



TENSORFLOW & KERAS

OVERVIEW

01

What is TensorFlow?

02

Objective of
TensorFlow

03

TensorFlow Basics

04

How Does
TensorFlow work?

05

How to install
TensorFlow?

06

Building Neural
Networks with
TensorFlow

07

Integrating
TensorFlow with
other Libraries



08

What is Keras?

09

Keras API
Components

10

Building a Model
in Keras

11

How to install
Keras?

What is TensorFlow?

01

TensorFlow is an end-to-end platform for machine learning (ML)

02

TensorFlow makes it easy to create machine learning (ML) models that can run in any environment.

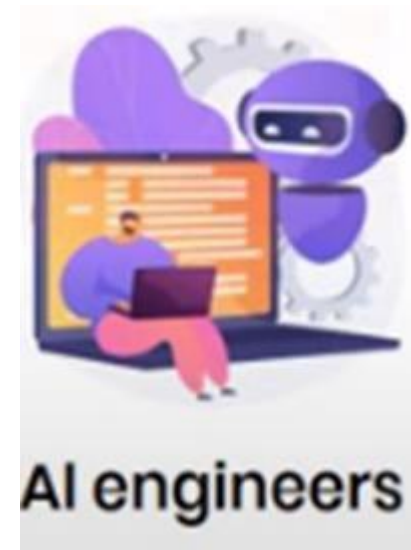
03

TensorFlow facilitates Estimation of Output at various scales



TensorFlow is the most in-demand tools for:

➤ Artificial Intelligent Engineers



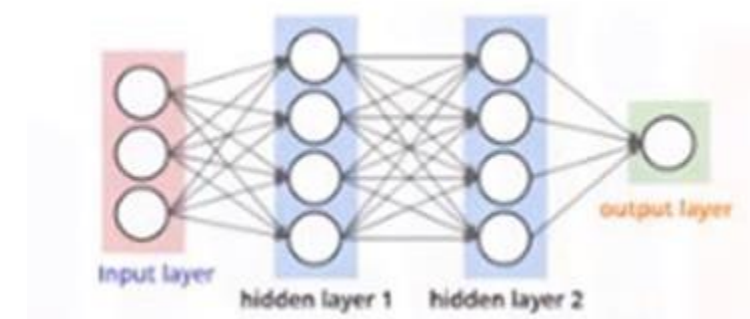
➤ Machine Learning Engineers



What is the objective of TensorFlow?

TensorFlow trains and executes:

