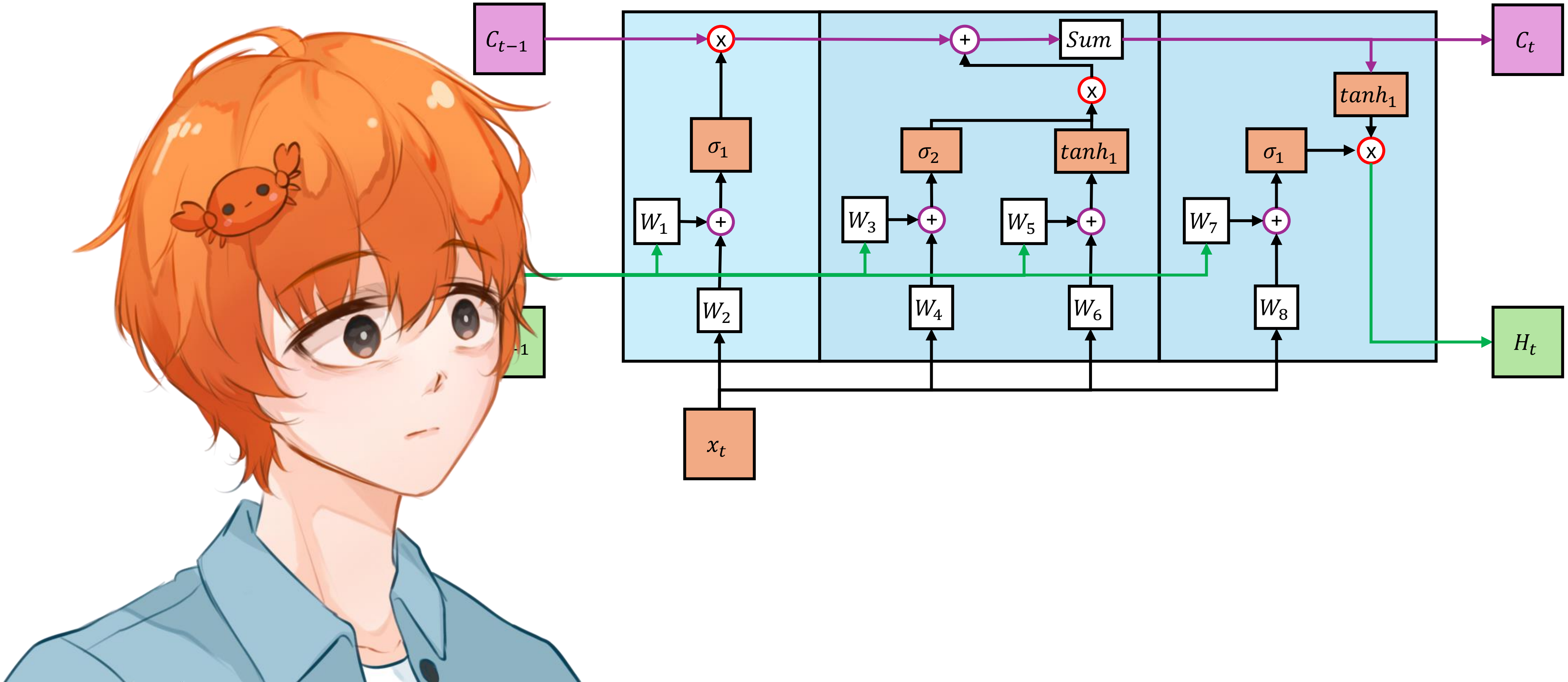
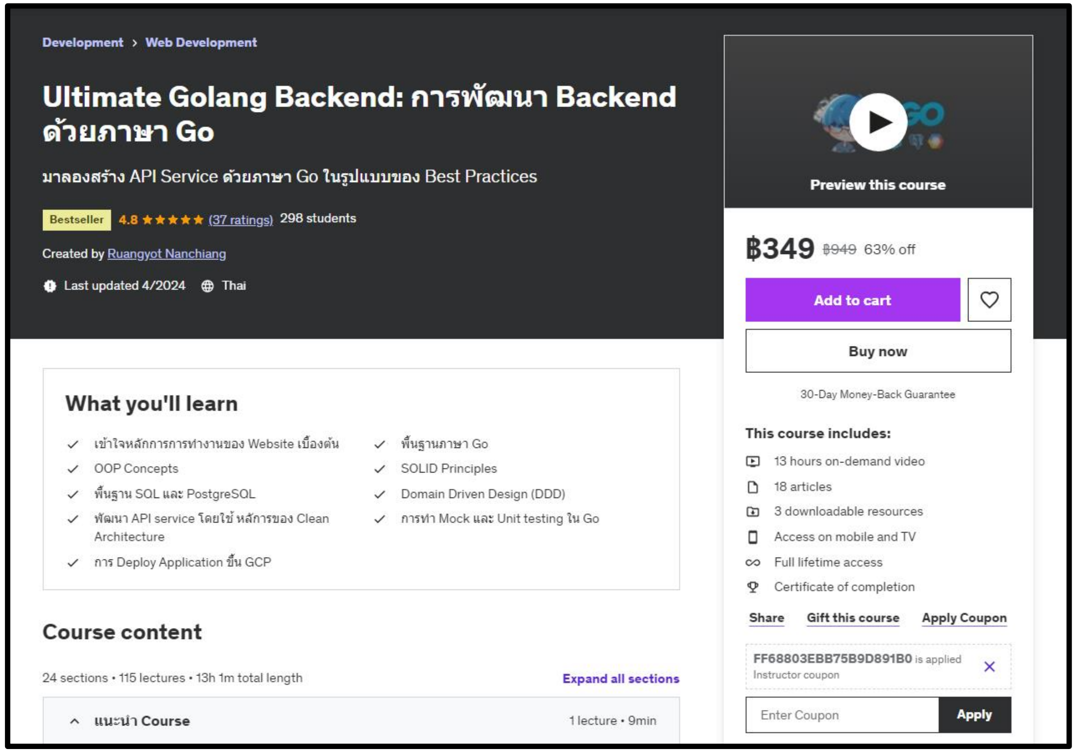
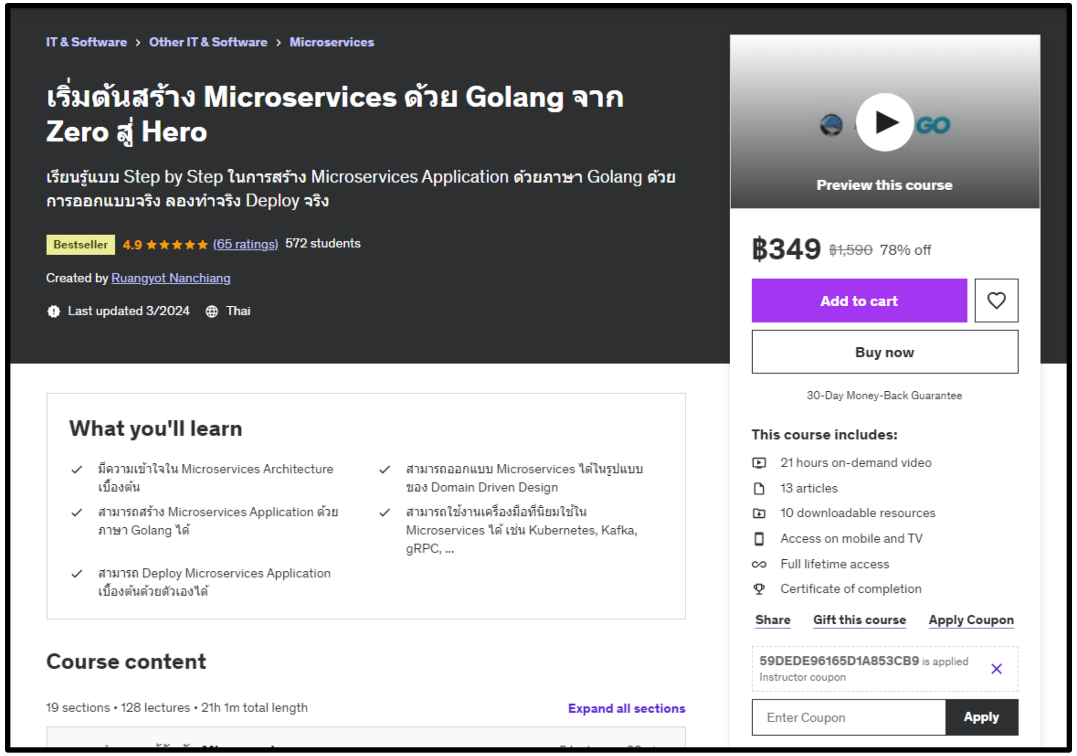
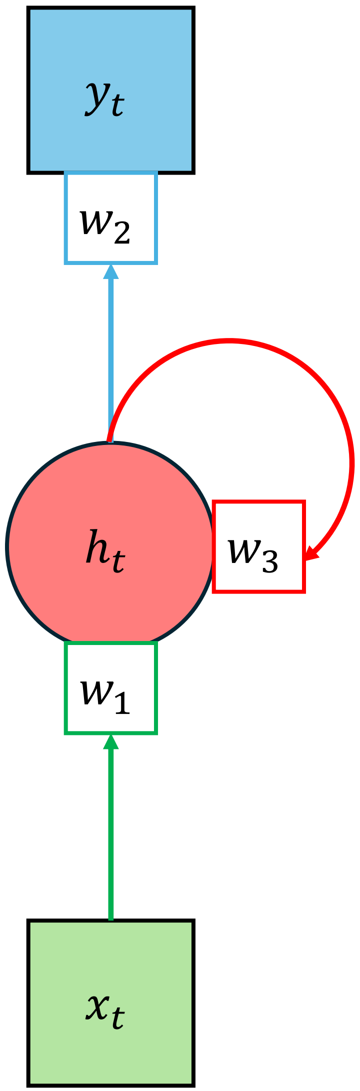
Long Short-Term Memory (LSTM)



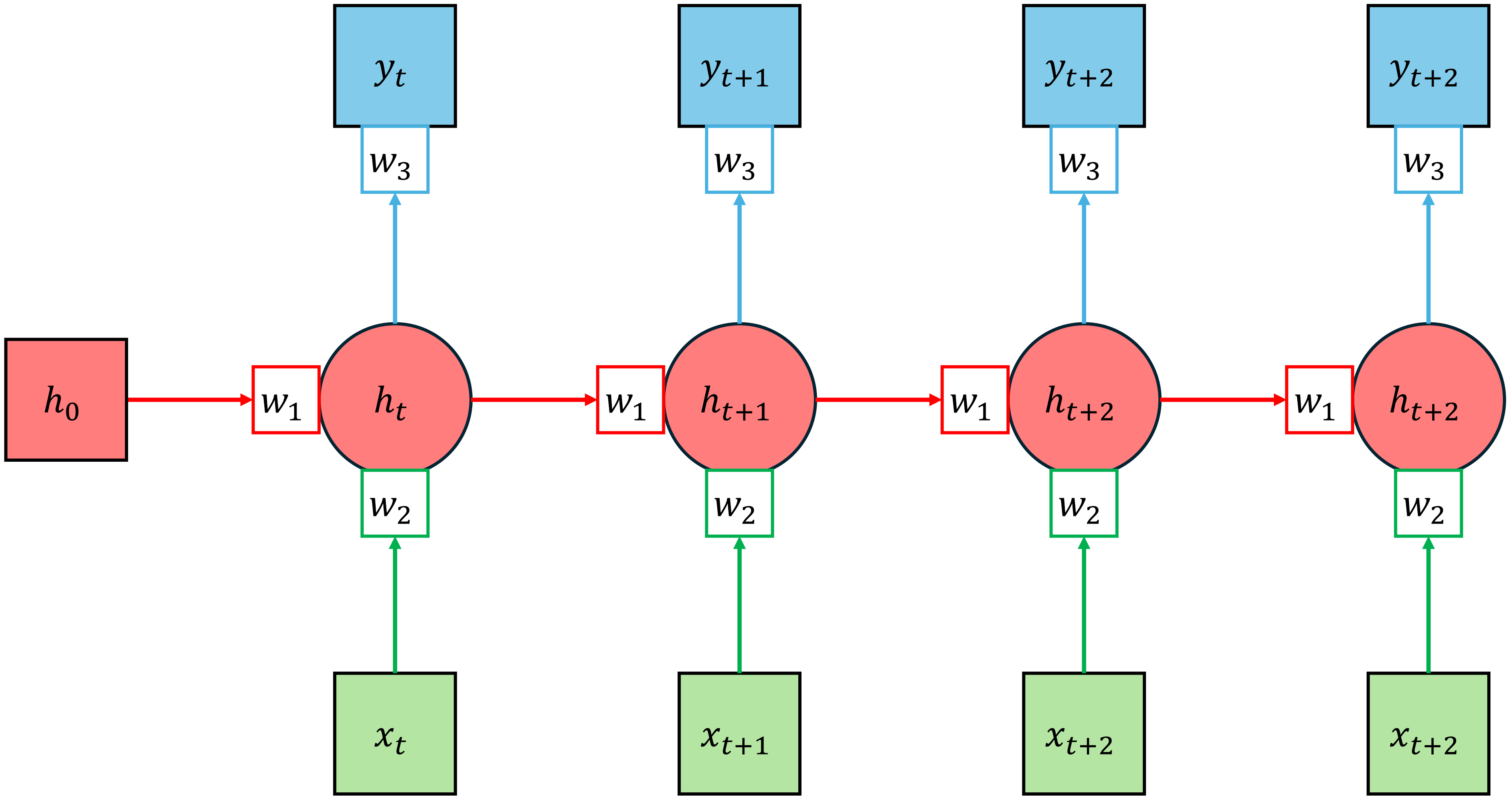

Development › Web Development
Ultimate Golang Backend: การพัฒนา Backend ด้วยภาษา Go
มาลองสร้าง API Service ด้วยภาษา Go ในรูปแบบของ Best Practices
Bestseller 4.8 ★★★★★ (37 ratings) 298 students
Created by Ruangyot Nanchiang
Last updated 4/2024 Thai
Preview this course
฿349 ฿949 63% off
Add to cart
Buy now
30-Day Money-Back Guarantee
This course includes:
13 hours on-demand video
18 articles
3 downloadable resources
Access on mobile and TV
Full lifetime access
Certificate of completion
Share Gift this course Apply Coupon
FF68803EBB75B9D891B0 is applied
Instructor coupon
Enter Coupon Apply
What you'll learn
เข้าใจหลักการการทำงานของ Website เบื้องต้น
พื้นฐานภาษา Go
OOP Concepts
SOLID Principles
พื้นฐาน SQL และ PostgreSQL
Domain Driven Design (DDD)
พัฒนา API service โดยใช้ หลักการของ Clean Architecture
การทำ Mock และ Unit testing ใน Go
การ Deploy Application ขึ้น GCP
Course content
24 sections • 115 lectures • 13h 1m total length
Expand all sections
แนะนำ Course 1 lecture • 9min


IT & Software › Other IT & Software › Microservices
เริ่มต้นสร้าง Microservices ด้วย Golang จาก Zero สู่ Hero
เรียนรู้แบบ Step by Step ในการสร้าง Microservices Application ด้วยภาษา Golang ด้วยการออกแบบจริง ลองทำจริง Deploy จริง
Bestseller 4.9 ★★★★★ (65 ratings) 572 students
Created by Ruangyot Nanchiang
Last updated 3/2024 Thai
Preview this course
฿349 ฿1,590 78% off
Add to cart
Buy now
30-Day Money-Back Guarantee
This course includes:
21 hours on-demand video
13 articles
10 downloadable resources
Access on mobile and TV
Full lifetime access
Certificate of completion
Share Gift this course Apply Coupon
59DEDE96165D1A853CB9 is applied
Instructor coupon
Enter Coupon Apply
What you'll learn
มีความเข้าใจใน Microservices Architecture เบื้องต้น
สามารถออกแบบ Microservices ได้ในรูปแบบของ Domain Driven Design
สามารถสร้าง Microservices Application ด้วยภาษา Golang ได้
สามารถใช้งานเครื่องมือที่นิยมใช้ใน Microservices ได้ เช่น Kubernetes, Kafka, gRPC, ...
สามารถ Deploy Microservices Application เบื้องต้นด้วยตัวเองได้
Course content
19 sections • 128 lectures • 21h 1m total length
Expand all sections

Recurrent Neural Network (RNN)



From the **vanishing gradient** and **Exploding Problem** problem in **RNN.**



y_t
y_{t+1}
y_{t+2}
y_{t+2}
w_3
w_3
w_3
w_3
h_0
w_1
h_t
w_1
h_{t+1}
w_1
h_{t+2}
w_1
h_{t+2}
w_2
w_2
w_2
w_2
x_t
x_{t+1}
x_{t+2}
x_{t+2}

$$\sum_{i=1}^{n}\frac{\partial CE_i}{\partial w_n}=\sum_{i=1}^{n}\frac{\partial CE_i}{\partial S_{fi}}\cdot\frac{\partial S_{fi}}{\partial h_{fi}}\cdot\frac{\partial h_{fi}}{\partial h_n}\cdot\frac{\partial h_n}{\partial w_n}+\sum_{i=1}^{n}\frac{\partial CE_i}{\partial S_{fi}}\cdot\frac{\partial S_{fi}}{\partial h_{fi}}\cdot\frac{\partial h_{fi}}{\partial h_2}\cdot\frac{\partial h_n}{\partial h_{n-1}}\cdot\frac{\partial h_{n-1}}{\partial w_n}$$

$$+\sum_{i=1}^{n}\frac{\partial CE_i}{\partial S_{fi}}\cdot\frac{\partial S_{fi}}{\partial h_{fi}}\cdot\frac{\partial h_{fi}}{\partial h_n}\cdot\frac{\partial h_n}{\partial h_{n-1}}\cdot\frac{\partial h_{n-1}}{\partial h_{n-2}}\cdot\frac{\partial h_{n-2}}{\partial w_n}+$$

$$+\sum_{i=1}^{n}\frac{\partial CE_i}{\partial S_{fi}}\cdot\frac{\partial S_{fi}}{\partial h_{fi}}\cdot\frac{\partial h_{fi}}{\partial h_n}\cdot\frac{\partial h_n}{\partial h_{n-1}}\cdot\frac{\partial h_{n-1}}{\partial h_{n-2}}\cdot\frac{\partial h_{n-2}}{\partial h_{n-3}}\cdot\frac{\partial h_{n-3}}{\partial w_n}$$

$$+\cdots$$

Vanishing Gradient Problem

$$\sum_{i=1}^{n}\frac{\partial CE_i}{\partial w_n}=\sum_{i=1}^{n}\frac{\partial CE_i}{\partial S_{fi}}\cdot\frac{\partial S_{fi}}{\partial h_{fi}}\cdot\frac{\partial h_{fi}}{\partial h_n}\cdot\frac{\partial h_n}{\partial w_n}+\sum_{i=1}^{n}\frac{\partial CE_i}{\partial S_{fi}}\cdot\frac{\partial S_{fi}}{\partial h_{fi}}\cdot\frac{\partial h_{fi}}{\partial h_2}\cdot\frac{\partial h_n}{\partial h_{n-1}}\cdot\frac{\partial h_{n-1}}{\partial w_n}$$

