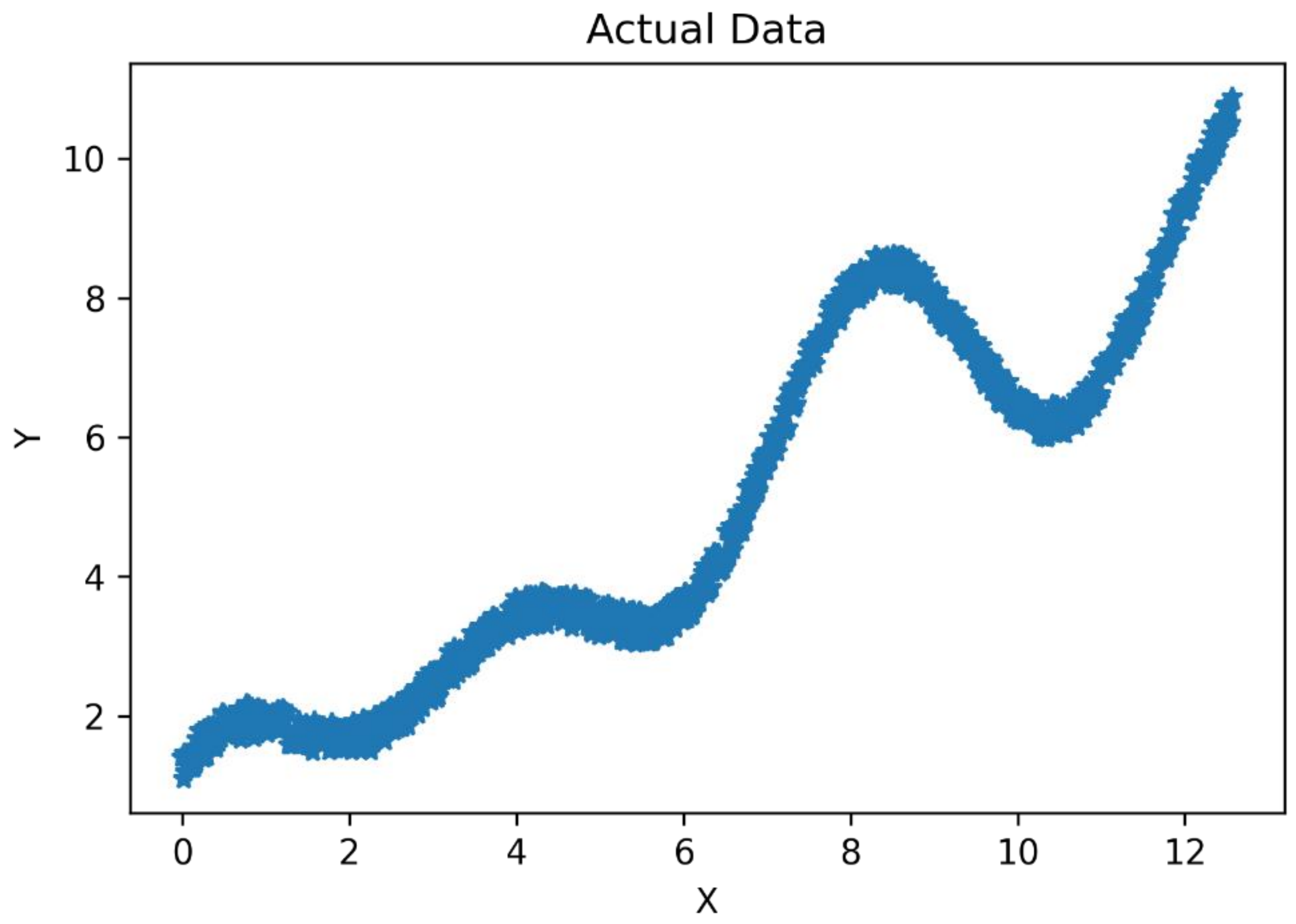
Artificial Intelligence Techniques *in* Manufacturing

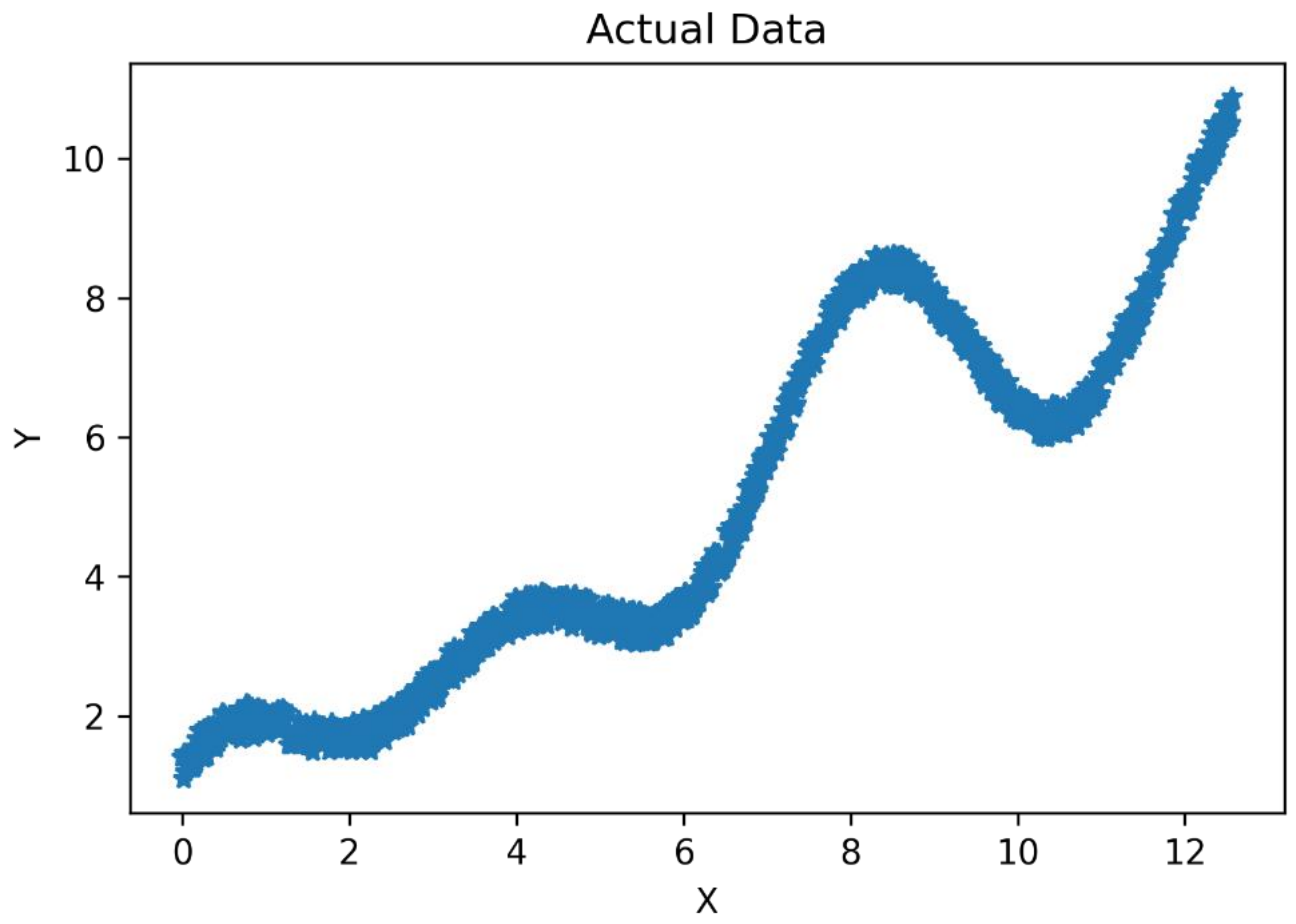
Machine Learning: State of the Art

Data	Type	Examples
Inhomogeneous Data	Gradient Boosting Machine	• XGBoost • CatBoost • LightGBM
Homogeneous Data	Artificial Neural Network	• Deep Neural Network • Convolutional Neural Network • Long Short-Term Memory Neural Network • Transformer Network

Artificial Neural Network

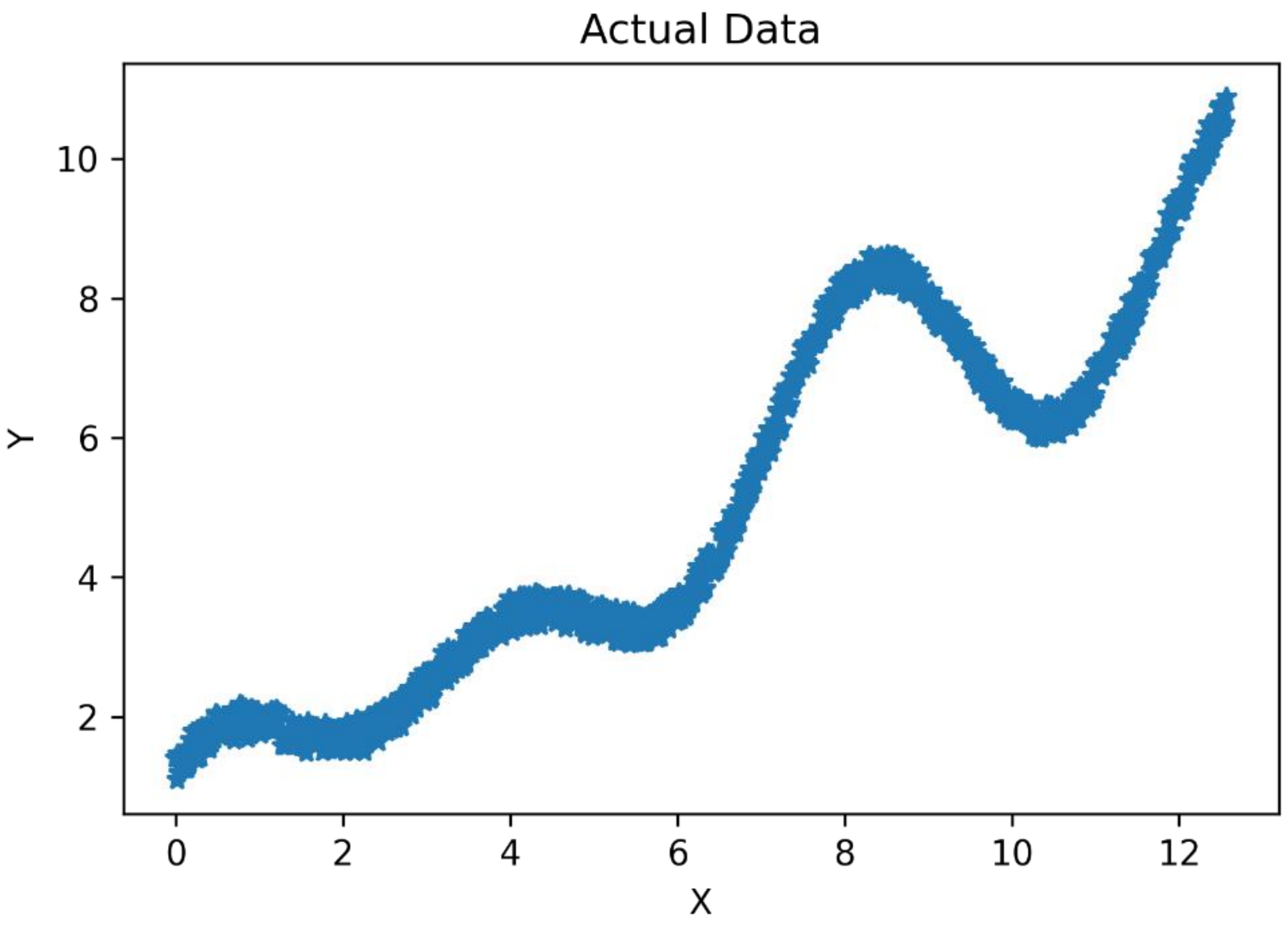


How do we fit a line to this data?



$$y = f(x)$$

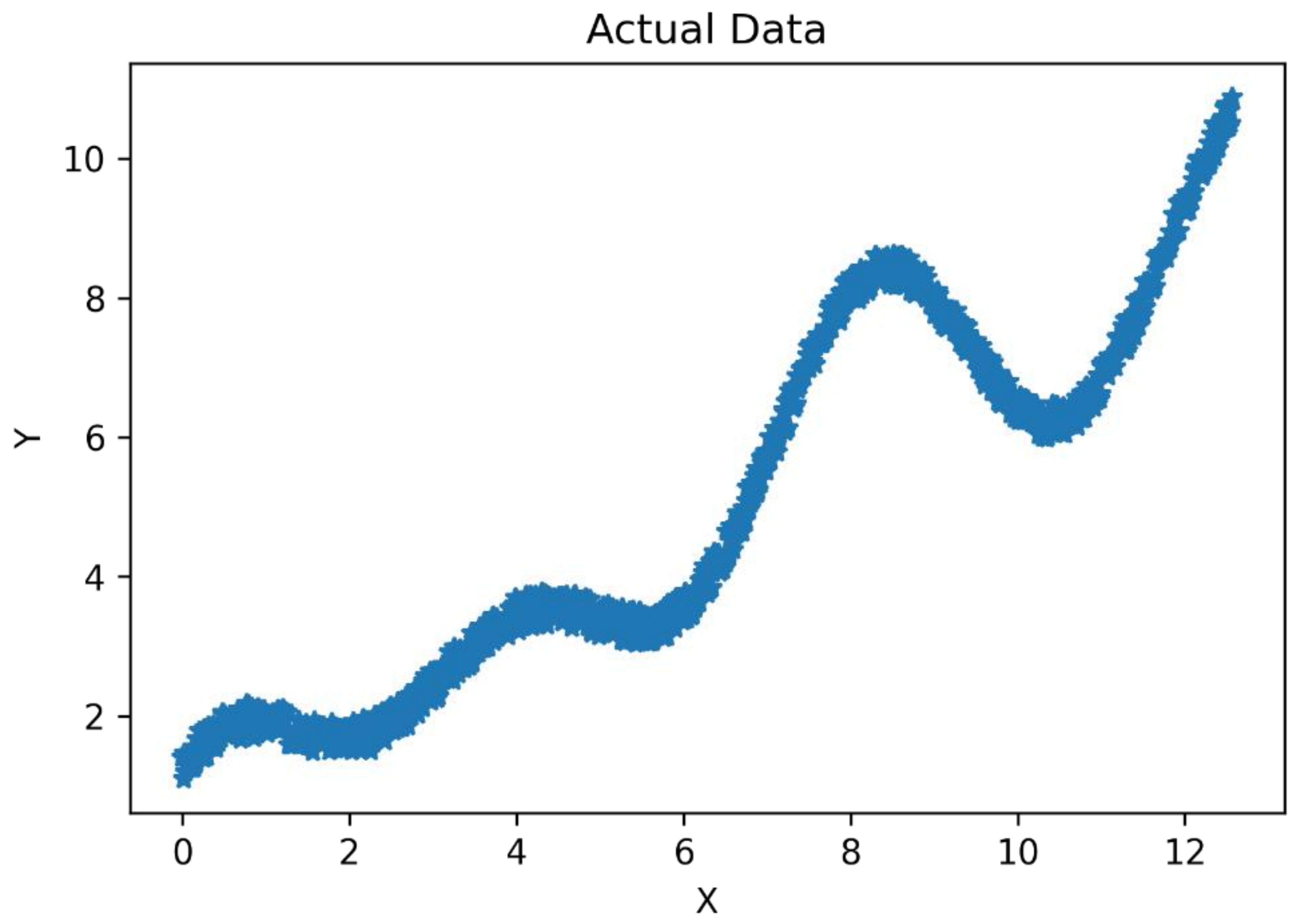
$$y = a_0 + a_1 x + a_2 x^2 + \cdots$$



$$y = f(x)$$

$$y = a_0 + a_1 x + a_2 x^2 + \cdots$$

$$y = a_0 + a_1 \sin(x) + a_2 \sin(2x) + \cdots$$



$$y = f(x)$$

$y = a_0 + a_1 x + a_2 x^2 + \cdots$

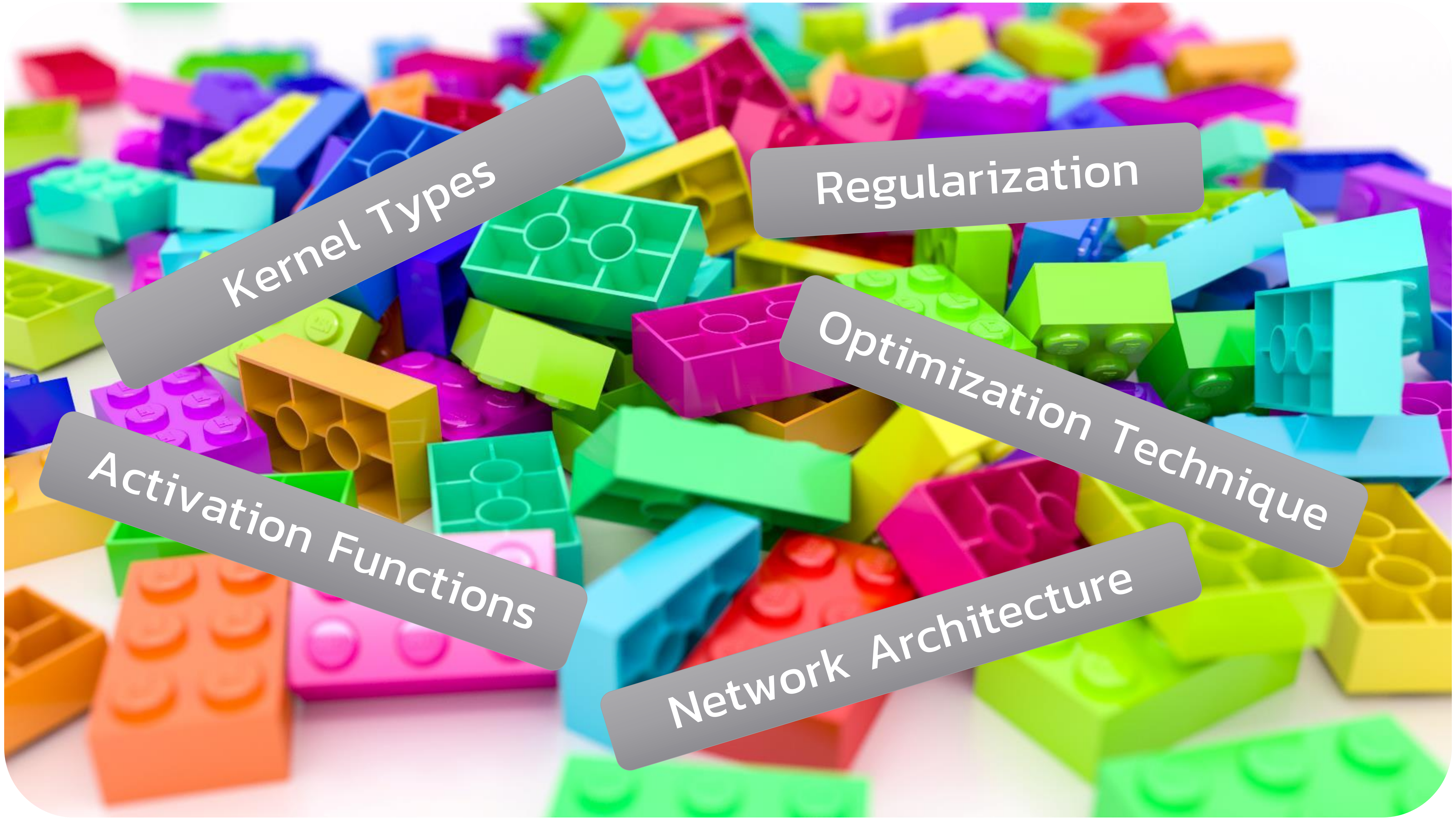
$y = a_0 + a_1 \sin(x) + a_2 \sin(2x) + \cdots$

Artificial Neural Network

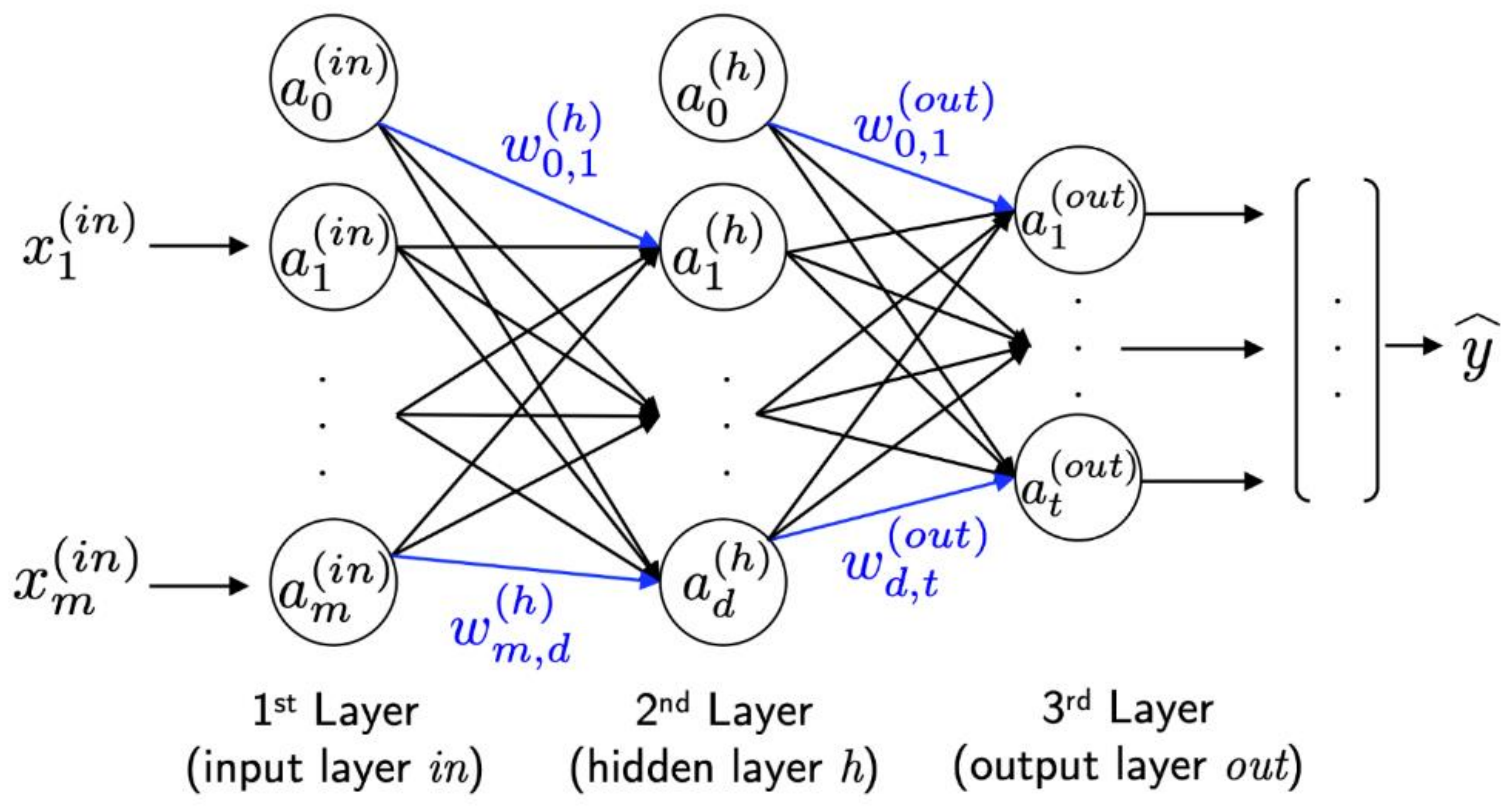
Universal Approximator

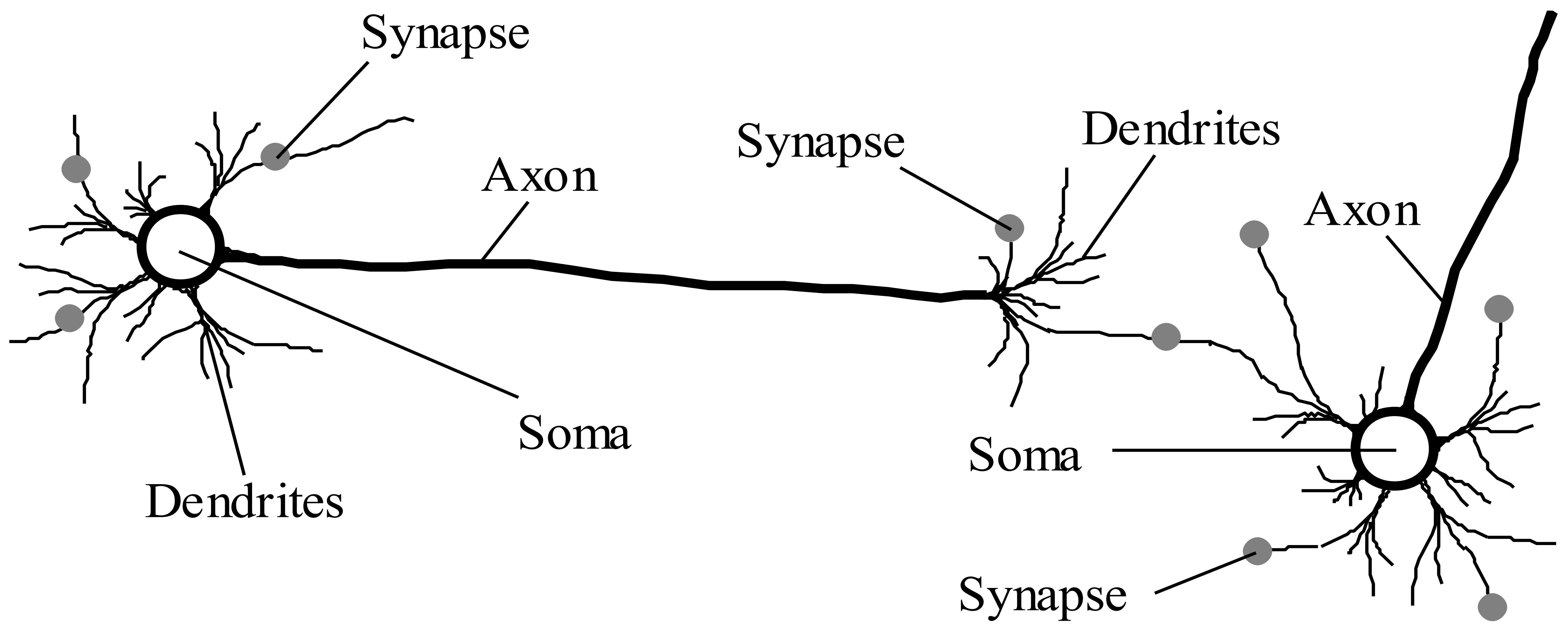
Artificial Neural Network

Artificial Neural Network



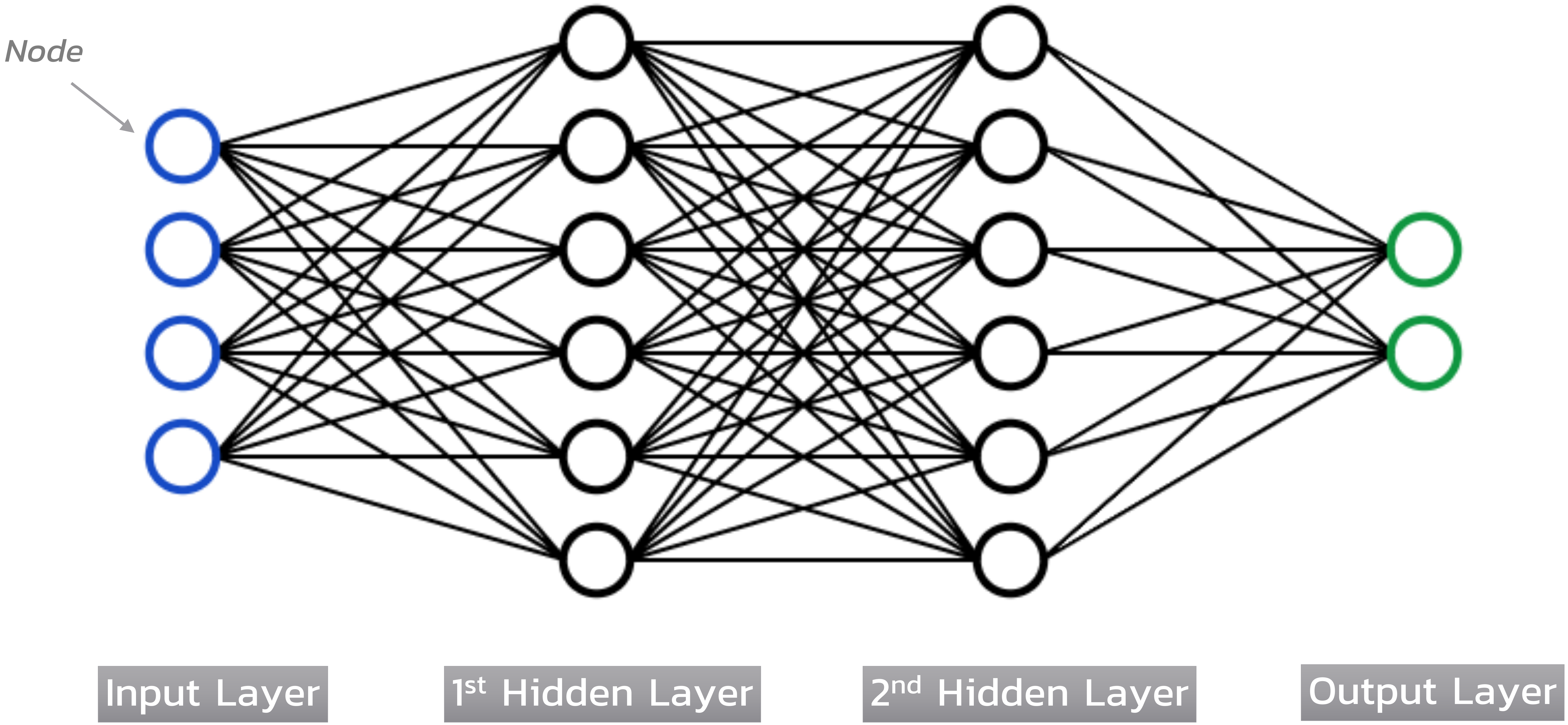
Connection to Biological Neural Networks



