



مجموعة جامعة
الجميع الذكية
للتعليم القابضة

MIDOCEAN
UNIVERSITY
BY EVERYONE'S SMART UNIVERSITY

HCIA-AI Huawei

Lecture 10

AI Frameworks

Prepared by
Eng. Eman Elgallad

www.midocean.ae

Lecture

10

Content:

Introduction to AI Frameworks

Tensor

Types of AI Frameworks

Popular AI Frameworks

AI Frameworks

AI frameworks, also known as machine learning frameworks, are software libraries or platforms that offer a collection of tools, functions, and pre-built algorithms to facilitate the development of AI and machine learning applications. They are designed to simplify the complex tasks associated with training and deploying machine learning models, making it accessible to a broader audience, including AI, ML, DL and Data Science Engineers.

Pre-Built Functions and Libraries:

One of the primary advantages of AI frameworks is the availability of pre-built functions and libraries. These libraries contain a wide range of machine learning and deep learning algorithms, mathematical operations, and data manipulation tools. For instance, libraries like TensorFlow, PyTorch, and scikit-learn provide ready-to-use implementations of popular algorithms such as neural networks, decision trees, and support vector machines.

Importance of AI Frameworks:

AI frameworks hold significant importance in the development of AI and machine learning applications for several reasons:

- a. **Efficiency:** AI frameworks help developers avoid reinventing the wheel. They offer pre-built implementations of common tasks, allowing engineers to focus on high-level design and optimization rather than starting from scratch.
- b. **Scalability:** Many AI frameworks are designed to work efficiently with large datasets and distributed computing resources. This scalability is crucial for training complex models on extensive datasets.
- c. **Community Support:** Leading AI frameworks have vibrant communities of developers and researchers. This means that you can find extensive documentation, tutorials, and a wealth of resources to assist you in your AI projects.

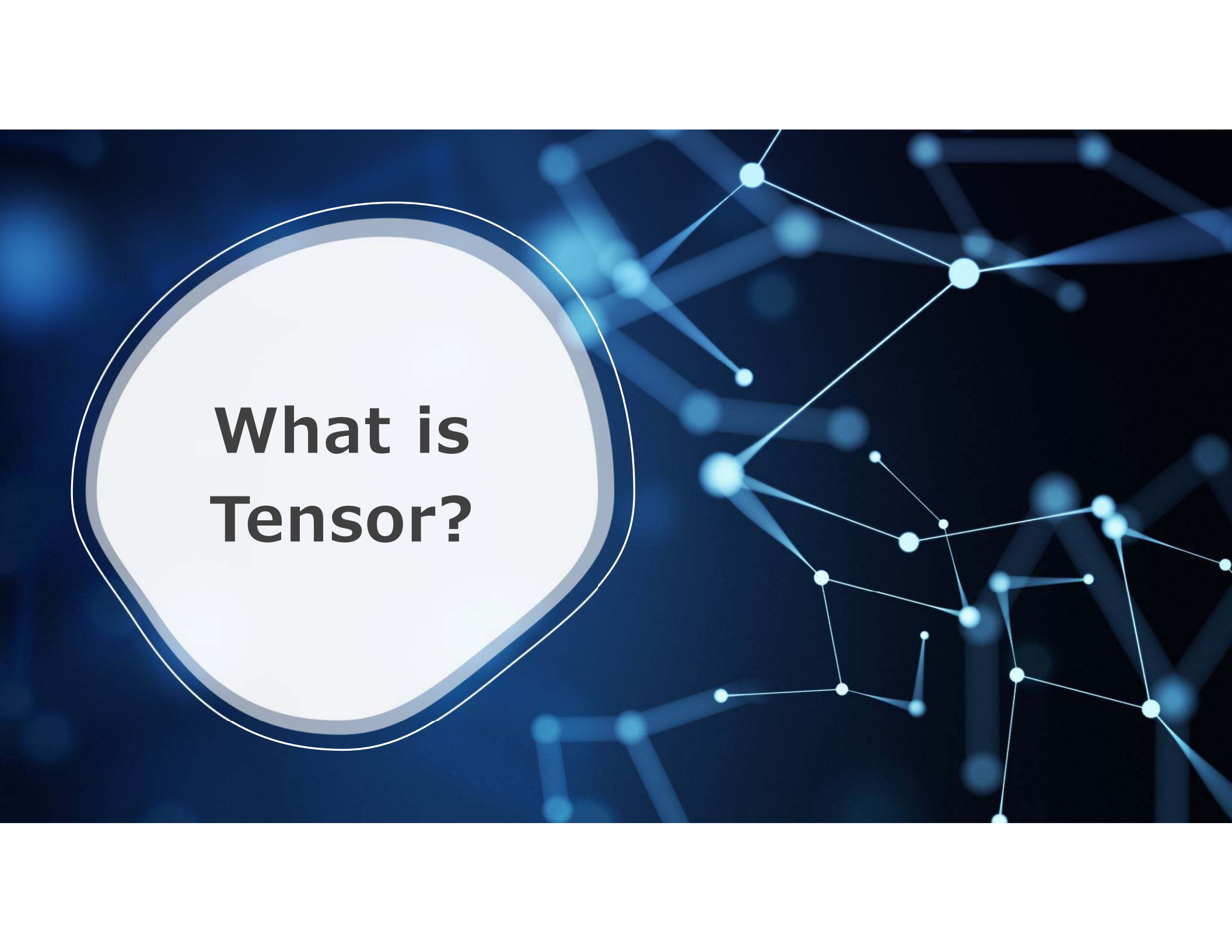
Importance of AI Frameworks Cont. :

