Artificial neural networks and deep learning

Analytical workflows for earth sciences: spring 2025

May 21

Goals for todays lecture:

1. What is deep learning and what does it have to do with neural networks?

2. Why have deep learning algorithms been so successful?

3. What is the structure of a neural network?

4. How do neural networks represent functions?

5. How are neural networks trained?

Deep learning is a catch all term for machine learning algorithms based around neural networks

Examples of deep learning algorithms:

1. Multilayer perceptron
2. Convolutional neural network
3. Recurrent neural networks
4. Transformers

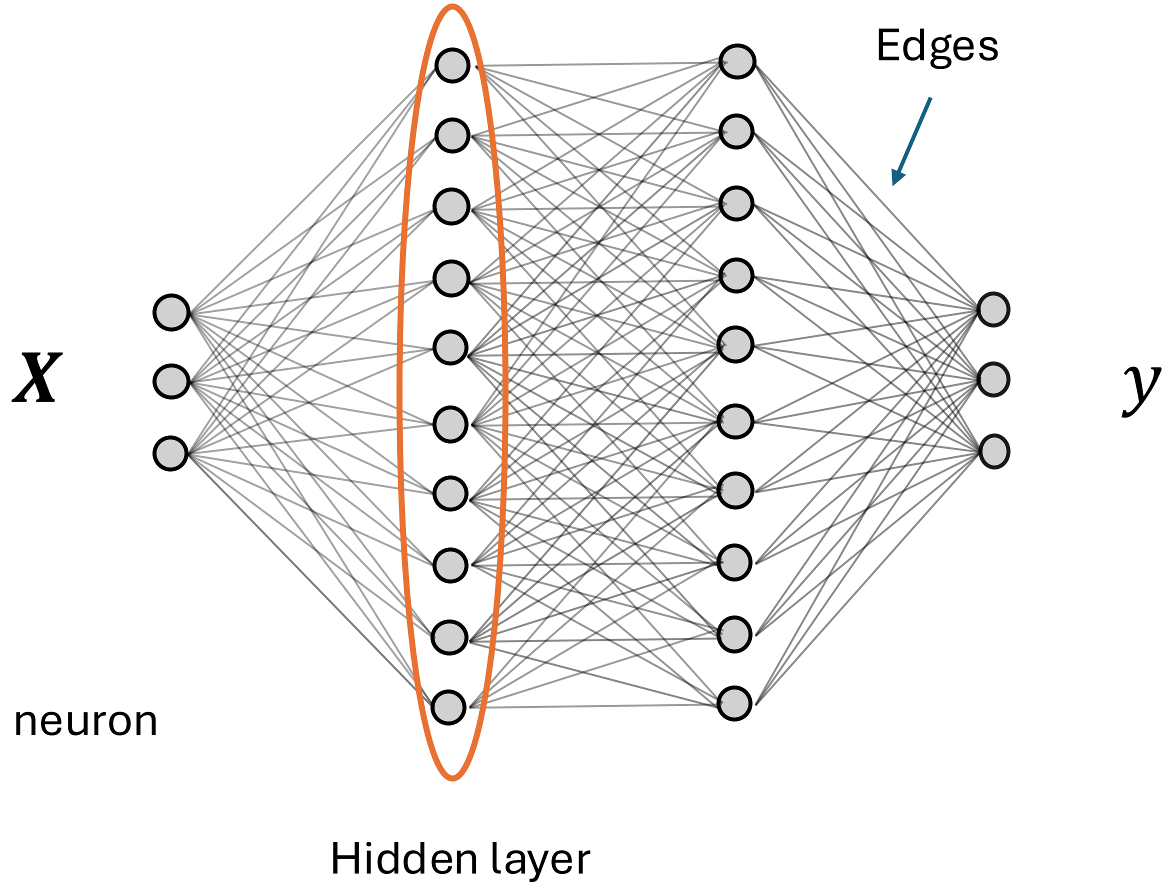
Use neural networks as the basic learning mechanisms

Neural networks are very flexible structures for learning functions from data

Passes an input **X** through a series of neurons

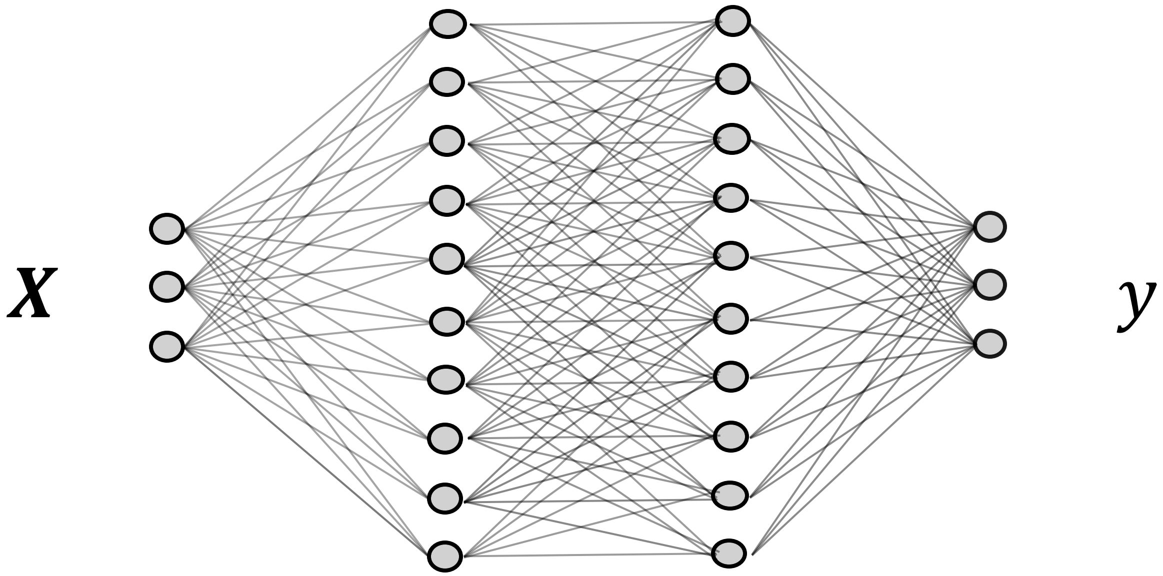
Each neuron in simple function of the inputs

The output **y** can represent very complex function of the inputs



Why do neural networks work?

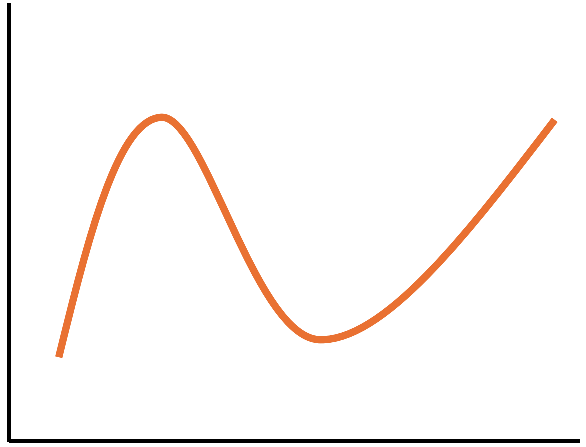
Neural networks are universal function approximators
Given enough neurons in each layer, they can represent any relationship between inputs and outputs



Why do neural networks work?

Neural networks can represent function with many inputs in a computationally efficient way

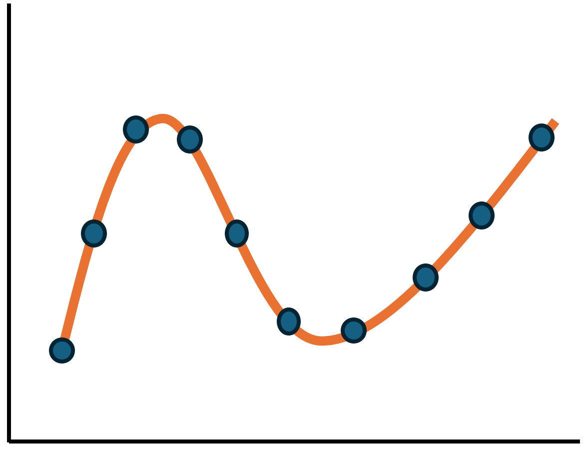
Typical function approximation methods suffer from the "curse of dimensionality"



Why do neural networks work?

Neural networks can represent function with many inputs in a computationally efficient way

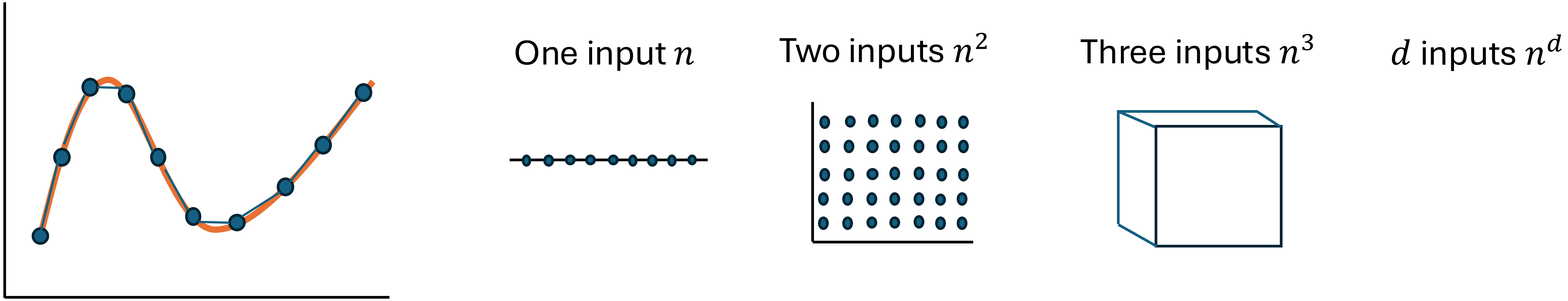
Typical function approximation methods suffer from the "curse of dimensionality"



Why do neural networks work?

Neural networks can represent function with many inputs in a computationally efficient way

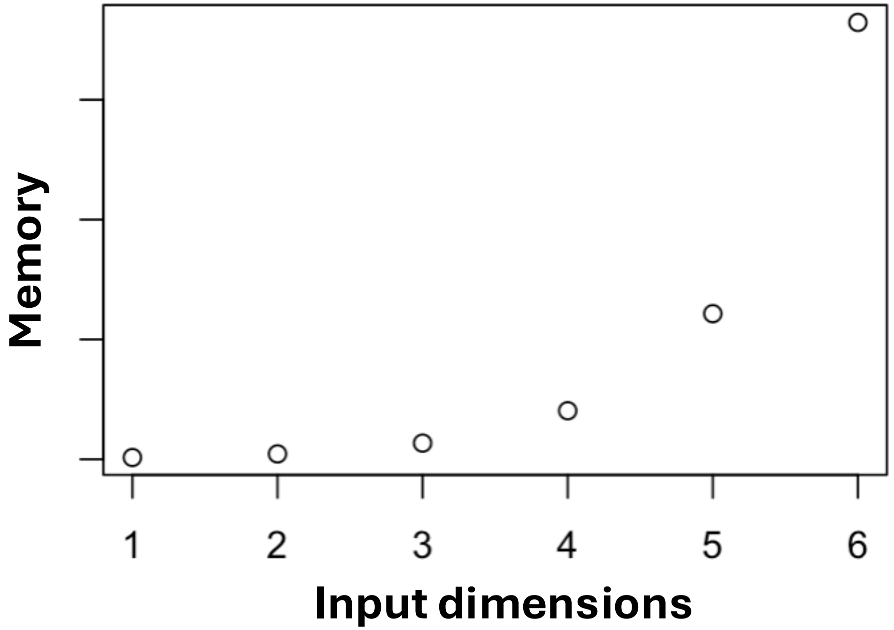
Typical function approximation methods suffer from the "curse of dimensionality"



Why do neural networks work?

Neural networks can represent function with many inputs in a computationally efficient way

Typical function approximation methods suffer from the "curse of dimensionality"

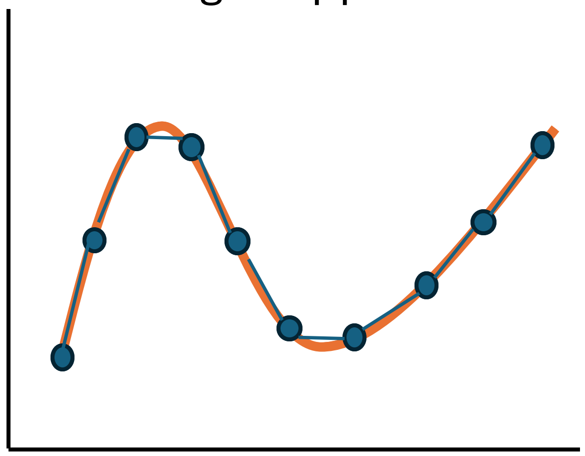


Why do neural networks work?

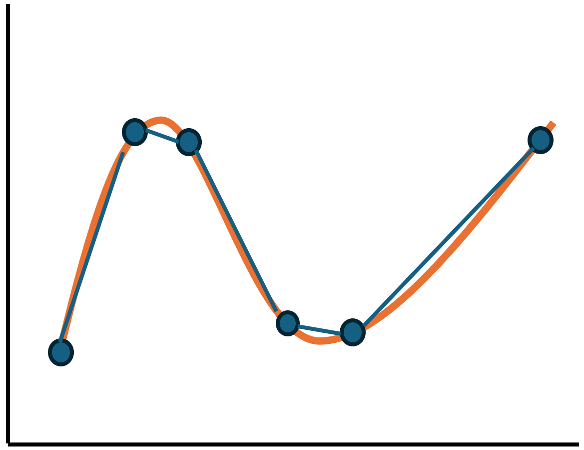
Neural networks can represent function with many inputs in a computationally efficient way

Neural networks require less memory by learning which features of a function are important

Naive grid approach

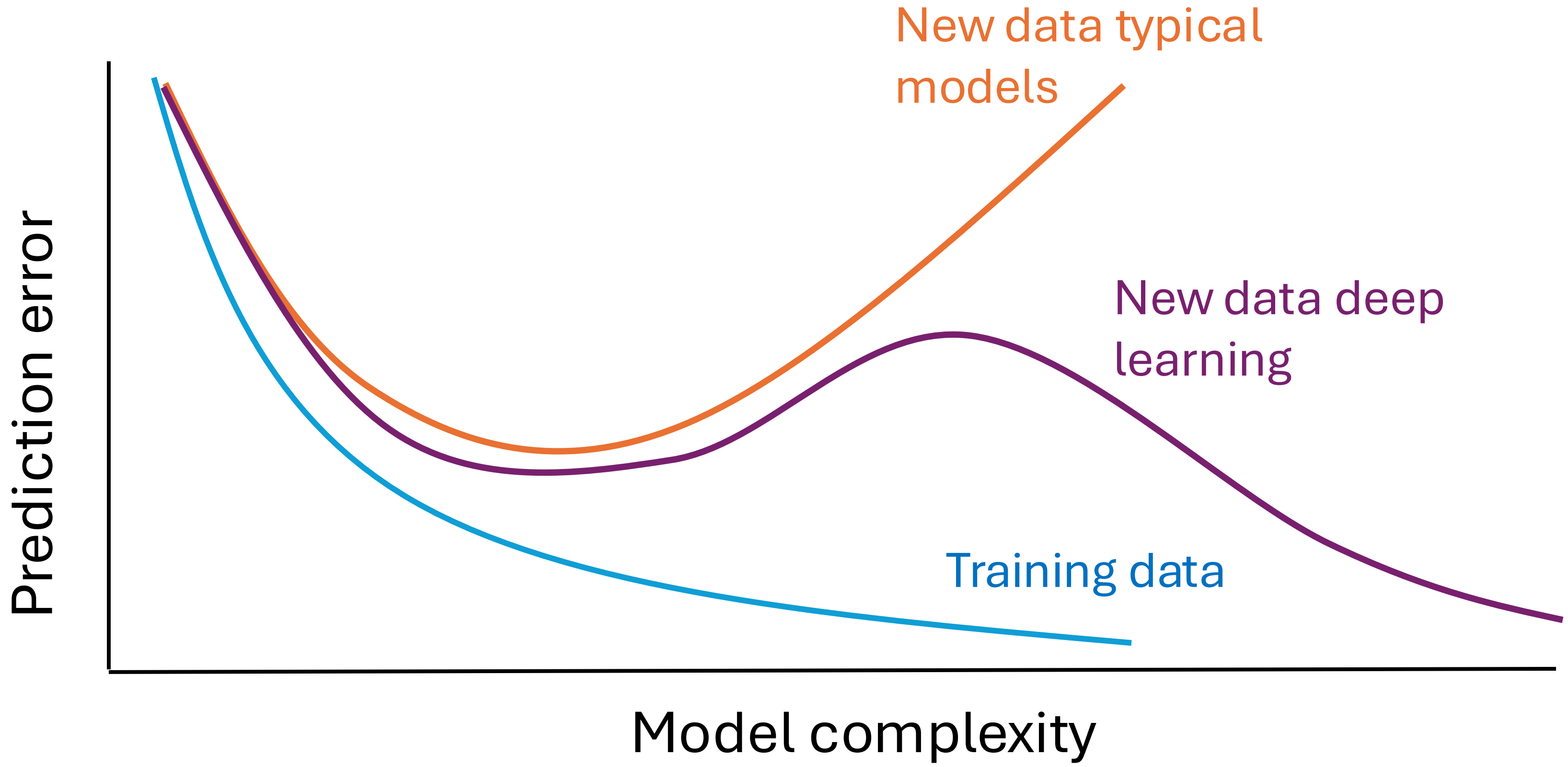


Optimized placement

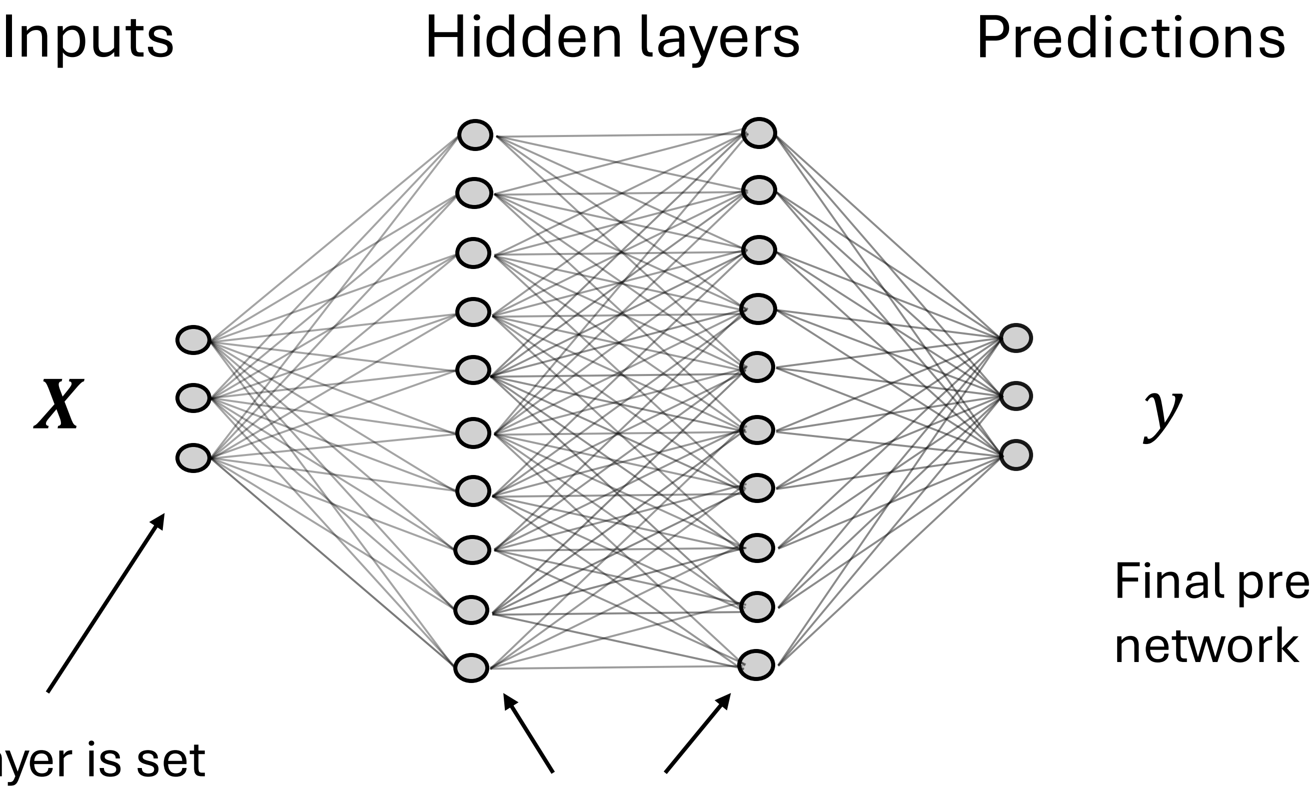


Why do neural networks work?

