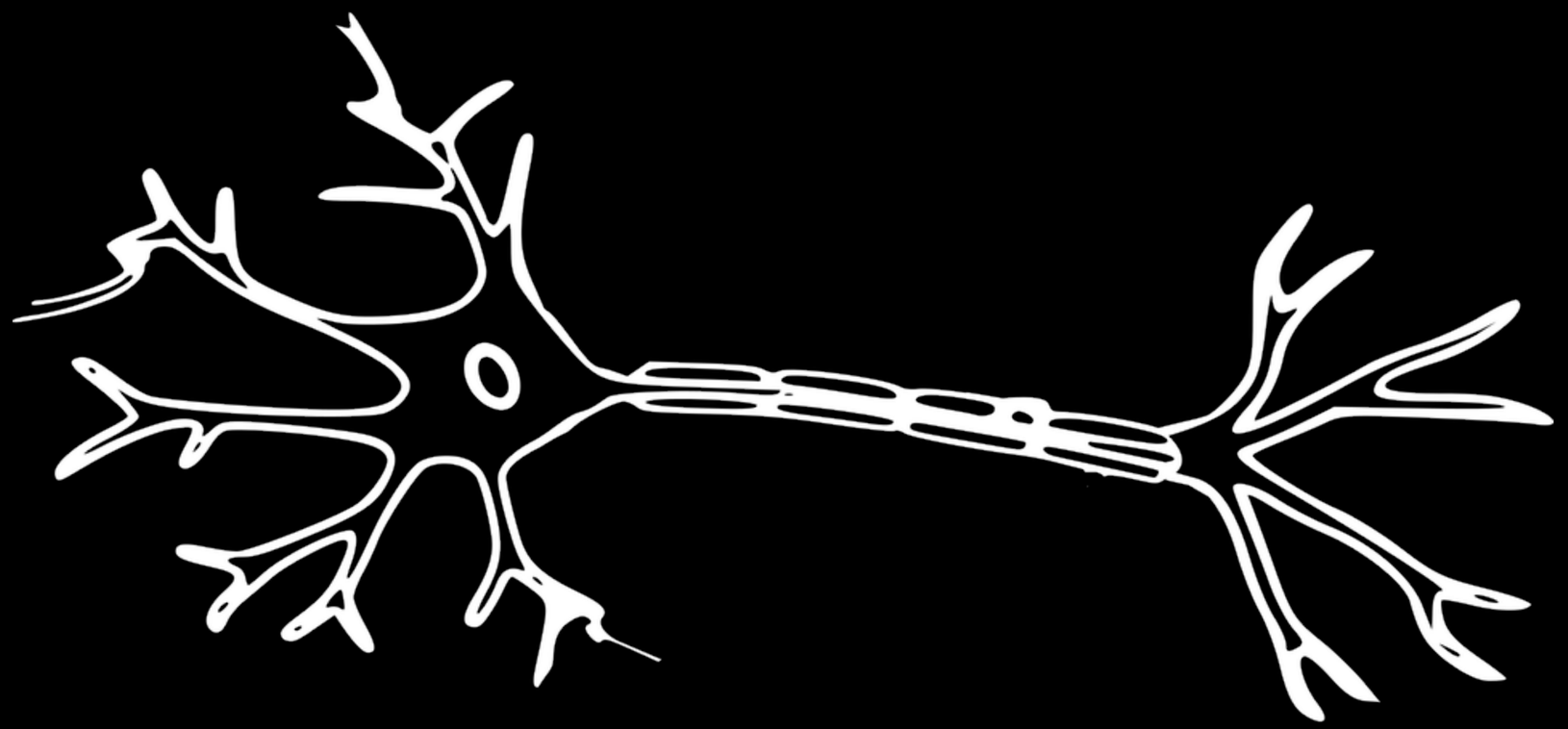
Introduction to **Artificial Intelligence** with Python

Neural Networks



Neural Networks

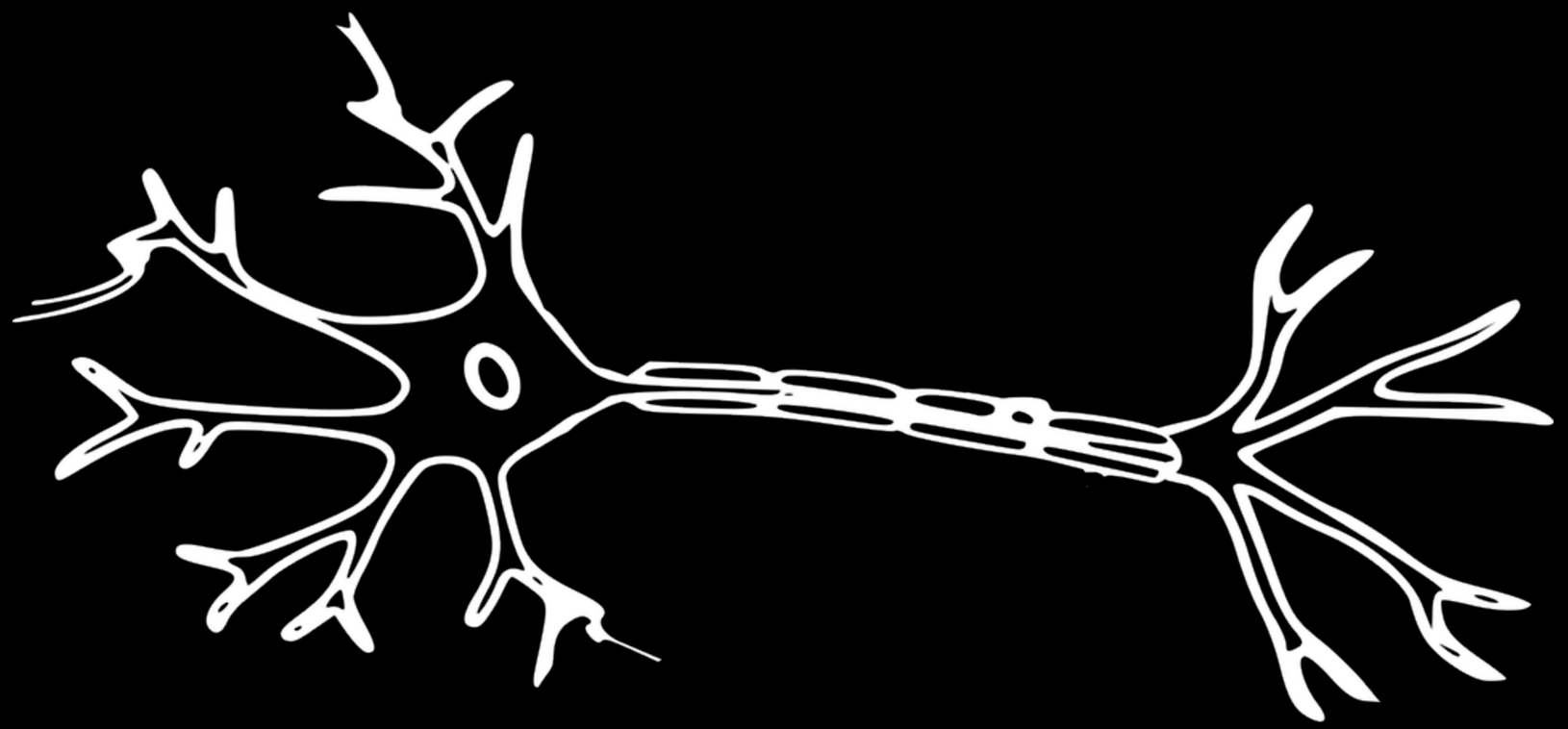
- Neurons are connected to and receive electrical signals from other neurons.
- Neurons process input signals and can be activated.

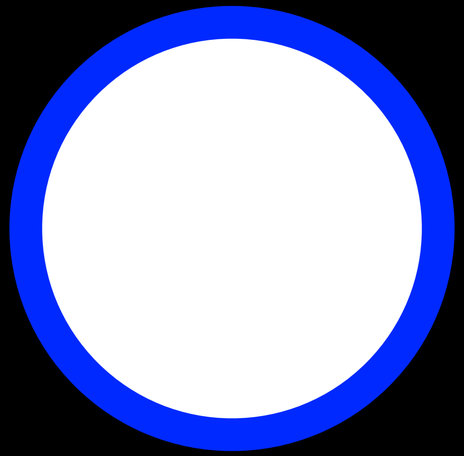
artificial neural network

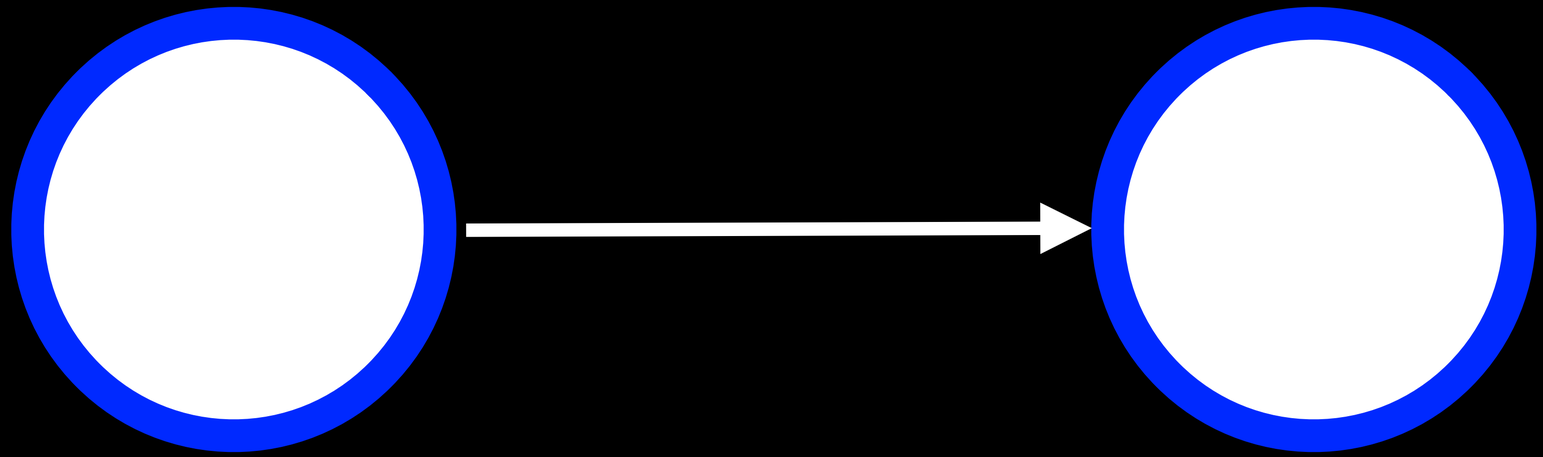
mathematical model for learning inspired by biological neural networks

Artificial Neural Networks

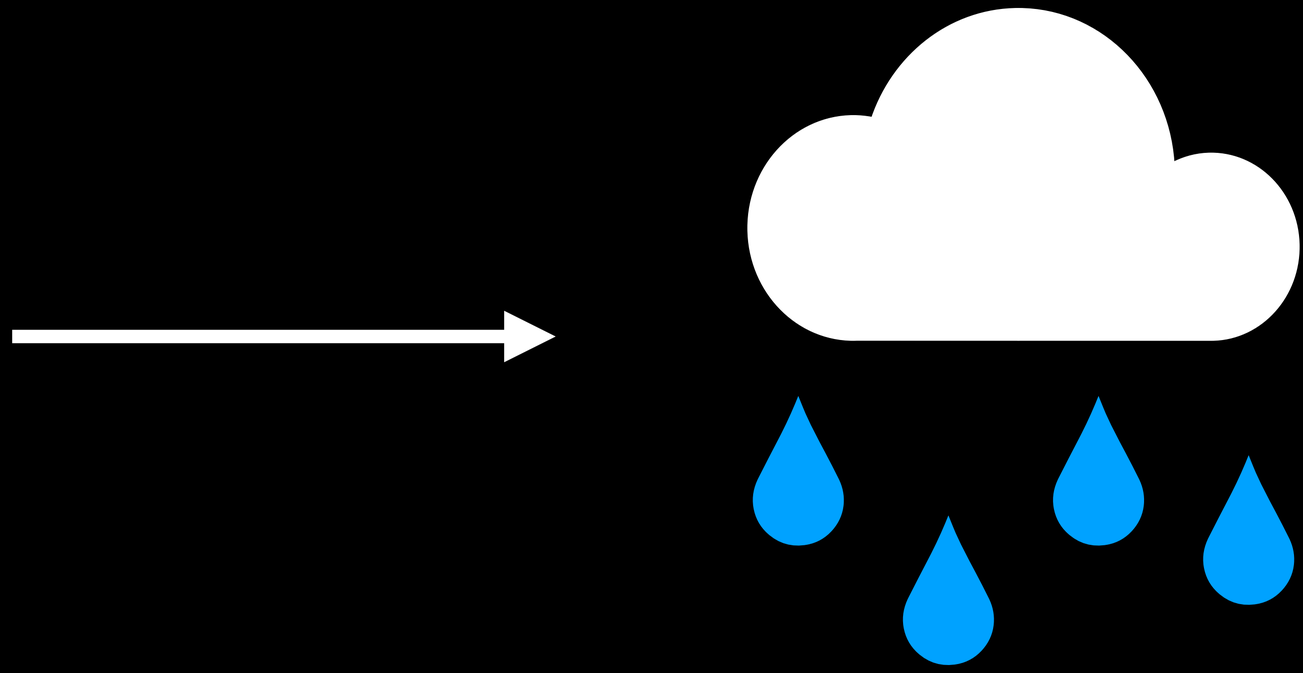
- Model mathematical function from inputs to outputs based on the structure and parameters of the network.
- Allows for learning the network's parameters based on data.







(x_1, x_2)



$$h(x_1, x_2) = w_0 + w_1 x_1 + w_2 x_2$$

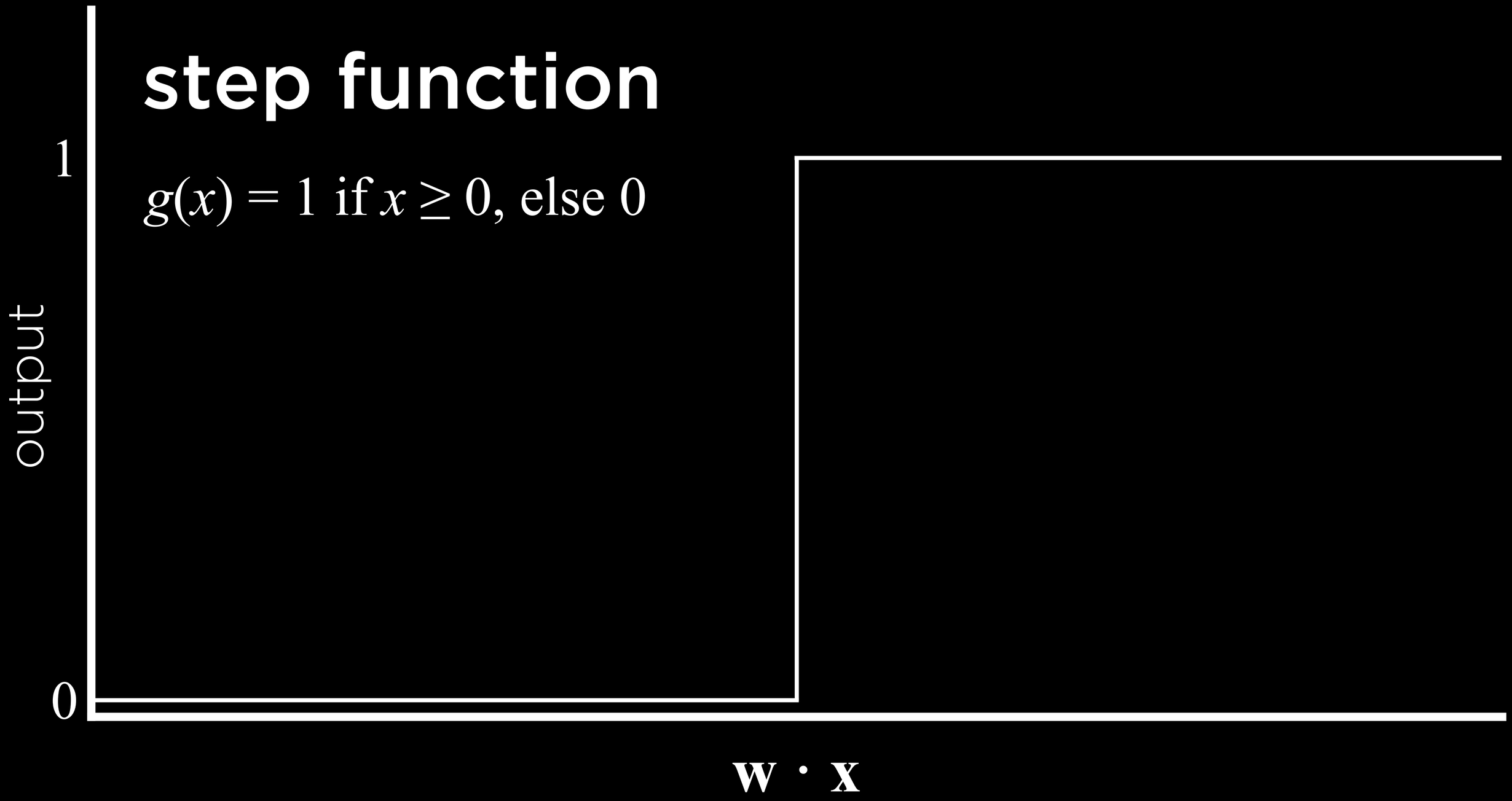
sometimes called "bias"

$$h(x_1, x_2) = \quad w_0 + w_1 x_1 + w_2 x_2$$

$$h(x_1, x_2) = w_0 + w_1 x_1 + w_2 x_2$$

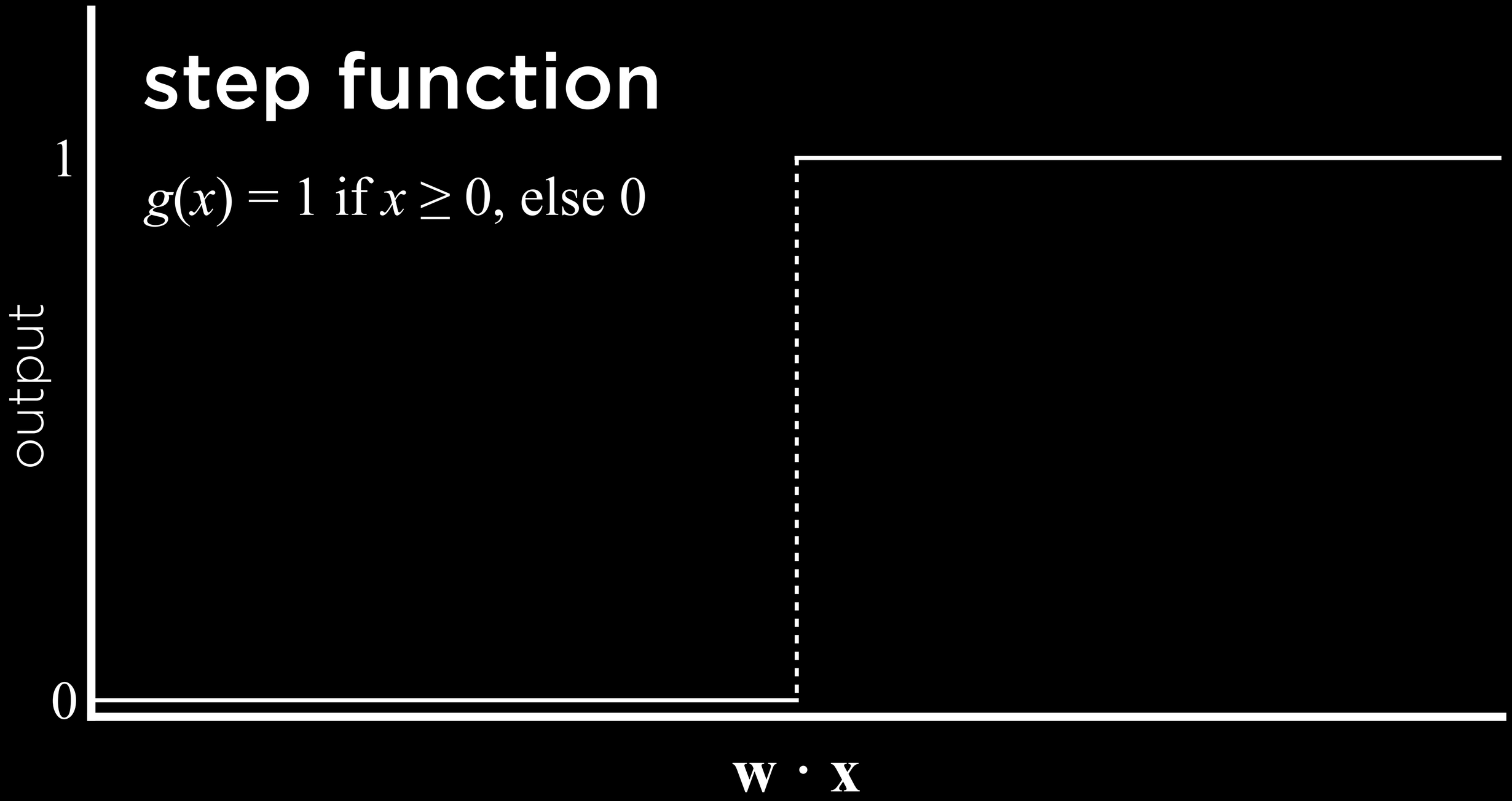
step function

$g(x) = 1$ if $x \geq 0$, else 0



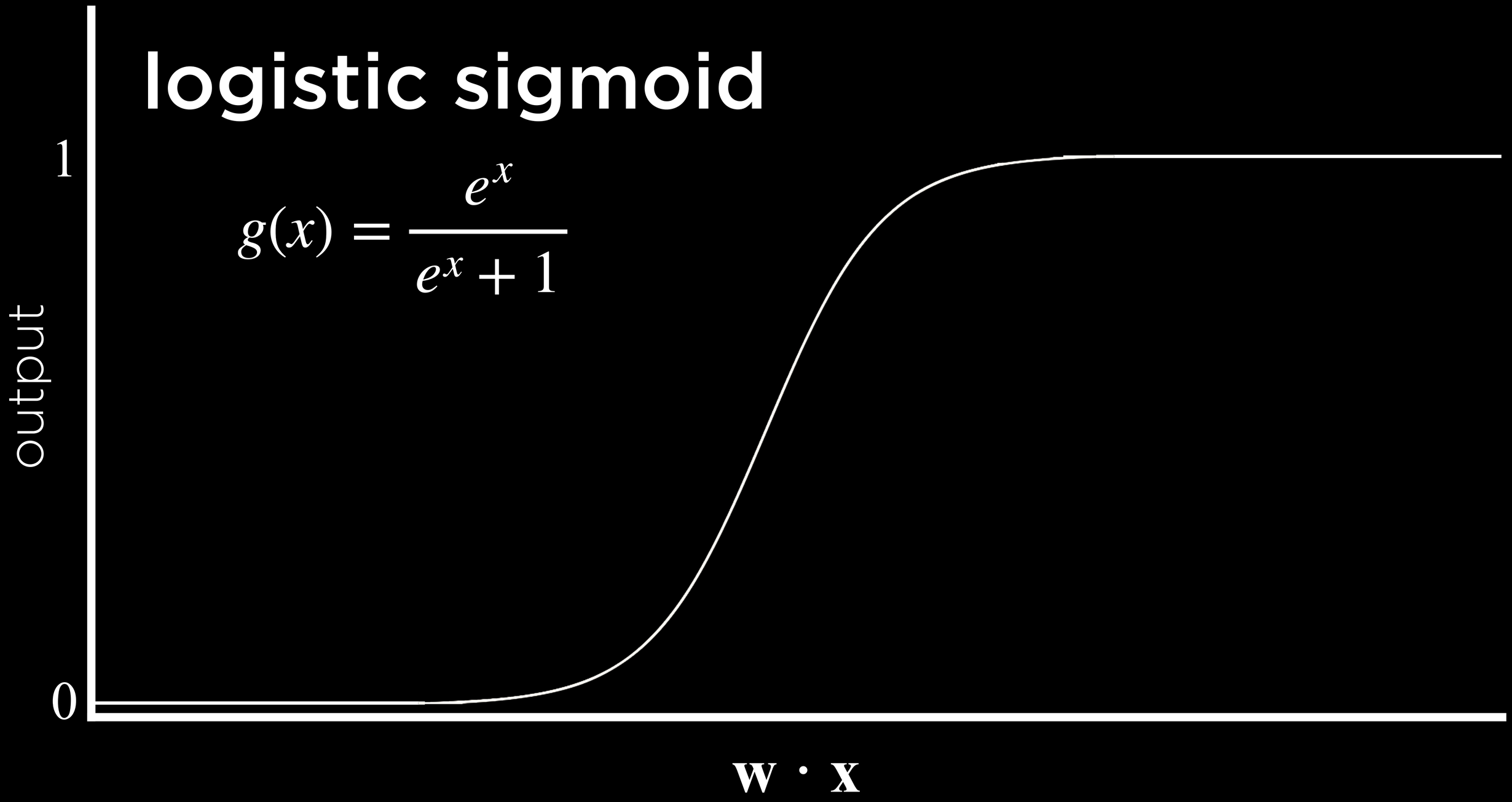
step function

$g(x) = 1$ if $x \geq 0$, else 0



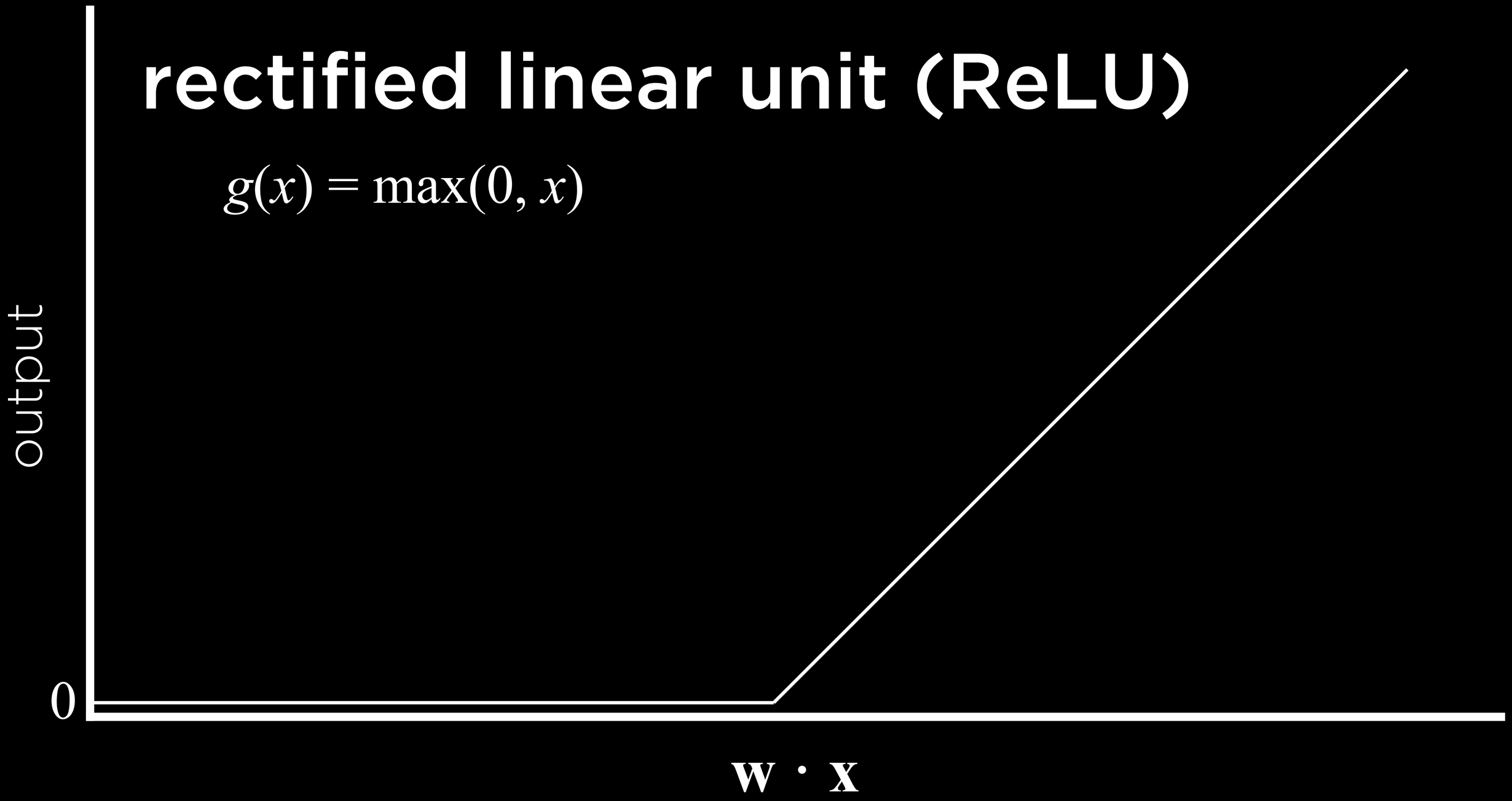
logistic sigmoid

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



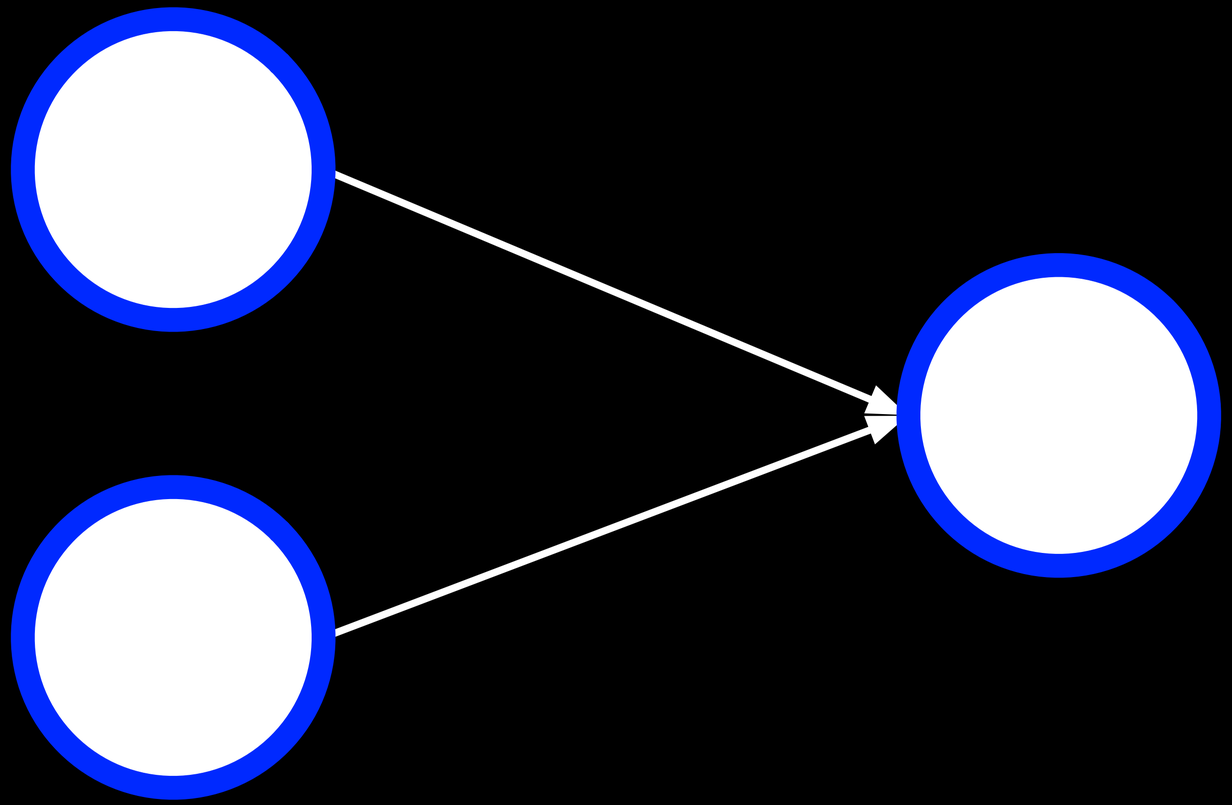
rectified linear unit (ReLU)

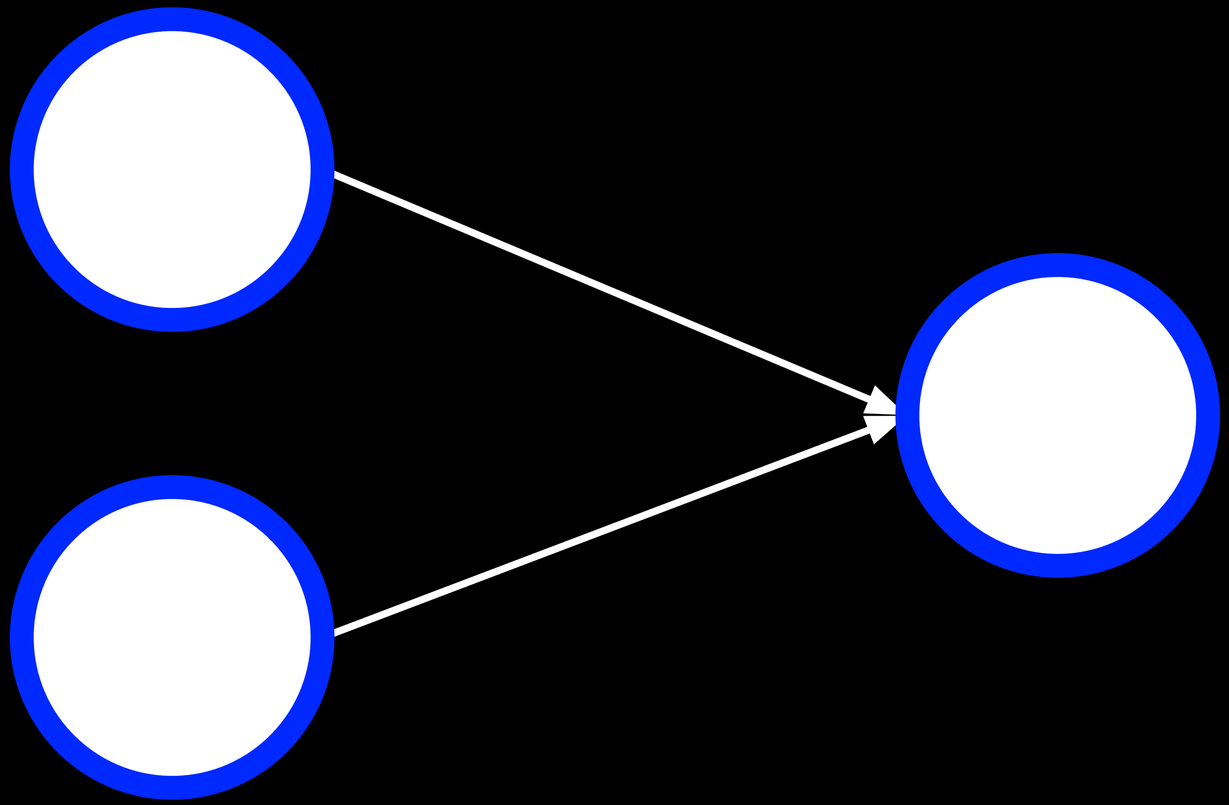
$$g(x) = \max(0, x)$$

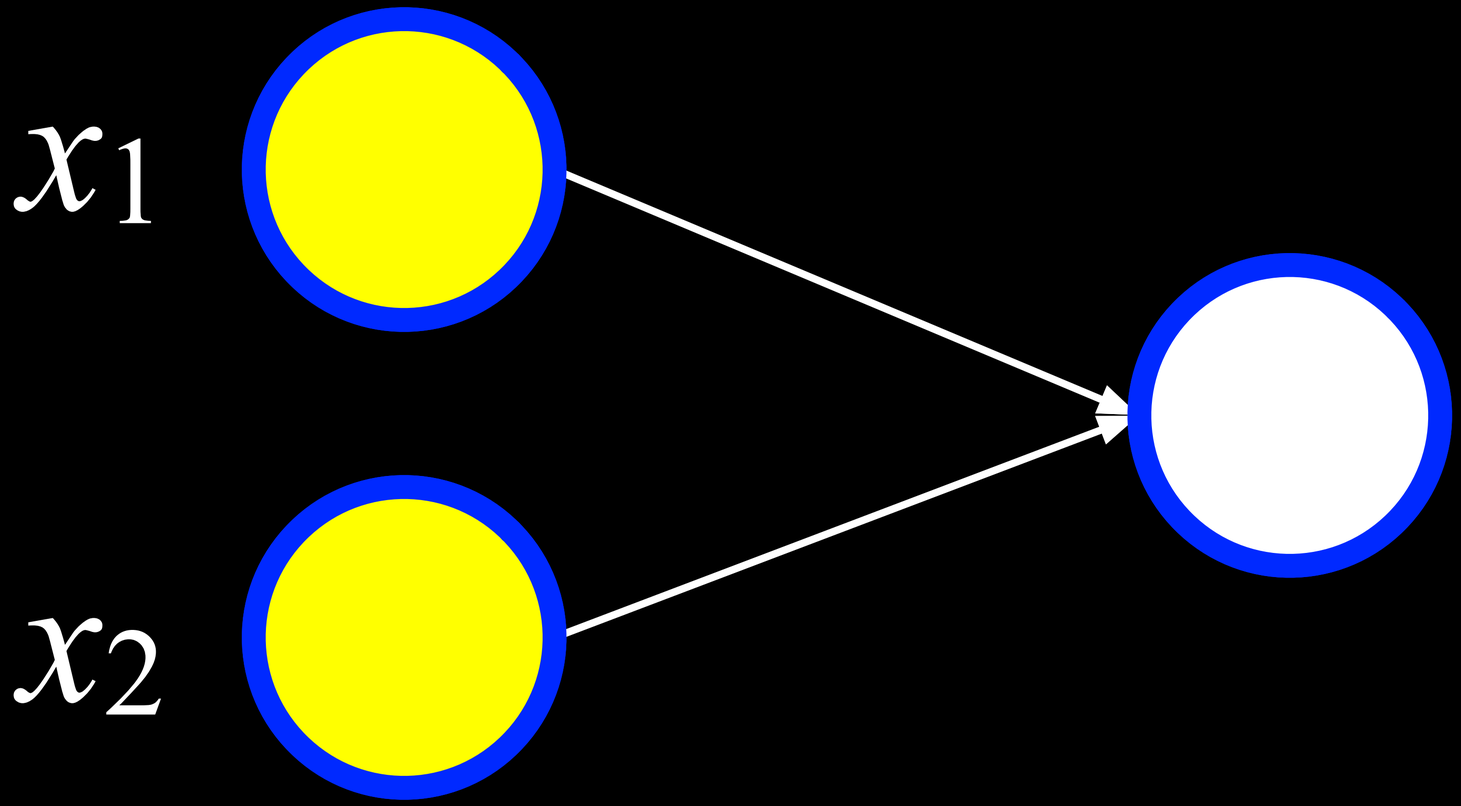


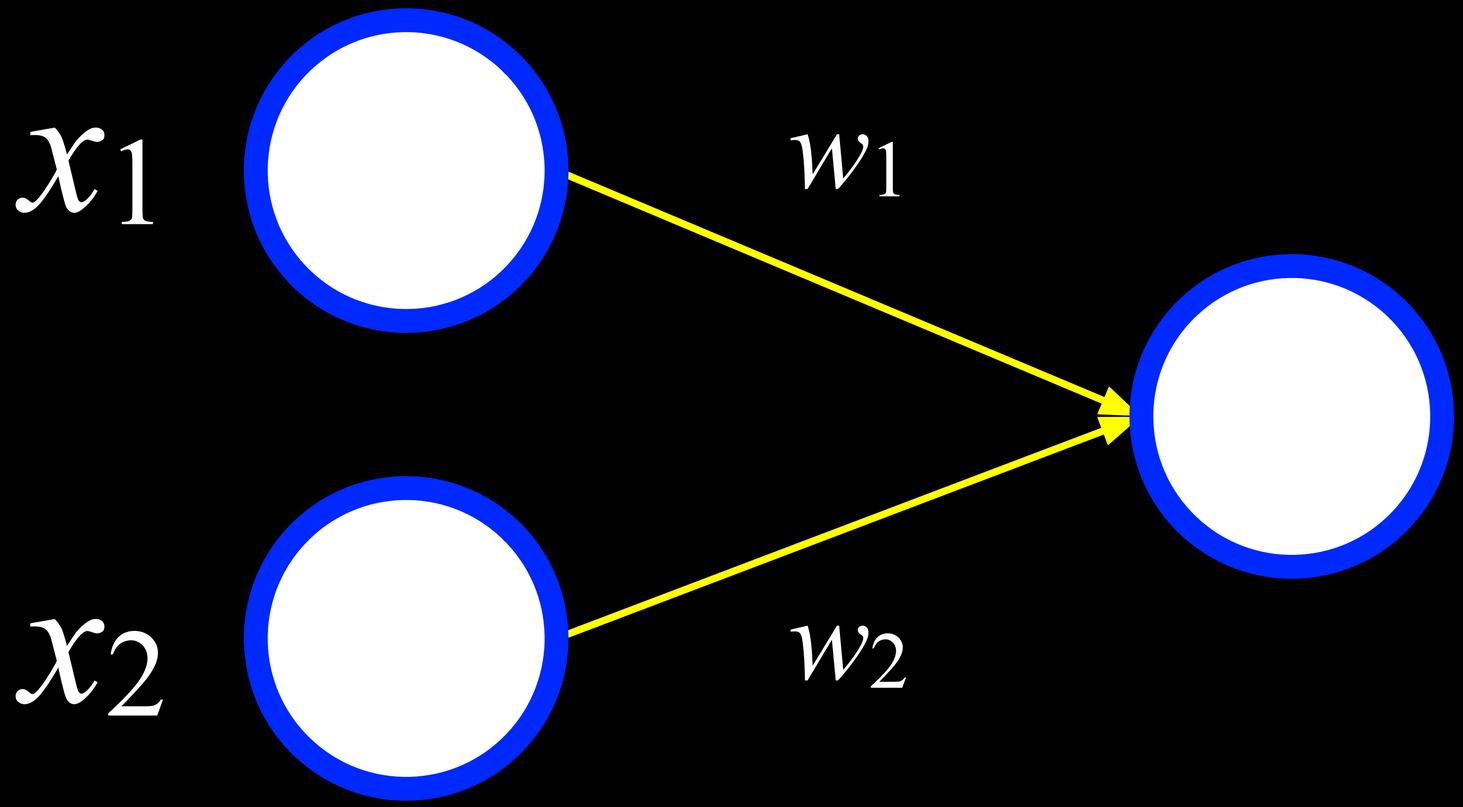
$$h(x_1, x_2) = \quad w_0 + w_1 x_1 + w_2 x_2$$

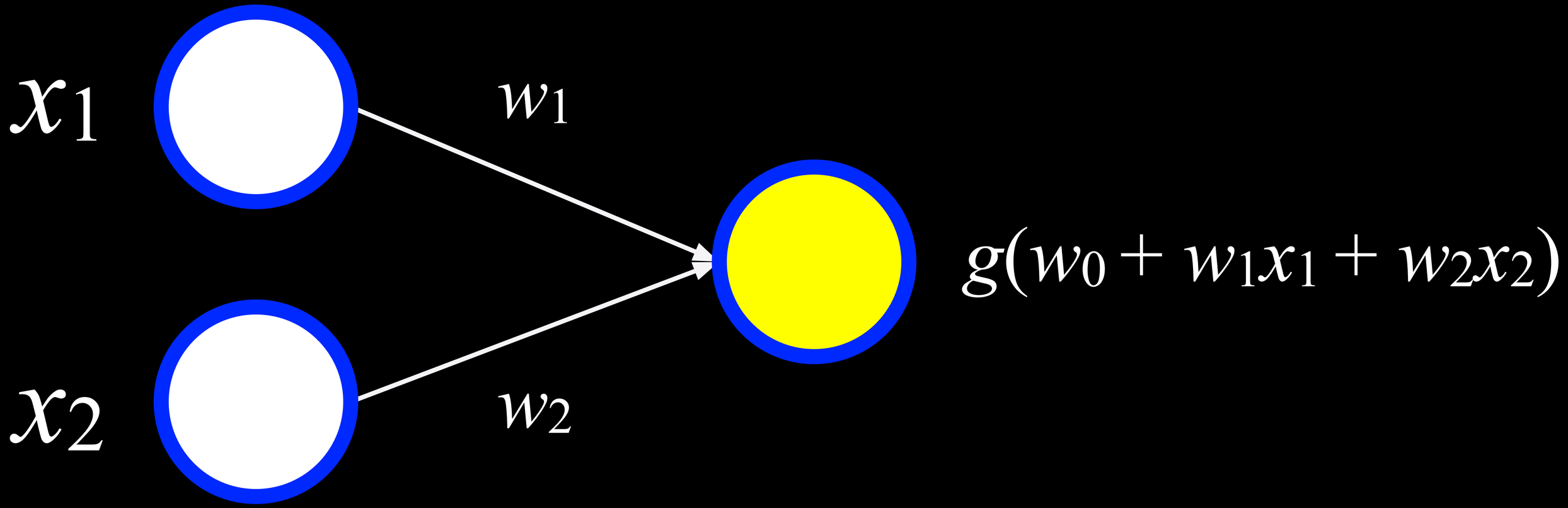
$$h(x_1, x_2) = g(w_0 + w_1 x_1 + w_2 x_2)$$

