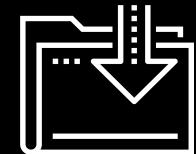


# Neural Networks

Fintech

Lesson 13.1



# Class Objectives

---

By the end of the class, you will be able to:



Explain the concept of neural networks, including how a neuron is related to logistic regression.



Preprocess data for neural network models.



Explain the intuition of how weights of neurons are determined.



List the Python libraries available for building neural networks.



Describe some of the differences between neural networks involving regression versus those involving classification.



Describe the pros and cons of using Keras for building neural networks.



Use the Quick, Draw! web app to experiment with neural network applications.



Implement neural networks with the TensorFlow Keras API.



**WELCOME**



What do you think a neural network is?

# Sample Answers

---

01

It's a computer representation of a human neuron.

02

It's a method to create artificial intelligence software.

03

Neural networks are machine learning algorithms that can learn to solve a problem.



Do you know some cool applications  
of neural networks?



Sample Answer:  
—  
Autonomous  
(self-driving) cars

**Sample Answer:**

---

Image processing  
in cancer detection



## **Sample Answer:**

---

Banning inappropriate  
content on  
social networks





**Sample Answer:**

—

Automated trading  
based on artificial  
intelligence

# Questions?



# Homework Preview

The venture capital industry uses data and models to determine how to make investments.



# Homework Preview

---

In this homework assignment, you will:



Take on the role of a venture capital analyst.



Use models to quantify the factors that lead to investing success and to evaluate future deals.



Create a neural network model that will predict whether an applicant will become a successful business.



You will learn how to build, evaluate, and optimize deep neural networks by using a “model-fit-predict” pattern.

# Questions?



# What Is a Neural Network?

# What Is a Neural Network?

---

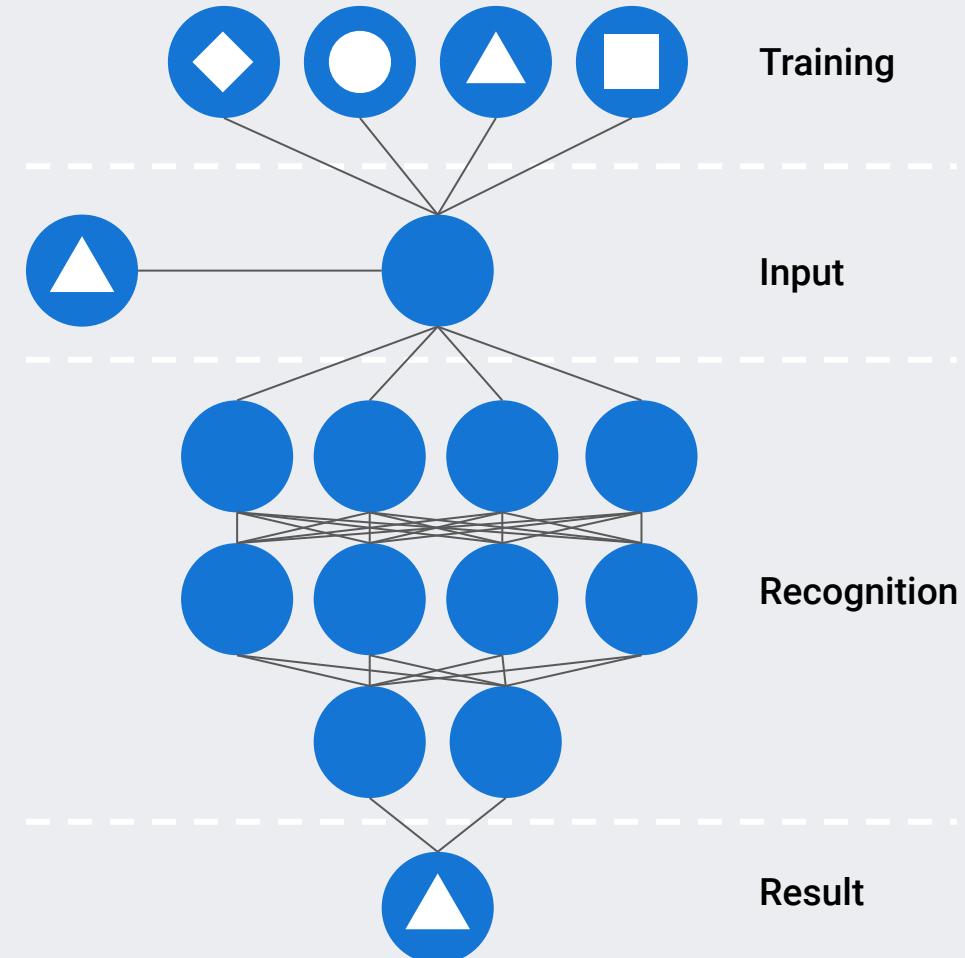
Neural networks, also known as artificial neural networks (ANN), are a set of algorithms that are modeled after the human brain.



# What Is a Neural Network?

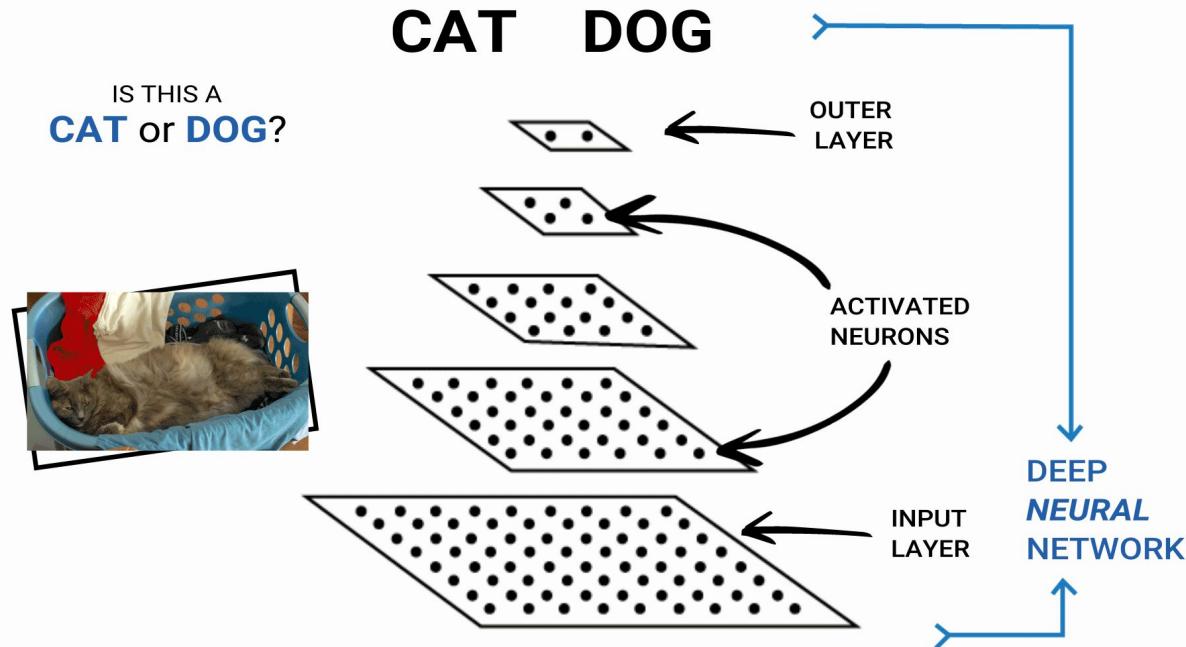
Neural networks are an advanced form of machine learning that:

- Recognizes patterns and features of input data
- Provides a clear quantitative output



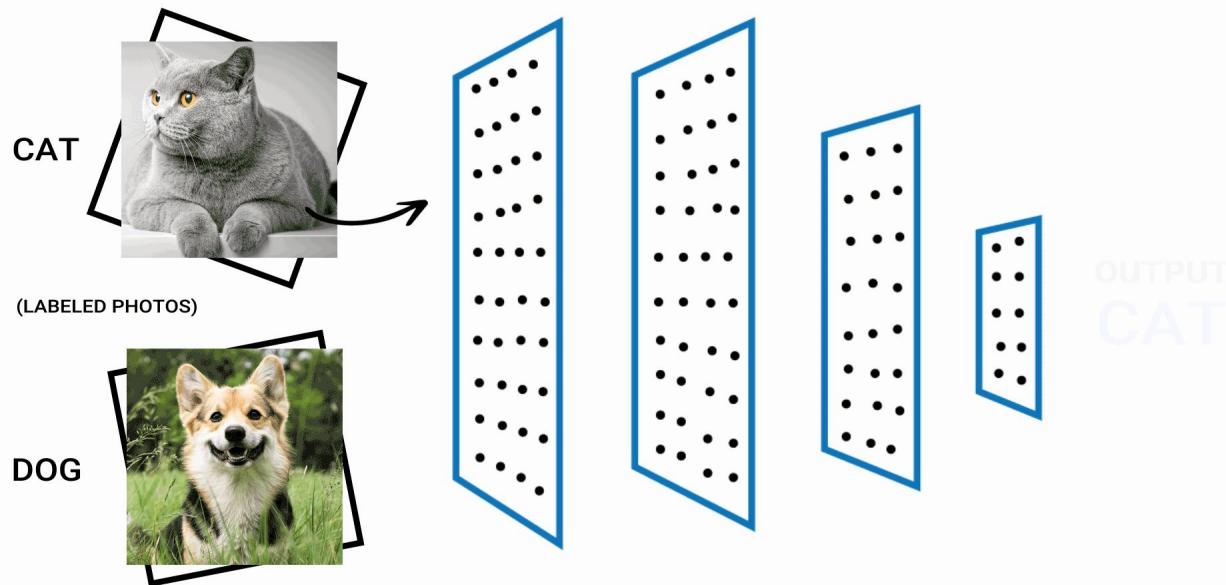
# What Is a Neural Network?

In its simplest form, a neural network contains layers of neurons that perform individual computations.



# What Is a Neural Network?

These computations are connected and weighed against one another until the neurons reach the final layer. In the final layer, the neurons return either a numerical result or an encoded categorical result.





Using a neural network instead  
of a traditional statistical or  
machine learning model  
provides many advantages.

# Advantages of Neural Networks

---



**Neural networks can detect complex relationships within datasets.**



**Neural networks are versatile.**

They can be used to predict future shopping behavior based on credit card transactions or to assess a loan applicant's likelihood of defaulting on a loan based on their application.



**Neural networks have greater tolerance for messy data.**

They can learn to ignore noisy characteristics within a large dataset.



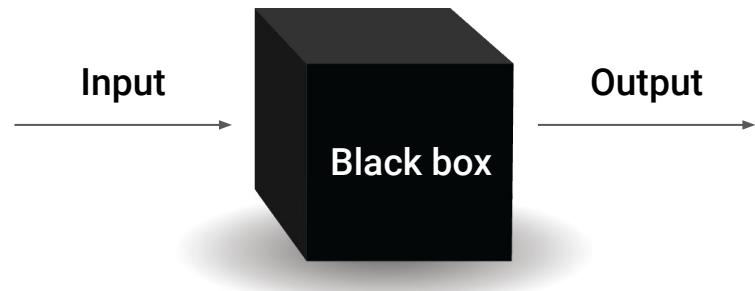
There are two primary **disadvantages** of using a neural network.

# Disadvantages of Neural Networks



Neural network algorithms can be too complex for humans to understand.

This is an example of something known as a **“black box” problem**, where the inputs are not visible to the user, and the resulting output cannot be traced back to discover the logic that was used.

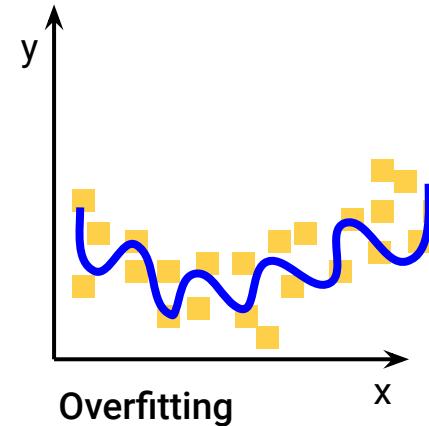


# Disadvantages of Neural Networks



Neural networks are more likely to overfit the data.

**Overfitting** occurs when a model gives too much importance to patterns within the training data that are not found in the test data. This results in a model learning the training data so well that it cannot create generalizations about other data.





Fortunately, we can use different model designs and optimization techniques to compensate for and negate both of these disadvantages.

# Neural Networks in Finance

---

Neural networks serve multiple purposes in the finance industry:



Fraud detection



Risk management



Money laundering prevention



Algorithmic stock trading

# Fraud Detection

---

\$118 billion worth of credit card transactions are incorrectly identified as fraud and declined every year.

Neural networks can efficiently evaluate the characteristics of a credit card transaction or loan application, allowing financial firms to predict fraud with more accuracy.

This saves time and money.



# Risk Management

The banking sector, insurance industry, and stock exchanges are all building more robust and efficient techniques for credit management by using neural networks.



