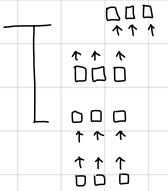
Recurrent Neural Networks

1. input-output

   ① 1-多 . image captioning

   ② 多-1 : sentiment classification

   ③ 多-多



: machine translation

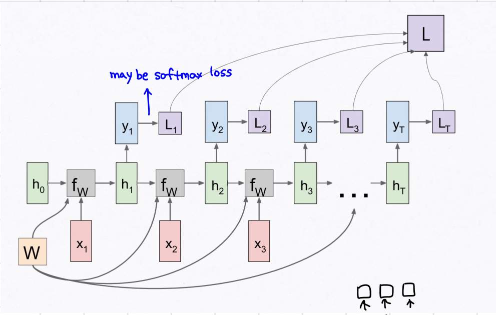
: Video classification on frame level

2. Sequential Processing of Non-Sequence data : 시계열성 없는 데이터에 시계열성 부여해 RNN 사용

   e.g. classify images. generate imgs one piece at a time

3. Vanilla RNN

   ① train time

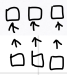


$h_{t+1} = f_W(h_t, x_{t+1})$

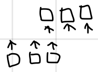
$= \tanh(W_{hh} h_t + W_{xh} x_{t+1})$ why tanh?

↳ usually set h_0 to 0

$y_{t+1} = W_{hy} h_{t+1}$

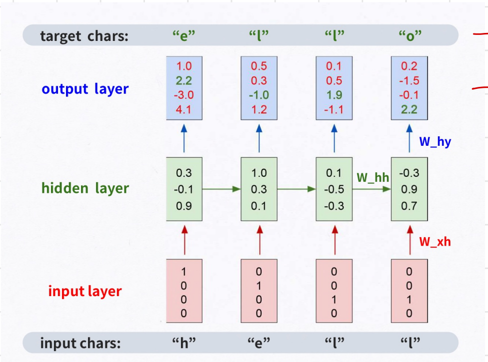


<many - to - many >



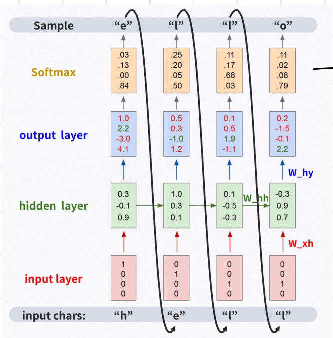
many-to-one, one-to-many도 같은 맥락 ⇒ many-to-many : many-to-one ⊕ one-to-many

encode — decode, produce output seq



→ softmax loss compute

② test time



→ 이 softmax에서 sampling

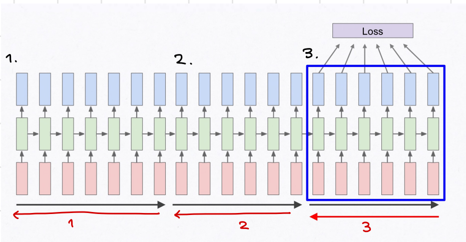
Why sample? get diversity in the model. 처음 들어온 input에 너무 의존하지 않도록.
In practice, argmax 쓰기도 하고 sample 쓰기도 함.

Why one-hot vector instead of softmax vector?

1. train time과 다른 architect → 성능 ↓ (train은 one-hot을 input으로 사용하므로)
2. vocab can be large → softmax는 dense, one-hot은 sparse → softmax 비쌈.

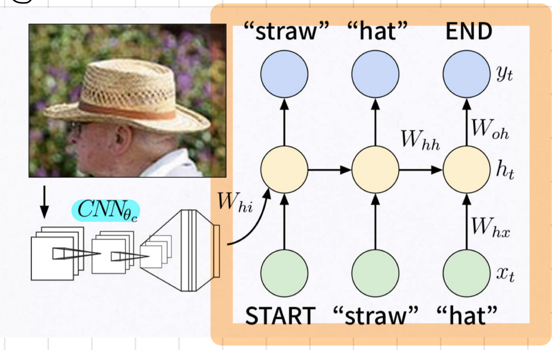
③ Backpropagation : through entire sequence to compute grad → 긴 seq에서 문제가 될 수 있음.

→ Truncated Backpropagation : through chunks of the sequence. not whole.



