

VGG Net

571

- 기존 네트워크는 5×5 or 7×7 conv layer $\xrightarrow{2}$ $1 \times \frac{w}{2}$
 - 3×3 conv layer 반복 사용하여 5×5 or 7×7 layer 구조 $\xrightarrow{11}$ 등
 - +1 padding 을 적용해면 feature map 크기 유지 가능

7121-

- 더 깊은 네트워크 학습 가능 (더 적은 memory로 더 큰 capacity)

$$\begin{aligned} \{y\} &\rightarrow 5 \times 5 \text{ conv. layer} \\ &\quad \leftarrow h = h - 5 + 1 = h - 4 \\ \hookrightarrow & h_2 - 3 + 1 = h_1 - 2, \quad h = h_2 - 3 + 1 = h = h_2 - 2 = h_1 - 4 \\ &\quad \leftarrow 3 \times 3 = 5 \times 5 \end{aligned}$$

၁၂၅

- 1) 3x3 Conv. layer + 1 padding
 - 2) 흑백정향수, Batchnorm
 - 3) 1, 2 반복
 - 4) 2x2 MAXpooling or stride

Summary

- 가볍고 편리
 - FC layer 단계으로 지적

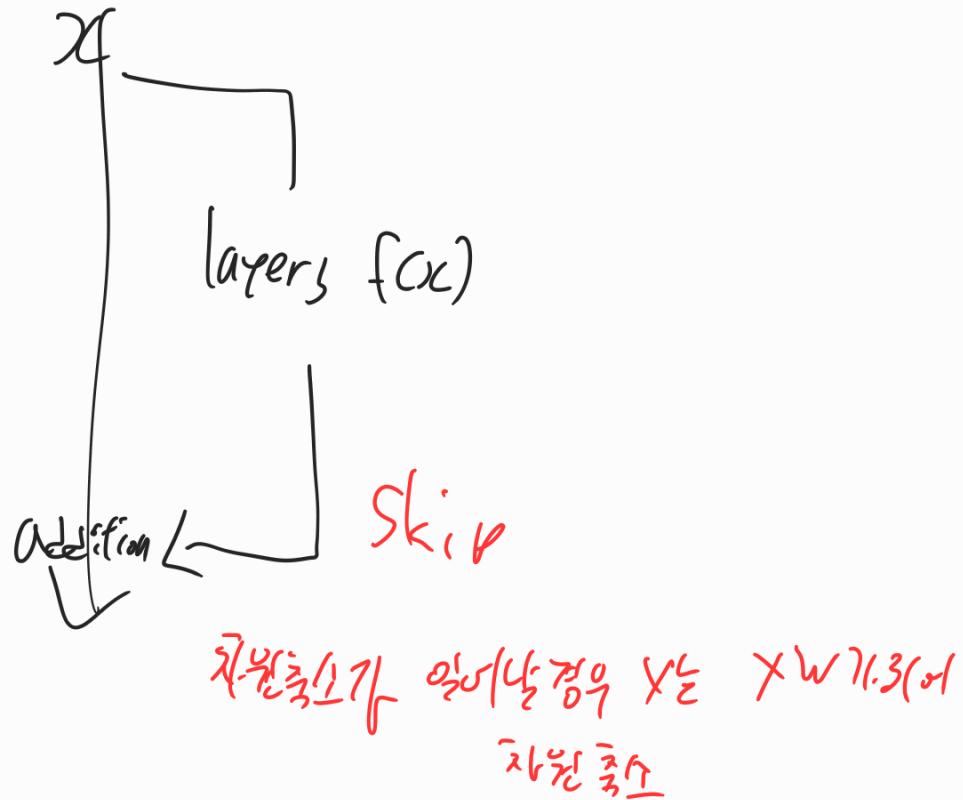
ResNet

등기

- 깊을수록 성능↑
- Gradient vanishing 깊은 네트워크 학습시키는데 미끄러워 많음

부분

- $F(x) = h(x) - x$
- $H(x) = F(x) + x$



ResNet은 gradient vanishing을 방지함

- 그래서 대부분의 큰 Network는 residual connection(?)

Transfer learning

- Conv layer은 feature을 추출하고자 학습됨
- So VggNet은 해당 task를 수행하기 위한 feature extraction 능력이 있을 것
- datatfr 다른 디렉토리 이미지를 활용한 task에 대해서 공통된 feature 존재

그러나 Transfer learning이란 특정 task를 학습한 weight를 다른 task에 활용하는 것

방법

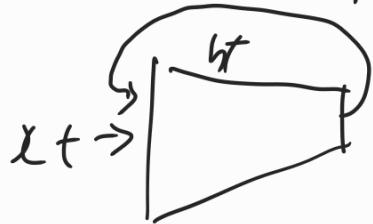
1. 기존은 weight를 seed로 활용 후 학습
2. 기존은 layer freeze 후 나머지 학습
3. 기존을 초기화하고 새로 학습한 layer의 learning rate를 조절해 학습