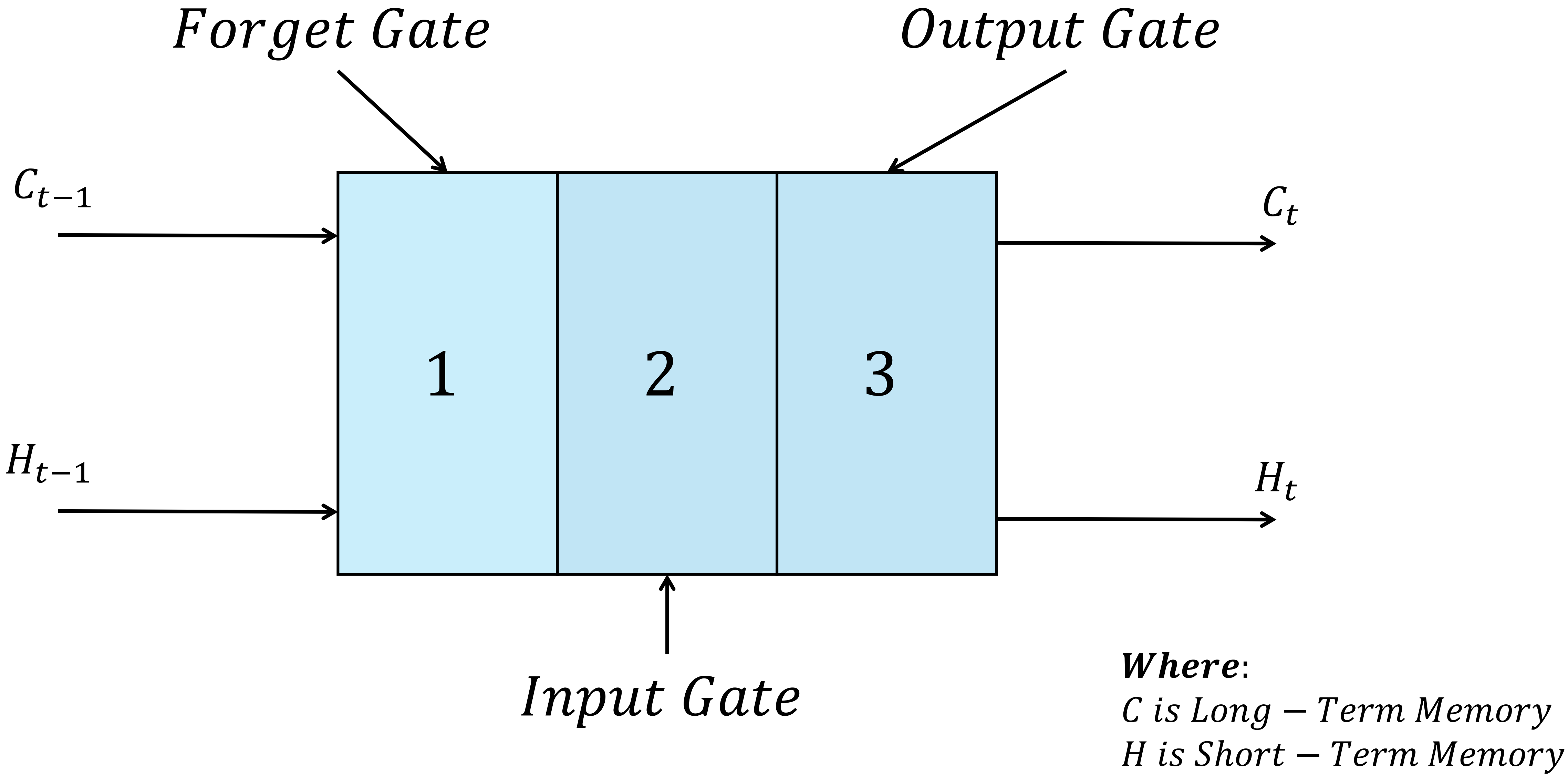
$$+\sum_{i=1}^{n}\frac{\partial CE_i}{\partial S_{fi}}\cdot\frac{\partial S_{fi}}{\partial h_{fi}}\cdot\frac{\partial h_{fi}}{\partial h_n}\cdot\frac{\partial h_n}{\partial h_{n-1}}\cdot\frac{\partial h_{n-1}}{\partial h_{n-2}}\cdot\frac{\partial h_{n-2}}{\partial w_n}+$$

$$+\sum_{i=1}^{n}\frac{\partial CE_i}{\partial S_{fi}}\cdot\frac{\partial S_{fi}}{\partial h_{fi}}\cdot\frac{\partial h_{fi}}{\partial h_n}\cdot\frac{\partial h_n}{\partial h_{n-1}}\cdot\frac{\partial h_{n-1}}{\partial h_{n-2}}\cdot\frac{\partial h_{n-2}}{\partial h_{n-3}}\cdot\frac{\partial h_{n-3}}{\partial w_n}$$

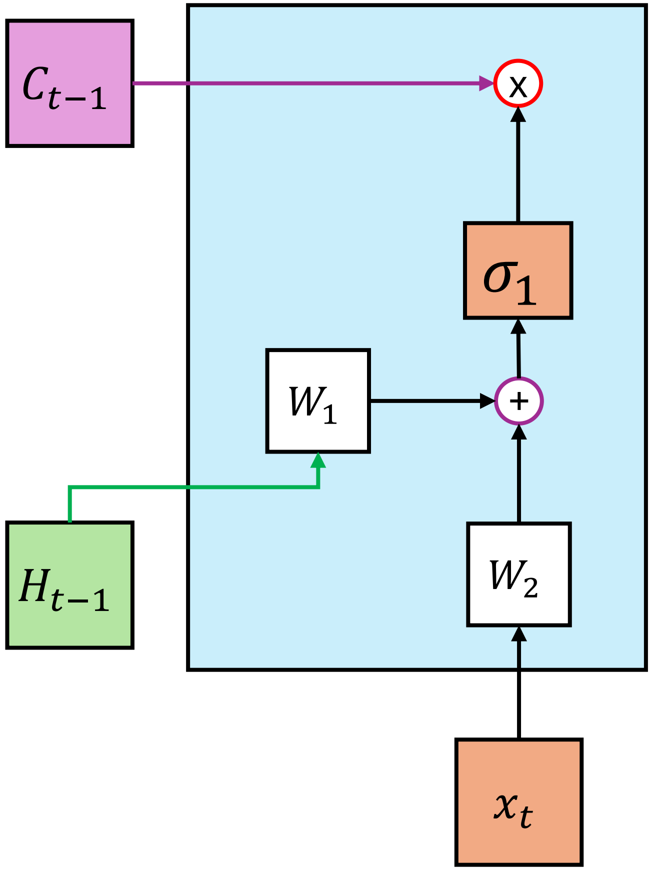
$$+\cdots$$

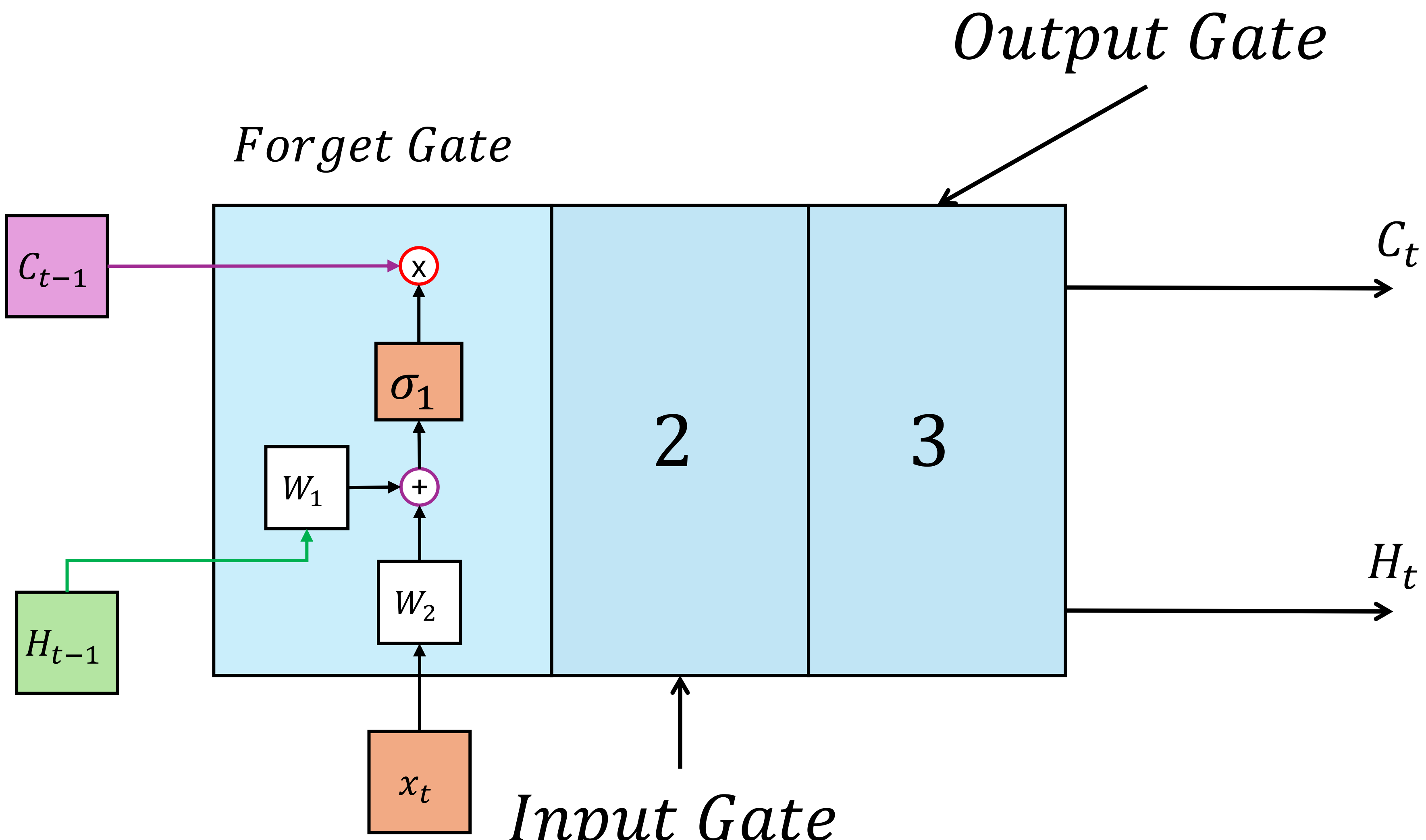
Exploding Problem

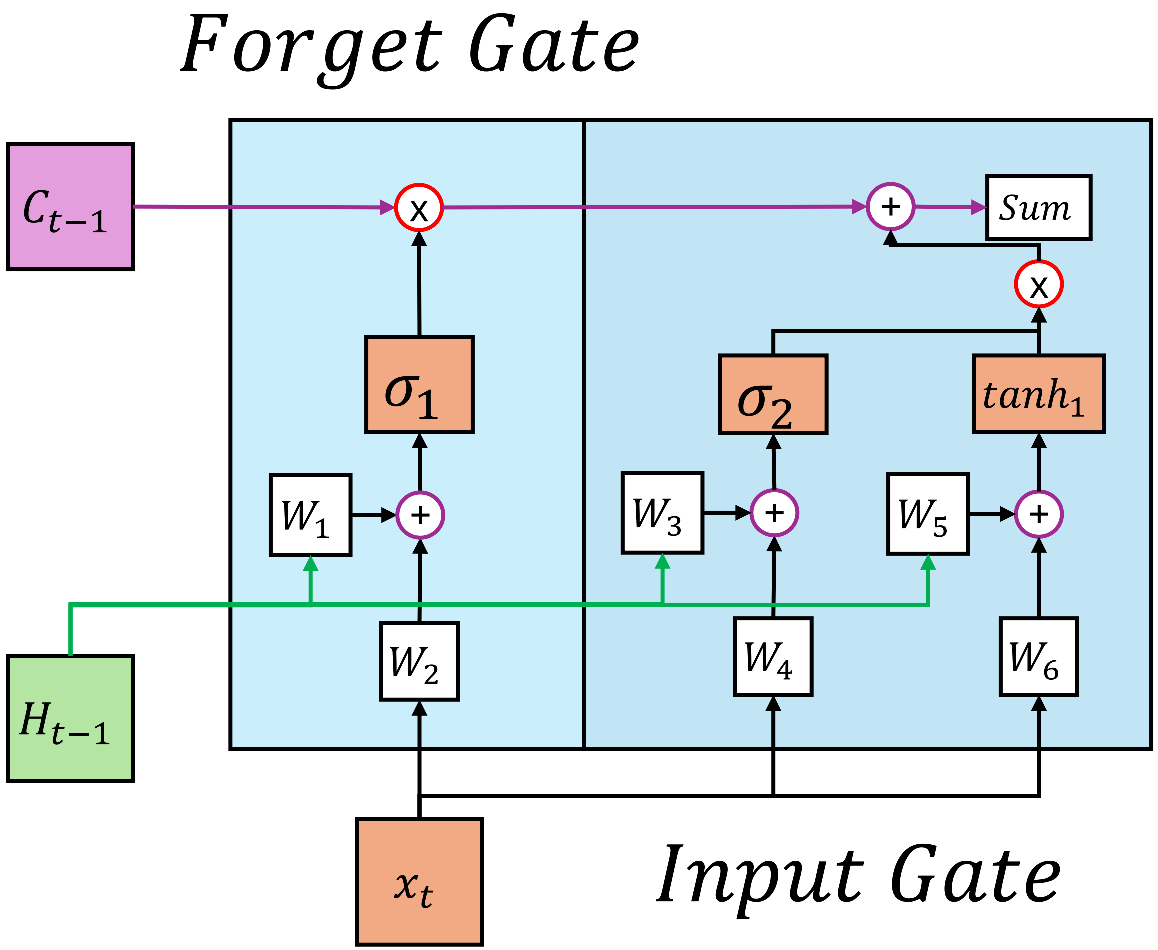
Now, we’re going to change the hidden state into something called **“Cell”**

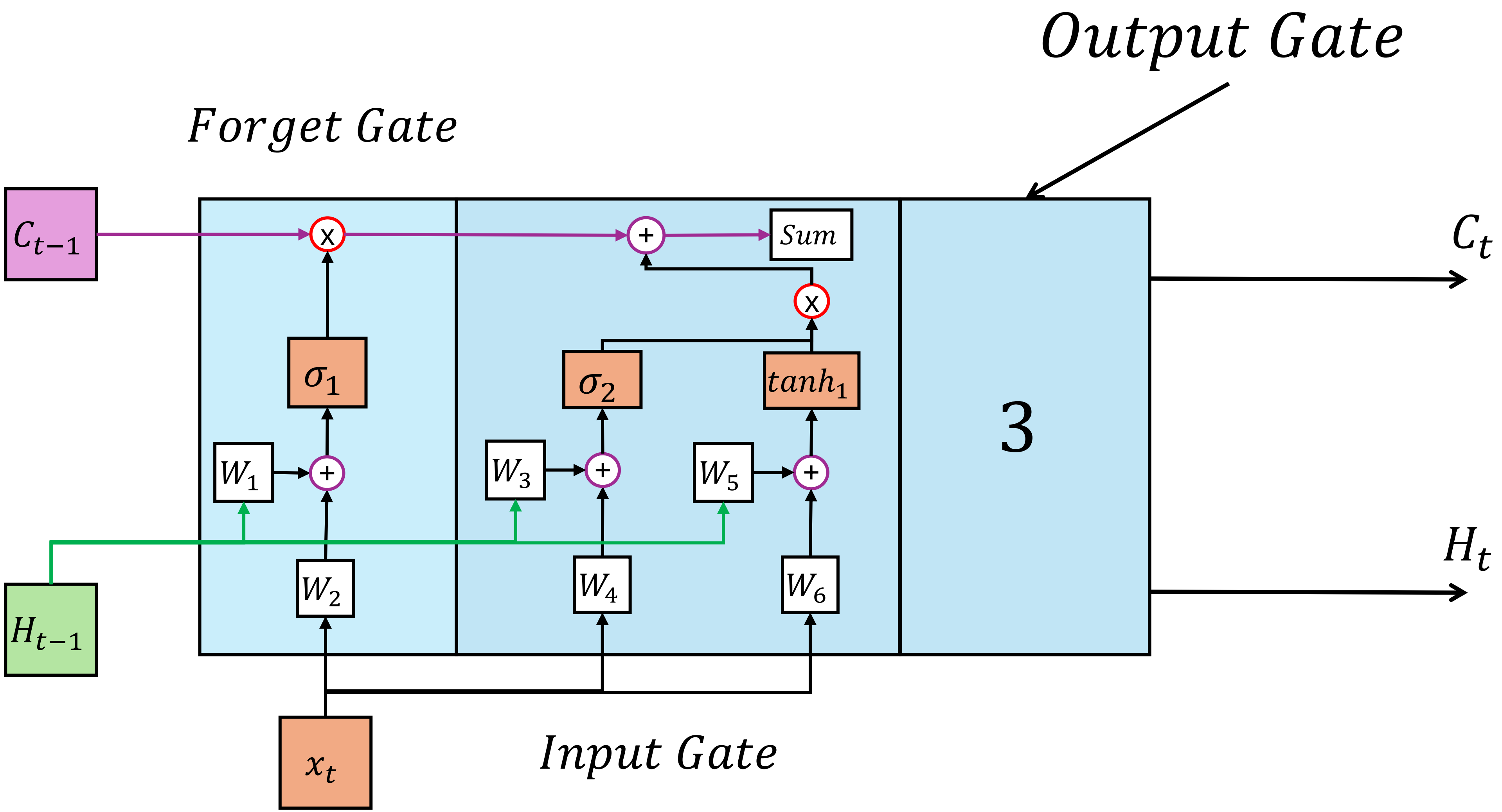

Forget Gate
Output Gate
C_{t-1}
C_t
1
2
3
H_{t-1}
H_t
Input Gate
Where:
C is Long − Term Memory
H is Short − Term Memory