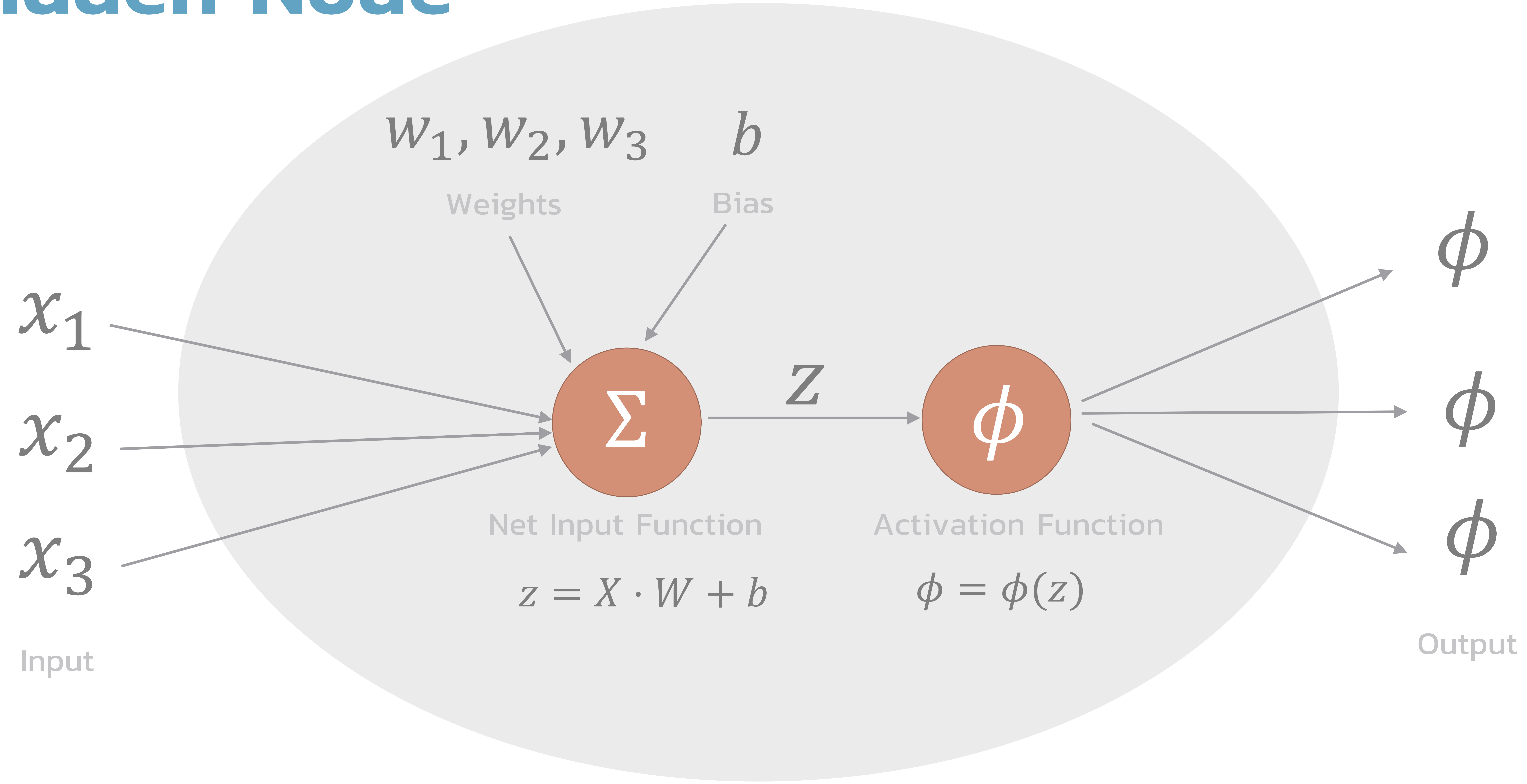


This connection is not relevant nowadays.

Architecture

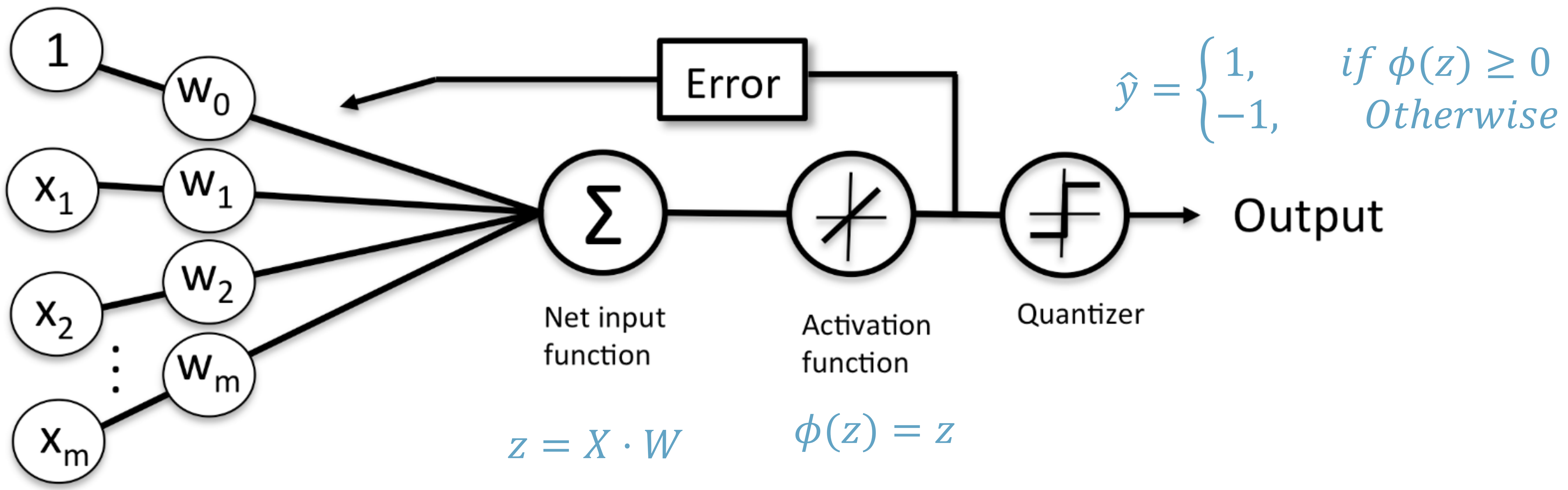


Hidden Node

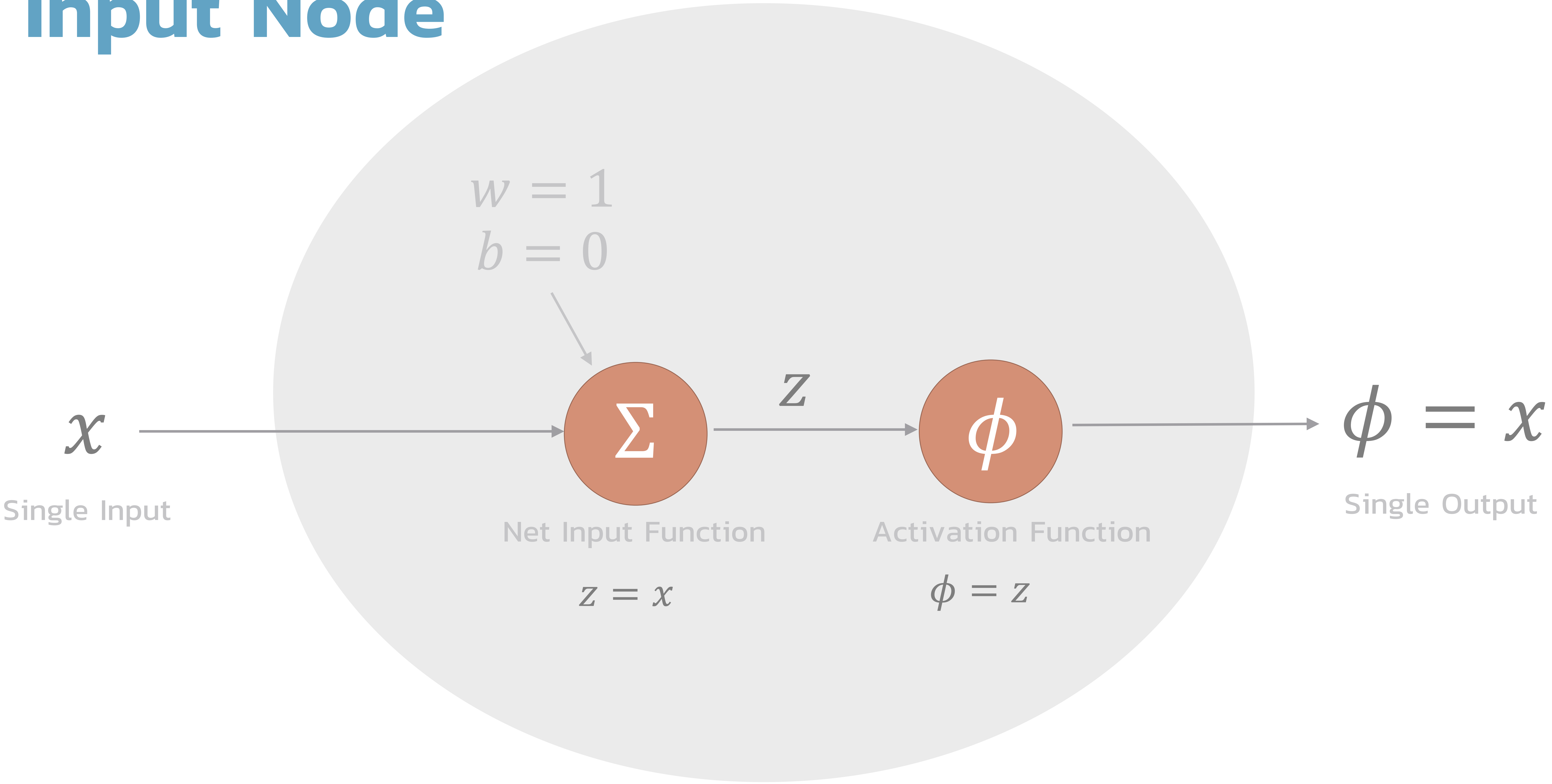


w_1, w_2, w_3
b
Weights
Bias
x_1
x_2
x_3
Σ
z
ϕ
Net Input Function
Activation Function
$z = X \cdot W + b$
$\phi = \phi(z)$
ϕ
ϕ
ϕ
Input
Output

Compared with Perceptron

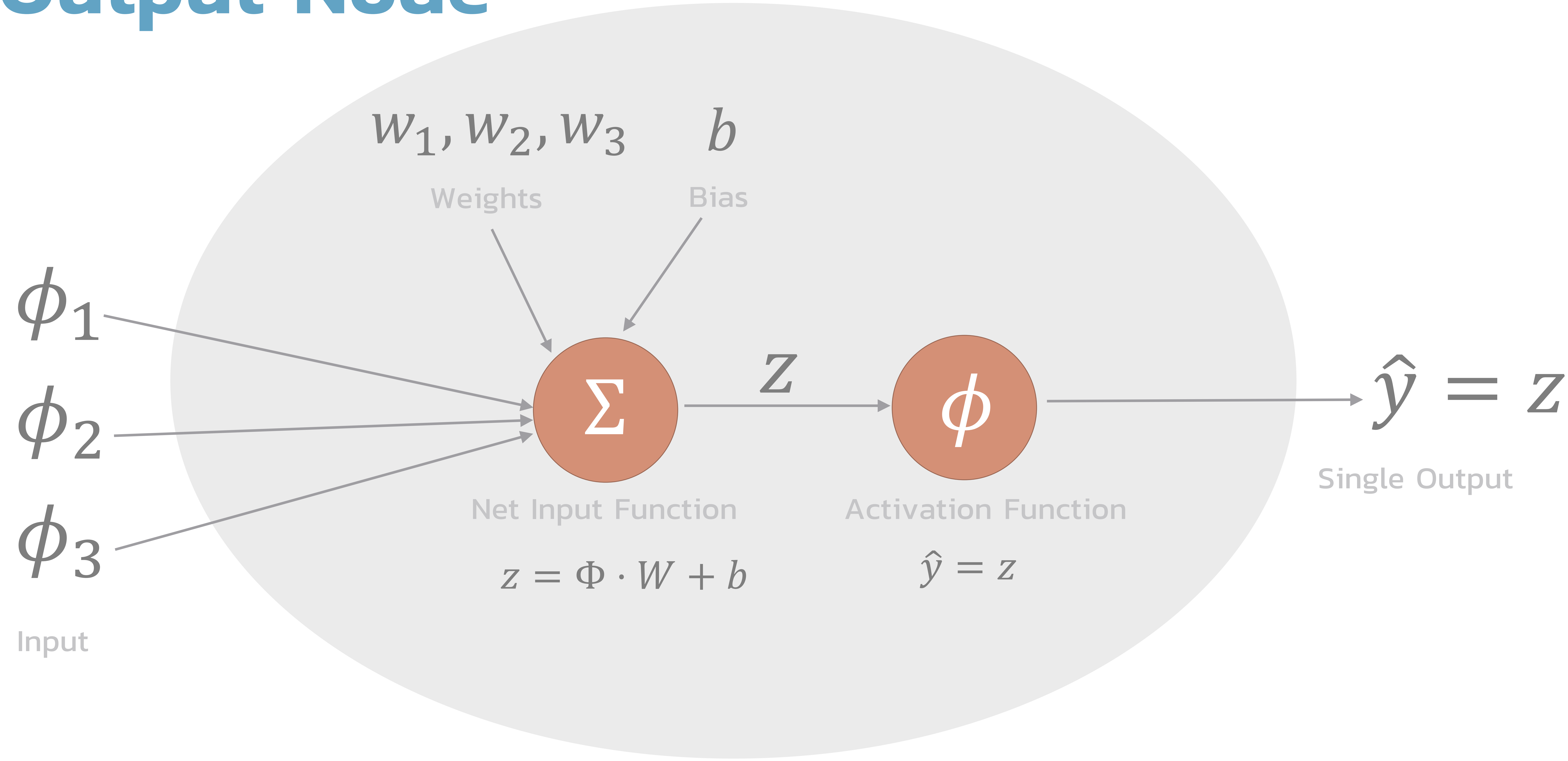


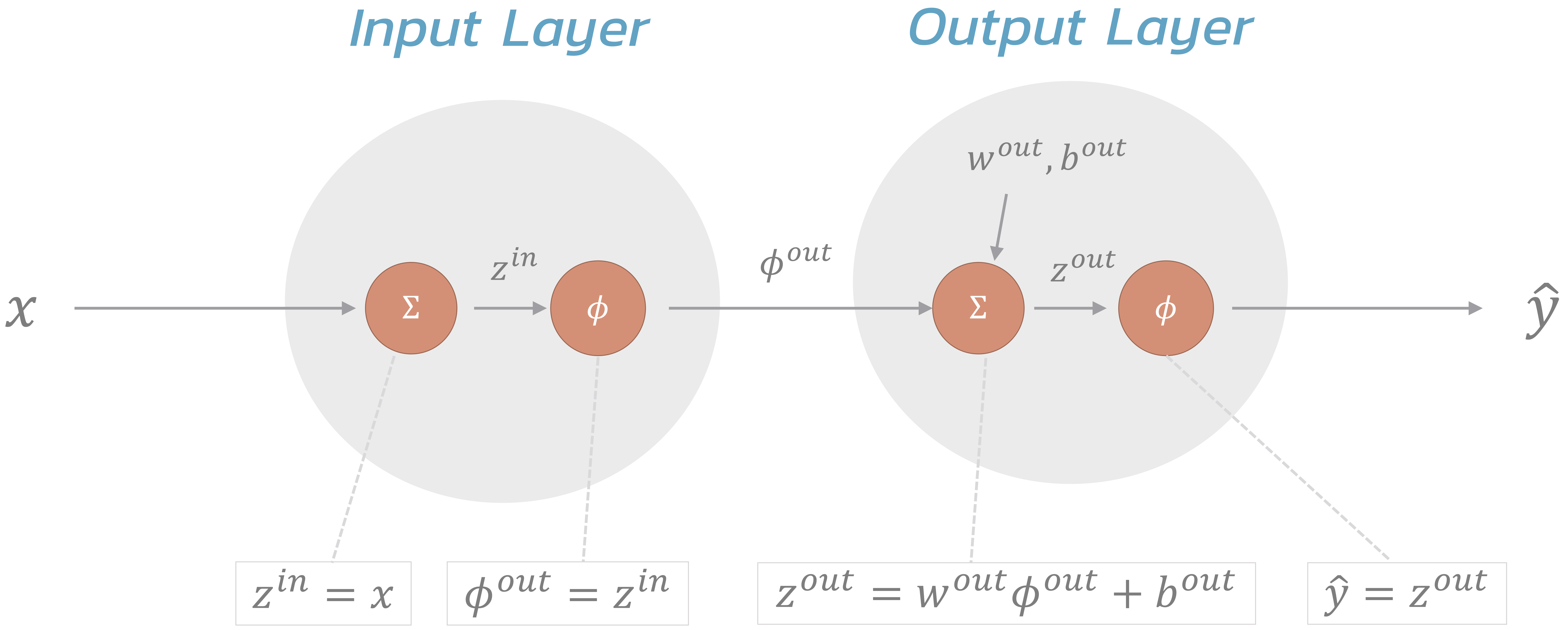
Input Node



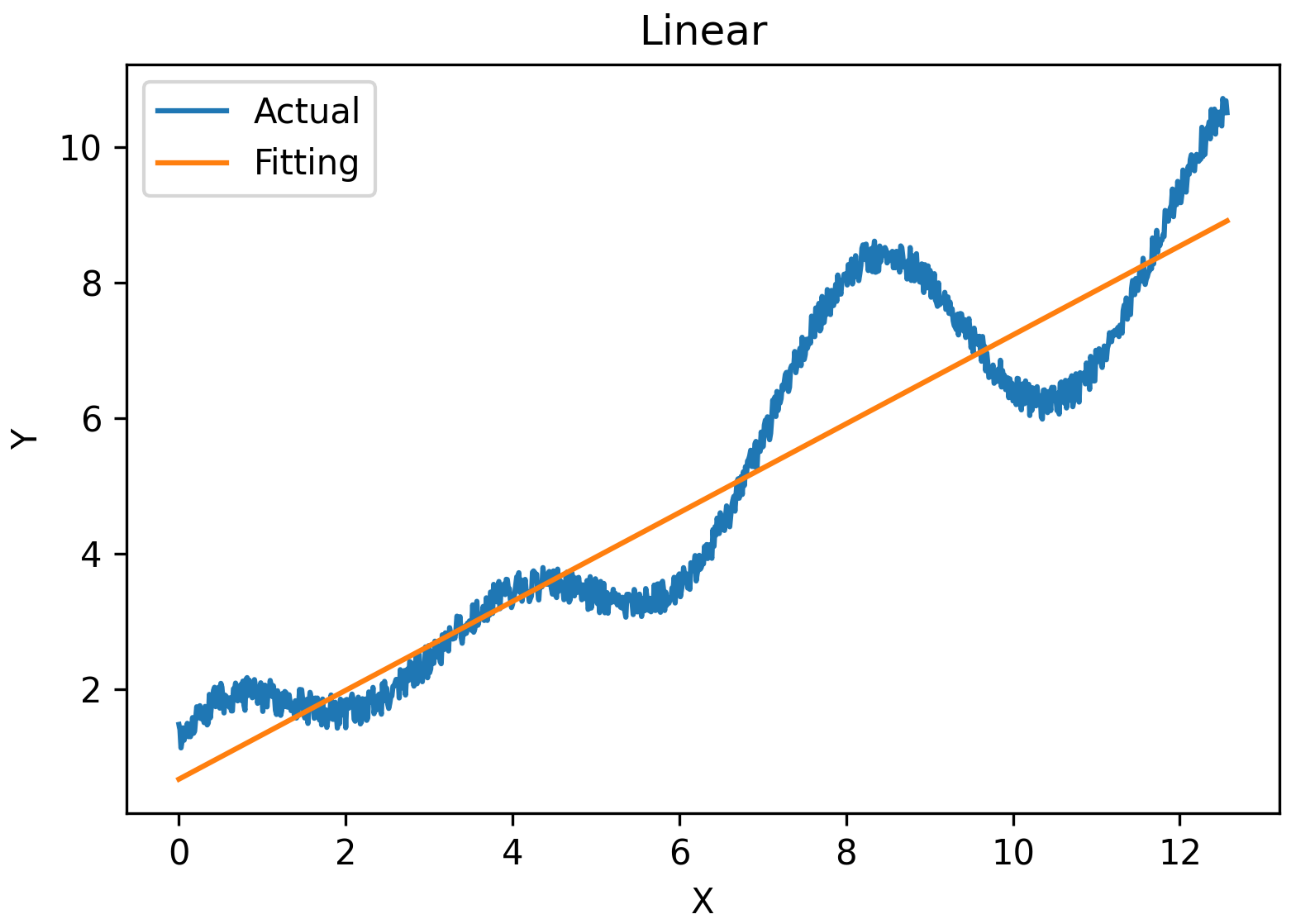
$w = 1$
$b = 0$
x
Single Input
Σ
Net Input Function
$z = x$
z
ϕ
Activation Function
$\phi = z$
$\phi = x$
Single Output