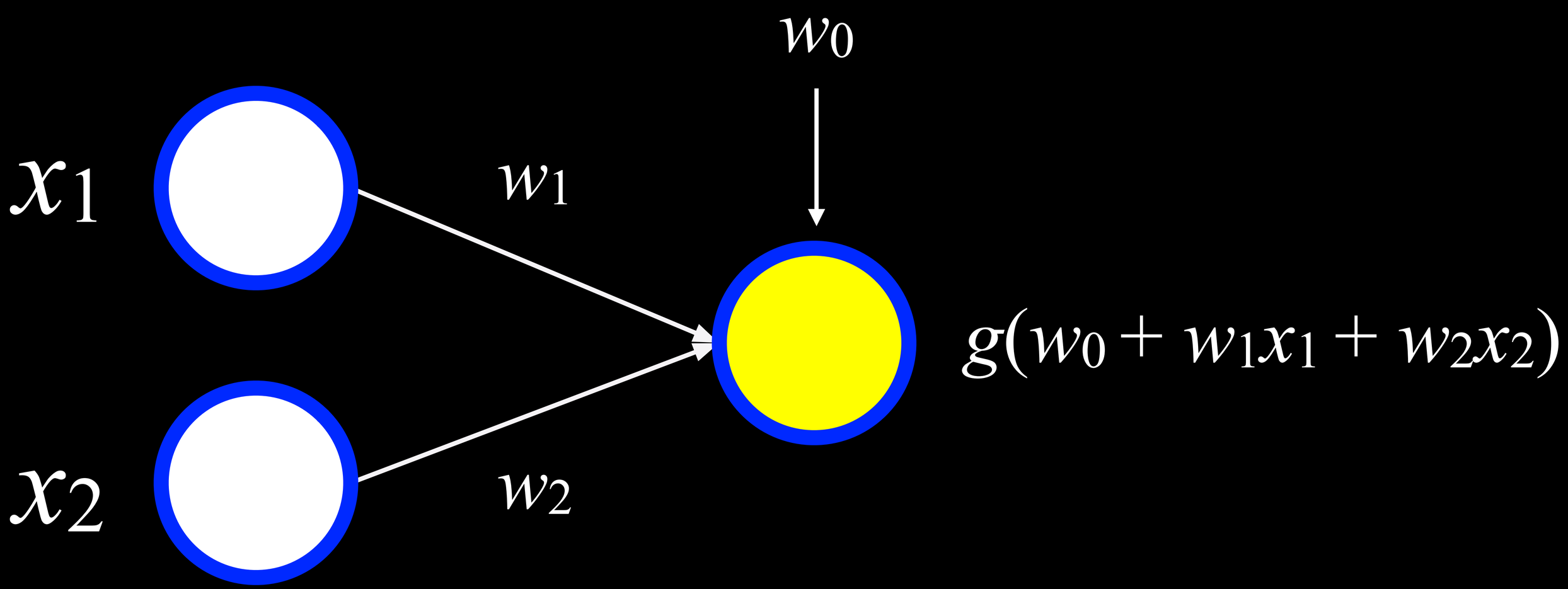


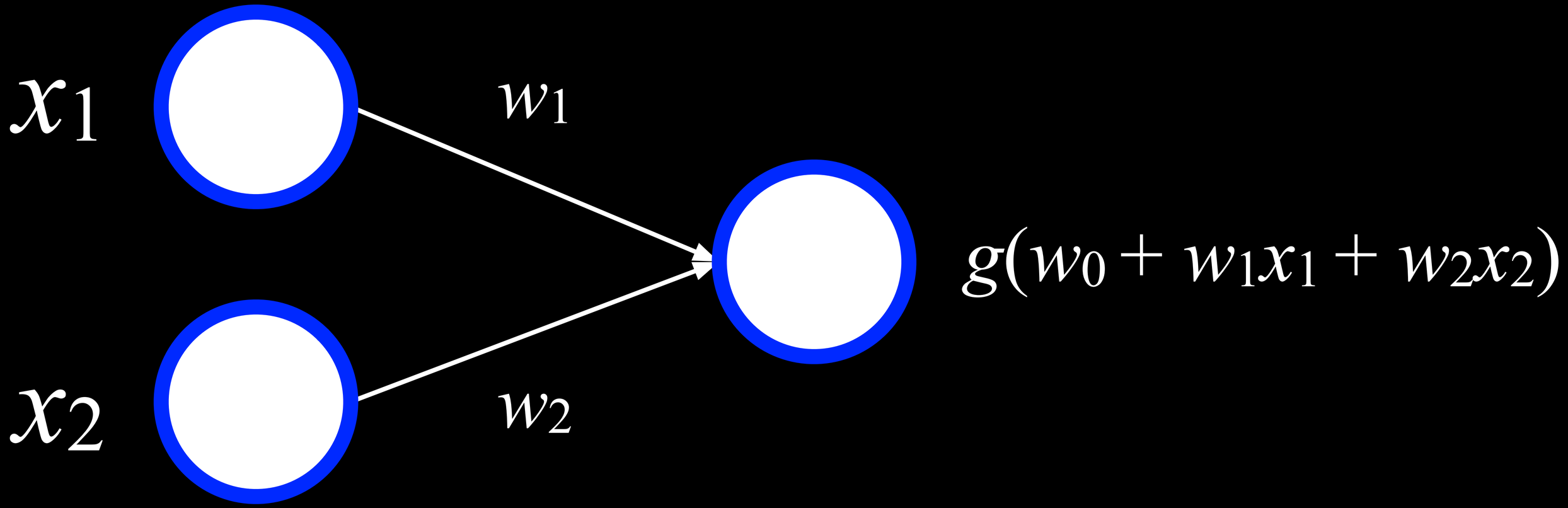



x_1
x_2


x_1
w_1
x_2
w_2

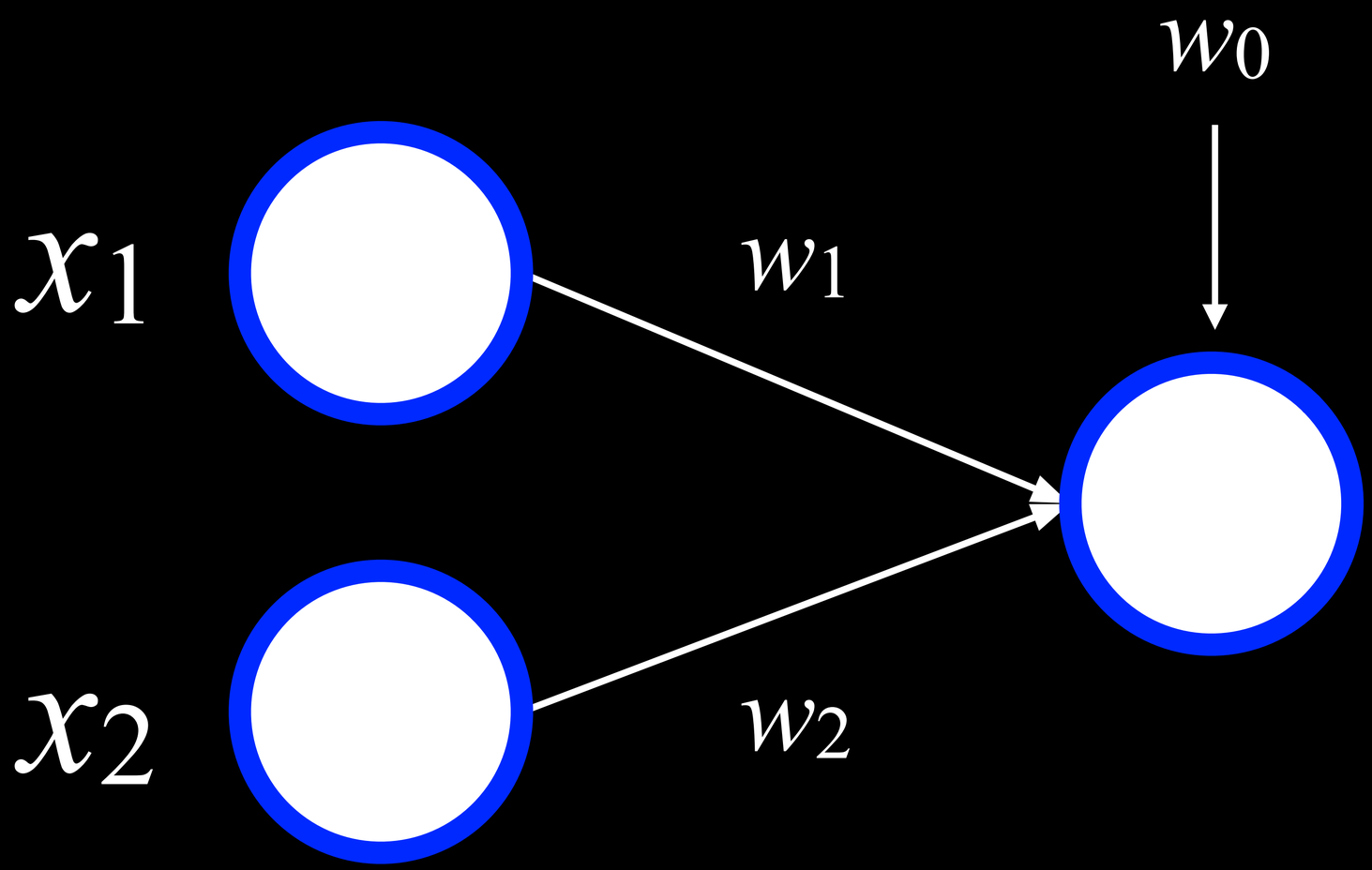

x_1
x_2
w_1
w_2
$g(w_0 + w_1x_1 + w_2x_2)$


w_0
x_1
w_1
x_2
w_2
$g(w_0 + w_1x_1 + w_2x_2)$

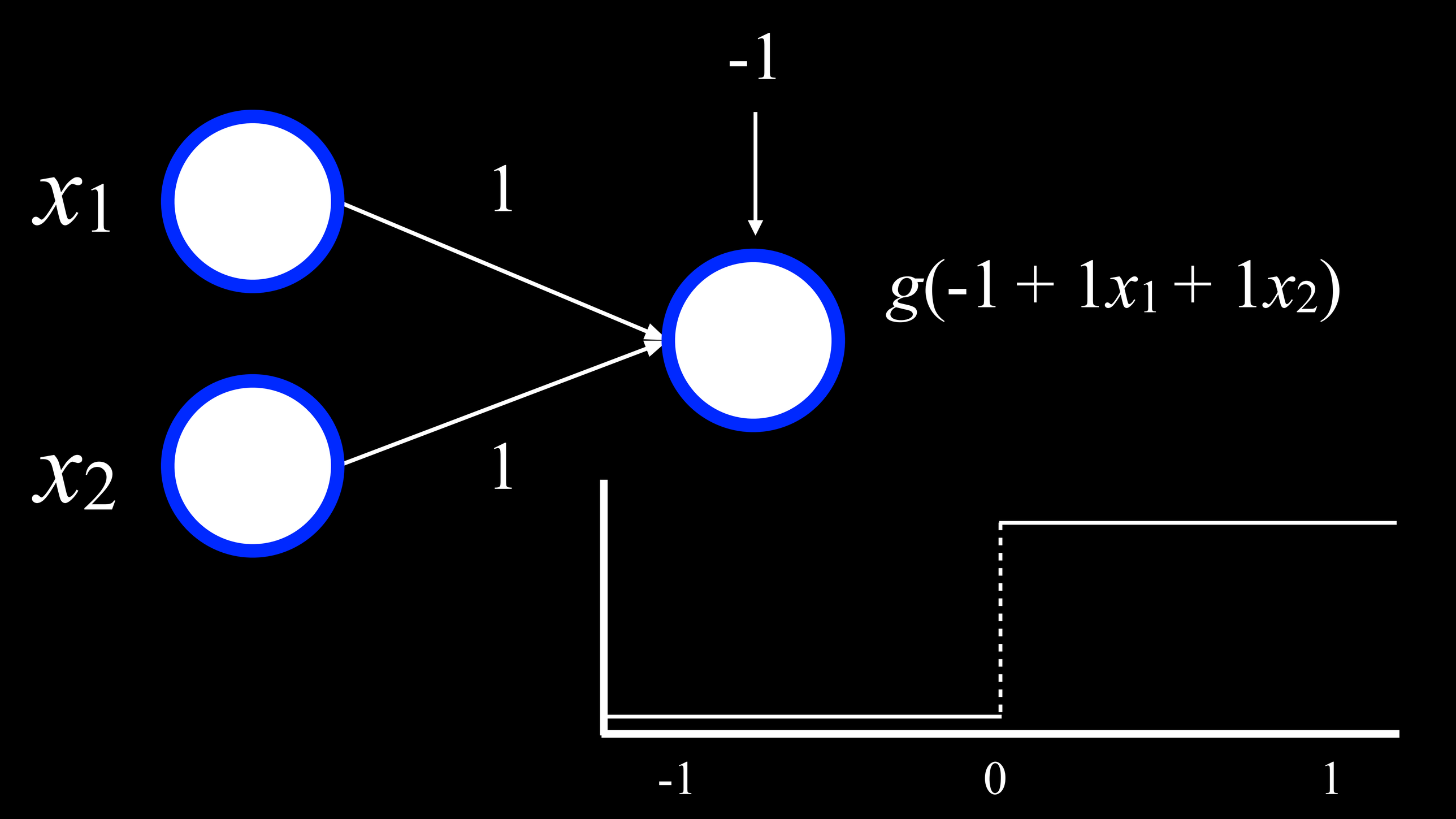

x_1
x_2
w_1
w_2
$g(w_0 + w_1x_1 + w_2x_2)$

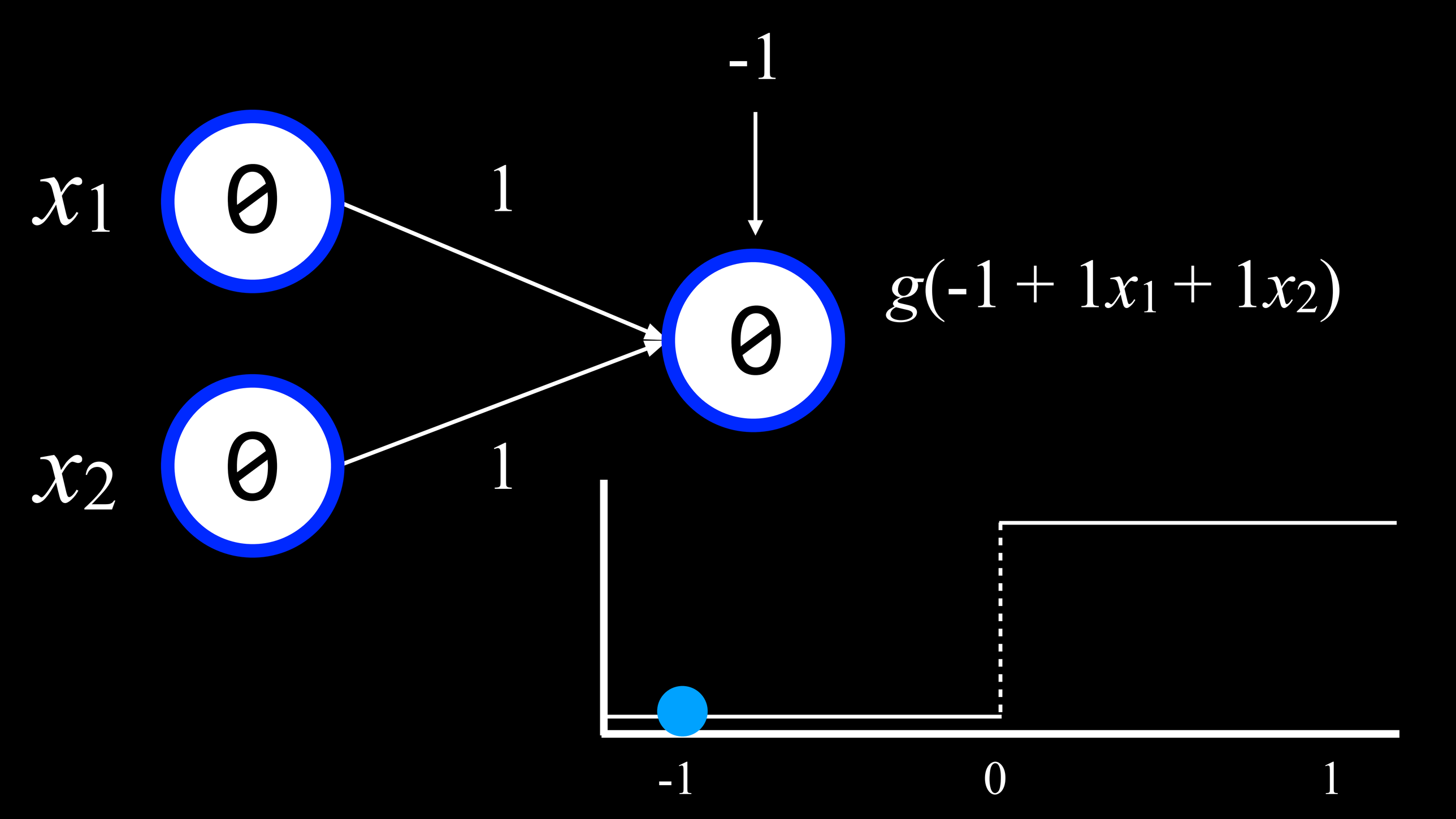
Or

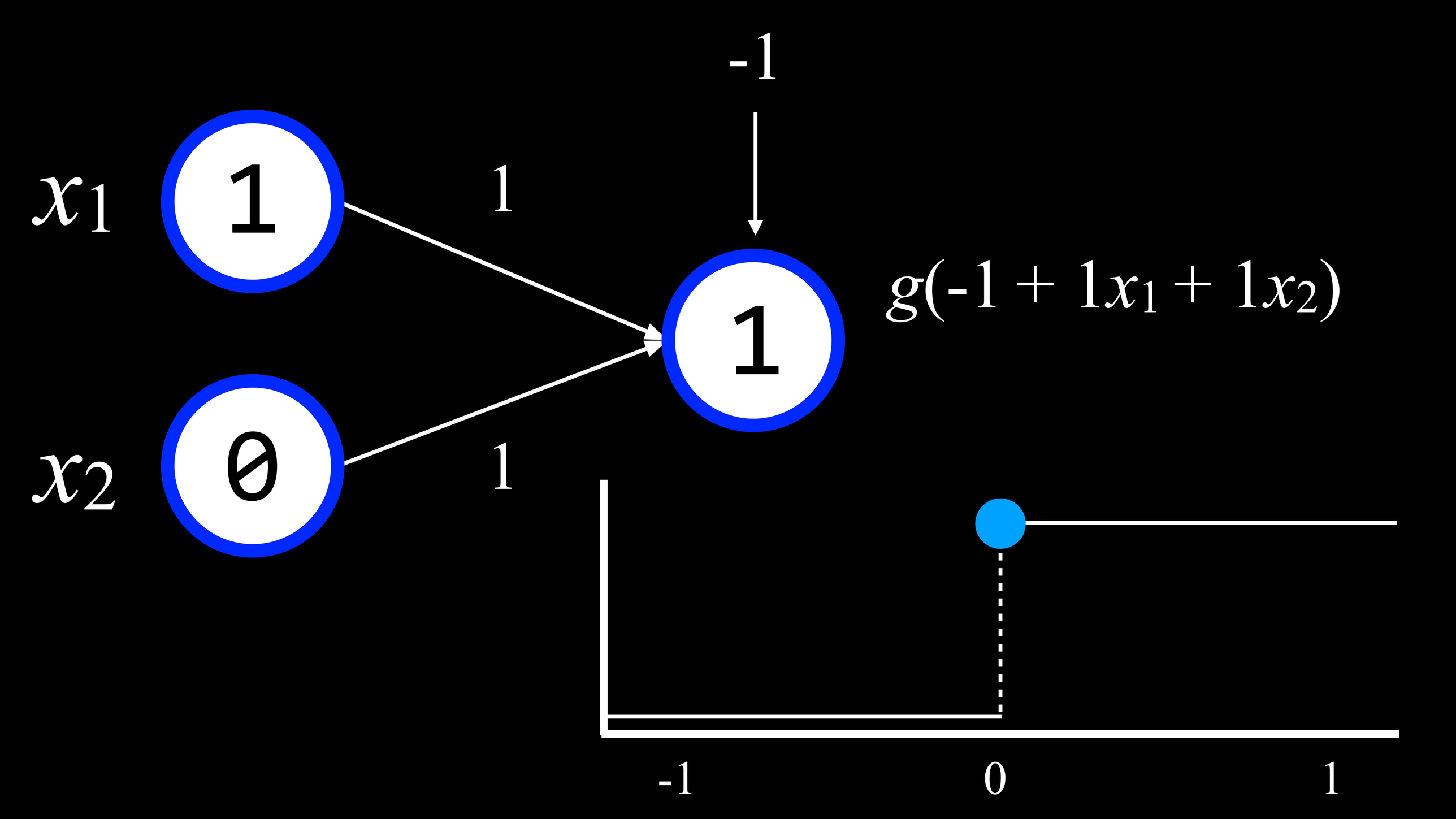
x	y	$f(x, y)$
0	0	0
0	1	1
1	0	1
1	1	1

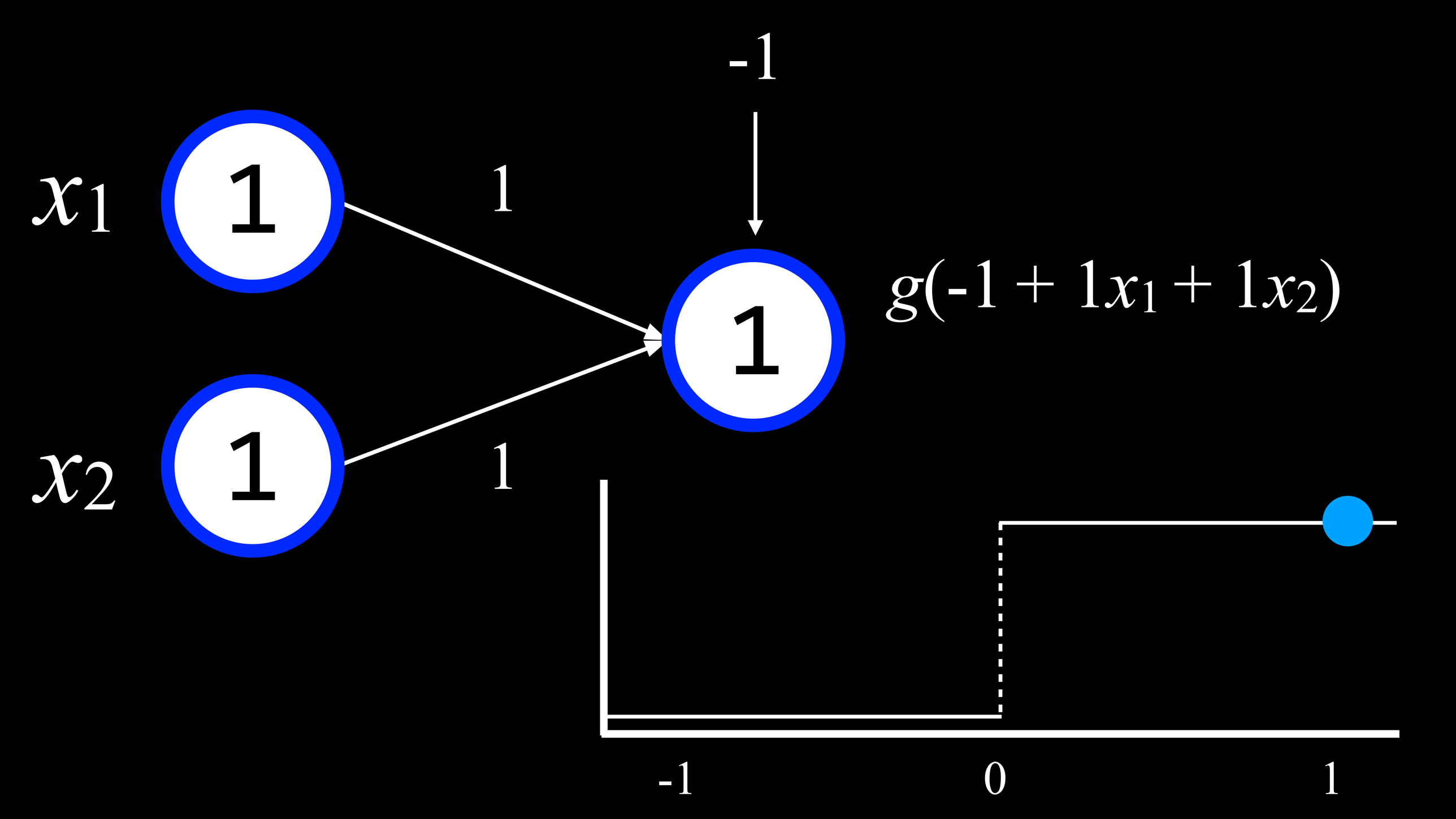


$$g(w_0 + w_1 x_1 + w_2 x_2)$$


-1
x_1
1
$g(-1 + 1x_1 + 1x_2)$
x_2
1
-1
0
1

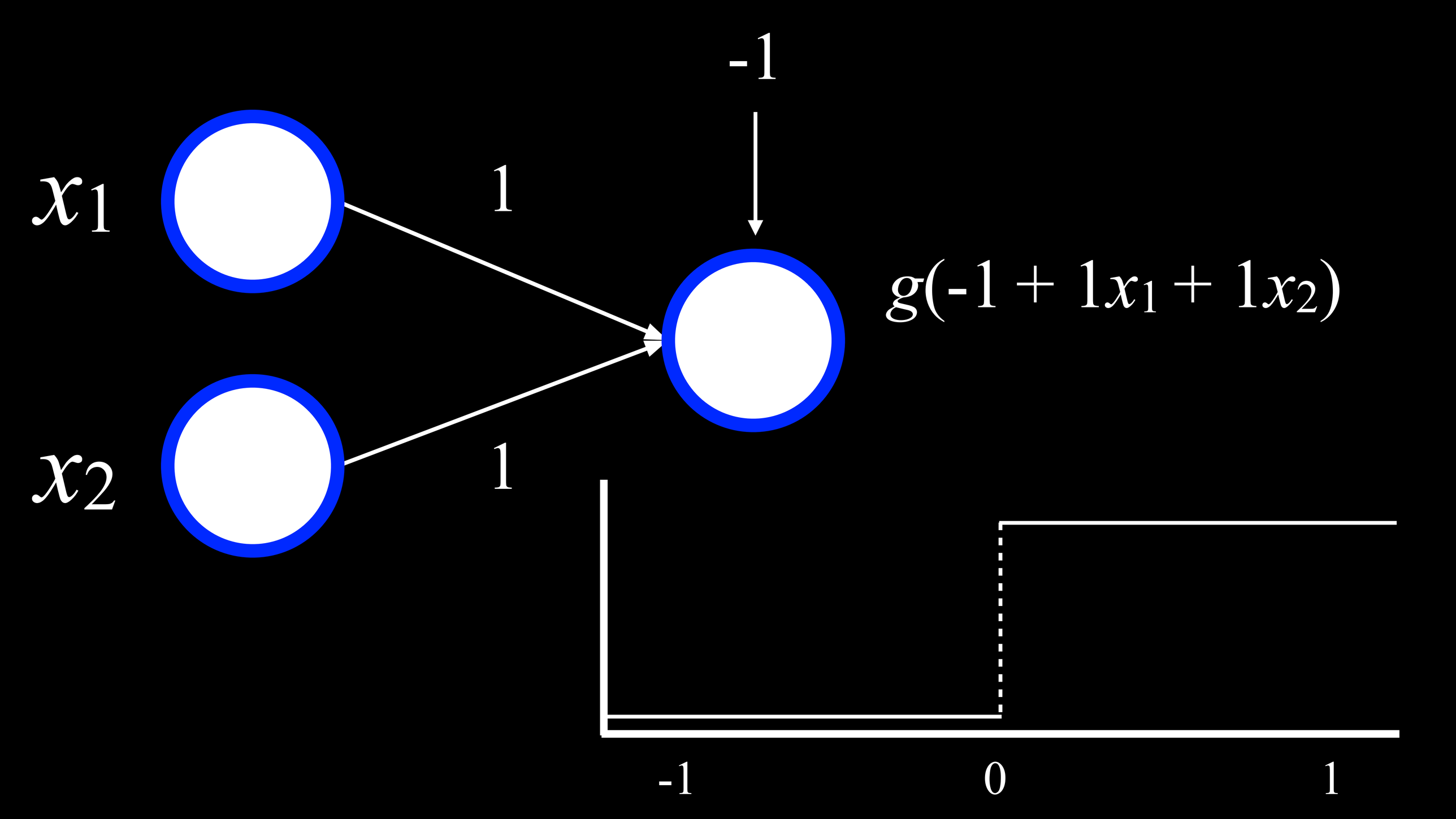

-1
x_1
0
1
$g(-1 + 1x_1 + 1x_2)$
0
x_2
0
1
-1
0
1

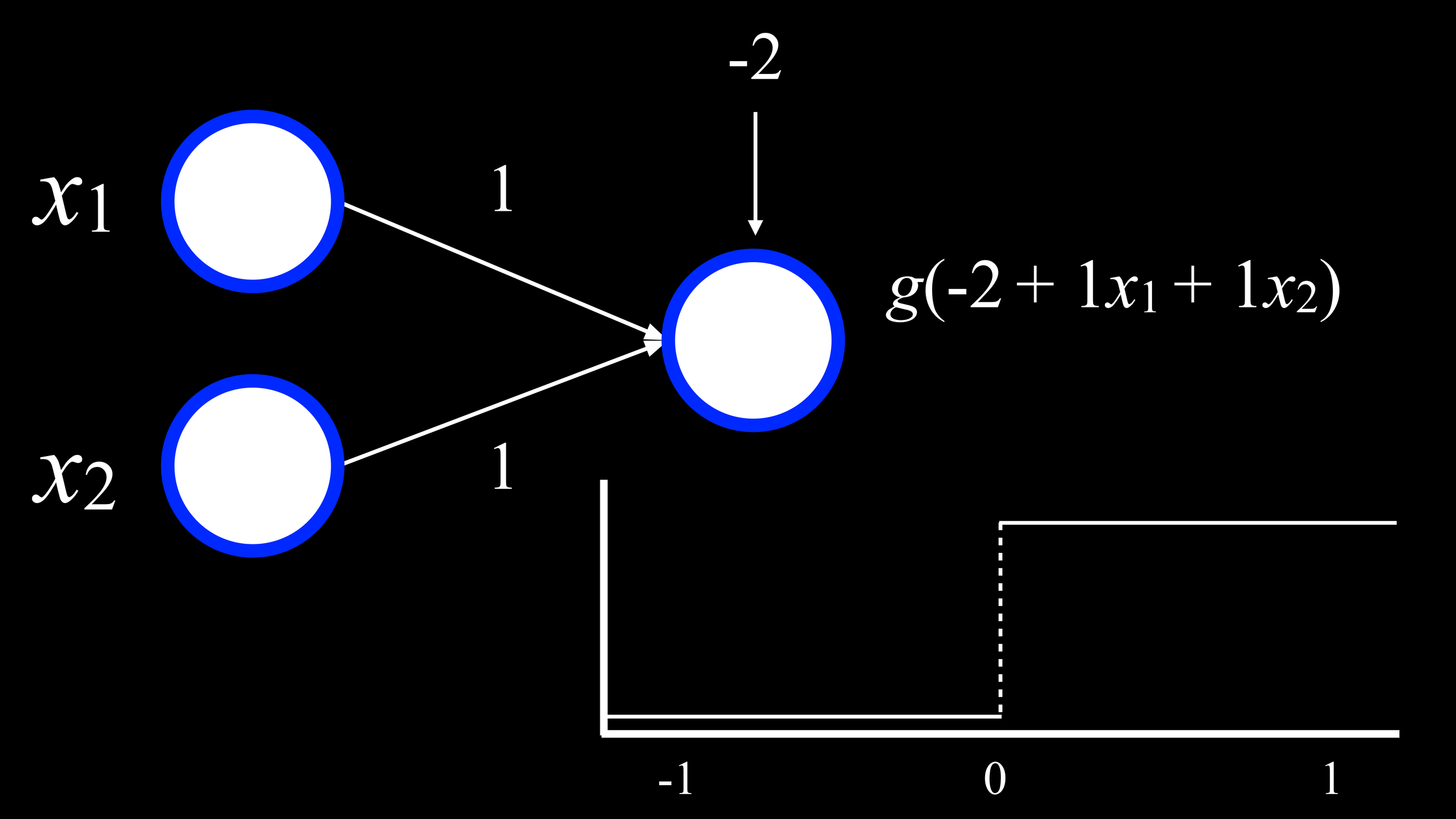

-1
x_1
1
1
1
$g(-1 + 1x_1 + 1x_2)$
x_2
0
1
-1
0
1

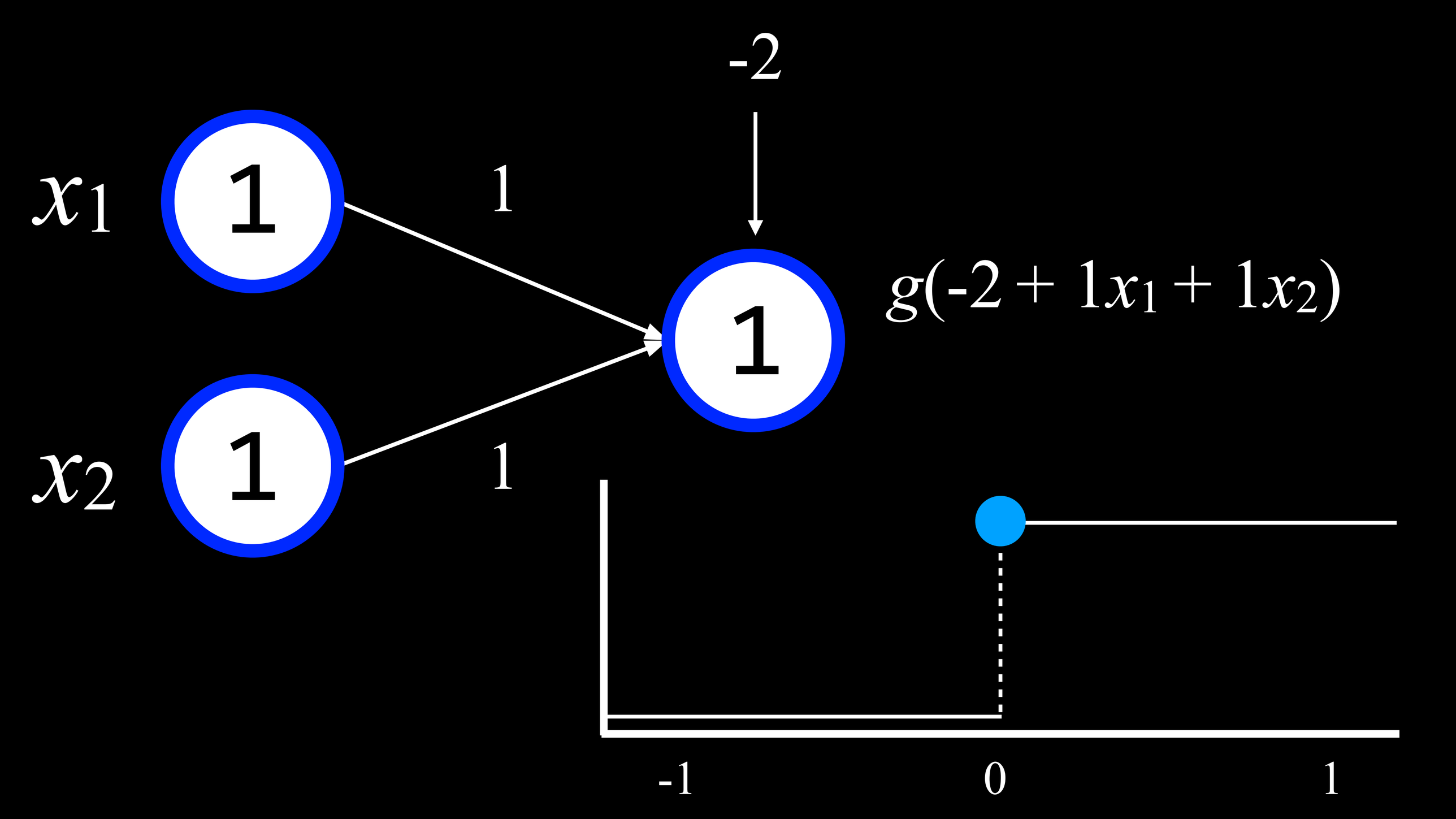

-1
x_1
1
1
1
$g(-1 + 1x_1 + 1x_2)$
x_2
1
1
-1
0
1

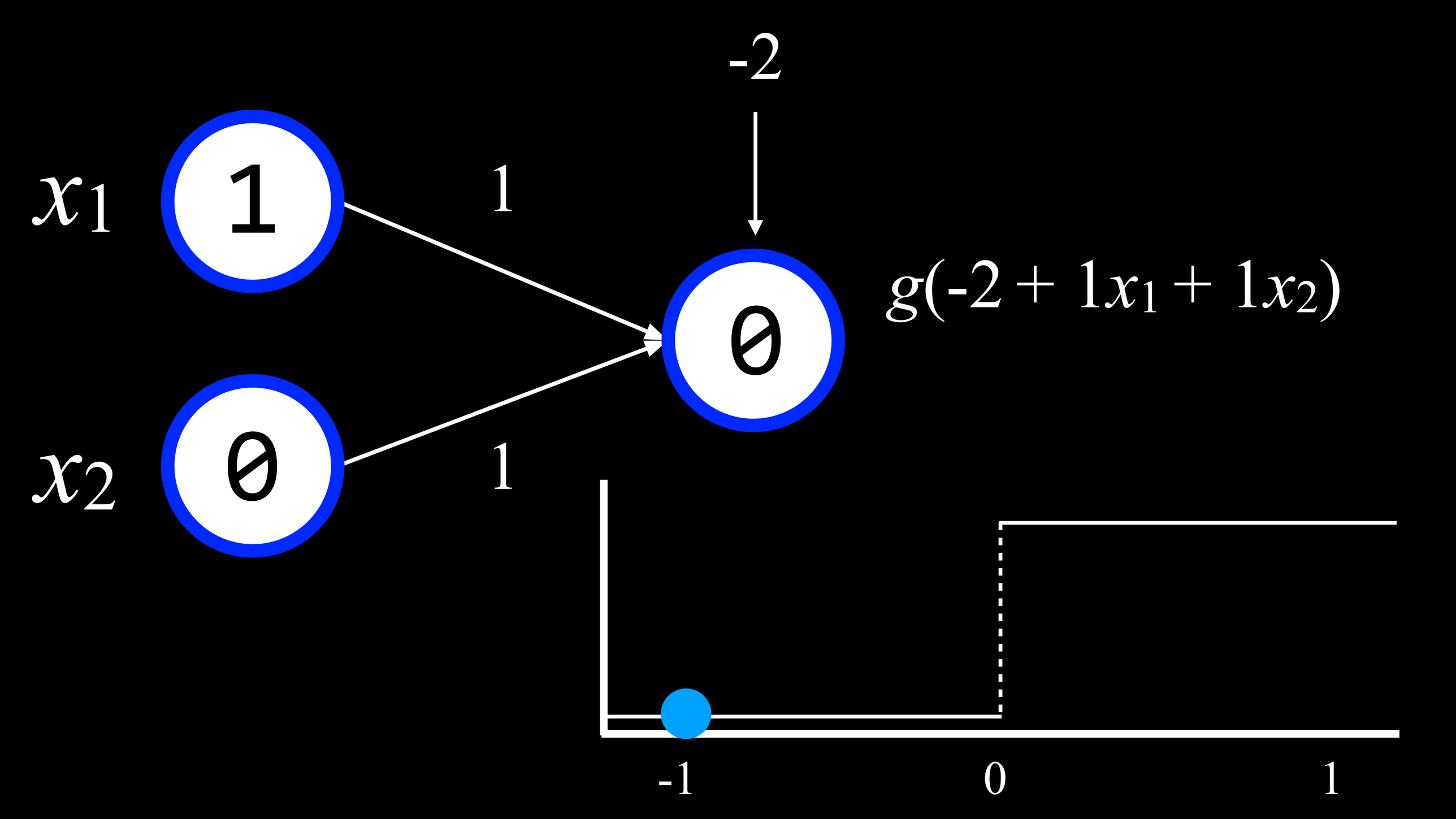
And

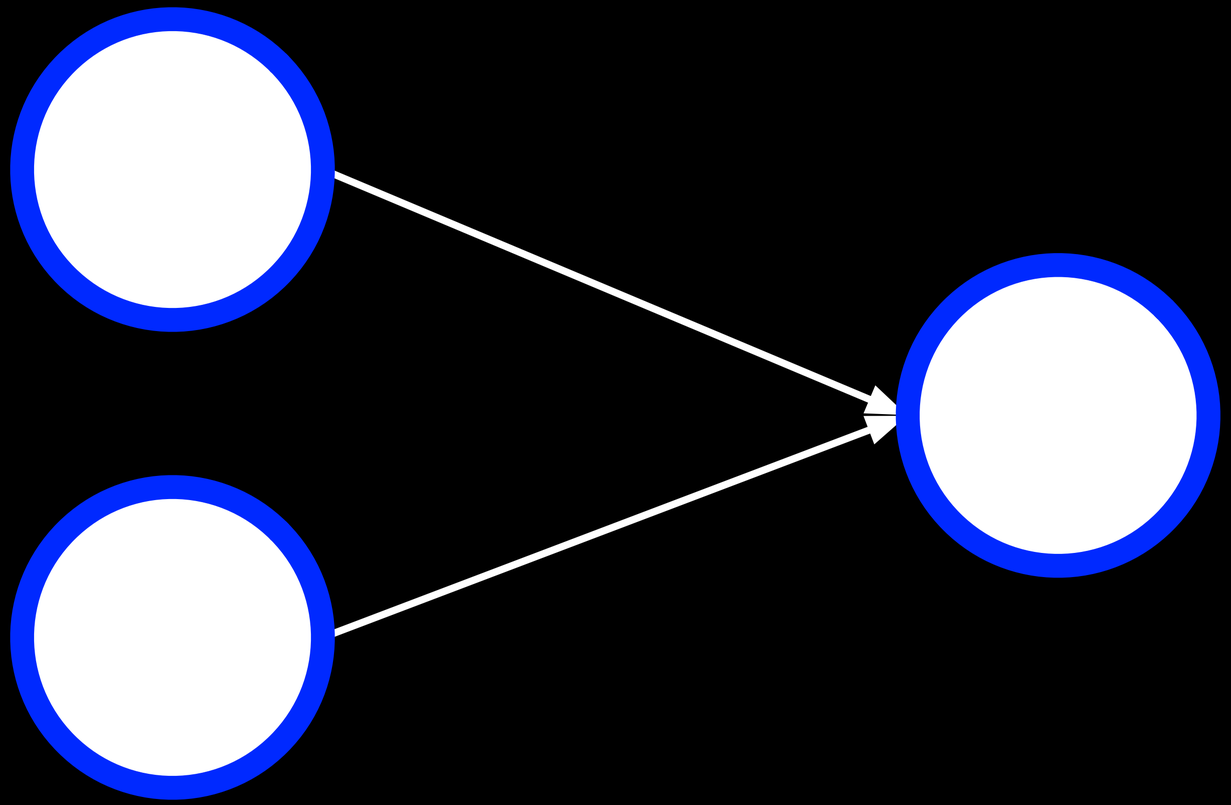
x	y	$f(x, y)$
0	0	0
0	1	0
1	0	0
1	1	1

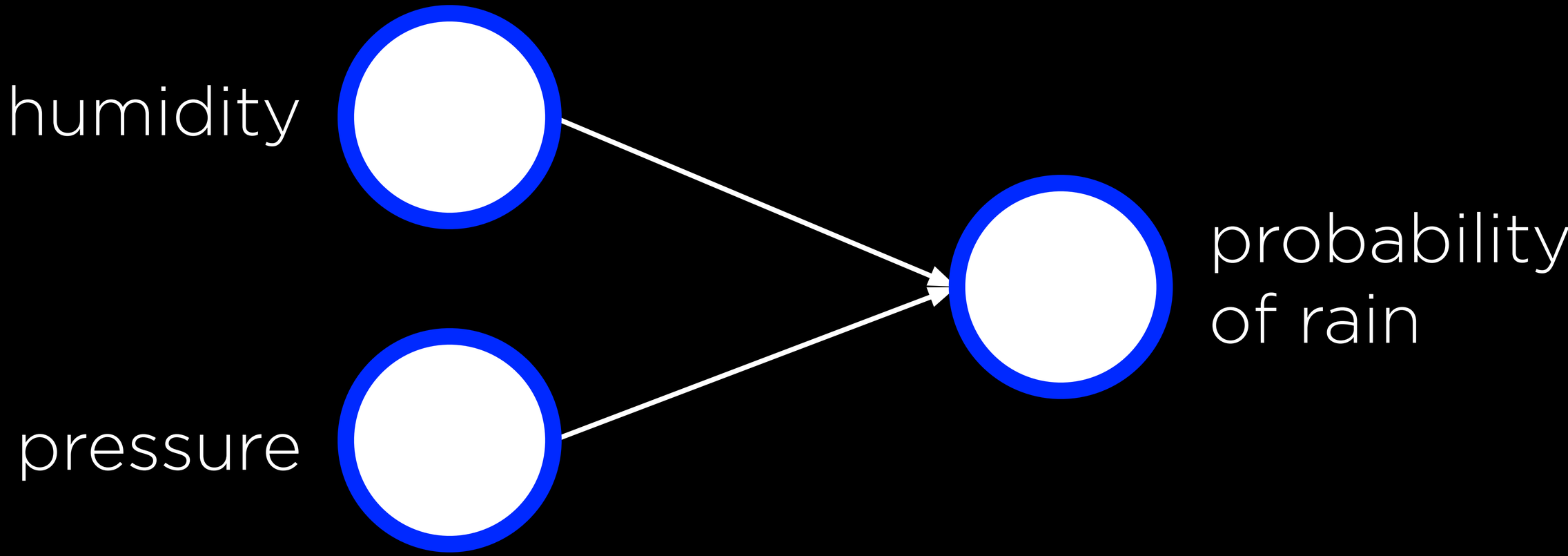

-1
x_1
1
$g(-1 + 1x_1 + 1x_2)$
x_2
1
-1
0
1

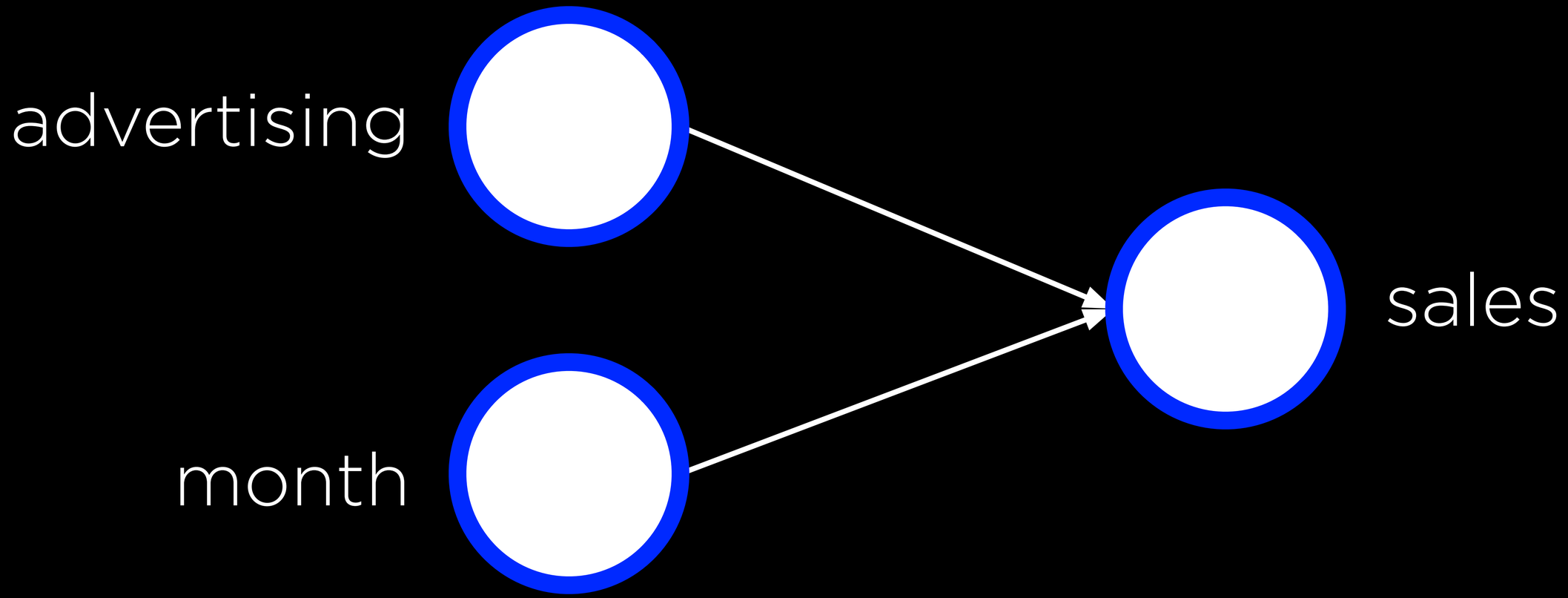

-2
x_1
1
$g(-2 + 1x_1 + 1x_2)$
x_2
1
-1
0
1

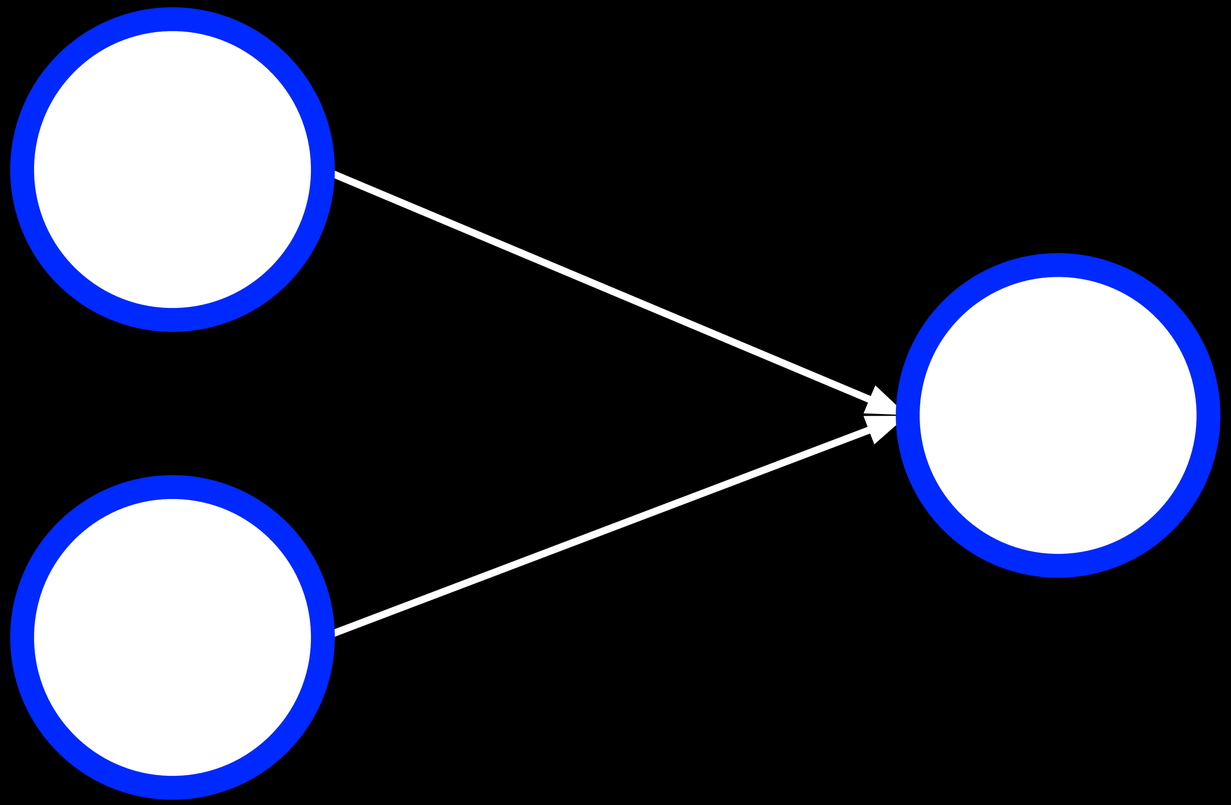

-2
x_1
1
1
1
$g(-2 + 1x_1 + 1x_2)$
x_2
1
1
-1
0
1

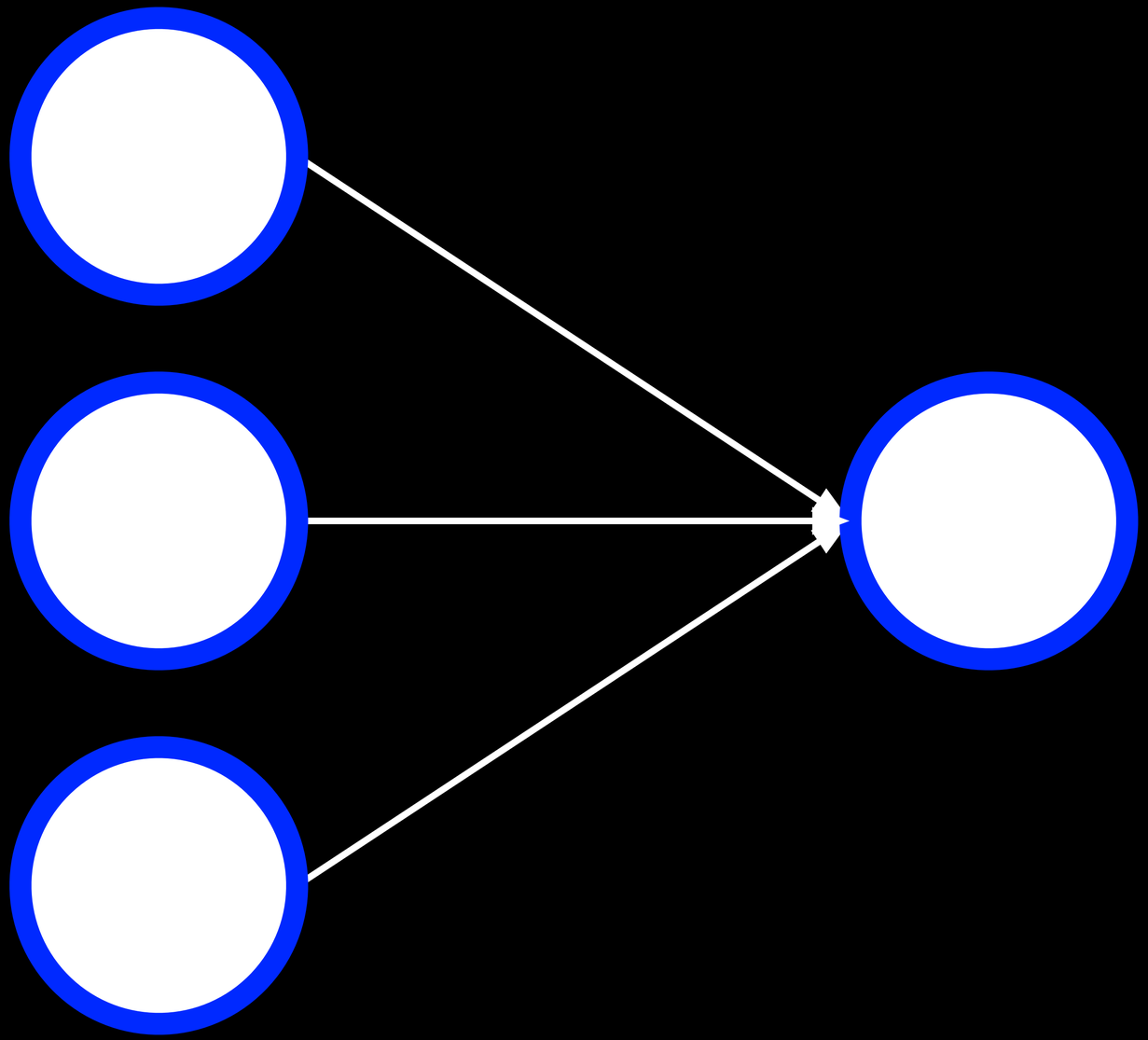

-2
x_1
1
1
0
$g(-2 + 1x_1 + 1x_2)$
x_2
0
1
-1
0
1

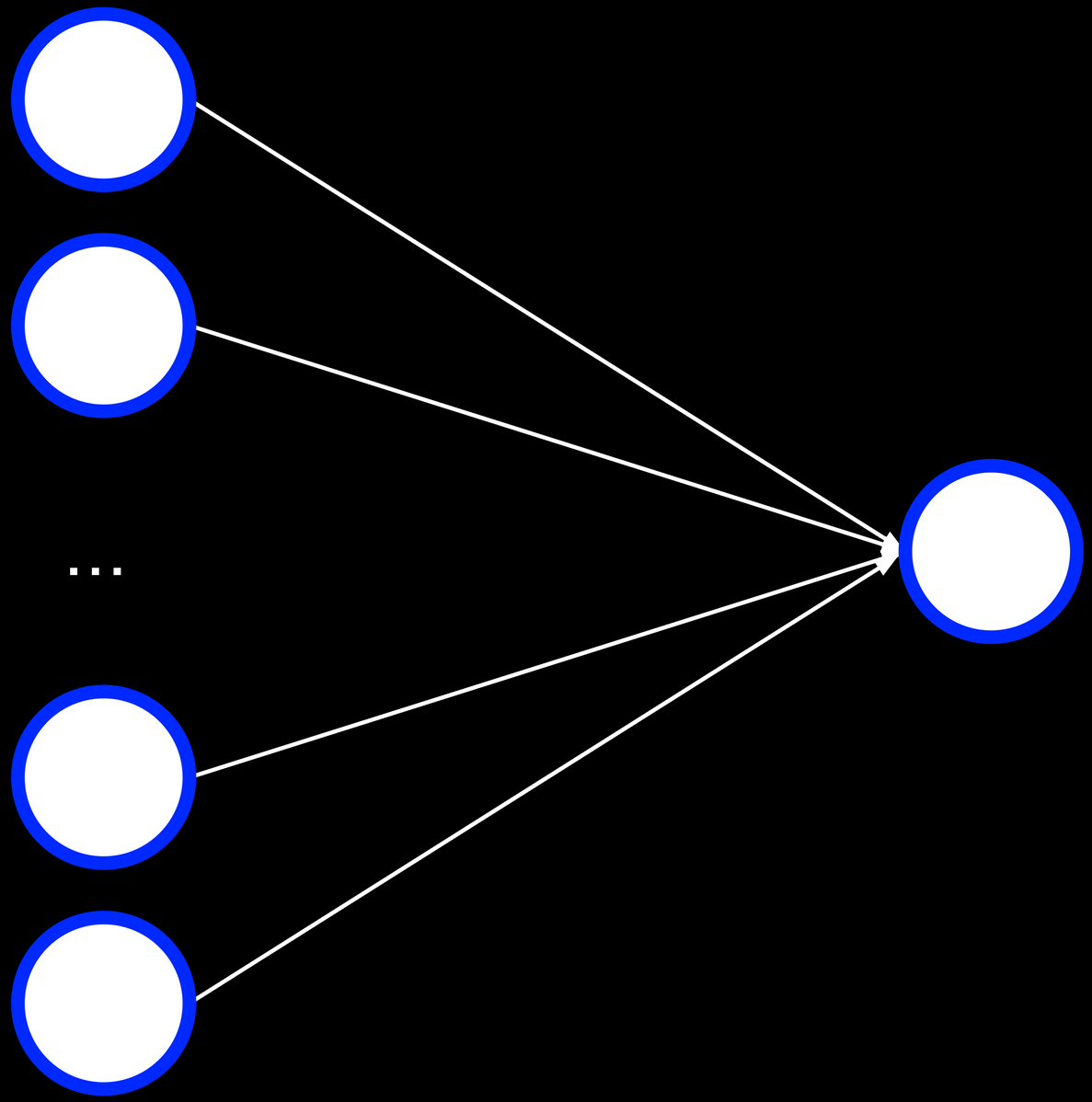


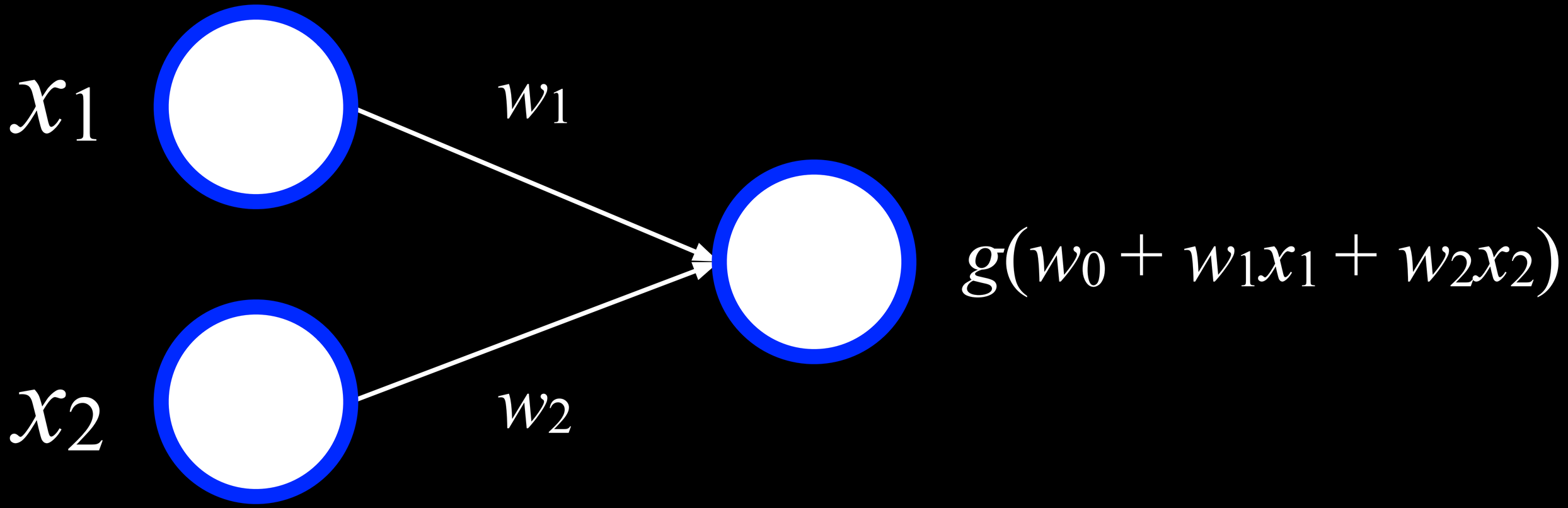

humidity
pressure
probability
of rain

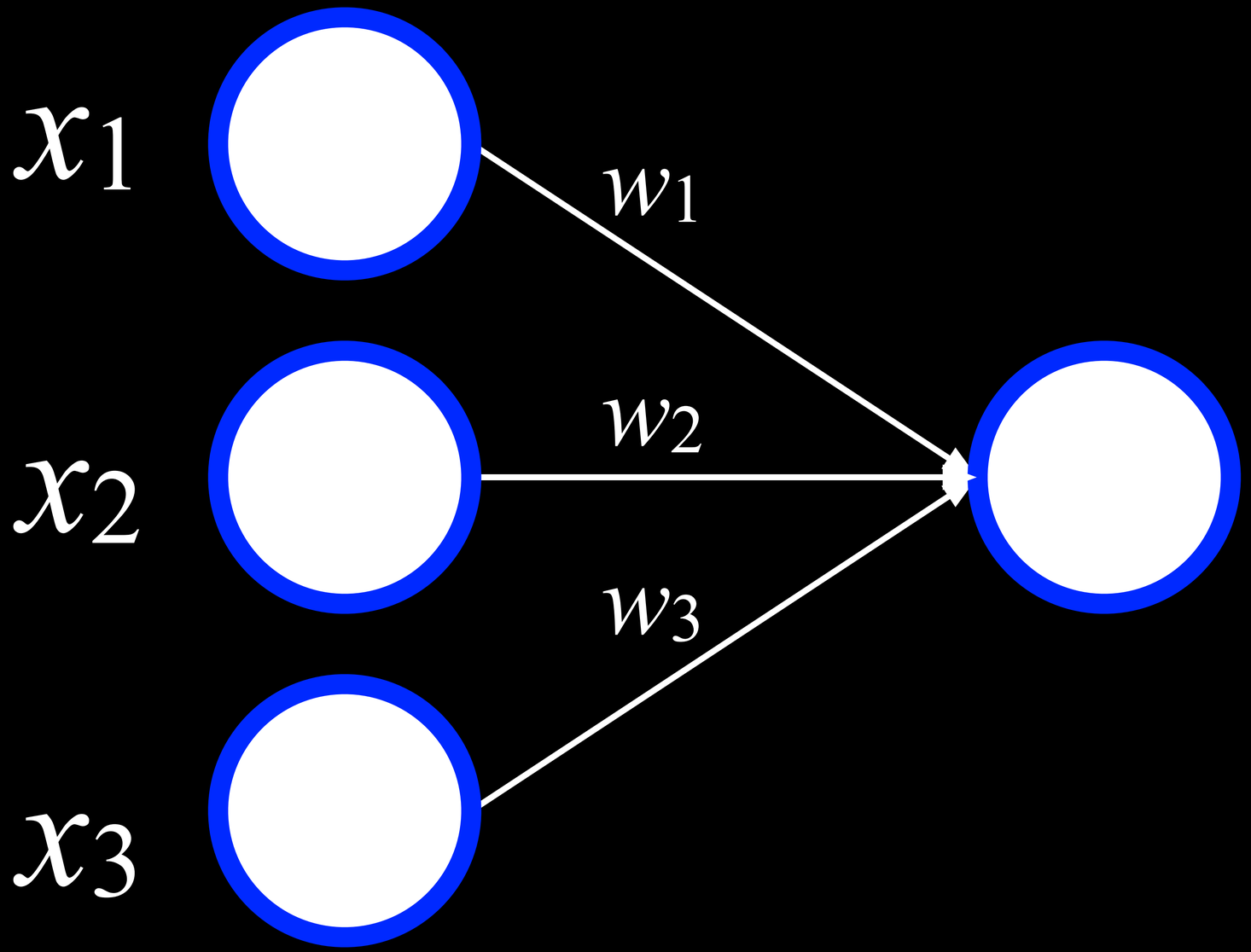

