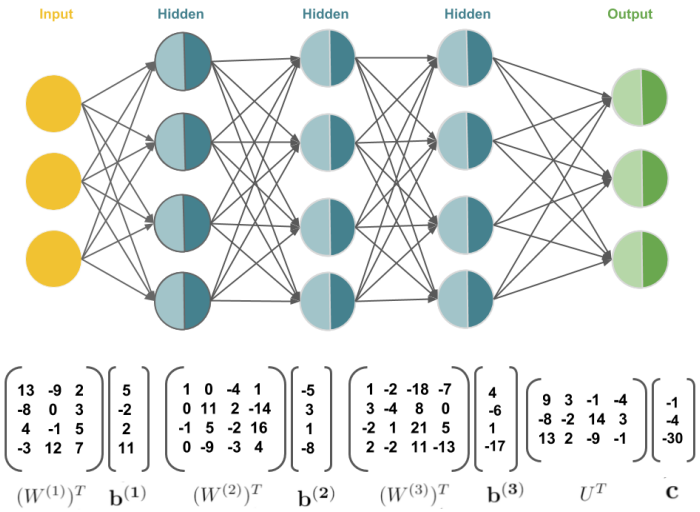
Deep Learning

MLP – Multi-Layer Feedforward Neural Networks



Learning goals

- Architectures of deep neural networks
- Deep neural networks as chained functions

FEEDFORWARD NEURAL NETWORKS

- We will now extend the model class once again, such that we allow an arbitrary amount l of 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. τ and ϕ 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

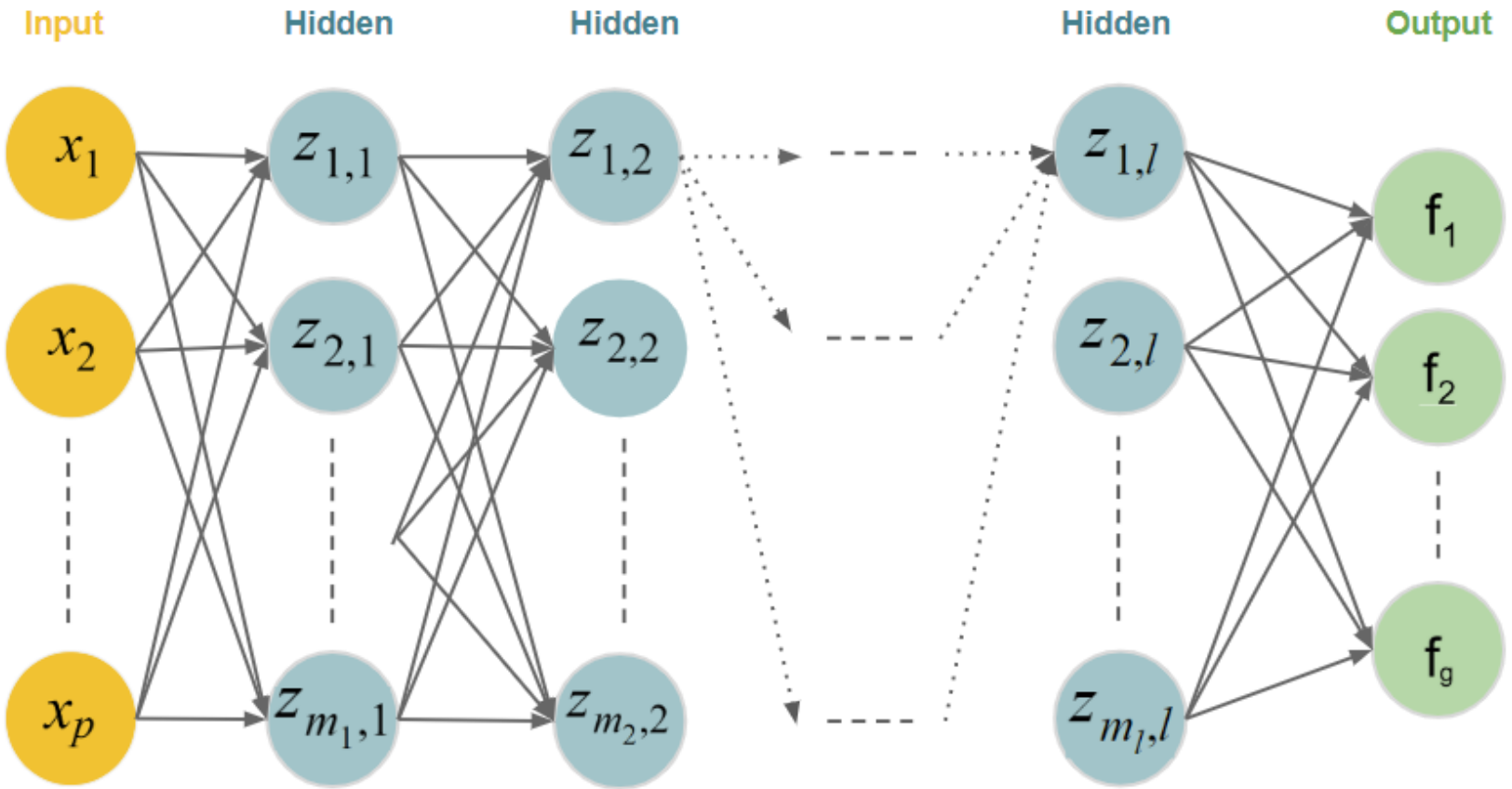
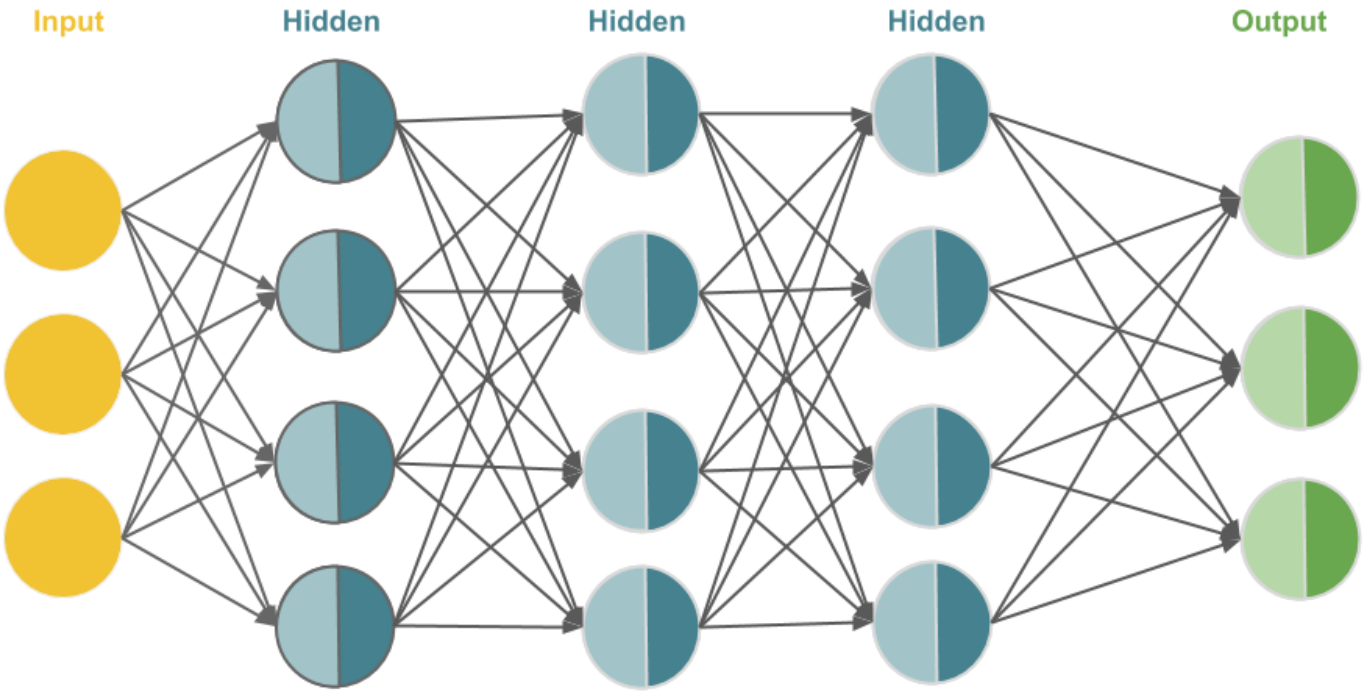


