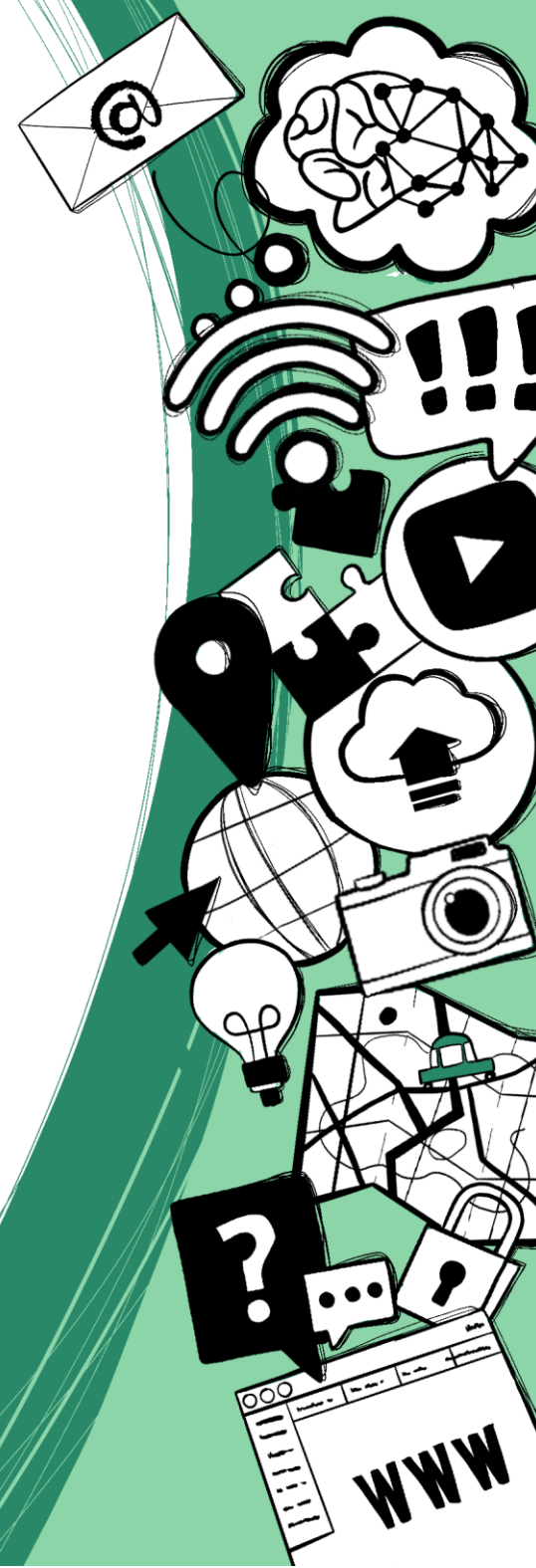


# Neural Networks II

## receptive Fields





On the left is a camera with 4 input fields. On the right are the output fields. The system should be able to distinguish between whole, vertical, diagonal and horizontal patterns.

A "camera", that should be able to detect simple patterns



solid



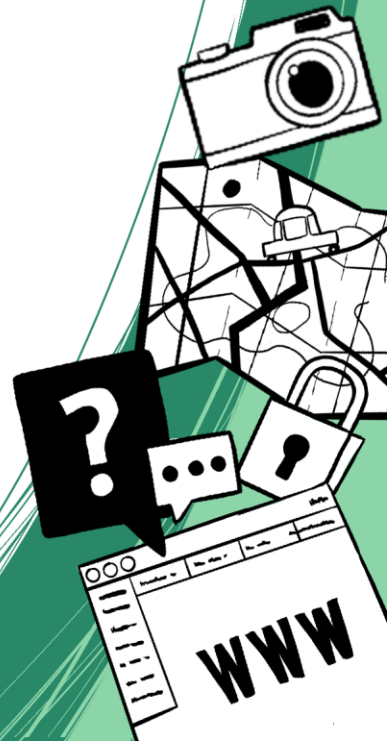
vertical



diagonal

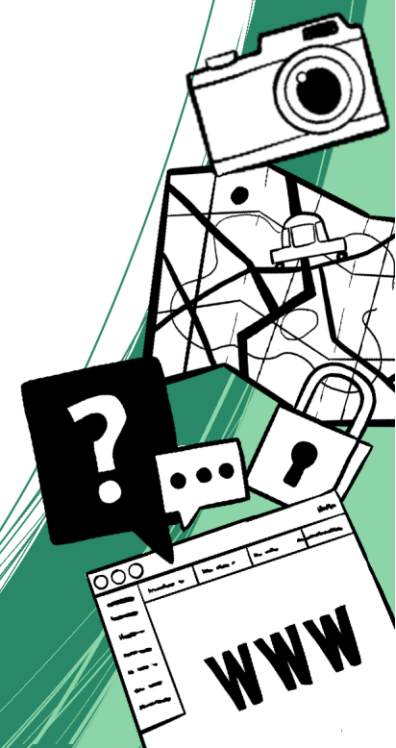


horizontal





The input fields can have continuous values in the range  $[-1, 1]$ .

## Brightness of the input fields

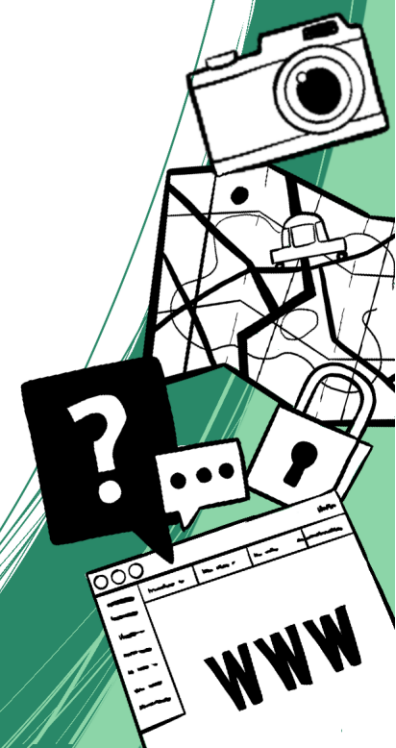


# Example for the recognition of a horizontal pattern

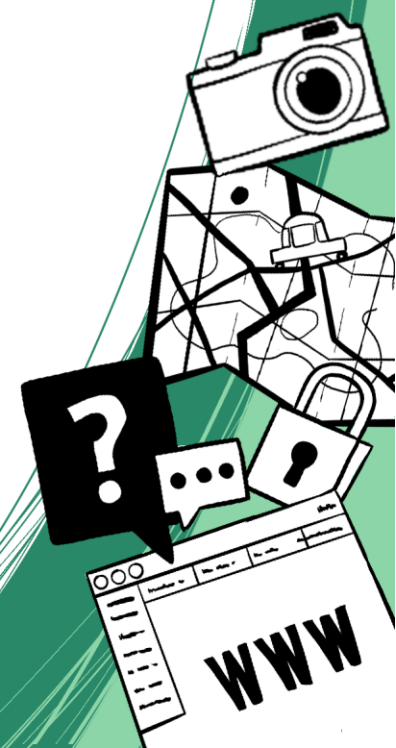
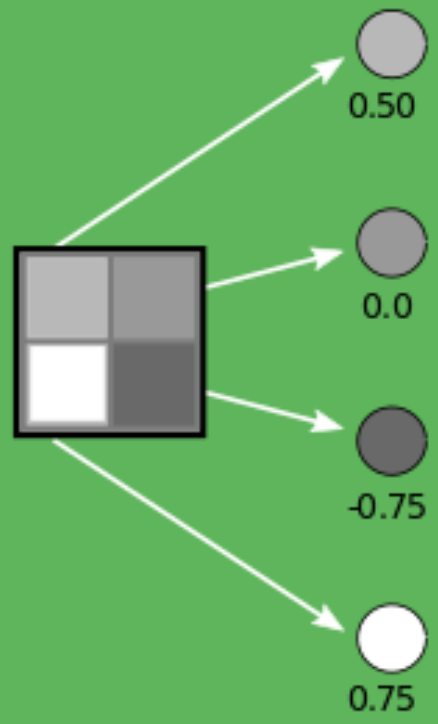
Same category



