LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

# Introduction to Machine Learning

## Chapter 3: Deep Learning- MLP

**Bernd Bischl**

Department of Statistics – LMU Munich

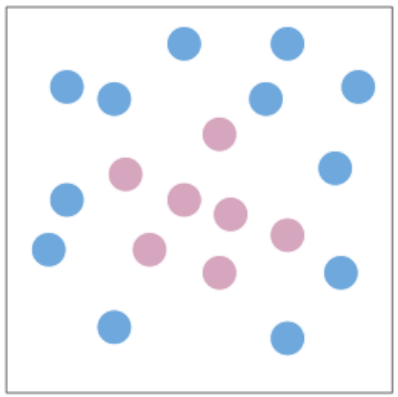
Winter term 2021

# MOTIVATION

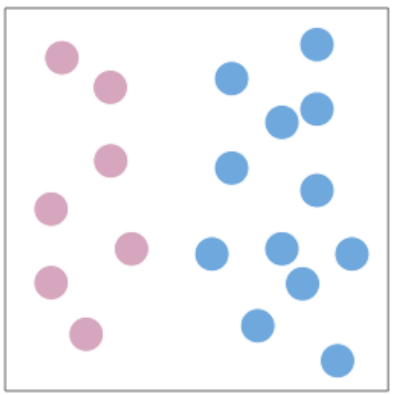
- The graphical way of representing simple functions/models, like logistic regression. Why is that useful?
- Because individual neurons can be used as building blocks of more complicated functions.
- Networks of neurons can represent extremely complex hypothesis spaces.
- Most importantly, it allows us to define the “right” kinds of hypothesis spaces to learn functions that are more common in our universe in a data-efficient way (see Lin, Tegmark et al. 2016).

# MOTIVATION

- As a single neuron is restricted to learning only linear decision boundaries, its performance on the following task is quite poor:

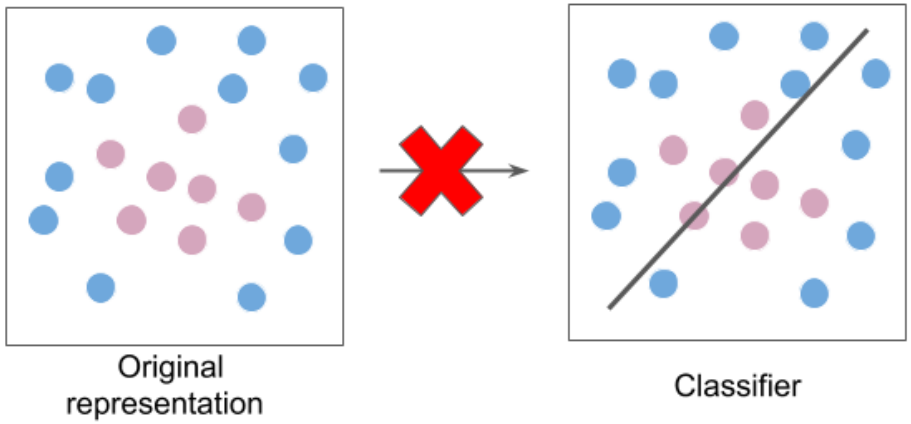


- However, the neuron can easily separate the classes if the original features are transformed (e.g., from Cartesian to polar coordinate):

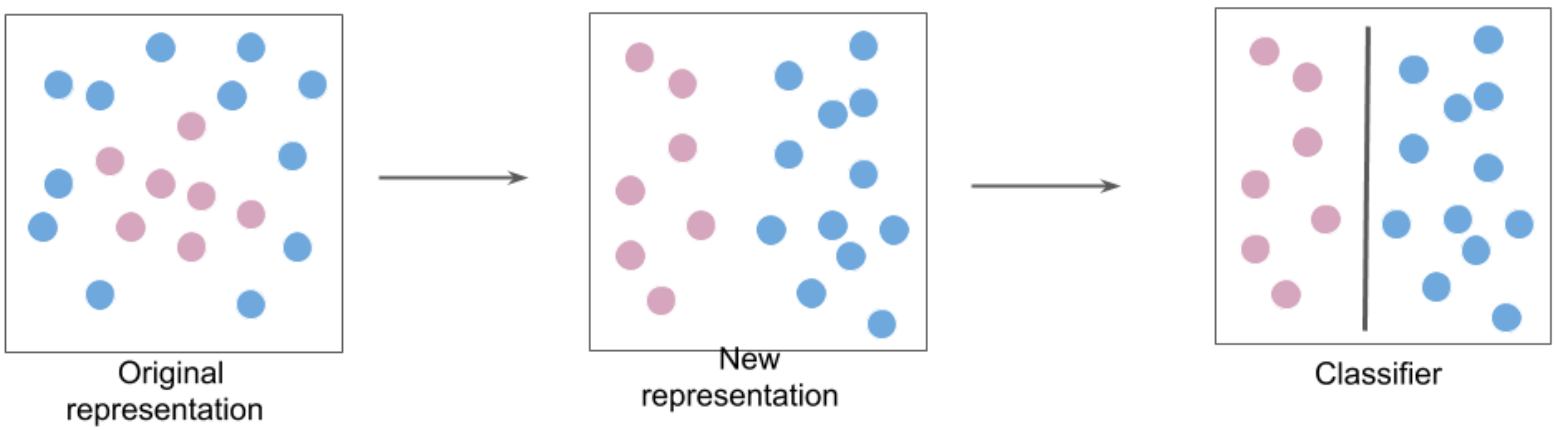


# MOTIVATION

- Instead of classifying the data in the original representation,

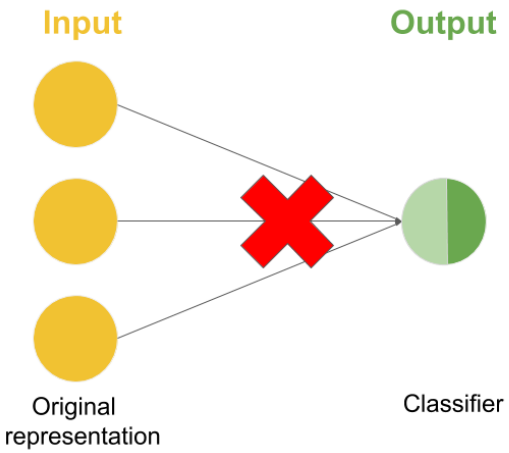



- we classify it in a new feature space.

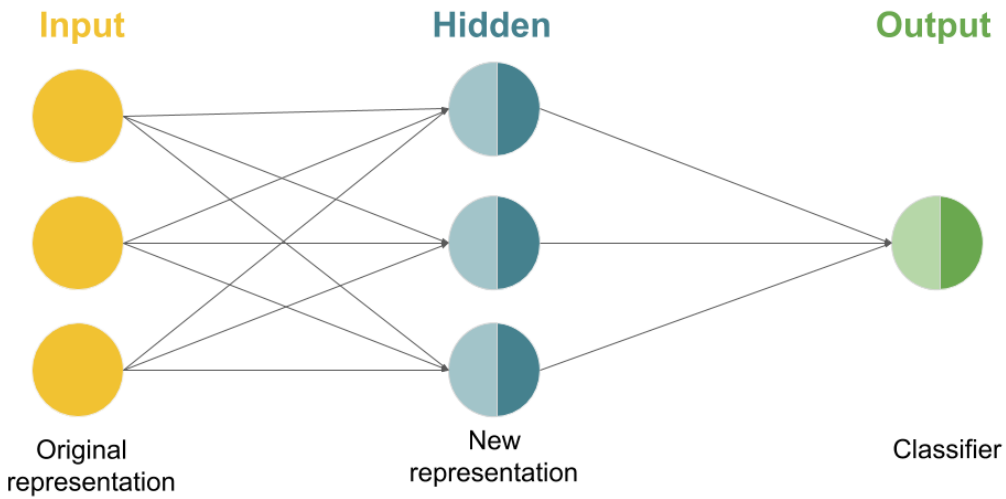


# MOTIVATION

- Analogously, instead of a single neuron,




- we use more complex networks.



# REPRESENTATION LEARNING

- It is therefore *very* critical to feed a classifier the “right” features in order for it to perform well.

- Before deep learning took off, features for tasks like machine vision and speech recognition were “hand-designed” by domain experts. This step of the machine learning pipeline is called **feature engineering**.

- The single biggest reason DL is so important is that it automates feature engineering. This is called **representation learning**.

# SINGLE HIDDEN LAYER NETWORKS: NOTATION

**General notation**:

- The network has $m$ hidden neurons $z_1, \ldots, z_m$ with

$$z_j = \sigma(\mathbf{W}_j^\top \mathbf{x} + b_j)$$

  - $z_{in,j} = \mathbf{W}_j^\top \mathbf{x} + b_j$
  - $z_{out,j} = \sigma(z_{in,j}) = \sigma(\mathbf{W}_j^\top \mathbf{x} + b_j)$

  for $j \in \{1, \ldots, m\}$.

