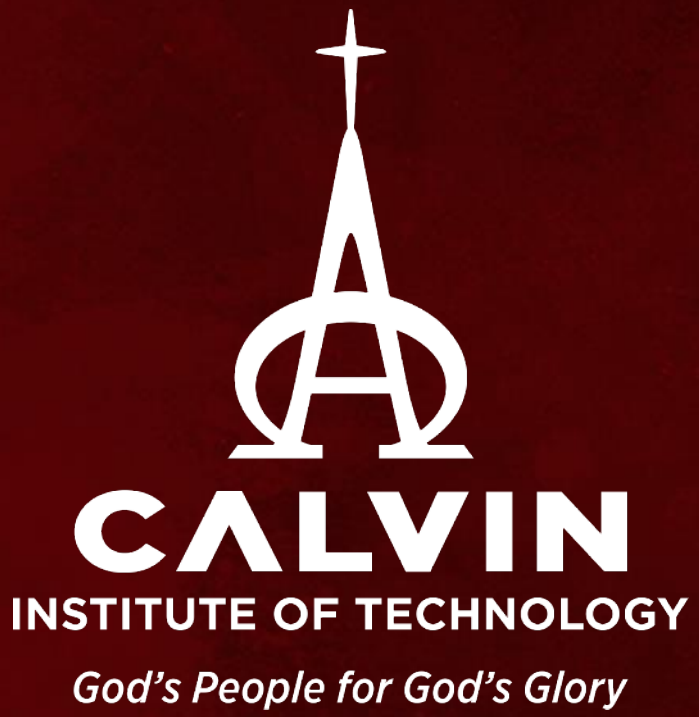


Sequence Model

Hendrik Santoso Sugiarto

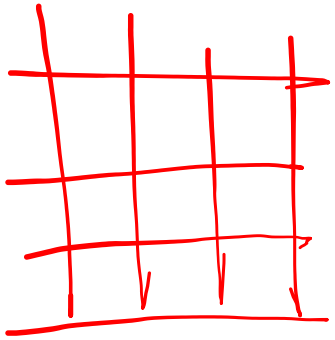
Capaian Pembelajaran

- Recurrent Neural Network
  - Mengerti dan mengaplikasikan RNN
- LSTM & ~~RNN~~ GRU
  - Mengerti kelemahan RNN
  - Mengeerti dan menggunakan LSTM & GRU untuk menyelesaikan kelemahan RNN

RNN

Recurrent Neural Network

- CNN efektif dalam memproses **grid** data
- RNN efektif dalam memproses **sequence** data



Motivasi

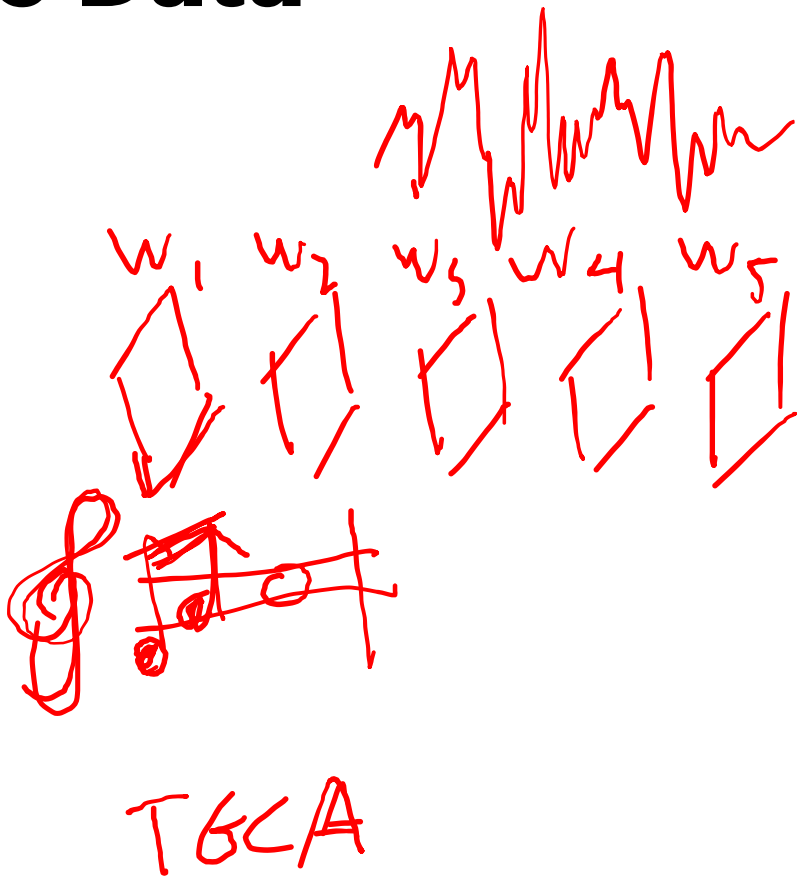
- Misalkan ada Neural Network yang memiliki input/output berikut
- Input: I am studying deep learning
- Output: Saya sedang belajar pembelajaran yang mendalam

- Contoh tersebut menunjukkan bahwa:
  - Konteks dan pola urutan sebelum-sesudah (sequence) sangat penting
  - Terdapat jenis data dengan input maupun output berbentuk sequence
  - Sulit untuk membuat fixed context window (mungkin ada kalimat yang lebih panjang dari sebelumnya)