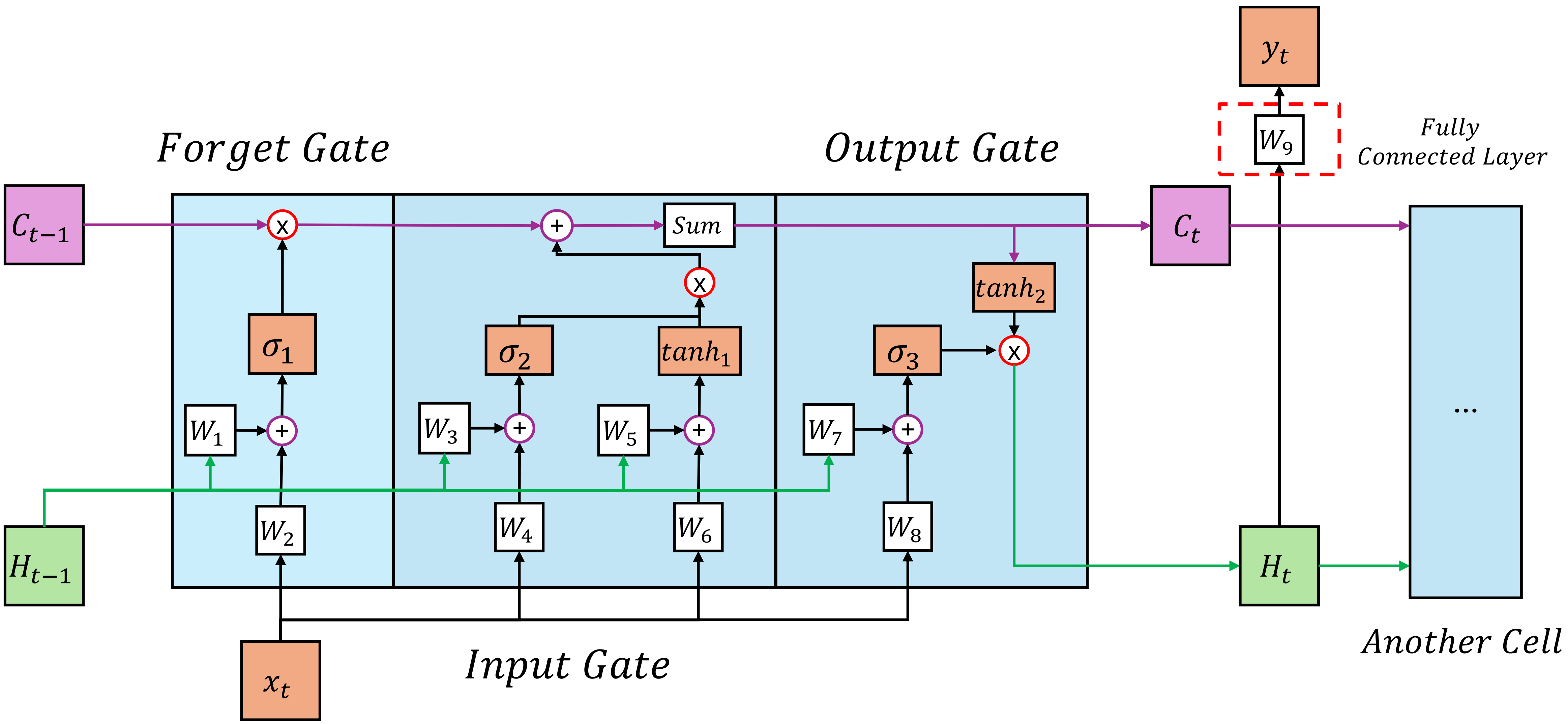
$Forget\ Gate$




Output Gate
Forget Gate
C_{t-1}
x
σ_1
W_1
+
W_2
H_{t-1}
x_t
2
3
C_t
H_t
Input Gate

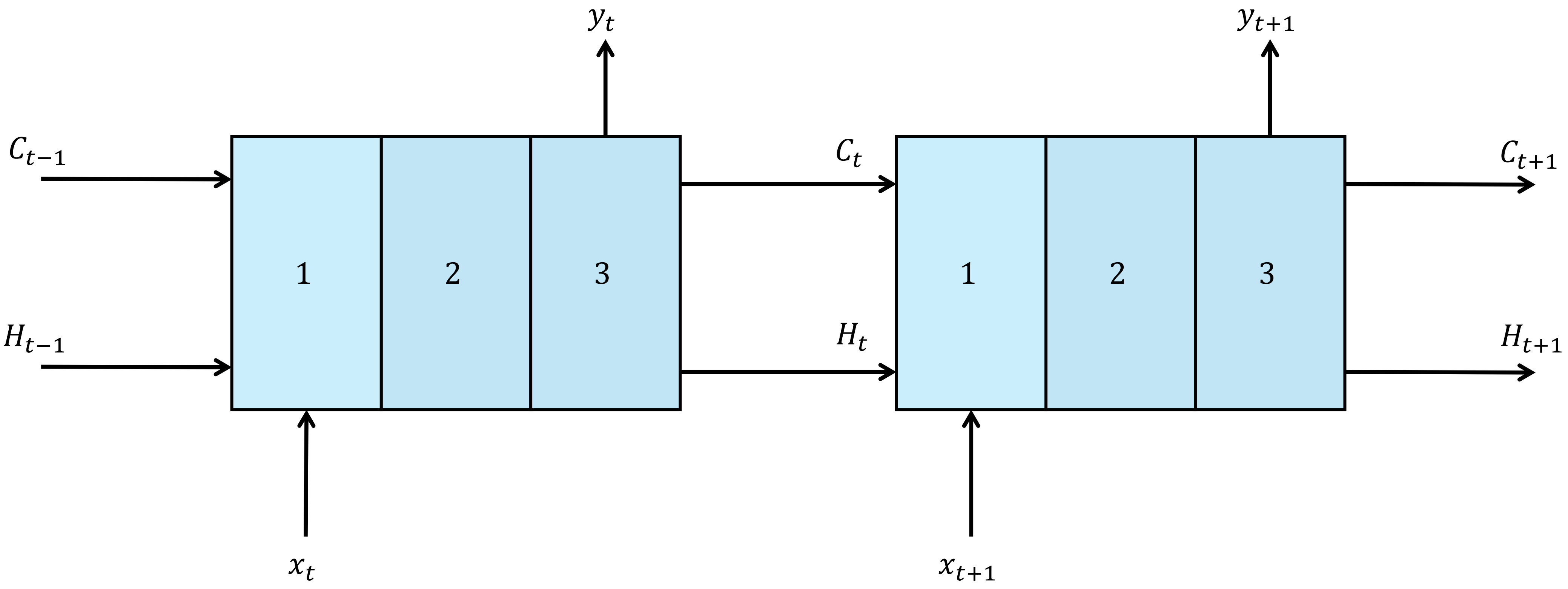

Forget Gate
C_{t-1}
x
+
Sum
x
σ_1
σ_2
$tanh_1$
W_1
+
W_3
+
W_5
+
W_2
W_4
W_6
H_{t-1}
x_t
Input Gate


Output Gate
Forget Gate
C_{t-1}
x
+
Sum
x
σ_1
σ_2
$tanh_1$
3
C_t
W_1
+
W_3
+
W_5
+
H_t
W_2
W_4
W_6
H_{t-1}
x_t
Input Gate

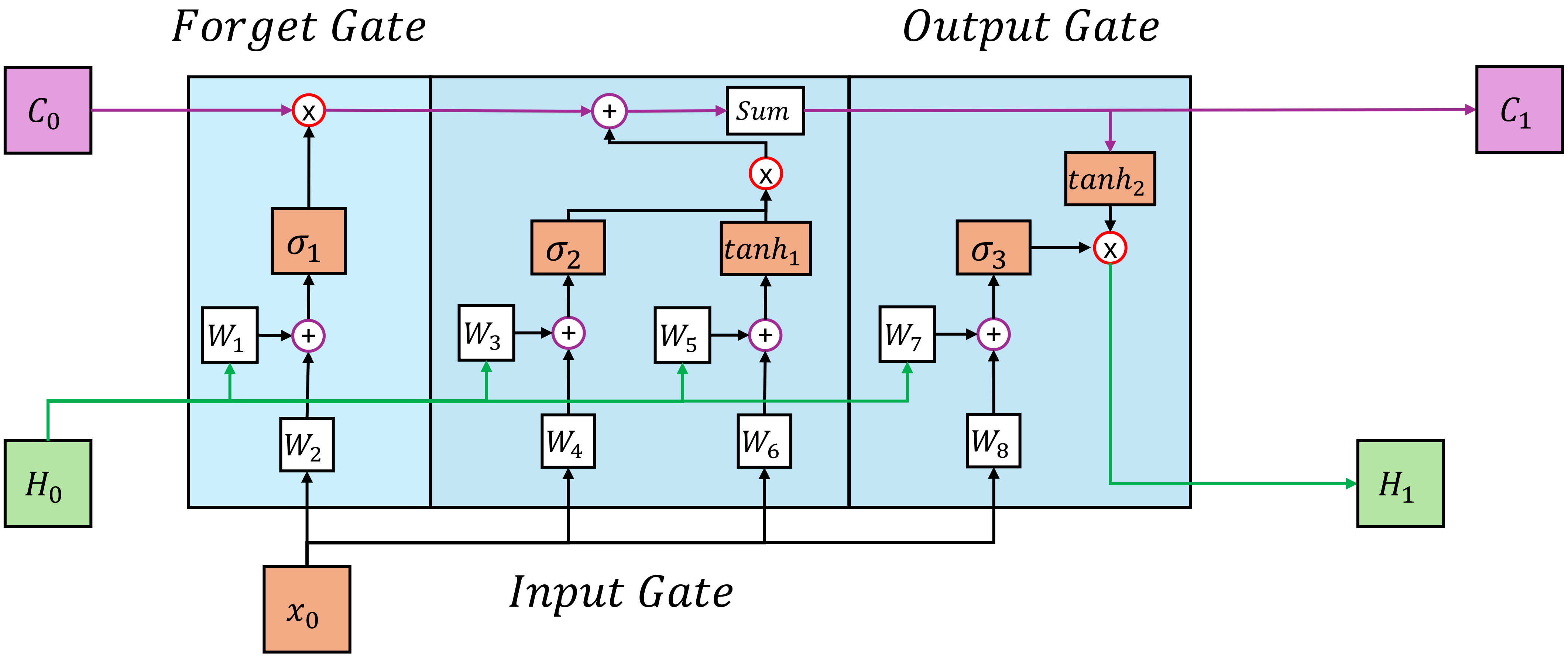


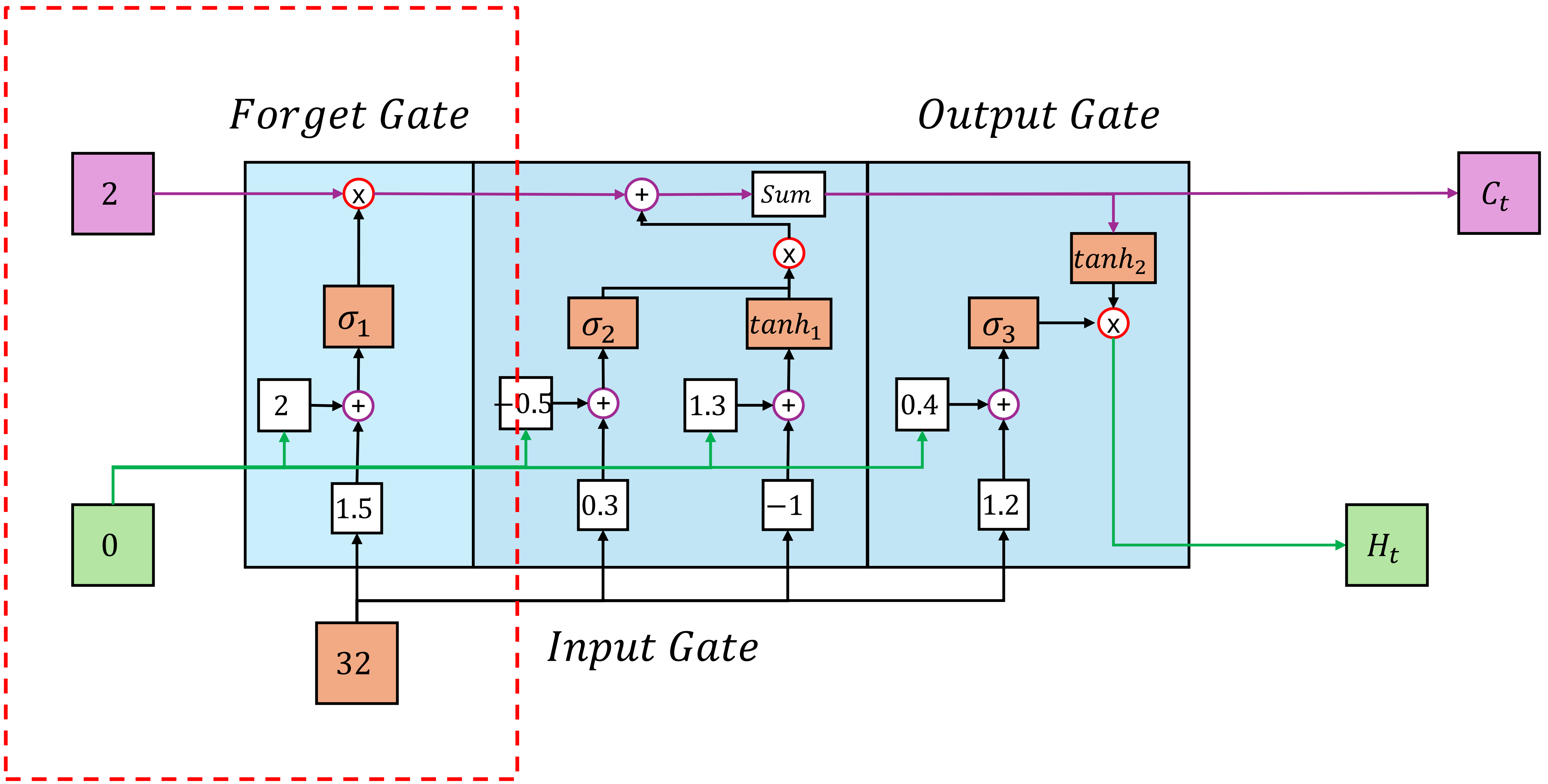
Forget Gate
Output Gate
Input Gate
Fully Connected Layer
Another Cell
C_{t-1}
H_{t-1}
x_t
C_t
H_t
y_t
W_1
W_2
W_3
W_4
W_5
W_6
W_7
W_8
W_9
σ_1
σ_2
σ_3
$tanh_1$
$tanh_2$
Sum
...

Let’s chain a cell together.

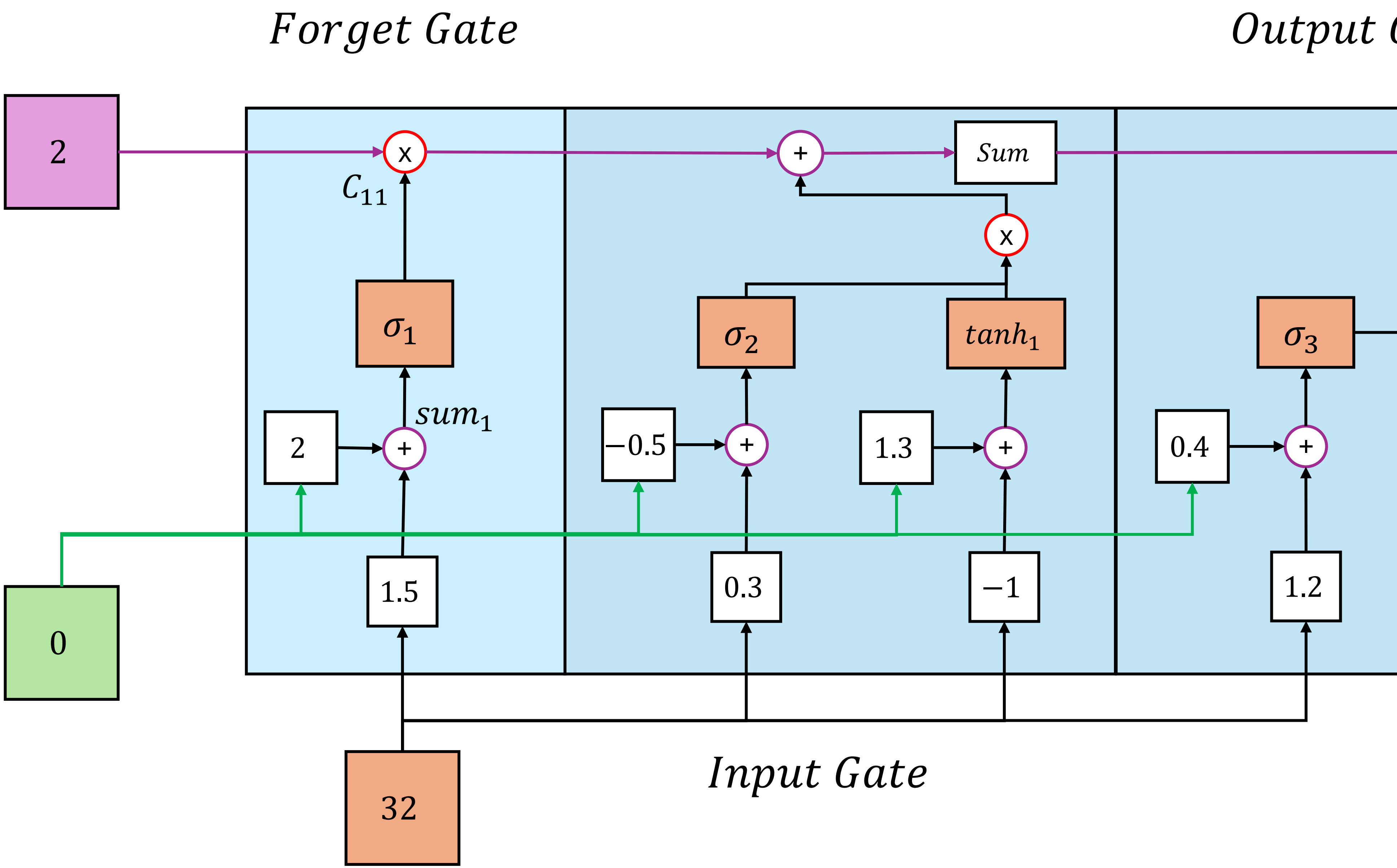


Let’s **forward propagation**.