# SINGLE HIDDEN LAYER NETWORKS: NOTATION

## General notation: Multiple inputs

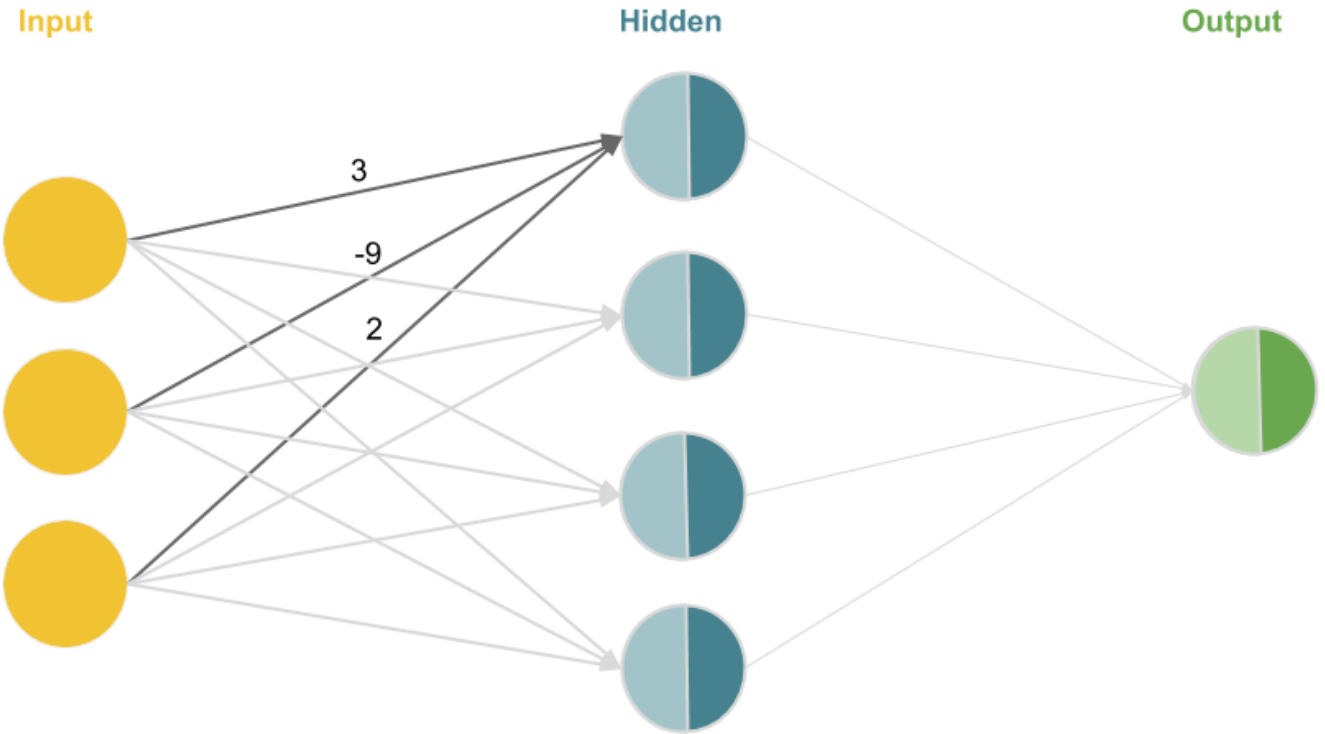
- It is possible to feed multiple inputs to a neural network simultaneously.
- The inputs $\mathbf{x}^{(i)}$, for $i \in \{1, \ldots, n\}$, are arranged as rows in the **design matrix X**.
  - **X** is a $(n \times p)$-matrix.
- The weighted sum in the hidden layer is now computed as $\mathbf{XW} + \boldsymbol{B}$, where,
  - **W**, as usual, is a $(p \times m)$ matrix, and,
  - $\boldsymbol{B}$ is a $(n \times m)$ matrix containing the bias vector **b** (duplicated) as the rows of the matrix.
- The *matrix* of hidden activations $\boldsymbol{Z} = \sigma(\mathbf{XW} + \boldsymbol{B})$
  - $\boldsymbol{Z}$ is a $(n \times m)$ matrix.

# SINGLE HIDDEN LAYER NETWORKS: NOTATION

- The final output of the network, which contains a prediction for each input, is $\tau(\boldsymbol{Z}\mathbf{u} + \boldsymbol{C})$, where
  - $\mathbf{u}$ is the vector of weights of the output neuron, and,
  - $\boldsymbol{C}$ is a $(n \times 1)$ matrix whose elements are the (scalar) bias $c$ of the output neuron.

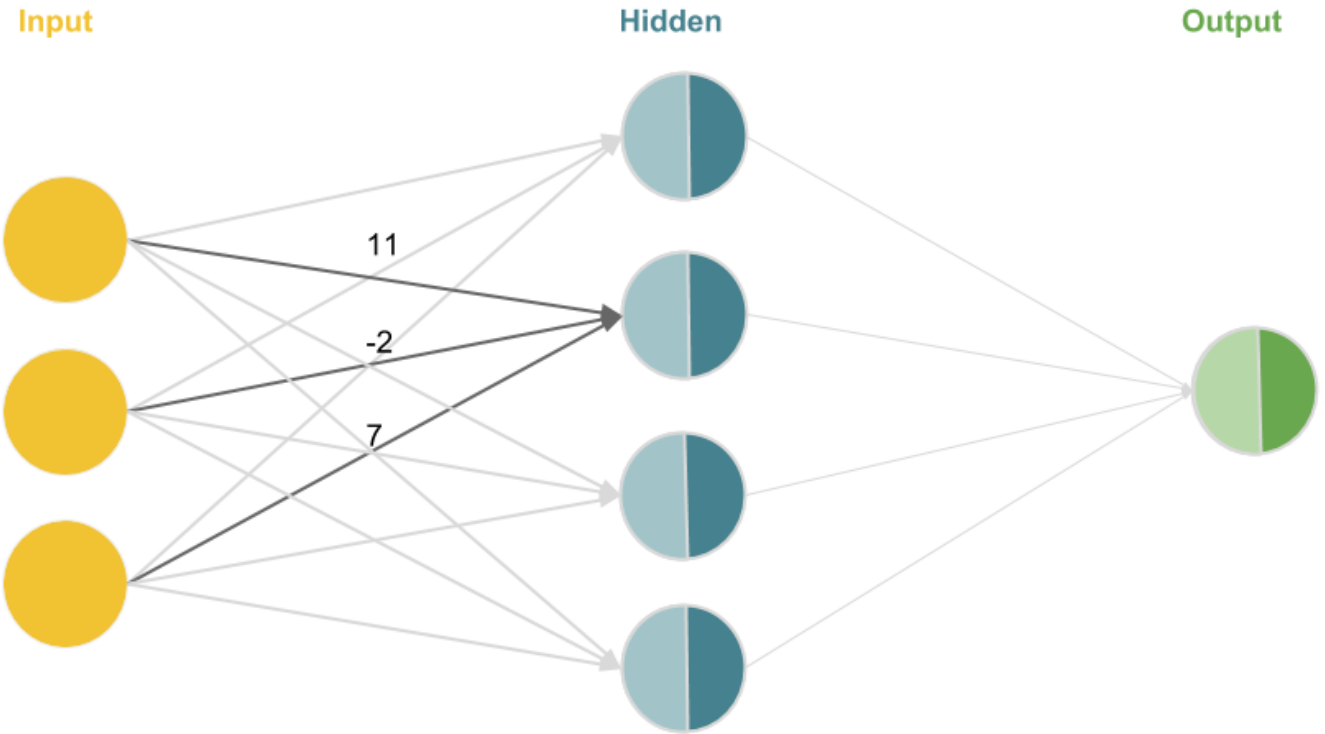
# SINGLE HIDDEN LAYER NETWORKS: EXAMPLE

- Weights (and biases) of the network.



