Deep Learning :
Recurrent Neural Networks (RNN)

Ridho Rahmadi

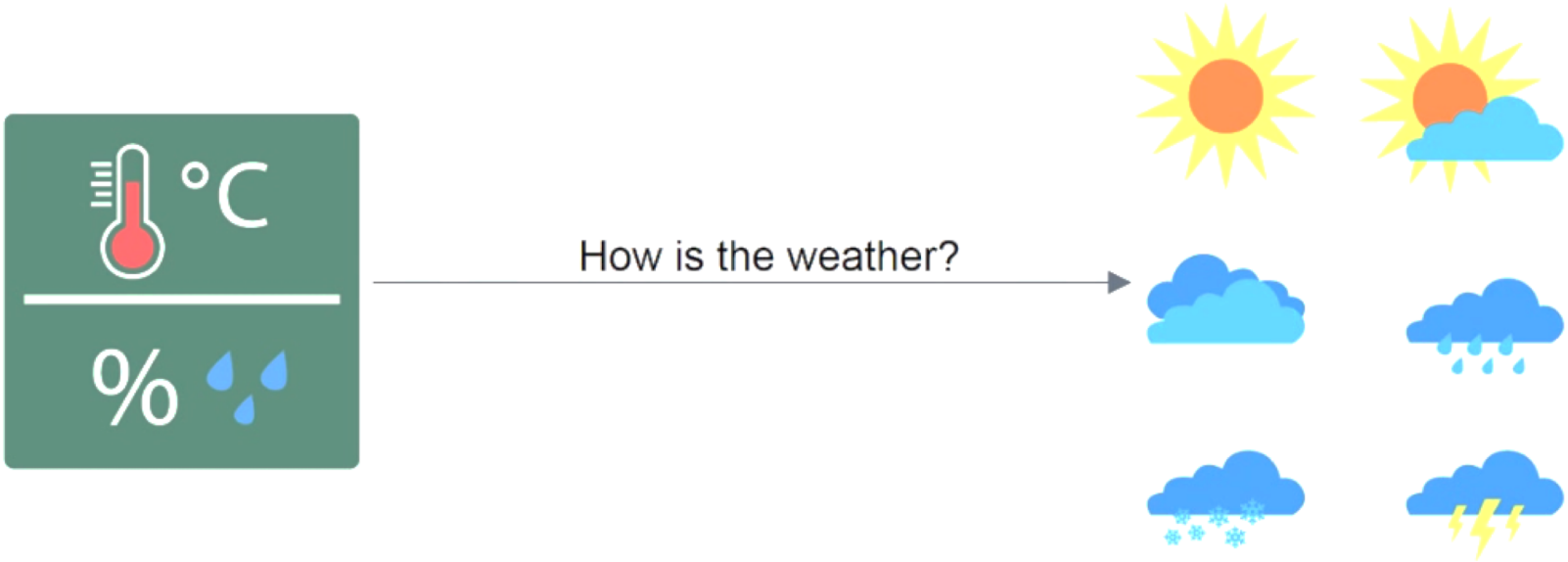
The Sequential Problem

Bagian 1

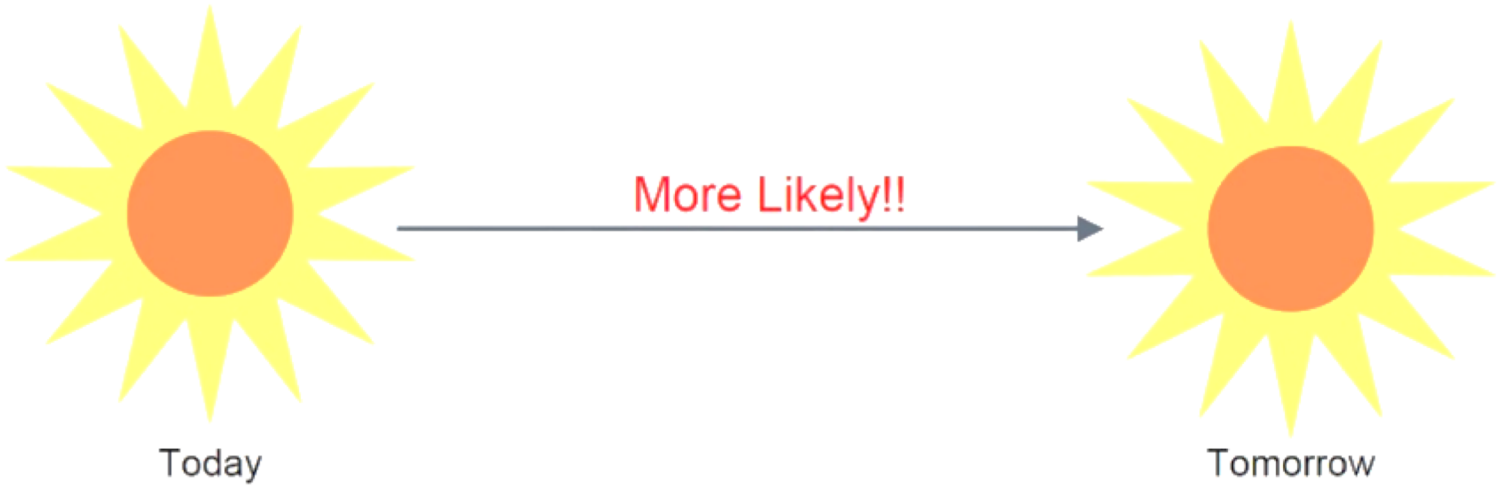
Sequential Data

- Sequential Data – data points with dependencies
- Time Series – observations over time
- Many Important datasets are sequential
- Not handled well by traditional Neural Networks

The Sequential Problem



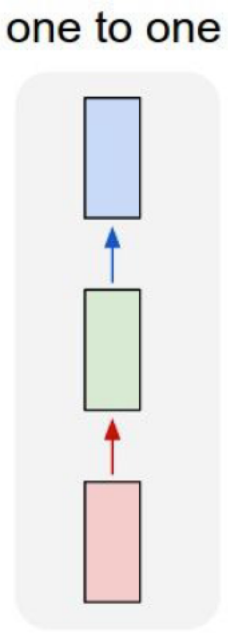
The Sequential Problem



The RNN Model

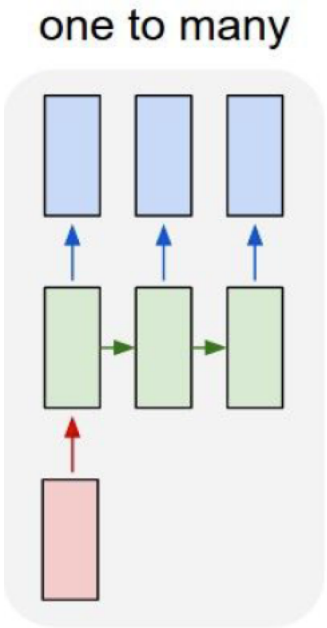
Bagian 2

Process sequence



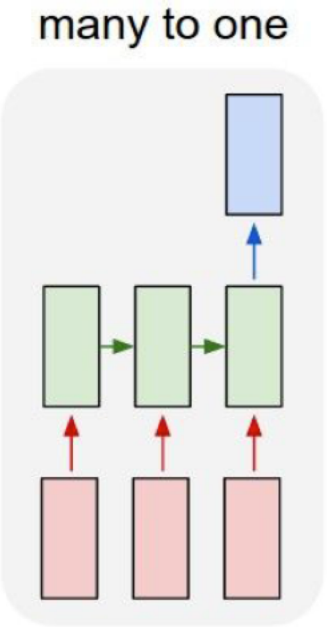
- Traditional neural network, atau disebut juga sebagai *vanilla neural networks*
- *Input → pemrosesan → output*