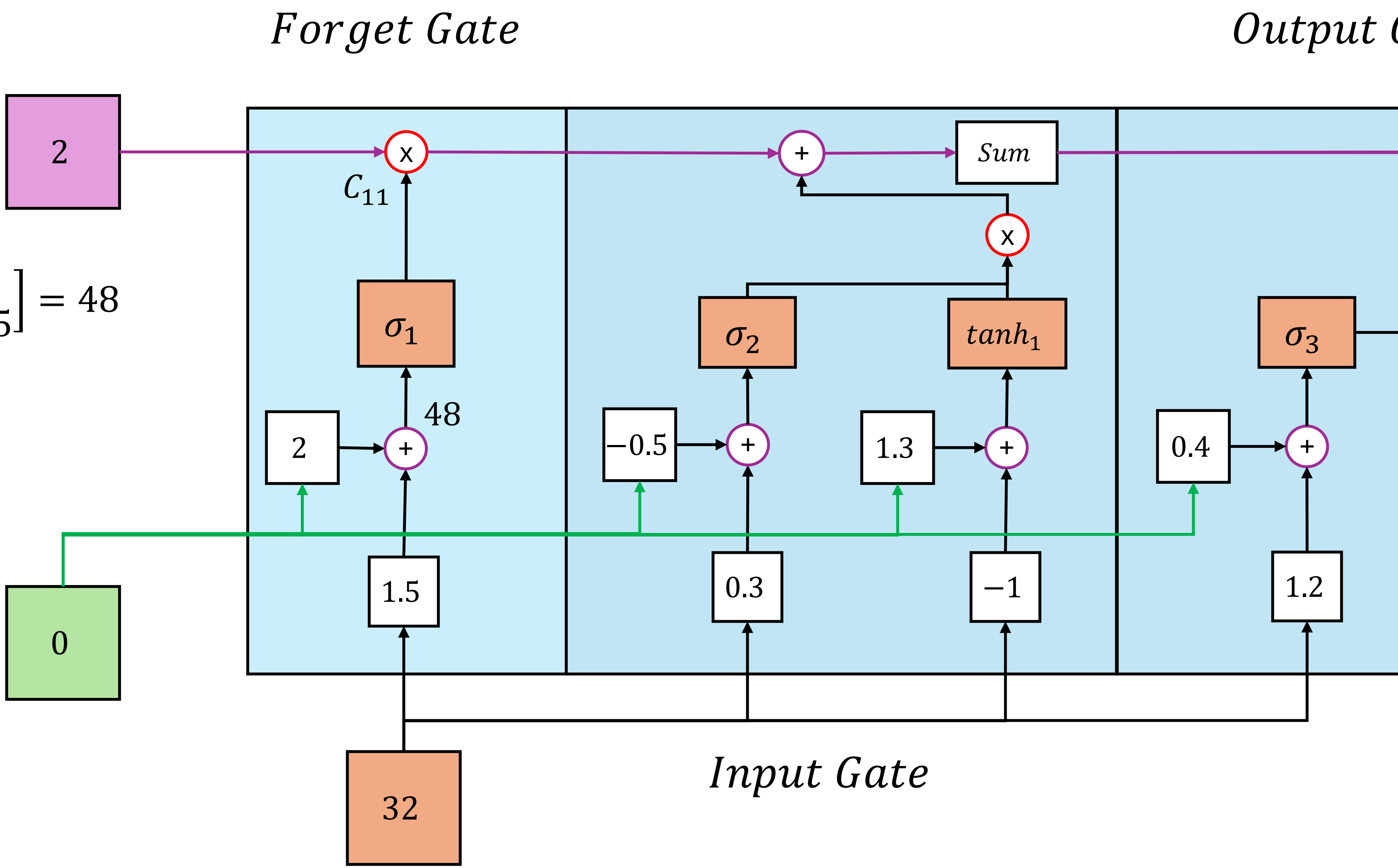

Forget Gate
Output Gate
Input Gate
C_0
C_1
H_0
H_1
x_0
Sum
σ_1
σ_2
σ_3
$tanh_1$
$tanh_2$
W_1
W_2
W_3
W_4
W_5
W_6
W_7
W_8



Forget Gate
Output Gate
Input Gate
2
Sum
x
+
C_t
$tanh_2$
σ_1
σ_2
$tanh_1$
σ_3
2
−0.5
1.3
0.4
0
1.5
0.3
−1
1.2
H_t
32

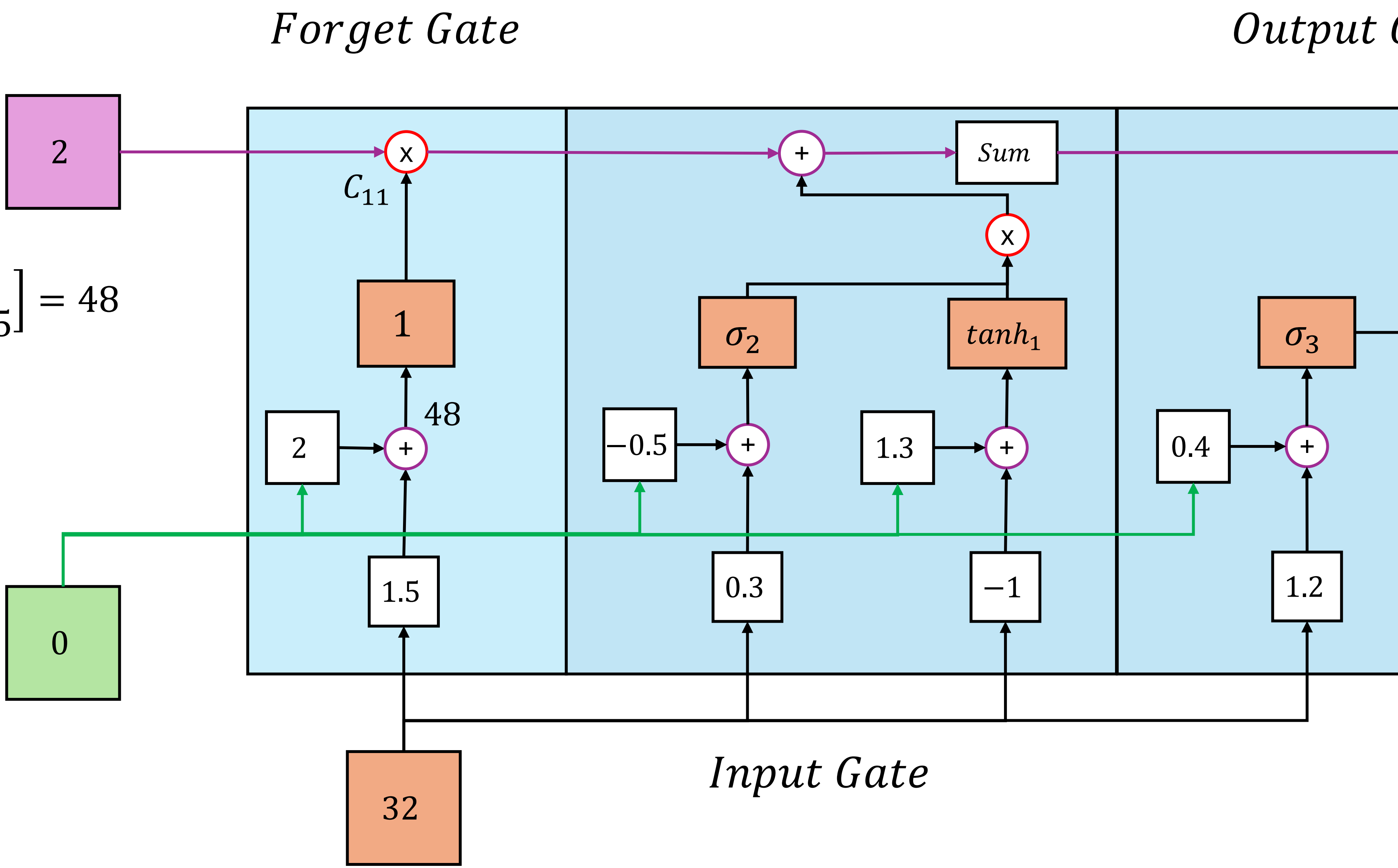

Forget Gate
Output
2
x
C_{11}
+
Sum
x
σ_1
σ_2
$tanh_1$
σ_3
sum_1
2
+
−0.5
+
1.3
+
0.4
+
1.5
0.3
−1
1.2
0
32
Input Gate

$$sum_1 = [0 \quad 32] \times \begin{bmatrix} 2 \\ 1.5 \end{bmatrix} = 48$$



$$sum_1 = [0 \quad 32] \times \begin{bmatrix} 2 \\ 1.5 \end{bmatrix} = 48$$

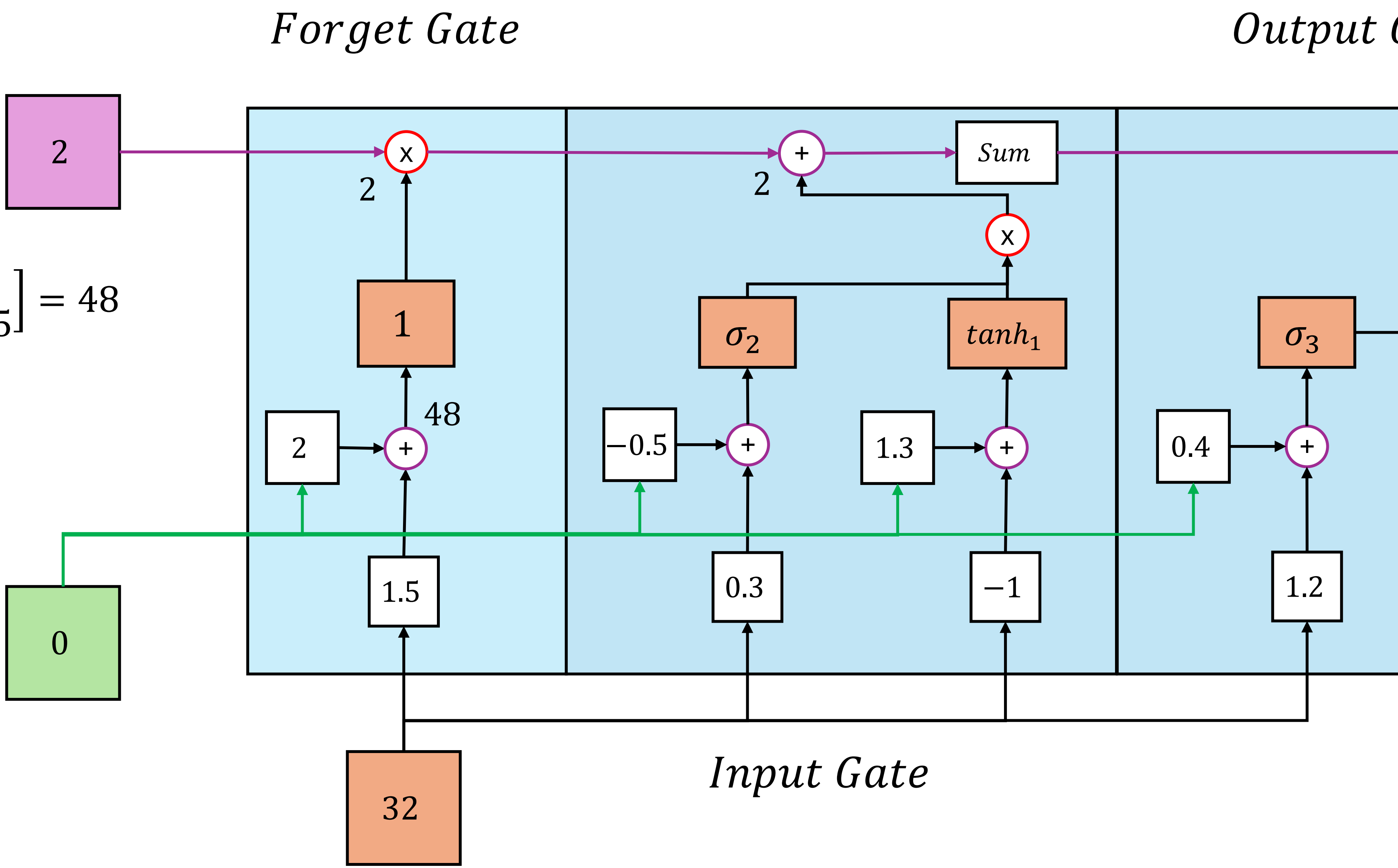
$$\sigma_1 = \frac{1}{(1 - e^{-48})} = 1$$



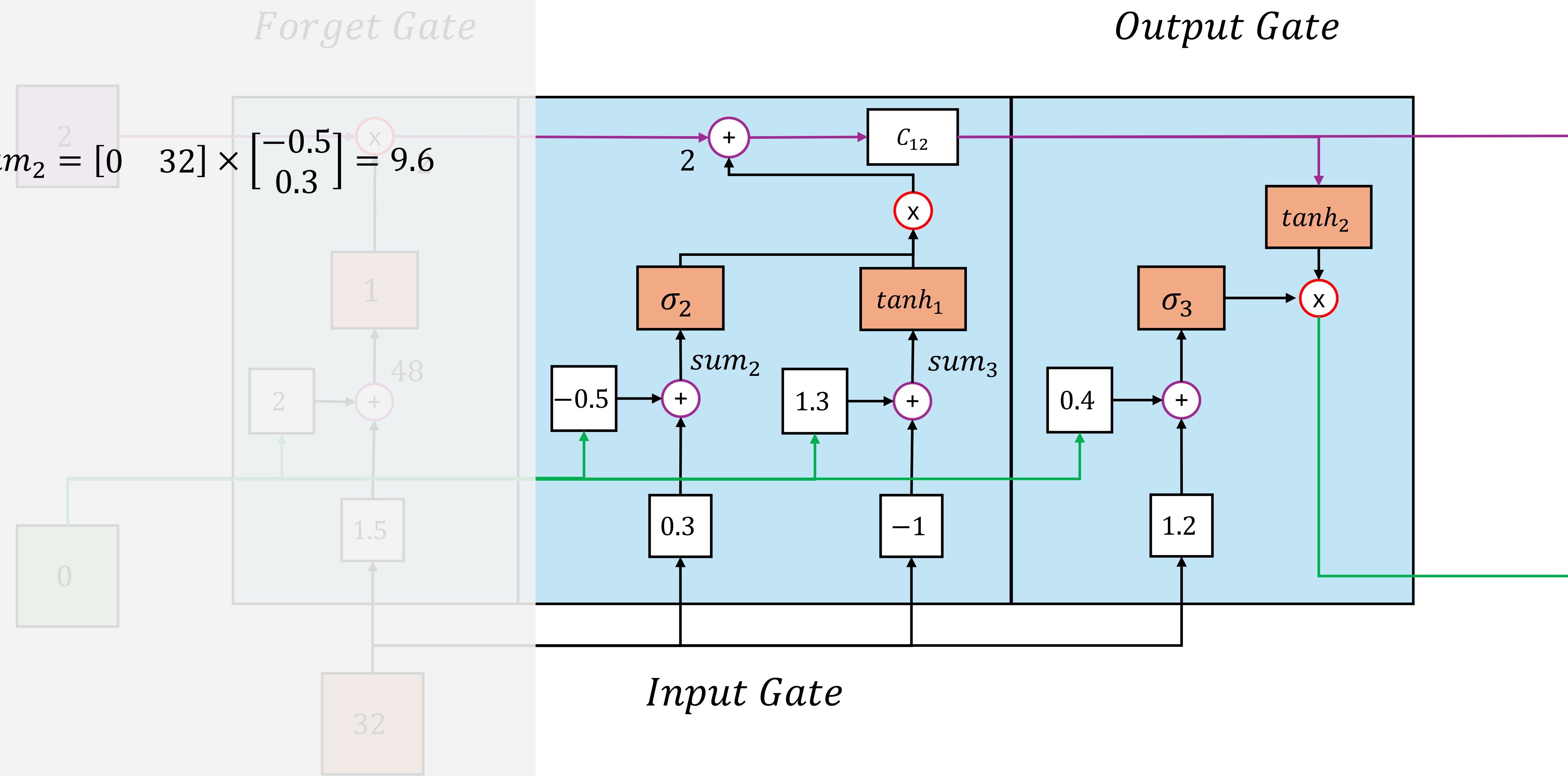
$$sum_1 = \begin{bmatrix} 0 & 32 \end{bmatrix} \times \begin{bmatrix} 2 \\ 1.5 \end{bmatrix} = 48$$