- d. **Flexibility**: AI frameworks offer flexibility in terms of model architecture and customization. You can build and experiment with a wide range of models tailored to your specific use case.
- e. **Deployment**: Many AI frameworks provide tools for deploying models to various platforms, including cloud services and edge devices, making it easier to bring AI applications to the real world.

Acceleration of AI Development:

AI frameworks significantly accelerate AI development in several ways:

- a. **Rapid Prototyping:** With pre-built functions and libraries, developers can quickly prototype and test different machine learning models, saving time and resources.
- b. **Parallel Processing:** Many AI frameworks are optimized for parallel processing, allowing you to leverage multiple GPUs or TPUs to speed up model training.
- c. **AutoML:** Some frameworks incorporate AutoML (Auto Machine Learning) features that automate tasks like hyperparameter tuning and model selection, further reducing development time.
- d. **Transfer Learning:** AI frameworks often support transfer learning, enabling you to fine-tune pre-trained models for specific tasks, which is faster than training from scratch.



**What is
Tensor?**

What is Tensor?

Tensor is a multi-dimensional array or data structure that can hold and represent data in multiple dimensions. In essence, it is a generalization of scalars, vectors, and matrices to higher-dimensional spaces.



Key Characteristics of Tensors:

Rank or Order: The rank or order of a tensor refers to the number of dimensions it has. Scalars have a rank of 0, vectors have a rank of 1, matrices have a rank of 2, and tensors can have a rank greater than 2. For example, a 3D tensor has a rank of 3, indicating that it has three dimensions.

Shape: The shape of a tensor defines the number of elements along each dimension. For a 2D tensor (matrix), the shape might be (m, n) , where "m" represents the number of rows, and "n" represents the number of columns. In higher-dimensional tensors, the shape specifies the size along each dimension.

- A square matrix can have dimensions $(2, 2)$.
- The level 3 tensor $(3, 5, 8)$ can have dimensions.

Key Characteristics of Tensors:

Data Type: Tensors can hold data of various types, such as integers, floating-point numbers, or even more complex data types like strings. The data type of a tensor determines the kind of data it can store.

- float32
- float64
- uint8
- int32
- int64

Operations: Tensors support various mathematical operations, including addition, subtraction, multiplication, and more. These operations are defined to work element-wise, which means they operate on corresponding elements of tensors with the same shape.