Figure: Structure of a deep neural network with *l* hidden layers (bias terms omitted).

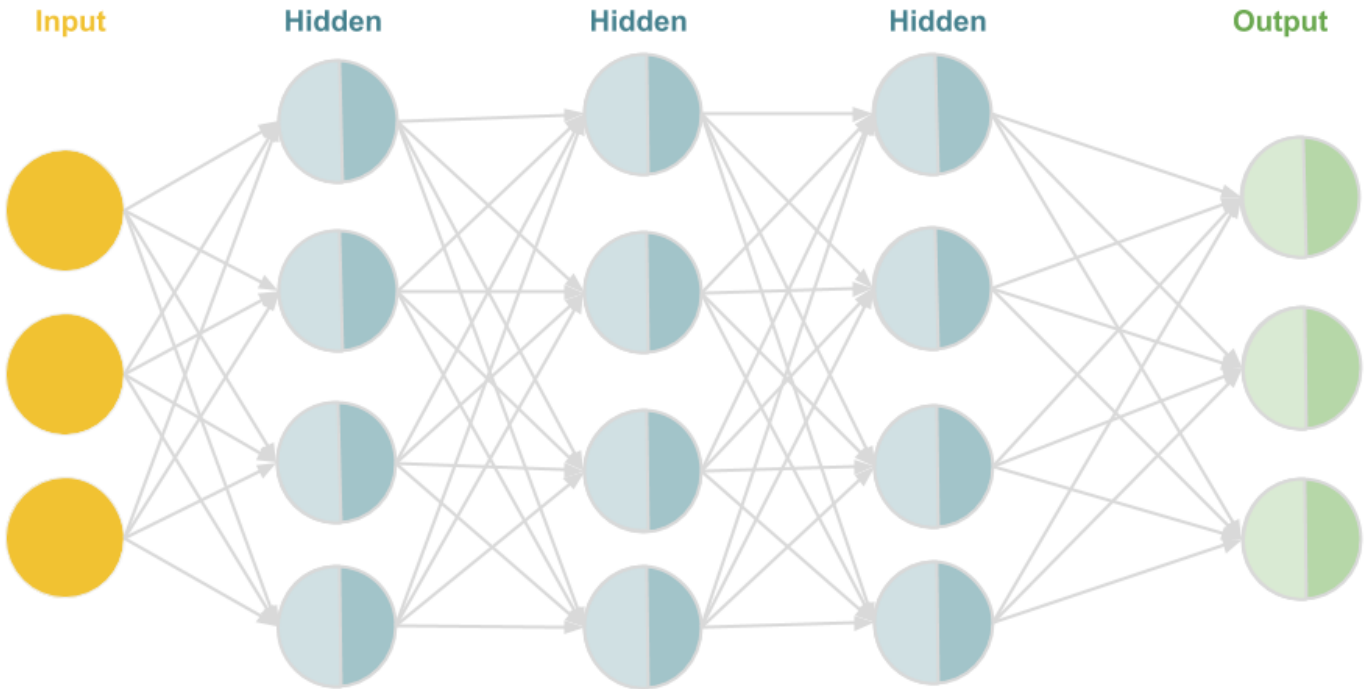
FEEDFORWARD NEURAL NETWORKS: EXAMPLE



$$\begin{bmatrix} 13 & -9 & 2 \\ -8 & 0 & 3 \\ 4 & -1 & 5 \\ -3 & 12 & 7 \end{bmatrix} \begin{bmatrix} 5 \\ -2 \\ 2 \\ 11 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & -4 & 1 \\ 0 & 11 & 2 & -14 \\ -1 & 5 & -2 & 16 \\ 0 & -9 & -3 & 4 \end{bmatrix} \begin{bmatrix} -5 \\ 3 \\ 1 \\ -8 \end{bmatrix} \quad \begin{bmatrix} 1 & -2 & -18 & -7 \\ 3 & -4 & 8 & 0 \\ -2 & 1 & 21 & 5 \\ 2 & -2 & 11 & -13 \end{bmatrix} \begin{bmatrix} 4 \\ -6 \\ 1 \\ -17 \end{bmatrix} \quad \begin{bmatrix} 9 & 3 & -1 & -4 \\ -8 & -2 & 14 & 3 \\ 13 & 2 & -9 & -1 \end{bmatrix} \begin{bmatrix} -1 \\ -4 \\ -30 \end{bmatrix}$$

$$(W^{(1)})^T \quad \mathbf{b}^{(1)} \quad (W^{(2)})^T \quad \mathbf{b}^{(2)} \quad (W^{(3)})^T \quad \mathbf{b}^{(3)} \quad U^T \quad \mathbf{c}$$