Increasing the complexity of neural networks can help them generalize to new data sets!



The structure of a neural network



Neurons store numeric values

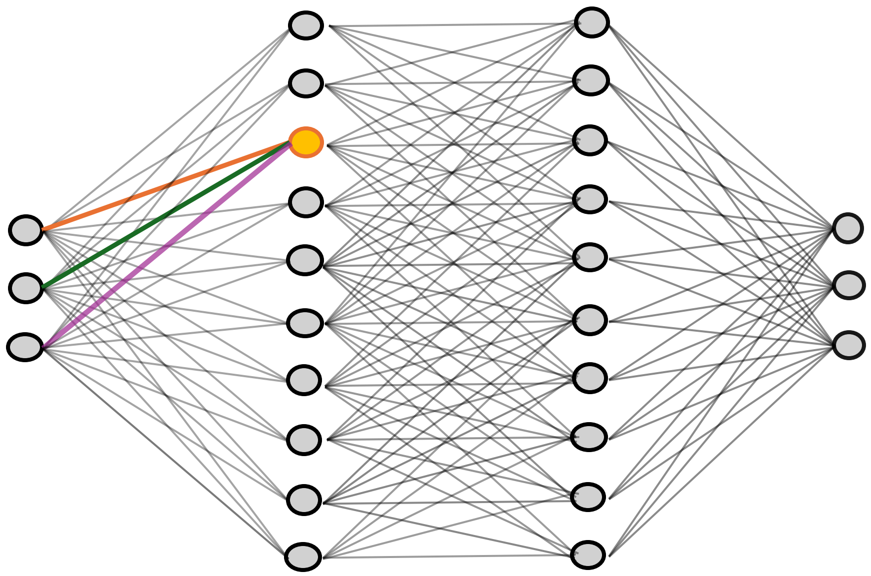
Edges determine relationships between neurons

Final prediction of the network

The first layer is set equal to the input $\boldsymbol{X}$

The value of the hidden layers determined by values in the prior layer and the edges

The value of a neuron depends on its connection to the previous layer and an activation function

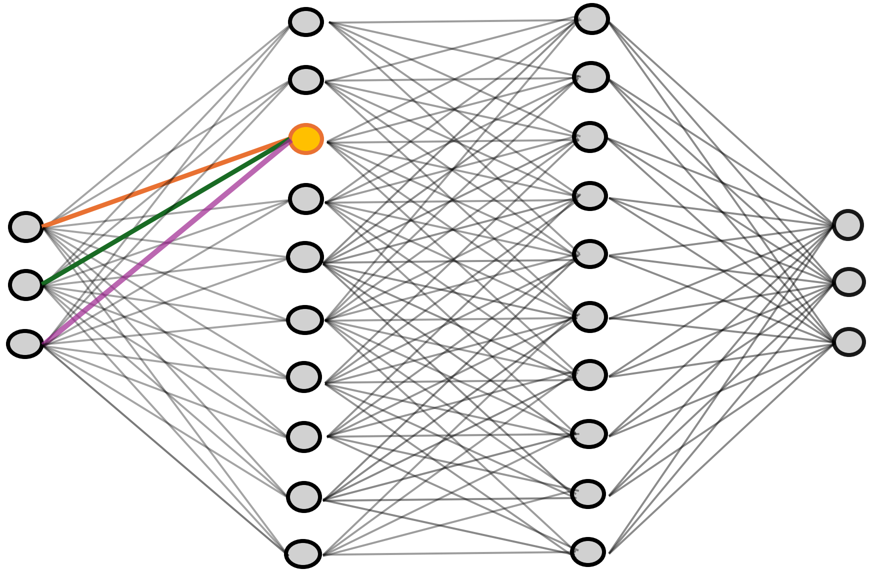


$$z_{1,i} = w_{1,i}^{1} x_1 + w_{2,i}^{1} x_2 + w_{3,i}^{1} x_3 + b_{1,i}$$

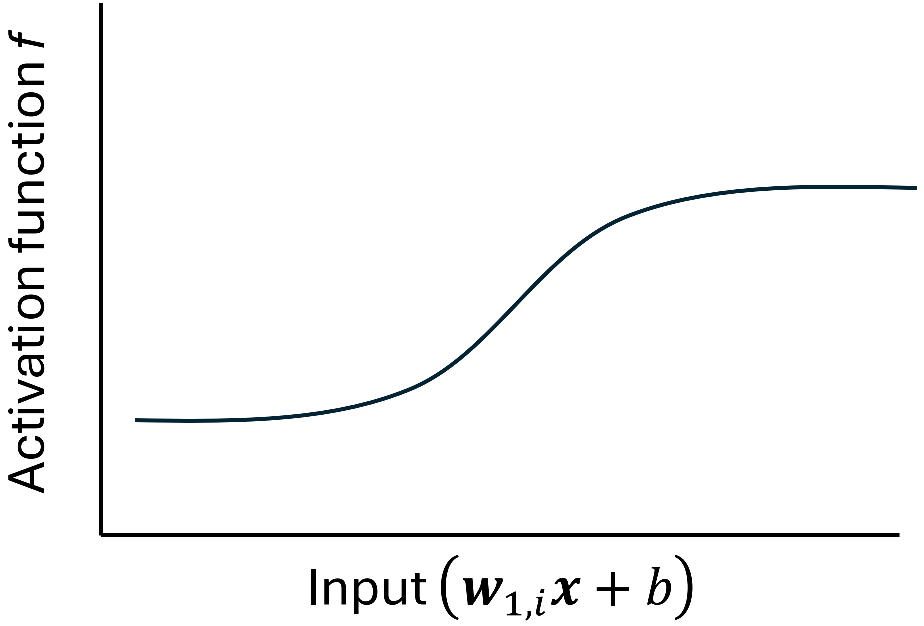
$$a_{1,i} = f_1\left(z_{1,i}\right)$$

$$a_{1,i} = f_1\left(\boldsymbol{w}_{1,i}\boldsymbol{x} + b\right)$$

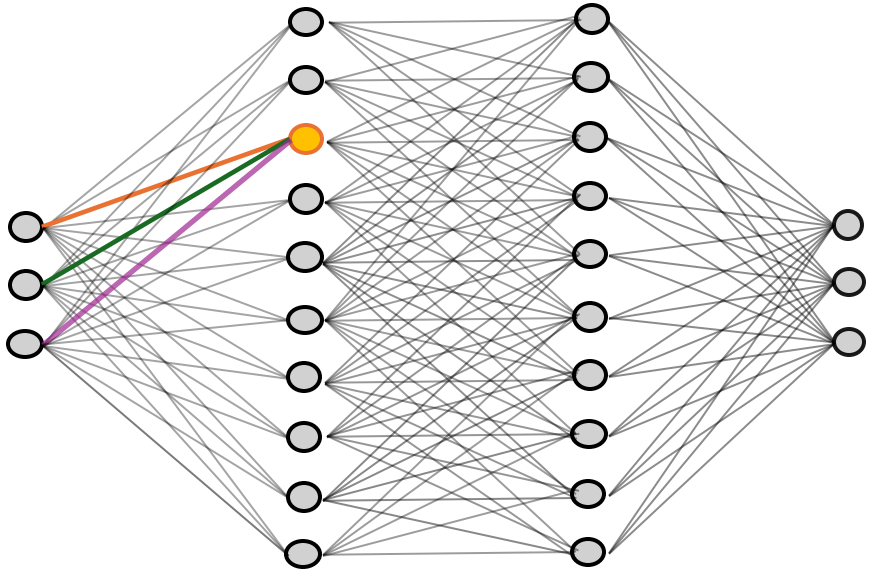
The structure of a neural network



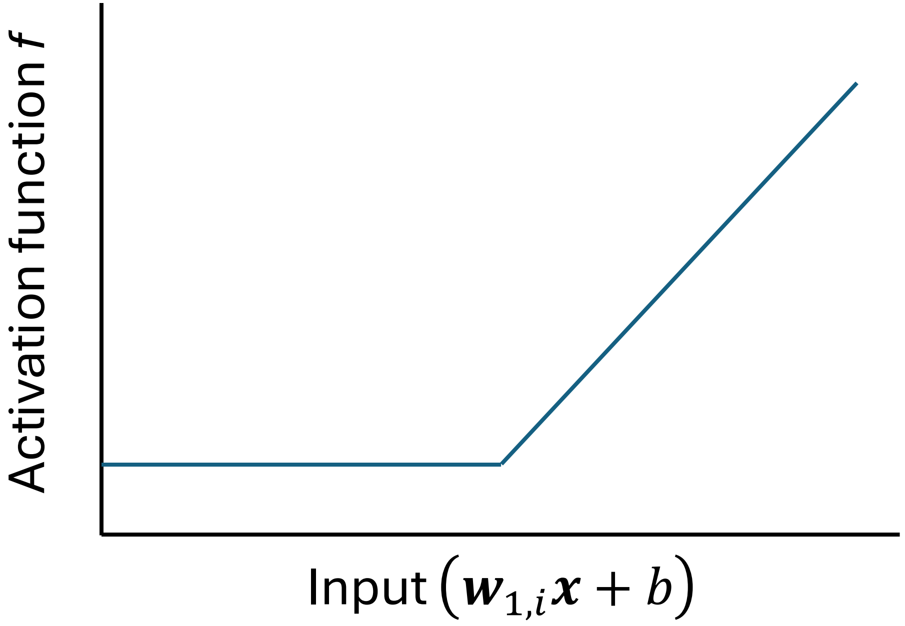
Activation function:
Hyperbolic tangent



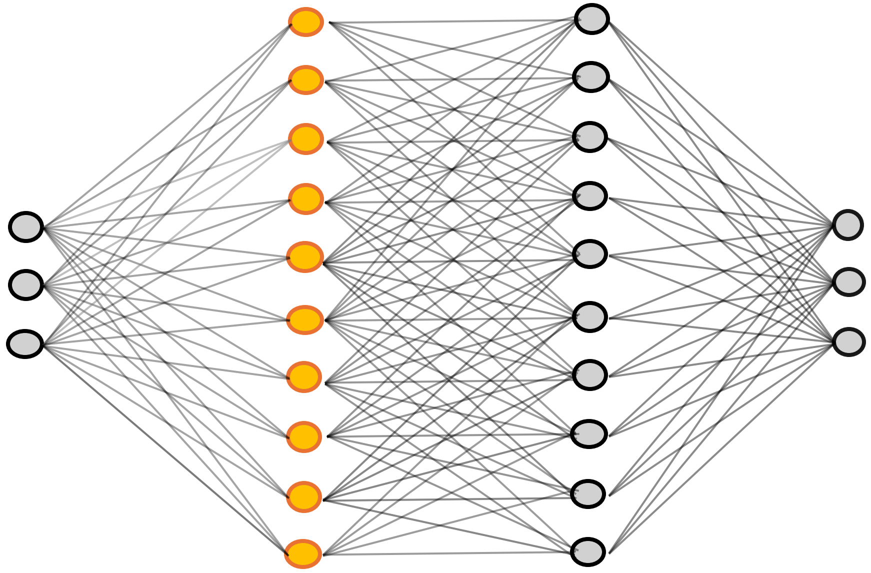
The value of a neuron depends on its connection to the previous layer and an activation function



Activation function:
ReLU



The operations performed by a neural network are typically expressed using matrix algebra



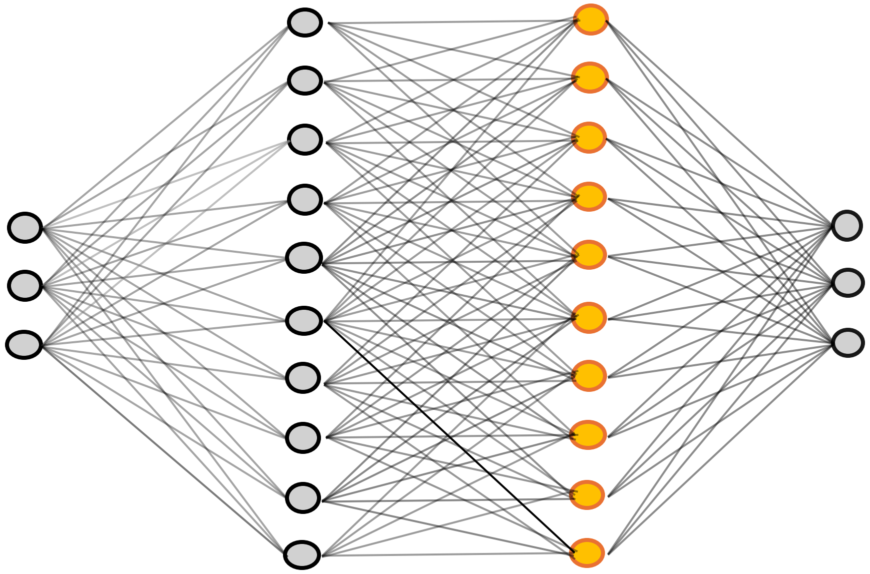
$$z_{1,i} = w^1_{1,i}x_1 + w^1_{2,i}x_2 + w^1_{3,i}x_3 + b^1_{1,i}$$

$$\boldsymbol{z}_1 = \begin{bmatrix} b^1_1 & w^1_{1,1} & w^1_{2,1} & w^1_{3,1} \\ b^1_2 & w^1_{1,2} & w^1_{2,2} & w^1_{3,2} \\ \dots & & & \\ b^1_h & w^1_{1,h} & w^1_{2,h} & w^1_{3,h} \end{bmatrix} \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$\boldsymbol{z}_1 = W^1\boldsymbol{x}$$

$$\boldsymbol{a_1} = f(\boldsymbol{z}_1)$$

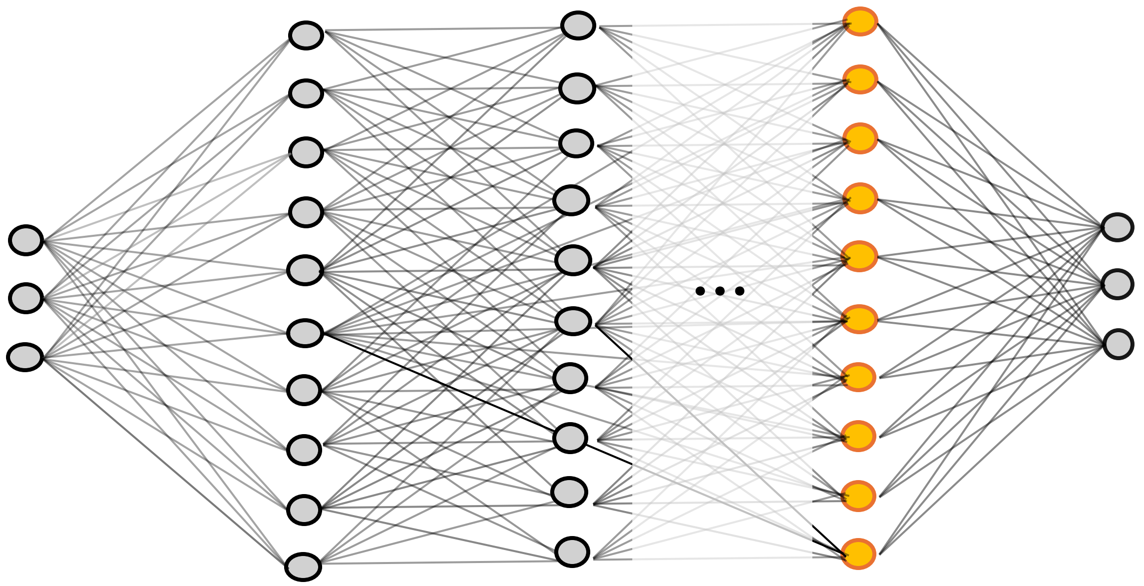
The activations in subsequent layers are determined by applying the weights and activations repeatedly



$$\boldsymbol{a}_2 = f_2(W^2 \boldsymbol{a}_{\mathbf{1}})$$

$$\boldsymbol{a}_{\mathbf{2}} = f_2\left(W^2 f_1(W^1 \boldsymbol{x})\right)$$

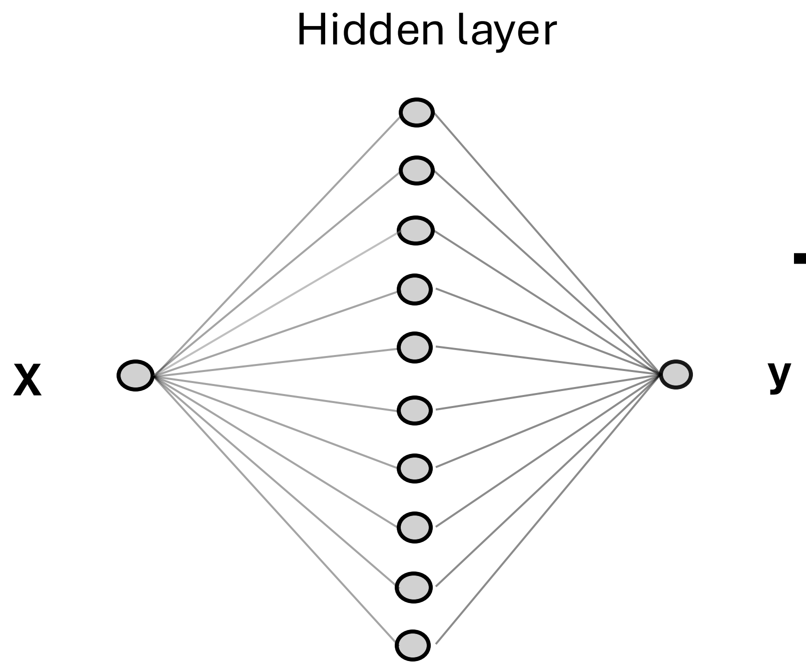
The activations in subsequent layers are determined by applying the weights and activations repeatedly