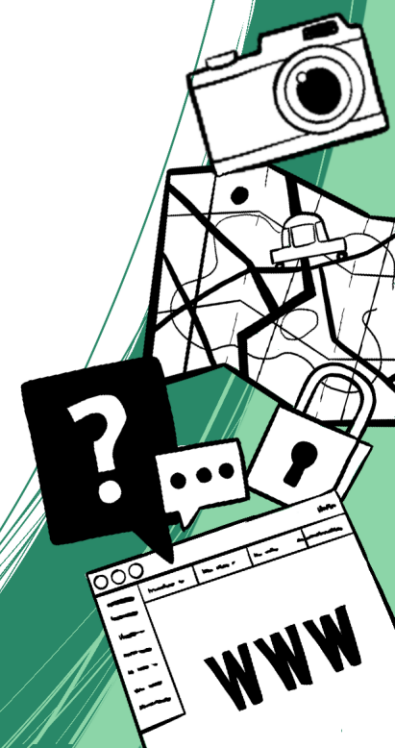
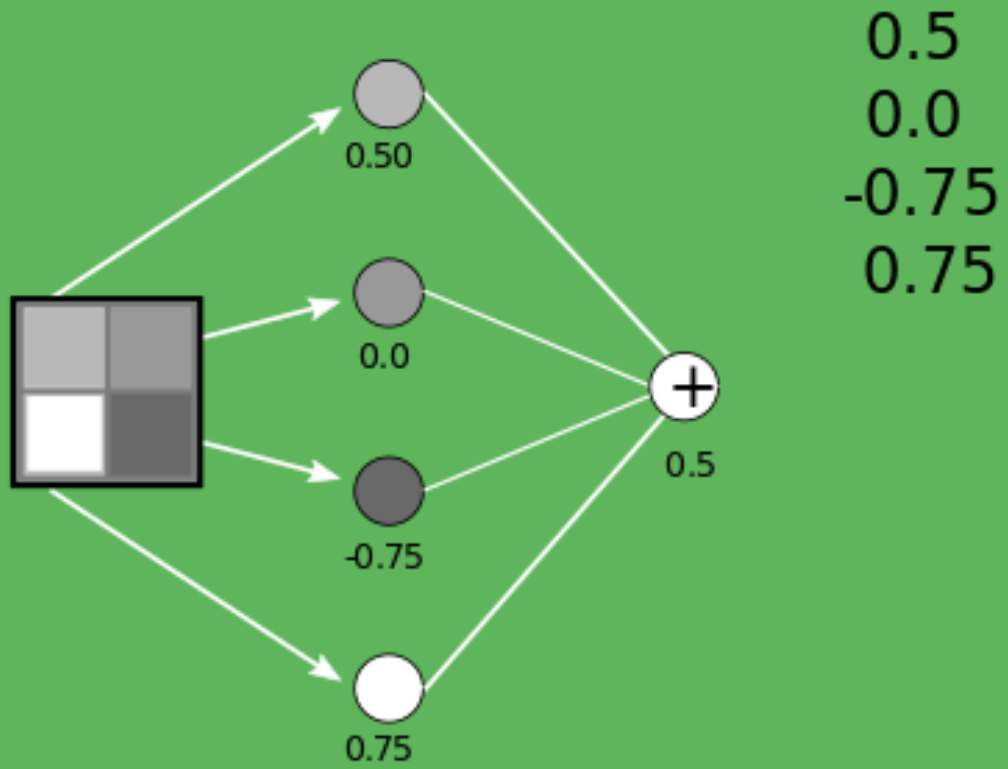
- solid
- vertical
- diagonal
- horizontal



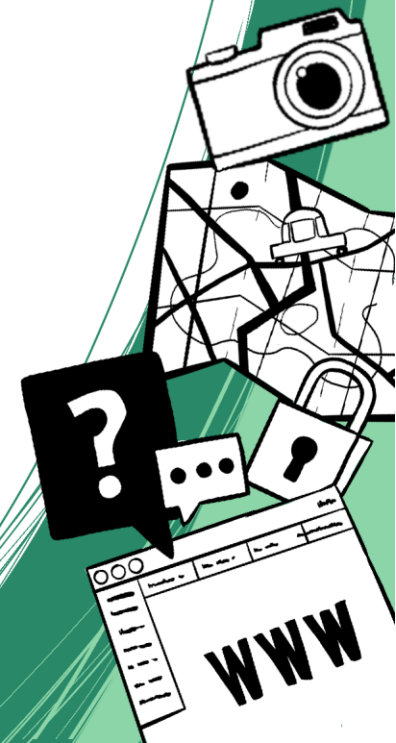
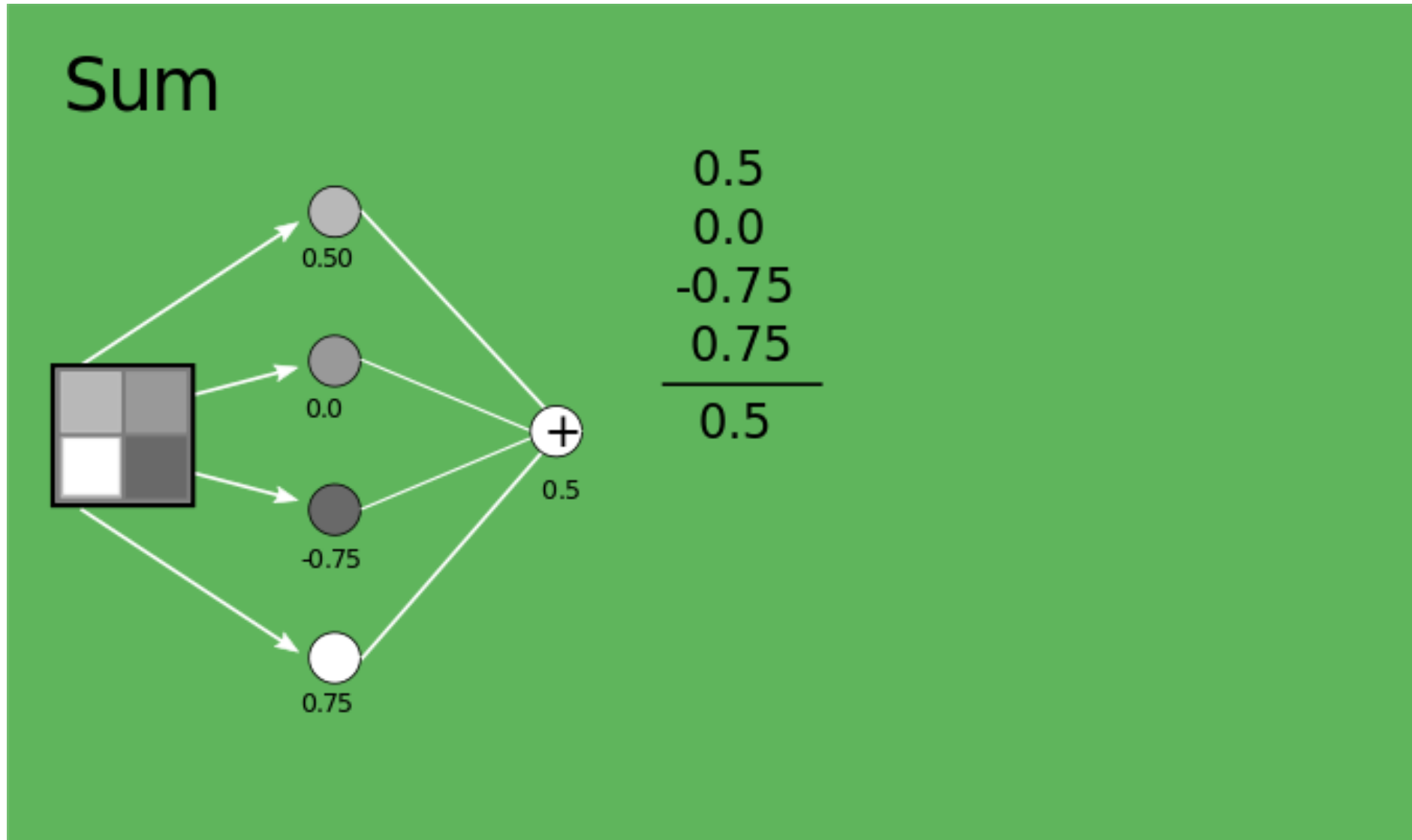
# Input vector



