Processing sequences using RNN and CNN

Hands-On Machine Learning Part2
& Deep Learning from Scratch 3

TAVE Research DL001
Heeji Won

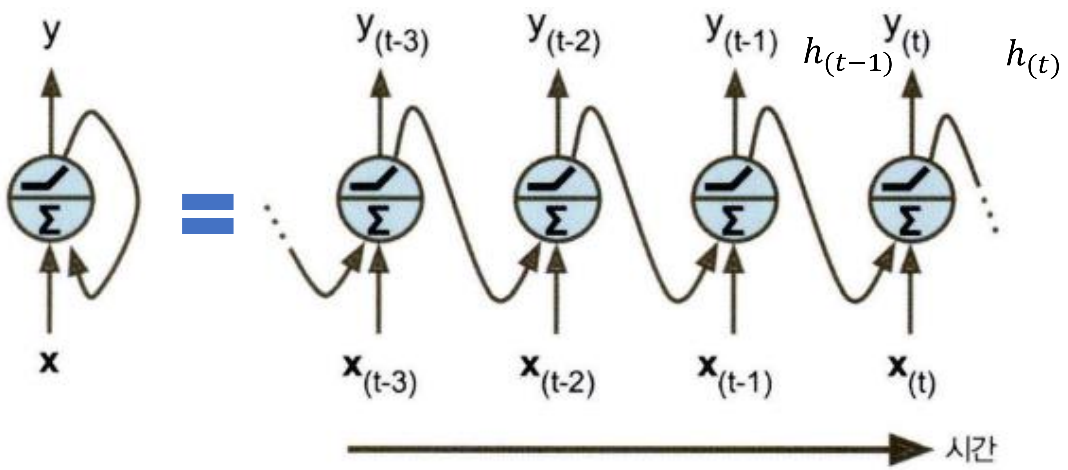
Contents

Contents

01. RNN

: Sequence model for sequential time series data

- Recurrent Neuron



Unrolling the network through time

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

✓ The same function and the same parameters are used at every time step!

- Memory Cell

– Outputs of recurrent neurons is a function of all inputs (form of memory)

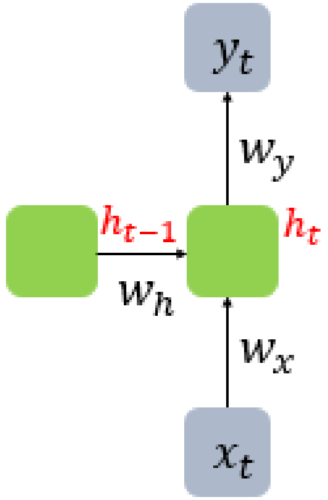
– Memory cell is a cell conserving a state