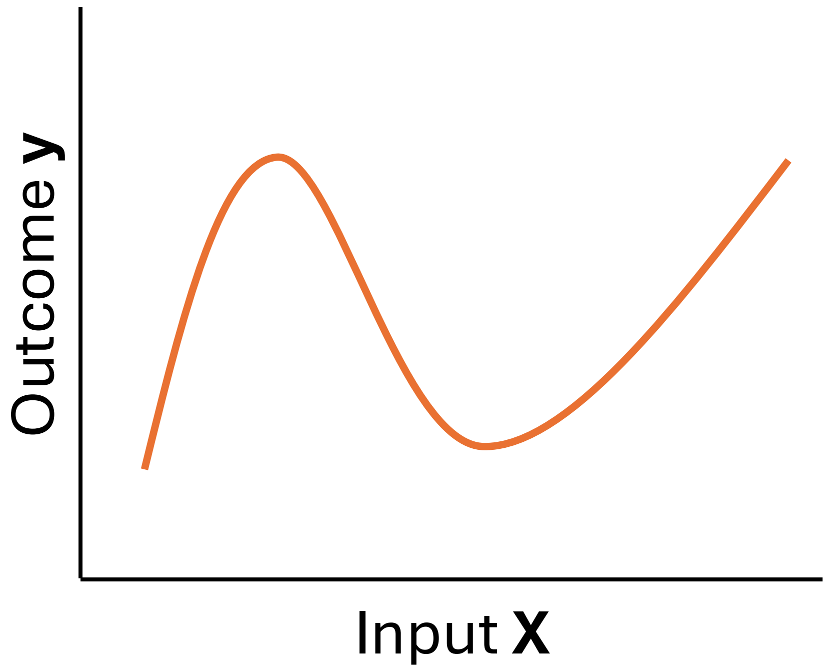


$$\boldsymbol{a}_l = f_l\left(W^l f_{l-1}\left(W^{l-1} \ \dots f_1(W^1\boldsymbol{x}) \dots\right)\right)$$

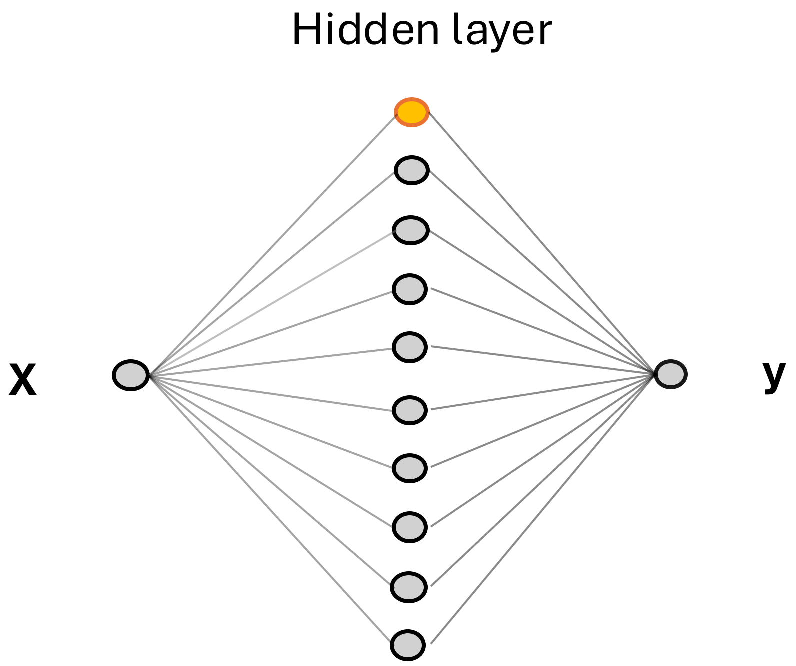
$$\boldsymbol{a}_l = f_l\left(W^l \boldsymbol{a}_{\boldsymbol{l-1}}\right)$$

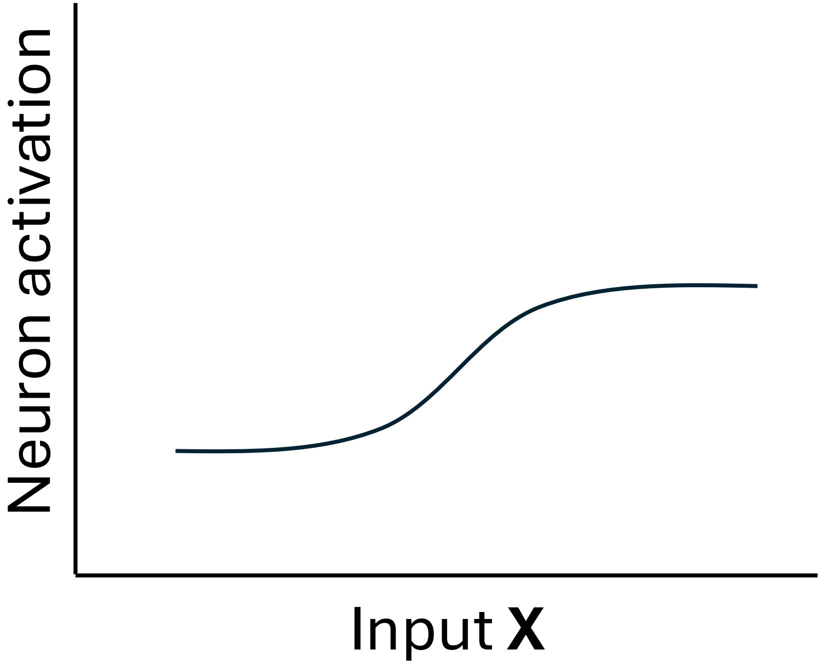
How do neural networks represent functions?



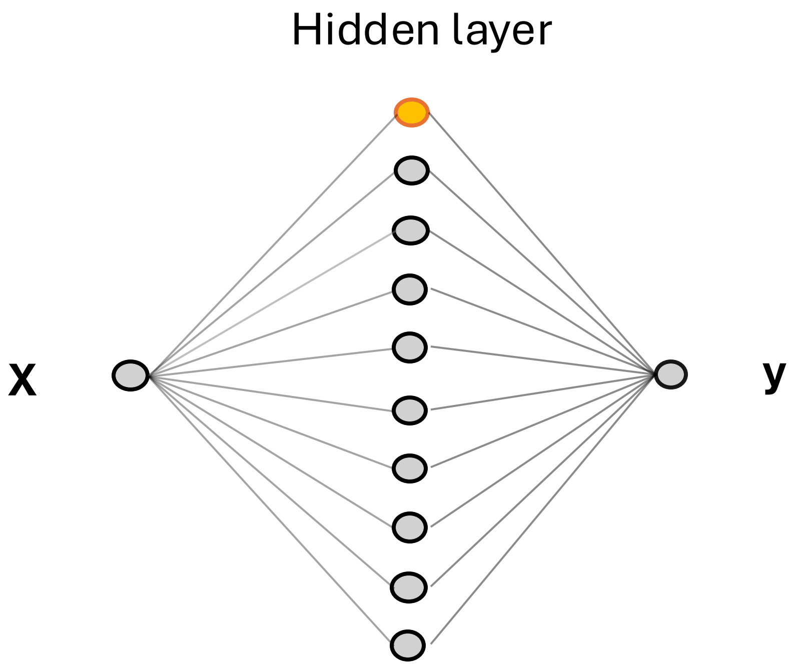


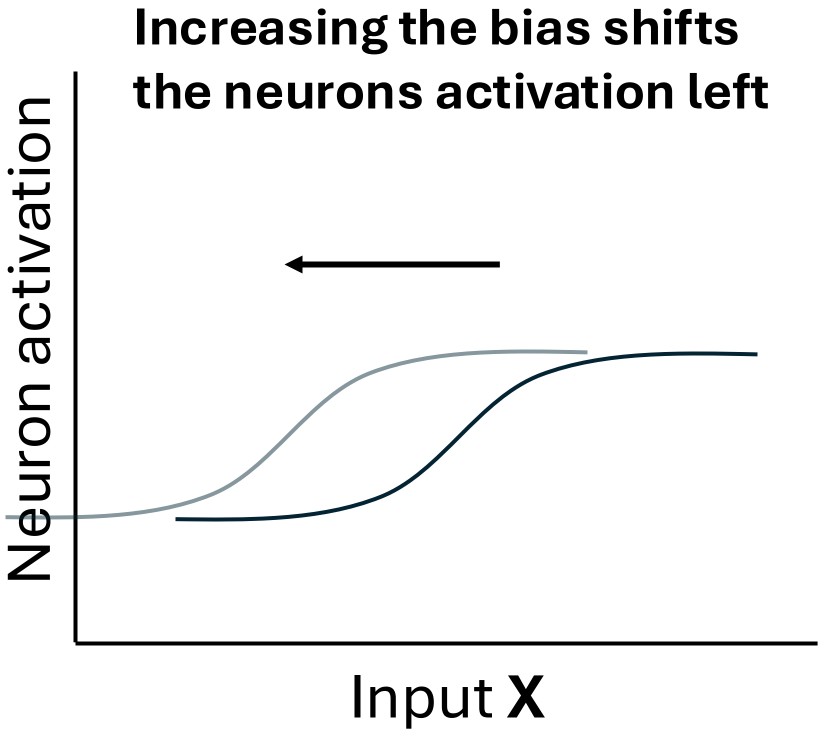
How do neural networks represent functions?



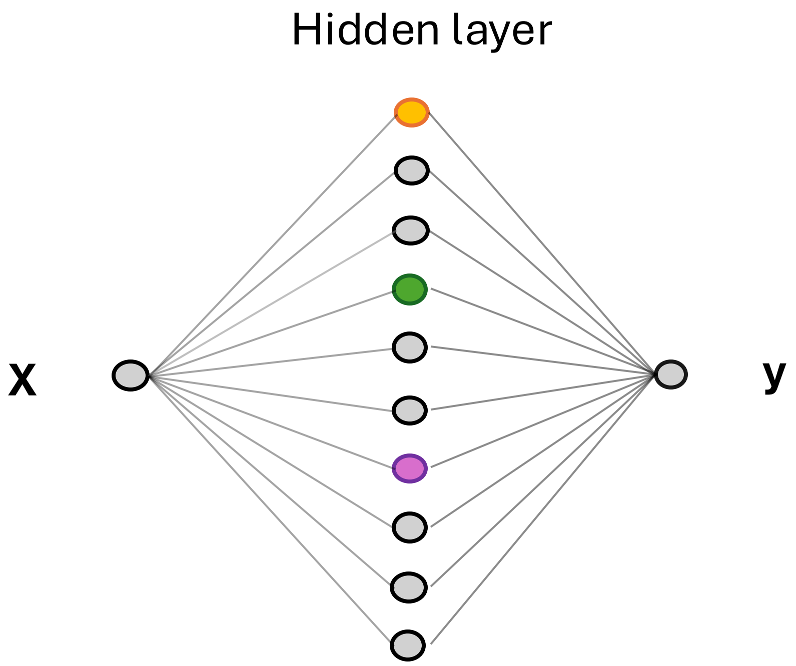


How do neural networks represent functions?

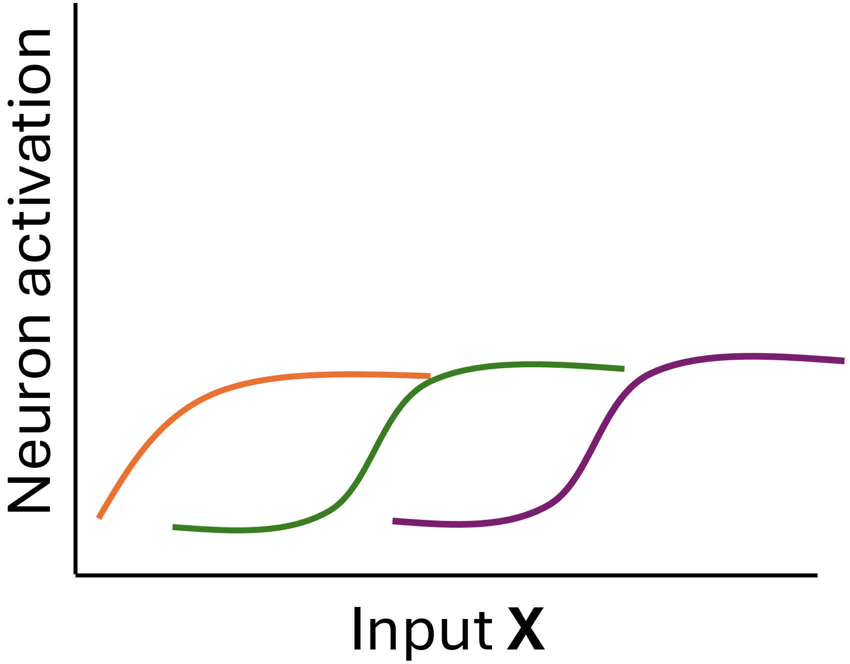




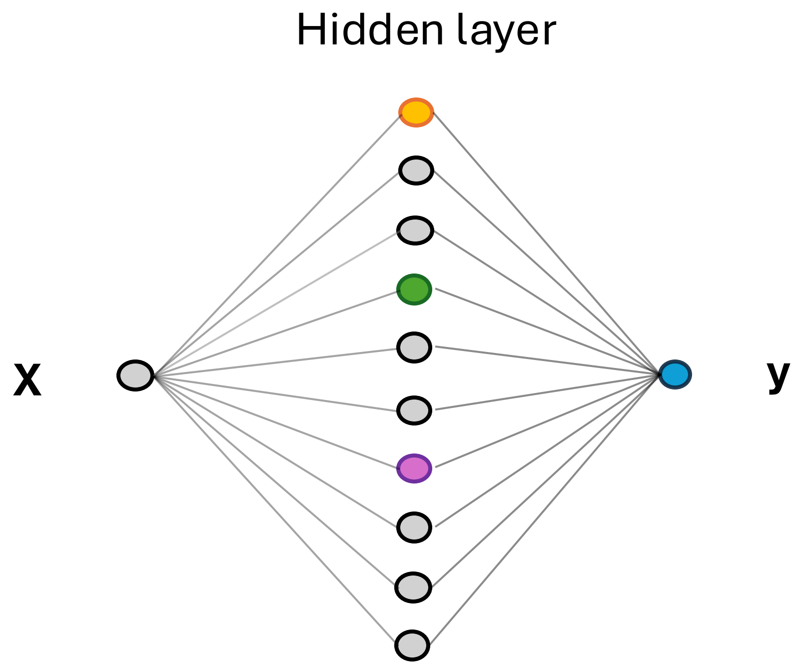
How do neural networks represent functions?



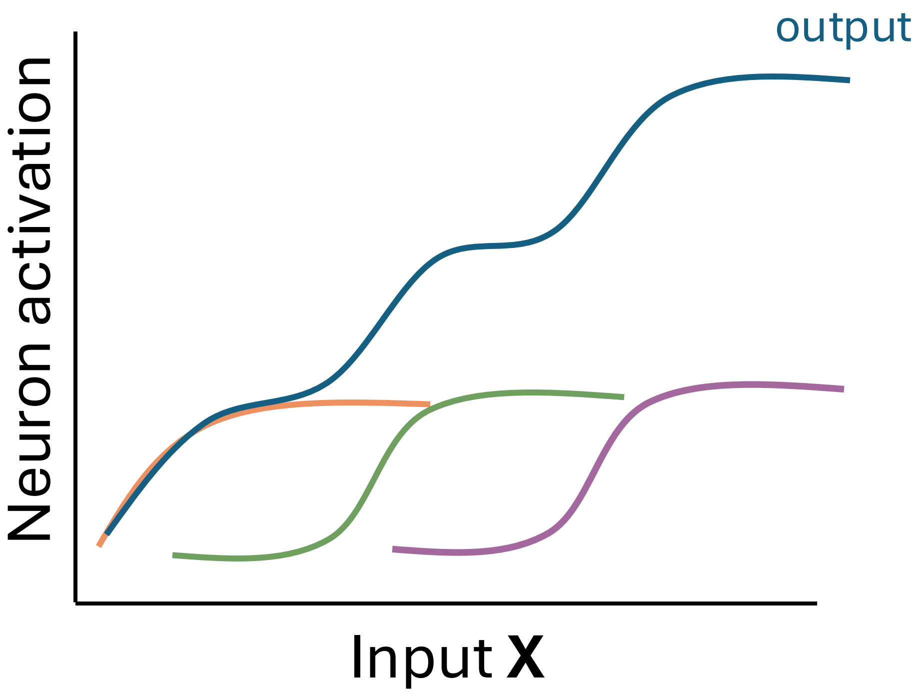
Different neurons in the hidden layer can be shifted different amounts



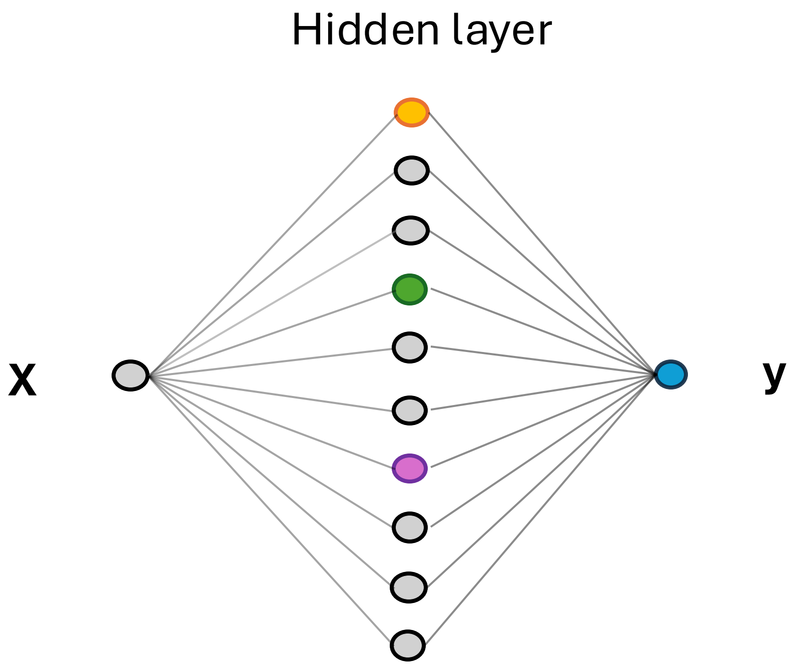
How do neural networks represent functions?