Process sequence



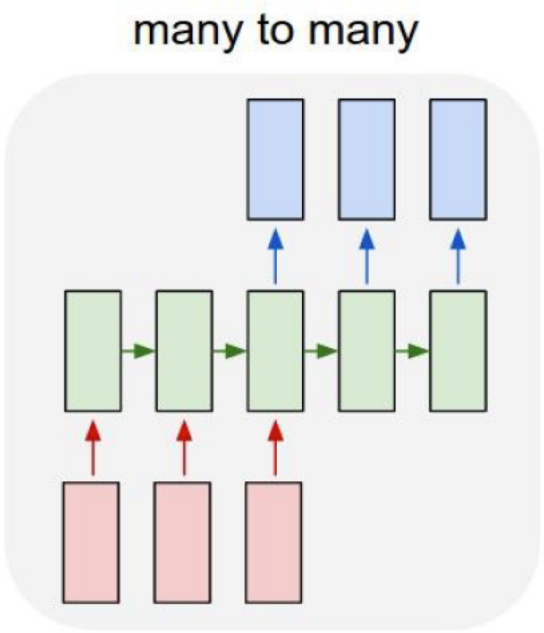
- Model one to many; satu input dengan banyak output
- Contoh penggunaan untuk memodelkan *image captioning*
  - Input: image → output: rangkaian kata-kata

Process sequence



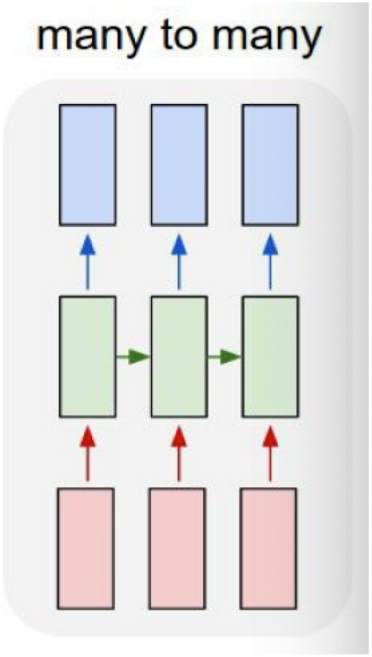
- Model many (input) to one (output)
- Contoh kasus: *sentiment classification*
  - Input: rangkaian kata-kata → output sentimen

Process sequence



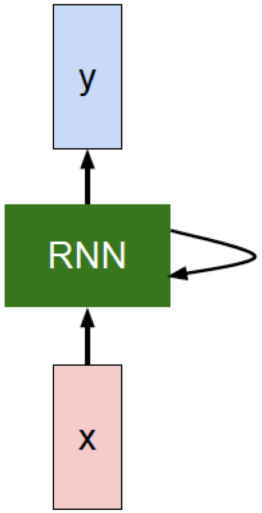
- Model many to many
- Dapat digunakan untuk *machine translation*
  - Input: rangkaian kata-kata → output: rangkaian kata-kata

Process sequence



- Model many to many lainnya
- Contoh kasus: *video classification on frame* atau *name entity recognition*

Model Recurrent



- Sebuah *Recurrent Neural Network* (RNN) cocok digunakan untuk data sekuensial atau process sequence
- RNN menggunakan *state* untuk merekam output pada setiap tahap yang dilalui. Dengan kata lain, state berfungsi sebagai memori bagi RNN

RNN

We can process a sequence of vectors **x** by applying a **recurrence formula** at every time step:

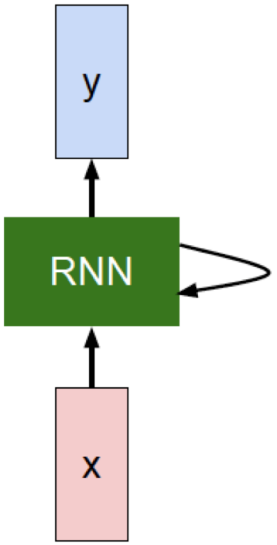
$$h_t = f_W(h_{t-1}, x_t)$$

new state

old state

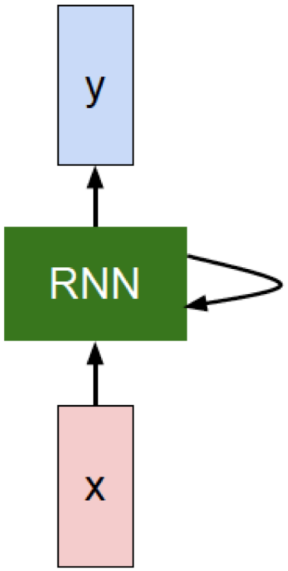
input vector at some time step

some function with parameters W



Note that the same function and
the same set of parameters are used at every time step

Vanilla RNN



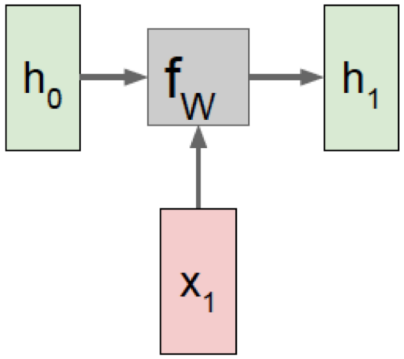
$$h_t = f_W(h_{t-1}, x_t)$$

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

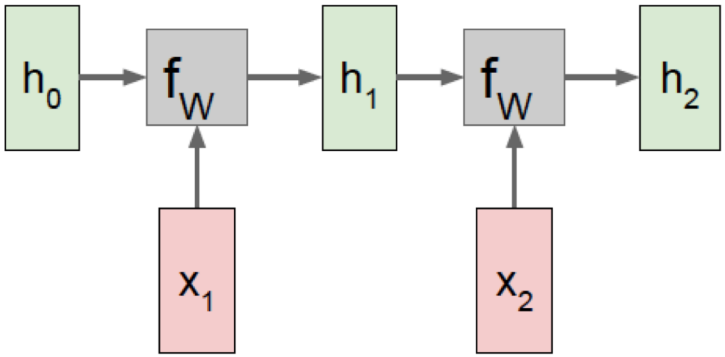
$$y_t = W_{hy}h_t$$

The state consists of a single “hidden” vector **h**.