# Money Laundering Prevention

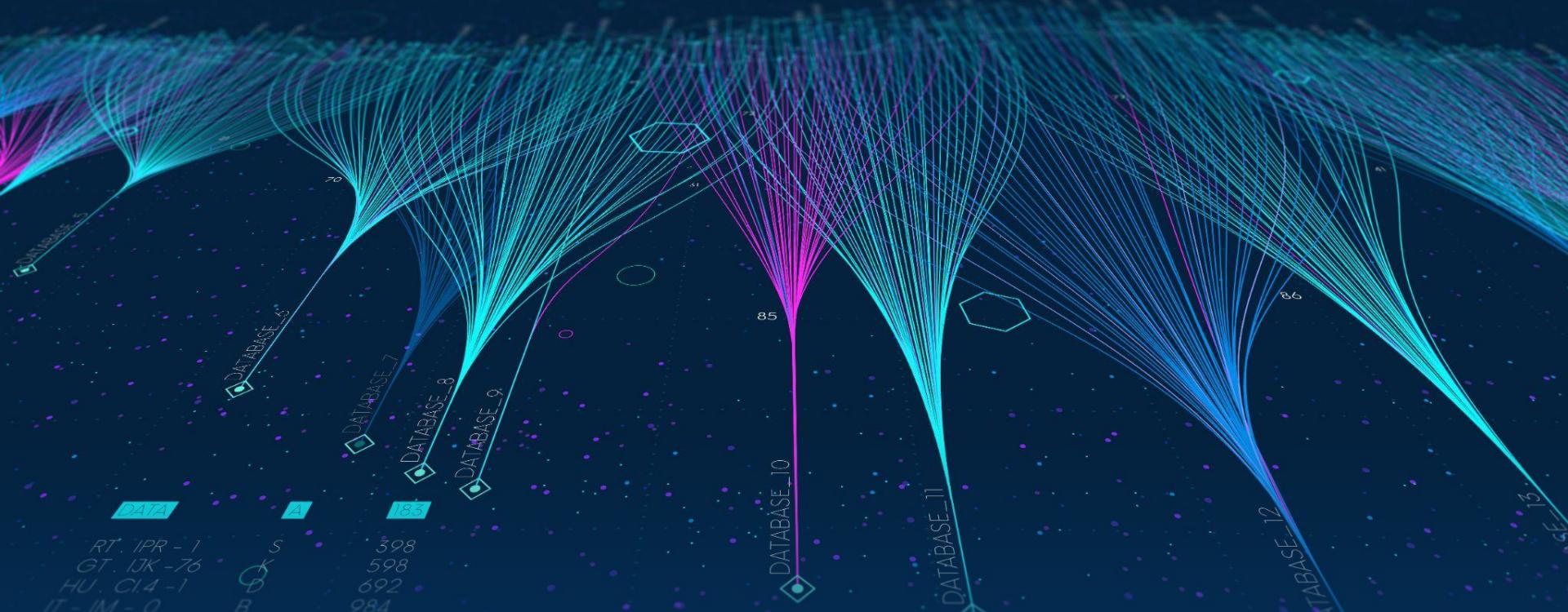
---

A United Nations study estimates that money laundering activity around the world accounts for 2–5% of annual global GDP.

Financial institutions now use neural nets as a tool in their fight against financial crime.



# Algorithmic Stock Trading



Thanks to trading algorithms that incorporate neural networks, you can automate your trading strategies and increase your trading profits.



Before we get into the code, let's review **the components of a neural network** and how they work together to learn.

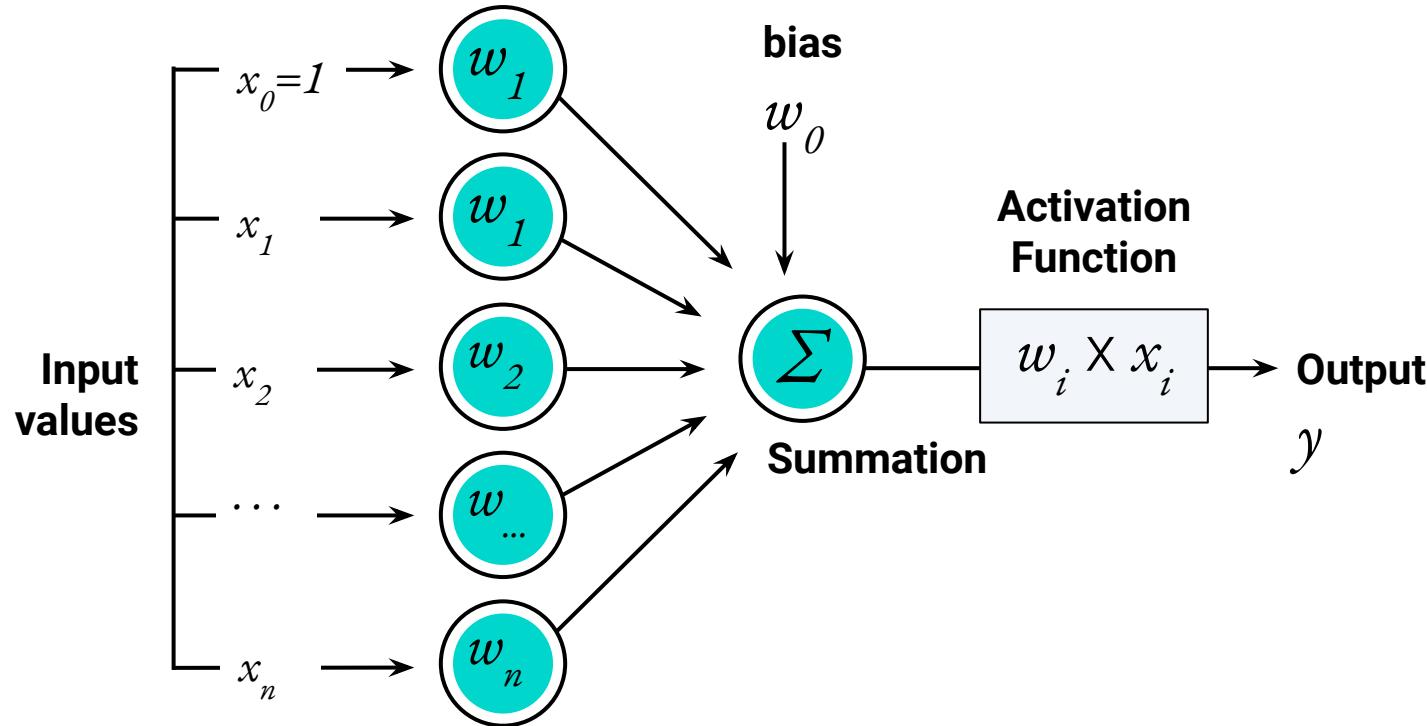
# Perceptron, the Computational Neuron

Artificial neural networks have become popular in recent years. But, the original design for a computational neuron, known as a **perceptron**, dates back to the late



# Meet the Perceptron

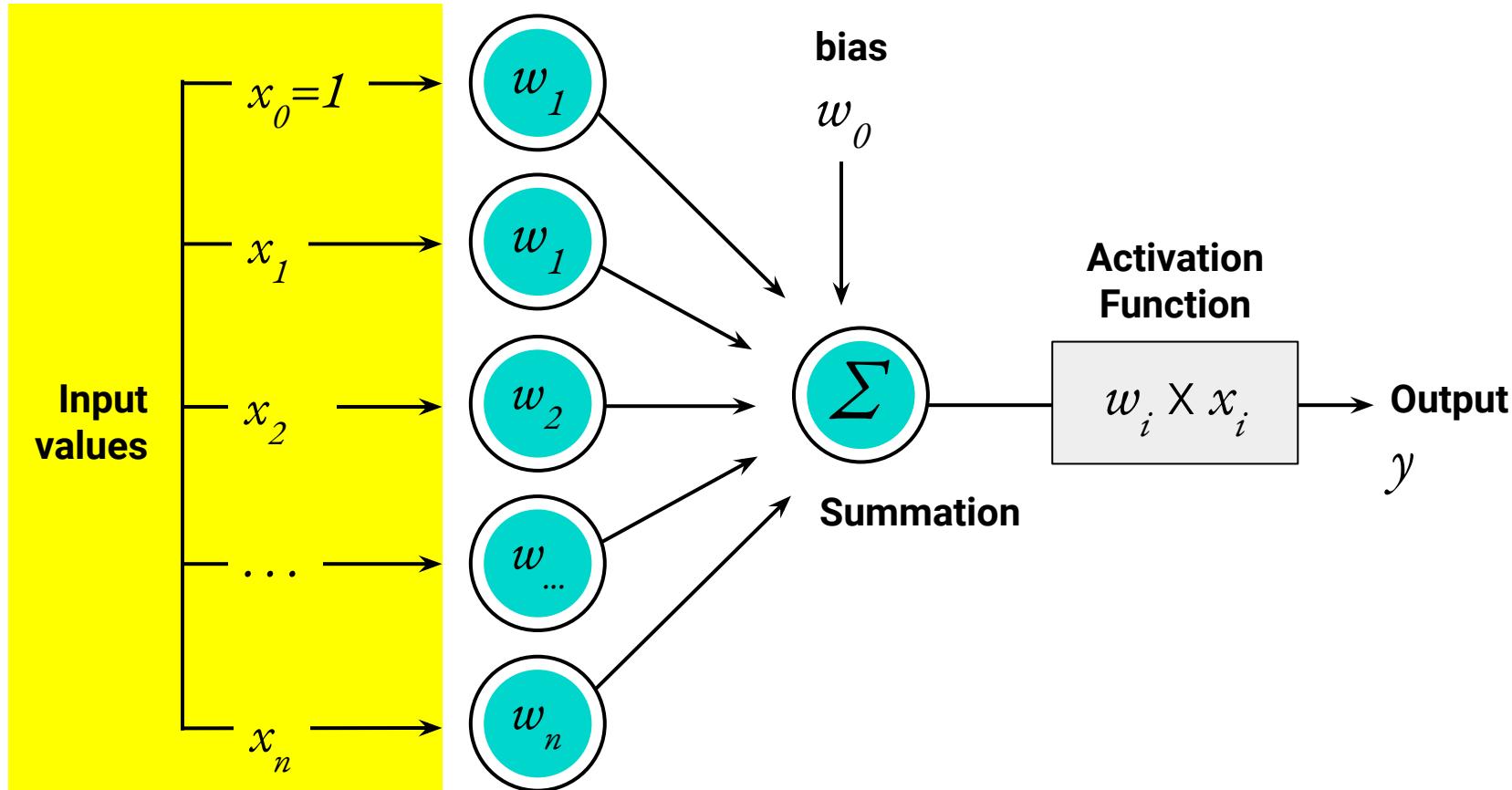
The **perceptron model** mimics a biological neuron by receiving input data, weighting the information, and producing a clear output.



