IDL

Neural Networks

TL;DR

Deep learning isn't magic.

But it is very good at finding patterns.

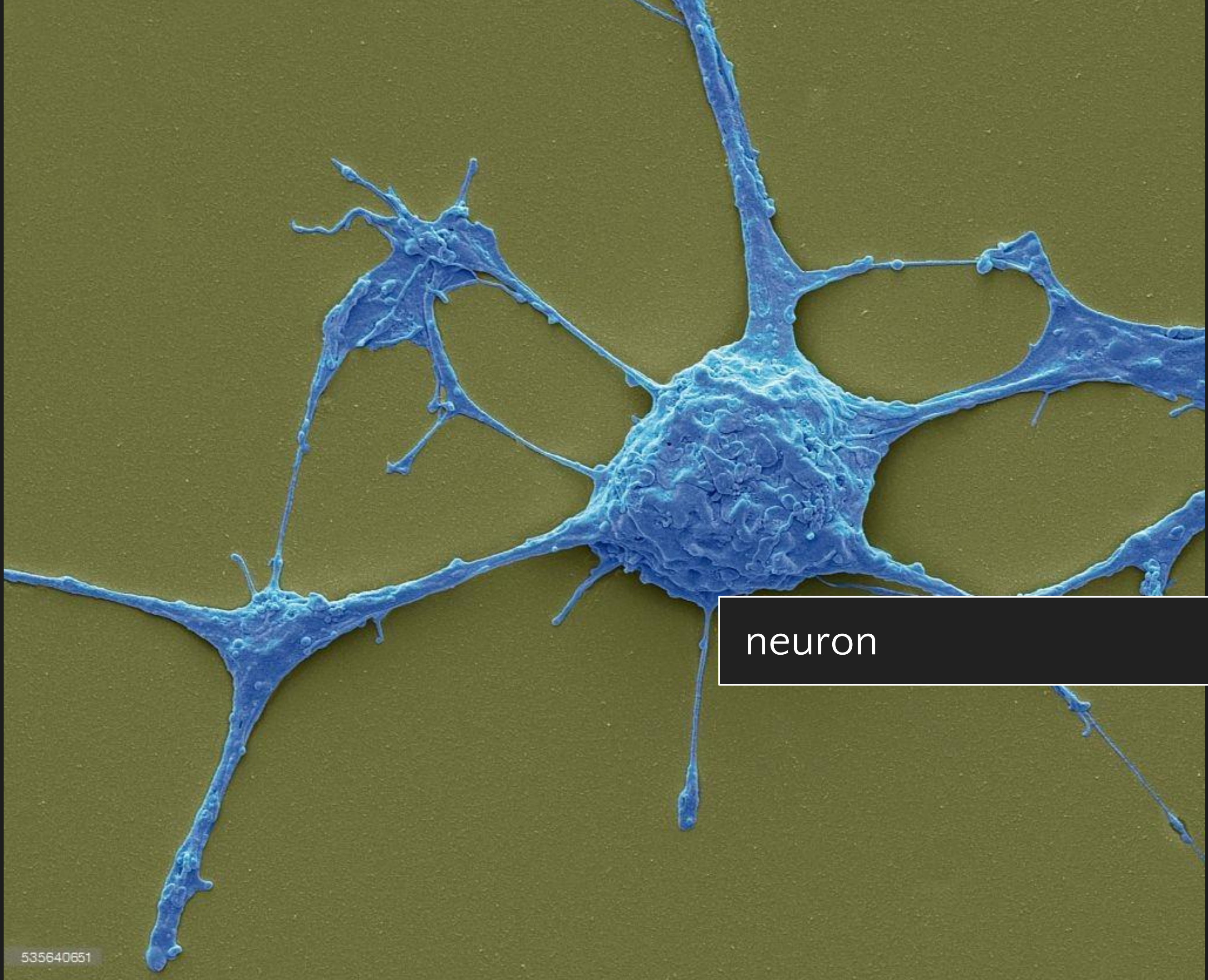
The brain and deep learning

The brain and deep learning

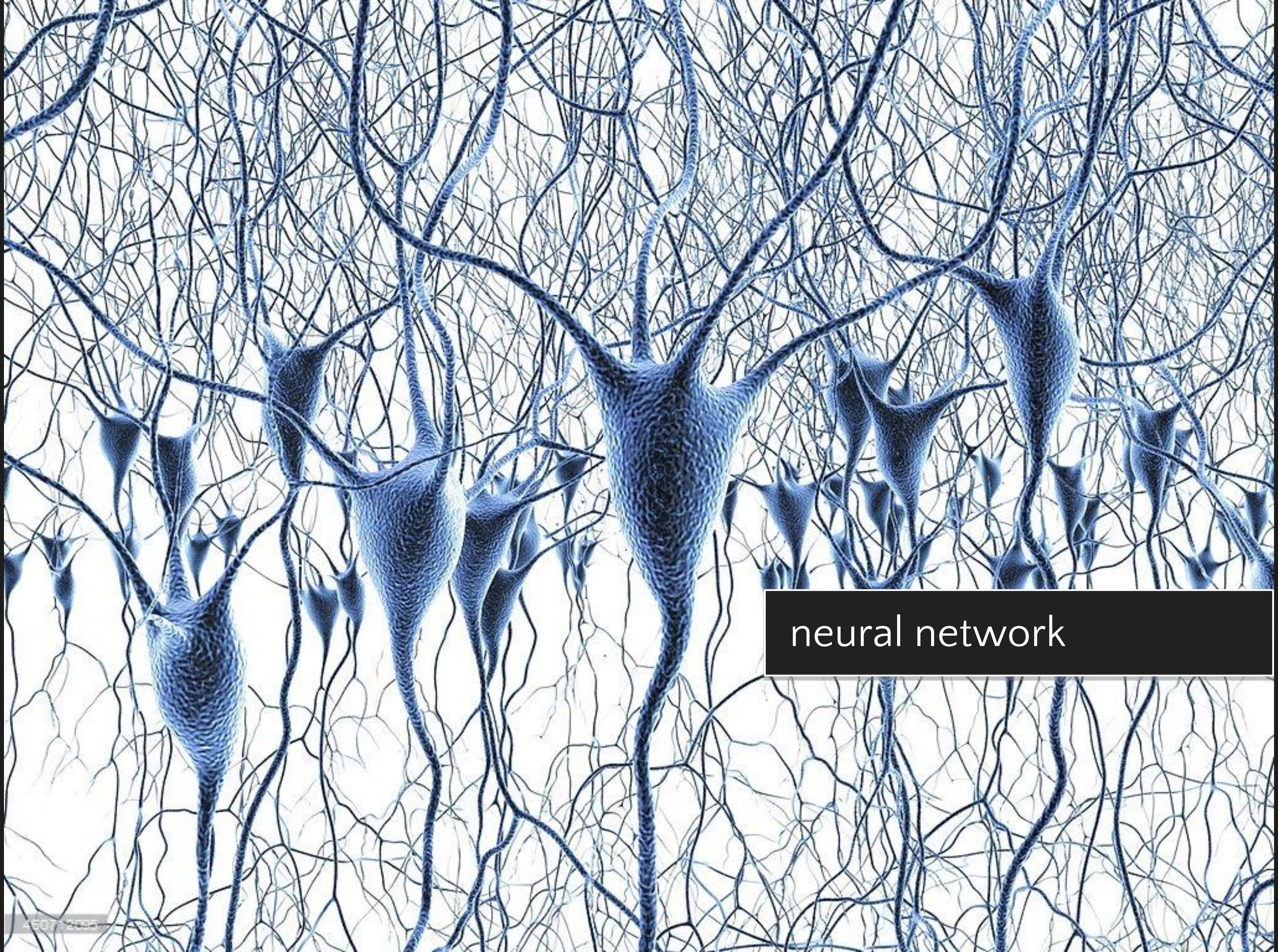
Creative Commons 3

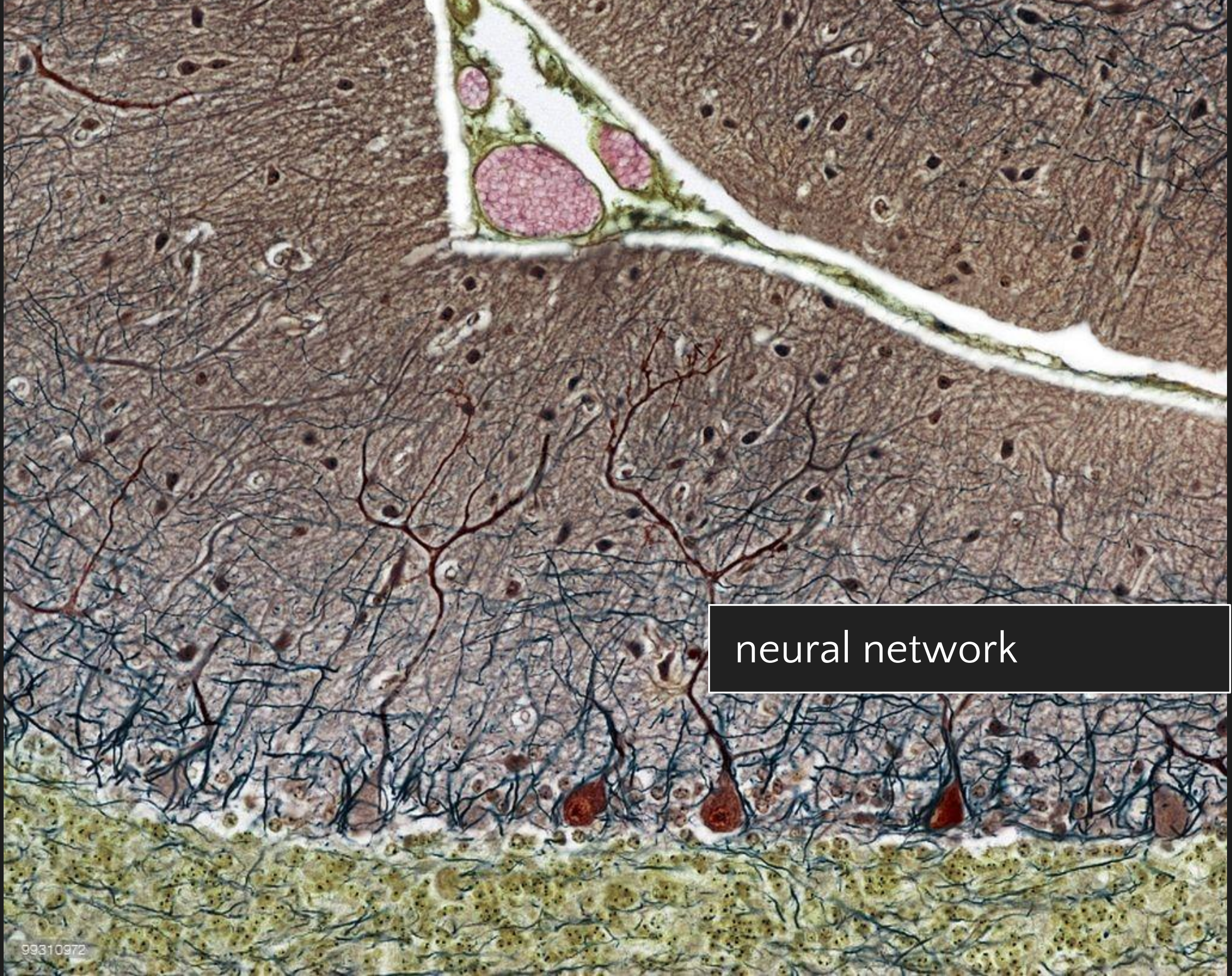
David Hemmings
public domain

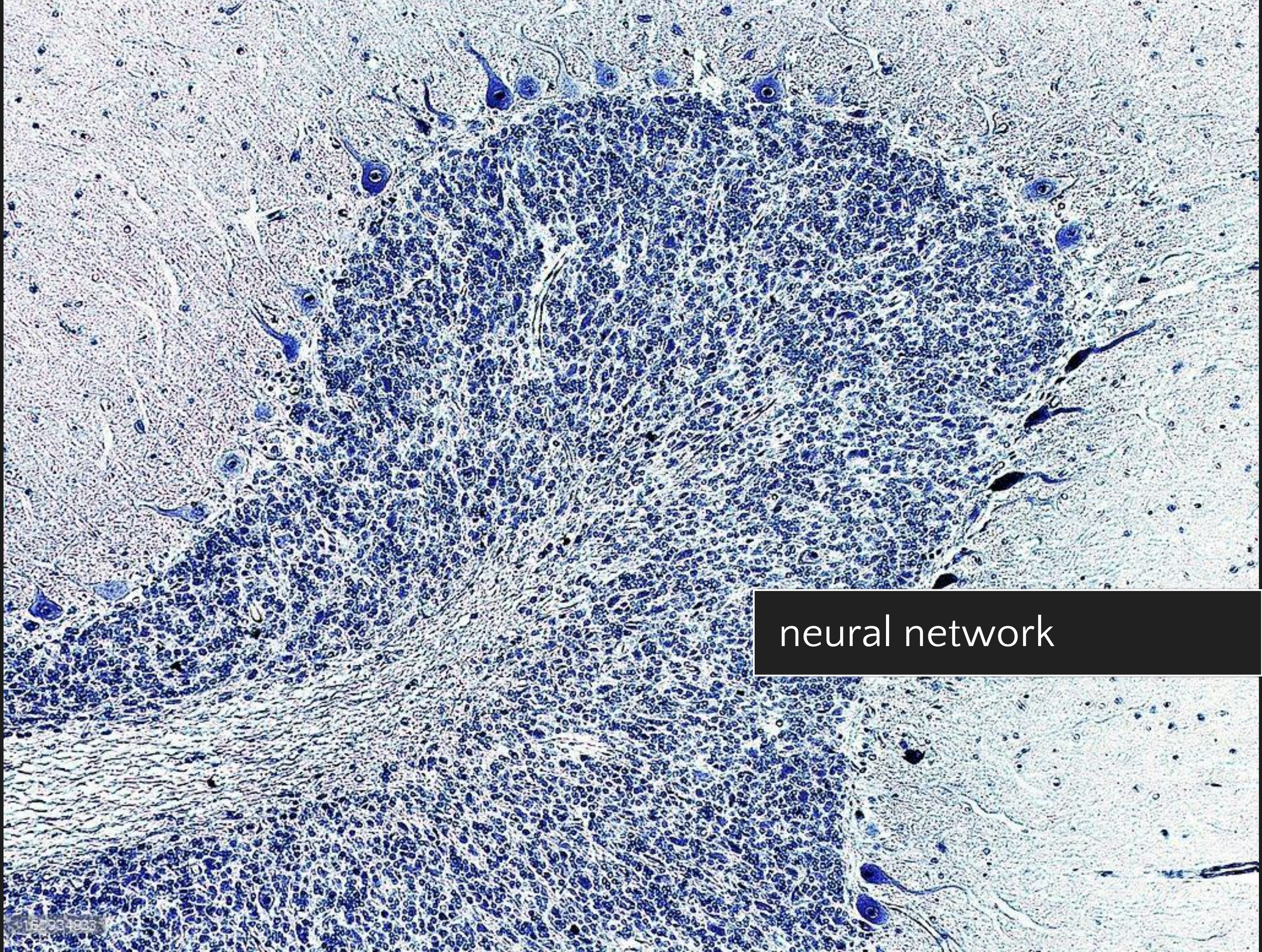
public domain



neuron


neural network


neural network
99310972


neural network

Enough of Biology

Neural Network

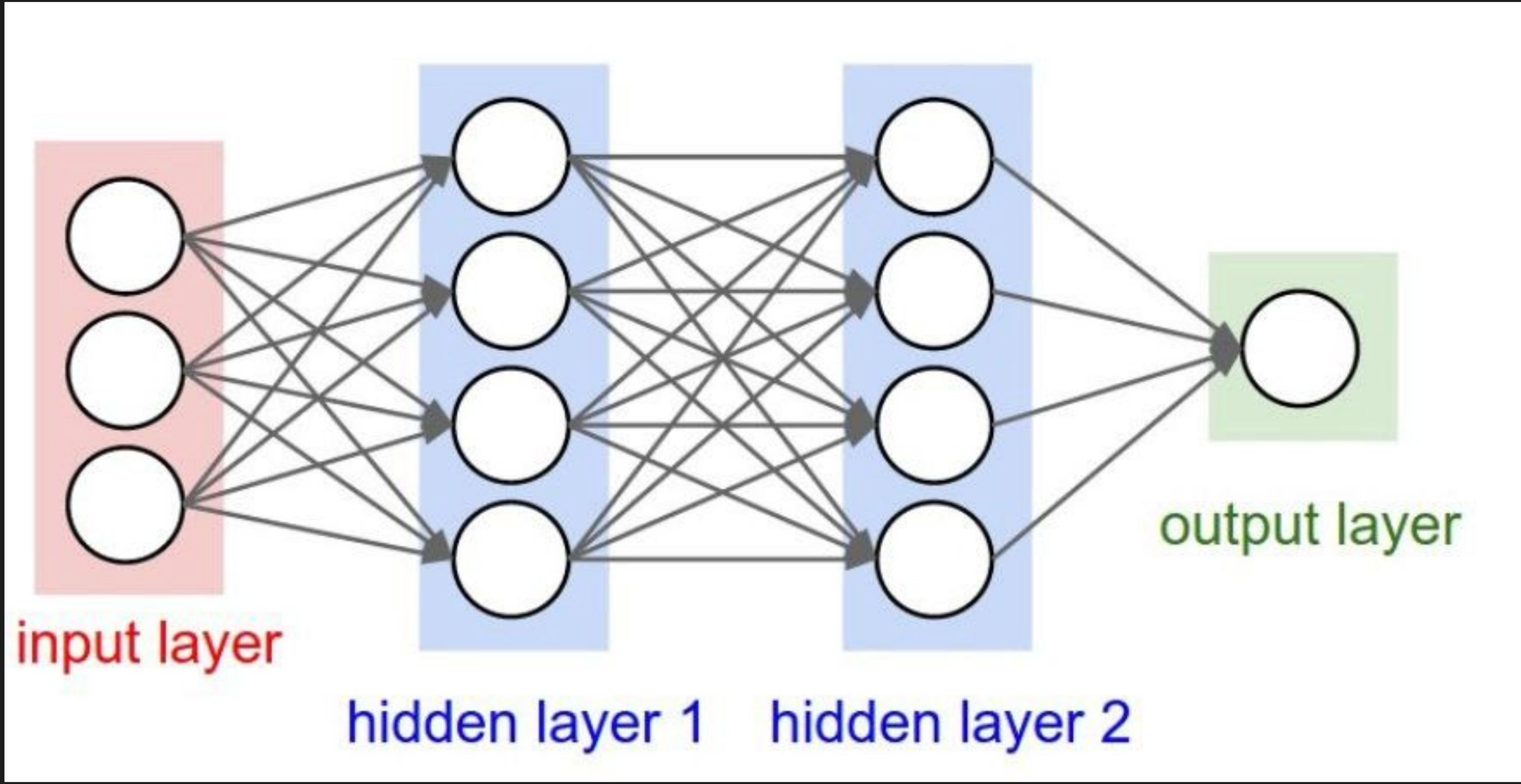
Neural - /ˈnjʊər(ə)l/
adjective
relating to a nerve or the nervous system.
"patterns of neural activity"

Network - /ˈnɛtwəːk/
Noun
a group or system of interconnected people or things.

English? Seriously???

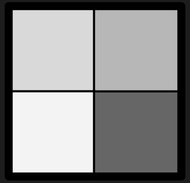
Structure of Neural Network

Structure of neural network

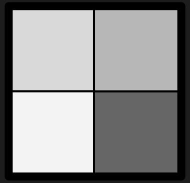


What’s up with all these layers??

A four pixel camera



Categorize images



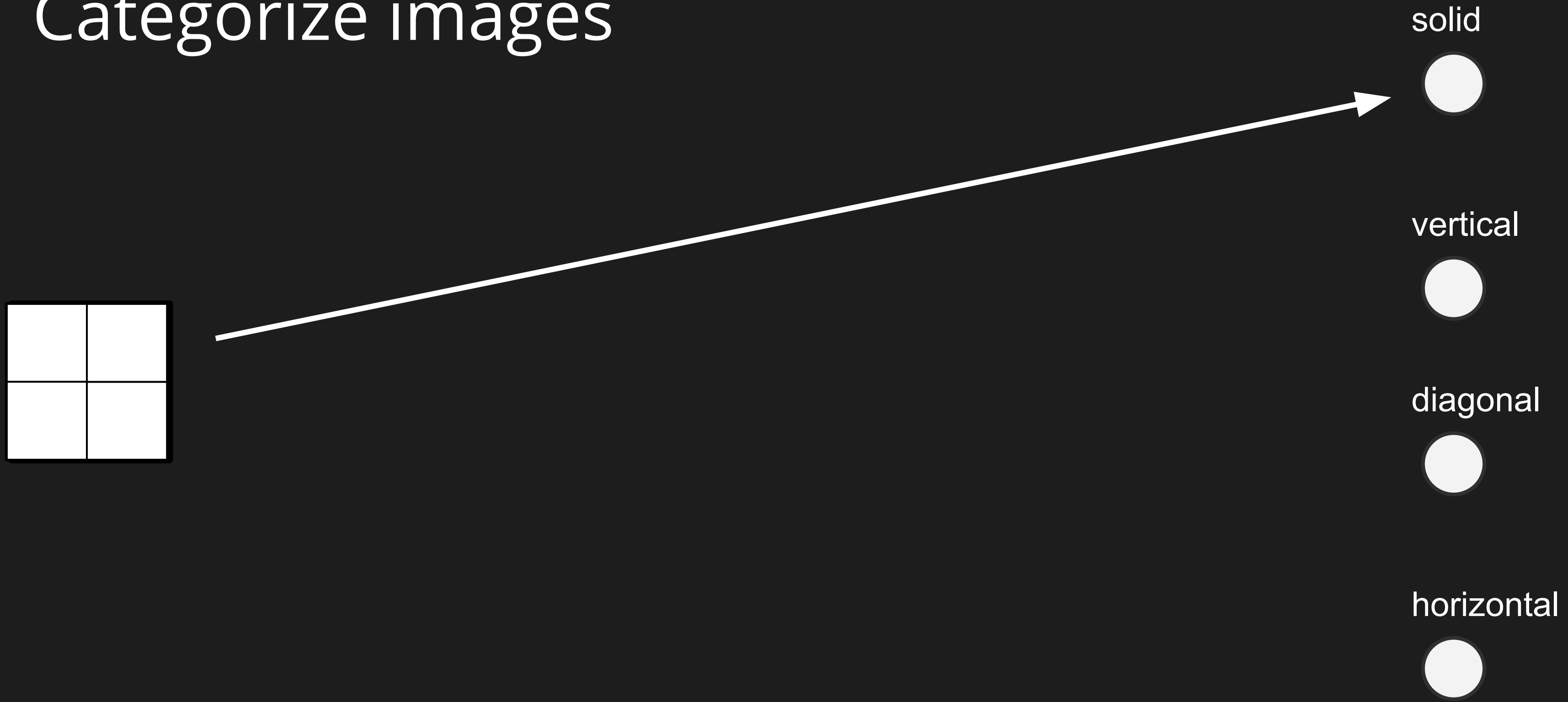
solid

vertical

diagonal

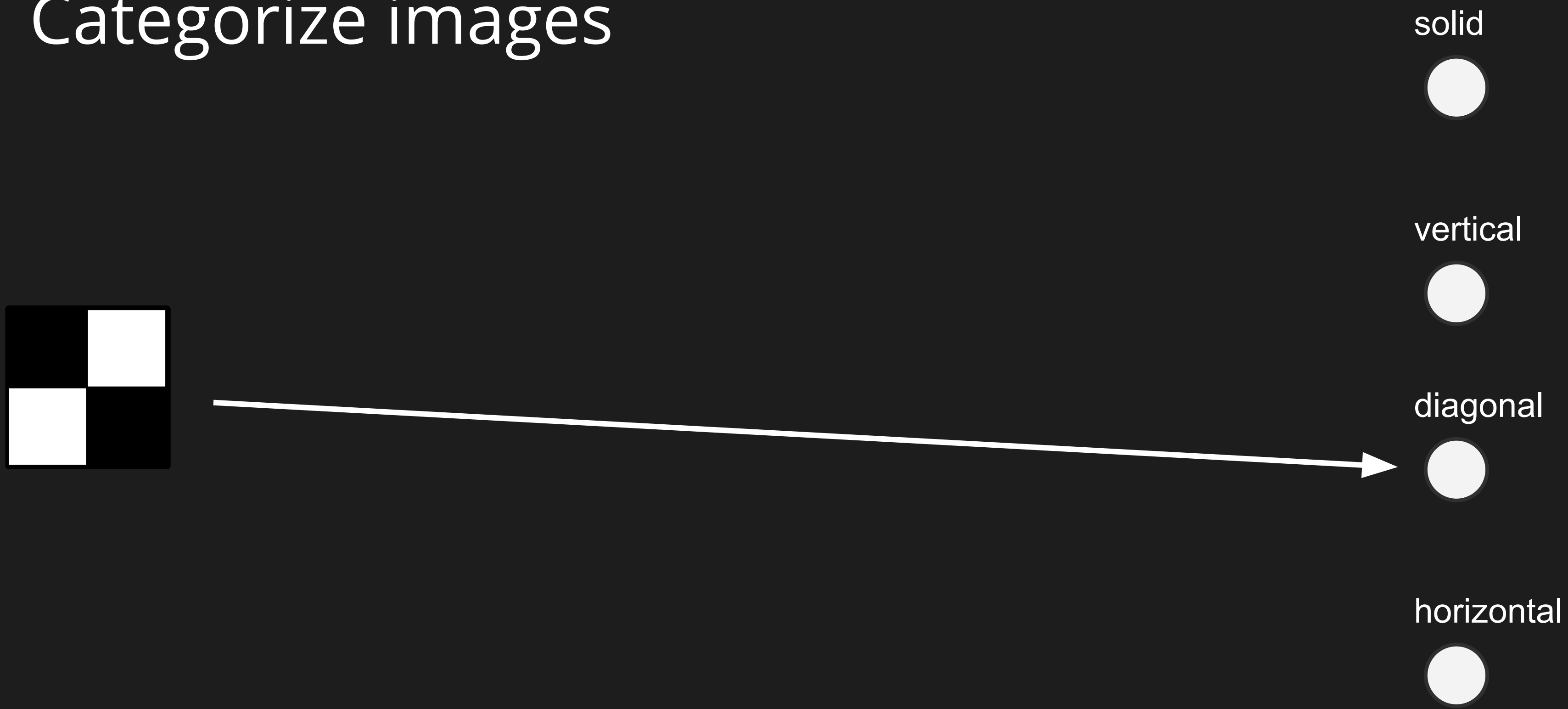
horizontal

Categorize images



Categorize images

Categorize images

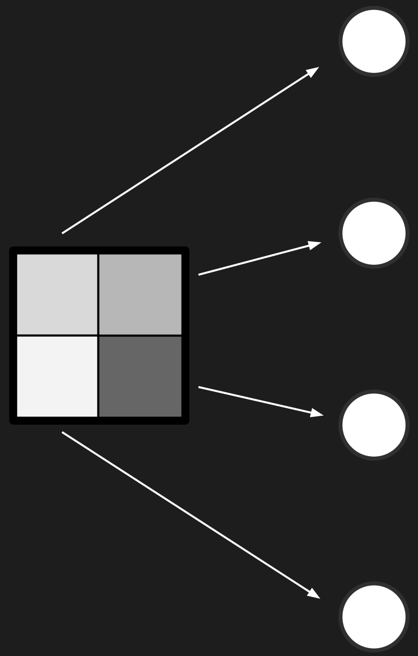


Simple rules can't do it

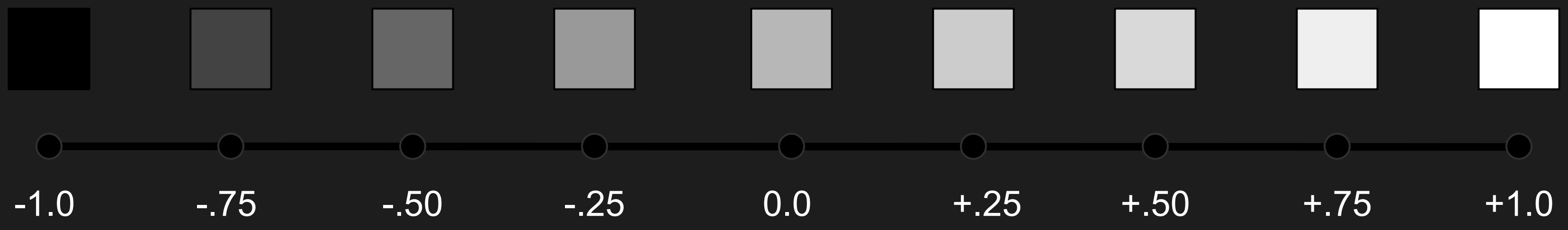
Simple rules can't do it

Neurons, activations, weights, etc.

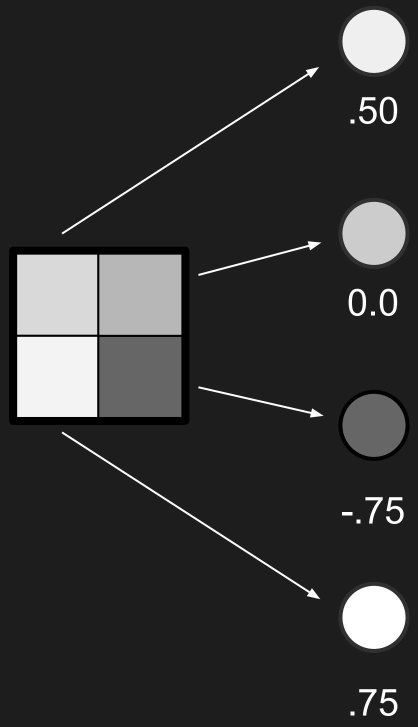
Input neurons



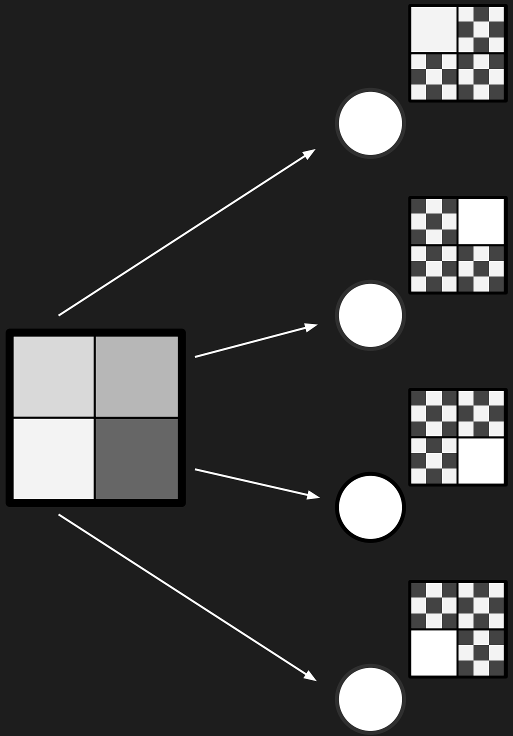
Pixel brightness



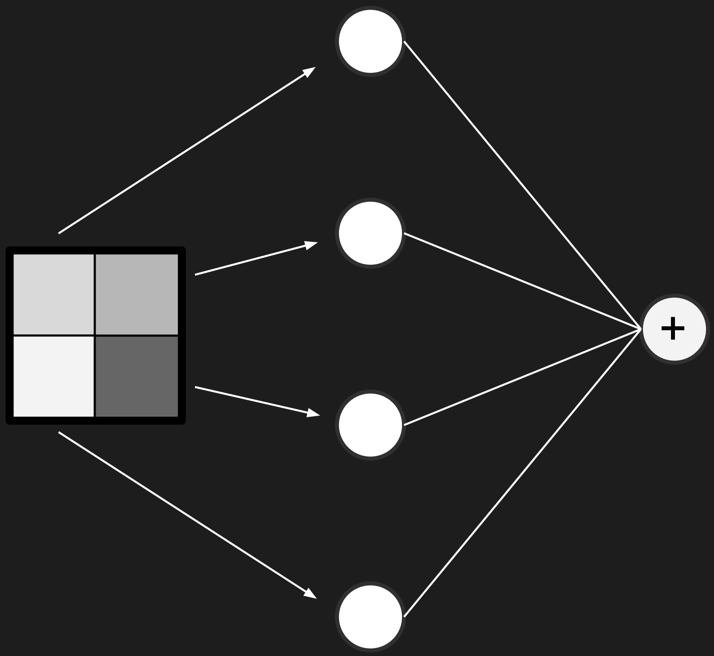
Input vector



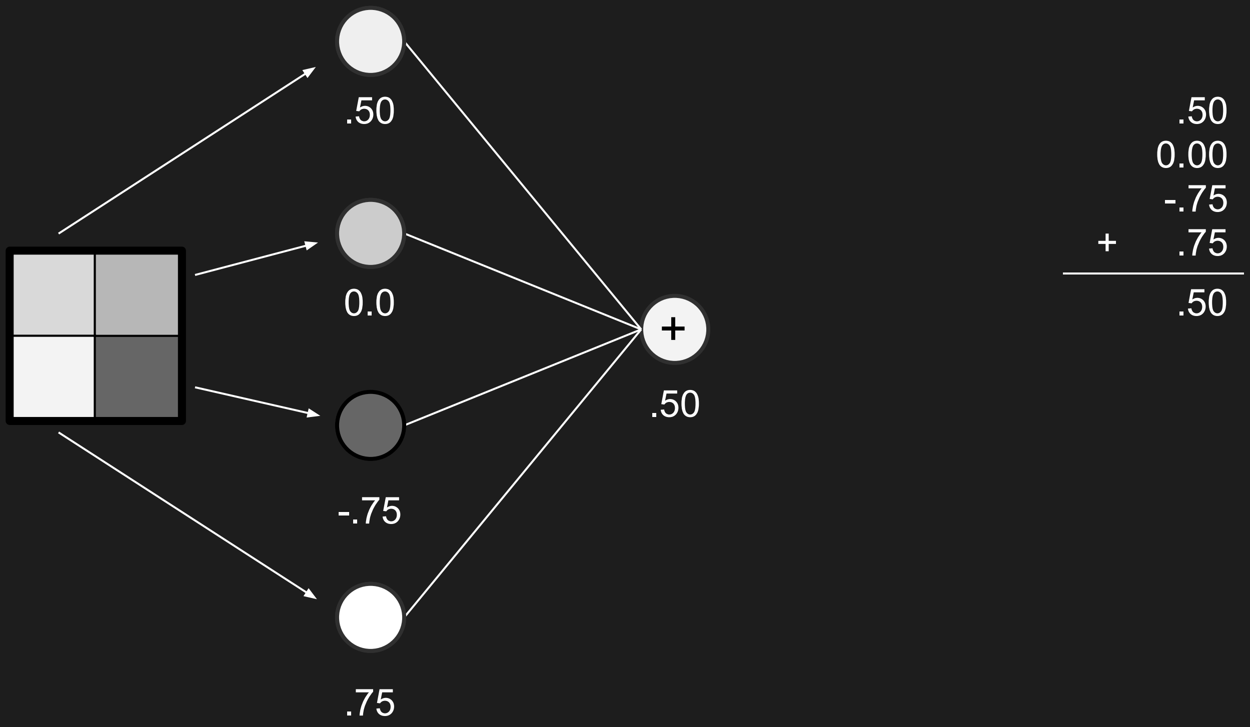
Receptive fields



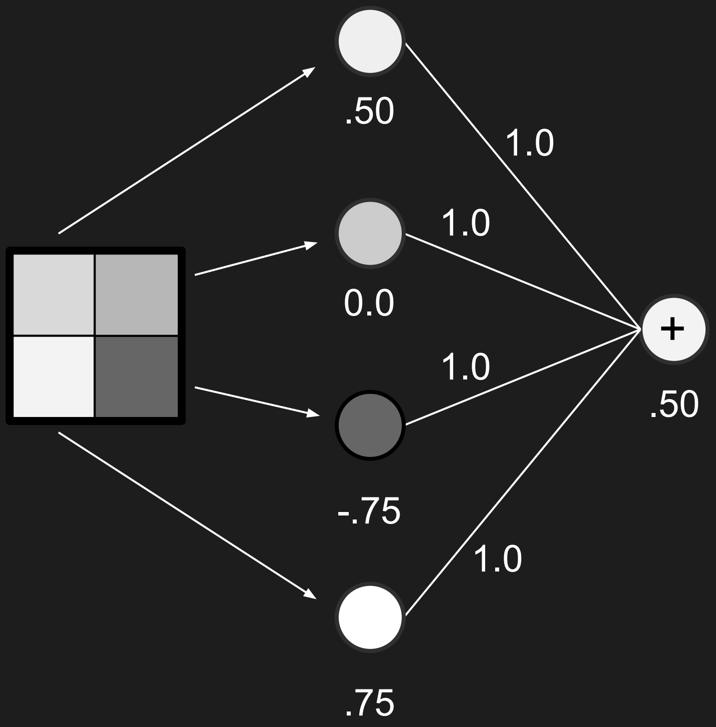
A neuron



Sum all the inputs

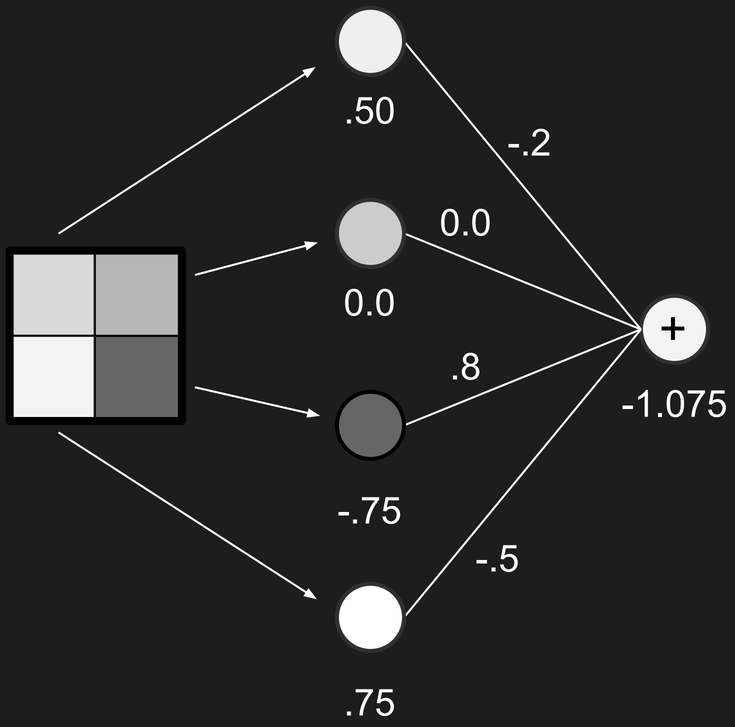


Weights



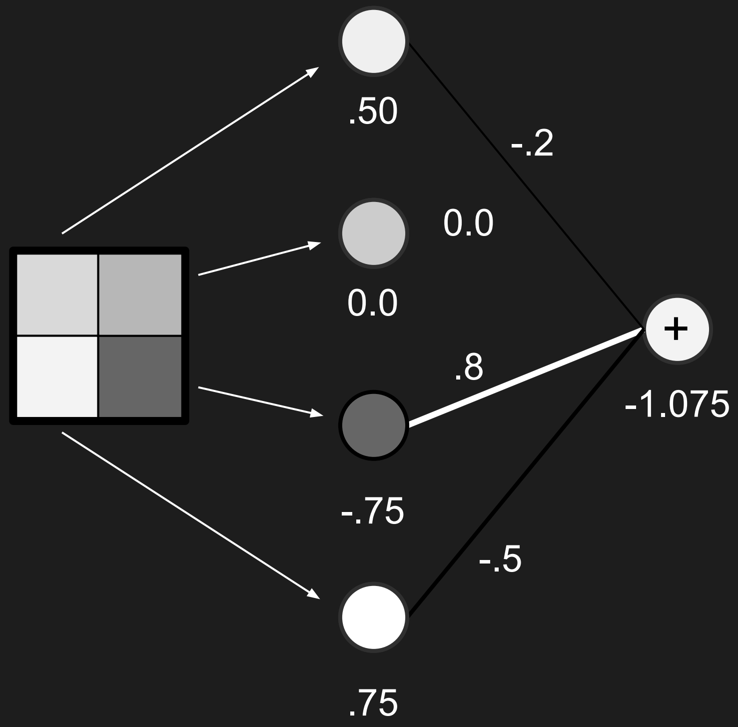
$$
\begin{array}{rrl}
 & .50 & \times\ 1.0 \\
 & 0.00 & \times\ 1.0 \\
 & -.75 & \times\ 1.0 \\
+ & .75 & \times\ 1.0 \\
\hline
 & & .50
\end{array}
$$

