Extra Class

Introduction to LSTM

Nguyen Quoc Thai

AI VIET NAM
@aivietnam.edu.vn

CONTENT

1 – Recurrent Neural Network

Language Model

- Estimate the probability of an upcoming words

P(w|h) = P(school|i,go,to)

w: token as word "school"

h: history tokens as "i,go,to"

$$P(w|h) = \frac{count(hw)}{count(h)}$$

$$P(school|i, go, to) = \frac{count(i, go, to, school)}{count(i, go, to)}$$

1 – Recurrent Neural Network

Language Model

- The probability of a word depends only on some previous words
- N-gram model with N = {1, 2,…}

$$P(w_{1:n}) = \prod_{i=1}^{n} P(w_i | w_{i-N+1:i-1})$$

$$P(w_i | w_{1:i-1}) = P(w_i | w_{i-N+1:i-1})$$

1 – Recurrent Neural Network

Language Model

- N = 1
- Unigram Model (1 – gram)

$$P(w_{1:n}) = \prod_{i=1}^{n} P(w_i | w_{i-N+1:i-1}) = \prod_{i=1}^{n} P(w_i)$$

P("i,go,to,school") → = P(i).P(go|i).P(to|i,go).P(school|i,go,to)

= P(i).P(go).P(to).P(school)

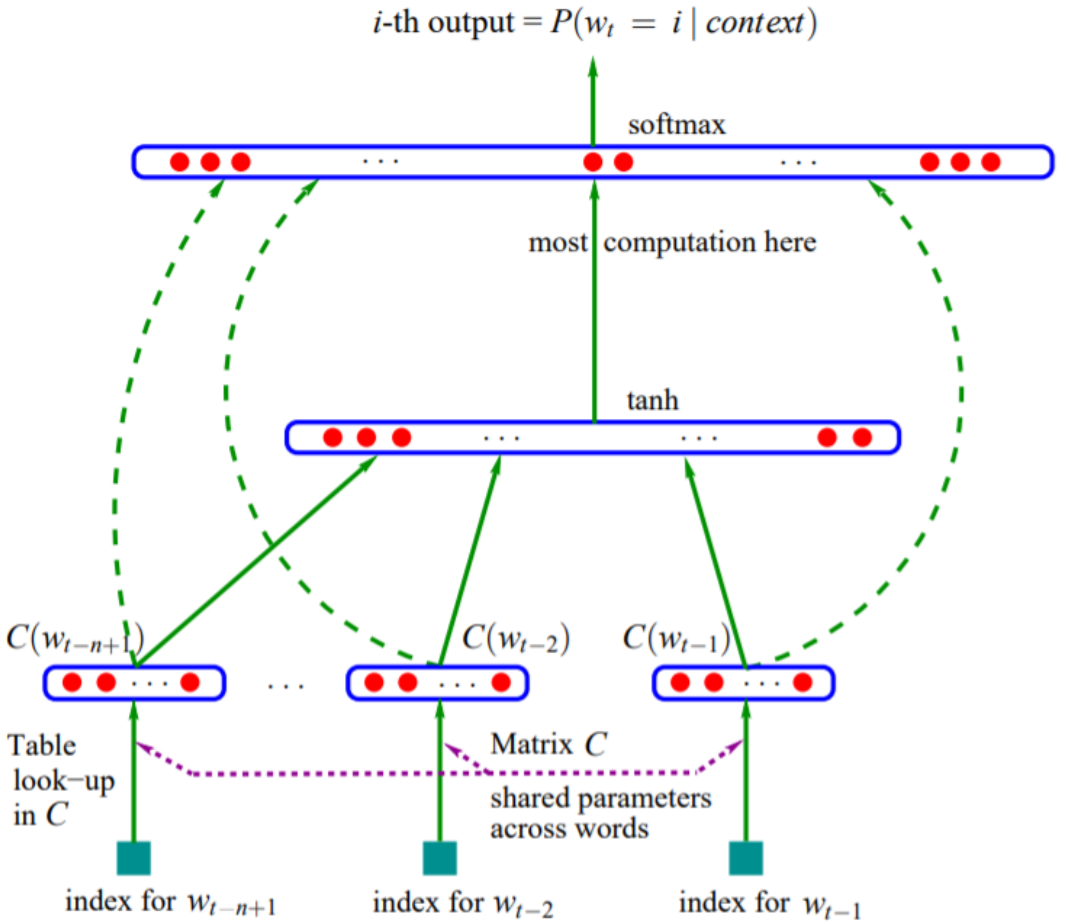
1 – Recurrent Neural Network

From Neural Network to Recurrent Neural Network

❖ A neural Probabilistic Language Model

“trăm năm trong cõi người ta”

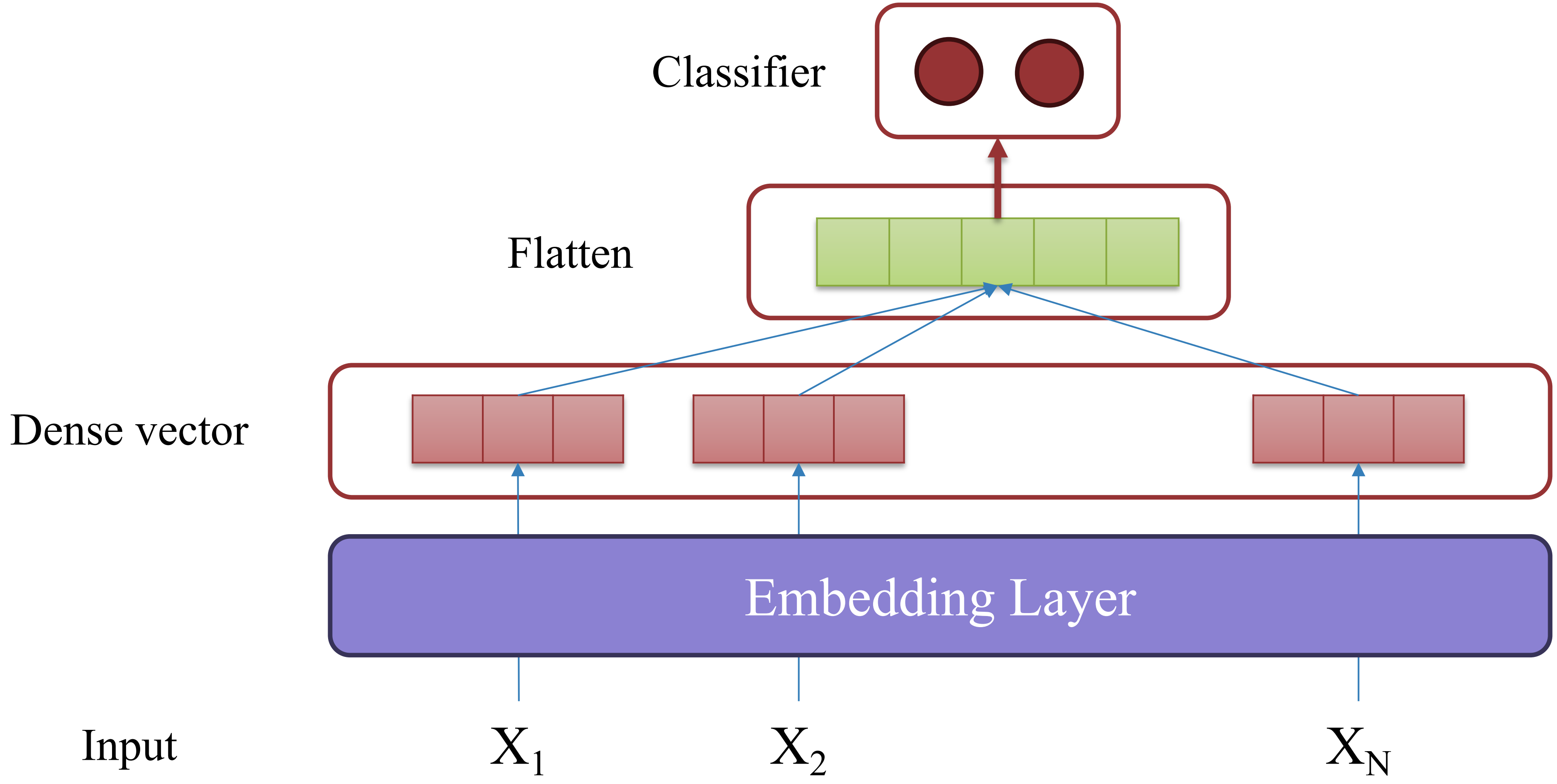
Source	Target
trăm	năm
…	…
trăm năm	trong
…	…
trăm năm trong	cõi
…	…
trăm năm trong cõi người	ta



1 – Recurrent Neural Network

From Neural Network to Recurrent Neural Network

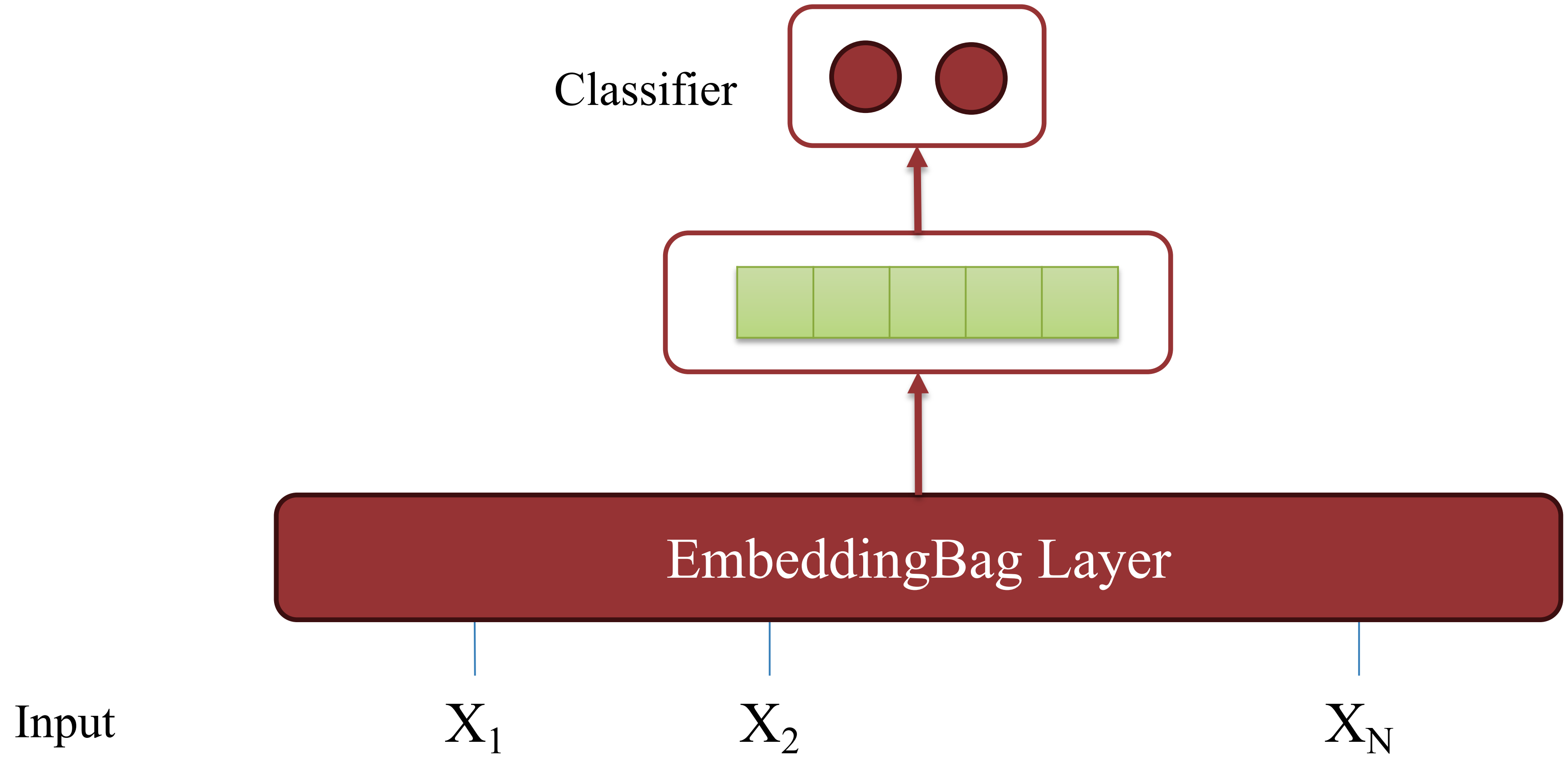
❖ Text Classification using Neural Network



1 – Recurrent Neural Network

From Neural Network to Recurrent Neural Network

- Text Classification using Neural Network



1 – Recurrent Neural Network

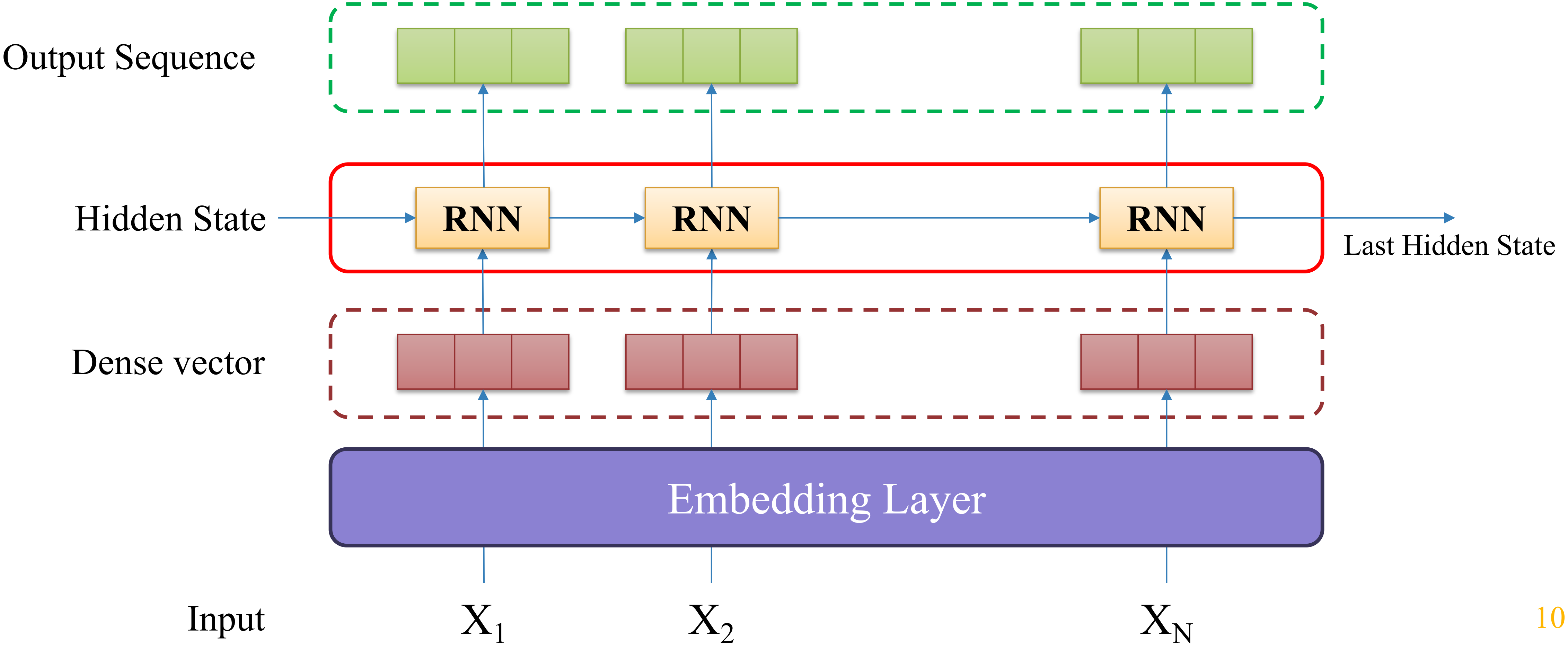
From Neural Network to Recurrent Neural Network

- Models need to learn the context in which words appear
- Need better network architectures…