advertising
month
sales



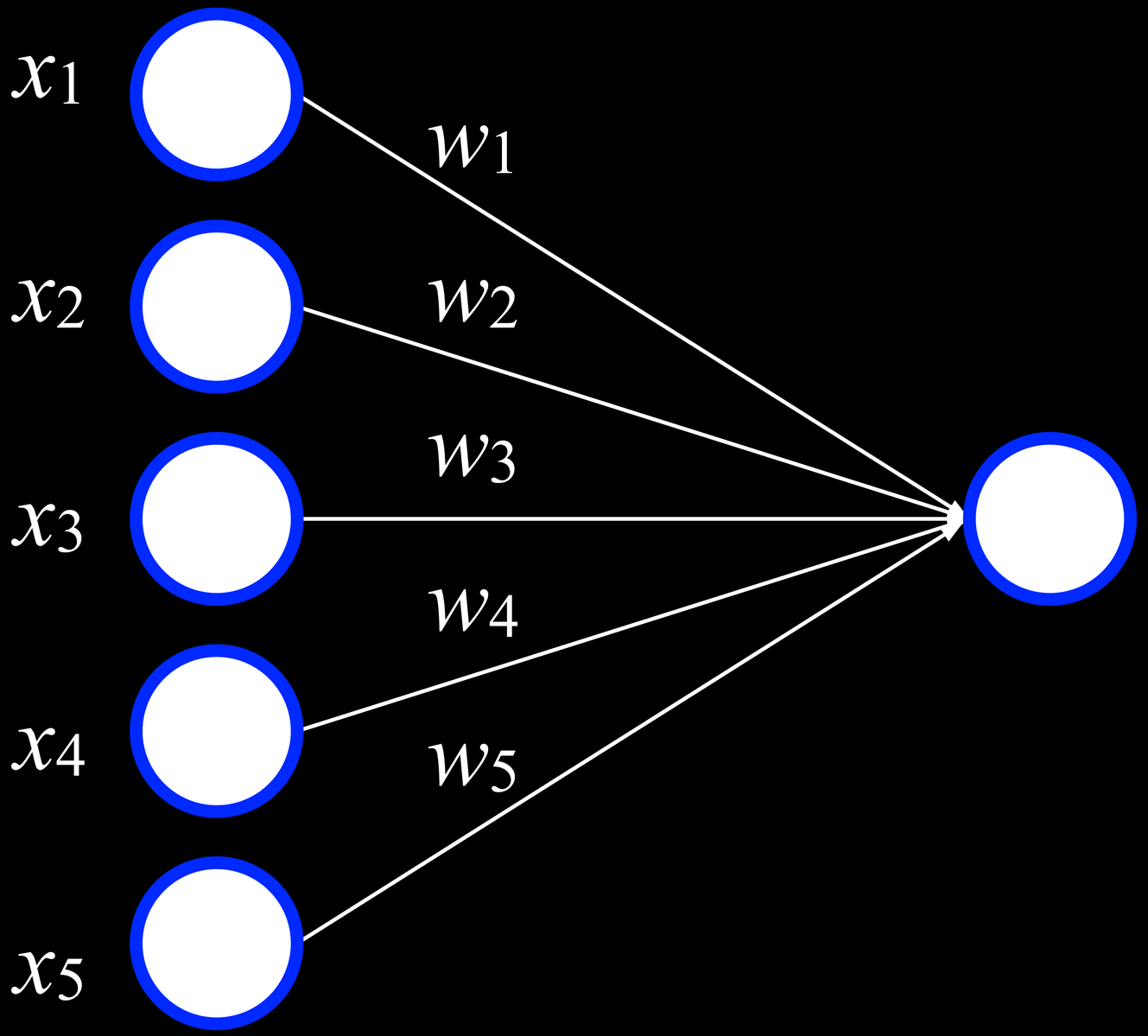




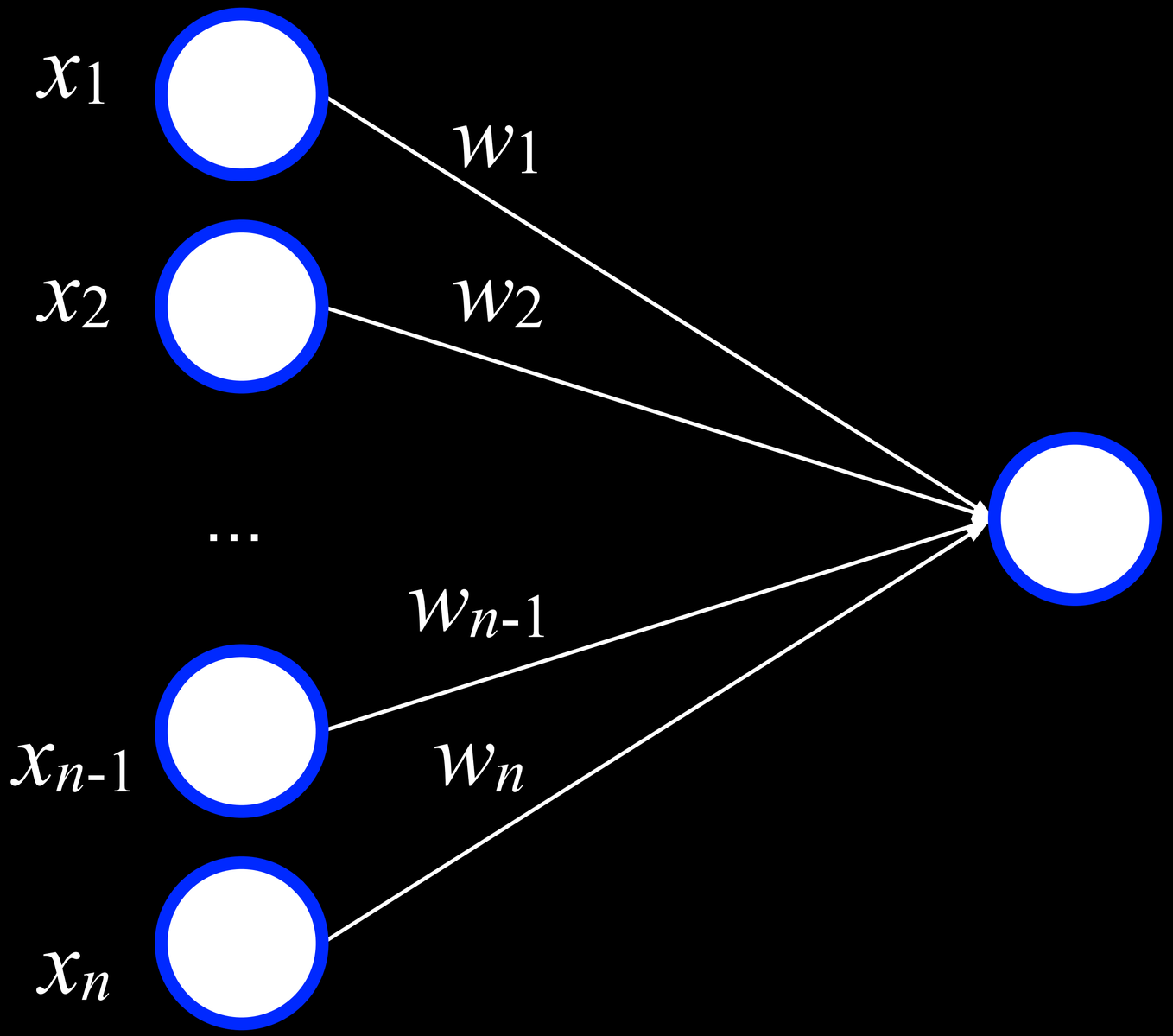

x_1
x_2
w_1
w_2
$g(w_0 + w_1x_1 + w_2x_2)$



$$g(w_0 + w_1x_1 + w_2x_2 + w_3x_3)$$



$$g\left(\sum_{i=1}^{5} x_i w_i + w_0\right)$$



$$g\left(\sum_{i=1}^{n} x_i w_i + w_0\right)$$

gradient descent

algorithm for minimizing loss when training neural network

Gradient Descent

- Start with a random choice of weights.
- Repeat:
  - Calculate the gradient based on all data points: direction that will lead to decreasing loss.
  - Update weights according to the gradient.

Gradient Descent

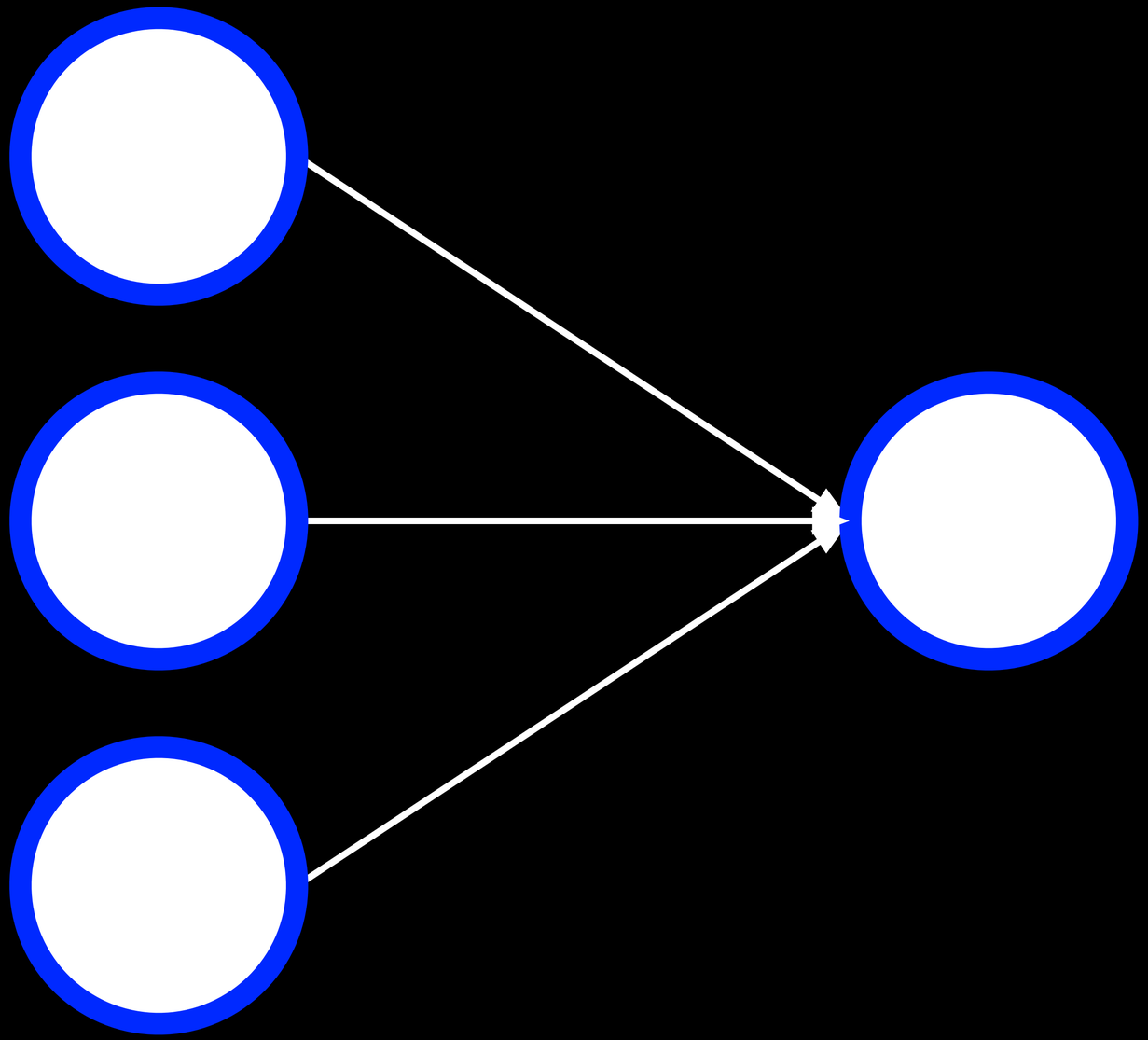
- Start with a random choice of weights.
- Repeat:
  - Calculate the gradient based on **all data points**: direction that will lead to decreasing loss.
  - Update weights according to the gradient.

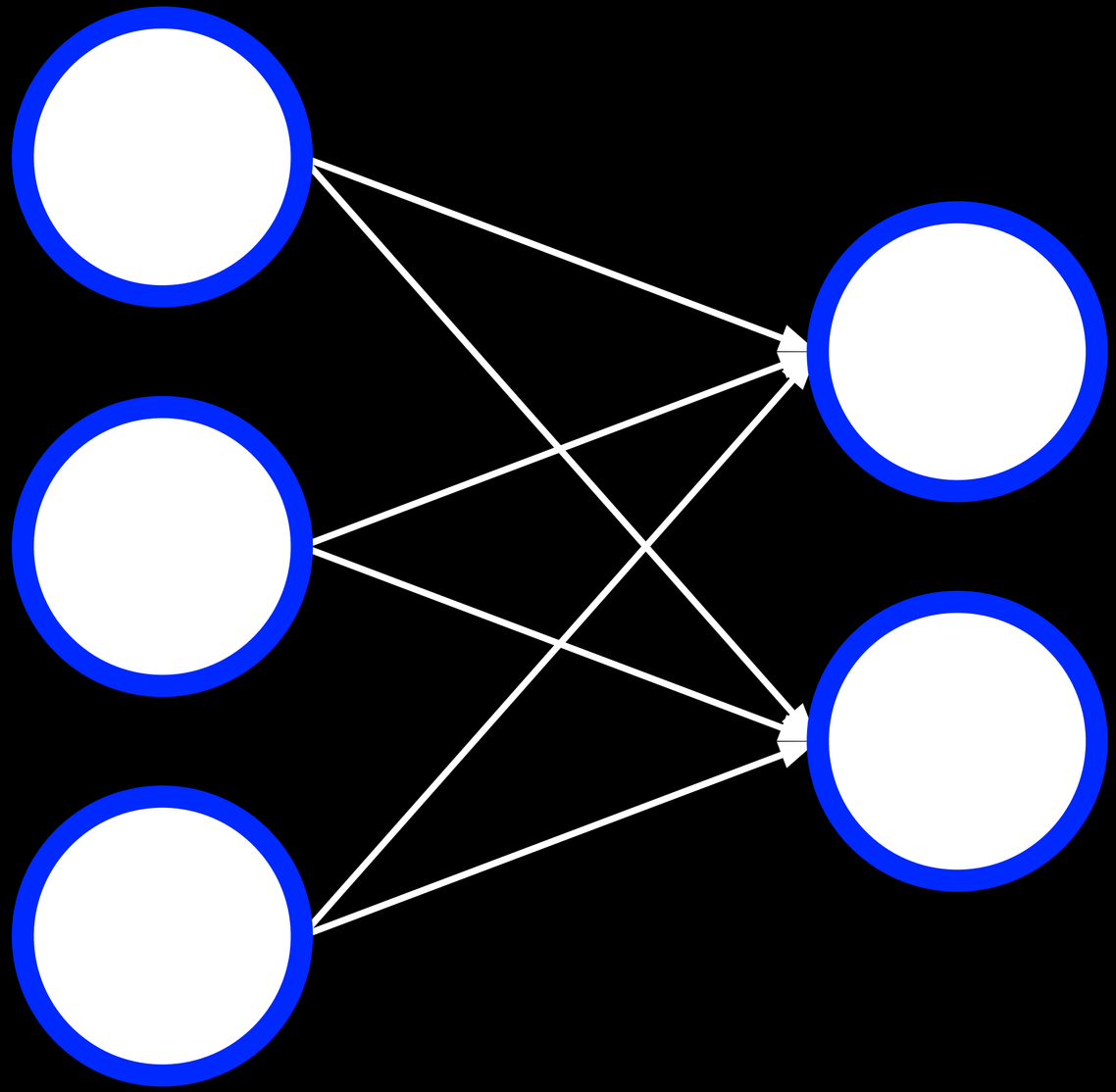
Stochastic Gradient Descent

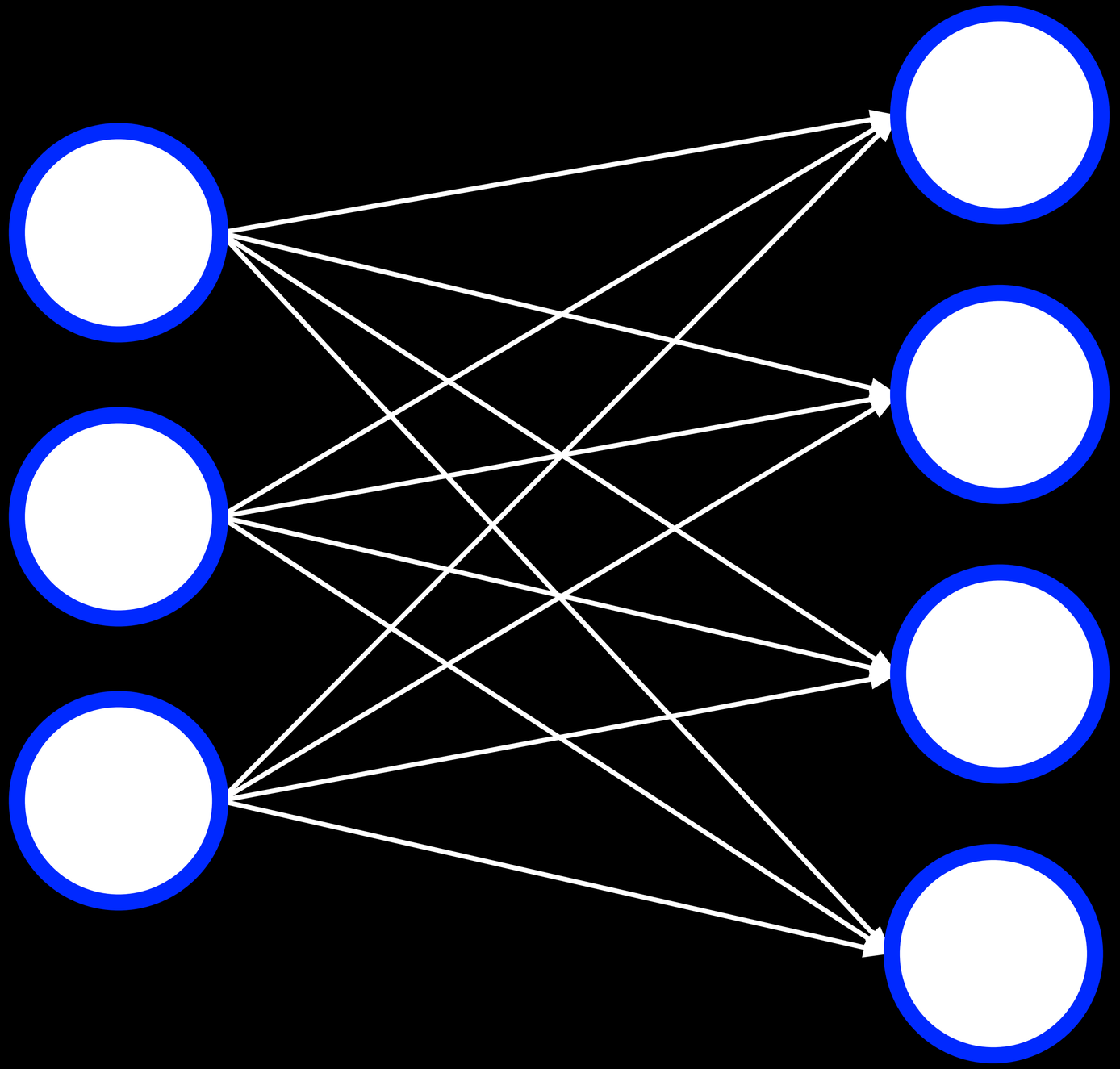
- Start with a random choice of weights.
- Repeat:
  - Calculate the gradient based on **one** **data point**: direction that will lead to decreasing loss.
  - Update weights according to the gradient.

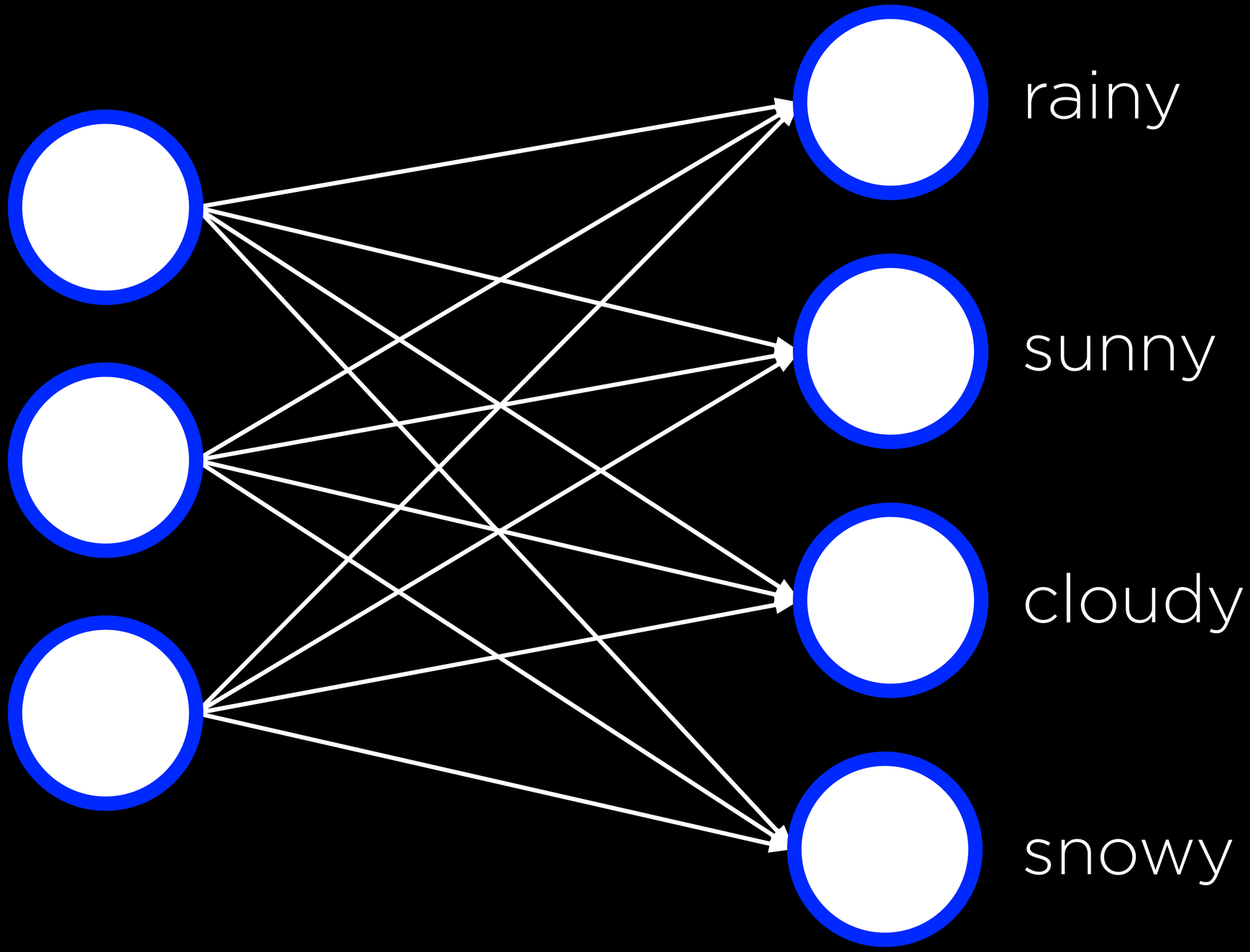
Mini-Batch Gradient Descent

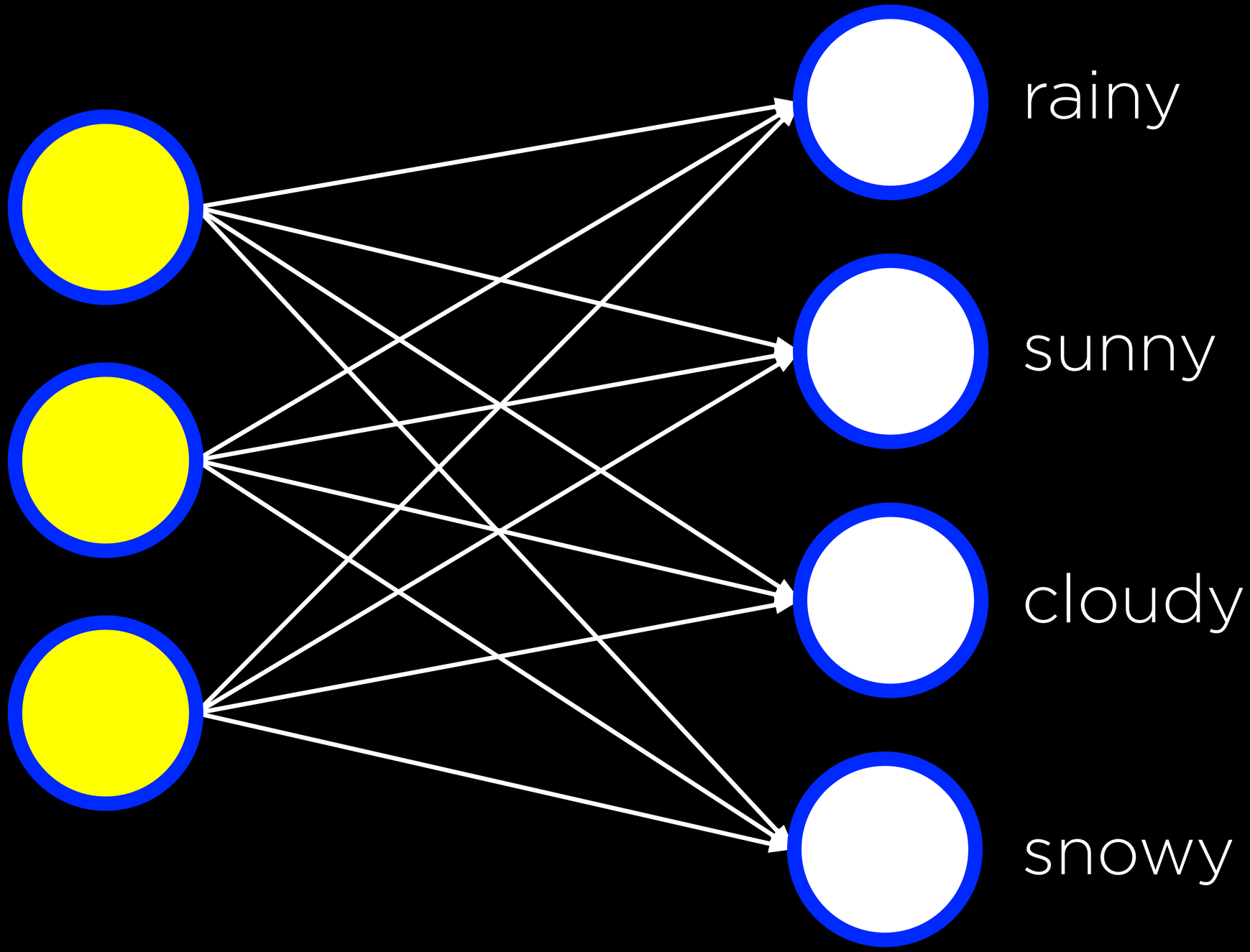
- Start with a random choice of weights.
- Repeat:
  - Calculate the gradient based on **one small batch**: direction that will lead to decreasing loss.
  - Update weights according to the gradient.