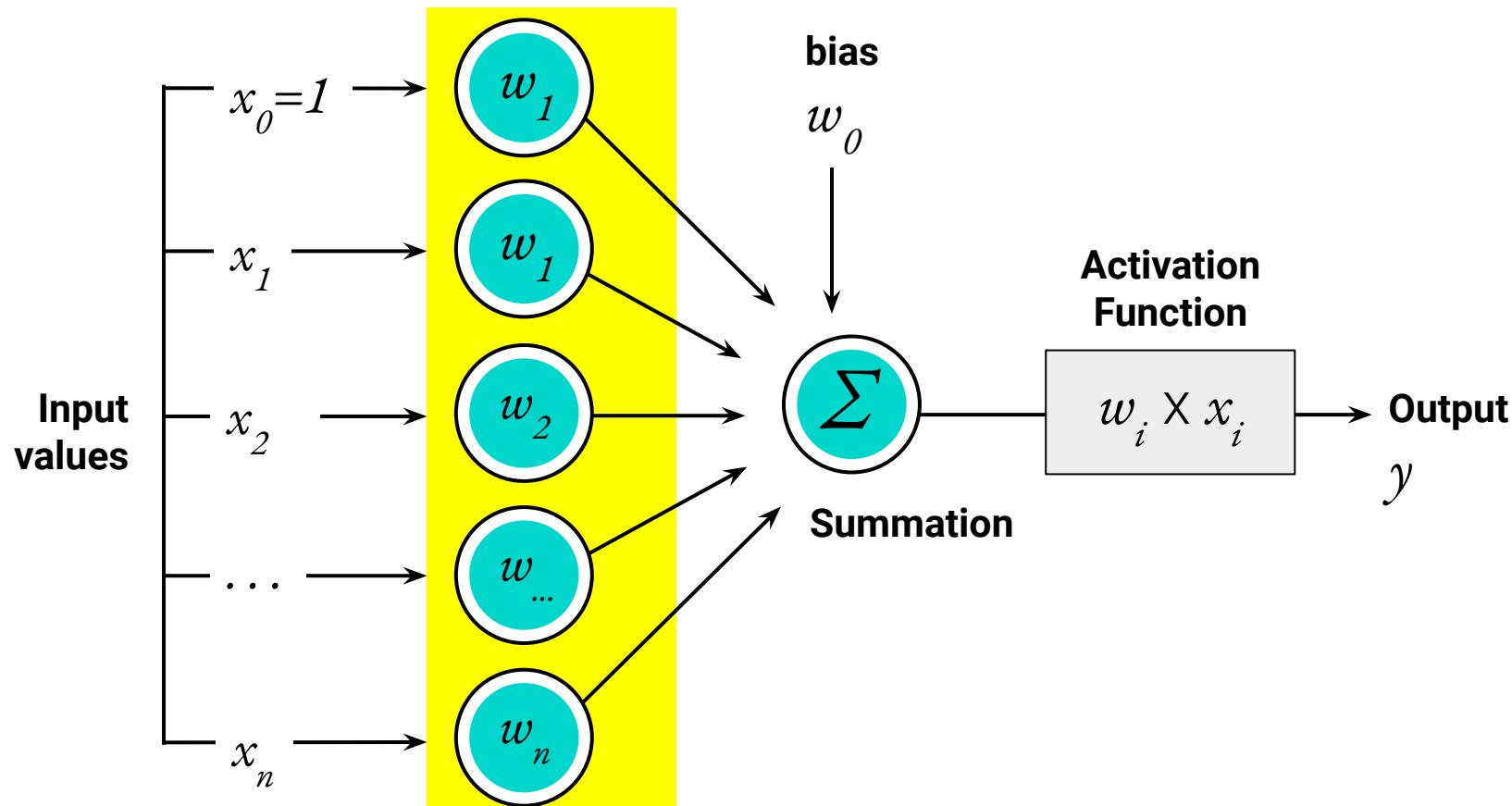


The perceptron model has  
**four major components.**

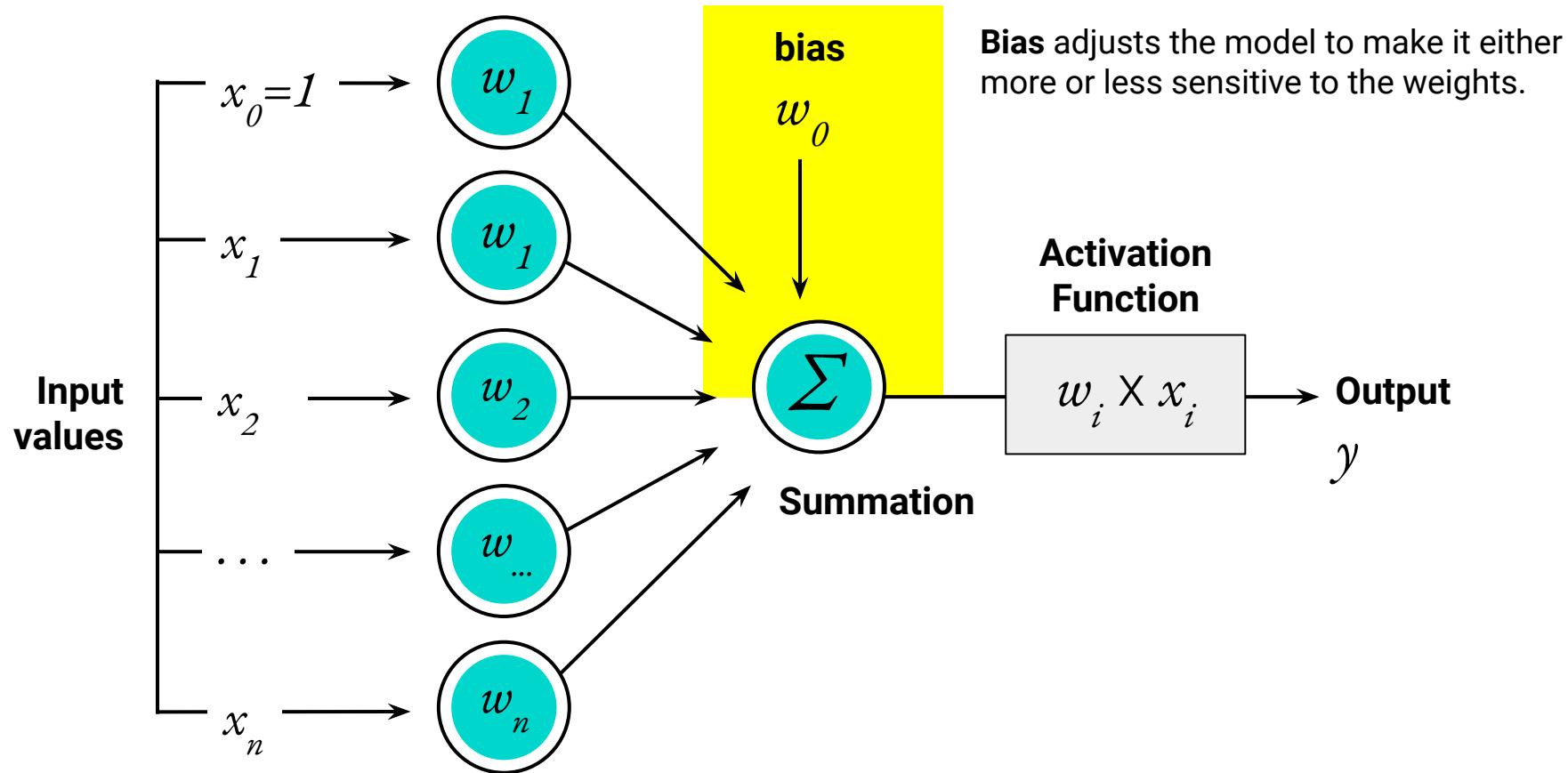
# Input Values (shown as x's)



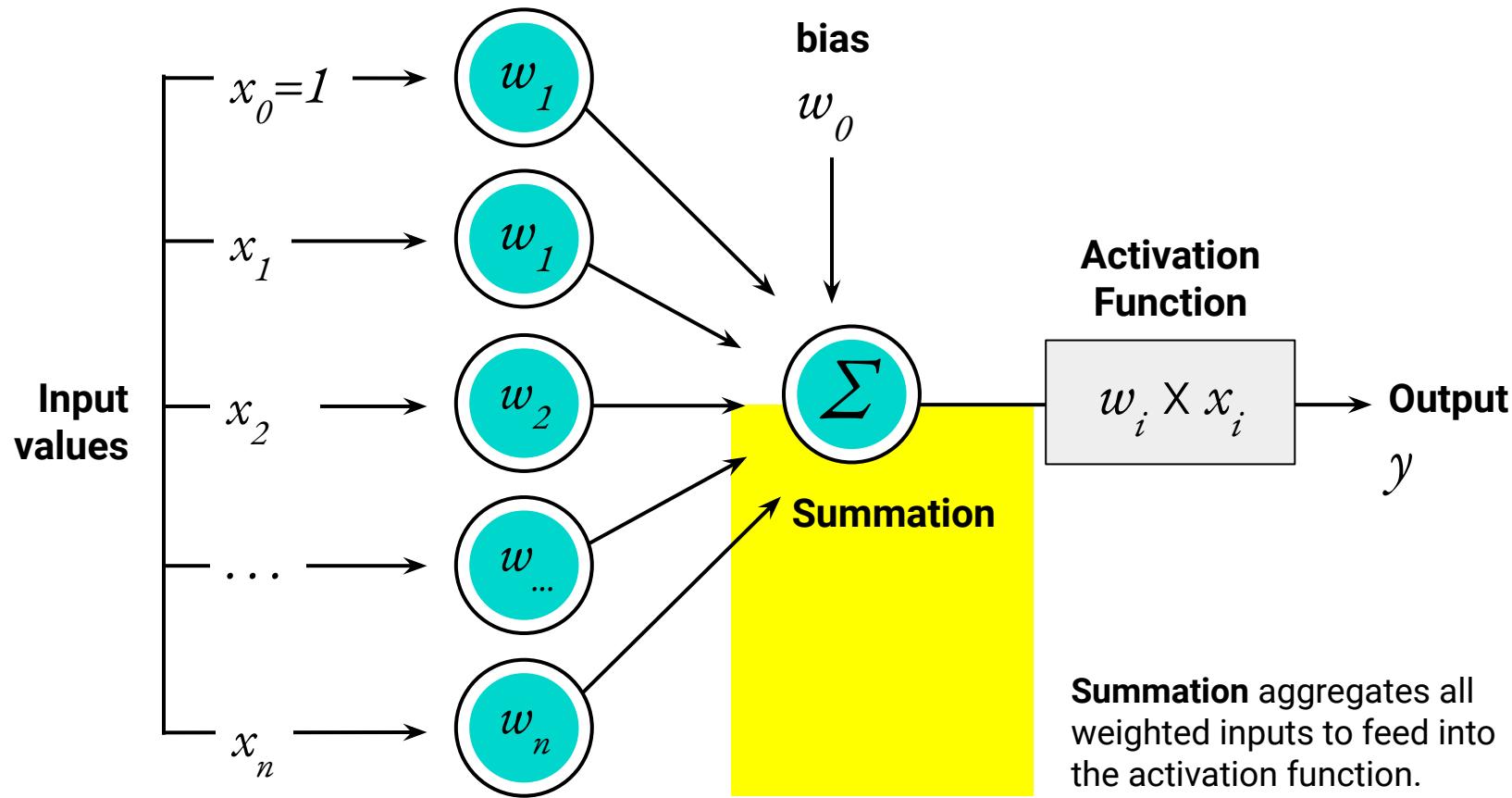
# Weight Coefficients (denoted as w's)



# A Constant Value Called Bias



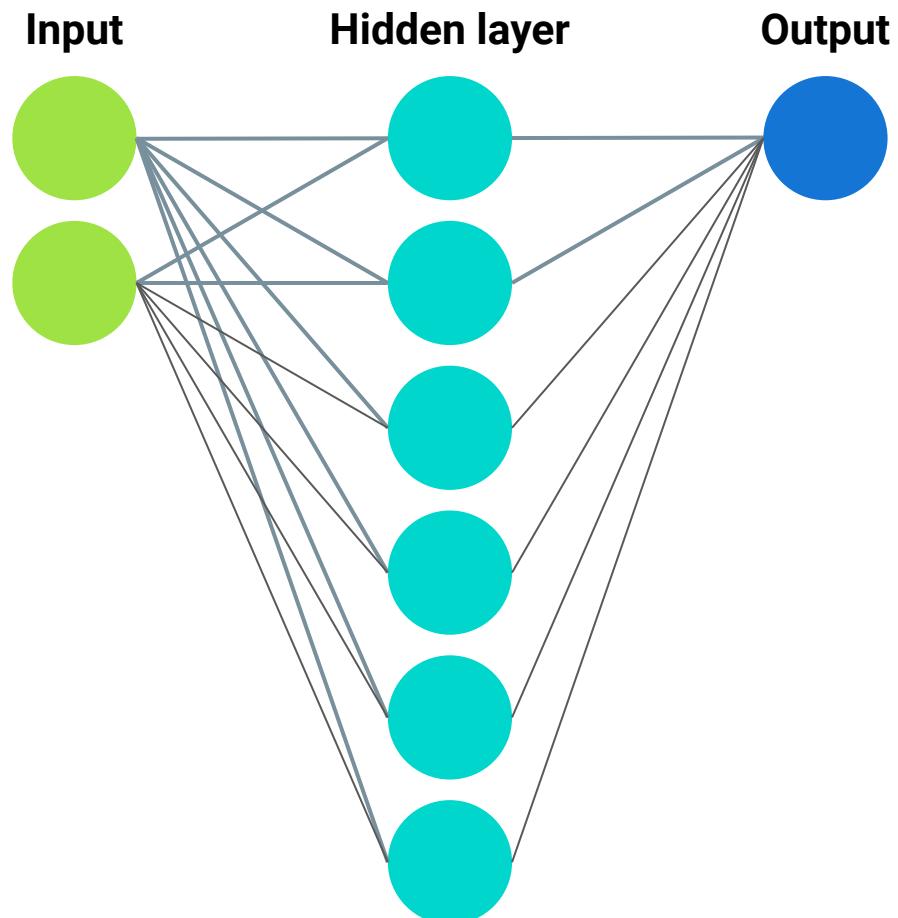
# Net Summary Function (the Summation)



# The Neural Network

---

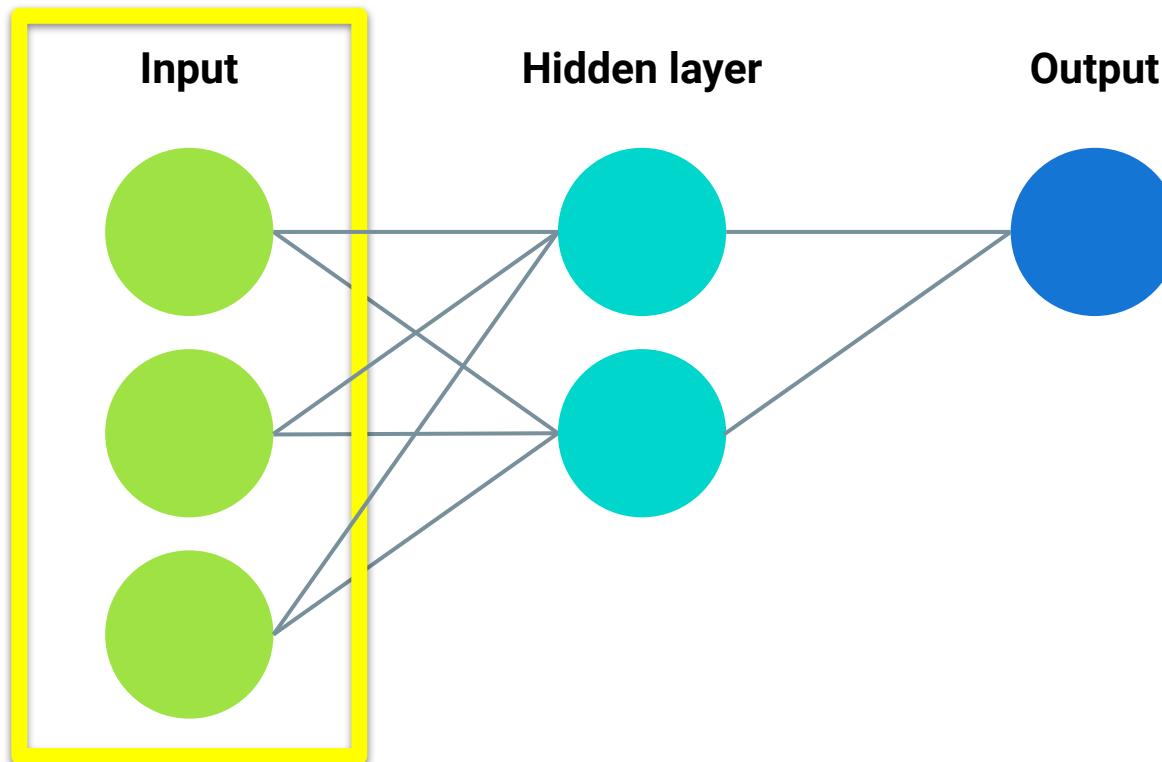
A modern neural network model is a structure composed of several connected perceptrons that learn from input data to produce an output.