Recurrent Neural Network

Sequential Data vs Time Series

- Time stamp의 유익한 데이터를 차이
 - Sequential data는 data의 순서가 점보기 매우 중요함
 - Time series는 해당 data 발생시각 점보기 중요함

$$h_t = f(x_t, h_{t-1}; \theta)$$



$$\hat{y}_t - h_t = f(x_t, h_{t-1}; \theta)$$

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

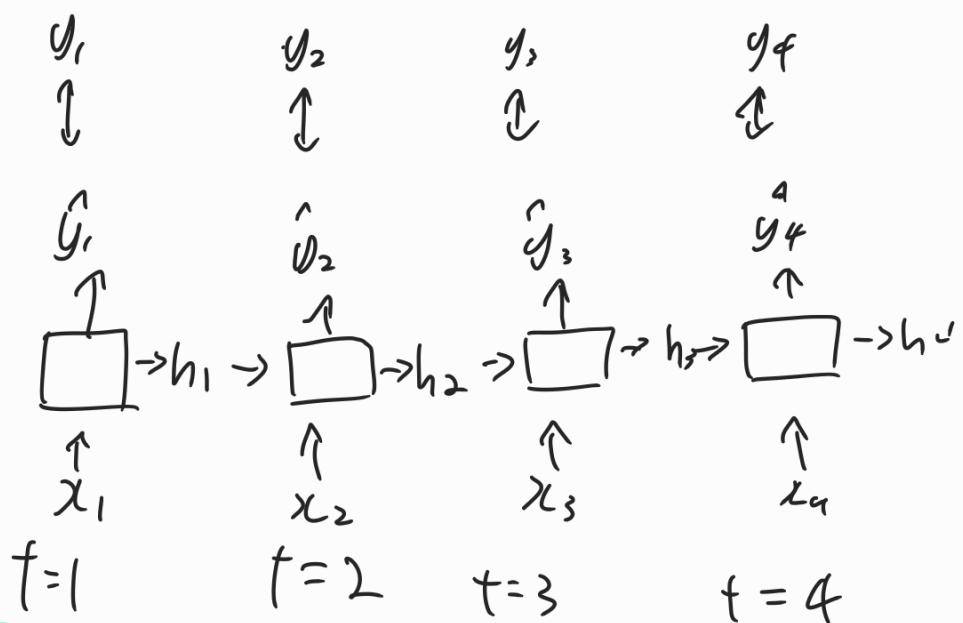
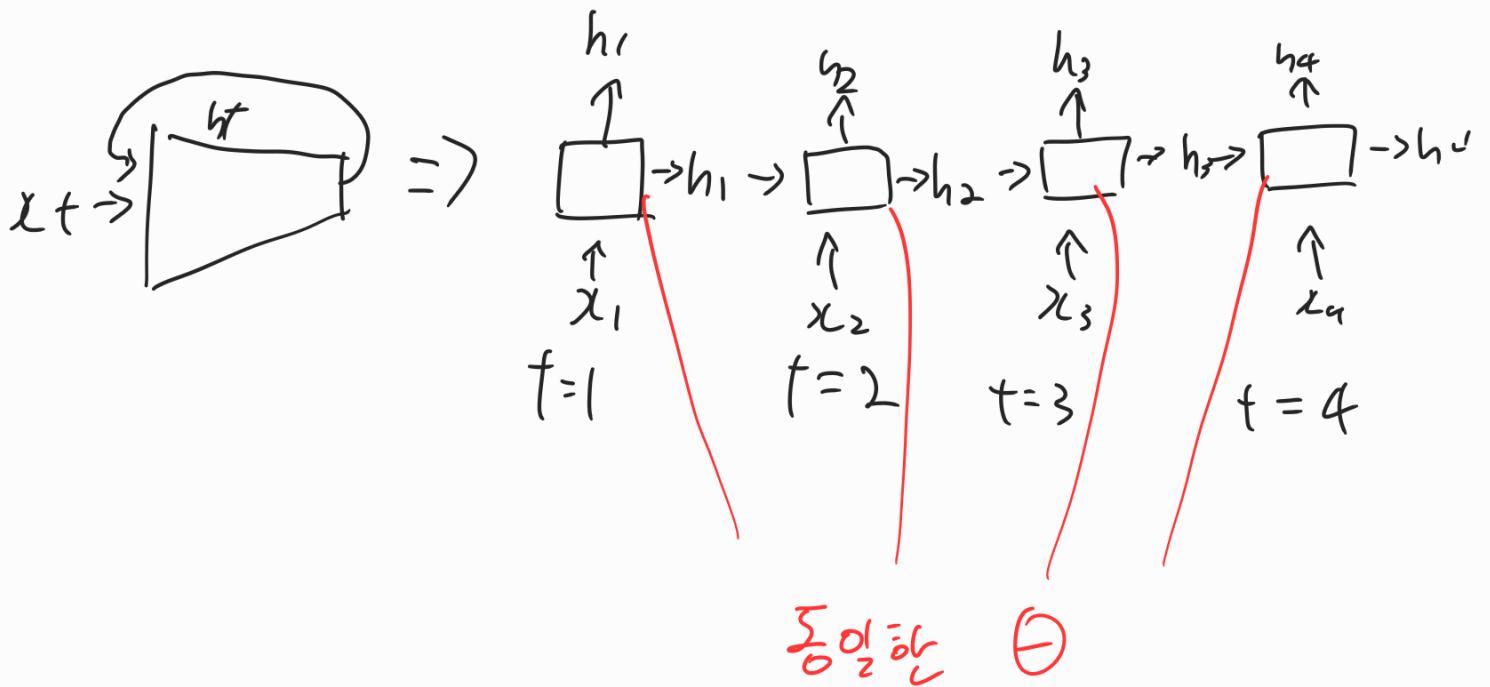
where $\theta = \{W_{ih}, b_{ih}, W_{hh}, b_{hh}\}$