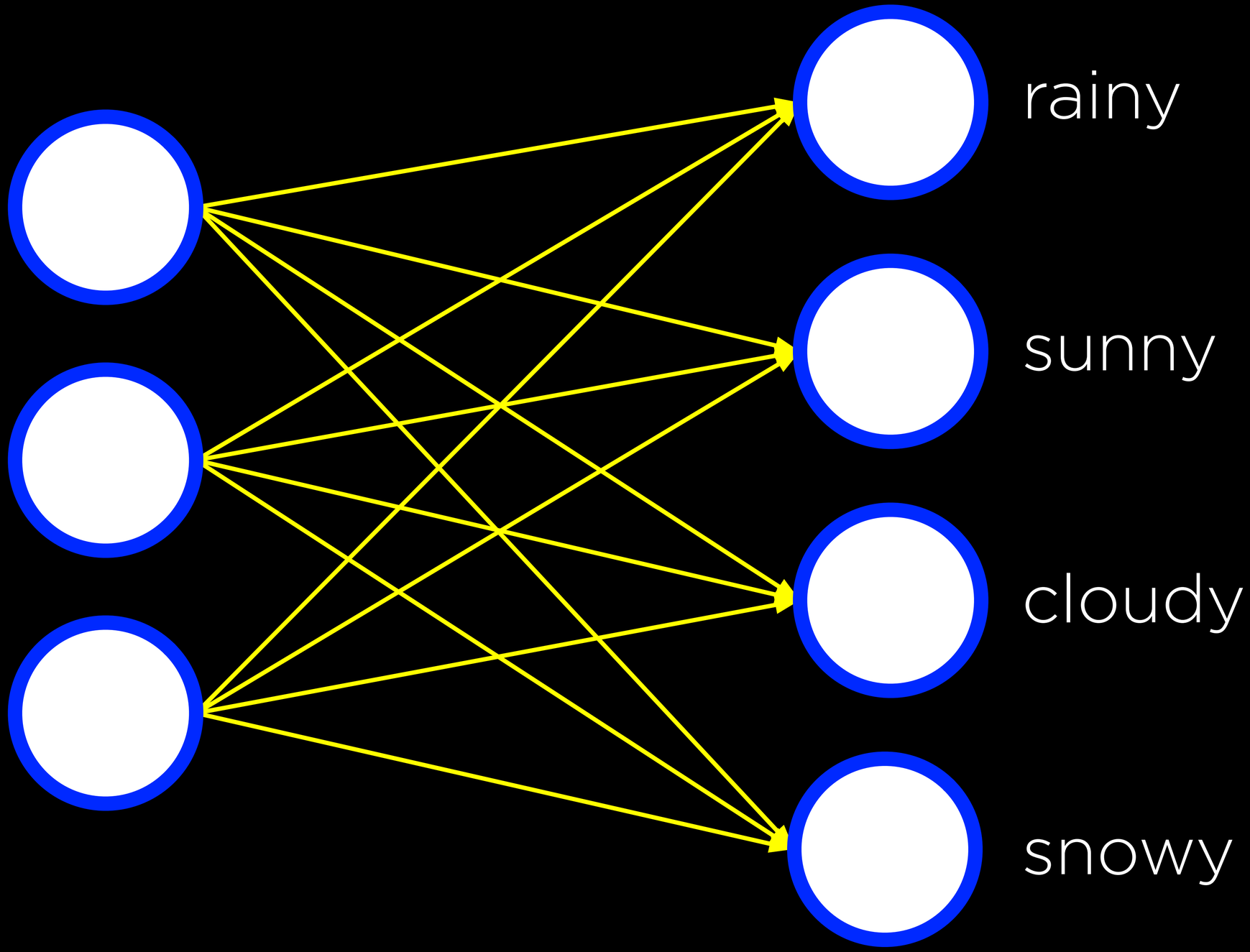


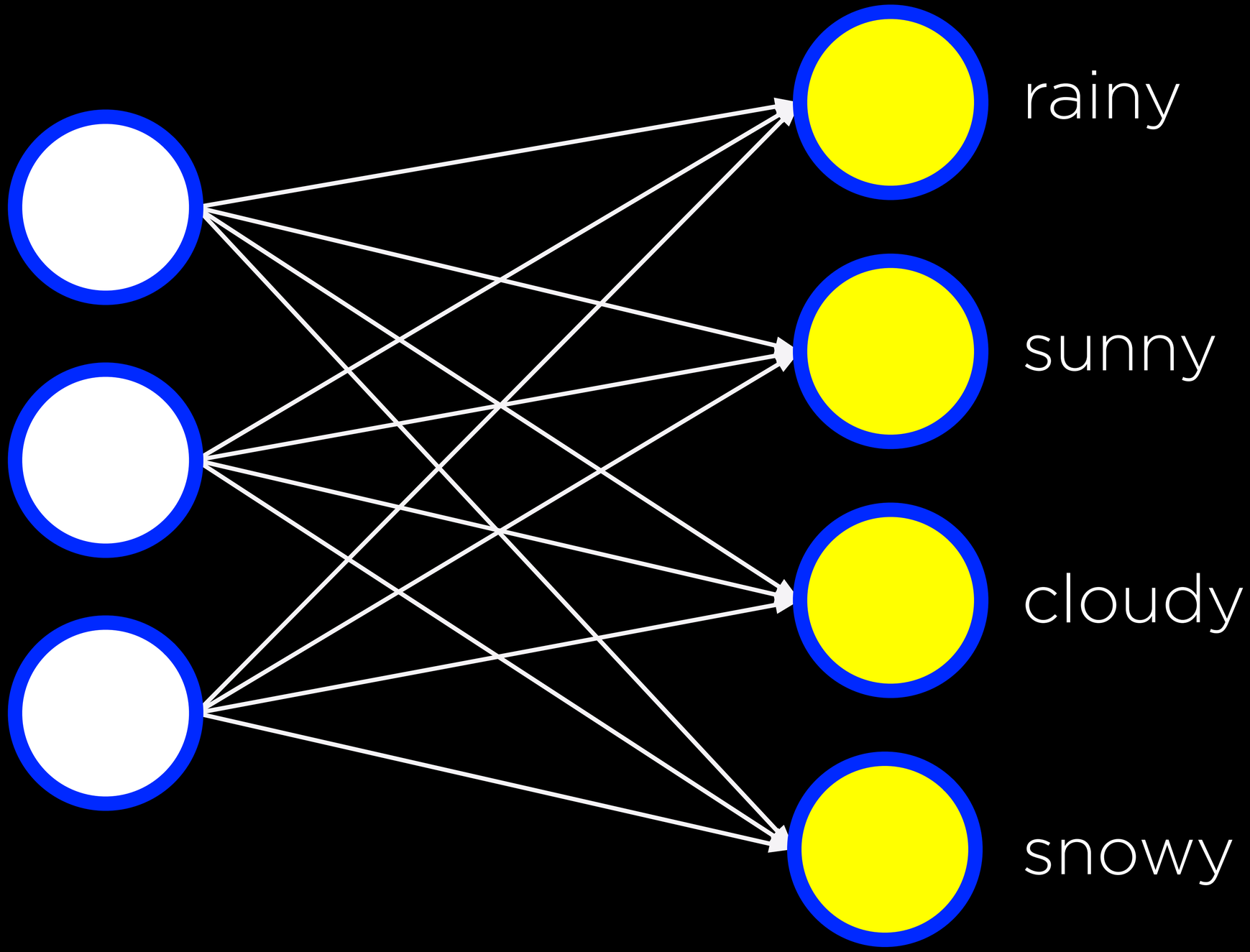


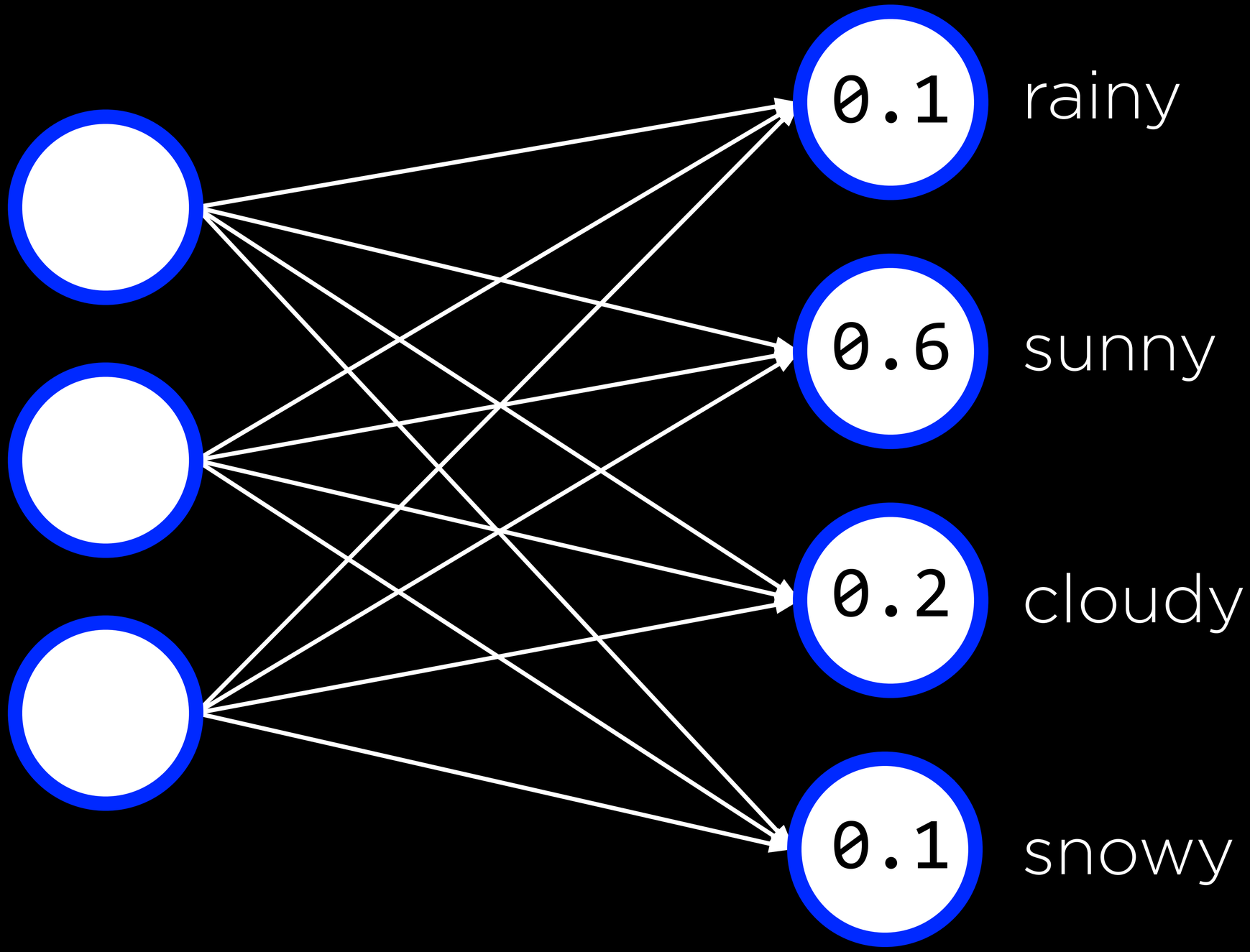


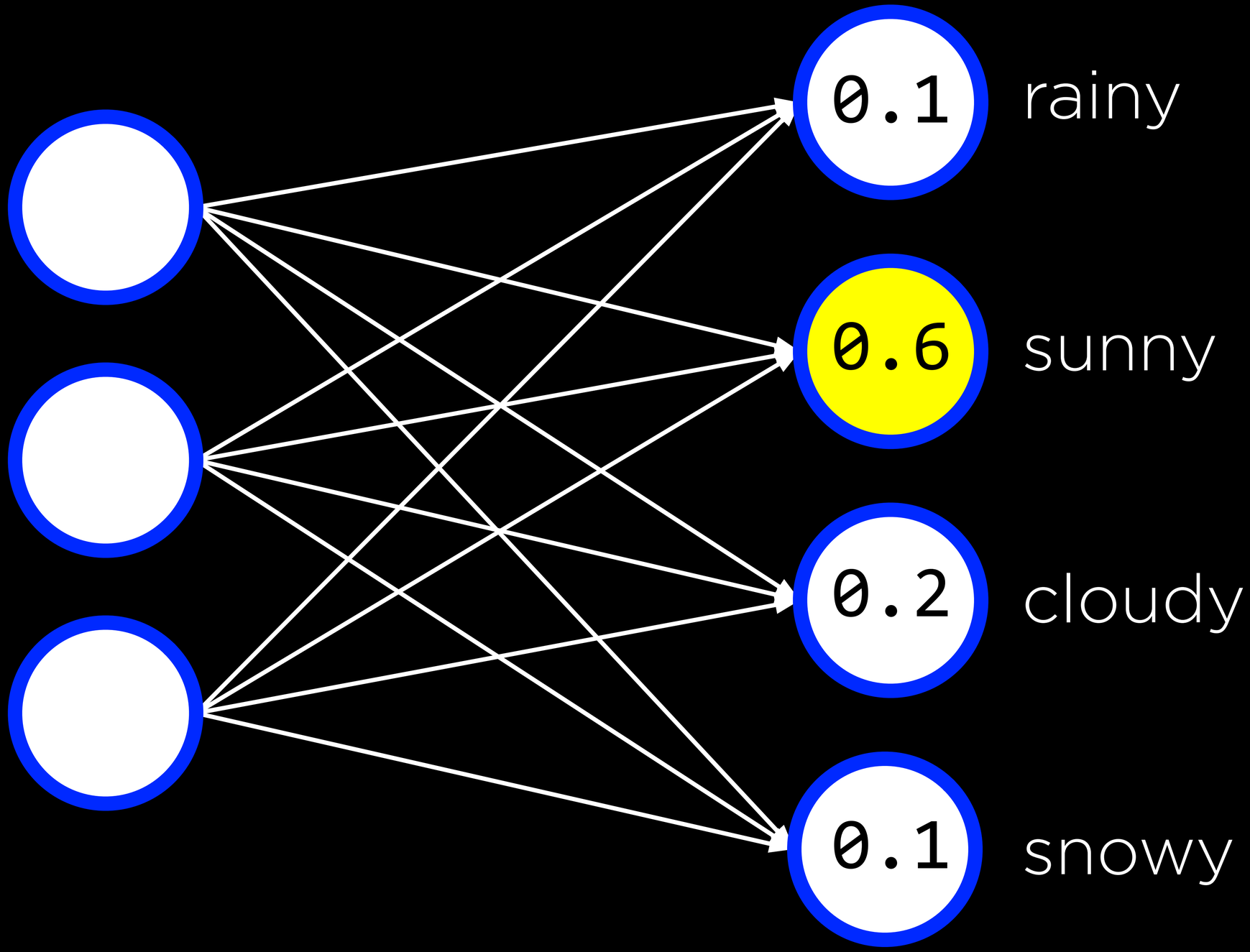

rainy
sunny
cloudy
snowy

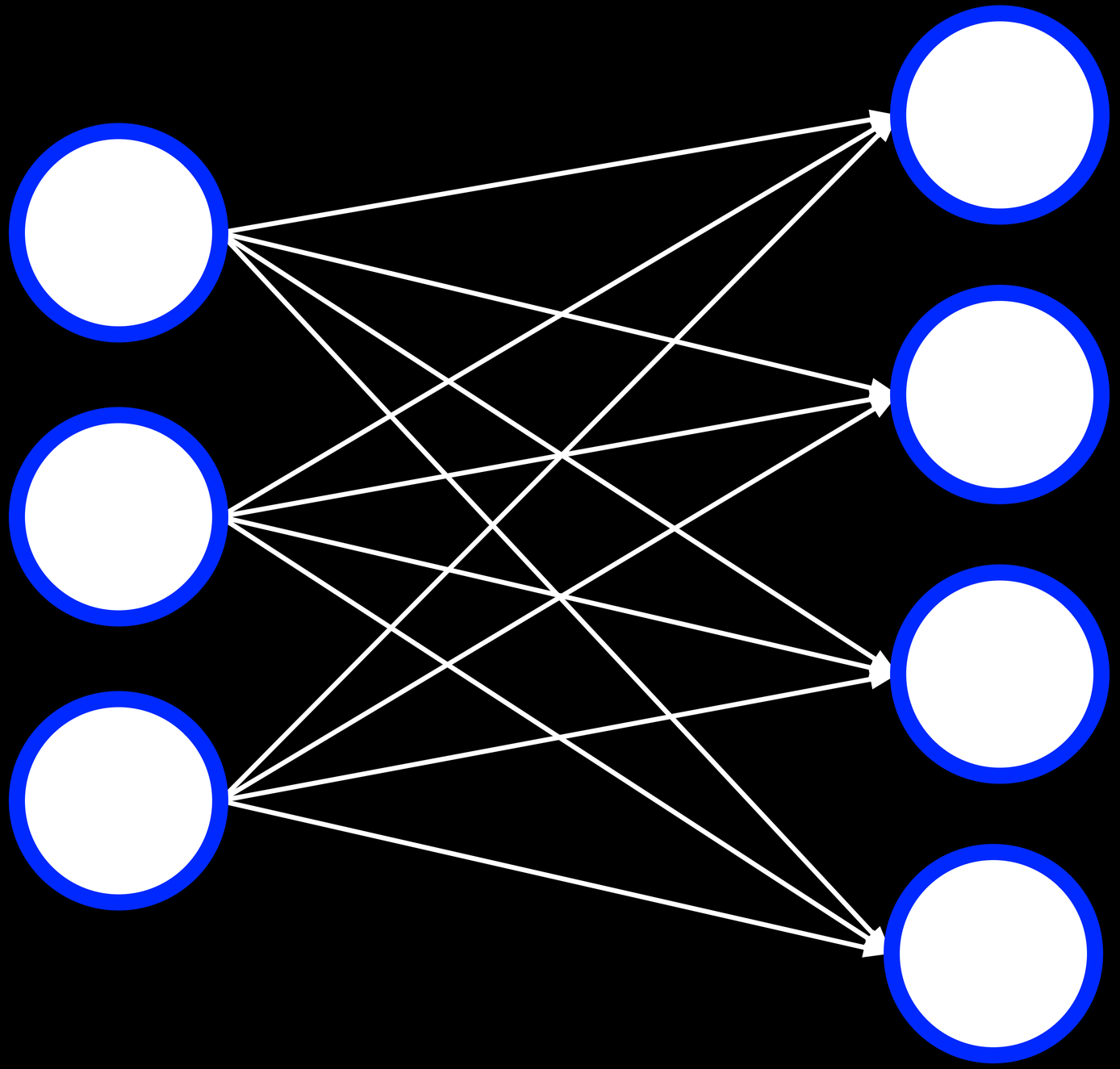

rainy
sunny
cloudy
snowy

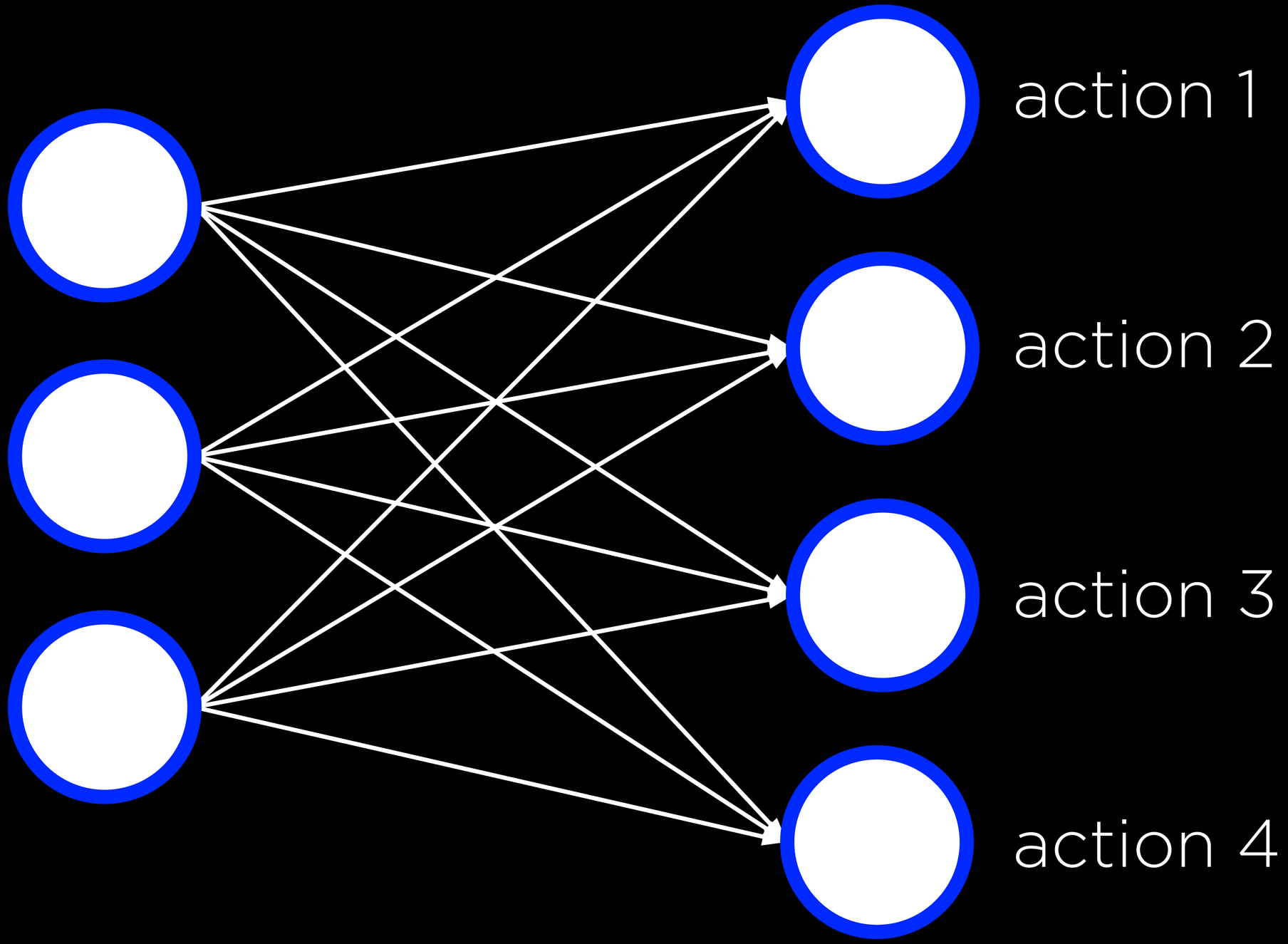

rainy
sunny
cloudy
snowy

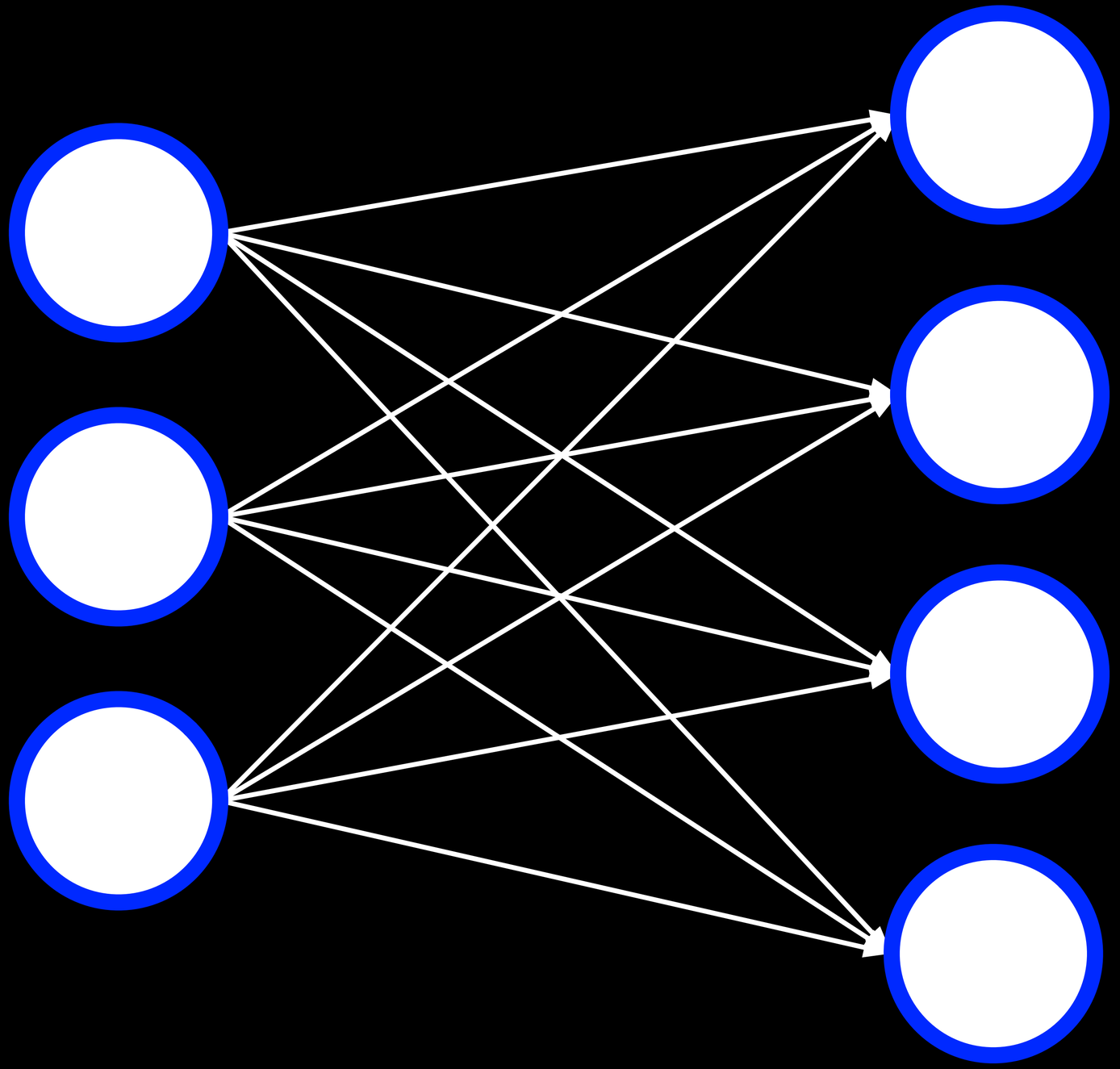

rainy
sunny
cloudy
snowy

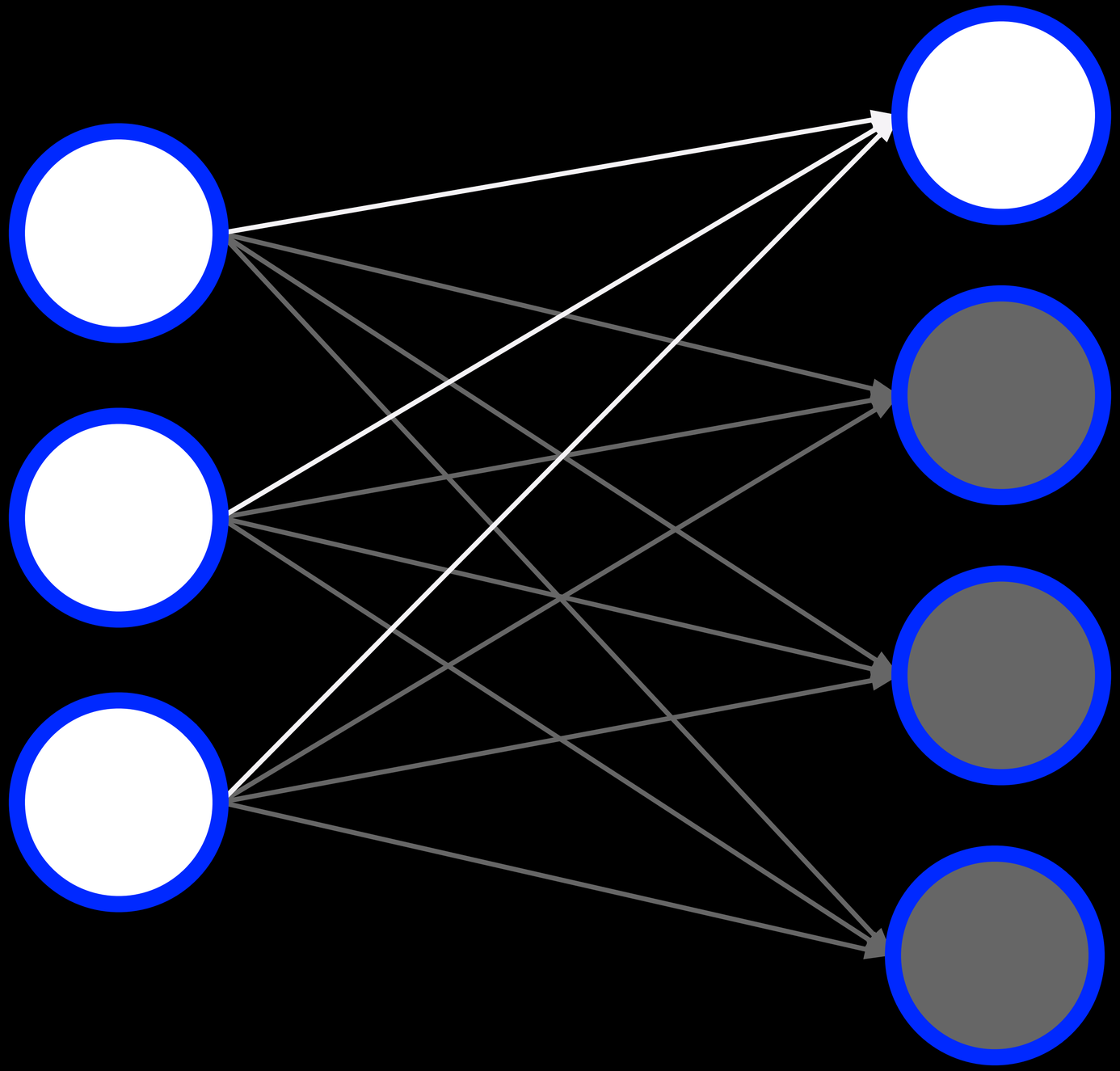

0.1
rainy
0.6
sunny
0.2
cloudy
0.1
snowy

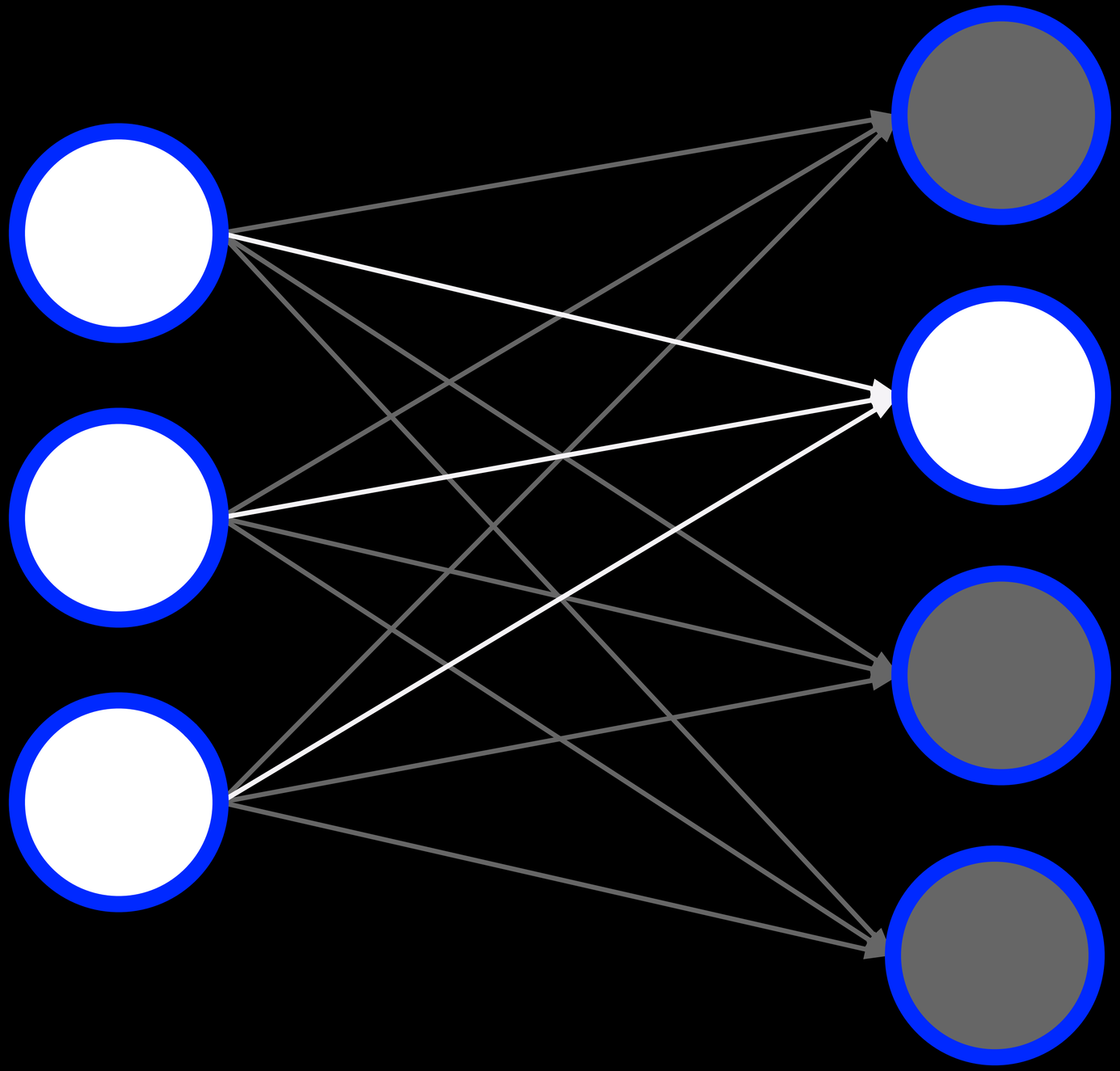

0.1
rainy
0.6
sunny
0.2
cloudy
0.1
snowy

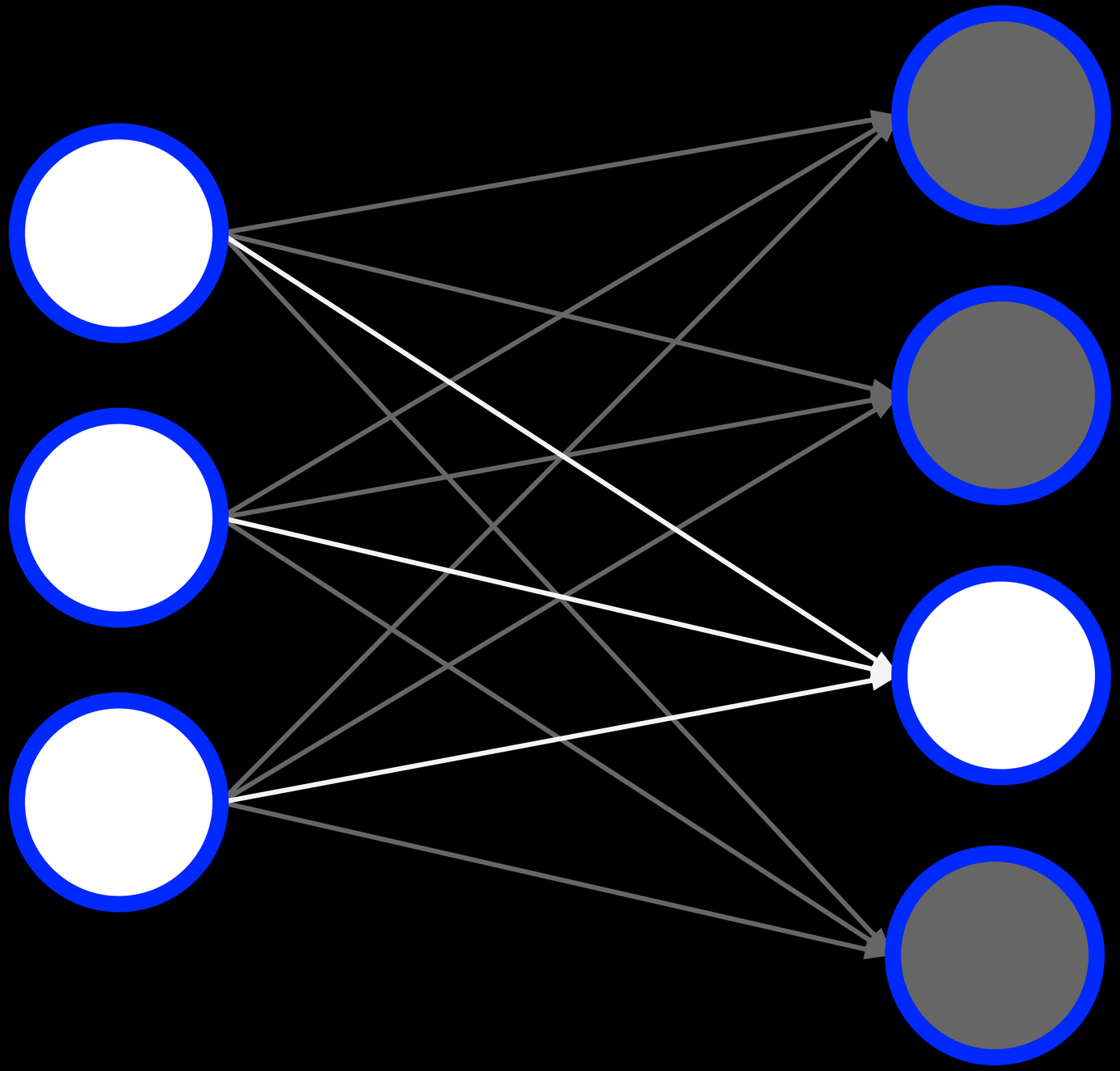


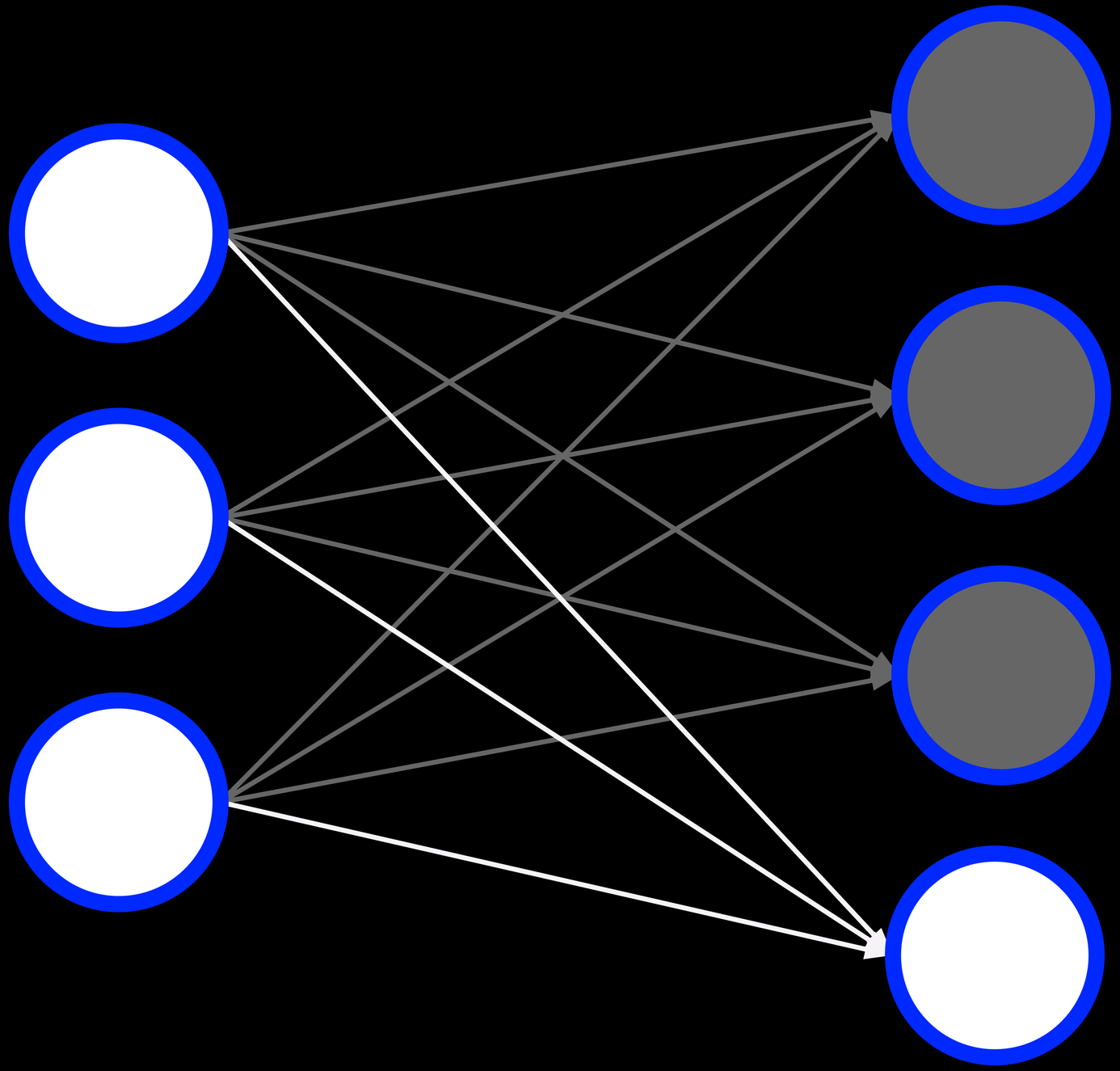

