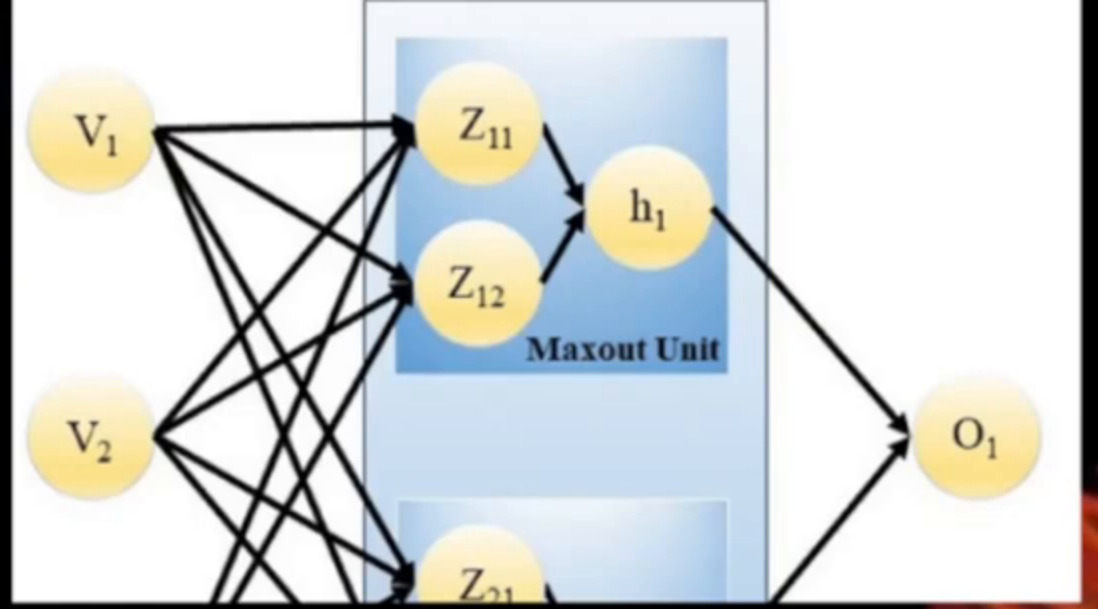
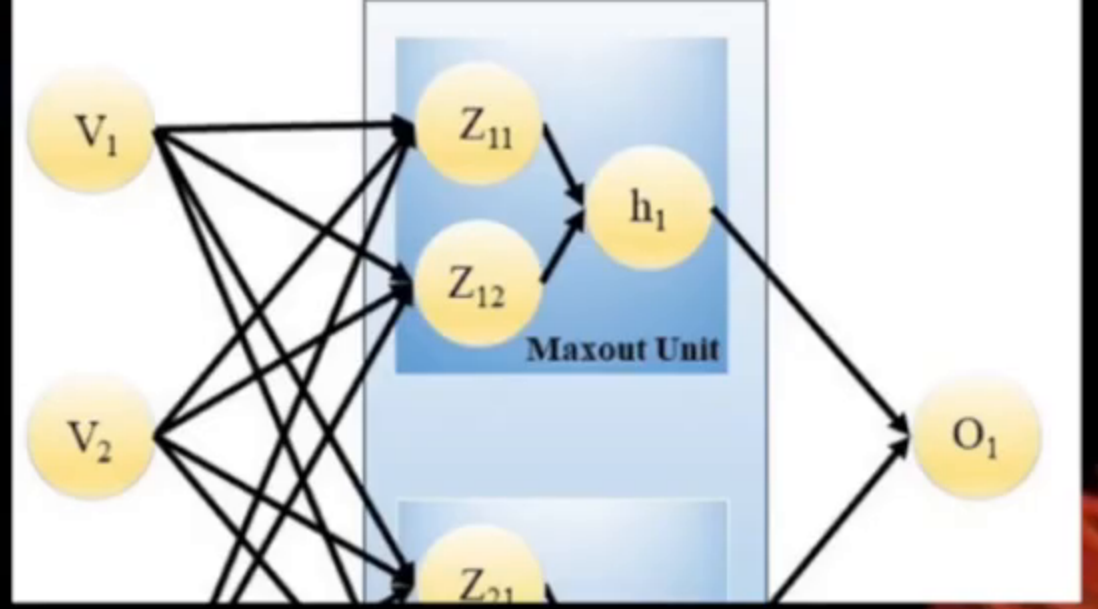
ACTIVATION FUNCTIONS



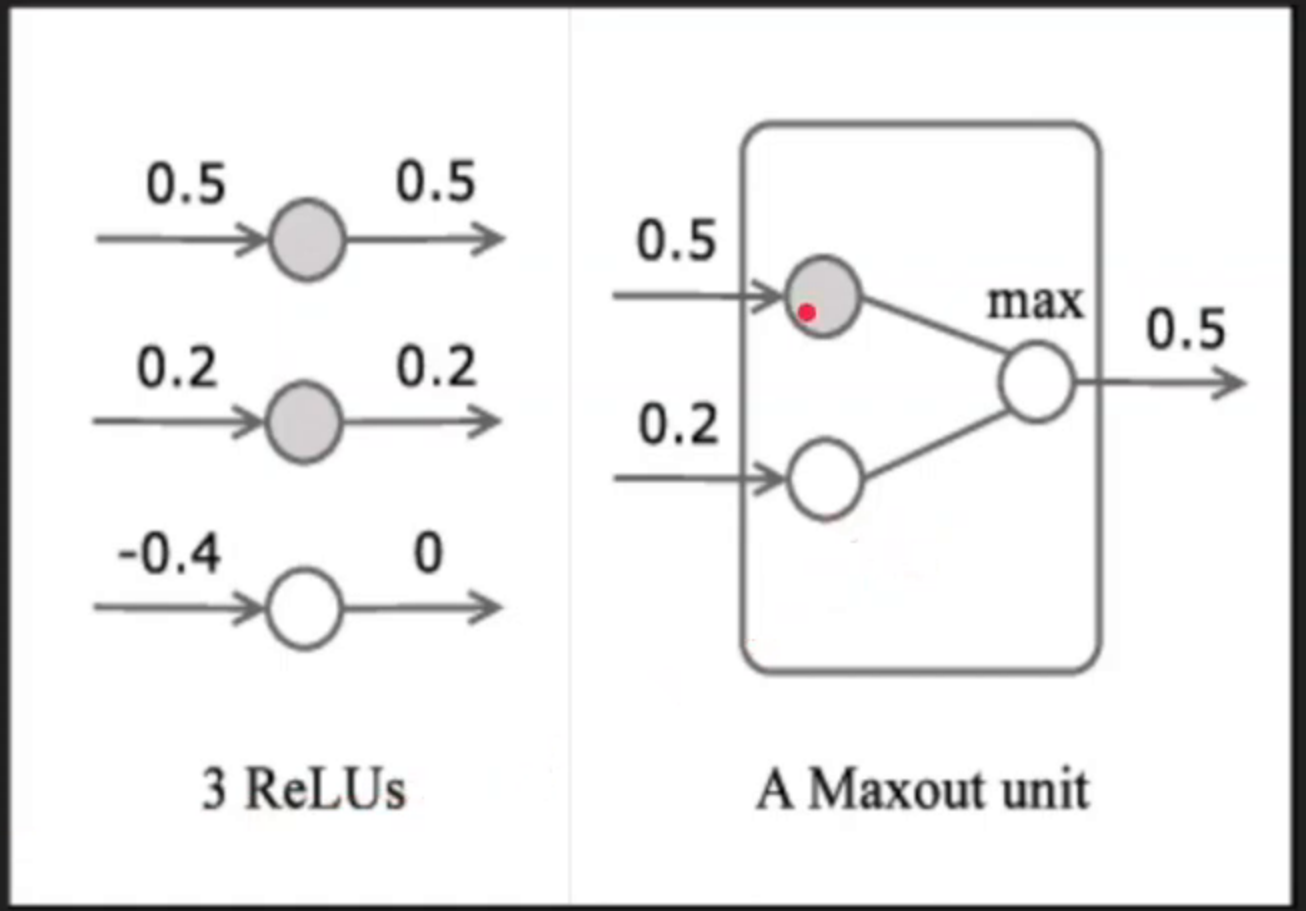
MAXOUT ACTIVATION

ACTIVATION FUNCTIONS



MAXOUT ACTIVATION

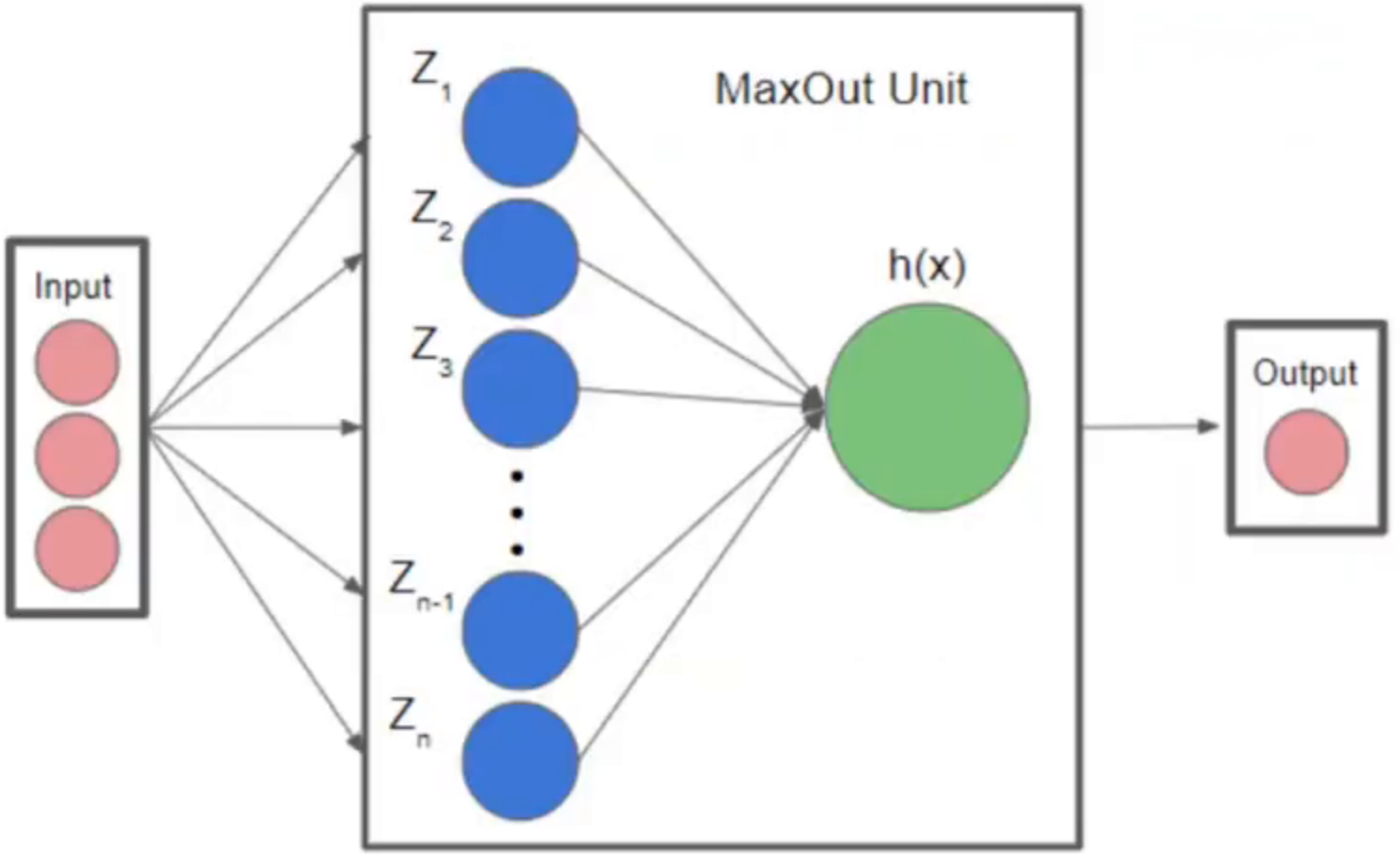
Illustration



Maxout

$$h(x) = \max\left(Z_1, Z_2, \ldots, Zn\right)$$

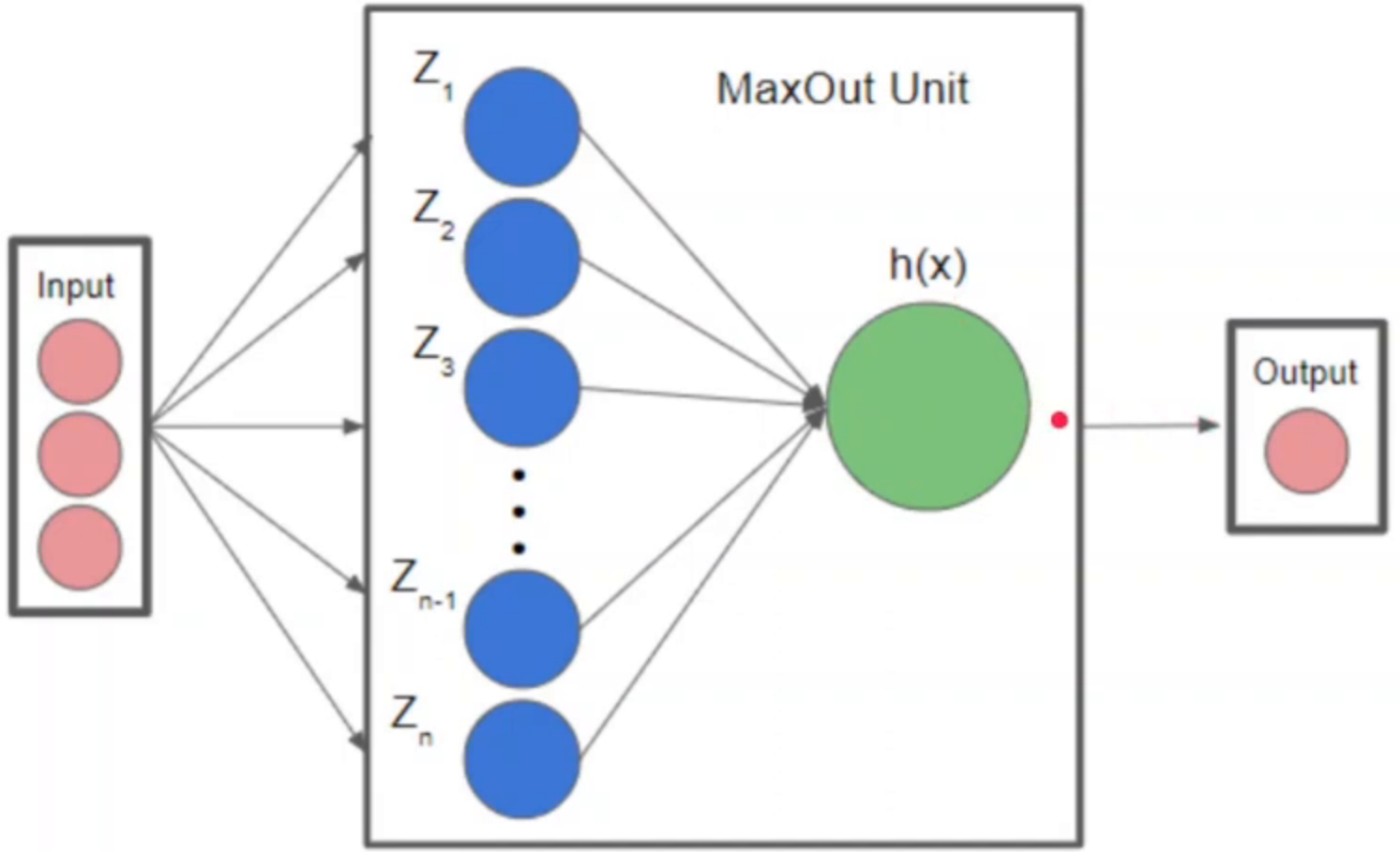
$$h(x) = \max\left(W_1 \cdot x + b_1, W_2 \cdot x + b_2, \ldots, Wn \cdot x + bn\right)$$