# Sum

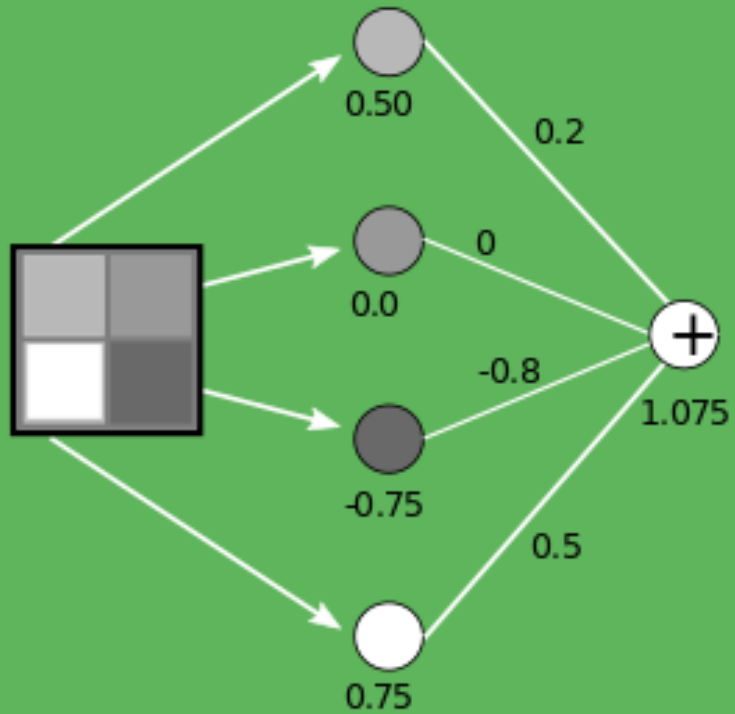


The neuron forms the sum of the input vector.

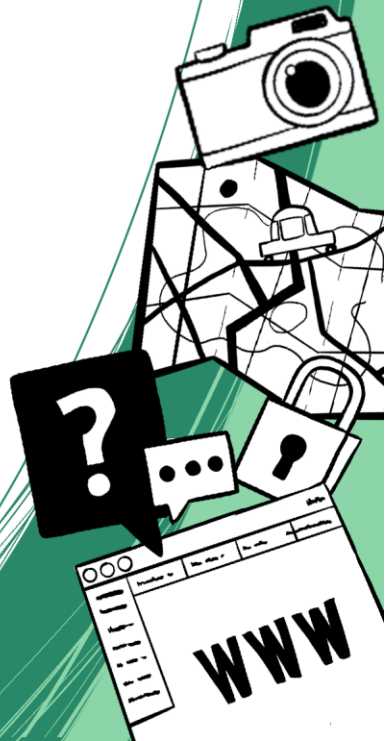


The connections have different weights.

## Weighted connections

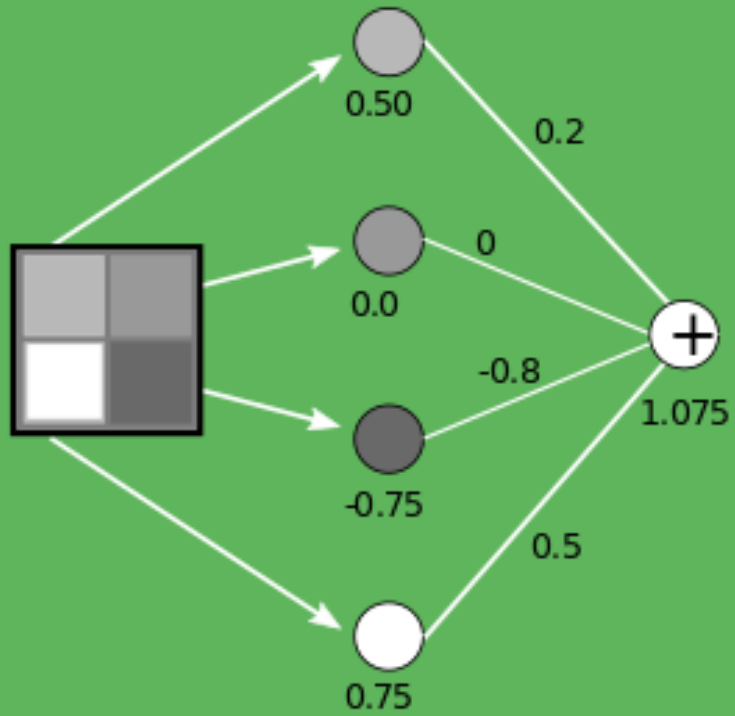


$$\begin{aligned} 0.5 * 0.2 &= 0.1 \\ 0.0 * 0.0 &= 0.0 \\ -0.75 * -0.8 &= 0.6 \\ 0.75 * 0.5 &= 0.375 \end{aligned}$$





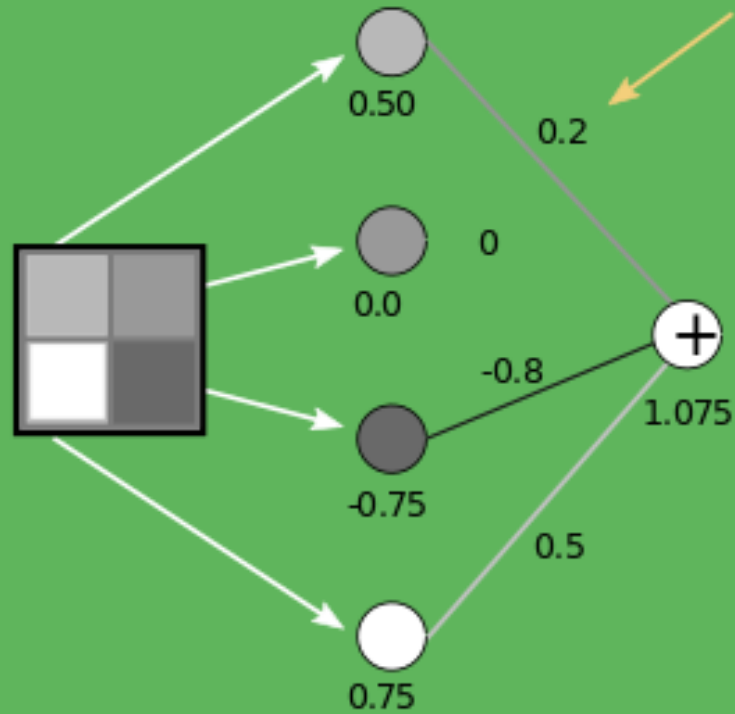
# Sum



$$\begin{array}{r} 0.5 * 0.2 = 0.1 \\ 0.0 * 0.0 = 0.0 \\ -0.75 * -0.8 = 0.6 \\ 0.75 * 0.5 = 0.375 \\ \hline 1.075 \end{array}$$



