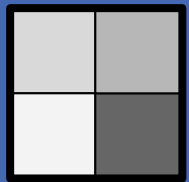


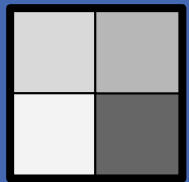
How neural networks work

Brandon Rohrer

A four pixel camera



Categorize images



solid



vertical



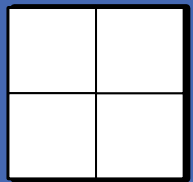
diagonal



horizontal



Categorize images



solid



vertical



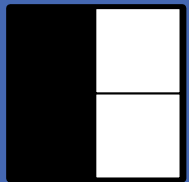
diagonal



horizontal



Categorize images



solid



vertical



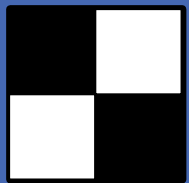
diagonal



horizontal



Categorize images



solid



vertical



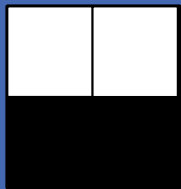
diagonal



horizontal



Simple rules can't do it



solid



vertical



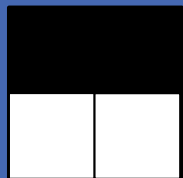
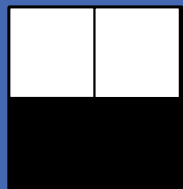
diagonal



horizontal



Simple rules can't do it



solid



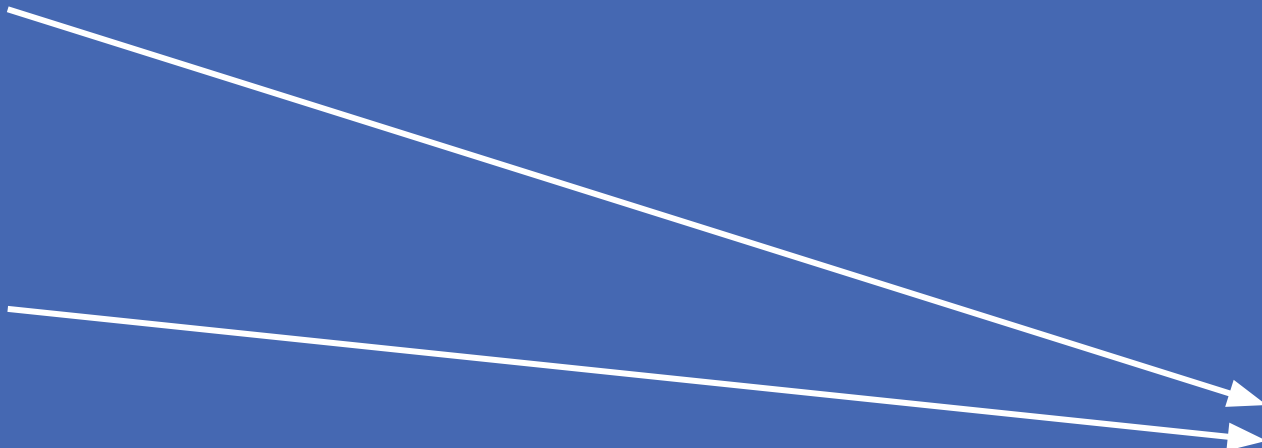
vertical



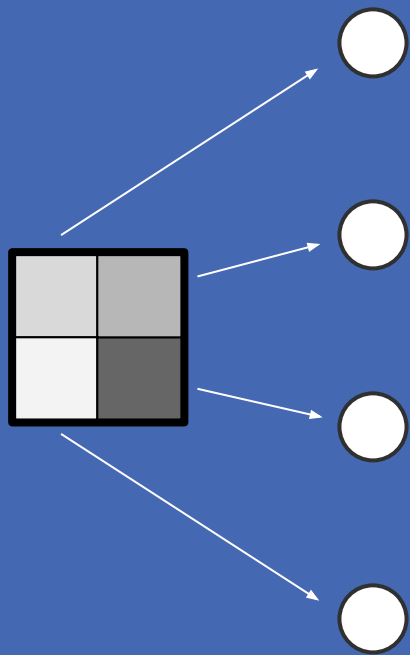
diagonal



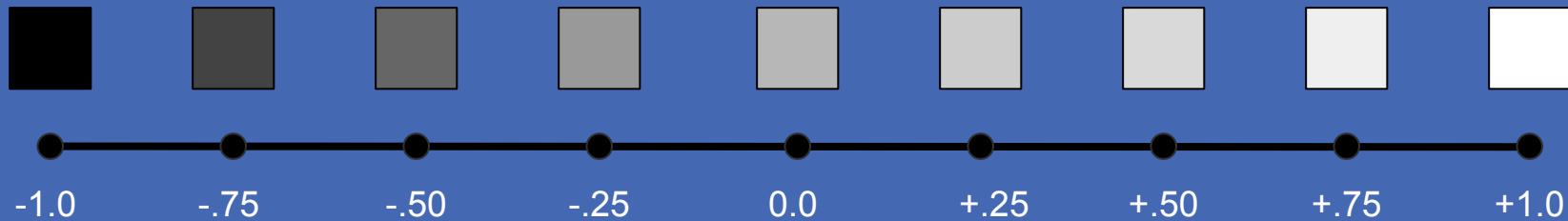
horizontal



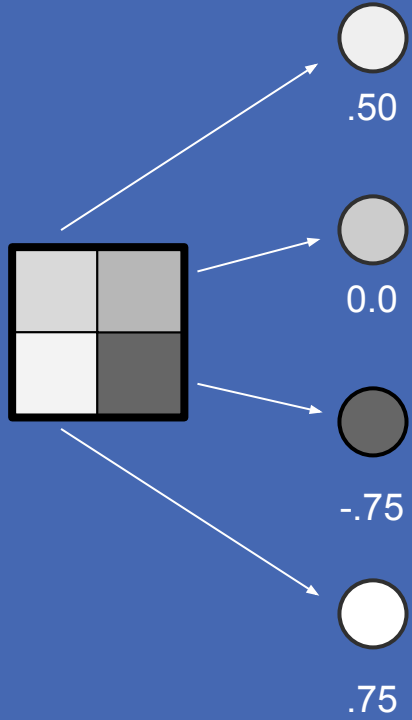
Input neurons



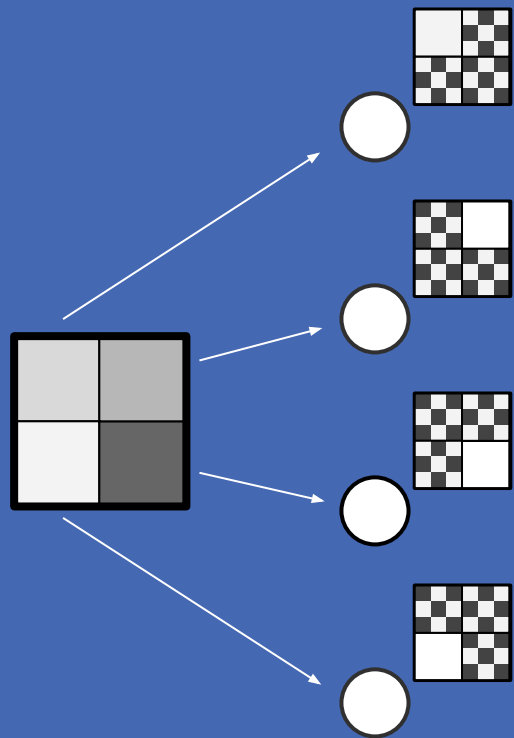
Pixel brightness



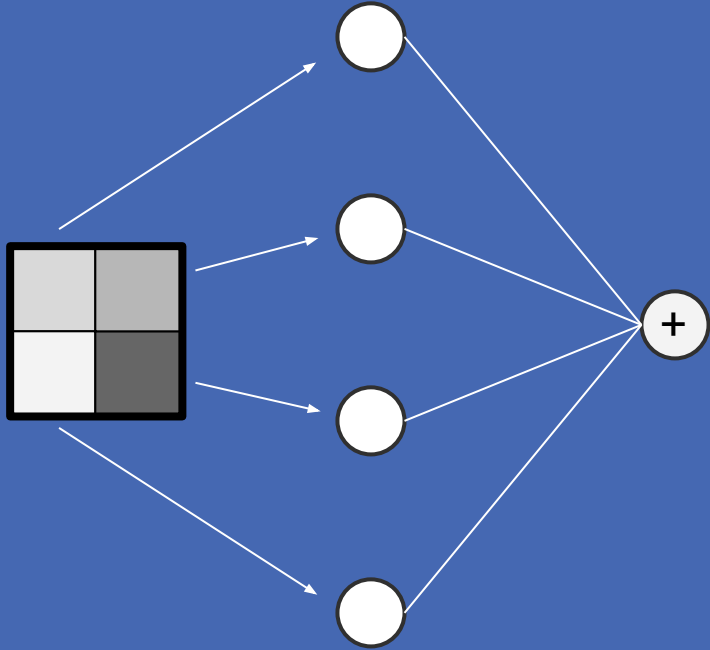
Input vector



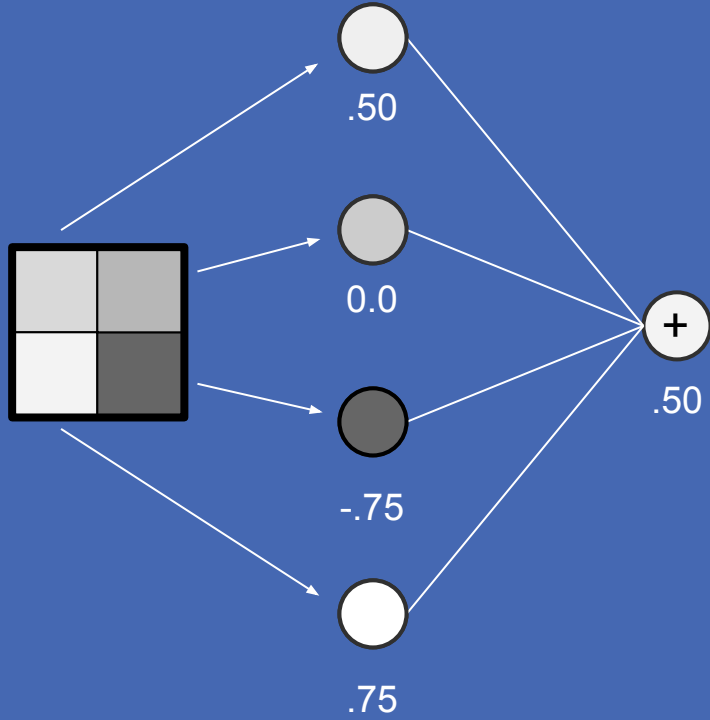
Receptive fields



A neuron

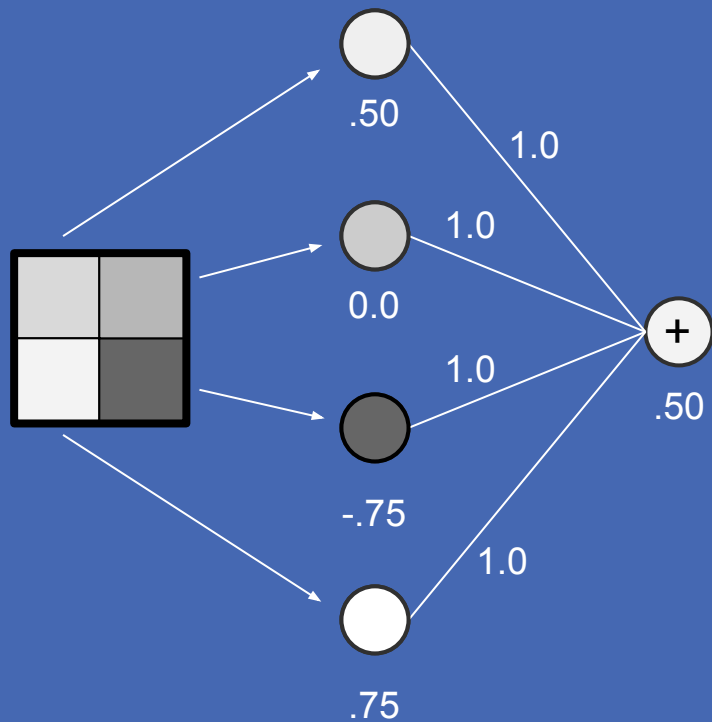


Sum all the inputs



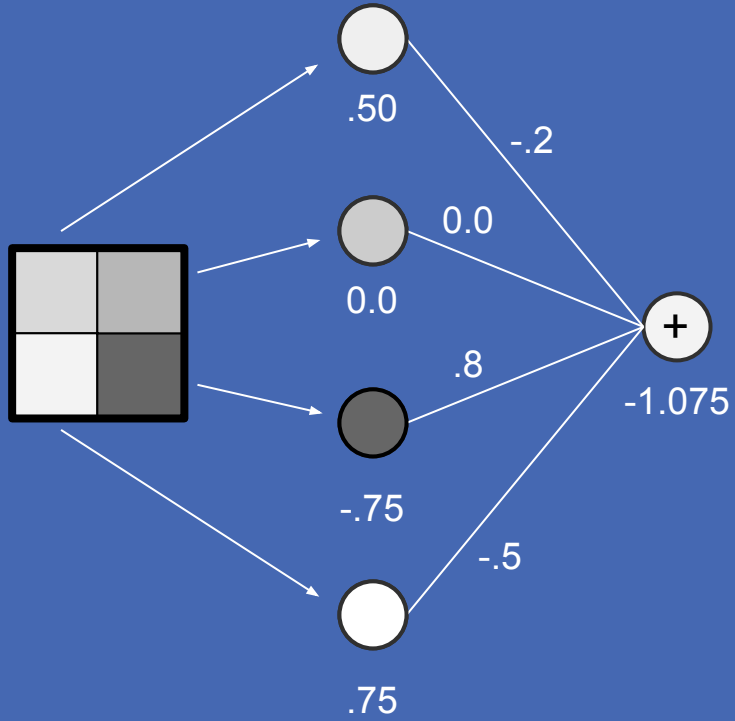
$$\begin{array}{r} .50 \\ 0.00 \\ -.75 \\ + \quad .75 \\ \hline .50 \end{array}$$

Weights



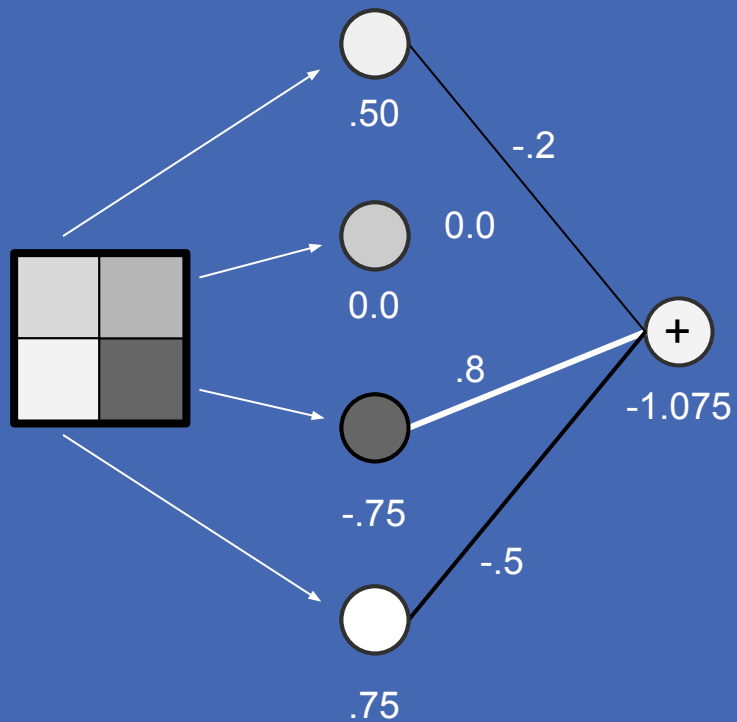
$$\begin{array}{r} .50 \times 1.0 \\ 0.00 \times 1.0 \\ -.75 \times 1.0 \\ + \quad .75 \times 1.0 \\ \hline .50 \end{array}$$

Weights



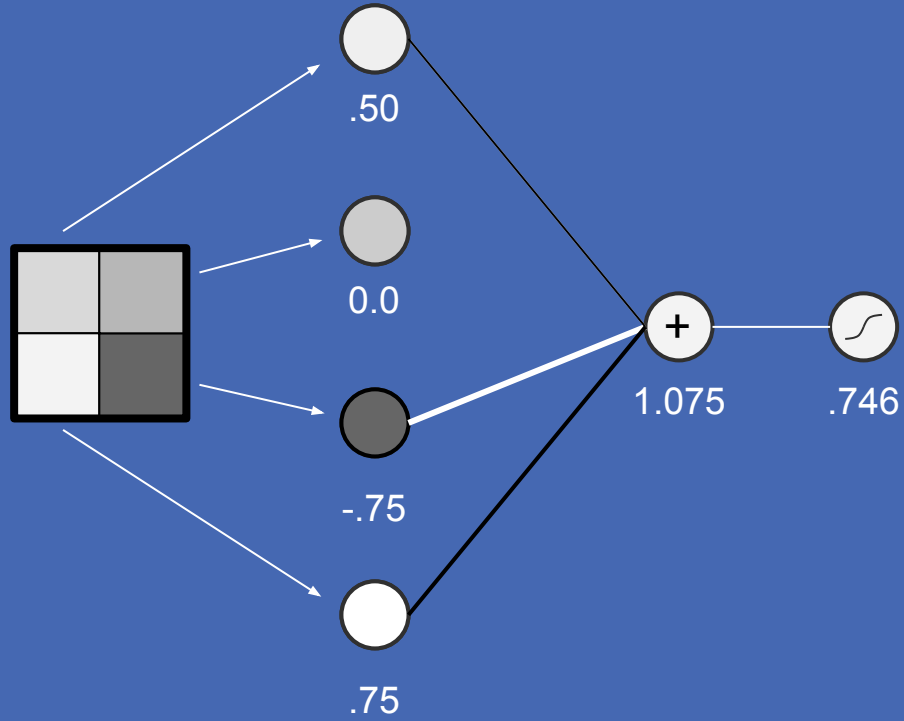
$$\begin{array}{r} .50 \times -.2 \\ 0.00 \times 0.0 \\ -.75 \times .8 \\ + .75 \times -.5 \\ \hline -1.075 \end{array}$$

Weights

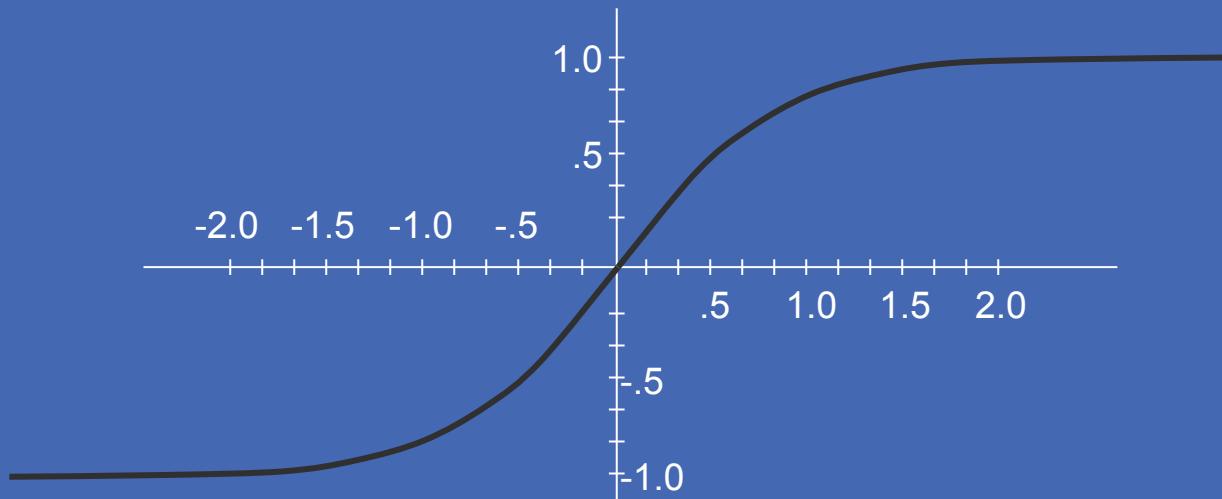


$$\begin{array}{rcl} .50 & \times & -.2 \\ 0.00 & \times & 0.0 \\ -.75 & \times & .8 \\ + & .75 & \times & -.5 \\ \hline & & -1.075 \end{array}$$

