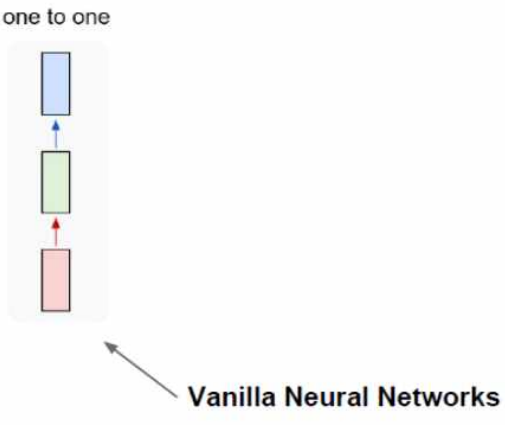
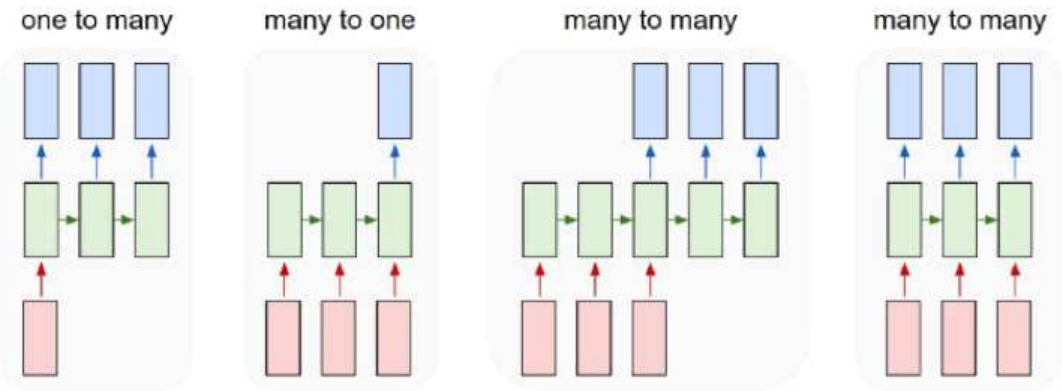
cs231n lecture 10 신지연

1. RNN이란?

- Vanilla Neural Network : 지금까지 배운 구조 / 입력이 하나 들어가고, 출력이 하나 나오는 형태, hidden layer를 거쳐 하나의 output 형태로 나옴



- RNN : 다양한 입출력을 다룰 수 있도록 만든 모델



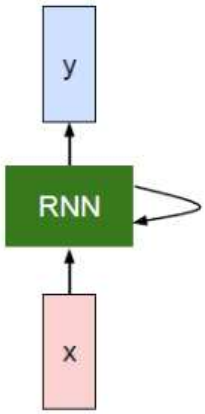
- Ex) 'one to many' : Image Captioning
  'many to one' : Sentiment Classification
  'many to many' : Machine Translation이나
  Video classification on frame level

We can process a sequence of vectors **x** by applying a **recurrence formula** at every time step:

