CHAPTER 3

SYSTEM REQUIREMENTS

3.1 FUNCTIONAL REQUIREMENTS

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***USE From Prashanth’s Report and alter the funcyional and non functional requirements as per our need.\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

3.3 SOFTWARE AND HARDWARE USED

**HARDWARE REQUIREMENTS**:

 Operating System: Windows 7 or above

 Processor: Intel i3 or higher

 RAM: 2 GB or more

 Hard Drive: 10 GB or more

 Android 6.0(or above) smartphone

**SOFTWARE REQUIREMENTS:**

 Languages Used: Python,JAVA

 Frameworks : Tensorflow 2.0

Jupyter Notebook

 Android Studio

**Languages/Frameworks Used**

# Python

**History of Python**

Python is an interpreted, high-level, general-purpose programming language created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aims to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Python was conceived in the late 1980s as a successor to the ABC language. Python 2.0, released 2000, introduced features like list comprehensions and a garbage collection system capable of collecting reference cycles. Python 3.0, released 2008, was a major revision of the language that is not completely backward-compatible, and much Python 2 code does not run unmodified on Python 3. Due to concern about the amount of code written for Python 2, support for Python 2.7 (the last release in the 2.x series) was extended to 2020. Language developer Guido van Rossum shouldered sole responsibility for the project until July 2018 but now shares his leadership as a member of a five-person steering council.

Python interpreters are available for many operating systems. A global community of programmers develops and maintains CPython, an open source reference implementation. A non-profit organization, the Python Software Foundation, manages Python and CPython.

Python 3.0, a major, backwards-incompatible release, was released on December 3, 2008 after a long period of testing. Many of its major features have also been backported to the backwards-compatible Python 2.6 and 2.7.

In February 1991, van Rossum published the code (labelled version 0.9.0) to alt.sources. Already present at this stage in development were classes with inheritance, exception handling, functions, and the core datatypes of list, dict, str and so on. Also in this initial release was a module system borrowed from Modula-3; Van Rossum describes the module as "one of Python's major programming units". Python's exception model also resembles Modula-3's, with the addition of an else clause. In 1994 comp.lang.python, the primary discussion forum for Python, was formed, marking a milestone in the growth of Python's userbase.

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by metaprogramming and metaobjects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming.

The latest version of Python is 3.8.3; however the latest stable release of Python is version 3.8.2, released on 24 February 2020.

# TensorFlow 2.0

TensorFlow started as an open-source [deep learning](https://courses.analyticsvidhya.com/courses/fundamentals-of-deep-learning?utm_source=blog&utm_source=tensorflow-2-tutorial-deep-learning) library and has today evolved into an end to end machine learning platform that includes tools, libraries and resources for the research community to push the state of the art in deep learning and developers in the industry to build ML & DL powered applications.

TensorFlow had its first public release back in 2015 by the Google Brain team. At the time, the evolving deep learning landscape for developers & researchers was occupied by Caffe and Theano. In a short time, TensorFlow emerged as the most popular library for deep learning.

TensorFlow is fast with backend written in C++ and has interfaces in Python, Java, Swift, and Android.

TensorFlow 2.0 makes development of ML applications much easier. With tight integration of Keras into TensorFlow, eager execution by default, and Pythonic function execution.

The standardized [SavedModel](https://www.tensorflow.org/guide/saved_model) file format can be used to run models on a variety of runtimes, including the cloud, web, browser, Node.js, mobile and embedded systems.