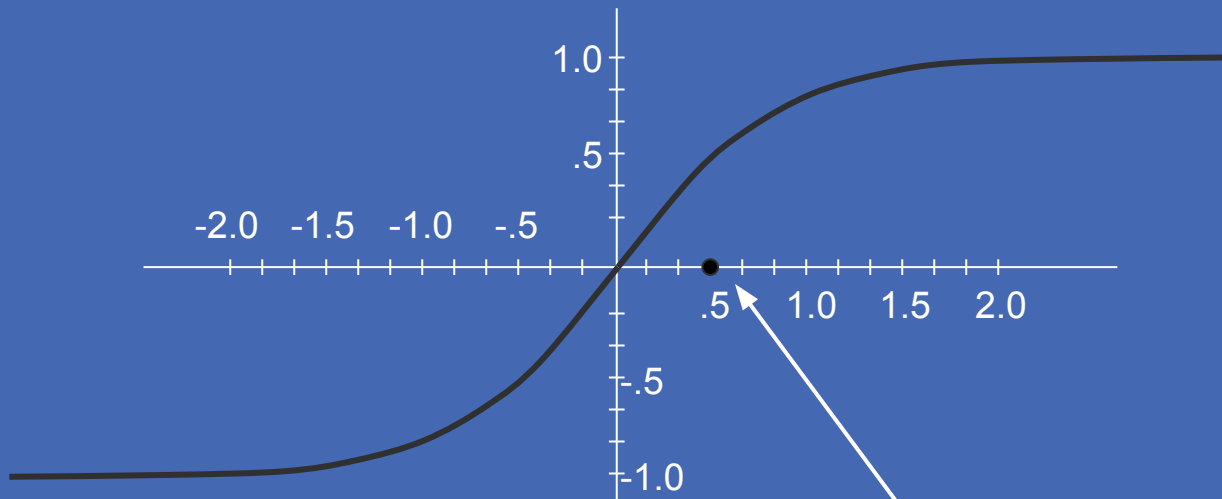
Squash the result



Sigmoid squashing function

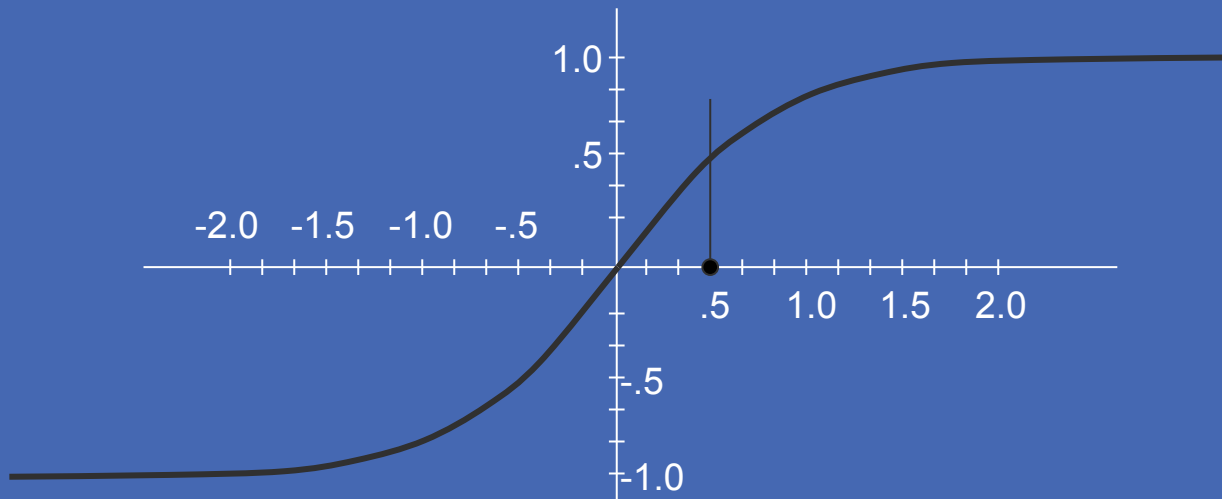


Sigmoid squashing function

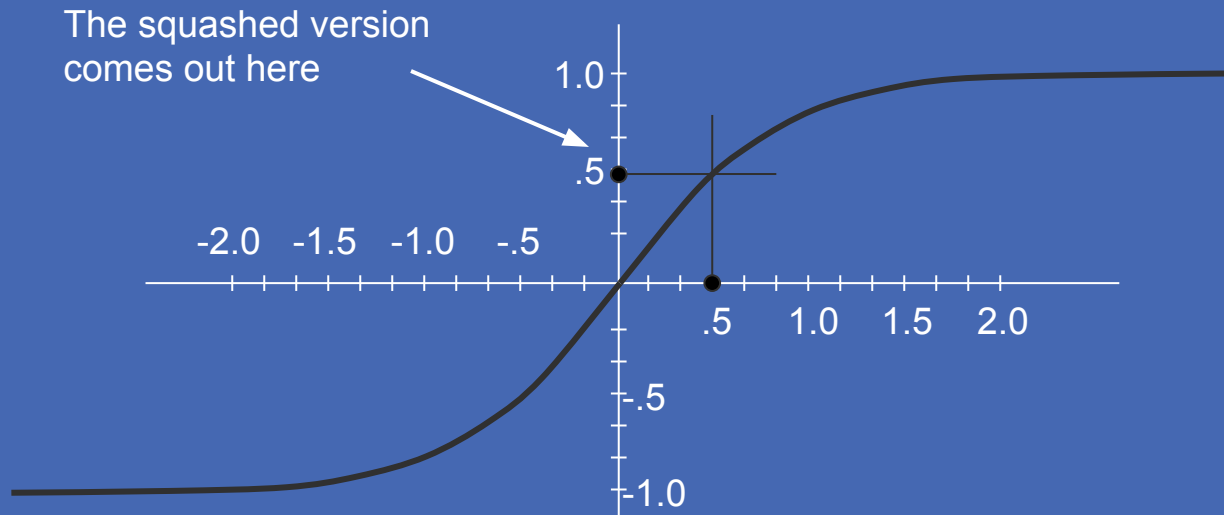


Your number goes in here

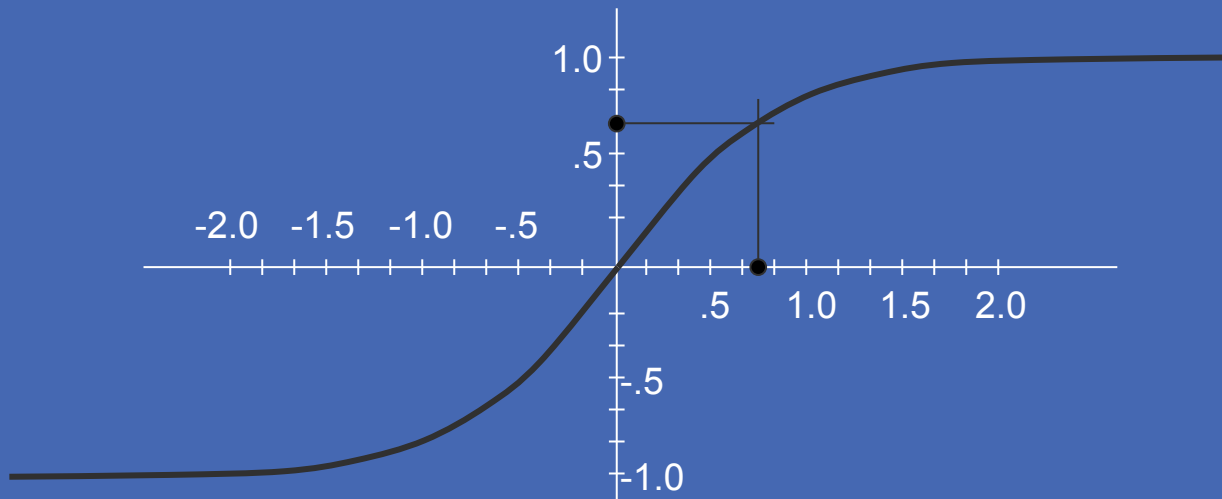
Sigmoid squashing function



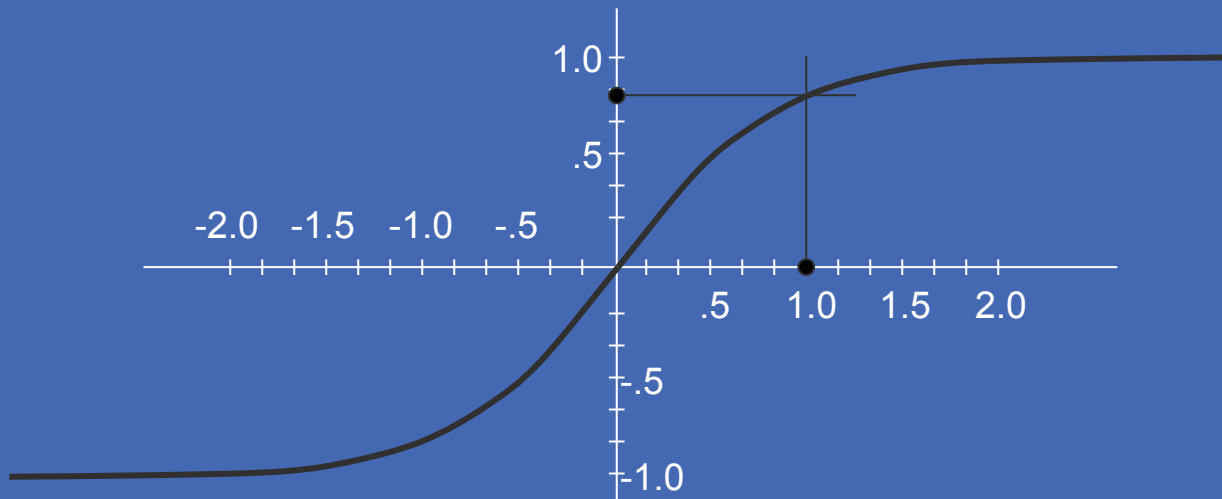
Sigmoid squashing function



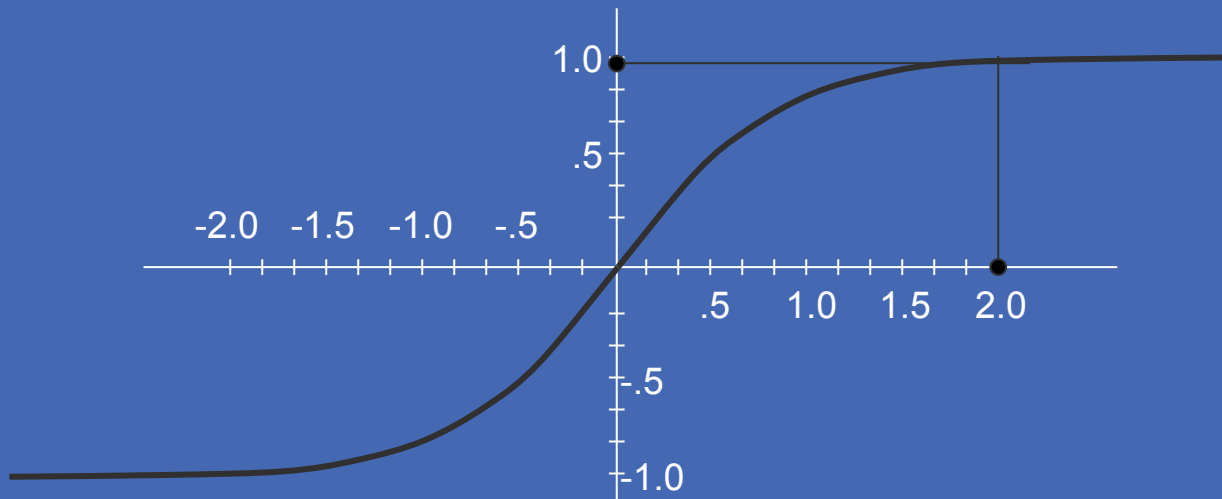
Sigmoid squashing function



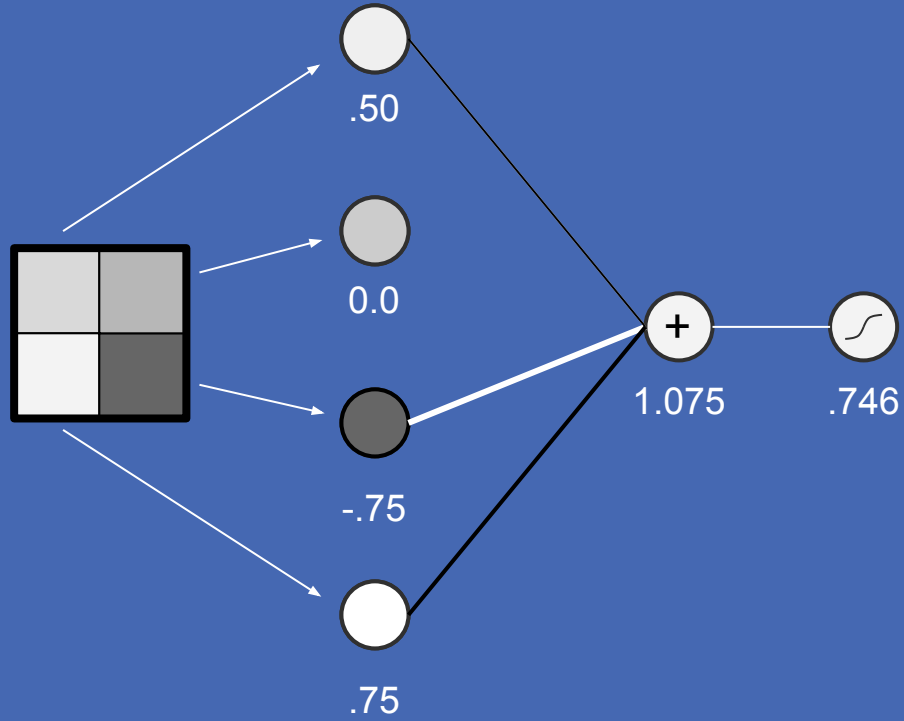
Sigmoid squashing function



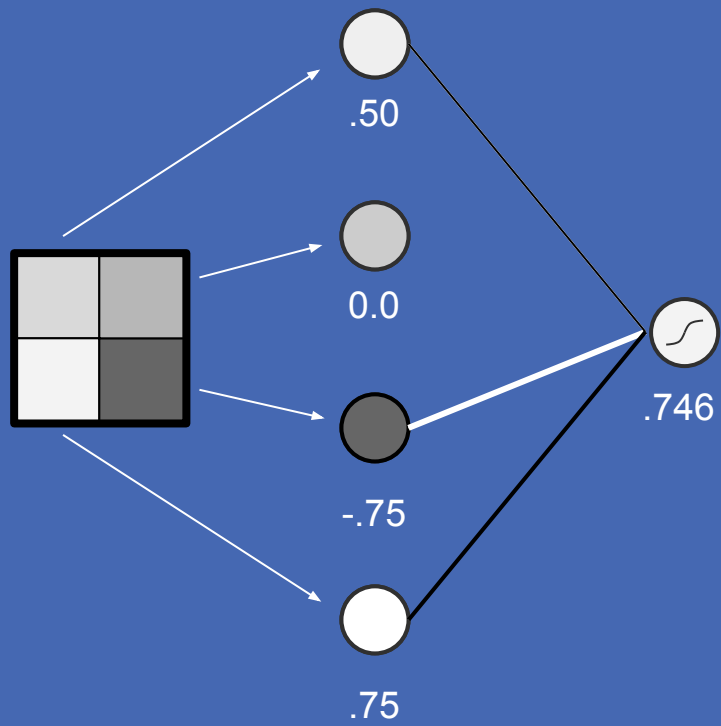
No matter what you start with, the answer stays between -1 and 1.



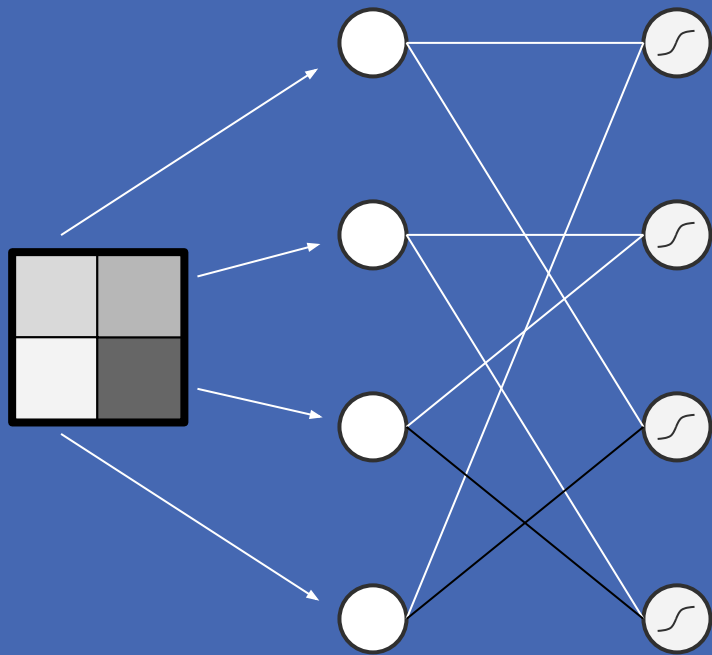
Squash the result



Weighted sum-and-squash neuron

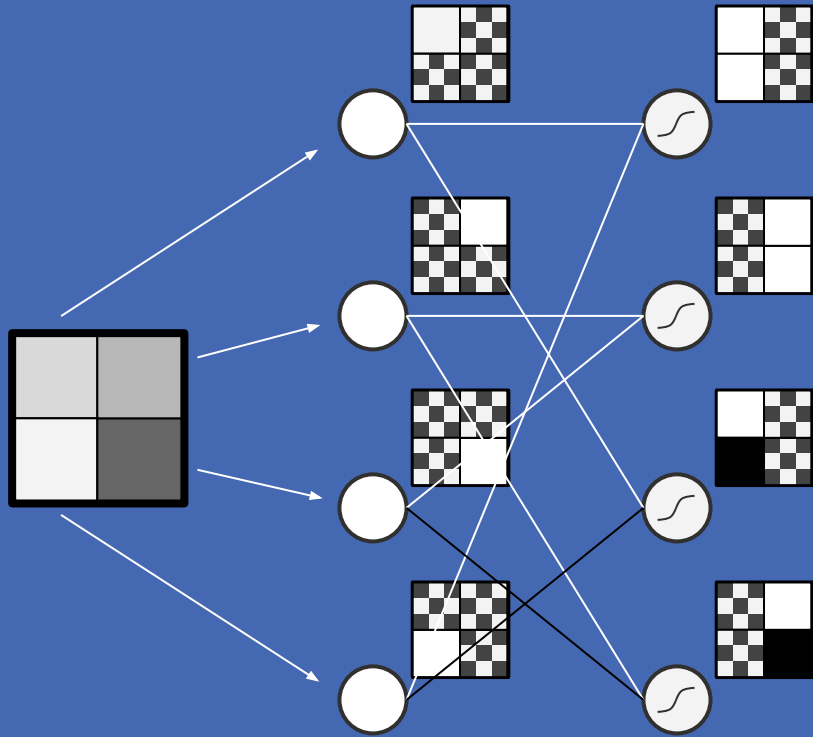


Make lots of neurons, identical except for weights

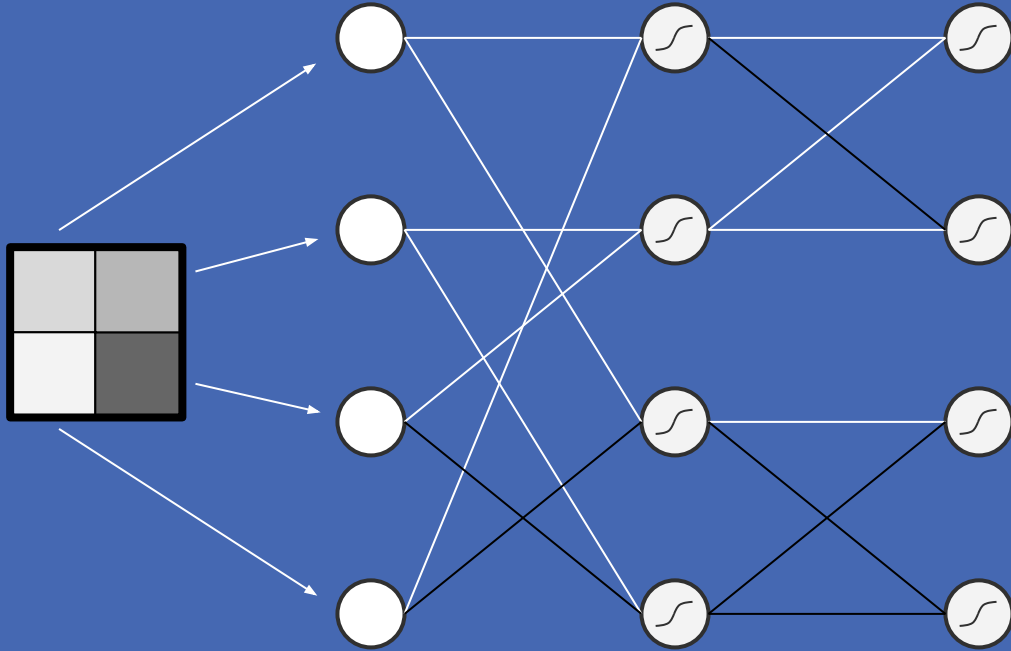


To keep our picture
clear, weights will either
be
1.0 (white)
-1.0 (black) or
0.0 (missing)