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

new state / old state / input vector at some time step

some function with parameters W



- 매 step마다 x_t가 들어오고 기존에 있던 h_t−1를 가지고, h_t를 update

- set의 parameter들을 가지고 매 step마다 update를 진행

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

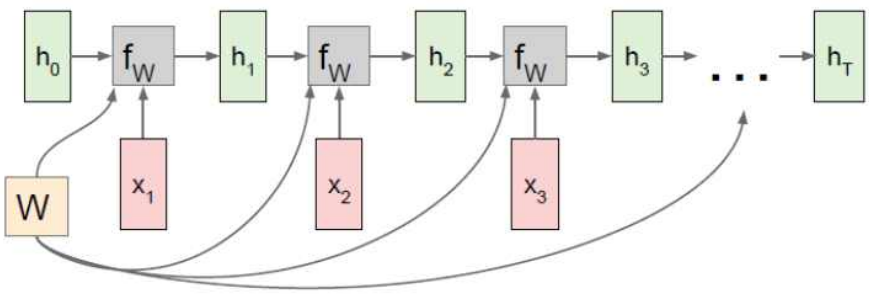
$$\downarrow$$

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

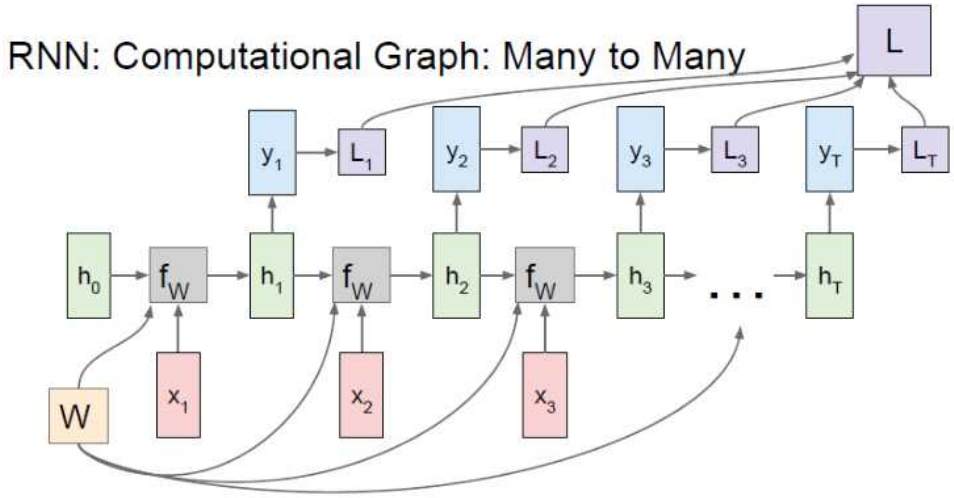
$$y_t = W_{hy}h_t$$

- 똑같은 set의 parameter를 가지고 update를 진행

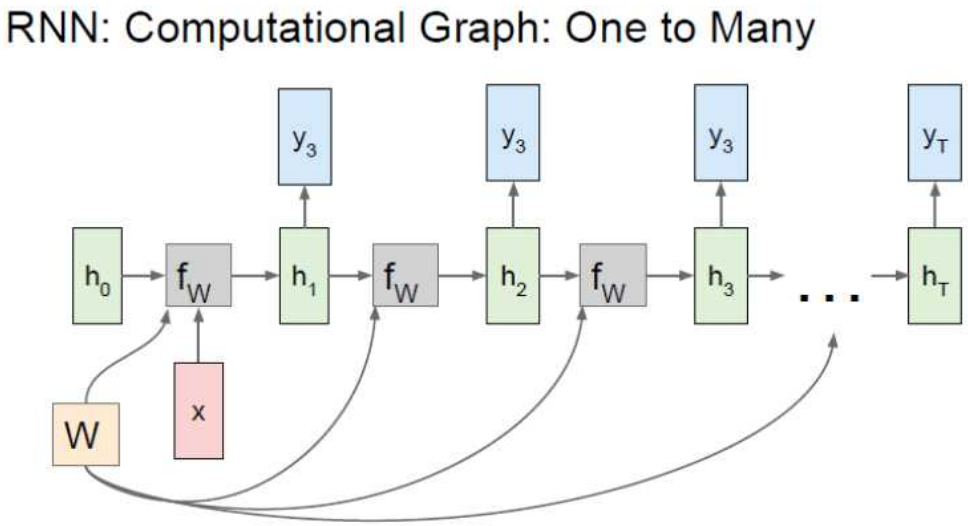
Re-use the same weight matrix at every time-step