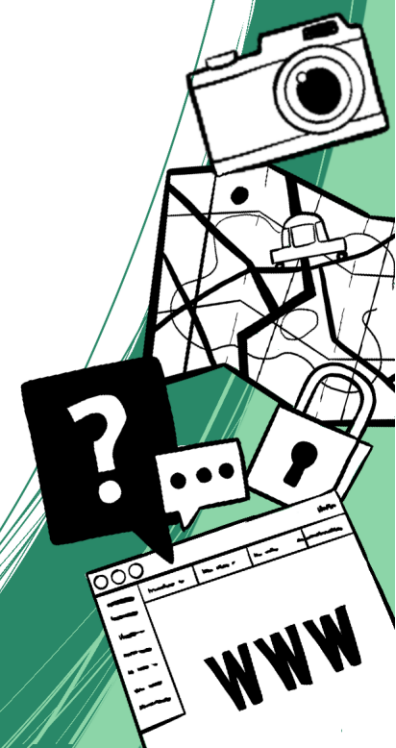
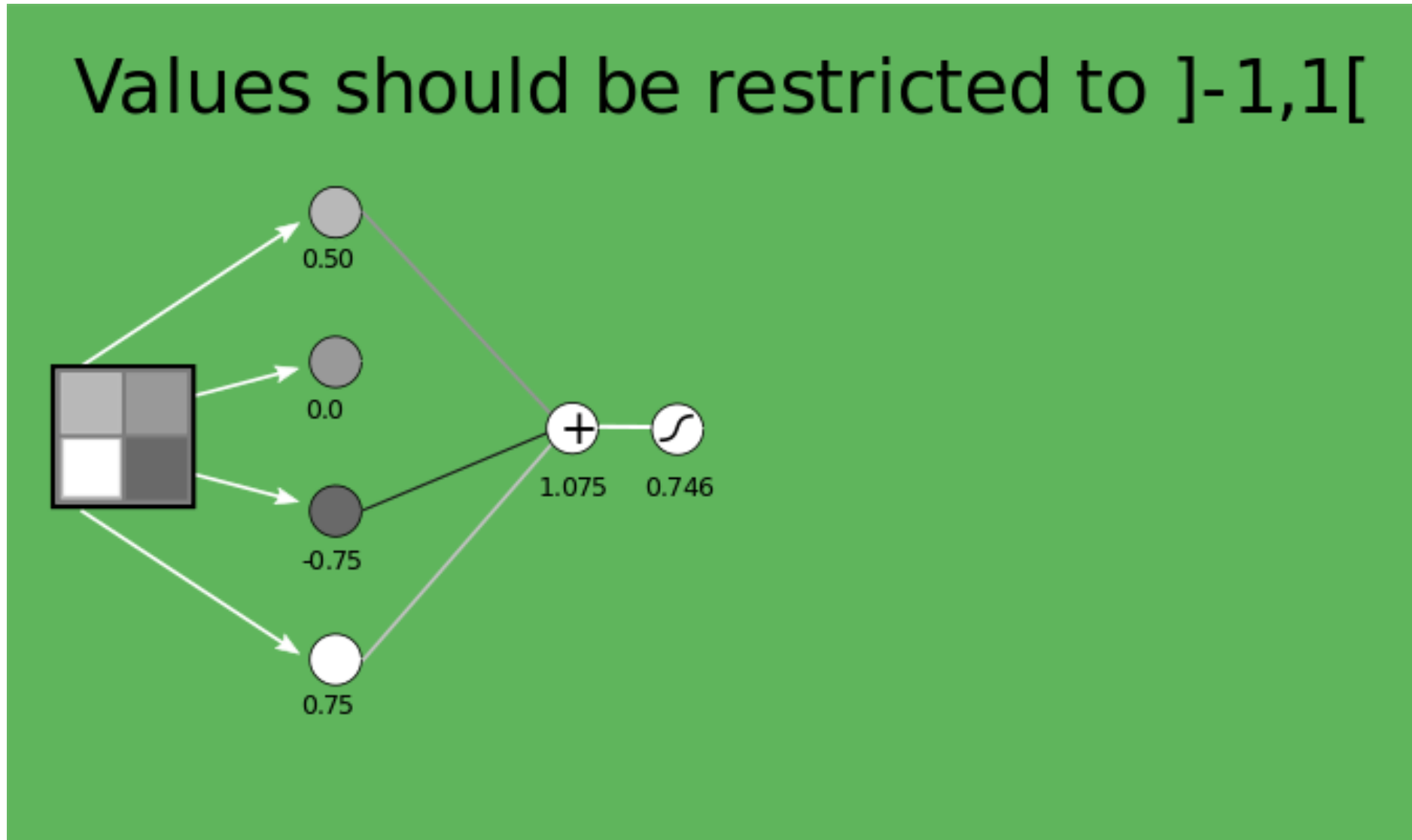
# Visualizing the weights



$$\begin{array}{r} 0.5 * 0.2 = 0.1 \\ 0.0 * 0.0 = 0.0 \\ -0.75 * -0.8 = 0.6 \\ 0.75 * 0.5 = 0.375 \\ \hline 1.075 \end{array}$$