$$= \tanh(w \cdot [x_t, h_{t-1}]) \quad \text{bias } b, \text{ concat}$$

x 는 n 차원

h 는 d 차원

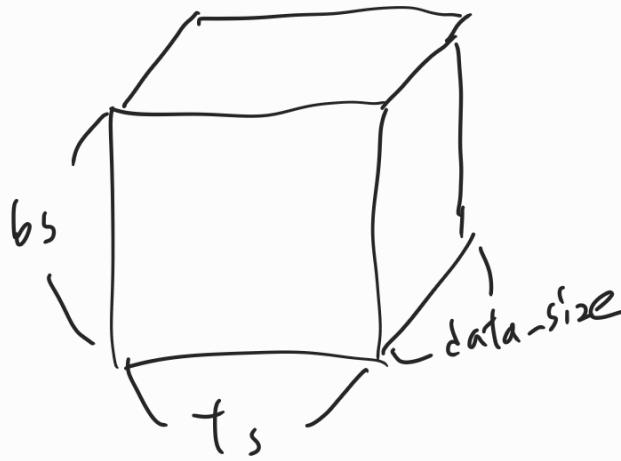
w 는 $n+d$ 차원



$$L(\theta) = \sum_{t=1}^T (|\hat{y}_t - y_t|)$$

Input tensor

(batch size, timestep, data)