CALVIN INSTITUTE OF TECHNOLOGY

Sequence Data

- Time-Series
- Text
- Video
- Music
- DNA



Basis Neurosains

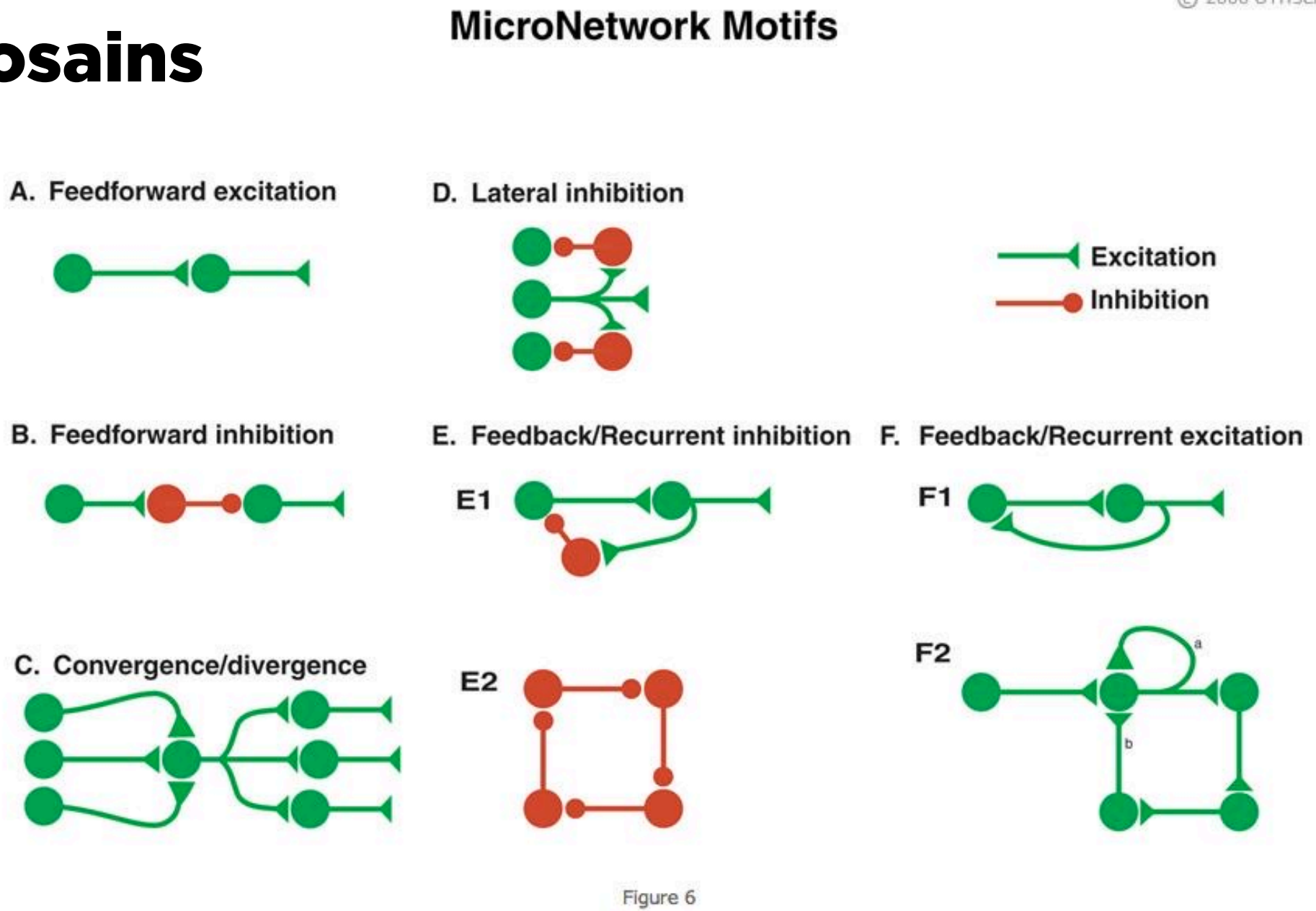
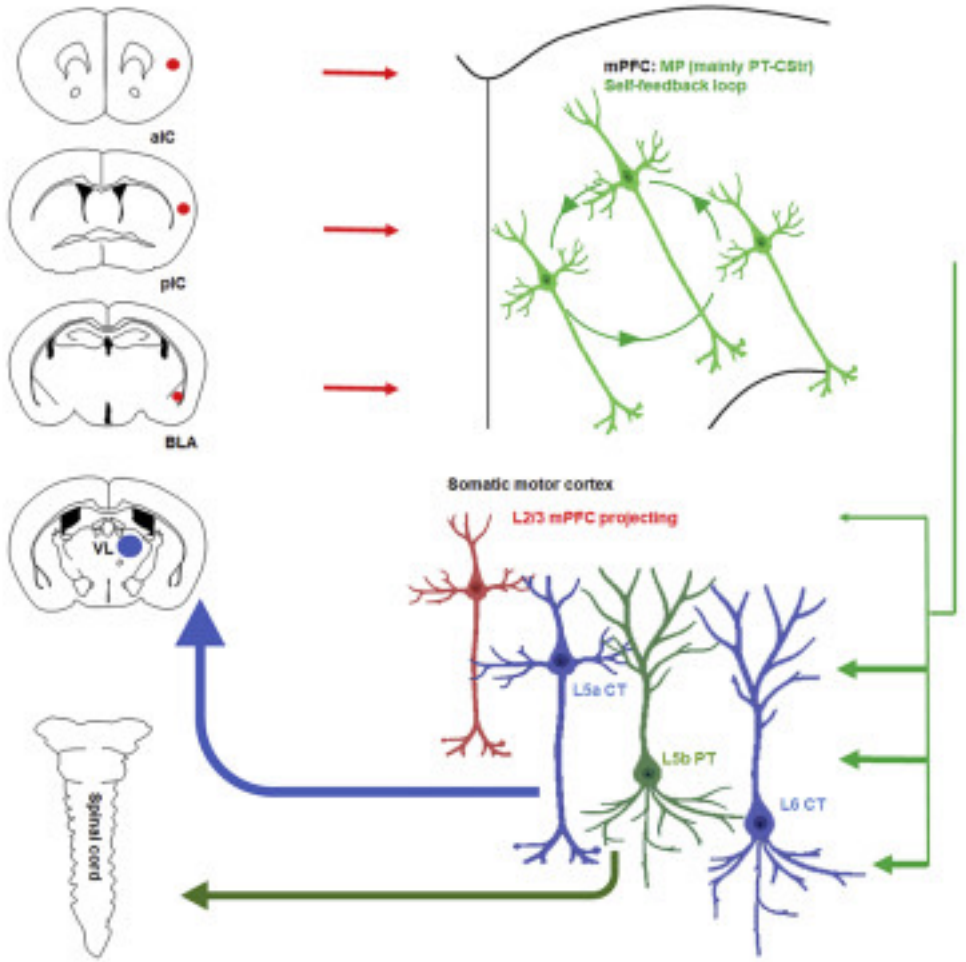
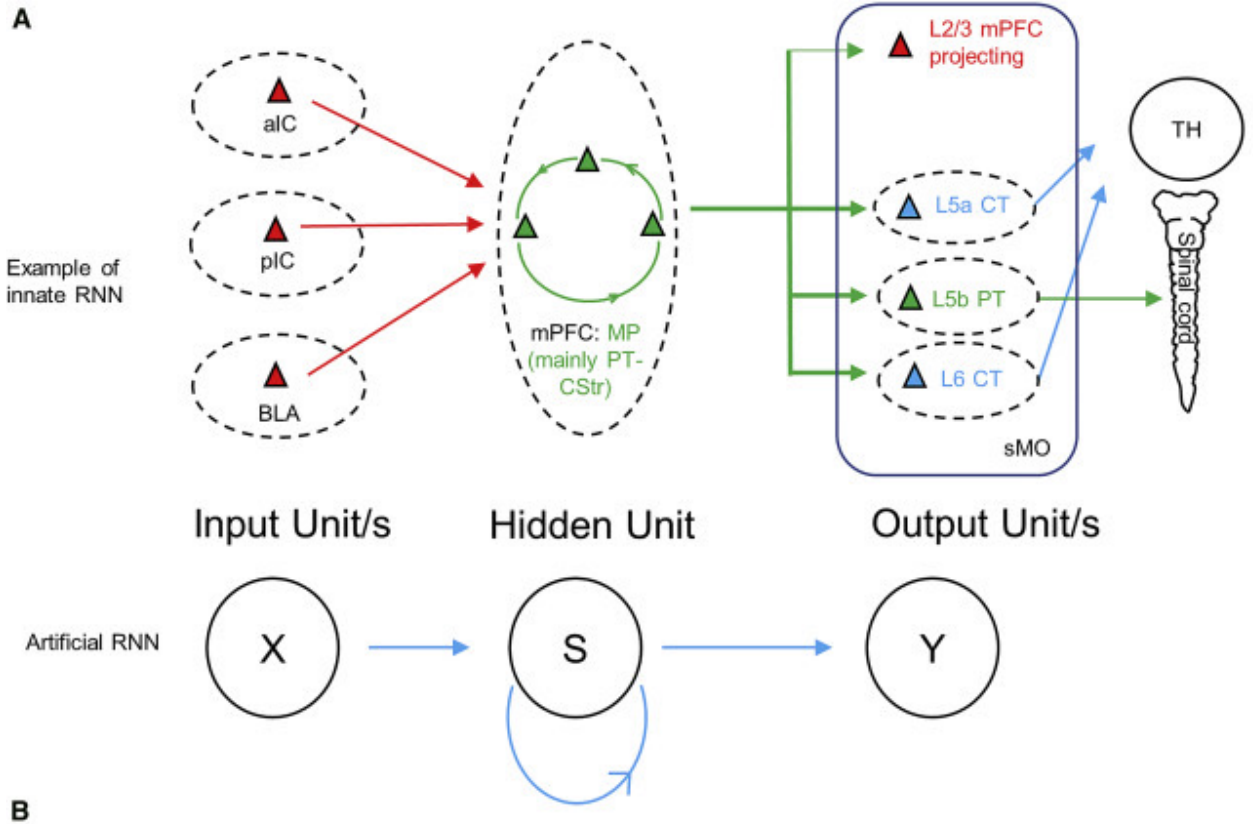
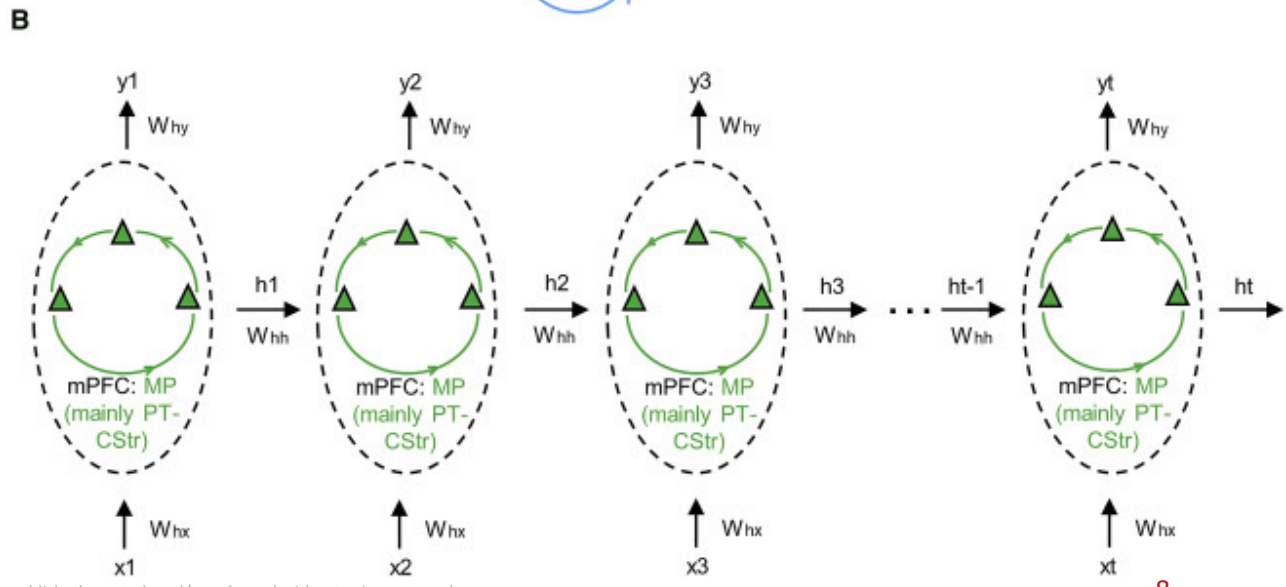


Figure 6

Basis Neurosains





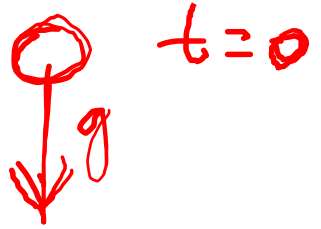


Recurrent

$v^{(1)} = v^{(0)} + g\Delta t = g$
$v^{(2)} = v^{(1)} + g\Delta t = g + g = 2g$

- Misalkan $\boldsymbol{s}^{(t)}$ adalah situasi (*state*) sebuah sistem dinamik pada waktu t
- Dan $\boldsymbol{\theta}$ adalah parameter lainnya pada fungsi $f()$
- Recurrent berarti:

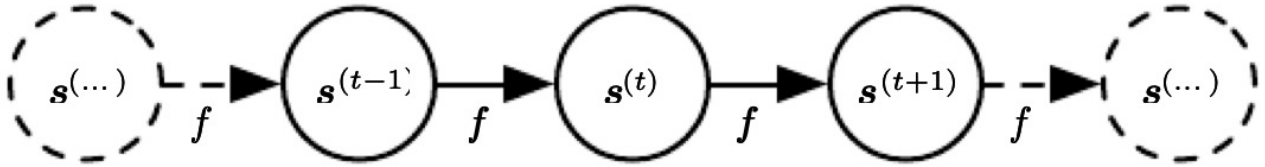
$$\boldsymbol{s}^{(t)} = f\left(\boldsymbol{s}^{(t-1)}; \boldsymbol{\theta}\right)$$



- Contoh:

$$\boldsymbol{s}^{(3)} = f\left(\boldsymbol{s}^{(2)}; \boldsymbol{\theta}\right)$$
$$\boldsymbol{s}^{(3)} = f\left(f\left(\boldsymbol{s}^{(1)}; \boldsymbol{\theta}\right); \boldsymbol{\theta}\right)$$

- Jejaring komputasi:

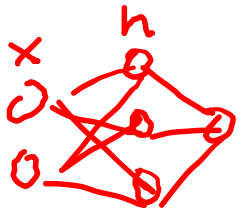


Deep Learning, p375

Figure 10.1: The classical dynamical system described by equation 10.1, illustrated as an unfolded computational graph. Each node represents the state at some time t and the function f maps the state at t to the state at $t+1$. The same parameters (the same value of $\boldsymbol{\theta}$ used to parametrize f) are used for all time steps.

