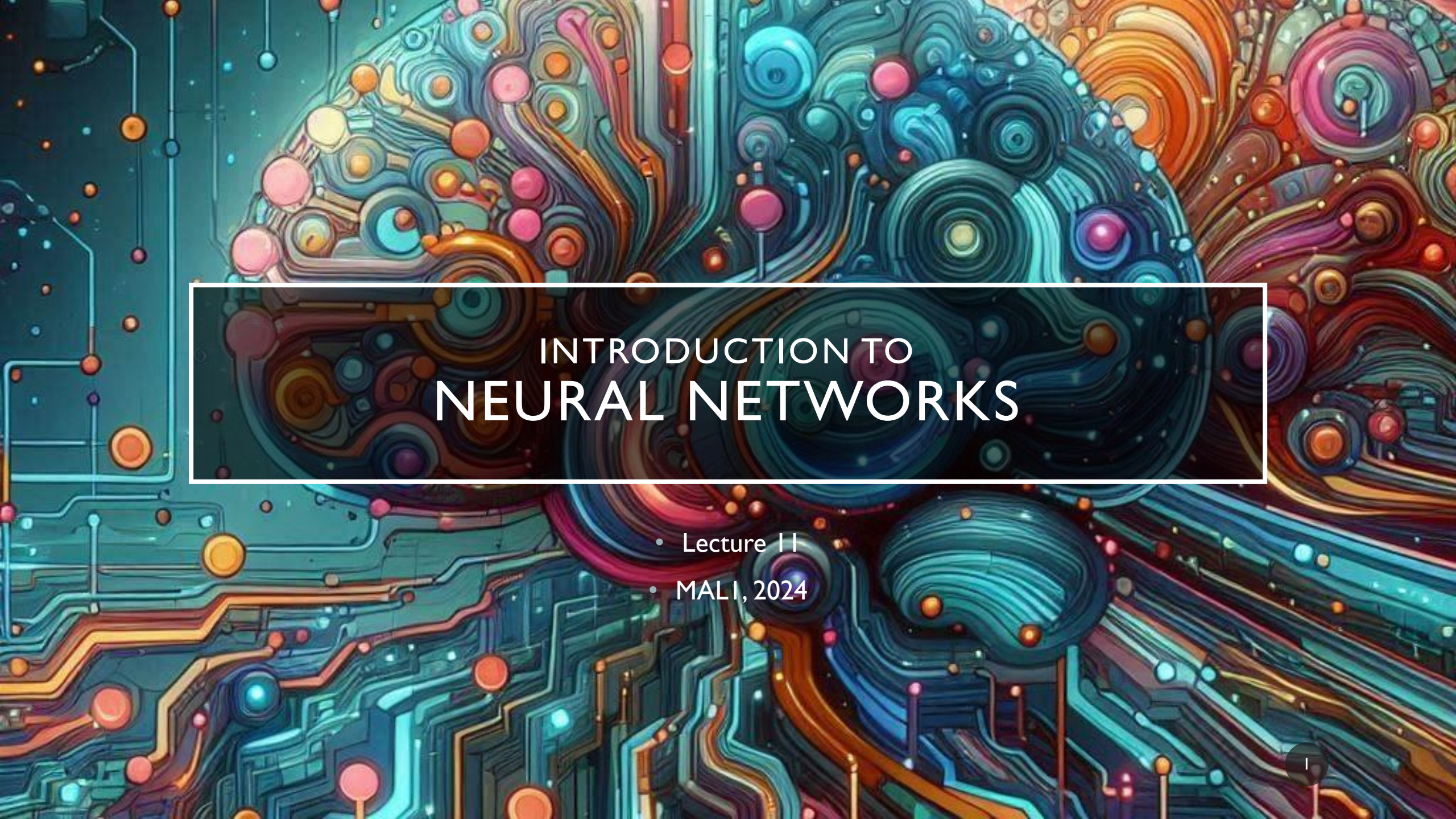
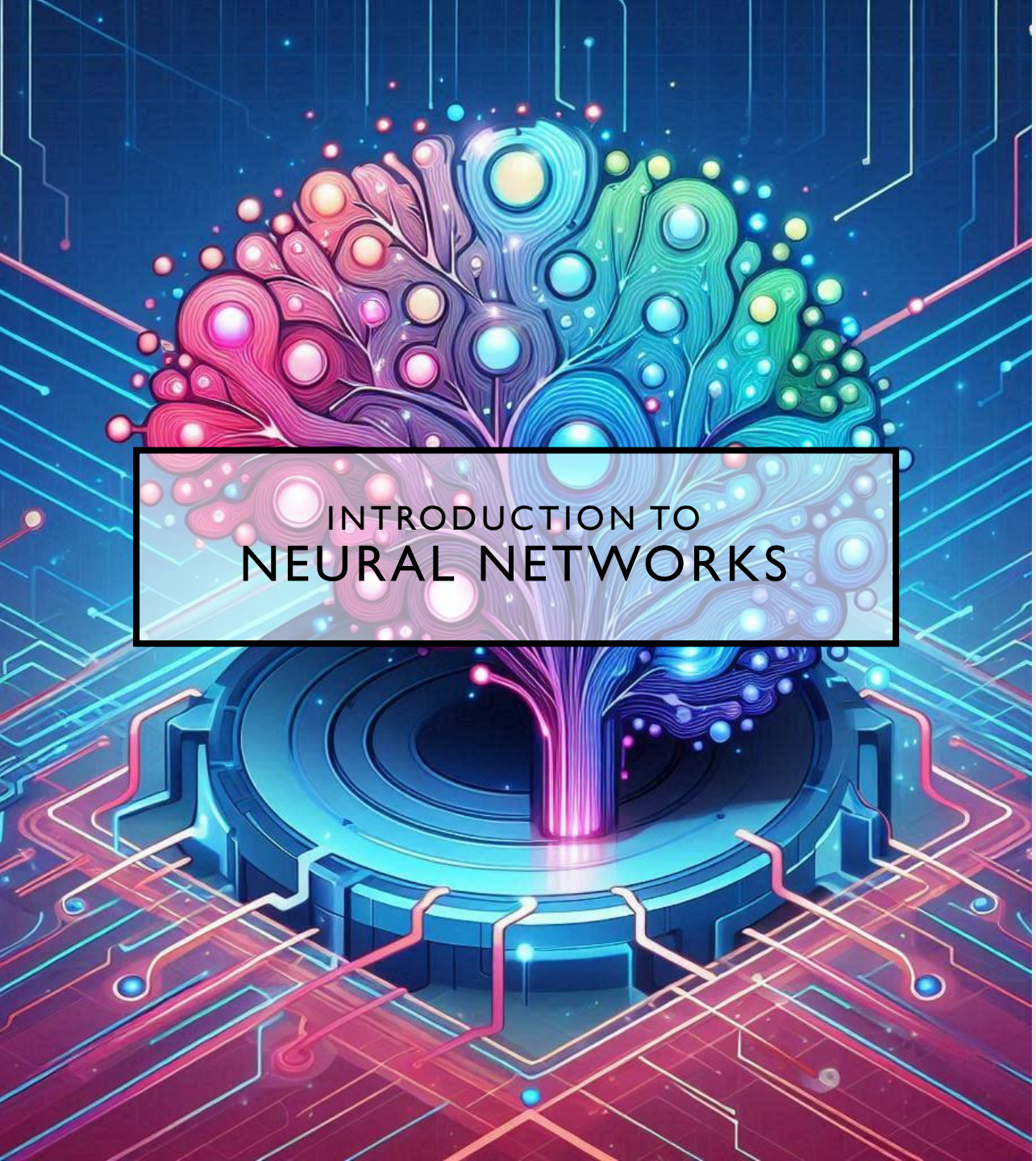


