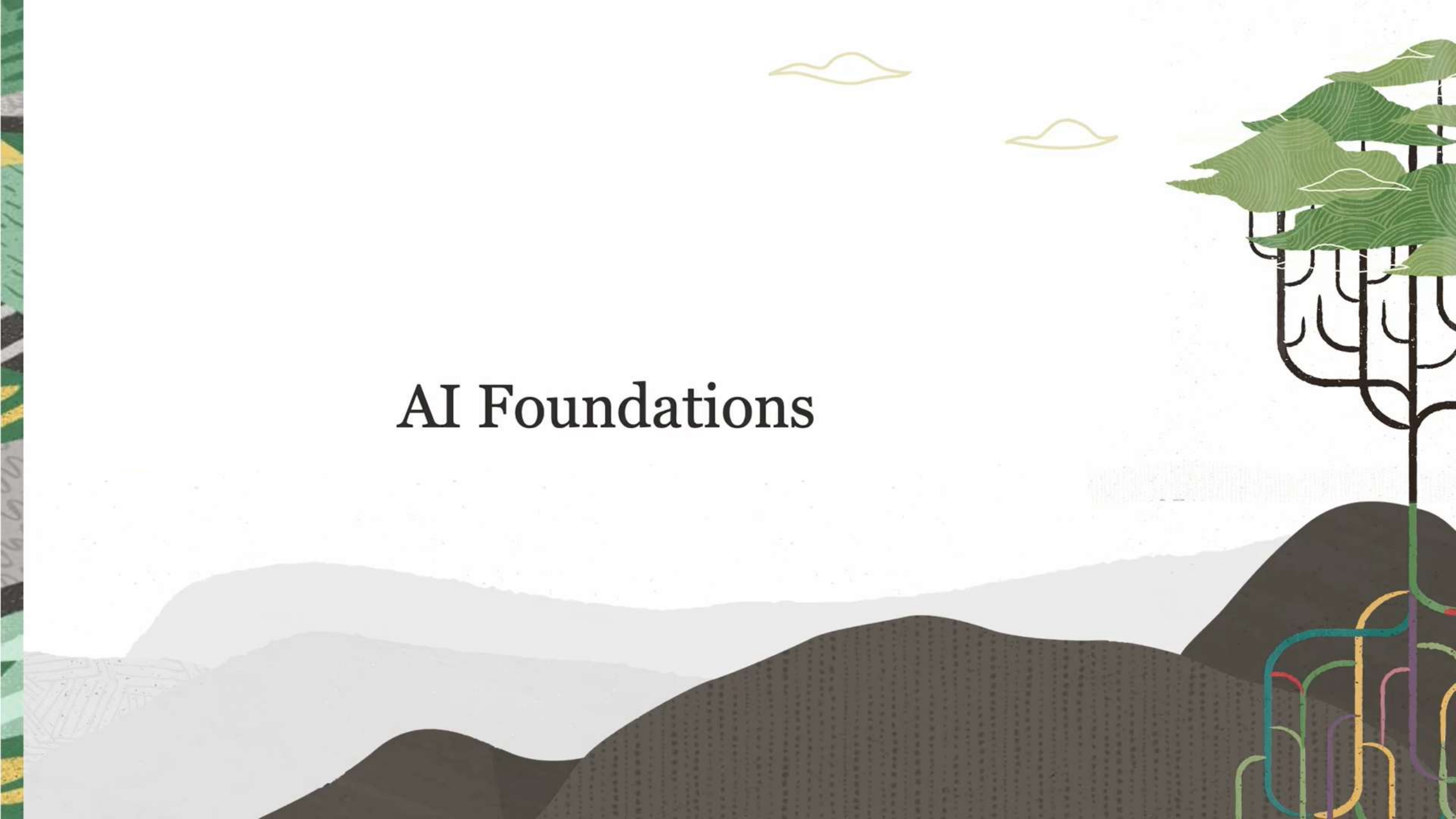
AI Foundations



1. Welcome to AI Foundations
Welcome to AI Foundations

2. AI Foundations
Introduction to AI
AI - Tasks and Data
AI vs ML vs DL

3. Machine Learning Foundations
Introduction to Machine Learning
Supervised Learning - Regression
Supervised Learning - Classification
Unsupervised Learning
Reinforcement Learning

4. Deep Learning Foundations
Introduction to Deep Learning
Deep Learning Models-Sequence Models
Deep Learning Models - CNN

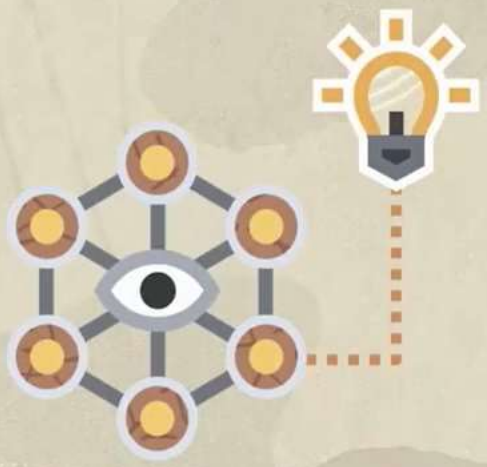
5. Generative AI and LLM Foundations
Introduction to Generative AI
Introduction to Large Language Models
Transformers Part-1
Transformers Part-2
Prompt Engineering
Customize LLMs with your data

6. OCI AI Portfolio
AI Services Overview
ML Services Overview
AI Infrastructure
GPUs and Superclusters in OCI
Responsible AI

7. OCI Generative AI Service
OCI Generative AI
Vector Search
Select AI

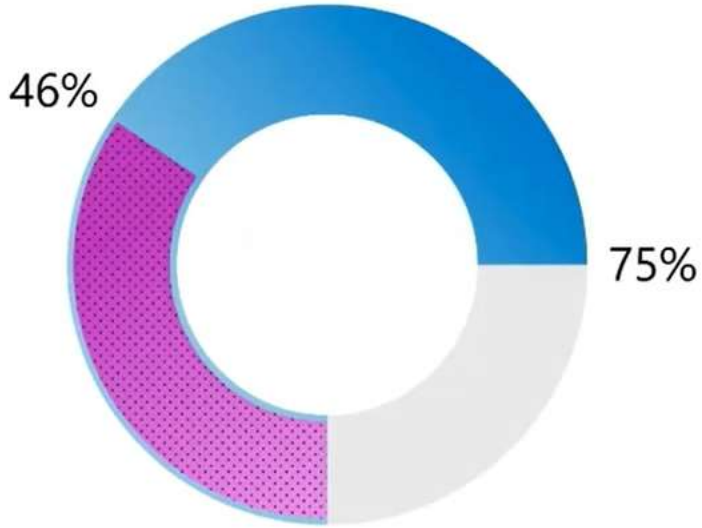
8. OCI AI Services
Language Intro
Speech Intro
Vision Intro
Document Understanding

Employees want AI at work



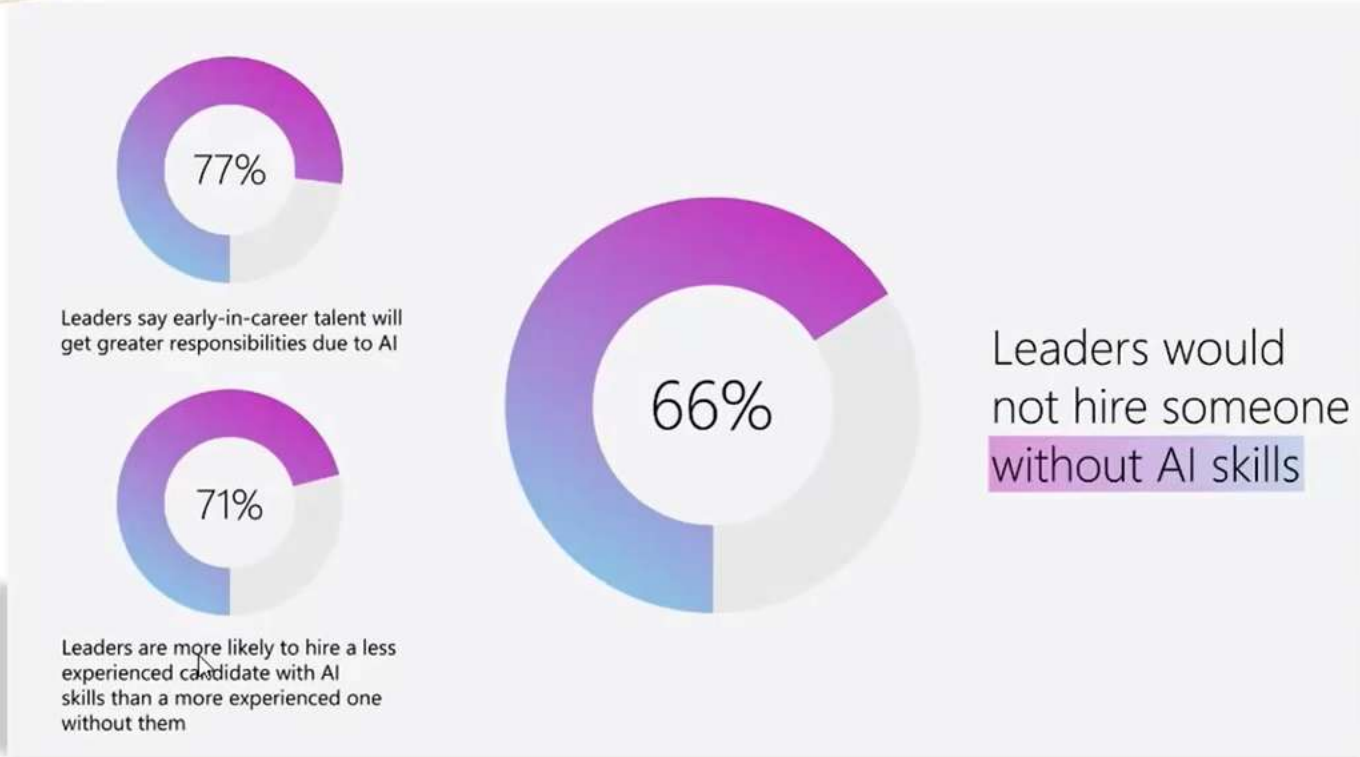
75% of people are
already using AI at work

46% of them started using
it less than 6 months ago



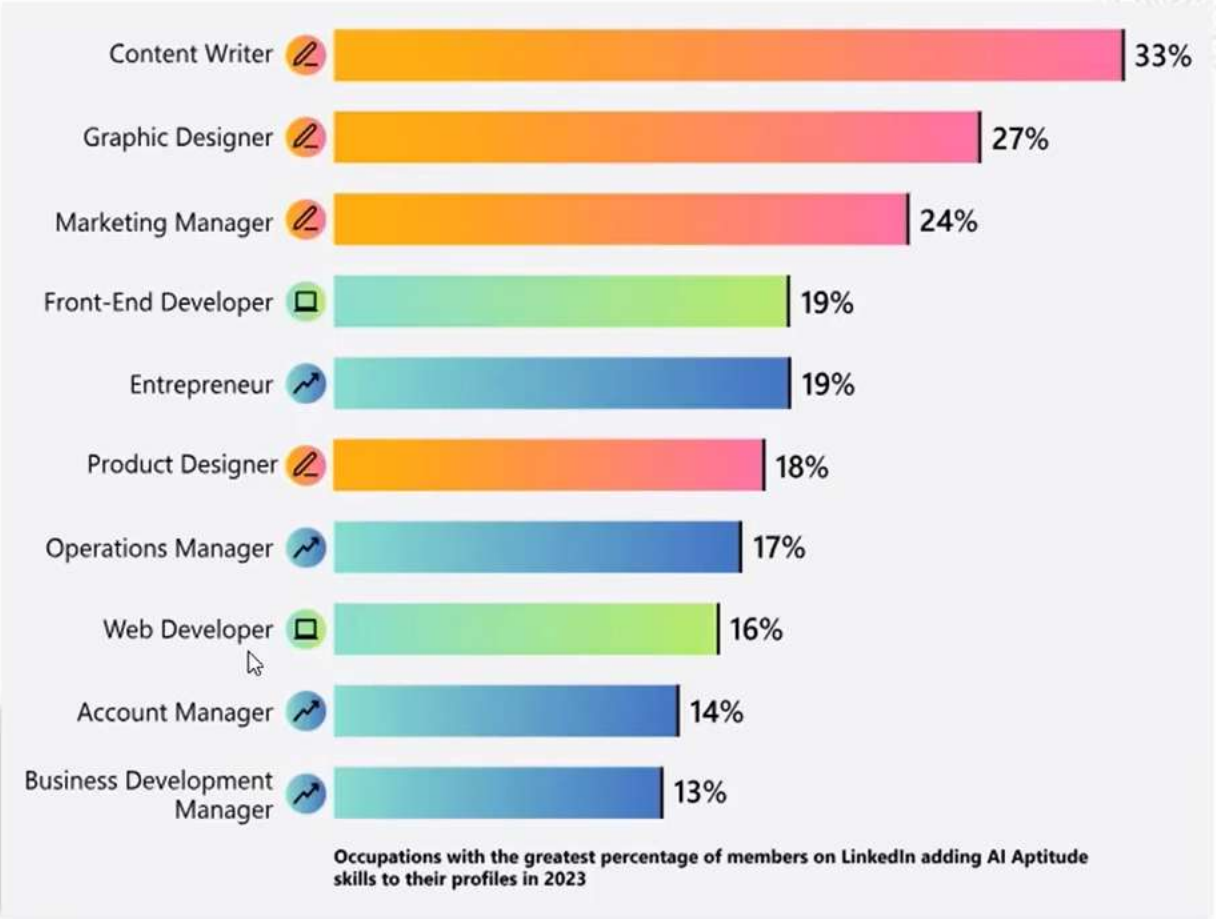
Source: 2024 Work Trend Index Annual Report,
Microsoft and LinkedIn

AI helps break the career ceiling



Source: 2024 Work Trend Index Annual Report, Microsoft and LinkedIn

AI is going Mainstream



Source: 2024 Work Trend Index Annual Report, Microsoft and LinkedIn

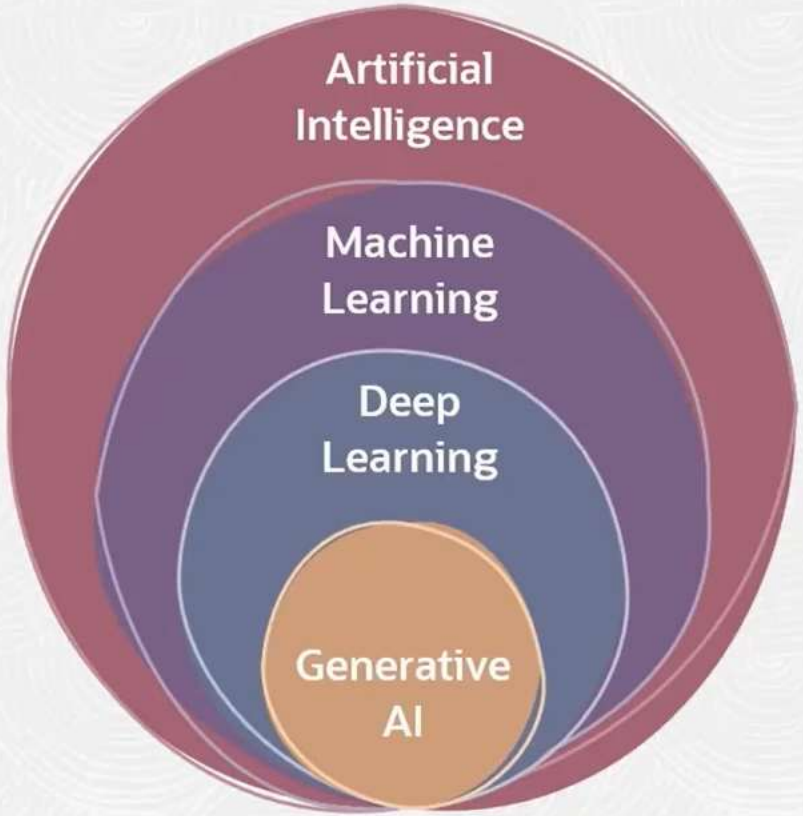
For Whom is this Course Intended?

Cloud Engineers and Architects

Beginners in AI and ML

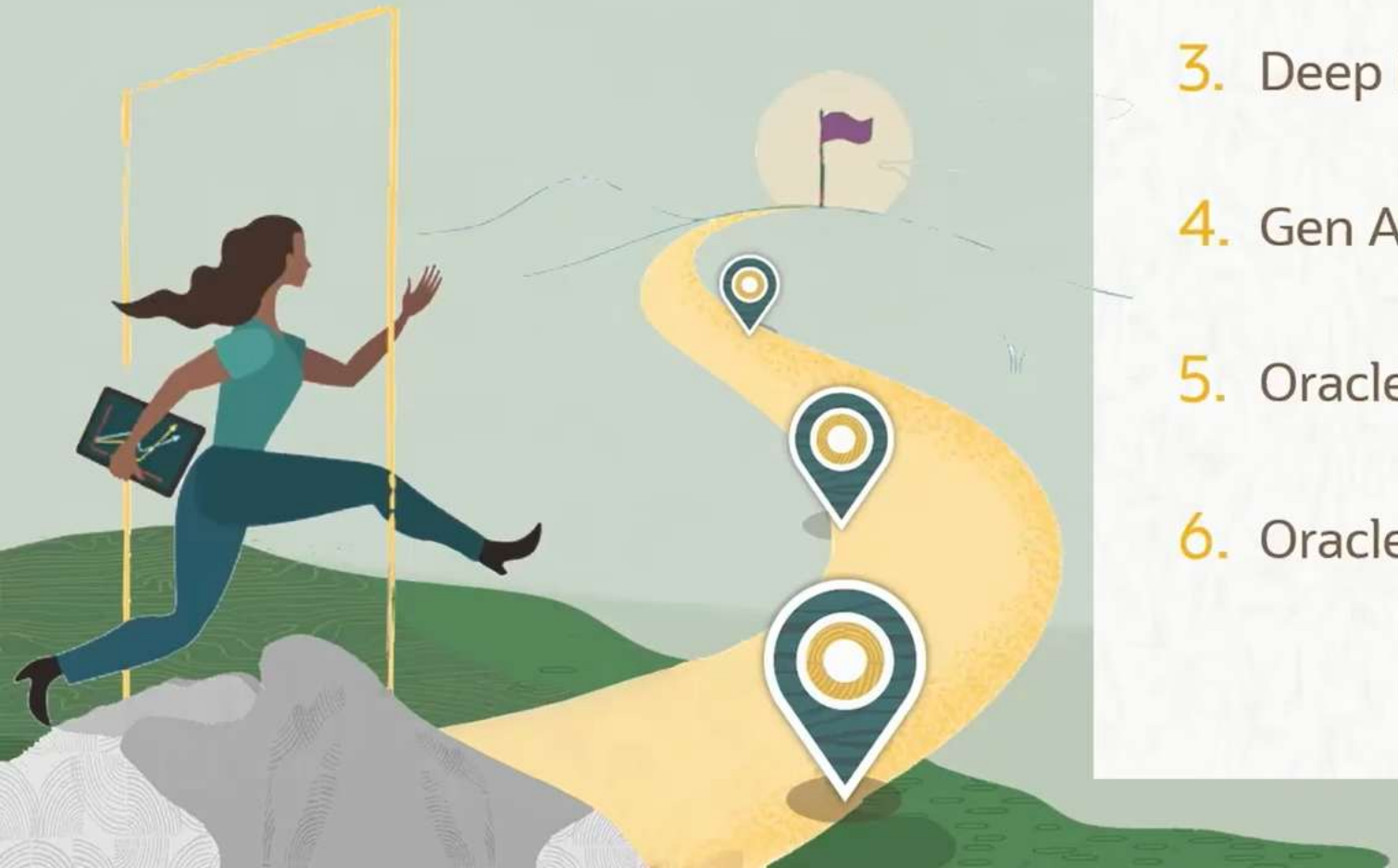
Everyone

Course Outline



AI Stack

Oracle AI Stack



Course Outline

1. AI Foundations
2. Machine Learning Foundations
3. Deep Learning Foundations
4. Gen AI and LLM Foundations
5. Oracle AI Stack
6. Oracle Generative AI Services

1. Welcome to AI Foundations
Welcome to AI Foundations

2. AI Foundations
Introduction to AI
AI - Tasks and Data
AI vs ML vs DL

3. Machine Learning Foundations
Introduction to Machine Learning
Supervised Learning - Regression
Supervised Learning - Classification
Unsupervised Learning
Reinforcement Learning

4. Deep Learning Foundations
Introduction to Deep Learning
Deep Learning Models-Sequence Models
Deep Learning Models - CNN

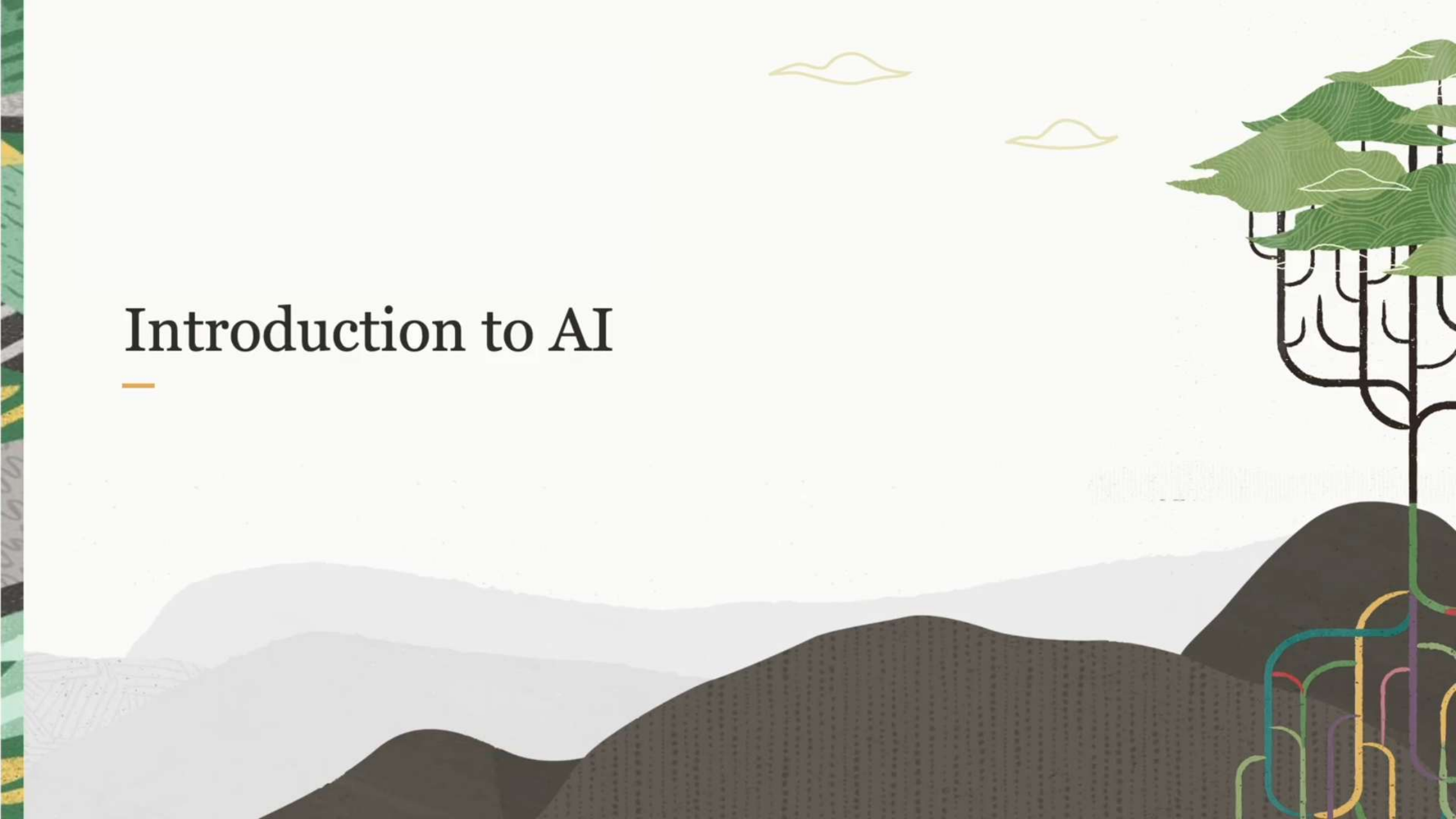
5. Generative AI and LLM Foundations
Introduction to Generative AI
Introduction to Large Language Models
Transformers Part-1
Transformers Part-2
Prompt Engineering
Customize LLMs with your data

6. OCI AI Portfolio
AI Services Overview
ML Services Overview
AI Infrastructure
GPUs and Superclusters in OCI
Responsible AI

7. OCI Generative AI Service
OCI Generative AI
Vector Search
Select AI

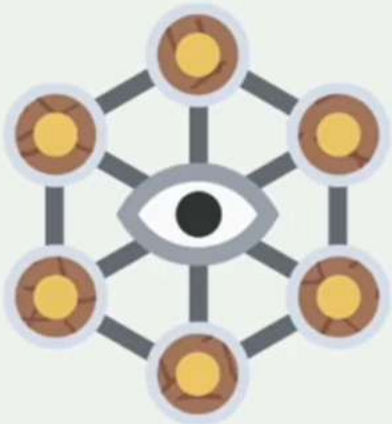
8. OCI AI Services
Language Intro
Speech Intro
Vision Intro
Document Understanding

Introduction to AI



What is Artificial Intelligence?

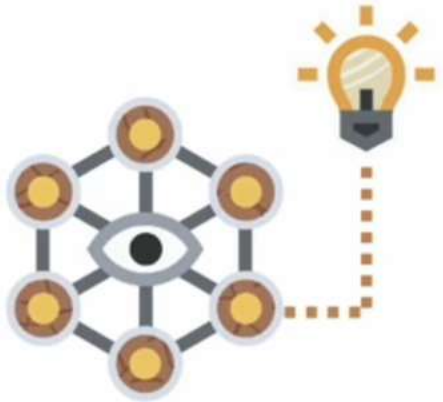
Ability of machines to mimic the cognitive abilities and problem-solving capabilities of human intelligence



Human Intelligence

- Learns new skills through observation
- Thinks abstractly and reasons
- Communicates using a language and non-verbal cues
- Handles complex situations in real time
- Plans short and long term
- Creates art, music, and inventions

If we can replicate any of these capabilities in machines, that is **Artificial General Intelligence (AGI).**

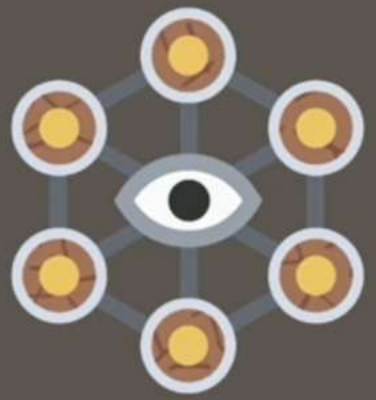


When we apply AGI to solve problems with specific, narrow objectives, we call it **Artificial Intelligence (AI).**

AI Examples

Classifying images

AI is all around us!



Spam mail classification

Writing computer language code

Predicting old car prices

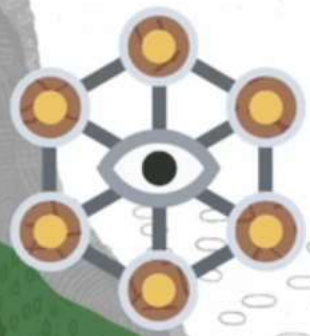
AI Terminology

Machine Learning
(ML)

Deep Learning
(DL)

Data Science
(DS)

Why do we need AI?



Automation and Decision Making

- Approve a credit card or loan.
- Process insurance claims.
- Recommend products to customers.
- Detect fraudulent transactions.
- Classify documents and images.

Creative Support

- Create content.
- Write stories and poems.
- Provide designs.
- Share code.
- Generate ideas.
- Crack jokes.

AI Domains and Examples

Language
Vision
Speech
Product Recommendations
Anomaly Detection
Learn by Reward
Forecasting
Generate Content

AI - Tasks and Data

Commonly Used AI Domains

Language-Related AI Tasks

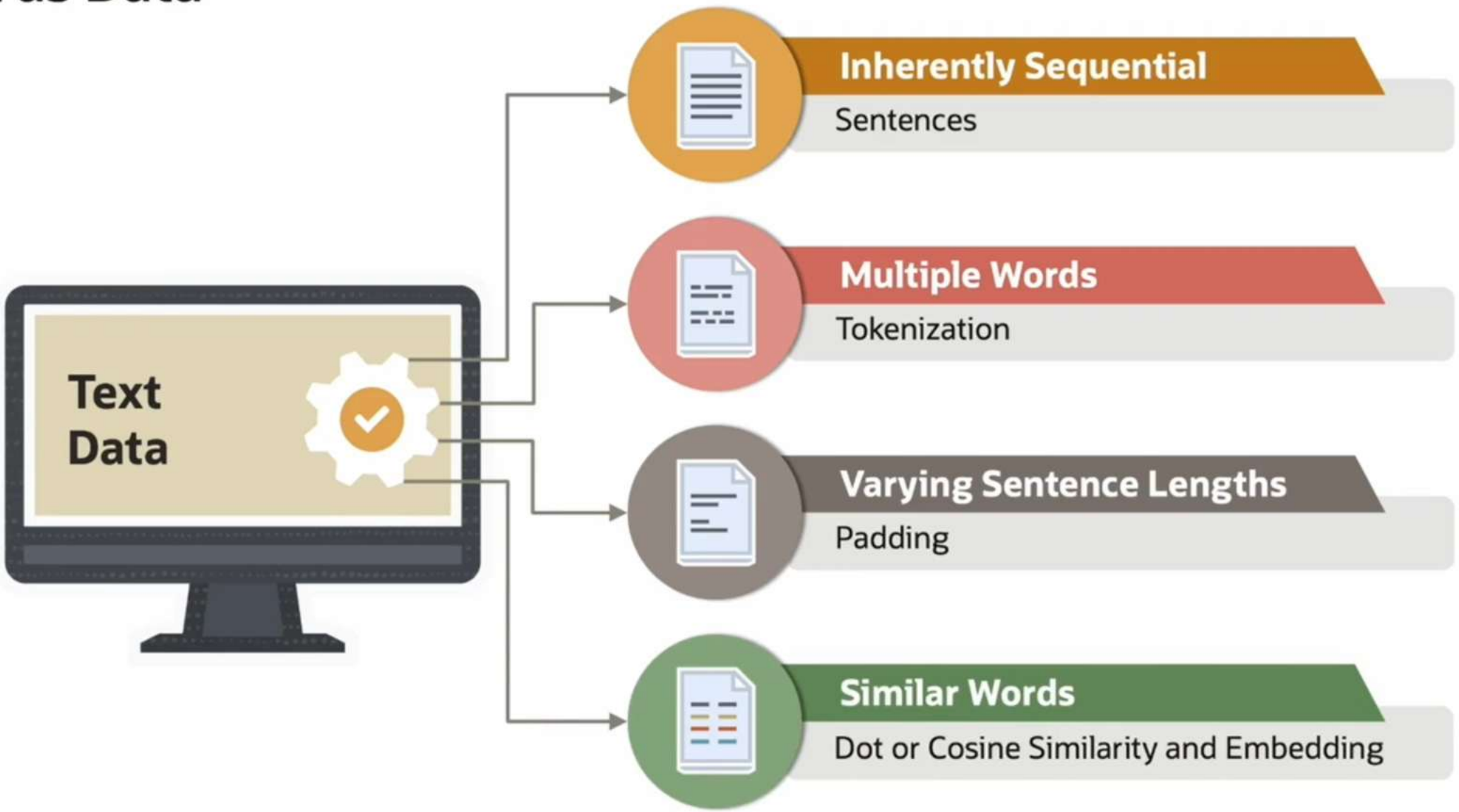
Text-Related AI Tasks

- Detect language.
- Extract entities in a text.
- Extract key phrases.
- Understand sentiment of a text.
- Classify text based on content.
- Translate text.

Generative AI Tasks

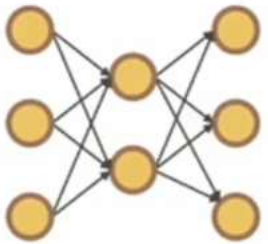
- Create story, poem, etc.
- Summarize text.
- Answer questions.
- Generate image captions.
- Complete text.
- Convert text to speech.

Text as Data


Text Data
Inherently Sequential
Sentences
Multiple Words
Tokenization
Varying Sentence Lengths
Padding
Similar Words
Dot or Cosine Similarity and Embedding

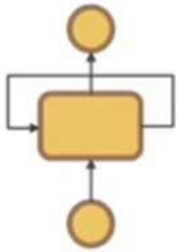
Language AI Models

AI models that are specifically designed to understand, process, and generate natural language



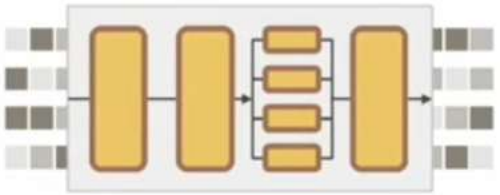
Recurrent Neural Networks

Processes data sequentially and stores hidden states



Long Short-Term Memory

Processes data sequentially and can retain the context better through use of gates



Transformers

Processes data in parallel. Uses concept of self attention to better understand the context

Speech-Related AI Tasks

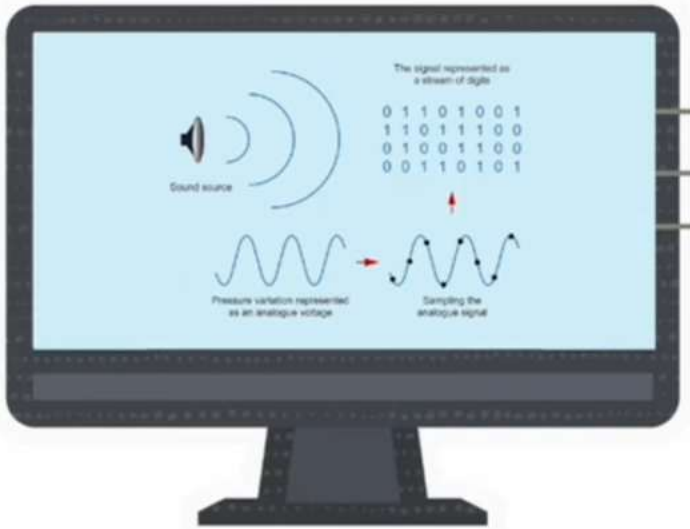
Speech-Related AI Tasks

- Turn speech to text.
- Recognize speaker.
- Perform voice conversion.
- Recognize speech emotions.
- Turn text to speech.

Generative AI Tasks

- Music composition
- Speech synthesis

Audio and Speech as Data



Digitized Snapshots in Time

Sample rate

Sampling Rate of 44.1kHz

Sound reconstructed at 44,100 times per sec

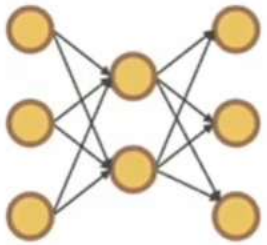
Bit Depth

Number of bits in each of the 44,100 audio pieces

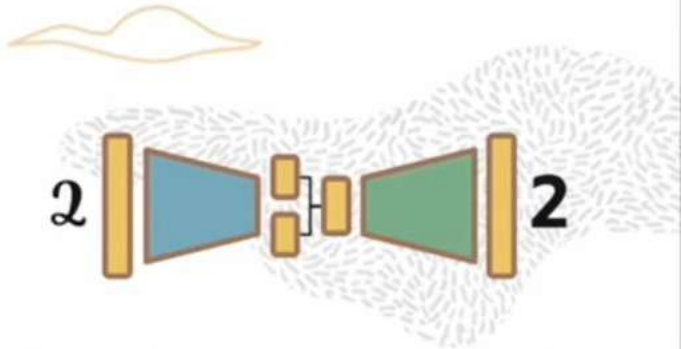
Not much can be inferred by looking at one audio sample.
For example, listening to a song for a fraction of a second.

Audio and Speech AI Models

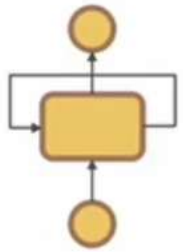
AI models that are specifically designed to process and manipulate audio and speech



Recurrent Neural Networks

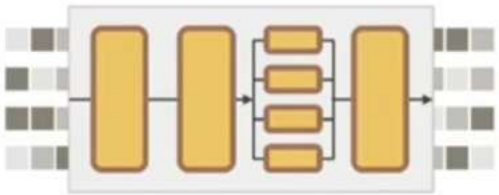


Variational Autoencoders

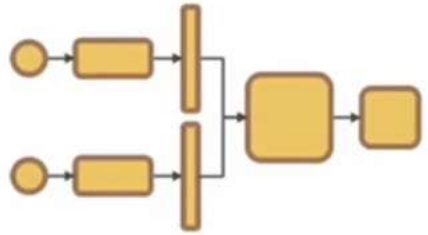


Long Short-Term Memory

Waveform Models



Transformers

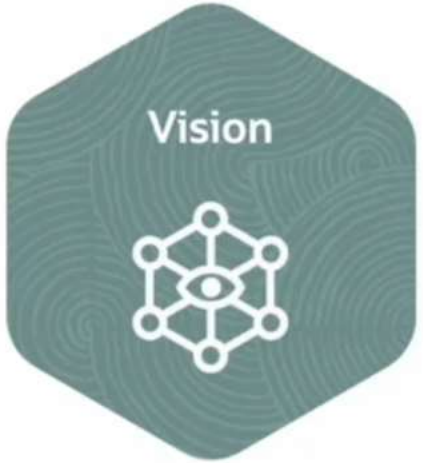


Siamese Networks

Vision-Related AI Tasks

Image-Related AI Tasks

- Classify image.
- Identify objects in an image.
- Identify boundaries in an image.
- Extract text in an image.
- Count objects in an image.



Generative AI Tasks

- Create image from a text.
- Generate images of specific style.
- Generate high-resolution images.
- Repair damaged images.
- Perform image to image translation.
- Get 3D views from 2D sketches.

Images as Data

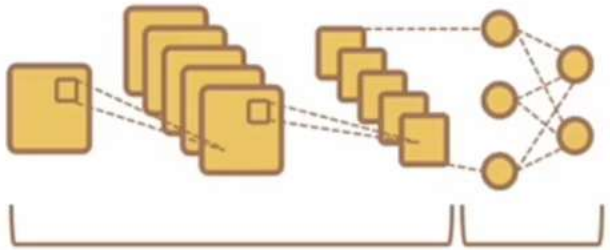
Images consist of pixels.

Pixels are grey scale or color.

We cannot make out what an image is by looking at a pixel.