CALVIN INSTITUTE OF TECHNOLOGY

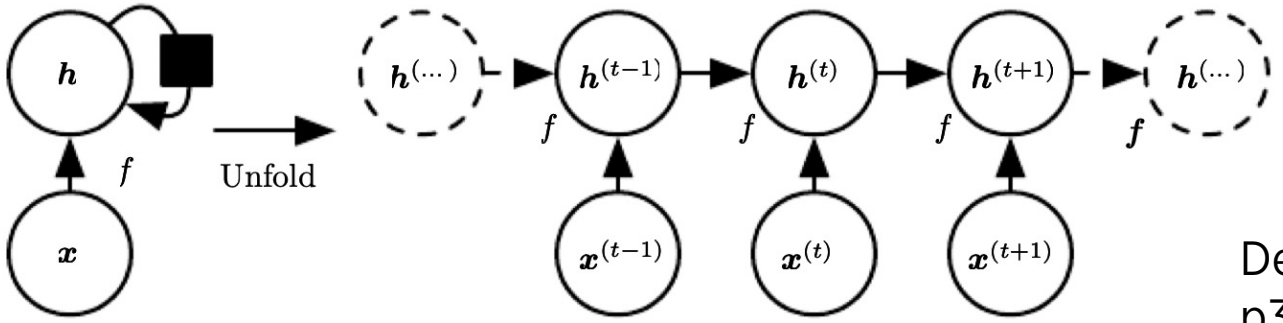
RNN



- Misalkan $\boldsymbol{h}^{(t)}$ adalah hidden unit sebuah neural network pada urutan t
- Dan $\boldsymbol{x}^{(t)}$ adalah external input terhadap neural network pada urutan t
- Dan $\boldsymbol{\theta}$ adalah parameter lainnya pada fungsi $f()$
- Recurrent Neural Network berarti:

$$\boldsymbol{h}^{(t)} = f\left(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta}\right)$$

- Jejaring komputasi:



Deep Learning, p376

Figure 10.2: A recurrent network with no outputs. This recurrent network just processes information from the input $\boldsymbol{x}$ by incorporating it into the state $\boldsymbol{h}$ that is passed forward through time. *(Left)*Circuit diagram. The black square indicates a delay of a single time step. *(Right)*The same network seen as an unfolded computational graph, where each node is now associated with one particular time instance.

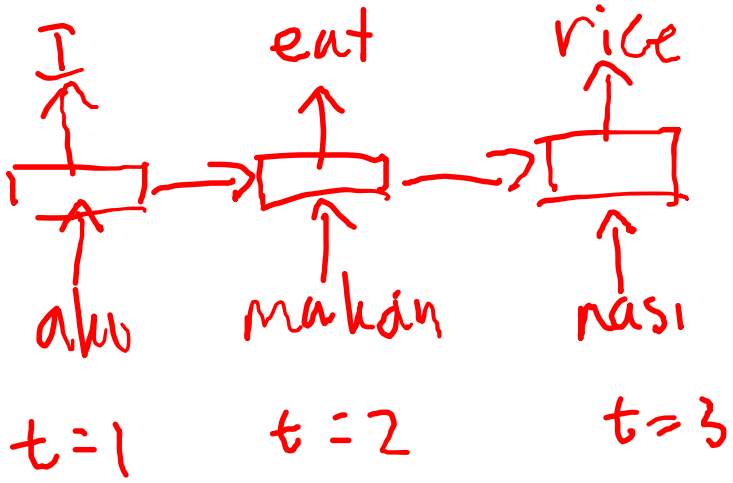
CALVIN INSTITUTE OF TECHNOLOGY

Kelebihan RNN

- Ukuran input selalu sama, tidak peduli kalimat sependek atau sepanjang apapun
- Menggunakan fungsi transisi dengan parameter yang sama

RNN Forward

- $a^{(t)} = b + Wh^{(t-1)} + Ux^{(t)}$
- $h^{(t)} = \tanh\left(a^{(t)}\right)$
- $o^{(t)} = c + Vh^{(t)}$
- $\hat{y}^{(t)} = softmax\left(o^{(t)}\right)$



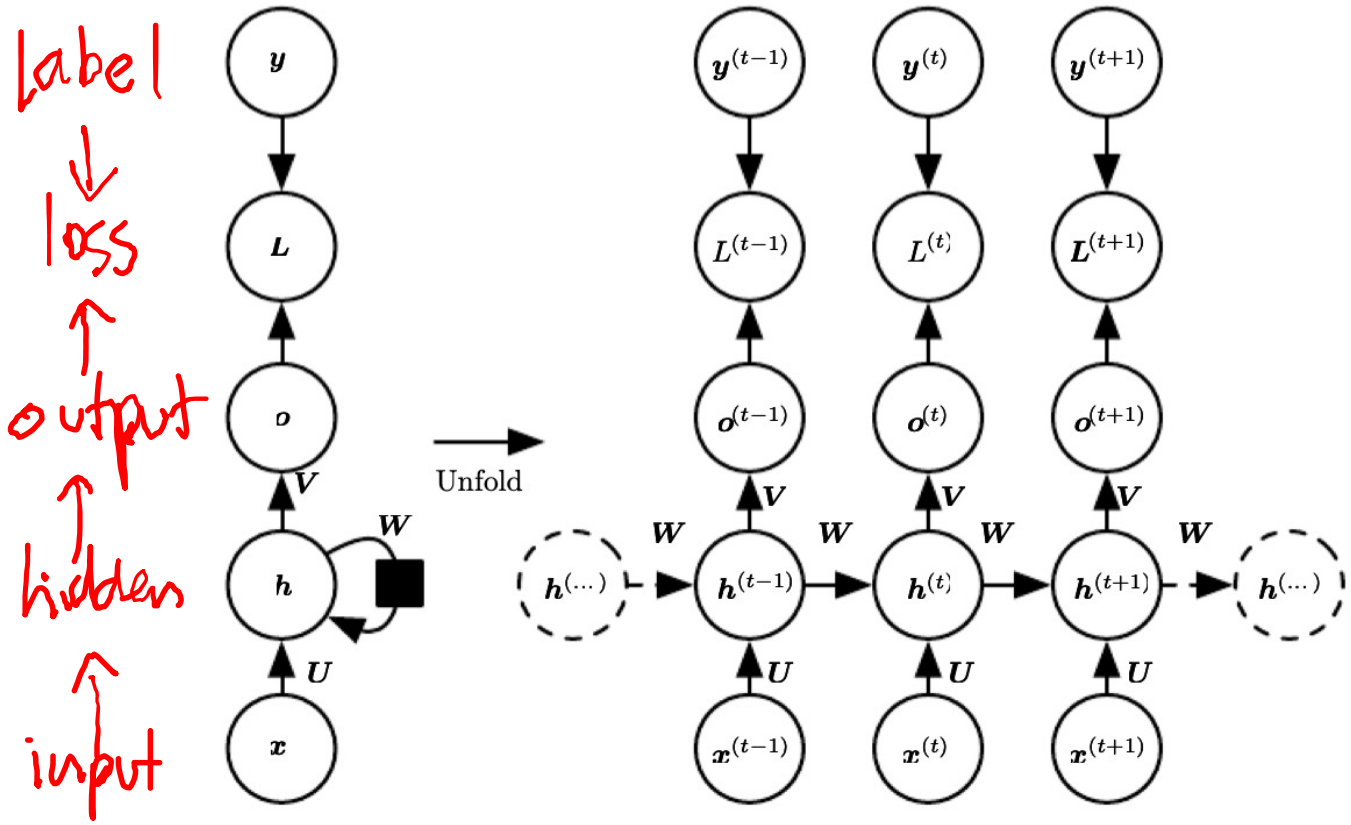
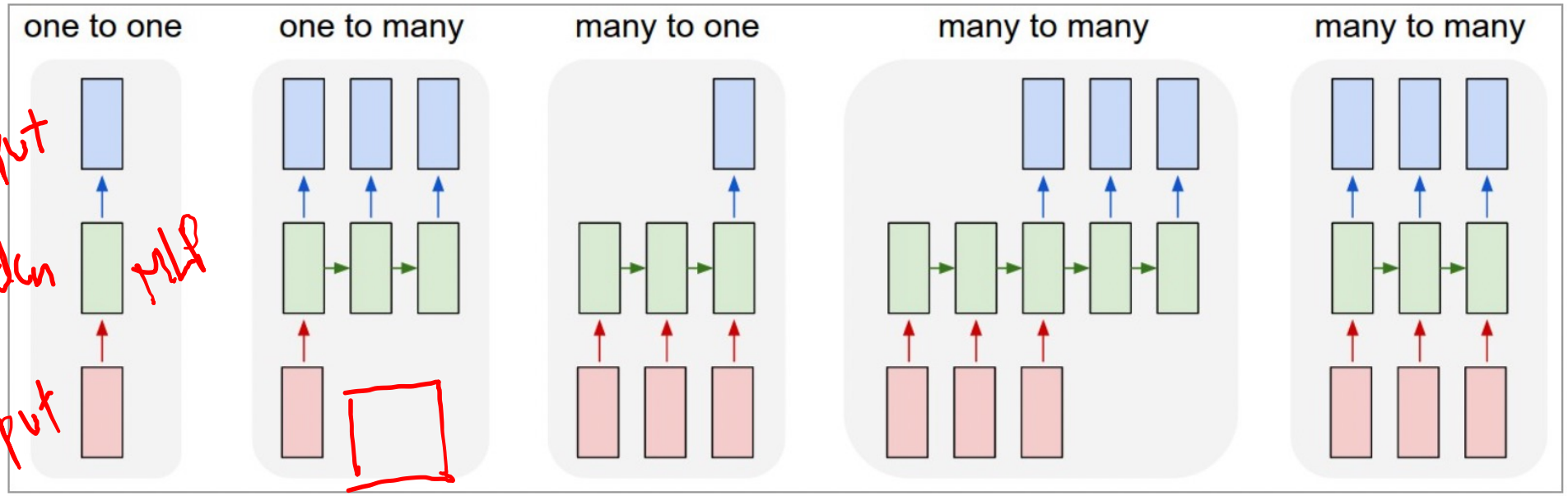


Figure 10.3: The computational graph to compute the training loss of a recurrent network that maps an input sequence of $\boldsymbol{x}$ values to a corresponding sequence of output $\boldsymbol{o}$ values. A loss L measures how far each $\boldsymbol{o}$ is from the corresponding training target $\boldsymbol{y}$. When using softmax outputs, we assume $\boldsymbol{o}$ is the unnormalized log probabilities. The loss L internally computes $\hat{\boldsymbol{y}} = \text{softmax}(\boldsymbol{o})$ and compares this to the target $\boldsymbol{y}$. The RNN has input to hidden connections parametrized by a weight matrix $\boldsymbol{U}$, hidden-to-hidden recurrent connections parametrized by a weight matrix $\boldsymbol{W}$, and hidden-to-output connections parametrized by a weight matrix $\boldsymbol{V}$. Equation 10.8 defines forward propagation in this model. *(Left)*The RNN and its loss drawn with recurrent connections. *(Right)*The same seen as an time-unfolded computational graph, where each node is now associated with one particular time instance.