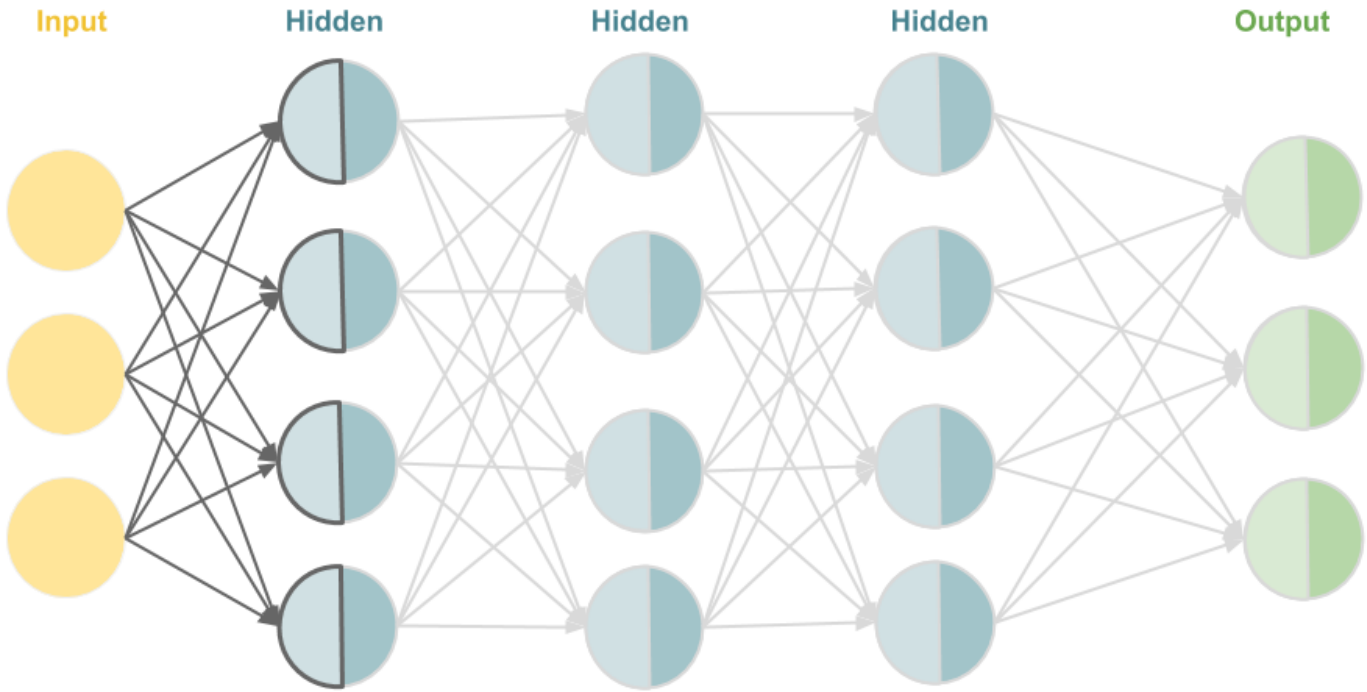
FEEDFORWARD NEURAL NETWORKS: EXAMPLE



7
1
-4

$\mathbf{x}$

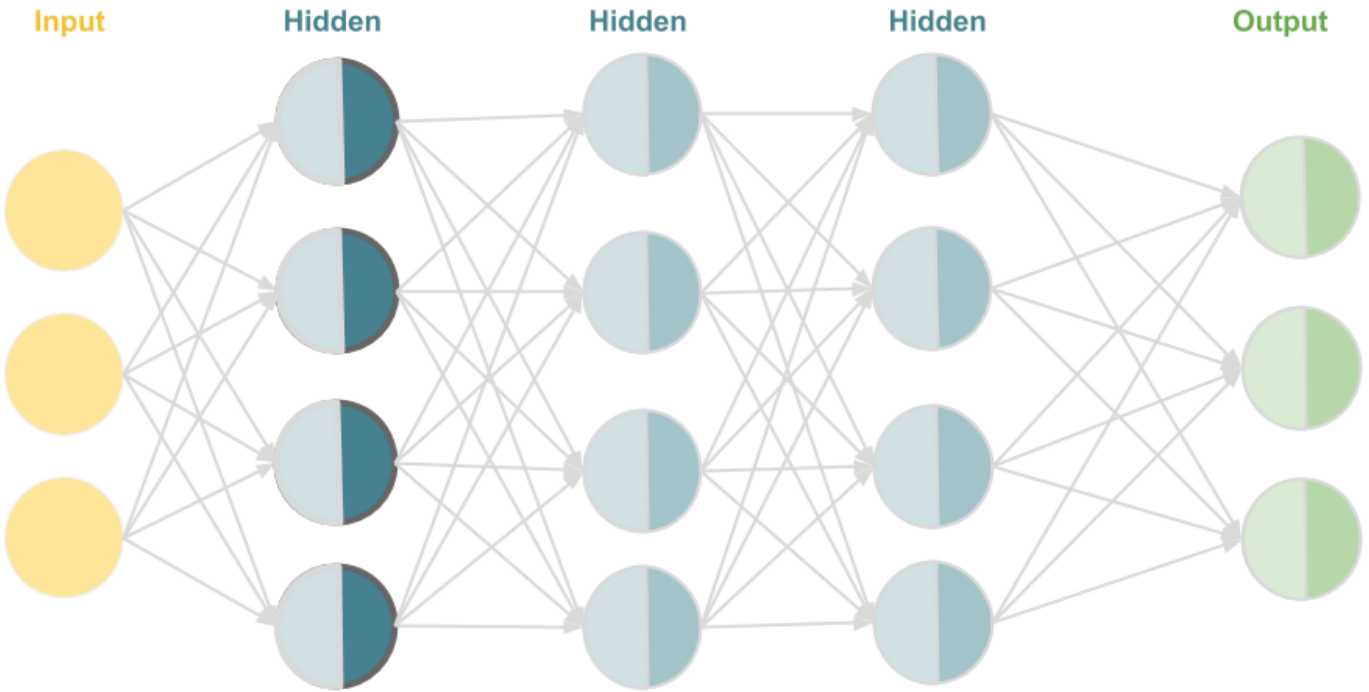
FEEDFORWARD NEURAL NETWORKS: EXAMPLE



$$\begin{matrix}7\\1\\-4\end{matrix}\begin{pmatrix}7*13 + 1*(-9) + (-4)*2 + 5\\7*(-8) + 1*0 + (-4)*3 + (-2)\\7*4 + 1*(-1) + (-4)*5 + 2\\7*(-3) + 1*12 + (-4)*7 + 11\end{pmatrix}$$