# SINGLE HIDDEN LAYER NETWORKS: EXAMPLE

- Weights (and biases) of the network.




$$\begin{pmatrix} 3 & -9 & 2 \\ \mathbf{11} & \mathbf{-2} & \mathbf{7} \end{pmatrix} \begin{pmatrix} 5 \\ \mathbf{2} \end{pmatrix}$$

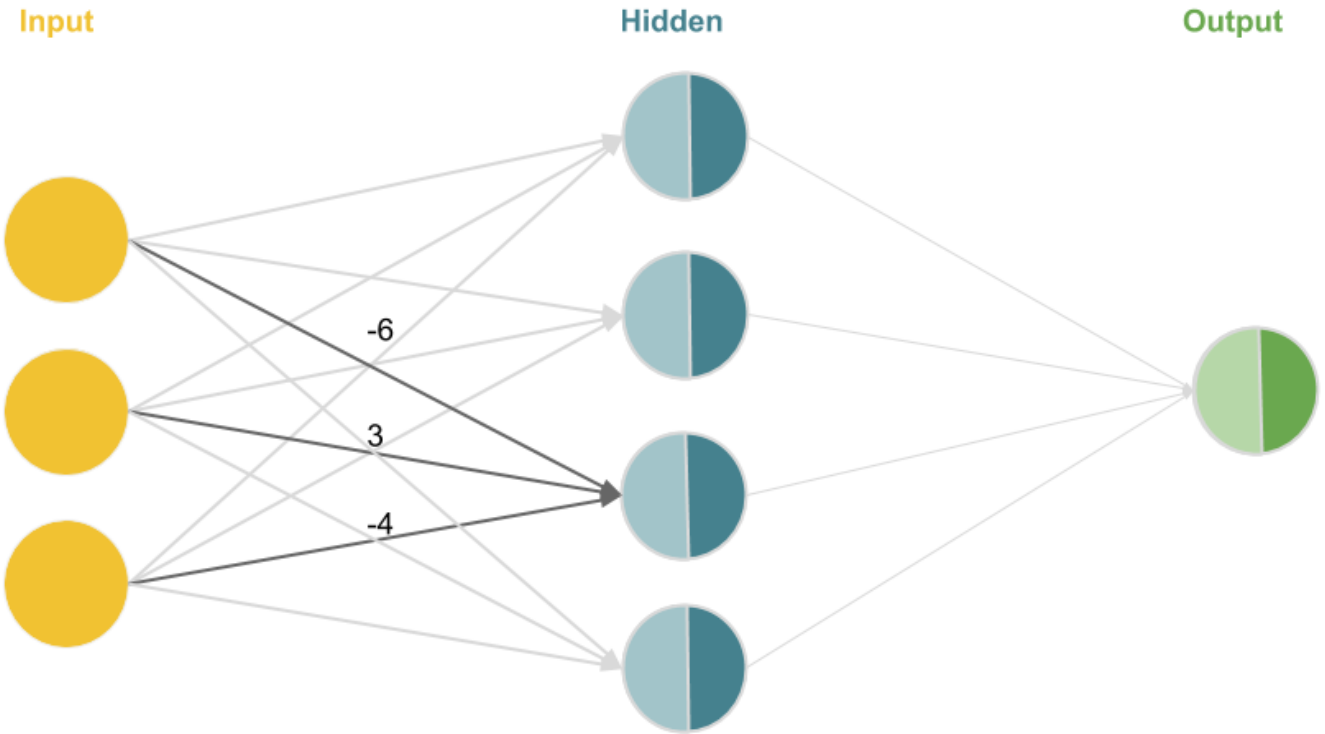
$W^T$ $\mathbf{b}$

$$\begin{pmatrix} & & \end{pmatrix} \begin{pmatrix} \end{pmatrix}$$

$\mathbf{u}^T$ $c$

# SINGLE HIDDEN LAYER NETWORKS: EXAMPLE

- Weights (and biases) of the network.




$$\begin{pmatrix} 3 & -9 & 2 \\ 11 & -2 & 7 \\ \mathbf{-6} & \mathbf{3} & \mathbf{-4} \end{pmatrix} \begin{pmatrix} 5 \\ 2 \\ \mathbf{-1} \end{pmatrix}$$

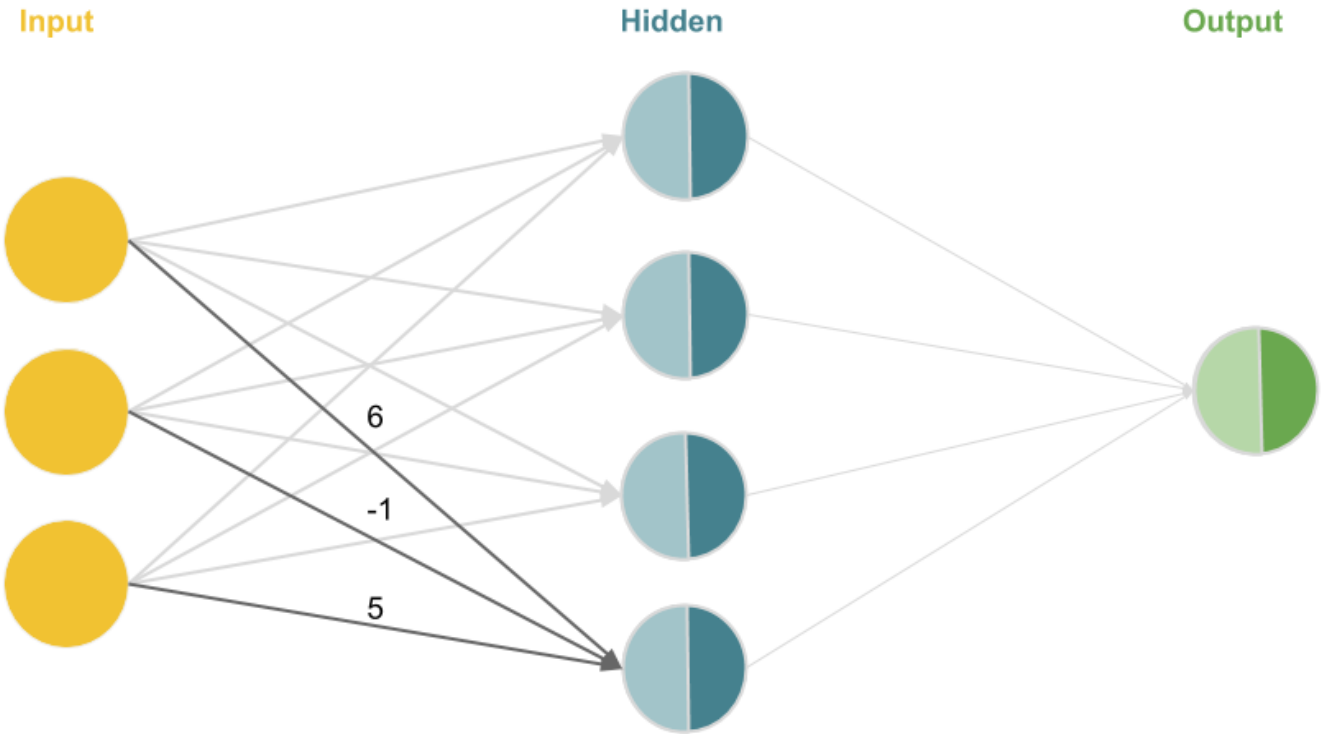
$W^T$ $\mathbf{b}$

$$\begin{pmatrix} \quad \end{pmatrix} \begin{pmatrix} \ \end{pmatrix}$$

$\mathbf{u}^T$ $c$

# SINGLE HIDDEN LAYER NETWORKS: EXAMPLE

- Weights (and biases) of the network.




$$\begin{pmatrix} 3 & -9 & 2 \\ 11 & -2 & 7 \\ -6 & 3 & -4 \\ \mathbf{6} & \mathbf{-1} & \mathbf{5} \end{pmatrix} \begin{pmatrix} 5 \\ 2 \\ -1 \\ \mathbf{1} \end{pmatrix}$$