RNNs for Sequence

1 – Recurrent Neural Network

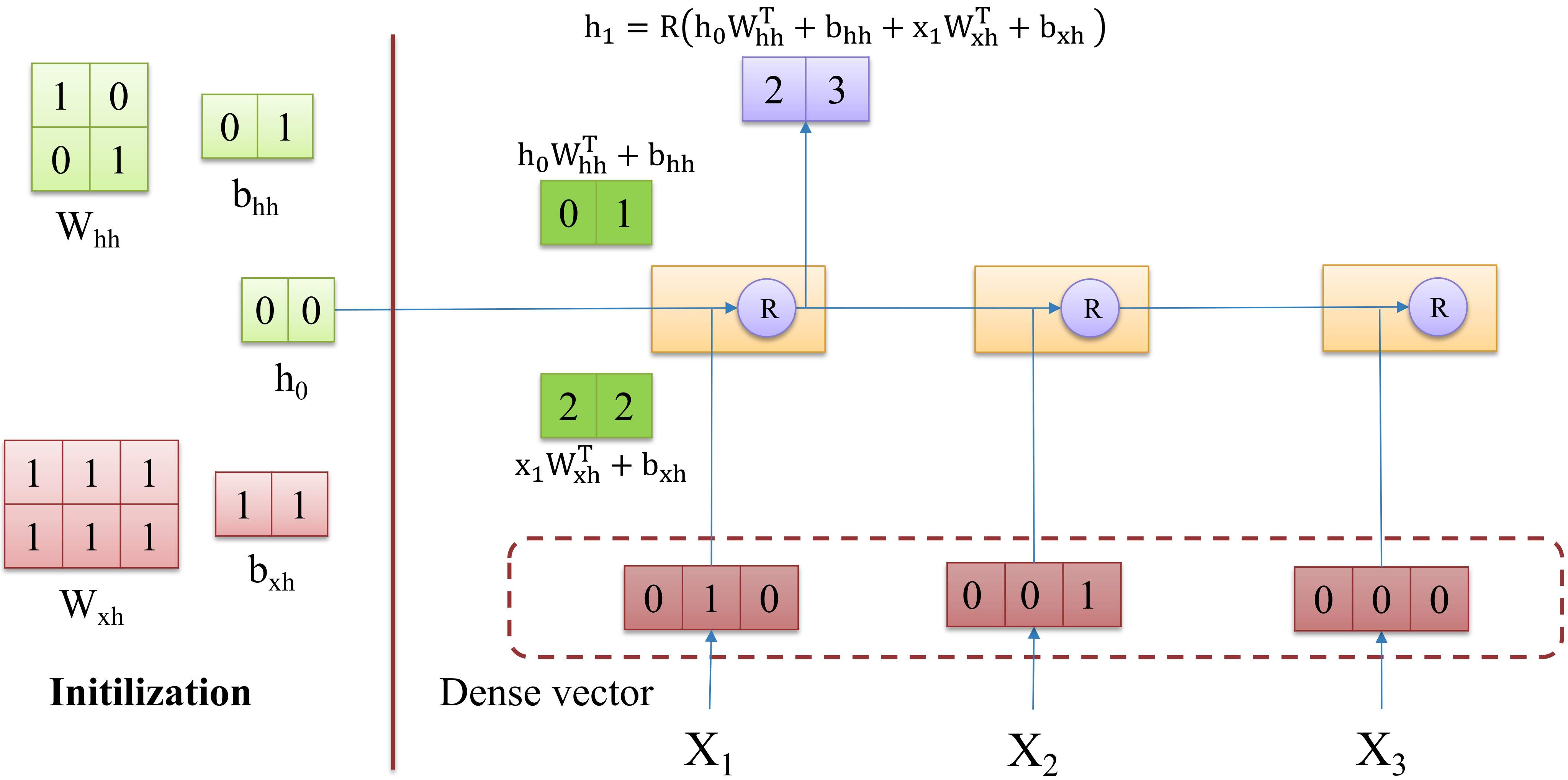
Recurrent Neural Network (RNN)



AI VIET NAM
@aivietnam.edu.vn

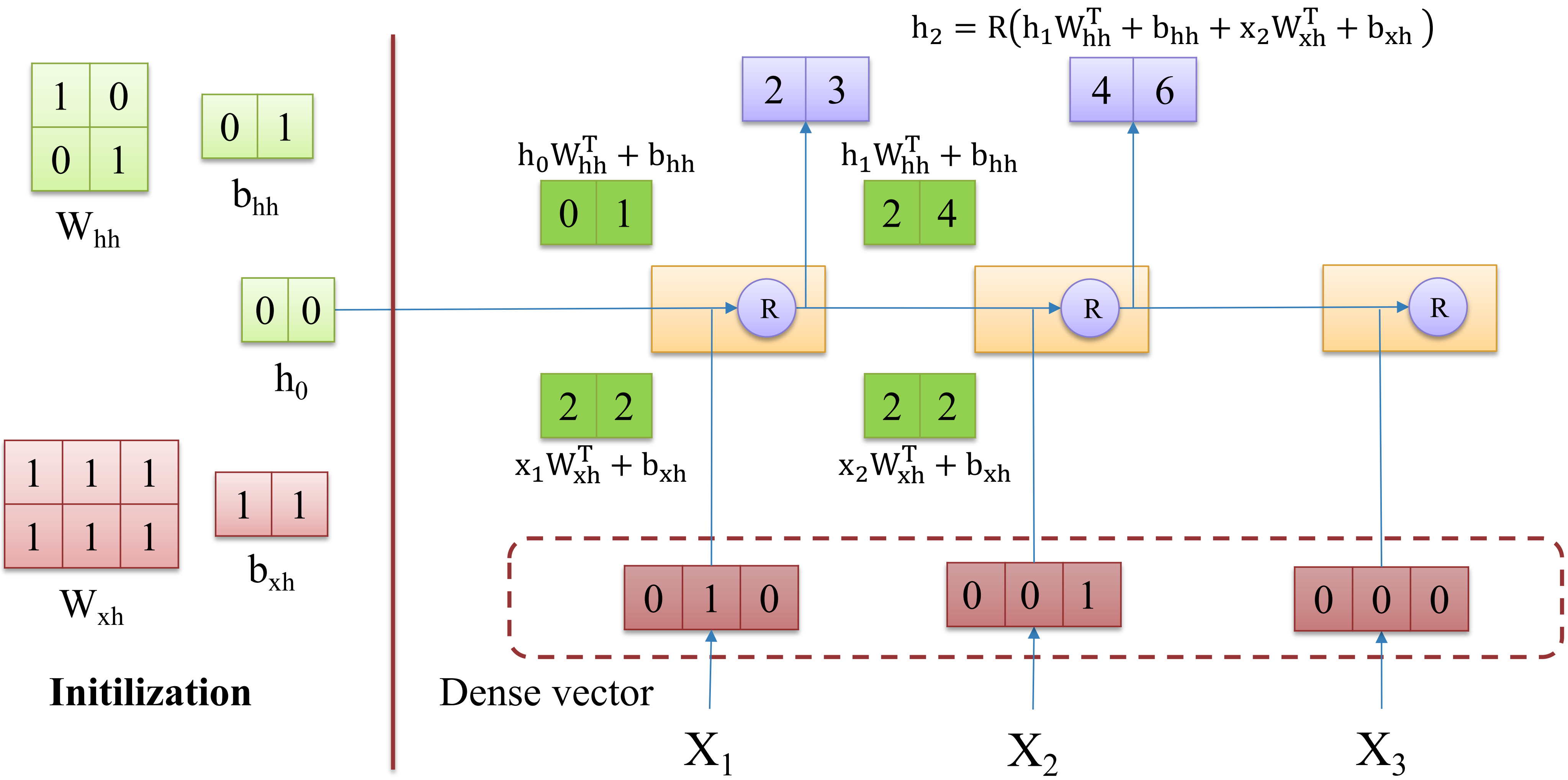
1 – Recurrent Neural Network

Recurrent Neural Network (RNN)



1 – Recurrent Neural Network

Recurrent Neural Network (RNN)

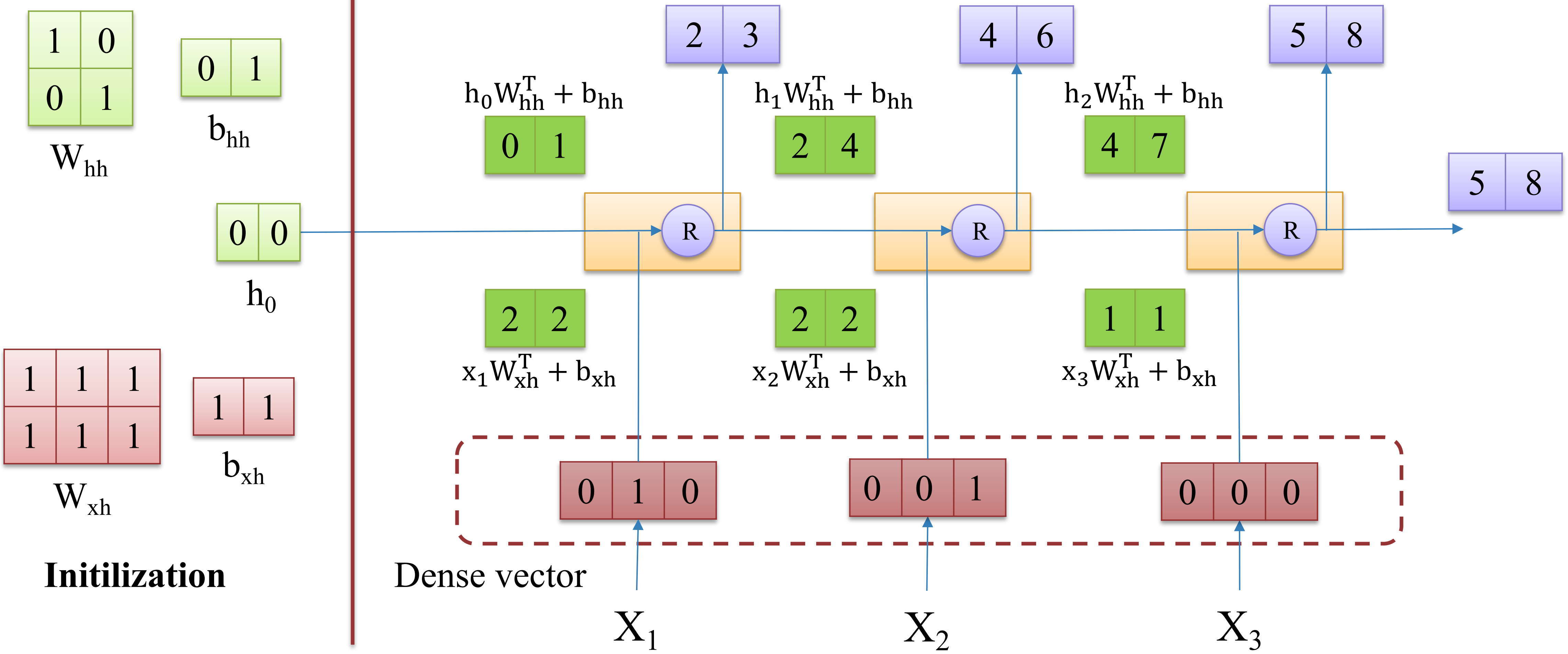


AI VIET NAM
@aivietnam.edu.vn

1 – Recurrent Neural Network

Recurrent Neural Network (RNN)

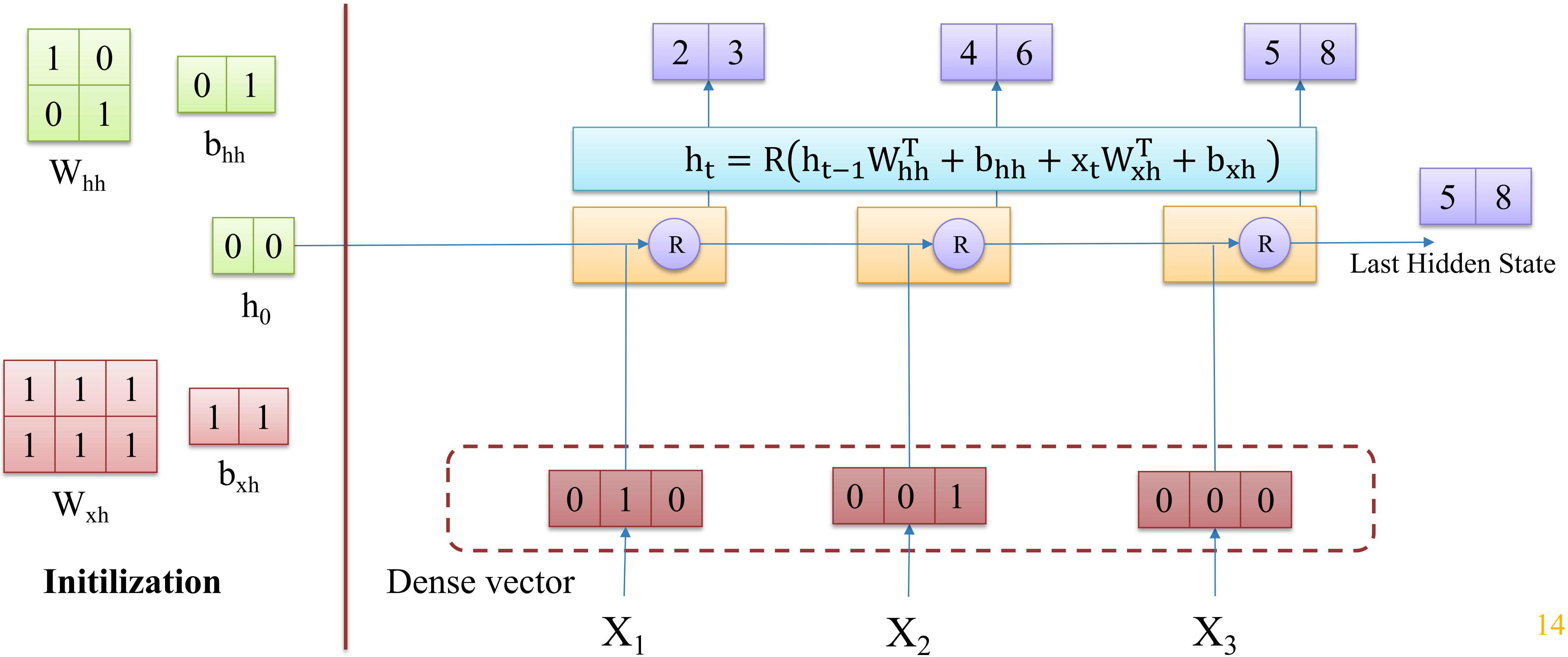
$$h_3 = R\left(h_2 W_{hh}^T + b_{hh} + x_3 W_{xh}^T + b_{xh}\right)$$



1 – Recurrent Neural Network

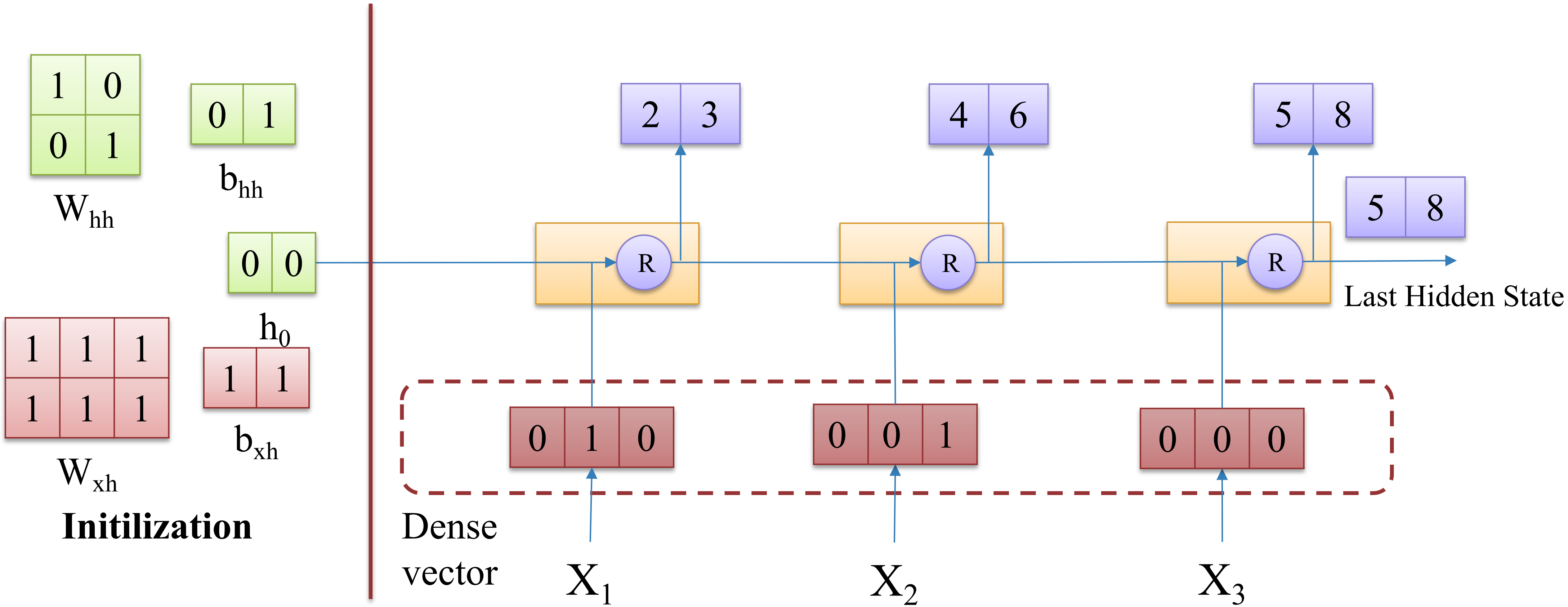
AI VIET NAM
@aivietnam.edu.vn

Recurrent Neural Network (RNN)