$\mathbf{x}$ $\quad \mathbf{z}_{in}^{(1)} = W^{(1)T}\mathbf{x} + \mathbf{b}^{(1)}$

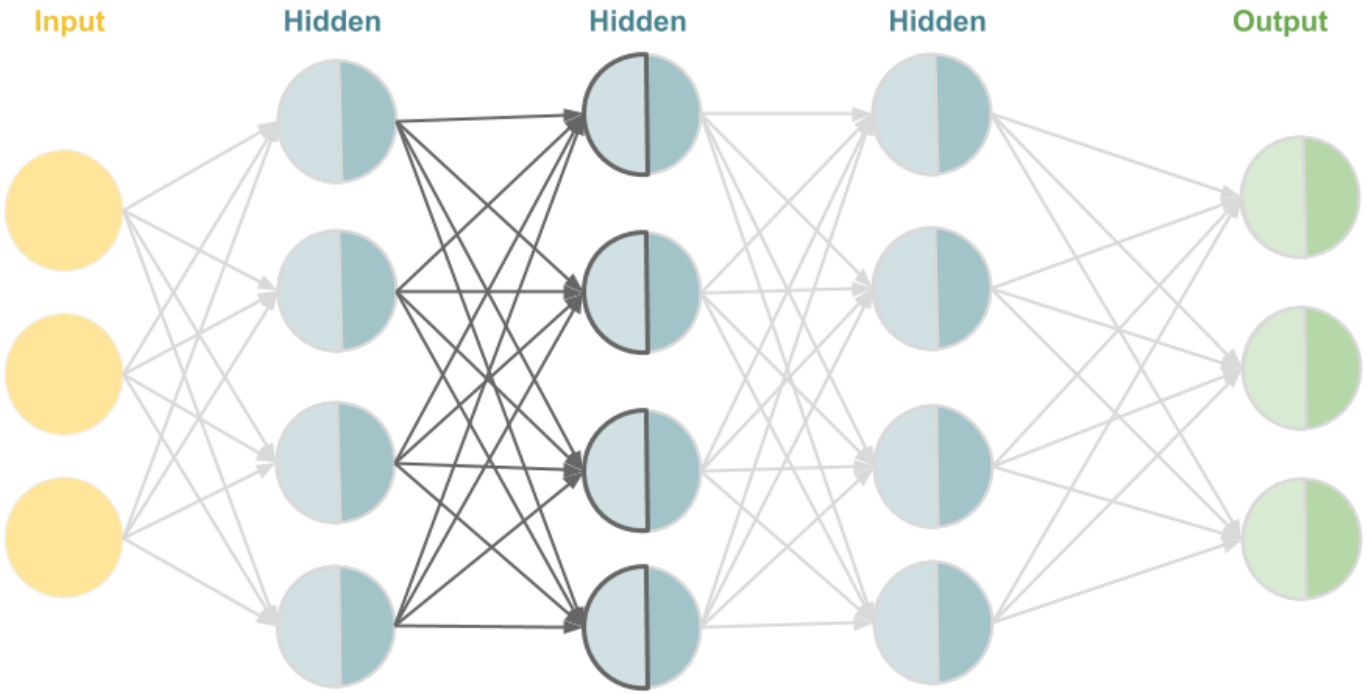
FEEDFORWARD NEURAL NETWORKS: EXAMPLE



$$\begin{bmatrix} 7 \\ 1 \\ -4 \end{bmatrix} \quad \begin{bmatrix} 79 \\ -70 \\ 9 \\ -26 \end{bmatrix} \quad \begin{bmatrix} \max(0, 79) \\ \max(0, -70) \\ \max(0, 9) \\ \max(0, -26) \end{bmatrix}$$

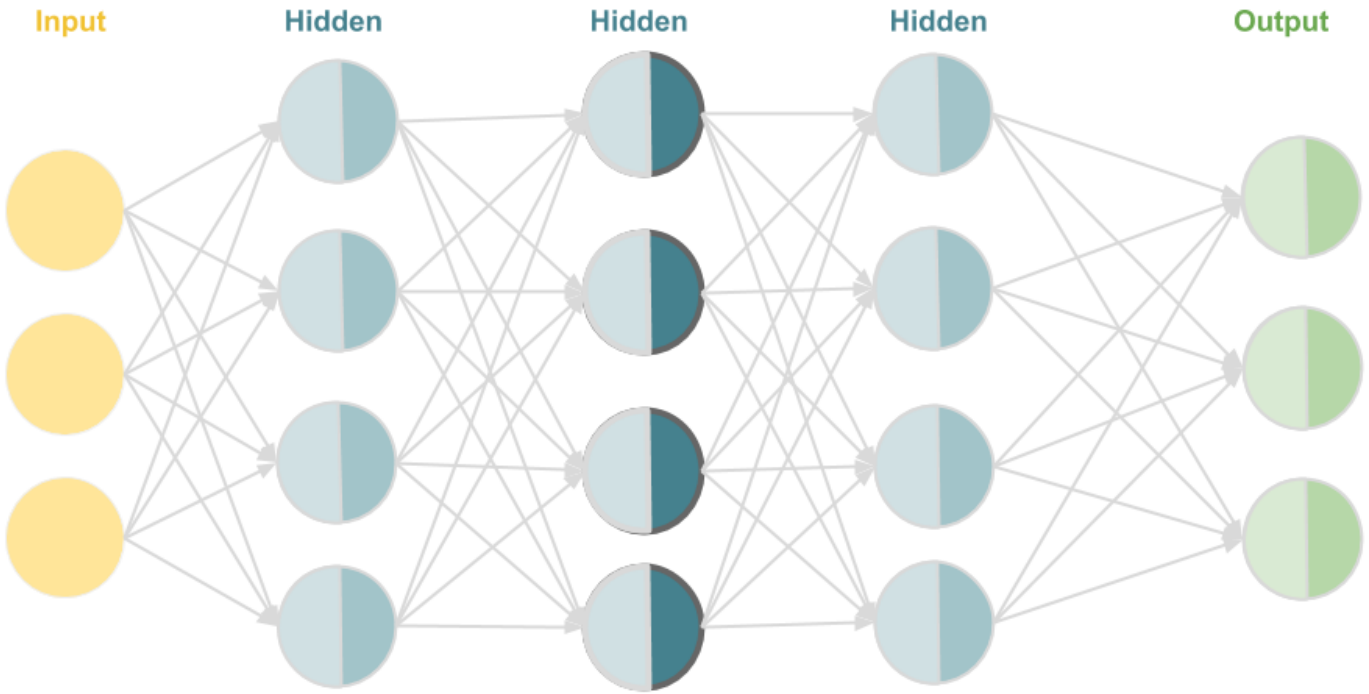
$\mathbf{x}$ $\quad \mathbf{z}_{in}^{(1)}$ $\quad \mathbf{z}^{(1)} = \mathbf{z}_{out}^{(1)} = \sigma(\mathbf{z}_{in}^{(1)})$