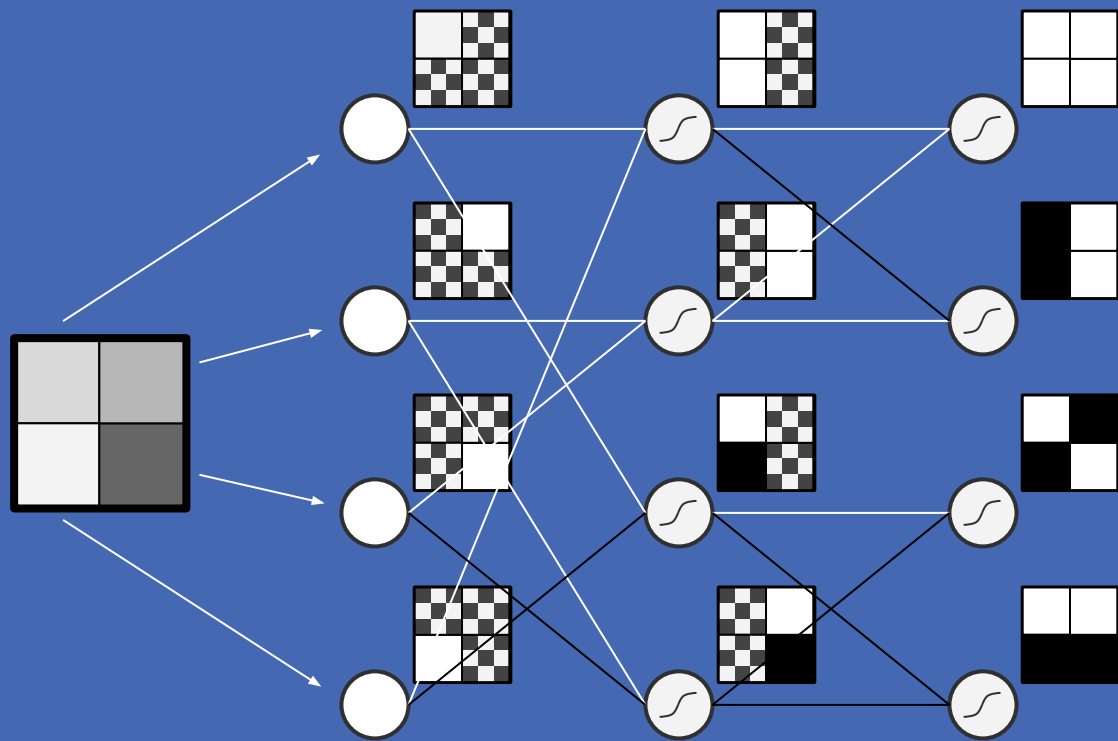
Receptive fields get more complex



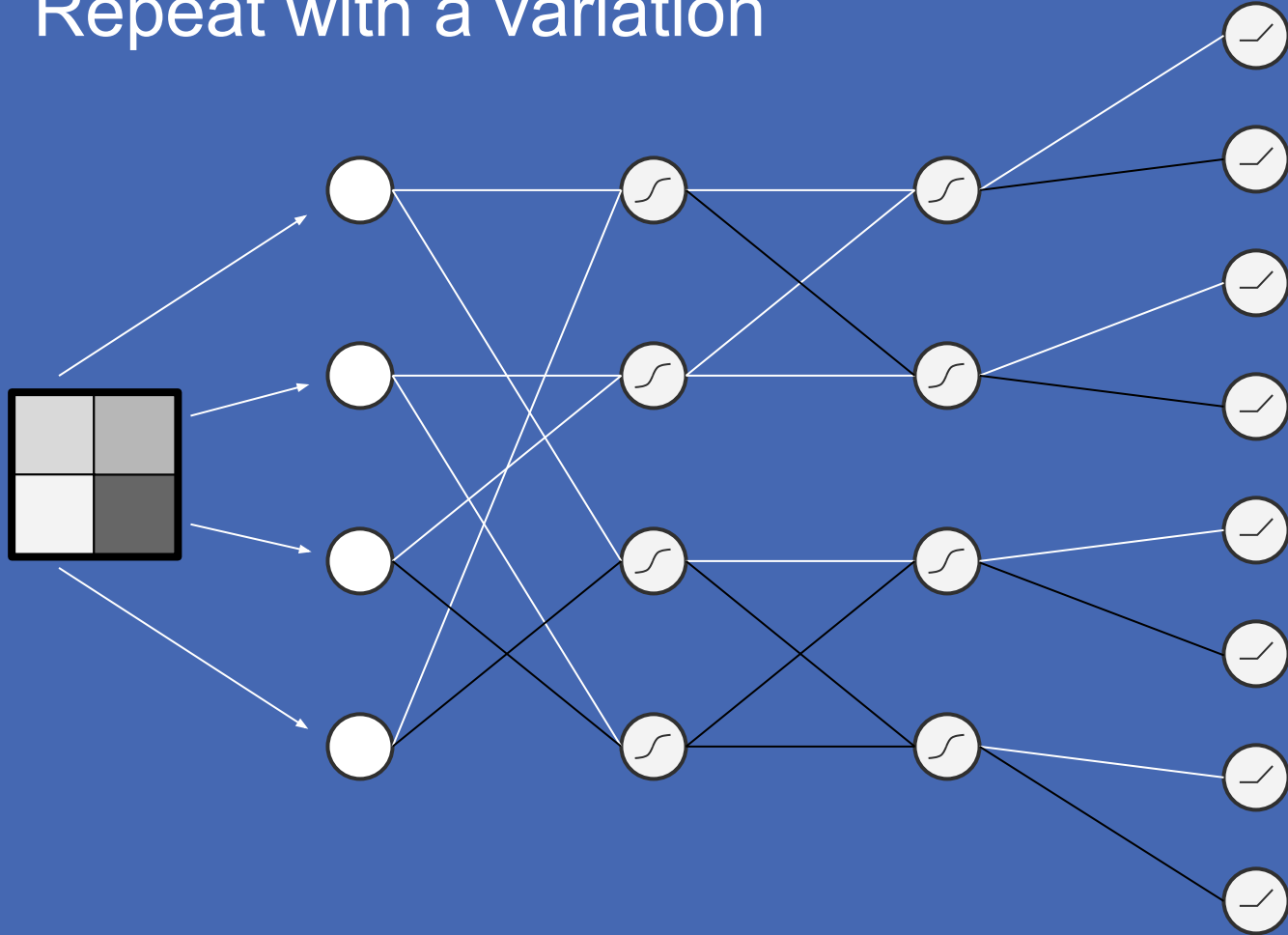
Repeat for additional layers



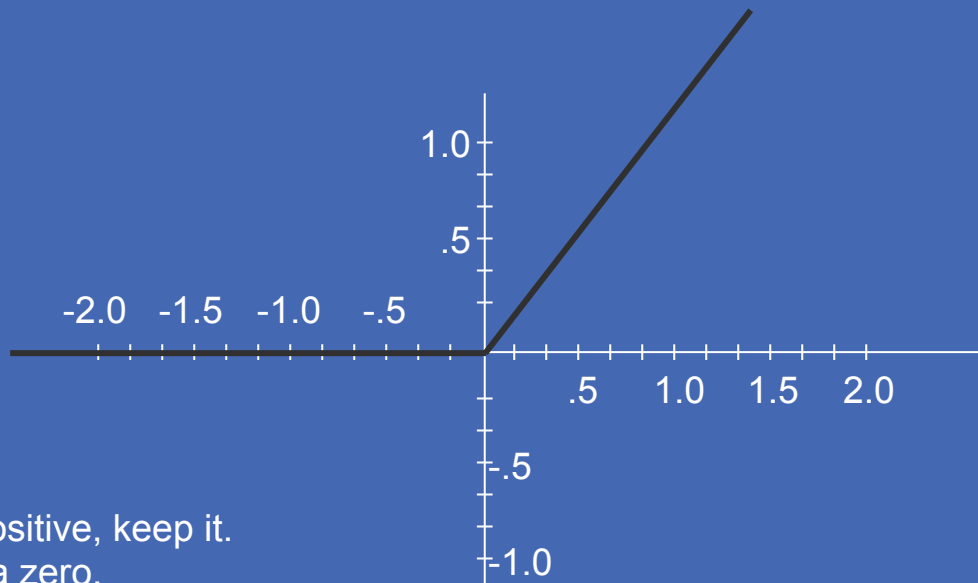
Receptive fields get still more complex



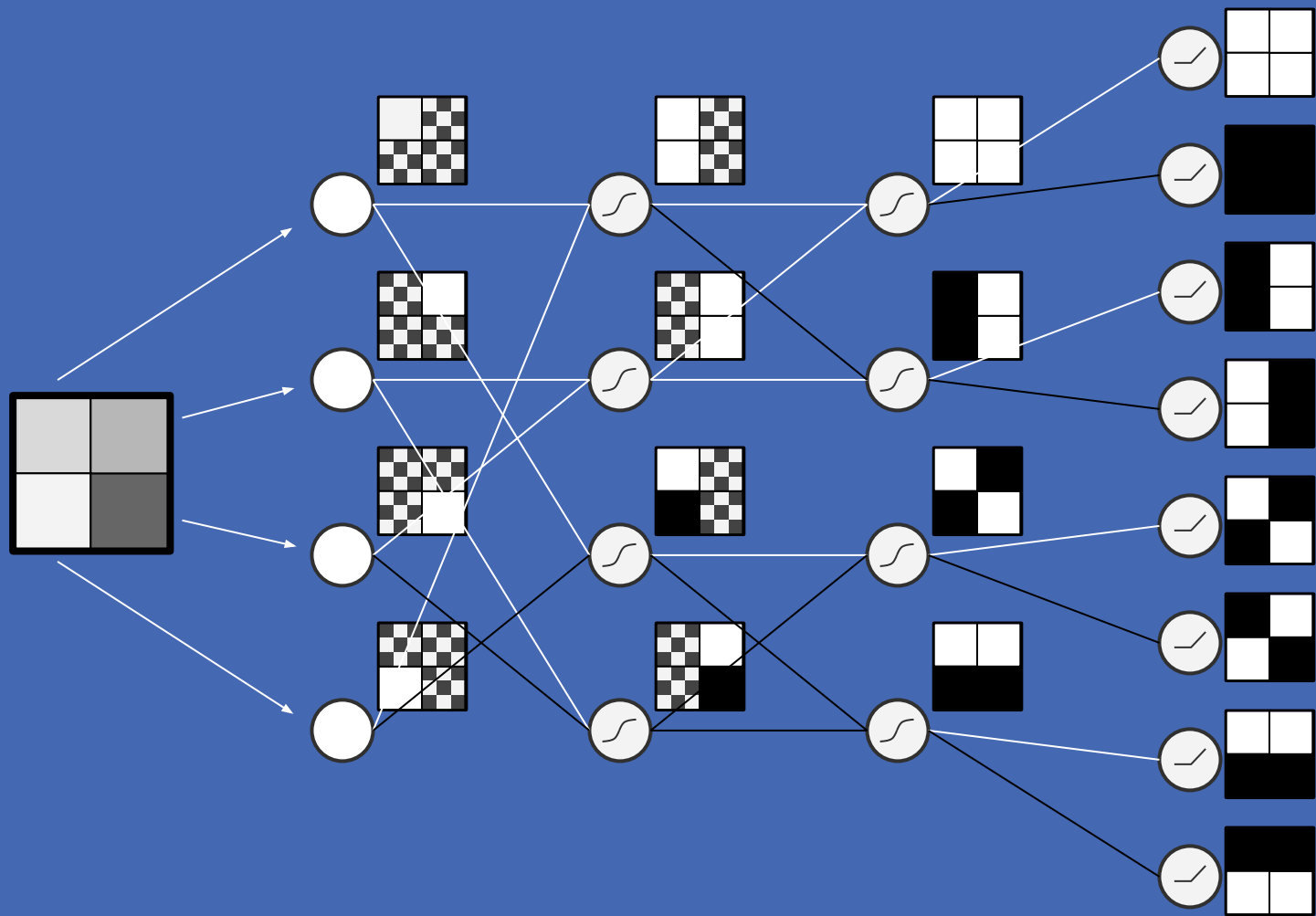
Repeat with a variation



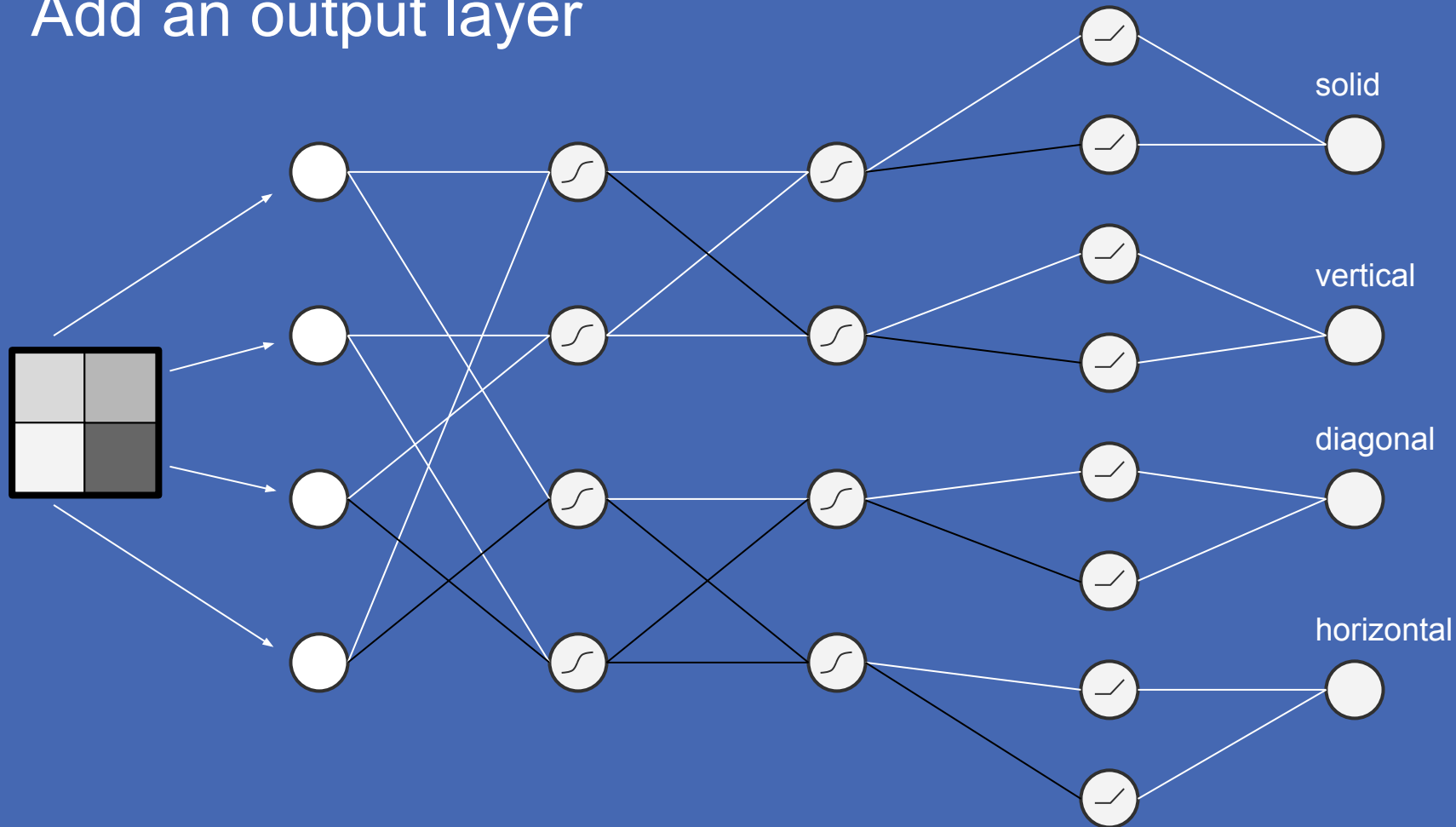
Rectified linear units (ReLUs)

