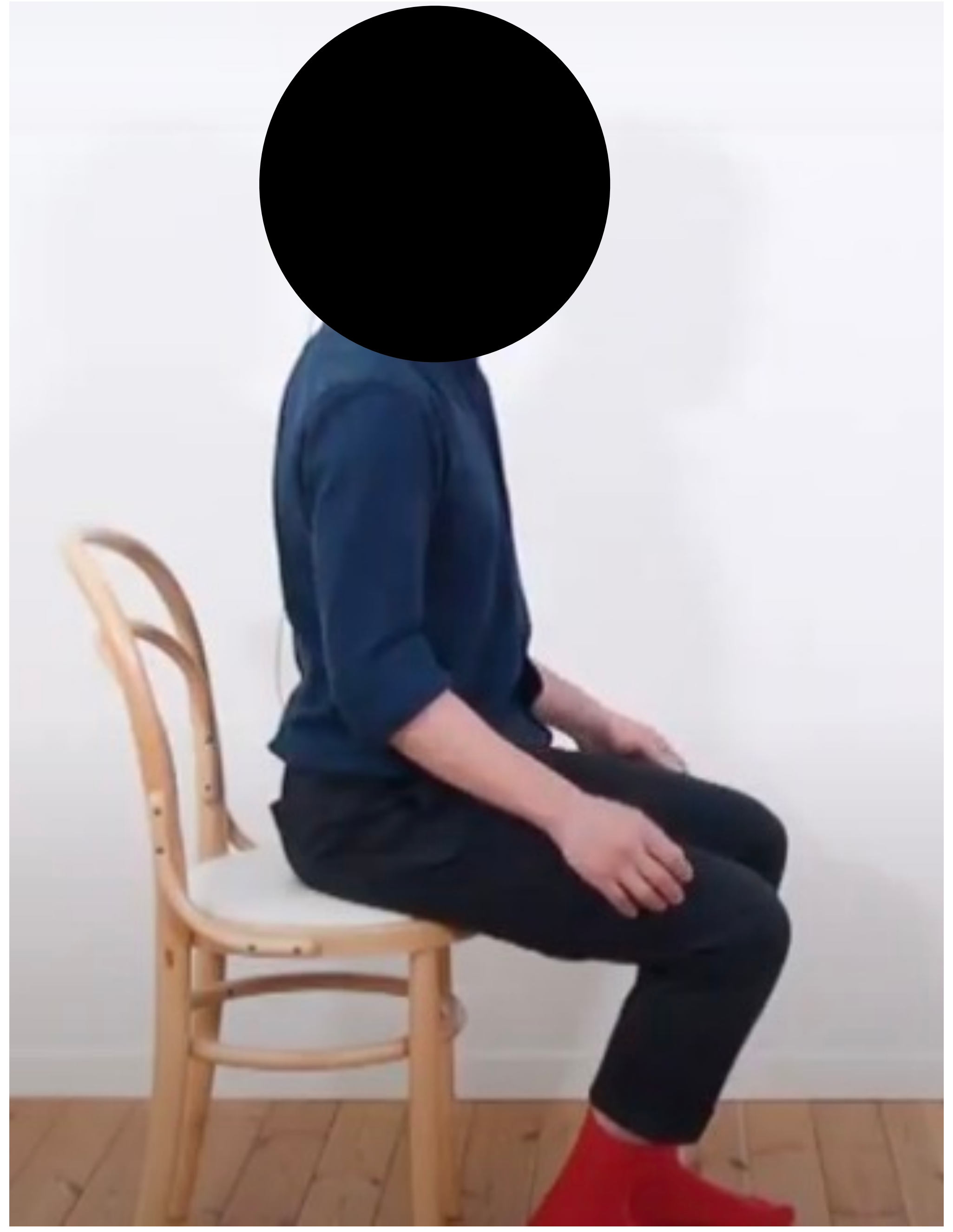
Temporal Convolutional Network & Recurrent Neural Network

COMP90051 Statistical Machine Learning

Qiuhong Ke

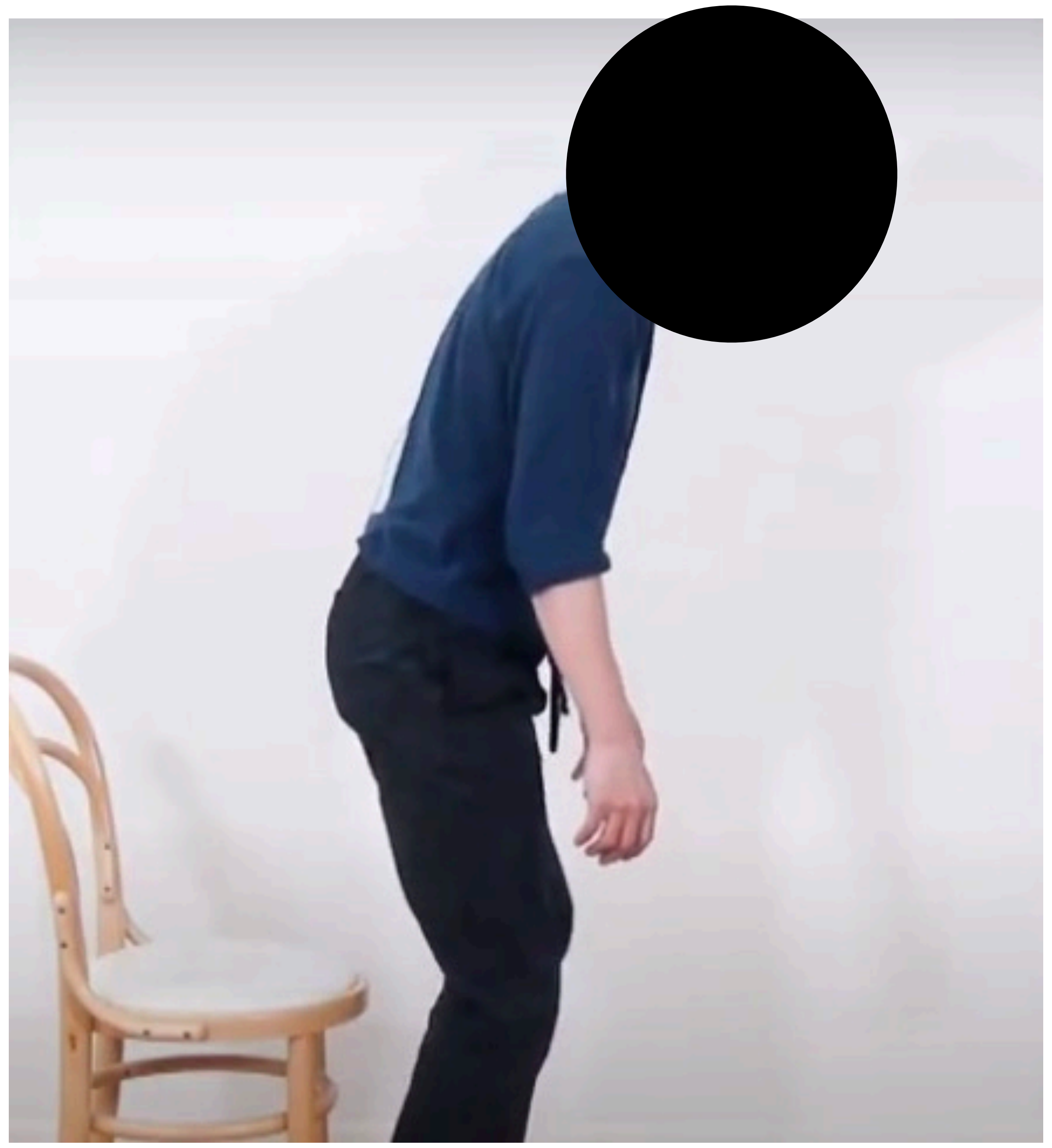
Sit down or stand up?

We need the learn the temporal information to understand the action



…

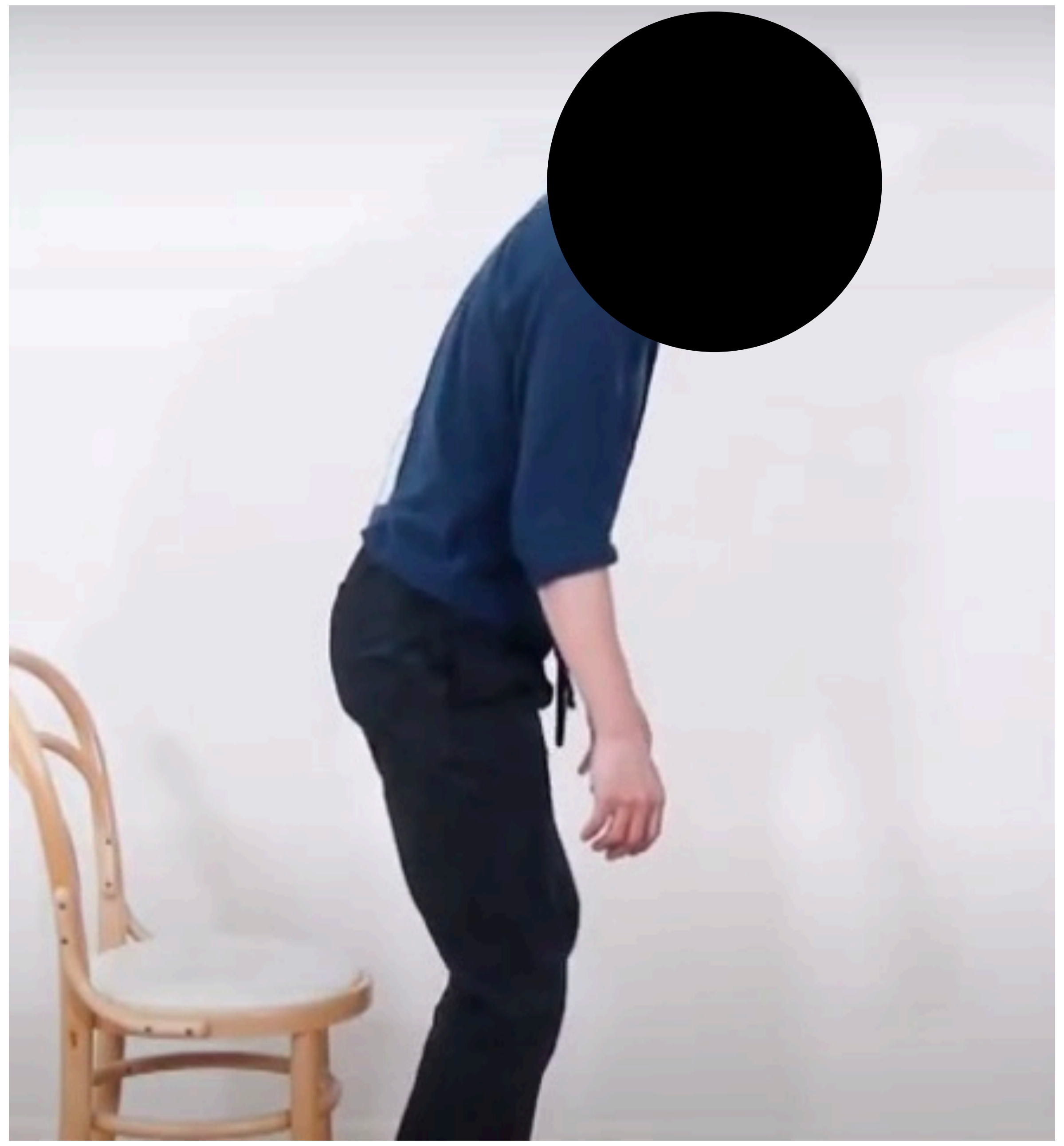
…



temporal information: temporal evolution (changes) of the human pose

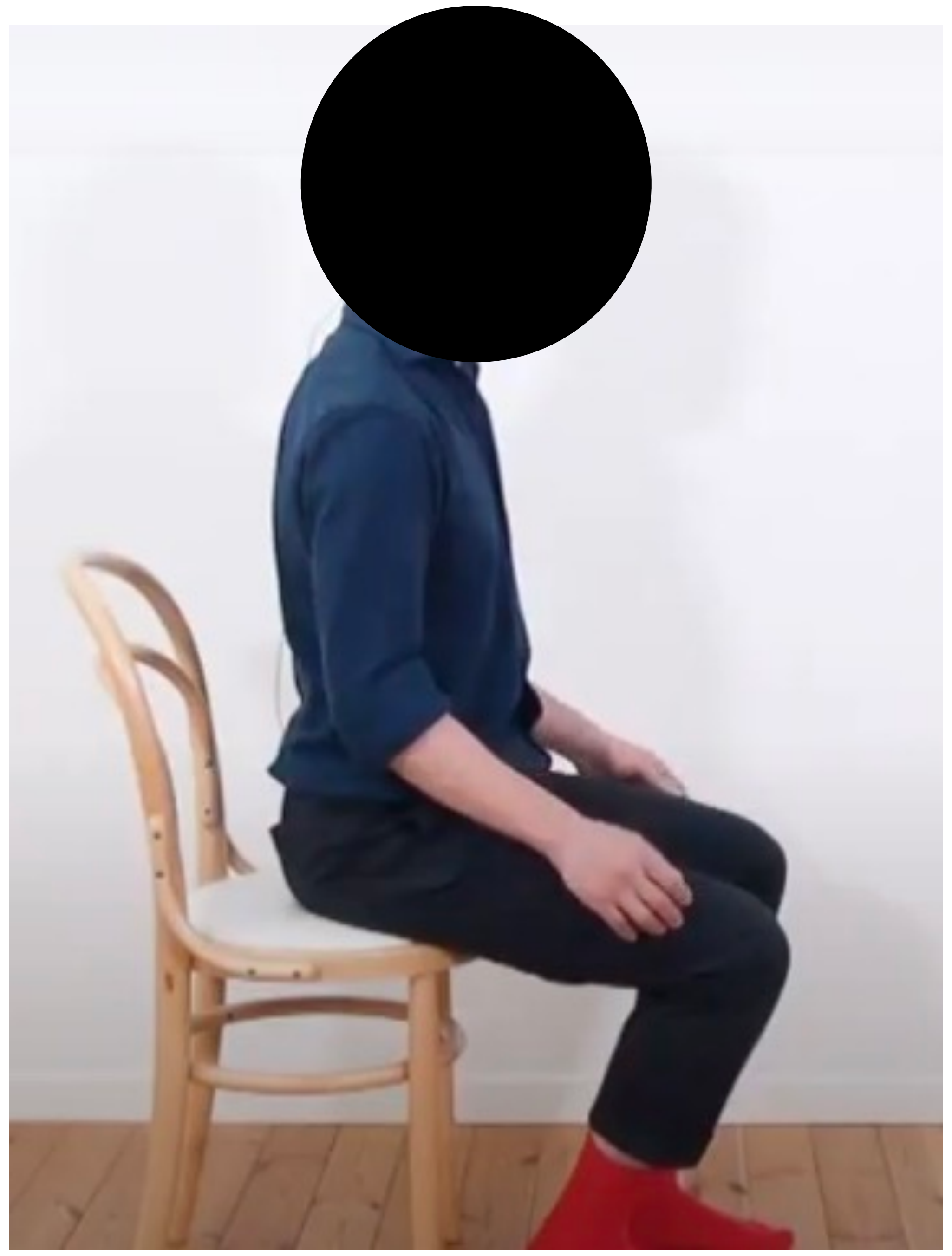
Temporal information does matter

Change the order: different actions



...

...



https://www.youtube.com/watch?time_continue=227&v=0N3PsjfXW9g&feature=emb_logo

What is the next word ?

