



# TensorFlow

师文轩

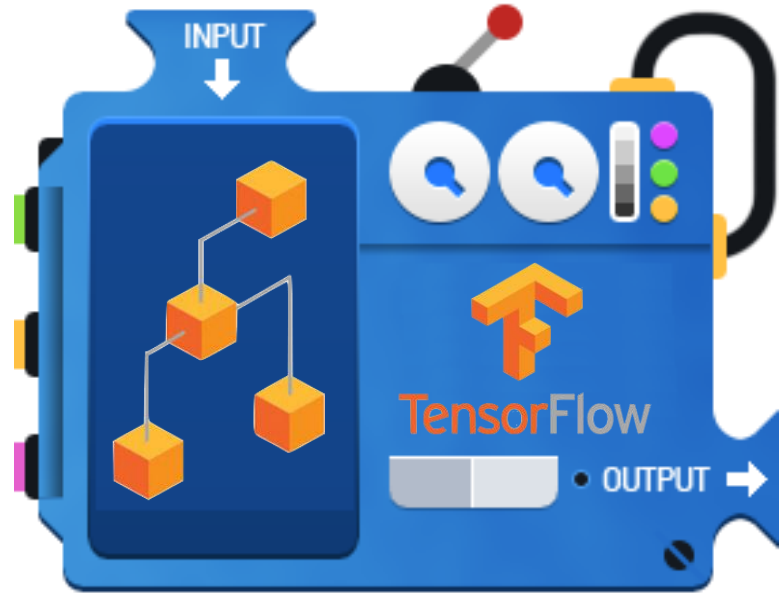
南开大学 软件学院



# TensorFlow Mechanics

2 feed data and run graph (operation)  
`sess.run (op, feed_dict={x: x_data})`

1 Build graph using  
TensorFlow operations



3 update variables  
in the graph  
(and return values)

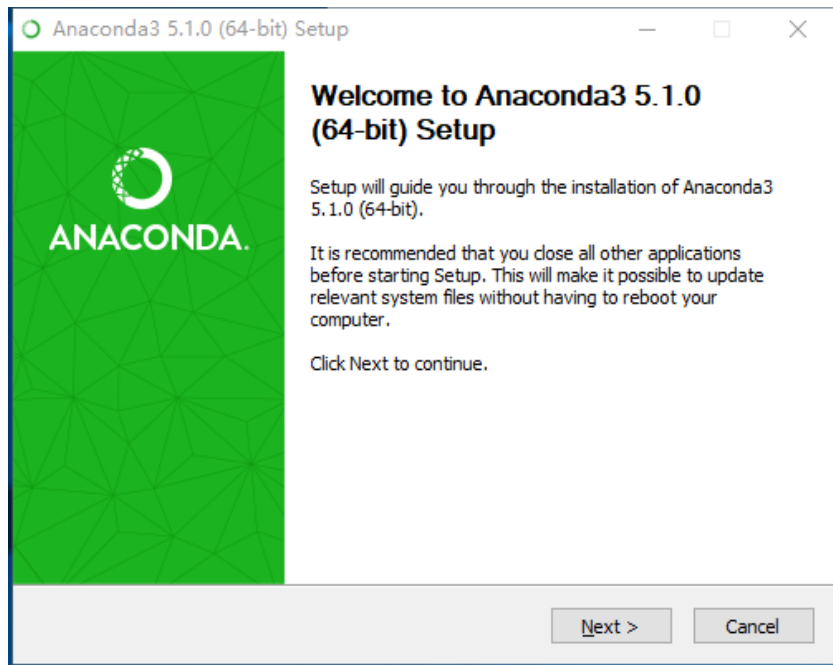
# TensorFlow 在许多不同领域的应用

- 天体物理学家使用 **TF** 分析开普勒任务中的大量数据，以发现新的行星。
- 医学研究人员利用 **TF** 机器学习技术来评估一个人心脏病发作和中风的几率。
- 空中交通管制员用 **TF** 来预测飞机最有可能行经的路线，以确保飞机安全着陆。
- 工程师使用 **TF** 分析热带雨林中的声音数据，以检测伐木车和其他非法活动。
- 科学家在非洲用 **TF** 检测木薯植物疾病，从而提高产量并帮助更好地满足非洲大陆的粮食需求。