Output Node

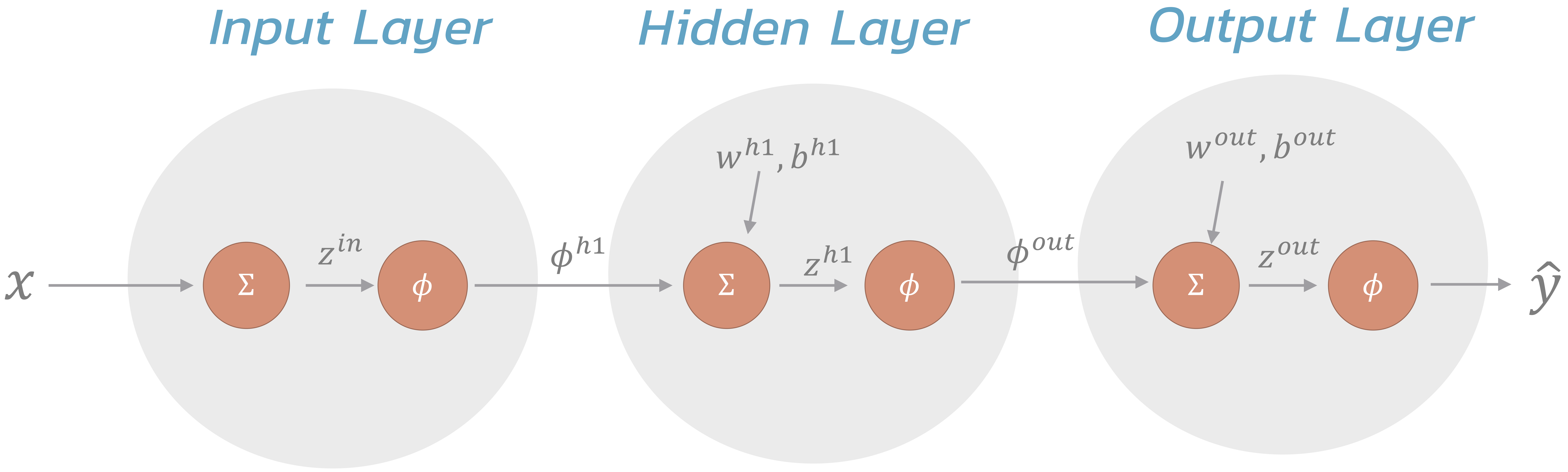

w_1, w_2, w_3
b
Weights
Bias
ϕ_1
ϕ_2
ϕ_3
Σ
z
ϕ
$\hat{y} = z$
Single Output
Net Input Function
Activation Function
$z = \Phi \cdot W + b$
$\hat{y} = z$
Input



$$\hat{y} = w^{out}x + b^{out}$$



#Parameters: 2



$$z^{in} = x$$

$$\phi^{h1} = z^{in}$$

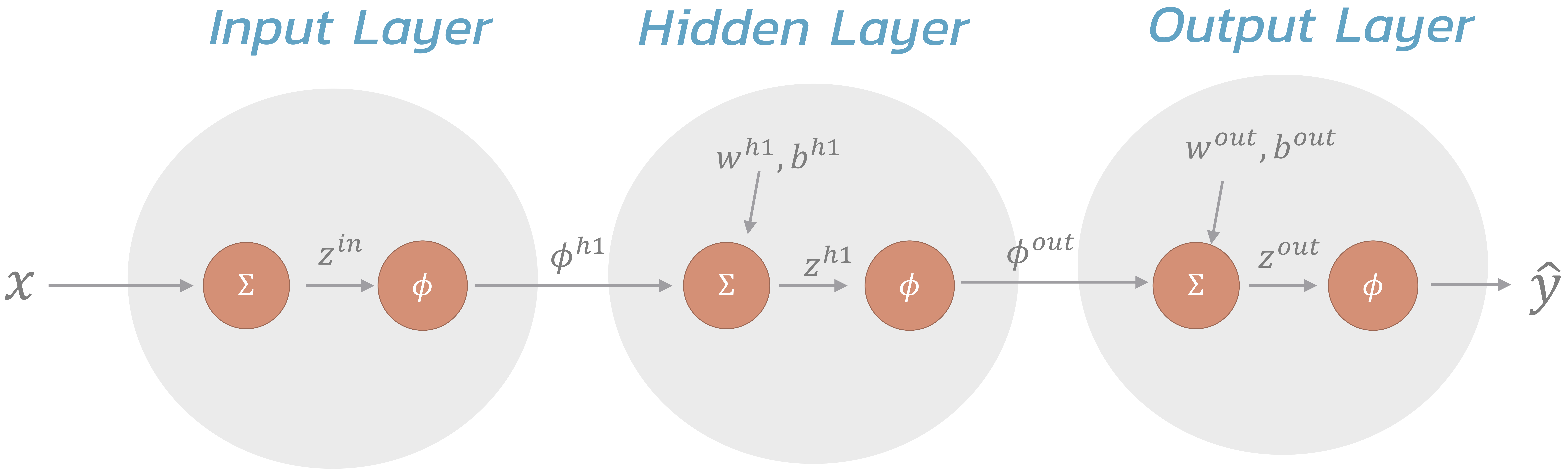
$$z^{h1} = w^{h1}\phi^{h1} + b^{h1}$$

$$\phi^{out} = \frac{1}{1 + e^{-z^{h1}}}$$

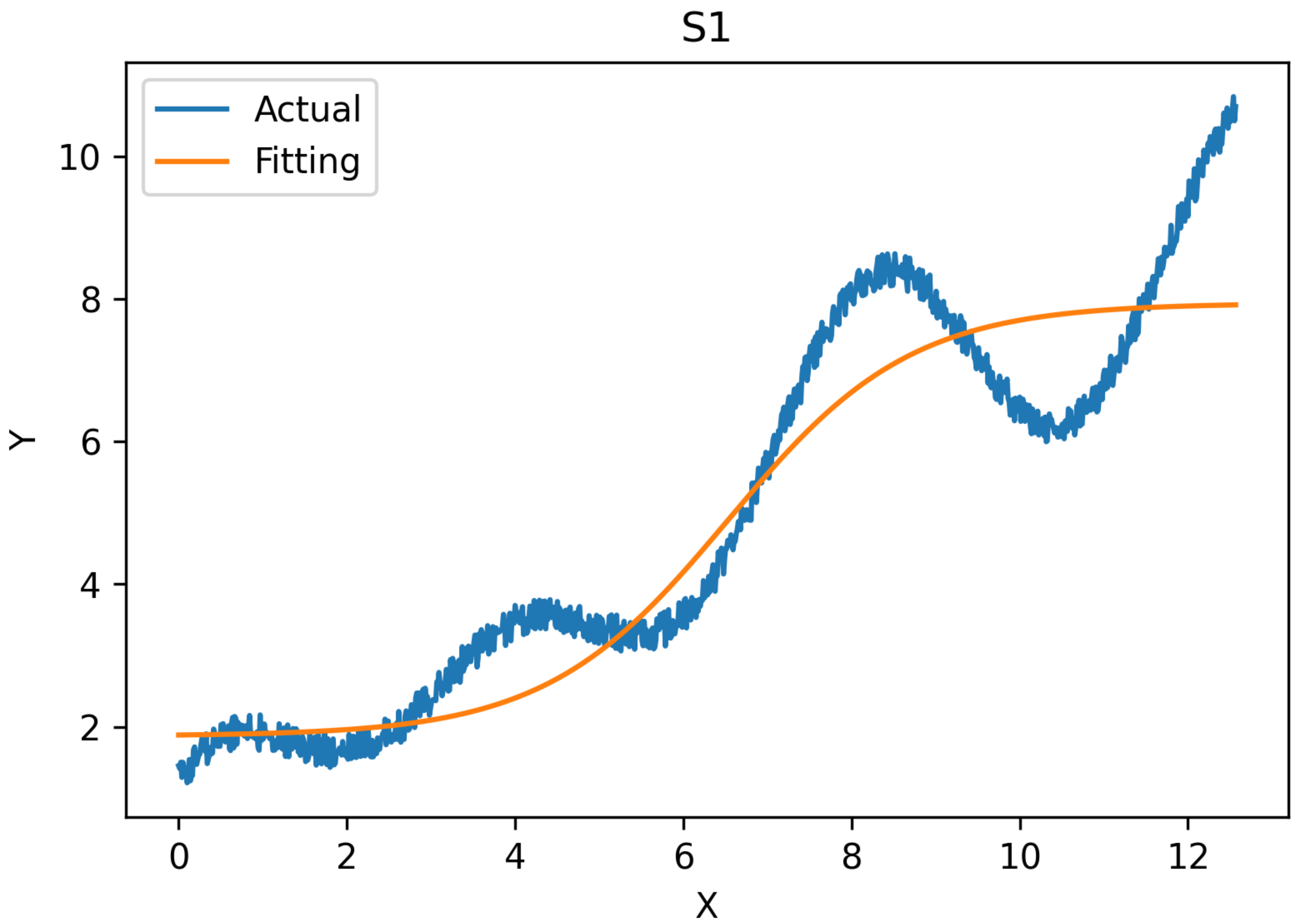
Sigmoid Function

$$z^{out} = w^{out}\phi^{out} + b^{out}$$

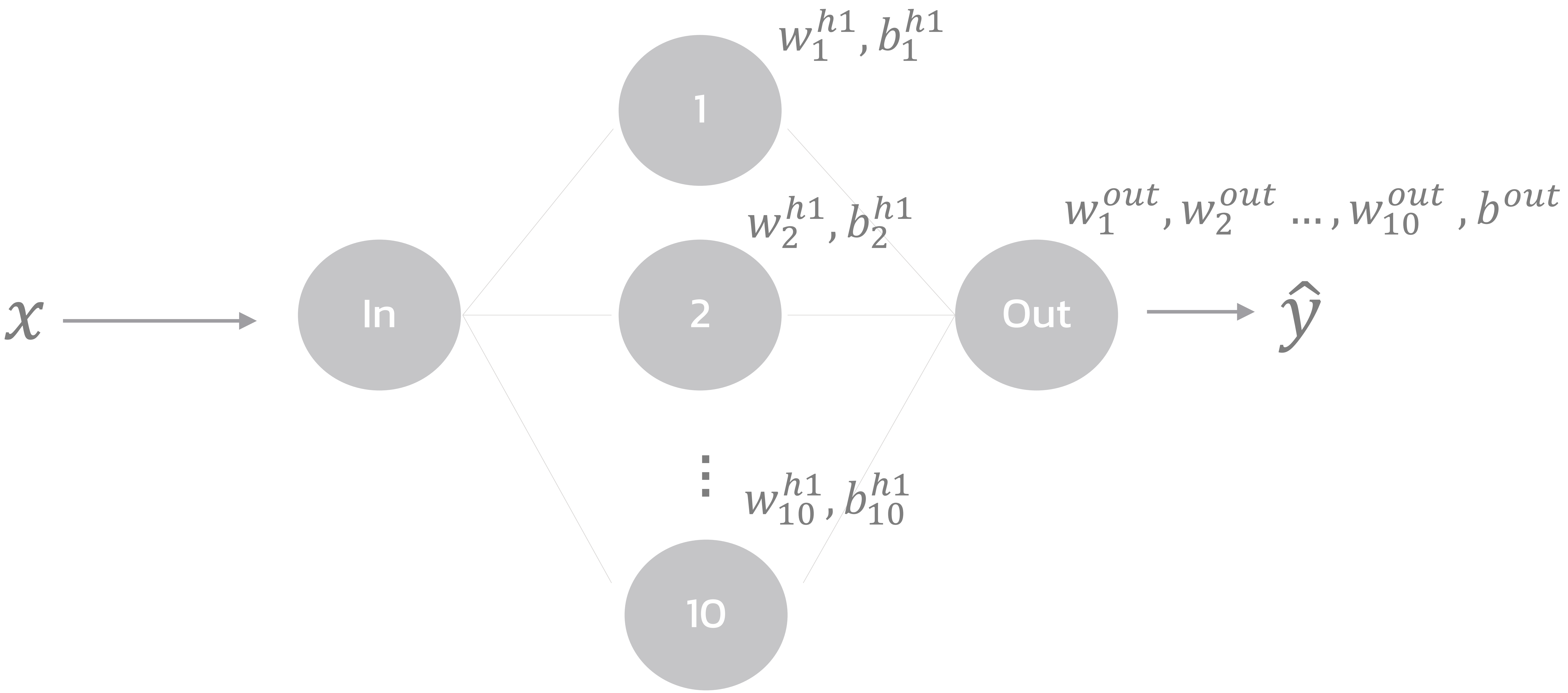
$$\hat{y} = z^{out}$$



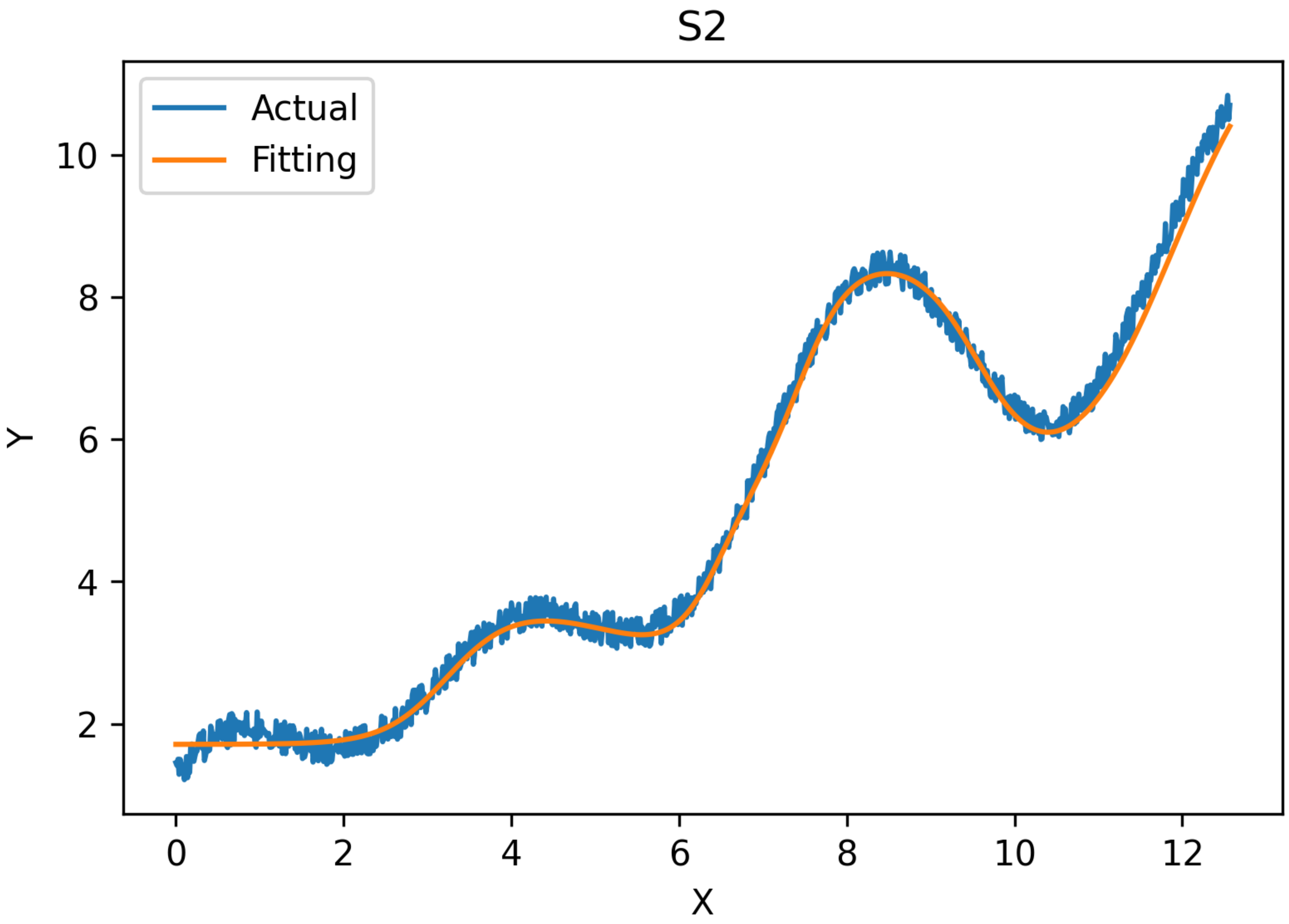
$$\hat{y} = w^{out}\left[\frac{1}{1+e^{-(w^{h1}x+b^{h1})}}\right] + b^{out}$$



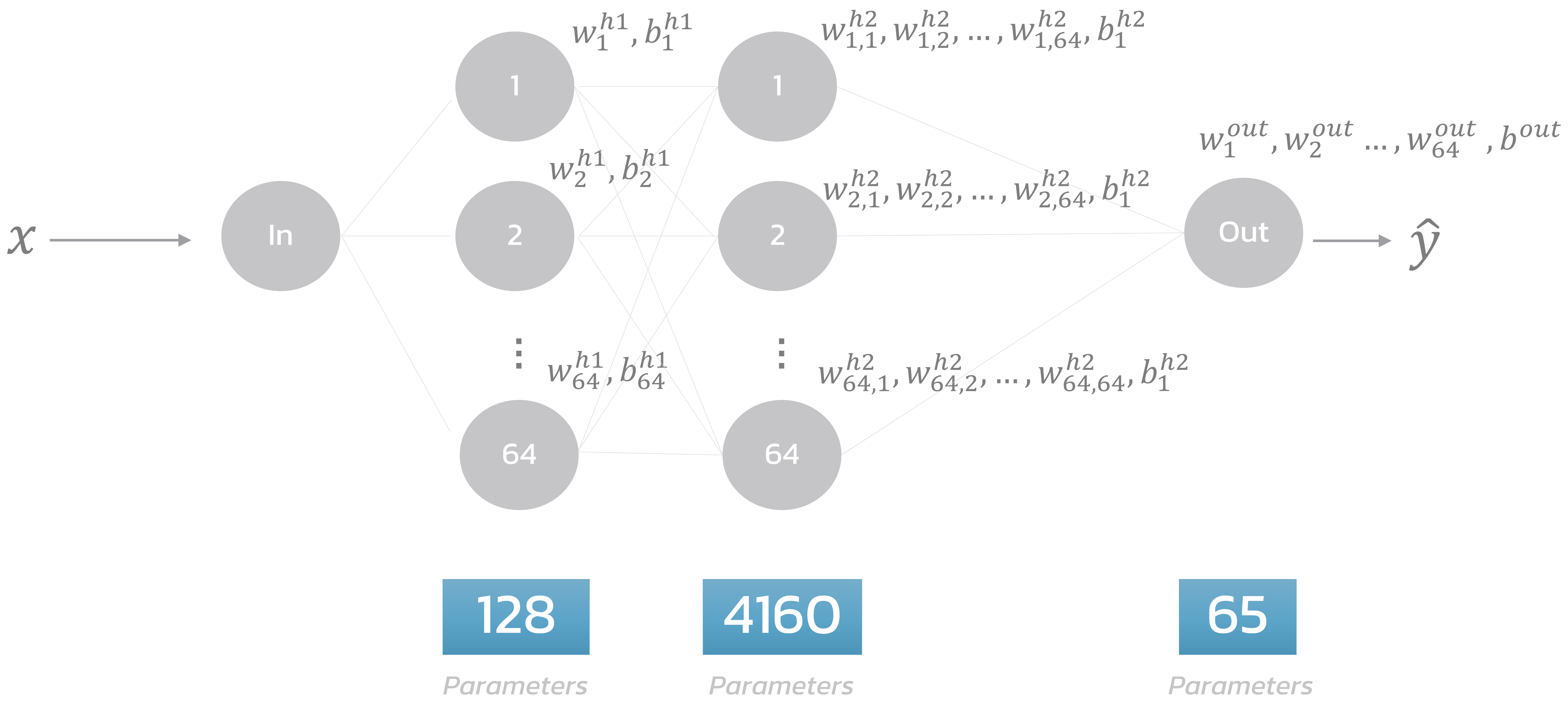
#Parameters: 4



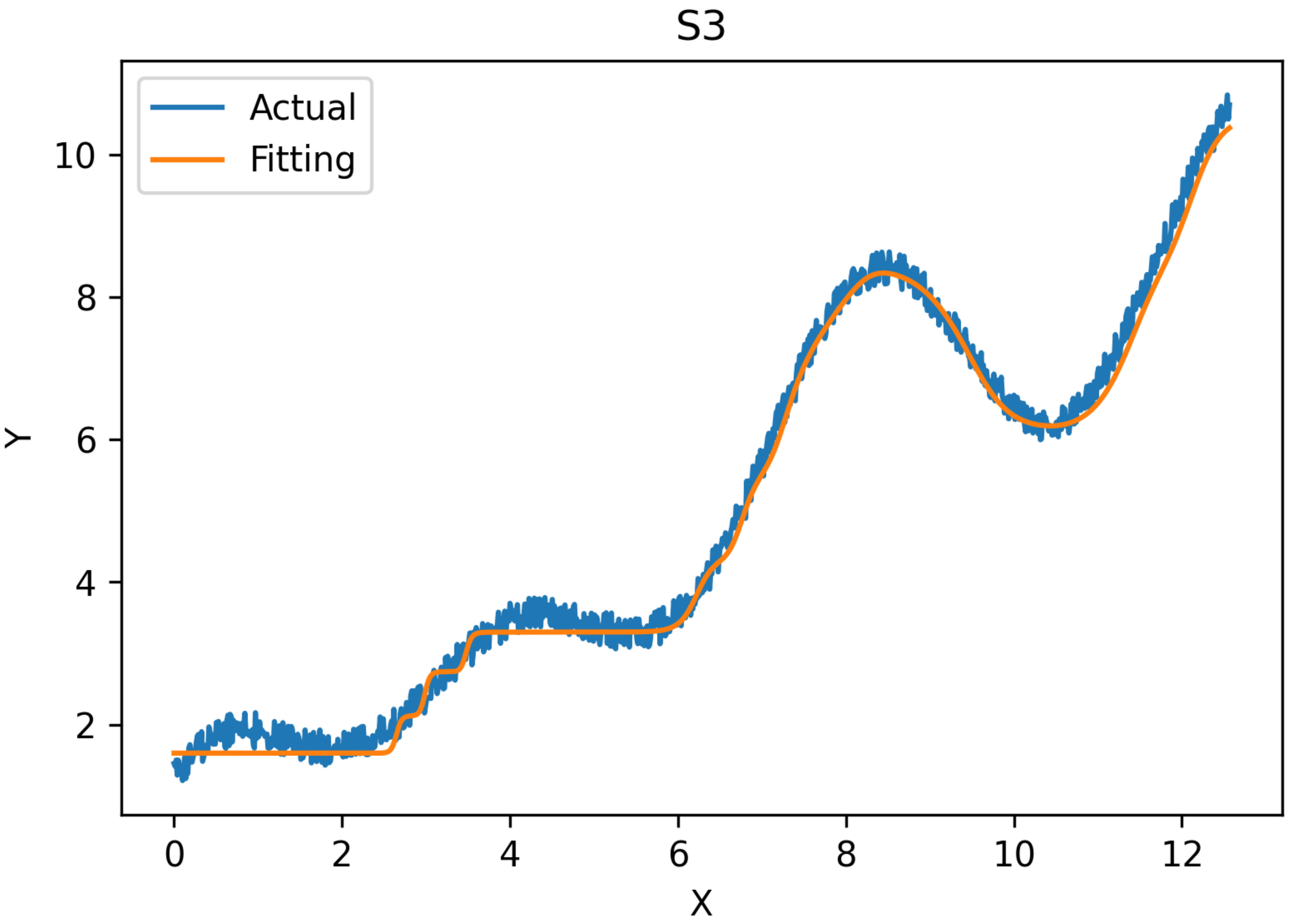
$$\hat{y} = \Sigma_{i=1}^{10} \left[w_i^{out} \frac{1}{1 + e^{-(w_i^{h1} x + b_i^{h1})}} \right] + b^{out}$$



#Parameters: 31

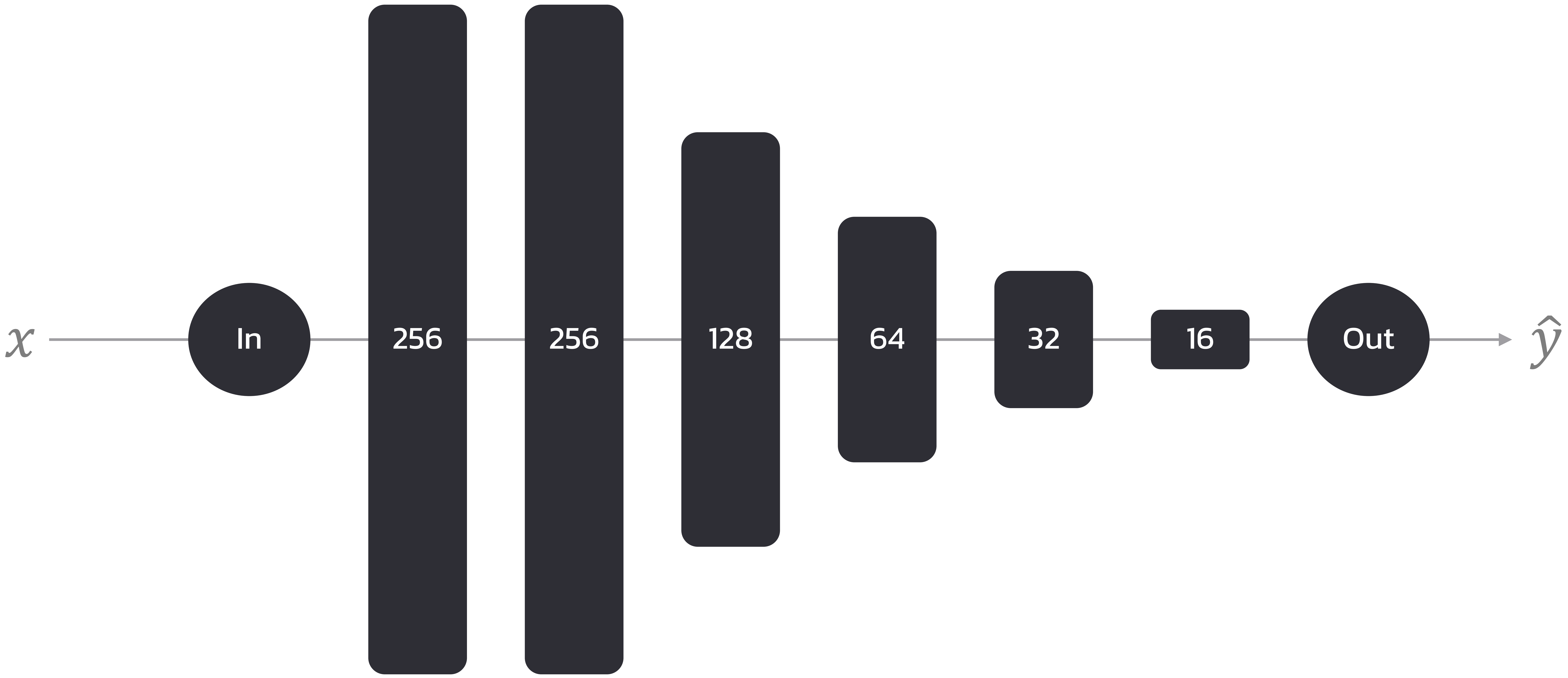


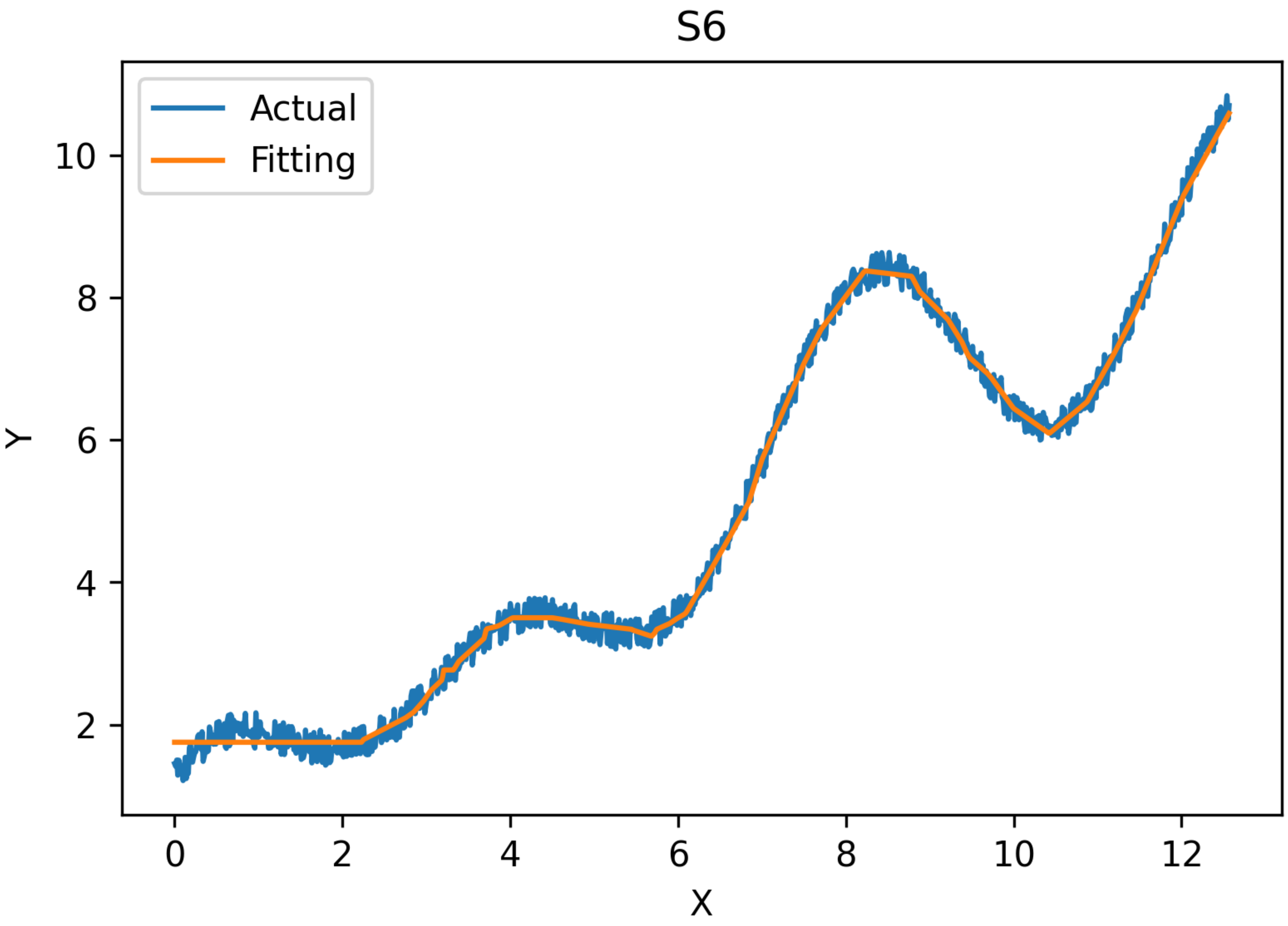
x
In
1
2
64
1
2
64
Out
$\hat{y}$
w_1^{h1}, b_1^{h1}
w_2^{h1}, b_2^{h1}
w_{64}^{h1}, b_{64}^{h1}
$w_{1,1}^{h2}, w_{1,2}^{h2}, \dots, w_{1,64}^{h2}, b_1^{h2}$
$w_{2,1}^{h2}, w_{2,2}^{h2}, \dots, w_{2,64}^{h2}, b_1^{h2}$
$w_{64,1}^{h2}, w_{64,2}^{h2}, \dots, w_{64,64}^{h2}, b_1^{h2}$
$w_1^{out}, w_2^{out} \dots, w_{64}^{out}, b^{out}$
128
Parameters
4160
Parameters
65
Parameters