Weights



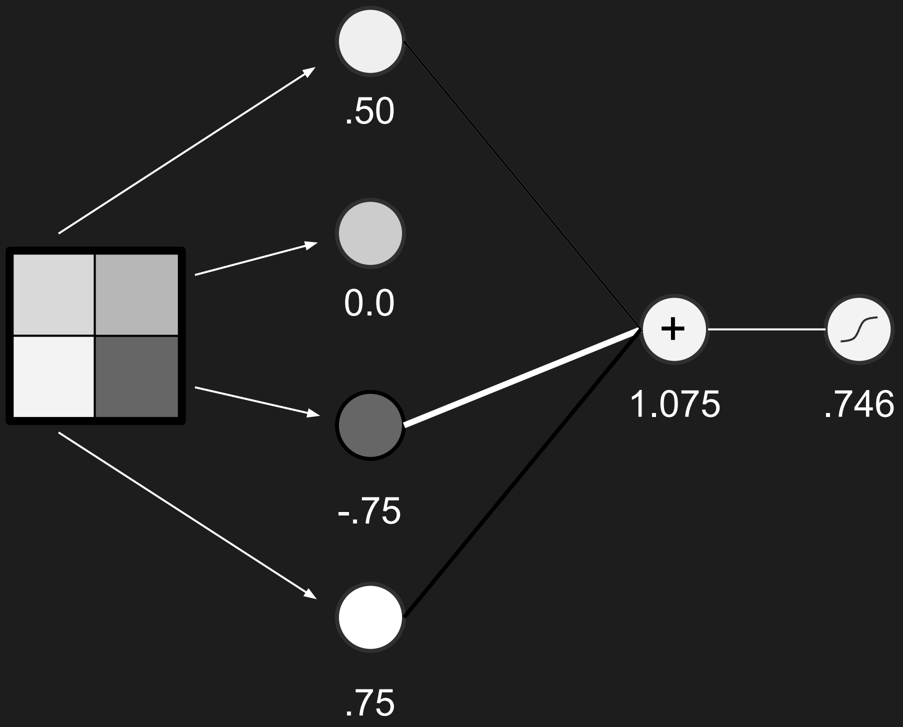
$$
\begin{array}{rrcr}
 & .50 & \times & -.2 \\
 & 0.00 & \times & 0.0 \\
 & -.75 & \times & .8 \\
+ & .75 & \times & -.5 \\
\hline
 & & -1.075 &
\end{array}
$$

Weights

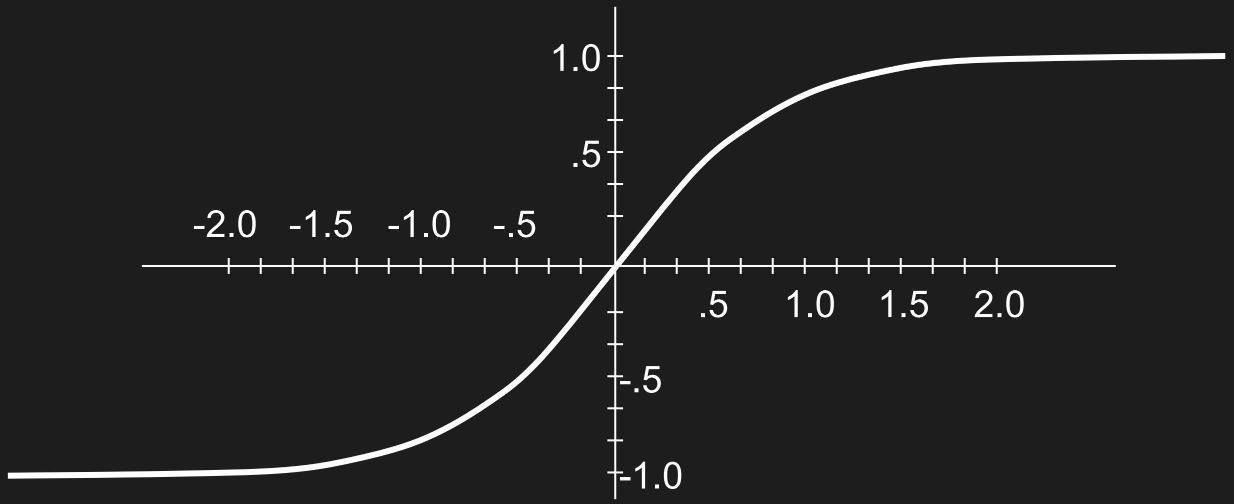


$$
\begin{array}{rrcr}
 & .50 & \times & -.2 \\
 & 0.00 & \times & 0.0 \\
 & -.75 & \times & .8 \\
+ & .75 & \times & -.5 \\
\hline
 & & -1.075 &
\end{array}
$$

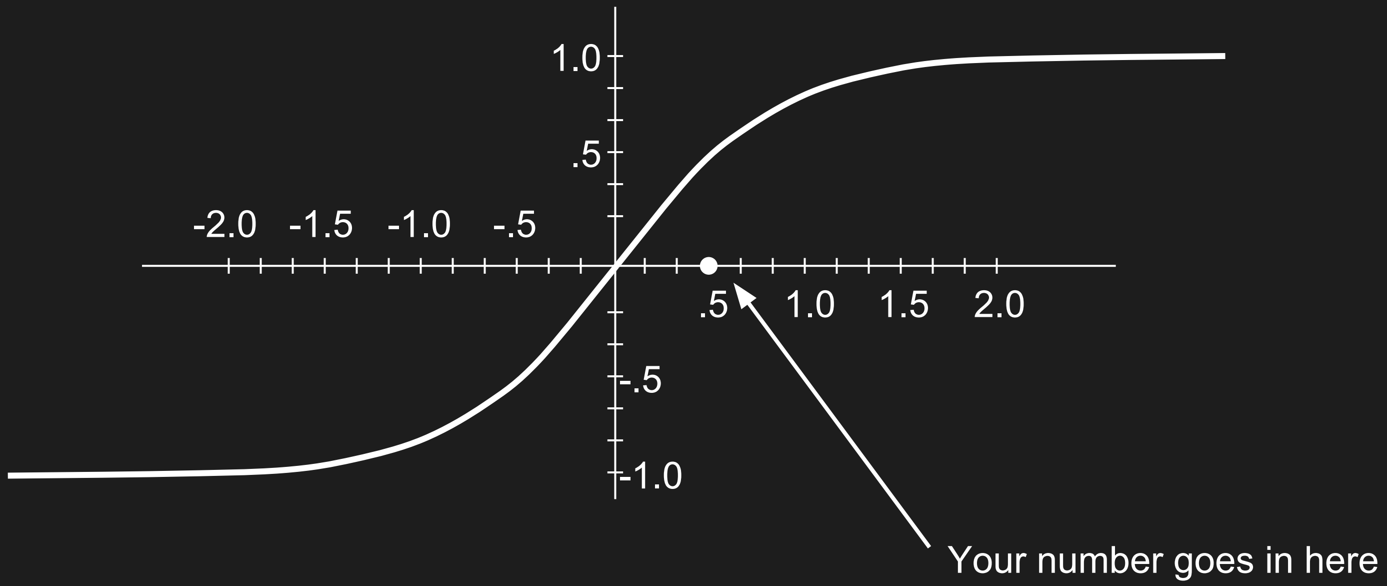
Squash the result



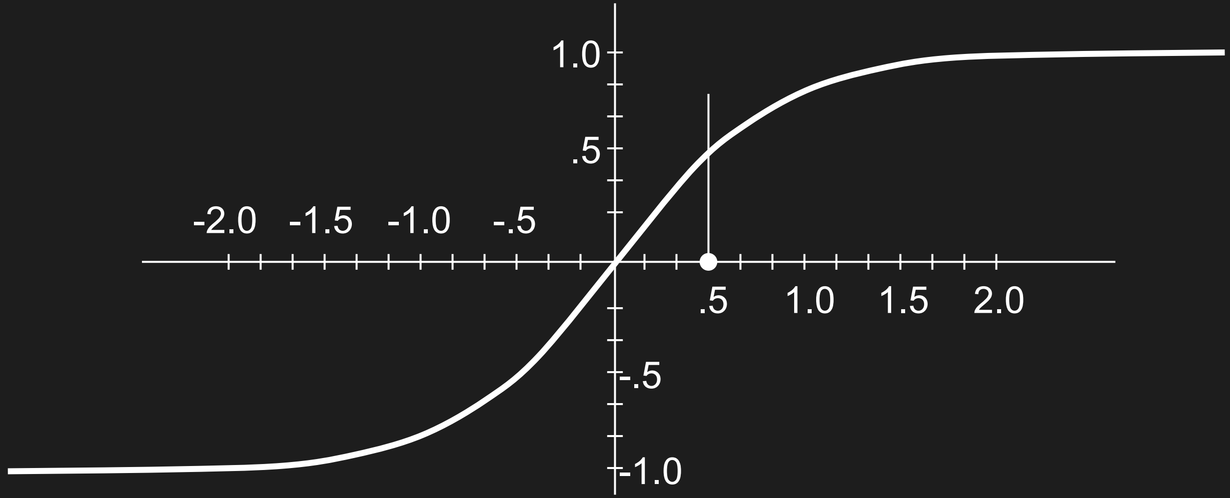
Tanh squashing function



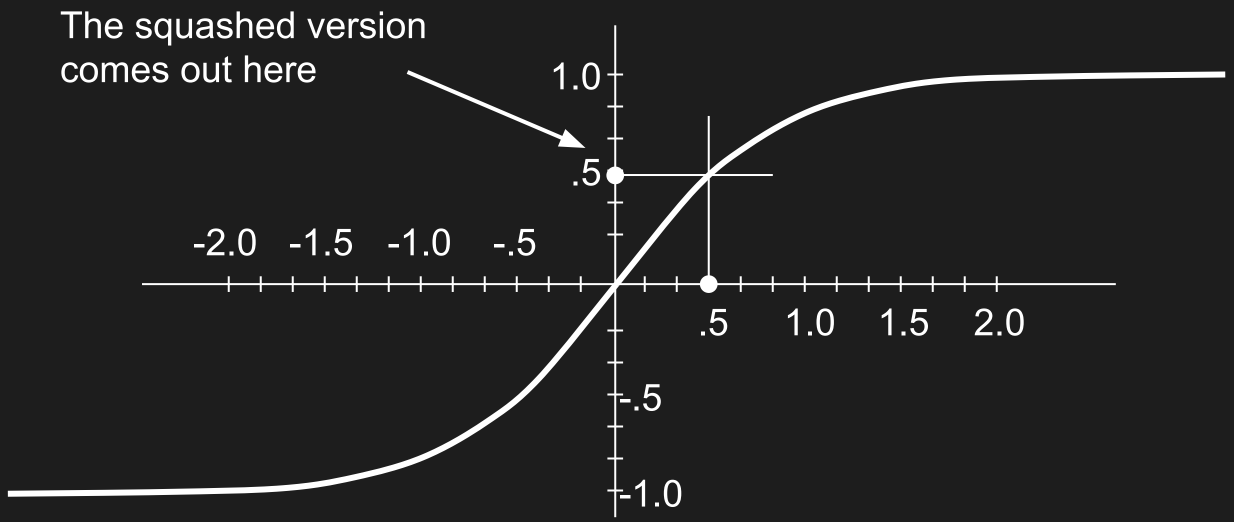
Tanh squashing function



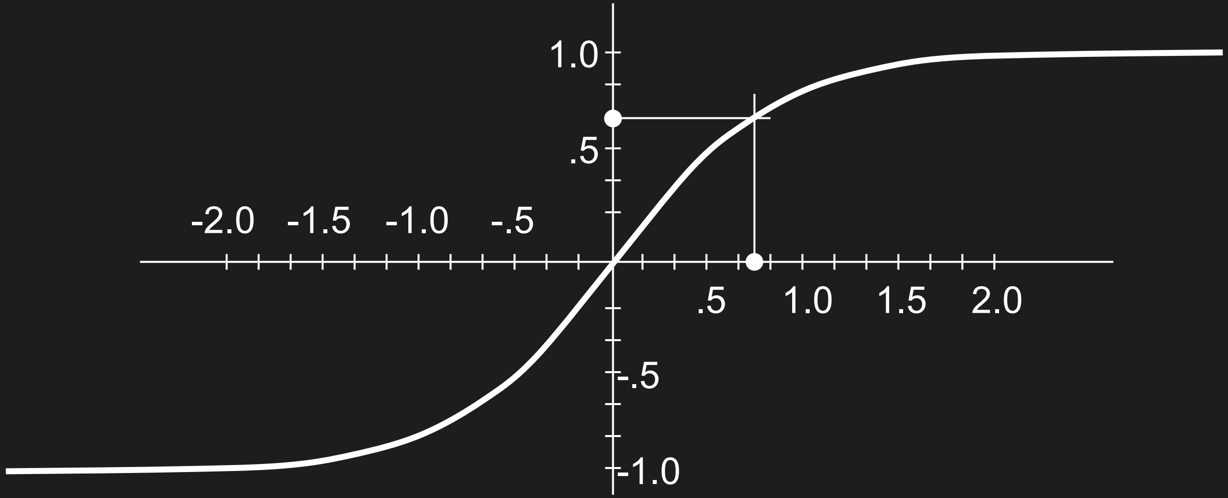
Tanh squashing function



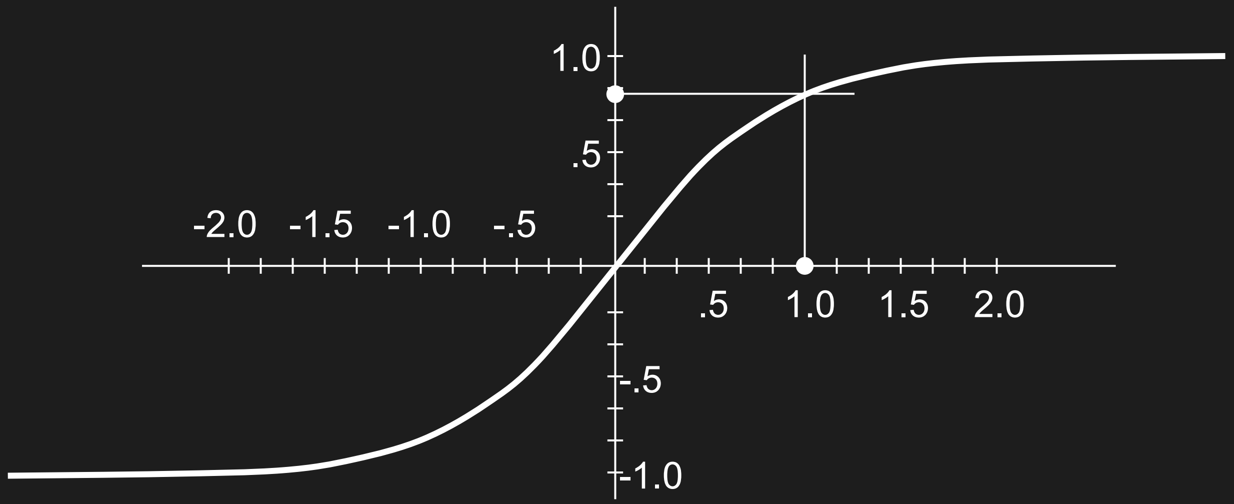
Tanh squashing function



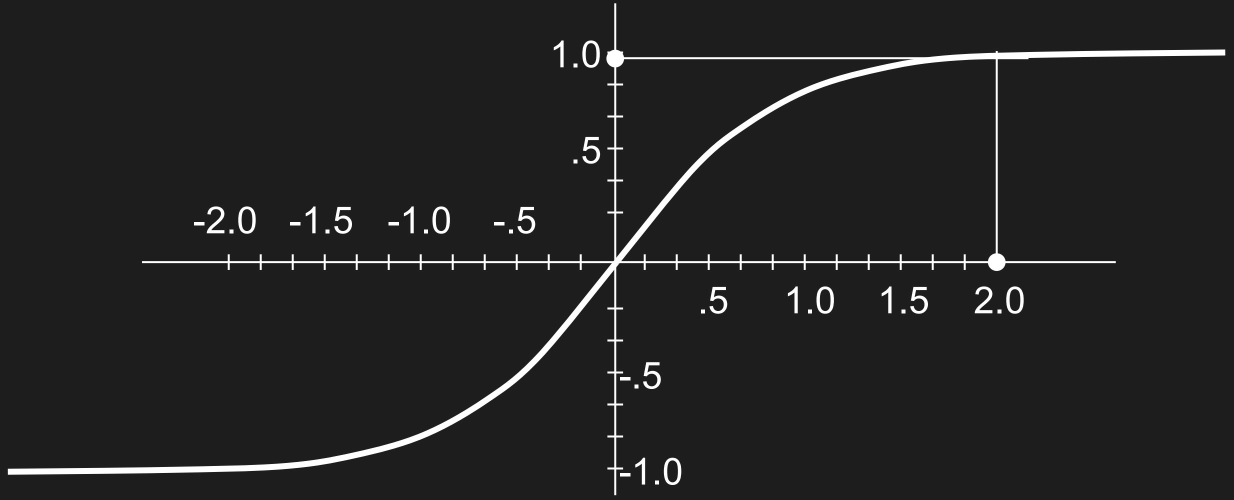
Tanh squashing function



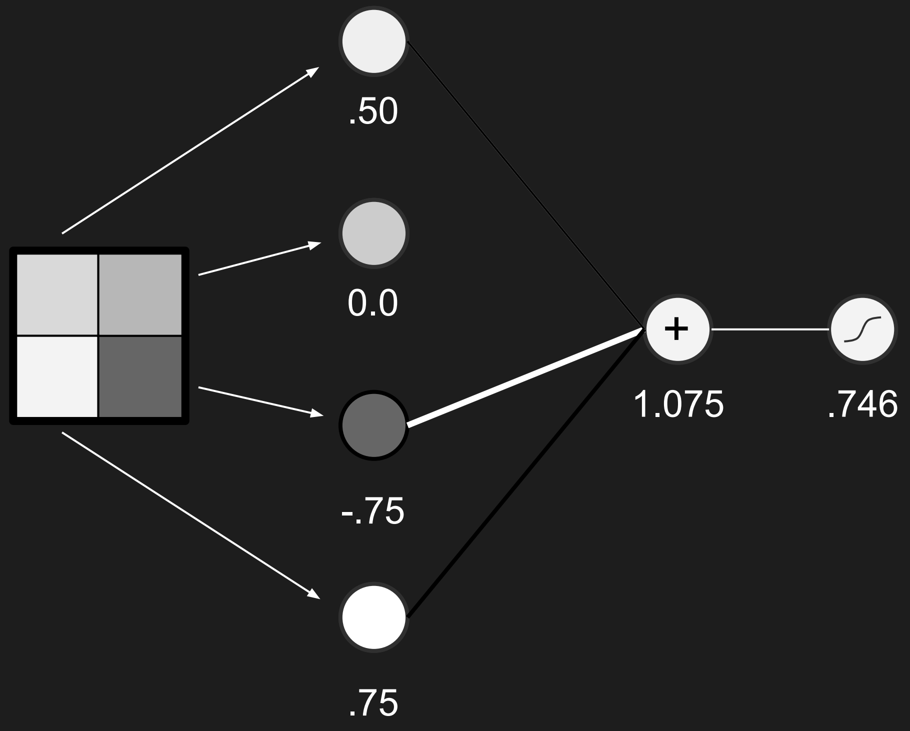
Tanh squashing function



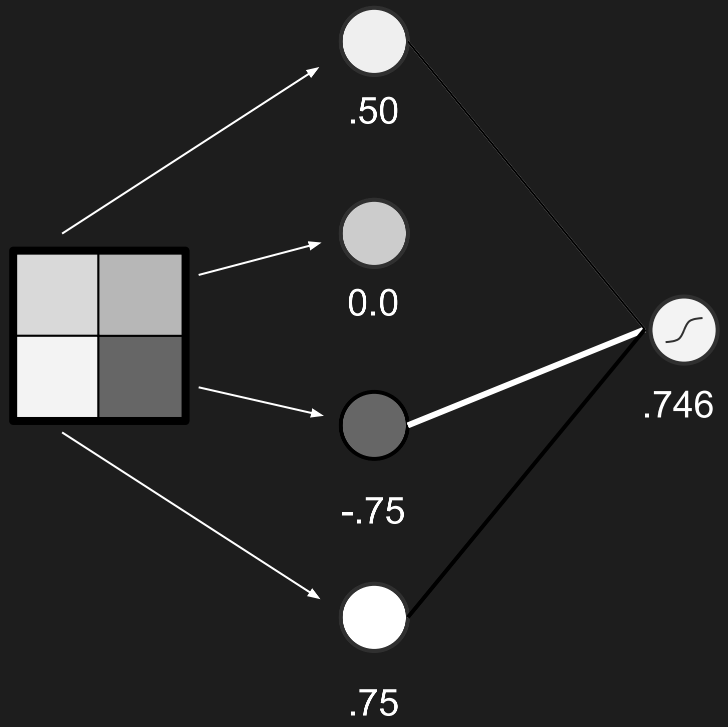
No matter what you start with, the answer stays between -1 and 1.



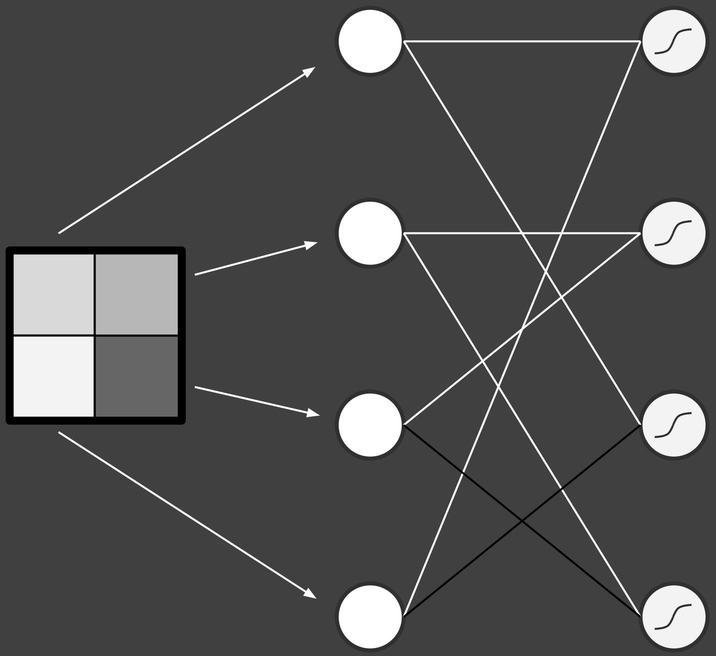
Squash the result



Weighted sum-and-squash neuron

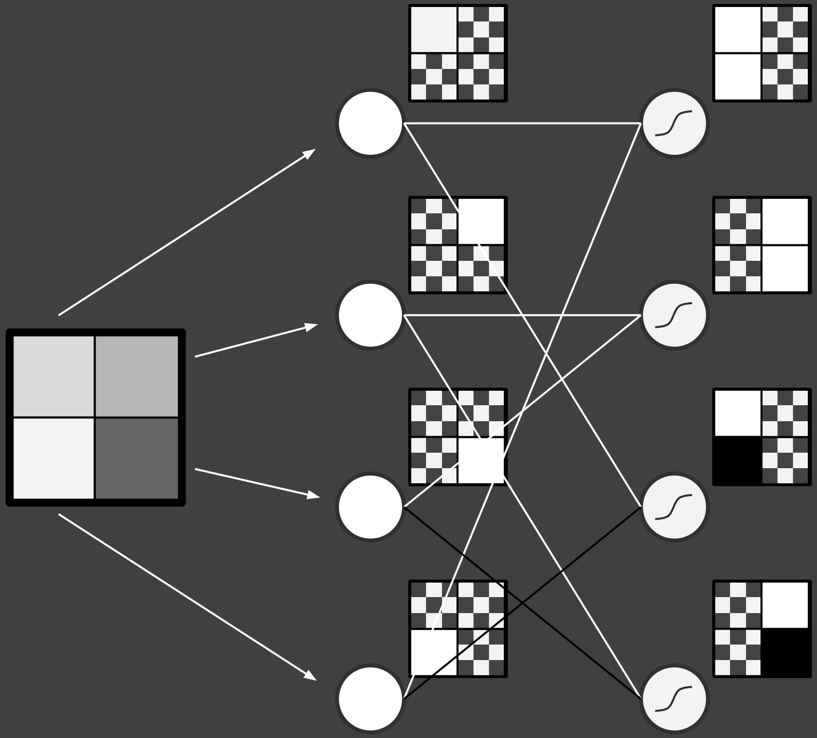


Make lots of neurons, identical except for weights

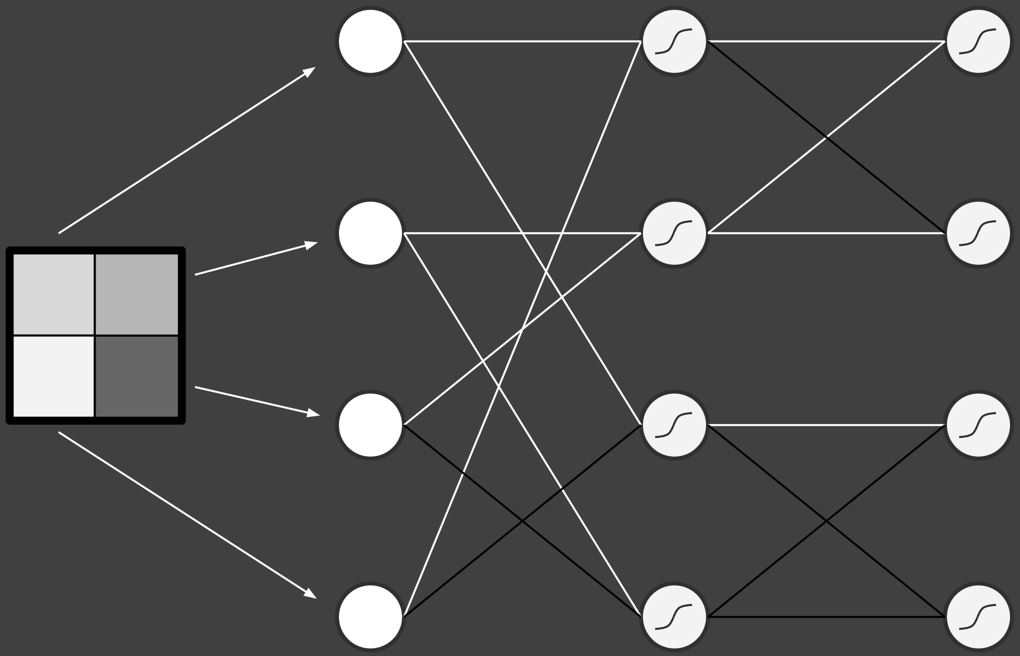


To keep our picture clear, weights will either be
1.0 (white)
-1.0 (black) or
0.0 (missing)