How To Create Tensor Using NumPy?

NumPy is a Python library that can be used for scientific and numerical applications and is the tool to use for linear algebra operations.

→ When working with NumPy, data in an ndarray is simply referred to as an array. It is a fixed-sized array in memory that contains data of the same type, such as integers or floating point values

0D tensor using NumPy

0D

```
import numpy as np
tensor = np.array(42)
```

```
tensor.shape
```

```
()
```

```
tensor.ndim
```

```
0
```

1D tensor using NumPy

1D

```
import numpy as np
tensor = np.array([8, 16, 32, 64])
tensor.shape
```

```
(4,)
```

```
tensor.ndim
```

```
1
```

2D tensor using NumPy

2D

```
import numpy as np
tensor = np.array([[2, 10, 20, 22],
                  [8, 16, 32, 64],
                  [5, 10, 15, 20]])
```

```
tensor.shape
```

```
(3, 4)
```

```
tensor.ndim
```

```
2
```

Numpy functions:

- **empty():** It will create a new array of the specified shape with random numbers each time.
- **zeros():** It will create a new array of the specified size with the contents filled with zero values
- **ones():** It will create a new array of the specified size with the contents filled with one values.

```
import numpy as np
empty_arr = np.empty([2,1])
zeros_arr = np.zeros([4])
ones_arr = np.ones([2,1])
print(empty_arr)
print(zeros_arr)
print(ones_arr)
```

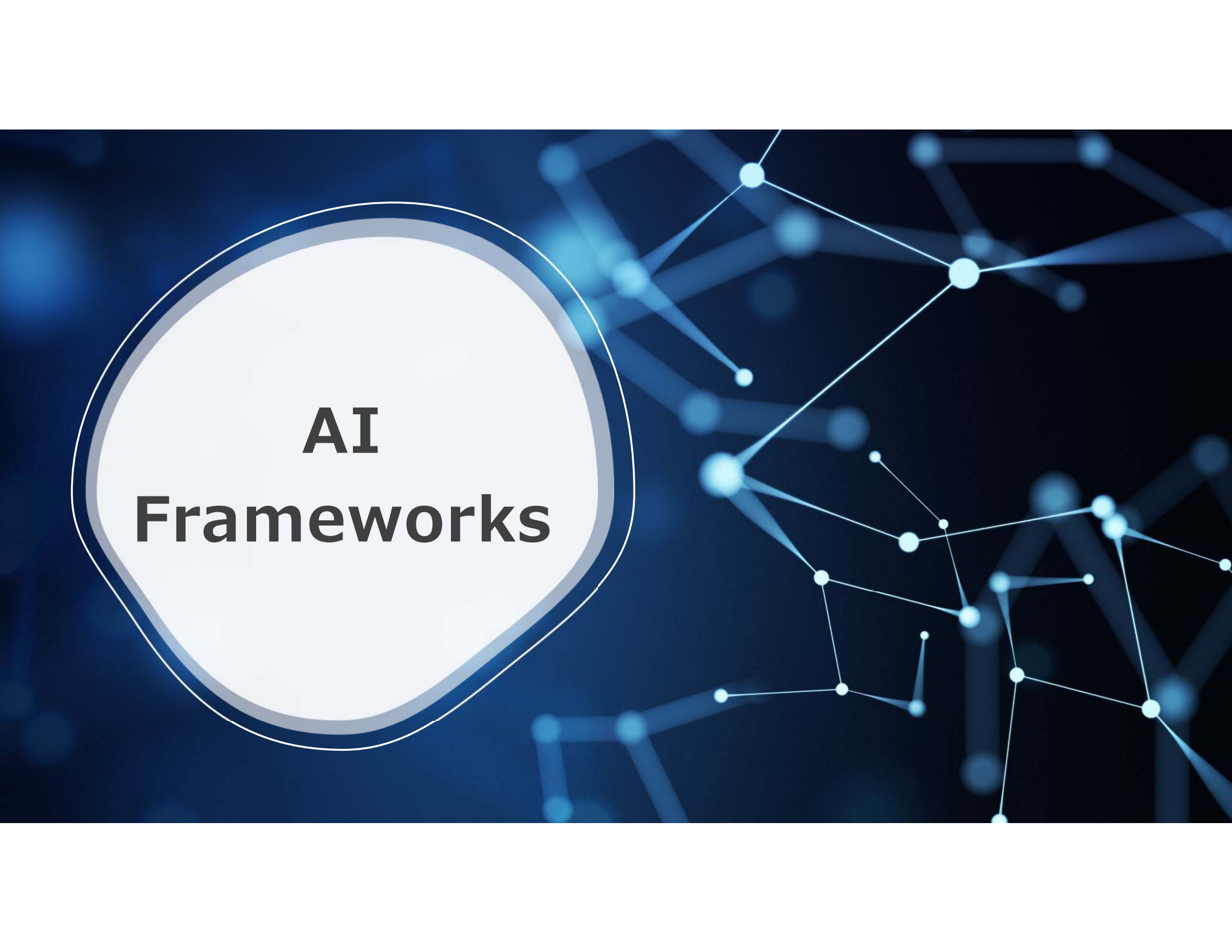
```
[[1.54087081e-173]
 [8.50147465e+260]]
[0. 0. 0. 0.]
[[1.]
 [1.]]
```