➤ Neural Network Image Recognition



➤ Natural Language Processing



➤ Digit Classification



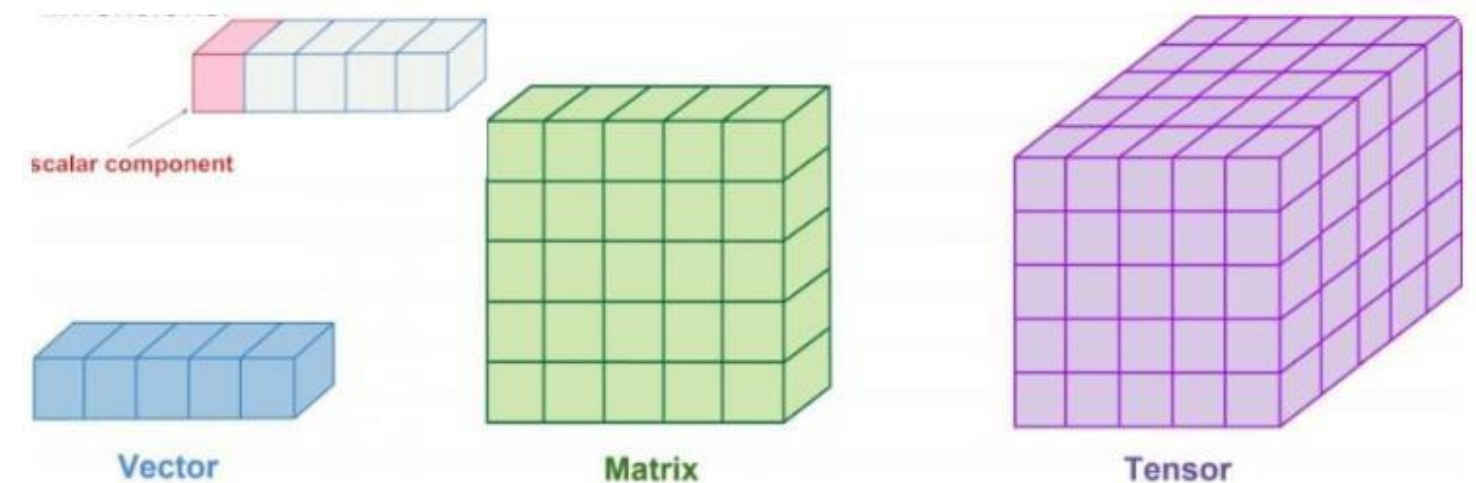
TensorFlow Basics

Tensors

TensorFlow operates on multidimensional arrays or tensors represented as `tf.Tensor` objects.

A tensor can have n dimensions.

A vector, for example, is a one-dimensional tensor, a matrix is a two-dimensional tensor, and a scalar is a zero-dimensional tensor.



Example:

Here is a two-dimensional tensor:

```
import tensorflow as tf

x = tf.constant([[1., 2., 3.],
                 [4., 5., 6.]])

print(x)
print(x.shape)
print(x.dtype)
```

The most important attributes of a `tf.Tensor` are its shape and dtype.

Shape of a Tensor:

`Tensor.shape`: tells you the size of the tensor along each of its axes.
The number of elements in each dimension defines a tensor's shape.

Scalar

Vector

Matrix

Tensor

1

$\begin{bmatrix} 1 \\ 2 \end{bmatrix}$

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$

$\begin{bmatrix} \begin{bmatrix} 1 & 2 \end{bmatrix} & \begin{bmatrix} 3 & 2 \end{bmatrix} \\ \begin{bmatrix} 1 & 7 \end{bmatrix} & \begin{bmatrix} 5 & 4 \end{bmatrix} \end{bmatrix}$

Type of a Tensor:

Tensor.dtype: tells you the type of all the elements in the tensor.

TensorFlow datatypes are as follows:

- integers
- floating point
- unsigned integers
- booleans
- strings
- integer with quantized ops
- complex numbers

TensorFlow's Basic Programming Elements

TensorFlow Python allows us to assign data to two kinds of data elements:

1. Constants

Constants are parameters with fixed values.
It is defined using the command `tf.constant()`.

2. Variables

Variables are created and tracked via the `tf.Variable()` class.
A `tf.Variable()` represents a tensor whose value can be changed.

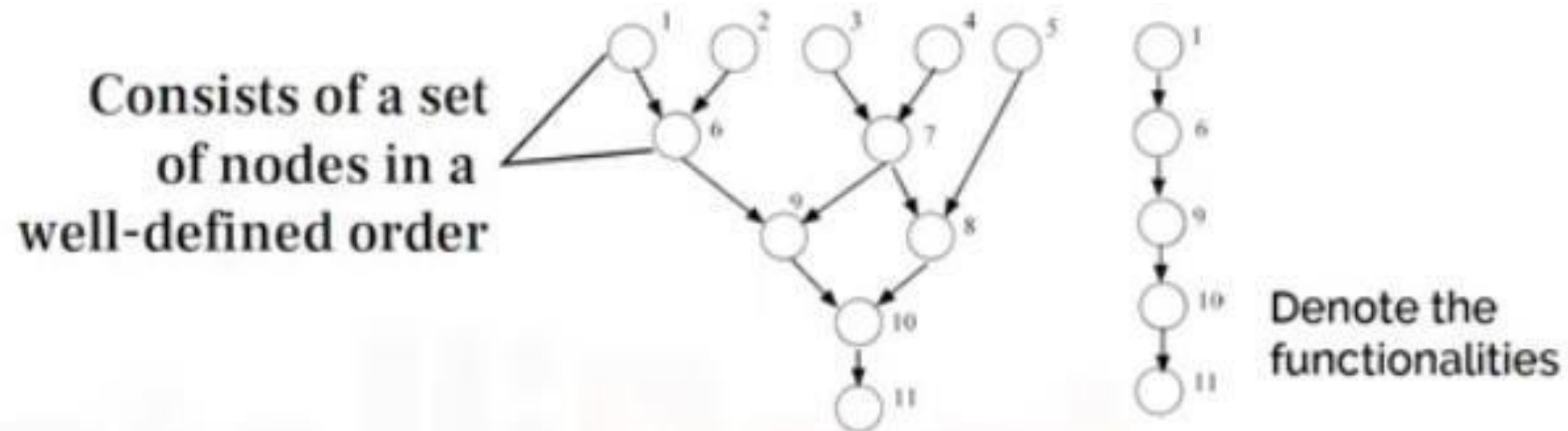
To create a variable, you need to provide an initial value.
The `tf.Variable` will have the same dtype as the initialization value.