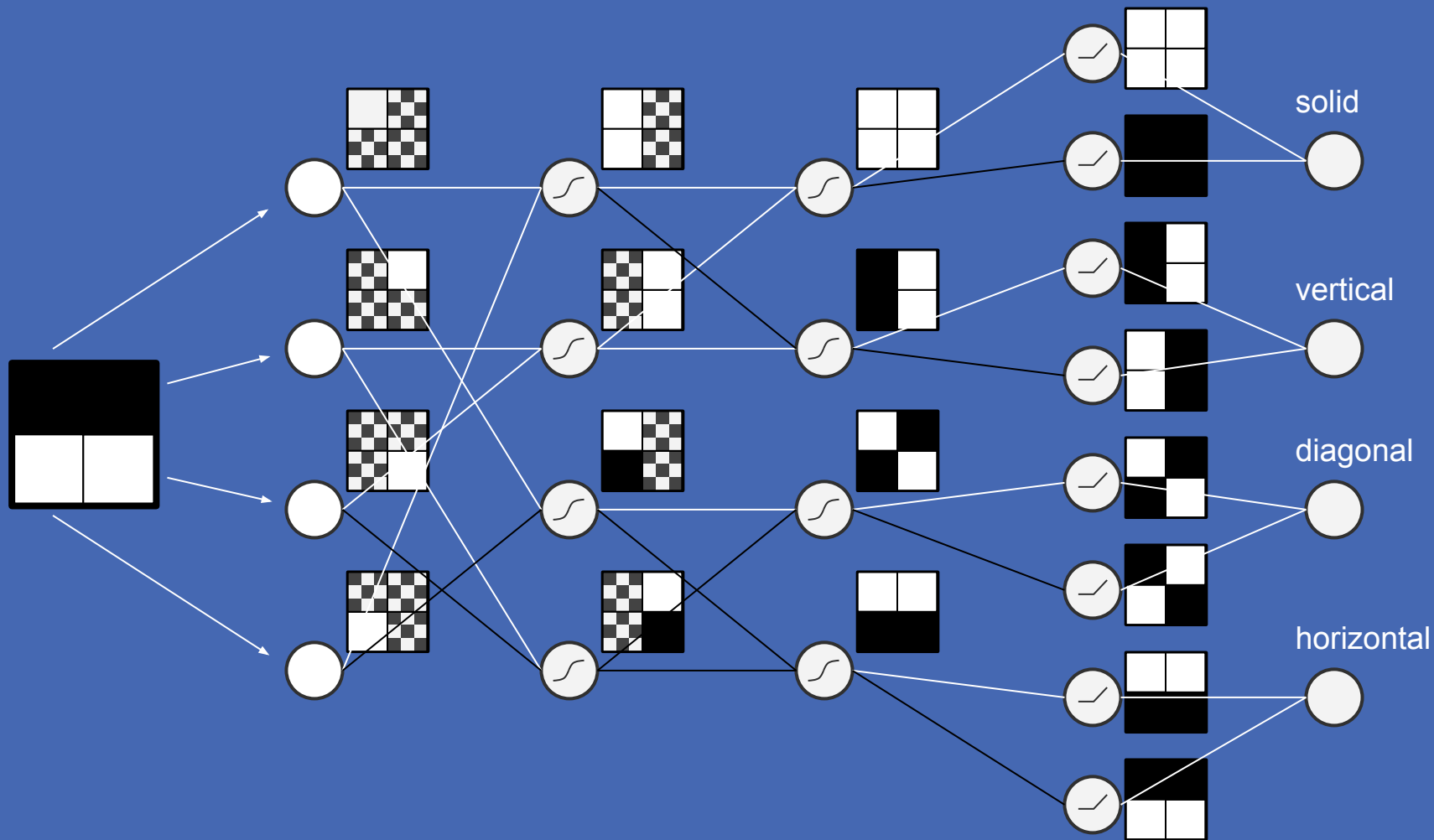


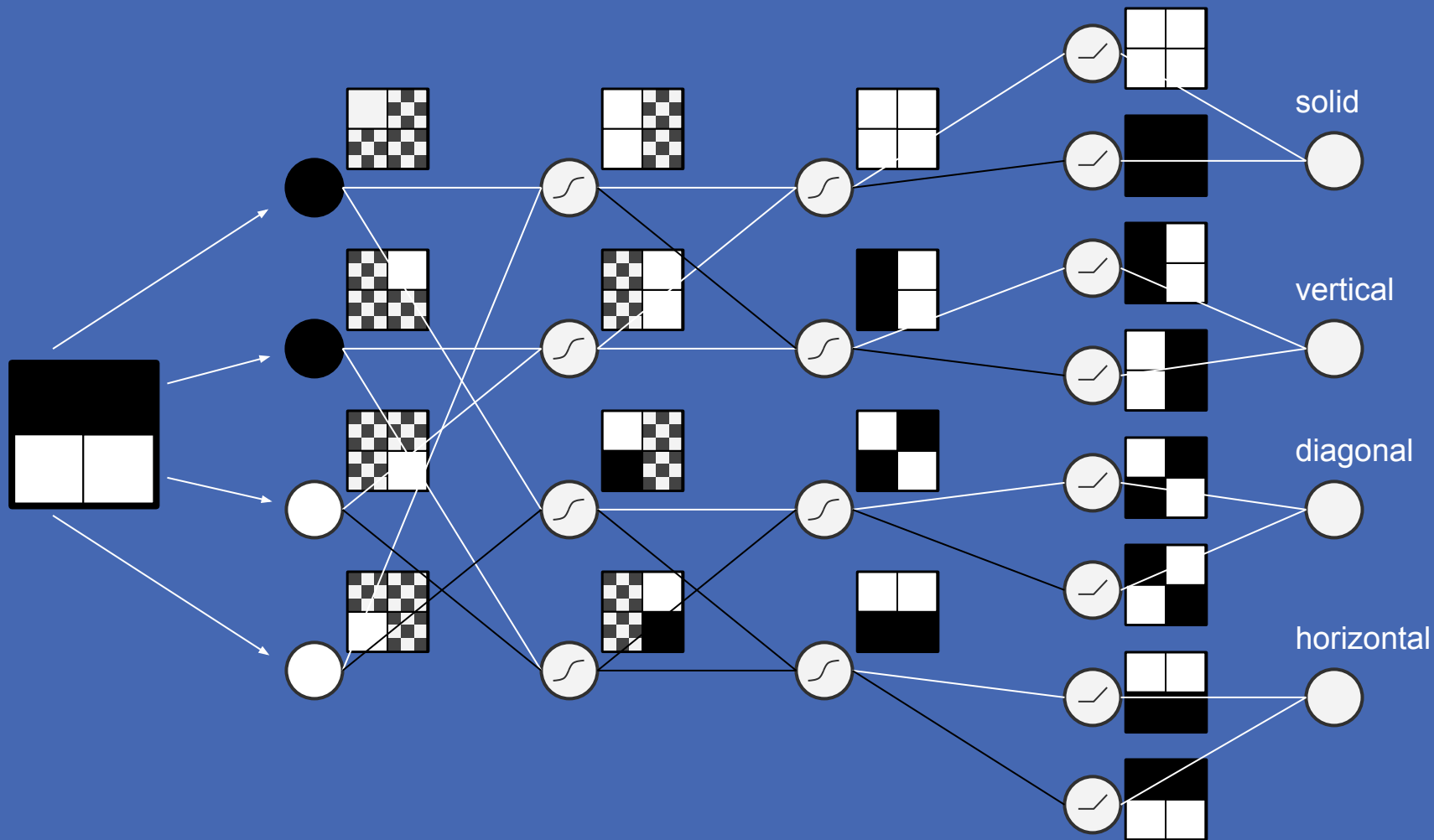
If your number is positive, keep it.
Otherwise you get a zero.

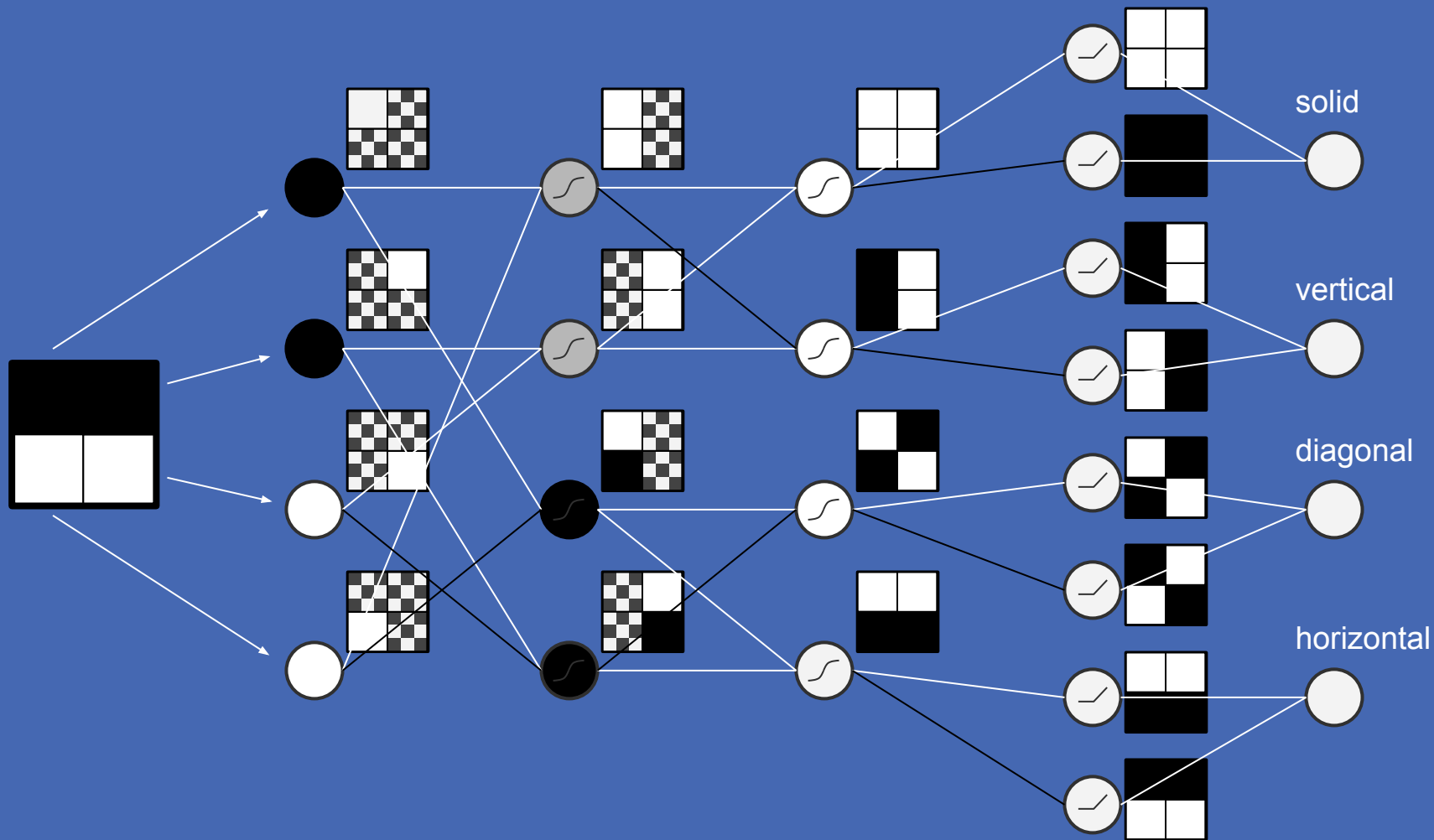


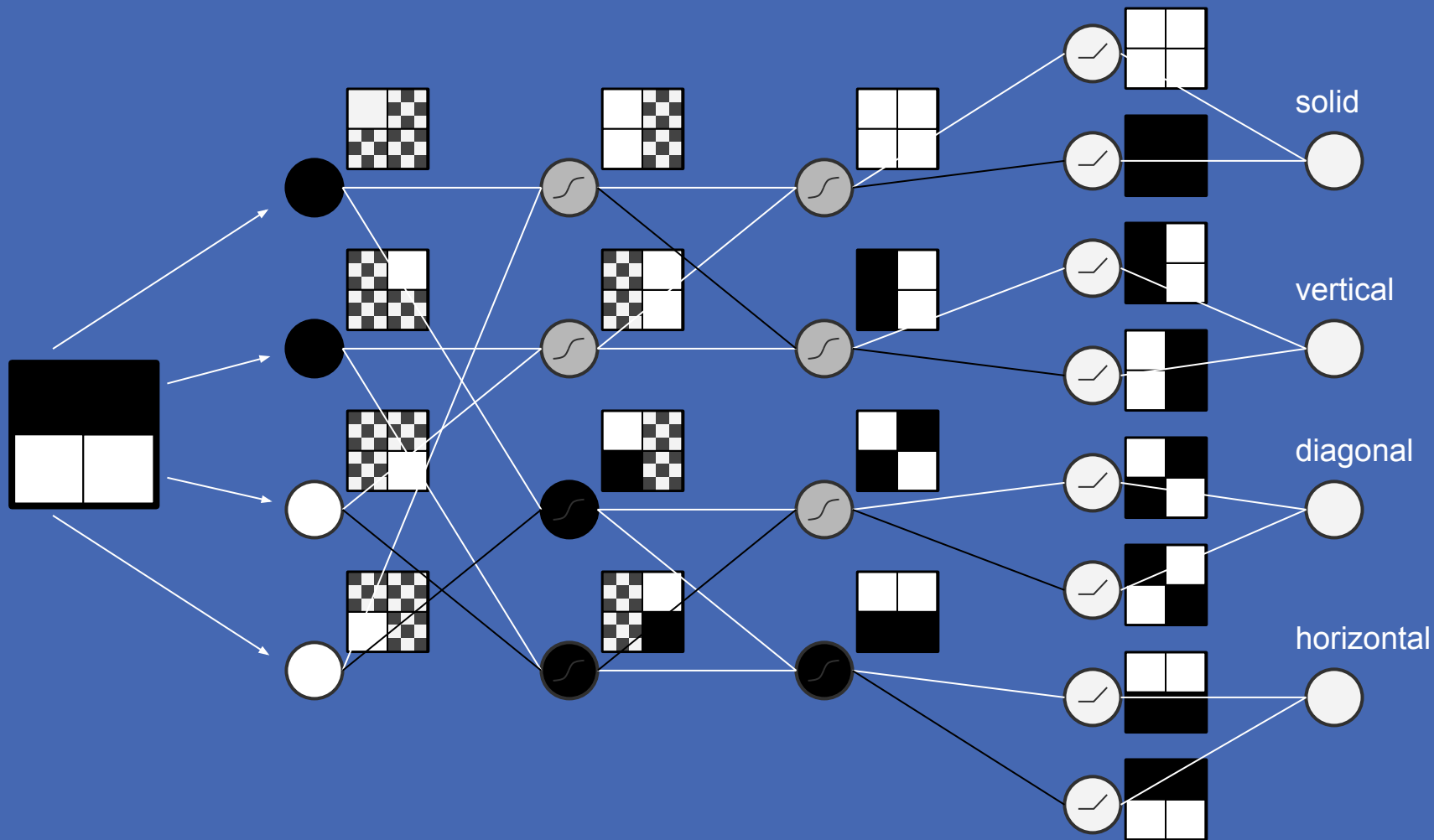
Add an output layer

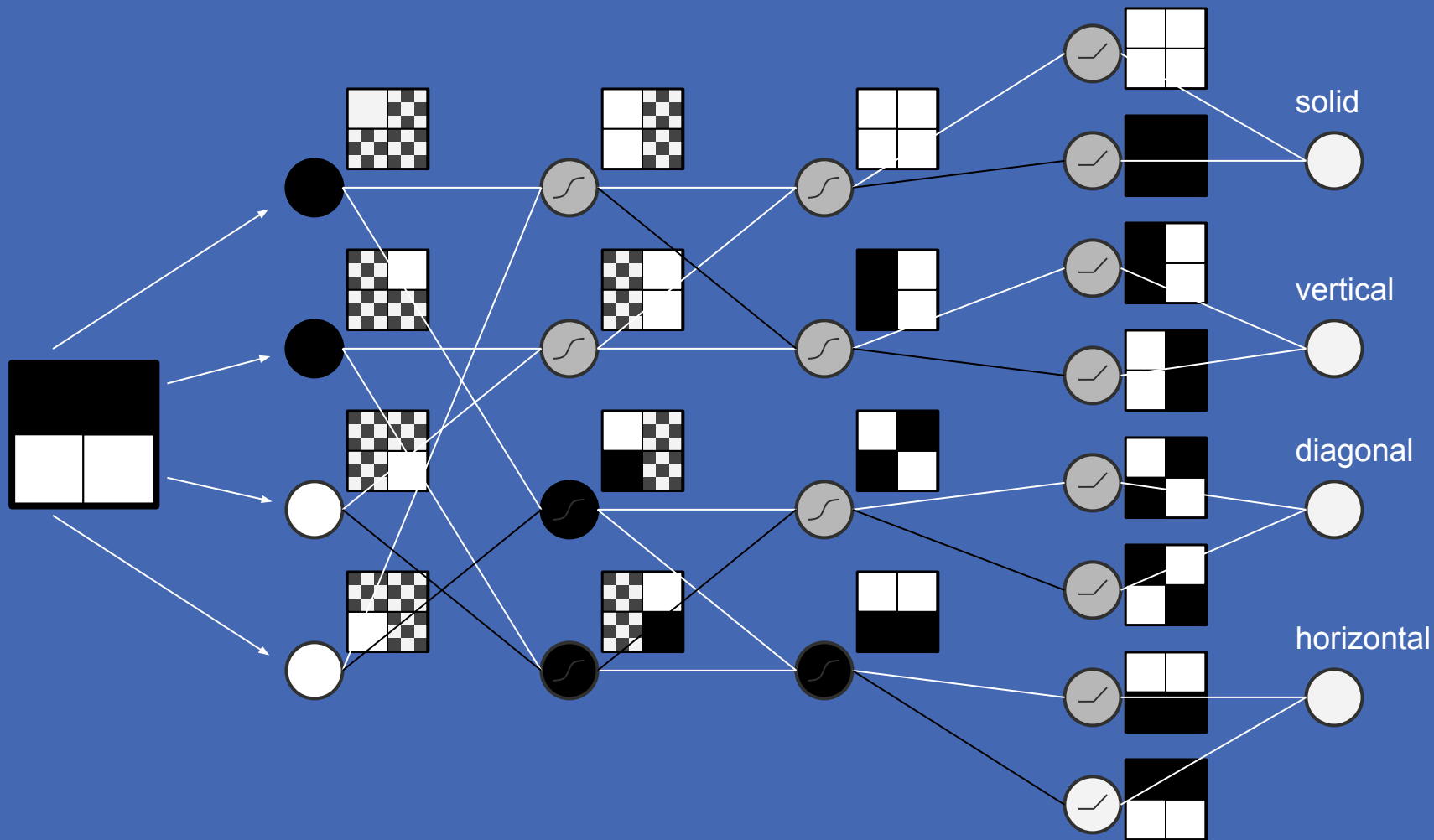


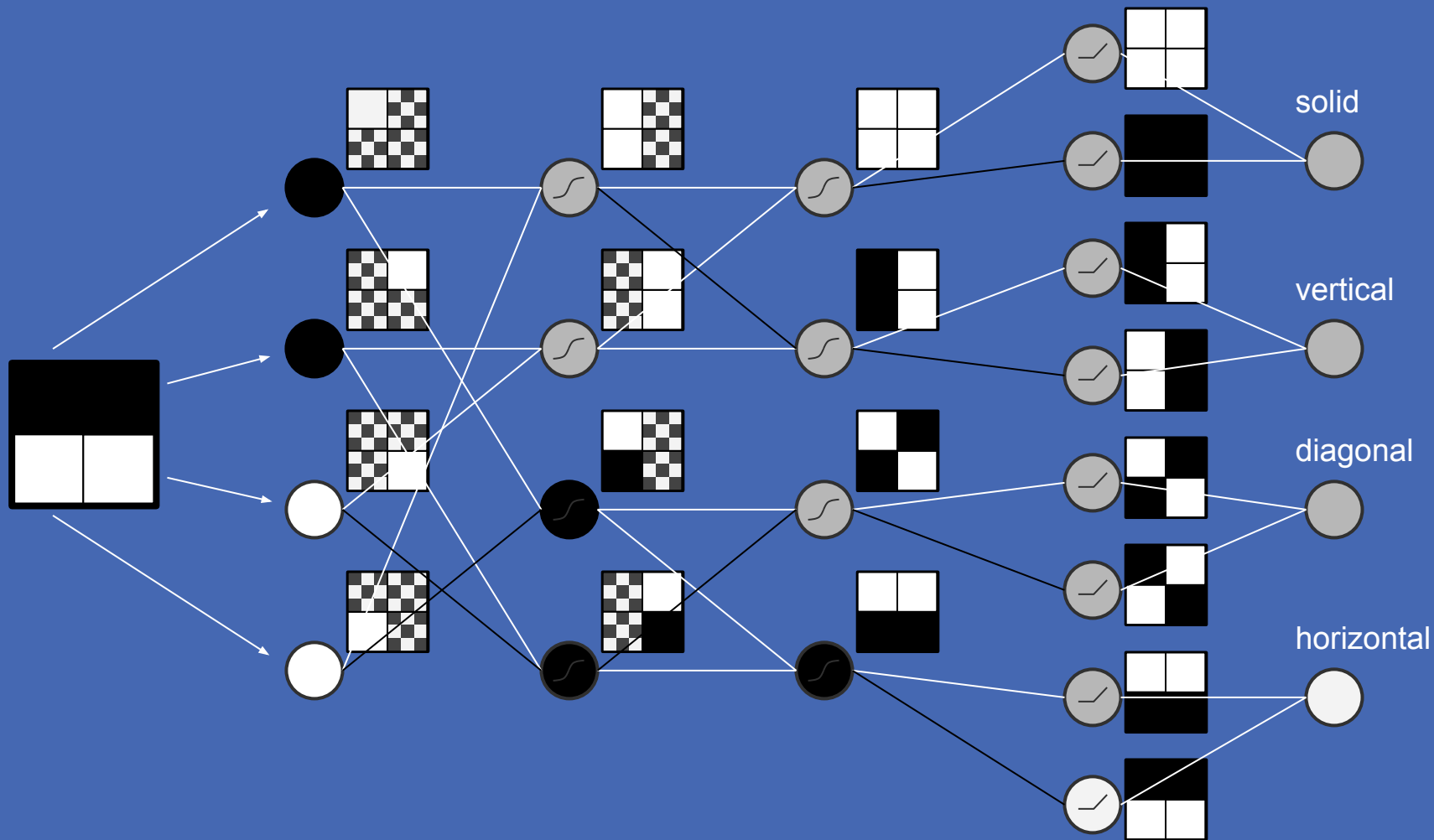


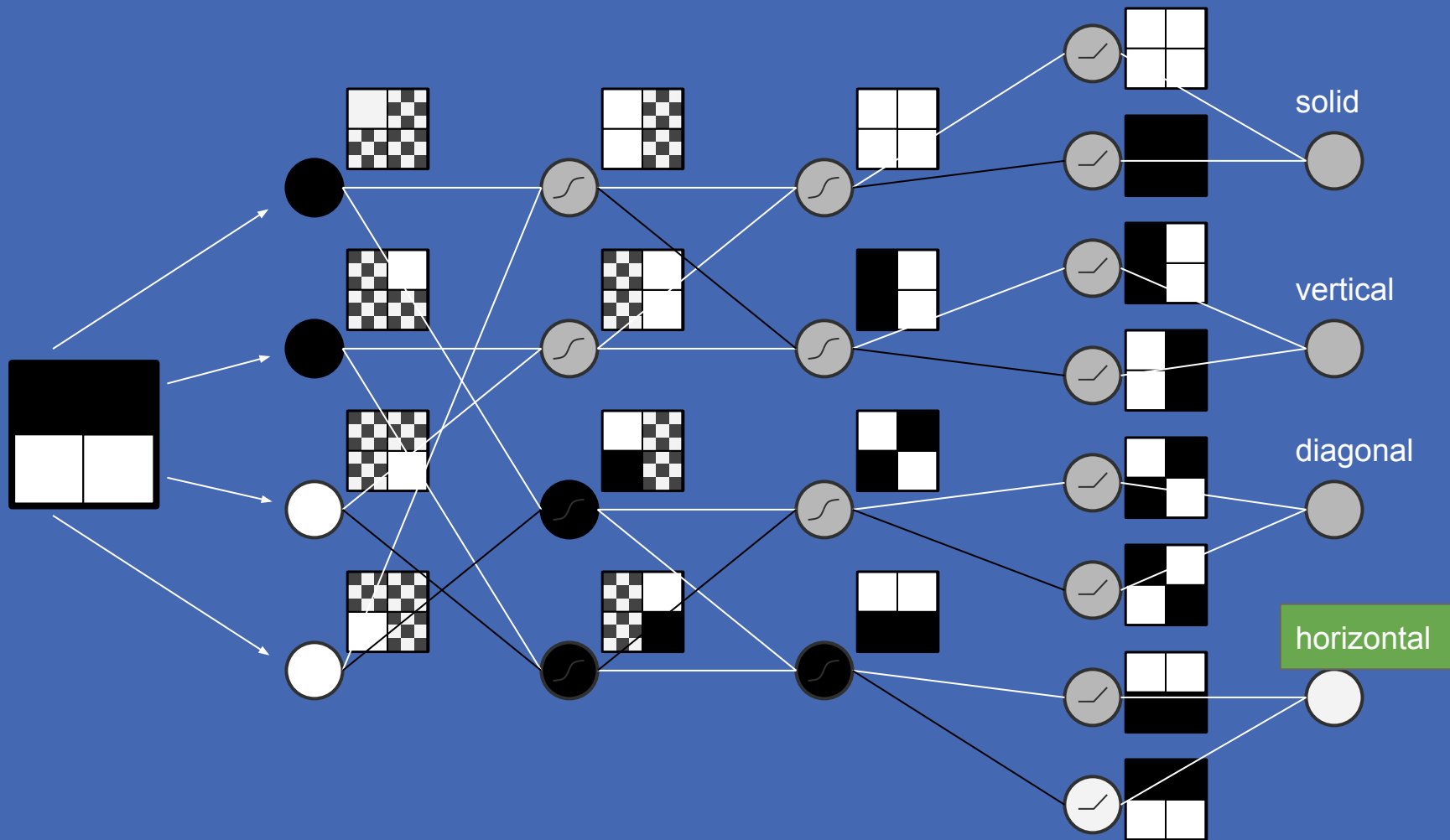




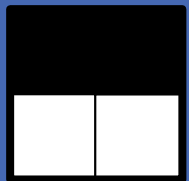








Errors



truth

0.

solid



vertical

0.



diagonal

0.