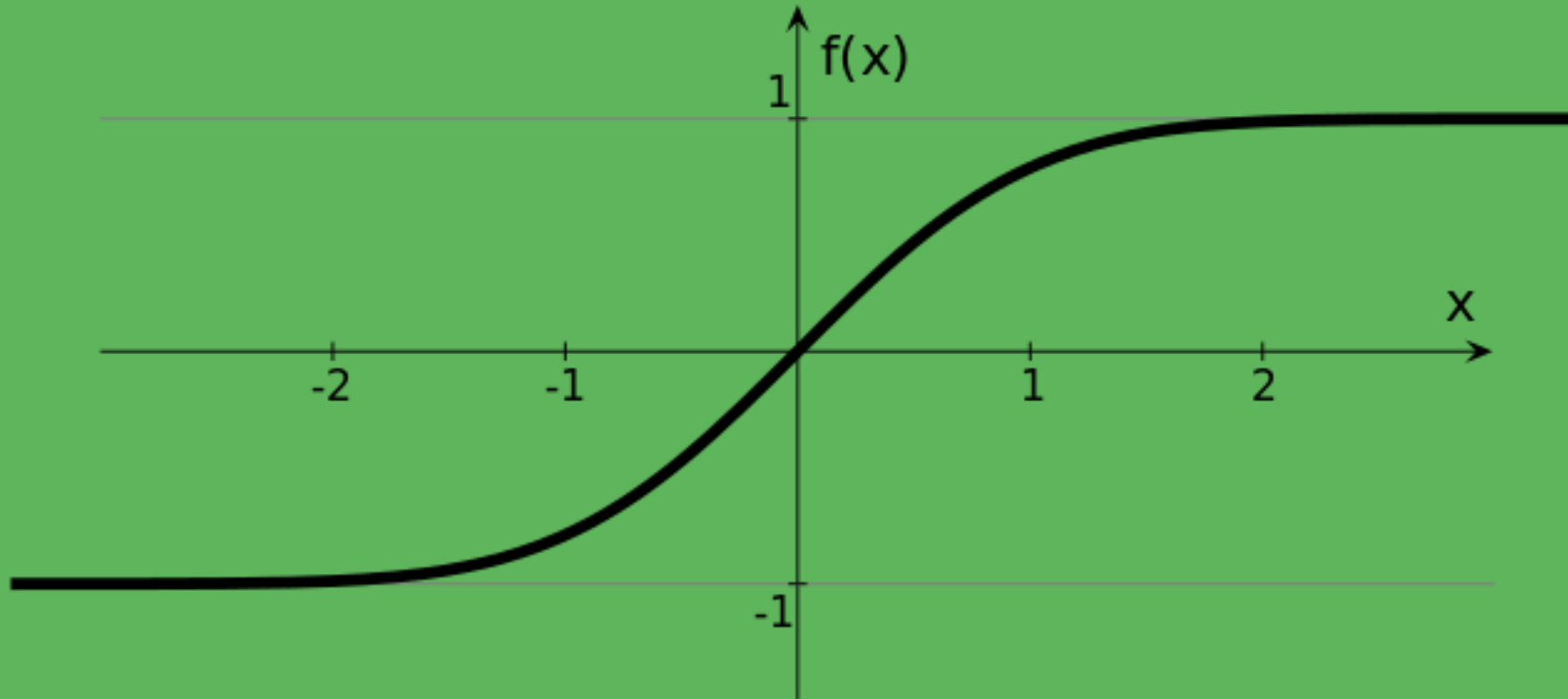
The result value should always remain in the interval  $]1,1[$ . This is ensured by the additional element on the f<sub>right</sub> ( $\sigma$ ). It is used as a restriction and at the same time as an activation function.



advanced content for the restriction function

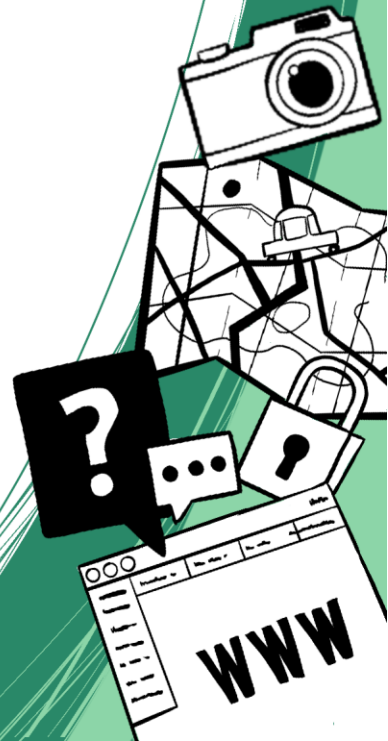
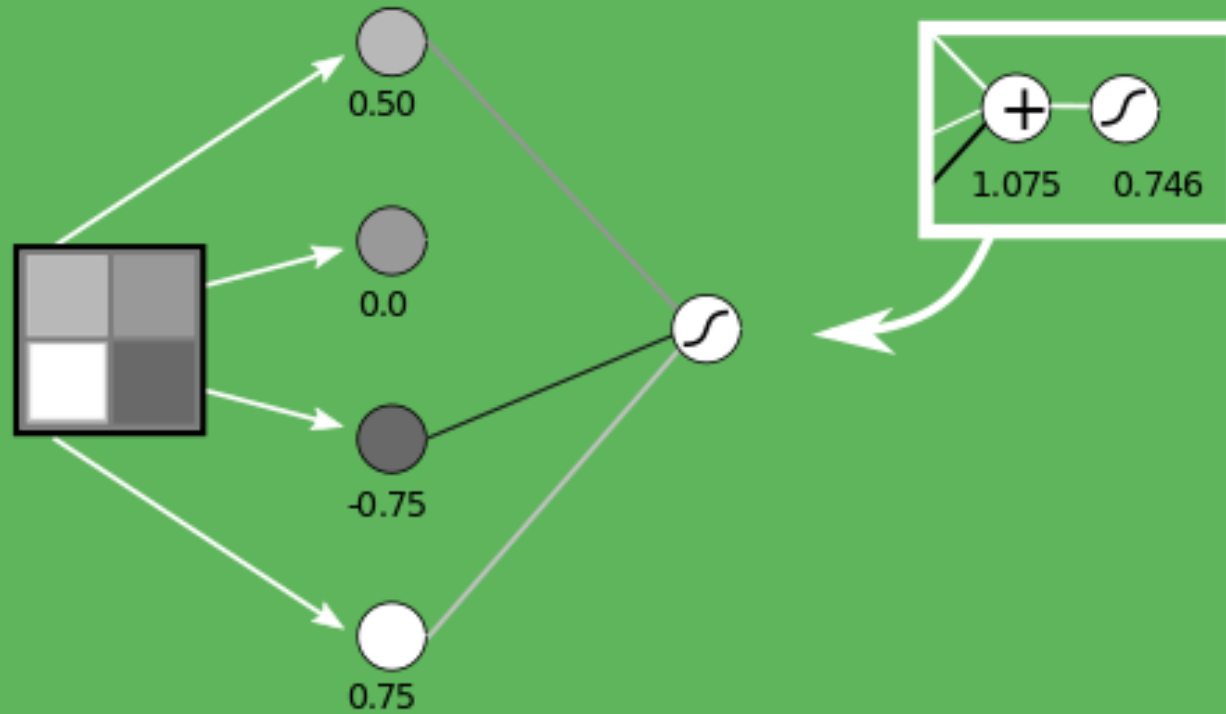
For example, the sigmoid function can be used as a restriction function.