1 – Recurrent Neural Network

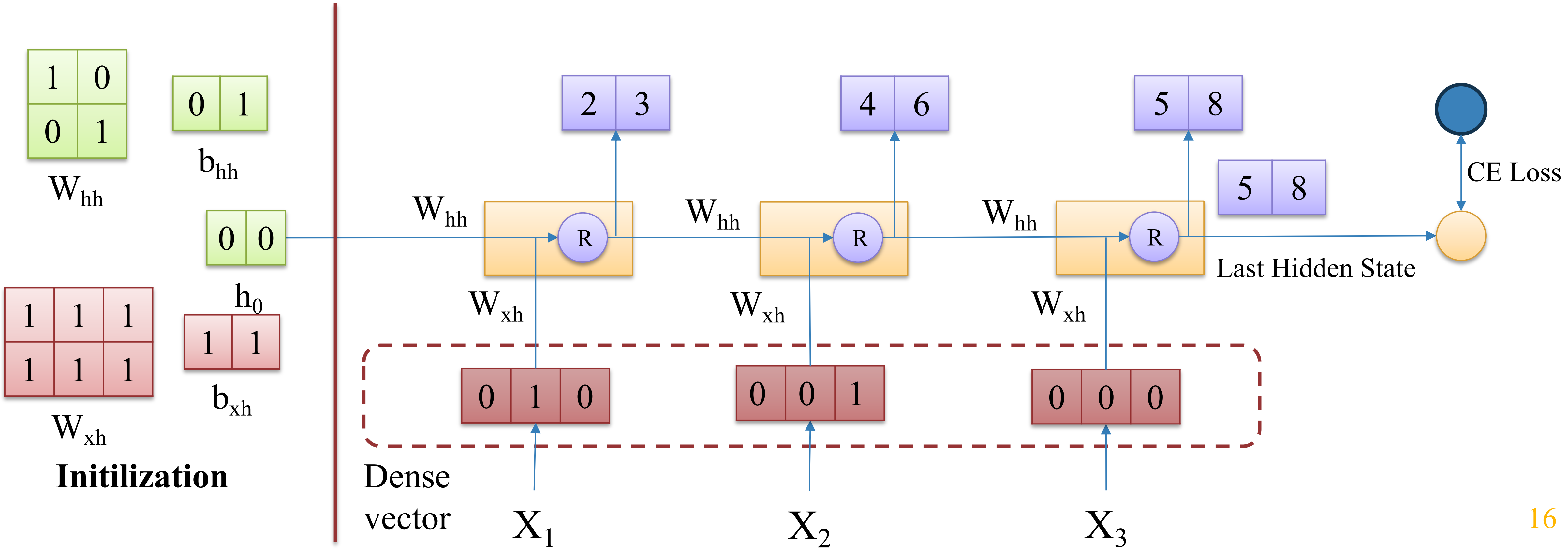
Loss Function



1 – Recurrent Neural Network

Loss Function

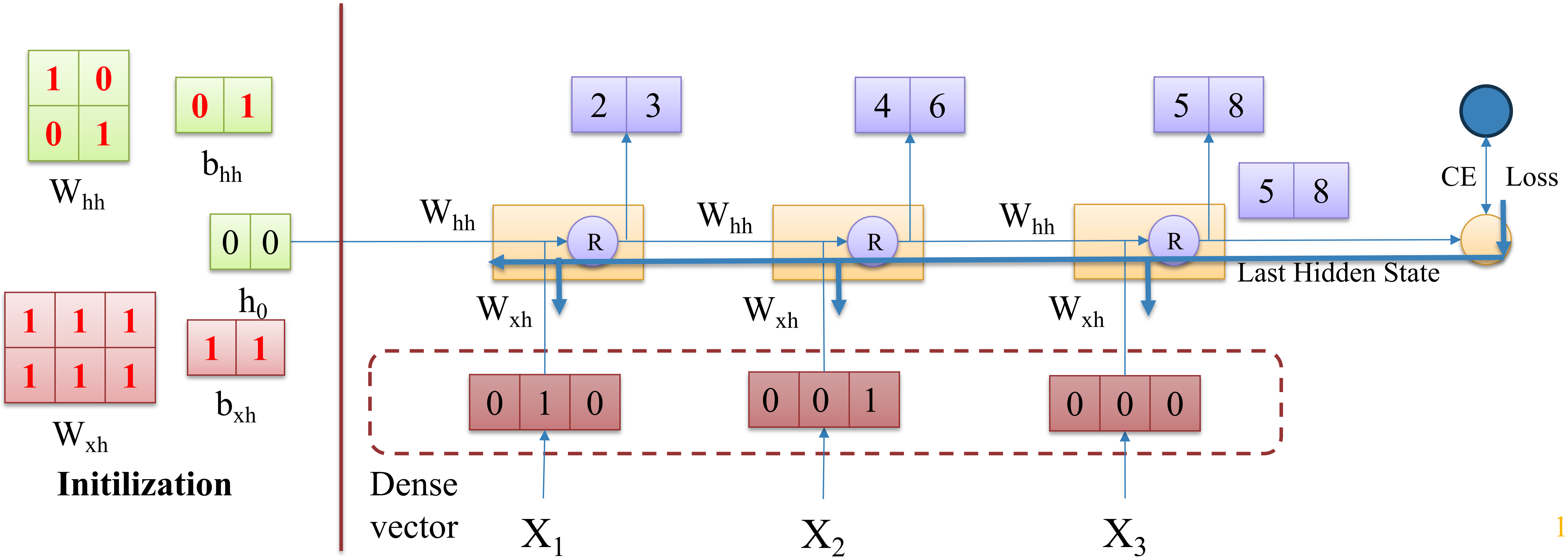
❖ Loss Function for Text Classification



1 – Recurrent Neural Network

Loss Function

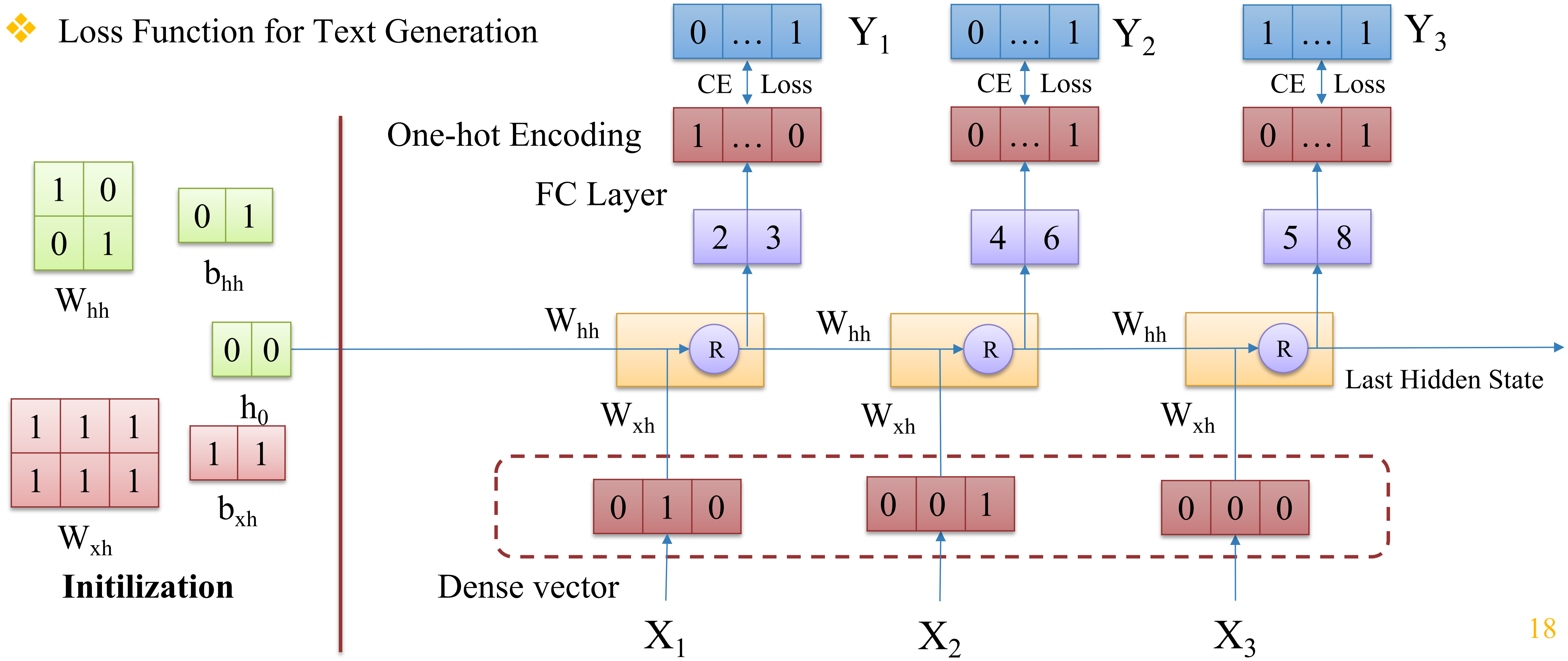
❖ Loss Function for Text Classification



1 – Recurrent Neural Network

Loss Function

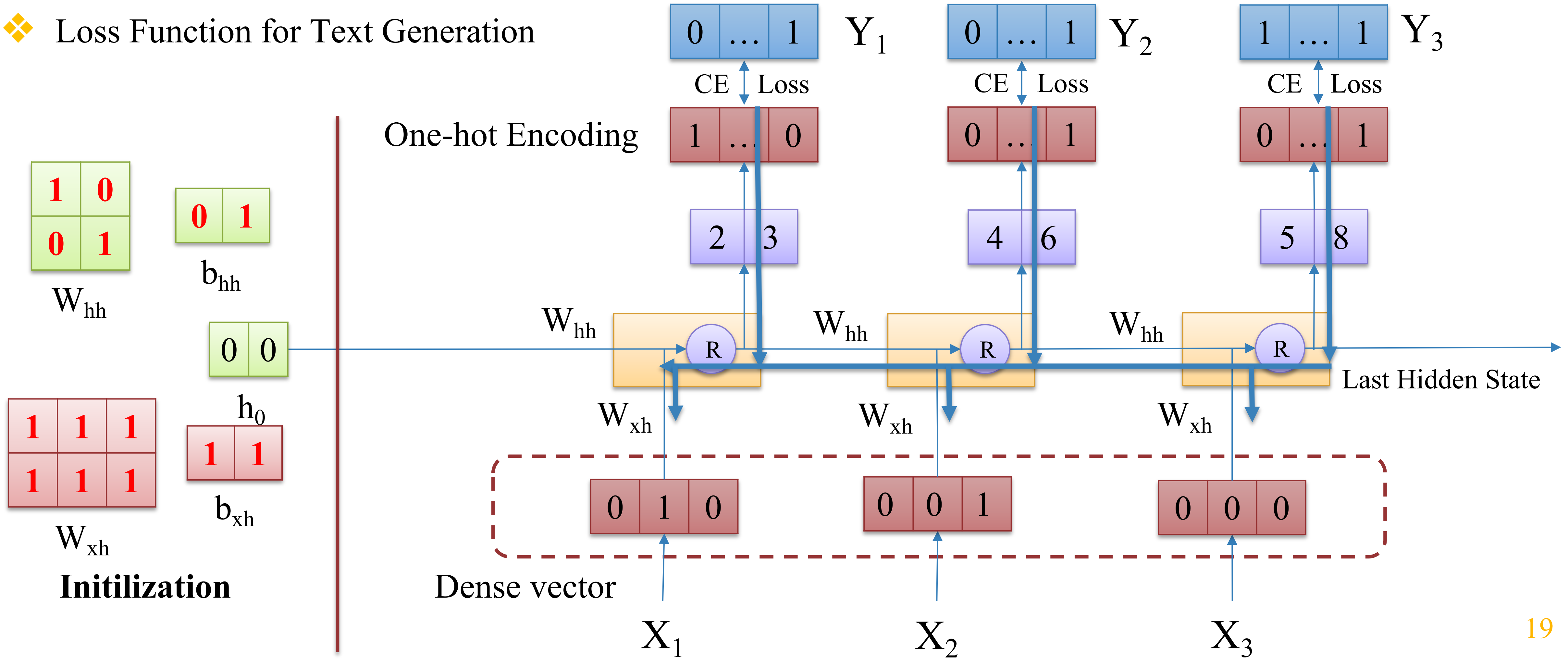
❖ Loss Function for Text Generation



1 – Recurrent Neural Network

Loss Function

- Loss Function for Text Generation



1 – Recurrent Neural Network

Pytorch - Demo

```
batch_size = 1
seq_length = 3
embedding_dim = 3

input = torch.randint(
    high=2,
    size=(batch_size, seq_length, embedding_dim),
    dtype=torch.float32
)

input
```

```
tensor([[[0., 1., 0.],
         [0., 0., 1.],
         [0., 0., 0.]]])
```

```
embedding_dim = 3
hidden_size = 2
activation = 'relu'

rnn_layer = nn.RNN(
    input_size=embedding_dim,
    hidden_size=hidden_size,
    nonlinearity=activation,
    batch_first=True
)
```

```
output, hn = rnn_layer(input)
```

```
output
```

```
tensor([[[2., 3.],
         [4., 6.],
         [5., 8.]]], grad_fn=<TransposeBackward1>)
```

```
hn
```

```
tensor([[[5., 8.]]], grad_fn=<StackBackward0>)
```

2 – Text Classification

NTC-SCV Dataset

❖ **NTC-CSV Dataset**

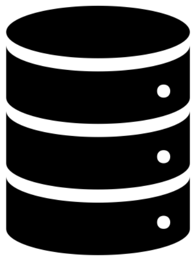
➢ Sentiment Analysis

Positive Example	Negative Example
Mình được 1 cô bạn giới_thiệu đến đây , tìm địa_chỉ khá dễ . Menu nước uống chất khỏi nói . Mình muốn cũng đc 8 loại nước ở đây , món nào cũng ngon và bổ_dưỡng cả .	Quán chế_biến đồ_ăn lâu , Cá_Sapa nướng ướp rất dở , sò Lông ko tươi , nước_chấm ko ngon\n Tóm_lại sẽ ko bao_giờ ghé nữa , ăn_dở mà uổng tiền
Mỗi lần thèm trà sữa là làm 1 ly . Quán dễ kiếm , không_gian lại rộng_rãi . Nhân_viên thì dễ_thương gần_gũi . Nói_chung thèm trà sữa là mình ghé Quán ở đây vì gần nhà .	Quán này thấy khá nhiều người bảo mình nên mình đã đi ăn thử , nhưng thực_sự ăn xong thấy không được như mong_đợi lắm .