This game is really fun

This cookie is really tasty

Like?Dislike? Sentiment analysis

I bought this charger in Jul 2003 and it worked OK for a while. The design is nice and convenient. ☺ *However, after about a year, the batteries would not hold a charge. Might as well just get alkaline disposables, or look elsewhere for a charger that comes with batteries that have better staying power.* ☹

It is important to understand the context the sentence

Amazon product data http://jmcauley.ucsd.edu/data/amazon/

Outline

- Temporal convolutional network (TCN)
- Recurrent neural network (RNN)

Recap: 2D Convolution

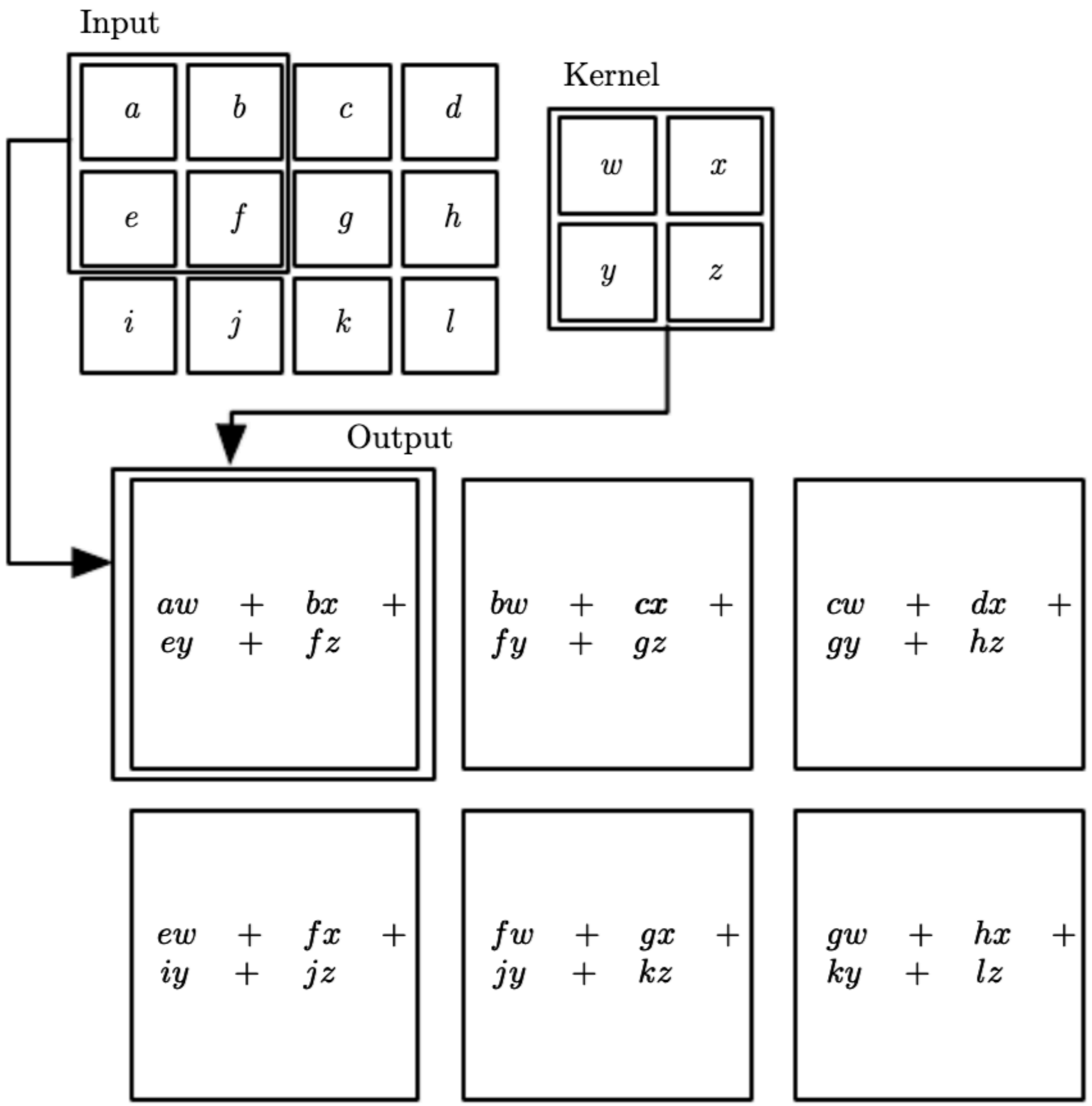
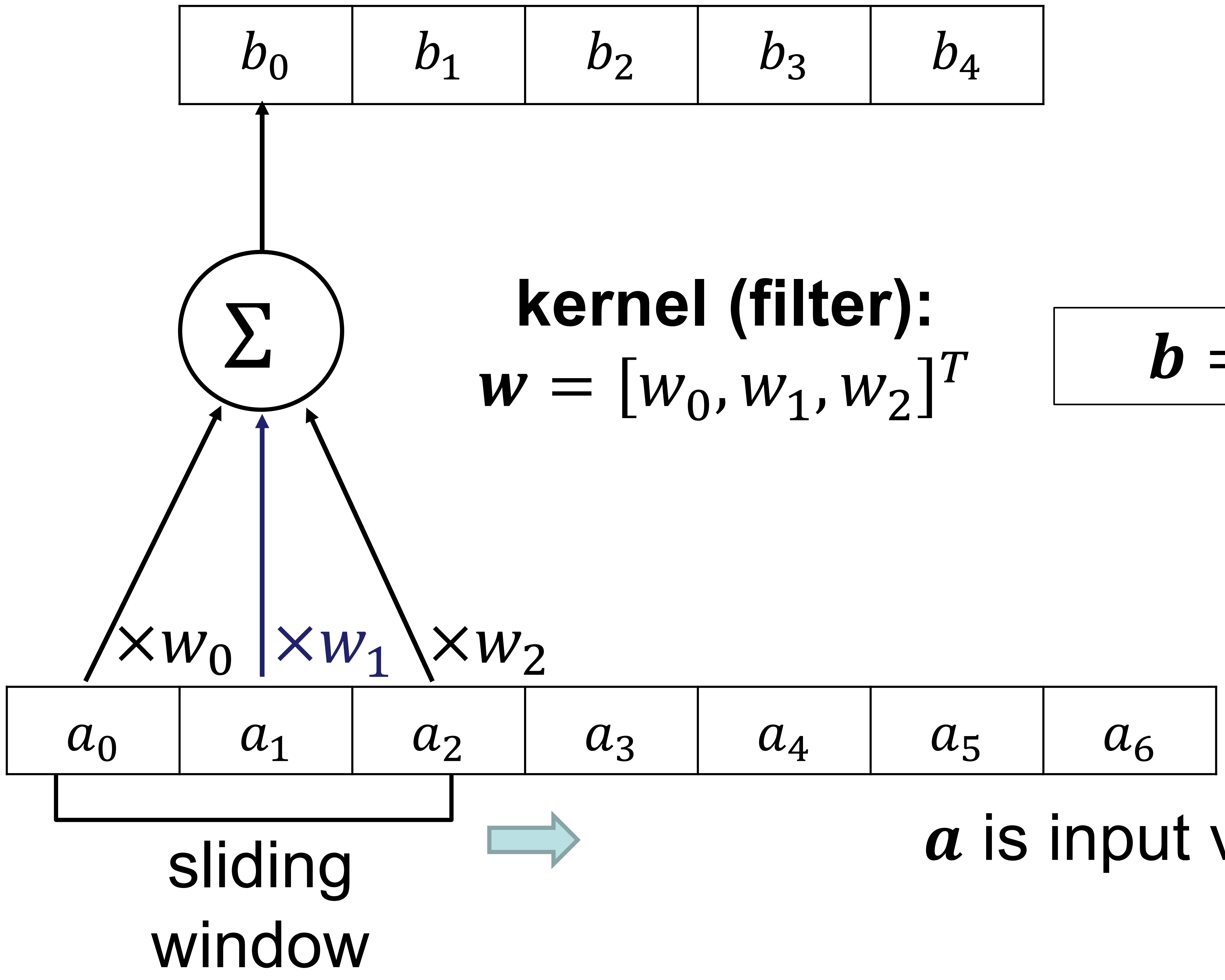


Figure 9.1 in Deep learning by Ian Goodfellow and Yoshua Bengio and Aaron Courville

Recap: 1D Convolution

$\boldsymbol{b}$ is output vector (feature map)

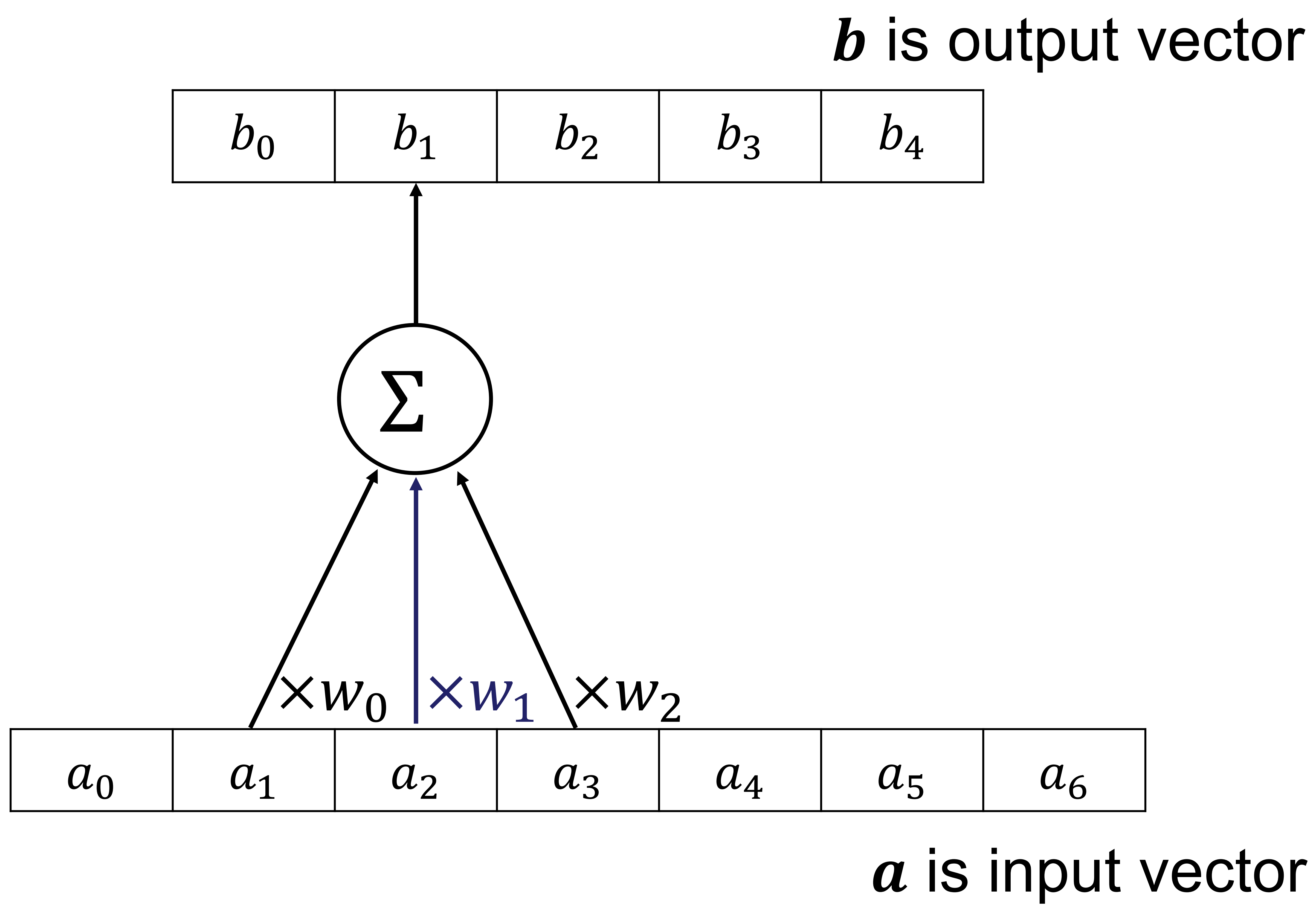


kernel (filter):
$\boldsymbol{w} = [w_0, w_1, w_2]^T$

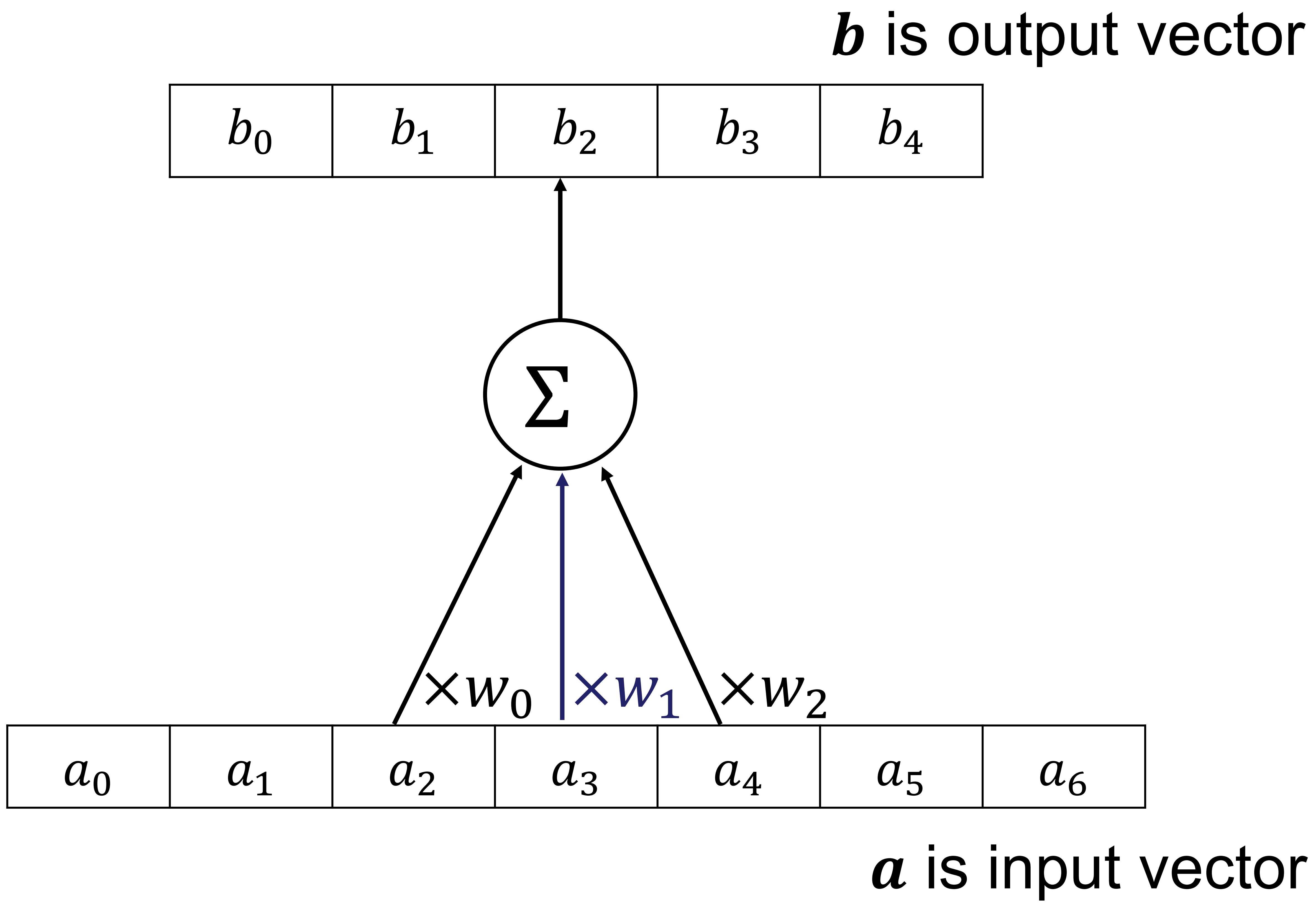
$$\boxed{\boldsymbol{b} = \boldsymbol{a} * \boldsymbol{w}}$$