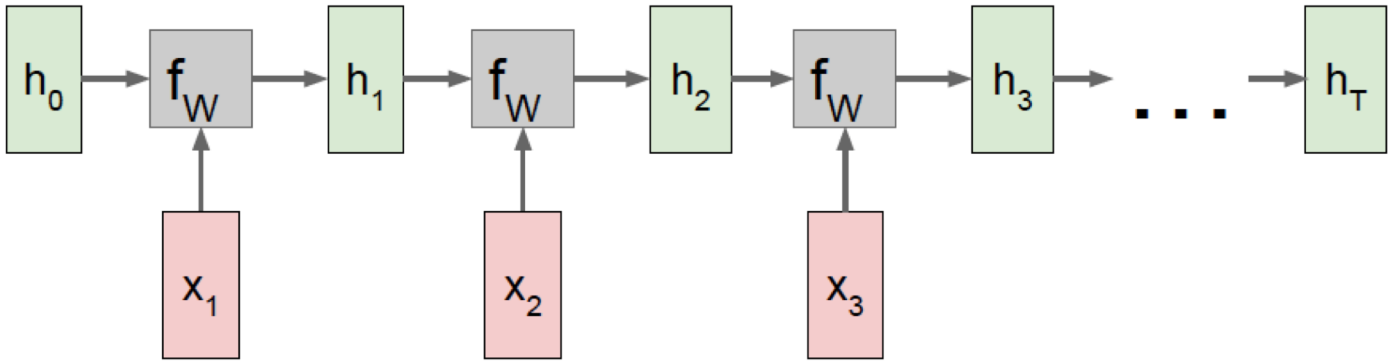
RNN



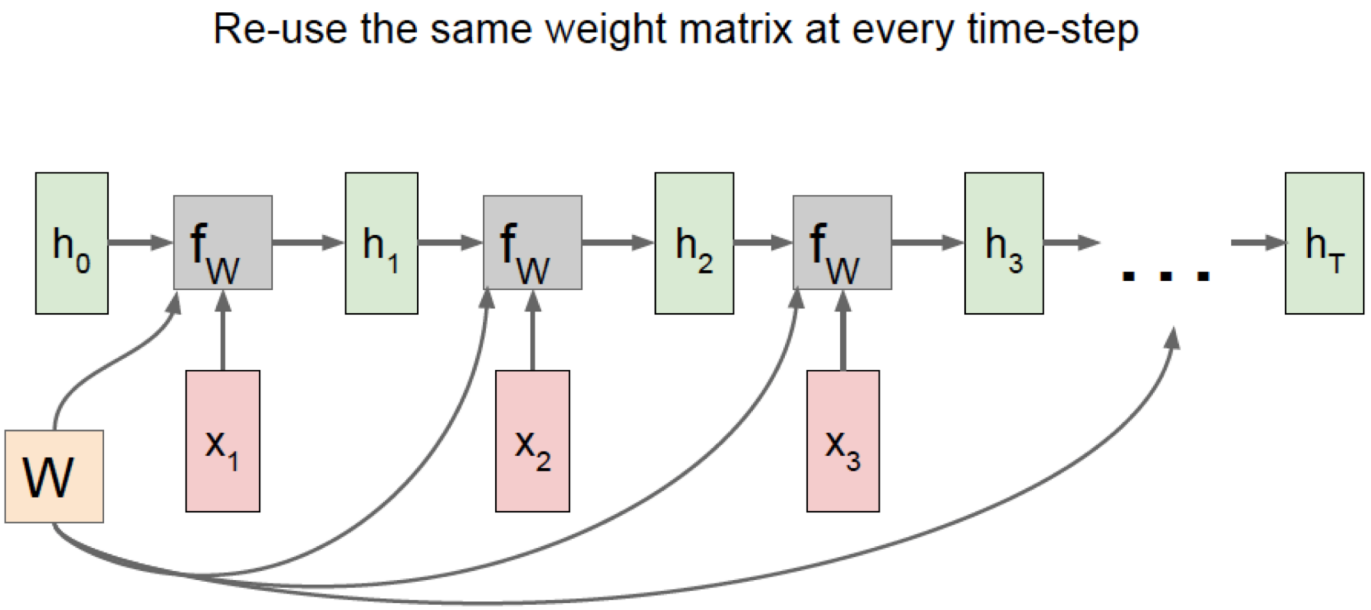
RNN



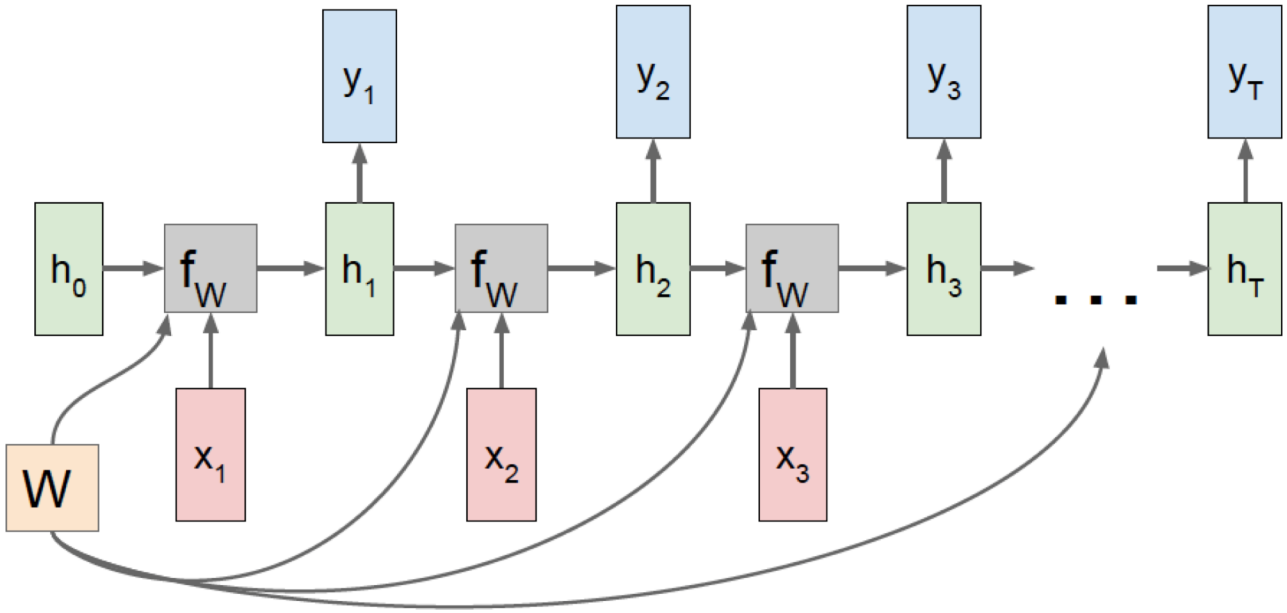
RNN



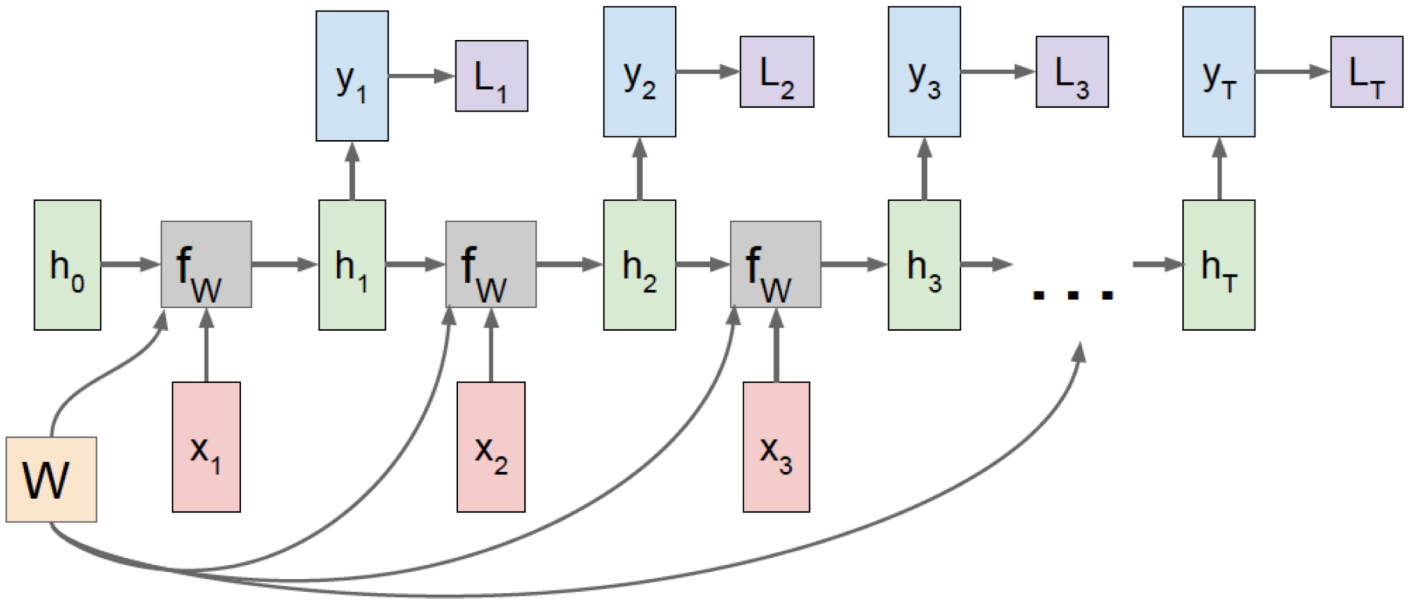
RNN


Re-use the same weight matrix at every time-step
h0
fW
h1
fW
h2
fW
h3
. . .
hT
W
x1
x2
x3