2 – Text Classification

Preprocessing

❖ Language Detection



Language Detector

langid library

Vietnamese Language

Quán này thấy khá nhiều người bảo mình nên mình đã đi ăn thử , nhưng thực_sự ăn xong thấy không được ngon. 👍 👍 </p>

Mình được 1 cô bạn giới_thiệu đến đây , tìm địa_chỉ khá dễ . Menu nước uống chất khỏi nói . https://foody.com

Other Language

Visiting_Da_Nang frequently but this is the first time I have found a coffee shop which has a creative design (korean style)

The room is cheap ! ! ! ! It ' s near the city center . The staff is so nice : - D 👍 👍 👍 👍 👍 👍\n

2 – Text Classification

Preprocessing

- Language Detection
- Text Cleaning

Vietnamese Language

Quán này thấy khá nhiều người bảo mình nên mình đã đi ăn thử , nhưng thực_sự ăn xong thấy không được ngon. 👍 👍 </p>

Mình được 1 cô bạn giới_thiệu đến đây , tìm địa_chỉ khá dễ . Menu nước uống chất khỏi nói . https://foody.com

1 – Removal URLs, HTML Tags

2 – Removal punctuations, digits

3 – Removal emoticons, flags,…

4 – Normalize whitespace

5 – Lowercasing

2 – Text Classification

Index-Based Represenation

N Samples

[dog, bites, man]
[man, bites, dog]

V Vocabulary

IDX	Token
0	<pad>
1	<unk>
2	dog

Input matrix
Index-based Representation

M: Sequence Length

$$\begin{bmatrix} 2 & 4 & 3 \\ 3 & 4 & 2 \end{bmatrix}$$

Input shape: N x M

Embedding Matrix
(Lookup Table)

0	0.1	3.1
1	0.5	2.5
2	1.3	0.6

D: Embedding Dim
Shape: V x D

0.6 1.4 0.1

1.3 0.7 0.4

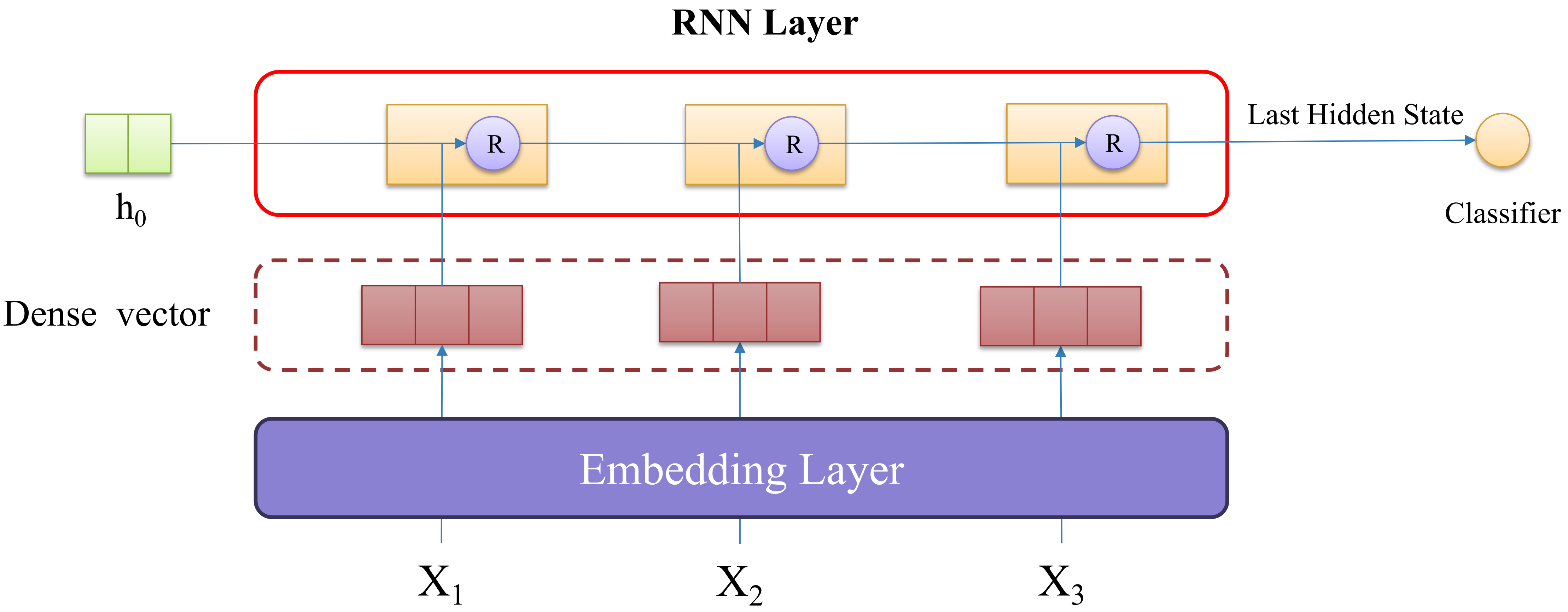
0.4 0.7 1.3

Output shape: N x M x D

Model

2 – Text Classification

Modeling



AI VIET NAM
@aivietnam.edu.vn

2 – Text Classification

Modeling – Demo

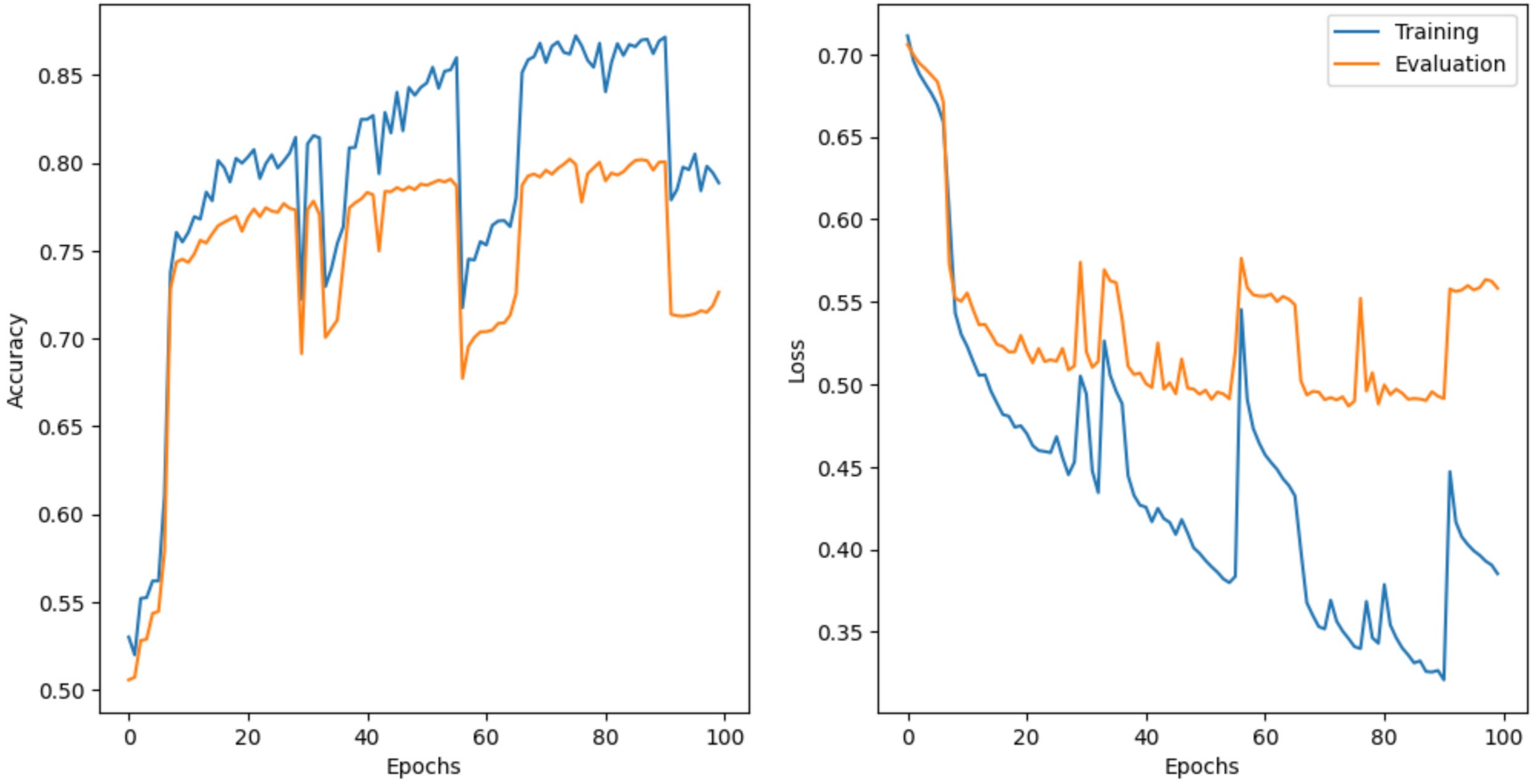
```python
embedding_dim = 200
hidden_size = 50

class RNNClassifier(nn.Module):
    def __init__(self, num_classes):
        super(RNNClassifier, self).__init__()
        self.embedding_layer = nn.Embedding(
            num_embeddings=vocab_size,
            embedding_dim=embedding_dim
        )
        self.rnn = nn.RNN(
            input_size=embedding_dim,
            hidden_size=hidden_size,
            batch_first=True
        )
        self.linear = nn.Linear(hidden_size, num_classes)

    def forward(self, X_batch, device):
        embeddings = self.embedding_layer(X_batch)
        output, hidden = self.rnn(
            embeddings,
            torch.randn(1, len(X_batch), hidden_size).to(device)
        )
        output = self.linear(output[: , -1])
        return output
```

2 – Text Classification

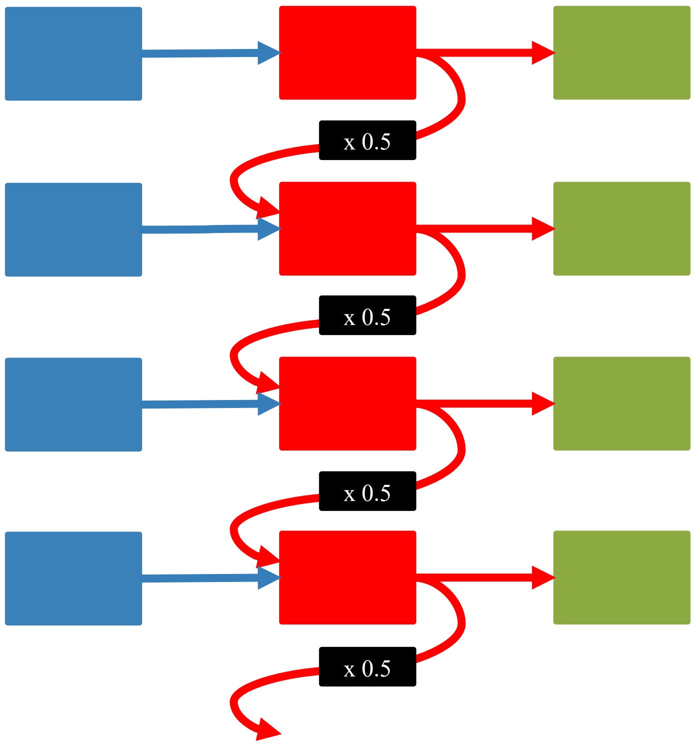
Training – Demo



3 – Long Short Term Memory

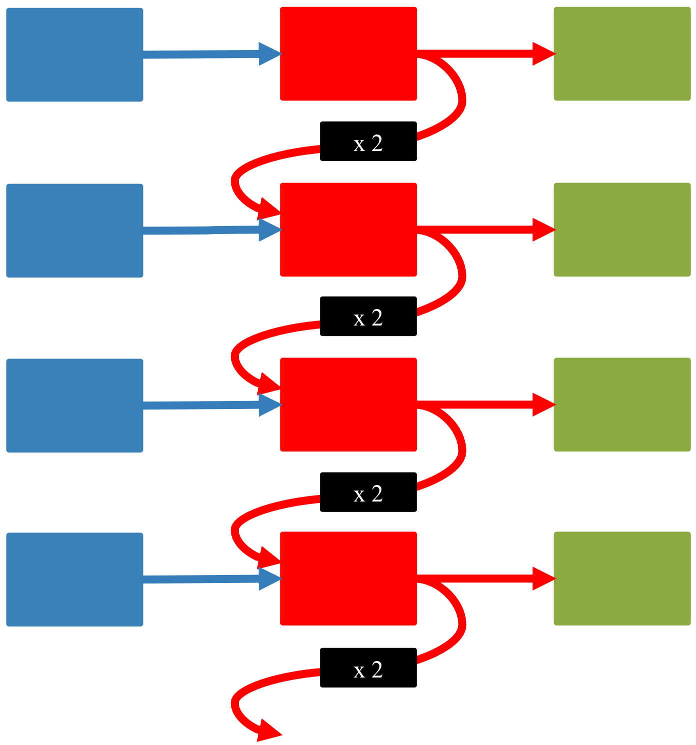
RNN Drawbacks

Vanishing Gradient



input * 0.5^n
n: unroll time

Exploding Gradient



input * 2^n
n: unroll time

3 – Long Short Term Memory

Prior Knowledge

Sigmoid

$$0 \leq \sigma(x) \leq 1$$

$$\sigma(x) = \frac{1}{1 + e^{-x}} = \frac{1}{1 + \frac{1}{e^x}} = \frac{1}{\frac{e^x + 1}{e^x}} = \frac{e^x}{e^x + 1}$$

Tanh

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

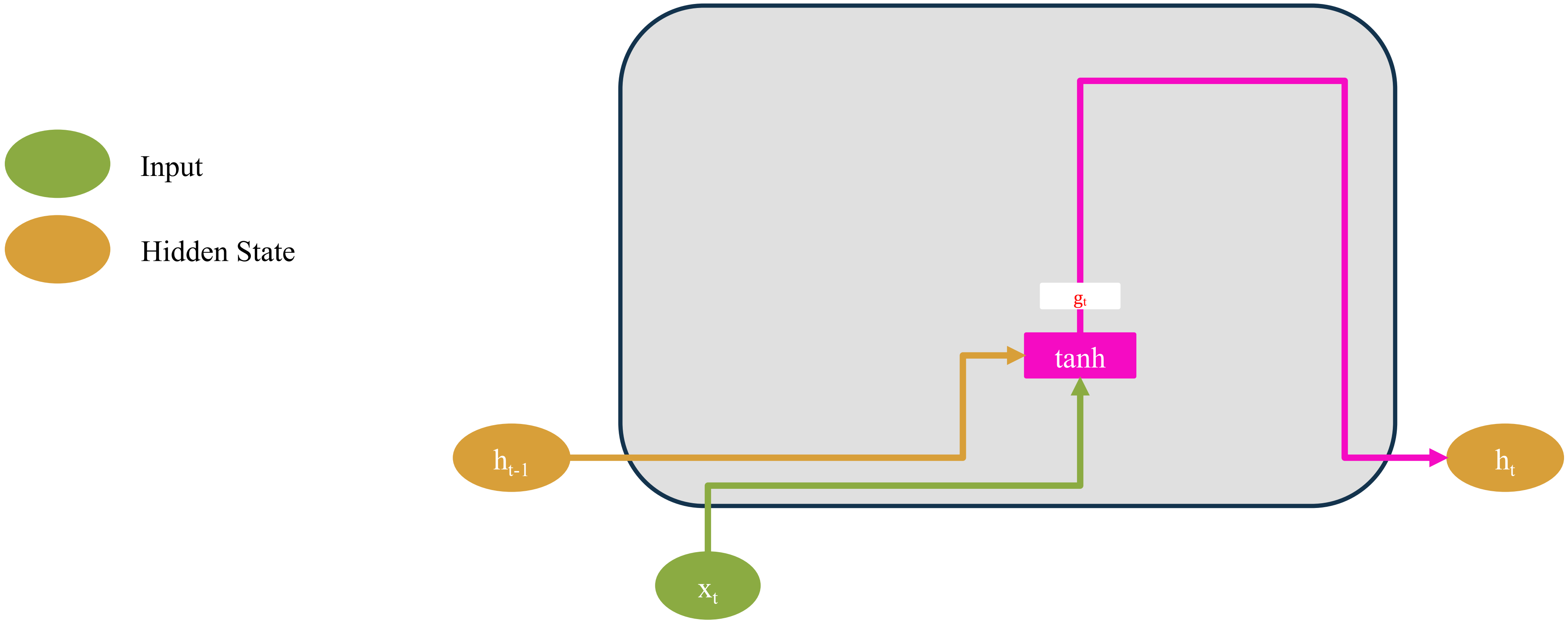
$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Matrix Multiplication

$$\begin{bmatrix} a_1 & b_1 \\ c_1 & d_1 \end{bmatrix} \cdot \begin{bmatrix} a_2 & b_2 \\ c_2 & d_2 \end{bmatrix} = \begin{bmatrix} a_1 a_2 + b_1 c_2 & a_1 b_2 + b_1 d_2 \\ c_1 a_2 + d_1 c_2 & c_1 b_2 + d_1 d_2 \end{bmatrix}$$