- h_t : Cell state at time step t

- (Vanilla) RNN

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

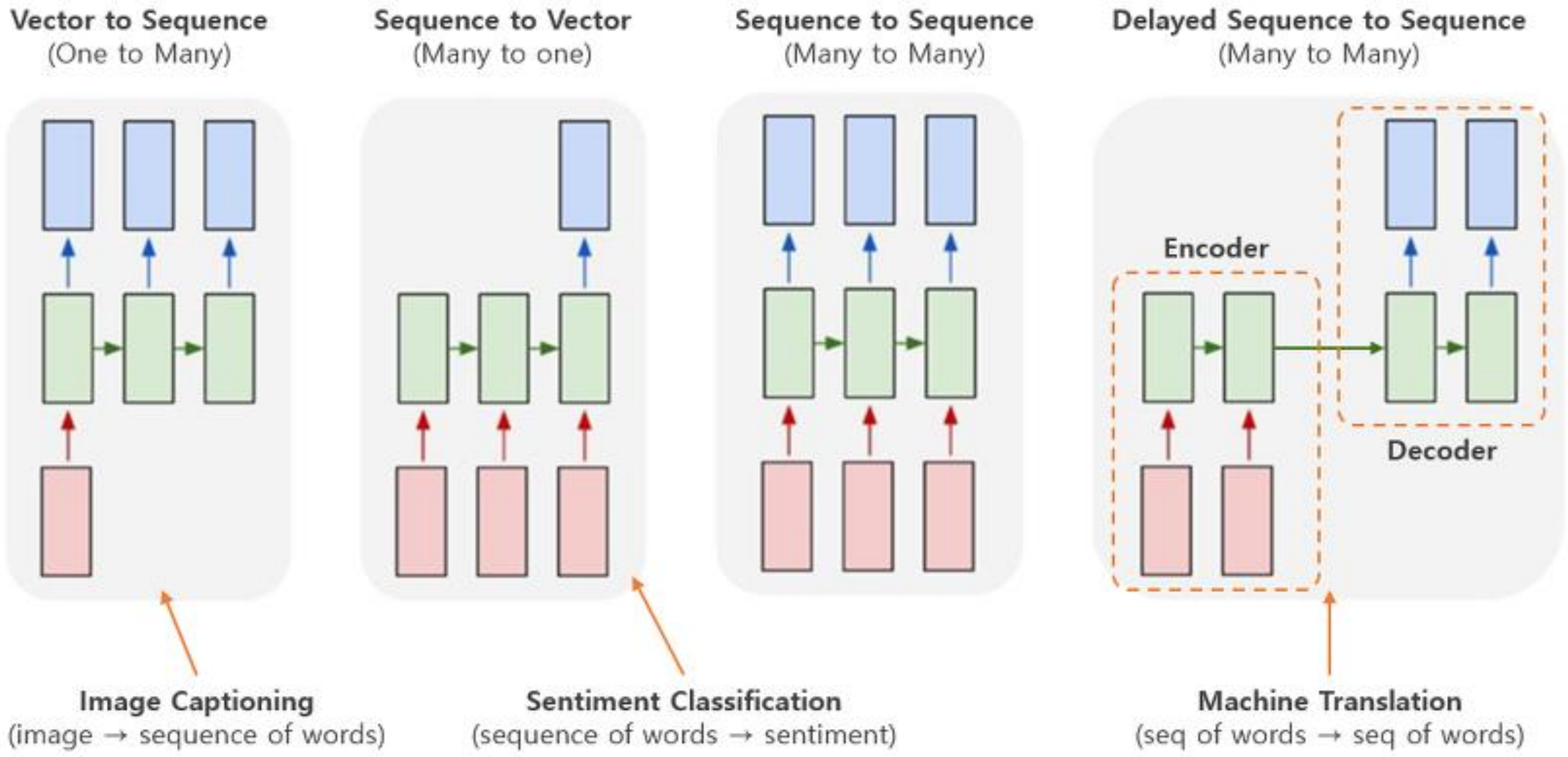
$$y_t = W_{hy}h_t$$



01. RNN

: Sequence model for sequential time series data

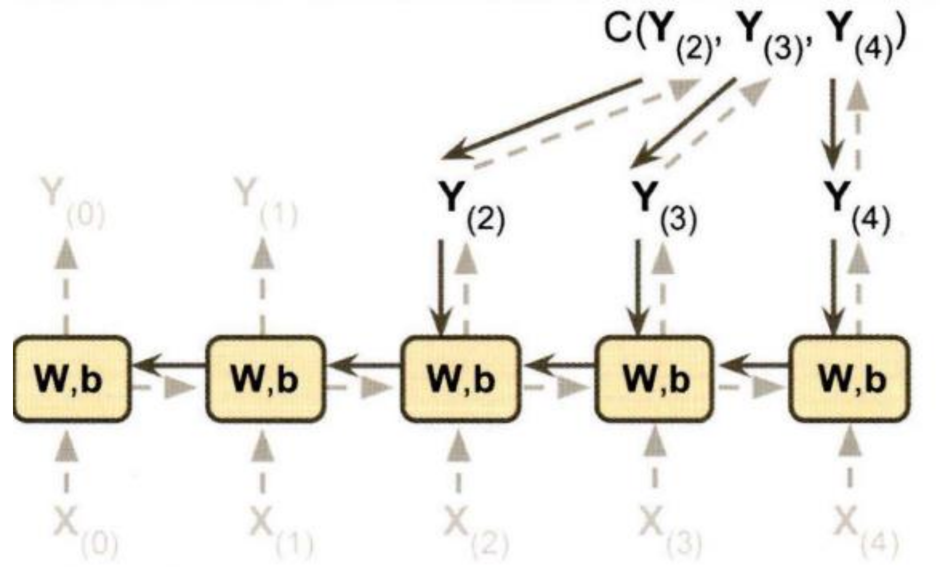
- Various Input and Output

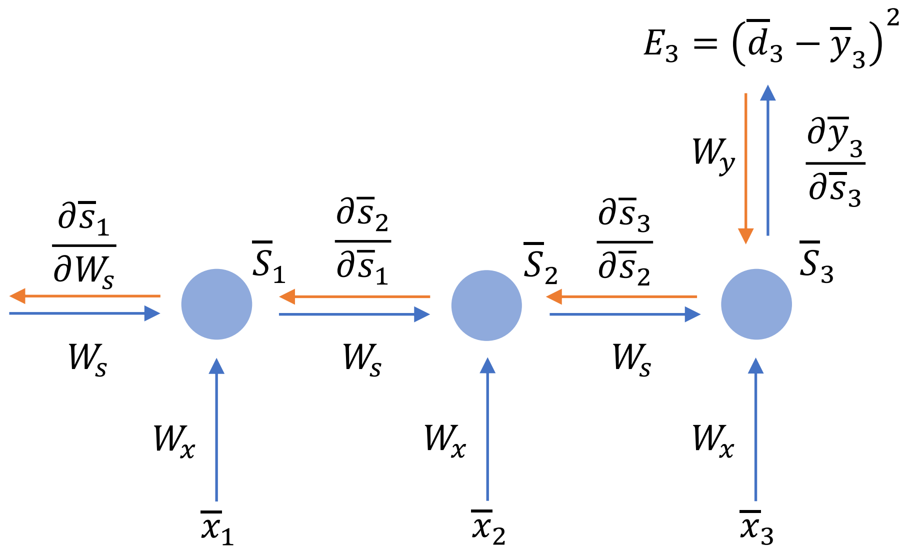


Contents

02. BPTT (backpropagation through time)

: Gradient-based technique for training RNN





- ✓ Backpropagate from outputs calculating cost function (ignoring other outputs)
- ✓ Accumulate gradients