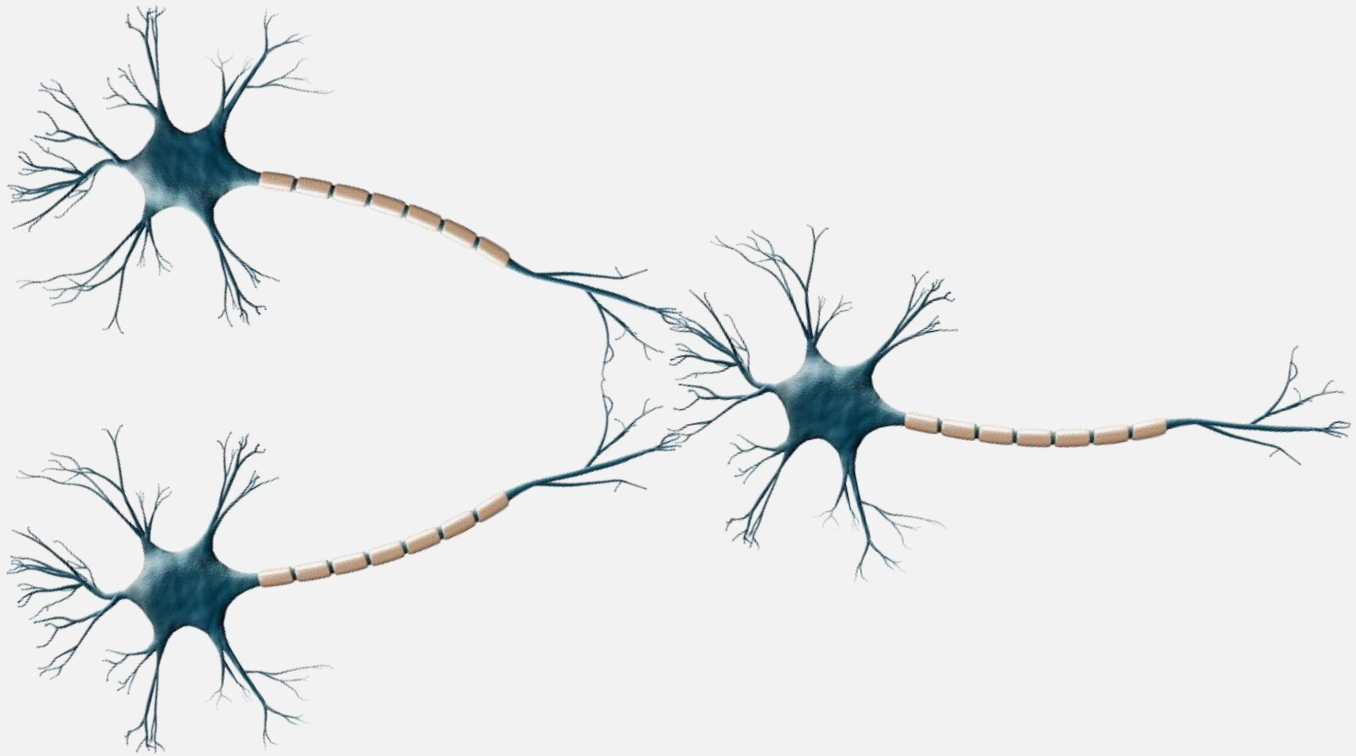
# INTRODUCTION TO NEURAL NETWORKS

- Lecture 11
- MAL1, 2024



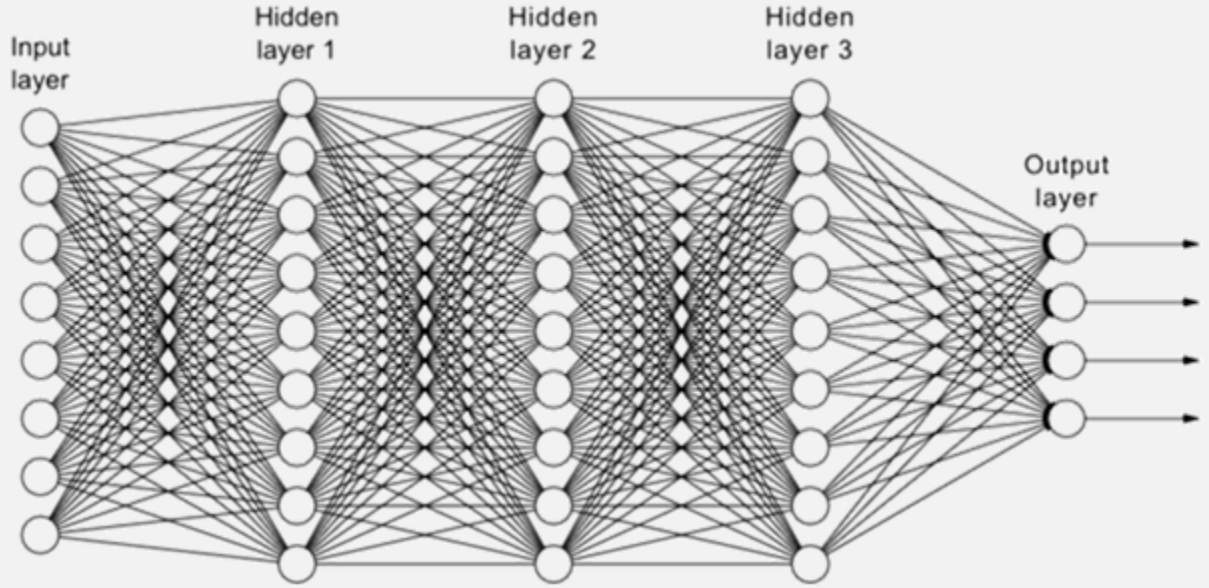

- What is a neural network?
- How do we structure it?
- How do we train it?
- How do we implement it?

# NEURONS



# ARTIFICIAL NEURONS

# ARTIFICIAL NEURAL NETWORKS

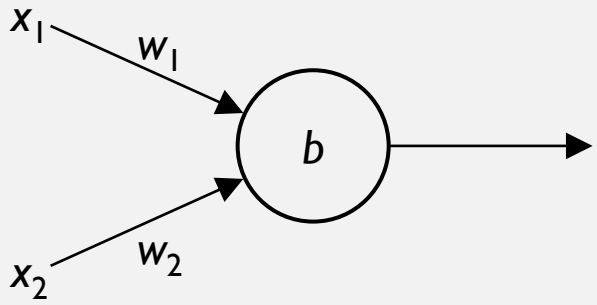