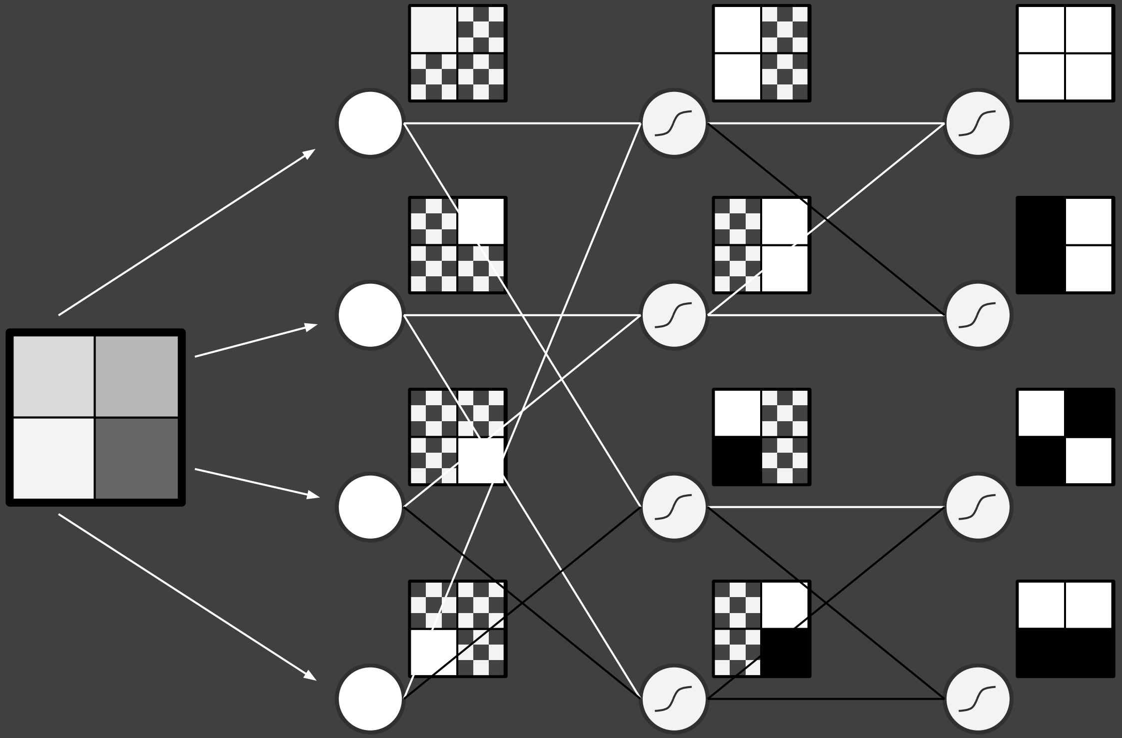
Receptive fields get more complex



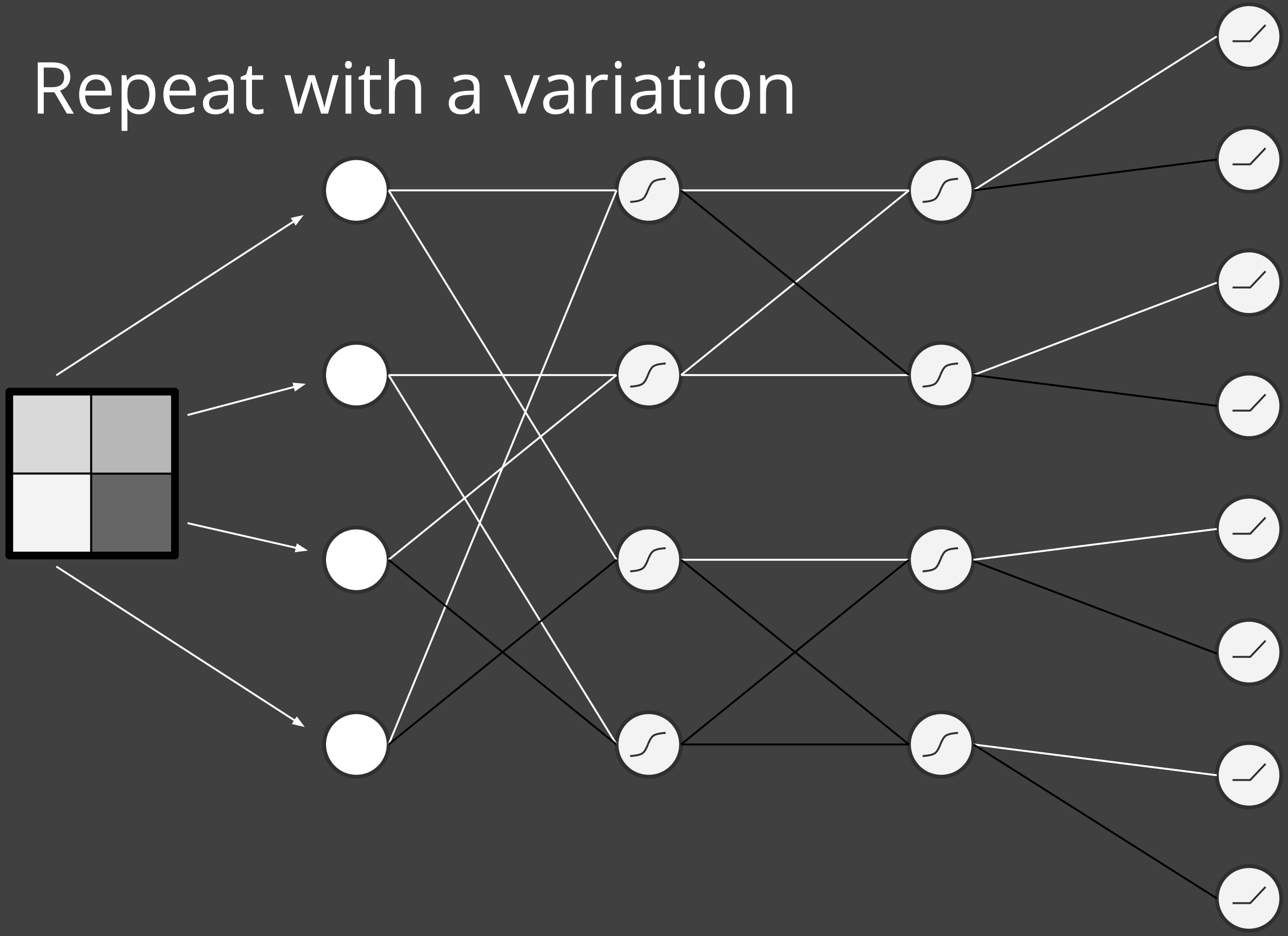
Repeat for additional layers

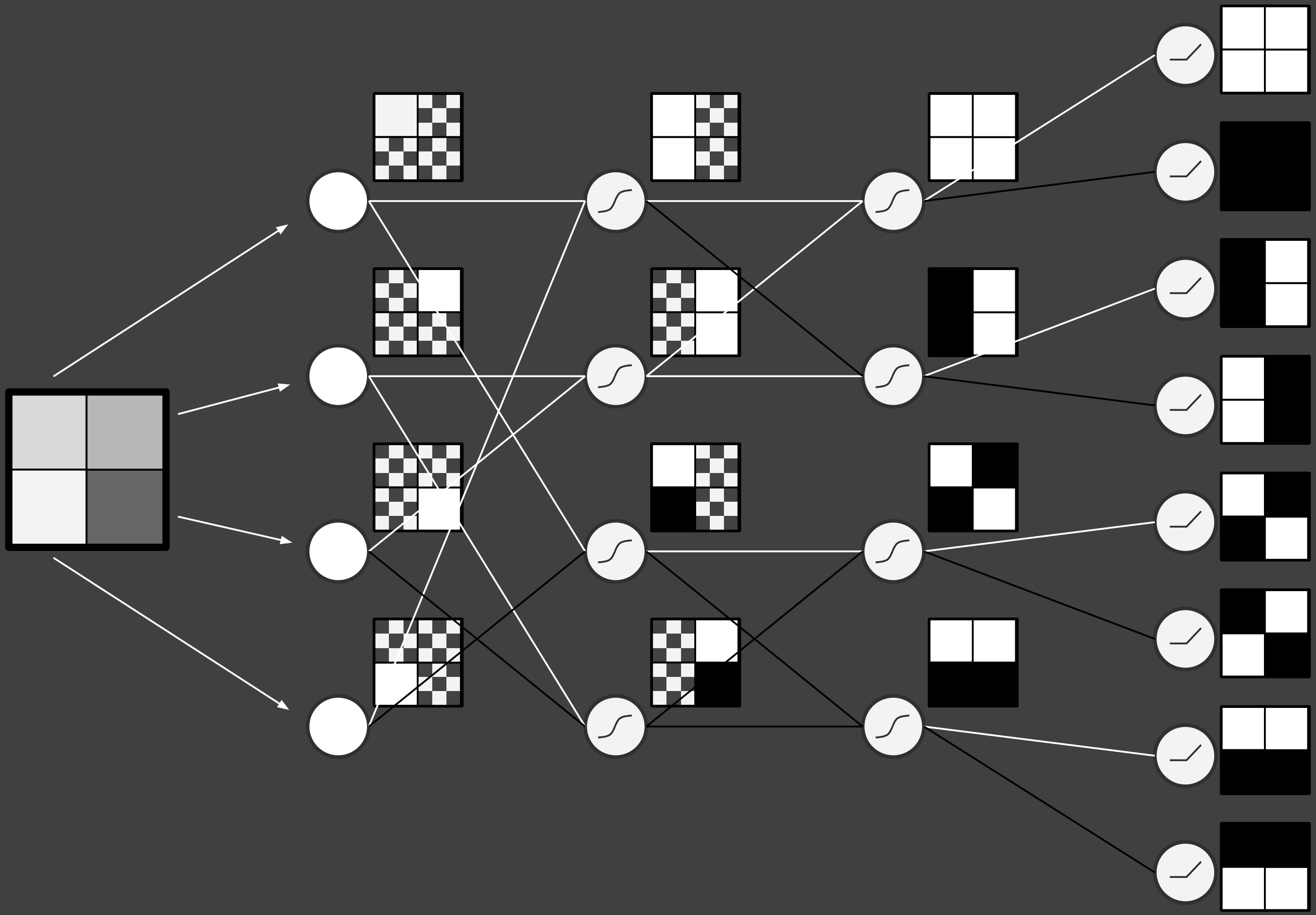


Receptive fields get still more complex

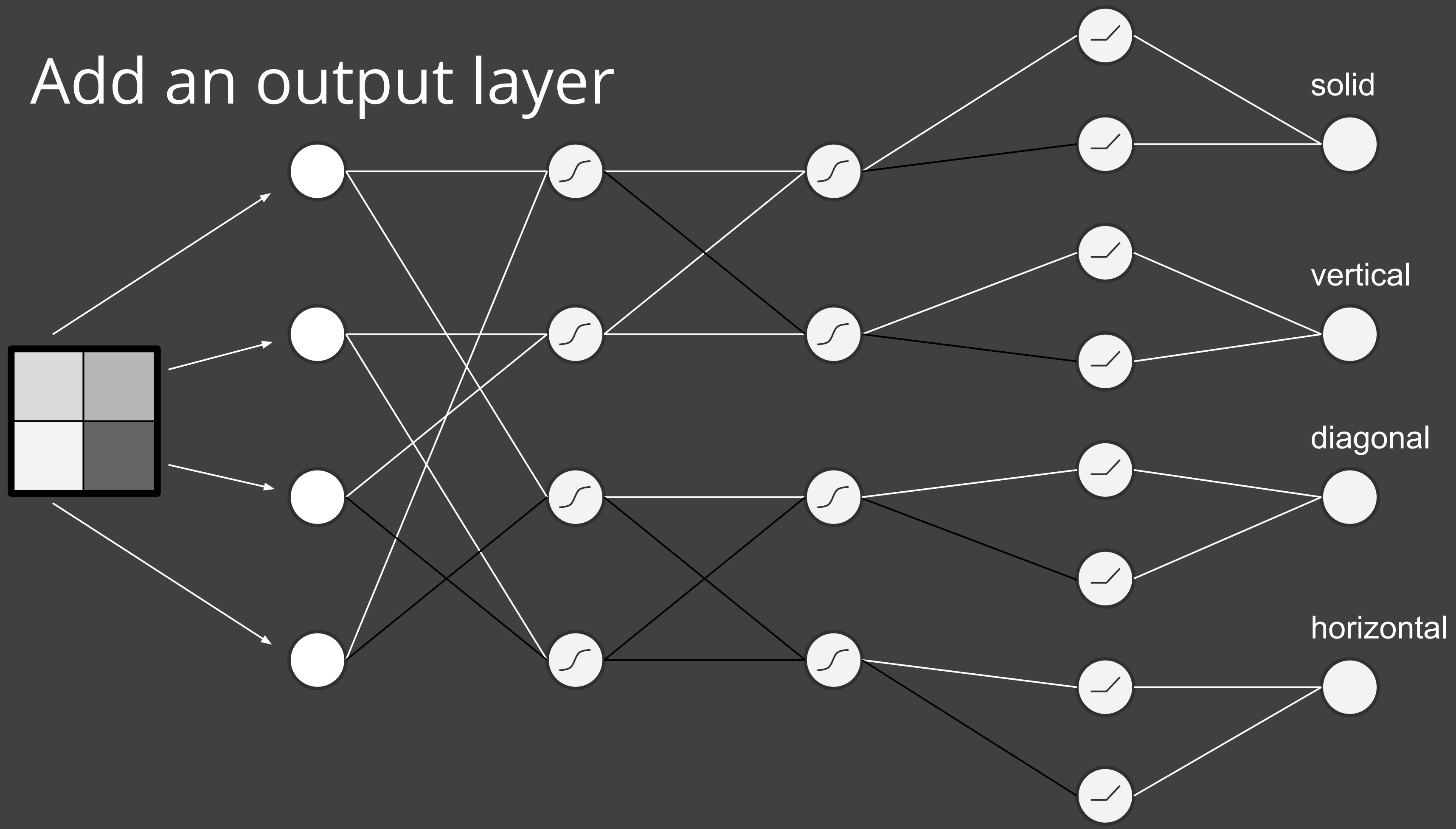


Repeat with a variation

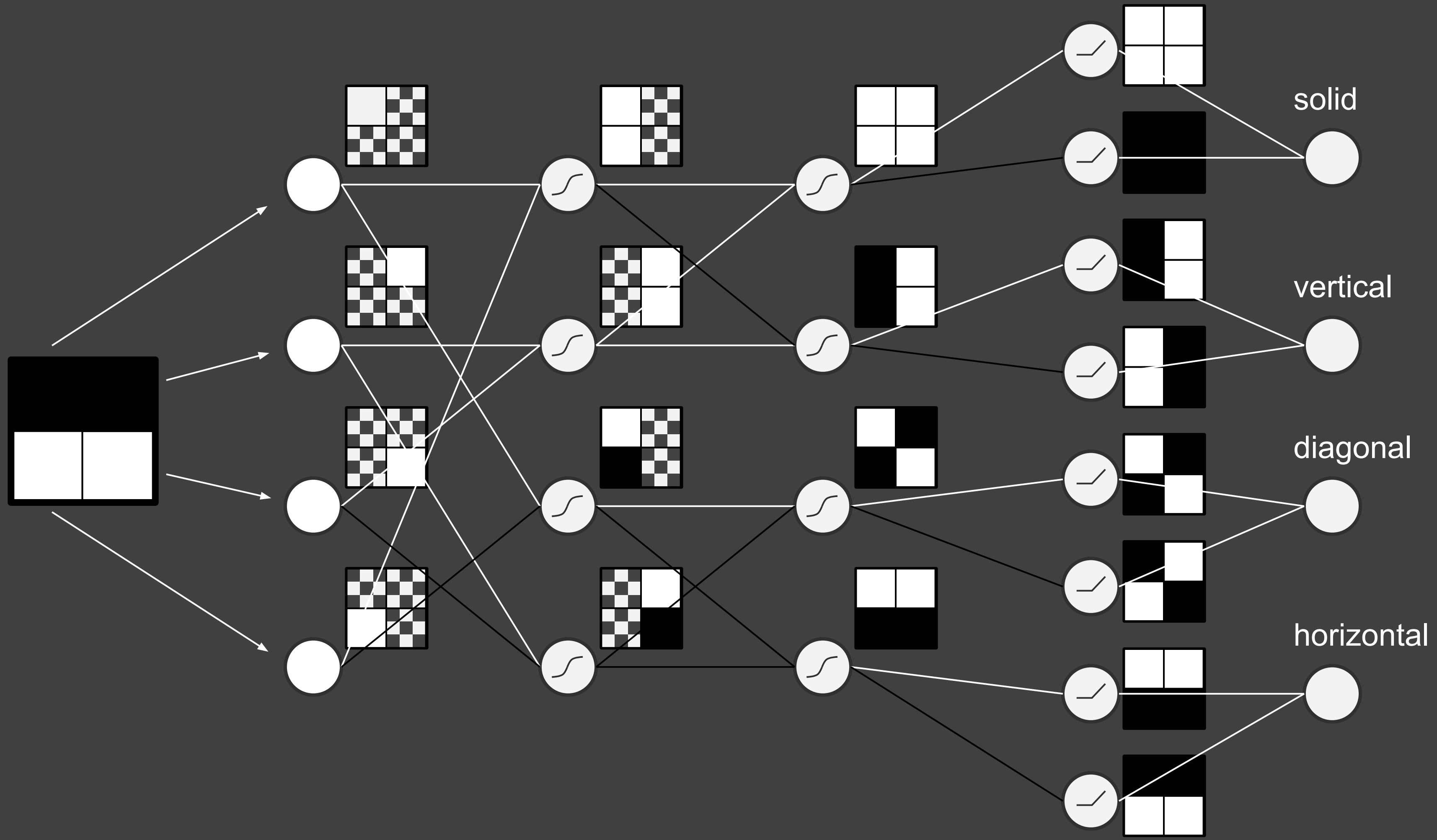




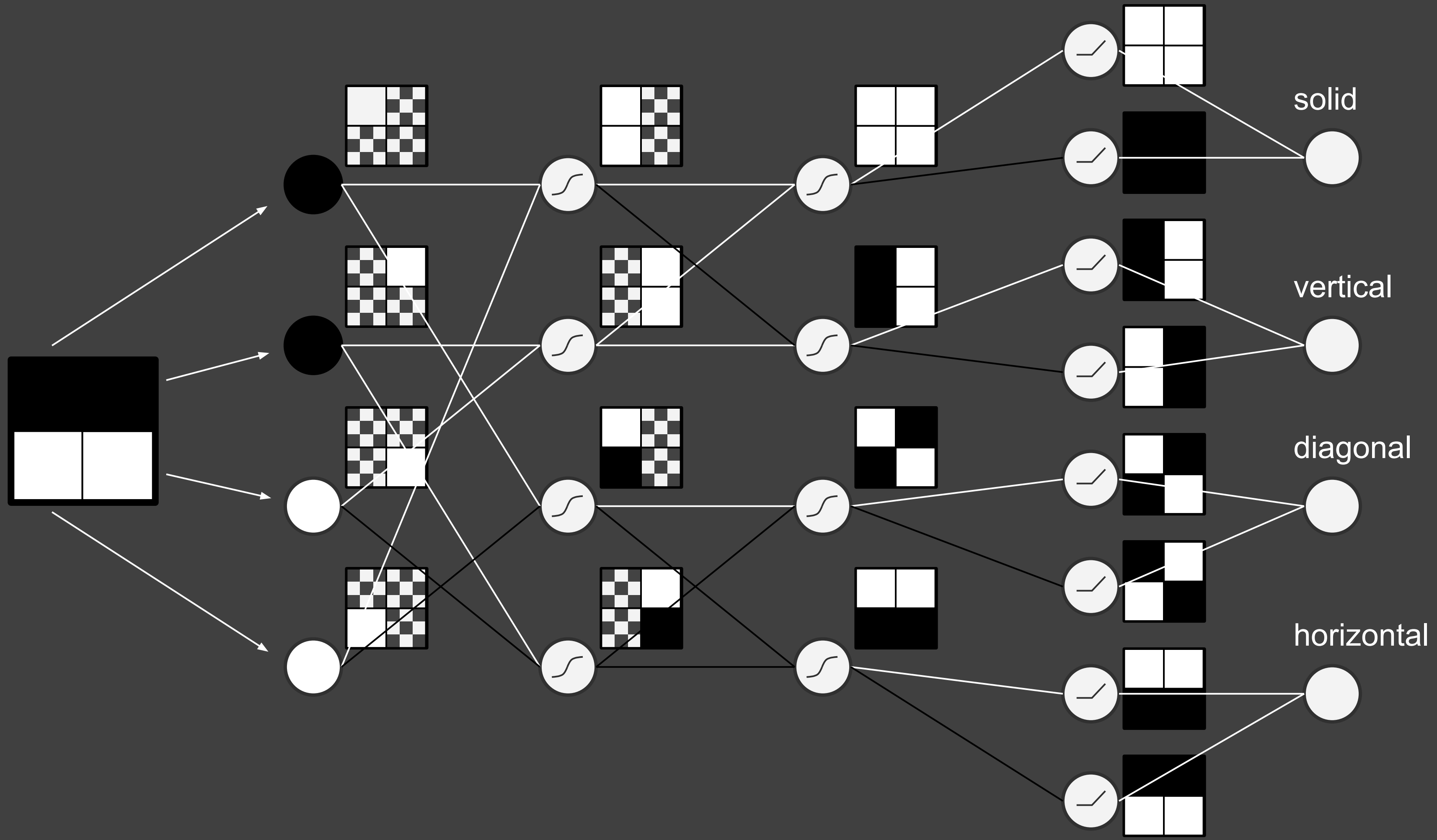
Add an output layer

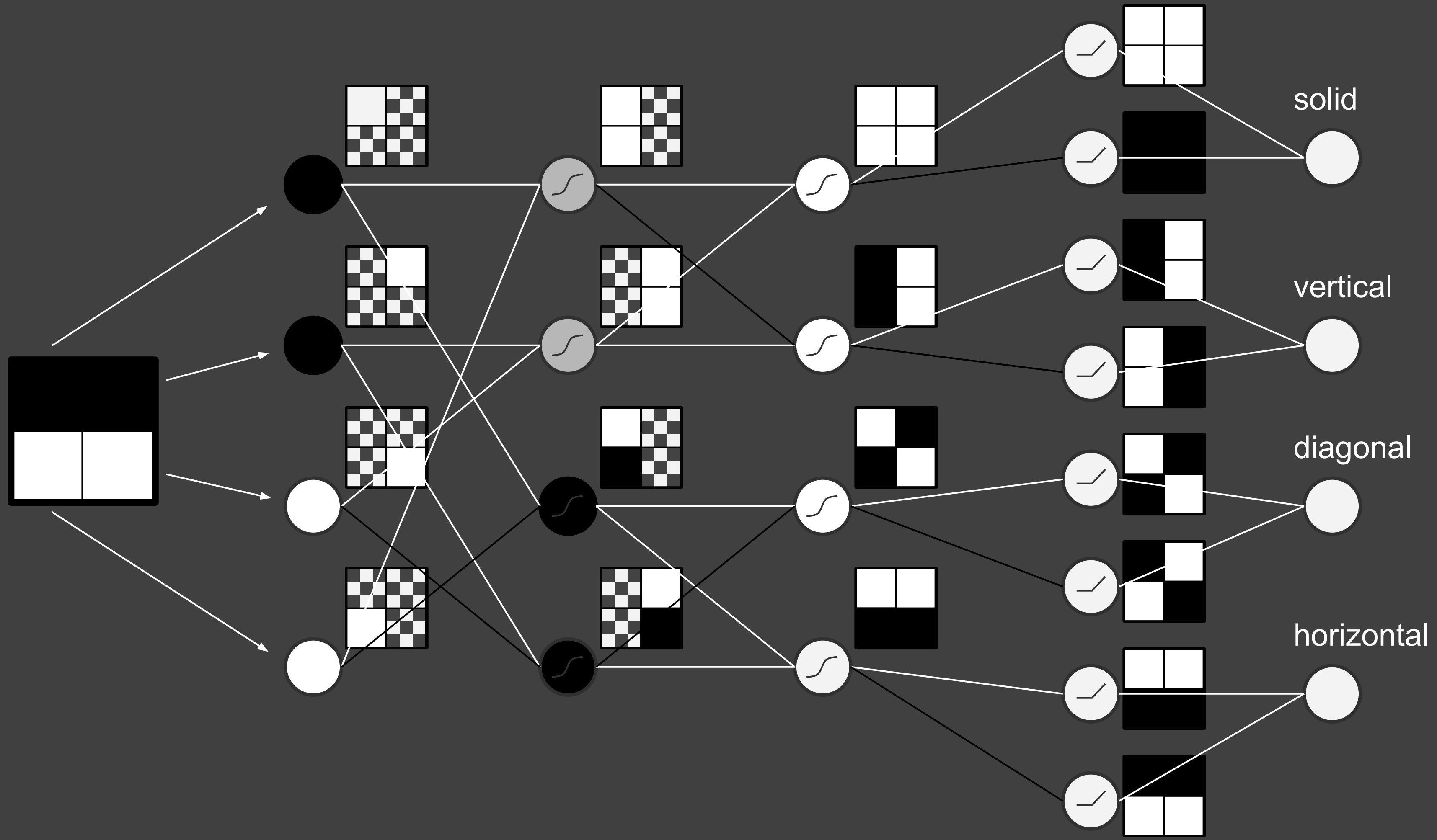


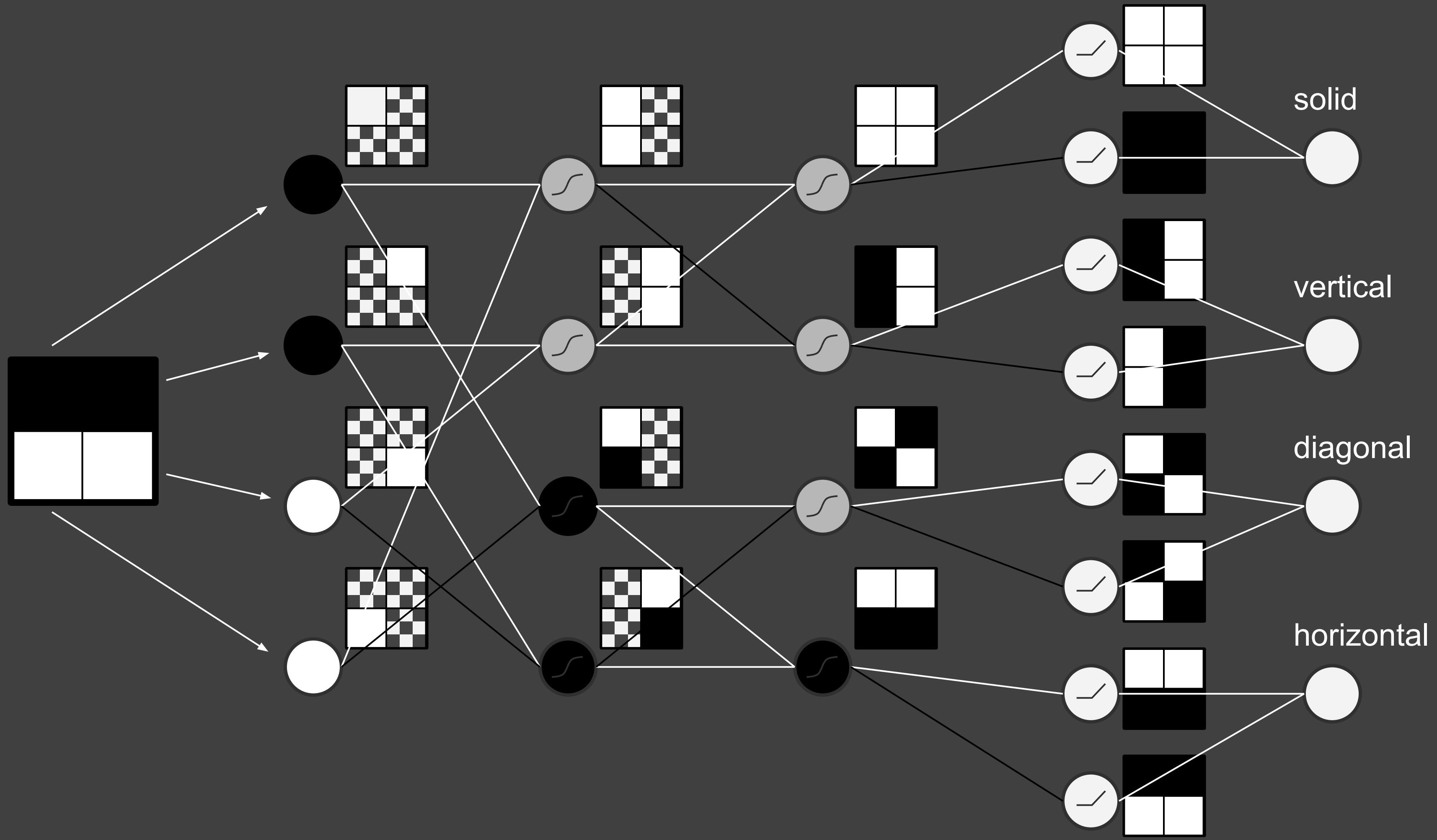
Forward propagation

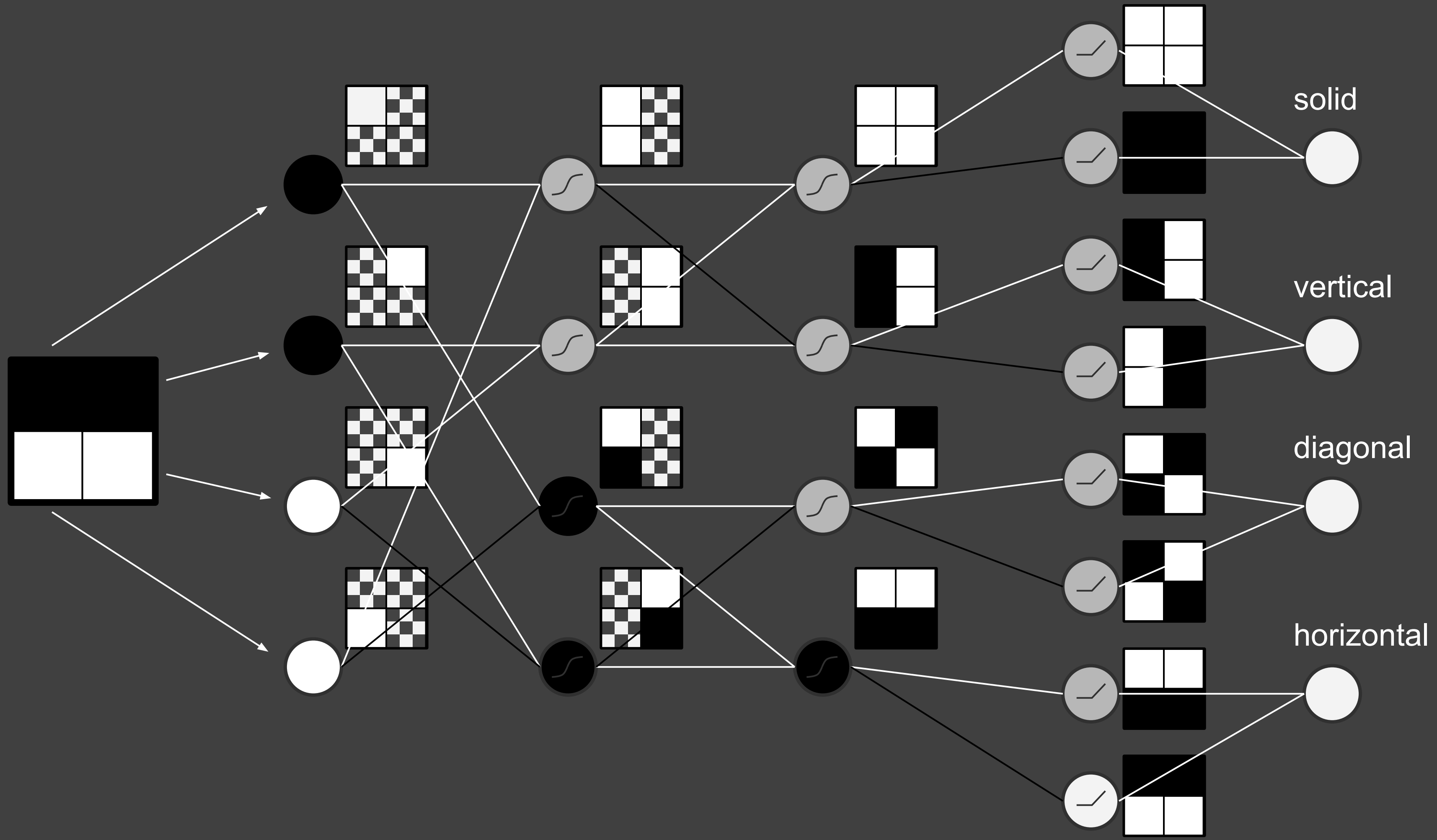


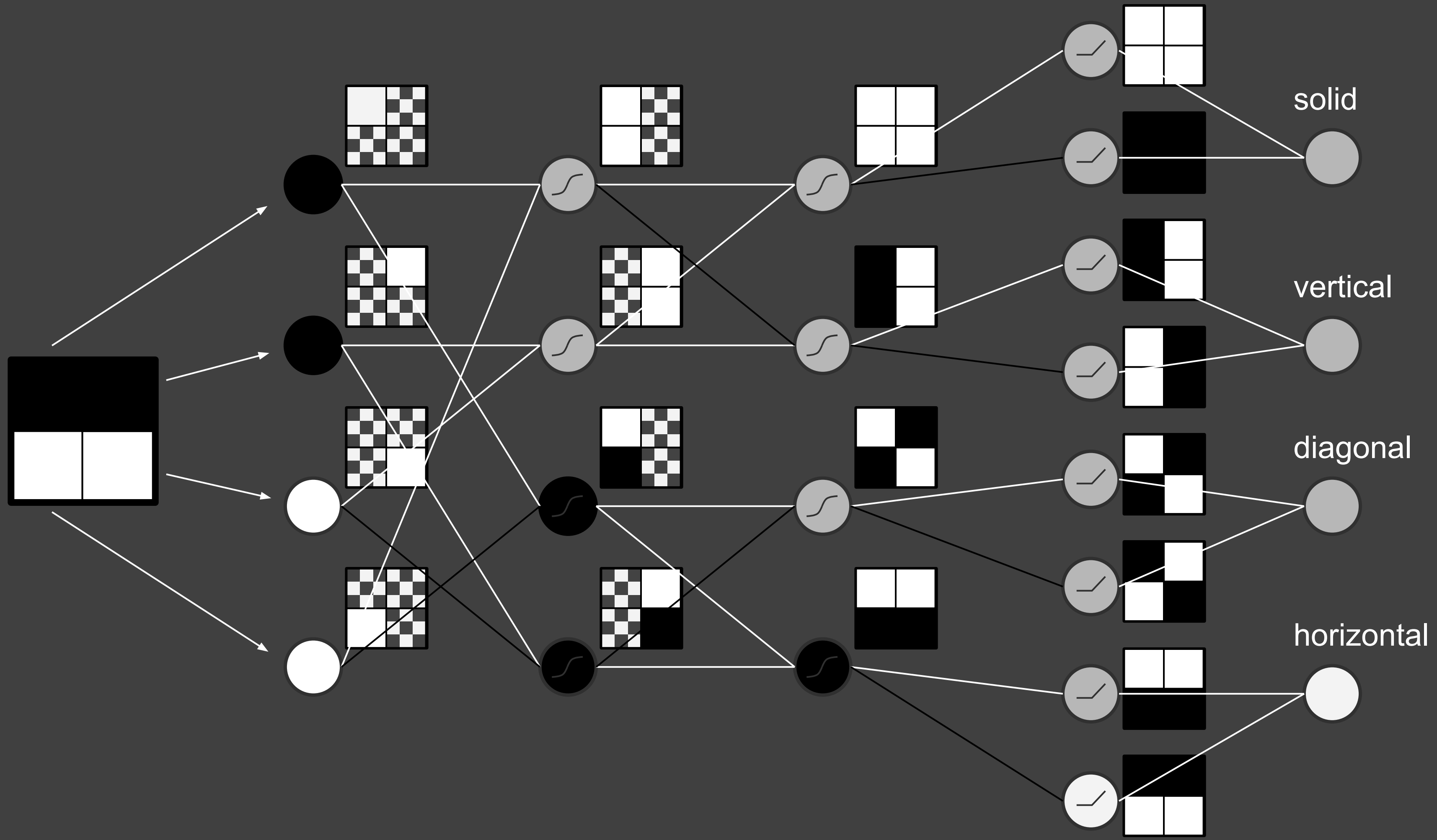
solid
vertical
diagonal
horizontal

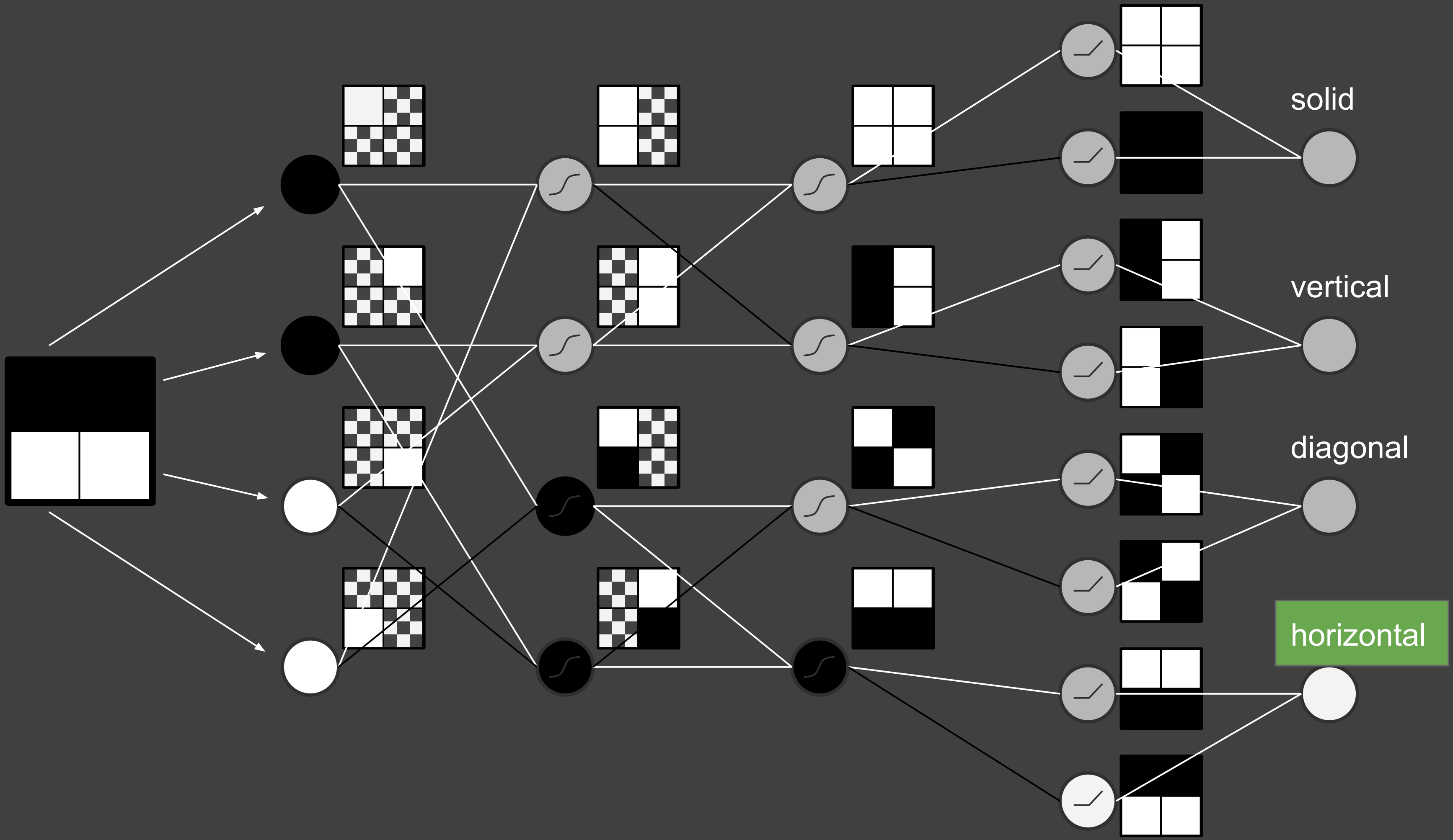

solid
vertical
diagonal
horizontal


solid
vertical
diagonal
horizontal


solid
vertical
diagonal
horizontal

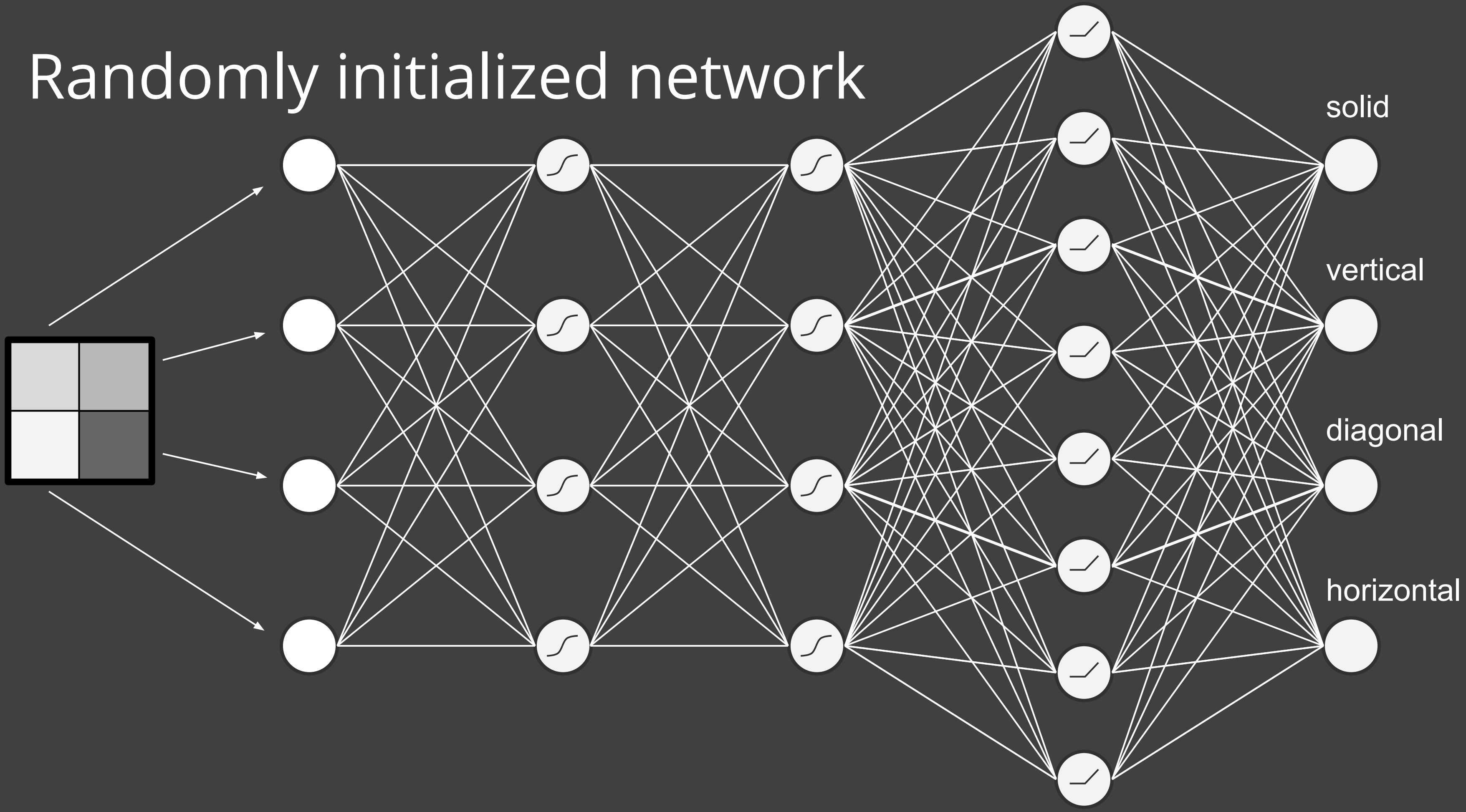

solid
vertical
diagonal
horizontal


solid
vertical
diagonal
horizontal

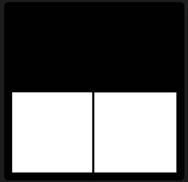


solid
vertical
diagonal
horizontal

Randomly initialized network

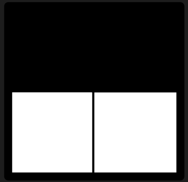


Errors



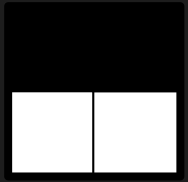
truth	
0.	solid
0.	vertical
0.	diagonal
1.	horizontal

Errors



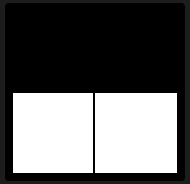
truth	answer	
		solid
0.	.5	
		vertical
0.	.75	
		diagonal
0.	-.25	
		horizontal
1.	-.75	

Errors



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

Errors



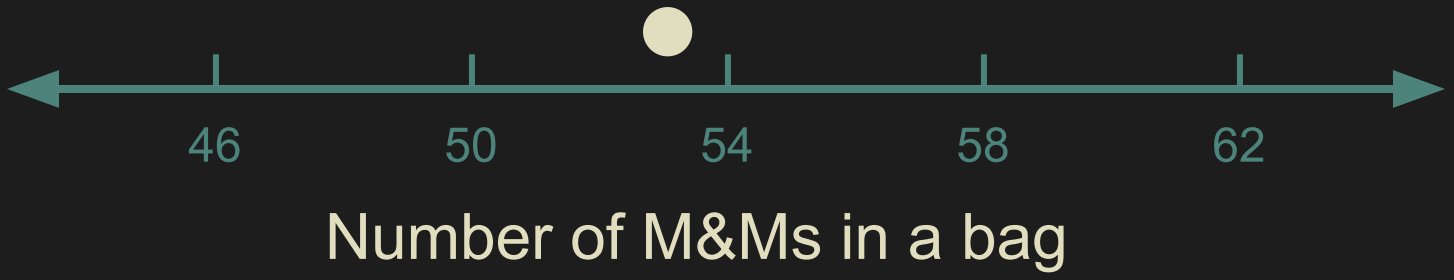
	error	truth	answer
solid	.5	0.	.5
vertical	.75	0.	.75
diagonal	.25	0.	-.25
horizontal	1.75	1.	-.75
total	3.25		

Loss Function

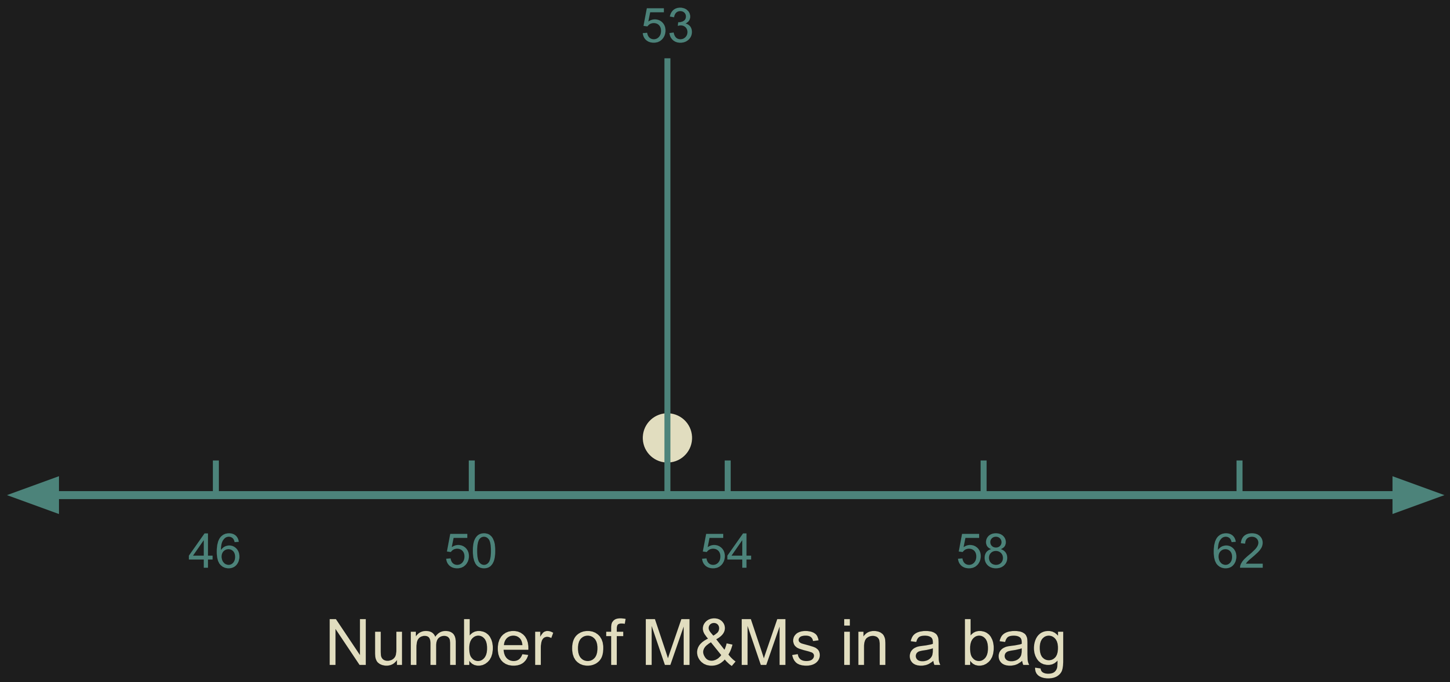
- ft. M&M (eminem?)

From data to model:
How many M&Ms in a bag?

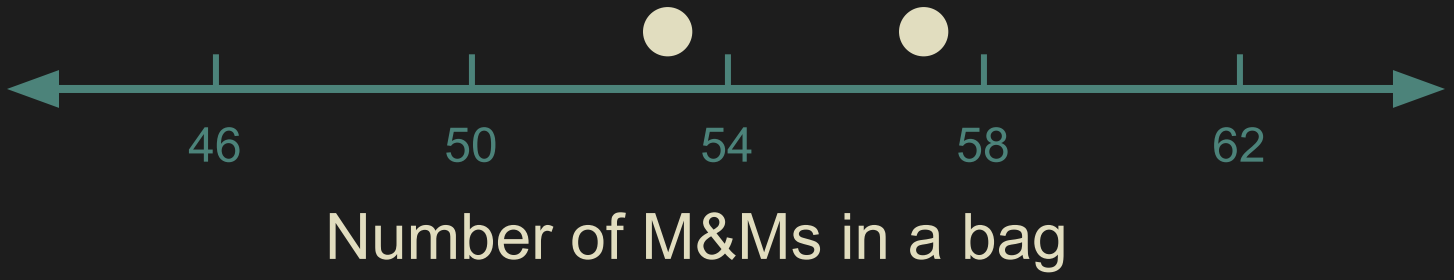
How many M&Ms in a bag?



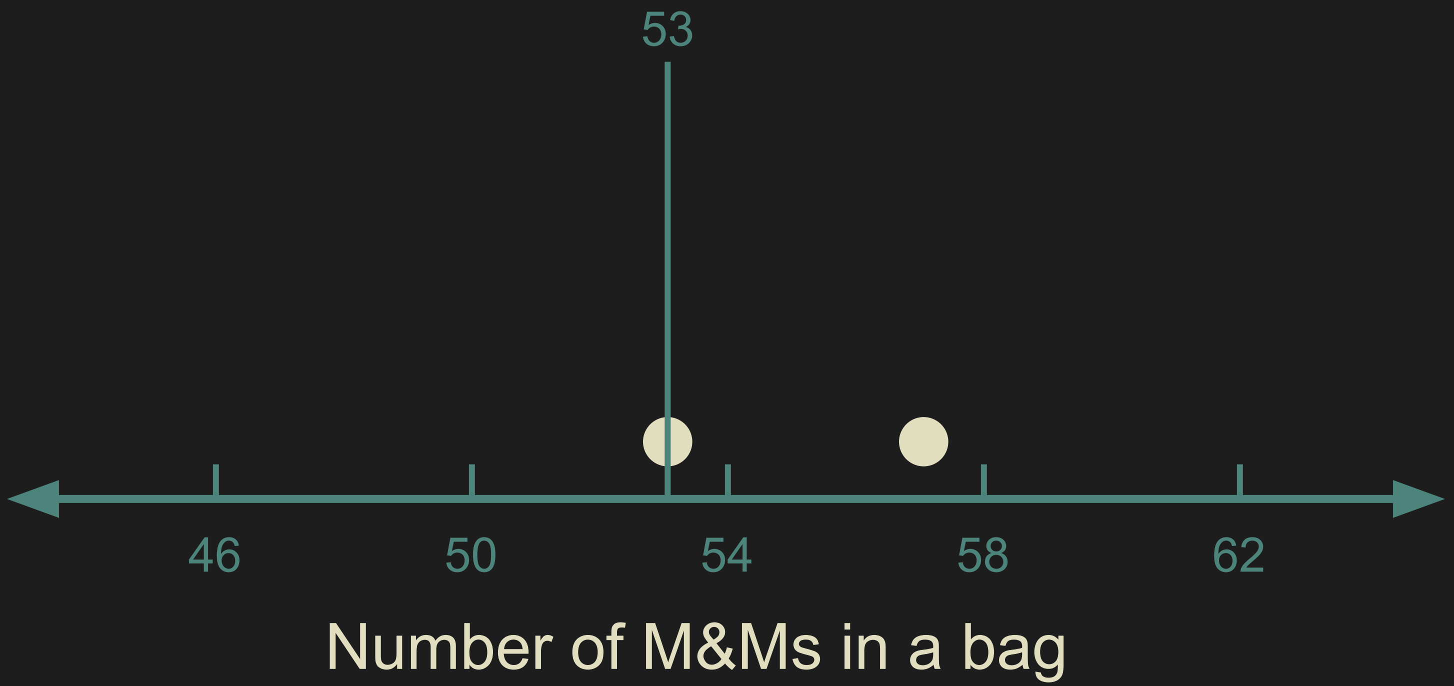
How many M&Ms in a bag?