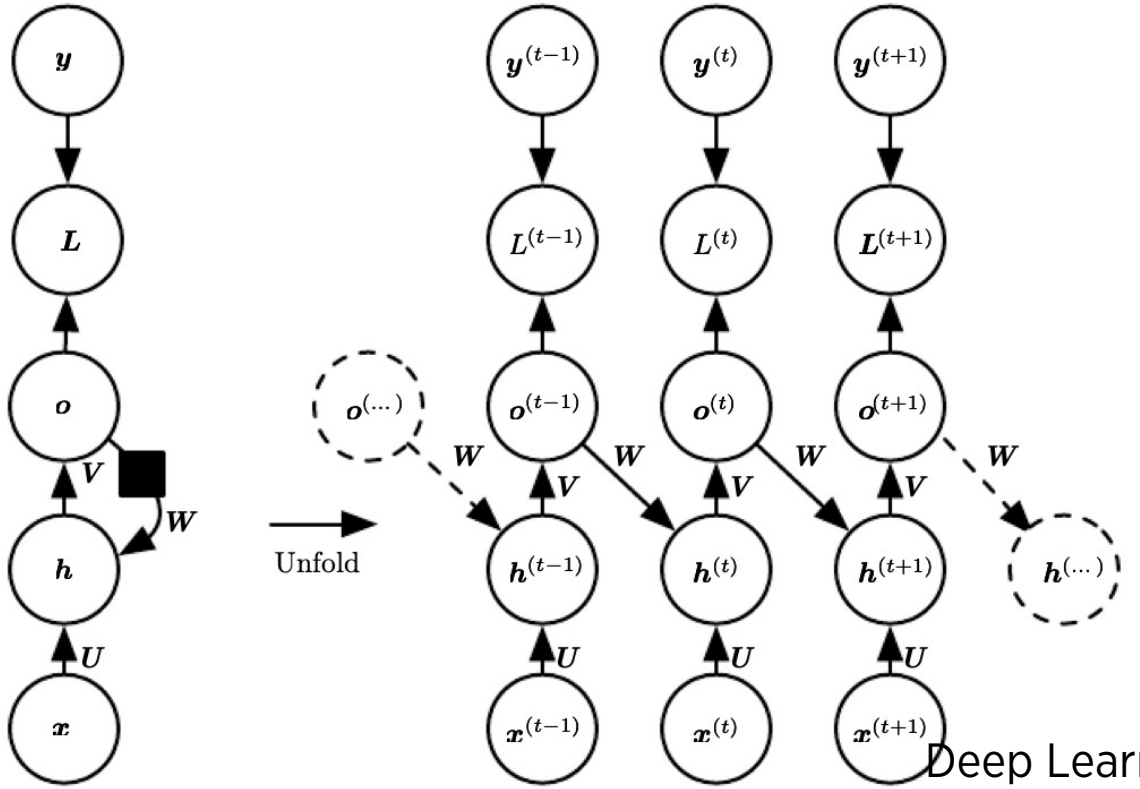
CALVIN INSTITUTE OF TECHNOLOGY

Ragam input-output RNN



Each rectangle is a vector and arrows represent functions (e.g. matrix multiply). Input vectors are in red, output vectors are in blue and green vectors hold the RNN's state (more on this soon). From left to right: **(1)** Vanilla mode of processing without RNN, from fixed-sized input to fixed-sized output (e.g. image classification). **(2)** Sequence output (e.g. image captioning takes an image and outputs a sentence of words). **(3)** Sequence input (e.g. sentiment analysis where a given sentence is classified as expressing positive or negative sentiment). **(4)** Sequence input and sequence output (e.g. Machine Translation: an RNN reads a sentence in English and then outputs a sentence in French). **(5)** Synced sequence input and output (e.g. video classification where we wish to label each frame of the video). Notice that in every case are no pre-specified constraints on the lengths sequences because the recurrent transformation (green) is fixed and can be applied as many times as we like.

Output to Input



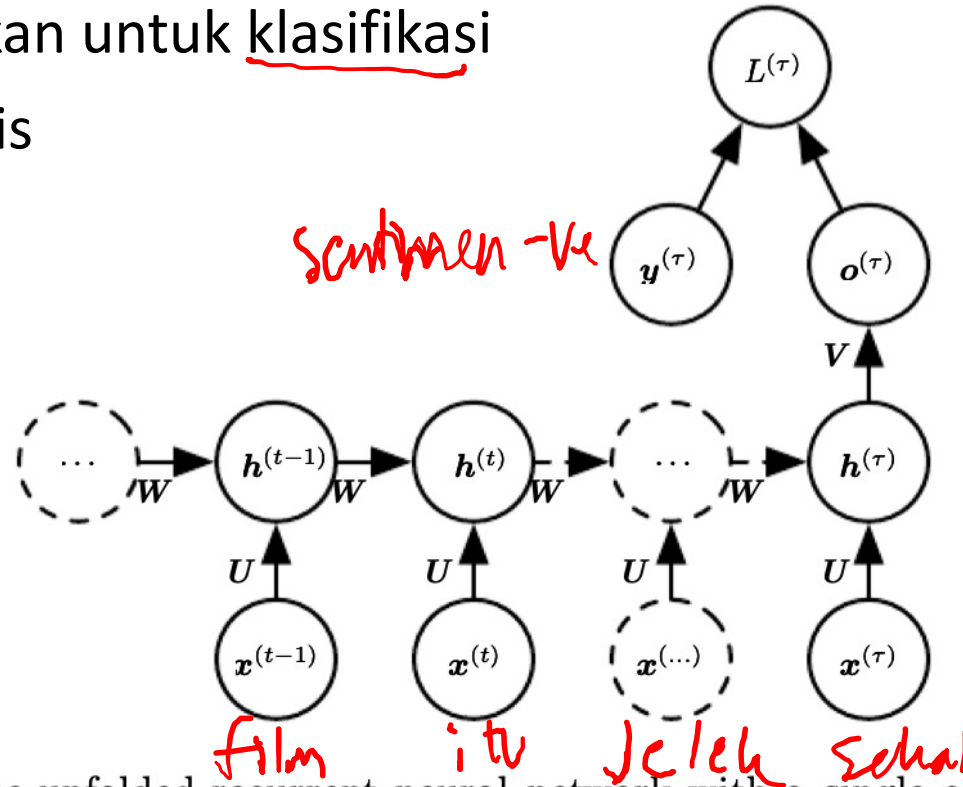
Deep Learning, p380

Figure 10.4: An RNN whose only recurrence is the feedback connection from the output to the hidden layer. At each time step t, the input is $\boldsymbol{x}_t$, the hidden layer activations are $\boldsymbol{h}^{(t)}$, the outputs are $\boldsymbol{o}^{(t)}$, the targets are $\boldsymbol{y}^{(t)}$ and the loss is $L^{(t)}$. *(Left)*Circuit diagram. *(Right)*Unfolded computational graph. Such an RNN is less powerful (can express a smaller set of functions) than those in the family represented by figure 10.3. The RNN in figure 10.3 can choose to put any information it wants about the past into its hidden representation $\boldsymbol{h}$ and transmit $\boldsymbol{h}$ to the future. The RNN in this figure is trained to put a specific output value into $\boldsymbol{o}$, and $\boldsymbol{o}$ is the only information it is allowed to send to the future. There are no direct connections from $\boldsymbol{h}$ going forward. The previous $\boldsymbol{h}$ is connected to the present only indirectly, via the predictions it was used to produce. Unless $\boldsymbol{o}$ is very high-dimensional and rich, it will usually lack important information from the past. This makes the RNN in this figure less powerful, but it may be easier to train because each time step can be trained in isolation from the others, allowing greater parallelization during training, as described in section 10.2.1.

CALVIN INSTITUTE OF TECHNOLOGY

Many to One RNN

- Biasa struktur seperti ini dapat digunakan untuk klasifikasi
- Misal: analisa sentimen, prediksi penulis