Vision AI Models

AI models designed to process and understand visual information from images and videos



Convolutional Neural Networks

Detects patterns in images, learning hierarchical representations of visual features

YOLO

Processes the image and detects objects within the image

Generative Adversarial Network

Generates real-looking images

Other AI Tasks

Anomaly Detection

- Detects anomalies in time series data
- Example: Fraud detection

Recommendations

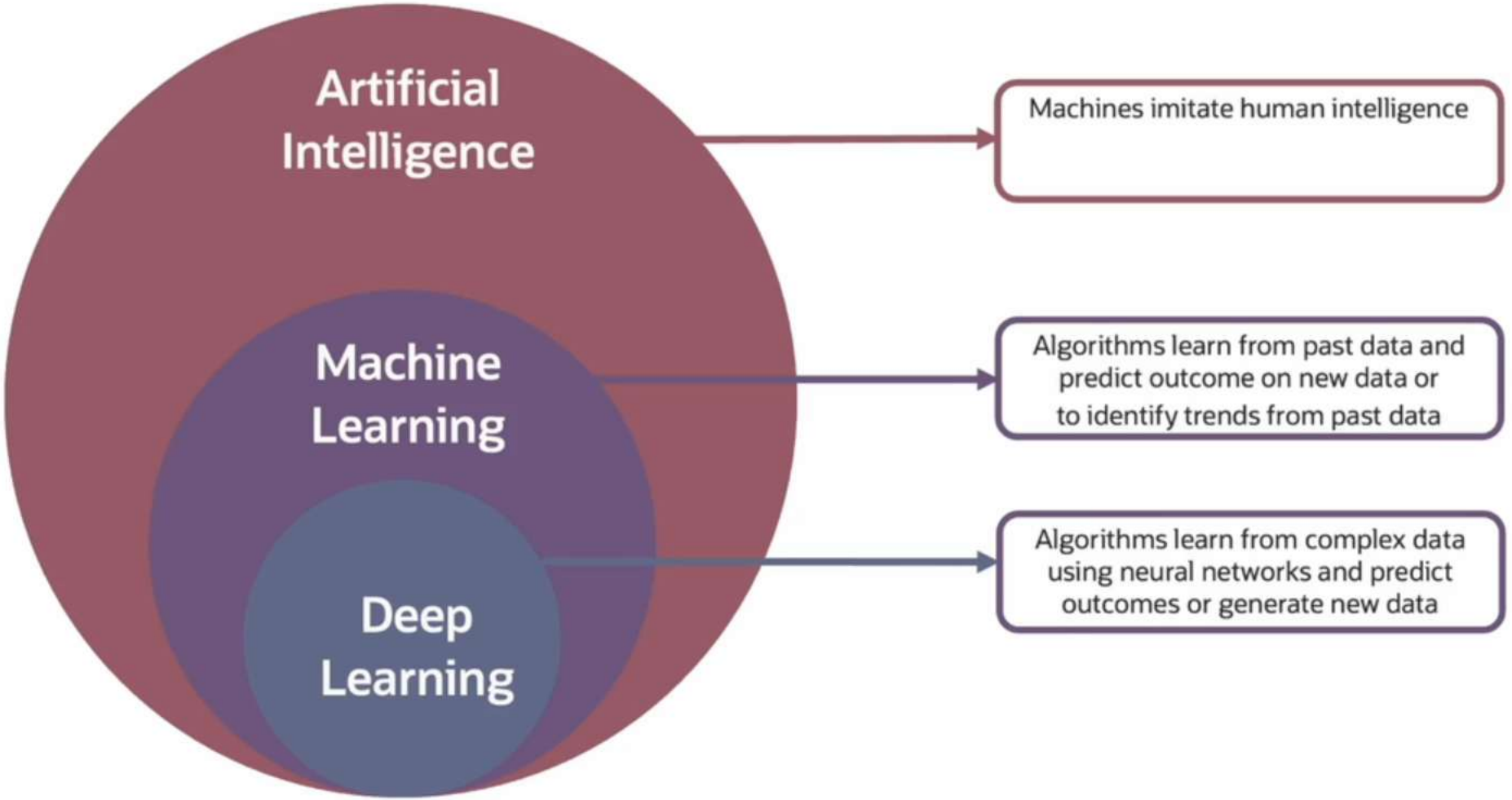
- Recommends products using data of similar products or users
- Example: Ecommerce websites

Forecasting

- Forecasts a future event or value using past data
- Examples: Weather forecast, stock prices

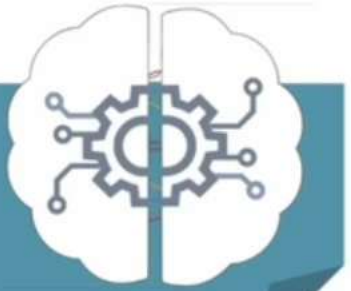
AI vs. ML vs. DL

Relationship Between AI, ML, and DL



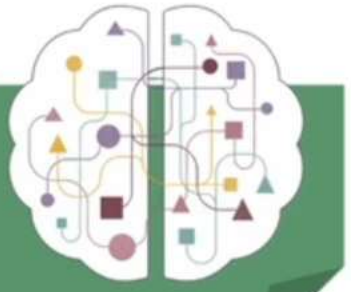
Machine Learning

Supervised Machine Learning



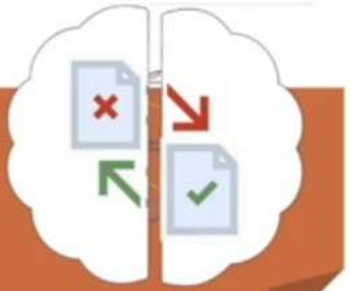
Extracting rules from data

Unsupervised Machine Learning



Extracting trends from data

Reinforcement Learning



Solving tasks by trial and error

Machine Learning

Supervised Machine Learning

Extracting rules from data

How Businesses Took Decisions

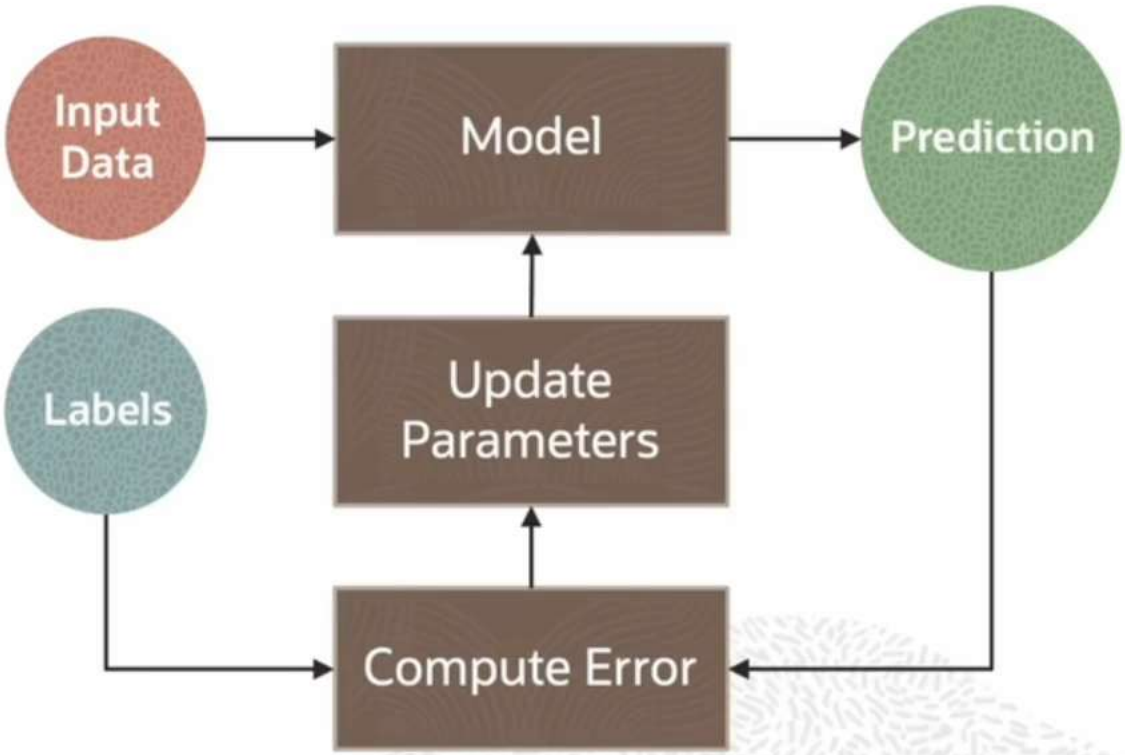
Manually or Using Rules Engine

- Slow
- Requires skilled people
- Adapts to changing rules

Can we build rules by looking at past data?

Train a Model to Predict Outcomes

- Through the process of training, a model can be built to have a specific intelligence to do a specific task.
- The algorithm incrementally updates the model by looking at the data samples.
- Once built, the model can be used to predict an outcome on a new data.



Supervised Machine Learning does the same – learns from labeled data.

Machine Learning

Unsupervised Machine Learning

Extracting trends from data

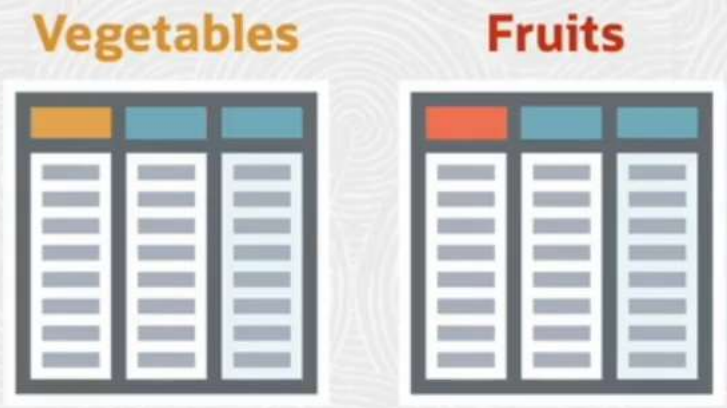
What story does the data tell?

Retail marketing and sales

Regulating streaming services

- Data does not have a specific outcome or label.
- Discovering trends in data can provide insights.
- Similar data can be grouped into clusters.

Gain Insights by Clustering Data

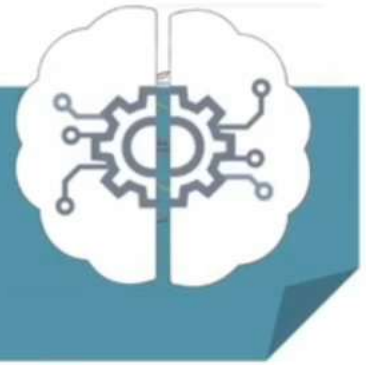


- Which vegetables and fruits are similar nutritionally?
- Cluster vegetable and fruits nutrition data.
- Gain insights to guide your daily diets.

Exploring patterns and grouping similar data into clusters drives Unsupervised Machine Learning.

Machine Learning

Reinforcement Learning



Solving tasks by trial and error

How do we learn to play a game like chess?

- Make a move (decision).
- Check if it's the right move (feedback).
- Keep the outcomes in memory for the next step you take (learning).

Reinforcement Learning does the same – learn by reward. It learns to make decisions by trying different actions and receiving feedback.

Deep Learning

Extracting features and rules from data

Raw data does not speak.

Can we identify if an image is a cat or a dog by looking at one pixel?

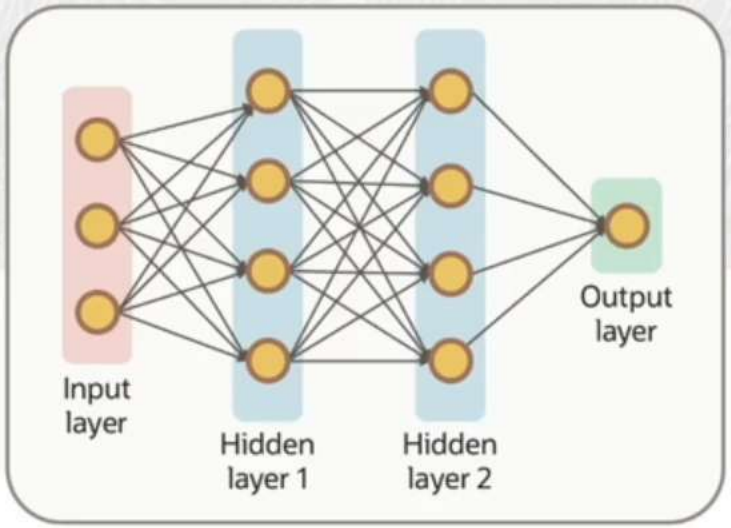
Can we write rules to identify a cat or a dog in an image?

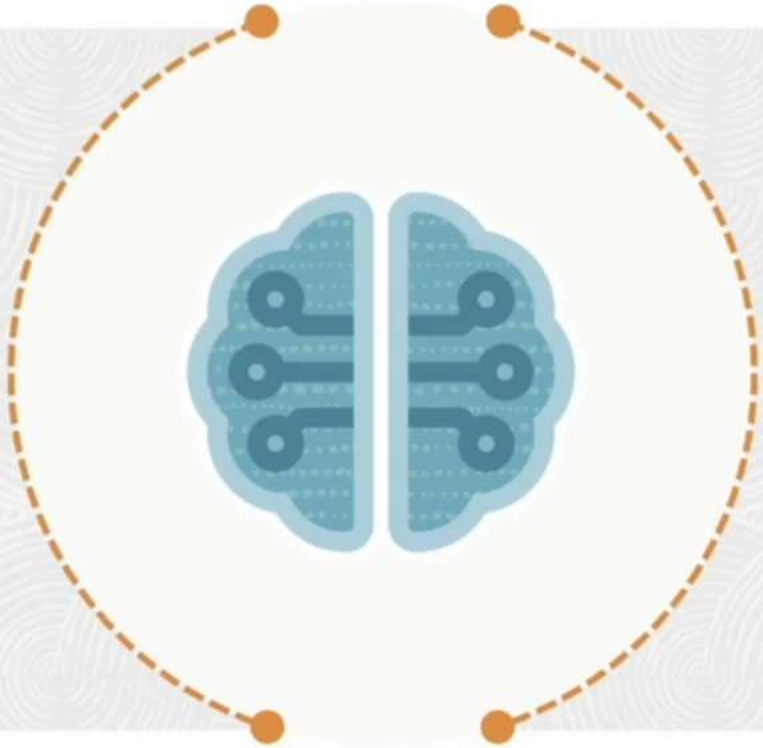
Deep Learning can help here!

It focuses on training neural networks with multiple layers, allowing them to automatically learn and extract complex features and rules from data.

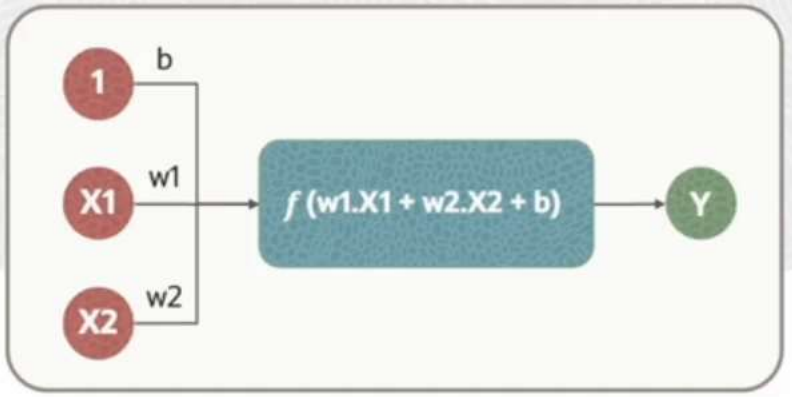
Neural Networks

Made up of interconnected nodes or neurons in a layered structure that resembles the human brain



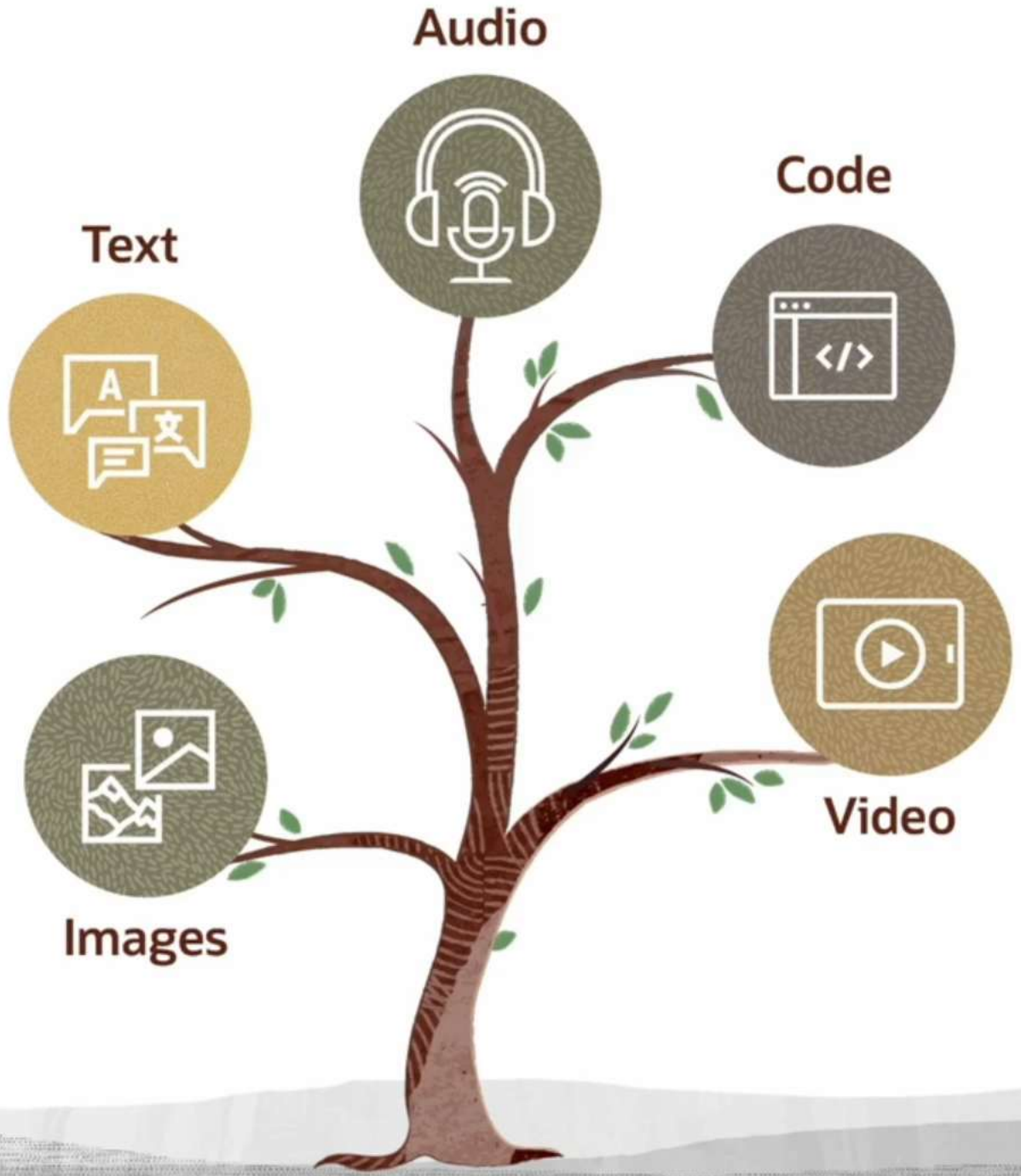


Function approximation is a technique for estimating an unknown underlying function using historical observations from the domain



Generative AI

Machine Learning that can produce content such as audio, text, code, video, images, and other data



1. Welcome to AI Foundations
Welcome to AI Foundations

2. AI Foundations
Introduction to AI
AI - Tasks and Data
AI vs ML vs DL

3. Machine Learning Foundations
Introduction to Machine Learning
Supervised Learning - Regression
Supervised Learning - Classification
Unsupervised Learning
Reinforcement Learning

4. Deep Learning Foundations
Introduction to Deep Learning
Deep Learning Models-Sequence Models
Deep Learning Models - CNN

5. Generative AI and LLM Foundations
Introduction to Generative AI
Introduction to Large Language Models
Transformers Part-1
Transformers Part-2
Prompt Engineering
Customize LLMs with your data

6. OCI AI Portfolio
AI Services Overview
ML Services Overview
AI Infrastructure
GPUs and Superclusters in OCI
Responsible AI

7. OCI Generative AI Service
OCI Generative AI
Vector Search
Select AI

8. OCI AI Services
Language Intro
Speech Intro
Vision Intro
Document Understanding

Machine Learning Foundations

Module 3

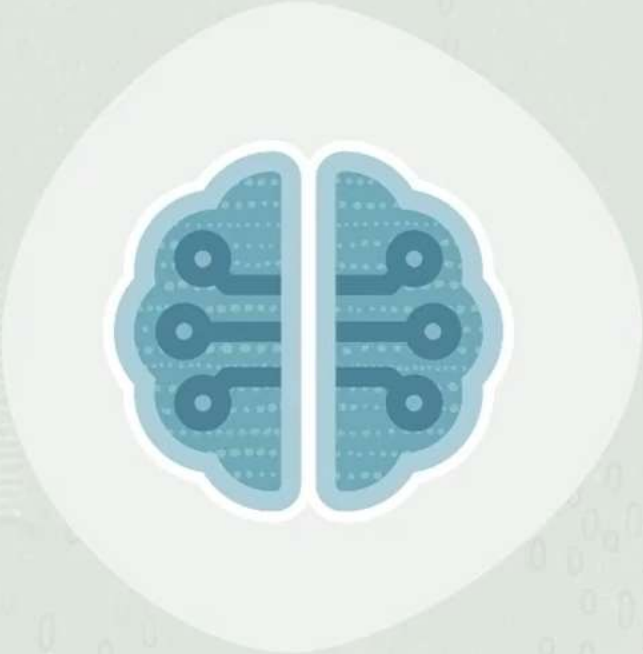
Objectives

- Explain Machine Learning and its different flavors.
- Describe how Supervised Learning works.
- Explain Linear Regression and Classification.
- Explain Unsupervised Learning.
- Describe how Reinforcement Learning works.

Machine Learning Foundations

What Is Machine Learning?

- A subset of artificial intelligence that focuses on creating computer systems that can learn and improve from experience
- Powered by algorithms that incorporate intelligence into machines



Machine Learning Is All Around Us

Online shopping

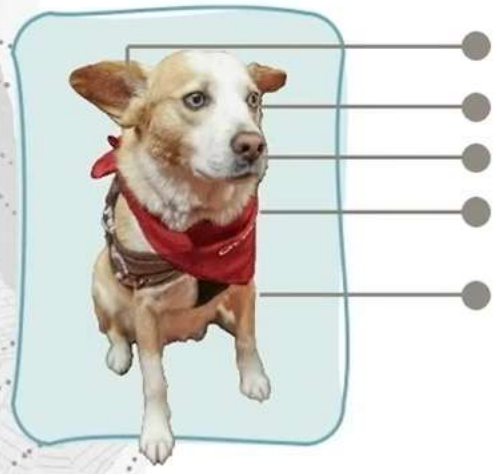
ML provides statistical tools to analyze, visualize, and make predictions from data.

Spam mail warning

Netflix movie suggestions

Self-driving cars

Input Features and Output Labels



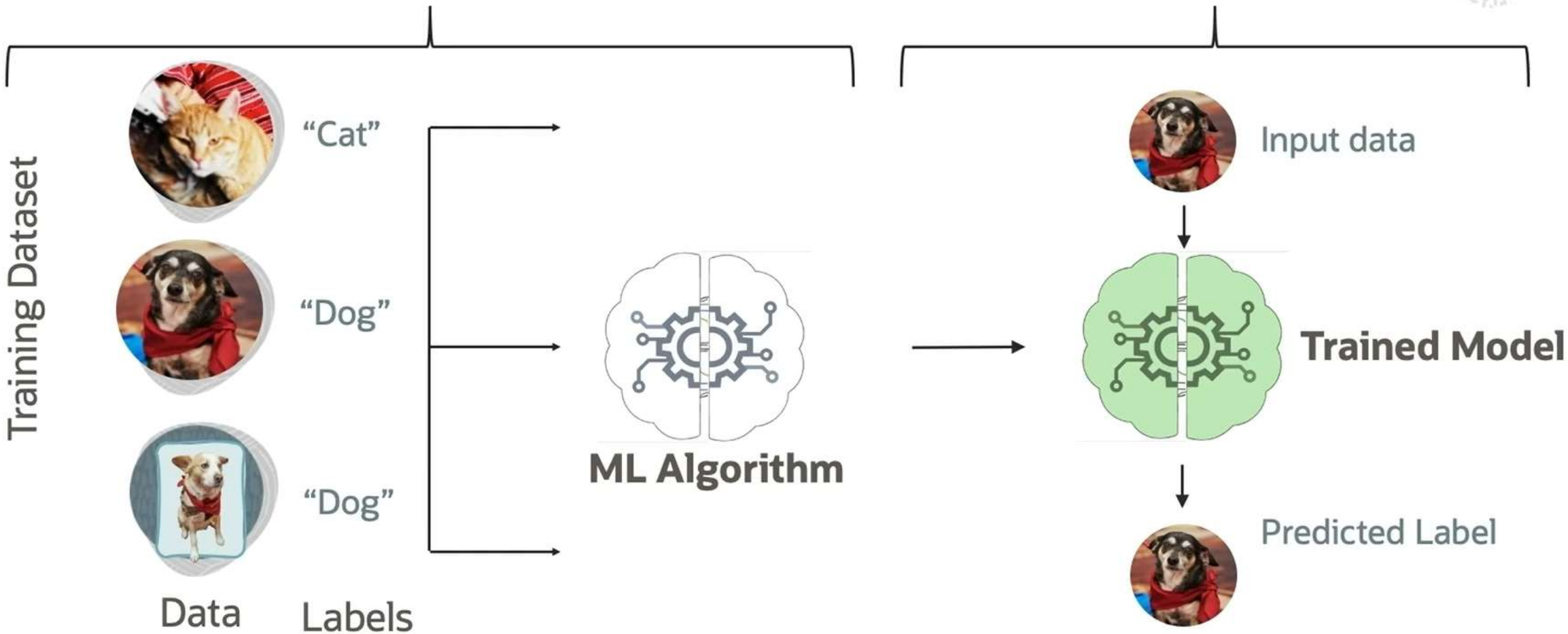
Output Label

Body Color	Texture	Eye Colour	Ear Shape	Nose Colour	Body Shape	Animal
Black and cream	Bumpy	Brown	Bat	Black	Ectomorphic	Dog
Black and tan	Flaky	Brown	Button	Pink	Ectomorphic	Dog
Black	Shiny	Blue	Rounded tips	White	Cobby	Cat
Chocolate and cream	Bumpy	Hazel	Drop	Brown	Endomorphic	Dog
Red	Smooth	Green	Pointed tips	White	Medium	Cat

Input Features

Training
Learning new capability from existing data

Inference
Applying this capability to new data



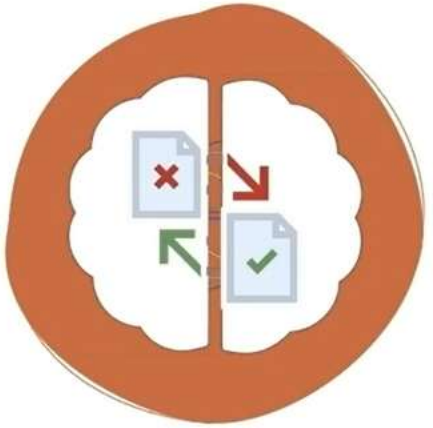
Types of Machine Learning

Supervised

Classify data or
make predictions

Unsupervised

Understand relationships
within datasets



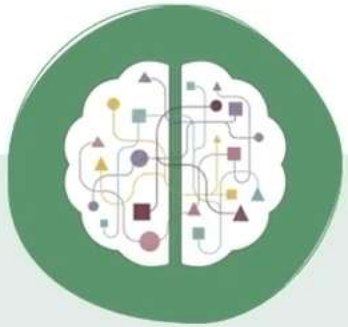
Reinforcement

Make decisions
or choices

Machine Learning Use Cases

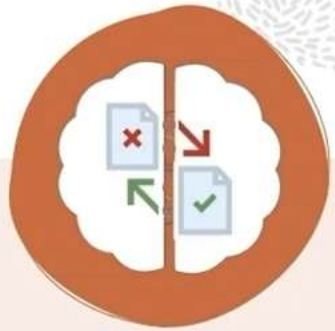
Supervised

- Disease detection
- Weather forecasting
- Stock price prediction
- Spam detection
- Credit scoring



Unsupervised

- Fraudulent transactions detection
- Customer segmentation
- Outlier detection
- Targeted marketing campaigns



Reinforcement

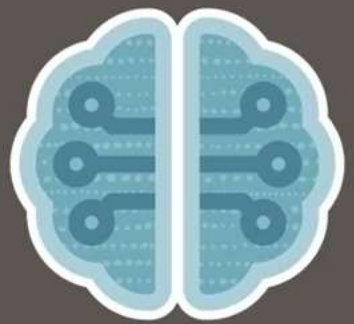
- Automated robots
- Autonomous cars
- Video games
- Healthcare

Supervised Learning: Regression

Supervised Learning

House price prediction

Machine learning model that learns from labeled data

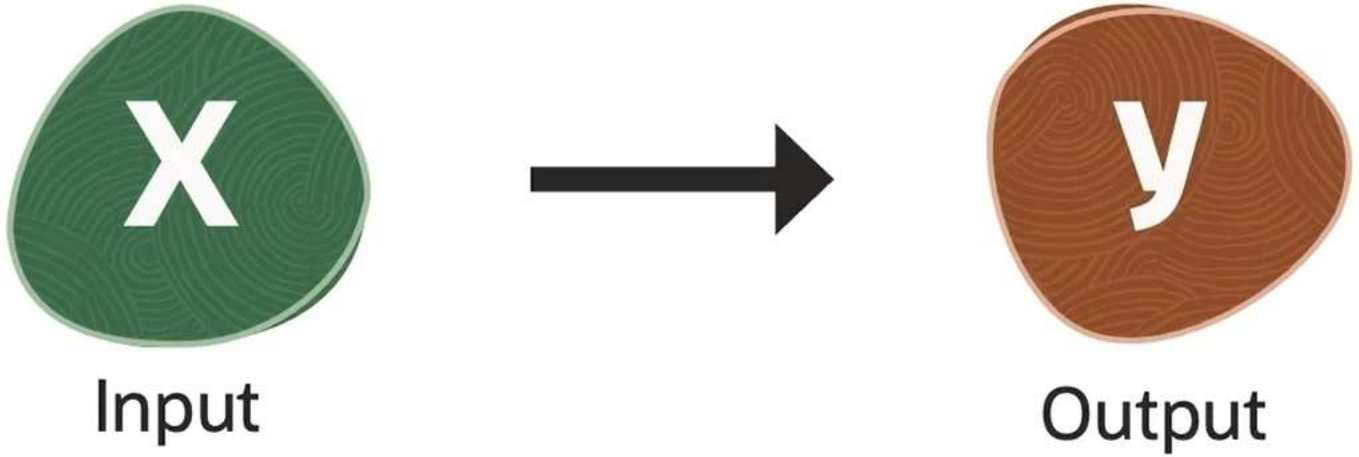


Sentiment analysis

Disease detection

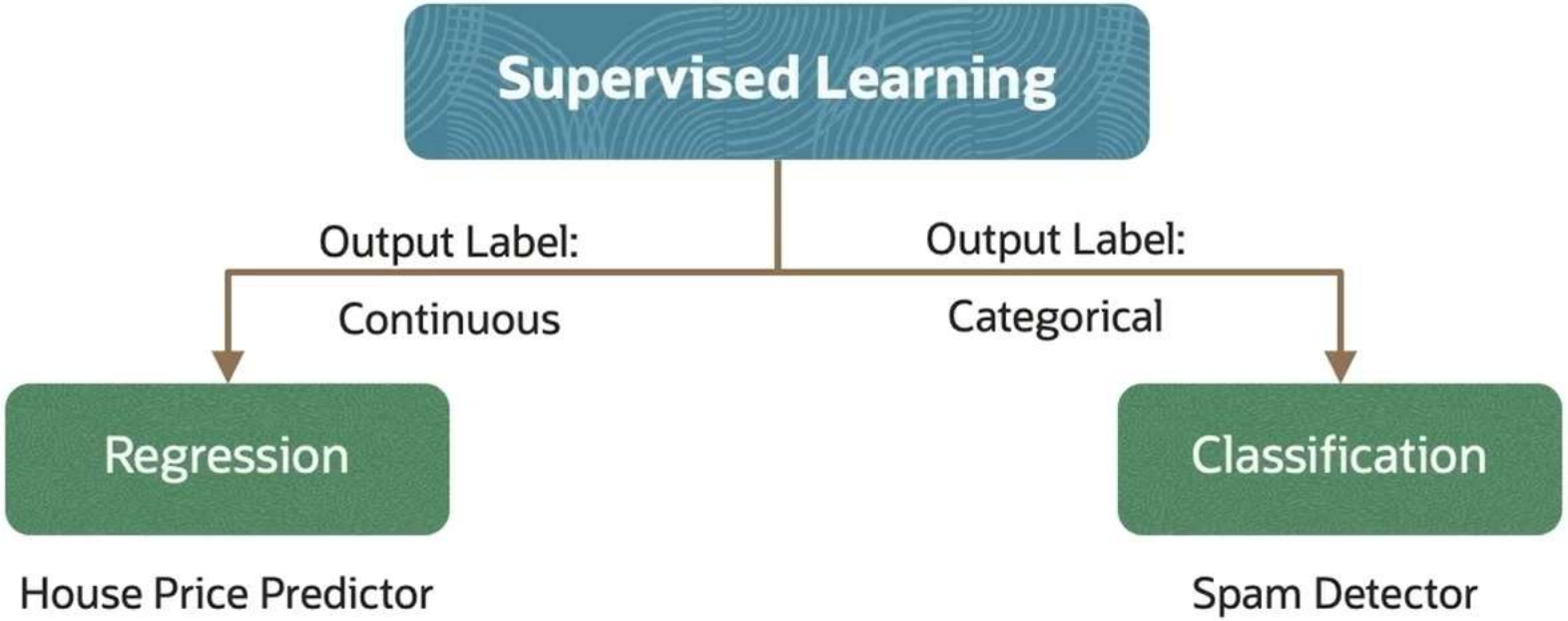
Stock price prediction

Supervised Learning



Learns from labeled data

Types of Supervised Learning



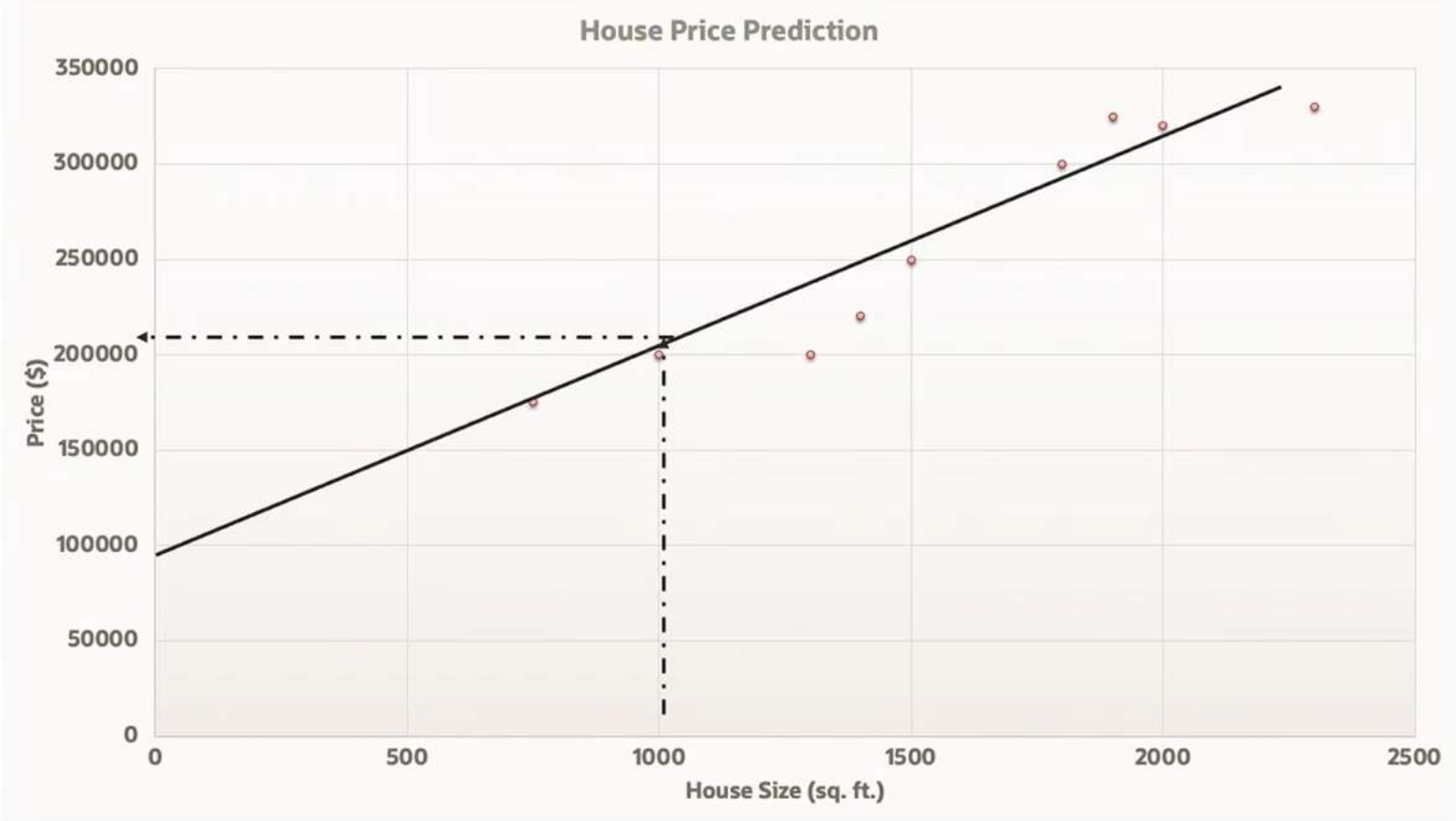
House Price Prediction

Independent Feature	Dependent Feature
House Size (sq. ft.)	**Price ($)**
1,400	220,000
1,800	300,000
2,000	320,000
1,300	200,000
1,900	325,000
2,300	330,000
1,500	250,000

Training data set

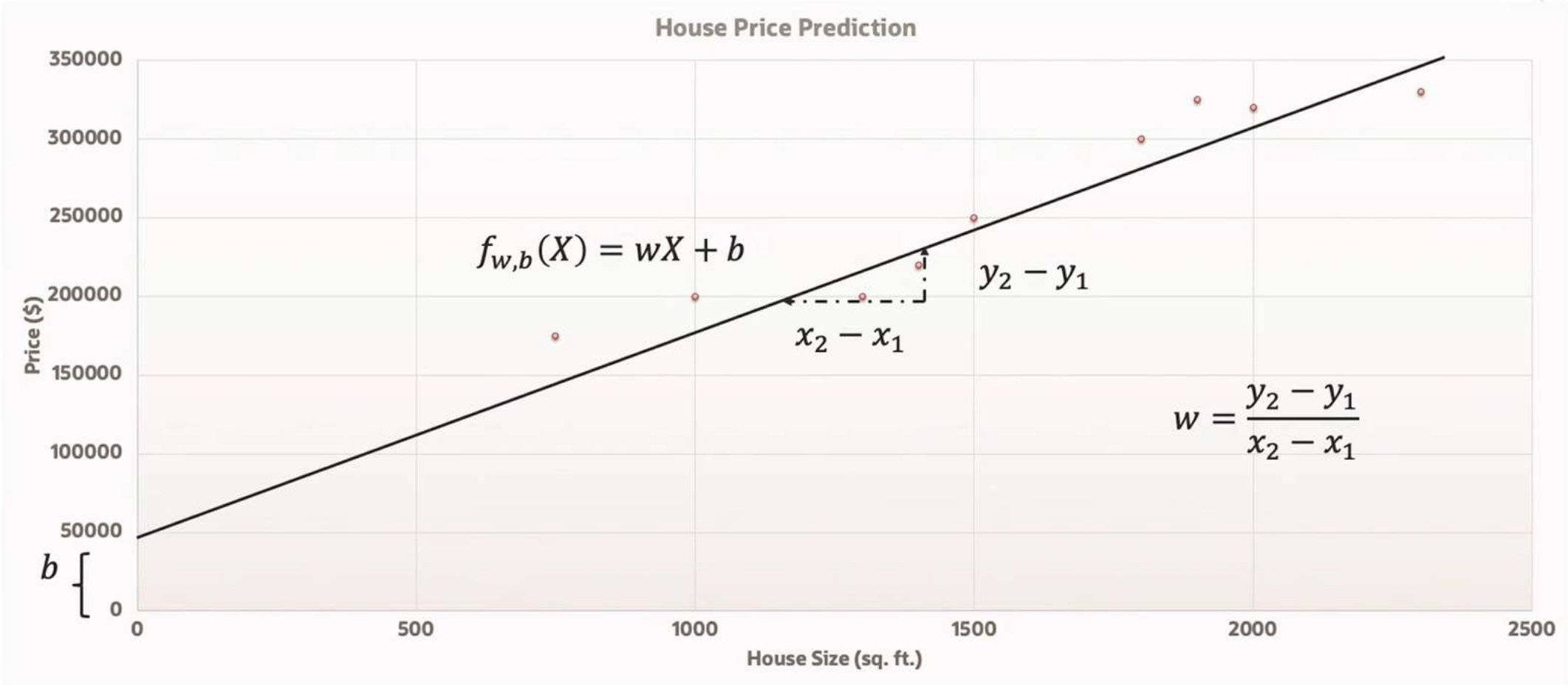
Predicting House Price Based on House Size

Scatter plots aid in visualizing the relationship between inputs and outputs.



House Size (sq. ft.)	Price ($)
1,400	220,000
1,800	300,000
2,000	320,000
1,300	200,000
1,900	325,000
2,300	330,000
1,500	250,000

Fitting the Line



Minimizing Loss

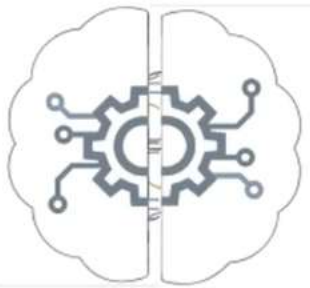
Loss is a number indicating how far the predicted value is from the actual value.

Squared error is the squared difference between the predicted point and the actual data point, which needs to be minimized during training.