- 출력이 다수라면 ‘many to many’ 분류, Loss function도 많음

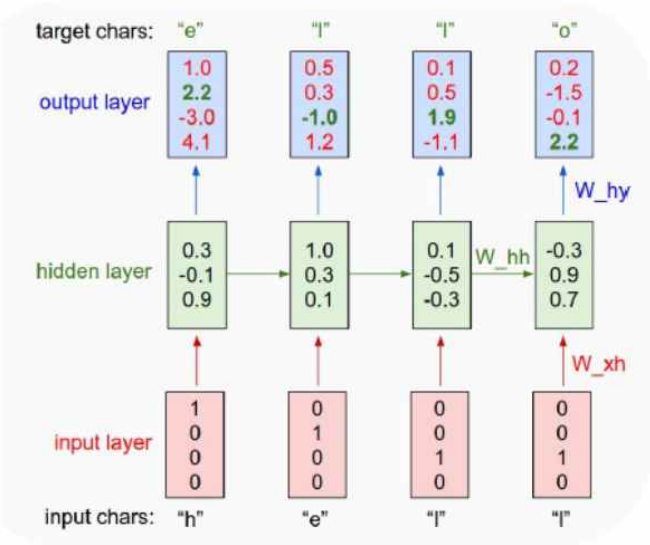


- ‘one to many'



2. Character-level language Model

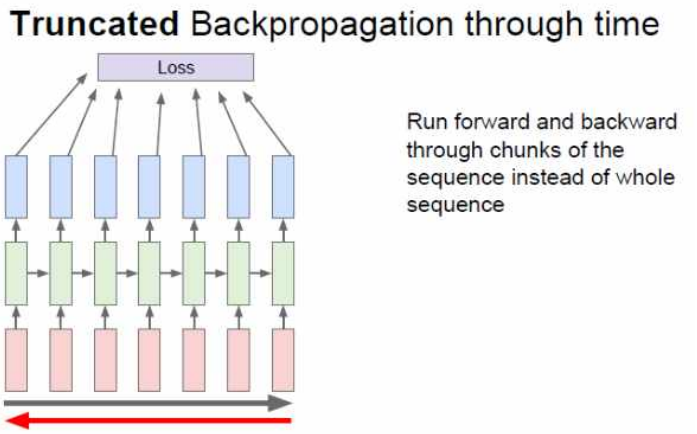
- [h,e,l,o]라는 문자열 시퀀스가 있고, “hello”라는 문자열을 만들려고 할 때



- Input 문자를 4d 벡터로 만든 후 h_t=tanh(W_hh*h_t-1 +W_xh * x_t) 식을 이용하여 hidden layer의 값을 구해줌
  (W_hh, W_xh 모두 같은 값 사용)

- target chars를 고를 때는 확률 분포에서 샘플링을 진행한 후,
  다음 Input chars로 넣어줄 문자를 골라줌
  (Softmax에서 가장 높은 값이 나온 문자가 우리가 원하는 문자가 아닐 수 있기 때문)

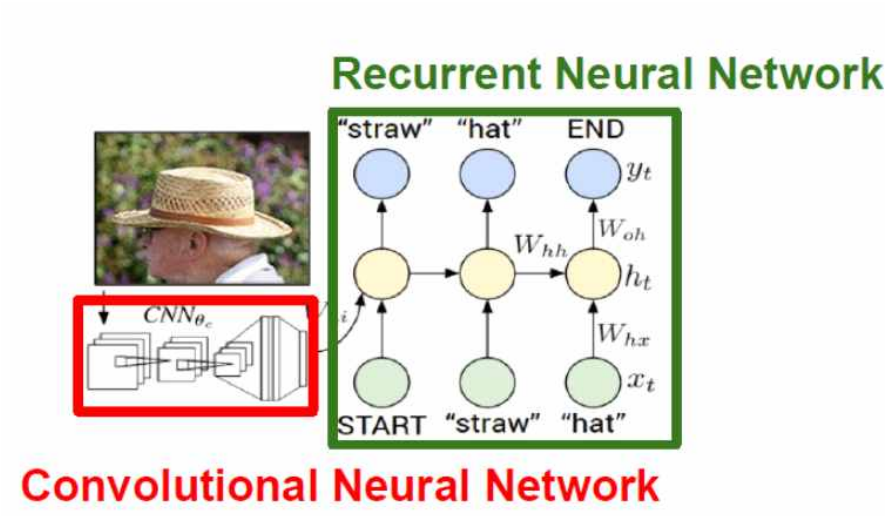
- train 할 때 한 스텝을 일정 단위로 나눈 후, 서브시퀀스의 Loss들만 계산