RNN

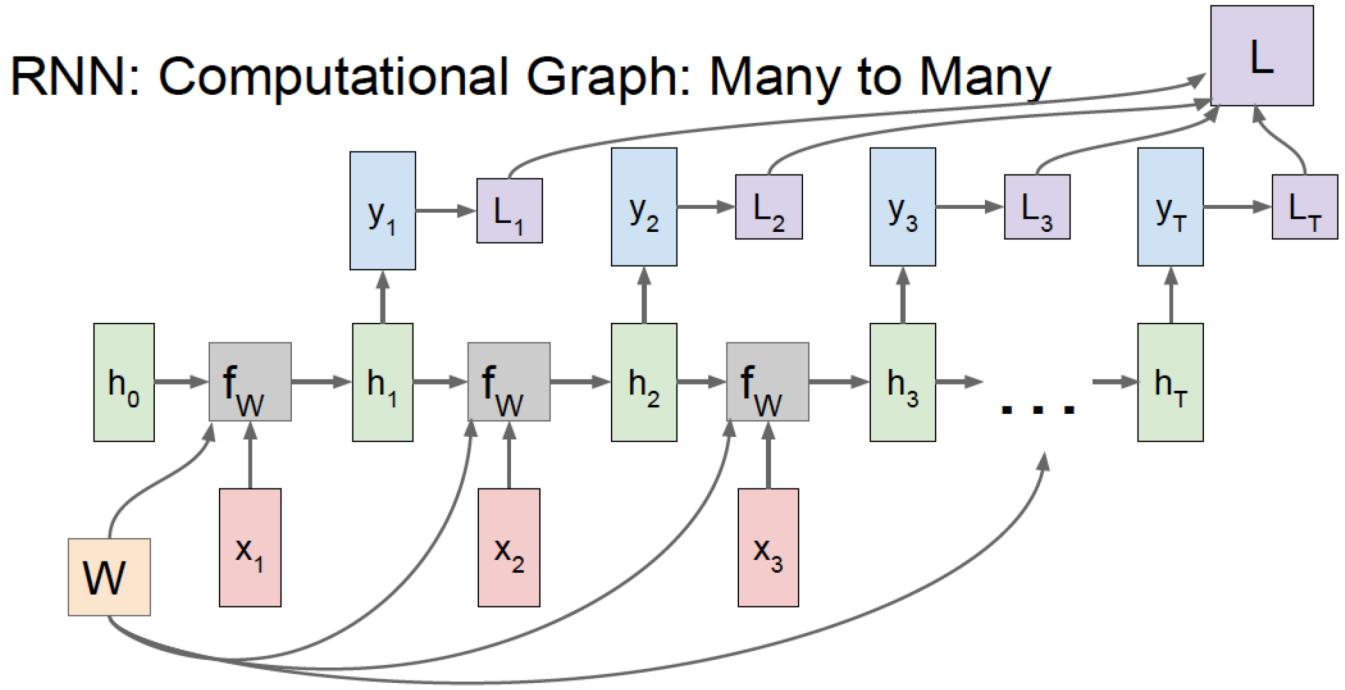


RNN



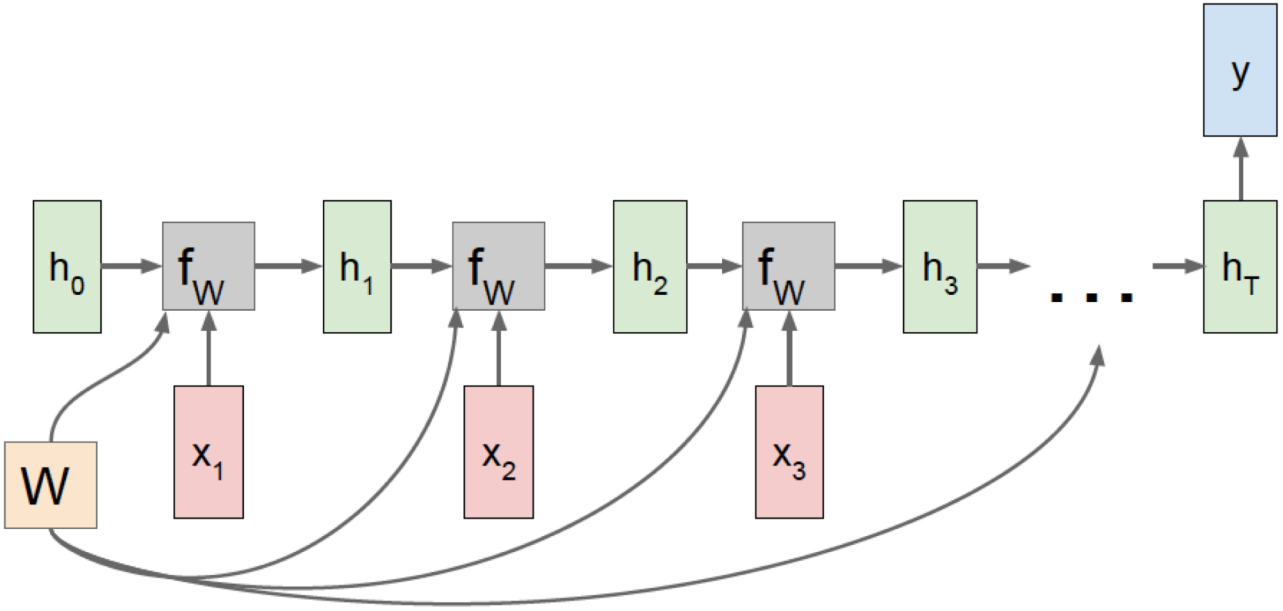
Menghitung loss function L, untuk mengevaluasi model