  Why? Outputs depends on previous time steps

Accumulative Gradient at time t=3

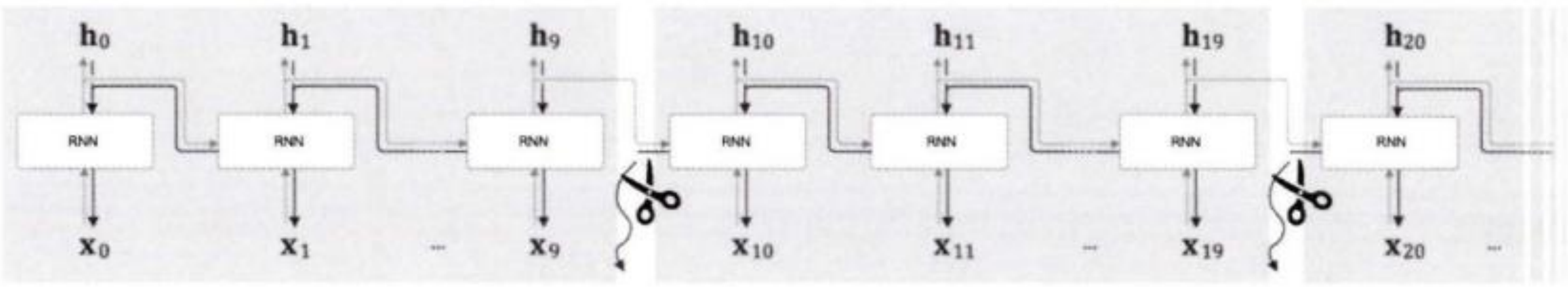
$$\frac{\partial E_3}{\partial W_s} = \frac{\partial E_3}{\partial \bar{y}_3} \cdot \frac{\partial \bar{y}_3}{\partial \bar{s}_3} \cdot \frac{\partial \bar{s}_3}{\partial W_s} + \frac{\partial E_3}{\partial \bar{y}_3} \cdot \frac{\partial \bar{y}_3}{\partial \bar{s}_3} \cdot \frac{\partial \bar{s}_3}{\partial \bar{s}_2} \cdot \frac{\partial \bar{s}_2}{\partial W_s}$$
$$+ \frac{\partial E_3}{\partial \bar{y}_3} \cdot \frac{\partial \bar{y}_3}{\partial \bar{s}_3} \cdot \frac{\partial \bar{s}_3}{\partial \bar{s}_2} \cdot \frac{\partial \bar{s}_2}{\partial \bar{s}_1} \cdot \frac{\partial \bar{s}_1}{\partial W_s}$$

$$\frac{\partial \mathrm{E_N}}{\partial \mathrm{W_s}} = \sum_{i=1}^{N} \frac{\partial E_N}{\partial \bar{y}_N} \cdot \frac{\partial \bar{y}_N}{\partial \bar{s}_i} \cdot \frac{\partial \bar{s}_i}{\partial W_s}, \quad \frac{\partial \mathrm{E_N}}{\partial \mathrm{W_x}} = \sum_{i=1}^{N} \frac{\partial E_N}{\partial \bar{y}_N} \cdot \frac{\partial \bar{y}_N}{\partial \bar{s}_i} \cdot \frac{\partial \bar{s}_i}{\partial W_x}$$

02. BPTT (backpropagation through time)

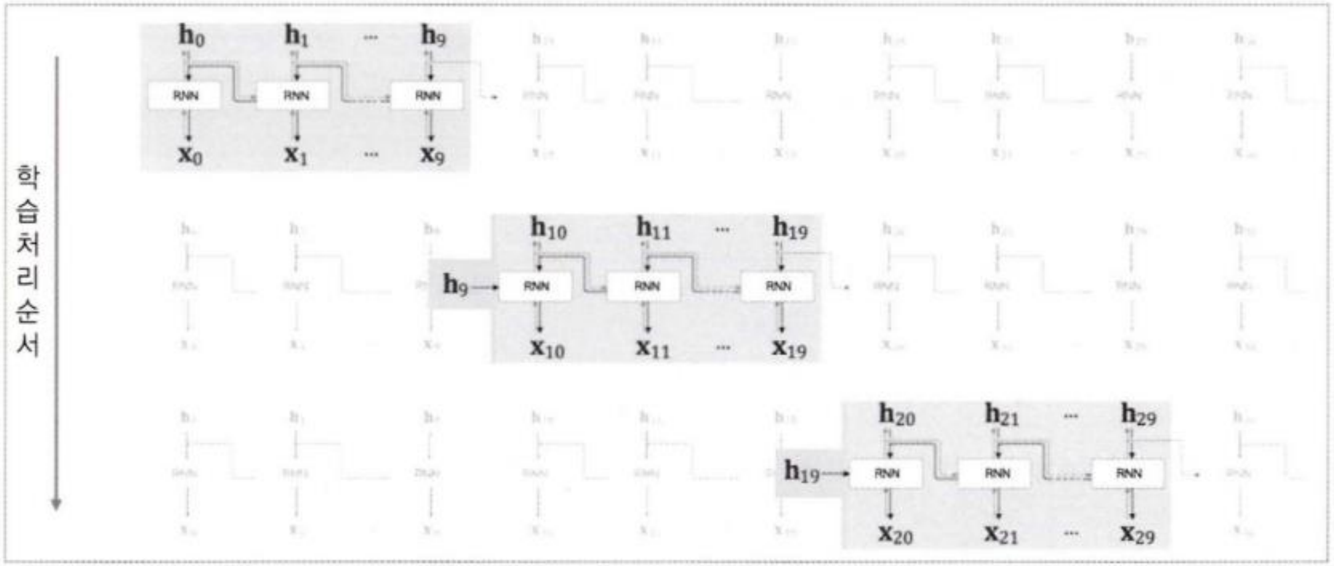
- Truncated BPTT

– BPTT has vanishing and exploding gradient problems and lots of calculations => **Truncate!**