# The Structure of a Neural Network

---

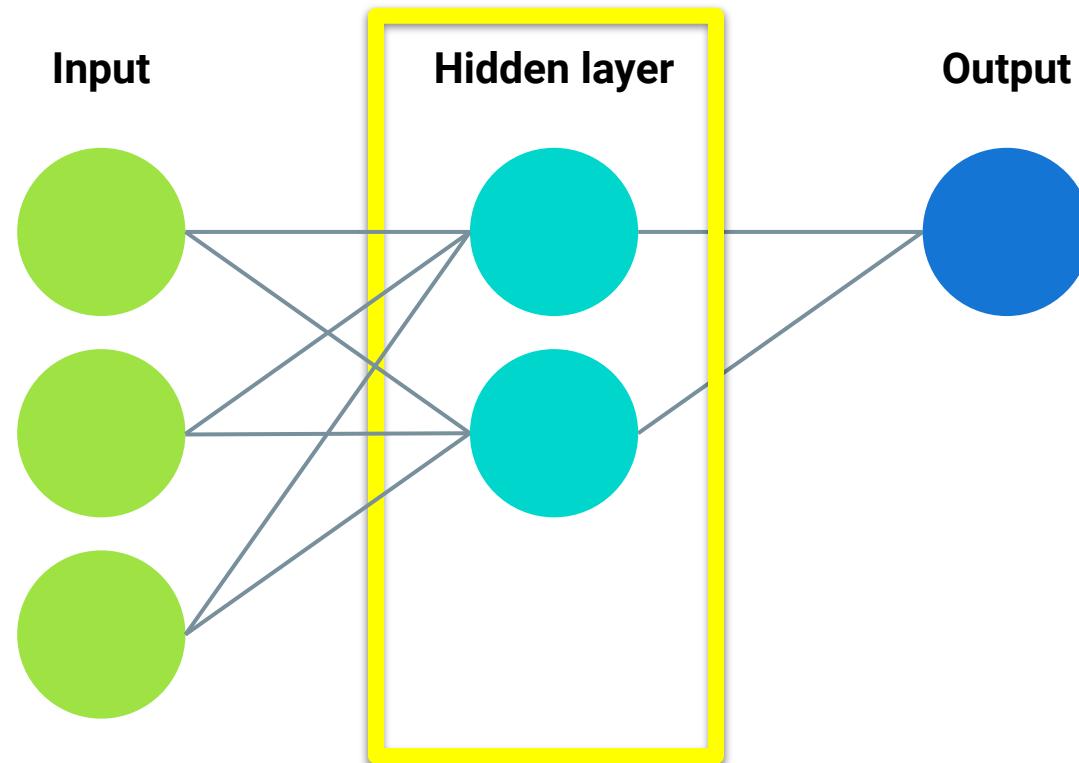
An **input layer** of input values transformed by weight coefficients



# The Structure of a Neural Network

---

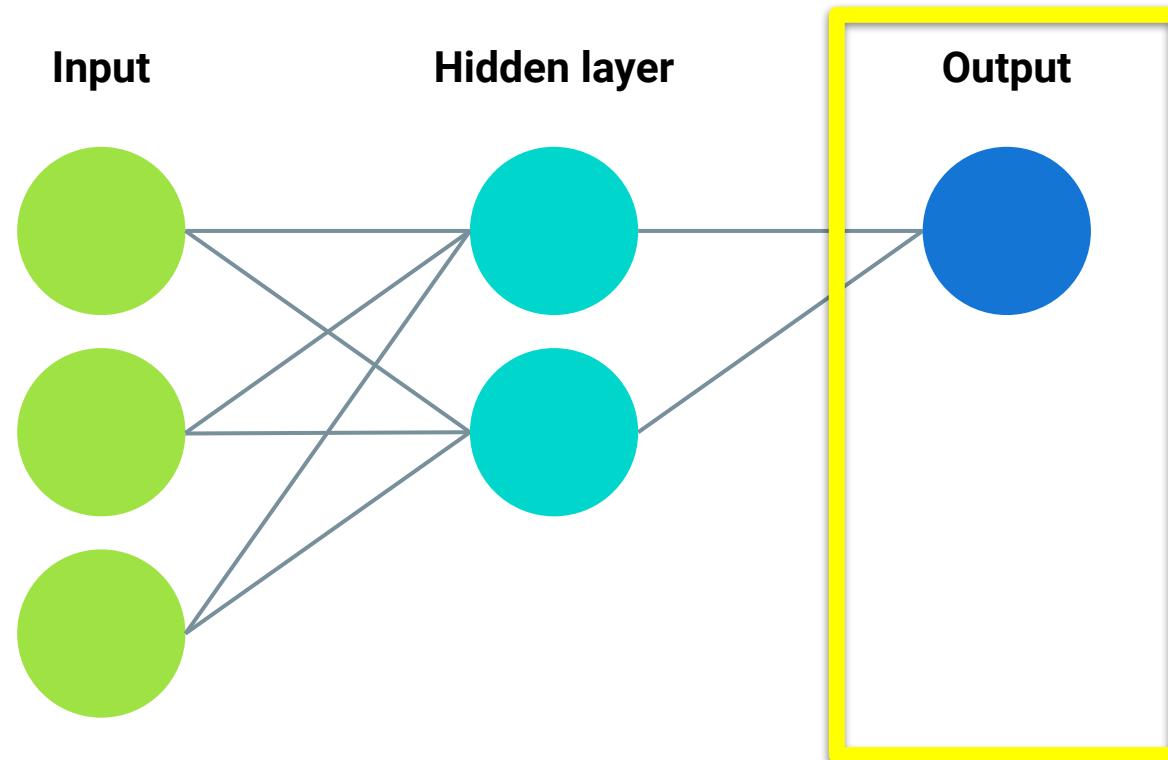
A single **hidden layer** that can contain a single neuron or multiple neurons



# The Structure of a Neural Network

---

An **output layer** that reports the outcome of the value



# Activation Functions

# Activation Functions

---

Neural networks link neurons that process input together to produce a clear, quantitative output.

An **activation function** combines all these outputs into a single classifier or regression model.



When building a model, we apply the activation function to the end of each neuron (each individual perceptron model).

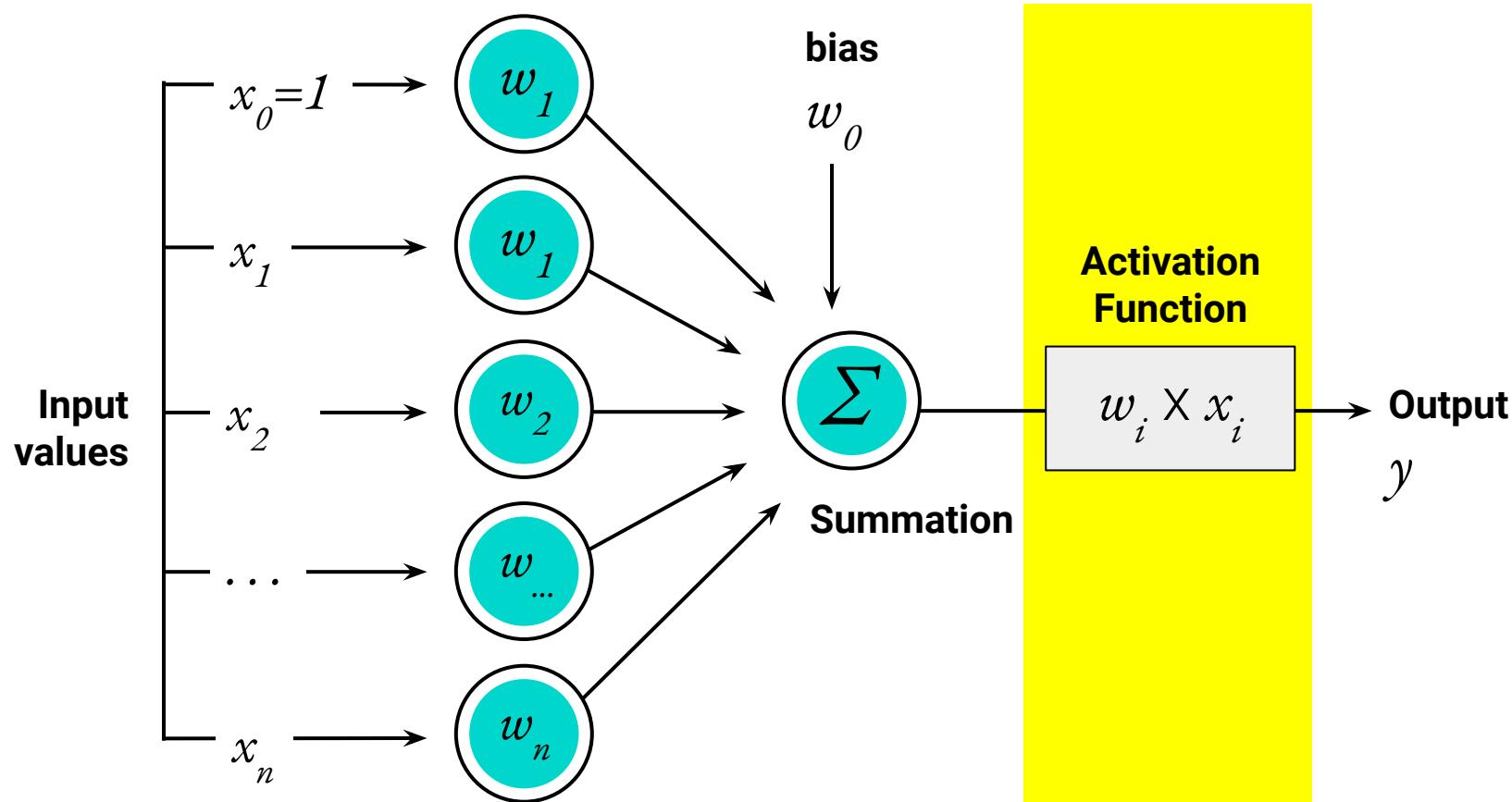


This mathematical function transforms each neuron's output into a quantitative value.



The quantitative output value becomes the input value for the next layer in the neural network model.

# Activation Function (Transformation After Summation)

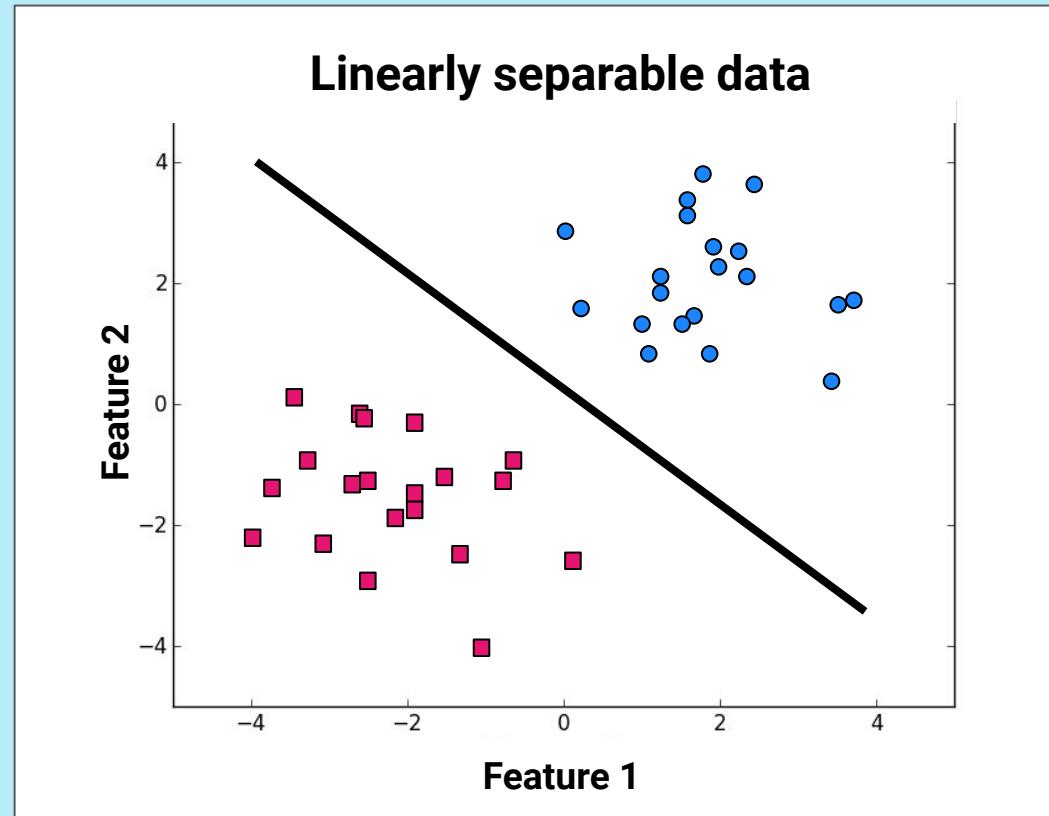




A wide variety of activation functions exist,  
and each has a specific purpose.  
However, most neural networks will use  
one of the following activation functions.