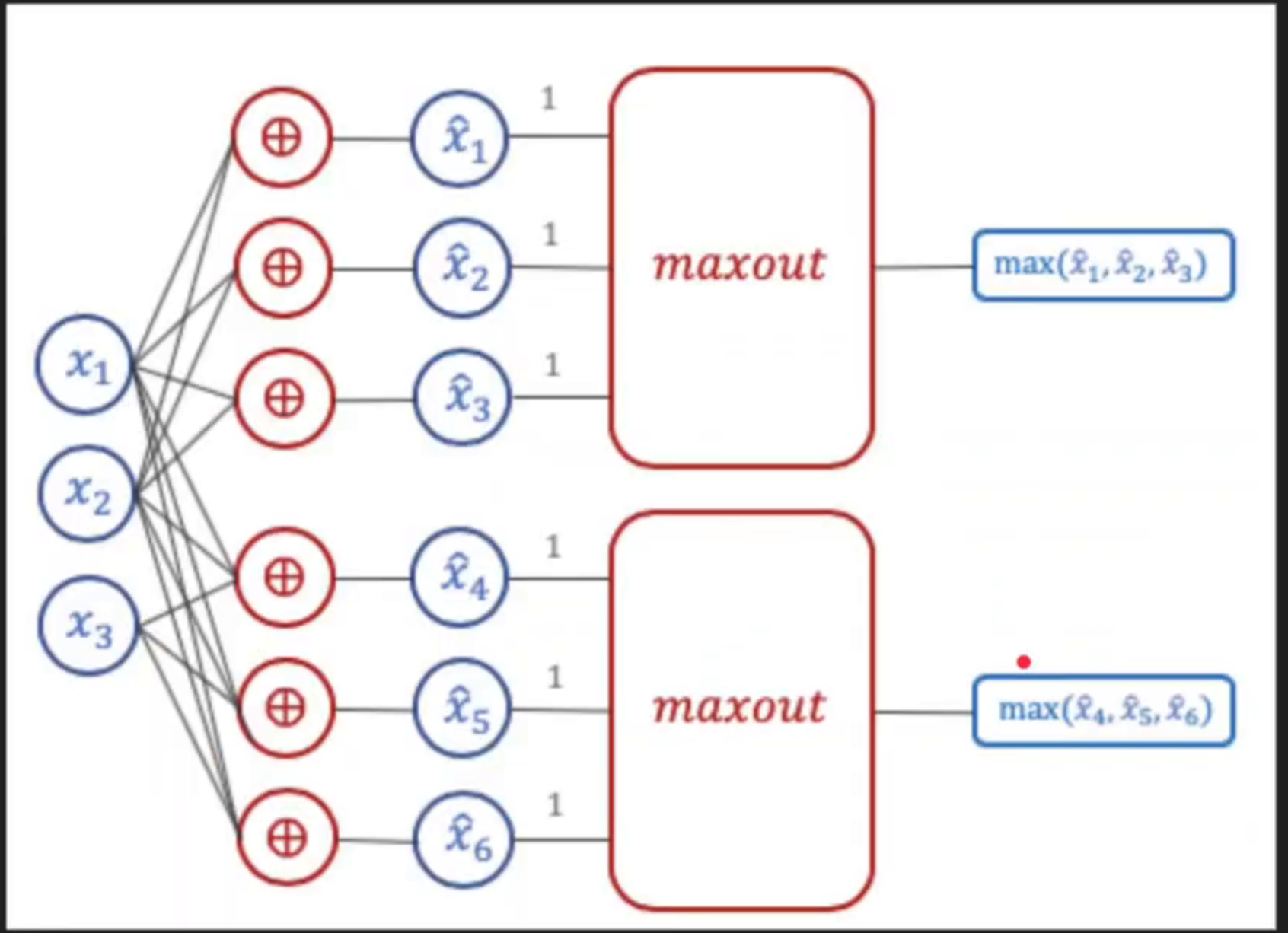
Maxout

$$h(x) = \max\left(Z_1, Z_2, \ldots, Zn\right)$$

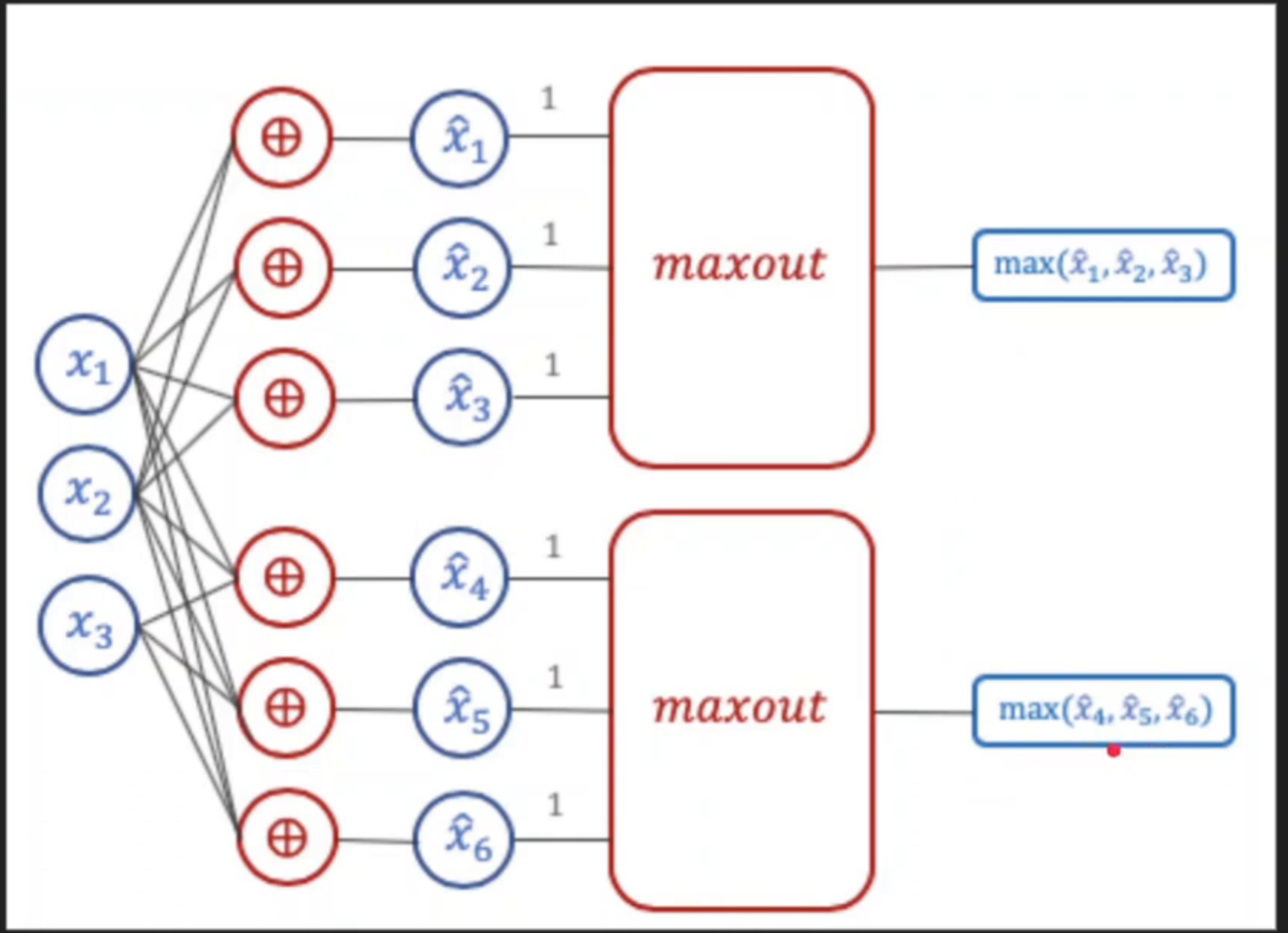
$$h(x) = \max\left(W_1 \cdot x + b_1, W_2 \cdot x + b_2, \ldots, Wn \cdot x + bn\right)$$



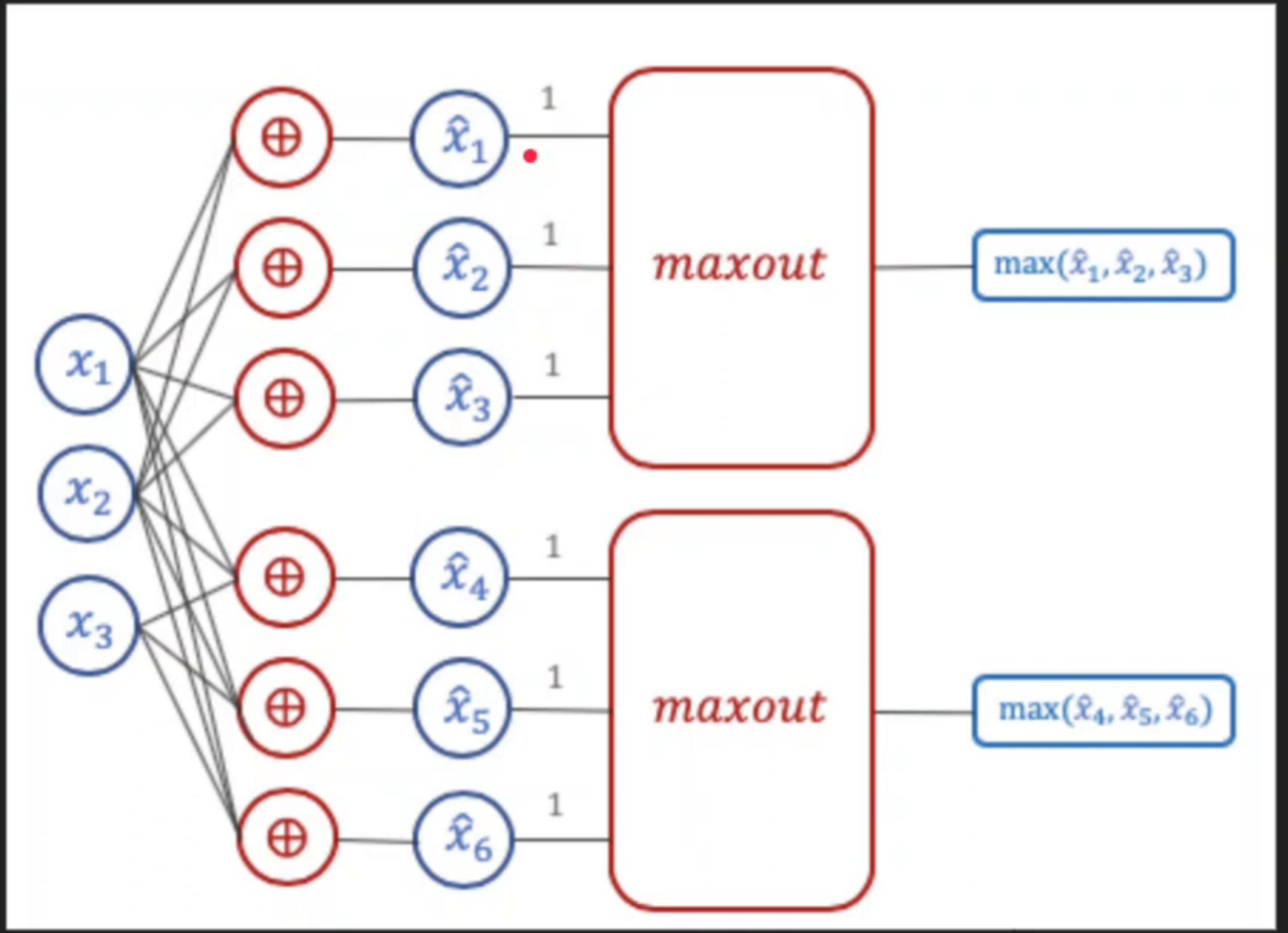
Illustration



Illustration

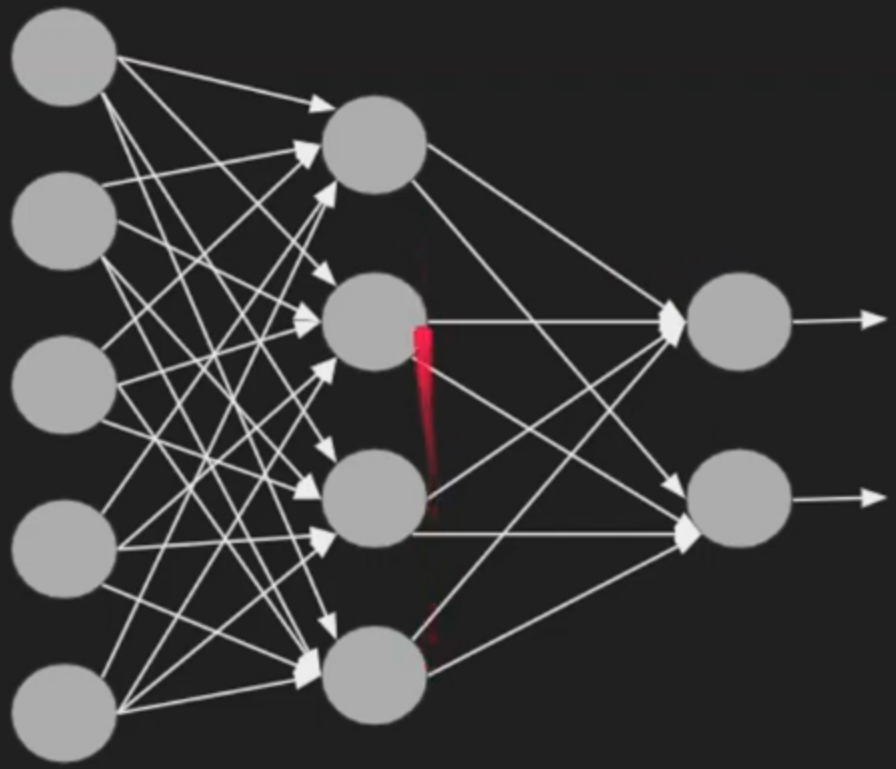


Illustration



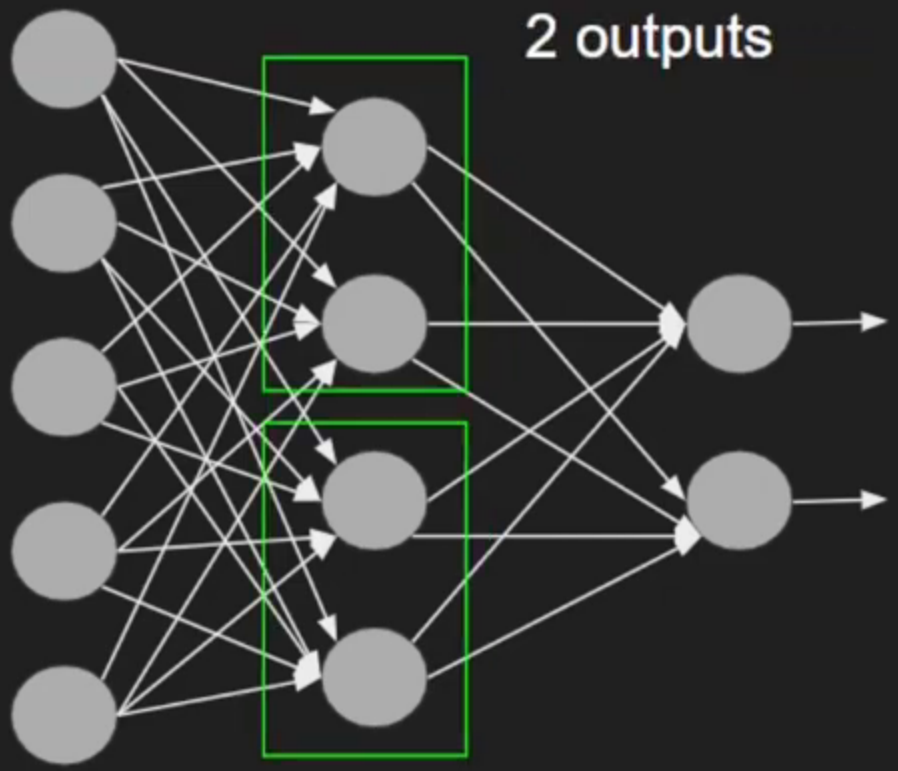
Network - (5,4,2)