✓ Truncate only **Backward Pass** (not forward pass)

✓ Generally truncate per about 5 steps



Contents

03. Overcoming the Unstable Gradient Problem

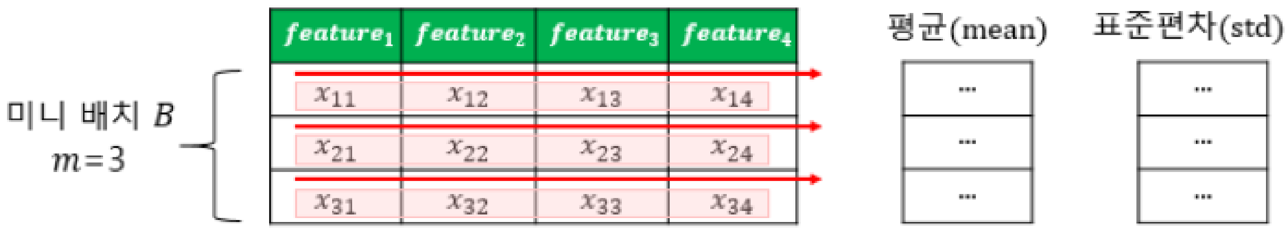
- ## Non-Converged function

✓ Non-converged function make exploding gradients (or outputs)
✓ Why? the same weights are used in every time steps