Training

Training data set

Learning algorithm



f

Prediction

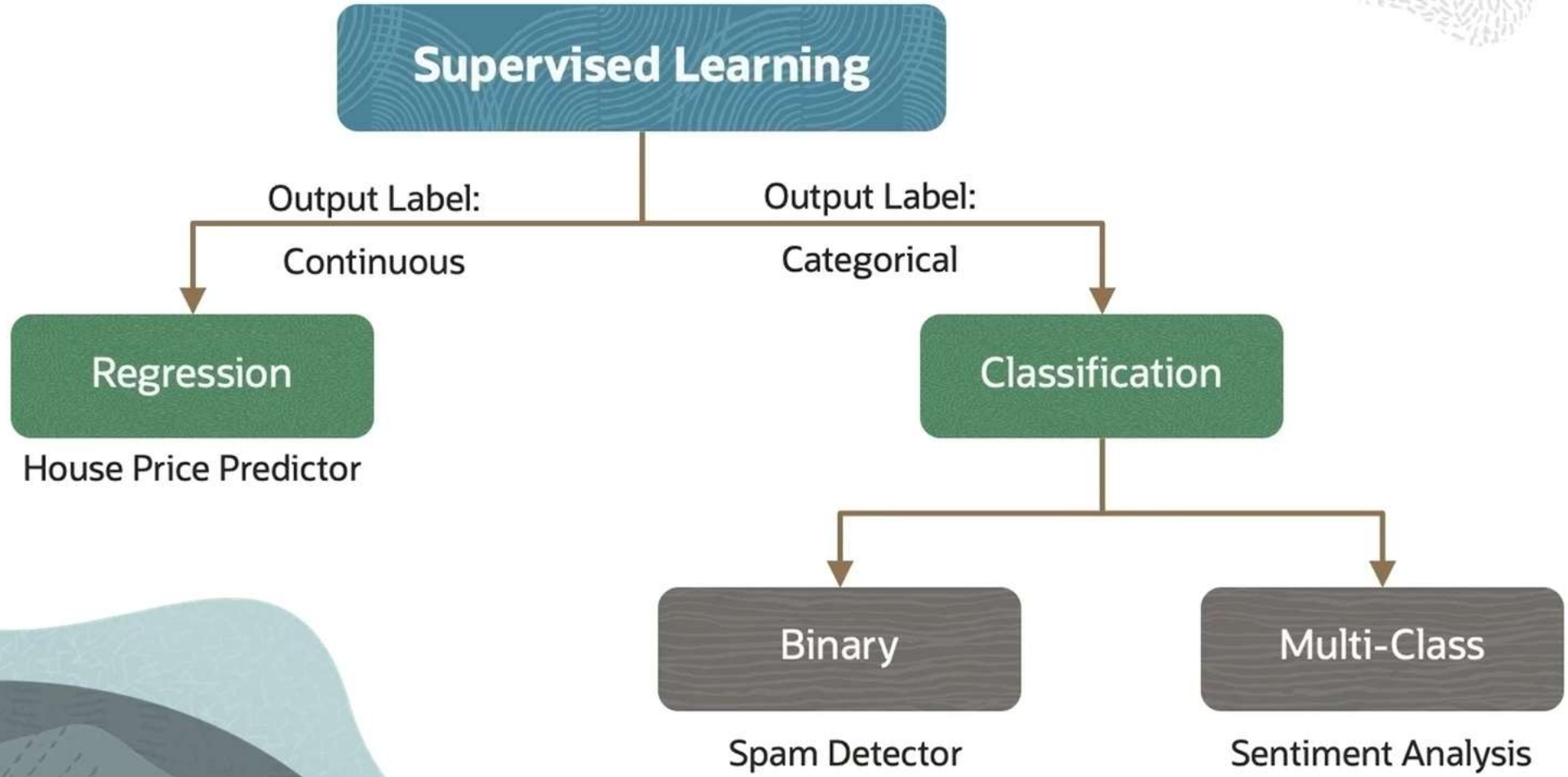
Size

f

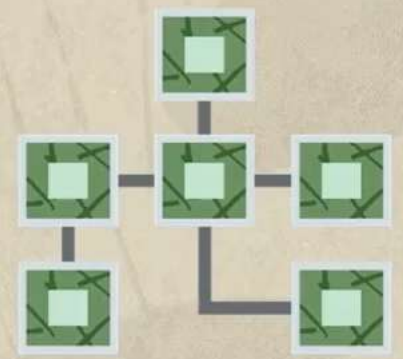
Price (estimated)

Supervised Learning: Classification

Types of Supervised Learning



Classification

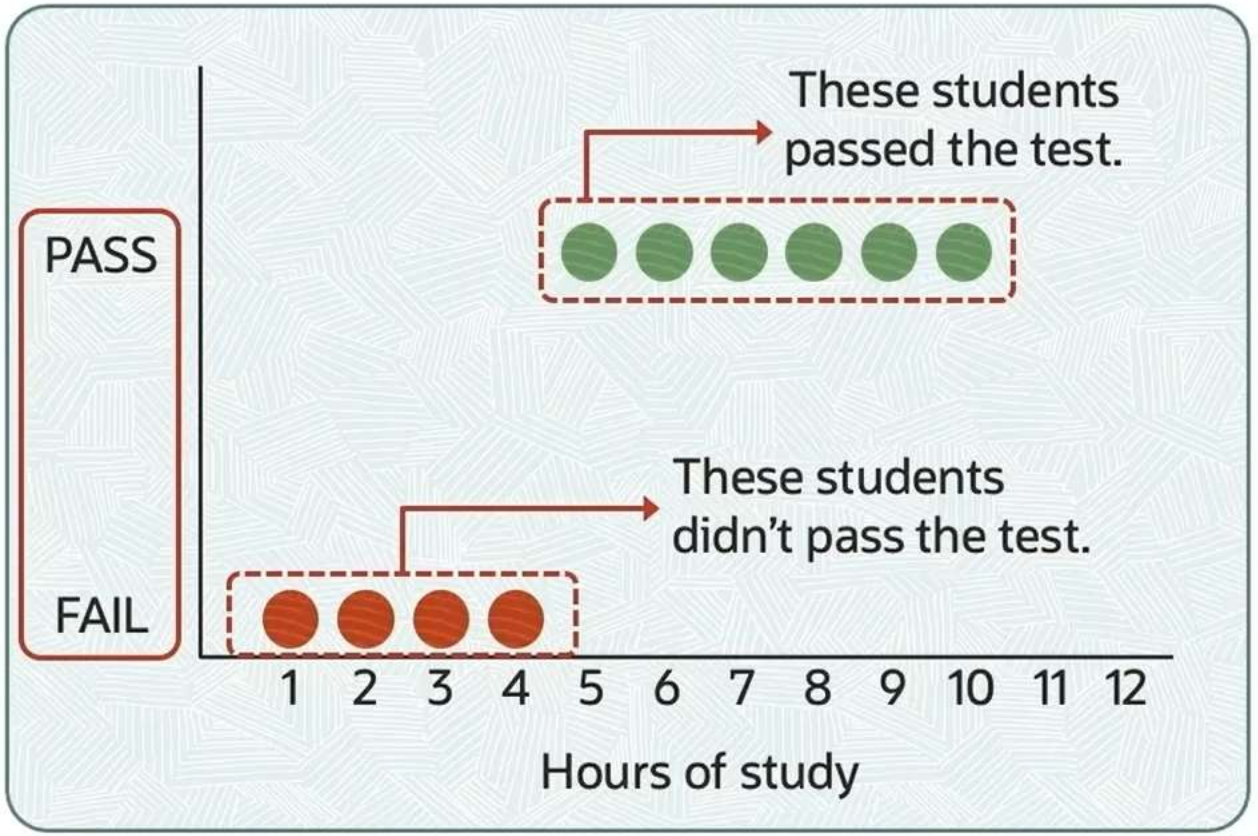


A supervised ML technique used to categorize or assign data points into predefined classes based on their features or attributes

Example: Spam classifier for emails

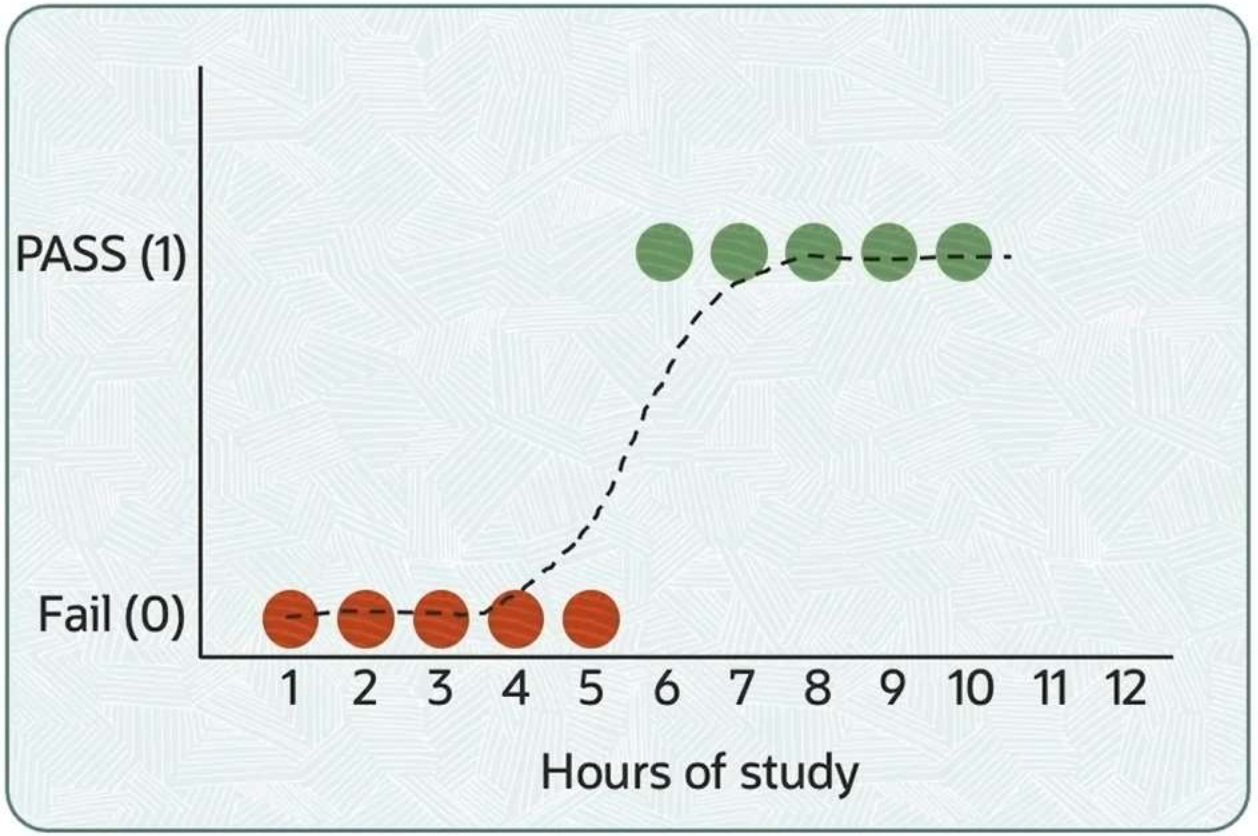
Logistic Regression

Logistic Regression helps in predicting something is **True** or **False** instead of predicting something continuous such as the house prices.



Logistic Regression

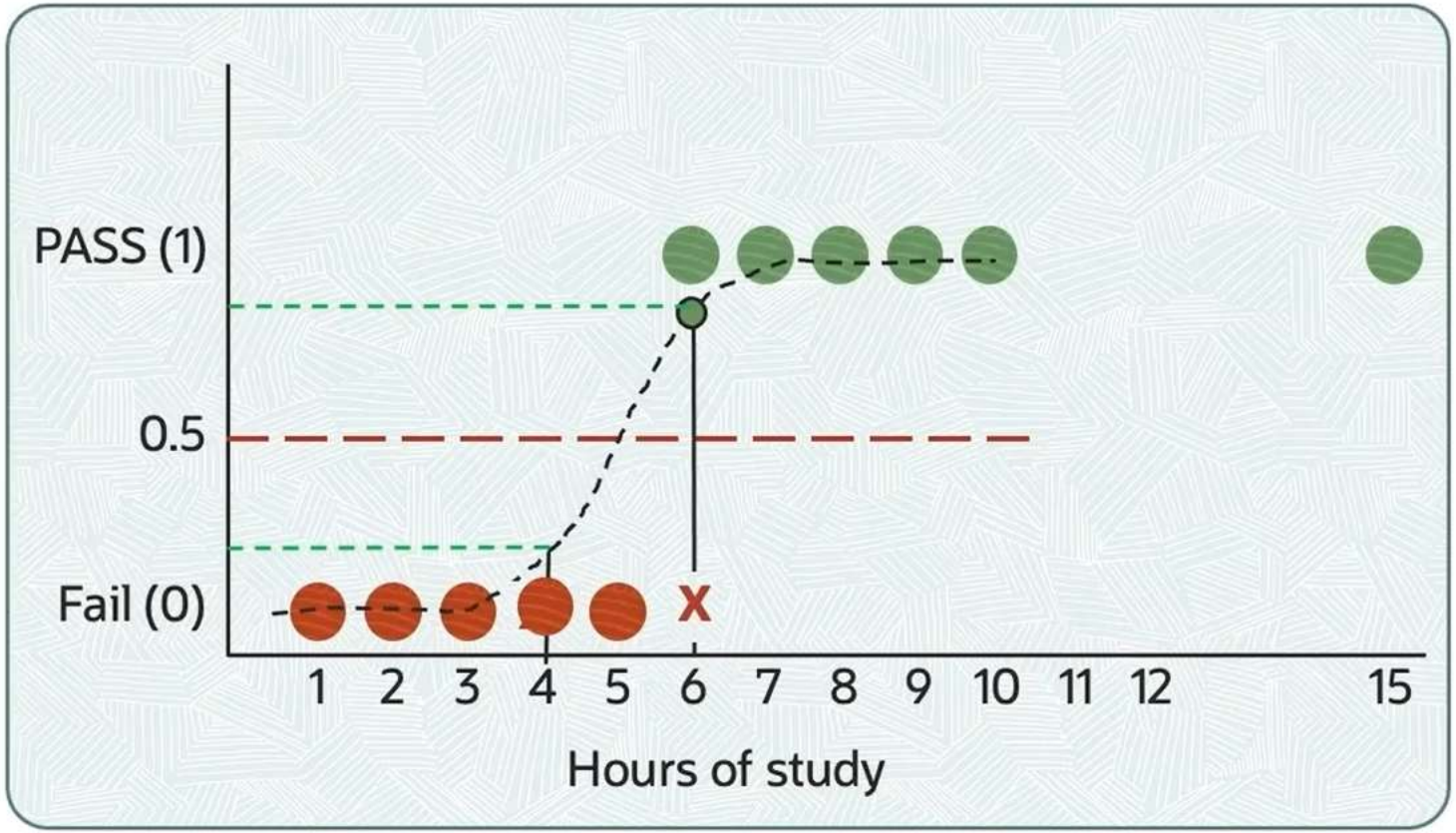
Instead of fitting a straight line to the data, Logistic Regression fits an S-shaped curve to the data.



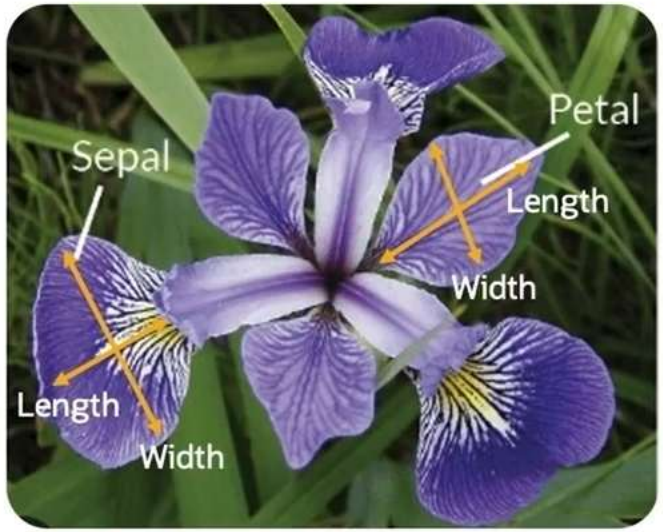
Logistic Regression

A model built using Logistic Regression can be used for classification.

While predicting, the probability of a student passing or failing the test is computed.



Demo: Iris Dataset



Output Label

Sepal Length	Sepal Width	Petal Length	Petal Width	Label
5.1	3.5	1.4	0.2	Iris-setosa
6.9	3.1	4.9	1.5	Iris-versicolor
7.2	3.6	6.1	2.5	Iris-virginica
7.1	3	5.9	2.1	Iris-virginica

Input Features

Basic Machine Learning
Part-1

Machine Learning Process

- Loading data,
- Preprocessing,
- Training a model,
- Evaluating the model
- Making predictions.

In [1]:
```
#Import necessary libraries
import pandas as pd
from sklearn.linear_model import LogisticRegression
```

In [2]:
```
#Loading the Dataset and Displaying the First Few Rows
iris_data = pd.read_csv('iris.csv')
iris_data.head()
```

Out[2]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

In [ ]:
```
# Split the data into features (X) and labels (y)
X = iris_data.drop(columns=['Id', 'Species'])
y = iris_data['Species']
```

1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

In [3]:
```
# Split the data into features (X) and labels (y)
X = iris_data.drop(columns=['Id', 'Species'])
y = iris_data['Species']
```

In [4]:
```
X.head()
```

Out[4]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

In [5]:
```
#create a ml model
model = LogisticRegression()
```

In [6]:
```
#train the model
model.fit(X.values, y)
```

Out[6]:
```
LogisticRegression()
```

In [7]:
```
#predict using the trained model
predictions = model.predict([[4.6, 3.5, 1.5, 0.2]])
```

In [8]:
```
# print the predictions
print(predictions)
```

```
['Iris-setosa']
```

In [ ]:

Machine Learning Process

- Loading data,
- Preprocessing,
- Training a model,
- Evaluating the model
- Making predictions.

In [ ]:
```
#Import necessary libraries
import pandas as pd
from sklearn.linear_model import LogisticRegression
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
```

In [ ]:
```
#Loading the Dataset and Displaying the First Few Rows
iris_data = pd.read_csv('iris.csv')
iris_data.head()
```

In [ ]:
```
# Split the data into features (X) and labels (y)
X = iris_data.drop(columns=['Id', 'Species'])
y = iris_data['Species']
```

In [ ]:
```
X.head()
```

In [ ]:
```
#create a ml model
model = LogisticRegression()
```

In [ ]:
```
#train the model
model.fit(X.values, y)
```

In [ ]:
```
#predict using the trained model
predictions = model.predict([[4.6, 3.5, 1.5, 0.2]])
```

In [ ]:
```
# print the predictions
print(predictions)
```

In [ ]:

In [2]:
```
#Loading the dataset and displaying the first few rows
iris_data = pd.read_csv('iris.csv')
iris_data.head()
```

Out[2]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

In [3]:
```
# Split the data into features (X) and labels (y)
X = iris_data.drop(columns=['Id', 'Species'])
y = iris_data['Species']
```

In [4]:
```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [5]:
```
# Standardize the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [6]:
```
#create a ml model
model = LogisticRegression()
```