RNN

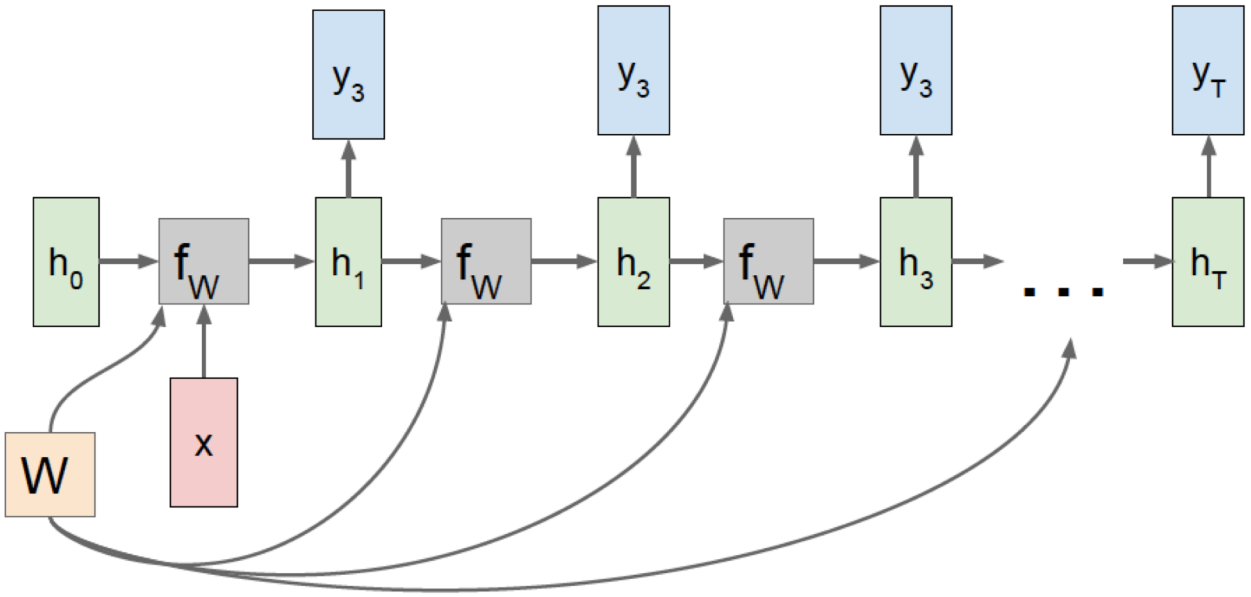


Menghitung loss function L, untuk mengevaluasi model

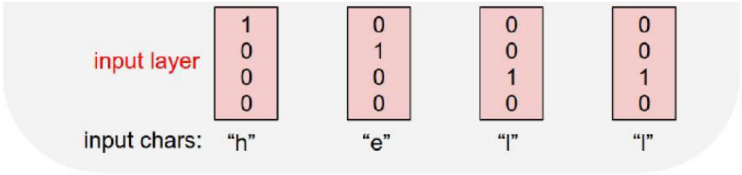
RNN: Many to One



RNN: One to Many



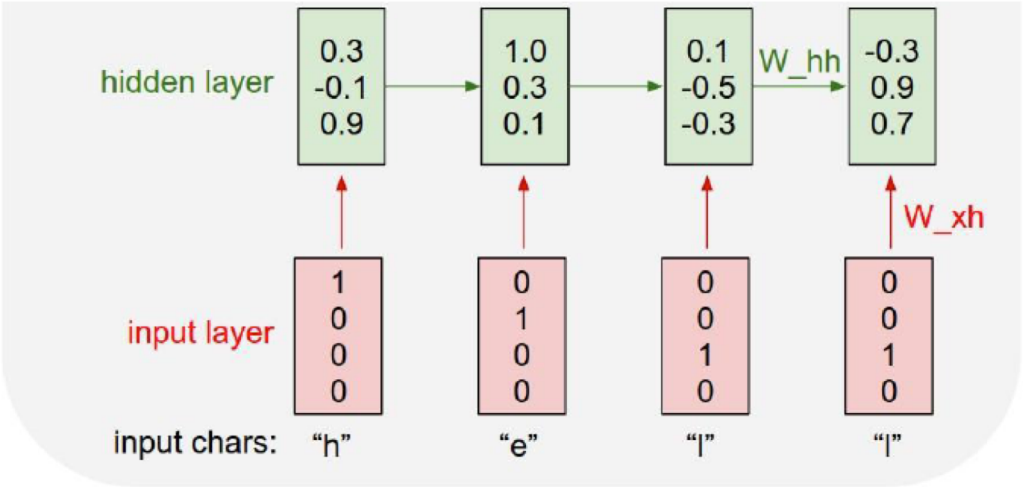
Ex: character-level language model



- Vocabulary: [h, e, l, o]
- Example training sequence: “hello”

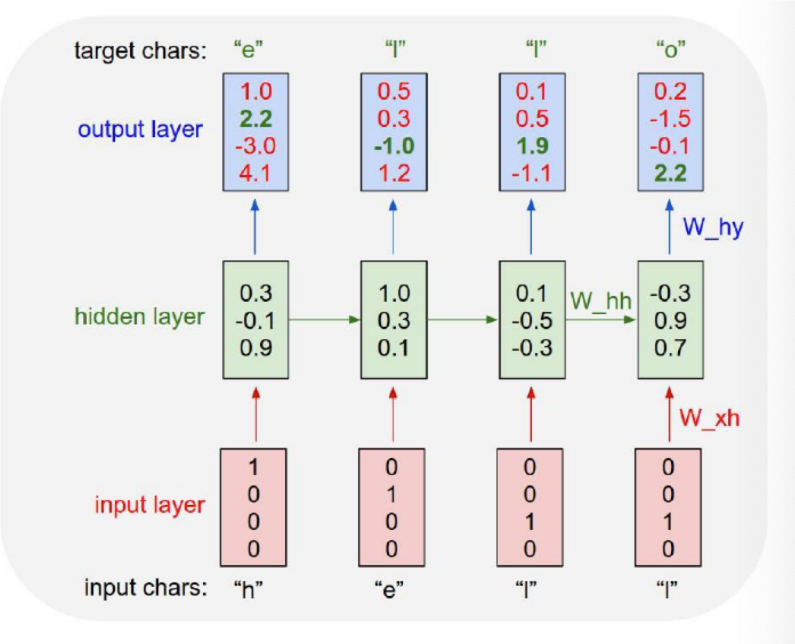
Ex: character-level language model

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



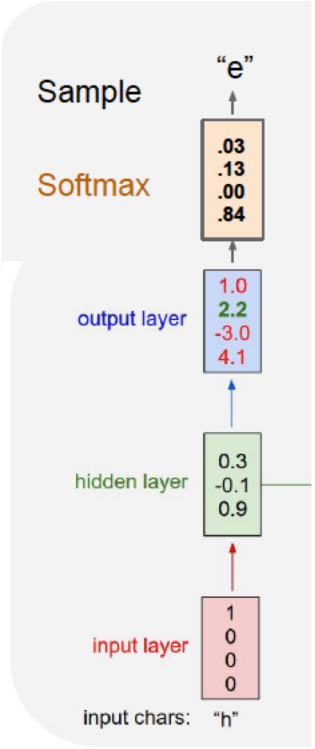
- Vocabulary: [h, e, l, o]
- Example training sequence: “hello”

Ex: character-level language model



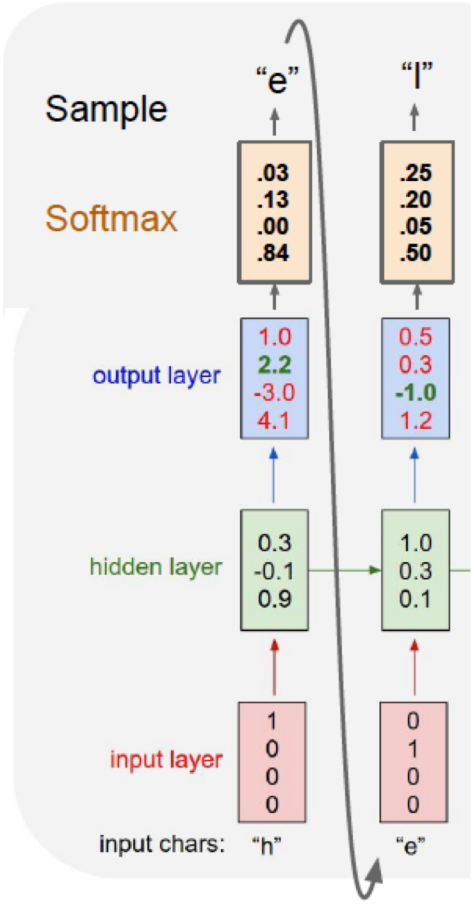
- Vocabulary: [h, e, l, o]
- Example training sequence: "hello"

Ex: character-level language model



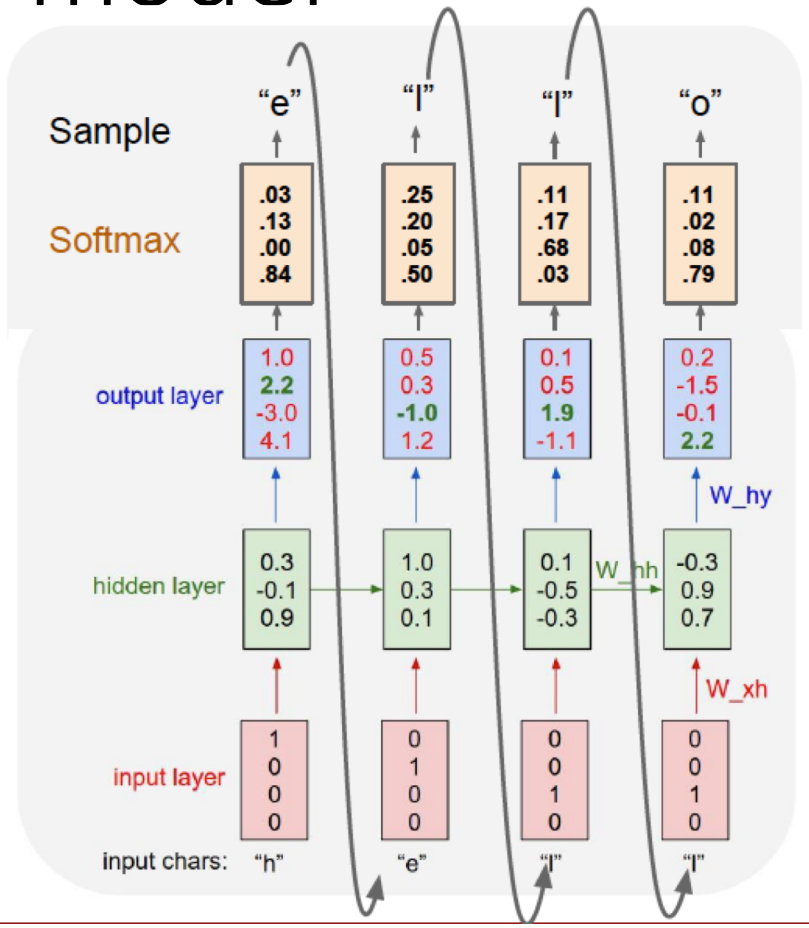
- Vocabulary: [h, e, l, o]
- Pada saat *test-time*, lakukan *sampling* 1 huruf/karakter, kemudian masukkan sebagai input *kembali*