# HOWEVER …

**… a neural network has absolutely nothing to do with a brain.**

# NOTATION: WEIGHTS AND BIASES

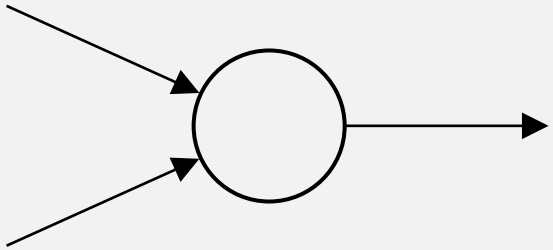
# PERCEPTRONS



# THE SNACK EXAMPLE

**You just sat down in the couch to watch your favorite tv show!**

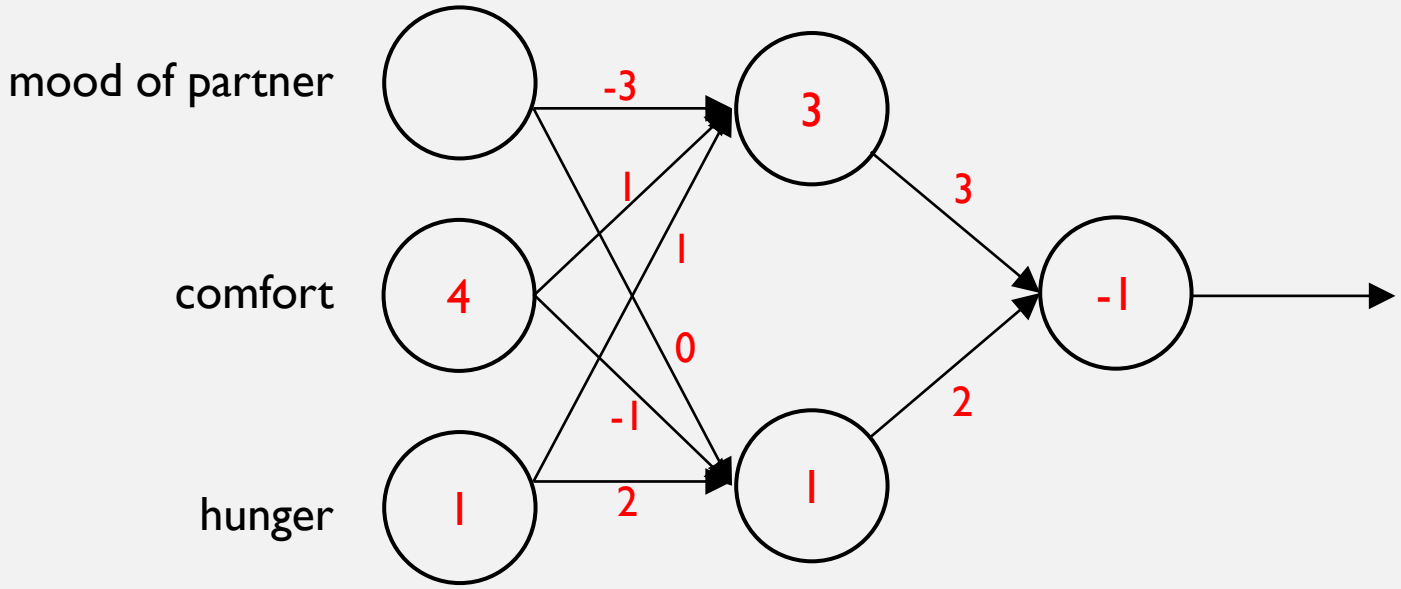
**Is it really worth it to get up again to get some snacks?**



