

Part I

Fundamentals

Introduction to TensorFlow

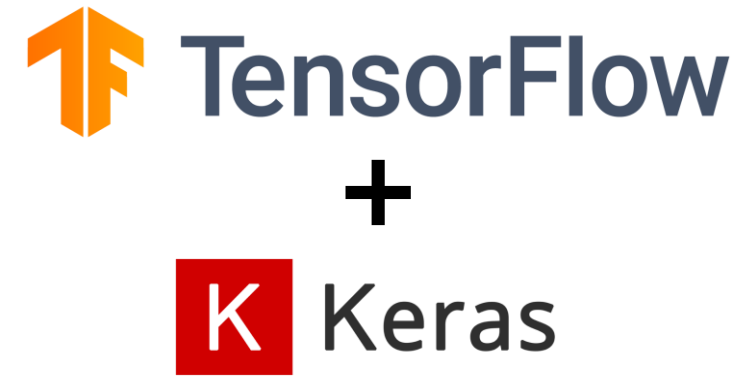
M.Sc. M. Fischer

Prof. Dr.-Ing. B. Yang



Introduction to TensorFlow

- Main objectives:
 - Get to know the underlying concepts
 - Get to know the provided functionality
- Try it out **yourselves!**
 - Several notebooks
 - Programming assignments
 - Online resources

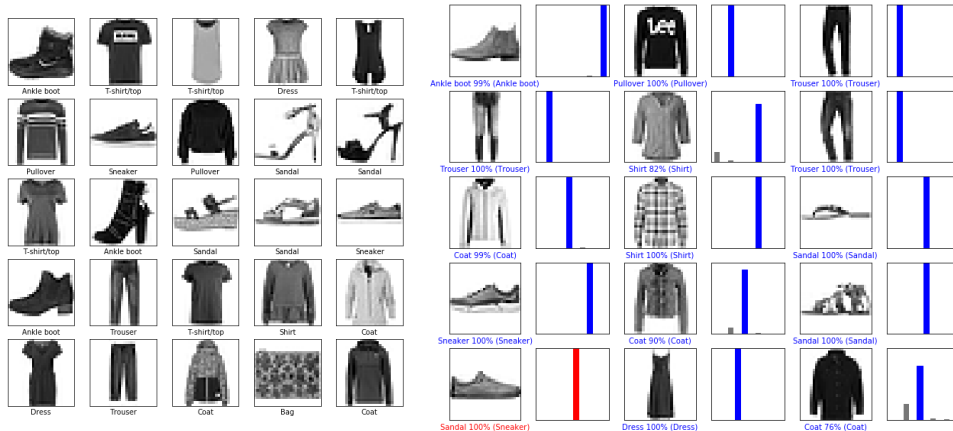


Introduction to TensorFlow

Examples

Image Classification

CNN



Unpaired Image Translation

Cycle GAN

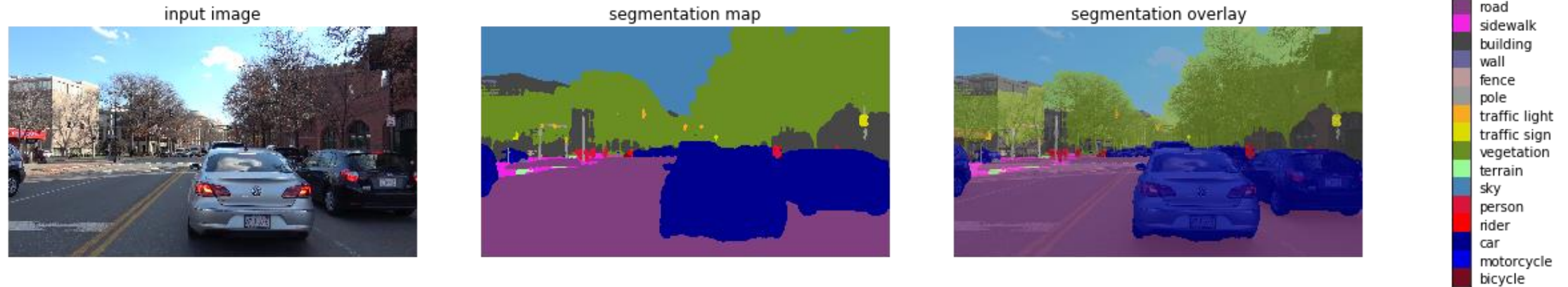


Introduction to TensorFlow

Examples

Image Segmentation

DeepLabv3+ & MobileNetV2 backbone



- ... many more examples could be shown
Natural Language Processing, Style Transfer, Object Detection/Tracking
- All shown examples are available to **you**, without any setup required!