$$\sigma_1 = \frac{1}{(1 - e^{-48})} = 1$$

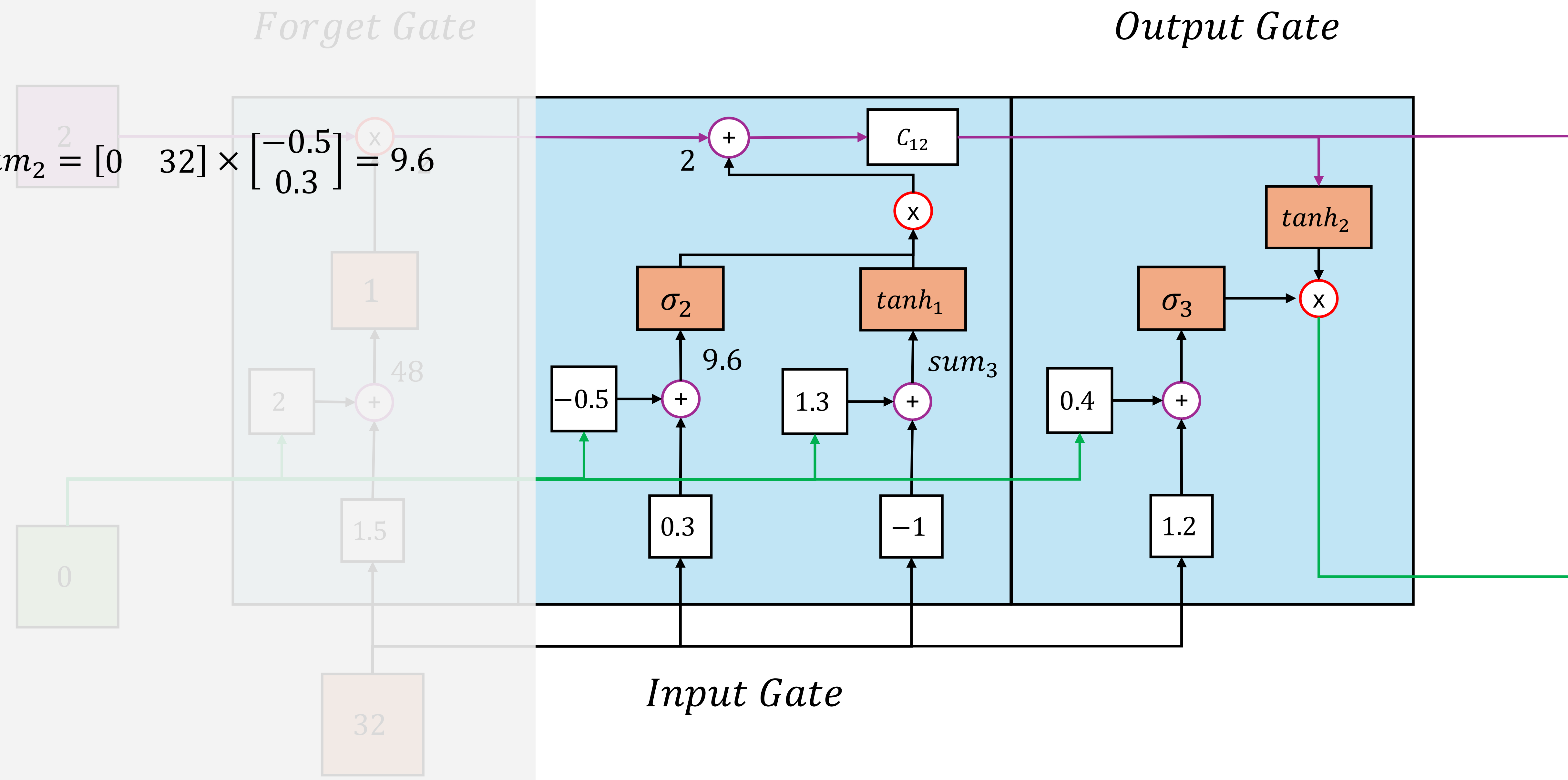
$$C_{11} = (2)(1) = 2$$



$$sum_2 = [0 \quad 32] \times \begin{bmatrix} -0.5 \\ 0.3 \end{bmatrix} = 9.6$$

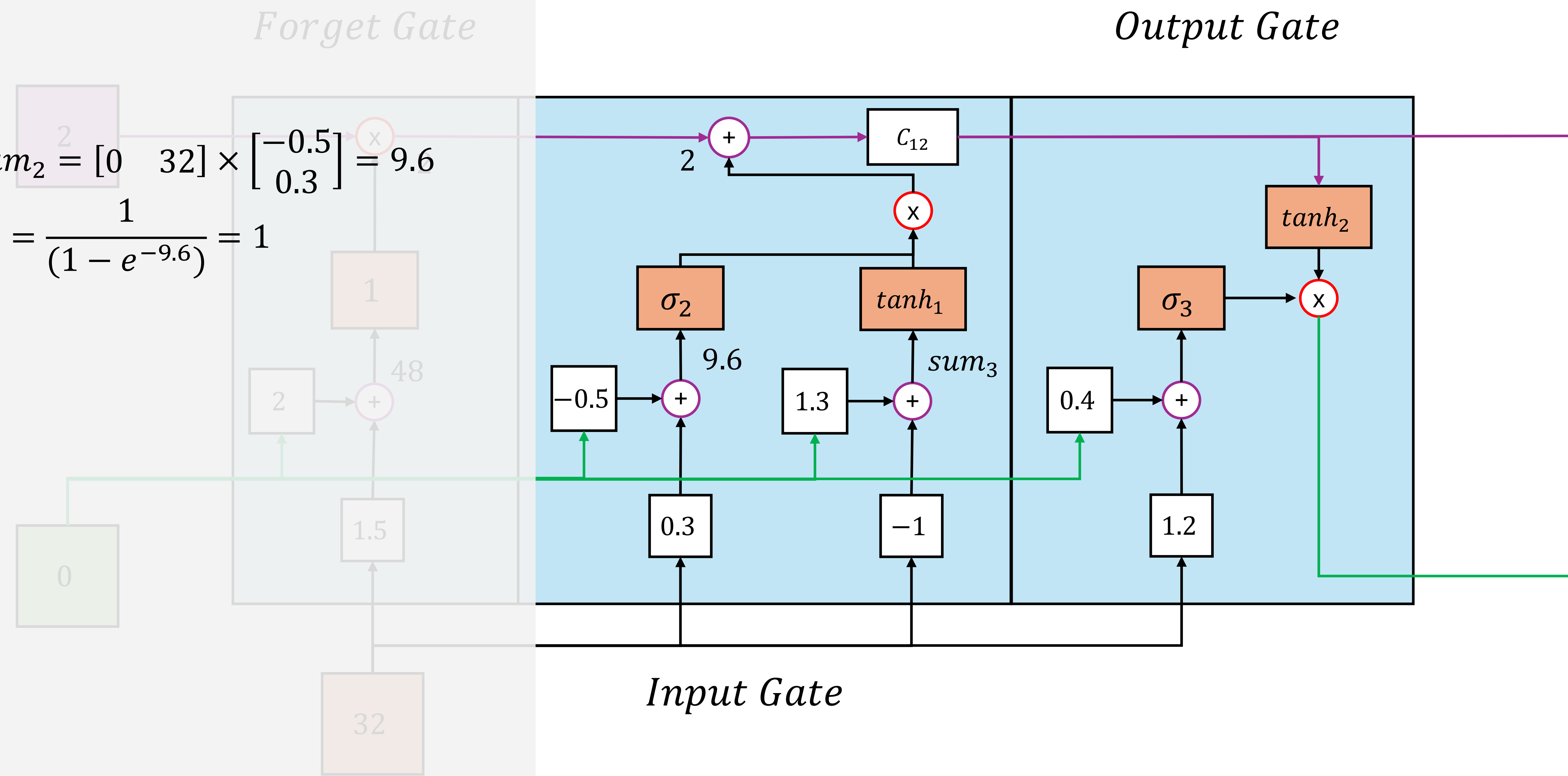


$$sum_2 = [0 \quad 32] \times \begin{bmatrix} -0.5 \\ 0.3 \end{bmatrix} = 9.6$$



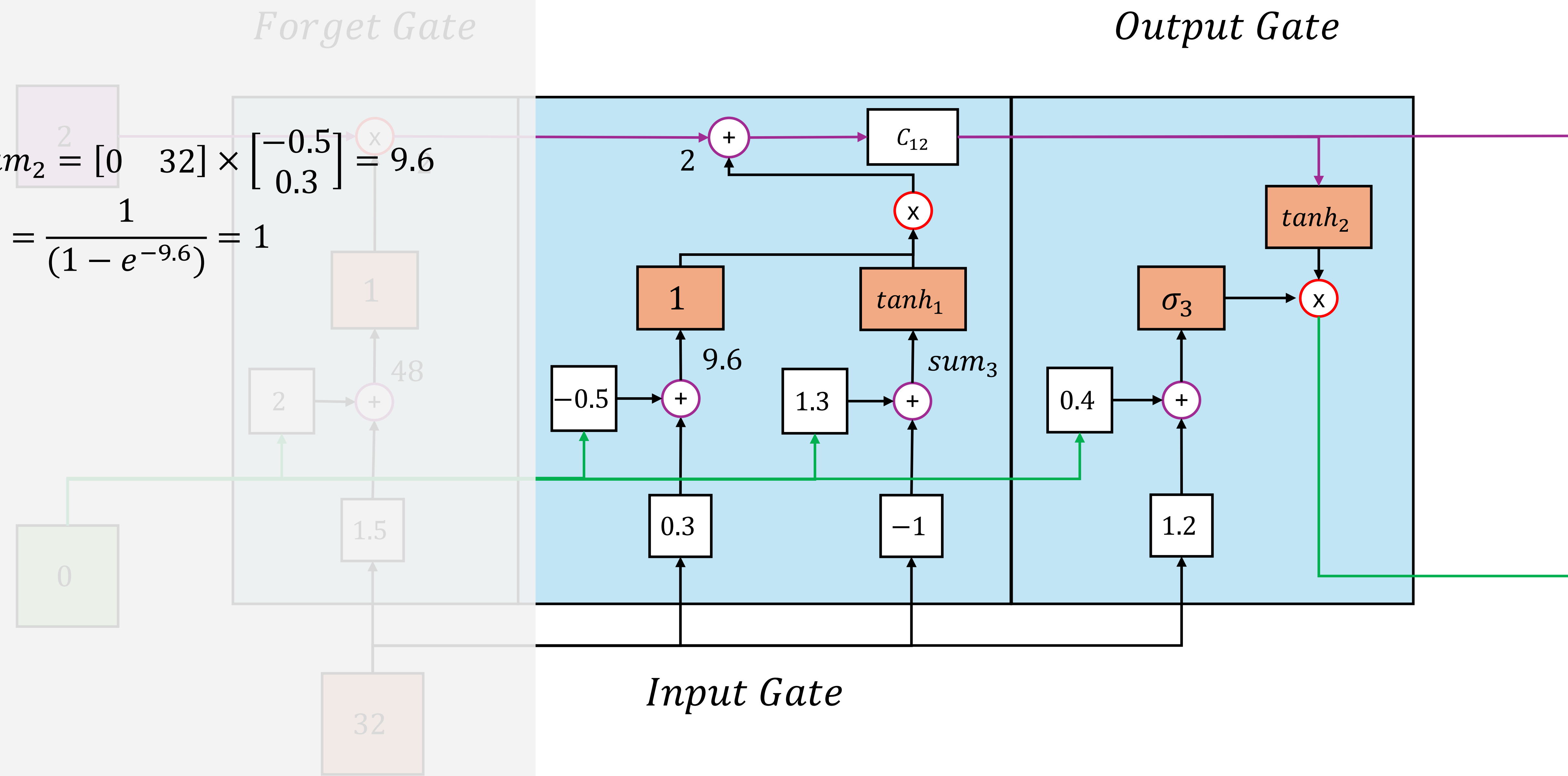
$$sum_2 = [0 \quad 32] \times \begin{bmatrix} -0.5 \\ 0.3 \end{bmatrix} = 9.6$$

$$\sigma_2 = \frac{1}{(1 - e^{-9.6})} = 1$$



$$sum_2 = [0 \quad 32] \times \begin{bmatrix} -0.5 \\ 0.3 \end{bmatrix} = 9.6$$

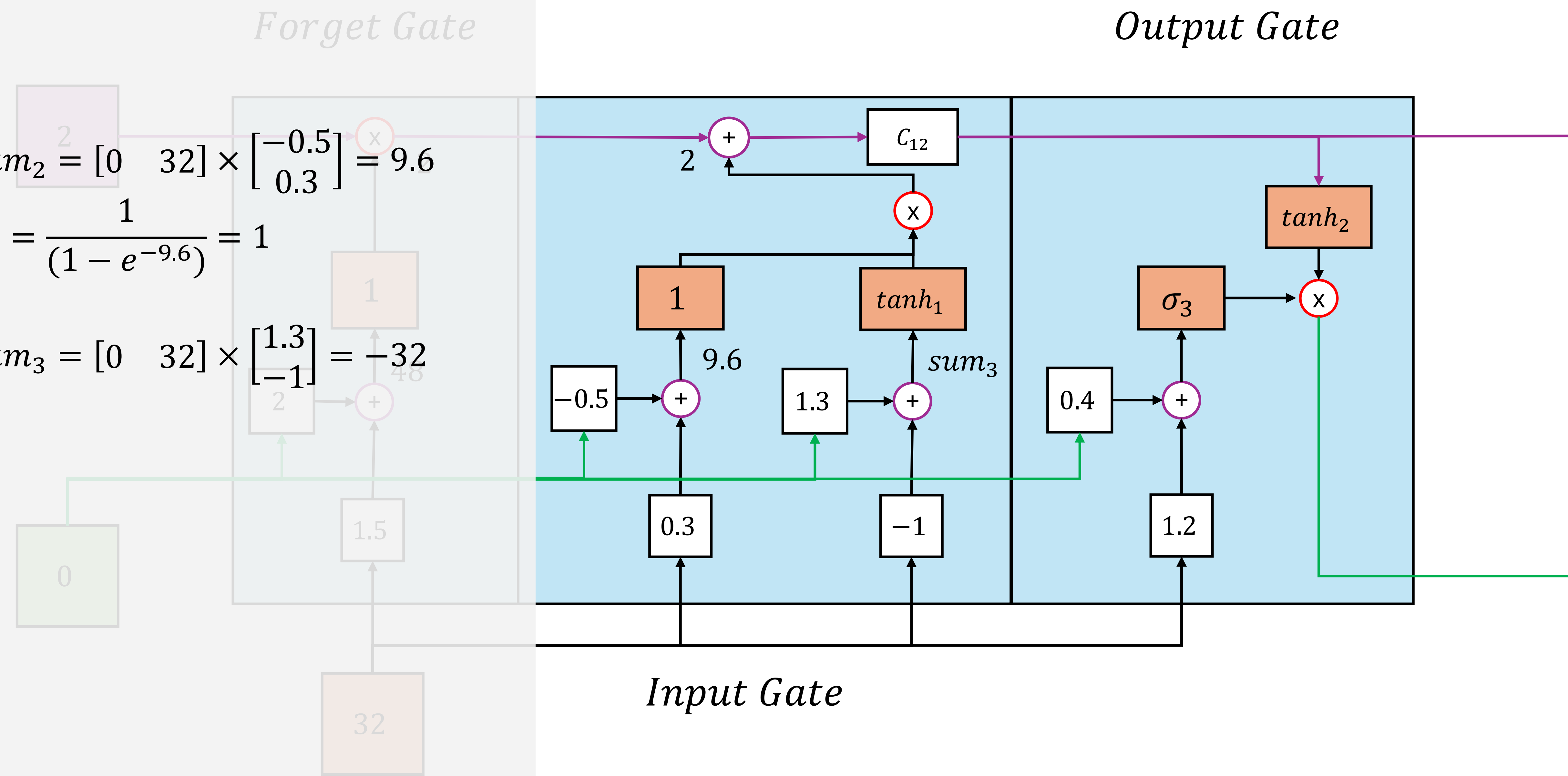
$$\sigma_2 = \frac{1}{(1 - e^{-9.6})} = 1$$



$$sum_2 = \begin{bmatrix} 0 & 32 \end{bmatrix} \times \begin{bmatrix} -0.5 \\ 0.3 \end{bmatrix} = 9.6$$

$$\sigma_2 = \frac{1}{(1 - e^{-9.6})} = 1$$

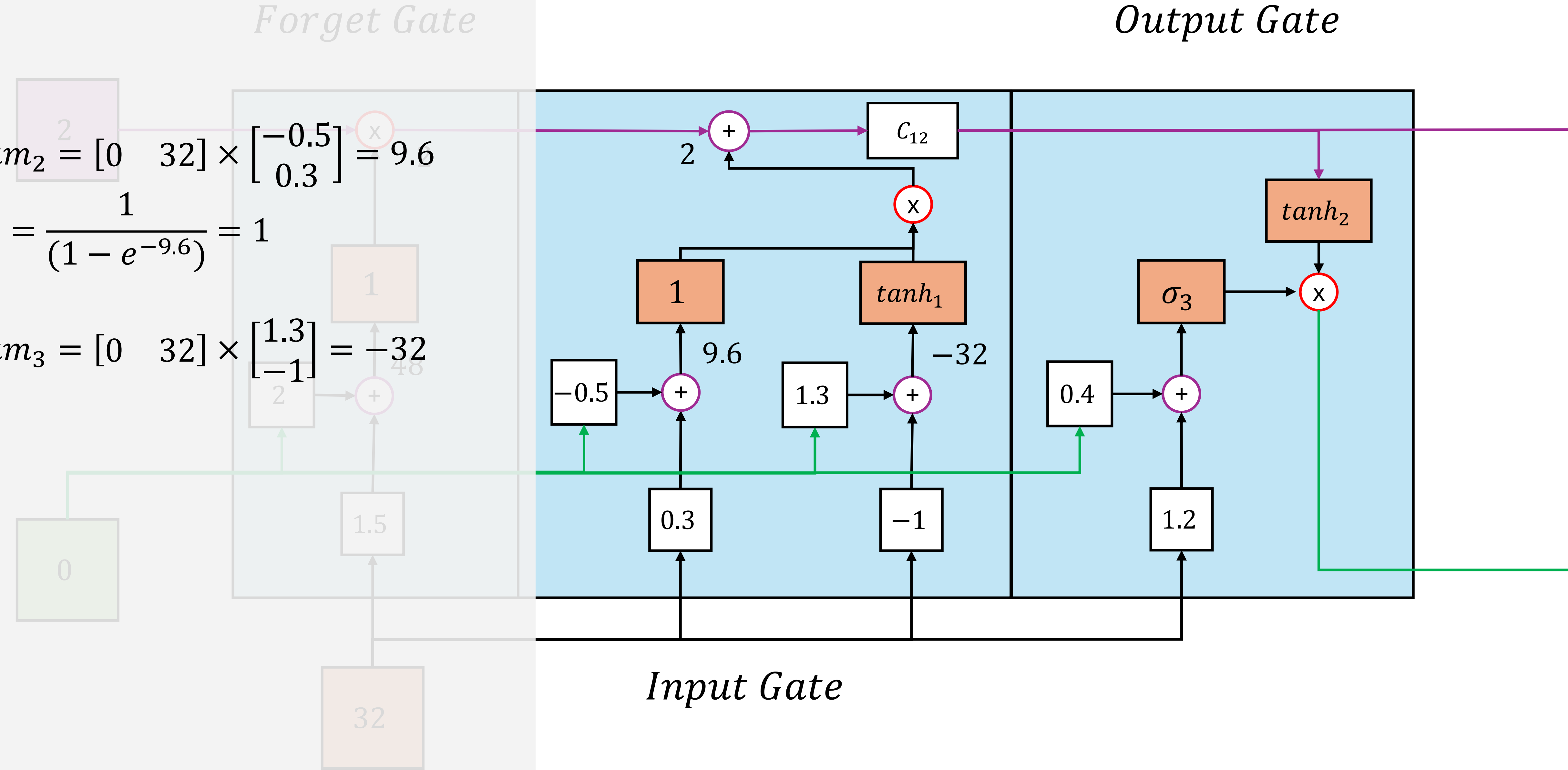
$$sum_3 = \begin{bmatrix} 0 & 32 \end{bmatrix} \times \begin{bmatrix} 1.3 \\ -1 \end{bmatrix} = -32$$