Input Hidden Output



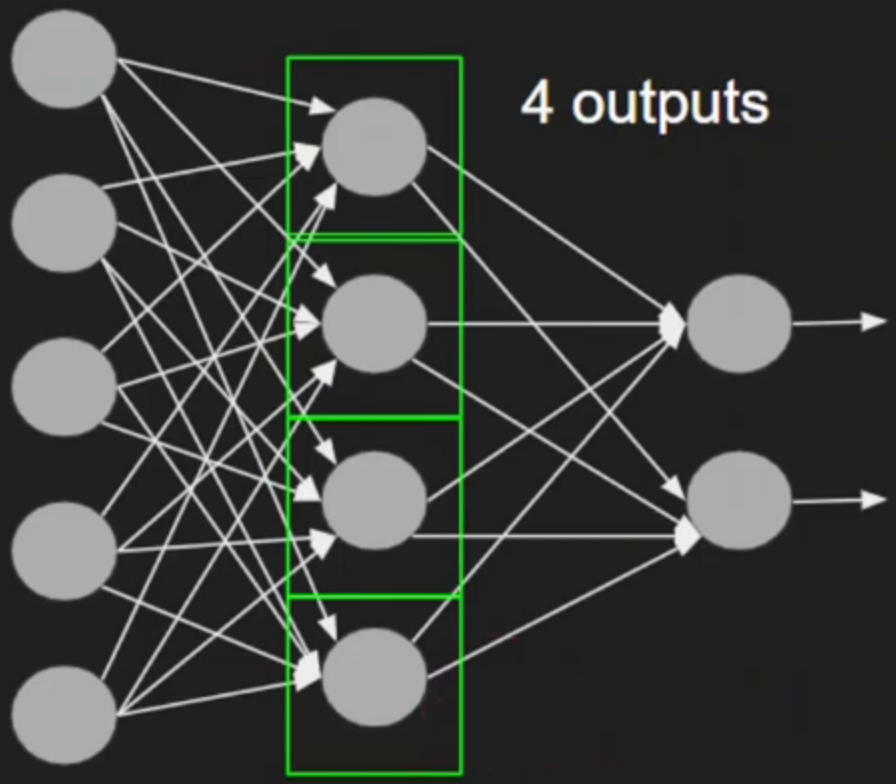
Network - (5,4,2)

Input Hidden Output



Network - (5,4,2)

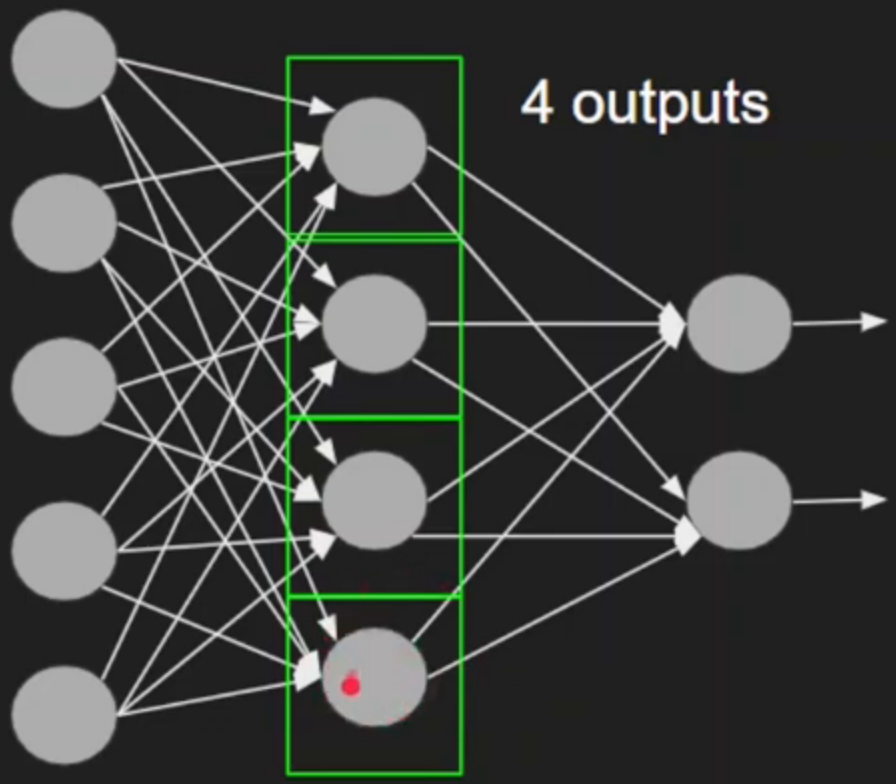
Input Hidden Output



Number of outputs = Number of Groups

Network - (5,4,2)

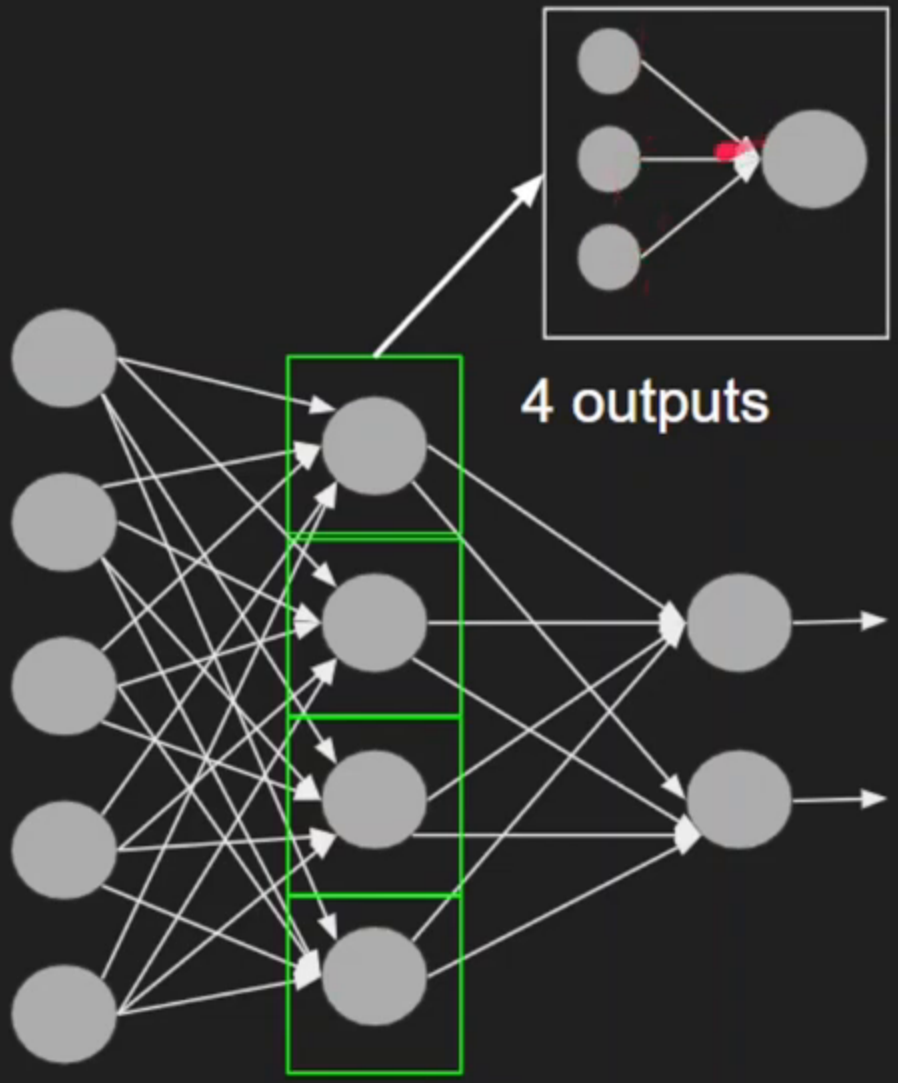
Input Hidden Output



Number of outputs = Number of Groups

Network - (5,4,2)

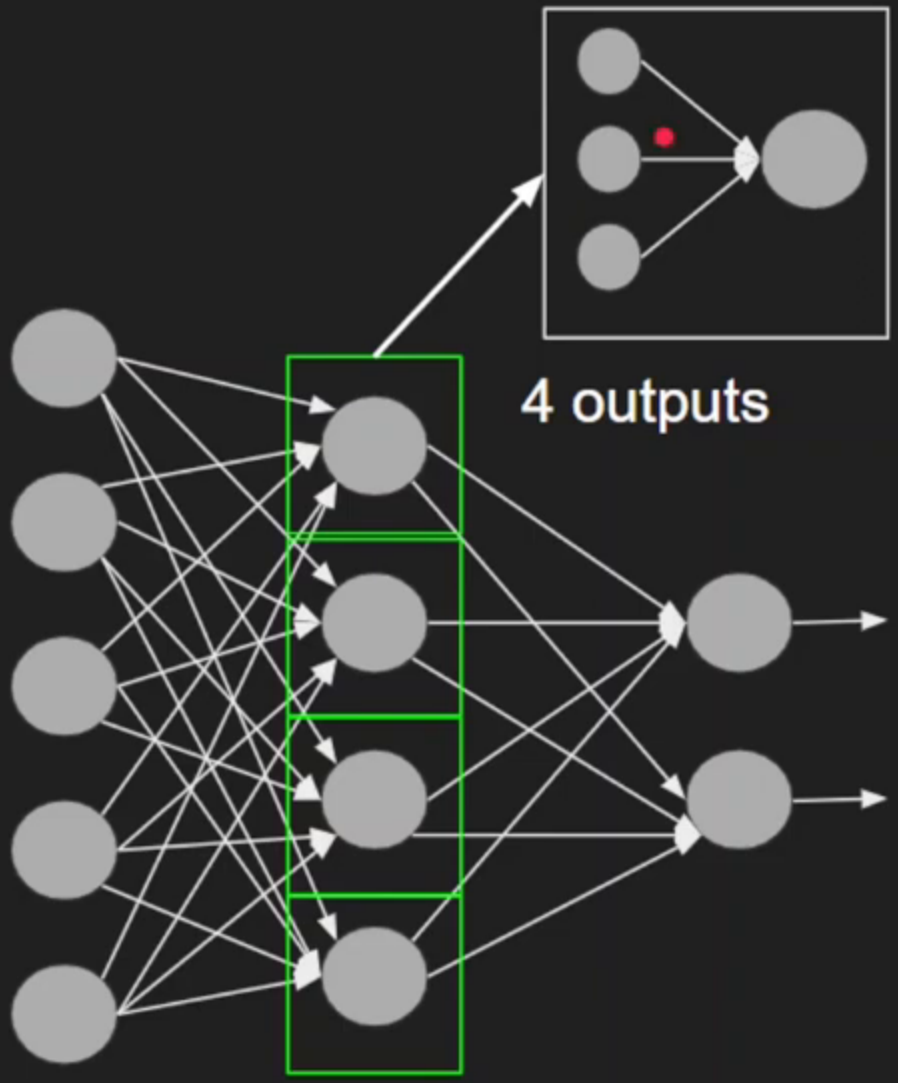
Input Hidden Output



Number of outputs = Number of Groups

Network - (5,4,2)

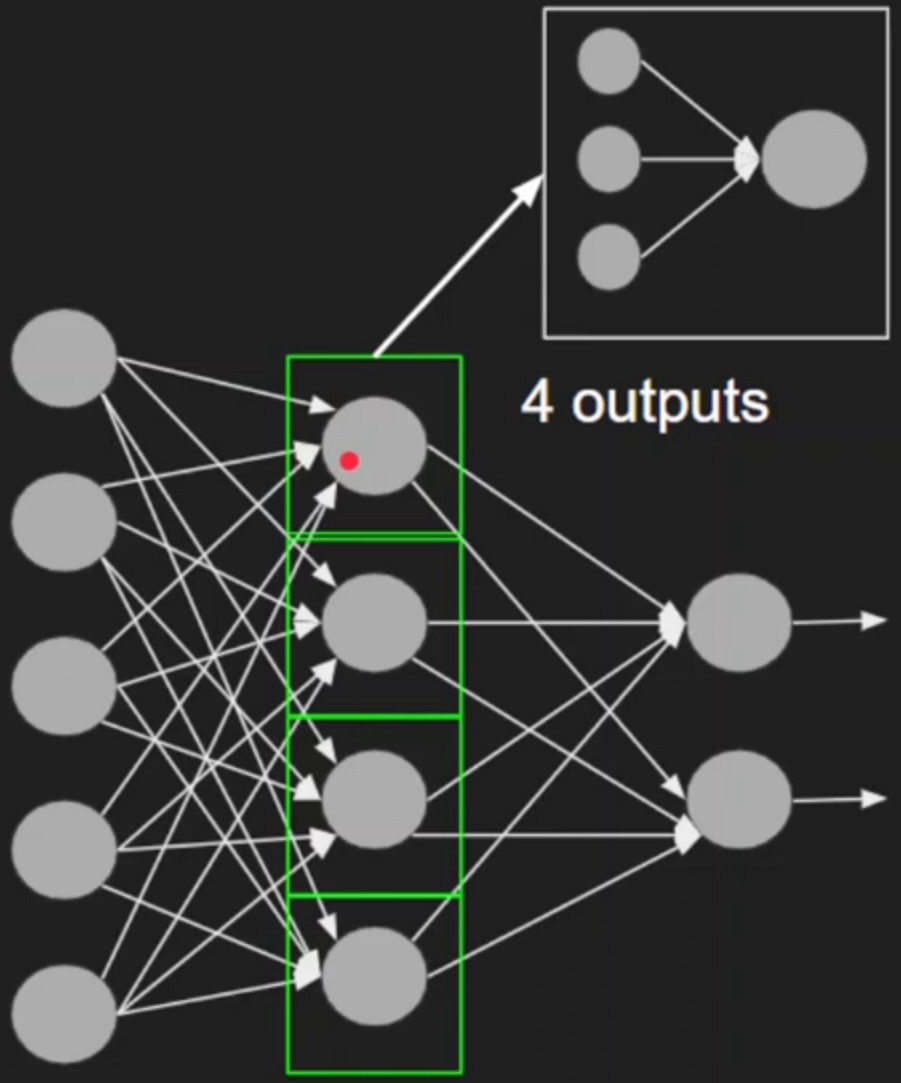
Input Hidden Output



Number of outputs = Number of Groups

Network - (5,4,2)