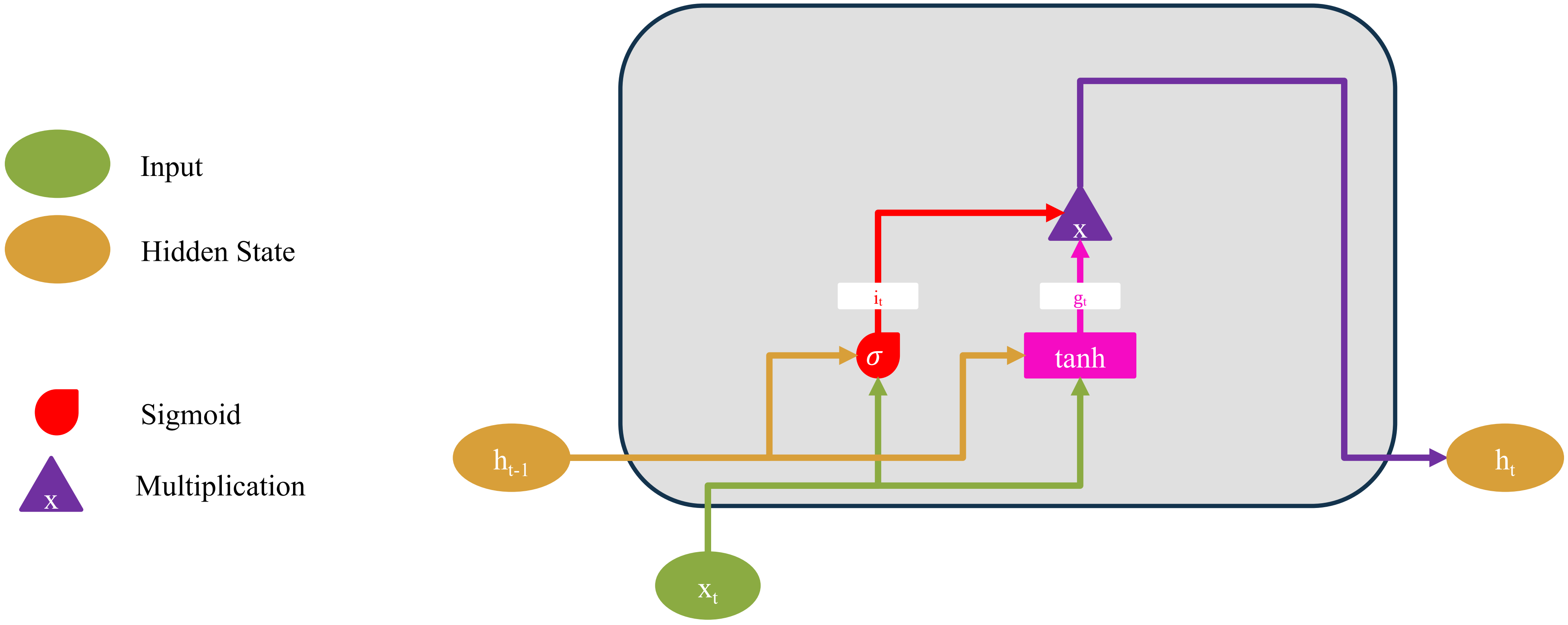
3 – Long Short Term Memory

RNN



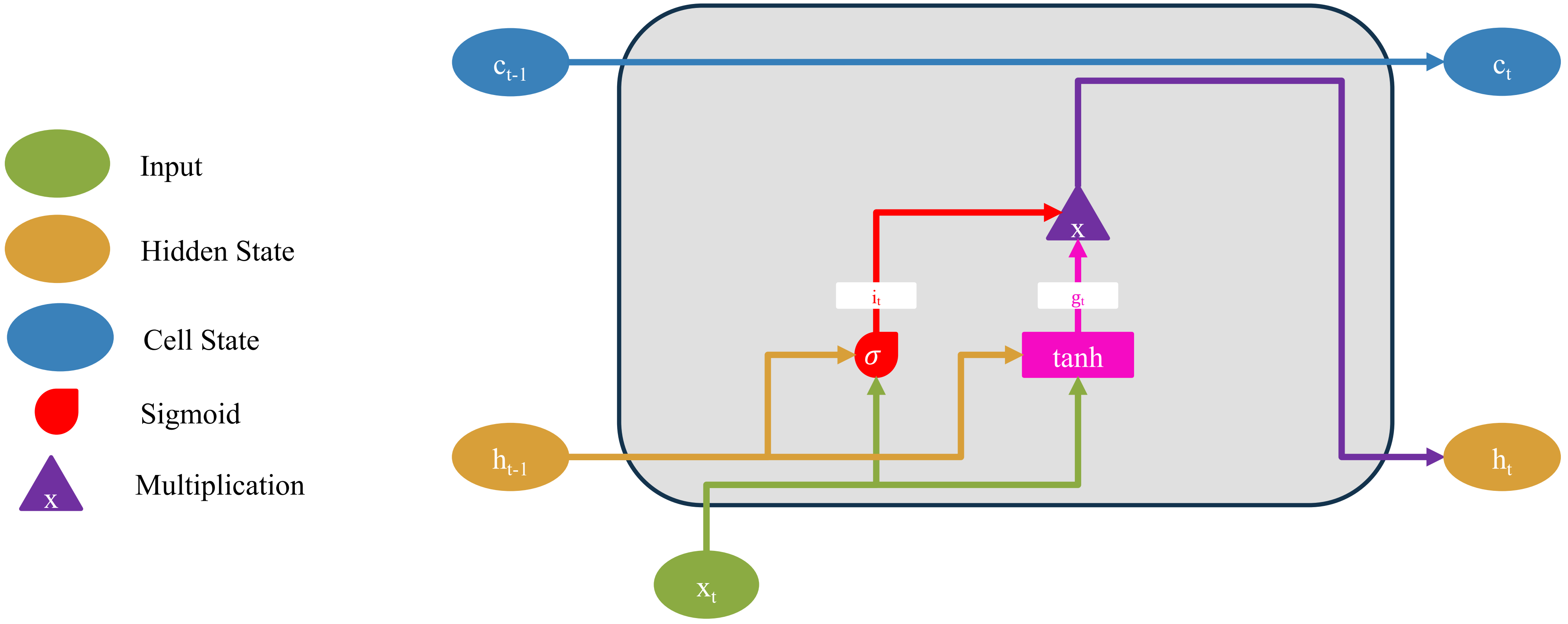
3 – Long Short Term Memory

Filter RNN's Output



3 – Long Short Term Memory

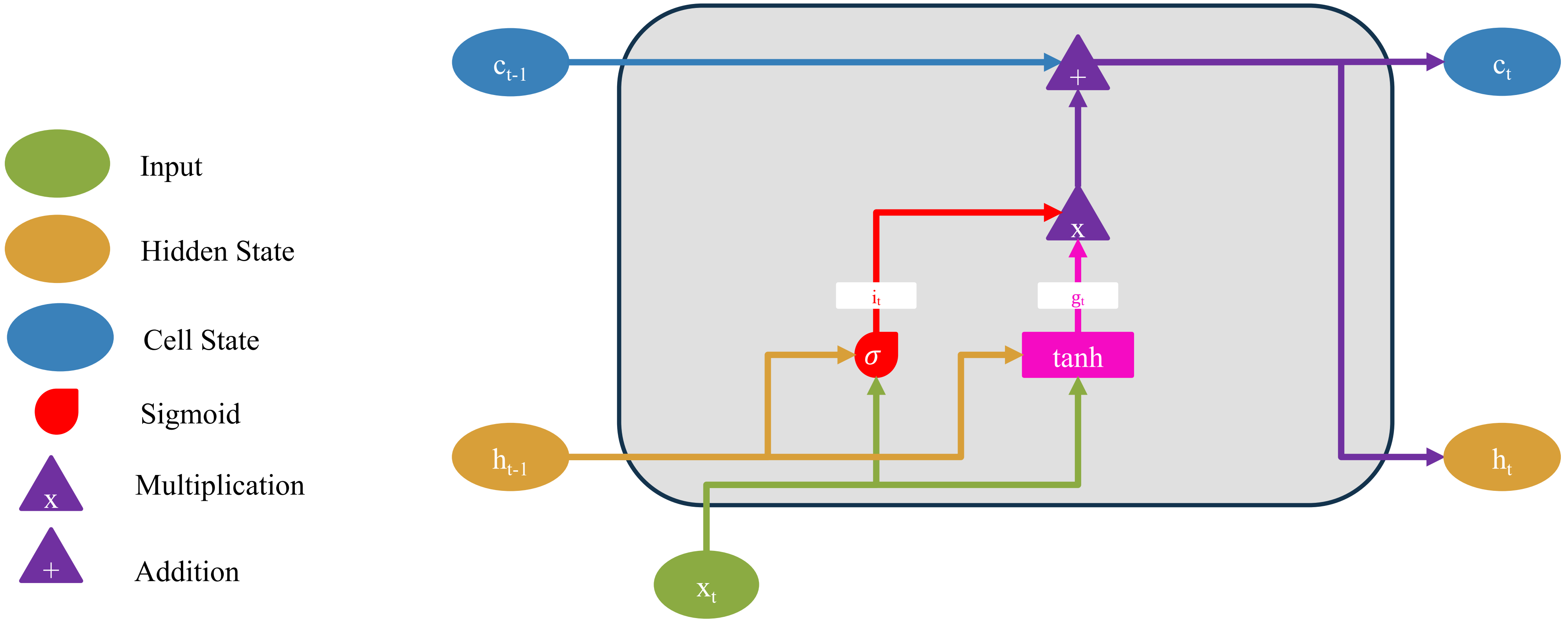
Long-Term Memory



AI VIET NAM
@aivietnam.edu.vn

3 – Long Short Term Memory

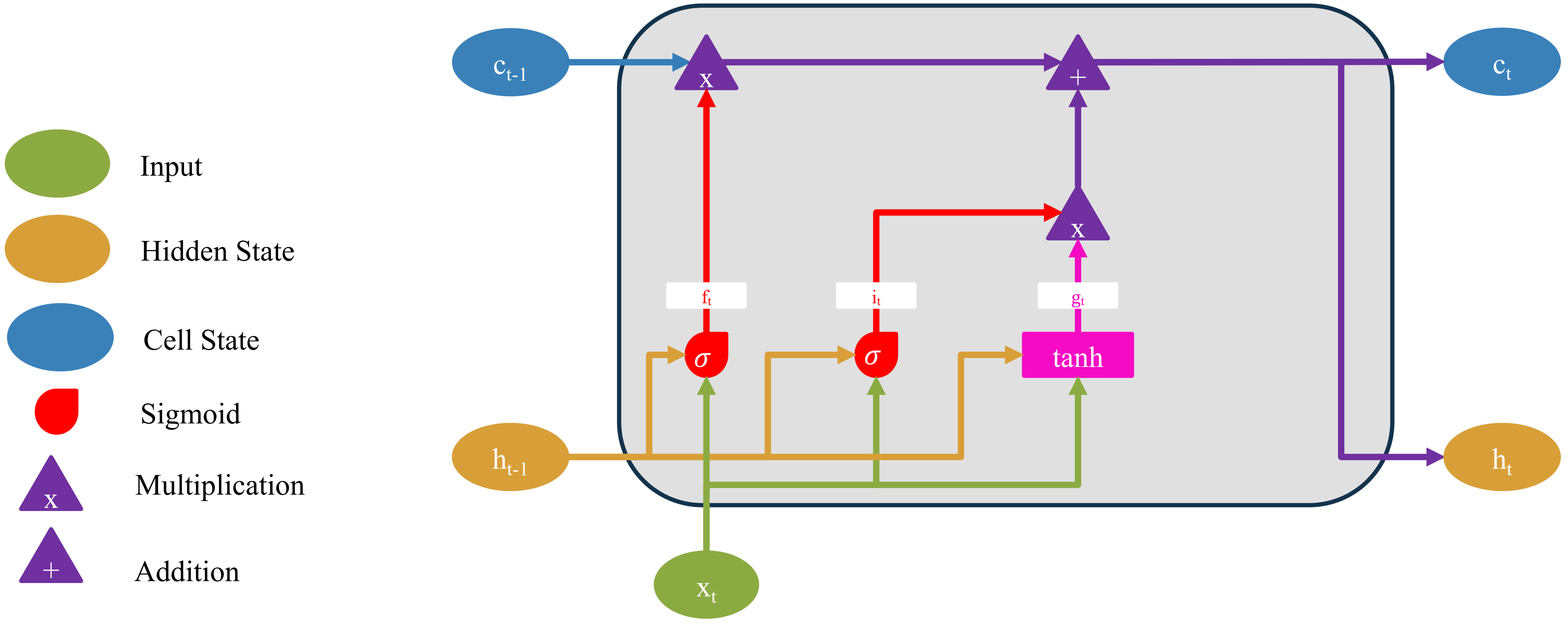
Add Information For Long-Term Memory



AI VIET NAM
@aivietnam.edu.vn

3 – Long Short Term Memory

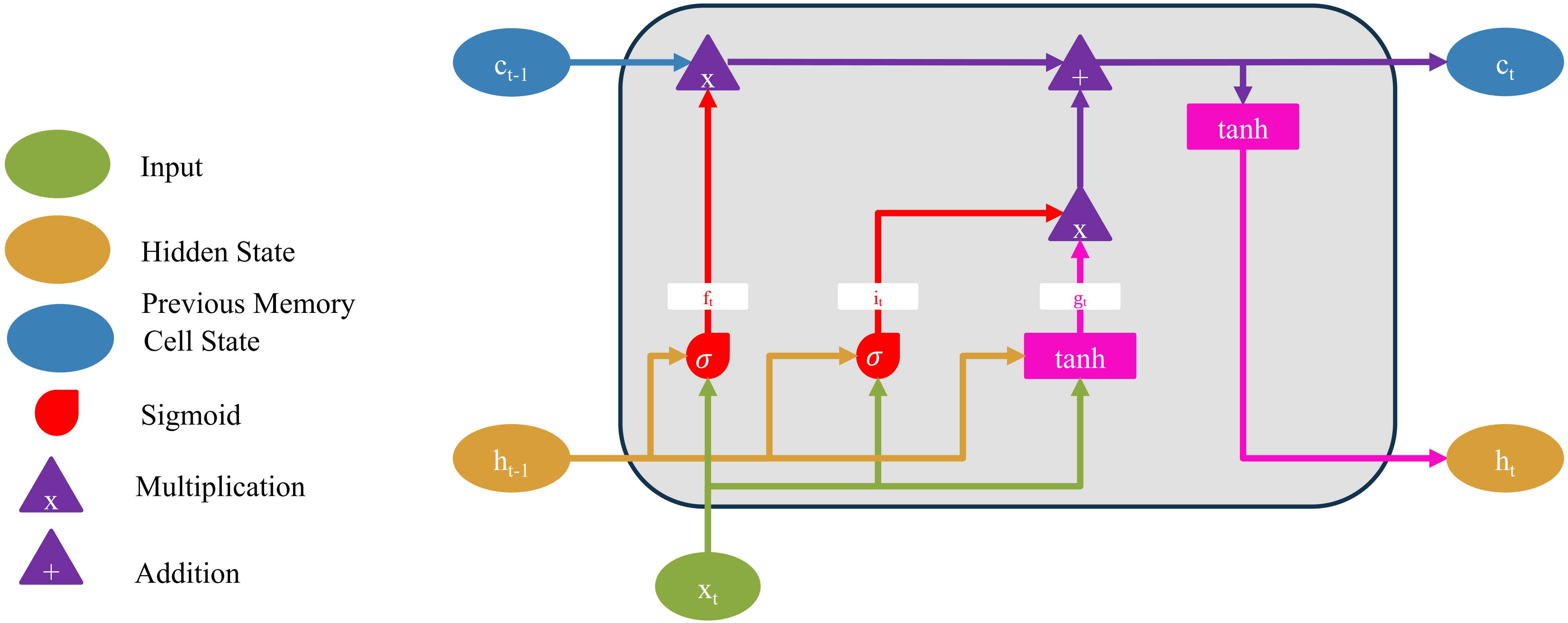
Filter Long-Term Memory



AI VIET NAM
@aivietnam.edu.vn

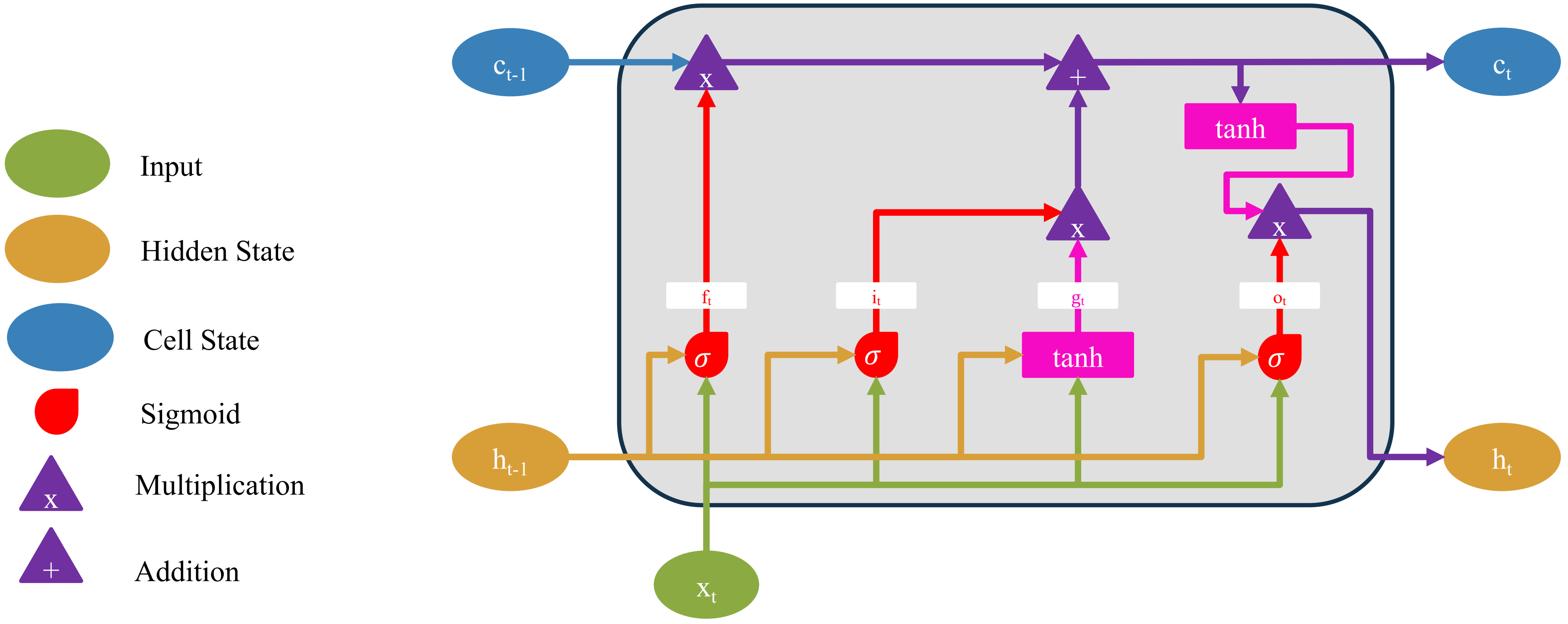
3 – Long Short Term Memory