FEEDFORWARD NEURAL NETWORKS: EXAMPLE

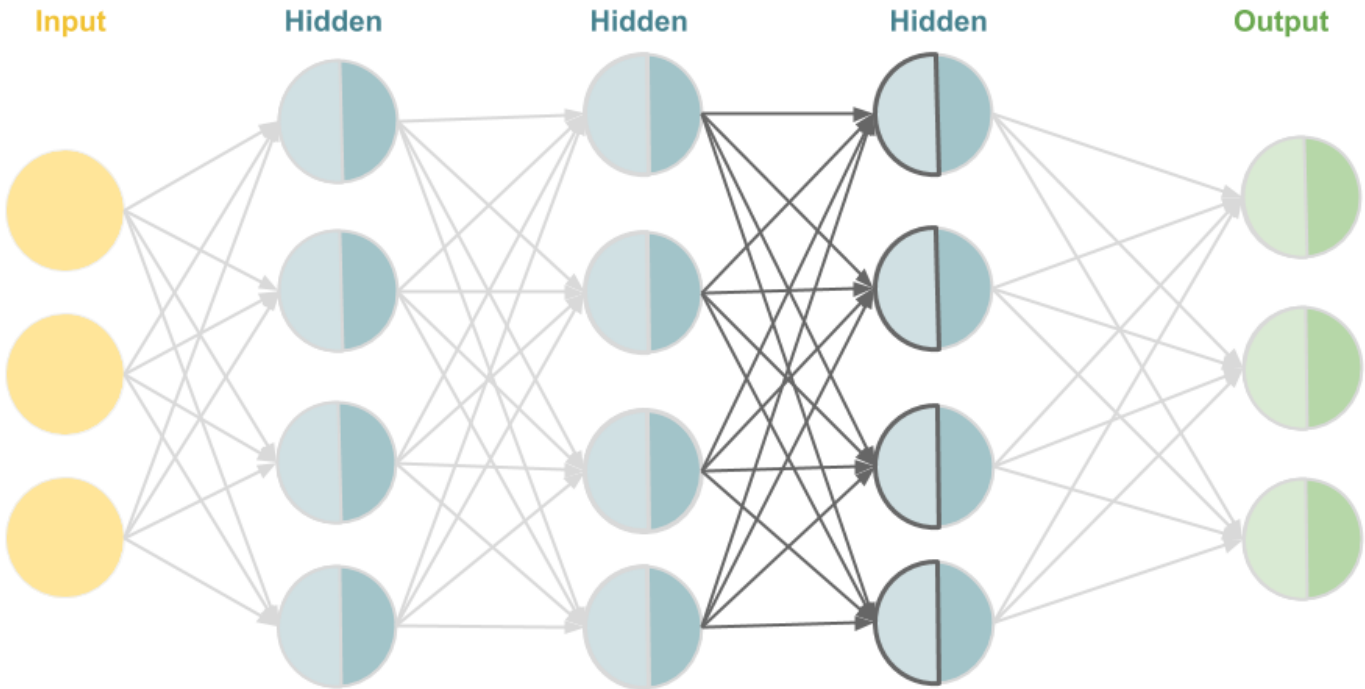


$\mathbf{x}$	$\mathbf{z}_{in}^{(1)}$	$\mathbf{z}^{(1)}$	$\mathbf{z}_{in}^{(2)} = W^{(2)T}\mathbf{z}^{(1)} + \mathbf{b}^{(2)}$
7 1 -4	79 -70 9 -26	79 0 9 0	79*1 + 0*0 + 9*(-4) + 0*1 + (-5) 79*0 + 0*11 + 9*2 + 0*(-14) + 3 79*(-1) + 0*5 + 9*(-2) + 0*16 + 1 79*0 + 0*(-9) + 9*(-3) + 0*4 + (-8)