$\boldsymbol{a}$ is input vector

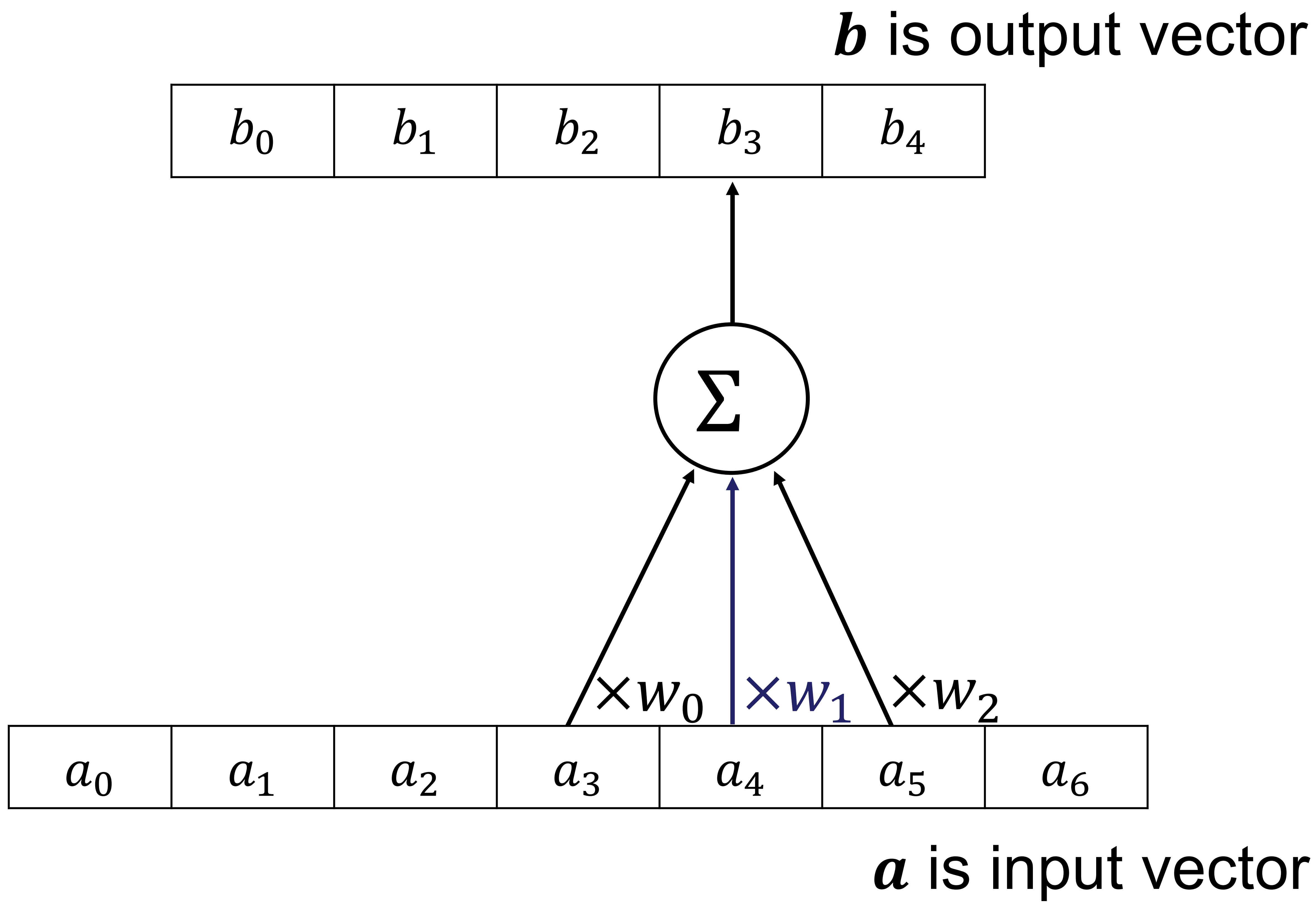
Recap: 1D Convolution



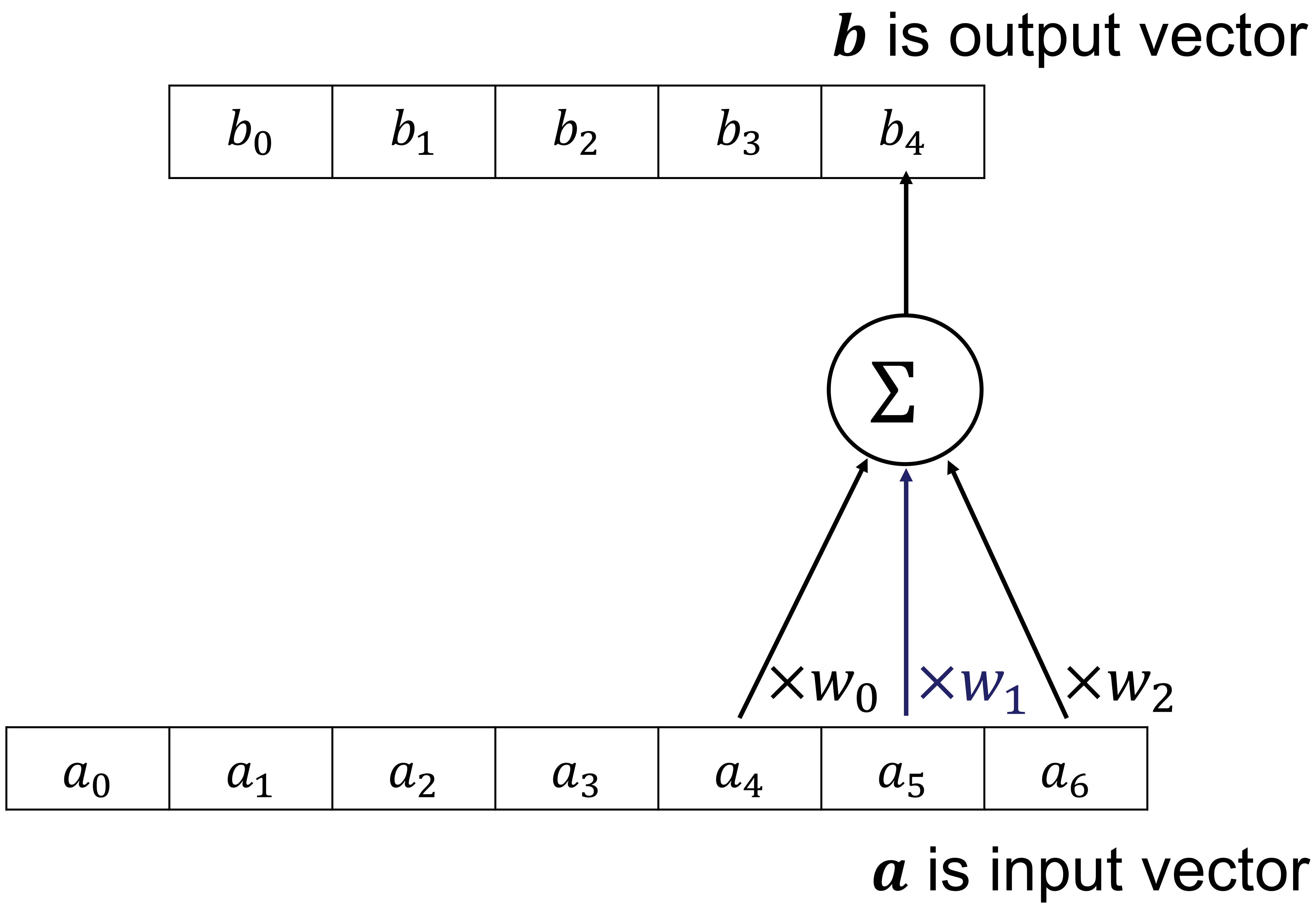
Recap: 1D Convolution



Recap: 1D Convolution

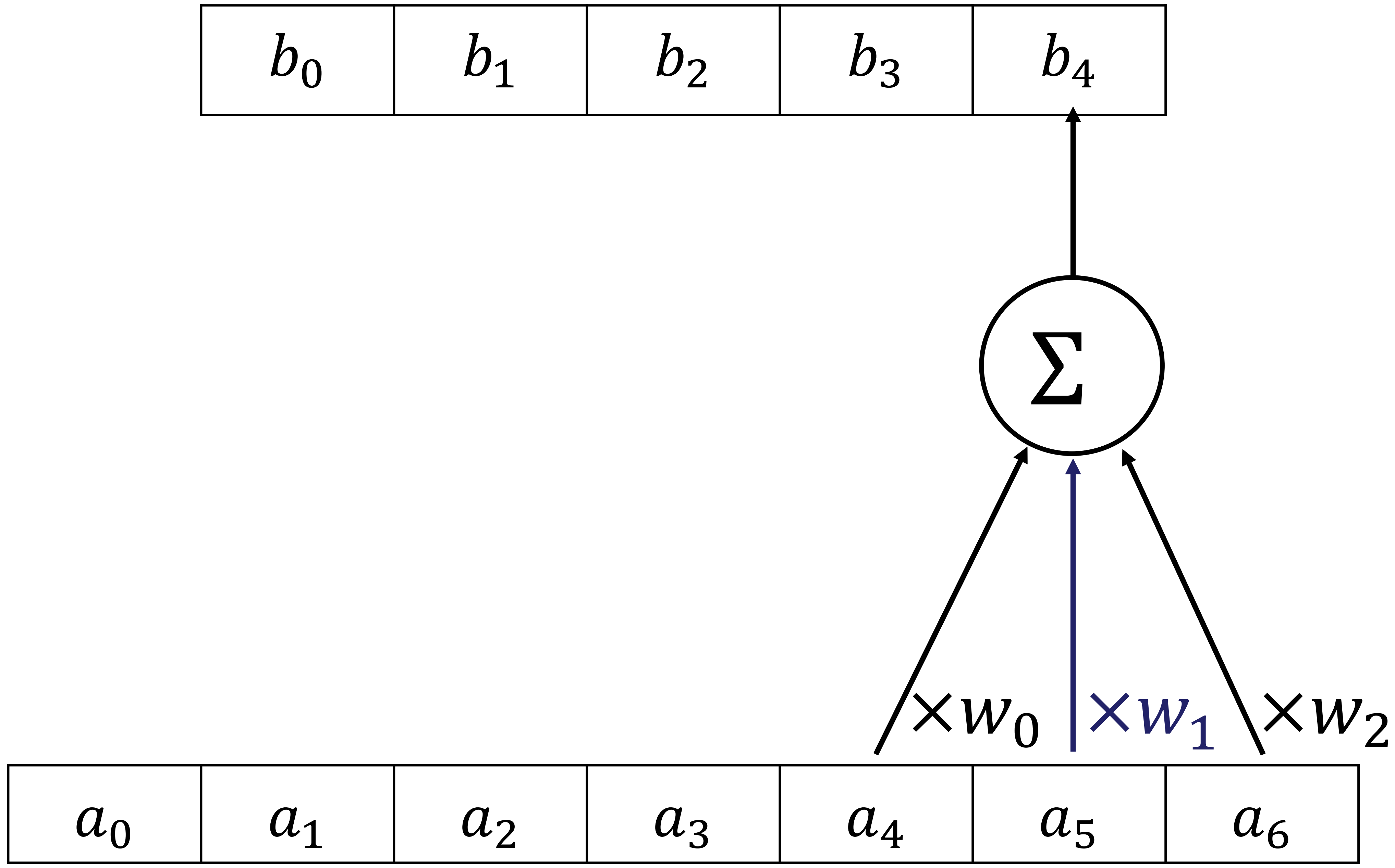


Recap: 1D Convolution



Temporal convolution: 1D convolution (on the temporal dimension)

1D input feature vector:
special case of sequence data



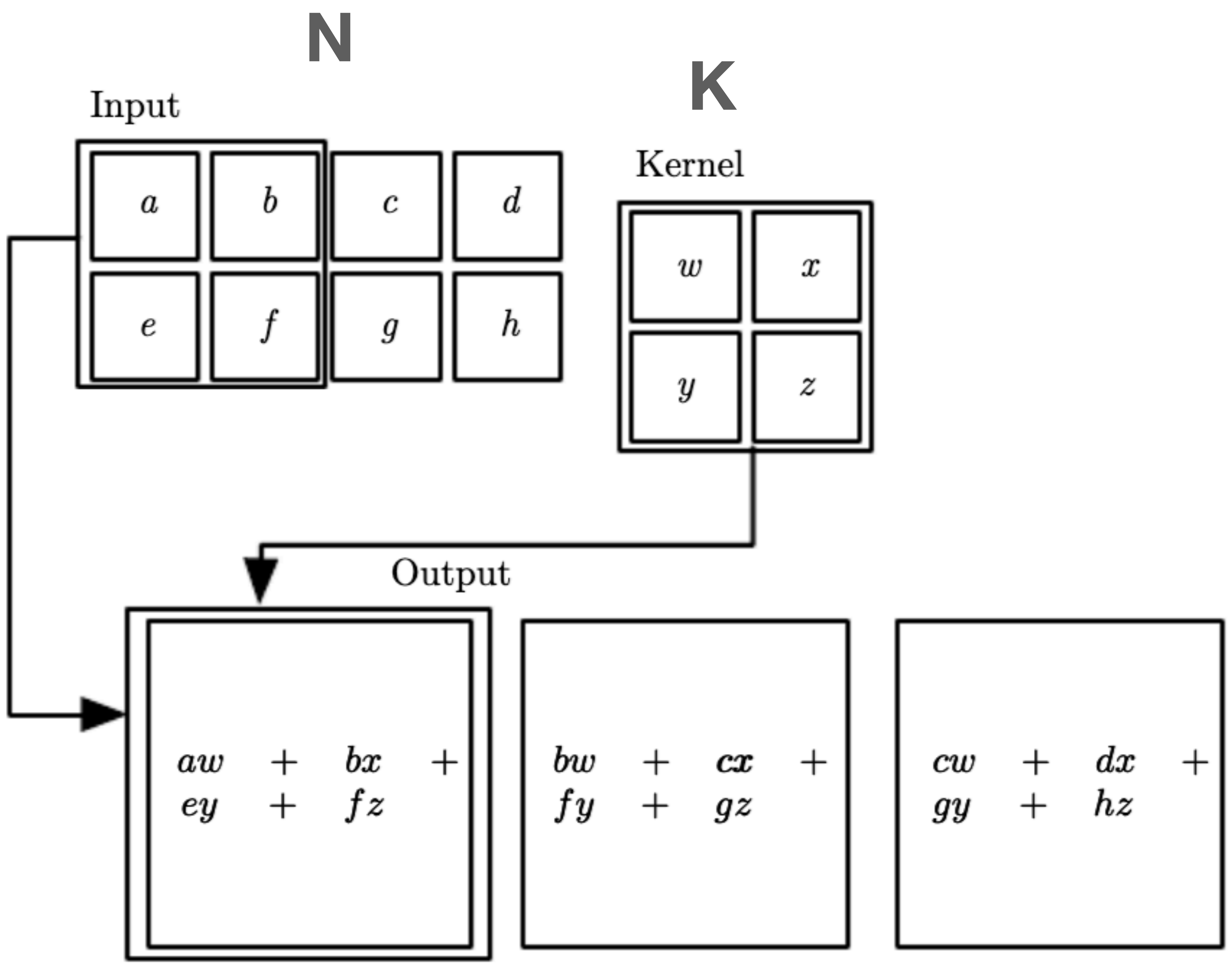
A sequence that consists of 7 time-steps:
One feature in each step

Temporal convolution on a sequence of feature vectors

Input:
- a sequence of N=4 time-steps
- Each time-step is a feature vector (dimension: d=2 (2 features))

Kernel: a weight matrix (dxK)
- One dimension is the same as d
- Another dimension: size of temporal convolutional window K