Sigmoid function 

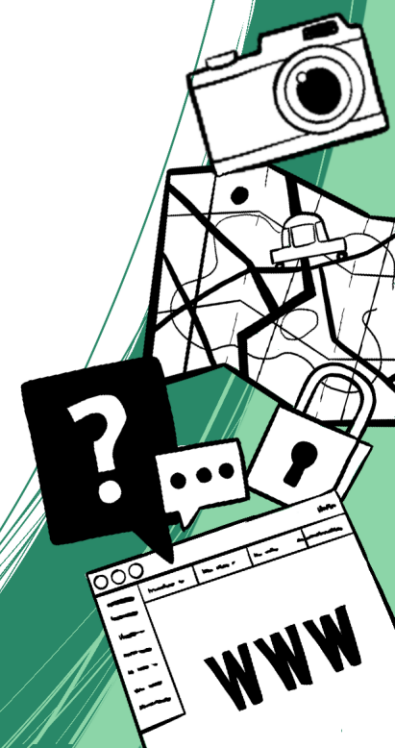
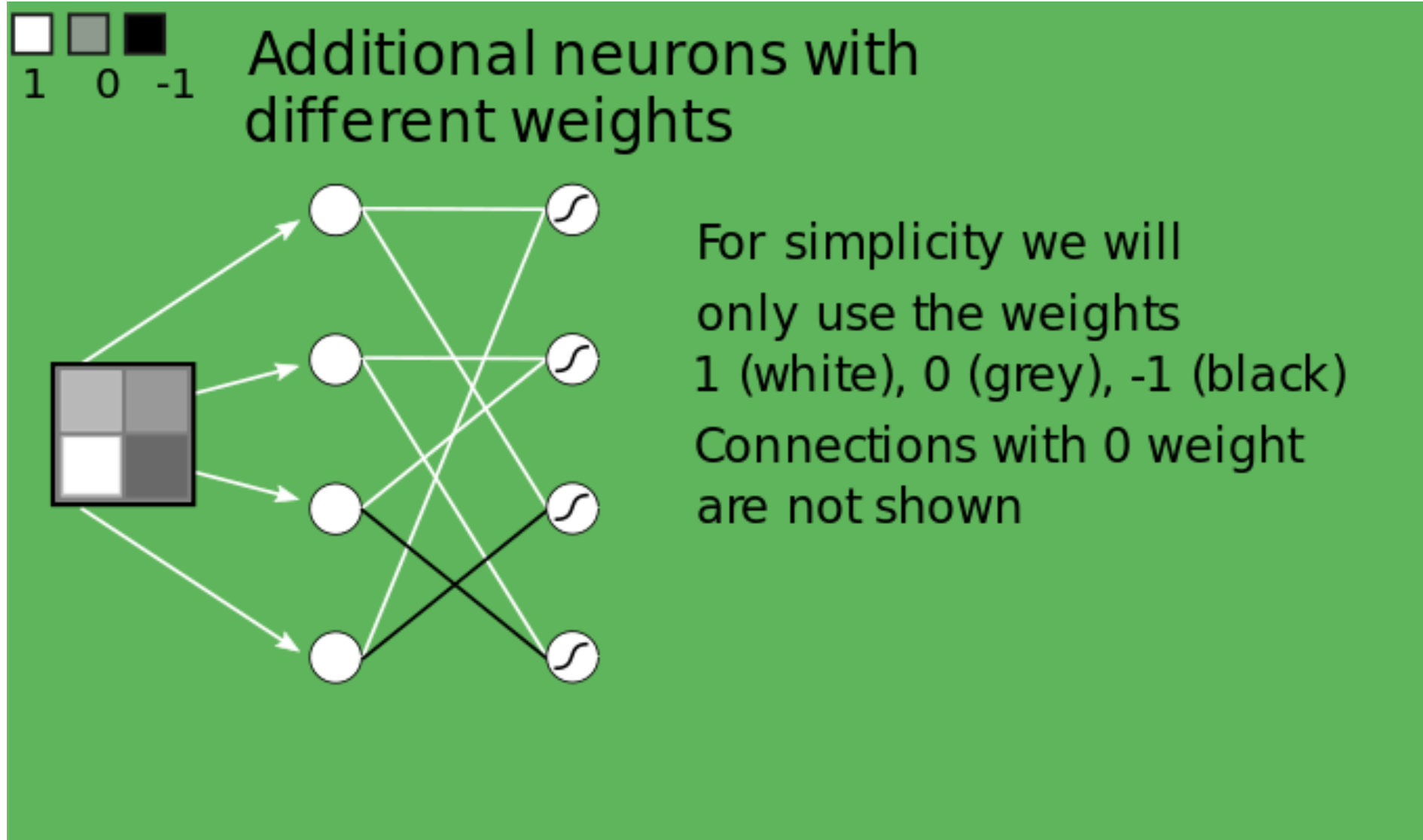


For the sake of simplicity, we combine both.

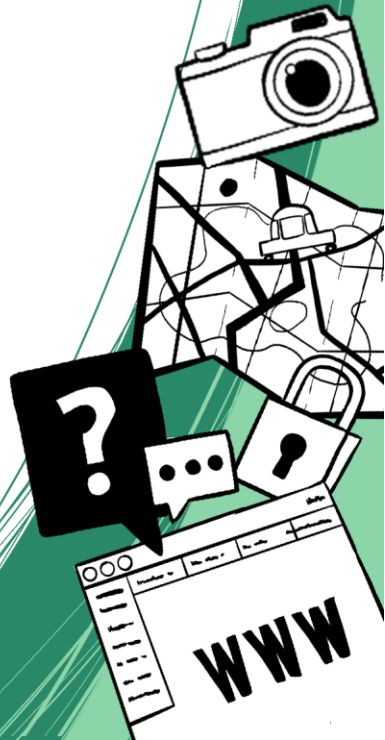
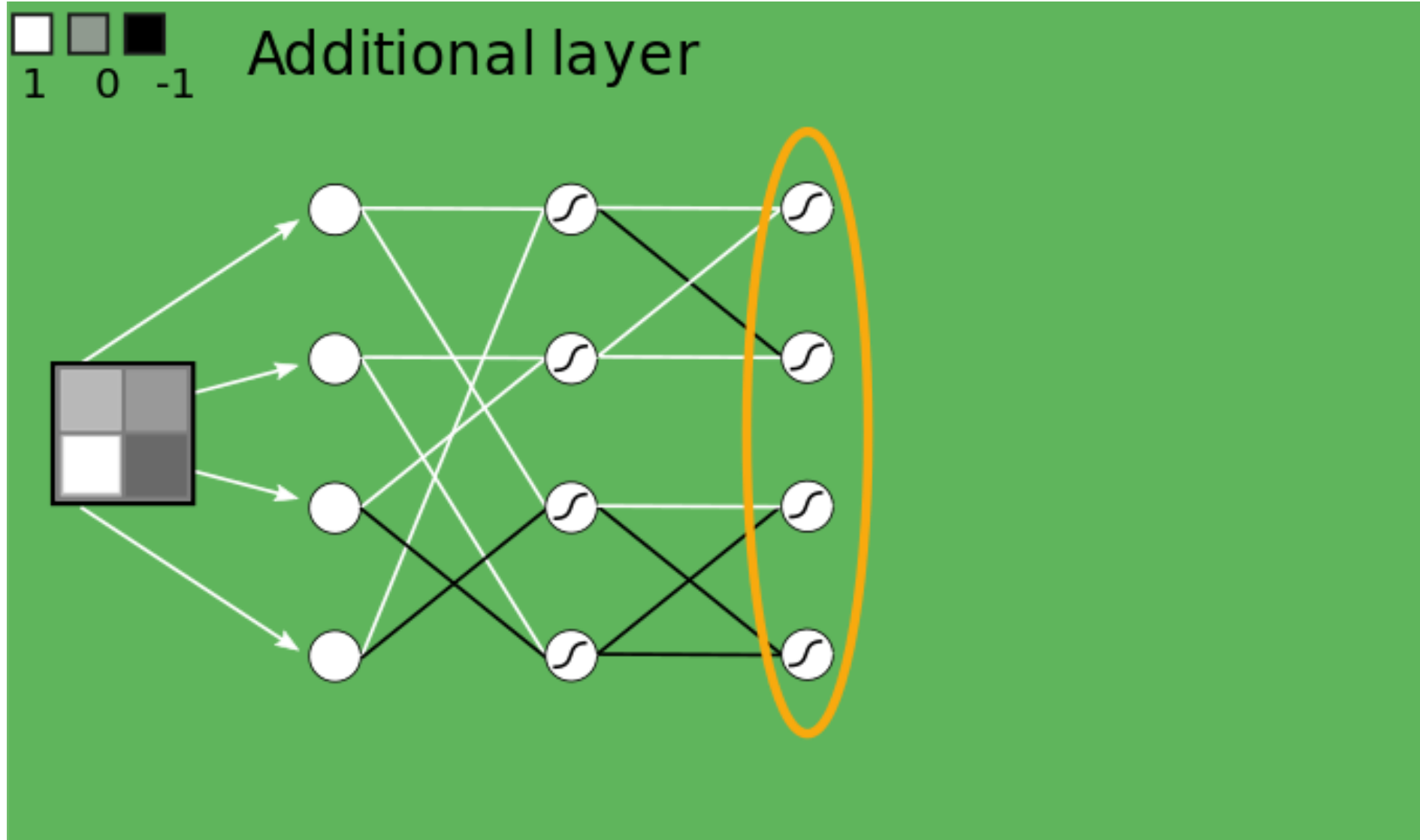
## Sum and activation function combined



We add more elements in this layer.