$$sum_2 = [0 \quad 32] \times \begin{bmatrix} -0.5 \\ 0.3 \end{bmatrix} = 9.6$$

$$\sigma_2 = \frac{1}{(1 - e^{-9.6})} = 1$$

$$sum_3 = [0 \quad 32] \times \begin{bmatrix} 1.3 \\ -1 \end{bmatrix} = -32$$

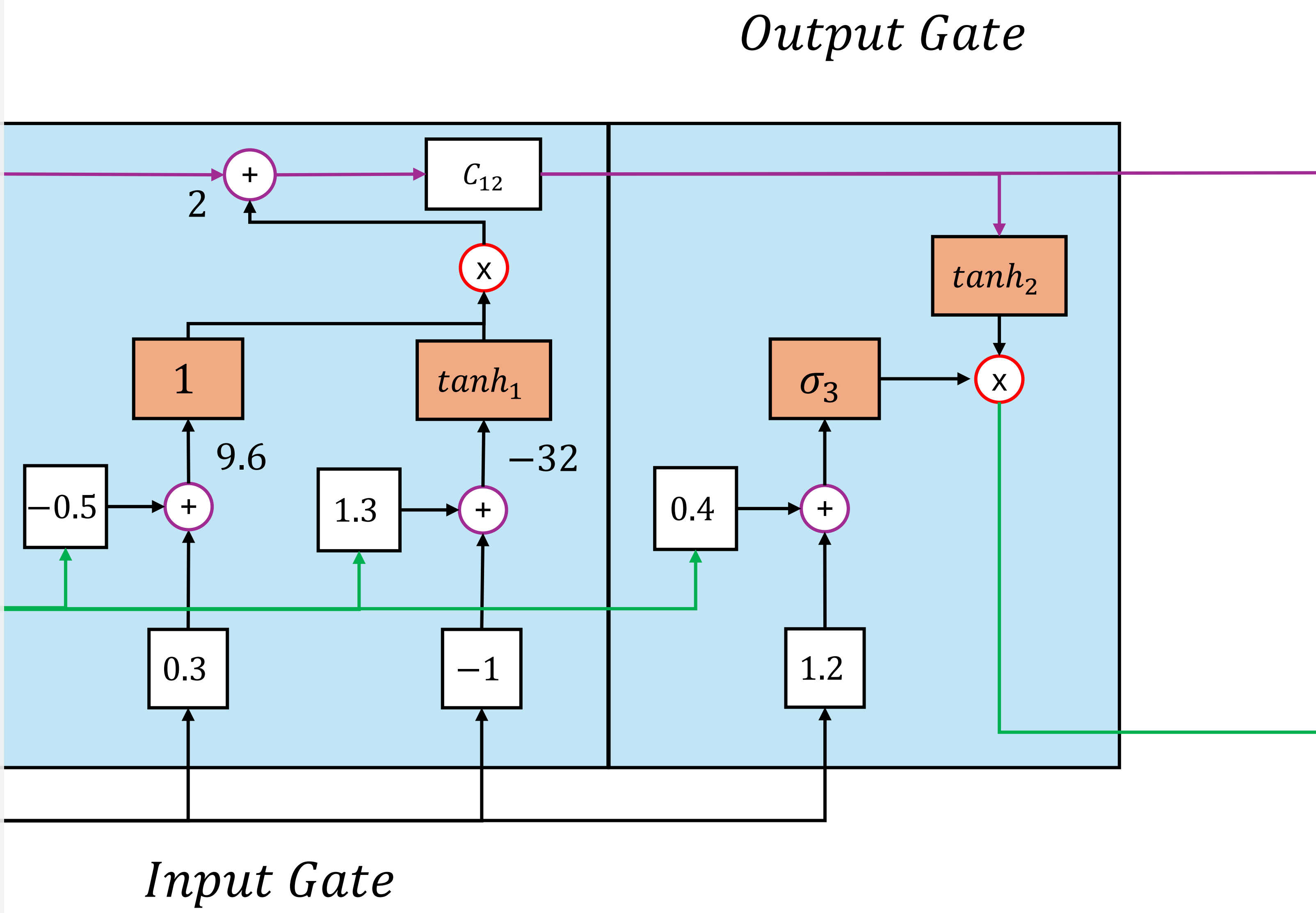


$$sum_2 = [0 \quad 32] \times \begin{bmatrix} -0.5 \\ 0.3 \end{bmatrix} = 9.6$$

$$\sigma_2 = \frac{1}{(1 - e^{-9.6})} = 1$$

$$sum_3 = [0 \quad 32] \times \begin{bmatrix} 1.3 \\ -1 \end{bmatrix} = -32$$

$$tanh_1 = \frac{(e^{(-32)} - e^{-(-32)})}{(e^{(-32)} + e^{-(-32)})} = -1$$

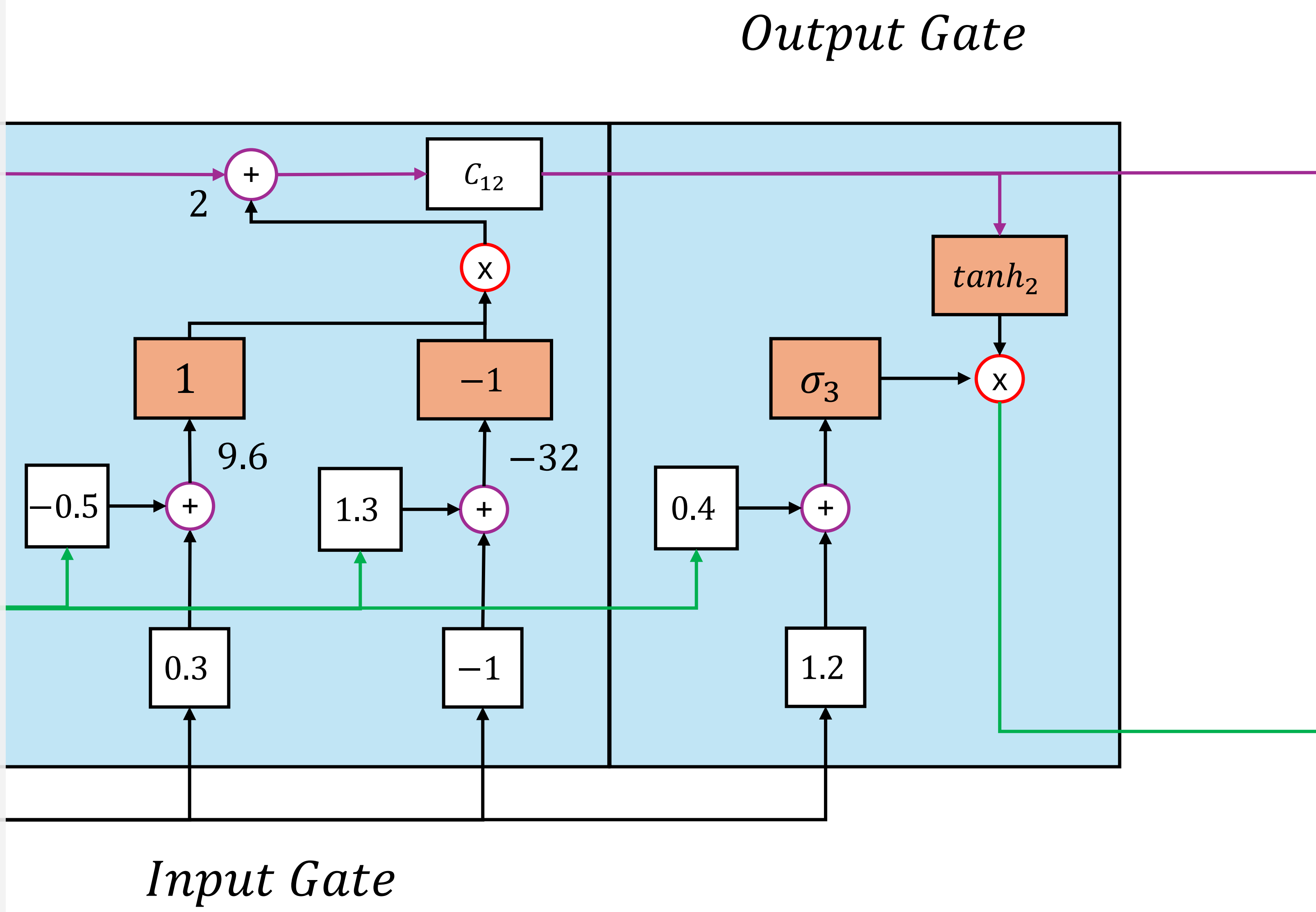


$$sum_2 = [0 \quad 32] \times \begin{bmatrix} -0.5 \\ 0.3 \end{bmatrix} = 9.6$$

$$\sigma_2 = \frac{1}{(1 - e^{-9.6})} = 1$$

$$sum_3 = [0 \quad 32] \times \begin{bmatrix} 1.3 \\ -1 \end{bmatrix} = -32$$

$$tanh_1 = \frac{(e^{(-32)} - e^{-(-32)})}{(e^{(-32)} + e^{-(-32)})} = -1$$



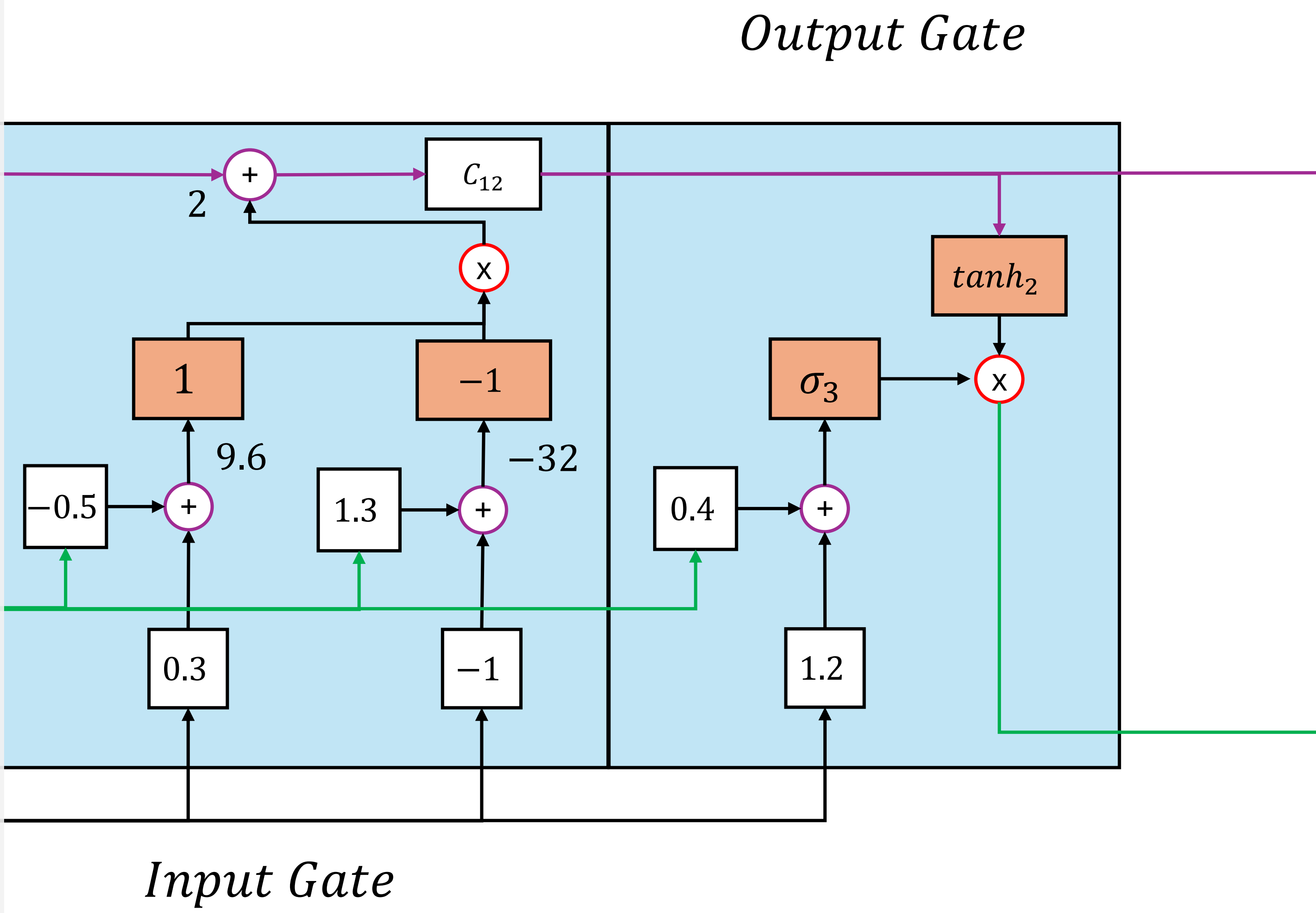
$$sum_2 = \begin{bmatrix} 0 & 32 \end{bmatrix} \times \begin{bmatrix} -0.5 \\ 0.3 \end{bmatrix} = 9.6$$

$$\sigma_2 = \frac{1}{(1 - e^{-9.6})} = 1$$

$$sum_3 = \begin{bmatrix} 0 & 32 \end{bmatrix} \times \begin{bmatrix} 1.3 \\ -1 \end{bmatrix} = -32$$

$$tanh_1 = \frac{(e^{(-32)} - e^{-(-32)})}{(e^{(-32)} + e^{-(-32)})} = -1$$

$$C_{12} = (1)(-1) + 2 = 1$$



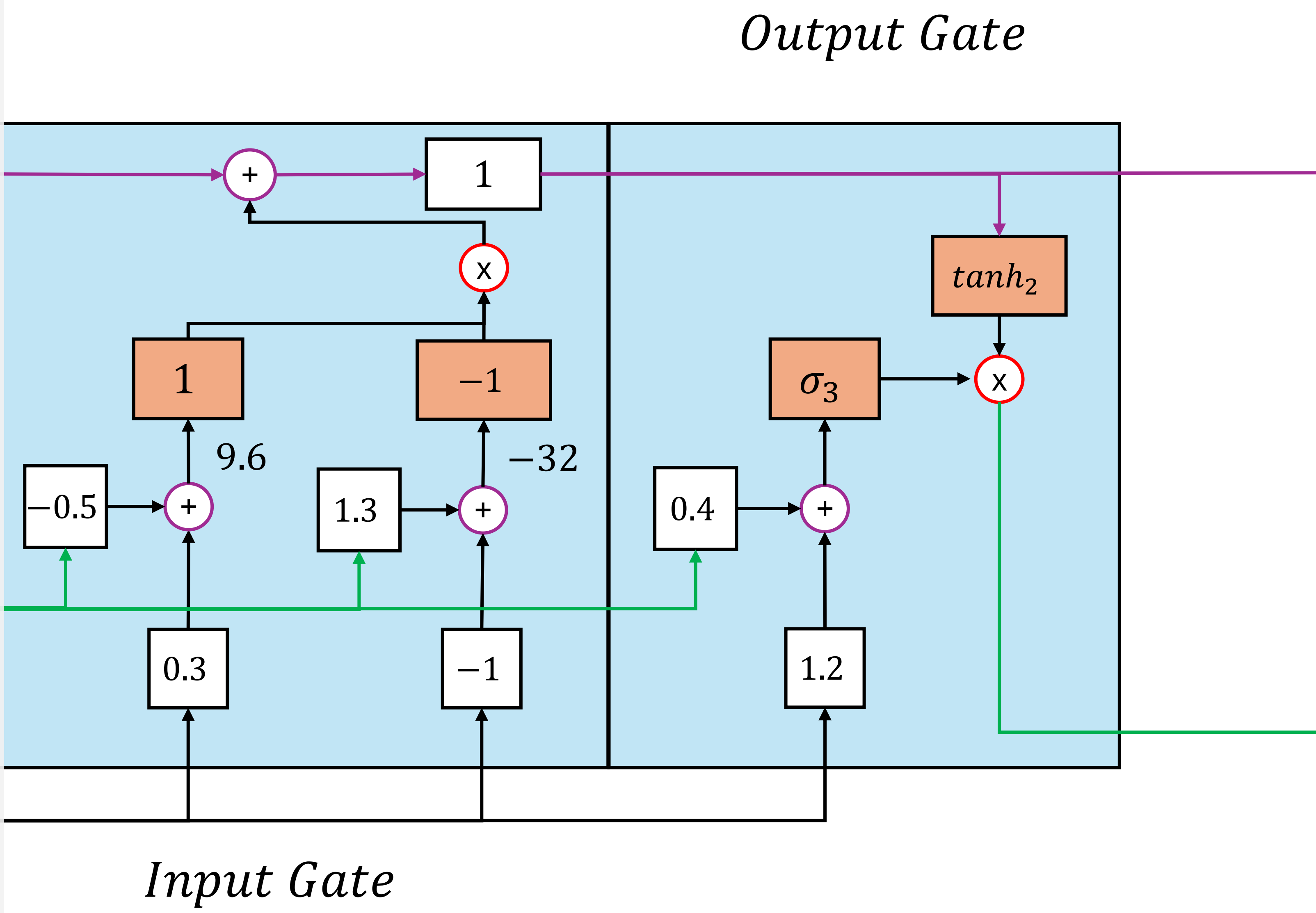
$$sum_2 = \begin{bmatrix} 0 & 32 \end{bmatrix} \times \begin{bmatrix} -0.5 \\ 0.3 \end{bmatrix} = 9.6$$

$$\sigma_2 = \frac{1}{(1 - e^{-9.6})} = 1$$

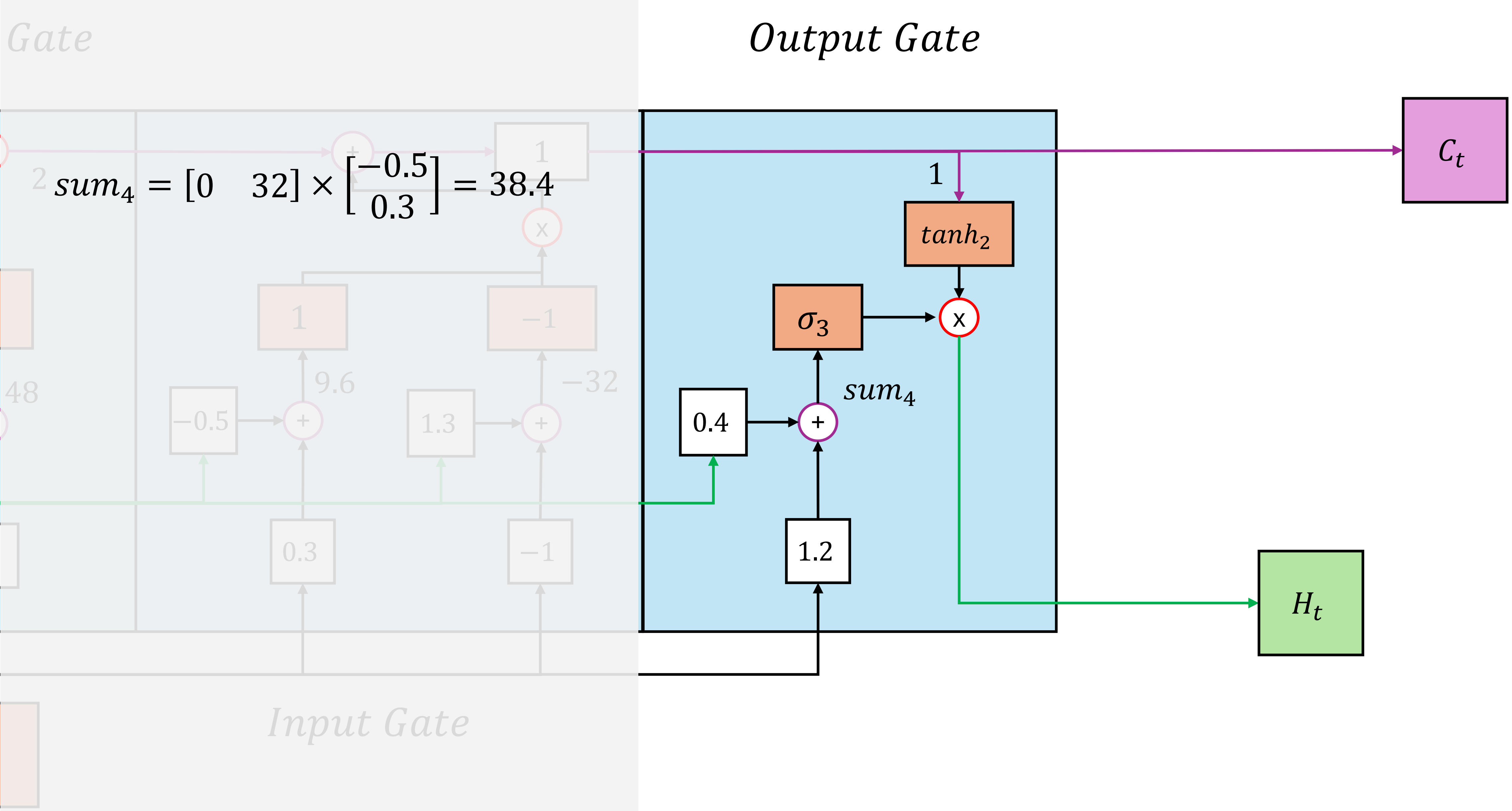
$$sum_3 = \begin{bmatrix} 0 & 32 \end{bmatrix} \times \begin{bmatrix} 1.3 \\ -1 \end{bmatrix} = -32$$