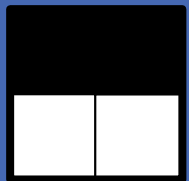






horizontal

1.

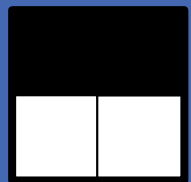






Errors



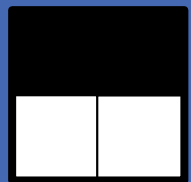
truth	answer	solid
0.	.5	
0.	.75	
0.	-.25	
1.	-.75	
		vertical
		diagonal
		horizontal





Errors



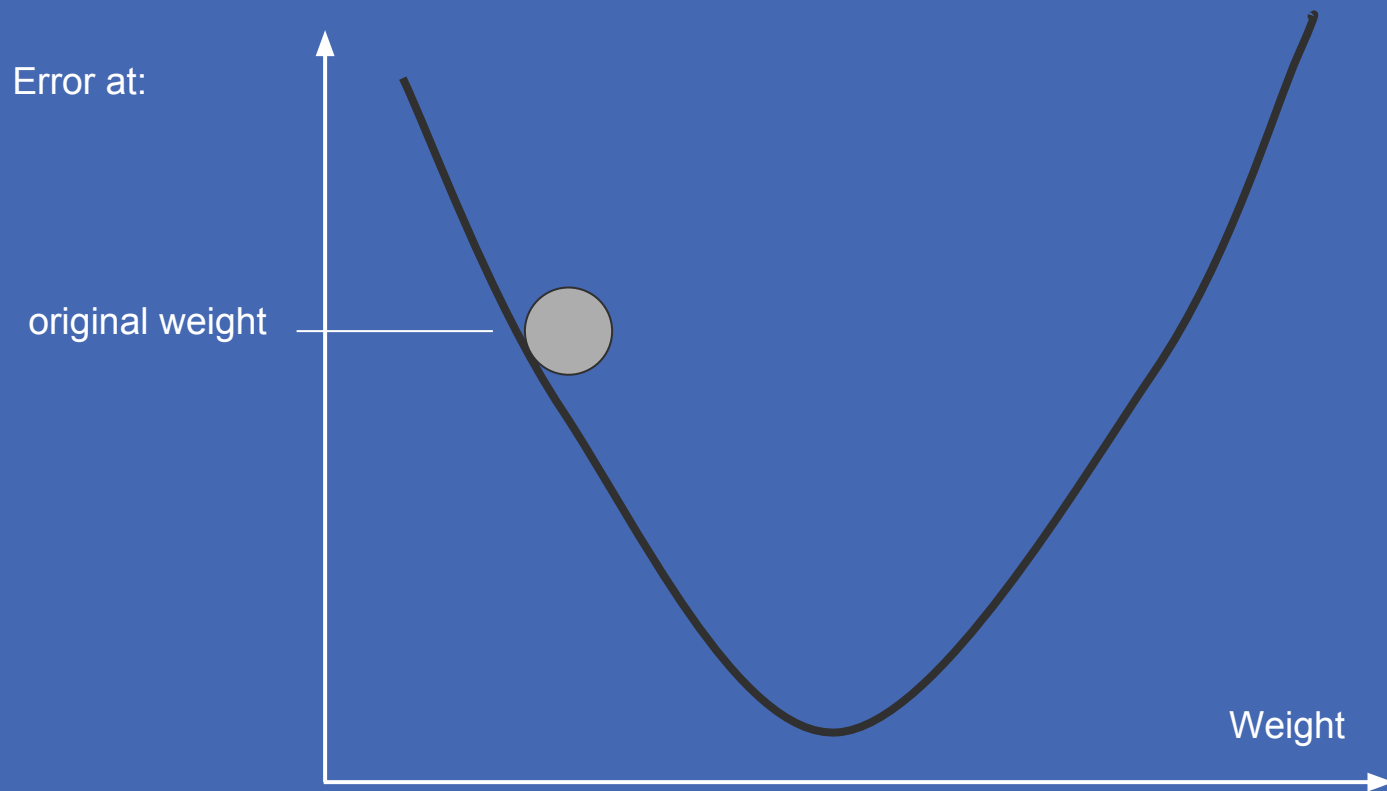
error	truth	answer	solid
.5	0.	.5	
.75	0.	.75	
.25	0.	-.25	
1.75	1.	-.75	
			vertical
			diagonal
			horizontal

Errors

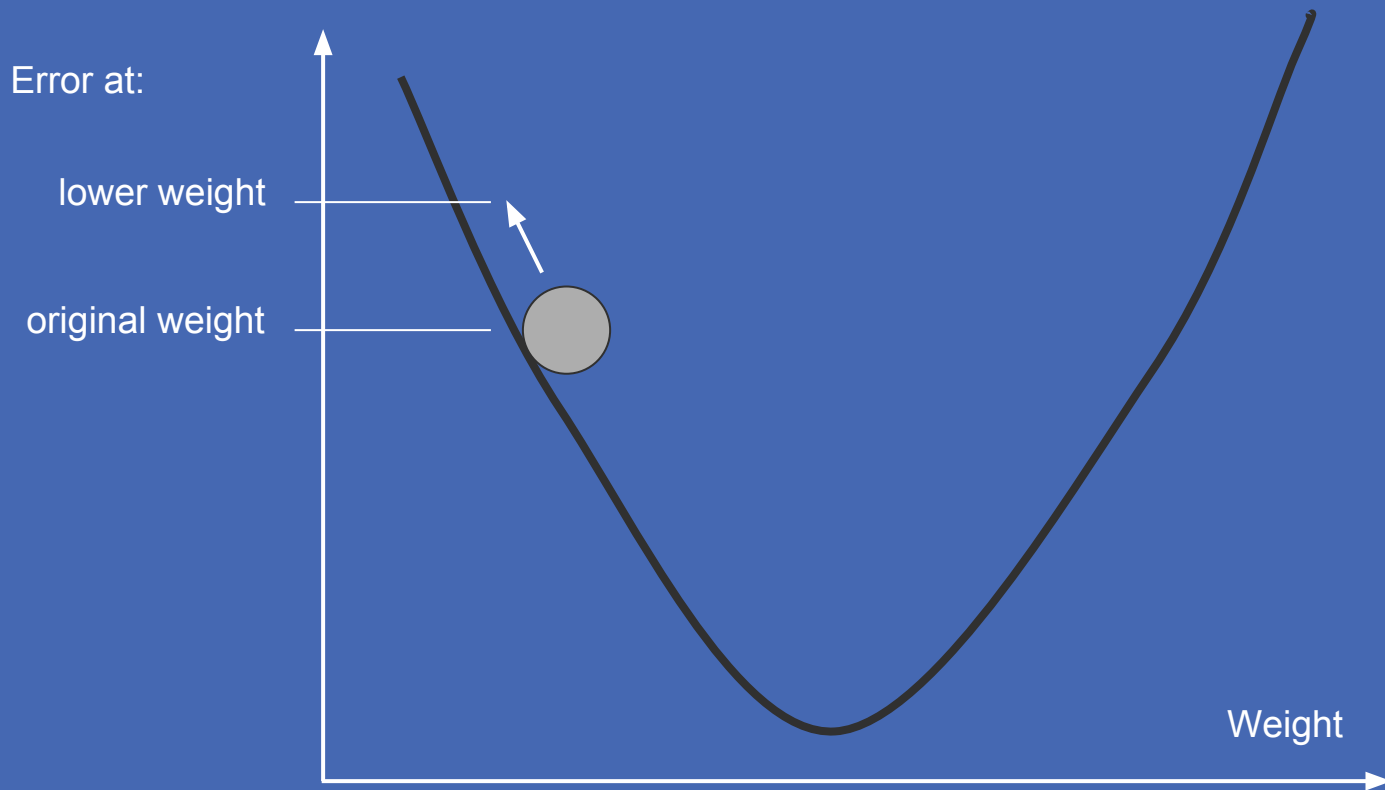


	error	truth	answer	solid
	.5	0.	.5	
	.75	0.	.75	
	.25	0.	-.25	
	1.75	1.	-.75	
total	3.25			
				vertical
				diagonal
				horizontal