Example:

```
my_tensor = tf.constant([[1.0, 2.0], [3.0, 4.0]])  
my_variable = tf.Variable(my_tensor)  
  
# Variables can be all kinds of types, just like tensors  
bool_variable = tf.Variable([False, False, False, True])  
complex_variable = tf.Variable([5 + 4j, 6 + 1j])
```

How does TensorFlow work?

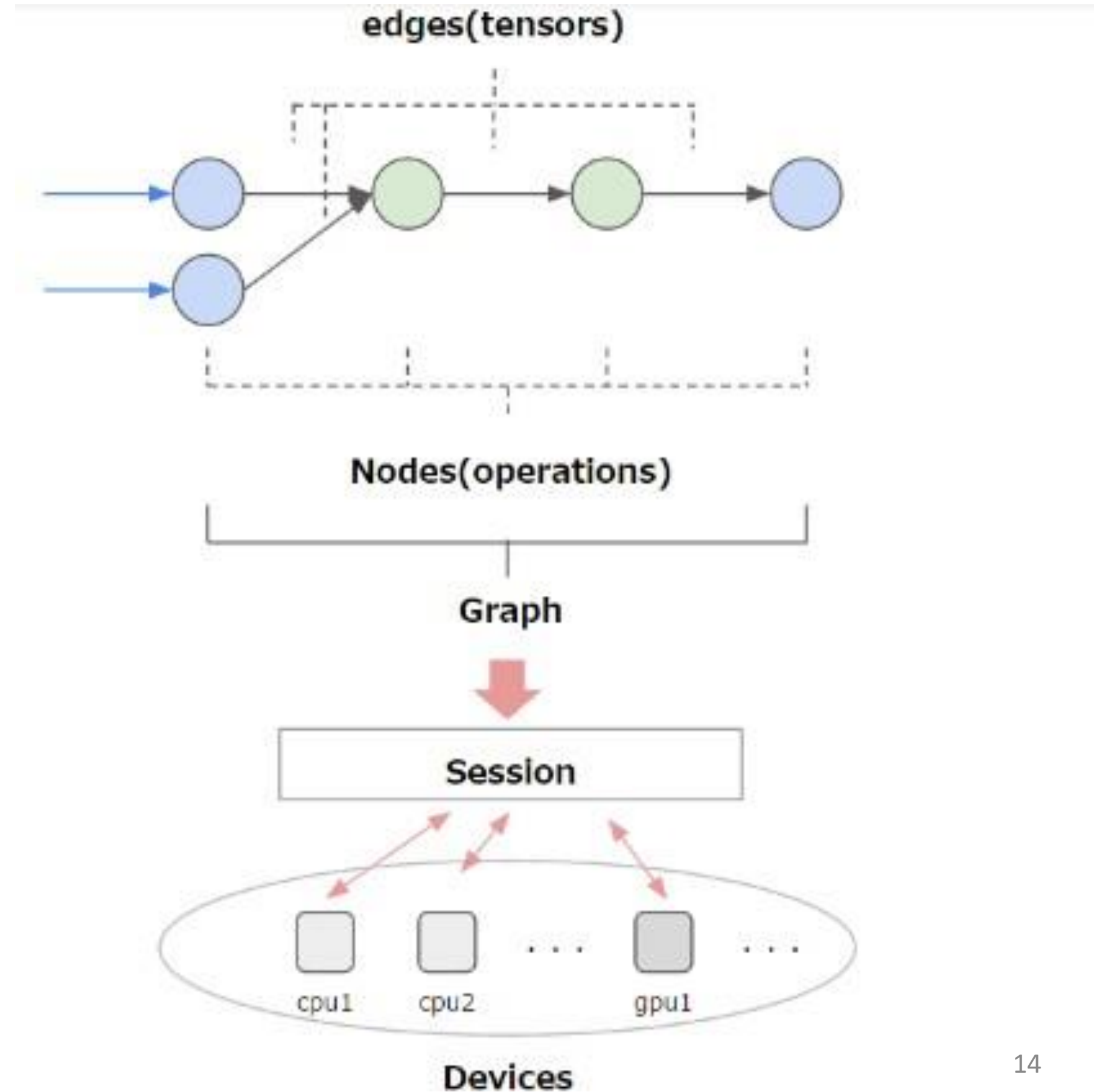
TensorFlow provides a feature that helps to create structures and machine learning models using a Dataflow graphs.