action 1
action 2
action 3
action 4

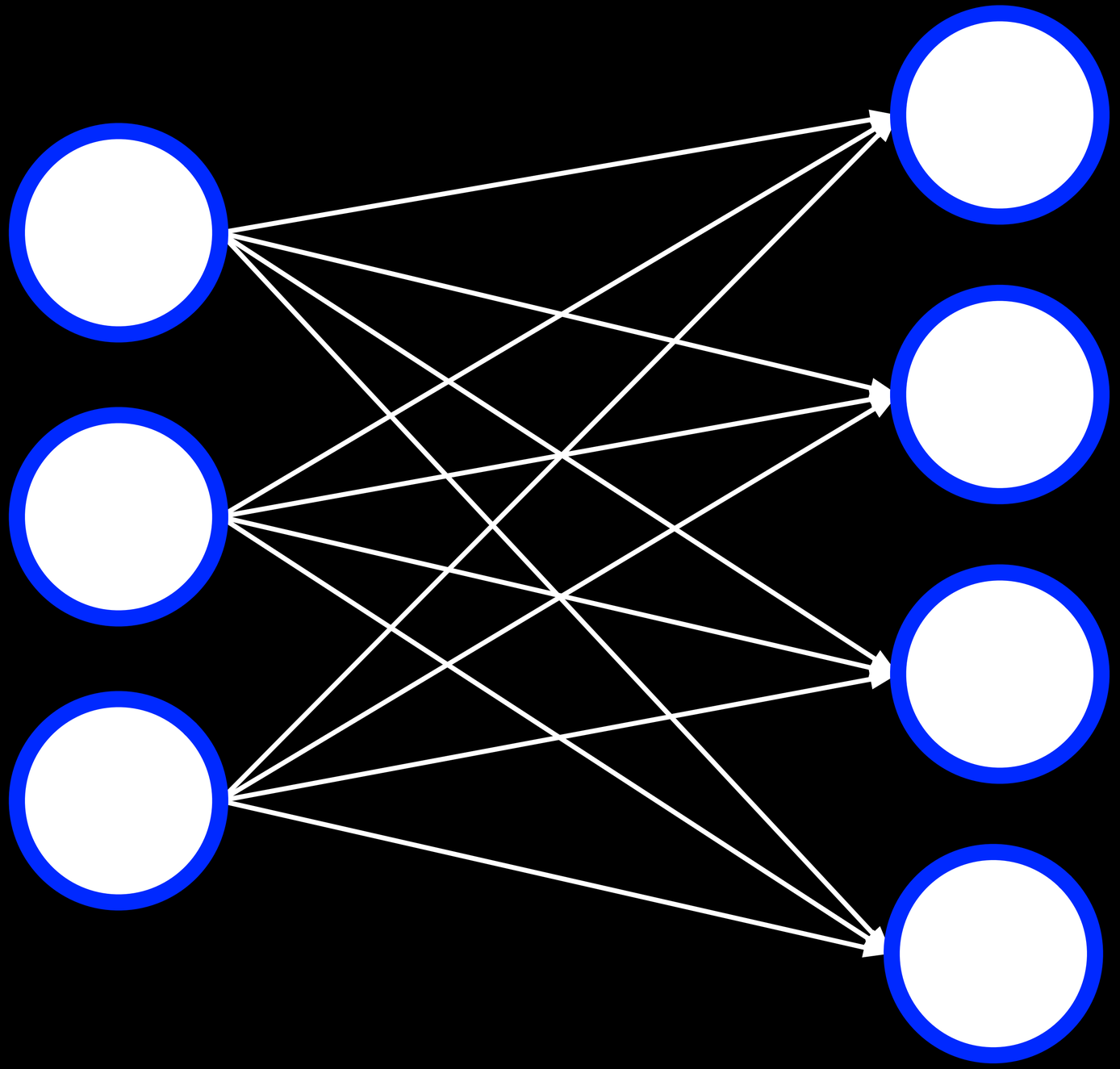


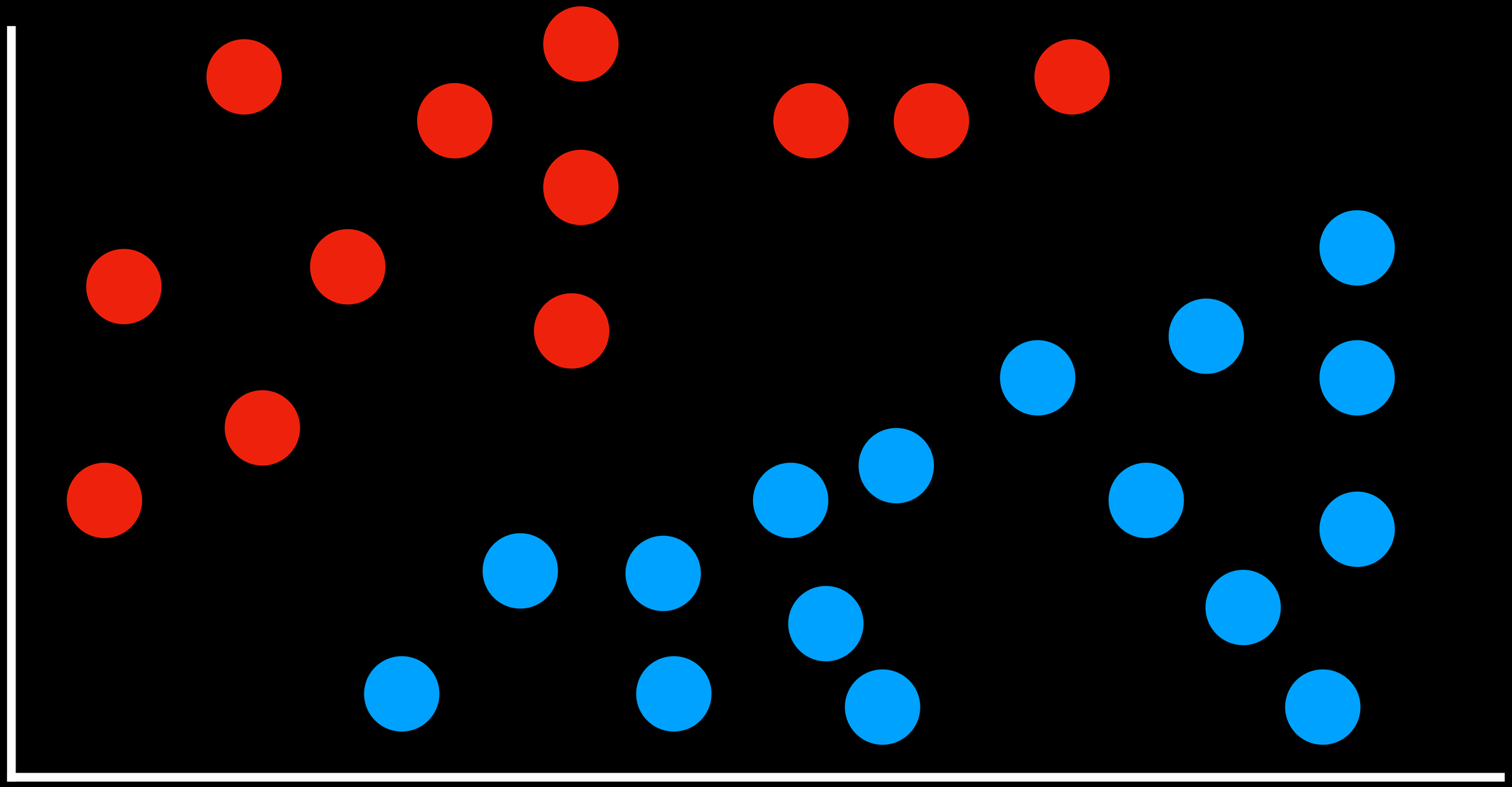


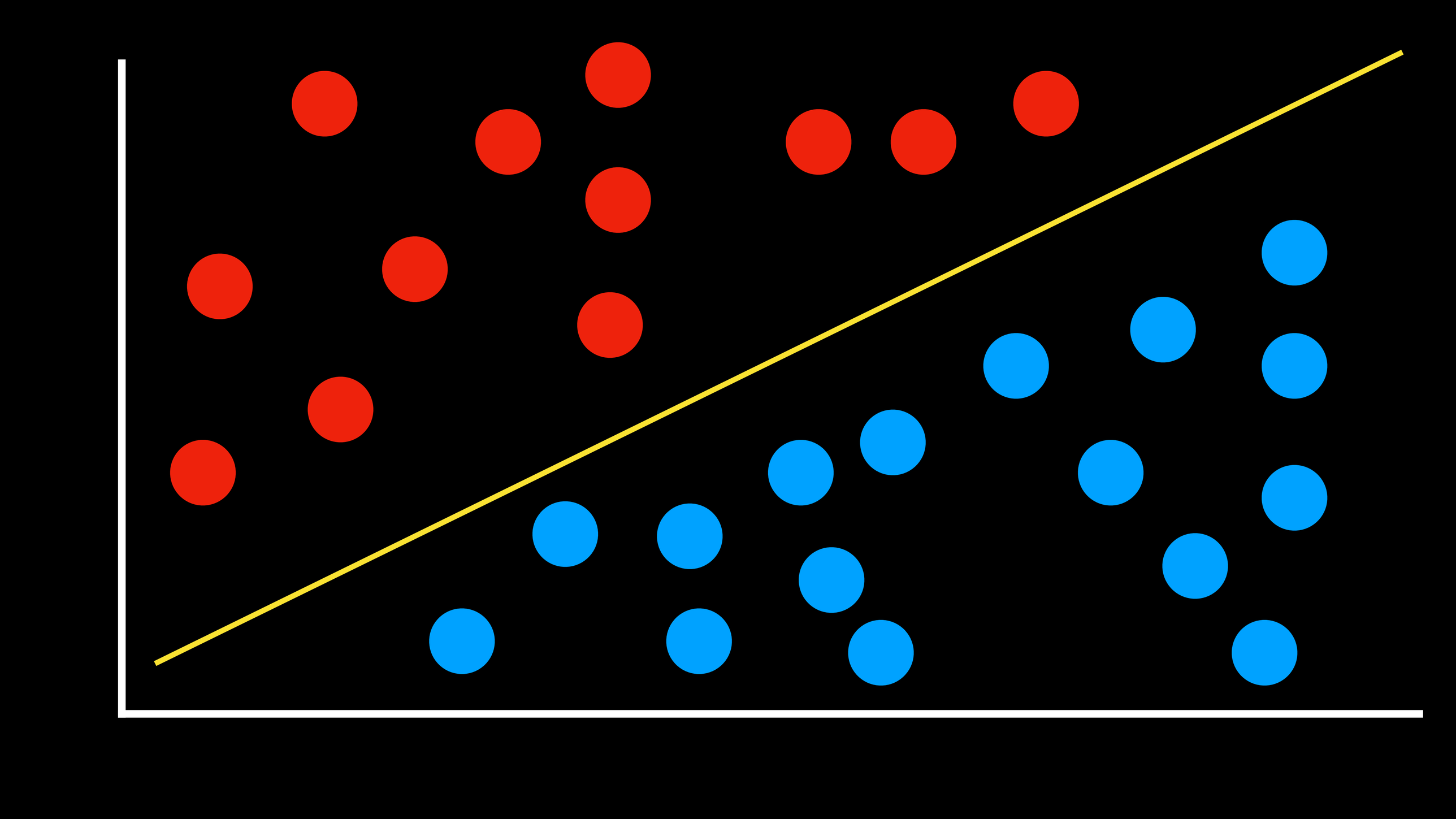


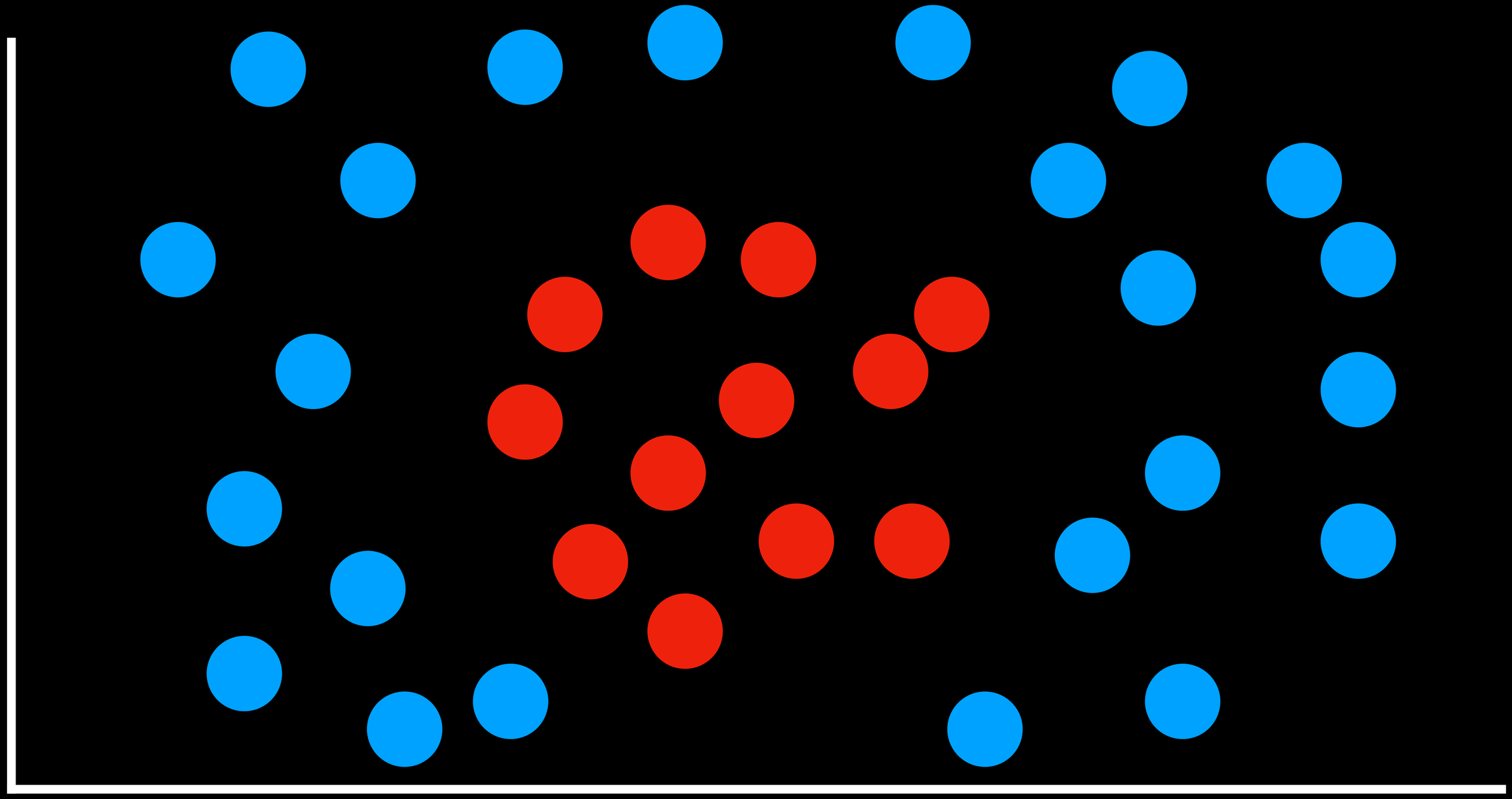










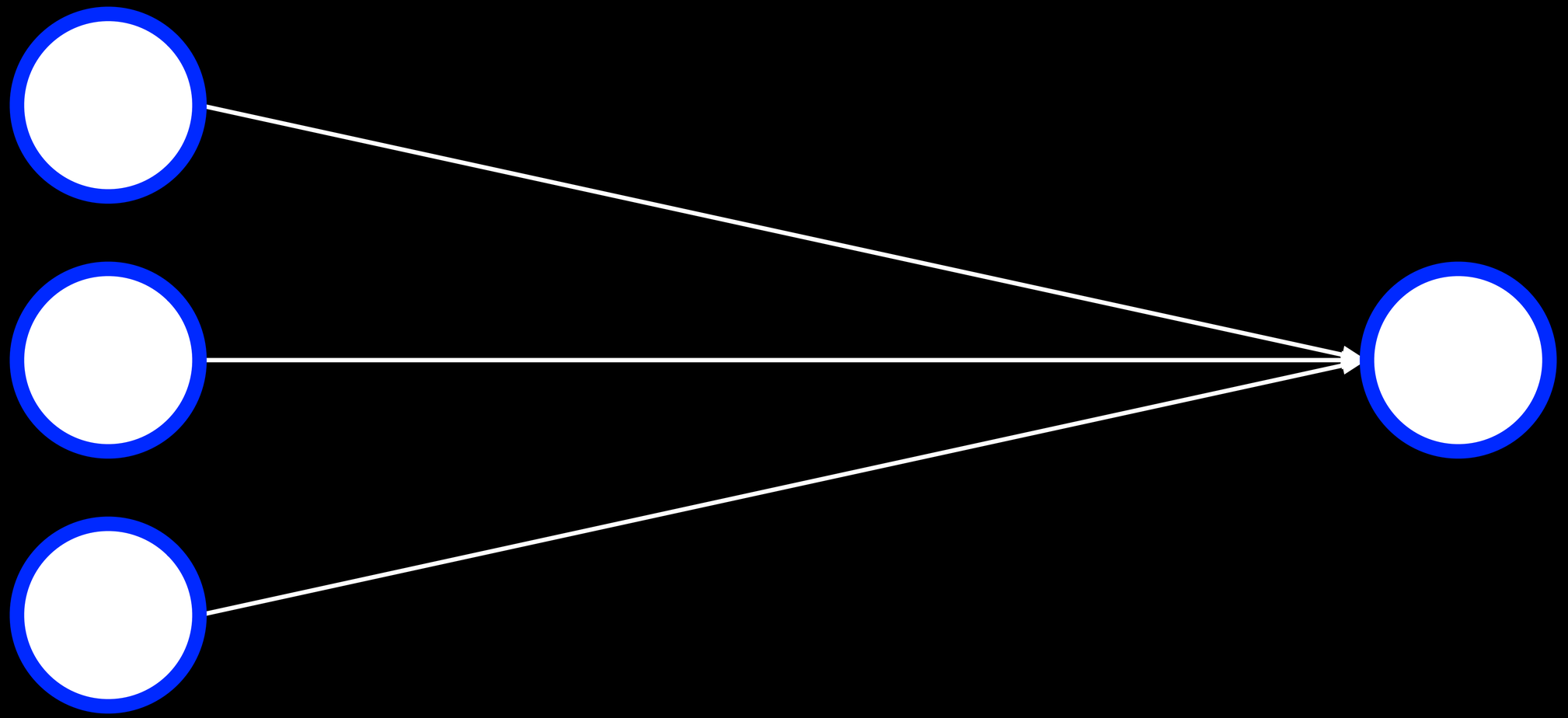


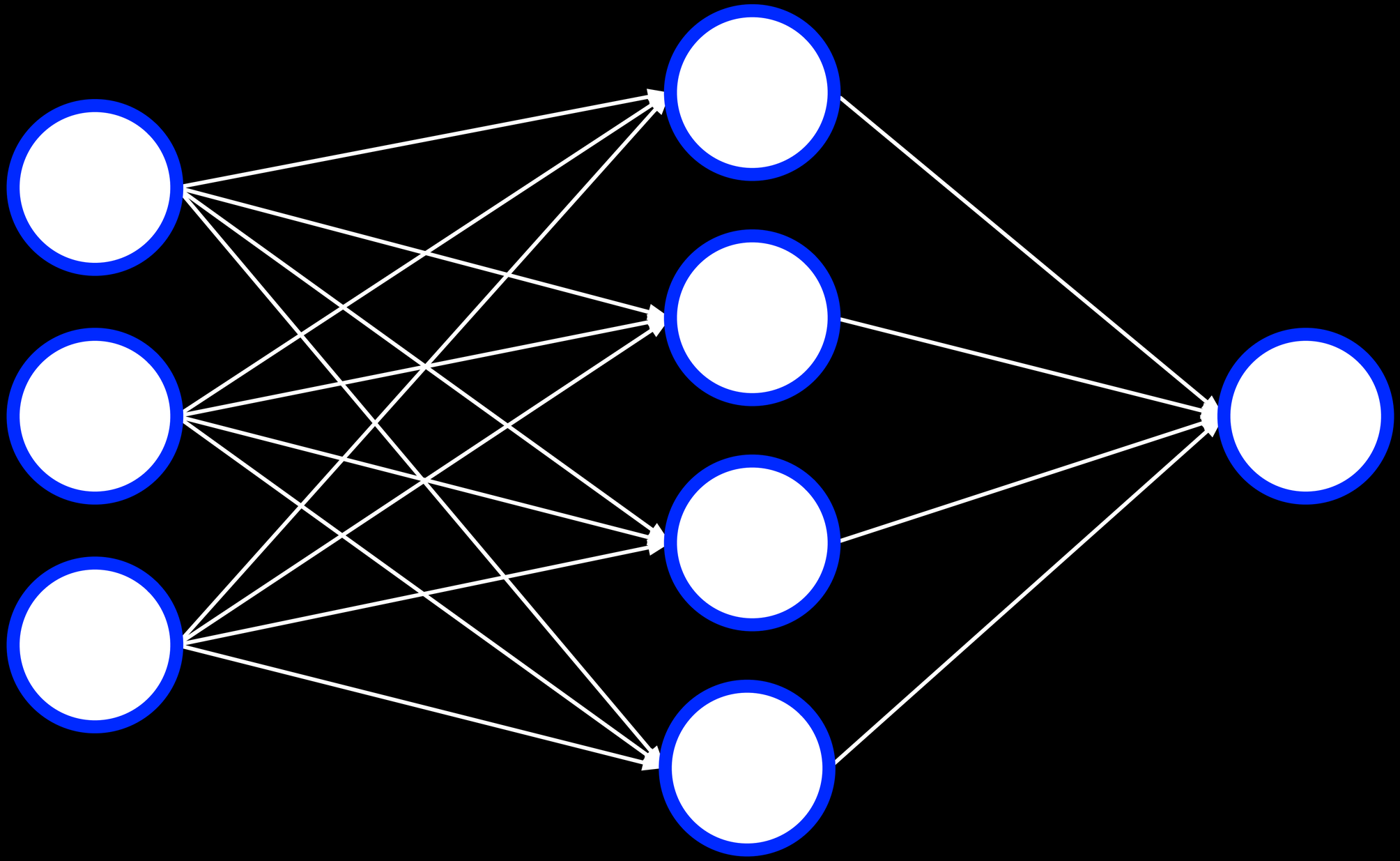
Perceptron

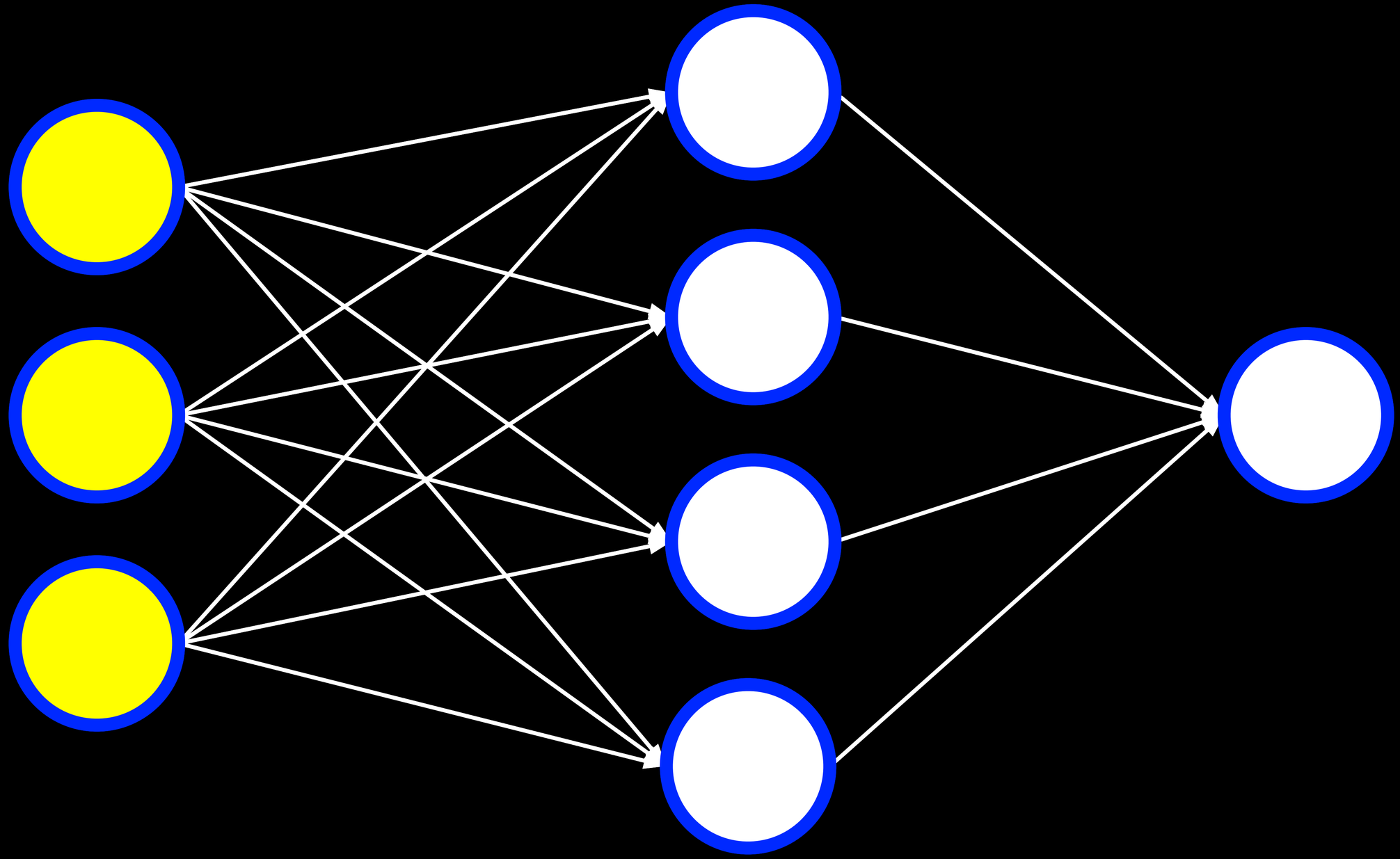
- Only capable of learning linearly separable decision boundary.

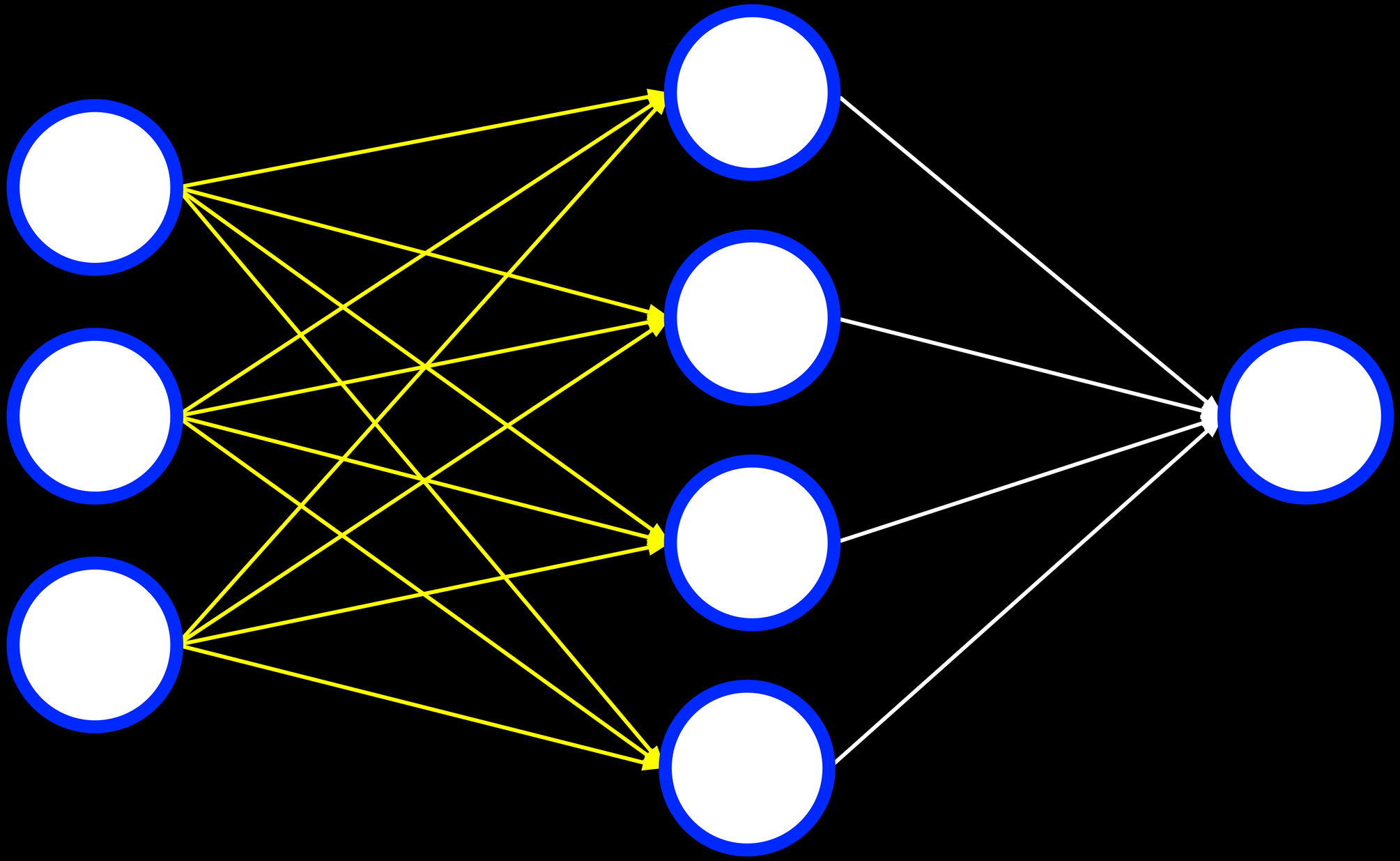
multilayer neural network

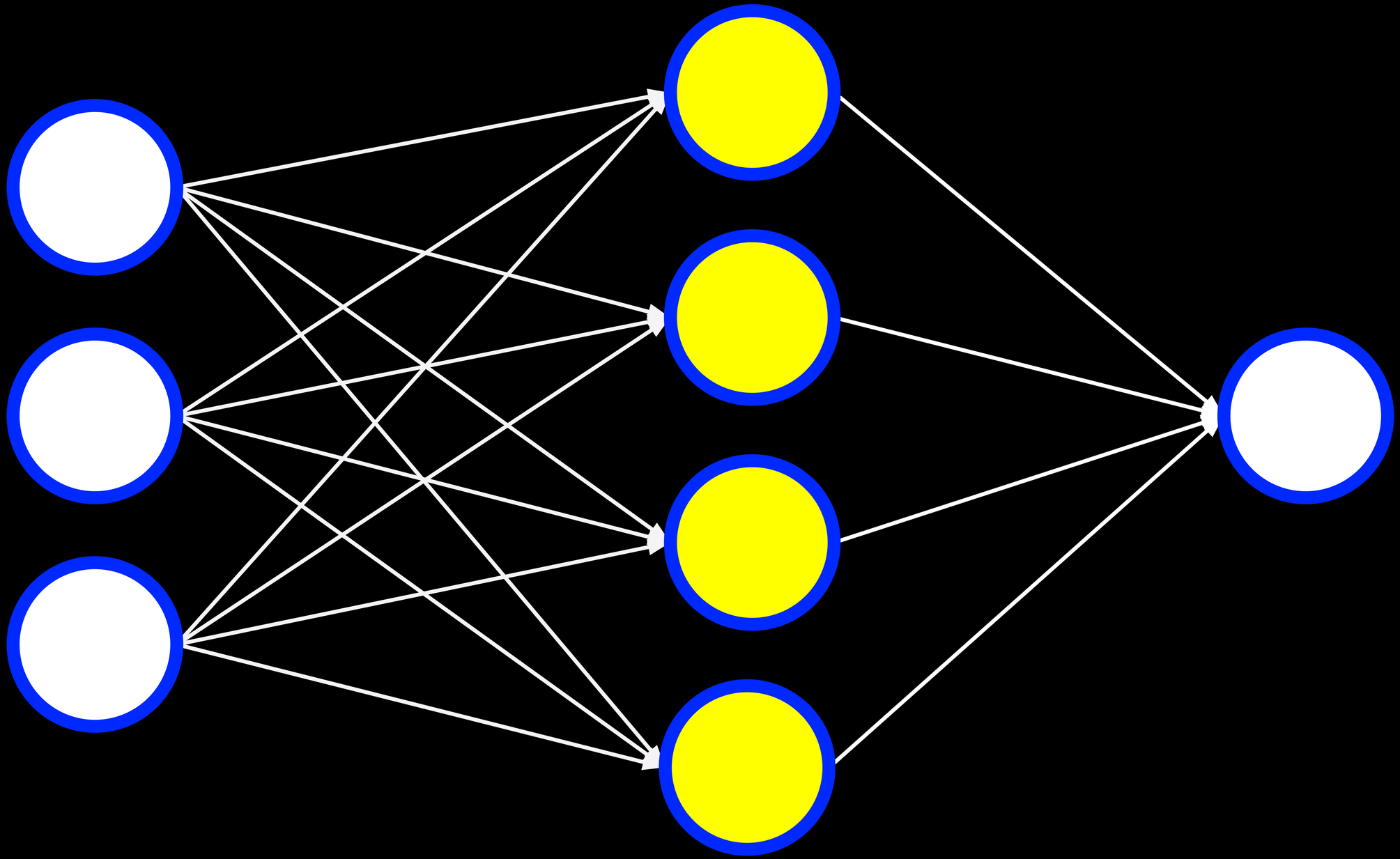
artificial neural network with an input layer,
an output layer, and at least one hidden layer