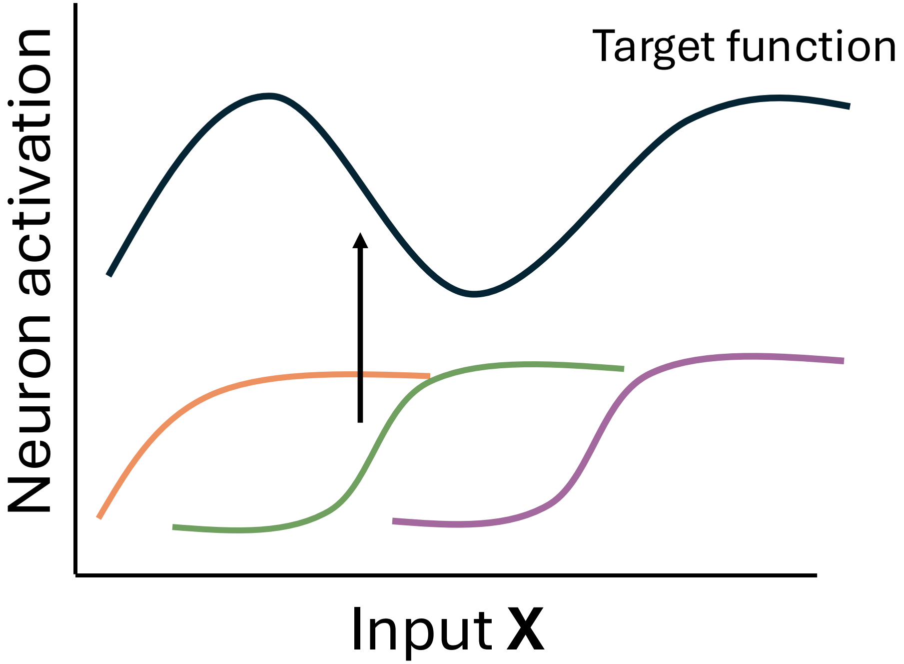
The final prediction of the network is the weighted sum of the hidden neurons



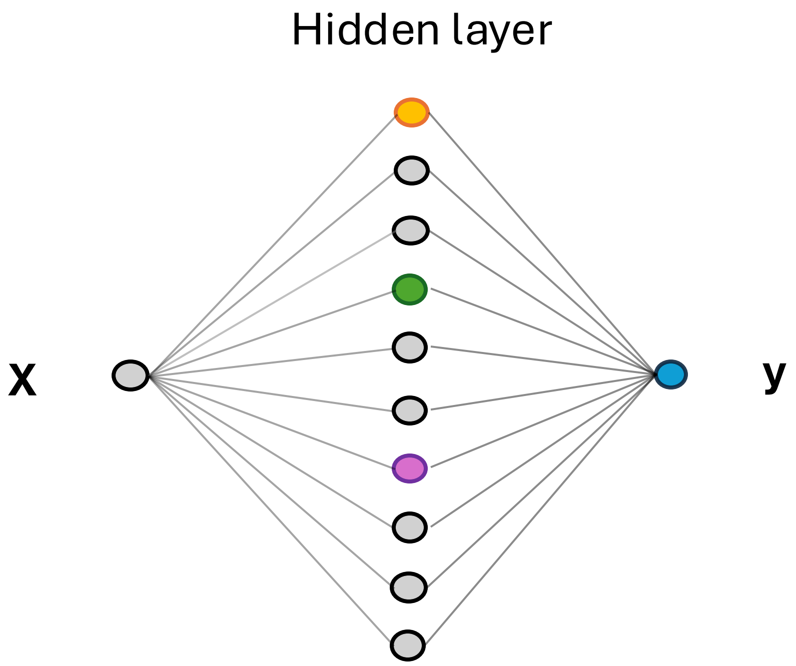
How do neural networks represent functions?



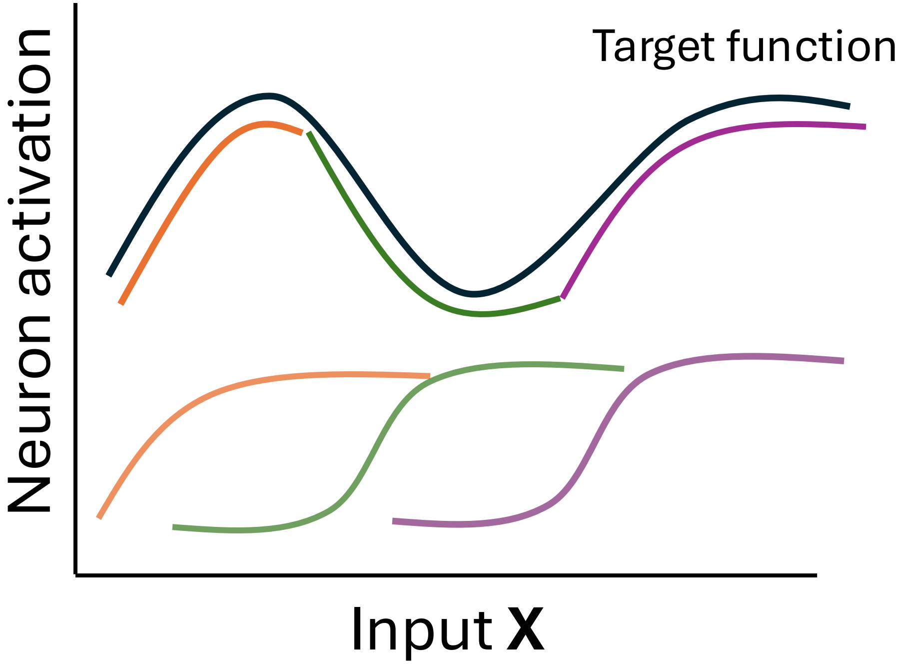
Set the weights on each neuron to match the slope of the target function



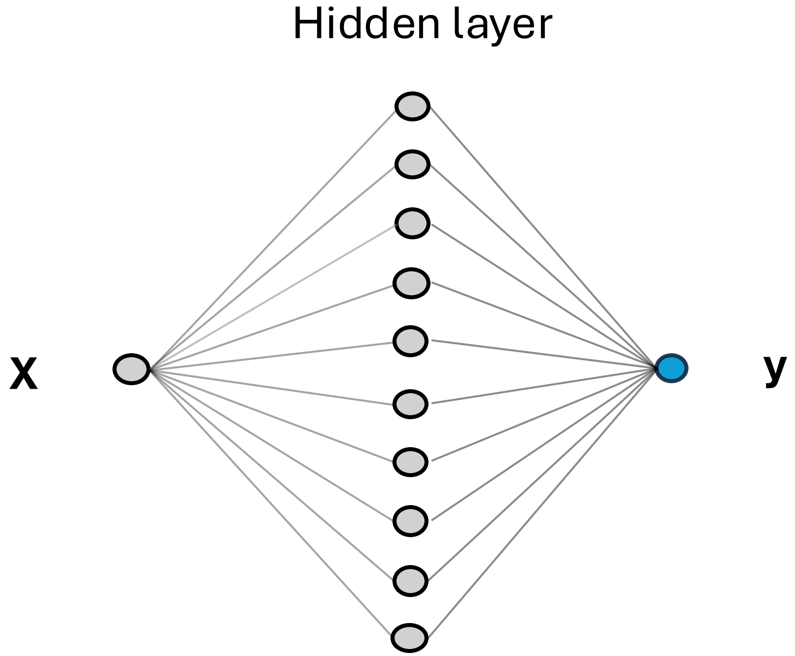
How do neural networks represent functions?



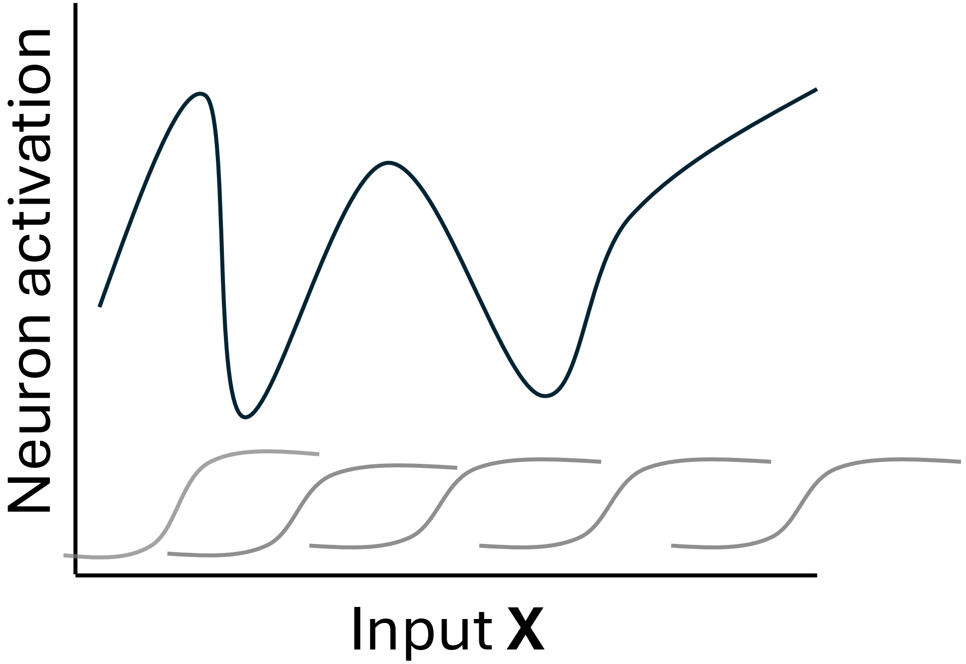
Set the weights on each neuron to match the slope of the target function



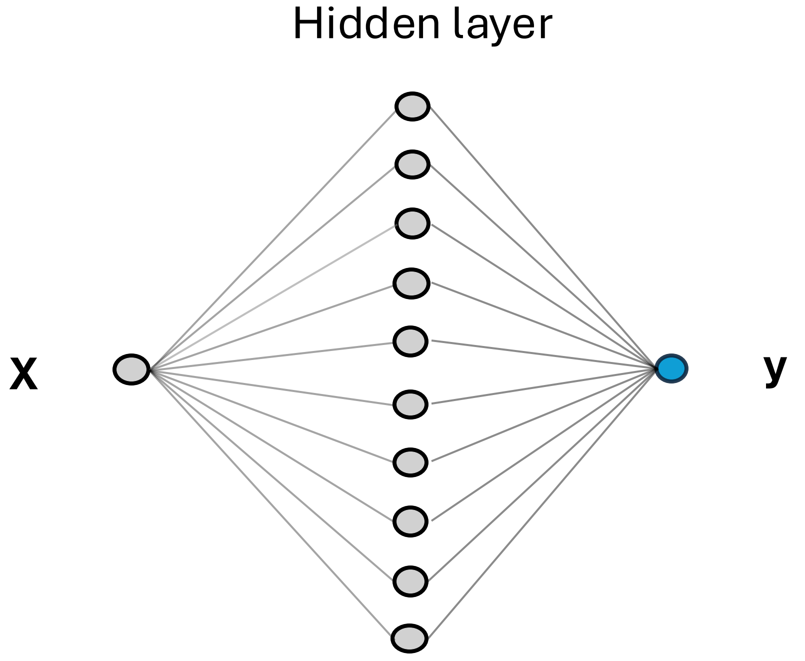
How do neural networks represent functions?

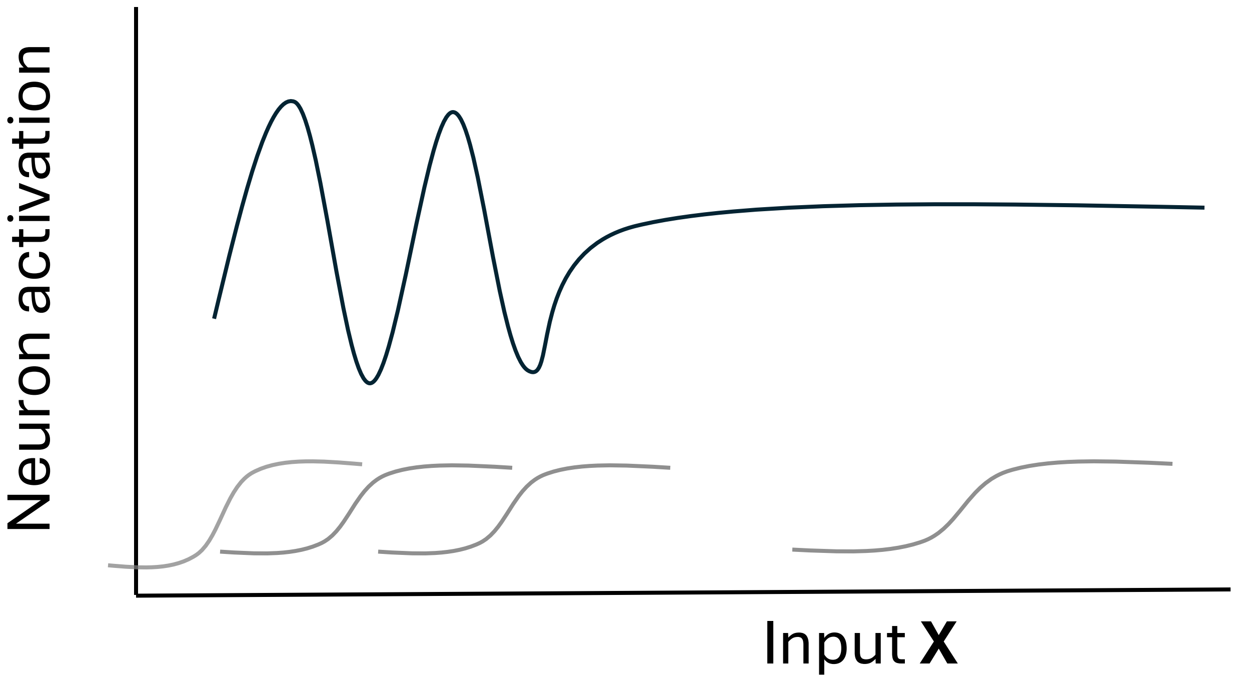


We can always s match complex functions provided we have enough neurons



The neurons in the hidden layer can be adapted to match match key features in the target function



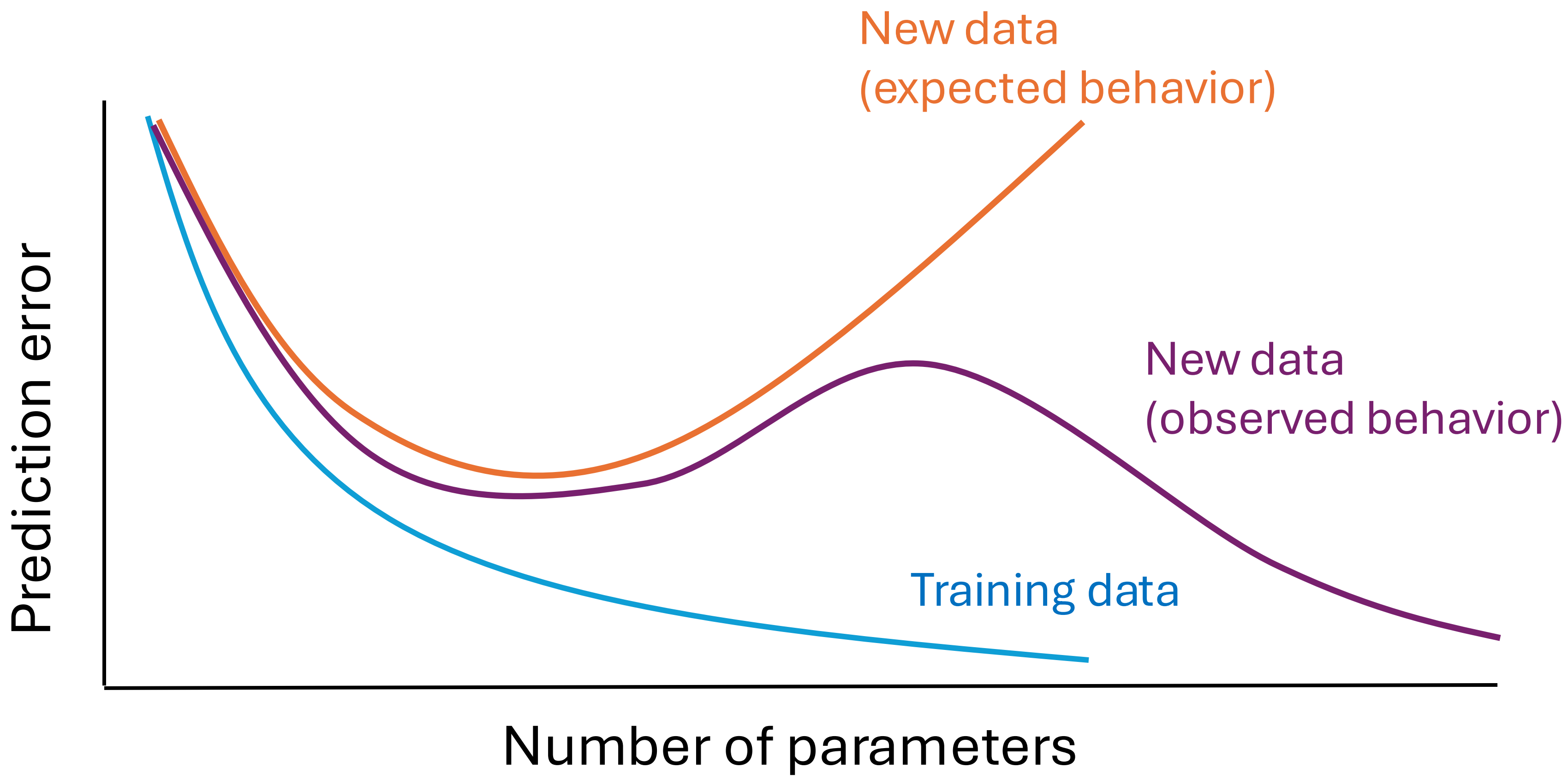


How do neural networks represent functions?