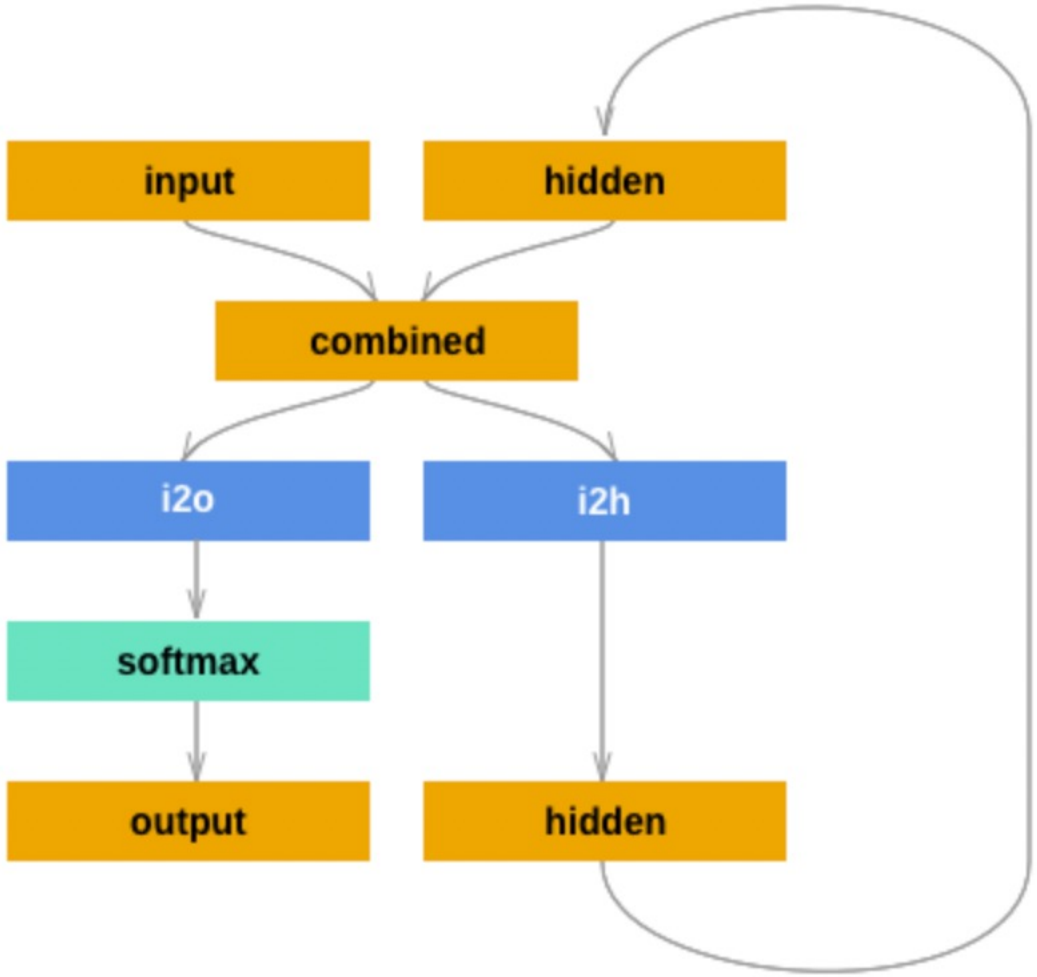


Deep Learning, p382

Figure 10.5: Time-unfolded recurrent neural network with a single output at the end of the sequence. Such a network can be used to summarize a sequence and produce a fixed-size representation used as input for further processing. There might be a target right at the end (as depicted here) or the gradient on the output $\boldsymbol{o}^{(t)}$ can be obtained by back-propagating from further downstream modules.

Aktivitas: RNN di PyTorch

- Buatlah RNN dengan struktur berikut
- Gunakan RNN ini untuk prediksi bahasa



Menghitung Gradien

Using this notation, the gradient on the remaining parameters is given by:

$$\nabla_{\boldsymbol{c}} L = \sum_t \left(\frac{\partial \boldsymbol{o}^{(t)}}{\partial \boldsymbol{c}} \right)^\top \nabla_{\boldsymbol{o}^{(t)}} L = \sum_t \nabla_{\boldsymbol{o}^{(t)}} L \tag{10.22}$$

$$\nabla_{\boldsymbol{b}} L = \sum_t \left(\frac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{b}^{(t)}} \right)^\top \nabla_{\boldsymbol{h}^{(t)}} L = \sum_t \operatorname{diag}\left(1 - \left(\boldsymbol{h}^{(t)} \right)^2 \right) \nabla_{\boldsymbol{h}^{(t)}} L \tag{10.23}$$

$$\nabla_{\boldsymbol{V}} L = \sum_t \sum_i \left(\frac{\partial L}{\partial o_i^{(t)}} \right) \nabla_{\boldsymbol{V}} o_i^{(t)} = \sum_t \left(\nabla_{\boldsymbol{o}^{(t)}} L \right) \boldsymbol{h}^{(t)\top} \tag{10.24}$$

$$\nabla_{\boldsymbol{W}} L = \sum_t \sum_i \left(\frac{\partial L}{\partial h_i^{(t)}} \right) \nabla_{\boldsymbol{W}^{(t)}} h_i^{(t)} \tag{10.25}$$

$$= \sum_t \operatorname{diag}\left(1 - \left(\boldsymbol{h}^{(t)} \right)^2 \right) \left(\nabla_{\boldsymbol{h}^{(t)}} L \right) \boldsymbol{h}^{(t-1)\top} \tag{10.26}$$

$$\nabla_{\boldsymbol{U}} L = \sum_t \sum_i \left(\frac{\partial L}{\partial h_i^{(t)}} \right) \nabla_{\boldsymbol{U}^{(t)}} h_i^{(t)} \tag{10.27}$$

$$= \sum_t \operatorname{diag}\left(1 - \left(\boldsymbol{h}^{(t)} \right)^2 \right) \left(\nabla_{\boldsymbol{h}^{(t)}} L \right) \boldsymbol{x}^{(t)\top} \tag{10.28}$$

We do not need to compute the gradient with respect to $\boldsymbol{x}^{(t)}$ for training because it does not have any parameters as ancestors in the computational graph defining the loss.

CALVIN INSTITUTE OF TECHNOLOGY

Bidirectional RNN

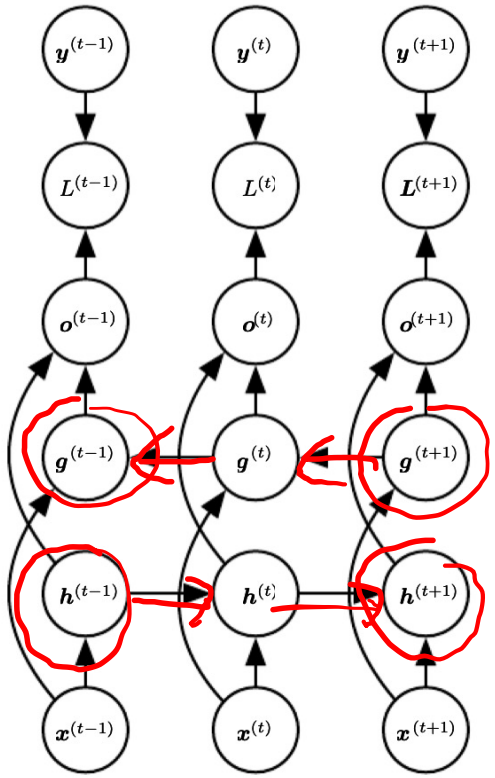


Figure 10.11: Computation of a typical bidirectional recurrent neural network, meant to learn to map input sequences $\boldsymbol{x}$ to target sequences $\boldsymbol{y}$, with loss $L^{(t)}$ at each step t. The $\boldsymbol{h}$ recurrence propagates information forward in time (towards the right) while the $\boldsymbol{g}$ recurrence propagates information backward in time (towards the left). Thus at each point t, the output units $\boldsymbol{o}^{(t)}$ can benefit from a relevant summary of the past in its $\boldsymbol{h}^{(t)}$ input and from a relevant summary of the future in its $\boldsymbol{g}^{(t)}$ input.

CALVIN INSTITUTE OF TECHNOLOGY

Encoder-Decoder

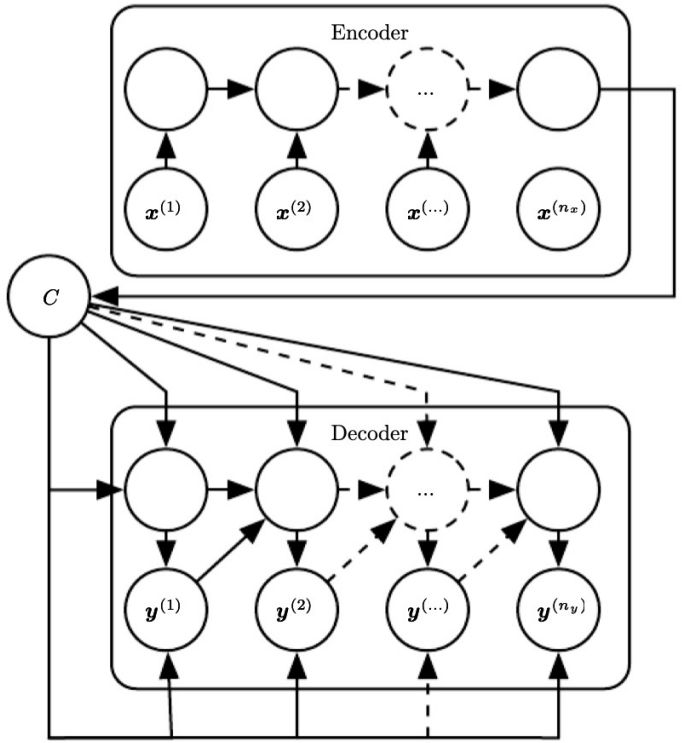
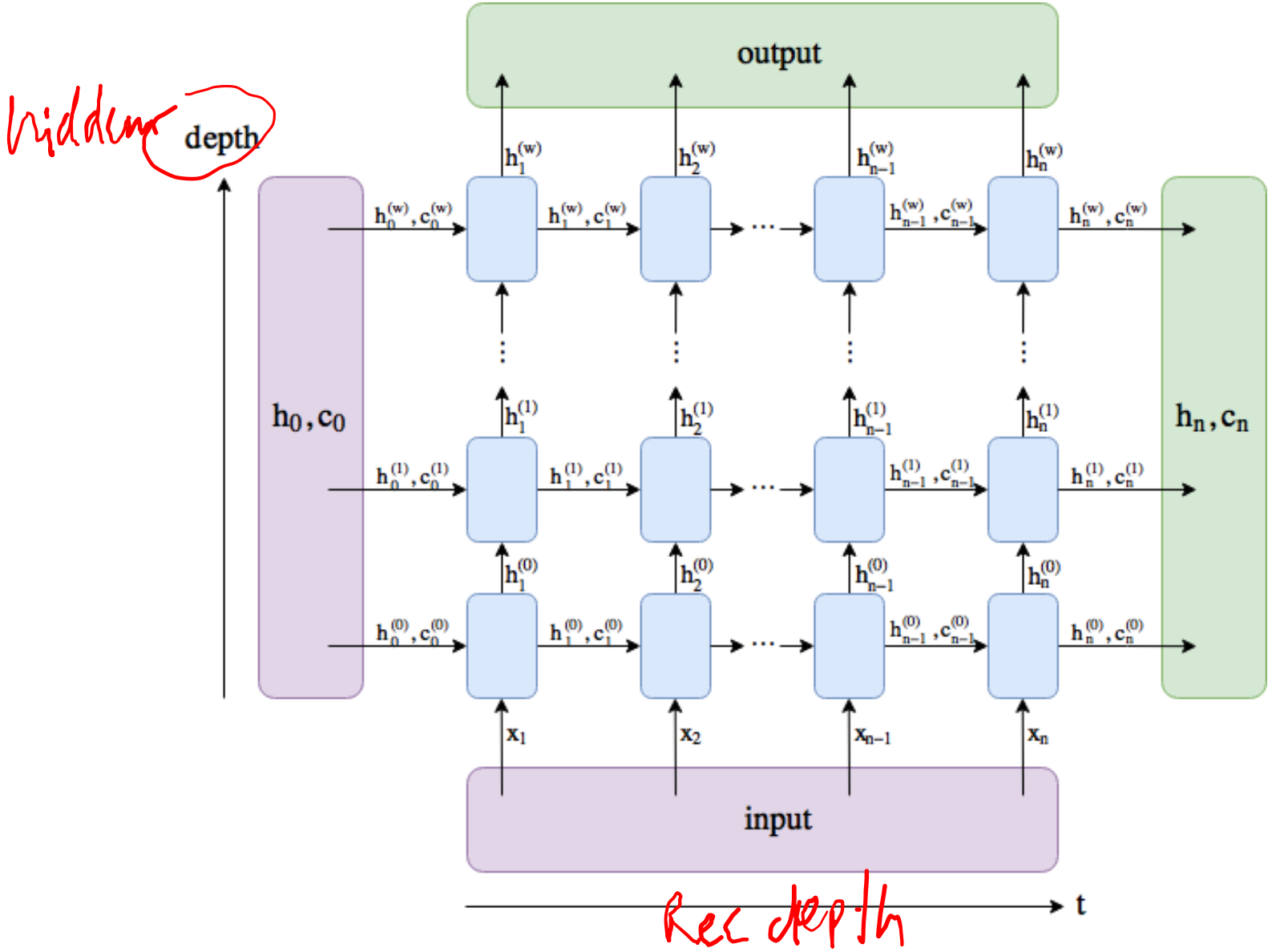