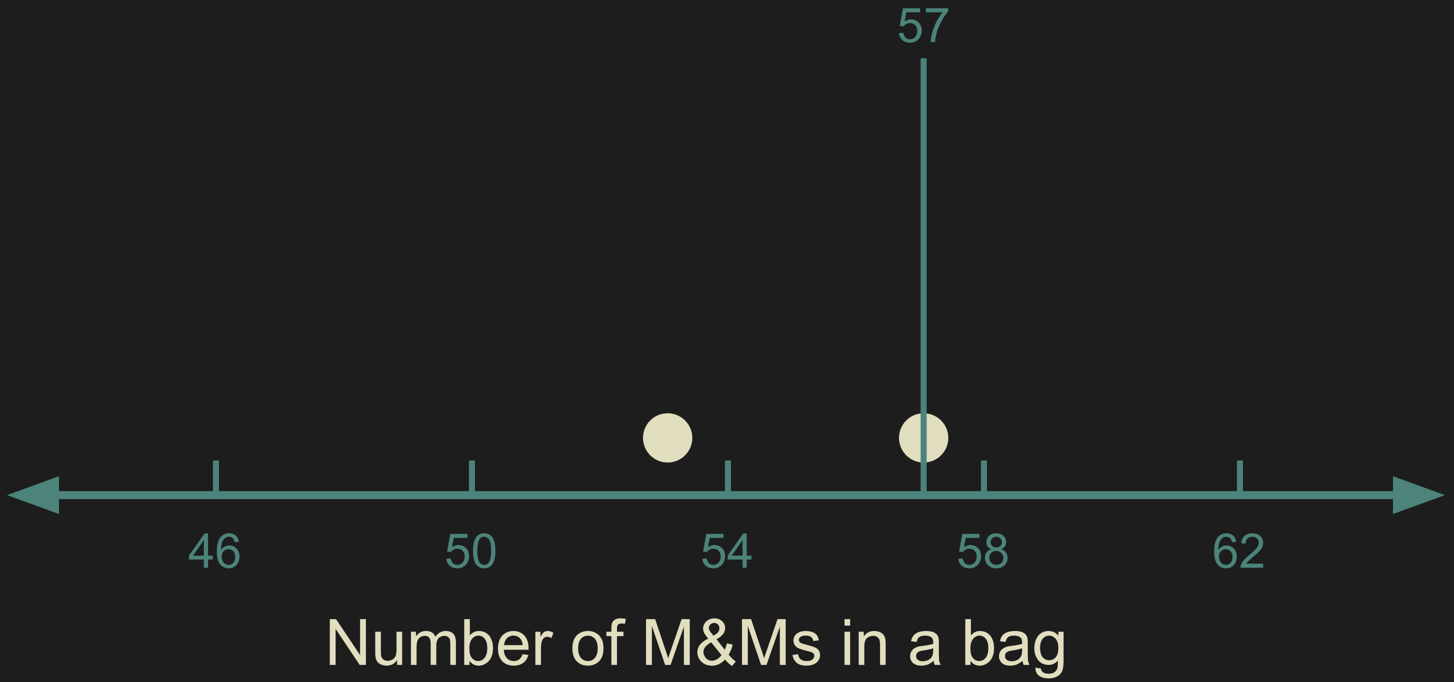
How many M&Ms in a bag?



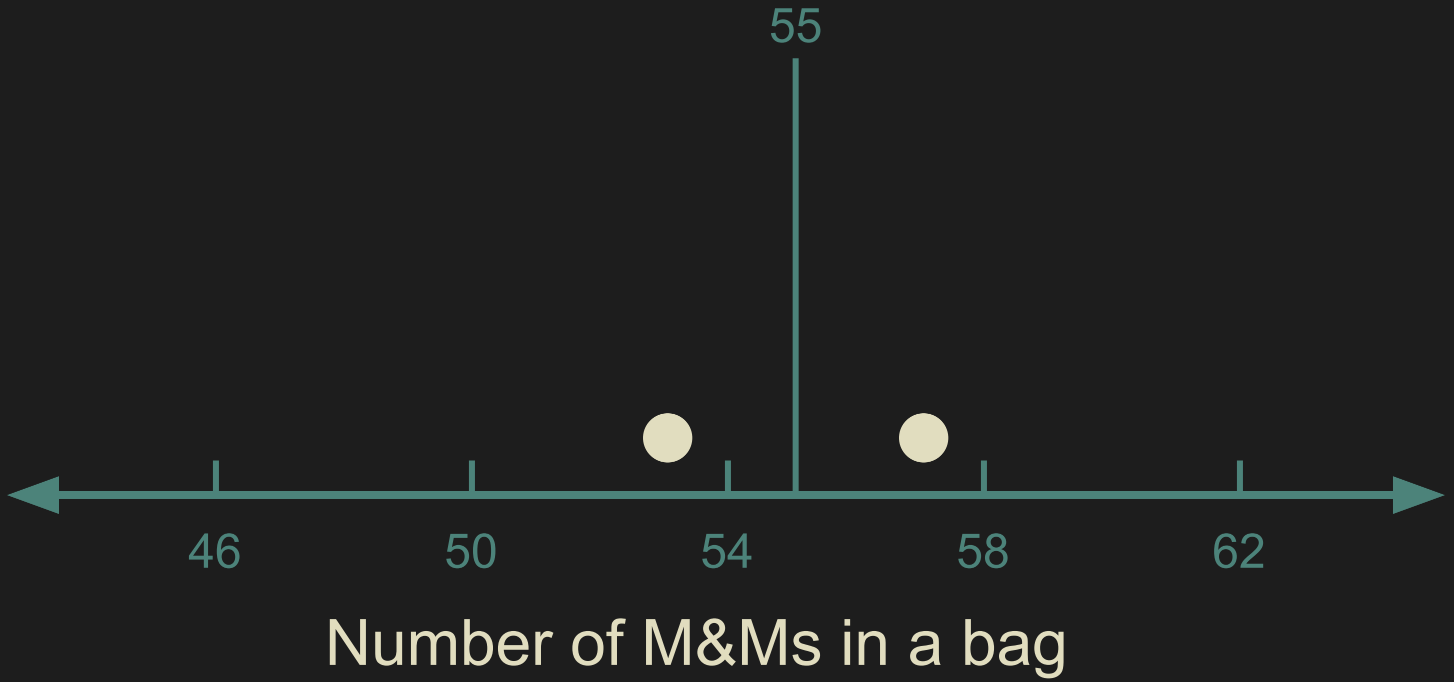
How many M&Ms in a bag?



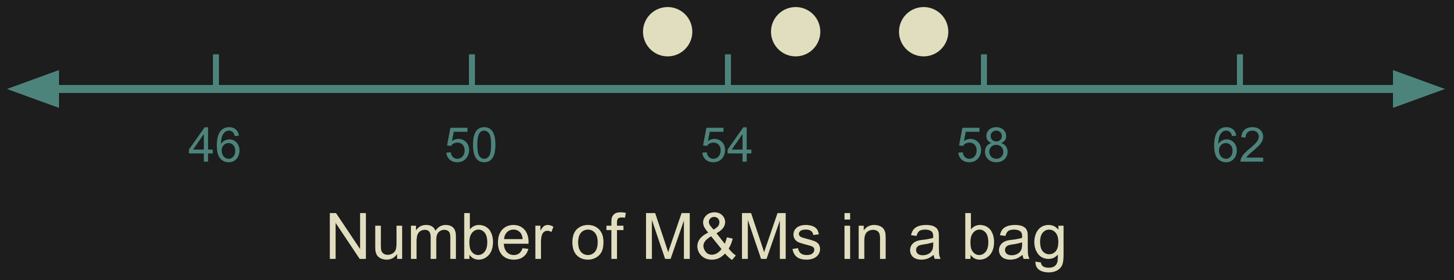
How many M&Ms in a bag?



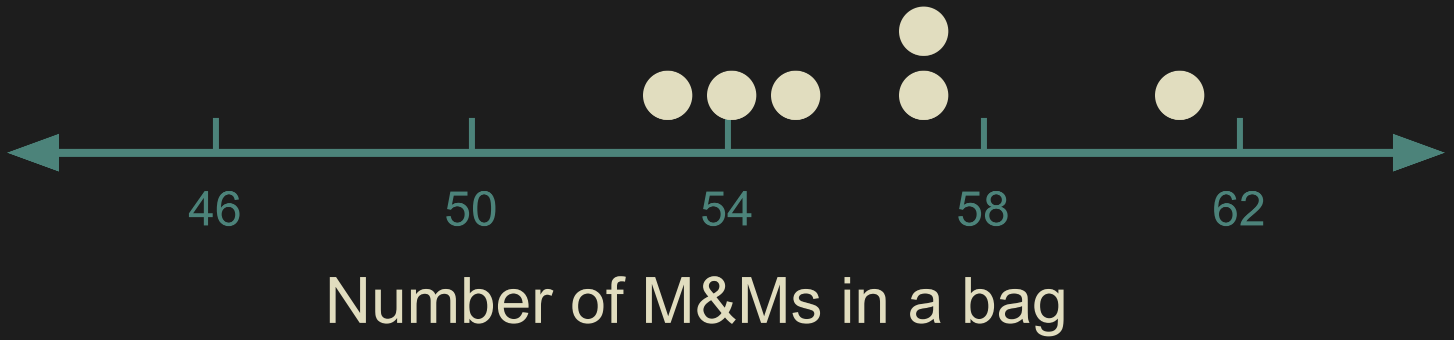
How many M&Ms in a bag?



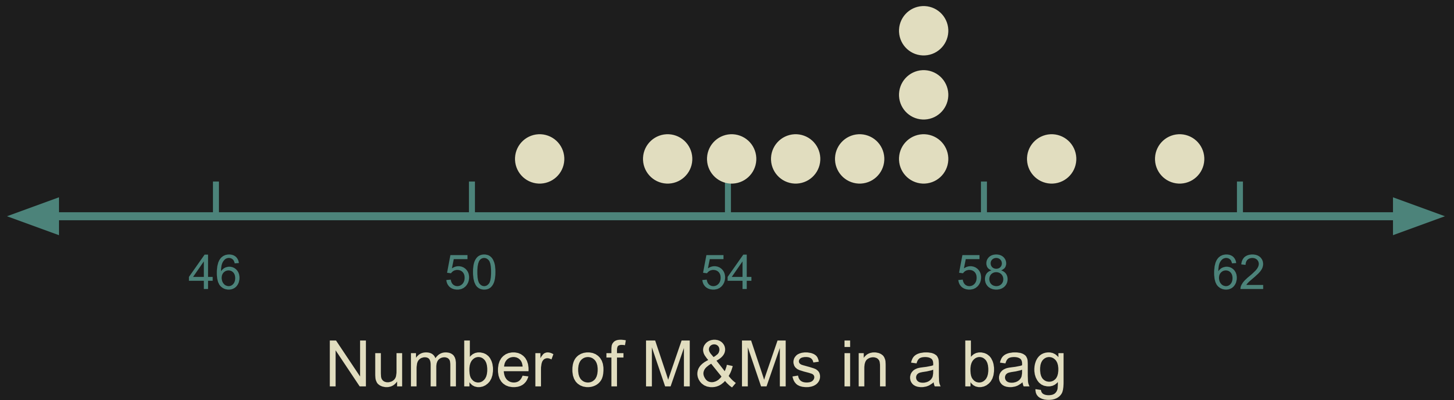
How many M&Ms in a bag?



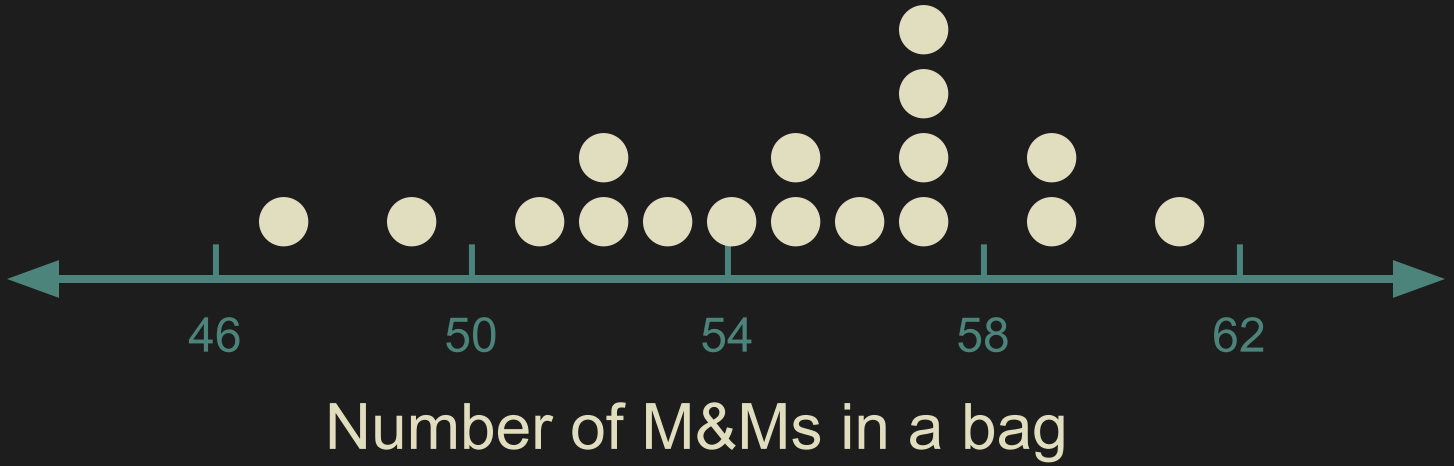
How many M&Ms in a bag?



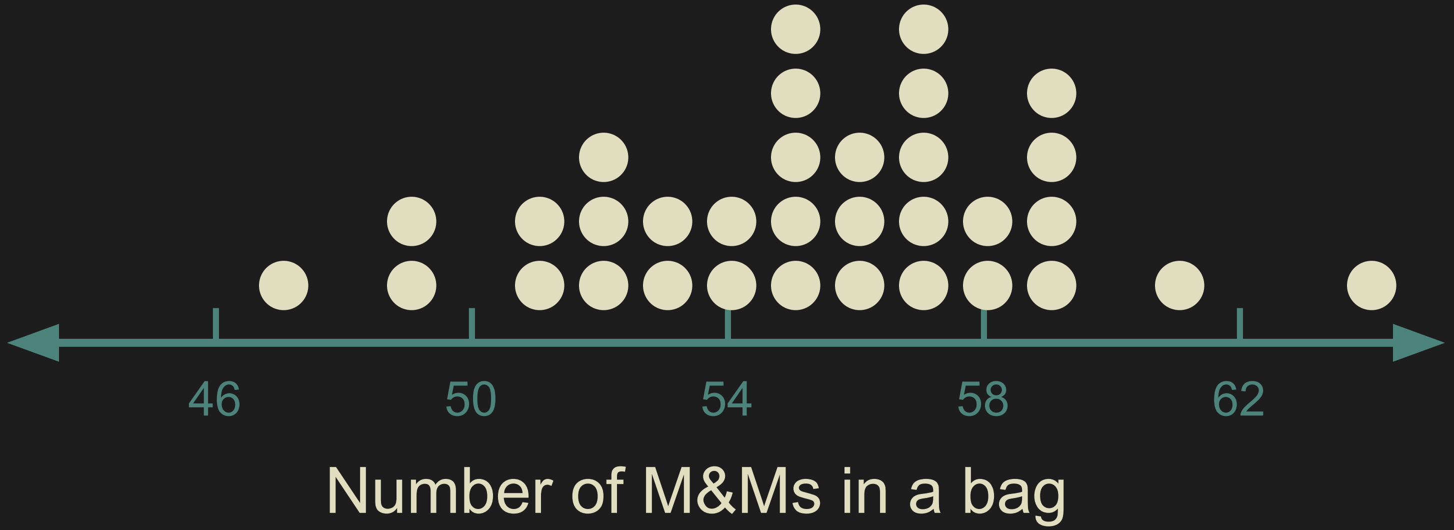
How many M&Ms in a bag?



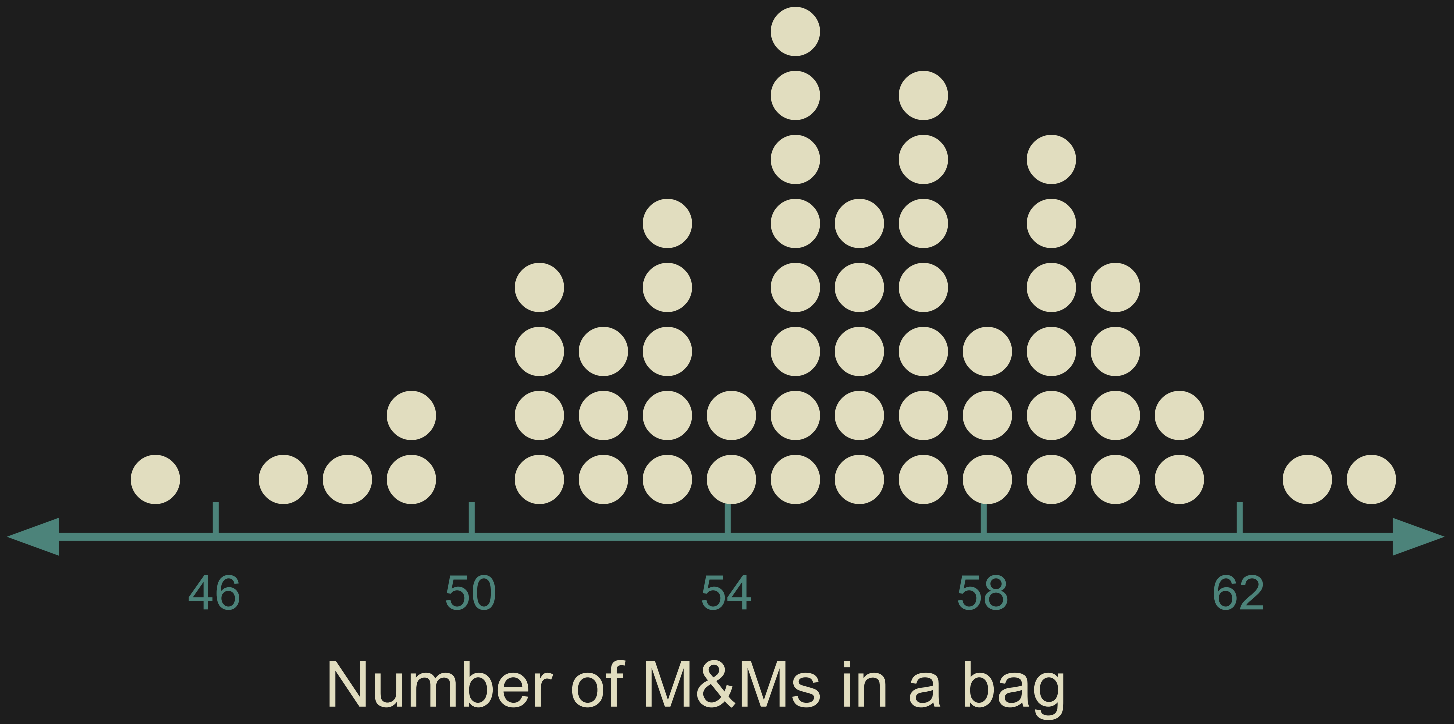
How many M&Ms in a bag?



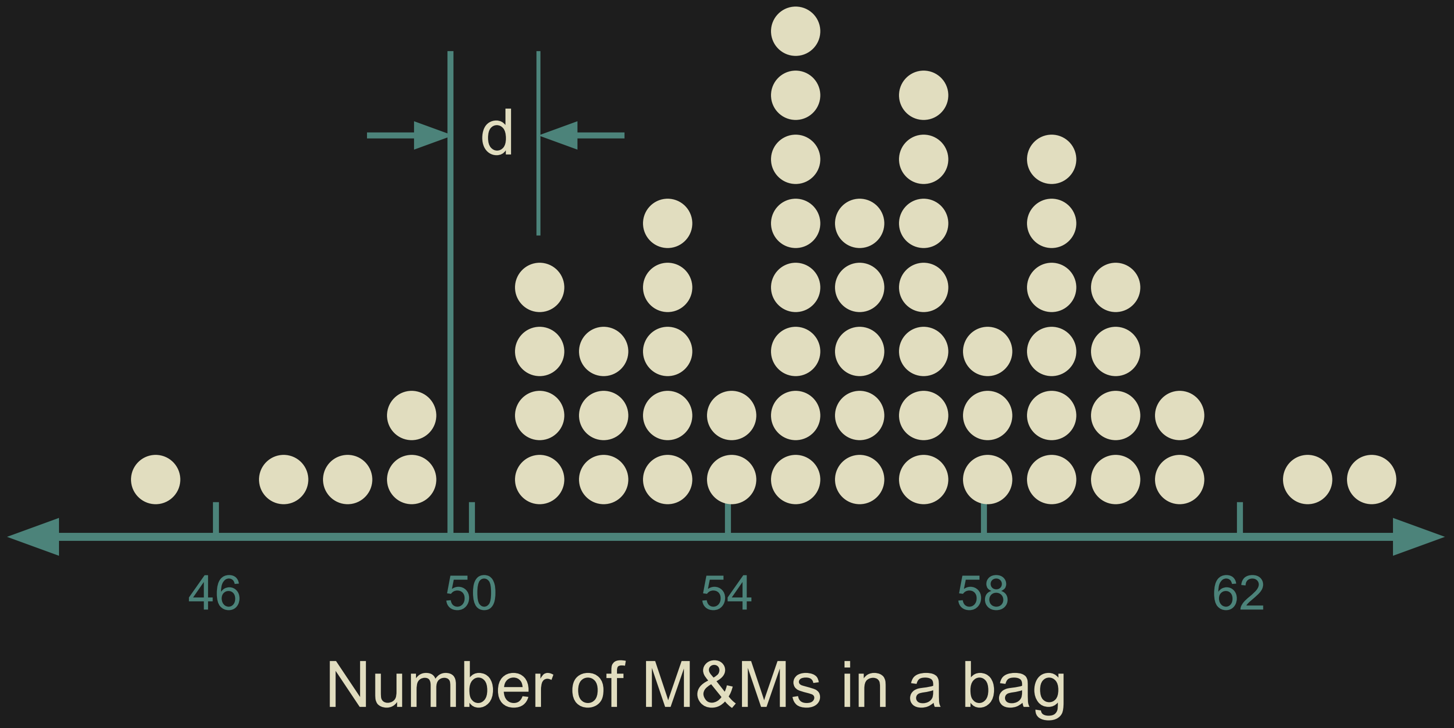
How many M&Ms in a bag?



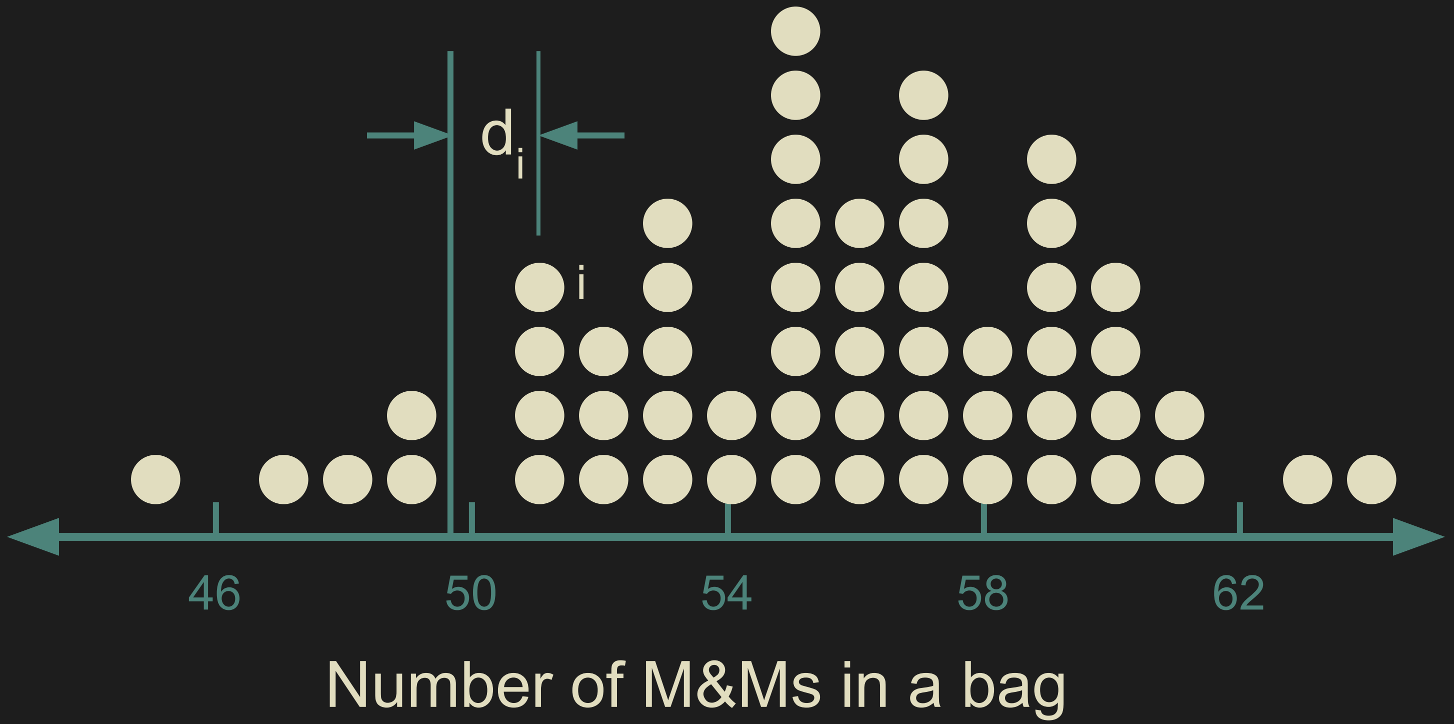
How many M&Ms in a bag?



How wrong is any answer?



How wrong is an answer?



What is the cost of being off by d?

$$d = n_{actual} - n_{guess}$$

What is the cost of being off by d?

$d = n_{actual} - n_{guess}$

cost	deviation 1	2	4	8
sqrt(|d|)	1	1.41	2	2.83
$\|d\|$	1	2	4	8
d^2	1	4	16	64
$10^{\|d\|-1}$	1	10	1000	10,000,000

What is the cost of being off by d?

$d = n_{actual} - n_{guess}$

cost	deviation 1	2	4	8
sqrt(|d|)	1	1.41	2	2.83
|d|	1	2	4	8
d^2	1	4	16	64
$10^{\|d\|-1}$	1	10	1000	10,000,000

What is the total cost of any guess?

For guess $n_{est,}$

$\mathcal{L}(n_{est})$

What is the total cost of any guess?

For guess $n_{est,}$

$$\mathcal{L}(n_{est}) = d_1^2 + d_2^2 + d_3^2 + \ldots + d_m^2$$

What is the total cost of any guess?

For guess $n_{est,}$

$$\mathcal{L}(n_{est}) = d_1^{\,2} + d_2^{\,2} + d_3^{\,2} + \ldots + d_m^{\,2}$$

$$\mathcal{L}(n_{est}) = (n_1 - n_{est})^2 + (n_2 - n_{est})^2 + (n_3 - n_{est})^2 + \ldots + (n_m - n_{est})^2$$

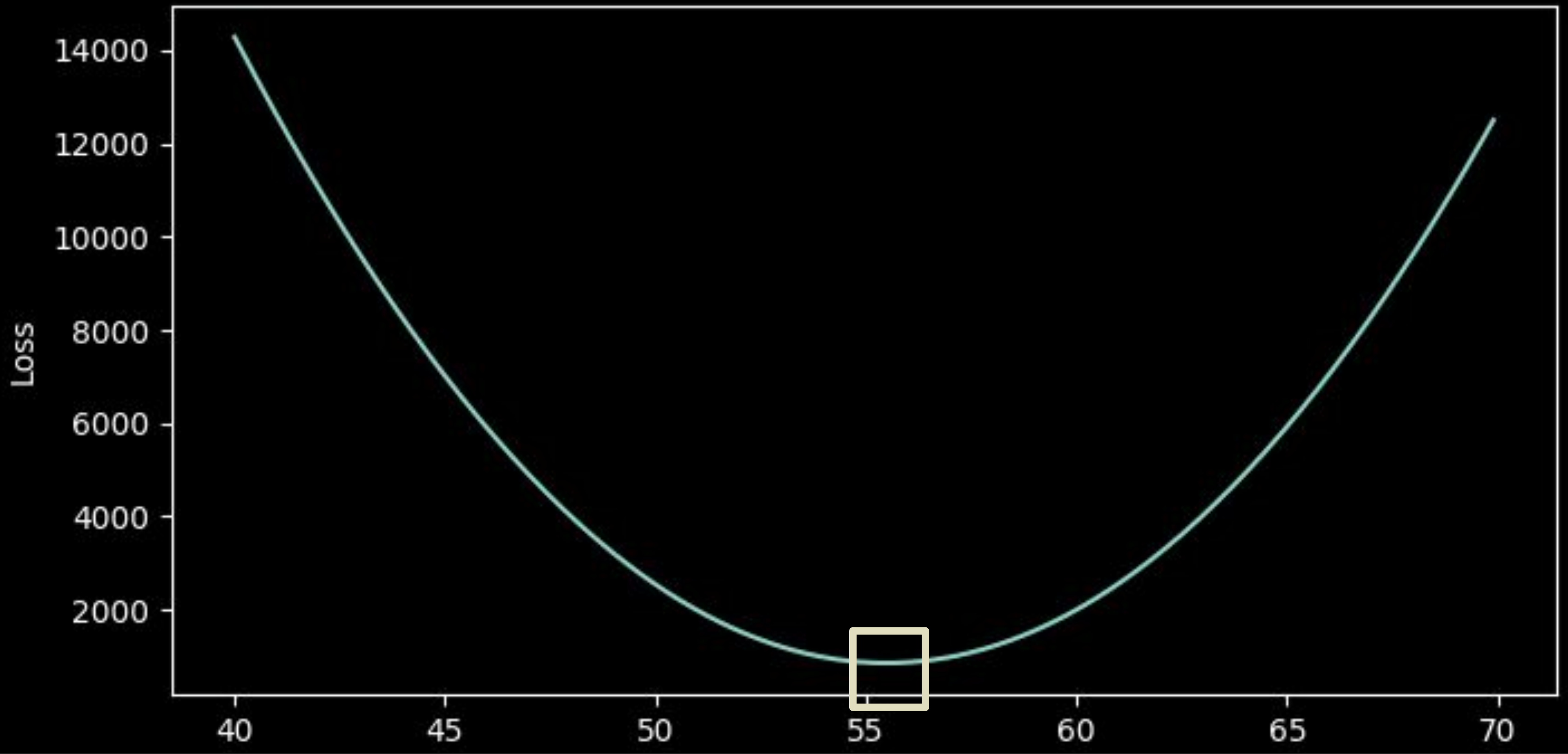
What is the total cost of any guess?

For guess $n_{est,}$

$$\mathcal{L}(n_{est}) = d_1^2 + d_2^2 + d_3^2 + \ldots + d_m^2$$

$$\mathcal{L}(n_{est}) = (n_1 - n_{est})^2 + (n_2 - n_{est})^2 + (n_3 - n_{est})^2 + \ldots + (n_m - n_{est})^2$$

$$\mathcal{L}(n_{est}) = \sum_i (n_i - n_{est})^2$$


Loss
14000
12000
10000
8000
6000
4000
2000
40
45
50
55
60
65
70

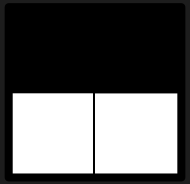
How does M&Ms relate to
neural networks
that we were talking about?

Are we in the right workshop?

“The estimation of
how wrong a prediction is
tells us how to get closer to a more correct prediction”

- some math guy

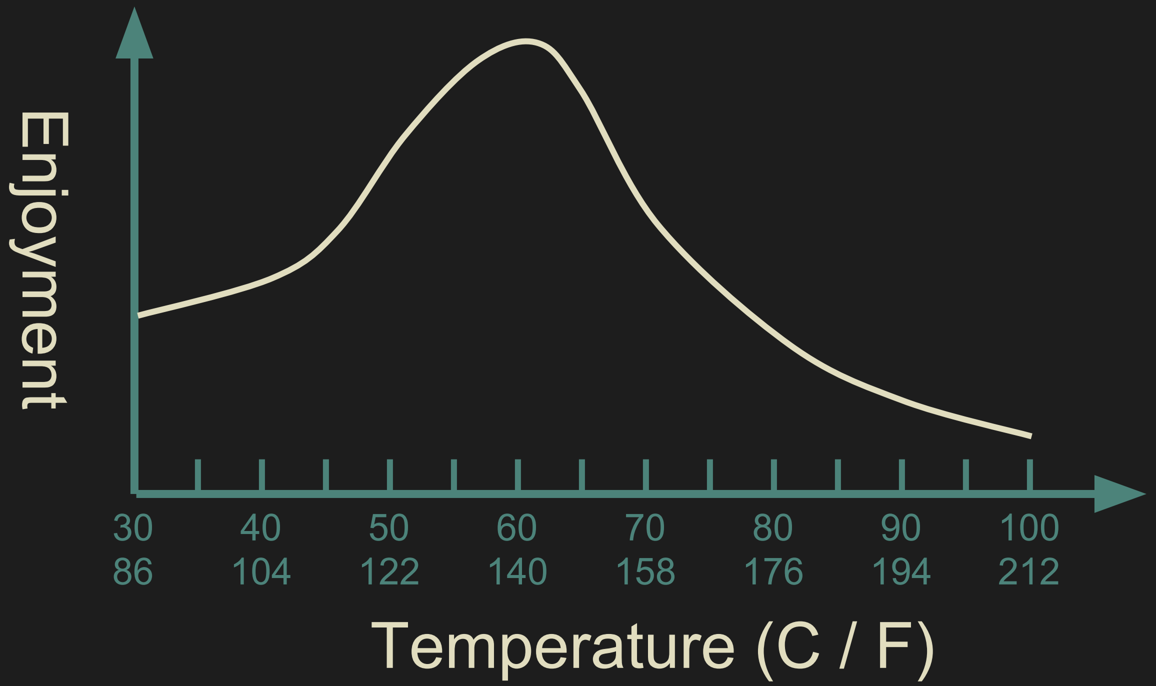
Errors



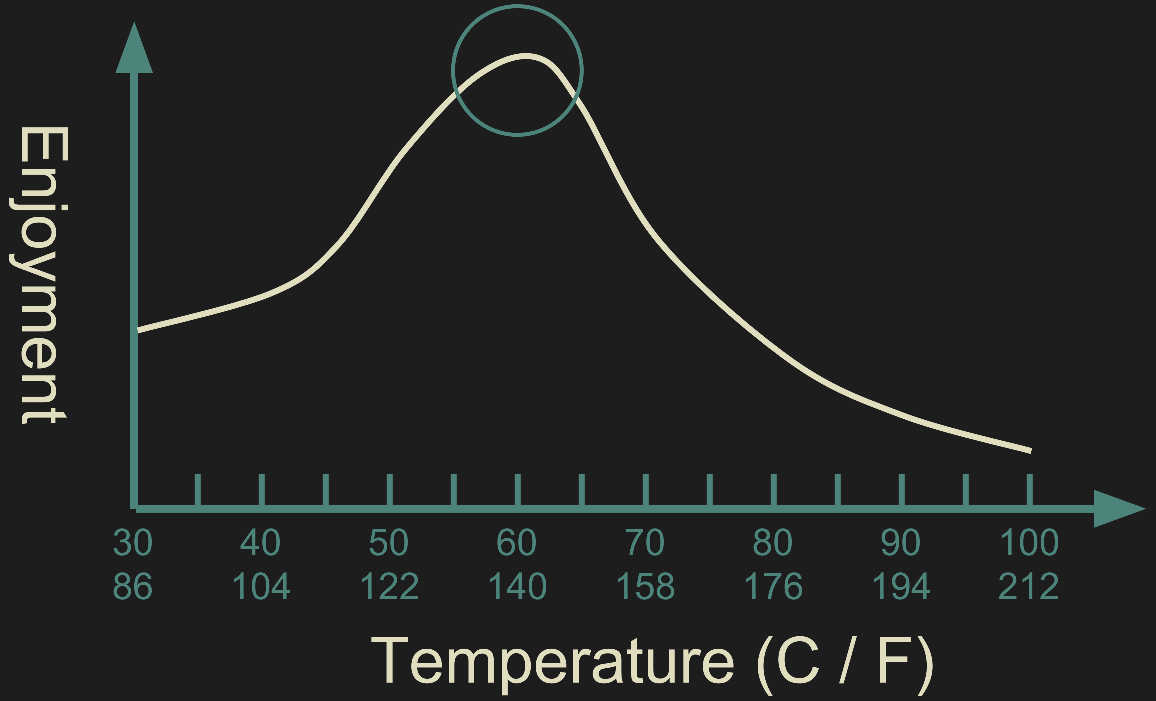
	error	truth	answer
solid	.5	0.	.5
vertical	.75	0.	.75
diagonal	.25	0.	-.25
horizontal	1.75	1.	-.75
total	3.25		