#Parameters: 4,353

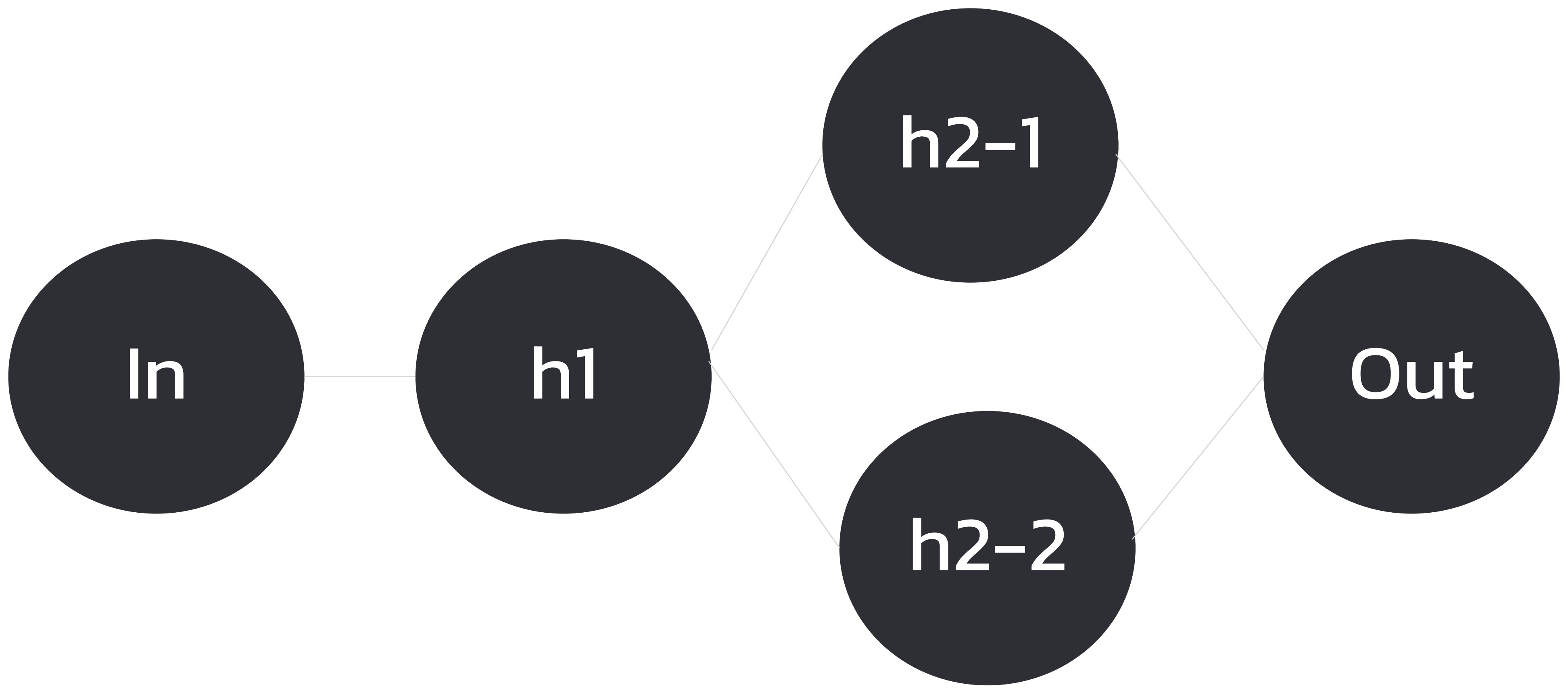
Deep Neural Network





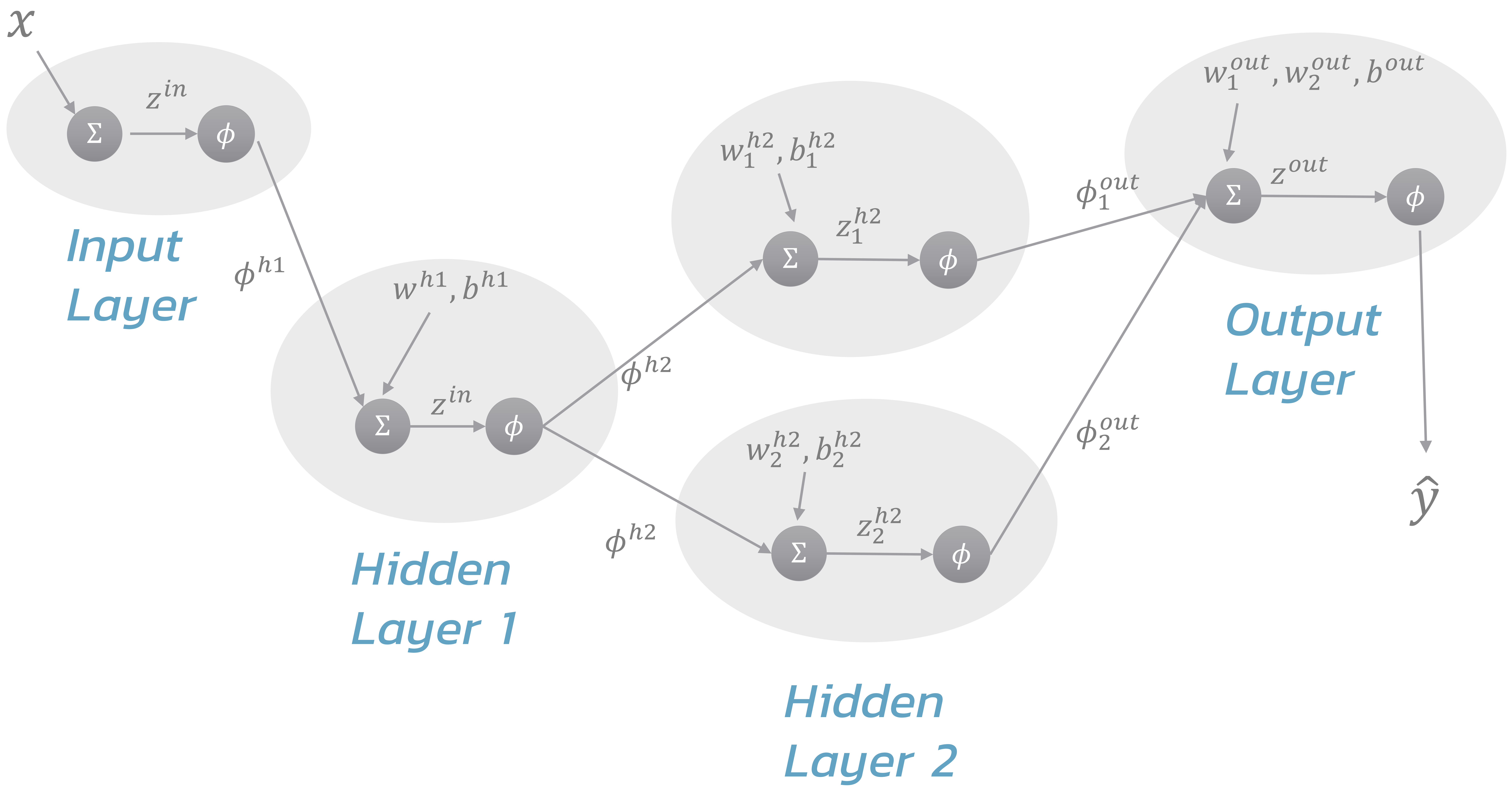
#Parameters: 110,081

Prediction

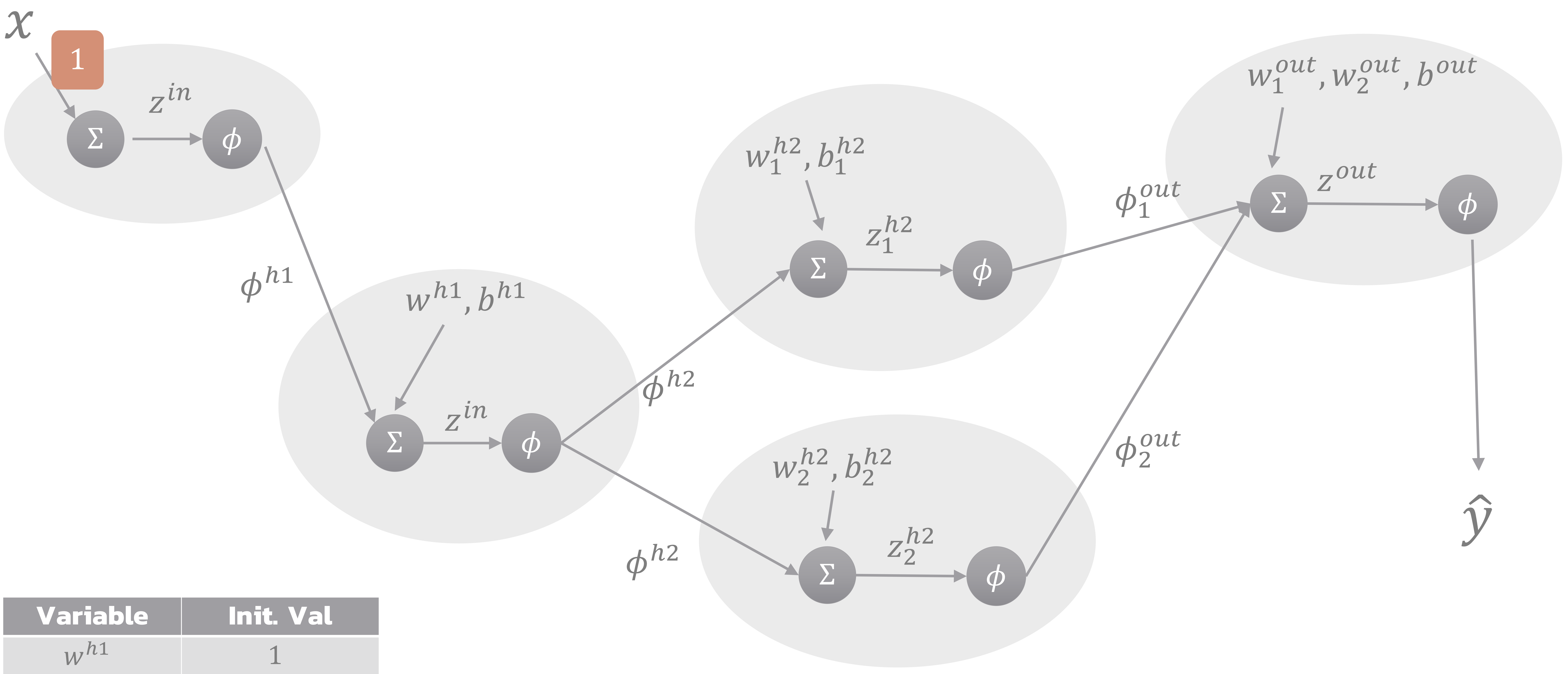


- Network
  - 2 hidden layers
  - Sigmoid activation
- Initialized weights and biases →
- Observation
  - $x = 1$
  - $y = 10$

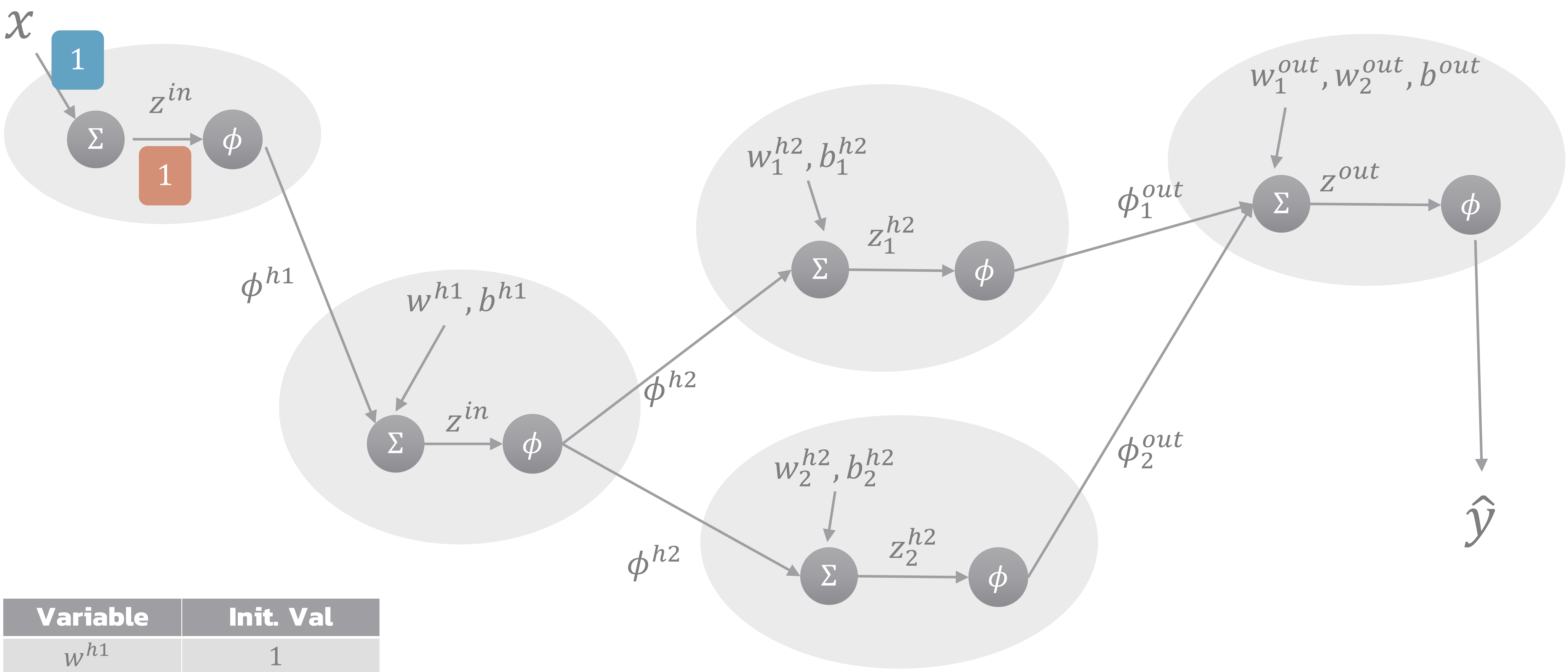
Variable	Init Val
w^{h1}	1
w_1^{h2}	2
w_2^{h2}	3
w_1^{out}	4
w_2^{out}	5
All biases	0



x
z^{in}
Σ
ϕ
Input Layer
ϕ^{h1}
w^{h1}, b^{h1}
z^{in}
Hidden Layer 1
ϕ^{h2}
ϕ^{h2}
w_1^{h2}, b_1^{h2}
z_1^{h2}
w_2^{h2}, b_2^{h2}
z_2^{h2}
Hidden Layer 2
ϕ_1^{out}
ϕ_2^{out}
$w_1^{out}, w_2^{out}, b^{out}$
z^{out}
Output Layer
$\hat{y}$

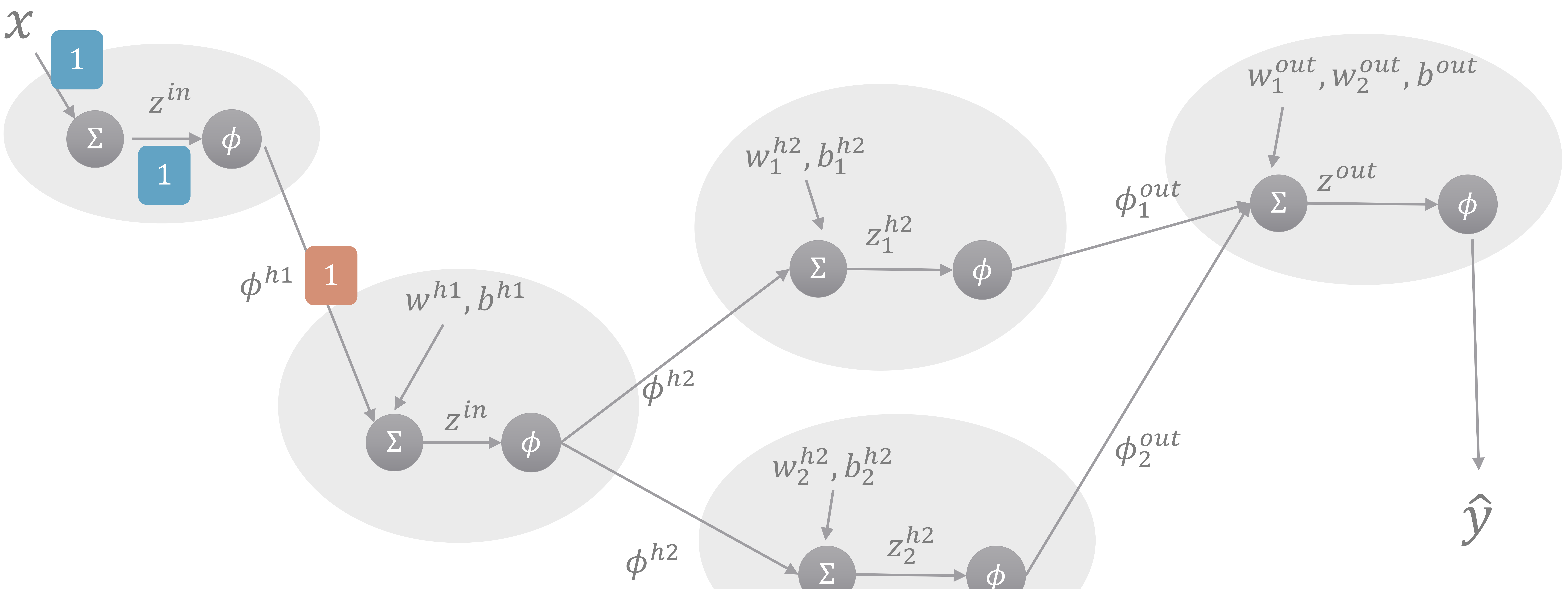


Variable	Init. Val
w^{h1}	1
w_1^{h2}	2
w_2^{h2}	3
w_1^{out}	4
w_2^{out}	5
All biases	0



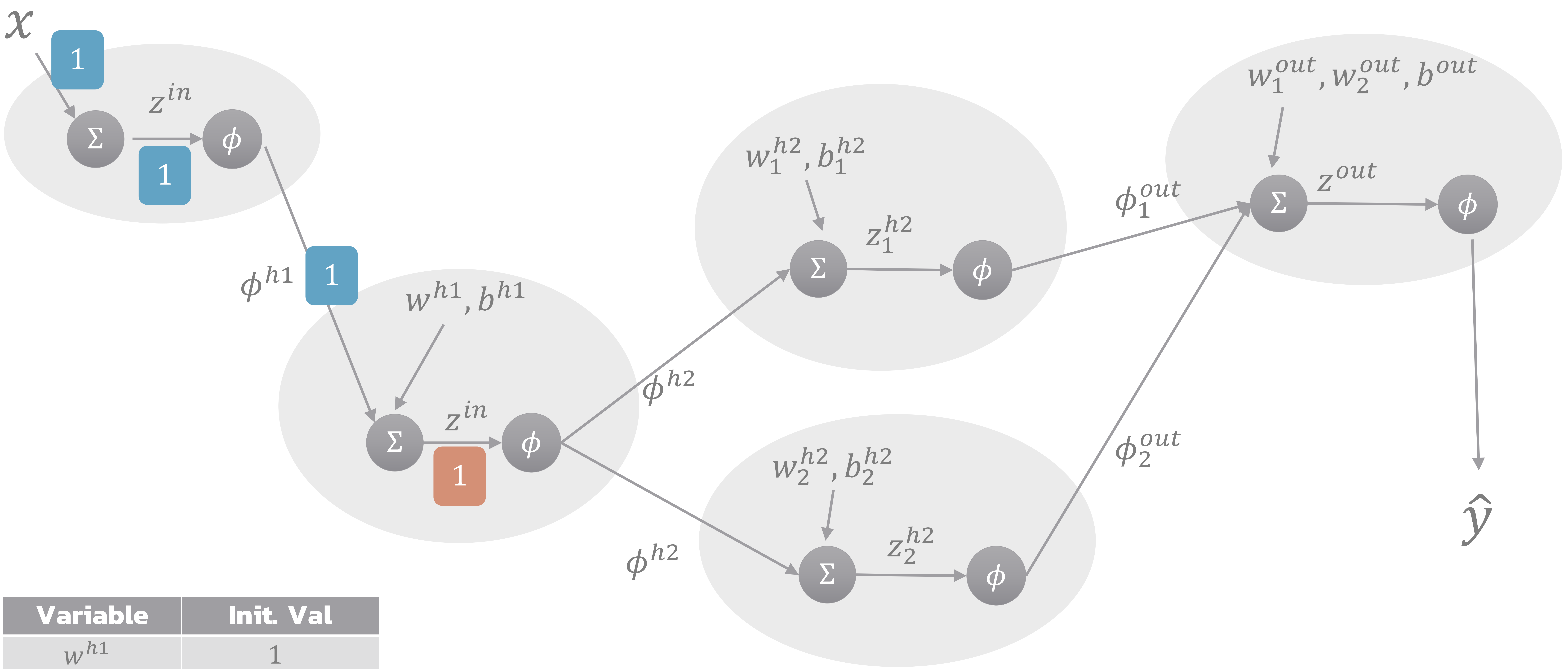
Variable	Init. Val
w^{h1}	1
w_1^{h2}	2
w_2^{h2}	3
w_1^{out}	4
w_2^{out}	5
All biases	0

$$z^{in} = x$$



Variable	Init. Val
w^{h1}	1
w_1^{h2}	2
w_2^{h2}	3
w_1^{out}	4
w_2^{out}	5
All biases	0

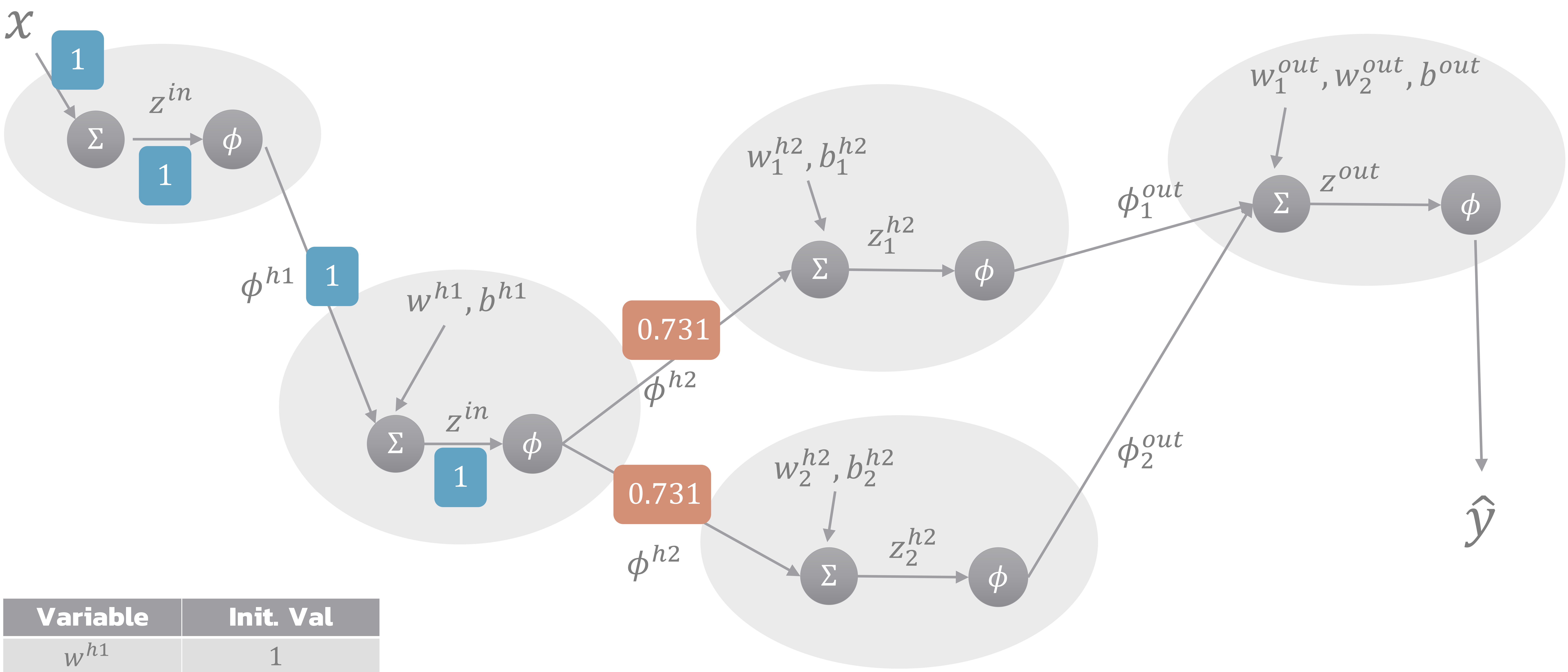
$$\phi^{h1} = z^{in}$$



Variable	Init. Val
w^{h1}	1
w_1^{h2}	2
w_2^{h2}	3
w_1^{out}	4
w_2^{out}	5
All biases	0