Figure 10.12: Example of an encoder-decoder or sequence-to-sequence RNN architecture, for learning to generate an output sequence $(\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(n_y)})$ given an input sequence $(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(n_x)})$. It is composed of an encoder RNN that reads the input sequence and a decoder RNN that generates the output sequence (or computes the probability of a given output sequence). The final hidden state of the encoder RNN is used to compute a generally fixed-size context variable C which represents a semantic summary of the input sequence and is given as input to the decoder RNN.

CALVIN INSTITUTE OF TECHNOLOGY

Hidden Depth vs Recurrent Depth



Variasi Deep Recurrent Network

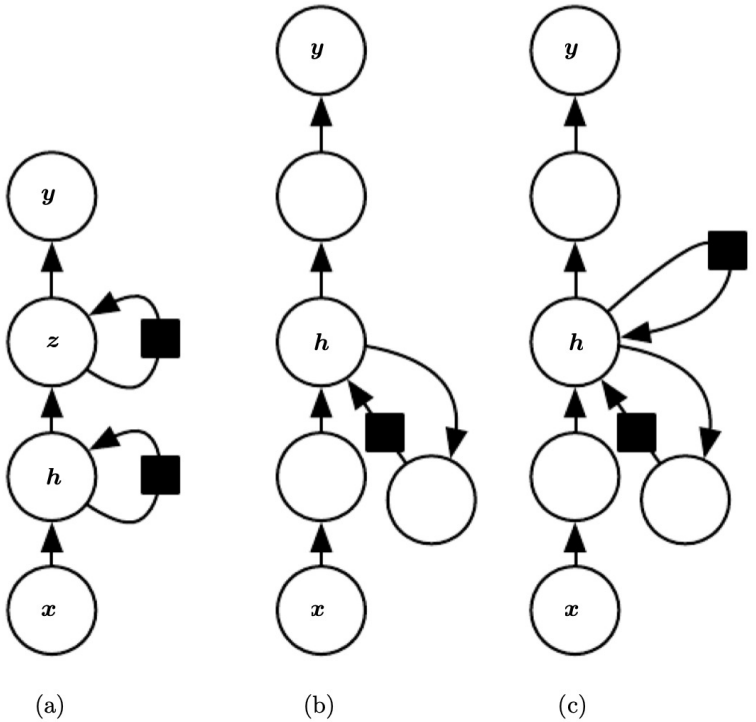


Figure 10.13: A recurrent neural network can be made deep in many ways (Pascanu *et al.*, 2014a). *(a)*The hidden recurrent state can be broken down into groups organized hierarchically. *(b)*Deeper computation (e.g., an MLP) can be introduced in the input-to-hidden, hidden-to-hidden and hidden-to-output parts. This may lengthen the shortest path linking different time steps. *(c)*The path-lengthening effect can be mitigated by introducing skip connections.

Recursive Network

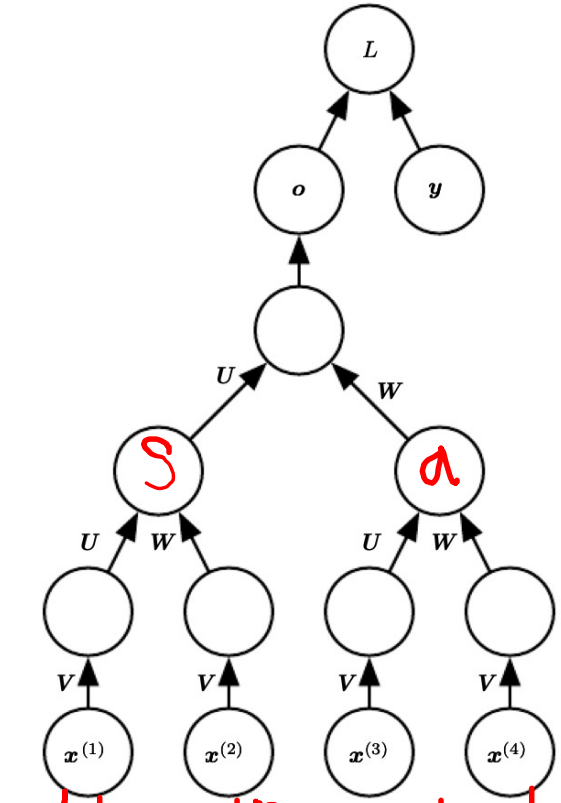
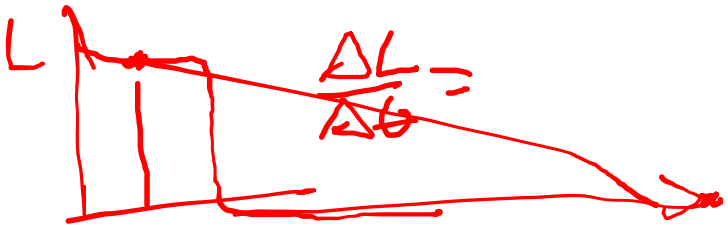


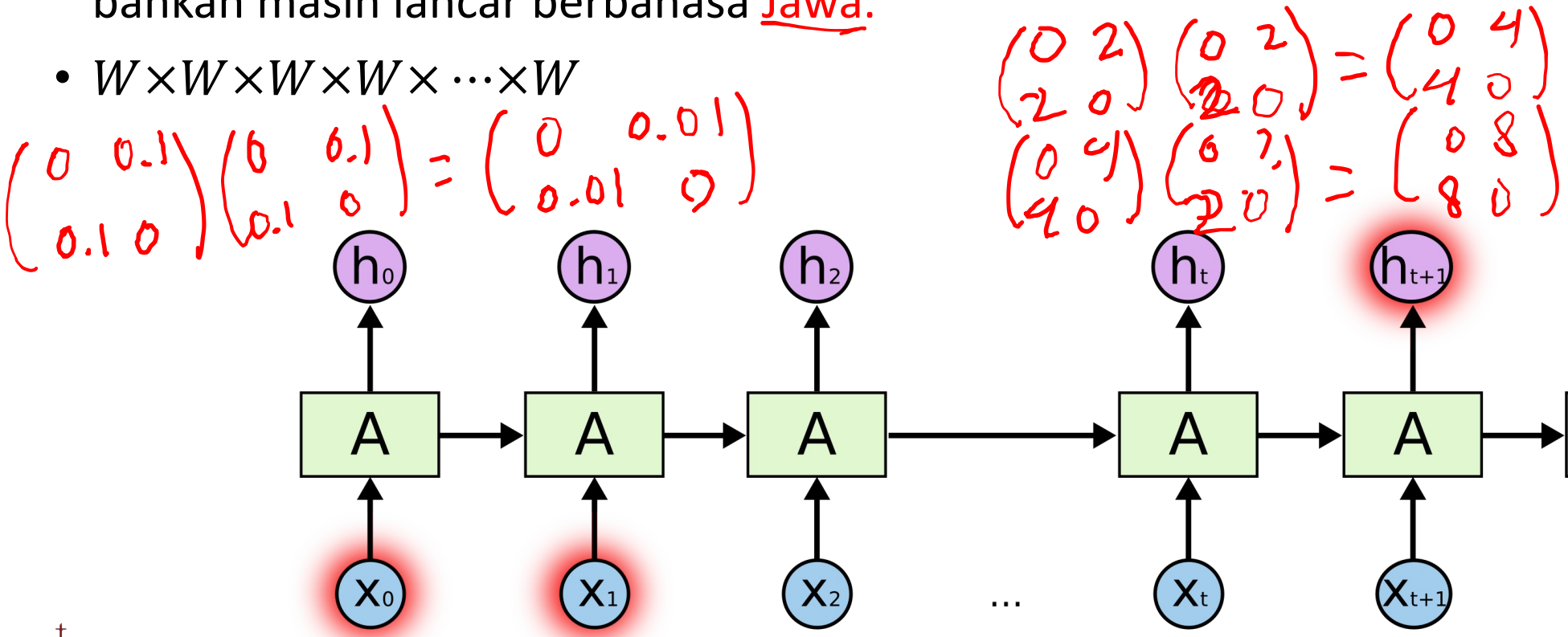
Figure 10.14: A recursive network has a computational graph that generalizes that of the recurrent network from a chain to a tree. A variable-size sequence $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(t)}$ can be mapped to a fixed-size representation (the output $\boldsymbol{o}$), with a fixed set of parameters (the weight matrices $\boldsymbol{U}$, $\boldsymbol{V}$, $\boldsymbol{W}$). The figure illustrates a supervised learning case in which some target $\boldsymbol{y}$ is provided which is associated with the whole sequence.