In [7]:
```
#train the model
model.fit(X_train_scaled, y_train)
```

Out[7]:
```
LogisticRegression()
```

In [ ]:
```
# Evaluate the model on the testing set
y_pred = model.predict(X_test_scaled)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

In [ ]:

In [4]:
```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [5]:
```
# Standardize the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [6]:
```
#create a ml model
model = LogisticRegression()
```

In [7]:
```
#train the model
model.fit(X_train_scaled, y_train)
```

Out[7]:
```
LogisticRegression()
```

In [8]:
```
# Evaluate the model on the testing set
y_pred = model.predict(X_test_scaled)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 1.0
```

In [9]:
```
# Sample new data for prediction
new_data = np.array([[5.1, 3.5, 1.4, 0.2],
                     [6.3, 2.9, 5.6, 1.8],
                     [4.9, 3.0, 1.4, 0.2]])
```

In [10]:
```
# Standardize the new data
new_data_scaled = scaler.transform(new_data)
```

In [11]:
```
# Make predictions
predictions = model.predict(new_data_scaled)
```

In [12]:
```
# Display the predicted classes
print("Predictions:", predictions)
```

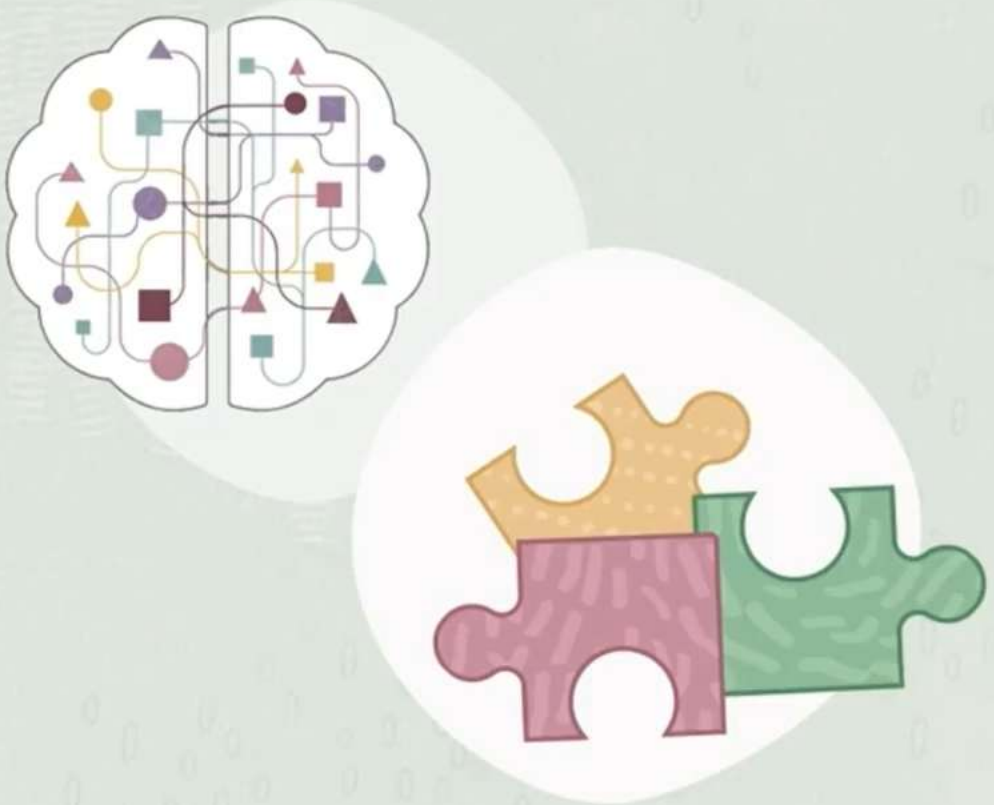
```
Predictions: ['Iris-setosa' 'Iris-virginica' 'Iris-setosa']
```

In [ ]:

Unsupervised Machine Learning

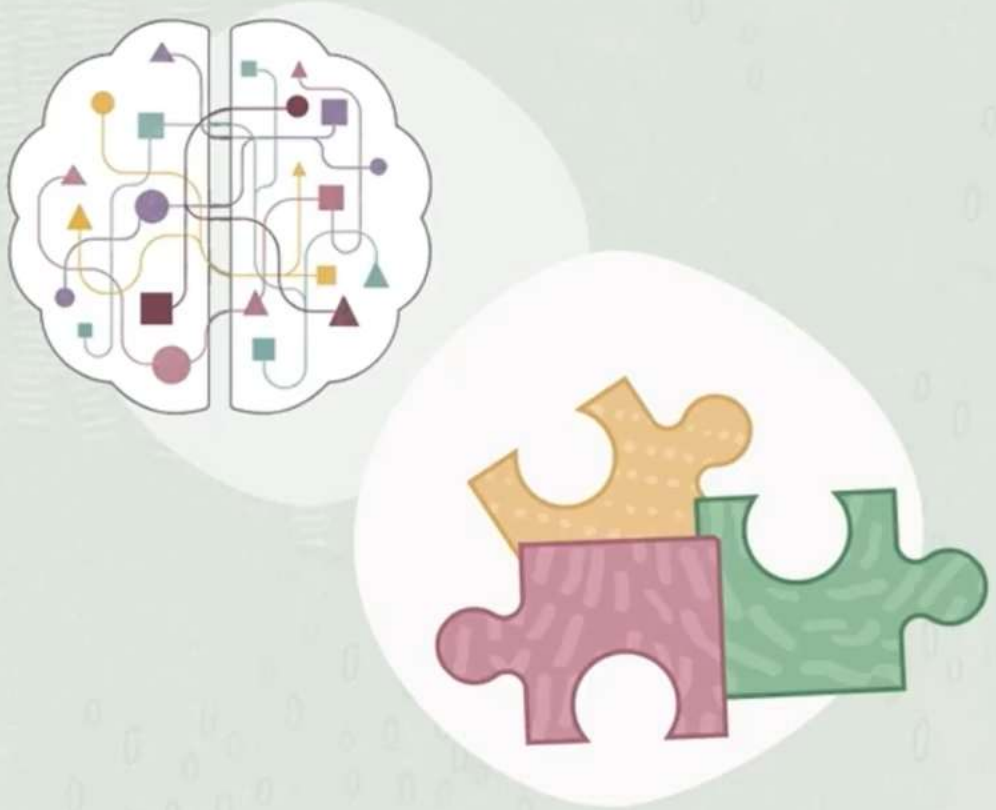
What is Unsupervised Learning?

- A type of machine learning where there are no labeled outputs
- Algorithms learn the patterns in the data and group similar data items



What is Unsupervised Learning?

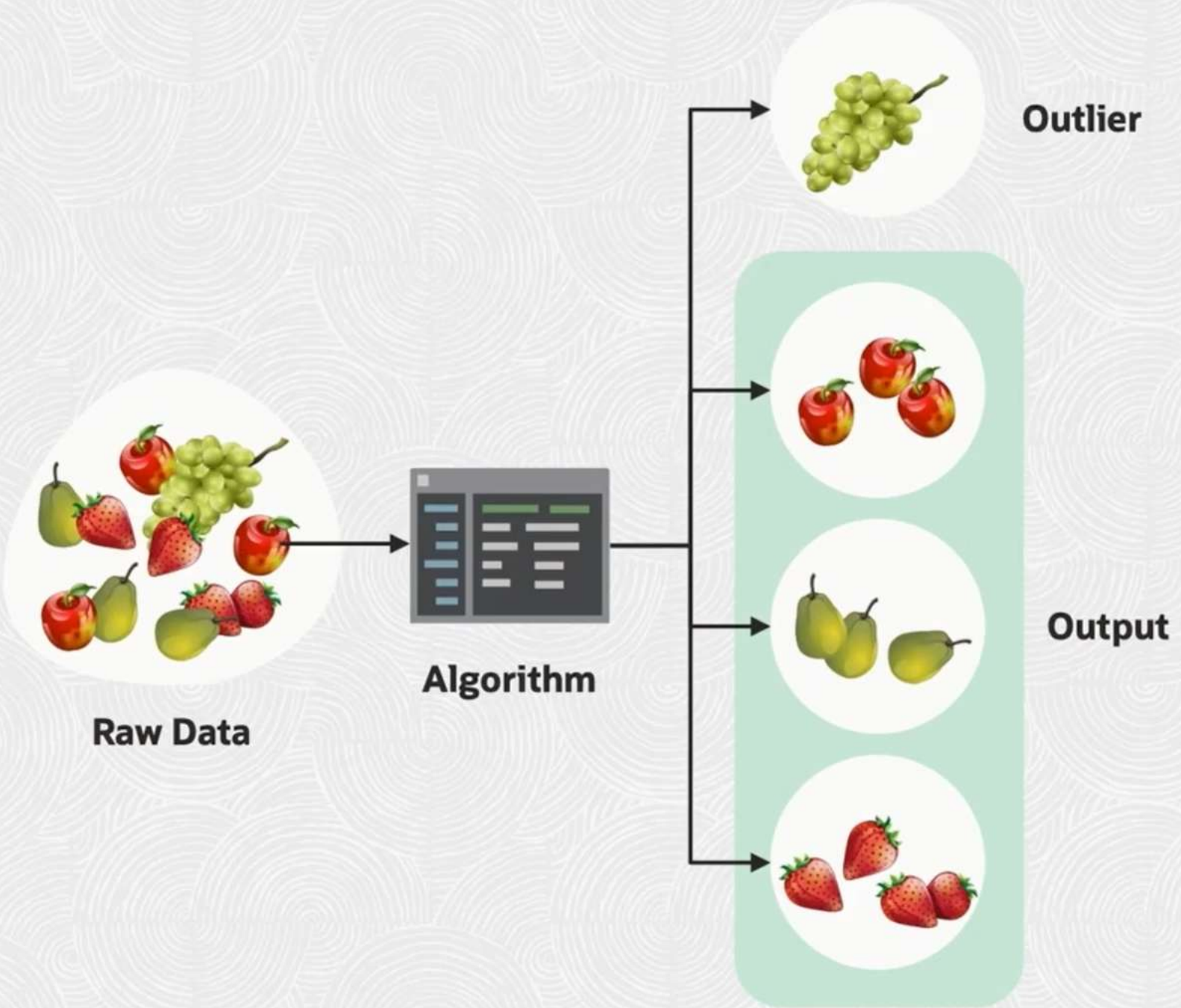
- A type of machine learning where there are no labeled outputs
- Algorithms learn the patterns in the data and group similar data items



Clustering

Clustering is the grouping of similar data items.

Data items are more similar within a cluster than the items outside it.



Use Case 1

Market Segmentation

Action

- Input is purchasing details.
- Identify similar customers based on purchasing behavior.

Output

Target advertisements

Use Case 2

Outlier Analysis

Action

- Input is credit card purchase details.
- Identify fraudulent transactions.

Output

Anomaly detection

Use Case 3

Recommendation Systems

Action

- Inputs are users' movie viewing history.
- Identify users based on genre of movies watched.

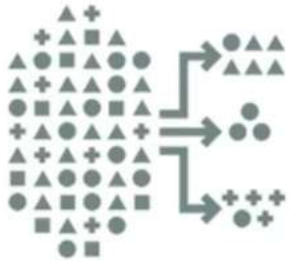
Output

Personalized movie recommendations

Similarity

Similarity is how close two data points are to each other and is a value between 0 and 1.

Unsupervised Workflow



Prepare the data

Remove missing values, normalize the data, and perform feature scaling.

Create similarity metrics

Choose a similarity metric based on nature of data and clustering algorithm used.

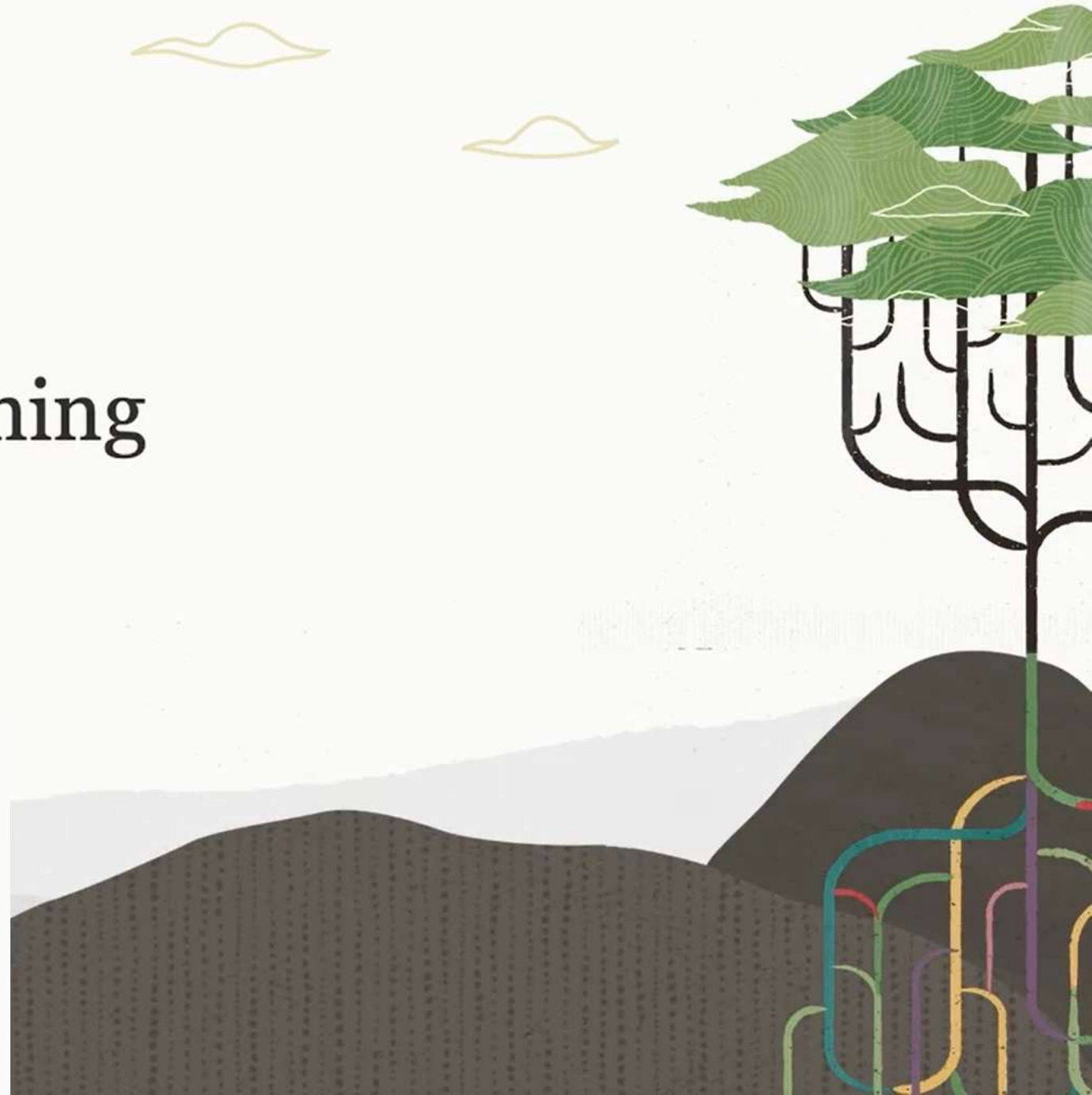
Run clustering algorithm

Use the chosen similarity metrics to cluster the data.

Interpret results and adjust clustering

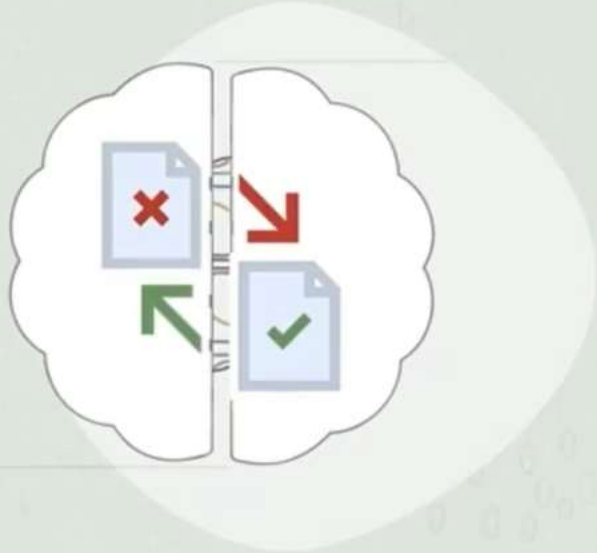
Check the quality of your clustering output by verifying against expectations iteratively.

Reinforcement Learning



Reinforcement Learning

- A type of Machine Learning that enables an agent to learn from its interactions with the environment
- Receives feedback in the form of rewards or penalties, without any labeled dataset



Reinforcement Learning Examples

Autonomous vehicles

Smart devices

Industrial automation

Gaming and entertainment

Terminology in RL

Agent

Interacts with environment, takes actions, learns from feedback

Environment

External system with which the agent interacts

State

Representation of the current situation of the environment at a particular time

Action

Possible moves or decisions that the agent can take in each state

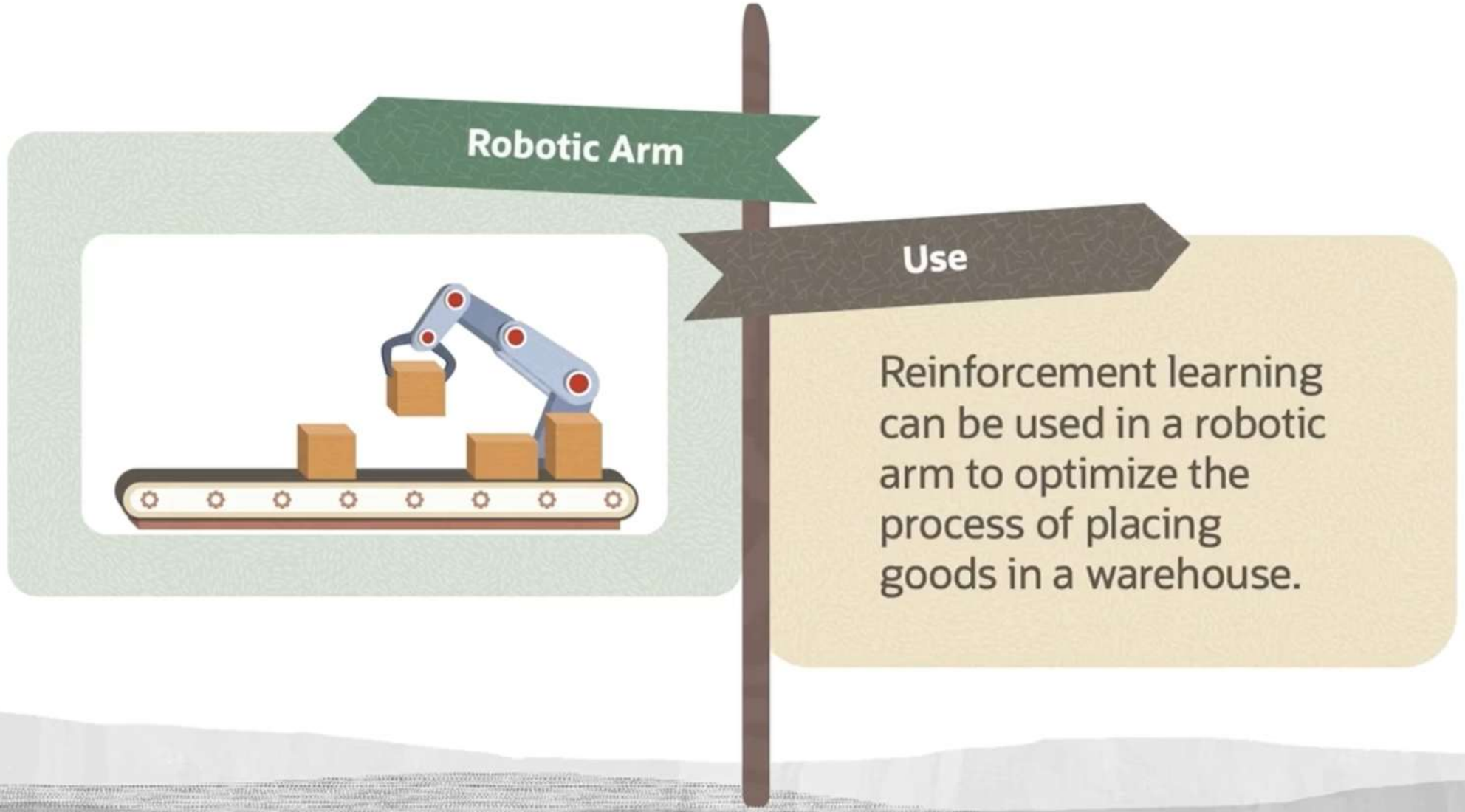
Policy

Mapping that the agent uses to decide which action to take in a given state

RL Training Loop

- Goal is to find a policy that will yield a lot of rewards for the agent.
- Optimal policy is learned through training by using algorithms like Q learning or Deep Q Learning.

RL Example


Robotic Arm
Use
Reinforcement learning can be used in a robotic arm to optimize the process of placing goods in a warehouse.

Train a Robotic Arm Using RL

Step 1

Environmental Setup

Set the robotic arm, warehouse layout, goods, and target locations.

Step 2

State Representation

State includes position and orientation of arm, items to be picked, and target locations.

Step 3

Action Space

Define the possible actions the robotic arm can take in each state.

Step 4

Rewards and Penalties

It learns by rewards and punishments.

Step 5

Training and Iterative Improvement

It starts in a random state and takes actions in the environment.

Through multiple training iterations, the robotic arm learns better strategies for picking up and placing items in the warehouse.

1. Welcome to AI Foundations
Welcome to AI Foundations

2. AI Foundations
Introduction to AI
AI - Tasks and Data
AI vs ML vs DL

3. Machine Learning Foundations
Introduction to Machine Learning
Supervised Learning - Regression
Supervised Learning - Classification
Unsupervised Learning
Reinforcement Learning

4. Deep Learning Foundations
Introduction to Deep Learning
Deep Learning Models-Sequence Models
Deep Learning Models - CNN

5. Generative AI and LLM Foundations
Introduction to Generative AI
Introduction to Large Language Models
Transformers Part-1
Transformers Part-2
Prompt Engineering
Customize LLMs with your data

6. OCI AI Portfolio
AI Services Overview
ML Services Overview
AI Infrastructure
GPUs and Superclusters in OCI
Responsible AI

7. OCI Generative AI Service
OCI Generative AI
Vector Search
Select AI

8. OCI AI Services
Language Intro
Speech Intro
Vision Intro
Document Understanding

Deep Learning Foundations

Module 4

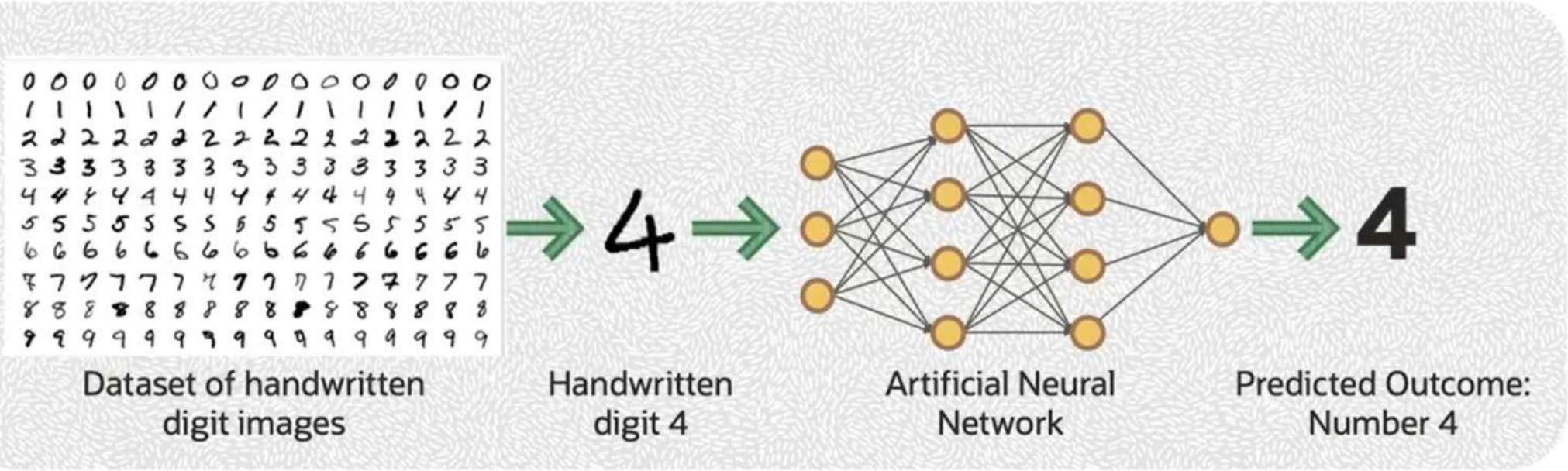
Objectives

- Explain Deep Learning concepts.
- Describe how Convolution Neural Networks work.
- Provide an overview of Sequence Models.

Deep Learning Fundamentals

What is Deep Learning?

- A subset of Machine Learning that focuses on training Artificial Neural Networks (ANNs) with multiple layers
- Allows them to automatically learn and extract intricate representations from data



Why do we need Deep Learning?

- ML needs us to specify features
- DL extracts features from raw and complex data
- Internal representation of data is built using extracted features
- DL algorithms allow parallel processing of data
- That leads to better scalability and performance

Brief History of Deep Learning

1950

Artificial Neuron / Multi-Layer Perceptrons

1980

Multi-Layer Perceptrons with Back Propagation

1990

Convolutional Neural Networks(CNN)

2010

Graphics Processing Units(GPU)

2012

AlexNet, Deep Q-network

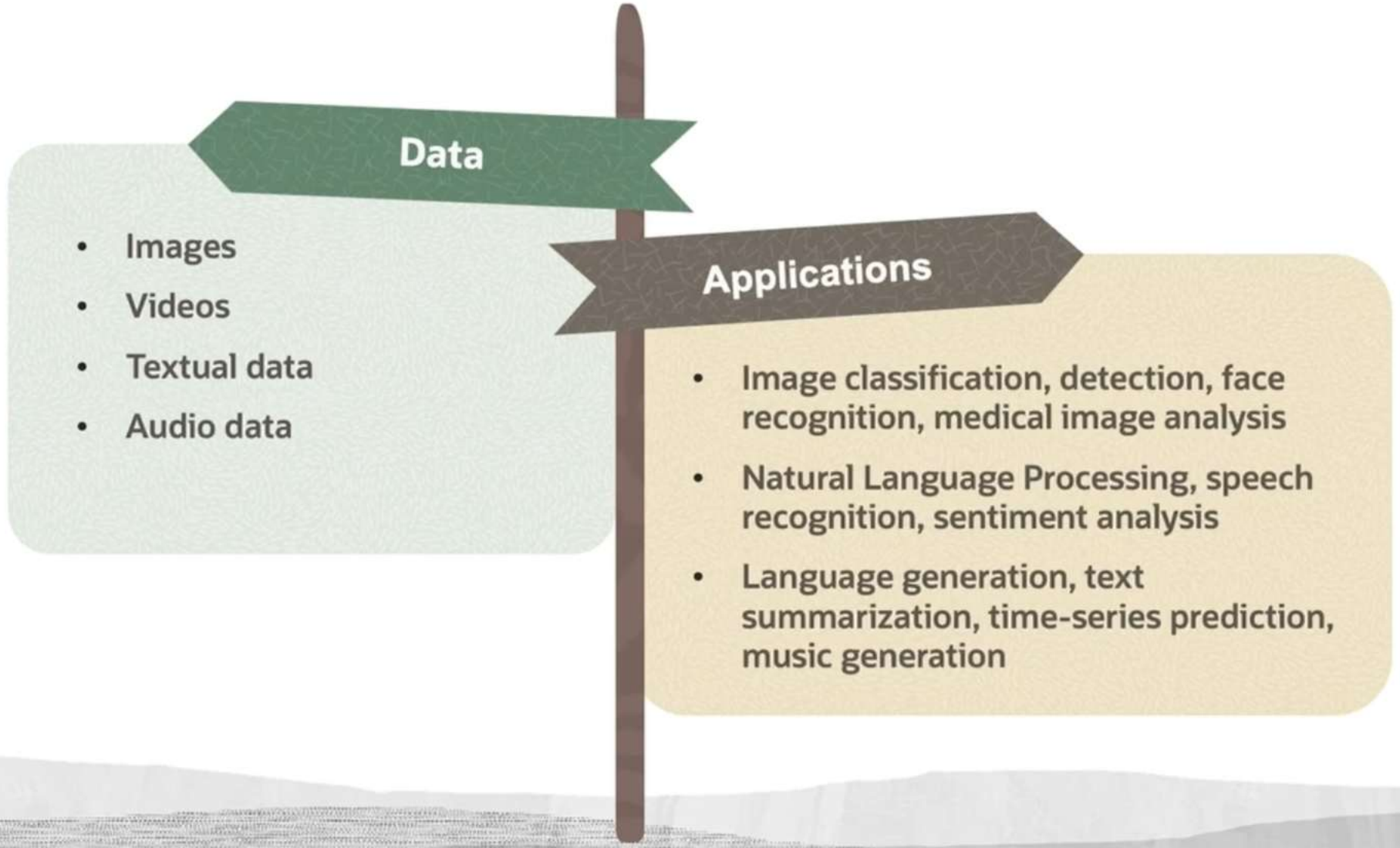
2016/2017

Generative Adversarial Networks (GAN), Transformers

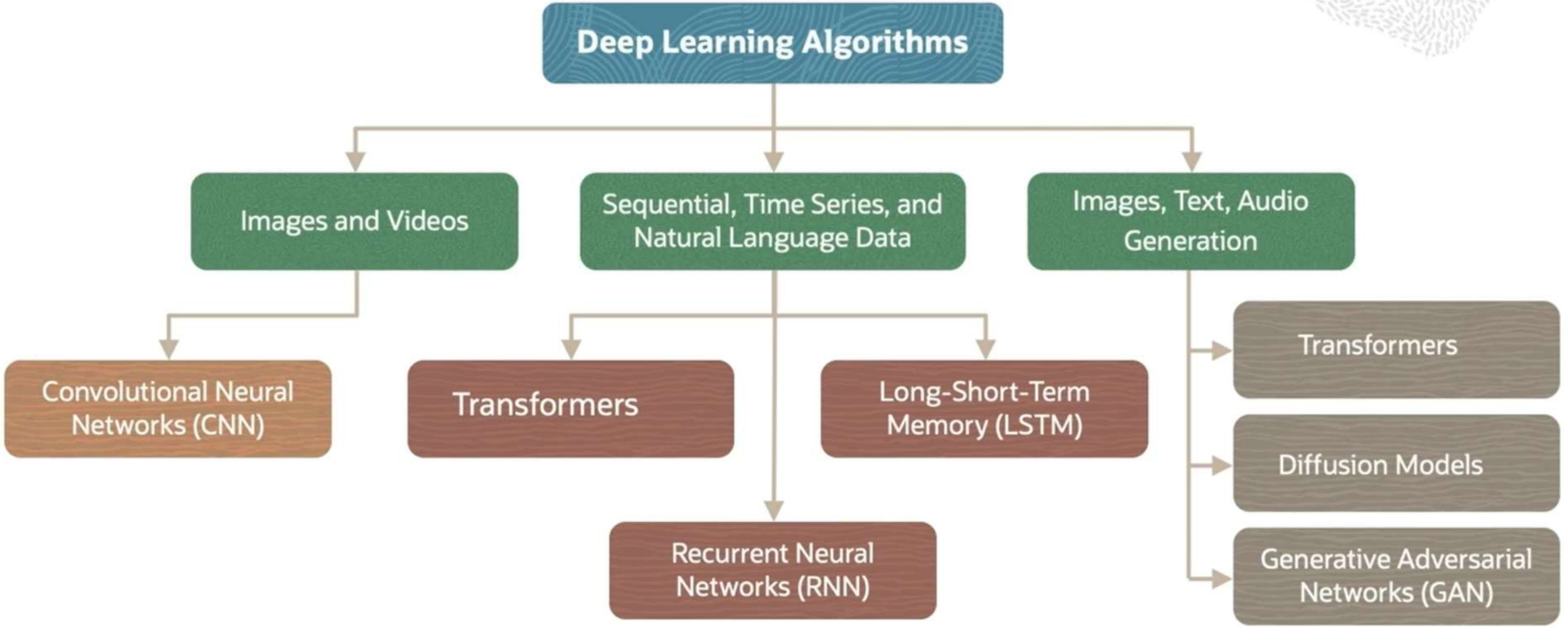
Today...

LLMs, Diffusion models

Types of Deep Learning Algorithms

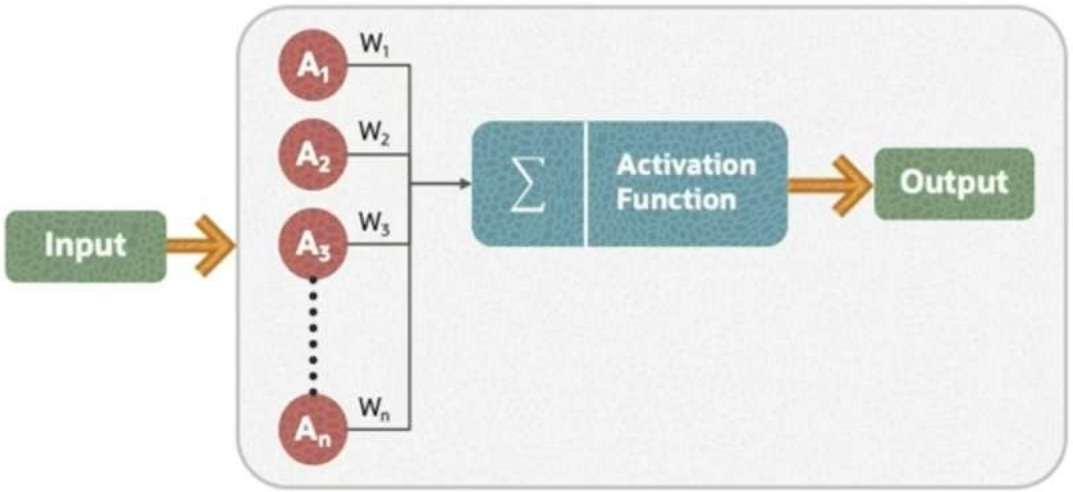


Select the Right Deep Learning Algorithm

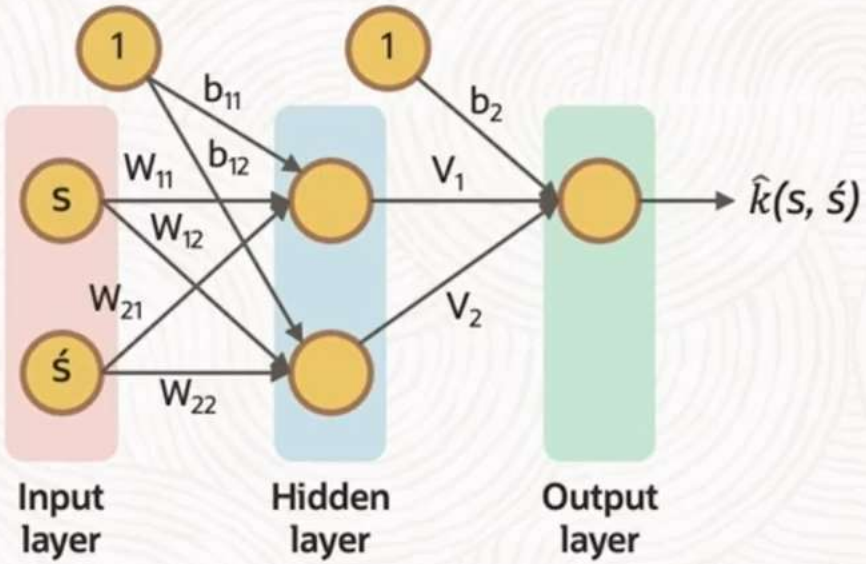


What is Artificial Neural Network (ANN)?

- Inspired by the human brain
- Consists of interconnected nodes called neurons
- Processes information



Building Blocks of ANN



Layers

Input, hidden, and output layers receive inputs, transform it, and produce output

Neurons

Computational unit from an input and produces an output

Weights

Determines the strength of connection between neurons

Activation Function

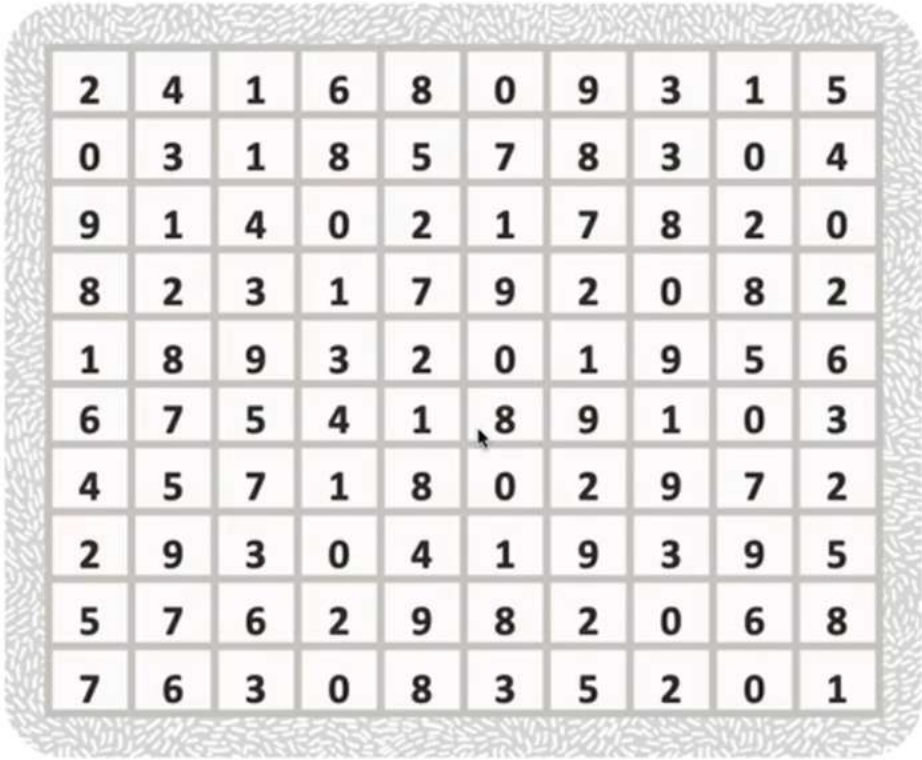
Works on the weighted sum of inputs to a neuron and produces an output

Bias

Additional input to a neuron that allows certain degree of flexibility

Handwritten Character Recognition

- Gathers large data set containing images of handwritten digits as training set
- ANN automatically infers rules for recognizing handwritten digits



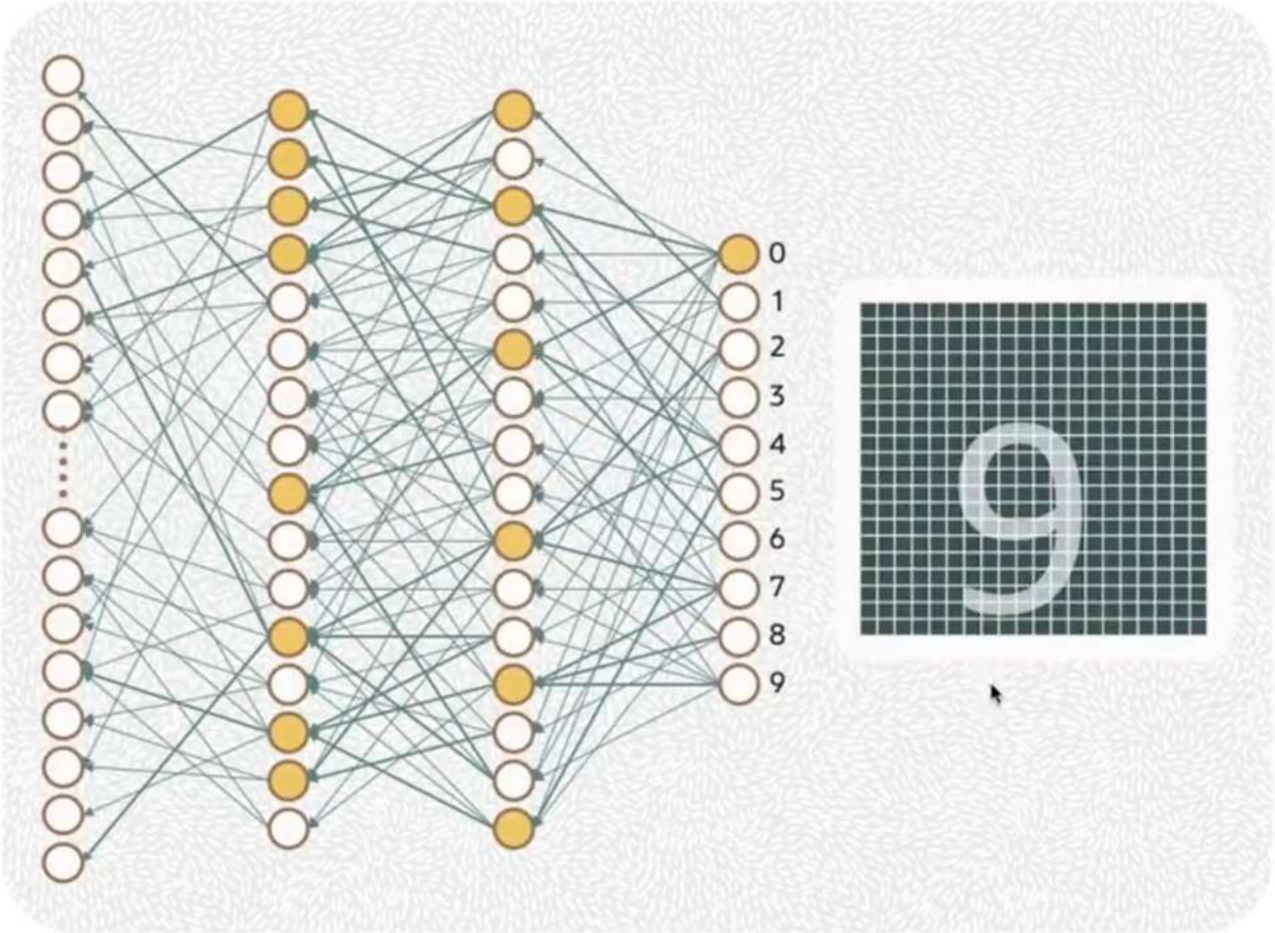
Network Architecture of Handwritten Character Recognition

The handwritten digits get converted to images of 28*28 pixels.

The output layer has 10 neurons to represent the 10 digits from 0 to 9.

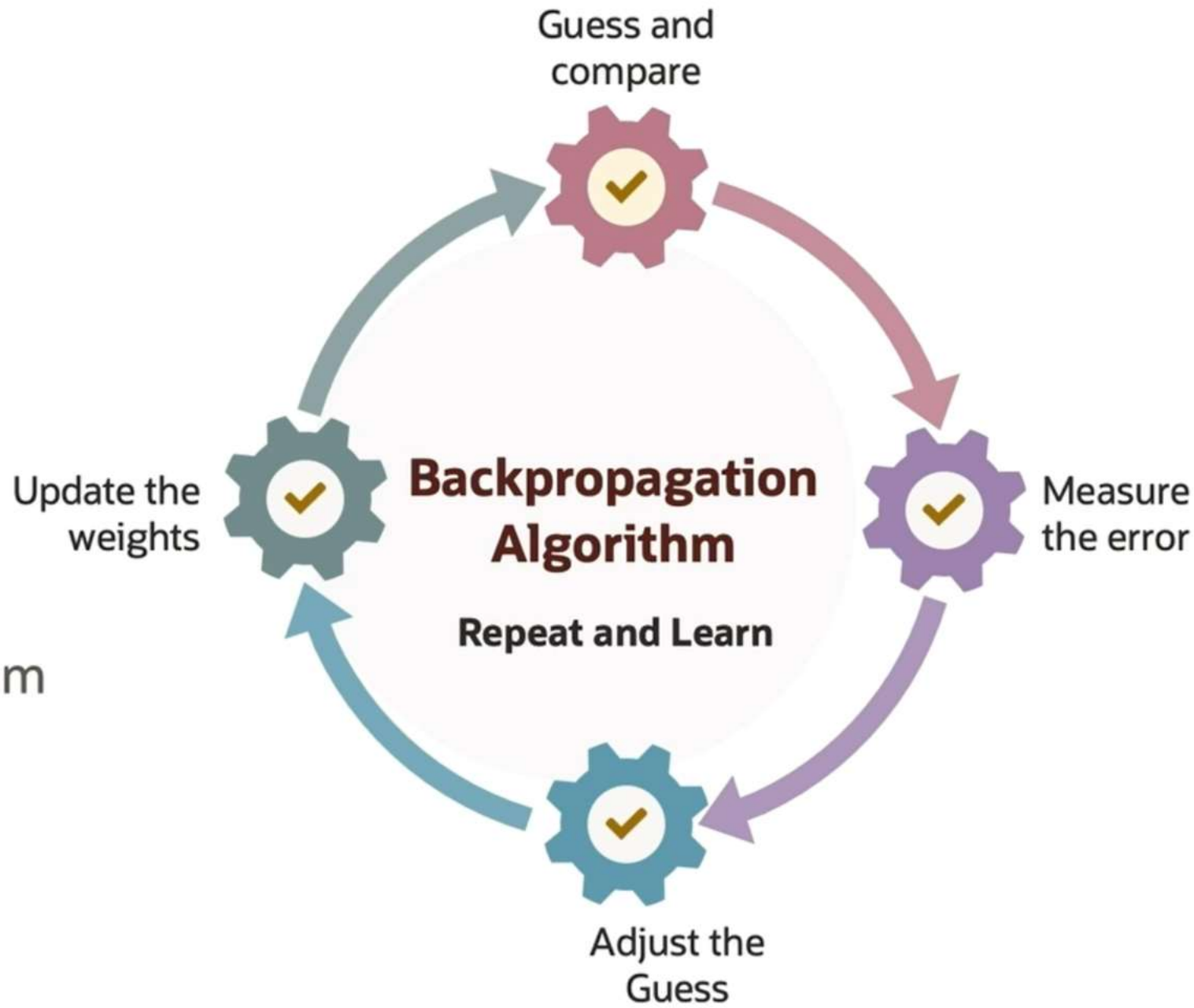
There are two hidden layers of neurons, each containing 16 neurons.

Layers are used for identifying the shapes that make up a handwritten digit.

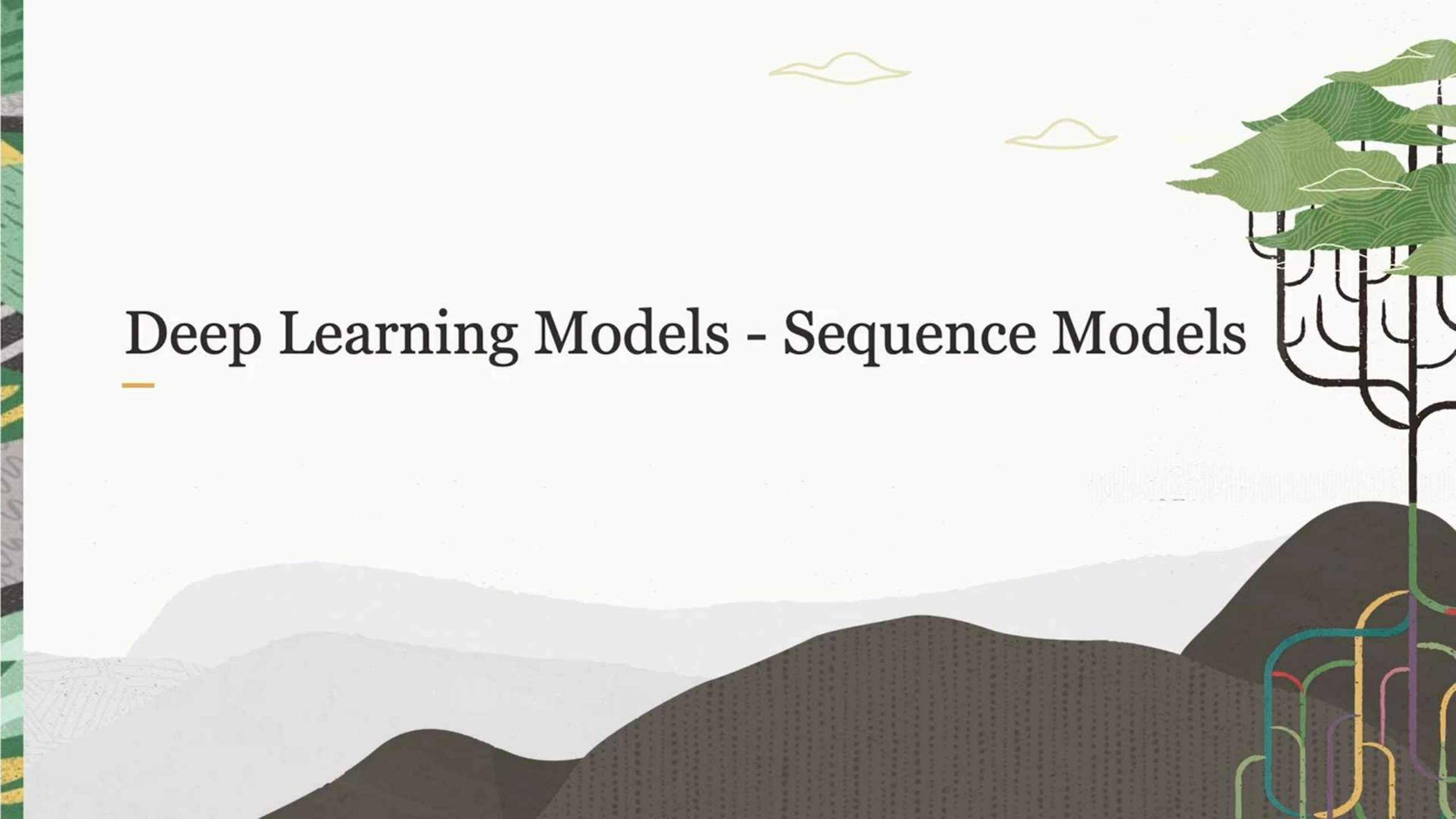


How are ANNs trained?

Learn using backpropagation algorithm

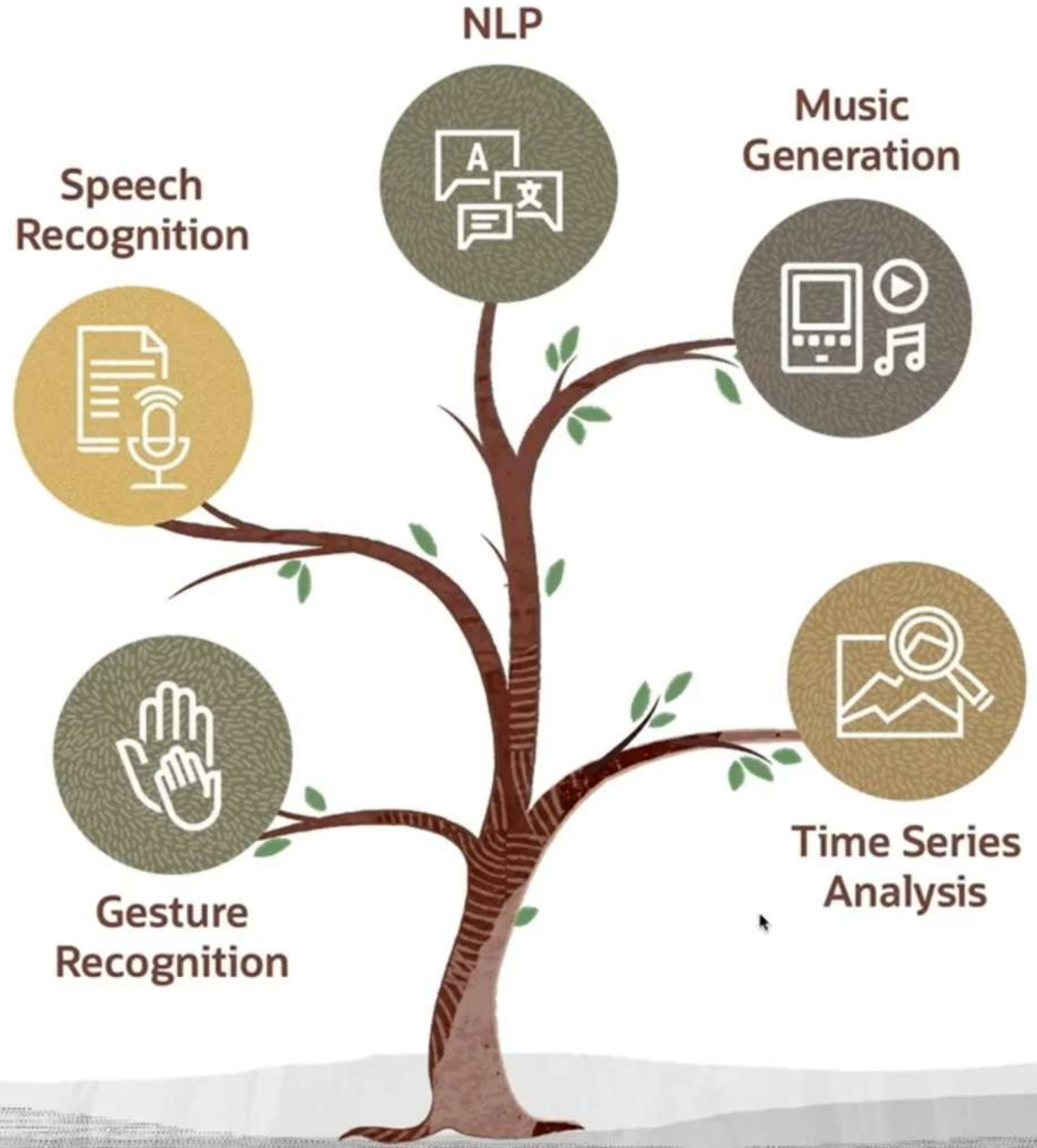


Deep Learning Models - Sequence Models



Sequence Models

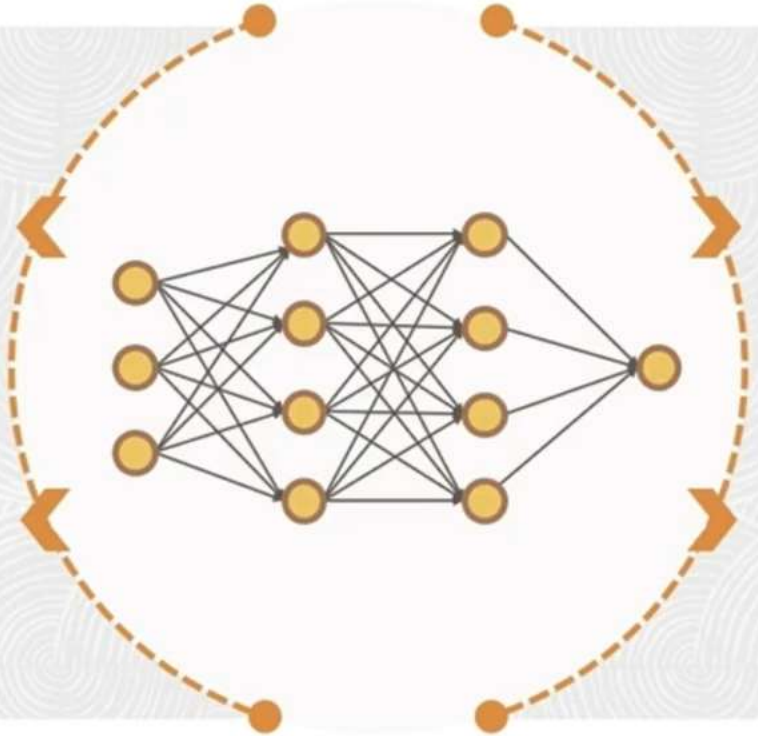
Input data is in the form of sequences. The goal is to find patterns and make predictions.



What is Recurrent Neural Network (RNN)?

Handles sequential data

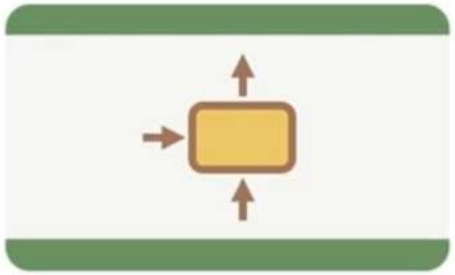
Maintains a hidden state or memory



Allows information to persist using a feedback loop

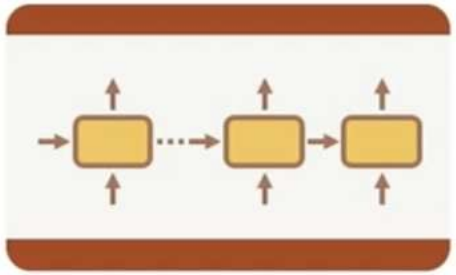
Captures dependencies

Types of RNN Architecture



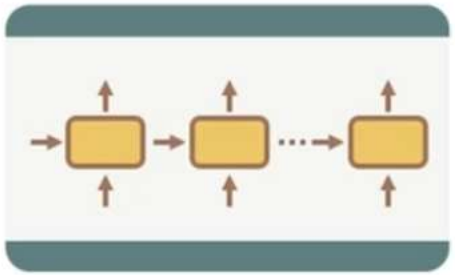
One-to-one

Standard non-sequential data like FNN



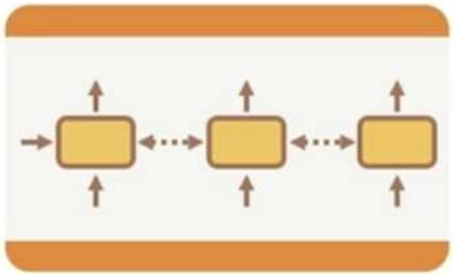
One-to-many

Music generation or sequence generation



Many-to-one

Sentiment analysis based on reviews



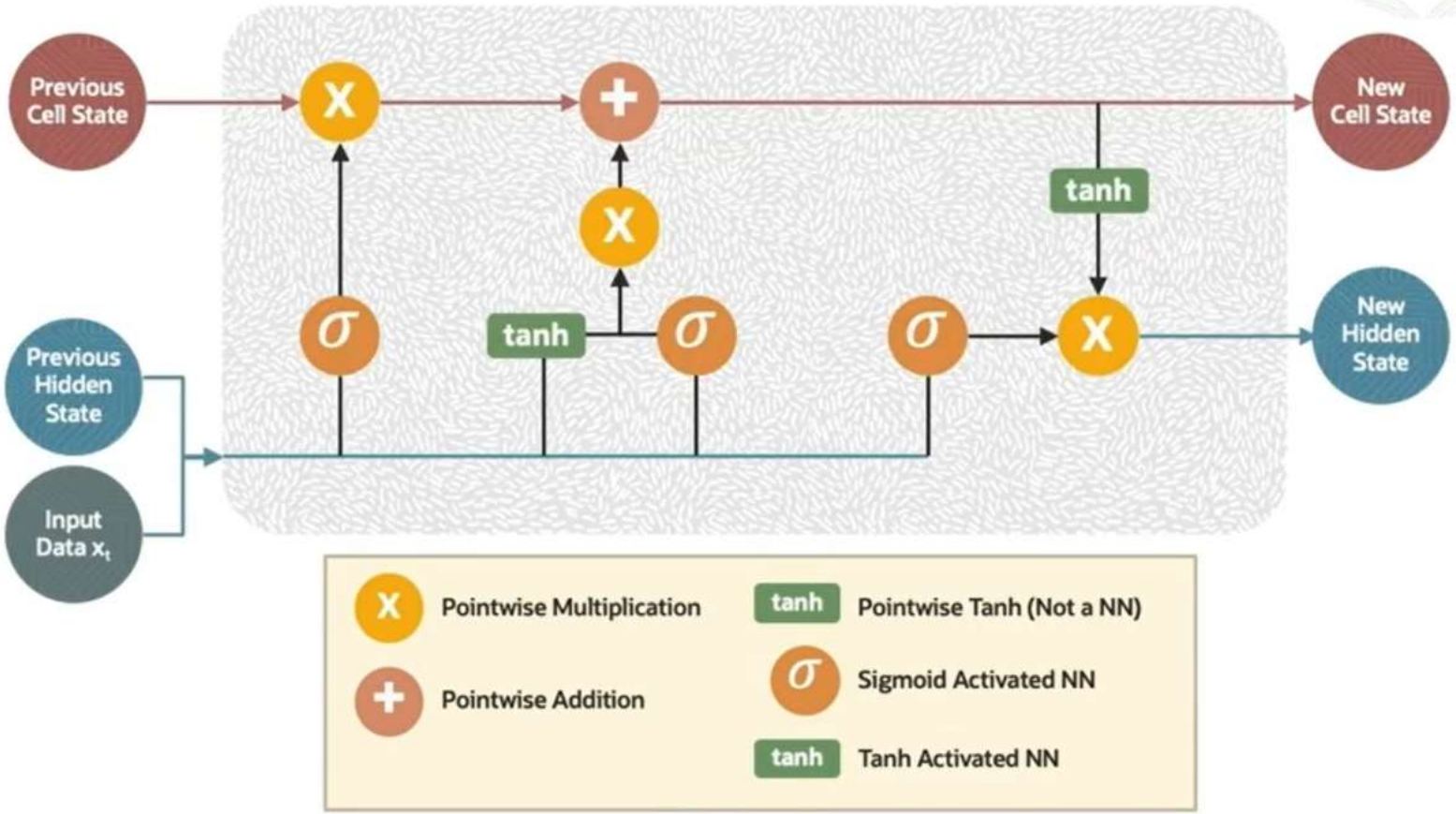
Many-to-many

Machine translation and named entity recognition

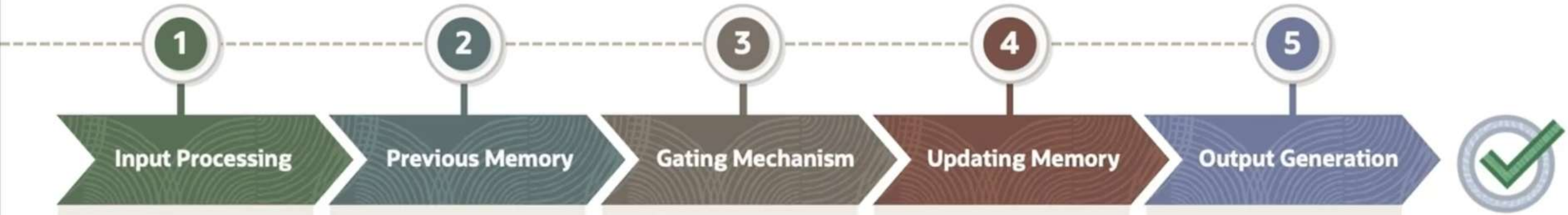
What is Long Short-Term Memory?

Works by using a specialized memory cell and gating mechanisms to capture long-term dependencies in sequential data

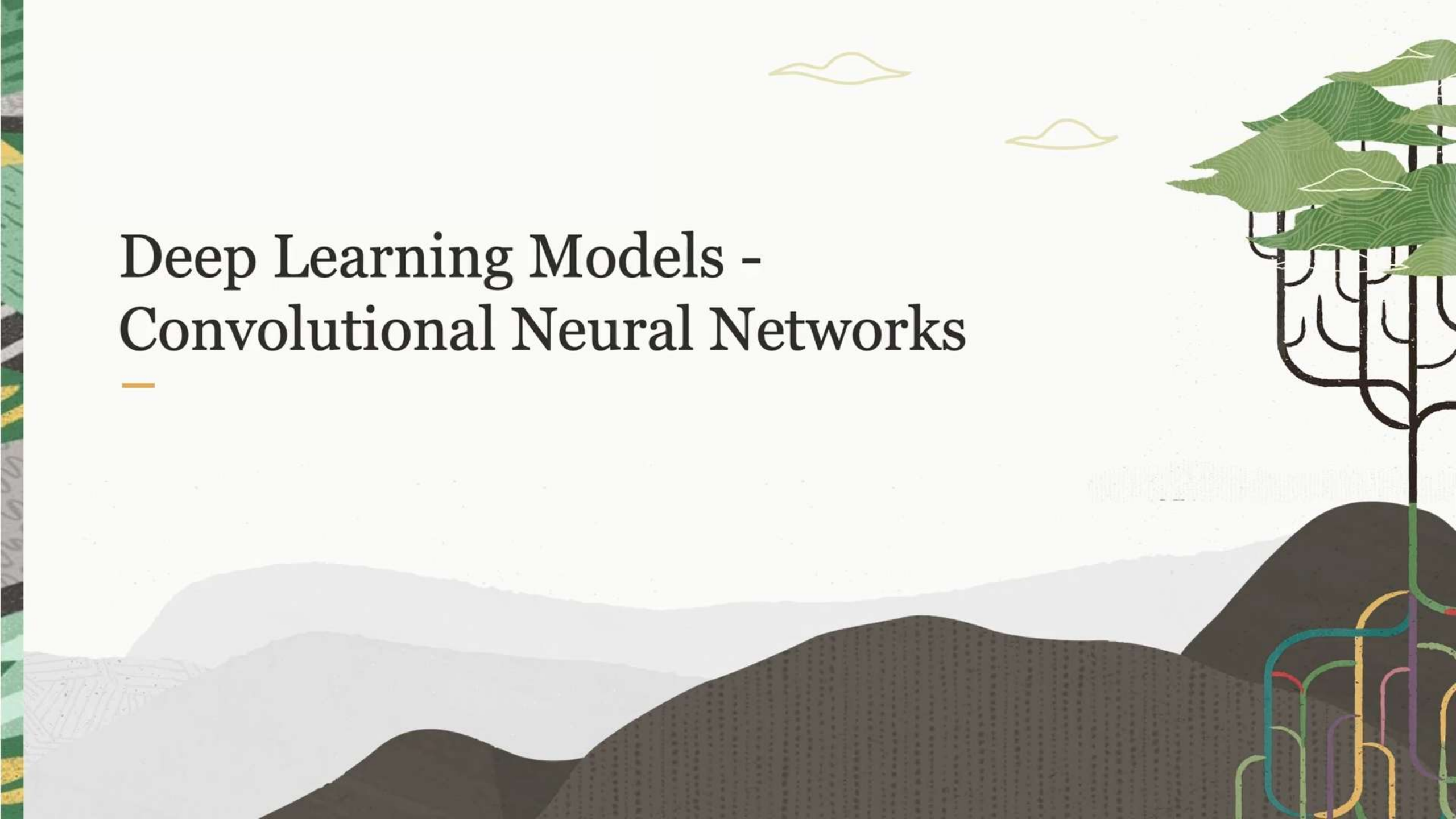
Selectively remembers or forgets information over time



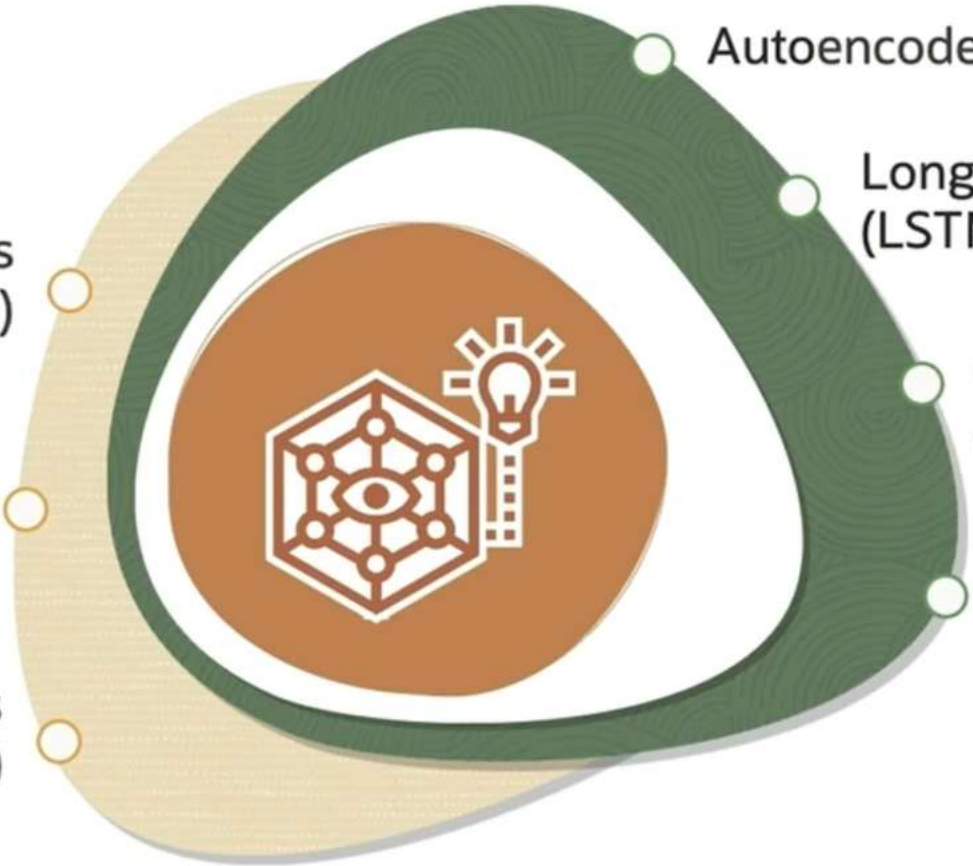
Step-by-Step Working of LSTM



Deep Learning Models - Convolutional Neural Networks



Deep Learning Models

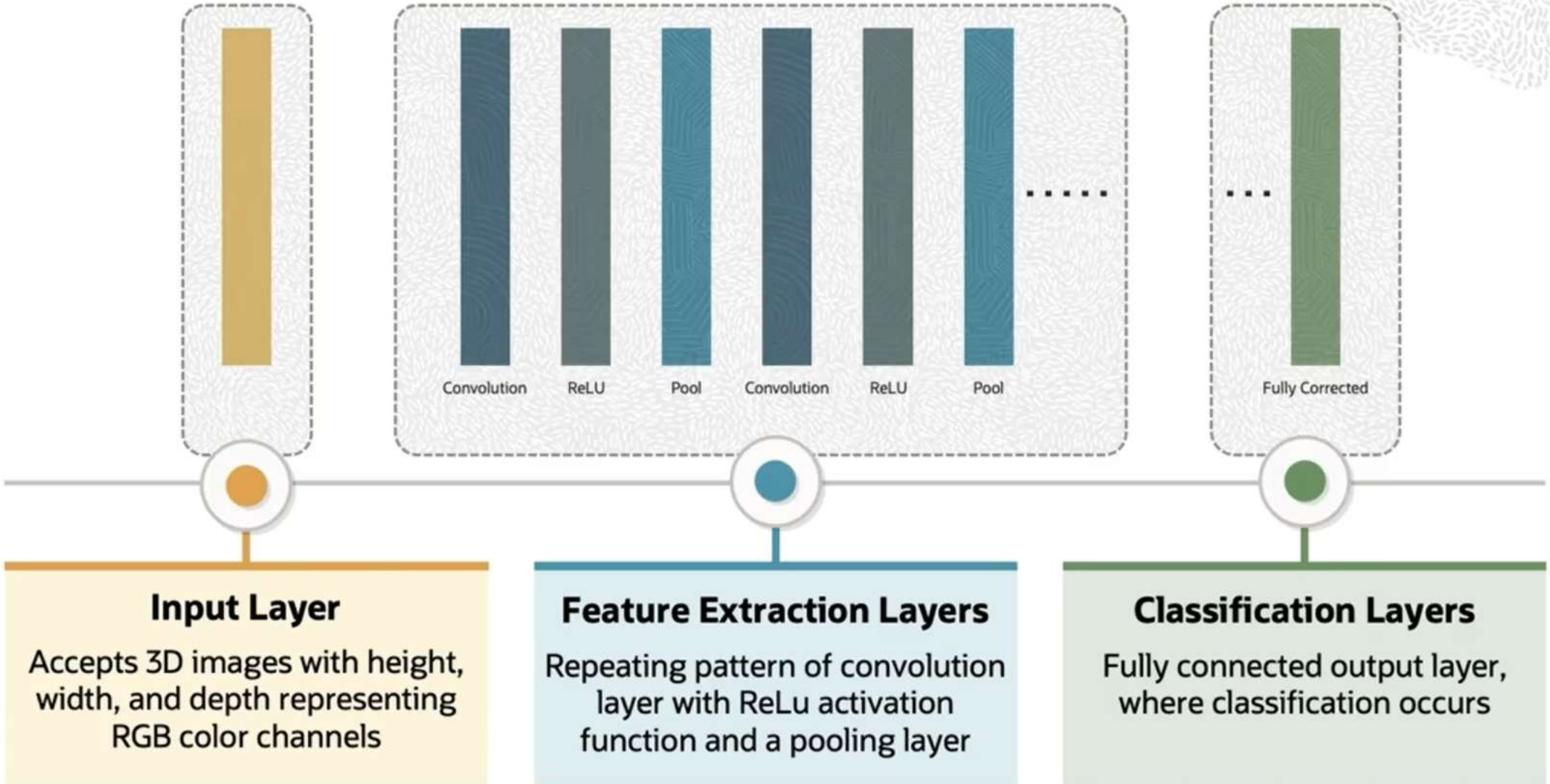


- Autoencoders
- Long Short-Term Memory (LSTM)
- Generative Adversarial Networks (GAN)
- Transformers
- Feed Forward Neural Networks (FNN)
- Convolution Neural Networks (CNN)
- Recurrent Neural Networks (RNN)

What is a Convolution Neural Network (CNN)?

- A type of deep learning model designed for processing grid-like data
- Reduces images into an easier-to-process form

CNN Layers - Overview



Input Layer

Accepts 3D images with height, width, and depth representing RGB color channels

Feature Extraction Layers

Repeating pattern of convolution layer with ReLu activation function and a pooling layer

Classification Layers

Fully connected output layer, where classification occurs

Robotic House Inspection

Blueprint Detector → Pattern Highlighter → Room Summarizer → House Expert → Guess Maker → Quality Checker

Feature Extraction Layers

Robotic House Inspection

- Blueprint Detector
- Pattern Highlighter
- Room Summarizer
- House Expert
- Guess Maker
- Quality Checker

Feature Extraction Layers

- Convolutional Layer
- Activation Function
- Pooling Layer
- Fully Connected Layer
- Softmax Layer
- Dropout Layer

Feature Extraction Layers

Automatically learns and extracts relevant patterns and features from the input images

Convolution Layer

Applies convolutional operations to the input image using small filters known as kernels

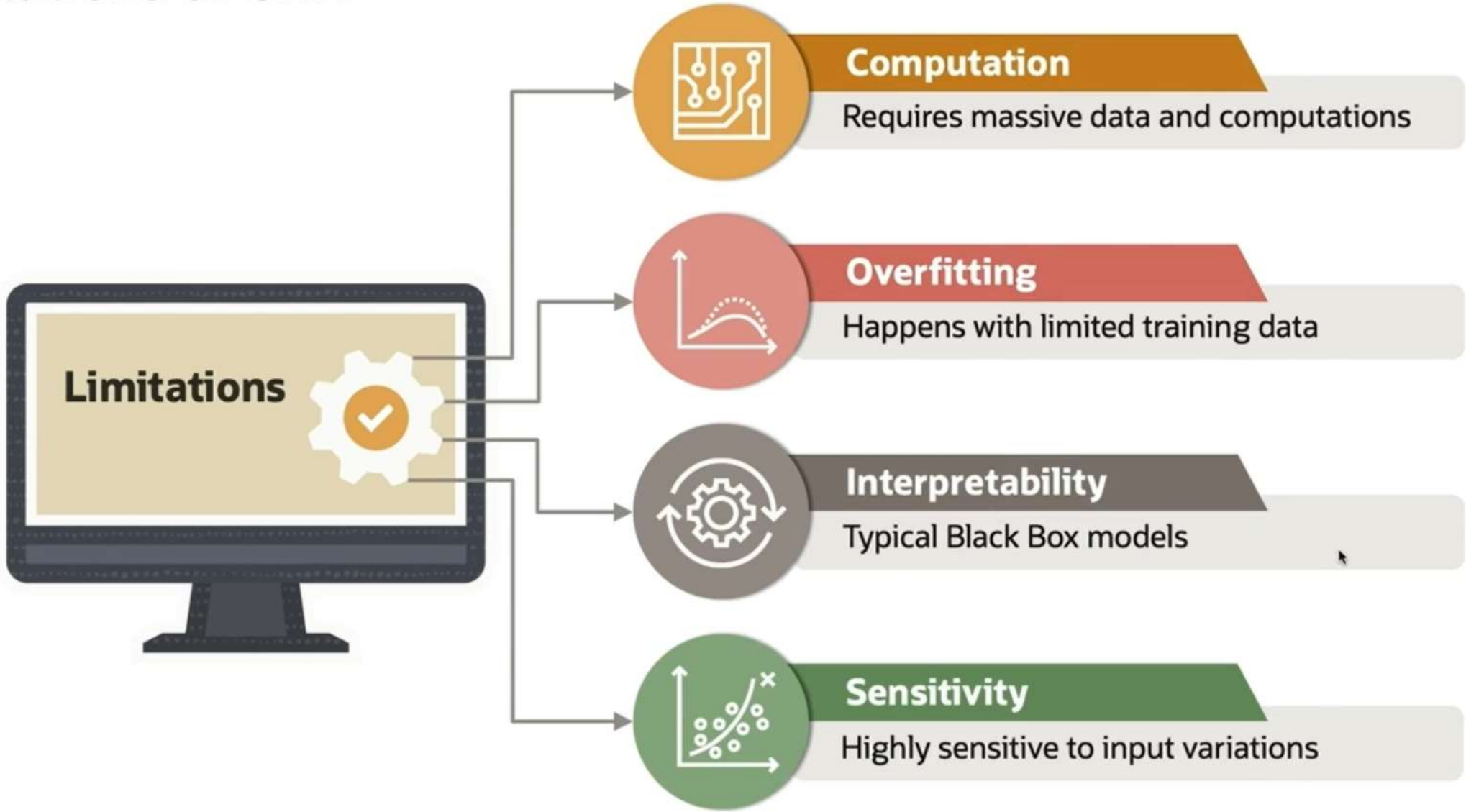
Activation Layer

Allows the network to learn more complex and non-linear relationships in the data

Pooling Layer

Helps reduce computational complexity and also the spatial dimensions of the feature maps generated by the convolutional layers

Limitations of CNN



Applications of CNN

Demo

Classification with Multilayer Perceptron

File Edit View Run Kernel Tabs Settings Help

demo-deep-learning.ipynb

Code

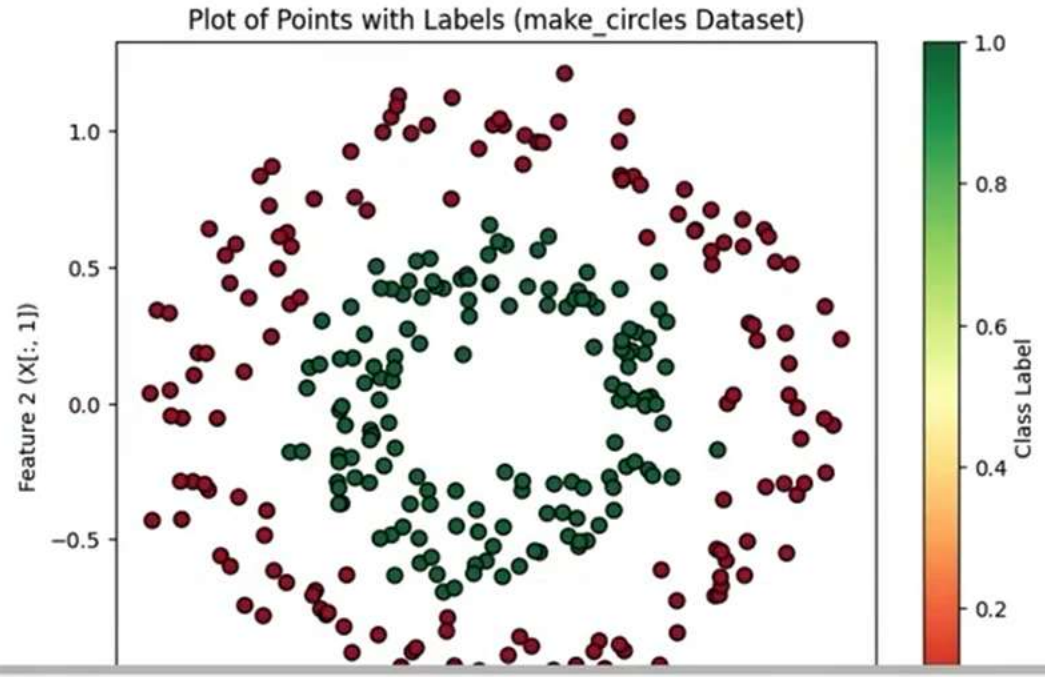
Notebook Python 3 (ipykernel)

Demo - Classification with Multilayer Perceptron

[15]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_circles

# Generate synthetic data (circles)
X, y = make_circles(n_samples=300, noise=0.1, factor=0.5, random_state=0)

# Plot the points with their labels
plt.figure(figsize=(8, 6))
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.RdYlGn, edgecolors='k', marker='o', s=50)
plt.xlabel('Feature 1 (X[:, 0])')
plt.ylabel('Feature 2 (X[:, 1])')
plt.title('Plot of Points with Labels (make_circles Dataset)')
plt.colorbar(label='Class Label')
plt.show()
```