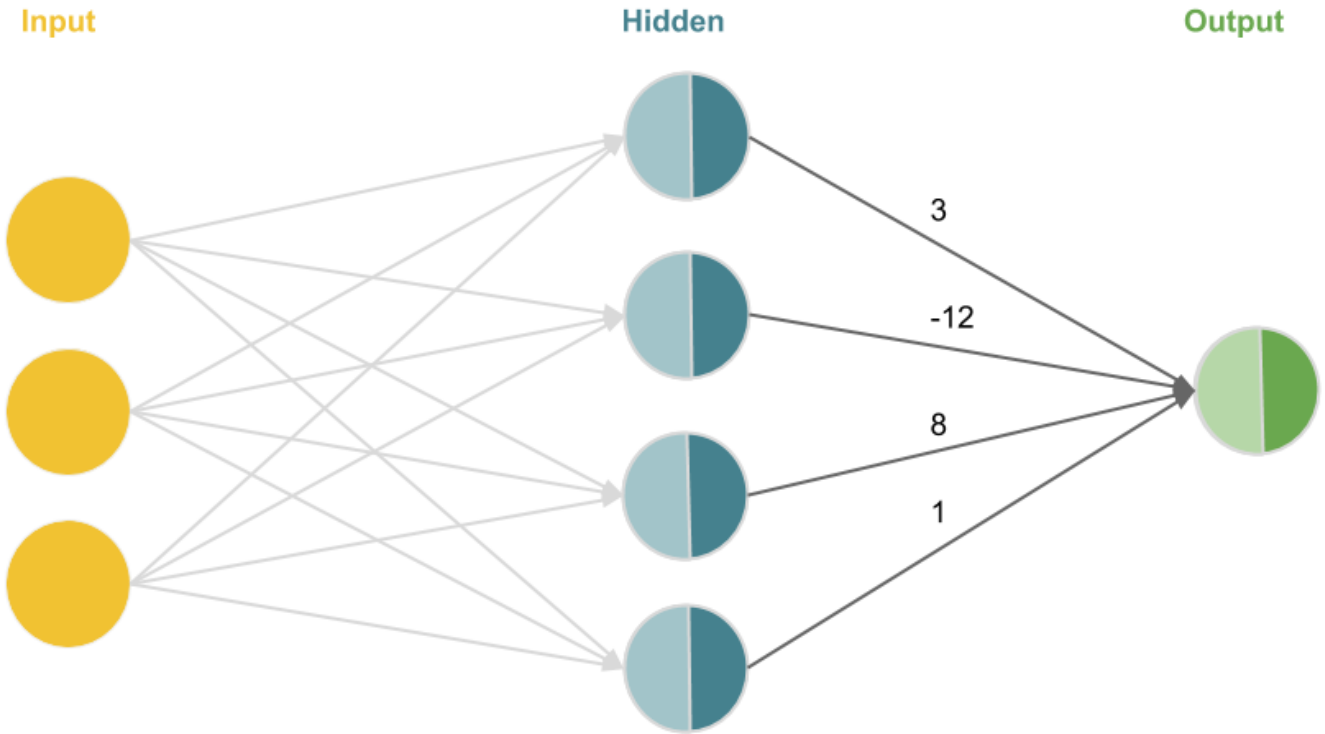
$W^T \quad \mathbf{b}$

$$\begin{pmatrix} \quad \end{pmatrix} \begin{pmatrix} \ \end{pmatrix}$$

$\mathbf{u}^T \quad c$

# SINGLE HIDDEN LAYER NETWORKS: EXAMPLE

- Weights (and biases) of the network.




$$\begin{pmatrix} 3 & -9 & 2 \\ 11 & -2 & 7 \\ -6 & 3 & -4 \\ 6 & -1 & 5 \end{pmatrix} \begin{pmatrix} 5 \\ 2 \\ -1 \\ 1 \end{pmatrix}$$

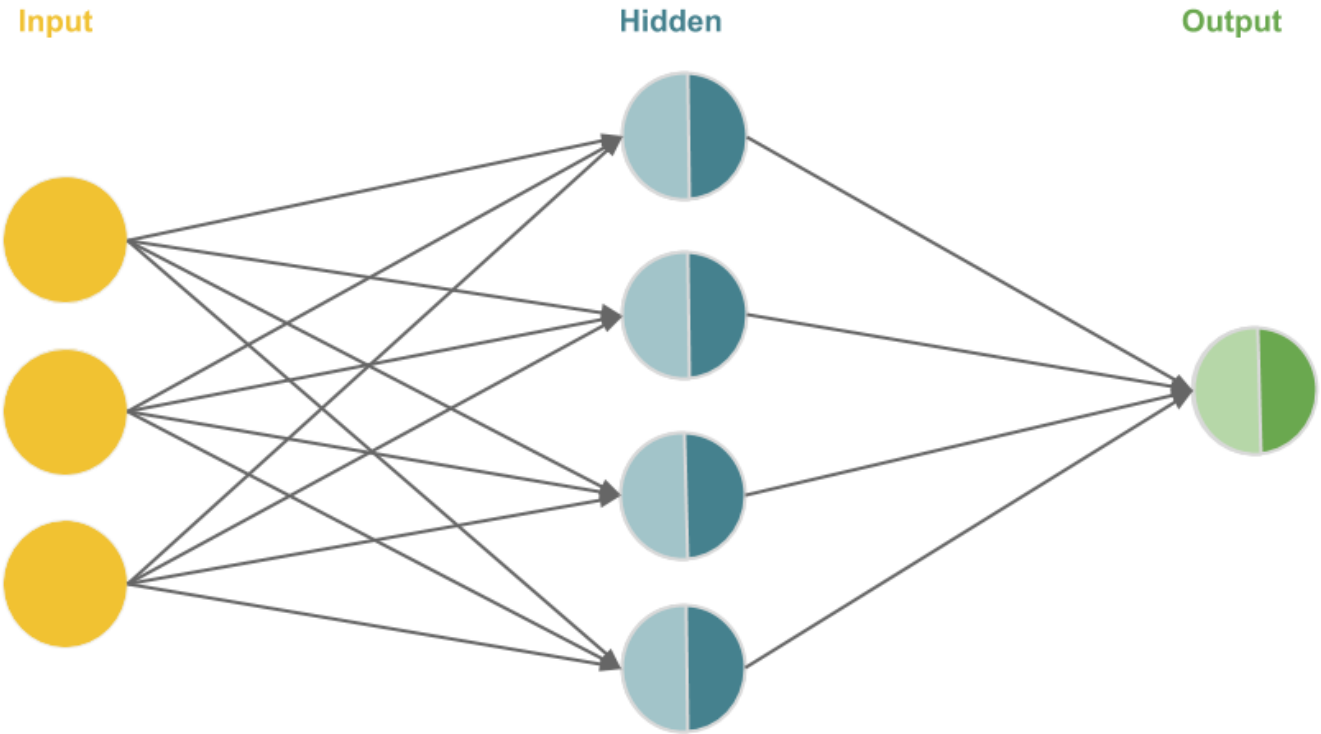
$W^T \quad \mathbf{b}$

$$\begin{pmatrix} 3 & -12 & 8 & 1 \end{pmatrix} \begin{pmatrix} 6 \end{pmatrix}$$

$\mathbf{u}^T \quad c$

# SINGLE HIDDEN LAYER NETWORKS: EXAMPLE

- Weights (and biases) of the network.




$$\begin{pmatrix} 3 & -9 & 2 \\ 11 & -2 & 7 \\ -6 & 3 & -4 \\ 6 & -1 & 5 \end{pmatrix} \begin{pmatrix} 5 \\ 2 \\ -1 \\ 1 \end{pmatrix}$$