Why TensorFlow?

- Framework for Deep Learning Models
- Fast prototyping for research
- **Not** just a research tool

Robust model deployment in production on any platform



→ see <https://www.tensorflow.org/about/case-studies>

Agenda – Fundamentals

- Development Environments
- Scripting Language
- Automatic Differentiation
- TensorFlow Basics

Development Environments



Jupyter Notebook

- Web IDE with executable documents



Google Colab

- Jupyter notebook stored on Google Drive
- Free Access to GPUs / no setup / runs in the cloud
- Go to <https://colab.research.google.com> (15 GB of free GPU space)



PyCharm

- Provides advanced IDE features

Scripting language



- General purpose language
- Domain specific modules
- Has become the most popular language (for scientific programming)
- Language gets improved continuously

Python

Python as a Programming Language

- Intuitive Scripting Language
- Fully object oriented
- Interpreted language
- Duck typing
- A natural wrapper for C/C++ code
- Main wrapper for several DL frameworks
- Multiple development environments to choose from (PyCharm, Spyder)

Python

Python as an eco system

- Package manager (pip) and Package Index (PyPI)
- NumPy
- SciPy
- matplotlib
- Scikit-learn
- Pandas
- OpenCV-python
- ...



Python Basics

Refresher – see *lecture slides for Python introduction*

- Built-in data types
 - floating point (*float*)
 - integer (*int*)
 - unsigned integer (*uint*)
 - string (*string*)
 - boolean (*bool*)
- Collections
 - *dictionary*
 - *tuple*
 - *list*
 - *set*
- There are no built-in arrays
 - NumPy provides n-d array support
 - Matrix operations implemented in C
 - Callable via python functions

Python Basics

A few remarks

- Python differentiates between
 - Iterables
range, list, tuple – i.e. you can iterate over it
 - Iterators
Agent that performs iteration, e.g. a *for*-loop provides you with an implicit iterator

- *With* context manager

```
with open('my_file.txt', 'r') as file_:
    for line in file_:
        print(line)
```

- Package imports

```
import tensorflow as tf
from tensorflow import keras
import numpy as np
```

Python Basics

Functions and Classes

- Functions

```
def my_func(a, b=0):  
    c = a + b  
    return c
```

- Classes

```
class MyClass(ParentClass):  
    def __init__(self, value, purpose=None):  
        super().__init__() # call parent constructor  
        self.value = value  
        self.purpose = purpose  
  
    def get_purpose(self):  
        return self.purpose  
  
    def store_something(self, x):  
        self.value = x
```

Automatic Differentiation

Backpropagation

- Algorithm to train Deep Learning models
- How to calculate the gradient for each component?

- (two-sided) finite differences

$$\frac{\partial}{\partial x_i} f(x_1, \dots, x_N) = \lim_{h \rightarrow 0} \frac{f(x_1, \dots, x_i + h, \dots, x_N) - f(x_1, \dots, x_i - h, \dots, x_N)}{2h}$$

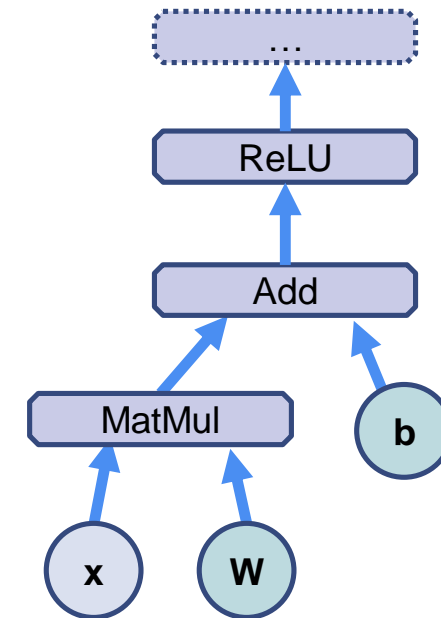
- symbolic differentiation

$$f(x) = \frac{1}{e^{-x} + 1} \rightarrow \frac{df(x)}{dx} = \frac{e^{-x}}{(e^{-x} + 1)^2} = f(x)(1 - f(x))$$