Numpy array Splitting data into X,y:

- It is common to split your loaded data into input variables (X) and the output variable (y).
- We can do this by slicing all rows and all columns up to, but before the last column, then separately indexing the last column.

```
data = np.array([[11, 22, 33],  
                 [44, 55, 66],  
                 [77, 88, 99]])  
  
X, y = data[:, :-1], data[:, -1]  
print(X)  
print(y)
```

```
[[11 22]  
 [44 55]  
 [77 88]  
 [33 66 99]]
```

The image features a dark blue background with a glowing network of white nodes and lines, suggesting a digital or AI theme. On the left side, there is a large, light blue, rounded shape with a double border. Inside this shape, the text "AI Frameworks" is written in a bold, black, sans-serif font.

AI Frameworks

categories of AI frameworks :

The two main categories of AI frameworks:

1-Deep Learning Frameworks: These are specialized for neural network-based deep learning tasks.

2-General-Purpose Frameworks: These are adaptable and support a wide range of machine learning tasks.



DL Frameworks:

Deep learning frameworks are specialized software tools that play a pivotal role in the development of artificial intelligence (AI) systems, particularly those based on neural networks. These frameworks provide a comprehensive ecosystem for designing, training, and deploying deep neural networks such as **TensorFlow, PyTorch, and Keras**.

Layer Abstractions: These frameworks provide high-level abstractions for creating different types of layers in neural networks, such as convolutional, recurrent, and fully connected layers. This simplifies the process of defining network architectures.

Automatic Differentiation: Deep learning frameworks implement automatic differentiation, allowing gradients to be computed efficiently during the training process. This is crucial for optimizing network parameters through techniques like backpropagation.

DL Frameworks:

Optimized GPU/TPU Support: Most deep learning frameworks are optimized to leverage powerful hardware accelerators like GPUs (Graphics Processing Units) and TPUs (Tensor Processing Units), speeding up the training of large neural networks significantly.

Pre-trained Models: Many deep learning frameworks offer access to pre-trained models, which can be fine-tuned for specific tasks. This transfer learning approach saves time and computational resources.

PyTorch

- PyTorch is a Python-based machine learning computing framework developed by Meta.
- It is developed based on Torch, a scientific computing framework supported by a large number of machine learning algorithms.
- Torch is a tensor operation library similar to NumPy, featured by high flexibility, but is less popular because it uses the programming language Lua. This is why PyTorch is developed.
- In addition to Meta, institutes such as Twitter, GMU also use PyTorch.