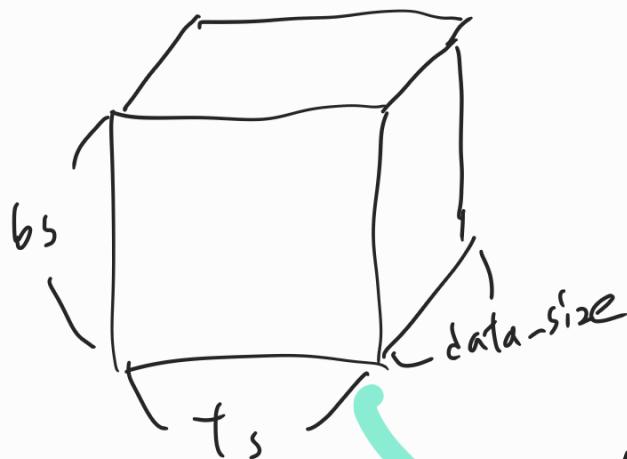
D



문장이라 가정하면

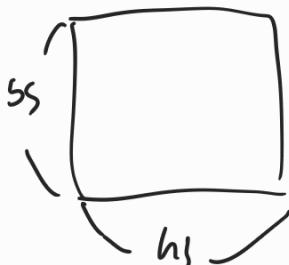
bs만큼의 문장 중 몇개를 처리

묶음

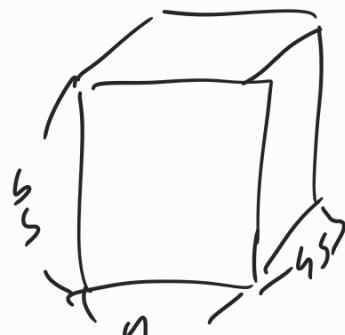


Minibatch size

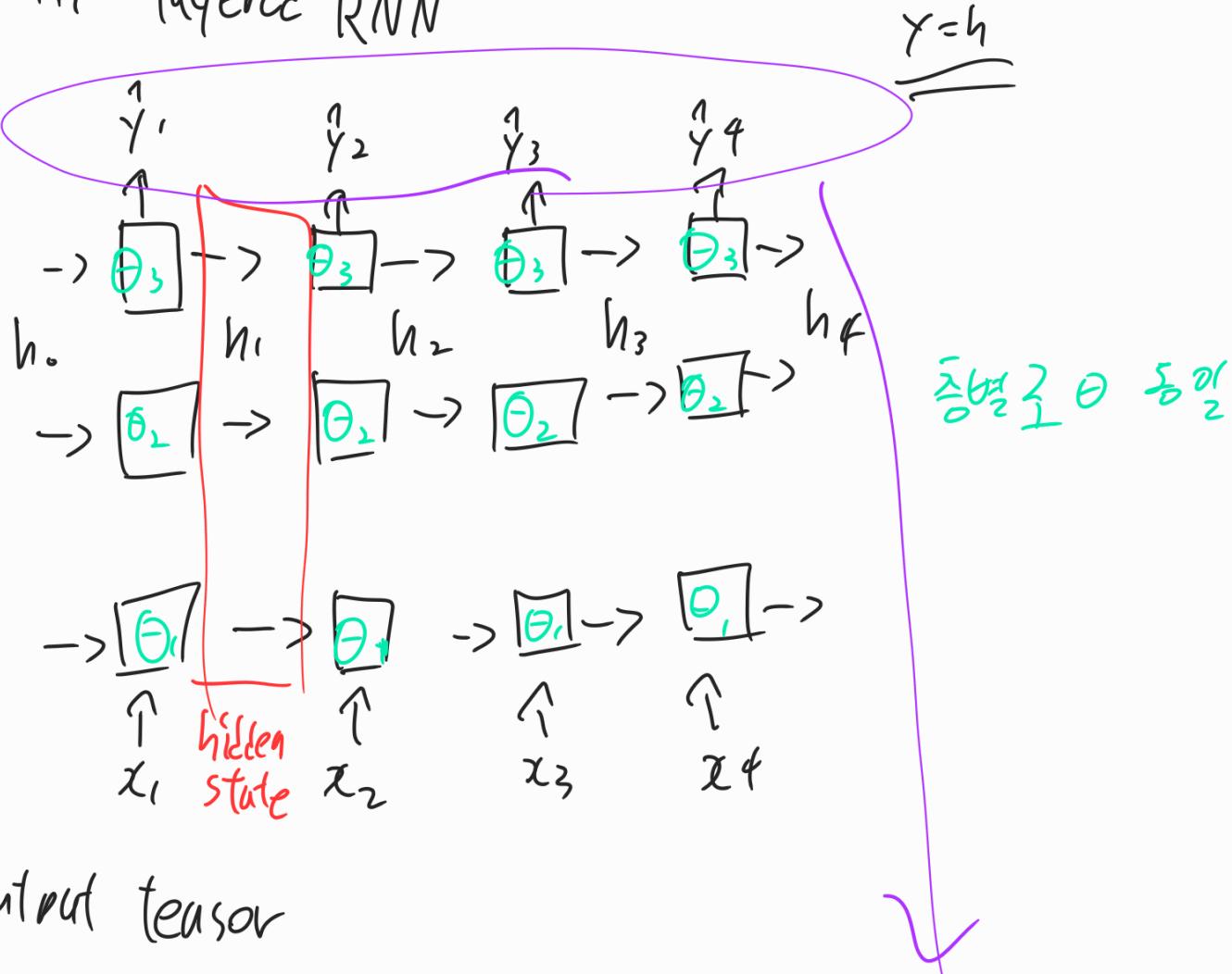
hidden tensor



(bs, 1, hs)



Multi-layered RNN

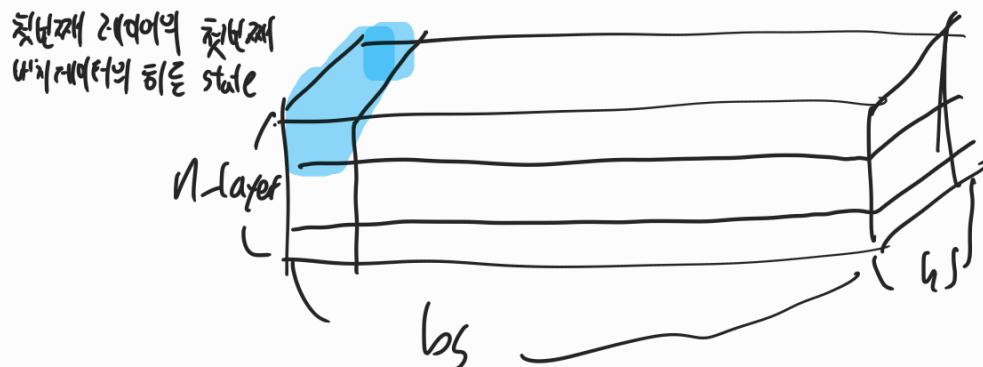


output tensor

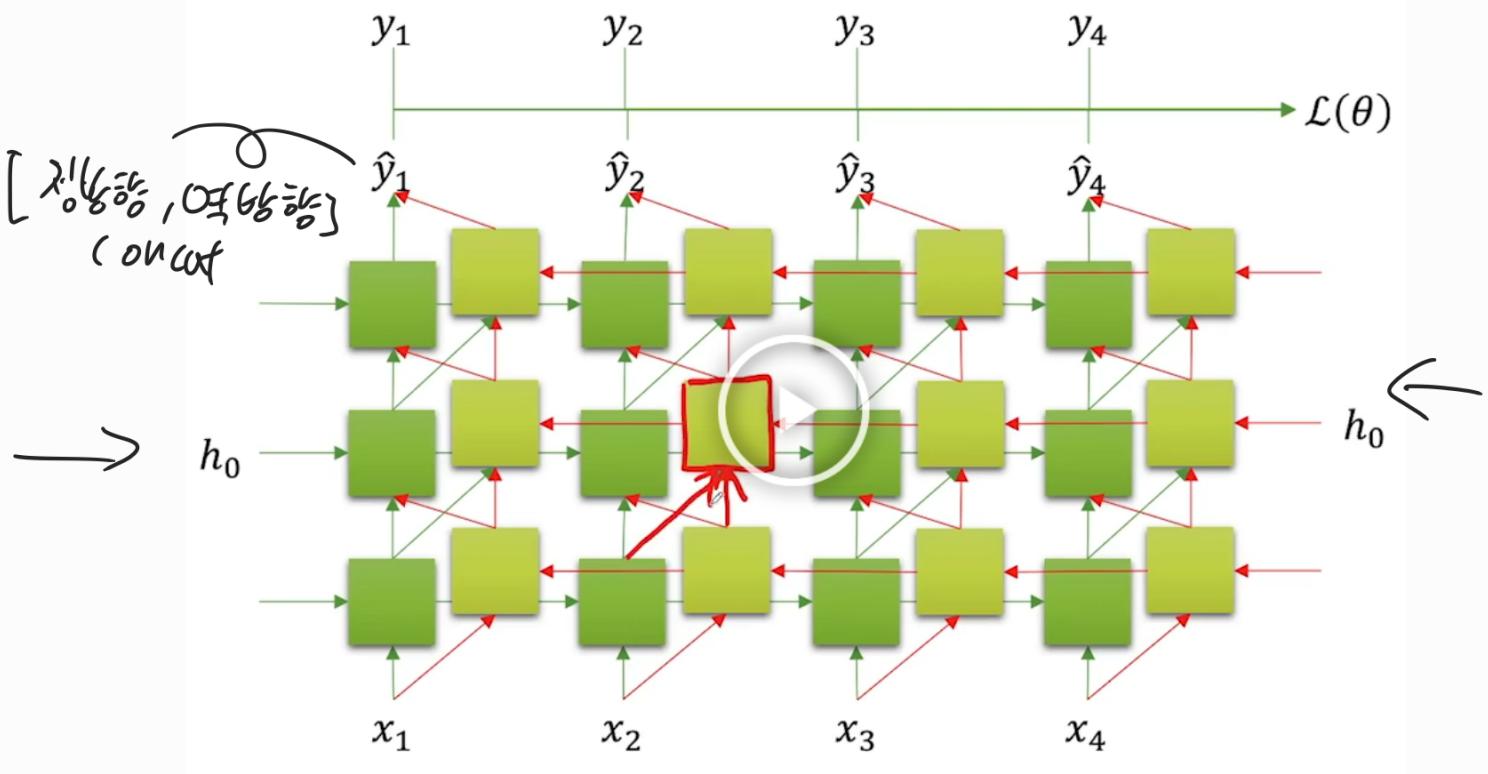
$$[y_{1:n}] = (\text{batch_size}, n, \text{hidden_size})$$