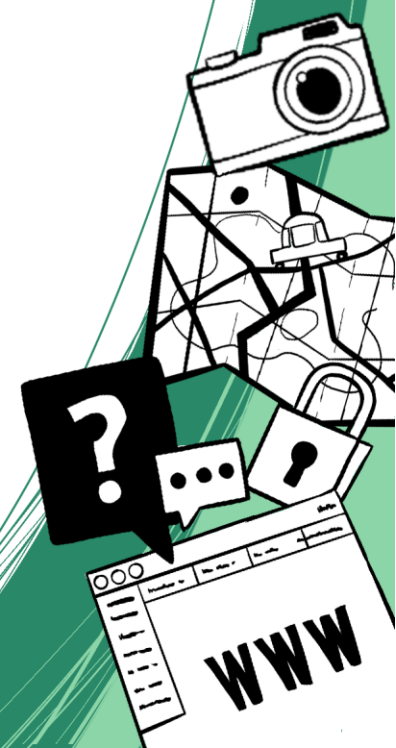
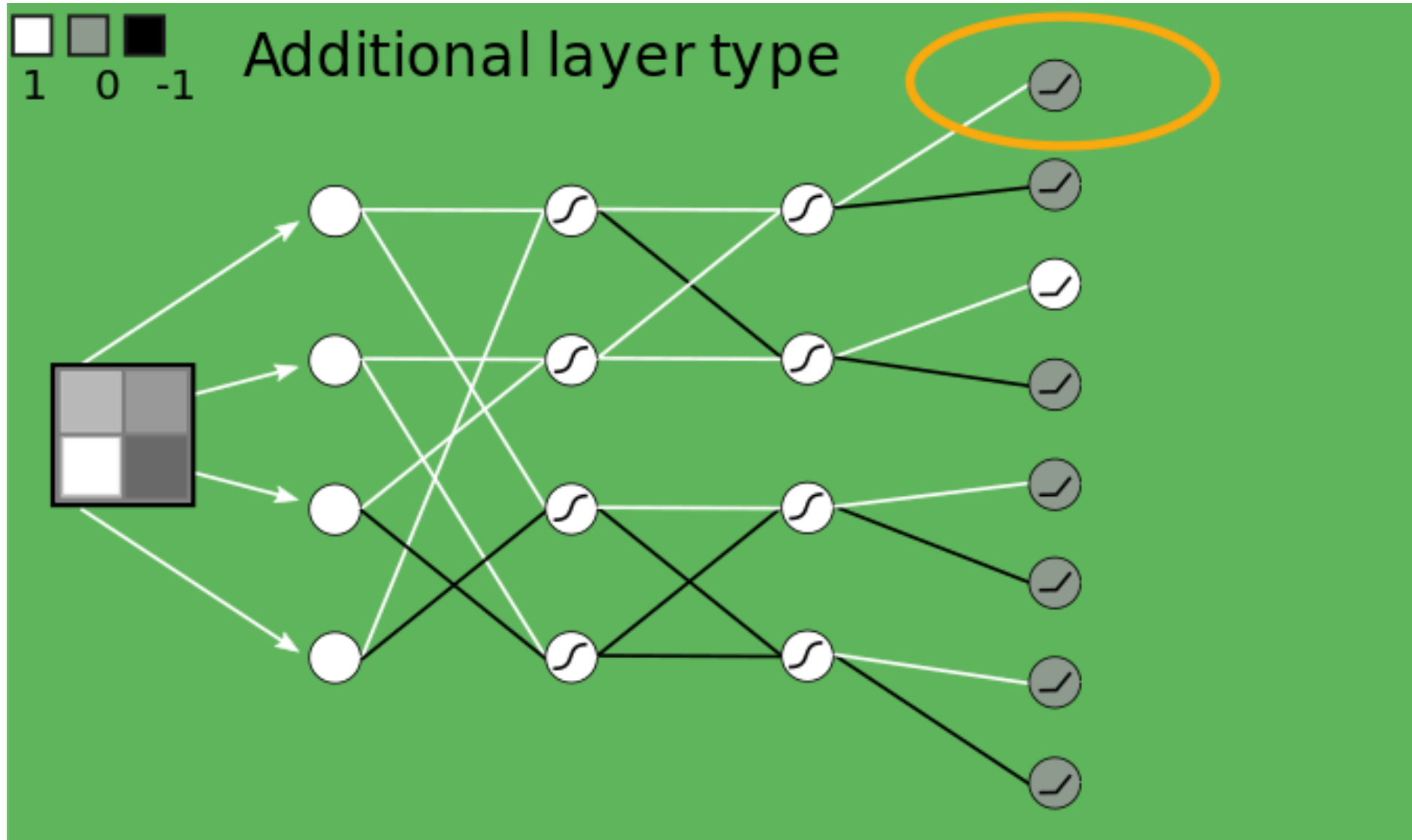


We now add another layer.



Finally, another layer with another type of neural cell.

In contrast to the previous activation function, this Function does not pass through negative values and sets them to 0.

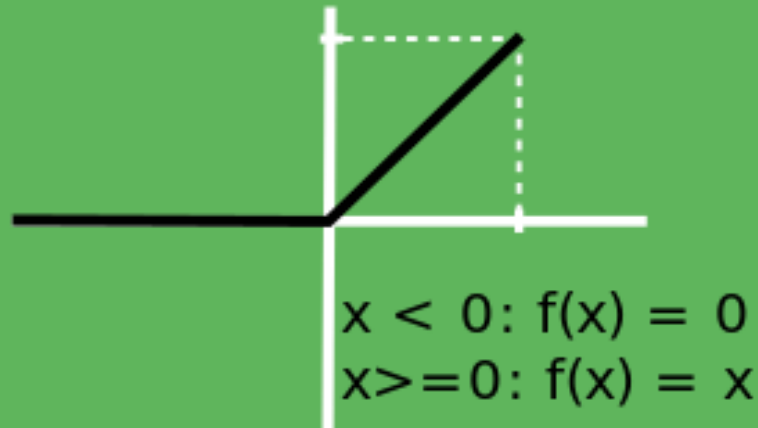




## Advanced content for the activation function

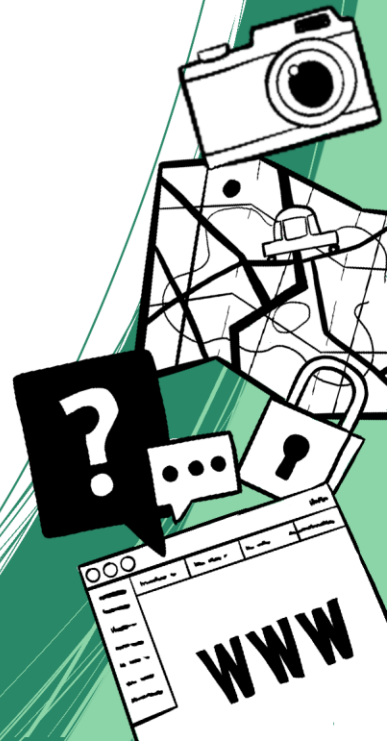
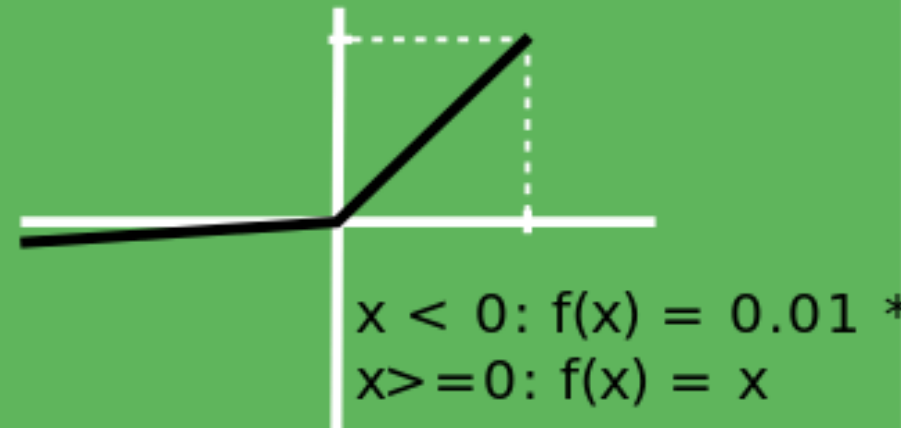
The ReLU function can be used here, or alternatively Leaky ReLU. Leaky ReLU has some advantages when training the network.