=> Gradient Clipping

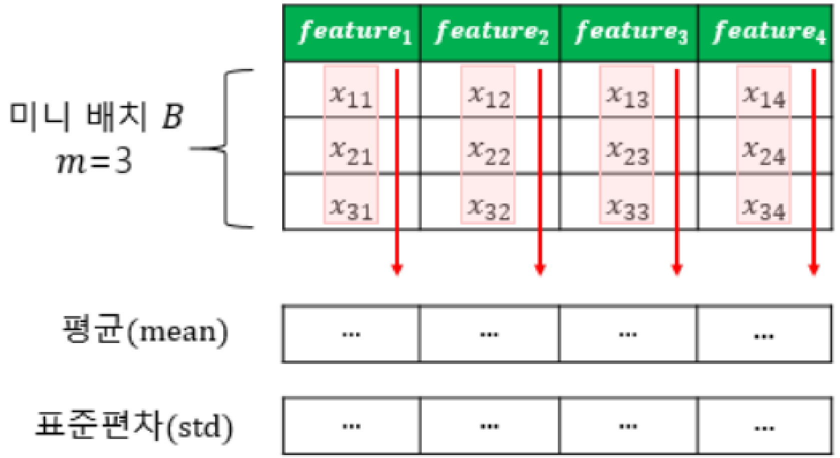
- ## Layer Normalization

: normalization features of each sample using each parameter



✓ Also, we can use Dropout

cf) Batch Normalization

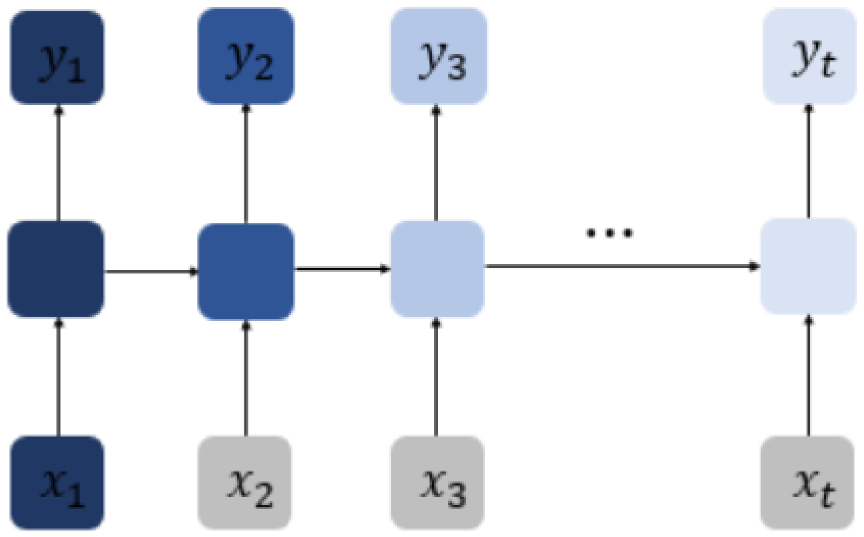


Contents

03. Overcoming the Short-Term Memory

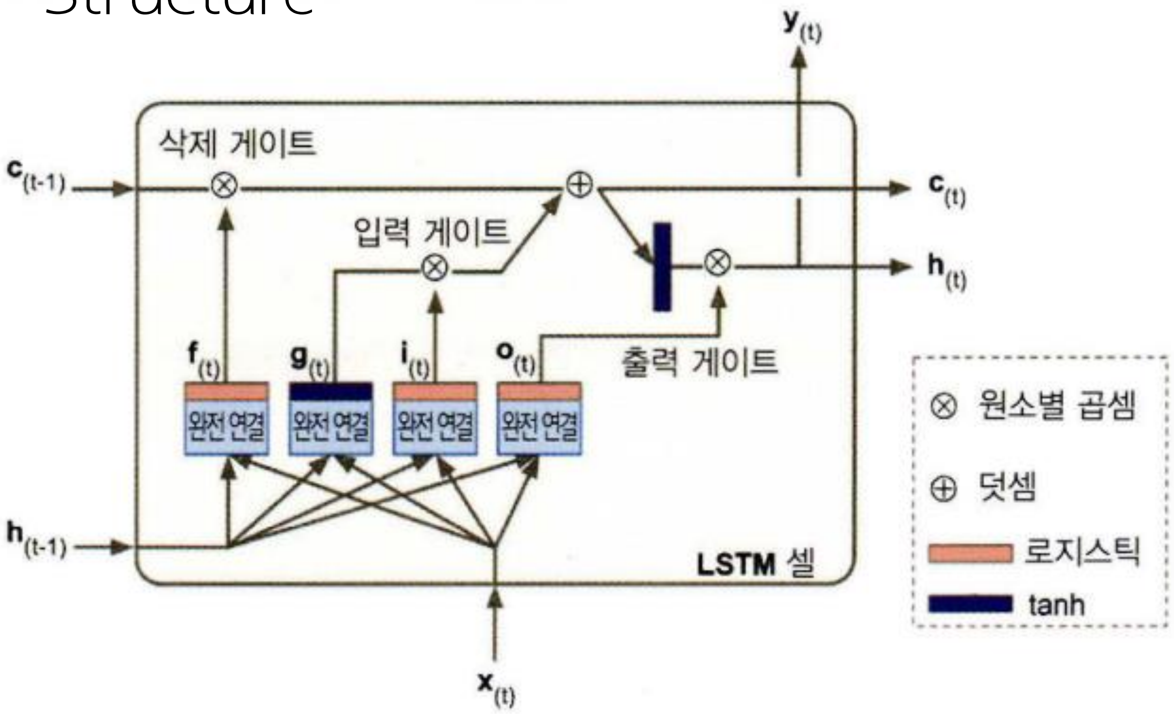
➢ LSTM Cell (Long short-term memory)

- Solve Long-Term Dependencies problem



✓ As t increases, RNN cannot preserve information over many time steps

- Structure



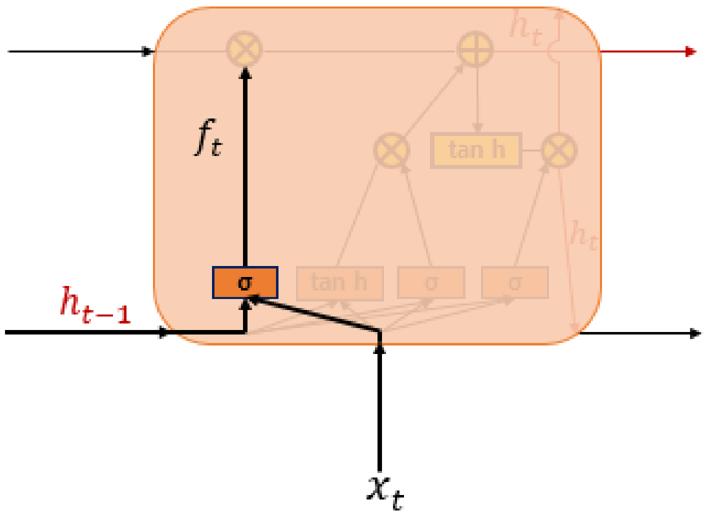
There are three gates
(Input gate, Forget gate, Output gate)

03. Overcoming the Short-Term Memory

➢ LSTM Cell (Long short-term memory)

- Forget Gate ($\boldsymbol{f_t}$)

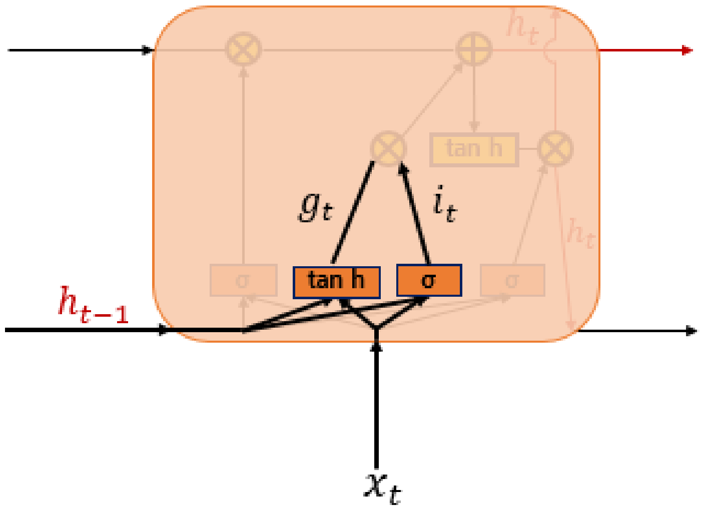
- Gate for forgetting the cell's memory
- Control which part of c_{t-1} have to be forgotten



$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f)$$

- Input Gate ($\boldsymbol{i_t}$)

- Gate for inputting current information
- Control which part of the candidate value, g_t have to be added to cell's memory



