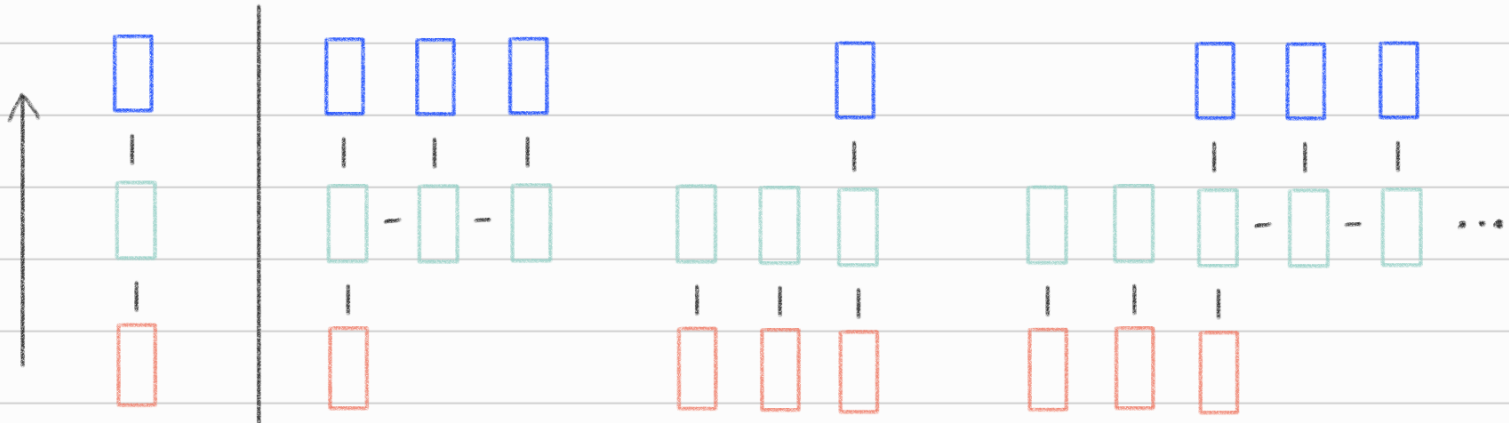
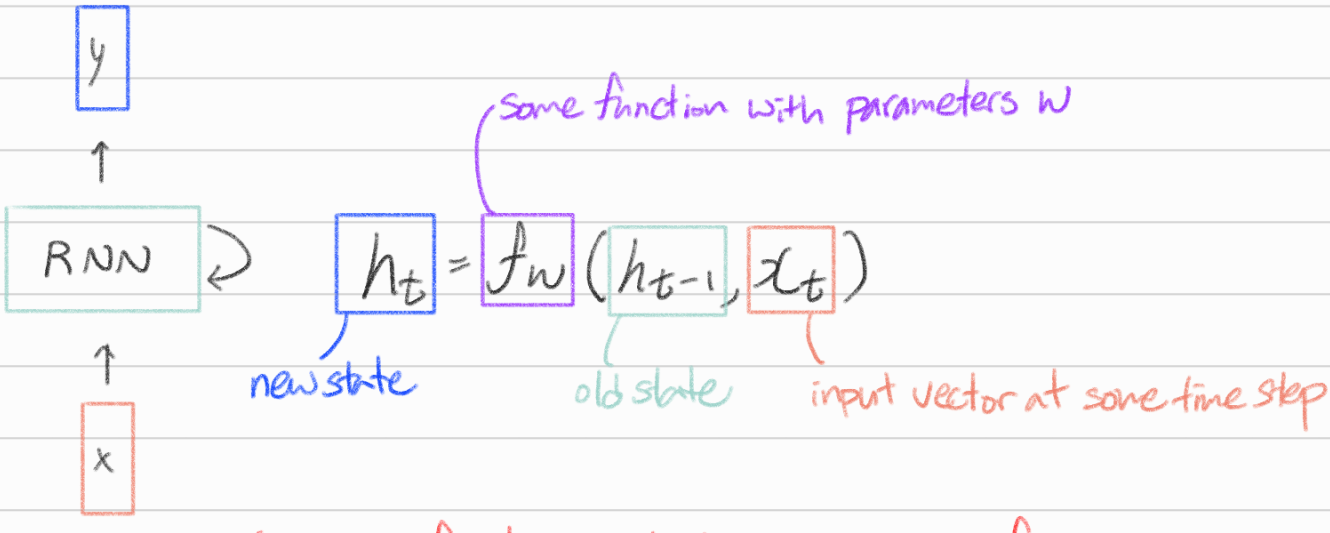
Recurrent Neural Networks

we want to have more flexibility in the types of data that our models can process ↴