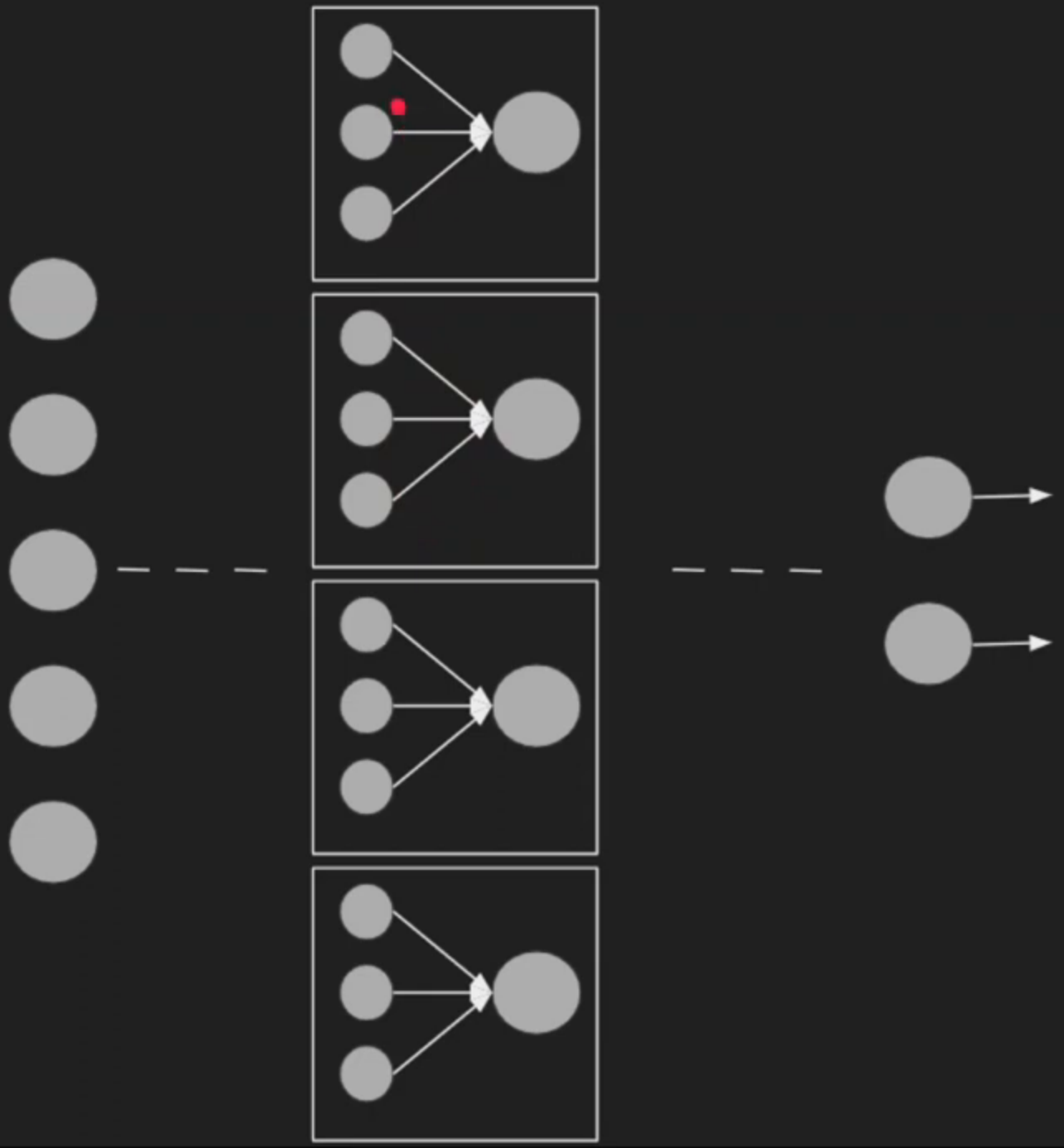
Input Hidden Output



Number of outputs = Number of Groups

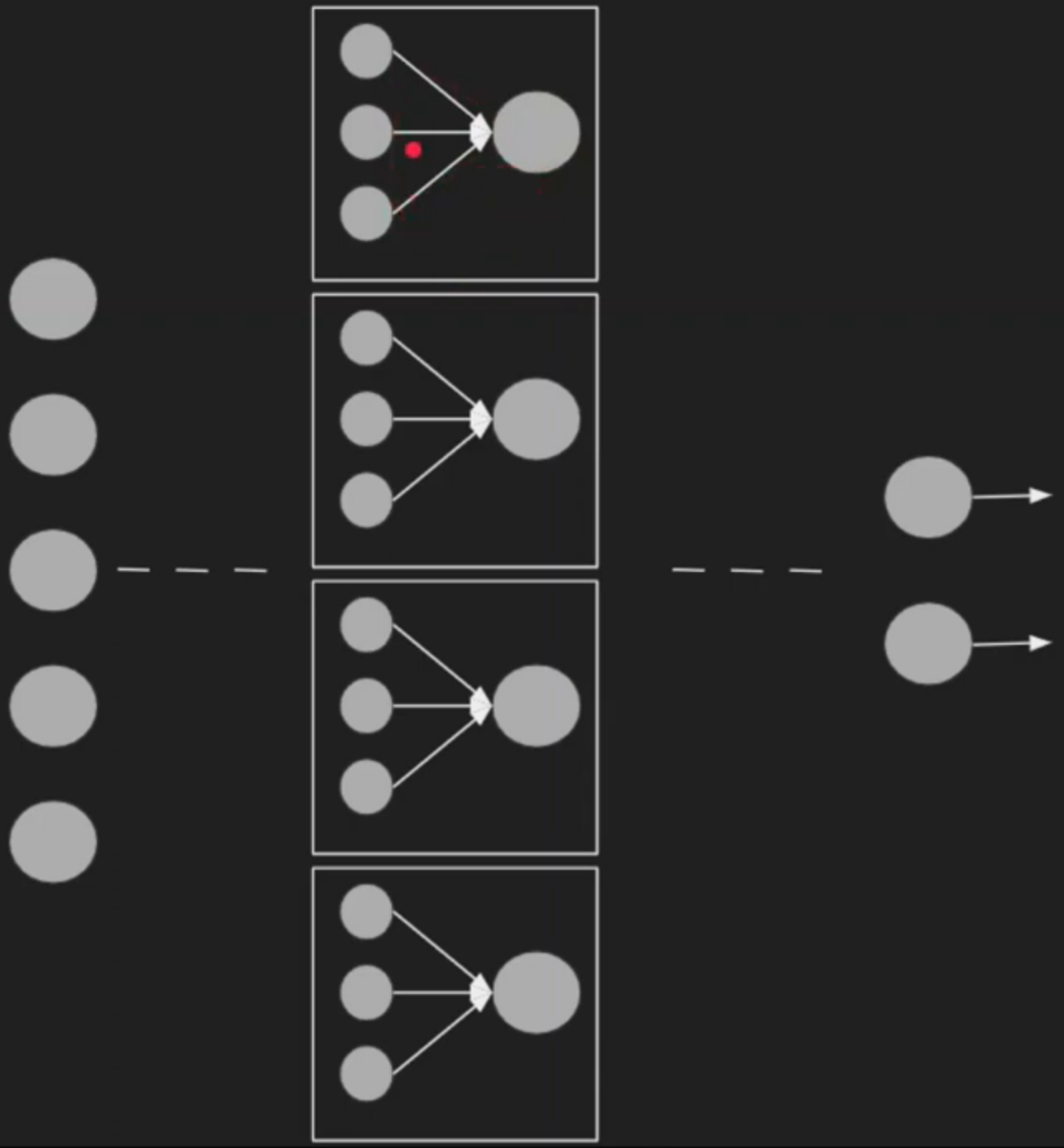
Network - (5,4,2)

Input Hidden Output



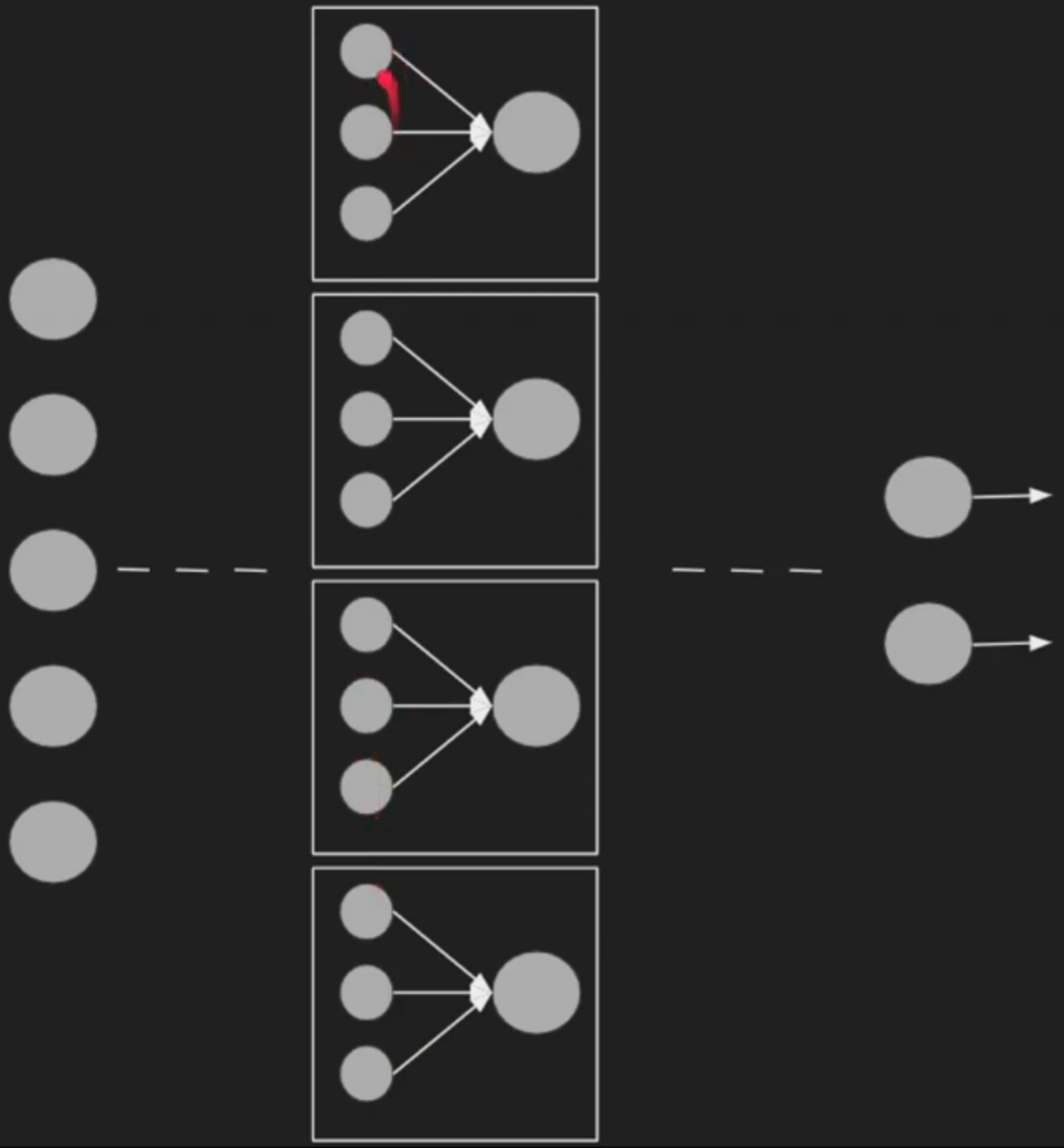
Network - (5,4,2)

Input Hidden Output



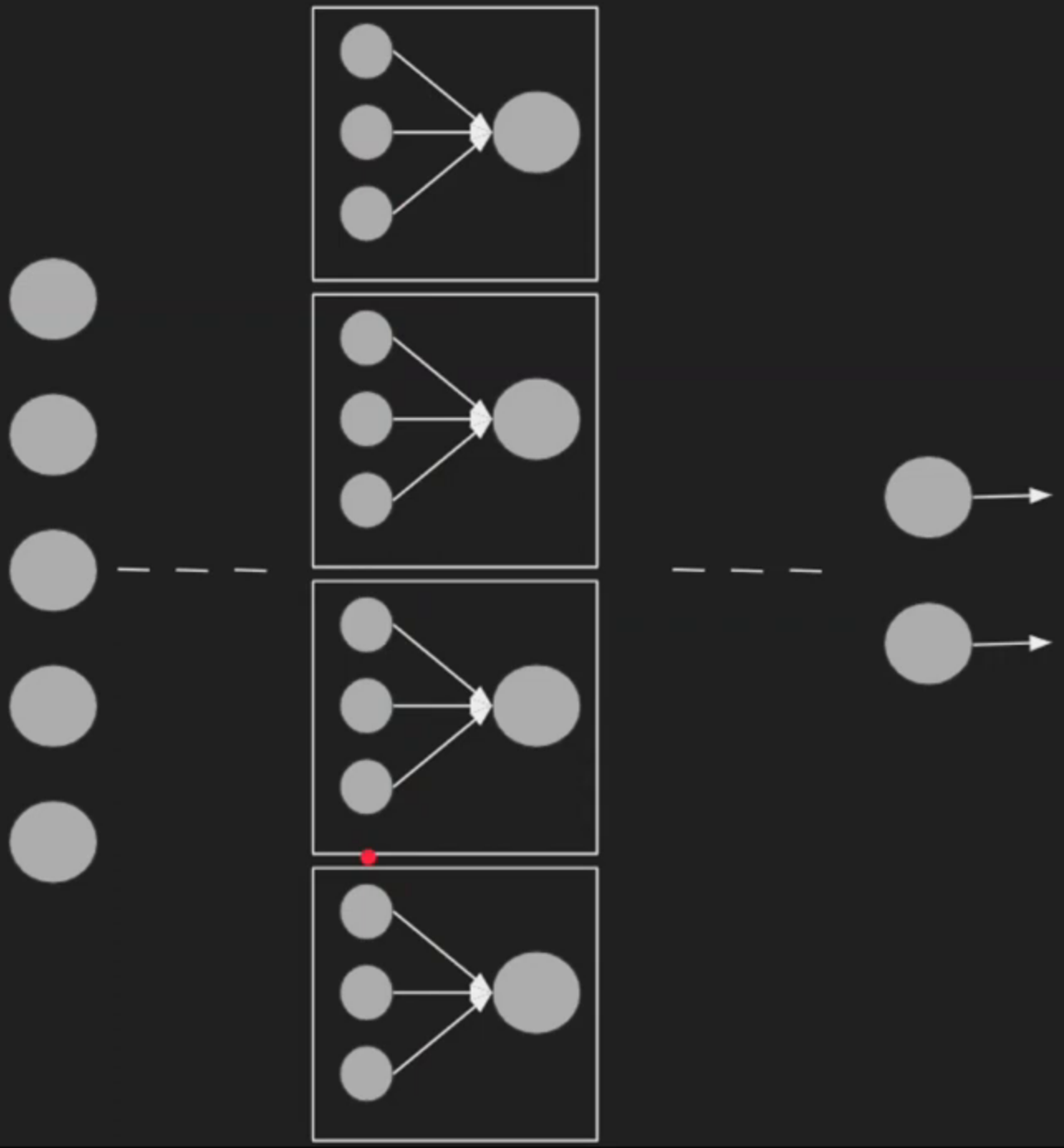
Network - (5,4,2)