Each node in the graph symbolizes a mathematical process.

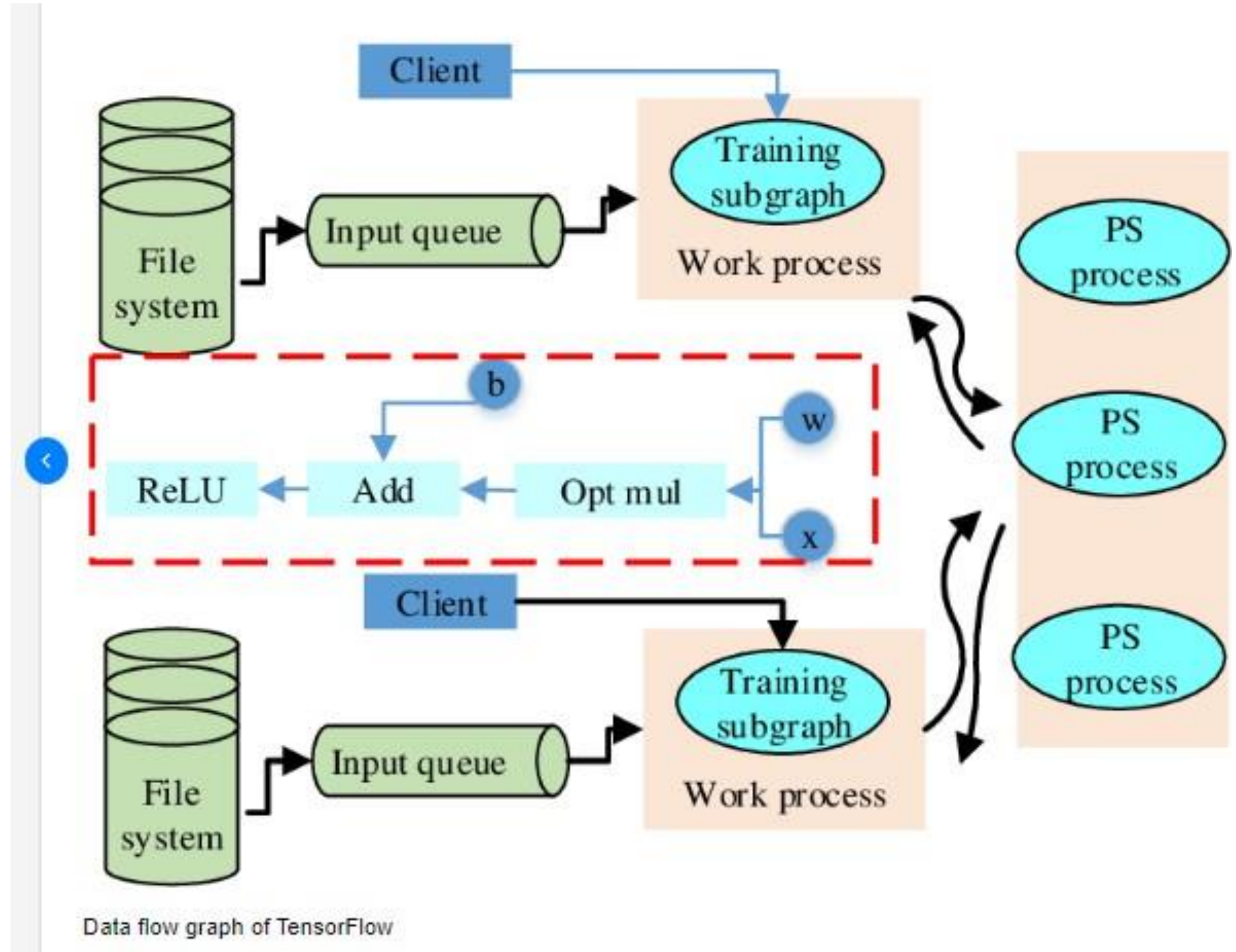
Each link or edge between nodes represents a tensor, which is a multidimensional data array.