The PyTorch logo is displayed within a white rectangular box. It features the word "PYTORCH" in a bold, black, sans-serif font. The letter "O" is replaced by a stylized orange flame icon with a small purple dot above it.

<http://PyTorch123.com/FirstSection/PyTorchIntro/>

Features of PyTorch

- **Python first:** PyTorch does not simply bind Python to a C++ framework. PyTorch directly supports Python access at a fine grain. Developers can use PyTorch as easily as using NumPy or Pandas. This not only lowers the threshold for understanding Python, but also ensures that the code is basically consistent with the native Python implementation.
- **Dynamic neural network:** Many mainstream frameworks such as TensorFlow 1.x do not support this feature. To run TensorFlow 1.x, developers must create static computational graphs in advance, and run the feed and run commands to repeatedly execute the created graphs. In contrast, PyTorch with this feature is free from such complexity, and PyTorch programs can dynamically build/adjust computational graphs during execution.
- **Easy to debug:** PyTorch can generate dynamic graphs during execution. Developers can stop an interpreter in a debugger and view output of a specific node.
- **PyTorch provides tensors that support CPUs and GPUs, greatly accelerating computing.**

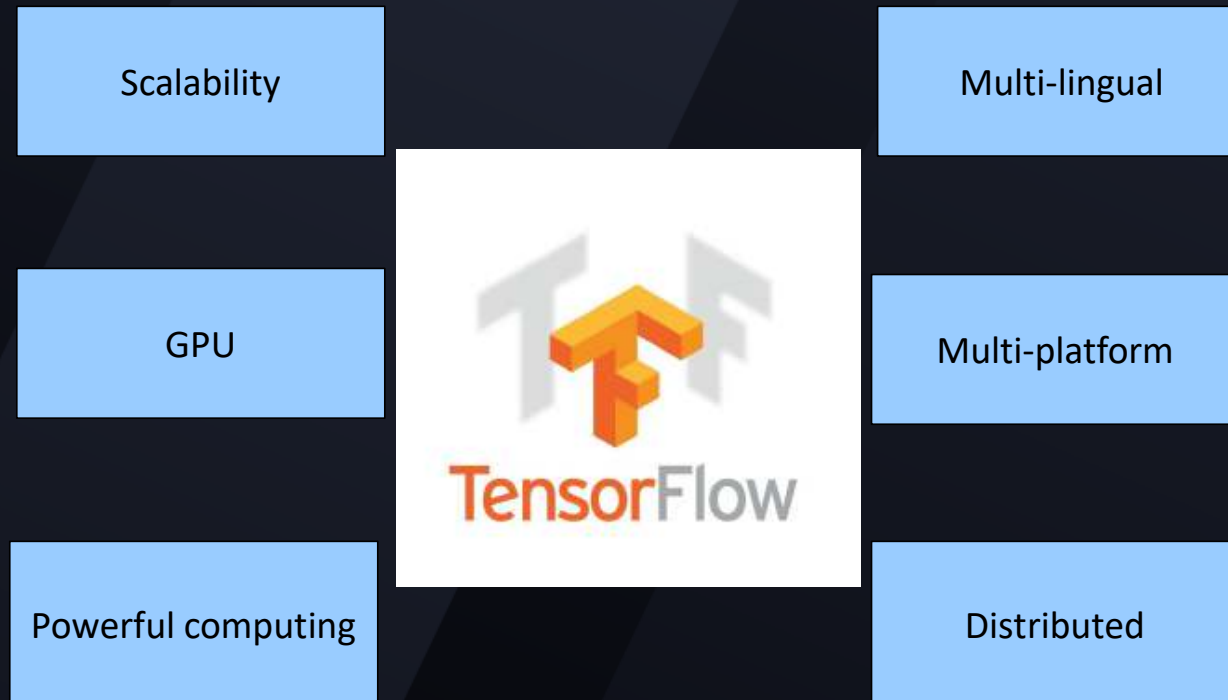
TensorFlow:

TensorFlow: It is an open-source software library for numerical computation. Developed by Google, TensorFlow is one of the most widely used deep learning frameworks. It provides a versatile platform for building various types of neural networks and is known for its scalability, making it suitable for both research and production applications.