Input Hidden Output



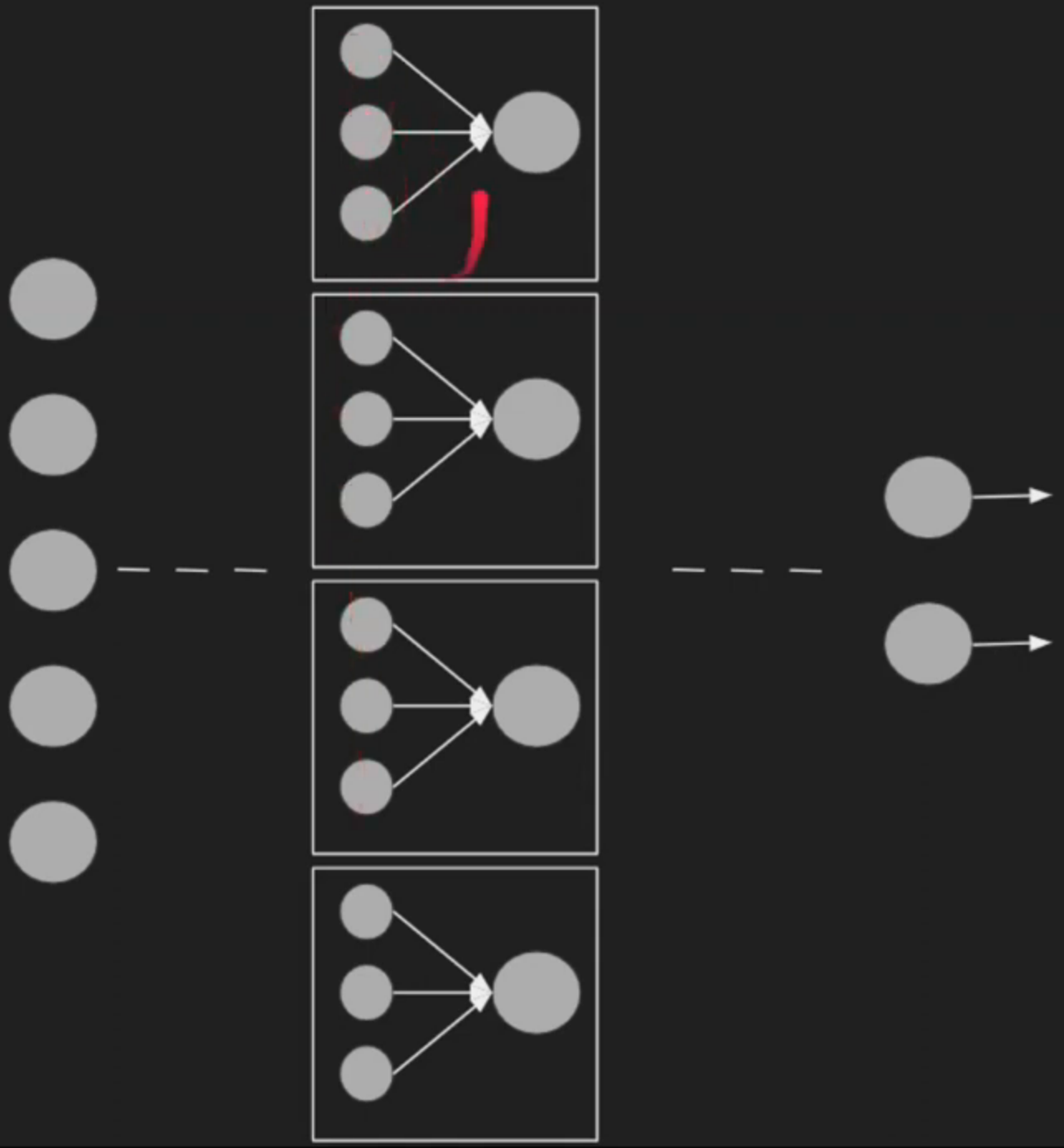
Network - (5,4,2)

Input Hidden Output



Network - (5,4,2)

Input Hidden Output

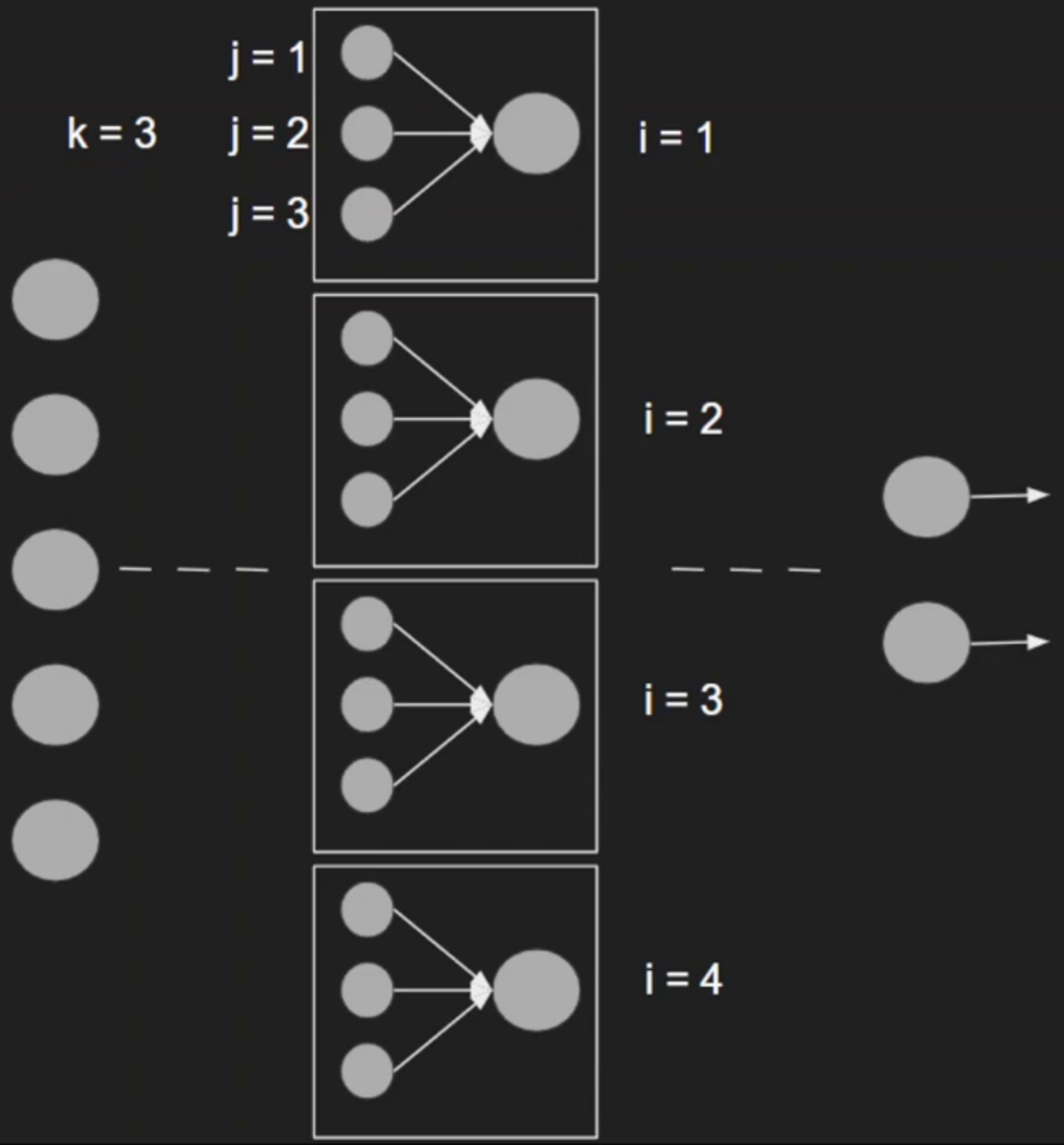


Notation

$$z_{ij} = x^T W_{...ij} + b_{ij}$$

$$h_i(x) = \max_{j \in [1,k]} z_{ij}$$

i - Index for outputs/groups
j - Index for neurons in a group
k - Number of neurons in a group

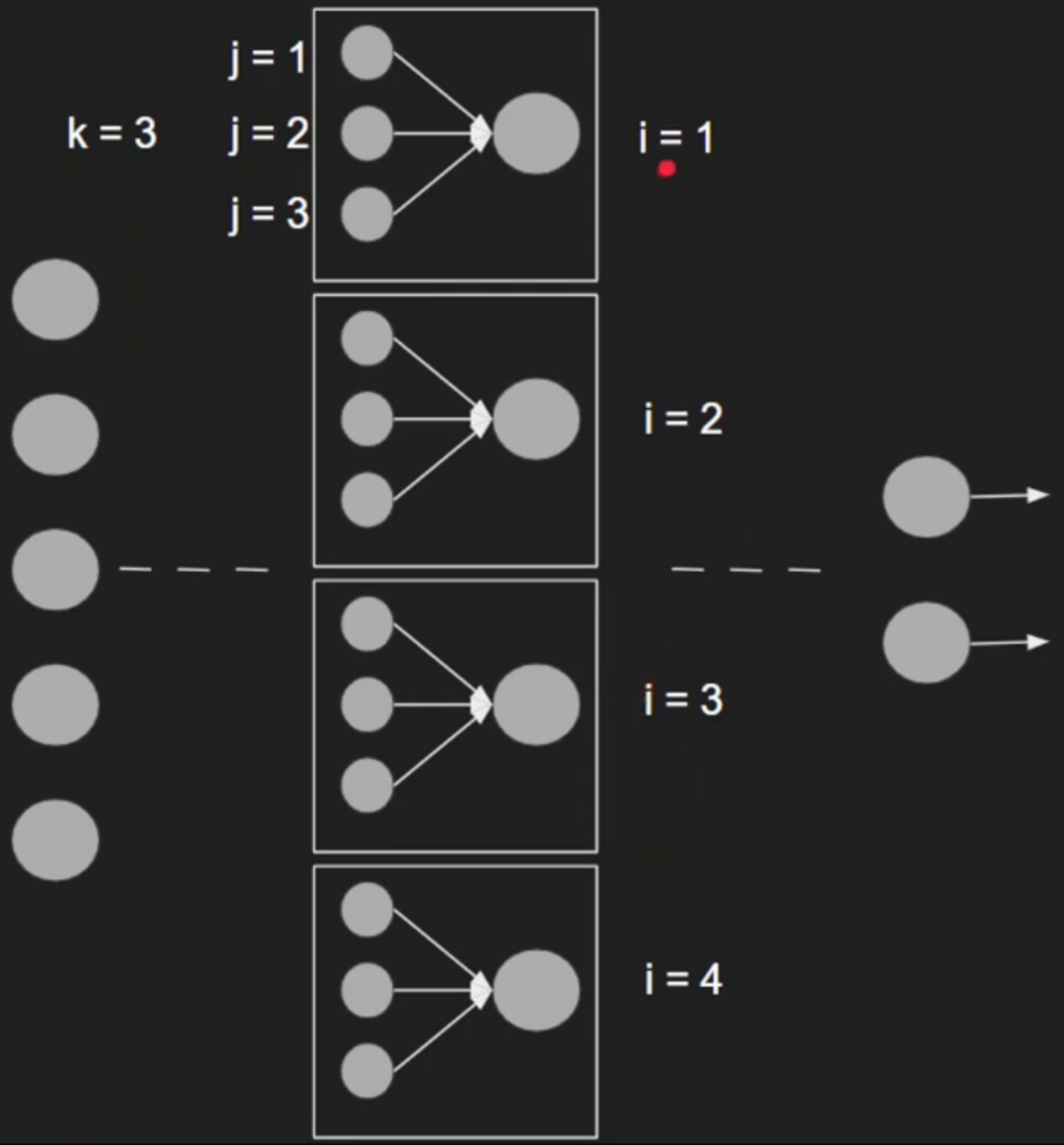


Notation

$$z_{ij} = x^T W_{...ij} + b_{ij}$$

$$h_i(x) = \max_{j \in [1,k]} z_{ij}$$

i - Index for outputs/groups
j - Index for neurons in a group
k - Number of neurons in a group