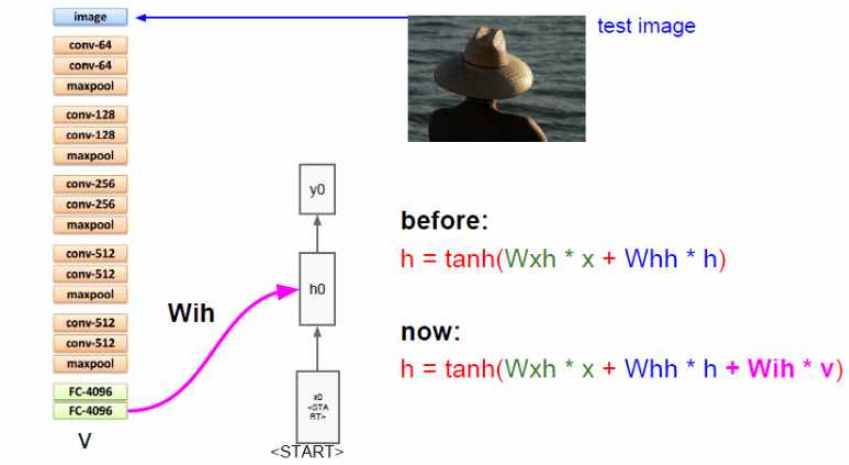
- 여러 분야에서 활용 : 다음 문자가 어떤 것이 나올지 예측하는 모델,
전체적인 문장 구조를 파악해서 알아서 구조를 학습

3. Image Captioning

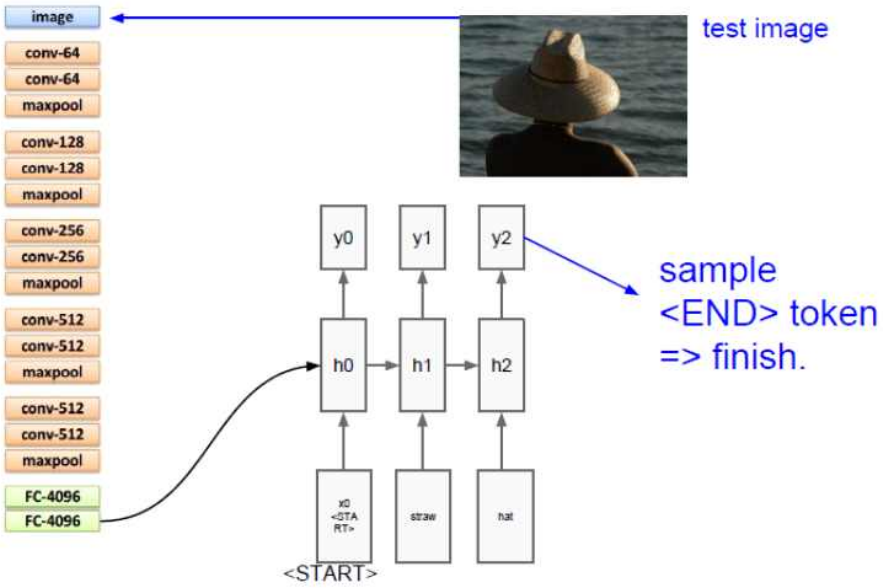
- Image Captioning: 하나의 사진을 보고 이 사진이 무엇인지 설명하는
문장을 만들어 내는 딥러닝 모델