$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \longrightarrow 0 \sim 1$$

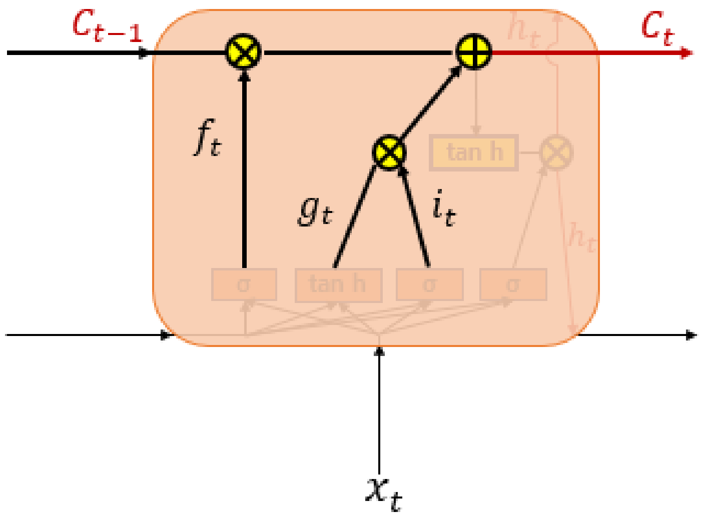
$$g_t = tanh(W_{xg}x_t + W_{hg}h_{t-1} + b_g) \longrightarrow -1 \sim 1$$

03. Overcoming the Short-Term Memory

➢ LSTM Cell (Long short-term memory)

- Cell state(long-term) c_t

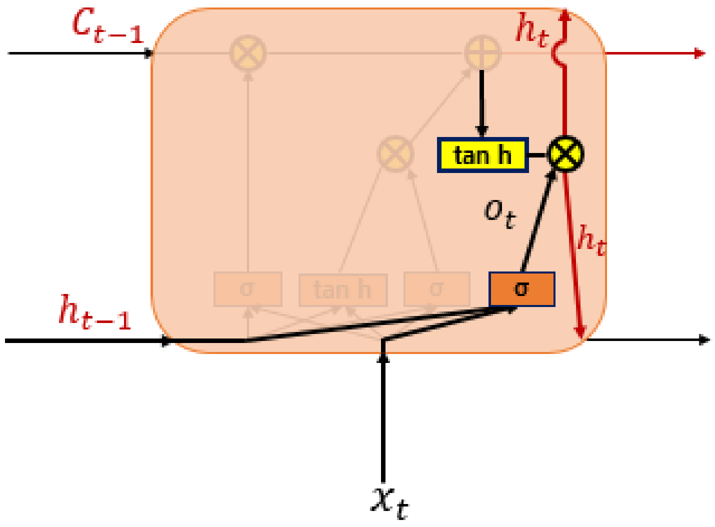
– Update the old cell state c_{t-1}



$$C_t = f_t \circ C_{t-1} + i_t \circ g_t$$

- Output gate ($\boldsymbol{o_t}$) & Cell state(short-term) $\boldsymbol{h_t}$

– Gate for deciding hidden state

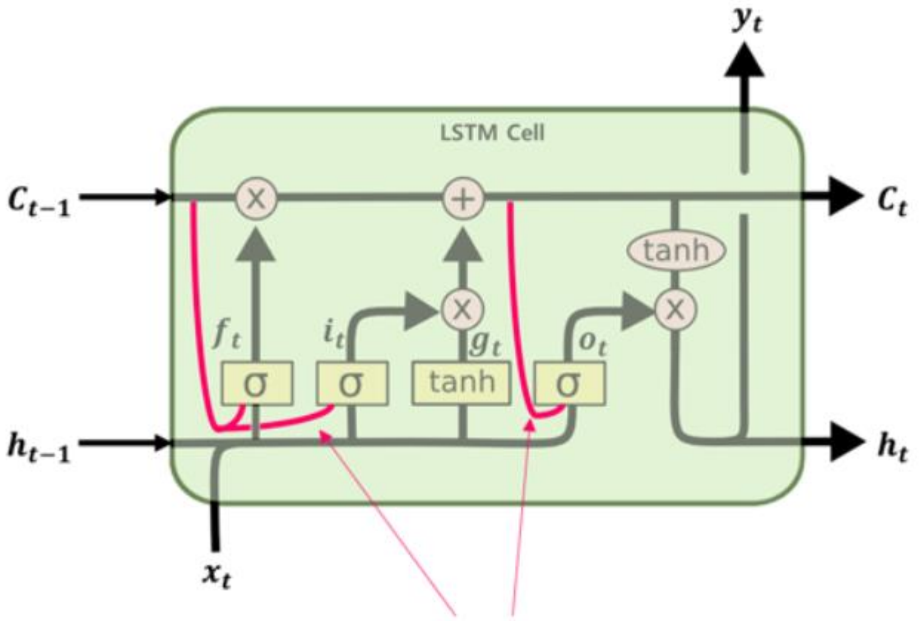


$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o)$$
$$h_t = o_t \circ tanh(c_t)$$

03. Overcoming the Short-Term Memory

➢ Peephole Connection

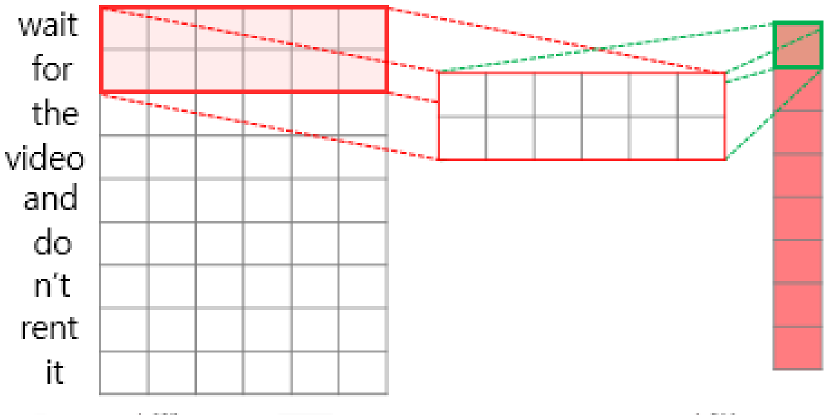
- Gates also consider the long-term memory, $\boldsymbol{c_{t-1}}$