- automatic differentiation:
numerically stable and efficient

Automatic Differentiation

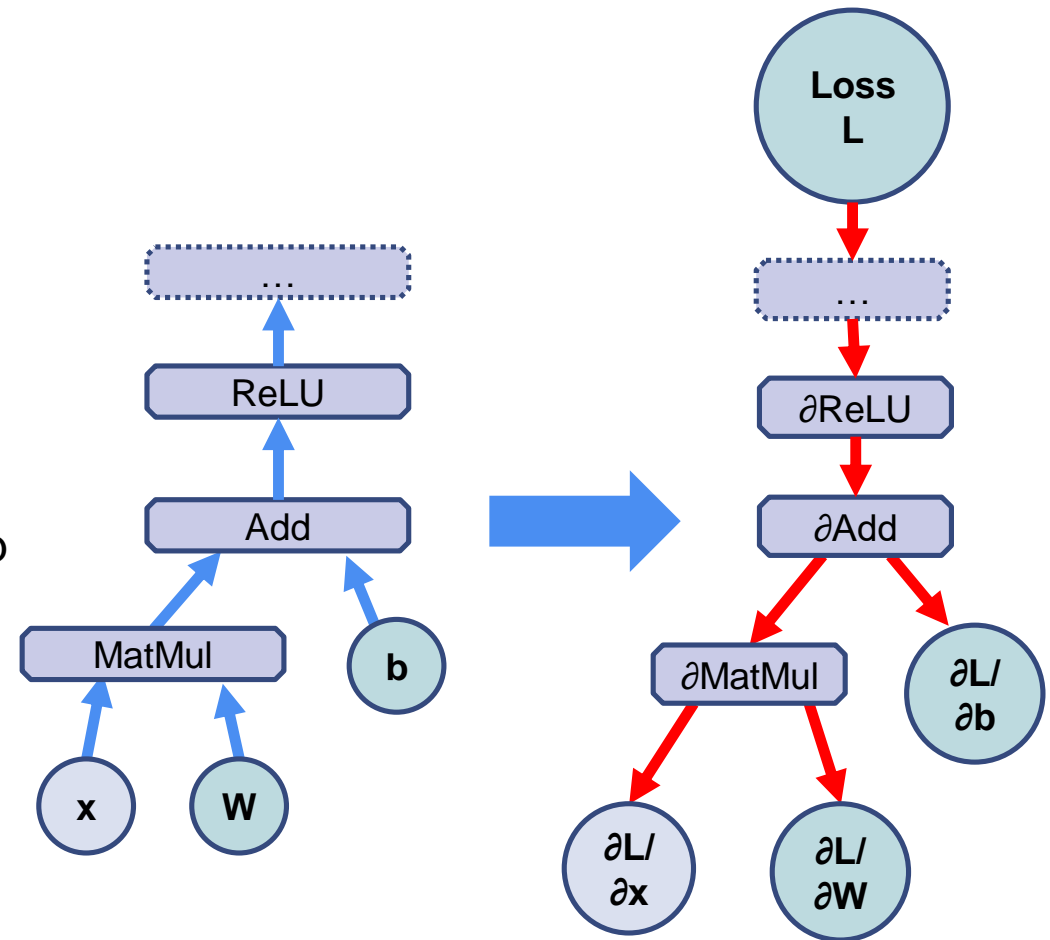
- **Main idea:** Computations are represented as graphs
 - Nodes are the operations (ops)
 - Edges are the data tensors (multidimensional arrays)
- Low-level programming model
- Automatic Differentiation
 - Assembling a computation graph
 - Applying backprop updates
 - Enables complex architectures / No derivatives calculated by hand



Automatic Differentiation

Reverse Mode Automatic Differentiation

- Forward pass
 - Edge values are generated (data tensors)
 - Values are recorded to a **tape**
- Backward Pass
 - Chain rule is applied step by step
 - Intermediate results are used in the next step