- 사진을 보고 CNN 구조를 통과한 결과가 RNN의 입력으로 들어가는 모습

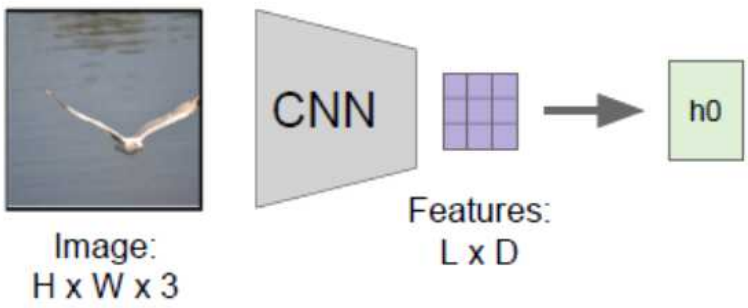


- 실제로 test Image가 들어가게 되면 Softmax를 통과하기 바로 전 Fully Connected Layer를 이용, 새로운 가중치 행렬을 추가하여 이미지 정보 추가



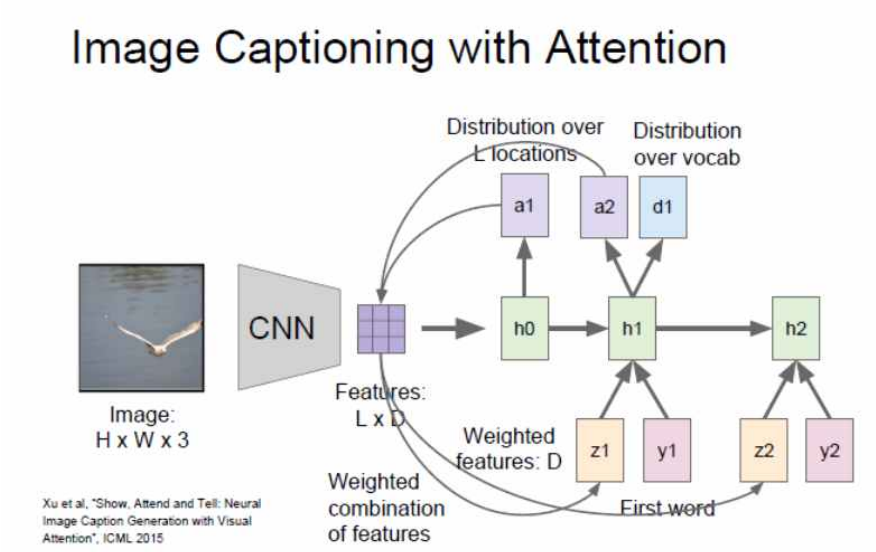
- (Supervised Learning)

- Image Captioning with Attention (조금 더 발전한 모델)