$$tanh_1 = \frac{(e^{(-32)} - e^{-(-32)})}{(e^{(-32)} + e^{-(-32)})} = -1$$

$$C_{12} = (1)(-1) + 2 = 1$$

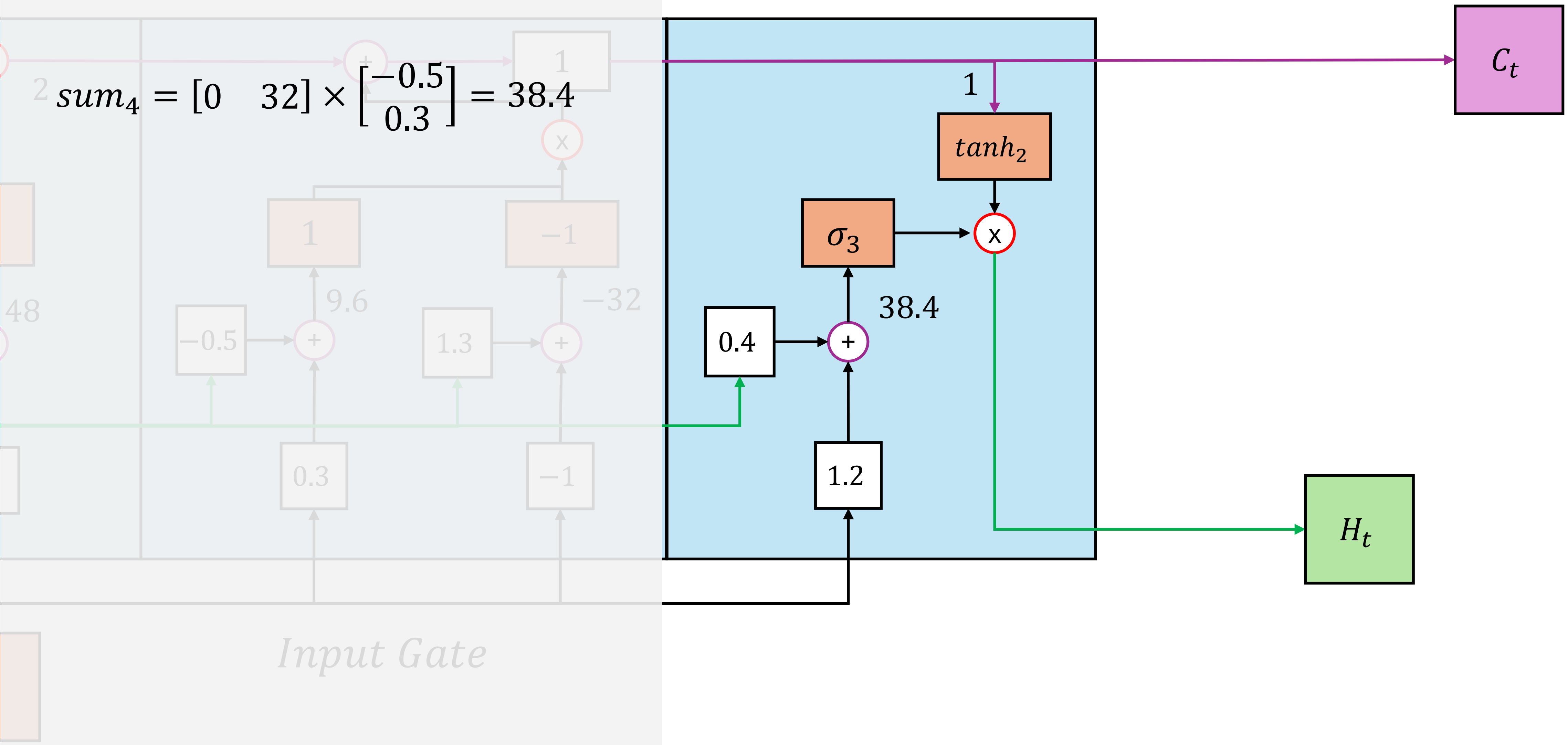


$$sum_4 = [0 \quad 32] \times \begin{bmatrix} -0.5 \\ 0.3 \end{bmatrix} = 38.4$$



Output Gate

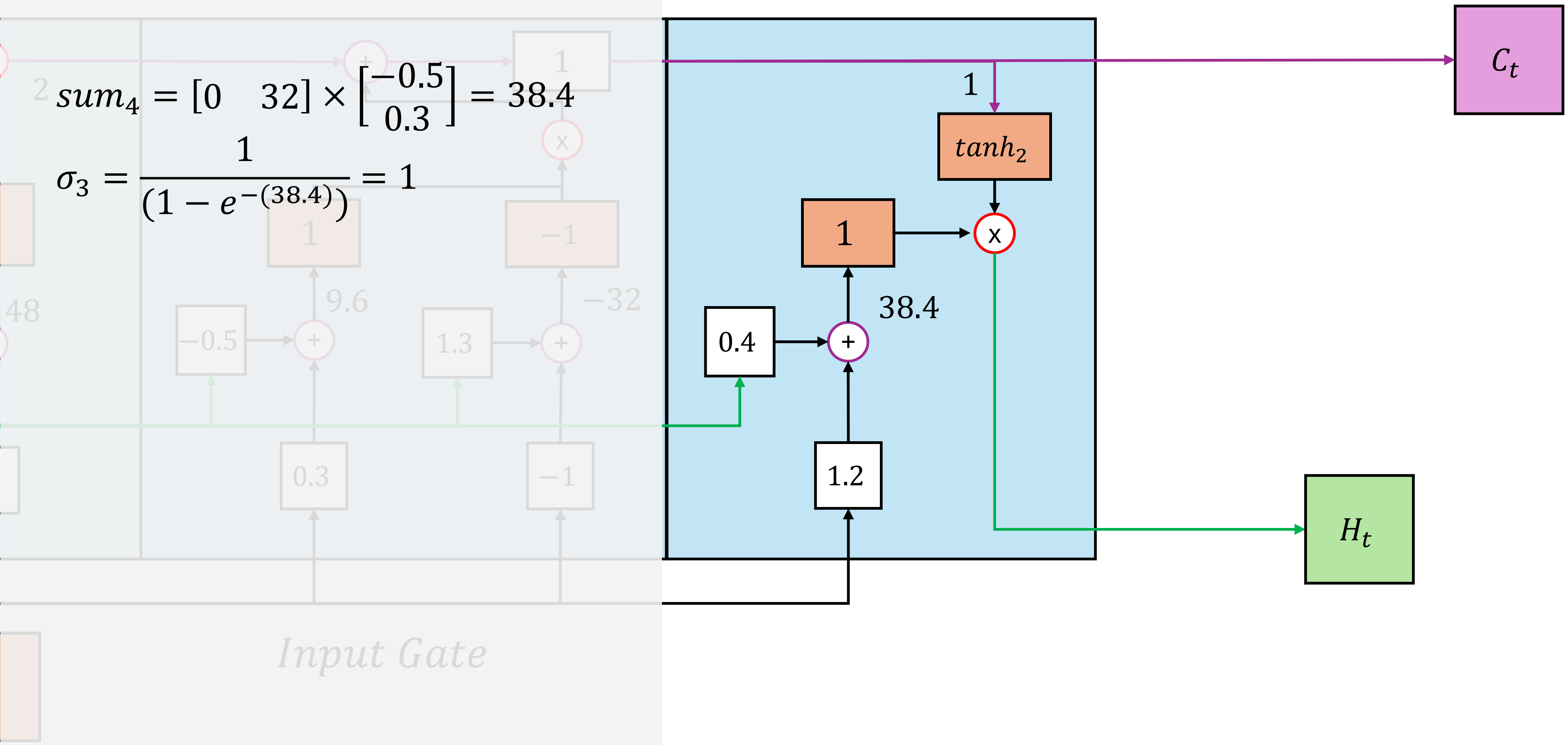
$$sum_4 = [0 \quad 32] \times \begin{bmatrix} -0.5 \\ 0.3 \end{bmatrix} = 38.4$$



Output Gate

$$sum_4 = [0 \quad 32] \times \begin{bmatrix} -0.5 \\ 0.3 \end{bmatrix} = 38.4$$

$$\sigma_3 = \frac{1}{(1 - e^{-(38.4)})} = 1$$

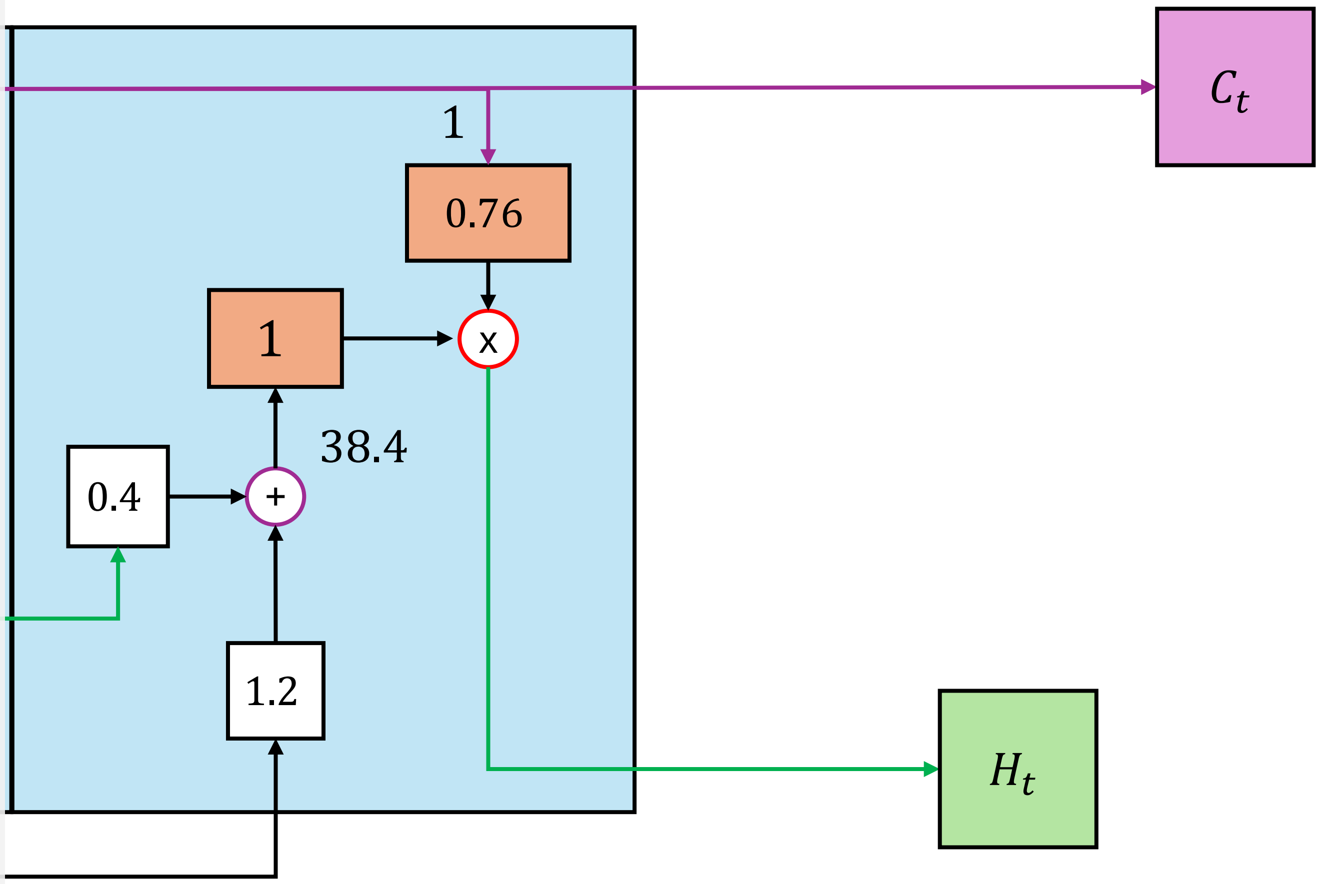


Output Gate

$$sum_4 = [0 \quad 32] \times \begin{bmatrix} -0.5 \\ 0.3 \end{bmatrix} = 38.4$$

$$\sigma_3 = \frac{1}{(1 - e^{-(38.4)})} = 1$$

$$tanh_2 = \frac{(e^{(1)} - e^{-(1)})}{(e^{(1)} + e^{-(1)})} = 0.76$$



Output Gate

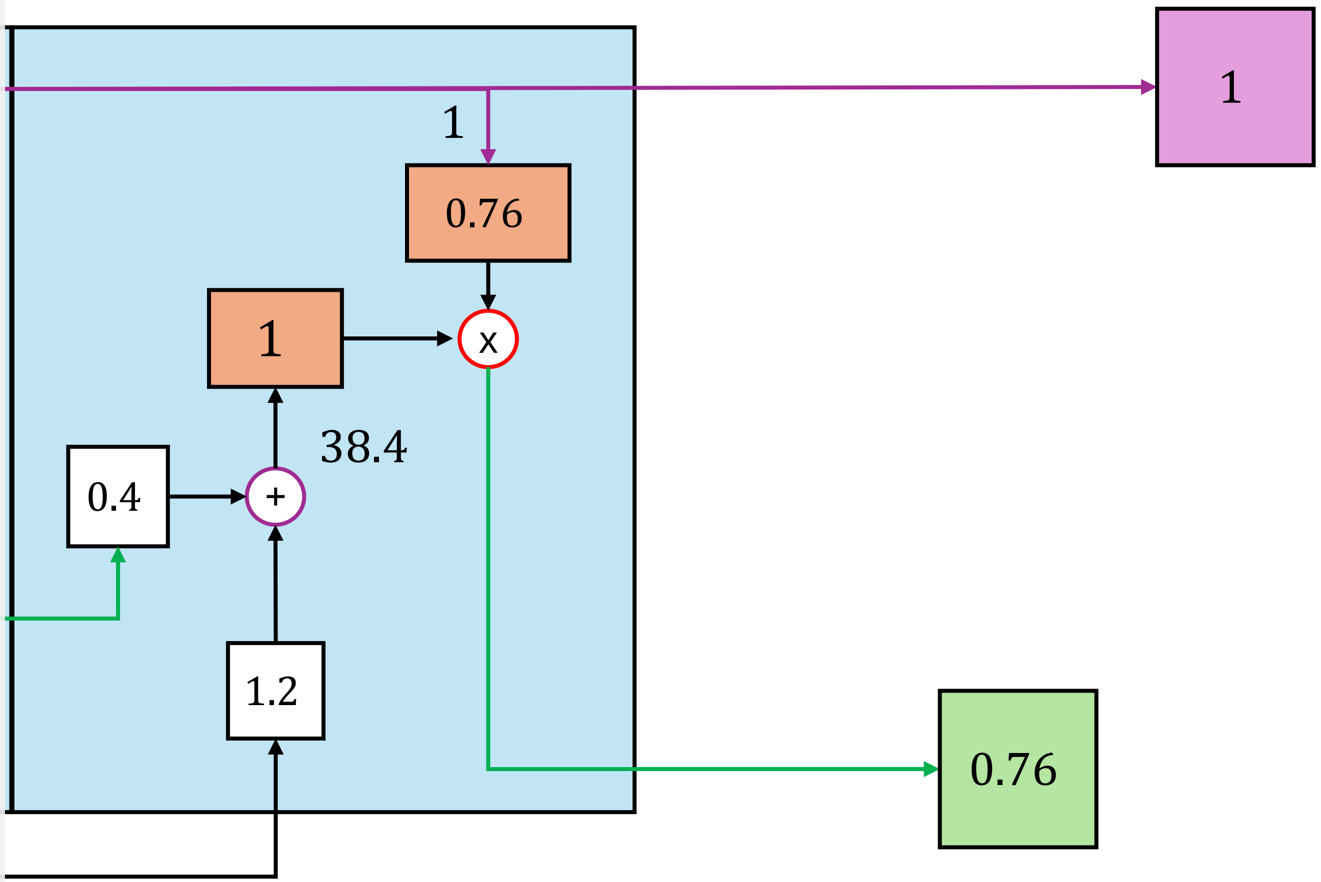
$$sum_4 = [0 \quad 32] \times \begin{bmatrix} -0.5 \\ 0.3 \end{bmatrix} = 38.4$$