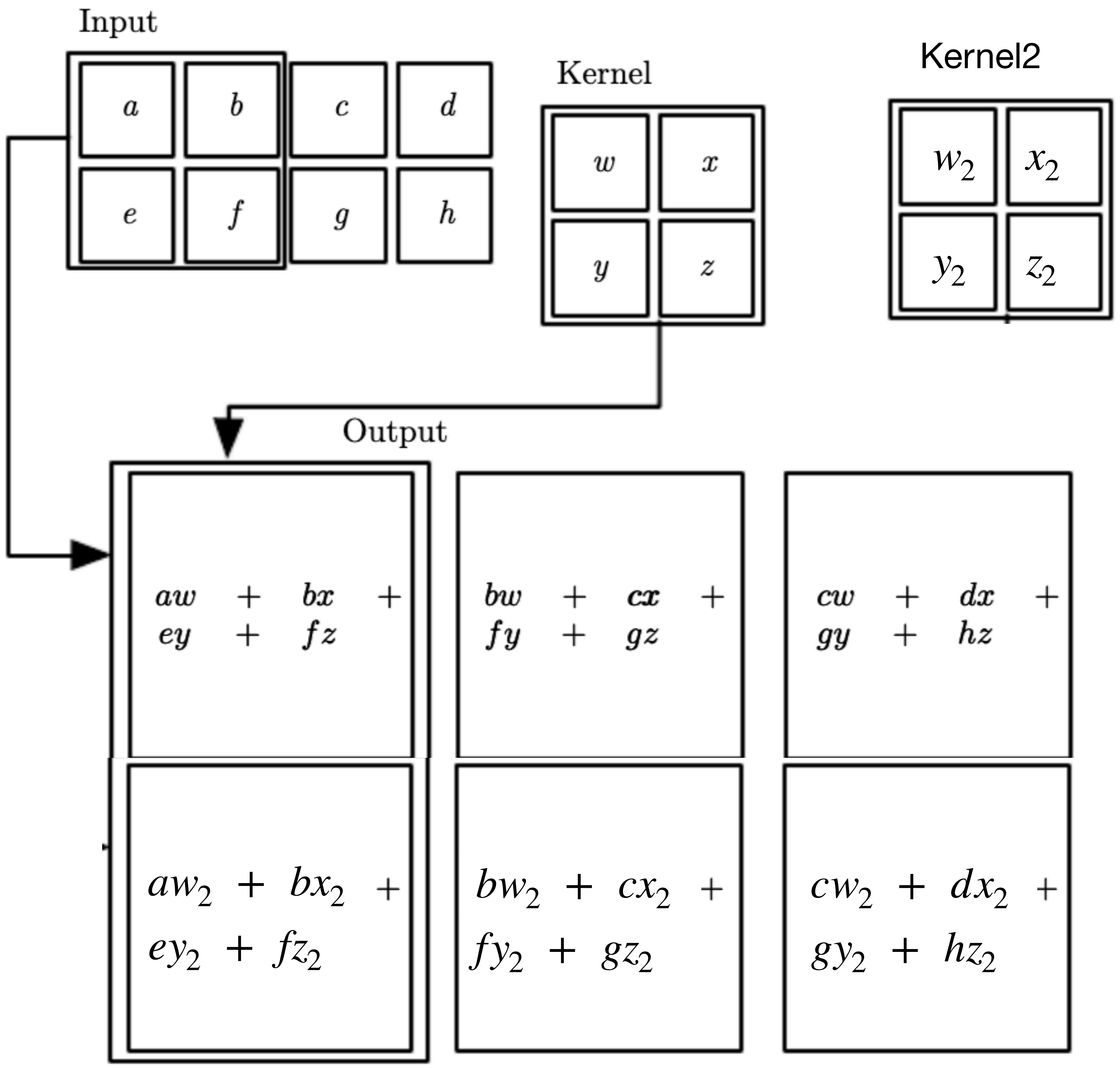
Output: A sequence
- Each time-step is a scalar (1 kernel)
- The number of time-steps (same as 2D):
  - No padding: ceiling (N-K+1)/stride
  - With padding: ceiling (N/stride)



Temporal convolution on a sequence of feature vectors

If there are m kernels in the layer
Output: m sequences==
a sequence of feature vectors:

- The number of time-steps:
  - No padding: ceiling (N-K+1)/stride
  - With padding: ceiling (N/stride)
- Each time-step is a vector
  - Dimension of each vector == kernel number m



Temporal convolution for text

I bought this charger in Jul 2003 and it worked OK for a while. The design is nice and convenient. However, after about a year, the batteries would not hold a charge. Might as well just get alkaline disposables, or look elsewhere for a charger that comes with batteries that have better staying power.

How to represent words at each time-step before temporal convolution

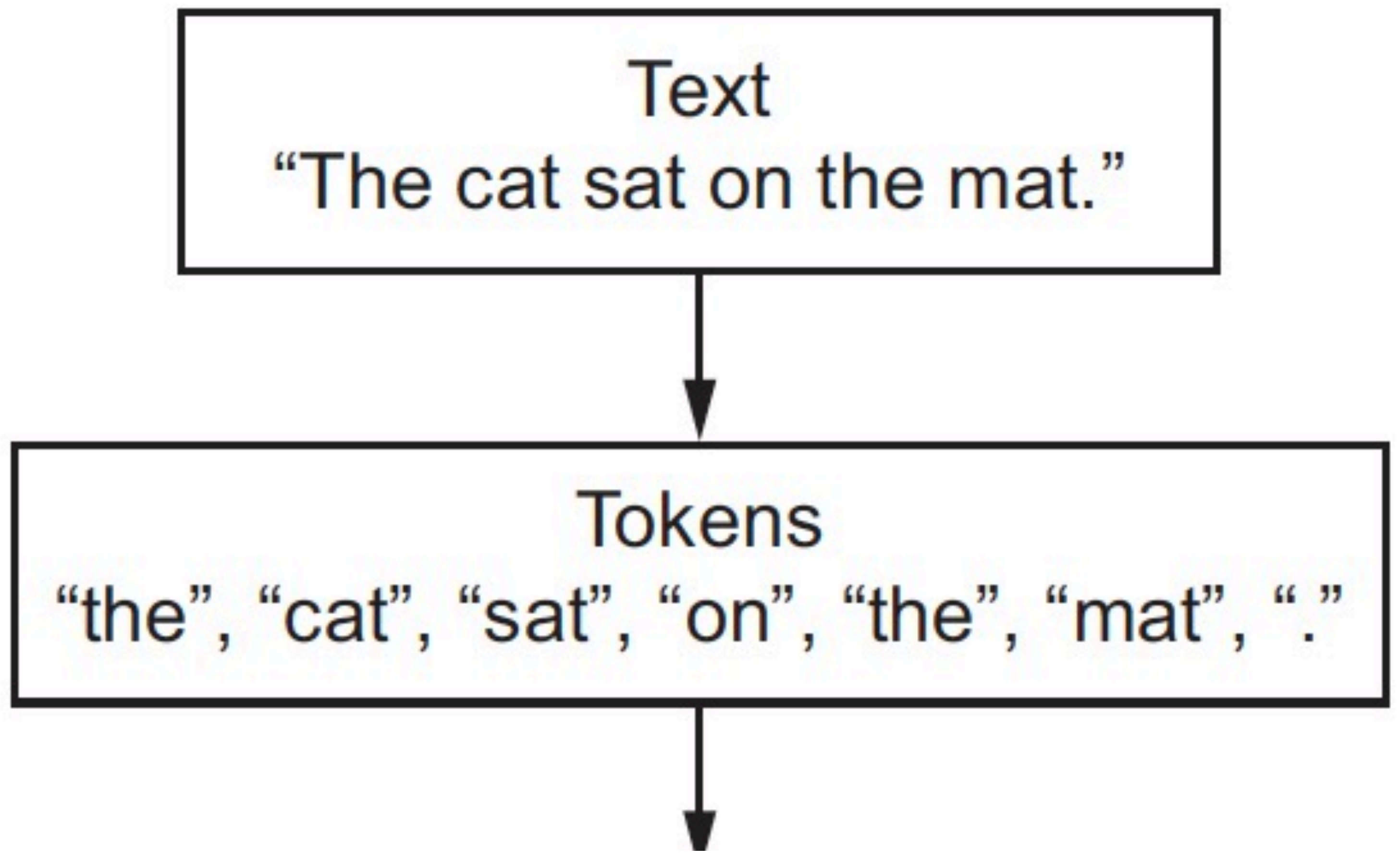
- One-hot encoding
- Word embeding

https://www.youtube.com/watch?time_continue=227&v=0N3PsjfXW9g&feature=emb_logo

One-hot encoding

- Create a feature vector
- Dimension of the vector == size of vocabulary
- If the word is the i^{th} word in the vocabulary, then the i^{th} element of the vector is 1, all the others are 0
- Sparse and high-dimensional

dic={“the”,”cat”,”sat”,”on”,”mat”,”.”,
“these”,”are”,”other”,”words”}



1	0	0	0	1	0	0
0	1	0	0	0	0	0
0	0	1	0	0	0	0
0	0	0	1	0	0	0
0	0	0	0	0	1	0
0	0	0	0	0	0	1
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Word embedding: learn low-dimensional embedding from data

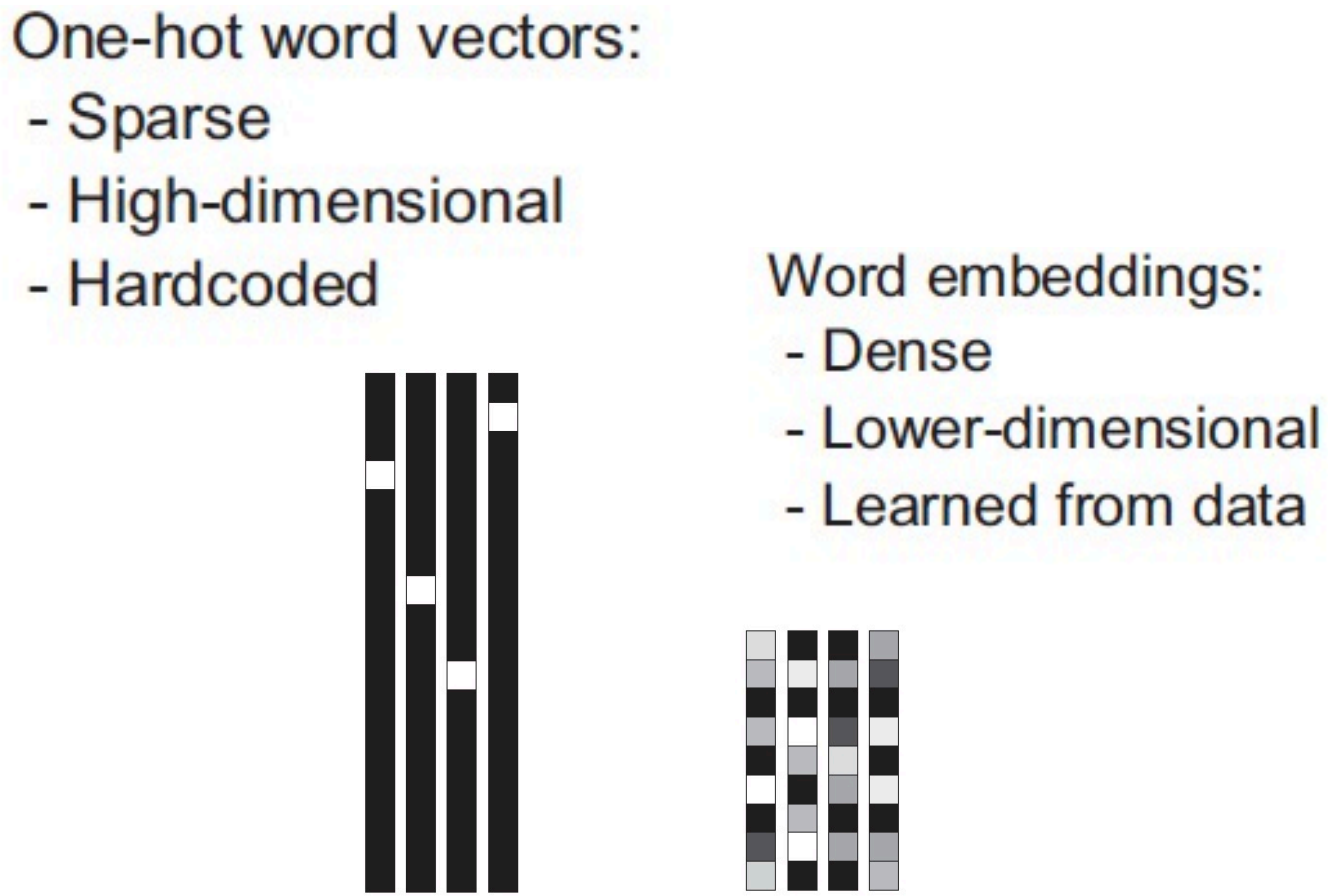


Figure 6.3 in Deep learning with python by Francois Chollet

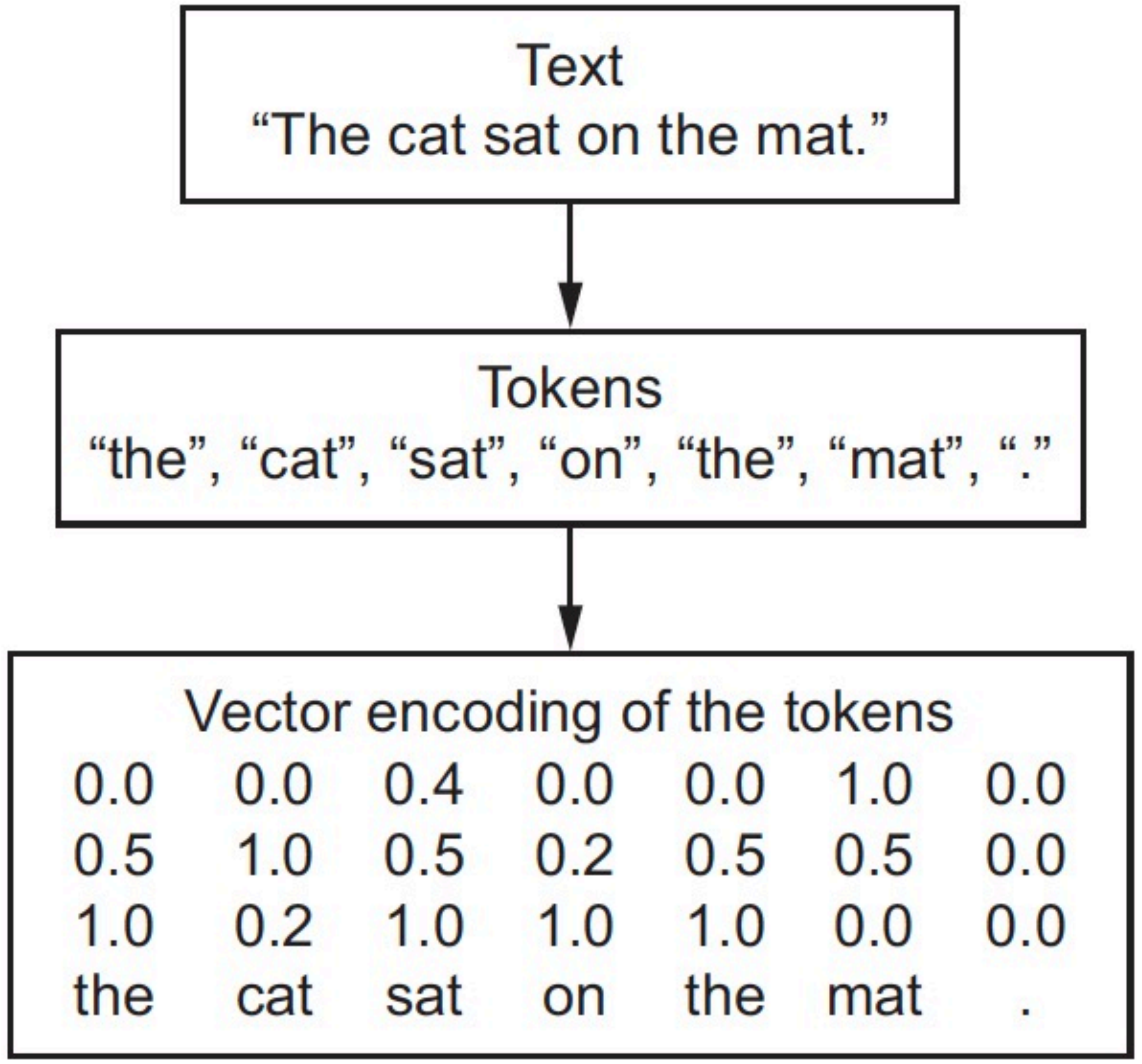


Figure 6.1 in Deep learning with python by Francois Chollet

Word embedding: learn low-dimensional embedding from data

```
Emb_layer = tf.keras.layers.Embedding(vocabulary_size, emb_dim)
```

Word index → Embedding layer → Corresponding word vector

Embedding layer:

- Create a weight matrix of d (emb_dim) rows, N (vocabulary_size) columns
- Return the i^{th} column as the feature vector for the i^{th} word
- Learn word embeddings jointly with the main task or load pretrained word embeddings

Figure 6.4 in Deep learning with python by Francois Chollet

Example

Input:

Word embedding

I	love	deep	learning
0.1	0.4	0.2	-0.5
0.3	0.3	0.1	0.3

Kernel:

1	0	0
0	0	1

element-wise multiplication and sum

Output:

$\boxed{0.2}$

Example

Input:

Word embedding	I	love	deep	learning
	0.1	0.4	0.2	-0.5
	0.3	0.3	0.1	0.3

Kernel:

1	0	0
0	0	1

element-wise multiplication and sum

Output:

0.2 0.7

Example

Input:

	I	love	deep	learning
Word embedding	0.1	0.4	0.2	-0.5
	0.3	0.3	0.1	0.3

2 Kernels:

0	0	1
0	1	0

element-wise multiplication and sum

Output:

0.0	0.4
0.5	

Example

Input:

	I	love	deep	learning
Word embedding	0.1	0.4	0.2	-0.5
	0.3	0.3	0.1	0.3

2 Kernels:

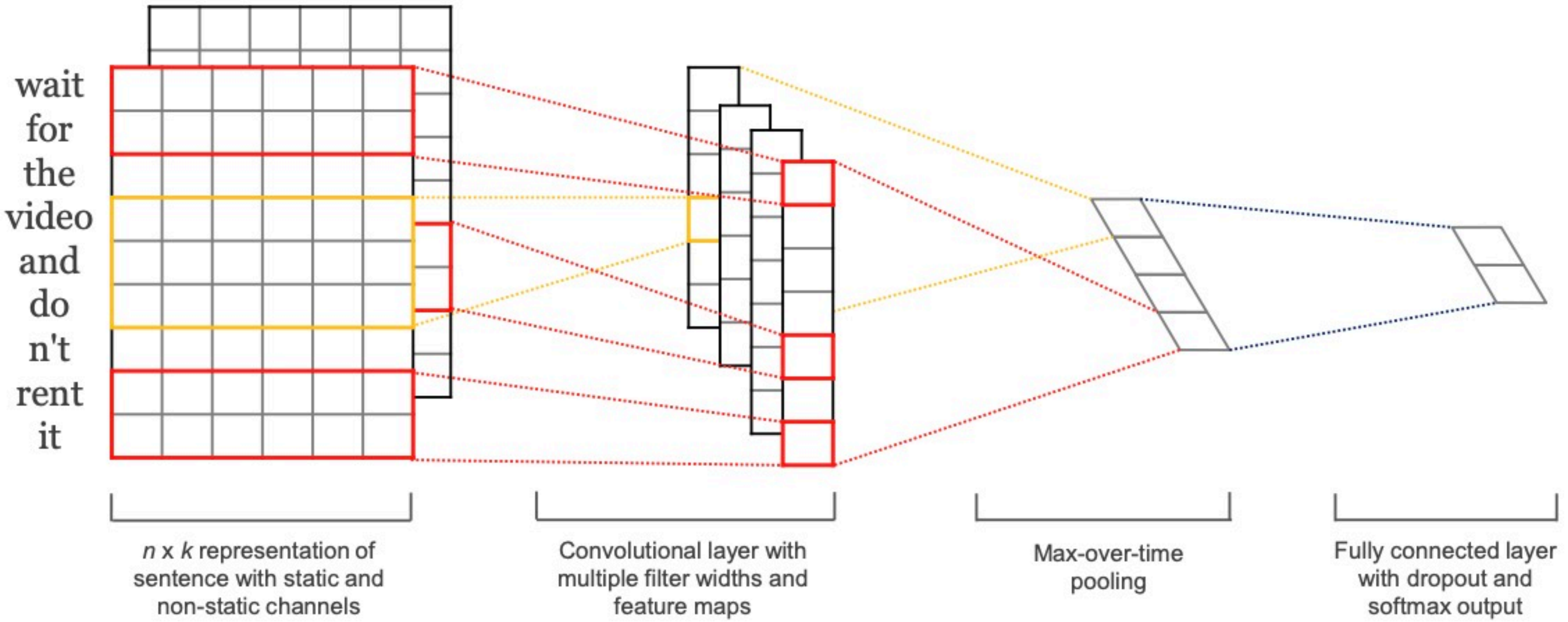
0	0	1
0	1	0

element-wise multiplication and sum

Output:

0.0	0.4
0.5	-0.4

TCN for Sentence Classification



Kim, Yoon. "Convolutional neural networks for sentence classification." *arXiv preprint arXiv:1408.5882* (2014).

Recap of Convolution on Multiple-channel input

R

G

B

Kernel: same channel (depth)

$* K$(Channel 1)

$* K$(Channel 2)

$* K$(Channel 3)

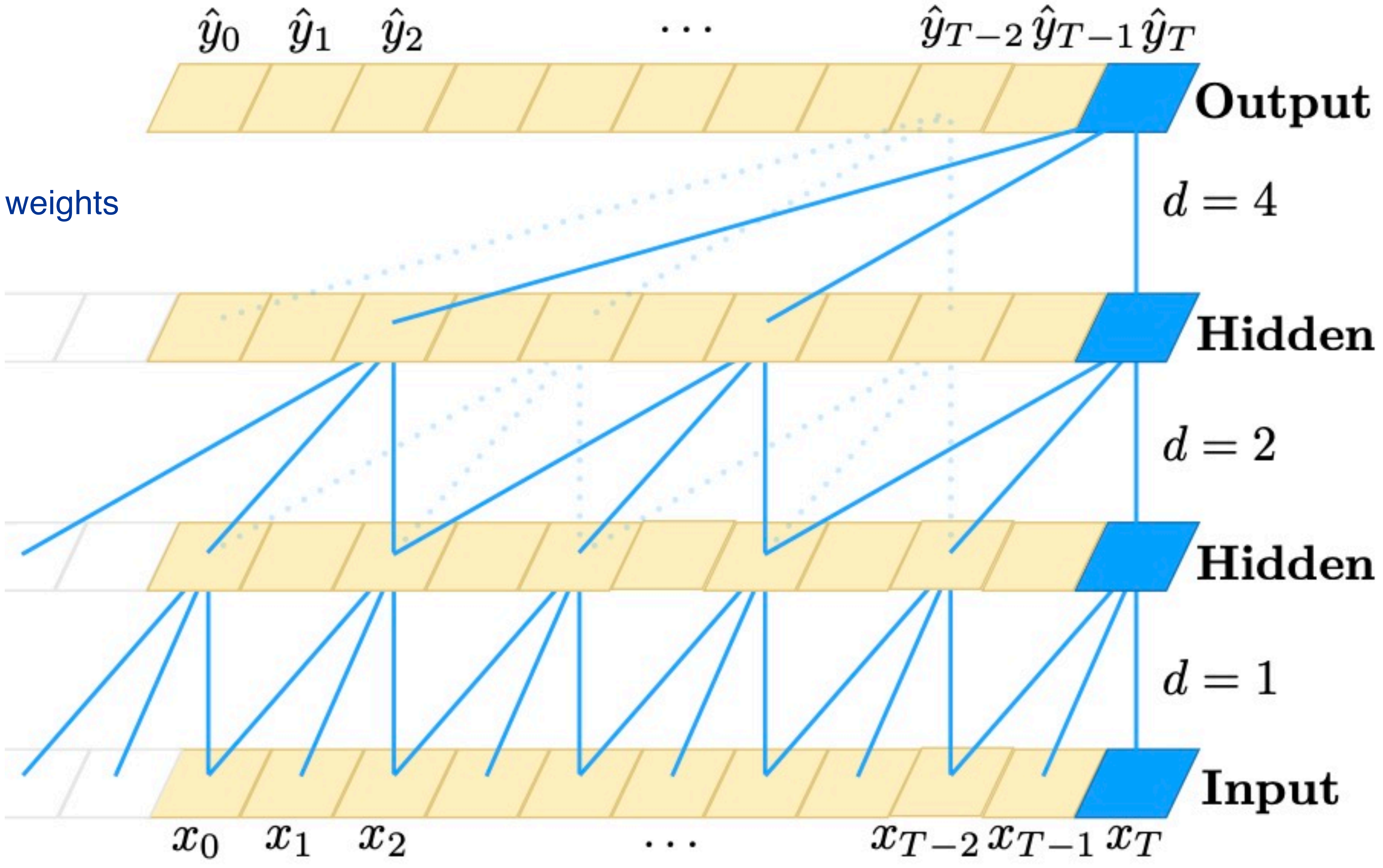
Element-wise sum

One channel

A special version: Dilated casual convolution

Dilated: expand the alignment of the kernel weights by dilation factor

Inserting weight between weights

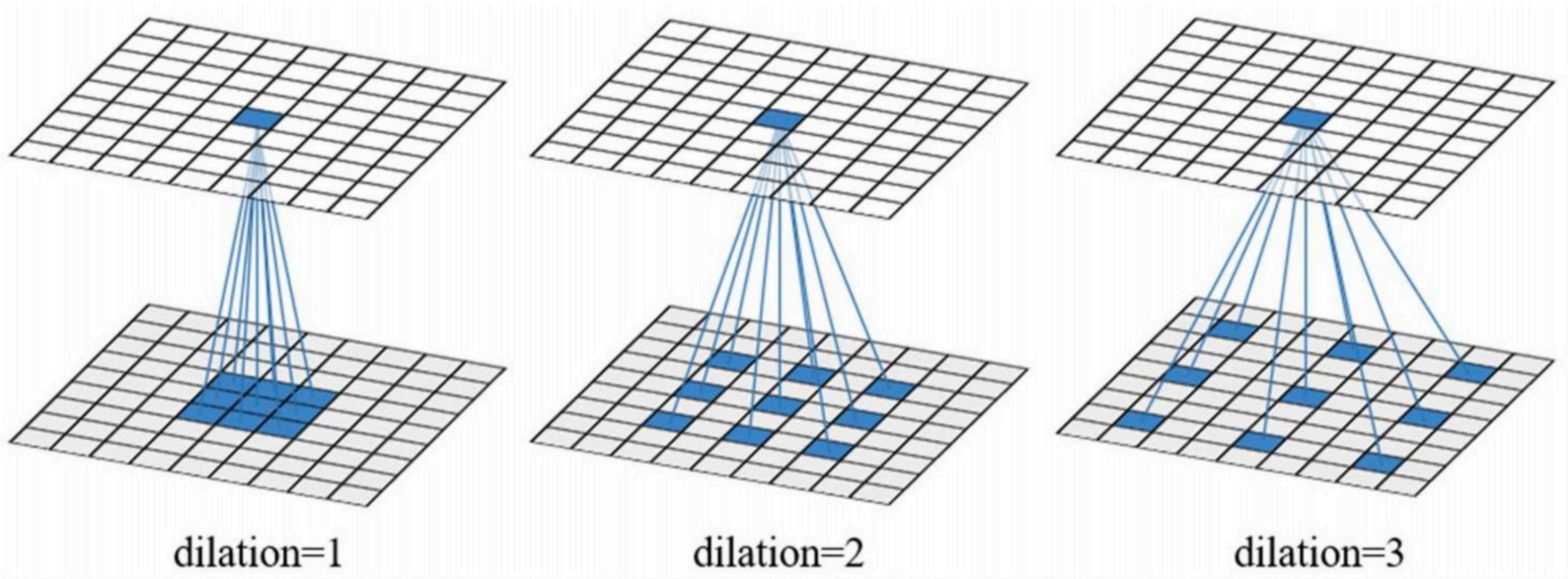


Dilation factor (d): step between every two adjacent filter taps.

Larger dilation factor: the weights are placed more far away (i.e., more sparse),

Bai, , J. Zico Kolter, and Vladlen Koltun. "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling." *arXiv preprint arXiv:1803.01271* (2018).

Dilated convolution in 2D

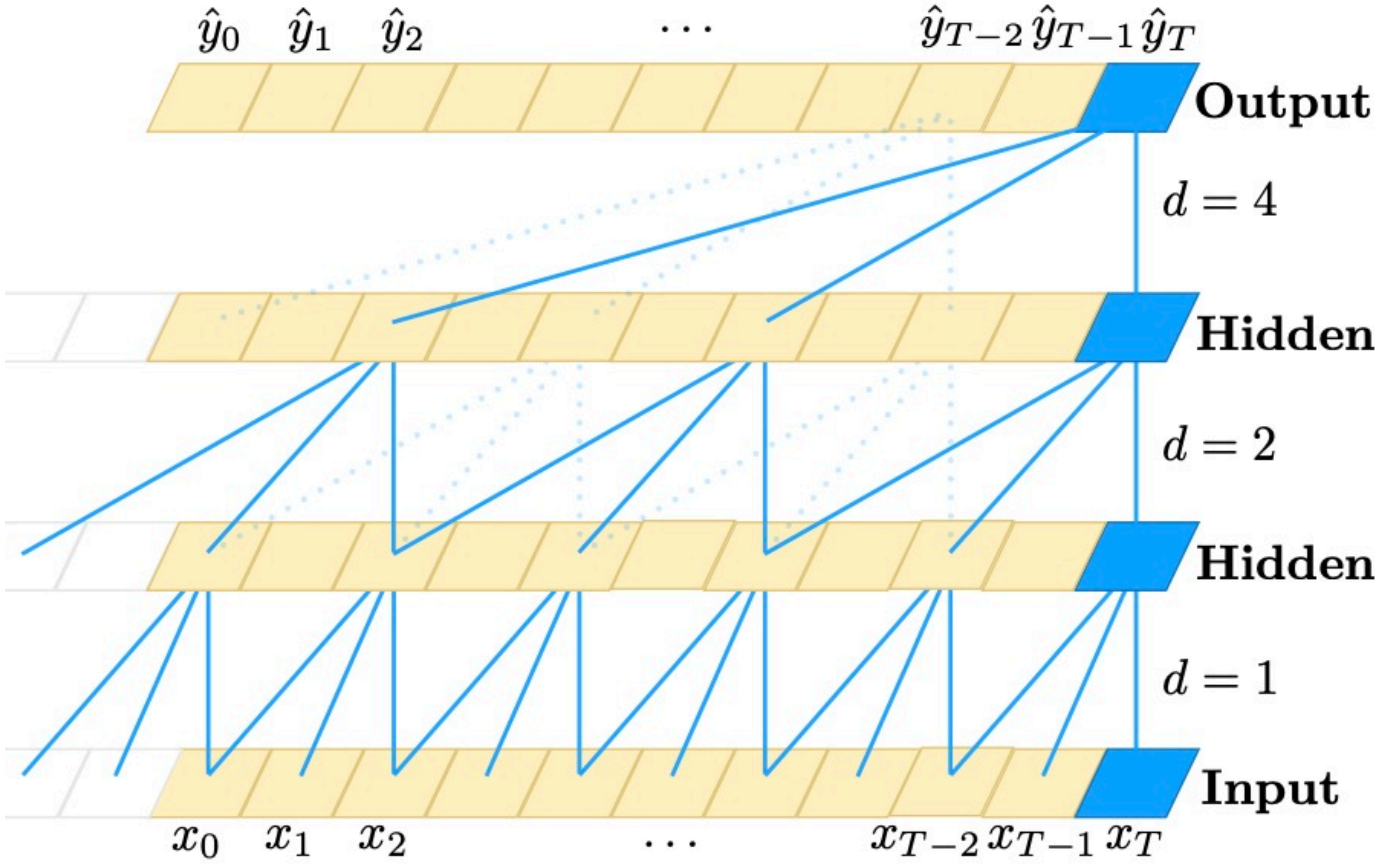


Increasing receptive fields

- Stacking multiple convolutional layers
- max-Pooling layer
- Dilated convolution

Dilated casual convolution

Causal: The output at at time-step t does not depend on the input information after time-step t (e.g., real-time recognition, prediction)



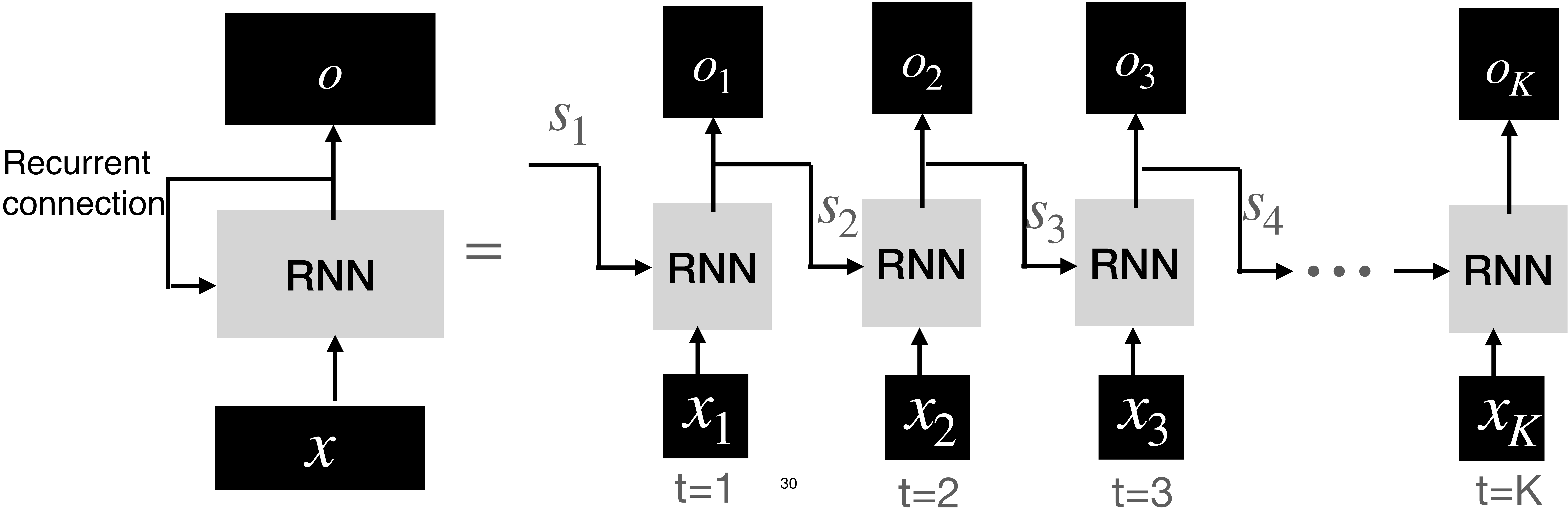
We perform padding between time step features where the current features are only depending on historical data and current data

Bai, , J. Zico Kolter, and Vladlen Koltun. "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling." *arXiv preprint arXiv:1803.01271* (2018).