# THE SNACK EXAMPLE II

**You decide that you absolutely want snacks.**

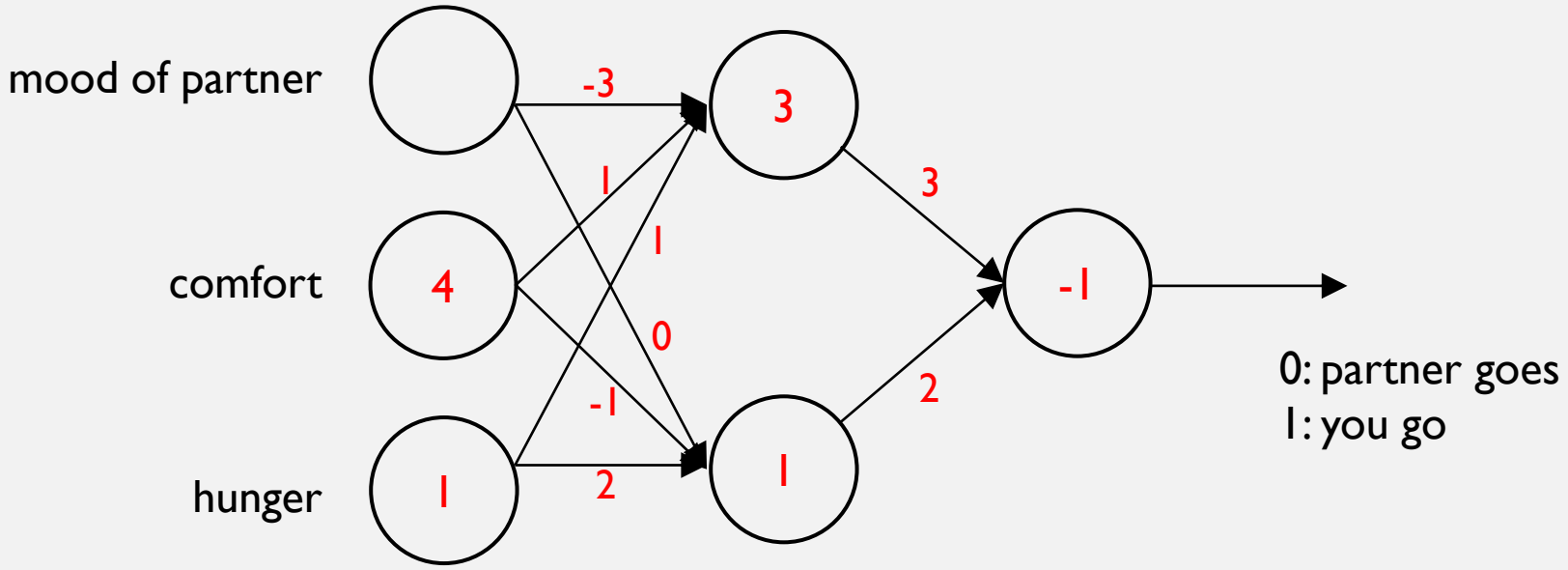
**Luckily, your partner is not sitting in the couch!**