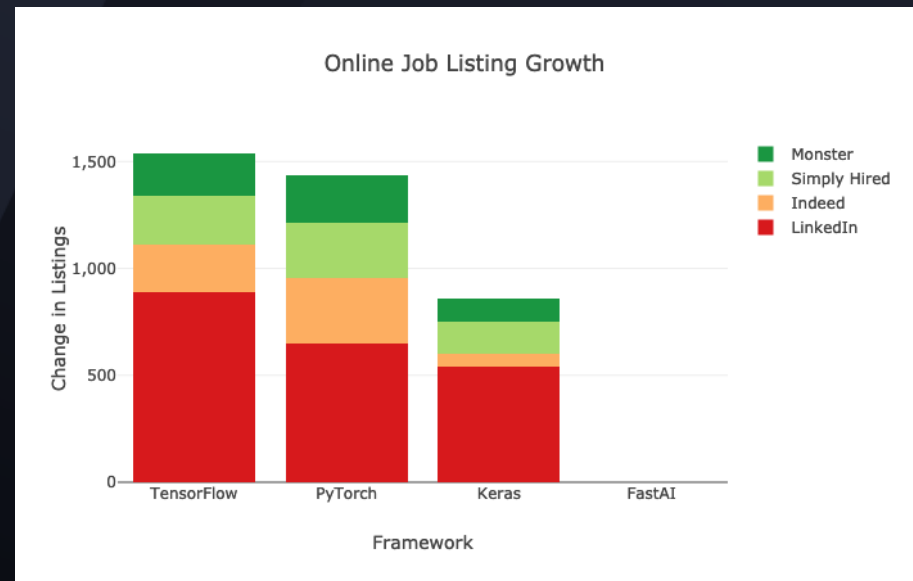
<https://www.TensorFlow.org/>

Features of TensorFlow



Why TensorFlow?

- TensorFlow is considered as one of the best libraries for neural networks and can reduce difficulty in deep learning development. In addition, as it is open-source, it can be conveniently maintained and updated, thus the efficiency of development can be improved.
- Keras, ranking third in the number of stars on GitHub, is packaged into an advanced API of TensorFlow 2.0, which makes TensorFlow 2.x more flexible, and easier to debug.



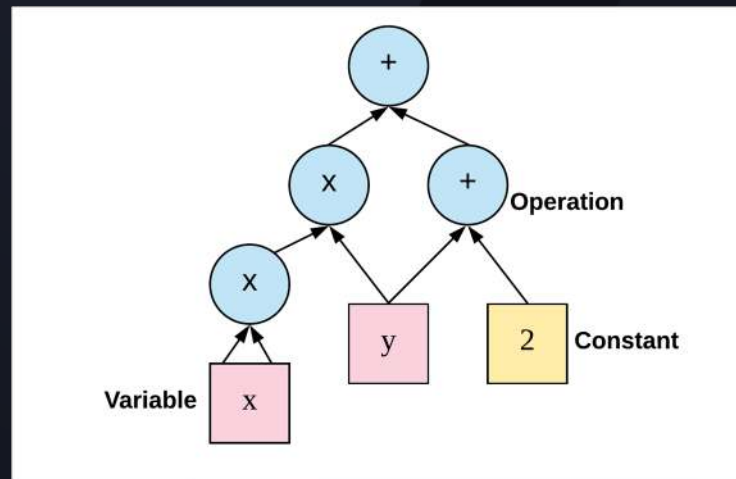
Demand on the
recruitment market

TensorFlow 2.x vs. TensorFlow 1.x

- Disadvantages of TensorFlow 1.0:
 - After a tensor is created in TensorFlow 1.0, the result cannot be returned directly. To obtain the result, the session mechanism needs to be created, which includes the concept of graph, and code cannot run without session.run. This style is more like the hardware programming language VHDL.
 - Compared with some simple frameworks such as PyTorch, TensorFlow 1.0 adds the session and graph concepts, which are inconvenient for users.
 - It is complex to debug TensorFlow 1.0, and its APIs are disordered (mixed), making it difficult for beginners. Learners will come across many difficulties in using TensorFlow 1.0 even after gaining the basic knowledge. As a result, many researchers have turned to PyTorch.