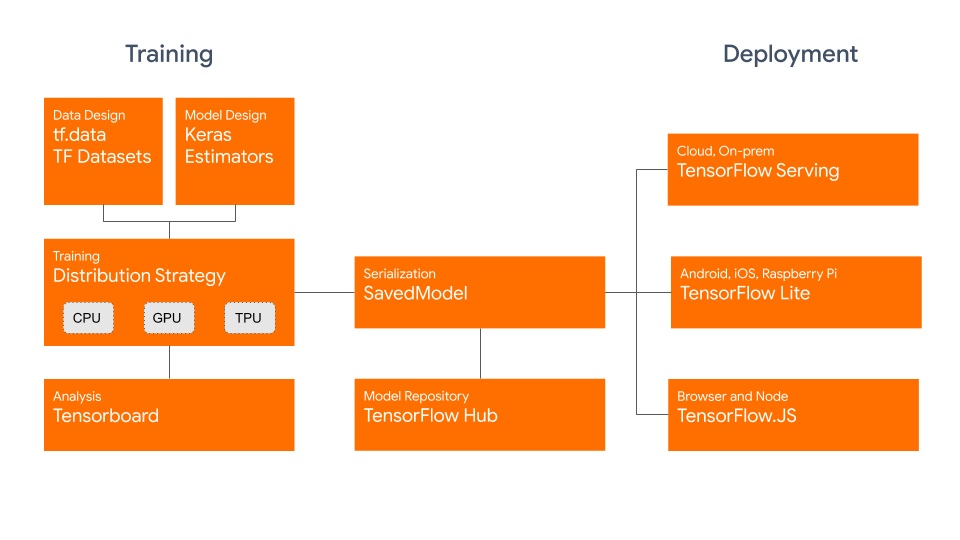


Fig:Tensorflow 2.0 Architecture

## Keras

tf.keras is TensorFlow's implementation of the Keras API specification. This is a high-level API to build and train models that includes first-class support for TensorFlow-specific functionality, such as eager execution, data pipelines, and Estimators. keras makes TensorFlow easier to use without sacrificing flexibility and performance.

In Keras, you assemble layers to build models. A model is (usually) a graph of layers. The most common type of model is a stack of layers: the [tf.keras.Sequential](https://www.tensorflow.org/api_docs/python/tf/keras/Sequential) model.

## TensorFlow Lite

TensorFlow Lite is a set of tools to help developers run TensorFlow models on mobile, embedded, and IoT devices. It enables on-device machine learning inference with low latency and a small binary size.

TensorFlow Lite consists of two main components:

* The **TensorFlow Lite interpreter** , which runs specially optimized models on many different hardware types, including mobile phones, embedded Linux devices, and microcontrollers.
* The **TensorFlow Lite converter** , which converts TensorFlow models into an efficient form for use by the interpreter, and can introduce optimizations to improve binary size and performance.

TensorFlow Lite is designed to make it easy to perform machine learning on devices, "at the edge" of the network, instead of sending data back and forth from a server. For developers, performing machine learning on-device can help improve:

* *Latency:* there's no round-trip to a server
* *Privacy:* no data needs to leave the device
* *Connectivity:* an Internet connection isn't required
* *Power consumption:* network connections are power hungry

TensorFlow Lite works with a huge range of devices, from tiny microcontrollers to powerful mobile phones.

Key features

* [*Interpreter*](https://www.tensorflow.org/lite/guide/inference) tuned for on-device ML, supporting a set of core operators that are optimized for on-device applications, and with a small binary size.
* Diverse platform support, covering [Android](https://www.tensorflow.org/lite/guide/android) and [iOS](https://www.tensorflow.org/lite/guide/ios) devices, embedded Linux, and microcontrollers, making use of platform APIs for accelerated inference.
* APIs for multiple languages including Java, Swift, Objective-C, C++, and Python.
* High performance, with [hardware acceleration](https://www.tensorflow.org/lite/performance/gpu) on supported devices, device-optimized kernels, and [pre-fused activations and biases](https://www.tensorflow.org/lite/guide/ops_compatibility).
* Model optimization tools, including [quantization](https://www.tensorflow.org/lite/performance/post_training_quantization), that can reduce size and increase performance of models without sacrificing accuracy.
* Efficient model format, using a [FlatBuffer](https://www.tensorflow.org/lite/convert/index) that is optimized for small size and portability.
* [*Pre-trained models*](https://www.tensorflow.org/lite/models) for common machine learning tasks that can be customized to your application.