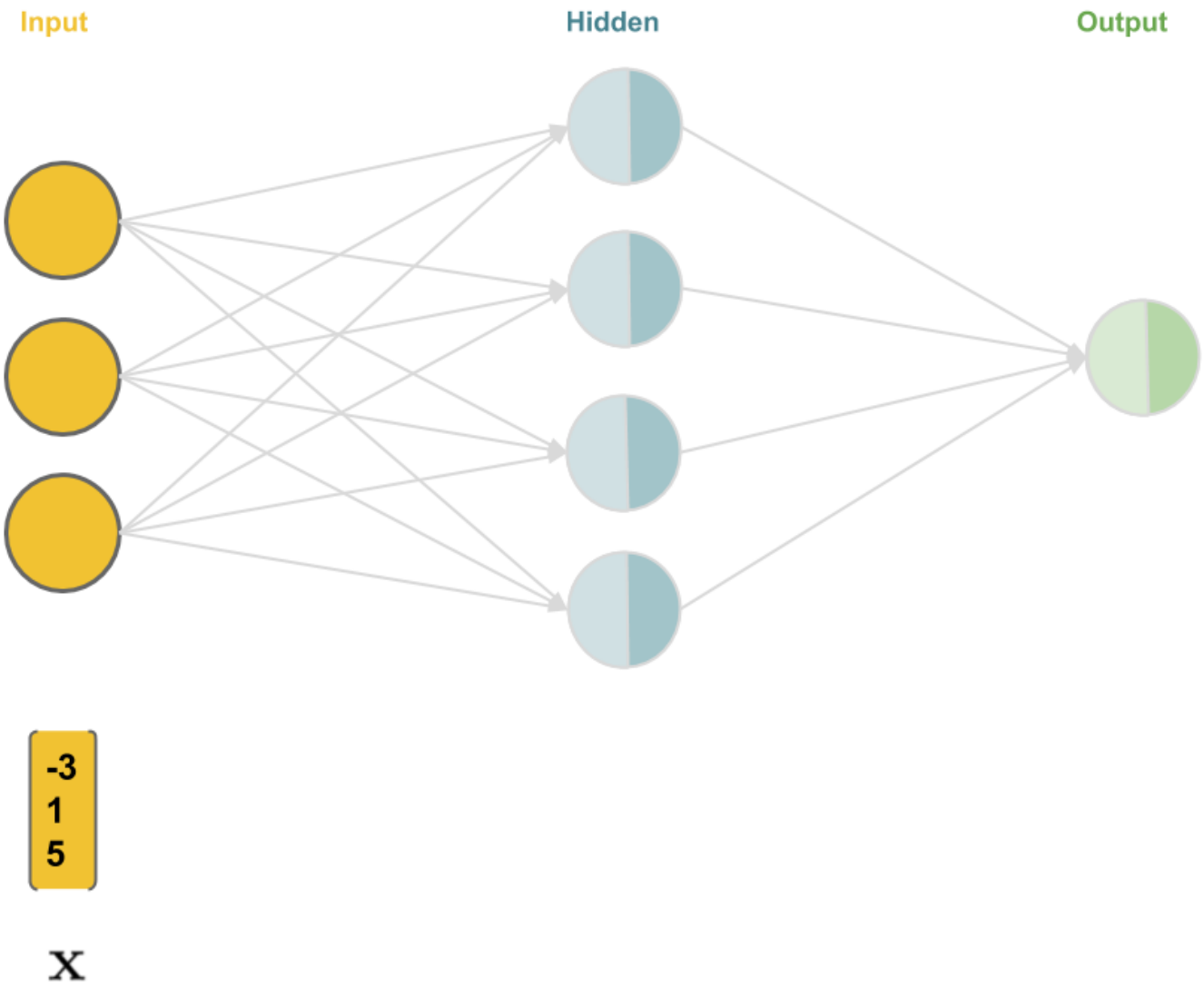
$W^T$ $\mathbf{b}$

$$\begin{pmatrix} 3 & -12 & 8 & 1 \end{pmatrix} \begin{pmatrix} 6 \end{pmatrix}$$

$\mathbf{u}^T$ $c$

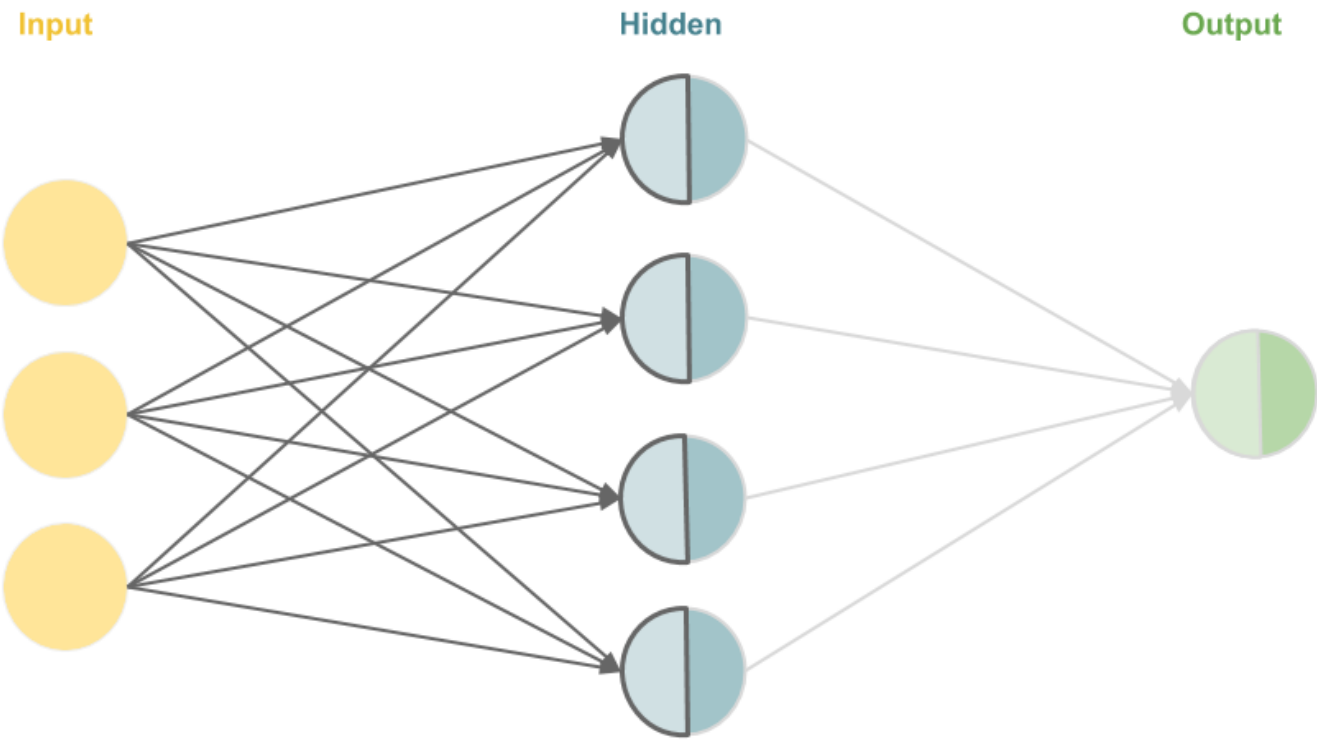
# SINGLE HIDDEN LAYER NETWORKS: EXAMPLE

Forward pass through the shallow neural network.



# SINGLE HIDDEN LAYER NETWORKS: EXAMPLE

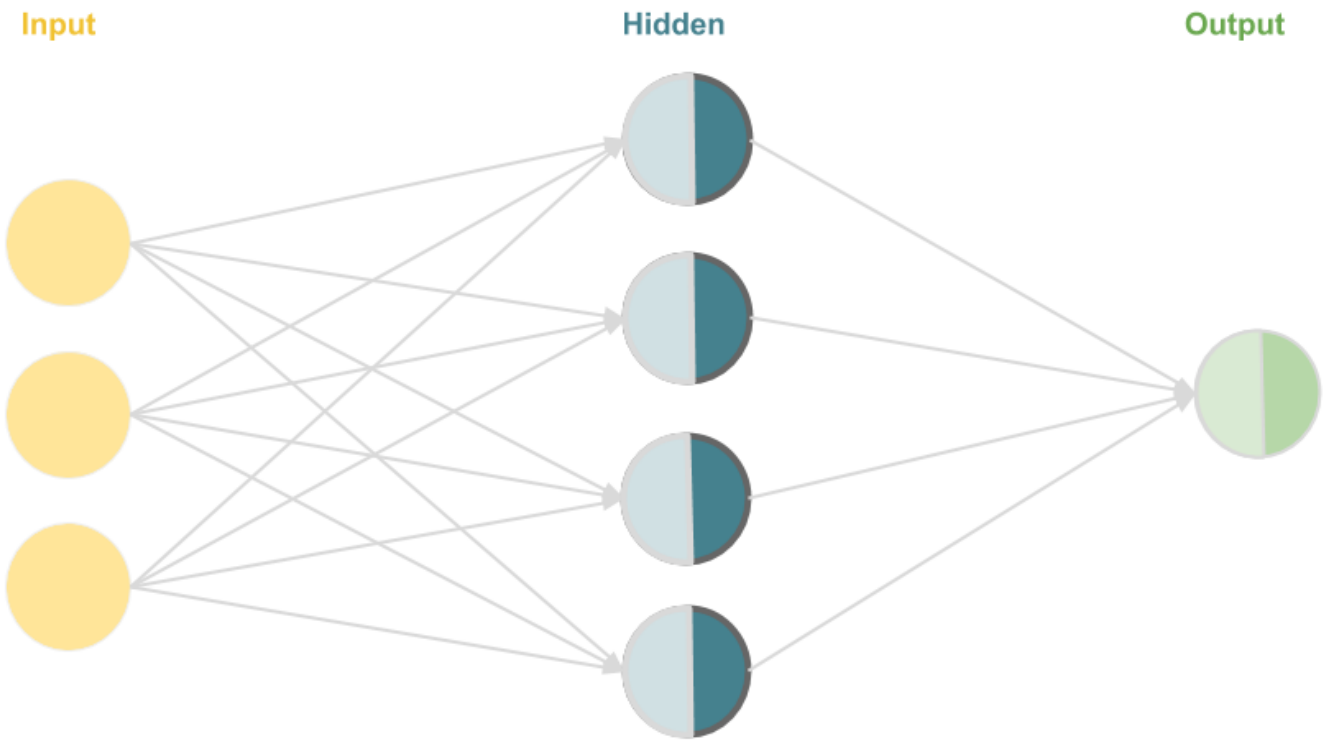
Forward pass through the shallow neural network.