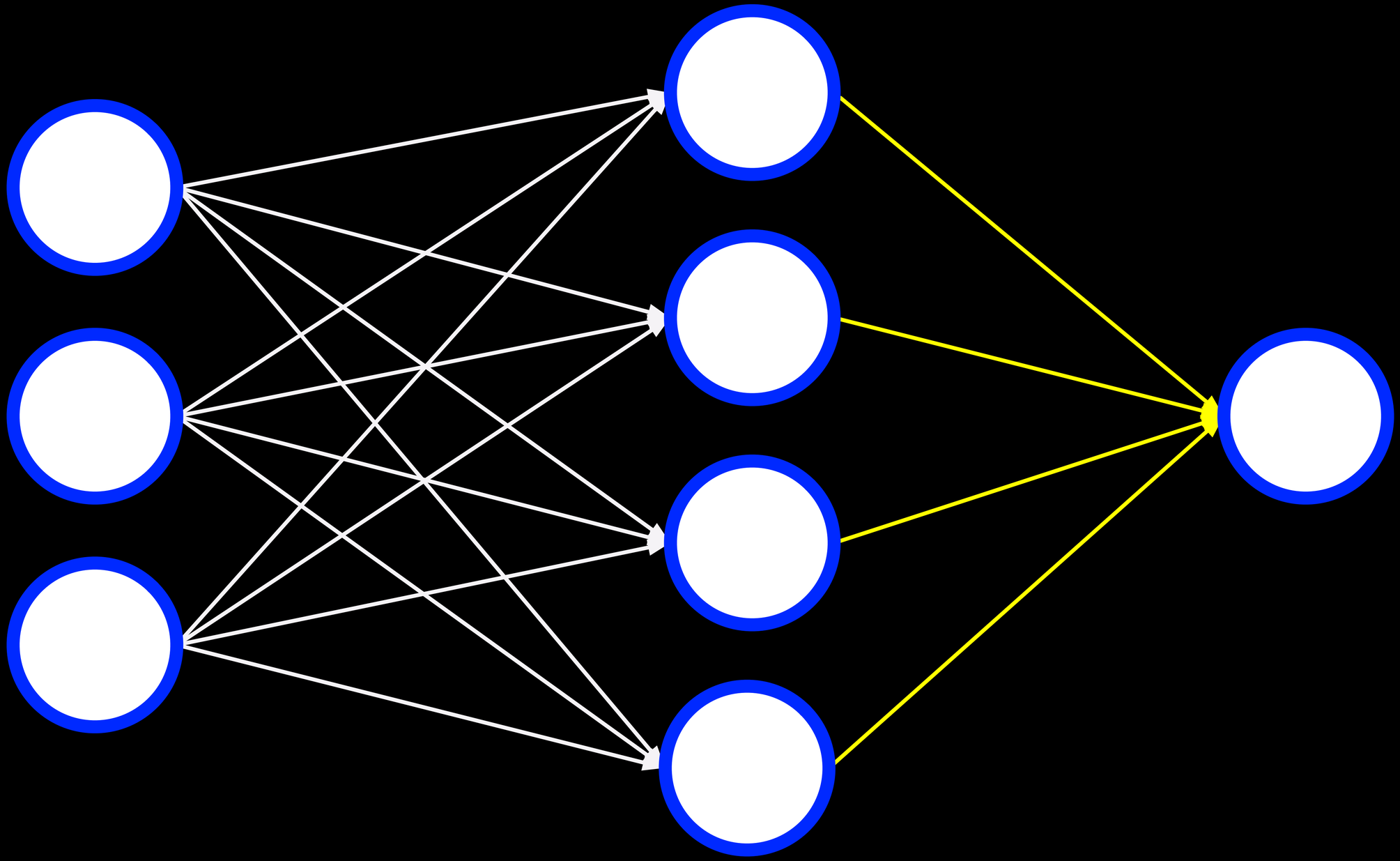


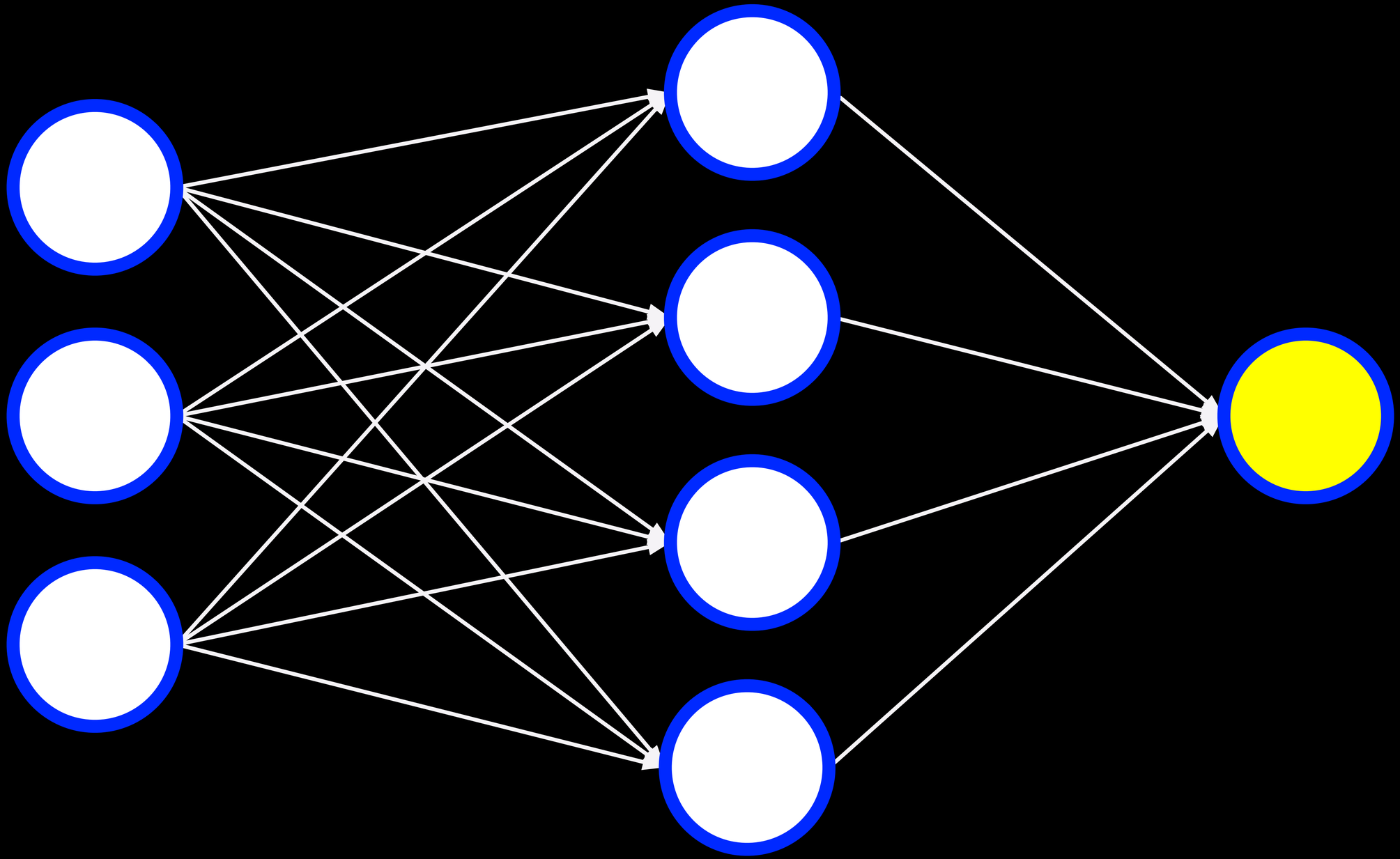


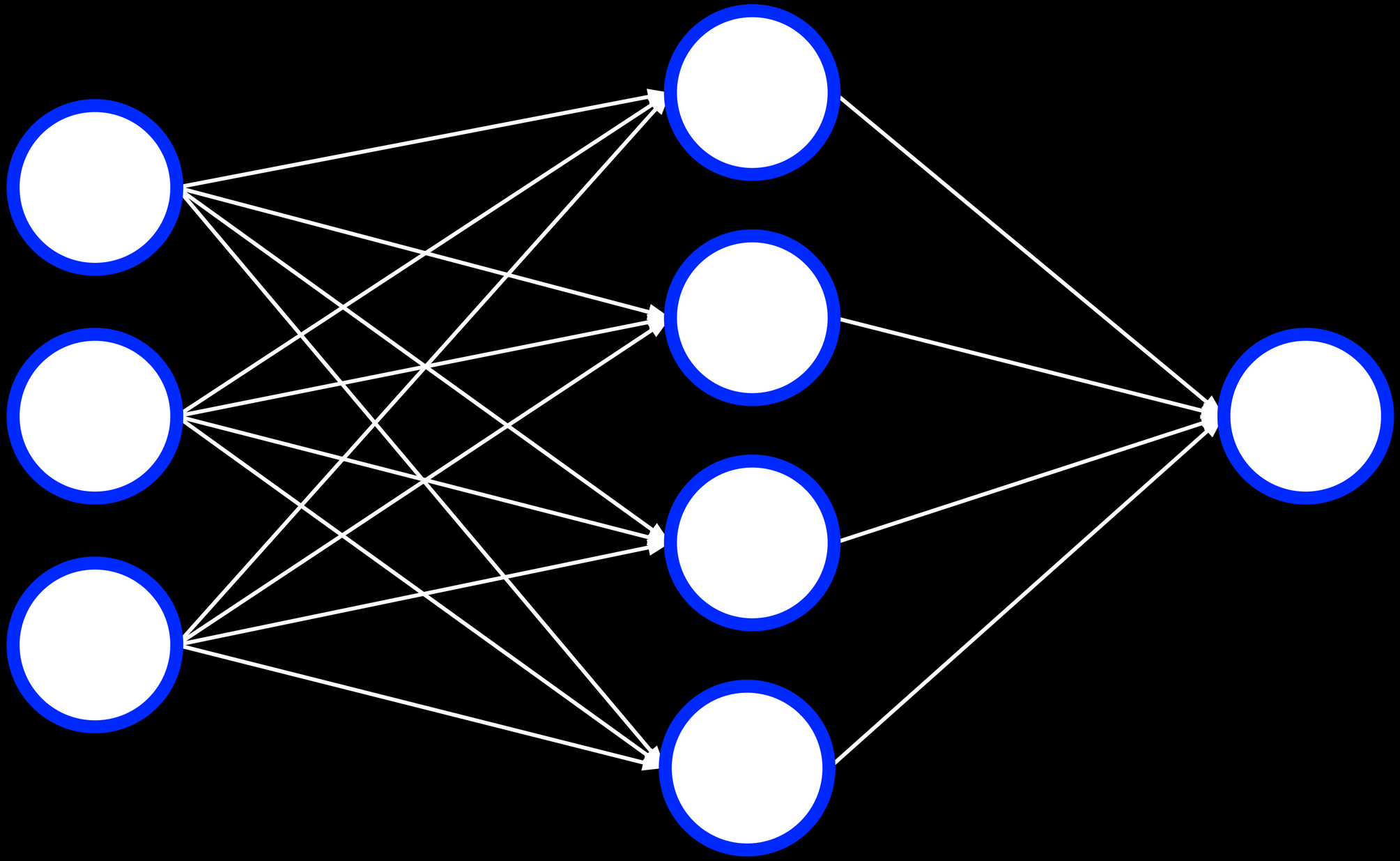










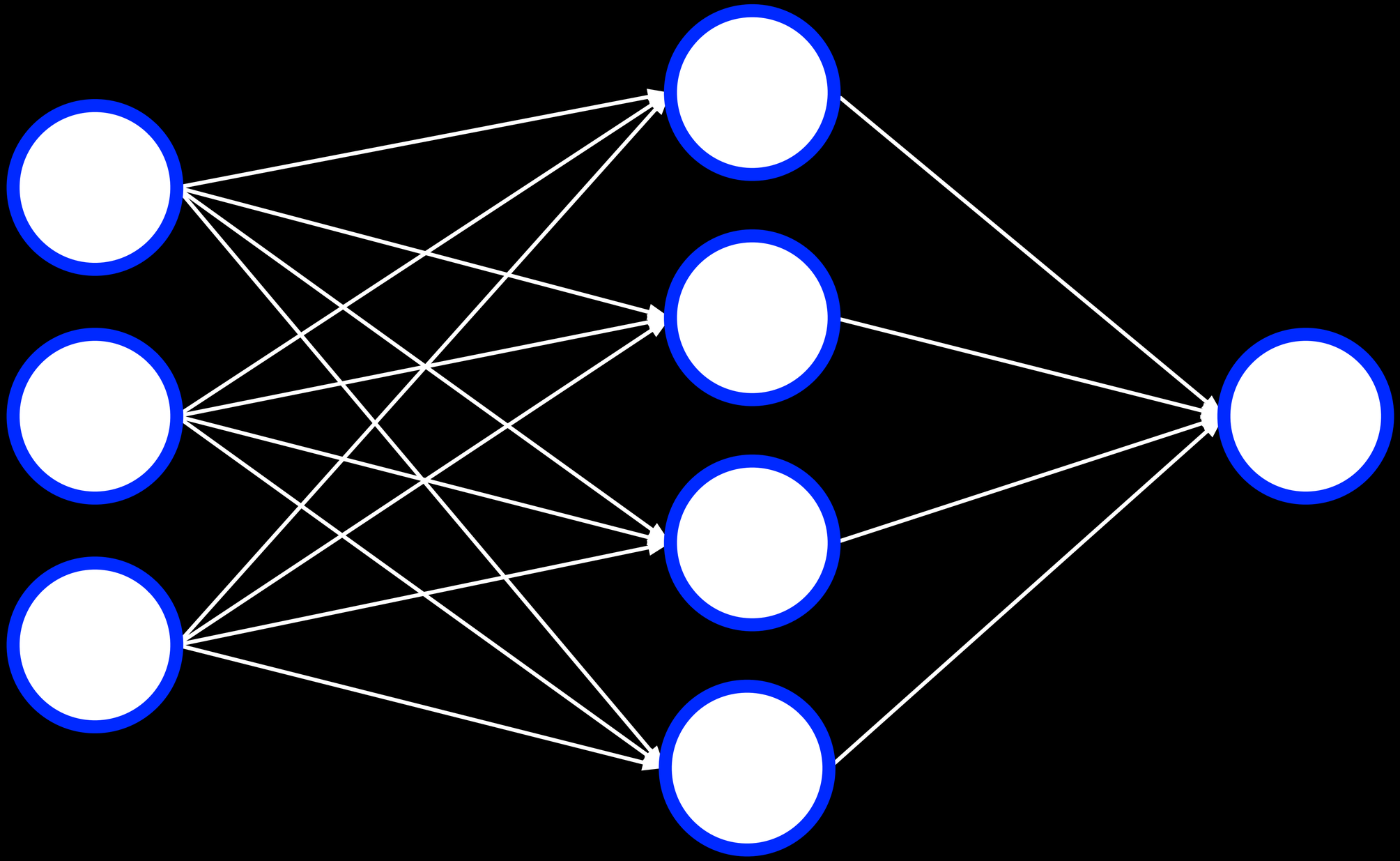


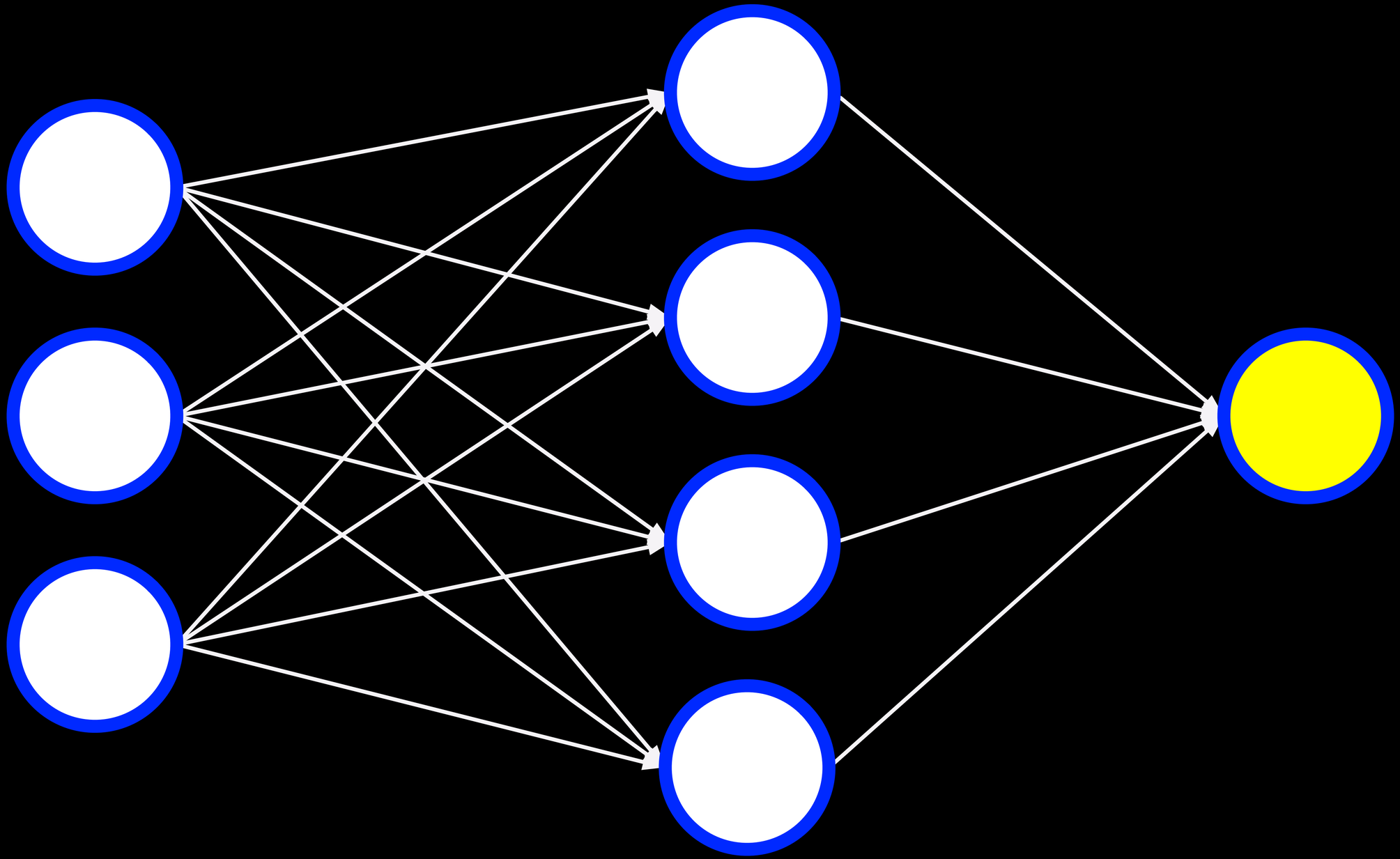
backpropagation

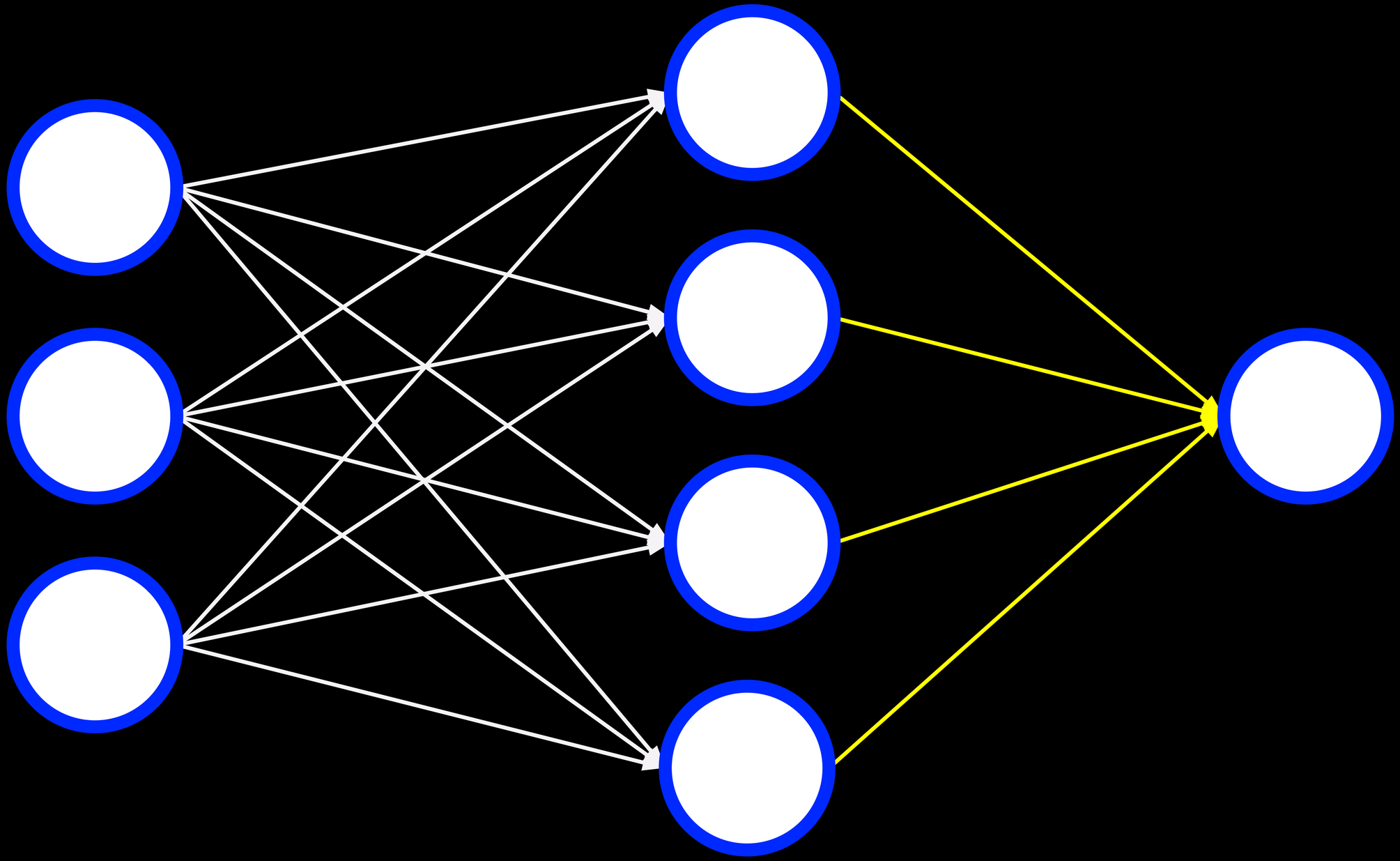
algorithm for training neural networks with hidden layers

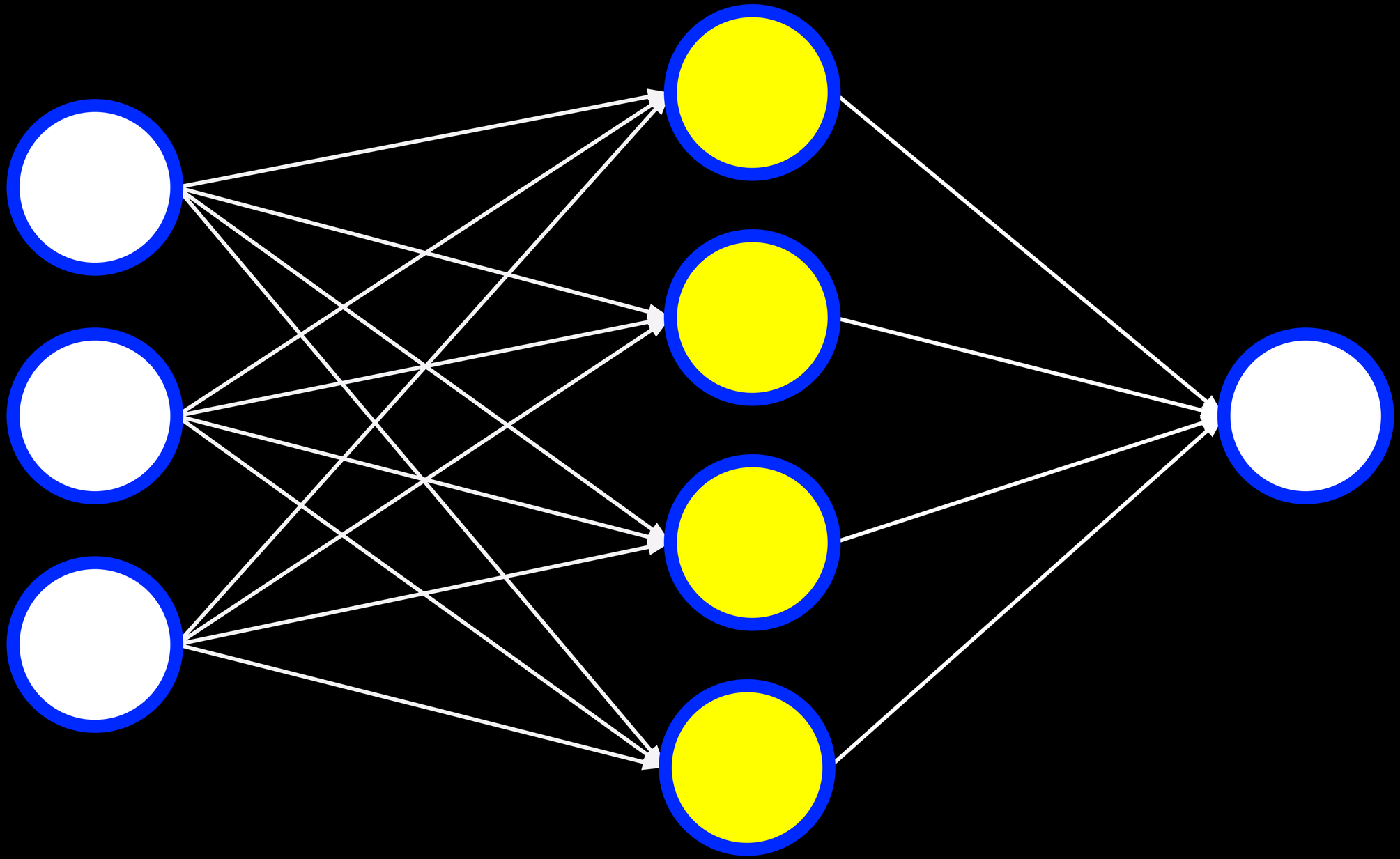
Backpropagation

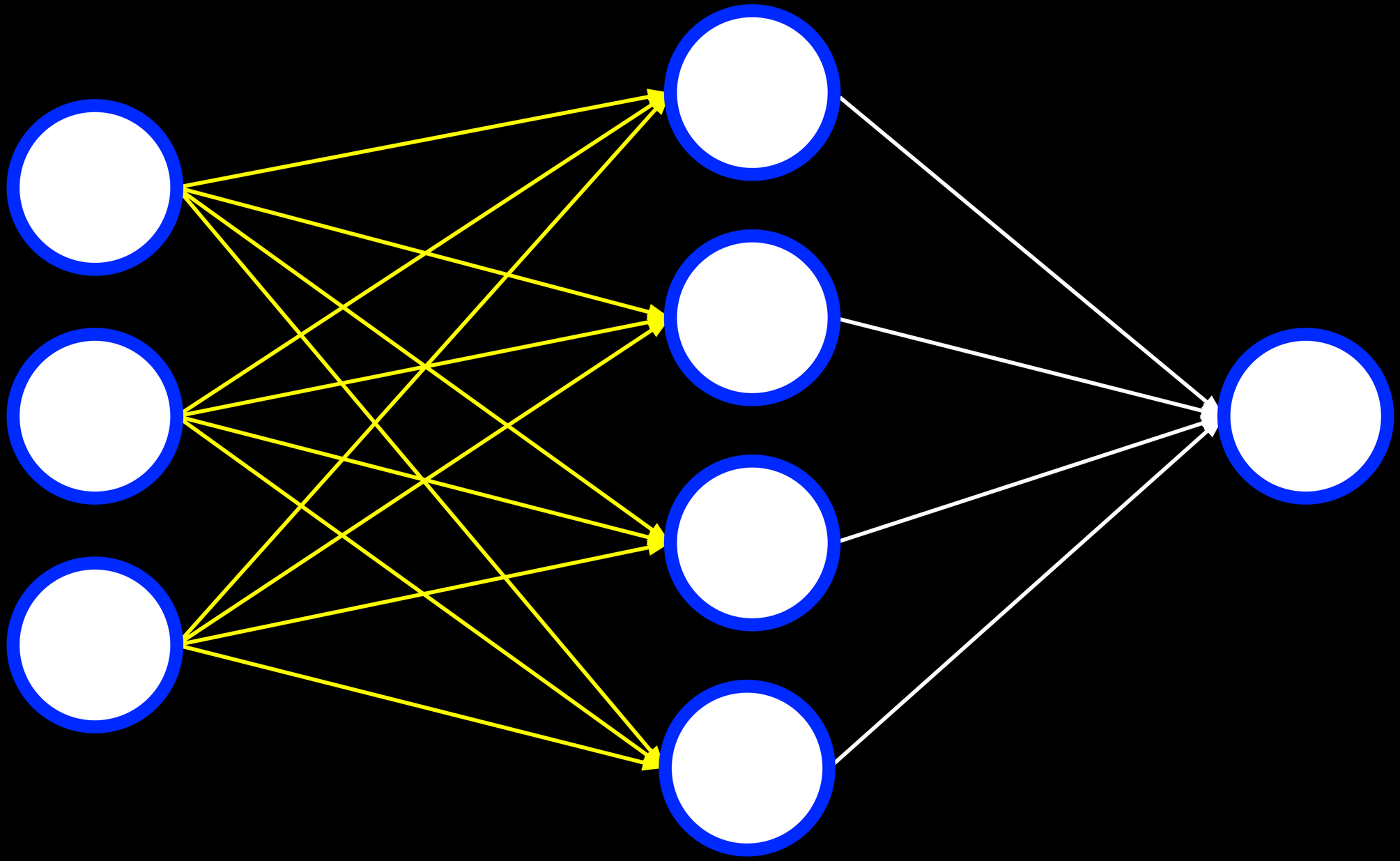
- Start with a random choice of weights.
- Repeat:
    - Calculate error for output layer.
    - For each layer, starting with output layer, and moving inwards towards earliest hidden layer:
        - Propagate error back one layer.
        - Update weights.

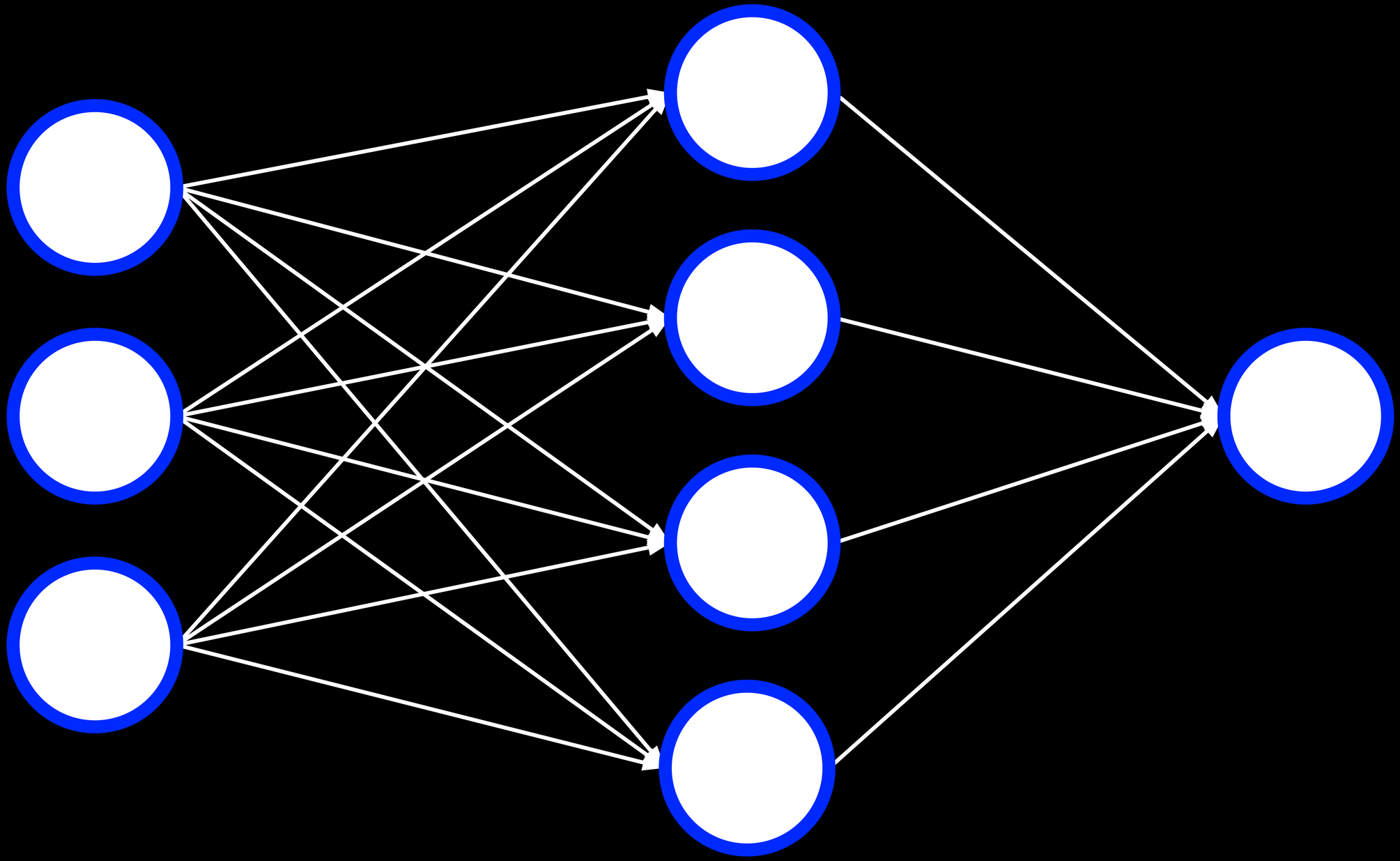


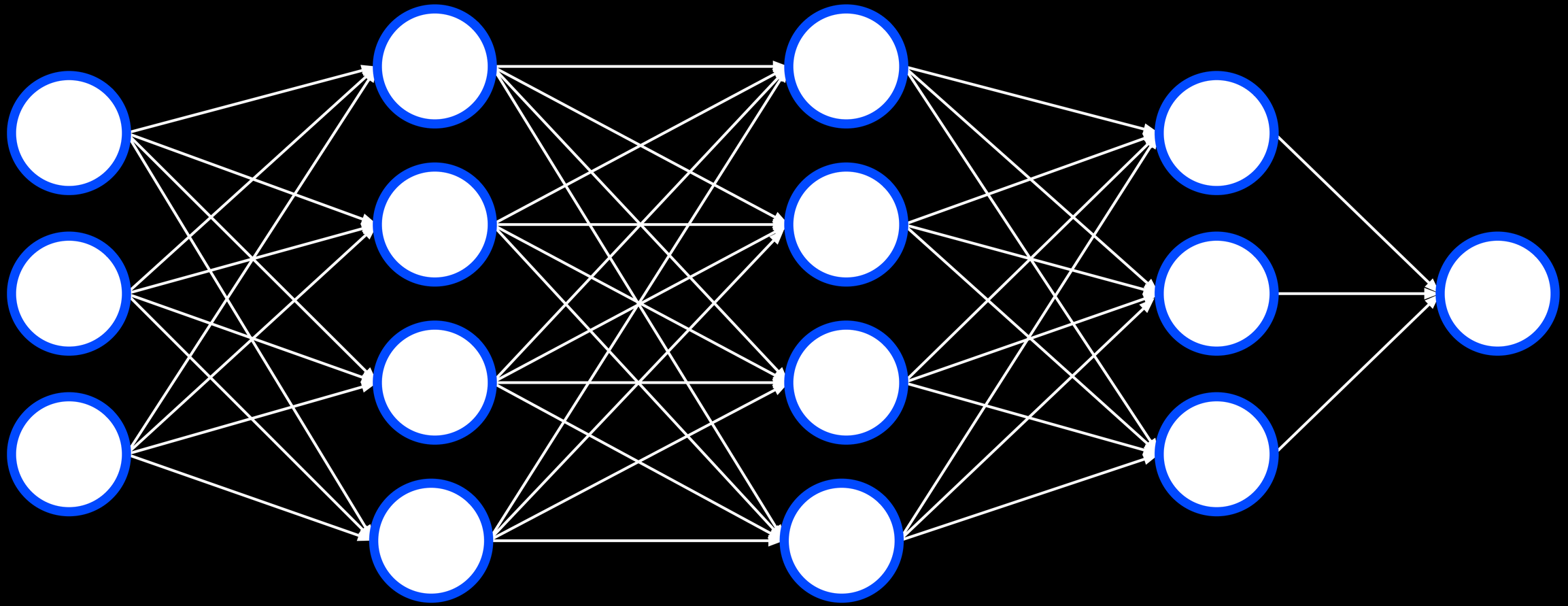






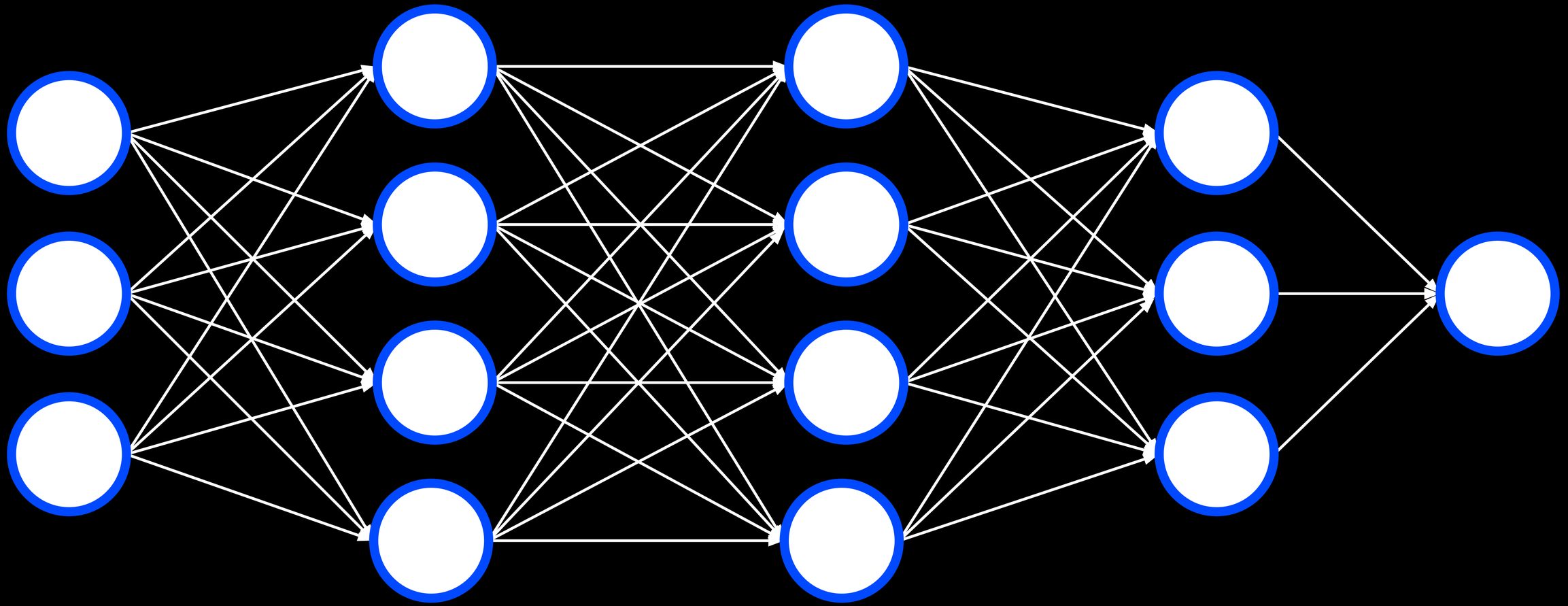






deep neural networks

neural network with multiple hidden layers

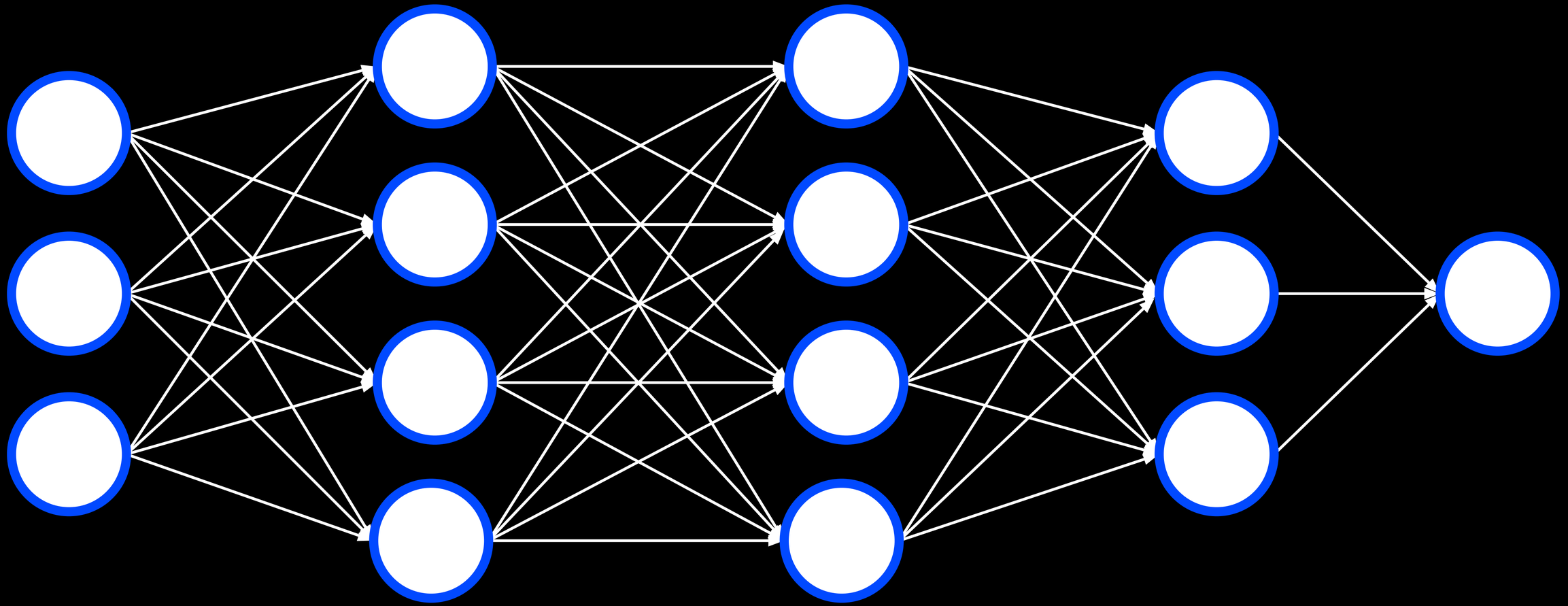


Overfitting

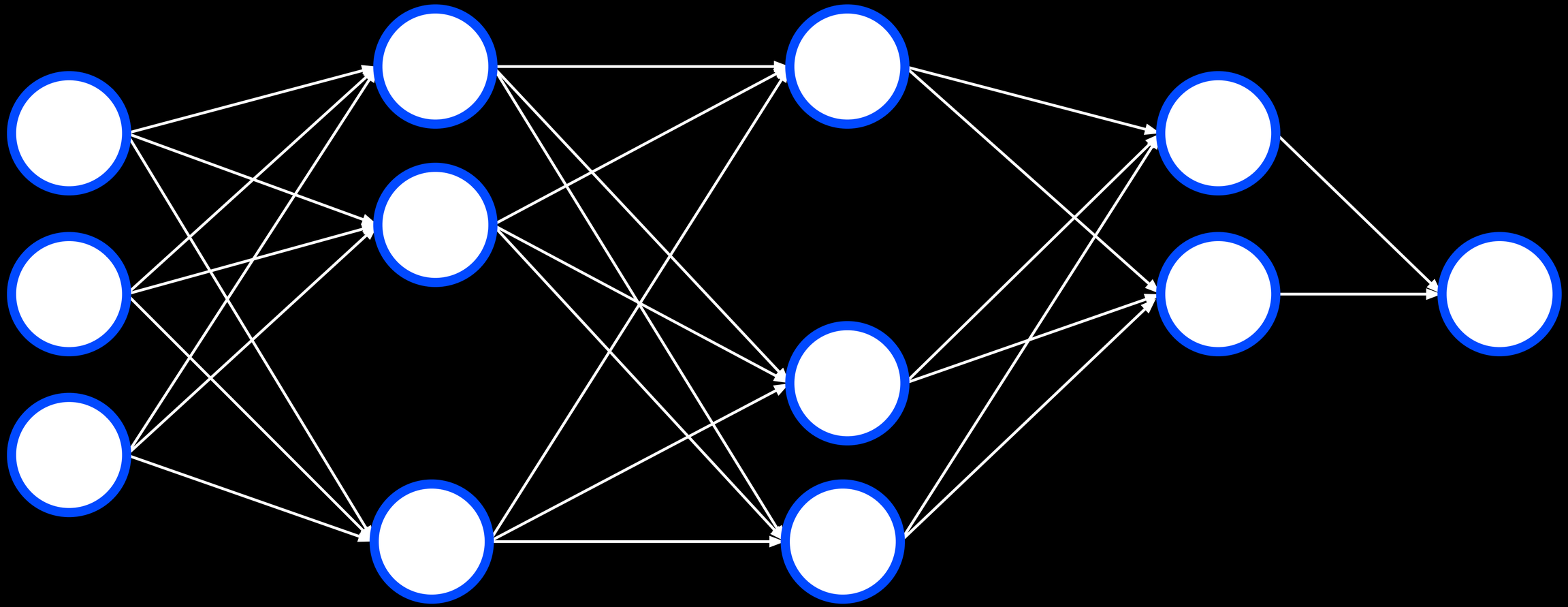
dropout

temporarily removing units — selected at random — from a neural network to prevent over-reliance on certain units

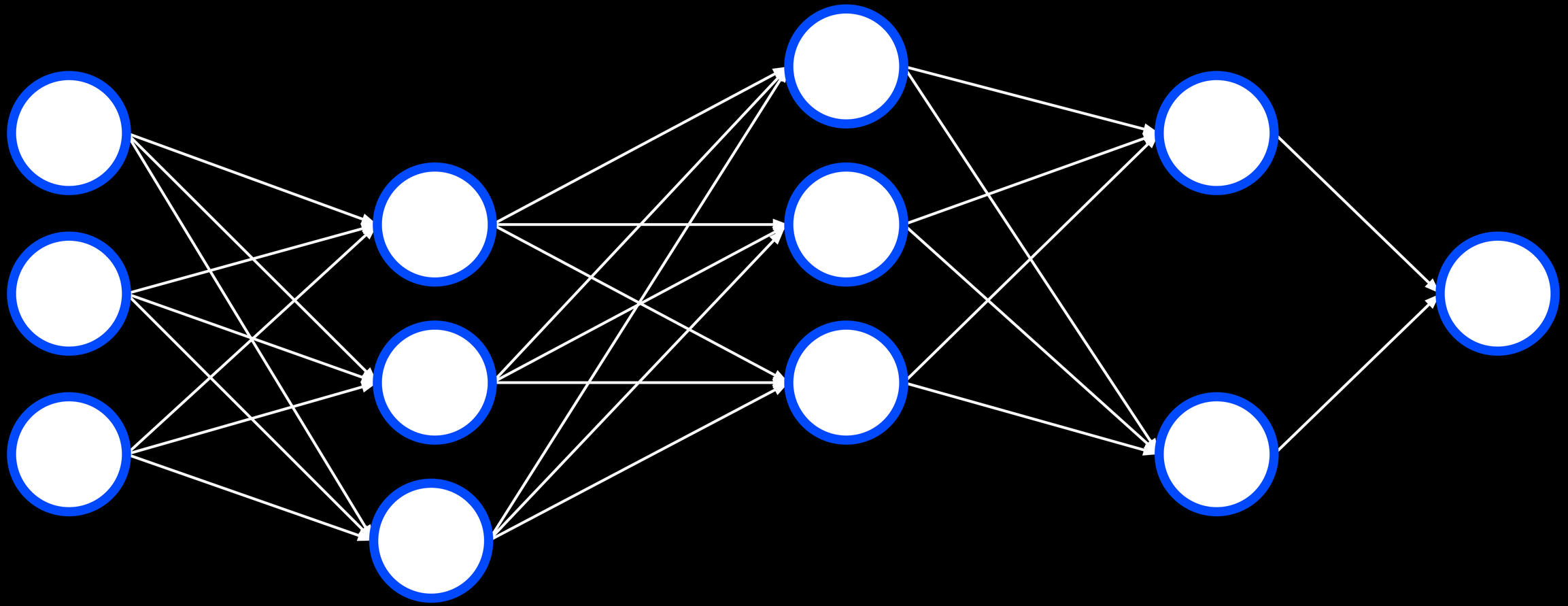
complete neural network
calculate the weights

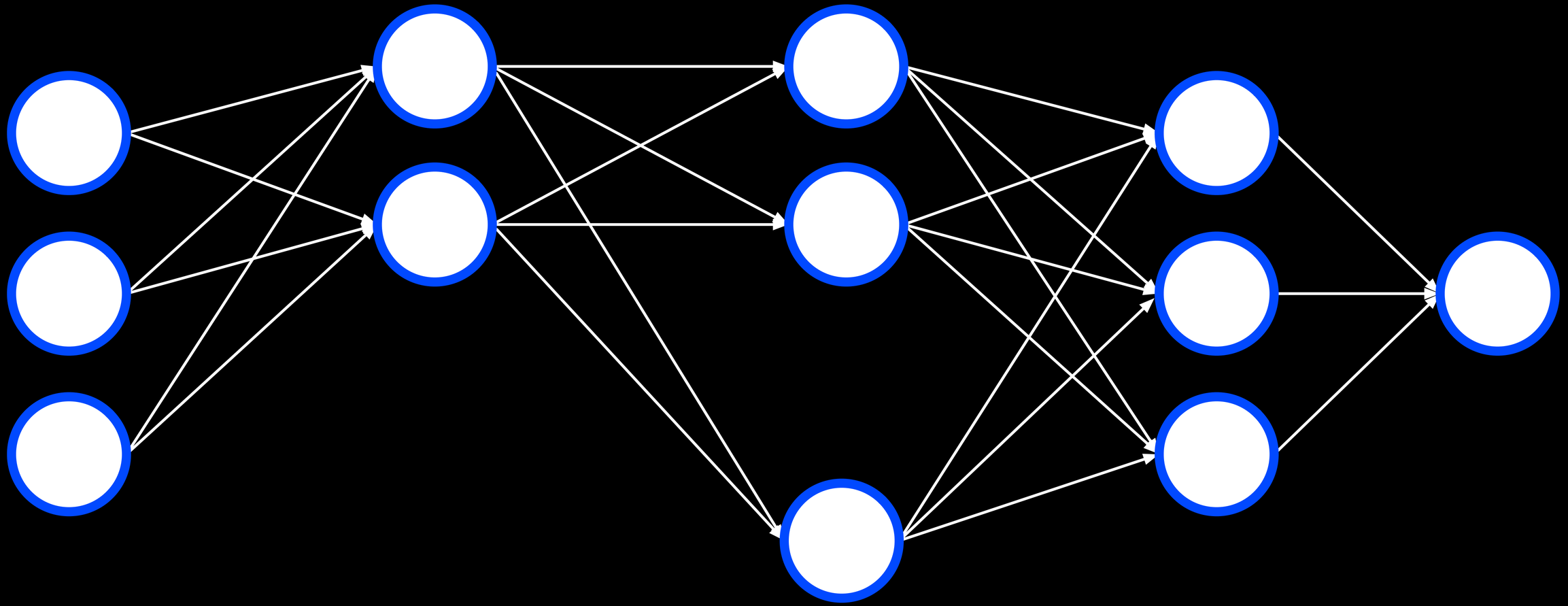


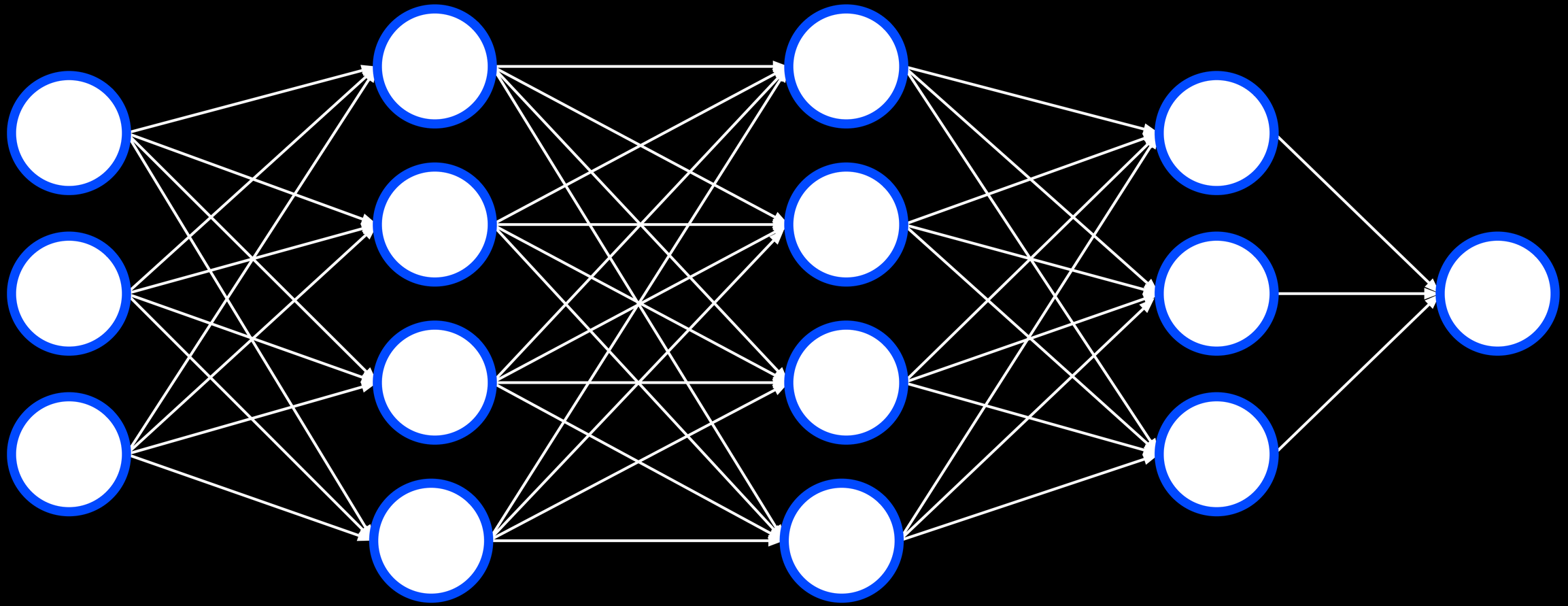
then randomly drop some fraction of inner nodes,
recalculate the weights