$$z^{in} = w^{h1}\phi^{h1} + b^{h1}$$

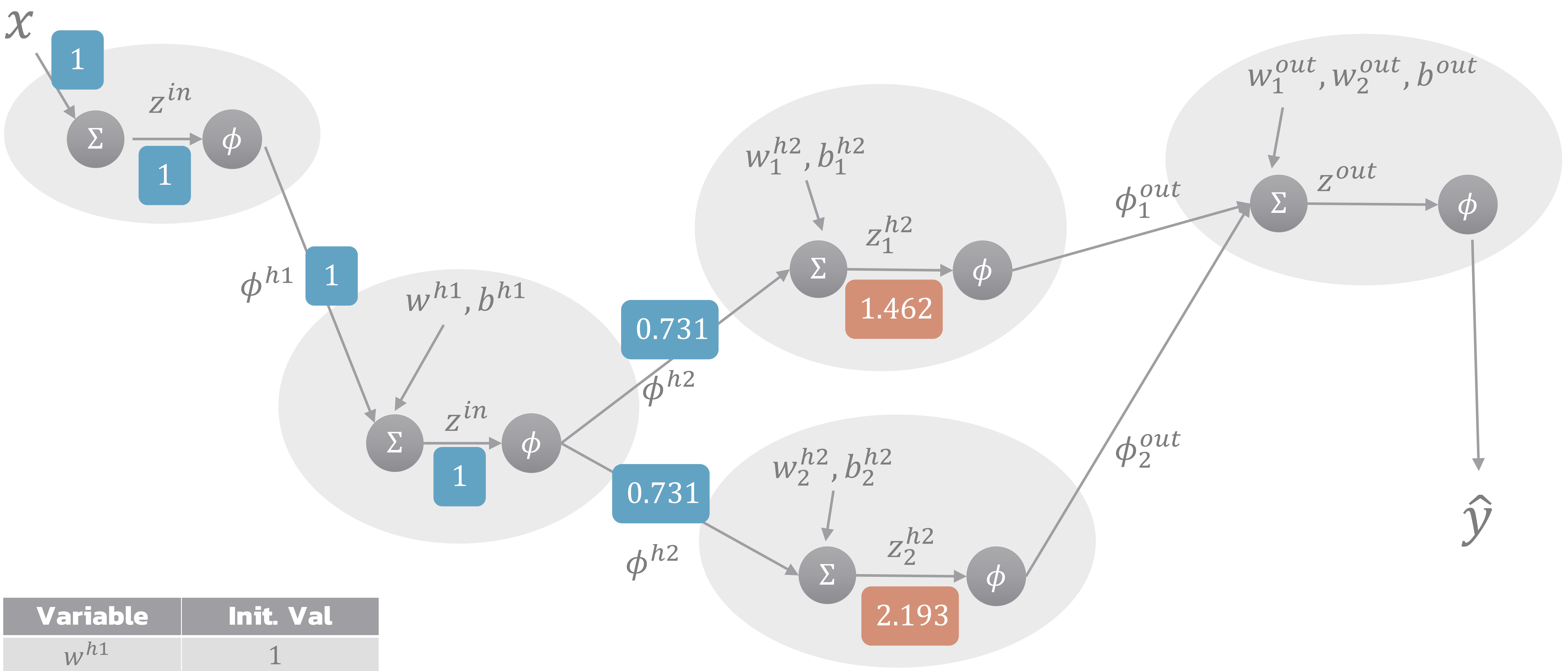
$$z^{in} = 1 \times 1 + 0 = 1$$



Variable	Init. Val
w^{h1}	1
w_1^{h2}	2
w_2^{h2}	3
w_1^{out}	4
w_2^{out}	5
All biases	0

$$\phi^{h2} = \frac{1}{1+e^{-z^{in}}}$$

$$\phi^{h2} = \frac{1}{1+e^{-1}} = 0.731$$



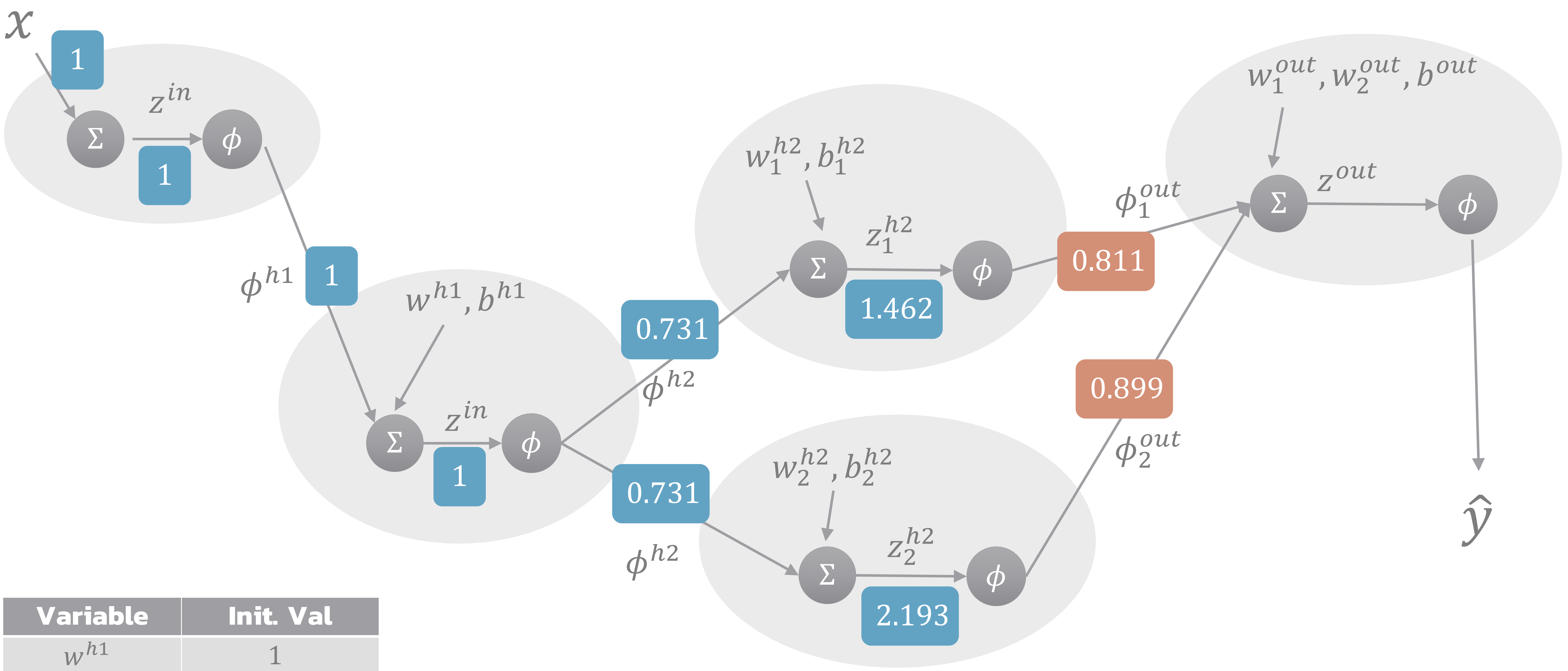
Variable	Init. Val
w^{h1}	1
w_1^{h2}	2
w_2^{h2}	3
w_1^{out}	4
w_2^{out}	5
All biases	0

$$z_1^{h2} = w_1^{h2}\phi^{h2} + b_1^{h2}$$

$$z_2^{h2} = w_2^{h2}\phi^{h2} + b_2^{h2}$$

$$z_1^{h2} = 2 \times 0.731 + 0 = 1.462$$

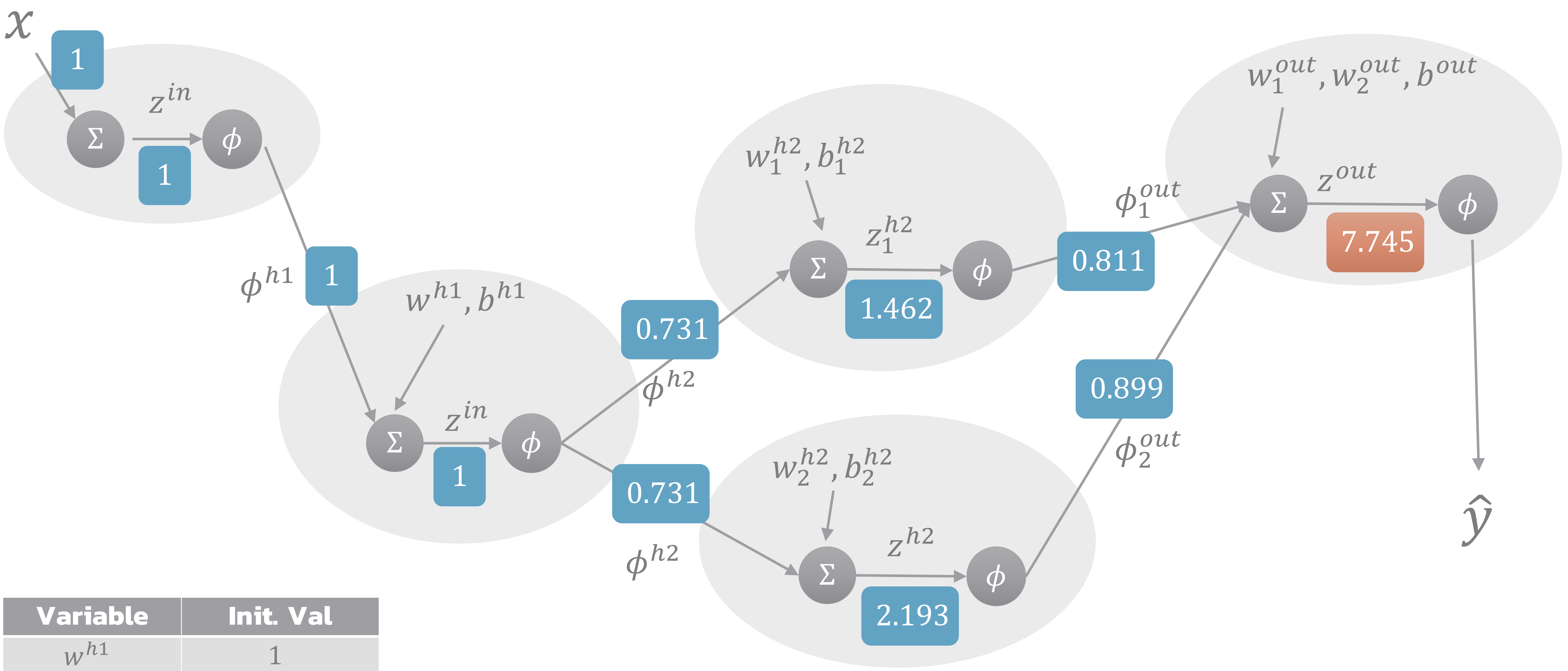
$$z_2^{h2} = 3 \times 0.731 + 0 = 2.193$$



Variable	Init. Val
w^{h1}	1
w_1^{h2}	2
w_2^{h2}	3
w_1^{out}	4
w_2^{out}	5
All biases	0

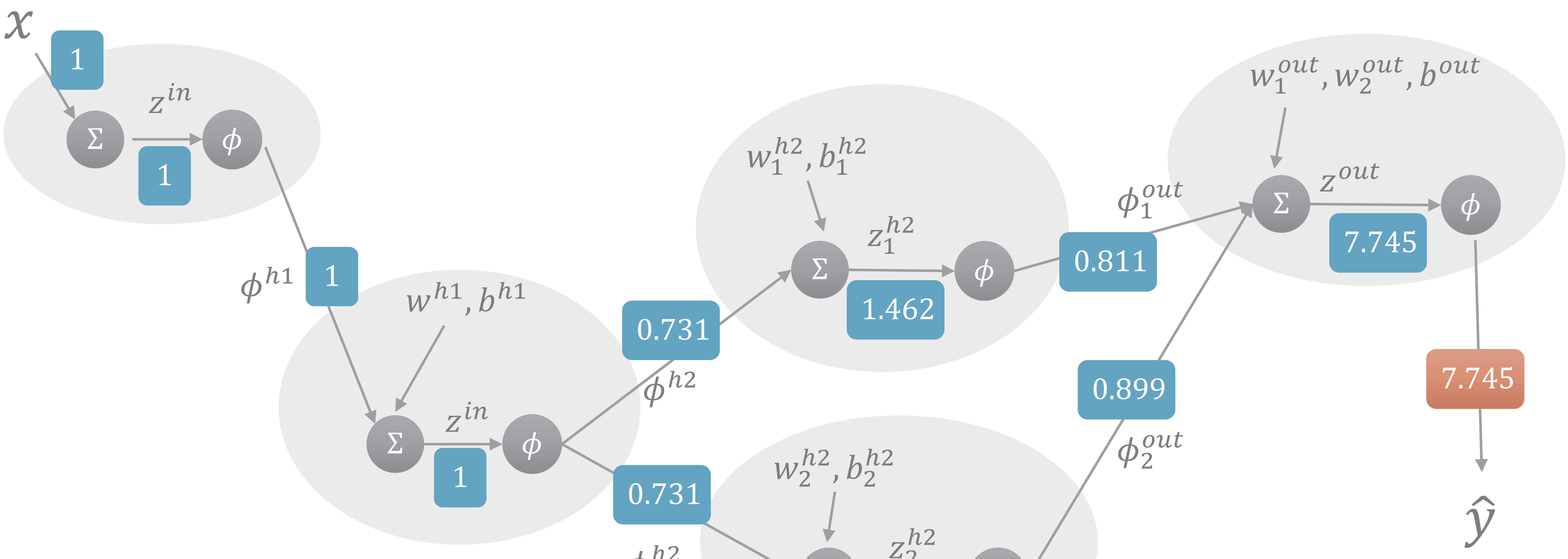
$$\phi_1^{out} = \frac{1}{1 + e^{-z_1^{h2}}}$$

$$\phi_2^{out} = \frac{1}{1 + e^{-z_2^{h2}}}$$



Variable	Init. Val
w^{h1}	1
w_1^{h2}	2
w_2^{h2}	3
w_1^{out}	4
w_2^{out}	5
All biases	0

$$z^{out} = w_1^{out}\phi_1^{out} + w_2^{out}\phi_2^{out} + b^{out}$$



Variable	Init. Val
w^{h1}	1
w_1^{h2}	2
w_2^{h2}	3
w_1^{out}	4
w_2^{out}	5
All biases	0

$$\hat{y} = z^{out}$$

Loss

- Quantify how *"bad"* our model is.
- Mean Square Error (MSE)
  - $L = \frac{1}{N} \Sigma_{i=1}^{N} (y_i - \hat{y}_i)^2$
- In our example
  - $L = (10 - 7.745)^2 = 5.082$

Parameter Update

- Backpropagation
  - Base on chain rule
- Computational graph
  - Compute input gradient signal at each node.
  - Send the gradient signal backward.

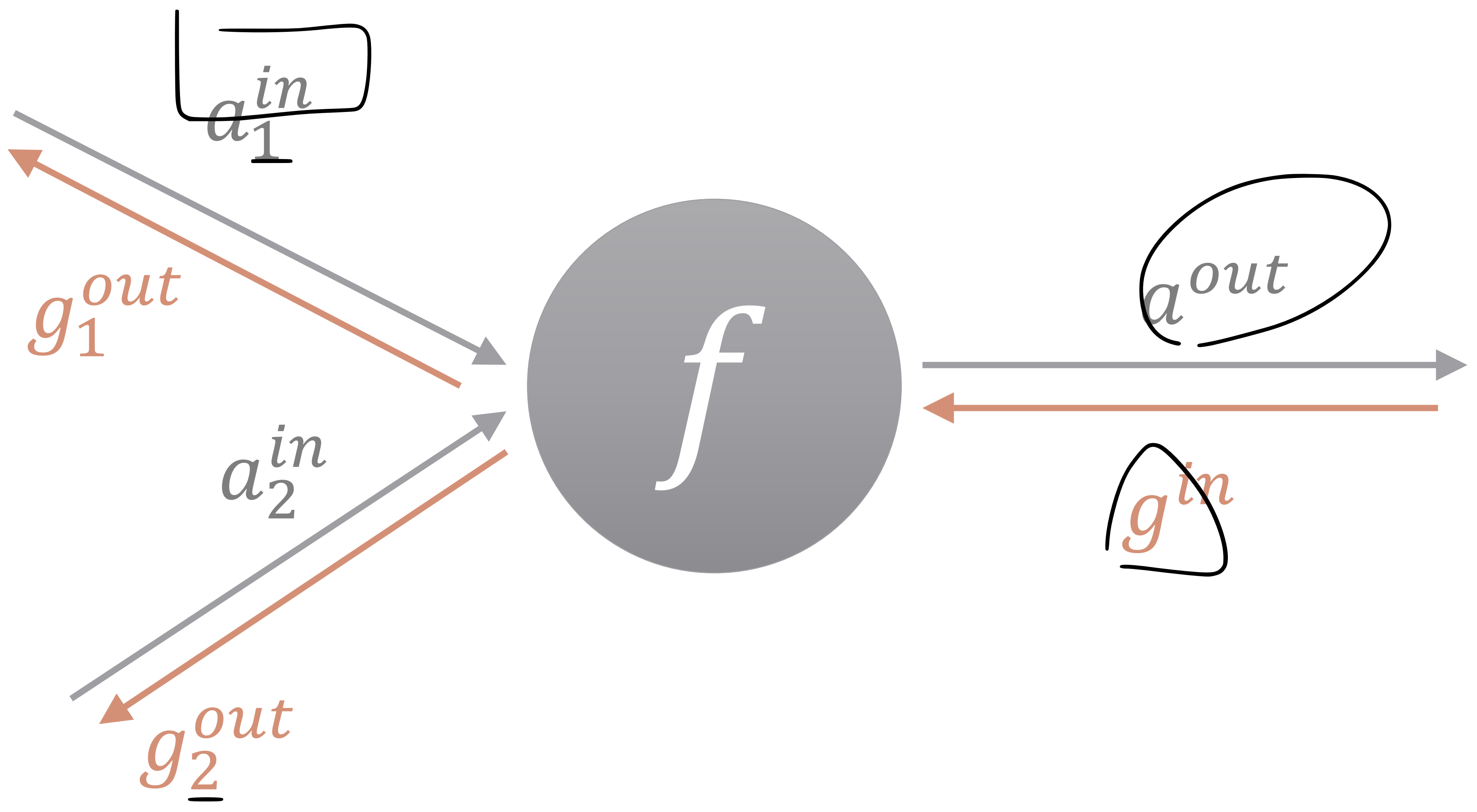
Computational Graph

$$g^{out} = g^{in} \frac{\partial a^{out}}{\partial a^{in}}$$

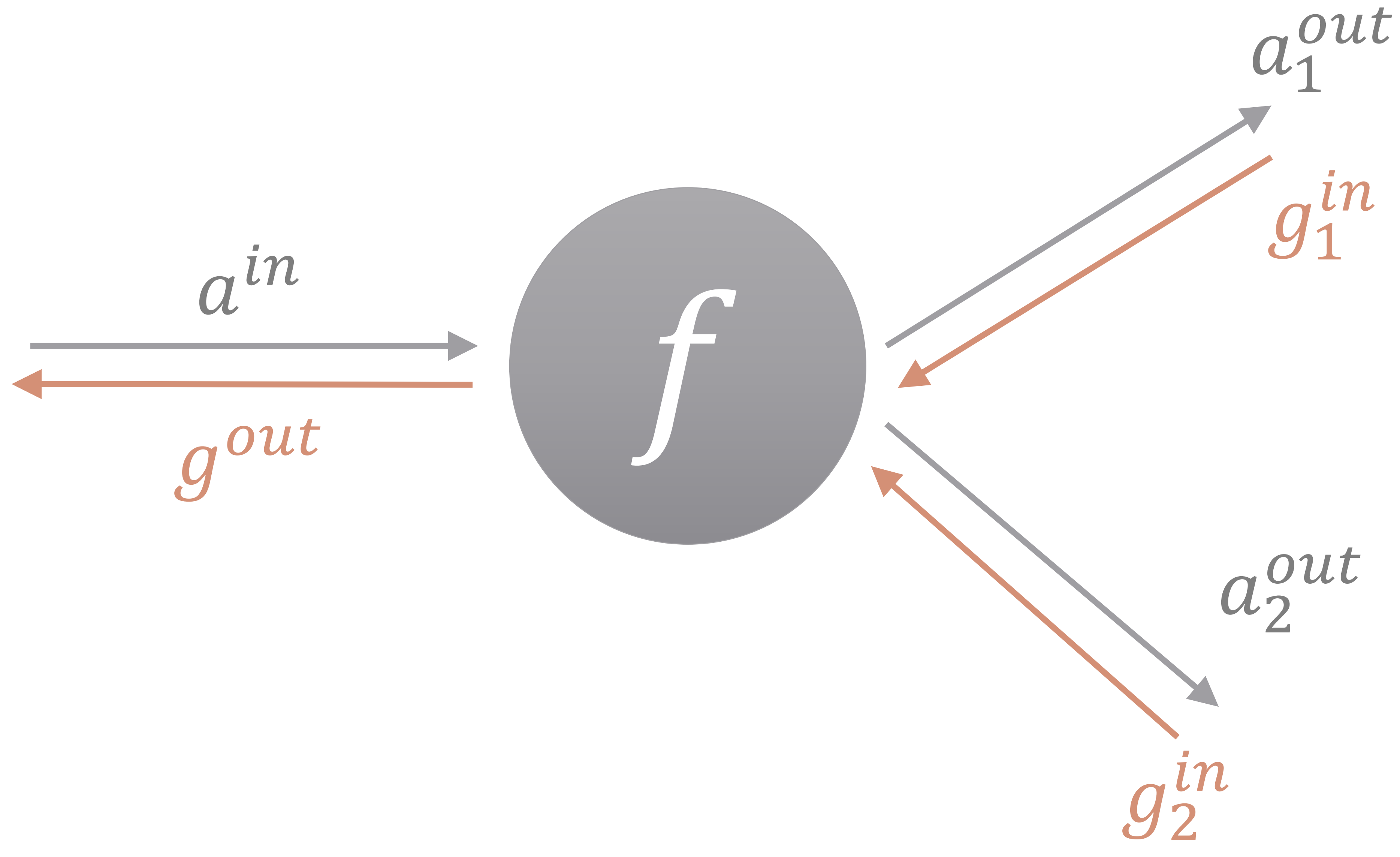
Computational Graph

$$g_1^{out} = g^{in} \frac{\partial a^{out}}{\partial a_1^{in}}$$

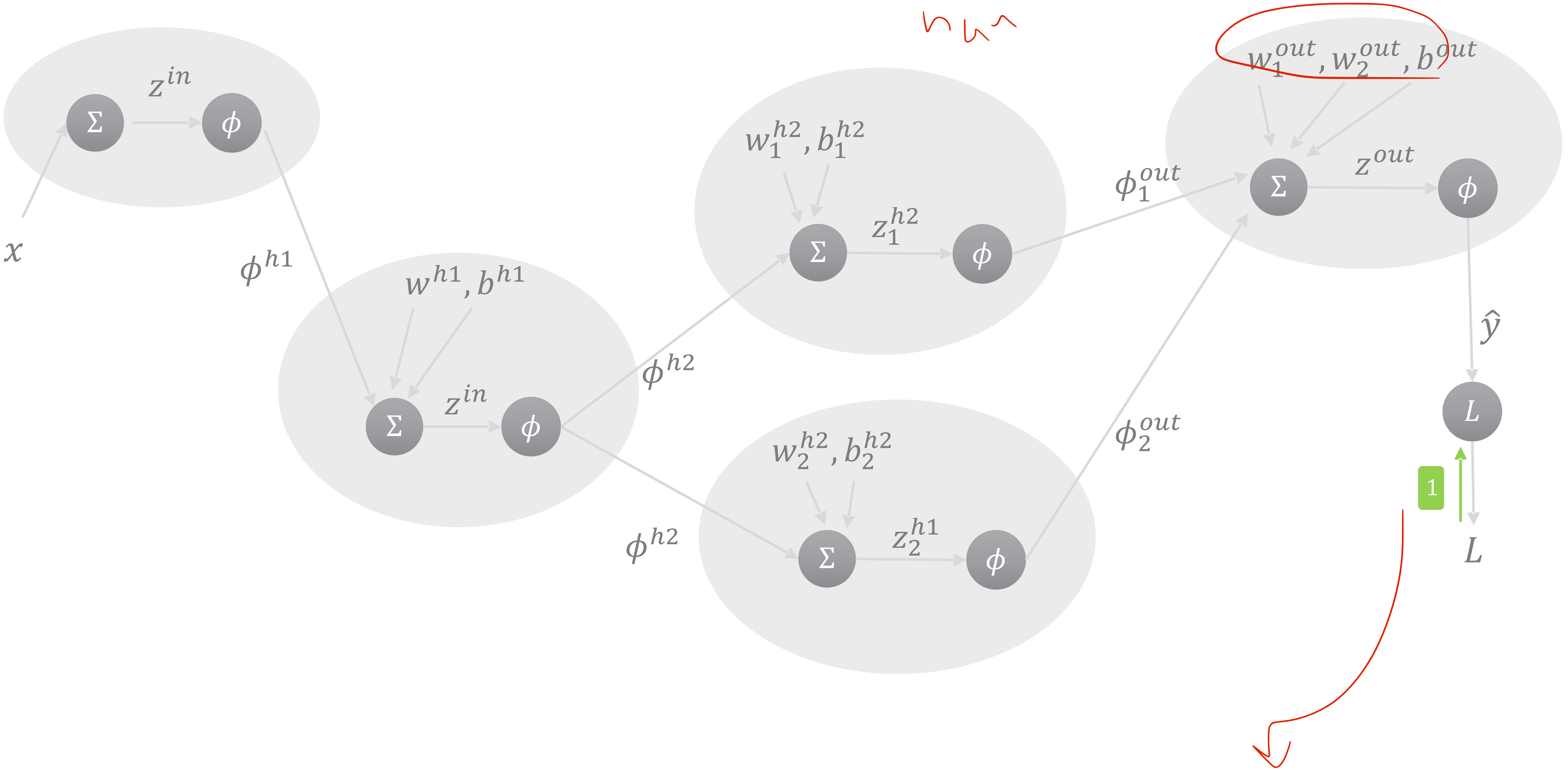
$$g_2^{out} = g^{in} \frac{\partial a^{out}}{\partial a_2^{in}}$$



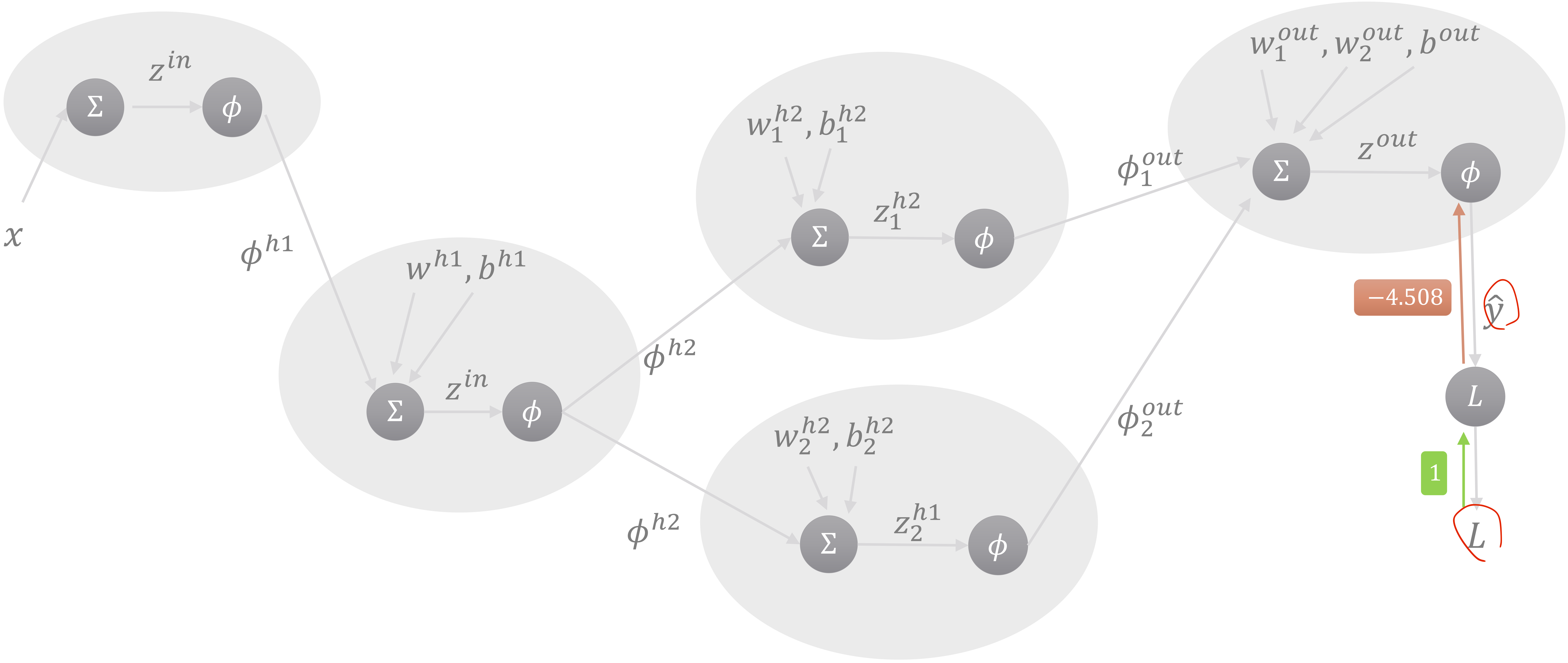
Computational Graph



$$g^{out} = g_1^{in} \frac{\partial a_1^{out}}{\partial a^{in}} + g_2^{in} \frac{\partial a_2^{out}}{\partial a^{in}}$$