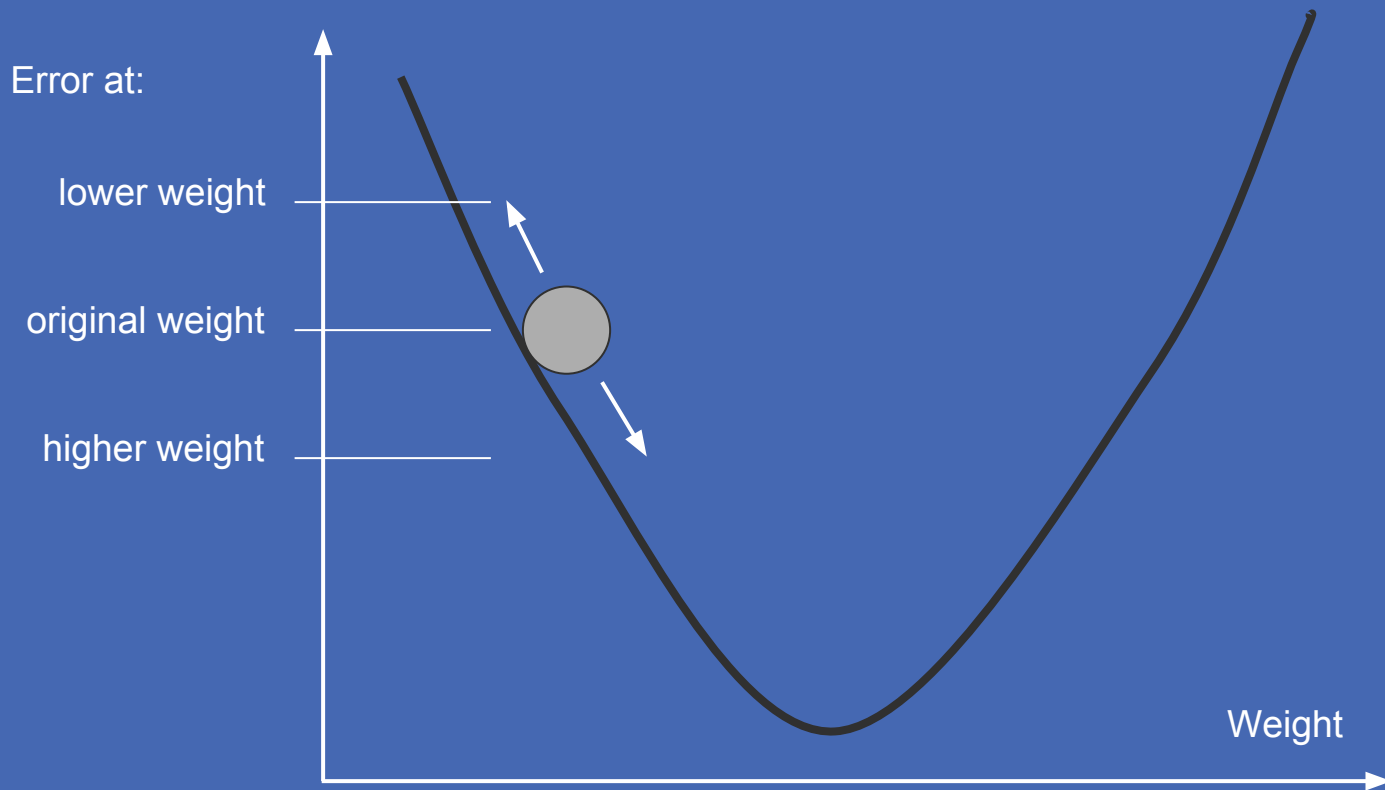
Learn all the weights: Gradient descent



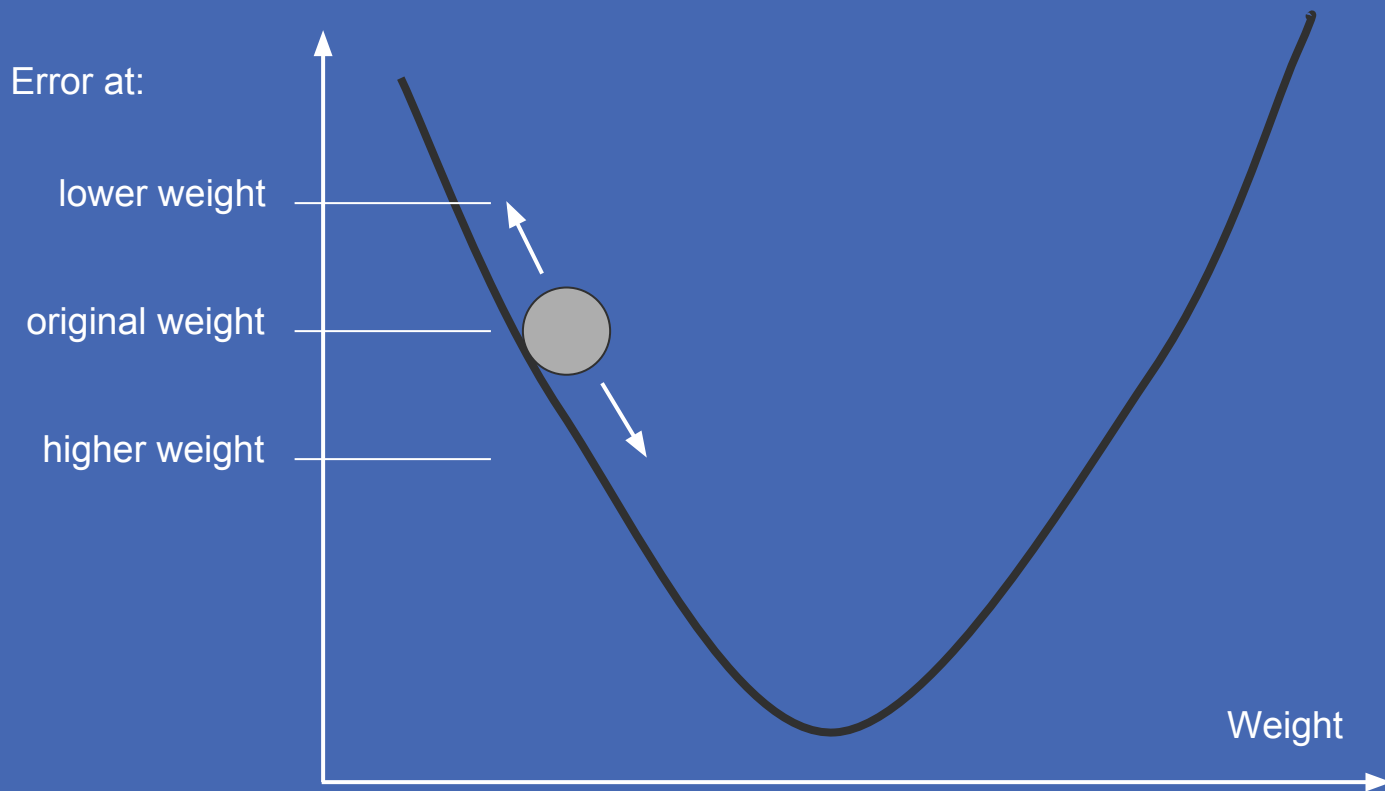
Learn all the weights: Gradient descent



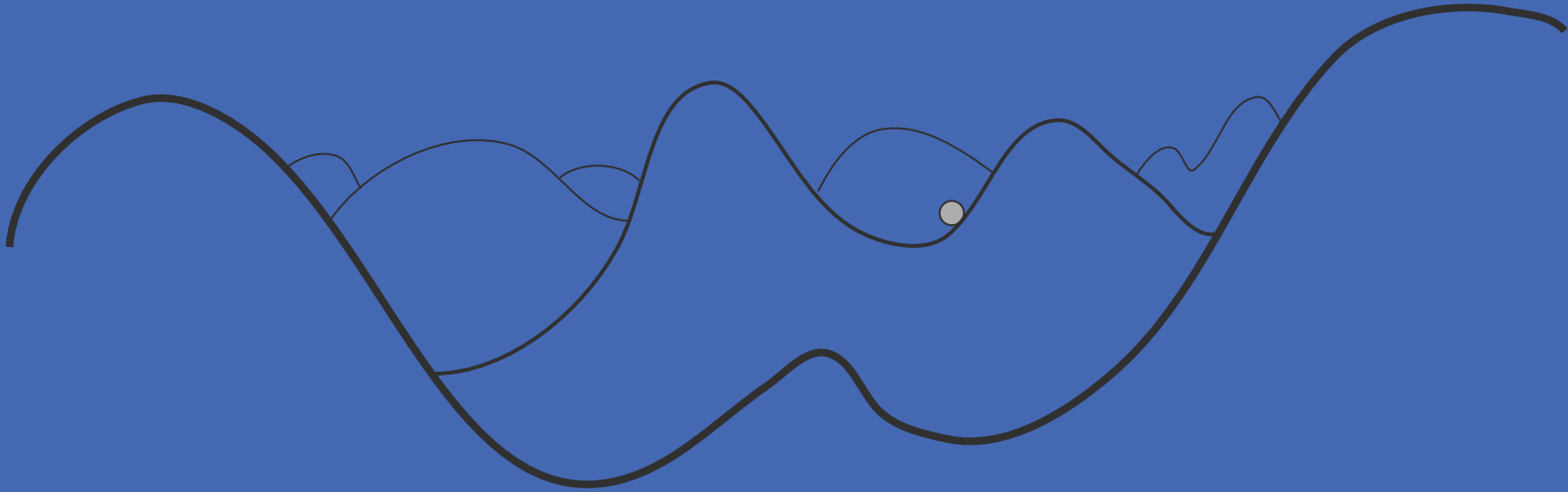
Learn all the weights: Gradient descent



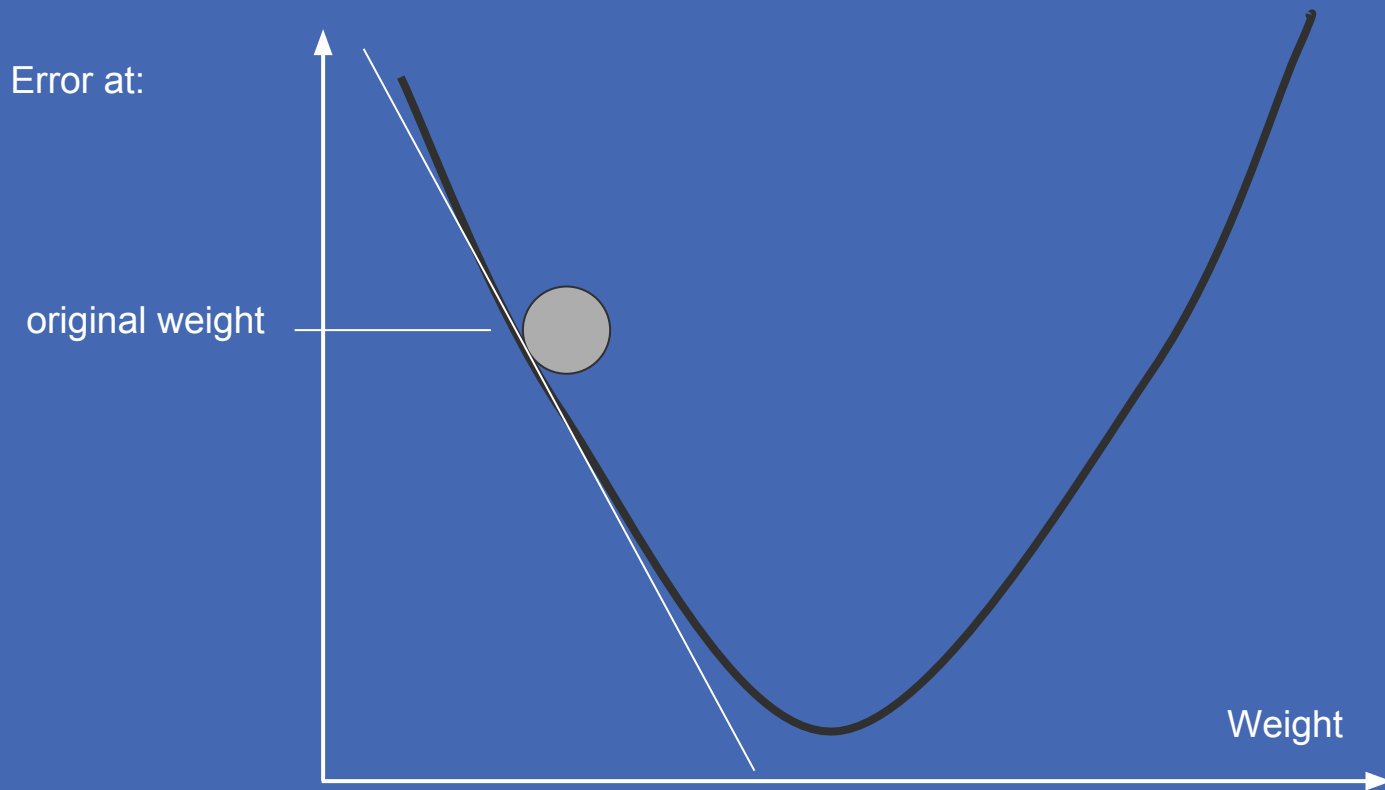
Numerically calculating the gradient is expensive



Numerically calculating the gradient is very expensive



Calculate the gradient (slope) directly



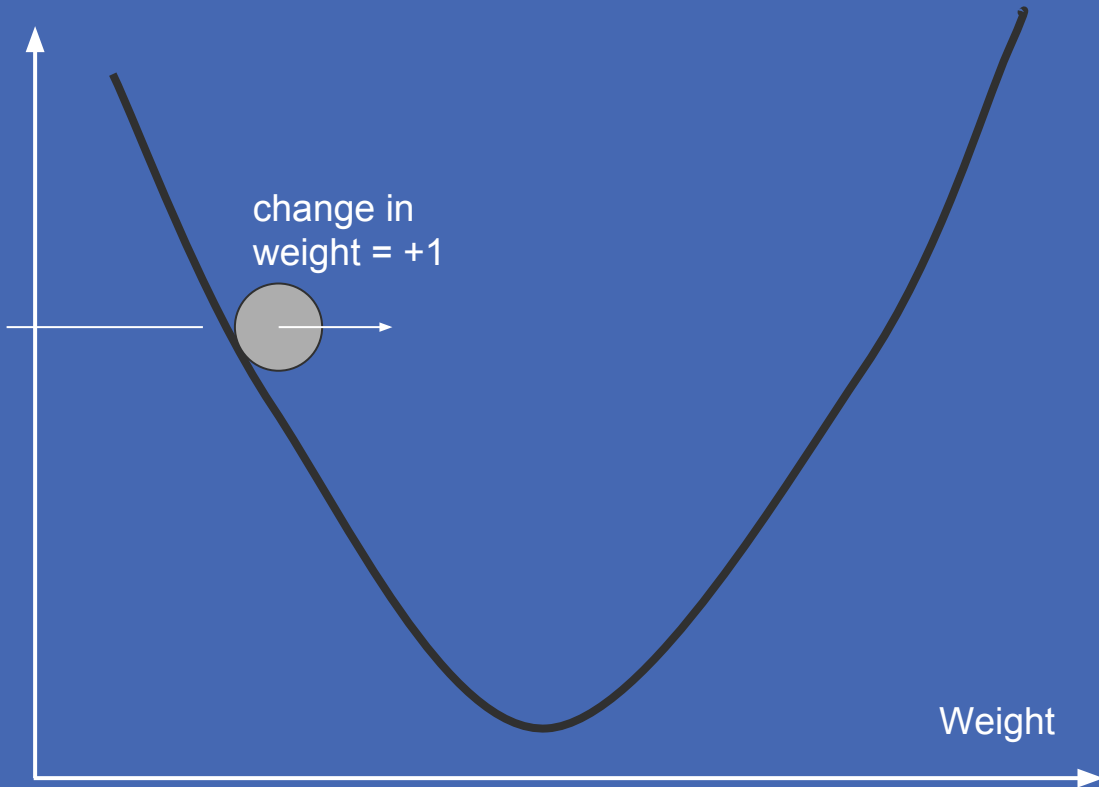
Slope

Error at:

original weight

change in
weight = +1

Weight



Slope

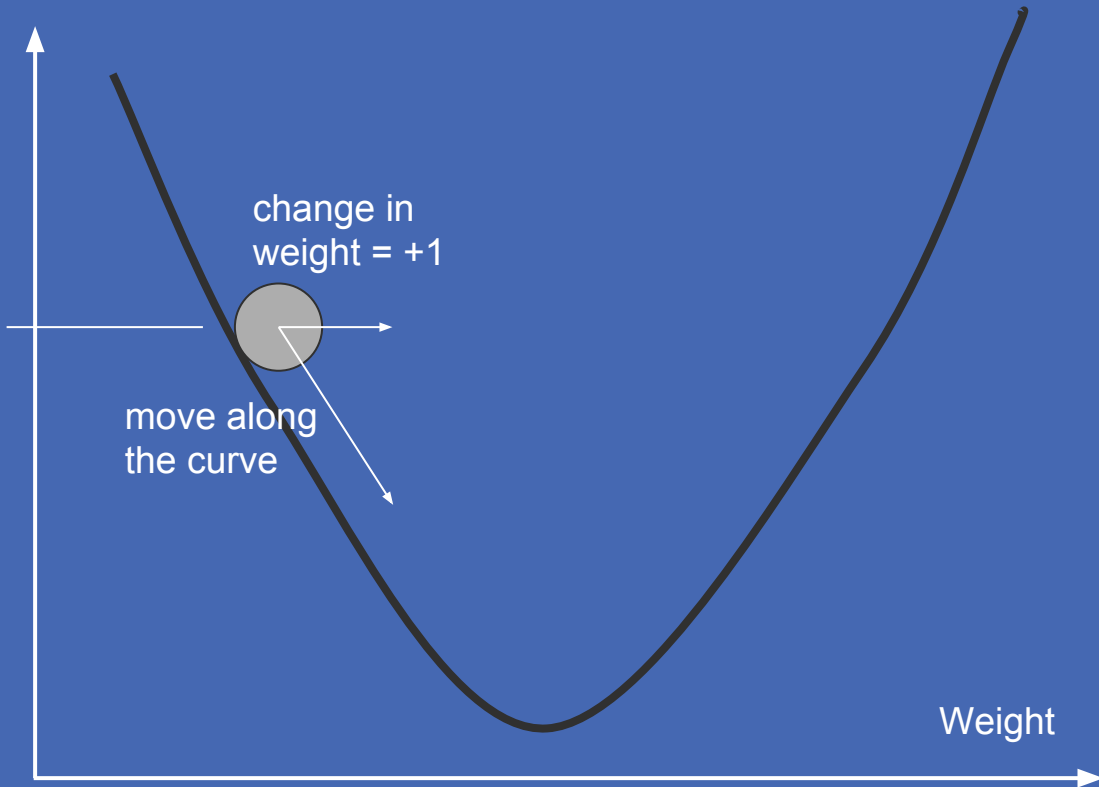
Error at:

original weight

change in
weight = +1

move along
the curve

Weight



Slope

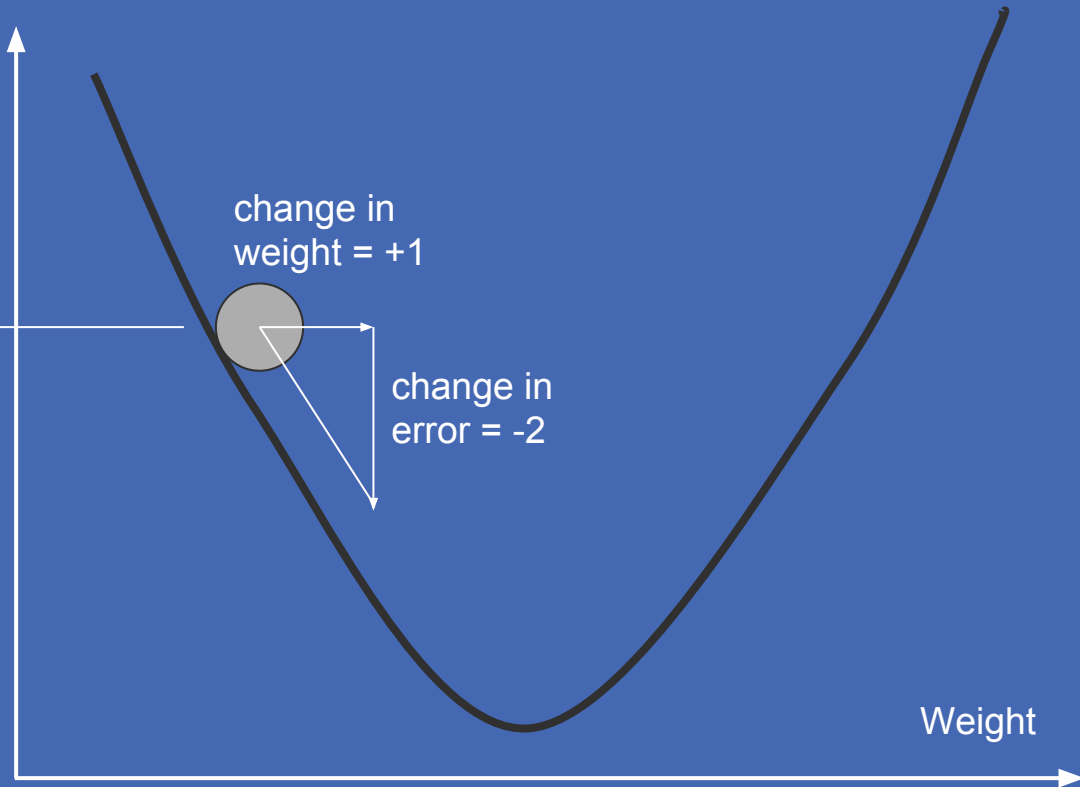
Error at:

original weight

change in
weight = +1

change in
error = -2

Weight



Slope

$$\text{slope} = \frac{\text{change in error}}{\text{change in weight}}$$

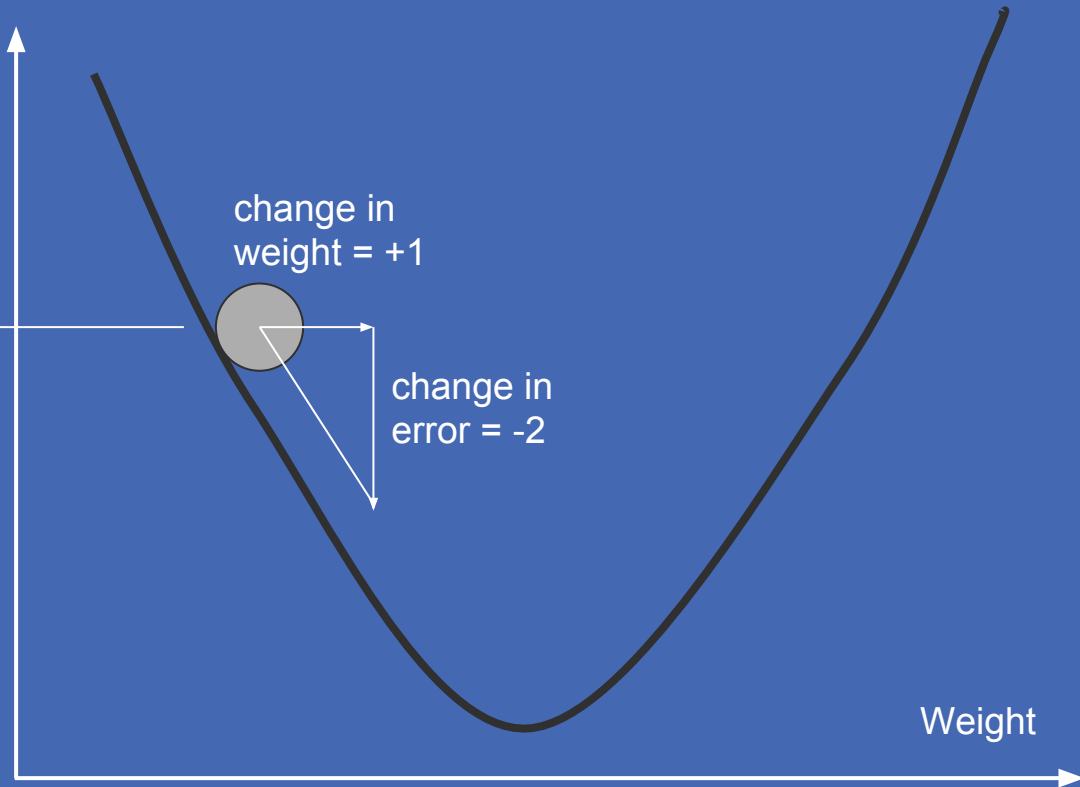
Error at:

original weight

change in
weight = +1

change in
error = -2

Weight



Slope

Error at:

original weight

change in
weight = +1

change in
error = -2

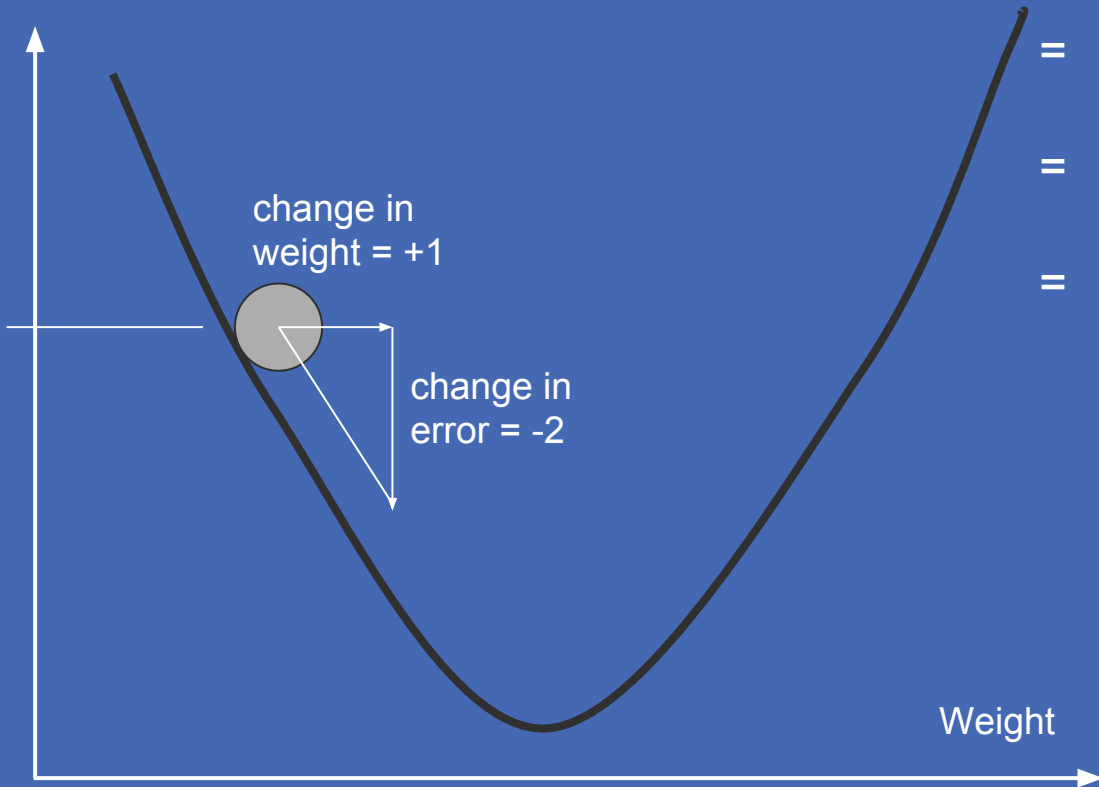
$$\text{slope} = \frac{\text{change in error}}{\text{change in weight}}$$

$$= \frac{\Delta \text{ error}}{\Delta \text{ weight}}$$

$$= \frac{d(\text{error})}{d(\text{weight})}$$

$$= \frac{\partial e}{\partial w}$$

Weight



Slope

Error at:

original weight

change in
weight = +1

change in
error = -2

$$\text{slope} = \frac{\text{change in error}}{\text{change in weight}}$$

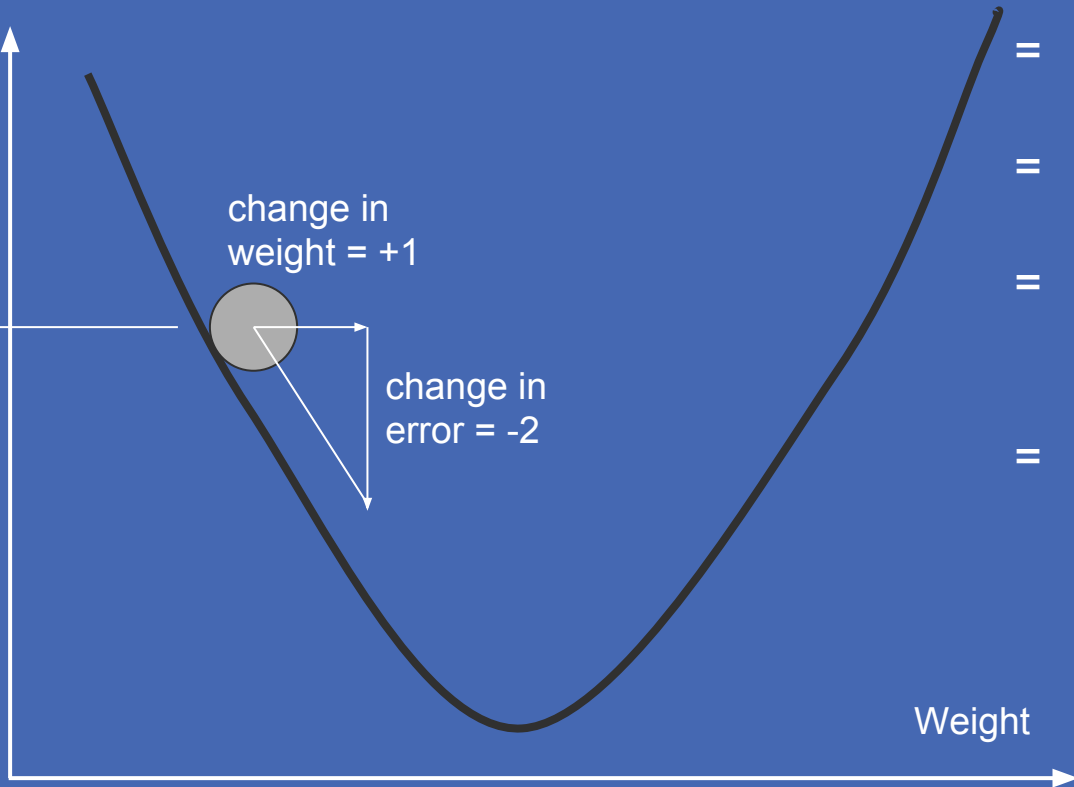
$$= \frac{\Delta \text{error}}{\Delta \text{weight}}$$