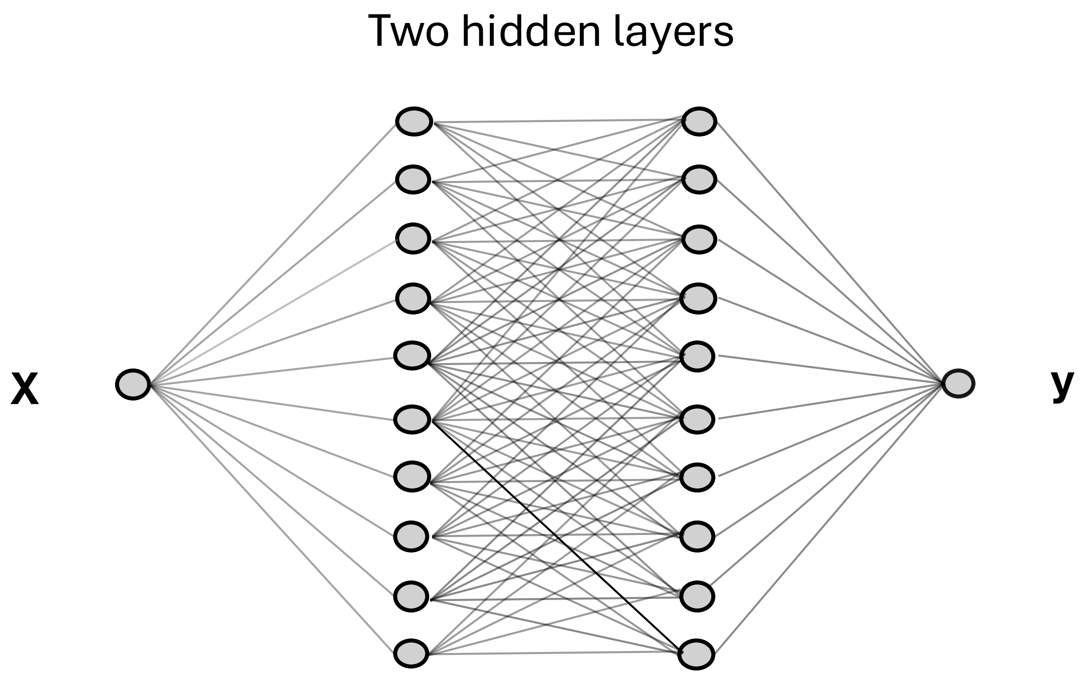
Universal approximation theorem:

A neural network with a single hidden layer with non-linear activation functions can represent any continuous target function provided it has enough neurons in the hidden layer

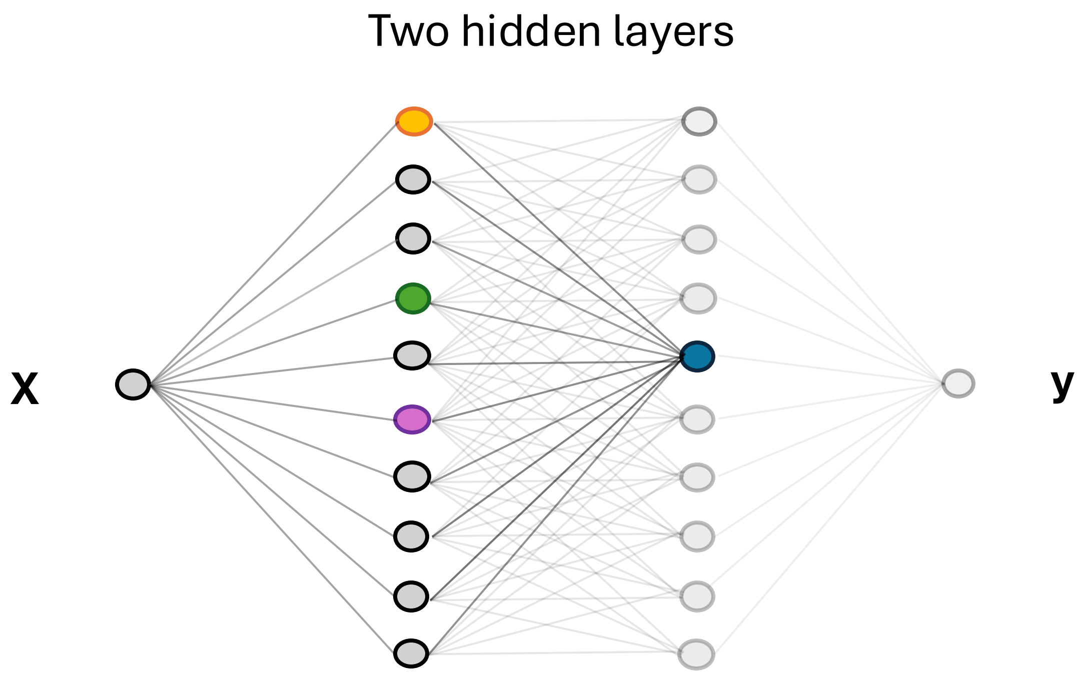
Why don’t neural networks always over fit?

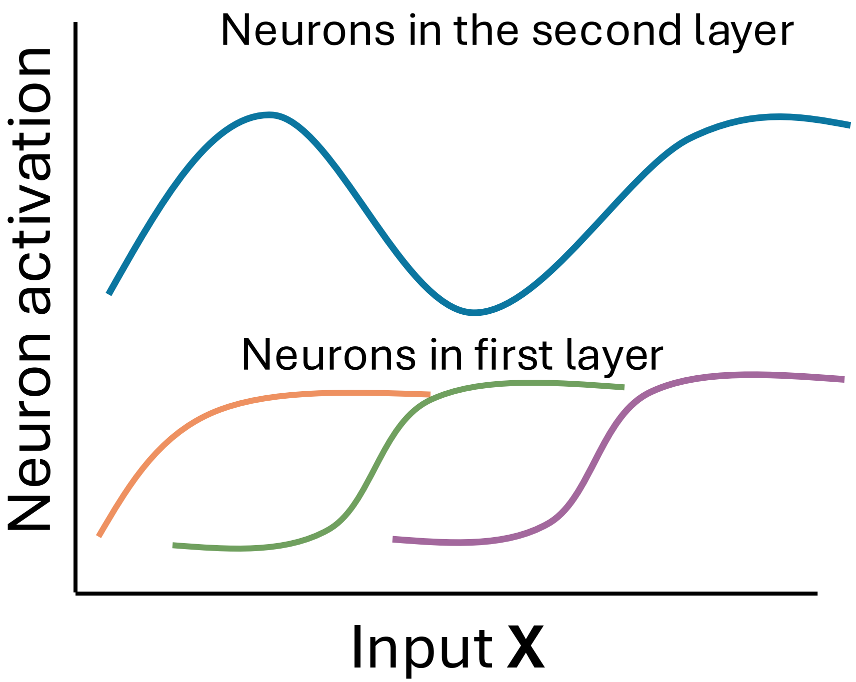


Deep neural networks can exhibit ensemble like behavior

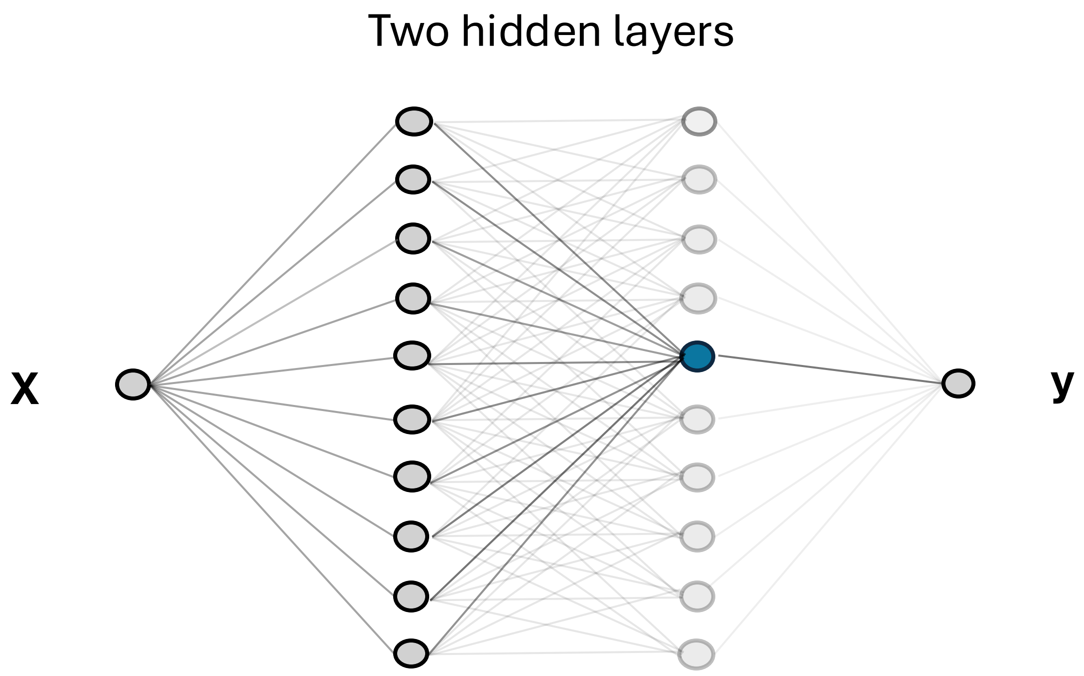


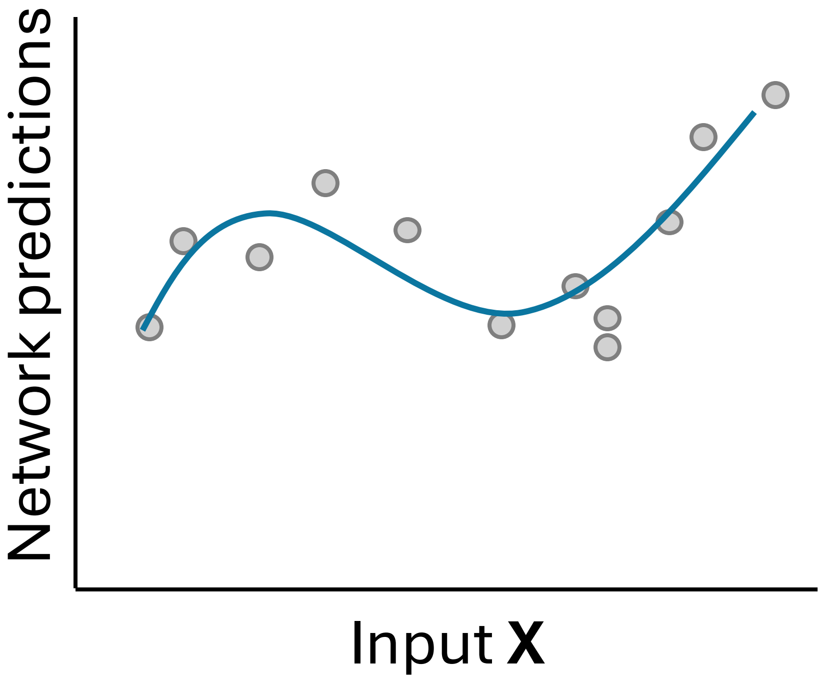
Deep neural networks can exhibit ensemble like behavior



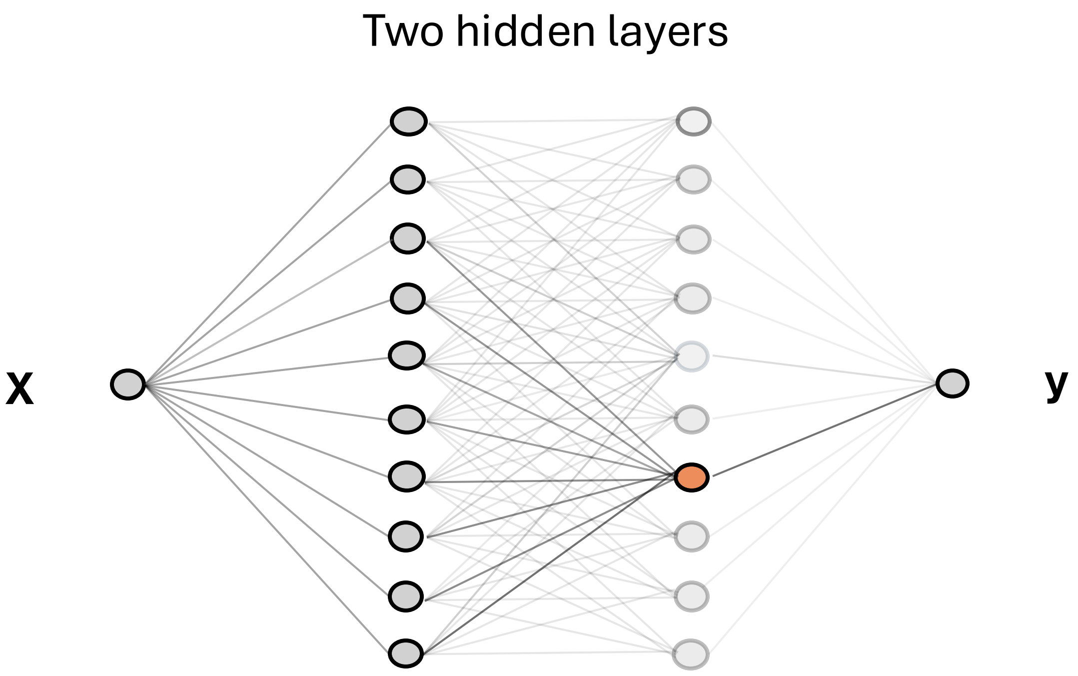


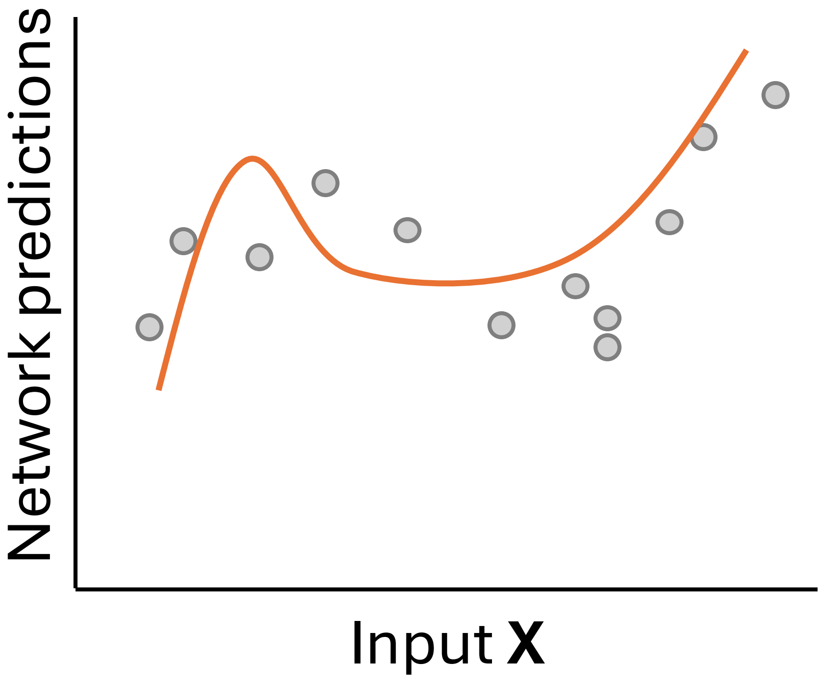
Deep neural networks can exhibit ensemble like behavior



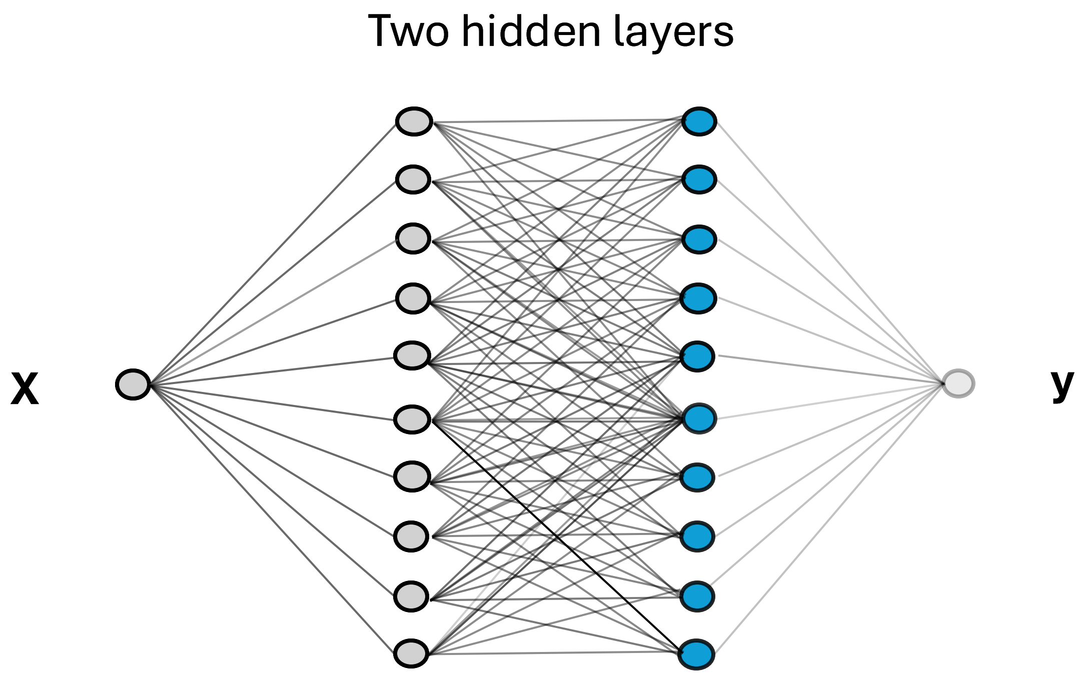


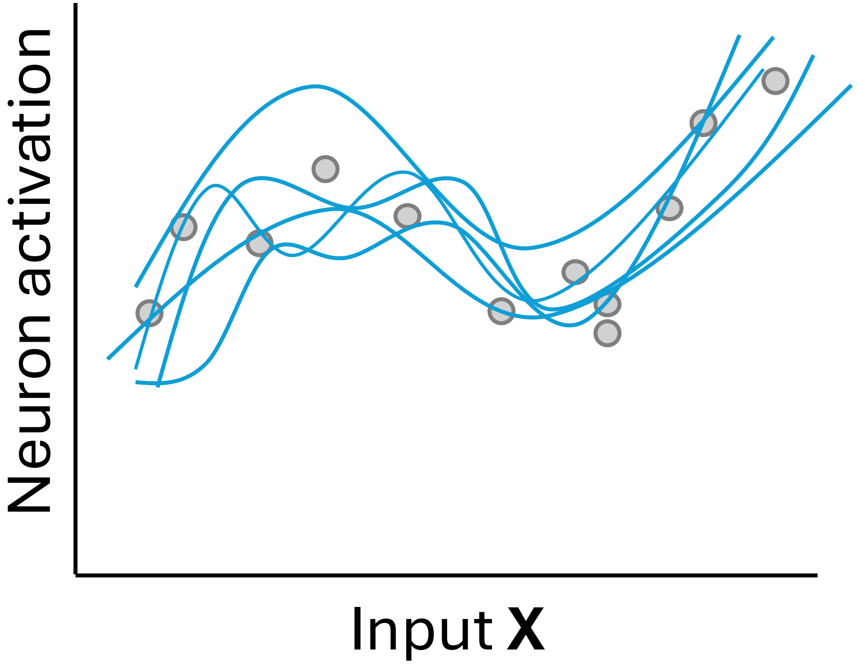
Deep neural networks can exhibit ensemble like behavior