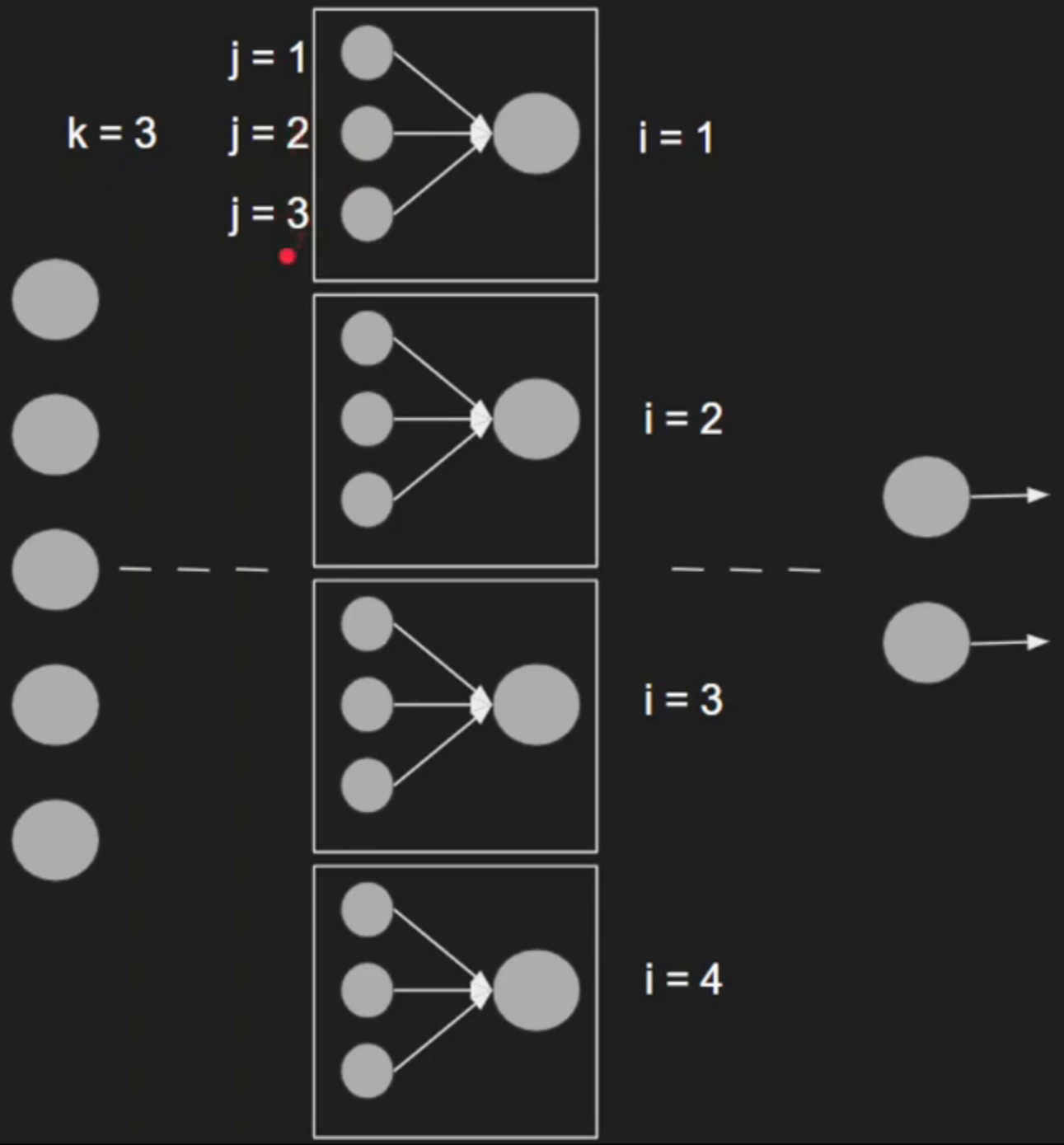


Notation

$$z_{ij} = x^T W_{...ij} + b_{ij}$$

$$h_i(x) = \max_{j \in [1,k]} z_{ij}$$

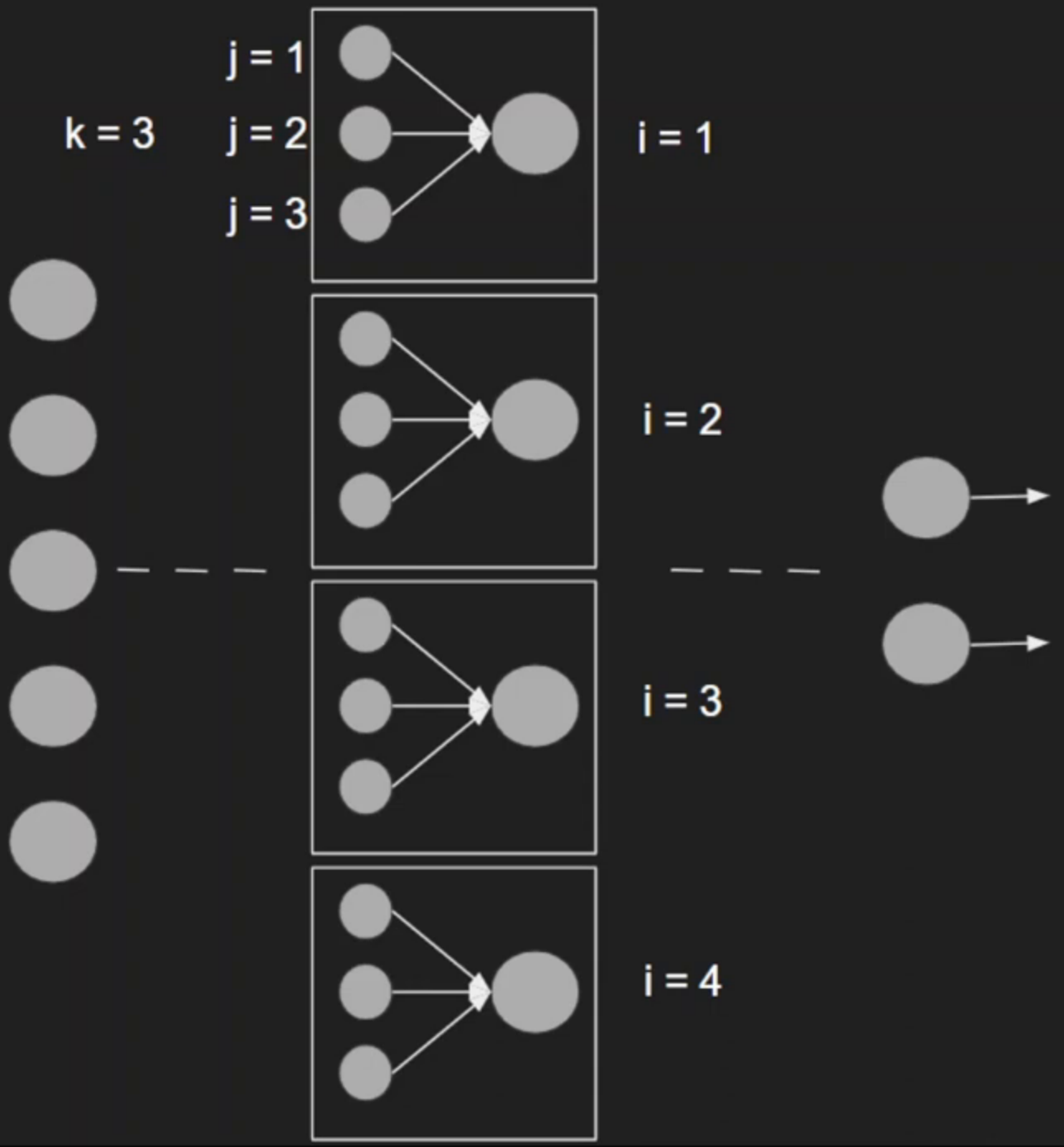
i - Index for outputs/groups
j - Index for neurons in a group
k - Number of neurons in a group



Notation

$$z_{ij} = x^T W_{...ij} + b_{ij}$$

$$h_i(x) = \max_{j \in [1,k]} z_{ij}$$



i - Index for outputs/groups
j - Index for neurons in a group
k - Number of neurons in a group

Example

If k=5

$$z1 = x^T W_1 + b_1$$

$$z2 = x^T W_2 + b_2$$

$$z3 = x^T W_3 + b_3$$

$$z4 = x^T W_4 + b_4$$

$$z5 = x^T W_5 + b_5$$

$$z = max(z1, z2, z3, z4, z5)$$

→ 1 Maxout unit

Example

If k=5

$$z1 = x^T W_1 + b_1$$

$$z2 = x^T W_2 + b_2$$

$$z3 = x^T W_3 + b_3$$

$$z4 = x^T W_4 + b_4$$

$$z5 = x^T W_5 + b_5$$

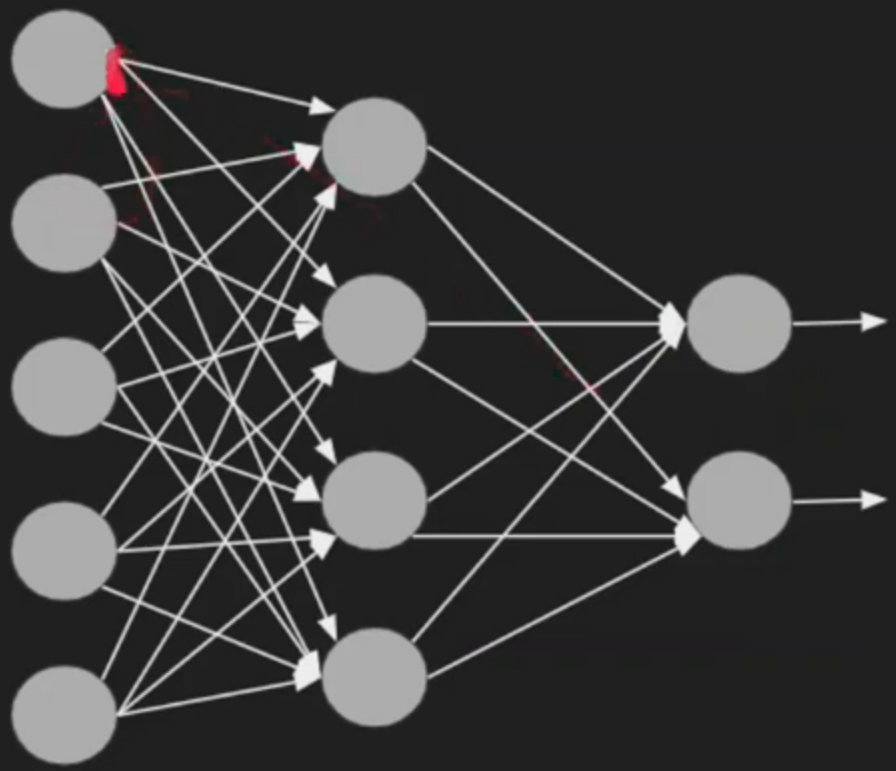
$$z = max(z1, z2, z3, z4, z5)$$

→ 1 Maxout unit

Example - ReLU

Input - [2,5]
Weights - [4,5]
Biases - [4]

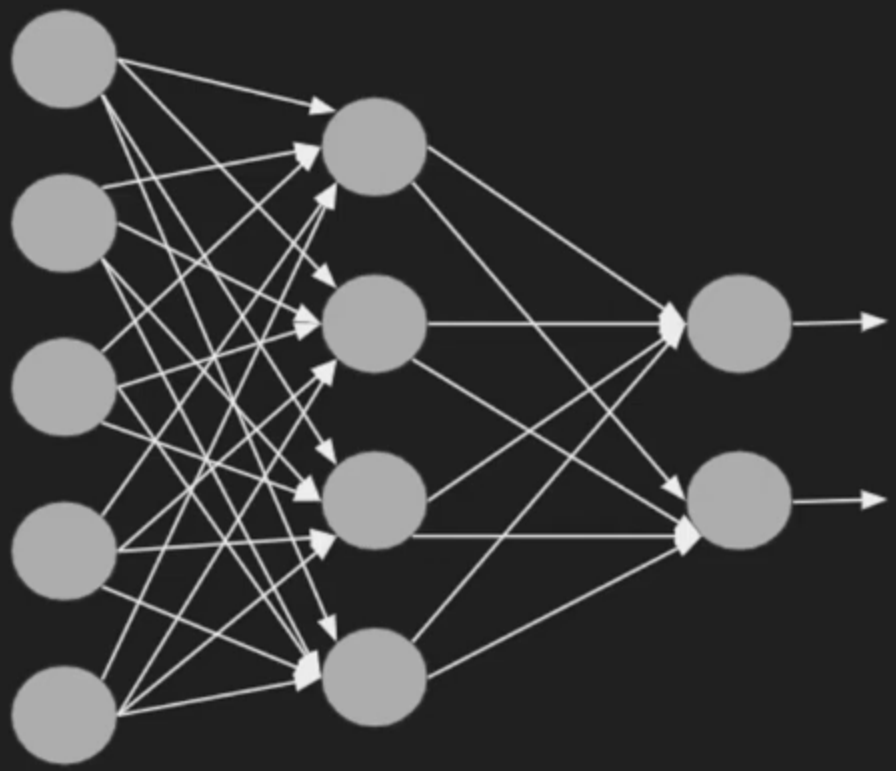
Batch size = 2
Input features = 5
Hidden neurons = 4



Example - ReLU

Input - [2,5]
Weights - [4,5]
Biases - [4]

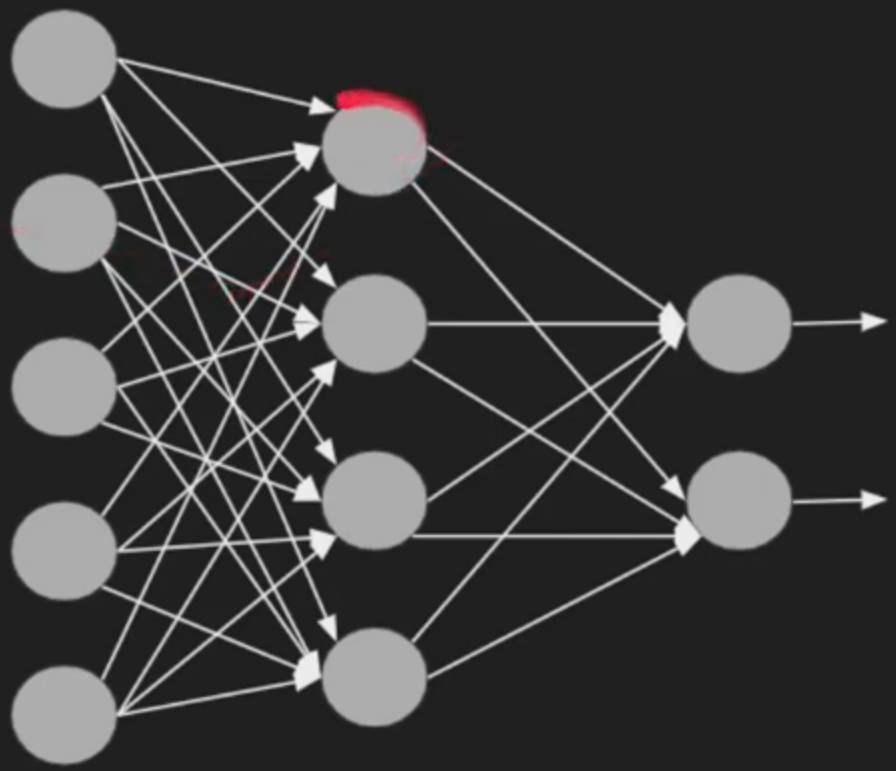
Batch size = 2
Input features = 5
Hidden neurons = 4