LSTM & GRU

Vanishing/Exploding Gradient



- Hochreiter (1991) [German] and Bengio, et al. (1994) menemukan bahwa RNN tidak dapat mempelajari long term dependencies
- Contoh: “saya dibesarkan di Surabaya. Selama itu, saya … Setelah itu, saya … Saya bahkan masih lancar berbahasa Jawa.”
- $W \times W \times W \times W \times \cdots \times W$



LSTM (Long Short-Term Memory)

Forget Gate

- $f_i^{(t)} = \sigma\left(b_i^f + \sum_j U_{i,j}^f x_j^{(t)} + \sum_j W_{i,j}^f h_j^{(t-1)}\right)$

Input Gate

- $g_i^{(t)} = \sigma\left(b_i^g + \sum_j U_{i,j}^g x_j^{(t)} + \sum_j W_{i,j}^g h_j^{(t-1)}\right)$

LSTM Cell

- $s_i^{(t)} = f_i^{(t)} s_i^{(t-1)} + g_i^{(t)} \sigma\left(b_i + \sum_j U_{i,j} x_j^{(t)} + \sum_j W_{i,j} h_j^{(t-1)}\right)$

Output Gate

- $q_i^{(t)} = \sigma\left(b_i^o + \sum_j U_{i,j}^o x_j^{(t)} + \sum_j W_{i,j}^o h_j^{(t-1)}\right)$
- $h_i^{(t)} = \tanh\left(s_i^{(t)}\right) q_i^{(t)}$

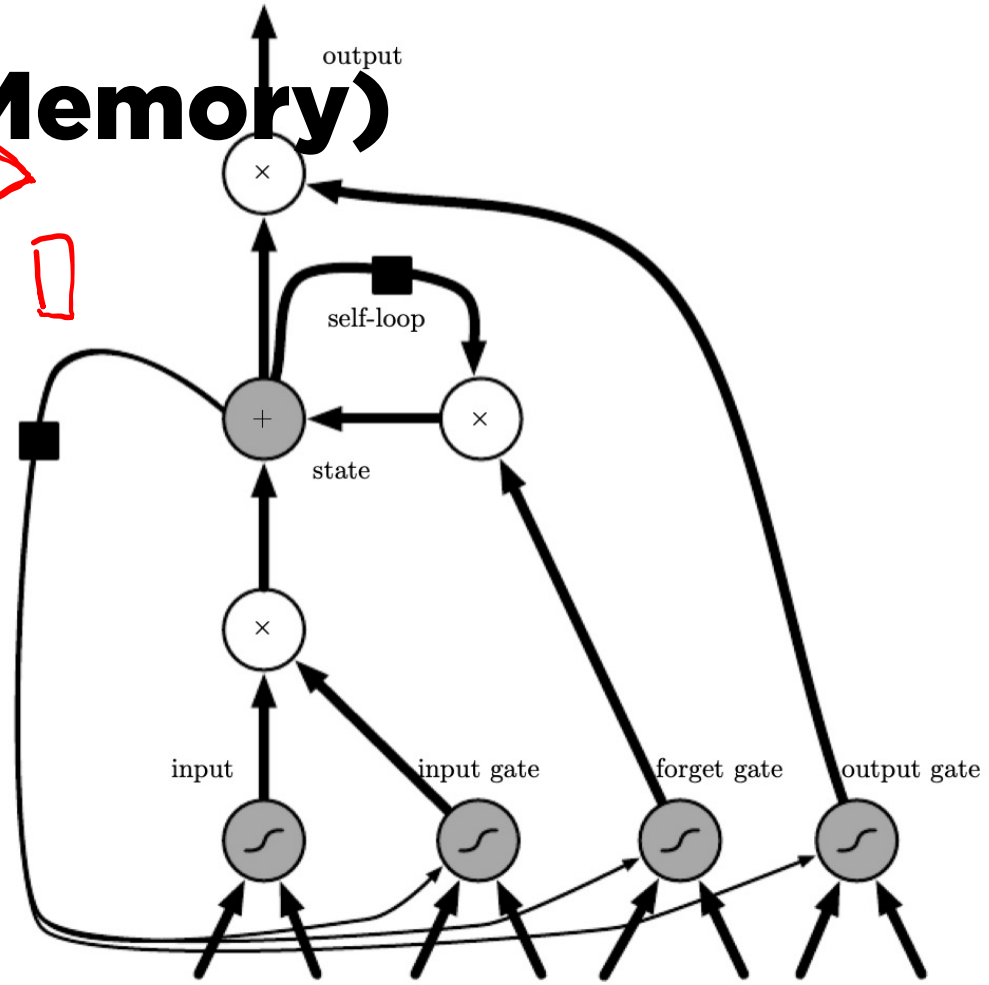


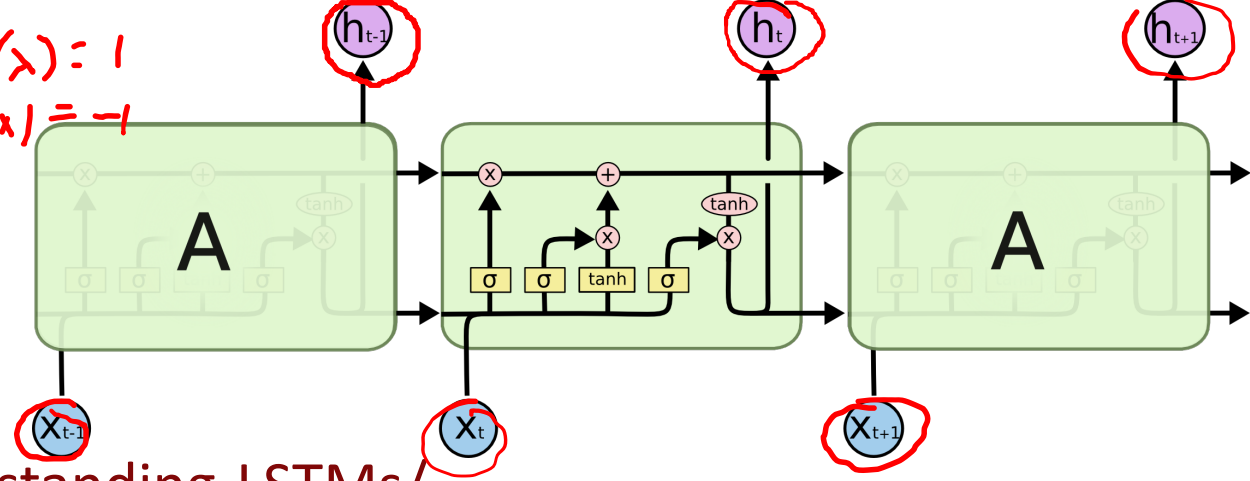
Figure 10.16: Block diagram of the LSTM recurrent network "cell." Cells are connected recurrently to each other, replacing the usual hidden units of ordinary recurrent networks. An input feature is computed with a regular artificial neuron unit. Its value can be accumulated into the state if the sigmoidal input gate allows it. The state unit has a linear self-loop whose weight is controlled by the forget gate. The output of the cell can be shut off by the output gate. All the gating units have a sigmoid nonlinearity, while the input unit can have any squashing nonlinearity. The state unit can also be used as an extra input to the gating units. The black square indicates a delay of a single time step.