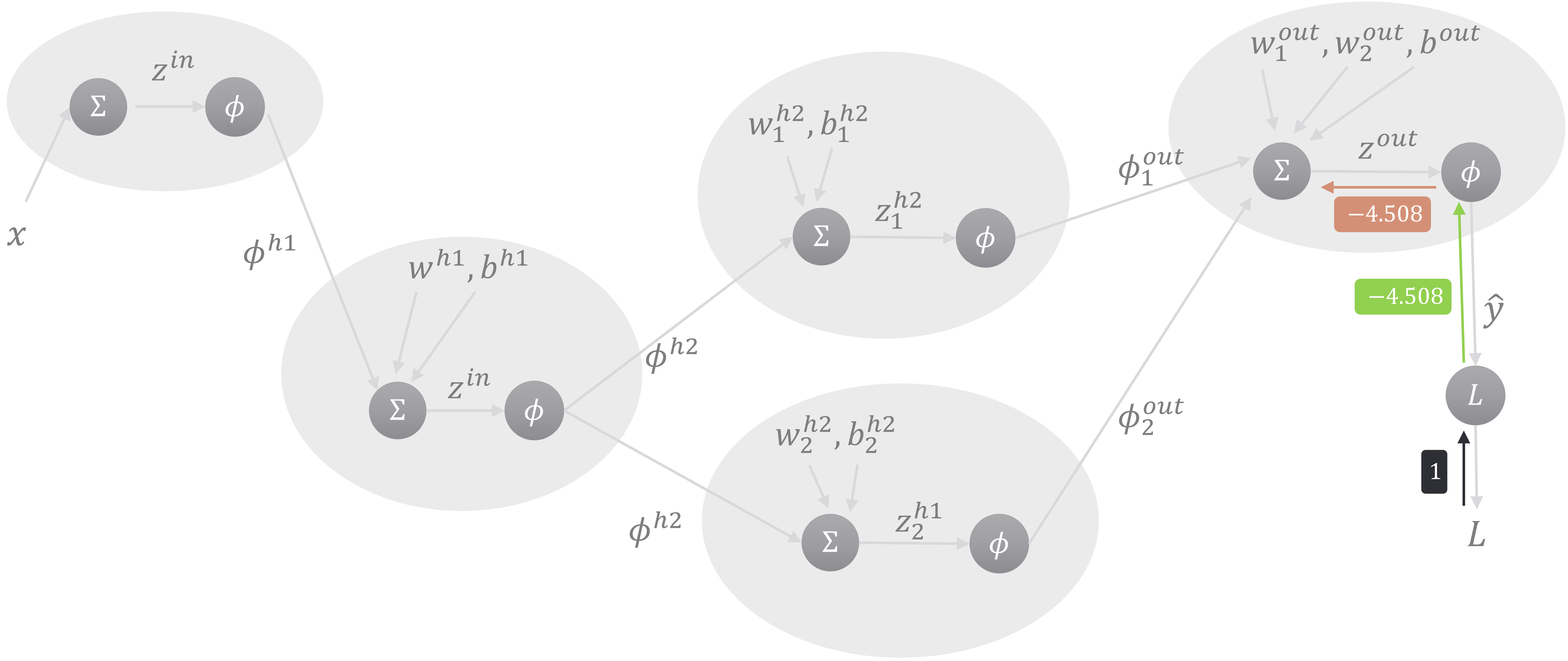
Add another "L" node with gradient input of 1.



$$g^{in} = \frac{\partial L}{\partial L} = 1$$

$$\frac{\partial L}{\partial \hat{y}} = 2(\hat{y} - y) = -4.508$$

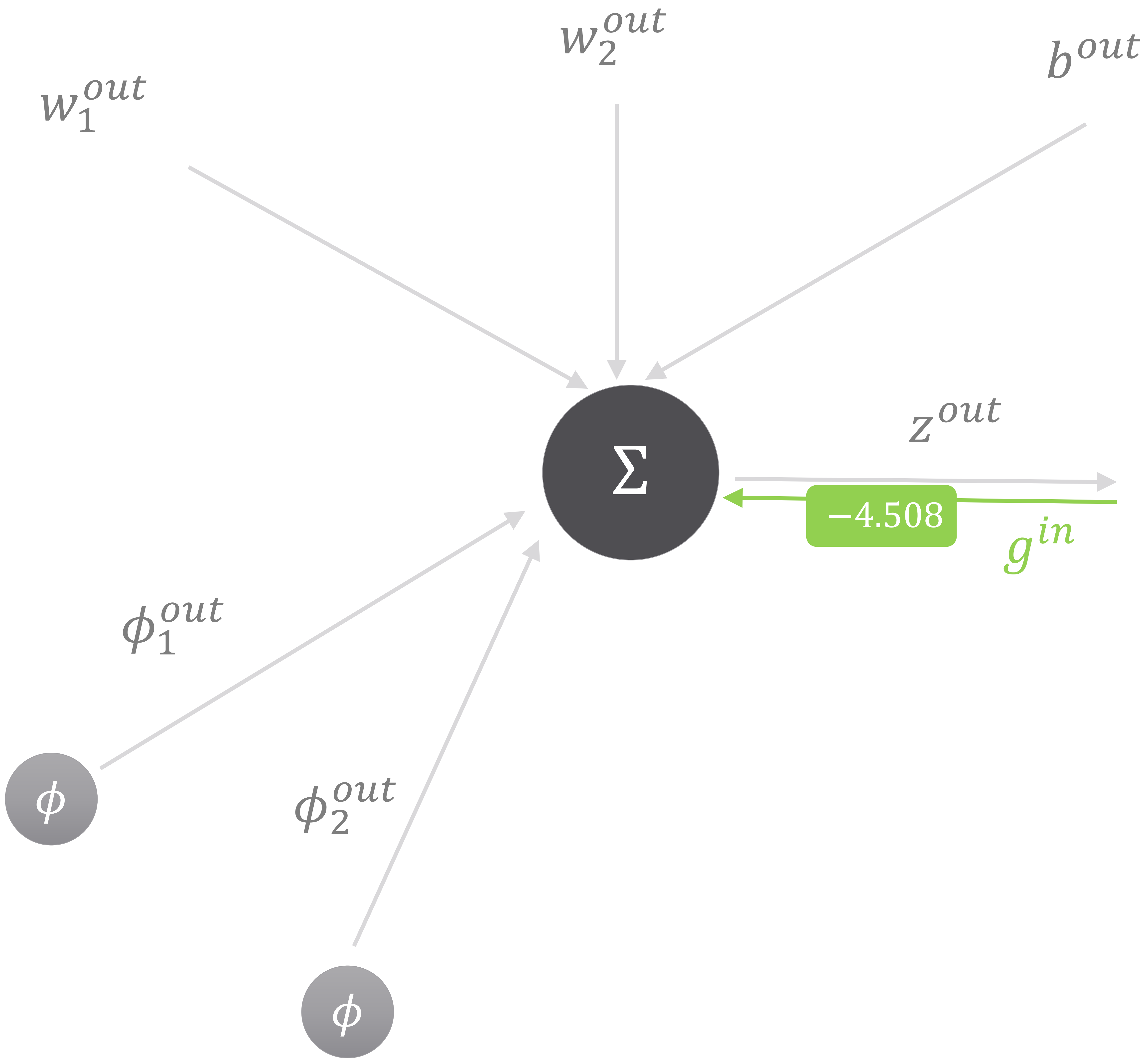
$$g^{out} = g^{in}\frac{\partial L}{\partial \hat{y}} = -4.508$$



$g^{in} = -4.508$

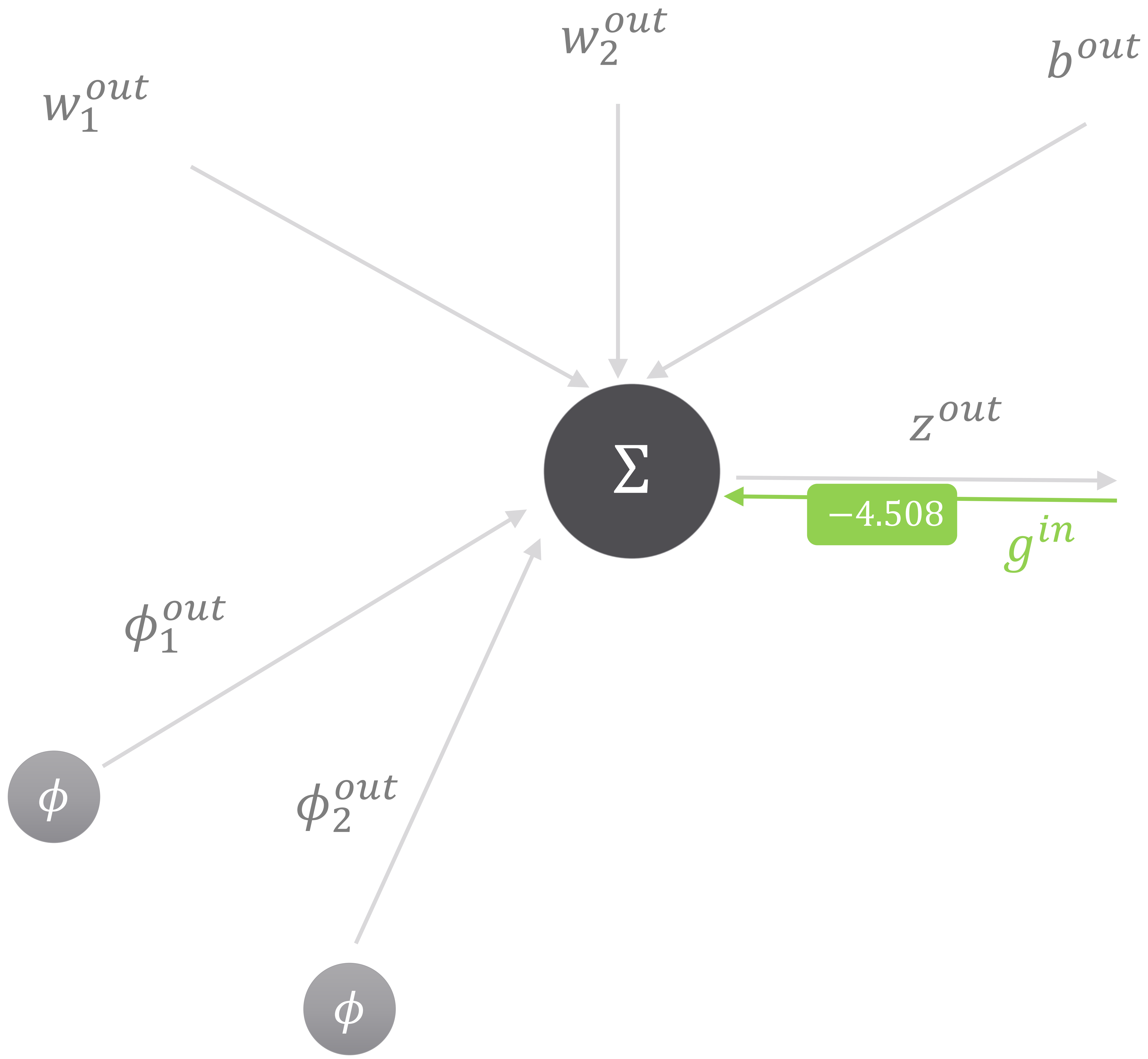
$\frac{\partial \hat{y}}{\partial z^{out}} = 1$

$g^{out} = g^{in} \frac{\partial \hat{y}}{\partial z^{out}} = -4.508$



$$z^{out} = w_1^{out}\phi_1^{out} + w_2^{out}\phi_2^{out} + b^{out}$$

$$g^{in} = -4.508$$



$$z^{out} = w_1^{out}\phi_1^{out} + w_2^{out}\phi_2^{out} + b^{out}$$

$$g^{in} = -4.508$$

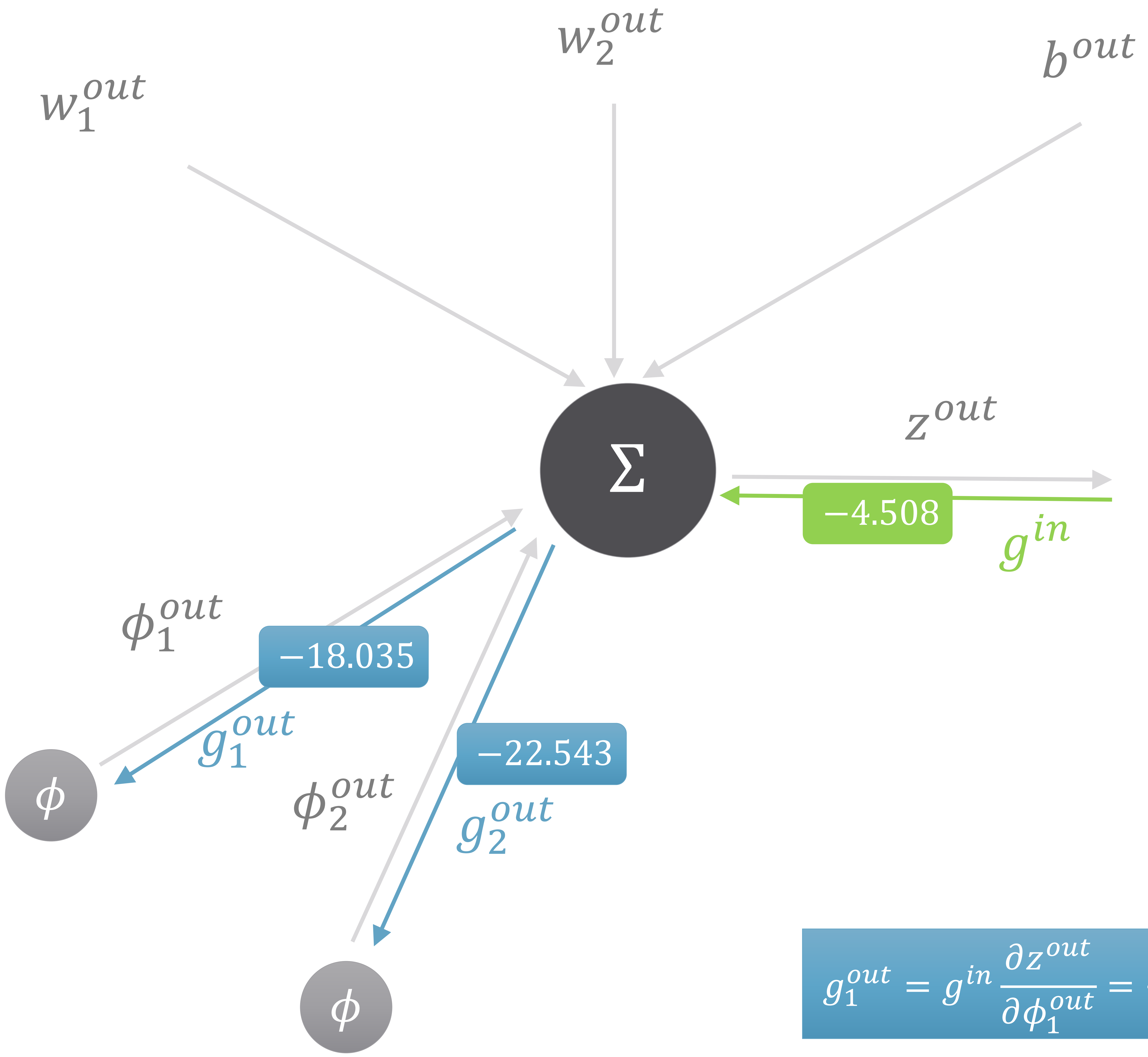
$$\frac{\partial z^{out}}{\partial \phi_1^{out}} = w_1^{out} = 4$$

$$\frac{\partial z^{out}}{\partial w_1^{out}} = \phi_1^{out} = 0.811$$

$$\frac{\partial z^{out}}{\partial \phi_2^{out}} = w_2^{out} = 5$$

$$\frac{\partial z^{out}}{\partial w_2^{out}} = \phi_2^{out} = 0.899$$

$$\frac{\partial z^{out}}{\partial b^{out}} = 1$$



$$z^{out} = w_1^{out}\phi_1^{out} + w_2^{out}\phi_2^{out} + b^{out}$$

$$g^{in} = -4.508$$

$$\frac{\partial z^{out}}{\partial \phi_1^{out}} = w_1^{out} = 4$$

$$\frac{\partial z^{out}}{\partial \phi_2^{out}} = w_2^{out} = 5$$

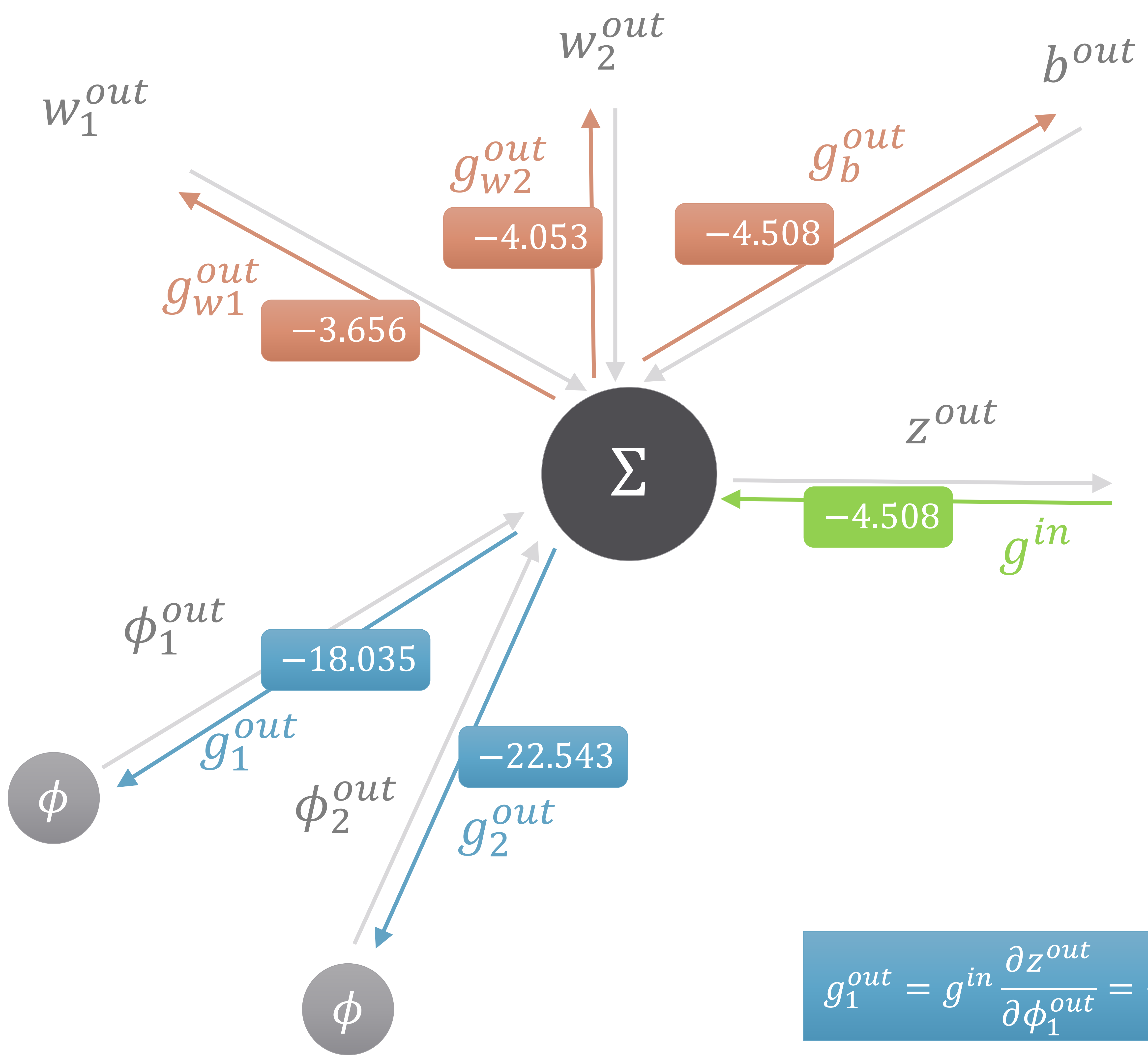
$$\frac{\partial z^{out}}{\partial w_1^{out}} = \phi_1^{out} = 0.811$$

$$\frac{\partial z^{out}}{\partial w_2^{out}} = \phi_2^{out} = 0.899$$

$$\frac{\partial z^{out}}{\partial b^{out}} = 1$$

$$g_1^{out} = g^{in}\frac{\partial z^{out}}{\partial \phi_1^{out}} = -4.508 \times 4 = -18.035$$

$$g_2^{out} = g^{in}\frac{\partial z^{out}}{\partial \phi_2^{out}} = -4.508 \times 5 = -22.543$$



$$z^{out} = w_1^{out}\phi_1^{out} + w_2^{out}\phi_2^{out} + b^{out}$$

$$g^{in} = -4.508$$

$$\frac{\partial z^{out}}{\partial \phi_1^{out}} = w_1^{out} = 4$$

$$\frac{\partial z^{out}}{\partial w_1^{out}} = \phi_1^{out} = 0.811$$

$$\frac{\partial z^{out}}{\partial \phi_2^{out}} = w_2^{out} = 5$$

$$\frac{\partial z^{out}}{\partial w_2^{out}} = \phi_2^{out} = 0.899$$

$$\frac{\partial z^{out}}{\partial b^{out}} = 1$$

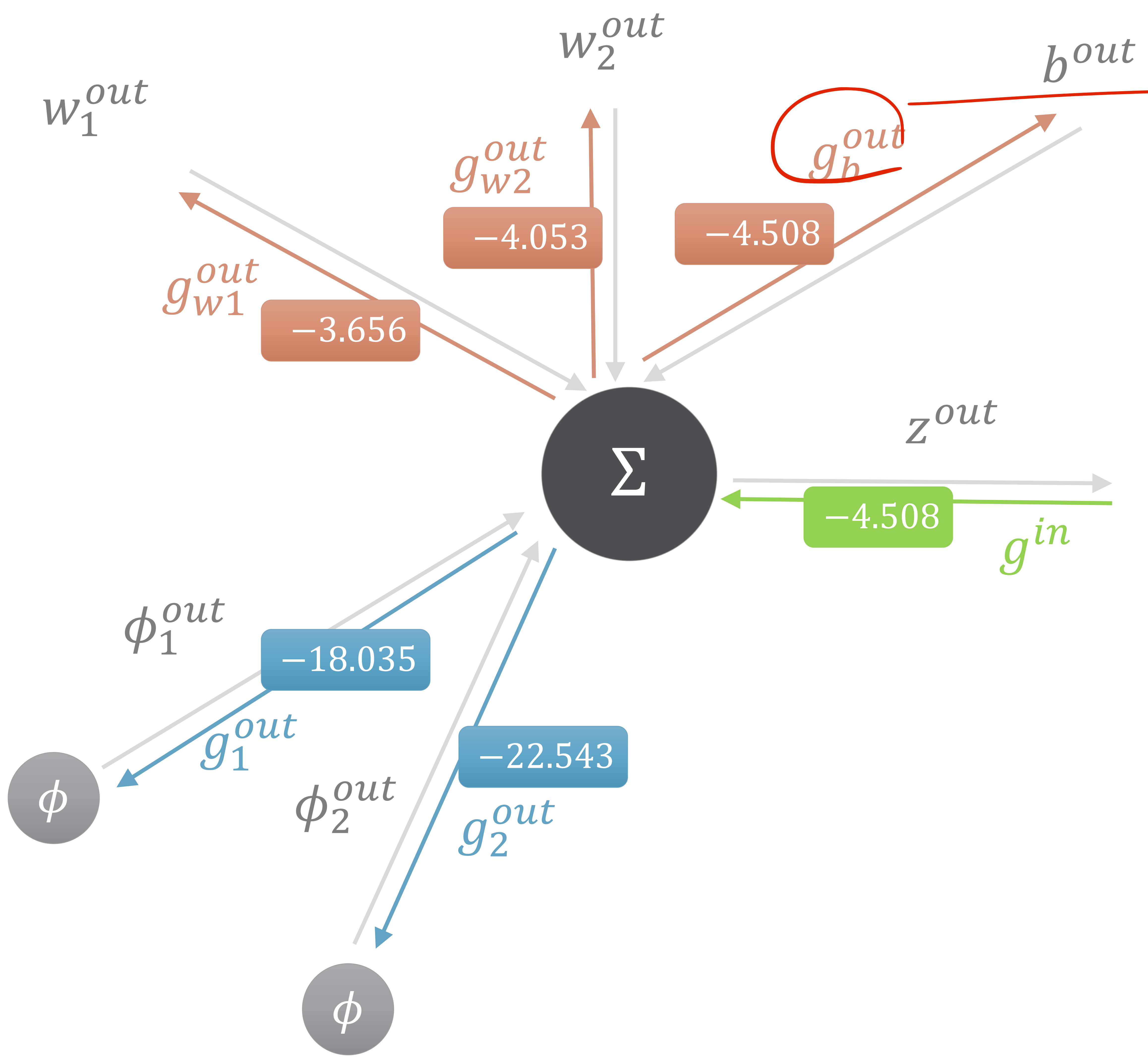
$$g_1^{out} = g^{in}\frac{\partial z^{out}}{\partial \phi_1^{out}} = -4.508 \times 4 = -18.035$$

$$g_{w1}^{out} = g^{in}\frac{\partial z^{out}}{\partial w_1^{out}} = -4.508 \times 0.811 = -3.656$$

$$g_2^{out} = g^{in}\frac{\partial z^{out}}{\partial \phi_2^{out}} = -4.508 \times 5 = -22.543$$

$$g_{w2}^{out} = g^{in}\frac{\partial z^{out}}{\partial w_2^{out}} = -4.508 \times 0.899 = -4.053$$