Ex: character-level language model



- Vocabulary: [h, e, l, o]
- Pada saat *test-time*, lakukan *sampling* 1 huruf/karakter, kemudian masukkan sebagai input *kembali*

Ex: character-level language model



- Vocabulary: [h, e, l, o]
- Pada saat *test-time*, lakukan *sampling* 1 huruf/karakter, kemudian masukkan sebagai input *kembali*

Image Captioning: CNN + RNN

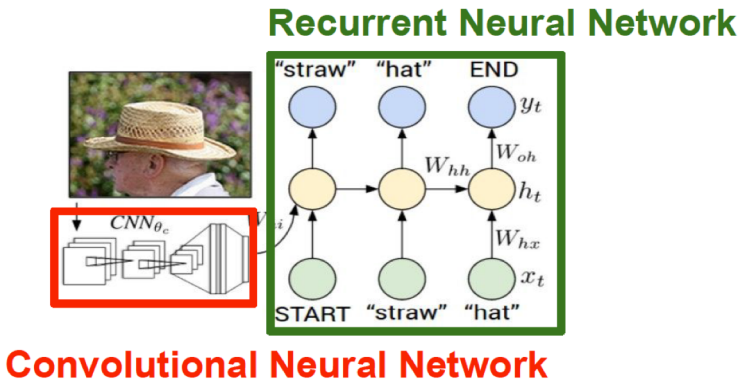


Image Captioning: CNN + RNN

Image Captioning: Example Results

Captions generated using neuraltalk2
All images are CC0 Public domain:
cat suitcase, cat tree, dog, bear,
surfers, tennis, giraffe, motorcycle

A cat sitting on a suitcase on the floor

A cat is sitting on a tree branch

A dog is running in the grass with a frisbee

A white teddy bear sitting in the grass

Two people walking on the beach with surfboards

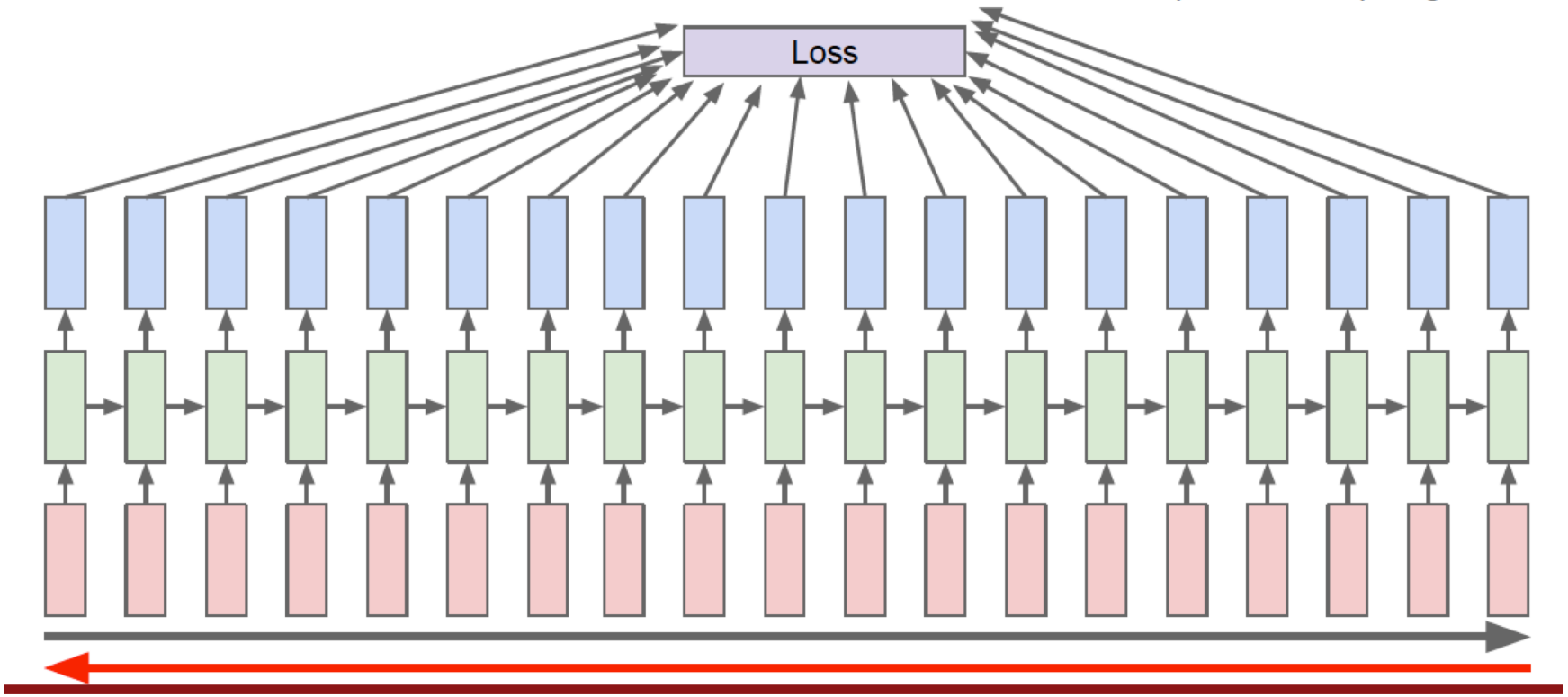
A tennis player in action on the court

Two giraffes standing in a grassy field

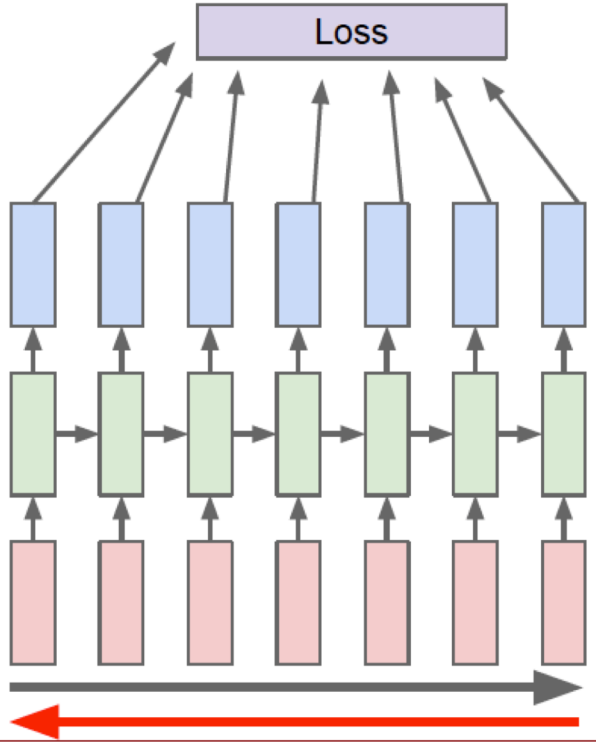
A man riding a dirt bike on a dirt track