Networks with loops

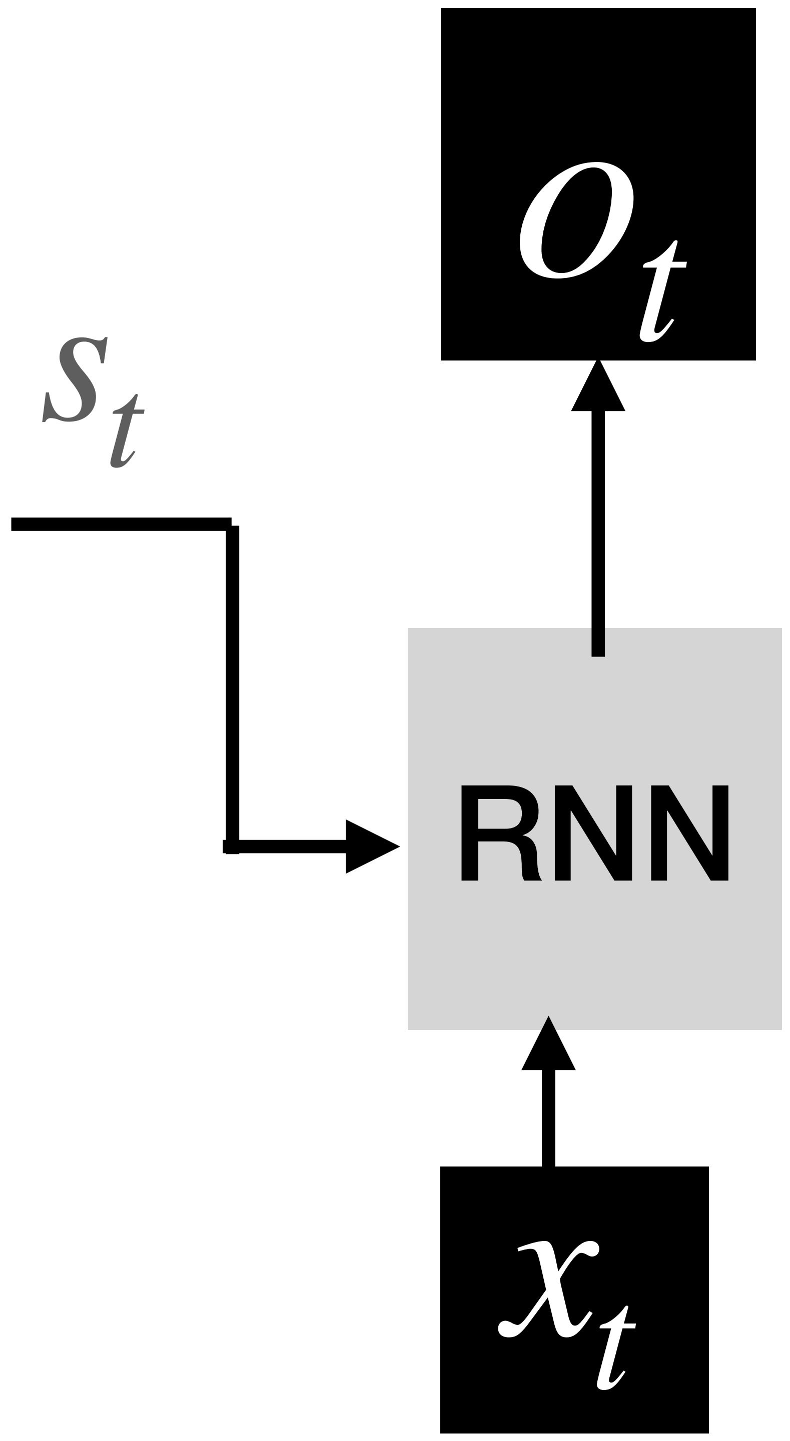
Process sequence step by step

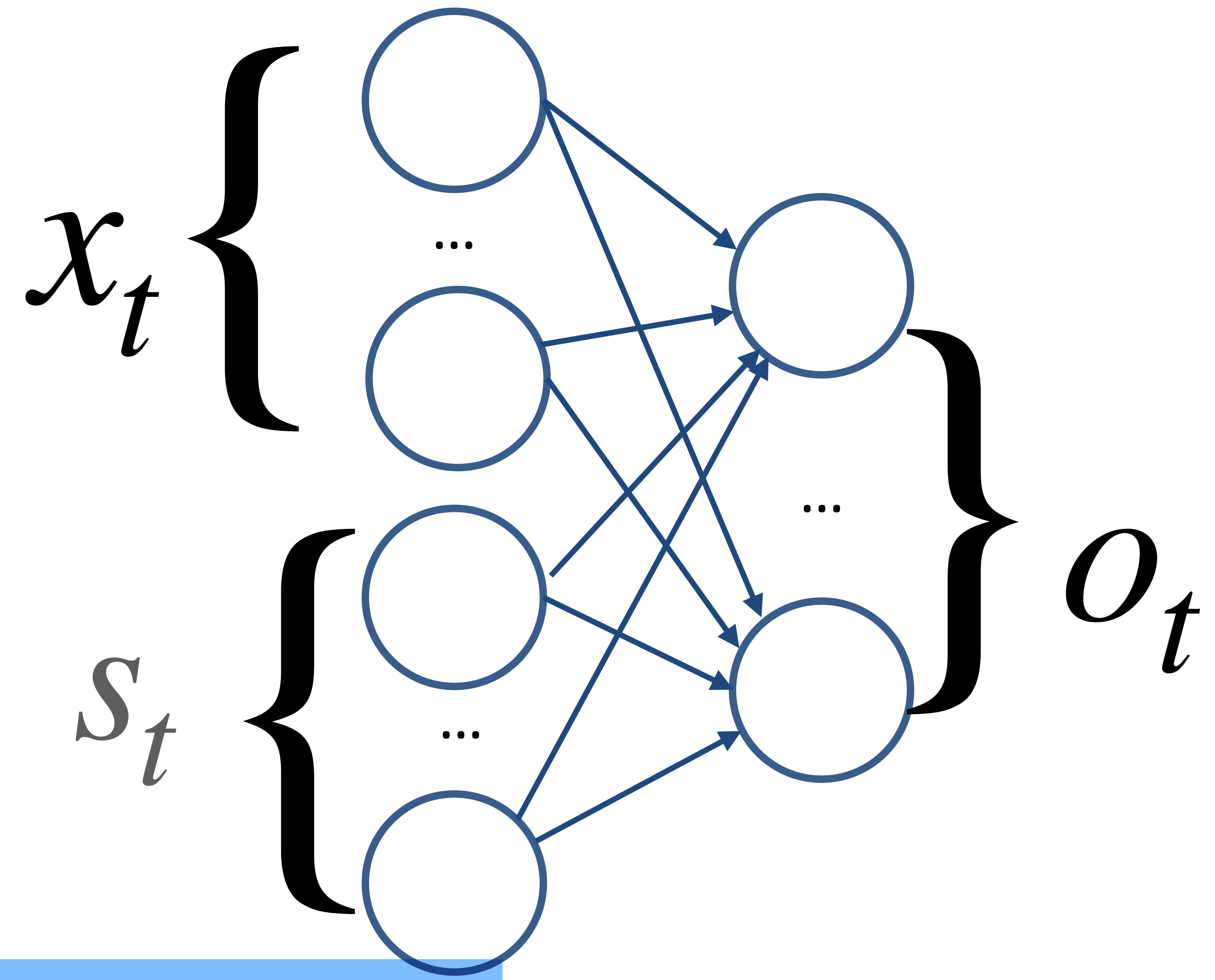
Recurrent connection: the output hidden feature vector (state) of each step is connected to the next step



RNN: Process sequence step by step

At each step t: combine current tilmestep x_t (feature vector of input sequence at tilmestep t) and historical information, i.e., state s_t (feature vector) to generate o_t





$$o_t = activation(W \cdot x_t + U \cdot s_t + b)$$

Shared over time

$s_t = o_{t-1}$

s_1 : zero feature vector

A simple RNN

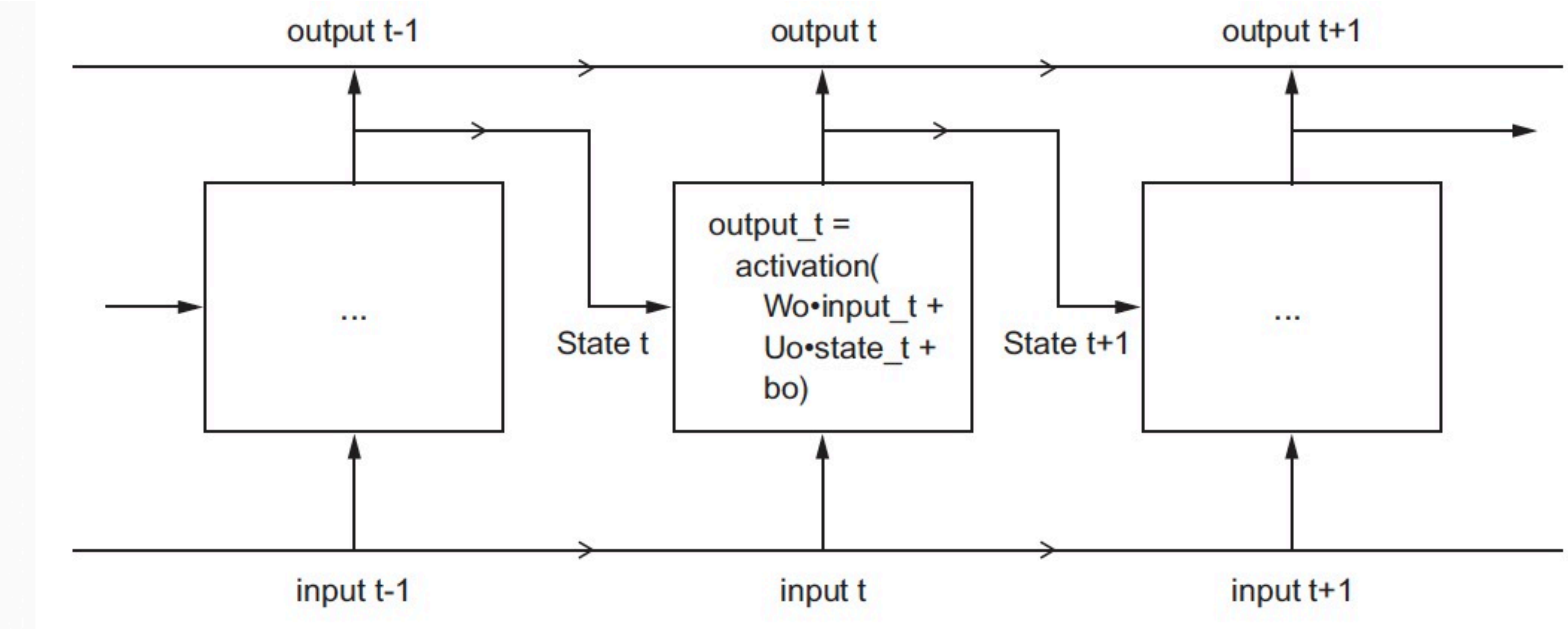
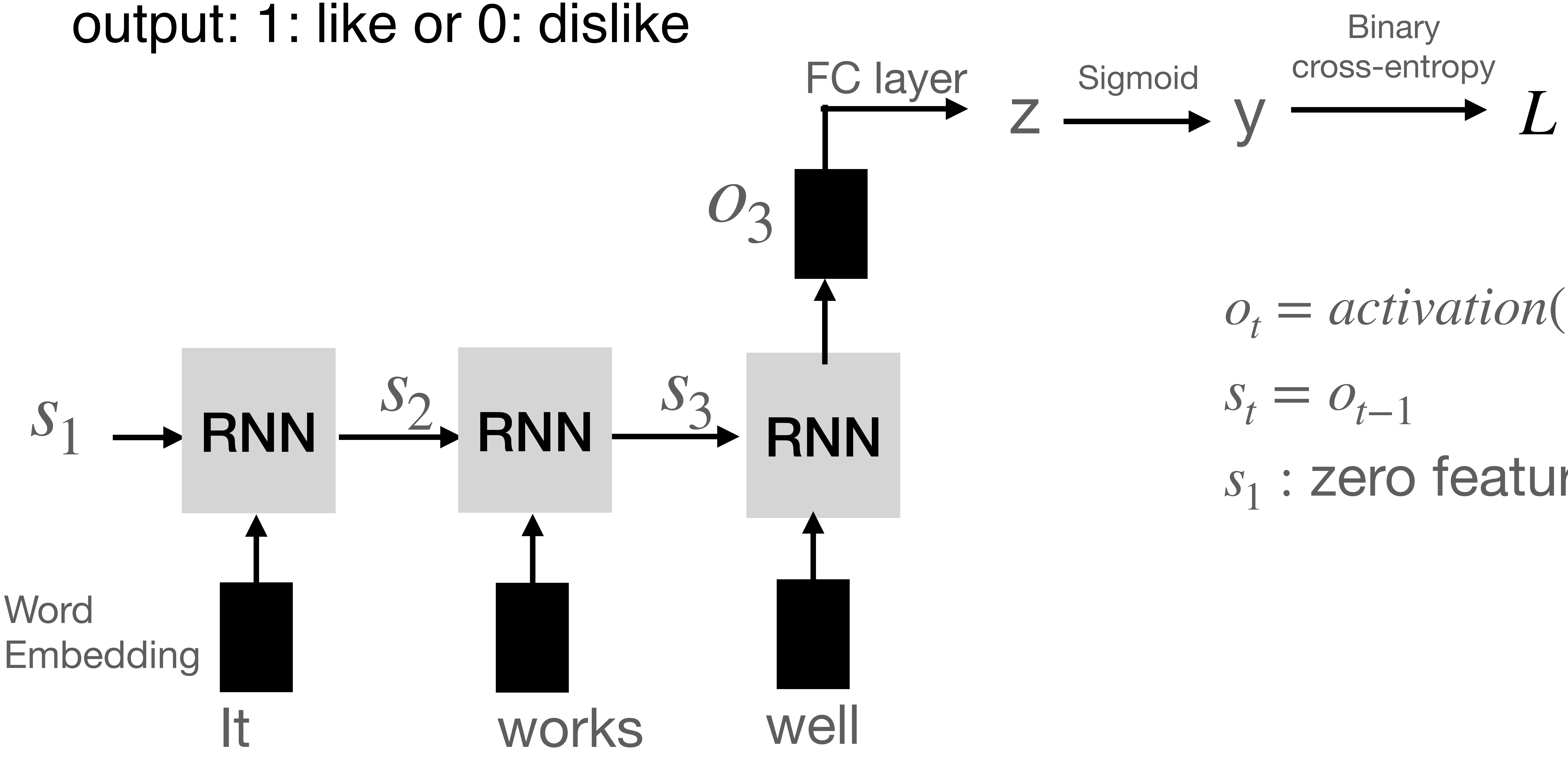


Figure 6.13 in Deep learning with python by Francois Chollet

Example: RNN for sentiment analysis

Input: review (a piece of text) e.g.,“It works well”
output: 1: like or 0: dislike