# Linear Function

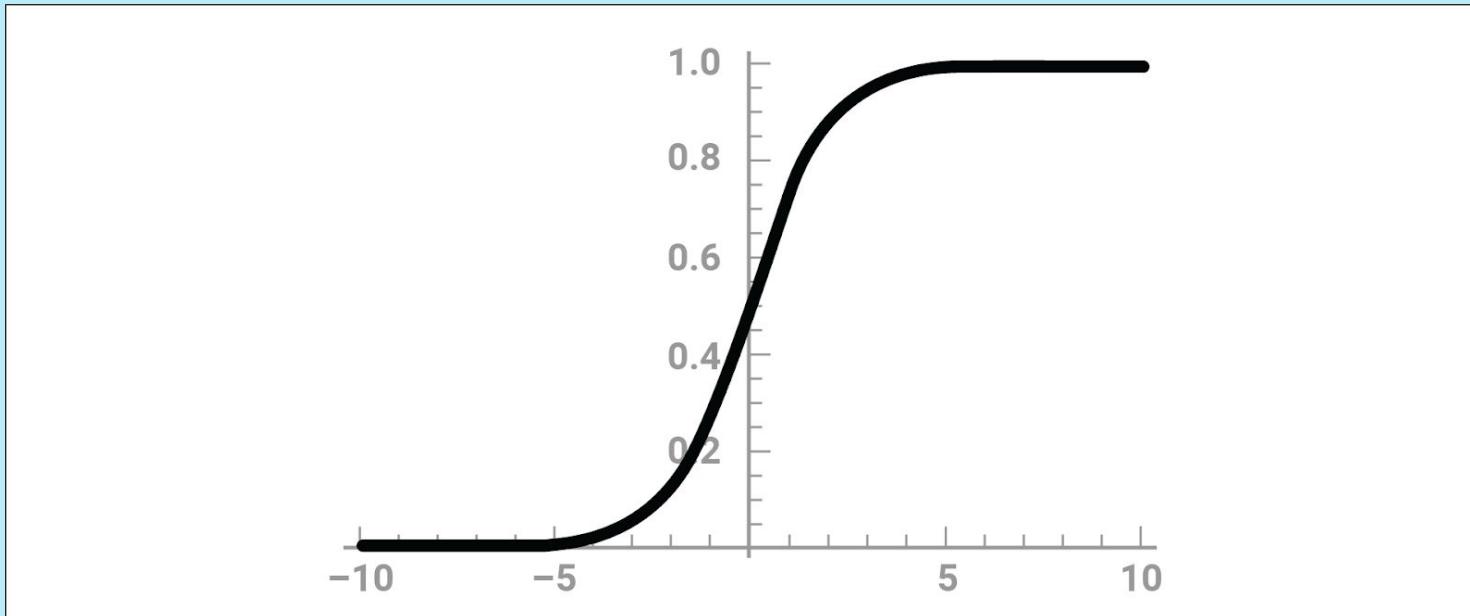
The linear function returns the sum of the weighted inputs without transformation.



# Sigmoid Function

---

The sigmoid function transforms the neuron's output to a range between 0 and 1, which is especially useful for predicting probabilities. A neural network that uses the sigmoid function will output a model with a characteristic S-curve.



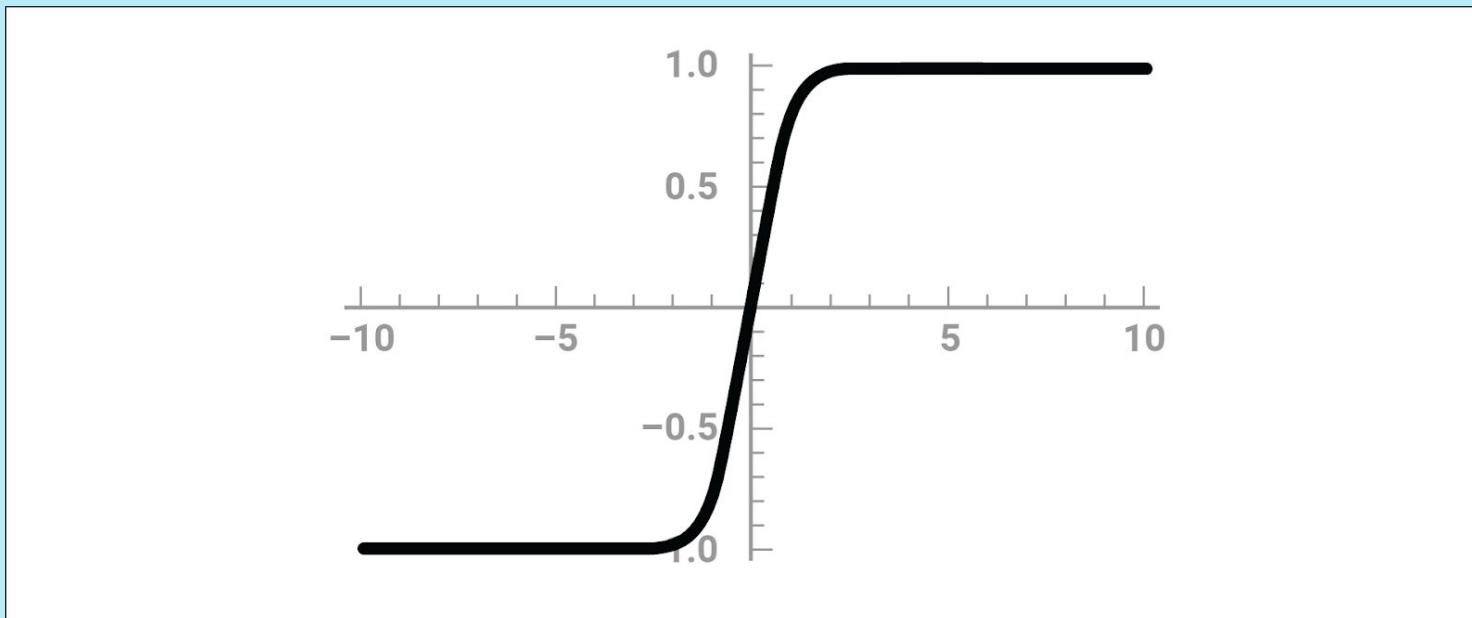
# Tanh Function

---

The tanh function transforms the output to a range between  $-1$  and  $1$ .

The output for a model using a tanh function also forms a characteristic S-curve.

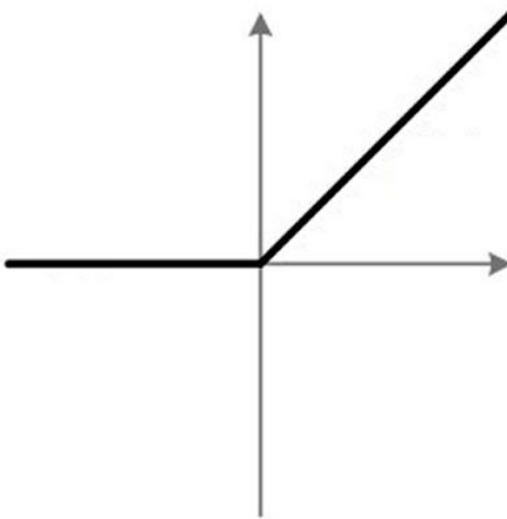
Its primary use is classifying data into one of two classes.



# Rectified Linear Unit (ReLU)

---

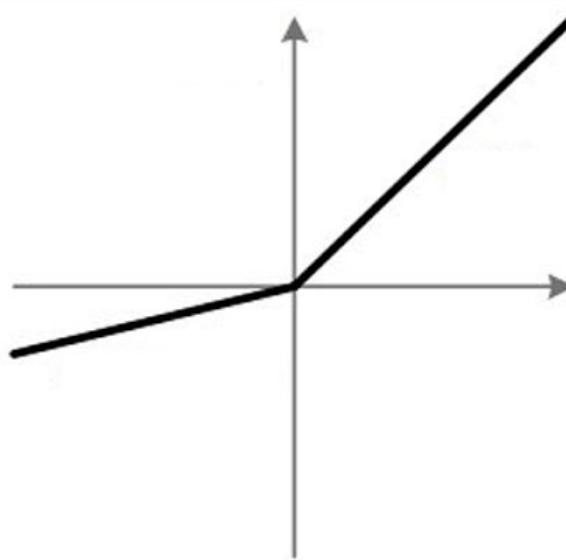
The rectified linear unit (ReLU) function returns a value from 0 to infinity. This activation function transforms any negative input to 0. It is the most commonly used activation function in neural networks due to its faster learning and simplified output. However, it is not always appropriate for simpler models.



# Leaky ReLU Function

---

The leaky ReLU function is an alternative to the ReLU function, which can sometimes work better. Instead of transforming negative input values to 0, it transforms them into much smaller negative values.



# Deep Dive

When designing a neural network, developers usually test different activation function combinations. You can learn more about the formulas and the mathematical fundamentals behind each function from:

The screenshot shows the Wikipedia page for "Activation function". The page header includes links for Article, Talk, Read, Edit, View history, and Search Wikipedia. Below the header, the text discusses the formalism used to approximate the influence of an extracellular electrical field on neurons. It mentions that for a linear system's transfer function, see transfer function. The page then describes how each neuron forms a weighted sum of its inputs and passes the resulting scalar value through an activation function or transfer function. A formula is given:  $a = g(w_1x_1 + w_2x_2 + \dots + w_nx_n + b)$ . The function  $g$  is referred to as the activation function. If  $g(z) = z$ , the neuron performs linear regression or classification. If  $g$  is taken to be a nonlinear function to do nonlinear regression and solve classification problems that are not linearly separable, the output value of the neuron can be interpreted as a YES/NO answer or binary decision. A graph titled "Logistic activation function" is shown, plotting the sigmoid curve  $y = \frac{1}{1+e^{-x}}$  against  $x$ . The page also includes a sidebar with navigation links like Main page, Contents, and a "Contents [hide]" section listing various sub-sections of the activation function.

[Wikipedia article](#)

The screenshot shows the "Activation Functions" page from the ML Glossary. The top navigation bar includes links for Docs, Activation Functions, and an "Edit on GitHub" button. The main content area has a sidebar with links to Linear, ELU, ReLU, LeakyReLU, Sigmoid, Tanh, Softmax, Layers, Loss Functions, Optimizers, Regularization, Architectures, ALGORITHMS (TODO), Classification, Clustering, Regression, Reinforcement Learning, and RESOURCES. The main content area is titled "Activation Functions" and lists the following activation functions: Linear, ELU, ReLU, LeakyReLU, Sigmoid, Tanh, and Softmax. Below this, a section titled "Linear" is shown with a graph of the linear function  $R(z, m) = \{z * m\}$  and its derivative  $R'(z, m) = \{m\}$ . The graph shows a straight line passing through the origin with a slope of  $m$ .

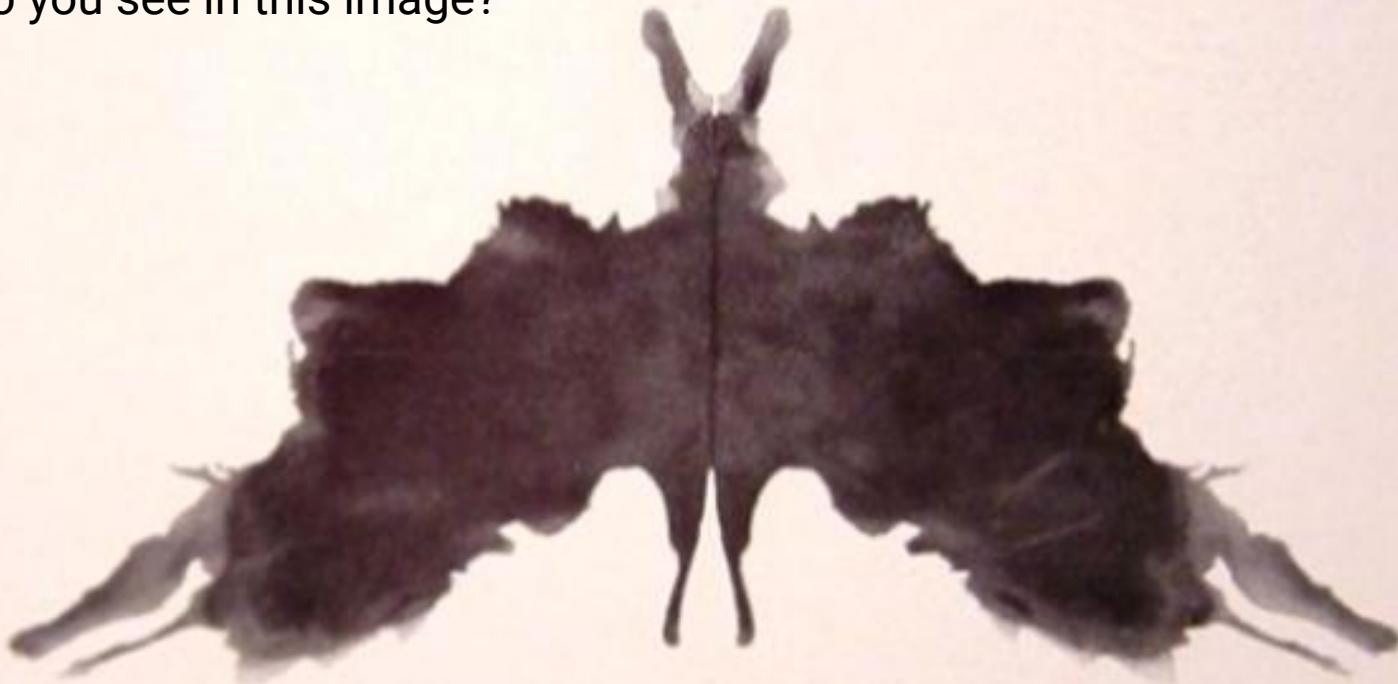
[ML Glossary's Activation Function](#)

# Neural Networks Are Cool!

# Neural Networks

---

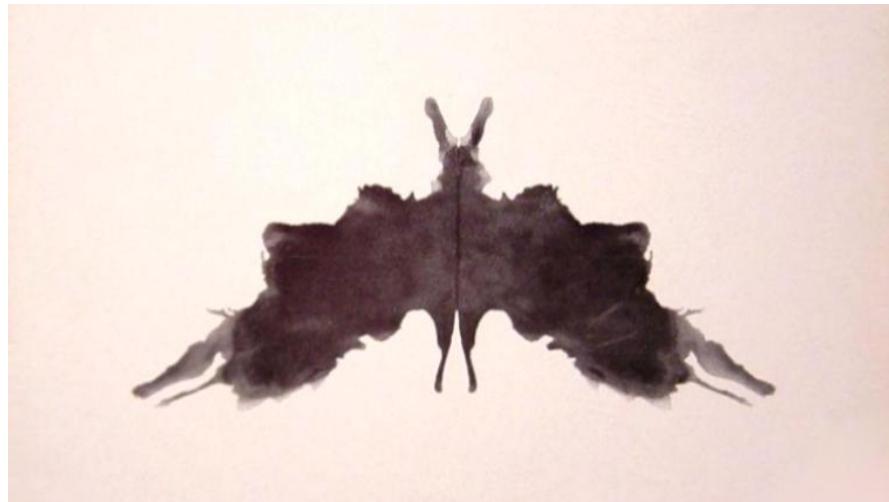
What do you see in this image?