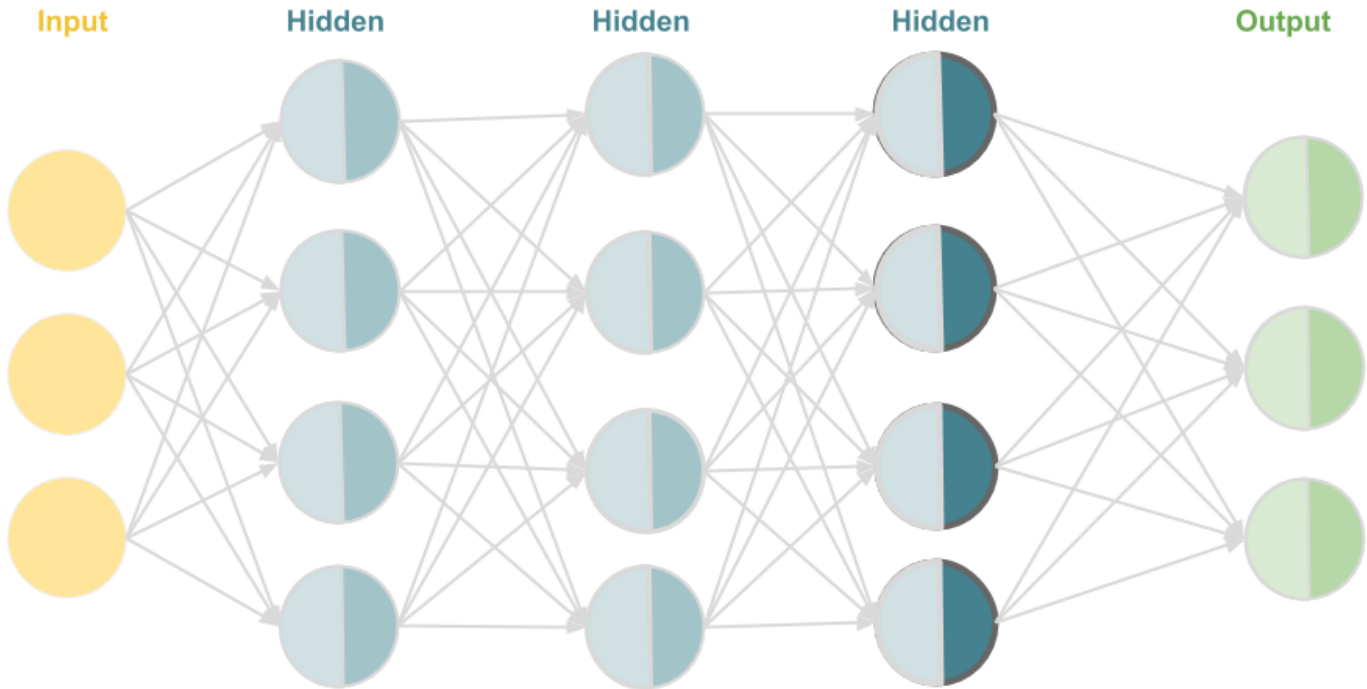
FEEDFORWARD NEURAL NETWORKS: EXAMPLE



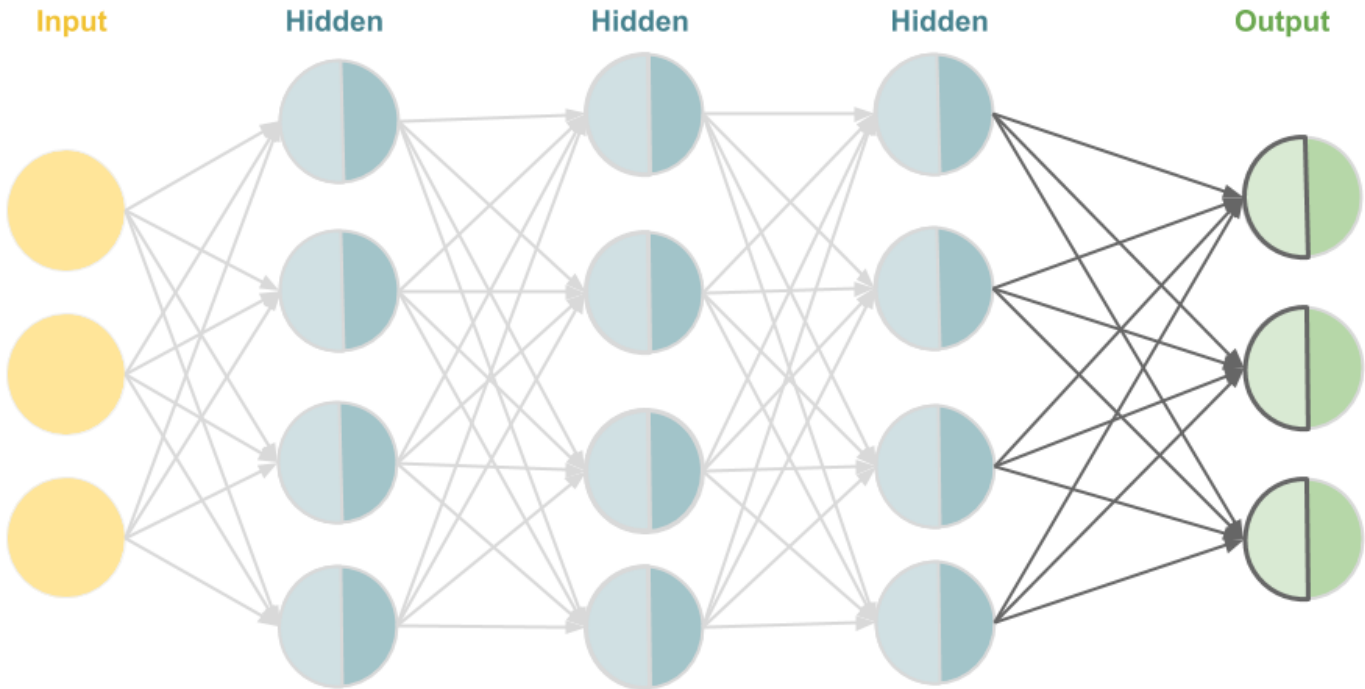
FEEDFORWARD NEURAL NETWORKS: EXAMPLE



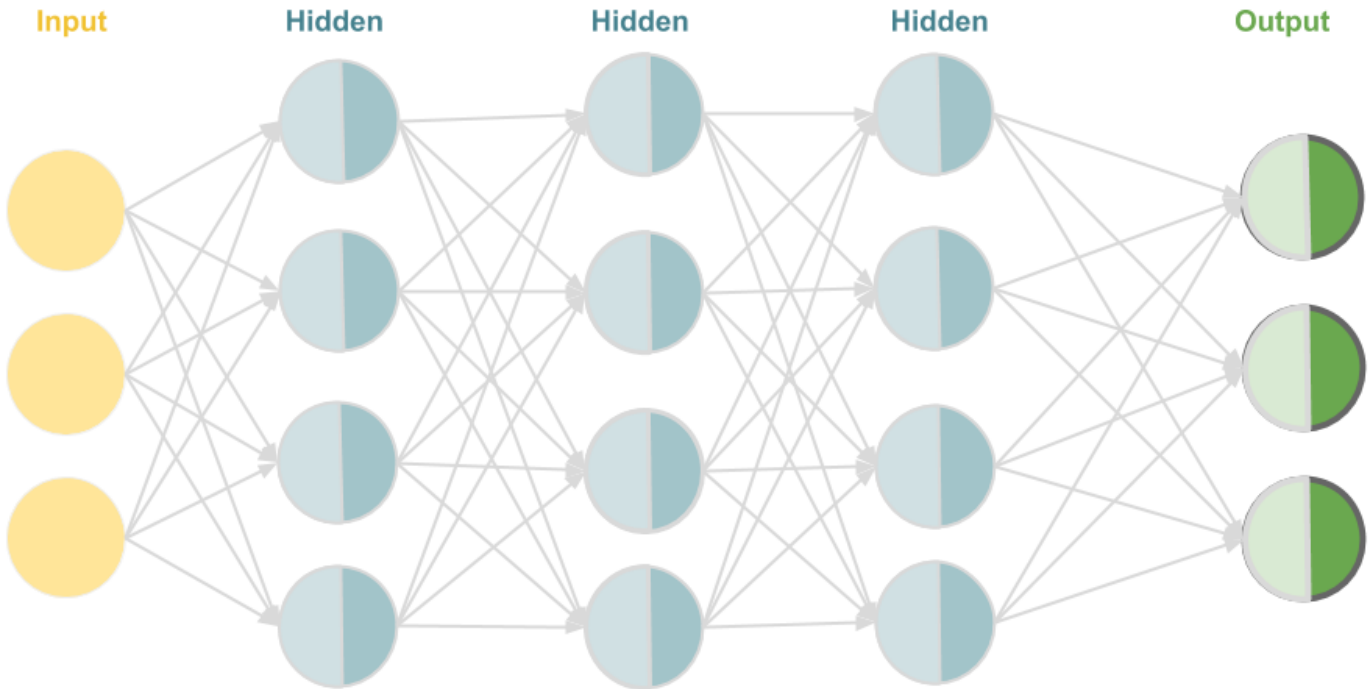
FEEDFORWARD NEURAL NETWORKS: EXAMPLE



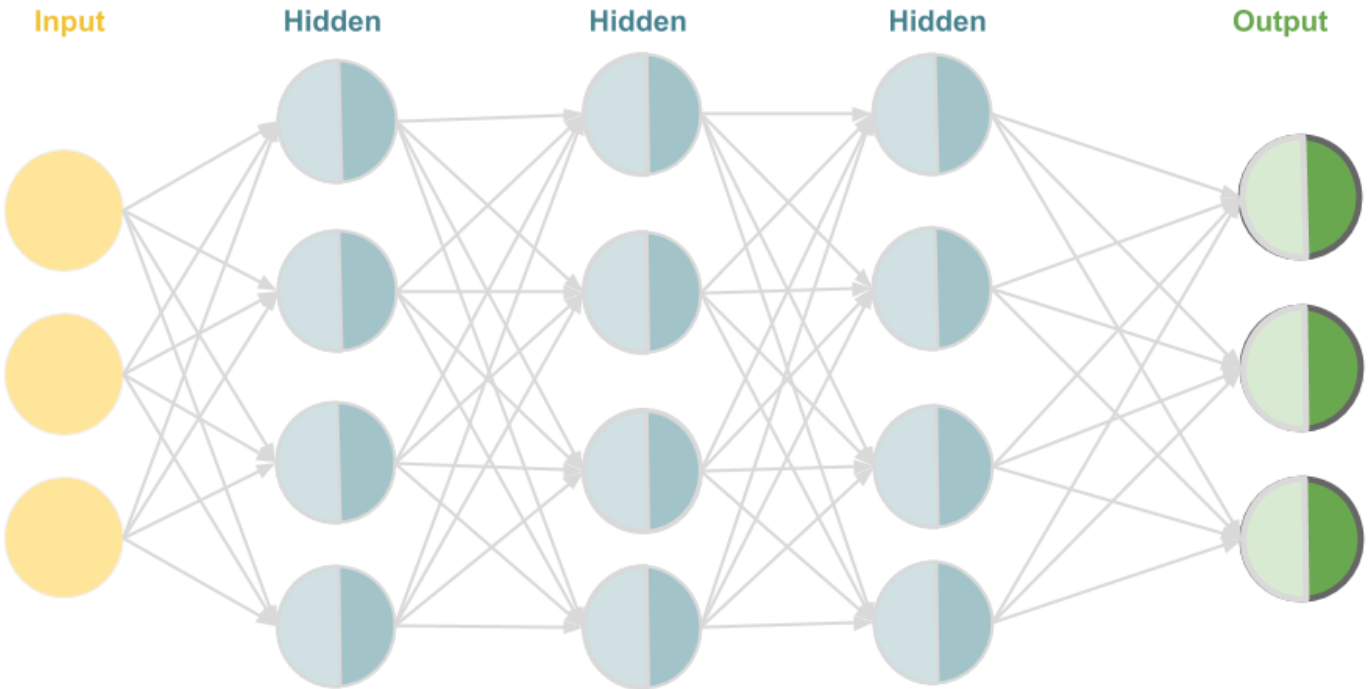
FEEDFORWARD NEURAL NETWORKS: EXAMPLE



FEEDFORWARD NEURAL NETWORKS: EXAMPLE



FEEDFORWARD NEURAL NETWORKS: EXAMPLE



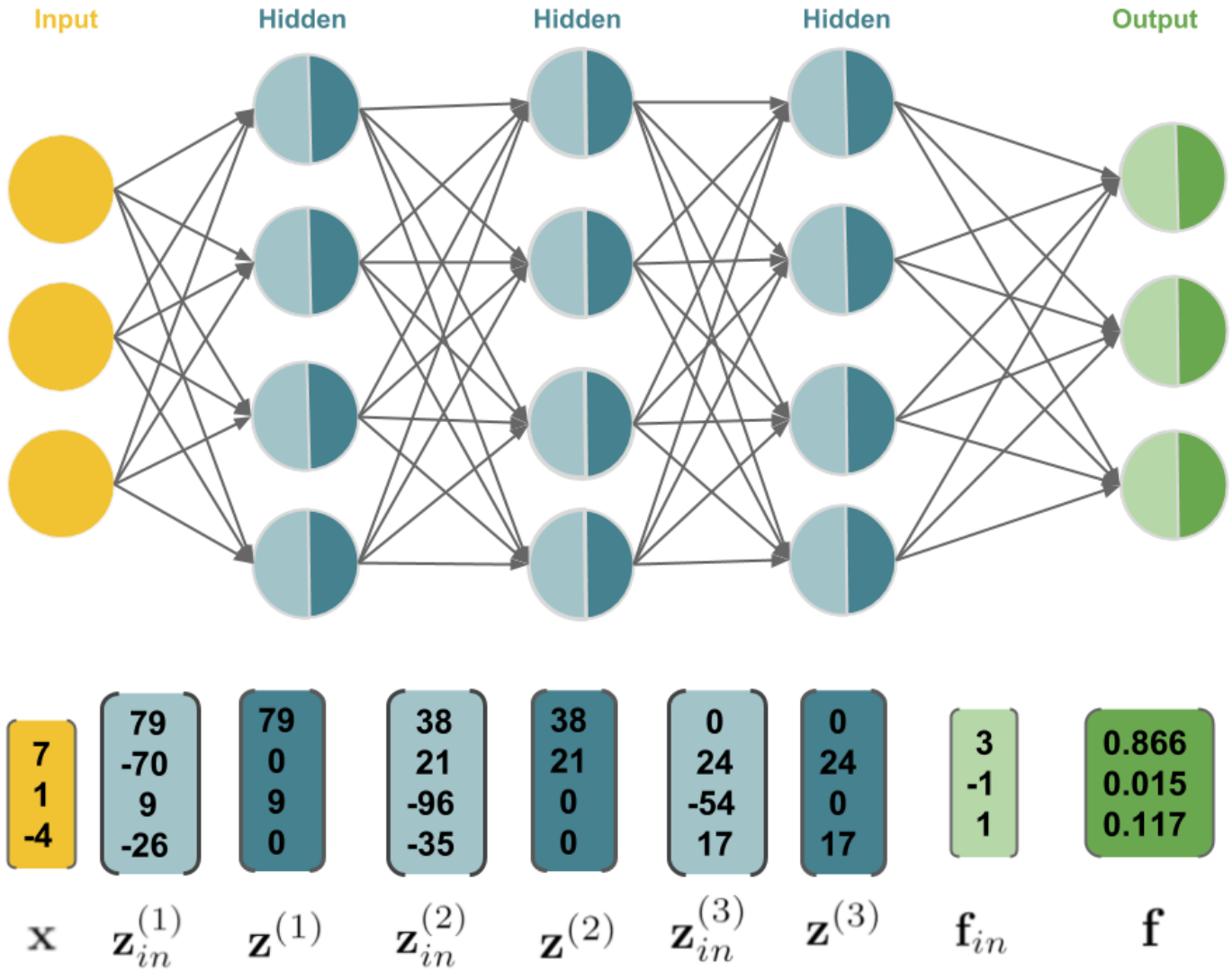
$$\mathbf{f}_{in}$$

3
-1
1

$$\frac{\exp(3)}{\exp 3+\exp(-1)+\exp(1)}$$
$$\frac{\exp(-1)}{\exp 3+\exp(-1)+\exp(1)}$$
$$\frac{\exp(1)}{\exp 3+\exp(-1)+\exp(1)}$$

$$\mathbf{f} = \mathbf{f}_{out} = \tau(\mathbf{f}_{in})$$

FEEDFORWARD NEURAL NETWORKS: EXAMPLE