$$\mathbf{f}_t = \sigma\left(\mathbf{W}_{cf}^T \cdot \mathbf{c}_{t-1} + \mathbf{W}_{xf}^T \cdot \mathbf{x}_t + \mathbf{W}_{hf}^T \cdot \mathbf{h}_{t-1} + \mathbf{b}_f\right)$$
$$\mathbf{i}_t = \sigma\left(\mathbf{W}_{ci}^T \cdot \mathbf{c}_{t-1} + \mathbf{W}_{xi}^T \cdot \mathbf{x}_t + \mathbf{W}_{hi}^T \cdot \mathbf{h}_{t-1} + \mathbf{b}_i\right)$$
$$\mathbf{o}_t = \sigma\left(\mathbf{W}_{co}^T \cdot \mathbf{c}_{t} + \mathbf{W}_{xo}^T \cdot \mathbf{x}_t + \mathbf{W}_{ho}^T \cdot \mathbf{h}_{t-1} + \mathbf{b}_o\right)$$

➢ 1D Convolution layer

- 1D Conv layer can help detect long-term pattern by reducing length of sequence



1D CNN (kernel size : 2)

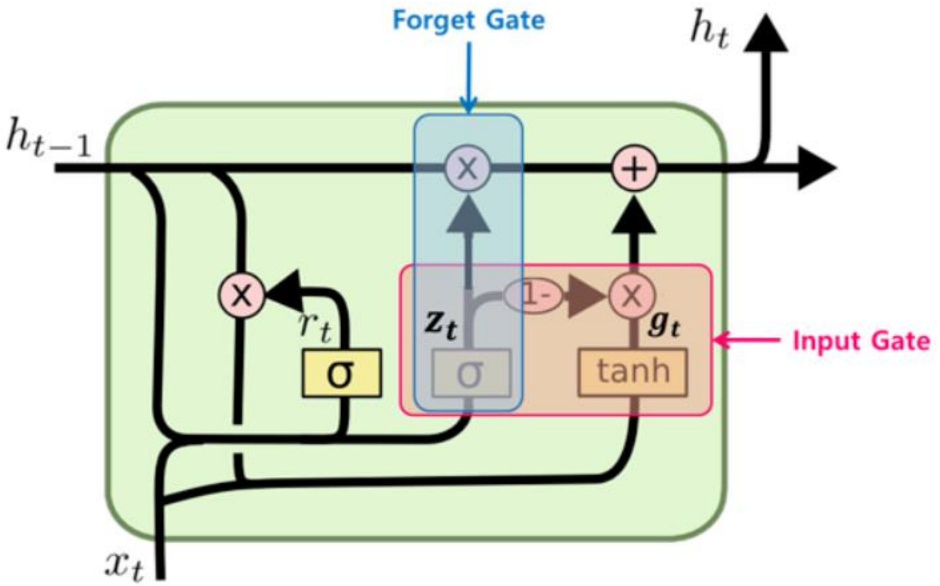
Q : Why is this a 1D CNN not a 2D CNN?

A : Kernel moves in **1 direction**

03. Overcoming the Short-Term Memory

➢ GRU Cell (gated recurrent unit)

- Simplified version of LSTM



$$\mathbf{r}_t = \sigma\left(\mathbf{W}_{xr}^T \cdot \mathbf{x}_t + \mathbf{W}_{hr}^T \cdot \mathbf{h}_{t-1} + \mathbf{b}_r\right)$$
$$\mathbf{z}_t = \sigma\left(\mathbf{W}_{xz}^T \cdot \mathbf{x}_t + \mathbf{W}_{hz}^T \cdot \mathbf{h}_{t-1} + \mathbf{b}_z\right)$$
$$\mathbf{g}_t = \tanh\left(\mathbf{W}_{xg}^T \cdot \mathbf{x}_t + \mathbf{W}_{hg}^T \cdot (\mathbf{r}_t \otimes \mathbf{h}_{t-1}) + \mathbf{b}_g\right)$$
$$\mathbf{h}_t = \mathbf{z}_t \otimes \mathbf{h}_{t-1} + (1 - \mathbf{z}_t) \otimes \mathbf{g}_t$$

✓ One cell state, h_t

✓ Two gates, Reset gate(r_t) and Update gate(z_t)

✓ Reset gate control how much information of g_t will be considered

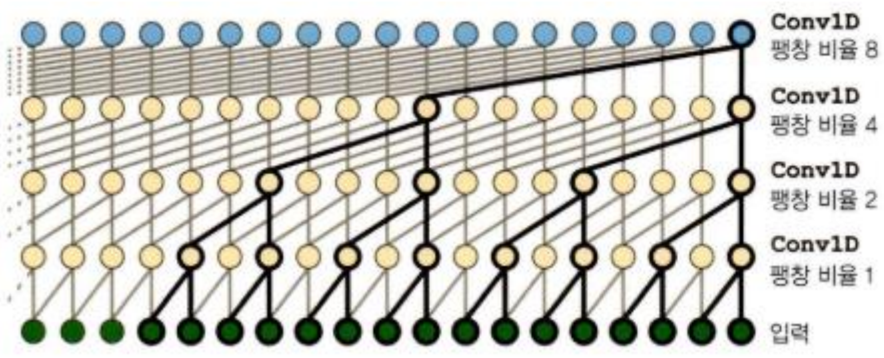
✓ Update gate control forget gate and input gate