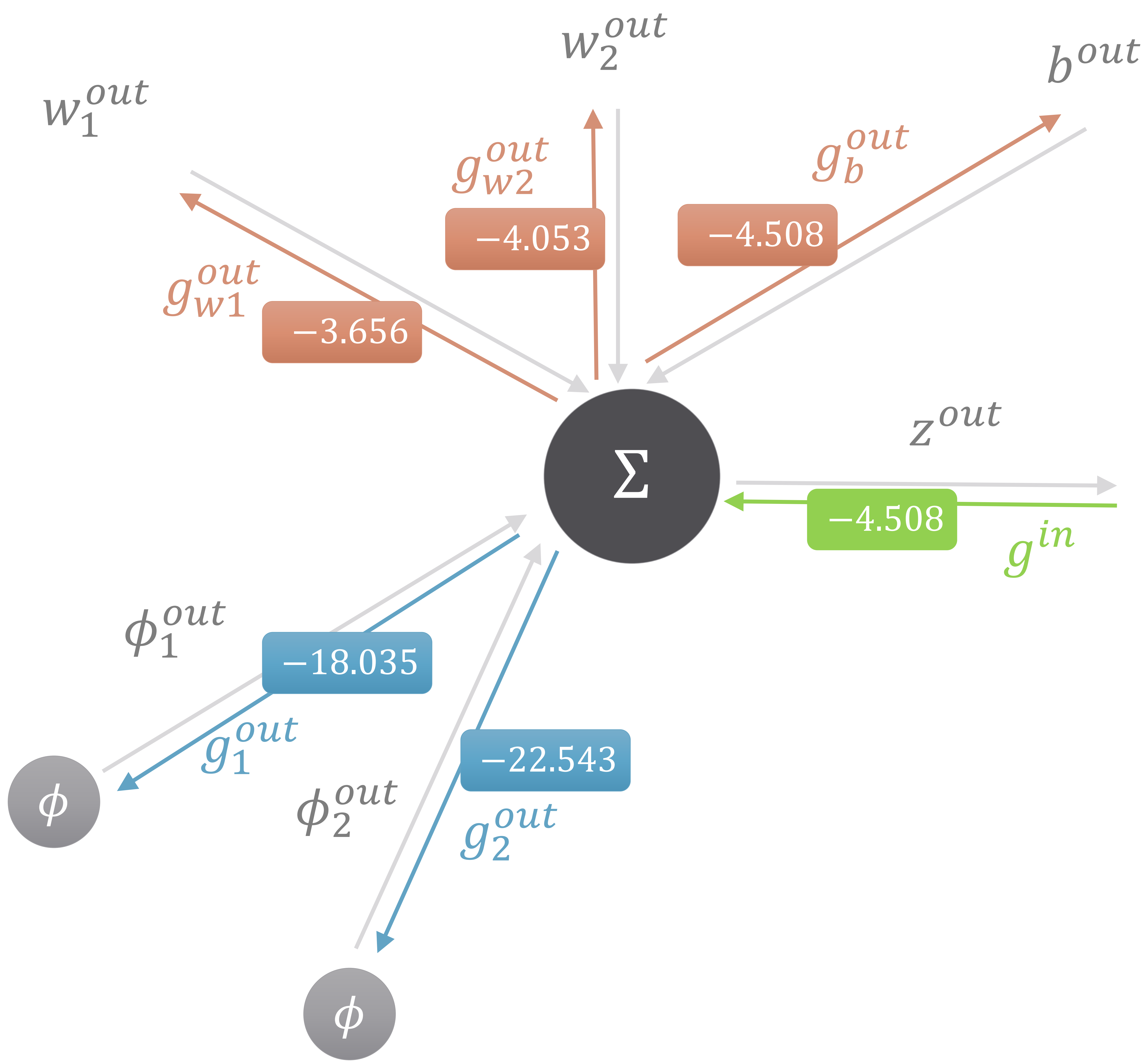
$$g_b^{out} = g^{in}\frac{\partial z^{out}}{\partial b^{out}} = -4.508 \times 1 = -4.508$$



Update Parameters

$$w^{new} = w^{old} - (\eta \times g)$$

Learning Rate
(0.1)

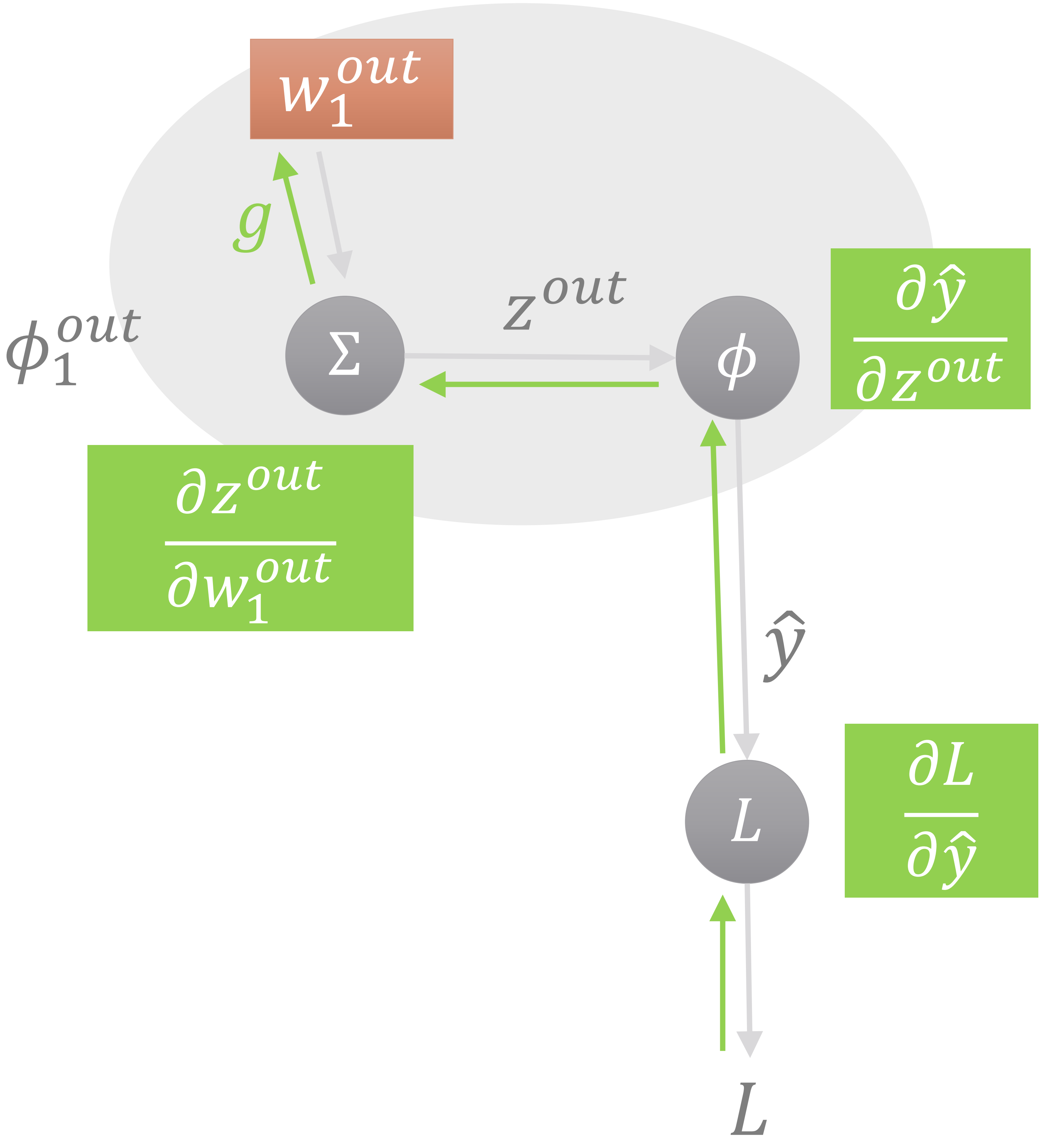


Update Parameters

$$w^{new} = w^{old} - (\eta \times g)$$

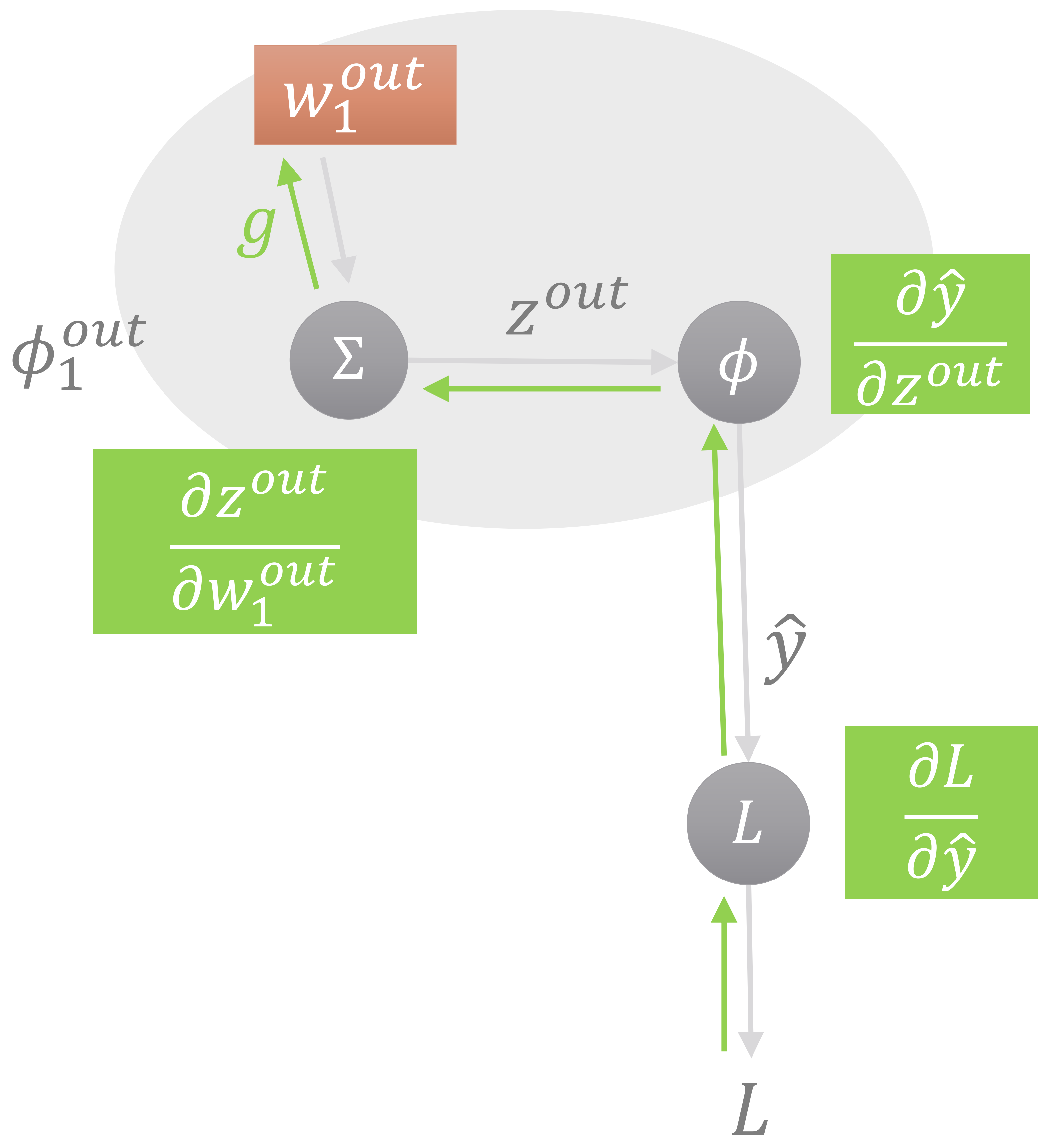
Learning Rate
(0.1)

Variable	Init Val	$\eta \times g$	Updated Val
w_1^{out}	4	-0.366	4.366
w_2^{out}	5	-0.405	5.405
b^{out}	0	-0.451	0.451



Chain Rule

$$g = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z^{out}} \frac{\partial z^{out}}{\partial w_1^{out}} = \frac{\partial L}{\partial w_1^{out}}$$



Chain Rule

$$g = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z^{out}} \frac{\partial z^{out}}{\partial w_1^{out}} = \frac{\partial L}{\partial w_1^{out}}$$

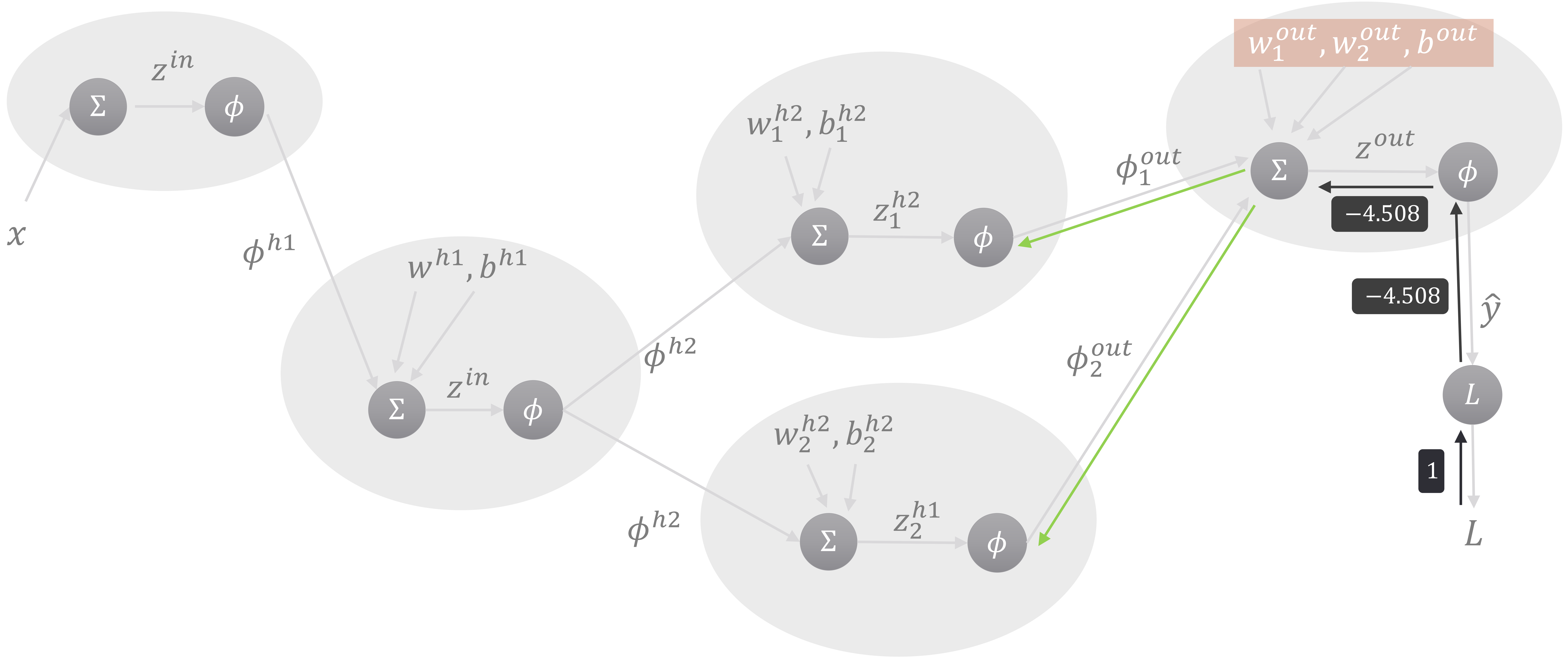
Previously

$$w^{new} = w^{old} - (\eta \times g)$$

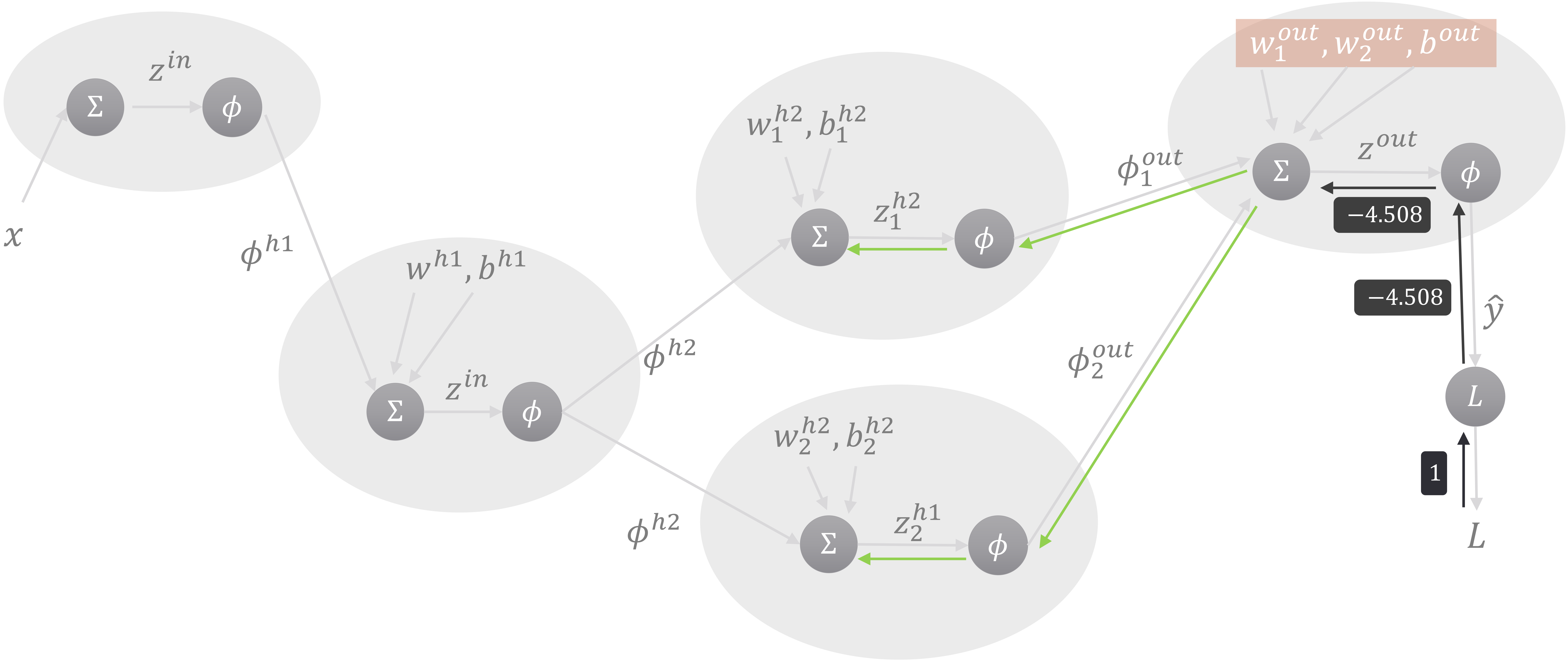
Formally

$$w^{new} = w^{old} - \eta \frac{\partial L}{\partial w_1^{out}}$$

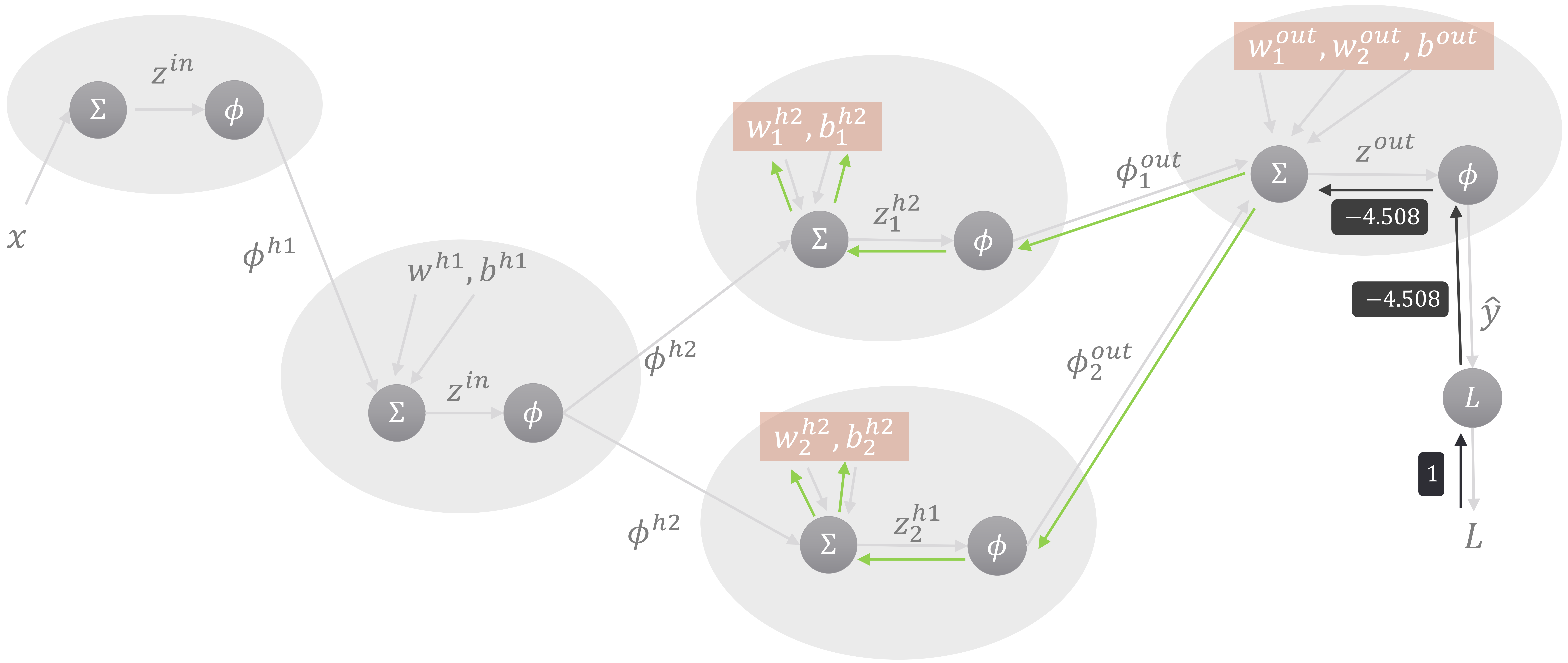
Gradient Descent



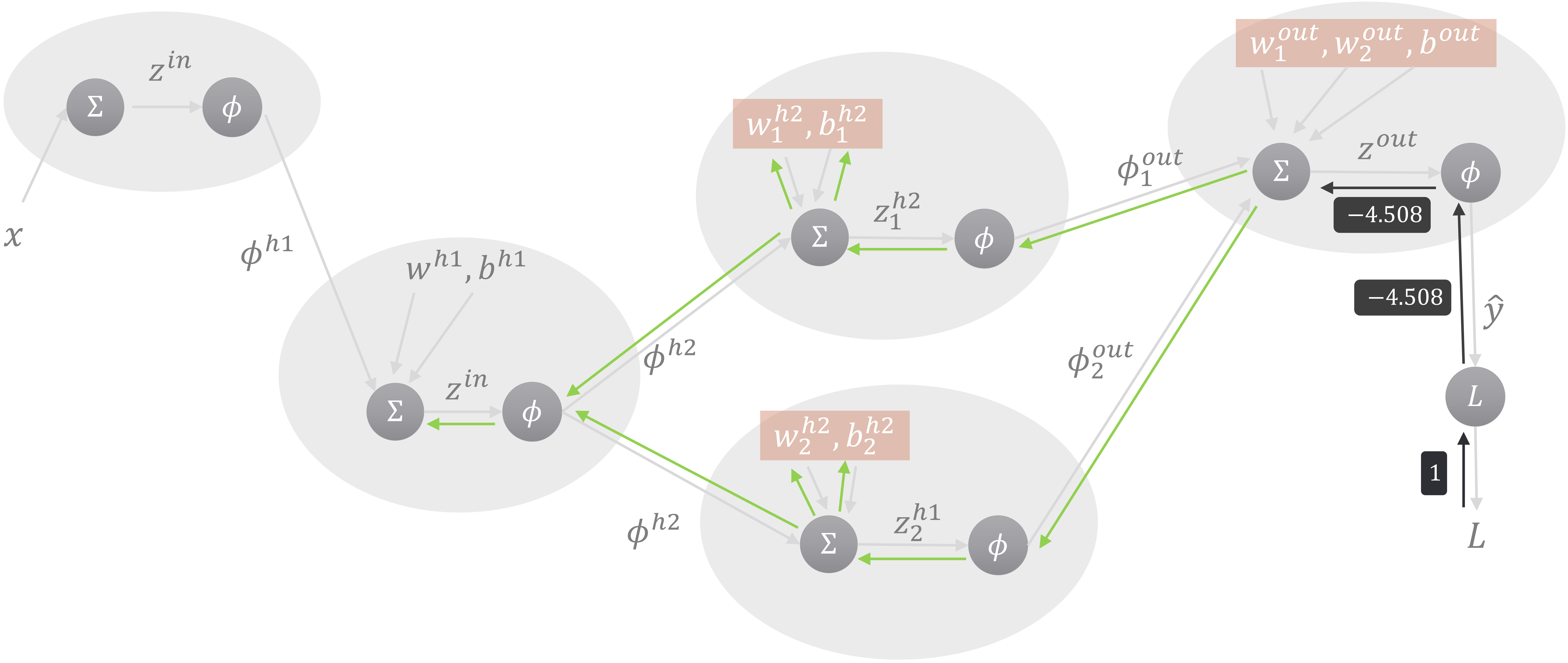
$w_1^{out}, w_2^{out}, b^{out}$
z^{in}
x
ϕ^{h1}
w^{h1}, b^{h1}
z^{in}
w_1^{h2}, b_1^{h2}
z_1^{h2}
ϕ^{h2}
ϕ^{h2}
w_2^{h2}, b_2^{h2}
z_2^{h1}
ϕ_1^{out}
ϕ_2^{out}
z^{out}
−4.508
−4.508
$\hat{y}$
L
1
L