Normalize Shot-Term Memory



3 – Long Short Term Memory

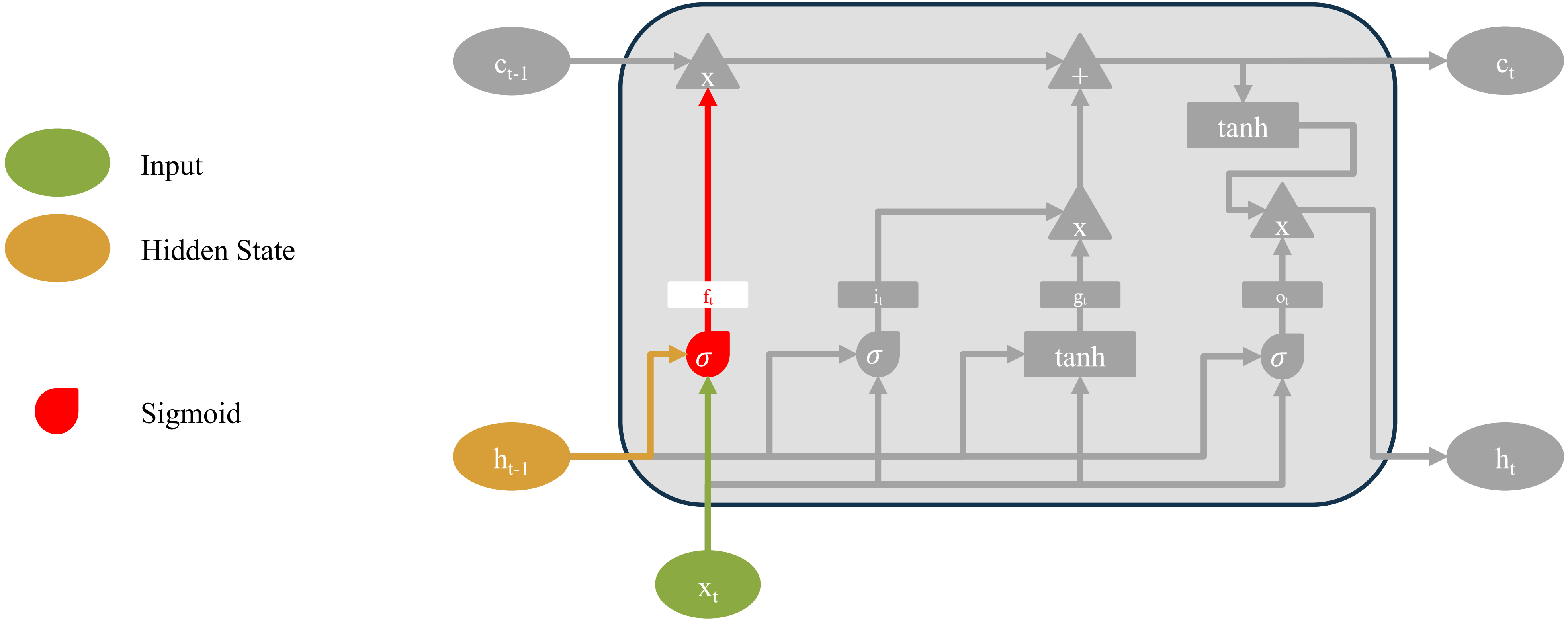
Filter The Normalized Short-Term Memory



3 – Long Short Term Memory

LSTM – Forget Gate

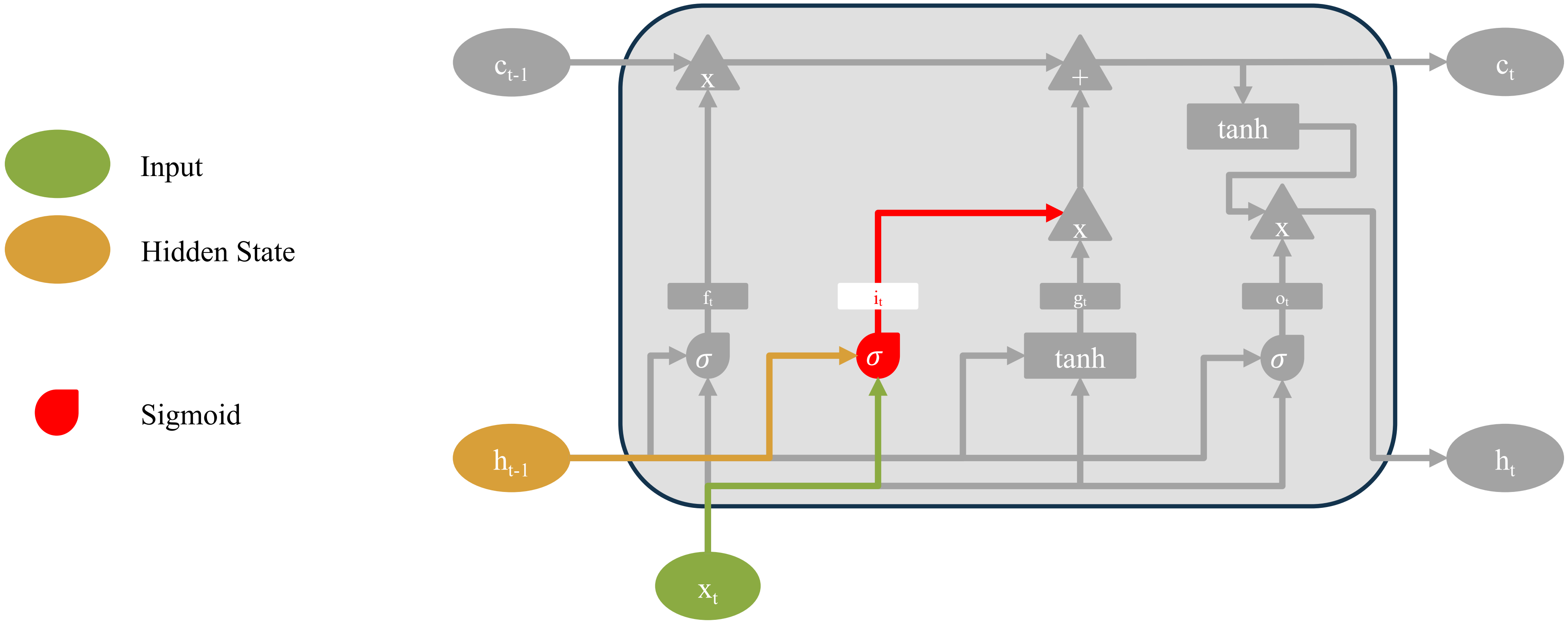
$$f_t = \sigma\left(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}\right)$$



3 – Long Short Term Memory

LSTM – Input Gate

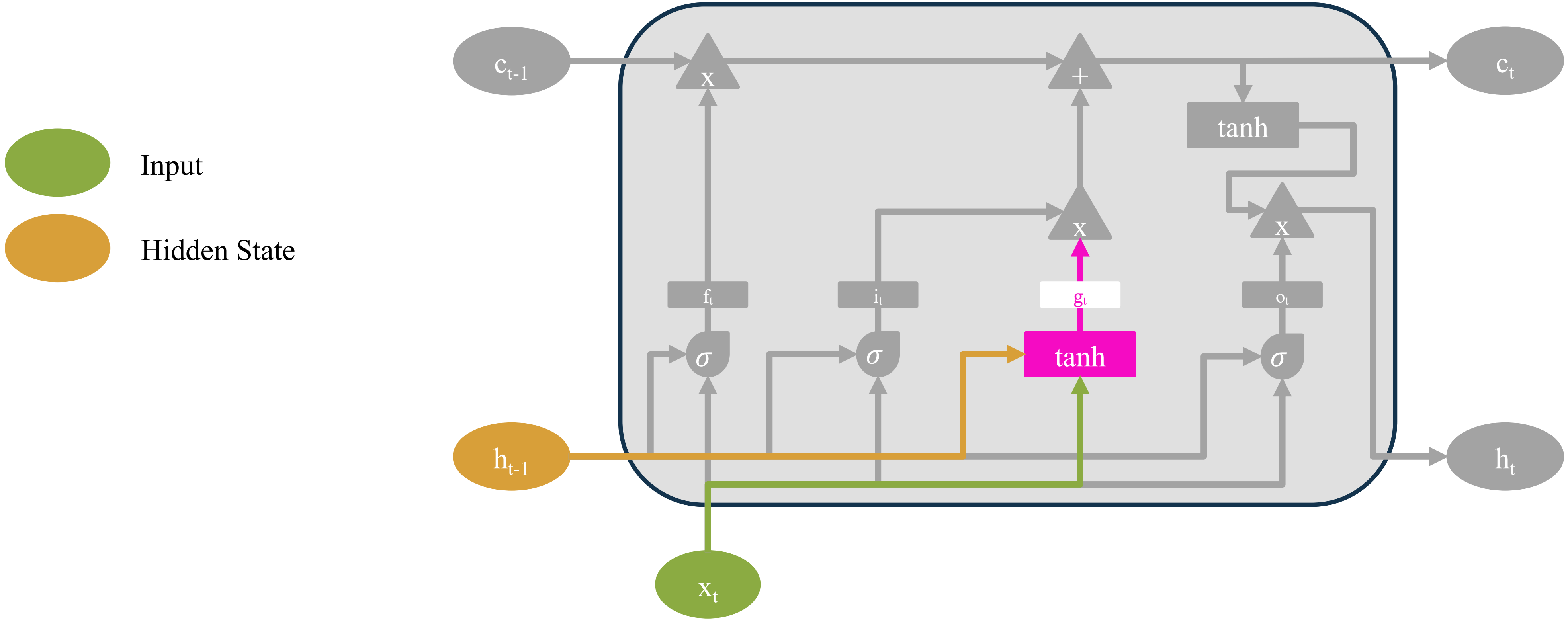
$$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi})$$



3 – Long Short Term Memory

LSTM – Candidate Memory

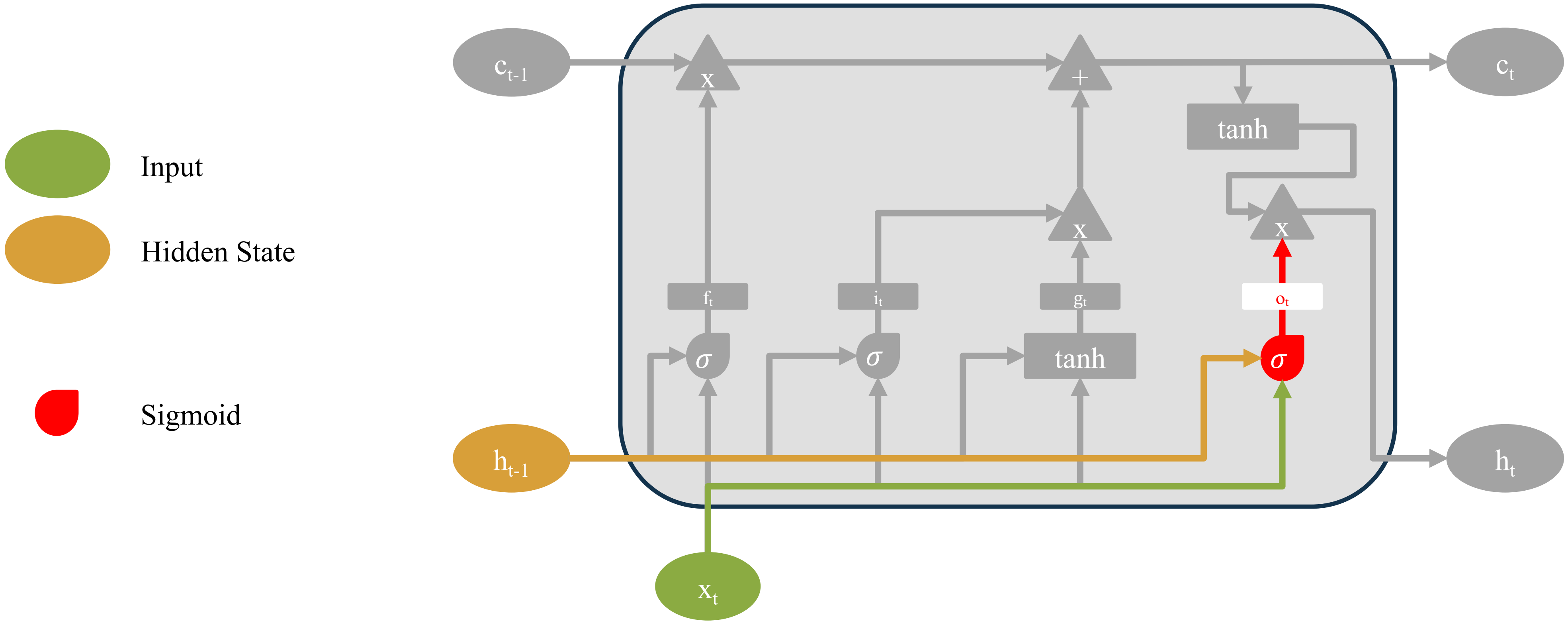
$$g_t = tanh\left(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}\right)$$