repeat random drop from complete NN,
recalculate weights





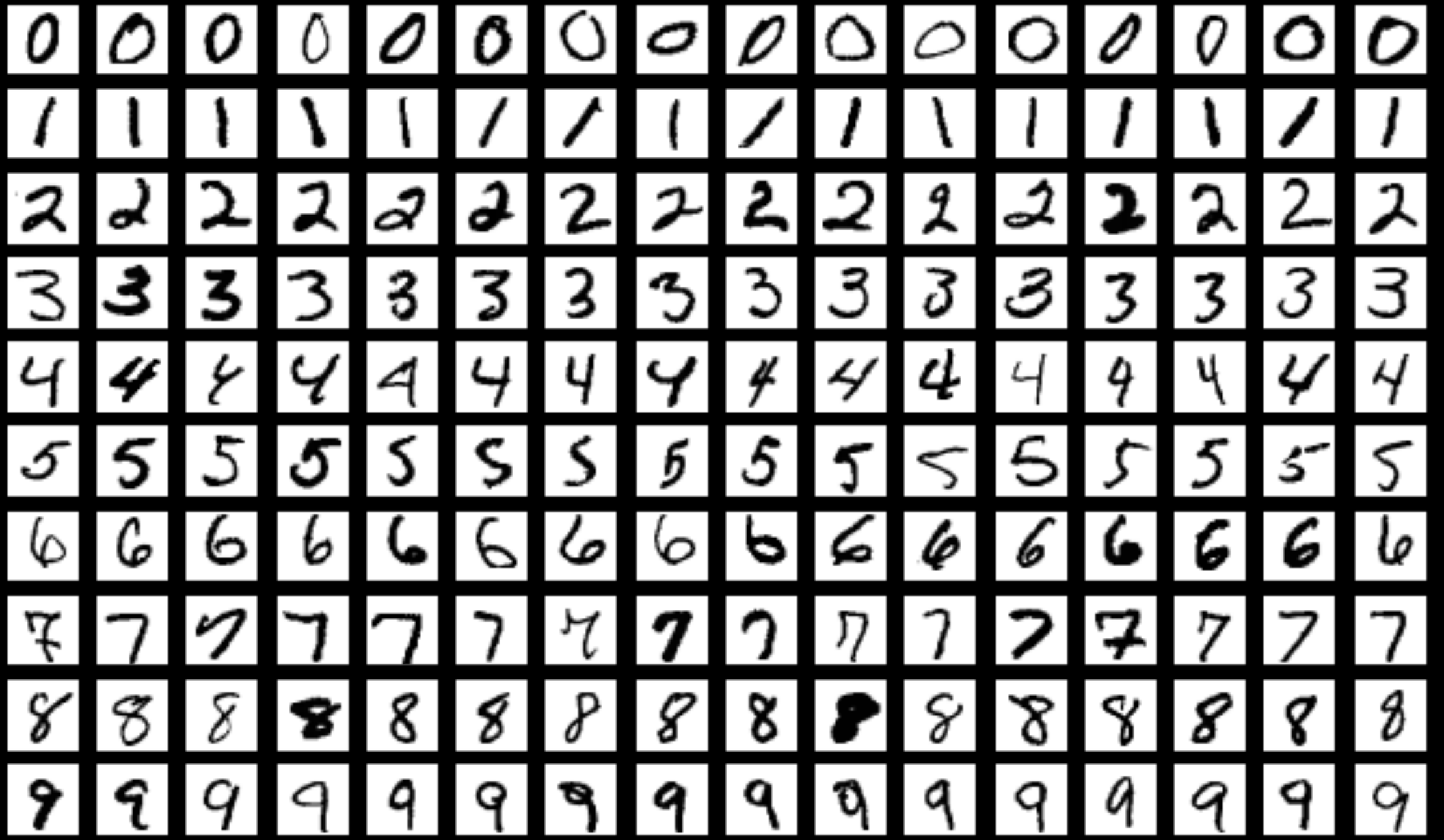


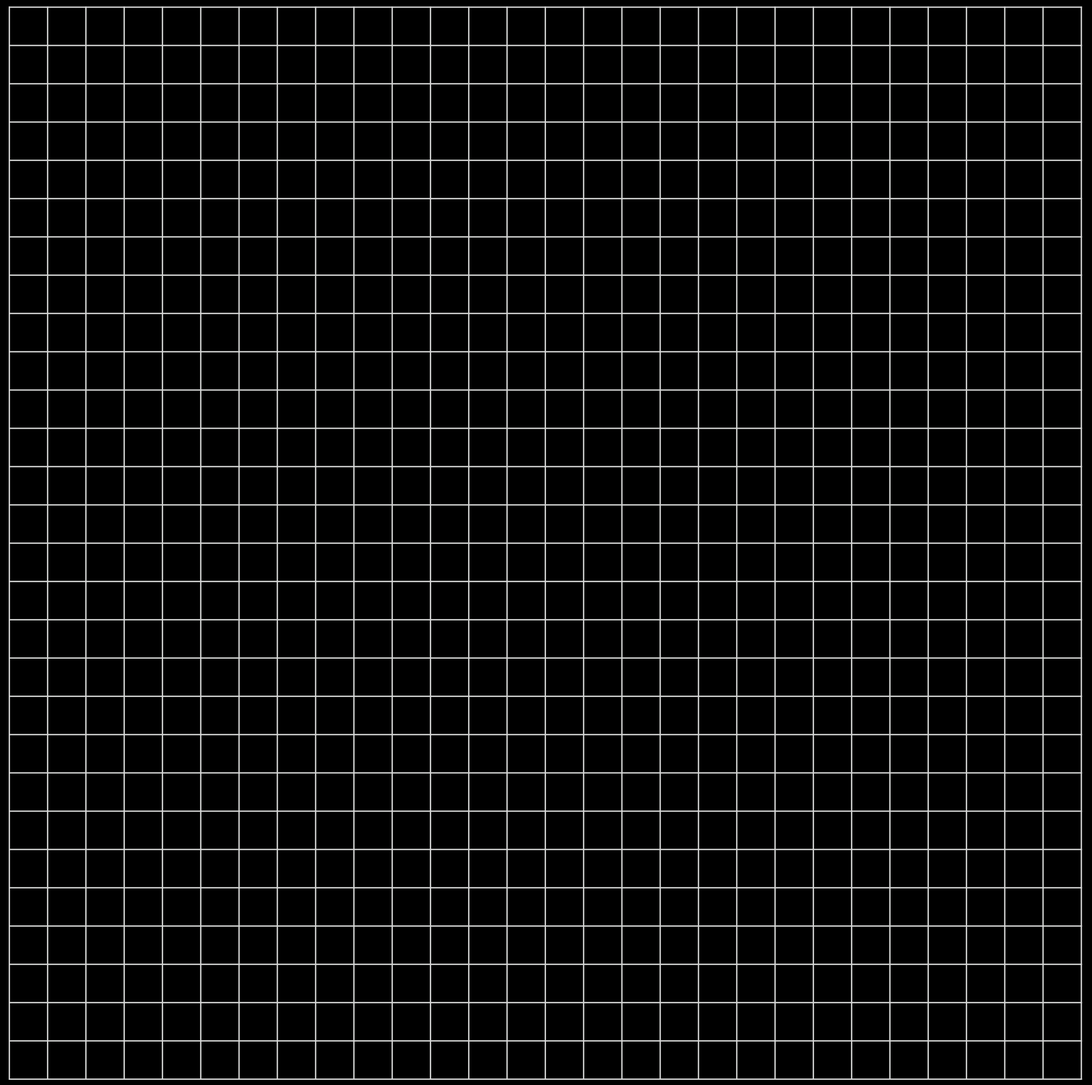
TensorFlow

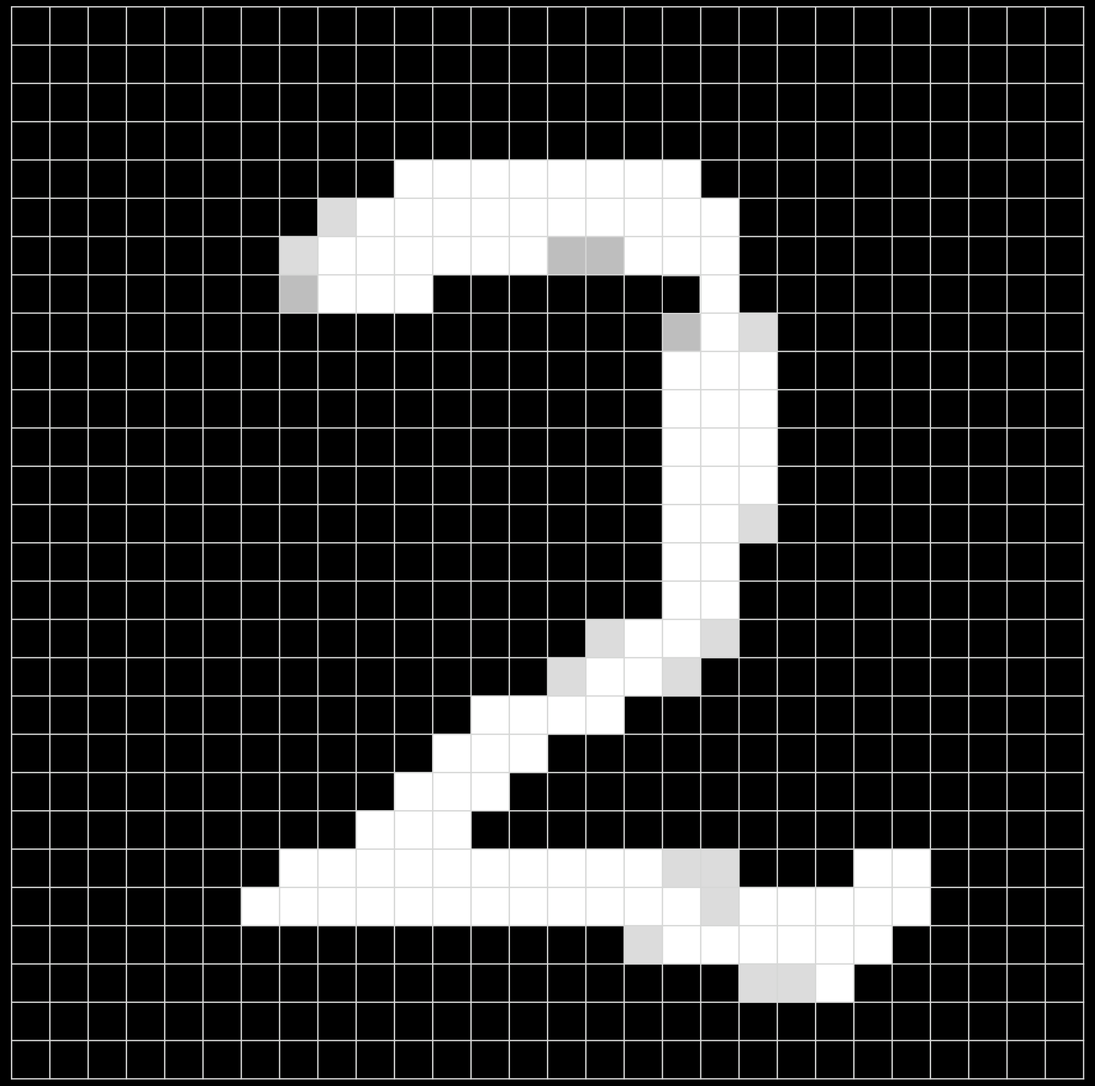
playground.tensorflow.org

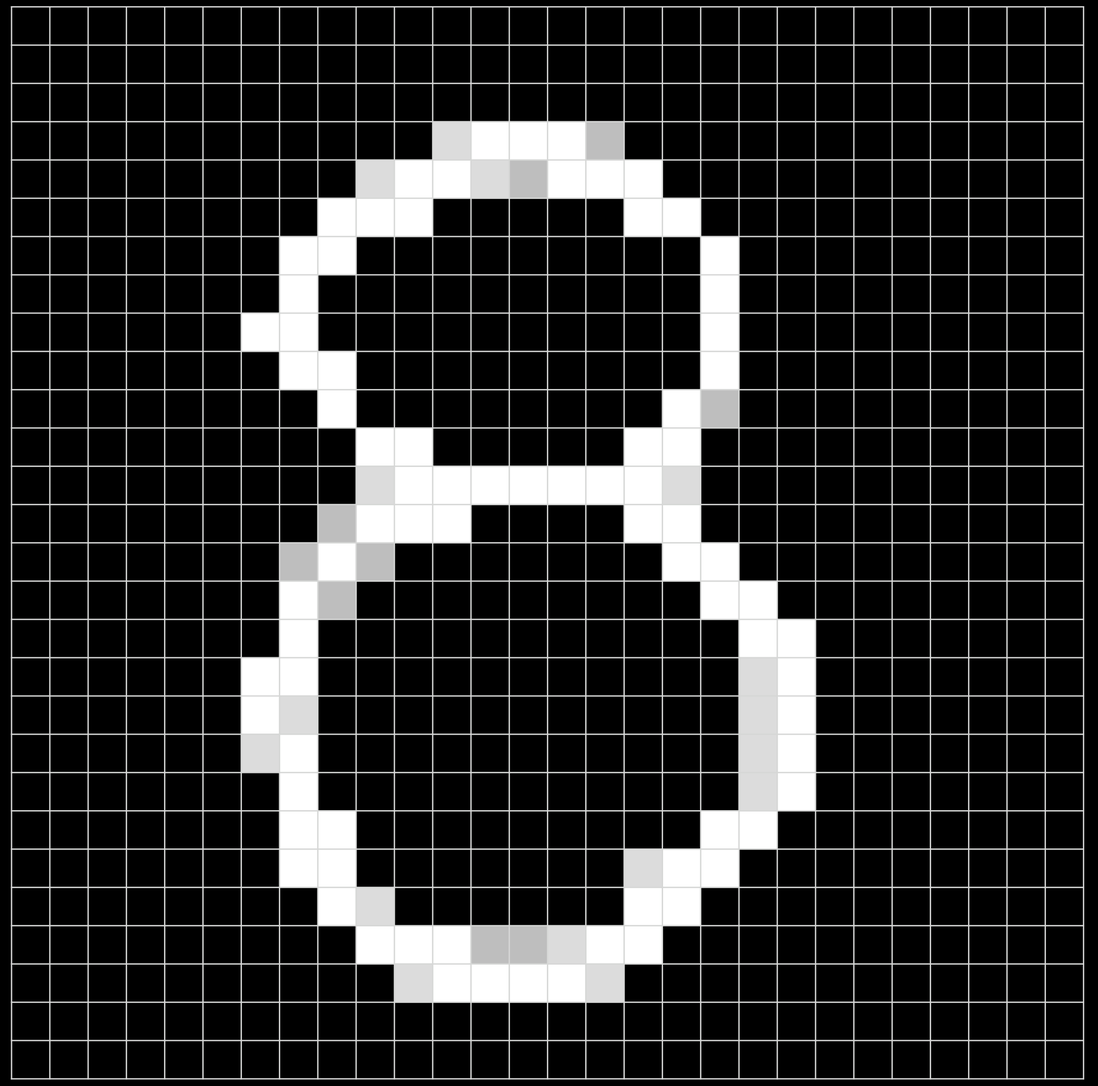
computer vision

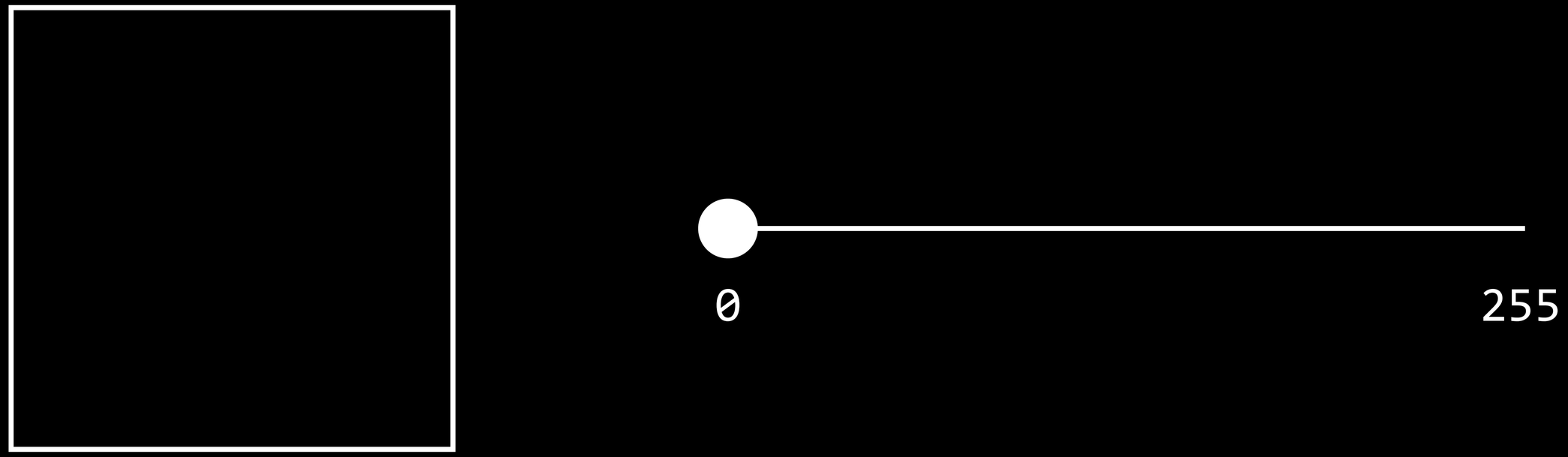
computational methods for analyzing and understanding digital images

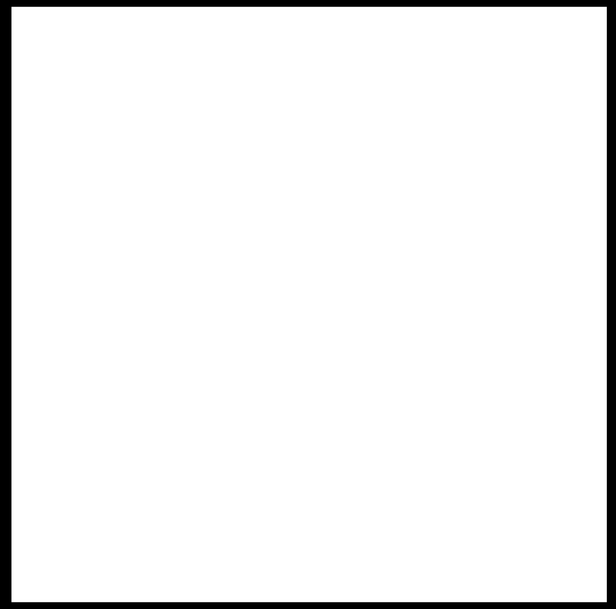




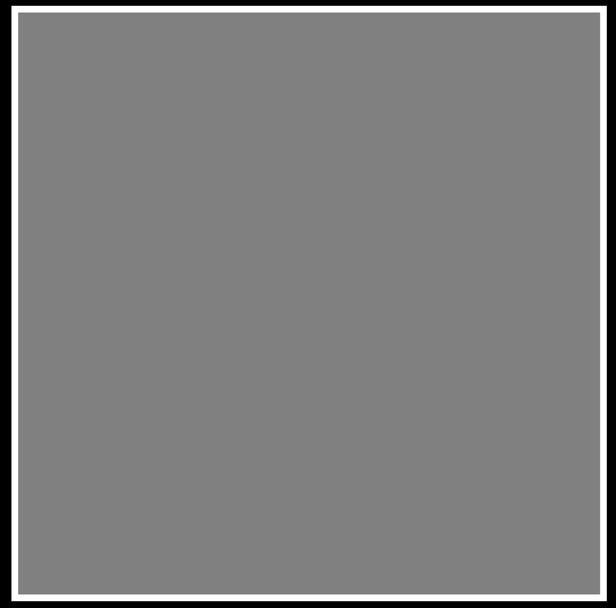


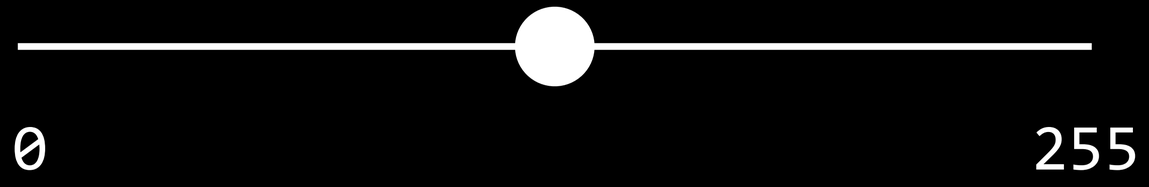


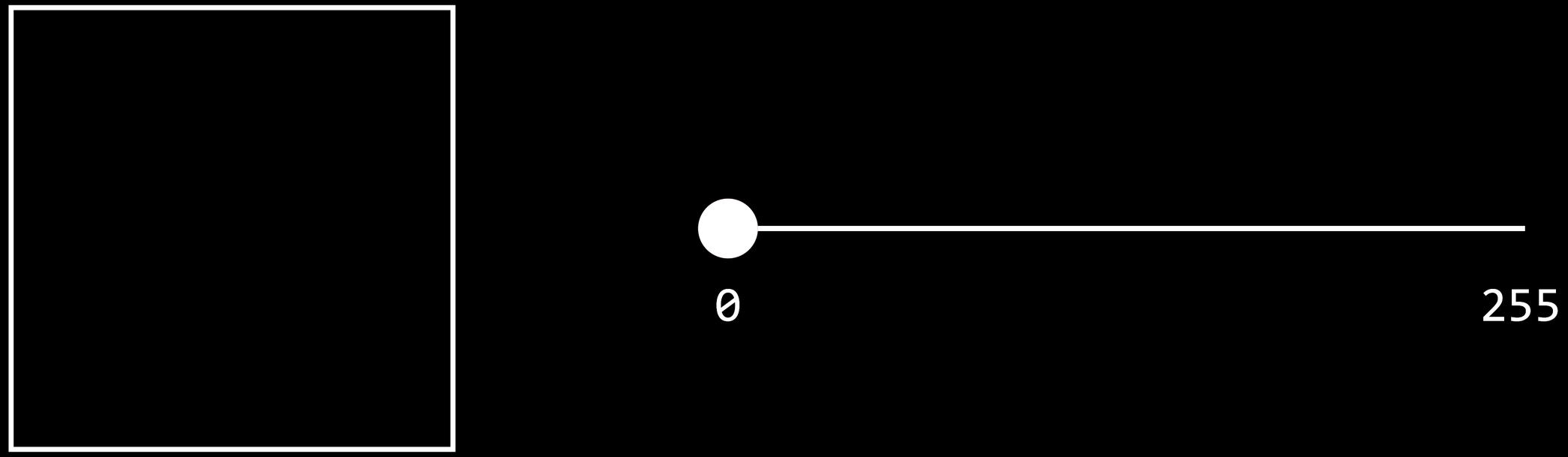

0
255

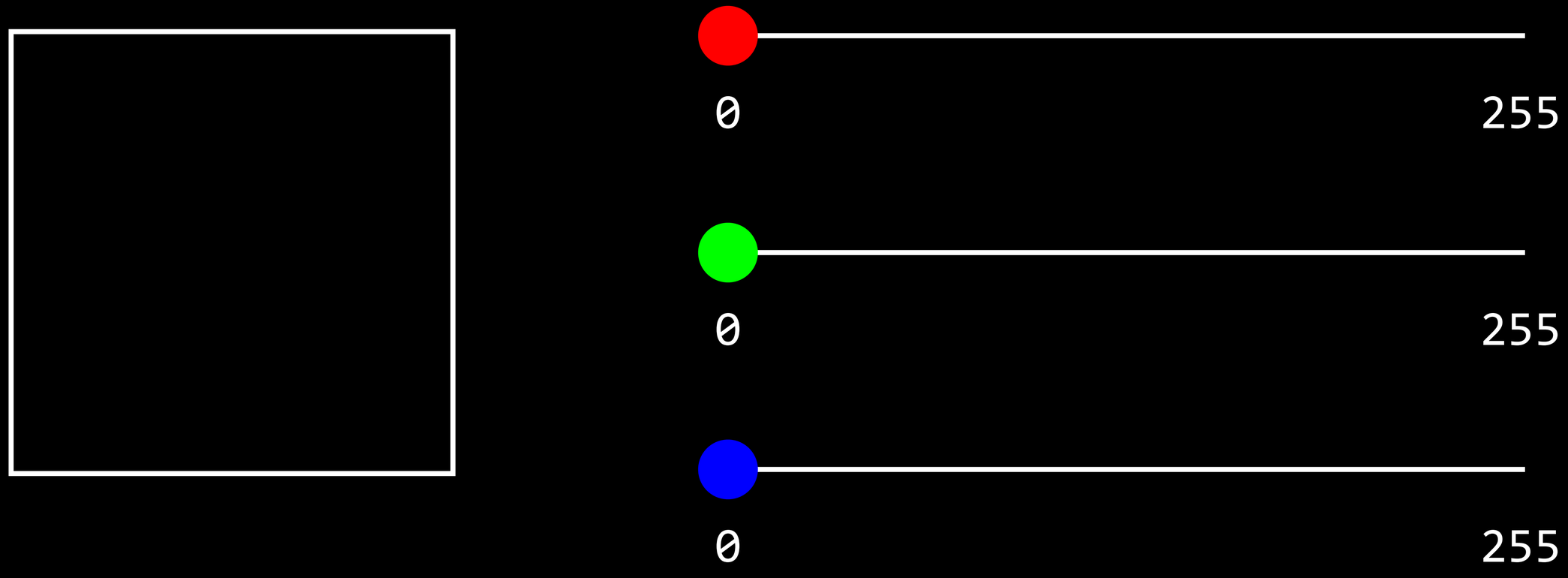


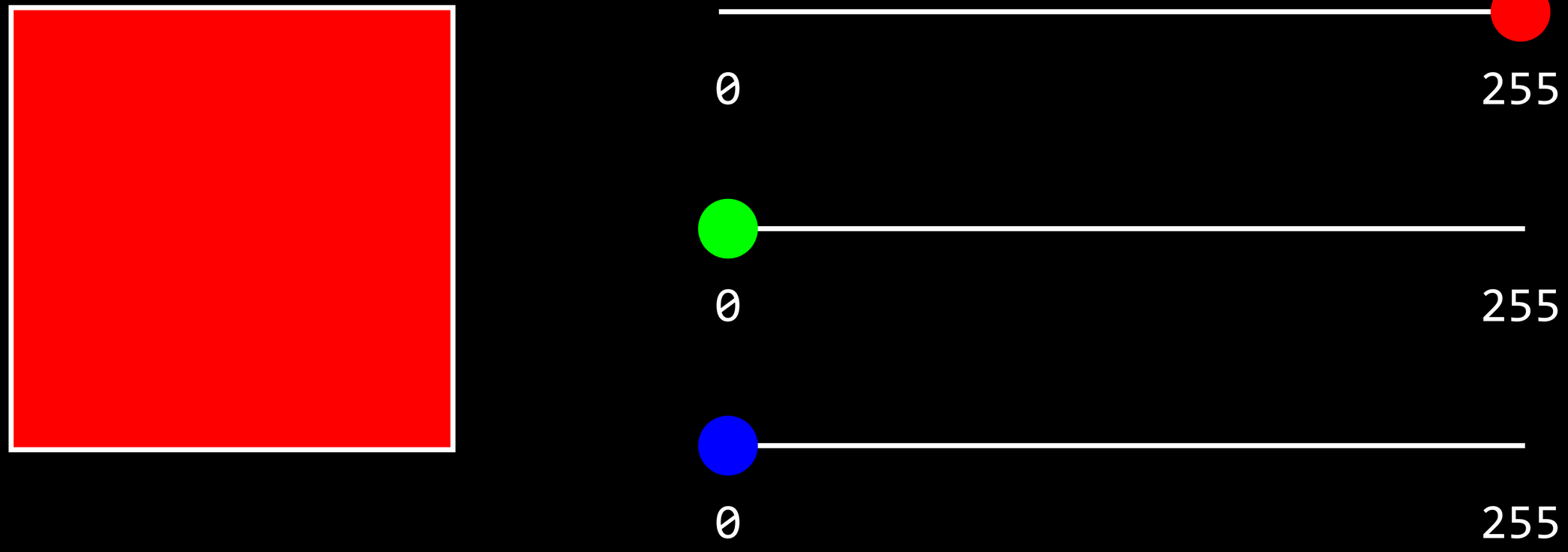

0
255

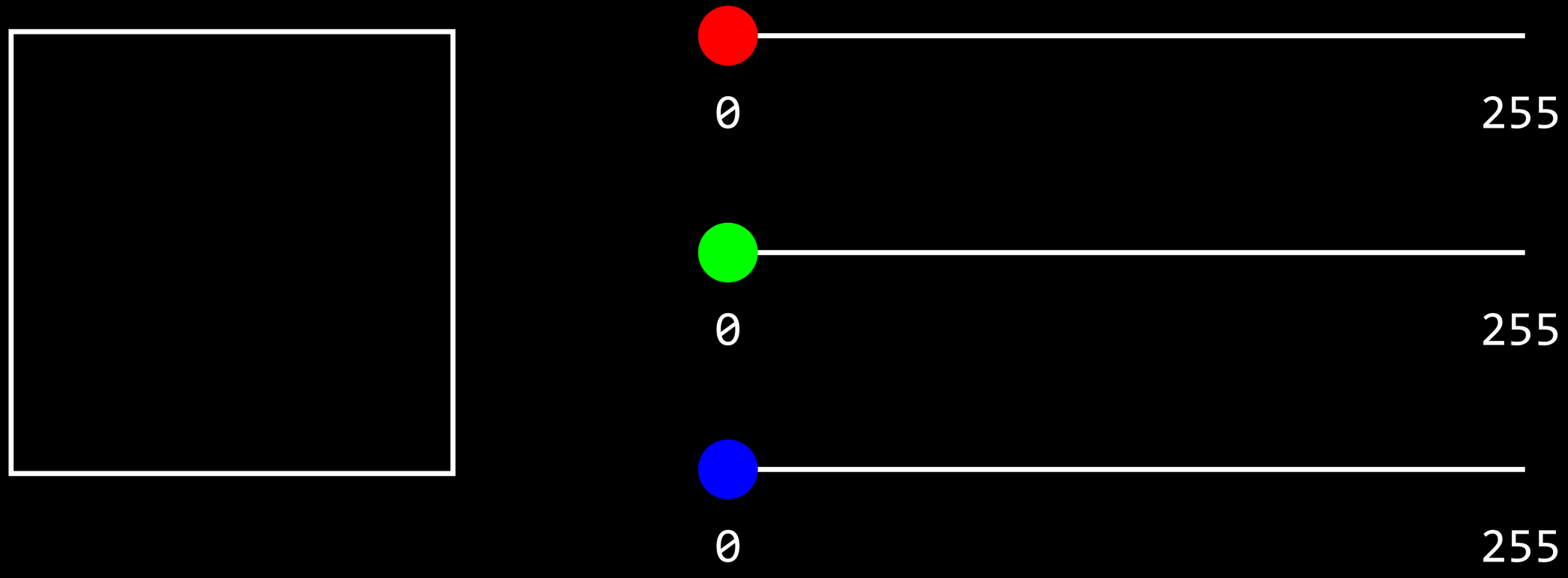


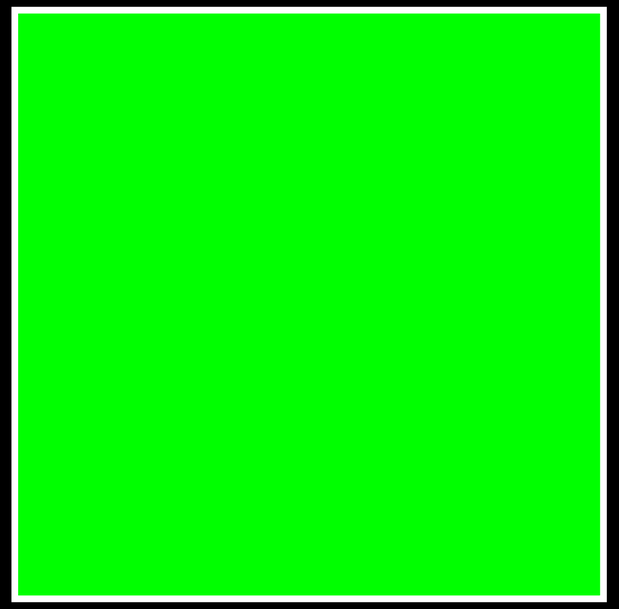

0
255


0
255


0
255
0
255
0
255


0
255
0
255
0
255

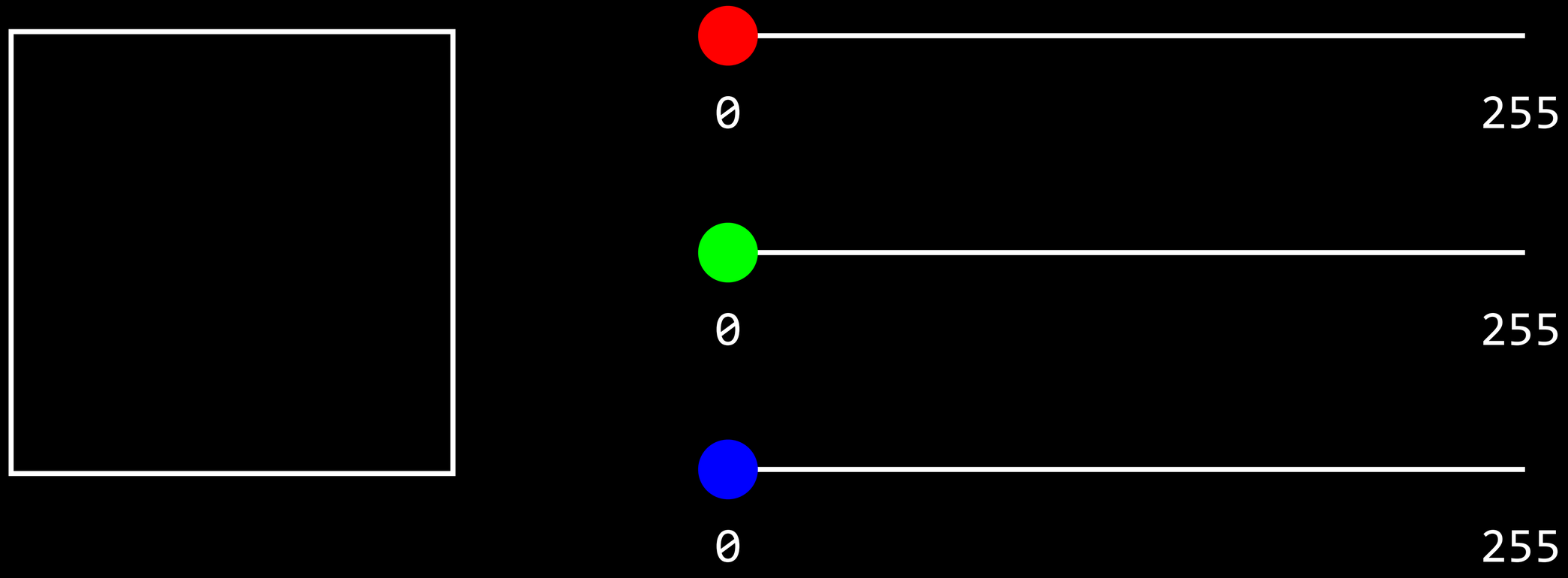

0
255
0
255
0
255

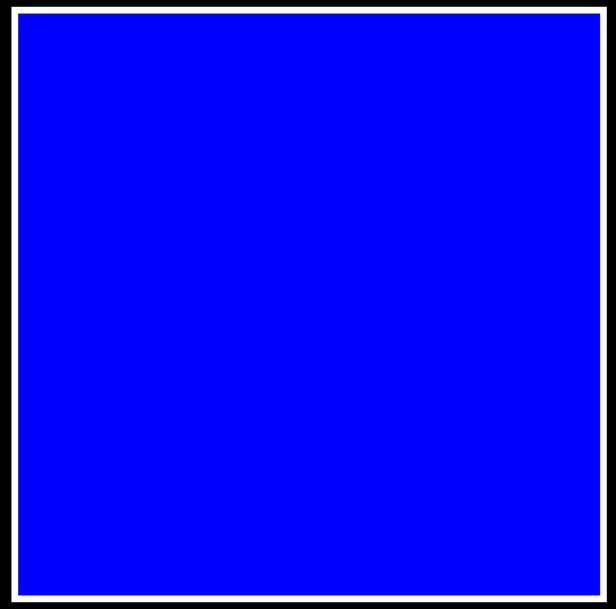


0 255

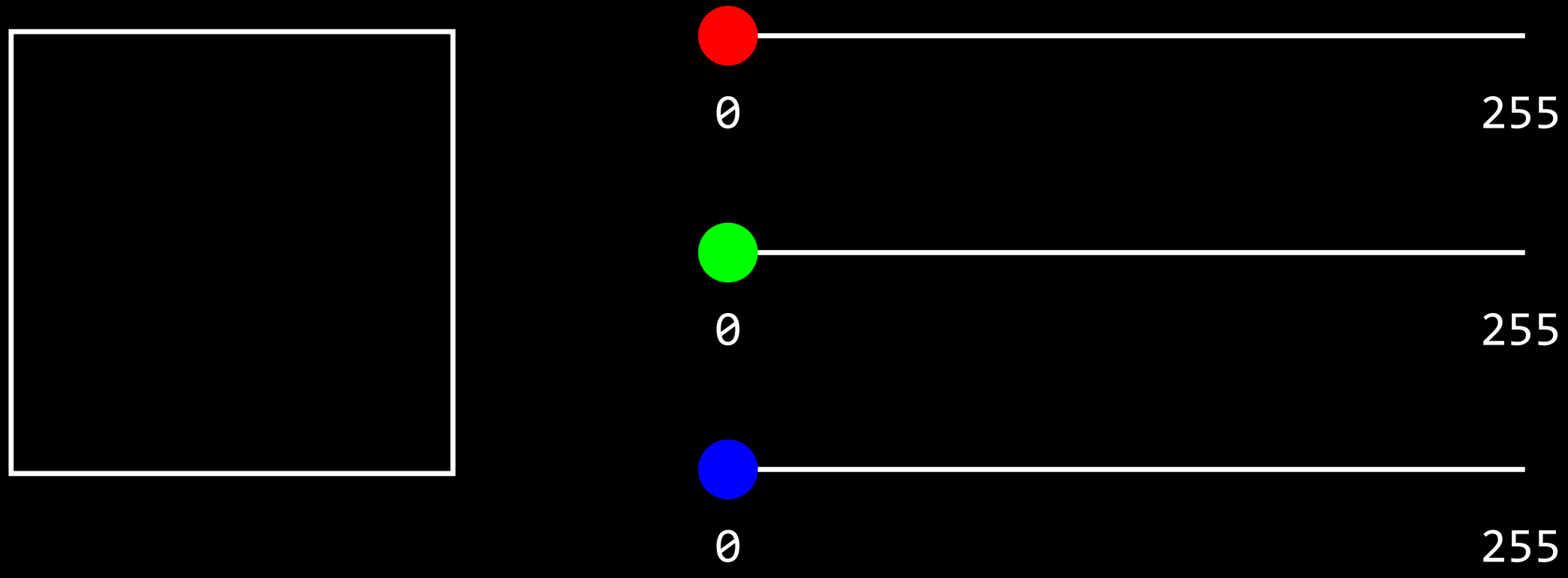
0 255

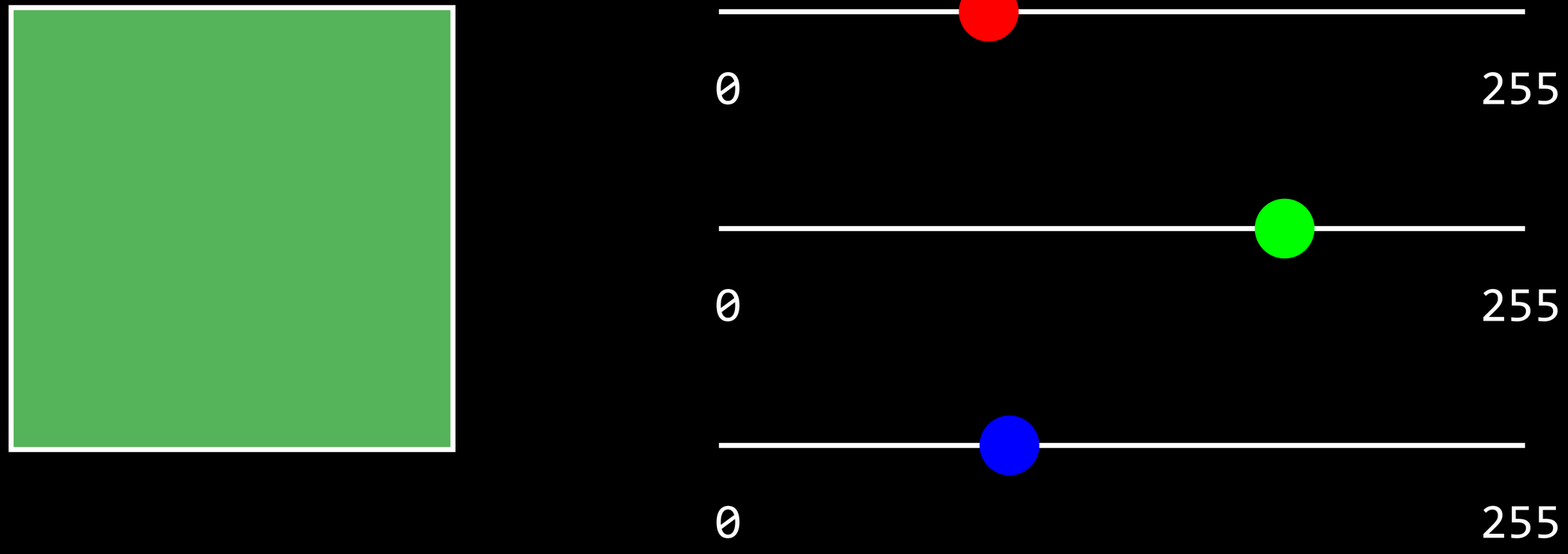
0 255

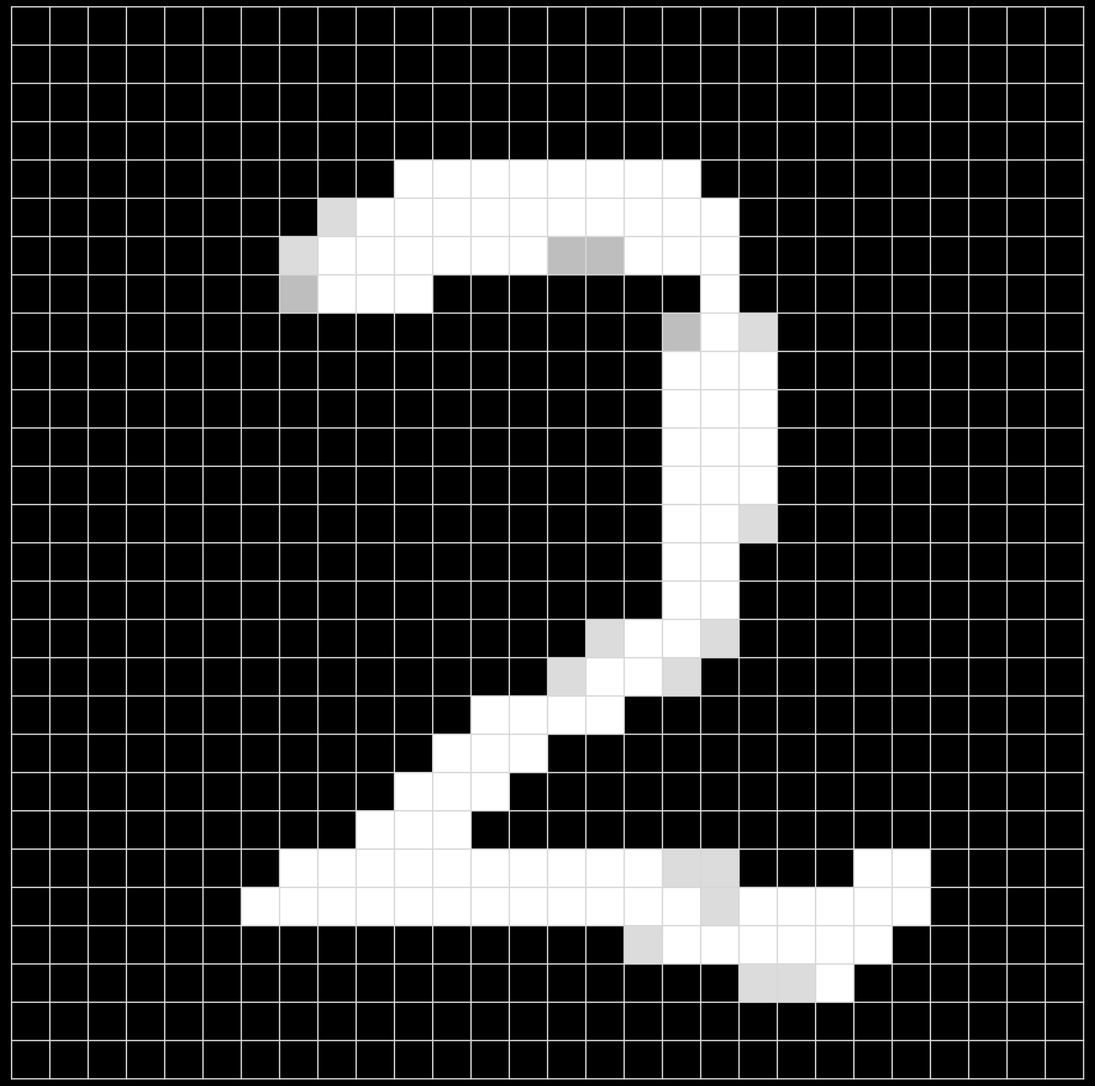

0
255
0
255
0
255

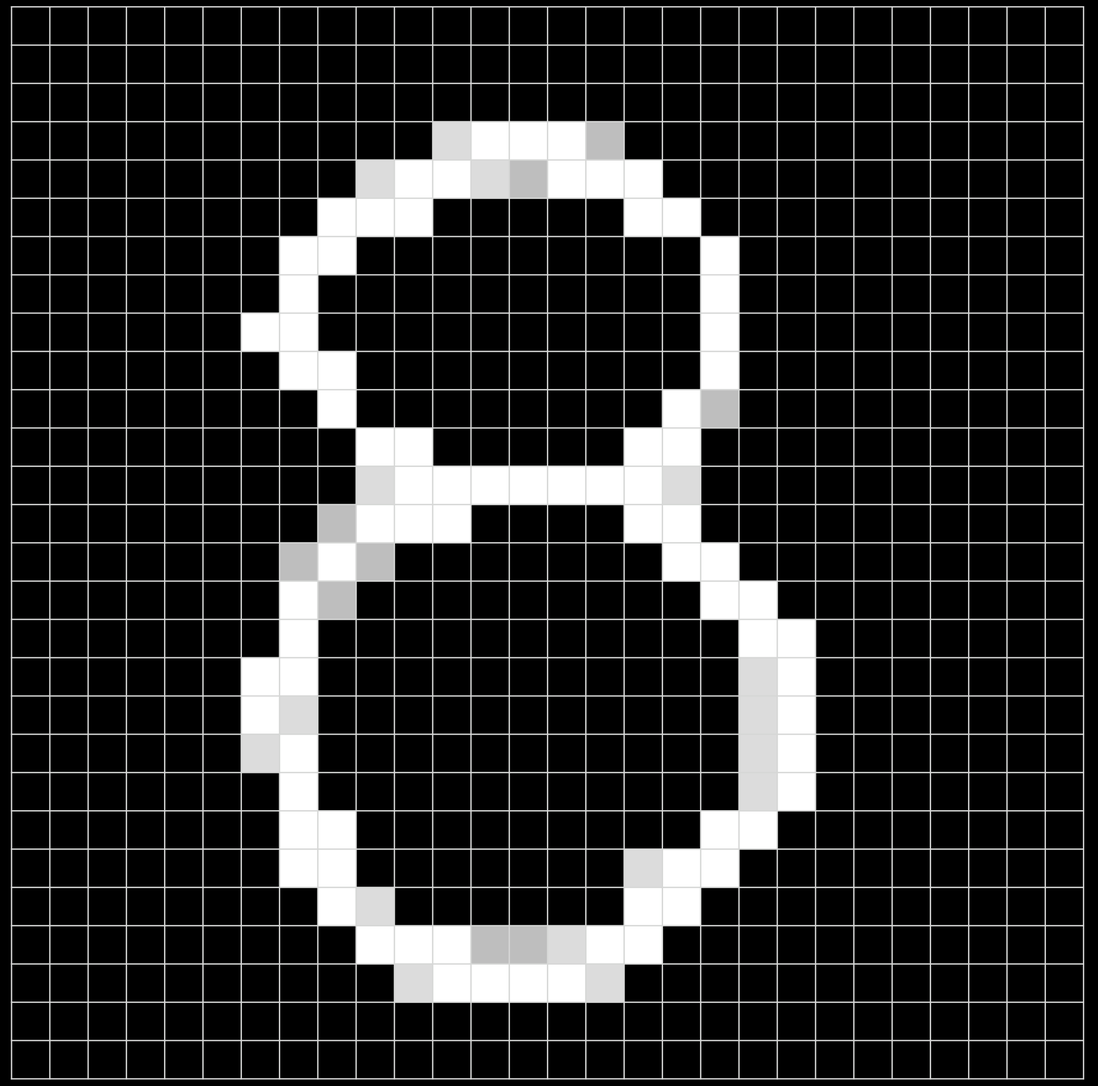



0
255
0
255
0
255


0
255
0
255
0
255


0
255
0
255
0
255





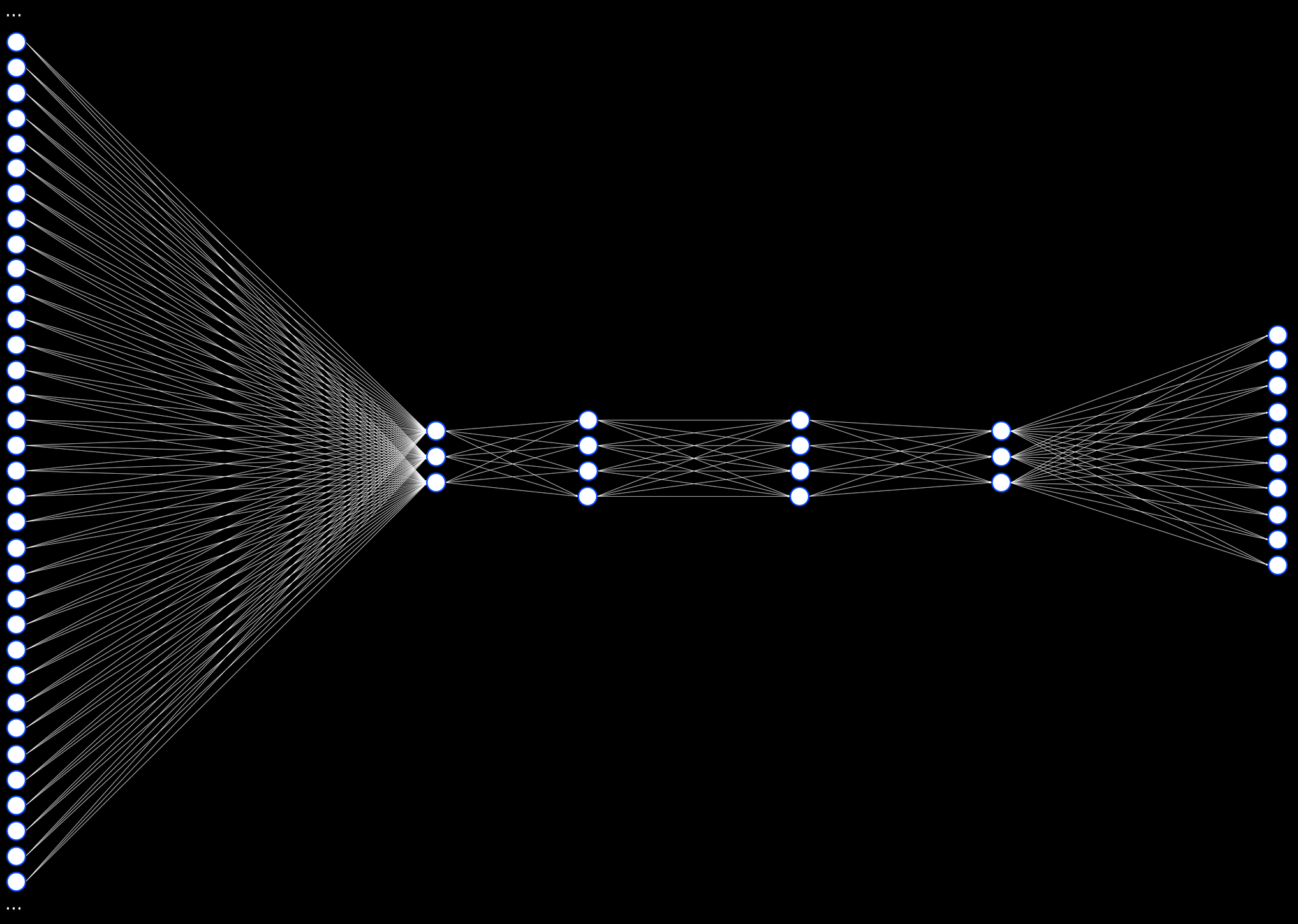


image convolution

applying a filter that adds each pixel value of an image to its neighbors, weighted according to a kernel matrix