Notebook - AutoDiff

Automatic Differentiation

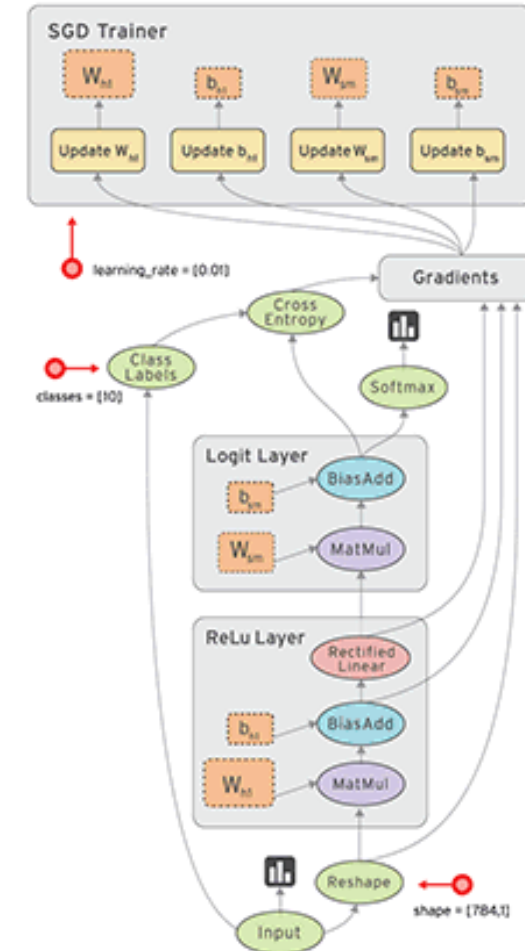
- Environment basics: Jupyter Notebook / Google Colab
- Underlying concept used for backpropagation
- Calculating losses for computational graphs
- Using Autograd

→ **see `intro_nb_autodiff.ipynb`**

TensorFlow Basics

Data flow graphs

- **TensorFlow** provides the API for generation of such graphs
- Typical program consisted of 2 phases (TF 1.0):
 - Construction phase: assembling a graph (model)
 - Execution phase: pushing data through the graph
- Computations defined as a Directed Acyclic Graph (DAG)
 - Graph is defined in high-level language
 - Graph is compiled and optimized



TensorFlow Basics

Data flow graphs are also

- “slow” because of overhead
- difficult to debug
- not “pythonic”

Solution: Eager Execution

- Evaluate operations immediately (PyTorch, TF2.0)
- No separate graph construction and execution

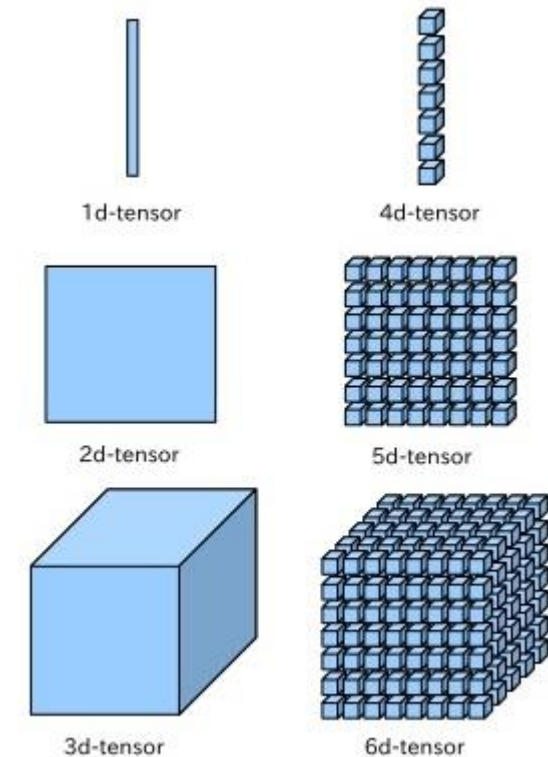
TensorFlow Basics

Tensors

- A tensor may be represented as multidimensional array (n-d array)
- Tensors are basis independent and can be formulated as multilinear map of two vector spaces V^* and V :

$$f : \underbrace{V^* \times \dots V^*}_{p \text{ copies}} \times \underbrace{V \times \dots V}_{q \text{ copies}} \rightarrow \mathbb{R}$$