ReLU - Rectified Linear Unit

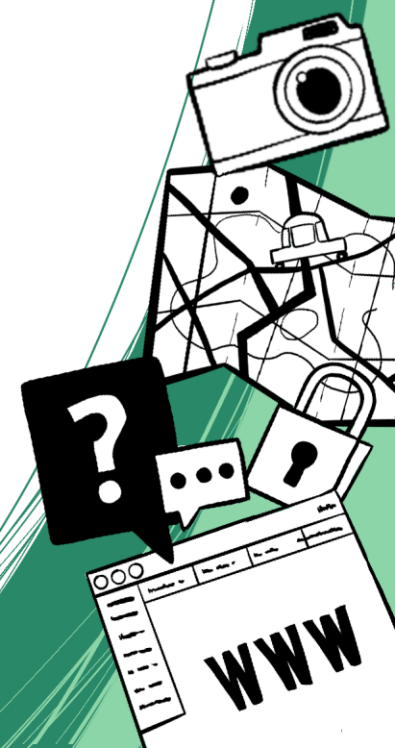
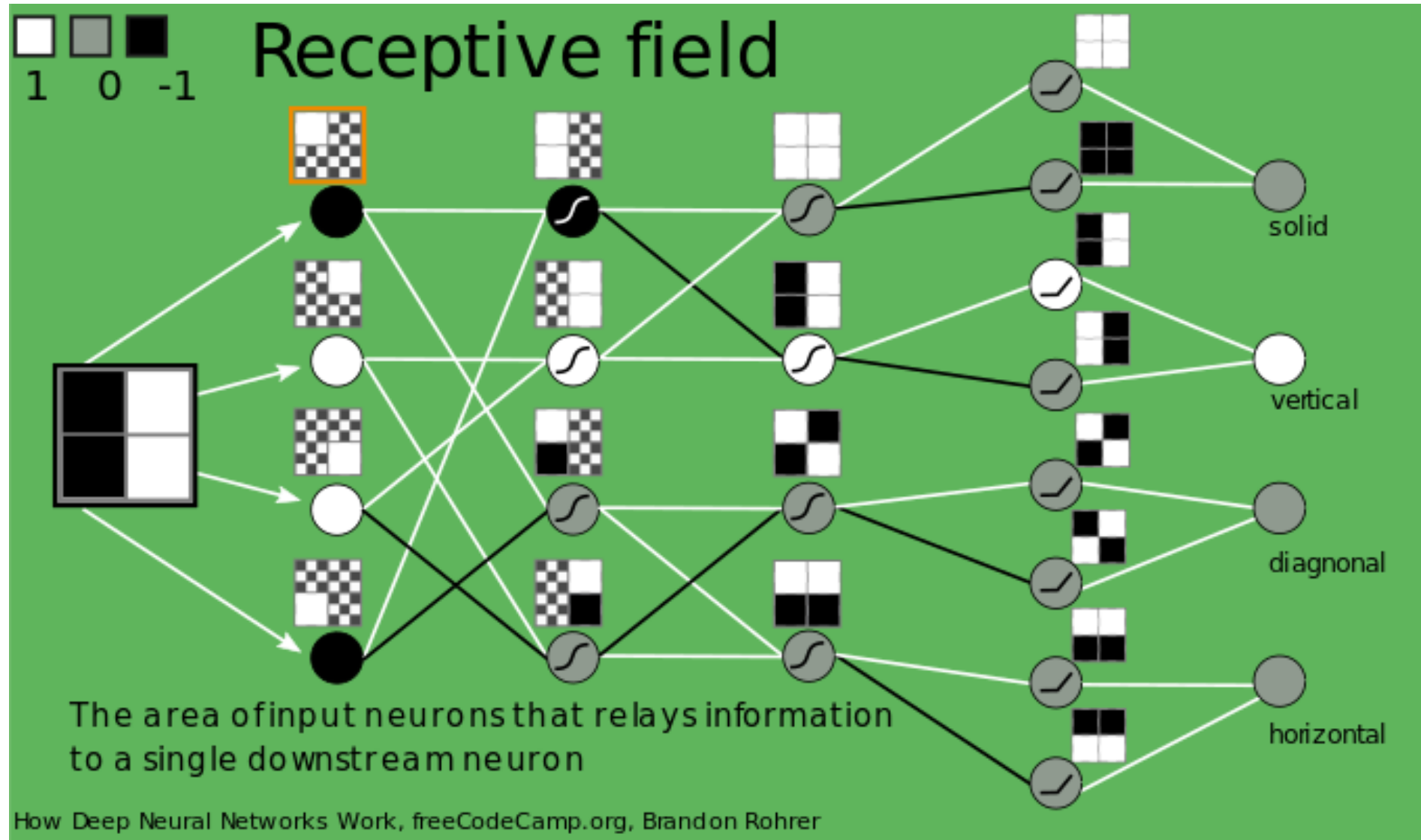


positive numbers are passed through  
negative numbers are set to 0

Leaky ReLU

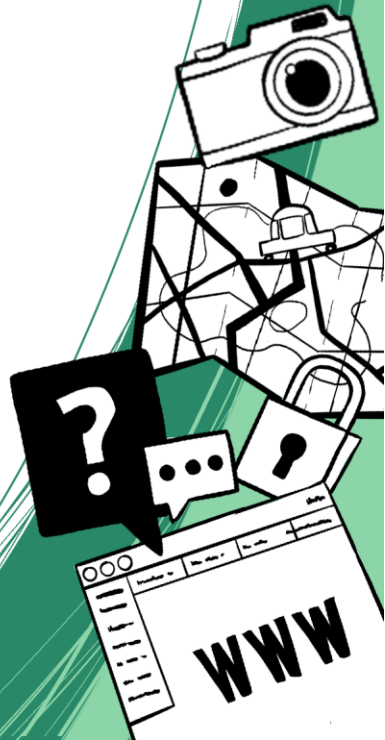


# Complete network showing the receptive field for each neuron



- You can try all of this in the simulator.
- The simulator is available in the automatic as well as in the manual version.

[simulator](#)





# Source

How Deep Neural Networks Work - Full Course for Beginners

[https://www.youtube.com/watch?](https://www.youtube.com/watch?v=dPWYUELwIdM&t=1264s&ab_channel=freeCodeCamp.org)

[v=dPWYUELwIdM&t=1264s&ab\\_channel=freeCodeCamp.org](https://www.youtube.com/watch?v=dPWYUELwIdM&t=1264s&ab_channel=freeCodeCamp.org)