$\mathbf{x}$ $\quad \mathbf{z}_{in} = W^\top \mathbf{x} + \mathbf{b}$

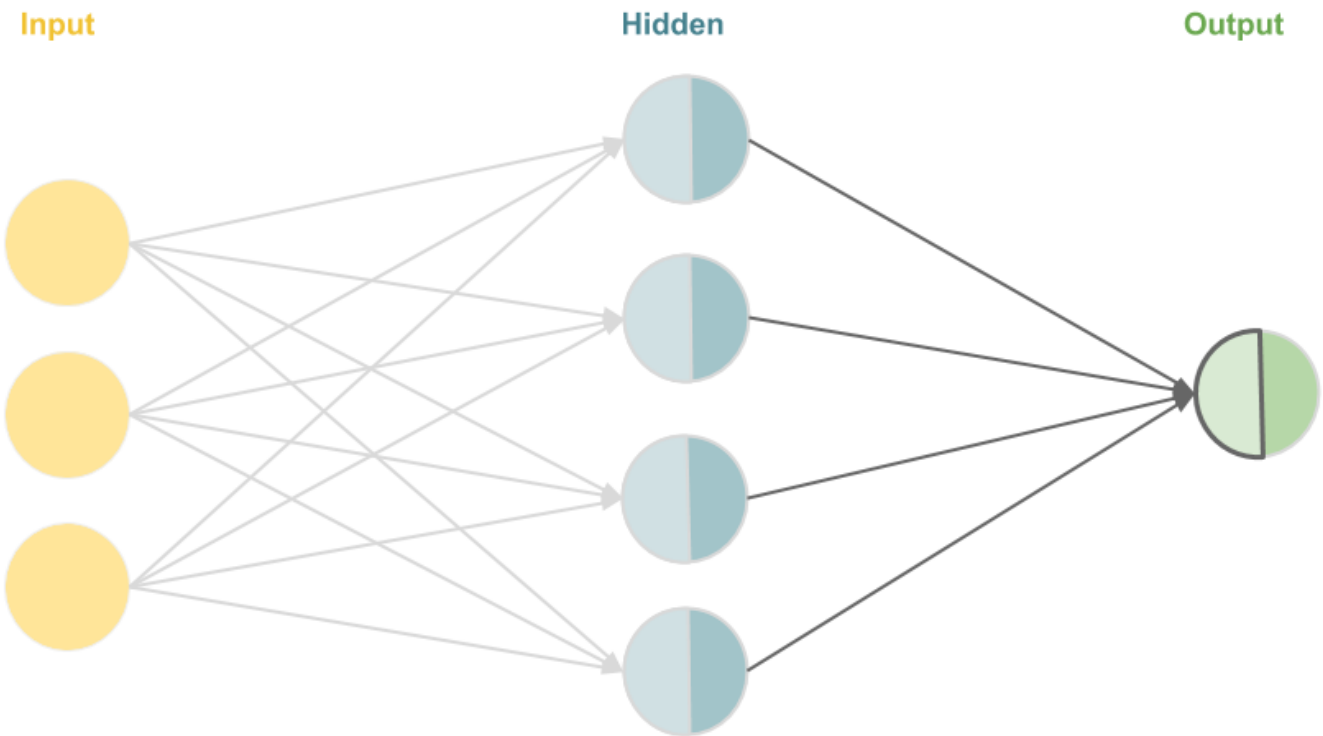
# SINGLE HIDDEN LAYER NETWORKS: EXAMPLE

Forward pass through the shallow neural network.



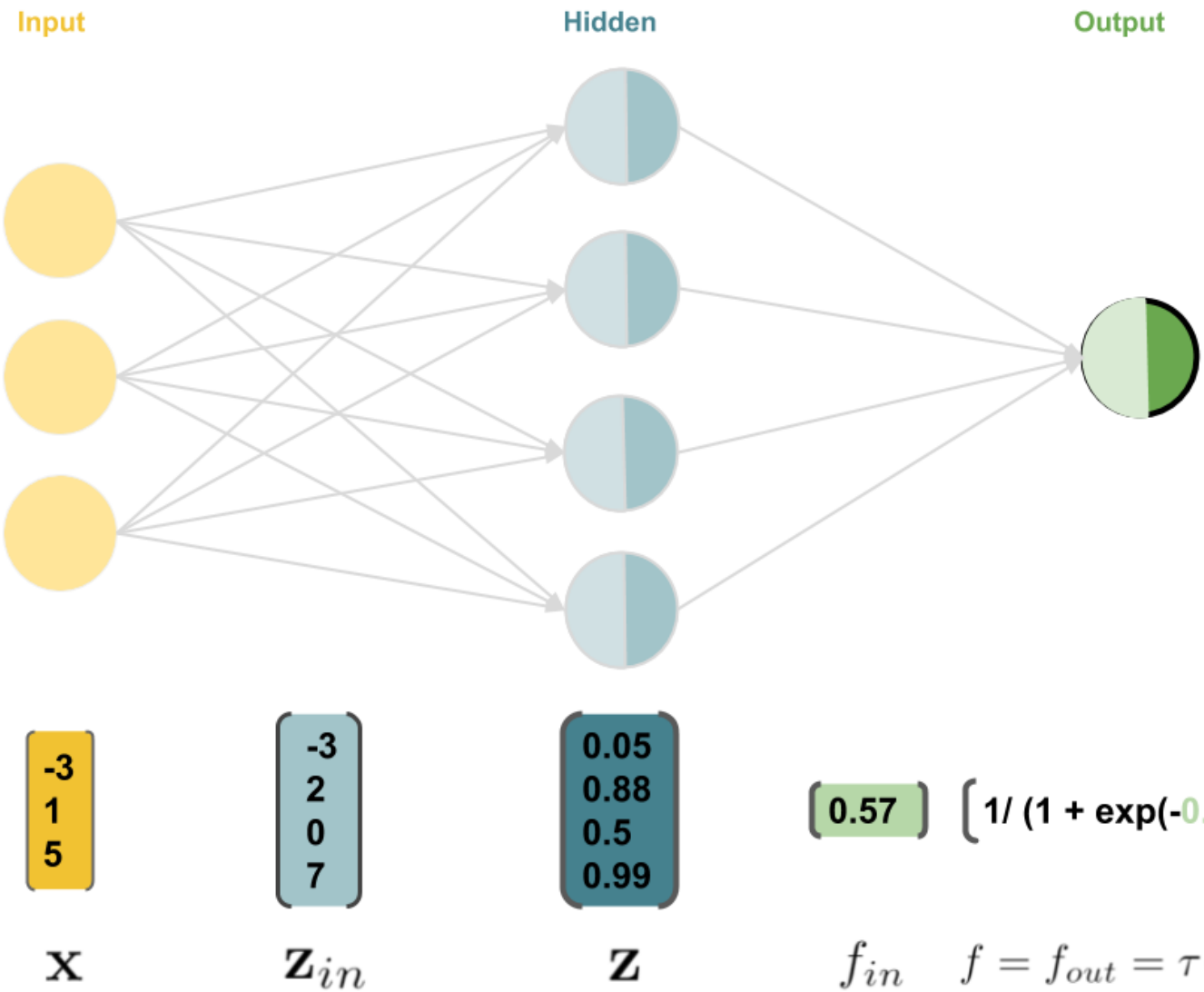
# SINGLE HIDDEN LAYER NETWORKS: EXAMPLE

Forward pass through the shallow neural network.