- For fixed basis, tensors representation as n-d array is sufficient

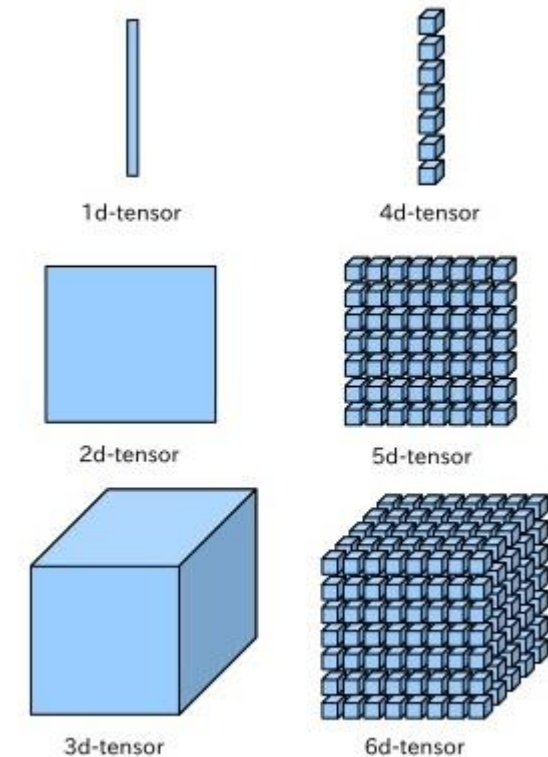


source: <https://blog.knoldus.com/>

TensorFlow Basics

TensorFlow Tensor

- Basic data structure
- Only one data type per tensor!
- *tf.variable* supports similar syntax to NumPy
- Identical data types to NumPy, i.e. *tf.float32* == *np.float32*
- **NumPy Compatibility** – seamless conversion / call *a.numpy()*
- Rank of a tensor:
 - Scalar rank 0 (0th-order tensor)
 - Vector rank 1
 - Matrix rank 2
 - Tensor rank ≥ 3
 - **Rank: order of the tensor == NumPy *ndim***



source: <https://blog.knoldus.com/>

TensorFlow Basics

What is TensorFlow?

C++ backend with functionality callable by Python frontend

- + Automatic Differentiation

- + Data Flow Graphs

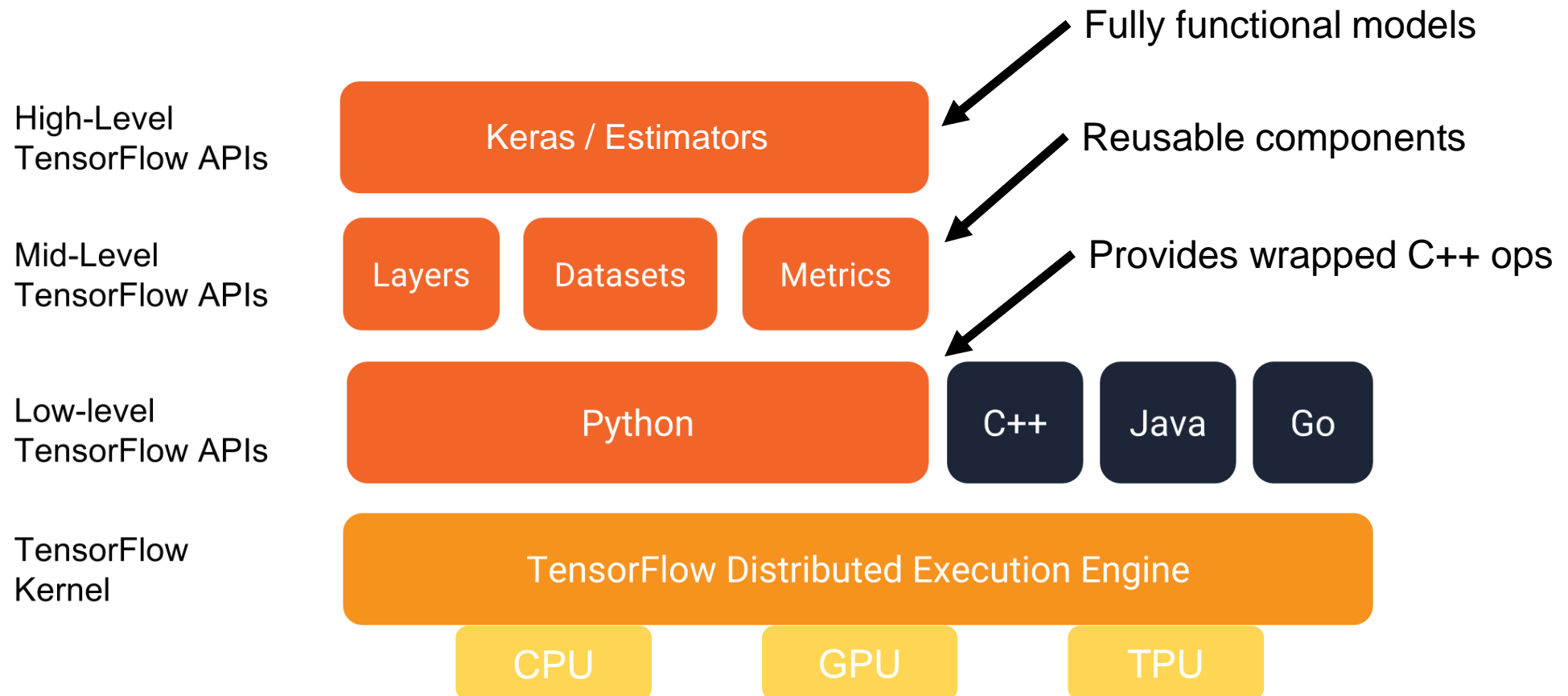
- + DL functionality (Optimizers, Losses, Layers, ...)