Optimization

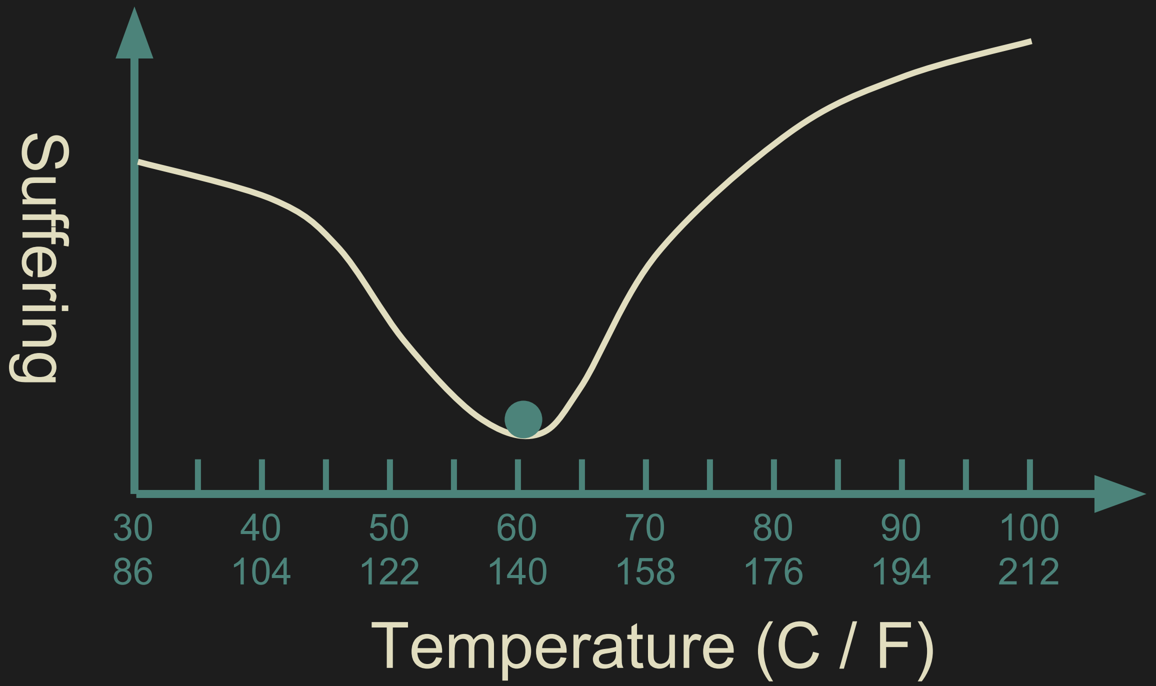
Tea drinking temperature



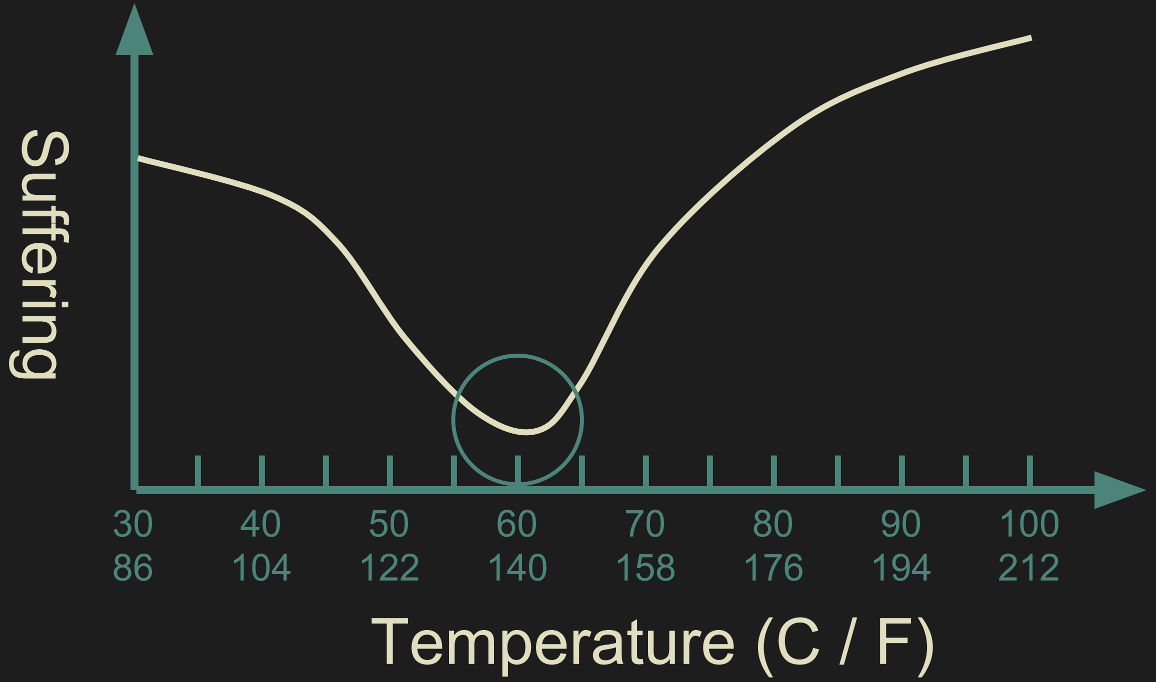
Tea drinking temperature



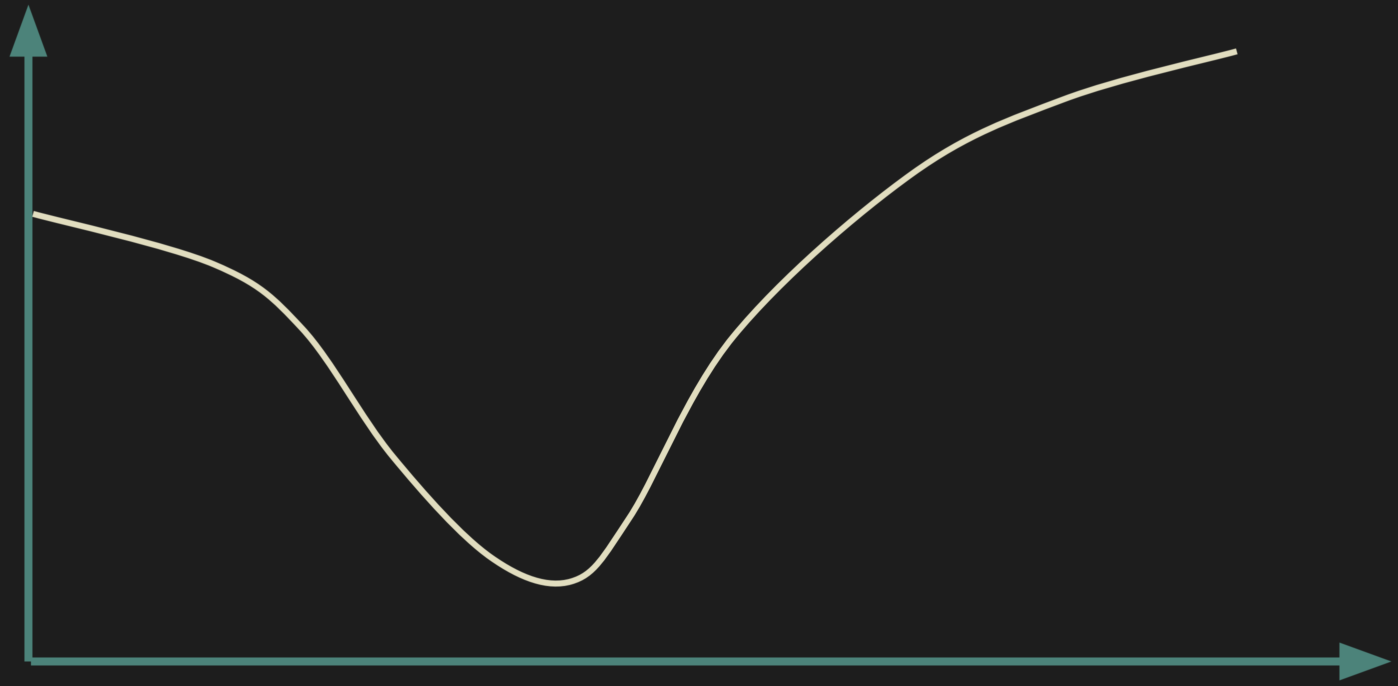
Tea drinking temperature



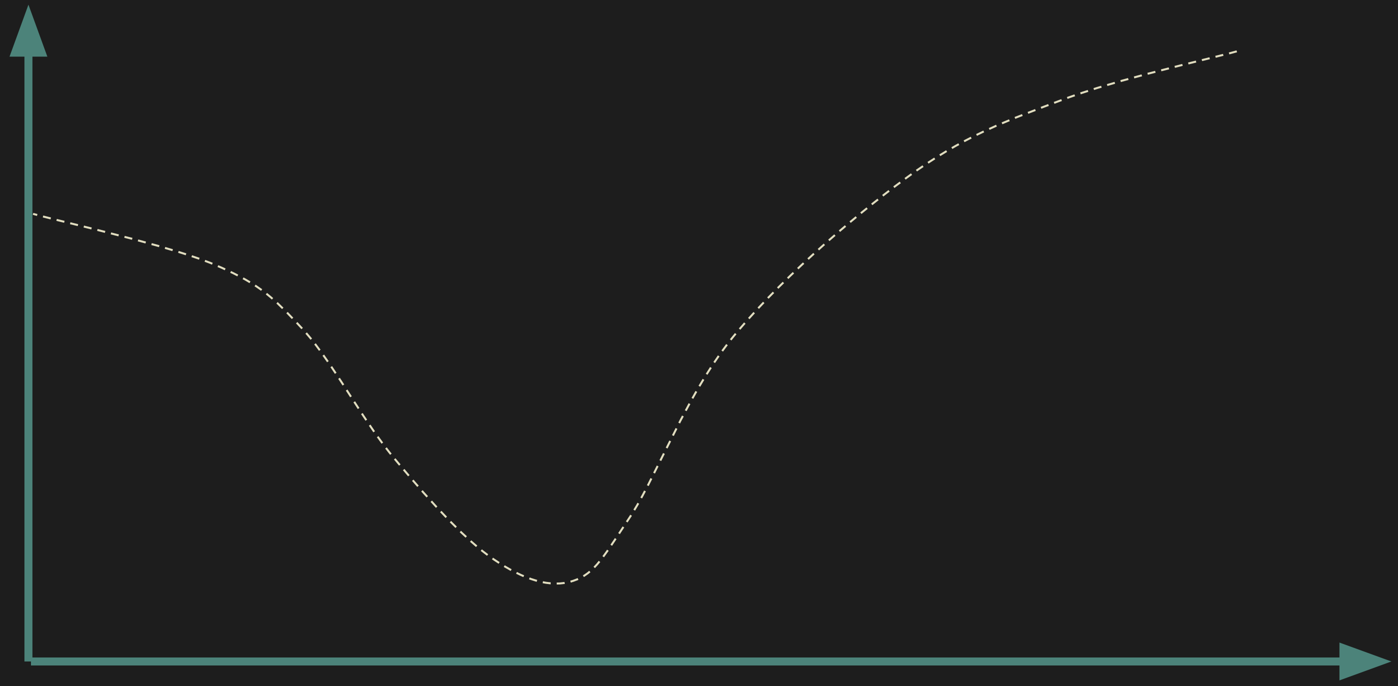
Tea drinking temperature



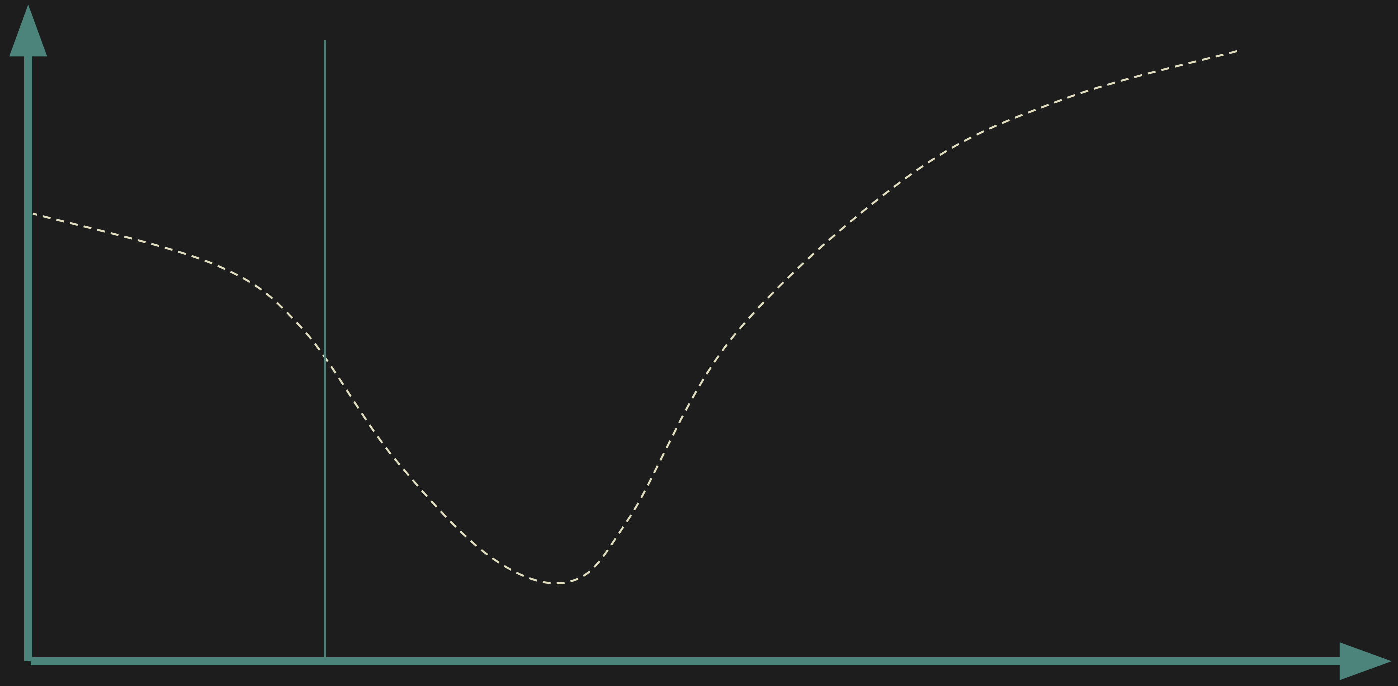
Pick the lowest point (Exhaustive search)



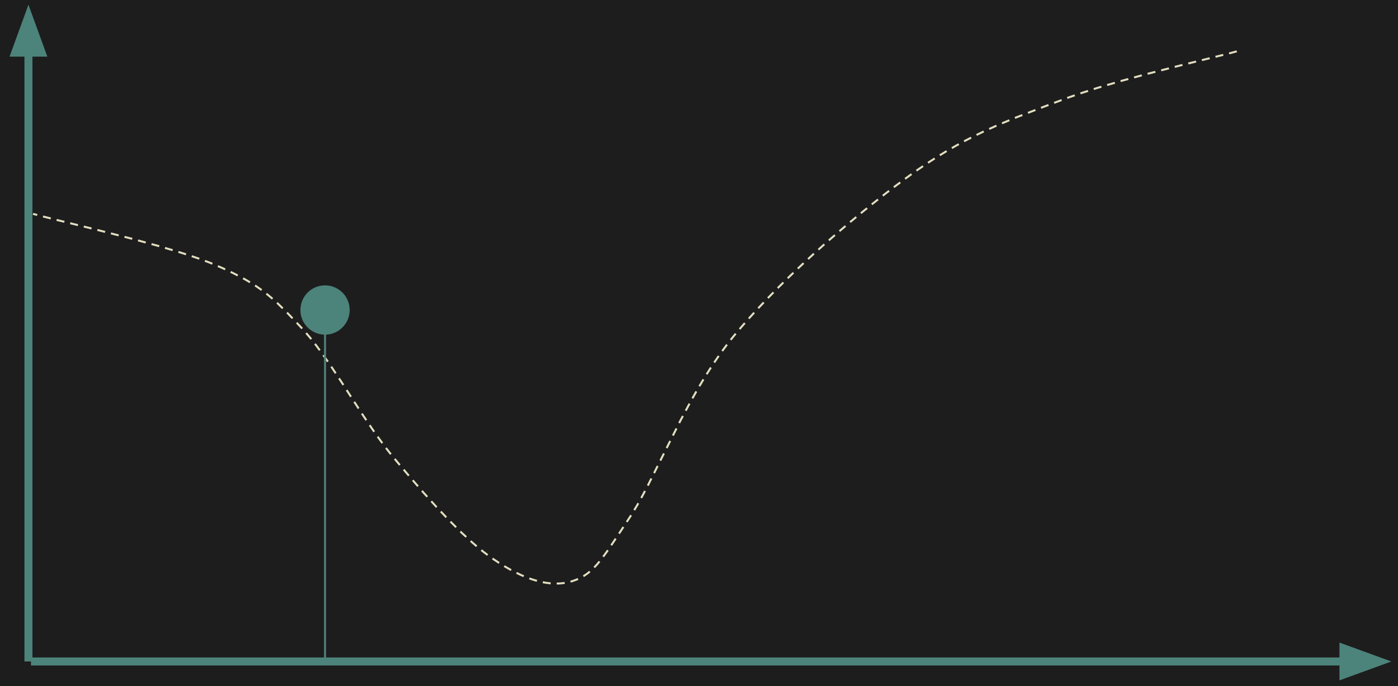
Exhaustive search



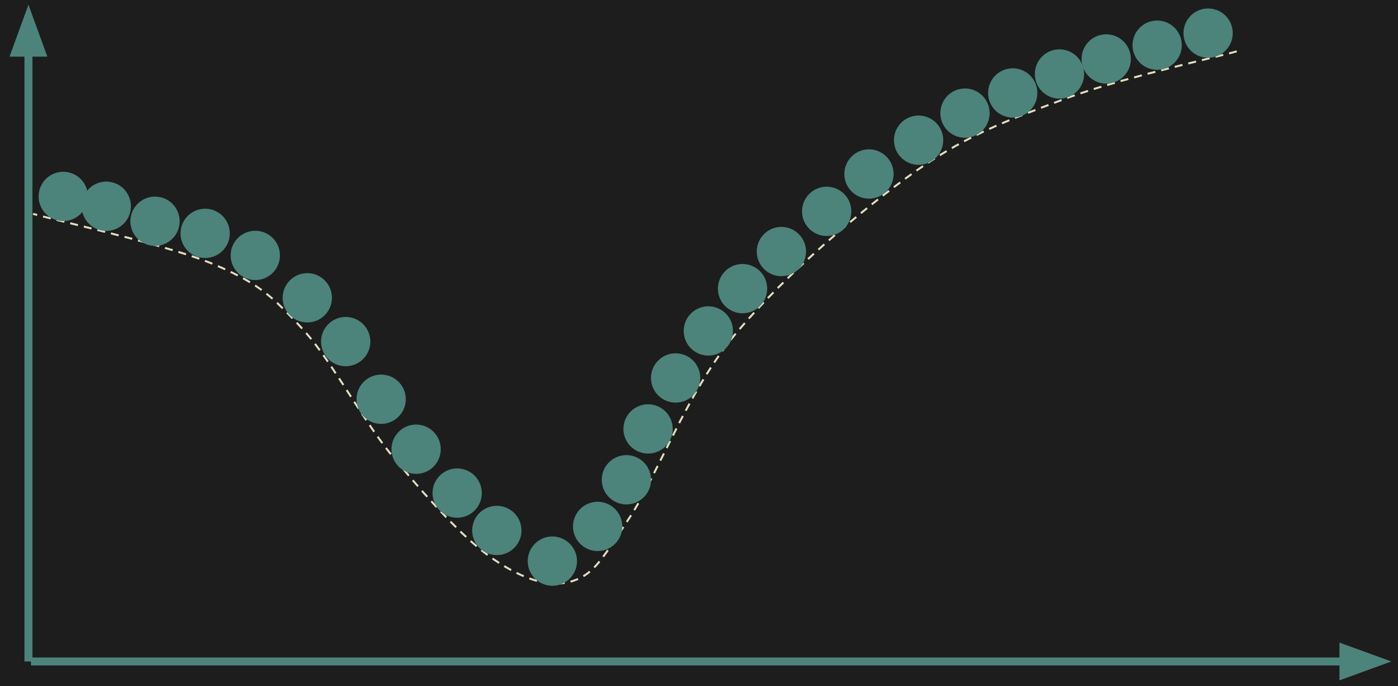
Exhaustive search



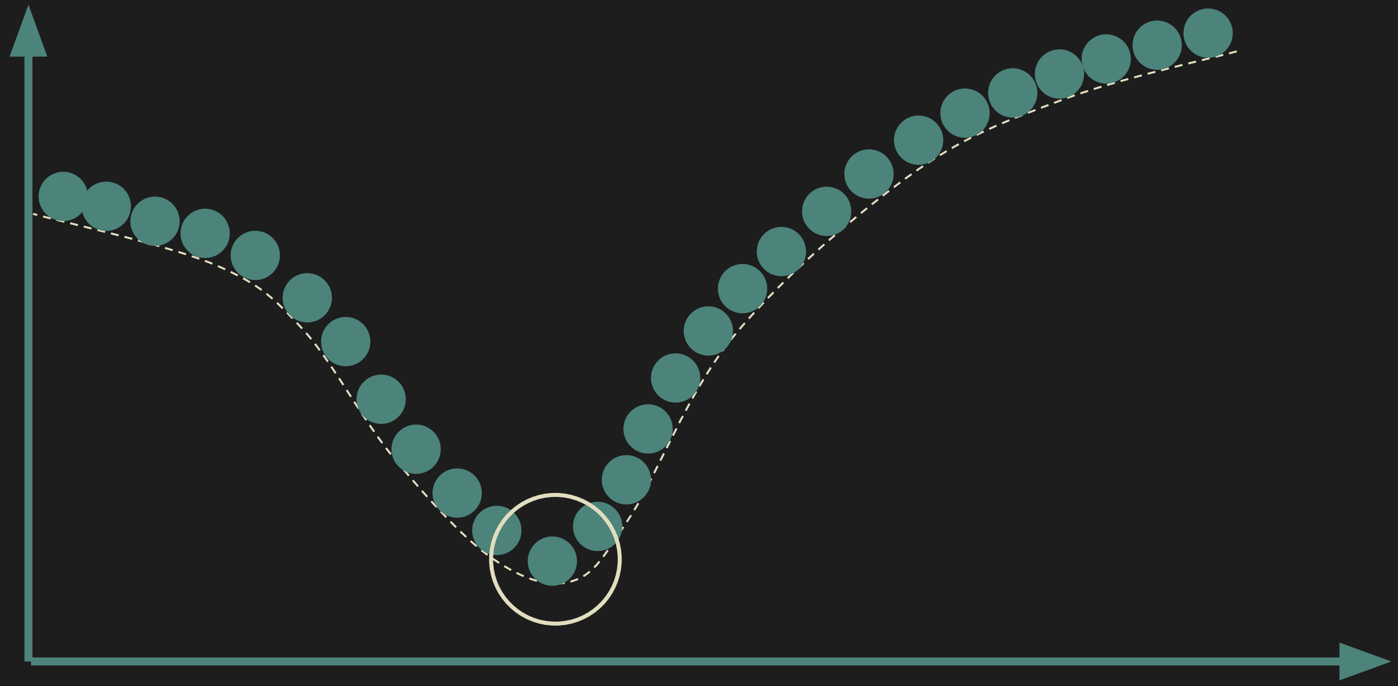
Exhaustive search



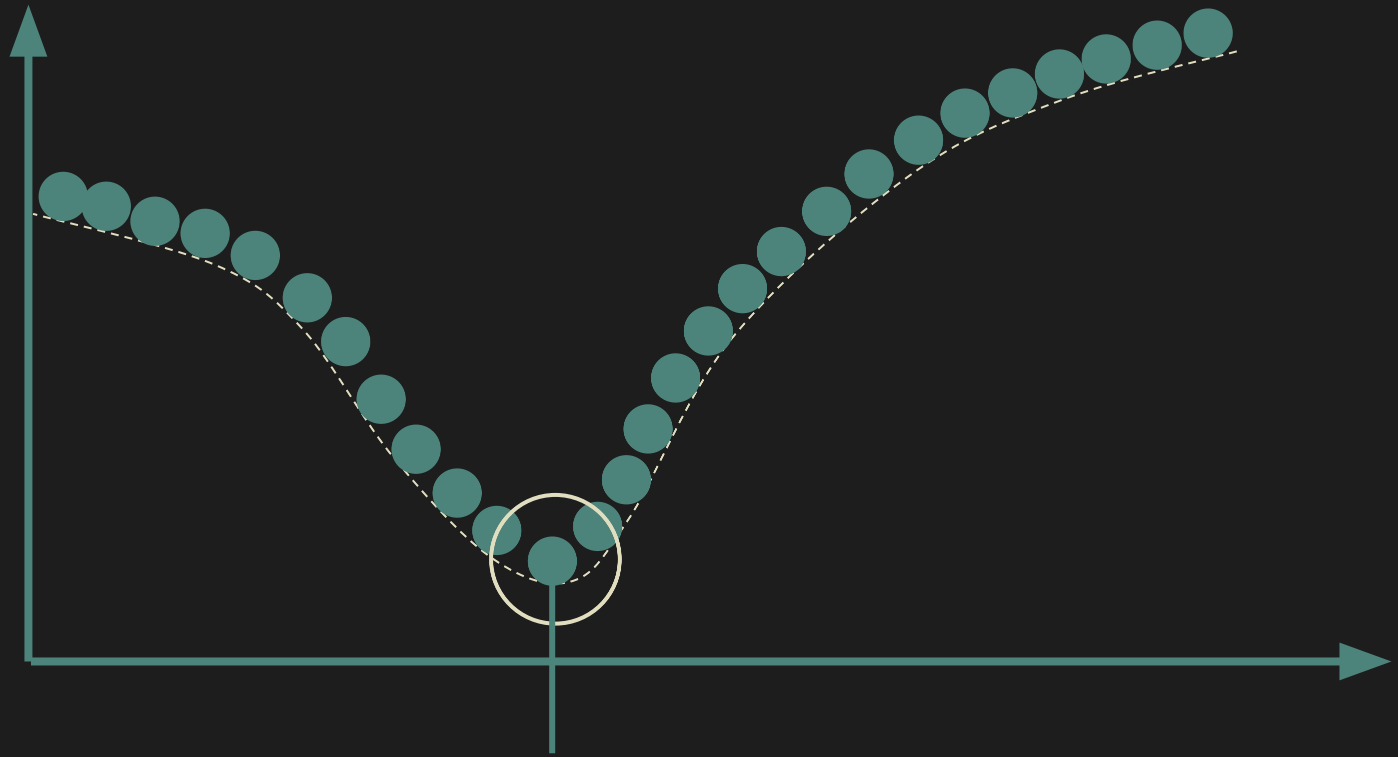
Exhaustive search



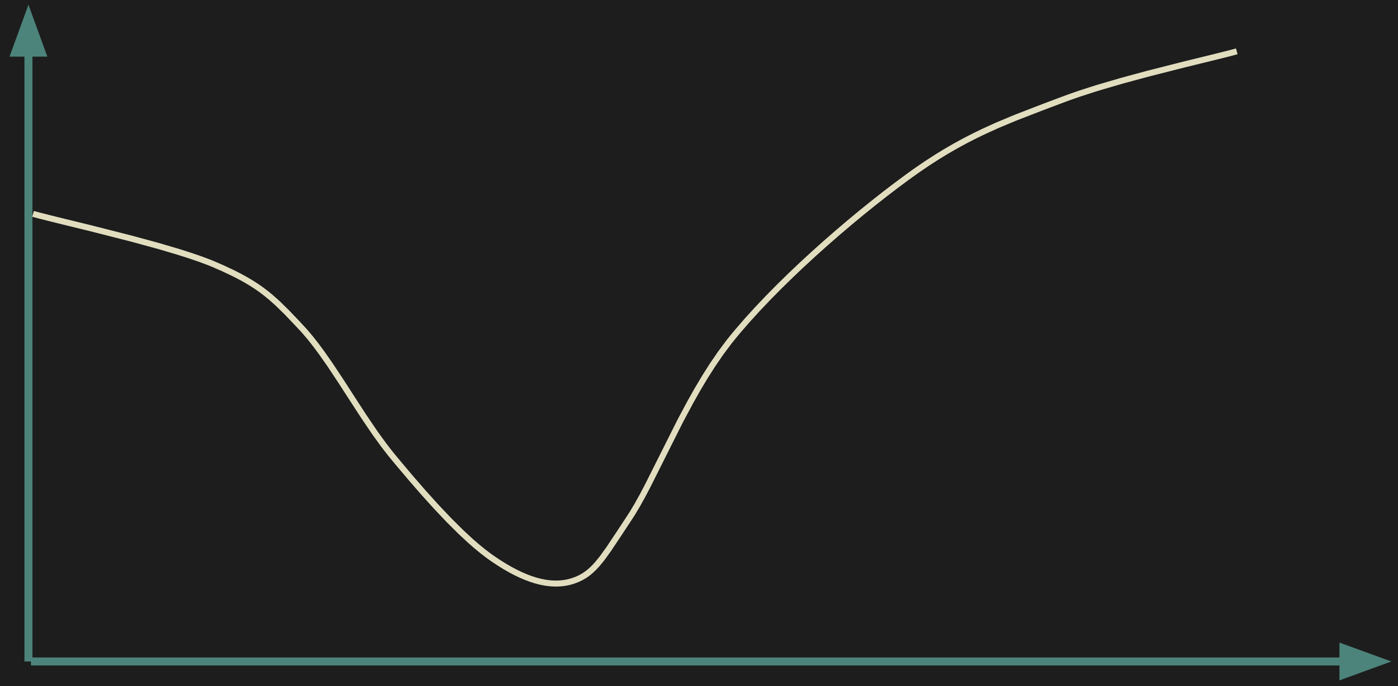
Exhaustive search



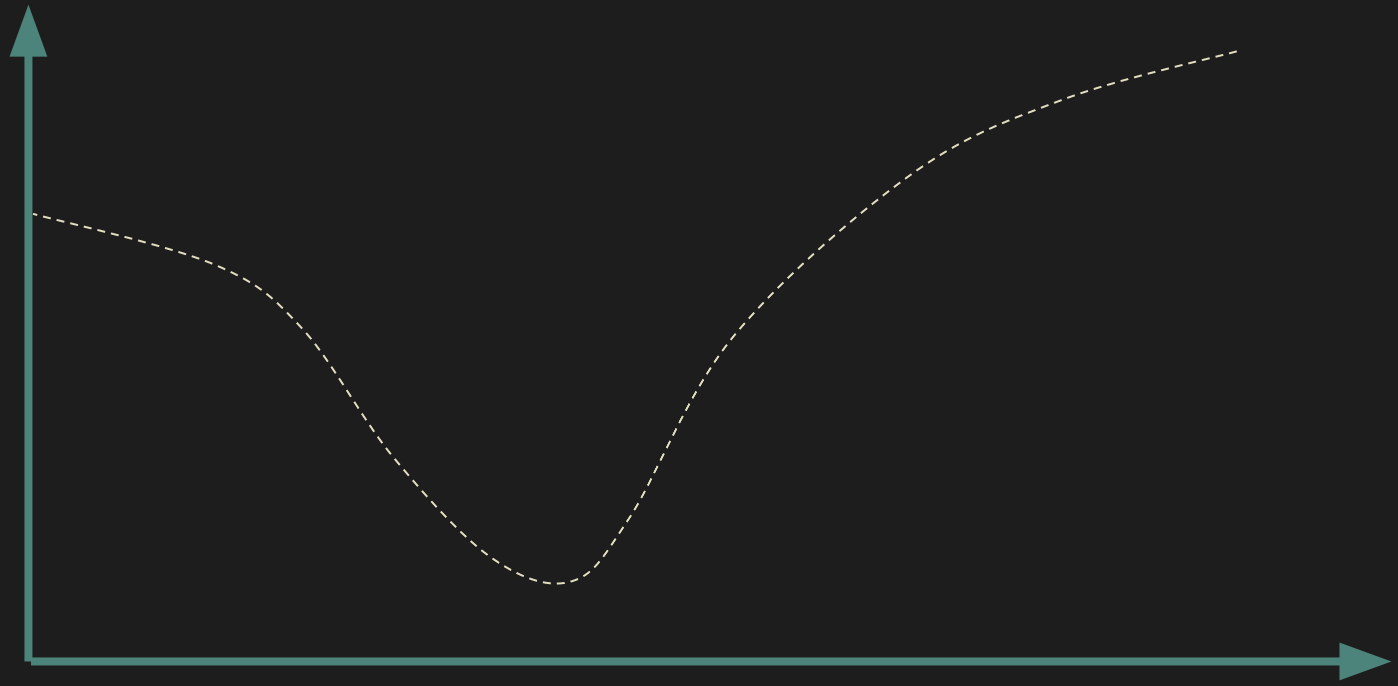
Exhaustive search



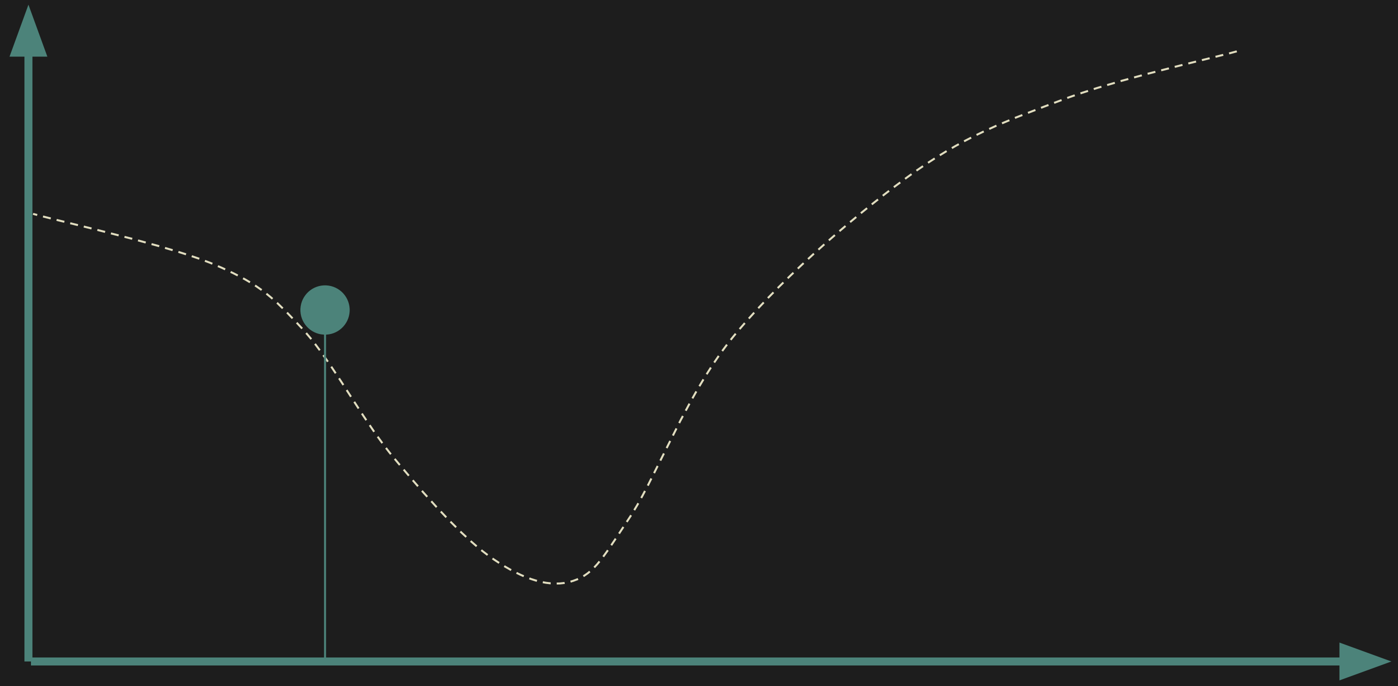
Let the marble roll downhill (Gradient descent)