TensorFlow makes all this available to developers via the Python language.





TensorFlow enables users to build dataflow graphs, which are structures that represent how data flows across a graph or a set of processing nodes.



How to install TensorFlow?

1. Check if Python is Installed:

First, ensure you have Python installed on your computer.

2. Install TensorFlow:

You can install it using a tool called “pip”, which comes with Python. Open a command prompt or terminal and type:

```
$ pip install --upgrade pip
```

```
$ pip install tensorflow
```

3. Check Installation:

Once the installation is complete, you can check if TensorFlow is installed correctly

```
import tensorflow as tf  
print("TensorFlow version:", tf.__version__)
```

Building Neural Networks with TensorFlow

01

Define Model

The first step in building a neural network with TensorFlow is to define the model architecture, including the input and output layers, as well as any hidden layers and activation functions.

02

Compile Model

Once the model is defined, you need to compile it by specifying the loss function, optimizer, and any additional metrics you want to track during training.

03

Train Model

With the model compiled, you can then train it on your dataset, leveraging TensorFlow's efficient training pipelines and optimization techniques to achieve optimal performance.

Integrating TensorFlow with other Libraries

1

Keras

Keras is a high-level neural networks API that runs on top of TensorFlow, providing a user-friendly and intuitive interface for building and training machine learning models.

2

Pandas

Pandas is a popular data manipulation and analysis library that can be integrated with TensorFlow, allowing developers to preprocess and prepare their data for machine learning tasks.

3

Scikit-learn

Scikit-learn is a powerful machine learning library that can be used in conjunction with TensorFlow, providing a wide range of algorithms and utilities for tasks like classification, regression, and clustering.

4

PyTorch

While TensorFlow is a powerful deep learning framework, it can also be integrated with PyTorch, another popular machine learning library, enabling developers to leverage the strengths of both frameworks in their projects.



What is Keras?

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.

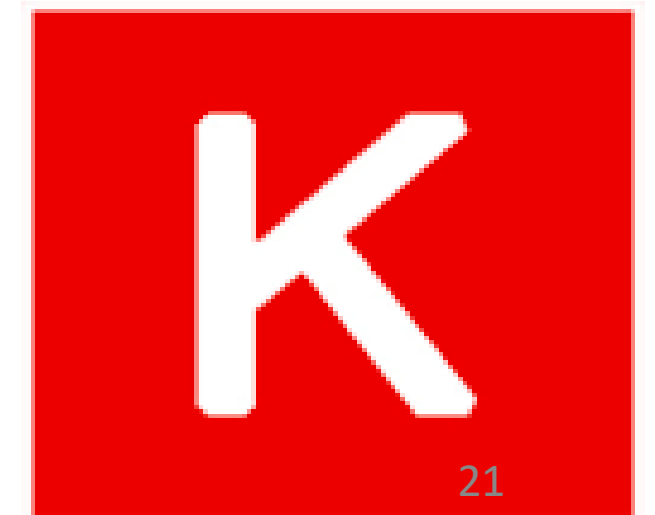
Keras covers every step of the machine learning workflow, from data processing to hyperparameter tuning to deployment. It was developed with a focus on enabling fast experimentation.



Why Keras?

Keras is a powerful and easy to use open-source API for developing complex deep learning models

It wraps efficient numerical computation libraries Theano and TensorFlow and allows you to define and train neural network in a few lines of code





Keras API components

The core data structures of Keras are layers and models.

Layers

A layer is a simple input/output transformation.

The `tf.keras.layers.Layer` class is the fundamental abstraction in Keras.

You can use layers to handle data preprocessing tasks like normalization and text vectorization.

Models

A model is an object that groups layers together and that can be trained on data.