Simple 0 4 Python 3 (ipykernel) | Idle Mode: Command Ln 1, Col 1 demo-deep-learning.ipynb 1

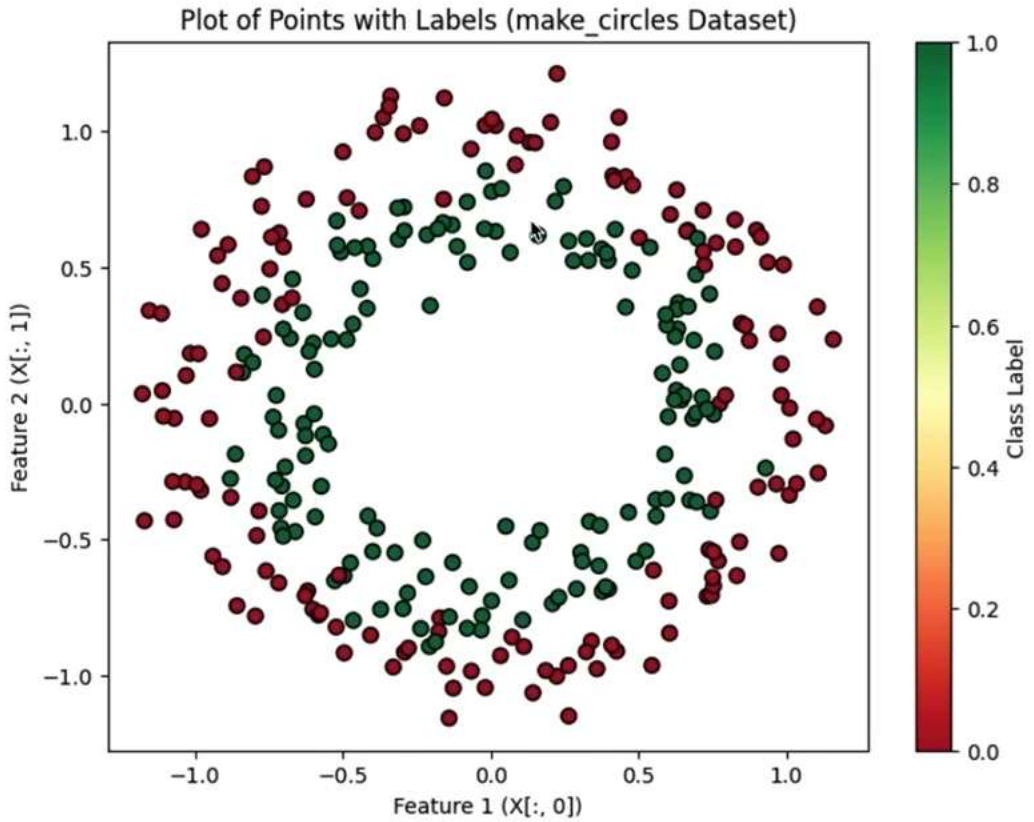
File Edit View Run Kernel Tabs Settings Help

demo-deep-learning.ipynb

Code

Notebook Python 3 (ipykernel)

```
plt.figure(figsize=(8, 6))
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.RdYlGn, edgecolors='k', marker='o', s=50)
plt.xlabel('Feature 1 (X[:, 0])')
plt.ylabel('Feature 2 (X[:, 1])')
plt.title('Plot of Points with Labels (make_circles Dataset)')
plt.colorbar(label='Class Label')
plt.show()
```



[16]:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_circles
from sklearn.neural_network import MLPClassifier
import ipywidgets as widgets
```

Simple 0 4 Python 3 (ipykernel) | Idle Mode: Command Ln 1, Col 1 demo-deep-learning.ipynb 1

File Edit View Run Kernel Tabs Settings Help

demo-deep-learning.ipynb

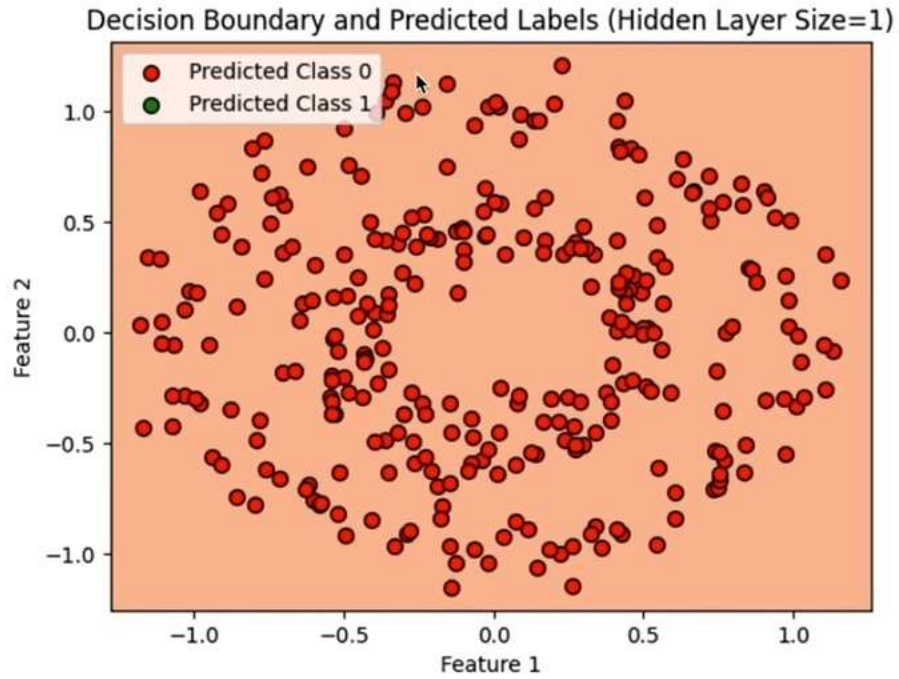
Code

Notebook

Python 3 (ipykernel)

```
# Display the widgets
display(interactive_plot)
```

Hidden Lay... 1



[ ]:

Simple 0 4 Python 3 (ipykernel) | Idle

Mode: Command Ln 1, Col 1 demo-deep-learning.ipynb 1

File Edit View Run Kernel Tabs Settings Help

demo-deep-learning.ipynb

Code

Notebook Python 3 (ipykernel)

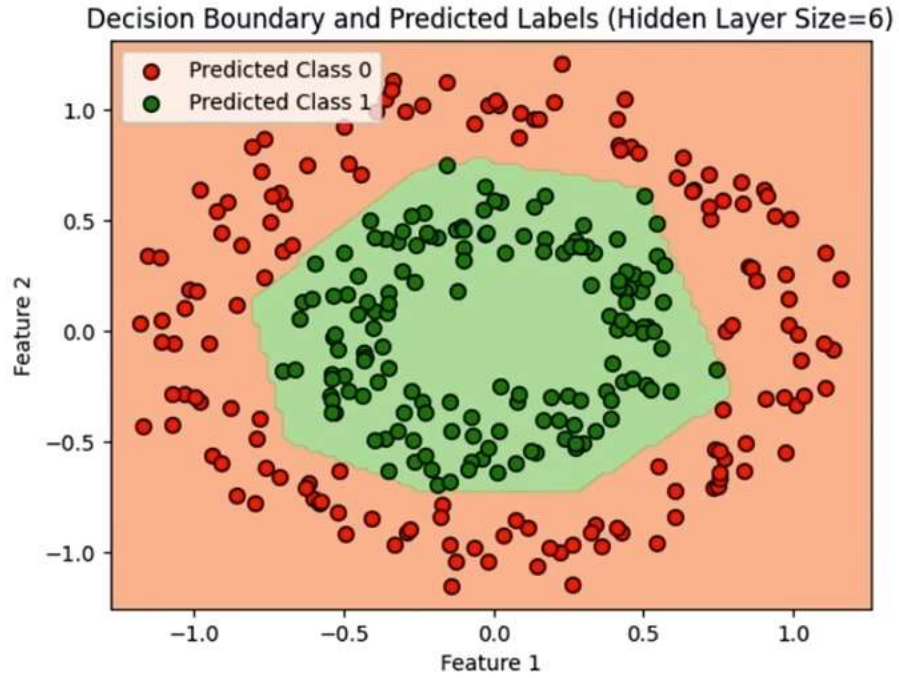
```
    plt.ylabel('Feature 2')
    plt.title(f'Decision Boundary and Predicted Labels (Hidden Layer Size={hidden_layer_size})')
    plt.legend()
    plt.show()

# Create a slider for hidden layer size
hidden_layer_size_slider = widgets.IntSlider(value=1, min=1, max=10, step=1, description='Hidden Layer Size')

# Create an interactive widget
interactive_plot = interactive(update_plot, hidden_layer_size=hidden_layer_size_slider)

# Display the widgets
display(interactive_plot)
```

Hidden Lay... 6



[ ]:

Simple 0 4 Python 3 (ipykernel) | Idle Mode: Command Ln 21, Col 5 demo-deep-learning.ipynb 1

File Edit View Run Kernel Tabs Settings Help

demo-deep-learning.ipynb

Code

Notebook Python 3 (ipykernel)

[22]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_circles
from sklearn.neural_network import MLPClassifier
import ipywidgets as widgets
from IPython.display import display
from ipywidgets import interactive

# Function to update and display the plot
def update_plot(hidden_layer_size):
    # Generate synthetic data (circle)
    #X, y = make_circles(n_samples=300, noise=0.1, factor=0.5, random_state=0)

    # Create a multi-layer perceptron (MLP) classifier
    clf = MLPClassifier(hidden_layer_sizes=(hidden_layer_size,),
                        activation='relu', max_iter=3000, random_state=1)

    # Fit the classifier to the data
    clf.fit(X, y)

    # Create a grid of points for visualization
    #These are 1D arrays of 100 values each, representing the x and y coordinates of the grid.
    x_vals = np.linspace(X[:, 0].min() - 0.1, X[:, 0].max() + 0.1, 100)
    y_vals = np.linspace(X[:, 1].min() - 0.1, X[:, 1].max() + 0.1, 100)

    #The resulting X_plane and Y_plane are both 100x100 arrays,
    #representing a grid of 10,000 points.
    X_plane, Y_plane = np.meshgrid(x_vals, y_vals)

    #grid_points is a single 2D array (grid_points) of shape (10000, 2),
    #where each row represents a point in the grid.
    grid_points = np.column_stack((X_plane.ravel(), Y_plane.ravel()))

    # Predict class labels for the grid points (for decision boundary)
    Z = clf.predict(grid_points)

    #Z.reshape(X_plane.shape) reshapes Z into a 100x100 array.
    Z = Z.reshape(X_plane.shape)

    # Predict class labels for the original data points
    y_pred = clf.predict(X)
```

Simple 0 4 Python 3 (ipykernel) | Idle Mode: Command Ln 21, Col 5 demo-deep-learning.ipynb 1

File Edit View Run Kernel Tabs Settings Help

demo-deep-learning.ipynb

Code

Notebook Python 3 (ipykernel)

```
    # Predict class labels for the grid points (for decision boundary)
    Z = clf.predict(grid_points)

    #Z.reshape(X_plane.shape) reshapes Z into a 100x100 array.
    Z = Z.reshape(X_plane.shape)

    # Predict class labels for the original data points
    y_pred = clf.predict(X)

    # Clear previous plot
    plt.clf()

    # Plot the decision boundary
    #it is often used to visualize the decision boundary of a model by plotting
    #the predicted class probabilities or labels across a grid of points.
    plt.contourf(X_plane, Y_plane, Z, levels=[-0.5, 0.5, 1.5], cmap=plt.cm.RdYlGn, alpha=0.6)

    # Plot the original data points with their predicted labels
    # Separate points for each predicted class
    class_0 = y_pred == 0  # Indices of points predicted as class 0
    class_1 = y_pred == 1  # Indices of points predicted as class 1

    plt.scatter(X[class_0, 0], X[class_0, 1], c='red', edgecolors='k', marker='o', s=50, label='Predicted Class 0')
    plt.scatter(X[class_1, 0], X[class_1, 1], c='green', edgecolors='k', marker='o', s=50, label='Predicted Class 1')

    # Add labels and title
    plt.xlabel('Feature 1')
    plt.ylabel('Feature 2')
    plt.title(f'Decision Boundary and Predicted Labels (Hidden Layer Size={hidden_layer_size})')
    plt.legend()
    plt.show()

# Create a slider for hidden layer size
hidden_layer_size_slider = widgets.IntSlider(value=1, min=1, max=10, step=1, description='Hidden Layer Size')

# Create an interactive widget
interactive_plot = interactive(update_plot, hidden_layer_size=hidden_layer_size_slider)

# Display the widgets
display(interactive_plot)
```

Hidden Lay... 6

Simple 0 4 Python 3 (ipykernel) | Idle Mode: Edit Ln 64, Col 15 demo-deep-learning.ipynb 1

1. Welcome to AI Foundations
Welcome to AI Foundations

2. AI Foundations
Introduction to AI
AI - Tasks and Data
AI vs ML vs DL

3. Machine Learning Foundations
Introduction to Machine Learning
Supervised Learning - Regression
Supervised Learning - Classification
Unsupervised Learning
Reinforcement Learning

4. Deep Learning Foundations
Introduction to Deep Learning
Deep Learning Models-Sequence Models
Deep Learning Models - CNN

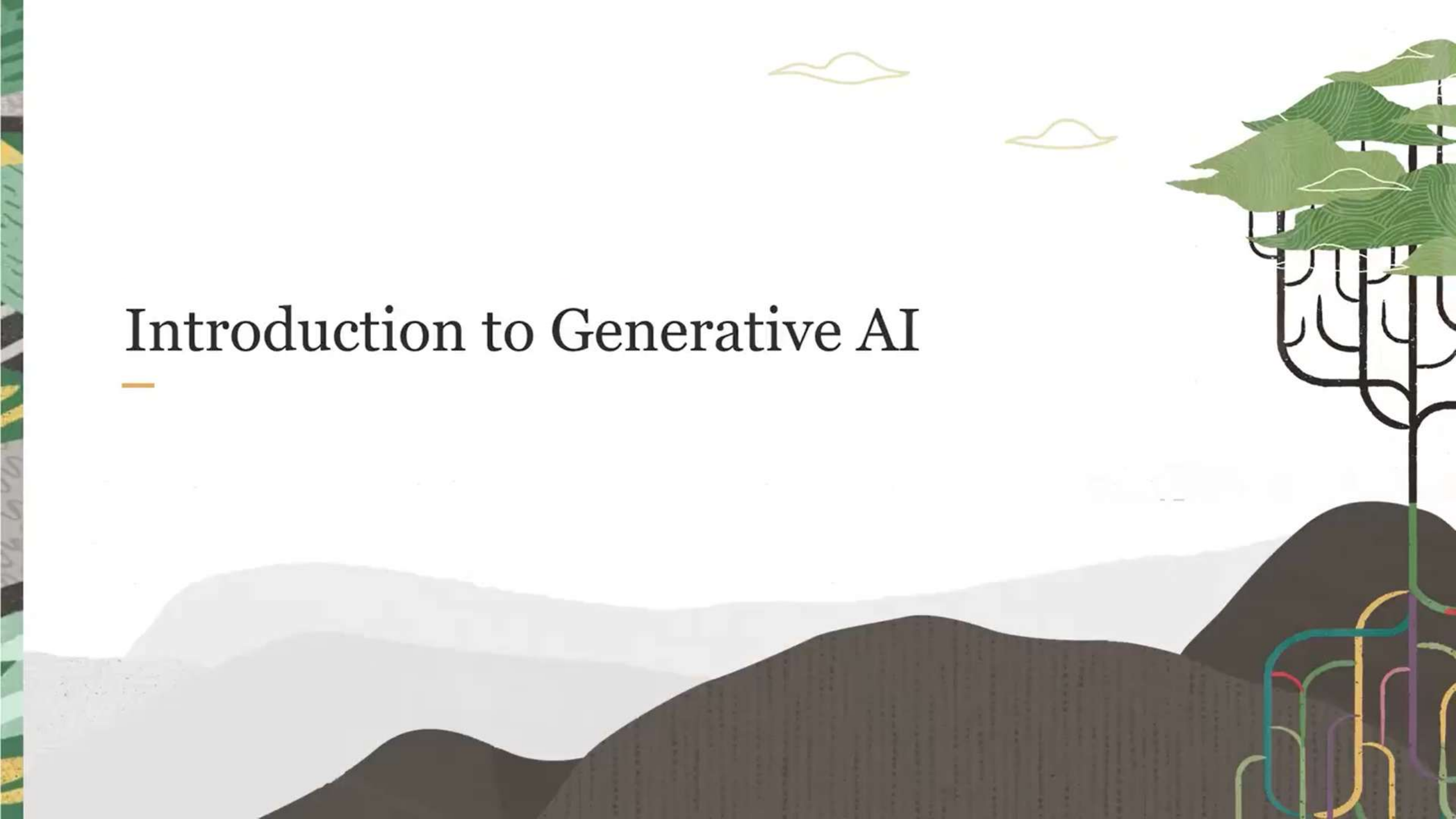
5. Generative AI and LLM Foundations
Introduction to Generative AI
Introduction to Large Language Models
Transformers Part-1
Transformers Part-2
Prompt Engineering
Customize LLMs with your data

6. OCI AI Portfolio
AI Services Overview
ML Services Overview
AI Infrastructure
GPUs and Superclusters in OCI
Responsible AI

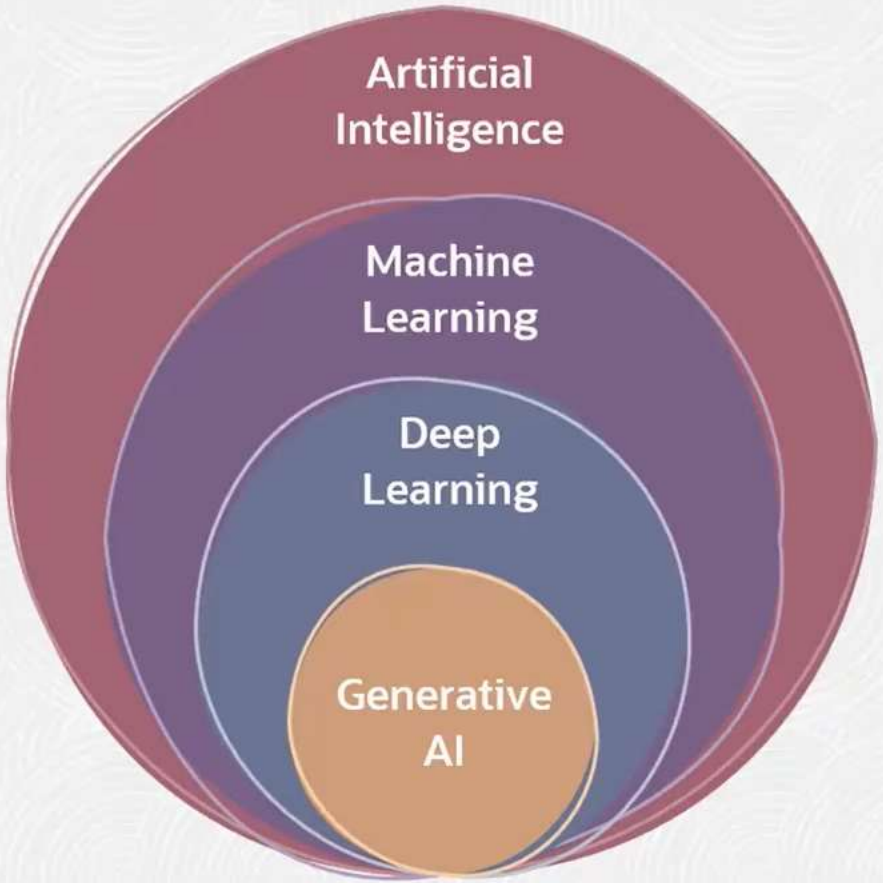
7. OCI Generative AI Service
OCI Generative AI
Vector Search
Select AI

8. OCI AI Services
Language Intro
Speech Intro
Vision Intro
Document Understanding

Introduction to Generative AI



What is Generative AI?

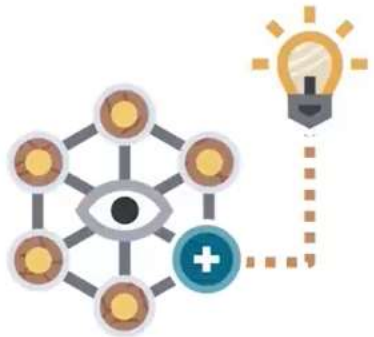


- A type of AI that can create new content.
- Models that can create a wide range of outputs such as text, images, music, videos, and other types of data.
- Subset of Deep Learning where the models are trained to generate output on their own.
- Opens exciting possibilities for creative tasks, automation, and new ideas.

How does Generative AI work?

Learns the underlying patterns in a given data set and uses that knowledge to **create new data** that shares those patterns

Generative AI Model

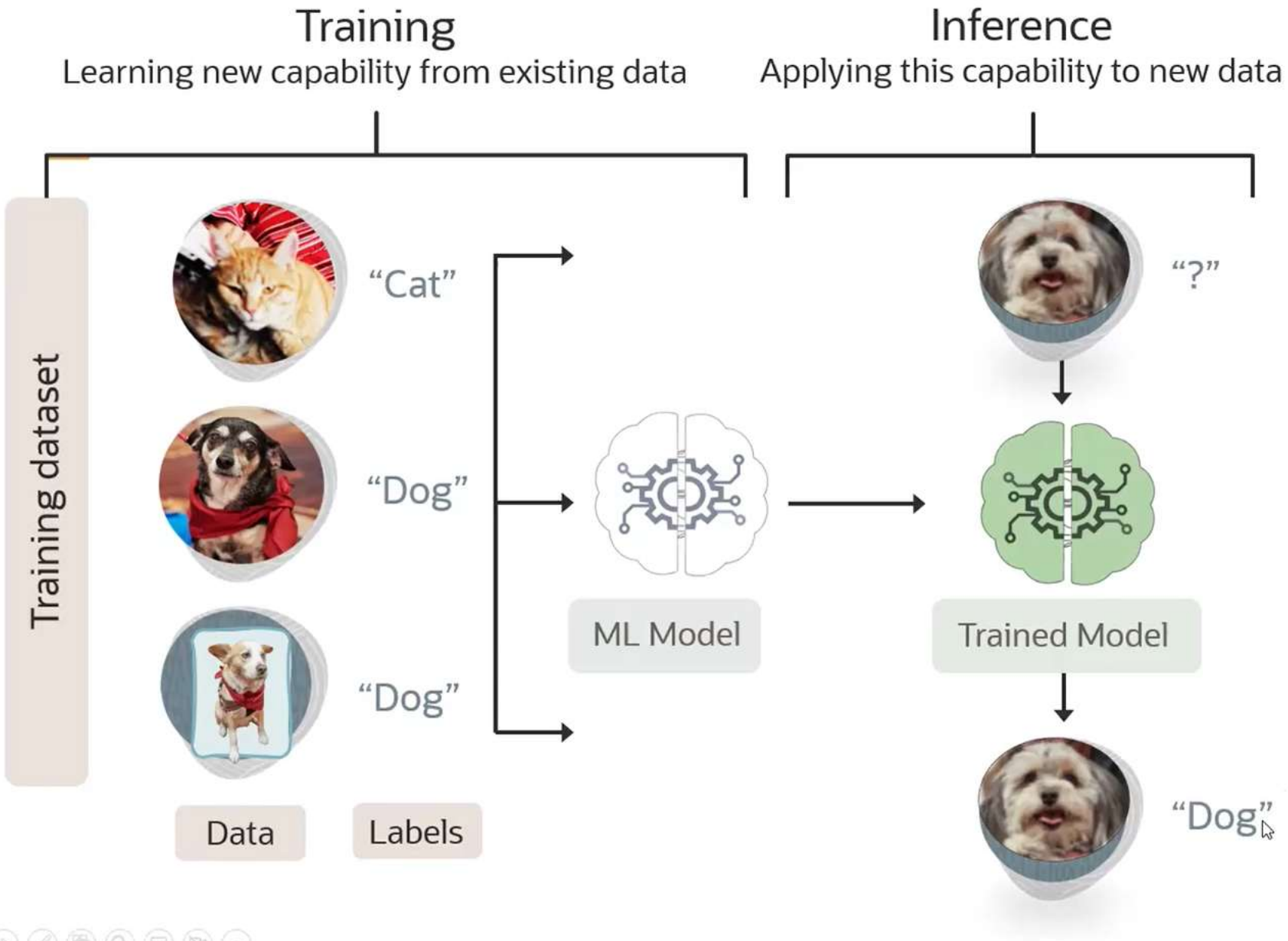


Training Data

Figures out common dog patterns and features

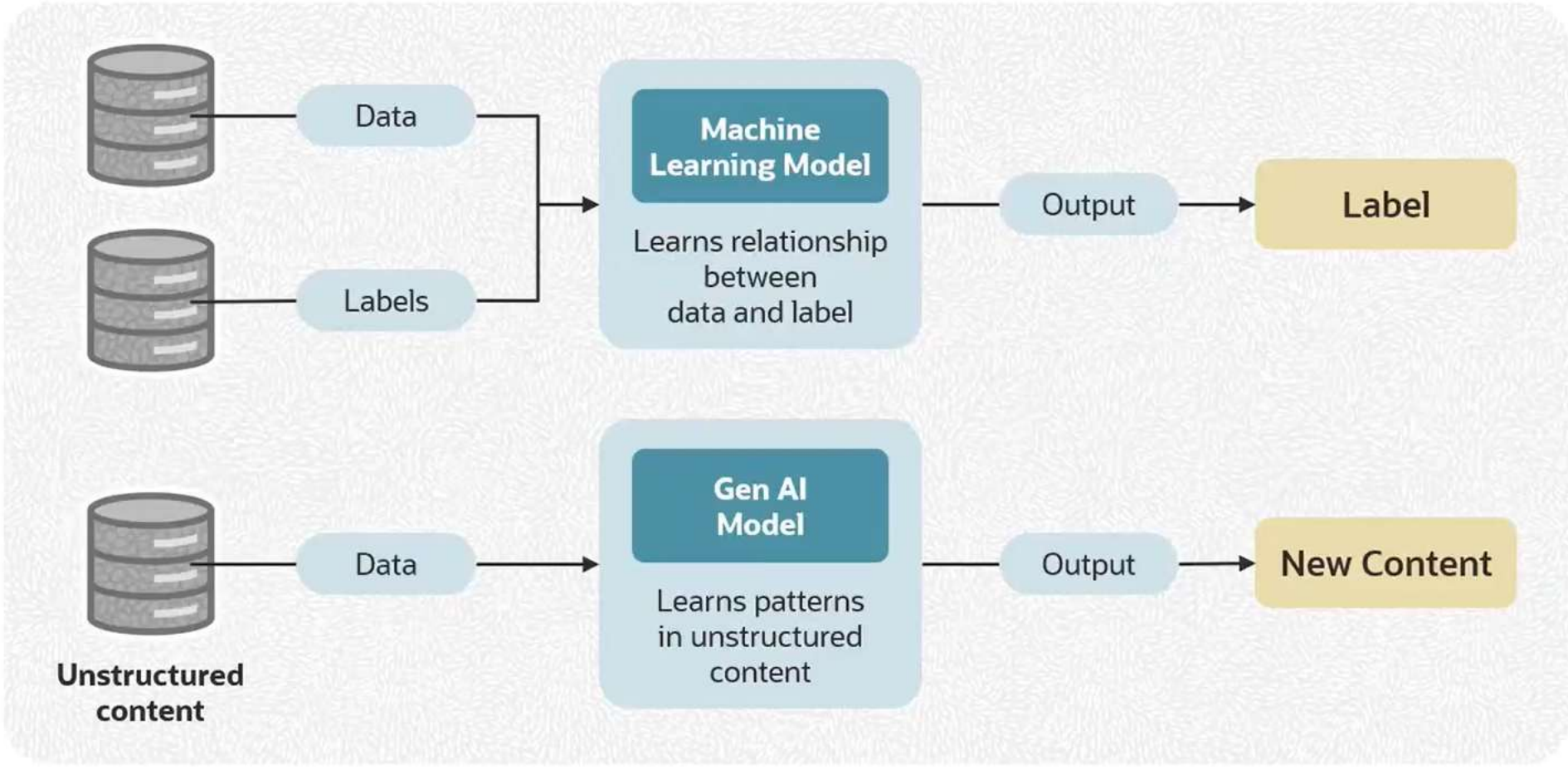
“Draw a picture of a dog”

Machine Learning



Identifies patterns in the pictures and uses those patterns to **recognize** and **classify** new examples of cats and dogs

How is Generative AI different from other AI approaches?