# Windows环境下安装配置 CPU版的Tensorflow

## ●安装Anaconda

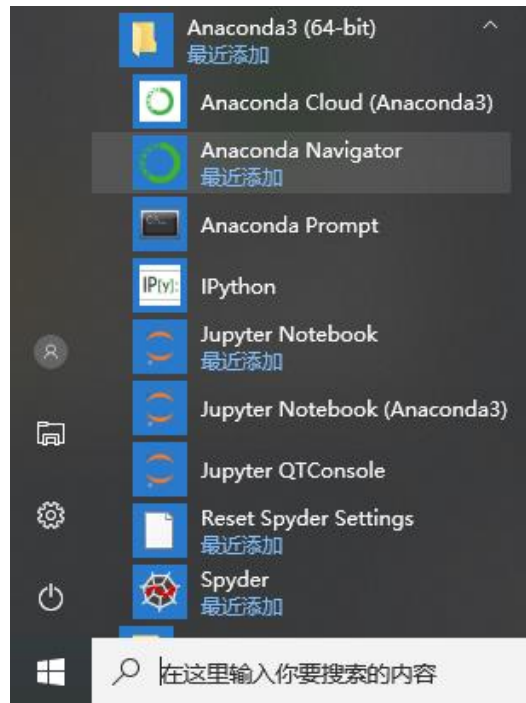
- anaconda是一个开源的[Python](#)发行版本，其包含了conda、Python等180多个科学包及其依赖项。