3 – Long Short Term Memory

LSTM – Output Gate

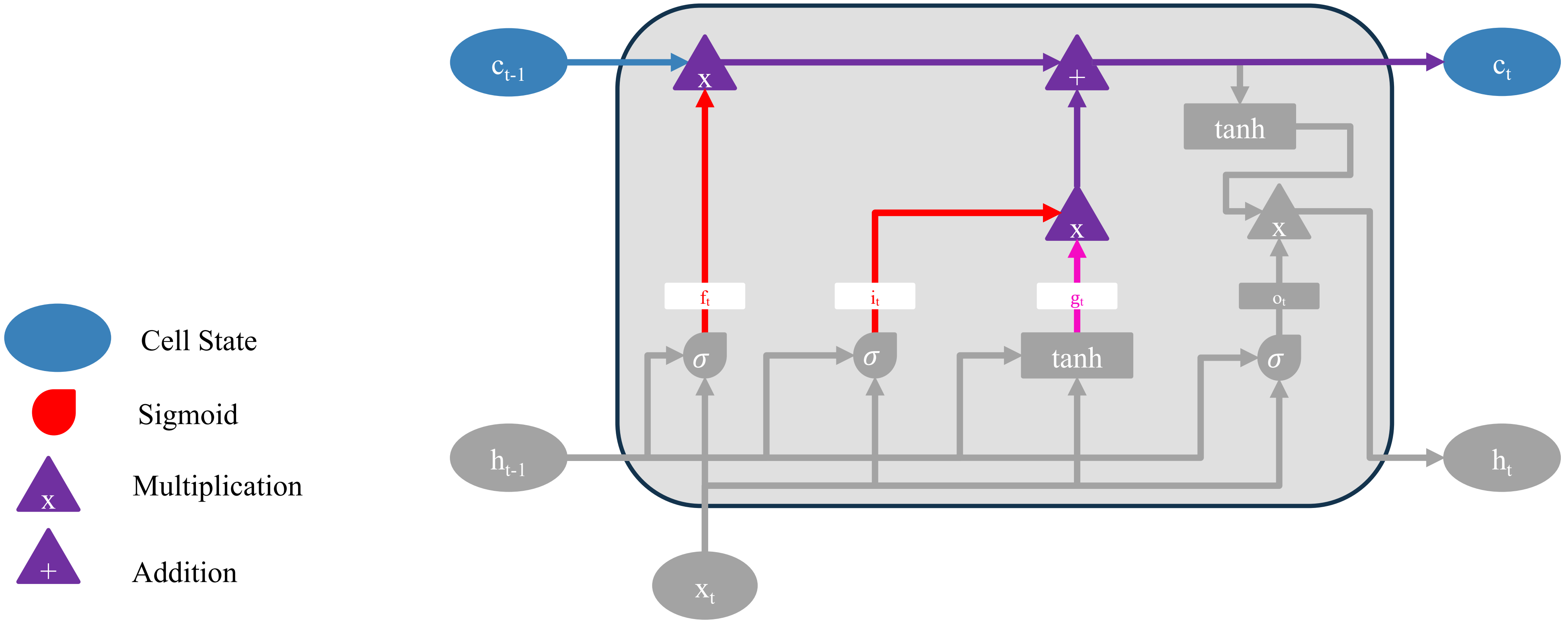
$$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho})$$



3 – Long Short Term Memory

LSTM– Current Cell State

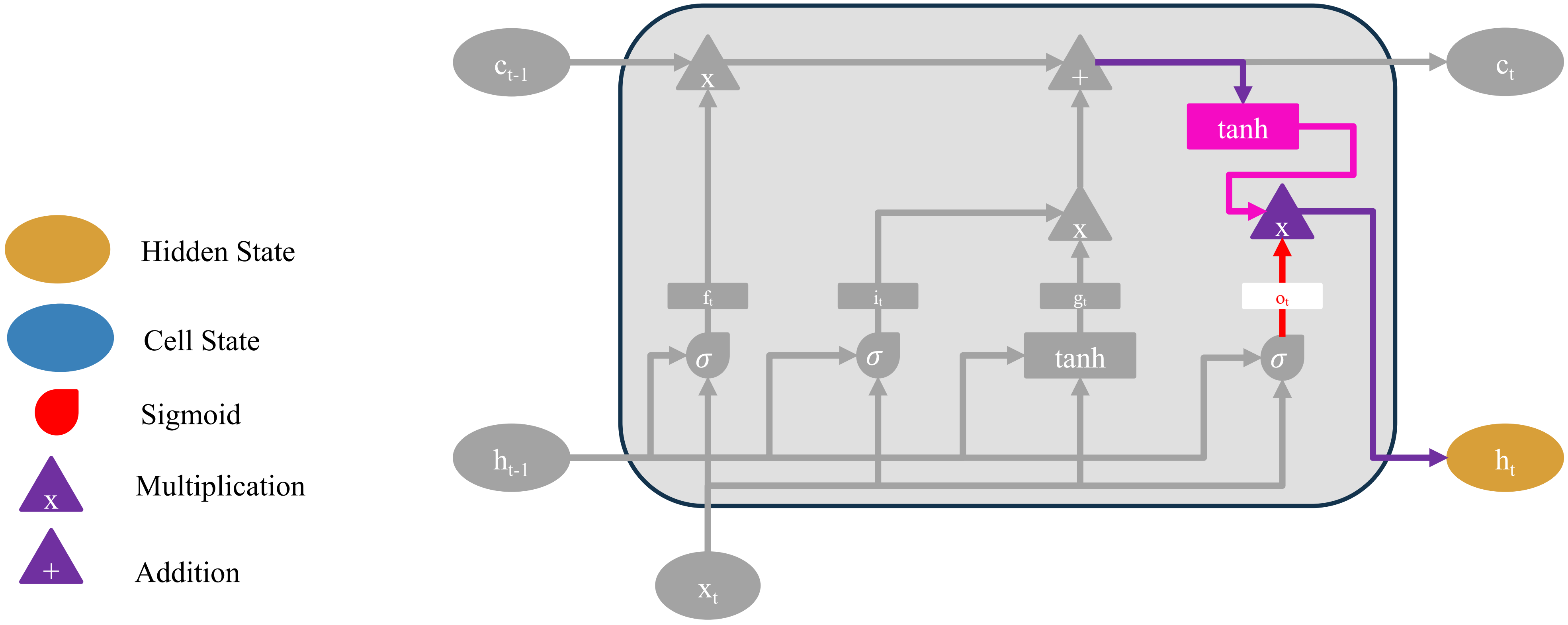
$$c_t = f_t \odot c_{t-1} + i_t \odot g_t$$



3 – Long Short Term Memory

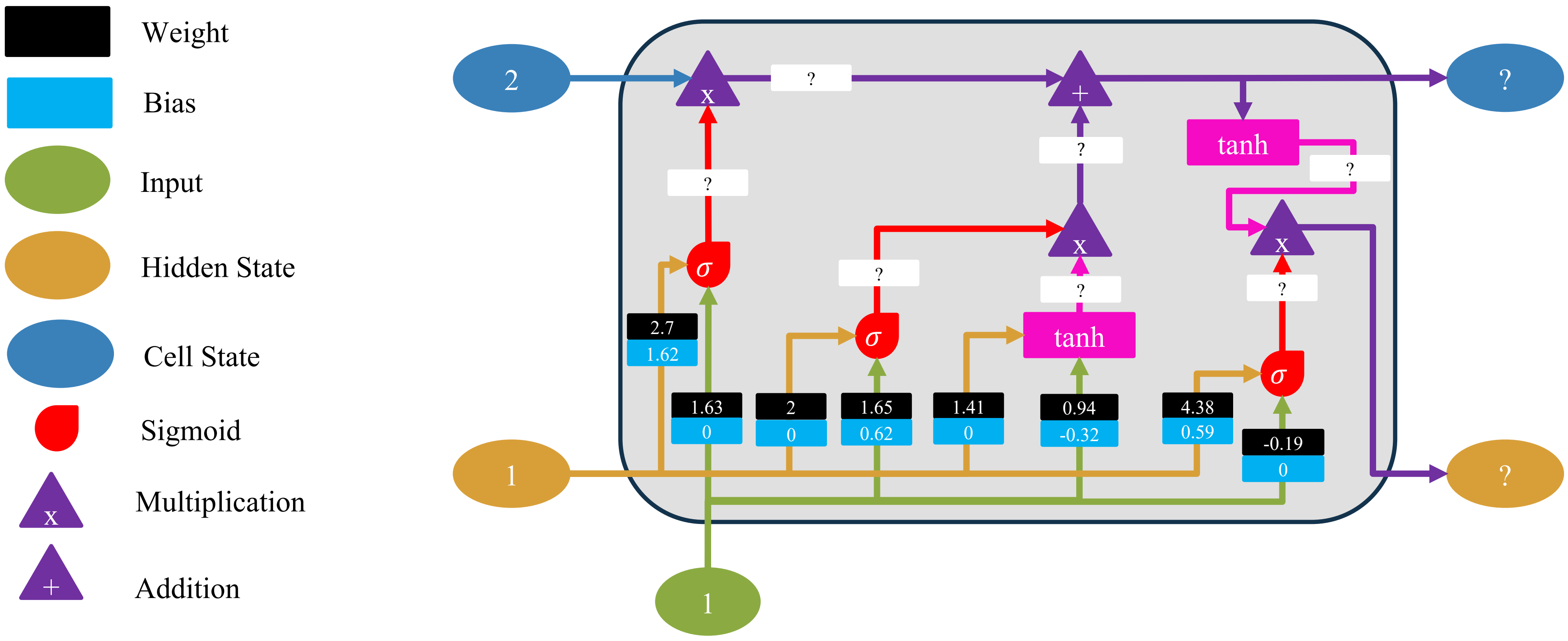
LSTM - Current Hidden State

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



3 – Long Short Term Memory

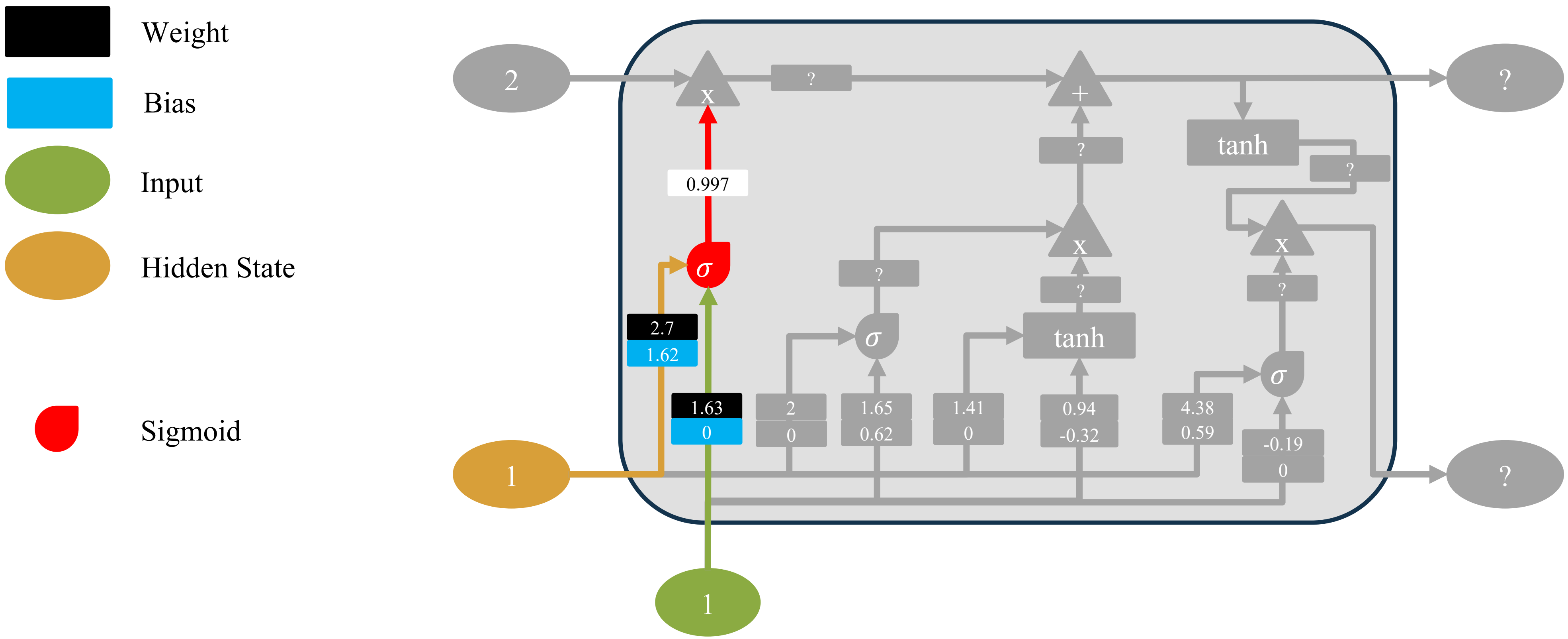
Example



3 – Long Short Term Memory

Example - Forget Gate

$$f_t = \sigma\left(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}\right)$$