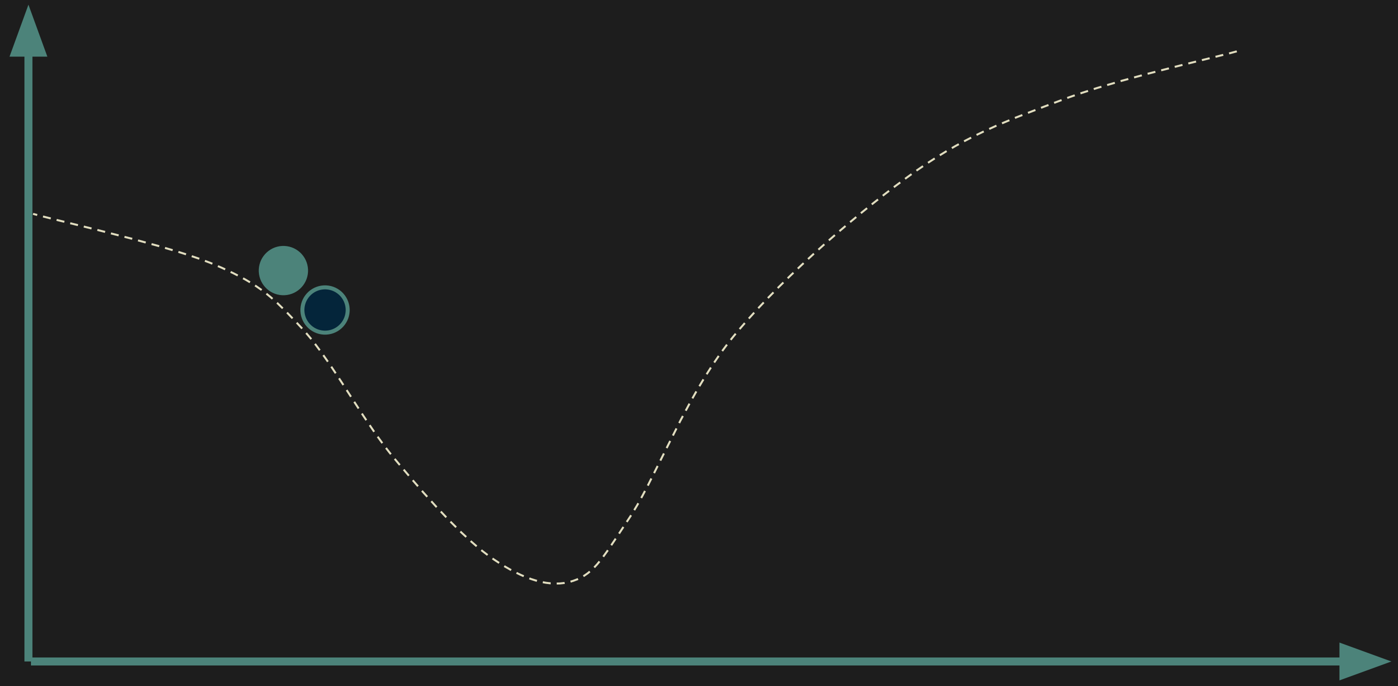
Gradient descent



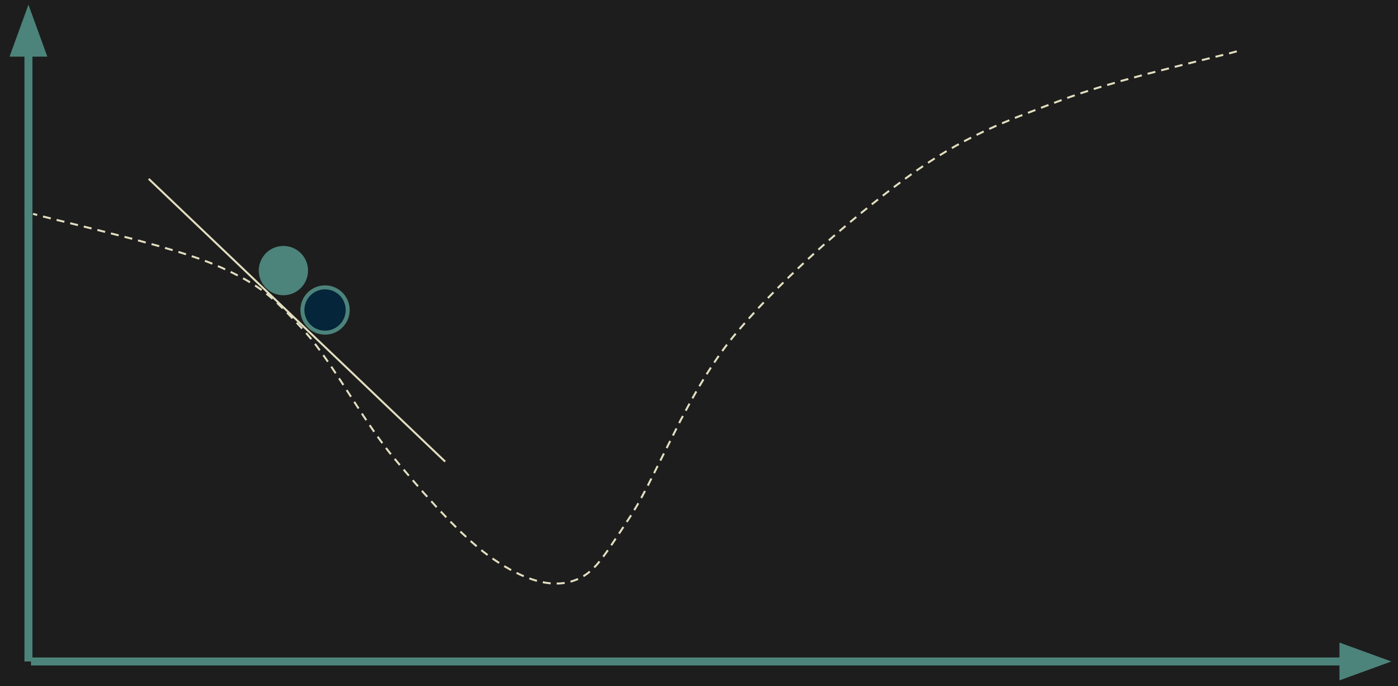
Gradient descent



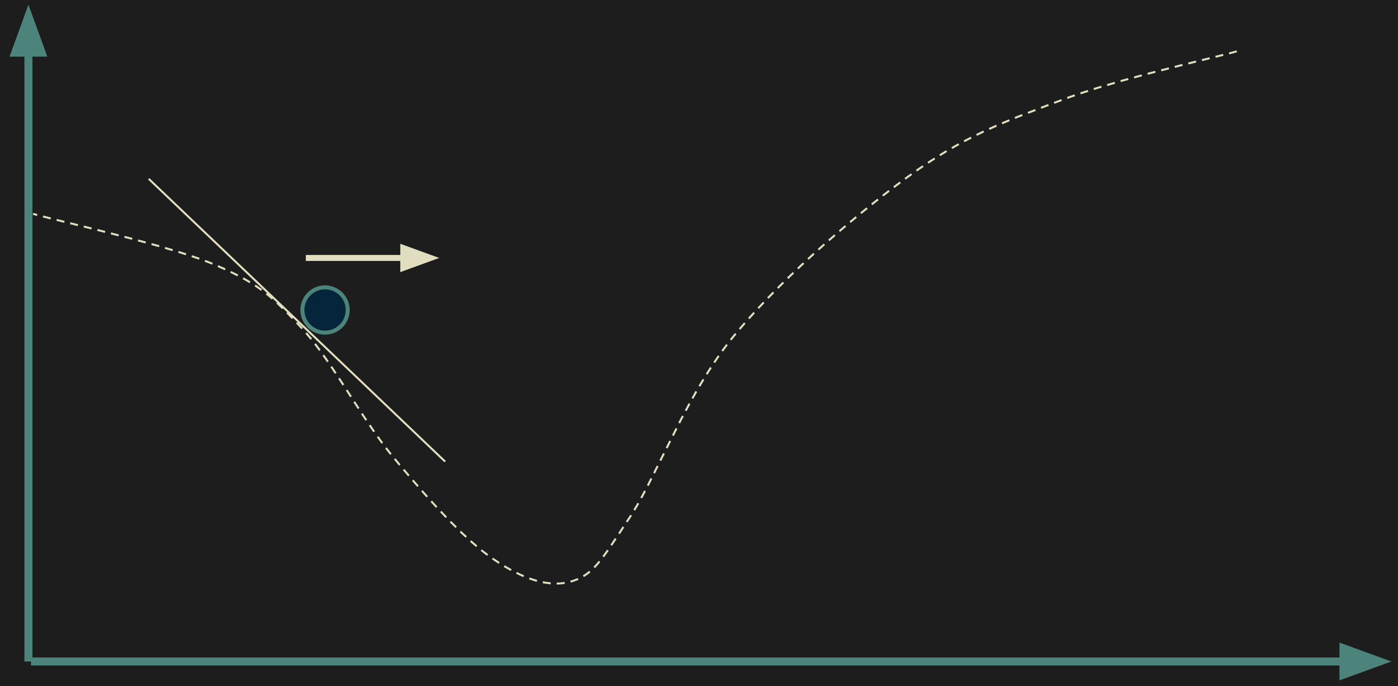
Gradient descent



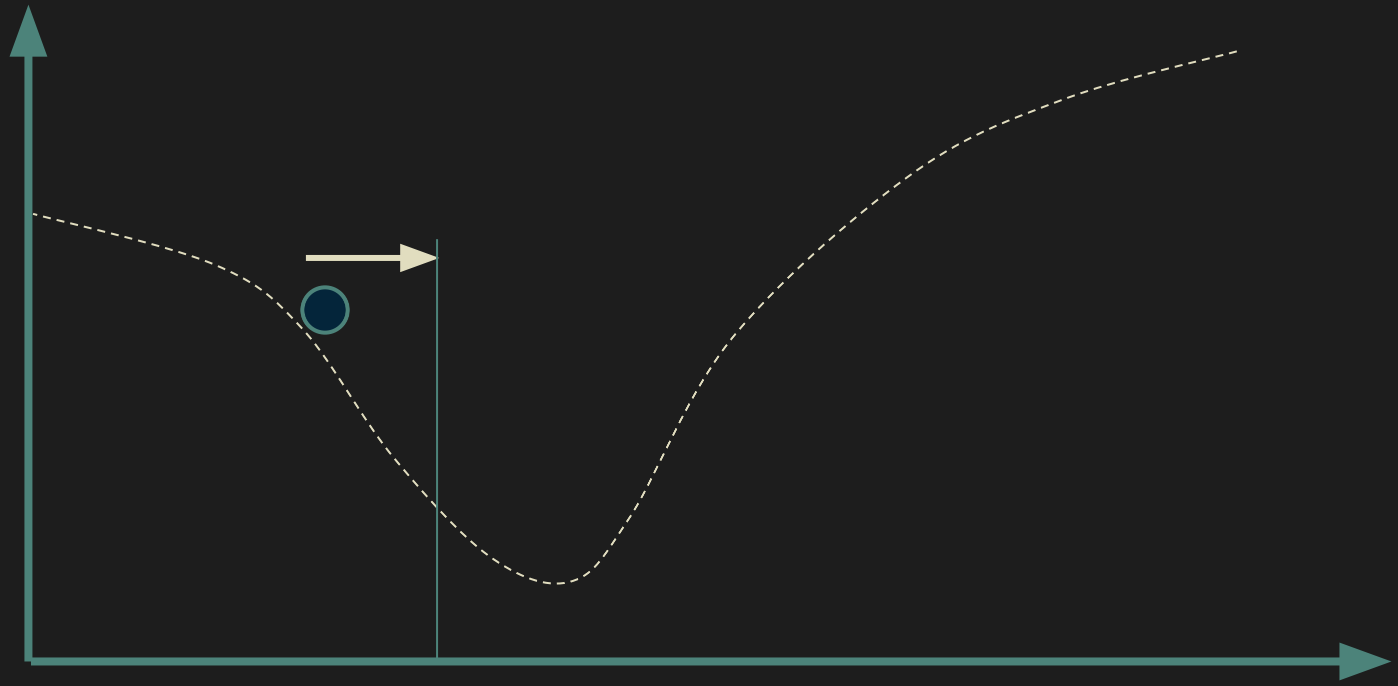
Gradient descent



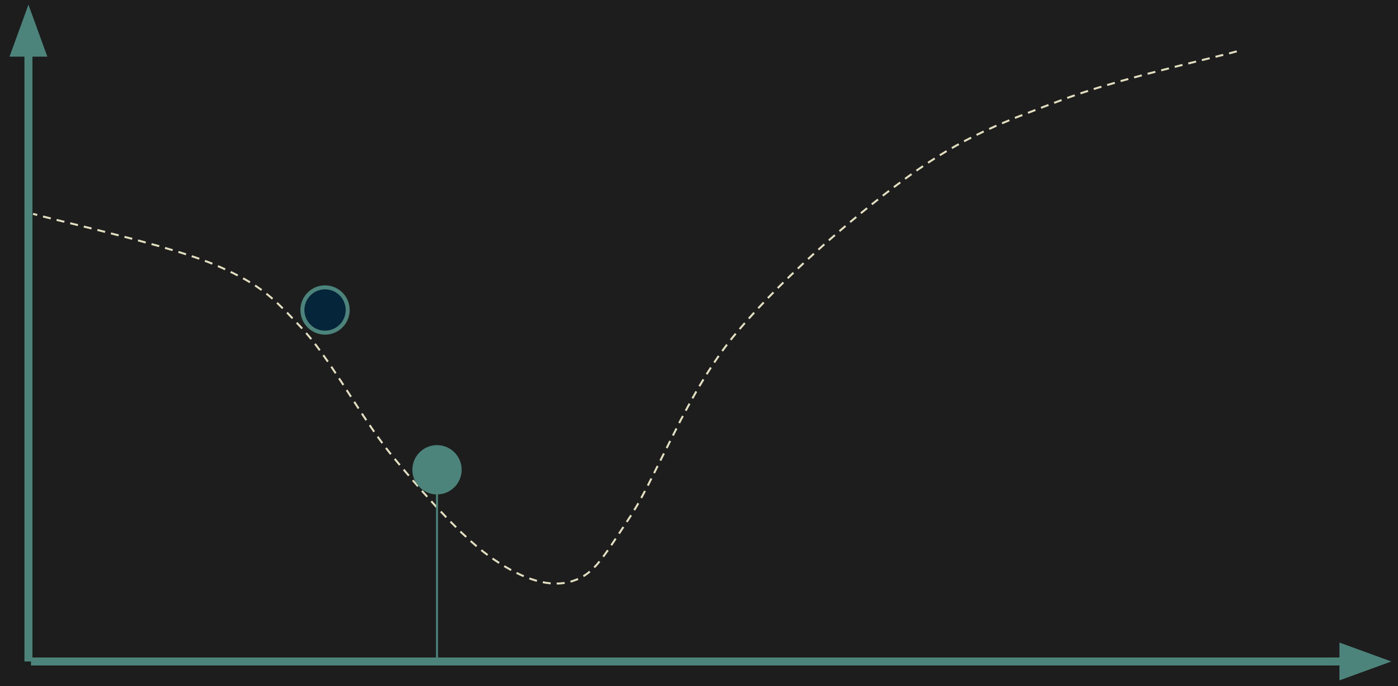
Gradient descent



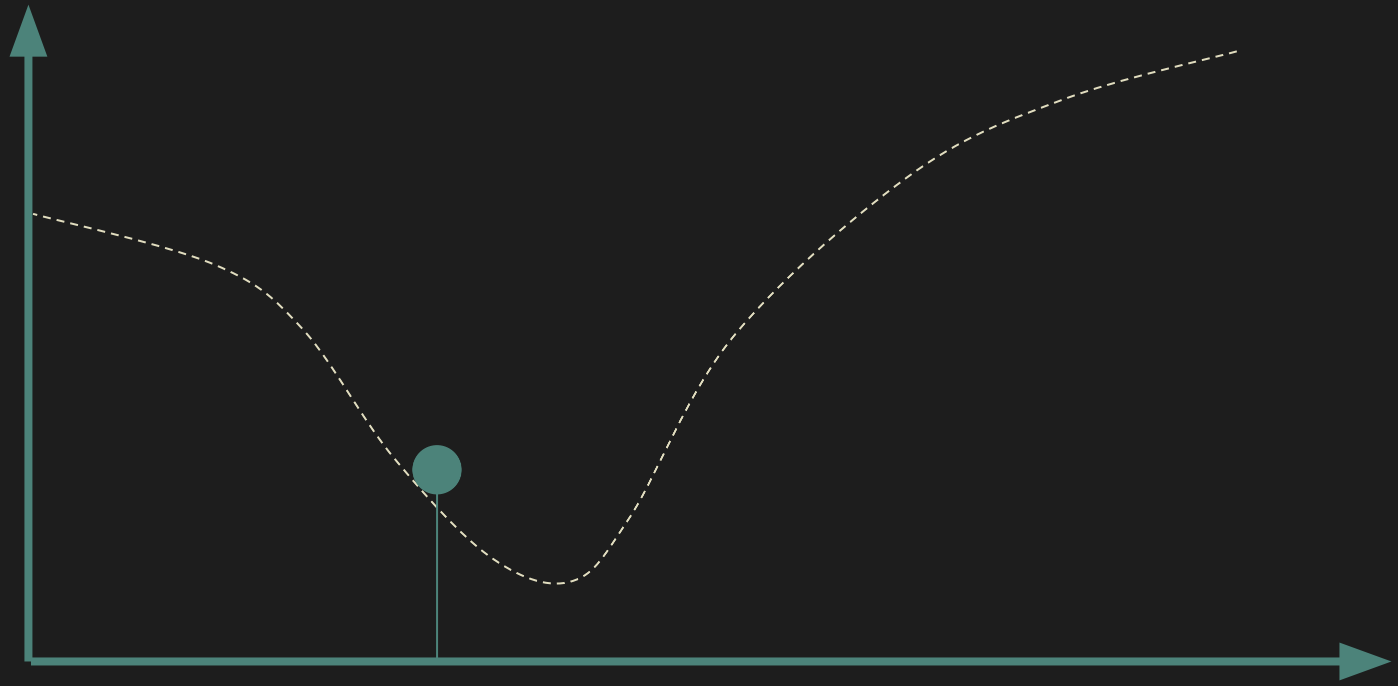
Gradient descent



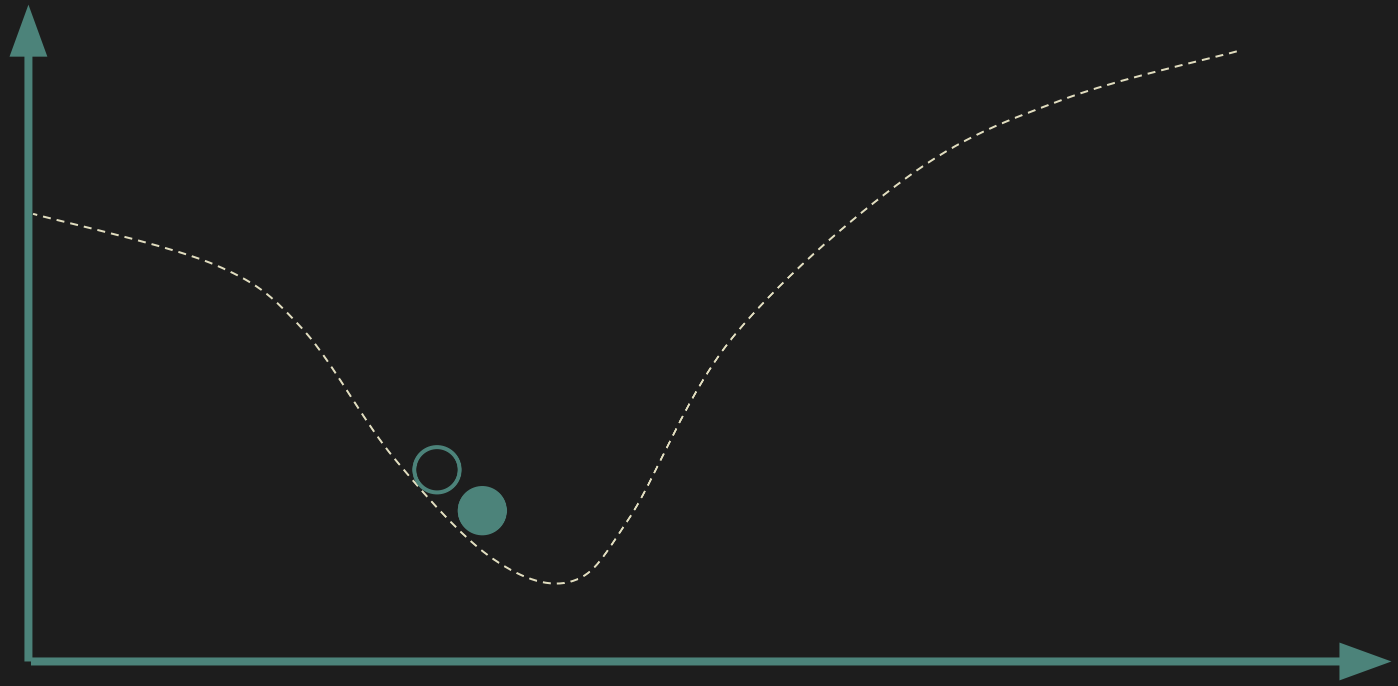
Gradient descent



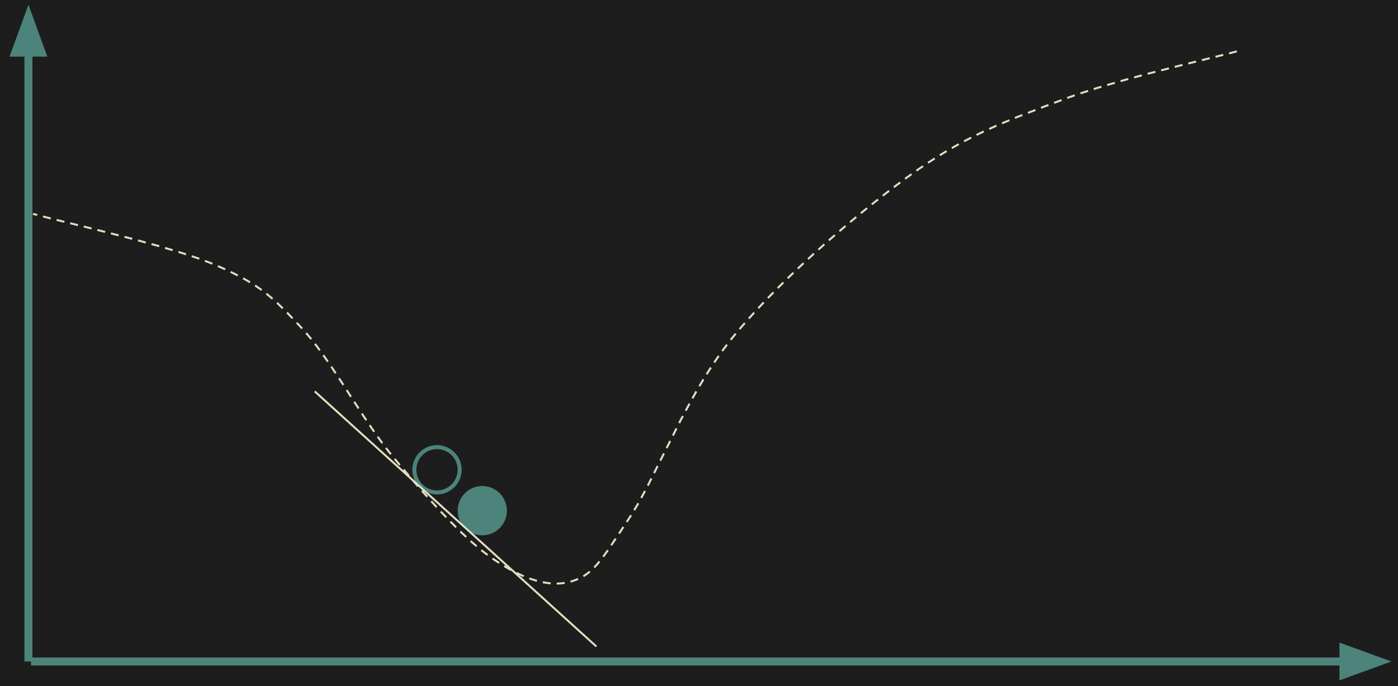
Gradient descent



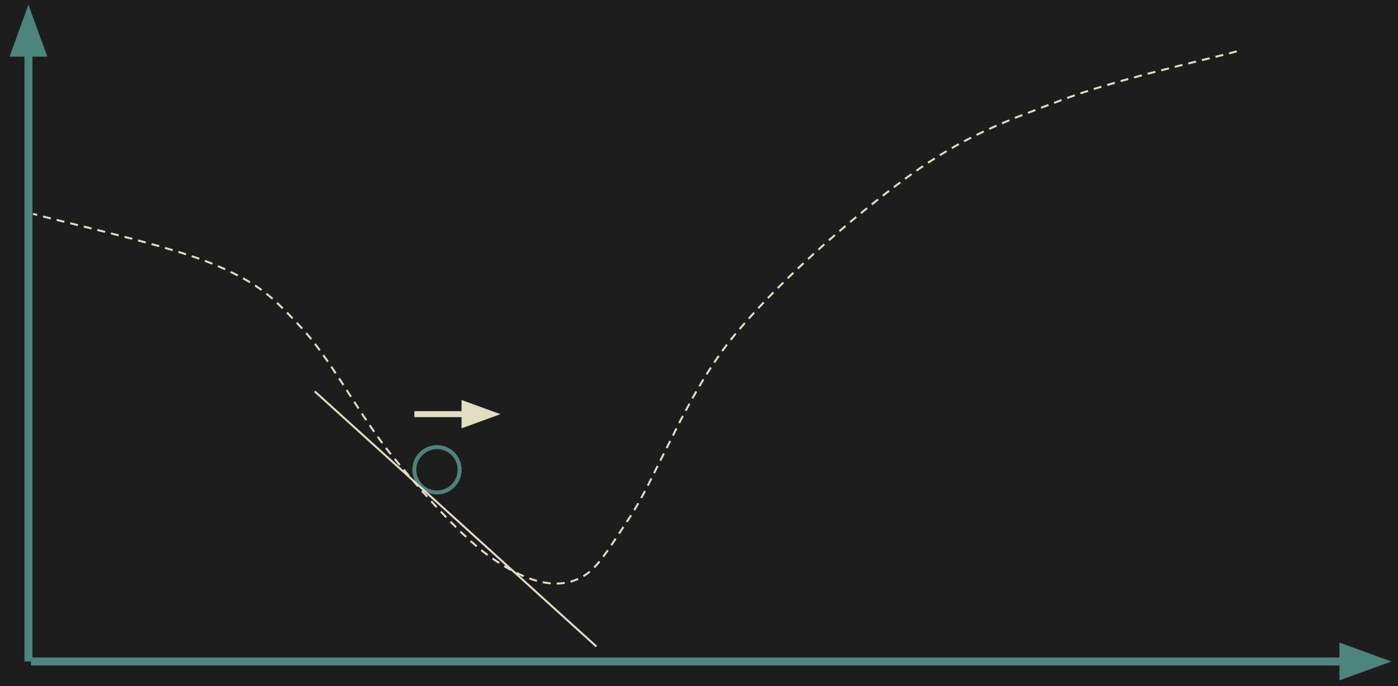
Gradient descent



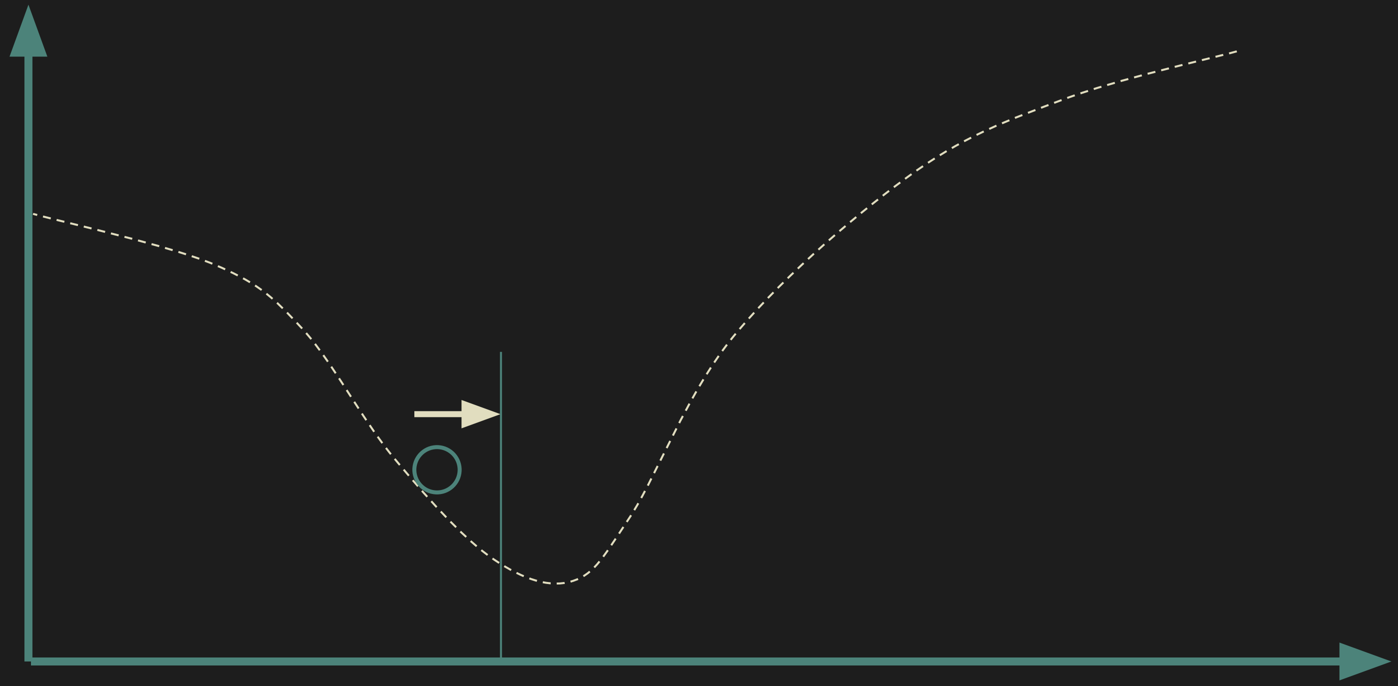
Gradient descent



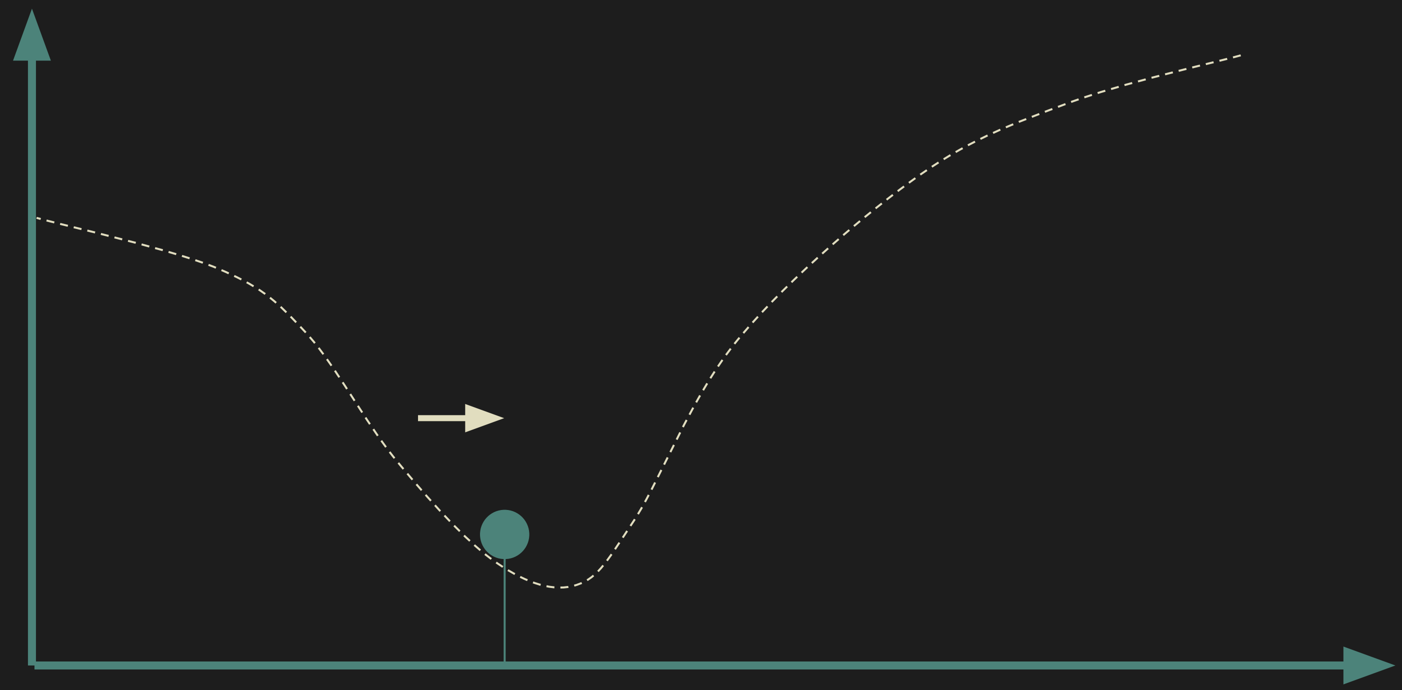
Gradient descent



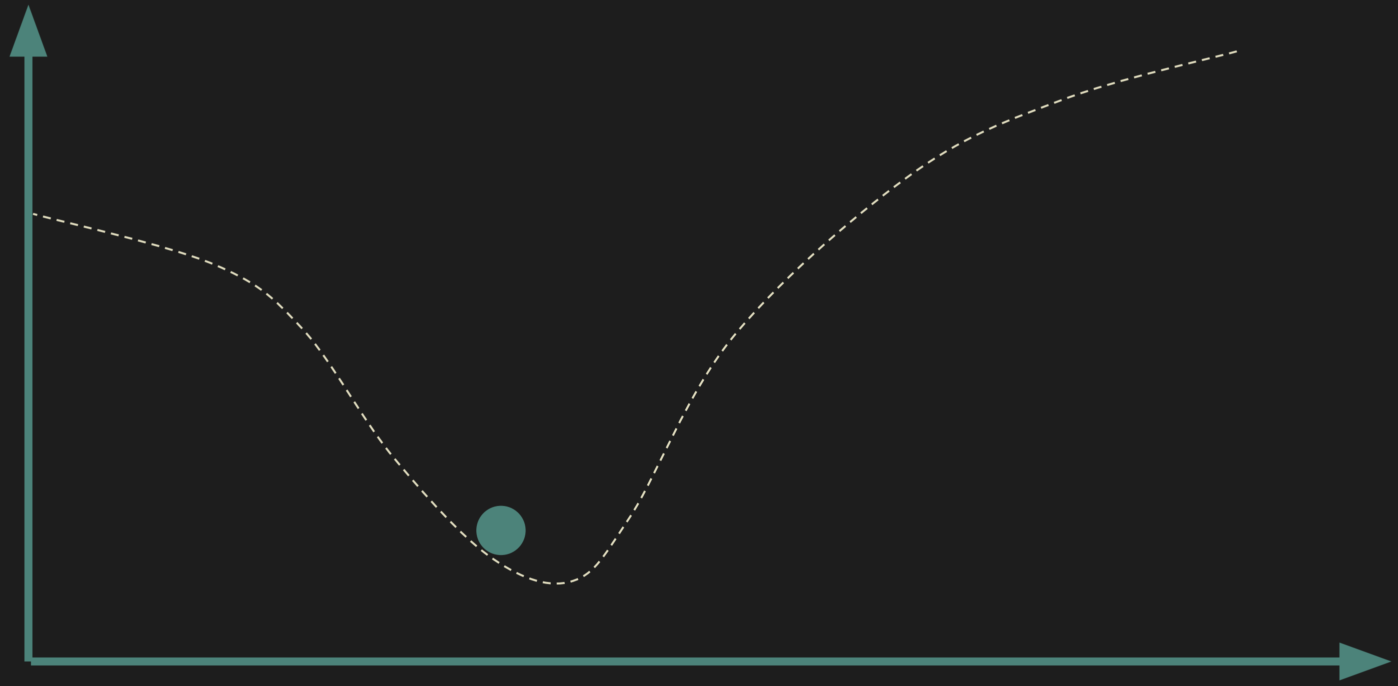
Gradient descent



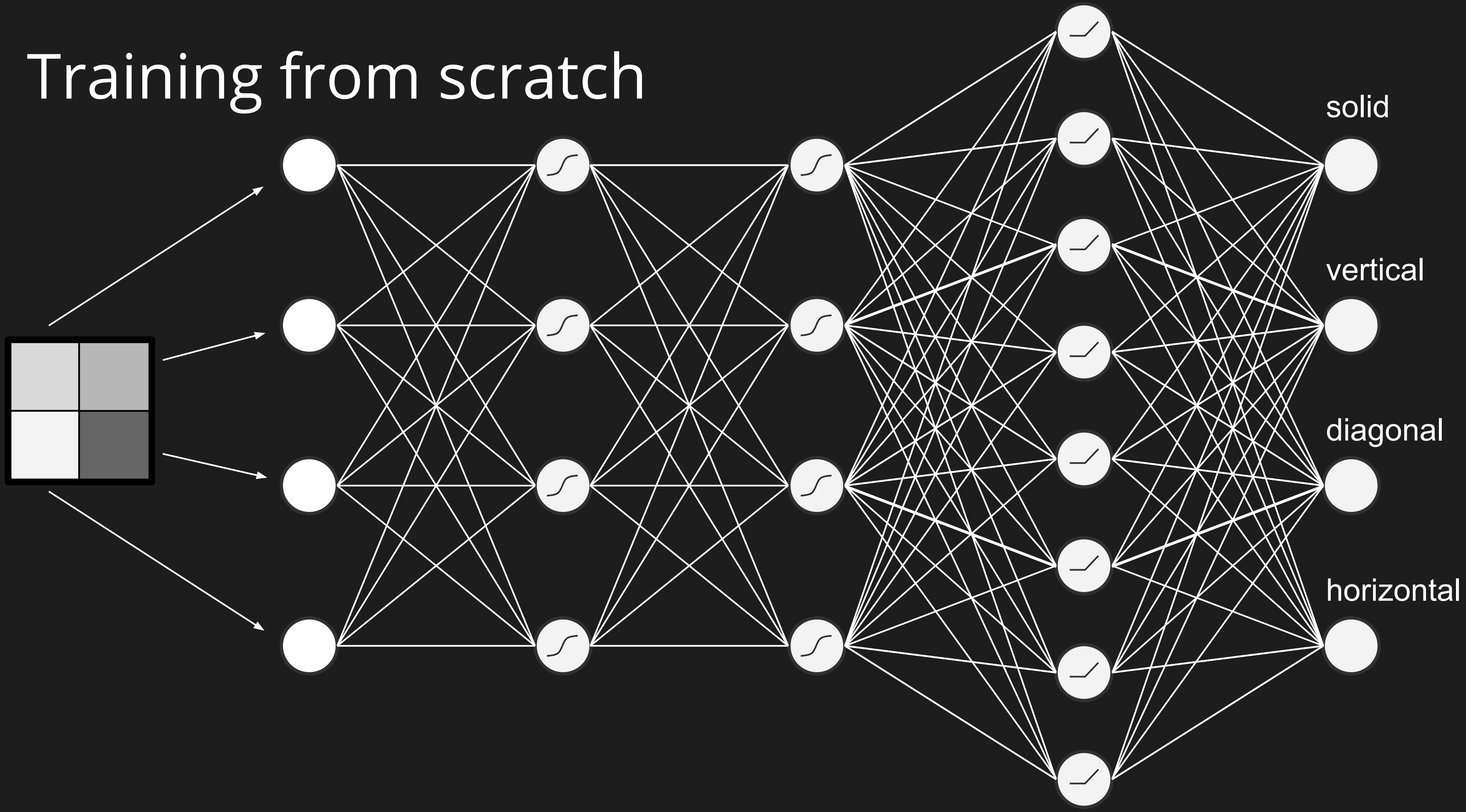
Gradient descent



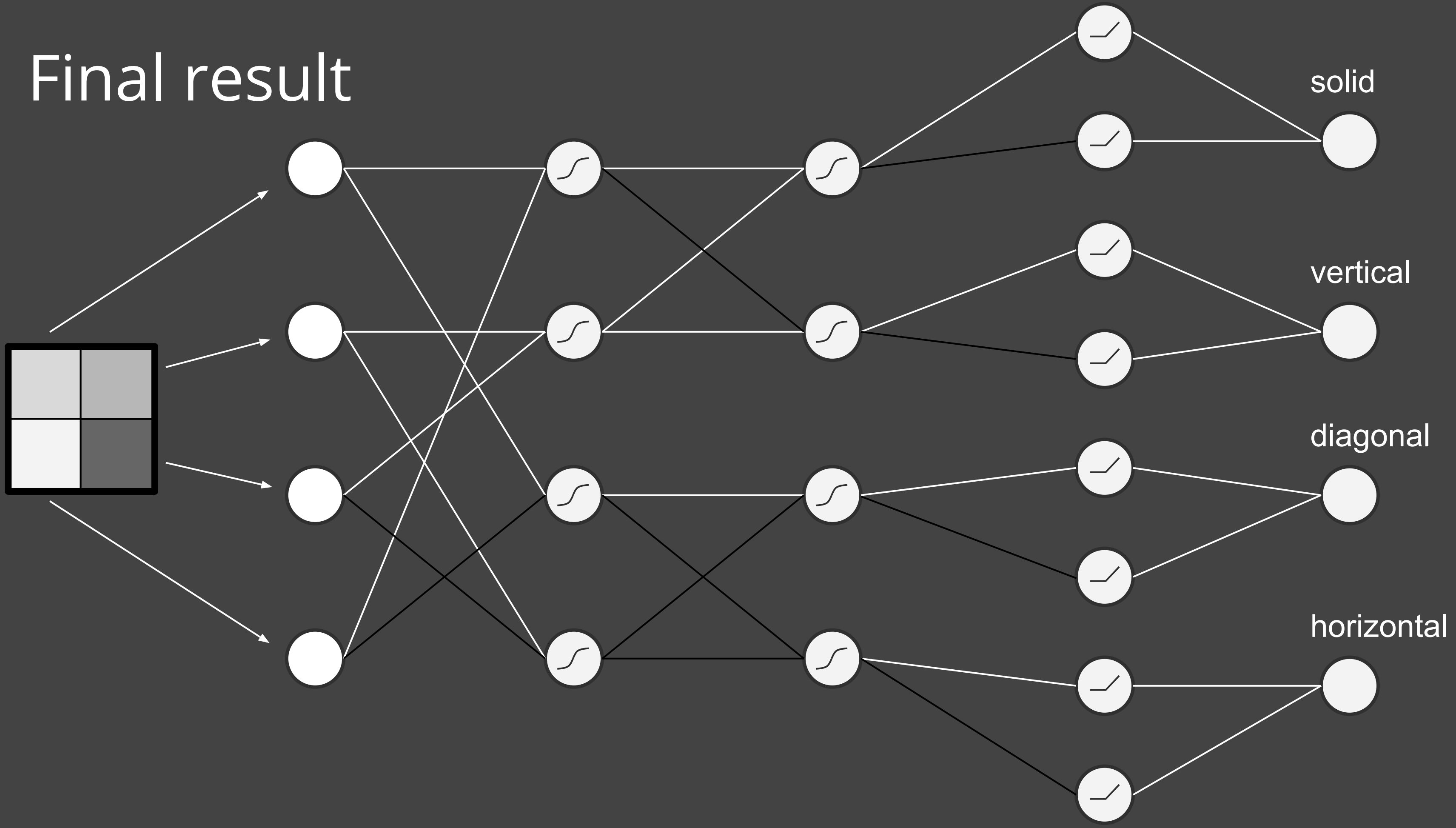
Gradient descent



Training from scratch



Final result



Backpropagation

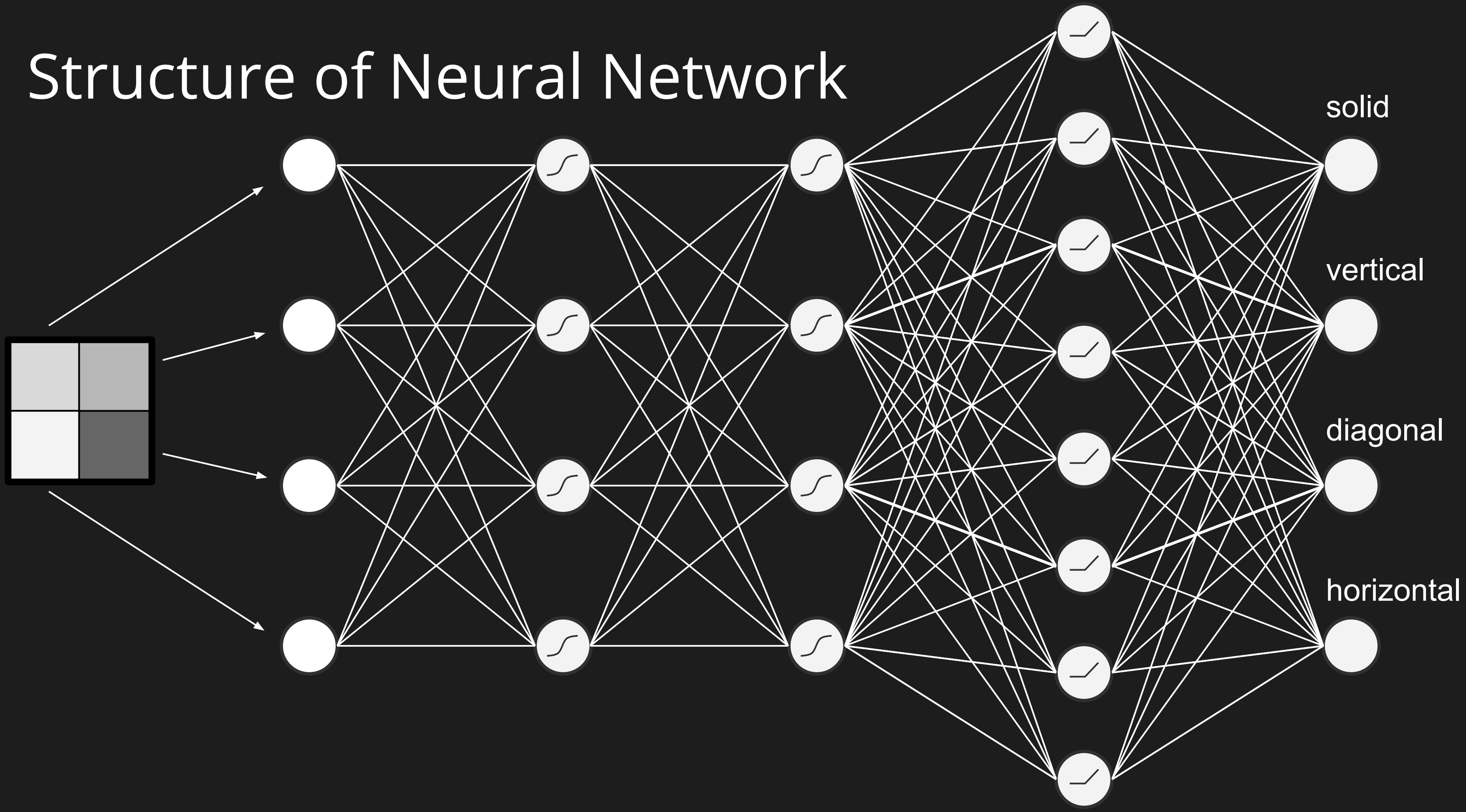
BREAK

RECAP

PASS II

Structure of Neural Network

Structure of Neural Network



Vectorization

$$a_0^{(1)} = \overset{\text{Sigmoid}}{\overset{\downarrow}{\sigma}}\left(w_{0,0}\ a_0^{(0)} + w_{0,1}\ a_1^{(0)} + \cdots + w_{0,n}\ a_n^{(0)} + \underset{\text{Bias}}{\underset{\uparrow}{b_0}}\right)$$