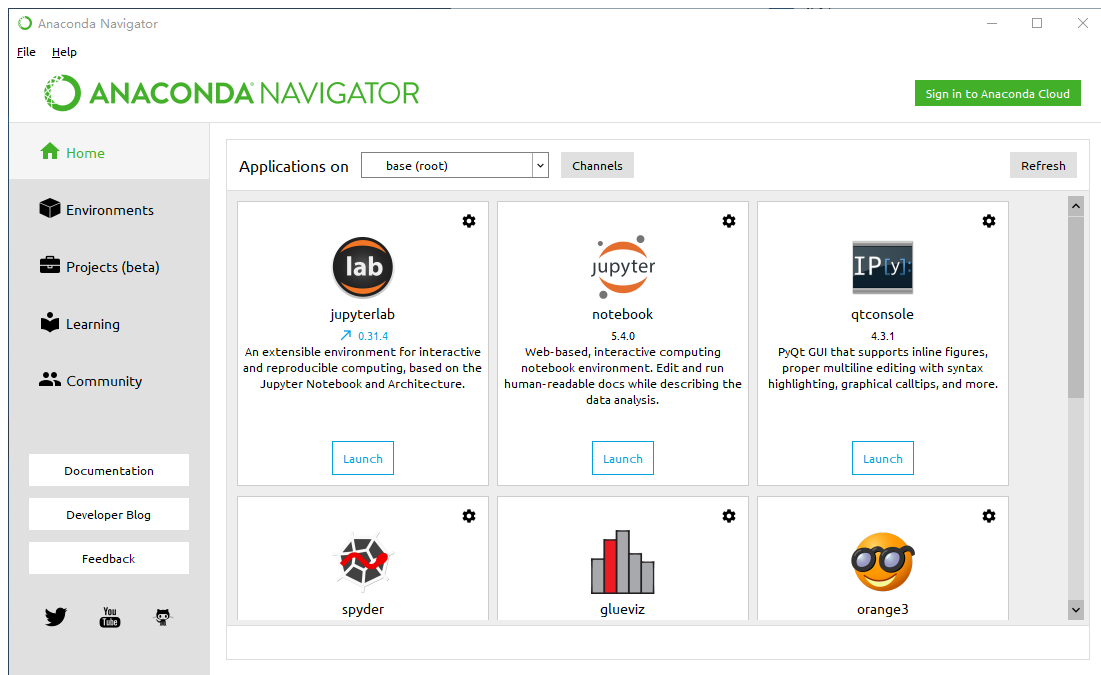
# Windows环境下安装配置 CPU版的Tensorflow

- 为Tensorflow配置独立环境

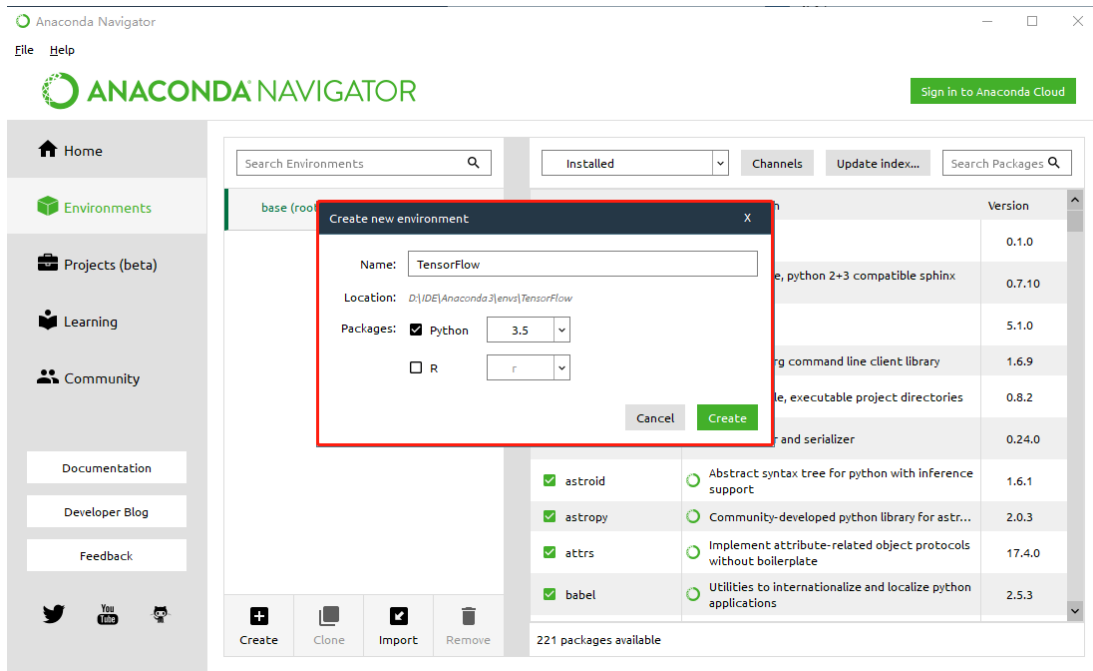
- 为什么在安装Tensorflow之前我们要先安装Anaconda呢？
- 因为目前python的版本非常多，各种辅助计算与富含多种功能的库也很多，其中不同的库可能会依赖于不同版本的python，不同的python库也有可能会互相影响。
- 所以推荐为Tensorflow配置一个独属于自己的环境。