- If output of z_t is 1, open **forget** gate and close input gate
- If output of z_t is 0, close forget gate and open **input** gate

03. Overcoming the Short-Term Memory

➢ WaveNet

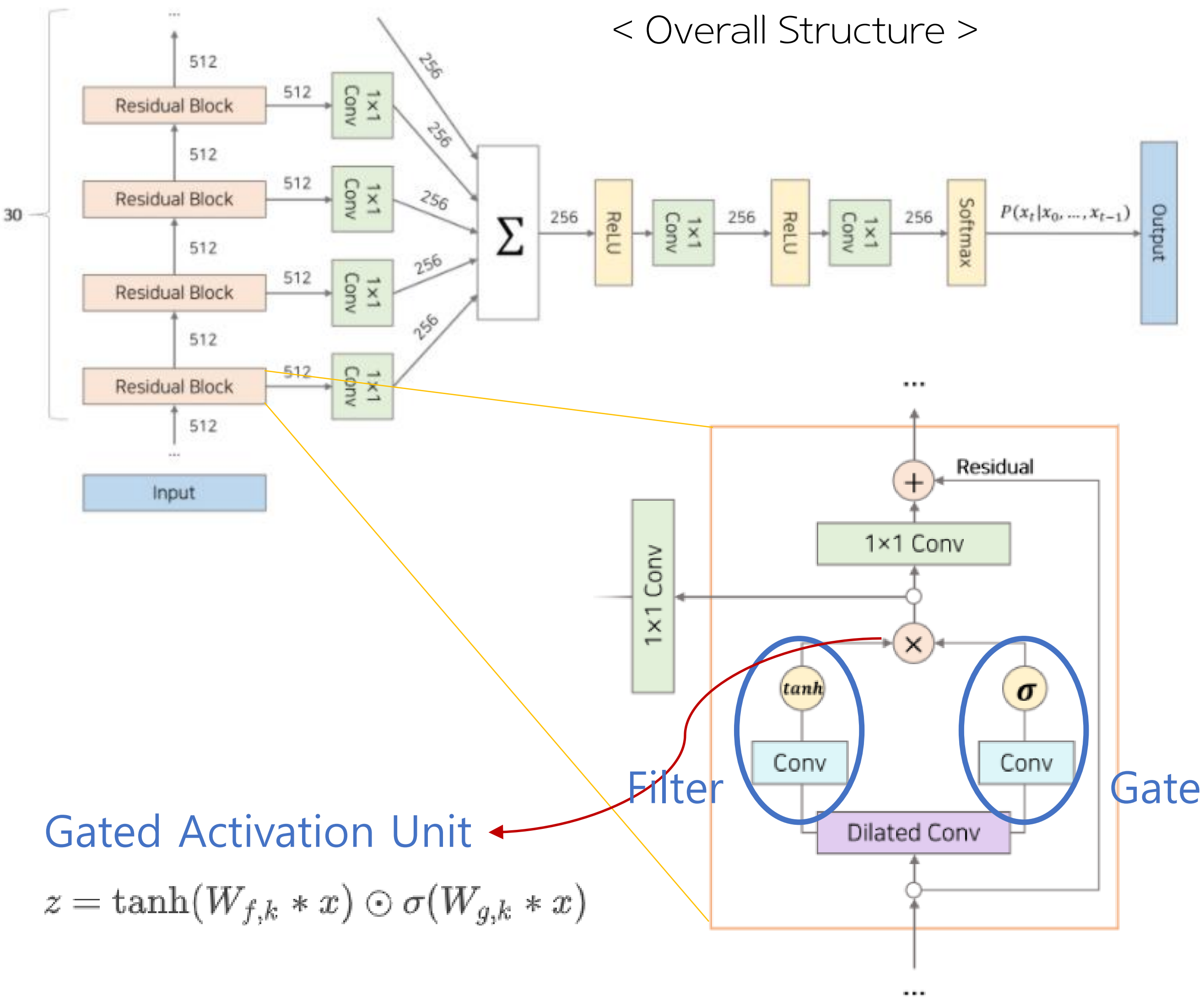
- Dilated Convolution Layer



- The dilation rate is doubled for each layer
- The lower layers may identify short-term patterns, while the higher layers may identify long-term patterns

✓ **Causal Padding** : padding the layer's input with zeros in the front

< Overall Structure >



Gated Activation Unit

$$z = \tanh(W_{f,k} * x) \odot \sigma(W_{g,k} * x)$$

Contents

Appendix. Deep Learning from Scratch

- Define-and-Run
  - Network is defined and fixed. And then data are fed into the predefined network
  - Easy to optimize
  - difficult to debug

```
# 계산 그래프 정의
a = Variable('a')
b = Variable('b')
c = a + b
d = c + Constant(1)

# 계산 그래프 컴파일
f = compile(d)

# 데이터 흘려보내기
d = f(a=np.array(2), b=np.array(3))
```

- Define-by-Run
  - Network is defined dynamically via the actual forward computation
  - This dynamic definition allows conditionals and loops into the network easily

Mini
Batch

```
a = Variable(np.ones(10))
b = Variable(np.ones(10) * 2)
c = b * a
d = c + 1
```

Thank you