# THE SNACK EXAMPLE II

**You decide that you absolutely want snacks.**

**Luckily, your partner is not sitting in the couch!**

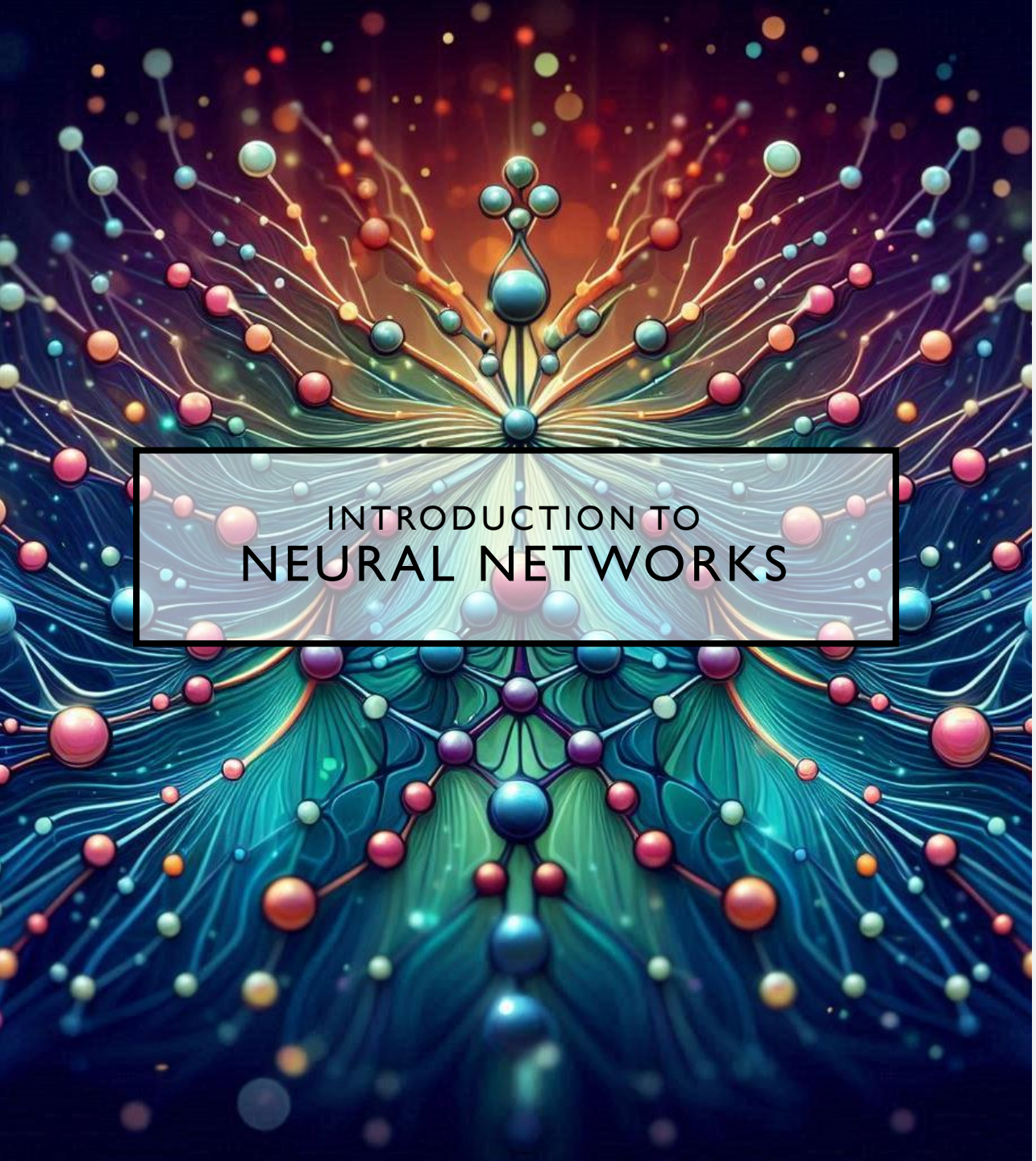


# TRAINING A NEURAL NETWORK

# THIS LEAVES TWO QUESTIONS

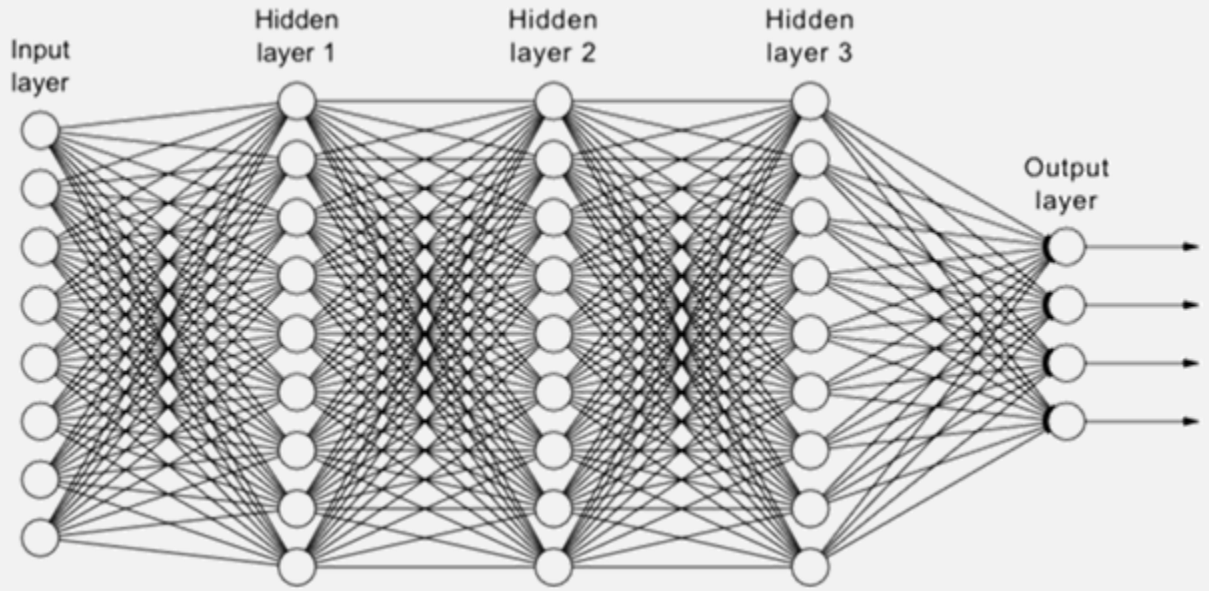
# THE TENSORFLOW PLAYGROUND

playground.tensorflow.org