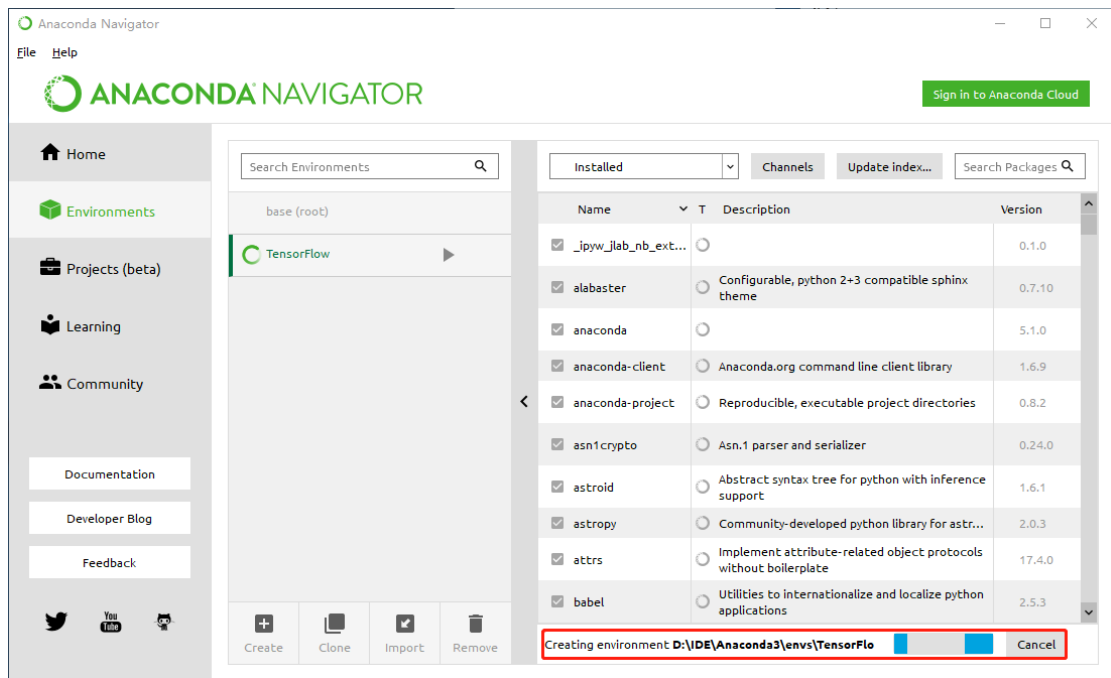
# Windows环境下安装配置 CPU版的Tensorflow



# Windows环境下安装配置CPU版的Tensorflow



# Windows环境下安装配置CPU版的Tensorflow



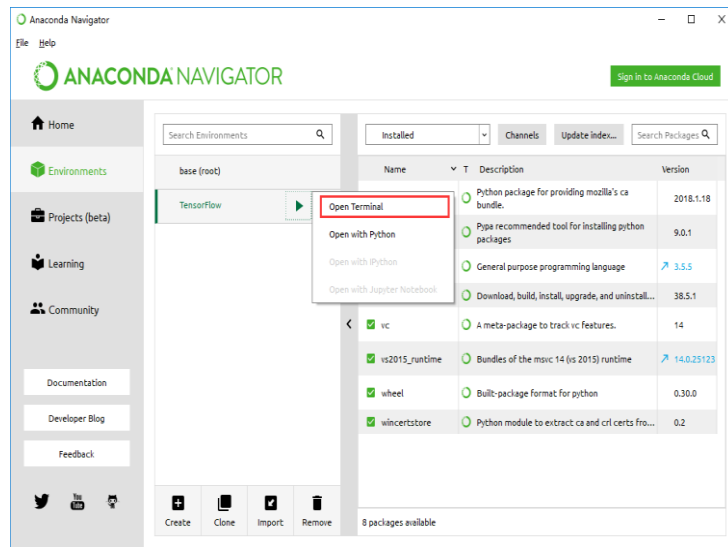


# Windows环境下安装配置CPU版的Tensorflow

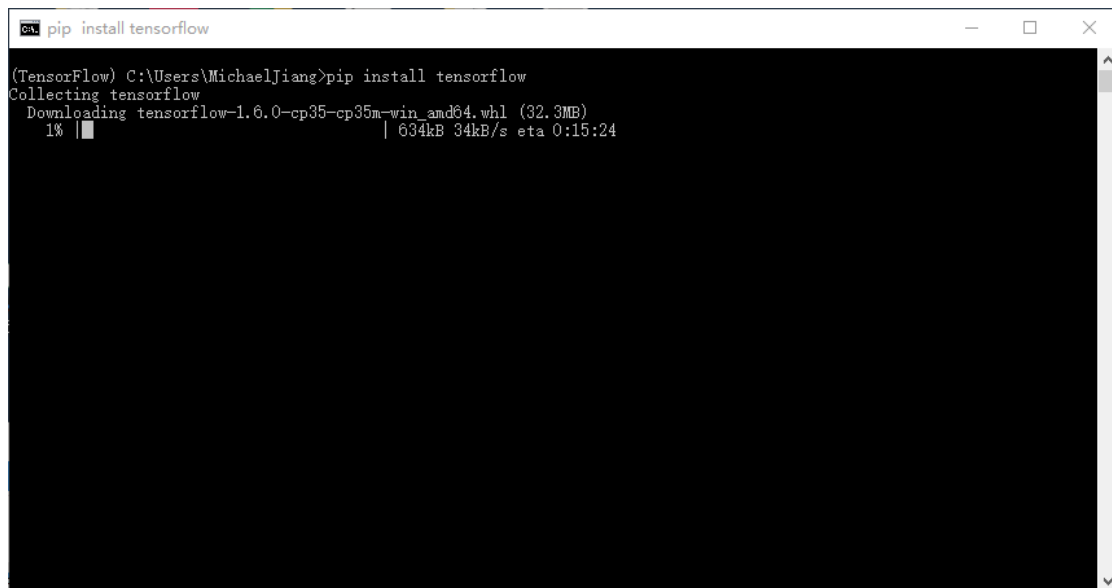
## ● 正式安装Tensorflow

-打开**Anaconda Navigator**， 找到**Environment**中的**Tensorflow**， 点击**Open Terminal**后输入以下命令

```
-pip install tensorflow
```



# Windows环境下安装配置 CPU版的Tensorflow



```
CA pip install tensorflow

(TensorFlow) C:\Users\MichaelJiang>pip install tensorflow
Collecting tensorflow
  Downloading tensorflow-1.6.0-cp35-cp35m-win_amd64.whl (32.3MB)
    1% |█| 634kB 34kB/s eta 0:15:24
```

# Windows环境下安装配置 CPU版的Tensorflow

## ●验证Tensorflow环境

-在刚才打开的Terminal中输入python，并输

入以下代码

```
import tensorflow as tf
hello = tf.constant('Hello, TensorFlow!')
sess = tf.Session()
print(sess.run(hello))
```