0	-1	0
-1	5	-1
0	-1	0

10	20	30	40
10	20	30	40
20	30	40	50
20	30	40	50

0	-1	0
-1	5	-1
0	-1	0

10	20	30	40
10	20	30	40
20	30	40	50
20	30	40	50

0	-1	0
-1	5	-1
0	-1	0

10	20	30	40
10	20	30	40
20	30	40	50
20	30	40	50

0	-1	0
-1	5	-1
0	-1	0

10	20	30	40
10	20	30	40
20	30	40	50
20	30	40	50

0	-1	0
-1	5	-1
0	-1	0

10	

10	20	30	40
10	20	30	40
20	30	40	50
20	30	40	50

0	-1	0
-1	5	-1
0	-1	0

10	

10	20	30	40
10	20	30	40
20	30	40	50
20	30	40	50

0	-1	0
-1	5	-1
0	-1	0

10	20

10	20	30	40
10	20	30	40
20	30	40	50
20	30	40	50

0	-1	0
-1	5	-1
0	-1	0

10	20
40	

10	20	30	40
10	20	30	40
20	30	40	50
20	30	40	50

0	-1	0
-1	5	-1
0	-1	0

10	20
40	

10	20	30	40
10	20	30	40
20	30	40	50
20	30	40	50

0	-1	0
-1	5	-1
0	-1	0

10	20
40	50

10	20	30	40
10	20	30	40
20	30	40	50
20	30	40	50

0	-1	0
-1	5	-1
0	-1	0

10	20
40	50

-1	-1	-1
-1	8	-1
-1	-1	-1

20	20	20
20	20	20
20	20	20

-1	-1	-1
-1	8	-1
-1	-1	-1

```
  (20)(-1) + (20)(-1) + (20)(-1)
+ (20)(-1) + (20)(8)  + (20)(-1)
+ (20)(-1) + (20)(-1) + (20)(-1)
```

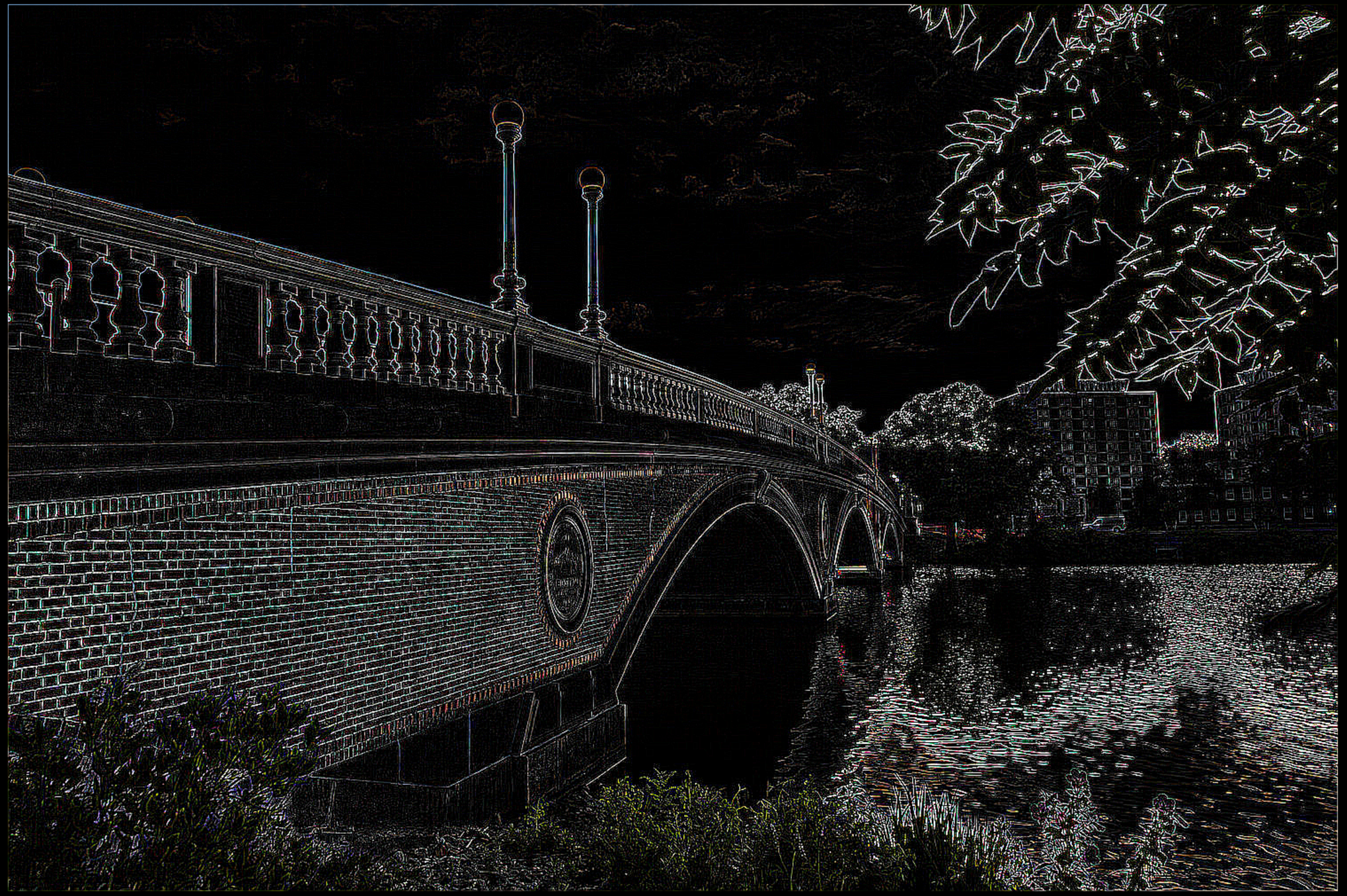
0

20	20	20
50	50	50
50	50	50

-1	-1	-1
-1	8	-1
-1	-1	-1

```
  (20)(-1) + (20)(-1) + (20)(-1)
+ (50)(-1) + (50)(8)  + (50)(-1)
+ (50)(-1) + (50)(-1) + (50)(-1)
```

90



pooling

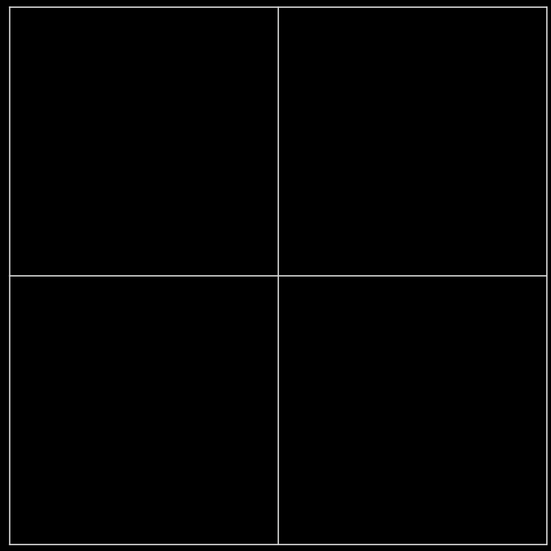
reducing the size of an input by sampling from regions in the input

max-pooling

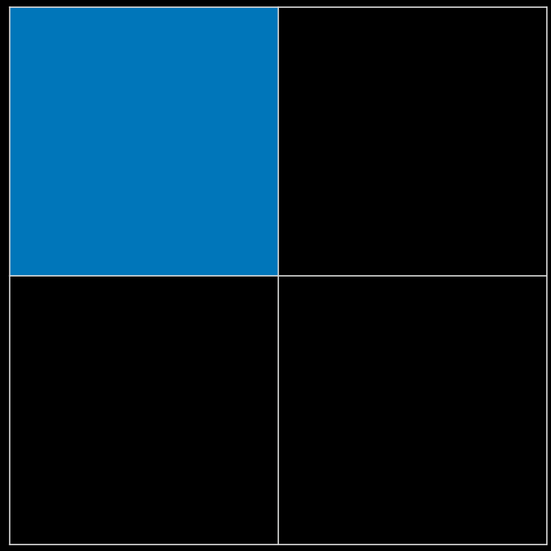
pooling by choosing the maximum value in each region

30	40	80	90
20	50	100	110
0	10	20	30
10	20	40	30

30	40	80	90
20	50	100	110
0	10	20	30
10	20	40	30



30	40	80	90
20	50	100	110
0	10	20	30
10	20	40	30



30	40	80	90
20	50	100	110
0	10	20	30
10	20	40	30

50	

30	40	80	90
20	50	100	110
0	10	20	30
10	20	40	30

50	

30	40	80	90
20	50	100	110
0	10	20	30
10	20	40	30

50	110

30	40	80	90
20	50	100	110
0	10	20	30
10	20	40	30

50	110

30	40	80	90
20	50	100	110
0	10	20	30
10	20	40	30

50	110
20	

30	40	80	90
20	50	100	110
0	10	20	30
10	20	40	30

50	110
20	

30	40	80	90
20	50	100	110
0	10	20	30
10	20	40	30

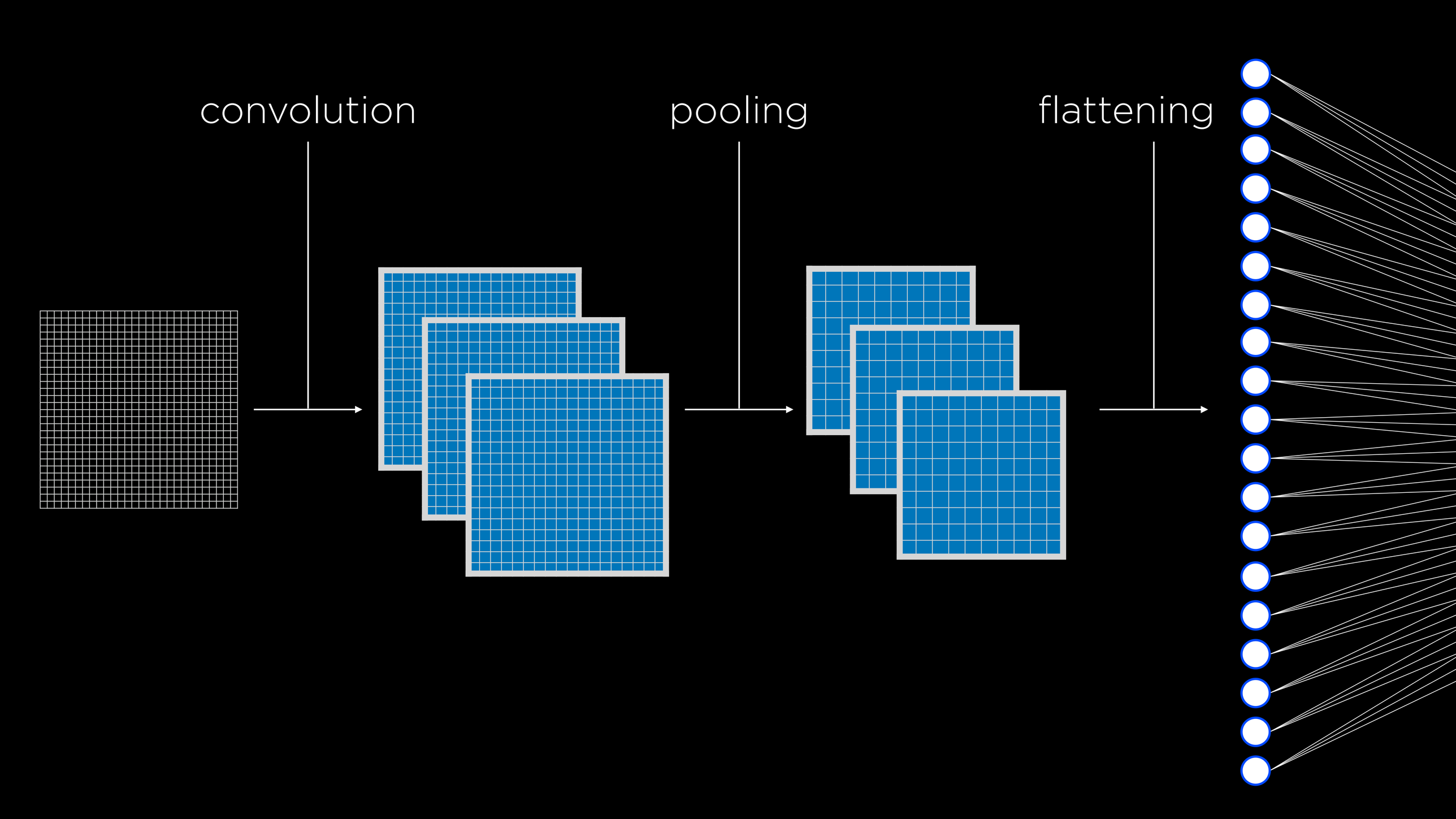
50	110
20	40

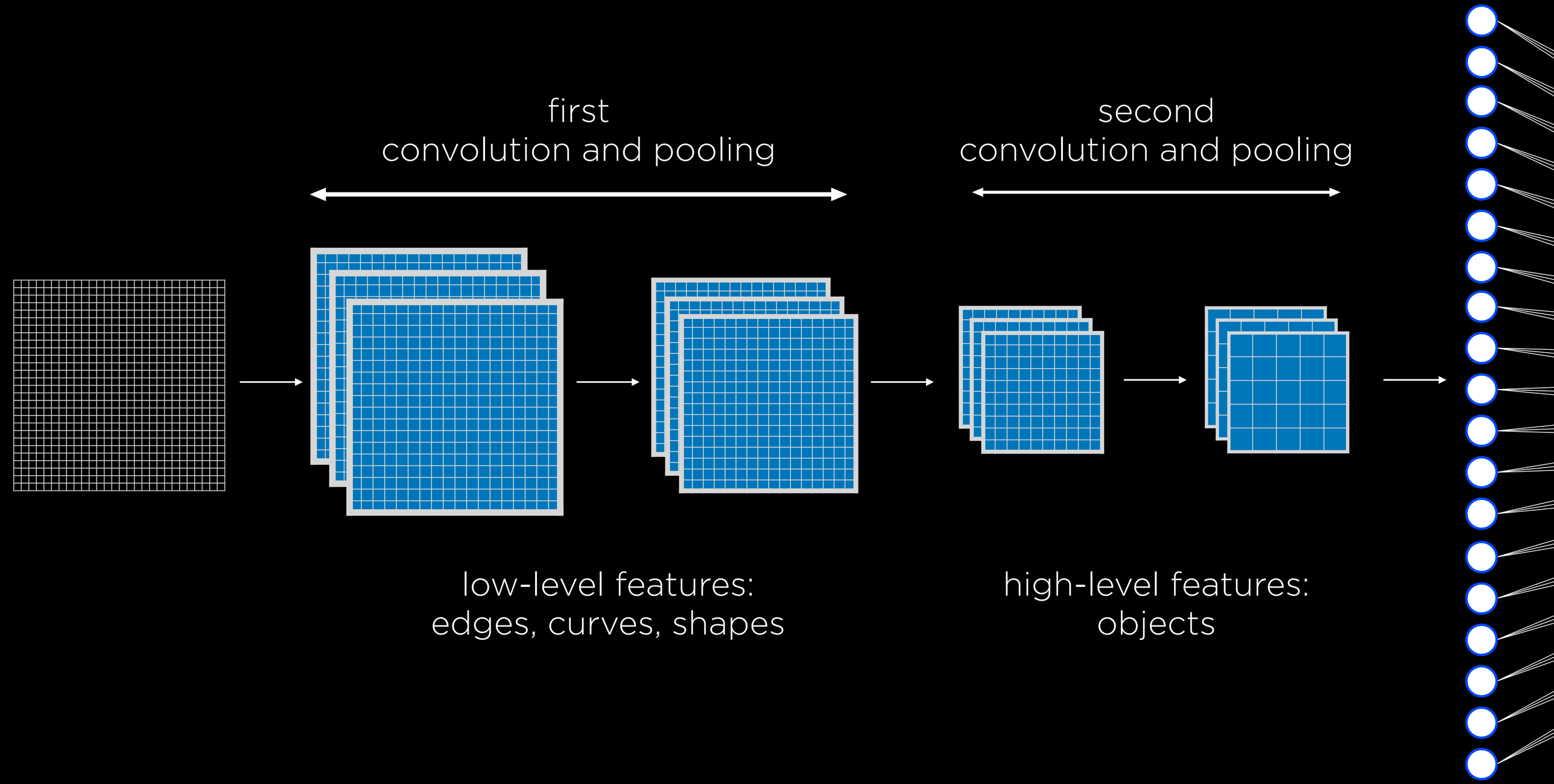
30	40	80	90
20	50	100	110
0	10	20	30
10	20	40	30

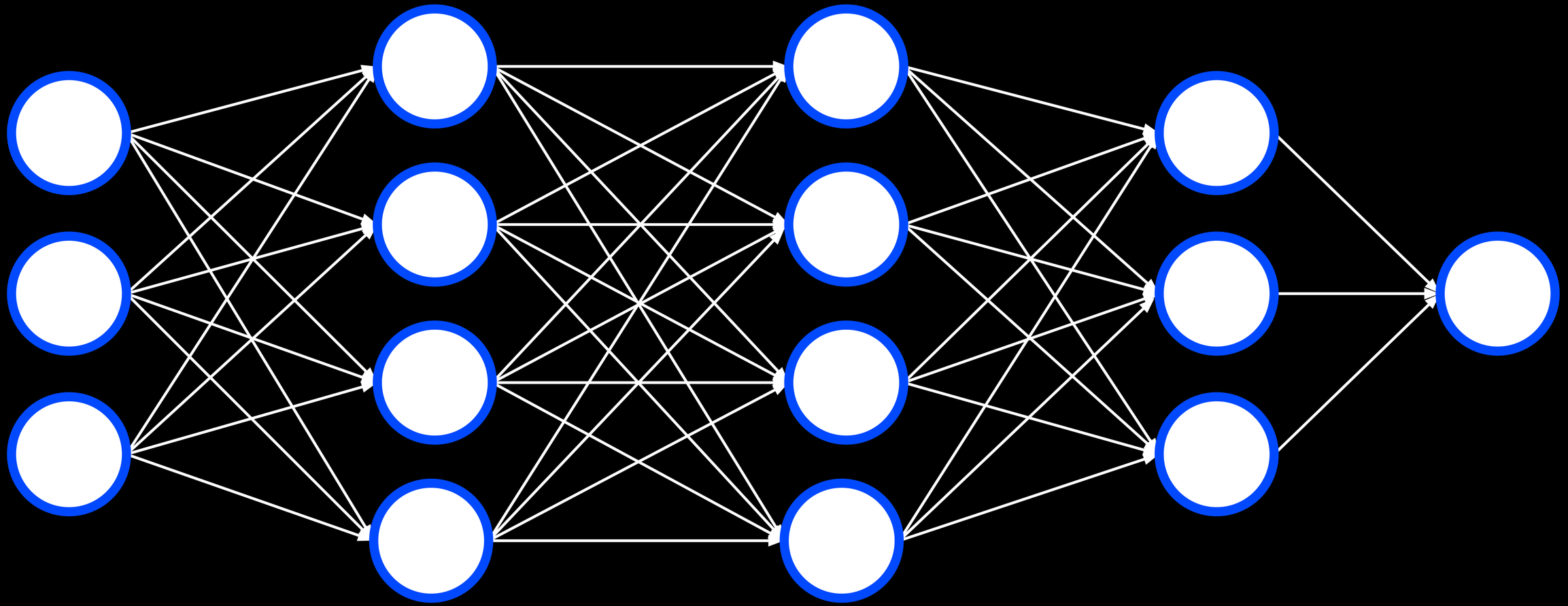
50	110
20	40

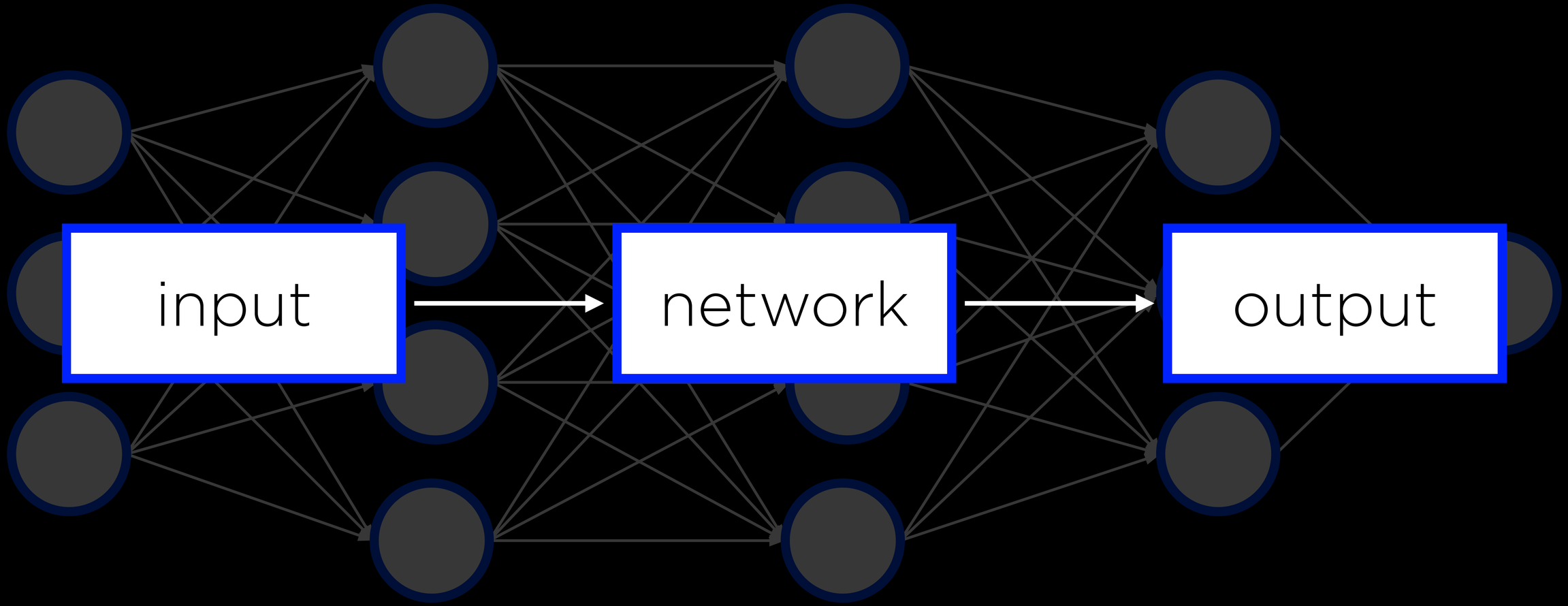
convolutional neural network

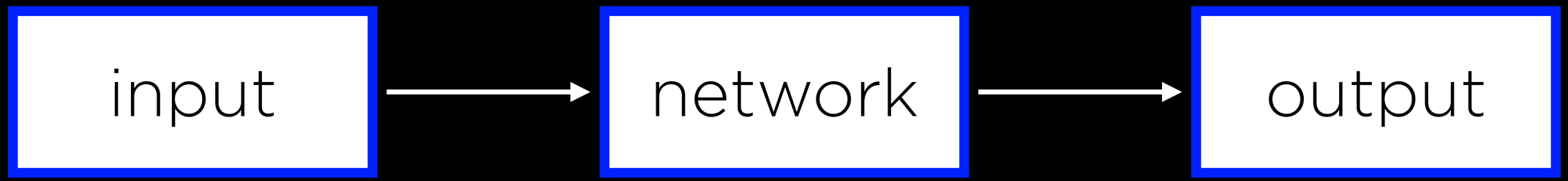
neural networks that use convolution,
usually for analyzing images


convolution
pooling
flattening


first
convolution and pooling
second
convolution and pooling
low-level features:
edges, curves, shapes
high-level features:
objects




input
network
output

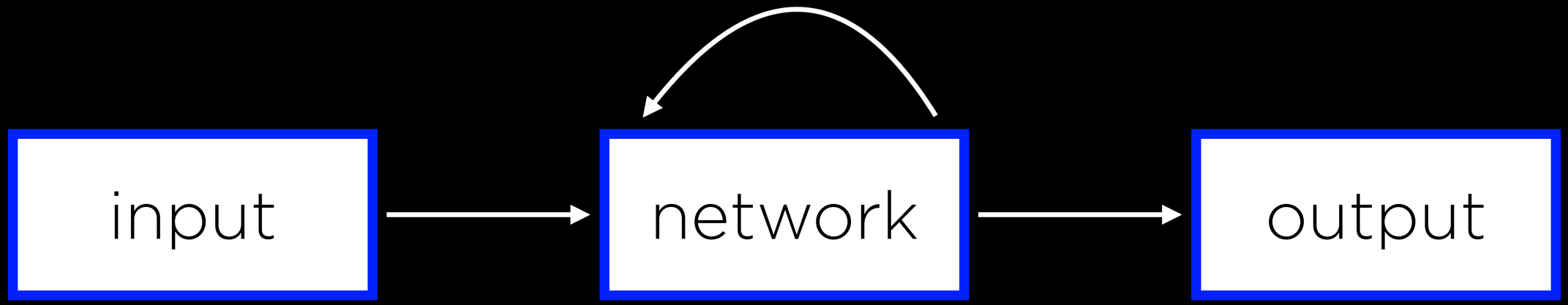

input
network
output

feed-forward neural network

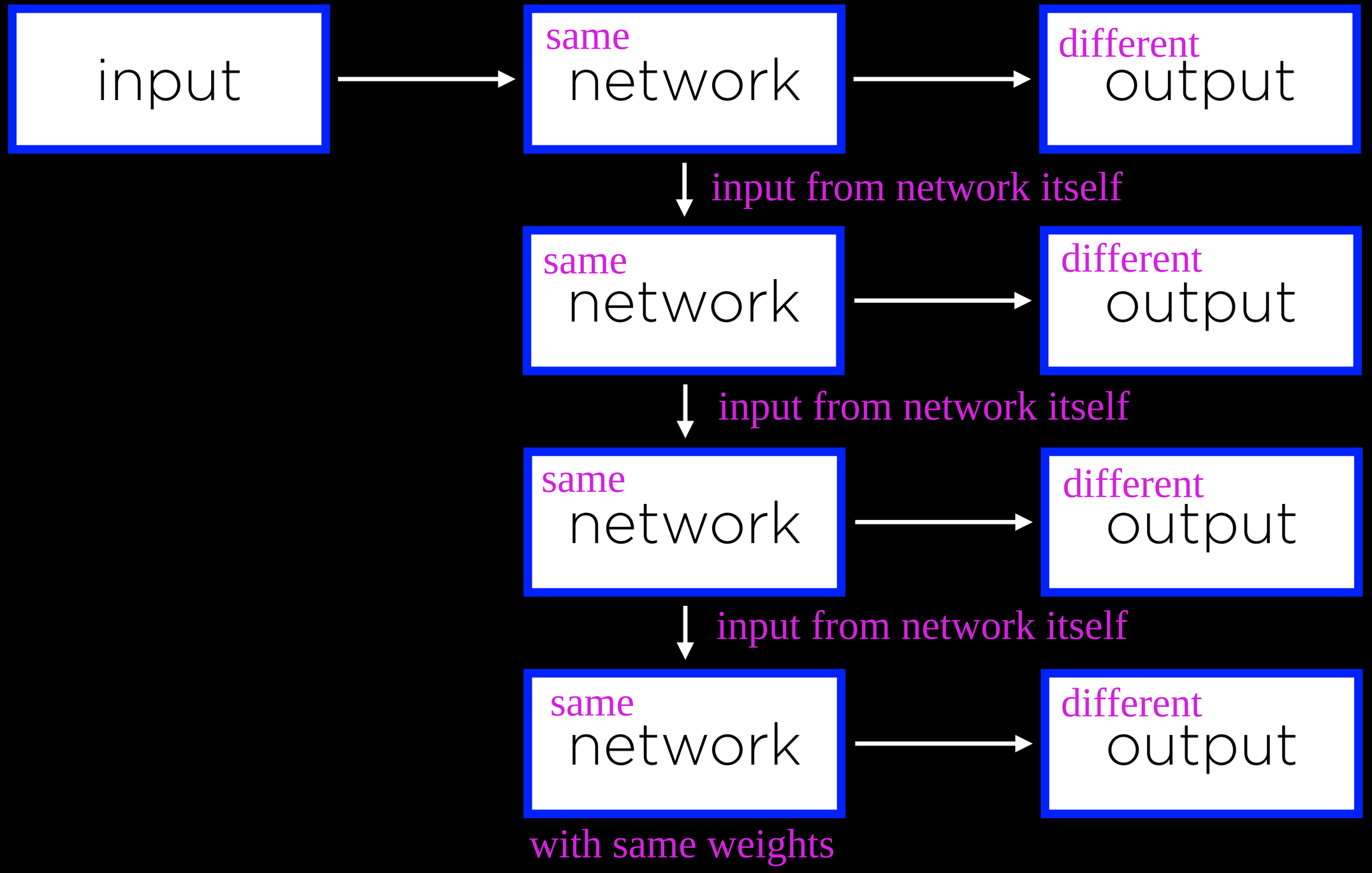
neural network that has connections only in one direction

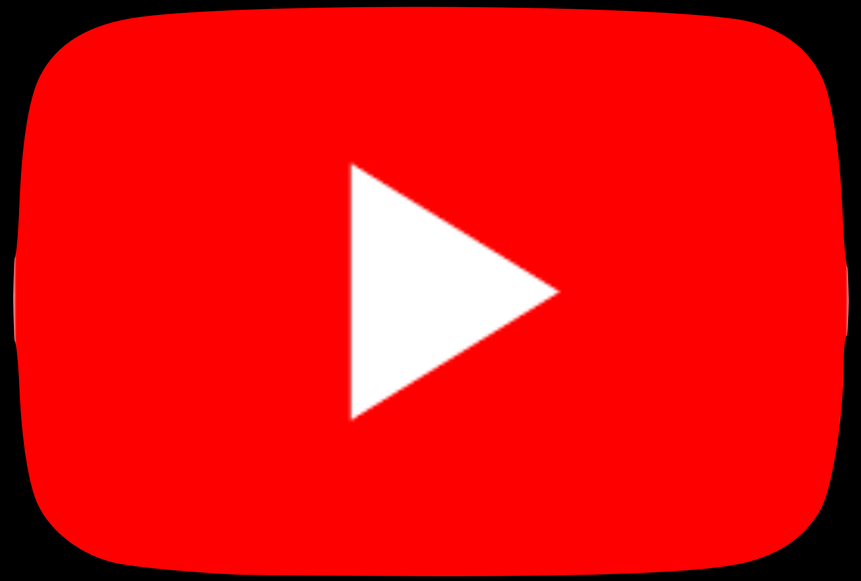
recurrent neural network

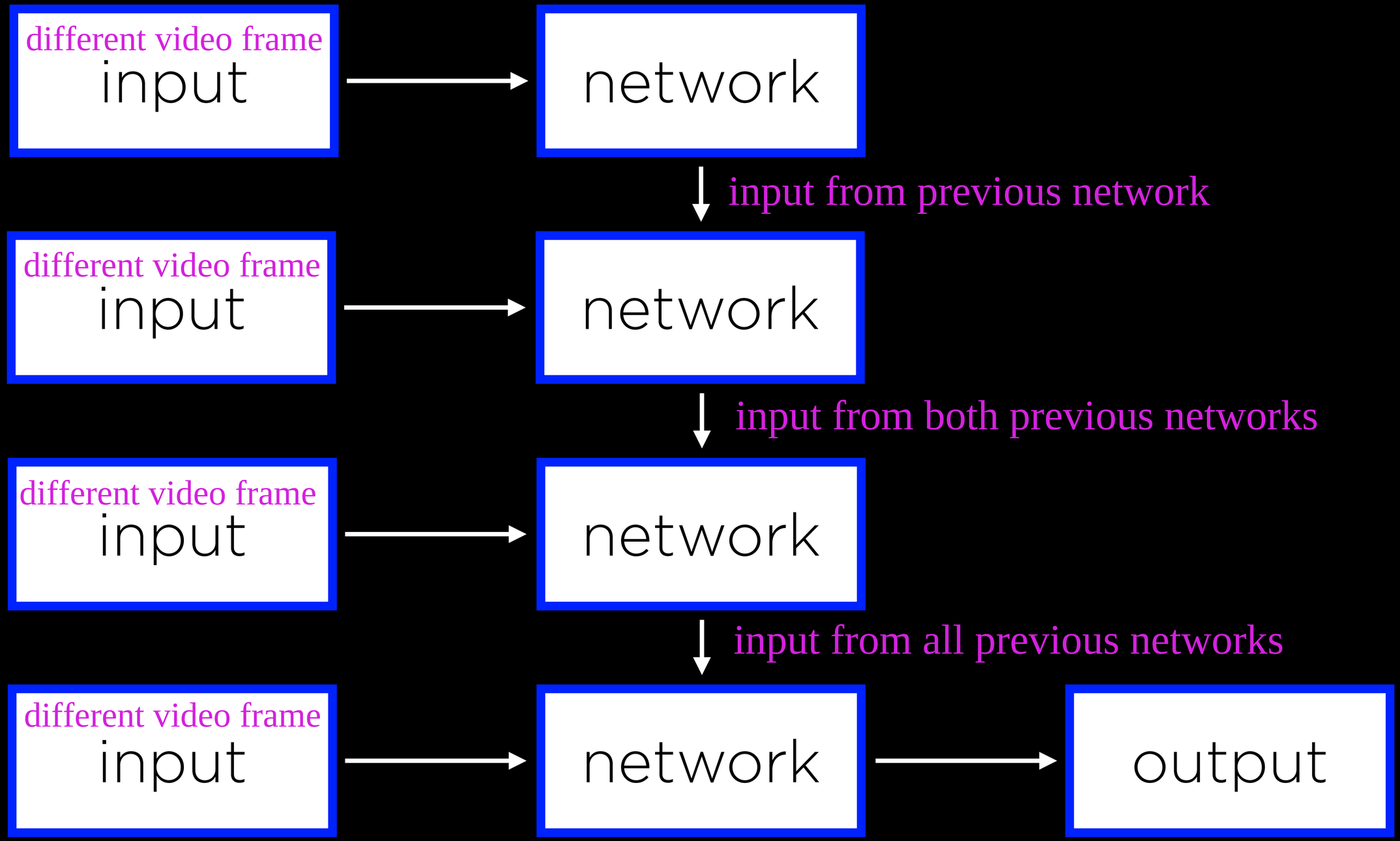
neural network that generates output that feeds back into its own inputs


input
network
output

group of people walking in front of a building


input
same
network
different
output
input from network itself
same
network
different
output
input from network itself
same
network
different
output
input from network itself
same
network
different
output
with same weights

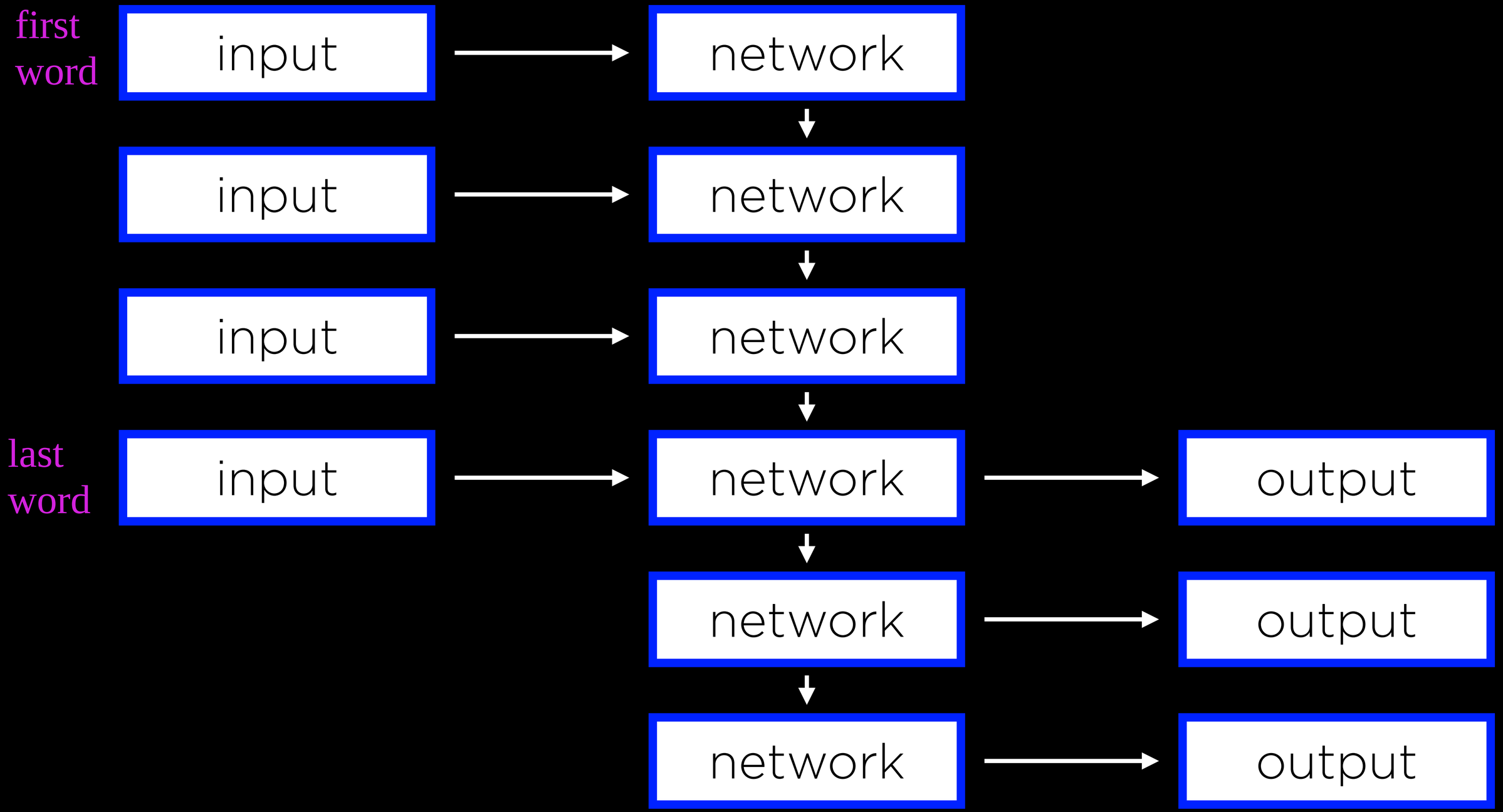



different video frame
input
network
input from previous network
different video frame
input
network
input from both previous networks
different video frame
input
network
input from all previous networks
different video frame
input
network
output

She is in the library.

↓

她在圖書館


first word
input
network
input
network
input
network
last word
input
network
output
network
output
network
output

Neural Networks

Introduction to **Artificial Intelligence** with Python