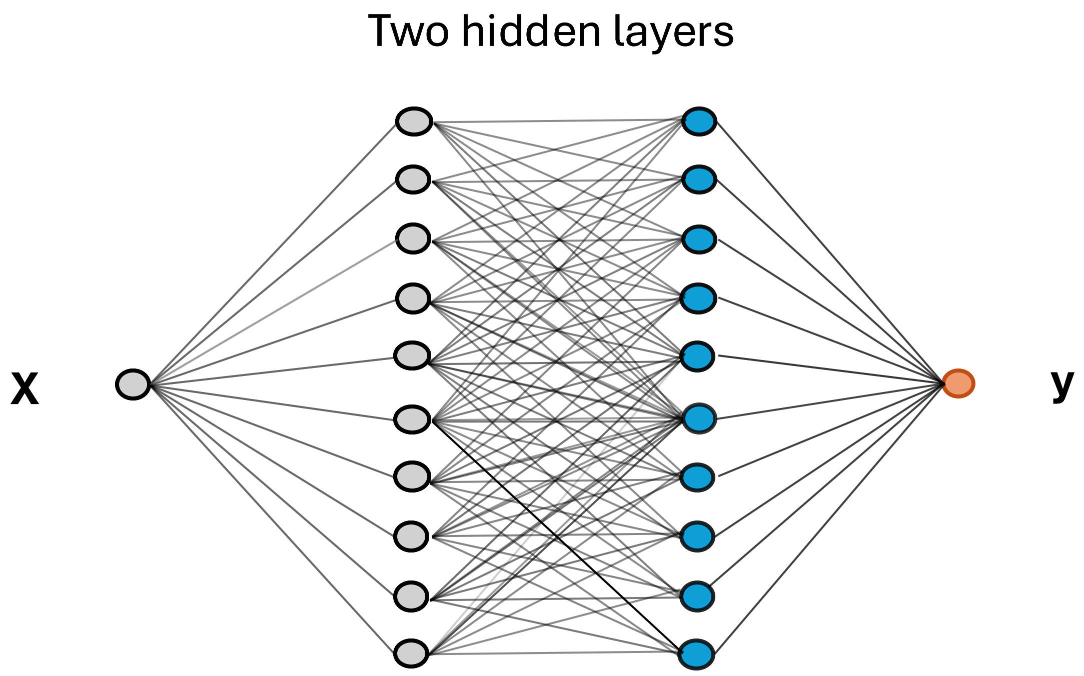


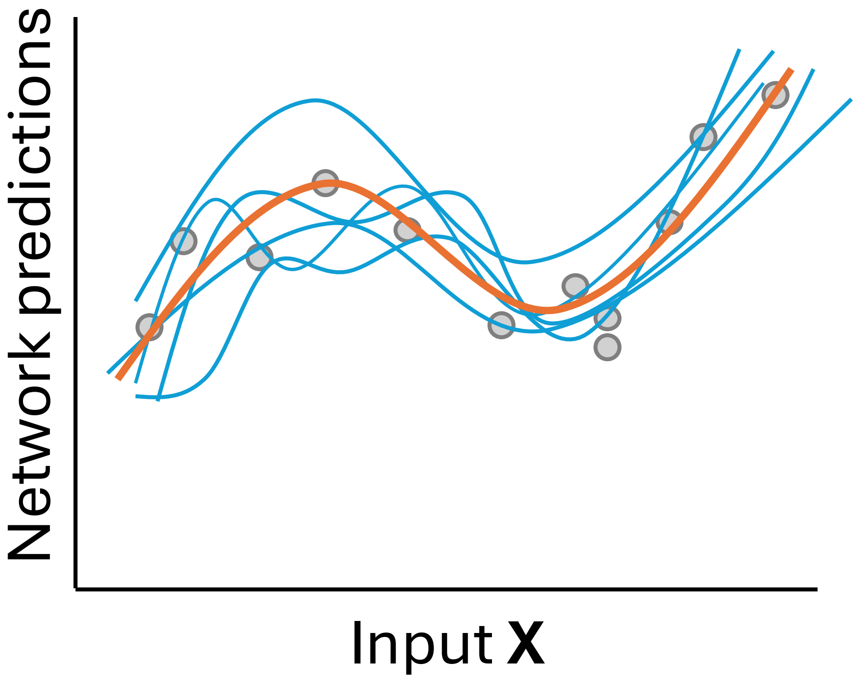
Deep neural networks can exhibit ensemble like behavior





Deep neural networks can exhibit ensemble like behavior





Deep learning so far

1. Neural networks can identify highly complex relationships between inputs and outputs

2. Deep neural networks can exhibit ensemble like behaviors that help the generalize to new data set.

How do we set the weights of a neural network in practice?

1. Initializes the model at random

How do we set the weights of a neural network in practice?

1. Initializes the model at random

2. Compare the model's predictions to a loss function

$$C(\boldsymbol{t}, \boldsymbol{y}) = \text{distance between training } \boldsymbol{t} \text{ and predictions } \boldsymbol{y}$$

$$\text{Squared error loss } - \; C(t, y) = (t - y)^2.$$

How do we set the weights of a neural network in practice?

1. Initializes the model at random

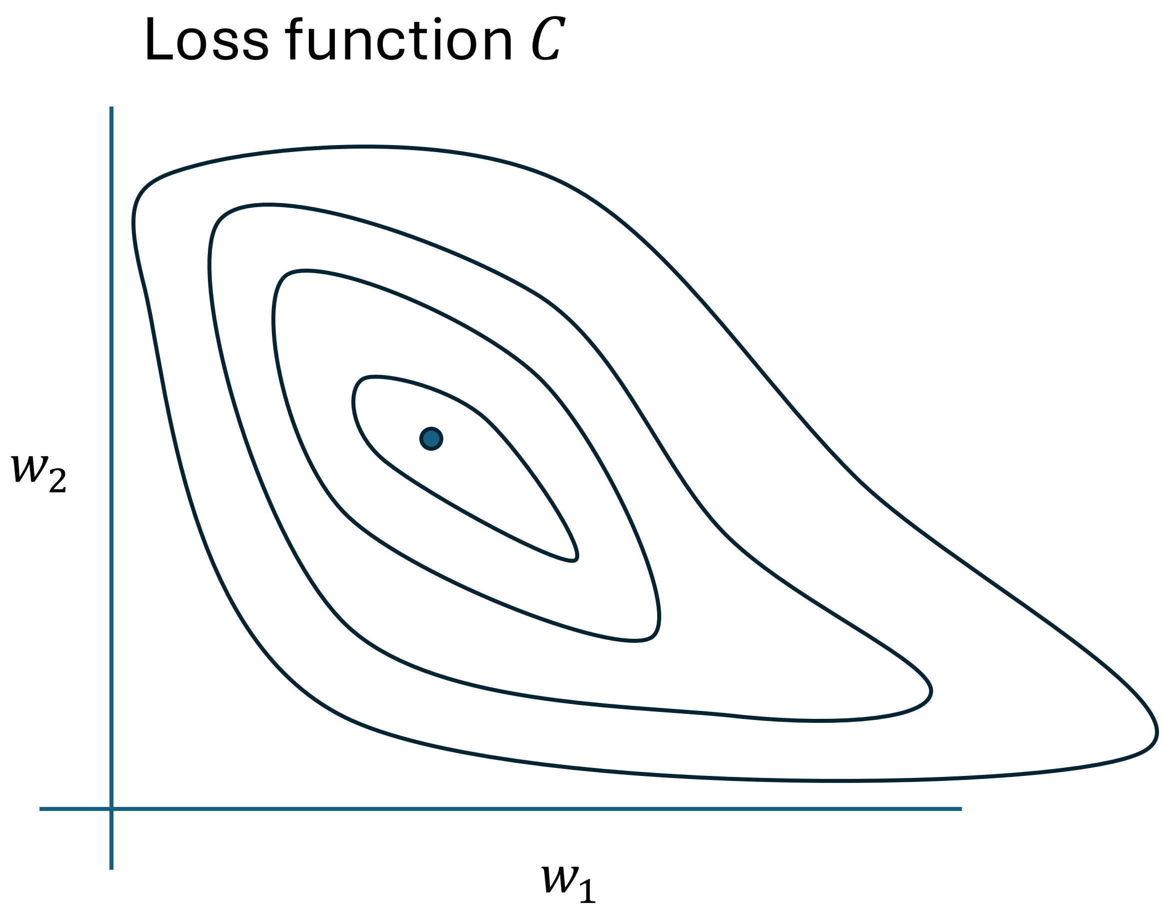
2. Compare the model's predictions to a loss function

$$C(\boldsymbol{t}, \boldsymbol{y}) = \text{distance between training } \boldsymbol{t} \text{ and predictions } \boldsymbol{y}$$

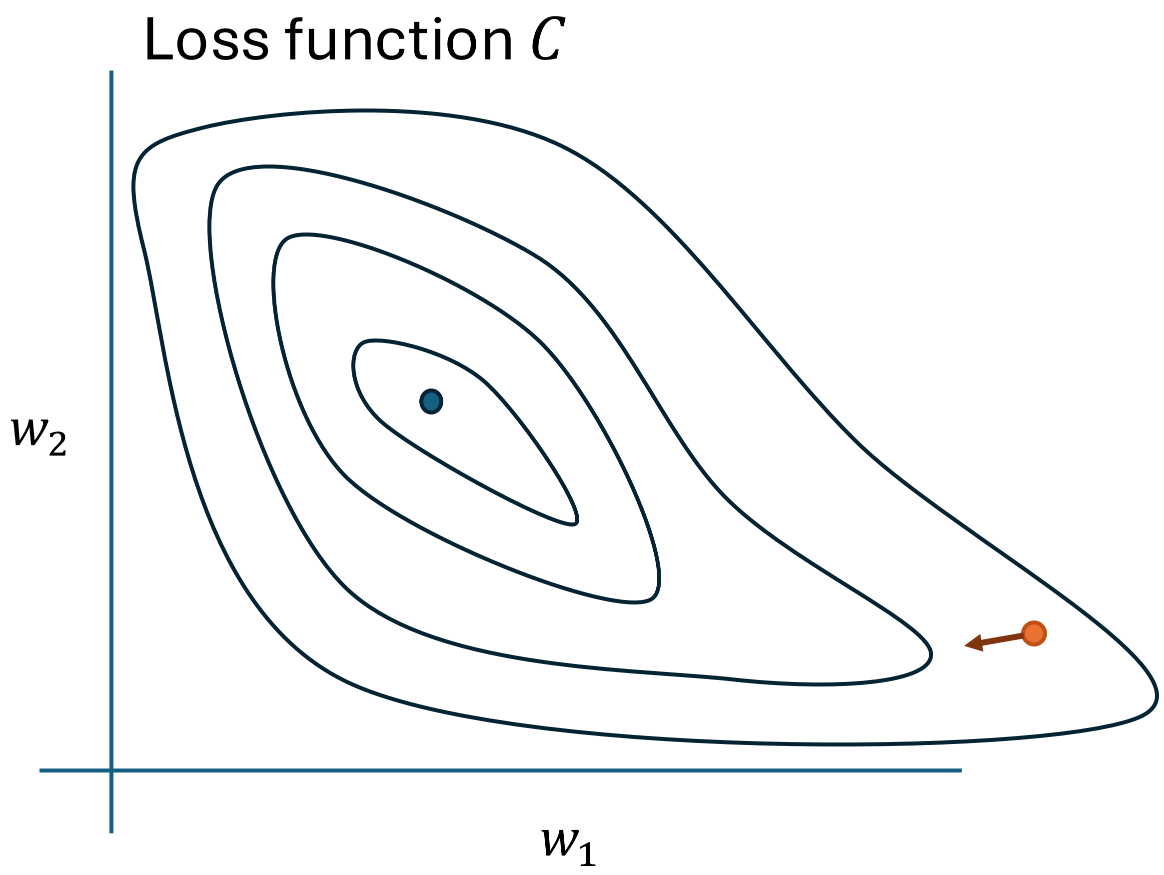
$$\text{Squared error loss } - \ C(t, y) = (t - y)^2.$$

3. Update the weights of the network to reduce the value of the loss function.

Training neural networks with gradient descent



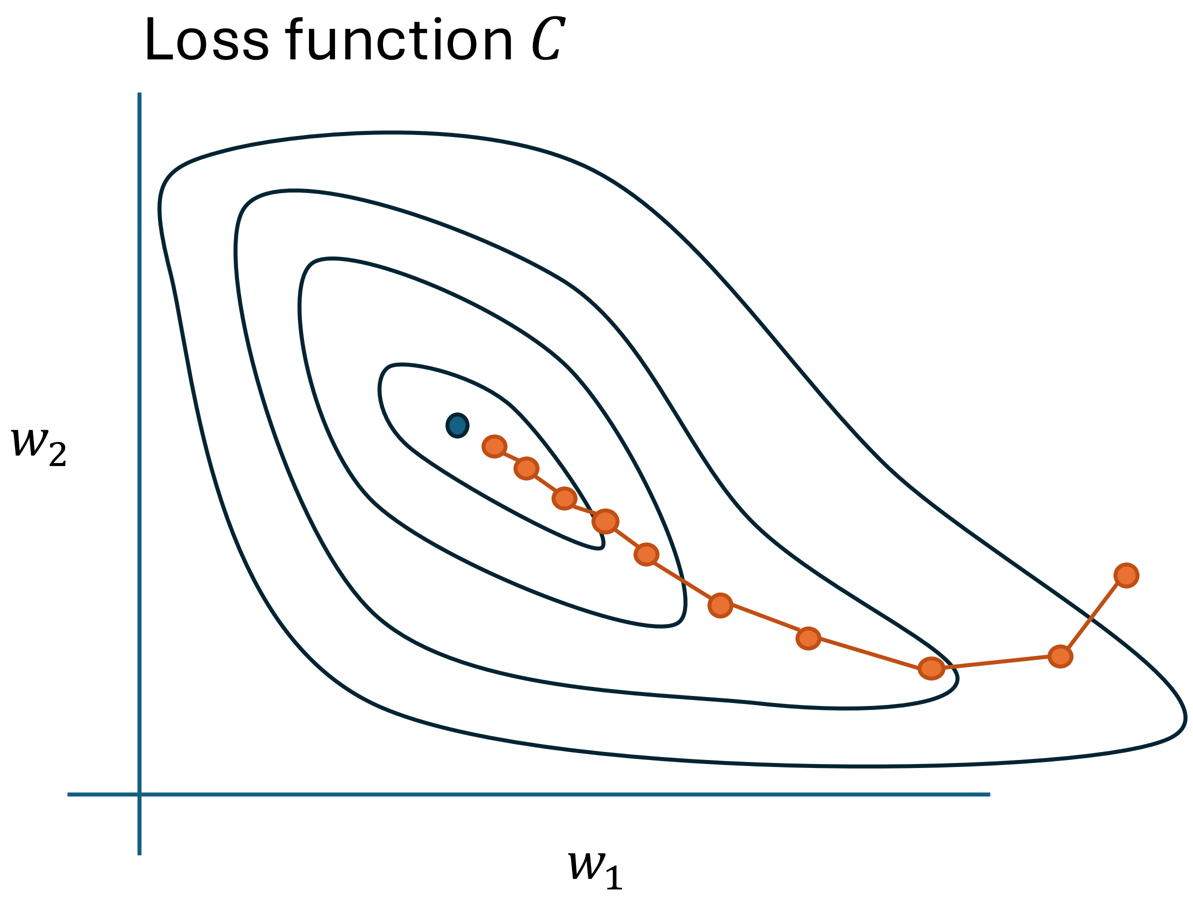
Training neural networks with gradient descent



1. Initializes the parameters in a random location

2. Update the parameters a small amount in the direction that maximizes the effect of that small step on the loss

Training neural networks with gradient descent



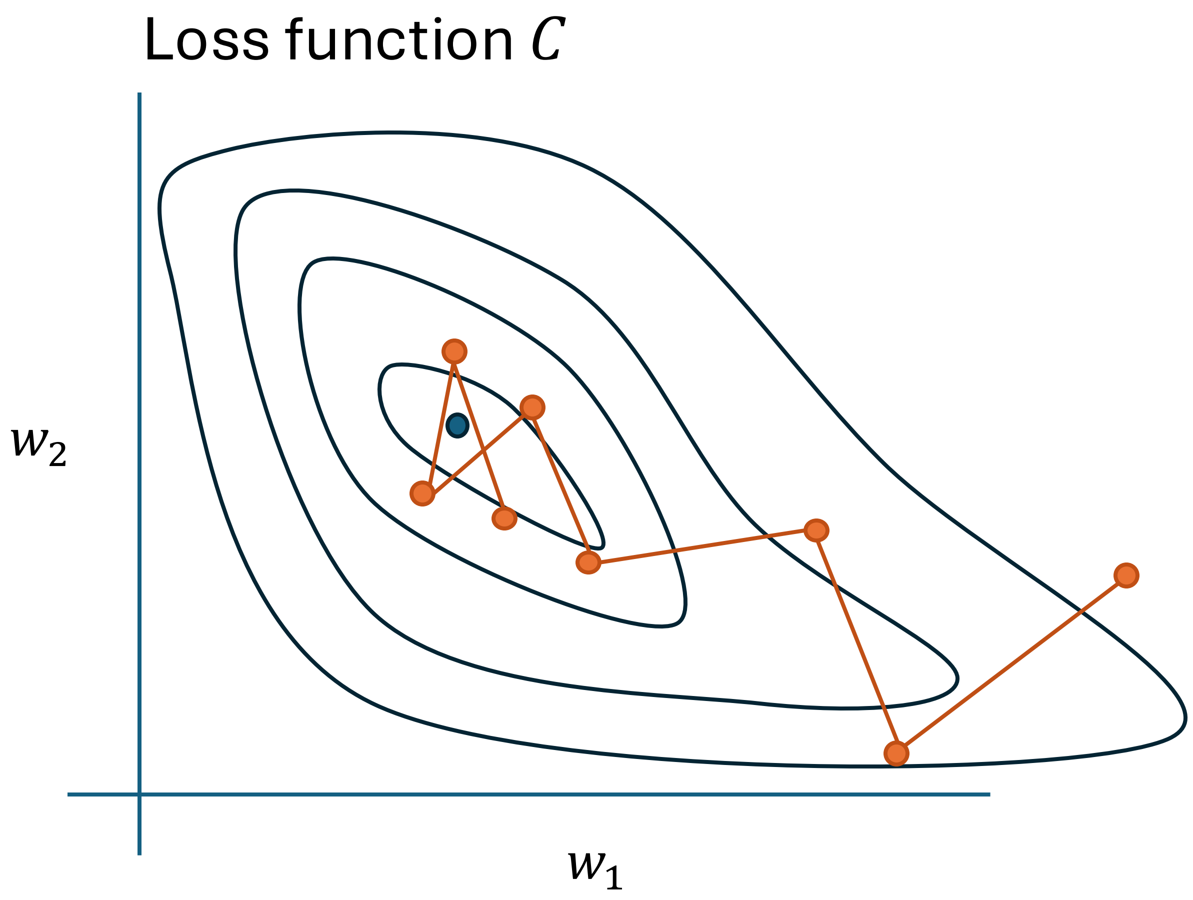
1. Initializes the parameters in a random location