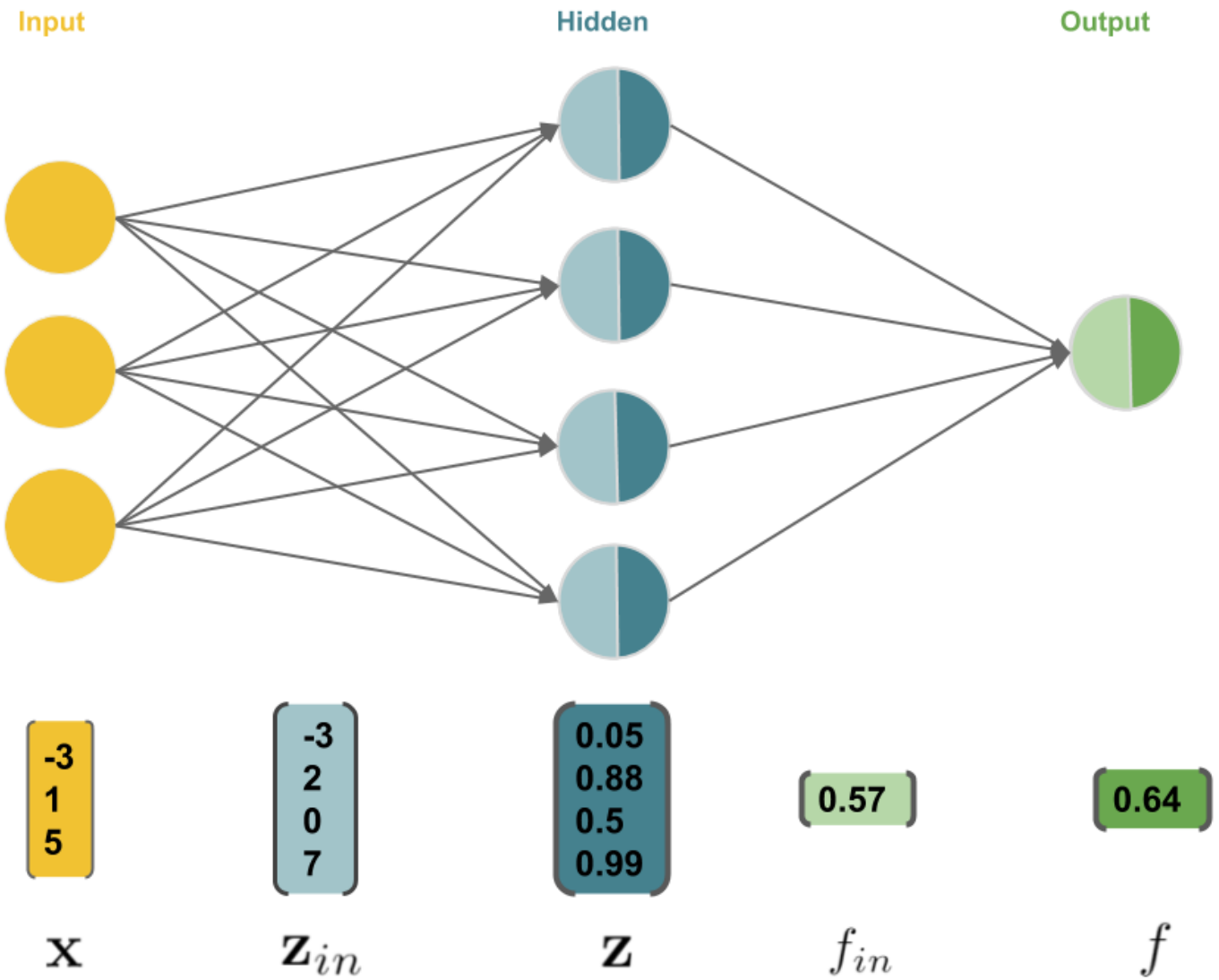
# SINGLE HIDDEN LAYER NETWORKS: EXAMPLE

Forward pass through the shallow neural network.



# SINGLE HIDDEN LAYER NETWORKS: EXAMPLE

Forward pass through the shallow neural network.



# HIDDEN LAYER: ACTIVATION FUNCTION

- It is important to note that if the hidden layer does not have a non-linear activation, the network can only learn linear decision boundaries.
- For simplification purposes, we drop the bias terms in notation and let $\sigma = \text{id}$. Then:

$$\begin{aligned} f(\mathbf{x}) &= \tau(\mathbf{u}^\top \mathbf{z}) = \tau(\mathbf{u}^\top \sigma(\mathbf{W}^\top \mathbf{x})) \\ &= \tau(\mathbf{u}^\top \sigma(\mathbf{W}^\top \mathbf{x})) \\ &= \tau(\mathbf{u}^\top \mathbf{W}^\top \mathbf{x}) = \tau(\mathbf{v}^\top \mathbf{x}) \end{aligned}$$

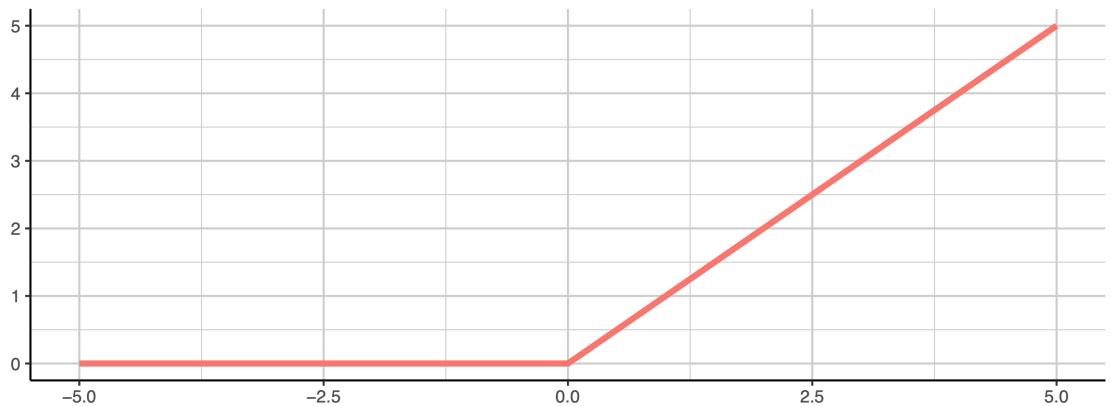
where $\mathbf{v} = \mathbf{W}\mathbf{u}$. It can be seen that $f(\mathbf{x})$ can only yield a linear decision boundary.

# HIDDEN LAYER: ACTIVATION FUNCTION

## ReLU activation:

- Currently the most popular choice is the ReLU (rectified linear unit):

$$\sigma(v) = \max(0, v)$$

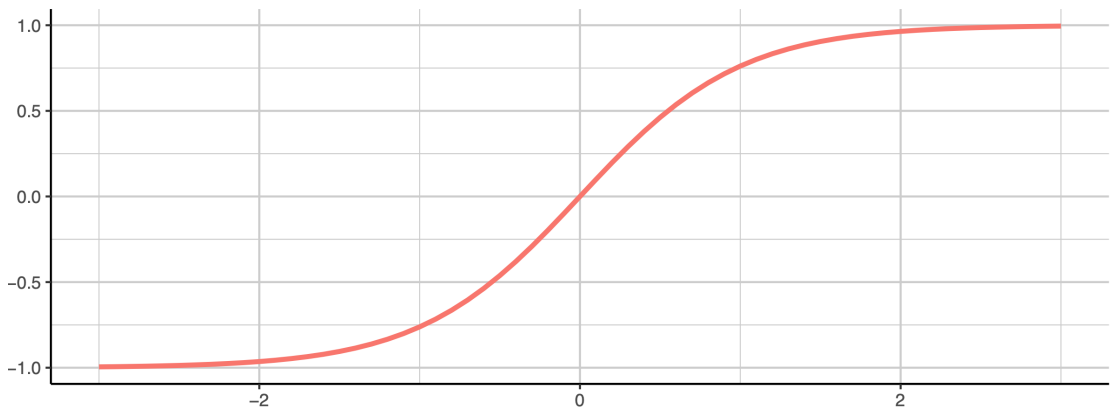


# HIDDEN LAYER: ACTIVATION FUNCTION

## Hyperbolic tangent activation:

- Another choice might be the hyperbolic tangent function:

$$\sigma(v) = \tanh(v) = \frac{\sinh(v)}{\cosh(v)} = 1 - \frac{2}{\exp(2v) + 1}$$

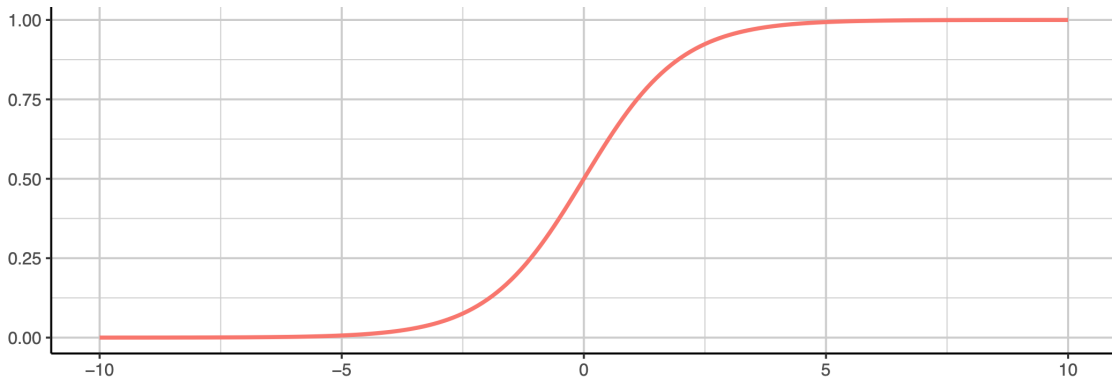


# HIDDEN LAYER: ACTIVATION FUNCTION

## Sigmoid activation function:

- Of course, as seen in the previous example, the sigmoid function can be used even in the hidden layer:

$$\sigma(v) = \frac{1}{1 + \exp(-v)}$$



# FEEDFORWARD NEURAL NETWORKS

- We will now extend the model class once again, such that we allow an arbitrary amount of $l$ (hidden) layers.

- The general term for this model class is (multi-layer) **feedforward networks** (inputs are passed through the network from left to right, no feedback-loops are allowed)

# FEEDFORWARD NEURAL NETWORKS

- We can characterize those models by the following chain structure:

$$f(\mathbf{x}) = \tau \circ \phi \circ \sigma^{(l)} \circ \phi^{(l)} \circ \sigma^{(l-1)} \circ \phi^{(l-1)} \circ \ldots \circ \sigma^{(1)} \circ \phi^{(1)}$$

  where $\sigma^{(i)}$ and $\phi^{(i)}$ are the activation function and the weighted sum of hidden layer $i$, respectively. $\tau$ and $\phi$ are the corresponding components of the output layer.