RNN: recurrence formula @ every time step



※ Same function and the same set of parameters are used at every time step

Vanilla RNN:

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

현재 입력값 (x_t)에 W_{xh} 곱한 것

+

과거 값 (x_{t-1})에 W_{hh} 곱한것 + 편향값 (b)

BPTT (Backpropagation through time)

: forward through entire sequance to compute loss,
then backward through entire seq to compute gradient

Truncated " : 오래 걸리니 쪼개서 근사치 사용)

"현재와 과거를 함께 고려해 미래를 예측"

Image Captioning

CNN ⟶ RNN

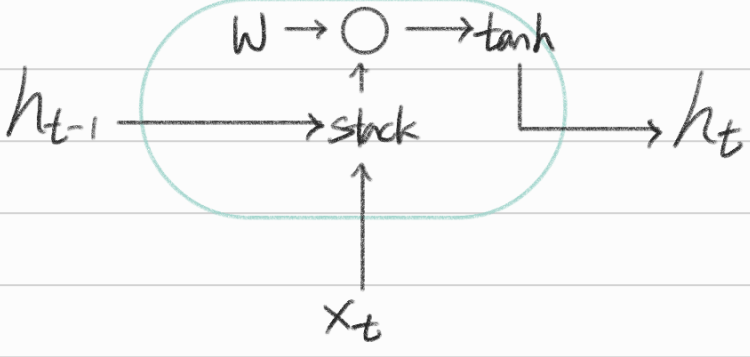
Summary vector of the image (v)

<start> <end>

$h = \tanh(W_{xh} * x + W_{hh} * h + W_{ih} * v)$

Visual Question Answering : RNNs with Attention

Vanilla RNN Gradient Flow



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

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

$= \tanh\left(W\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$

$h_0 \quad h_1 \quad h_2 \quad h_3 \quad h_4 \quad h_5$

← Backprop

모든 layers에서 동일한 값

Computing gradient of h_0 involves many factors of W, Vanishing G
같은 값을 반복해서 곱하면 1이 아닌 이상 gradient가 0으로 수렴하거나 explode
exploding G

explode하면 'Gradient clipping' (if its norm is too big)
vanish하면 change RNN architecture

→ LSTM (Long Short Term Memory) ≒ GRU

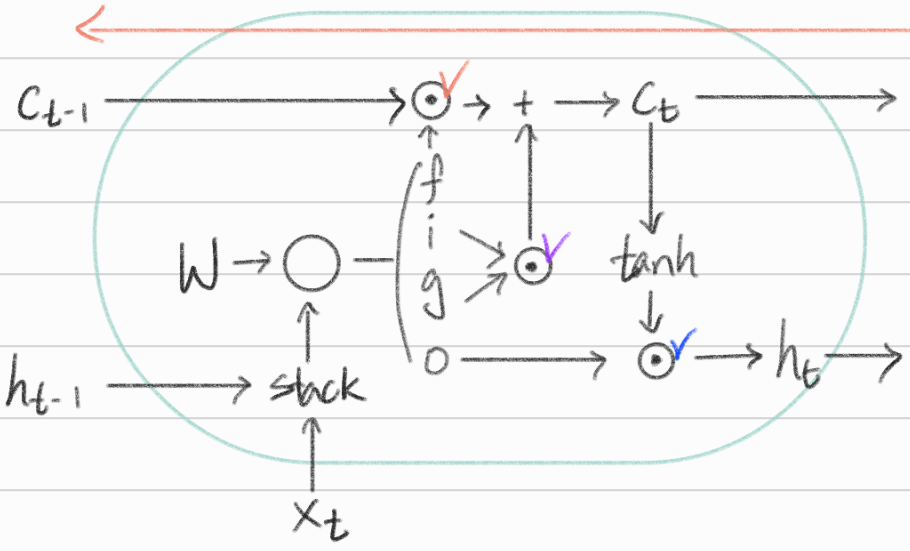
$$\begin{pmatrix} i: \text{input gate} \\ f: \text{forget gate} \\ o: \text{output gate} \\ g: \text{gate gate} \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

Sigmoid $0 \le \le 1$

$-1 \le \le 1$

Cell state: $C_t = f \odot C_{t-1} + i \odot g$
hidden state: $h_t = o \odot \tanh(C_t)$

$\odot$: multiply element-wise



BACKPROP:

no matrix multiplied by W

∴ Uninterrupted gradient flow
(Similar to ResNet)