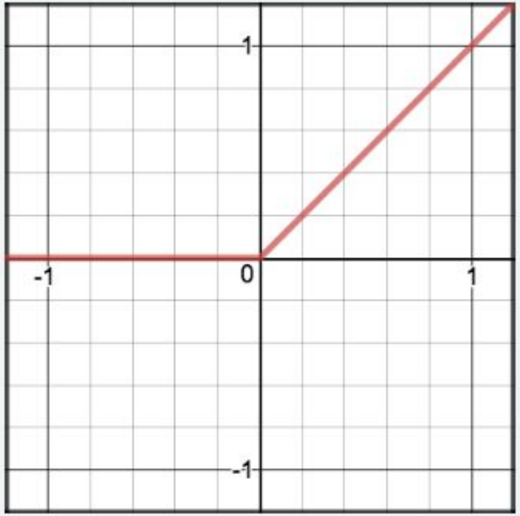
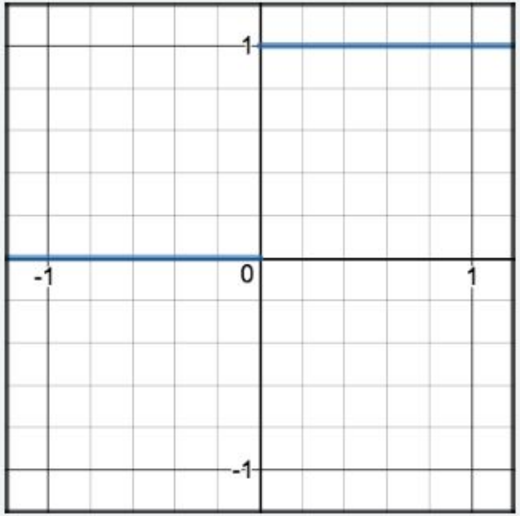
$$\begin{bmatrix} w_{0,0} & w_{0,1} & \cdots & w_{0,n} \\ w_{1,0} & w_{1,1} & \cdots & w_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k,0} & w_{k,1} & \cdots & w_{k,n} \end{bmatrix} \begin{bmatrix} a_0^{(0)} \\ a_1^{(0)} \\ \vdots \\ a_n^{(0)} \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{bmatrix}$$

Fancy representation

$$\mathbf{a}^{(1)} = \sigma\left(\mathbf{W}\mathbf{a}^{(0)} + \mathbf{b}\right)$$

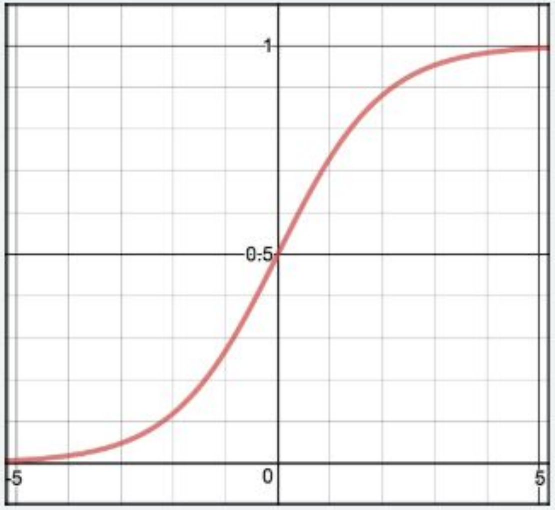
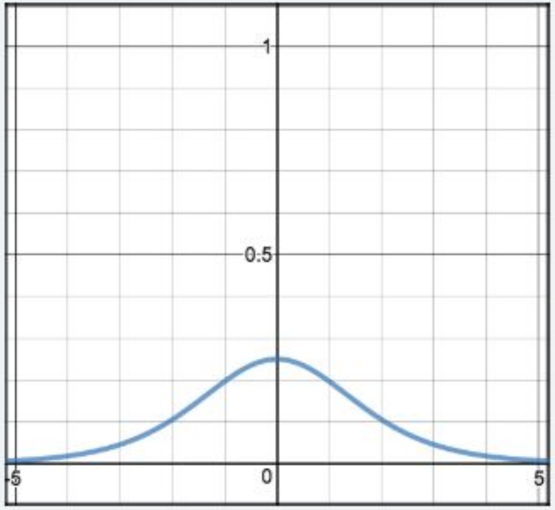
Activation Functions

ReLU

Function	Derivative
$R(z) = \begin{cases} z & z > 0 \\ 0 & z <= 0 \end{cases}$	$R'(z) = \begin{cases} 1 & z > 0 \\ 0 & z < 0 \end{cases}$
	

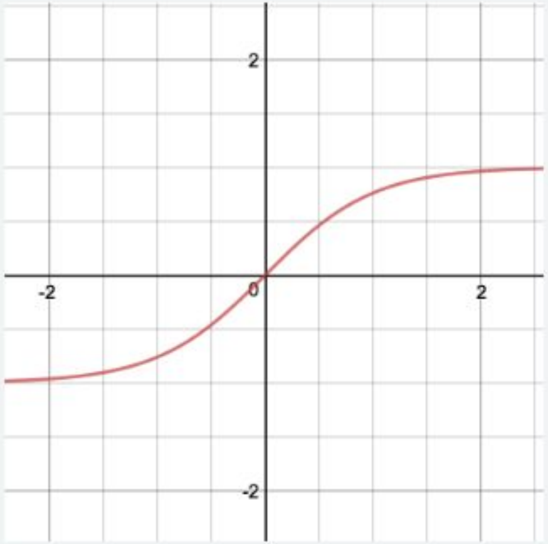
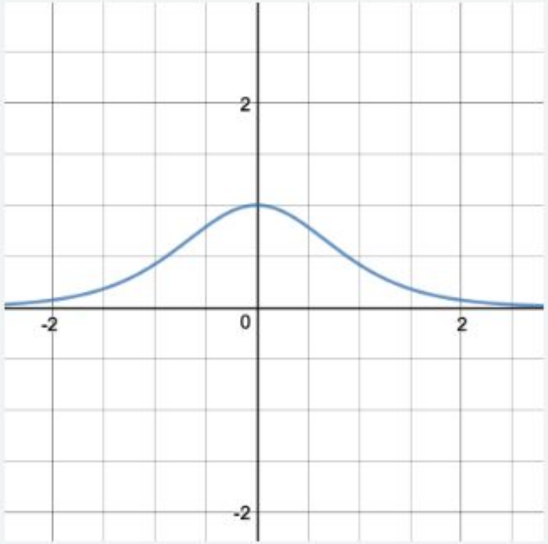
Activation Functions

Sigmoid

Function	Derivative
$S(z) = \dfrac{1}{1 + e^{-z}}$	$S'(z) = S(z) \cdot (1 - S(z))$
	

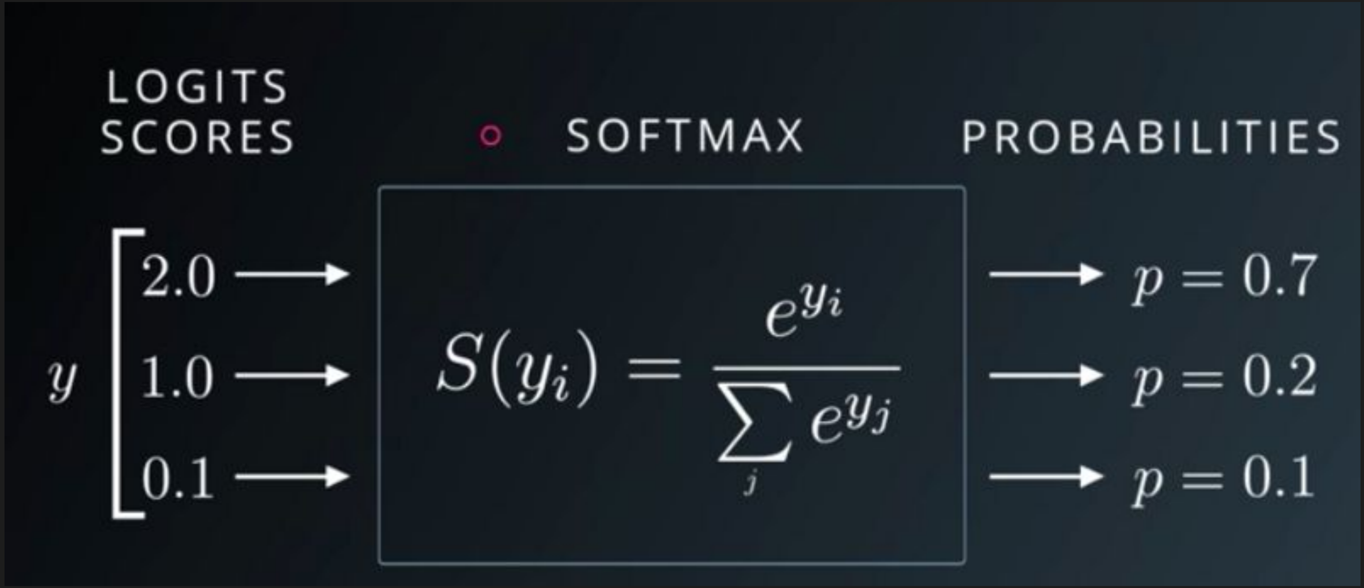
Activation Functions

Tanh

Function	Derivative
$$tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$	$$tanh'(z) = 1 - tanh(z)^2$$
	

Activation Functions

Softmax



Loss Functions

MAE

Loss Functions

MSE

Loss Functions

Binary Cross-Entropy

$$-(y\log(p) + (1-y)\log(1-p))$$

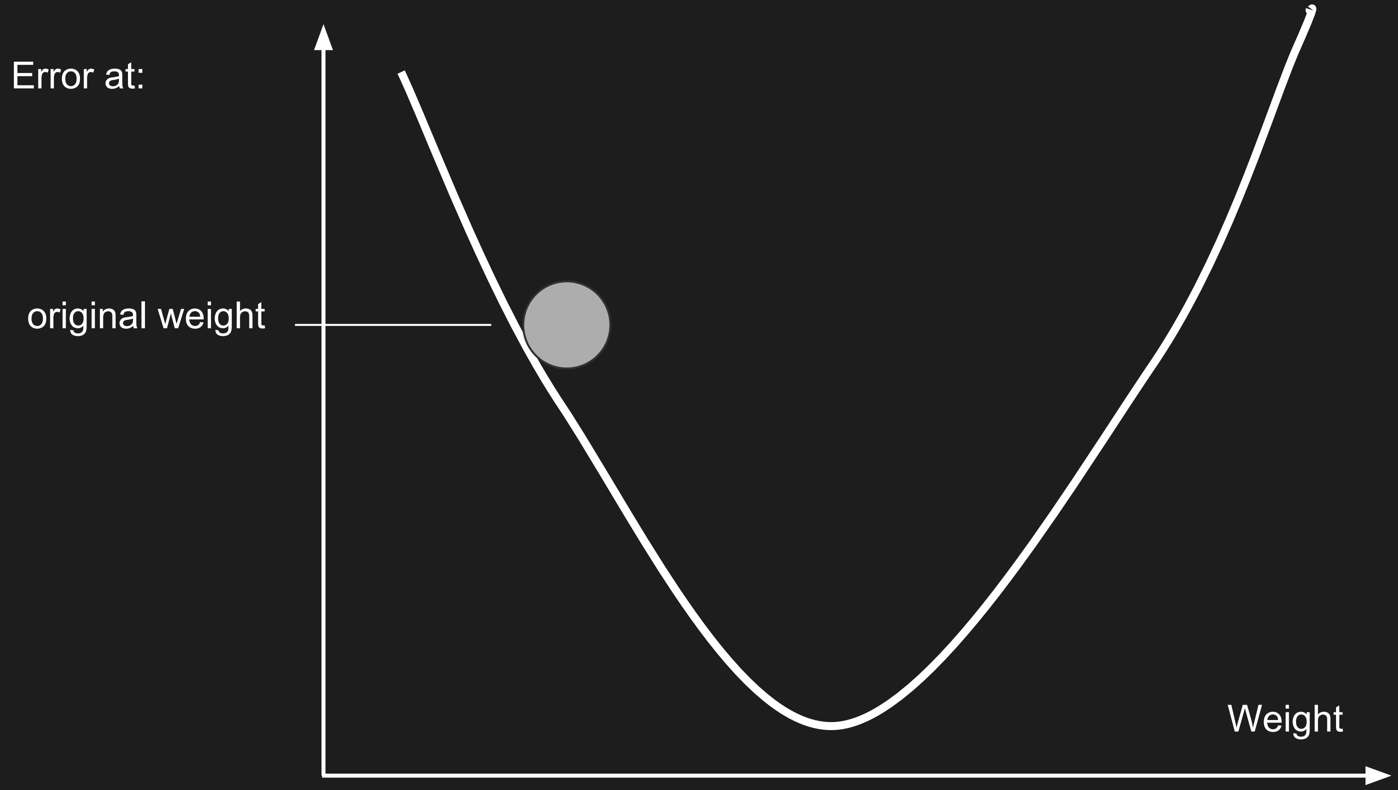
Loss Functions

Multi-Class Cross-Entropy (Categorical)

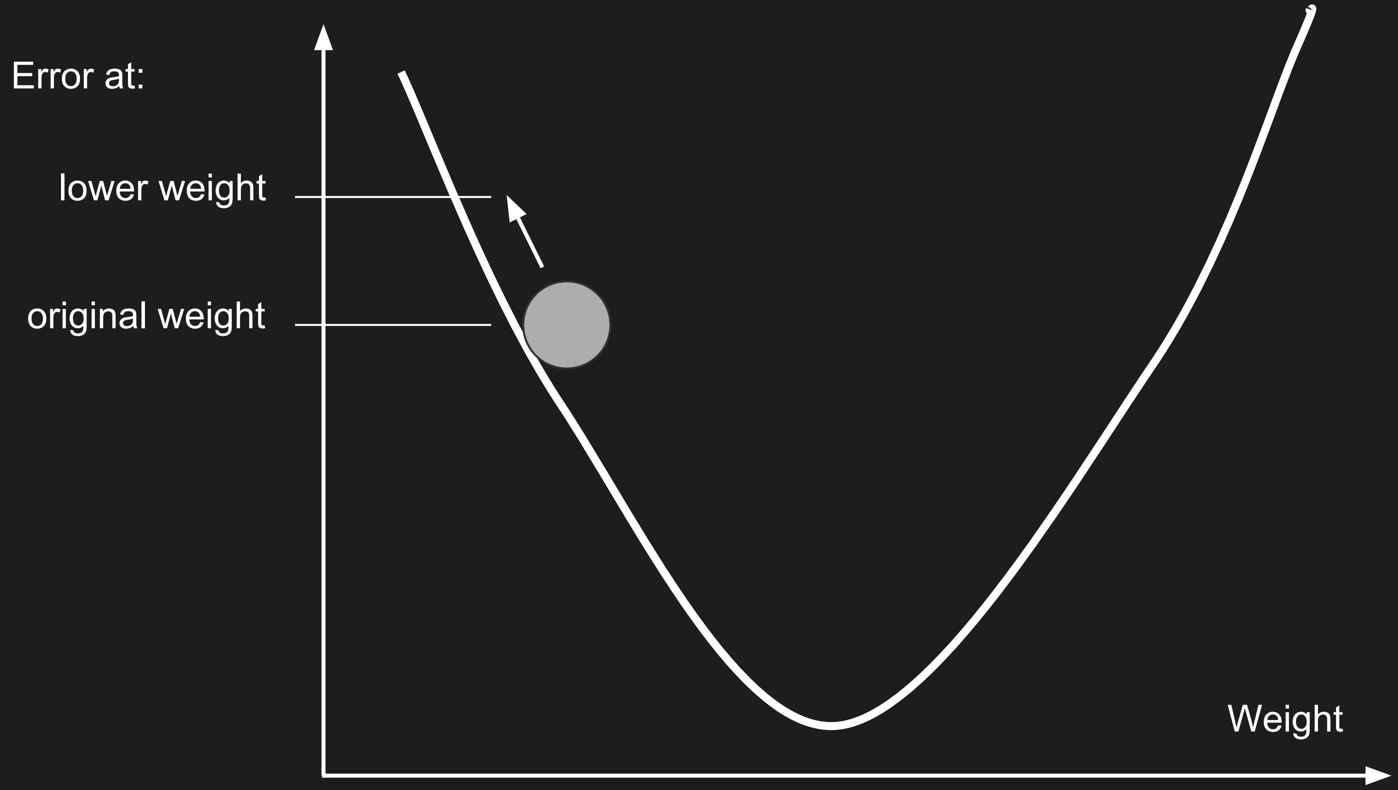
$$-\sum_{c=1}^{M} y_{o,c} \log(p_{o,c})$$

Gradient Descent
- ft. Lil Math

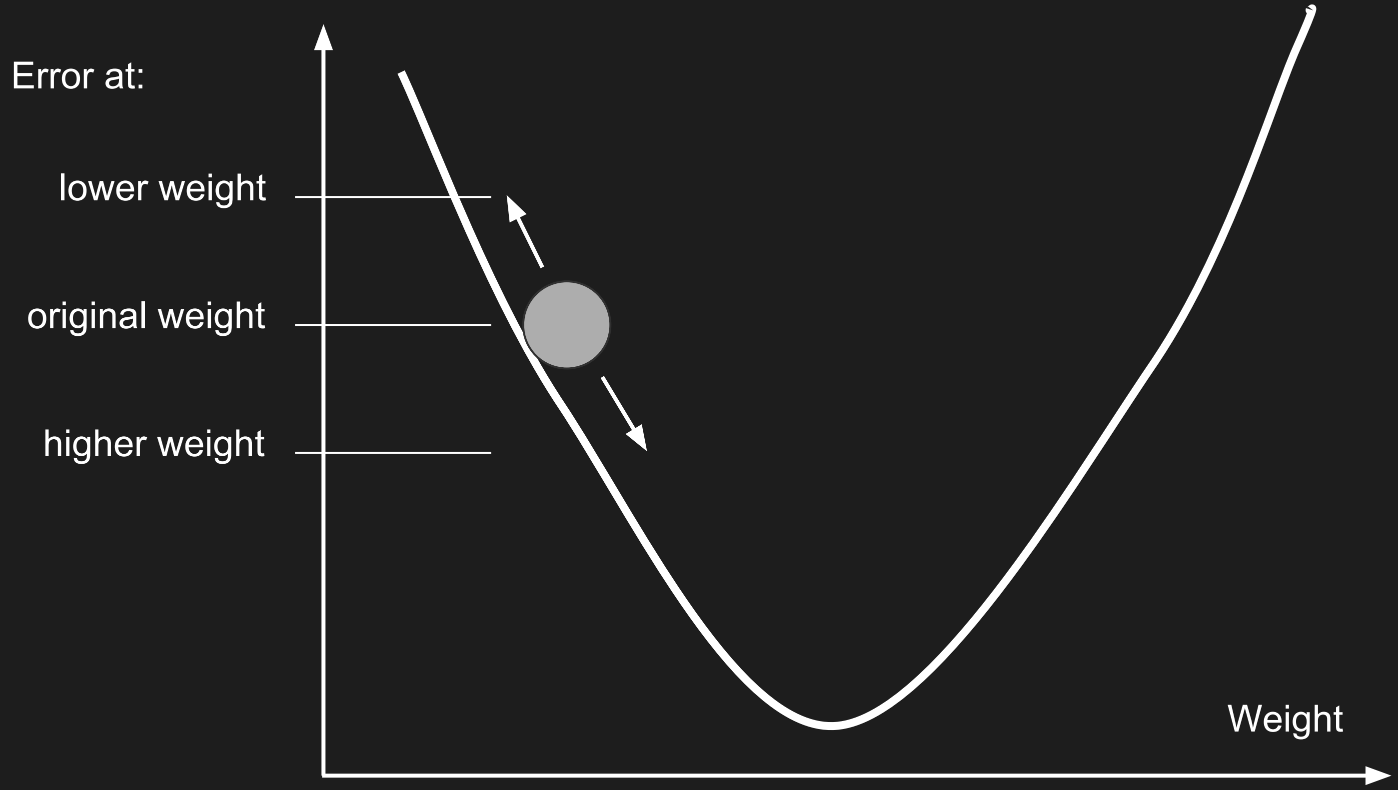
Learn all the weights: Gradient descent



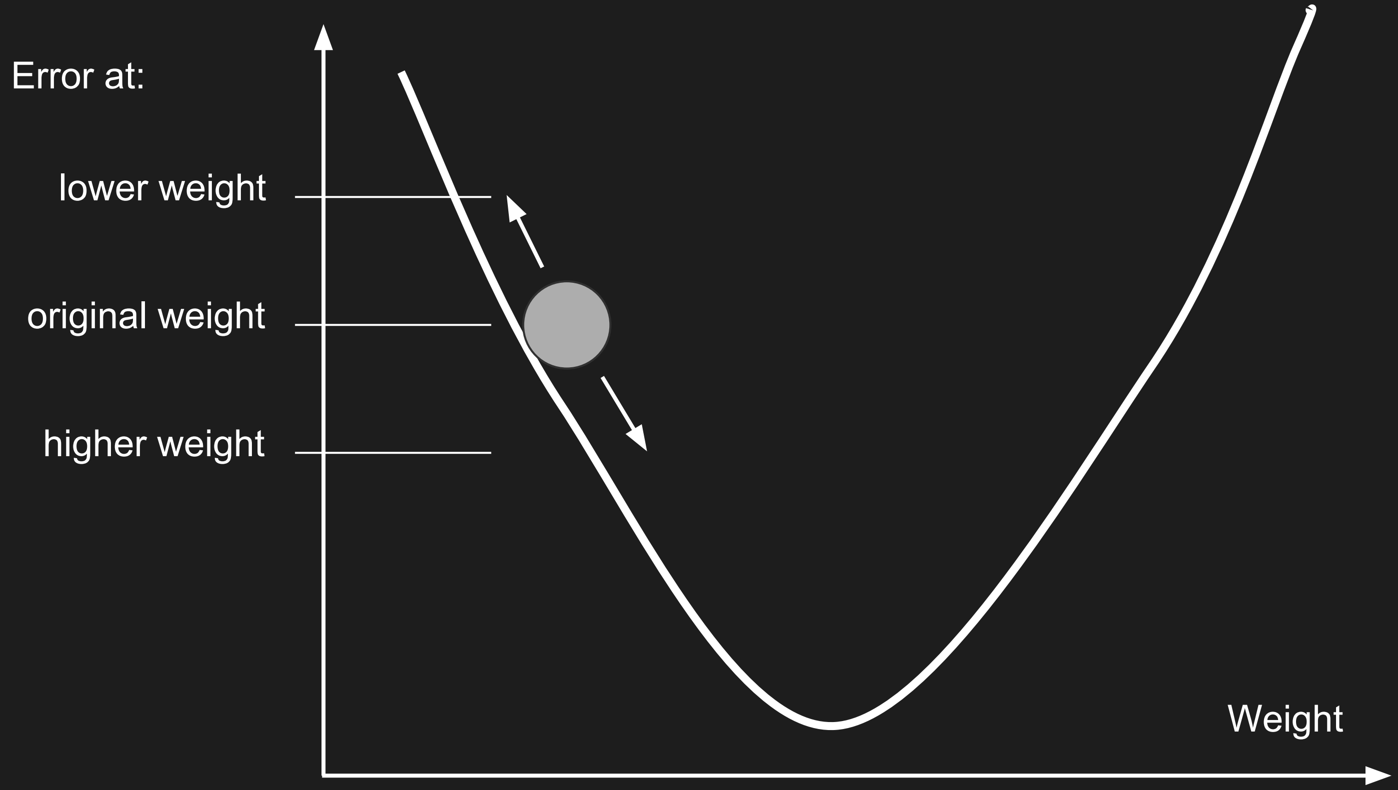
Learn all the weights: Gradient descent



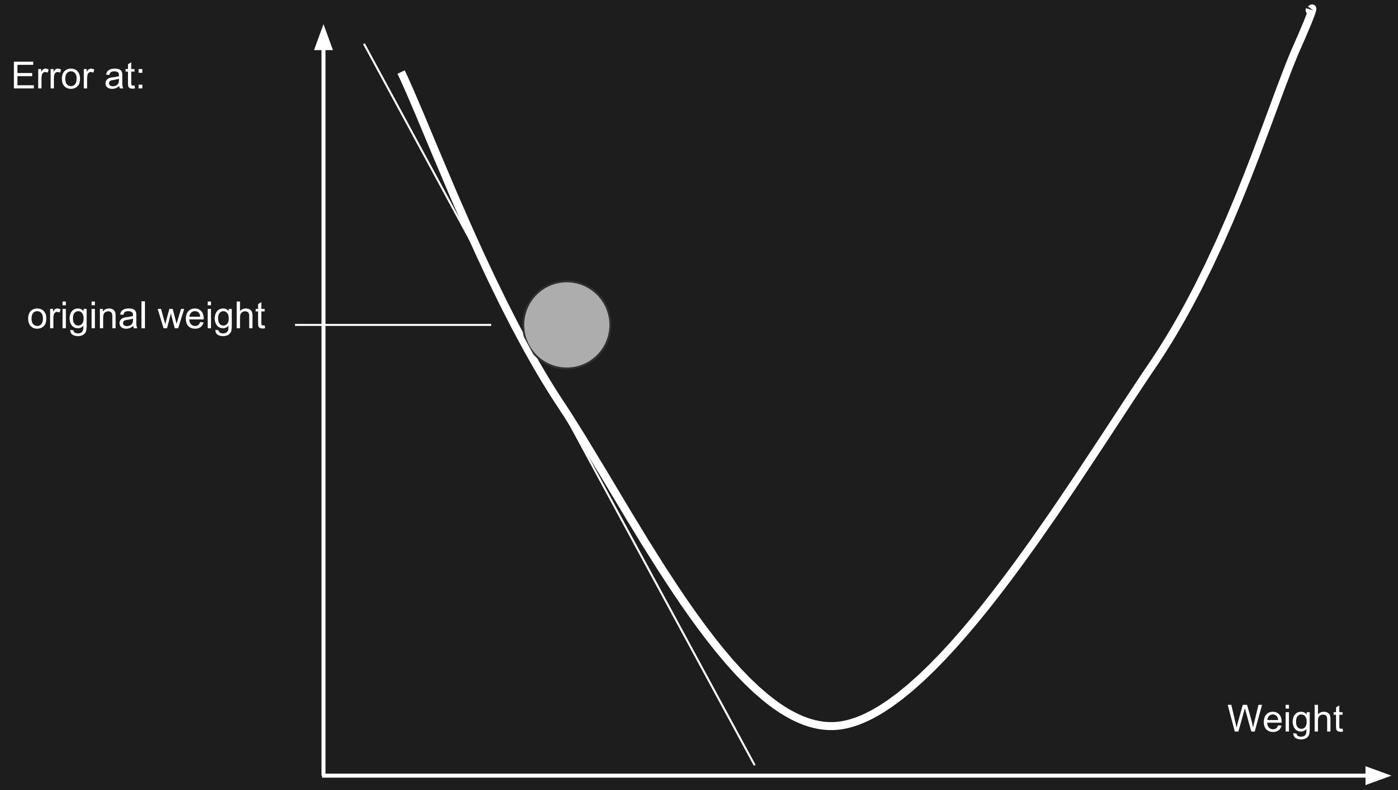
Learn all the weights: Gradient descent



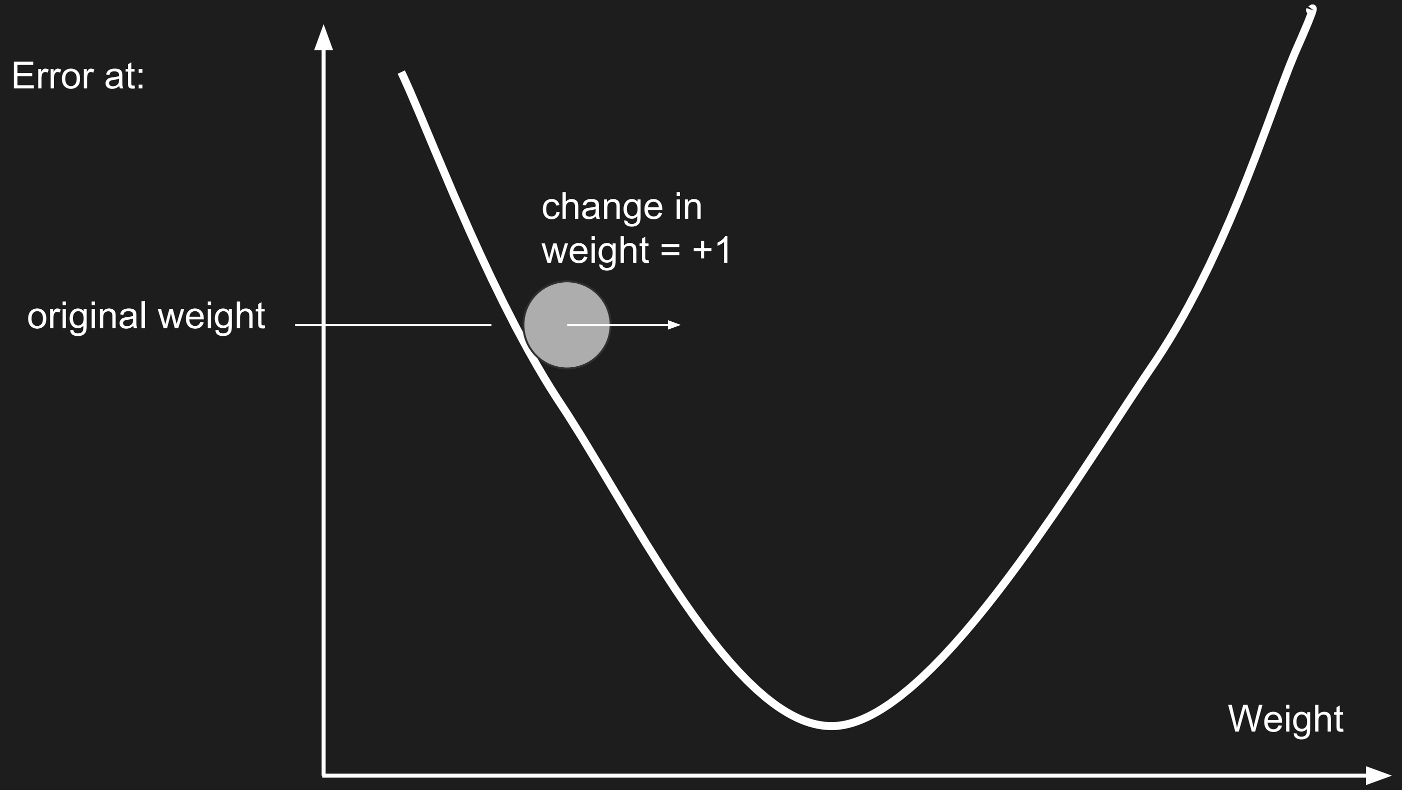
Numerically calculating the gradient is expensive