Example - ReLU

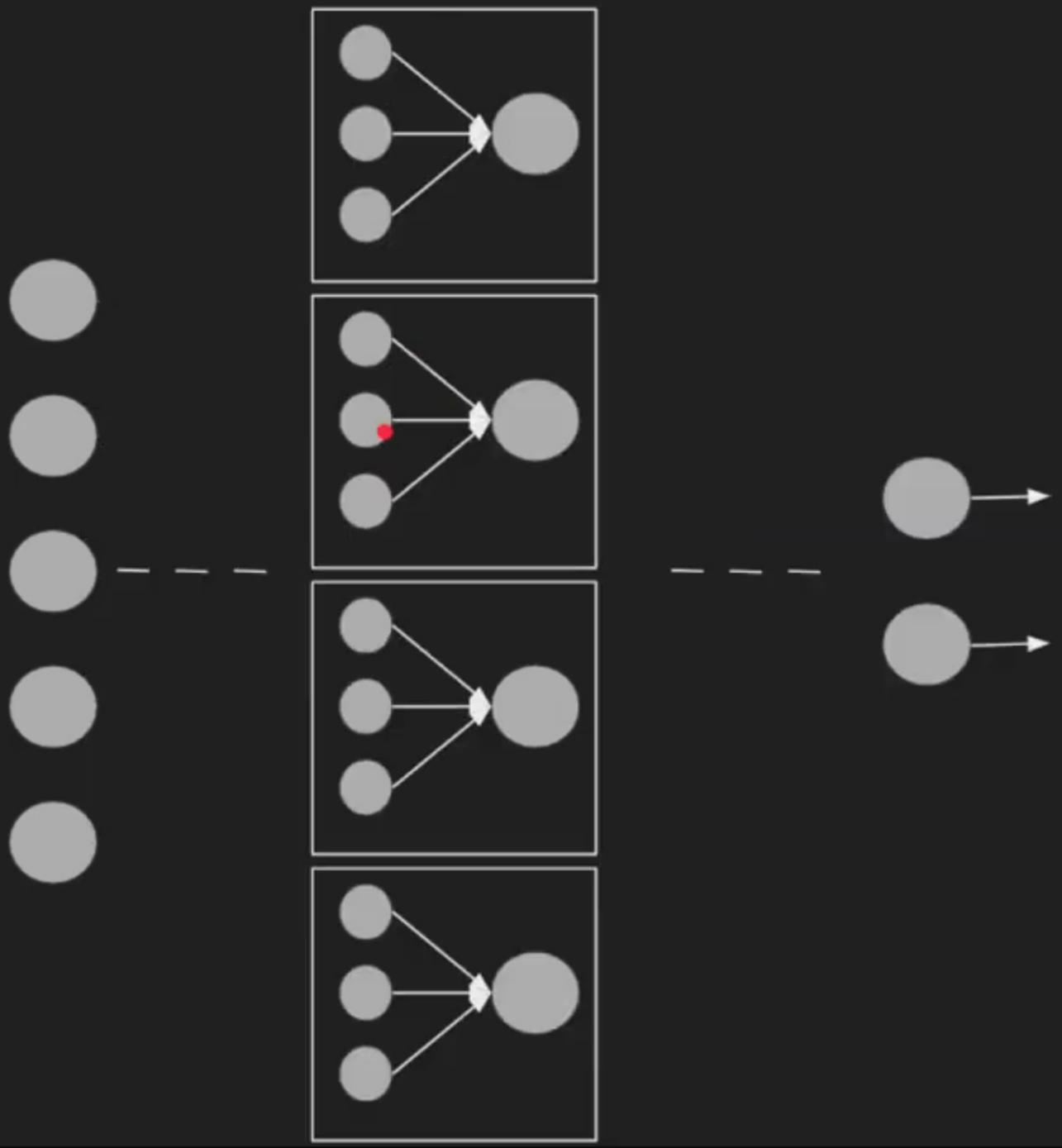
Input - [2,5]
Weights - [4,5]
Biases - [4]

Batch size = 2
Input features = 5
Hidden neurons = 4



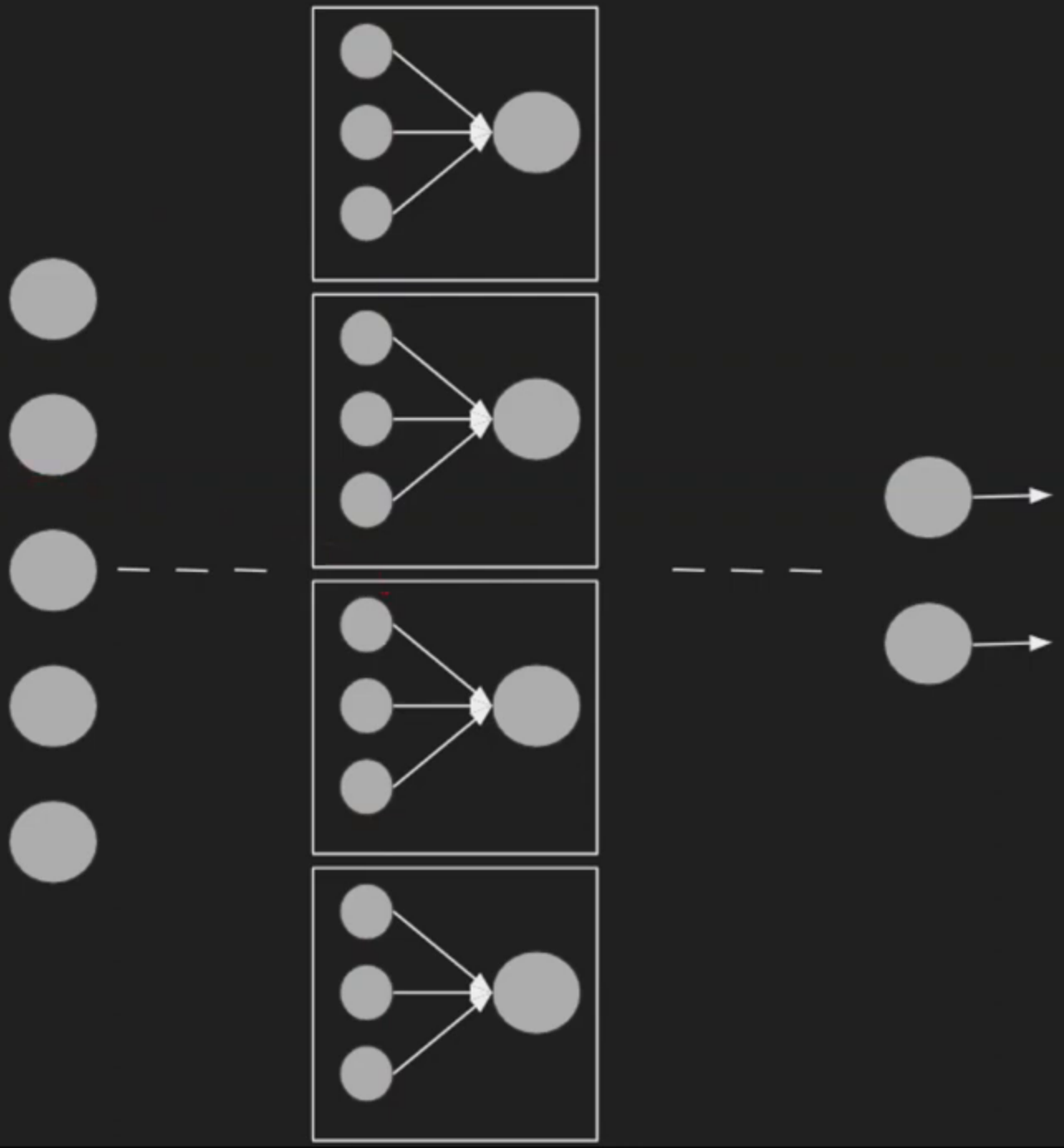
Example - Maxout

Input - [2,5]
Weights - [12,5]
Biases - [12]



Example - Maxout

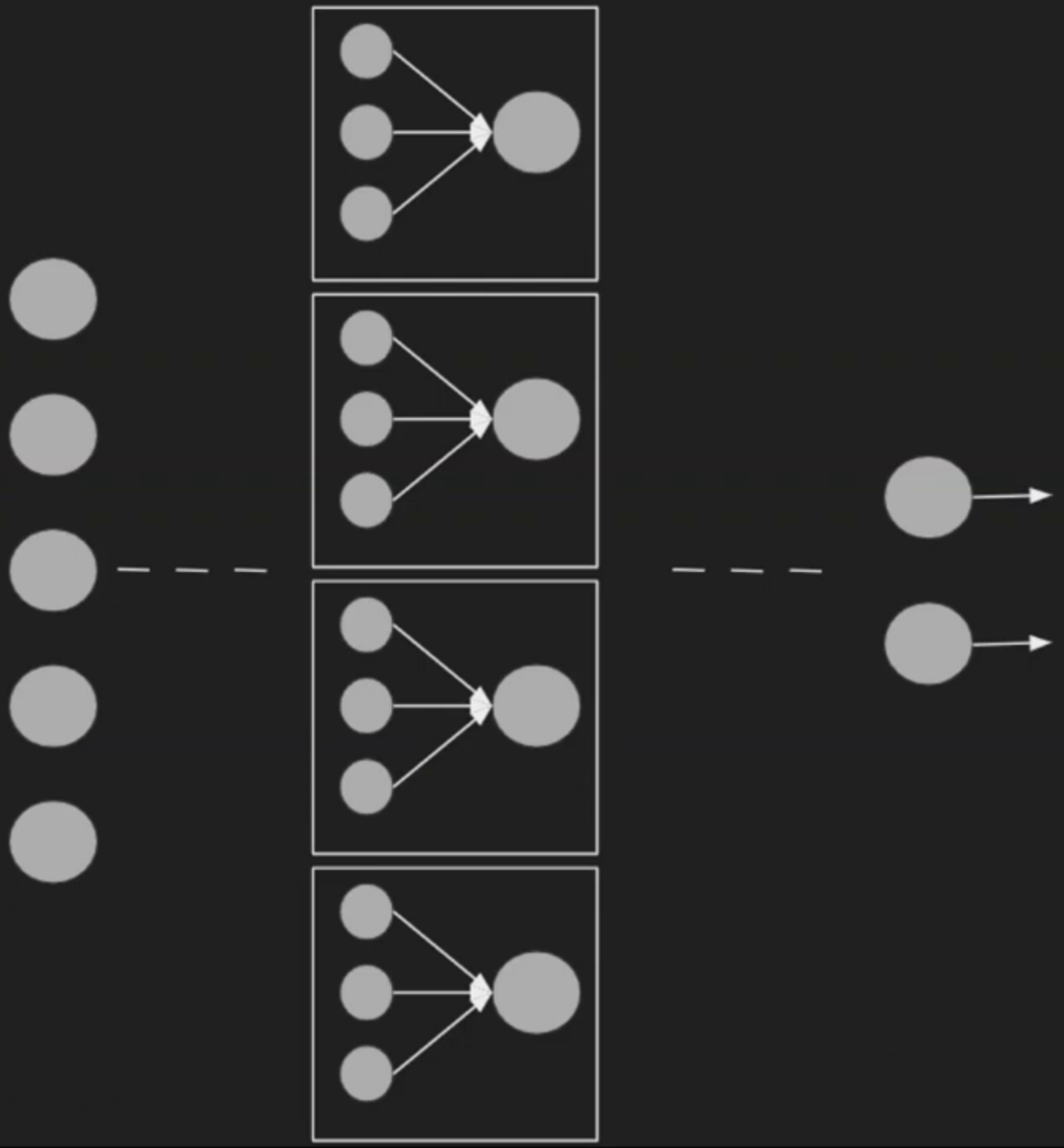
Input - [2,5]
Weights - [12,5]
Biases - [12]



Example - Maxout

Input - [2,5]
Weights - [12,5]
Biases - [12]

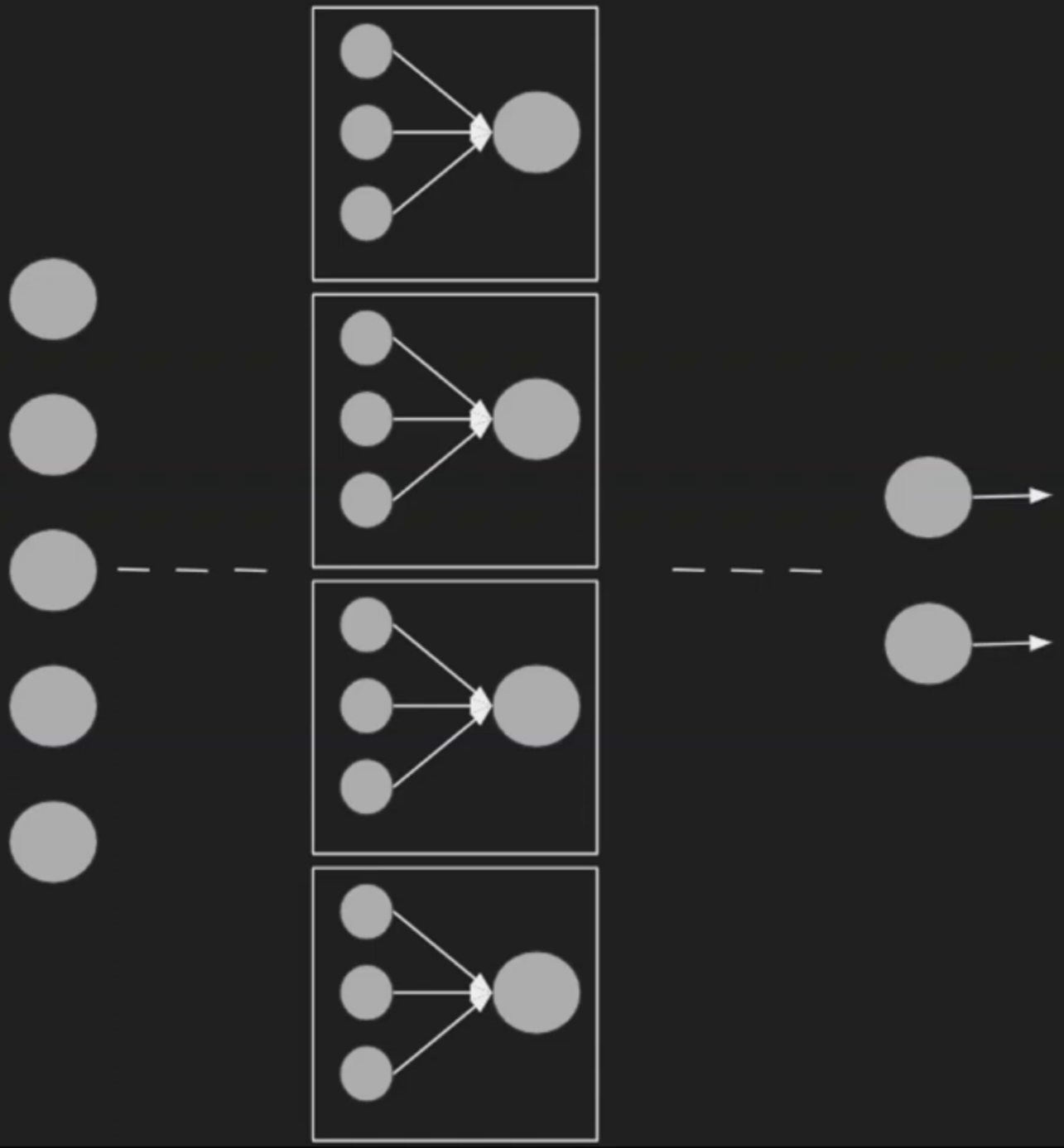
Z - [2,12] - [2,4,3]
O = max(Z, axis=2) - [2,4]