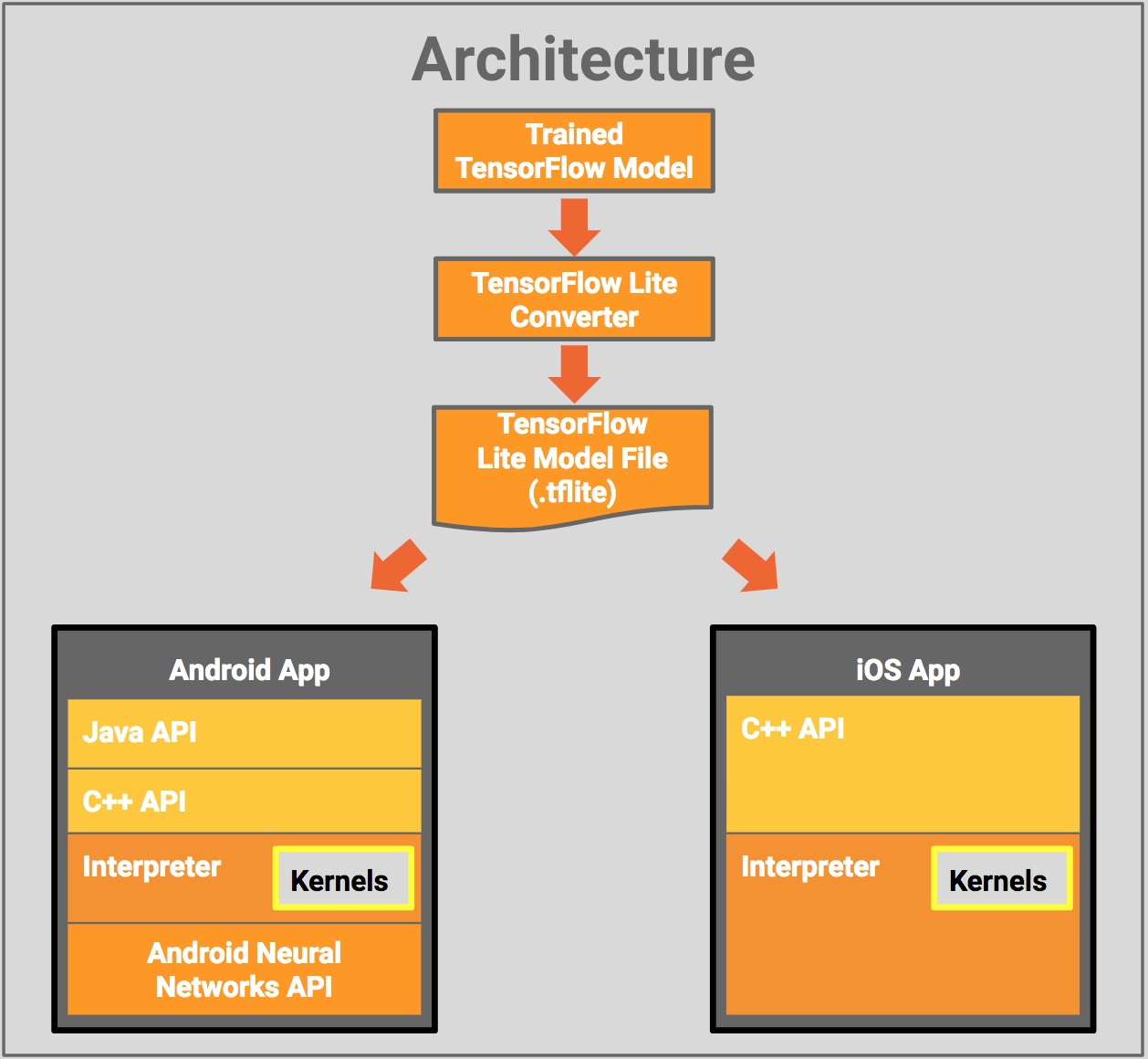


Fig: Tensorflow Lite Architecture