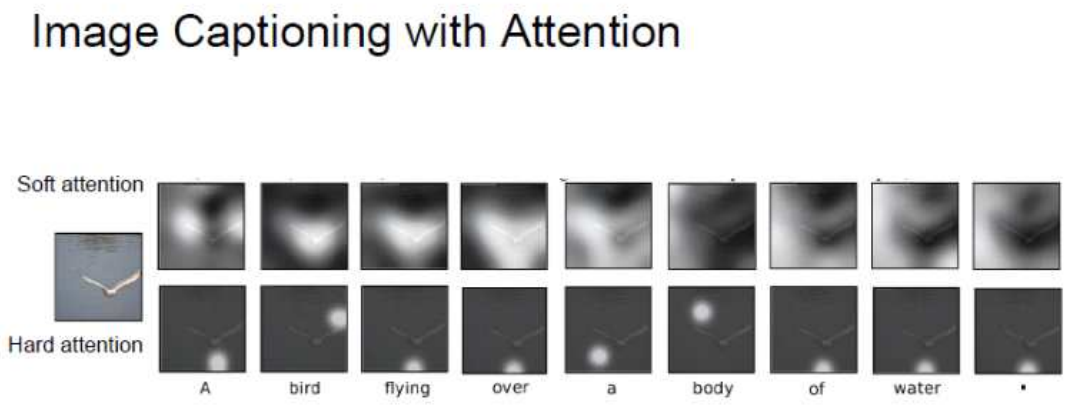


- a1, a2 등이 이미지의 분포를 나타낸 출력값,
  d1, d2 등이 단어의 분포를 나타낸 출력값

$$z = \sum_{i=1}^{L} p_i v_i$$



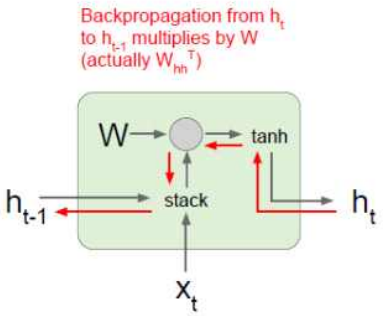
- 의미가 있다고 생각하는 부분에 스스로 attention을 집중하는 모습



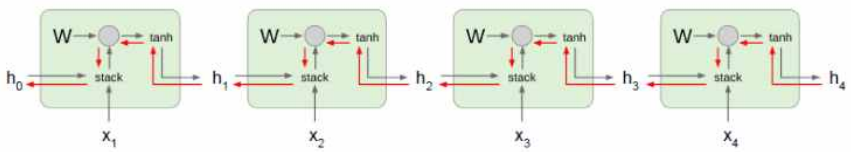
- VQA(Visual Question Answering)

- Vanilla RNN Gradient Flow : 하나의 cell이 아닌 전체적인 cell로 gradient를 구할 때 많은 W 행렬들이 개입하게 되어 계산량이 늘어나고, Weight 행렬에서 특이값의 최대값에 따라 Gradient값이 폭발적으로 증가할 수도, 너무 작게 감소할 수도 있는 문제점이 발생

-> LSTM이 해결



$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$
$$= \tanh\left((W_{hh} \quad W_{hx})\begin{pmatrix}h_{t-1}\\x_t\end{pmatrix}\right)$$
$$= \tanh\left(W\begin{pmatrix}h_{t-1}\\x_t\end{pmatrix}\right)$$



Computing gradient of h_0 involves many factors of W (and repeated tanh)

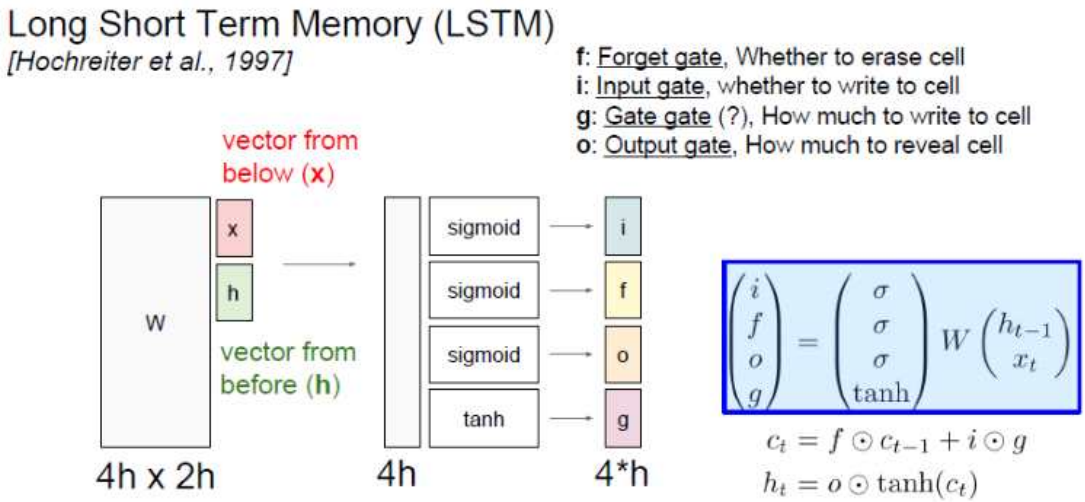
Largest singular value > 1: **Exploding gradients**