2. Update the parameters a small amount in the direction that maximizes the effect of that small step on the loss

3. Calculate the value of the loss function

4. Repeat 1-3 until the loss stops reducing between iterations

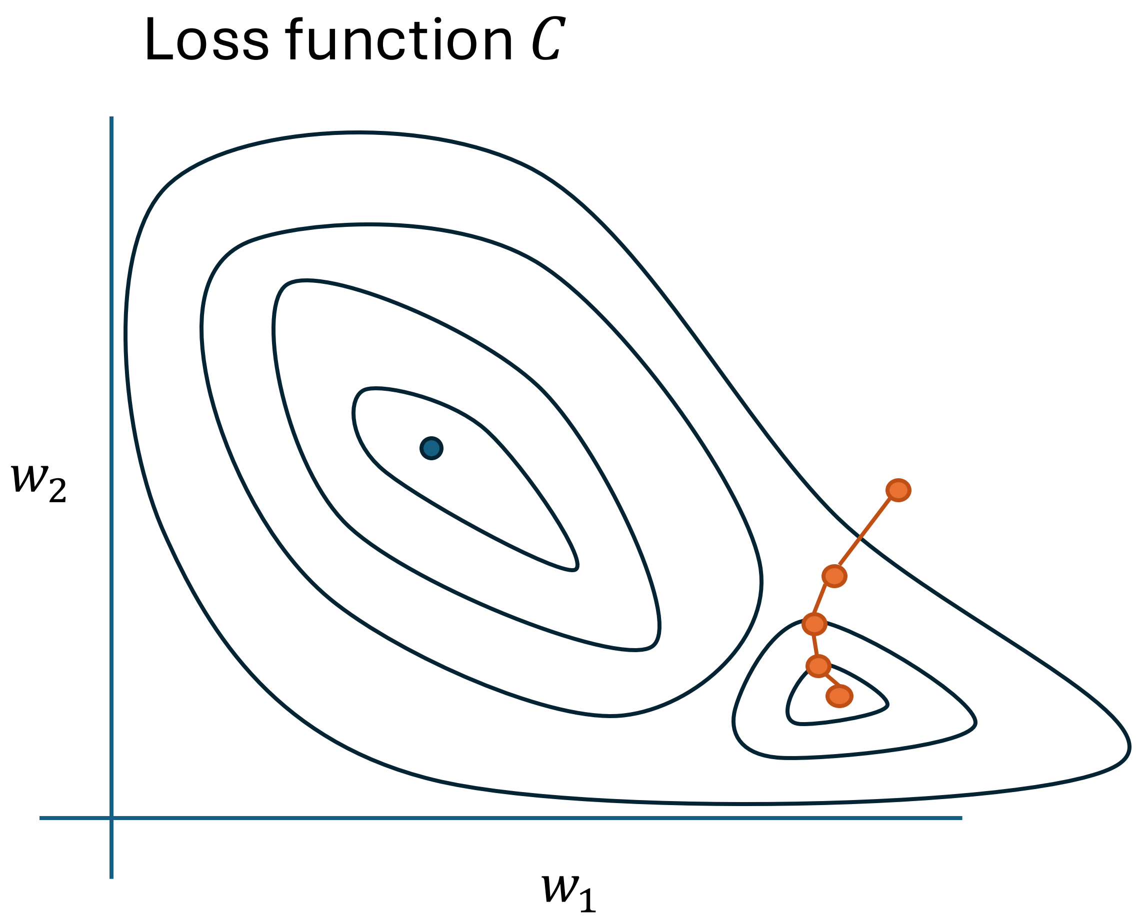
Training neural networks with gradient descent



The gradient descent algorithm need not converge on the best of all possible solutions

Too large of steps cause the algorithm to jump around the optimal solution

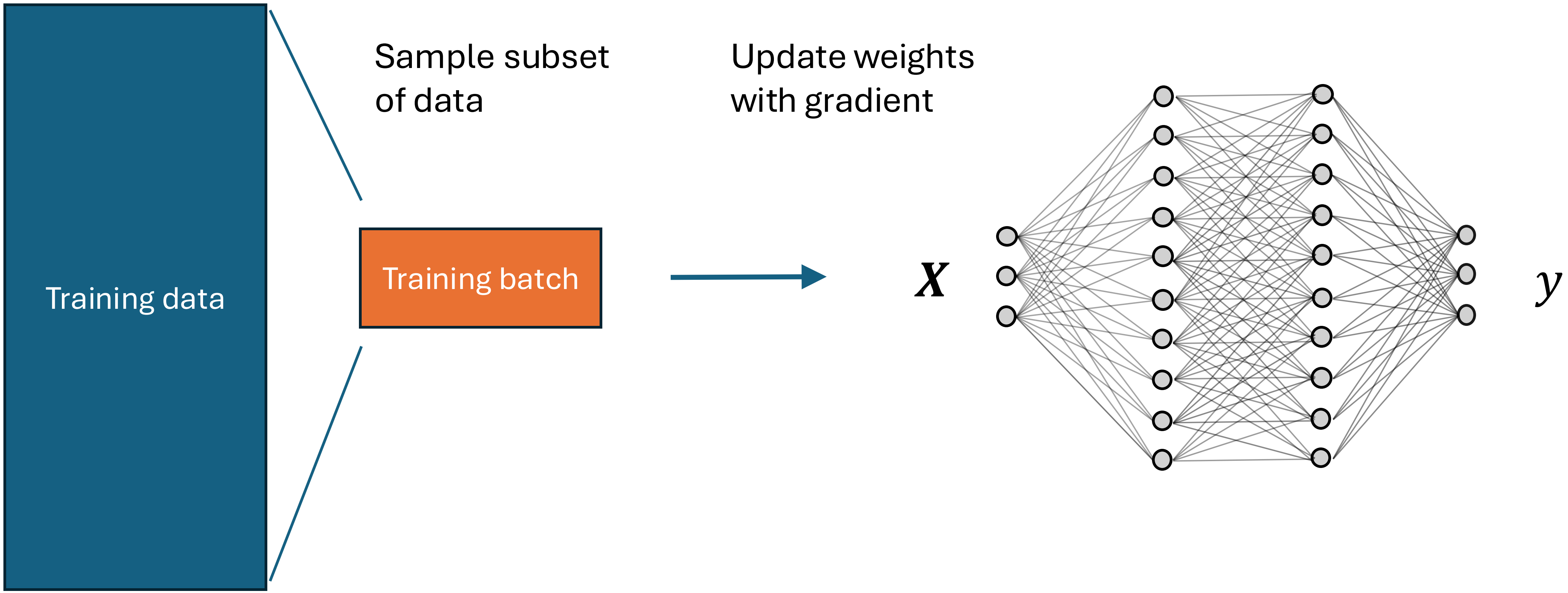
Training neural networks with gradient descent



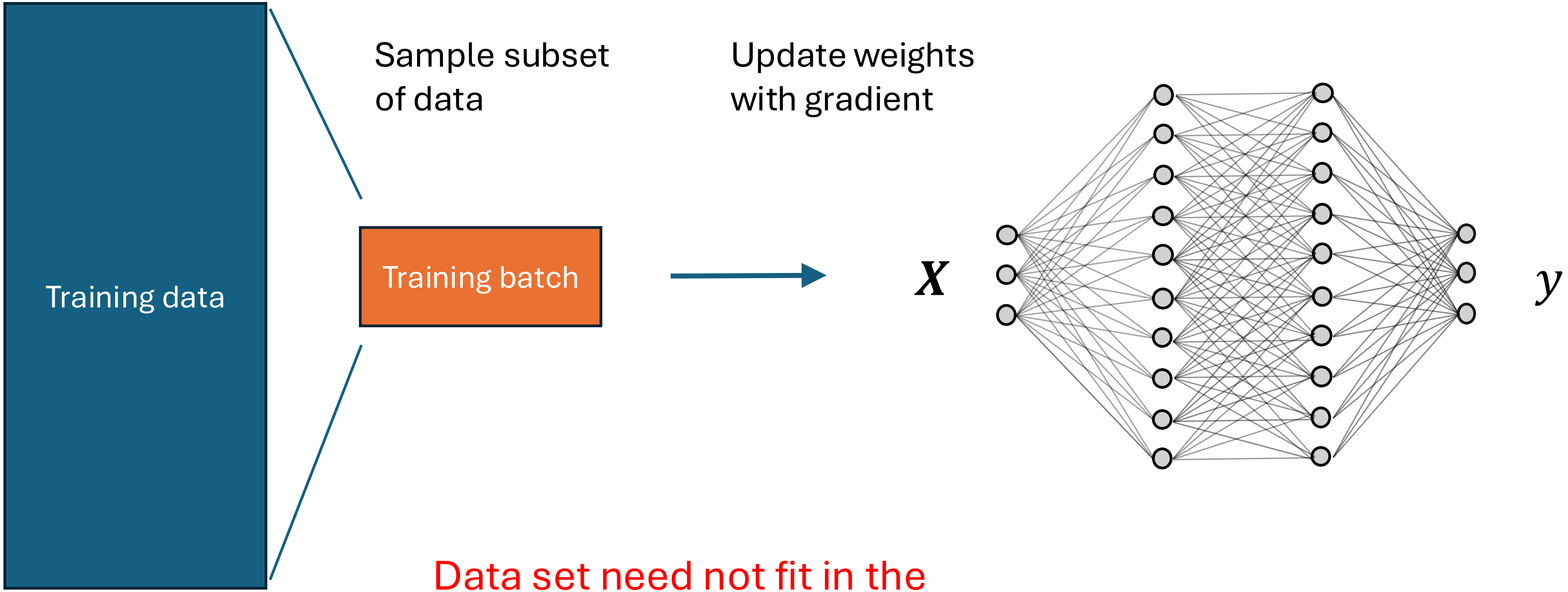
The gradient descent algorithm need not converge on the best of all possible solutions

Gradient descent can get caught at local minimum solutions that don't maximize performance

Scaling neural networks to big data: stochastic gradient descent



Scaling neural networks to big data: stochastic gradient descent



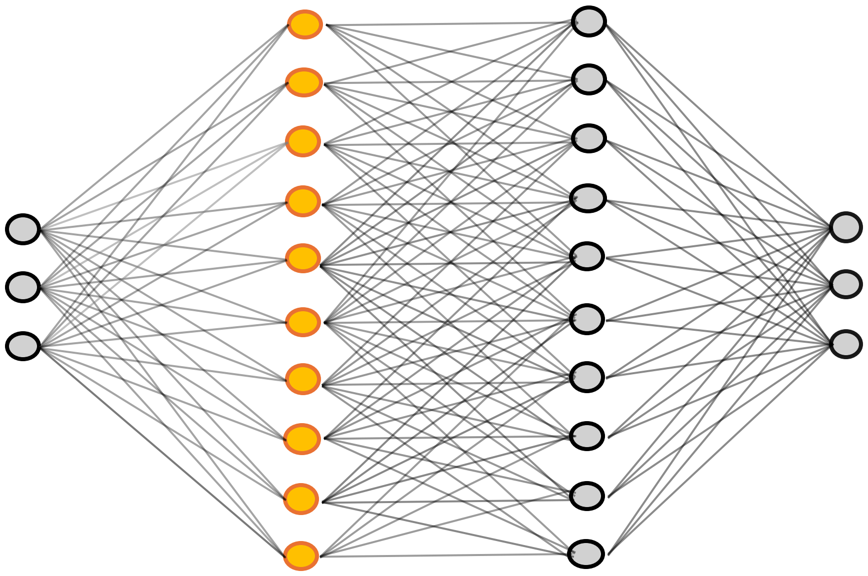
Data set need not fit in the computers working memory!

Scaling neural networks to big data: Graphics process in units

Evaluating neural networks requires computing large sums for each neuron in every layer of the work

- Matrix multiplication

$$\boldsymbol{z}_1 = \begin{bmatrix} b_1^1 & w_{1,1}^1 & w_{2,1}^1 & w_{3,1}^1 \\ b_2^1 & w_{1,2}^1 & w_{2,2}^1 & w_{3,2}^1 \\ \dots & & & \\ b_h^1 & w_{1,h}^1 & w_{2,h}^1 & w_{3,h}^1 \end{bmatrix} \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$$



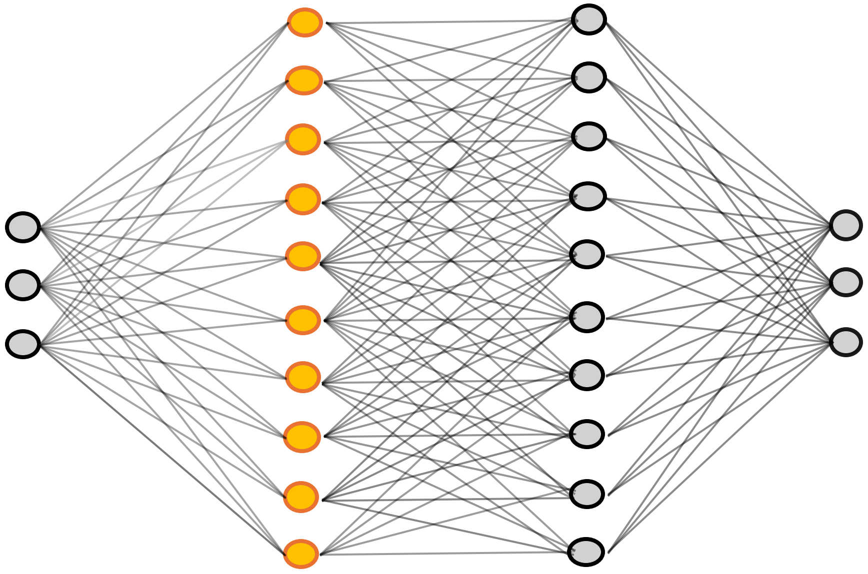
Scaling neural networks to big data: Graphics process in units

Evaluating neural networks requires computing large sums for each neuron in every layer of the work

- Matrix multiplication

These sums do not depend on one another and can be computed at the same time

$$\boldsymbol{z}_1 = \begin{bmatrix} b_1^1 & w_{1,1}^1 & w_{2,1}^1 & w_{3,1}^1 \\ b_2^1 & w_{1,2}^1 & w_{2,2}^1 & w_{3,2}^1 \\ \dots & & & \\ b_h^1 & w_{1,h}^1 & w_{2,h}^1 & w_{3,h}^1 \end{bmatrix} \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$$



Scaling neural networks to big data: Graphics process in units

Evaluating neural networks requires computing large sums for each neuron in every layer of the work

- Matrix multiplication

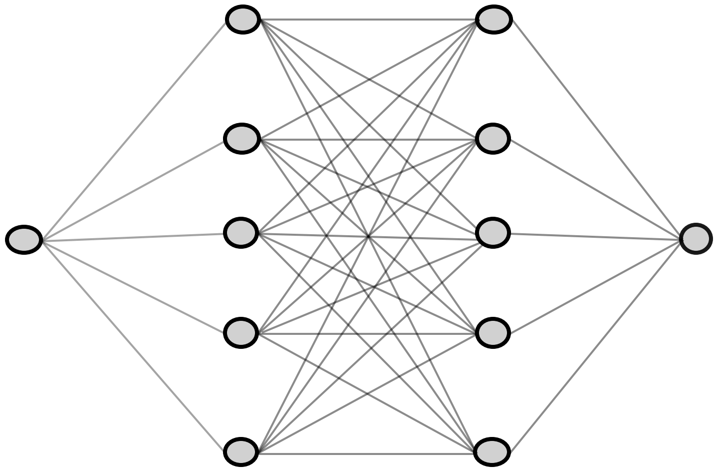
These sums do not depend on one another and can be computed at the same time

Special computer chips called graphic processing units (GPUs) are specially designed for this task!

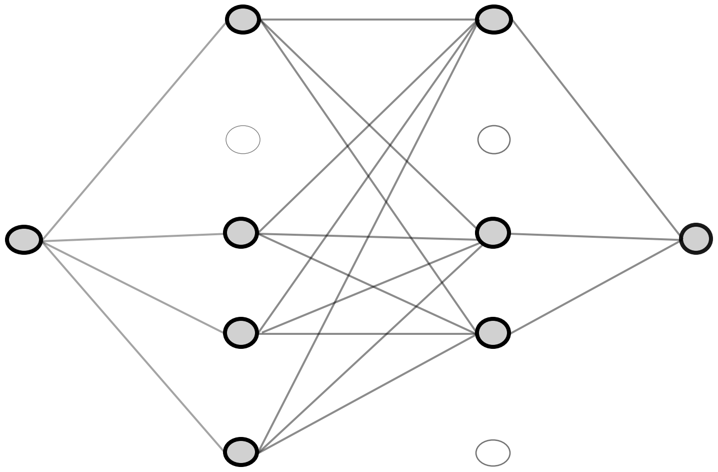
- Increase the speed of training and evaluation by calculating the activations for each neuron in parallel

Reducing overfitting using dropout

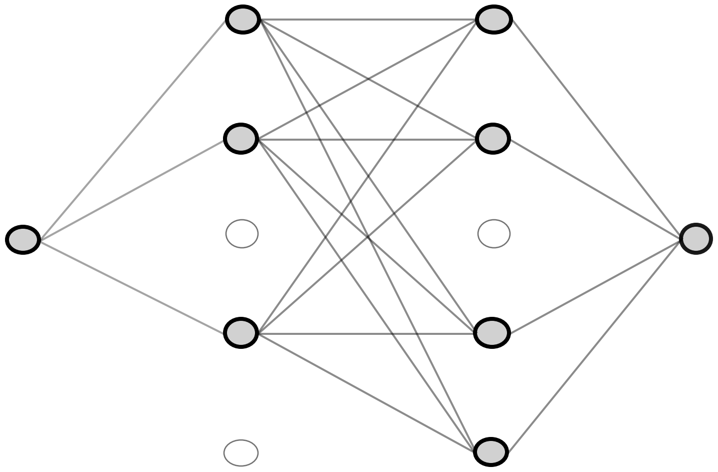
Full network



Subnetwork



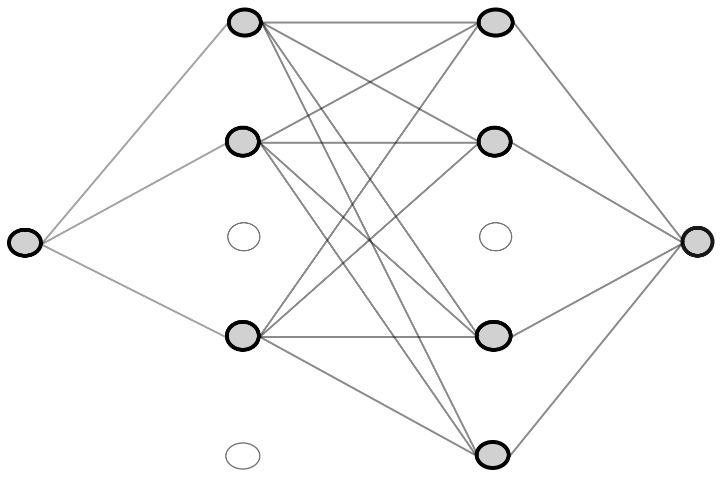
Subnetwork



Reducing overfitting using dropout

At each training step remove each neuron with probability p

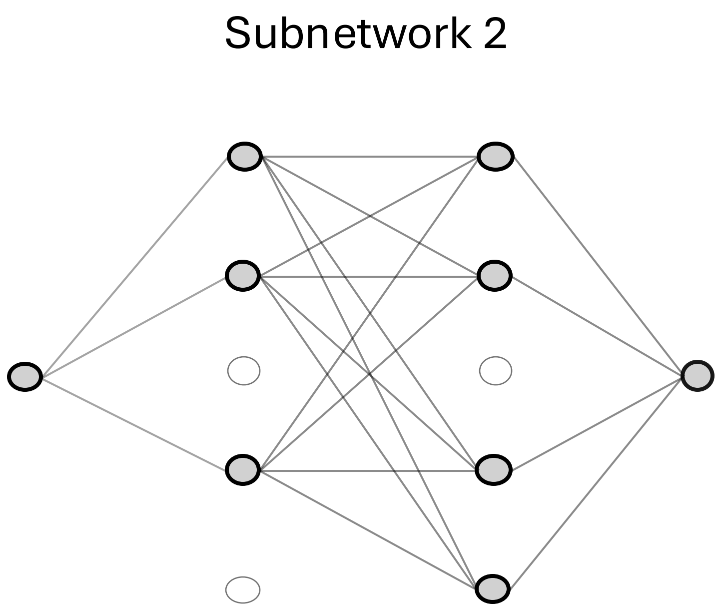
Subnetwork 2



Reducing overfitting using dropout

At each training step remove each neuron with probability p

Scale the remaining weights in the network by a factor $\frac{1}{p}$



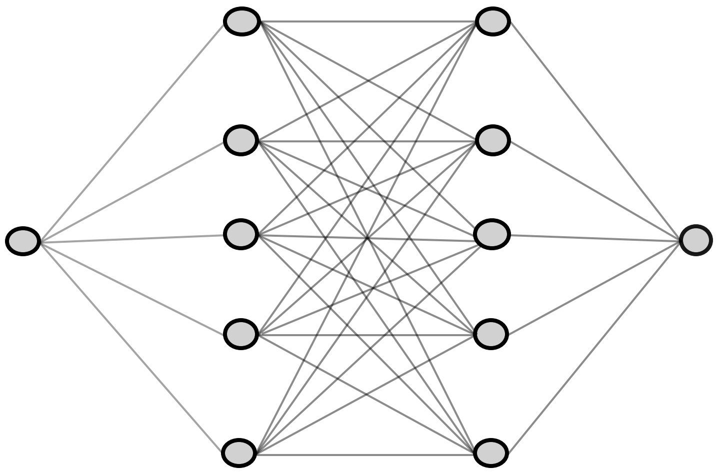
Reducing overfitting using dropout

At each training step remove each neuron with probability p

Scale the remaining weights in the network by a factor $\frac{1}{p}$

Make prediction using all of the neurons without drop out.