Example - Maxout

Input - [2,5]
Weights - [12,5]
Biases - [12]

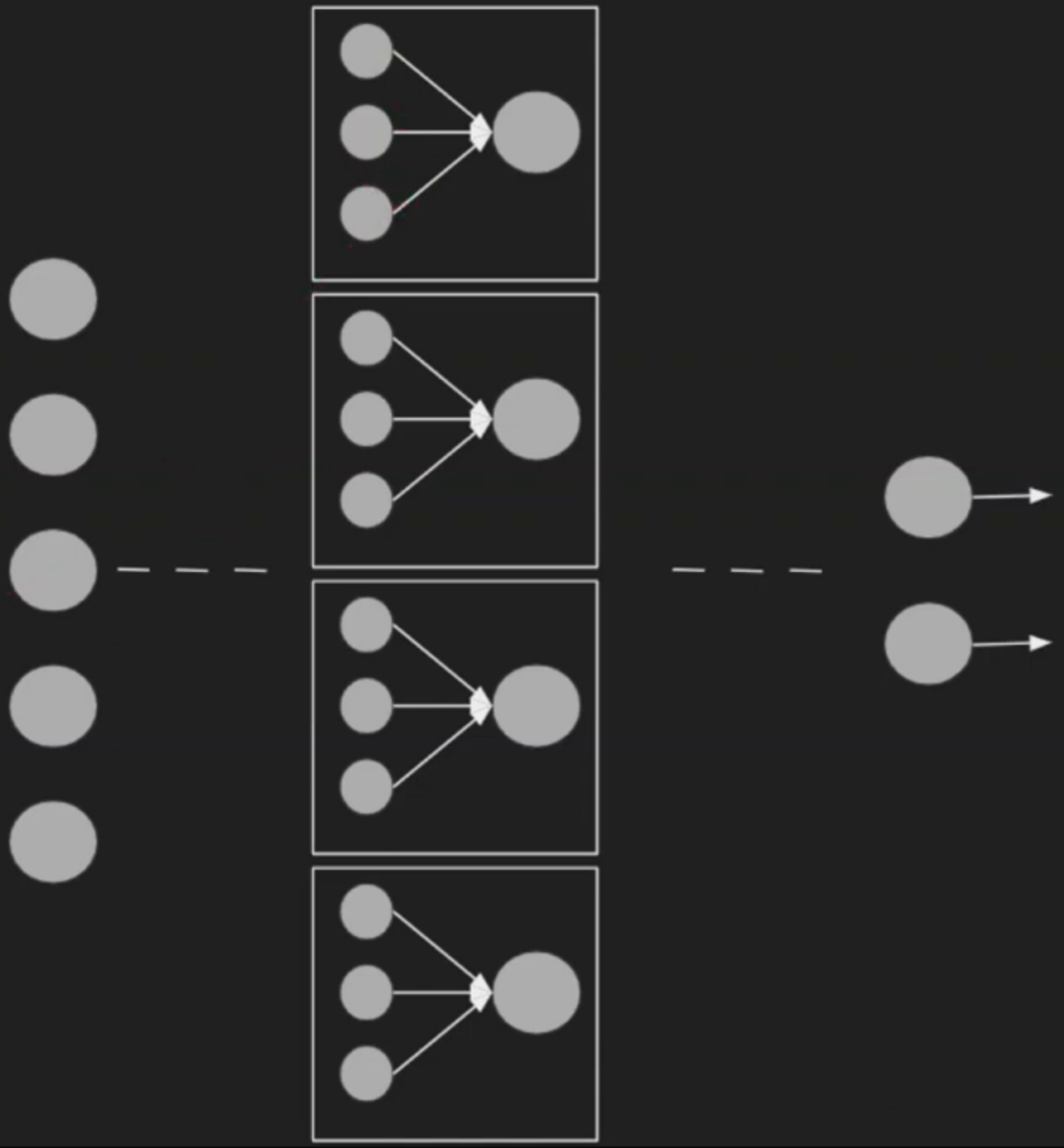
Z - [2,12] - [2,4,3]
O = max(Z, axis=2) - [2,4]



Example - Maxout

Input - [2,5]
Weights - [12,5]
Biases - [12]

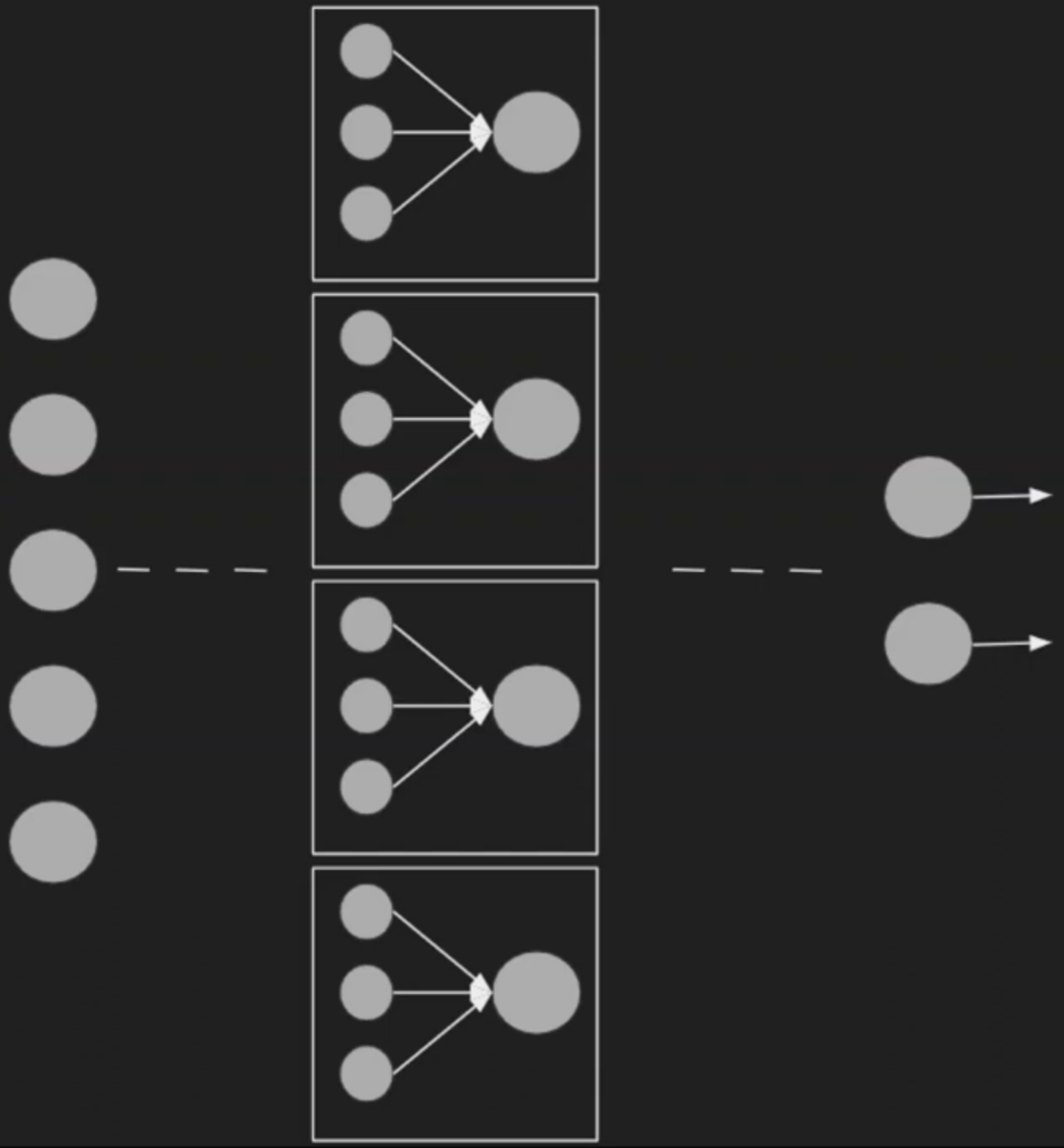
Z - [2,12] - [2,4,3]
O = max(Z, axis=2) - [2,4]



Example - Maxout

Input - [2,5]
Weights - [12,5]
Biases - [12]

Z - [2,12] - [2,4,3]
O = max(Z, axis=2) - [2,4]



Implementation

```
import numpy as np

x = np.random.random((2,5))
print(x.shape)

# 12 hidden neurons
W = np.random.random((12,5))
b = np.random.random((12))
print(W.shape,b.shape)
z = np.dot(x, W.T) + b
print(z.shape)
z_ = z.reshape((2,4,3))
o = np.max(z_, axis=2)
print(o.shape)
```

```
(2, 5)
(12, 5) (12,)
(2, 12)
(2, 4)
```

Implementation

```python
import numpy as np

x = np.random.random((2,5))
print(x.shape)

# 12 hidden neurons
W = np.random.random((12,5))
b = np.random.random((12))
print(W.shape,b.shape)
z = np.dot(x, W.T) + b
print(z.shape)
z_ = z.reshape((2,4,3))
o = np.max(z_, axis=2)
print(o.shape)
```

```
(2, 5)
(12, 5) (12,)
(2, 12)
(2, 4)
```

Implementation

```
import numpy as np

x = np.random.random((2,5))
print(x.shape)

# 12 hidden neurons
W = np.random.random((12,5))
b = np.random.random((12))
print(W.shape,b.shape)
z = np.dot(x, W.T) + b
print(z.shape)
z_ = z.reshape((2,4,3))
o = np.max(z_, axis=2)
print(o.shape)
```

```
(2, 5)
(12, 5) (12,)
(2, 12)
(2, 4)
```

Implementation

```python
import numpy as np

x = np.random.random((2,5))
print(x.shape)

# 12 hidden neurons
W = np.random.random((12,5))
b = np.random.random((12))
print(W.shape,b.shape)
z = np.dot(x, W.T) + b
print(z.shape)
z_ = z.reshape((2,4,3))
o = np.max(z_, axis=2)
print(o.shape)
```

```
(2, 5)
(12, 5) (12,)
(2, 12)
(2, 4)
```

Implementation

```python
import numpy as np

x = np.random.random((2,5))
print(x.shape)

# 12 hidden neurons
W = np.random.random((12,5))
b = np.random.random((12))
print(W.shape,b.shape)
z = np.dot(x, W.T) + b
print(z.shape)
z_ = z.reshape((2,4,3))
o = np.max(z_, axis=2)
print(o.shape)
```

```
(2, 5)
(12, 5) (12,)
(2, 12)
(2, 4)
```

Advantages

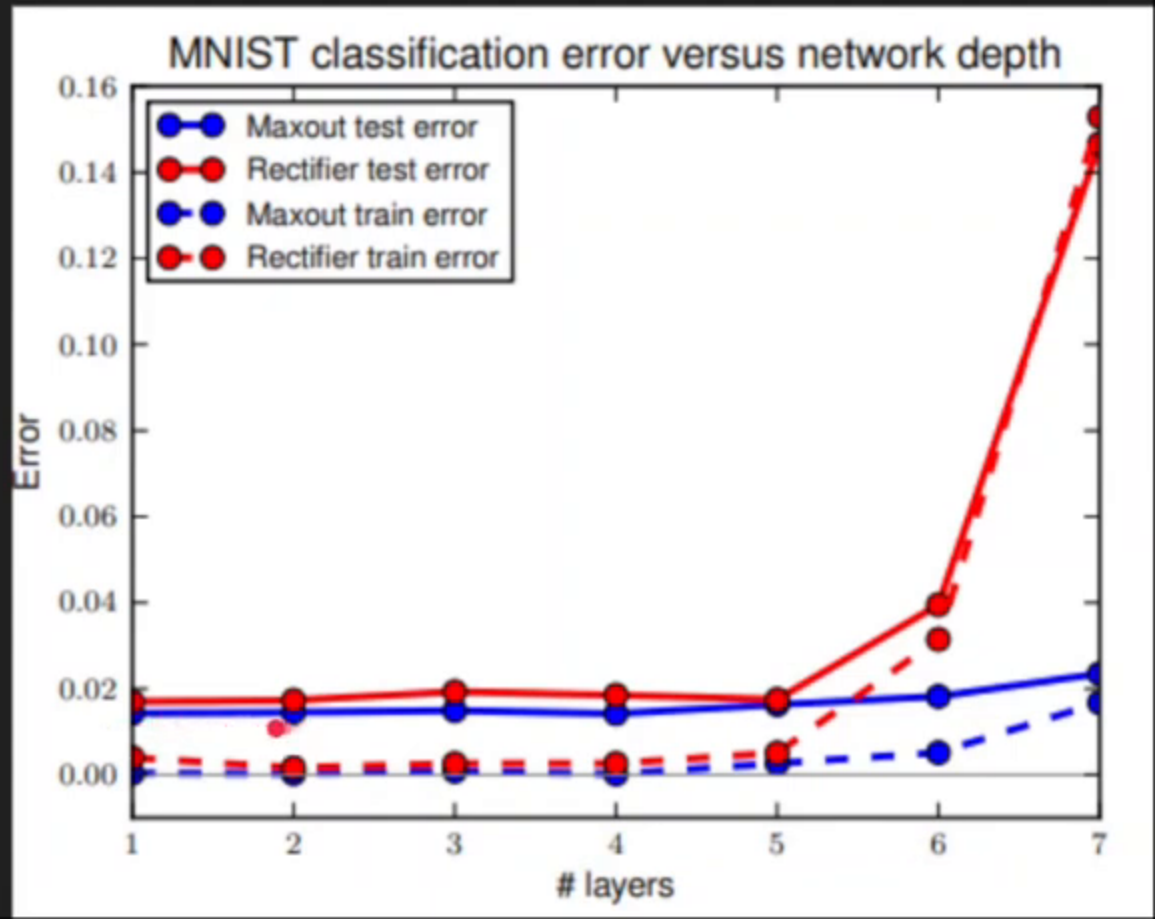
- Universal Approximator
- Can generalize ReLU and Leaky ReLU
- Works better than ReLU

Advantages

- Universal Approximator
- Can generalize ReLU and Leaky ReLU
- Works better than ReLU

Advantages

- Universal Approximator
- Can generalize ReLU and Leaky ReLU
- Works better than ReLU



Drawbacks

- Prone to overfitting
- Used along with dropout regularization
- Doubles the number of parameters compared to other activations

Drawbacks

- Prone to overfitting
- Used along with dropout regularization
- Doubles the number of parameters compared to other activations

Drawbacks

- Prone to overfitting
- Used along with dropout regularization
- Doubles the number of parameters compared to other activations

Drawbacks

- Prone to overfitting
- Used along with dropout regularization
- Doubles the number of parameters compared to other activations

Drawbacks

- Prone to overfitting
- Used along with dropout regularization
- Doubles the number of parameters compared to other activations

Drawbacks

- Prone to overfitting
- Used along with dropout regularization
- Doubles the number of parameters compared to other activations

Drawbacks

- Prone to overfitting
- Used along with dropout regularization
- Doubles the number of parameters compared to other activations

Drawbacks

- Prone to overfitting
- Used along with dropout regularization
- Doubles the number of parameters compared to other activations