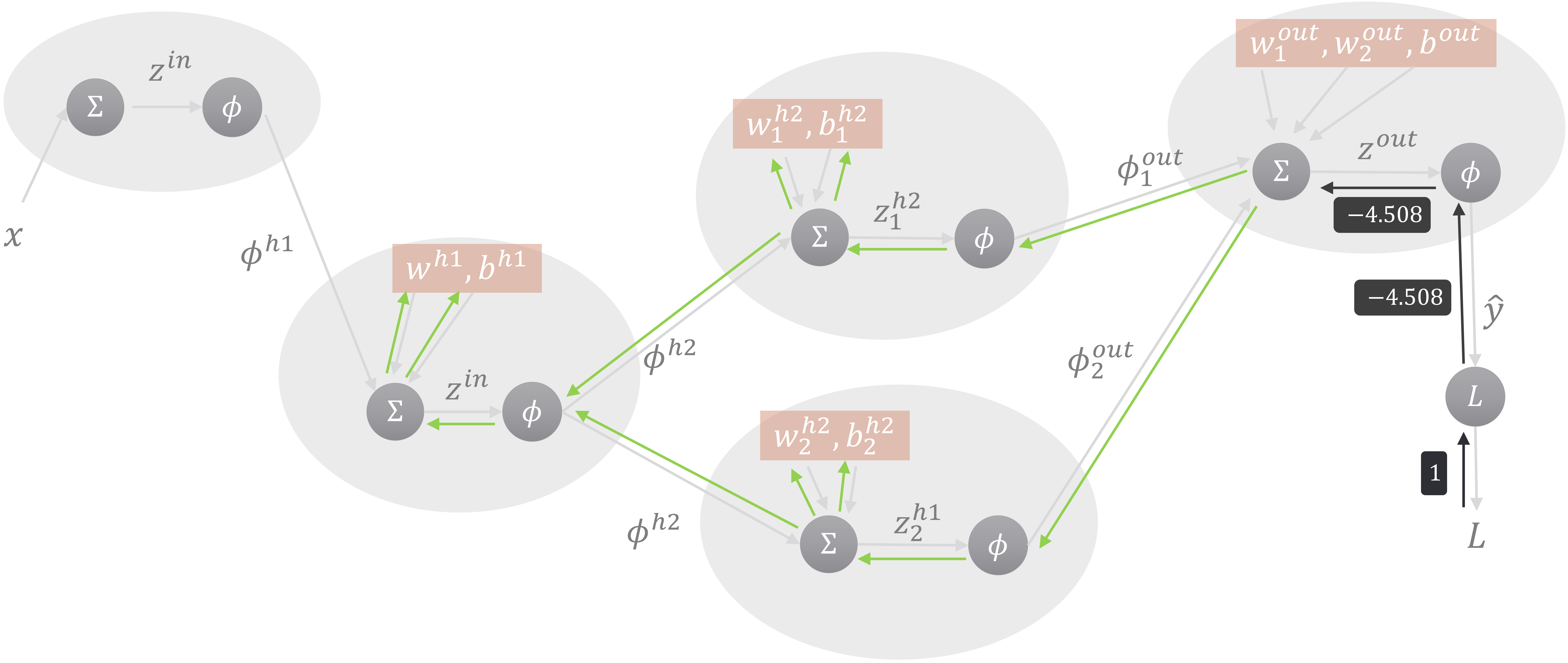
z^{in}
x
ϕ^{h1}
w^{h1}, b^{h1}
z^{in}
ϕ^{h2}
ϕ^{h2}
w_1^{h2}, b_1^{h2}
z_1^{h2}
w_2^{h2}, b_2^{h2}
z_2^{h1}
ϕ_1^{out}
ϕ_2^{out}
$w_1^{out}, w_2^{out}, b^{out}$
z^{out}
−4.508
−4.508
$\hat{y}$
1
L
L



z^{in}
Σ
ϕ
x
ϕ^{h1}
w^{h1}, b^{h1}
z^{in}
ϕ^{h2}
ϕ^{h2}
w_1^{h2}, b_1^{h2}
z_1^{h2}
w_2^{h2}, b_2^{h2}
z_2^{h1}
ϕ_1^{out}
ϕ_2^{out}
$w_1^{out}, w_2^{out}, b^{out}$
z^{out}
−4.508
−4.508
$\hat{y}$
L
1
L



z^{in}
Σ
ϕ
x
ϕ^{h1}
w^{h1}, b^{h1}
z^{in}
Σ
ϕ
ϕ^{h2}
ϕ^{h2}
w_1^{h2}, b_1^{h2}
z_1^{h2}
Σ
ϕ
w_2^{h2}, b_2^{h2}
z_2^{h1}
Σ
ϕ
ϕ_1^{out}
ϕ_2^{out}
$w_1^{out}, w_2^{out}, b^{out}$
z^{out}
Σ
ϕ
−4.508
−4.508
$\hat{y}$
L
1
L



z^{in}
x
ϕ^{h1}
w^{h1}, b^{h1}
z^{in}
ϕ^{h2}
ϕ^{h2}
w_1^{h2}, b_1^{h2}
z_1^{h2}
w_2^{h2}, b_2^{h2}
z_2^{h1}
ϕ_1^{out}
ϕ_2^{out}
$w_1^{out}, w_2^{out}, b^{out}$
z^{out}
−4.508
−4.508
$\hat{y}$
1
L
L

Updated Values

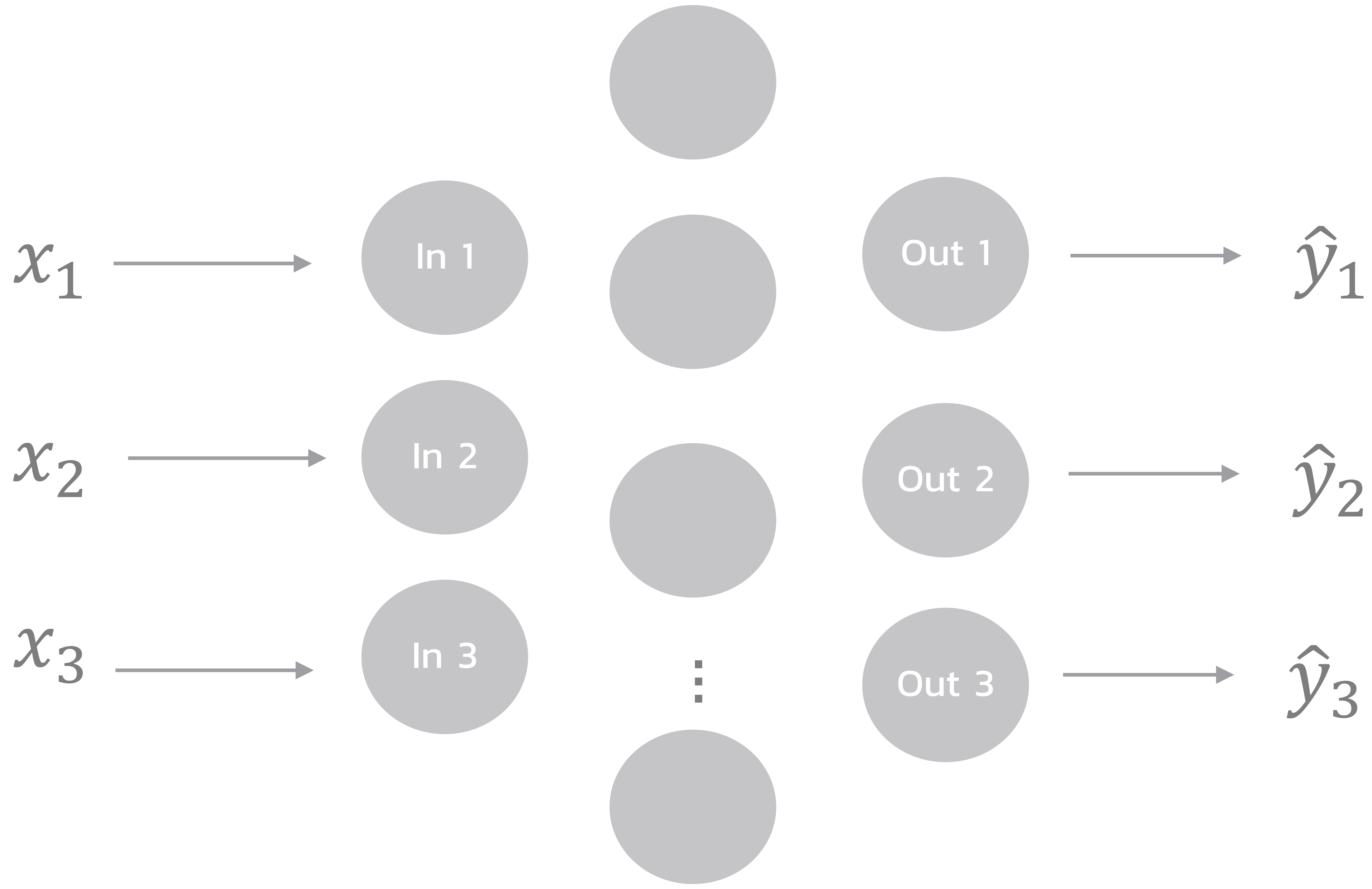
Variable	Init Val	Updated Val
w^{h1}	1	1.122
b^{h1}	0	0.228
w_1^{h2}	2	2.201
b_1^{h2}	0	0.275
w_2^{h2}	3	3.148
b_2^{h2}	0	0.203
w_1^{out}	4	4.366
w_2^{out}	5	5.405
b^{out}	0	0.451

Updated Prediction/Loss

Observed: $x = 1, y = 10$

Variable	Before	After
$\hat{y}$	7.745	9.798
L	5.082	0.352

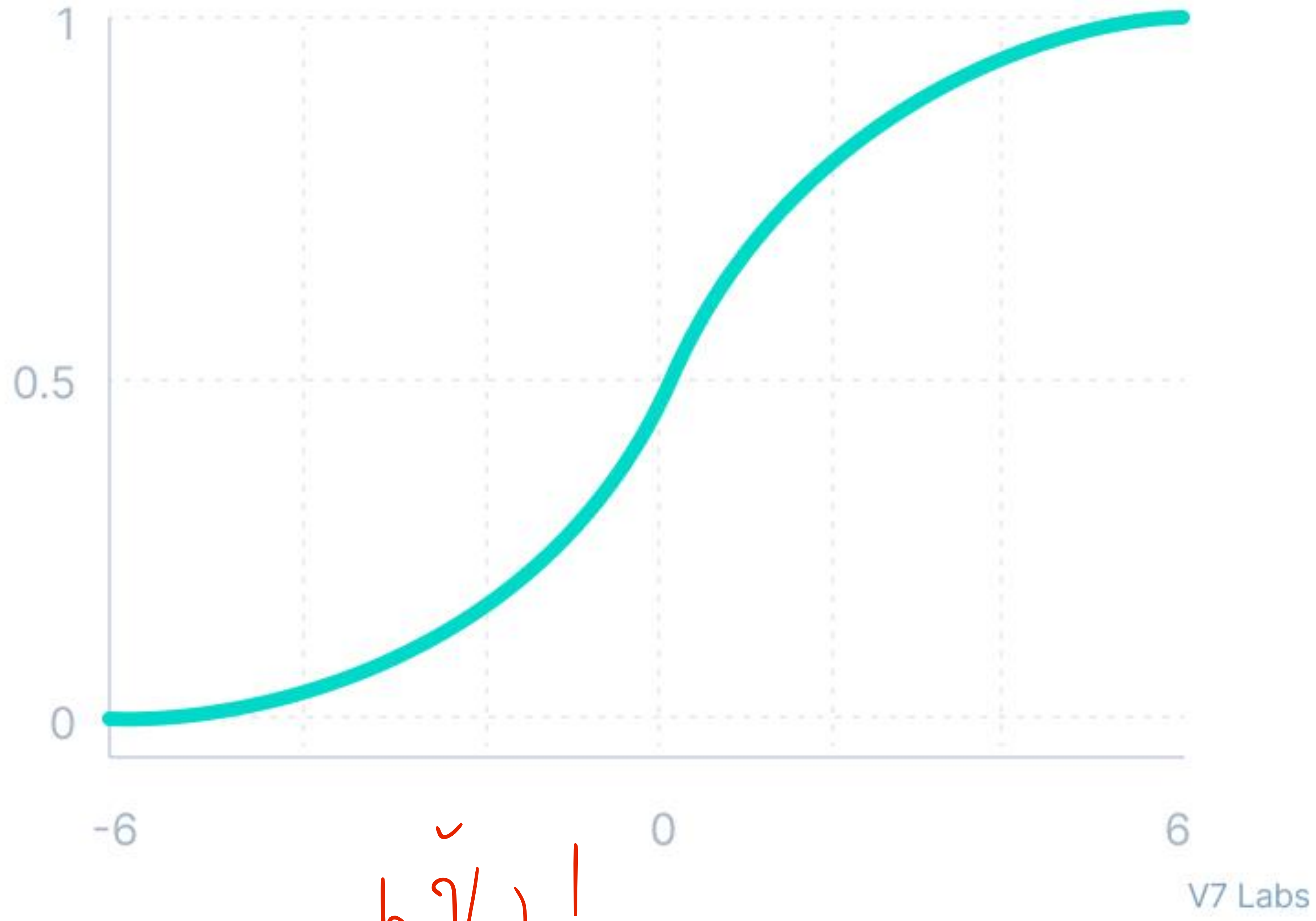
Multiple Inputs and Outputs



Activation Functions

Sigmoid / Logistic

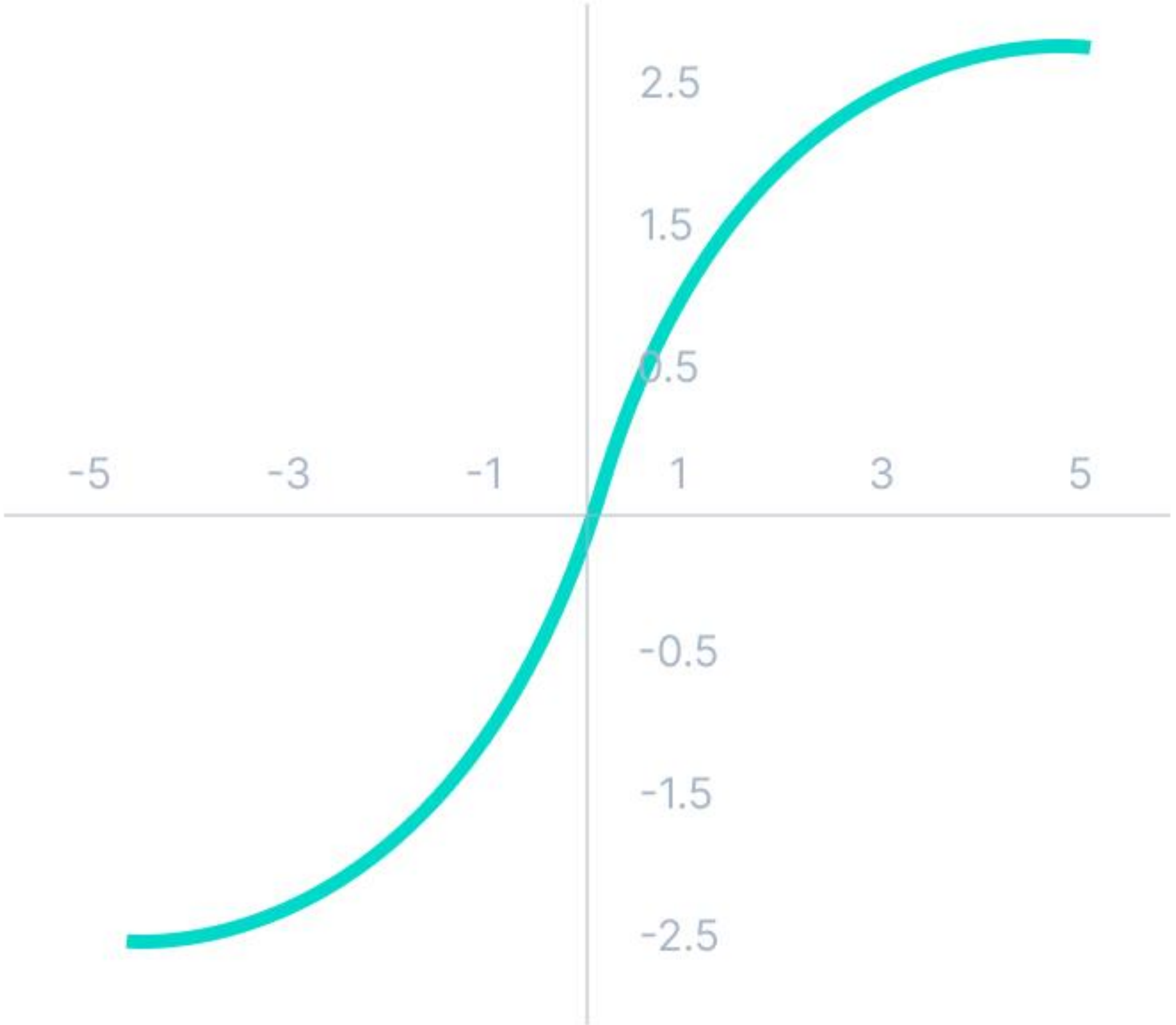
บวก!



เข้า!

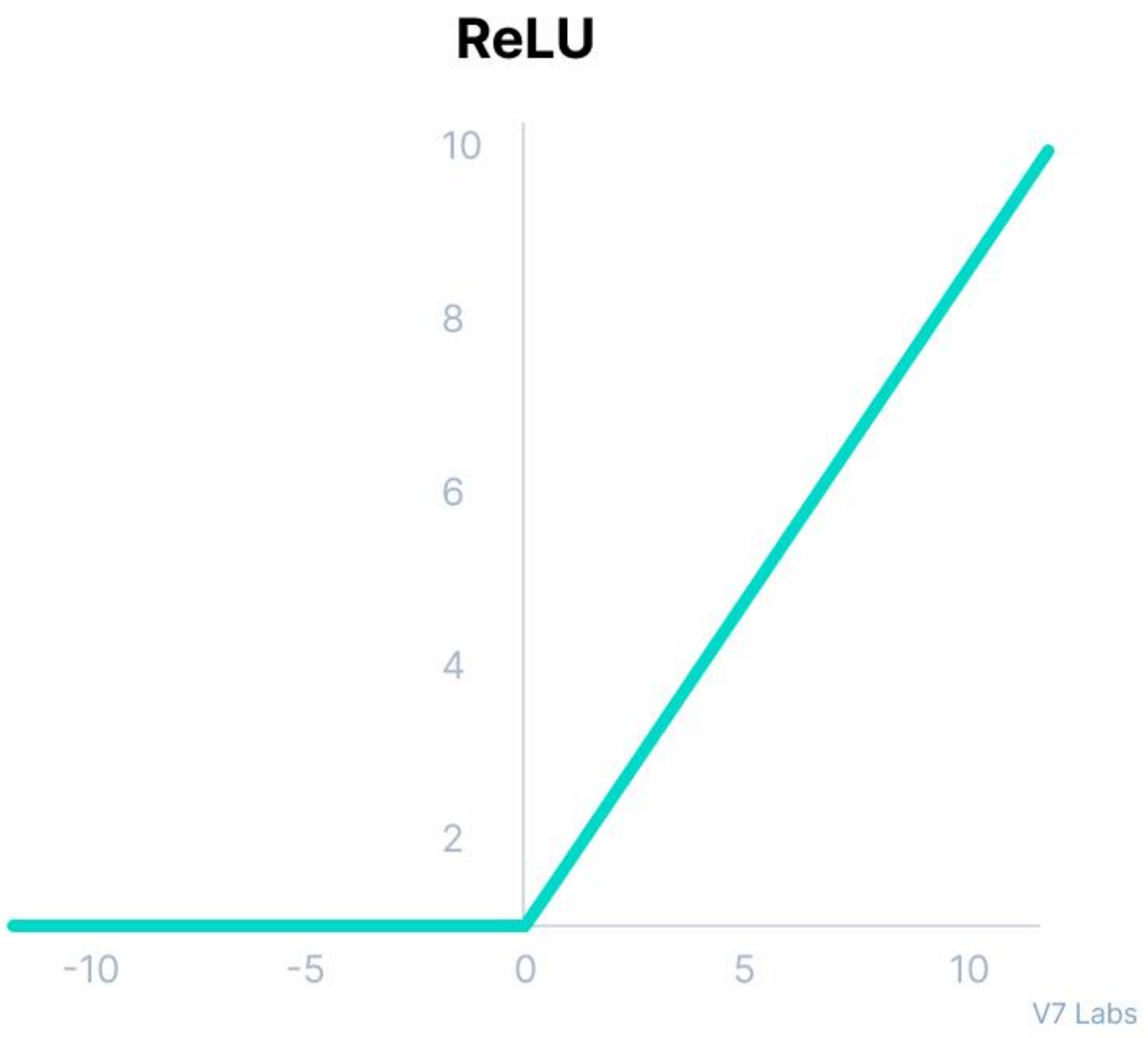
$$f(x) = \frac{1}{1 + e^{-x}}$$

Tanh

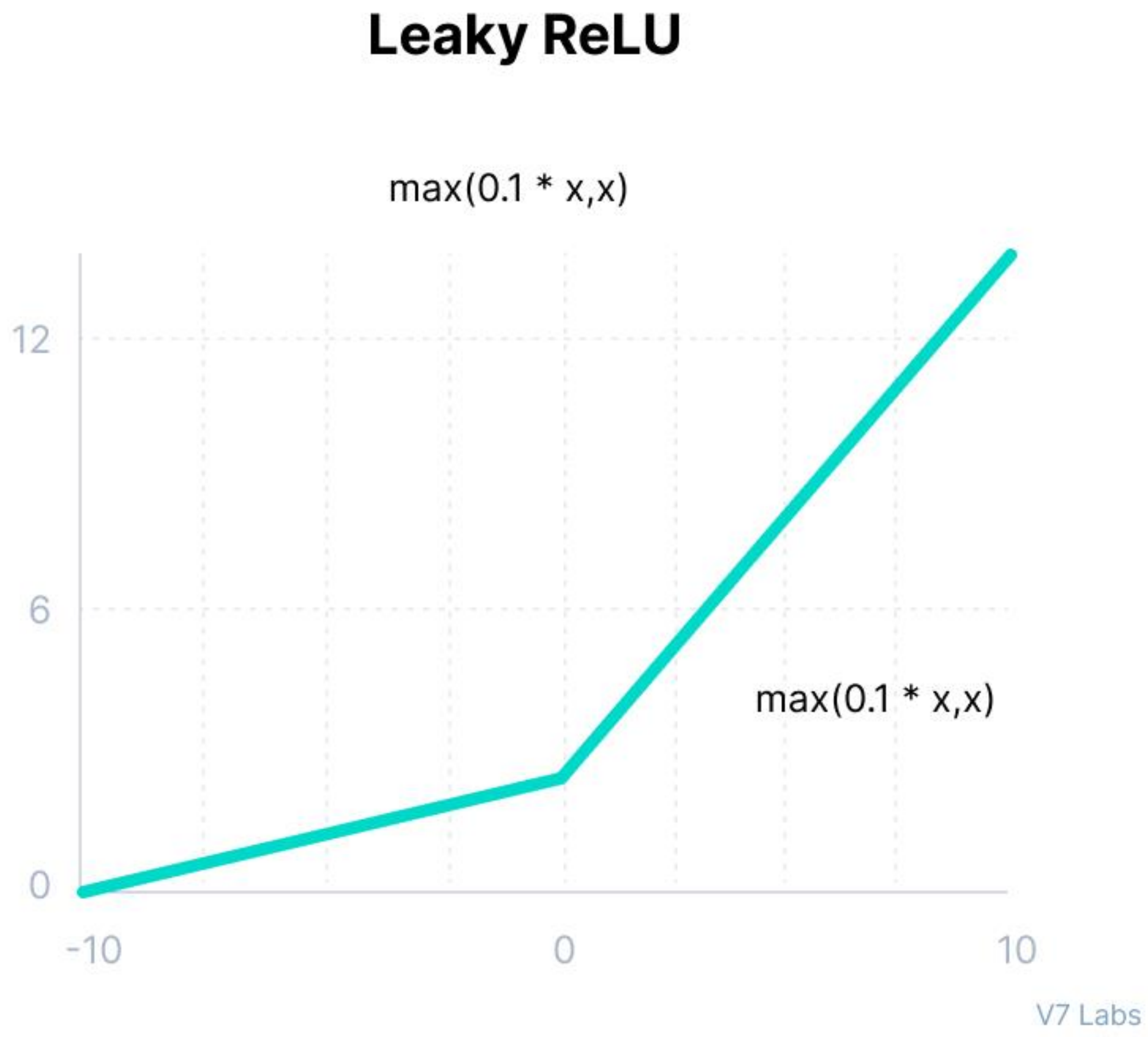


$$f(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})}$$

https://www.v7labs.com/blog/neural-networks-activation-functions



$$f(x) = max\ (0, x)$$



$$f(x) = max\ (0.1x, x)$$

https://www.v7labs.com/blog/neural-networks-activation-functions

Optimizer

- Stochastic gradient descent (SGD)
  - Estimate the actual gradient (calculated from the entire data set) by a value from subset of data (batch).
- Adaptive moment estimation (ADAM)
  - Stochastic gradient descent with adaptive learning rate optimization algorithm

Classification

- Output nodes equal to number of class.
  - One-hot encoding
- Use *softmax* function to calculate probability of each class.
  - $p_j = softmax(\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_C) = \frac{e^{\hat{y}_j}}{\Sigma_{k=1}^{C} e^{\hat{y}_k}}$
- Loss function
  - *Categorical cross entropy (CCE)*
  - $CCE = -\frac{1}{N} \Sigma_{i=1}^{N} \Sigma_{k=1}^{C} \chi_{y_k \in C_k} \ln(p_k)$

CCE Example

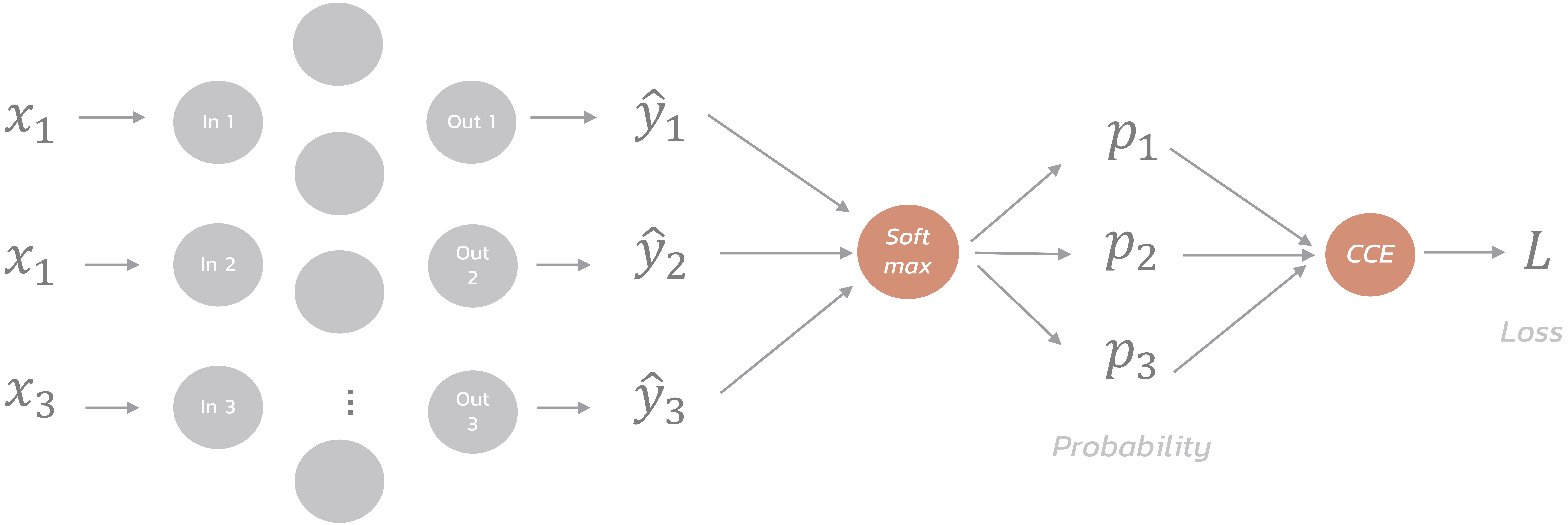
$y_1 = [0, 1, 0]$

$p_1 = [0.05, 0.95, 0]$

$y_2 = [0, 0, 1]$

$p_2 = [0.1, 0.8, 0.1]$

$$CCE = -\frac{1}{2}(\ln 0.95 + \ln 0.1) = 1.177$$


x_1
x_1
x_3
In 1
In 2
In 3
Out 1
Out 2
Out 3
$\hat{y}_1$
$\hat{y}_2$
$\hat{y}_3$
Soft max
p_1
p_2
p_3
Probability
CCE
L
Loss

Loss function

Loss function	Usage	Examples: Using probabilities (from_logits=False)	Examples: Using logits (from_logits=True)
BinaryCrossentropy	Binary classification	y_true: 1 y_pred: 0.69	y_true: 1 y_pred: 0.8
CategoricalCrossentropy	Multiclass classification	y_true: 0 0 1 y_pred: 0.30 0.15 0.55	y_true: 0 0 1 y_pred: 1.5 0.8 2.1
Sparse CategoricalCrossentropy	Multiclass classification	y_true: 2 y_pred: 0.30 0.15 0.55	y_true: 2 y_pred: 1.5 0.8 2.1

We will use this one

Regression

- Output layer
  - No *“softmax”*
- Loss

  Mean squared error
  - Mean absolute (percentage) error
- *Scale both X and y data*
  - Scaling X: more stable model (small weights)
  - Scaling y: matching output of activation function / smaller gradient