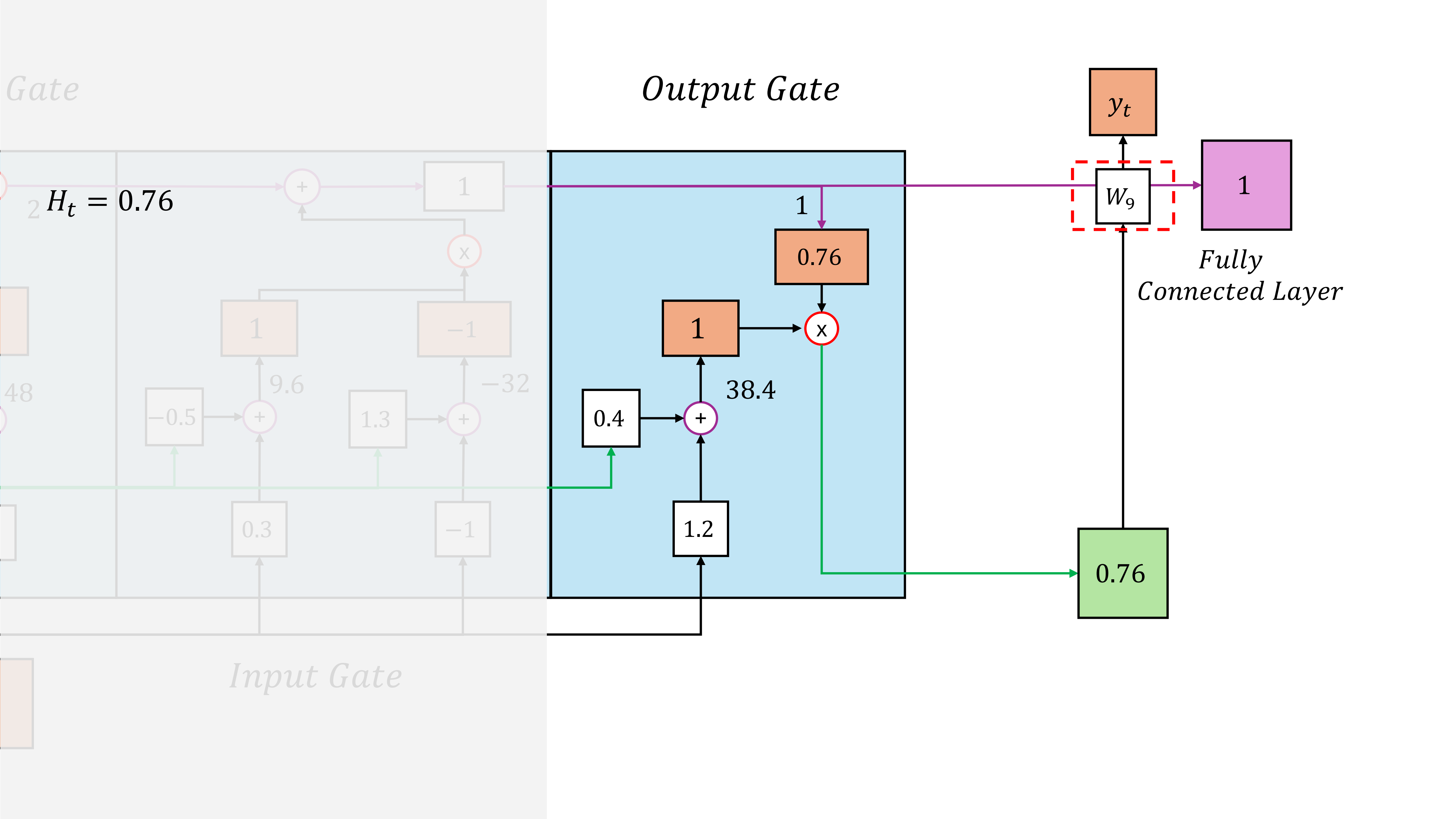
$$\sigma_3 = \frac{1}{(1 - e^{-(38.4)})} = 1$$

$$tanh_2 = \frac{(e^{(1)} - e^{-(1)})}{(e^{(1)} + e^{-(1)})} = 0.76$$

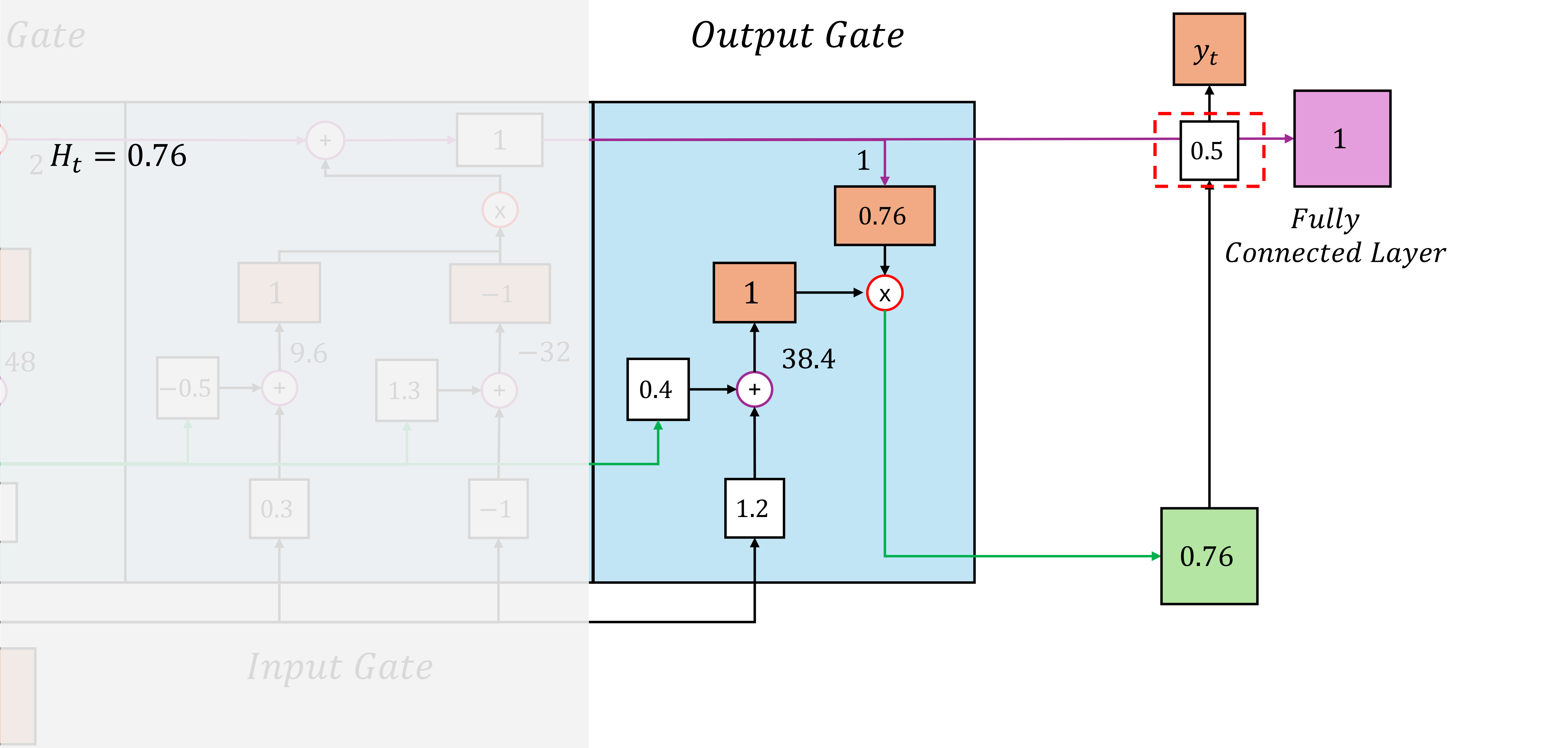
$$C_t = 1$$

$$H_t = (1)(0.76) = 0.76$$

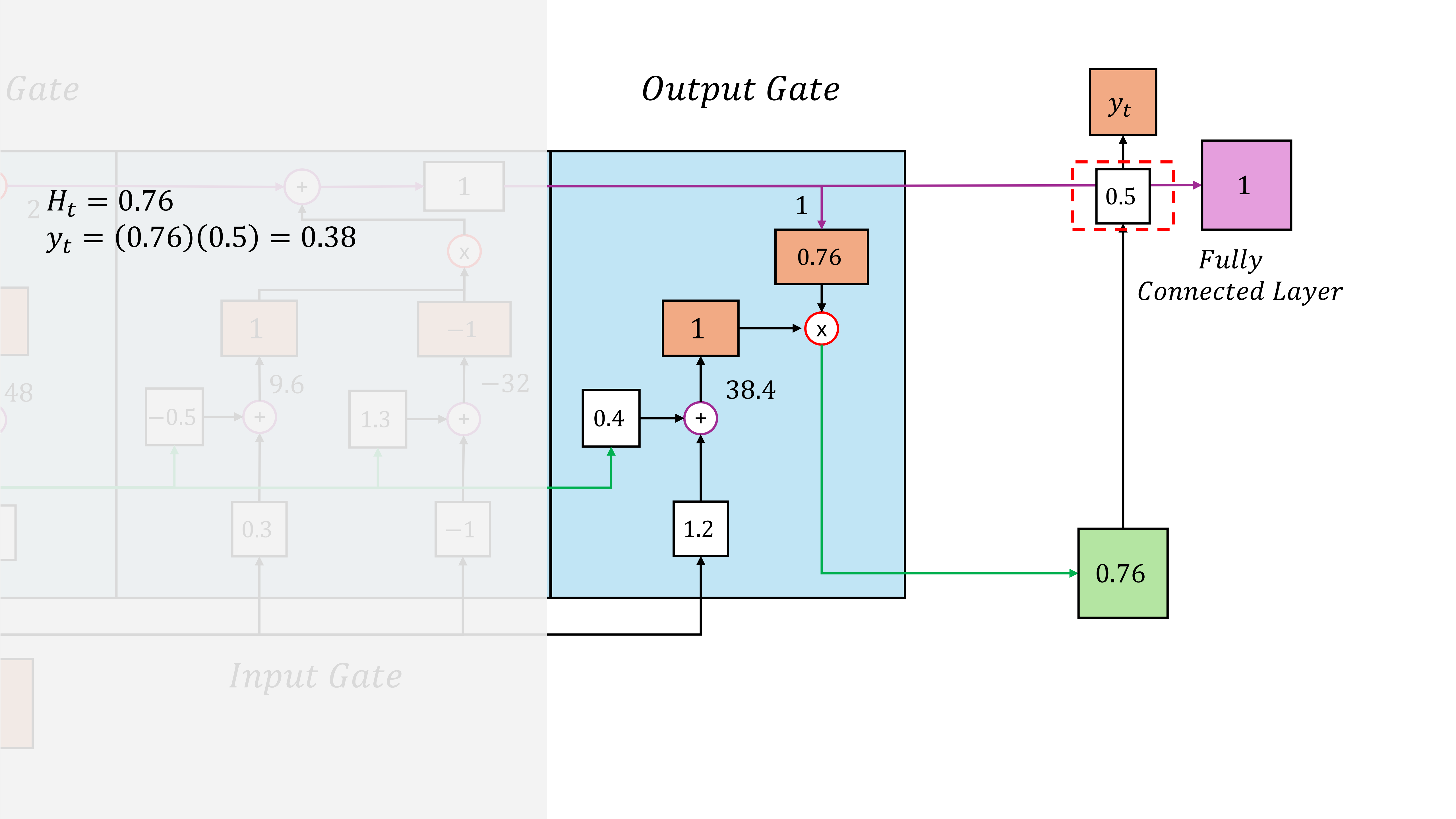




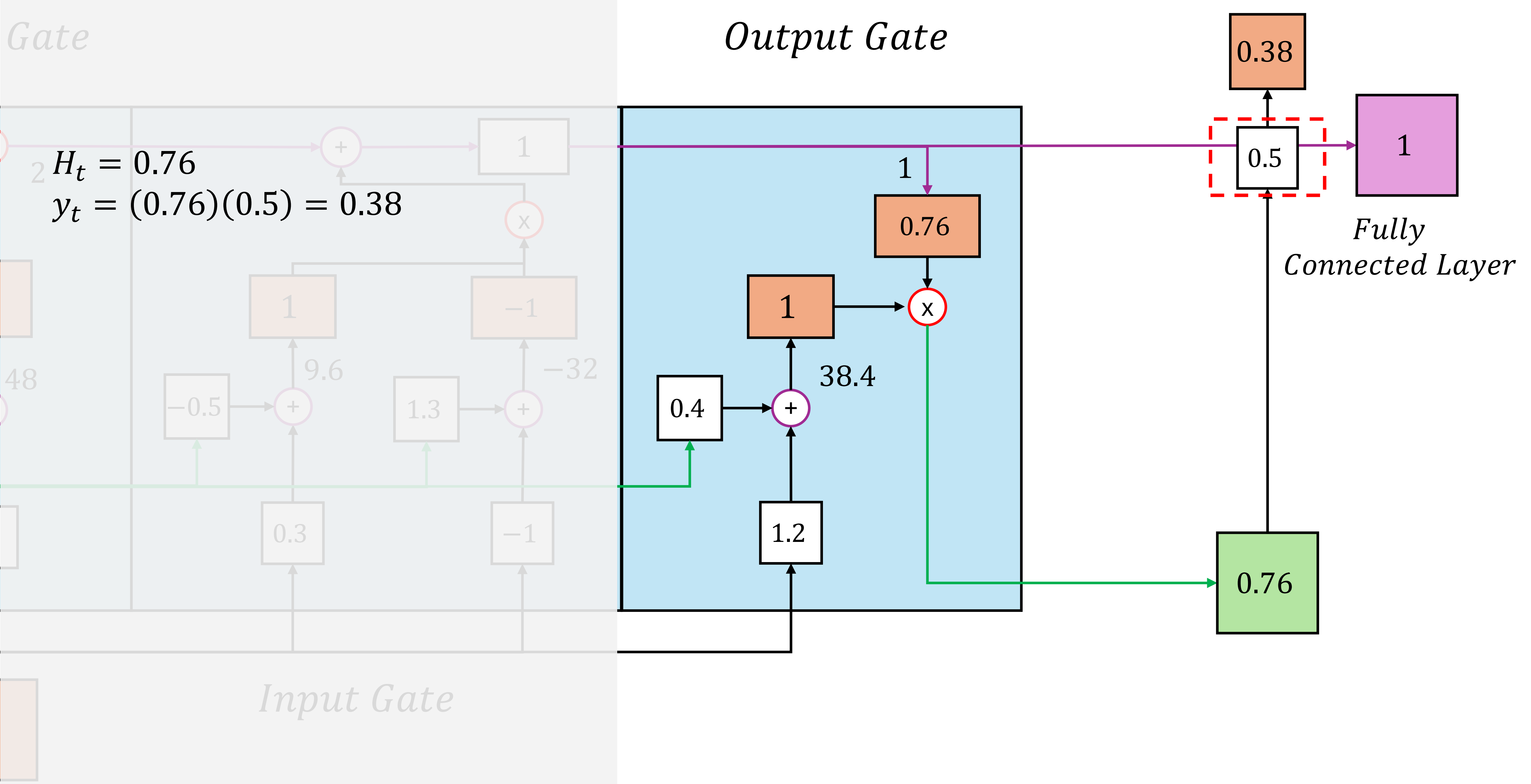
Gate
Output Gate
$H_t = 0.76$
2
1
1
0.76
1
−1
1
48
9.6
−32
38.4
−0.5
1.3
0.4
0.3
−1
1.2
Input Gate
y_t
W_9
1
Fully Connected Layer
0.76



Output Gate
y_t
$H_t = 0.76$
1
0.76
1
38.4
0.4
1.2
0.76
0.5
1
Fully Connected Layer
Input Gate



Gate
Output Gate
$H_t = 0.76$
$y_t = (0.76)(0.5) = 0.38$
1
0.76
1
38.4
0.4
1.2
y_t
0.5
1
Fully
Connected Layer
0.76
9.6
−32
48
−0.5
1.3
0.3
−1
−1
1
1
Input Gate



Gate
Output Gate
$H_t = 0.76$
$y_t = (0.76)(0.5) = 0.38$
0.38
1
0.5
1
0.76
1
x
38.4
0.4
+
1.2
0.76
Fully Connected Layer
Input Gate

Next step:

backpropagation

Time to code in

PyTorch

1. Import Library

```python
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import TensorDataset, DataLoader
import numpy as np
```

Import library

```python
device = ("cuda" if torch.cuda.is_available() else "cpu")
print(f"Using {device} device")
```

Declare to using **GPU,** in case GPU nor found it’s going to use CPU instead.

2. Define Train|Test Data

```
training_x = torch.tensor([
    [32.],
    [37.],
    [25.]
])

training_y = torch.tensor([
    [34.],
    [36.],
    [24.]
])
```

Training Data

Testing Data

```
testing_x = torch.tensor([
    [27.],
])

testing_y = torch.tensor([
    [28.]
])
```

2. Define Train|Test Data

```
dataset = TensorDataset(training_x, training_y)
train_loader = DataLoader(dataset, batch_size=1, shuffle=True)
```

Put training data into **DataLoader**

3. Define Model

```python
class LSTMModeler(nn.Module):
    def __init__(self, input_size, hidden_layer_size, output_size):
        super(LSTMModeler, self).__init__()

        self.lstm = nn.LSTM(input_size, hidden_layer_size)
        self.linear = nn.Linear(hidden_layer_size, output_size)
        self.hidden_cell = (torch.zeros(1, 1, hidden_layer_size
                            torch.zeros(1, 1, hidden_layer_size

    def forward(self, x):
        lstm_out, self.hidden_cell = self.lstm(x.view(len(x), 1, -1), self.hidden_cell)
        out = self.linear(lstm_out.view(len(x), -1))
        return out[-1]
```

w_n

y_t

x_t

C_t, H_t

```python
for x in training_x:
    print(x.view(len(x), 1, -1))
```

[13] ✓ 0.0s

```
tensor([[[32.]]])
tensor([[[37.]]])
tensor([[[25.]]])
```

4. Setup Loss Function and Optimizer

```
losses = []
hidden_layer_size = 10
input_size = 1
output_size = 1
model = LSTMModeler(input_size, hidden_layer_size, output_size).to(device)
loss_function = nn.MSELoss()
optimizer = optim.SGD(model.parameters(), lr=0.001)

print(model)
```

5. Training Step

```
epochs = 100

for epoch in range(epochs):
    total_loss = 0

    for (x, y) in train_loader:
        x, y = torch.tensor(x).to(device), torch.tensor(y).to(device)

        model.zero_grad()

        model.hidden_cell = (torch.zeros(1, 1, hidden_layer_size).to(device),
                             torch.zeros(1, 1, hidden_layer_size).to(device))

        y_hat = model(x)

        loss = loss_function(y, y_hat)

        loss.backward()
        optimizer.step()
        total_loss += loss.item()

    losses.append(total_loss)
```

Training data

Gradient reset

Comparing losses

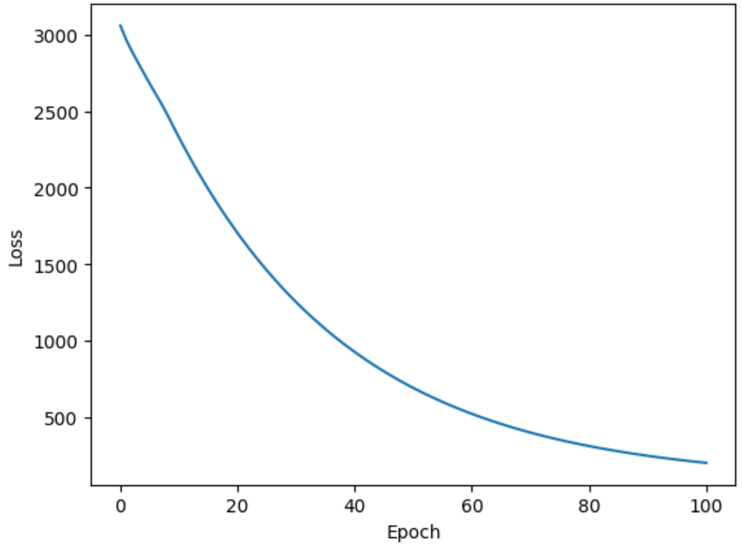
Backpropagation

6. Losses Plotting

```python
import matplotlib.pyplot as plt

def plot_losses(ax, t, losses):
    ax.plot(t, losses)
    ax.set_xlabel("Epoch")
    ax.set_ylabel("Loss")

fig, ax = plt.subplots()
plot_losses(ax, np.linspace(0., len(losses), len(losses)), losses)
```



7. Result Inspection

```python
for x, y in zip(testing_x, testing_y):
    # Get predicted vector
    with torch.no_grad():
        model.hidden_cell = (torch.zeros(1, 1, hidden_layer_size).to(device),
                             torch.zeros(1, 1, hidden_layer_size).to(device))

        x = x.view(-1, 1, 1).to(device)  # (sequence_length, batch_size, input_size)

        pred = model(x)

    print(f"y_true: {int(y.item())}, y_hat: {int(pred.item())}")
```

y_true: 28, y_hat: 25