$o_t = activation(W \cdot x_t + U \cdot s_t + b)$

$s_t = o_{t-1}$

s_1 : zero feature vector

Recap of Chain rule

$$Given\ z = g(u) \quad u = f(x) \quad \Rightarrow \quad \frac{dz}{dx} = \frac{dz}{du}\frac{du}{dx}$$

$$Example : z = \sin(x^2)$$

$$z = sin(u)$$

$$u = x^2$$

$$\frac{dz}{du} = \cos(u)$$

$$\frac{dz}{dx} = \frac{dz}{du}\frac{du}{dx} = 2x\cos(u)$$

Recap of Multi-layer perceptron: Function composition

Forward prediction

$$z = f(s) = sigmoid(s) = \frac{1}{1+e^{-s}}$$

$$\uparrow$$

$$s = \sum_{j=0}^{p} w_j u_j$$

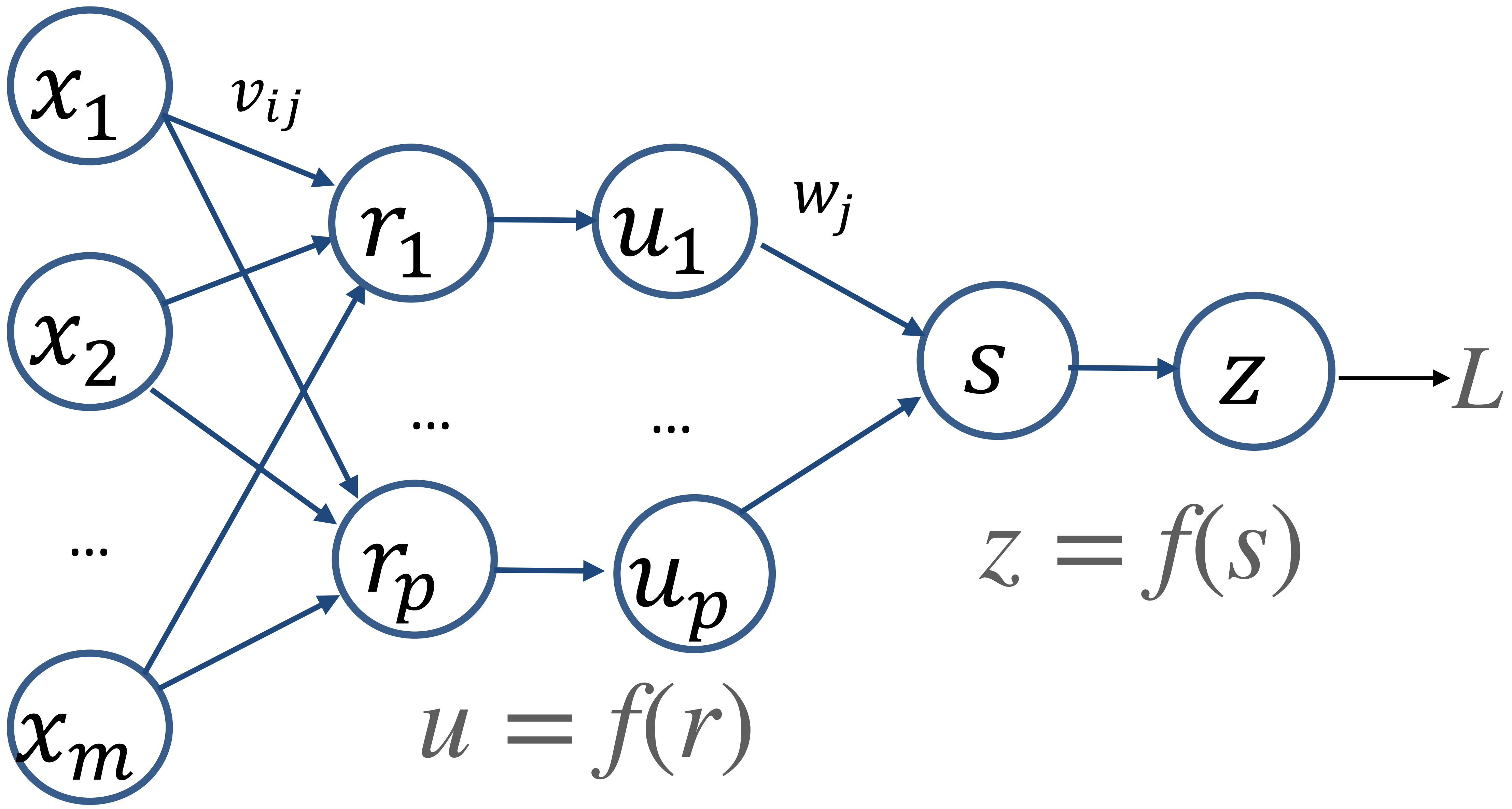
$$\uparrow$$

$$u_j = f(r_j) = sigmoid(r_j) = \frac{1}{1+e^{-r_j}}$$

$$\uparrow$$

$$r_j = \sum_{i=0}^{m} x_i v_{ij}$$

$$x \rightarrow r \rightarrow u \rightarrow s \rightarrow z$$



$f:$ *activation functions*

Recap of Multi-layer perceptron: Chain rule

Backward propagation:

$$x \rightarrow r \rightarrow u \rightarrow s \rightarrow z$$

$$\frac{\partial L}{\partial v_{ij}} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial s}\frac{\partial s}{\partial u_j}\frac{\partial u_j}{\partial r_j}\frac{\partial r_j}{\partial v_{ij}}$$

$$s = \sum_{j=0}^{p} u_j w_j$$

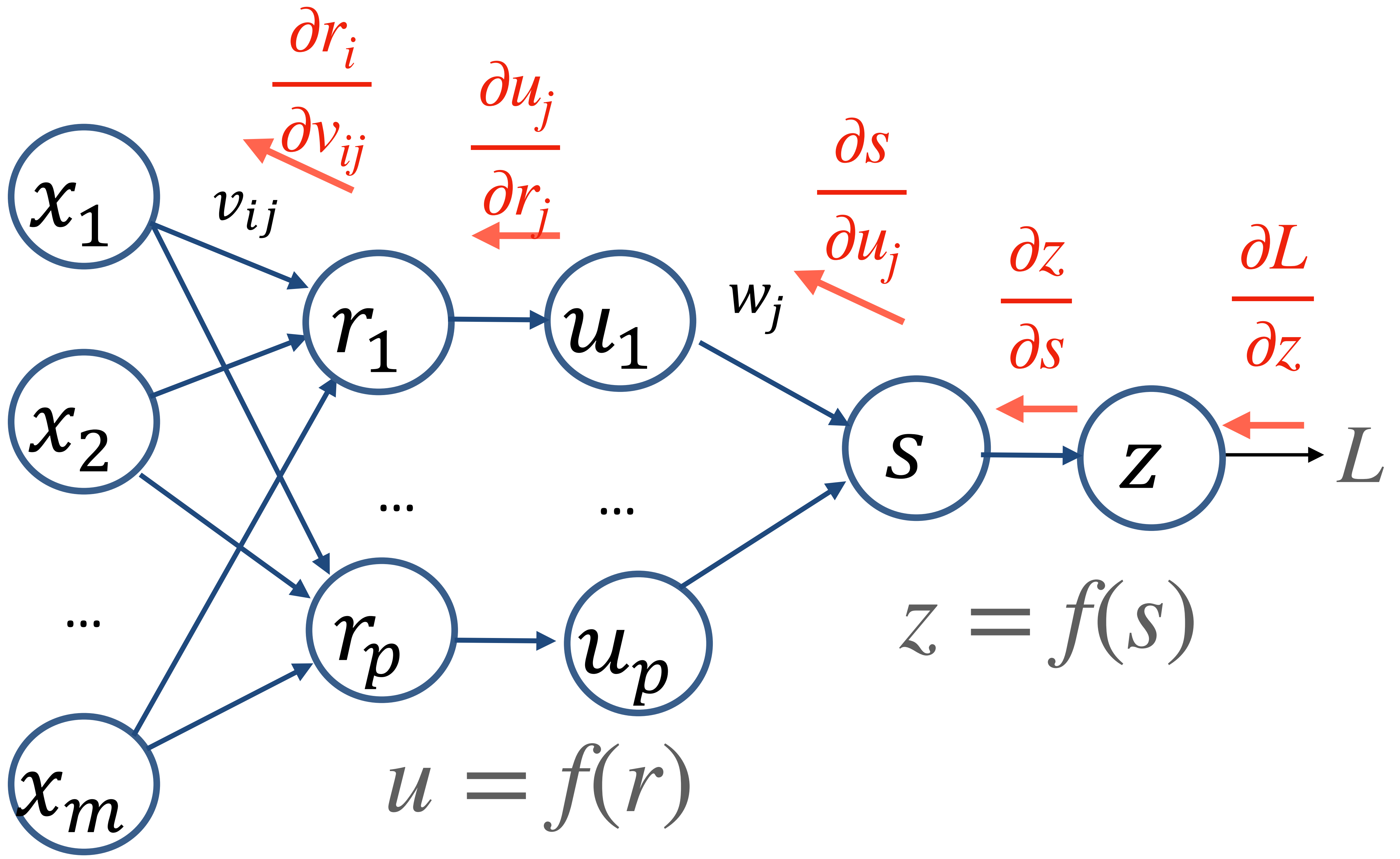
$$\uparrow$$

$$u_j = sigmoid(r_j) = \frac{1}{1+e^{-r_j}}$$

$$\uparrow$$

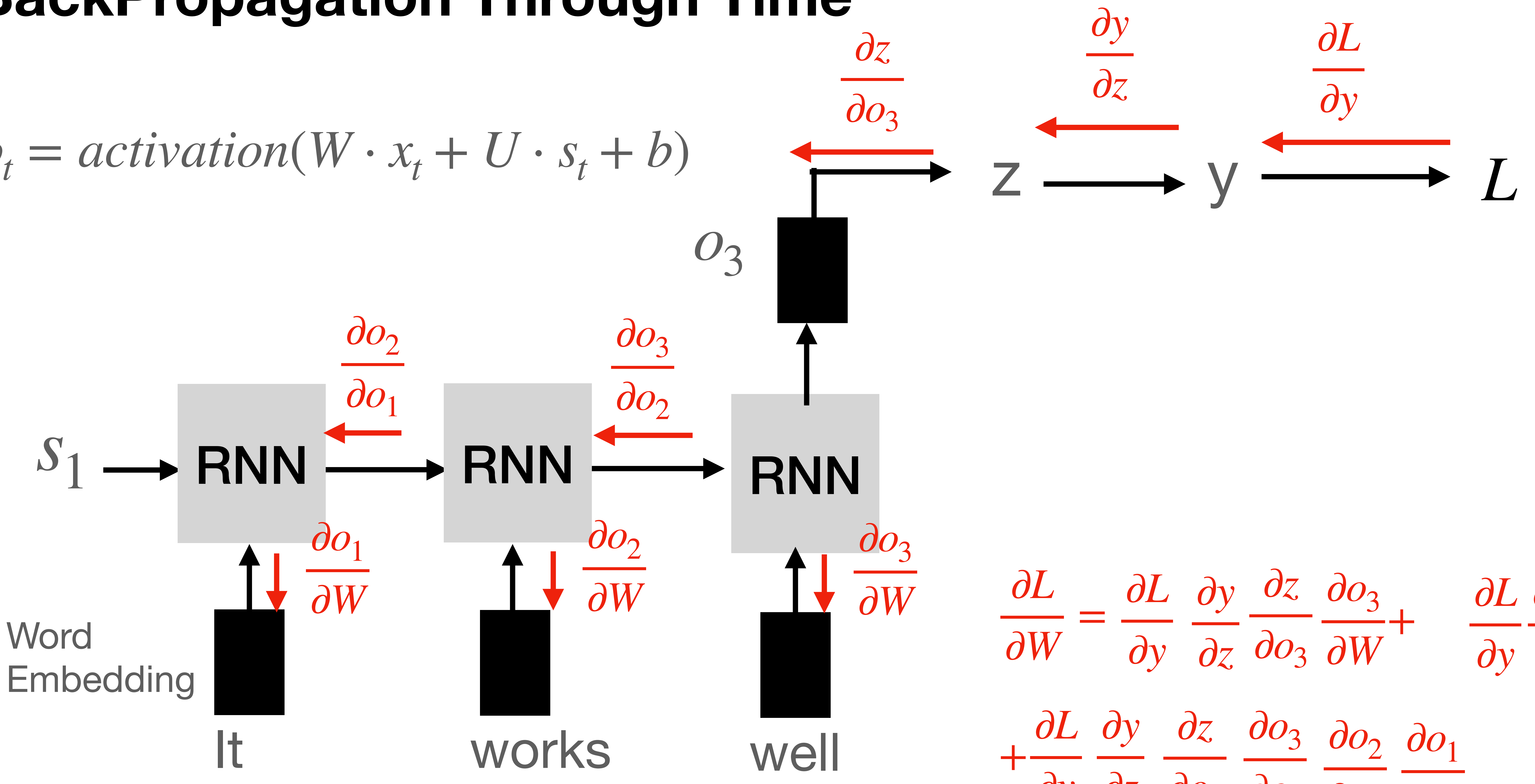
$$r_j = \sum_{i=0}^{m} x_i v_{ij}$$

Forward



BackPropagation Through Time

$$o_t = activation(W \cdot x_t + U \cdot s_t + b)$$



$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial z}\frac{\partial z}{\partial o_3}\frac{\partial o_3}{\partial W} + \frac{\partial L}{\partial y}\frac{\partial y}{\partial z}\frac{\partial z}{\partial o_3}\frac{\partial o_3}{\partial o_2}\frac{\partial o_2}{\partial W} + \frac{\partial L}{\partial y}\frac{\partial y}{\partial z}\frac{\partial z}{\partial o_3}\frac{\partial o_3}{\partial o_2}\frac{\partial o_2}{\partial o_1}\frac{\partial o_1}{\partial W}$$

A special RNN: Long short-term memory (LSTM)

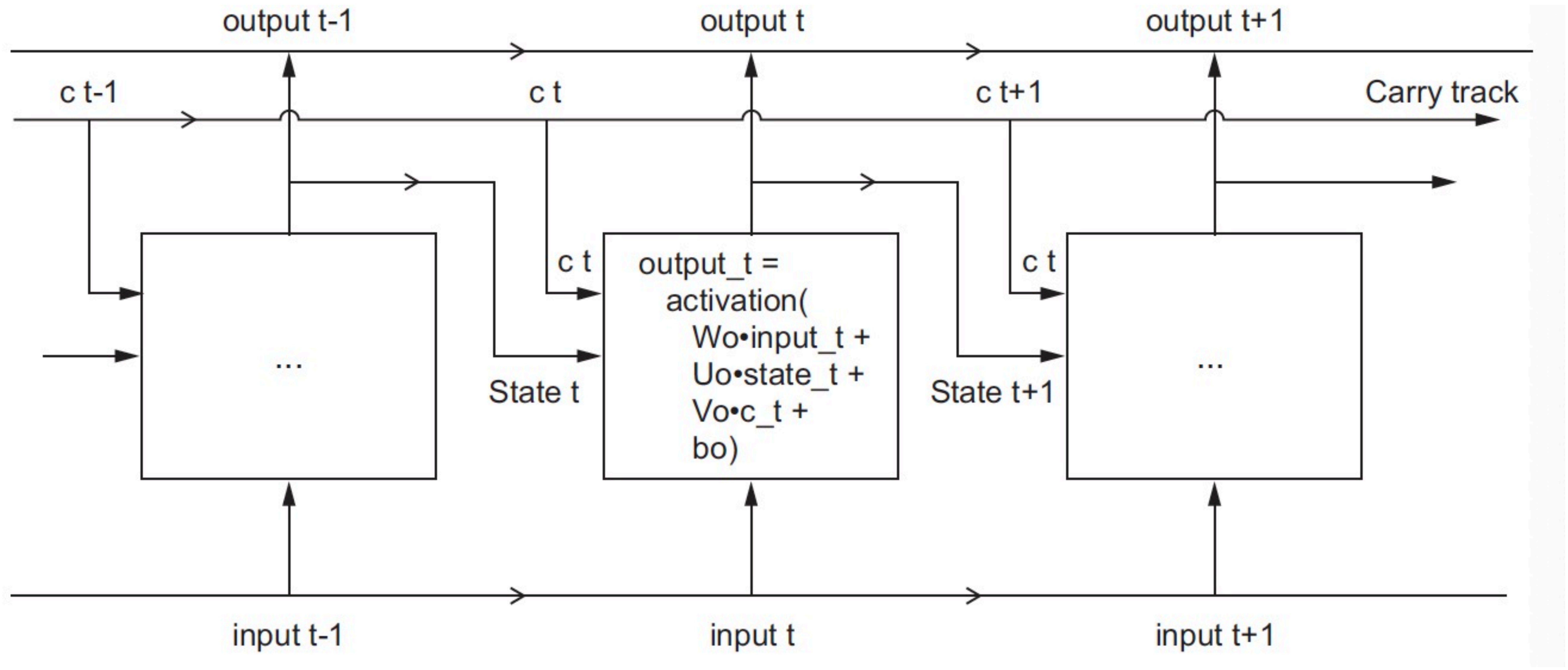


Figure 6.14 in Deep learning with python by Francois Chollet

A special RNN: Long short-term memory (LSTM)