Full network

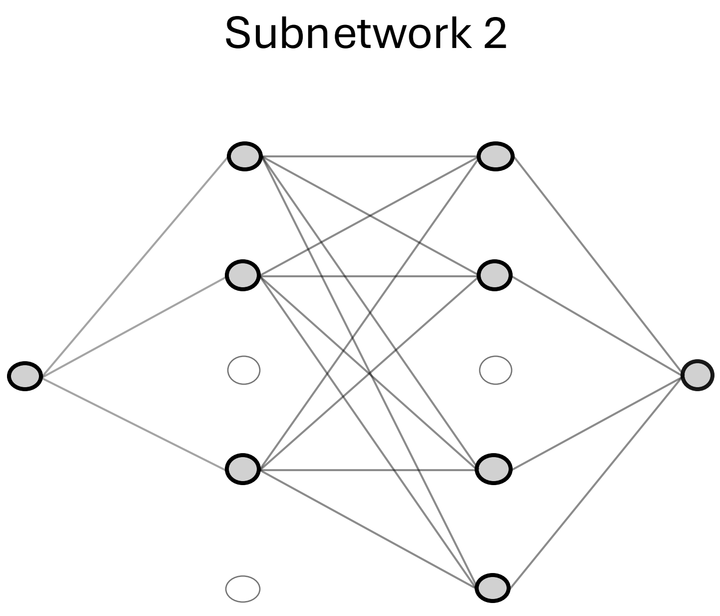


Reducing overfitting using dropout

Prevents neurons from "co adapting"

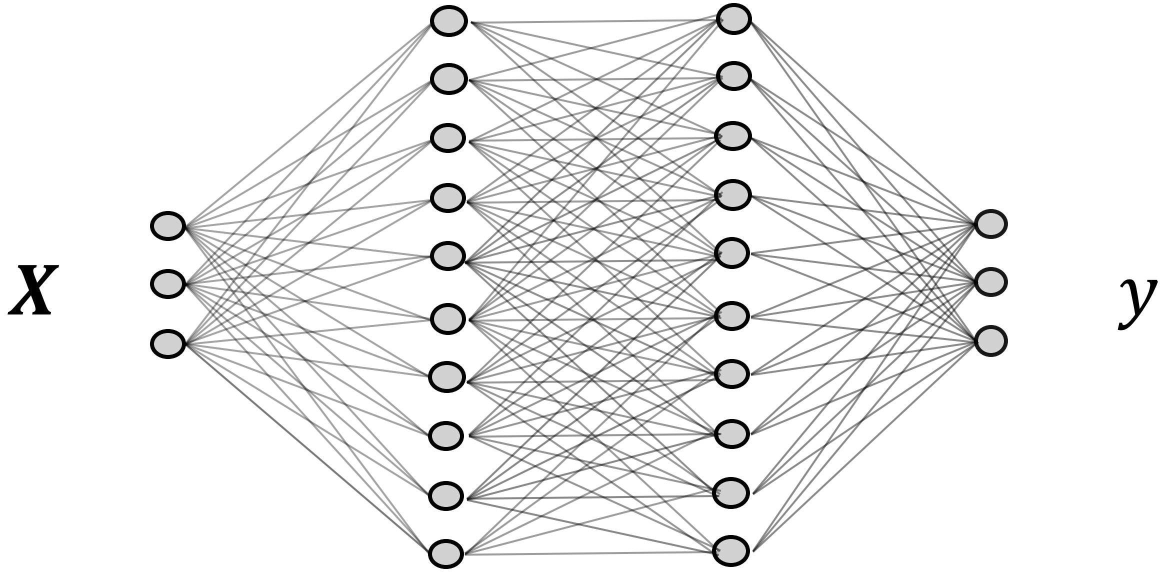
- Learning complex interrelationship between neuron activations

When combined with stochastic gradient descent each sub-network is trained on different subsets of data



Neural networks can be combined in different architectures to address unique problems and data types

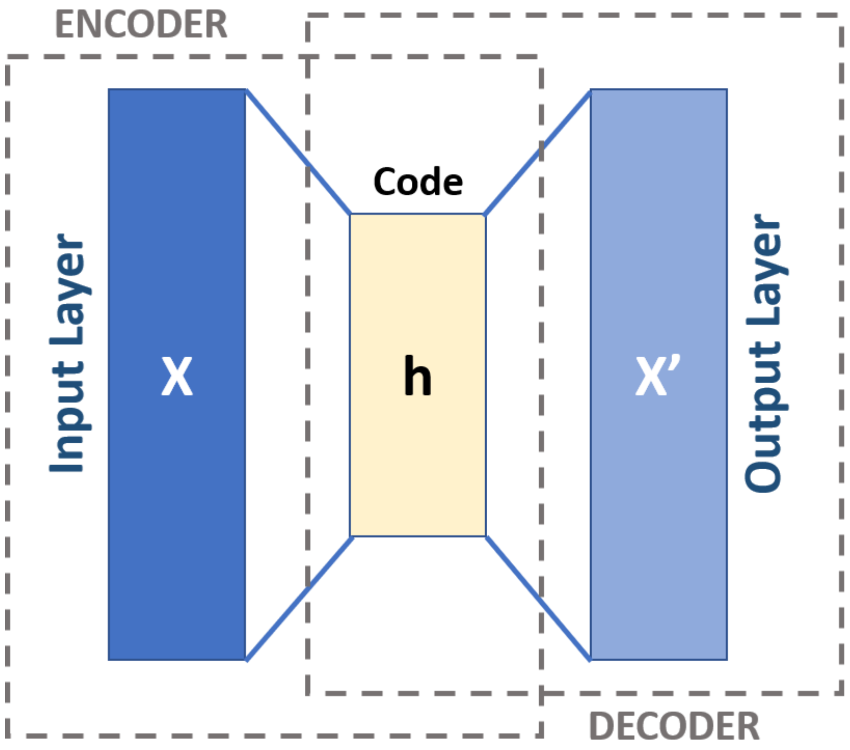
Feed forward networks (multi-layer perceptron) are designed for classification and regression with vector inputs $\boldsymbol{X}$



Neural networks can be combined in different architectures to address unique problems and data types

Unsupervised learning with autoencoders

Input ***X***



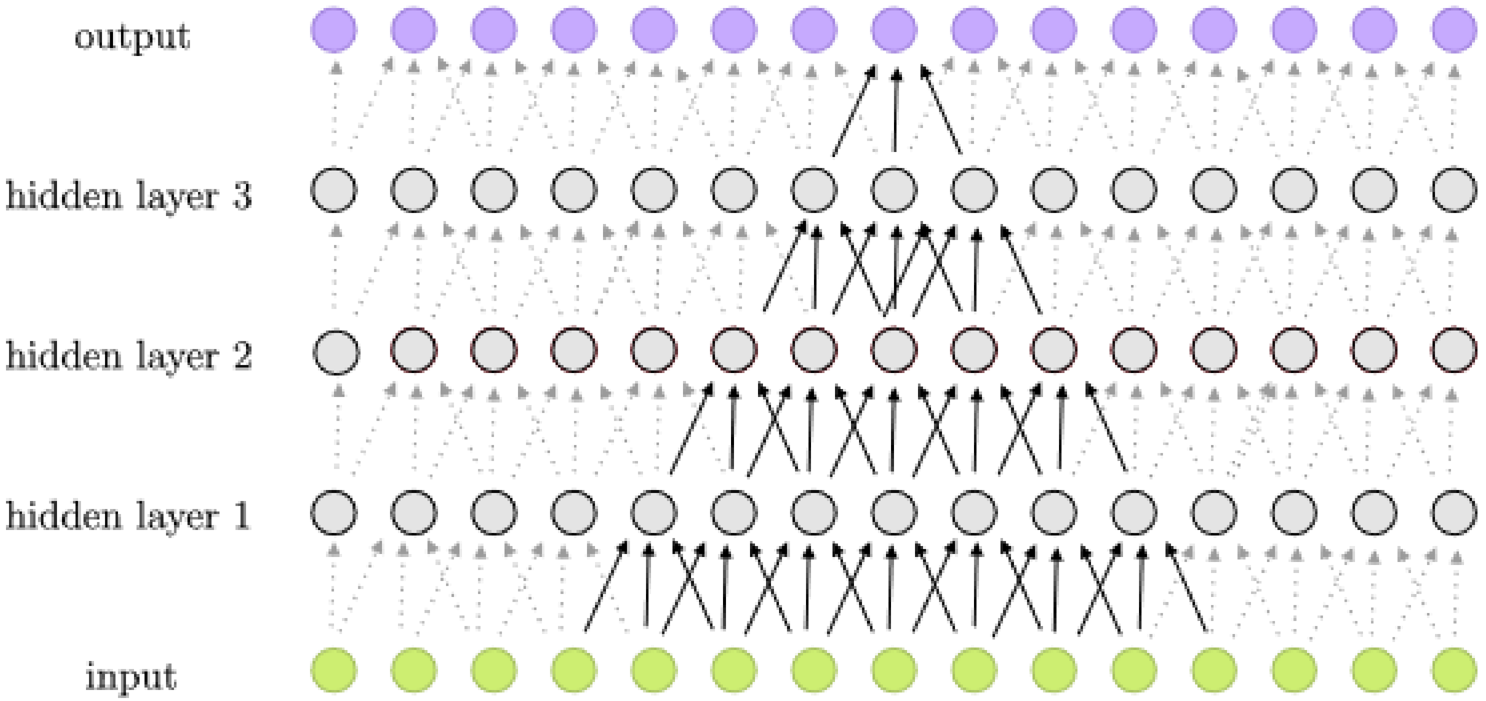
Output ***X***

Simplified representation of the input!

https://en.wikipedia.org/wiki/Autoencoder

Neural networks can be combined in different architectures to address unique problems and data types

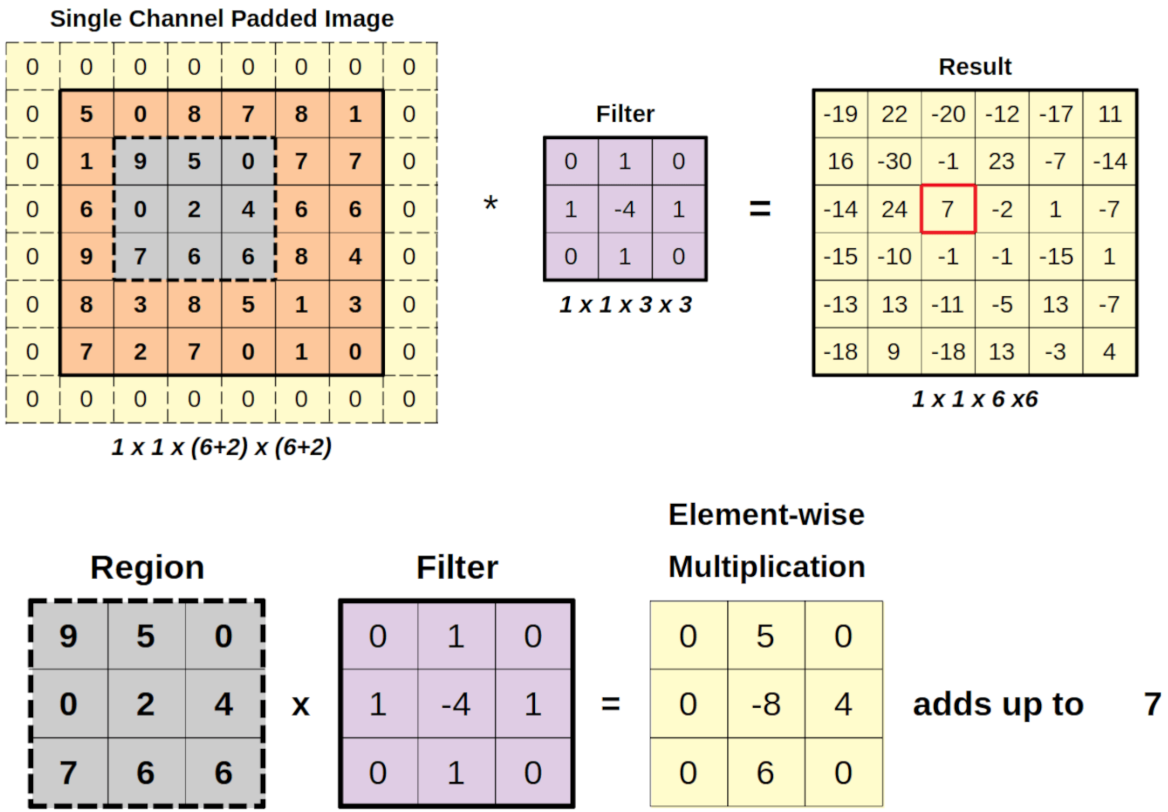
Image and spatial data with convolutional neural networks



https://en.wikipedia.org/wiki/Convolutional_neural_network

Neural networks can be combined in different architectures to address unique problems and data types

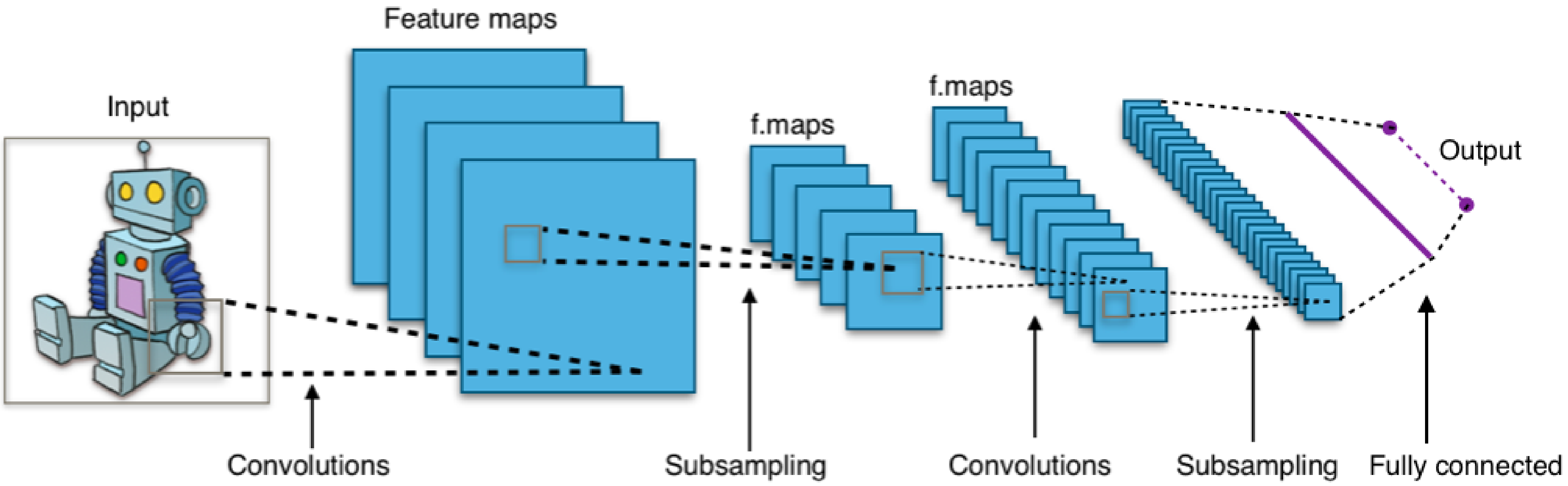
Image and spatial data with convolutional neural networks



https://en.wikipedia.org/wiki/Convolutional_neural_network

Neural networks can be combined in different architectures to address unique problems and data types

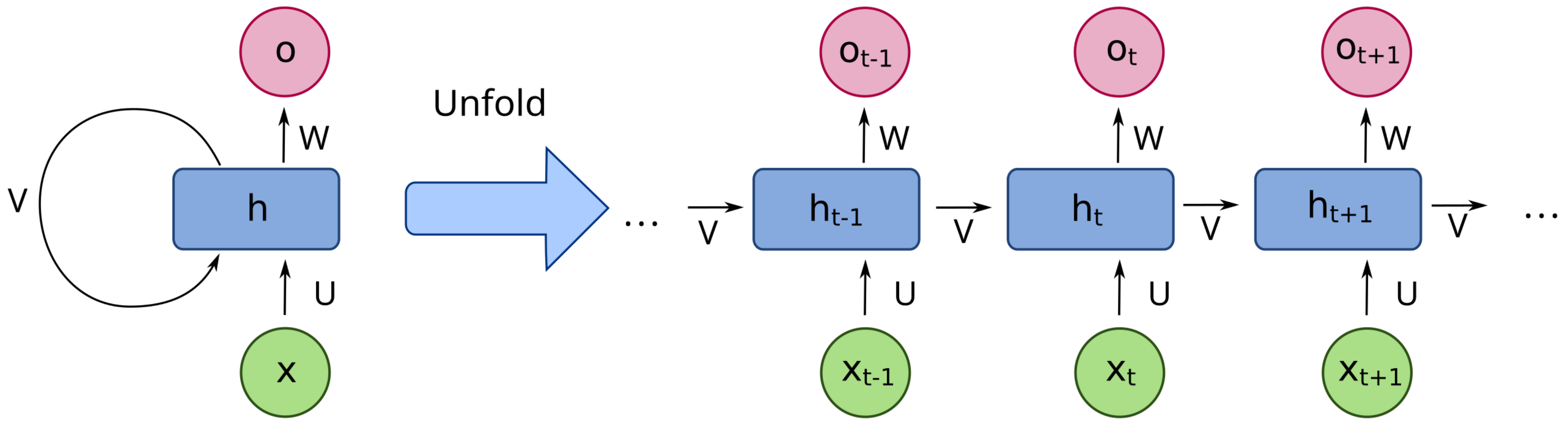
Image and spatial data with convolutional neural networks



https://en.wikipedia.org/wiki/Convolutional_neural_network

Neural networks can be combined in different architectures to address unique problems and data types

Sequential data with recurrent neural networks



Deep learning overview

Deep neurula networks are an efficient way of representing complex high dimensional functions

Complex deep networks often generalize well on new data due to the backward bending bias variance trade off

Neural networks are a useful tool that can form the basis of more complex models.