Skema

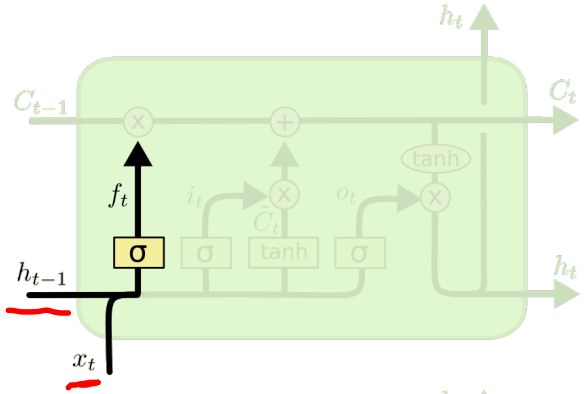
$x = 725 \rightarrow \tanh(x) = 1$

$x = -9.53 \rightarrow \tanh(x) = -1$

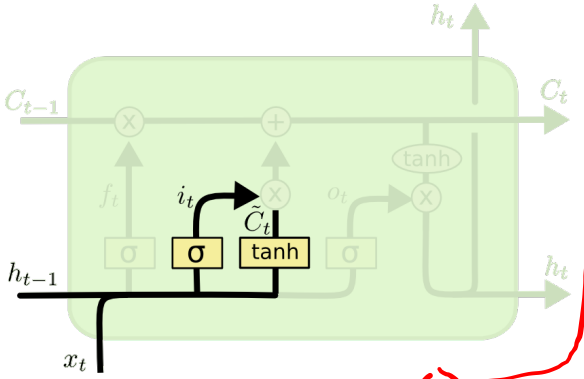
$-1 < \tanh(x) < 1$



https://colah.github.io/posts/2015-08-Understanding-LSTMs/

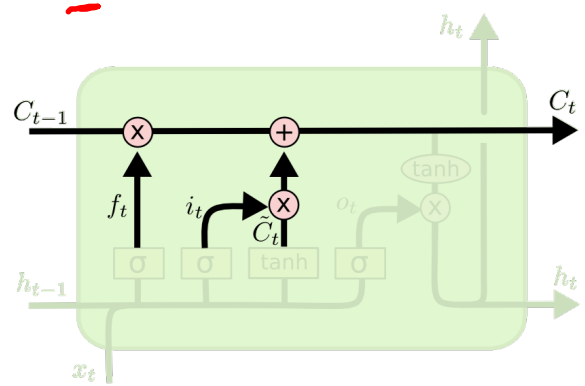


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

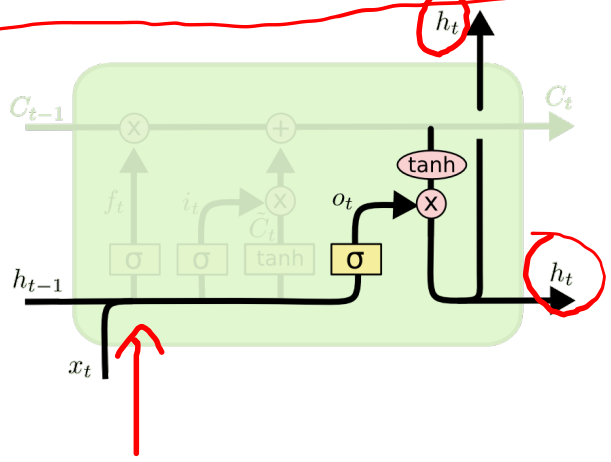


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$



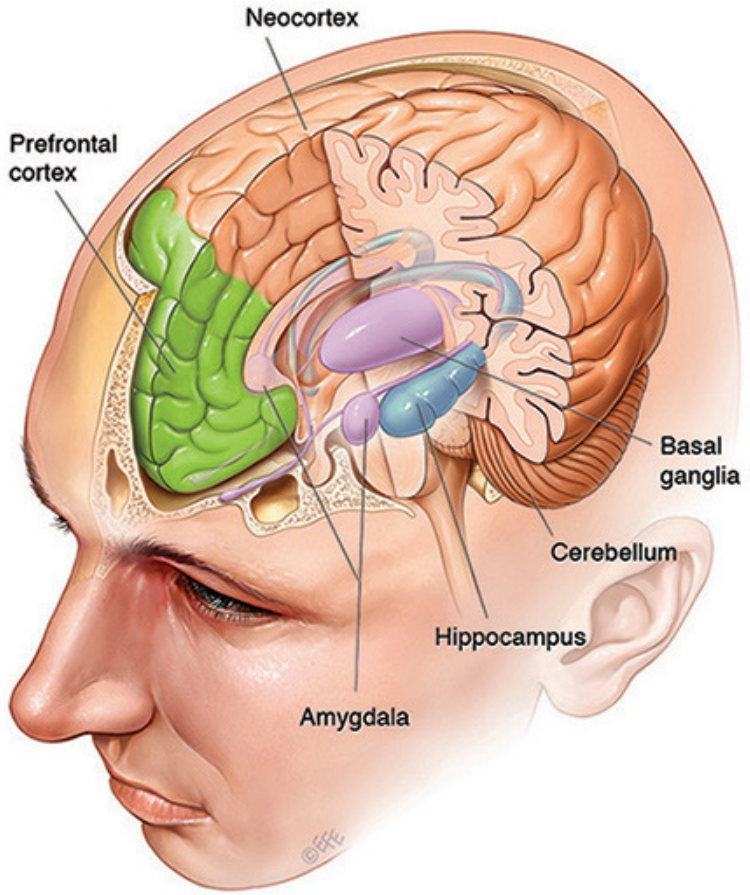
$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

CALVIN INSTITUTE OF TECHNOLOGY

Hippocampus

- In 1953, Henry Molaison had his hippocampus surgically removed to treat his epilepsy. His epilepsy was cured. However, after the surgery he was only able to form episodic memories that lasted a matter of minutes (completely unable to permanently store new information). As a result, Molaison's memory became mostly limited to events that occurred years before his surgery, in the distant past. He was, however, still able to improve his performance on various motor tasks, even though he had no memory of ever encountering or practising them. This indicated that although the hippocampus is crucial for laying down memories, it is not the site of permanent memory storage and isn't needed for motor memories (basal ganglia and cerebellum).



https://www.telegraph.co.uk/culture/10047050/Henry-Molaison-The-incredible-story-of-the-man-with-no-memory.html

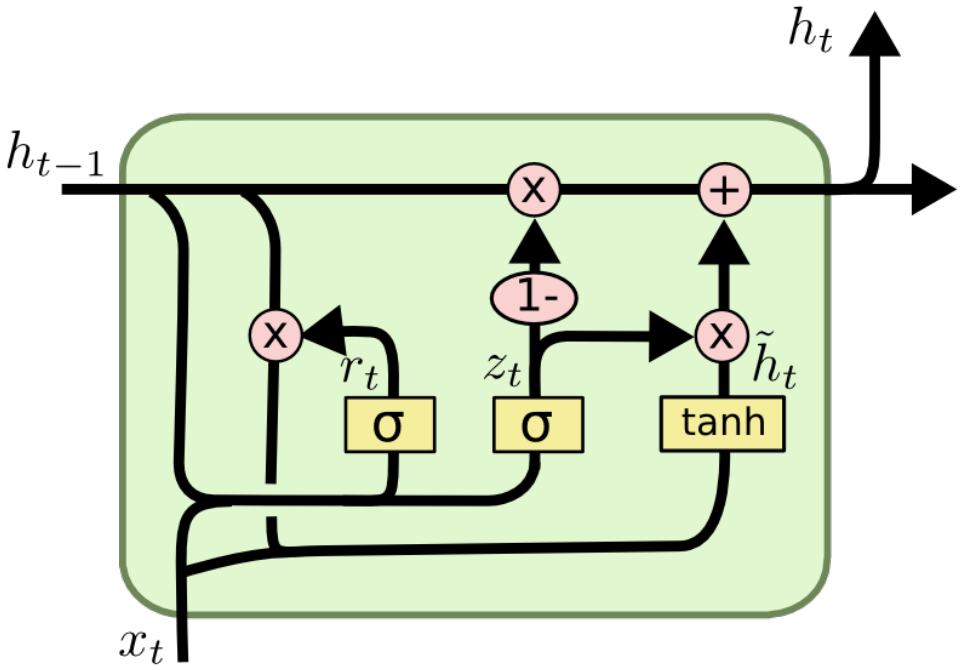
GRU (Gated Recurrent Units)

- *Update Gate*
- $u_i^{(t)} = \sigma\left(b_i^u + \sum_j U_{i,j}^u x_j^{(t)} + \sum_j W_{i,j}^u h_j^{(t)}\right)$

- *Reset Gate*
- $r_i^{(t)} = \sigma\left(b_i^r + \sum_j U_{i,j}^r x_j^{(t)} + \sum_j W_{i,j}^r h_j^{(t)}\right)$

- *Update Equation*
- $h_i^{(t)} = u_i^{(t-1)} h_i^{(t-1)} + \left(1 - u_i^{(t-1)}\right) \sigma\left(b_i + \sum_j U_{i,j} x_j^{(t-1)} + \sum_j W_{i,j} r_j^{(t-1)} h_j^{(t-1)}\right)$

Skema



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

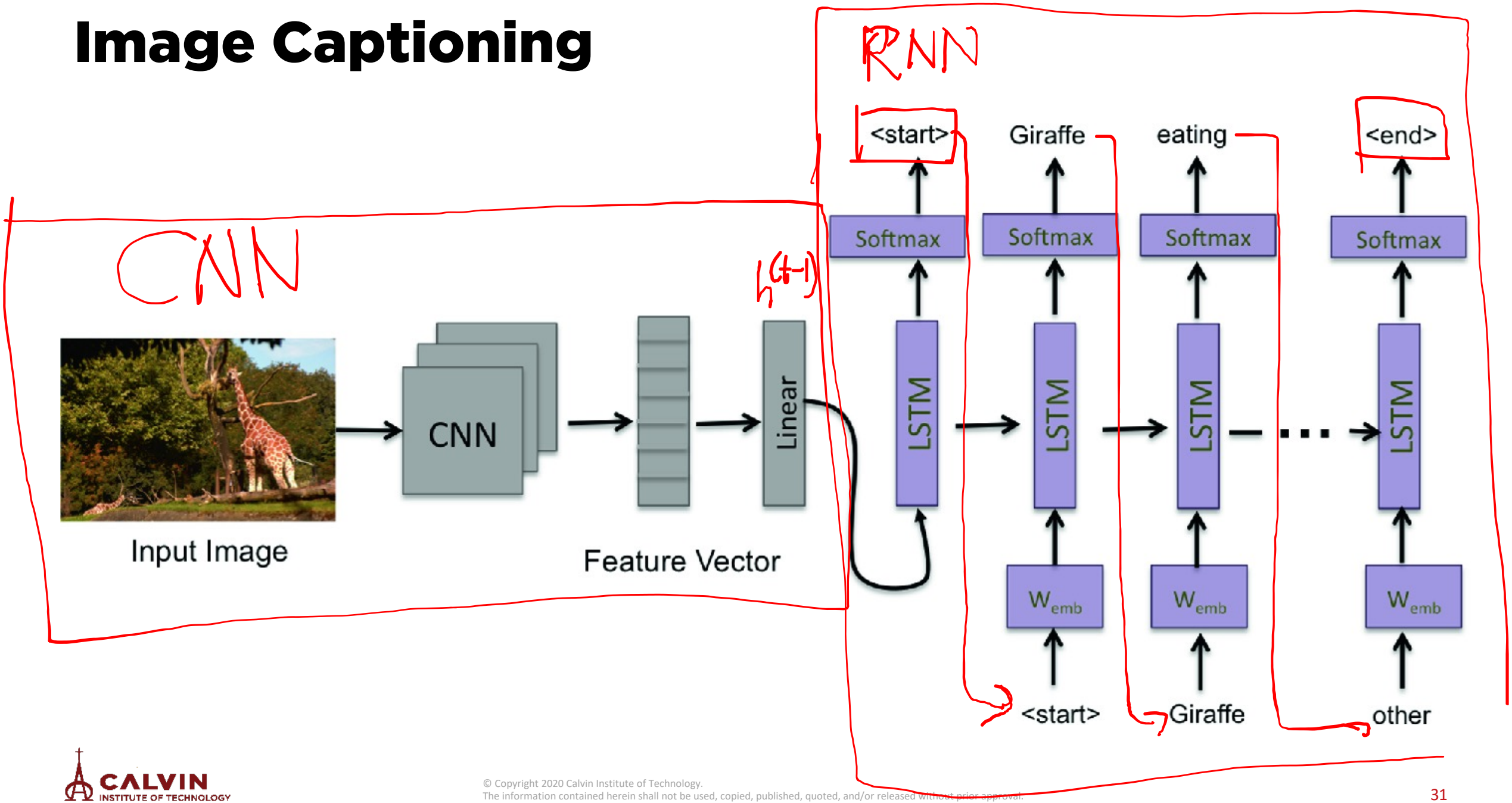
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Aktivitas: Time-Series Menggunakan Sequence Model

- Gunakan sequence model untuk memprediksi pola time-series pada 2 dataset yang sudah disediakan:
  - Temperature Melbourne
  - Sunspots
- Gunaan 80% awal series sebagai data latih dan 20% akhir series sebagai data tes
- Bandingkan RNN, LSTM, GRU:
  - MAE
  - Latensi train dan test

CALVIN INSTITUTE OF TECHNOLOGY

Image Captioning



Summary

- Masing-masing sebutkan konsep utama yang diingat

Tuhan Memberkati

God's People for God's Glory
CALVIN
INSTITUTE OF TECHNOLOGY