- Classification
- Regression
- Recommendation systems

- Text Generation
- Image Generation
- Video Generation

Types of Generative AI Models

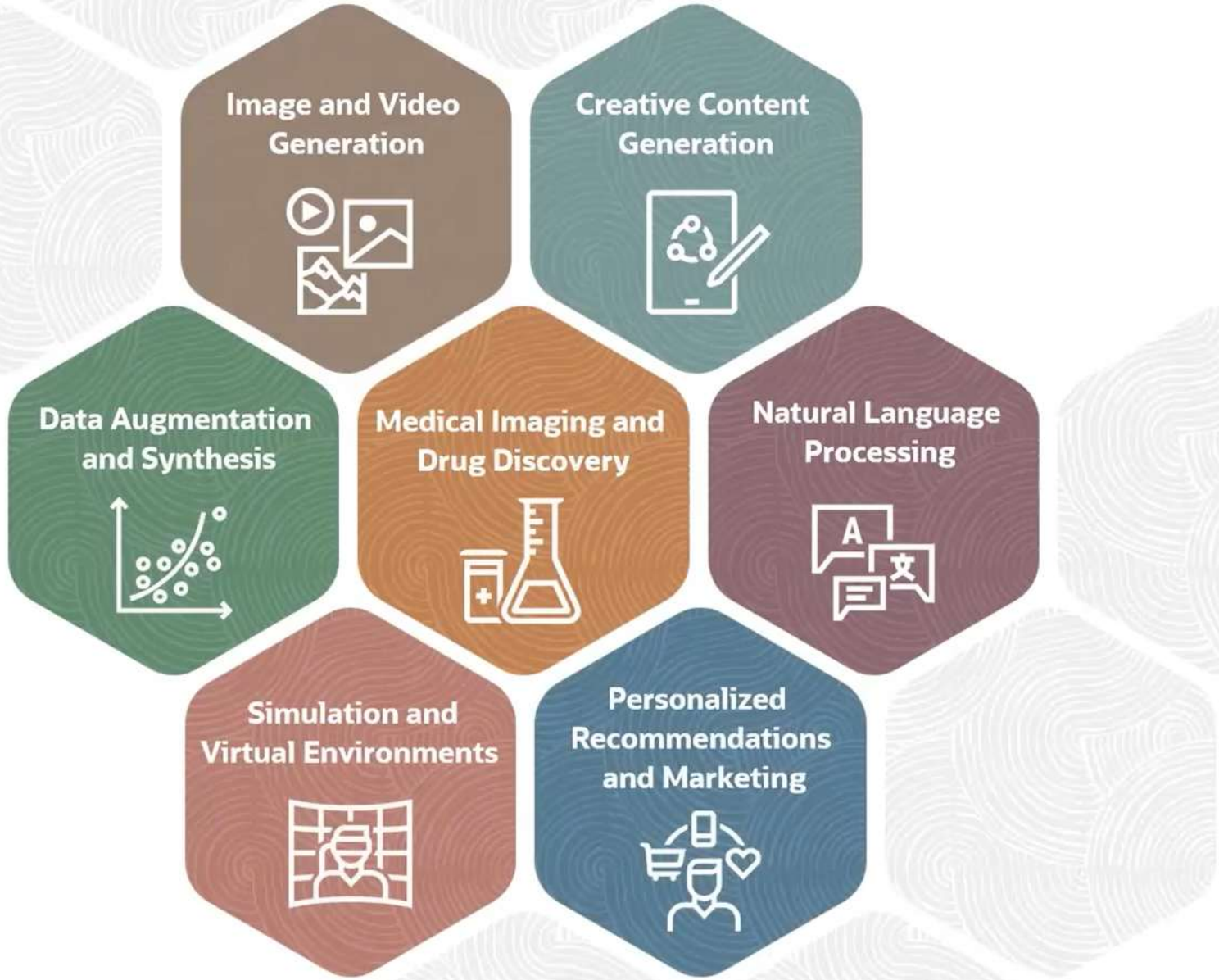
Text-Based

- Models can generate text, code, dialogue, articles
- Learns from large collections of text data to capture patterns, language structures, and semantic relationships

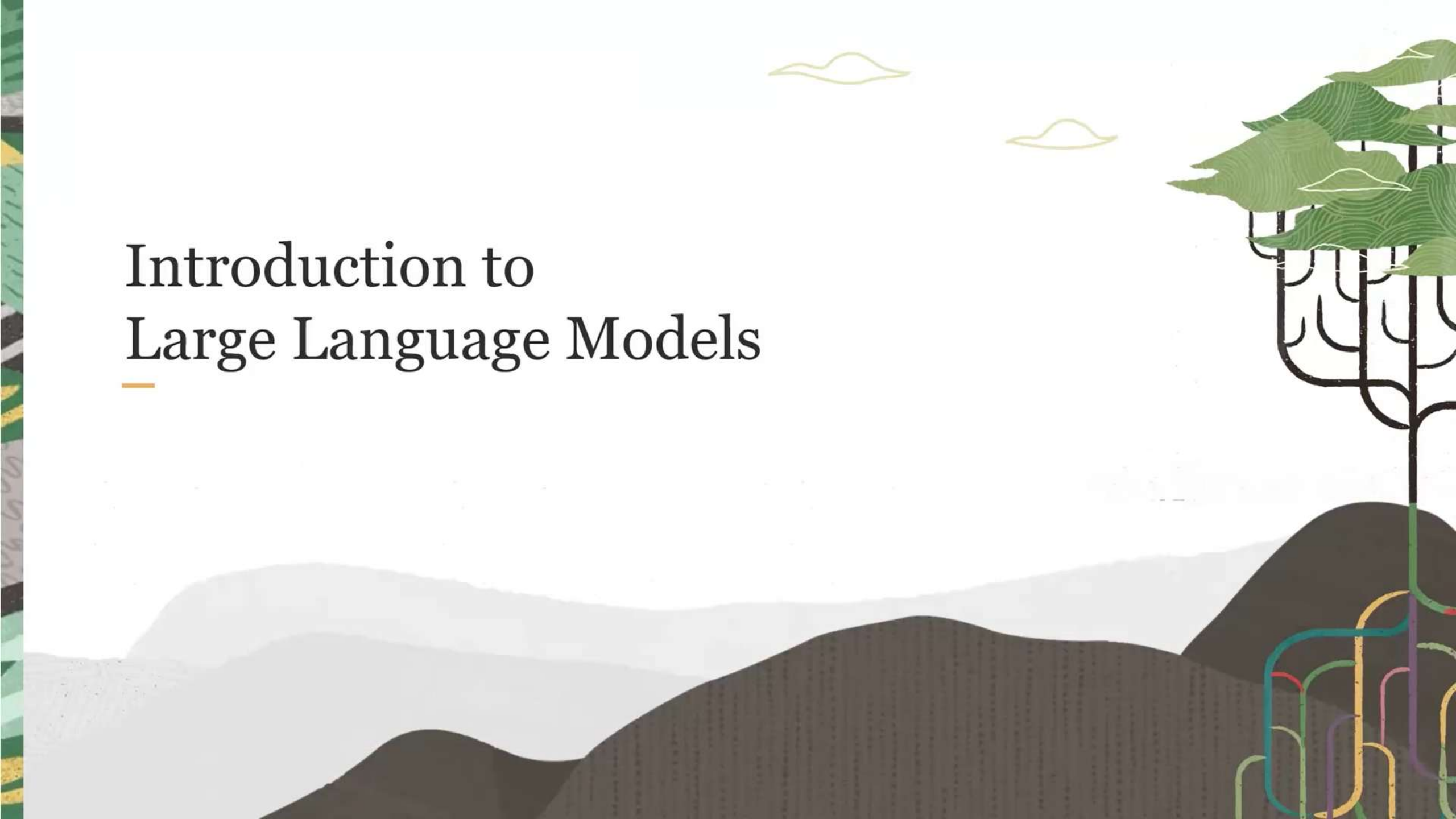
Multimodal

- Models capable of processing and generating multiple modalities.
- Modalities include text, images, audio, and video.
- Learns from multiple types of data simultaneously.

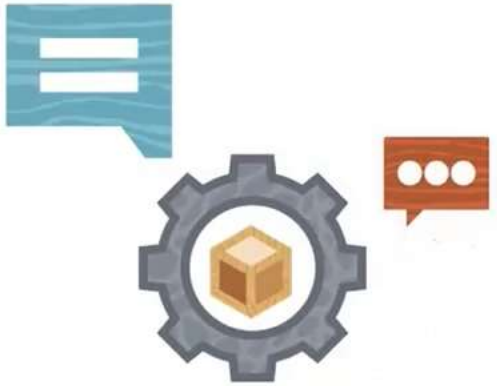
Generative AI Applications



Introduction to Large Language Models



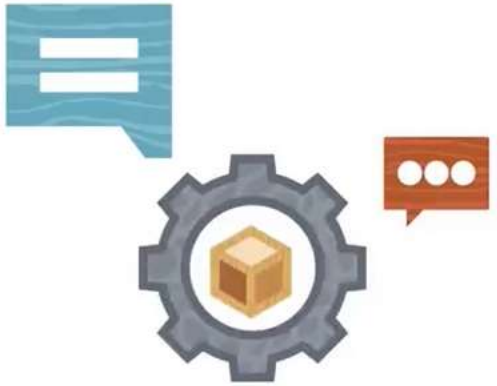
What is a Large Language Model?



A language model (LM) is **a probabilistic model of text**

What is a Large Language Model?

A language model (LM) is **a probabilistic model of text**



I wrote to the zoo to send me a pet. They sent me a _____

-- Opening of *Dear Zoo* by Rod Campbell

What is a Large Language Model?

A language model (LM) is **a probabilistic model of text**

I wrote to the zoo to send me a pet. They sent me a _____

Word	...	lion	elephant	dog	cat	panther	alligator	...
Probability		0.03	0.02	0.45	0.4	0.05	0.01	...

The LM gives a probability to every word in its vocabulary of appearing in the blank

What is a Large Language Model?

A language model (LM) is **a probabilistic model of text**

`I wrote to the zoo to send me a pet. They sent me a _____`

Word	...	lion	elephant	dog	cat	panther	alligator	...
Probability		0.03	0.02	0.45	0.4	0.05	0.01	

The LM gives a probability to every word in its vocabulary of appearing in the blank

"**Large**" in "large language model" (**L**LM) refers to # of parameters; no agreed-upon threshold

What is a Large Language Model?

Pick the highest probability word at each step

I wrote to the zoo to send me a pet. They sent me a _____

Word	...	lion	elephant	dog	cat	panther	alligator	...
Probability		0.03	0.02	0.45	0.4	0.05	0.01	

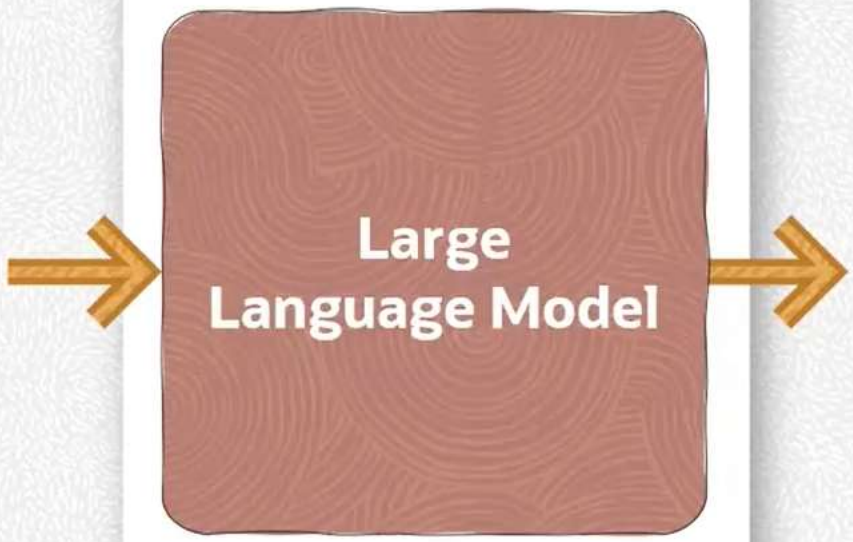
I wrote to the zoo to send me a pet. They sent me a **dog** _____

Word	...	EOS	elephant	dog	cat	panther	alligator	...
Probability		0.99	0.001	0.001	0.001	0.05	0.001	

Output: I wrote to the zoo to send me a pet. They sent me a dog.

Large Language Model Examples

1. *What is the capital of France?*
2. *Write an essay on French Revolution.*
3. *Translate “How are you” into French.*

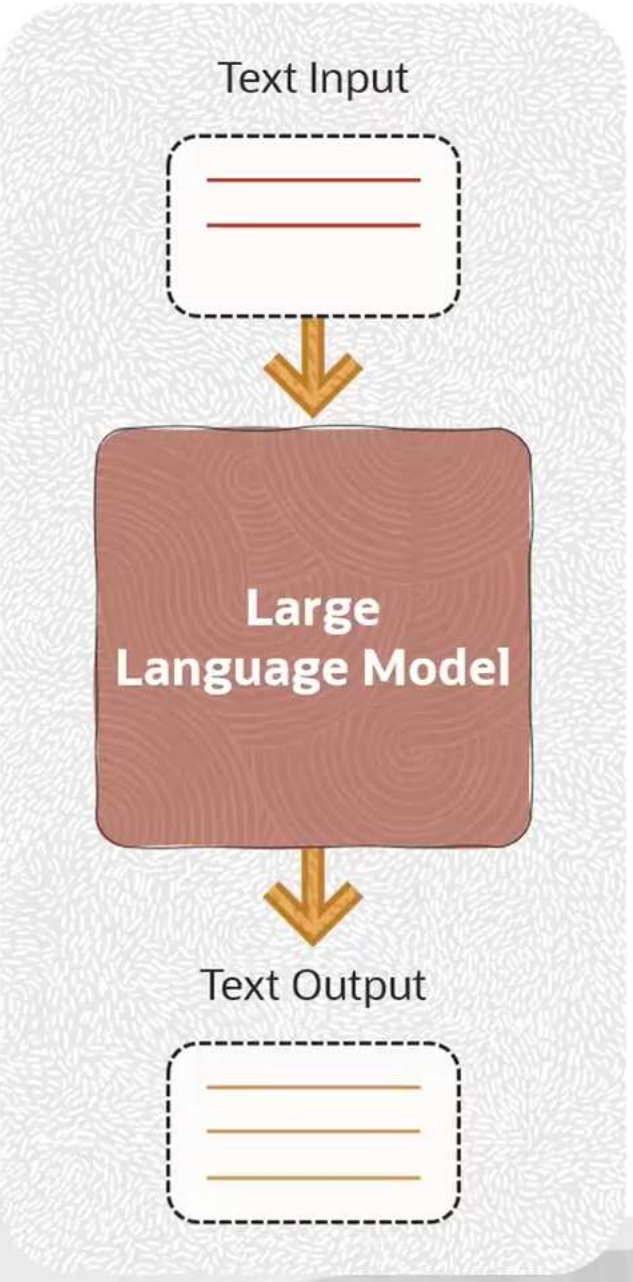


1. The capital of France is Paris.
2. **Title**: The French Revolution: A Turning Point in History

   Introduction: The French Revolution, which took place between 1789 and 1799, stands as one of the most influential and transformative events in human history...
3. "Comment allez-vous?"

Large Language Model Features

- Based on Deep Learning architecture (Transformer)
- Enhanced contextual understanding
- Natural language understanding and processing tasks
- Trained on vast language data to recognize patterns
- Scaled to hundreds of millions to billions of parameters



Model Size and Parameters

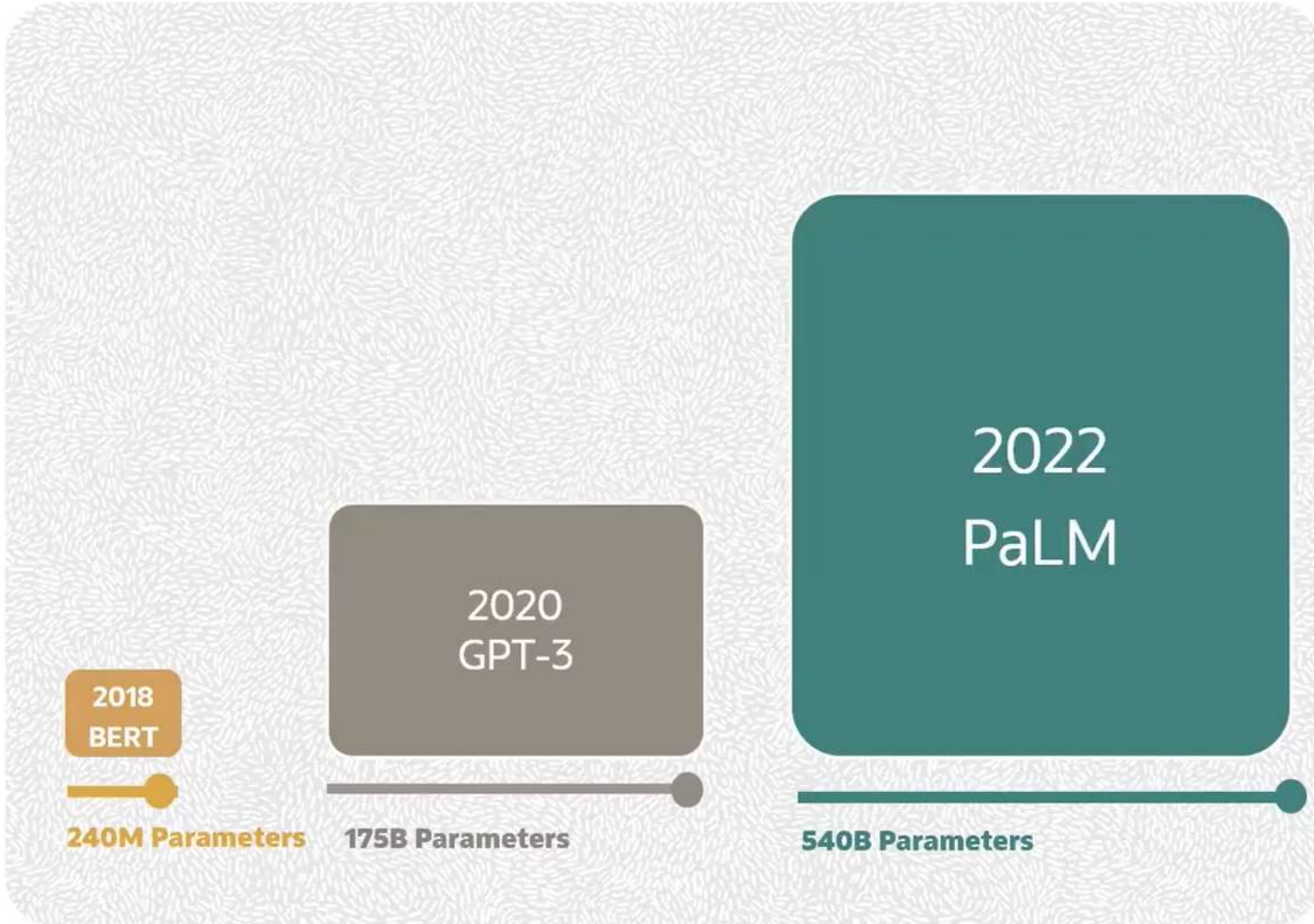
Parameters

Adjustable weights in the model's neural network.

A model with a larger number of parameters has more "capacity," (it can fit a more complex set of patterns in the data it was trained on).

Model Size

Memory required to store the model's parameters.



Transformers (Part 1)

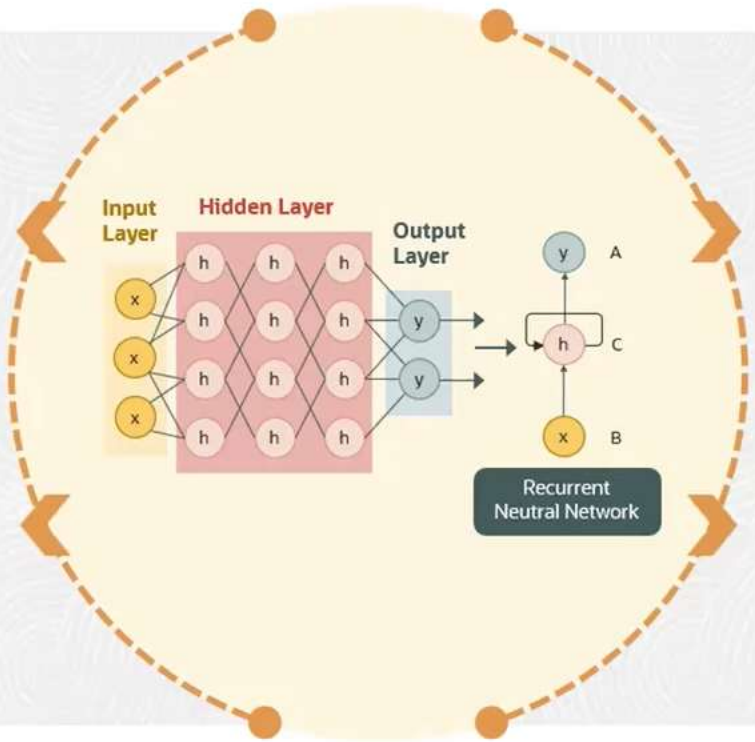
Understanding Language for Machines can be tricky

As a human, we understand easily that "it" refers to the frisbee.
But for a machine, this can be tricky.

Recurrent Neural Networks (RNN) – used for input data (sequence)

Handles Sequential Data

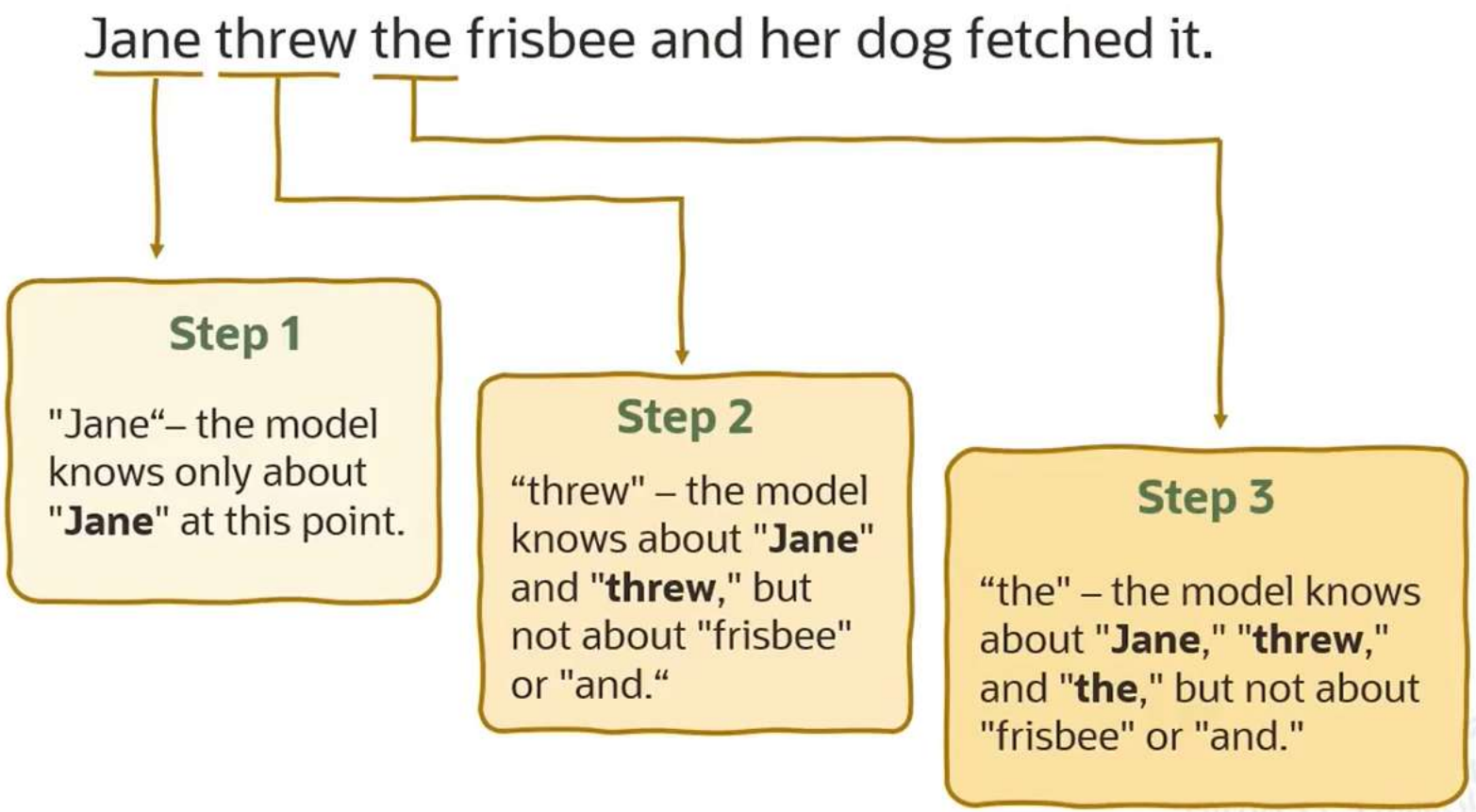
Maintains a hidden state or memory



Allows information to persist using a feedback loop

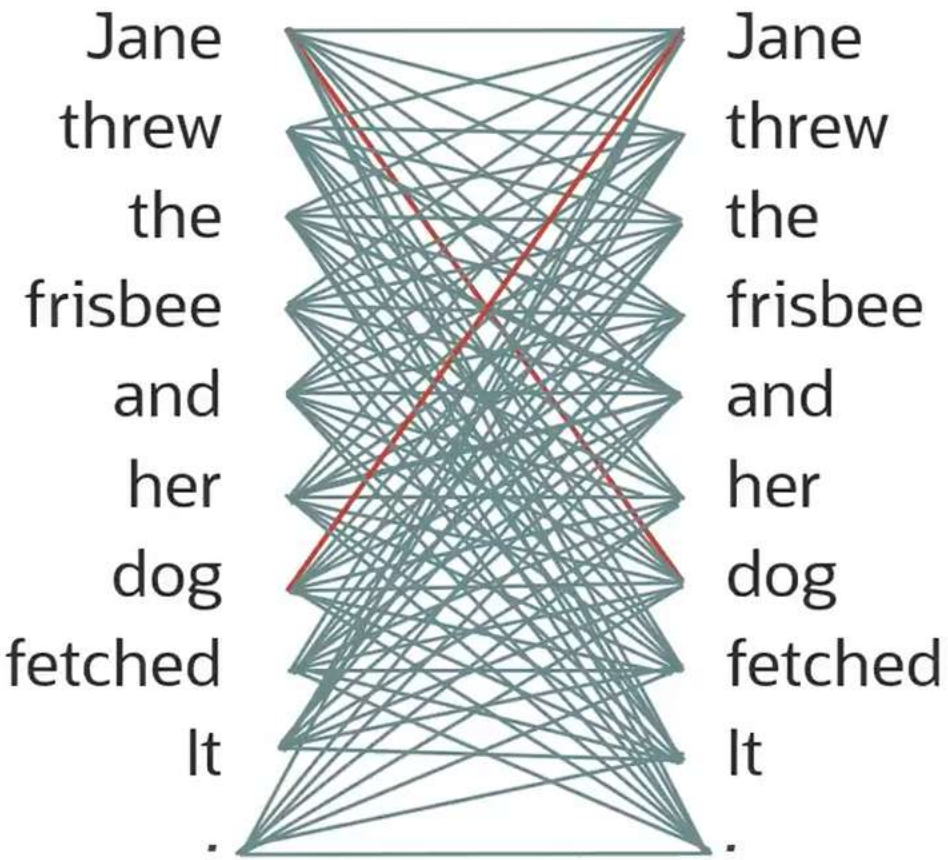
Captures dependencies

But RNNs struggle with Long-Range Dependencies



- RNNs process input data one element at a time, maintaining a hidden state.
- They can only capture dependencies between nearby words and struggle to connect words that are farther apart in the sentence.

Transformers understand relationships between all the words in a sentence



- Transformers can look at all the words in the sentence at the same time and understand how they relate to each other.
- Transformers can simultaneously understand the connections between "Jane" and "dog," even though they are far apart in the sentence.

Attention Mechanism: adds context to the Text

- Transformer doesn't just look at "it" but also "pays attention" to all the other words
- The model can figure out that "it" likely refers to the "frisbee".

Transformers

Type of Deep Learning

- "Attention is All You Need"
- Introduced by Vaswani in 2017, it differs significantly from RNN (and LSTM) models.

Self-Attention

Determining the relevance and importance of each word

Encoder-Decoder

Encoder processes input, whereas the decoder generates output.

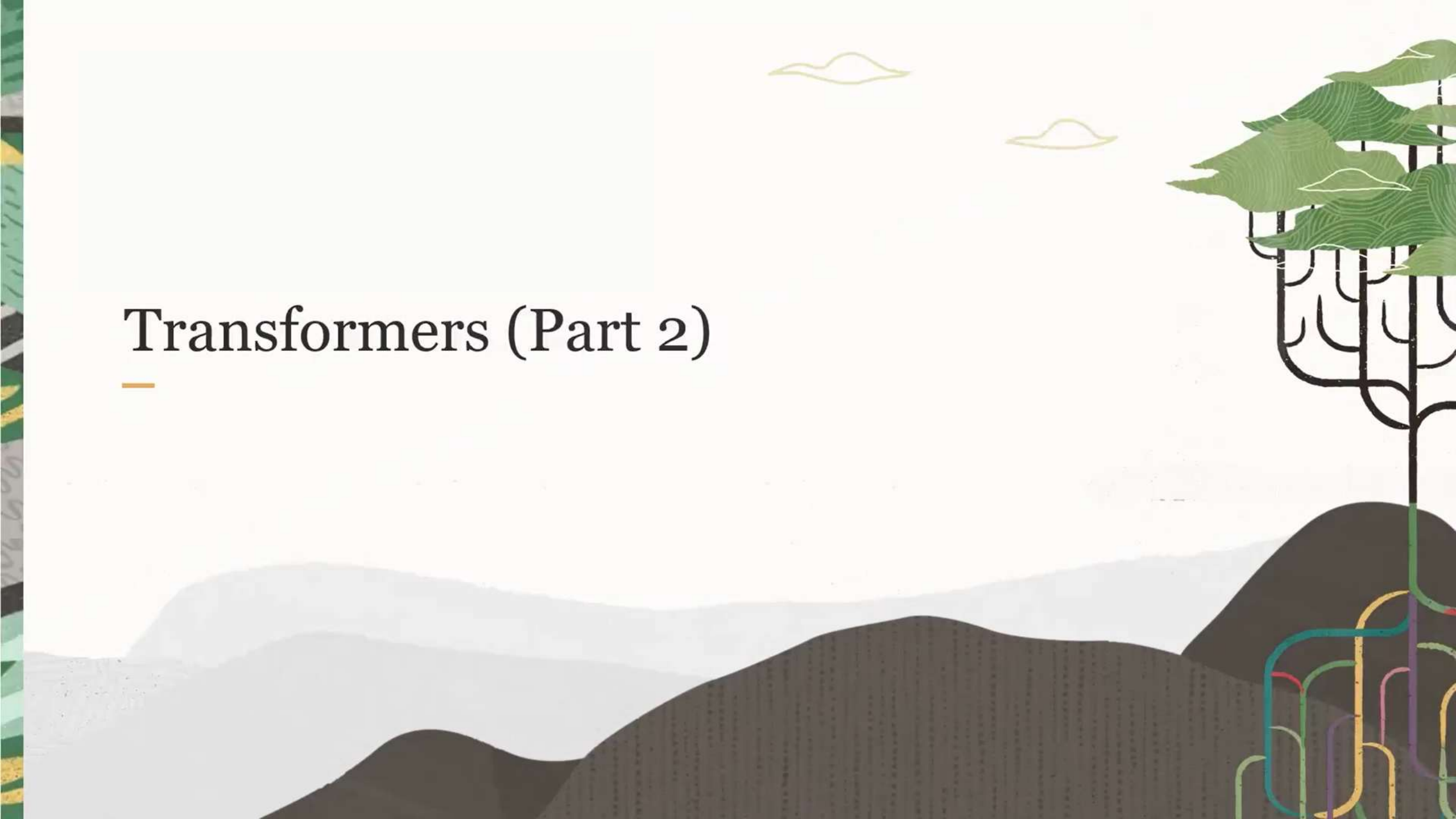
Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

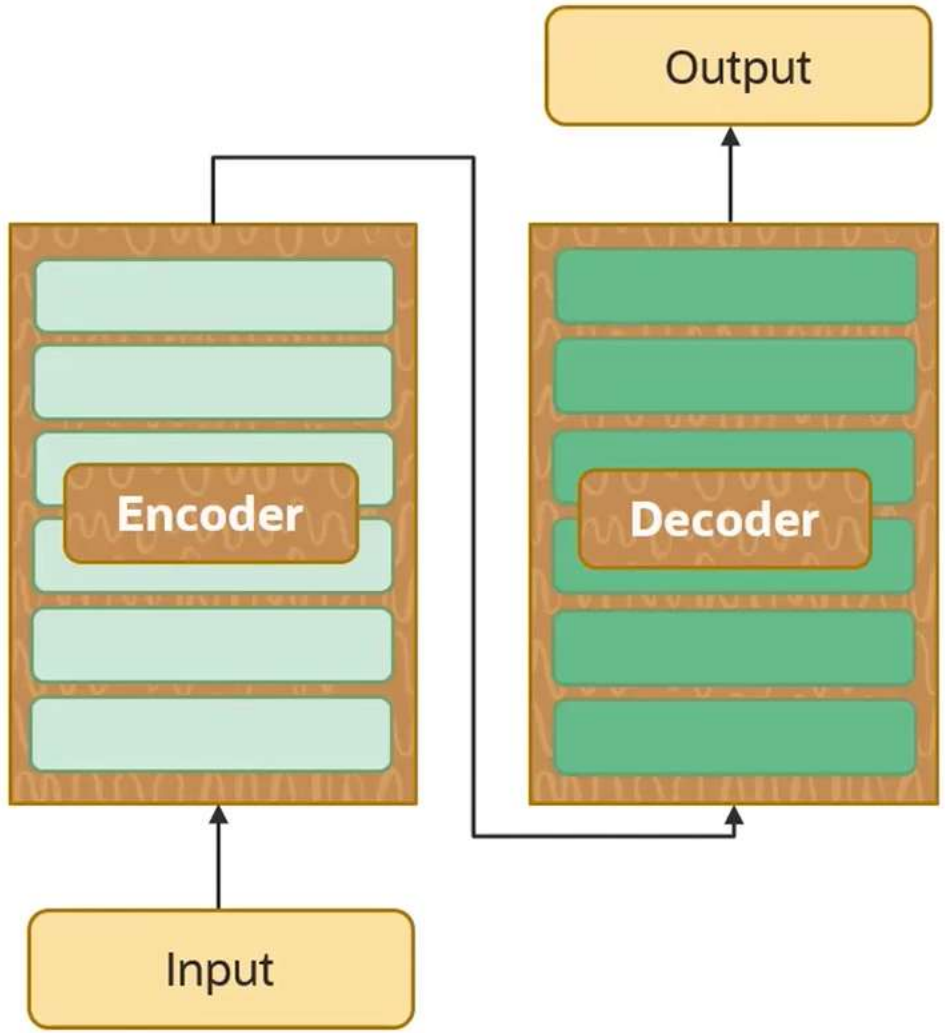
Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

Transformers (Part 2)



Encoder – Decoder



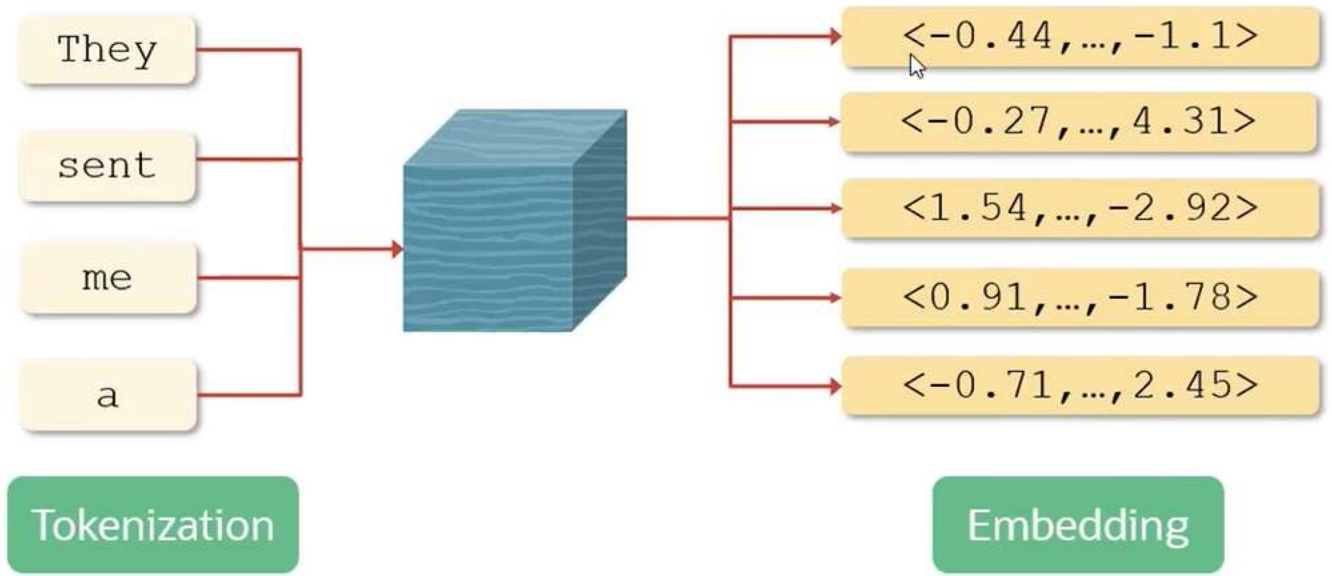
- Transformer Models have two main parts: Encoder and Decoder.
- Encoder reads the input text, encodes it into embeddings that capture meaning of the input text (using Attention mechanism).
- Then, the Decoder uses these embeddings to generate the output text.

Tokens

- Language models understand "tokens" rather than words.
- One token can be a part of a word, an entire word, or punctuation.
  - A common word such as “apple" is a token.
  - A word such as “friendship” is made up of two tokens – “friend” and “ship.”
- Number of Tokens/Word depend on the complexity of the text.
  - Simple text: 1 token/word (Avg.)
  - Complex text (fewer common words): 2-3 tokens/word (Avg.)

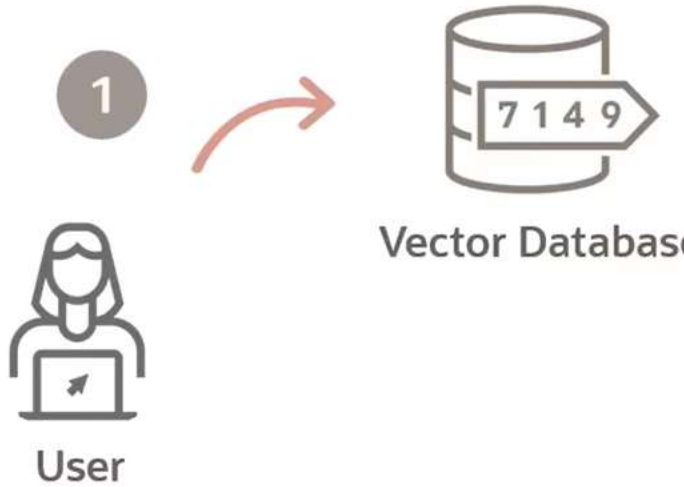
Embeddings

- Embeddings are numerical representations of a piece of text converted to number sequences.
- A piece of text could be a word, phrase, sentence, paragraph or one or more paragraphs.
- Embeddings make it easy for computers to understand the relationships between pieces of text.



Embeddings use case

1 The user's question is encoded as a vector and sent to a Vector database



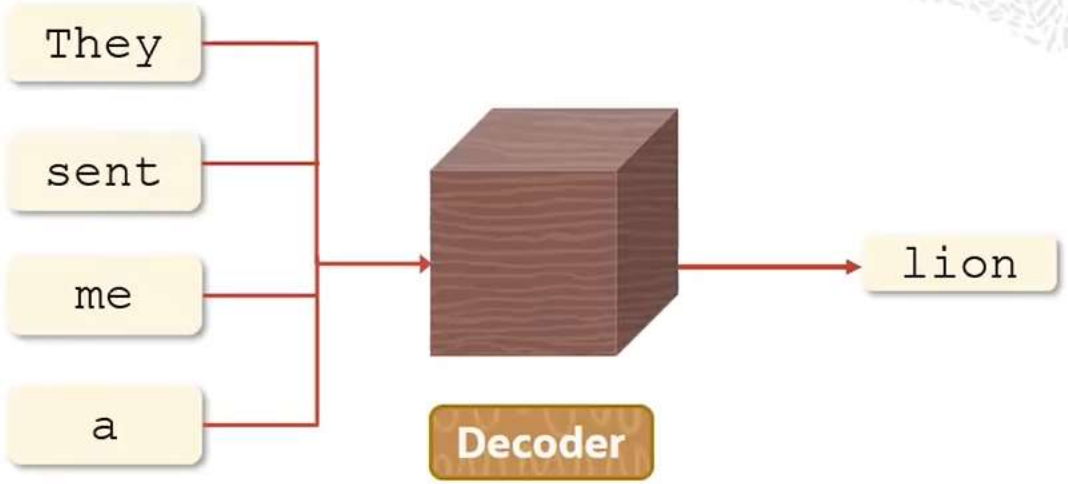
2 Vector DB finds private content (e.g. documents) that closely match the user's question

3 The content is sent to the LLM to help answer the user's question

4 LLM uses the content plus general knowledge to provide an informed answer

Decoders

Decoder – models take a sequence of words and output next word

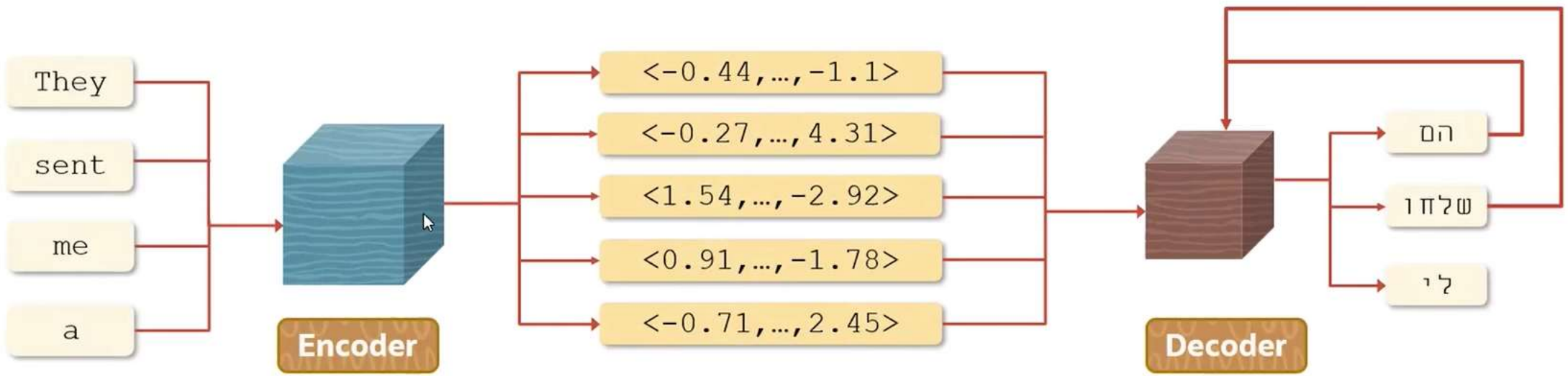


Primary uses: text generation, chat-style models, (including QA, etc…)

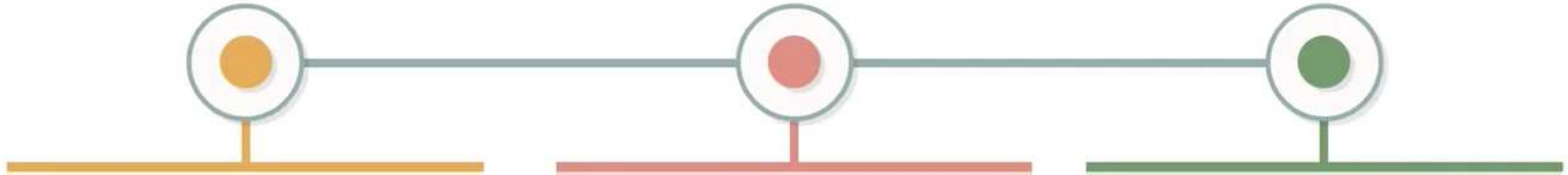
Encoder – Decoder

Encoder-decoder – encoder encodes a sequence of words to a set of vectors and the decoder generates the output sequence from the set of vectors.