Backpropagation through time

Forward through entire sequence to compute loss, then backward through entire sequence to compute gradient

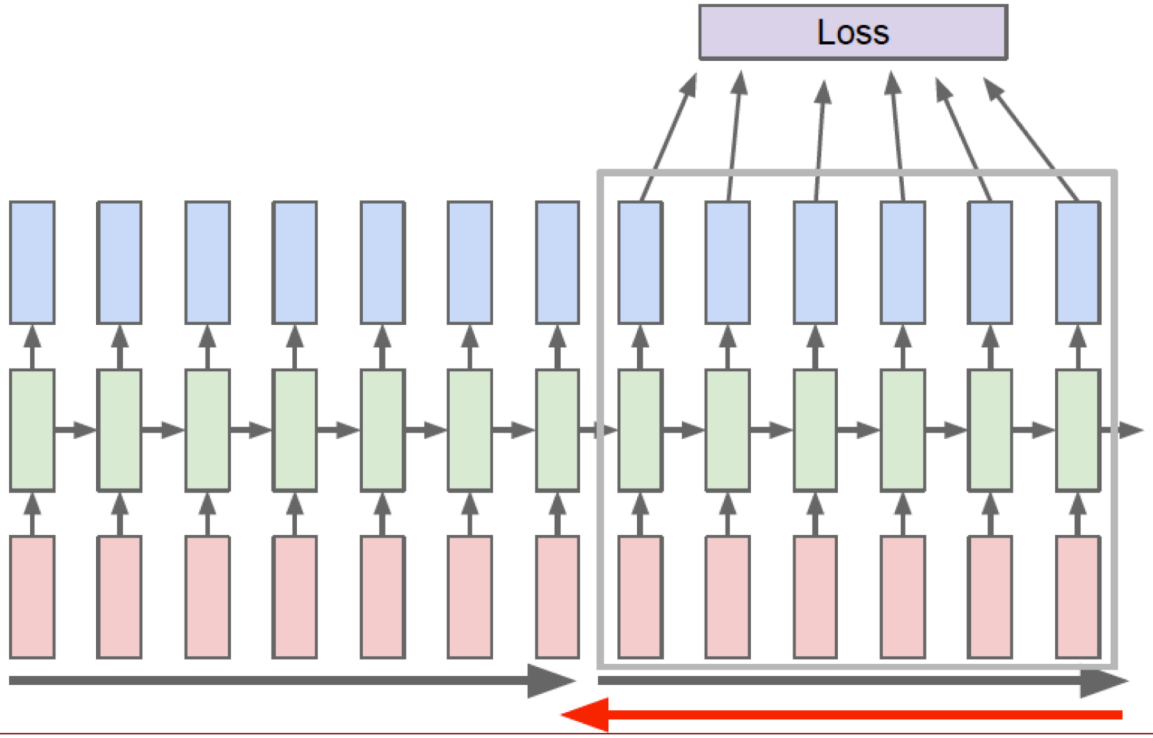


Truncated Backpropagation through time



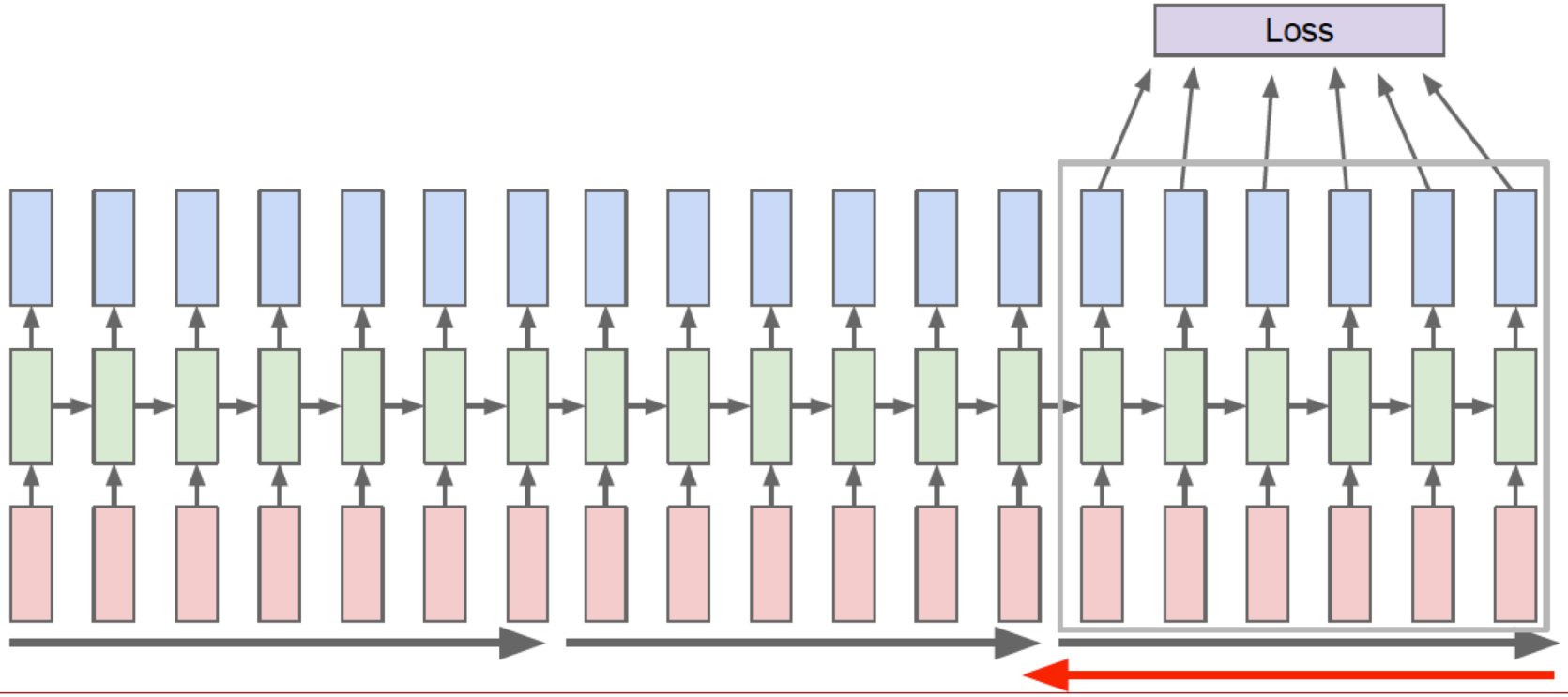
Run forward and backward through chunks of the sequence instead of whole sequence

Truncated Backpropagation through time

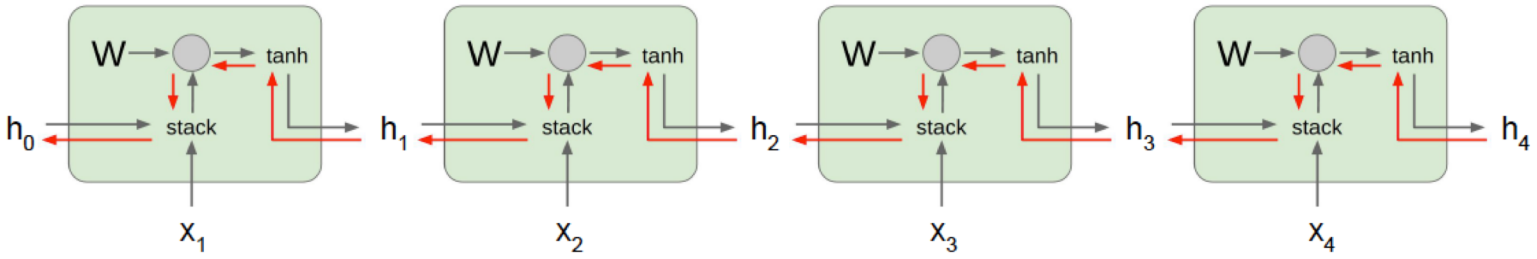


Carry hidden states forward in time forever, but only backpropagate for some smaller number of steps