# Neural Networks

---

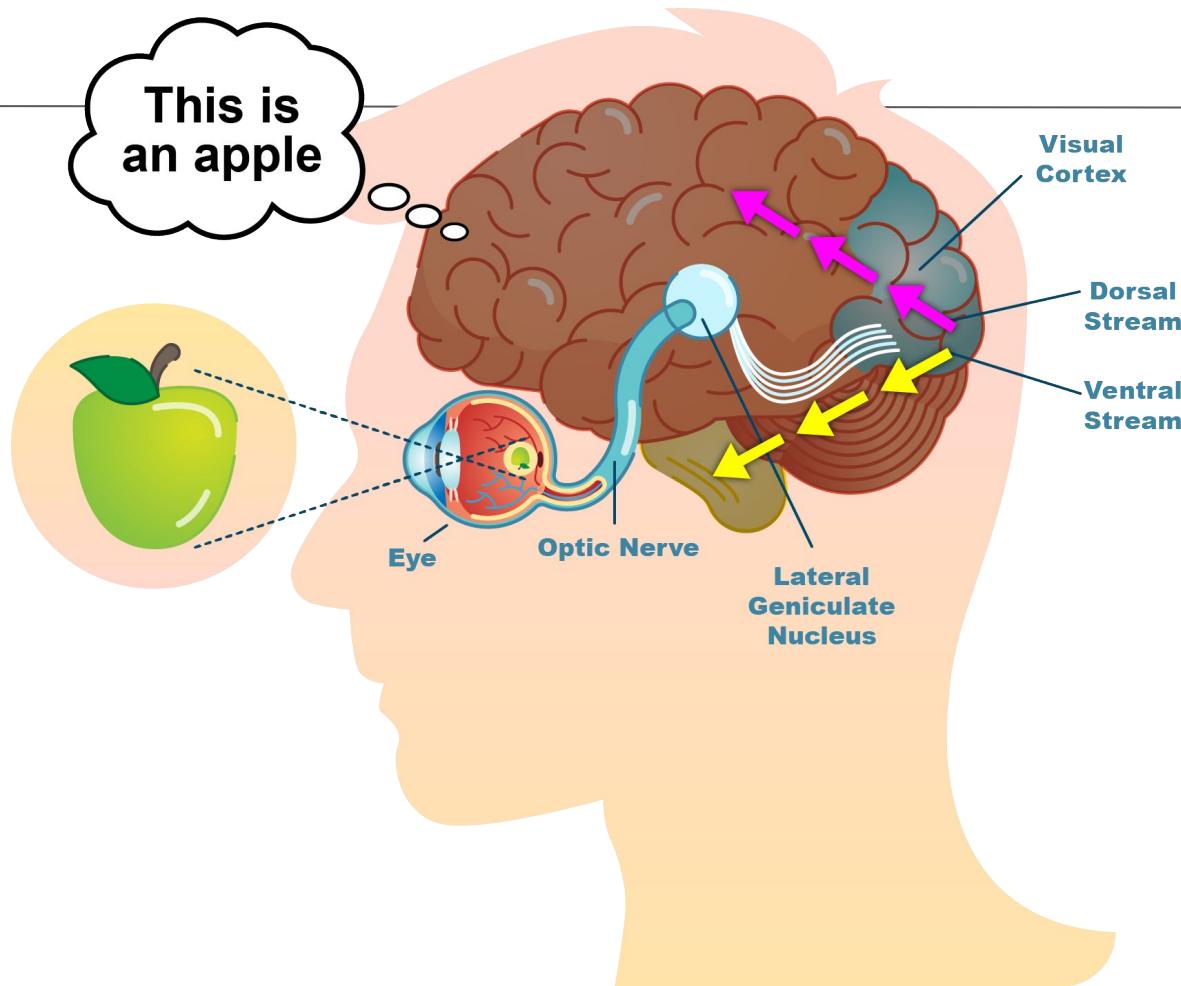
We can find similarities between this image and a real object because our brain uses thousands of neuron connections to find a match between the visual input and a mental representation of an object.



# Neural Networks

How our brain works:

In order to recognize an image, our brain uses thousands of neuron connections to find a match between the visual input and a mental representation of an object.

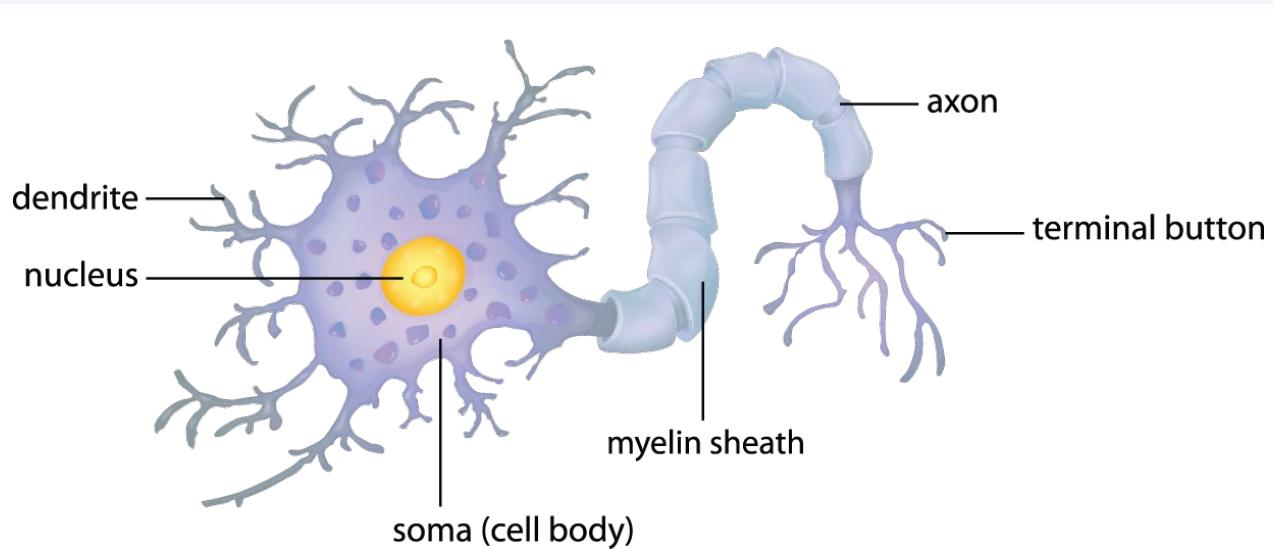


# Neural Networks

The ability of the brain to process information and make predictions or interpretations is what inspired neurophysiologists and mathematicians to start the development of artificial neural networks (ANN).

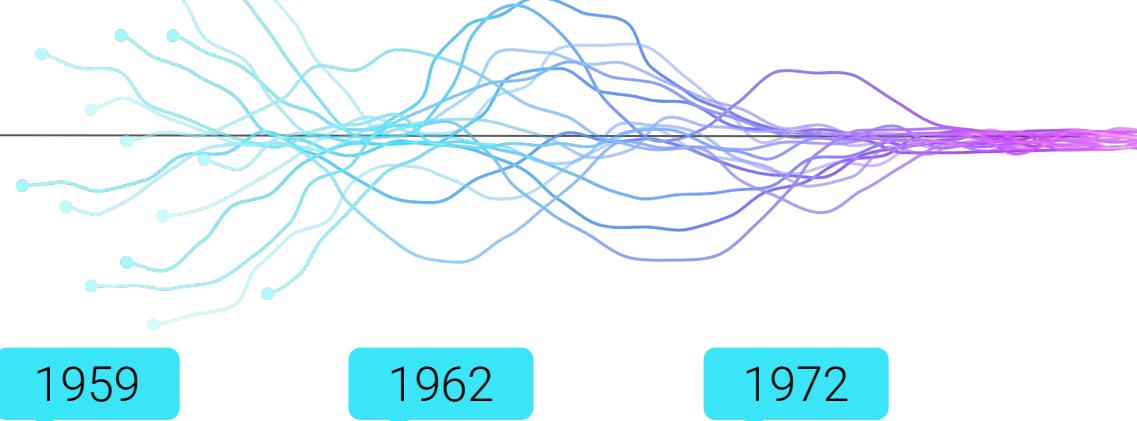
In the same way that biological neurons receive input signals through the dendrites, an ANN receives input variables and processes them by using an activation function.

The output of an ANN is similar to the neuron nucleus in the brain.



# Neural Networks

## The History of Neural Networks:



1943

Neurophysiologist **Warren McCulloch** and mathematician **Walter Pitts** wrote a paper on how neurons might work.

1949

**Donald Hebb** wrote *The Organization of Behavior*, which pointed out the fact that neural pathways are strengthened each time they are used.

1959

**Bernard Widrow** and **Marcian Hoff** of Stanford developed models called ADALINE and MADALINE.

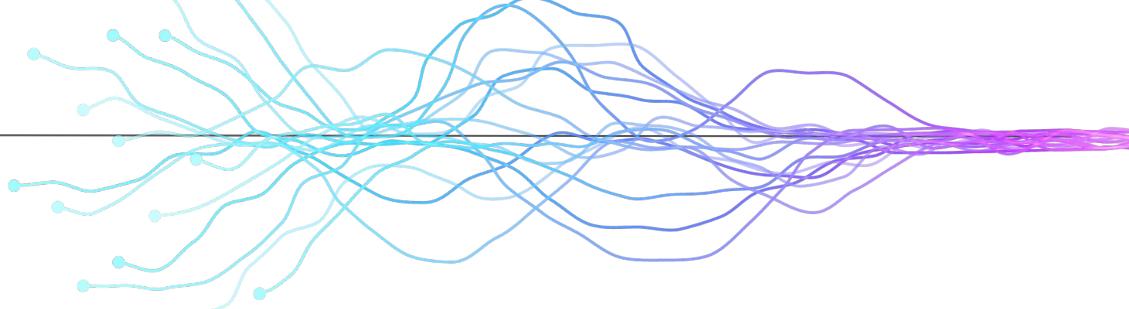
1962

**Widrow** and **Hoff** developed a learning procedure that examines the value before the weight adjusts it (i.e., 0 or 1) according to the rule: Weight Change = (Pre-Weight line value).

1972

**Teuvo Kohonen** and **James A. Anderson** each developed a similar network independently of one another. They both used matrix mathematics to describe their ideas but did not realize that what they were doing was creating an array of analog ADALINE circuits.

# Neural Networks



## The History of Neural Networks: Timeline

1982

**John Hopfield** of Caltech presented a paper to the National Academy of Sciences. His approach was to create more useful machines by using bidirectional lines. Previously, the connections between neurons was only one way.

1982

Joint US-Japan conference on **Cooperative/Competitive Neural Networks**. Japan announced a new Fifth Generation effort on neural networks, and US papers generated worry that the US could be left behind in the field.

1986

Three independent groups of researchers, including **David Rumelhart**, a former member of Stanford's psychology department, came up with similar ideas which are now called back propagation networks.

1997

A recurrent neural network framework, LSTM was proposed by **Jürgen Schmidhuber** and **Sepp Hochreiter**.