- Each hidden layer has:
  - an associated weight matrix $\mathbf{W}^{(i)}$, bias $\mathbf{b}^{(i)}$ and activations $\mathbf{z}^{(i)}$ for $i \in \{1 \ldots l\}$
  - $\mathbf{z}^{(i)} = \sigma^{(i)}(\phi^{(i)}) = \sigma^{(i)}(\mathbf{W}^{(i)T}\mathbf{z}^{(i-1)} + \mathbf{b}^{(i)})$ , where $\mathbf{z}^{(0)} = \mathbf{x}$.

- Again, without non-linear activations in the hidden layers, the network can only learn linear decision boundaries.

# FEEDFORWARD NEURAL NETWORKS

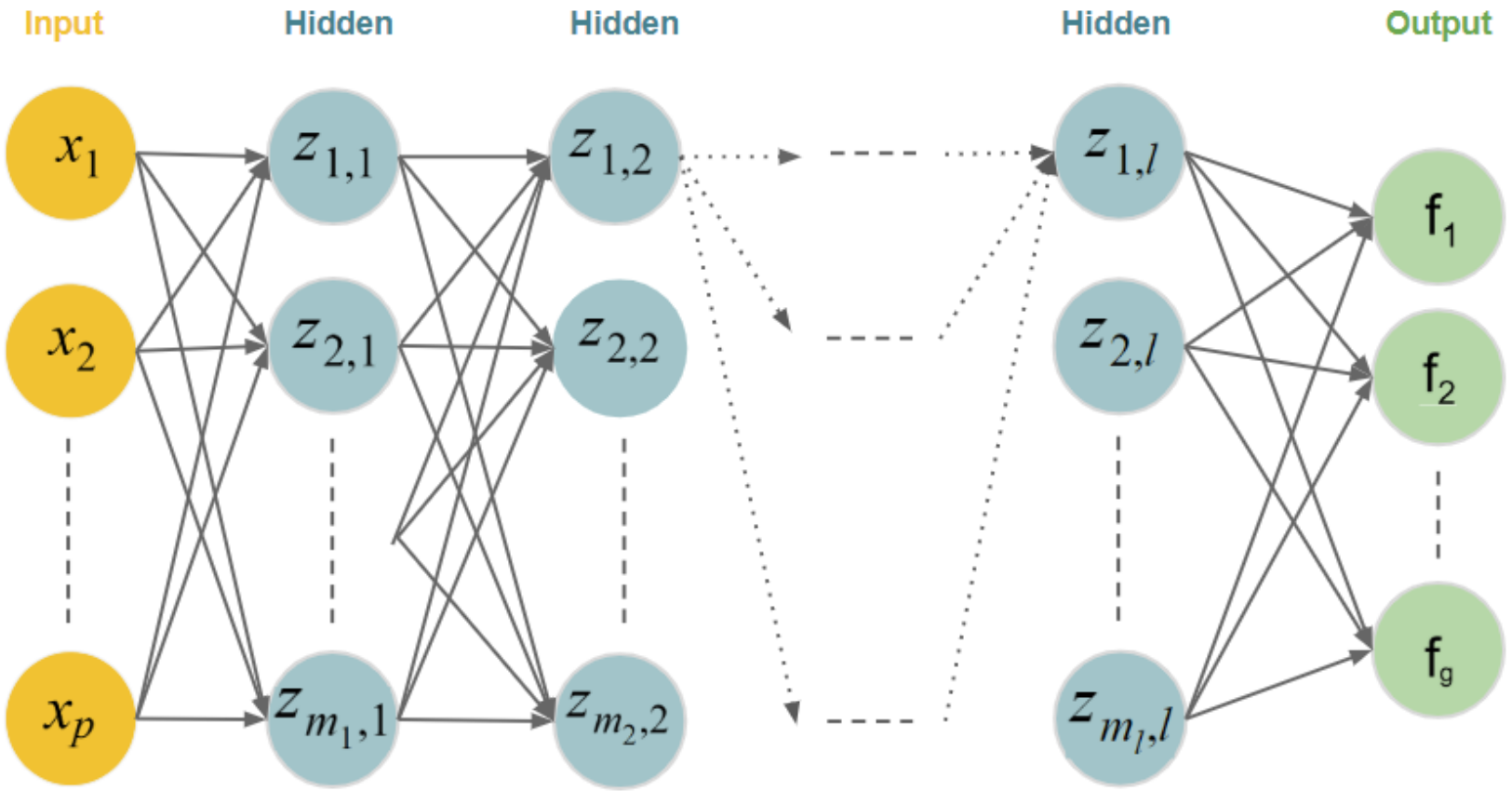



**Figure:** Structure of a deep neural network with $l$ hidden layers (bias terms omitted).

# WHY ADD MORE LAYERS?

- Multiple layers allow for the extraction of more and more abstract representations.
- Each layer in a feed-forward neural network adds its own degree of non-linearity to the model.

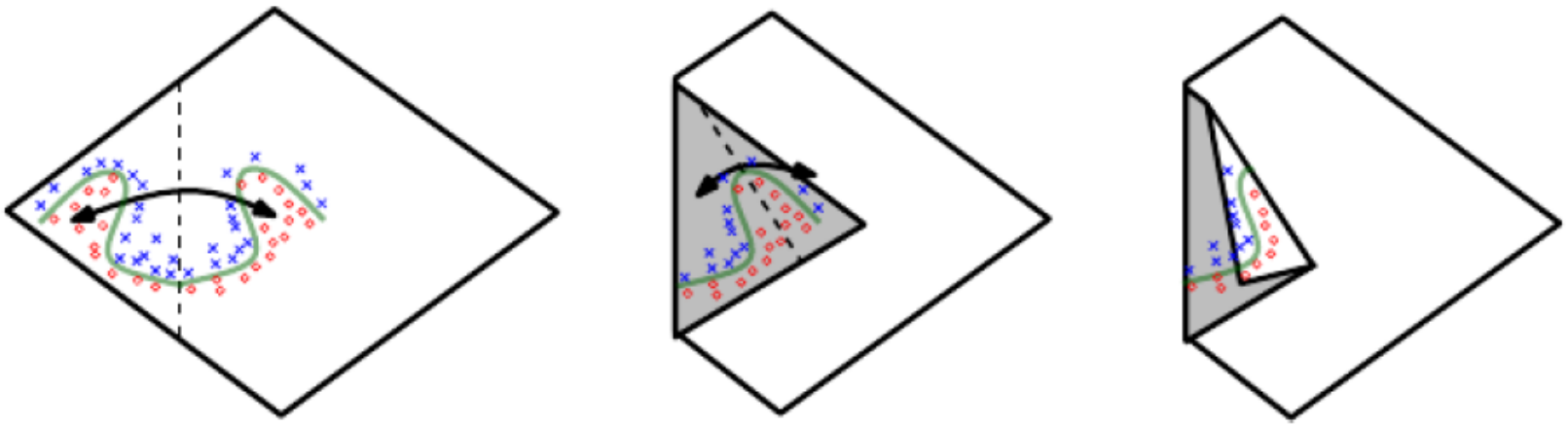


**Figure:** An intuitive, geometric explanation of the exponential advantage of deeper networks formally (Montúfar et al. (2014)).

# DEEP NEURAL NETWORKS

- Neural networks today can have dozens or even hundreds of hidden layers. The greater the number of layers, the "deeper" the network.
- Historically, deep neural networks were very challenging to train for several reasons.
- For one thing, the use of sigmoid activations (such as logistic sigmoid and tanh) significantly slowed down training due to a phenomenon known as “vanishing gradients”. The introduction of the ReLU activation largely solved this problem.
- Additionally, training deep neural networks on CPUs was too slow to be practical. Switching over to GPUs (Graphics Processing Units) cut down training time by more than an order of magnitude.
- Another reason neural networks were not popular until the late ’00s is that when dataset sizes are small, other models (such as SVMs) and techniques (such as feature engineering) outperform them.

# DEEP NEURAL NETWORKS

- Therefore, the availability of large datasets (such as ImageNet) and novel architectures that are capable to handle even complex tensor-shaped data (e.g. CNNs for image data), significantly faster hardware, and equally better optimization and regularization methods made it feasible to successfully implement deep neural networks in the last decade.
- An increase in depth often translates to an increase in performance on a given task.
- State-of-the-art neural networks, however, are much more sophisticated than the simple architectures we have encountered so far. (Stay tuned!)
- The term "**deep learning**" encompasses all of these developments and refers to the field as a whole.