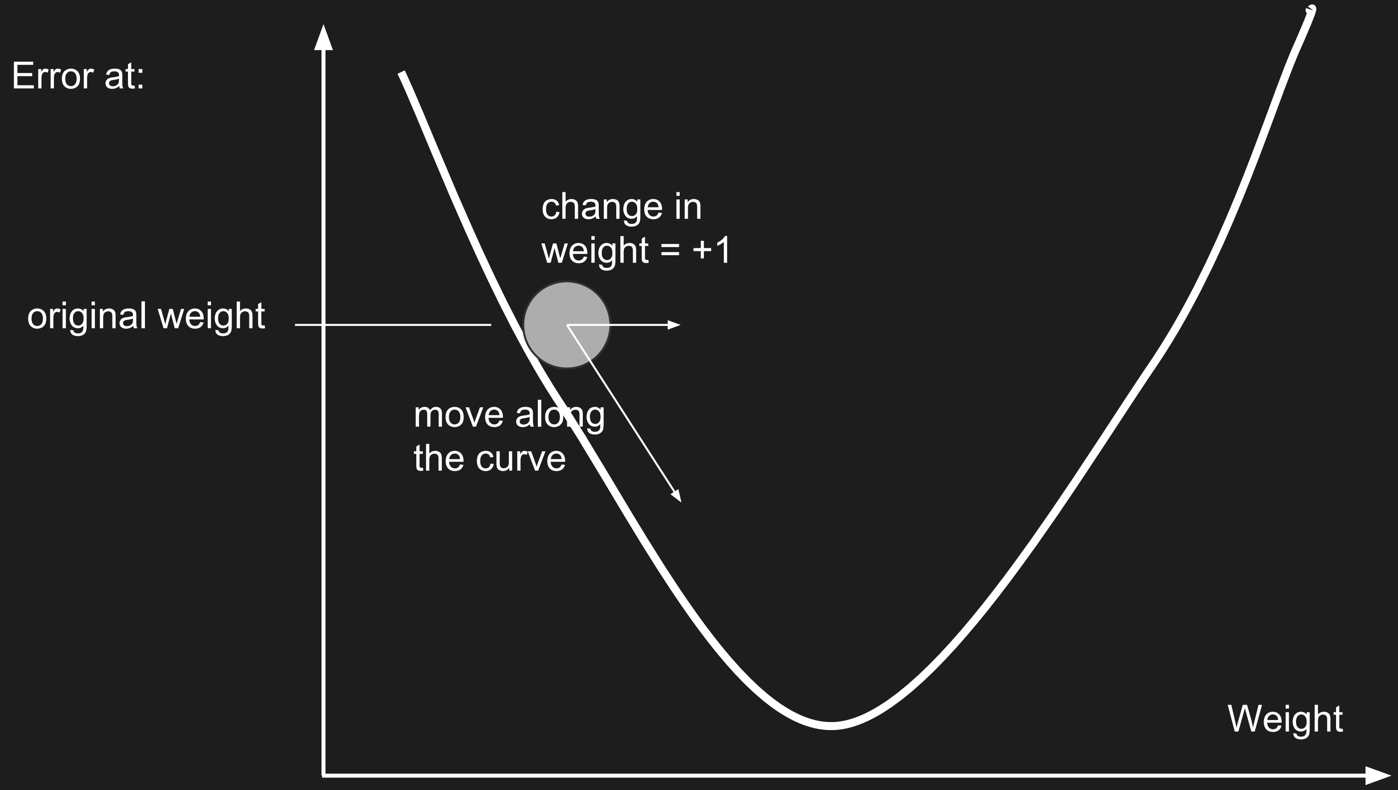
Calculate the gradient (slope) directly



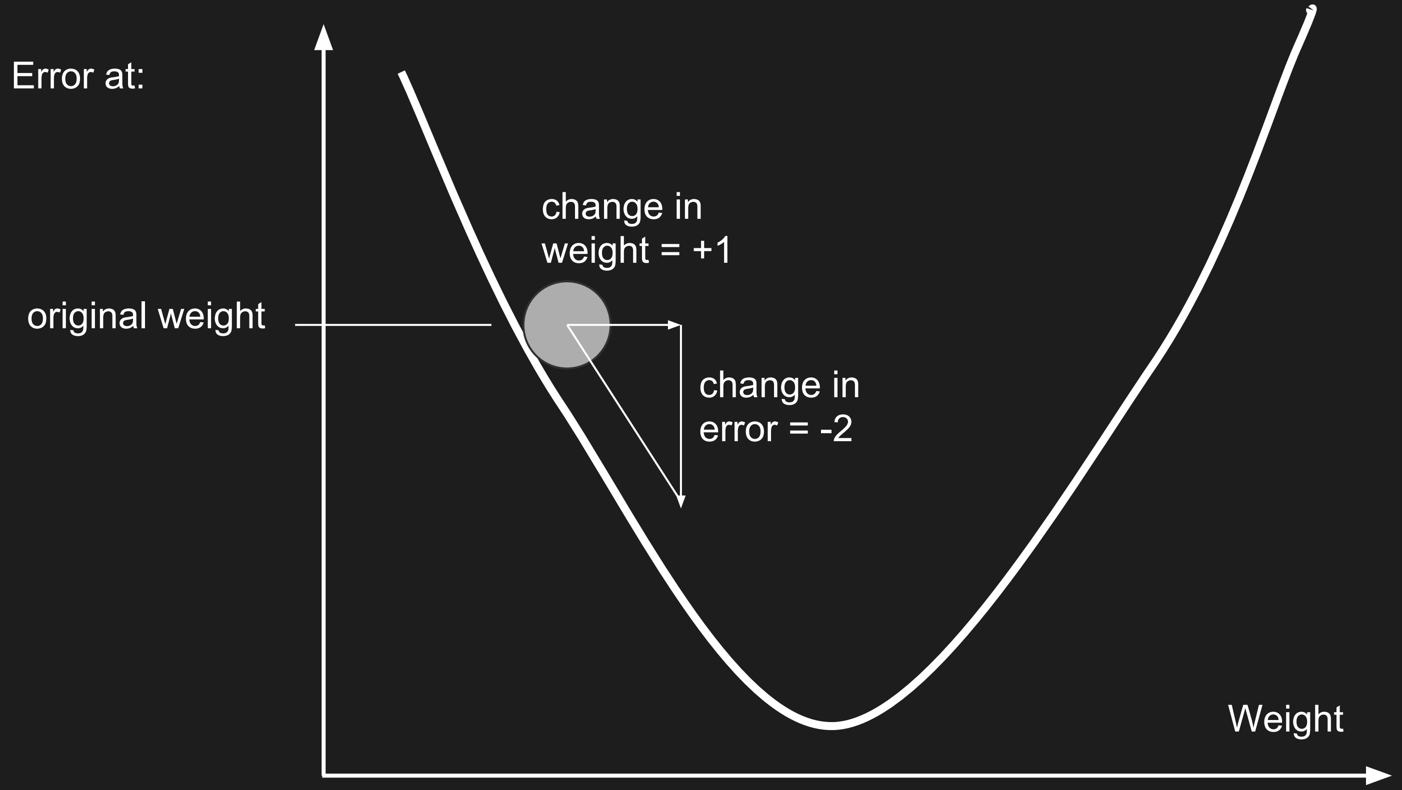
Slope



Slope

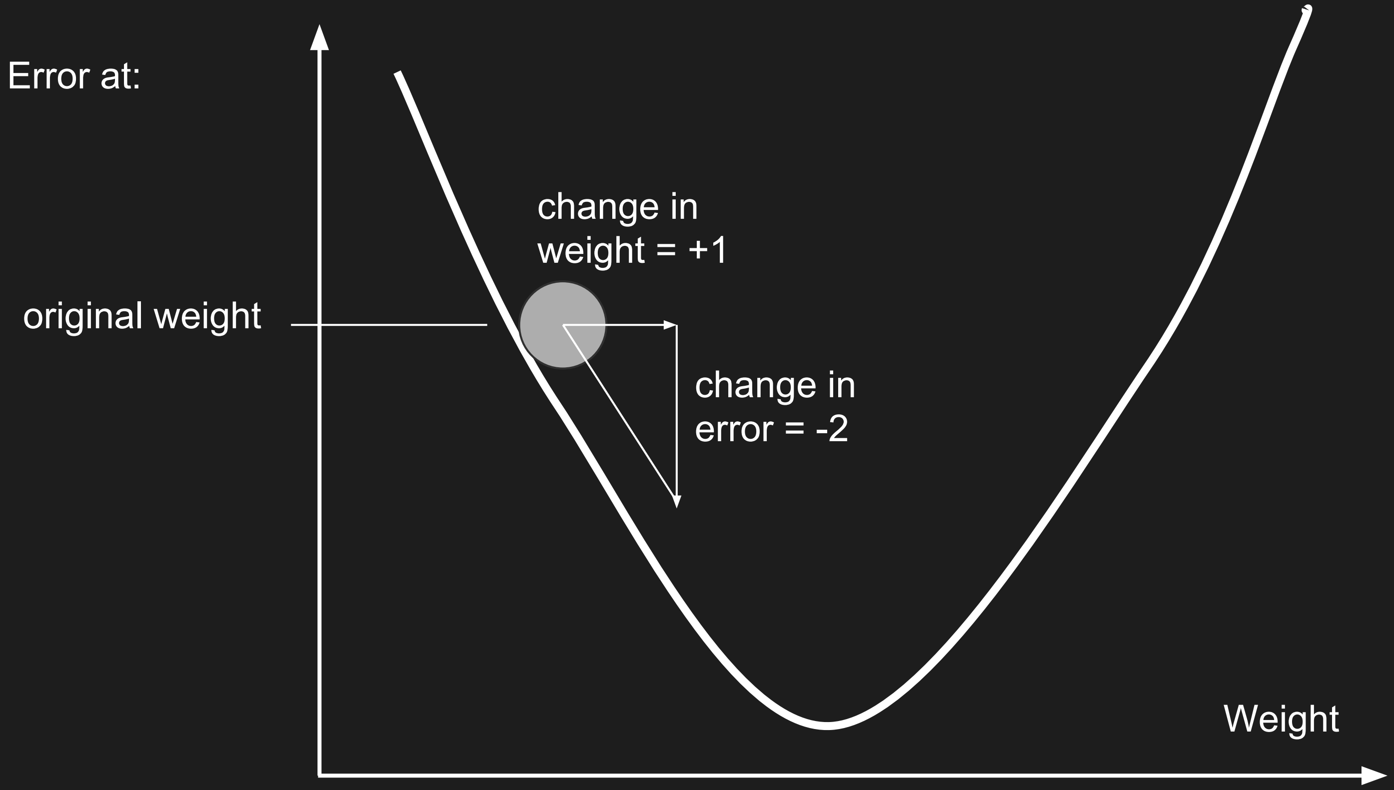

Error at:
original weight
change in
weight = +1
move along
the curve
Weight

Slope


Error at:
change in
weight = +1
original weight
change in
error = -2
Weight

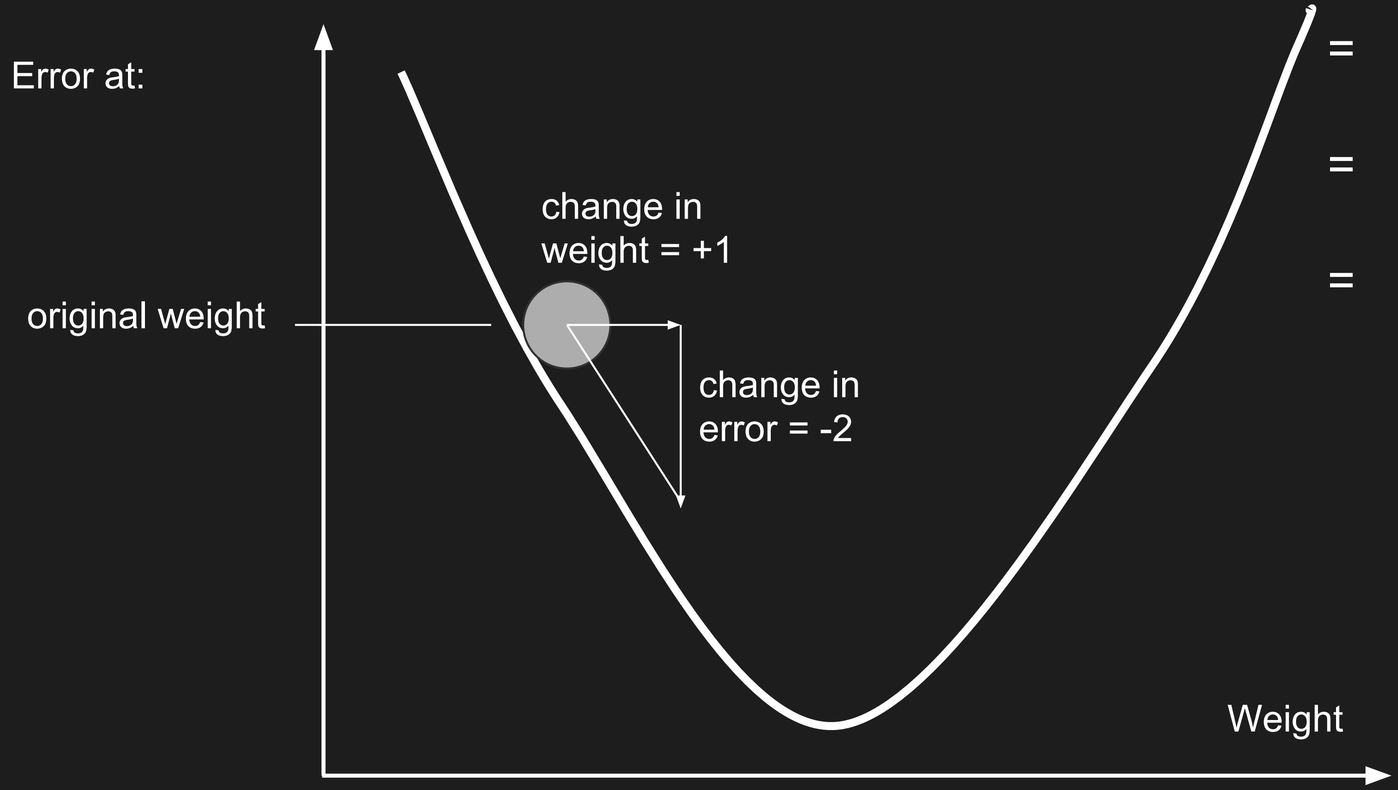
Slope

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

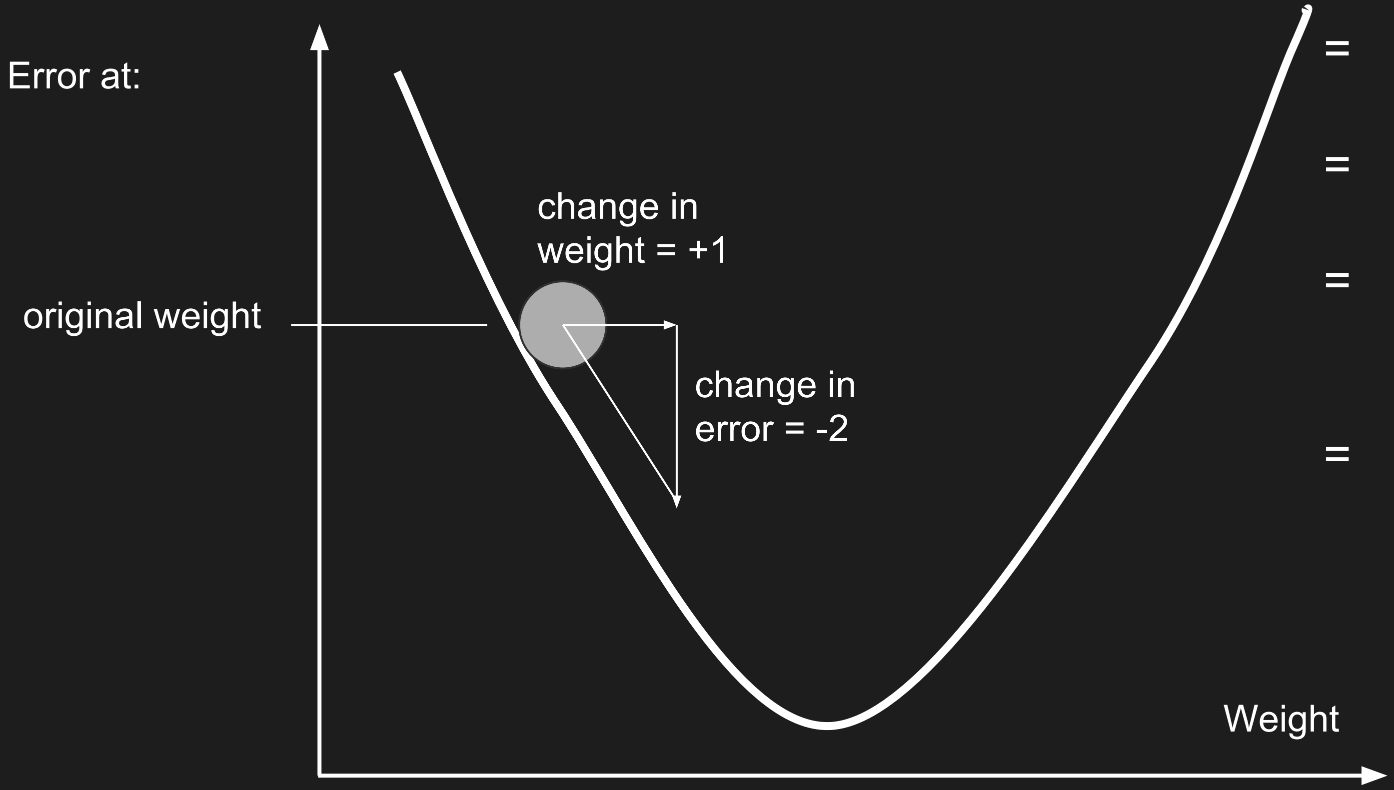


Slope

$$\text{slope} = \frac{\text{change in error}}{\text{change in weight}} = \frac{\Delta\ \text{error}}{\Delta\ \text{weight}} = \frac{d(\text{error})}{d(\text{weight})} = \frac{\partial e}{\partial w}$$



Slope

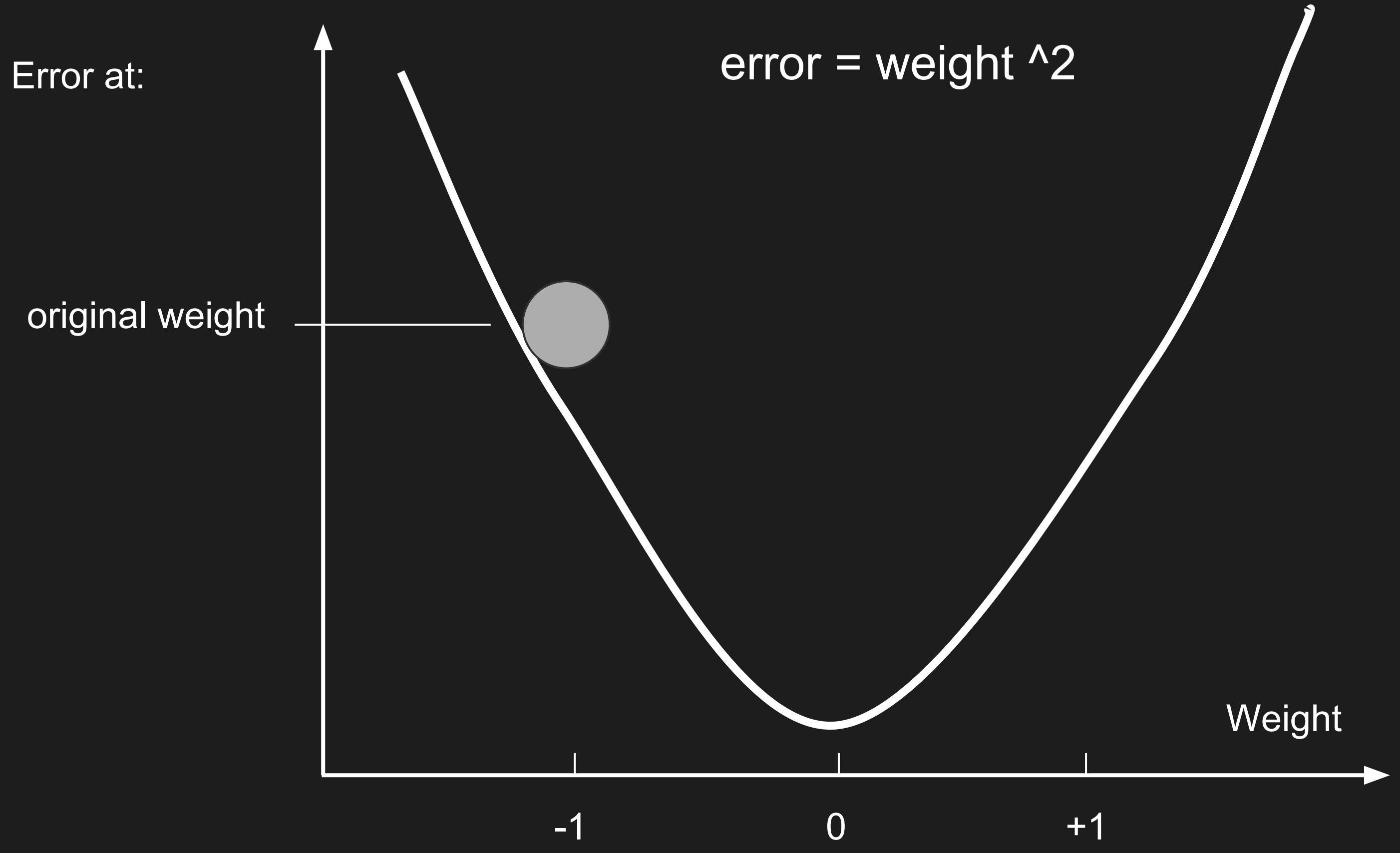


$$\text{slope} = \frac{\text{change in error}}{\text{change in weight}} = \frac{\Delta\ \text{error}}{\Delta\ \text{weight}} = \frac{d(\text{error})}{d(\text{weight})} = \frac{\partial e}{\partial w} = \frac{-2}{+1} = -2$$

Slope

You have to know your error function.
For example:

error = weight ^2

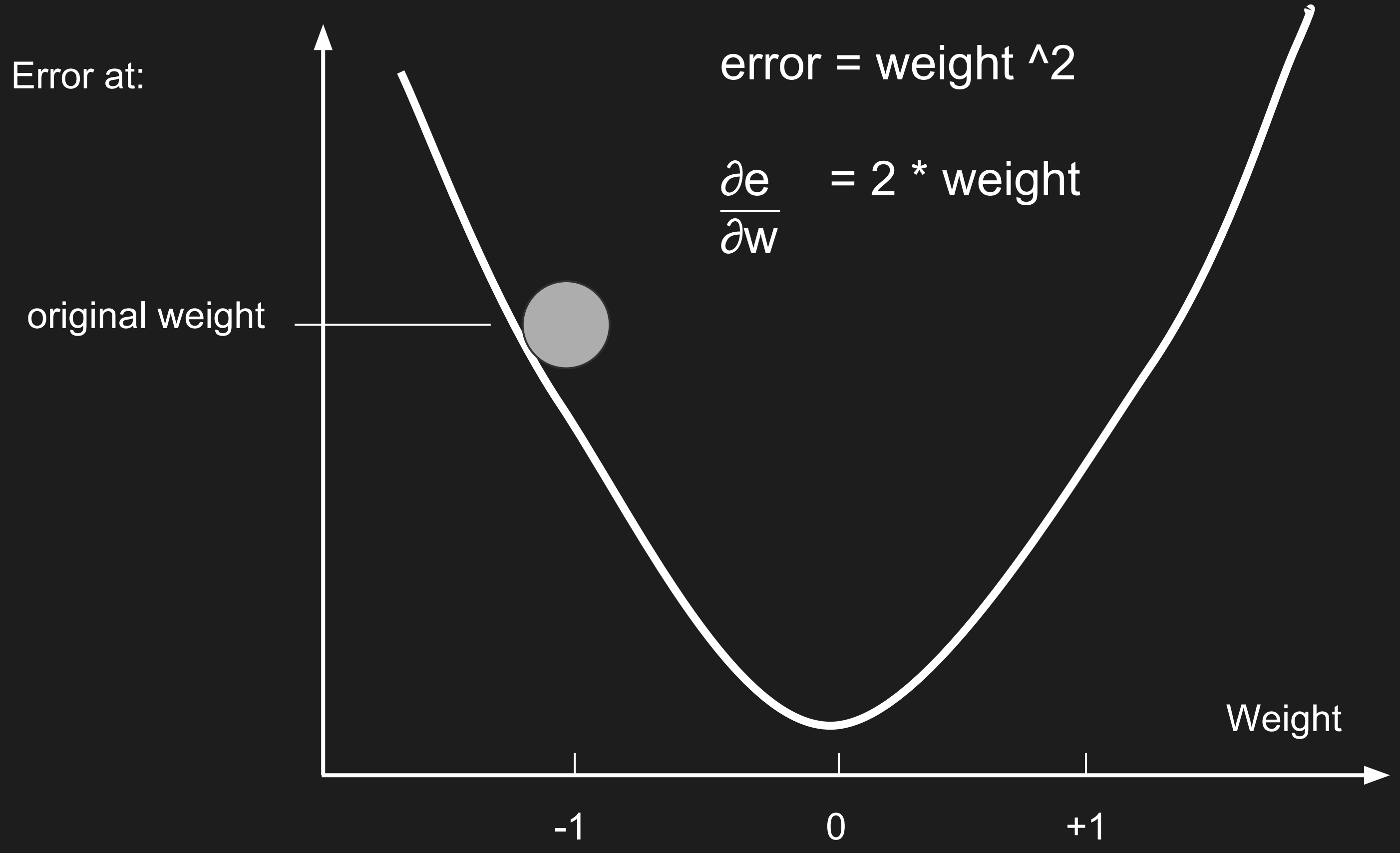


Slope

You have to know your error function.
For example:

error = weight ^2

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



Slope

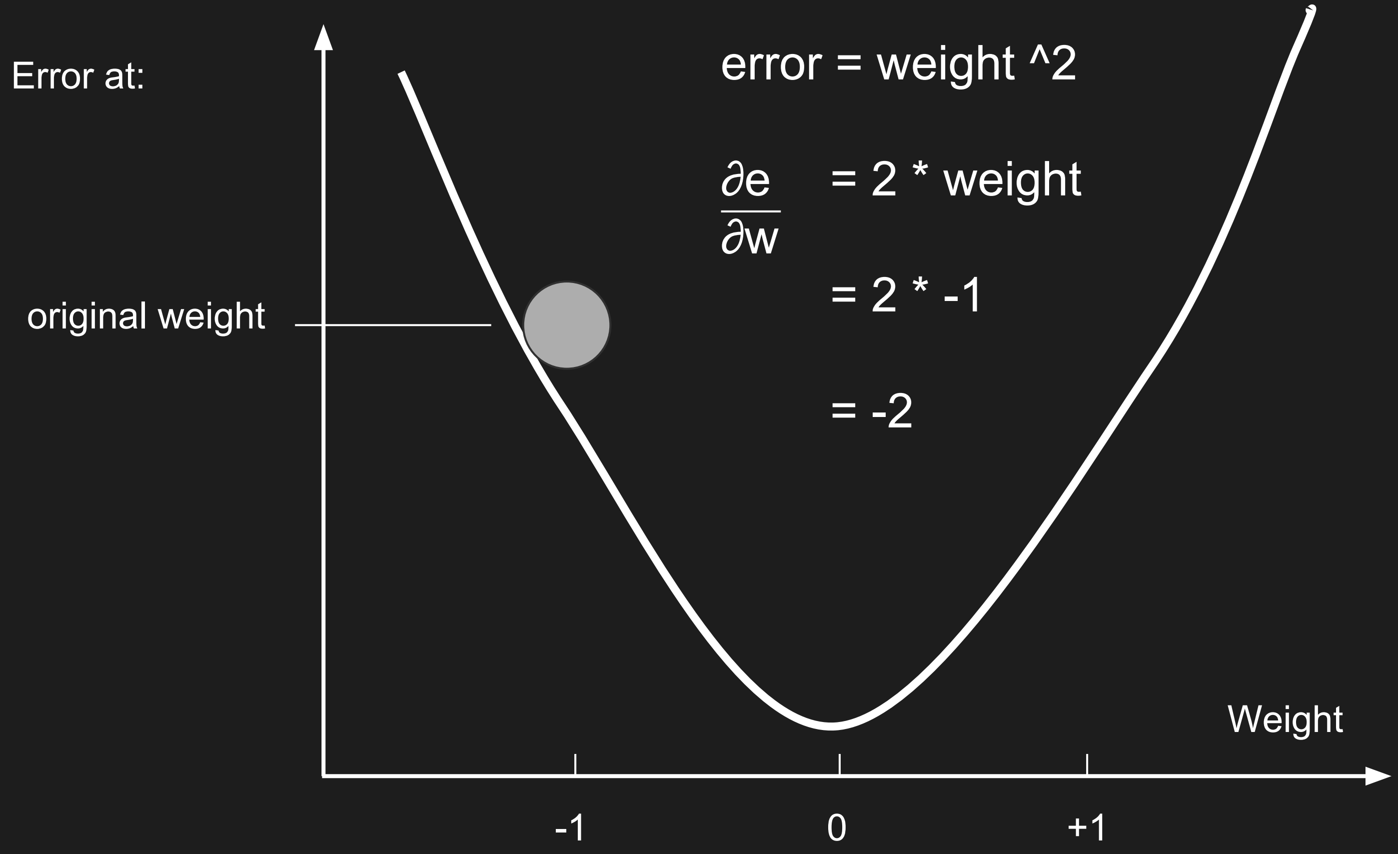
You have to know your error function.
For example:

error = weight ^2

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

$$= 2 * -1$$

$$= -2$$

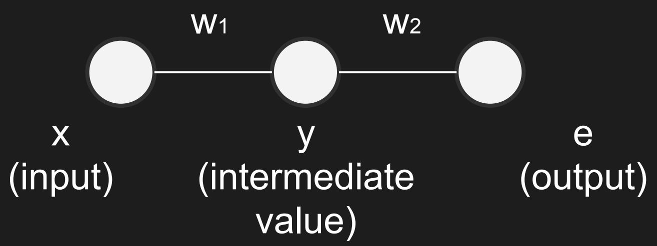


Backpropagation

- *The not so hard math*

Chaining

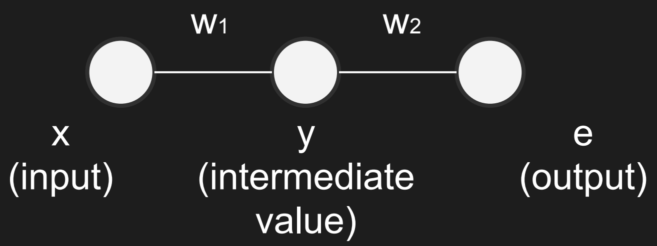
$$y = x * w_1$$



Chaining

$$y = x * w_1$$

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



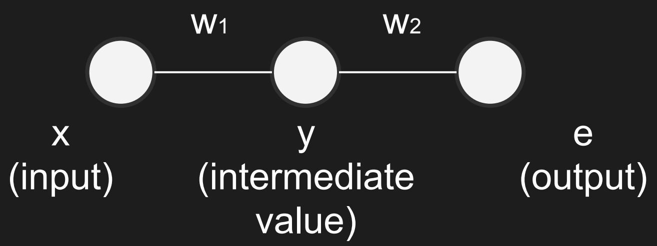
Chaining

$$y = x * w_1$$

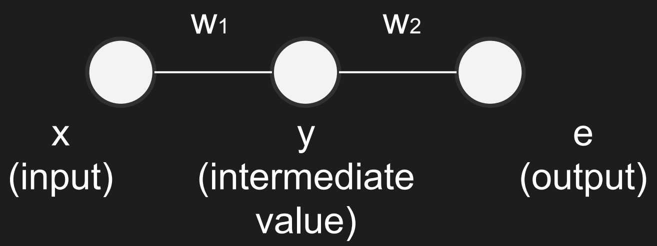
$$\frac{\partial y}{\partial w_1} = x$$

$$e = y * w_2$$

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



Chaining



$$y = x * w_1$$

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

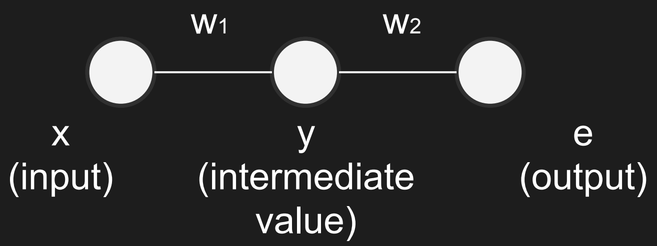
$$e = y * w_2$$

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

$$e = x * w_1 * w_2$$

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

Chaining



$$y = x * w_1$$

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

$$e = y * w_2$$

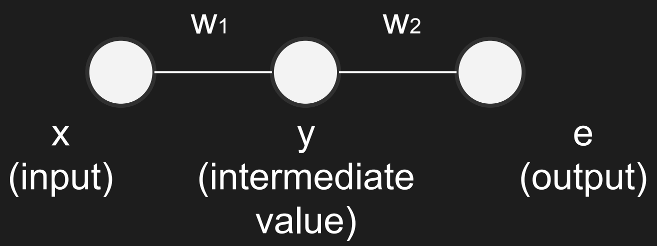
$$\frac{\partial e}{\partial y} = w_2$$

$$e = x * w_1 * w_2$$

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

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

Chaining



$$y = x * w_1$$

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

$$e = y * w_2$$

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

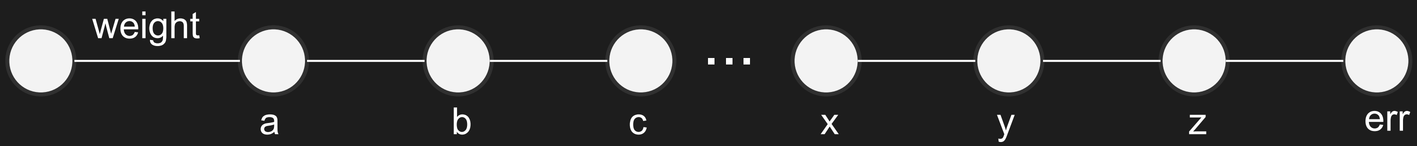
$$e = x * w_1 * w_2$$

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

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

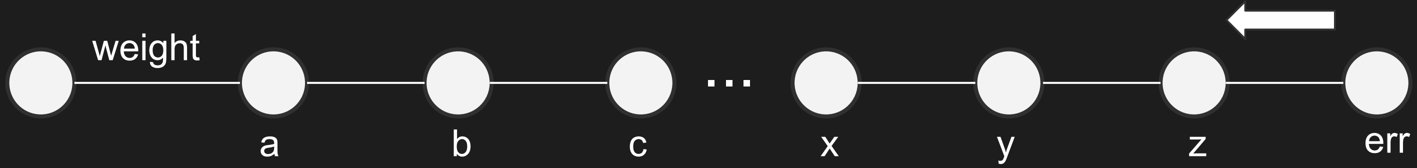
Chaining

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



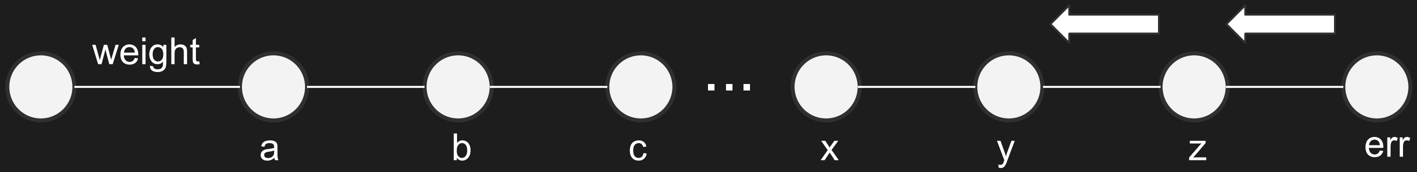
Backpropagation

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



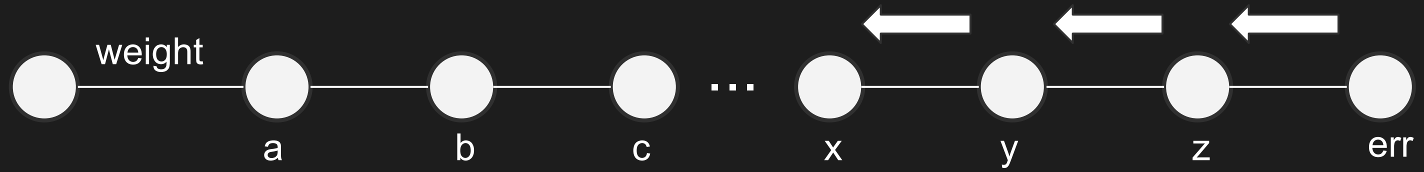
Backpropagation

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



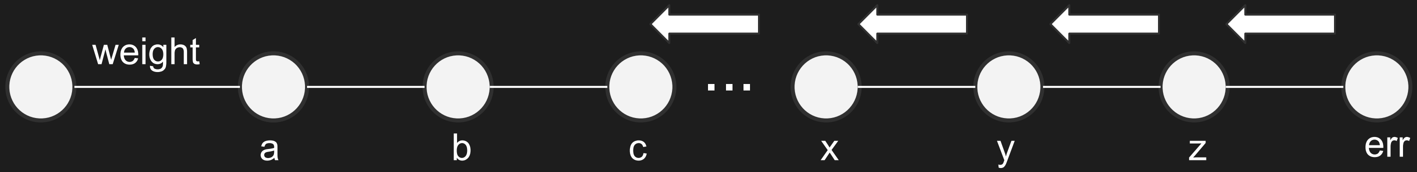
Backpropagation

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



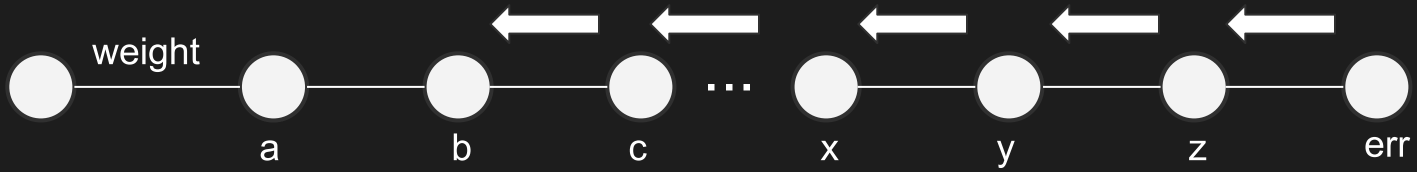
Backpropagation

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



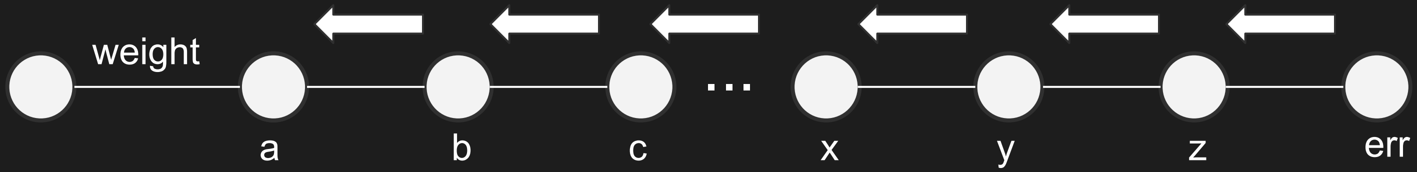
Backpropagation

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



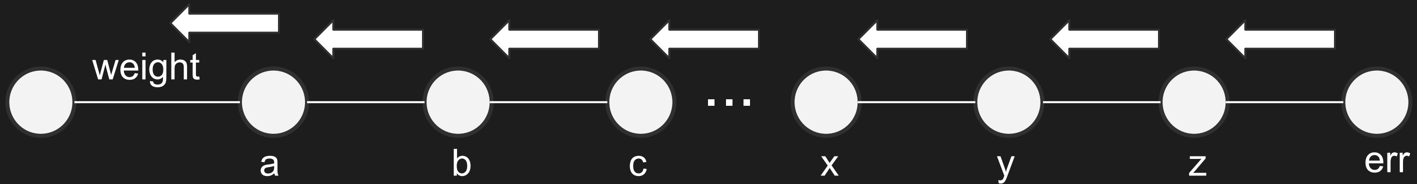
Backpropagation

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

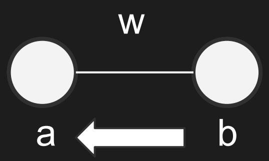


Backpropagation

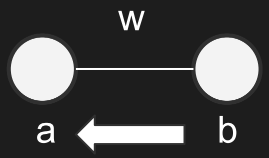
$$\frac{\partial err}{\partial weight} = \frac{\partial a}{\partial weight} * \frac{\partial b}{\partial a} * \frac{\partial c}{\partial b} * \frac{\partial d}{\partial c} * \ldots * \frac{\partial y}{\partial x} * \frac{\partial z}{\partial y} * \frac{\partial err}{\partial z}$$



Backpropagation challenge: weights

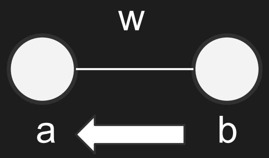


Backpropagation challenge: weights



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

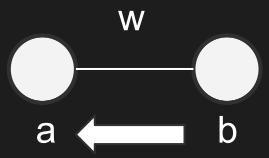
Backpropagation challenge: weights



$$b = wa$$

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

Backpropagation challenge: weights

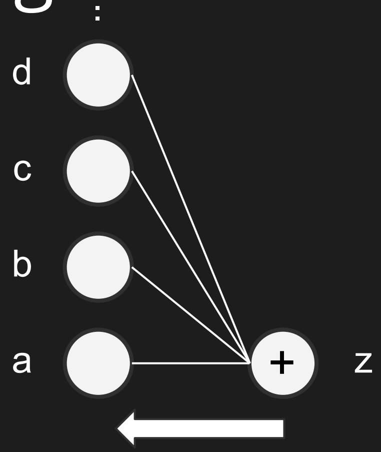


$$\frac{\partial err}{\partial a} = \frac{\partial b}{\partial a} * \frac{\partial err}{\partial b}$$

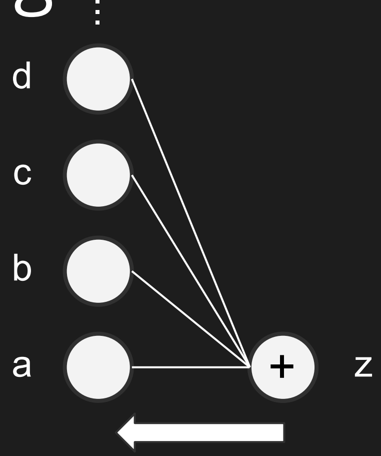
$$b = wa$$

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

Backpropagation challenge: sums

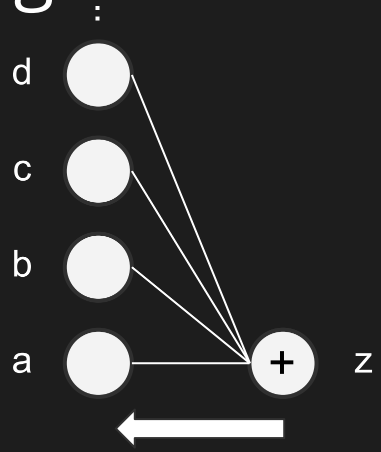


Backpropagation challenge: sums



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

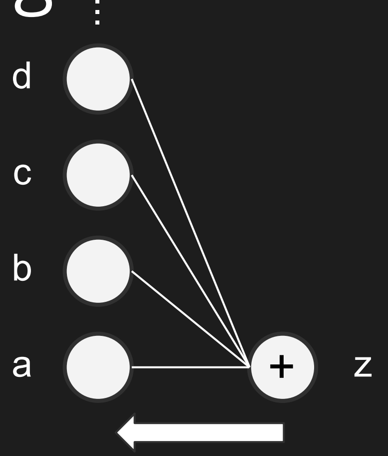
Backpropagation challenge: sums



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

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

Backpropagation challenge: sums

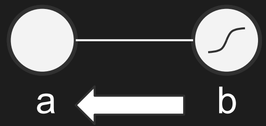


$$\frac{\partial err}{\partial a} = \frac{\partial z}{\partial a} * \frac{\partial err}{\partial z}$$

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

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

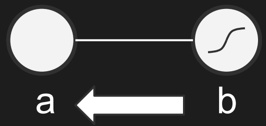
Backpropagation challenge: sigmoid



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

Backpropagation challenge: sigmoid

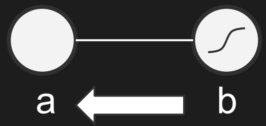
$$b = \frac{1}{1 + e^{-a}}$$



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

Backpropagation challenge: sigmoid

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



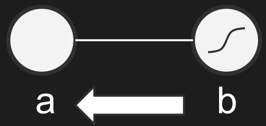
$$\frac{\partial err}{\partial a} = \frac{\partial b}{\partial a} * \frac{\partial err}{\partial b}$$

Backpropagation challenge: sigmoid

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

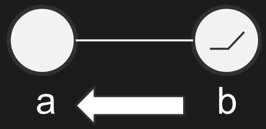
Because math is beautiful / dumb luck:

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



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

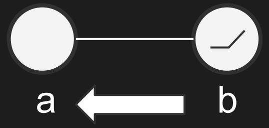
Backpropagation challenge: ReLU



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

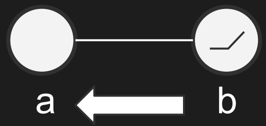
Backpropagation challenge: ReLU

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



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

Backpropagation challenge: ReLU

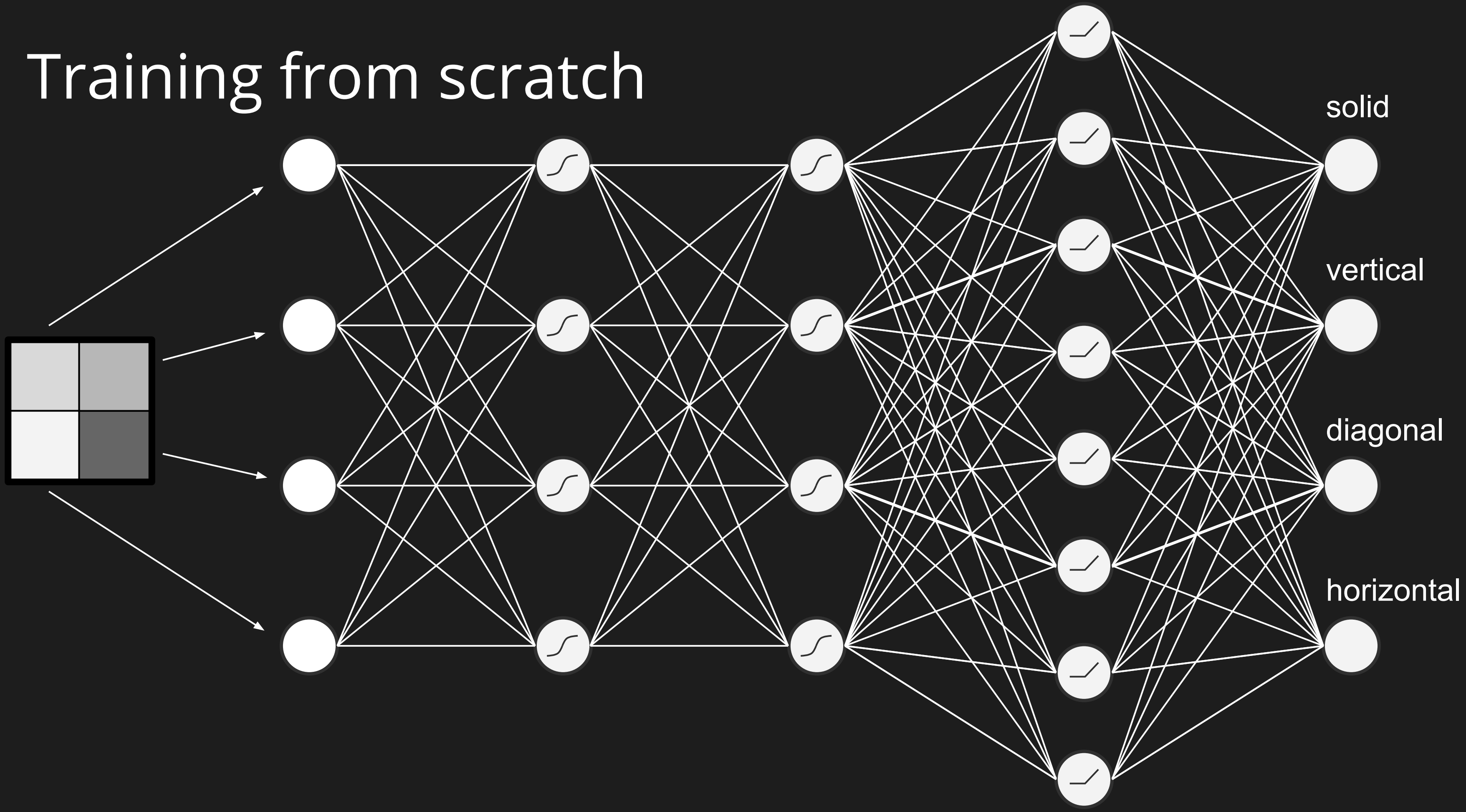


$$\frac{\partial err}{\partial a} = \frac{\partial b}{\partial a} * \frac{\partial err}{\partial b}$$

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

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

Training from scratch



Desired Network