hidden state

$$\underline{h_t} = (n\text{-layers}, \text{batch_size}, \text{hidden_size})$$



Bidirectional Multi-layered RNN



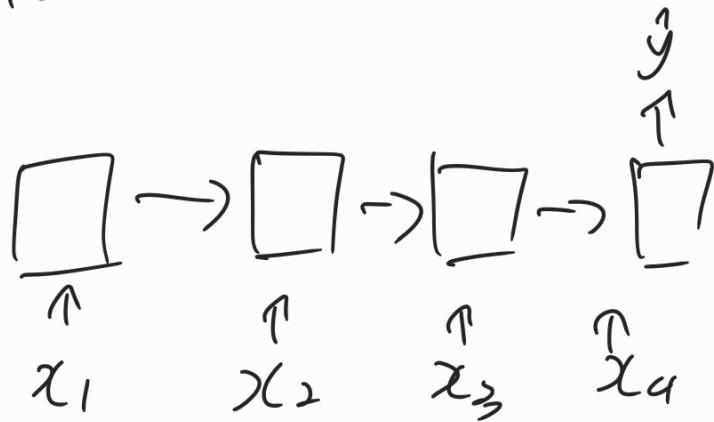
output tensor

$$|h_{1:n}| = (\text{batch-size}, n, \text{hidden-size} \times \underbrace{\text{direction}}_2)$$

hidden state tensor

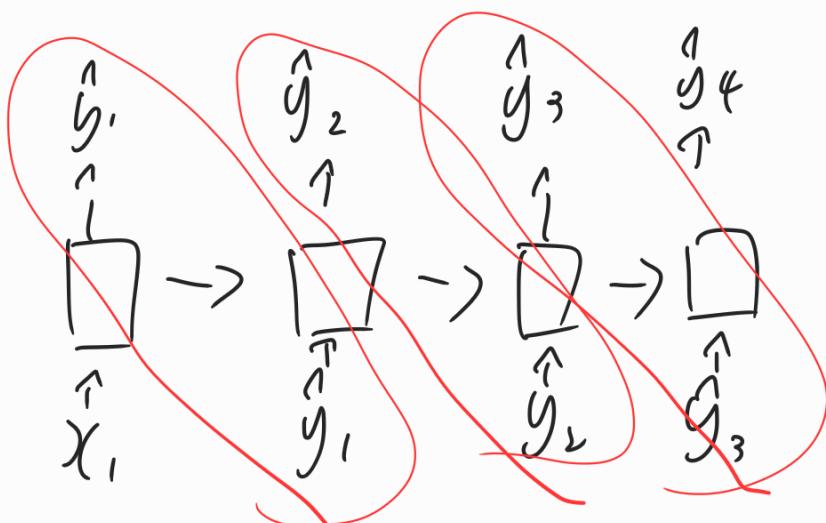
$$h^+ = (\text{direction} \times \text{layers}, \text{batch-size}, \text{hidden-size})$$

Method



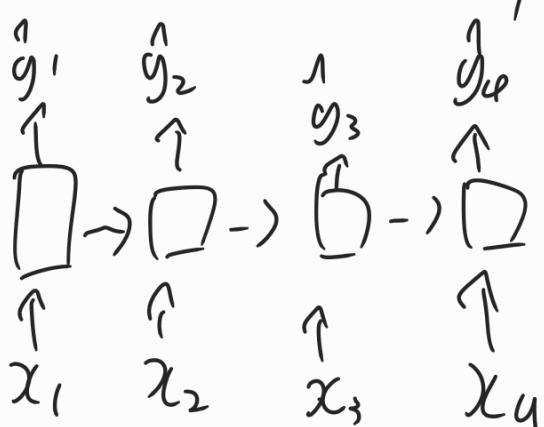
Text classification

Many to one



NLG
Machine translation
generation

one to Many



영어로 번역...