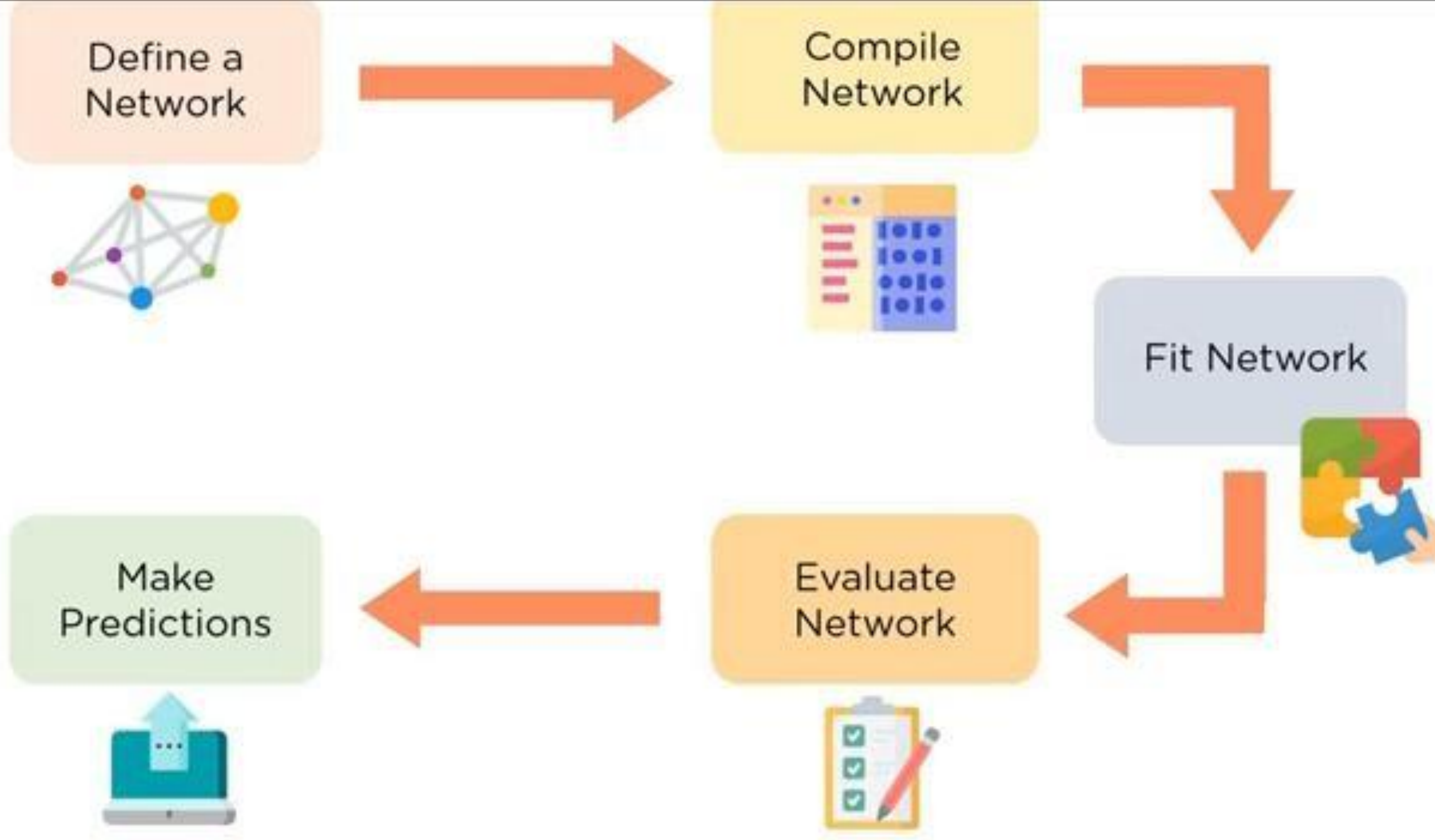
The `tf.keras.Model` class features built-in training and evaluation methods:

- `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.





Building a model in Keras



How to install Keras?

You can install Keras from PyPI via:

```
pip install --upgrade keras
```

You can check your local Keras version number via:

```
import keras  
print(keras.__version__)
```





Resources

1. TensorFlow Official Website (<https://www.tensorflow.org/>)
2. Keras Official Website (<https://www.keras.io/>)



**FACULTY OF
ENGINEERING**

RL4Eng



Co-funded by the
Erasmus+ Programme
of the European Union

THANK YOU