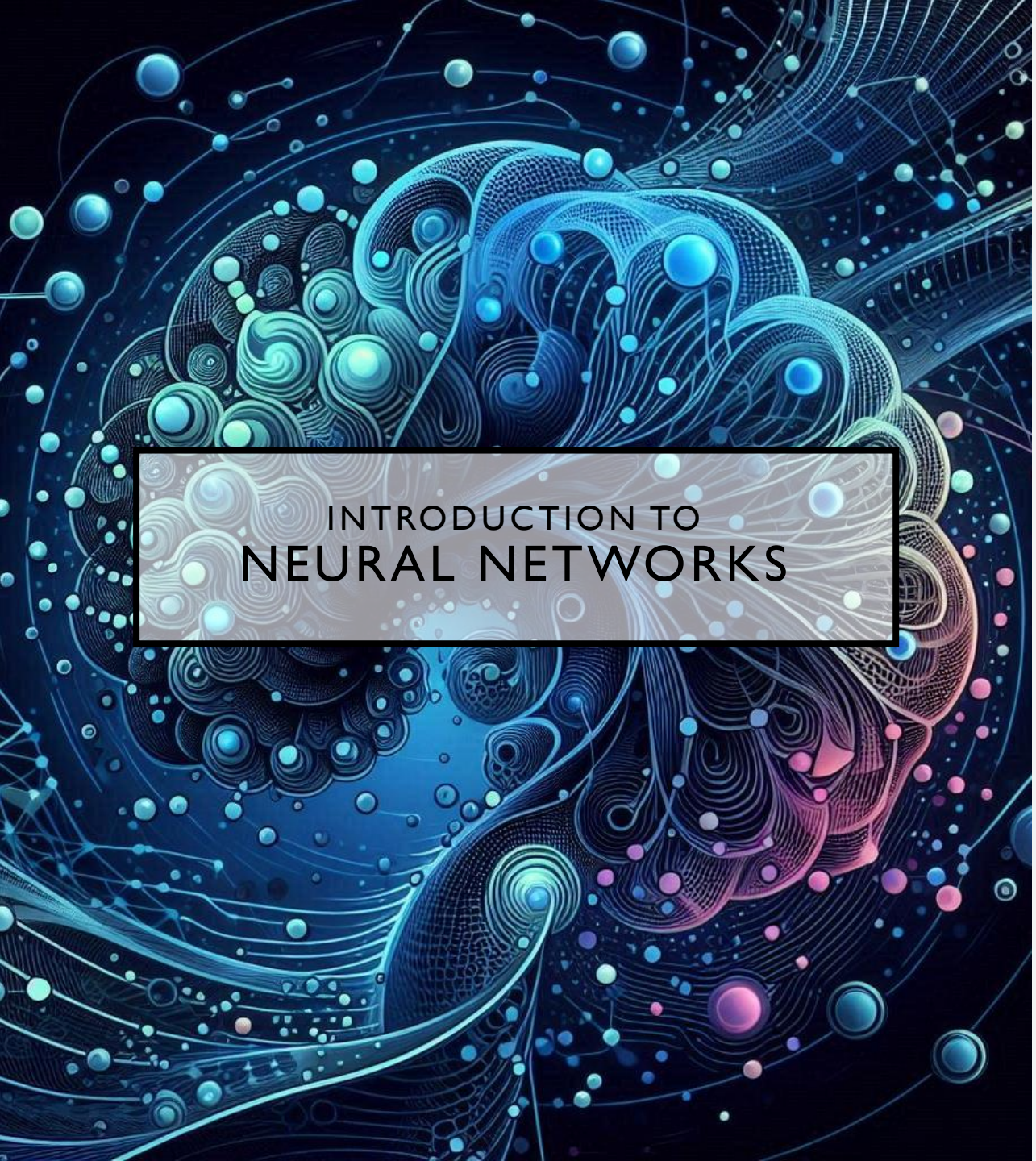



- What is a neural network?
- How do we structure it?
- How do we train it?
- How do we implement it?

# HOW TO STRUCTURE THE NETWORK




**Rules of thumb:**




- What is a neural network?
- How do we structure it?
- How do we train it?
- How do we implement it?