$$= \frac{d(\text{error})}{d(\text{weight})}$$

$$= \frac{\partial e}{\partial w}$$

$$= \frac{-2}{+1} = -2$$

Weight



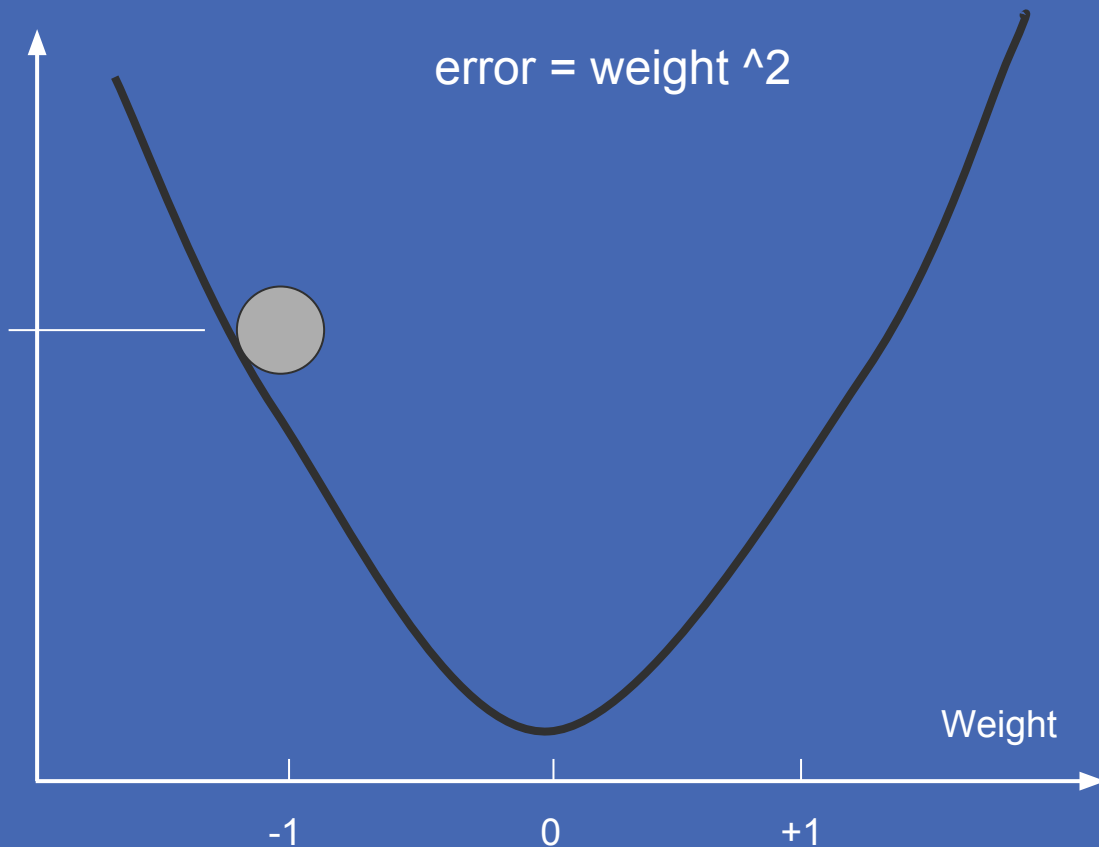
Slope

You have to know your error function.
For example:

$$\text{error} = \text{weight}^2$$

Error at:

original weight



Slope

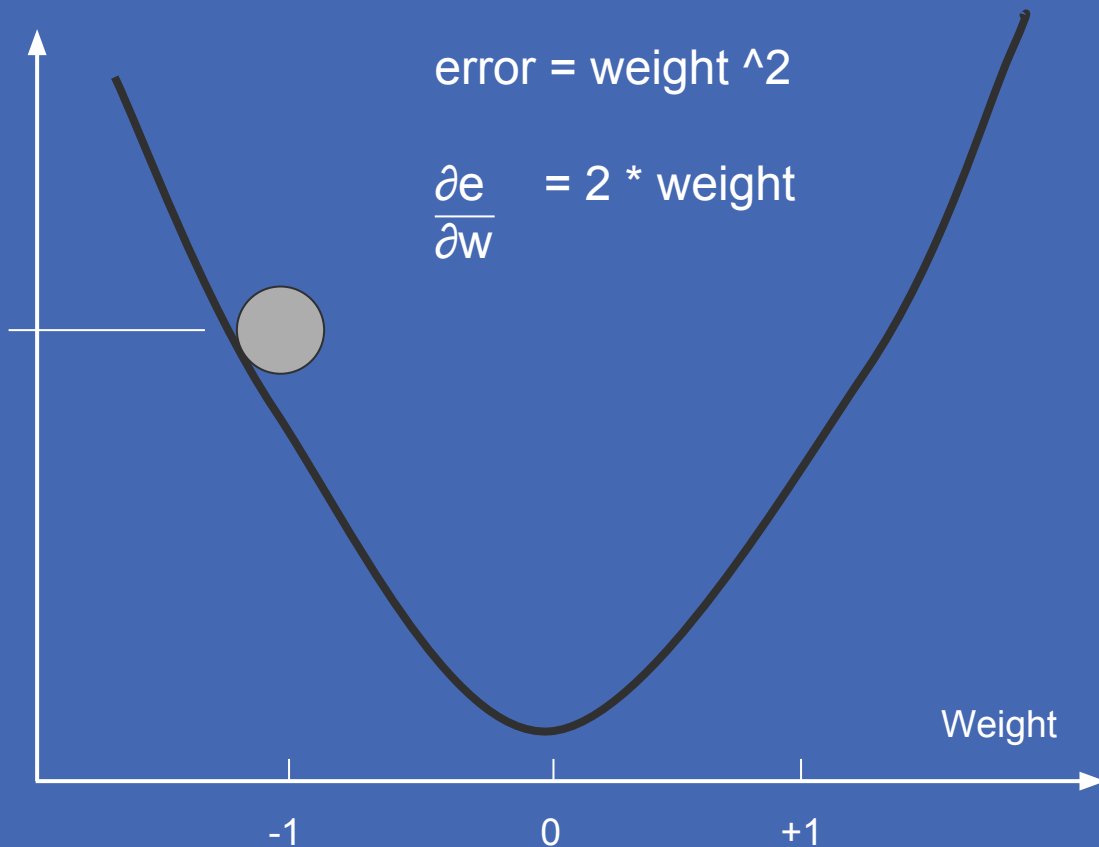
You have to know your error function.
For example:

$$\text{error} = \text{weight}^2$$

$$\frac{\partial e}{\partial w} = 2 * \text{weight}$$

Error at:

original weight



Slope

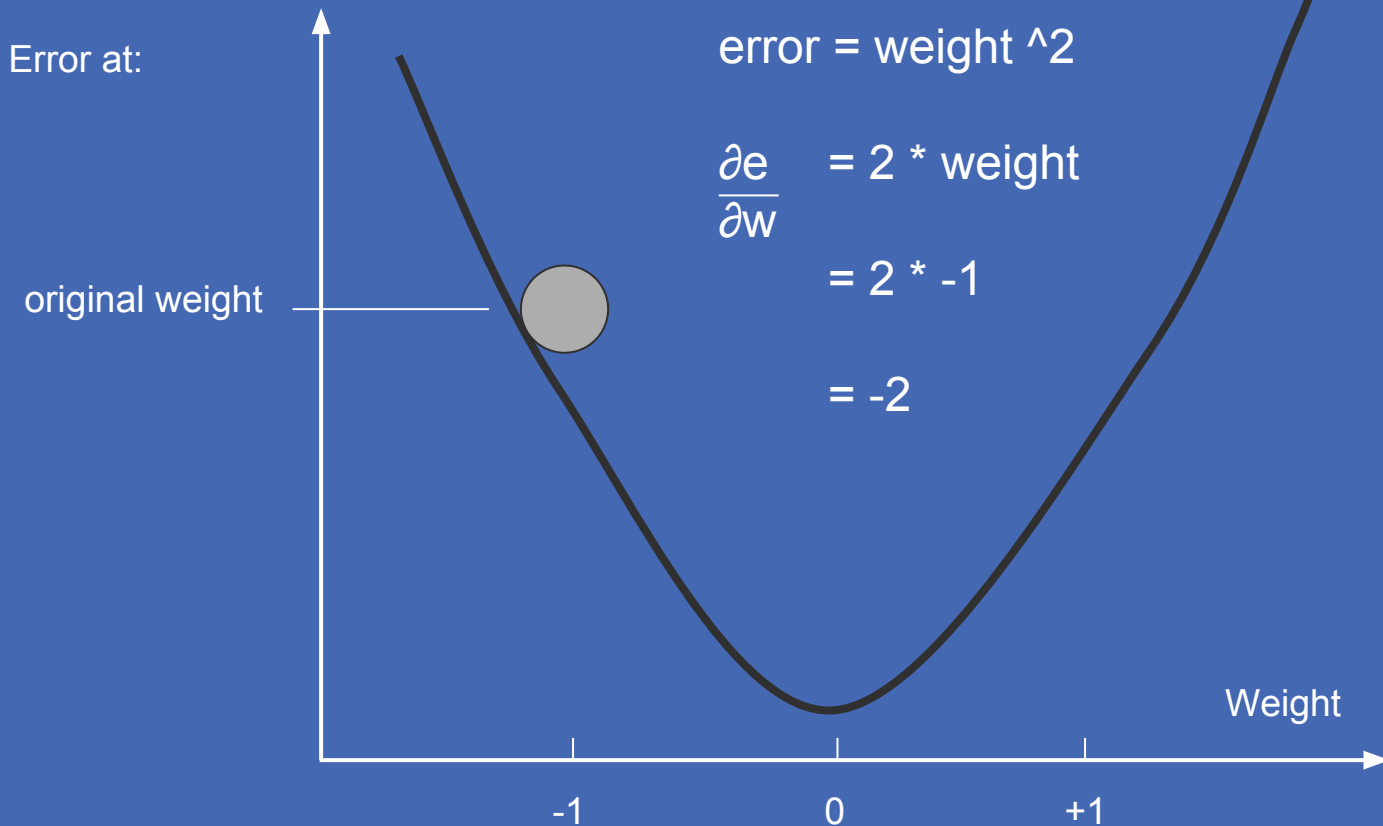
You have to know your error function.
For example:

$$\text{error} = \text{weight}^2$$

$$\frac{\partial e}{\partial w} = 2 * \text{weight}$$

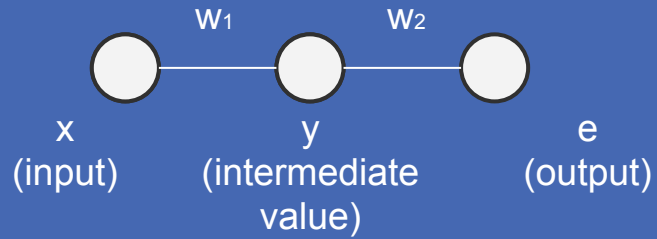
$$= 2 * -1$$

$$= -2$$



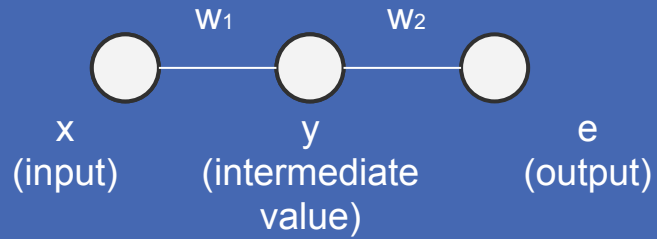
Chaining

$$y = x * w_1$$



Chaining

$$y = x * w_1$$
$$\frac{\partial y}{\partial w_1} = x$$



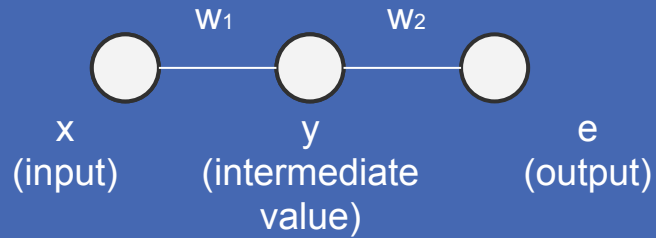
Chaining

$$y = x * w_1$$

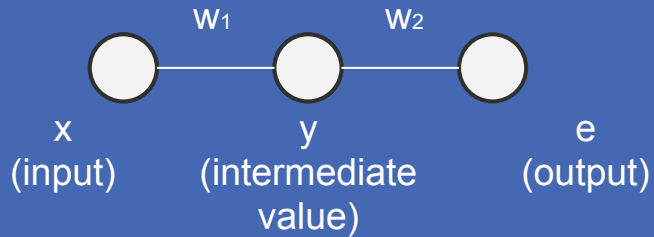
$$\frac{\partial y}{\partial w_1} = x$$

$$e = y * w_2$$

$$\frac{\partial e}{\partial y} = w_2$$



Chaining

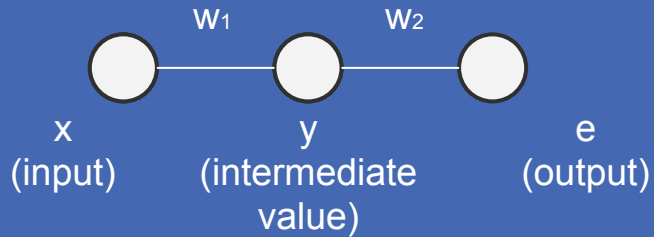


$$y = x * w_1$$
$$\frac{\partial y}{\partial w_1} = x$$

$$e = y * w_2$$
$$\frac{\partial e}{\partial y} = w_2$$

$$e = x * w_1 * w_2$$
$$\frac{\partial e}{\partial w_1} = x * w_2$$

Chaining



$$y = x * w_1$$

$$\frac{\partial y}{\partial w_1} = x$$

$$e = y * w_2$$

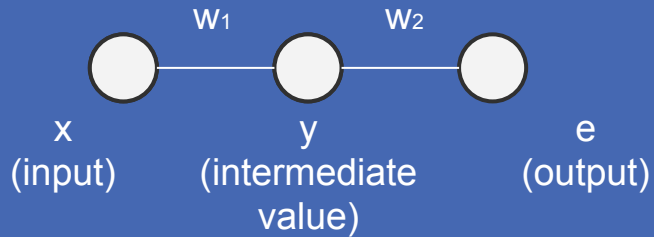
$$\frac{\partial e}{\partial y} = w_2$$

$$e = x * w_1 * w_2$$

$$\frac{\partial e}{\partial w_1} = x * w_2$$

$$\frac{\partial e}{\partial w_1} = \frac{\partial y}{\partial w_1} * \frac{\partial e}{\partial y}$$

Chaining



$$y = x * w_1$$
$$\frac{\partial y}{\partial w_1} = x$$

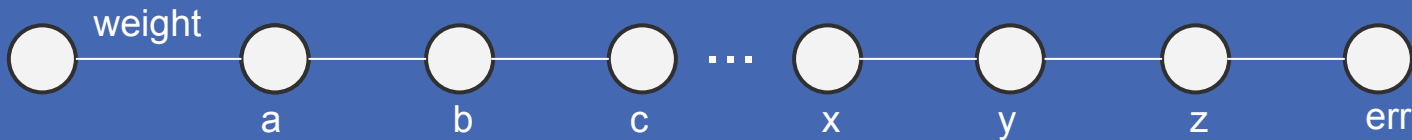
$$e = y * w_2$$
$$\frac{\partial e}{\partial y} = w_2$$

$$e = x * w_1 * w_2$$
$$\frac{\partial e}{\partial w_1} = x * w_2$$

$$\frac{\partial e}{\partial w_1} = \frac{\partial y}{\partial w_1} * \frac{\partial e}{\partial y}$$

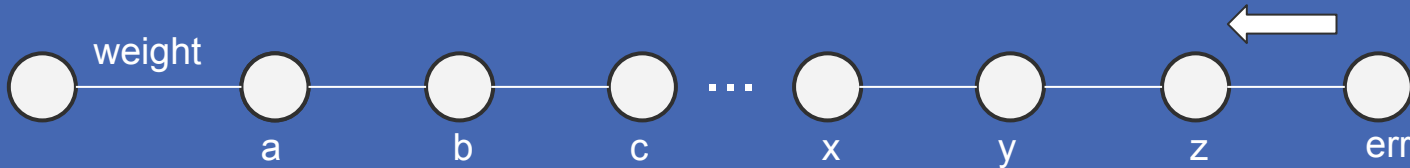
Chaining

$$\frac{\partial \text{err}}{\partial \text{weight}} = \frac{\partial a}{\partial \text{weight}} * \frac{\partial b}{\partial a} * \frac{\partial c}{\partial b} * \frac{\partial d}{\partial c} * \dots * \frac{\partial y}{\partial x} * \frac{\partial z}{\partial y} * \frac{\partial \text{err}}{\partial z}$$



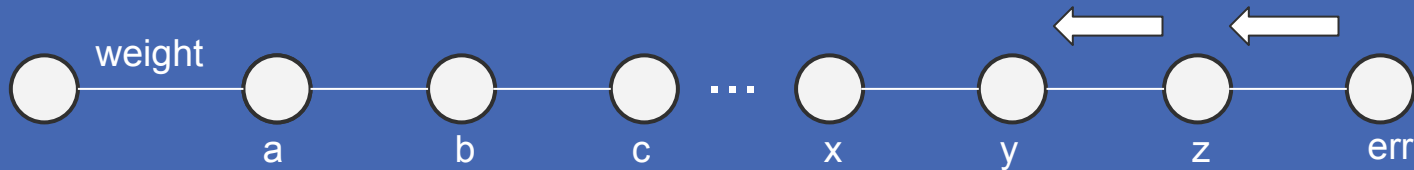
Backpropagation

$$\frac{\partial \text{err}}{\partial \text{weight}} = \frac{\partial a}{\partial \text{weight}} * \frac{\partial b}{\partial a} * \frac{\partial c}{\partial b} * \frac{\partial d}{\partial c} * \dots * \frac{\partial y}{\partial x} * \frac{\partial z}{\partial y} * \frac{\partial \text{err}}{\partial z}$$



Backpropagation

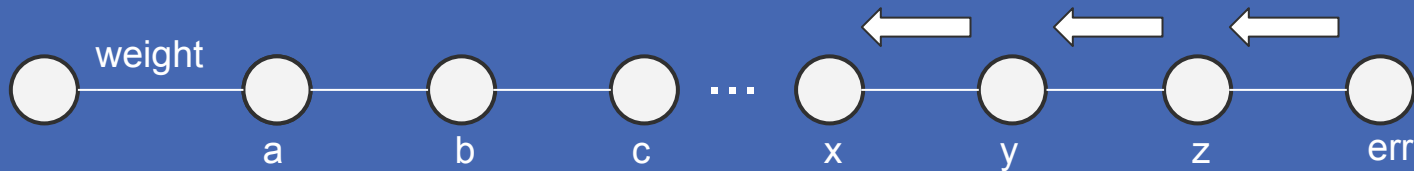
$$\frac{\partial \text{err}}{\partial \text{weight}} = \frac{\partial a}{\partial \text{weight}} * \frac{\partial b}{\partial a} * \frac{\partial c}{\partial b} * \frac{\partial d}{\partial c} * \dots * \frac{\partial y}{\partial x} * \frac{\partial z}{\partial y} * \frac{\partial \text{err}}{\partial z}$$