- + Utilities (I/O, data pipeline, ...)

- + Portability / Distribution (CPU, GPU, TPU)

TensorFlow Basics

Underlying Structure



Notebook – Machine Learning

Machine Learning example using TensorFlow

- Syntax examples: NumPy vs TensorFlow
- Automatic Differentiation with TensorFlow
- Linear and Polynomial Regression in TensorFlow

→ **see `intro_nb_ml.ipynb`**

TensorFlow API

- TensorFlow website provides you with rich information
- API can be easily navigated and is searchable
- Head over to https://www.tensorflow.org/api_docs/python/tf

The screenshot shows the TensorFlow API documentation website for TensorFlow Core v2.2.0, Python interface. The page has a navigation bar with links for Install, Learn, API, Resources, Community, and Why TensorFlow. A search bar is located in the top right corner. The main content area is titled "TensorFlow Core v2.2.0" and has tabs for Overview, Python, JavaScript, C++, and Java. The Python tab is selected, and the left sidebar shows a list of symbols under "Python v2.2.0", including tf, tf.audio, tf.autodiff, tf.autograph, tf.bitwise, tf.compat, tf.config, tf.data, tf.debugging, tf.distribute, tf.dtypes, tf.errors, tf.estimator, tf.experimental, tf.feature_column, tf.graph_util, tf.image, tf.io, tf.keras, tf.linalg, tf.lite, tf.lookup, tf.math, tf.mixed_precision, tf.mlir, tf.nest, tf.nn, tf.profiler, tf.quantization, tf.queue, and tf.nn. The main content area displays the "Module: tf" page, which includes a "See Stable" button, a "See Nightly" button, and a "View source on GitHub" button. Below this, the "TensorFlow" module is described, and a list of "Modules" is provided, including audio, autodiff, autograph, bitwise, compat, config, and data.

TensorFlow

TensorFlow Core v2.2.0

Overview Python JavaScript C++ Java

Overview

All Symbols

Python v2.2.0

- tf
- tf.audio
- tf.autodiff
- tf.autograph
- tf.bitwise
- tf.compat
- tf.config
- tf.data
- tf.debugging
- tf.distribute
- tf.dtypes
- tf.errors
- tf.estimator
- tf.experimental
- tf.feature_column
- tf.graph_util
- tf.image
- tf.io
- tf.keras
- tf.linalg
- tf.lite
- tf.lookup
- tf.math
- tf.mixed_precision
- tf.mlir
- tf.nest
- tf.nn
- tf.profiler
- tf.quantization
- tf.queue
- tf.nn

TensorFlow

Module: tf

See Stable See Nightly

TensorFlow 1 version View source on GitHub

TensorFlow

pip install tensorflow

Modules

- audio module: Public API for tf.audio namespace.
- autodiff module: Public API for tf.autodiff namespace.
- autograph module: Conversion of plain Python into TensorFlow graph code.
- bitwise module: Operations for manipulating the binary representations of integers.
- compat module: Compatibility functions.
- config module: Public API for tf.config namespace.
- data module: tf.data.Dataset API for input pipelines.