TF1

- TensorFlow 1 Graph consist of nodes and edges
- Session consist of two operations fetch for retrieving values and feed to give nodes value
- Tensor (name , shape , Type)



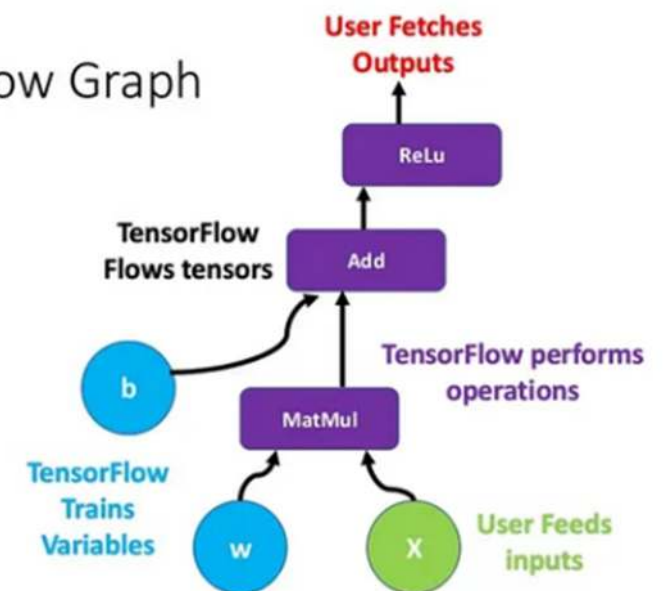
For example, this image represents a graph in TensorFlow. **W**, **x** and **b** are tensors over the edges of this graph.

MatMul is an operation over the tensors **W** and **x**, after that

Add is called and add the result of the previous operator with **b**.

The resultant tensors of each operation cross the next one until the end where it's possible to get the wanted result.

TensorFlow Graph



TensorFlow 2.x vs. TensorFlow 1.x

- **Features of TensorFlow 2.x:**
 - Advanced API Keras:
 - Easy to use: The graph and session mechanisms are removed. What you see is what you get, just like Python and PyTorch.
- **Major improvements:**
 - The core function of TensorFlow 2.x is the dynamic graph mechanism called eager execution. It allows users to compile and debug models like normal programs, making TensorFlow easier to learn and use.
 - Multiple platforms and languages are supported, and compatibility between components can be improved via standardization on exchange formats and alignment of APIs.
 - Compatibility and continuity: TensorFlow 2.x provides a module enabling compatibility with TensorFlow 1.x.
 - The tf.contrib (It is meant to contain features and contributions that eventually should get merged into core TensorFlow) module is removed. Maintained modules are moved to separate repositories. Unused and unmaintained modules are removed.

Basic Operations of TensorFlow 2.x

- The following describes common APIs in TensorFlow by focusing on code. The main content is as follows:
 - Methods for creating constants and variables
 - Tensor slicing and indexing
 - Dimension changes of tensors
 - Arithmetic operations on tensors
 - Tensor concatenation and splitting
 - Tensor sorting

Auto Graph

- Eager execution is enabled in TensorFlow 2.x by default. Eager execution is intuitive and flexible for users (easier and faster to run a one-time operation), but may compromise performance and deployability.
- Eager execution is a command-based programming method, which is the same as native Python. A result is returned immediately after an operation is performed.
- To achieve optimal performance and make a model deployable anywhere, you can run `@tf.function` to add a decorator to build a graph from a program, making Python code more efficient.
- `tf.function` can build a TensorFlow operation in the function into a graph. In this way, this function can be executed in graph mode. Such practice can be considered as encapsulating the function as a TensorFlow operation of a graph.