$W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}$

$= 1.63*1+0+2.7*1+1.62$

$= 5.95$

$\sigma\left(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}\right)$

$= \sigma(5.95)$

$= \dfrac{e^{5.95}}{e^{5.95}+1}$

$= 0.997$

AI VIET NAM
@aivietnam.edu.vn

3 – Long Short Term Memory

Example - Input Gate

$$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi})$$

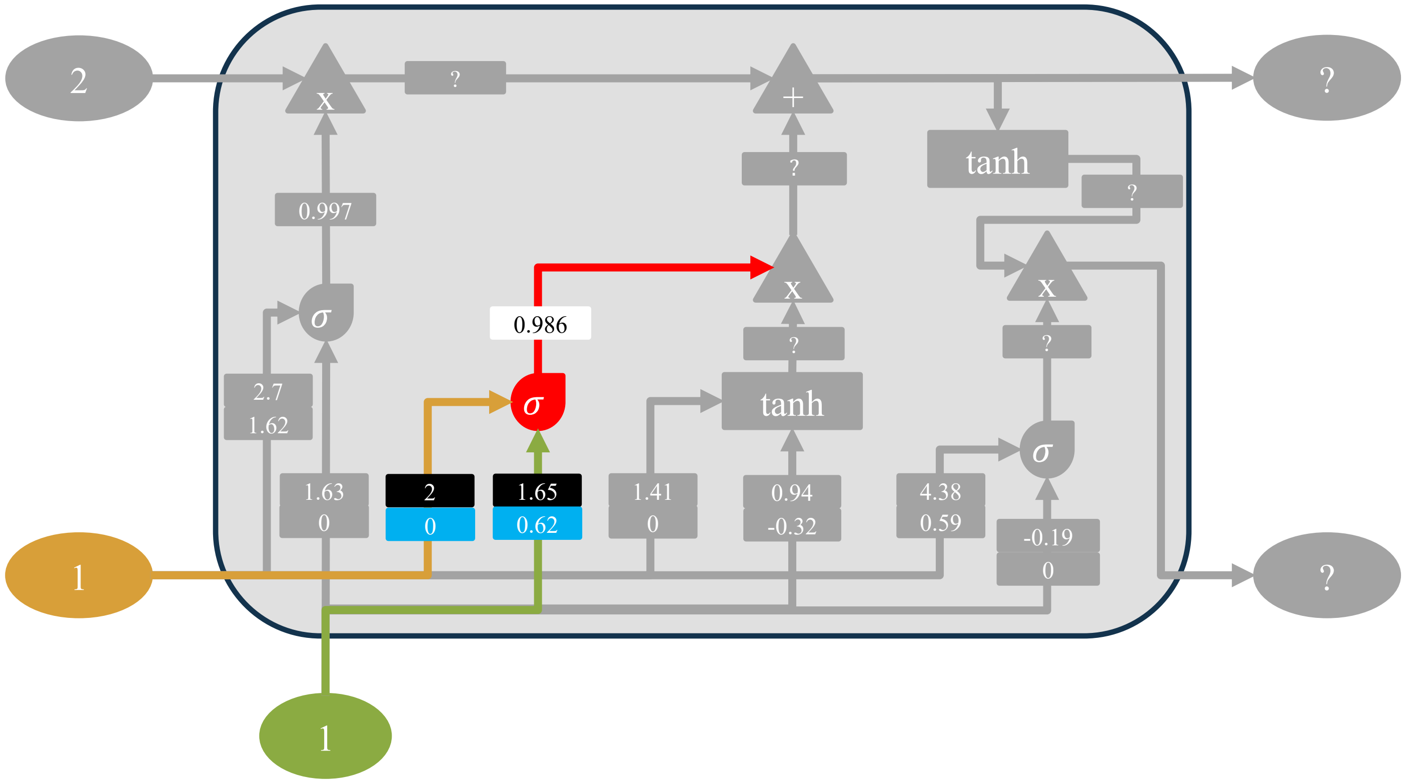
Weight

Bias

Input

Hidden State

Sigmoid



$W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}$

$= 1.65*1+0.62+2*1+0$

$= 4.27$

$\sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi})$

$= \sigma(4.27)$

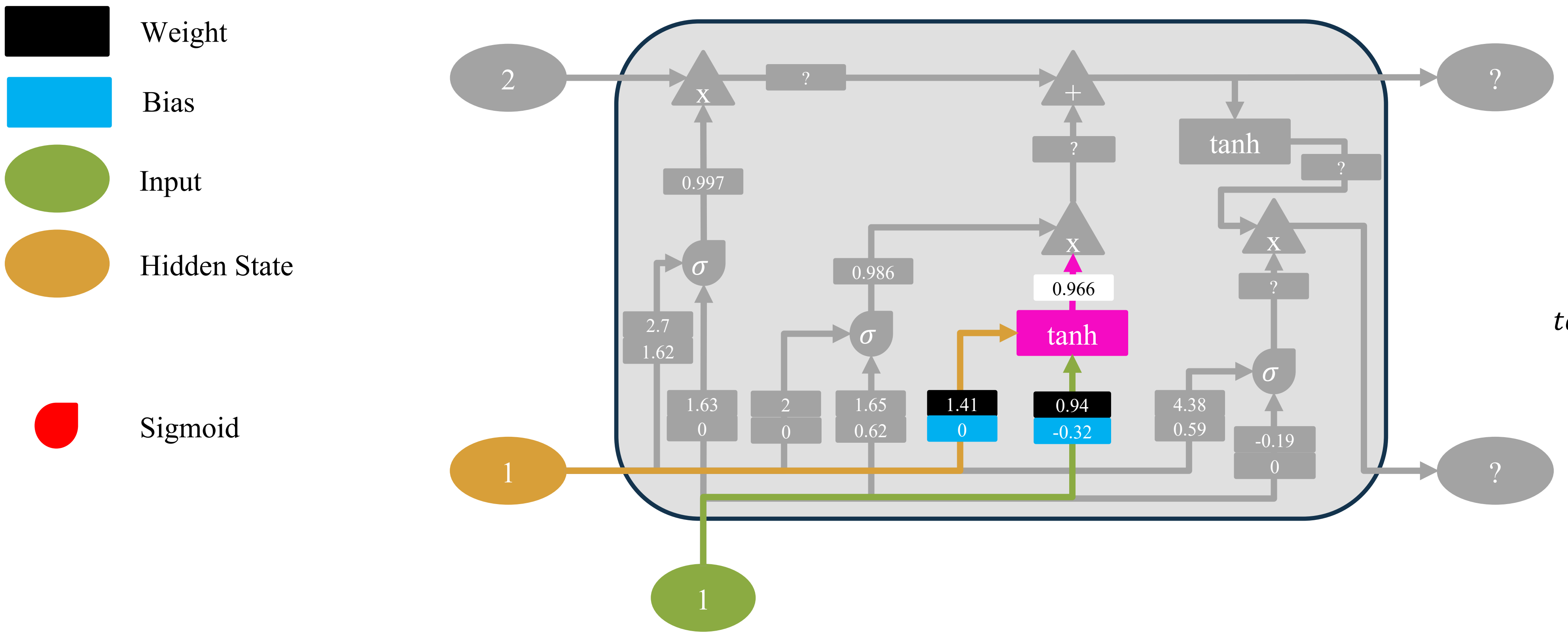
$= \dfrac{e^{4.27}}{e^{4.27}+1}$

$= 0.986$

2 – Long Short Term Memory

Example - Candidate Memory

$$g_t = tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg})$$



$W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}$

$= 0.94*1+(-0.32)+1.41*1+0$

$= 2.03$

$tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg})$

$= tanh(2.03)$

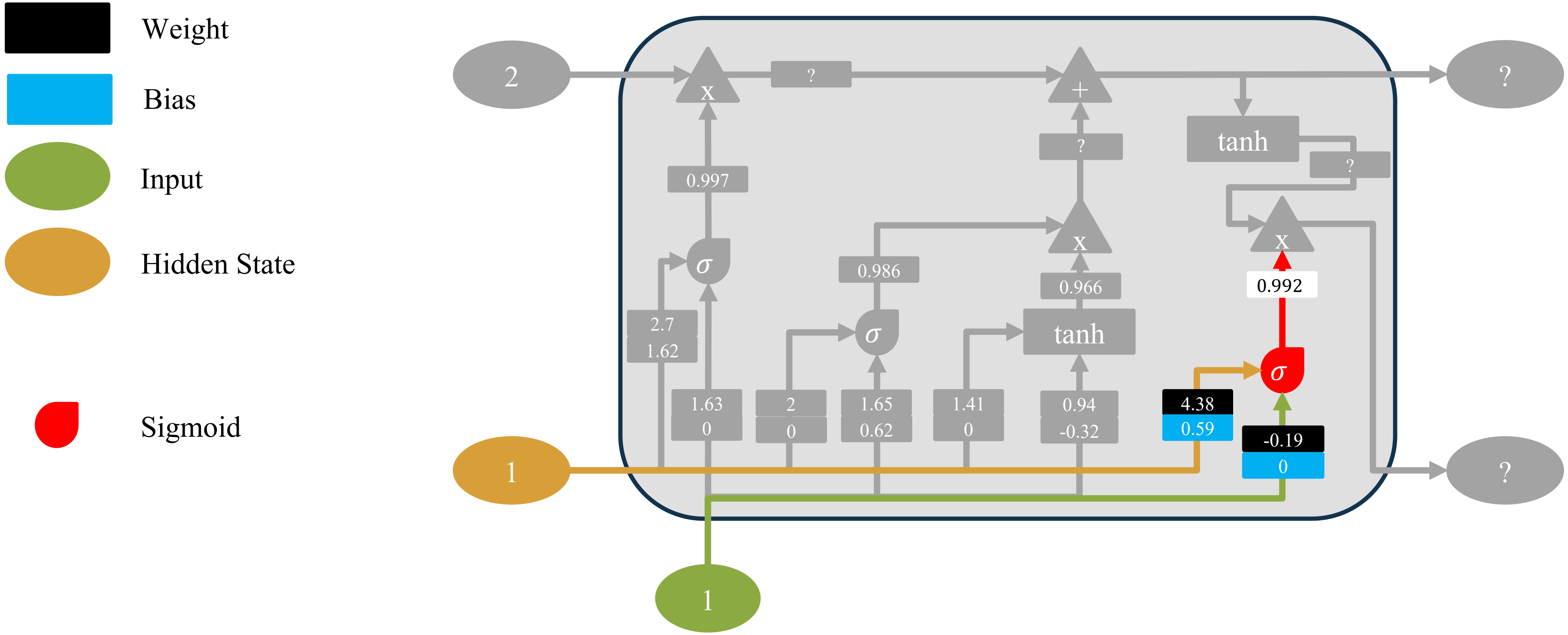
$= \frac{e^{2.03} - e^{-2.03}}{e^{2.03} + e^{-2.03}}$

$= 0.966$

3 – Long Short Term Memory

Example - Output Gate

$$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho})$$



$W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}$

$= -0.19*1+0+4.38*1+0.59$

$= 4.78$

$\sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho})$

$= \sigma(4.78)$

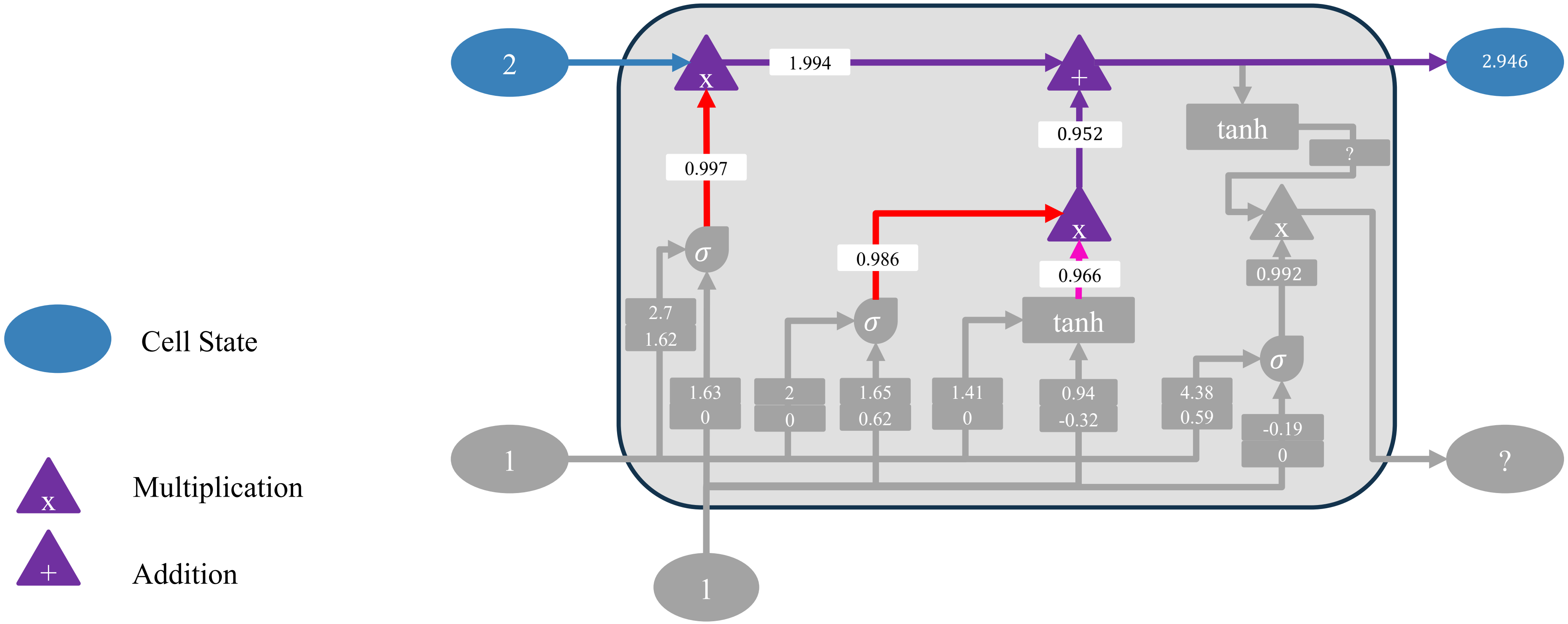
$= \dfrac{e^{4.78}}{e^{4.78}+1}$

$= 0.992$

3 – Long Short Term Memory

Example – Current Cell State

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t$$



$f_t \odot c_{t-1} + i_t g_t$

$= 0.997 * 2 + 0.986 * 0.966$

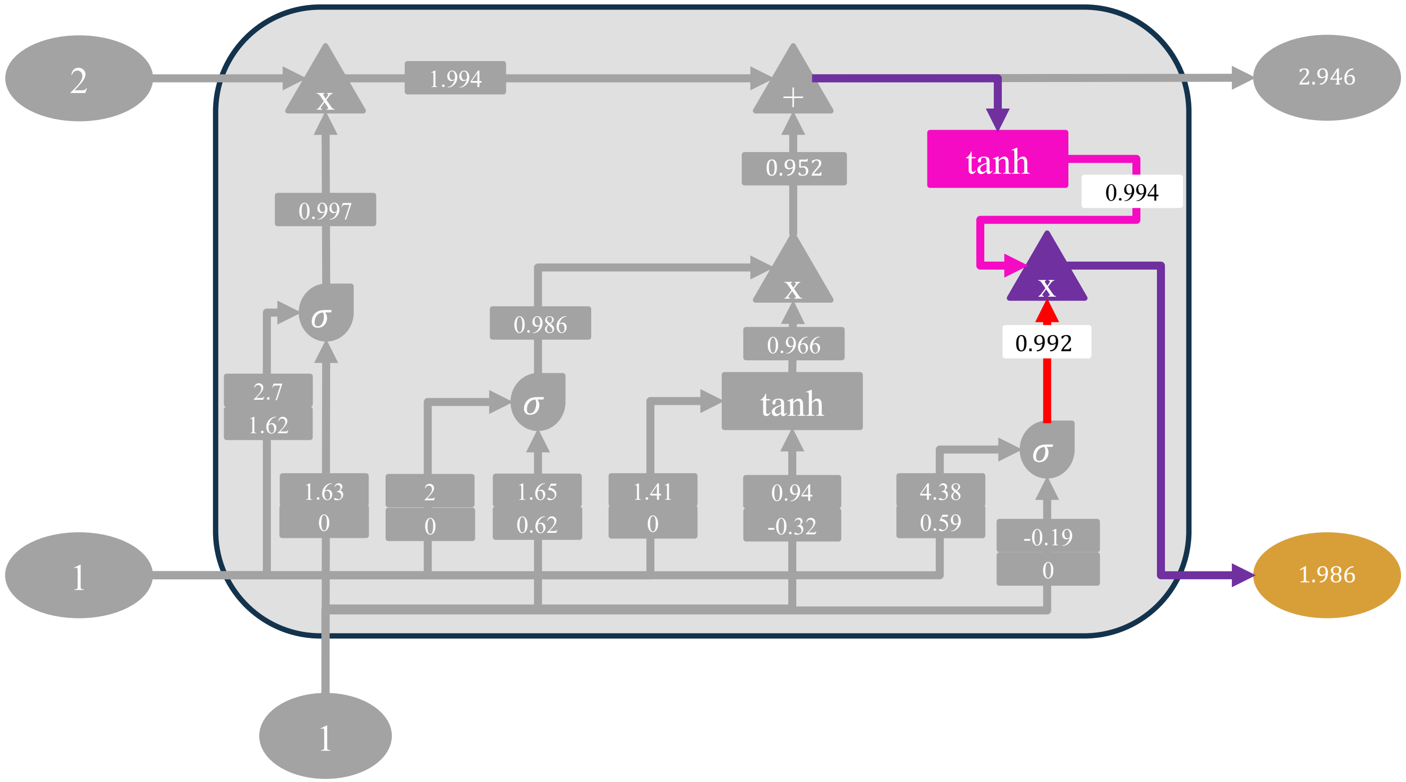
$= 1.994 + 0.952$

$= 2.946$

3 – Long Short Term Memory

Example - Current Hidden State

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



Hidden State

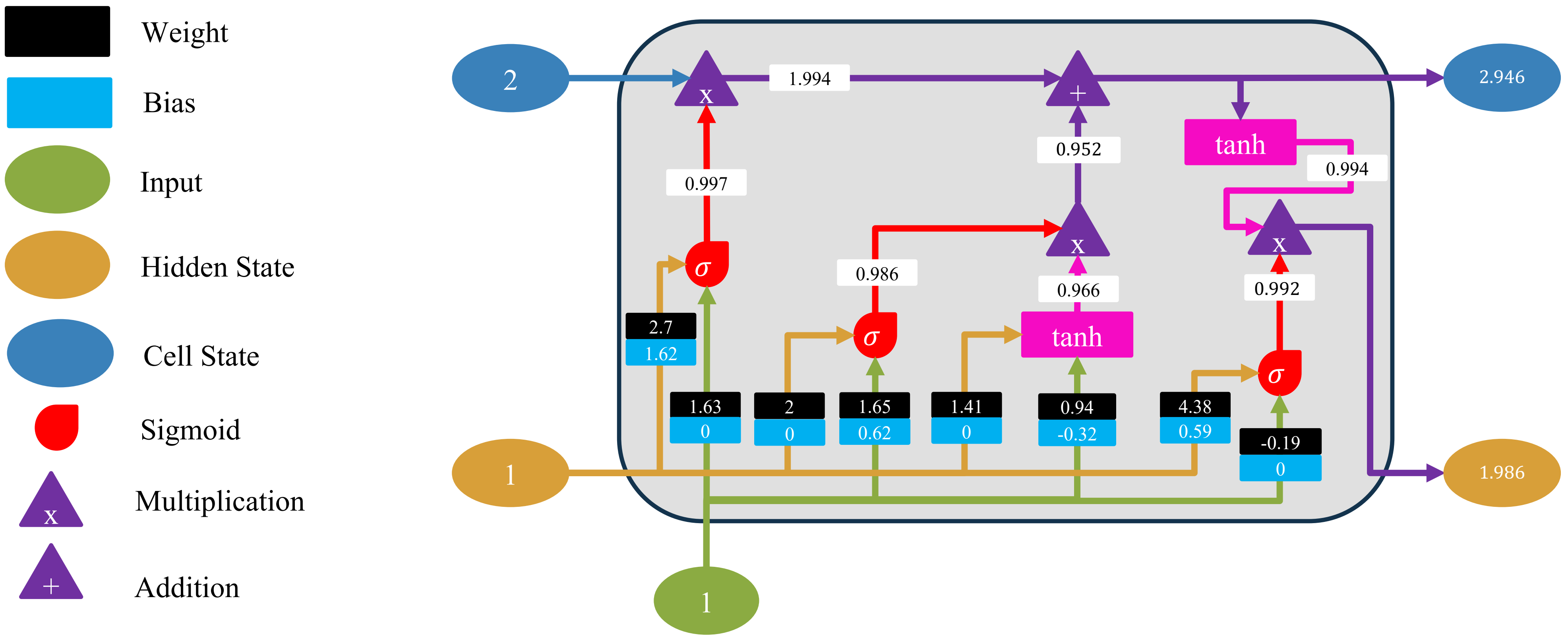
Multiplication

$$\tanh(c_t)$$
$$= \tanh(2.946)$$
$$= \frac{e^{2.946} - e^{-2.946}}{e^{2.946} + e^{-2.946}}$$
$$= 0.994$$

$$o_t \odot \tanh(c_t)$$
$$= 0.992 + 0.994$$
$$= 1.986$$

3 – Long Short Term Memory

Example - Final Result



Thanks!

Any questions?