# HOW TO OPTIMIZE WEIGHTS AND BIASES

**Longitude & latitude:**

**Altitude:**

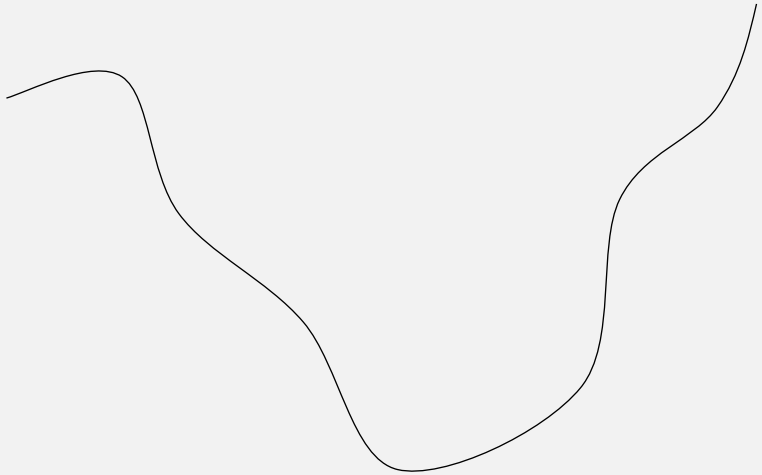
# GRADIENT DESCENT

1. Find the direction in which the descent is steepest

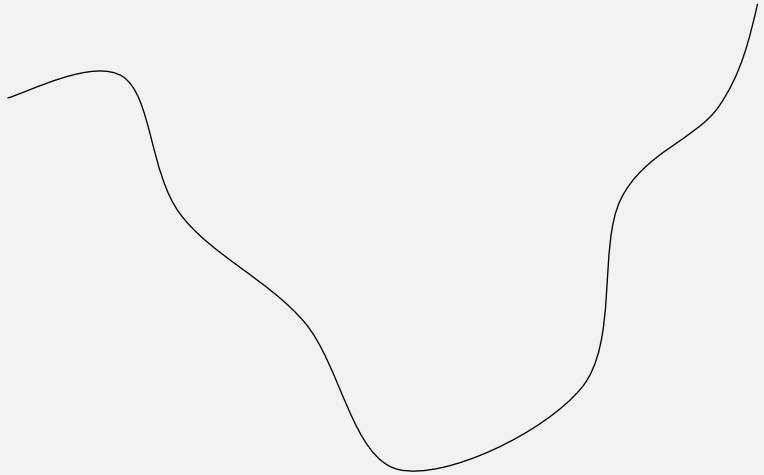
2. Take a step in that direction

3. Repeat until you reach the bottom

# THE LEARNING RATE



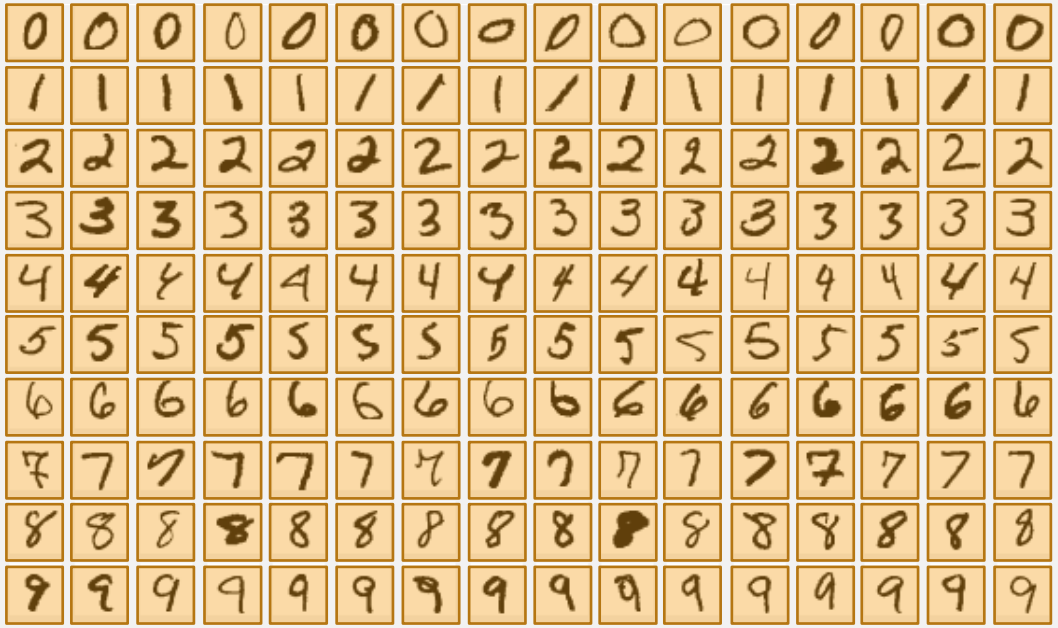
**η too big**



**η too small**

# STOCHASTIC GRADIENT DESCENT

Don't update weights and biases based on **all** your data every time. Instead,



When you have gone through all your samples, you finish a **training epoch.**

# BUT WHAT IF …

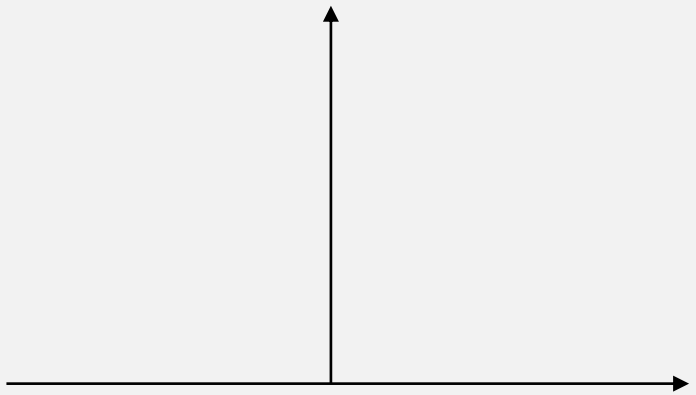
# SUDDEN CHANGES

1. Perceptrons

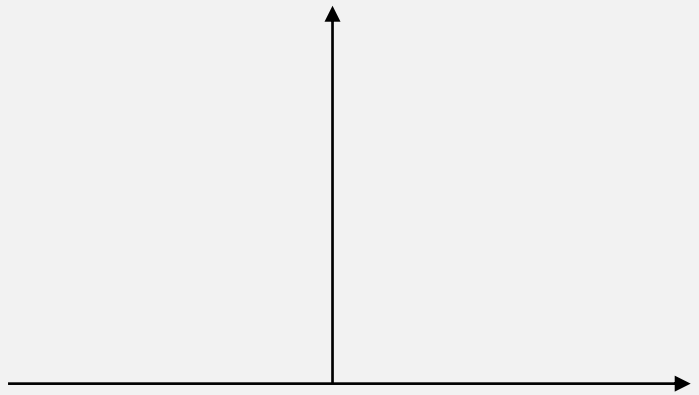
2. The number of misclassifications

# FIXING THE PERCEPTRON PROBLEM

$$\text{output} = \begin{cases} 0 & \text{if} \quad \boldsymbol{wx} + b \leq 0 \\ 1 & \text{if} \quad \boldsymbol{wx} + b > 0 \end{cases}$$