Backpropagation

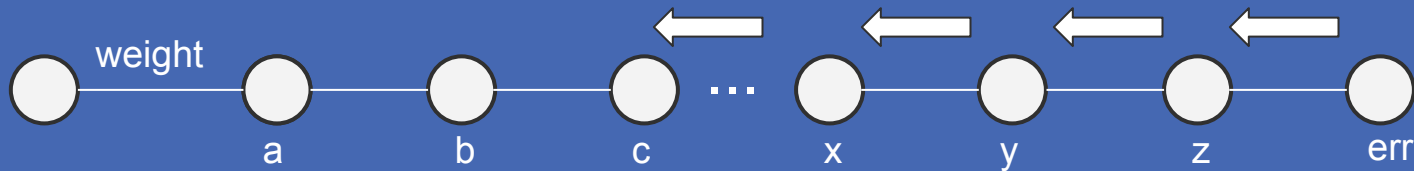
$$\frac{\partial \text{err}}{\partial \text{weight}} = \frac{\partial a}{\partial \text{weight}} * \frac{\partial b}{\partial a} * \frac{\partial c}{\partial b} * \frac{\partial d}{\partial c} * \dots * \frac{\partial y}{\partial x} * \frac{\partial z}{\partial y} * \frac{\partial \text{err}}{\partial z}$$

← ← ←



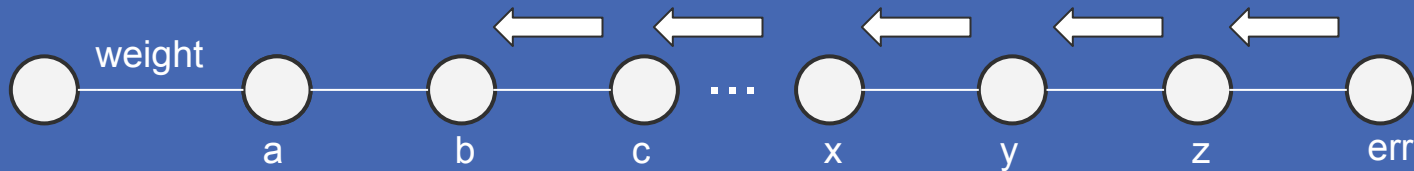
Backpropagation

$$\frac{\partial \text{err}}{\partial \text{weight}} = \frac{\partial a}{\partial \text{weight}} * \frac{\partial b}{\partial a} * \frac{\partial c}{\partial b} * \frac{\partial d}{\partial c} * \dots * \frac{\partial y}{\partial x} * \frac{\partial z}{\partial y} * \frac{\partial \text{err}}{\partial z}$$



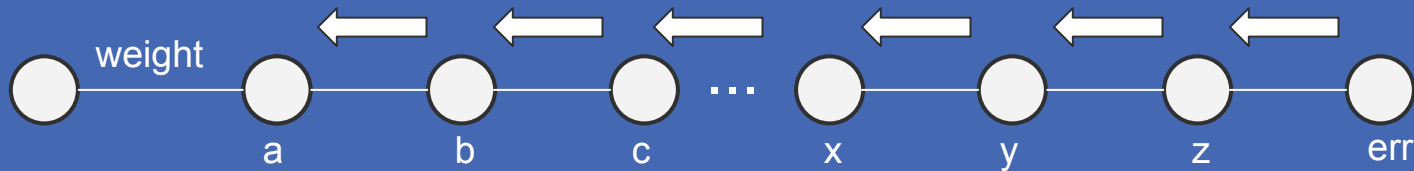
Backpropagation

$$\frac{\partial \text{err}}{\partial \text{weight}} = \frac{\partial a}{\partial \text{weight}} * \frac{\partial b}{\partial a} * \frac{\partial c}{\partial b} * \frac{\partial d}{\partial c} * \dots * \frac{\partial y}{\partial x} * \frac{\partial z}{\partial y} * \frac{\partial \text{err}}{\partial z}$$



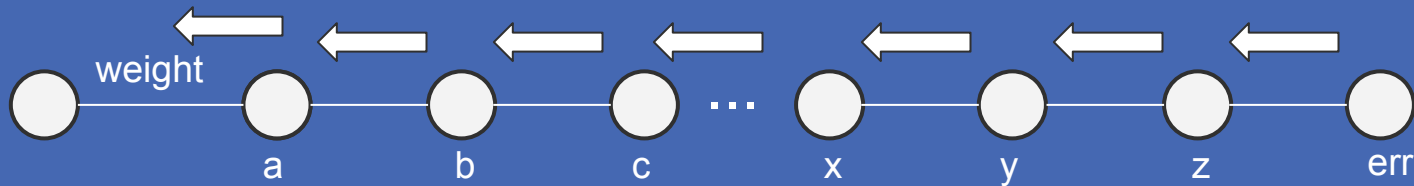
Backpropagation

$$\frac{\partial \text{err}}{\partial \text{weight}} = \frac{\partial a}{\partial \text{weight}} * \frac{\partial b}{\partial a} * \frac{\partial c}{\partial b} * \frac{\partial d}{\partial c} * \dots * \frac{\partial y}{\partial x} * \frac{\partial z}{\partial y} * \frac{\partial \text{err}}{\partial z}$$

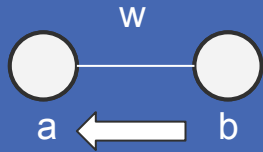


Backpropagation

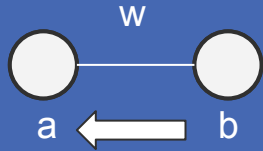
$$\frac{\partial \text{err}}{\partial \text{weight}} = \frac{\partial a}{\partial \text{weight}} * \frac{\partial b}{\partial a} * \frac{\partial c}{\partial b} * \frac{\partial d}{\partial c} * \dots * \frac{\partial y}{\partial x} * \frac{\partial z}{\partial y} * \frac{\partial \text{err}}{\partial z}$$



Backpropagation challenge: weights

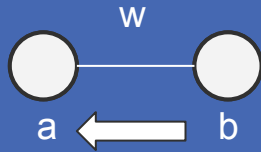


Backpropagation challenge: weights



$$\frac{\partial \text{err}}{\partial a} = \frac{\partial b}{\partial a} * \frac{\partial \text{err}}{\partial b}$$

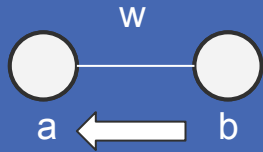
Backpropagation challenge: weights



$$b = wa$$


$$\frac{\partial \text{err}}{\partial a} = \frac{\partial b}{\partial a} * \frac{\partial \text{err}}{\partial b}$$

Backpropagation challenge: weights

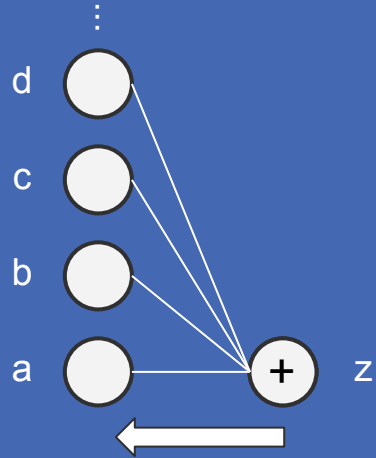


$$\frac{\partial \text{err}}{\partial a} = \frac{\partial b}{\partial a} * \frac{\partial \text{err}}{\partial b}$$

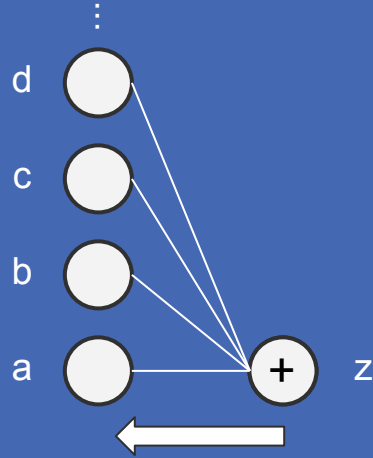
$$b = wa$$


$$\frac{\partial b}{\partial a} = w$$

Backpropagation challenge: sums

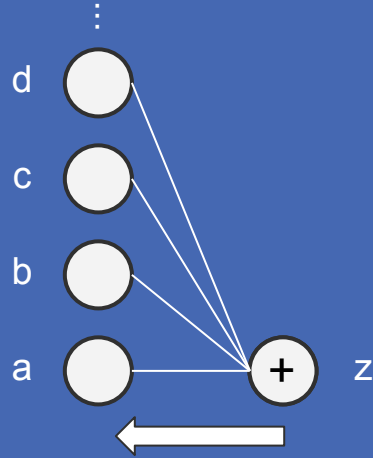


Backpropagation challenge: sums



$$\frac{\partial \text{err}}{\partial a} = \frac{\partial z}{\partial a} * \frac{\partial \text{err}}{\partial z}$$

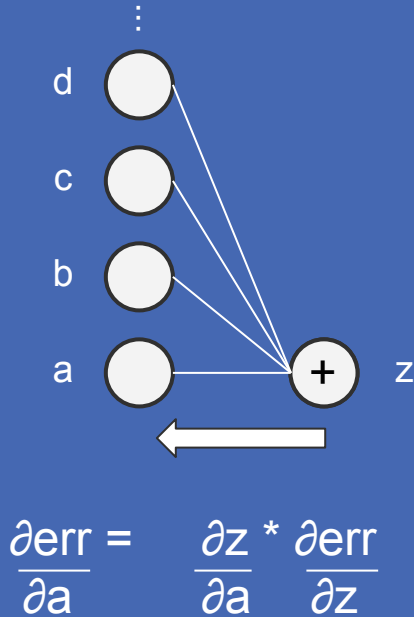
Backpropagation challenge: sums




$$z = a + b + c + d + \dots$$

$$\frac{\partial \text{err}}{\partial a} = \frac{\partial z}{\partial a} * \frac{\partial \text{err}}{\partial z}$$

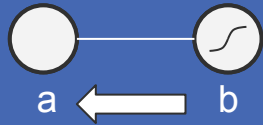
Backpropagation challenge: sums



$$z = a + b + c + d + \dots$$


$$\frac{\partial z}{\partial a} = 1$$

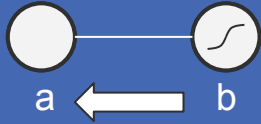
Backpropagation challenge: sigmoid



$$\frac{\partial \text{err}}{\partial a} = \frac{\partial b}{\partial a} * \frac{\partial \text{err}}{\partial b}$$

Backpropagation challenge: sigmoid

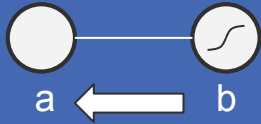
$$b = \frac{1}{1 + e^{-a}}$$



$$\frac{\partial \text{err}}{\partial a} = \frac{\partial b}{\partial a} * \frac{\partial \text{err}}{\partial b}$$

Backpropagation challenge: sigmoid

$$\begin{aligned} b &= \frac{1}{1 + e^{-a}} \\ &= \sigma(a) \end{aligned}$$

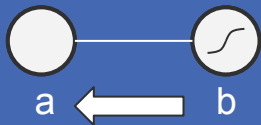


$$\frac{\partial \text{err}}{\partial a} = \frac{\partial b}{\partial a} * \frac{\partial \text{err}}{\partial b}$$


Backpropagation challenge: sigmoid

$$\begin{aligned}b &= \frac{1}{1 + e^{-a}} \\ &= \sigma(a)\end{aligned}$$

Because math is beautiful /
dumb luck:



$$\frac{\partial \text{err}}{\partial a} = \frac{\partial b}{\partial a} * \frac{\partial \text{err}}{\partial b}$$


$$\frac{\partial b}{\partial a} = \sigma(a) * (1 - \sigma(a))$$

Backpropagation challenge: ReLU



$$\frac{\partial \text{err}}{\partial a} = \frac{\partial b}{\partial a} * \frac{\partial \text{err}}{\partial b}$$

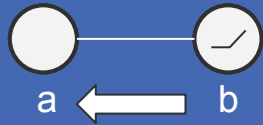
Backpropagation challenge: ReLU



$$\frac{\partial \text{err}}{\partial a} = \frac{\partial b}{\partial a} * \frac{\partial \text{err}}{\partial b}$$


$$\begin{aligned} b &= a, a > 0 \\ &= 0, \text{ otherwise} \end{aligned}$$

Backpropagation challenge: ReLU

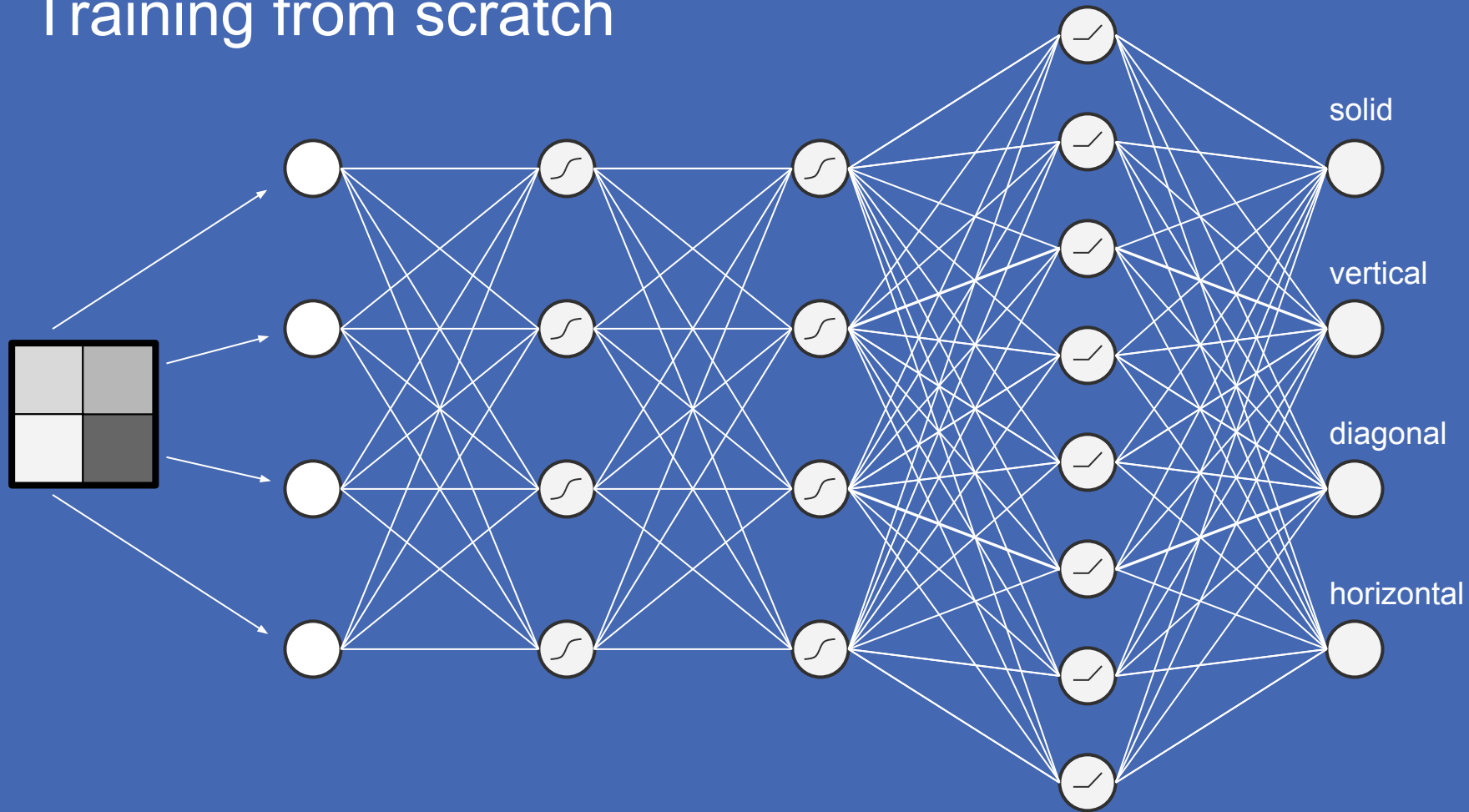


$$\frac{\partial \text{err}}{\partial a} = \frac{\partial b}{\partial a} * \frac{\partial \text{err}}{\partial b}$$

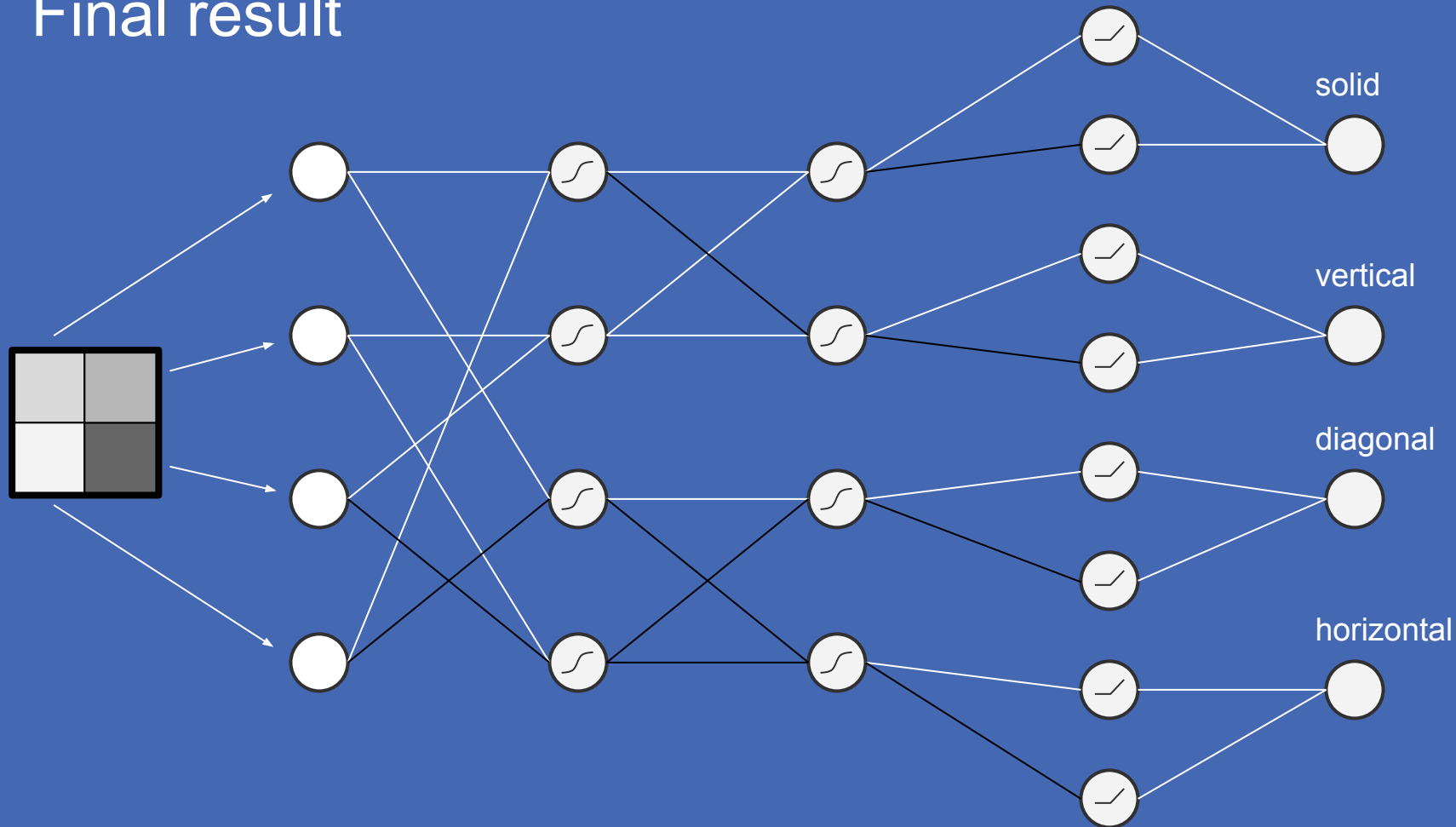
$$\begin{aligned} b &= a, a > 0 \\ &= 0, \text{ otherwise} \end{aligned}$$


$$\frac{\partial b}{\partial a} = \begin{aligned} &1, a > 0 \\ &0, \text{ otherwise} \end{aligned}$$

Training from scratch



Final result



Advanced topics

Bias neurons

Dropout

Backpropagation details

Andrej Karpathy's [Stanford CS231 lecture](#)

Backpropagation gotchas

Andrej Karpathy's article ["Yes you should understand backprop"](#)

Tips and tricks

Nikolas Markou's article ["The Black Magic of Deep Learning"](#)

Data Science and Robots Blog

For more How it Works:

[How Deep Learning works](#)

[How Convolutional Neural Networks work](#)

[How Bayes Law works](#)

[How data science works](#)

[How linear regression works](#)

[These slides](#)

<https://docs.google.com/presentation/d/1AAEFCgC0Ja7QEI3-wmuvlizbvaE-aQRksc7-W8LR2GY/edit?usp=sharing>