Ideal for sequence-to-sequence tasks like machine translation.



Transformer Model Types



Encoder only

Transforms input data into a sequence of vectors

Useful for understanding input data without generating a new sequence

(Semantic Search)

Decoder only

Generate sequences, such as text, based on a given input

Useful for tasks that involve generating data

(Text Generation)

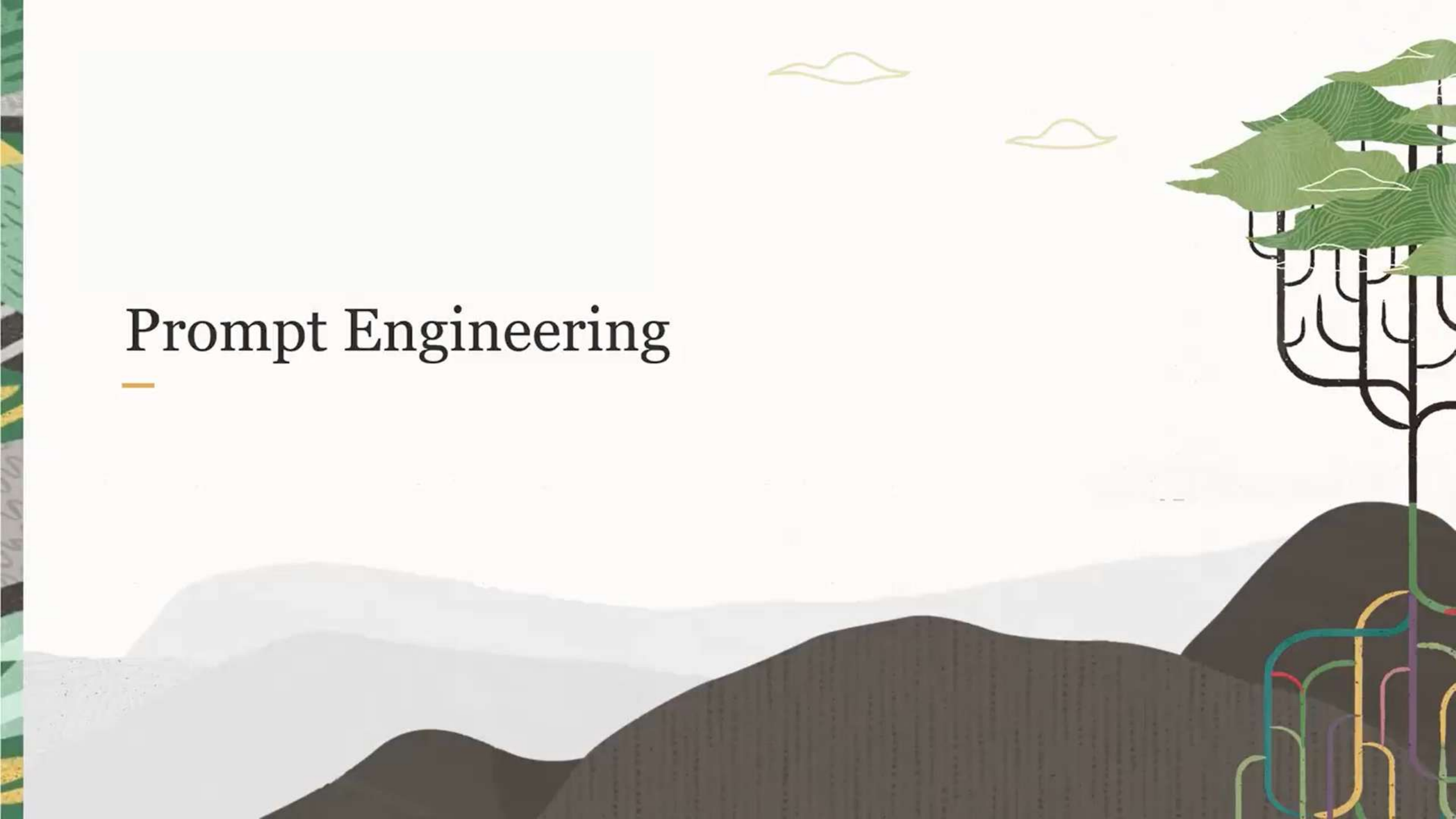
Encoder-Decoder

Combines an encoder for input processing and a decoder for sequence generation

Useful for sequence-to-sequence tasks

(Machine Translation)

Prompt Engineering



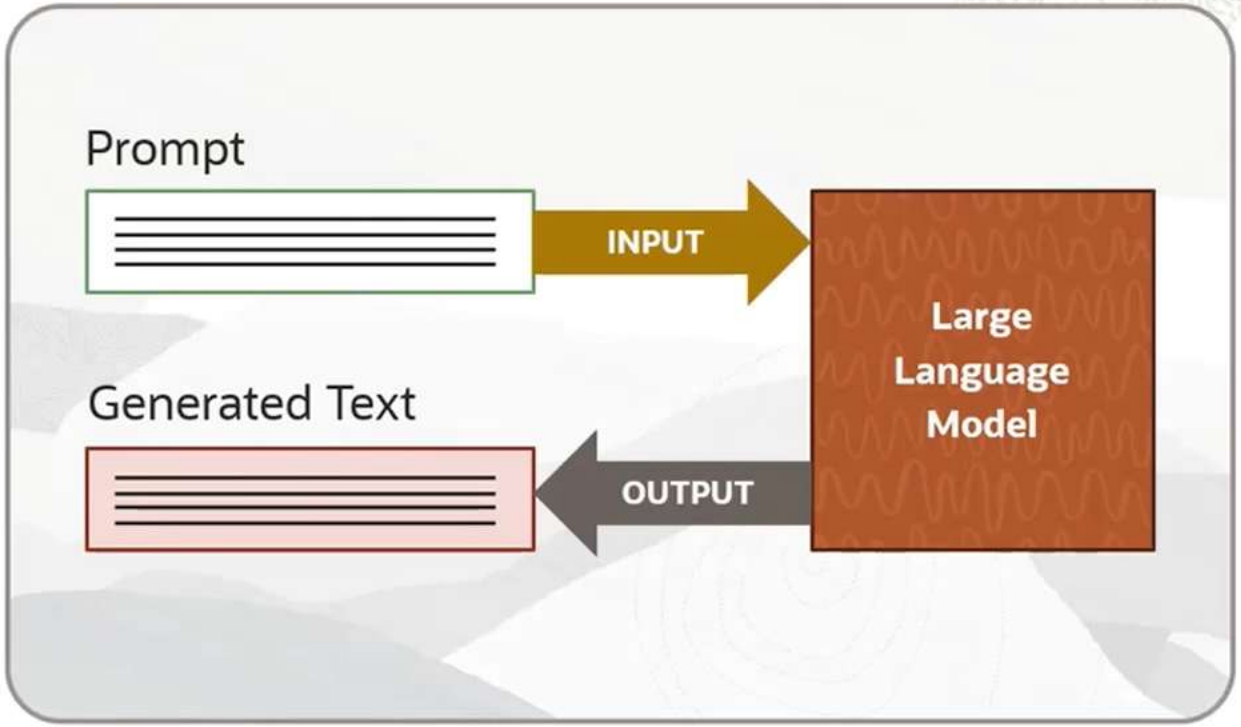
Prompt & Prompt Engineering

Prompt

- The input or initial text provided to the model

Prompt Engineering

- The process of iteratively refining a prompt for the purpose of eliciting a particular style of response



LLMs as next word predictors

Text prompts are how users interact with Large Language Models.

LLM models attempt to produce the next series of words that are most likely to follow from the previous text.

Prompt	Completion
Four score and seven years ago our	Forefathers brought forth a new nation, conceived in Liberty, and dedicated to the proposition that all men are created equal…. that this nation, under God, shall have a new birth of freedom -- and that government of the people, by the people, for the people, shall not perish from the earth.

Aligning LLMs to follow instructions

- Completion LLMs are trained to predict the next word on a large dataset of Internet text, rather than to safely perform the language task that the user wants.
- Cannot give instructions or ask questions to a completion LLM. Instead, need to formulate your input as a prompt whose natural continuation is your desired output.
- Instruction tuning is a critical step in LLM alignment. It involves fine-tuning a pre-trained LLM on a varied set of instructions, each paired with a desired output.

LLAMA 2: Open Foundation and Fine-Tuned Chat Models

Hugo Touvron* Louis Martin† Kevin Stone†
Peter Albert Amjad Almahairi Yasmine Babaei Nikolay Bashlykov Soumya Batra
Prajjwal Bhargava Shruti Bhosale Dan Bikel Lukas Blecher Cristian Canton Ferrer Moya Chen
Guillem Cucurull David Esiobu Jude Fernandes Jeremy Fu Wenyin Fu Brian Fuller
Cynthia Gao Vedanuj Goswami Naman Goyal Anthony Hartshorn Saghar Hosseini Rui Hou
Hakan Inan Marcin Kardas Viktor Kerkez Madian Khabsa Isabel Kloumann Artem Korenev
Punit Singh Koura Marie-Anne Lachaux Thibaut Lavril Jenya Lee Diana Liskovich
Yinghai Lu Yuning Mao Xavier Martinet Todor Mihaylov Pushkar Mishra
Igor Molybog Yixin Nie Andrew Poulton Jeremy Reizenstein Rashi Rungta Kalyan Saladi
Alan Schelten Ruan Silva Eric Michael Smith Ranjan Subramanian Xiaoqing Ellen Tan Binh Tang
Ross Taylor Adina Williams Jian Xiang Kuan Puxin Xu Zheng Yan Iliyan Zarov Yuchen Zhang
Angela Fan Melanie Kambadur Sharan Narang Aurelien Rodriguez Robert Stojnic
Sergey Edunov Thomas Scialom*

GenAI, Meta

Abstract

In this work, we develop and release Llama 2, a collection of pretrained and fine-tuned large language models (LLMs) ranging in scale from 7 billion to 70 billion parameters. Our fine-tuned LLMs, called LLAMA 2-CHAT, are optimized for dialogue use cases. Our models outperform open-source chat models on most benchmarks we tested, and based on our human evaluations for helpfulness and safety, may be a suitable substitute for closed-source models. We provide a detailed description of our approach to fine-tuning and safety improvements of LLAMA 2-CHAT in order to enable the community to build on our work and contribute to the responsible development of LLMs.

Reinforcement Learning from Human Feedback (RLHF) is used to fine-tune LLMs to follow a broad class of written instructions

Source: Touvron, Hugo; Martin, Louis; et al. (18 Jul 2023). "LLaMA-2: Open Foundation and Fine-Tuned Chat Models": 2307.09288:

In-context Learning and Few-shot Prompting

- **In-context learning –** conditioning (prompting) an LLM with instructions and or demonstrations of the task it is meant to complete.
- **k-shot prompting –** explicitly providing *k* examples of the intended task in the prompt.

```
1  Translate English to French:          ← task description
2  sea otter => loutre de mer            ← examples
3  peppermint => menthe poivrée          ←
4  plush girafe => girafe peluche        ←
5  cheese =>                             ← prompt
```

[Brown et al, 2020]

Few-shot prompting is widely believed to improve results over 0-shot prompting.

Source: Brown, Tom, et al. "Language models are few-shot learners." *Advances in neural information processing systems* 33 (2020): 1877-1901

Chain-of-Thought Prompting

- Provide examples in a prompt to show responses that include a reasoning step.
- The example answer contains a reasoning step, describing the calculation logic to get to the final answer, before giving the final answer.

```
Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can
has 3 tennis balls. How many tennis balls does he have now?
A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis
balls. 5 + 6 = 11. The answer is 11.

Q: The Oracle cafeteria in building 250 had 20 bananas. If they used 10 to
make pancakes and bought 8 more, how many bananas do they have?
A:"""
```

```
Model output: The cafeteria started with 20 bananas - 10 bananas = 10
bananas. They then bought 8 more bananas, which means they have 10 bananas
+ 8 bananas = 18 bananas. Therefore, the final answer is 18.
```

Source: Wei, Jason, et al. "Chain-of-thought prompting elicits reasoning in large language models." *Advances in Neural Information Processing Systems* 35 (2022): 24824-24837.

Hallucination

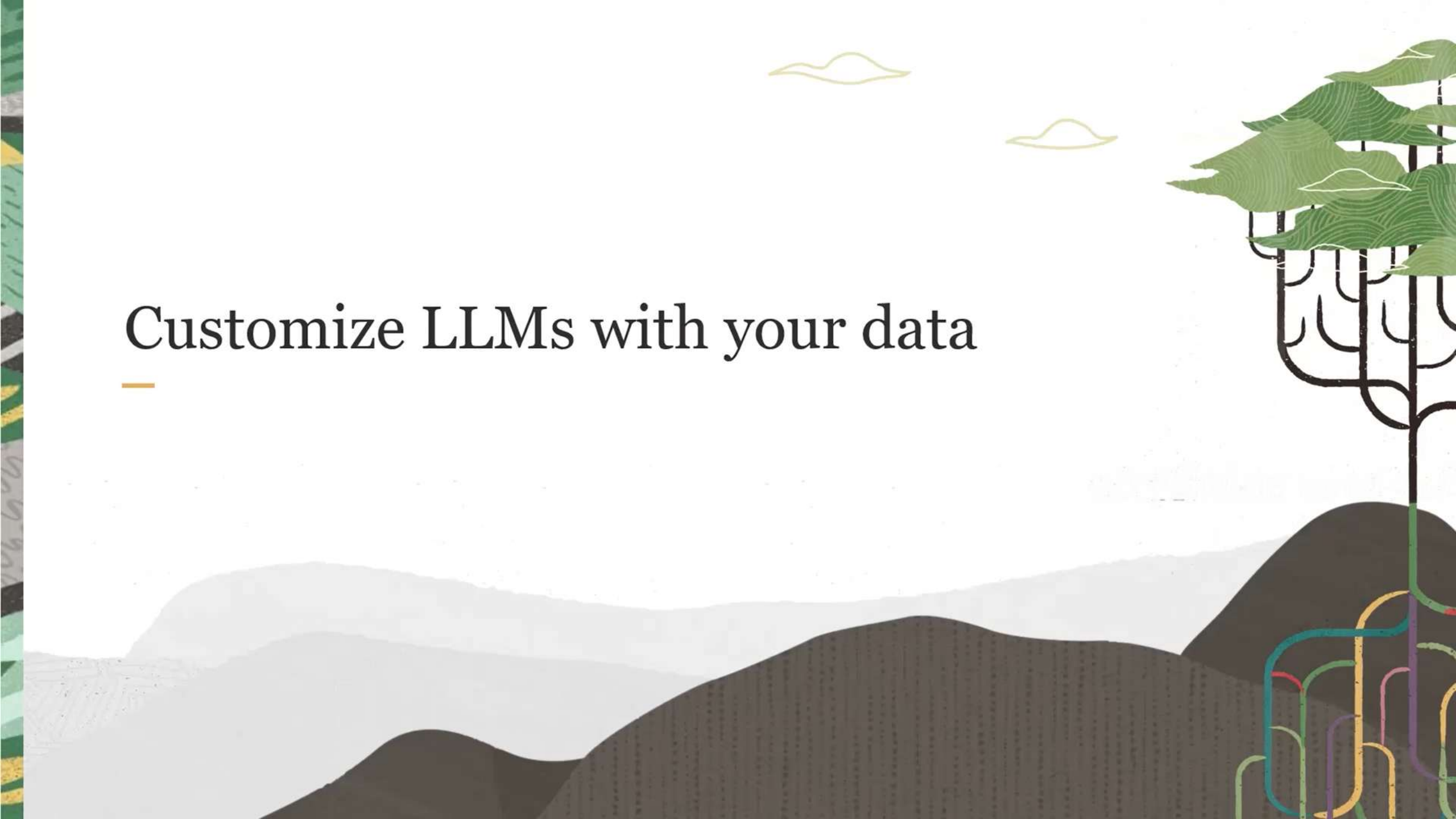
Hallucination – model generated text that is non-factual and/or ungrounded.

The current driving convention in the United States is to drive on the right side of the road, in the same direction as traffic flows on streets and highways. This is based on the system used in the United Kingdom and most of Europe, which has been in use since the 19th century. **During the first half of the 20th century, the United States gradually adopted the system of driving on the left side of the road.** In the 1950s, most states had converted to this convention.

FLAN-T5

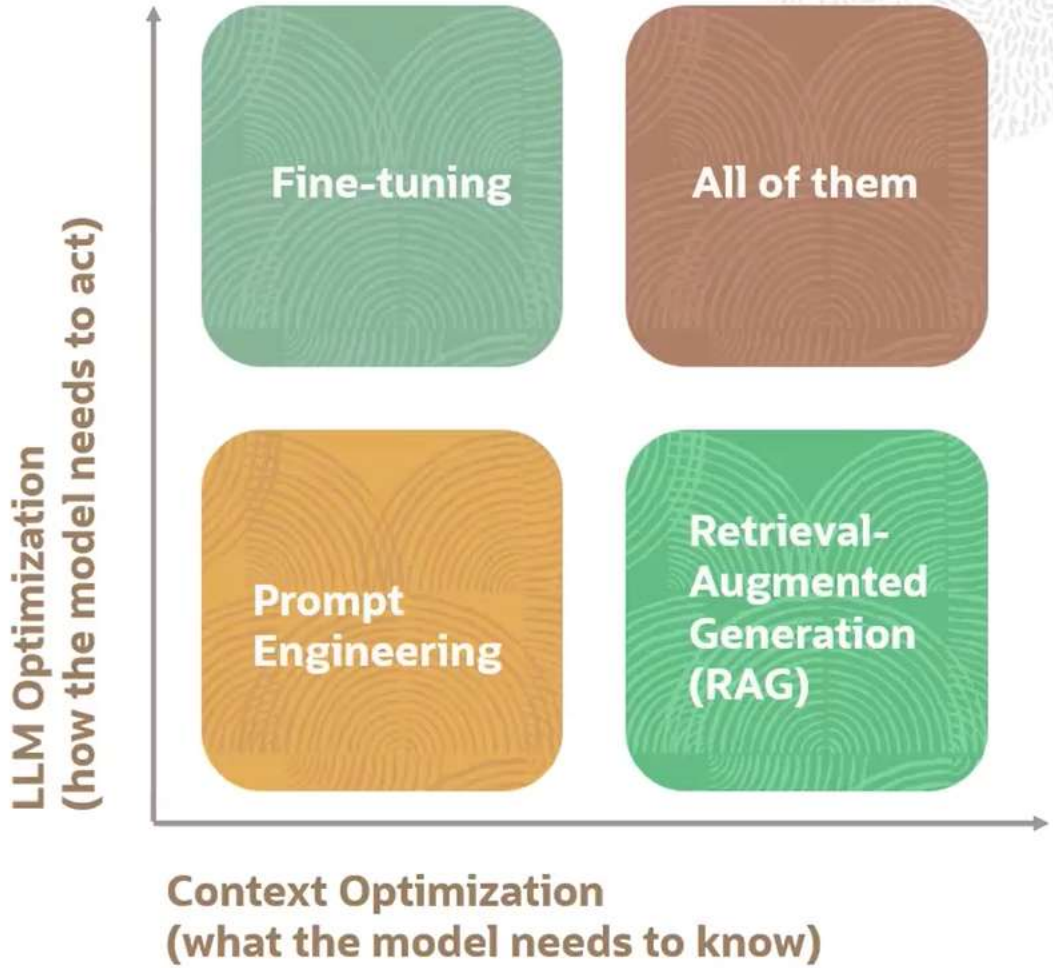
- There are some methods that are claimed to reduce hallucination (e.g., retrieval-augmentation).
- **There is no known methodology to reliably keep LLMs from hallucinating.**

Customize LLMs with your data

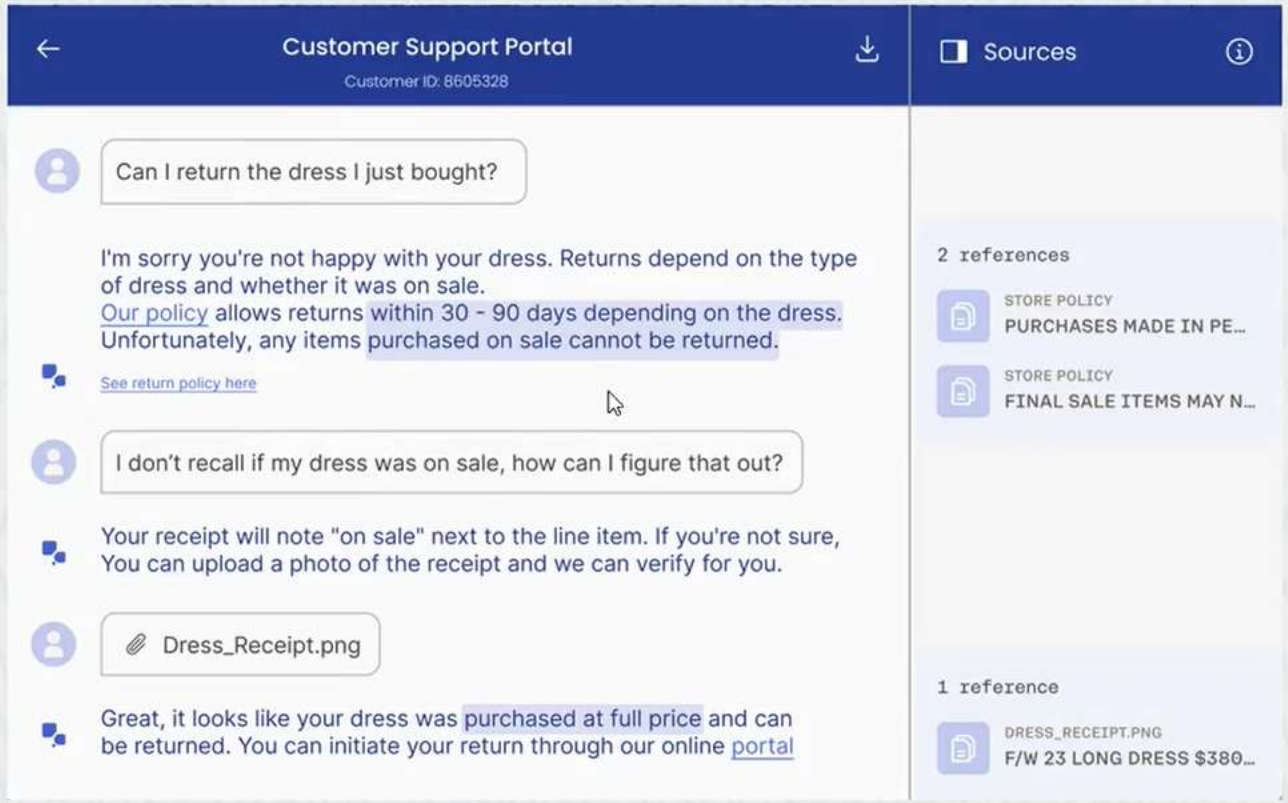


Customize LLMs with your data

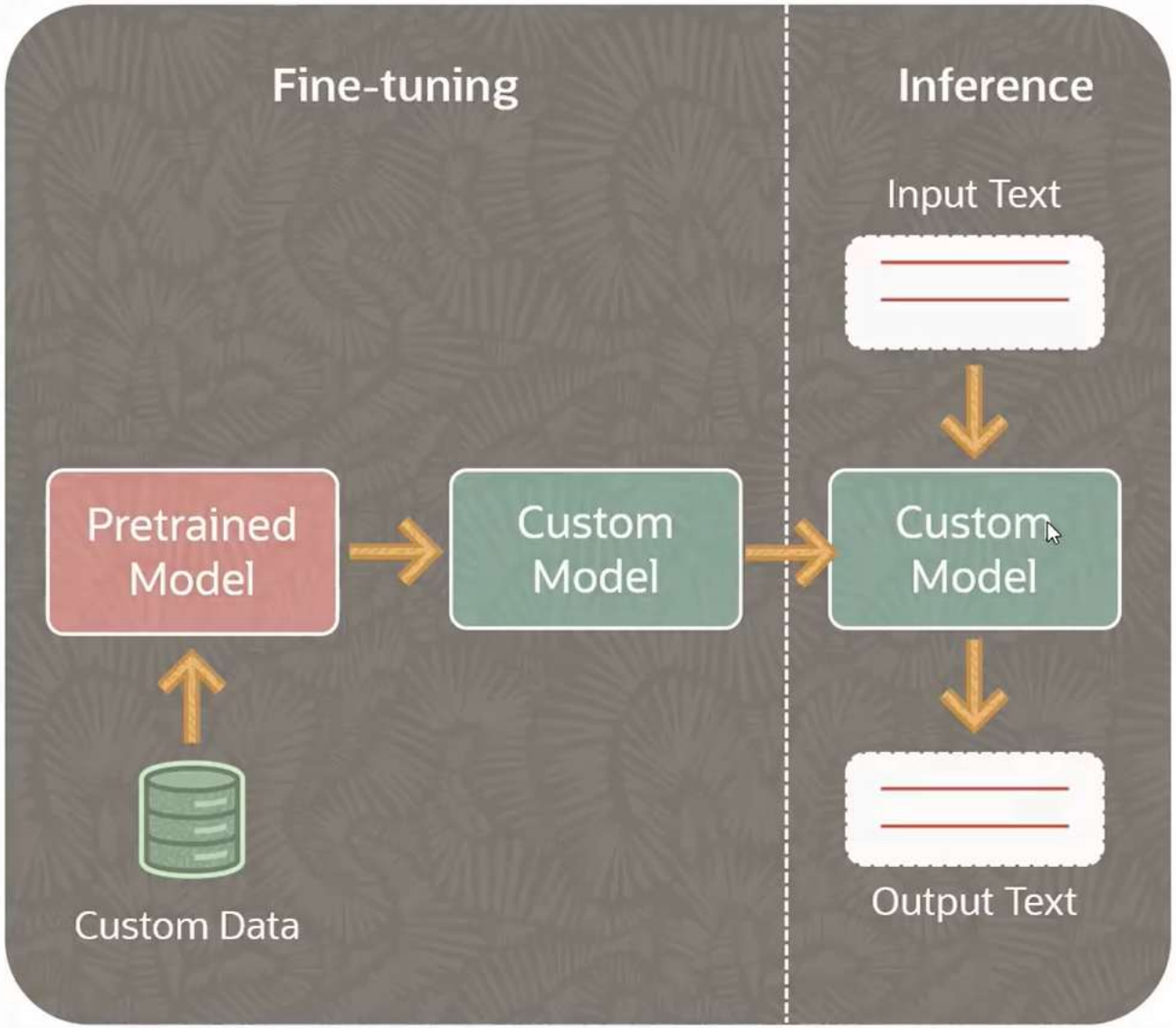
- Prompt Engineering is the easiest to start with; test and learn quickly.
- If you need more context, then use Retrieval-Augmented Generation (RAG).
- If you need more instruction following, then use Fine-tuning.



Retrieval-Augmented Generation (RAG)

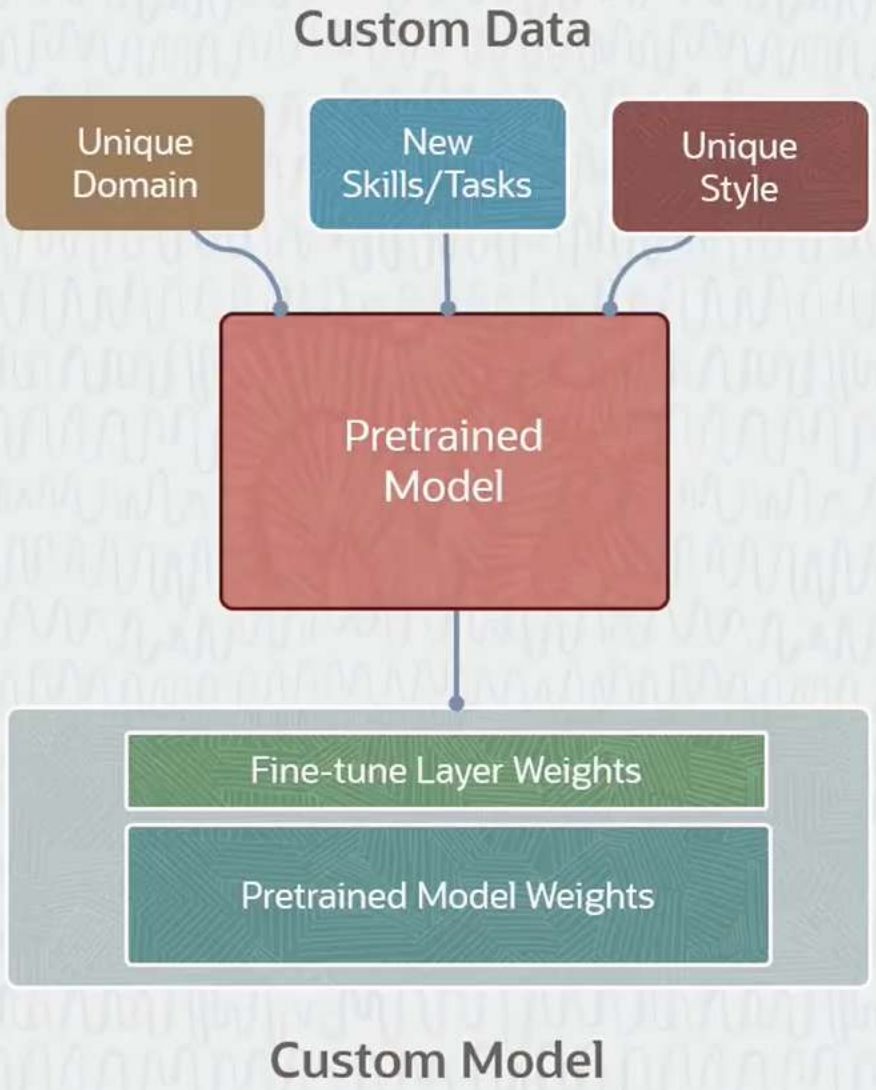


- Language model can query enterprise knowledge bases (databases, wikis, vector databases, etc.) to provide grounded responses.
- RAG does not require fine-tuning.



LLM Fine-tuning and Inference

- Fine-tuning – LLM is fine-tuned by taking a pretrained foundational model and providing additional training using custom data.
- Inference - model receives new text as input and generates output text based on what it has learned during pretraining and fine-tuning.



Fine-tuning a pretrained model

- Optimize a model on a smaller domain-specific dataset.
- Recommended when a pretrained model doesn't perform your task well or when you want to teach it something new.
- Adapt to specific style and tone and learn domain-specific words and phrases.

Fine-tuning Benefits

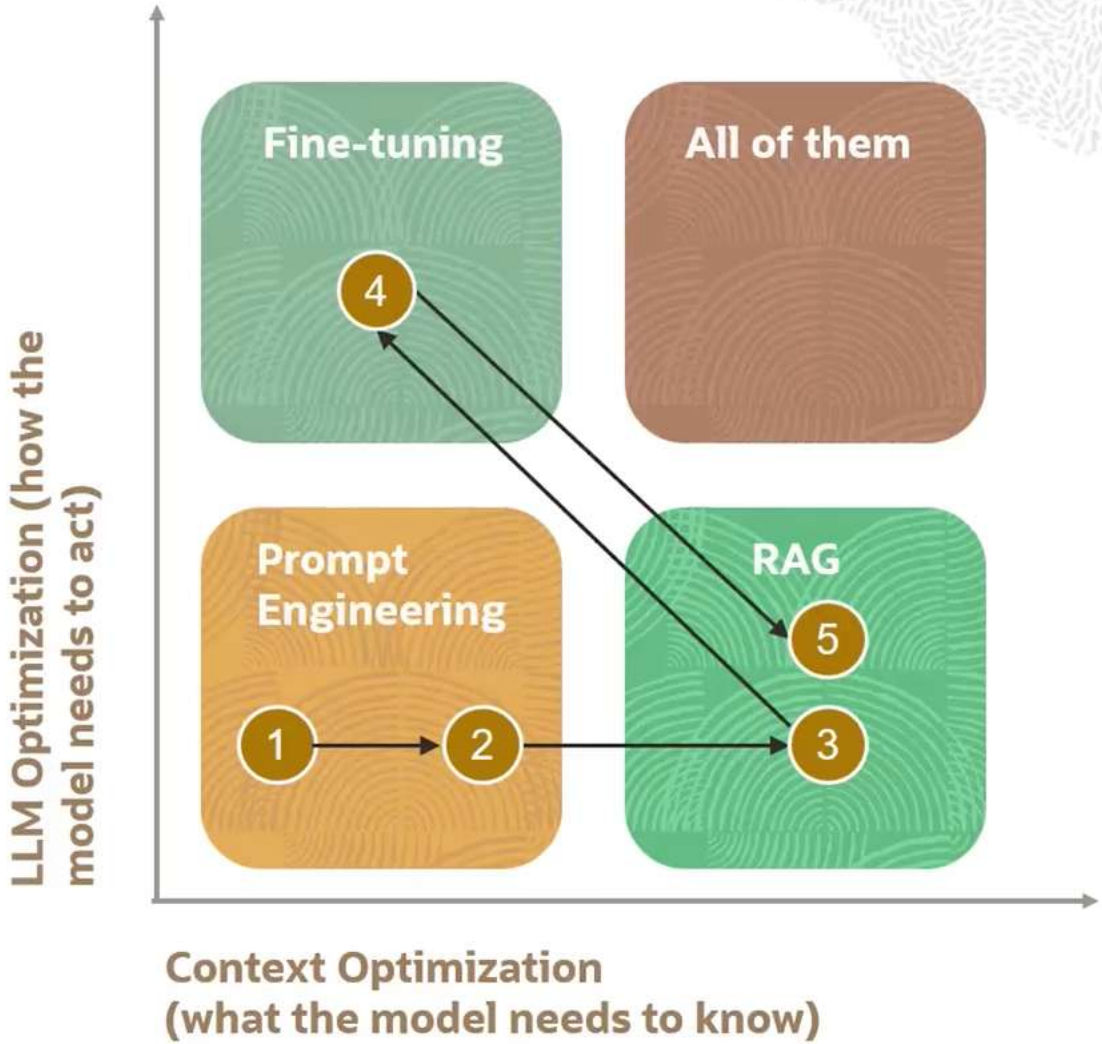
- **Improve Model Performance on specific tasks**
  - More effective mechanism of improving model performance than Prompt Engineering.
  - By customizing the model to domain-specific data, it can better understand and generate contextually relevant responses.
- **Improve Model Efficiency**
  - Reduce the number of tokens needed for your model to perform well on your tasks.
  - Condense the expertise of a large model into a smaller, more efficient model.

Customize LLMs with your data

Method	Description	When to use?	Pros	Cons
Prompt Engineering	Provide examples in the prompt to steer the model to better responses	• LLM already understands topics that are necessary for the text generation	• Very simple • No training cost	• Adds latency to each request
Retrieval-Augmented Generation (RAG)	Optimize the output of a LLM with targeted information without modifying the underlying model itself	• When the data changes rapidly • When you want to mitigate hallucinations by grounding answers (in enterprise data)	• Access the latest data • Grounds the result • Does not require fine-tuning	• More complex to setup • Requires a compatible data source
Fine-tuning	Adapt a pretrained LLM to perform a specific task on custom data	• LLM does not perform well on a particular task • Data required to adapt the LLM is too large for prompt engineering • Latency with the current LLM is too high	• Increase in model performance on a specific task • No impact on model latency	• Requires a labeled dataset which can be expensive and time-consuming to acquire

Customize LLMs with your data

1. Start with a simple Prompt.
2. Add Few shot Prompting.
3. Add simple retrieval using RAG.
4. Fine-tune the model.
5. Optimize the retrieval on fine-tuned model.



1. Welcome to AI Foundations
Welcome to AI Foundations

2. AI Foundations
Introduction to AI
AI - Tasks and Data
AI vs ML vs DL

3. Machine Learning Foundations
Introduction to Machine Learning
Supervised Learning - Regression
Supervised Learning - Classification
Unsupervised Learning
Reinforcement Learning

4. Deep Learning Foundations
Introduction to Deep Learning
Deep Learning Models-Sequence Models
Deep Learning Models - CNN

5. Generative AI and LLM Foundations
Introduction to Generative AI
Introduction to Large Language Models
Transformers Part-1
Transformers Part-2
Prompt Engineering
Customize LLMs with your data

6. OCI AI Portfolio
AI Services Overview
ML Services Overview
AI Infrastructure
GPUs and Superclusters in OCI
Responsible AI

7. OCI Generative AI Service
OCI Generative AI
Vector Search
Select AI

8. OCI AI Services
Language Intro
Speech Intro
Vision Intro
Document Understanding

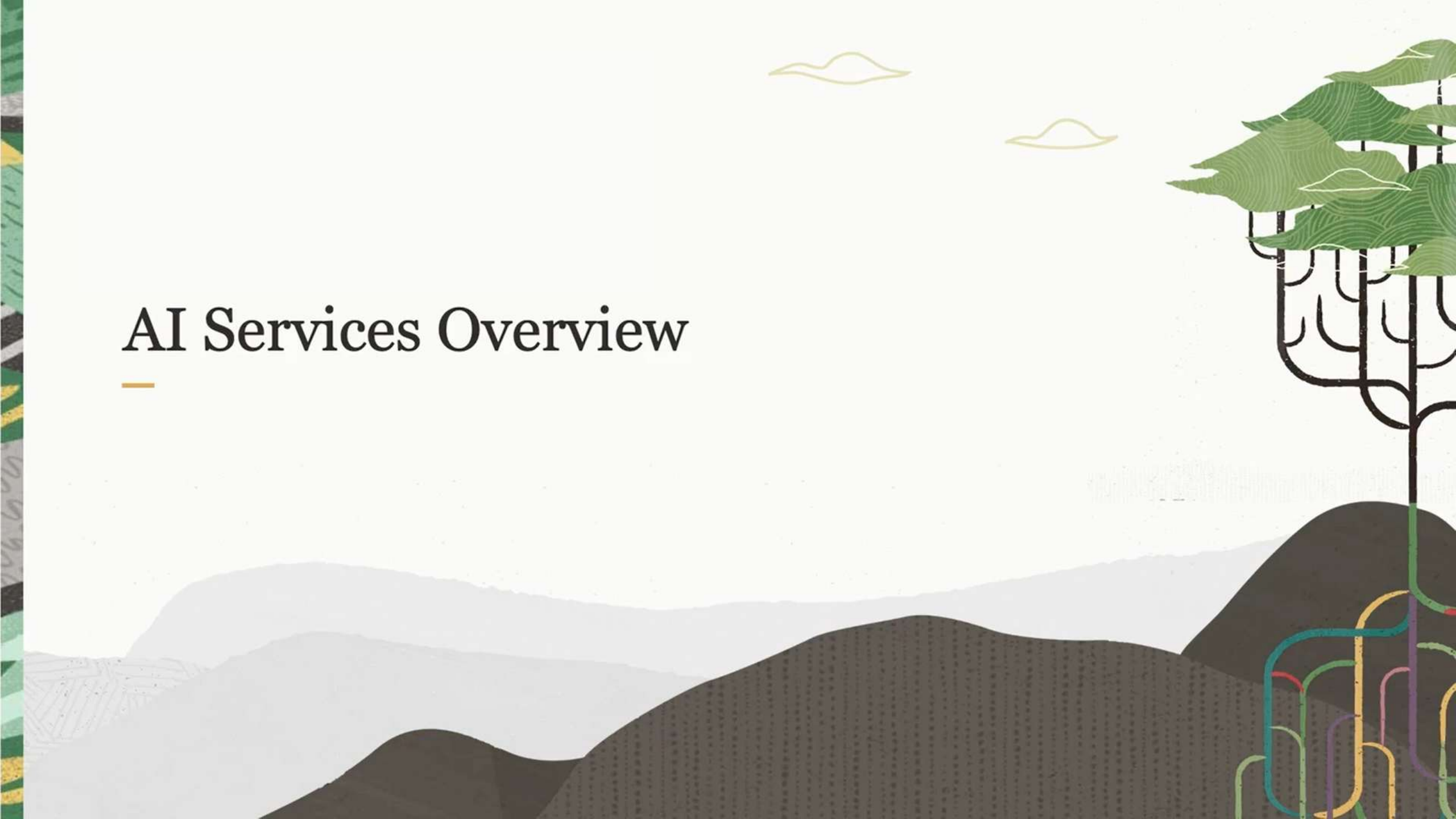
OCI AI Portfolio

Module 6

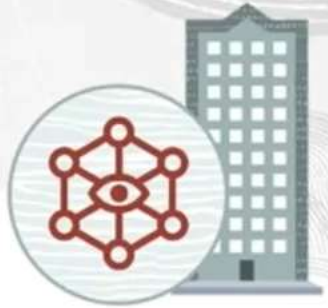
Objectives

- Provide an overview of OCI AI Services.
- Explain OCI Data science concepts.
- Explain OCI ML services, including Data Labeling and OML.
- Describe OCI AI Infrastructure components.
- Explain Oracle Cloud Responsible AI principles.