Now

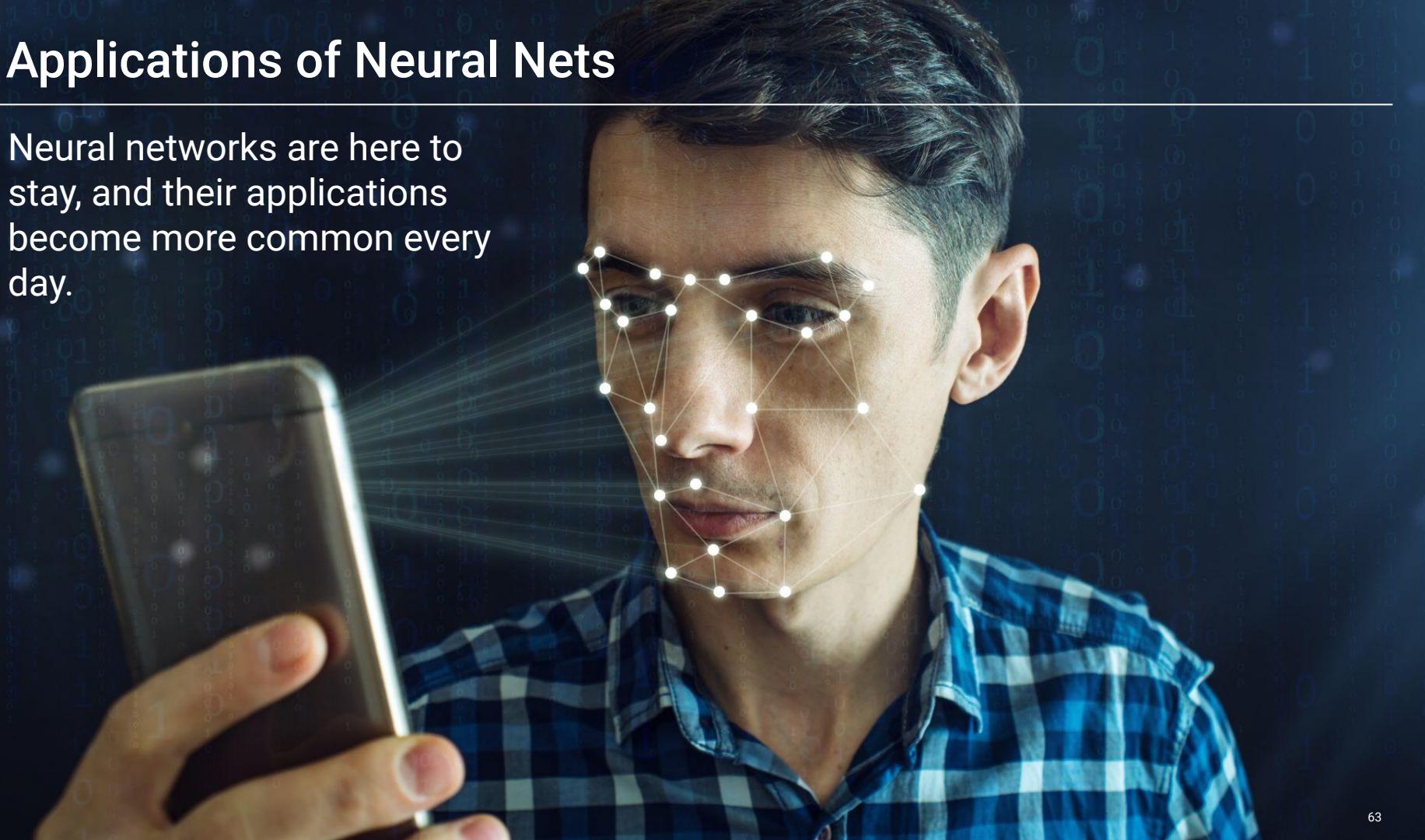
Neural networks discussions are prevalent; the future is here!

**Examples:**  
self driving cars,  
algorithmic trading,  
robo advisors,  
customized marketing offers.

# Applications of Neural Nets

---

Neural networks are here to stay, and their applications become more common every day.



# Applications of Neural Nets

---

Companies like Google and Tesla are using neural networks to develop self-driving cars.



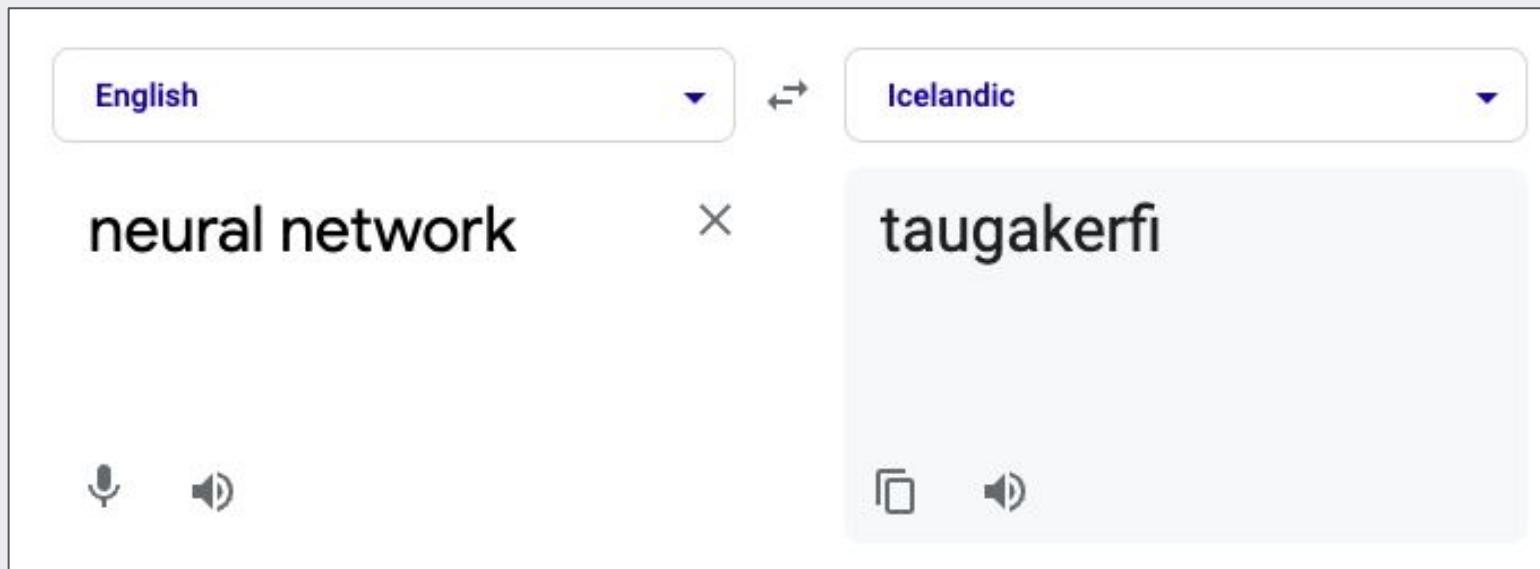
WAYMO



TESLA

# Applications of Neural Nets

The power of neural networks allows you to communicate in any language thanks to automated tools that can perform real-time translation (Google Translate, Skype, etc.).



# Applications of Neural Nets

---



Black and white photos can gain a new context through automatic image colorization.

# Applications of Neural Nets

---

Neural networks are also used to help create a better world, not only for humans but for wildlife.

**For example:**

The U.S. National Oceanic and Atmospheric Administration is helping to save Right whales by tracking their North Atlantic population, using neural networks for image recognition.



Let's play a short game powered by neural networks to illustrate how clever such models can be.

The image shows a screenshot of the Quick, Draw! game. At the top center is a banner with the text "QUICK, DRAW!" in a stylized font. Below the banner is a yellow hand pointing towards a collection of small doodles. The doodles include a bicycle, a hot air balloon, a pizza slice, a coffee cup, a bell pepper, a bomb, and a soccer ball. In the top left corner, there is a purple button with a white question mark icon. In the top right corner, there are icons for Twitter and Facebook. Below the main title, the text "Can a neural network learn to recognize doodling?" is displayed. Underneath that, instructions say "Help teach it by adding your drawings to the [world's largest doodling data set](#), shared publicly to help with machine learning research." A large yellow button labeled "Let's Draw!" is centered at the bottom. The footer contains links for "This is an A.I. Experiment" (with Google logo), "Made with some friends from Google", "Privacy & Terms", and a language selection dropdown set to "English".

Can a neural network learn to recognize doodling?

Help teach it by adding your drawings to the [world's largest doodling data set](#), shared publicly to help with machine learning research.

Let's Draw!

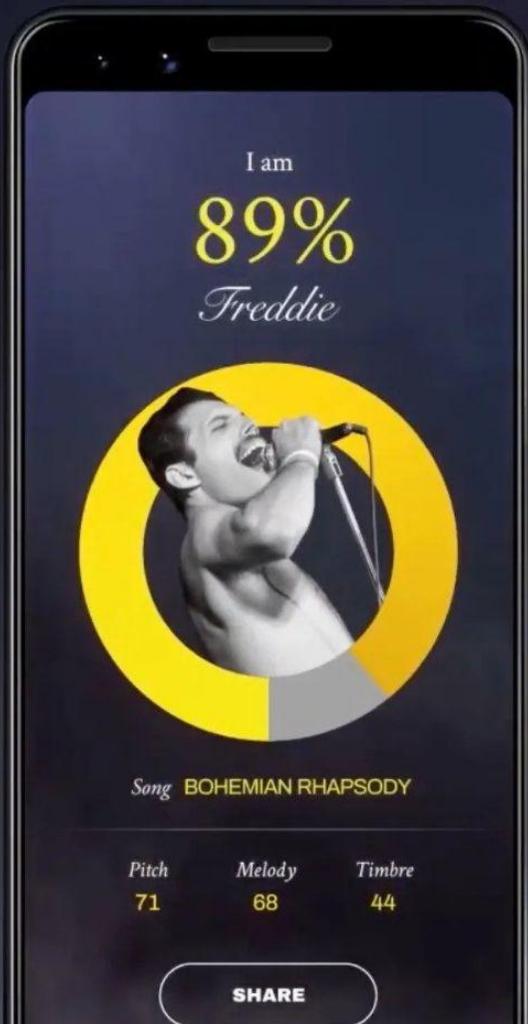
This is an A.I. Experiment

Made with some friends from Google

Privacy & Terms

English

Let's visit the  
[Google AI](#)  
[Experiments website](#)  
and select an  
experiment to try.



# Questions?



# Creating a Neural Network

# Creating a Neural Network

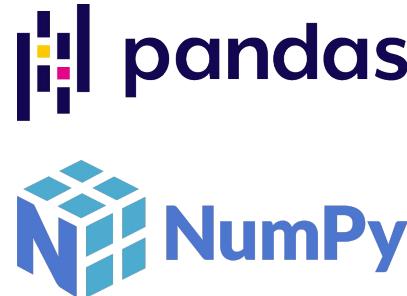
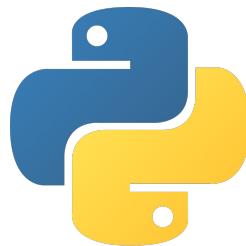
---

There are two ways to code a neuron:

01

The hard way (code from scratch)

You can do all the math and code it from scratch using Python, Pandas, and NumPy.



02

The easy way (API or framework)

You can use an industry-standard API or framework to speed up your implementation and focus your efforts on improving your model.

This allows you to spend more time learning about the business problem you want to solve.

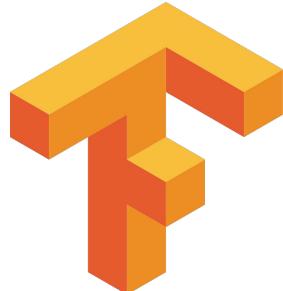
# Creating a Neural Network

---

We are going to use [TensorFlow](#) and [Keras](#) to build our neural networks.

## TensorFlow

TensorFlow is an open-source platform for machine learning that allows us to run our code across multiple platforms in a highly efficient way.



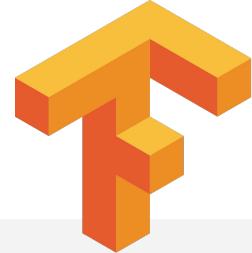
## Keras

Keras is an abstraction layer on top of TensorFlow that makes it easier to build models. You can relate this to using Plotly Express to create charts instead of using the more verbose Matplotlib library.



# Creating a Neural Network

---



## Why TensorFlow?



Machine learning library



Used to build neural networks



Runs on multiple platforms



Supports distributed computing



Allows detailed fine-tuning

# Creating a Neural Network



## Why Keras?



An API designed for human beings



Runs on top of TensorFlow, CNTK, or Theano



Facilitates to turn models into products



Has broad adoption in the industry and the research community

# Keras + TensorFlow

---

Keras allows interaction with TensorFlow through a simplified interface  
Model → Fit → Predict (with a few other steps)

```
from keras.models import Sequential  
  
model = Sequential()
```

```
# x_train and y_train are Numpy arrays, just like in the Scikit-Learn API.  
model.fit(x_train, y_train, epochs=5, batch_size=32)
```

```
classes = model.predict(x_test, batch_size=128)
```

# Keras + TensorFlow

---

There are two types of models in Keras.

01

Sequential model

The **sequential** model is a linear stack of layers in which data flows from one layer to the next.

02

Functional model

The functional **model** class allows us to create sophisticated and more customizable models.



We'll use the **sequential** model with the **Dense** class to add layers to the neural network.



## Instructor Demonstration

---

# Creating a Neural Network in Keras

# Questions?





# Activity: Preventing Credit Card Defaults, Part 1

In this activity, you will use Keras and several features (variables) to build a neural network model that predicts whether a credit card customer will default on their debt.

Suggested Time:

---

20 minutes



Time's Up! Let's Review.

# Questions?



Countdown timer

15:00

(with alarm)

Break



# Building, Fitting, and Predicting a Neural Network Model

Defining the neural network's structure is like designing blueprints for a house, and compiling the model is like building the house.





To compile and fit a model,  
we need to specify **loss** and  
**optimization** functions.