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-lnearity to the model.

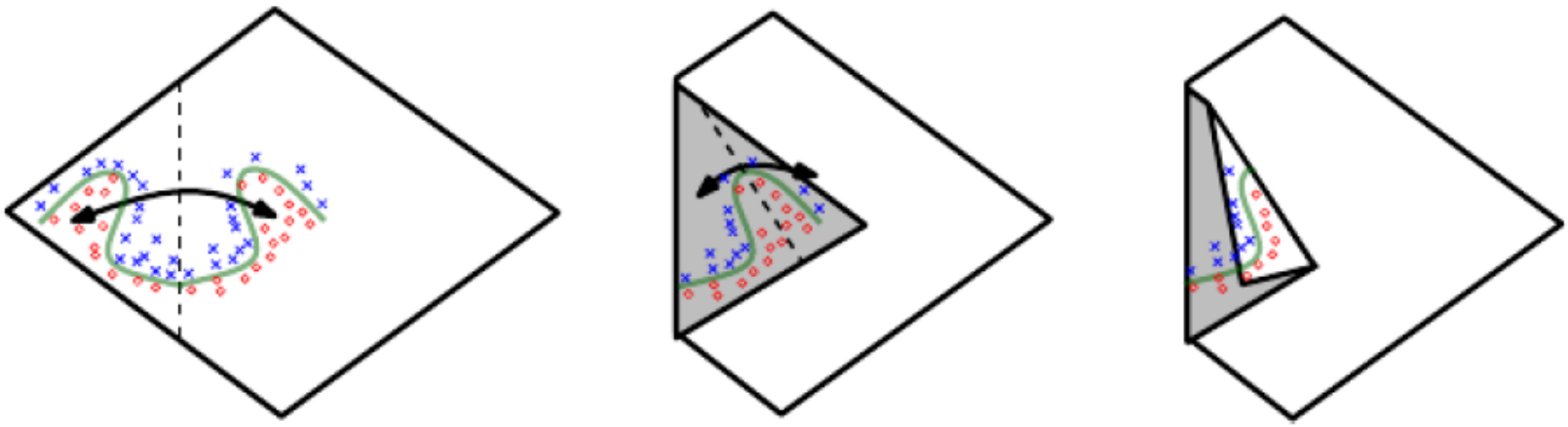


Figure: An intuitive, geometric explanation of the exponential advantage of deeper networks formally (Montúfar et al., 2014).

DEEP NEURAL NETWORKS

Neural networks today can have hundreds of hidden layers. The greater the number of layers, the "deeper" the network. Historically DNNs were very challenging to train and not popular until the late ’00s for several reasons:

- The use of sigmoid activations (e.g., 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.
- Training DNNs on CPUs was too slow to be practical. Switching over to GPUs cut down training time by more than an order of magnitude.
- When dataset sizes are small, other models (such as SVMs) and techniques (such as feature engineering) often outperform them.

DEEP NEURAL NETWORKS

- The availability of large datasets and novel architectures that are capable of handling even complex tensor-shaped data (e.g. CNNs for image data), faster hardware, and better optimization and regularization methods made it feasible to successfully implement deep neural networks.

- 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.

The term "**deep learning**" encompasses all of these developments and refers to the field as a whole.

REFERENCES

Montúfarr, G., Pascanu, R., Cho, K., & Bengio, Y. (2014). *On the Number of Linear Regions of Deep Neural Networks*.