Many to Many

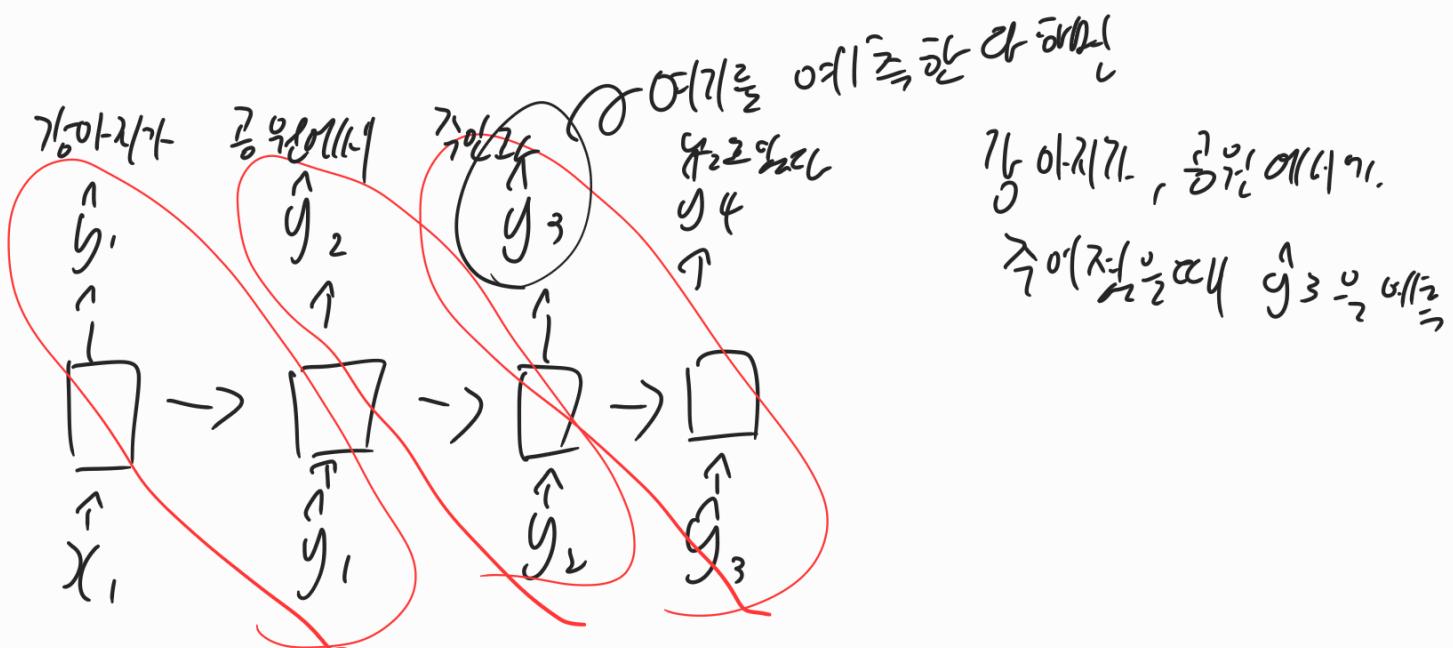
기반 학습법

No 1 - Auto regressive (Non-generative)

- 현재 상태가 앞/뒤 상태를 통해 정해지는 경우
ex) 문장 전체를 분석하는 경우

Auto regressive (Generative)

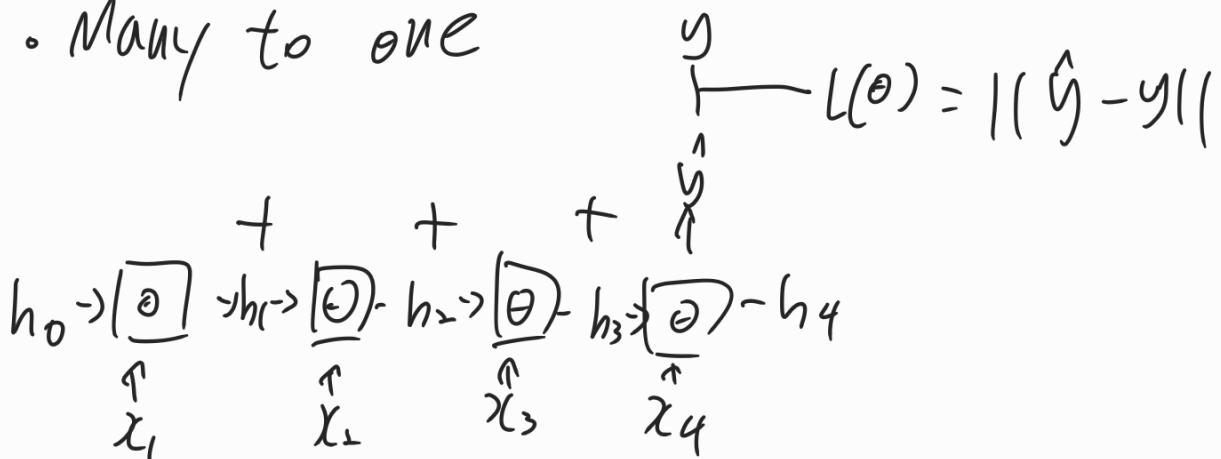
- 현재 상태가 과거 상태에 의존하여 정해지는 경우
ex) NLG, Machine translation
- One to Many 예측
- Bidirectional Rnn 사용 불가



