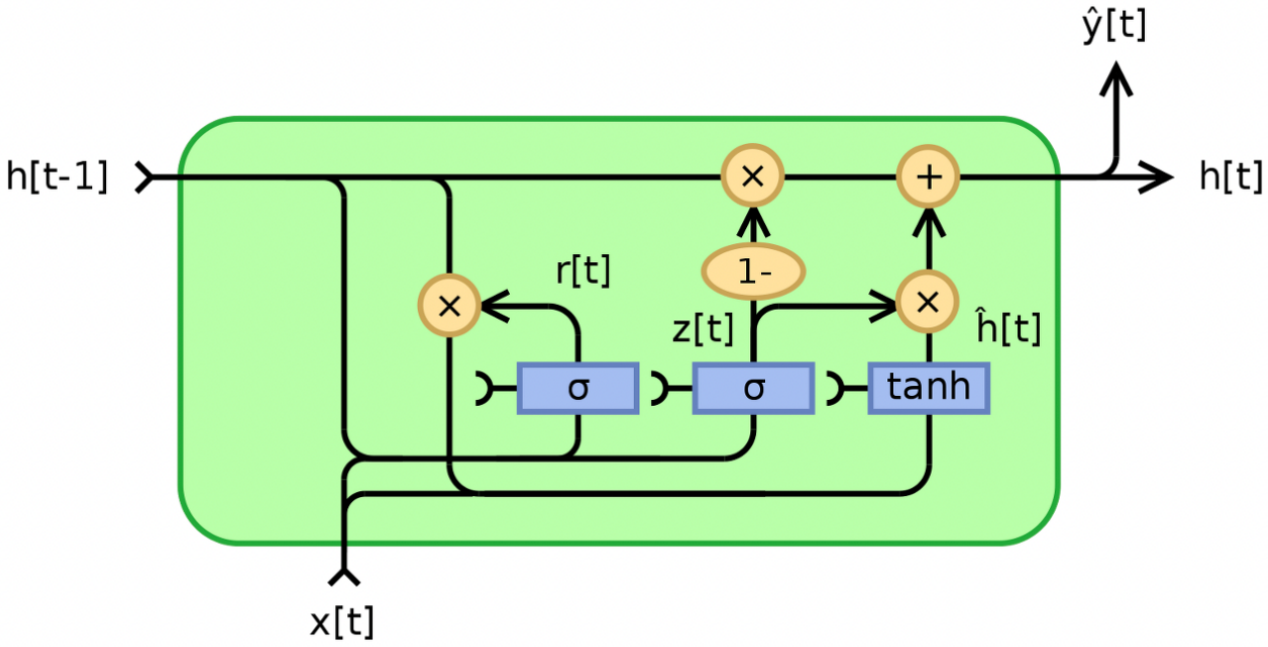
: LSTM 구조를 보다 경량화해서 적은 메모리로도 빠른 계산이 가능하도록 만든 모델

전체적인 동작은 LSTM과 비슷!

GRU에는 LSTM에서 입력데이터로 사용되던 cell state와 hidden state 두개가 아닌 이를 일원화한 hidden state만 사용한다는 것이 특징

즉, GRU에서의 hidden state는 LSTM에서의 cell state와 유사한 역할을 한다고 볼 수 있음

GRU의 구조



LSTM에서는 input gate와 forget gate의 독립적인 두 개의 gate 결과를 가지고 cell state를 업데이트했다면, GRU에서는 하나의 gate에서 hidden state를 연산하는 것을 볼 수 있음!

이로 인해 구조적으로 GRU는 LSTM에 비해 경량화된 모델로 볼 수 있음

정보를 담는 주된 벡터인 LSTM의 셀스테이트 또는 GRU에서의 hidden state를 업데이트 되는 과정이 기존 RNN처럼 동일한 것을 계속 곱하는 연산이 아니라 매 time step마다 값이 다른 forget gate를 곱하고, 필요로 하는 정보를 곱셈 뿐만이 아니라 덧셈을 통해서 만들어 낼 수 있다는 특징으로 인해 그라디언트 소실 또는 폭발 문제가 많이 사라지는 것으로 알려져있음