Truncated Backpropagation through time



Vanilla RNN gradient flow



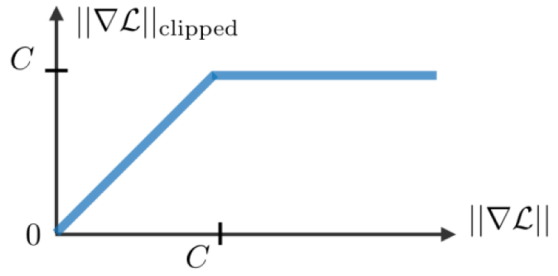
Computing gradient of h_0 involves many factors of W (and repeated tanh)

Largest singular value > 1: **Exploding gradients**

→ **Gradient clipping**: Scale gradient if its norm is too big

```
grad_norm = np.sum(grad * grad)
if grad_norm > threshold:
  grad *= (threshold / grad_norm)
```

Largest singular value < 1: **Vanishing gradients**



Mengapa LSTM?

RNN memiliki kelemahan yaitu:

- Harus merekam semua state (boros komputasi dan memori)
- Vanishing gradient
- Exploding gradient (solusi: Clipping)

Akibatnya RNN tidak dapat menangani long sequences dengan baik.

For the problem of vanishing gradients → change the RNN architecture to the Long Short-Term Memory (LSTM)

The LSTM Model

Bagian 3

LSTM

Vanilla RNN

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

LSTM

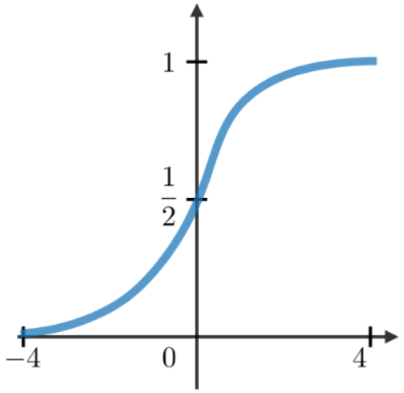
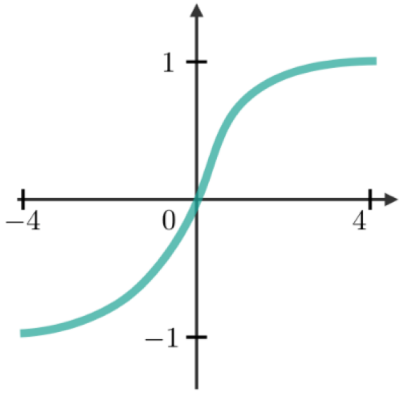
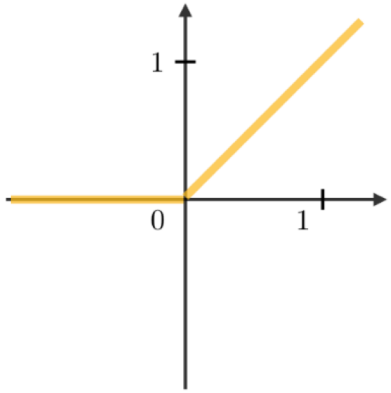
$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

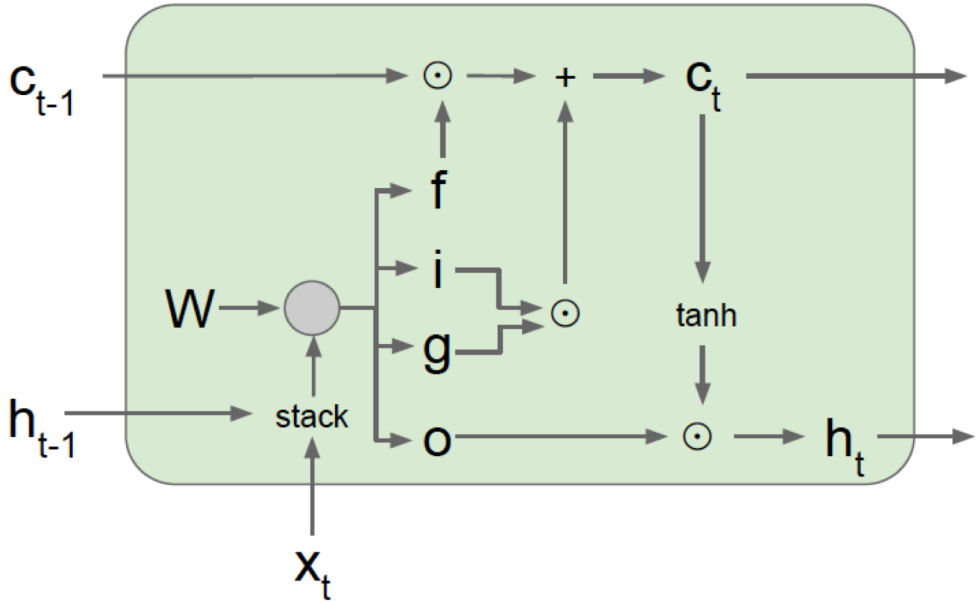
f: Forget gate, Whether to erase cell
i: Input gate, whether to write to cell
g: Gate gate (?), How much to write to cell
o: Output gate, How much to reveal cell

σ is sigmoid function

Sigmoid	Tanh	RELU
$g(z) = \frac{1}{1+e^{-z}}$	$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	$g(z) = \max(0, z)$
		

LSTM

.ong Short Term Memory (LSTM)
Hochreiter et al., 1997]



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

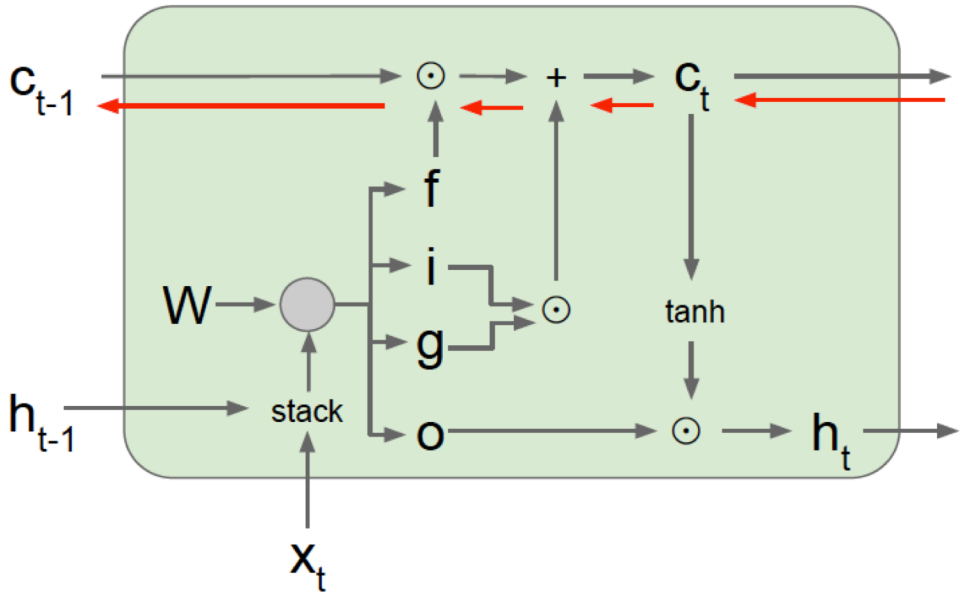
$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

LSTM

Long Short Term Memory (LSTM): Gradient Flow

[Hochreiter et al., 1997]



Backpropagation from c_t to c_{t-1} only elementwise multiplication by f, no matrix multiply by W

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

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

LSTM

Long Short Term Memory (LSTM): Gradient Flow
[Hochreiter et al., 1997]