BPTT

- Back propagation through time

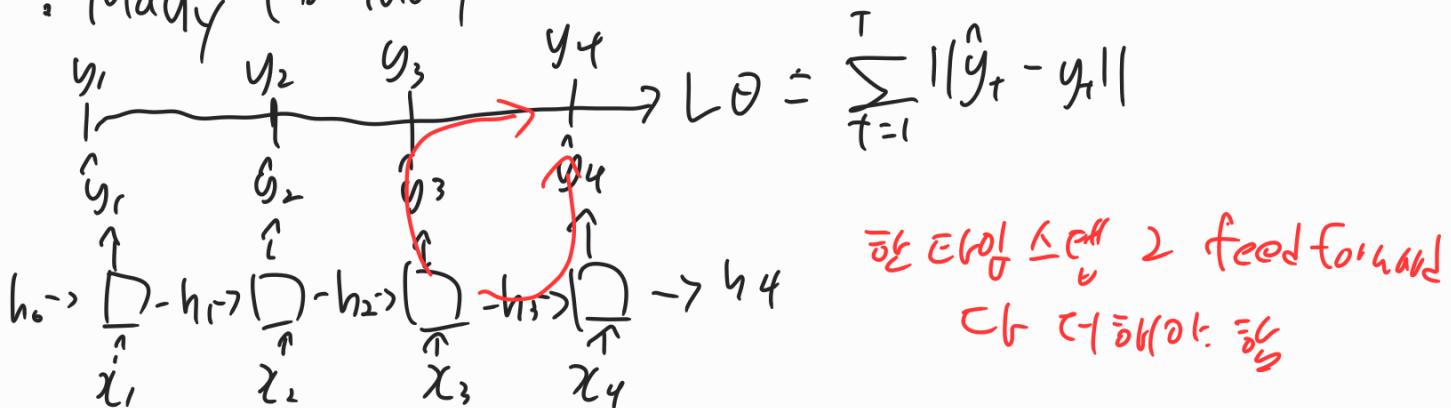
- Many to one



η_t cell의 gradient \hat{y}_t 셀에 update

Sequence η_t 의 gradient 흐름이 일어날 수록 있다.

- Many \rightarrow Many



한 번에 θ 를 2 feed forward
으로 업데이트하는

- Many to Many with Multi-layered RNN

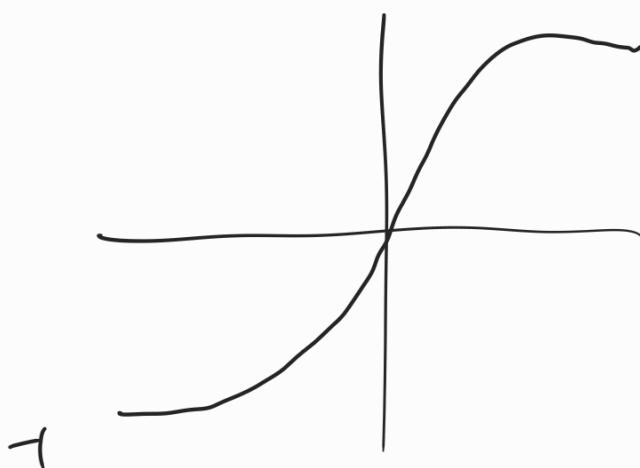
Many to Many 를 하지 않아 그 이유는 gradient 를
传播하지 않다.

$$\hat{y}_t = h_t = f(x_t, h_{t-1}; \theta)$$

$$= \tanh(w_{in}x_t + b_{in} W_{hh}h_{t-1} + b_{hh})$$

$$\text{where } \Theta = \{w_{in}, b_{in}, W_{hh}, b_{hh}\}$$

\tanh



$$\frac{d\tanh}{dx}$$

$$1 - \tanh^2(x \neq 0)$$

그러나 gradient 를 계산할 때 gradient vanishing

이 있는 경우.

Long sequence \leftarrow gradient update

$D \rightarrow D \rightarrow D \rightarrow \dots$

gradient vanishing으로 gradient 를 계산할 때 네트워크는 최근 값만 update 한다

Summary

- 초기 gradient 값은 각 경로의 gradient들의 총 합이 된다.
- time step T 레이어의 값이 진짜 fail \hat{y} 를 활용함으로 사용하는 RNN의 gradient vanishing이 발생한다.