TensorFlow Importing :

Importing TensorFlow

To use TensorFlow, we need to import the library. We imported it and optionally gave it the name “tf”, so the modules can be accessed by **tf.module-name**

.

```
import tensorflow as tf
```

Keras Interface

TensorFlow 2.x recommends Keras for network building. Common neural networks are included in **Keras.layers**.

Keras is the high-level API of the TensorFlow platform. It provides an approachable, highly-productive interface for solving machine learning (ML) problems, with a focus on modern deep learning.

Keras covers every step of the machine learning workflow, from data processing to hyperparameter tuning to deployment. It was developed with a focus on enabling fast experimentation.

Keras Interface

Keras is designed to reduce cognitive load by achieving the following goals:

- Offer simple, consistent interfaces.
- Minimize the number of actions required for common use cases.
- Provide clear, actionable error messages.
- Follow the principle of progressive disclosure of complexity: It's easy to get started, and you can complete advanced workflows by learning as you go.
- Help you write concise, readable code.

Common Keras Methods and Interfaces

- The following describes common methods and interfaces of tf.keras by focusing on code. The main content is as follows:
 - Dataset processing: datasets and preprocessing
 - Neural network model creation: Sequential, Model, Layers...
 - Network compilation: compile, Losses, Metrics, and Optimizers
 - Network training and evaluation: fit, fit_generator, and evaluate

Common Keras Methods and Interfaces

The **tf.keras.Model** class features built-in training and evaluation methods:

- **tf.keras.Model.fit**: Trains the model for a fixed number of epochs.
- **tf.keras.Model.predict**: Generates output predictions for the input samples.
- **tf.keras.Model.evaluate**: Returns the loss and metrics values for the model; configured via the **tf.keras.Model.compile** method.

General Purpose Frameworks:

General-purpose AI frameworks are versatile tools that support a broader range of machine learning tasks, including both traditional machine learning algorithms and deep learning. They are not limited to neural networks and can handle various types of data analysis and modeling.
example: **Scikit-Learn**

<https://scikit-learn.org/stable/>



General Purpose Frameworks:

Scikit is Python package that can perform many useful machine learning tasks:

- Linear regression
- Decision tree regressions
- Random Forest regressions
- K-Nearest neighbor
- SVMs
- Stochastic Gradient Descent models And more

Scikit provides model analysis tools like the confusion matrix for assessing how well a model performed. Many times, you can start an ML job in scikit-learn and then move to another framework.

For example, scikit-learn has excellent data pre-processing tools for one-hot encoding categorical data. Once the data is pre-processed through Scikit, you can move it into TensorFlow or PyTorch.

Choosing the right framework :

Choosing the right framework for your AI project is a crucial decision that can impact the efficiency and success

of your work. Here are the steps to help to choose the most suitable framework:

- **Understand Your Project Requirements:**

Start by clearly defining the goals and requirements of your project. Consider factors such as:

The nature of your data (structured, unstructured, images, text).

The complexity of the model you need (simple regression, deep neural network).

The scale of your project (research, prototype, production).

Any specific libraries or tools required for your domain (e.g., natural language processing libraries for NLP tasks).

Choosing the right framework Cont. :

- **Know Your Skill Set:**

Evaluate your team's expertise and familiarity with different frameworks. Choose a framework that aligns with your team's skills to reduce the learning curve and increase productivity.

- **Consider Community and Support:**

Check the availability of community support and resources for the frameworks you're considering. A strong community can provide valuable documentation, tutorials, and assistance when you encounter problems.

- **Scalability and Performance:**

Assess the scalability and performance requirements of your project. Some frameworks are optimized for distributed computing and can handle large datasets more efficiently. Consider whether you need GPU or TPU support for accelerated training.



مجموعة جامعة
الجميع الذكية
للتعليم القابضة

MIDOCEAN
UNIVERSITY
BY EVERYONE'S SMART UNIVERSITY



Thank You

www.midocean.ae