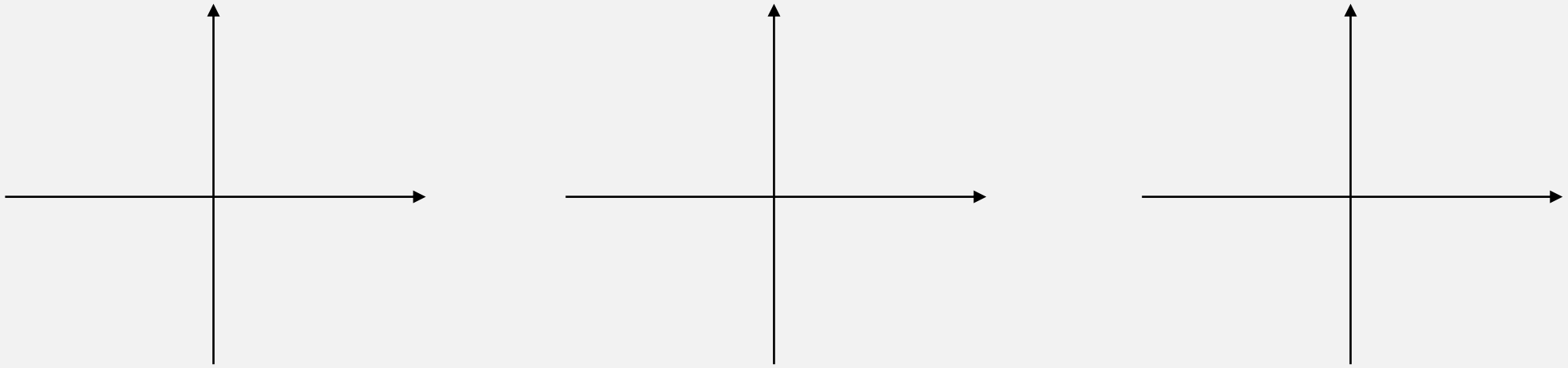


$$\text{output} = \frac{1}{1 + e^{-(\boldsymbol{wx}+b)}}$$



# DIFFERENT ACTIVATION FUNCTIONS

**sigmoid** **tanh** **ReLU**



# ACTIVATION IN THE OUTPUT LAYER

# FIXING THE ACCURACY PROBLEM

# THE LOSS FUNCTION

**The quadratic loss function**

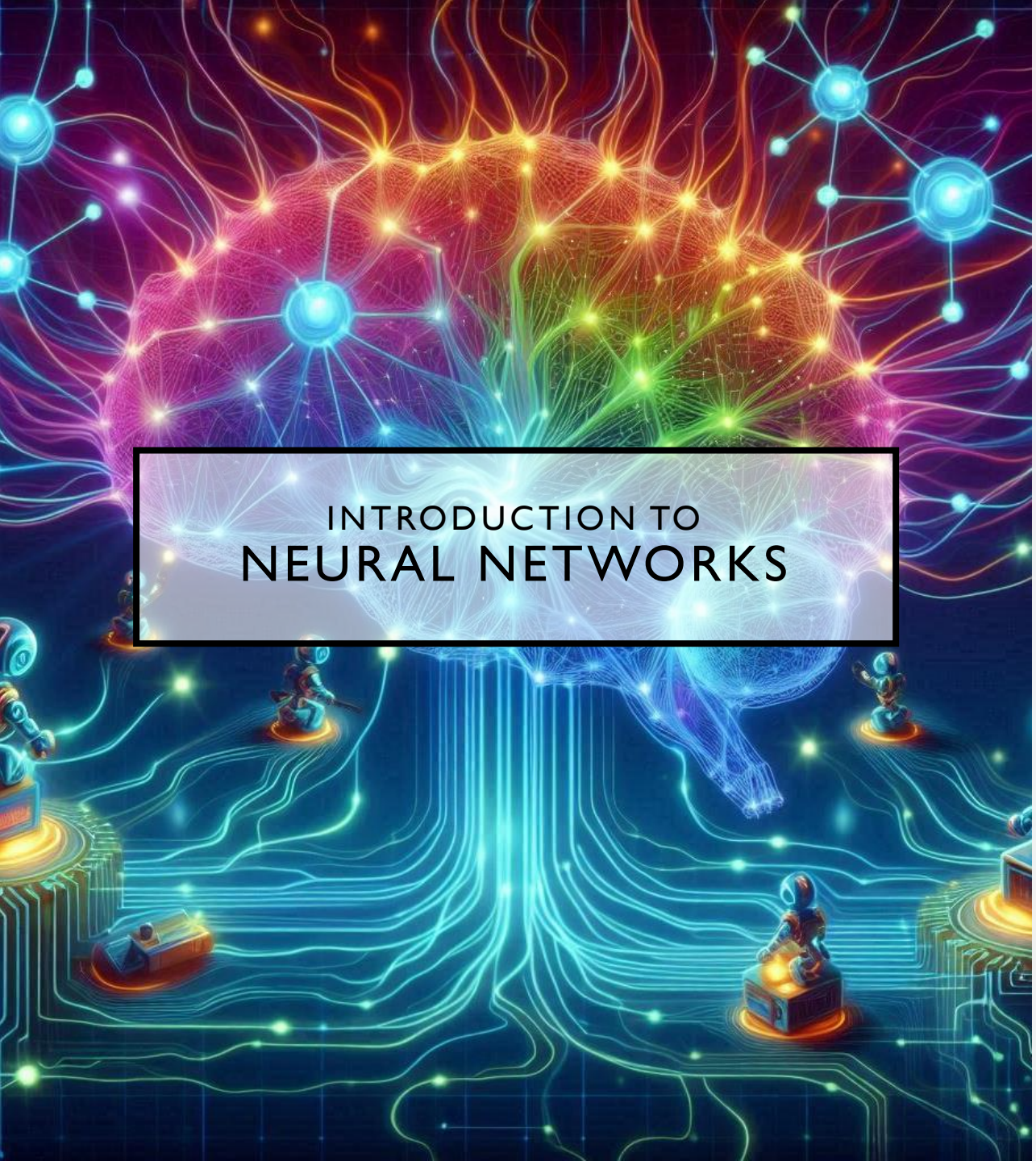
$$L(\boldsymbol{w}, \boldsymbol{b}) = \frac{1}{2n} \sum_{x} \left| \left| y(x) - a(x, \boldsymbol{w}, \boldsymbol{b}) \right| \right|^2$$

**but usually we use**

Example from before

$$y(x_1) = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

$$a(x_1, \boldsymbol{w}, \boldsymbol{b}) = \begin{bmatrix} 0.1 \\ 0.8 \\ 0.07 \\ 0.03 \end{bmatrix}$$




- What is a neural network?
- How do we structure it?
- How do we train it?
- How do we implement it?

# LET'S TRY TO MAKE ONE

*Jupyter Notebook* **Neural networks - Digits**