-如果看到Hello, TensorFlow! 则证明安装成功

```
Python 3.5.4 |Continuum Analytics, Inc. | (default
) on win32
Type "help", "copyright", "credits" or "license
(>>> import tensorflow as tf
(>>> hello = tf.constant('Hello, TensorFlow!')
(>>> sess = tf.Session()
2018-04-14 11:12:56.496917: I T:\src\github\ten
:140] Your CPU supports instructions that this
(>>> print(sess.run(hello))
b'Hello, TensorFlow!'
(>>>
```

# 动态图机制Eager Execution

- TensorFlow 引入了“Eager Execution”

- 开启Eager模式后，TensorFlow会从原先的声明式（declarative）编程形式变成命令式（imperative）编程形式。

- 当写下语句"`c = tf.matmul(a, b)`"后（以及其他任何tf开头的函数），就会直接执行相应的操作并得到值，而不再像之前那样，生成一个Tensor，通过`sess.run()`才能拿到值。

# 动态图机制Eager Execution

- 安装**Eager**模式

- pip install tf-nightly

- 开启**Eager**模式（开启就不能被关闭）

- import** tensorflow as tf

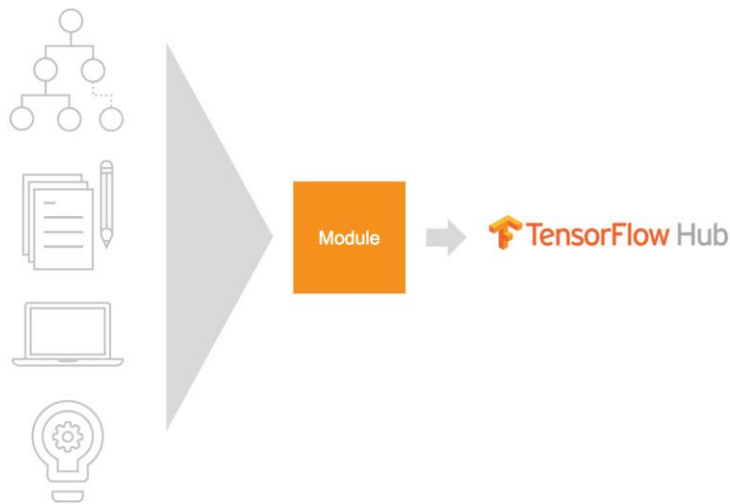
- import** tensorflow.contrib.eager as tfe

- tfe.enable\_eager\_execution()