4. Image Captioning

① Architecture



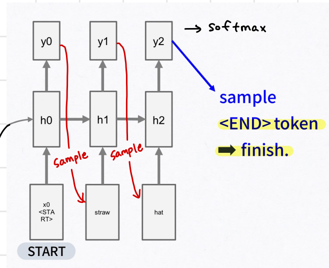
test : image

CNN

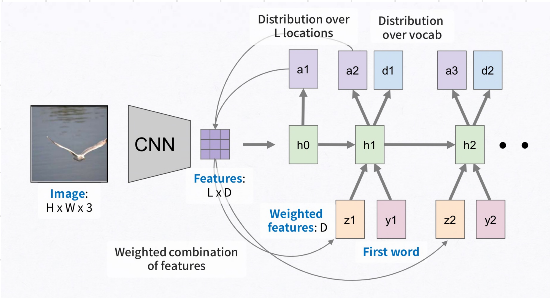
- FC 1000, softmax

4096-D vector

W_{ih}



② Attention



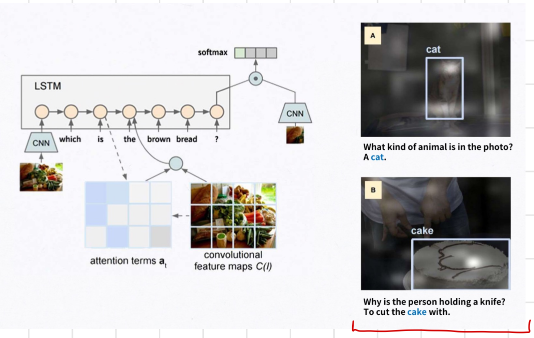
$z = \sum_{i=1}^{L} p_i v_i$

p from a_i

v from features

⇒ 어떤 location에 집중해서 볼건지.

③ VQA (← NLP & CV)



soft-spatial attention

Question → single question vector
CNN → summarize image

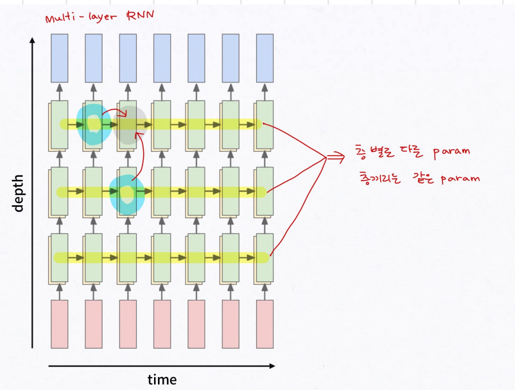
→ combine → predict distribution over answers

↓

how? 'concat' in common

Somtimes more fancier ways ..

5. Multi-layer RNN : 2-3 layer in common. no super-deep



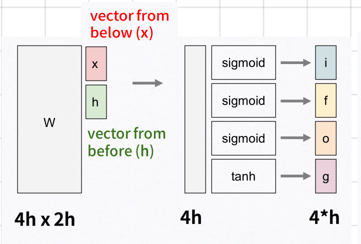
6. Problems of RNN

$\frac{\partial L}{\partial h_0}$ 구하는 과정에서 계속해서 W^T를 multiply →

① W의 최댓값 >1 : gradient explode → Gradient clipping ⇒ norm을 줄임

② W의 최솟값 <1 : gradient vanishing → new architecture

7. LSTM (Long Short term Memory)



$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$c_t = f \odot c_{t-1} + i \odot g$ → cell state (internal)

$h_t = o \odot \tanh(c_t)$ → hidden state

i : whether to input into cell
f : whether to erase cell
o : how much to reveal cell
} ∈ (0,1)

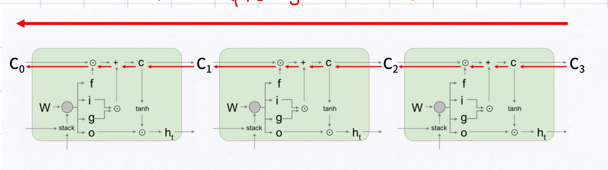
g : how much to write to cell → ∈ (-1,1)

→ c_t에서 c_{t-1}로 backpropagate 시 f만 element wise product하면 됨

장점 :
(1) element-wise multiplication is better than matrix multiplication
(2) multiply by different forget gate value at each time step
(3) forget gate is from σ → f ∈ (0,1) → grad explode ⊗
grad vanish : bias를 양수로 설정해서 1에 가깝도록 함. → 완화

→ BP

Uninterrupted gradient flow



8. Highway Network : highway BP처럼.

Highway Networks

$g = T(x, W_T)$ → candidate activation.

$y = g \odot H(x, W_H) + (1-g) \odot x$ → gating function : perev input ⊙ candidate activation ...

9. RNN Variants

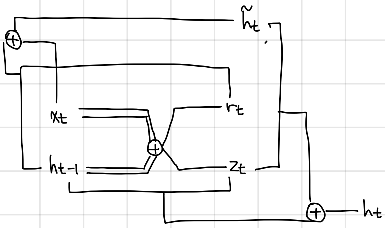
① GRU :

$r_t = \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r)$

$z_t = \sigma(W_{xz}x_t + W_{hz}h_{t-1} + b_z)$

$\tilde{h}_t = \tanh(W_{xh}x_t + W_{hh}(r_t \odot h_{t-1}) + b_h)$

$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t$



② LSTM, GRU architecture를 tweak해도 비슷한 performance 보임.