Moving to one backpropagation

$$h_T = f(x_T, h_{T-1}; \psi) \quad L(\theta) = ||\hat{y} - y||$$

$$\hat{y} = g(h_T; \phi)$$

$$\theta = \{\phi, \psi\}$$

$$\frac{\partial L(\theta)}{\partial \theta} = \frac{\partial L(\theta)}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \phi}$$

$$\frac{\partial L(\theta)}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial h_4} \frac{\partial h_4}{\partial \psi} + \frac{\partial L(\theta)}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial h_3} \frac{\partial h_3}{\partial \psi} + \frac{\partial L(\theta)}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial h_2} \frac{\partial h_2}{\partial \psi} + \frac{\partial L(\theta)}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial h_1} \frac{\partial h_1}{\partial \psi}$$

$$= \sum_{t=1}^T \frac{\partial L(\theta)}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial h_t} \left(\prod_{i=t}^{T-1} \frac{\partial h_{i+1}}{\partial h_i} \right) \frac{\partial h_t}{\partial \psi}$$

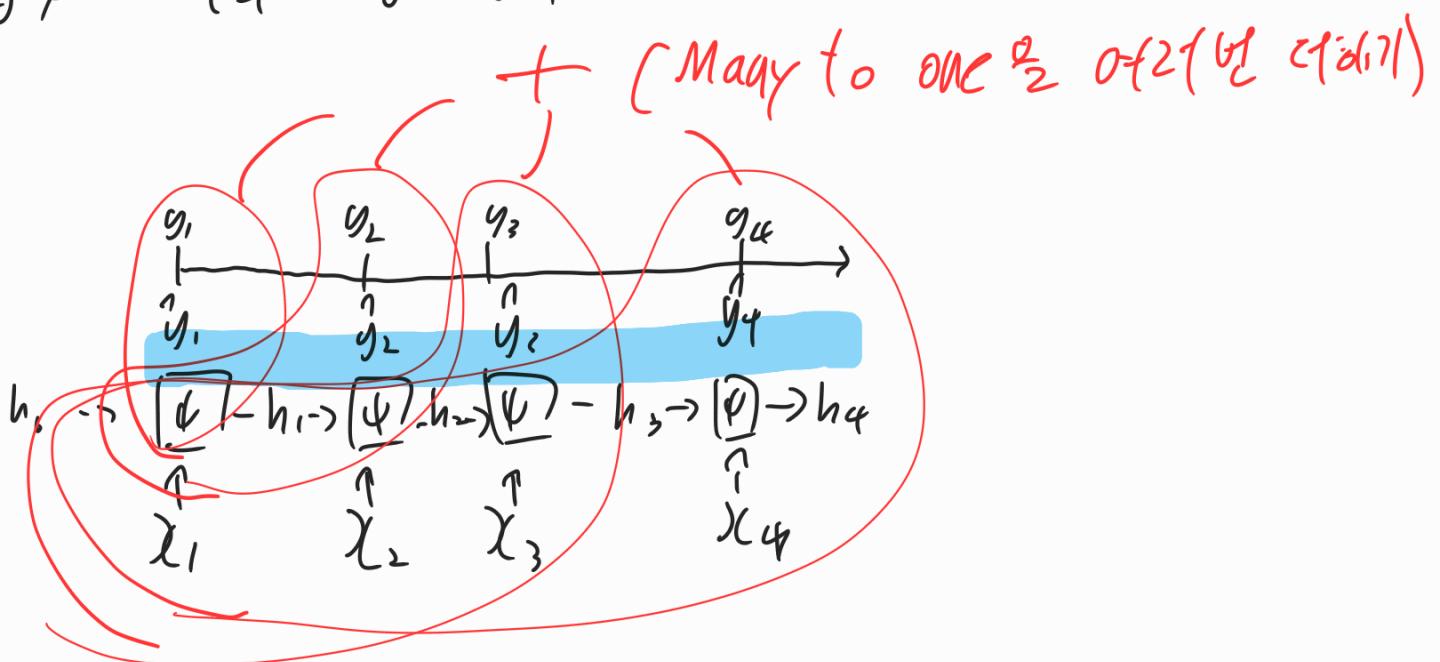
$$y_1 \quad y_2 \quad y_3 \quad y_4 \rightarrow L(\theta) = \sum_{t=1}^T ||\hat{y}_t - y_t||$$

$$h_i \rightarrow \boxed{\psi} - h_i \rightarrow \boxed{\psi} - h_i \rightarrow \boxed{\psi} - h_i \rightarrow \boxed{\psi} \rightarrow h_4$$

$\uparrow \quad \uparrow \quad \uparrow \quad \uparrow$

$x_1 \quad x_2 \quad x_3 \quad x_4$

$$\frac{\partial L(\theta)}{\partial \phi} = \sum_{t=1}^T \frac{\partial L(\theta)}{\partial y_t} \frac{\partial y_t}{\partial \phi}$$



LSTM

- Long Short Term Memory
- RNN의 Gradient vanishing의 문제를 개선하기 위해 등장

Sigmoid를 통한 초기 고정(기능적으로는 문을 열고 닫는 것처럼 표현)

- LSTM은 여러 미트를 가지고 있고 이는 상태의 일부의 초기를 조절하여 학습하고 그의 hidden state를 출력한다.

GRU

- LSTM과 동일하게 gradient vanishing을 해결한다.
- LSTM과 유사한 구조다.

Summary

- LSTM vs GRU
 - LSTM의 층이란 주제
- LSTM은 RNN의 초기 청진 문제는 파라미터를 가진다.
- LSTM의 gradient vanishing을 해결 하였지만, 긴 대여제를 모두 가능 X
 - Attention을 통해 해결