AI Services Overview



AI for the enterprise



SaaS Apps

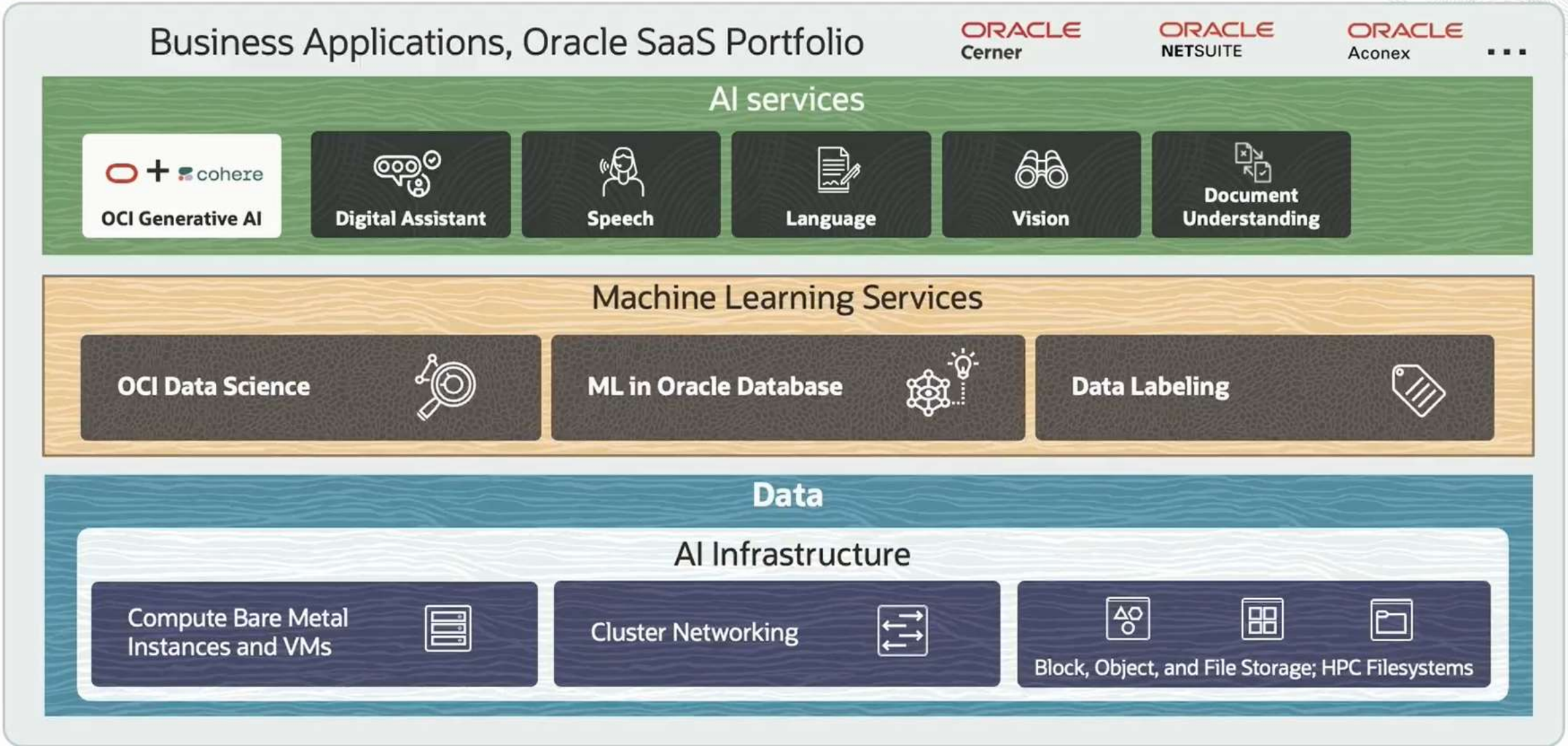
AI Services

Infrastructure

Data

- Oracle has been focusing on bringing AI to the enterprise at every layer of our stack.
- This approach involved extensive investment from infrastructure to SaaS apps.
- Generative AI and massive scale models are the more recent steps taken.

Oracle AI Stack



Ways to Access Oracle Cloud Infrastructure AI Services

OCI Console

Provides easy-to-use browser-based interface; enables access to notebook sessions and all service features

Rest API

Provides access to service functionality; requires programming expertise

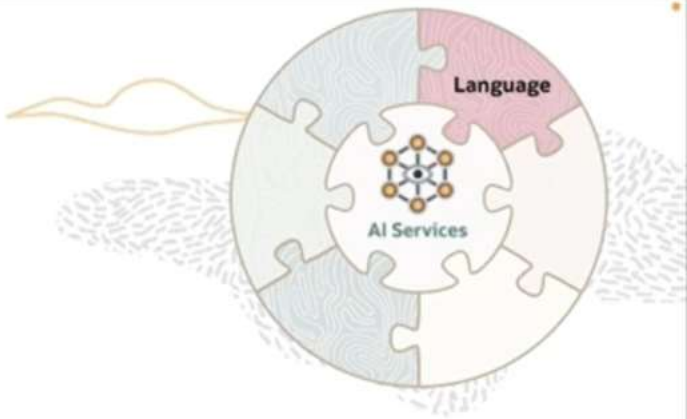
Language SDKs

Provides programming language SDKs for Java, Python, .Net, Go, Ruby, and TypeScript/JavaScript

CLI

Provides quick access and full functionality without the need for scripting

Language Overview



Pretrained Models

Language Detection, Sentiment Analysis, Key Phrase Extraction

Custom Models

Named entity recognition, Text Classification

Text Translation

Neural Machine translation to translate to various languages

Vision

Image Analysis
Pretrained Model

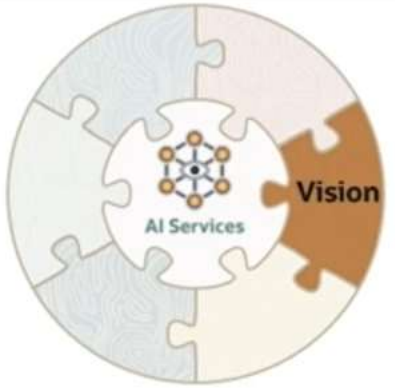
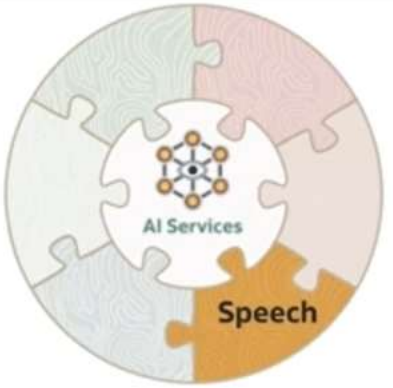
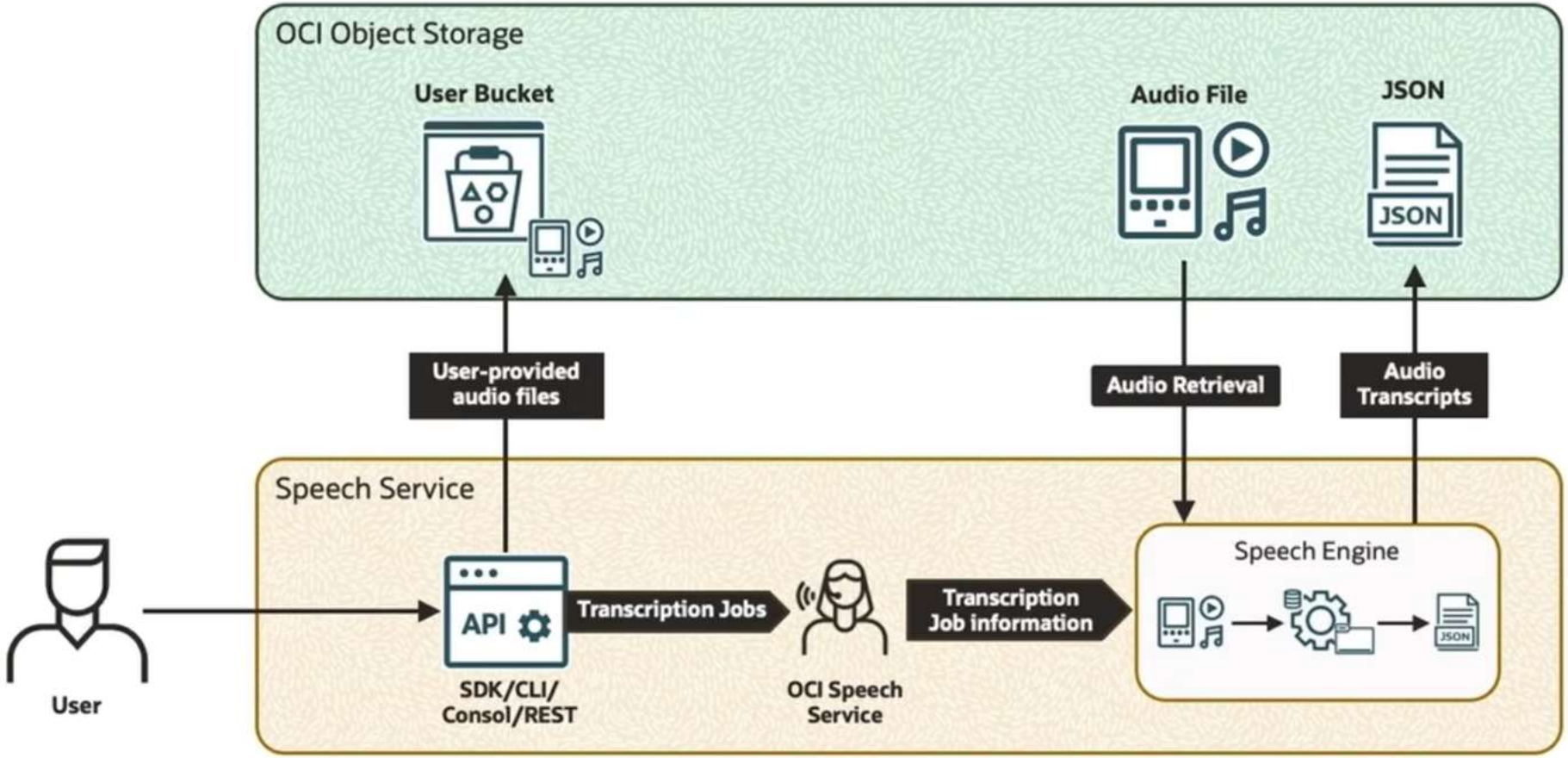


Image Analysis
Custom Model

Speech

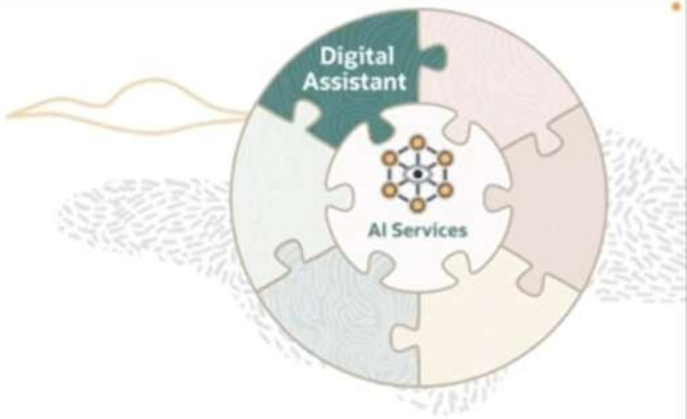

AI Services
Speech

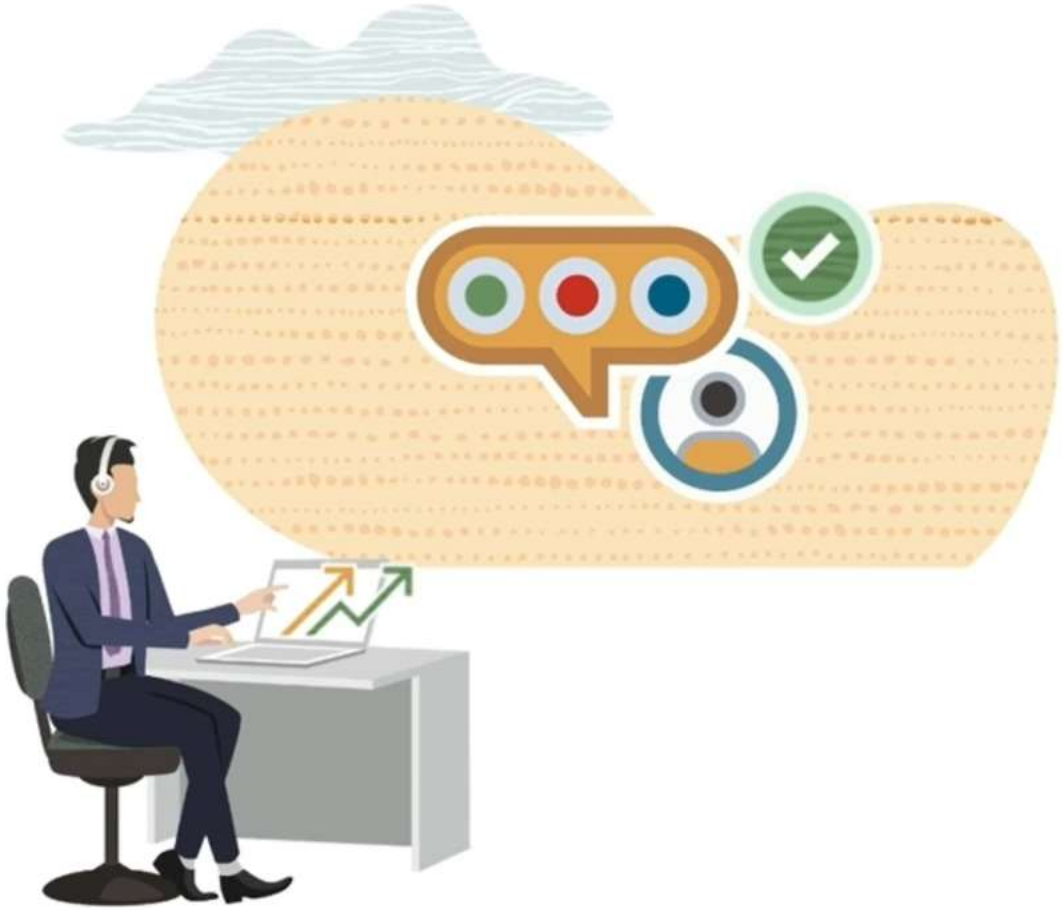

OCI Object Storage
User Bucket
Audio File
JSON
JSON
User-provided audio files
Audio Retrieval
Audio Transcripts
Speech Service
Speech Engine
API
Transcription Jobs
Transcription Job information
JSON
User
SDK/CLI/ Consol/REST
OCI Speech Service

Document Understanding

- Optical Character Recognition - detects texts
- Text Extraction – word-level and line-level text
- Key Value Extraction - extracts key–value pair
- Table Extraction - extracts content in tabular format
- Document classification – classifies based on features

Digital Assistant

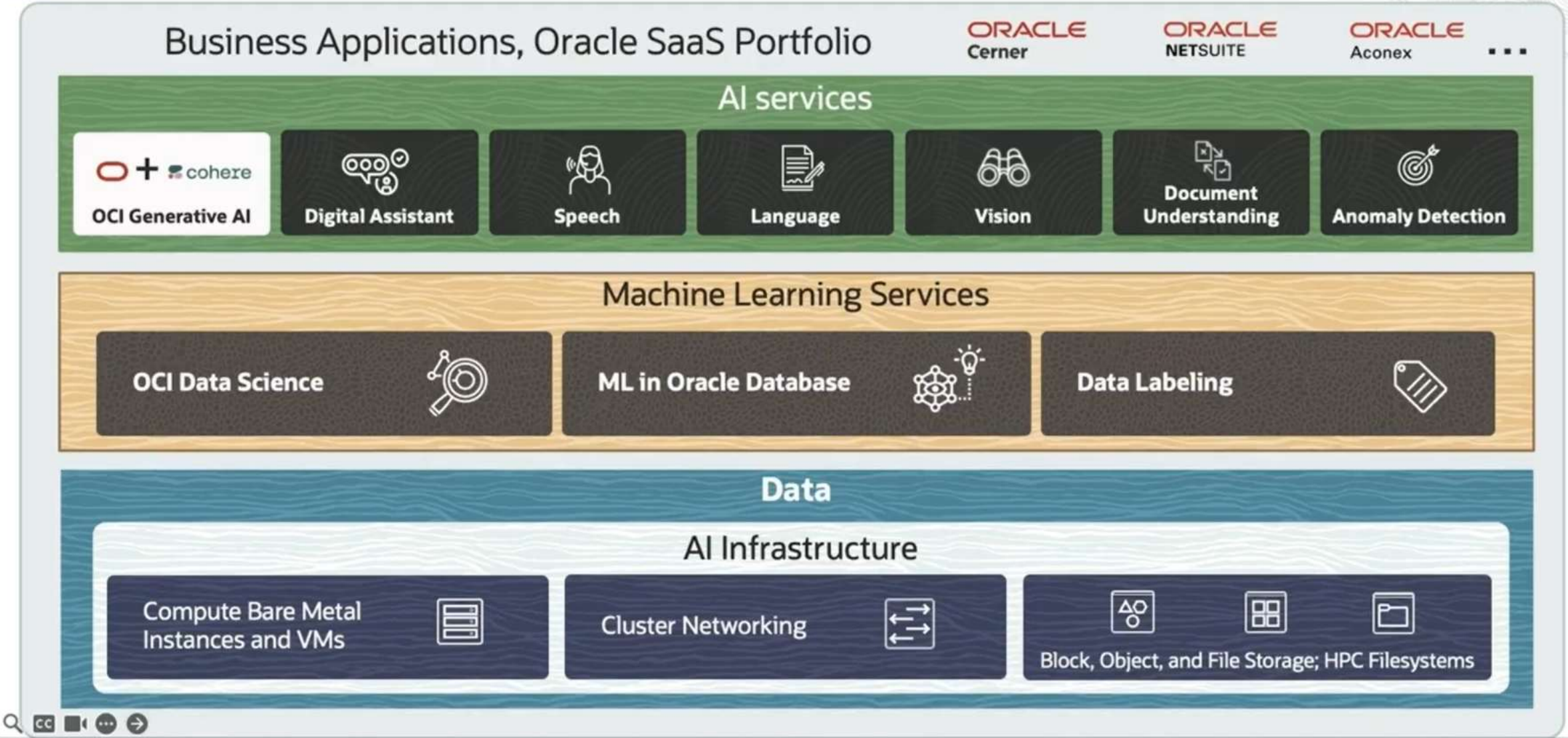




- Interacts with the user
- Lists what it can do
- Routes user's requests to skills
- Handles disambiguation

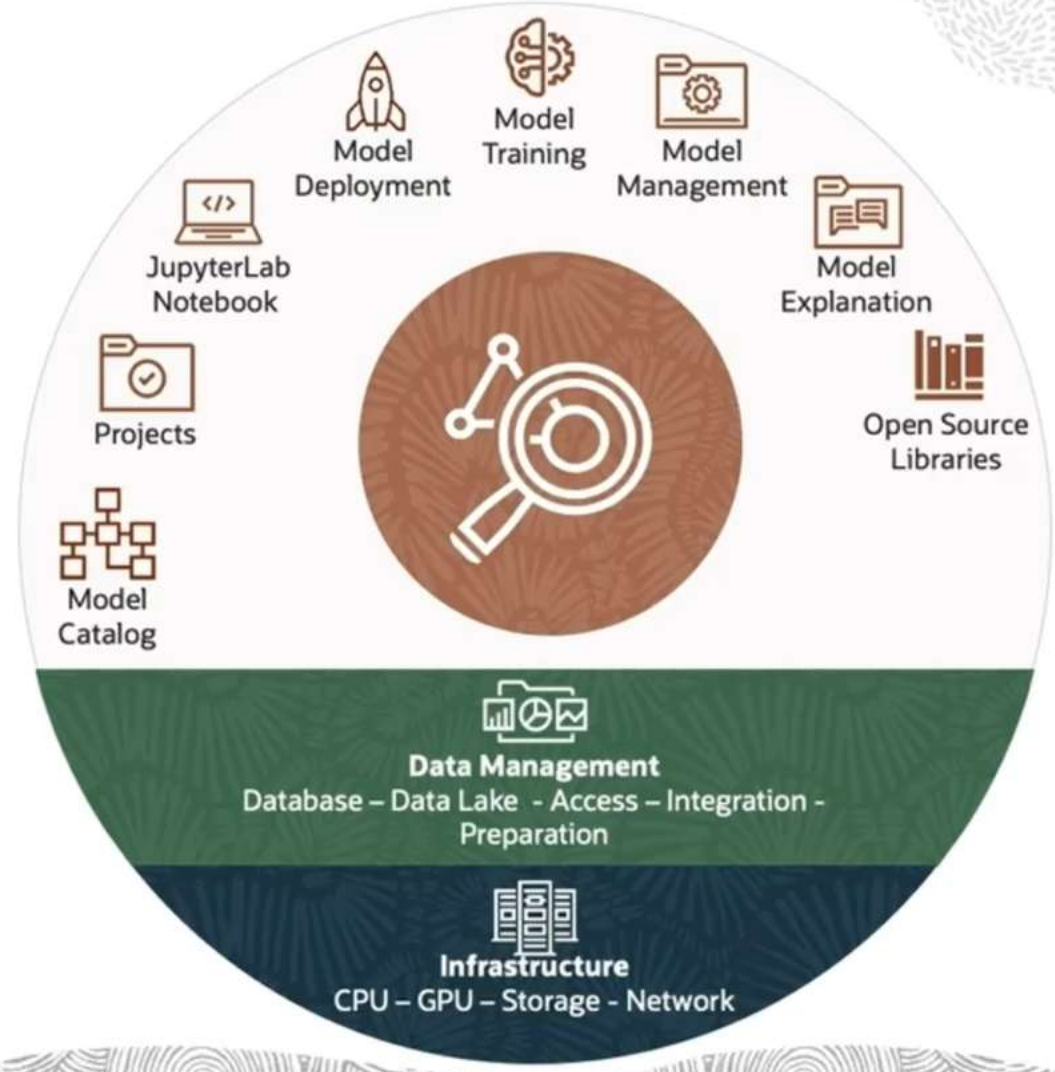
ML Services Overview

The Oracle AI Stack

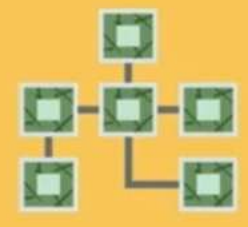


What is Oracle Cloud Infrastructure Data Science?

A cloud service focused on serving data scientists throughout the full machine learning life cycle, with support for Python and open-source libraries



Core Principles of OCI Data Science

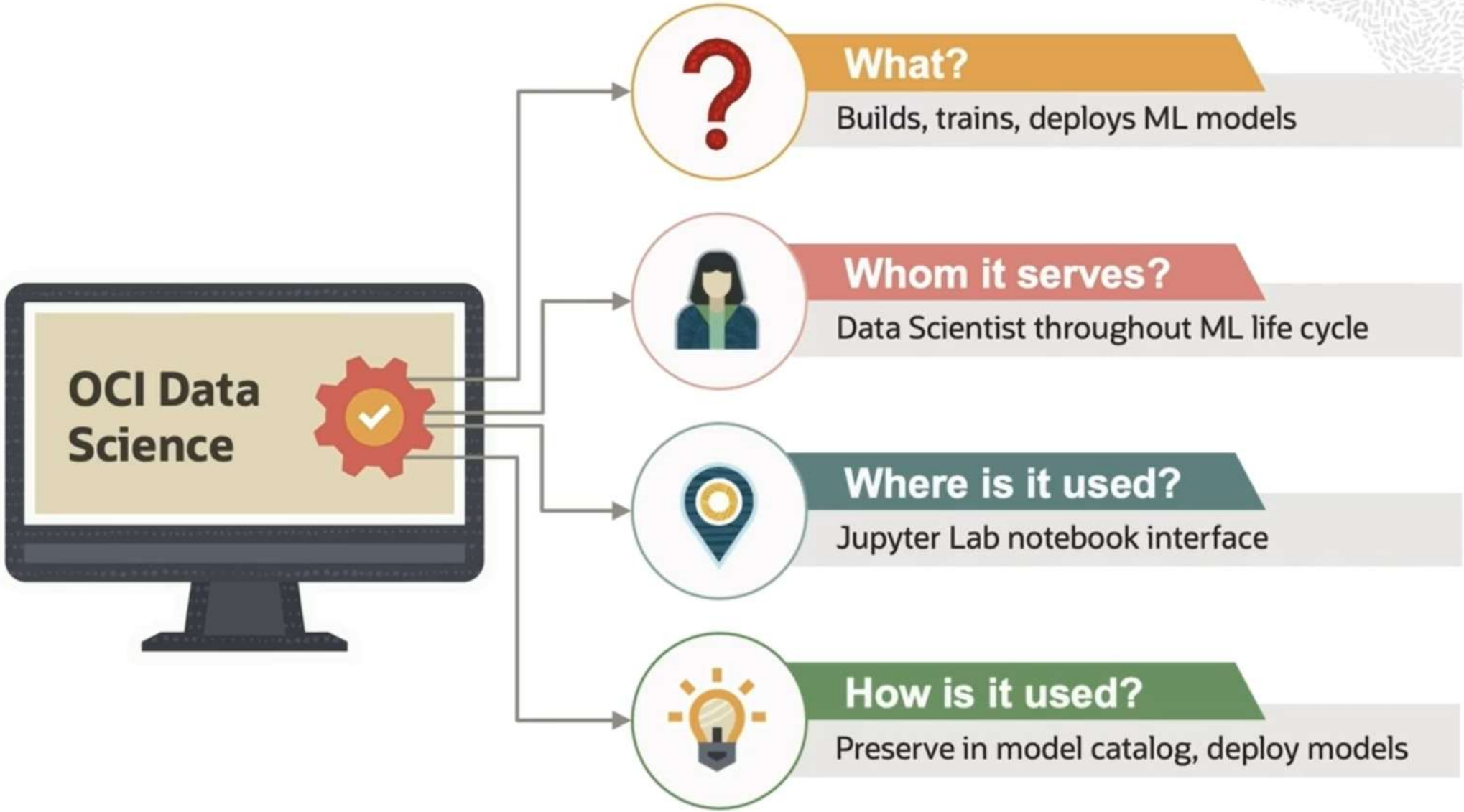


Accelerated: Allow data scientists to work how they want, and provide access to automated workflows, the best of open-source libraries, and a streamlined approach to building models

Collaborative: Enable data science teams to work together with ways to share and reproduce models in a structured, secure way for enterprise-grade results

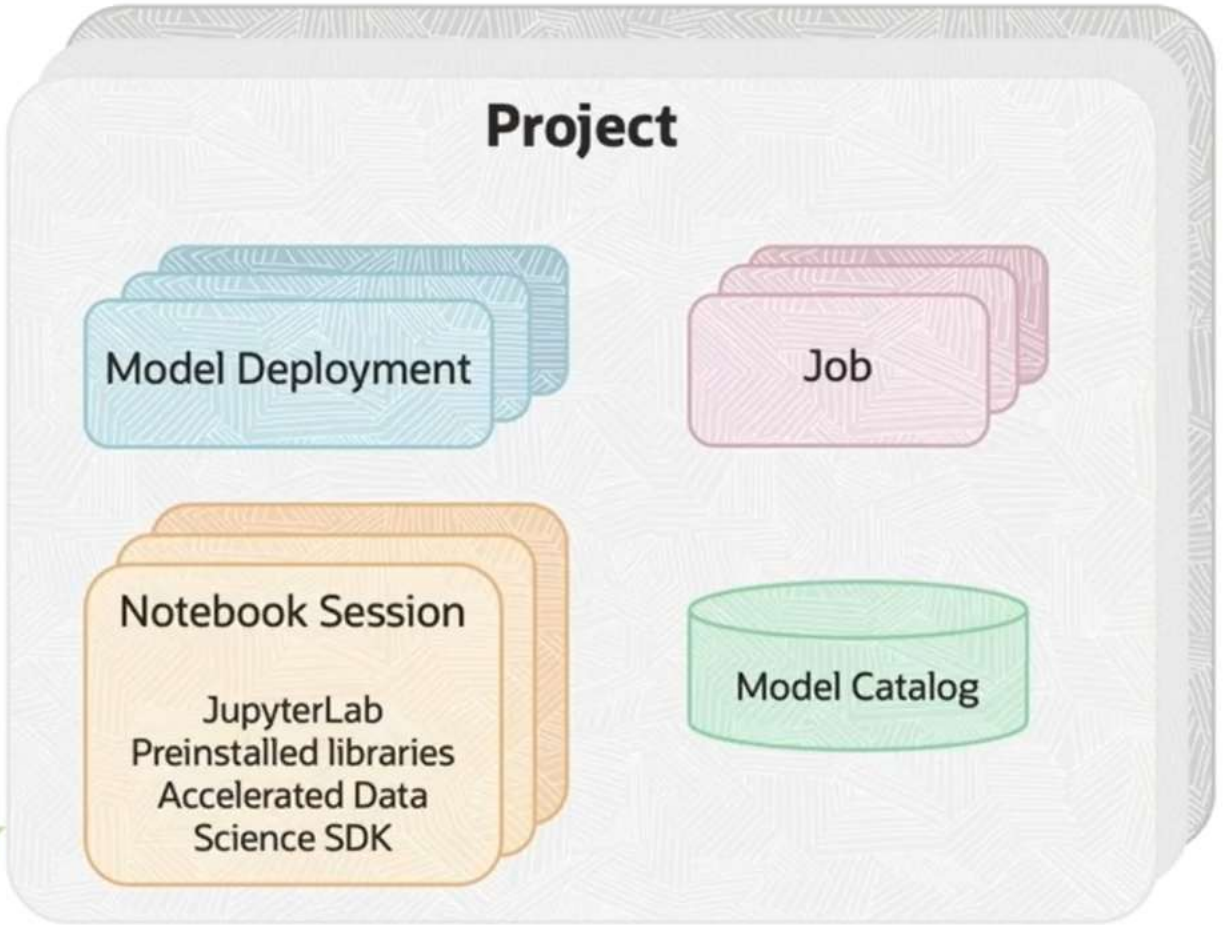
Enterprise-Grade: Provide a fully managed platform, built to meet the needs of the modern enterprise

What, Whom, Where, and How of Data Science



Data Science Features and Terminology

- Projects
- Notebook sessions
- Conda environments
- Accelerated Data Science (ADS) SDK
- Models
- Model catalog
- Model deployments
- Jobs

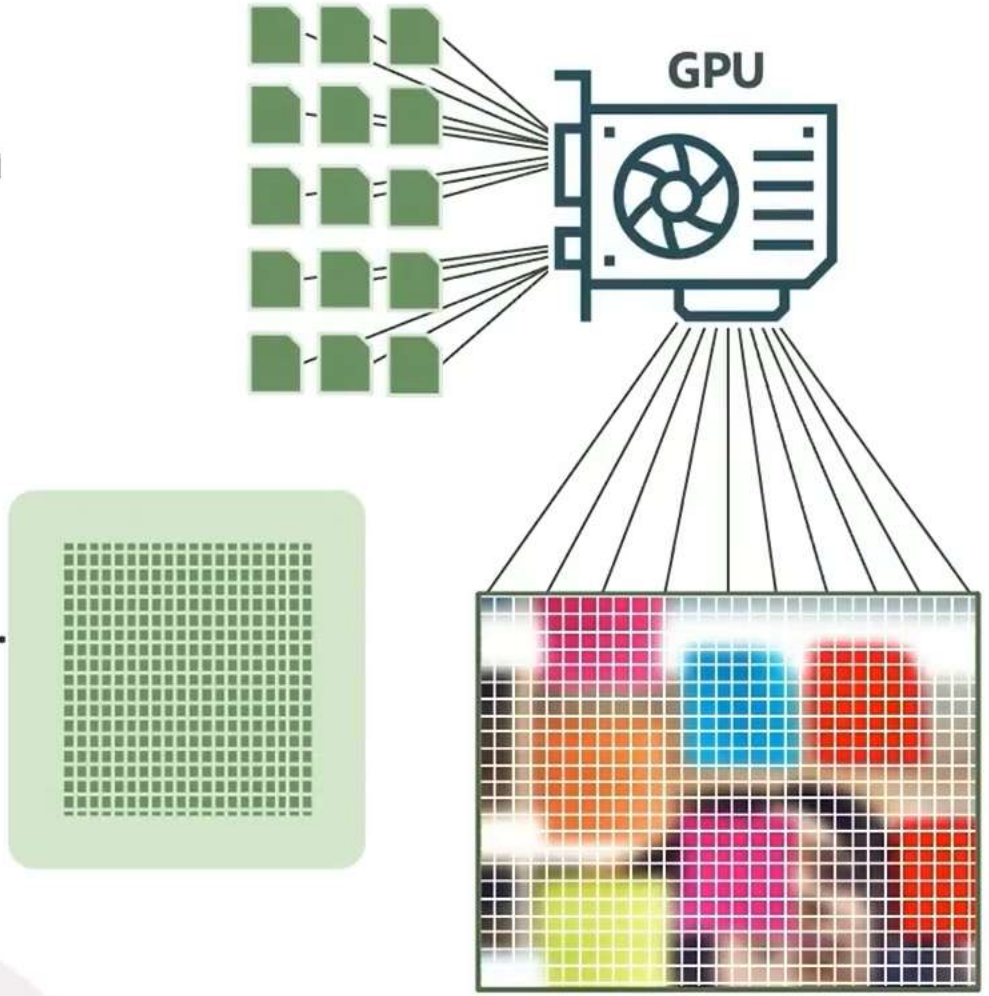


AI Infrastructure

AI Foundations

What is a GPU?

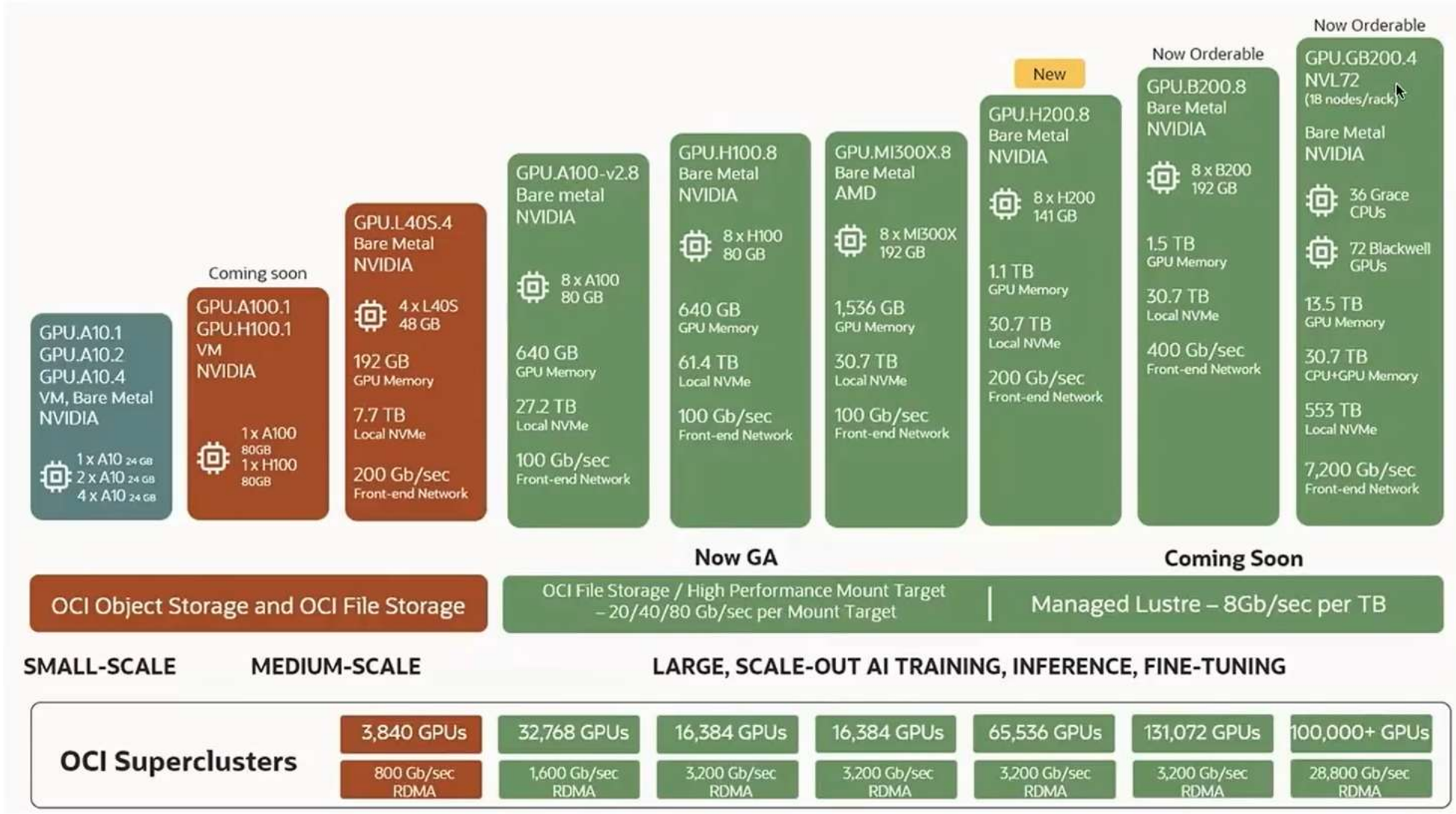
- Performs simple operations very rapidly.
- Allows many processes at the same time with parallel computing. Crunches through extremely large datasets at tremendous speeds.
- GPUs can process many inference tasks in parallel leading to higher throughput.
- GPUs are optimized for deep learning frameworks.
- These frameworks leverage GPU-specific libraries (e.g., CUDA, cuDNN) to accelerate inference.



NVIDIA GPU Comparison

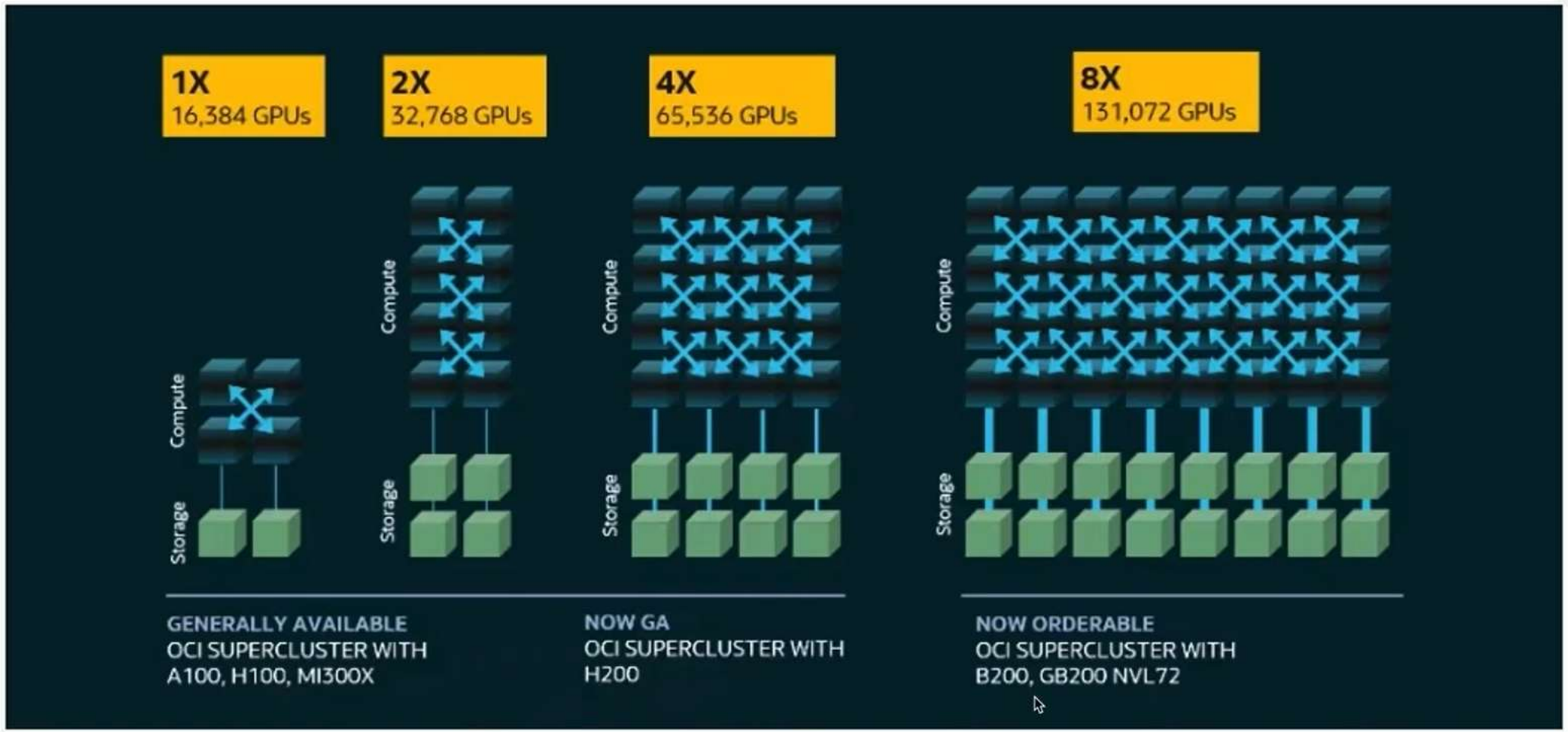
Feature	A100 (2020)	H100 (2022)	H200 (2024)	B200 (2025)	GB200 (2025)
Architecture	Ampere	Hopper	Hopper	Blackwell	Grace + Blackwell
Purpose	AI training & inference	Faster AI, supports bigger models	More memory for larger AI models	Cutting-edge AI, faster & more efficient	A supercomputer in a single chip
AI Speed	Fast	3x Faster than A100	Like H100 but with more memory	2x Faster than H100	Superfast, for the biggest AI models
Memory (Storage for AI models)	80 GB	80 GB	141 GB	192 GB	Integrated CPU & GPU memory
Efficiency	Good	Better	Better	Best for AI efficiency	Designed for massive AI workloads
Best Used For	AI research, cloud computing	Chatbots, AI apps, deep learning	Bigger AI models like GPT-4	Next-gen AI models, very powerful AI tasks	AI supercomputers, biggest AI models

OCI AI Infrastructure



OCI also offers older-generation NVIDIA P100 and V100 GPUs

OCI Supercluster with Nvidia Blackwell and Hopper GPUs



GPU Use Case

Model Training

Input

Untrained neural network

Training

OCI Data Science

Bare metal GPU Compute

OCI Storage

RDMA cluster Networking

Output

Trained model with new capability

1. An untrained model consists of random parameters that enable it to make a guess on the output based on the input.
2. Training is a resource-intensive process, often powered by OCI bare metal instances, OCI cluster networking based on RDMA, and NVIDIA GPUs. OCI Data Science can be used to build, train, and deploy models.
3. The best candidate models are saved for proof of concept, experimentation, or production deployment.

Model Inferencing

Input

"AI solving business problems"

Inferencing

Trained model (DALL-E 2)

OCI Data Science

GPU Compute

Output

Model output

4. The model is deployed for consumption using automation or a service such as OCI Data Science. New data is fed into the deployed model.
5. The trained model usually runs on GPU compute instances to provide accurate results at low latency.
6. Outputs are generated and can be optimized further by retraining and redeploying candidate models.

Responsible AI

Trustworthy AI

For us to trust AI, it must be driven by ethics` that guide us as well!

What are guiding principles for AI to be trustworthy?

AI should follow applicable laws.

AI should be ethical.

AI should be robust.