# TensorFlow Hub

## •TensorFlow Hub:重用机器学习模块的库

- 旨在促进模型的可重复使用模块的发布、发现和使用。
- 这些模块是一块块独立的 TensorFlow 计算图，可以在不同的任务中重复使用。



# TensorFlow Hub

- 例子：图像再训练

- 现代图像识别模型具有数百万个参数，从头开始进行培训需要大量的标记数据和计算能力；
- 使用称为图像重新训练的技术，可以使用少得多的数据来训练模型，并且计算时间少得多。

```
# Download and use NASNet feature vector module.
```

```
module = hub.Module(  
    "https://tfhub.dev/google/imagenet/nasnet_large/feature_vector/1")
```

```
features = module(my_images)
```

```
logits = tf.layers.dense(features, NUM_CLASSES)
```

```
probabilities = tf.nn.softmax(logits)
```

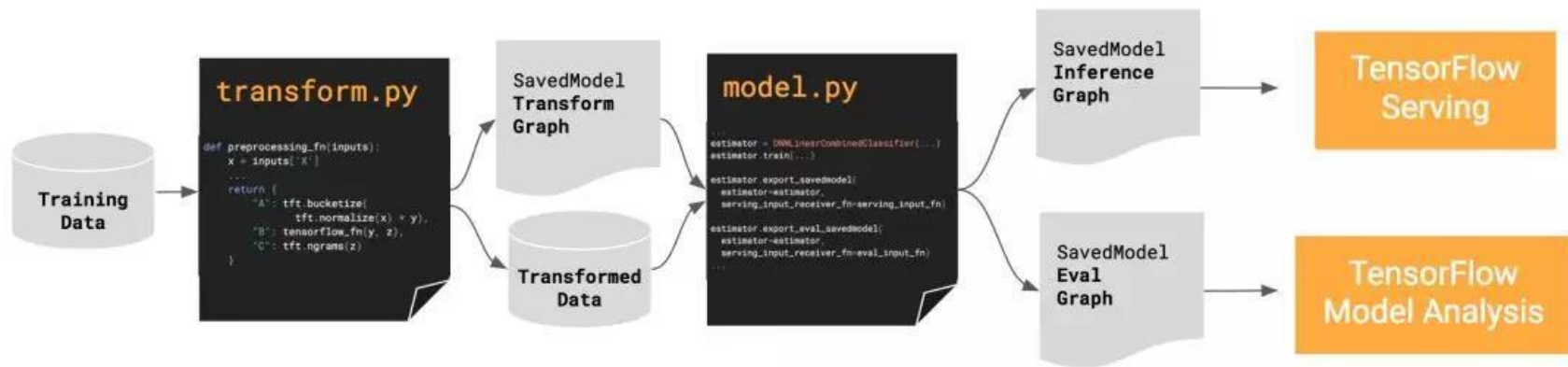
# TensorFlow Hub

- 上述例子的基本思想是重用现有的图像识别模块从图像中提取特征，然后在这些特征之上训练一个新的分类器。
  - TensorFlow Hub模块可以在构建TensorFlow图形时从URL（或者从文件系统路径）实例化。
  - TensorFlow Hub上有各种模块供选择，包括各种类型的NASNet， MobileNet， Inception， ResNet等。



# TensorFlow Extended (TFX)

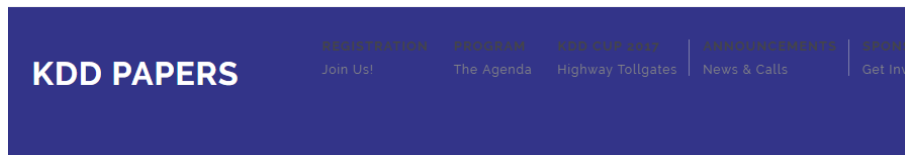
- TFX是一个机器学习平台，可让开发者准备数据、训练、验证并把训练好的模型快速部署在生产环境提供可用的服务。



# TensorFlow Extended (TFX)

- **TFX**来自KDD 2017的一篇文章，作者是一堆Googlers。

- 文章的意义并不在于手把手教你如何搭建一套**TFX**，而是阐释了搭建一套**TFX**系统所需要包含的主要组件，以及这些组件在实现时需要考虑哪些关键点，还有作者团队在这其中积累的经验教训。