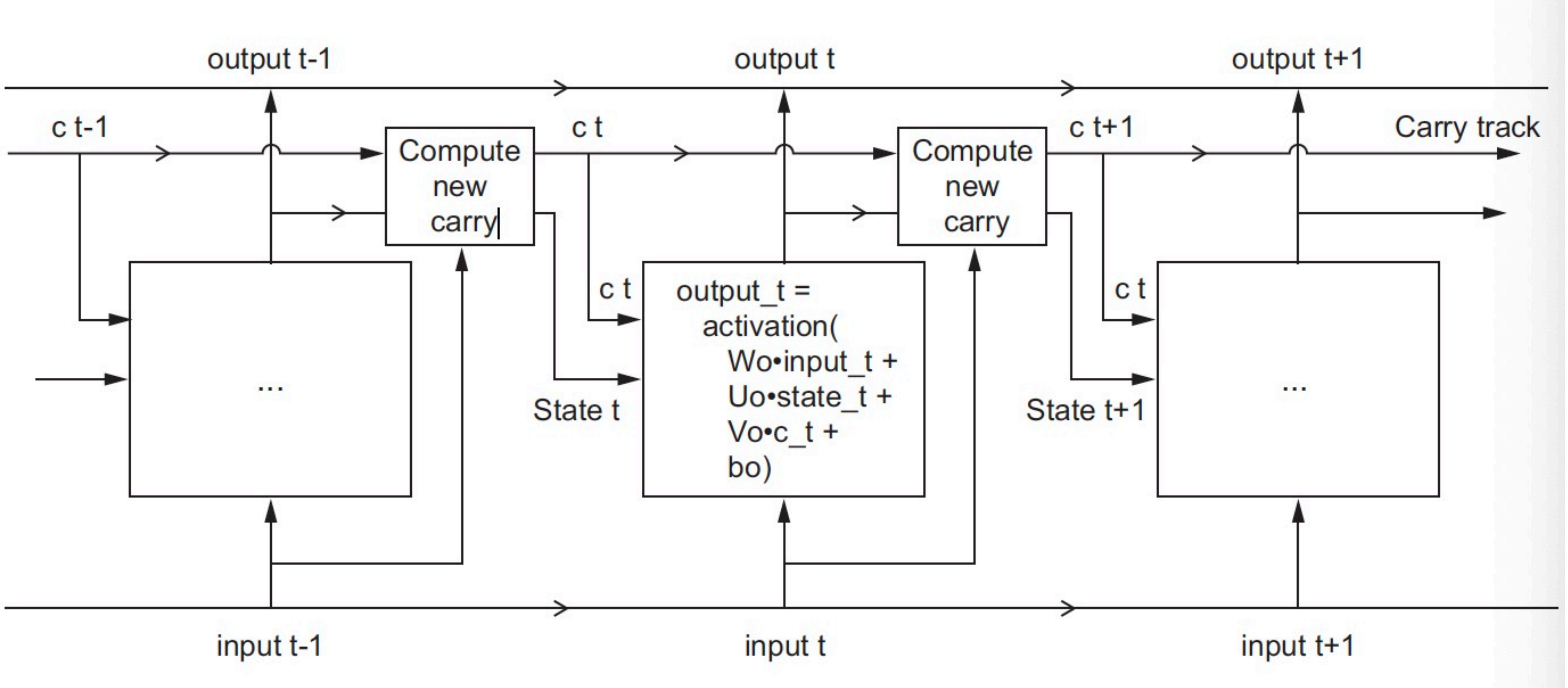
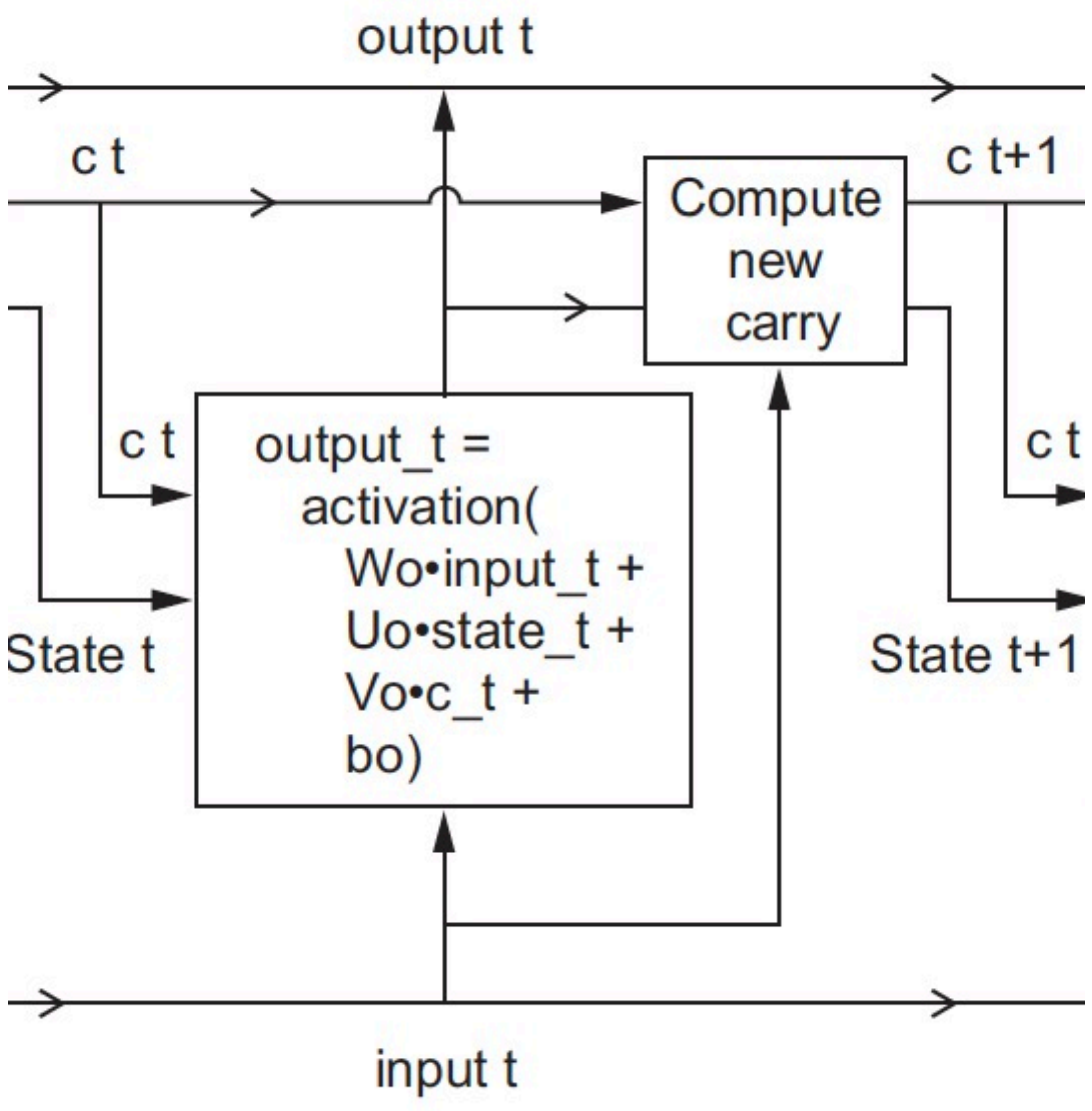


Figure 6.15 in Deep learning with python by Francois Chollet

A special RNN: Long short-term memory (LSTM)



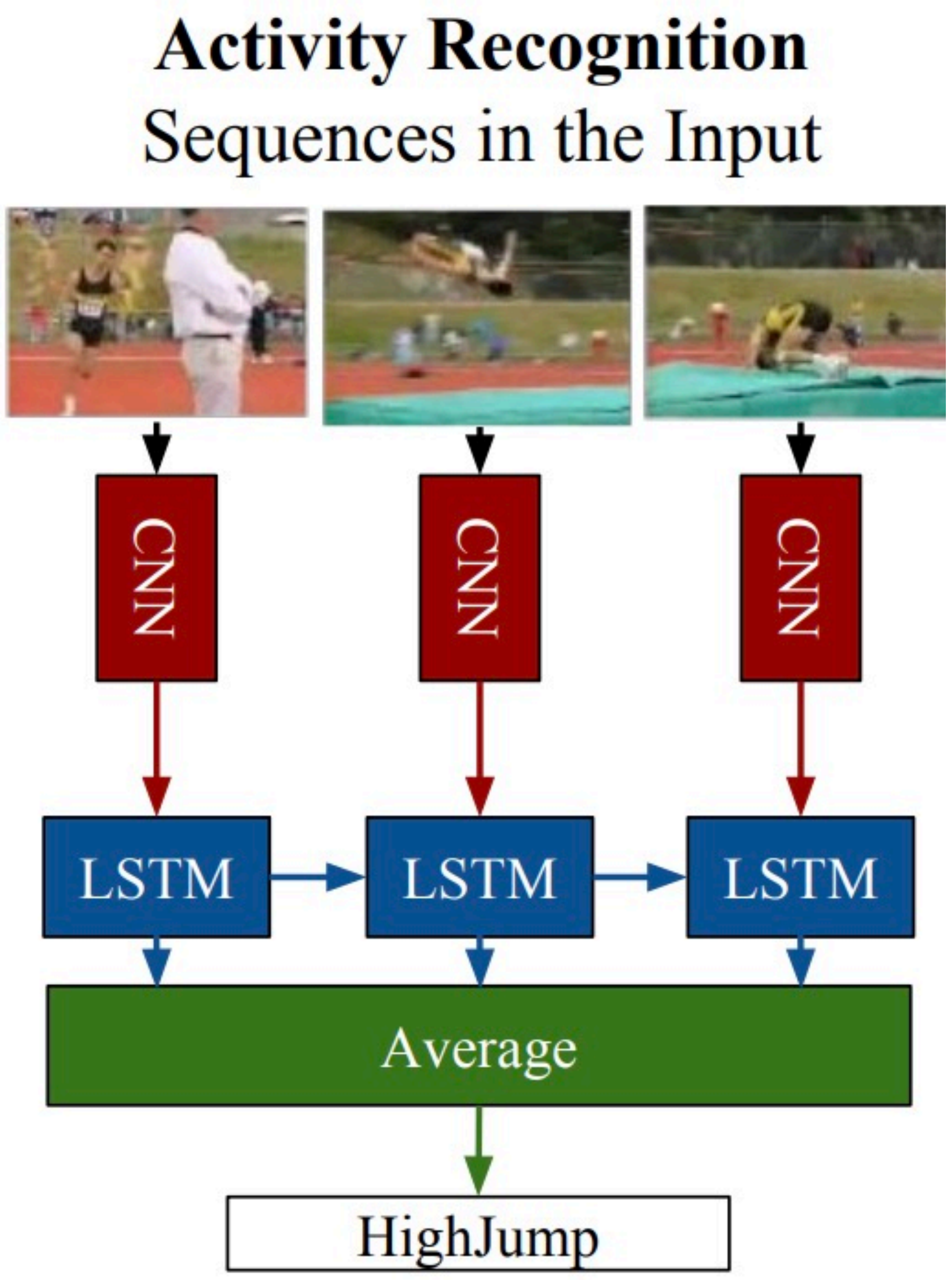
Forget gate: $f_t = sigmoid(W_f \cdot x_t + U_f \cdot s_t + b_f)$

Input gate: $i_t = sigmoid(W_i \cdot x_t + U_i \cdot s_t + b_i)$

$$\tilde{c}_{t+1} = tanh(W_c \cdot x_t + U_c \cdot s_t + b_c)$$

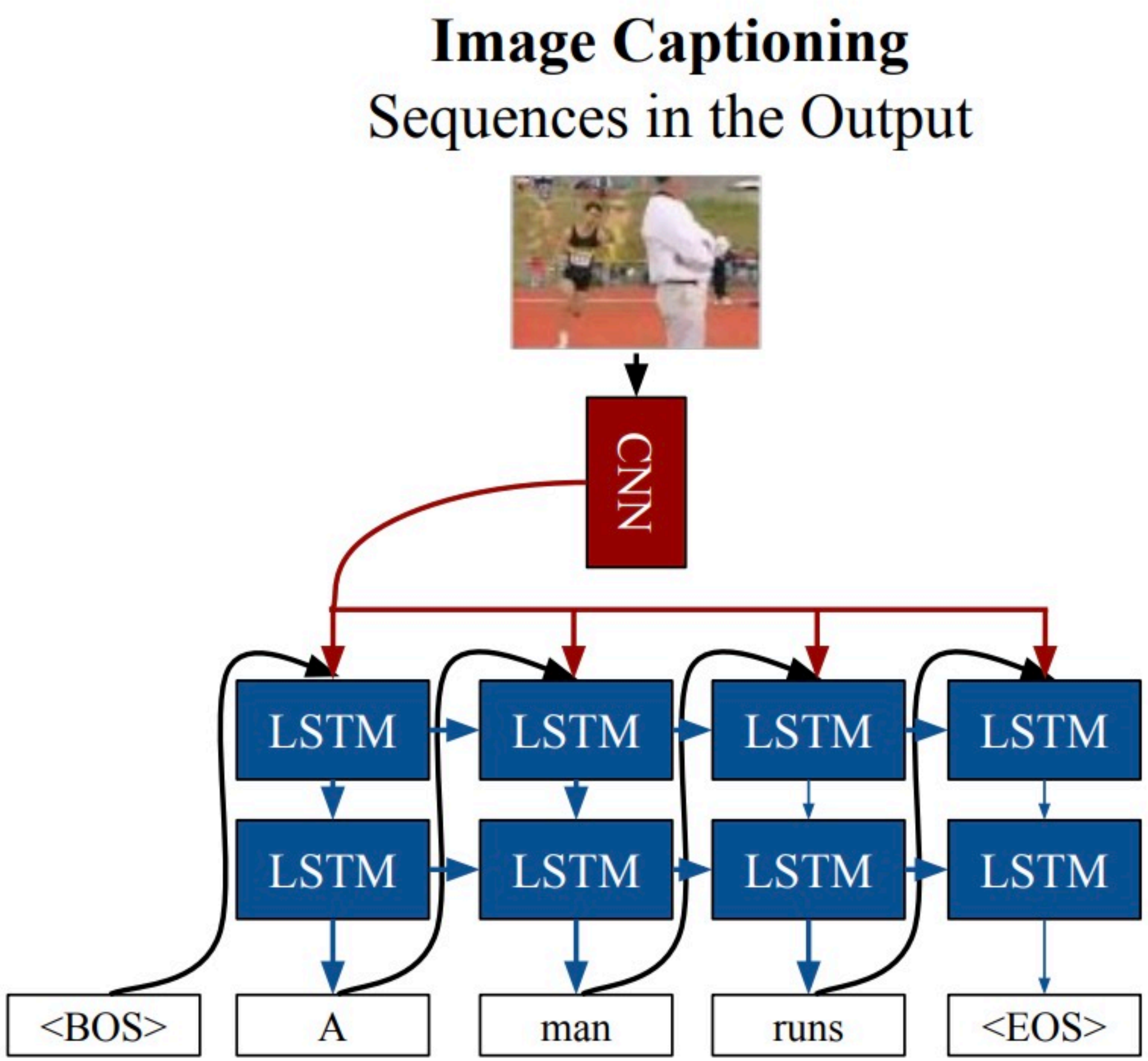
$$c_{t+1} = c_t * f_t + \tilde{c}_{t+1} * i_t$$

Applications



Jeff Donahue et al, Long-term Recurrent Convolutional Networks for Visual Recognition and Description

Applications



Jeff Donahue et al, Long-term Recurrent Convolutional Networks for Visual Recognition and Description

Summary

- How does temporal convolution work?
- How does dilated casual convolution work?
- How does recurrent neural network work?
- How does LSTM work?

Next: Graph Convolution Network (Guest lecture)