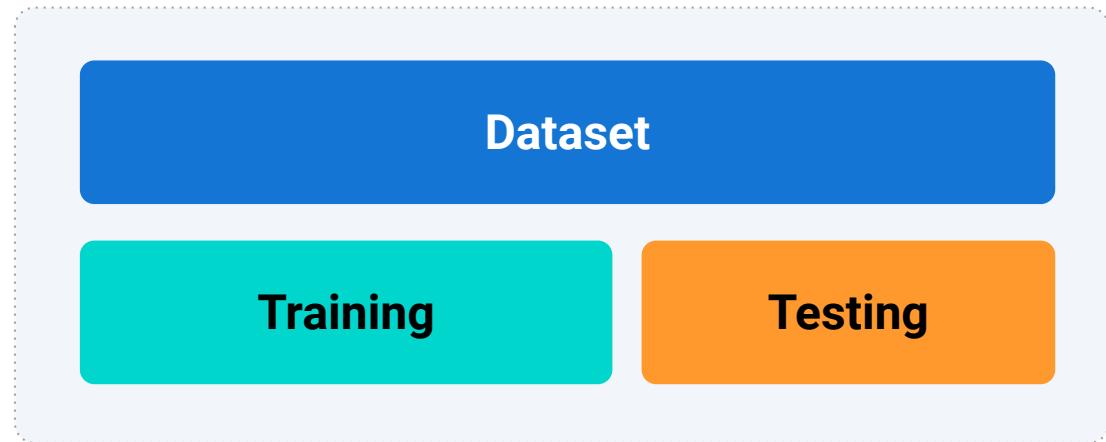
# Building, Fitting, and Predicting a Neural Network Model

---

When we train our neural network model on a dataset, we pass our training dataset through the model multiple times.

The loss function allows us to observe how the model's performance changes over each iteration.

We can then determine when the model reaches maximum performance (after a specific number of iterations).



# How a Neural Network Optimizes

---



The optimization function shapes and molds a neural network model while the model is trained on the data.

This ensures that the model performs to the best of its ability.

An optimization function reduces the model's losses and provides the most accurate output possible.

# Building, Fitting, and Predicting a Neural Network Model

When we fit the model, we also want to keep track of two evaluation metrics:

## Accuracy

- We use model predictive accuracy (`accuracy`) for classification models that are designed to answer “whether or not” questions.
- A higher accuracy value indicates more accurate predictions, and the highest possible accuracy value is 1.
- We would use a classification model to predict if a borrower is likely to default on a loan.

## Mean squared error (MSE)

- We use MSE (`mse`) for regression models in which we're trying to predict some continuous outcome, like the percentage stock return for the next day.
- As the MSE gets closer to 0, the model's predictions become increasingly accurate.

# Mean Squared Error (MSE)

---

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$$

MSE = mean squared error

$Y_i$  = observed values

$n$  = number of data points

$\hat{Y}_i$  = predicted values



## Instructor Demonstration

---

Building, Fitting, and Predicting  
a Neural Network Model

# Questions?





## Instructor Demonstration

---

Review Preventing Credit Card Defaults,  
Part 2 Solutions

# Questions?





# Activity: Preventing Credit Card Defaults, Part 2

In this activity, you will continue working on the neural network model you created in the previous activity. You'll compile, fit, and evaluate a model that predicts credit card defaults.

Suggested Time:

---

20 minutes



Time's Up! Let's Review.

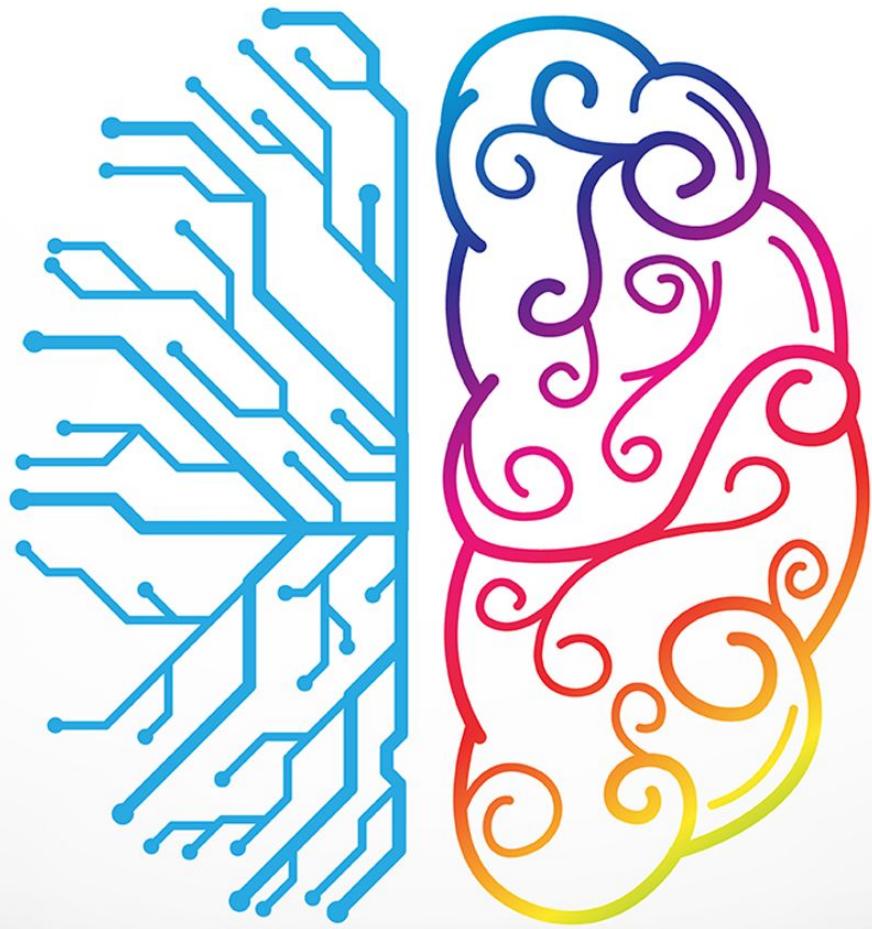
# Questions?





Modeling neural networks  
is part art and part science.

Be patient while defining models.

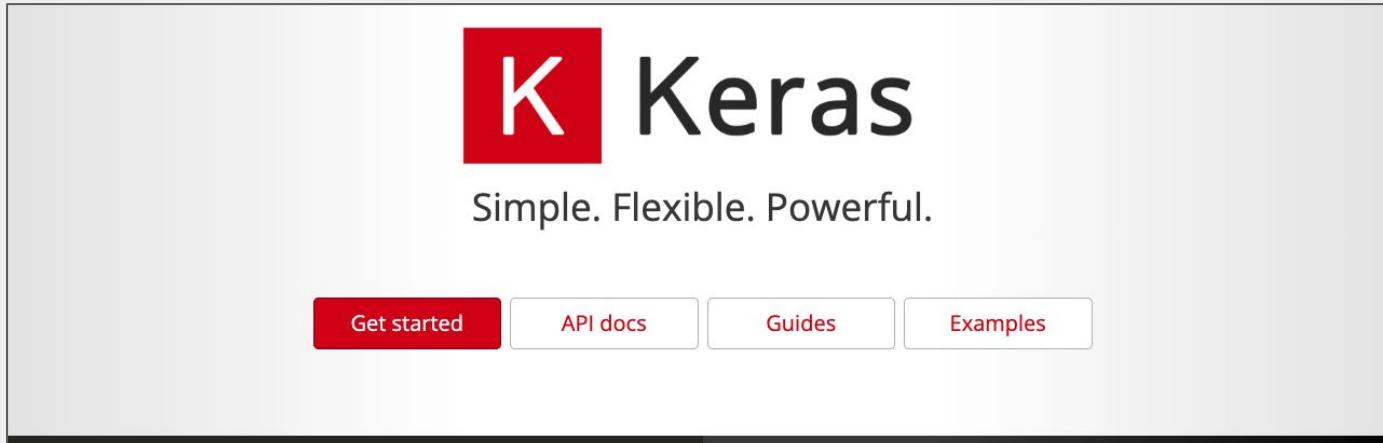




There are several frameworks  
for implementing neural networks.

# Recap

Keras is a business-class framework that you can use for prototyping or deploying production models.



```
from tensorflow import keras
from tensorflow.keras import layers

# Instantiate a trained vision model
vision_model = keras.applications.ResNet50()

# This is our video encoding branch using the trained vision_model
video_input = keras.Input(shape=(100, None, None, 3))
encoded_frame_sequence = layers.TimeDistributed(vision_model)(video_input)
encoded_video = layers.LSTM(256)(encoded_frame_sequence)

# This is our text-processing branch for the question input
question_input = keras.Input(shape=(100,), dtype='int32')
embedded_question = layers.Embedding(10000, 256)(question_input)
encoded_question = layers.LSTM(256)(embedded_question)

# And this is our video question answering model
merged = keras.layers.concatenate([encoded_video, encoded_question])
output = keras.layers.Dense(1000, activation='softmax')(merged)
video_qa_model = keras.Model(inputs=[video_input, question_input],
                             outputs=output)
```

## Deep learning for humans.

Keras is an API designed for human beings, not machines. Keras follows best practices for reducing cognitive load: it offers consistent & simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear & actionable error messages. It also has extensive documentation and developer guides.



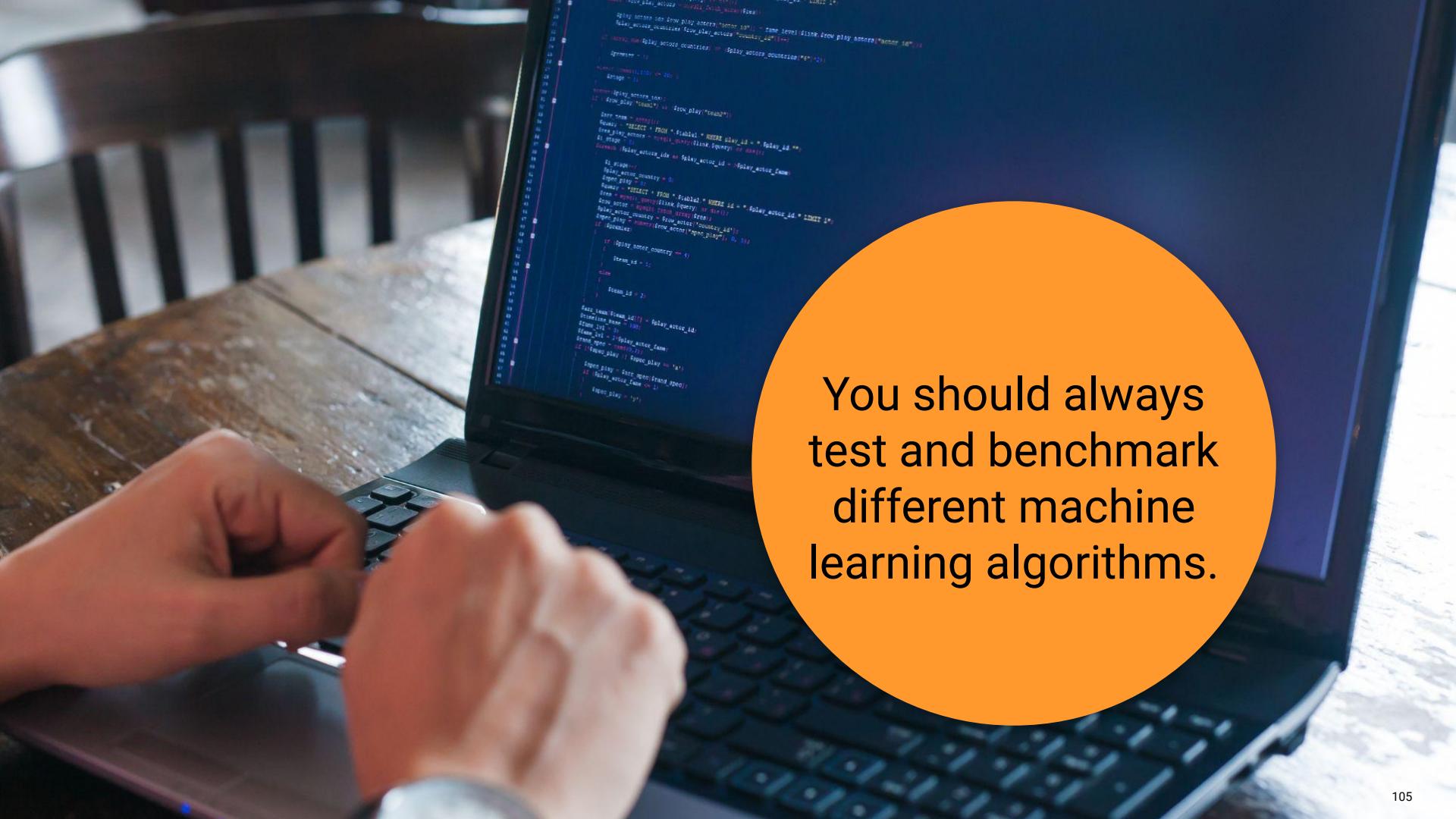
Mastering neural networks  
takes time.

Thanks to frameworks like Keras, you don't need a PhD to start using neural networks to solve real-world problems.





Neural networks are not a  
universal best solution.

A photograph of a person's hands typing on a laptop keyboard. The laptop screen displays a large amount of code, likely Python, with syntax highlighting. An orange circle is overlaid on the right side of the image, containing the text.

You should always  
test and benchmark  
different machine  
learning algorithms.

Congratulations! You've learned new skills that will add value to your professional portfolio.



# Questions?



*The  
End*