## TFX: A TensorFlow-Based Production-Scale Machine Learning Platform

Denis Baylor (Google Inc.);Eric Breck (Google Inc.);Heng-Tze Cheng (Google Inc.);Noah Fiedel (Google Inc.);Chuan Yu Foo (Google Inc.);Zakaria Haque (Google Inc.);Salem Haykal (Google Inc.);Mustafa Ispir (Google Inc.);Vihan Jain (Google Inc.);Levent Koc (Google Inc.);Chiu Yuen Koo (Google Inc.);Lukasz Lew (Google Inc.);Clemens Mewald (Google Inc.);Akshay Modi (Google Inc.);Neoklis Polyzotis (Google Inc.);Sukriti Ramesh (Google Inc.);Sudip Roy (Google Inc.);Steven Whang (Google Inc.);Martin Wicke (Google Inc.);Jarek Wilkiewicz (Google Inc.);Xin Zhang (Google Inc.);Martin Zinkevich (Google Inc.)

### Abstract

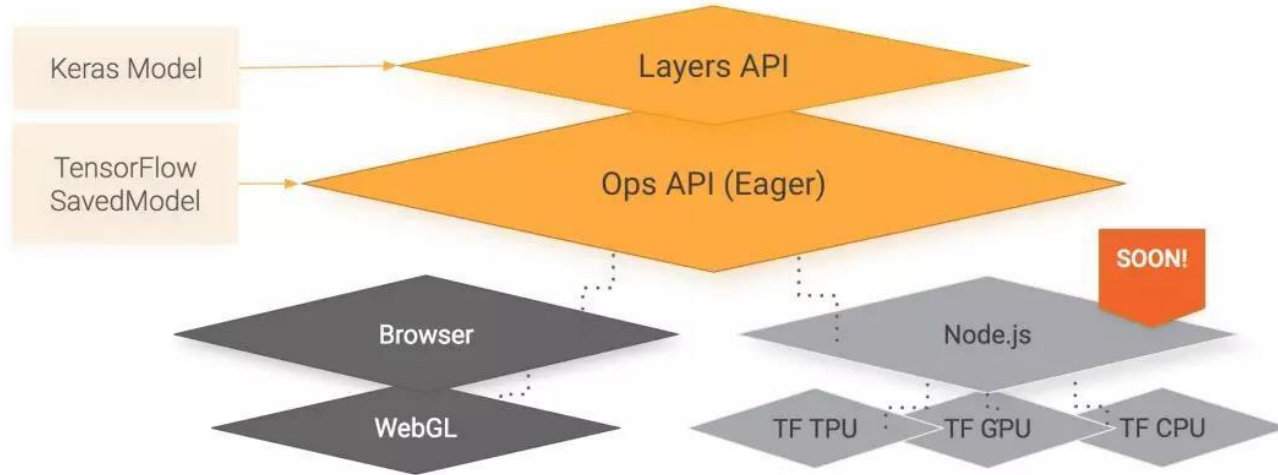
Creating and maintaining a platform for reliably producing and deploying machine learning models requires careful orchestration of many components—a learner for generating models based on training data, modules for analyzing and validating both data as well as models, and finally infrastructure for serving models in production. This becomes particularly challenging when data changes over time and fresh models need to be produced continuously. Unfortunately, such orchestration is often done ad hoc using glue code and custom scripts developed by individual teams for specific use cases, leading to duplicated effort and fragile systems with high technical debt. We present the anatomy of a general-purpose machine learning platform and one implementation of such a platform at Google. By integrating the aforementioned

# TensorFlow Extended (TFX)

- TFX的核心设计原则

- 构建可服务于多个学习任务的统一平台：这要求系统具有足够的通用性和可扩展性。
- 支持持续训练和服务：这两个事情看起来简单，但是如果考虑到其中的风险控制和自动化问题发现等细节的话，也并不简单。
- 人工干预：如何优雅地让人参与整个流程，解决机器不好解决的问题，也是一个挑战。
- 可靠性和稳定性：这里的可靠性和稳定性不只指的是服务不崩溃这个级别，还包括在数据层面发生问题的时候服务的效果依然可以保持稳定可靠。

# TensorFlow.js



# TensorFlow.js

- **TensorFlow.js** 是给 JavaScript 开发者的一个新的机器学习框架，它可以完全在浏览器里定义和训练模型；
- **TensorFlow.js** 还可以导入离线训练的 TensorFlow 和 Keras 模型进行预测，并可以对 WebGL 实现无缝支持。