Largest singular value < 1: **Vanishing gradients**

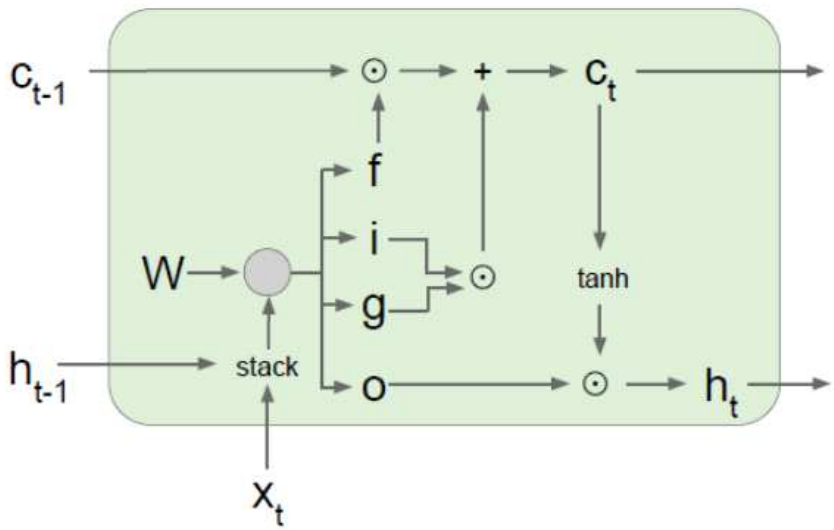
3. LSTM

- LSTM은 두개의 hidden state가 존재하고
  내부에만 존재하는 변수인 c_t 존재

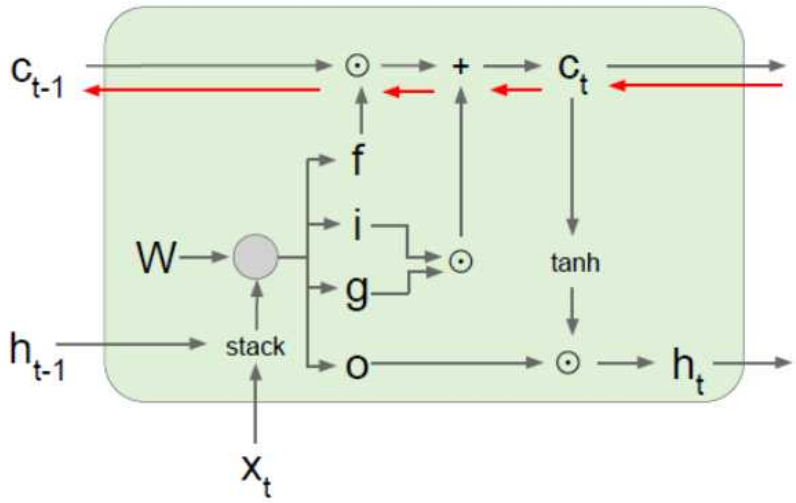
- 4가지의 gate :
  i - Input gate : cell에 대한 입력 (x_t에 대한 가중치)
  f - Forget gate : 이전 cell에 대한 정보를 얼마나 지울 것인가?
  o - Output gate : c_t를 얼마나 밖으로 드러낼 것인가?
  g - Gate gate(?) : Cell을 얼마나 포함시킬 것인가?



- LSTM에 대한 Flow chart



- Gradient Flow



- 하나의 cell에서 Weight행렬을 거치지 않고도 cell 단위에서 gradient 계산

- 여러 cell을 거치더라도 계산량이 크게 증가하지 않는다는 것을 알 수 있다.

- forget gate와 곱해지는 연산이 행렬 단위의 곱셈 연산이 아니라 element-wise (원소별 연산)이기 때문에 계산량이 크게 증가x